ΕΘΝΙΚΌ ΜΕΤΣΌΒΙΟ ΠΟΛΥΤΕΧΝΕΊΟ
ΣΧΟΛΉ ΗΛΕΚΤΡΟΛΌΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΏΝ
ΤΟΜΈΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΉΣ ΚΑΙ ΣΥΣΤΗΜΆΤΩΝ ΠΛΗΡΟΦΟΡΙΚΉΣ

Μηχανή εκτέλεσης ροών εργασίας για συστήματα χαμηλών προδιαγραφών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ
του

Νικολάου-Σταύρου Ζευγολάτη

Επιβλέπων : Θεοδώρα, Α., Βαρβαρίγου
Καθηγήτρια

Αθήνα, Νοέμβριος 2009

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Επικοινωνιών, Ηλεκτρονικής και Συστημάτων
Πληροφορικής

Μηχανή εκτέλεσης ροών εργασίας για συστήματα χαμηλών προδιαγραφών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ
του

Νικολάου-Σταύρου Ζευγολάτη

Επιβλέπων : Θεοδώρα, Α., Βαρβαρίγου
Καθηγήτρια

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 3η Νοεμβρίου 2009.

.....	.....	.....
Θεοδώρα Βαρβαρίγου	Παπαβασιλείου Συμεών	Βασίλειος Λούμος
Καθηγήτρια	Επίκουρος Καθηγητής	Καθηγητής

Αθήνα, Νοέμβριος 2009

.....
Νικόλας-Σταύρος, Α., Ζευγολάτης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Πρόλογος

Η παρούσα διπλωματική εργασία εκπονήθηκε κατά τη διάρκεια του εαρινού εξαμήνου του ακαδημαϊκού έτους 2008-2009 από τον Ζευγολάτη Νικόλαο-Σταύρο.

Θα ήθελα να ευχαριστήσω την καθηγήτρια κα Θεοδώρα Βαρβαρίγου για την ανάθεση της εργασίας, την παρακολούθηση και παροχή κατευθύνσεων για την υλοποίηση της, την εξασφάλιση ιδανικών συνθηκών εργασίας και μελέτης καθώς και για την ευκαιρία που μου έδωσε να ασχοληθώ με ένα άκρως ενδιαφέρον κεφάλαιο του σύγχρονου αλλά κατά βάση και μελλοντικού διαδικτύου.

Επίσης, θα ήθελα να ευχαριστήσω τον διδακτορικό φοιτητή του εργαστηρίου Τηλεπικοινωνιών, κο Σπύρο Γωγουβίτη για την άψογη συνεργασία που είχαμε όλο αυτό το διάστημα και τον τρόπο που με καθοδήγησε σε ένα ομολογουμένως αχανές γνωστικό αντικείμενο και που με αμέριστη υπομονή και συμπαράσταση, συνέβαλε με τον τρόπο του στην ολοκλήρωση αυτού του έργου.

Η συμβολή των δύο ανωτέρω ήταν καθοριστική τόσο στην περίοδο έρευνας του αντικειμένου της εργασίας όσο και κατά τη διάρκεια της συγγραφής της.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου, καθώς δίχως την συνεχή συμπαράστασή της όλα αυτά τα χρόνια και τις πολλές θυσίες που έχει κάνει για την ολοκλήρωση των σπουδών μου, όλα αυτά θα ήταν αδύνατα.

Νοέμβριος 2009

Ζευγολάτης Νικόλαος-Σταύρος

Περίληψη

Σκοπός της παρούσας διπλωματικής εργασίας είναι να μελετηθεί η ανάπτυξη της τεχνολογίας των δικτυακών υπηρεσιών καθώς και των τεχνολογιών που χρησιμοποιούνται ή βρίσκονται υπό ανάπτυξη προκειμένου να ενταχθούν οι δικτυακές υπηρεσίες σε αυτόνομες ροές εκτέλεσης ή επιχειρησιακές διαδικασίες. Η υλοποίηση είναι ανοικτού λογισμικού και αφορά την δημιουργία μίας μηχανής όπου δέχεται ένα αρχείο ροής εργασίας μεταξύ διαφορετικών δικτυακών υπηρεσιών σε γλώσσα bpel, το αναλύει και στην συνέχεια το υλοποιεί χρησιμοποιώντας σύγχρονη επικοινωνία μεταξύ των υπηρεσιών. Στόχος της εργασίας ήταν η υπηρεσία αυτή να απαιτεί λίγους υπολογιστικούς πόρους και να είναι γρήγορη, γεγονός που επετεύχθη.

Λέξεις κλειδιά:

WS-BPEL, WSDL, δικτυακές υπηρεσίες, Ροή εκτέλεσης δικτυακών υπηρεσιών, επιχειρησιακές διαδικασίες, πλέγμα, σύννεφο

Abstract

The purpose of this thesis is to study the design technology of web services and as well as as the technology that is used or is under development to integrate web services into independent streams of execution or operational procedures. The implementation is open source and it involves the creation of an engine which accepts a workflow file between different web services in bpel language, analyzes it and then it implements it, using synchronous communication. The aim of this study was to insure that the service requires few computing resources and is fast, a goal which that was reached.

Keywords :

Web services, Business processes, WS-BPEL, execution flows, workflows, bpel enactor, grid, cloud, computing, synchronous communication

ΕΙΣΑΓΩΓΗ

Τις τελευταίες δεκαετίες έχει παρατηρηθεί μια τεράστια εξάπλωση του διαδικτύου (Internet) σε διάφορους τομείς της ανθρώπινης δραστηριότητας. Αρχικά επρόκειτο για μεταφορά απλών html σελίδων και βασικών υπηρεσιών όπως email και ftp. Με την πάροδο, όμως, του χρόνου σημειώθηκε ραγδαία ανάπτυξη και προσθήκη όλο και πιο πολύπλοκων διαδικασιών.

Το διαδίκτυο αποτελεί στην ουσία του, ένα δίκτυο διασυνδεδεμένων υπολογιστών, οι οποίοι αλληλεπιδρούν και παρέχουν σημαντικές πληροφορίες και υπηρεσίες, ένα δίκτυο υπολογιστικών συστημάτων πολλών διαφορετικών κατασκευαστών, διαφορετικών αρχιτεκτονικών και διαφορετικών λειτουργικών συστημάτων. Στόχος και σκοπός του συγκεκριμένου δικτύου είναι να μπορέσει να προσφέρει υπηρεσίες: υπηρεσίες ενημέρωσης, υπηρεσίες διεκπεραίωσης, υπηρεσίες ψυχαγωγίας και υπηρεσίες διασύνδεσης. Σε αυτό το δίκτυο που προσφέρει υπηρεσίες, εταιρείες και ιδιώτες προσπαθούν να διαχειριστούν τις υπηρεσίες με έναν κατανεμημένο τρόπο, ανεξάρτητα της αρχιτεκτονικής των υπολογιστών, σε μία γλώσσα κοινή σε όλους.

Μία γλώσσα που προσπαθεί να λύσει το παραπάνω ζήτημα τα τελευταία χρονιά είναι η WS-BPEL(Business Process Execution Language – Web Services). Με δομή(structure) απλή, καλά ορισμένη και ανεξάρτητη πλατφόρμας έχει αρχίσει να κυριαρχεί στο διαδίκτυο, ενώ συνεχώς επεκτείνεται και βελτιώνεται. Η WS-BPEL, όπως μαρτυρά και το όνομά της, αποτελεί μία "γλώσσα" που διαχειρίζεται διαγράμματα ροής διεργασιών προς εκτέλεση. Συνδυάζει απλότητα με δύναμη, ταχύτητα με αποτελεσματικότητα. Άμεση συνέπεια τούτου είναι να προσελκύει ολοένα και περισσότερο το ενδιαφέρον όλο και περισσότερων προγραμματιστών αλλά και μεγάλων εταιρειών του χώρου που επιθυμούν να την υιοθετήσουν ή να την υποστηρίξουν.

Πίνακας περιεχομένων

1. Web services

1.1 Τί είναι τα web services(δικτυακές υπηρεσίες)

Ο καθαρός ορισμός μιας υπηρεσίας διαδικτύου (web service) σύμφωνα με το World Wide Web Consortium είναι "ένα σύστημα λογισμικού σχεδιασμένο για να υποστηρίζει διαλειτουργική αλληλεπίδραση από μηχανή-σε-μηχανή μέσω δικτύου. Έχει μια διεπαφή που περιγράφεται από μία εύκολα επεξεργάσιμη μορφή από μηχανή (συγκεκριμένα την WSDL). Άλλα συστήματα αλληλεπιδρούν με την υπηρεσία διαδικτύου σε μια διαδικασία που προβλέπεται από την περιγραφή της, χρησιμοποιώντας SOAP μηνυμάτα, που συνήθως μεταφέρονται χρησιμοποιώντας το πρωτόκολλο HTTP με σειριοποίηση ενός XML σε συνδυασμό με άλλα πρότυπα που σχετίζονται με το διαδίκτυο"[1].

Ο όρος web services, όπως υπονοεί το όνομά του, αναφέρεται σε υπηρεσίες προσβάσιμες μέσω του διαδικτύου. Στον όρο, όμως, εμπεριέχονται πολύ περισσότερες έννοιες απ' ότι υπονοεί η ονομασία του. Με την έννοια web services αναφερόμαστε και στην αρχιτεκτονική, τα πρότυπα, την τεχνολογία και τα επιχειρησιακά μοντέλα που απαιτούνται για να υπάρξουν και να υλοποιηθούν οι web services. Ακολουθεί ένας επιπλέον ορισμός των web services, όπως αυτός δίνεται από την IBM: "Οι web services αποτελούν την νέα γενιά δικτυακών εφαρμογών. Είναι αυτοπεριεχόμενες, αυτοπεριγραφούμενες εφαρμογές, οι οποίες δημοσιεύονται και τοποθετούνται αλλά και καλούνται μέσω του διαδικτύου. Οι web services υλοποιούν υπηρεσίες που μπορεί να είναι από απλές μέχρι σύνθετες επιχειρησιακές διαδικασίες".

Πιο απλά, μια web service είναι μια υπηρεσία διαθέσιμη στο διαδίκτυο, η οποία χρησιμοποιεί ένα καθορισμένο σύστημα μηνυμάτων βασισμένα σε XML και δεν είναι δεμένη με κανένα λειτουργικό σύστημα ή γλώσσα προγραμματισμού.

Από τα παραπάνω προκύπτει ότι θέλουμε μια web service να παρουσιάζει δύο χαρακτηριστικά:

1 http://www.w3.org/TR/ws-gloss/

- Μια web service πρέπει να είναι αυτοπεριγραφούμενη. Όταν δημοσιεύεται μια web service πρέπει να δημοσιεύεται επίσης και μια διεπαφή (interface) προς αυτήν, τουτέστιν μια περιγραφή αυτής έτσι, ώστε σε ανάπτυξη ολοκληρωμένων συστημάτων να μπορεί να ενοποιηθεί μαζί με άλλες και να είναι εύκολα υλοποιήσιμη. Αυτό γίνεται μέσω του αρχείου περιγραφής της υπηρεσίας, WSDL, για το οποίο θα μιλήσουμε παρακάτω.

- Μια web service πρέπει να είναι ανακαλύψιμη. Όταν δημιουργείται μια web service τότε πρέπει να γνωστοποιείται αυτό το γεγονός. Πρέπει, δηλαδή, να δημοσιεύεται ένα interface αυτής ώστε να είναι σε θέση οι ενδιαφερόμενοι να την βρουν και να την καλέσουν.

- Είναι αυτόνομα μέρη, τα οποία επεξεργάζονται καλώς ορισμένα XML αρχεία

Συνοπτικά μια ολοκληρωμένη web service είναι μια υπηρεσία η οποία:

1. Είναι διαθέσιμη στο διαδίκτυο ή σε ιδιωτικά-εταιρικά δίκτυα
2. Χρησιμοποιεί ένα καθορισμένο σύστημα μηνυμάτων βασισμένο σε XML
3. Δεν είναι δεμένη με κανένα λειτουργικό σύστημα ή γλώσσα προγραμματισμού
4. Είναι αυτό-περιγραφόμενη μέσω κοινής XML γραμματικής
5. Είναι ανακαλύψιμη μέσω μιας απλής διαδικασίας εύρεσης (find)

1.2 Η τεχνολογία των web services

Για πιο εύκολη κατανόηση και διαχείριση των web services, έχουν κατηγοριοποιηθεί σε ένα μοντέλο επιπέδων. Η μοντελοποίηση, όπως και στο μοντέλο 7-επιπέδων του OSI, ξεκινάει από το κατώτερο επίπεδο, το οποίο επιτρέπει τη μεταφορά δεδομένων από το ένα μηχάνημα στο άλλο, ενώ προχωρώντας προς τα ανώτερα επίπεδα, κρύβονται λεπτομέρειες των προηγούμενων επιπέδων.

Web Services Technology Stack

Layer Description	Implementation(s)	Other Concerns			
Standard Messaging	Electronic Business XML Initiative (ebXML)	Quality of Service	Management	Security	Service Development
Service Composition	Business Process Execution Service for Web Services (BPEL4WS)				
Service Registry	Universal Description, Discovery and Integration (UDDI) ebXML Registries				
Service Description	Web Services Description Language (WSDL)				
Service Messaging	Simple Object Access Protocol (SOAP)/Extensible Markup Language (XML)				
Service Transport	Hypertext Transfer Protocol (HTTP) Simple Mail Transfer Protocol (SMTP) File Transfer Protocol (FTP)				

Εικόνα 2: Στρώμα Υπηρεσίας Διαδικτύου

Μία γρήγορη ανάλυση των επιπέδων αυτών είναι:

- **Service Transport (Επίπεδο Μεταφοράς)**

Η λειτουργία του επιπέδου μεταφοράς είναι να μεταφέρει δεδομένα από ένα σημείο σε ένα άλλο. Αποτελεί το πρωτόκολλο για τη μετάδοση δεδομένων των web services. Οι web services χρησιμοποιούν διάφορα πρωτόκολλα για να μεταφέρουν τα δεδομένα από υπηρεσία σε υπηρεσία, συμπεριλαμβανομένων των HTTP, SMTP, FTP, χωρίς όμως να περιορίζονται μόνο σε αυτά.

Το πιο διαδεδομένο πρωτόκολλο μέχρι στιγμής που χρησιμοποιούν οι web services είναι το HTTP. Χρησιμοποιείται περισσότερο γιατί δεν μπλοκάρεται από firewalls και αποτελεί κατ' αυτόν τον τρόπο το πρότυπο για διαλειτουργικά συστήματα.

- **Service Messaging (Επίπεδο Μηνυμάτων)**

Το επίπεδο μηνυμάτων περιγράφει σε τι μορφές θα είναι τα δεδομένα ώστε να μπορούν να μεταφερθούν από τη μια υπηρεσία στην άλλη. Η XML είναι η βασική μορφή που χρησιμοποιείται για τις web services. Τα δεδομένα που διαμορφώνονται με βάση την XML είναι σαν δέντρο (tree) με στοιχεία (elements) και ολόκληρη η δενδρική μορφή καλείται έγγραφο (document). Η XML δεν έχει ξεχωριστή περιγραφή δεδομένων γιατί τα ίδια τα δεδομένα περιγράφουν τον εαυτό τους κι αυτό γιατί έχουν

ειδικές ετικέτες που τους δίνουν ένα όνομα καθώς και την θέση τους μέσα στη δενδρική μορφή.

Το SOAP (Simple Object Access Protocol) είναι μια προδιαγραφή που λέει στον καταναλωτή μιας υπηρεσίας και στον προμηθευτή αυτής πώς να διαμορφώσουν και να διαβάσουν ένα μήνυμα γραμμένο σε XML που χρησιμοποιείται από μια υπηρεσία. Ένα SOAP μήνυμα έχει τρεις περιοχές: τον φάκελο, την επικεφαλίδα και το σώμα. Το SOAP επιτρέπει να εκκινήσει μια λειτουργία με την συγκεκριμένη ακολουθία χαρακτήρων στέλνοντας σε αυτή ένα SOAP μήνυμα. Το SOAP ορίζει την μορφή των XML κειμένων που στένονται πάνω από το HTTP είτε σαν ερωτήσεις είτα σαν απαντήσεις από μια δικτυακή υπηρεσία.

- **Service Description (Επίπεδο Περιγραφής)**

Το επίπεδο περιγραφής καθορίζει τρεις πλευρές μιας web service:

- Μεθόδους τις οποίες μια web service κάνει διαθέσιμες
- Τα μηνύματα που δέχεται μια web service
- Το πρωτόκολλο, το οποίο πρέπει να χρησιμοποιήσει ο πελάτης για να καλέσει τη web service

Πρωταρχικής σημασίας θεωρείται ο ορισμός μιας συγκεκριμένης μεθόδου περιγραφής της κάθε δικτυακής υπηρεσίας. Για τον σκοπό αυτόν δημιουργήθηκε η περιγραφική γλώσσα WSDL(Web Services Description Language)[2] για να καθορίσει μια σύμβαση υπηρεσίας. Η WSDL περιγράφει κάθε δικτυακή υπηρεσία σαν ένα σύνολο από θύρες, οι οποίες κατηγοριοποιούν μια σειρά από διεργασίες που μπορούν να πραγματοποιηθούν ανάμεσα σε έναν χρήστη και στην δικτυακή υπηρεσία. Οι διεργασίες αυτές ονομάζονται λειτουργίες και έχουν ένα μήνυμα εισόδου και ενίοτε ένα σημείο εξόδου. Η κάθε λειτουργία περιγράφει μια διεργασία η οποία μπορεί να πραγματοποιηθεί μεταξύ χρήστη και δικτυακής υπηρεσίας. Αυτό μπορεί να σημαίνει πως η εφαρμογή που την χρησιμοποιεί ζητά από την δικτυακή υπηρεσία να κάνει κάτι και να επιστρέψει το αποτέλεσμα, αλλά και ότι η δικτυακή υπηρεσία μπορεί να εκκινήσει μια διαδικασία για την οποία η εφαρμογή πρέπει να αντιδράσει αναλόγως.

2 http://www.w3.org/TR/wsdl20/#intro

Ο πελάτης μιας υπηρεσίας χρησιμοποιεί την περιγραφή αυτή με δύο τρόπους:

1. Κατά την ώρα της ανάπτυξης της υπηρεσίας (development time), ο πελάτης μπορεί να δημιουργήσει ένα στέλεχος αυτής (stub) χρησιμοποιώντας την περιγραφή αυτής. Το στέλεχος (stub) είναι μια κλάση η οποία συμφωνεί με την περιγραφή της υπηρεσίας. Ο πελάτης συνδέεται με το στέλεχος κατά την ώρα της σύνταξης (compile time). Αυτό καλείται αρχικό δέσιμο (early binding). Ο πελάτης γνωρίζει την διεπαφή την οποία θα χρησιμοποιήσει για να επικοινωνήσει με την δικτυακή υπηρεσία.

2. Οι web services υποστηρίζουν επίσης και την ιδέα του καθυστερημένου δεσίματος (late binding), ένα δέσιμο που γίνεται μόνο κατά την ώρα που εκτελείται μια υπηρεσία μεταξύ του πελάτη και του παροχέα της υπηρεσίας. Αυτό γίνεται χρησιμοποιώντας ένα δυναμικό πληρεξούσιο (dynamic proxy) που διαμορφώνεται δυναμικά κατά την ώρα της εκτέλεσης χρησιμοποιώντας την WSDL περιγραφή της υπηρεσίας.

Η WSDL περιγράφει μία δικτυακή υπηρεσία στα πλαίσια την λειτουργιών που αυτή μπορεί να υλοποιήσει. Με τον τρόπο αυτό δεν περιγράφεται όμως η ακριβής μέθοδος που θα χρησιμοποιήσει μια εφαρμογή για να επικοινωνήσει με την δικτυακή υπηρεσία. Η WSDL επιτρέπει ένα συγκεκριμένο προτύπο σύνδεσης με την δικτυακή υπηρεσία, μέσω του πρωτοκόλλου επικοινωνίας SOAP. Περισσότερα για την WSDL θα δοθούν παρακάτω.

- **Service Registry - Κατάλογος Υπηρεσιών**

Με την βοήθεια του SOAP και της WSDL μπορούμε να περιγράψουμε μια δικτυακή υπηρεσία και να χρησιμοποιήσουμε δικτυακές υπηρεσίες μέσα από εφαρμογές. Το μόνο που έμενε να καθοριστεί ήταν ένας τρόπος ανεύρεσης δικτυακών υπηρεσιών. Για τον σκοπό αυτό δημιουργήθηκε ένα πρότυπο όσον αφορά την δημιουργία ευρετηρίων για την ανεύρεση διαφόρων δικτυακών υπηρεσιών. Οι web services υποστηρίζουν αυτή την ιδέα, της δυναμικής εύρεσης υπηρεσιών. Ένας πελάτης χρησιμοποιεί την αποθήκη υπηρεσιών για να ανακαλύψει υπηρεσίες που τον ενδιαφέρουν . Το UDDI (Universal Description, Discovery and Integration) είναι μια web service από μόνη της που υποστηρίζει ένα σύνολο υπηρεσιών οι οποίες

επιτρέπουν σε ένα χρήστη μιας υπηρεσίας να ανακαλύψει δυναμικά και να εντοπίσει την περιγραφή μιας υπηρεσίας διαδικτύου. Τα UDDI ευρετήρια είναι από μόνα τους υπηρεσίες διαδικτύου που εκθέτουν μία διεπαφή προγραμματισμού εφαρμογών (application programming interface - API) ως ένα σύνολο καλά διαμορφωμένων SOAP μηνυμάτων. Τόσο ο παροχέας μιας υπηρεσίας όσο και ο καταναλωτής αυτής χρησιμοποιούν το SOAP και το HTTP για να δημοσιεύσουν μια υπηρεσία στο ευρετήριο και να λάβουν πληροφορίες για όποιες άλλες ενδιαφέρονται. Τα δημόσια ευρετήρια UDDI υπηρεσιών παρέχουν πληροφορίες για το σημείο στο οποίο μπορείς να καλέσεις την υπηρεσία, τον ιδιοκτήτη της εταιρείας που παρέχει την υπηρεσία και κάποιες τεχνικές πληροφορίες για την web service. Το UDDI υποστηρίζει δύο τύπους συνομιλίας:

▪Ο παροχέας υπηρεσίας χρησιμοποιεί την UDDI αποθήκη για να δημοσιεύσει πληροφορίες σχετικά με τις υπηρεσίες που παρέχει.

▪Ο καταναλωτής μιας υπηρεσίας στέλνει XML μηνύματα διαμορφωμένα σύμφωνα με το SOAP προς την αποθήκη υπηρεσιών για να παραλάβει μια λίστα από υπηρεσίες που ταιριάζουν στα κριτήριά του.

- **Service Composition - Σύνθεση Υπηρεσιών**

Η σύνθεση υπηρεσιών αναφέρεται στην δυνατότητα να συνδυάζουμε υπηρεσίες σε μια ροή εργασίας. Η σύνθεση υπηρεσιών αναφέρεται στην ικανότητα να βάζεις σε μια λογική σειρά και να κατευθύνεις συνομιλίες μεταξύ υπηρεσιών σε μια μεγαλύτερη συναλλαγή. Αυτό αναφέρεται επίσης ως «ενορχήστρωση υπηρεσιών» ή ως «χορογραφία υπηρεσιών» (*service orchestration* ή *service choreography)*.

Ενορχήστρωση: Αναφέρεται σε μια εκτελέσιμη επιχειρηματική διαδικασία που μπορεί να αλληλεπιδράσει με εσωτερικές και εξωτερικές υπηρεσίες διαδικτύου. Η ενορχήστρωση περιγράφει πώς υπηρεσίες διαδικτύου μπορούν να αλληλεπιδράσουν στο επίπεδο μηνύματος, συμπεριλαμβανομένης της επιχειρηματικής λογικής καθώς και της εκτέλεσης των αλληλεπιδράσεων. Η ενορχήστρωση[3] πραγματοποιείται από κάποιον που μπορεί να συντονίζει τις υπηρεσίες. Ένα μηχάνημα ή μία υπηρεσία, καλεί

3 http://www2.sys-con.com/itsg/virtualcd/webservices/archives/0307/peltz/index.html

όλα τα υπόλοιπα και διαχειρίζεται εκείνο/εκείνη τις αιτήσεις αλλά και τα αποτελέσματα που αυτές του/της επιστρέφουν.

Χορογραφία: μεγαλύτερη συνεργασία από την φύση της, όπου καθένα από τα μέρη που εμπλέκονται στη διαδικασία περιγράφει το ρόλο τους στην αλληλεπίδραση. Στην χορογραφία καταγράφεται η ακολουθία των μηνυμάτων που μπορεί να περιλαμβάνει πολλά μέρη και πολλαπλές πηγές. Στην χορογραφία, οι ίδιες οι υπηρεσίες γνωρίζουν μόλις πάρουν ένα αίτημα για να εξυπηρετήσουν πού ακριβώς θα πρέπει να απευθυνθούν για να δώσουν το αποτέλεσμά τους. Συνεπώς, στην δεύτερη περίπτωση δεν απαιτείται κάποιος κεντρικός συντονιστής προκειμένου να περατωθεί μία υπηρεσία.

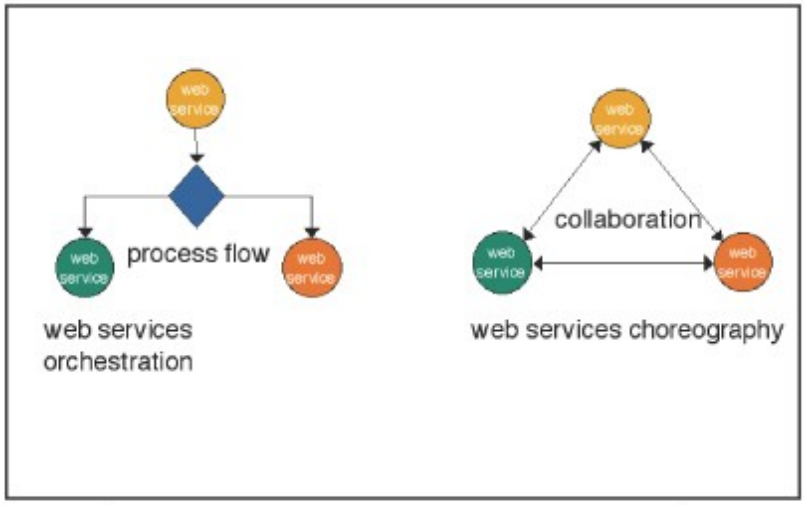


FIGURE 1 | **Orchestration and choreography**

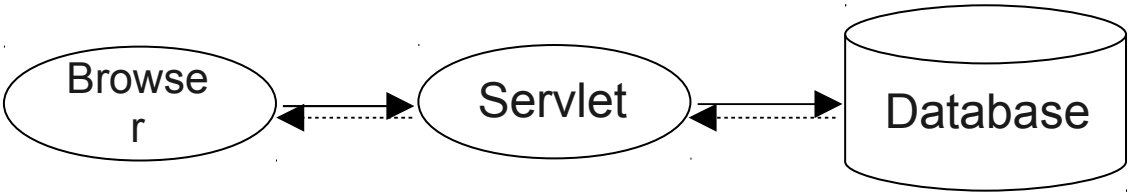
Υπάρχει πληθώρα προτεινόμενων γλωσσών για να περιγράψουν ένα επιχειρηματικό διαδικαστικό διάγραμμα. Οι δύο περισσότερο γνωστές είναι οι WSFL και XLANG. Πρόσφατα οι εταιρείες IBM, Microsoft και BEA συνδύασαν τις δύο παραπάνω γλώσσες σε *μία* προδιαγραφή που ονομάζεται BPEL4WS (Business Process Execution Language for Web Services), με την οποία θα ασχοληθούμε παρακάτω.

1.3 Σενάρια υλοποίησης

Αφού μια web service μπορεί να αναπτυχθεί σε πολλές διαφορετικές πλατφόρμες, μπορεί και να ακολουθήσει πολλά μοντέλα υλοποίησης. Το μοντέλο υλοποίησης που χρησιμοποιεί εξαρτάται από το πρόβλημα που πρέπει να λυθεί και τις διαθέσιμες πλατφόρμες για την δημιουργία του μοντέλου. Το μοντέλο για την ολοκλήρωση μιας web service μπορεί να κατηγοριοποιηθεί σε 4 τύπους: ***απλής υπηρεσίας, σύνθετης υπηρεσίας, ενδιάμεσης υπηρεσίας και ενός διαδρόμου υπηρεσιών.***

- **Απλή υπηρεσία**

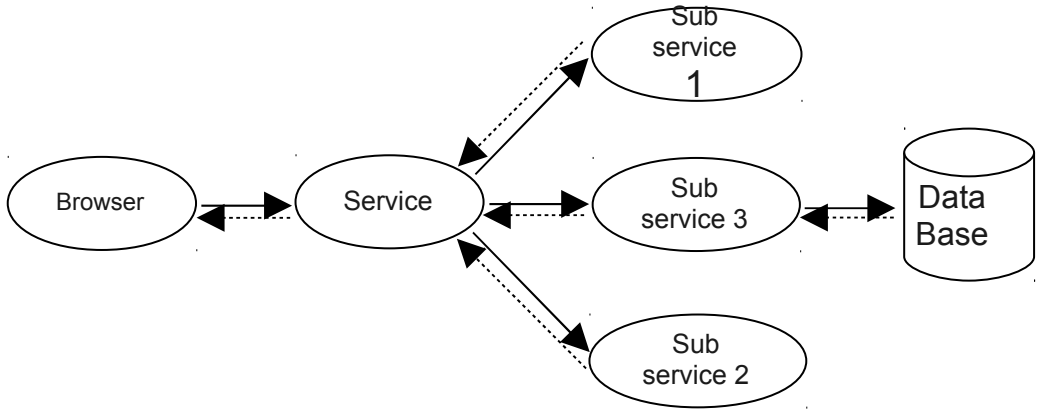
Η πιο απλή υπηρεσία είναι να έχουμε έναν web service το οποίο θα μας επιστρέφει μία σελίδα Http στον φυλλομετρητή μας. Έτσι, κάθε φορά που ένας χρήστης εισάγει στον φυλλομετρητή του την τοποθεσία της υπηρεσίας μου, εκείνη θα του επιστρέφει ένα σύνολο από bytes που θα του επιτρέπουν αν "δει" την ιστοσελίδα μου.



Εικόνα 3: Σενάριο απλής υπηρεσίας

- **Σύνθετη Υπηρεσία**

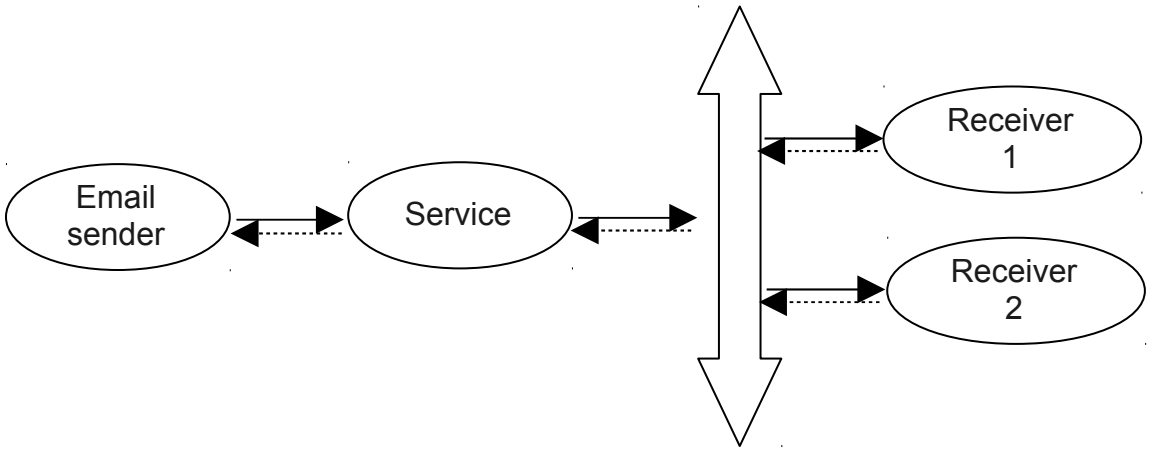
Μια σύνθετη υπηρεσία στην ουσία είναι μία ένωση πολλών απλών ή ακόμα και επιπλέον σύνθετων υπηρεσιών. Παραδείγματος χάριν, θα μπορούσε μία σύνθετη υπηρεσία να είναι η εξής: Ο χρήστης να δίνει τα credentials του καθώς και έναν κωδικό προϊόντος και η υπηρεσία να αναθέτει σε μία υπηρεσία να ελέγξει εάν τα credentials είναι σωστά, σε μία άλλη να χρεώσει την πιστωτική του καρτα, ενώ σε μία τρίτη υπηρεσία να μειώνει το προϊόν από την αποθήκη κατά 1 και ταυτόχρονα να αποστέλλει ένα προϊόν στο ταχυδρομείο. Κατ'αυτόν τον τρόπο οι επιμέρους υπηρεσίες μπορούν να χρησιμοποιηθούν ξεχωριστά αλλά και συνεργατικά, όποτε αυτό κρίνεται σκόπιμο, όπως σε αυτό το παράδειγμα.



Εικόνα 4: Σενάριο Σύνθετης Υπηρεσίας

- **Ενδιάμεση Υπηρεσία**

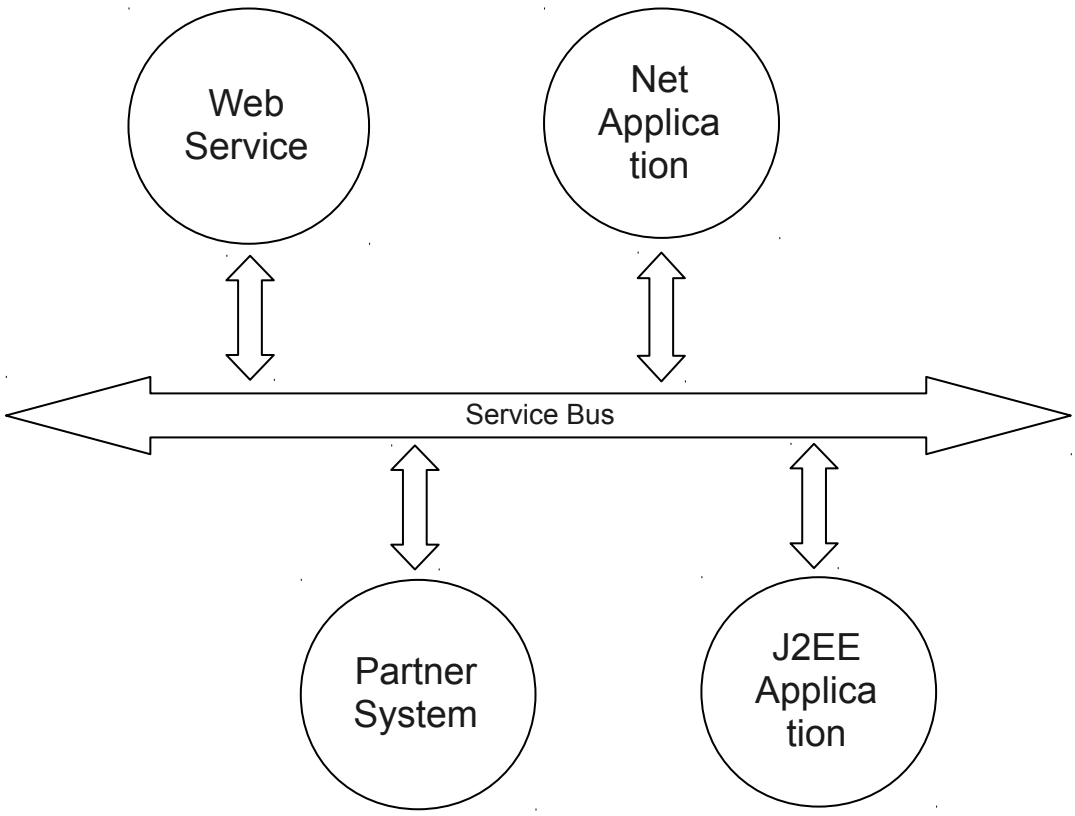
Μια δικτυακή υπηρεσία που τοποθετεί αιτήσεις προς εξυπηρέτηση σε έναν ενδιάμεσο διάδρομο είναι ένα στιγμιότυπο υλοποίησης ενδιάμεσης υπηρεσίας. Παραδείγματος χάριν, μπορεί μία εταιρεία να υλοποιήσει κατ' αυτόν τον τρόπο τα emails των εργαζομένων της. Δηλαδή, δε χρειάζεται να γνωρίζει ο αποστολέας πού βρίσκεται ο παραλήπτης, την διαδικασία αυτή την αναλαμβάνει να την κάνει η υπηρεσία.



Εικόνα 5: Σενάριο Ενδιάμεσης Υπηρεσίας

- **Διάδρομος Υπηρεσιών**

Ο διάδρομος υπηρεσιών επιτρέπει στις web services να επικοινωνούν μεταξύ τους μέσω τρίτων. Παραδείγματος χάριν, έστω μία τράπεζα και ένας χρήστης ο οποίος έχει έναν λογαριασμό στην τράπεζα αυτή. Αν ο χρήστης μεταφέρει μέσω internet κάποια χρήματα από έναν λογαριασμό και δώσει πάγια εντολή, η δικτυακή υπηρεσία θα πρέπει να στείλει ένα μήνυμα στον διάδρομο υπηρεσίας. Ο διάδρομος υπηρεσίας αναλαμβάνει να ενημερώσει την υπηρεσία ανάληψης χρημάτων όπως και την υπηρεσία καταθέσεων. Όταν η υπηρεσία αναλήψεων μπορέσει να επεξεργαστεί (ενδεχομένως να έχει ουρά εξυπηρέτησης) το αίτημα και πιστοποιήσει ότι ο λογαριασμός δεν έχει αρνητικό υπόλοιπο, τότε αποστέλλει ένα μήνυμα στην υπηρεσία καταθέσεων για να καταθέσει τα χρήματα στον 2ο λογαριασμό, σε αυτόν της πάγιας εντολής.



Εικόνα 12: Σενάριο Διαδρόμου υπηρεσιών

Αυτή η μέθοδος δρομολόγησης μηνυμάτων μεταξύ των web services είναι πολυδύναμη. Ταυτόχρονα με την υπηρεσία καταθέσεων και αναλήψεων, μπρούν και άλλες υπηρεσίες ζητήσουν να αλλάξουν τον λογαριασμό, όπως φερειπείν τα στοιχεία του πελάτη. Αυτή είναι μια multi-cast μέθοδος επικοινωνίας, με την οποία ένα χρήστης μπορεί να αποστείλει μηνύματα σε πολλαπλές υπηρεσίες ταυτόχρονα. Ο χρήστης δεν γνωρίζει επιπλέον ποιες άλλες υπηρεσίες συμμετέχουν στην αίτησή του. Το λογισμικό διαχείρισης του διαδρόμου υπηρεσιών καθοδηγεί τη δρομολόγηση αυτών των μηνυμάτων.

1.4 Πλεονεκτήματα και μειονεκτήματα

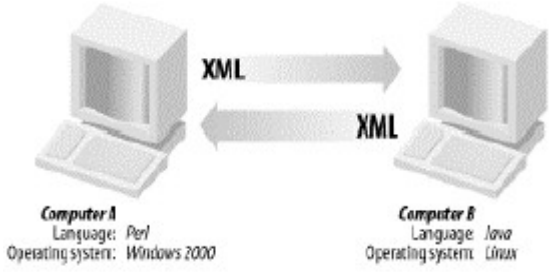
1.4.1 Πλεονεκτήματα

Τα web services σε αντίθεση με τα παραδοσιακά αντικειμονεστραφή components είναι πιο χαλαρά συνδεδεμένα με τους πελάτες τους, απ'ότι τα παραδοσιακά αντικειμενοστραφή components. Για παράδειγμα, μια αλλαγή σε μία μέθοδο και κατ' επέκταση στην διεπαφή ενός παραδοσιακού αντικειμενοστραφούς κομματιού κώδικα θα απαιτούσε αλλαγή και επαναμετάφραση του κώδικα του πελάτη, καθώς αυτά είναι ισχυρά συζευγμένα. Τα web services, απαλλάσσονται από αυτόν τον μπελά και ο πελάτης μπορεί να φτιάξει εκ νέου και δυναμικά το στέλεχος (stub) της υπηρεσίας, μέσω του WSDL εγγράφου που περιγράφει την υπηρεσία. Η ιδιότητα αυτή αναφέρεται στην βιβλιογραφία ως *χαλαρή σύζευξη (loose coupling)*.

Το WSDL έγγραφο, επίσης, προσφέρει στα web services ακόμα περισσότερη ελευθερία και δύναμη. Το WSDL έγγραφο μπορεί να τροποποιηθεί κατάλληλα έτσι, ώστε να ξεπερνά τις προδιαγραφές του πρωτοκόλλου SOAP. Μπορεί να υιοθετηθεί από την υπηρεσία επιπλέον ασφάλεια, είδος κρυπτογράφησης, προστασία από μη εξουσιοδοτημένους χρήστες κτλ.

Τα web services δεν εξαρτώνται από την πλατφόρμα που υλοποιούνται καθώς επίσης και από την γλώσσα υλοποίησής τους, λόγω της κοινά αποδεκτής γλώσσας XML. Αυτό σημαίνει ότι μπορεί ένα πρόγραμμα πελάτη να έχει υλοποιηθεί σε Perl και

να εκτελείται σε περιβάλλον Windows, ενώ το web service να είναι προγραμματισμένο σε Java και να εκτελείται σε περιβάλλον Linux.



Εικόνα 4: Μια απλή υπηρεσία διαδικτύου, όπου διαφαίνεται ότι η γλώσσα επεξεργασίας των μηνυμάτων xml είναι διαφορετική, όπως και το λειτουργικό σύστημα.

Τα περισσότερα web services χρησιμοποιούν το πρωτόκολλο HTTP για την μετάδοση των μηνυμάτων τους (service request and service response). Αυτό αποτελεί μεγάλο πλεονέκτημα γιατί έτσι αποφεύγονται ζητήματα συμπλοκής με firewalls, αφού η πόρτα 80 σχεδόν πάντα είναι ανοικτή.

1.4.2 Μειονεκτήματα

Τα web services αν και αρκετά σταθερά, δεν παύουν να βρίσκονται σε αρχικό στάδιο, γεγονός που τα κάνουν ακόμα επιρρεπή σε αστάθειες, παρά το γεγονός ότι βασίζονται σε απλά, σταθερά και ασφαλή πρότυπα (XML, WSDL) και πρωτόκολλα (HTTP, SOAP). Ωστόσο, αυτό το μειονέκτημα σε λίγο καιρό θα είναι αμελητέο διότι ο ρυθμός ανάπτυξής τους είναι πολύ γρήγορος.

Επίσης, αν και όχι εγγενές μειονέκτημα των web services, είναι το γεγονός ότι χρησιμοποιούν μεγάλη ποσότητα δεδομένων για να επικοινωνήσουν. Η μετάδοση των δεδομένων βάσει της XML μορφής αντί της δυαδικής προσθέτει βάρος στο δίκτυο, αλλά και πολυπλοκότητα στο ίδιο το μηχάνημα που τρέχει την υπηρεσία. Το κέρδος σε φορητότητα, ουσιαστικά, που προσφέρει η XML χάνεται σε απόδοση.

1.5 XML

1.5.1 Εισαγωγή

Σε ένα κόσμο όπου οι πληροφορίες παρέχονται μέσω του παγκόσμιου διαδικτύου, τα έγγραφα πρέπει να είναι εύκολα προσβάσιμα, μεταφέρσιμα και ευέλικτα. Πρέπει επίσης να είναι ανεξάρτητα οποιουδήποτε συστήματος και περιεχομένου. Οι γενικευμένες γλώσσες έχουν τέτοια χαρακτηριστικά , παρέχοντας στα έγγραφα αυτά μια δυνατότητα η οποία δεν υπάρχει σε άλλες γλώσσες περιγραφής εγγράφων. Η HTML είναι προβληματική και περιοριστική γλώσσα. Η XML έλυσε πολλά από τα προβλήματα που αντιμετώπισαν οι σχεδιαστές του web και είναι υπεύθυνη για την XHTML, μια ανασχεδιασμένη HTML. Θα χρησιμοποιείται για πολλά χρόνια επειδή προσφέρει αποτελεσματικές και δυναμικές λύσεις.

Η XML σχεδιάστηκε να ικανοποιήσει πολλές ανάγκες δίνοντας στα έγγραφα ένα μεγαλύτερο επίπεδο προσαρμοστικότητας στο στυλ και τη δομή από αυτό που υπήρχε παλαιότερα στην HTML. Η XML προσφέρει στους σχεδιαστές της HTML τη δυνατότητα να προσθέτουν περισσότερα στοιχεία στη γλώσσα. Δεν αναφέρεται μονάχα στους σχεδιαστές του web αλλά σε οποιονδήποτε ασχολείται με εκδόσεις.

Στην πραγματικότητα, η XML είναι γλώσσα προσαύξησης (markup) για έγγραφα που περιέχουν δομημένες πληροφορίες. Γλώσσα προσαύξησης είναι ένας μηχανισμός που καθορίζει δομές σε ένα έγγραφο. Οι δομημένες πληροφορίες περιλαμβάνουν περιεχόμενο και κάποιες διευκρινίσεις για το ρόλο που παίζει το περιεχόμενο. Σχεδόν όλα τα έγγραφα έχουν την ίδια δομή.

1.5.2 Τι είναι η XML;

- Η XML είναι συντομογραφία για το E**X**tensible **M**arkup **L**anguage
- Η XML είναι **markup** γλώσσα, η οποία μοιάζει πολύ με την HTML
- Η XML σχεδιάστηκε για να **περιγράφει δεδομένα**
- Τα XML tags δεν είναι προκαθορισμένα. Πρέπει να ορίσουμε τα **δικά μας tags**

- Η XML χρησιμοποιεί το **Document Type Definition** (DTD) ή το **XML Schema** για να περιγράψει τα δεδομένα

- Ένα XML κείμενο με ένα DTD ή ένα XML Schema σχεδιάστηκε για να **περιγράφει επαρκώς τον εαυτό του**

1.5.3 Διαφορές μεταξύ XML και HTML

- Η XML σχεδιάστηκε για τη μεταφορά δεδομένων.

- Η XML δεν είναι ένα αντικατάστατο της HTML.

- Η XML σχεδιάστηκε για να **περιγράφει** τα δεδομένα και εστιάζει στο **τι είναι** τα δεδομένα.

- Η HTML σχεδιάστηκε για να **παρουσιάζει** τα δεδομένα και εστιάζει στο **πώς φαίνονται** τα δεδομένα.

- Η HTML έχει να κάνει με την παρουσίαση των δεδομένων ενώ η XML έχει να κάνει με την περιγραφή των δεδομένων.

1.5.4 Η XML δεν κάνει κάτι συγκεκριμένο

```
<email>
<to>Nick</to>
<from>Mom</from>
<heading>Goodbye</heading>
<body>Have a great time at the camp!</body>
</email>
```

Μπορεί να είναι λίγο δύσκολο να το καταλάβει κανείς, αλλά η XML δεν κάνει κάτι συγκεκριμένο. Η XML σχεδιάστηκε για τη δόμηση, την αποθήκευση και την αποστολή δεδομένων. Το παρακάτω παράδειγμα είναι ένα email της μητέρας ενός παιδιού στο παιδί της, αποθηκευμένο σε XML:

Το σημείωμα έχει μια κεφαλή (heading) και ένα σώμα (body), το οποίο περιέχει το μήνυμα. Περιέχει επίσης πληροφορίες για τον αποστολέα και τον αποδέκτη. Παρ'

όλα αυτά όμως το XML κείμενο εξακολουθεί να μην κάνει κάτι συγκεκριμένο. Περιέχει απλά πληροφορίες, τοποθετημένες μέσα σε XML tags. Κάποιος πρέπει να γράψει ένα κομμάτι software για να το στείλει, να το λάβει ή να το παρουσιάσει.

1.5.5 Η XML είναι απλή και επεκτάσιμη

Τα tags που χρησιμοποιούνται στα HTML κείμενα είναι προκαθορισμένα. Ο συγγραφέας ενός HTML κειμένου μπορεί να χρησιμοποιήσει μόνο τα tags που ορίζονται στο HTML standard (όπως <p>,
 κ.λ.π.).

Η XML επιτρέπει στο συγγραφέα ενός XML κειμένου να ορίσει τα δικά του tags. Για παράδειγμα τα tags <to> και <from> στο παραπάνω παράδειγμα δεν είναι ορισμένα σε κανένα XML standard. Αυτά τα tags «επινοήθηκαν» από το συγγραφέα αυτού του XML κειμένου.

1.5.6 Η XML είναι ένα συμπλήρωμα της HTML

Είναι σημαντικό να καταλάβει κανείς ότι η XML δεν αποτελεί ένα αντικατάστατο της HTML. Είναι πολύ πιθανόν ότι οι μελλοντικές Web εφαρμογές θα χρησιμοποιούν την XML για να περιγράφουν τα δεδομένα, ενώ η HTML θα χρησιμοποιείται για την απεικόνιση των δεδομένων αυτών.

Για αυτό και η καλύτερη περιγραφή της XML είναι η εξής: η XML είναι ένα ανεξάρτητο πλατφόρμας, ανεξάρτητο από software και hardware εργαλείο για την μετάδοση πληροφοριών.

1.5.7 Η XML στην ανάπτυξη μελλοντικών Web εφαρμογών

Είναι συναρπαστικό το πόσο γρήγορα το XML standard αναπτύχθηκε και πόσο γρήγορα ένας μεγάλος αριθμός από σχεδιαστές λειτουργικών συστημάτων υιοθέτησαν το standard αυτό.

Πιστεύεται ότι η XML θα είναι τόσο σημαντική για το μέλλον του Web όσο ήταν η HTML για τη θεμελίωση του. Πιστεύεται επίσης ότι η XML θα είναι το πιο κοινό εργαλείο για την επεξεργασία των δεδομένων και τη μετάδοση τους.

1.6 WSDL (Web Service Description Language)

Είδαμε στην προηγούμενη παράγραφο ότι οι υπηρεσίες διαδικτύου πρέπει να παρέχουν μία καλώς ορισμένη διεπαφή (interface), με την χρήση της WSDL. Το WSDL αρχείο καλείται συμβόλαιο (contract) και στην ουσία περιγράφει web services και χρησιμοποιείται για τον εντοπισμό τους.

Η WSDL είναι μια προδιαγραφή μίας γλώσσας που καθορίζει τον τρόπο περιγραφής των web services σε μία XML γραμματική, που δεν είναι ακόμα W3C standard. Η WSDL περιγράφει 4 σημαντικά κομμάτια δεδομένων:

◆ Πληροφορία για όλες τις δημόσια διαθέσιμες λειτουργίες

◆ Πληροφορίες τύπων δεδομένων για όλα τα μηνύματα αίτησης και απάντησης

◆ Πληροφορίες σύνδεσης σχετικά με τα πρωτόκολλα μεταφοράς

◆ Πληροφορίες διευθύνσεων για τον εντοπισμό της συγκεκριμένης υπηρεσίας

Χρησιμοποιώντας την WSDL ένας πελάτης μπορεί να εντοπίσει μια web service και να καλέσει οποιαδήποτε από τις δημόσιες λειτουργίες της.

1.6.1 Η WSDL προδιαγραφή

Η WSDL προδιαγραφή αποτελείται από 2 μέρη:το λογικό και το φυσικό. Το λογικό κομμάτι ενός αρχείου WSDL περιγράφει τα "αφηρημένα"(abstract) χαρακτηριστικά της δικτυακής υπηρεσίας και συμπεριλαμβάνει τα παρακάτω κομμάτια:

- **Definitions**

Το στοιχείο definitions πρέπει να είναι η ρίζα του δένδρου για κάθε WSDL αρχείο. Καθορίζει το όνομα της Web service, δηλώνει τα πολλαπλά namespaces (εκτός περιπτώσεων όπου υπάρχει υπερκάλυψη-override) που χρησιμοποιούνται στο υπόλοιπο κείμενο και περιέχει όλα τα υπόλοιπα στοιχεία της υπηρεσίας. Η χρήση των namespaces είναι σημαντική για να ξεχωρίζουμε τα στοιχεία μεταξύ τους και δίνει την δυνατότητα στο wsdl κείμενο να αναφέρεται σε εξωτερικές προδιαγραφές, συμπεριλαμβανομένων και των WSDL, SOAP και XML Schema προδιαγραφών. Το στοιχείο definitions, επίσης, δηλώνει και μια ιδιότητα targetNamespace. Αυτή η ιδιότητα είναι μια σύμβαση του XML Σχήματος η οποία δίνει τη δυνατότητα στο WSDL κείμενο να αναφέρεται στον εαυτό του. Θα πρέπει να σημειώσουμε, όμως, ότι η προδιαγραφή σχετικά με το namespace δεν απαιτεί να υπάρχει όντως το κείμενο WSDL στην τοποθεσία. Το σημαντικό είναι να καθορίζεται μια τιμή που είναι μοναδική και διαφορετική από τα υπόλοιπα namespaces. Τέλος, το στοιχείο definitions δηλώνει και ένα δικό του namespace, το *xmlns=http://schemas.xmlsoap.org/wsdl/*. Όλα τα στοιχεία που δεν έχουν κάποιο πρόθεμα, όπως το message, το portType κτλ., θεωρούνται ότι αναφέρονται σε αυτό το namespace.

- **Types**

Το στοιχείο types περιγράφει όλους τους τύπους δεδομένων που χρησιμοποιούνται μεταξύ του πελάτη και του εξυπηρετητή. Η WSDL δεν ακολουθεί αποκλειστικά ένα συγκεκριμένο σύστημα δήλωσης τύπων, αλλά χρησιμοποιεί την W3C XML προδιαγραφή σχήματος (schema) ως προκαθορισμένη επιλογή της. Αν η υπηρεσία χρησιμοποιεί μόνο XML σχήμα δομημένο σε απλούς τύπους, όπως αλφαριθμητικά και ακεραίους, το στοιχείο types δεν είναι απαραίτητο. Εάν ωστόσο έχει πιο σύνθετους τύπους, αυτό το κομμάτι είναι ο μόνος τρόπος να οριστούν. ορίζει τους τύπους δεδομένων που χρησιμοποιεί το Web Service.

- **Message**

Το στοιχείο message περιγράφει ένα μήνυμα μιας κατεύθυνσης είτε αυτό είναι ένα μήνυμα αίτησης είτε απάντησης. Καθορίζει το όνομα του μηνύματος και αν περιέχει μηδέν στοιχεία part υποθέτει ότι δεν μεταδίδεται τίποτα. Εάν έχει ένα ή περισσότερα στοιχεία ‘part’ τότε αποστέλλει όσα αναφέρονται στις παραμέτρους του μηνύματος ή στις επιστρεφόμενες τιμές του. Επίσης, Περιγράφει τους τύπους

δεδομένων που χρησιμοποιεί μια operation. Κάθε μήνυμα αποτελείται από ένα ή περισσότερα κομμάτια. Τα κομμάτια αυτά μπορούν να συγκριθούν με τις παραμέτρους μιας συνάρτησης σε μια παραδοσιακή προγραμματιστική γλώσσα. Για την αίτηση, αυτό το στοιχείο καθορίζει τις παραμέτρους που δέχεται μια συνάρτηση. Η ιδιότητα type του στοιχείου part καθορίζει έναν τύπο σύμφωνο με το XML Schema. Η τιμή της ιδιότητας αυτής πρέπει να δοθεί ως XML Schema QName, δηλαδή να έχει ένα πρόθεμα που να αναφέρεται σε ένα namespace. Συνήθως, η ιδιότητα type των στοιχείων έχουν τιμή 'xsd:...'. Το xsd πρόθεμα αναφέρεται στο namespace του XML σχήματος που ορίστηκε μέσα στα namespaces του στοιχείου definitions.

- **Operation**

Το στοιχείο operation περιγράφει μία πράξη που μπορεί να κάνει μία υπηρεσία, προσδιορίζοντας μηνύματα εισόδου και εξόδου που χρειάζονται ως παράμετροι για αυτή την πράξη.

- **PortType**

Το στοιχείο PortType συνδυάζει πολλαπλά στοιχεία 'message' για να συνθέσει μια απλή λειτουργία είτε μιας ή αμφίδρομης κατεύθυνσης. Το portType στοιχείο είναι το πιο σημαντικό στοιχείο ενός WSDL κειμένου. Περιγράφει μια Web Service, τις λειτουργίες (operations) που μπορούν να εκτελεστούν, καθώς και τα μηνύματα που ανταλλάσσονται. Το portType μπορεί να συγκριθεί με μια βιβλιοθήκη συναρτήσεων ή ένα module ή μια κλάση στις απλές προγραμματιστικές γλώσσες. Για παράδειγμα, ένα portType μπορεί να συνδυάζει ένα μήνυμα αίτησης και απάντησης σε μια μοναδική λειτουργία αίτησης-απάντησης η οποία χρησιμοποιείται πιο συχνά στις SOAP υπηρεσίες. Αξίζει να σημειωθεί ότι ένα portType συχνά καθορίζει πολύπλοκες λειτουργίες. Όπως και στην ιδιότητα type που αναφέραμε στο στοιχείο message, η ιδιότητα message πρέπει να οριστεί σαν XML Schema QName, δηλαδή να αναφέρεται σε ένα namespace.

Τύπος	**Ορισμός**
One-way	Η λειτουργία λαμβάνει ένα μήνυμα αλλά δεν επιστρέφει απάντηση Η υπηρεσία λαμβάνει το μήνυμα. Έτσι το στοιχείο

	operation έχει ένα μοναδικό στοιχείο input.
Request-response	Η λειτουργία λαμβάνει μια αίτηση και επιστρέφει μια απάντηση Η υπηρεσία λαμβάνει ένα μήνυμα αίτησης και στέλνει ένα μήνυμα απάντησης. Έτσι το στοιχείο operation έχει ένα στοιχείο input που ακολουθείται από ένα στοιχείο output
Solicit-response	Η λειτουργία στέλνει μια αίτηση και περιμένει μια απάντηση Η υπηρεσία στέλνει ένα μήνυμα και λαμβάνει έπειτα μια απάντηση. Έτσι, το στοιχείο operation έχει ένα στοιχείο output που ακολουθείται από ένα στοιχείο input
Notification	Η λειτουργία στέλνει ένα μήνυμα αλλά δεν περιμένει απάντηση Η υπηρεσία στέλνει ένα μήνυμα. Έτσι, το στοιχείο operation έχει ένα μοναδικό στοιχείο output

Το φυσικό κομμάτι ενός αρχείου WSDL περιγράφει τα συμπαγή χαρακτηριστικά μιας υπηρεσίας διαδικτύου και συμπεριλαμβάνει τα παρακάτω κομμάτια:

- **Binding**

Το στοιχείο binding περιγράφει το συγκεκριμένο τρόπο υλοποίησης της υπηρεσίας πάνω στο καλώδιο, συνδέει ένας συνεπές πρωτόκολλο και τις προδιαγραφές των μηνυμάτων με operations και messages, ορισμένα μέσα σε ένα συγκεκριμένο porttype που έχει οριστεί στο λογικό κομμάτι του αρχείου. Πιο απλά, το στοιχείο binding παρέχει ειδικές λεπτομέρειες για το πώς μια λειτουργία που καθορίζεται από το στοιχείο portType θα μεταφερθεί πάνω στο καλώδιο. Οι συνδέσεις μπορούν να γίνουν μέσω διαφόρων πρωτοκόλλων, συμπεριλαμβανομένων των HTTP GET, HTTP POST ή SOAP. Στην πραγματικότητα μπορείς να ορίσεις πολλές συνδέσεις (bindings) για κάθε στοιχείο portType. Το στοιχείο binding έχει 2 χαρακτηριστικά (attributes) – το όνομα και τον τύπο. Το name (μπορούμε να χρησιμοποιήσουμε ό,τι όνομα θέλουμε) καθορίζει το όνομα του binding και το type αναφέρεται στο στοιχείο portType που έχει οριστεί πιο πάνω στο WSDL κείμενο. Το soap:binding έχει 2 χαρακτηριστικά – το

χαρακτηριστικό style και το χαρακτηριστικό transport. Το χαρακτηριστικό style μπορεί να είναι “rpc” ή “document”. Το χαρακτηριστικό transport καθορίζει το SOAP πρωτόκολλο που θα χρησιμοποιηθεί. Στην περίπτωσή μας χρησιμοποιούμε το HTTP. Το operation ορίζει κάθε λειτουργία που εκθέτει το port. Για κάθε λειτουργία η αντίστοιχη SOAP ενέργεια πρέπει να οριστεί. Πρέπει επίσης να οριστεί και ο τρόπος που θα κωδικοποιηθεί η είσοδος (input) και η έξοδος (output) (συνήθως χρησιμοποιούμε το “literal”).

- **Port**

Το στοιχείο port εγκαθιδρύει ένα endpoint συνδέοντας ένα binding με μια συνεπή διεύθυνση δικτύου.

- **Service**

Το στοιχείο service καθορίζει τη διεύθυνση για την κλήση της συγκεκριμένης υπηρεσίας και περιέχει το URL της υπηρεσίας.

Επιπρόσθετα στα παραπάνω επτά βασικά στοιχεία, η WSDL καθορίζει επίσης τα ακόλουθα χρήσιμα στοιχεία:

- **Documentation**

Το στοιχείο documentation χρησιμοποιείται για να παρέχει μια κατανοητή από τον άνθρωπο περιγραφή της υπηρεσίας και μπορεί να εμπεριέχεται σε οποιοδήποτε WSDL στοιχείο.

- **Import**

Το στοιχείο import χρησιμοποιείται για να εισάγει άλλα αρχεία WSDL ή σχήματα XML κυρίως για την υιοθέτηση τύπων (types σε αρχεία xsd).

1.6.2 Παράδειγμα

Έστω το wsdl αρχείο:

```
<?xml version="1.0" encoding="utf-8"?>
<definitions name ="poService"
        xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
```

```
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns="http://schemas.xmlsoap.org/wsdl/"
    targetNamespace="http://localhost/WebServices/ch1/poService">
<message name="getOrderStatusInput">
    <part name="body" element="xsd:string"/>
</message>
<message name="getOrderStatusOutput">
    <part name="body" element="xsd:string"/>
</message>
<portType name="poServicePortType">
    <operation name="getOrderStatus">
      <input message="tns:getOrderStatusInput"/>
      <output message="tns:getOrderStatusOutput"/>
    </operation>
</portType>
<binding name="poServiceBinding" type="tns:poServicePortType">
    <soap:binding style="rpc"
        transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="getOrderStatus">
      <soap:operation
        soapAction=
          "http://localhost/WebServices/ch1/getOrderStatus"/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
</binding>
<service name="poService">
    <port name="poServicePort" binding="tns:poServiceBinding">
      <soap:address
        location=
          "http://localhost/WebServices/ch1/SOAPserver.php"/>
    </port>
</service>
```

```
</definitions>
```

Θα δούμε αυτό το αρχείο σε λεπτομέρεια προκειμένου να καταλάβουμε στην πράξη την σύνταξη ενός WSDL αρχείου.

Το στοιχείο definitions είναι η ρίζα κάθε WSDL αρχείου, περικλείοντας όλες τους ορισμούς που χρησιμοποιούνται στο αρχείο. Επιπλέον, έχει όλες τα namespaces που θα χρησιμοποιούνται γενικά μέσα στο αρχείο.

```
<definitions name ="poService"
        xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns="http://schemas.xmlsoap.org/wsdl/"
          targetNamespace=
            "http://localhost/WebServices/ch1/poService">
```

Στην συνέχεια, ορίζουμε τους αφηρημένους ορισμούς για τα μηνύματα που θα χρησιμοποιηθούν για την ανταλλαγή δεδομένων. Εδώ παρέχεται ο αφηρημένος ορισμός για το μήνυμα που θα χρησιμοποιηθεί για να μεταφέρει την παράμετρο εισόδου για την συνάρτηση getOrderStatus:

```
<message name="getOrderStatusInput">
      <part name="body" element="xsd:string"/>
  </message>
```

Έπειτα, ορίζουμε αφηρημένα το μήνυμα που θα χρησιμοποιηθεί για να σταλεί πίσω το αποτέλεσμα της συνάρτησης getOrderStatus:

```
<message name="getOrderStatusOutput">
      <part name="body" element="xsd:string"/>
  </message>
```

Άπαξ και έχουν οριστεί τα μηνύματα, μπορούν να ομαδοποιηθούν σε operations, οι οποίες με την σειρά τους ομαδοποιούνται σε μία διεπαφή service. Εδώ παρατίθεται το τμήμα portType που αντιπροσωπεύει μία αφηρημένη αποτύπωση της διεπαφής του service,το οποίο, σε αυτό το παράδειγμα, υποστηρίζει μόνο μία operation:

```
<portType name="poServicePortType">
      <operation name="getOrderStatus">
        <input message="tns:getOrderStatusInput"/>
```

```
        <output message="tns:getOrderStatusOutput"/>
      </operation>
    </portType>
```

Τώρα που έχουμε ορίσει την αφηρημένη διεπαφή της υπηρεσίας, μπορούμε να προσδιορίσουμε τις φυσικές λεπτομέρειες της ανταλλαγής δεδομένων. Σε ένα τμήμα binding, χαρτογραφούμε την αφηρημένη διεπαφή της υπηρεσίας μέσα σε ένα portType τμήμα, προσδιορίζοντας ένα συνεπές πρωτόκολλο για μετάδοση δεδομένων και τις προδιαγραφές των μηνυμάτων. Σε αυτό το παράδειγμα, το τμήμα binding χρησιμοποιείται για να γίνει δημοσιοποίηση (deploy) η operation getOrderStatus – η μοναδική operation που υποστηρίζεται από την υπηρεσία.

```
<binding name="poServiceBinding" type="tns:poServicePortType">
    <soap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="getOrderStatus">
      <soap:operation soapAction=
        "http://localhost/WebServices/ch1/getOrderStatus"/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>
```

Στο παραπάνω παράθεμα, ορίζουμε ένα binding πρωτοκόλλου SOAP της μορφής request-response RPC operation πάνω από HTTP και προσδιορίζουμε το συνεπές URI υποδεικνύοντας τον σκοπό του SOAP HTTP request.

Τέλος, χρησιμοποιούμε το στοιχείο service που μας δείχνει το στοιχείο port για να συγκεκριμενοποιηθεί η φυσική διεύθυνση της υπηρεσίας

```
<service name="poService">
    <port name="poServicePort" binding="tns:poServiceBinding">
      <soap:address location=
        "http://localhost/WebServices/ch1/SOAPServer.php"/>
```

```
        </port>
    </service>
```

Στο παραπάνω παράδειγμα, η λειτουργία getOrderStatus που είναι μια δικτυακή υπηρεσία, λαμβάνει μόνο μία παράμετρο εισόδου. Τί γίνεται, όμως, αν χρειαστεί να περαστούν περισσότερες από μία παράμετροι για μια δικτυακή υπηρεσία; Ας υποθέσουμε ότι τροποποιούμε την συνάρτηση getOrderStatus, έτσι ώστε να λαμβάνει μία επιπλέον παράμετρο, έστω την poDate που προσδιορίζει την ημερομηνία που δόθηκε η παραγγελία. Θα πρέπει να περιλαμβάνεται ένα νέο στοιχείο part στο τμήμα message που να περιγράφει την αφηρημένη λογική ενός μηνύματος εισόδου στο έγγραφο WSDL:

```
<definitions name ="poService"
        xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns="http://schemas.xmlsoap.org/wsdl/"
        targetNamespace=
                "http://localhost/WebServices/ch1/poService">
  <message name="getOrderStatusInput">
      <part name="poNumber" element="xsd:string"/>
      <part name="poDate" element="xsd:string"/>
    </message>
    <message name="getOrderStatusOutput">
      <part name="body" element="xsd:string"/>
    </message>
...
</definitions>
```

Τώρα ένα SOAP Μήνυμα που θα εκδίδεται από μία δικτυακή υπηρεσία για να πάρει το αποτέλεσμα της απομακρυσμένης συνάρτησης getOrderStatus θα είναι όπως φαίνεται παρακάτω:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
```

```
<SOAP-ENV:Body>
 <SOAP-ENV:getOrderStatus>
  <poNumber>US-247860</poNumber>
  <poDate>21-jan-07</poDate>
 </SOAP-ENV:getOrderStatus>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Εάν θέλουμε να στείλουμε και να πάρουμε πιο πολύπλοκες παραμέτρους οφείλουμε να χρησιμοποιήσουμε το στοιχείο types και να τροποποιήσουμε το αρχείο μας ως εξής:

```
<?xml version="1.0" encoding="utf-8"?>
<definitions name ="poService"
        xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:xsd1="http://localhost/WebServices/schema/"
        xmlns="http://schemas.xmlsoap.org/wsdl/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace=
                "http://localhost/WebServices/ch1/po.wsdl">
 <types>
  <xsd:schema
   targetNamespace="http://localhost/WebServices/schema/">
   <xsd:element name="poInfo">
    <xsd:complexType>
     <xsd:sequence>
      <xsd:element name="pono" type="xsd:string" />
      <xsd:element name="shippingDate" type="xsd:string" />
      <xsd:element name="status" type="xsd:string" />
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
```

```
    </xsd:schema>
   </types>
  <message name="getOrderStatusInput">
      <part name="poNumber" element="xsd:string"/>
      <part name="poDate" element="xsd:string"/>
   </message>
   <message name="getOrderStatusOutput">
      <part name="poStatus" element="xsd1:poInfo"/>
   </message>
...
</definitions>
```

Σε αυτή την περίπτωση, ένα μήνυμα απάντησης που θα μας σταλεί από την υπηρεσία σε μία υπηρεσία που ζήτησε απάντηση, μπορεί αν είναι σαν το κάτωθι:

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Body>
  <SOAP-ENV:getOrderStatusResponse>
   <poStatus>
    <pono>US-247860</pono>
    <shippingDate>21-jan-07</shippingDate>
    <status>Shipped</status>
   </poStatus>
   </SOAP-ENV:getOrderStatusResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Αν και το παράδειγμα μας δείχνει πώς να ορίζουμε XML Schema types μέσα στο τμήμα types ενός WSDL αρχείου, μπορεί να υπάρξει μεγαλύτερη επαναχρησιμοποίηση και περισσότερο modularity βάζοντας τους τύπους που

επαναχρησιμοποιούνται και από άλλες υπηρεσίες σε ένα xsd αρχείο. Το xsd αρχείο του παραδείγματος θα είναι:

```
<?xml version="1.0"?>
 <schema targetNamespace="http://localhost/WebServices/schema/"
                    xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="poInfo">
    <complexType>
     <sequence>
       <element name="pono" type="string" />
       <element name="shippingDate" type="string" />
       <element name="status" type="string" />
     </sequence>
    </complexType>
   </element>
 </schema>
</schema>
```

Οπότε, αν αυτό το αρχείο είναι τοποθετημένο στο http://localhost/WebServices/schema/po.xsd, τότε το αρχείο wsdl τροποποιείται κατάλληλα (αφαιρώντας το τμήμα types και εισάγοντας το αρχείο μέσω του tag import).

```
<?xml version="1.0" encoding="utf-8"?>
<definitions name ="poService"
          xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns:xsd1="http://localhost/WebServices/schema/"
          xmlns="http://schemas.xmlsoap.org/wsdl/"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema"
          targetNamespace=
                  "http://localhost/WebServices/ch1/po.wsdl">
  <import namespace="http://localhost/WebServices/schema/"
       location="http://localhost/WebServices/schema/po.xsd"/>
  <message name="getOrderStatusInput">
      <part name="poNumber" element="xsd:string"/>
      <part name="poDate" element="xsd:string"/>
```

```
    </message>
    <message name="getOrderStatusOutput">
      <part name="poStatus" element="xsd1:poInfo"/>
    </message>
  ...
</definitions>
```

2. Αρχιτεκτονικές

2.1 Υπηρεσιοστραφής Αρχιτεκτονική (Service Oriented Architecture - SOA)

2.1.1 Γενικά

Όπως έχει ήδη σημειωθεί, για να σχεδιαστεί και υλοποιηθεί μία υπηρεσιοστραφής αρχιτεκτονική (Service Oriented Architecture – SOA) βασισμένη σε δικτυακές υπηρεσίες, χρειάζεται να ακολουθηθούν τα υπηρεσιοστραφή αξιώματα όταν συνδέονται οι υπηρεσίες προκειμένου να υλοποιηθεί μία εφαρμογή. Εδώ παρουσιάζονται συνοπτικά βασικά αξιώματα της υπηρεσιοστραφούς αρχιτεκτονικής που χρησιμεύουν σε υλοποίηση SOA εφαρμογών:

- **Χαλαρή σύζευξη** (Loose coupling) αντιπροσωπεύει μία σχέση που επιτρέπει στην υποκείμενη λογική μίας υπηρεσίας να αλλάζει με ελάχιστο ή και καθόλου αντίκτυπο στις υπόλοιπες υπηρεσίες που χρησιμοποιούνται στην ίδια SOA. Η χαλαρή σύζευξη είναι το κεντρικό αξίωμα της υπηρεσιοστραφούς αρχιτεκτονικής. Υλοποιώντας υπηρεσίες όσο το δυνατόν πιο χαλαρά συζευγμένες επιτρέπει τον σχεδιαστή και τον χρήστη να τηρεί τις προδιαγραφές και για τις υπόλοιπα αξιώματα της υπηρεσιοστραφούς αρχιτεκτονικής, όπως είναι η επαναχρησιμοποίηση της υπηρεσίας, την αυτονομία της υπηρεσίας και την αδυναμία μνήμης προηγούμενης κατάστασης της υπηρεσίας (statelessness).

- **Το συμβόλαιο της υπηρεσίας** αντιπροσωπεύει περιγραφές της υπηρεσίας και άλλα αρχεία που περιγράφουν πώς μία υπηρεσία μπορεί προγραμματιστικά να κληθεί. Αυτό μπορεί να γίνει κυρίως από ένα αρχείο wsdl (μαζί με ενδεχόμενα xsd αρχεία), που περιγράφει μία υπηρεσία, ορίζοντας το συμβόλαιο για αυτή την υπηρεσία.

- **Απόκρυψη της υποκείμενης λογικής** σημαίνει ότι μία υπηρεσία εκθέτει δημόσια μόνο την λογική που περιγράφεται στο συμβόλαιο της υπηρεσίας, αποκρύπτοντας τις λεπτομέρειες υλοποίησης από τους πελάτες της υπηρεσίας. Αυτό σημαίνει ότι οι υπηρεσίες αλληλεπιδρούν η μία με την άλλη μόνο διαμέσω των δημόσιων διεπαφών τους. Το αρχείο wsdl, έχει τέτοιο τμήμα που δίνει τις διεπαφές μιας υπηρεσίας.

- **Αυτονομία** σημαίνει ότι οι υπηρεσίες ελέγχουν μόνο την λογική που εγκολπώνονται. Διαιρώντας την λογική των εφαρμογών σε ένα σετ από αυτόνομες υπηρεσίες, επιτρέπει στον σχεδιαστή να φτιάξει ευέλικτες SOA λύσεις για τα διάφορα εγχειρήματα, κατορθώνοντας ταυτόχρονα χαλαρή σύζευξη, επαναχρησιμοποίηση και σύνθεσης.

- **Η επαναχρησιμοποίηση** σε υπηρεσιοστραφή αρχιτεκτονική επιτυγχάνεται με τον διαμοιρασμό της λογικής της εφαρμογής σε διάφορες υπηρεσίες έτσι, ώστε κάθε υπηρεσία να μπορεί δυνητικά να χρησιμοποιηθεί από περισσότερους του ενός αιτούντος την υπηρεσία. Σχεδιάζοντας και κατασκευάζοντας επαναχρησιμοποιούμενες υπηρεσίες υποστηρίζει το αξίωμα της σύνθεσης.

- **Η σύνθεση** αντιπροσωπεύει την δυνατότητα των υπηρεσιών να ομαδοποιούνται σε σύνθετες υπηρεσίες που συντονίζουν μία ανταλλαγή δεδομένων μεταξύ των υπηρεσιών που συναθροίζονται. Παραδείγματος χάριν, χρησιμοποιώντας μία γλώσσα ενορχήστρωσης, όπως η WS-BPEL, επιτρέπει να συντεθούν λεπτά σχεδιασμένες υπηρεσίες σε πιο χοντρά σχεδιασμένες υπηρεσίες. Η WS-BPEL θα συζητηθεί παρακάτω.

- **Η απουσία μνήμης προηγούμενης κατάστασης της υπηρεσίας** (statelessness) σημαίνει οι υπηρεσίες δεν κρατούν την κατάστασή τους σε μία συγκεκριμένη διεργασία. Η κατασκευή τέτοιων υπηρεσιών ενθαρρύνει την χαλαρή σύζευξη, την επαναχρησιμοποίηση και την σύνθεση. Βέβαια, κάτι τέτοιο δεν είναι πάντοτε επιθυμητό, όπως σε εφαρμογές πλέγματος (grid).

- **Η διαλειτουργικότητα** μεταξύ των υπηρεσιών εύκολα επιτυγχάνεται αρκεί οι υπηρεσίες να αλληλεπιδρούν διαμέσου των διεπαφών οι οποίες είναι ανεξάρτητες των γλωσσών υλοποίησης καθώς και της πλατφόρμας.

- **Η ανιχνευσιμότητα** αναφέρεται σε standard μηχανισμούς που επιτρέπουν σε υπηρεσίες να ανακαλύπτονται από διάφορους αιτούντες των υπηρεσιών τους. Η UDDI (Universal Description, Discovery, and Integration) προδιαγραφή παρέχει τέτοιο μηχανισμό, ο οποίος επιτρέπει την δημοσιοποίηση των descriptors υπηρεσιών σε μία XML-βασισμένη αποθήκη και ως εκ τούτου κάνοντας τες διαθέσιμες στο κοινό.

Όπως εύκολα μπορεί να παρατηρηθεί, τα περισσότερα από αυτά τα αξιώματα είναι στενά συνδεδεμένα. Για παράδειγμα, εάν κάποιος έχει κατά νου την αυτονομία όταν διαιρεί την λογική μιας εφαρμογής για να φτιάξει υπηρεσίες που θα χρησιμοποιηθούν σε SOA λύσεις, θα έχει σίγουρα επαναχρησιμοποιούμενα, σύνθετα και χαλαρά συζευγμένα κομμάτια κώδικα λογισμικού, τα οποία θα μπορεί να τα ξαναχρησιμοποιήσει σε μελλοντικά του σχέδια.

Από την άλλη, αγνοώντας έστω και ένα αξίωμα της υπηρεσιοστραφούς αρχιτεκτονικής δυσχεραίνει το γεγονός να κρατηθούν όλα τα υπόλοιπα. Παραδείγματος χάριν, εάν αγνοηθεί το αξίωμα της απουσίας μνήμης προηγούμενης κατάστασης όταν σχεδιάζονται οι υπηρεσίες, υπάρχει σοβαρό ενδεχόμενο να καταλήξει ο σχεδιαστής με λιγότερο επαναχρησιμοποιούμενα και λιγότερο συνθέσιμα κομμάτια κώδικα για τις SOA λύσεις του.

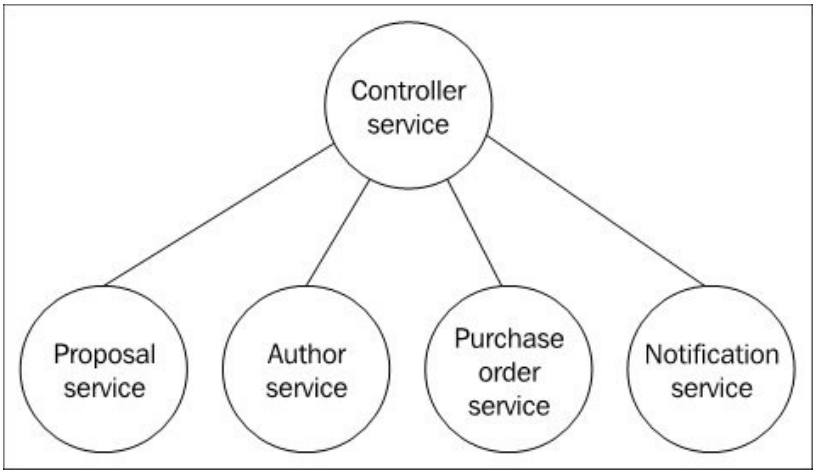
2.1.2 SOA Συνθέσεις (SOA Compositions)

Άμα τη δημιουργία όλων των υπηρεσιών που απαιτούνται για να αυτοματοποιηθεί η διαδικασία της κατάθεσης προτάσεων, ο σχεδιαστής χρειάζεται να γνωρίζει πώς θα τα θέσει αυτά σε δράση.

Βασικά, υπάρχουν πληθώρα τρόπων που μπορούν οι υπηρεσίες να οργανωθούν για να συντεθεί μία λύση SOA. Παραδείγματος χάριν, μπορεί να δημιουργηθεί μία σύνθετη υπηρεσία ως ένα php script που θα "εκτίθεται" ως δικτυακή υπηρεσία και που, προγραμματιστικά, καλεί άλλες υπηρεσίες όπως οφείλει. Εντούτοις, ο πιο συνηθισμένος τρόπος να σχεδιαστούν και να συντεθούν SOA λύσεις είναι χρησιμοποιώντας WS-BPEL, μία γλώσσα ενορχήστρωσης που επιτρέπει στον σχεδιαστή να φτιάξει ενορχηστρώσεις – σύνθετες και ελεγχόμενες υπηρεσίες ορίζοντας το πώς οι υπηρεσίες που καλούνται θα διαδράσουν προκειμένου να επιτευχθεί το επιθυμητό αποτέλεσμα.

2.1.2.1 Ενορχήστρωση

Μία ενορχήστρωση συγκεντρώνει υπηρεσίες σε μία επιχειρησιακή εκτελέσιμη διαδικασία που χρειάζεται να εκτελεσθεί από μία μηχανή ενορχήστρωσης. Σχηματικά, μία ενορχήστρωση μπορεί να ομοιάζει με το παρακάτω:



Εικόνα 5: Η controller service αναλαμβάνει χρέη ενορχηστρωτή.

Όπως διαφαίνεται από την παραπάνω εικόνα, ένας ενορχηστρωτής (controller service) αναλαμβάνει να συντονίσει, βάσει της λογικής με την οποία έχει σχεδιαστεί, διάφορες υπηρεσίες. Αυτός ο ενορχηστρωτής μπορεί να είναι μία WS-BPEL επιχειρησιακή διεργασία, η οποία όταν εκτελείται σε μία μηχανή ενορχήστρωσης ολοκληρώνει μία συγκεκριμένη επιχειρηματική εργασία.

Αξίζει εδώ να σημειωθεί ότι, εάν μία υπηρεσία συντονισμού έχει υλοποιηθεί βάσει μίας WS-BPEL, τότε θα πρέπει να γνωρίζει τα αντίστοιχα WSDL αρχεία που θα χρησιμοποιεί προκειμένου να καλεί τις αντίστοιχες υπηρεσίες, όπως και επίσης και η ίδια η υπηρεσία του controller να έχει ένα wsdl αρχείο για να μπορεί να γίνεται γνωστή η υπηρεσία που προσφέρει στο κοινό.

Φυσικά, αναδρομικά, μπορεί μία ενορχηστρωμένη υπηρεσία να αποτελέσει κομμάτι μίας μεγαλύτερης ενορχήστρωσης.

2.1.2.2 Χορογραφία

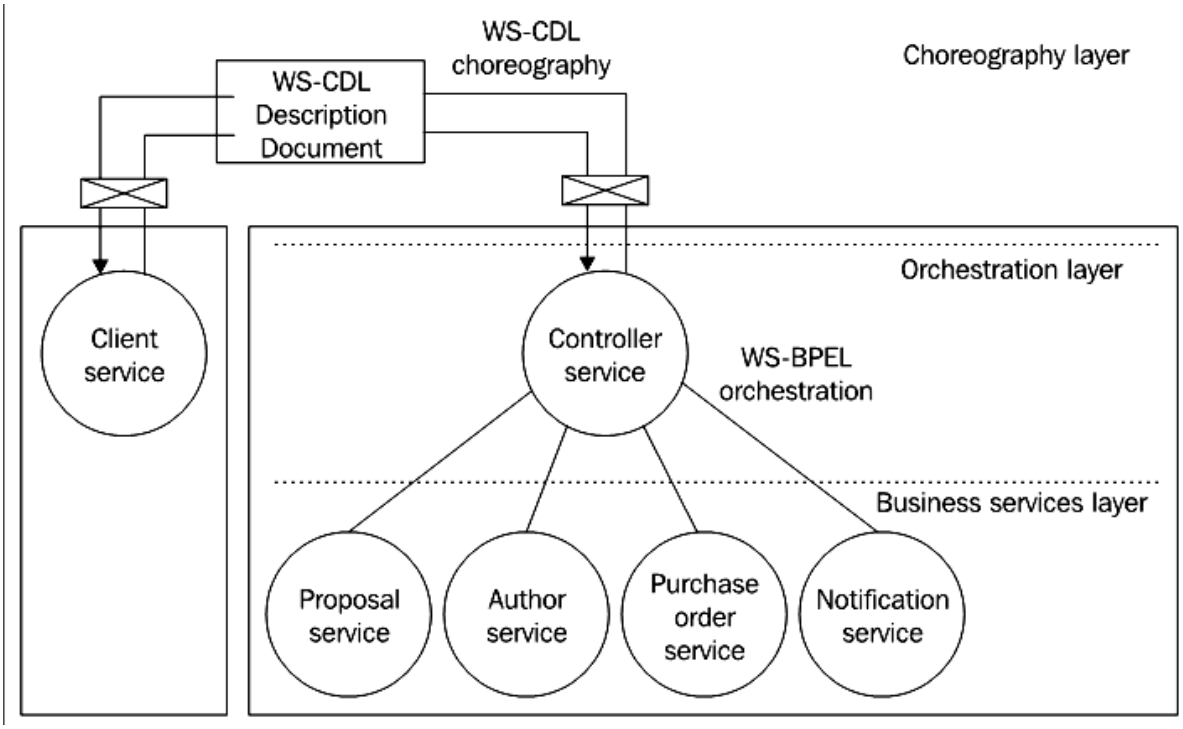
Η προδιαγραφή χορογραφίας δικτυακών υπηρεσιών μαζί με την αντίστοιχη γλώσσα περιγραφής χορογραφίας δικτυακών υπηρεσιών (Web Services Choreography Description Language - WS-CDL) παρέχει έναν επιπλέον τρόπο για την σχεδίαση SOA συνθέσεων. Ενώ η WS-BPEL χρησιμοποιείται για να ενορχηστρώσει υπηρεσίες σε σύνθετες λύσεις που συνήθως εκφράζουν -συγκεκριμένα σε επιχειρήσεις- διαγράμματα επιχειρηματικών διαδικασιών, η WS-CDL επιτρέπει στον σχεδιαστή να

περιγράψει ομότιμες (peer-to-peer) σχέσεις μεταξύ των υπηρεσιών Web και / ή με άλλους συμμετέχοντες εντός ή πέρα από τα όρια εμπιστοσύνης.

Σε αντίθεση με την ενορχήστρωση, η χορογραφία δεν έχει ένα συγκεντρωτικό μηχανισμό ελέγχου, με την προϋπόθεση ο έλεγχος να κατανέμεται μεταξύ των συμμετεχόντων αλληλεπίδρασης. Αυτό σημαίνει ότι μια ενορχήστρωση αντιπροσωπεύει μία εκτελέσιμη διαδικασία που πρέπει να εκτελεστεί από έναν μηχανισμό ενορχήστρωσης σε ένα μέρος, ενώ η χορογραφία στην ουσία αποτελεί μια περιγραφή του τρόπου διανομής του ελέγχου μεταξύ των συνεργαζομένων συμμετεχόντων, δίχως να χρησιμοποιείται ένας κεντρικός συντονιστής με σκοπό την ολοκλήρωση της εργασίας.

Για να οριστεί σωστά η χορογραφία, πρέπει ο σχεδιαστής να δημιουργήσει ένα WS-CDL περιγραφικό έγγραφο της χορογραφίας που θα χρησιμεύσει ως το συμβόλαιο/σύμβαση μεταξύ των συμμετεχόντων αλληλεπίδρασης. Συγκεκριμένα, ένα WS-CDL έγγραφο περιγράφει τις ανταλλαγές μηνυμάτων μεταξύ των συνεργαζομένων συμμετεχόντων, ορίζοντας πώς αυτοί οι συμμετέχοντες πρέπει να χρησιμοποιούνται μαζί για την επίτευξη ενός κοινού επιχειρηματικού στόχου. Για παράδειγμα, μπορεί να υπάρξει μια χορογραφία που επιτρέπει τη συνεργασία μεταξύ ενός ενορχήστρωση, μια υπηρεσία-ελεγκτής που αντιπροσωπεύει μία WS-BPEL διαδικασία, και μία υπηρεσία-πελάτης που αλληλεπιδρά με την εν λόγω υπηρεσία-ελεγκτή.

Σχηματικά, αυτό μπορεί να αναπαρασταθεί στην παρακάτω εικόνα:



Στο σενάριο που απεικονίζεται στο σχήμα, το στρώμα της χορογραφία χρησιμοποιείται για να καθορίσει την ομότιμη (peer-to-peer) συνεργασία των δύο υπηρεσιών. Ειδικότερα, το WS-CDL έγγραφο της χορογραφίας περιγράφει την ανταλλαγή μηνυμάτων μεταξύ μίας σύνθετης υπηρεσίας που συζητήθηκε στην προηγούμενη παράγραφο και μίας από τους πελάτες της.

2.2 Πλεγματικά συστήματα (Grid & cloud computing)

2.2.1 Γενικά

Υπάρχουν φορές όπου ένας υπολογιστής ή τουλάχιστον ένα εταιρικό δίκτυο υπολογιστών δεν επαρκεί για τις υπολογιστικές (grid) ή και αποθηκευτικές ανάγκες της επιχείρησης (cloud). Το "πλέγμα" (grid) κατορθώνει ακριβώς αυτό: επιτυγχάνει μεγαλύτερη απόδοση συγκεντρώνοντας πόρους από διαφορετικές τοποθεσίες. Δίχως το "πλέγμα", ένας οργανισμός, μία επιχείρηση, πρέπει να περιορίζεται στους πόρους τους οποίους διαθέτει φυσικά. Με το "πλέγμα", πόροι από διαφορετικούς οργανισμούς συγκεντρώνονται δυναμικά για να σχηματιστούν *εικονικοί οργανισμοί* με σκοπό την επίλυση συγκεκριμένων προβλημάτων. Η παρακάτω εικόνα δείχνει πιο παραστατικά πώς οι πόροι από τρεις ξεχωριστούς 'πραγματικούς' οργανισμούς μπορούν να συνδυαστούν σε δύο εικονικούς οργανισμούς. Περαιτέρω, αν και δεν φαίνεται στην εικόνα, απλοί *χρήστες* μπορούν να σχηματίσουν εικονικούς οργανισμούς, όπως γίνεται σε διάφορα projects ανά την υφήλιο (βλ. folding@home, boinc κτλ.)

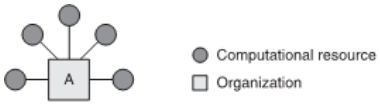


Figure 1.2: An organization and its resources

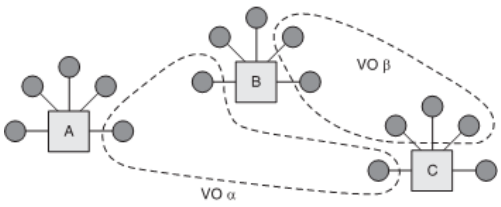


Figure 1.3: Virtual organizations

Εννοιολογικά, όλα αυτά μπορεί να φαίνονται σαν μια απλή σκέψη. Στην πραγματικότητα, "η χρήση πόρων από πολλούς οργανισμούς" μοιάζει σαν ένα πολύ έξυπνο τρόπο δράσης όταν ένα πρόβλημα δεν μπορεί να επιλυθεί με τη χρησιμοποίηση των υπολογιστικών πόρων ενός μεμονωμένου οργανισμού. Ωστόσο, στην πράξη, αυτό δεν είναι καθόλου απλό. Για παράδειγμα, ρωτήστε τον εαυτό σας τα εξής:

- Πώς θα αποφασίσουμε ποιοι πόροι είναι μέρος της κάθε εικονικού οργανισμού;

- Με δεδομένη μια υπολογιστική εργασία, πώς θα αποφασιστεί ποιοι πόροι θα διατεθούν για την περάτωση του έργο; Για πόσο καιρό; Σίγουρα, δεν θέλουμε να δώσουμε σε άλλους οργανισμούς απρόσκοπτη πρόσβαση σε πόρους μας!

- Πώς θα κάνουμε τους πόρους να επικοινωνούν μεταξύ τους; Πρέπει να ληφθεί υπόψη ότι αυτοί οι ετερογενείς πόροι προέρχονται από διάφορους οργανισμούς!

- Αν θέλουμε να "χωριστεί" ένα έργο, ώστε να μπορεί να εκτελεστεί παράλληλα από διάφορους υπολογιστές σε διαφορετικούς οργανισμούς, με ποιον τρόπο θα πρέπει πραγματικά να "κατατμηθεί" το πρόγραμμα; Και πάλι, μας απασχολούν οι ετερογενείς πόροι: κάθε οργανισμός μπορεί να χρησιμοποιεί ένα διαφορετικό λειτουργικό σύστημα, ή ακόμη και μια διαφορετική αρχιτεκτονική υπολογιστή.

- Ο διαμοιρασμός των πόρων με άλλους οργανισμούς μπορεί να ακούγεται σαν μια πραγματικά ιδανική σκέψη, αλλά ενέχει πολλές προκλήσεις ασφαλείας. Για παράδειγμα, πώς μπορεί ένας οργανισμός να είναι σίγουρος ότι οι πόροι του θα χρησιμοποιούνται μόνο από έμπιστους χρήστες και ότι δεν καταχρώνται από κακόβουλους χρήστες;

Έτσι, θα μπορούσαμε να πούμε ότι το Grid computing, σε γενικές γραμμές, μας δίνει τη δυνατότητα πρόσβασης σε ετερογενείς πόρους που προέρχονται από διάφορους οργανισμούς, παρέχοντας ένα σύνολο πρωτοκόλλων, τεχνολογιών, και μεθοδολογιών που παρέχουν μια απάντηση στα ερωτήματα αυτά. Προτού μελετήσουμε περισσότερο τη γενική αρχιτεκτονική των συστημάτων Grid, ας ρίξουμε μια ματιά σε ένα πιο περιεκτικό ορισμό του Grid Computing.

2.2.2 Ορισμός

Αν και έχουν δοθεί πάρα πολλοί ορισμοί σχετικά με το Grid computing, εμείς θα χρησιμοποιήσουμε τον ορισμό του Ian Foster από το βιβλίο του *What is the Grid? A Three Point Checklist*[4]. Αν και ο ίδιος ο συγγραφέας εξηγεί ότι η λίστα αφήνει χώρο για συζήτηση, ωστόσο παρέχει έναν συνεπή και ακριβή ορισμό του grid. Παραθέτοντας, λέει:

4 I. Foster. “What is the Grid? A Three Point Checklist”.GRIDToday, July 20, 2002.

“Πλέγμα είναι ένα σύστημα που

1. συντονίζει τους πόρους που δεν υπόκεινται σε κεντρικό έλεγχο

2. χρησιμοποιώντας τυποποιημένα, ανοικτού και γενικού σκοπού πρωτόκολλα και διεπαφές

3. **για να παραδώσει σύνθετες ποιοτικές υπηρεσίες.**

Ας ρίξουμε μια πιο προσεκτική ματιά στα δύο πρώτα σημεία:

1. συντονίζει τους πόρους που δεν υπόκεινται σε κεντρικό έλεγχο

Αυτό παραπέμπει στην έννοια του “χρησιμοποιώντας πόρους από διάφορους οργανισμούς” που έχουμε αναφέρει νωρίτερα. Για να είμαστε πιο συνεπείς, στην πραγματικότητα, αναφερόμαστε στην “ενσωμάτωση και τον συντονισμό των πόρων και των χρηστών που ζουν σε διαφορετικούς τομείς ελέγχου(control domains)”. Έχουμε μιλήσει για διαμοιρασμό πόρων διάφορων οργανισμών, αλλά αυτοί οι τομείς ελέγχου θα μπορούσε επίσης να είναι διαφορετικές διοικητικές μονάδες μέσα σε μια εταιρεία. Είναι ακριβώς τότε, όταν έχουμε να αντιμετωπίσουμε αυτούς τους διάφορους τομείς ελέγχου, ότι θα προκύψουν όλα τα ζητήματα που προαναφέρθηκαν. Διαφορετικά, θα πρέπει να κάνουμε με ένα πρόβλημα που μπορεί να επιλυθεί χρησιμοποιώντας *τοπικές λύσεις διαχείρισης*, που δεν χρησιμοποιούν πλεγματικό σύστημα.

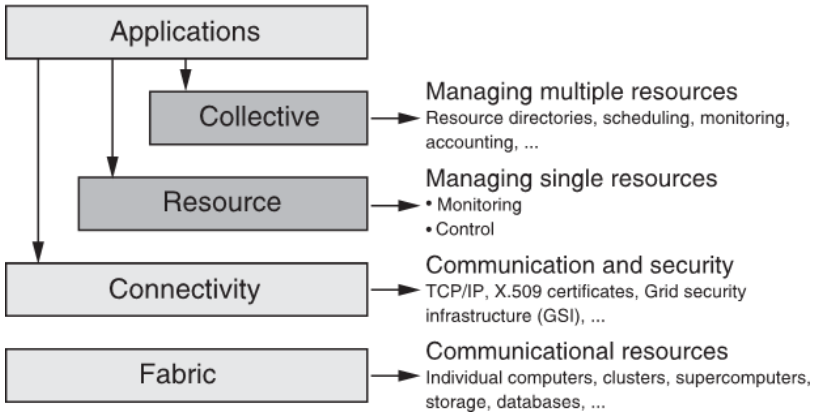
2. χρησιμοποιώντας τυποποιημένα, ανοικτού και γενικού σκοπού πρωτόκολλα και διεπαφές

Έχουμε αναφέρει ότι οι πόροι σε ένα Grid είναι ανομοιογενείς. Σε ένα σύστημα πλέγματος, πρέπει με κάποιο τρόπο πρέπει να λάβουμε το σύνολο των διαφορετικών πόρων (ανεξάρτητοι υπολογιστές, ομάδες/clusters, υπερυπολογιστές, αποθηκευτικοί χώροι, κ.λπ.) που χρησιμοποιούν μια μεγάλη ποικιλία λειτουργικών συστημάτων και αρχιτεκτονικών, και με κάποιο τρόπο να τους κάνουμε να συνεργάζονται για την επίλυση υπολογιστικών καθηκόντων. Ο μόνος τρόπος για να γίνει αυτό είναι με τη χρήση "πρότυπων, ανοικτών, γενικού σκοπού πρωτοκόλλων και διεπαφών" που παρέχουν μια «κοινή γλώσσα» που ο καθένας στο Grid μπορεί να καταλάβει. Τα

πρωτόκολλα επιτρέπουν την ανομοιογένεια, και, το γεγονός ότι αυτά βασίζονται σε πρότυπα, προωθείται η διαλειτουργικότητα.

2.2.3 Η αρχιτεκτονική πλέγματος (Grid Architecture)

Η δημιουργία ενός πλήρους συστήματος Grid απαιτεί μια ευρεία ποικιλία πρωτοκόλλων, υπηρεσιών, και κιτ ανάπτυξης λογισμικού. Στο ορόσημο κείμενο των Foster, Kesselman και Tuecke, *Anatomy of the Grid*[5], καθορίζεται μία γενική αρχιτεκτονική Grid στην οποία όλα τα διάφορα στοιχεία που απαρτίζουν ένα σύστημα πλέγματος (Grid) κατηγοριοποιούνται ανάλογα με τη λειτουργία και τον σκοπό τους. Η αρχιτεκτονική Grid μπορεί να εκφραστεί βάσει ενός πολυεπίπεδου διαγράμματος. Θα περιγραφεί κάθε στρώμα από κάτω προς τα πάνω.



1.Fabric (Υφανση)

Αυτό το στρώμα αφορά τους πόρους που στην πραγματικότητα πρόκειται να κατανεμηθούν στο Grid σύστημά μας, όπως μεμονωμένοι υπολογιστές, clusters, υπερυπολογιστές, αποθηκευτικοί χώροι δικτύου, βάσεις δεδομένων, κλπ.

2.Connectivity (Συνδεσιμότητα)

Φυσικά, οι πόροι μας πρέπει να είναι σε θέση να επικοινωνούν μεταξύ τους. Το στρώμα συνδεσιμότητας αναφέρεται σε όλα τα πρωτόκολλα που θα επιτρέψουν στους πόρους μας να επικοινωνούν. Θεμελιώδη πρωτόκολλα Internet, όπως το TCP / IP, HTTP, DNS, κλπ. εμπίπτουν σε αυτό το στρώμα. Ωστόσο, τα συστήματα Grid επίσης

5 I. Foster, C. Kesselman, and S. Tuecke. "The Anatomy of the Grid: Enabling Scalable Virtual Organizations".International J. Supercomputer Applications15(3), 2001

απαιτούν *ασφαλή* επικοινωνία, έτσι αυτό το στρώμα περιλαμβάνει επίσης πολλά πρωτόκολλα που έχουν προκύψει από την κοινότητα Grid σε απάντηση των ιδιόρρυθμων προκλήσεων ασφαλείας που αντιμετωπίζουν τα Grid συστήματα.

3.Resource (Πόροι)

Αυτό το στρώμα αναφέρεται σε όλες τις υπηρεσίες και τα πρωτόκολλα που θα μας επιτρέπουν να διαχειριζόμαστε τους ανεξάρτητους πόρους. Αυτή η διαχείριση μπορεί να περιλαμβάνει εργασίες όπως η αρχικοποίηση των πόρων, την παρακολούθησή τους, και την μέτρησή τους. Σε αυτό το στρώμα, δεν ασχολούνται με τις παγκόσμιες αλληλεπιδράσεις μεταξύ των πόρων στο Grid, μόνο με ανεξάρτητους πόρους. Συγκεκριμένα, υπάρχουν δύο κύριες κατηγορίες των πρωτοκόλλων σε αυτό το επίπεδο:

- Πρωτόκολλα πληροφοριών: Τα πρωτόκολλα αυτά θα μας επιτρέψουν να έχουν πρόσβαση σε πληροφορίες ενός πόρου. Για παράδειγμα, στην περίπτωση ενός υπολογιστή, ίσως να μας ενδιαφέρει να μάθουμε το CPU load, την διαθέσιμη μνήμη του, τον αριθμός των διαδικασιών λειτουργίας σε αυτό, κλπ.

- Πρωτόκολλα διαχείρισης: Τα πρωτόκολλα αυτά στην πραγματικότητα μας επιτρέπουν να ελέγχουμε τον πόρο, σε κάποιο βαθμό. Σε ένα σύστημα πλέγματος, εμείς δεν θα έχουμε συνήθως πλήρη έλεγχο πάνω σε έναν πόρο. Συνεπώς, αυτές οι υπηρεσίες και τα πρωτόκολλα γενικά θα είναι υπεύθυνα να ελέγχουν ότι το είδος των υπηρεσιών που θα ζητήσουμε σε έναν πόρο είναι συνεπείς με τις πολιτικές ενός οργανισμού.

4 Collective (Συλλογικό)

Αυτό το στρώμα περιλαμβάνει τις υπηρεσίες και τα πρωτόκολλα που ασχολούνται με τη διαχείριση πολλαπλών πόρων. Με βάση τις υπηρεσίες που παρέχονται στο στρώμα των πόρων (resource), αυτό το στρώμα θα μας επιτρέψει να λαμβάνουμε πράγματι ένα σύνολο πόρων και να τους κάνουμε να συνεργάζονται για την επίλυση ένα κοινού έργου. Τα παρακάτω είναι μόνο ένα δείγμα από το είδος των υπηρεσιών που βρίσκονται συνήθως σε αυτό το στρώμα:

- Μητρώα πόρων (Resource registries): Μας επιτρέπουν να ανακαλύψουμε μέσα σε έναν εικονικό οργανισμό πόρους και να αιτηθούμε ιδιότητές τους.

- Υπηρεσίες κατανομής και δρομολογητών: Όταν θέλουμε να εκτελέσουμε ένα πρόγραμμα σε ένα σύστημα πλέγματος, εμείς γενικά δεν πρέπει να αποφασίσουμε πού ακριβώς αυτό θα τρέξει στο δίκτυο. Μία υπηρεσία κατανομής θα χρησιμοποιήσει έναν κατάλογο των πόρων για να ανακαλύψει έναν πόρο (ή πόρους) κατάλληλα για το πρόγραμμά μας (που συνήθως ονομάζεται *εργασία*), και θα διαθέσει πόρους για τη εργασία μας. Μια υπηρεσία προγραμματισμού θα αποφασίσει ποιες εργασίας πραγματικά τρέχουν πού και πότε.

- Υπηρεσίες παρακολούθησης: Μας επιτρέπουν να παρακολουθούμε ότι όλοι οι πόροι μας δουλεύουν κανονικά.

- Υπηρεσίες διαχείρισης δεδομένων: Οι εργασίες μας θα απαιτούν γενικά σύνολα δεδομένων για να εργαστούν. Οι υπηρεσίες διαχείρισης δεδομένων παρακολουθούν αυτές τις σειρές δεδομένων και τις εναλλαγές δεδομένων με τον πόρο που τους χρειάζεται.

5.Εφαρμογές

Αυτό το στρώμα αναφέρεται στις πραγματικές εφαρμογές που θα τρέχουν σε ένα σύστημα πλέγματος. Να σημειωθεί ότι οι εφαρμογές αυτές δεν χρειάζεται να αλληλεπιδρούν άμεσα με το υπηρεσίες συλλογικού (collective services). Είναι επίσης ελεύθερες να έχουν πρόσβαση στο κομμάτι των Πόρων(resource) και των υπηρεσιών συνδεσιμότητας (Connectivity) κάθε φορά που απαιτείται.

2.2.4 Παραδείγματα Grid Συστημάτων

- EGEE: Enabling Grids for E-Science στην Ευρώπη (http://public.eu-egee.org/): Ένα φιλόδοξο σχέδιο πλέγματος που θα δώσουν επιστήμονες πρόσβαση σε

υπολογιστικούς πόρους σε 27 χώρες. Το EGEE θα είναι επίσης υπεύθυνο για την παροχή της τρομερής υπολογιστικής ισχύος που απαιτείται από το LHC του CERN

- NEESit (http://it.nees.org/): Παρέχει μια εκτεταμένη υποδομή για την NEES (Δίκτυο για την Προσομοίωσης Αντισεισμικής Μηχανικής) συμμαχίας, με τη διασύνδεση ερευνητικών κέντρων σεισμών στις ΗΠΑ.

- TeraGrid (http://www.teragrid.org/): Ένα σύστημα Grid που παρέχει μια ισχυρή υποδομή για την ανοικτή επιστημονική έρευνα. Το 2004, το TeraGrid είχε 20 teraflops υπολογιστικής ισχύος και 1 Petabyte δυνατότητα αποθήκευσης.

- eDiaMoND (http://www.ediamond.ox.ac.uk/): Το eDiaMoND έργο είναι ένα παράδειγμα του πώς το Grid computing μπορεί να χρησιμοποιηθεί για την ηλεκτρονική Υγεία. Αυτό το έργο "μαζεύει και διανέμει πληροφορίες για τη θεραπεία του καρκίνου του μαστού, δίνει τη δυνατότητα έγκαιρης ανίχνευσης και διάγνωσης, καθώς και παρέχει εργαλεία και πληροφορίες για τη θεραπεία της ασθένειας".

3. Διαγράμματα ροής δεδομένων

3.1. Γενικά

3.1.1 Ορισμός

Μια ροή εργασίας αποτελείται από μια σειρά συνδεδεμένων βημάτων. Πρόκειται για μια παράσταση μιας ακολουθίας πράξεων, που δηλώθηκαν ως εργασία ενός προσώπου, μιας ομάδας ατόμων[6], μια οργάνωσης με προσωπικό, ή γενικά ενός ή περισσότερων απλών ή σύνθετων μηχανισμών. Ως ροή εργασίας μπορεί να θεωρηθεί οποιαδήποτε αφηρημένη θεώρηση μιας πραγματικής εργασίας, που διαχωρίζεται σε επιμέρους εργασίες, σε τμήματα εργασίας ή σε άλλα είδη της ταξινόμησης(ή επιμέρισης). Για ελεγκτικούς σκοπούς, η ροή εργασίας μπορεί να είναι μια άποψη, μια οπτική γωνία για την πραγματική εργασία σε μία συγκεκριμένη, επιλεγμένη πτυχή[7], ούσα ως εκ τούτου μία εικονική αναπαράσταση της πραγματικής εργασίας. Η ροή που περιγράφεται, συχνά αναφέρεται ως *έγγραφο* που μεταφέρεται από το ένα στάδιο στο άλλο.

Μια ροή εργασίας είναι ένα μοντέλο για να αντιπροσωπεύει μία πραγματική δουλειά για περαιτέρω εκτίμηση, παραδείγματος χάριν, για την περιγραφή μίας αξιόπιστης επαναλαμβανόμενης αλληλουχίας ενεργειών. Πιο αφηρημένα, μια ροή εργασίας είναι ένα μοτίβο δραστηριότητας που έχει προκύψει με τη συστηματική οργάνωση των πόρων, καθορισμένων ρόλων και μαζικών, ενεργειακών και πληροφοριακών ροών εργασίας, σε μια *διαδικασία εργασίας* που μπορεί να τεκμηριωθεί και να γίνει γνωστή.[8] [9] Οι ροές εργασίας αποσκοπούν στην επίτευξη να επεξεργαστούν κάποιου είδους προθέσεων, όπως η φυσική μεταμόρφωση, η παροχή υπηρεσιών, ή η επεξεργασία πληροφοριών.

Οι έννοιες των ροών εργασίας είναι έννοιες που συνδέονται στενά με άλλες έννοιες που χρησιμοποιούνται για να περιγράψουν οργανωτική δομή, όπως σιλό, λειτουργίες, ομάδες, σχέδια, πολιτικές και ιεραρχίες. Οι ροές εργασίας μπορεί να

6 ISO 12052:2006, http://www.iso.org/
7 ISO/TR 16044:2004, http://www.iso.org/
8 http://www.workflowpatterns.com/documentation/documents/BPM-06-22.pdf
9 ftp://ftp.informatik.uni-stuttgart.de/pub/library/medoc.ustuttgart_fi/STUD-2052/STUD-2052.pdf

θεωρηθούν ως ένα πρωταρχικό δομικό στοιχείο των οργανισμών. Οι σχέσεις μεταξύ αυτών των εννοιών περιγράφονται παρακάτω σε αυτήν την καταχώρηση.

Ο όρος "ροή εργασίας" χρησιμοποιείται στο πλαίσιο του προγραμματισμού ηλεκτρονικών υπολογιστών για να συλλάβει και να αναπτύξει τις σχέσεις ανθρώπων-μηχανών. Τα λογισμικά διαχείρισης ροών εργασίας[10] έχουν ως στόχο να παρέχουν στους τελικούς χρήστες έναν ευκολότερο τρόπο να ενορχηστρώνουν ή/και να περιγράφουν σύνθετες επεξεργασίες δεδομένων σε μια οπτική μορφή, περίπου όπως τα διαγράμματα ροής αλλά δίχως την ανάγκη να γίνουν κατανοητοί οι υπολογιστές ή ο προγραμματισμός.

3.1.2 Σχετικές έννοιες

Η έννοια της ροής εργασίας σχετίζεται στενά με πολλούς άλλους τομείς στην έρευνα λειτουργιών (operations research) και σε άλλα πεδία που μελετούν την φύση της εργασίας, είτε ποιοτικά είτε ποσοτικά, όπως η τεχνητή νοημοσύνη (ιδίως, η υποπειθαρχία του σχεδιασμού τεχνητής νοημοσύνης) και η εθνογραφία. Η ροή εργασίας είναι όρος που χρησιμοποιείται πιο συχνά σε συγκεκριμένους κλάδους, όπως στην εκτύπωση, και σε επαγγελματικούς τομείς, όπου μπορεί να έχει ιδιαίτερα εξειδικευμένες σημασίες.

1. Διεργασίες: Μια διεργασία (ενίοτε αναφέρεται και ως διαδικασία) είναι μια πιο συγκεκριμένη έννοια από τη ροή εργασίας, και μπορεί να εφαρμοστεί, για παράδειγμα, σε φυσικές ή βιολογικές διεργασίες. Στο πλαίσιο των εννοιών γύρω από την εργασία, μια διαδικασία μπορεί να διακριθεί από μια ροή εργασίας από το γεγονός ότι έχει σαφώς καθορισμένες εισροές, εκροές και σκοπούς, ενώ η έννοια της ροής εργασίας ενδέχεται να ισχύει γενικότερα για κάθε συστηματικό τρόπο διεξαγωγής των δραστηριοτήτων.

2. Σχεδιασμός και δρομολόγηση: Η σχεδίαση είναι μια περιγραφή του λογικά αναγκαίου, εν μέρει διατεταγμένου συνόλου των δραστηριοτήτων που απαιτούνται για την επίτευξη ενός συγκεκριμένου στόχου υπό ορισμένες αρχικές συνθήκες. Ένα

10 MEgha Workflow Management System, http://www3.interscience.wiley.com/journal/122522172/abstract

σχέδιο, όταν προσαυξηθεί με ένα χρονοδιάγραμμα και με υπολογισμούς της κατανομής των πόρων, καθορίζει απόλυτα ένα συγκεκριμένο στιγμιότυπο (*instance*) της συστηματικής επεξεργασίας για την επίτευξη ενός στόχου. Μια ροή εργασίας μπορεί να θεωρηθεί ως μία (συχνά βέλτιστη ή σχεδόν βέλτιστη) υλοποίηση των μηχανισμών που απαιτούνται για να εκτελεστεί το ίδιο σχέδιο επανειλημμένα.

3. Ο έλεγχος της ροής είναι μια έννοια ελέγχου που εφαρμόζεται στις ροές των εργασιών για την εκτροπή από στατικές έννοιες ελέγχου που εφαρμόζονται στο απόθεμα, το οποίο διαχειρίζεται απλά τις παραγγελίες, σε μια πιο δυναμική έννοια ελέγχου, που διαχειρίζεται η ταχύτητα της ροής και η ροή όγκου στην κίνηση και στη διαδικασία.

31.3 Ιστορική αναδρομή

Στη δεκαετία του 1980, ο όρος ροή εργασίας χρησιμοποιήθηκε για πρώτη φορά στη σύγχρονη μορφή της, στη βιομηχανία λογισμικού, από τον αντιπρόεδρο της FileNet David Siegel. Η εταιρεία αποκάλεσε το λογισμικό της αυτοματοποίησης της επιχειρηματικής διαδικασίας "WorkFlo"(ομόηχο του workflow στα αγγλικά, που σημαίνει ροής εργασίας).

Το 1995, η εκδοτική βιομηχανία μελέτησε πώς οι παραδοσιακές διαδικασίες έκδοσης μπορεί να είναι προσδιορισθούν και να αποτυπωθούν γραμμικά σε ψηφιακές μεθόδους, προκειμένου να μειωθεί ο χρόνος καθυστέρησης, καθώς και το σημαντικό κόστος εκτύπωσης και αποστολής για την παροχή αντιγράφων βιβλίων και περιοδικών σε αποθήκες και συνδρομητές.

Το ενδιαφέρον για τις επιχειρησιακές διαδικασίες εκφράστηκε στο χώρο των ηλεκτρονικών επιχειρήσεων σε διάφορες μορφές.

Το UN/CEFACT - Ηνωμένα Έθνη/ Κέντρο διευκόλυνσης εμπορίου και ηλεκτρονικών επιχειρήσεων, επικέντρωσε στα τέλη της δεκαετίας του 90 το μεγαλύτερο μέρος των εργασιών του στο Open-edi Reference Model. Το Open-edi Reference Model, που στη συνέχεια έγινε πρότυπο (ISO standard 14662), καθόριζε ένα τρόπο επικοινωνίας και αλληλεπίδρασης εταιριών με εμπορικούς συνεργάτες. Το συγκεκριμένο πρότυπο, σε αντίθεση με το έως τότε χρησιμοποιούμενο μοντέλο,

περιέγραφε λεπτομερώς την τεχνολογία που οι επιμέρους συνεργάτες σε μια διαδικασία χρησιμοποιούσαν και αναγνώριζε τις επιχειρησιακές διαδικασίες περιεχομένου και των μελών που τις χρησιμοποιούσαν. Αναγνώριζε επίσης τις διενέργειες και τα μηνύματα προς και από τις διαδικασίες αυτές που μπορούσαν να πραγματοποιηθούν, καθώς και την ερμηνεία αυτών των μηνυμάτων.

Η πρωτοποριακή εργασία του UN/CEFACT συνέβαλε δραστικά στη δημιουργία του Business Process Specification Schema ή BPSS που αναπτύχθηκε σαν πρωτοβουλία της ebXML. Η BPSS συμπεριέλαβε και τα δύο πρότυπα που αναφέρονται παρακάτω σε ένα νέο πλαίσιο. Το πλαίσιο αυτό περιελάμβανε μεταξύ άλλων και ευρετήρια, ανταλλαγή μηνυμάτων, συμφωνίες μεταξύ των εμπορικών συνεργατών, επικοινωνία μέσω μηνυμάτων και ενιαίο τρόπο ερμηνείας των. Το συγκεκριμένο πλαίσιο υλοποιήθηκε τόσο σε XML όσο και σε UML και στηρίχθηκε στην αρχιτεκτονική του ebXML.

Το ως άνω περιγραφόμενο συγκεντρωτικό ενδιαφέρον στις επιχειρησιακές διαδικασίες επηρέασε σε μεγάλο βαθμό τις εργασίες του RosettaNet . Ένα νέο πλαίσιο για τις ηλεκτρονικές υπηρεσίες σχεδιάστηκε και μάλιστα έγινε άμεσα αποδεκτό ως μοντέλο από πολλές επιχειρήσεις. Οι αρχιτέκτονες του RosettaNet αντιστάθηκαν στην παρόρμηση να αναπτύξουν ένα λεξιλόγιο επικοινωνίας αποκλειστικά βασισμένο σε XML και συγκεντρώθηκαν στον ορισμό επιχειρησιακών διαδικασιών για την ηλεκτρονική βιομηχανία που αυτοί περιέγραψαν ως Partner Interface Processes ή PIPs. Το κάθε PIP αναπαριστά μία ή περισσότερες διαδράσεις και περιλαμβάνει τα ηλεκτρονικά επιχειρησιακά έγγραφα που ανταλλάσσονται, το λεξιλόγιο που χρησιμοποιείται για το καθένα. Το RosettaNet δημοσίευσε 6 συλλογές από PIPs, τα οποία καλύπτουν ισάριθμες επιχειρησιακές λειτουργίες, μία εκ των οποίων αφορά την διαχείριση.

Ανάμεσα στα ήδη υπάρχοντα XML λεξικά για ηλεκτρονικές επιχειρήσεις το RosettaNet αποτέλεσε μια από τις πιο επιτυχημένες πρωτοβουλίες, χάρη στο ότι ένα μεγάλο μέρος του αφιερώθηκε στο να περιγράψει business processes.

3.1.4 Χαρακτηριστικά και φαινομενολογία

- Μοντελοποίηση: Τα προβλήματα ροής εργασίας μπορούν να μοντελοποιηθούν και να αναλυθούν με βάση διαγραμματικούς φορμαλισμούς όπως τα δίκτυα Petri

- Μέτρηση: Πολλές από τις έννοιες που χρησιμοποιούνται για τη μέτρηση των συστημάτων προγραμματισμού σε ερευνητικές δραστηριότητες είναι χρήσιμες για τη μέτρηση των γενικών ροών εργασίας. Αυτές περιλαμβάνουν την δυναμικότητα, τον χρόνο επεξεργασίας, καθώς και άλλες τακτικές μετρήσεις.

- Εξειδικευμένες συνδηλώσεις: Η ροή εργασίας έχει εξειδικευμένες συνδηλώσεις στην τεχνολογία πληροφοριών, στην διαχείριση εγγράφων και εικόνων. Από το 1993, μια κοινοπραξία εμπόρων που αφορά ειδικότερα στη διαχείριση της ροής εργασίας και τη διαλειτουργικότητα των συστημάτων διαχείρισης ροής εργασίας έχει σχηματίσει τον Συνασπισμό Διαχείρισης Ροών Εργασίας (Workflow Management Coalition)

- Επιστημονικές ροές εργασίας: Οι επιστημονικές ροές εργασίας βρήκαν ευρείας αποδοχής στους τομείς της βιοπληροφορικής και της χημειοπληροφορικής στις αρχές της δεκαετίας του 2000, όπου κάλυψαν με επιτυχία την ανάγκη για πολλαπλά διασυνδεδεμένα εργαλεία, τον χειρισμό πολλαπλών μορφών δεδομένων και μεγάλων ποσοτήτων δεδομένων. Επίσης, το πρότυπο της επιστημονικής ροής εργασίας ήταν κοντά στην παγιωμένη παράδοση της εγγραφής κομματιών κώδικα σε Perl στην επιστήμη της ζωοφυσικής έρευνα, έτσι ώστε αυτή αποδοχή να αποτελέσει ένα φυσικό βήμα προς μια πιο συγκροτημένη υποδομή των ροών εργασίας.

3.1.5 Συνιστώσες ροών εργασίας

Μια ροή εργασίας συνήθως μπορεί να περιγραφεί χρησιμοποιώντας επίσημες ή ανεπίσημες τεχνικές διαγραμμάτων ροής, που δείχνουν κατευθυνόμενες ροές μεταξύ των σταδίων επεξεργασίας.

Μεμονωμένα στάδια επεξεργασίας ή συνιστώσες μιας ροής εργασίας μπορούν κατά βάση να οριστούν από τρεις παραμέτρους:

1. περιγραφή των εισροών: οι πληροφορίες, το υλικό και η ενέργεια που απαιτείται για να ολοκληρωθεί το βήμα

2. κανόνες μετατροπής, αλγόριθμοι, οι οποίοι μπορούν να πραγματοποιηθούν από ανθρώπους ή μηχανές, ή συνδυασμός των δύο

3. περιγραφή των εκροών: οι πληροφορίες, το υλικό και η ενέργεια που παράγεται από το βήμα και που παρέχονται ως εισροές στα μεταγενέστερα βήματα.

Οι συνιστώσες μπορούν να συνδεθούν μαζί, αν το αποτέλεσμα ενός προηγούμενου (συνόλου) στοιχείου (-ων) είναι ίση με τις υποχρεωτικές απαιτήσεις εισόδου του επόμενου στοιχείου.

Όταν οι συνιστώσες δεν είναι τοπικές υπηρεσίες, αλλά καλούνται εξ αποστάσεως υπηρεσίες για να εξυπηρετηθούν, όπως στις **υπηρεσίες διαδικτύου**, επιπλέον περιγραφικές οντότητες (όπως η ποιότητα της υπηρεσίας και η διαθεσιμότητα) πρέπει επίσης να εξεταστούν.

3.2 BPEL

3.2.1 Ορισμός

Η Γλώσσα Εκτέλεσης Επιχειρησιακών Διαδικασιών (Business Process Execution Language - BPEL), συντόμευση της πλήρους ονομασίας Γλώσσα Εκτέλεσης Επιχειρησιακών Διαδικασιών για υπηρεσίες διαδικτύου (Web Services Business Process Execution Language - WS-BPEL) είναι ένα πρότυπο εκτελέσιμης γλώσσας κατά τον OASIS για τον προσδιορισμό των αλληλεπιδράσεων με δικτυακές υπηρεσίες. Διεργασίες στο Business Process Execution Language εξαγωγής και εισαγωγής πληροφοριών με τη χρήση Web Service διασυνδέσεις αποκλειστικά.

3.2.2 Γενικά

Οι αλληλεπιδράσεις των υπηρεσιών διαδικτύου μπορούν να περιγραφούν με δύο τρόπους: ως εκτελέσιμες επιχειρησιακές διαδικασίες ή ως αφηρημένες επιχειρησιακές διαδικασίες. Οι εκτελέσιμες επιχειρησιακές διαδικασίες μοντελοποιούν την πραγματική συμπεριφορά ενός συμμετέχοντος σε μια επιχειρησιακή αλληλεπίδραση.

Οι αφηρημένες επιχειρησιακές διαδικασίες είναι εν μέρει καθορισμένες διαδικασίες που δεν έχουν ως σκοπό να εκτελεστούν. Μια αφηρημένη διαδικασία μπορεί να κρύβει μερικές από τις απαιτούμενες συγκεκριμένες επιχειρησιακές λεπτομέρειες. Οι αφηρημένες διαδικασίες εξυπηρετούν στο να δίνεται ένας περιγραφικός ρόλο, με περισσότερες από μία πιθανές περιπτώσεις χρήσης (use cases), συμπεριλαμβανομένων παρατηρήσιμων συμπεριφορών και προτύπων διαδικασίας. Η WS-BPEL προορίζεται να χρησιμοποιηθεί και για την μοντελοποίηση της συμπεριφοράς των αφηρημένων αλλά και των εκτελέσιμων διαδικασιών.

Η WS-BPEL παρέχει μια γλώσσα για την προδιαγραφή των εκτελέσιμων και των αφηρημένων επιχειρησιακών διαδικασιών. Με αυτόν τον τρόπο, επεκτείνει το μοντέλο αλληλεπίδρασης των υπηρεσιών διαδικτύου και τους επιτρέπει να υποστηρίξουν επιχειρησιακές συναλλαγές. Η WS-BPEL ορίζει ένα διαλειτουργικό πρότυπο ολοκλήρωσης που μπορεί να διευκολύνει την επέκταση της αυτοματοποιημένης διαδικασίας ολοκλήρωσης, τόσο εντός όσο και μεταξύ των επιχειρήσεων.

Η προέλευση του BPEL μπορούν να εντοπιστούν στο WSFL και την XLANG. Σειριοποιείται μέσω της XML και έχει ως στόχο να διευκολύνει τον *προγραμματισμό των μεγάλων, σύνθετων προγραμμάτων* . Οι έννοιες του προγραμματισμού των μεγάλων και τον προγραμματισμό των μικρών γίνεται στην διάκριση μεταξύ δύο πτυχών της γραφής κώδικα για τις μακροχρόνιες ασύγχρονες διαδικασίες που συνήθως συναντά κανείς στις επιχειρησιακές διαδικασίες.

Ο προγραμματισμός των μεγάλων αναφέρεται γενικά σε υψηλού επιπέδου αλληλεπιδράσεις μεταβατικής κατάστασης μιας διαδικασίας – η BPEL αναφέρεται στην έννοια αυτή ως αφηρημένη διαδικασία. Μια BPEL αφηρημένη διαδικασία αντιπροσωπεύει ένα σύνολο δημόσιων παρατηρούμενων συμπεριφορών σε έναν τυποποιημένο τρόπο. Μια αφηρημένη διαδικασία περιλαμβάνει πληροφορίες όπως το πότε να περιμένει για μηνύματα, πότε να προετοιμάζεται για την αποστολή μηνυμάτων, πότε να αντισταθμίζει τις αποτυχημένες συναλλαγές, κτλ. Ο προγραμματισμός των μικρών, αντιθέτως, ασχολείται με την βραχύβια προγραμματική συμπεριφορά, που συχνά εκτελείται ως μεμονωμένη συναλλαγή και αφορά την πρόσβαση σε τοπική λογική και πόρους, όπως αρχεία, βάσεις δεδομένων κλπ. Η

ανάπτυξη της BPEL προήρθε από την ιδέα ότι για τον προγραμματισμό των μεγάλων και για τον προγραμματισμό των μικρών απαιτούνται διαφορετικά είδη γλωσσών.

3.2.3 Ιστορία

Η ανάγκη για αναπαράσταση επιχειρησιακών διαδικασιών σε εφαρμογές με δικτυακές υπηρεσίες, όπως φάνηκε από την επιτυχία του RosettaNet, οδήγησε στη δημιουργία πολλών άλλων XML λεξιλογίων και προτύπων δικτυακών υπηρεσιών που αφορούν αυτά τα ζητήματα. Οι δύο πιο σημαντικές συμβολές είναι η Business Process Modeling Language (BPML) αλλά και -όπως είδαμε- η Business Process Execution Language for Web Services (BPEL-WS).

Η IBM και η Microsoft είχε καθεμιά ορίσει τις δικές τους, αρκετά παρόμοιες, γλώσσες για «προγραμματισμός των μεγάλες», τις WSFL και XLANG, αντίστοιχα. Με τη δημοτικότητα και την έλευση του BPML, και την αυξανόμενη επιτυχία του BPMI.org καθώς και το ανοιχτό κίνημα BPMS υπό την ηγεσία της JBoss και της Intalio Inc, η IBM και η Microsoft αποφάσισαν να συνδυάσουν αυτές τις γλώσσες σε μια νέα γλώσσα, την BPEL4WS. Τον Απρίλιο του 2003, οι BEA Systems, IBM, Microsoft, SAP και Siebel Systems υπέβαλαν BPEL4WS 1.1 στον OASIS για την τυποποίηση μέσω της τεχνικής επιτροπής της BPEL για δικτυακές υπηρεσίες. Παρόλο που BPEL4WS εμφανίστηκε τόσο ως 1.0 όσο και ως 1.1 εκδοχή, η OASIS WS-BPEL τεχνική επιτροπή ψήφισε στις 14 Σεπτεμβρίου του 2004 να ονομάσουν την προδιαγραφή WS-BPEL 2.0. Αυτή η αλλαγή ονόματος έγινε για να ευθυγραμμιστεί η BPEL με τις υπόλοιπες τυποποιημένες συμβάσεις ονομασίας δικτυακών υπηρεσιών που αρχίζουν με το WS – καθώς και μαρτυρά τις σημαντικές βελτιώσεις μεταξύ BPEL4WS 1,1 και WS-BPEL 2.0. Αν δεν συζητάμε για μια συγκεκριμένη έκδοση, τότε συνήθως χρησιμοποιούμε σκέτο το ακρωνύμιο BPEL

Τον Ιούνιο του 2007, οι Active Endpoints, Adobe, BEA, IBM, Oracle και SAP δημοσίευσαν τις BPEL4People και WS-HumanTask προδιαγραφές, οι οποίες περιγράφουν τον τρόπο που μπορούν να εφαρμοστεί η αλληλεπίδραση του ανθρώπου σε BPEL διαδικασίες

3.2.4 Σχεδιαστικοί στόχοι της BPEL

Υπήρχαν δέκα αρχικοί σχεδιαστικοί στόχοι που σχετίζονται με BPEL:

1. Ο ορισμός επιχειρησιακών διαδικασιών που αλληλεπιδρούν με εξωτερικούς φορείς μέσω λειτουργιών δικτυακών υπηρεσιών, οι οποίες θα ορίζονται βάσει του προτύπου WSDL 1.1, καθώς και θα είναι δηλώνουν την εμφάνισή τους ως υπηρεσίες διαδικτύου πάλι βάσει προτύπου WSDL 1.1. Οι αλληλεπιδράσεις είναι «αφηρημένες» υπό την έννοια ότι η εξάρτηση είναι σε τύπου portType ορισμούς και όχι σε κατευθείαν ορισμούς θυρών (port).

2. Καθορισμός επιχειρησιακών διαδικασιών με χρήση γλώσσα βασισμένη σε XML. Να μην ορίζεται μια γραφική αναπαράσταση των διαδικασιών ή να παρέχεται οποιαδήποτε συγκεκριμένη μεθοδολογία σχεδιασμού για τις διαδικασίες.

3. Ορισμός ενός συνόλου εννοιών ενορχήστρωσης δικτυακών υπηρεσιών που προορίζονται να χρησιμοποιηθούν τόσο από την εξωτερική (αφηρημένη) όσο και από την εσωτερική (εκτελέσιμη) άποψη μιας επιχειρηματικής διαδικασίας. Μια τέτοια επιχειρησιακή διαδικασία καθορίζει την συμπεριφορά μιας ενιαίας αυτόνομης οντότητας, που συνήθως λειτουργεί σε αλληλεπίδραση με άλλες παρόμοιες ομότιμες οντότητες. Εξυπακούεται ότι κάθε πρότυπος τρόπος χρήσης (π.χ. αφηρημένη άποψη και εκτελέσιμο άποψη) θα απαιτήσει λίγες εξειδικευμένες επεκτάσεις, αλλά αυτές οι επεκτάσεις πρέπει να περιορίζονται στο ελάχιστο και να δοκιμάζονται ως προς τις απαιτήσεις, όπως η εισαγωγή / εξαγωγή και ο έλεγχος συμμόρφωσης που συνδέουν τις δύο μορφές χρήσης .

4. Παροχή τόσο ιεραρχικής όσο και γραφικής παράστασης τομέων ελέγχου, και επίτρεψη των χρήσεών τους να αναμειχθούν όσο απρόσκοπτα είναι δυνατό. Αυτό θα πρέπει να μειώνει τον κατακερματισμό του χώρου της μοντελοποίησης της διαδικασία.

5. Παροχή λειτουργιών διαχείρισης δεδομένων για τον απλό χειρισμό των δεδομένων που απαιτούνται για τον καθορισμό των δεδομένων της διαδικασίας και για τον έλεγχο της ροής.

6. Υποστήριξη ενός μηχανισμού αναγνώρισης για διαδικασίες που επιτρέπει τον ορισμό των αναγνωριστικών σε επίπεδο μηνύματος αίτησης της εκάστοτε εφαρμογής.

7. Η υποστήριξη της έμμεσης δημιουργίας και τερματισμού των διαδικασιών ως βασικό μηχανισμό του κύκλου ζωής. Σύνθετες εργασίες του κύκλου ζωής, όπως η

"διακοπή" και "συνέχιση" μπορούν να προστεθούν σε μελλοντικές εκδόσεις για τη βελτίωση της διαχείρισης του κύκλου ζωής.

8. Ορισμός ενός μοντέλου μακροχρόνιας συναλλαγής που θα βασίζεται σε δοκιμασμένες τεχνικές, όπως ενέργειες αποζημίωσης και οριοθέτησης του πεδίου με σκοό την υποστήριξη για ανάκτηση από αποτυχία σε μέρη μακροχρόνιων επιχειρησιακών διαδικασιών.

9. Χρήση των υπηρεσιών διαδικτύου ως μοντέλο για την αποσύνθεση και την συναρμολόγηση διαδικασιών

10. Βάση στα πρότυπα των υπηρεσιών διαδικτύου, όσο το δυνατόν περισσότερο σε ένα συνθετικό και σπονδυλωτό τρόπο.

3.2.5 Η γλώσσα BPEL

Η BPEL είναι μια γλώσσα ενορχήστρωσης, όχι μια γλώσσα χορογραφίας. Η κύρια διαφορά μεταξύ ενορχήστρωση και η χορογραφία είναι δυνατότητα εκτέλεσης και ελέγχου. Μια ενορχήστρωση καθορίζει μία εκτελέσιμη διαδικασία που περιλαμβάνει ανταλλαγές μηνυμάτων με άλλα συστήματα, έτσι ώστε οι ακολουθίες ανταλλαγής μηνυμάτων να ελέγχονται από το σχεδιαστή της ενορχήστρωσης. Μια χορογραφία καθορίζει ένα πρωτόκολλο για ομότιμες αλληλεπιδράσεις. Τέτοιο πρωτόκολλο δεν είναι άμεσα εκτελέσιμο, καθώς επιτρέπει πολλές διαφορετικές υλοποιήσεις (διαδικασίες που συμμορφώνονται με αυτό). Η χορογραφία μπορεί να πραγματοποιηθεί, αφού τροποποιηθεί σε ενορχήστρωση, με τη σύνταξη ενός ενορχήστρωση (π.χ. με τη μορφή μιας διαδικασίας BPEL) για κάθε μέλος που εμπλέκεται σε αυτό. Η ενορχήστρωση και η χορογραφία είναι διακρίσεις που βασίζονται στις αναλογίες: ενορχήστρωση παραπέμπει στον κεντρικό έλεγχο (από τον μαέστρο) της συμπεριφοράς ενός κατανεμημένου συστήματος (η ορχήστρα που αποτελείται από πολλούς παίκτες), ενώ η χορογραφία αναφέρεται σε ένα κατανεμημένο σύστημα (η ομάδα χορού) που λειτουργούν σύμφωνα με τους κανόνες αλλά δεν έχει κεντρικό έλεγχο.

Η εστίαση της BPEL για τις σύγχρονες επιχειρηματικές διαδικασίες, καθώς και η ιστορία των WSFL και XLANG, οδήγησε την BPEL να υιοθετήσει τις διαδικτυακές

υπηρεσίες ως έναν εξωτερικό μηχανισμό ανακοίνωσής της. Έτσι τα μηνύματα της BPEL εξαρτώνται από τη χρήση του Web Services Description Language (WSDL) 1.1 για την περιγραφή των εξερχόμενων και εισερχόμενων μηνυμάτων.

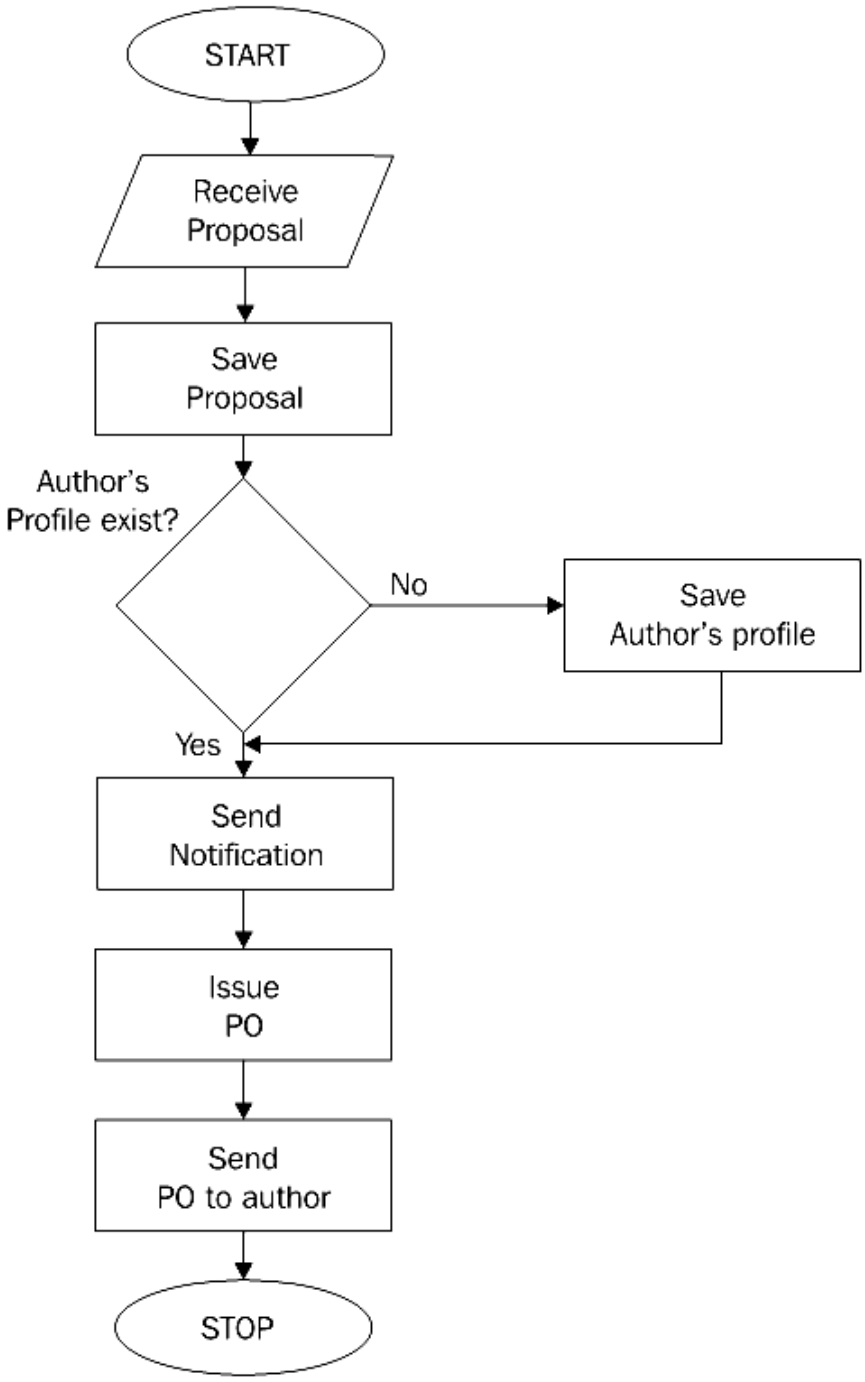
Όπως αναφέρθηκε προηγουμένως, WS-BPEL είναι μια γλώσσα που χρησιμοποιείται για να περιγράψει τη λογική εκτέλεση των υπηρεσιών εφαρμογών διαδικτύου με τον καθορισμό των ροών ελέγχου τους και παρέχοντας έναν τρόπο για τις συμμετέχουσες υπηρεσίες να μοιράζονται ένα κοινό πλαίσιο. Ως συμμετέχουσες υπηρεσίες ορίζουμε αυτές που αλληλεπιδρούν με την WS-BPEL διαδικασία.

Η WS-BPEL βασίζεται σε διάφορες προδιαγραφές, όπως SOAP, WSDL και XML Schema, όπου η WSDL είναι ίσως η πιο σημαντική. Η WSDL είναι αυτό που κάνει μια υπηρεσία να μπορεί να χρησιμοποιηθεί εντός των σύνθετων υπηρεσιών που βασίζοναι στην WS-BPEL. Η WS-BPEL επιτρέπει να οριστούν επιχειρησιακές διαδικασίες που αλληλεπιδρούν με συνεργαζόμενες υπηρεσίες μέσω WSDL περιγραφών. Αυτό θα εξηγηθεί λεπτομερώς στην παράγραφο *WSDL Ορισμοί για Σύνθετες Υπηρεσίες*.

3.2.6 BPEL διαδικασίες

Με την WS-BPEL, μπορεί να δημιουργηθεί μια επιχειρησιακή διαδικασία με την ενσωμάτωση μιας συλλογής υπηρεσιών διαδικτύου σε μια ροή εργασίας επιχειρησιακών διαδικασιών. Μια WS-BPEL επιχειρησιακή διαδικασία ορίζει πώς να συντονιστούν οι αλληλεπιδράσεις μεταξύ ενός στιγμιότυπου (instance) της εν λόγω διαδικασίας και των συμμετεχουσών υπηρεσιών

Η παρακάτω εικόνα δείχνει ένα παράδειγμα ενός διαγράμματος ροής εργασίας που αντιπροσωπεύει μια WS-BPEL επιχειρησιακή διαδικασία:



Όπως μπορείτε να δείτε, η WS-BPEL διαδικασία που απεικονίζεται στην εικόνα ενσωματώνει τις υπηρεσίες που απαιτούνται για την ολοκλήρωση των βημάτων που λαμβάνονται μετά την αποδοχή μιας πρότασης σε μια διαδικασία. Σε αυτό το

συγκεκριμένο παράδειγμα, η διαδικασία ενσωματώνει τέσσερις υπηρεσιών διαδικτύου, όπως απεικονίζεται στην τιμή που αναγράφεται στο τμήμα Ενορχήστρωσης σε προηγούμενη παράγραφο (SOA Συνθέσεις). Όπως θα δείτε στο επόμενο τμήμα, μία WS-BPEL διαδικασία συνδέεται με μια υπηρεσία διαδικτύου μέσω μιας συμμετέχουσας σύνδεσης (partner link) που ορίζεται στο έγγραφο WSDL.

Εκτός από τη δυνατότητα να ***καλέσει*** **(invoke)** πολλαπλές υπηρεσίες, μία WS-BPEL διαδικασία μπορεί να χειρίζεται XML δεδομένα και να εκτελεί **παράλληλες εκτελέσεις (flow)**, **υπό όρους διακλαδώσεις (switch - if), και να ελέγχει βρόχους (while)** ελέγχοντας πάντα την ροή της διαδικασίας. Για παράδειγμα, στην παραπάνω διαδικασία χρησιμοποιείται η δραστηριότητα switch, για τη σύσταση των δύο κλάδων. Εάν η υπό επεξεργασία πρόταση έχει υποβληθεί από ένα νέο δημιουργό, η διαδικασία θα καλέσει τη λειτουργία της υπηρεσίας Συγγραφέας (Author) που είναι αρμόδια για την αποθήκευση των πληροφοριών σχετικά με τον συγγραφέα στη βάση δεδομένων. Διαφορετικά, αυτό το βήμα παραλείπεται.

Για λόγους απλούστευσης, το τμήμα αυτό απεικονίζει μόνο ένα διάγραμμα ροής της WS-BPEL διαδικασίας και όχι τον εκτελέσιμο κώδικα BPEL της εν λόγω διαδικασίας.

3.2.7 WSDL Ορισμοί για Σύνθετες Υπηρεσίες

Στο 2ο κεφάλαιο WSDL, μελετήθηκε η χρήση της WSDL προδιαγραφής, παρέχοντας έναν τρόπο για τον πελάτη υπηρεσιών να κατανοήσει μια υπηρεσίας διαδικτύου. Στην πραγματικότητα, η WSDL διαδραματίζει σημαντικό ρόλο στην ανάπτυξη υπηρεσιοστραφούς αρχιτεκτονικής. Όπως αναφέρθηκε, ακόμη και μια WS-BPEL ενορχήστρωση μπορεί να συνδέεται με WSDL ορισμούς, καθιστώντας κατ' αυτόν τον τρόπο δυνατή την αντιμετώπιση αυτής της ενορχήστρωσης ως αυτόνομη υπηρεσία που μπορεί να καλέσει άλλη/άλλες υπηρεσία/υπηρεσίες ή μπορεί να είναι ένα μέρος μιας άλλης ενορχήστρωσης ή χορογραφίας.

Όντας η ίδια υπηρεσία, μία WS-BPEL διαδικασία θα πρέπει να έχει αντίστοιχο WSDL έγγραφο, που καθιστά δυνατό για τις υπηρεσίες των πελατών να καλέσουν

αυτή την διαδικασία. Για παράδειγμα, η WS-BPEL διαδικασία που απεικονίζεται στην προηγούμενη εικόνα ενδέχεται να σχετίζεται με τον εξής WSDL ορισμό:

```
<?xml version="1.0" encoding="utf-8"?>
<definitions name="proposalProcessingService"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsd1="http://localhost/WebServices/schema/"
    xmlns:plink=
          "http://schemas.xmlsoap.org/ws/2004/03/partner-link/"
    targetNamespace=
          "http://localhost/WebServices/ch1/proposalProcessing/">
  <types>
   <xsd:schema targetNamespace=
        "http://localhost/WebServices/ch1/proposalProcessing.wsdl">
   <xsd:import namespace="http://localhost/WebServices/schema/"
    schemaLocation="http://localhost/WebServices/schema/
                                propProc.xsd"/>
   </xsd:schema>
  </types>
  <message name="receiveProposalInput">
     <part name="body" element="xsd1:proposalDocType"/>
  </message>
  <message name="receiveProposalOutput">
     <part name="body" element="xsd:string"/>
  </message>
  <portType name="proposalProcessingServicePortType">
    <operation name="receiveProposal">
      <input message="tns:receiveProposalInput"/>
      <output message="tns:receiveProposalOutput"/>
    </operation>
  </portType>
  <plink:partnerLinkType name="proposalProcessingService">
     <plink:role name="proposalProcessingServiceRole">
```

```
        <portType="tns:proposalProcessingServicePortType"/>
      </plink:role>
    </plink:partnerLinkType>
  </definitions>
```

Όπως μπορεί να απρατηρηθεί, αυτό το έγγραφο WSDL δεν περιέχει στοιχεία binding ή service. Το γεγονός είναι ότι ένα έγγραφο WSDL της WS-BPEL υπηρεσίας διαδικασιών περιλαμβάνει μόνο τον αφηρημένο ορισμό της υπηρεσίας και partnerLinkType ενότητες που αντιπροσωπεύουν την αλληλεπίδραση μεταξύ των υπηρεσιών της διαδικασίας και των υπηρεσιών των πελατών της. Σε αυτό το συγκεκριμένο παράδειγμα, ο ορισμός της WSDL περιέχει μόνο ένα τμήμα partnerLinkType, υποστηρίζοντας μία λειτουργία που χρησιμοποιείται από έναν πελάτη να αρχίσει την διαδικασία.

Αξίζει να σημειωθεί εδώ, ότι ένα partnerLinkType τμήμα ορίζει μέχρι δύο ρόλους, καθένας από τους οποίους με τη σειρά του συνδέεται με ένα portType που έχει οριστεί στο WSDL έγγραφο νωρίτερα. Η WS-BPEL χρησιμοποιεί τον μηχανισμό των συμμετεχουσών συνδέσεων (partner link) για τον καθορισμό μιας σχέσης μεταξύ μίας WS-BPEL διαδικασίας και των εμπλεκόμενων εργασιών. Ένας WS-BPEL ορισμός διαδικασίας περιέχει partnerLink στοιχεία για να καθοριστούν οι αλληλεπιδράσεις μεταξύ της WS-BPEL διαδικασίας και των πελατών της ή των συμμετεχουσών διεργασιών της. Κάθε ορισμός διαδικασίας partnerLink σε ένα WS-BPEL έγγραφο συνδέεται με ένα partnerLinkType που έχει οριστεί στο αντίστοιχο έγγραφο WSDL. Σχηματικά, αυτό φαίνεται όπως και στην επόμενη εικόνα (εμφανίζεται στην οπίσθια όψη).

Δεδομένου ότι έχει δημιουργηθεί ο WSDL ορισμός (το έγγραφο WSDL) για μια υπηρεσία-διαδικασία, χρειάζεται να τροποποιήσουμε τα WSDL έγγραφα των υπηρεσιών που πρόκειται να κληθούν κατά τη διάρκεια της διαδικασίας εκτέλεσης, όπως είναι τα έγγραφα των συμμετεχουσών υπηρεσιών. Για να ενεργοποιηθεί μια υπηρεσία που να είναι μέρος μίας WS-BPEL ενορχήστρωσης, ίσως να χρειάζεται να προστεθεί ένα partnerLinkType στοιχείο για το αντίστοιχο έγγραφο WSDL. Για παράδειγμα, για να μπορέσει η υπηρεσία ειδοποίησης (Notification) να συμμετάσχει

στην ενορχήστρωση που απεικονιζόταν στο προηγούμενο σχήμα, θα πρέπει να δημιουργήσετε το ακόλουθο έγγραφο WSDL:

```
<?xml version="1.0" encoding="utf-8"?>
<definitions name="notificationService"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:plink=
  "http://schemas.xmlsoap.org/ws/2004/03/partner-link/"
   targetNamespace="http://localhost/WebServices/ch1/notification/">
  <message name="sendMessageInput">
     <part name="body" element="xsd:string"/>
  </message>
  <message name="sendMessageOutput">
     <part name="body" element="xsd:string"/>
  </message>
  <portType name="notificationServicePortType">
    <operation name="sendMessage">
      <input message="tns:sendMessageInput"/>
      <output message="tns:sendMessageOutput"/>
    </operation>
  </portType>
  <binding name="notificationServiceBinding"
              type="tns:notificationServicePortType">
    <soap:binding style="document"
           transport="http://schemas.xmlsoap.org/soap/http"/>
   <operation name="sendMessage">
     <soap:operation soapAction=
           "http://localhost/WebServices/ch1/sendMessage"/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
```

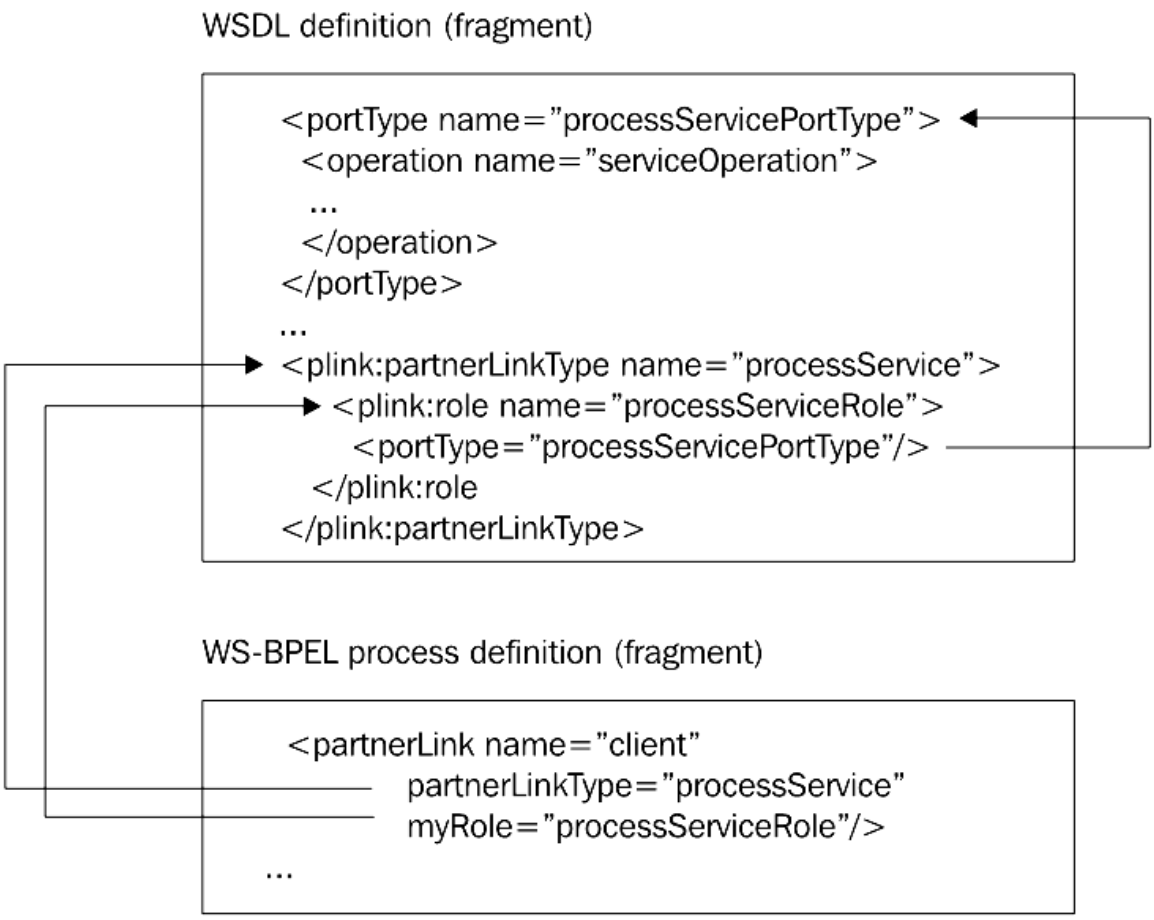
```
            </output>
          </operation>
        </binding>
        <service name="notificationService">
            <port name="notificationServicePort"
                  binding="tns:notificationServiceBinding">
              <soap:address
             location="http://localhost/WebServices/ch1/SOAPserver.php"/>
            </port>
        </service>
        <plink:partnerLinkType name="notificationService">
            <plink:role name="notificationServiceRole">
              <portType="tns:notificationServicePortType"/>
            </plink:role>
        </plink:partnerLinkType>
    </definitions>
```

Στον παραπάνω κώδικα έχει σημειωθεί με έντονα γράμματα το partnerLinkType μπλοκ. Με τη συμπερίληψη αυτής της ενότητας στο τέλος του εγγράφου WSDL που περιγράφει την υπηρεσία, θα μπορέσει η υπηρεσία να είναι συμμετέχων σύνδεσμος (partner link), καθιστώντας το δυνατό να είναι μέρος μίας ενορχήστρωσης. Όπως αναφέρθηκε, η WS-BPEL χρησιμοποιεί τον μηχανισμό των συμμετεχουσών συνδέσεων για να μοντελοποιήσει με τον τύπο της αλληλεπίδρασης των υπηρεσιών στο πλαίσιο της επιχειρησιακής διαδικασίας. Εδώ είναι ένα κομμάτι του ορισμού της WS-BPEL επιχειρησιακής διαδικασίας που απεικονίζεται στην προηγούμενη εικόνα και εκπροσωπεί την proposalProcessingService υπηρεσία-διαδικασία, που δείχνει τη χρήση των συμμετεχουσών συνδέσεων:

```
    <process name="BusinessTravelProcess"
      targetNamespace="http://localhost/WebServices/ch1/
    proposalProcessing/"
        xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
          xmlns:prc="http://localhost/WebServices/ch1/
    proposalProcessing/"
          xmlns:ntf="http://localhost/WebServices/ch1/notification/"
```

```
  <partnerLinks>
    <partnerLink name="client"
                 partnerLinkType="prc:proposalProcessingService"
                 myRole="proposalProcessingServiceRole"/>
    <partnerLink name="sendingNotification"
                 partnerLinkType="ntf:notificationService"
                 partnerRole="notificationServiceRole"/>
...
  </partnerLinks>
...
</process>
```

Για να εξοικονομηθεί χώρος, το παραπάνω απόσπασμα δείχνει μόνο δύο τμήματα partnerLink στον ορισμό της διαδικασίας. Το πρώτο τμήμα partnerLink προσδιορίζει την αλληλεπίδραση μεταξύ της WS-BPEL διαδικασία και των πελατών της, και το άλλο partnerLink προσδιορίζει την αλληλεπίδραση μεταξύ της WS-BPEL διαδικασία και της υπηρεσία ειδοποίησης (Notification) που λειτουργεί εδώ. Το ακόλουθο σχήμα παρατίθεται για την καλύτερη κατανόηση του τρόπου που ο μηχανισμός συμμετεχουσών συνδέσεων χρησιμοποιούνται σε WS-BPEL διαδικασίες.



Το παράδειγμα που απεικονίζεται στο σχήμα αναπαριστά τη σχέση μεταξύ ενός partnerLinkType που ορίζεται στο έγγραφο WSDL που περιγράφει μία WS-BPEL υπηρεσία-διαδικασία και ένα partnerLink που προβλέπεται στον ορισμό της διαδικασίας. Κατά τον καθορισμό των partnerLinkType, χρησιμοποιείται το χαρακτηριστικό myRole έτσι, ώστε να διευκρινιστεί ο ρόλος της WS-BPEL διαδικασίας. Σε αντίθεση, για να διευκρινιστεί ο ρόλος των συμμετεχουσών διαδικασιών, θα χρησιμοποιηθεί το partnerRole χαρακτηριστικό, όπως φαίνεται στο έγγραφο ορισμού διαδικασίας που συζητήθηκε παραπάνω.

Εξετάζοντας τα partnerLink τμήματα στο έγγραφο ορισμού διαδικασίας που συζητήθηκε εδώ, μπορεί να παρατηρηθεί ότι δεν παρέχουν ουσιαστικά κάποια πληροφορία σχετικά με τη θέση των εγγράφων WSDL που περιέχουν τους

αντίστοιχους τύπους συμμετεχουσών συνδέσεων. Αυτές οι πληροφορίες αποθηκεύονται στο αρχείο περιγραφής εγκατάστασης της εκάστοτε διαδικασίας (deployment descriptor file).

Η μορφή του εγγράφου αυτού ποικίλλει ανάλογα με το WS-BPEL εργαλείο που χρησιμοποιείται. Για παράδειγμα, αν σχεδιαστεί μία SOA λύση με την Oracle BPEL Process Manager, το αρχείο περιγραφής εγκατάστασης της διαδικασίας ονομάζεται bpel.xml και μπορεί να μοιάζει ως εξής:

```
<?xml version="1.0" encoding="UTF-8"?>
<BPELSuitcase>
  <BPELProcess src="proposalProcess.bpel" id="proposalProcess">
    <partnerLinkBindings>
      <partnerLinkBinding name="client">
        <property name="wsdlLocation">proposalProcess.wsdl</property>
      </partnerLinkBinding>
      <partnerLinkBinding name="sendingNotification">
        <property name="wsdlLocation">
        http://localhost/WebServices/ch1/notification?wsdl</property>
      </partnerLinkBinding>
...
    </partnerLinkBindings>
  </BPELProcess>
</BPELSuitcase>
```

Στο πρόγραμμα Designer ActiveBPEL, το αρχείο περιγραφής εγκατάστασης της διαδικασίας έχει την επέκταση PDD. Αυτό το έγγραφο περιέχει περιγραφή των πληροφοριών που απαιτούνται για το διακομιστή της ActiveBPEL έτσι, ώστε να εκτελέσει τις αντίστοιχες WS-BPEL διαδικασίες. Μια PDD περιγραφή καθορίζει τη θέση των εγγράφων WSDL που περιγράφουν τα μέρη που εμπλέκονται στην ενότητα των αναφορών (references), όπως φαίνεται στο ακόλουθο απόσπασμα:

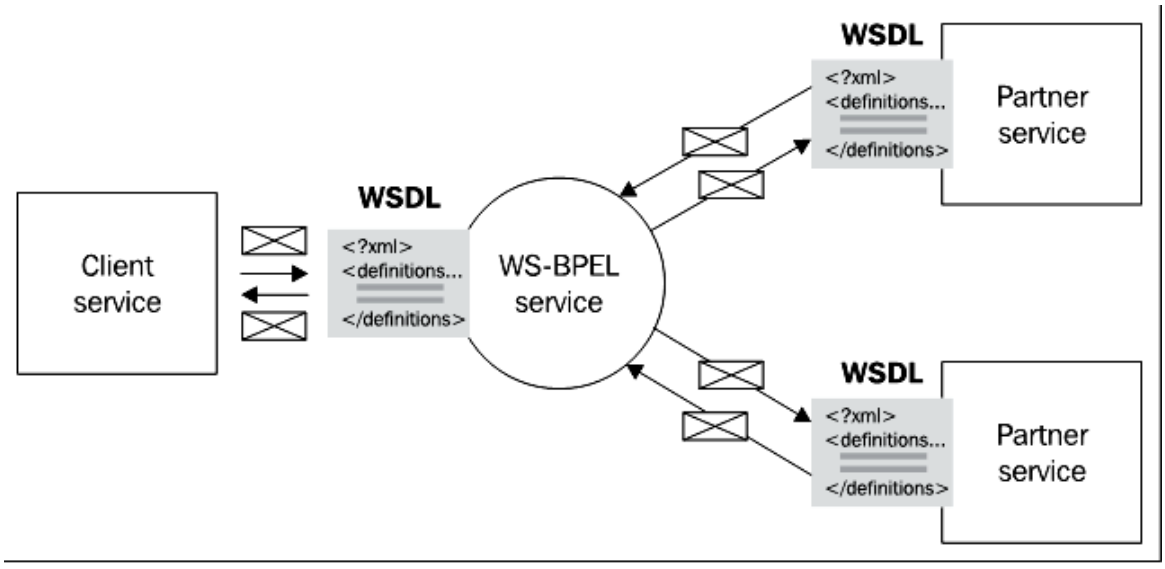
```
<?xml version="1.0" encoding="UTF-8"?>
<process ...>
 ...
  <references>
```

```
    <wsdl location="project:/proposalProcess/WSDL/
                    proposalProcess.wsdl"
  namespace="http://localhost/WebServices/ch1/proposalProcessing/"/>
    <wsdl location="project:/proposalProcess/WSDL/notification.wsdl"
  namespace="http://localhost/WebServices/ch1/notification/"/>
  </references>
</process>
```

3.2.8 Τρόπος λειτουργίας

Προηγουμένως, σημειώθηκε ότι η WS-BPEL χρησιμοποιεί διάφορες προδιαγραφές XML, όπου η WSDL είναι η πιο σημαντική. Μιλώντας πρακτικά, ούσα η ίδια η WS-BPEL μια υπηρεσία, μία WS-BPEL διαδικασία θα πρέπει να έχει ένα αντίστοιχο έγγραφο WSDL που θα την περιγράφει στους πελάτες της. Έτσι, τόσο η WS-BPEL διαδικασία ως υπηρεσία και οι συμμετέχουσες υπηρεσίες που χρησιμοποιούνται στο πλαίσιο αυτής της διαδικασίας είναι "εκτεθειμένοι" στο διαδίκτυο μέσω του WSDL ορισμού τους.

Το ακόλουθο σχήμα απεικονίζει το παραπάνω μοντέλο αλληλεπίδρασης διαγραμματικά:

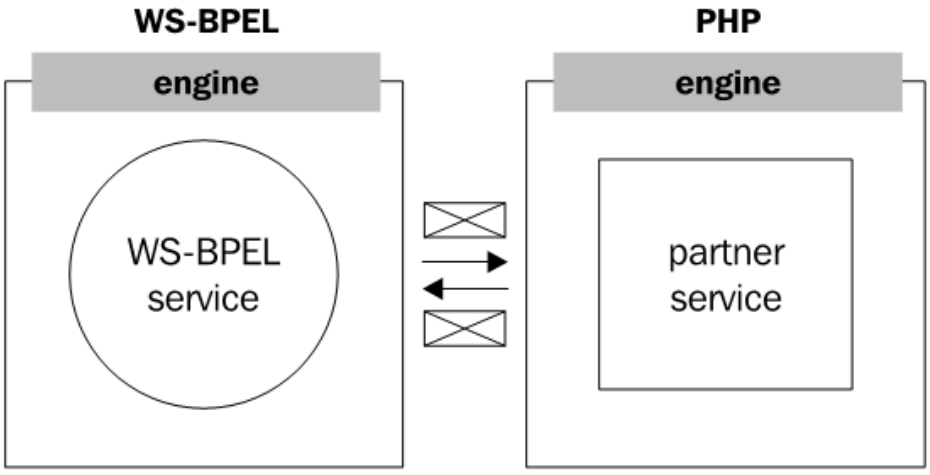


Αν και το διάγραμμα στην παραπάνω εικόνα δεν δείχνει τους μηχανισμούς που κρύβονται πίσω από την WS-BPEL, μπορεί να χρησιμοποιηθεί ως βοήθημα για να

κατανοηθεί πώς μια υπηρεσία-διαδικασία WS-BPEL εντάσσεται στην μεγαλύτερη εικόνα της σύνθετης λύσης που θα βασίζεται στη χρήση της ενορχήστρωσης βάσει της ίδιας της WS-BPEL. Συγκεκριμένα, δείχνει ότι η WS-BPEL υπηρεσία-διαδικασία, όπως και κάθε άλλη υπηρεσία, εκθέτει τη λειτουργία της μέσω μιας διεπαφής WSDL και, επομένως, μπορούν να κληθεί από ένα πελάτη που και αυτός μπορεί να έχει βάσει του προτύπου WSDL.

Στην πραγματικότητα, υπάρχουν διάφοροι τρόποι με τους οποίους Μία WS-BPEL υπηρεσία-διαδικασία μπορεί να κληθεί. Για παράδειγμα, μία WS-BPEL διαδικασία μπορεί να κληθεί μέσα από μια άλλη WS-BPEL διαδικασία, με αποτέλεσμα να αποτελεί, αναδρομικά, μέρος μίας άλλης ενορχήστρωσης.

Όταν δημιουργείται μια WS-BPEL υπηρεσία, δημιουργείται ένας WS-BPEL ορισμός(έγγραφο) που στη συνέχεια θα εκτελεστεί σε μία WS-BPEL μηχανή, ενώ μία συμμετέχουσα υπηρεσία που έχει γραφτεί στην επέκταση PHP & SOAP θα πρέπει να εκτελεστεί σε μια μηχανή PHP. Η ακόλουθη εικόνα δίνει μια γραφική απεικόνιση της κατάστασης αυτής:



Μία WS-BPEL μηχανή διαβάζει έναν εκτελέσιμο ορισμό (αρχείο) μίας WS-BPEL διαδικασίας, καθώς και άλλα έγγραφα, όπως WSDL και XSD και περιμένει για ένα εισερχόμενο μήνυμα να ζητήσει από τον καταναλωτή της WS-BPEL διαδικασίας την εκκίνησή της. Όταν ένα τέτοιο μήνυμα φτάνει, δημιουργεί ένα στιγμιότυπο (instance) της διαδικασίας εκτέλεσης και αλληλεπιδρά με τις συμμετέχουσες υπηρεσίες, όπως περιγράφεται στον ορισμό της διαδικασίας

3.2.9 Η δομή ενός αρχείου WS-BPEL

Ένα WS-BPEL αρχείο επιχειρησιακής διαδικασίας διευκρινίζει πώς να συντονίζονται οι αλληλεπιδράσεις μεταξύ ενός τρέχοντος παραδείγματος αυτής της διαδικασίας και των συμμετεχουσών υπηρεσιών. Η παράγραφος αυτή ασχολείται με τη δομή ενός WS-BPEL εγγράφου, παρέχοντας μια σύντομη περιγραφή των πιο σημαντικών δομών της WS-BPEL γλώσσας.

Γραφικά, η γενική διάρθρωση ενός αρχείου μίας WS-BPEL διαδικασίας μπορεί να μοιάζει κάπως με το ακόλουθο σχήμα:

```
<process...>
  <partnerLinks>
  ...
  </partnerLinks>
  <variables>
  ...
  </variables>
  <faultHandlers>
  ...
  </faultHanders>
  <sequence>
     <receive.../>
     <invoke.../>
     <reply...>
      ...
  </sequence>
</process>
```

Όπως φαίνεται στην εικόνα, ένας WS-BPEL ορισμός αντιπροσωπεύει ένα έγγραφο XML που περιέχει WS-BPEL γλωσσικά τμήματα που εκτελούν τη λογική διαδικασία. Ένας WS-BPEL ορισμός, καθώς και ορισμένα άλλα έγγραφα που σχετίζονται με τη διαδικασία ορισμού χρησιμοποιούνται στη συνέχεια σε μία WS-BPEL μηχανή κατά την οποία ο ορισμός της διαδικασίας θα πρέπει να εκτελεστεί.

Εδώ παρατίθενται οι πιο σημαντικοί WS-BPEL τμήματα που χρησιμοποιούνται κατά τον καθορισμό WS-BPEL ορισμοί:

WS-BPEL τμήμα	Περιγραφή
process	Αυτό είναι το στοιχείο ρίζα ενός ορισμού μίας WS-BPEL διαδικασίας και χρησιμοποιεί χαρακτηριστικά για να δηλώσει αρκετά namespaces που σχετίζονται με την διαδικασίας
partnerLinks	Περιέχει ένα σύνολο partnerLink στοιχείων. Κάθε partnerLink στοιχείο καθορίζει τη σχέση μεταξύ της υπηρεσίας-διαδικασία και των συμμετεχουσών υπηρεσιών.
variables	Περιέχει ένα σύνολο στοιχείων μεταβλητών. Κάθε τέτοιο στοιχείο ορίζει μια μεταβλητή που χρησιμοποιείται από τη διαδικασία.
faultHandlers	Περιέχει χειριστές σφαλματων που περιγράφουν τις ενέργειες που πρέπει να ληφθούν ως απάντηση στις δυσλειτουργίες που ενδέχεται να προκύψουν κατά την εκτέλεση της διαδικασίας.
sequence	Περιέχει μία ή περισσότερες δραστηριότητες που εκτελούνται η μία μετά την άλλη, όπως προκύπτει κατά την κατασκευή. Σε ένα περίπλοκο σενάριο, διάφορες ακολουθίες ίσως ενταχθούν στα τμήματα flow, τα οποία επιτρέπουν συγχρονισμό και συγχρονισμός.
flow	Ενεργοποιεί τον συγχρονισμό μεταξύ των δραστηριοτήτων που που βρίσκονται μέσα σε αυτό το τμήμα. Για παράδειγμα, μπορείτε να ξεκινήσετε να καλείτε διάφορες δραστηριότητες ταυτόχρονα, επιτρέποντας εξαρτήσεις συγχρονισμού μεταξύ τους.
receive	Λαμβάνει ένα μήνυμα από μια συμμετέχουσα υπηρεσία-πελάτη. Συνήθως, η δραστηριότητα αυτή χρησιμοποιείται με την χαρακτηριστικό createInstance να είναι ναι(yes), προκειμένου να αρχίσει η επιχειρησιακή διαδικασία.
invoke	Καλεί την συμμετέχουσα υπηρεσία που ορίζεται από

	το χαρακτηριστικό partnerLink της δραστηριότητας.
reply	Στέλνει ένα μήνυμα απάντησης στο αίτημα που έχει λάβει η δραστηριότητα μας, υποθέτοντας ότι η λειτουργία invoke είναι της μορφής request-response.
assign	Περιέχει τα ζεύγη από (from) και προς (to) τυλιγμένο σε ένα στοιχείο αντιγραφής (copy), τα οποία χρησιμοποιούνται για την ενημέρωση των τιμών των μεταβλητών που έχουν οριστεί στο τμήμα μεταβλητές.
If	Επιτρέπει να προστεθεί λογική διακλάδωσης στην διαδικασία. Με τη χρήση ένθετων στοιχείων elseif και ένα προαιρετικό else στοιχείο, μπορείτε να ορίσετε έναν ή περισσότερους όρους διακλάδωσης εντός της if δραστηριότητας. Να σημειωθεί ότι τα if/elseif/else είναι νέα στοιχεία στην WS-BPEL 2.0. Διαφορετικά, στην BPEL4WS 1.1, χρησιμοποιούνται τα switch/case/instead.
wait	Ορίζει ρητά στην υπηρεσία πόσο να παραμένει ή μέχρι πότε να παραμένει αδρανής.
throw	Ρητά ρίχνει ένα ονοματισμένο λάθος στο εσωτερικό της επιχειρησιακής διαδικασίας. Μπορούν να χρησιμοποιηθούν τα προαιρετικά faultVariable χαρακτηριστικά για να καθοριστεί μία μεταβλητή με τα στοιχεία σφάλματος. Ένα τέτοιο σφάλμα θα πρέπει να το διαχειρίζεται ένας εξυπηρέτησης σφαλμάτων που ορίζεται εντός του τμήματος faultHandlers.

Το παράδειγμα στην παράγραφο που ακολουθεί δείχνει τον τρόπο χρήσης των ανωτέρω WS-BPEL τμημάτων σε έναν ορισμό μιας WS-BPEL διαδικασίας.

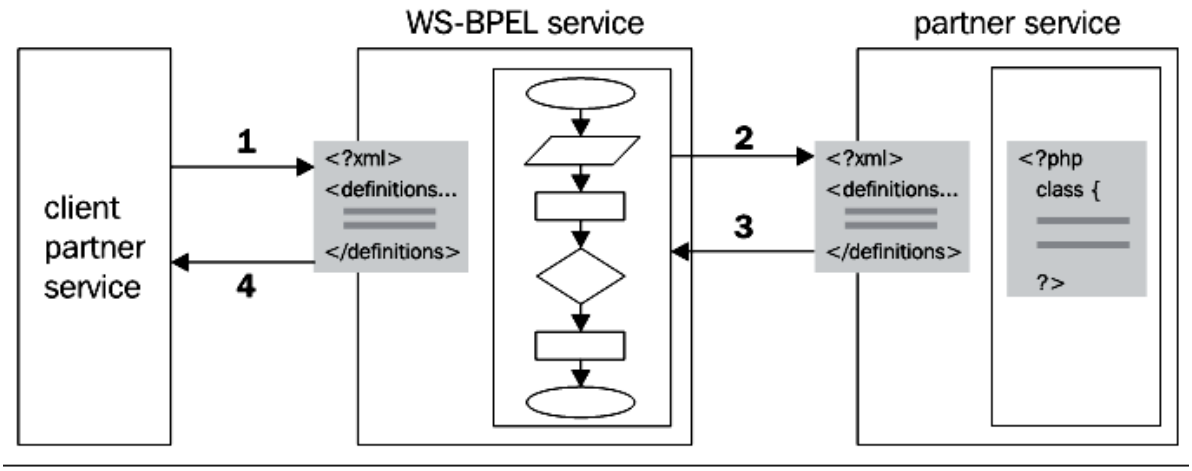
3.2.10 Παράδειγμα WS-BPEL ορισμού

Ας υποθέσουμε ότι χρειάζεται να κατασκευάσουμε έναν ορισμό μίας WS-BPEL διαδικασίας η οποία θα επιστρέψει τις απαιτούμενες πληροφορίες σχετικά με μια παραγγελία. Για λόγους απλούστευσης, η WS-BPEL διαδικασία που αναφέρεται σε

αυτό το τμήμα θα αλληλεπιδρά με μια μόνο συμμετέχουσα υπηρεσία που θα έχει υλοποιηθεί βάσει της γλώσσας PHP. Ειδικότερα, η διαδικασία θα αξιοποιήσει την poOrderDocService συμμετέχουσα υπηρεσία. Η poOrderDocService υπηρεσία επιστρέφει ολόκληρο το έγγραφο po του οποίου το pono είναι ίσο με αυτό που έχει καθοριστεί στο μήνυμα αιτήματος που έχει παραληφθεί από τον πελάτη.

Έτσι, σε αυτό το παράδειγμα γίνεται η υπόθεση ότι δεν υπάρχει λογική διακλάδωσης στη WS-BPEL διαδικασία, δεδομένου ότι έχει μια πολύ απλή δομή – η μόνη πληροφορία που επιστρέφεται σχετικά με την συγκεκριμένη δομή θα είναι το σύνολο του εγγράφου που αντιπροσωπεύει αυτή τη σειρά.

Σχηματικά, το παραπάνω σχέδιο μπορεί να μοιάζει ως εξής:



Τα βήματα της εικόνας, από την οπτική γωνία του παραδείγματος της WS-BPEL διαδικασίας, είναι :

- **Βήμα 1:** Η WS-BPEL διαδικασία λαμβάνει ένα μήνυμα αιτήματος από την συμμετέχουσα υπηρεσία πελάτη. Αυτό γίνεται με τη βοήθεια της λαμβάνουσας δομής της γλώσσας WS-BPEL. Στην πραγματικότητα, αυτό είναι το πρώτο βήμα στον κύκλο ζωής ενός παραδείγματος μιας WS-BPEL διαδικασίας, δεδομένου ότι ένα ***νέο στιγμιότυπο*** (instance) της επιχειρησιακής διαδικασίας δημιουργήθηκε όταν ελήφθη ένα μήνυμα από αίτημα του πελάτη.

- **Βήμα 2, 3:** Η επιχειρησιακή διαδικασία καλεί την συμμετέχουσα υπηρεσία που έχει δημιουργηθεί σε γλώσσα PHP. Το στιγμιότυπο αυτό βασίζεται στην παραδοχή μιας σύγχρονης λειτουργίας αίτησης-απόκρισης (request-response) μεταξύ της

διαδικασίας και του πελάτη. Αυτός είναι ο λόγος για τον οποίο τα δύο βήματα 2 και 3 μπορεί να εφαρμοστούν με τη χρήση μιας ενιαίας κλήσης (invoke) στην WS-BPEL διαδικασία.

- **Βήμα 4:** Η επιχειρησιακή διαδικασία απαντά στο αίτημα του πελάτη που έχει γίνει στο Βήμα 1. Για να γίνει αυτό, η δραστηριότητα απάντησης (reply) μπορεί να χρησιμοποιηθεί.

Όπως φαίνεται και από το παραπάνω στιγμιότυπο, η WS-BPEL διαδικασία που απεικονίζεται στο σχήμα θα πρέπει να περιλαμβάνει τουλάχιστον τις ακόλουθες τρεις δραστηριότητες:

- Λήψη
- Κλήση
- Απάντηση

Πριν προχωρήσουμε στον ορισμό της WS-BPEL διαδικασίας όμως, ας εξετάσουμε το έγγραφο WSDL που θα μπορούσε να χρησιμοποιηθεί για να εκθέσει την υπηρεσία-διαδικασία αυτή στους πελάτες της.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="poInfo"
   targetNamespace="http://localhost:8081/active-bpel/services/
poInfoService.wsdl"
   xmlns:tns="http://localhost:8081/active-bpel/services/
poInfoService.wsdl"
   xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:ns2="http://localhost/WebServices/wsdl/poOrderDoc"
   xmlns="http://schemas.xmlsoap.org/wsdl/">
  <import namespace="http://localhost/WebServices/wsdl/poOrderDoc"
        location="http://localhost/WebServices/wsdl/
          po_orderdoc.wsdl"/>
  <types>
   <schema attributeFormDefault="qualified"
```

```
            elementFormDefault="qualified"
            targetNamespace=
        "http://localhost:8081/active-bpel/services/poInfoService.wsdl"
            xmlns="http://www.w3.org/2001/XMLSchema">
        <element name="poInfoRequest">
         <complexType>
          <sequence>
           <element name="input" type="xsd:string"/>
          </sequence>
         </complexType>
        </element>
       </schema>
    </types>
    <message name="poInfoResponseMessage">
     <part name="payload" type="xsd:string"/>
    </message>
    <message name="poInfoRequestMessage">
     <part name="payload" element="tns:poInfoRequest"/>
    </message>
    <portType name="poInfoPT">
     <operation name="getInfo">
      <input message="tns:poInfoRequestMessage"/>
      <output message="tns:poInfoResponseMessage"/>
     </operation>
    </portType>
    <plnk:partnerLinkType name="poInfoLT">
     <plnk:role name="poInfoProviderRole">
      <plnk:portType name="tns:poInfoPT"/>
     </plnk:role>
    </plnk:partnerLinkType>
    <plnk:partnerLinkType name="poDocLT">
     <plnk:role name="poDocProviderRole">
      <plnk:portType name="ns2:poOrderDocServicePortType"/>
     </plnk:role>
    </plnk:partnerLinkType>
   </definitions>
```

Το πιο ενδιαφέρον πράγμα που πρέπει να σημειωθεί σχετικά με τον παραπάνω ορισμό WSDL είναι ότι δεν περιέχει κανένα δεσμευτικό τμήμα (binding) ούτε τμήματα υπηρεσιών (service). Η WS-BPEL μηχανή θα παράξει αυτούς τους ορισμούς εμμέσως, παρέχοντας έτσι την υπηρεσία-πελάτη με τις απαιτούμενες δεσμευτικές (binding) πληροφορίες.

Ένα άλλο σημαντικό πράγμα που πρέπει να σημειωθεί εδώ είναι ότι η ανωτέρω WSDL εισάγωγει ένα άλλο έγγραφο WSDL, δηλαδή το po_orderdoc.wsdl, το οποίο περιγράφει την poOrderDocService συμμετέχουσα υπηρεσία.

Το po_orderdoc.wsdl έγγραφο καθορίζει το poOrderDocServicePortType τύπο θύρας (portType) που χρησιμοποιείται στο παραπάνω έγγραφο κατά τον καθορισμό του συμμετέχοντος τύπου θύρας poDocLT. Αυτή η συμμετέχουσα σύνδεση τύπου μαζί με την onepoInfoLT, η οποία χρησιμοποιείται για να αναπαριστά την αλληλεπίδραση μεταξύ της επιχειρησιακής διαδικασίας και την εξυπηρέτηση του πελάτη, στη συνέχεια χρησιμοποιούνται για τον ορισμό συμμετεχουσών συνδέσεων στον ορισμό της WS-BPEL διαδικασίας που αναλύεται κατωτέρω.

Τώρα που έχει εξεταστεί μέσα από το έγγραφο WSDL η περιγραφή της WS-BPEL διαδικασίας που απεικονίζεται στην προηγούμενη εικόνα, ήρθε η στιγμή να εξεταστεί ο ορισμός της ίδιας της WS-BPEL διαδικασίας. Για τους σκοπούς αυτής της συζήτησης, ο ορισμός-έγγραφο της WS-BPEL διαδικασία διαιρείται σε κομμάτια, καθένα από τα οποία εξηγείται λεπτομερώς.

Όπως και κάθε άλλος ορισμός WS-BPEL διαδικασίας, αρχίζει με το στοιχείο process που αποτελεί την ρίζα του δέντρου XML, ορίζοντας ως χαρακτηριστικά τα namespaces που σχετίζονται με τη διαδικασία:

```
<?xml version="1.0" encoding="UTF-8"?>
<process xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/
executable"
    xmlns:ns1=
     "http://localhost:8081/active-
bpel/services/poInfoService.wsdl"
    xmlns:ns2="http://localhost/WebServices/wsdl/poOrderDoc"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
        name="poInfo.bpel"
        suppressJoinFailure="yes"
        targetNamespace="http://poInfo.bpel">
```

Στη συνέχεια, εισάγονται οι ορισμοί WSDL που χρησιμοποιούνται στον ορισμό της WS-BPEL. Ειδικότερα, εισάγεται το έγγραφο WSDL που περιγράφει τη διαδικασία για τους πελάτες και η WSDL που περιγράφει τη poOrderDocService συμμετέχουσα υπηρεσία:

```
        <import importType="http://schemas.xmlsoap.org/wsdl/"
            location="wsdl/poInfo.wsdl"
            namespace=
         "http://localhost:8081/active-
bpel/services/poInfoService.wsdl"/>
        <import importType="http://schemas.xmlsoap.org/wsdl/"
            location=
                "http://localhost/WebServices/wsdl/po_orderdoc.wsd
l"
           namespace="http://localhost/WebServices/wsdl/poOrderD
oc"/>
```

Εξετάζοντας το παραπάνω απόσπασμα, ίσως να αναρωτιέστε γιατί το χαρακτηριστικό location που έχει σχέση με την WSDL περιγραφή της WS-BPEL διαδικασίας δεν περιέχει πλήρη διαδρομή προς το έγγραφο. Το γεγονός είναι ότι αυτό το παράδειγμα υποθέτει ότι η διαδικασία θα αναπτυχθεί σε μία μηχανή ActiveBPEL, η οποία, όπως και η μηχανή που αναπτύσσεται στην παρούσα διπλωματική, υποθέτει ότι το αρχείο wsdl μπορεί να βρίσκεται μέσα σε ένα φάκελο δράσης όπου θα βρίσκονται όλα τα αρχεία bpel και wsdl. Το επόμενο βήμα είναι να καθοριστούν οι συμμετέχουσες σχέσεις που εκπροσωπούν τις σχέσεις μεταξύ της WS-BPEL διαδικασία και τους συμμετεχουσών υπηρεσιών της:

```
        <partnerLinks>
         <partnerLink myRole="poInfoProviderRole" name="poInfoProvider"
                    partnerLinkType="ns1:poInfoLT"/>
         <partnerLink name="poDocRequester"
         partnerLinkType="ns1:poDocLT"
partnerRole="poDocProviderRole"/>
       </partnerLinks>
```

Στην ενότητα των συμμετεχουσών συνδέσεων (Partnerlinks), ορίστηκαν δύο συμμετέχουσες συνδέσεις. Η πρώτη προσδιορίζει τη σχέση μεταξύ της διαδικασίας και του πελάτη της. Η δεύτερη προσδιορίζει τη σχέση μεταξύ της διαδικασίας και των συμμετεχουσών υπηρεσιών.

Αν θυμάστε από το κεφάλαιο 1, η απουσία μνήμης προηγούμενης κατάστασης της υπηρεσίας (statelessness) είναι μία από τις βασικές αρχές του υπηρεσιοστραφούς προγραμματισμού, γεγονός που σημαίνει ότι οι υπηρεσίες δεν τηρούν την κατάσταση τους σε μια συγκεκριμένη δραστηριότητα. Όπως αναφέρθηκε, προγραμματίζοντας τέτοιες υπηρεσίες ενθαρρύνει την χαλαρή σύζευξη, την επαναχρησιμοποίηση και την σύνθεση. Ωστόσο, όταν συνδυάζονται υπηρεσίες σε μια εφαρμογή SOA, πιθανότατα να ενδιαφέρει ο σχεδιασμός και υλοποίηση μιας λύσης που να υποστηρίζει μνήμη προηγούμενης κατάστασης στις αλληλεπιδράσεις μεταξύ των συμμετεχουσών υπηρεσιών. Για την επίτευξη αυτού του στόχου σε WS-BPEL, χρησιμοποιούνται μεταβλητές. Στην επόμενη ενότητα των μεταβλητών (variables), μπορούν να οριστούν μεταβλητές που στη συνέχεια θα χρησιμοποιηθούν για την αποθήκευση πληροφοριών για την κατάσταση μεταξύ των αλληλεπιδράσεων των διαδικασιών:

```
<variables>
  <variable messageType=
      "ns1:poInfoRequestMessage" name="poInfoRequestMessage"/>
  <variable messageType=
      "ns1:poInfoResponseMessage"
name="poInfoResponseMessage"/>
  <variable messageType=
      "ns2:getOrderDocInput" name="poDocRequestMessage"/>
  <variable messageType=
      "ns2:getOrderDocOutput" name="poDocResponseMessage"/>
</variables>
```

Τώρα ξεκινά ο καθορισμός της δραστηριότητας sequence, η οποία αναθέτει στην WS-BPEL μηχανή να ξεκινήσει την διαδοχική επεξεργασία των δραστηριοτήτων που βρίσκονται ένθετες μέσα σε αυτήν.

```
<sequence>
```

Η πρώτη δραστηριότητα που απασχολεί την μηχανή κατά την ακολουθία sequence είναι η “λήψη” (receive), η οποία καθορίζει την συμμετέχουσα σύνδεση

partnerLink που περιέχει το myRole, το οποίο χρησιμοποιείται για να ληφθεί ένα μήνυμα αίτησης από μια υπηρεσία-πελάτη. Το αποτέλεσμα του μηνύματος αίτησης (στο παράδειγμα μας θα είναι το τμήμα payload) θα περάσει στην poInfoRequestMessage μεταβλητή που έχει οριστεί στο χαρακτηριστικό μεταβλητής (variable) της δραστηριότητας receive:

```
<receive createInstance="yes"
        operation="getInfo"
        partnerLink="poInfoProvider"
        portType="ns1:poInfoPT"
        variable="poInfoRequestMessage"/>
```

Όπως φαίνεται από το παράδειγμα, το χαρακτηριστικό createInstance της δραστηριότητας receive έχει οριστεί σε ‘‘yes’’, γεγονός που σημαίνει ότι άσκηση αυτής της δραστηριότητας θα δημιουργήσει το στιγμιότυπο της επιχειρησιακής διαδικασίας.

Στην πραγματικότητα, χρησιμοποιώντας το χαρακτηριστικό createInstance με τη receive ή pick δραστηριότητα είναι ο μόνος τρόπος με τον οποίο μπορείτε να δημιουργήσετε ένα στιγμιότυπο μίας WS-BPEL διαδικασίας.

Στη συνέχεια, αντιγράφετε την αξία του στοιχείου της εισόδου (input), το οποίο είναι ένθετο στο XML κομμάτι που πραγματοποιήθηκε στην poInfoRequestMessage μεταβλητή, στην έξοδο (output) η οποία είναι το τμήμα pono της poDocRequestMessage μεταβλητής, όπως φαίνεται παρακάτω:

```
<assign>
  <copy>
    <from part="payload" variable="poInfoRequestMessage">
      <query>ns1:input</query>
    </from>
    <to part="pono" variable="poDocRequestMessage"/>
  </copy>
</assign>
```

Τώρα μπορεί να κληθεί η συμμετέχουσα υπηρεσία poOrderDocService, χρησιμοποιώντας την δραστηριότητα κλήσης (invoke), όπως φαίνεται παρακάτω:

```
<invoke inputVariable="poDocRequestMessage"
        outputVariable="poDocResponseMessage"
        operation="getOrderDoc"
```

```
            partnerLink="poDocRequester"
            portType="ns2:poOrderDocServicePortType ">
        </invoke>
```

Εδώ καθορίζεται η poDocRequestMessage μεταβλητή, που έχει διαμορφωθεί από το προηγούμενο βήμα, ως μεταβλητή εισόδου και καθορίζεται η poDocResponseMessage μεταβλητή, ως η μεταβλητή εξόδου στην ανωτέρω δραστηριότητα κλήσης (invoke). Αξίζει να σημειωθεί ότι το χαρακτηριστικό λειτουργίας (operation) της δραστηριότητας έχει οριστεί σε getOrderDoc, η οποία, είναι το όνομα της λειτουργίας (operation) που έχει οριστεί στο po_orderdoc.wsdl WSDL έγγραφο που περιγράφει την poOrderDocService υπηρεσία.

Είναι σημαντικό να γίνει κατανοητό ότι μια συμμετέχουσα υπηρεσία που έχει κληθεί κατά τη διάρκεια της εκτέλεσης της διαδικασίας, ενδεχομένως να είναι και η ίδια μία διαδικασία-υπηρεσία WS-BPEL. Σε μια τέτοια περίπτωση, η μητρική WS-BPEL διαδικασία χρησιμοποιείται για να περιγράψει διασυνοριακές αλληλεπιδράσεις επιχειρησιακής διαδικασίας που εκτελούνται μέσω WSDL διασυνδέσεων.

Μόλις ολοκληρωθεί αυτή η δραστηριότητα "κλήσης", η poDocResponseMessage μεταβλητή εξόδου θα πρέπει να περιέχει το μήνυμα-αποτέλεσμα που επιστρέφεται από την συμμετέχουσα υπηρεσία poOrderDocService. Προτού οριστεί η δραστηριότητα "απάντησης" (reply), η οποία θα ολοκληρώσει την λειτουργία που άρχισε από την δραστηριότητα "λήψης" (receive) που ορίζεται κατά την έναρξη της ακολουθίας, θα πρέπει να αντιγράψετε την αξία του μέρους doc της μεταβλητής poDocResponseMessage στο μέρος του payload της poInfoResponseMessage μεταβλητής. Ένας τρόπος να γίνει αυτό είναι ως εξής:

```
        <assign>
          <copy>
            <from>$poDocResponseMessage.doc</from>
            <to>$poInfoResponseMessage.payload</to>
          </copy>
        </assign>
```

Τώρα μπορεί να οριστεί η δραστηριότητα "απάντησης" (reply) που θα σχετίζεται με την δραστηριότητα "λήψης" (receive) που είχε οριστεί κατά την έναρξη της ακολουθίας:

```
            <reply operation="getInfo"
              partnerLink="poInfoProvider"
              portType="ns1:poInfoPT"
              variable="poInfoResponseMessage"/>
        </sequence>
    </process>
```

Η δραστηριότητα «απάντησης» είναι η τελευταία σε αυτό το WS-BPEL αρχείο. Μετά από αυτή τη δραστηριότητα, μπορείτε ρητά να τερματίσετε την ακολουθία και στη συνέχεια την διαδικασία. Όπως αναφέρθηκε, η ανωτέρω διαδικασία, αφού έχει αναπτυχθεί σε μία WS-BPEL μηχανή, εκτελείται όταν ένας πελάτης στέλνει ένα μήνυμα-αίτημα. Το μήνυμα αίτησης πρέπει να περιέχει το "pono" του οποίου η αξία έχει περάσει στη συνέχεια στην κλήση της συμμετέχουσας υπηρεσίας poOrderDocService μέσα από την διαδικασία. Η poOrderDocService υπηρεσία, με τη σειρά της, επιστρέφει το σύνολικό "po" έγγραφο με την καθορισμένη μεταβλητή "pono". Η διαδικασία επιστρέφει στη συνέχεια το έγγραφο στον πελάτη που έφτιαξε το στιγμιότυπο της διαδικασία.

3.2.11 Διαφορές με ροές εργασίας σε πλέγματα[11]

Οι Ροές εργασίας σε πλέγματα διαφέρουν από τις "κανονικές" ροές εργασίας, όπως οι ροές των εργασιών των επιχειρήσεων ή οι ροές εργασίας των τμημάτων IT των επιχειρήσεων, κυρίως στα εξής σημεία:

- Μνήμη προηγούμενης κατάστασης της υπηρεσίας: Οι δραστηριότητες των ροών εργασίας σε πλέγμα συνδέονται συχνά με υπηρεσίες που “θυμούνται” την προηγούμενη κατάσταση (ροή εργασίας σε υπηρεσίες πλέγματος), ενώ οι άλλες ροές εργασίας σχετίζονται κυρίως με την κλήση των υπηρεσιών που δεν “θυμούνται” την προηγούμενη κατάσταση(ροές εργασίας σε υπηρεσίες διαδικτύου)

11 http://www.gridworkflow.org/snips/gridworkflow/space/Grid+Workflow

- Αξιοπιστία: Σε ένα πλέγμα οι υπολογιστικών πόροι μπορεί να αποτύχουν κατά τη διάρκεια του χρόνου εκτέλεσης της ροής εργασίας. Αυτό πρέπει να λαμβάνεται υπόψη από προηγμένες τεχνικές διαχείρισης βλάβης ροής εργασιών (π.χ. σημεία ελέγχου της ροής εργασίας και ανάκτηση, παρακολούθηση κτλ.)

- Απόδοση: Ένας από τους στόχους του υπολογισμού σε πλέγμα είναι η παροχή υψηλής απόδοσης υπολογιστικής ισχύος. Ως εκ τούτου σε ροές εργασίας σε πλέγματα πρέπει κανείς να ασχοληθεί με την μεσιτεία/διαχείριση πόρων, με την χρονοδρομολόγηση (εξισορρόπηση φορτίου), και με κατανεμημένες εφαρμογές (parallel computing).

3.3. Μηχανές εκτέλεσης ροών εργασίας BPEL

Λογικό είναι, άμα την προτυποποίηση της BPEL να εμφανίζονται μεγάλες εταιρείες που να προσπαθούν να κερδίσουν μερίδιο της αγοράς των μηχανών εκτέλεσης BPEL. Μία συλλογή μηχανών είναι η εξής:

- IBM BPEL4J (http://www.alphaworks.ibm.com/tech/bpws4j)
- IBM WBI Server Foundation (http://www.ibm.com/software/integration/wbisf)
- Oracle BPEL Process Manager (http://otn.oracle.com/bpel)
- Collaxa BPEL Orchestation Server (http://www.collaxa.com/)
- OpenStorm ChoreoServer (http://www.openstorm.com/)
- FiveSight PXE (http://www.fivesight.com/pxe.shtml)
- Active Endpoints ActiveWebflow Server (http://www.active-endpoints.com/products/activewebflow/index.html?wiki)
- **ActiveBPEL Open Source BPEL Engine** (http://www.activebpel.org?wikipedia)
- bexee BPEL Execution Engine (http://bexee.sourceforge.net)
- Twister

Στα θετικά των μηχανών αυτών είναι ότι ανανεώνονται συχνά και ότι εφαρμόζουν όλα τις πτυχές του προτύπου της BPEL κάθε φορά που ανανεώνεται. Ωστόσο, **απαιτούν μεγάλους υπολογιστικούς πόρους και αρκετή μνήμη προκειμένου να μπορούν να τρέξουν**. Κάτι τέτοιο καθιστά απαγορευτικό το γεγονός να μπορούν να ενσωματωθούν σε συστήματα πολλαπλών χρήσεων ή και εξειδικευμένης χρήσης. Απαιτούν εξυπηρετητές που θα σηκώνουν μόνο αυτή την υπηρεσία.

Προς τον στόχο της σύγχρονης επικοινωνίας υπηρεσιών, η διπλωματική αυτή δείχνει ότι μπορεί να υπάρξει πιο γρήγορες λύσεις και εφαρμογές της BPEL. Αυτό συμβαίνει διότι στην διπλωματική χρησιμοποιούνται εργαλεία τα οποία είναι πολύ πιο καινούργια από τα πεπαλαιωμένα συστήματα που χρησιμοποιούν οι εταιρείες. Ως εκ τούτου, ο κώδικας των μηχανών αυτών αποτελεί σημαντικά κεφάλαια για να ξαναγραφτεί από την αρχή. Όπως αποδεικνύεται από την διπλωματική, όμως, κάτι τέτοιο έχει τεράστια οφέλη.

Ακόμα και η ActiveBPEL, η οποία είναι ανοιχτού λογισμικού πάσχει από τα παραπάνω μειονεκτήματα. Στα αρνητικά, δε, των υπολοίπων συγκαταλέγεται το κόστος που χρειάζεται να καταβάλει μία επιχείρηση ή ακόμα και ένας ιδιώτης για να αγοράσει μία άδεια χρήσης.

4. Εργαλεία

4.1 Εγκατάσταση των εργαλείων

Για την υλοποίηση της εφαρμογής χρειάστηκε η εγκατάσταση των παρακάτω εργαλείων: **Java**, **Tomcat, Axis, Eclipse & Vtd-xml**. Παρακάτω παρατίθεται μια επεξήγηση των εργαλείων αυτών.

4.1.1 Java Development Kit (JDK)

4.1.1.1 Έκδοση του JDK

Η έκδοση του JDK που χρησιμοποιήθηκε ήταν η JDK 6 Update 16 with Java EE, αν και να σημειωθεί ότι σε πολλές περιπτώσεις αναγκαστήκαμε (λόγω συμβατότητας) να επιβάλλουμε μετάφραση σε περιβάλλον JRE 4 και όχι JRE 6.

4.1.1.2 Εγκατάσταση

Η Java είναι μια γλώσσα προγραμματισμού που αναπτύχθηκε αρχικά από τον James Gosling στη Sun Microsystems και κυκλοφόρησε το 1995 ως ένα βασικό συστατικό της πλατφόρμας Java Sun Microsystems. Η γλώσσα αντλεί μεγάλο μέρος της σύνταξης της από την C, αλλά έχει ένα απλούστερο αντικειμενοστραφές μοντέλο και λιγότερες χαμηλού επιπέδου εφαρμογές. Οι εφαρμογές Java μεταφράζονται σε δυαδικό κώδικα (bytecode - αρχείο class) που μπορεί να λειτουργεί σε οποιαδήποτε Java Virtual Machine (JVM) ανεξάρτητα από την αρχιτεκτονική του υπολογιστή.

Είναι μία από τις πιο γνωστές γλώσσες προγραμματισμού, με πάρα πολλές λειτουργίες και με δυνατότητες που συνεχώς αυξάνονται. Επίσης προσφέρει έτοιμες συναρτήσεις οι οποίες διευκολύνουν πολύ την ανάπτυξη γρήγορων εφαρμογών, αφού η υλοποίηση των έτοιμων αυτών συναρτήσεων βελτιώνεται συνεχώς.

Επίσης, η Java αποτελεί πλέον την γλώσσα προγραμματισμού που χρησιμοποιείται στις περισσότερες υπηρεσίες διαδικτύου, κυρίως χάριν στην ευκολία της και στην ταχύτητα που πραγματοποιεί τις ζητούμενες διαδικασίες.

Για να γράψει και να μεταγλωττίσει κανείς κώδικα σε Java θα πρέπει να κατεβάσει την εκδοχή του κιτ ανάπτυξης λογισμικού σε Java από την ιστοσελίδα http://java.sun.com/javase/downloads/index.jsp.

4.1.1.3 Γιατί χρησιμοποιήθηκε η Java

Χρησιμοποιήσαμε την Java για τους εξής λόγους:

- Είναι γρήγορη
- Είναι εύκολη
- Παρέχει έτοιμες χρήσιμες συναρτήσεις
- Μπορεί να επεκταθεί με επιπλέον βιβλιοθήκες ή αρχεία jars που εμπεριέχουν έτοιμες κλάσεις, όπως αυτή της vtd-xml
- Οι εξυπηρετητές Tomcat και Axis παρέχουν μεγαλύτερη συμβατότητα σε αυτή την γλώσσα από οποιαδήποτε άλλη γλώσσα.

4.1.2 Tomcat

4.1.2.1 Έκδοση του Tomcat

Η έκδοση του Tomcat που χρησιμοποιήθηκε είναι η 4.1.40.

4.1.2.2 Εγκατάσταση του Tomcat

Ο Tomcat είναι μια servlet μηχανή. Τα servlets είναι ειδικού τύπου Java εφαρμογές, οι οποίες ενσωματώνονται σε ειδικούς web servers, τους υποδοχείς των servlets ή αλλιώς τις servlet-μηχανές. Ο Tomcat είναι λογισμικό ανοικτού κώδικα και κατά συνέπεια ο οποιοσδήποτε μπορεί να το κατεβάσει από τη σελίδα http://tomcat.apache.org/ και να το εγκαταστήσει.

4.1.2.3 Γιατί χρησιμοποιήθηκε ο Tomcat

Χρησιμοποιούμε τον Tomcat ως HTTP server, δηλαδή ως μια πλατφόρμα που περιέχει κλάσεις για την υποστήριξη των servlets. Τα θετικά στοιχεία του είναι ότι:

- Δεν κολλάει όταν πάθει κάτι ένα servlet.
- Υποστηρίζει τις προδιαγραφές ασφαλείας του J2EE.

- Είναι ένας πολυνηματικός εξυπηρετητής.

Για να εγκατασταθεί, κατεβάζουμε την zip επιλογή (light distributions for JDK 1.4 or later) από τη σελίδα http://tomcat.apache.org/ και την τρέχουμε πηγαίνοντας στον φάκελο όπου εγκαταστάθηκε στον φάκελο bin και τρέχουμε το startup.bat. Αφού ξεκινήσουμε την υπηρεσία, μπαίνουμε στη σελίδα http://localhost:8080/index.jsp (ή σκέτο http://localhost:8080/) όπου θα πρέπει να εμφανίζεται η αρχική σελίδα του Tomcat.

4.1.3 AXIS

4.1.3.1 Έκδοση του Axis

Η έκδοση του Axis που χρησιμοποιήσαμε είναι η Axis-1.4.

4.1.3.2 Γενικά για τον AXIS

Ο Axis είναι μια SOAP μηχανή.

Τι σημαίνει SOAP μηχανή; Είναι ένα σύνολο από κλάσεις, οι οποίες διευκολύνουν την ανάπτυξη των SOAP εφαρμογών είτε από τη μεριά του εξυπηρετητή είτε από τη μεριά του πελάτη.

Γιατί χρειάζεται μια SOAP μηχανή; Ο σκοπός των web services είναι να παρέχουν μια πλατφόρμα ανεξάρτητη γλώσσας προγραμματισμού και λειτουργικού συστήματος. Αυτό επιτυγχάνεται με τη βοήθεια του πρωτοκόλλου SOAP (Simple Object Access Protocol), το οποίο επιτρέπει σε μια εφαρμογή πελάτη να καλέσει μια υπηρεσία διαδικτύου ανεξαρτήτως γλώσσας και λειτουργικού συστήματος που έχει υλοποιηθεί είτε η ίδια η υπηρεσία είτε η εφαρμογή πελάτη. Η Java παρέχει την Java διεπαφή προγραμματισμού εφαρμογών (application programming interface - API) για το XML-RPC (JAX-RPC) που επιτρέπει στους πελάτες να εκτελούν απομακρυσμένες κλήσεις σε μεθόδους (Remote Procedure Calls-RPC). Το JAX-RPC υλοποιεί μια Java έκδοση του πρωτοκόλλου SOAP.

Πρακτικά μια SOAP μηχανή περιέχει βασικές κλάσεις οι οποίες υποστηρίζουν τις παρακάτω λειτουργίες:

•Σειριοποιούν κλήσεις μεθόδων σε πακέτα SOAP

•Αποσειριοποιούν πακέτα SOAP σε κλήσεις μεθόδων

•Ενσωματώνουν XML κείμενα σε πακέτα SOAP

•Εξάγουν τα XML κείμενα από τα πακέτα SOAP

•Υποβάλλουν SOAP αιτήσεις και χειρίζονται τις απαντήσεις

•Δέχονται SOAP αιτήσεις και επιστρέφουν τις απαντήσεις

•Υποστηρίζουν ασύγχρονες λειτουργίες

Η λίστα με τις λειτουργίες είναι αρκετά μεγάλη και κρίνεται σκόπιμο να σταματήσουμε εδώ.

Παρακάτω παρατίθενται επίλογές για SOAP μηχανές, οι επιλογές είναι πολλές:

- Apache SOAP 2.2 (http://ws.apache.org/soap/)

Η Apache SOAP 2.2 είναι μάλλον η πιο ώριμη επιλογή, αλλά δεν υποστηρίζει τη γλώσσα WSDL (Web Services Description Language)

- Apache SOAP 3.0 (ws.apache.org/axis/)

Αυτή η SOAP μηχανή είναι γνωστή και ως Axis. Δεν είναι η επόμενη έκδοση της Apache SOAP μηχανής, αφού είναι κάτι εντελώς καινούργιο. Το όνομα Axis προέρχεται από το Apache eXtensible Interaction System.

- Web Services Developer Pack (http://java.sun.com/webservices/)

Αυτό το πακέτο της Sun Microsystems είναι στην πραγματικότητα μια ένωση πολλών τεχνολογιών. Για παράδειγμα μπορούμε να χρησιμοποιήσουμε το Java API για XML-Based Procedure Call (JAX-RPC), το Java API για XML Processing (JAXP) των SOAP μηνυμάτων κτλ.

- IBM Web Services Toolkit (WSTK) (http://www.alphaworks.ibm.com/tech/webservicestoolkit)

Αυτό το πακέτο είναι παρόμοιο με το προηγούμενο από την άποψη ότι χρησιμοποιεί διάφορες τεχνολογίες. Χρησιμοποιεί τον Axis σαν SOAP μηχανή και τον WebShpere σαν web server.

- IBM WebSphere SDK for Web Services (WSDK) (http://www-106.ibm.com/developerworks/webservices/wsdk)

Αυτό είναι ένα πιο επίσημο πακέτο της IBM, σε σχέση με το WSTK. Επιπλέον συμπεριλαμβάνει και τον Axis μαζί με διάφορα άλλα εξαρτήματα, όπως ο Xerces, ένα προσωπικό UDDI registry, μια βάση δεδομένων, τον WebSphere web server κτλ.

4.1.3.3 Γιατί χρησιμοποιήθηκε ο Axis;

Για την παρούσα διπλωματική επιλέχθηκε ο Axis ως SOAP μηχανή για τους παρακάτω λόγους:

- Ο Axis είναι ανοιχτού λογισμικού
- Ο Axis είναι αυτόνομος εξυπηρετητής
- Ο Axis ενσωματώνεται μέσα σε servlet μηχανές όπως ο Tomcat
- Παρέχει εκτεταμένη υποστήριξη για την Web Service Description Language (WSDL)
- Μπορεί να κατασκευάσει Java κλάσεις από την WSDL
- Παρέχει ένα εργαλείο, το TCP/IP monitoring για να εποπτεύει ο διαχειριστής τα πακέτα που στέλνονται και λαμβάνονται
- Ο Axis έχει υιοθετηθεί από πολλές εταιρείες όπως η IBM που αναφέραμε με το WSTK και άλλες όπως η Macromedia.
- Ο Axis παρέχει υλοποίηση του JAX-RPC.

- Ο Axis έχει μια πολύ επεκτάσιμη υλοποίηση που επιτρέπει εξαιρετικά ευέλικτες κινήσεις και εφαρμογές.

Πριν μπορέσουμε να δουλέψουμε με τον Axis θα πρέπει να έχουμε εγκαταστήσει το Java Development Kit και έναν XML parser όπως ο Xerces.

Για την εγκατάσταση του axis, κατεβάζουμε μια έκδοση του Axis και βάζουμε το 'axis' webapp που βρίσκεται στο φάκελο webapps, μέσα στις webapps του Tomcat. Επιπλέον χρειάζονται και δύο jars αρχεία (ανάλογα με την έκδοση που θα έχουμε κατεβάσει μπορεί να υπάρχουν προεγκατεστημένα – η έκδοση 1.4 που εγκαθιστούμε εμείς τα έχει ήδη μέσα), το xercesImpl.jar και το xmlParserAPIs.jar.

Ελέγχουμε αν όλα εγκαταστάθηκαν σωστά μπαίνοντας στη σελίδα: http://localhost:8080/axis/happyaxis.jsp και βλέποντας αν η happyaxis.jsp μας επιστρέφει σωστά αποτελέσματα.

4.1.4 Eclipse

4.1.4.1 Έκδοση

Η έκδοση που χρησιμοποιήθηκε ήταν η 3.2

4.1.4.2 Εγκατάσταση

Αρκεί κανείς να κατευθυνθεί στο http://www.eclipse.org/downloads/ για να κατεβάσει την πλατφόρμα του eclipse προκειμένου να αρχίσει να προγραμματίζει σε Java. Από εκεί κατεβάζει ένα αρχείο zip που μπορεί να το αποσυμπιέσει όπου θέλει. Έπειτα με ένα διπλό κλικ μέσα στον φάκελο όπου έχει αποσυμπιεστεί το αρχείο zip, θα βρίσκεται το eclipse.exe. Αφού ο χρήστης ερωτηθεί για τον φάκελο όπου θα αποθηκεύονται τα σχέδιά του, τότε μπορεί να αρχίσει να προγραμματίζει. Ο eclipse έχει πολλούς υποστηρικτές στο διαδίκτυο, οπότε εύκολα μπορεί να κανείς να βρει λύσεις στις απορίες του.

4.1.4.3 Γιατί χρησιμοποιήθηκε ο Eclipse

Ο Eclipse είναι μία πλατφόρμα αρκετά δυναμική. Μπορεί κανείς να ρυθμίσει το classpath του, δίχως να πειράξει το classpath του συστήματος, μπορεί να ορίσει ποια

από τις java εκδόσεις θα χρησιμοποιήσει δίχως να πειράξει τις εγκατεστημένες ρυθμίσεις του συστήματος, μπορεί να έχει προειδοποιήσεις για πιθανά σφαλμάτων κατά την διάρκεια συγγραφής του κώδικα προτού τρέξει το πρόγραμμα, είναι γρήγορος και απαιτεί πολύ μικρές ποσότητες μνήμης στο σύστημα στο οποίο τρέχει. Μία από τις σημαντικές ιδιότητές του είναι η φορητότητα που προσφέρει. Μπορεί κανείς να το τρέξει και από τον φορητό δίσκο usb του, σε διαφορετικά λειτουργικά συστήματα. Δοκιμάστηκε σε συστήματα Windows XP®, Windows Vista® και σε Windows 7®.

Ο Eclipse χρησιμοποιείται ευρέως στην ελεύθερη αγορά για τον σχεδιασμό εφαρμογών σε Java και ως εκ τούτου υπάρχει πολύ καλή υποστήριξη σε τυχόντα θέματα που μπορούν να προκύψουν. Επίσης, παρέχει δωρεάν πολλά εργαλεία που είναι αρκετά χρήσιμα για πιο εξεζητημένες εφαρμογές. Παραδείγματος χάριν, μπορεί κανείς με ιδιαίτερη ευκολία να σχεδιάσει γραφικές εφαρμογές, μέσω της swing πλατφόρμας, με την οποία με ένα απλό drag 'n' drop να φτιάξει GUIs (γραφικές διεπαφές για τον χρήστη) που θα χρειάζονταν ώρες εργασίας.

Ο Eclipse έχει σχεδιαστεί με γνώμονα την φορητότητα και σε άλλα λειτουργικά πέραν των λειτουργικών της Microsoft. Έχει το ίδιο γραφικό περιβάλλον και στα δύο άλλα δημοφιλή λειτουργικά: τις διανομές του Linux και στα λειτουργικά MacOS. Οπότε μπορεί κανείς να λειτουργήσει με ιδιαίτερη εύκολα σε όποιο λειτουργικό και εάν βρίσκεται. Το παραπάνω χαρακτηριστικό δεν μένει μόνο στο γραφικό περιβάλλον. Επεκτείνεται και στα αρχεία που χρησιμοποιεί ο eclipse για να φτιάξει κανείς τον φάκελο εργασίας του με τις διάφορες ρυθμίσεις. Όλος ο φάκελος του χώρου εργασίας του σχεδιαστή (workspace) μπορεί να μεταφερθεί από το ένα σύστημα στο άλλο.

Σε αυτό το σημείο μπορώ να πω με ιδιαίτερη ευχαρίστηση ότι ήμουν μάρτυρας σε αυτό το χαρακτηριστικό. Χρησιμοποιούσα συνεχώς το pithos.grnet.gr για να αποθηκεύω κάθε μέρα την πρόοδο της εργασίας μου (και για λόγους ασφαλείας), στον υπολογιστή του εργαστηρίου που χρησιμοποιεί Windows XP, για να φτάνω σπίτι το βράδυ και να εργάζομαι σε λειτουργικό Windows Vista. Ενίοτε, καθώς το λειτουργικό που χρησιμοποιώ περισσότερο είναι η διανομή linux Ubuntu, μπορούσα και εργαζόμουν από εκεί.

Συνοψίζοντας, τα χαρακτηριστικά που με έπεισαν να εργαστώ σε Eclipse και όχι σε κάποια άλλη πλατφόρμα ανάπτυξης Java εφαρμογών, είναι:

- Σπουδαία φορητότητα και στην μεταφορά του ιδίου του κώδικα, όσο και διαλειτουργικό
- Εύχρηστο γραφικό περιβάλλον
- Εύκολο στην χρήση
- Γρήγορο στην ανταπόκριση
- Ελαφρύ
- Απαιτεί ελάχιστες απαιτήσεις για να τρέξει
- Ταυτόχρονη εύρεση σφαλμάτων κατά την συγγραφή του κώδικα
- Drag 'n drop εργαλεία
- Είναι δωρεάν
- Είναι ανοικτού λογισμικού (πατέντα EPL)
- Αναβαθμίζεται συχνά
- Μπορεί να ενσωματώσει jar αρχεία στα διάφορα έργα
- Μπορεί να αλλάξει το περιβάλλον σύγχρονης εκτέλεσης (runtime environment) με λίγα κλικ
- Προσφέρει πληθώρα δωρεάν εργαλείων που εξοικονομούν σπουδαίο χρόνο στους προγραμματιστές
- Έχει σπουδαία υποστήριξη στο διαδίκτυο από λοιπούς χρήστες του προγράμματος

- Είναι πολύ εύκολο στην εκμάθησή του

Για αυτούς τους λόγους, η χρήση του Eclipse θεωρήθηκε ευθύς εξαρχής επιτακτική έως επιβεβλημένη.

4.1.5 VTD-XML

4.1.5.1 Έκδοση

Η έκδοση που χρησιμοποιήθηκε ήταν η 2.5

4.1.5.2 Εγκατάσταση

Η μηχανή της vtd-xml είναι πολύ εύκολο να χρησιμοποιηθεί, αφού από το site http://vtd-xml.sourceforge.net/ μπορεί κανείς να κατεβάσει πολύ γρήγορα το zip αρχείο που έχει μέσα το jar αρχείο, με το οποίο μπορούμε να φτιάξουμε τις κλάσεις μας. Αρκεί αυτό το αρχείο να μπει στο classpath μας, προκειμένου να συμπεριληφθούν οι κλάσεις του με τις δικές μας. Στον eclipse αρκεί να σύρουμε το αρχείο vtd-xml.jar στο έργο μας, και αμέσως θα αναγνωριστεί ως ένα σετ κλάσεων που μπορούν να κληθούν από το έργο μας.

4.1.5.3 Γιατί χρησιμοποιήθηκε το vtd-xml

Το πρώτο βήμα των περισσότερων αλγορίθμων επεξεργασίας XML αρχείων, συνήθως είναι να αποσπάται (εξόρυξη - extraction) το αλφαριθμητικό-συμβολικό περιεχόμενο από το έγγραφο προέλευσης σε πολλά διακριτά αντικείμενα. Το vtd-xml προτείνει μια "μη εξορυκτική" προσέγγιση εξαγωγής των αλφαριθμητικών (string) που διατηρεί το έγγραφο ανέπαφο στη μνήμη. Χρησιμοποιώντας μια δυαδική προδιαγραφή κωδικοποίησης που ονομάζεται Virtual Token Descriptor (VTD), το μοντέλο αντιστοίχησης αντιπροσωπεύει τα τμήματα που χρησιμοποιούνται αποκλειστικά σε σχέση με την αρχική μετατόπιση από την ρίζα και το μήκος από αυτήν. Για να δημιουργηθεί μια ιεραρχική βάση με τα δεδομένα που βρίσκονται σε ένα XML, ο parser στην συνέχεια κατηγοροποιεί περαιτέρω τους δείκτες που βρίσκονται στο ίδιο βάθος χρησιμοποιώντας δομές που μοιάζουν με εγκολπωμένους φακέλους. Μέσα από μια

επίδειξη της ιεραρχικής αυτής πλοήγησης του έγγραφου χρησιμοποιώντας VTD, έχει αποδείχθει ότι είναι πράγματι δυνατόν να δημιουργηθεί ένας δρομέας που διατηρεί τις περισσότερες από τις δυνατοτήτων τυχαίας πρόσβασης που προσφέρει ο DOM κατά ένα μόνο μέρος της μνήμης του. Τα αποτέλεσματα από διάφορα test δείχνουν ότι η εφαρμογή του μοντέλου αναφοράς επεξεργασίας vtd-xml ξεπερνά σημαντικά τον Xerces DOM όσον αφορά **τόσο τη μνήμη όσο και τις επιδόσεις επεξεργασίας**.

Δεδομένου ότι χρησιμοποιεί συγκριτικά με τον DOM πολύ λίγη μνήμη, μπορεί κανείς να κατανοήσει ότι αυτό έχει άμεσες θετικές επιπτώσεις στην διπλωματική. Βελτιώνει τους χρόνους που γίνεται ανάλυση ενός xml αρχείου έως και 10 φορές πιο γρήγορα, βαθμός που είναι σημαντικός για μικρά αλλά και για μικρά αρχεία, δεδομένου ότι η υλοποίησή μας επιτρέπει ταυτοχρονισμό, γεγονός που απαιτεί ταυτόχρονη επεξεργασία του ιδίου μεν αρχείου, διαφορετικών δεικτών που θα επεξεργάζονται από το vtd-xml.

Συνοψίζοντας τις λειτουργίες του vtd-xml, θα μπορούσαμε να πούμε ότι είναι:

- Ο πιο αποδοτικός αναλυτής XML αρχείων του κόσμου σε ζητήματα τυχαίας προσπέλασης καθώς απαιτεί **στην μνήμη περίπου μόνο το 1.3x ~ 1.5x** το μέγεθος ενός εγγράφου XML. Σε XML αρχεία όπου φυλάσσονται δεδομένα μπορεί να φτάσει τα 2x.

- Ο ταχύτερος XML αναλυτής του κόσμου: Σε έναν Core2® 2.5Ghz laptop, ο VTD-XML υπερτερεί των DOM αναλυτών από **5x ~ 12x**, **παρέχοντας 90 ~ 120 MB / sec** σταθερή απόδοση ανά πυρήνα.

- Ο ταχύτερος υλοποιητής του Xpath 1.0

- Αποτελεσματικότερη καταγραφή σε κατάλογο (Indexing) XML

- Ενσωματώνεται τέλεια σε οποιαδήποτε XML εφαρμογή, αφού είναι γενικού προσανατολισμού, όπως άλλωστε και τα xml αρχεία.

- Έυκολος να κατανοηθεί

- Δεν αλλάζει την δομή του πηγαίου εγγράφου xml

- Ελαφρύς, απαιτεί ελάχιστη μνήμη

- Γρήγορος

- Μπορεί να πραγματοποιήσει αυξητική προσθήκη σε XML αρχεία: να κόψει, να επικολλήσει, να χωρίσει και συναρμολογήσει XML έγγραφα με μέγιστη απόδοση.

- Ο μοναδικός αναλυτής του κόσμου XML που επιτρέπει να χρησιμοποιηθεί XPath ορισμός για την επεξεργασία έγγραφων XML μέχρι και 256 GB.

- Μια "έγγραφο-κεντρική" ανάλυση του XML αρχείου

- Αναβαθμίζεται συχνά

- Προσφέρει την δυνατότητα επιτάχυνσης βάσει υλικού (hardware acceleration). Συγκεκριμένα βάσει υλοποίησης ASIC.

- Ένας από τους ελάχιστους τρόπους για να εισαχθεί η επεξεργασία xml σε chip.

Το τελευταίο στοιχείο αποτελεί πολύ σημαντικός παράγοντας. Τα τεστ που έχουν γίνει στον vtd-xml μας δείχνουν ότι μπορεί να υπάρξει σε **ενσωματωμένα συστήματα.**

5.Υλοποίηση

5.1. Σκοπός

Σκοπός της διπλωματικής είναι να κατασκευαστεί μία μηχανή WS-BPEL αρχείων που θα φέρει σε πέρας την κάθε διαδικασία. Η ίδια η μηχανή θα πρέπει να μπορεί να είναι υπηρεσία διαδικτύου και να είναι παραμετροποιήσιμη για εταιρικές εφαρμογές. Στόχος, επίσης της μηχανής είναι να είναι **γρήγορη και ελαφριά**.

5.2. Πλεονεκτήματα Υλοποίησης

Κατ' αρχάς πρέπει να σημειωθεί ότι η υλοποίησή μας εύκολα μπορεί να μπει στον globus και ως επέκταση να αποτελέσει η ίδια ως διαδικτυακή υπηρεσία. Αυτό αυξάνει τις δυνατότητες της αρκετά. Μπορεί αν αναλάβει, φερειπείν, μία εταιρεία την περάτωση γενικευμένων ροών εργασίας. Η υπηρεσία, δηλαδή, δε θα δεσμεύεται να κάνει κάτι συγκεκριμένο ή να επιστρέφει συγκεκριμένο αποτέλεσμα, όπως οι στατικές υπηρεσίες που περιγράφηκαν στα προηγούμενα κεφάλαια.

Είναι μία δυναμική υπηρεσία που υλοποιεί κάθε διάγραμμα ροής που σέβεται τους ορισμούς του προτύπου WS-BPEL 2.0. Αναλαμβάνει να περατώσει την διαδικασία που περιγράφεται στο αρχείο που θα της αποστέλλεται. Η γενική αυτή προσέγγιση αυξάνει δραματικά τις δυνατότητες της. Μπορεί η ίδια η υπηρεσία να διατεθεί από μία εταιρεία στο διαδίκτυο για να εξυπηρετήσει άλλες εταιρείες, ή μπορούν και οι ίδιες εταιρείες να την σηκώσουν στο διαδίκτυο για να εξυπηρετούν εσωτερικά τις δικές τους ανάγκες κάθε φορά που θα χρειάζονται μία συγκεκριμένη υπηρεσία. Μία εταιρεία, μπορεί να έχει μεν τους υπολογιστικούς πόρους, αλλά να μην χρειάζεται να τους χρησιμοποιεί όλους μαζί για μία μόνο υπηρεσία. Ενδεχομένως να χρειάζεται να πραγματοποιεί μία υπηρεσία της κατά ένα χρονικό διάστημα της ημέρας, μία άλλη κατά ένα άλλο χρονικό διάστημα κ.ο.κ. Συνεπώς, δεν χρειάζεται να μισθώσει μία εταιρεία να της διαθέσει πόρους, εφόσον και η ίδια θα τους έχει και θα μπορεί να τους εκμεταλλευθεί κατάλληλα. Οπότε μπορεί να σηκώσει την υπηρεσία που υλοποιήθηκε στην διπλωματική αυτή εργασία, για να εξυπηρετεί την μία περίοδο μία εργασία-υπηρεσία της γραμμένη σε WS-BPEL, μία άλλη πιο μετά πάλι γραμμένη σε WS-BPEL, δίχως να χρειάζεται να έχει στατικά μία και μόνο υπηρεσία “σηκωμένη” στον globus.

Συνεπώς, ένα από τα προτερήματα της διπλωματικής αυτής εργασίας είναι ότι είναι **γενικού σκοπού**, γεγονός που αφήνει στον καθένα να υλοποιήσει δυναμικά μία υπηρεσία. Αυτό φυσικά, συνεπάγεται και μία δυναμικότητα.

Το γεγονός ότι είναι γραμμένη στην γλώσσα Java, μία ευρέως διαδεδομένη και γνωστή γλώσσα προγραμματισμού δίνει την δυνατότητα σε αρκετούς να "πειράξουν" τον κώδικα κατά το δοκούν., να τον βελτιώσουν, να απενεργοποιήσουν ορισμένες από τις λειτουργίες του, να παραμετροποιήσουν ορισμένες μεταβλητές. Σε αυτό το σημείο αξίζει να σημειωθεί ότι σε αρκετά σημεία είναι εμφανής η χρήση των futures που έχει εισάγει η Java. Τα futures ομοιάζουν στα νήματα με την διαφορά ότι ο χρήστης έχει περισσότερες δυνατότητες ελέγχου στα futures. Στην παρούσα διπλωματική, "σηκώναμε" τα futures και περιμέναμε να τελειώσουν προτού προχωρήσουμε σε παρακάτω ενέργειες. Επίσης, μπορεί κανείς να ορίσει πόσα futures μπορεί να χρησιμοποιεί μία διεργασία. Για παράδειγμα, μπορεί στην περίπτωση που μία εταιρεία έχει διαθέσει αυτόν τον κώδικα ως υπηρεσία, να θέλει να περιορίσει τους εκμεταλλεύσιμους πόρους που θα χρησιμοποιούν οι πελάτες της ανάλογα με τα χρήματα που θα της δίνουν. Ανάλογα με το πόσο υψηλό είναι το χρηματικό πόσο που θα καταβάλει ο πελάτης, τόσο θα αυξάνει η εταιρεία και τα threads-futures που θα μπορεί ο πελάτης να χρησιμοποιήσει. Αντίστοιχα, μία εταιρεία που χρησιμοποιεί αυτόν τον κώδικα για εσωτερική κατανάλωση, μπορεί να μη θέλει να δεσμεύει για τις υπηρεσίες της όλους τους δυνατούς πόρους, αλλά να θέλει να κρατά κάποιους πόρους για άλλες διεργασίες. Οπότε με αυτή την **παραμετροποίηση** θα μπορεί να περιορίσει την γενικότητα της υπηρεσίας μας. Παρακάτω παρατίθεται ο κώδικας που υλοποιεί την παραμετροποίηση αυτή:

```
final ExecutorService execSvc =
Executors.newFixedThreadPool(10);
```

Όπως είναι εύκολα κατανοητό από το παραπάνω παράδειγμα, δημιουργούμε μία δεξαμενή από 10 νήματα τα οποία είναι δυνατόν να χρησιμοποιηθούν. Αξίζει να σημειωθεί εδώ ότι η υλοποίηση της δεξαμενής δεν σημαίνει απαραίτητα ότι "σηκώνονται" 10 futures, αλλά ότι είναι δυνατόν να "σηκωθούν" μέχρι 10 futures. Κάτι τέτοιο σημαίνει ότι υπάρχει εξοικονόμηση πόρων σε περίπτωση που αυτοί δεν είναι απαραίτητοι ή χρήσιμοι στην περάτωση της διαδικασίας.

Επίσης, είναι αυτονόητο ότι εάν χρειάζονται παραπάνω από 10 νήματα, τότε η μηχανή εκτέλεσης της Java περιμένει να τελειώσουν με τις εργασίες τους τα υπόλοιπα 10 νήματα και στην συνέχεια, μόλις τελειώσει έστω και ένα, τότε το αιτόν νήμα παίρνει την θέση του στην χρονοδρομολόγηση και στην υλοποίησή του.

Όπως ειπώθηκε και στην παράγραφο των εργαλείων, χρησιμοποιούμε τον vtd-xml αναλυτή. Αυτό συνεπάγεται **άμεσα μικρές απαιτήσεις σε υπολογιστική ισχύ, και κυρίως σε μικρές απαιτήσεις στην μνήμη** καθώς ο vtd-xml αναλυτής φτιάχνει έναν κατάλογο που απαιτεί 1.3 ~ 1.5 το μέγεθος του αρχείου xml σε μνήμη. Κάτι τέτοιο είναι πολύ βολικό, καθώς η υλοποίηση με τα DOMs της Java απαιτεί 10 φορές μνήμη του μεγέθους του αρχείου. Η μείωση της δέσμευσης αυτής της μνήμης σε 6-7 φορές λίγοτερη συνεπάγεται και πιο γρήγορη προσπέλαση στα διάφορα σημεία στα οποία μπορούν να ανακτηθούν δεδομένα από το αρχείο XML. Ο κώδικάς μας, αφού φτιάξει τις κατάλληλες κλάσεις από τις διάφορες μεταβλητές που θα χρησιμοποιήσει, χρόνος που δεν μπορεί να μειωθεί παρά μόνο από την ίδια την υλοποίηση της εικονικής μηχανής της JAVA, συνεχίζει με την υλοποίηση του τμήματος sequence της WS-BPEL. Αυτό το τμήμα, λόγω του ότι χρησιμοποιείται η vtd-xml πραγματοποιείται γρήγορα, με ελάχιστους δεσμευμένους πόρους και με ακρίβεια. Η μόνη καθυστέρηση που εισάγεται πλέον είναι από τις δυνατότητες του δικτύου αλλά και απάντησης από τις υπόλοιπες υπηρεσίες.

Ένα επιπλέον χαρακτηριστικό της vtd-xml είναι η δυνατότητα εκτέλεσής μέσω επιτάχυνσης υλικού (hardware acceleration), μέσω ASIC υλοποίησης. Το χαρακτηριστικό αυτό μαζί με το γεγονός των μικρών απαιτήσεων σε μνήμη και σε επεξεργαστική ισχύ σημαίνει ότι η υλοποίησή μας με κατάλληλες τροποποιήσεις μπορεί να τρέξει σε ενσωματωμένα συστήματα. Στην ουσία, αφήνει την δυνατότητα να φτιάξει ένας οργανισμός/εταιρεία/ιδιώτης μικρά συστήματα που θα είναι ειδικά σχεδιασμένα μόνο για αυτό. Αυτό ανεβάζει τον πήχη των προσφερόμενων υπηρεσιών, αφού η ταχύτητα και η χαμηλή ενεργειακή κατανάλωση είναι σημαντικοί παράγοντες και στην κοινωνική συνείδηση και στην ελεύθερη αγορά που ψάχνει τρόπους να αυξήσει τις υπηρεσίες της, ενώ παράλληλα να ελαχιστοποιήσει τα έξοδά της. Κάτι τέτοιο εύκολα γίνεται αισθητό με την δική μας υλοποίηση καθώς και πιο γρήγορο είναι το πρόγραμμά μας, αλλά και απαιτεί λιγότερη ενέργεια, μνήμη και επεξεργαστική ισχύ. Αξίζει να

σημειωθεί ότι ο vtd-xml ανανεώνεται αρκετά συχνά προσφέροντας ανανεώσεις τόσο στον κώδικά του, που έχουν σημαντικές βελτιώσεις στην ταχύτητά του, όσο και στις δυνατότητές του. Ο συνδυασμός αυτός, μαζί με την ίδιότητα που έχει να είναι ένα από τους ελάχιστους τρόπους που μπορεί να εισαχθεί η ανάλυση xml αρχείων σε ολοκληρωμένα, δίνει σημαντικές προοπτικές υιοθέτησης του κώδικα μας.

5.3 Δομή Υλοποίησης

Ο κώδικας της υλοποίησης απαρτίζεται από λίγα αρχεία. Παρακάτω ορίζεται η λειτουργία των αρχείων αυτών

- ReadBpel.java είναι το αρχικό μας αρχείο, αυτό που ξεκινά την ανάγνωση του αρχείου WS-BPEL και αρχίζει σειριακά να το αναλύει. Το κάθε τμήμα του WS-BPEL αρχείου αποστέλλεται στο κάτωθι αρχείο.

- Runlogic.java είναι το αρχείο το οποίο στην ουσία είναι και ο πυρήνας της εφαρμογής μας. Κάθε σημείο που έρχεται από την ReadBpel.java έρχεται αδώ και αναλύεται για να υλοποιηθεί. Ανάλογα με το τμήμα, η runlogic εκτελεί και κάτι διαφορετικό.

- Createwsdl2javafiles.java είναι το αρχείο το οποίο χρησιμοποιείται εσωτερικά από την runlogic στο κομμάτι της δημιουργίας των μεταβλητών (variables). Συλλέγει τις υπηρεσίες που καλεί η WS-BPEL, μέσω των WSDL αρχείων τους και στη συνέχεια φτιάχνει τις μεταβλητές ως αρχεία java που θα χρησιμοποιηθούν καθώς και άλλα αρχεία τα οποία είναι χρήσιμα κατά την εκτέλεση του κώδικα. Εσωτερικά η ίδια τα μεταφράζει και σε κλάσεις χρησιμοποιώντας την CreateClassesFromJavaFiles

Συνεπώς ο κώδικάς μας αποτελείται από αυτά τα 4 αρχεία τα οποία υλοποιούν τα απαιτούμενα.

Ο κώδικας έχει σχόλια έτσι, ώστε κάθε προγραμματιστής να μπορεί να τον καταλάβει γρήγορα και να μπορεί να τον τροποποιήσει ακόμα πιο γρήγορα. Μόνη προϋπόθεση του σχολιασμού είναι να γνωρίζει αρκετά καλά και το αντικείμενο εφαρμογής έτσι, ώστε να καταλαβαίνει και τι τελείται σε κάθε σημείο, καθώς επίσης και

να καταλαβαίνει την αγγλική γλώσσα, μιας και η διεθνής σύμβαση σχολιασμού των προγραμμάτων που προορίζονται σε παγκόσμιο κοινό είναι η χρήση της αγγλικής γλώσσας.

Σε αυτό το σημείο πρέπει να σημειωθεί ότι ο κώδικάς δεν έχει την υλοποίηση ορισμένων στοιχείων τα οποία είναι ελάσσονος σημασίας, τα οποία είναι τα if, faulthandlers και throw.

Επίσης, η χρήση της receive και της reply δεν είναι αναγκαίες καθώς η υλοποίησή μας είναι για σύγχρονη επικοινωνία υπηρεσιών.

5.4 Παράδειγμα χρήσης

Έστω το παρακάτω bpel αρχείο:

```
<?xml version="1.0" encoding="UTF-8"?>
<process name="newProcess"
  targetNamespace="http://enterprise.netbeans.org/bpel/Diplomatiki_Nick/newProcess"
  xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:sxt="http://www.sun.com/wsbpel/2.0/process/executable/SUNExtension/Trace"
  xmlns:sxed="http://www.sun.com/wsbpel/2.0/process/executable/SUNExtension/Editor"
  xmlns:tns="http://enterprise.netbeans.org/bpel/Diplomatiki_Nick/newProcess"
  xmlns:impl="urn:fibonacci">

  <import namespace="urn:add" location="add.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"/>
  <import namespace="urn:fibonacci" location="fib.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"/>
  <import namespace="urn:returnresult" location="returnresult.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"/>
  <import namespace="urn:square" location="square.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"/>
  <import namespace="urn:createrandom" location="createrandom.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"/>

  <partnerLinks>
```

```
    <partnerLink name="PartnerLink2"
xmlns:tns="http://enterprise.netbeans.org/bpel/createrandomWrapper"
partnerLinkType="tns:CreaterandomLinkType" partnerRole="CreaterandomRole"/>
    <partnerLink name="FIB" xmlns:tns="http://enterprise.netbeans.org/bpel/fibWrapper"
partnerLinkType="tns:FibonacciLinkType" partnerRole="FibonacciRole"/>
    <partnerLink name="STRING"
xmlns:tns="http://enterprise.netbeans.org/bpel/returnresultWrapper"
partnerLinkType="tns:ReturnresultstringLinkType" partnerRole="ReturnresultstringRole"/>
    <partnerLink name="SQUARE"
xmlns:tns="http://enterprise.netbeans.org/bpel/squareWrapper"
partnerLinkType="tns:SquareLinkType" partnerRole="SquareRole"/>
    <partnerLink name="PartnerLink1" xmlns:tns="http://enterprise.netbeans.org/bpel/fibWrapper"
partnerLinkType="tns:FibonacciLinkType" partnerRole="FibonacciRole"/>
  </partnerLinks>

  <variables>
    <variable name="CreaterandomOut" xmlns:impl="urn:createrandom"
messageType="impl:createrandomResponse"/>
    <variable name="CreaterandomIn" xmlns:impl="urn:createrandom"
messageType="impl:createrandomRequest"/>
    <variable name="SquareOut" xmlns:impl="urn:square"
messageType="impl:squareResponse"/>
    <variable name="SquareIn" xmlns:impl="urn:square" messageType="impl:squareRequest"/>
    <variable name="CalculateFibonacciOut" xmlns:impl="urn:fibonacci"
messageType="impl:calculateFibonacciResponse"/>
    <variable name="CalculateFibonacciIn" xmlns:impl="urn:fibonacci"
messageType="impl:calculateFibonacciRequest"/>
    <variable name="Variable2" type="xsd:int"/>
    <variable name="Variable1" type="xsd:int"/>
  </variables>

  <sequence>
    <invoke name="Invoke6" partnerLink="PartnerLink2" operation="createrandom"
xmlns:impl="urn:createrandom" portType="impl:Createrandom" inputVariable="CreaterandomIn"
outputVariable="CreaterandomOut"/>
    <assign name="Assign2">
```

```
        <copy>
          <from variable="CreaterandomOut"/>
          <to variable="SquareIn"/>
        </copy>
      </assign>
      <flow name="Flow1">
        <sequence name="FlowSequence">
          <invoke name="Invoke5" partnerLink="SQUARE" operation="square"
xmlns:impl="urn:square" portType="impl:Square" inputVariable="SquareIn"
outputVariable="SquareOut"/>
        </sequence>
      </flow>
      <assign name="Assign1">
        <copy>
          <from variable="SquareOut"/>
          <to variable="SquareIn"/>
        </copy>
      </assign>
      <invoke name="Invoke5" partnerLink="SQUARE" operation="square"
xmlns:impl="urn:square" portType="impl:Square" inputVariable="SquareIn"
outputVariable="SquareOut"/>
    </sequence>

</process>
```

Εάν έχουμε δύο υπηρεσίες που η μία μας επιστρέφει έναν τυχαίο αριθμό μεταξύ 0 και 10, ενώ η δεύτερη μας επιστρέφει το τετράγωνο ενός αριθμού, ο παραπάνω κώδικας καλεί την πρώτη και έπειτα καλεί δύο φορές διαδοχικά την δεύτερη υπηρεσία. Στο τέλος καταλήγουμε με την τέταρτη δύναμη του τυχαίου αριθμού.

Ο κώδικάς μας για να τρέξει χρειάζεται ως πρώτο όρισμα τον φάκελο που είναι αποθηκευμένοι τα bpel αρχεία μαζί με τα wsdls αρχεία, ως δεύτερο όρισμα το bpel αρχείο και ως τρίτο την τιμή false (λέει στην μηχανή να μην αντιστοιχήσει από μόνη της τα namespaces). Ένα τυχαίο αποτέλεσμα (καθότι ο αριθμός που μας επιστρέφεται από την πρώτη υπηρεσία είναι τυχαίος), που μας δείχνει κάθετι που γίνεται στην πλατφόρμα είναι το εξής:

```
outer Element name ==> process
Trying to wait for the wsdl2java procedure to stop
Parsing XML file:  ./services/square.wsdl
Parsing XML file:  ./services/fib.wsdl
Parsing XML file:  ./services/add.wsdl
Parsing XML file:  ./services/createrandom.wsdl
Parsing XML file:  ./services/returnresult.wsdl
Generating add\_addIntegers.java
Generating returnresult\_returnresultstring.java
Generating square_pkg\_square.java
Generating fibonacci\_calculateFibonacciRangeResponse.java
Generating createrandom_pkg\_createrandom.java
Generating add\_addIntegersResponse.java
Generating square_pkg\_squareResponse.java
Generating fibonacci\_calculateFibonacciResponse.java
Generating createrandom_pkg\_createrandomResponse.java
Generating returnresult\_returnresultstringResponse.java
AddSoapBindingImpl.java already exists, WSDL2Java will not overwrite
it.
Generating add\Add.java
SquareSoapBindingImpl.java already exists, WSDL2Java will not
overwrite it.
Generating square_pkg\Square.java
Generating add\AddSoapBindingStub.java
CreaterandomSoapBindingImpl.java already exists, WSDL2Java will not
overwrite it.
Generating createrandom_pkg\Createrandom.java
Generating square_pkg\SquareSoapBindingStub.java
Generating returnresult\ReturnresultstringService.java
Generating fibonacci\_calculateFibonacciRange.java
Generating add\AddSoapBindingSkeleton.java
Generating square_pkg\SquareSoapBindingSkeleton.java
Generating createrandom_pkg\CreaterandomSoapBindingStub.java
Generating fibonacci\_calculateFibonacci.java
Generating returnresult\ReturnresultstringServiceLocator.java
Generating square_pkg\SquareService.java
Generating add\AddService.java
Generating fibonacci\FibonacciService.java
ReturnresultSoapBindingImpl.java already exists, WSDL2Java will not
overwrite it.
Generating returnresult\Returnresultstring.java
Generating fibonacci\FibonacciServiceLocator.java
Generating returnresult\ReturnresultSoapBindingStub.java
Generating createrandom_pkg\CreaterandomSoapBindingSkeleton.java
Generating add\AddServiceLocator.java
FibonacciSoapBindingImpl.java already exists, WSDL2Java will not
overwrite it.
Generating fibonacci\Fibonacci.java
Generating add\deploy.wsdd
Generating createrandom_pkg\CreaterandomService.java
Generating square_pkg\SquareServiceLocator.java
Generating createrandom_pkg\CreaterandomServiceLocator.java
Generating fibonacci\FibonacciSoapBindingStub.java
Generating returnresult\ReturnresultSoapBindingSkeleton.java
Generating square_pkg\deploy.wsdd
Generating createrandom_pkg\deploy.wsdd
Generating add\undeploy.wsdd
```

```
Generating returnresult\deploy.wsdd
Generating createrandom_pkg\undeploy.wsdd
Generating fibonacci\FibonacciSoapBindingSkeleton.java
Generating returnresult\undeploy.wsdd
Generating square_pkg\undeploy.wsdd
Generating fibonacci\deploy.wsdd
Generating fibonacci\undeploy.wsdd
square_pkg\SquareSoapBindingStub.java:27: warning: as of release 5,
'enum' is a keyword, and may not be used as an identifier
(use -source 5 or higher to use 'enum' as a keyword)
        oper.setStyle(org.apache.axis.enum.Style.DOCUMENT);
                                              ^
square_pkg\SquareSoapBindingStub.java:28: warning: as of release 5,
'enum' is a keyword, and may not be used as an identifier
(use -source 5 or higher to use 'enum' as a keyword)
        oper.setUse(org.apache.axis.enum.Use.ENCODED);
                                           ^
createrandom_pkg\CreaterandomSoapBindingStub.java:27: warning: as of
release 5, 'enum' is a keyword, and may not be used as an identifier
(use -source 5 or higher to use 'enum' as a keyword)
        oper.setStyle(org.apache.axis.enum.Style.DOCUMENT);
                                              ^
createrandom_pkg\CreaterandomSoapBindingStub.java:28: warning: as of
release 5, 'enum' is a keyword, and may not be used as an identifier
(use -source 5 or higher to use 'enum' as a keyword)
        oper.setUse(org.apache.axis.enum.Use.ENCODED);
                                           ^
add\AddSoapBindingStub.java:27: warning: as of release 5, 'enum' is a
keyword, and may not be used as an identifier
(use -source 5 or higher to use 'enum' as a keyword)
        oper.setStyle(org.apache.axis.enum.Style.DOCUMENT);
                                              ^
add\AddSoapBindingStub.java:28: warning: as of release 5, 'enum' is a
keyword, and may not be used as an identifier
(use -source 5 or higher to use 'enum' as a keyword)
        oper.setUse(org.apache.axis.enum.Use.ENCODED);
                                           ^
2 warnings
2 warnings
.\returnresult\ReturnresultSoapBindingStub.java:27: warning: as of
release 5, 'enum' is a keyword, and may not be used as an identifier
(use -source 5 or higher to use 'enum' as a keyword)
        oper.setStyle(org.apache.axis.enum.Style.DOCUMENT);
                                              ^
.\returnresult\ReturnresultSoapBindingStub.java:28: warning: as of
release 5, 'enum' is a keyword, and may not be used as an identifier
(use -source 5 or higher to use 'enum' as a keyword)
        oper.setUse(org.apache.axis.enum.Use.ENCODED);
                                           ^
2 warnings
.\fibonacci\FibonacciSoapBindingStub.java:27: warning: as of release
5, 'enum' is a keyword, and may not be used as an identifier
(use -source 5 or higher to use 'enum' as a keyword)
        oper.setStyle(org.apache.axis.enum.Style.DOCUMENT);
                                              ^
.\fibonacci\FibonacciSoapBindingStub.java:28: warning: as of release
5, 'enum' is a keyword, and may not be used as an identifier
```

```
(use -source 5 or higher to use 'enum' as a keyword)
        oper.setUse(org.apache.axis.enum.Use.ENCODED);
                                    ^
.\fibonacci\FibonacciSoapBindingStub.java:37: warning: as of release
5, 'enum' is a keyword, and may not be used as an identifier
(use -source 5 or higher to use 'enum' as a keyword)
        oper.setStyle(org.apache.axis.enum.Style.DOCUMENT);
                                     ^
.\fibonacci\FibonacciSoapBindingStub.java:38: warning: as of release
5, 'enum' is a keyword, and may not be used as an identifier
(use -source 5 or higher to use 'enum' as a keyword)
        oper.setUse(org.apache.axis.enum.Use.ENCODED);
                                    ^
2 warnings
returnresult\ReturnresultSoapBindingStub.java:27: warning: as of
release 5, 'enum' is a keyword, and may not be used as an identifier
(use -source 5 or higher to use 'enum' as a keyword)
        oper.setStyle(org.apache.axis.enum.Style.DOCUMENT);
                                     ^
returnresult\ReturnresultSoapBindingStub.java:28: warning: as of
release 5, 'enum' is a keyword, and may not be used as an identifier
(use -source 5 or higher to use 'enum' as a keyword)
        oper.setUse(org.apache.axis.enum.Use.ENCODED);
                                    ^
4 warnings
fibonacci\FibonacciSoapBindingStub.java:27: warning: as of release 5,
'enum' is a keyword, and may not be used as an identifier
(use -source 5 or higher to use 'enum' as a keyword)
        oper.setStyle(org.apache.axis.enum.Style.DOCUMENT);
                                     ^
fibonacci\FibonacciSoapBindingStub.java:28: warning: as of release 5,
'enum' is a keyword, and may not be used as an identifier
(use -source 5 or higher to use 'enum' as a keyword)
        oper.setUse(org.apache.axis.enum.Use.ENCODED);
                                    ^
fibonacci\FibonacciSoapBindingStub.java:37: warning: as of release 5,
'enum' is a keyword, and may not be used as an identifier
(use -source 5 or higher to use 'enum' as a keyword)
        oper.setStyle(org.apache.axis.enum.Style.DOCUMENT);
                                     ^
fibonacci\FibonacciSoapBindingStub.java:38: warning: as of release 5,
'enum' is a keyword, and may not be used as an identifier
(use -source 5 or higher to use 'enum' as a keyword)
        oper.setUse(org.apache.axis.enum.Use.ENCODED);
                                    ^
Trying to wait for the wsdl2java procedure to stop
2 warnings
4 warnings
Trying to wait for the wsdl2java procedure to stop
Trying to wait for the wsdl2java procedure to stop
Trying to wait for the wsdl2java procedure to stop
urn:createrandom
So far      I've got filetolook:./services/createrandom.wsdl, class
before the one to look: createrandomResponse
Found where I'm supposed to look createrandom._createrandomResponse
and gonna load it dynamically.
Finding class
```

```
createrandom_pkg._createrandomResponse: Class loaded successfully
urn:createrandom
So far      I've got filetolook:./services/createrandom.wsdl, class
before the one to look: createrandomRequest
Found where I'm supposed to look createrandom._createrandom and gonna
load it dynamically.
Finding class
createrandom_pkg._createrandom: Class loaded successfully
urn:square
So far      I've got filetolook:./services/square.wsdl, class before
the one to look: squareResponse
Found where I'm supposed to look square._squareResponse and gonna load
it dynamically.
Finding class
square_pkg._squareResponse: Class loaded successfully
urn:square
So far      I've got filetolook:./services/square.wsdl, class before
the one to look: squareRequest
Found where I'm supposed to look square._square and gonna load it
dynamically.
Finding class
square_pkg._square: Class loaded successfully
urn:fibonacci
So far      I've got filetolook:./services/fib.wsdl, class before the
one to look: calculateFibonacciResponse
Found where I'm supposed to look fibonacci._calculateFibonacciResponse
and gonna load it dynamically.
Finding class
fibonacci._calculateFibonacciResponse: Class loaded successfully
urn:fibonacci
So far      I've got filetolook:./services/fib.wsdl, class before the
one to look: calculateFibonacciRequest
Found where I'm supposed to look fibonacci._calculateFibonacci and
gonna load it dynamically.
Finding class
fibonacci._calculateFibonacci: Class loaded successfully
CreaterandomOut   createrandom_pkg._createrandomResponse
      createrandom.wsdl
CreaterandomIn    createrandom_pkg._createrandom
      createrandom.wsdl
SquareOut   square_pkg._squareResponse    square.wsdl
SquareIn    square_pkg._square      square.wsdl
CalculateFibonacciOut   fibonacci._calculateFibonacciResponse
      fib.wsdl
CalculateFibonacciIn    fibonacci._calculateFibonacci fib.wsdl
Variable2   java.lang.Integer noFile
Variable1   java.lang.Integer noFile
Variables section ended...



Found CreaterandomIn in variablesList in position (starting from 0) 3
Found CreaterandomOut in variablesList in position (starting from 0) 0
Finding class
createrandom_pkg.CreaterandomServiceLocator: Class loaded successfully
Calling service @ http://localhost:8080/axis/services/createrandom
```

```
Beginning to transmit....
Please wait....
Invoke returned=8
Type of variable to pass is of:int and method's name is
setCreaterandomReturn
Invoke ending...
Invoke with nameInvoke6 section ended...


Going to child(from assign)
Found CreaterandomOut in variablesList in position (starting from 0) 0
Found SquareIn in variablesList in position (starting from 0) 9
Assign says that those two variables have the same amount of parts...
so gonna copy them
Method's name is getCreaterandomReturn
Type of variable to pass is of:int and method's name is setIn0
8
Ended with assign command


Going to children (from flow with name Flow1)
Got from flow index number 185 and
filelocation:./services/testcase3.bak.bpel
Found index with number 185 so I'm gonna stop.
Trying to wait for them to stop
Going to children (from sequence with name FlowSequence)
Found SquareIn in variablesList in position (starting from 0) 9
Found SquareOut in variablesList in position (starting from 0) 6
Finding class
square_pkg.SquareServiceLocator: Class loaded successfully
Calling service @ http://localhost:8080/axis/services/square
Beginning to transmit....
Please wait....
Invoke returned=64
Type of variable to pass is of:int and method's name is
setSquareReturn
Invoke ending...
Invoke with nameInvoke5 section ended...


Going to child(from assign)
Found SquareOut in variablesList in position (starting from 0) 6
Found SquareIn in variablesList in position (starting from 0) 9
Assign says that those two variables have the same amount of parts...
so gonna copy them
Method's name is getSquareReturn
Type of variable to pass is of:int and method's name is setIn0
64
Ended with assign command


Found SquareIn in variablesList in position (starting from 0) 9
Found SquareOut in variablesList in position (starting from 0) 6
Finding class
square_pkg.SquareServiceLocator: Class loaded successfully
Calling service @ http://localhost:8080/axis/services/square
Beginning to transmit....
```

```
Please wait....
Invoke returned=4096
Type of variable to pass is of:int and method's name is
setSquareReturn
Invoke ending...
Invoke with nameInvoke5 section ended...
```

Και όντως η 4η δύναμη του 8 είναι το 4096. Αξίζει να σημειωθεί ότι, για να υπάρχει συμβατότητα μεταξύ των εργαλείων που χρησιμοποιήθηκαν, έπρεπε να χρησιμοποιηθεί η έκδοση 1.4 της java, γεγονός που όταν εκτελείται σε 1.6 περιβάλλον, εγείρει προειδοποιητικά μηνύματα. Στο παράρτημα φαίνεται ο κώδικας της υλοποίησής μας.

Παράρτημα Α

Ο κώδικας της αρχικού java αρχείου, το οποίο αναλαμβάνει να ανοίγει το αρχείο και να μετακινεί τον κέρσορα κατά ένα στοιχείο κάθε φορά, είναι:

```
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.Callable;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;

import com.ximpleware.*;

public class ReadBpel {
	static List <Object> variablesList=new ArrayList();	//it is
the List that will hold any variable that is going to be used in the
whole process
	static List <Object> receiveList=new ArrayList();	//it is
the List that awaits for objects to be returned
	static List <String> namespacesResolved=new ArrayList();

	public static void main(String[] args) throws Exception{

		String filelocation="";
		boolean filehasnamespaces=true;

		if (args.length==3){
			filelocation=args[0]+args[1];
			if (args[2].equalsIgnoreCase("true"))
				filehasnamespaces=true;
			else filehasnamespaces=false;
		} else {
			System.out.println(args.length+"Usage: Readbpel
(BPEL file location) (BPEL has namespaces:type true or anything else
for false)\n");
			System.exit(1);
		}

	VTDGen vg = new VTDGen();
	// declareXPathNameSpace associates name space with a prefix
	  //ap.declareXPathNameSpace(namespace,namespaceURI);//"xsd","ht
tp://www.w3.org/2001/XMLSchema"
	  /* File temp=new File(filelocation);
	  System.out.println("File exists:"+temp.exists());
	  System.out.println(vg.parseFile("./bin/SynchronousSample.bpel"
, true));*/
	if (vg.parseFile(filelocation, filehasnamespaces)){
	//"./src/SynchronousSample.wsdl","true"
		VTDNav vn = vg.getNav();
		AutoPilot ap = new AutoPilot(vn);
		try {
				ap.selectXPath("/process");//found the root
element
				// AutoPilot moves the cursor for you, as it
returns the index value of the qualified node
```

```
                        if (ap.evalXPath()!=-1){      //means that it
found a process element and moves the autopilot there
                                System.out.println("outer Element name
==> " +vn.toString(vn.getCurrentIndex()) );
                                vn.toElement(VTDNav.FIRST_CHILD);//go to
the first child so as to start the parsing of the program
                                //begin iteration
                                do {
                                        final ExecutorService execSvc1 =
Executors.newFixedThreadPool(30);
                                        Callable<String>
executionCallable1 = new runlogic(vn, args[0], args[1],
filehasnamespaces); //go to runlogic function where each tag of the
sequence is processed
                                        Future<String> future1 =
execSvc1.submit(executionCallable1);
                                        future1.get();
                                } while
(vn.toElement(VTDNav.NEXT_SIBLING));
                                System.exit(0);
                        }
            } catch (Exception e) {
                        System.out.println("XPathParseException");//
in case something went wrong in XPATH
                        e.printStackTrace();
                }
      } else {
            System.out.println("File could not be parsed! Probably not
a true bpel file (e.g. bad tagging)! \n");
      };
    }

      public static void sendReceive(String operation, Object object){
            receiveList.add(operation);   //the name of the operation
            receiveList.add(object);      //the variable that has been
sent back
      }
}
```

Στην συνέχεια ακολουθεί ο πυρήνας του προγράμματος, το οποίο υλοποιεί κάθε στοιχείο, η runlogic.java

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.List;
import java.util.Locale;
import java.util.TimeZone;
import java.util.concurrent.Callable;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.ExecutorService;
```

```
import java.util.concurrent.Executors;
import java.util.concurrent.Future;
import java.util.concurrent.FutureTask;
import java.util.concurrent.LinkedBlockingQueue;
import java.util.concurrent.ThreadFactory;
import java.util.concurrent.ThreadPoolExecutor;
import java.util.concurrent.TimeUnit;

import java.io.File;
import java.lang.Long;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLClassLoader;
import java.util.Calendar;

import javax.xml.datatype.DatatypeConfigurationException;
import javax.xml.datatype.DatatypeFactory;
import javax.xml.namespace.QName;

import org.apache.axis.client.Call;
import org.apache.axis.client.Service;

import sun.util.resources.CalendarData_el;

import wsdl2java.createwsdl2javafiles;

import bpelactionscontrol.documentation;
import bpelactionscontrol.invoke;

import com.ximpleware.AutoPilot;
import com.ximpleware.NavException;
import com.ximpleware.PilotException;
import com.ximpleware.VTDGen;
import com.ximpleware.VTDNav;

class runlogic implements Callable <String>{


            int curindex;
            VTDNav temp;
            VTDGen vg3 = new VTDGen();
      String filefolder, filename, filelocation;
      boolean filehasnamespaces;
        VTDGen vg = new VTDGen(); //in case some one needs it
(like invoke or receive)
        //TO-DO perhaps this list should be accessed by ReadBpel
because it is that has the responsibility to go to the next node...
so I don;t want it to create another list each time... or does it
not create another child?
        static List <String> namespacesResolved=new ArrayList();
        List <String> namespacesResolved2=new ArrayList();
        static List <Object> variablesList=new ArrayList();
```

```
        static List <Object> receiveList=new ArrayList();

        //1st constructor
        public runlogic(VTDNav temp, String filefolder, String
filename, boolean filehasnamespaces){
            this.temp=temp;
            this.filefolder=filefolder;
            this.filename=filename;
            this.filehasnamespaces=filehasnamespaces;
            filelocation=filefolder+filename;
        }

        //2nd constructor
        //needed for flow command
        //needed for concurrency as one cannot take the pointer and
alter it, cause they all share one common pointer in the previous
way, so changing it would result in chaotic results
        public runlogic(int curindex, String filefolder, String
filename, boolean filehasnamespaces) throws PilotException,
NavException{
             this.curindex=curindex;
             this.filename=filename;
            this.filefolder=filefolder;
            filelocation=filefolder+filename;
             this.filehasnamespaces=filehasnamespaces;
             System.out.println("Got from flow index number
"+this.curindex+ " and filelocation:"+ this.filelocation);
//             System.out.println("Found flow with int index "+
curindex+ " so I'm gonna attempt to start a new navigator from
there...");
             if (vg3.parseFile(filelocation, filehasnamespaces)){
                  //"./src/SynchronousSample.wsdl","true"
                  VTDNav vnena=vg3.getNav();   //problem because
there can be only one!
                  AutoPilot apena = new AutoPilot(vnena); // ap is
going to move the cursor
                  apena.selectElement("*"); // select every element
(* matches all strings)
                  while(vnena.getCurrentIndex()!=this.curindex) //
iterate will iterate thru all elements
                  {
                        apena.iterate();
                  }
                  temp=vnena;
                  System.out.println("Found index with number
"+temp.getCurrentIndex()+" so I'm gonna stop.");
//                  System.out.println("Done. Starting a new navigator
from index"+temp.getCurrentIndex());
            } else {
               System.out.println("Could not parse file:
"+this.filelocation);
            }
        }
```

```
            public String call() throws IllegalAccessException,
InstantiationException, MalformedURLException,
ClassNotFoundException{
            try {

                  //NOTE TO SELF:: Multiple ifs not good, better is
else if, even better the switch/case(which probably can be done via
enumeration?!!!!


//-------------------------------------------------------------------
-----------------------------------------
                  //import
                  if (temp.matchElement("import")){ //finished!

                                    ThreadPoolExecutor threadpool = new
ThreadPoolExecutor(5,5, 50000L, TimeUnit.MILLISECONDS,new
LinkedBlockingQueue<Runnable>(), new PoolHandlerFactory());
                                    final ExecutorService execSvc =
Executors.newFixedThreadPool(1);
                                    ArrayList<FutureTask> t = new
ArrayList<FutureTask>();

                                    //so, for every import I will give it
an FutureTask so as to perform simultaneously
                                    do {
                                          String
attrnamespace=temp.toNormalizedString(temp.getAttrVal("namespace"))
;
                                    String
attrlocation=temp.toNormalizedString(temp.getAttrVal("location"));
                                          String
attrimportType=temp.toNormalizedString(temp.getAttrVal("importType"
));
                                          if
(attrimportType.equalsIgnoreCase("http://schemas.xmlsoap.org/wsdl/"
)){
                                                t.add(new
FutureTask<String> (new
createwsdl2javafiles(filefolder+attrlocation))); // Create the
classes in their respective location);
                                          }

                                    } while
(temp.toElement(VTDNav.NEXT_SIBLING) &&
temp.matchElement("import"));

                                    //starting to parallelize the threads
                                    for (int i = 0; i < t.size(); i++) {
                                        threadpool.execute(t.get(i));
                                    }
```

```
                                for (int i = 0; i < t.size(); i++) {
                                      try{
                                            System.out.println("Trying
to wait for the wsdl2java procedure to stop");
                                            t.get(i).get();
                                      }catch (InterruptedException ee)
{} catch (ExecutionException e) {
                                            // TODO Auto-generated
catch block
                                            e.printStackTrace();
                                      }
                                }
                                t.clear();

                  }
//-----------------------------------------------------------------
-----------------------------------------
                  else if (temp.matchElement("variables")){ //almost
done --in case of three attributes I'm not sure if it is correct
                        Thread.sleep(2000);
                        if(temp.toElement(VTDNav.FIRST_CHILD)){
                              String
whereto look="",whatvariabletolook="", filetolook="",
classtoload="", almostreadyclass="", almostthere[];
//                            ThreadPoolExecutor threadpool =
new ThreadPoolExecutor(5,5, 50000L, TimeUnit.MILLISECONDS,new
LinkedBlockingQueue<Runnable>(), new PoolHandlerFactory());
//                            final ExecutorService execSvc =
Executors.newFixedThreadPool(1);
//                            ArrayList<FutureTask> t = new
ArrayList<FutureTask>();
                              Class var1 = null, var2 = null, var3 =
null;

                              do {
                                    String
attrname=temp.toNormalizedString(temp.getAttrVal("name"));
                                    int
attrcount=temp.getAttrCount();
                                    switch (attrcount){
                                    case 2:
                                    {
                                          if
(temp.getAttrVal("type")!=-1){     //if there is no such element
throws back an -1
                                                String
attrType=temp.toNormalizedString(temp.getAttrVal("type"));     //e.
g. <variable name="UploadOpRes" type="xsd:anyType"/>
//                                              t.add(new
FutureTask<Object> (new Variables2Objects(attrname, attrType)));


      variablesList.add(attrname);
                                                if
```

```
(attrType.equalsIgnoreCase("xsd:int")){
                                                    int
classvariable = 0;

      variablesList.add(classvariable);
                                                    }else if
(attrType.equalsIgnoreCase("xsd:long")){
                                                          long
classvariable = 0;

      variablesList.add(classvariable);
                                                    }
                                                    else if
(attrType.equalsIgnoreCase("xsd:short")){
                                                          short
classvariable = 0;//short typeclass;

      variablesList.add(classvariable);
                                                    }
                                                    else if
(attrType.equalsIgnoreCase("xsd:float")){
                                                          float
classvariable = 0;//float typeclass;

      variablesList.add(classvariable);
                                                    }
                                                    else if
(attrType.equalsIgnoreCase("xsd:double")){
                                                          double
classvariable = 0;//double typeclass;

      variablesList.add(classvariable);
                                                    }
                                                    else if
(attrType.equalsIgnoreCase("xsd:boolean")){
                                                          boolean
classvariable = false;//boolean typeclass;

      variablesList.add(classvariable);
                                                    }
                                                    else if
(attrType.equalsIgnoreCase("xsd:byte")){
                                                          byte
classvariable = 0;//byte typeclass;

      variablesList.add(classvariable);
                                                    }
                                              else {
                                                    Class var =
xsd2type(attrType);
                                                    Object
classvariable = var.newInstance();
```

```
      variablesList.add(classvariable);
                                                }

      variablesList.add("noFile");

                                                break;      //means
that it will pass cases 3-5
                                            }

                                                if
(temp.getAttrVal("messageType")!=-1){
                                                    String
attrMType=temp.toNormalizedString(temp.getAttrVal("messageType"));
                                                    //find where to
look for the operation from the process attribute
                                                    String[]
tokenize = attrMType.split(":");
                                                    if
(tokenize.length==2){  //means that it has 2 tokens, one of the
namespace and one of the portType

      whatvariabletolook=tokenize[1];

                                                        //resolve
namespace from variable and see what connection has with <process>
                                                        for (int
i=0; i<namespacesResolved.size(); i+=2){      //namespacesResolved
has in every *second* line the namespace(impl) and in every other
line the location of the namespace (e.g. "urn:fibonacci"). That is
normally located as an attribute to the "process" child
                                                            if
(tokenize[0].equals(namespacesResolved.get(i))){    //xmlns:impl
from tag

      wheretolook=namespacesResolved.get(i+1);      //e.g.
urn:fibonacci from process

      break;      //found it so it will quit for loop
                                                            }
                                                        };
                                                        if
(wheretolook.isEmpty()){     //if previous for loop didn't give any
result from the list where we keep all current namespaces
                                                            if
(vg.parseFile(filelocation, false)){                //false:do not
separate namespaces

      VTDNav vn = vg.getNav();

      AutoPilot ap = new AutoPilot(vn);
                                                            try
{
```

```
      ap.selectXPath("/process");//

            // AutoPilot moves the cursor for you, as it returns
the index value of the qualified node

            if (ap.evalXPath()!=-1){     //means that it found a
sequence and moves the autopilot there

                  if (vn.getAttrVal("xmlns:"+tokenize[0])!=-1){
      //means that there is a valid namepsace in process



whereTolook=vn.toNormalizedString(vn.getAttrVal("xmlns:"+tokenize[0
]));

                        namespacesResolved.add(tokenize[0]);
      //e.g. "impl"

                        namespacesResolved.add(whereTolook);
      //e.g. "urn:fibonacci"

                  }

                  else System.out.println("Did not find "+
tokenize[0] +" from the variables section to the process section");

            }
                                                                  }
catch(Exception e) {

            System.out.println("XPathParseException");// in case
something went wrong in XPATH

            e.printStackTrace();
                                                                  }
                                                                  }
else {

      System.out.println("Could not open file...");
                                                                  }
                                                            }
                                                      }



                                                      //I've got the
namespace connection from the variable=>process, now I'm gonna find
the connection from process=>import and see what file it
corresponds to
                                                      String []
location = whereTolook.split(":");//location[0]==urn and
location[1]==fibonacci
                                                      if
```

```
(location[0].equalsIgnoreCase("http")){
                                                        //TO-DO...
this is the case that the process gives back an http link
                                                    } else {
                                                        //resolve
namespace from import and see what connection has with a file
                                                        for (int
i=0; i<namespacesResolved2.size(); i+=2){     //namespacesResolved2
has in every *second* line the namespace(process) and in every
other line the file which it corresponds to (e.g.
"urn:fibonacci=>fib.wsdl")
                                                            if
(wheretolook.equals(namespacesResolved2.get(i))){   //e.g
wheretolook=urn:fibonacci

      filetolook=namespacesResolved2.get(i+1);      //e.g. fib.wsdl
from import

      break;      //found it so it will quit for loop
                                                            }
                                                        };
                                                        if
(filetolook.isEmpty()){
                                                            if
(vg.parseFile(filelocation, false)){                //false:do not
separate namespaces

      VTDNav vn = vg.getNav();

      AutoPilot ap = new AutoPilot(vn);
                                                            try
{

      ap.selectXPath("/process/import");//

            // AutoPilot moves the cursor for you, as it returns
the index value of the qualified node

            if (ap.evalXPath()!=-1){     //means that it found a
sequence and moves the autopilot there

                  do {

                        if (vn.getAttrVal("namespace")!=-1){
      //means that there is a valid namespace in import

                              if
(vn.toNormalizedString(vn.getAttrVal("namespace")).matches(wheretol
ook)){



      filetolook=vn.toNormalizedString(vn.getAttrVal("location"));
```

```
                                namespacesResolved2.add(wheretolook);

                                namespacesResolved2.add(filetolook);

                                }

                        }

                } while (vn.toElement(VTDNav.NEXT_SIBLING));

                /*if (vn.getAttrVal("xmlns:"+tokenize[0])!=-1){
      //means that there is a valid namespace in process


filetolook=vn.toNormalizedString(vn.getAttrVal("xmlns:"+tokenize[0]
));

                        namespacesResolved2.add(tokenize[0]);

                        namespacesResolved2.add(wheretolook);

                }*/

                /*else if (vn.getAttrVal("namespace")!=-1){
      //means that there is a valid namespace in process


wheretolook=vn.toNormalizedString(vn.getAttrVal("namespace"));

                        System.out.println("Test case... Variable id
found in ");

                        namespacesResolved.add(tokenize[0]);

                        namespacesResolved.add(wheretolook);

                }*/

            }else {

                        System.out.println("Could not find from
import the link to filelocation");

            }


                                                                    }
catch(Exception e) {
```

```
            System.out.println("XPathParseException");// in case
something went wrong in XPATH

            e.printStackTrace();
                                                                    }
                                                                    }
else {

      System.out.println("Could not open file...");
                                                                    }
                                                                }

      System.out.println("So far   I've got
filetolook:./services/"+filetolook+", class before the one to look:
"+tokenize[1]);

                                                                //opening
file so as to find the right class to load, usually it has the same
name but there are times that this is not the case (see fib.wsdl
and calculateFibonacciRequest example -> the class to load is not
calculateFibonacciRequest but calculate Fibonacci
                                                                if
(vg.parseFile("./services/"+filetolook, false)){
      //false:do not separate namespaces

      VTDNav vn = vg.getNav();
                                                                AutoPilot
ap = new AutoPilot(vn);
                                                                try {

      ap.selectXPath("/definitions/message");//

      // AutoPilot moves the cursor for you, as it returns the
index value of the qualified node

      if (ap.evalXPath()!=-1){     //means that it found a message
and moves the autopilot there

            do {

                  if (vn.getAttrVal("name")!=-1){    //means that
there is a valid namespace in import

                        if
(vn.toNormalizedString(vn.getAttrVal("name")).matches(tokenize[1]))
{

                              if (vn.toElement(VTDNav.FIRST_CHILD)){
```

```
almostreadyclass=vn.toNormalizedString(vn.getAttrVal("element"));

      almostthere=almostreadyclass.split(":");
                                        if
(almostthere[0].equalsIgnoreCase(tokenize[0])) {
                                              classtoload=almostthere[1];
                                 }
                                 //classesFromVariablesResolved.add(whe
retolook);
                                 //classesFromVariablesResolved.add(fil
etolook);
                                 vn.toElement(VTDNav.PARENT);
                                 }
                            //break;    //found the one so no need to
check the rest messages
                            }
                     }
               } while (vn.toElement(VTDNav.NEXT_SIBLING));
      }else {
                     System.out.println("Could not find from import the
link to filelocation");
      }
                                                                  }
catch(Exception e) {
      System.out.println("XPathParseException");// in case
something went wrong in XPATH
      e.printStackTrace();
                                                                  }
                                                                  } else {

      System.out.println("Could not open file
./services/"+filetolook);
```

```
                                                                    }


      System.out.println("Found where I'm supposed to look
"+location[1]+"._"+classtoload+" and gonna load it dynamically.");

                                                                    String
currentdir = System.getProperty("user.dir");
                                                                    File
outputdir = new File(currentdir);
                                                                    URL[] urls
= new URL[]{outputdir.toURI().toURL()};

      ClassLoader cl = runlogic.class.getClassLoader();

      System.out.println("Finding class");
                                                                    if (cl !=
null) {
                                                                        //Sy
stem.out.println("Default ClassLoader : " + cl);

      URLClassLoader ucl = new URLClassLoader(urls, cl);
                                                                        File
temp=new File("./"+location[1]+"/_"+classtoload+".class");
                                                                    if
(temp.exists()){
                                                                        var1
= ucl.loadClass(location[1]+"._"+classtoload);
                                                                        }
                                                                    else{
                                                                        var1
= ucl.loadClass(location[1]+"_pkg._"+classtoload);
                                                                    }

      System.out.println(var1.getName()+ ": Class loaded
successfully");

                                                                    }
                                                                    //var1 =
Class.forName(location[1]+"._"+classtoload/*, true,
Thread.currentThread().getContextClassLoader()*/);
                                                                    Object
classvariable = var1.newInstance();
                                                                    //list
that holds in every odd line the name and in every even line the
class object...

      variablesList.add(attrname);

      variablesList.add(classvariable);

      variablesList.add(filetolook);
```

```
                                                        }



                                                    break; //it was
case 2 so there is no point looking for the rest.
                                                }
                                            }
                                            //perhaps this needs
alteration, it must take the xmlns:impl from the same field and not
from the process=>import...
                                            case 3:{
                                                if
(temp.getAttrVal("type")!=-1){     //if there is no such element
throws back an -1
                                                String
attrType=temp.toNormalizedString(temp.getAttrVal("type"));     //e.
g. <variable name="UploadOpRes" type="xsd:anyType"/>
                                                if
(temp.getAttrVal("xsi:nillable")!=-1)
                                                    if
(temp.toNormalizedString(temp.getAttrVal("xsi:nillable")).equals("t
rue")){
                                                        Class
var=xsd2type(attrType+" xsi:nillable=true");
                                                        Object
classvariable= var.newInstance();
                                                    }
                                                break;      //means
that it will pass cases 4-5
                                            }
                                                if
(temp.getAttrVal("messageType")!=-1){
                                                    String
attrMType=temp.toNormalizedString(temp.getAttrVal("messageType"));
                                                    //find where to
look for the operation from the process attribute
                                                    String[]
tokenize = attrMType.split(":");
                                                    if
(tokenize.length==2){  //means that it has 2 tokens, one of the
namespace and one of the name of the variable

      whatvariabletolook=tokenize[1];
                                                        //check if
there is an override for the namespace
                                                        if
(temp.getAttrVal("xmlns:"+tokenize[0])!=-1){

      wheretolook =
temp.toNormalizedString(temp.getAttrVal("xmlns:"+tokenize[0]));
                                                        }
                                                        //otherwis
e take the usual route
```

```
                                                        }


                                                        //I've got the
namespace connection from the variable=>import, now I'm gonna find
the connection from process=>import and see what file it
corresponds to
                                                        String []
location = wheretolook.split(":");//location[0]==urn and
location[1]==fibonacci

      System.out.println(wheretolook);
                                                        if
(location[0].equalsIgnoreCase("http")){
                                                              //TO-DO...
this is the case that the process gives back an http link
                                                        } else {
                                                                    if
(vg.parseFile(filelocation, false)){                    //false:do not
separate namespaces

      VTDNav vn = vg.getNav();

      AutoPilot ap = new AutoPilot(vn);
                                                                    try
{

      ap.selectXPath("/process/import");//

            // AutoPilot moves the cursor for you, as it returns
the index value of the qualified node

            if (ap.evalXPath()!=-1){     //means that it found a
sequence and moves the autopilot there

                  do {

                        if (vn.getAttrVal("namespace")!=-1){
      //means that there is a valid namespace in import

                              if
(vn.toNormalizedString(vn.getAttrVal("namespace")).matches(wheretol
ook)){


      filetolook=vn.toNormalizedString(vn.getAttrVal("location"));

                              namespacesResolved2.add(wheretolook);

                              namespacesResolved2.add(filetolook);
```

```
                                }

                            }

                        } while (vn.toElement(VTDNav.NEXT_SIBLING));


                }else {

                            System.out.println("Could not find from
import the link to filelocation");

                }


                                                                    }
catch(Exception e) {

                System.out.println("XPathParseException");// in case
something went wrong in XPATH

                e.printStackTrace();
                                                                    }
                                                                    }
else {

        System.out.println("Could not open file...");
                                                                    }


        System.out.println("So far   I've got
filetolook:./services/"+filetolook+", class before the one to look:
"+tokenize[1]);

                                                                //opening
file so as to find the right class to load, usually it has the same
name but there are times that this is not the case (see fib.wsdl
and calculateFibonacciRequest example -> the class to load is not
calculateFibonacciRequest but calculate Fibonacci
                                                                if
(vg.parseFile("./services/"+filetolook, false)){
        //false:do not separate namespaces

        VTDNav vn = vg.getNav();
                                                                AutoPilot
ap = new AutoPilot(vn);
                                                                try {

        ap.selectXPath("/definitions/message");//

        // AutoPilot moves the cursor for you, as it returns the
index value of the qualified node
```

```
      if (ap.evalXPath()!=-1){     //means that it found a message
and moves the autopilot there

            do {

                  if (vn.getAttrVal("name")!=-1){    //means that
there is a valid namespace in import

                        if
(vn.toNormalizedString(vn.getAttrVal("name")).matches(tokenize[1]))
{

                              if (vn.toElement(VTDNav.FIRST_CHILD)){



almostreadyclass=vn.toNormalizedString(vn.getAttrVal("element"));


      almostthere=almostreadyclass.split(":");

                                    if
(almostthere[0].equalsIgnoreCase(tokenize[0])) {

                                          classtoload=almostthere[1];

                              }

                              //classesFromVariablesResolved.add(whe
retolook);

                              //classesFromVariablesResolved.add(fil
etolook);

                              vn.toElement(VTDNav.PARENT);

                              }

                        //break;    //found the one so no need to
check the rest messages

                        }

                  }

            } while (vn.toElement(VTDNav.NEXT_SIBLING));

      }else {

                  System.out.println("Could not find from import the
link to filelocation");
```

```
      }

                                                                  }
catch(Exception e) {

      System.out.println("XPathParseException");// in case
something went wrong in XPATH

      e.printStackTrace();
                                                                  }
                                                                  } else {

      System.out.println("Could not open file
./services/"+filetolook);
                                                                  }


      System.out.println("Found where I'm supposed to look
"+location[1]+"._"+classtoload+" and gonna load it dynamically.");

                                                                  String
currentdir = System.getProperty("user.dir");
                                                                  File
outputdir = new File(currentdir);
                                                                  URL[] urls
= new URL[]{outputdir.toURI().toURL()};

      ClassLoader cl = runlogic.class.getClassLoader();

      System.out.println("Finding class");
                                                                  if (cl !=
null) {
                                                                      //Sy
stem.out.println("Default ClassLoader : " + cl);

      URLClassLoader ucl = new URLClassLoader(urls, cl);
                                                                      File
temp=new File("./"+location[1]+"/_"+classtoload+".class");
                                                                    if
(temp.exists()){
                                                                      var1
= ucl.loadClass(location[1]+"._"+classtoload);
                                                                      }
                                                                    else{
                                                                      var1
= ucl.loadClass(location[1]+"_pkg._"+classtoload);
                                                                    }

      System.out.println(var1.getName()+ ": Class loaded
successfully");

                                                                  }
                                                                  //var1 =
```

```
Class.forName(location[1]+"._"+classtoload/*, true,
Thread.currentThread().getContextClassLoader()*/);
                                                                Object
classvariable = var1.newInstance();
                                                                //list
that holds in every odd line the name and in every even line the
class object...

      variablesList.add(attrname);

      variablesList.add(classvariable);

      variablesList.add(filetolook);


                                                          }



                                                          break; //it was
case 3 so there is no point looking for the rest.
                                                               }
                                                    }

      default:System.out.println("don't know what to do with such
variable...");
                                                    }

                                          } while
(temp.toElement(VTDNav.NEXT_SIBLING) &&
temp.matchElement("variable"));

                                          temp.toElement(VTDNav.PARENT);
                        }
                        for (int i=0; i<variablesList.size(); i+=3){
      //variablesList has in every even line the Class Object and
in every odd the name of The Class

      System.out.print(variablesList.get(i)+"\t");

      System.out.print(variablesList.get(i+1).getClass().getName()
+"\t");

      System.out.println(variablesList.get(i+2));
                                };
                                System.out.println("Variables section
ended...\n\n\n\n");
                    }

//-----------------------------------------------------------------
-----------------------------------------
                    //invoke
                    //else if
```

```
(temp.matchRawTokenString(temp.getCurrentIndex(), "invoke"))
                  else if (temp.matchElement("invoke"))
                        {
                int invoke1=6;


    //--------------   invoke without using classes, just by
opening the wsdl file and reading the bpel file     -------------
                  if (invoke1==1){
                              String
attrname=temp.toNormalizedString(temp.getAttrVal("name"));
                              String
attrpartnerLink=temp.toNormalizedString(temp.getAttrVal("partnerLin
k"));
                              String
attroperation=temp.toNormalizedString(temp.getAttrVal("operation"))
;
                              String
attrportType=temp.toNormalizedString(temp.getAttrVal("portType"));
                              String wheretolook="", templook="",
templook2="";

                              try {
                                    if (vg.parseFile(filelocation,
false)){
                                          VTDNav vn = vg.getNav();
                                          AutoPilot ap = new
AutoPilot(vn);

      ap.selectXPath("/process/partnerLinks/partnerLink");
                                          // AutoPilot moves the
cursor for you, as it returns the index value of the qualified node
                                          if (ap.evalXPath()!=-1){
      //means that it found a sequence and moves the autopilot
there
                                                do{
                                                      if
(vn.toNormalizedString(vn.getAttrVal("name")).equals(attrpartnerLin
k)){

      templook=vn.toNormalizedString(vn.getAttrVal("xmlns:impl"));
                                                            break;
                                                }
                                                }while
(vn.toElement(VTDNav.NEXT_SIBLING));
//
      System.out.println("Found it:"+templook+"\t in:
"+attrpartnerLink);
                                          }
                                          if (!templook.equals("")){

      ap.selectXPath("/process/import");
                                                if (ap.evalXPath()!=-
1){   //means that it found a sequence and moves the autopilot
```

```
there
                                                        do{
                                                            if
(vn.toNormalizedString(vn.getAttrVal("namespace")).equals(templook)
){

      templook2=vn.toNormalizedString(vn.getAttrVal("location"));

      break;
                                                            }
                                                        }while
(vn.toElement(VTDNav.NEXT_SIBLING));
//
      System.out.println("Found it:"+templook2);
                                                    }
                                                }
                                            }

                                            if
(vg.parseFile(filefolder+templook2, false)){
                                                VTDNav vn = vg.getNav();
                                                AutoPilot ap = new
AutoPilot(vn);

      ap.selectXPath("/definitions/service/port");
                                                if (ap.evalXPath()!=-1){
                                                    if
(vn.toElement(VTDNav.FC)){

      wheretolook=vn.toNormalizedString(vn.getAttrVal("location"));
                                                    }else{

      System.out.println("Can't find the (wsdlsoap:address) of wsdl
file");
                                                    }
                                                }else{

      System.out.println("Can't find the
path.../definitions/service/port");
                                                }
                                            }
                                        } catch(Exception e) {

      System.out.println("XPathParseException");// in case
something went wrong in XPATH
                                            e.printStackTrace();
                                    }

                                    //find where to look for the operation
from the process attribute
/*                                  String[] tokenize =
attrportType.split(":");
                                    if (tokenize.length==2){     //means
```

```
that is has 2 tokens, one of the namespace and one of the portType

                                    try {
                                          if
(vg.parseFile(filelocation, false)){
                                                VTDNav vn =
vg.getNav();
                                                AutoPilot ap = new
AutoPilot(vn);


ap.selectXPath("/process");//"/definitions/types/xsd:schema/xsd:imp
ort"
                                                // AutoPilot moves
the cursor for you, as it returns the index value of the qualified
node
                                                if (ap.evalXPath()!=-
1){   //means that it found a sequence and moves the autopilot
there


whereTolook=vn.toNormalizedString(vn.getAttrVal("xmlns:"+tokenize[0
]));
                                                }
                                          }
                                    } catch(Exception e) {

      System.out.println("XPathParseException");// in case
something went wrong in XPATH
                                          e.printStackTrace();
                                    }
                                    }  */
                                    int object0=15;

                                    final ExecutorService execSvc =
Executors.newFixedThreadPool(3);
                                    Callable<String> executionCallable =
new invoke(attrname, attrpartnerLink, attroperation, attrportType,
whereTolook, object0);
                                    Future<?> future =
execSvc.submit(executionCallable);
                                    future.get();

                                    //not sure if it has children...well
it may have documentation
                                  if(temp.toElement(VTDNav.FIRST_CHILD)){
//goes into children
                                          System.out.println("Going to
child(from invoke)");
                                          do {
                                                final ExecutorService
execSvc1 = Executors.newFixedThreadPool(3);
                                                Callable<String>
executionCallable1 = new runlogic(temp,filefolder, filename,
```

```
filehasnamespaces);
                                                Future<?> future1 =
execSvc1.submit(executionCallable1);
                                                future1.get();
                                          } while
(temp.toElement(VTDNav.NEXT_SIBLING));

                                          temp.toElement(VTDNav.PARENT);
//returns from children to continue with the workflow
                                }
                                System.out.println("exiting from invoke
with "+attrportType);
                        }

    //--------------   invoke using classes   -------------
                        else
                        {
                                String
inputvariable=temp.toNormalizedString(temp.getAttrVal("inputVariabl
e"));
                                String
outputvariable=temp.toNormalizedString(temp.getAttrVal("outputVaria
ble"));
                                String
attrname=temp.toNormalizedString(temp.getAttrVal("name"));
                                String
attrpartnerLink=temp.toNormalizedString(temp.getAttrVal("partnerLin
k"));
                                String
attroperation=temp.toNormalizedString(temp.getAttrVal("operation"))
;
                                String
attrportType=temp.toNormalizedString(temp.getAttrVal("portType"));
                                String attrxmlns="";
                                if (temp.getAttrVal("xmlns:impl")!=-1)

attrxmlns=temp.toNormalizedString(temp.getAttrVal("xmlns:impl"));
                                String wheretolook="", templook="",
templook2="", nameofservice="";
                                Class var1=null, var2=null, var3=null,
var4=null;
                                int k=-1,m=-1,p=0,n=0,nm=-1;
                                List<String> inputList=new
ArrayList(), outputList=new ArrayList();

                                try {
                                        for (int i=0;
i<variablesList.size(); i=i+3){
                                                if
(variablesList.get(i).equals(inputvariable)){
                                                        k=i;  //store where
it found that variable

      System.out.println("Found "+ inputvariable+ " in
```

```
variablesList in position (starting from 0) "+k);
                                                    break;
                                                }
                                            }
                                    for (int i=0; i<variablesList.size();
i=i+3){
                                                if
(variablesList.get(i).equals(outputvariable)){
                                                    m=i;  //store where
it found that variable

      System.out.println("Found "+ outputvariable+ " in
variablesList in position (starting from 0) "+m);
                                                    break;
                                                }
                                            }

                                    String classpath1[];
                                    if (attrxmlns.equals("")){
                                        String
classpath0[]=attrportType.split(":");
                                        System.out.println("There is no
xmlns:impl attribute so gonna search from the List");
                                        for (int i=0;
i<namespacesResolved.size(); i+=2){     //namespacesResolved has in
every *second* line the namespace(impl) and in every other line the
location of the namespace (e.g. "urn:fibonacci"). That is normally
located as an attribute to the "process" child
                                            if
(classpath0[0].equals(namespacesResolved.get(i))){  //xmlns:impl
from process
                                                String
nameofservice2[]=((String) namespacesResolved.get(i+1)).split(":");
      //e.g. urn:fibonacci from process

      nameofservice=nameofservice2[1];

      System.out.println(nameofservice);
                                                nm=i;
                                                break;      //found it
so it will quit for loop
                                            }
                                        }
                                    }
                                    else{
                                        String
classpath0[]=attrxmlns.split(":");
                                        nameofservice=classpath0[1];
                                    }

                                String
nameOfService1[]=attrportType.split(":");

                                    // get paths to be used for
```

```
loading
                                    /*ClassLoader base =
ClassLoader.getSystemClassLoader();
                                URL[] urls;
                                if (base instanceof URLClassLoader) {
                                 urls = new URL[]{ new
File(".").toURI().toURL() };
                                } else {
                                 urls =
((URLClassLoader)base).getURLs();
                                }

                                // list the paths actually being used
                                System.out.println("Loading from
paths:");
                                for (int i = 0; i < urls.length; i++) {
                                    System.out.println(" " + urls[i]);
                                }

                                ------------------------------//or
                                */
                                        File outputdir = new
File(System.getProperty("user.dir"));
                                        URL[] urls = new URL[]
{outputdir.toURI().toURL()};
                                        ClassLoader cl =
runlogic.class.getClassLoader();
                                        System.out.println("Finding
class");
                                        if (cl != null) {
                                              //System.out.println("Defau
lt ClassLoader : " + cl);
                                              URLClassLoader ucl = new
URLClassLoader(urls, cl);
                                              File temp=new
File("./"+nameofservice+"/"+nameOfService1[1]+"ServiceLocator.class
");
                                           if (temp.exists()){
                                              var2 =
ucl.loadClass(nameofservice+"."+nameOfService1[1]+"ServiceLocator")
;
                                              }
                                           else{
                                              var2 =
ucl.loadClass(nameofservice+"_pkg."+nameOfService1[1]+"ServiceLocat
or");
                                            }

      System.out.println(var2.getName()+ ": Class loaded
successfully");

                                        }
                                        //var1 =
Class.forName(location[1]+"._"+classtoload, true,
```

```
Thread.currentThread().getContextClassLoader());
                                    Object locateURI =
var2.newInstance();


                                    if
(vg.parseFile("./services/"+variablesList.get(k+2), false)){
            //false:do not separate namespaces. it goes into the
wsdl file to make a match
                                        //System.out.println("Got
into "+"./services/"+variablesList.get(k+2)+ " file");
                                        VTDNav vn = vg.getNav();
                                    AutoPilot ap = new
AutoPilot(vn);
                                    try {

      ap.selectXPath("/definitions");
                                        // AutoPilot moves the
cursor for you, as it returns the index value of the qualified node
                                                if (ap.evalXPath()!=-
1){   //means that it found a message and moves the autopilot there

      vn.toElement(VTDNav.FIRST_CHILD);//go to wsdl

      vn.toElement(VTDNav.FIRST_CHILD);//go to schema

      vn.toElement(VTDNav.FIRST_CHILD);//go to element
                                                        do {
                                                            if
(vn.getAttrVal("name")!=-1){ //means that there is a valid name

      String[] toseeifitmatchesInput=
(variablesList.get(k+1).getClass().getName()).split("_");

      String[] toseeifitmatchesOutput=
(variablesList.get(m+1).getClass().getName()).split("_");
                                                                int
lastofoutputlist=toseeifitmatchesOutput.length-1;//if we had a _pkg
in the name of the wsdl2java files then ther would be a problem, so
we are trying to resolve the last one anyway
                                                                int
lastofinputlist=toseeifitmatchesInput.length-1;
                                                                if
(vn.toNormalizedString(vn.getAttrVal("name")).matches(toseeifitmatc
hesInput[lastofinputlist])){

      vn.toElement(VTDNav.FIRST_CHILD);//complexType

      vn.toElement(VTDNav.FIRST_CHILD);//sequence

      vn.toElement(VTDNav.FIRST_CHILD);//element

      do{
```

```
      inputList.add(vn.toNormalizedString(vn.getAttrVal("name")));

            //System.out.println(vn.toNormalizedString(vn.getAttrVa
l("name")));


      inputList.add(vn.toNormalizedString(vn.getAttrVal("type")));

            //System.out.println(vn.toNormalizedString(vn.getAttrVa
l("type")));

            p++;  //p stores the number of the parts in a variable

      } while (vn.toElement(VTDNav.NEXT_SIBLING));

      vn.toElement(VTDNav.PARENT);//go to sequence

      vn.toElement(VTDNav.PARENT);//go to complexType

      vn.toElement(VTDNav.PARENT);//go to element
                                                                  }
                                                                  if
(vn.toNormalizedString(vn.getAttrVal("name")).matches(toseeifitmatc
hesOutput[lastofoutputlist])){

      vn.toElement(VTDNav.FIRST_CHILD);//complexType

      vn.toElement(VTDNav.FIRST_CHILD);//sequence

      vn.toElement(VTDNav.FIRST_CHILD);//element

      do{


      outputList.add(vn.toNormalizedString(vn.getAttrVal("name")));

            //System.out.println(vn.toNormalizedString(vn.getAttrVa
l("name")));


      outputList.add(vn.toNormalizedString(vn.getAttrVal("type")));

            //System.out.println(vn.toNormalizedString(vn.getAttrVa
l("type")));

            n++;  //n stores the number of the parts in a variable

      } while (vn.toElement(VTDNav.NEXT_SIBLING));
```

```
      vn.toElement(VTDNav.PARENT);//go to sequence

      vn.toElement(VTDNav.PARENT);//go to complexType

      vn.toElement(VTDNav.PARENT);//go to element
                                                                  }
                                                              }
                                                          } while
(vn.toElement(VTDNav.NEXT_SIBLING));

                                                  } else {

      System.out.println("Could not autopilot to the element");
                                                  }

                                      } catch(Exception e) {

      System.out.println("XPathParseException");// in case
something went wrong in XPATH
                                                  e.printStackTrace();
                                      }
                                      } else {
                                            System.out.println("Could
not open file ./services/"+variablesList.get(k+2));
                                      }
                                      //Trying to make the Object List
that will be the input of the invoke method
                                      Object[] inputObjectList=new
Object[inputList.size()/2];
                                      Object[] outputObjectList=new
Object[outputList.size()/2];
                                      for (int i=0;
i<inputList.size(); i=i+2){
                                            int tempi = i / 2;
                                            //System.out.println(tempi)
;
                                            String methodToCall=
inputList.get(i).substring(0, 1).toUpperCase() +
inputList.get(i).substring(1)/*.toLowerCase()*/;
                                            Method methodURI =
(variablesList.get(k+1).getClass()).getMethod("get"+methodToCall,
null); //getIn0(null) =>getMethod(methodName, param1.class,
param2.class, ..);


inputObjectList[tempi]=methodURI.invoke(variablesList.get(k+1),
null); //method.invoke (objectToInvokeOn, params)
//objectToInvokeOn is of type Object and is the object you want to
invoke the method on "method name" is the name of the method you
want to call
                                      }
                                //----method invoking#1----//
                                      //Method methodURI=
```

```
cl2.getMethod("getfibonacciAddress", java.lang.String.class );
                                Method methodURI=
var2.getMethod("get"+nameofservice+"Address", null );
//getfibonacciAddress(null)
                                String endpoint = (String)
methodURI.invoke(locateURI, null);
                                System.out.println("Calling service @
"+endpoint);

                                Service  service = new Service();
                                 Call call = (Call)
service.createCall();
                                 call.setOperationName(new
QName(endpoint, attroperation));
                                 call.setTargetEndpointAddress( new
java.net.URL(endpoint));
                                 System.out.println("Beginning to
transmit....");


      /*System.out.println(variablesList.get(k+1).getClass().getNam
e());
                                 Object tester = new Object[]
{(variablesList.get(k+1))};

      System.out.println(tester.getClass().getName());*/

                                 Object returnvar =
call.invoke(inputObjectList);

                                 //works with this one
                                 //int returnvar = (Integer)
call.invoke(new Object[]{(10)}) ;
                                 //int ret=new Integer(returnvar);
                                 System.out.println("Please wait....");
                                 System.out.println("Invoke returned="
+ returnvar);

                                 //beginning to saving to the right
variable...
                                 for (int i=0; i<outputList.size();
i=i+2){
                                              int tempi = i / 2;
                                              //System.out.println(tempi)
;
                                              String methodToCall =
outputList.get(i).substring(0, 1).toUpperCase() +
outputList.get(i).substring(1);
//outputList.get(i).substring(1).toLowerCase()
                                              Class methodtype =
xsd2type(outputList.get(i+1));
                                              System.out.println("Type of
variable to pass is of:" +methodtype.getName()+" and method's name
is set"+ methodToCall);
```

```
                                          //Thread.sleep(2000);
                                          methodURI =
(variablesList.get(m+1).getClass()).getMethod("set"+methodToCall,
methodtype); //setIn0(int) =>getMethod(methodName, param1.class,
param2.class, ..);

      methodURI.invoke(variablesList.get(m+1), returnvar);
//method.invoke (objectToInvokeOn, params) //objectToInvokeOn is of
type Object and is the object you want to invoke the method on
"method name" is the name of the method you want to call
                                          System.out.println("Invoke
ending...");
                                    }

                              k=m=-1;
                              } catch (Exception e){
                                    e.printStackTrace();
                              }

                              System.out.println("Invoke with name"+
attrname +" section ended...\n\n");
                  }

                        }
 //-------------------------------------------------------------------
-----------------------------------------
                  //assign => almost done... nedd to fix not only
copy parts but whole variables
                  else if (temp.matchElement("assign"))
                        {
                        String attrname="";
                        if (temp.getAttrVal("name")==-1) {
                              attrname=
temp.toNormalizedString(temp.getAttrVal("name"));
                        }

                        int n=0;
                              List<String> outputList=new
ArrayList();
                              List<String> inputList=new
ArrayList();
                              inputList.clear();
                              outputList.clear();

                              String fromtype="", totype="";
                              int fromnumber=0, tonumber=0;

                        String
attname=temp.toNormalizedString(temp.getAttrVal("name"));
                              Class<?> var1=null, var2=null;

                              if(temp.toElement(VTDNav.FIRST_CHILD))
{ //goes into copy child
                                    System.out.println("Going to
```

```
child(from assign)");
                                        if (temp.matchElement("copy")){

      temp.toElement(VTDNav.FIRST_CHILD);
                                                  if
(temp.matchElement("from")){


                                                  int k=-1, m=-1;
//auxiliary integers so as to determine whether or not the two
variables have been found
                                                  String
fromvariable=temp.toNormalizedString(temp.getAttrVal("variable"));

                                                  if
(temp.toElement(VTDNav.NEXT_SIBLING)){
                                                        String
tovariable=temp.toNormalizedString(temp.getAttrVal("variable"));

                                                  for (int i=0;
i<variablesList.size(); i=i+3){
                                                        if
(variablesList.get(i).equals(fromvariable)){
                                                              k=i;
      //store where it found that variable

      System.out.println("Found "+ fromvariable+ " in variablesList
in position (starting from 0) "+k);
                                                              break;
                                                        }
                                                  }
                                                  //To-Do: would it be
better to make those two for loops, checking in one pass? The if
clause, that we would use to check if both k and m have been found
in each iteration, wouldn;t that cost much more?
                                                  for (int i=0;
i<variablesList.size(); i=i+3){
                                                        if
(variablesList.get(i).equals(tovariable)){
                                                              m=i;
      //store where it found that variable

      System.out.println("Found "+ tovariable+ " in variablesList
in position (starting from 0) "+m);
                                                              break;
                                                        }
                                                  }


                                                  //----------------
we have a copy that involves whole variables
                                                  //in case there is no
parts in each variable and we want to just copy from one to another
                                                  if
(temp.getAttrVal("part")==-1 && temp.getAttrVal("variable")!=-1){
```

```
                                                        //starting to
determine what sort of <b>type</b> the "from" variable is
                                                    if
(variablesList.get(k+2).equals("noFile")){ //means that we created
it and there is no class for that thang
                                                        fromtype =
(String) variablesList.get(k+1).getClass().getName(); //now we have
what sort of type we have got
                                                        fromnumber
= 1;
                                                    }
                                                    else{
                                                        //trying
to figure out what sort of type this variable is so as to call the
right function
                                                        if
(vg.parseFile("./services/"+variablesList.get(k+2), false)){
            //false:do not separate namespaces

      VTDNav vn = vg.getNav();
                                                        AutoPilot
ap = new AutoPilot(vn);
                                                        try {

      ap.selectXPath("/definitions");
                                                                //
AutoPilot moves the cursor for you, as it returns the index value
of the qualified node

      if (ap.evalXPath()!=-1){     //means that it found a message
and moves the autopilot there

            vn.toElement(VTDNav.FIRST_CHILD);//go to wsdl

            vn.toElement(VTDNav.FIRST_CHILD);//go to schema

            vn.toElement(VTDNav.FIRST_CHILD);//go to element

            do {

                  if (vn.getAttrVal("name")!=-1){    //means that
there is a valid name

                        String[] toseeifitmatchesInput=
(variablesList.get(k+1).getClass().getName()).split("_"); //it may
be square_pkg._square so we have 3 split parts or it may be
fibonacci._fibonacciResponse so we have 2 parts. either way we want
only the last one

                        int
lastofinputlist=toseeifitmatchesInput.length-1;
```

```
                        if
(vn.toNormalizedString(vn.getAttrVal("name")).matches(toseeifitmatc
hesInput[lastofinputlist])){


      vn.toElement(VTDNav.FIRST_CHILD);//complexType


      vn.toElement(VTDNav.FIRST_CHILD);//sequence


      vn.toElement(VTDNav.FIRST_CHILD);//element

                              do {

                                    if (vn.getAttrVal("name")!=-1){


      inputList.add(vn.toNormalizedString(vn.getAttrVal("name")));

                                          //System.out.println(vn.toN
ormalizedString(vn.getAttrVal("name")));


      inputList.add(vn.toNormalizedString(vn.getAttrVal("type")));

                                          //System.out.println(vn.toN
ormalizedString(vn.getAttrVal("type")));

                                    }

                                    fromnumber++;     //fromnumber
stores the number of the parts in a variable that will need to be
copied

                              } while
(vn.toElement(VTDNav.NEXT_SIBLING) );

                              vn.toElement(VTDNav.PARENT);//go to
sequence

                              vn.toElement(VTDNav.PARENT);//go to
complexType

                              vn.toElement(VTDNav.PARENT);//go to
element

                        }

                  }
```

```
            } while (vn.toElement(VTDNav.NEXT_SIBLING));


      } else {

                  System.out.println("Could not autopilot to the
element");

      }

                                                                  }
catch(Exception e) {

      System.out.println("XPathParseException");// in case
something went wrong in XPATH

      e.printStackTrace();
                                                                  }
                                                                  } else {


      System.out.println("Could not open file
./services/"+variablesList.get(k+2));
                                                                  }

                                                            }

                                                            //starting to
determine what sort of <b>type</b> the "to" variable is
                                                            if
(variablesList.get(m+2).equals("noFile")){ //means that we created
and there is no class for that thang
                                                                  totype =
(String) variablesList.get(m+1).getClass().getName(); //now we have
what sort of type we have got
                                                                  tonumber =
1;
                                                            }
                                                            else{
                                                                  //trying
to figure out what sort of type this variable is so as to call the
right function
                                                                  if
(vg.parseFile("./services/"+variablesList.get(m+2), false)){
            //false:do not separate namespaces

      VTDNav vn = vg.getNav();
                                                                  AutoPilot
ap = new AutoPilot(vn);
                                                                  try {

      ap.selectXPath("/definitions");
                                                                        //
AutoPilot moves the cursor for you, as it returns the index value
```

```
of the qualified node

      if (ap.evalXPath()!=-1){     //means that it found a message
and moves the autopilot there

            vn.toElement(VTDNav.FIRST_CHILD);//go to wsdl

            vn.toElement(VTDNav.FIRST_CHILD);//go to schema

            vn.toElement(VTDNav.FIRST_CHILD);//go to element

            do {

                  if (vn.getAttrVal("name")!=-1){    //means that
there is a valid name

                        String[] toseeifitmatchesOutput=
(variablesList.get(m+1).getClass().getName()).split("_"); //it may
be square_pkg._square so we have 3 split parts or it may be
fibonacci._fibonacciResponse so we have 2 parts. either way we want
only the last one

                        int
lastofoutputlist=toseeifitmatchesOutput.length-1;

                        if
(vn.toNormalizedString(vn.getAttrVal("name")).matches(toseeifitmatc
hesOutput[lastofoutputlist])){


      vn.toElement(VTDNav.FIRST_CHILD);//complexType


      vn.toElement(VTDNav.FIRST_CHILD);//sequence


      vn.toElement(VTDNav.FIRST_CHILD);//element

                              do {

                                    if (vn.getAttrVal("name")!=-1){


      outputList.add(vn.toNormalizedString(vn.getAttrVal("name")));

                                          //System.out.println(vn.toN
ormalizedString(vn.getAttrVal("name")));


      outputList.add(vn.toNormalizedString(vn.getAttrVal("type")));
```

```
                                        //System.out.println(vn.toN
ormalizedString(vn.getAttrVal("type")));

                                    }

                                    tonumber++; //tonumber stores the
number of the parts in the "to" variable that will need to be saved

                                } while
(vn.toElement(VTDNav.NEXT_SIBLING) );

                                vn.toElement(VTDNav.PARENT);//go to
sequence

                                vn.toElement(VTDNav.PARENT);//go to
complexType

                                vn.toElement(VTDNav.PARENT);//go to
element

                            }

                        }

                } while (vn.toElement(VTDNav.NEXT_SIBLING));


            } else {

                    System.out.println("Could not autopilot to the
element");

            }

                                                                    }
catch(Exception e) {

            System.out.println("XPathParseException");// in case
something went wrong in XPATH

            e.printStackTrace();
                                                                    }
                                                                    } else {


            System.out.println("Could not open file
./services/"+variablesList.get(k+2));
                                                                    }

                                                                }

                                                                //now that we
have the two lists we can save each part of the "from" variable to
each part of the "to" variable
```

```
                                                        if (fromnumber!
=tonumber){

      System.out.println ("Those two variables have not the same
parts! So ignoring the assign/copy command");
                                                        }
                                                        else{
                                                              //if we
have both xsd filetypes and they are the same filetype then

      System.out.println("Assign "+ attrname + "says that those two
variables have the same amount of parts... so gonna copy them");
                                                              if
(variablesList.get(k+2).equals("noFile") &&
variablesList.get(m+2).equals("noFile") &&
variablesList.get(m+2).getClass().getName().equals(variablesList.ge
t(k+2).getClass().getName())){

      variablesList.set(m+1, variablesList.get(k+1));
                                                              }
                                                              //if just
the from variable is an xsd type
                                                              else
if(variablesList.get(k+2).equals("noFile")){ //the from variable is
just an xsd type so going right away to save it to the right
variable using the set...function. We are gusessing that there is
just one, a well put assumption

      String methodToCall2= outputList.get(0).substring(0,
1).toUpperCase() +
outputList.get(0).substring(1)/*.toLowerCase()*/;//calling the
method so as to save using the right function with the right class

      Class methodtype = xsd2type(outputList.get(1));

                                                                    if
(methodtype.equals((variablesList.get(k+1).getClass().getName()))){
//they should both be of the same Type class

      System.out.println("Type of variable to pass is of:"
+methodtype.getName()+" and method's name is set"+ methodToCall2);

      Method methodURI =
(variablesList.get(m+1).getClass()).getMethod("set"+methodToCall2,
methodtype); //setIn(5) if there is a way to say what kind of
variable the variabletocopy will be I can write
variabletocopy.getclass()

      methodURI.invoke(variablesList.get(m+1),
variablesList.get(k+1) );
                                                                    }
                                                                    else
{
```

```
      System.out.println("They are not of the same class so gonna
quit assign");
                                                                    }
                                                                }
                                                                //if just
the to variable is an xsd filetype
                                                                else
if(variablesList.get(m+2).equals("noFile")){
                                                                    //ca
lling the get function

      String methodToCall= inputList.get(0).substring(0,
1).toUpperCase() + inputList.get(0).substring(1)/*.toLowerCase()*/;

      Method methodURI =
(variablesList.get(k+1).getClass()).getMethod("get"+methodToCall,
null); //getIn0(null) =>getMethod(methodName, param1.class,
param2.class, ..);

      Object variabletocopy =
methodURI.invoke(variablesList.get(k+1), null); //method.invoke
(objectToInvokeOn, params) //objectToInvokeOn is of type Object and
is the object you want to invoke the method on "method name" is the
name of the method you want to call



                                          //params is of type Object
[] and declares the parameters to be passed to the method

      Class methodtype = xsd2type(inputList.get(1));
                                                                    if
(methodtype.equals((variablesList.get(m+1).getClass().getName()))){
//they should both be of the same Type class

      variablesList.set(m+1, variabletocopy);
                                                                    }


      else{

      System.out.println("they are not of the same class so gonna
quit the assign");
                                                                    }
                                                                }
                                                                else {
                                                                    int
iteration =inputList.size();

      for(int i=0; i<iteration; i=i+2){

      //get variable
```

```
      //Class<?>
c=Class.forName(variablesList.get(k+1).getClass().getName());//Clas
s<?> c = Class.forName("class name");

      String methodToCall= inputList.get(i).substring(0,
1).toUpperCase() + inputList.get(i).substring(1)/*.toLowerCase()*/;

      System.out.println("Method's name is get"+ methodToCall);

      Method methodURI =
(variablesList.get(k+1).getClass()).getMethod("get"+methodToCall,
null); //getIn0(null) =>getMethod(methodName, param1.class,
param2.class, ..);

      Object variabletocopy =
methodURI.invoke(variablesList.get(k+1), null); //method.invoke
(objectToInvokeOn, params) //objectToInvokeOn is of type Object and
is the object you want to invoke the method on "method name" is the
name of the method you want to call



                                                  //params is of type
Object [] and declares the parameters to be passed to the method


      String methodToCall2= outputList.get(i).substring(0,
1).toUpperCase() +
outputList.get(i).substring(1)/*.toLowerCase()*/;

      //calling the method so as to save using the right function
with the right class

      Class methodtype = xsd2type(outputList.get(i+1));

      Class methodtype2 = xsd2type(inputList.get(i+1));

      if (methodtype.equals(methodtype2)){

            System.out.println("Type of variable to pass is of:"
+methodtype2.getName()+" and method's name is set"+ methodToCall2);

            methodURI =
(variablesList.get(m+1).getClass()).getMethod("set"+methodToCall2,
methodtype2); //setIn(5) if there is a way to say what kind of
variable the variabletocopy will be I can write
variabletocopy.getclass()

            methodURI.invoke(variablesList.get(m+1), variabletocopy
);

System.out.println(variabletocopy);
```

```
      }

      else{

            System.out.println("Parts are not of the same type one
by one... To be exact "+methodtype+" of variable part
"+outputList.get(i)+"is not the same as"+methodtype2+" of variable
part "+inputList.get(i));

      }
                                                                  };
                                                              }

                                                          }

      System.out.println("Ended with assign command \n\n");
                                                          }

                                                          //-------------------
-----------------------------we have a parts copy
                                                          if
(temp.getAttrVal("part")!=-1 && temp.getAttrVal("variable")!=-1){
                                                              String
frompart=temp.toNormalizedString(temp.getAttrVal("part"));

                                                          while
(temp.toElement(VTDNav.NEXT_SIBLING)){
                                                                  if
(temp.matchElement("to")==true); //found the "to" part
                                                                  break;
                                                          }
                                                          //found to part
so gonna copy

                                                          String
topart=temp.toNormalizedString(temp.getAttrVal("part"));

      System.out.println("Gonna copy from "+fromvariable+"
variable's part "+frompart+" and give it to "+ tovariable +"
variable's part "+topart);


                                                          //now the
classes are located in k+1 and in m+1 and I'm gonna call the get
and set methods to copy the variables
                                                          if (k!=-1 && m!
=-1){
                                                                  //gonna
use their methods through reflection =>see those three lines to
understand
                                                                  //Object
obj; =>our instantiated class
                                                                  //Method
method = obj.getClass().getMethod("methodName", null);
```

```
                                                                    //method.i
nvoke(obj, null);

                                                                    //Class<?>
c=Class.forName(variablesList.get(k+1).getClass().getName());//Clas
s<?> c = Class.forName("class name");
                                                                    String
methodToCall= frompart.substring(0, 1).toUpperCase() +
frompart.substring(1)/*.toLowerCase()*/;
                                                                    Method
methodURI =
(variablesList.get(k+1).getClass()).getMethod("get"+methodToCall,
null); //getIn0(null) =>getMethod(methodName, param1.class,
param2.class, ..);
                                                                    Object
variabletocopy = methodURI.invoke(variablesList.get(k+1), null);
//method.invoke (objectToInvokeOn, params) //objectToInvokeOn is of
type Object and is the object you want to invoke the method on
"method name" is the name of the method you want to call



                                        //params is of type Object []
and declares the parameters to be passed to the method

                                                                    String
methodToCall2= topart.substring(0, 1).toUpperCase() +
topart.substring(1)/*.toLowerCase()*/;

                                                                    //trying
to figure out what sort of type this variable is so as to call the
right function
                                                                    if
(vg.parseFile("./services/"+variablesList.get(k+2), false)){
            //false:do not separate namespaces

      VTDNav vn = vg.getNav();
                                                                    AutoPilot
ap = new AutoPilot(vn);
                                                                    try {


      ap.selectXPath("/definitions");
                                                                          //
AutoPilot moves the cursor for you, as it returns the index value
of the qualified node

      if (ap.evalXPath()!=-1){     //means that it found a message
and moves the autopilot there

            vn.toElement(VTDNav.FIRST_CHILD);//go to wsdl

            vn.toElement(VTDNav.FIRST_CHILD);//go to schema

            vn.toElement(VTDNav.FIRST_CHILD);//go to element
```

```
            do {

                    if (vn.getAttrVal("name")!=-1){    //means that
there is a valid name

                            String[] toseeifitmatchesOutput=
(variablesList.get(m+1).getClass().getName()).split("_");

                            if
(vn.toNormalizedString(vn.getAttrVal("name")).matches(toseeifitmatc
hesOutput[1])){


      vn.toElement(VTDNav.FIRST_CHILD);//complexType


      vn.toElement(VTDNav.FIRST_CHILD);//sequence


      vn.toElement(VTDNav.FIRST_CHILD);//element

                                do {

                                        if
(vn.toNormalizedString(vn.getAttrVal("name")).equalsIgnoreCase(meth
odToCall2)){


      outputList.add(vn.toNormalizedString(vn.getAttrVal("name")));

                                                //System.out.println(vn.toN
ormalizedString(vn.getAttrVal("name")));


      outputList.add(vn.toNormalizedString(vn.getAttrVal("type")));

                                                //System.out.println(vn.toN
ormalizedString(vn.getAttrVal("type")));

                                        }

                                        n++;  //n stores the number of
the parts in a variable

                                } while
(vn.toElement(VTDNav.NEXT_SIBLING) );
```

```
                                vn.toElement(VTDNav.PARENT);//go to
sequence

                                vn.toElement(VTDNav.PARENT);//go to
complexType

                                vn.toElement(VTDNav.PARENT);//go to
element

                           }

                     }

               } while (vn.toElement(VTDNav.NEXT_SIBLING));


         } else {

                     System.out.println("Could not autopilot to the
element");

         }

                                                                    }
catch(Exception e) {

         System.out.println("XPathParseException");// in case
something went wrong in XPATH

         e.printStackTrace();
                                                                    }
                                                                    } else {

         System.out.println("Could not open file
./services/"+variablesList.get(k+2));
                                                                    }

                                                                    //calling
the method so as to save using the right function with the right
class
                                                                    Class
methodtype = xsd2type(outputList.get(1));

         System.out.println("Type of variable to pass is of:"
+methodtype.getName()+" and method's name is set"+ methodToCall2);

                                                                    methodURI
=
(variablesList.get(m+1).getClass()).getMethod("set"+methodToCall2,
methodtype); //setIn(5) if there is a way to say what kind of
variable the variabletocopy will be I can write
variabletocopy.getclass()
```

```
                                                                    //Object[]
oo = new Object[1];
                                                                    //oo[0] =
Integer.valueOf(15);

      methodURI.invoke(variablesList.get(m+1), variabletocopy );
//oo instead of variabletocopy

System.out.println(variabletocopy);

                                                          }

                                                    }
                                                }
                                            }
                                        }
                                        temp.toElement(VTDNav.PARENT);
//returns from children -from/to children - to continue with the
assign. Is goes to the assign...
                                        temp.toElement(VTDNav.PARENT);
//returns from copy to continue with the workflow
                                    }
                                }

 //-------------------------------------------------------------------
-----------------------------------------
                                //receive
                                else if (temp.matchElement("receive"))
                                {     //to-do: it should be done more
asynchronously. see, now, it has to check every 2 seconds to see if
something has been received in the receiveList
                                    String
attname=temp.toNormalizedString(temp.getAttrVal("name"));
                                    String attpartnerLink =
temp.toNormalizedString(temp.getAttrVal("partnerLink"));
                                    String attoperation =
temp.toNormalizedString(temp.getAttrVal("operation"));
                                    String attportType =
temp.toNormalizedString(temp.getAttrVal("portType"));
                                    String attvariable =
temp.toNormalizedString(temp.getAttrVal("variable"));
                                    String attcreateInstance =
temp.toNormalizedString(temp.getAttrVal("createInstance"));
                                    int k=-1, m=-1;

                                    if (attcreateInstance.equals("yes")){

                                        //finding the right variable
                                        for (int i=0;
i<variablesList.size(); i=i+3){ //ToDo: I should check whether it
is from the right wsdl file or not because many wsdl files may have
the same name for a different variable
                                            if
(variablesList.get(i).equals(attvariable)){
```

```
                                                k=i;  //store where
it found that variable

      System.out.println("Found "+ attvariable + " in variablesList
in position (starting from 0) "+k);
                                                break;
                                          }
                                    }

                                    //finding the right flag that
corresponds to the variable
                                    while (true){
                                          for (int i=0;
i<receiveList.size(); i=i+2){ //The receiveList has in every even
line the name of the operation and in the next one the
object/variable
                                                if
(receiveList.get(i).equals(attoperation)){
                                                      m=i;  //store
where it found that variable

      System.out.println("Found operation"+ attoperation + " in
receiveList in position (starting from 0) "+m);
                                                      break;
                                                }
                                          }
                                          Thread.sleep(2000);
      //allow the cpu to rest for a while until something has been
sent.
                                          if (m!=-1) {
                                                break; //if you have
found that there is something in the List that has been rece3ived,
stop searching the List and exit the while loop
                                          }
                                    }

                                    variablesList.set((k+1),
receiveList.get(m+1));
                              }

                        }

//-------------------------------------------------------------------
-----------------------------------------
                        //documentation
                        else if (temp.matchElement("documentation"))

                        {
                                    final ExecutorService execSvc =
Executors.newFixedThreadPool(3);
                                    Callable<String>
executionCallable = new
documentation(temp.toNormalizedString(temp.getText()));
                                    Future<?> future =
```

```
execSvc.submit(executionCallable);
                                    future.get();
                        }

//-------------------------------------------------------------------
-----------------------------------------
                  //wait
                        else if (temp.matchElement("wait"))
                        {
                              long milliseconds=0L, prep2=0L;
                              String prep="0";
                              if
(temp.toElement(VTDNav.FIRST_CHILD)){

                                    if (temp.matchElement("for")){
                                          String
forattr=temp.toNormalizedString(temp.getText());
                                          String
forattr2[]=forattr.split("T");
                                          if
(forattr2[0].startsWith("'P")){
                                                char
[]arr=forattr2[0].toCharArray();
                                                char
[]arr1=forattr2[1].toCharArray();
                                                for (int i=0;
i<arr.length; i++)
                                                      {
                                                            if
(arr[i]=='\'') continue;
                                                            else if
(arr[i]=='P') continue;
                                                            else if
(arr[i]>='0' && (arr[i]<='9'))
                                                            {

      String prep3=Character.toString(arr[i]);

      prep=prep + prep3;
                                                            }
                                                            else if
(arr[i]=='Y') {

      prep2=new Long(prep);

      milliseconds=milliseconds+(prep2*31556926000L);
//31556926000L is the number of milliseconds in a gregorian year

      prep="0";
                                                            }
                                                            else if
(arr[i]=='M') {

      prep2=new Long(prep);
```

```
      milliseconds=milliseconds+(prep2*2629743830L); //2629743830LL
is the number of milliseconds in a month

      prep="0";
                                                                  }
                                                                  else if
(arr[i]=='D') {

      prep2=new Long(prep);

      milliseconds=milliseconds+(prep2*86400000L); //86400000L is
the number of milliseconds in a day

      prep="0";
                                                                  }

                                                              }//end of for
loop
                                                        for (int i=0;
i<arr1.length; i++)
                                                              {
                                                                  if
(arr1[i]=='\'') continue;
                                                                  else if
(arr1[i]=='T') continue;
                                                                  else if
(arr1[i]>='0' && (arr1[i]<='9'))
                                                                  {
                                                                  String
prep3=Character.toString(arr1[i]);
                                                                  prep=prep
+ prep3;
                                                                  }
                                                                  else if
(arr1[i]=='H'){

      prep2=new Long(prep);

      milliseconds=milliseconds+(prep2*3600000L); //31556926000L is
the number of milliseconds in a year

      prep="0";
                                                                  }
                                                                  else if
(arr1[i]=='M'){

      prep2=new Long(prep);

      milliseconds=milliseconds+(prep2*60000L); //31556926000L is
the number of milliseconds in a year

      prep="0";
```

```
                                                                    }
                                                                    else if
(arr1[i]=='S'){

      prep2=new Long(prep);

      milliseconds=milliseconds+(prep2*1000L); //31556926000L is
the number of milliseconds in a year

      prep="0";
                                                                    }
                                                                }//end of for
loop
                                                }
                                                System.out.println("\nGonna
wait for "+milliseconds);
                                                Thread.sleep(milliseconds);
                                        }

                                        else if
(temp.matchElement("until")){
                                                String
forattr=temp.toNormalizedString(temp.getText());
                                                if (!forattr.equals("''")){
                                                      String
forattr2[]=forattr.split("'");
                                                      String
forattr3[]=forattr2[1].split("T");
                                                      long currenttime =
GregorianCalendar.getInstance().getTimeInMillis();
                                                      Calendar
datetostopwaiting= new GregorianCalendar();
                                                      for (int i=0;
i<forattr3.length; i++)

      System.out.println(forattr3[i]);

                                                      String
[]arr=forattr3[0].split("-");
                                                      String
[]arr1=forattr3[1].split(":");


      datetostopwaiting.set(Integer.parseInt(arr[0]),
Integer.parseInt(arr[1])-1, Integer.parseInt(arr[2]),
Integer.parseInt(arr1[0]), Integer.parseInt(arr1[1]),
Integer.parseInt(arr1[2]));
                                                      long
timetostop=datetostopwaiting.getTimeInMillis();

      milliseconds=timetostop-currenttime;
```

```
      System.out.println("\nGonna wait for "+milliseconds);

      Thread.sleep(milliseconds);
                                          }
                                    }

                              temp.toElement(VTDNav.PARENT);
//returns from children to continue with the workflow
                              }
                        }


//-------------------------------------------------------------------
-----------------------------------------
                        //flow
                        else if (temp.matchElement("flow")){
                              System.out.println("Going to children
(from flow with name
"+temp.toNormalizedString(temp.getAttrVal("name"))+")");
                              temp.toElement(VTDNav.FIRST_CHILD);
//goes into children

                              ThreadPoolExecutor threadpool = new
ThreadPoolExecutor(5,5, 50000L, TimeUnit.MILLISECONDS,new
LinkedBlockingQueue<Runnable>(), new PoolHandlerFactory());
                              final ExecutorService execSvc =
Executors.newFixedThreadPool(10);
                              ArrayList<FutureTask> t = new
ArrayList<FutureTask>();
                              //FutureTask t[] = new FutureTask[40];

                              //so, for every child of the flow
process I will give it an FutureTask so as to perform
similtaneously
                              do {
                                    //System.out.println(temp.toNorm
alizedString(temp.getCurrentIndex()));
                                    t.add(new FutureTask<String>
(new runlogic(temp.getCurrentIndex(), filefolder, filename,
filehasnamespaces)));
                              } while
(temp.toElement(VTDNav.NEXT_SIBLING));

                              //starting to parallelize the threads
                              for (int i = 0; i < t.size(); i++) {
                                  threadpool.execute(t.get(i));
                              }

                              for (int i = 0; i < t.size(); i++) {
                                    try{
                                          System.out.println("Trying
to wait for them to stop");
                                          t.get(i).get();
                                    }catch (InterruptedException ee)
```

```
{} catch (ExecutionException e) {
                                        // TODO Auto-generated
catch block
                                        e.printStackTrace();
                                  }
                            }

                            temp.toElement(VTDNav.PARENT);
//returns from children to continue with the workflow

                            }

//-------------------------------------------------------------------
----------------------------------------
                      //sequence
                      else if (temp.matchElement("sequence"))
                      {     //logic same as invoke regarding
futuretasks
                            if (temp.getAttrCount()>0)
                            System.out.println("Going to children
(from sequence with name
"+temp.toNormalizedString(temp.getAttrVal("name"))+")");
                            temp.toElement(VTDNav.FIRST_CHILD);
//goes into children
                            do {
                                  final ExecutorService execSvc1 =
Executors.newFixedThreadPool(1);
                                  Callable<String>
executionCallable1 = new runlogic(temp,filefolder, filename,
filehasnamespaces);
                                  Future<?> future1 =
execSvc1.submit(executionCallable1);
                                  future1.get();
                            } while
(temp.toElement(VTDNav.NEXT_SIBLING));
                            temp.toElement(VTDNav.PARENT);
//returns from children to continue with the workflow
                      }



                } catch (NavException e) {
                      // TODO Auto-generated catch block
                      e.printStackTrace();
                } catch (InterruptedException e) {
                      // TODO Auto-generated catch block
                      e.printStackTrace();
                } catch (ExecutionException e) {
                      // TODO Auto-generated catch block
                      e.printStackTrace();
                } catch (IllegalArgumentException e) {
                      // TODO Auto-generated catch block
                      e.printStackTrace();
```

```
                } catch (InvocationTargetException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (SecurityException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (NoSuchMethodException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }




    return "ok";
    }

            public Class xsd2type(String type){
                if (type.equalsIgnoreCase("xsd:int")){
                        return Integer.TYPE;
                }
                else if (type.equalsIgnoreCase("xsd:long")){
                        return Long.TYPE;//long typeclass;
                }
                else if (type.equalsIgnoreCase("xsd:short")){
                        return Short.TYPE;//short typeclass;
                }
                else if (type.equalsIgnoreCase("xsd:float")){
                        return Float.TYPE;//float typeclass;
                }
                else if (type.equalsIgnoreCase("xsd:double")){
                        return Double.TYPE;//double typeclass;
                }
                else if (type.equalsIgnoreCase("xsd:boolean")){
                        return Boolean.TYPE;//boolean typeclass;
                }
                else if (type.equalsIgnoreCase("xsd:byte")){
                        return Byte.TYPE;//byte typeclass;
                }
                else if
(type.equalsIgnoreCase("xsd:base64Binary")){
                        byte[] typeclass = null;
                        return typeclass.getClass();//byte[]
typeclass;
                }
                else if (type.equalsIgnoreCase("xsd:string")){
                        return String.class;//java.lang.String
typeclass;
                }
                else if (type.equalsIgnoreCase("xsd:integer")){
                        java.math.BigInteger typeclass = null;
                        return
typeclass.getClass();//java.math.BigInteger typeclass;
                }
```

```
                else if (type.equalsIgnoreCase("xsd:decimal")){
                      java.math.BigDecimal typeclass = null;
                      return
typeclass.getClass();//java.math.BigDecimal typeclass;
                }
                else if (type.equalsIgnoreCase("xsd:dateTime")){
                      java.util.Calendar typeclass = null;
                      return
typeclass.getClass();//java.util.Calendar typeclass;
                }
                else if (type.equalsIgnoreCase("xsd:date")){
                      java.util.Date typeclass = null;
                      return typeclass.getClass();//java.util.Date
typeclass;
                }
                else if (type.equalsIgnoreCase("xsd:boolean
xsi:nillable=true")){
                      java.lang.Boolean typeclass = null;
                      return
typeclass.getClass();//java.lang.Boolean typeclass;
                }
                else if (type.equalsIgnoreCase("xsd:float
xsi:nillable=true")){
                      java.lang.Float typeclass = null;
                      return
typeclass.getClass();//java.lang.Float typeclass;
                }
                else if (type.equalsIgnoreCase("xsd:double
xsi:nillable=true")){
                      java.lang.Double typeclass = null;
                      return
typeclass.getClass();//java.lang.Double typeclass;
                }
                else if (type.equalsIgnoreCase("xsd:int
xsi:nillable=true")){
                      java.lang.Integer typeclass = null;
                      return
typeclass.getClass();//java.lang.Integer typeclass;
                }
                else if (type.equalsIgnoreCase("xsd:short
xsi:nillable=true")){
                      java.lang.Short typeclass = null;
                      return
typeclass.getClass();//java.lang.Short typeclass;
                }
                else if (type.equalsIgnoreCase("xsd:byte
xsi:nillable=true")){
                      java.lang.Byte typeclass = null;
                      return typeclass.getClass();//java.lang.Byte
typeclass;
                }
                else if (type.equalsIgnoreCase("xsd:anyType")){
                      java.lang.Object typeclass = null;
                      return
```

```
typeclass.getClass();//java.lang.Object typeclass;
                    }
                    return null;
              }

}

class PoolHandlerFactory implements ThreadFactory {
      public Thread newThread(Runnable r) {
            Thread t = new Thread(r);
            t.setDaemon(true);
            return t;
      }
}
```

Για να τρέξει αυτός ο κώδικας χρειάζεται και ένα πακέτο μέσα στο βασικό που να ονομάζεται wsdl2java και να έχει μέσα τα εξής 2 αρχεία:

createwsdl2javafiles.java

```
package wsdl2java;


import java.io.File;
import java.net.URL;
import java.net.URLClassLoader;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.concurrent.Callable;

import org.apache.axis.wsdl.toJava.Emitter;


public class createwsdl2javafiles implements Callable<String>{
       // Instantiate the emitter
      Emitter emitter = new Emitter();

      String wsdl_location;

    public createwsdl2javafiles (String wsdllocation){
      //reading the wsdl(and optionally the xsd---it does create the
SimpleProcess file) file and converting it to Java classes
            wsdl_location = wsdllocation;
            //String wsdl_location1 []= new String [] {wsdl_location};
            //to be done in args
      }

    public String call(){

      // set any properties you want, such as:
      //Emitter from Axis2
```

```
      //Emitter from axis 1
      emitter.setVerbose(true);
      emitter.setImports(true);
      emitter.setAllWanted(true);//critical for making the types as
java files
      emitter.setSkeletonWanted(true);
      emitter.setServerSide(true); //so as to make deploy and
undeploy.wsdd as well as bindingImpl*/
      //emitter.setTestCaseWanted(true);
      //emitter.setHelperWanted(true);
      //emitter.setTimeout(3600);
            //emitter.setPackageName(projectname);//so as to make it
one package where any class can have access to anything


            // Run the emitter
            try {
                  emitter.run(wsdl_location);   //makes the java files
for the specified service out of the wsdl file
                  List<?> list2 = emitter.getGeneratedFileNames();
//with / in file names so as to compile them
                  List<?> list =
emitter.getGeneratedClassNames();//with dots in case there is a chance
to import
                  List<String> ClassesCorrespondence = new
ArrayList();

                  Iterator<?> itr2=list2.iterator();

                  //start compiling the java files (created from the
wsdl file -by the emitter) to classes
                  for (Iterator<?> itr=list.iterator();
/*itr2.hasNext() &&*/ itr.hasNext(); )
                  {
                        Object p = itr.next();  //objects in a form
type to load classes
                        if (p==null) continue;  //so it won't compile
deploy and the rest of the files that are not .java ...
                              String[] st= ((String) p).split("\\.");
      //tokenize the class so as to get the last token which has the
name of the service

      ClassesCorrespondence.add(st[(st.length)-1]);   //fill the list
so as to have both the name of the class as well as the name of the
class in dots...
                              ClassesCorrespondence.add((String)p);

                        Object o = itr2.next(); //objects in a form
type to compile

                        //why do it with futures??? Fork would be good
here...
                        CreateClassesFromJavaFiles.call((String) o);
                  }
```

```
            } catch (Exception e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
            return "0";
      }
}
```

Και

CreateClassesFromJavaFiles.java

```
package wsdl2java;
import com.sun.tools.javac.Main;

public class CreateClassesFromJavaFiles /*implements
Callable<String>*/ {
      String filetocompile;

      public CreateClassesFromJavaFiles (String filetocompile){
            this.filetocompile=filetocompile;
      }

      public static String call(String filetocompile) {
                        //creating the classes programmatically


      String[] options=new String[] {"-source", "1.4", "-d", ".", "-
classpath", "./extras/axis.jar;./extras/jaxrpc.jar;" , "-sourcepath",
".", filetocompile };
//          javac.compile(options);
      Main.compile(options);


            return filetocompile;

      }
}
```

Τέλος, με σκοπό να γίνει μία πιο Modular εκδοχή της μηχανής για όσους το απαιτήσουν, παρατίθενται και δύο αρχεία μέσα σε ένα package bpelactionscontrol, τα invoke.java και documentation.java, τα οποία επί τούτου είναι απαραίτητα για να μεταγλωττιστεί η εφαρμογή μας. Σημειώνεται ότι η υλοποίηση του Invoke δεν είναι αυτή εδώ, αλλά αυτή που φαίνεται στον κεντρικό κώδικα

invoke.java

```
package bpelactionscontrol;
import java.util.concurrent.Callable;
```

```
import javax.xml.namespace.QName;
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import java.lang.reflect.*;

public class invoke implements Callable<String> {
      String name_receive, partnerLink_receive, operation_receive,
                portType_receive, wheretolook;
      Object parameter;

      public invoke(String name_receive, String partnerLink_receive,
String operation_receive, String portType_receive, String wheretolook,
Object parameter) {
            this.name_receive = name_receive;
            this.partnerLink_receive = partnerLink_receive;
            this.operation_receive = operation_receive;
            this.portType_receive = portType_receive;
            this.wheretolook =wheretolook;
            this.parameter=parameter;
            }

      public String call() {
            System.out.println( "Invoke with attributes, name:
"//"just inner receive. Element name ==> " this.getId()+
                        +name_receive+"\tpartnerLink:
"+partnerLink_receive+ "\toperation:
"+operation_receive+"\n\tportType: "
                        +portType_receive+"\nGonna look
in:"+wheretolook);

            try {

                  //firstArg=args[0];
//                secondArg=args[1];

                  String endpoint = wheretolook;
                  Service  service = new Service();
                  Call call = (Call) service.createCall();
                  call.setOperationName(new QName(endpoint,
operation_receive));
                  call.setTargetEndpointAddress( new
java.net.URL(endpoint) );
                  System.out.println("Beginning to transmit....");

                  int returnvar = (Integer) call.invoke(new Object[]
{(parameter)}) ;
                  int ret=new Integer(returnvar);
                  System.out.println("Please wait....");
                  System.out.println("Fibonacci(15))="  + returnvar);


               } catch (Exception e) {
                     System.err.println("Execution failed. Exception:
" + e);
                     e.printStackTrace();
               }
```

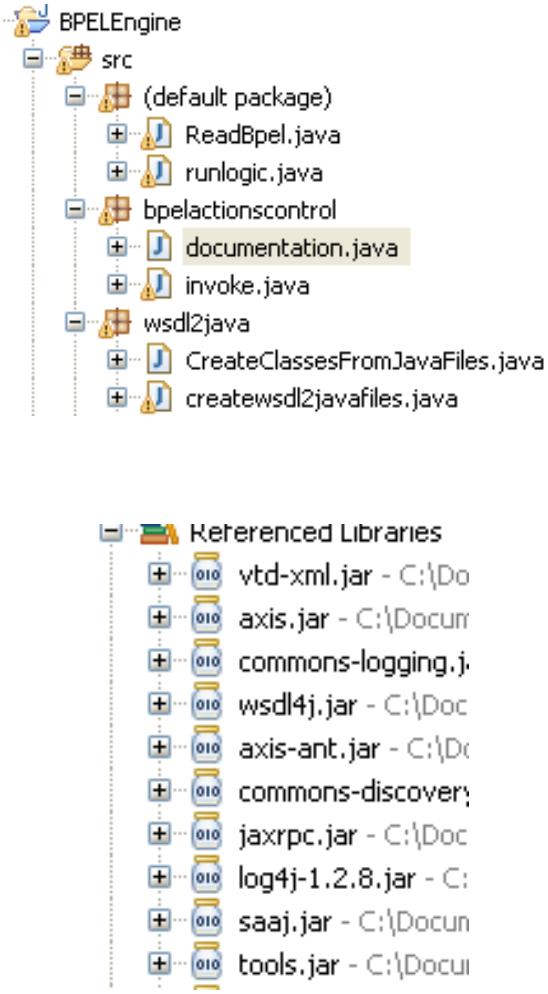
```
            return "ok";
      }
}
```

Και

documentation.java

```
package bpelactionscontrol;
import java.util.concurrent.Callable;

public class documentation implements Callable<String> {
      String text;

      public documentation(String text){
            this.text=text;
            }

      public String call() {
            try {
                  //two different types for retrieving the same thing
                  //System.out.println("Documentation. Element name
==> " +vn.toString(vn.getCurrentIndex()) );
                  System.out.println("Documentation says ==> " +text);
      } catch (Exception e) {
                  e.printStackTrace();
            }
      return "ok";
    }

      public void run() {



      }
}
```

Στον eclipse το project μας θα πρέπει να έχει την δομή που φαίνεται στο σχήμα:


BPELEngine
src
(default package)
ReadBpel.java
runlogic.java
bpelactionscontrol
documentation.java
invoke.java
wsdl2java
CreateClassesFromJavaFiles.java
createwsdl2javafiles.java
Referenced Libraries
vtd-xml.jar - C:\Do
axis.jar - C:\Docum
commons-logging.j
wsdl4j.jar - C:\Doc
axis-ant.jar - C:\D
commons-discovery
jaxrpc.jar - C:\Doc
log4j-1.2.8.jar - C:
saaj.jar - C:\Docun
tools.jar - C:\Docu

ΒΙΒΛΙΟΓΡΑΦΙΑ

- James McGovern, Sameer Tyagi, Michael Stevens and Sunil Matthew: Java Web Services Architecture, Morgan Kaufmann, 2003
- Ethan Cerami: Web Services Essentials, O'Reilly, 2002
- Ramesh Nagappan, Robert Skoczylas, Rima Patel Sriganesh: Developing Java Web Services, Wiley, 2003
- Eric Armstrong, Stephanie Bodoff, Debbie Carson, Maydene Fisher, Scott Fordin, Dale Green, Kim Haase, Eric Jendrock: The Java Web Services Tutorial, Addison Wesley, 2003
- Henry Bequet, Meeraj Moidoo Kunnumpurath, Sean Rhody, Andre Tost: Beginning Java Web Services, Wrox Press Ltd, 2002
- I. Foster. "What is the Grid? A Three Point Checklist". GRIDToday, July 20, 2002.
- I. Foster, C. Kesselman, and S. Tuecke. "The Anatomy of the Grid: Enabling Scalable Virtual Organizations".International J. Supercomputer Applications15(3), 2001
- ISO 12052:2006, http://www.iso.org/
- ISO/TR 16044:2004, http://www.iso.org/
- http://www.workflowpatterns.com/documentation/documents/BPM-06-22.pdf
- ftp://ftp.informatik.uni-stuttgart.de/pub/library/medoc.ustuttgart_fi/STUD-2052/STUD-2052.pdf

Ιστότοποι

1. Business Process with BPEL4WS: Understanding BPEL4WS, Part 1, http://www-106.ibm.com/developerworks/webservices/library/ws-bpelcol1/
2. Business Process with BPEL4WS: Learning BPEL4WS, Part 2, http://www-106.ibm.com/developerworks/webservices/library/ws-bpelcol2/
3. Business Process with BPEL4WS: Learning BPEL4WS, Part 3, http://www-106.ibm.com/developerworks/webservices/library/ws-bpelcol3/
4. Business Process with BPEL4WS: Learning BPEL4WS, Part 5, http://www-106.ibm.com/developerworks/webservices/library/ws-bpelcol5/
5. Business Process with BPEL4WS: Learning BPEL4WS, Part 7, http://www-106.ibm.com/developerworks/webservices/library/ws-bpelcol7/
6. Business processes in a Web services world, http://www-128.ibm.com/developerworks/webservices/library/ws-bpelwp/
7. Web Services Business Process Execution Language - Working Draft 01, 08 September 2004, http://www.oasis-open.org/apps/org/workgroup/wsbpel/
8. http://www.w3.org/TR/ws-gloss/
9. http://www.w3.org/TR/wsdl20/#intro
10. http://www2.sys-con.com/itsg/virtualcd/webservices/archives/0307/peltz/index.html
11. MEgha Workflow Management System, http://www3.interscience.wiley.com/journal/122522172/abstract
12. http://www.gridworkflow.org/snips/gridworkflow/space/Grid+Workflow