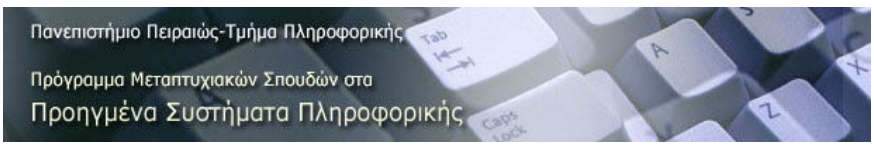
Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	«Ανάπτυξη Μηχανής Εκτέλεσης Διαδικασιών WS-BPEL» «Development of a WS-BPEL engine»
Ονοματεπώνυμο Φοιτητή	Θεόδωρος Τρίμης Κόρος
Αριθμός Μητρώου	ΜΠΣΠ/10050
Κατεύθυνση	Συστήματα Υποστήριξης Αποφάσεων
Επιβλέπων	Δημήτρης Αποστόλου, Επ. Καθηγητής



Ημερομηνία Παράδοσης Απρίλιος 2014

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Δημ. Αποστόλου Επ. Καθηγητής	Μιχάλης Ψαράκης Επ. Καθηγητής	Παν. Κοντζανικολάου Λέκτωρ

Περιεχόμενα

Κατάλογος Εικόνων

Περίληψη

Η παρούσα εργασία ασχολείται με την αυτοματοποίηση επιχειρησιακών διαδικασιών στο πλαίσιο της αρχιτεκτονικής που βασίζεται σε υπηρεσίες. Οι επιχειρησιακές διαδικασίες εξωτερικεύονται ως υπηρεσίες, που μπορούν να καταναλωθούν από συνεργαζόμενες διαδικασίες, σε ετερογενή περιβάλλοντα εντός ενός οργανισμού ή μεταξύ περισσότερων οργανισμών, χωρίς να απαιτείται η γνώση των λεπτομερειών της υλοποίησης των διαδικασιών. Με αυτό τον τρόπο επιτυγχάνεται η χαλαρή σύζευξη μεταξύ των διαδικασιών που επιτρέπει την απεξάρτηση από τις λεπτομέρειες υλοποίησης των διαδικασιών, είτε αυτέ αφορούν πλατφόρμες λογισμικού και υλικού, είτε υποδομές είτε τα συστατικά μέρη των διαδικασιών κάθε αυτά. Τέλος παρουσιάζεται και αναλύεται η υλοποίηση μίας μηχανή εκτέλεσης επιχειρησιακών διαδικασιών που μοντελοποιούνται με το πρότυπο WS-BPEL.

Λέξεις κλειδιά:

Business Process Execution Language (BPEL), WS-BPEL, Service Oriented Architecture (SOA), Web Services, Web Services Description Language (WSDL), SOAP, UDDI, PHP.

Abstract

This thesis is concerned with the automation of business processes in the framework of the service oriented architecture. Business processes are externalized as services that can be consumed by partner processes, in heterogynous environments, inside a single organization or between many organizations, without the need of detailed knowledge of how the processes are actually implemented. In this way a loose coupling is possible, which decouples the use of the business processes from the details of the implementation, software or hardware platforms and infrastructure or the components that comprise the business processes. Also the development of a software machine that runs business process model in WS-BPEL is presented.

Keywords:

Business Process Execution Language (BPEL), WS-BPEL, Service Oriented Architecture (SOA), Web Services, Web Services Description Language (WSDL), SOAP, UDDI, PHP.

Εισαγωγή

Οι επιχειρησιακές εφαρμογές και τα πληροφοριακά συστήματα αποτελούν σημαντικά εργαλεία για τις επιχειρήσεις, οι οποίες βασίζονται σε αυτά προκειμένου να φέρουν σε πέρας τις επιχειρησιακές λειτουργίες. Τα πληροφοριακά συστήματα μίας επιχείρησης μπορούν να συνεισφέρουν στην αύξηση της αποδοτικότητα της αυτοματοποιώντας τις επιχειρησιακές διαδικασίες. Τα πληροφοριακά συστήματα και οι επιχειρησιακές εφαρμογές θα πρέπει να είναι σε θέση να υποστηρίζουν αποτελεσματικά το σύνολο των επιχειρησιακών διαδικασιών και για να το πετύχουν αυτό θα πρέπει να είναι “ευθυγραμμισμένα” με αυτές. Οι απαιτήσεις για συχνές αλλαγές των επιχειρησιακών διαδικασιών καθιστούν αυτό το στόχο δύσκολο. Η δυσκολίες στη μεταφορά των αλλαγών που παρουσιάζονται στις επιχειρησιακές διαδικασίες στις εφαρμογές οδηγεί συχνά σε ένα χρονικά χάσμα στην απόκριση των δεύτερων στις αλλαγές. Το μέγεθος και η πολυπλοκότητα των αλλαγών, η κατάσταση των εφαρμογών από πλευράς συντήρησης τους, η σαφώς προδιαγεγραμμένη αρχιτεκτονική τους και η δυνατότητες επεκτασιμότητας αποτελούν βασικούς παράγοντες για τη γρήγορη, επιτυχή και αποτελεσματική υλοποίηση των αλλαγών. Επιπλέον οι αλλαγές τείνουν να καθιστούν τις εφαρμογές, ειδικά αυτές που έχουν προβληματικές ή πεπαλαιωμένες αρχιτεκτονικές ή έχουν υποστεί πολλαπλές τροποποιήσεις, λιγότερο ανθεκτικές (robust). Επίσης πολλές επιχειρήσεις στηρίζονται σε παλαιές εφαρμογές, εφαρμογές που υποστηρίζουν αποσπασματικά και όχι ολοκληρωμένα τις επιχειρησιακές διαδικασίες. Όλα τα παραπάνω καθιστούν το στόχο της ευθυγράμμισης των εφαρμογών και των πληροφοριακών συστημάτων με τις επιχειρησιακές διαδικασίες δύσκολο.

Για να είναι σε θέση το πληροφοριακό “οικοσύστημα” μίας επιχείρησης να υποστηρίζει αποτελεσματικά την αυτοματοποίηση των επιχειρησιακών διαδικασιών, που όπως αναφέρθηκε προϋποθέτει την ευθυγράμμιση των εφαρμογών με τις διαδικασίες, σύμφωνα με τις σύγχρονες θεωρήσεις, απαιτείται να:

- Παρέχει τυποποιημένες μεθόδους εξωτερίκευσης (παροχής) και πρόσβασης της λειτουργικότητας των εφαρμογών ως υπηρεσίες
- Παρέχει ολοκληρωμένη αρχιτεκτονική μεταξύ των διαφόρων υπηρεσιών, των υπαρχόντων και τυχόν νέων εφαρμογών
- Παρέχει μία εξειδικευμένη γλώσσα για τη σύνθεση της λειτουργικότητας που παρέχουν οι εφαρμογές σε επιχειρησιακές διαδικασίες
- Παρέχει την κατάλληλη υποδομή για την επικοινωνία και διαχείριση των υπηρεσιών, των μηνυμάτων που αυτές ανταλλάσουν, της δρομολόγησης, μετατροπής δεδομένων

Οι Web Services, μονάδες δηλαδή λογισμικού που χαρακτηρίζονται από χαλαρή ζεύξη, και οι οποίες ενθυλακώνουν τη λογική που υλοποιούν εντός διακριτών πλαισίων και μπορούν εύκολα να συνδυαστούν σε σύνθετες λύσεις εμφανίζονται σήμερα ως η πλέον κατάλληλη τεχνολογία για την παροχή της λειτουργικότητας των εφαρμογών ως υπηρεσίες και παρέχουν δυνατότητες σύγχρονης και ασύγχρονης επικοινωνίας.

Η αρχιτεκτονική που βασίζεται σε υπηρεσίες (Service-Oriented Architecture, SOA), έχει υιοθετηθεί σήμερα από πολλές επιχειρήσεις ως μία αποτελεσματική μέθοδος για την ολοκλήρωση επιχειρησιακών εφαρμογών που υλοποιούνται με Web Services.

Συνήθης σήμερα αρχιτεκτονική ανάπτυξης σύνθετων εφαρμογών που βασίζεται σε υπηρεσιο-κεντρικές (service-oriented) αρχές είναι η συνδυασμένη χρήση των τεχνολογιών SOA, Web Services και WS-BPEL.

Τέλος, η τεχνολογία Enterprise Service Bus (ESB) παρέχει τη δυνατότητα διαλειτουργικότητας μεταξύ υπαρχόντων εφαρμογών και Web Services, υποστηρίζοντας την κεντρική, καλά συντονισμένη διαχείριση των υπηρεσιών και της επικοινωνίας μεταξύ τους.

Σκοπός της εργασίας

Σκοπός της εργασίας είναι η μελέτη της αρχιτεκτονικής που βασίζεται σε υπηρεσίες (SOA) και η εμβάθυνση στο πρότυπο WS-BPEL. Προκειμένου να ο μεγαλύτερος δυνατός βαθμός κατανόησης επιλέχτηκε να μελετηθεί κατά πόσο είναι δυνατό να αναπτυχθεί μία μηχανή εκτέλεσης BPEL. Και αυτό κυρίως με δεδομένο ότι οι υπάρχουσες υλοποιήσεις μηχανών εκτέλεσης BPEL είναι ολοκληρωμένα πακέτα, όπως για παράδειγμα οι λύσεις της εταιρείας

Oracle, που προσφέρουν εξαιρετικά μεγάλο αριθμό δυνατοτήτων, σε γραφικό περιβάλλον και δεν θα παρείχαν δυνατότητες εμβάθυνσης.

Ως γλώσσα υλοποίησης επιλέχτηκε η PHP, για τρείς λόγους:

- Οι υπάρχουσες υλοποιήσεις βασίζονται σχεδόν αποκλειστικά σε Java
- Η PHP αποτελεί ενδεχομένως την πλέον διαδεδομένη γλώσσα server-side scripting
- Ο γράφων είχε ικανοποιητική γνώση της συγκεκριμένης γλώσσας
- η συγκεκριμένη υλοποίηση θα δώσει την ευκαιρία να αντιμετωπιστούν προβλήματα, τα οποία η PHP πιθανόν δεν είναι σχεδιασμένη ή δεν συνηθίζεται να αντιμετωπίζει

Δομή της εργασίας

Η εργασία αποτελείται από τέσσερα μέρη:

Στο πρώτο μέρος παρουσιάζονται οι υπηρεσίες και το αρχιτεκτονική που βασίζεται σ αυτές.

Στο δεύτερο μέρος γίνεται μία σχετικά αναλυτική παρουσίαση του προτύπου WS-BPEL 2.0, το οποίο θα υποστηρίζεται από τη συγκεκριμένη μηχανή που θα υλοποιηθεί. Στο μέρος αυτό υπάρχει και παρουσίαση των WSDL και SOAP, που αποτελούν τεχνολογίες άμεσα σχετικές με το πρότυπο.

Στο τρίτο μέρος αποτελεί το σημαντικότερο τμήμα της εργασίας, δηλαδή η υλοποίηση. Εδώ παρουσιάζεται η μηχανή εκτέλεσης BPEL σε PHP και ένα συγκεκριμένο παράδειγμα εφαρμογής.

Τέλος στο τέταρτο μέρος παρουσιάζονται τα συμπεράσματα που προέκυψαν.

Περιορισμοί της εργασίας

Σκοπός της εργασίας είναι η εμβάθυνση στο πρότυπο WS-BPEL μέσα από την υλοποίηση μίας μηχανής εκτέλεσης BPEL διαδικασιών σε PHP. Στο πλαίσιο της μεταπτυχιακής διατριβής δεν είναι εφικτό να υλοποιηθεί μία ολοκληρωμένη λύση που θα παρέχει πλήρη λειτουργικότητα. Κύρια επιδίωξη είναι να αναζητηθούν αρχιτεκτονικές υλοποίησης που θα παρέχουν βασική λειτουργικότητα, θα υποστηρίζονταν βασικές δραστηριότητες και ενδεχομένως κάποια από τα προηγμένα χαρακτηριστικά, όπως οι μακροχρόνιες διαδικασίες, και να μελετηθεί αν θα ήταν εφικτό σε μία ολοκληρωμένη υλοποίηση να παρέχουν πλήρη λειτουργικότητα, υποστηρίζοντας ολοκληρωμένα το πρότυπο.

1. Υπηρεσίες και Αρχιτεκτονική βασισμένη στις Υπηρεσίες

1.1 Τι είναι υπηρεσίες

Από την άποψη της υλοποίησης επιχειρησιακών διαδικασιών με πληροφοριακά συστήματα οι υπηρεσίες μπορούν να ορισθούν ως μία συλλογή λειτουργιών, η επικοινωνία με τις οποίες πραγματοποιείται με συγκεκριμένο τρόπο και μπορούν να κληθούν από εξωτερικές οντότητες/καταναλωτές αυτών των υπηρεσιών.

Οι υπηρεσίες αντιπροσωπεύουν ανεξάρτητες λειτουργικές μονάδες που μπορούν να χρησιμοποιηθούν ανεξάρτητα, μπορούν όμως να χρησιμοποιηθούν προκειμένου να χτιστούν σύνθετες υπηρεσίες βασισμένες σε υπηρεσιο-κεντρικές αρχιτεκτονικές.

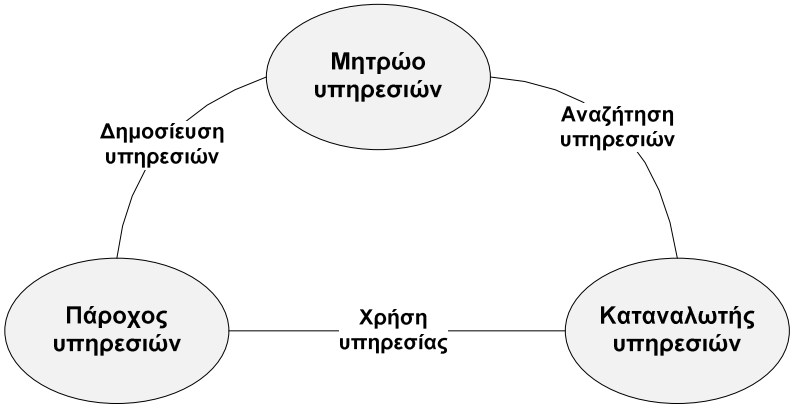
Οι υπηρεσίες ανάλογα με τη εύρος/πολυπλοκότητα των λειτουργιών που παρέχουν μπορούν να κατηγοριοποιηθούν ως εξής:

- Επιχειρησιακές υπηρεσίες που παρέχουν συγκεκριμένη και ενδεχομένως πολύπλοκη λειτουργικότητα που ορίζεται σε επιχειρησιακούς όρους και αντιστοιχεί σε επιχειρησιακές διαδικασίες, όπως για παράδειγμα μία ολοκληρωμένη ηλεκτρονική πληρωμή.
- Βασικές υπηρεσίες που παρέχουν περιορισμένου εύρους λειτουργικότητα, όπως για παράδειγμα η αναζήτηση αποθέματος ενός αγαθού που πωλείται σε ένα ηλεκτρονικό κατάστημα. Συχνά περιλαμβάνονται μέσα σε πιο σύνθετες επιχειρησιακές υπηρεσίες.
- Τεχνικές υπηρεσίες που δεν σχετίζονται άμεσα με μία επιχειρησιακή υπηρεσία και συνήθως υποστηρίζουν άλλες υπηρεσίες παρέχοντας πολύ βασική λειτουργικότητα, που μπορεί να είναι εξαρτώμενη από συγκεκριμένες τεχνολογίες υλοποίησης, όπως για παράδειγμα μετατροπή δεδομένων ή καταγραφή συμβάντων.

1.2 Τι είναι οι Web Services

Οι Web Services είναι αρθρωτές, αυτο-περιεχόμενες εφαρμογές που εμπεριέχονται στον εαυτό τους ή λογική εφαρμογής που έχει αναπτυχθεί πάνω σε ένα συγκεκριμένο σύνολο προτύπων. Οι web services δεν είναι συνήθως πλήρεις εφαρμογές, παρότι δεν υπάρχουν περιορισμοί από το πρότυπο σχετικά με το μέγεθος ή την πολυπλοκότητα που μπορεί να έχουν. Στην πραγματικότητα χρησιμοποιούνται περισσότερο για να ενεργοποιήσουν υπάρχουσες εφαρμογές παρά ως ολοκληρωμένες εφαρμογές.

Το πρότυπο Web Service έχει αναπτυχθεί από το World Wide Web Consortium (W3C, www.w3c.org). Στην πραγματικότητα οι web services είναι ένα σύνολο τεχνολογιών που παρέχουν τη δυνατότητα μέσω δικτύων και συνήθως του διαδικτύου να επιτευχθεί η σύνδεση υπαρχόντων υπηρεσιών (services). Στη πράξη υπάρχει ένας πάροχος μίας υπηρεσίας και ένας καταναλωτής, και ενδεχομένως ένα μητρώο (registry) για τις υπηρεσίες, όπως φαίνεται στην παρακάτω εικόνα [3, σελ 24]:



Εικόνα 1 Οι ρόλοι και οι σχέσεις σε μία υπηρεσία

Το World Wide Web Consortium περιγράφει τις Web Services ως εξής [14]:

> *"Οι Web Services παρέχουν τυποποιημένες μεθόδους διαλειτουργικότητας μεταξύ διαφορετικών εφαρμογών λογισμικού, εκτελούνται σε διαφορετικές πλατφόρμες ή/και πλαίσια (frameworks). Χαρακτηρίζονται από το μεγάλο βαθμό διαλειτουργικότητας και επεκτασιμότητας, καθώς και την μηχανικά προσβάσιμη περιγραφή χάρις στη χρήση της XML. Έχουν τη δυνατότητα συνδυασμού μέσω χαλαρών συζεύξεων (loosely coupled) να επιτύχουν πολύπλοκες λειτουργίες. Εφαρμογές που παρέχουν απλές υπηρεσίες μπορούν να αλληλεπιδρούν μεταξύ τους προκειμένου να παρέχουν εξελιγμένες (sophisticated) υπηρεσίες με προστιθέμενη αξία."*

Οι Web Services παρέχουν ένα αποδοτικό τρόπο για την κοινή χρήση εφαρμογών ανάμεσα σε διαφορετικά υπολογιστικά συστήματα, απομακρυσμένα ή όχι, κάτω από κοινό έλεγχο ή όχι, εντός του ίδιου οργανισμού ή μεταξύ διαφορετικών οργανισμών, που μπορεί να χρησιμοποιούν διαφορετικές τεχνολογίες λογισμικού και εργαλεία ανάπτυξης εφαρμογών. Παρέχουν τη δυνατότητα της διαλειτουργικότητας μεταξύ εφαρμογών με τη χρήση διαδικτυακών προτύπων. Βασικό χαρακτηριστικό τους είναι ότι δεν έχουν επίγνωση κατάστασης (stateless).

Δύο βασικές προσεγγίσεις υπάρχουν στην υλοποίηση των Web Services:

- Με τη χρήση τυποποιημένων προδιαγραφών που ορίζουν το μορφότυπο των λειτουργιών και των μηνυμάτων εκ των προτέρων. Οι SOAP Web Services ανήκουν σε αυτή την κατηγορία.
- Με τη χρήση χαλαρών συζεύξεων όπου το σύνολο των επόμενων λειτουργιών που μπορεί να καλέσει ο καταναλωτής υποδεικνύεται από την τελευταία απόκριση της υπηρεσίας. Οι RESTfull Web Services ανήκουν σε αυτή την κατηγορία

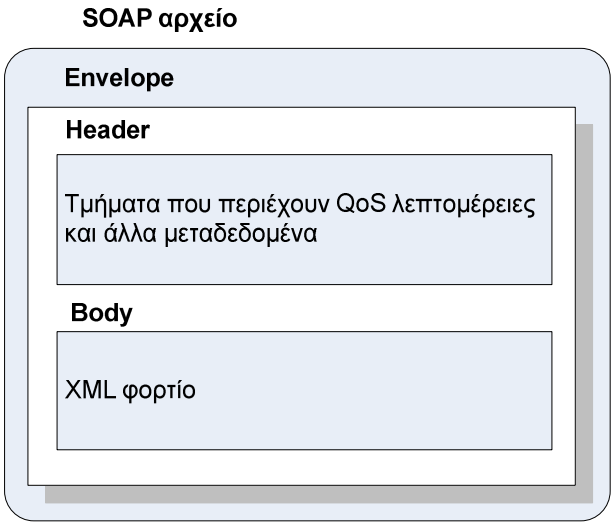
1.2.1 Η στοιβάδα των τεχνολογιών Web Services

Οι Web Services από μόνες τους δεν είναι σε θέση να προσφέρουν ποιότητα υπηρεσιών(Quality of Service, QoS), όπως ασφάλεια, ατομικές συναλλαγές, χαρακτηριστικά που υποστηρίζονται από άλλες τεχνολογίες. Για να αντιμετωπιστεί αυτό το μειονέκτημα ένα πλήθος από πρόσθετα πρότυπα έχει σχεδιαστεί:

- **WS-Security**, υποστηρίζει αυθεντικοποίηση και ασφάλεια σε επίπεδο μηνύματος, και επιτρέπει ασφαλή επικοινωνία με χρήση Web Services.
- **WS-Coordination**, ορίζει ένα πλαίσιο συντονισμού για τις Web Services και αποτελεί το θεμέλιο για τα WS-AtomicTransaction και WS-BusinessActivity.
- **WS-AtomicTransaction**, υποστηρίζει κατανεμημένες συναλλαγές και συναλλαγές που μπορούν να χαρακτηριστούν ACID (Atomicity, Consistency, Isolation, Durability).
- **WS-BusinessActivity**, ορίζει μακρόβιες (long-running) συναλλαγές ή αλλιώς συναλλαγές αντιστάθμισης (compensation transactions)
- **WS-ReliableMessaging**, υποστηρίζει την αξιόπιστη επικοινωνία
- **WS-Addressing**, ορίζει ζητήματα συντονισμού και διόδευσης
- **WS-Inspection**, υποστηρίζει τη δυναμική ενδοσκόπηση των περιγραφών των Web Services
- **WS-Policy**, ορίζει πως δηλώνονται οι πολιτικές και πως ανταλλάσσονται μεταξύ Web Services
- **WS-Eventing**, ορίζει ένα μοντέλο συμβάντων για την ασύγχρονη ενημέρωση των συνεργαζόμενων μερών με Web Services

1.2.2 SOAP Web Services

Το πρότυπο SOAP είναι πρότυπο του World Wide Web Consortium. Περιγράφει την αποστολή μηνυμάτων μεταξύ παρόχων και καταναλωτών υπηρεσιών, χρησιμοποιώντας το πρότυπο XML. Ένα μήνυμα SOAP είναι ένα κείμενο (document) XML που περιέχει ένα μήνυμα που ενθυλακώνεται από μεταδεδομένα που σχετίζονται με την μεταφορά του κειμένου. Η βασική του μορφή φαίνεται στην παρακάτω εικόνα [3, σελ. 106]:



Εικόνα 2 Ένα απλό SOAP κείμενο

Το φορτίο (payload) εντός του στοιχείου body είναι το μήνυμα. Το στοιχείο header μπορεί να περιέχει μεταδεδομένα σχετικά με το μήνυμα. Η δομή του SOAP κειμένου είναι η ίδια είτε πρόκειται για αίτηση προς μία Web Service είτε πρόκειται για την απάντηση που επιστρέφει.

Οι SOAP Web Services χαρακτηρίζονται από μία τυποποιημένη μέθοδο επικοινωνίας μεταξύ των εφαρμογών, που παρέχει τα παρακάτω πλεονεκτήματα [3]:

- Τα μηνύματα είναι αυστηρά προδιαγεγραμμένα, ο καταναλωτής είναι σε θέση να γνωρίζει τι δεδομένα περιμένει στην απάντηση
- Λόγω της αυξημένης τυποποίησης υπάρχει πλήθος εργαλείων που υποστηρίζουν την ανάπτυξη SOAP Web Services
- Υποστηρίζονται πολλά πρωτόκολλα για την επικοινωνία μεταξύ καταναλωτή και παρόχου, όπως: SOAP over HTTP και SOAP over JMS
- Υποστηρίζονται επιπλέον χαρακτηριστικά και πρότυπα όπως WS-Addressing, WS-Security και άλλα

Μειονέκτημα του προτύπου SOAP θεωρείται η αυστηρή τυποποίηση που έχει ως αποτέλεσμα την αύξηση του μεγέθους των μηνυμάτων μεταξύ καταναλωτών και παρόχων υπηρεσιών καθώς και βαρύτερων υλοποιήσεων, προκειμένου να υποστηριχτεί το πρότυπο SOAP.

1.2.3 RESTFull Web Services

Εκτός από τις Web Sevices που ακολουθούν το πρότυπο SOAP, υπάρχει η δυνατότητα ανάπτυξης Web Services που δεν ακολουθούν συγκεκριμένο πρότυπο, αλλά η υλοποίηση τους βασίζεται σε υπάρχουσες γνωστές τεχνολογίες και πρότυπα. Αυτές Web Services είναι γνωστές ως REST-based από τον όρο Represantational State Transfer (REST) και ο όρος REST αντιστοιχεί περισσότερο σε μία αρχιτεκτονική μορφή ανάπτυξης εφαρμογών . Τα συνήθη πρότυπα στα οποία βασίζονται τέτοιες υπηρεσίες είναι τα XML, HTTP και οι μέθοδοι του GET, HEAD, POST, PUT και DELETE και MIME (Multipurpose Internet Mail Extensions). Όταν καλείται μία REST Web Service, επιστρέφει την απάντησή της σε XML, το οποίο όμως δεν ακολουθεί συγκεκριμένο πρότυπο. Η δομή του εξαρτάται από την Web Service και το πώς αυτή υλοποιείται στην πράξη. Οι υλοποίηση REST Web Services θεωρείται συχνά ως ελαφρύτερη και ευκολότερη από μία αντίστοιχη υλοποίηση σε SOAP.

1.2.4 Web Services Description Language (WSDL)

Η Web Services Description Language (WSDL) αποτελεί τη βάση των web services. Η WSDL βασίζεται στο πρότυπο XML και είναι μία γλώσσα περιγραφής διεπαφών (interface description language). Περιγράφει τη λειτουργικότητα που παρέχει μία web service, δηλαδή πως καλείται η web service, ποιες παραμέτρους έχει ως είσοδο και τι δομές δεδομένων επιστρέφει [15]. Η τρέχουσα έκδοση είναι η WSDL 2.0 και αποτελεί σύσταση του W3C. Σύμφωνα με το ορισμό του W3C [16]:

"Η WSDL είναι ένα XML μορφότυπο που περιγράφει δικτυακές υπηρεσίες ως σύνολο απολήξεων (endpoints) που εκτελούν λειτουργίες σε μηνύματα που περιέχουν πληροφορία σχετική με κείμενο ή διαδικασίες. Οι λειτουργίες και τα μηνύματα περιγράφονται αφηρημένα, και στη συνέχεια δεσμεύονται σε ένα συγκεκριμένο δικτυακό πρωτόκολλο και μορφότυπο μηνύματος προκειμένου να ορίσουν μία απόληξη."

Το WSDL περιγράφει μία web service ως συλλογές από δικτυακές απολήξεις (endpoints) ή θύρες (ports). Μία θύρα σχετίζεται με μία δικτυακή διεύθυνση μέσω ενός binding και μία συλλογή θυρών ορίζει μία υπηρεσία. Το σύνολο των δεδομένων που ανταλλάσσονται με μία υπηρεσία περιγράφονται ως μηνύματα. Τόσο οι πόρτες όσο και τα μηνύματα περιγράφονται αφηρημένα, αποτελώντας στην πραγματικότητα τη δημόσια διεπαφή (public interface) της web service.

Παρακάτω ακολουθεί μία περιγραφή των οντοτήτων που περιλαμβάνονται σε ένα αρχείο WSDL στη τρέχουσα έκδοση 2.0 και στην έκδοση 1.1 που χρησιμοποιείται στο πρότυπο BPEL.

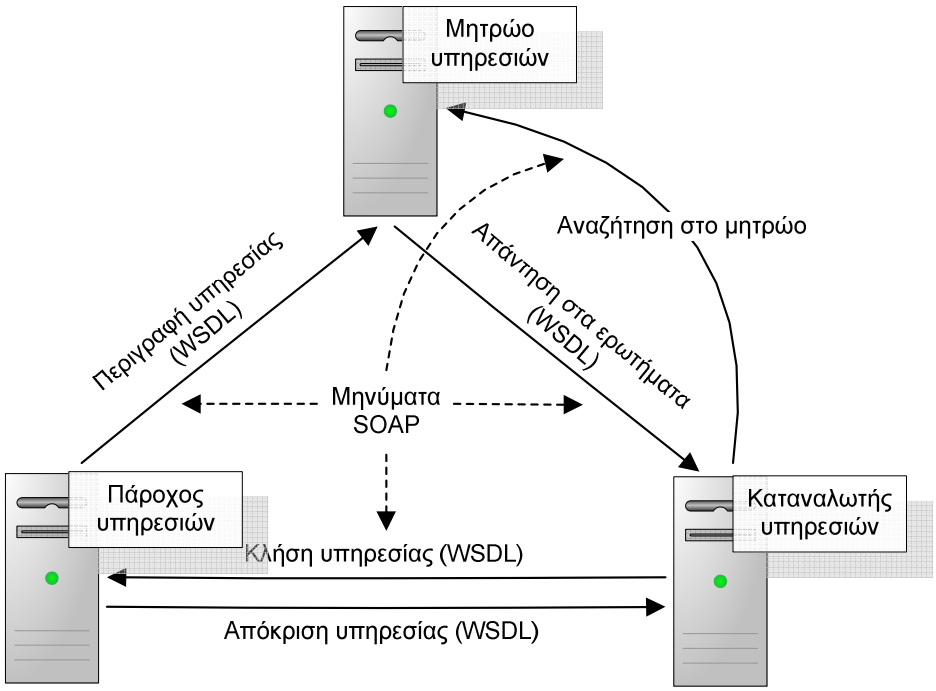
WSDL 1.1	WSDL 2.0	Περιγραφή
Service	Service	Περιέχει το σύνολο των συναρτήσεων που δημοσιεύονται στη συγκεκριμένη web service.
Port	Endpoint	Ορίζει τη διεύθυνση ή το σημείο σύνδεσης στη web service. Συνήθως είναι ένα HTTP URL αλφαριθμητικό.
Binding	Binding	Προδιαγράφει τη διεπαφή και ορίζει το SOAP binding style (RPC ή document), το πρωτόκολλο transport (SOAP), καθώς και τις λειτουργίες (operations).
PortType	Interface	Ορίζει μία web service, τις λειτουργίες που μπορούν να εκτελεστούν και τα μηνύματα που απαιτούνται για την εκτέλεση.
Operation	Operation	Ορίζει τις SOAP actions και την κωδικοποίηση των μηνυμάτων.
Message	n/a	Περιέχει την πληροφορία που απαιτείται για την εκτέλεση μίας λειτουργίας και αποτελείται από ένα ή περισσότερα μέρη (parts). Ένα μέρος περιγράφει το περιεχόμενο ενός μηνύματος και είναι μοναδικό στο ίδιο μήνυμα.
Types	Types	Περιγράφει τα δεδομένα με τη γλώσσα XML Schema

Πίνακας 1 WSDL οντότητες

1.2.5 Universal Description, Discovery, and Integration (UDDI)

Ο ορισμός μίας web service αποθηκεύεται σε ένα μητρώο. Σύνηθες πρότυπο registry αποτελεί το Universal Description, Discovery, and Integration (UDDI). Το πρότυπο UDDI ορίζει ία τυποποιημένη μέθοδο για τη δημοσίευση και την ανακάλυψη υπηρεσιών. Τα service registries, όπως το UDDI χρησιμοποιούνται κυρίως για την αναζήτηση υπηρεσιών σε πραγματικό χρόνο και όχι στη περίπτωση όπου υπηρεσίες πρέπει να είναι γνωστές κατά την ανάπτυξη μίας εφαρμογής. Σε ένα UDDI registry μπορεί να αναζητηθούν διαθέσιμες web services που παρέχουν διαφορετικοί πάροχοι.

Στην επόμενη εικόνα [2, σελ. 23] παρουσιάζεται η διαδικασία αναζήτησης μίας Web Service που υλοποιείται με SOAP, η κλήση της και η επιστροφή του αποτελέσματος της εκτέλεσης της.



Εικόνα 3 Τα βασικά βήματα χρήσης μίας Web Service

1. Ο πάροχος μίας web services περιγράφει την υπηρεσία σε ένα WSDL αρχείο που δημοσιεύεται στο μητρώο των υπηρεσιών χρησιμοποιώντας UDDI ή κάποιο άλλο πρότυπο.
2. Ο καταναλωτής αποστέλλει ερωτήματα στον κατάλογο για να εντοπίσει την υπηρεσία που αναζητά και να ενημερωθεί για τον τρόπο με τον οποίο θα επικοινωνήσει με την υπηρεσία.
3. Μέρος της WSDL περιγραφής της υπηρεσίας αποστέλλεται στον καταναλωτή προκειμένου να ενημερωθεί για τις προδιαγραφές της κλήσης και της απάντησης που θα λάβει.
4. Ο καταναλωτής χρησιμοποιεί τη WSDL προκειμένου να συντάξει και να καλέσει την υπηρεσία
5. Ο πάροχος επιστρέφει την απάντηση στον καταναλωτή

Το σύνολο των μηνυμάτων που ανταλλάσσονται χρησιμοποιούν το πρότυπο SOAP. Το SOAP κατά κανόνα χρησιμοποιεί το πρότυπο HTML για την ανταλλαγή των μηνυμάτων.

1.3 Τι είναι Service Oriented Architecture (SOA)

Η Service Oriented Architecture (SOA) αποτελεί ένα σχεδιαστικό πρότυπο (design pattern) για τη σχεδίαση της αρχιτεκτονικής μίας εφαρμογής λογισμικού που βασίζεται σε αυτοτελείς εφαρμογές που προσφέρουν τη λειτουργικότητα τους σε άλλες εφαρμογές ως υπηρεσίες. Είναι ένας τρόπος οργάνωσης των διαδικασιών και των εφαρμογών που τις υλοποιούν προσανατολισμένη στη λογική των υπηρεσιών. Η λειτουργικότητα των εφαρμογών γίνεται διαθέσιμη αυτοματοποιημένα μέσω υπηρεσιών που είναι επαναχρησιμοποιούμενες. Βασικό χαρακτηριστικό της SOA είναι ότι δεν απαιτείται γνώση του τρόπου με τον οποίο υλοποιείται μία υπηρεσία. Είναι ανεξάρτητη από συγκεκριμένους κατασκευαστές, προϊόντα ή τεχνολογίες.

Η SOA είναι μία αρχιτεκτονική προσέγγιση με βασικό στόχο και πλεονέκτημα ταυτόχρονα τη δυνατότητα της αλληλεπίδρασης μεταξύ των υπηρεσιών, με ταυτόχρονη όμως ελαχιστοποίηση τυχόν αλληλεξαρτήσεων μεταξύ τους, δηλαδή την απόζευξη των υπηρεσιών (decoupling). Το χαρακτηριστικό αυτό δίνει τη δυνατότητα ευελιξίας (agility), δηλαδή τη γρήγορη υλοποίηση αλλαγών ή προσθήκη νέων λειτουργιών σε μία υλοποίηση που βασίζεται σε τέτοια αρχιτεκτονική.

Η SOA μπορεί να αναλυθεί από διάφορες πλευρές: από την πλευρά των επιχειρησιακών διαδικασιών που υποστηρίζει μία υλοποίηση βασισμένη σε SOA, από την πλευρά της

αρχιτεκτονικής που ακολουθεί η υλοποίηση και τέλος από την πλευρά της ίδιας της υλοποίησης. Από επιχειρησιακή άποψη η SOA υπόσχεται επιχειρησιακή ευελιξία, δηλαδή τη δυνατότητα προσαρμογής των επιχειρησιακών διαδικασιών και των τεχνολογικών υποδομών σε νέες απαιτήσεις.

Βασικά χαρακτηριστικά της SOA είναι [4]:

- Χαλαρή σύζευξη (loose coupling) που επιτρέπει τις αλλαγές στην υλοποίηση με ελάχιστες ή καθόλου επιπτώσεις στις υπόλοιπες υπηρεσίες
- Service contract που αντιπροσωπεύει τη περιγραφή της υπηρεσίας και το τρόπο που καλείται
- Η αφαιρετική προσέγγιση της υλοποίησης, δηλαδή η υπηρεσία δημοσιοποιεί μόνο ότι περιλαμβάνεται στην περιγραφή της, αποκρύπτοντας την υλοποίηση
- Αυτονομία, δηλαδή το γεγονός ότι μία υπηρεσία ασκεί έλεγχο μόνο στον εαυτό της
- Δυνατότητα επαναχρησιμοποίησης των υπηρεσιών
- Δυνατότητα σύνθεσης των υπηρεσιών σε σύνθετες (composite) υπηρεσίες
- Statelessness, δηλαδή την αποφυγή της συσχέτισης της κατάστασης που βρίσκεται μία υπηρεσία σε σχέση με μία συγκεκριμένη δραστηριότητα που εκτελείται
- Διαλειτουργικότητα μεταξύ των υπηρεσιών
- Ανακαλυψιμότητα μέσω προτυποποιημένων μεθόδων

1.3.1 Αρχιτεκτονικά χαρακτηριστικά

- Υπηρεσίες
- Αυτο-προσδιοριζόμενες, προτυποποιημένες διεπαφές και coarse granulation
- Ανταλλαγή μηνυμάτων
- Υποστήριξη σύγχρονης και ασύγχρονης επικοινωνίας
- Χαλαρή ζεύξη
- Επαναχρησιμοποίηση
- Μητρώα και αποθετήρια υπηρεσιών
- Ποιότητα υπηρεσιών
- Σύνθεση υπηρεσιών σε επιχειρησιακές διαδικασίες

1.3.2 Συστατικά μέρη της SOA

- BPEL
- Υπηρεσίες
- Επιχειρησιακός Δίαυλος Υπηρεσιών (Enterprise Service Bus)
- Μητρώα και αποθετήρια
- Υποστήριξη ανθρώπινων ενεργειών
- Παρακολούθηση διαδικασιών και Παρακολούθηση Επιχειρησιακών Δραστηριοτήτων (Business Activity Monitoring)
- Συστήματα Διαχείρισης Επιχειρησιακών Κανόνων (Business Rules Management Systems)
- Προσαρμογές (adapters)

1.4 Επιχειρησιακές διαδικασίες και BPEL

Οι οργανισμοί βασίζονται σήμερα στις πληροφοριακές υποδομές τους για να υποστηρίζουν τις επιχειρησιακές τους λειτουργίες. Βασική προϋπόθεση για την ικανοποιητική υποστήριξη είναι η ευθυγράμμιση των εφαρμογών με τις επιχειρησιακές διαδικασίες. Ως επιχειρησιακή διαδικασία εννοούμε [17]:

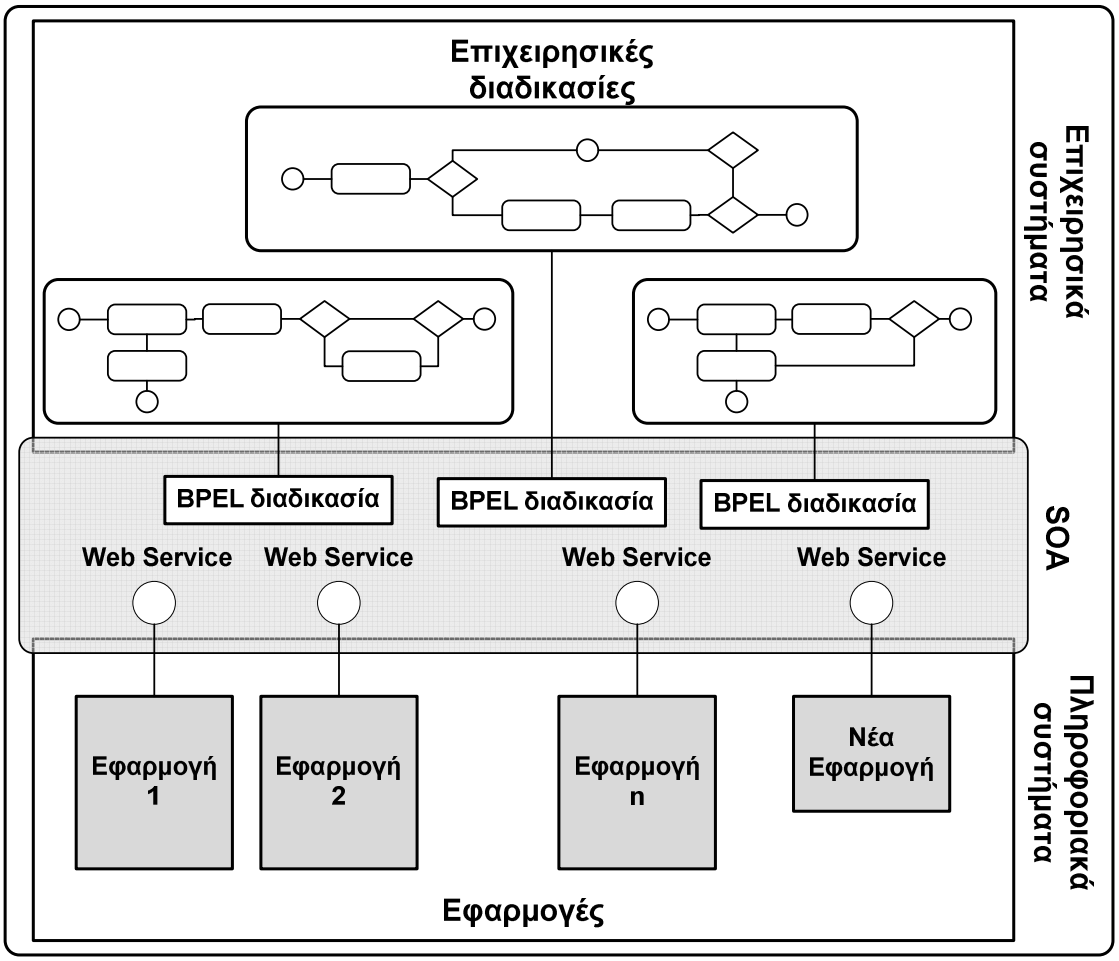
"μία συλλογή σχετικών, δομημένων δραστηριοτήτων ή ενεργειών, που παράγουν ένα συγκεκριμένο αποτέλεσμα για ένα συγκεκριμένο πελάτη ή πελάτες"

Στη πράξη οι περισσότεροι οργανισμοί διαθέτουν συχνά πολύπλοκες πληροφοριακές υποδομές, που αποτελούνται από ετερογενή συστήματα που μπορεί να είναι συγκεντρωμένα εντός του οργανισμού ή κατανεμημένα ακόμη και εκτός, από παλαιότερες και νεότερες εφαρμογές, που πολλές φορές βασίζονται σε διαφορετικές τεχνολογίες και αρχιτεκτονικές. Επιπρόσθετα οι διαδικασίες μεταβάλλονται συχνά. Είναι σημαντικό από επιχειρησιακή άποψη οι πληροφοριακές υποδομές να είναι σε θέση να υποστηρίζουν τις επιχειρησιακές διαδικασίες ολοκληρωμένα από την αρχή μέχρι το τέλος, δηλαδή στο σύνολο τους, με ευελιξία και χωρίς να είναι απαραίτητη η γνώση των λεπτομερειών των διαδικασιών στους χρήστες τους.

Η Business Process Execution Language (BPEL) ανήκει στην κατηγορία των γλωσσών που περιγράφουν επιχειρησιακές διαδικασίες. Είναι σχεδιασμένη να εκτελεί επιχειρησιακές διαδικασίες χρησιμοποιώντας ένα εξυπηρετητή διαδικασιών (process server). Έχει σαν σκοπό την δημιουργία ενός περιβάλλοντος για την εύκολη, γρήγορη και ευέλικτη ανάπτυξη των επιχειρησιακών διαδικασιών, την εκτέλεση τους, την παρακολούθηση τους και την άμεση απόκριση σε αλλαγές χωρίς να απαιτείται μεγάλος κόπος.

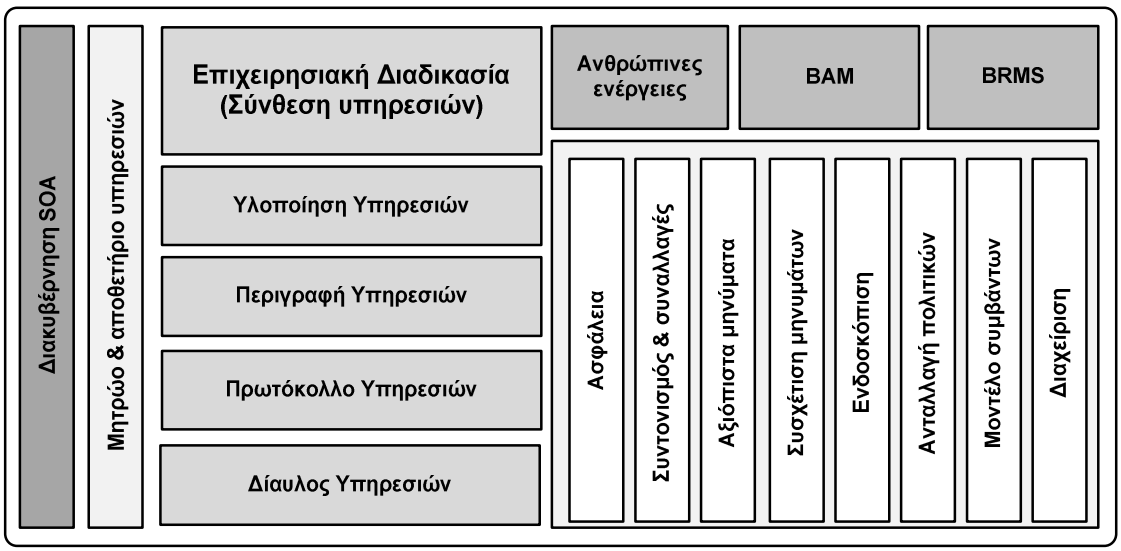
Περιλαμβάνει ένα ικανό λεξιλόγιο για την περιγραφή της συμπεριφοράς που μπορεί να εμφανίζουν οι επιχειρησιακές διαδικασίες. Έχει τη δυνατότητα να περιγράψει τη σύνθεση, οργάνωση και συντονισμό Web Services με στόχο την υποστηρίξει των επιχειρησιακών διαδικασιών. Αποτελεί ένα από τα βασικά εργαλεία για την υλοποίηση υπηρεσιο-κεντρικών αρχιτεκτονικών. Βασικό χαρακτηριστικό της BPEL είναι ότι συνεργάζεται με άλλες Web Services και εξωτερικεύει τη λειτουργικότητα που υλοποιεί ως Web Service.

Οι θέση των διαδικασιών που μοντελοποιούνται με BPEL, εντός της υπηρεσιο-κεντρικής αρχιτεκτονικής φαίνεται στην επόμενη εικόνα [4 σελ. 12]:

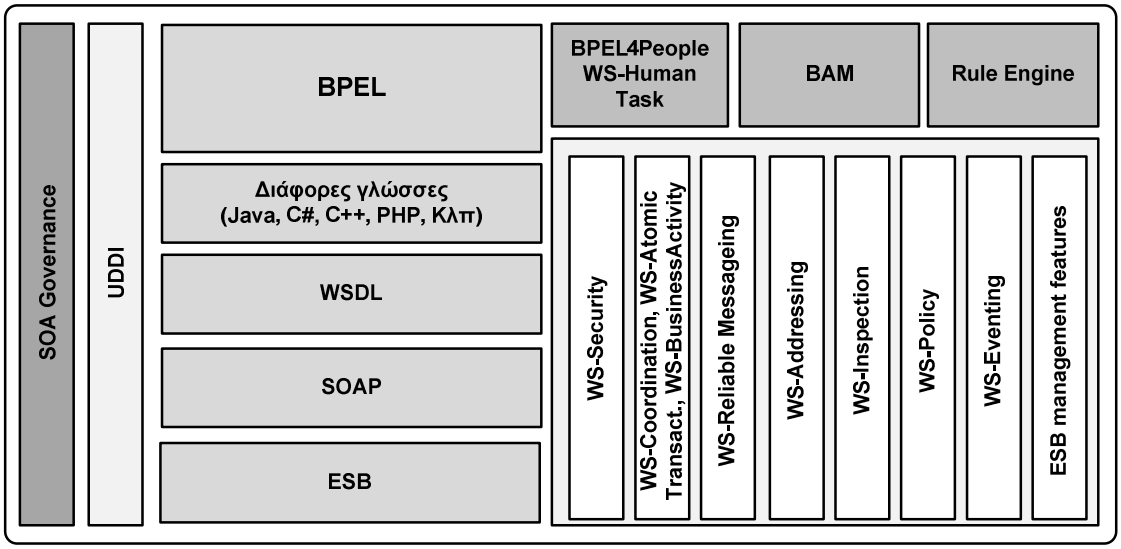


Εικόνα 4 SOA, BPEL διαδικασίες, πληροφοριακά και επιχειρησιακά συστήματα

Στις επόμενες δύο εικόνες [4 σελ. 20] απεικονίζεται η SOA από τεχνολογική σκοπιά , δηλαδή από τα συστατικά μέρη που εμπλέκονται στη συγκεκριμένη αρχιτεκτονική φιλοσοφία, καθώς και οι συγκεκριμένες τεχνολογίες , πρότυπα και πρωτόκολλα που τα υλοποιούν.



Εικόνα 5 Τα συστατικά μέρη της SOA

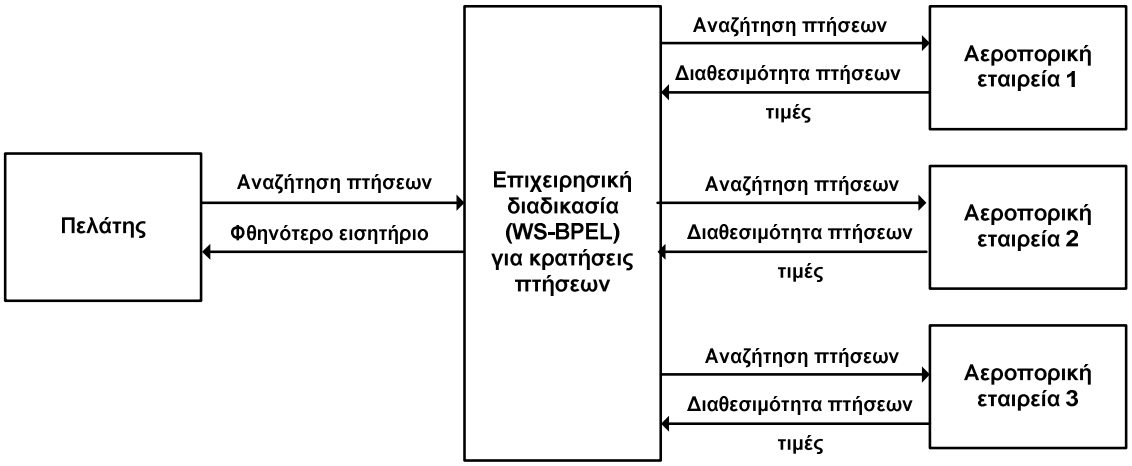


Εικόνα 6 Οι τεχνολογίες στη SOA

Τι είναι η WS-BPEL

Η Web Services Business Process Execution Language (WS-BPEL) είναι σήμερα ένα πρότυπο του Organisation for the Advancement of Structured Information Standards (OASIS) [10]. Η τρέχουσα έκδοση είναι η WS-BPEL 2.0. Η πρώτη έκδοση ως BPEL αναπτύχθηκε το 2002 από τις εταιρείες IBM, BEA και Microsoft. Βασίστηκε πάνω στην XLANG της Microsoft και την Web Service Flow Language (WSFL) της IBM. Στη συνέχεια στην ανάπτυξη εντάχθηκαν οι εταιρείες SAP και Siebel και το Μάρτιο του 2003 δημοσιοποίησαν την έκδοση 1.1. Στη συνέχεια η BPEL υποβλήθηκε στον OASIS για προτυποποίηση. Η σημερινή έκδοση δημοσιοποιήθηκε το 2007.

Η WS-BPEL είναι η πλέον διαδομένη γλώσσα αυτοματοποίησης επιχειρησιακών διαδικασιών. Αποτελεί ένα μοντέλο για την περιγραφή της συμπεριφοράς των επιχειρησιακών διαδικασιών βασιζόμενο στις αλληλεπιδράσεις τους, περιλαμβάνοντας την κατάσταση τους, με τις οντότητες με τις οποίες συνεργάζονται (partners). Με τη WS-BPEL μπορούμε να υλοποιήσουμε μία επιχειρησιακή διαδικασία μέσω της ολοκλήρωσης πολλών Web Services σε μία ροή επιχειρησιακών διαδικασιών. Μια επιχειρησιακή διαδικασία υλοποιημένη σε WS-BPEL ορίζει πως θα συντονιστεί μία παρουσία (instance) της διαδικασίας και οι συνεργαζόμενες υπηρεσίες. Χαρακτηρίζεται από φορητότητα (portability) μεταξύ πλατφορμών διαφορετικών κατασκευαστών. Μπορεί να αποτελέσει τη βάση για την υλοποίηση ετερογενών, κατανεμημένων εφαρμογών. Ένα παράδειγμα φαίνεται στην επόμενη εικόνα [5 σελ. 22]:



Εικόνα 7 Παράδειγμα επιχειρησιακής διαδικασίας κράτησης πτήσεων υλοποιημένης σε WS-BPEL

Η WS-BPEL παρέχει τη δυνατότητα ορισμού απλών ή πολύπλοκων διαδικασιών, εκτελέσιμων ή και αφηρημένων διαδικασιών. Οι εκτελέσιμες επιχειρησιακές διαδικασίες μοντελοποιούν την πραγματική συμπεριφορά μίας οντότητας σε μία επιχειρησιακή συναλλαγή. Οι αφηρημένες επιχειρησιακές διαδικασίες είναι μερικώς ορισμένες διαδικασίες που δεν έχουν στόχο την εκτέλεση αλλά μπορούν να έχουν ένα περιγραφικό ρόλο. Η WS-BPEL ορίζει ένα μοντέλο ολοκλήρωσης που παρέχει διαλειτουργικότητα και διευκολύνει την επέκταση της ολοκλήρωσης αυτοματοποιημένων διαδικασιών εντός ενός οργανισμού αλλά και σε διεπιχειρησιακό περιβάλλον.

Η WS-BPEL 2.0 χρησιμοποιεί τα παρακάτω XML πρότυπα:

- WSDL 1.1
- XML schema 1.0
- XPath 1.0
- XSLT 1.0

Τα μηνύματα WSDL και οι ορισμοί τύπων δεδομένων μέσω της XML Schema παρέχουν το μοντέλο δεδομένων που χρησιμοποιεί μία WS-BPEL process. Η WSDL επιτρέπει σε μία υπηρεσία να χρησιμοποιηθεί μέσα σε μία σύνθετη συλλογή υπηρεσιών. Τα πρότυπα XPath και XSLT υποστηρίζουν τη διαχείριση των δεδομένων. Όλοι οι εξωτερικοί πόροι και συνεργάτες

ορίζονται ως WSDL services. Η WS-BPEL παρέχει επεκτασιμότητα προκειμένου να υποστηριχθούν μελλοντικές εκδόσεις αυτών των προτύπων.

2.1 Τα βασικά χαρακτηριστικά της WS-BPEL

Η WS-BPEL έχει τα παρακάτω βασικά χαρακτηριστικά:

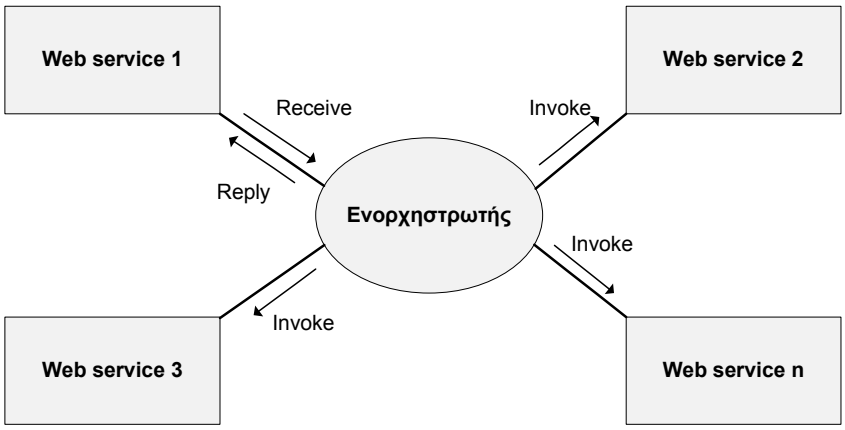
1. Αποτελεί ένα αναδρομικό συγκεντρωτικό μοντέλο (recursive aggregation) για τις Web services

   Συγκεντρωτικό γιατί ομαδοποιεί ένα σύνολο Web Services σε μία νέα Web Service που αντιστοιχεί σε μία επιχειρησιακή διαδικασία.

   Αναδρομικό γιατί αυτή η νέα Web Service μπορεί με τη σειρά της να ομαδοποιηθεί με άλλες Web Services σε μία νέα Web Service που αντιστοιχεί με τη σειρά της σε μία επιχειρησιακή διαδικασία.

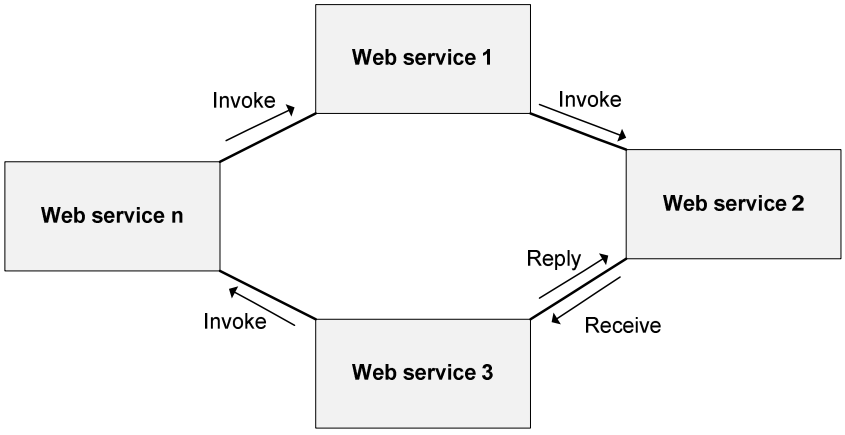
2. Είναι μία γλώσσα ενορχήστρωσης (orchestration).

   Η ενορχήστρωση είναι ένα μοντέλο για το συντονισμό υπηρεσιών με τη βοήθεια μίας κεντρικής διαδικασίας. Οι υπηρεσίες που συμμετέχουν δεν έχουν γνώση της συμμετοχής τους σε μία σύνθετη διαδικασία, μόνο η κεντρική διαδικασία το γνωρίζει. Συνεπώς η ενορχήστρωση που έχει συγκεντρωτικό χαρακτήρα και χαρακτηρίζεται από το ρητό και σαφή ορισμό των λειτουργιών και της διαδοχής των κλήσεων των υπηρεσιών. Κατά κανόνα χρησιμοποιείται σε υλοποιήσεις επιχειρησιακών διαδικασιών εντός ενός οργανισμού. Στο επόμενο σχήμα [5 σελ. 20] φαίνεται η βασική μορφή της ενορχήστρωσης υπηρεσιών:



Εικόνα 8 Ενορχήστρωση Web Services

Αντίθετα στο μοντέλο της χορογραφίας (choreography) υπηρεσιών, κάθε υπηρεσία γνωρίζει πότε και ποια πρέπει να καλέσει προκειμένου να υλοποιηθεί μια επιχειρησιακή διαδικασία, όπως φαίνεται στο επόμενο σχήμα [5 σελ. 20]:



Εικόνα 9 Χορογράφηση Web Services

Η ενορχήστρωση έχει τα παρακάτω πλεονεκτήματα:

- Είναι γνωστό ποια οντότητα είναι υπεύθυνη για την ολοκληρωμένη εκτέλεση του επιχειρησιακής διαδικασίας
- Είναι δυνατό να ολοκληρωθούν υπηρεσίες χωρίς να γνωρίζουν ότι αποτελούν τμήμα μία επιχειρησιακής διαδικασίας
- Παρέχει μεγαλύτερη ευελιξία στην υλοποίηση εναλλακτικών σεναρίων στη περίπτωση εμφάνισης σφαλμάτων

3. Εκτελέσιμες και αφηρημένες διαδικασίες

   Με τη WS-BPEL μπορούμε να περιγράψουμε λεπτομερώς την υλοποίηση μίας επιχειρησιακής διαδικασίας ώστε αυτή να είναι εκτελέσιμη μέσω μία μηχανής ενορχήστρωσης. Σε αυτή την περίπτωση ορίζουμε μία Web Service που αποτελεί τη σύνθεση των επιμέρους Web Services. Η διεπαφή της σύνθετης Web Service αποτελείται από port types, μέσω των οποίων παρέχονται λειτουργίες όπως στις συνήθεις Web Services.

Μπορούμε όμως να περιγράψουμε μόνο τη δημόσια ανταλλαγή μηνυμάτων μεταξύ των συμμετεχόντων σε μία διαδικασία, χωρίς να περιλαμβάνονται οι εσωτερικές λεπτομέρειες της ροής και χωρίς να είναι εκτελέσιμη, ακλουθώντας το παράδειγμα της χορογράφησης υπηρεσιών. Οι αφηρημένες διαδικασίες δεν είναι συνηθισμένες αλλά μπορούν να χρησιμοποιηθούν ως υποδείγματα (templates) για εκτελέσιμες διαδικασίες.

4. Programming in the large - programming in the small

   Η σύνθεση των υπηρεσιών σε επιχειρησιακές διαδικασίες μπορεί να πραγματοποιηθεί με τις γνωστές παραδοσιακές γλώσσες προγραμματισμού, όπως Java, C# και άλλες. Όμως η σύνθεση υπηρεσιών ενοποιούμε λειτουργικότητες σε μεγάλες υπηρεσίες και διαδικασίες, δηλαδή προγραμματίζουμε σε εύρος, κάτι που είναι διαφορετικό από τον συνήθη προγραμματισμό, που ασχολείται με την υλοποίηση σε επίπεδο λεπτομέρειας. Οι συνήθεις γλώσσες προγραμματισμού δύναται να οδηγήσουν συχνά σε μη ευέλικτες λύσεις, καθότι δεν είναι συχνά εύκολος ο διαχωρισμός μεταξύ της ροής σε επίπεδο διαδικασιών και της επιχειρησιακής λογικής, οι οποίες δεν πρέπει να είναι στενά συζευγμένες.

2.2 Οι στόχοι του προτύπου WS-BPEL

Ο κύριος στόχος του προτύπου είναι η τυποποίηση της διαδικασίας αυτοματοποίησης των αλληλεπιδράσεων μεταξύ Web Services. Οι αρχικοί στόχοι που τέθηκαν κατά την προτυποποίηση του WS-BPEL από τον OASIS ήταν (το πρότυπο αναφερόταν ως BPEL4WS) [6]:

1. Οι Web Services ως βάση

   Η BPEL4WS Θα πρέπει να ορίζει επιχειρησιακές διαδικασίες και να αλληλεπιδρά με εξωτερικές οντότητες μέσω λειτουργιών που παρέχονται από Web Services, και εκδηλώνονται ως Web Services οι οποίες ορίζονται με τη χρήση του προτύπου WSDL 1.1. Οι αλληλεπιδράσεις είναι αφηρημένες με την έννοια ότι εξαρτούνται από ορισμούς portType και όχι ορισμούς port.

2. XML ως τύπος

   Η BPEL4WS ορίζει επιχειρησιακές διαδικασίες βασιζόμενη στη γλώσσα XML. Δεν εμπλέκεται στη γραφική αναπαράσταση των διαδικασιών ούτε ορίζει συγκεκριμένη σχεδιαστική μεθοδολογία για τις διαδικασίες.

3. Κοινό σύνολο ιδεών (concepts)

   Η BPEL4WS Θα πρέπει να ορίζει ένα σύνολο ιδεών για την ενορχήστρωση Web Services, που θα χρησιμοποιούνται κοινά τόσο για την εξωτερική (αφηρημένη) όσο και για την εσωτερική (εκτελέσιμη) όψη μία επιχειρησιακής διαδικασίας. Τέτοιες επιχειρησιακές διαδικασίες ορίζουν τη συμπεριφορά μίας αυτόνομης οντότητας που συνήθως λειτουργία αλληλεπιδρώντας με άλλες παρόμοιες ομότιμες οντότητες.

4. Ελεγχόμενη συμπεριφορά

   Η BPEL4WS Θα παρέχει τόσο ιεραρχικό όσο και γραφικού χαρακτήρα καθεστώς ελέγχου, και θα επιτρέπει τη μεικτή χρήση τους με τρόπο κατά το δυνατόν συνεχή. Αυτό θα

έχει ως αποτέλεσμα την ελαχιστοποίηση του κατακερματισμού του χώρου της διαδικασίας μοντελοποίησης.

5. Χειρισμός δεδομένων

Η BPEL4WS Θα παρέχει περιορισμένες λειτουργίες χειρισμού δεδομένων, επαρκείς για τον απλό χειρισμό δεδομένων που απαιτείται για τον ορισμό διαδικασιών των σχετικών δεδομένων και τον έλεγχο ροών.

6. Ιδιότητες και συσχετίσεις (properties and correlations)

Η BPEL4WS Θα πρέπει να υποστηρίζει ένα μηχανισμό ταυτοποίησης για την παρουσία (instance) διαδικασιών που θα επιτρέπει τον ορισμό αναγνωριστικών παρουσιών διαδικασιών σε επίπεδο μηνυμάτων της εφαρμογής. Τα αναγνωριστικά παρουσιών θα πρέπει να ορίζονται σε σχέση με το συνεργάτη και μπορούν να τροποποιούνται στο χρόνο.

7. Χρόνος ζωής (lifecycle)

Η BPEL4WS Θα πρέπει να υποστηρίζει την έμμεση δημιουργία και τερματισμό των παρουσιών διαδικασιών ως βασικό μηχανισμό χρόνου ζωής. Προχωρημένες λειτουργίες χρόνου ζωής, όπως αναστολή, επανεκκίνηση θα μπορούν να προστεθούν σε επόμενες εκδόσεις για εμπλουτισμένη διαχείριση του χρόνου ζωής.

8. Μοντέλο μακροχρόνιων συναλλαγών (long-running transaction model)

Η BPEL4WS Θα πρέπει να ορίζει ένα μοντέλο μακροχρόνιων συναλλαγών που θα βασίζεται σε πρακτικά αποδεδειγμένες τεχνικές όπως δράσεις αντιστάθμισης (compensation actions) και χρήση εύρους για να υποστηρίζει ανάκτηση από σφάλματα των μακροχρόνιων επιχειρησιακών διαδικασιών.

9. Άρθρωση (modularization)

Η BPEL4WS Θα πρέπει να χρησιμοποιεί Web Services ως μοντέλο για την σύνθεση και αποσύνθεση των διαδικασιών. Συνδυάζοντας αυτή τη προσέγγιση με WS-Policy το μοντέλο αυτό μπορεί να ενισχυθεί.

10. Σύνθεση με υπάρχουσα λειτουργικότητα Web Service

Η BPEL4WS Θα βασιστεί κατά το δυνατόν πάνω σε συμβατά υπάρχοντα και προτεινόμενα πρότυπα Web Services, με τρόπο που θα επιτρέπει τη σύνθεση και την άρθρωση.

2.3 Πλεονεκτήματα της WS-BPEL

Η WS-BPEL παρέχει το επίπεδο ενορχήστρωσης το μοντέλο SOA μέσω του οποίου επιτυγχάνουμε τα παρακάτω πλεονεκτήματα, σύμφωνα με τον OASIS [7]:

1. Ένα βιομηχανικό πρότυπο για την περιγραφή και υλοποίηση επιχειρησιακών διαδικασιών, που μπορεί να υποστηρίζει πολύπλοκες απαιτήσεις και αποτελεί μία συνεκτική λύση για την ενορχήστρωση διαδικασιών
2. Την ενδυνάμωση ενός κοινού συνόλου ικανοτήτων και γλωσσών, επιτρέποντας την μείωση του συνολικού κόστους μέσω της φορητότητας της γνώσης, σε αντιπαραβολή με τη χρήση πολύπλοκων ιδιόκτητων τεχνολογιών και την παροχή και ενδυνάμωση των βέλτιστων πρακτικών, των υποδειγμάτων, της εμπειρίας και της εκπαίδευσης από πληθώρα κατασκευαστών.
3. Την υλοποίηση αφηρημένης επιχειρησιακή λογική, δηλαδή τη σχεδίαση εφαρμογών και επιχειρησιακών υπηρεσιών χωρίς να απαιτείται γνώση των λεπτομερειών υλοποίησης των διαδικασιών, που είναι επαναχρησιμοποιούμενες.
4. Την ένταξη με φυσικό τρόπο εντός του Web Services Stack. Με την WS-BPEL μία επιχειρησιακή διαδικασία αλληλεπιδρά με άλλες διαδικασίες μέσω Web Services και εξωτερικεύεται ως Web Service, επιτρέποντας σε μία BPEL διαδικασία να ενισχύσει (leverage) την διαλειτουργικότητα που παρέχεται από τα WSDL, SOAP και Web-Addressing (χαμηλότερα επίπεδα του Web Services Stack).
5. Την αποκλειστική χρήση της γλώσσας XML. Οι επιχειρησιακές διαδικασίες που εκφράζονται σε XML μπορούν να είναι καταληπτές από ανθρώπους και μπορούν να χρησιμοποιηθούν σε οποιαδήποτε λειτουργία επεξεργάζεται XML, εντός του XML stack.
6. Τη χρήση και επέκταση του προτύπου WSDL 1.1 με στόχο την παροχή και κατανάλωση Web Services με αφηρημένο τρόπο
7. Τη χρήση του XML Schema 1.0 ορισμού τύπων (type definitions) για το μοντέλο δεδομένων.

8. Τη φορητότητα μεταξύ πλατφορμών και κατασκευαστών, που μειώνει την εξάρτηση από συγκεκριμένους κατασκευαστές και βοηθάει τη μετεγκατάσταση (migration) από μία πλατφόρμα ενός κατασκευαστή σε μία άλλη.

2.4 Ιστορική εξέλιξη WS-BPEL και η σχέση της με άλλες γλώσσες

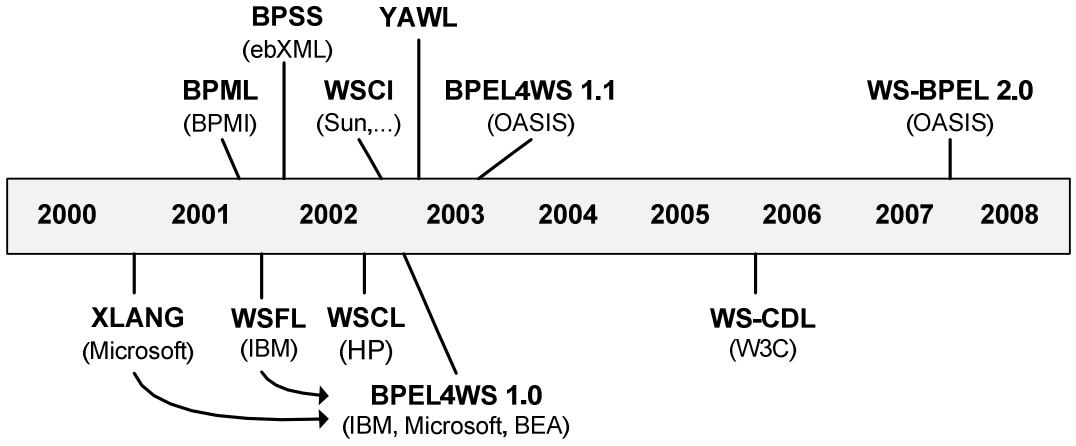
Η BPEL δεν είναι η μοναδική γλώσσα για την περιγραφή και την εκτέλεση επιχειρησιακών διαδικασιών. Οι σημαντικότερες γλώσσες ενορχήστρωσης που υπάρχουν είναι:

- XLANG και η XLANG/S της Microsoft
- WSFL (Web Services Flow Language), της IBM
- BPML (Business Process Modeling Language), του Business Process Management Initiative
- BPSS (Business Process Specification Schema), τμήμα του ebXML framework
- YAWL (Yet Another Workflow Language), μία γλώσσα ανοικτού κώδικα για επιχειρησιακές ροές

Οι πλέον σημαντικές γλώσσες χορογραφίας είναι:

- WSCL (Web Services Conversation Language) της HP, και έχει υποβληθεί για προτυποποίηση στο W3C
- WSCI (Web Services Choreography Interface), που αναπτύχθηκε από τις Sun, SAP, BEA, και Intalio και έχει υποβληθεί στο W3C και είναι σε κατάσταση Candidate Recommendation
- WS-CDL (Web Services Choreography Description Language), που έχει υποβληθεί W3C και είναι σε κατάσταση Candidate Recommendation

Παρακάτω φαίνεται η χρονική εξέλιξη της BPEL μέχρι την προτυποποίηση της [5 σελ 45]:

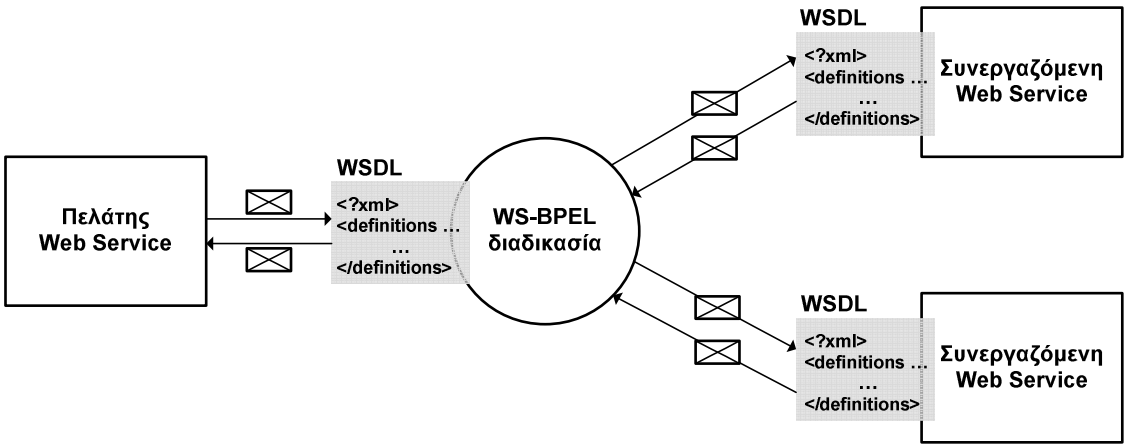


Εικόνα 10 Εξέλιξη της WS-BPEL

2.5 Αναλυτική παρουσίαση του προτύπου WS-BPEL

Η WS-BPEL είναι μία γλώσσα ενορχήστρωσης. Μας επιτρέπει να ορίσουμε μία επιχειρησιακή διαδικασία με τη χρήση Web Services, περιγράφοντας την ανταλλαγή μηνυμάτων μεταξύ των υπηρεσιών που συνεργάζονται, στο πλαίσιο εκτέλεσης της συγκεκριμένης διαδικασίας που υλοποιείται.

Στη πράξη η υλοποιούμενη διαδικασία οργανώνει ή αλλιώς “ενορχηστρώνει” την κλήση των Web Services με τις οποίες συνεργάζεται και την ανταλλαγή μηνυμάτων μεταξύ τους. Η WS-BPEL διαδικασία είναι και αυτή μία Web Service που περιγράφεται από ένα κείμενο WSDL, ώστε να μπορεί να καταναλωθεί ως τέτοια. Αυτό φαίνεται στο επόμενο σχήμα [8 σελ. 164]:



Εικόνα 11 Μία υπηρεσία WS-BPEL

Οι διαδικασίες WS-BPEL απαιτούν μία μηχανή για την εκτέλεσή τους. Η μηχανή αυτή διαβάζει την περιγραφή της εκτελέσιμης διαδικασίας, και ενδεχομένως σχετικά WSDL και αρχεία και αναμένει ένα μήνυμα κλήσης (request message) από τον καταναλωτή της διαδικασίας. Όταν λάβει το μήνυμα δημιουργεί μία παρουσία της διαδικασίας, την εκτελεί αλληλεπιδρώντας με τις συνεργαζόμενες Web Services, σύμφωνα με την WSDL περιγραφή της.

Κάποια από τα βασικά χαρακτηριστικά της WS-BPEL 2.0 που θα παρουσιαστούν παρακάτω είναι τα εξής:

- Επιχειρησιακοί συνεργάτες
- Μεταβλητές
- Δραστηριότητες
- Εμβέλειες
- Αντιστάθμιση
- Συσχέτιση και σύνολα συσχετίσεων
- Διαχείριση σφαλμάτων
- Διαχείριση συμβάντων

2.5.1 Η δομή μίας WS-BPEL διαδικασίας

Μία WS-BPEL διαδικασία αλληλεπιδρά με κάποιες συνεργαζόμενες Web Services που περιγράφονται με partnerLinks. Ανταλλάσει μηνύματα με αυτές που κρατούνται σε μεταβλητές (variables). Επίσης υποστηρίζει τη δυνατότητα διαχείρισης σφαλμάτων με τη βοήθεια των σχετικών χειριστών (faultHandlers).

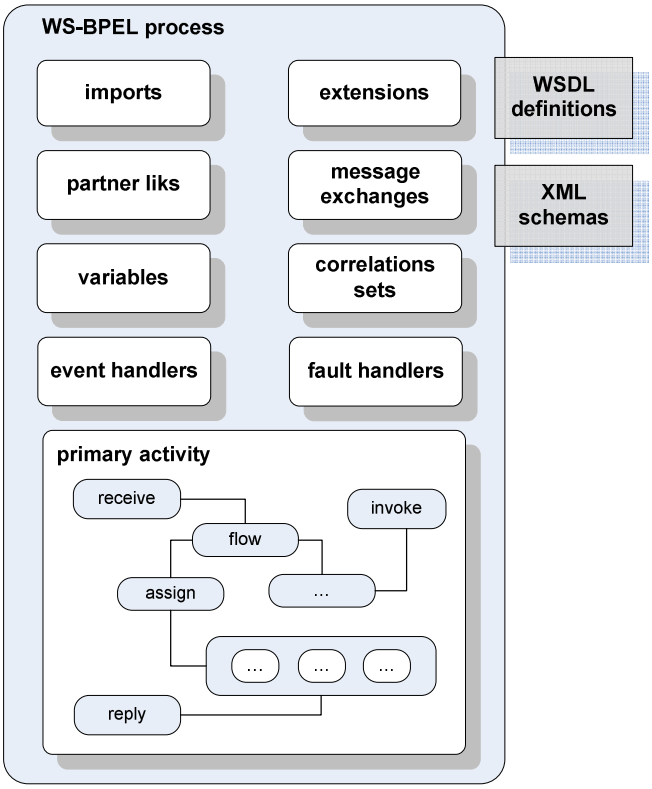
Το βασικό συστατικό στοιχείο σε μία WS-BPEL διαδικασία που αποτυπώνει την επιχειρησιακή λογική είναι οι δραστηριότητες (activities). Υπάρχουν δύο βασικοί τύποι δραστηριοτήτων, οι βασικές και οι δομημένες. Οι βασικές δραστηριότητες εκτελούν μόνο συγκεκριμένες λειτουργίες όπως για παράδειγμα παραλαμβάνουν ένα μήνυμα από μία συνεργαζόμενη Web Service, ή χειρίζονται δεδομένα και δεν περιέχουν επιπλέον λογική ή άλλες δραστηριότητες. Οι δομημένες δραστηριότητες μπορούν να περιέχουν άλλες δραστηριότητες και να περιλαμβάνουν επιχειρησιακή λογική μεταξύ τους.

Η βασική δομή ενός αρχείου στο οποίο ορίζεται μία WS-BPEL διαδικασία και περιλαμβάνει τα παραπάνω συστατικά μέρη, είναι η εξής:

```
<process …>
  <partnerLinks>
     …
  </partnersLinks>
  <variables>
     …
```

```
    </variables>
    <faultHandlers>
      …
    </faultHandlers>
    <sequence>
     <receive …/>
     <invoke …/>
     <reply…>
       …
    </sequence>
  </process>
```

Τα συστατικά μέρη [9] μίας διαδικασίας WS-BPEL φαίνονται στην επόμενη εικόνα:



Εικόνα 12 Η δομή μίας WS-BPEL διαδικασίας

Οι πλέον σημαντικές σημαντικές WS-BPEL constructs είναι οι παρακάτω:

WS-BPEL construct	Περιγραφή
<process>	Είναι το ριζικό (root) στοιχείο (element) του WS-BPEL ορισμού. Χρησιμοποιεί ιδιότητες (attributes) για να δηλώσει τούς σχετικούς με τη διαδικασία χώρους ονομάτων (namespaces)
<partnerLink>	Περιέχει ένα σύνολο partnerLinks. Κάθε ένα από αυτά περιγράφει την σχέση μεταξύ της διαδικασίας και των συνεργαζόμενων Web Services
<variables>	Περιέχει ένα σύνολο των μεταβλητών που χρησιμοποιούνται στη διαδικασία
<faultHandlers>	Περιέχει τους χειριστές σφαλμάτων που περιγράφουν τις αντιδράσεις σε τυχόν σφάλματα που μπορεί προκύψουν κατά την εκτέλεση της διαδικασίας

<sequence>	Περιέχει μία ή περισσότερες δραστηριότητες (activities) που εκτελούνται διαδοχικά
<flow>	Επιτρέπει την παράλληλη εκτέλεση δραστηριοτήτων
<receive>	Παραλαμβάνει ένα μήνυμα από την Web Service που κάλεσε τη WS-BPEL διαδικασία
<invoke>	Καλεί μία συνεργαζόμενη Web Service
<reply>	Αποστέλλει ένα μήνυμα απάντησης στην Web Service που κάλεσε τη WS-BPEL διαδικασία
<assign>	Χρησιμοποιείται για την απόδοση τιμής σε μία μεταβλητή
<if>	Η υπό συνθήκες εκτέλεση ενός τμήματος της διαδικασίας
<throw>	Ρητή δημιουργία ενός επώνυμου (named) σφάλματος εντός της διαδικασίας.

Πίνακας 2 Βασικές WS-BPEL constructs

Παρακάτω δίνεται με περισσότερες λεπτομέρειες η βασική δομή της γλώσσας, όπως αυτή παρατίθεται στο πρότυπο [10]:

```
<process name=”NCName” targetNamespace =”anyURI”
  queryLanguage =”anyURI” ?
  expressionLanguage =”anyURI” ?
  suppressJoinFailure=”yes|no” ?
  exitOnStandardFault=”yes|no” ?
  xmlns =”http://docs.oasis-open.org/wsbpel/2.0/process/executable” >

      <extensions >?
      <extension  namespace=”anyURI” mustUnderstand=”yes|no” />+
      </extensions >
  <import namespace=”anyURI” ?
  location =”anyURI” ?
  importType =”anyURI” />*

      <partnerLinks>?
      <!-- Note: At least one role must be specified. -->
      <partnerLink name=”NCName”
  partnerLinkType =”QName”
  myRole=”NCName” ?
  partnerRole=”NCName” ?
  initializePartnerRole=”yes|no” ?>+
  </partnerLink>
  </partnerLinks>

  <messageExchanges>?
  <messageExchange name=”NCName” />+
  </messageExchanges>

  <variables>?
  <variable name=”BPELVariableName”
    messageType=”QName”?
    type=”QName”?
    element=”QName”?>+
    from-spec?
```

```
  </variable>
  </variables>

      <correlationSets >?
      <correlationSet  name=”NCName” properties =”QName-list”  />+
      </correlationSets >
      <faultHandlers >?
      <!-- Note: There must be at least one faultHandler -->
      <catch faultName=”QName”?
      faultVariable =”BPELVariableName” ?
      ( faultMessageType=”QName” |  faultElement=”QName” )? >*
      activity
  </catch >
  <catchAll >?
      activity
  </catchAll >
  </faultHandlers >

      <eventHandlers >?
      <!-- Note: There must be at least one onEvent or onAlarm. -->
      <onEvent  partnerLink=”NCName”
  portType =”QName”?
  operation=”NCName”
  ( messageType=”QName”  | element=”QName”  )?
  variable =”BPELVariableName” ?
  messageExchange =”NCName” ?>*
  <correlations>?
      <correlation set =”NCName” initiate =”yes|join|no” ? />+
      </correlations>
      <fromParts>?
      <fromPart part=”NCName” toVariable =”BPELVariableName” />+
  </fromParts>
  <scope ... >... </scope >
  </onEvent>
  <onAlarm>*
    <!-- Note: There must be at least one expression. -->
              (
      <for expressionLanguage =”anyURI” ?>duration-expr </for >
              |
      <until expressionLanguage =”anyURI” ?>deadline-expr </until >
              )?
      <repeatEvery expressionLanguage =”anyURI” ?>
      duration-expr
    </repeatEvery>?
    <scope ... >... </scope >
  </onAlarm>
  </eventHandlers >
  activity
</process>
```

2.5.2 Η BPEL διαδικασία

Μία WS-BPEL διαδικασία αποτελεί ένας περιέκτης (container) μέσα στον οποίο δηλώνονται οι σχέσεις που έχει η διαδικασία με τις εξωτερικές συνεργαζόμενες Web Services, μπορεί να περιέχονται δηλώσεις σχετικές με την επεξεργασία δεδομένων, και κυρίως οι δραστηριότητες (activities) που εκτελούνται στο πλαίσιο της διαδικασίας.

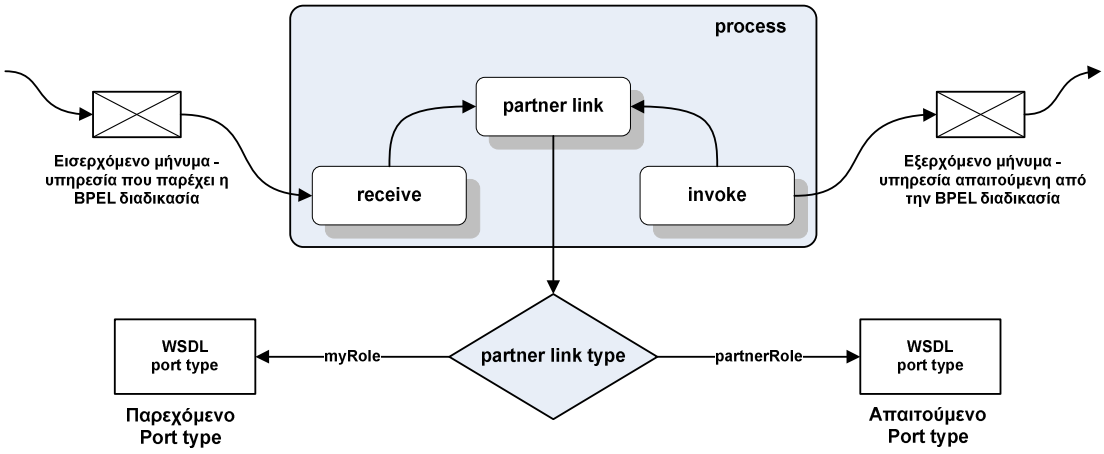
```
<process  name="exampleProcess"
targetNamespace ="http://exampleNamespace/"
xmlns ="http://docs.oasis-open.org/wsbpel/2.0/process/executable"/>
```

2.5.3 Οι επιχειρησιακοί συνεργάτες

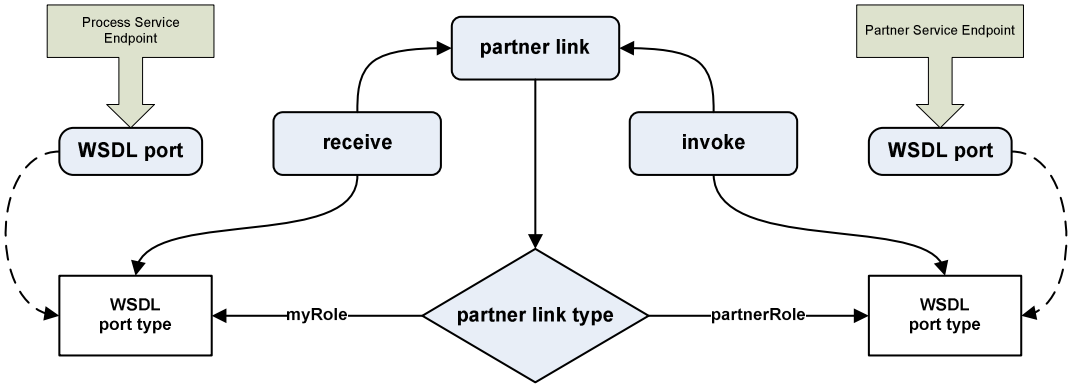
Η WS-BPEL συγκεντρώνει Web Services και ορίζει την επιχειρησιακή λογική μεταξύ τους ή αλλιώς ενορχηστρώνει τις αλληλεπιδράσεις μεταξύ τους. Οι αλληλεπιδράσεις αυτές μπορούν να θεωρηθούν ως επικοινωνία με επιχειρησιακούς συνεργάτες και περιγράφονται με τη βοήθεια των partnerLinks. Η επικοινωνία μιας διαδικασίας με του επιχειρησιακούς συνεργάτες είναι σε ομότιμο επίπεδο και μπορεί να είναι μονόδρομη ή αμφίδρομη. Τα partnerLinks περιγράφουν τα WSDL ports που προσφέρει και απαιτεί η διαδικασία στον επιχειρησιακό συνεργάτη που αντιστοιχεί στο συγκεκριμένο partnerLink. Κάθε partnerLink χαρακτηρίζεται από ένα partnerLinkType και ένα role.

```
<partnerLinks>
      <partnerLink name="clientLink"
      partnerLinkType ="wsdl: clientPLT" myRole="Client" />
   </partnerLinks>
```

Στις δύο επόμενες εικόνες παρουσιάζονται γραφικά οι σχέσεις μεταξύ <partnerLinks>, <partnerLinkTypes>, <ports>, <portTypes>, δραστηριοτήτων <receive> και <invoke> και μηνυμάτων που ανταλλάσσονται με τις καλούσες και τις συνεργαζόμενες υπηρεσίες [9].

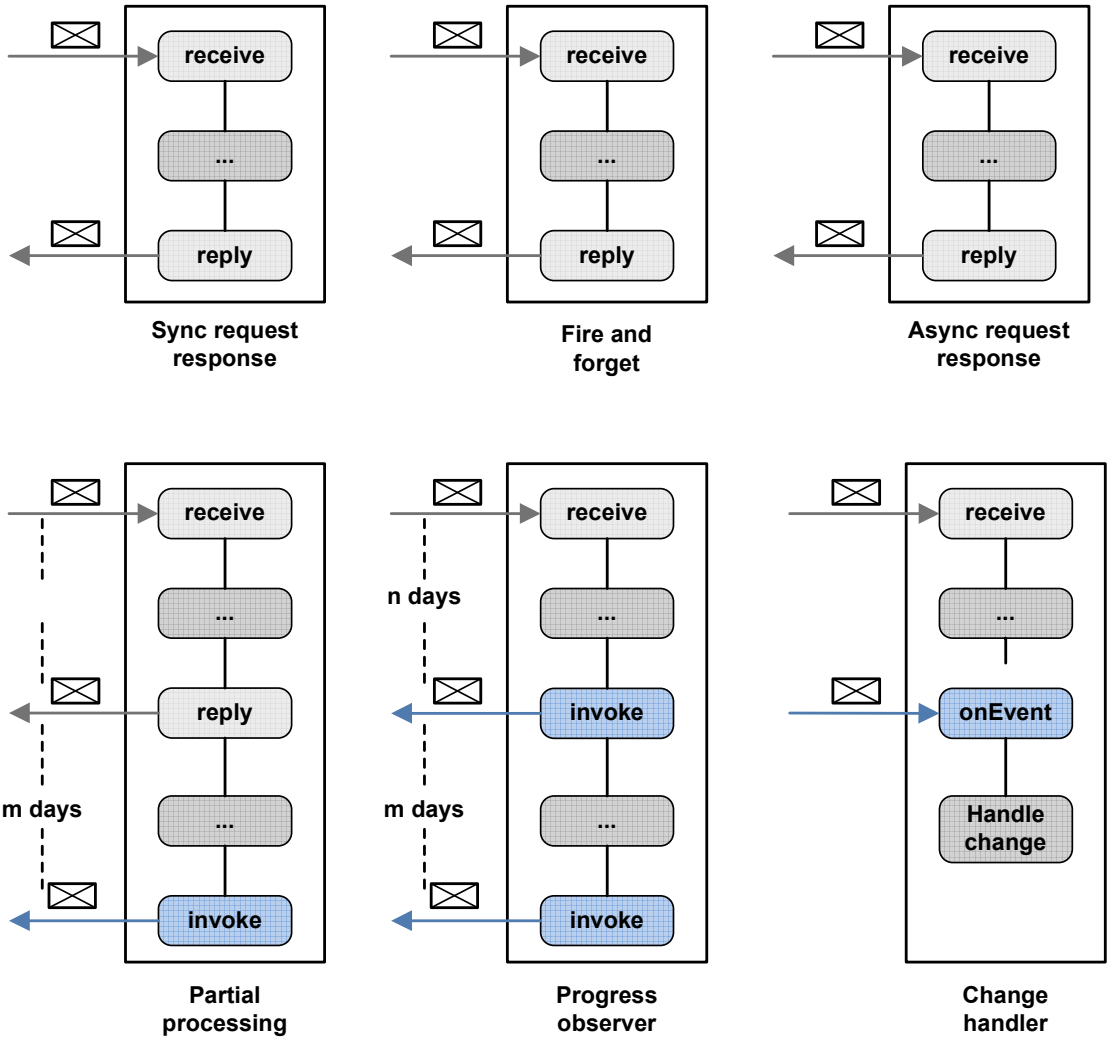


Εικόνα 13 Επιχειρησιακοί συνεργάτες



Εικόνα 14 Επιχειρησιακοί συνεργάτες

Οι αλληλεπιδράσεις μεταξύ επιχειρησιακών διαδικασιών και συνεργαζόμενων υπηρεσιών μπορεί να βασίζονται σε σύγχρονη ή ασύγχρονη, μονόδρομη ή αμφίδρομη επικοινωνία, να εκτελούνται σε σύντομο ή σε παρατεταμένο χρονικό διάστημα και να είναι αυτόνομες (να μην απαιτείται άλλη ενδιάμεση επικοινωνία μεταξύ διαδικασίας και συνεργαζόμενων υπηρεσιών) ή όχι. Στην επόμενη εικόνα φαίνονται οι βασικές κατηγορίες αλληλεπιδράσεων [11].



Εικόνα 15 Παραδείγματα αλληλεπιδράσεων

2.5.4 Διαχείριση δεδομένων και μεταβλητές

Οι επιχειρησιακές διαδικασίες περιγράφουν stateful αλληλεπιδράσεις, που έχουν δηλαδή επίγνωση της κατάστασης τους και περιλαμβάνουν την ανταλλαγή μηνυμάτων μεταξύ των επιχειρησιακών συνεργατών1. Η επίγνωση της κατάστασης μπορεί να απαιτεί εκτός από τα ανταλλασσόμενα μηνύματα και δεδομένα που προκύπτουν από την εκτέλεση της επιχειρησιακής λογικής. Οι μεταβλητές (variables) είναι το μέσο με το οποίο κρατούνται τα μηνύματα και τα δεδομένα που παράγονται κατά την εκτέλεση της διαδικασίας και δεν ανταλλάσσονται με τους συνεργάτες.

Οι μεταβλητές πρέπει να είναι WSDL message types, ή XML schema simple types, ή XML schema complex types, ή XML schema elements.

```
<variables>
   <variable name="var1" messageType="myNS:myWSDLMessageDataType" />
   <variable name="var1" element="myNS:myXMLElement"  />
   <variable name="var2" type="xsd:string"  />
   <variable name="var2" type="myNS:myComplexType" />
</variables>
```

Στη WS-BPEL δυνατότητες διαχείρισης και αναζήτησης μεταβλητών παρέχει η γλώσσα XPath 1.0.

2.5.5 Δραστηριότητες

Οι δραστηριότητες που υποστηρίζει η γλώσσα είναι οι εξής:

- <receive>
- <reply>
- <invoke>
- <assign>
- <throw>
- <exit>
- <wait>
- <empty>
- <sequence>
- <if>
- <while>
- <repeatUntil>
- <forEach>
- <pick>
- <flow>
- <scope>
- <compensate>
- <compensateScope>
- <rethrow>
- <validate>
- <extensionActivity>

[1] Το πρότυπο χρησιμοποιεί ιδιαίτερα την έννοια της κατάστασης μίας διαδικασίας και σε αυτή την έννοια εντάσσει τα δεδομένα, όχι ως αυτόνομη έννοια.

Παρακάτω παρουσιάζονται συνοπτικά οι συγκεκριμένες δραστηριότητες. Στις πλέον σημαντικές από αυτές παραθέτεται η βασική σύνταξή τους.

<receive>
Αυτή η δραστηριότητα επιτρέπει στη διαδικασία να αναμένει την άφιξη ενός συγκεκριμένου μηνύματος και ολοκληρώνεται όταν ληφθεί το μήνυμα.

<reply>
Η δραστηριότητα <reply> επιτρέπει στη διαδικασία ένα μήνυμα σε απάντηση ενός μηνύματος που έλαβε.

<invoke>
Αυτή η δραστηριότητα χρησιμοποιείται από τη διαδικασία προκειμένου να καλέσει μία μονόδρομη η αμφίδρομη λειτουργία σε ένα <portType> που παρέχει μία συνεργαζόμενη υπηρεσία. Σε περίπτωση αμφίδρομης λειτουργίας η δραστηριότητα invoke ολοκληρώνεται όταν ληφθεί η απάντηση.

<assign>
Όταν απαιτείται να επικαιροποιηθούν οι τιμές κάποιον μεταβλητών χρησιμοποιείται η δραστηριότητα <assign>.

<throw>
Η δραστηριότητα <throw> εγείρει ένα σφάλμα εντός της διαδικασίας.

<exit>
Η δραστηριότητα <exit> επιφέρει τον άμεσο τερματισμό της εκτέλεσης της δραστηριότητας εντός της οποίας βρίσκεται.

<wait>
Όταν απαιτείται η διαδικασία να τεθεί σε αναμονή για συγκεκριμένη χρονική περίοδο ή μέχρι ένα συγκεκριμένο σημείο χρησιμοποιείται αυτή η δραστηριότητα.

<empty>
Η δραστηριότητα <empty> δεν εκτελεί καμία λειτουργία (no-operation) και μπορεί να χρησιμοποιηθεί για να συγχρονιστούν δραστηριότητες που εκτελούνται ταυτόχρονα.

<sequence>
Η δραστηριότητα <sequence> ορίζει ένα σύνολο δραστηριοτήτων που θα εκτελεστούν στη σειρά.

<if>
Η δραστηριότητα <if> χρησιμοποιείται στη περίπτωση της εκτέλεσης αποκλειστικά μίας δραστηριότητας από ένα σύνολο επιλογών.

<while>
Η δραστηριότητα <while> επιτρέπει την εκτέλεση μίας δραστηριότητας μόνο στη περίπτωση που η συνθήκη ελέγχου είναι αληθής.

<repeatUntil>
Στη περίπτωση που είναι επιθυμητή η εκτέλεση μίας δραστηριότητας μέχρι η συνθήκη ελέγχου να γίνει αληθής χρησιμοποιείται η δραστηριότητα <repeatUntil>.

<forEach>
Η δραστηριότητα <forEach> εκτελεί την/τις δραστηριότητες παιδιά για συγκεκριμένο αριθμό επαναλήψεων. Παρέχει τη δυνατότητα παράλληλης εκτέλεσης των δραστηριοτήτων (δημιουργείται δηλαδή δυναμικά μία δραστηριότητα flow) και επίσης παρέχει τη δυνατότητα προσθήκης συνθήκης ελέγχου που αποτρέπει την εκτέλεση του βρόγχου εφόσον είναι αληθής.

<pick>
Στη περίπτωση που η εκτέλεση μίας ή περισσότερων δραστηριοτήτων προϋποθέτει την άφιξη ενός μηνύματος ή την παρέλευση ενός χρονικού διαστήματος (timeout) χωρίς το μήνυμα να παραληφθεί, χρησιμοποιείται η δραστηριότητα <pick>.

<flow>
Η παράλληλη εκτέλεση δραστηριοτήτων είναι εφικτή με τη δραστηριότητα <flow>.

<scope>
Η δραστηριότητα <scope> περιορίζει το εύρος των <partnerLinks>, <messageExchange>, <variables>, <correlationSets>, <faultHandlers>, <compenstationHandler>, <terminationHandler> και <eventHandler> εντός των εμφωλιασμένων δραστηριοτήτων.

<compensate>
Η δραστηριότητα <compensate> χρησιμοποιείται για να ξεκινήσει μία διαδικασία αντιστάθμισης (compensation) εντός των εσωτερικών εμβελειών (inner scopes) που έχουν ολοκληρωθεί επιτυχώς. Πρέπει αν χρησιμοποιείται πάντα εντός fault handler, ενός άλλου compensation handler ή ενός termination handler.

<compensateScope>
Η δραστηριότητα <compensateScope> ξεκινάει μία διαδικασία αντιστάθμισης (compensation) εντός μίας εσωτερικής περιοχής εύρους (inner scope) που έχει ολοκληρωθεί επιτυχώς. Πρέπει αν χρησιμοποιείται πάντα εντός fault handler, ενός άλλου compensation handler ή ενός termination handler.

<rethrow>
Η δραστηριότητα <rethrow> διαδίδει ένα σφάλμα που αρχικά είχε συληφθεί από τον άμεσα περιβάλλοντα fault handler. Πρέπει να χρησιμοποιείται μόνο εντός ενός fault Handler.

<validate>
Η δραστηριότητα <validate> χρησιμοποιείται όταν απαιτείται να επιβεβαιωθούν οι τιμές μεταβλητών σε σύγκριση με τους σχετικούς XML και WSDL ορισμούς δεδομένων. Στη περίπτωση όπου η επιβεβαίωση αποτύχει τότε πρέπει να εγερθεί ένα σφάλμα.

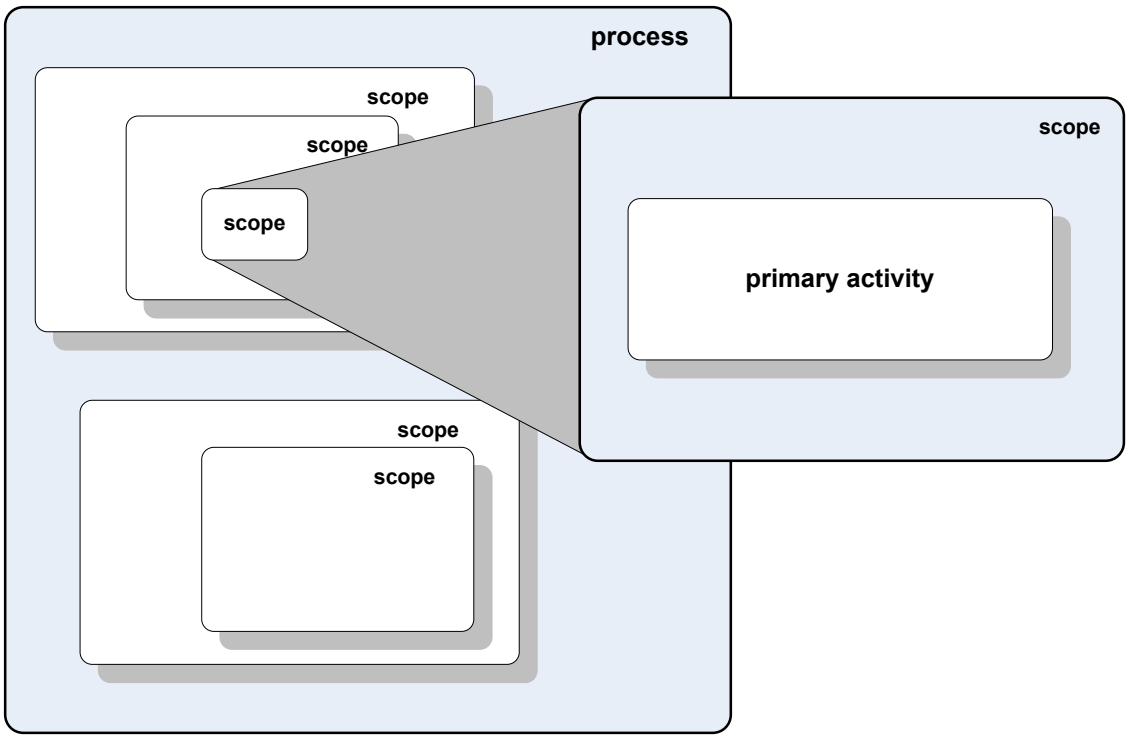
<extensionActivity>
Το στοιχείο <extensionActivity> χρησιμοποιείται για την εισαγωγή νέων τύπων δραστηριοτήτων. Τα περιεχόμενα του στοιχείου πρέπει να είναι ένα στοιχείο που περιέχει standard WS-BPEL ιδιότητες και στοιχεία.

2.5.6 Εμβέλειες (Scopes)

Η WS-BPEL υποστηρίζει τη δόμηση μίας διαδικασίας που περιέχει εμφωλιασμένες εμβέλειες (nested scopes). Κάθε εμβέλεια μπορεί να περιέχει τους δικούς της ορισμούς συνεργατών, μεταβλητών, μηνυμάτων, συνόλων συσχέτισης και χειριστών (handlers) οι οποίοι ορίζονται με αυτή τη σειρά. Μία εμβέλεια μπορεί να χρησιμοποιηθεί σαν μία κανονική δραστηριότητα.

Η εμβέλεια ολοκληρώνει την εργασία της είτε επιτυχώς είτε ανεπιτυχώς με τρεις διαφορετικούς τρόπους:

- Ομαλή ολοκλήρωση, όταν η κύρια δραστηριότητα ολοκληρωθεί χωρίς εμφάνιση σφάλματος και κανένα εισερχόμενο μήνυμα δεν ανιχνευθεί, τότε απενεργοποιούνται οι χειριστές συμβάντων, ενεργοποιείται ο χειριστής αντιστάθμισης και η εμβέλεια τερματίζεται επιτυχώς.
- Εσωτερικό σφάλμα, όταν ένα σφάλμα εγερθεί εντός αυτής, όλες οι δραστηριότητες και οι χειριστές τερματίζονται και ο σχετικός χειριστής σφάλματος εκτελείται. Η εμβέλεια τερματίζεται ανεπιτυχώς.
- Εξωτερικός τερματισμός, όταν ληφθεί ένα σήμα τερματισμού, είτε λόγω εξωτερικού σφάλματος είτε λόγω επαλήθευσης μίας συνθήκης ολοκλήρωσης, όλες οι δραστηριότητες και οι χειριστές συμβάντων τερματίζονται. Η εμβέλεια τερματίζεται ανεπιτυχώς.



Εικόνα 16 Εμβέλειες [9]

2.5.7 Διαχείριση σφαλμάτων

Σε μία διαδικασία μπορεί να εμφανιστούν σφάλματα λόγω προβλημάτων στην επικοινωνία με τις συνεργαζόμενες Web Services (για παράδειγμα αστοχία στο δίκτυο), στα μηνύματα που αποστέλλουν οι Web Services, λόγω εσωτερικών σφαλμάτων σε αυτές, ή και εξ αιτίας αστοχιών στην υποδομές που υλοποιούν τη διαδικασία (τόσο υλικού όσο και λογισμικού). Διακρίνονται οι εξής περιπτώσεις σφαλμάτων:

- Μία σύγχρονα καλούμενη Web Service επιστρέφει ένα WSDL σφάλμα, που είχε σαν αποτέλεσμα ένα BPEL σφάλμα
- Η BPEL διαδικασία ρητά εγείρει ένα σφάλμα
- Ένα σφάλμα εγείρεται αυτόματα
- Ο εξυπηρετητής BPEL μπορεί να παρουσιάσει σφάλματα (για παράδειγμα στο περιβάλλον εκτέλεσης, ή στο υλικό). Το πρότυπο ορίζει αρκετά τυποποιημένα σφάλματα.

Ο χειρισμός των σφαλμάτων γίνεται με τους σχετικούς χειριστές (faultHandlers). Η αντιστάθμιση είναι μία επέκταση της κλασικής έννοιας της διαχείρισης σφαλμάτων, ειδικότερα στο πλαίσιο της εκτέλεσης μακροχρόνιων συναλλαγών (Long-Running Transactions).

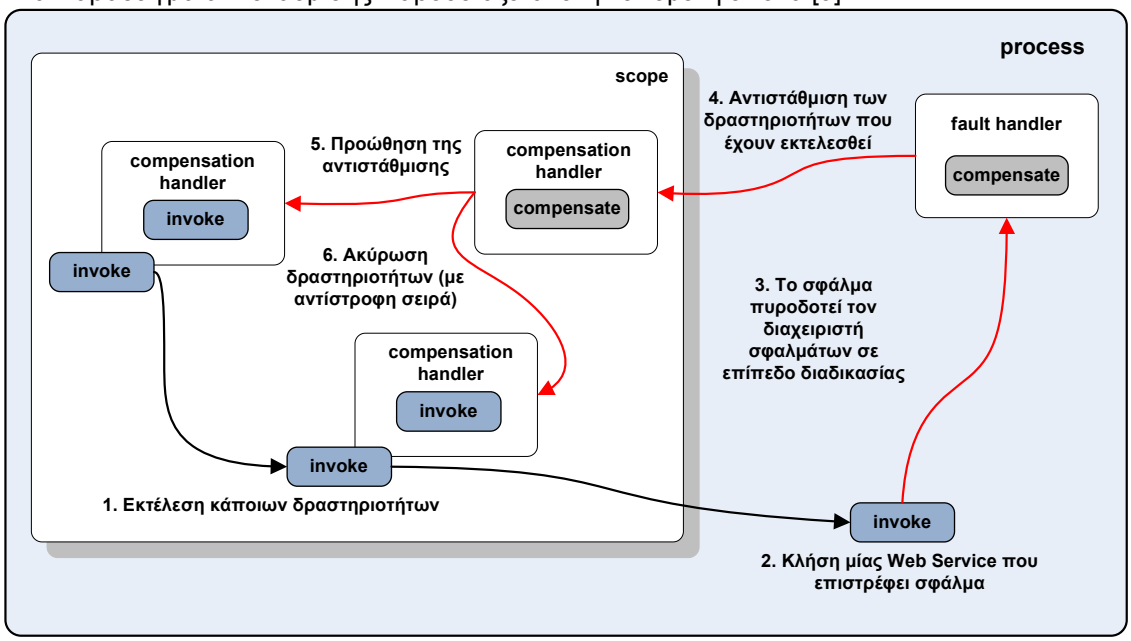
2.5.8 Αντιστάθμιση (compensation)

Οι επιχειρησιακές διαδικασίες μπορεί να απαιτούν για την ολοκλήρωση τους μεγάλα χρονικά διαστήματα, ειδικότερα στις περιπτώσεις όπου απαιτείται ανθρώπινη δραστηριότητα για την ολοκλήρωση, όπως για παράδειγμα ή έγκριση μίας ενέργειας. Σε τέτοιες περιπτώσεις δεν είναι εφικτή η ολοκλήρωση της διαδικασίας ως μίας ατομικής συναλλαγής. Τυχόν συναλλαγές που έχουν πραγματοποιηθεί πριν την ολοκλήρωση της διαδικασίας δημιουργούν μόνιμες επιπτώσεις για τις οποίες θα πρέπει να υπάρχει δυνατότητα αντιστάθμισης, δηλαδή της επαναφοράς στην προ της πραγματοποίησης τους κατάστασης, με άλλα λόγια της ακύρωσης των συνεπειών που αυτές είχαν στην κατάσταση της διαδικασίας.

Ο μηχανισμός για την αντιστάθμιση που παρέχει η WS-BPEL είναι η construct <compensationHandler> και οι δραστηριότητες <compensate> και <compansateScope>.

```
<scope name="scope1">
  <faultHandlers >
  <catchAll >
      <compensateScope target="scope2"/>
  </catchAll >
  </faultHandlers >
  <sequence >
      <scope name="scope2">
           <compensationHandler>
           <!—ακύρωση ενεργειών -->
           </compensationHandler>
               ...
      </scope >
           ...
      <!-- αν εδώ εγερθεί σφάλμα οι επιπτώσεις της scope2  πρέπει να
   ακυρωθούν -->
  </sequence >
</scope >
```

Ένα παράδειγμα αντιστάθμισης παρουσιάζεται στην επόμενη εικόνα [9]:



Εικόνα 17 Αντιστάθμιση

2.5.9 Διαχείριση συμβάντων

Σε κάθε εμβέλεια, περιλαμβανομένης και αυτής που αντιστοιχεί στη ριζική διαδικασία (στοιχείο process), μπορεί να ορίζονται χειριστές συμβάντων, που χρησιμοποιούνται για την επεξεργασία μηνυμάτων αιτήσεων που λαμβάνονται από Web Services παράλληλα με την εκτέλεση της κύριας διαδικασίας της συγκεκριμένης εμβέλειας.

Δύο είδη συμβάντων υποστηρίζονται: συμβάντα μηνυμάτων (message events) και συμβάντα συναγερμών (alarm events). Για το χειρισμό συμβάντων μηνυμάτων χρησιμοποιείται το στοιχείο onEvent, που παρέχει παρόμοια λειτουργικότητα με τη δραστηριότητα receive. Το στοιχείο onAlarm χρησιμοποιείται για το χειρισμό συμβάντων χρόνου και παρέχει παρόμοια λειτουργικότητα με τη δραστηριότητα <pick>.

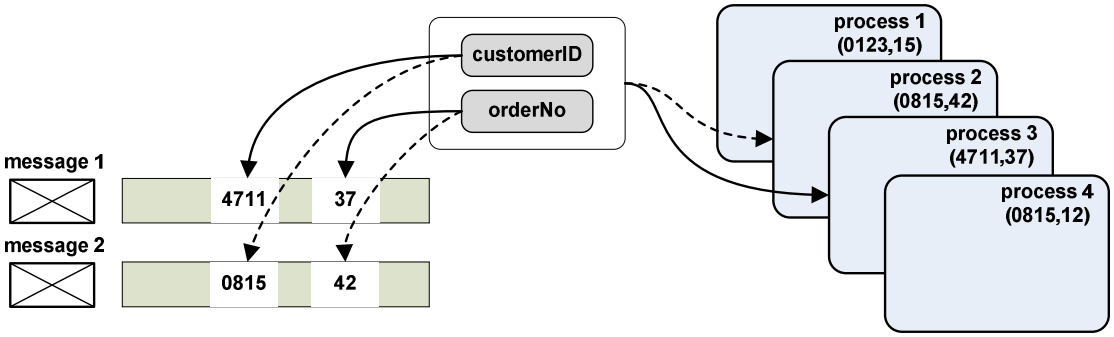
Η δραστηριότητα παιδί εντός ενός χειριστή συμβάντων πρέπει να είναι η scope.

```
<process  name="eventExampleProcess"  ... >
        ...
  <eventHandlers >
  <onEvent  partnerLink="eventExamplePL"
    operation="eventExampleOP1" ... >
    <scope >... </scope >
  </onEvent>
  <onEvent  partnerLink=" eventExamplePL "
    operation=" eventExampleOP2" ... >
    <scope >... </scope >
  </onEvent>
  </eventHandlers >
        ...
</process>
```

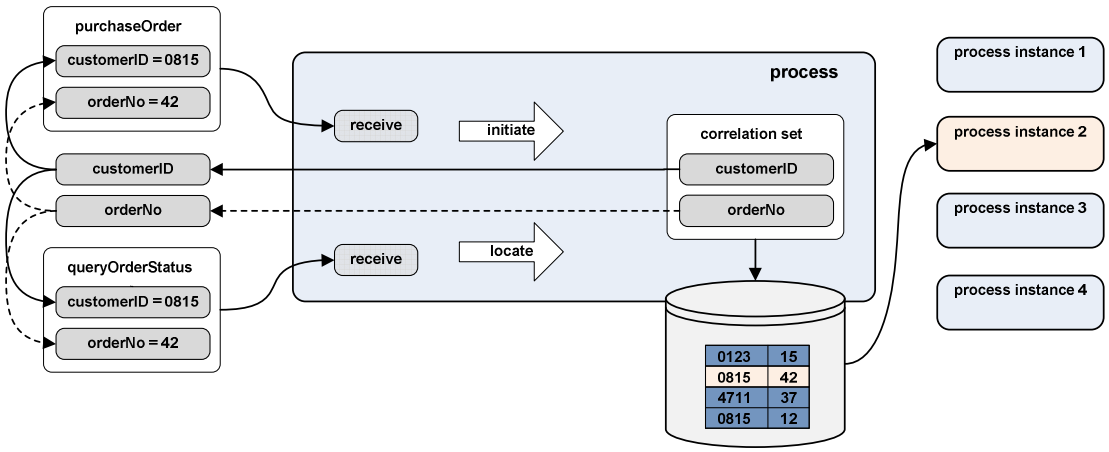
2.5.10 Συσχέτιση (correlation)

Όταν η καλούσα Web Service καλεί τη διαδικασία δημιουργείται μία παρουσία αυτής η οποία ζει κατά τη διάρκεια εκτέλεσης της επιχειρησιακής διαδικασίας. Υπάρχει η πιθανότητα να είναι ζωντανές ταυτόχρονα πολλές παρουσίες της ίδιας διαδικασίας. Σε αυτή τη περίπτωση, προκειμένου τα μηνύματα από την καλούσα Web Service να δρομολογούνται στην σωστή παρουσία της διαδικασίας, η WS-BPEL παρέχει το μηχανισμό της συσχέτισης και των συνόλων συσχετίσεων (correlation sets). Η συσχέτιση επιτυγχάνεται με τη χρήση επιχειρησιακών δεδομένων, που περιέχονται στα μηνύματα που ανταλλάσσονται μεταξύ των συνεργατών. Τα επιχειρησιακά δεδομένα σε πολλές περιπτώσεις περιέχουν πληροφορία που μπορεί να χρησιμοποιηθεί για να χαρακτηρίσει μοναδικά μία συγκεκριμένη παρουσία διαδικασίας, όπως για παράδειγμα ο κωδικός παραγγελίας για ένα πελάτη που έχει σε εκκρεμότητα παραπάνω από μία παραγγελίες, ή ο κωδικός πελάτη όταν ταυτόχρονα η διαδικασία υλοποίησης παραγγελιών εκτελεί παραγγελίες από διαφορετικούς πελάτες.

Προκειμένου να οριστούν τα δεδομένα που χρησιμοποιούνται για τη συσχέτιση, χρησιμοποιούνται οι ιδιότητες των μηνυμάτων (message properties). Με τη δήλωση property aliases (στο αρχείο WSDL) αντιστοιχείται μία ιδιότητα με συγκεκριμένο στοιχείο ή χαρακτηριστικό του μέρους (part) του συγκεκριμένου μηνύματος. Στη συνέχεια περισσότερες από μία ιδιότητες σχετίζονται σε ένα σύνολο συσχέτισης, που χρησιμοποιείται για να σηματοδοτεί μοναδικά μία παρουσία διαδικασίας κατά την ανταλλαγή μηνυμάτων, μεταξύ της διαδικασίας και των συνεργαζόμενων Web Services. Στις δύο επόμενες εικόνες παρουσιάζονται γραφικά η λειτουργία των συσχετίσεων [9]



Εικόνα 18 Συσχετίσεις



Εικόνα 19 Συσχετίσεις σε μακροχρόνιες διαδικασίες

Τα σύνολα συσχετίσεων μπορούν να χρησιμοποιηθούν στις δραστηριότητες <invoke>, <receive>, <reply>, <onMessage> σε δραστηριότητες <pick>, και <onEvent> σε χειριστές συμβάντων.

2.6 Η επικοινωνία με τις Web Services

Όπως αναφέρθηκε η WS-BPEL χρησιμοποιεί τη WSDL προκειμένου να περιγράψει την Web Service με την οποία εξωτερικεύει τη λειτουργικότητα της επιχειρησιακής διαδικασίας. Παρακάτω ακολουθεί μία συνοπτική παρουσίαση της δομής και των βασικών συστατικών ενός αρχείου WSDL.

2.6.1 Η βασική δομή του αρχείου WSDL

Η βασική δομή που έχει ένα αρχείο WSDL είναι η παρακάτω:

```
<definitions>

<types>
 <!-- ορισμοί μορφότυπων δεδομένων -->
</types>

<message>
 <!--  ορισμοί των δεδομένων που επικοινωνούνται -->
</message>

<portType>
 <!-- λειτουργίες -->
</portType>

<binding>
 <!—πρωτόκολλα και σχήμα κωδικοποίησης -->
</binding>

</definitions>
```

Το αρχείο μπορεί να περιέχει και άλλα στοιχεία, όπως επεκτάσεις και το στοιχείο <service> που ομαδοποιεί περισσότερες Web Services σε ένα αρχείο WSDL.

<types>
Το στοιχείο <types> ορίζει τους τύπους δεδομένων που χρησιμοποιούνται στη Web Service χρησιμοποιώντας XML Schema. Οι τύποι μπορεί να είναι εγγενείς (native), όπως αλφαριθμητικά, ακέραιοι και άλλα, ή μπορεί να είναι προσαρμοσμένοι (custom).

<message>
Το στοιχείο <message> ορίζει τα δεδομένα των μηνυμάτων μίας λειτουργίας. Κάθε μήνυμα μπορεί να αποτελείται από ένα ή περισσότερα στοιχεία <parts>.

<portType>
Το στοιχείο <portType> ορίζει την Web Service, τις λειτουργίες (operations) που μπορούν να εκτελεστούν και τα μηνύματα που χρησιμοποιούνται στη λειτουργία. Ουσιαστικά ορίζει ένα σημείο επικοινωνίας με την Web Service. Είναι το πιο σημαντικό στοιχείο.

Το στοιχείο <portType> περιέχει στοιχεία <operation>. Αυτά περιέχουν τα στοιχεία <input> και <output>, τα οποία αντιστοιχούν άμεσα σε στοιχεία <message>. Μπορεί επίσης να περιέχουν στοιχεία <fault> που ορίζουν τα σφάλματα που η λειτουργία μπορεί να εγείρει.

Οι παρακάτω τέσσερις τύποι λειτουργίας υποστηρίζονται:

Τύπος	Περιγραφή
One-way	Η λειτουργία μπορεί να λάβει μήνυμα αλλά δεν επιστρέφει απάντηση
Request-response	Η λειτουργία μπορεί να λάβει αίτηση και να επιστρέψει απάντηση
Solicit-responce	Η λειτουργία μπορεί να αποστείλει αίτηση και θα περιμένει για να λάβει απάντηση
Notification	Η λειτουργία μπορεί να αποστείλει αίτηση αλλά δεν περιμένει για να λάβει απάντηση

Πίνακας 3 Τύποι λειτουργιών

Ένα παράδειγμα μονόδρομης (one-way) λειτουργίας είναι το εξής:

```
<message name=”inputInfoMessage”>
  <part name=”infoType” type=”xs:string”/>
  <part name=”infoValue” type=”xs:string”/>
</message>
<portType name=”oneWayPT”>
  <operation name=”getInfo”>
    <input name=”newInfo” message=” inputMessage “/>
  </operation>
</portType >
```

Το επόμενο είναι ένα παράδειγμα αμφίδρομης (request-response) λειτουργίας:

```
<message name=” inputMessage “>
  <part name=”infoType” type=”xs:string”/>
  <part name=”infoValue” type=”xs:string”/>
</message>

<message name=” outputMessage “>
  <part name=”returnValue” type=”xs:string”/>
</message>

<portType name=”requestResonsepPT”>
  <operation name=”getInfo”>
    <input message=” inputMessage “/>
    <output message=” outputMessage “/>
  </operation>
</portType>
```

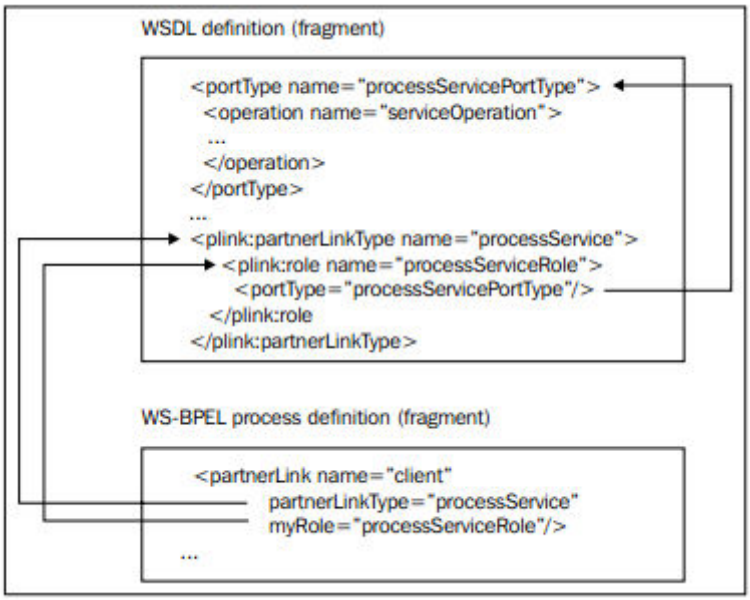
<binding>

Το στοιχείο <binding> ορίζει το σχήμα κωδικοποίησης των δεδομένων και τα πρωτόκολλα μεταφοράς (transport protocols) που χρησιμοποιούνται για την επικοινωνία σε κάθε <portType>. Ο πλέον συνηθισμένος συνδυασμός κωδικοποίησης και πρωτοκόλλου είναι SOAP πάνω από HTTP.

Το στοιχείο <binding> αντιστοιχίζεται σε ένα <portType>, και κάθε λειτουργία του τελευταίου ορίζεται και αυτή στο <binding>. Ένα παράδειγμα με SOAP πάνω από HTTP είναι:

```
<binding type=" requestResonsepPT " name="binding1">
   <soap:binding style="document"
   transport="http://schemas.xmlsoap.org/soap/http" />
   <operation>
    <soap:operation soapAction="http://example.com/getInfo"/>
    <input><soap:body use="literal"/></input>
    <output><soap:body use="literal"/></output>
   </operation>
</binding>
```

Η σχέση μεταξύ μίας περιγραφής WSDL και της WS-BPEL διαδικασίας φαίνεται στην επόμενη εικόνα [8]:



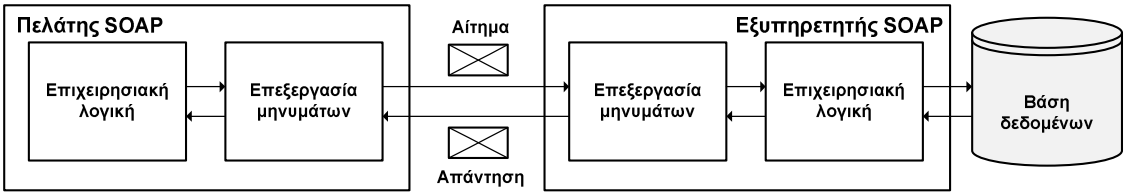
Εικόνα 20 Η σχέση μεταξύ WSDL και WS-BPEL διαδικασίας

2.6.2 Η δομή ενός μηνύματος SOAP

Η επικοινωνία μεταξύ του καταναλωτή και του παρόχου μίας SOAP Web Service γίνεται μέσω SOAP μηνυμάτων, που έχουν συγκεκριμένη δομή. Η βασική γνώση της δομής των μηνυμάτων, που παρουσιάζεται παρακάτω, είναι απαραίτητη για την αποσφαλμάτωση της επικοινωνίας παρόχου και καταναλωτή Web Services:

```
  <Envelope>
    <Header>...</Header>
    <Body>...</Body>
  </Envelope>
```

Η ανταλλαγή των μηνυμάτων παρουσιάζεται γραφικά στην παρακάτω εικόνα [12 σελ. 297]:



Εικόνα 21 Ανταλλαγή μηνυμάτων SOAP

<envelope>
Κάθε μήνυμα SOAP πρέπει να περιέχει ένα στοιχείο <envelope>, το οποίο σχετίζεται με μία περιοχή ονομάτων, που χαρακτηρίζει το μήνυμα ως SOAP 1.1.

<header>
Το συγκεκριμένο στοιχείο είναι προαιρετικό και χρησιμοποιείται για την επέκταση των μηνυμάτων με αρθρωτό τρόπο. Για παράδειγμα εδώ μπορεί να υλοποιηθεί λειτουργικότητα σχετική με τη ασφάλεια της Web Service.

<body>
Στο στοιχείο <body> περιέχει το σύνολο της πληροφορίας που λαμβάνει ο κάθε παραλήπτης. Η δομή της πληροφορίας εξαρτάται από τη δόμηση του μηνύματος στη σχετική WSDL περιγραφή της Web Service.

<fault>
Το στοιχείο <fault> χρησιμοποιείται στη περίπτωση όπου θα επιστραφεί ένα σφάλμα.

Παρακάτω παρατίθεται ένα παράδειγμα αίτησης SOAP:

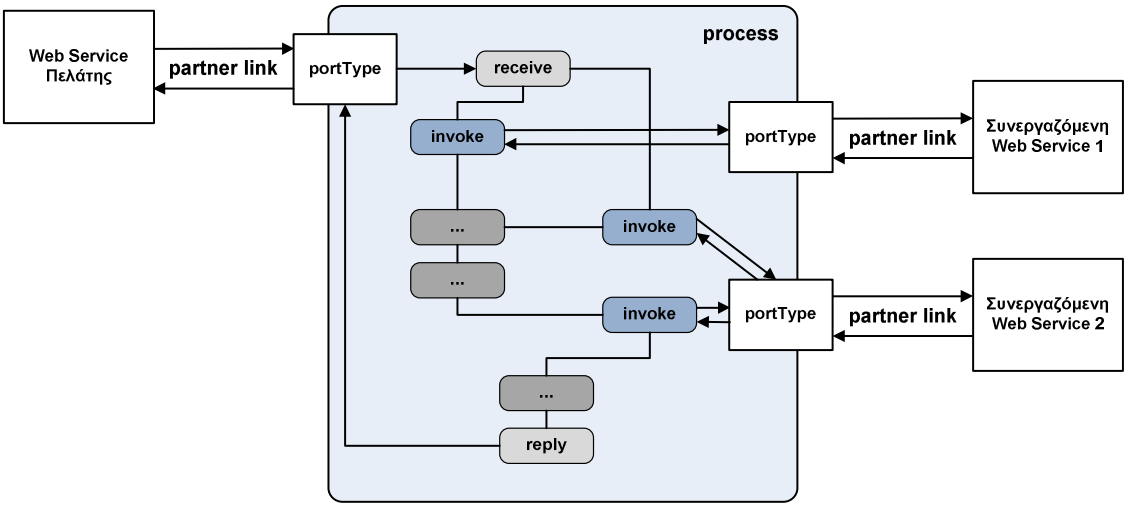
```
<?xml version=”1.0” encoding=”UTF-8”?>
<SOAP-ENV:Envelope xmlns:SOAP-
ENV=”http://schemas.xmlsoap.org/soap/envelope/”
  xmlns:ns1=”urn:ExampleAPI”>
  <SOAP-ENV:Body>
  <ns1:getInfo>
    <input1>alfa</input1>
    < input2>beta</input2>
  </ns1: getInfo >
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Και ένα παράδειγμα απάντησης στο προηγούμενο αίτημα θα μπορούσε να είναι:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<SOAP-ENV:Envelope xmlns:SOAP-
ENV=”http://schemas.xmlsoap.org/soap/envelope/”
xmlns:SOAP-ENC=”http://schemas.xmlsoap.org/soap/encoding/”>
  <SOAP-ENV:Body>
  <result>
    <SOAP-ENC:Struct>
    <id>1</id>
    <info>ALFA</info >
    </SOAP-ENC:Struct>
    <SOAP-ENC:Struct>
```

```
    <id>2</id>
    < info>BETA</info>
    </SOAP-ENC:Struct>
  </result>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Ένα σχηματοποιημένο παράδειγμα των παραπάνω παρουσιάζεται στην επόμενη εικόνα [5 σελ. 57]:



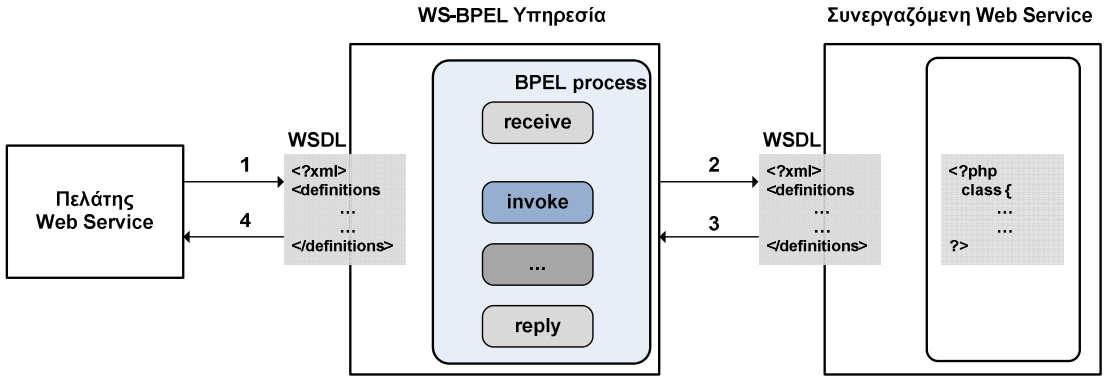
Εικόνα 22 Παράδειγμα επικοινωνίας BPEL διαδικασίας με Web Services

2.7 Ένα απλό παράδειγμα μίας WS-BPEL διαδικασίας

Ένα απλό παράδειγμα που δείχνει τις βασικές αλληλεπιδράσεις μίας WS-BPEL διαδικασίας με τις συνεργαζόμενες Web Services είναι το παρακάτω:

1. Βήμα 1: Η WS-BPEL διαδικασία λαμβάνει ένα μήνυμα από την Web Service πελάτη (ή αλλιώς καταναλωτή) με τη χρήση της δραστηριότητας receive. Αυτή είναι η τυπική έναρξη του κύκλου ζωής της παρουσίας της διαδικασίας, που δημιουργείται κατά τη λήψη της κλήσης από τη Web Service πελάτη
2. Βήμα 2: Η WS-BPEL διαδικασία καλεί με τη χρήση της δραστηριότητας invoke τη συνεργαζόμενη Web Service.
3. Βήμα 3: Η συνεργαζόμενη Web Service επιστρέφει την απάντησή της στην καλούσα WS-BPEL διαδικασία. Στο παράδειγμα μας έχουμε μία σύγχρονη επικοινωνία με την Web Service και η απάντηση μπορεί να ληφθεί μέσω της δραστηριότητας invoke. Στη περίπτωση ασύγχρονης επικοινωνίας τότε η απάντηση λαμβάνεται με τη χρήση της δραστηριότητας receive.
4. Βήμα 4: Η WS-BPEL διαδικασία επιστρέφει την απάντηση στη αιτούσα Web Service πελάτη με τη χρήση της δραστηριότητας reply.

Η WS-BPEL διαδικασία φαίνεται στο επόμενο σχήμα [8 σελ. 186]:



Εικόνα 23 Ένα απλό παράδειγμα μίας WS-BPEL διαδικασίας

Ένα παράδειγμα αρχείου που ορίζει τη συγκεκριμένη WS-BPEL διαδικασία θα μπορούσε να είναι το παρακάτω [8 σελ. 169]:

```
<?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">
   <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.wsdl"
           namespace="http://localhost/WebServices/wsdl/poOrderDoc"/>
   <partnerLinks>
      <partnerLink myRole="poInfoProviderRole" name="poInfoProvider"
                           partnerLinkType="ns1:poInfoLT"/>
      <partnerLink name="poDocRequester"
      partnerLinkType="ns1:poDocLT" partnerRole="poDocProviderRole"/>
   </partnerLinks>
   <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>
      <receive createInstance="yes"
               operation="getInfo"
               partnerLink="poInfoProvider"
               portType="ns1:poInfoPT"
               variable="poInfoRequestMessage"/>
      <assign>
         <copy>
            <from part="payload" variable="poInfoRequestMessage">
               <query>ns1:input</query>
            </from>
            <to part="pono" variable="poDocRequestMessage"/>
```

```
         </copy>
      </assign>
      <invoke inputVariable="poDocRequestMessage"
              outputVariable="poDocResponseMessage"
              operation="getOrderDoc"
              partnerLink="poDocRequester"
              portType="ns2:poOrderDocServicePortType ">
      </invoke>
      <assign>
         <copy>
            <from>$poDocResponseMessage.doc</from>
            <to>$poInfoResponseMessage.payload</to>
         </copy>
      </assign>
      <reply operation="getInfo"
             partnerLink="poInfoProvider"
             portType="ns1:poInfoPT"
             variable="poInfoResponseMessage"/>
   </sequence>
</process>
```

Το αρχείο WSDL που περιγράφει την Web Service με την οποία εξωτερικεύεται η συγκεκριμένη επιχειρησιακή διαδικασία είναι το παρακάτω:

```
<?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>
```

Η απλή διαδικασία του παραδείγματος δέχεται μία κλήση από την υπηρεσία πελάτη, αναθέτει τη τιμή ενός <part> της μεταβλητής εισόδου σε ένα <part> της μεταβλητής που χρησιμοποιεί στην κλήση στη συνεργαζόμενη υπηρεσία, λαμβάνει την απάντηση, αναθέτει τη τιμή ενός <part> της μεταβλητής που έλαβε στην απάντηση από τη συνεργαζόμενη υπηρεσία σε ένα <part> της μεταβλητής που επιστρέφει στην απάντηση της στην καλούσα υπηρεσία.

Υλοποίηση μίας μηχανής εκτέλεσης WS-BPEL σε PHP

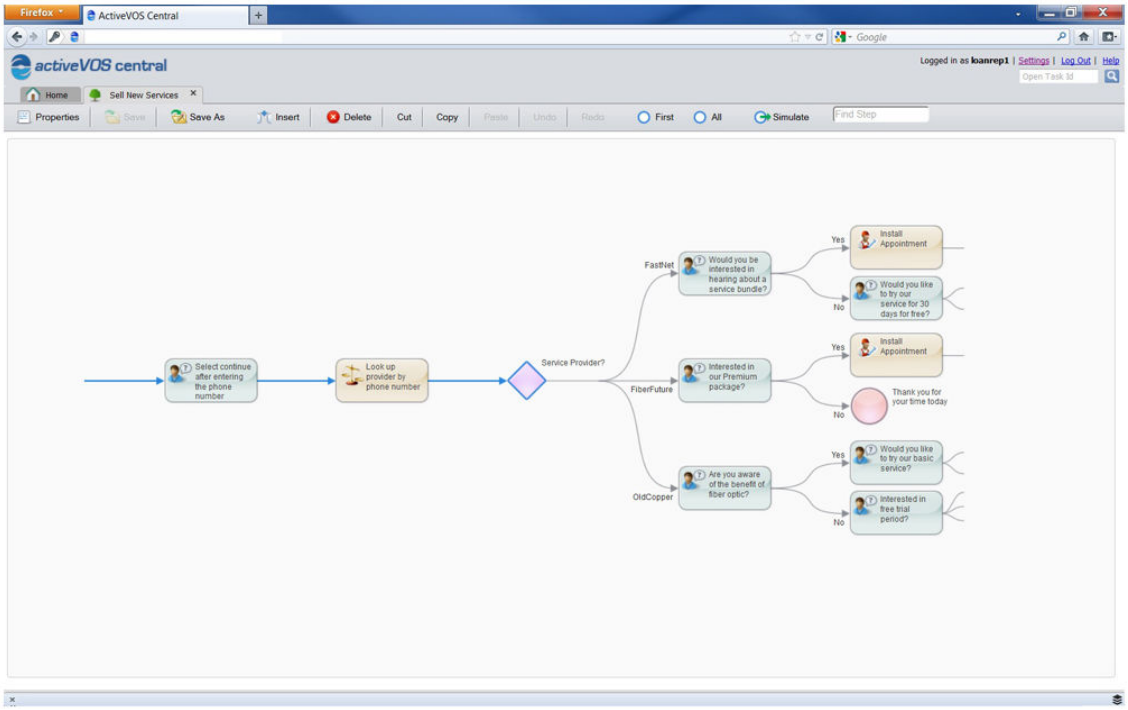
3.1 Υπάρχουσες υλοποιήσεις μηχανών BPEL

Σήμερα υπάρχει ένα μεγάλος αριθμός υλοποιήσεων μηχανών BPEL σε εμπορικές εκδόσεις είτε ως ελεύθερο λογισμικό, όπως φαίνονται στο παρακάτω πίνακα [18]. Οι περισσότερες από αυτές βασίζονται σε πλατφόρμες Java.

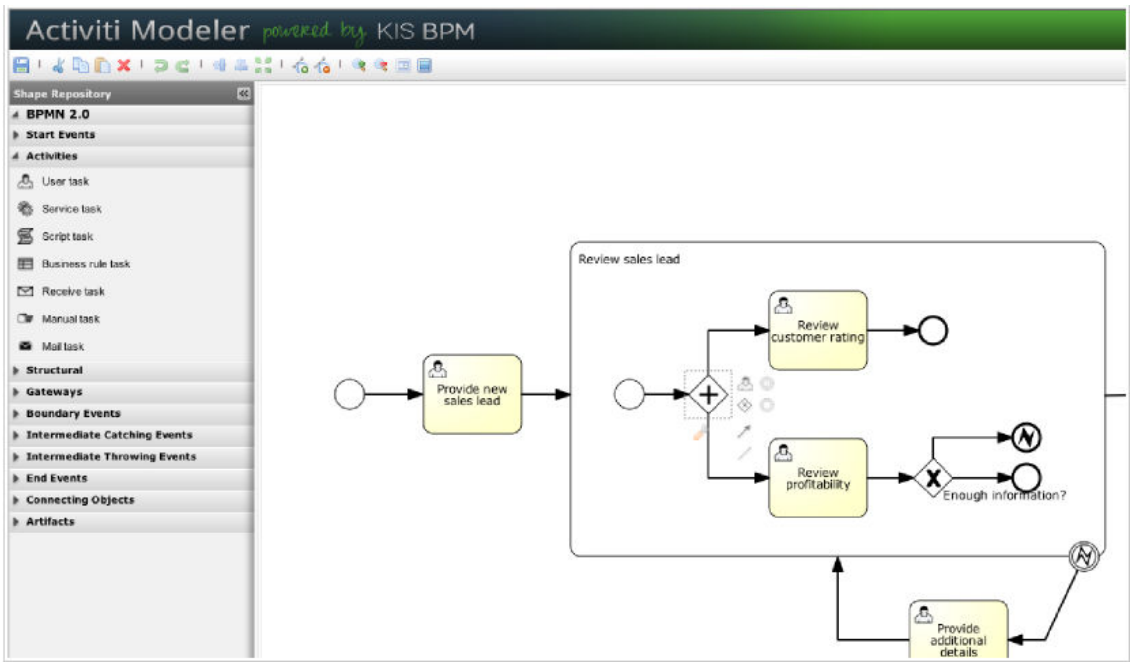
Μηχανή	Εταιρεία	Πλατφόρμα	Υποστηριζόμενα πρότυπα	Άδεια
ActiveVOS	Active Endpoints	Servlet or Java EE	BPMN 2.0; WS-BPEL; BPEL4People / WS-HumanTask	Εμπορική
Activiti	Alfresco and the Activiti community	Java	BPMN 2.0	Apache 2.0
Apache ODE	ASF	Apache Axis, JBI Java EE	BPEL4WS 1.1, WS-BPEL 2.0 WS-HumanTask με Apache HISE	Apache
BizTalk Server	Microsoft	.NET	BPEL, BPMN, RFID, WSDL, UDDI, WS-*	Εμπορική
ExpressBPEL BPM	CodeBrew Technologies	Apache Axis	WS-BPEL 2.0, BPEL4WS 1.1	Εμπορική
iBolt Server	Magic Software Enterprises	Java EE	BPEL4WS	Εμπορική
jBPM	jBoss	Java EE	BPMN 2.0	Apache 2.0
Open ESB	OpenESB Community	Java EE, JBI	WS-BPEL 2.0	CDDL
Oracle BPEL Process Manager	Oracle Corporation	Java EE	WS-BPEL 2.0, BPMN 2.0	Εμπορική
OW2 Orchestra	OW2	Apache Axis Apache CXF OSGi Java EE	WS-BPEL 2.0	LGPL
Parasoft BPEL Maestro	Parasoft	Servlet	WS-BPEL, BPEL4People / WS-HumanTask	Εμπορική
Petals BPEL Engine	Petals Link	Java EE	WS-BPEL 2.0, WSDL 1.1 and 2.0	LGPL
SAP Exchange Infrastructure	SAP AG		BPEL	Εμπορική
Virtuoso Universal Server	OpenLink Software		UDDI, WS-BPEL, WS-*	GPL και εμπορική
WebSphere Process Server	IBM	Java EE	WS-BPEL	Εμπορική

Πίνακας 4 Μηχανές BPEL, BPEL4WS και BPMN

Οι περισσότερες υλοποιήσεις και ειδικά οι εμπορικές παρέχουν στο χρήστη ένα γραφικό περιβάλλον διαχείρισης, μέσω του οποίου διευκολύνεται η σχεδίαση των BPEL διαδικασιών (ή των BPMN διαδικασιών που υποστηρίζει ο jBoss) και αυτοματοποιείται η ανάπτυξη τους στην πληροφοριακή υποδομή στην οποία υλοποιούνται οι επιχειρησιακές διαδικασίες.

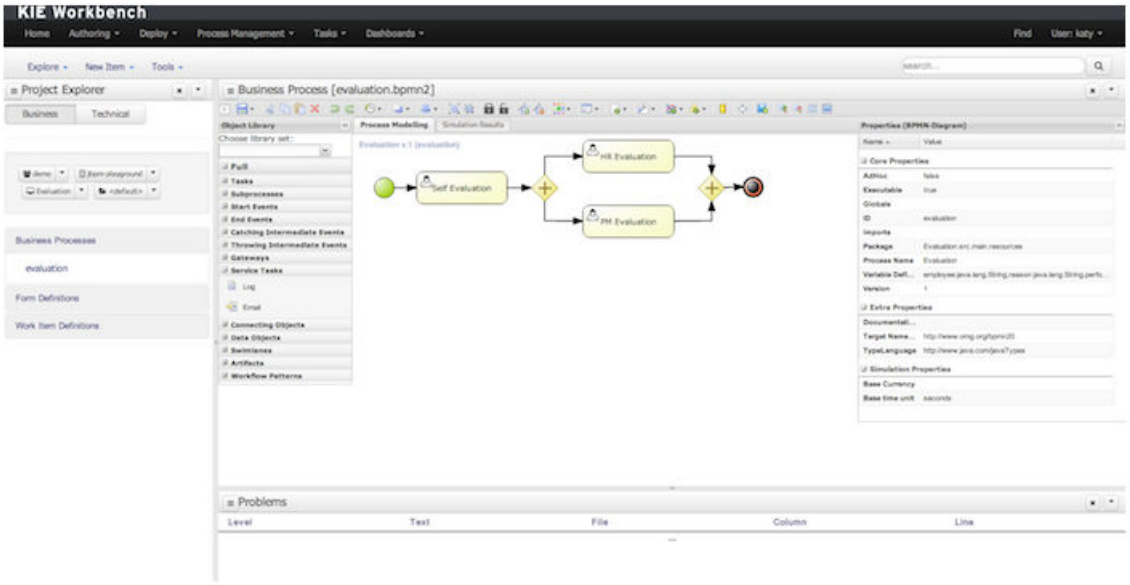


Εικόνα 24 Γραφικό περιβάλλον AciveVos[2]



Εικόνα 25 Γραφικό περιβάλλον Activiti Modeler[3]

[2] http://www.activevos.com/products/activevos/features/screenflow
[3] http://www.activiti.org/screenshots.html#images/screenshots/activiti-modeler.png



Εικόνα 26 Γραφικό περιβάλλον jBoss[4]

Μηχανές όπως ο Business Process Server της Oracle που είναι μέρος του Oracle Suite 11g ή ο BizTalk Server της Microsoft, ολοκληρωμένες λύσεις., που περιλαμβάνουν εκτός της μηχανής BPEL και επιπλέον εφαρμογές και λειτουργικότητα όπως Enterprise Service Bus, Business Rules Engine, Business Activity Monitoring, Human Workflow Service, Mediators, Adapters και άλλα.

3.2 Προβλήματα στην υλοποίηση σε PHP

Η PHP είναι κατά κύριο λόγω server-side scripting γλώσσα και λιγότερο μία γενική γλώσσα προγραμματισμού, που έχει σχεδιαστεί ειδικά για το διαδίκτυο. Αρχικά αναπτύχθηκε γύρω στο 1995 για τη διαχείριση μικρών σε μέγεθος ιστοχώρων και ονομαζόταν Personal Home Page. Στη συνέχεια όμως αναπτύχθηκε σε μία ιδιαίτερα ανθεκτική γλώσσα, με δυνατότητες επέκτασης, ικανή να υποστηρίζει δυναμικές ιστοσελίδες. Από την έκδοση 4 μπορεί να χαρακτηριστεί αντικειμενοστραφής γλώσσα προγραμματισμού (object-oriented language, OOP). Σήμερα στην έκδοση 5.3 η δυνατότητες αντικειμενοστραφούς προγραμματισμού είναι σε ιδιαίτερα ικανοποιητικό επίπεδο και είναι σε θέση να υποστηρίζει εγγενώς τα πρότυπα XML και SOAP.

Τα βασικά προβλήματα της υλοποίησης σε PHP είναι δύο:

1. Μη ικανοποιητική υποστήριξη μακροχρόνιων διαδικασιών

   Η PHP από τη φύση της είναι σχεδιασμένη για να δέχεται ένα αίτημα από τον επισκέπτη μίας ιστοσελίδας, να παράγει την ιστοσελίδα στον διαδικτυακό εξυπηρετητή και αυτός να τη σερβίρει στον επισκέπτη. Διαδικασίες που θα εκτελούνται επί μακρό χρονικό διάστημα δεν είναι συνηθισμένες στο περιβάλλον εκτέλεσης της PHP σε ένα εξυπηρετητή όπως για παράδειγμα ο Apache, Το περιβάλλον εκτέλεσης cli (command line interface) της PHP παρέχει δυνατότητες εκτέλεσης μακροχρόνιων διαδικασιών, αλλά η αρχική κλήση της WS-BPEL διαδικασίας θα πρέπει να εξυπηρετηθεί σε περιβάλλον εξυπηρετητή.
2. Μη ικανοποιητική υποστήριξη πολυνηματικής επεξεργασίας (multithreading)

   Η υλοποίηση της δραστηριότητας <flow> απαιτεί την ταυτόχρονη εκτέλεση των δραστηριοτήτων που περιέχονται σε αυτή και συνεπώς απαιτεί την ταυτόχρονη εκτέλεση δύο ή περισσότερων διεργασιών. Αυτό μπορεί να επιτευχθεί είτε με multiprocessing είτε με multithreading. Η PHP δεν υποστηρίζει εγγενώς πολυνηματική επεξεργασία. Πρόσθετες βιβλιοθήκες παρέχουν τέτοιες δυνατότητες, όπως η pthreads, η οποία όμως δεν είναι διαθέτει ικανοποιητική τεκμηρίωση. Επίσης το πρόβλημα μπορεί να αντιμετωπιστεί με τη χρήση ασύγχρονων κλήσεων σε άλλα PHP scripts, ή με τη

[4] http://docs.jboss.org/jbpm/v6.0.1/userguide/jBPMOverview.html#d0e32

βοήθεια της βιβλιοθήκης gearman. Η gearman είναι μία πλατφόρμα που παρέχει τη δυνατότητα ανάθεσης της εκτέλεσης εργασιών σε άλλες μηχανές ή διεργασίες, ώστε να επιτυγχάνεται παράλληλη εκτέλεση και χρησιμοποιεί τη λογική των πελατών (client) και εργατών (workers) [19]. Παρέχει διεπαφές προγραμματισμού εφαρμογών (application programming interfaces) για διάφορες γλώσσες, μεταξύ αυτών και για PHP.

Στη συγκεκριμένη υλοποίηση, για την υλοποίηση της δραστηριότητας <flow>, χρησιμοποιήθηκε η τεχνική της ασύγχρονης κλήσης άλλων PHP scripts.

3. Η επέκταση SOAP της PHP δεν υποστηρίζει την μεταφορά δεδομένων τύπου array. Η μοναδική δυνατότητα μεταφοράς σύνθετων δεδομένων είναι σε μορφή αντικειμένου PHP που θα αντιστοιχεί σε ένα xsd:complexType στο αρχείο WSDL. Επομένως η χρήση messages που περιέχουν πολλά parts δεν είναι εφικτή στη συγκεκριμένη υλοποίηση.
4. Η έλλειψη προηγμένων δυνατοτήτων αποσφαλμάτωσης (debug) κατά την ανάπτυξη του κώδικα. Επειδή η PHP δεν προορίζεται για την ανάπτυξη λογισμικού που εκτελεί πολύπλοκες λειτουργίες, τα διαθέσιμα περιβάλλοντα ανάπτυξης δεν παρέχουν ικανοποιητικές δυνατότητες αποσφαλμάτωσης. Στο πλαίσιο της εργασίας χρησιμοποιήθηκαν σε μεγάλο βαθμό έμμεσες τεχνικές, όπως εκτύπωση στην οθόνη ή σε αρχείο. Ειδικά η ασύγχρονη κλήση σε άλλα PHP scripts δεν ήταν δυνατόν να αποσφαλματωθεί διαφορετικά.

3.3 Τι υλοποιεί η συγκεκριμένη μηχανή

Η μηχανή εκτέλεσης BEL σε PHP, στο πλαίσιο της συγκεκριμένης εργασίας και των περιορισμών της PHP που αναφέρθηκαν προηγουμένως, περιορίστηκε όσον αφορά τη λειτουργικότητα του προτύπου που θα υποστήριζε. Δεδομένου του στόχου της εργασίας, δηλαδή της γνωριμίας και της κατά το δυνατό εμβάθυνσης στο πρότυπο WS-BPEL πρωτίστως και όσο απαιτείτο στο SOA και τις Web Services, δεν κρίθηκε απαραίτητο να υλοποιηθεί οι πλέον απαιτητικές λειτουργικότητες, όπως η αντιστάθμιση. Επίσης δραστηριότητες, η υλοποίηση των οποίων δεν θα προσέφερε πρόσθετη αξία, δεν υποστηρίζονται.

Από το σύνολο των δραστηριοτήτων υποστηρίζονται οι παρακάτω:

- <receive>
- <reply>
- <invoke>
- <assign>
- <sequence>
- <if>
- <exit>

Για τη δραστηριότητα <flow> έχει αναπτυχθεί κώδικας που βασίζεται στην ασύγχρονη κλήση τρίτων PHP scripts, ο οποίος όμως είναι ημιτελής και δεν έχει ελεγχθεί επαρκώς. Η βασική λογική της συγκεκριμένης τεχνικής φαίνεται να είναι επιτυχής και μία μελλοντική ολοκληρωμένη υλοποίηση πιθανόν να μπορεί να βασισθεί σε αυτή.

Από τα βασικά χαρακτηριστικά της WS-BPEL δεν υποστηρίζονται η διαχείριση σφαλμάτων, η αντιστάθμιση, οι εμβέλειες και η διαχείριση συμβάντων. Αντίθετα υποστηρίζονται οι συσχετίσεις και τα σύνολα συσχετίσεων.

3.4 Περιγραφή της υλοποίησης

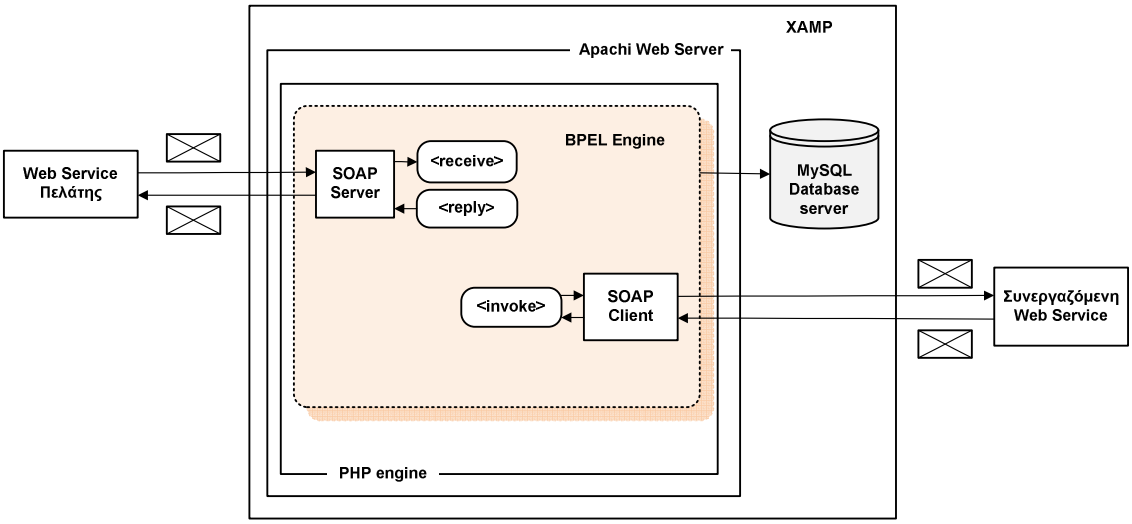
3.4.1 Αρχιτεκτονική της υλοποίησης

Η υλοποίηση της μηχανής εκτέλεσης BPEL έχει γίνει σε PHP 5. Τα PHP scripts μπορούν να εκτελεστούν σε οποιοδήποτε Web Server υποστηρίζει PHP. Στο περιβάλλον ανάπτυξης χρησιμοποιήθηκε η ολοκληρωμένη λύση XAMPP 1.8.0 για Windows, που περιέχει ένα Apache Server, μία μηχανή PHP, μία DBMS MySQL. Τα βασικά συστατικά μέρη είναι αυτά που φαίνονται στην Εικόνα 28 Η ανάπτυξη Web Services με τη χρήση της επέκτασης SOAP της PHP, με την προσθήκη του κώδικα PHP που εκτελεί τις BPEL διαδικασίες, δηλαδή:

1. Web Server

2. Database Server
3. PHP SOAP Web Service Client/Server
4. PHP BPEL engine
5. PHP database store/retrieve engine for BPEL process data

Τα δύο πρώτα παρέχονται από το XAMPP, ενώ τα άλλα τρία αναπτύχθηκαν. Στο επόμενο σχήμα παρουσιάζεται η αρχιτεκτονική της υλοποίησης.



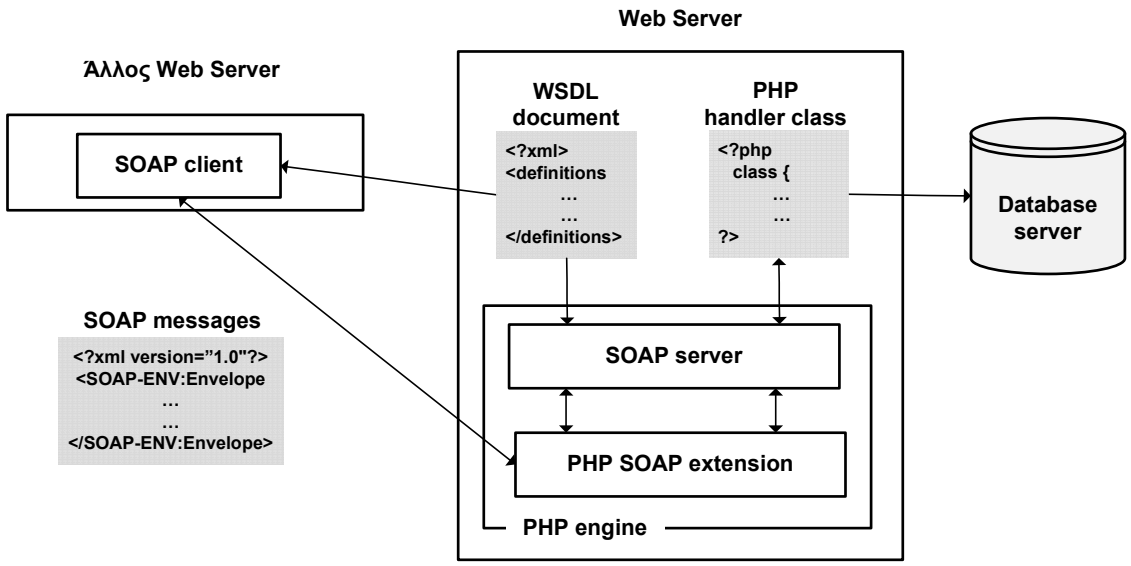
Εικόνα 27 Η αρχιτεκτονική της μηχανής εκτέλεσης BPEL

Όπως έχει ήδη αναφερθεί αναλυτικά μία διαδικασία BPEL εξωτερικεύει τη λειτουργικότητα της μέσω Web Service και επομένως καλείται ως τέτοια. Η διαδικασία συνεπώς μπορεί να κληθεί από οποιαδήποτε εφαρμογή υποστηρίζει Web Services. Η διαχείριση των κλήσεων που δέχεται από τις καλούσες Web Services η μηχανή, υλοποιείται SOAP Server που υλοποιείται με την σχετική επέκταση της PHP. Η διαχείριση των κλήσεων προς τις συνεργαζόμενες υπηρεσίες υλοποιείται επίσης με SOPA Client της ίδιας επέκτασης. Ο SOAP Server αναλαμβάνει να ικανοποιήσει την κλήση και καλεί τις PHP κλάσεις που υλοποιούν την μηχανή εκτέλεσης BPEL. Όταν απαιτηθεί να κληθεί μία συνεργαζόμενη Web Service οι PHP κλάσεις που υλοποιούν τη μηχανή καλούν ένα SOAP Client για την επικοινωνία με αυτέ. Η υποστήριξη χαρακτηριστικών της BPEL, όπως συσχετίσεις και διαδικασίες που απαιτούν για την ολοκλήρωση τους μεγάλα χρονικά διαστήματα απαιτεί την αποθήκευση δεδομένων της διαδικασίας, όπως για παράδειγμα τιμές μεταβλητών ή το ID που αποδίδεται στη παρουσία μίας διαδικασίας, και μεταδεδομένων που αφορούν στη διαδικασία, όπως τα σχετικά αρχεία WSDL.

3.4.2 PHP SOAP client/server

Για την υποστήριξη των Web Services από την PHP χρησιμοποιήθηκε η επέκταση SOAP της PHP [20]. Η επέκταση έχει αναπτυχθεί σε C και είναι ταχύτερη από την υλοποίηση PEAR::SOAP, που έχει αναπτυχθεί σε PHP. Η επέκταση υποστηρίζει τα SOAP 1.1 και SOAP 1.2 πρωτόκολλα, μπορεί να λειτουργήσει με ή χωρίς τη χρήση WSDL κειμένων για την περιγραφή της Web Service που υλοποιείται, και χρησιμοποιεί δύο βασικές PHP κλάσεις: την SOAP client και την SOAP server.

Με την πρώτη υλοποιούμε τις κλήσεις σε ένα SOAP server. Όταν η υλοποίηση χρησιμοποιεί WSDL η SOAP client κλάση αναλαμβάνει τις απαραίτητες μετατροπές τύπων δεδομένων και τις κωδικοποιήσεις που απαιτούνται προκειμένου να δημιουργήσει τα κατάλληλα μηνύματα SOAP από τις υπάρχουσες PHP κλάσεις και τύπους. Αντίστοιχα η κλάση SOAP server αναλαμβάνει την εξυπηρέτηση της κλήσης που δέχεται η υλοποιούμενη Web Service. Οι operations που περιλαμβάνονται στο WSDK κείμενο υλοποιούνται ως συναρτήσεις (functions) στον SOAP server. Στο επόμενο σχήμα παρουσιάζεται σχηματικά μία τέτοια υλοποίηση [12 σελ. 298]:



Εικόνα 28 Η ανάπτυξη Web Services με τη χρήση της επέκτασης SOAP της PHP

Ένα απλό παράδειγμα υλοποίησης του SOAP client παρουσιάζεται παρακάτω:

```
$wsdl = "wsdl/test.wsdl";
ini_set("soap.wsdl_cache_enabled", "0");
$client = new SoapClient($wsdl, array('trace' => 1));
$input="Hello world";
try {
     $response = $client->hello($input);
} catch(SoapFault $fault) {
    echo 'Request : <br/><xmp>',
    $client->__getLastRequest(),
    '</xmp><br/><br/> Error Message : <br/>',
    $fault->getMessage();
}
```

Ο αντίστοιχος SOAP server είναι:

```
function hello($input) {
     return "The client said : " . $input;
}
ini_set("soap.wsdl_cache_enabled", "0");
$server = new SoapServer("wsdl/Simple_SOAP_WS.wsdl");
$server->addFunction("hello");
$server->handle();
```

Και το WSDL αρχείο:

```
<?xml version='1.0' encoding='UTF-8' ?>
<definitions targetNamespace='myHello'
xmlns='http://schemas.xmlsoap.org/wsdl/'
xmlns:plnk='http://schemas.xmlsoap.org/ws/2003/05/partner-link/'
xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
xmlns:tns='myHello'
xmlns:xsd='http://www.w3.org/2001/XMLSchema'>
```

```
<message name='outputMessage'>
<part name='hello' type='xsd:string' />
</message>
<message name='inputMessage'>
<part name='firstName' type='xsd:string' />
</message>

<portType name='helloServicePT'>
<operation name='hello'>
<input message='tns:inputMessage' name='inputMessage' />
<output message='tns:outputMessage' name='outputMessage' />
</operation>
</portType>

<binding name='helloServicePTBinding' type='tns:helloServicePT'>
  <soap:binding style='rpc'
  transport='http://schemas.xmlsoap.org/soap/http'
  xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' />
  <operation name='hello'>
  <soap:operation soapAction='' style='rpc'
  xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' />
  <input>
  <soap:body encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'
  use='encoded'
  xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' />
  </input>
  <output>
  <soap:body encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'
  use='encoded'
  xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' />
  </output>
  </operation>
</binding>

<service name='helloServicePT'>
 <port binding='tns:helloServicePTBinding' name='helloServicePTPort'>
 <soap:address location='http://localhost/Simple_SOAP_WS_Server.php'
 xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'/>
 </port>
</service>

</definitions>
```

Σημειώνεται ότι συνολικά απαιτούνται δύο αρχεία WSDL, ένα που περιγράφει τη Web Service και ένα δεύτερο που περιέχει την πληροφορία για τις συσχετίσεις. Και αυτό γιατί αν στο αρχείο που περιγράφεται η Web Service προστεθούν και οι συσχετίσεις, οι οποίες ουσιαστικά είναι επέκταση του προτύπου WSDL, η καλούσα Web Service πελάτης, ενδέχεται να μην αναγνωρίζει τις επεκτάσεις και να παρουσιάζει σφάλμα. Ο PHP SOAP client αναφέρεται στο πρώτο WSDL αρχείο. Το δεύτερο αναφέρεται στη εγγραφή της συγκεκριμένης BPEL διαδικασίας στη βάση δεδομένων.

3.4.3 Επεξεργασία των αρχείων XML

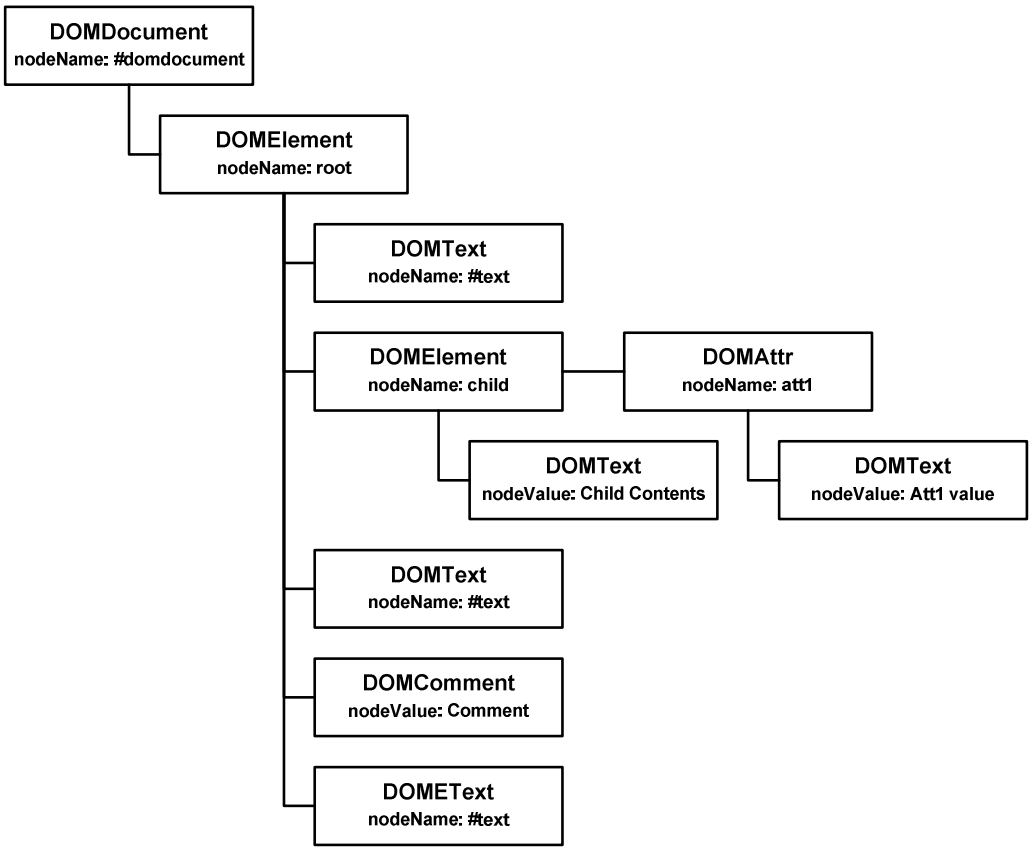
Για την ανάγνωση και επεξεργασία των XML αρχείων περιγραφής των Web Services και της WS-BPEL διαδικασίας (WSDL και BPEL αρχείων αντίστοιχα) χρησιμοποιήθηκε η επέκταση DOM της PHP [21]. Η συγκεκριμένη επέκταση παρέχει τη δυνατότητα εκτέλεσης εργασιών σε ένα XML κείμενο μέσω της DOM API της PHP 5.

Το Document Object Management είναι ένα σύνολο διεπαφών που επιτρέπει την πρόσβαση σε κείμενα και την επεξεργασίας τους. Αποτελεί μόνο το πρότυπο του World Wide Web Consortium (W3C), αλλά στη πραγματικότητα είναι σύνολο προτύπων. Η βασική

λειτουργικότητα του αναλύεται σε τρία επίπεδα, Level 1, Level 2 και Level 3, καθένα από τα οποία παρέχει αυξανόμενη λειτουργικότητα.

Στο DOM, ένα κείμενο αντιμετωπίζεται ως δένδρο που αποτελείται από κόμβους (nodes). Οι τύποι κόμβων είναι οι παρακάτω η ιεραρχία τους παρουσιάζεται στην επόμενη εικόνα [13 σελ. 183]:

- Attr: Attribute node
- CDATASection: CDATA section node
- Comment: Comment node
- DocumentFragment: Document fragment node
- Document: Document node
- DocumentType: Document type node
- Element: Element node
- Entity: Entity node
- EntityReference: Entity reference node
- Notation: Notation node
- ProcessingInstruction: PI node
- Text: Text node



Εικόνα 29 Η DOM object view ενός δένδρου DOM

Κάθε κόμβος αντιστοιχεί σε ένα αντικείμενο DOM. Επιπρόσθετα υπάρχουν αντικείμενα που δεν κληρονομούνται από τους κόμβους, όπως: NodeList , NameNodeMap, DOMImplementation, DOMException και CharacterData. Κάθε κόμβος είναι προσπελάσιμος και η λειτουργικότητα που υποστηρίζεται εξαρτάται από τον τύπο του κόμβου.

3.4.4 Βασικές κλάσεις και αντικείμενα

Η μηχανή εκτέλεσης BPEL σε PHP έχει υλοποιηθεί με τη μέθοδο του αντικειμενοστραφούς προγραμματισμού.

Τα βασικά αντικείμενα και οι αντίστοιχες κλάσεις που τα προδιαγράφουν στα οποία βασίζεται η υλοποίηση είναι τρία:

1. ReadWsdl

   Αντιστοιχεί στο αρχείο WSDL. Οι ιδιότητες του περιλαμβάνουν το σύνολο της πληροφορίας που περιέχει το WSDL αρχείο: messages, parts, portTypes, bindings, services, το operation που χρησιμοποιεί η Web Service που καλεί την BPEL διαδικασία.

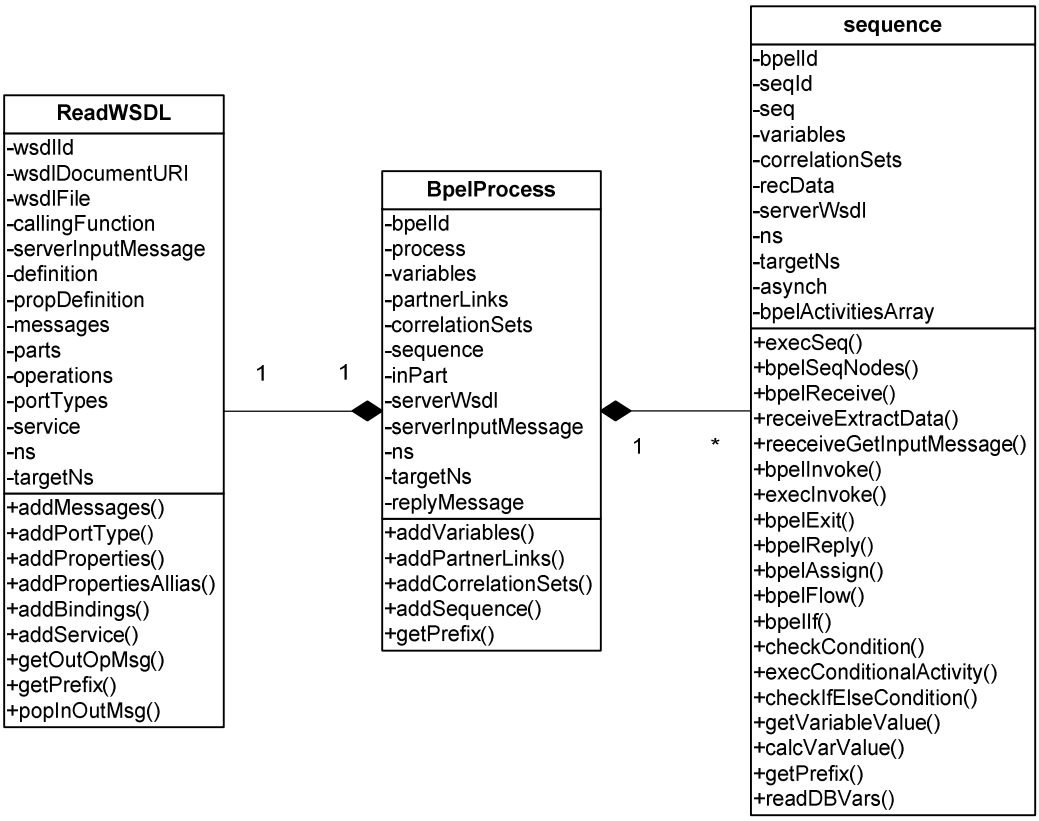
2. BpelProcess

   Αντιστοιχεί στο αρχείο που περιγράφει τη BPEL διαδικασία. Οι ιδιότητες του περιλαμβάνουν την ίδια τη διαδικασία, variables, partnerLinks, sequence,

3. Sequence

   Η κλάση Sequence αποτελεί το πυρήνα της υλοποίησης. Η συνάρτηση execSeq εκτελεί ουσιαστικά τη διαδικασία. Περιλαμβάνει συναρτήσεις για κάθε μία από τις δραστηριότητες που μπορεί να περιέχει μία διαδικασία BPEL. Σημειώνεται ότι η ύπαρξη μίας πρωτεύουσας δραστηριότητας <sequence> είναι προϋπόθεση στη συγκεκριμένη υλοποίηση.

Σχεδόν το σύνολο του κώδικα που υλοποιεί τη μηχανή BPEL βρίσκεται σε αυτές τις τρεις κλάσεις. Επιπρόσθετα έχει αναπτυχθεί κώδικας για την αποσφαλμάτωση, που λόγω των περιορισμών που αναφέρθηκαν παραπάνω, βασίστηκε σε μεγάλο βαθμό σε εγγραφές σε αρχεία συμβάντων (log files). Αυτός ο κώδικας δεν περιγράφεται στη παρούσα εργασία.

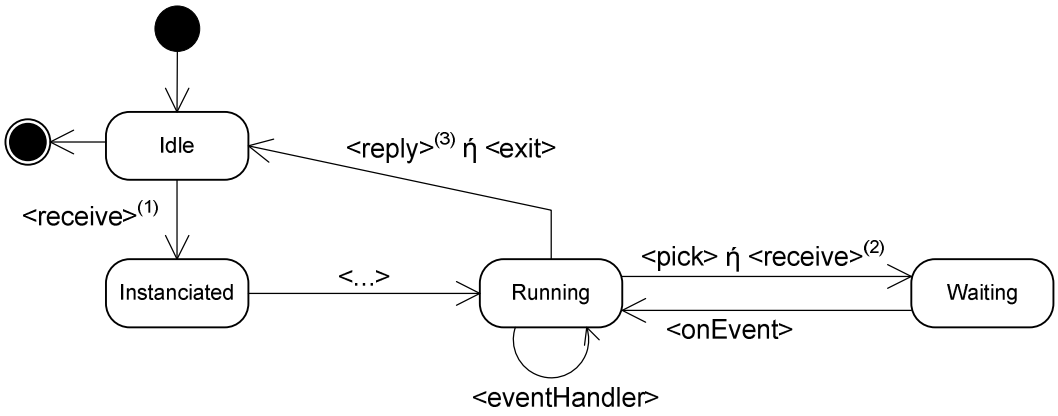


Εικόνα 30 Τα βασικά αντικείμενα της υλοποίησης

3.4.5 Οι καταστάσεις μίας BPEL διαδικασίας

Μία διαδικασία από τη στιγμή που θα δημιουργηθεί το αρχείο BPEL που την περιγράφει, καθώς και τα δύο αρχεία WSDL, που περιγράφουν τις Web Services και τις συσχετίσεις, πρέπει να

καταχωρηθεί στη βάση δεδομένων, ώστε να είναι διαθέσιμη αν κληθεί. Η διαδικασία είναι ανενεργή (idle). Αν μία Web Service πραγματοποιήσει μία κλήση με αναφορά στο σχετικό WSDL η μηχανή θα διαβάσει τα αρχεία WSDL και BPEL, θα δημιουργήσει τα σχετικά αντικείμενα. Στη συνέχεια θα ελέγξει αρχικά αν στη βάση δεδομένων υπάρχει εγγραφή που αντιστοιχεί στη συγκεκριμένη BPEL διαδικασία, βάσει του συγκεκριμένου αρχείου WSDL. Αν βρεθεί σχετική εγγραφή τότε η μηχανή θα αναζητήσει δραστηριότητα <receive> και θα ελέγξει αν αντιστοιχεί σε αυτή που αναμένεται ως "αρχική" <receive> ή οποία εκκινεί τη διαδικασία (createInstance="yes"). Τότε δημιουργείται μία παρουσία (instance) της διαδικασίας, δηλαδή η διαδικασία είναι instantiated, και η διαδικασία πλέον εκτελείται (running). Αν κατά την εκτέλεση βρεθεί μία δραστηριότητα <pick> ή <receive> που δεν προϋποθέτει τη δημιουργία παρουσίας (createInstance="no"), τότε η διαδικασία τίθεται σε κατάσταση αναμονής (waiting). Από την αναμονή θα αρχίσει να εκτελείται και πάλι όταν θα δεχθεί μία κλήση είτε ως συμβάν (<onEvent>) είτε από τη Web Service που αντιστοιχεί στη δραστηριότητα <receive> που την έθεσε σε αναμονή (βάσει operation που κάνει την κλήση). Τα παραπάνω φαίνονται στο επόμενο σχήμα:



(1): createInstance="yes"
(2): createInstance="no"
(3): Αν η <reply> είναι η τελευταία δραστηριότητα

Εικόνα 31 Οι δυνατές καταστάσεις μίας διαδικασίας

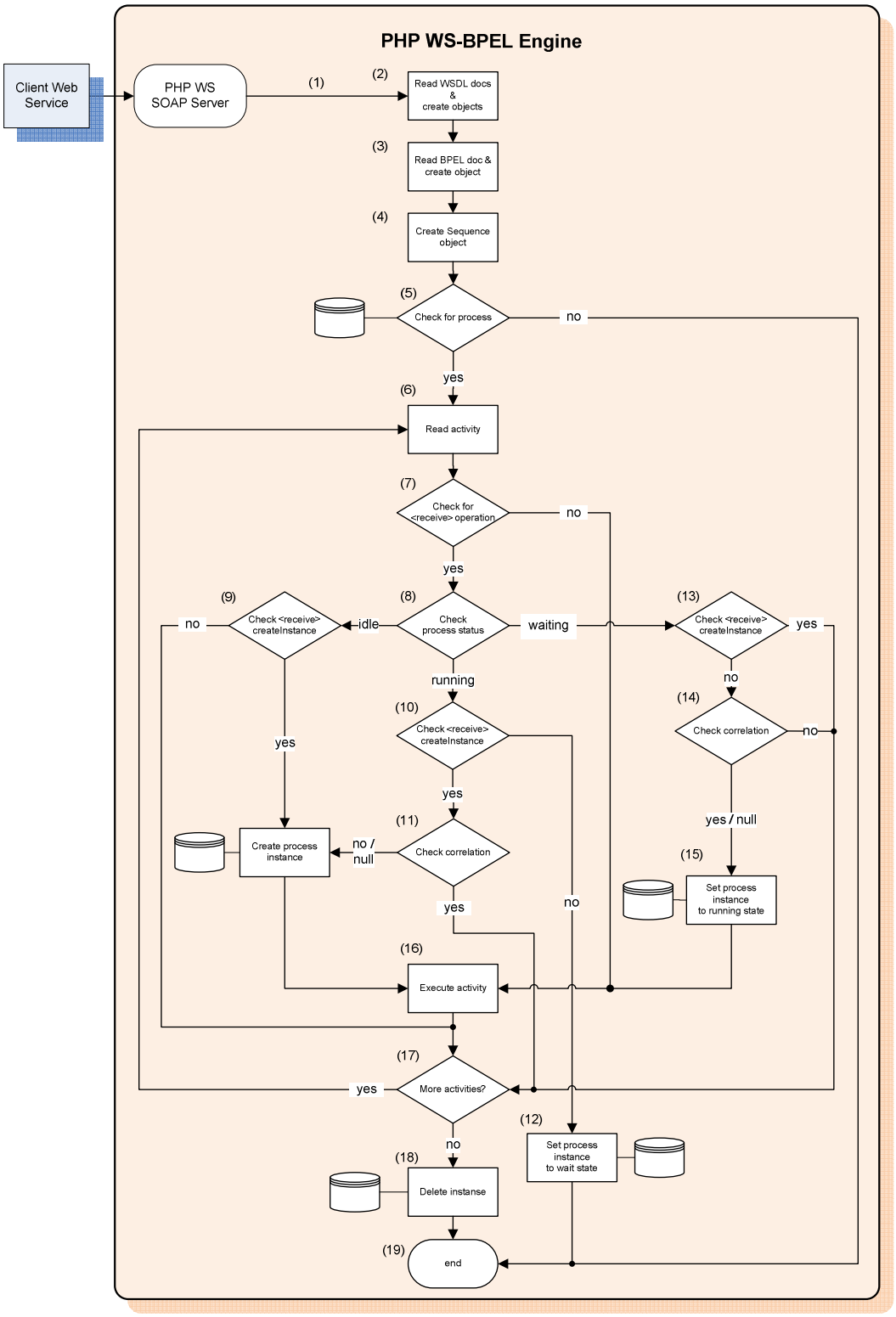
3.4.6 Η εκτέλεσης μίας BPEL διαδικασίας

Όταν γίνει μία κλήση προς την Web Service που αντιστοιχεί σε μία BPEL διαδικασία, η μηχανή εκτελεί τις παρακάτω ενέργειες:

1. Ο PHP SOAP Server δέχεται μία κλήση από τη πελάτη Web Service της διαδικασίας και καλεί τη μηχανή εκτέλεσης BPEL
2. Η μηχανή διαβάζει τα αρχεία WSDL και δημιουργεί το αντίστοιχο αντικείμενο
3. Η μηχανή διαβάζει το αρχείο BPEL και δημιουργεί το αντίστοιχο αντικείμενο
4. Η μηχανή δημιουργεί το αντικείμενο Sequence
5. Η μηχανή αναζητά στη βάση δεδομένων αν υπάρχει εγγραφή για τη συγκεκριμένη διαδικασία
6. Η μηχανή διαβάζει τη δραστηριότητα
7. Η μηχανή ελέγχει αν η δραστηριότητα είναι <receive>
8. Εάν ναι ελέγχεται στη βάση δεδομένων αν υπάρχει παρουσία σε κατάσταση εκτέλεσης (running), αναμονής (waiting)
9. Αν δεν υπάρχει παρουσία, ελέγχεται αν η δραστηριότητα <receive> εκκινεί τη διαδικασία (createInstance="yes") και εάν ναι, δημιουργείται μία παρουσία της διαδικασίας εφόσον η function της καλούσας Web Sevice είναι ίδια με την <operation> της <receive>
10. Αν υπάρχει παρουσία που εκτελείται, ελέγχεται αν η δραστηριότητα <receive> εκκινεί τη διαδικασία (createInstance="yes")

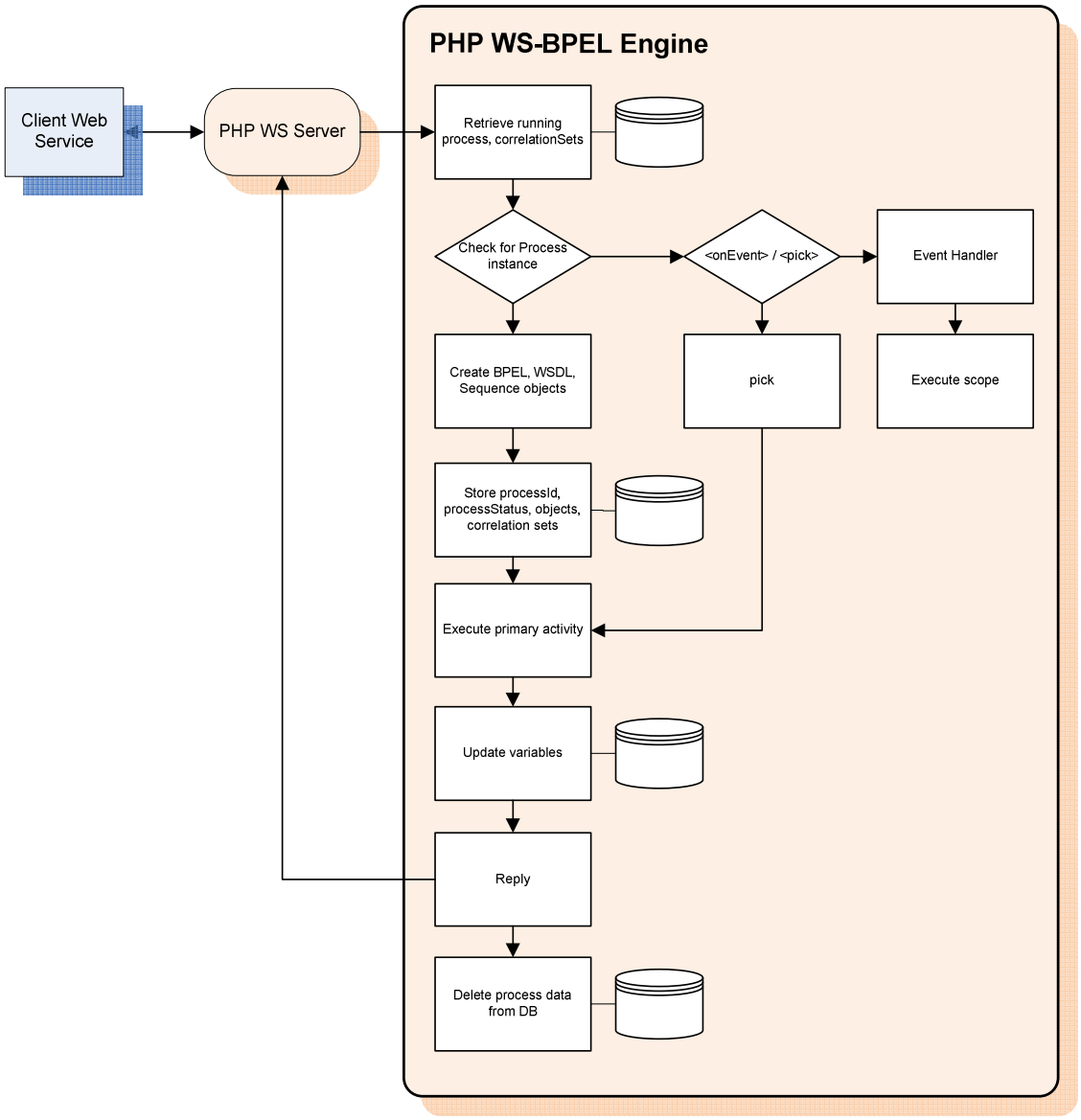
11. Εάν ναι, ελέγχεται τυχόν συσχέτιση των τιμών των <variables> που δέχτηκε η διαδικασία με τις τιμές των αντίστοιχων αποθηκευμένων <variables> της παρουσίας που εκτελείται, και εφόσον δεν υπάρχει ή η <reveive> δεν περιέχει <correlations>, δημιουργείται μία παρουσία της διαδικασίας
12. Εάν όχι, τότε η παροσία της διαδικασίας τίθεται σε κατάσταση αναμονής
13. Εάν υπάρχει παρουσία σε κατάσταση αναμονής, ελέγχεται αν η δραστηριότητα <receive> εκκινεί τη διαδικασία (createInstance="yes")
14. Εάν όχι, ελέγχεται τυχόν συσχέτιση των τιμών των <variables> που δέχτηκε η διαδικασία με τις τιμές των αντίστοιχων αποθηκευμένων <variables> της παρουσίας που είναι σε αναμονή, και εάν υπάρχει ή <reveive> δεν περιέχει <correlations>, η παρουσία τίθεται σε κατάσταση εκτέλεσης
15. Εάν υπάρχει συσχέτιση ή <reveive> δεν περιέχει <correlations>, η παρουσία τίθεται σε κατάσταση εκτέλεσης
16. Ολοκληρώνεται η εκτέλεση της διαδικασίας < reveive >
17. Ελέγχεται αν υπάρχει επόμενη δραστηριότητα προς εκτέλεση και εάν ναι, επαναλαμβάνεται το βήμα (6)
18. Εάν όχι, διαγράφεται από τη βάση δεδομένων η παρουσία που μόλις εκτελέστηκε
19. Η εκτέλεση της διαδικασίας ολοκληρώνεται

Τα παραπάνω απεικονίζονται στο επόμενο διάγραμμα:



Εικόνα 32 Διάγραμμα ροής εκτέλεσης μίας BPEL διαδικασίας

Στο επόμενο διάγραμμα φαίνεται απλοποιημένα ο τυπικός κύκλος εκτέλεσης μίας διαδικασίας που ολοκληρώνεται με τη δραστηριότητα <reply> που επιστρέφει τη τιμή της μεταβλητής που έχει ορισθεί ως output στην καλούσα operation.



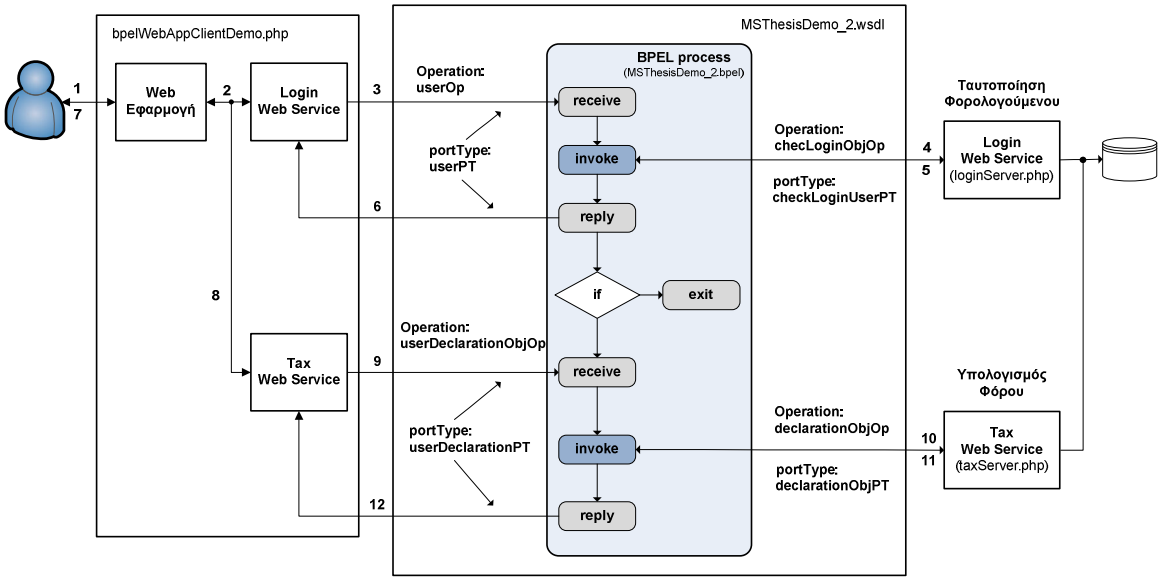
Εικόνα 33 Τυπικός κύκλος εκτέλεσης μίας διαδικασίας

3.5 Παράδειγμα εφαρμογής

Παρακάτω παρουσιάζεται ένα παράδειγμα διαδικασίας.

Η διαδικασία του παραδείγματος προσομοιάζει την παροχή μίας φορολογικής υπηρεσίας υπολογισμού και πληρωμής φόρου που προσφέρει το Υπουργείο Οικονομικών . Ο χρήστης φορολογούμενος εισέρχεται στην υπηρεσία μέσω μίας διαδικτυακής επαφής που υλοποιείται σε μία ιστοσελίδα. Από την ιστοσελίδα καλείται η διαδικασία, η οποία ταυτοποιεί τον φορολογούμενο χρήστη και τον ενημερώνει για το φόρο που οφείλει, βάσει του τρέχοντος εισοδήματος που εισάγει ο χρήστης και προηγούμενων οφειλών.

Στο επόμενο σχήμα παρουσιάζεται διαγραμματικά η διαδικασία:



Εικόνα 34 Το παράδειγμα εφαρμογής

Η BPEL διαδικασία περιγράφεται στο αρχείο MSThesisDemo_2.bpel. Τα δύο σχετικά wsdl αρχεία είναι τα MSThesisDemo_2.wsdl και MSThesisDemoProp_2.wsdl. Η Web εφαρμογή καλεί τρία SOAP Client php scripts, καθένα από τα οποία υλοποιεί την αντιστοιχεί κλήση προς τη διαδικασία, χρησιμοποιώντας διαφορετικά operation τα οποία αντιστοιχούν σε τρείς διαφορετικές διαδικασίες receive. Η διαδικασία με τη σειρά της καλεί δύο Web Services που ταυτοποιούν το χρήστη, και υπολογίζουν το φόρο βάσει εισοδήματος της τρέχουσας περιόδου και ιστορικών προηγούμενων οφειλών. Τέλος ο χρήστης λαμβάνει το ποσό που οφείλει.

Αναλυτικά οι ενέργειες που πραγματοποιεί ο φορολογούμενος χρήστης, η BPEL διαδικασία και είναι τα παρακάτω:

1. Ο χρήστης εισάγει τα στοιχεία του, δηλαδή όνομα χρήστη, κωδικό πρόσβασης και ΑΦΜ στη Web εφαρμογή.
2. Η Web εφαρμογή καλεί τον SOAP Client που υλοποιεί την καλούσα Login Web Service.
3. Ο SOAP Client καλεί τη διαδικασία και στο SOAP μήνυμα αποστέλλει τα στοιχεία του χρήστη.
4. Η διαδικασία καλεί (δραστηριότητα invoke) τη Web Service που ταυτοποιεί το χρήστη και υλοποιείται στο php script loginServer.php. Σημειώνεται ότι ο SOAP Client που πραγματοποιεί κλήσεις προς τις συνεργαζόμενες Web Services είναι υλοποιημένος στον κώδικα της δραστηριότητας invoke.
5. Η Web Service ταυτοποιεί το χρήστη, ελέγχοντας τα στοιχεία του στη βάση δεδομένων και αν η ταυτοποίηση είναι επιτυχής επιστρέφει ένα μοναδικό κωδικό αναγνώρισης του συγκεκριμένου session.
6. Η διαδικασία, στη περίπτωση επιτυχούς ταυτοποίησης, επιστρέφει στην καλούσα Web Service μέσω της δραστηριότητας reply τον κωδικό και μπαίνει σε κατάσταση wait, αναμένοντας την ενεργοποίηση της δραστηριότητας receive. Σε περίπτωση αδυναμία ταυτοποίησης η διαδικασία τερματίζεται.
7. Η καλούσα Web Service ενημερώνει μέσω της Web εφαρμογής το χρήστη ο οποίος στη συνέχεια εισάγει σε αυτή τα εισοδήματα στα οποία θα υπολογιστεί ο φόρος.
8. Η Web εφαρμογή καλεί τον SOAP Client που υλοποιεί την καλούσα Tax Web Service
9. Ο SOAP Client καλεί τη διαδικασία και στο SOAP μήνυμα αποστέλλει το εισόδημα του χρήστη και το μοναδικό κωδικό αναγνώρισης βάσει του οποίου γίνεται η συσχέτιση για να εντοπισθεί η παρουσία της διαδικασία που είναι σε κατάσταση wait.
10. Η διαδικασία επανέρχεται σε κατάσταση running και καλεί τη Web Service που υπολογίζει το φόρο και υλοποιείται στο php script loginServer.php
11. Η Web Service υπολογίζει το φόρο βάσει του εισοδήματος του χρήστη και τυχόν παλαιότερων οφειλών που βρίσκονται καταχωρημένες στη βάση δεδομένων και επιστρέφει το ποσό της συνολικής οφειλής.

12. Η διαδικασία επιστρέφει στην καλούσα Web Service το ποσό της συνολικής οφειλής στην καλούσα Web Service και τερματίζεται.

Το αρχείο που περιγράφει τη BPEL διαδικασία είναι το παρακάτω:

```
<?xml version="1.0" encoding="UTF-8"?>
<process name="MSThesisNoFlow"
  xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  xmlns:lns="MSThesisDemo"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  suppressJoinFailure="yes"
  targetNamespace="MSThesisDemo">

  <partnerLinks>
    <partnerLink myRole="taxCalculation"
                 name="user"
                 partnerLinkType="lns:userPLT"/>
    <partnerLink name="userLogin"
                 partnerLinkType="lns:userLoginPLT"/>
    <partnerLink name="tacCalc"
                 partnerLinkType="lns:tacCalcPLT"/>
  </partnerLinks>

  <bpws:variables>
    <variable messageType="lns:userMsg" name="userVar"/>
    <variable messageType="lns:userReturnMsg" name="userReturnVar"/>
    <variable messageType="loginStatusMes" name="loginStatusVar"/>
    <variable messageType="userDeclarationMes"
     name="userDeclarationVar"/>
    <variable messageType="taxMes" name="taxVar"/>
    <variable messageType="userPaymentMes" name="userPaymentVar"/>
  </bpws:variables>

  <correlationSets>
    <correlationSet name="userCS"
                    properties="userProp"/>
    <correlationSet name="userDecLoginIdCS"
                    properties="userDecLoginIdProp"/>
  </correlationSets>

  <sequence>
    <receive createInstance="yes"
             operation="userOp"
             partnerLink="customer"
             portType="lns:userPT"
             variable="userVar">
      <correlations>
        <correlation set ="userCS"  initiate ="yes" />
      </correlations>
    </receive>

    <invoke partnerLink="loginPL"
            portType="lns:checkLoginUserPT"
            operation="checLoginObjOp"
            inputVariable="userVar"
            outputVariable="loginStatusVar"/>

    <assign>
      <copy>
        <from part="loginStatus" variable="loginStatusVar"/>
        <to part="userLoginStatus" variable="userReturnVar"/>
      </copy>
```

```
    </assign>

    <reply operation="userOp"
           partnerLink="customer"
           portType="lns:userPT"
           variable="userReturnVar"/>

    <if  xmlns:inventory ="http://supply-chain.org/inventory"
      xmlns:FLT="http://example.com/faults">
      <condition>
        bpel:getVariableProperty('userReturnVar') != 0
      </condition>
      <assign>
        <copy>
          <from part="userAFM" variable="userVar"/>
          <to part="userAFM" variable="userDeclarationVar"/>
        </copy>
      </assign>
      <else>
        <exit/>
      </else>
   </if>

   <assign>
     <copy>
       <from part="userAFM" variable="userVar"/>
       <to part="userAFM" variable="userDeclarationVar"/>
     </copy>
   </assign>

   <assign>
     <copy>
       <from part="loginStatus" variable="loginStatusVar"/>
       <to part="loginId" variable="userDeclarationVar"/>
     </copy>
   </assign>

   <receive createInstance="no"
            operation="userDeclarationObjOp"
            partnerLink="customer"
            portType="lns:userDeclarationPT"
            variable="userDeclarationVar">
     <correlations>
       <correlation set ="userDecLoginIdCS"  initiate ="yes" />
     </correlations>
   </receive>

   <invoke partnerLink="taxPL"
           portType="lns:declarationObjPT"
           operation="declarationObjOp"
           inputVariable="userDeclarationVar"
           outputVariable="taxVar"/>

   <reply operation="userDeclarationObjOp"
          partnerLink="customer"
          portType="lns:userDeclarationPT"
          variable="taxVar"/>

 </sequence>

</process>
```

Το αρχείο WSDL που περιγράφει τις Web Services που εξωτερικεύει τη διαδικασία είναι:

```
<?xml version='1.0' encoding='UTF-8' ?>
<definitions targetNamespace='MSThesisDemo'
  xmlns='http://schemas.xmlsoap.org/wsdl/'
  xmlns:plnk='http://schemas.xmlsoap.org/ws/2003/05/partner-link/'
  xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
  xmlns:tns='MSThesisDemo'
  xmlns:xsd='http://www.w3.org/2001/XMLSchema'>

<!-- complex types for objects -->
  <xsd:complexType name="userType">
    <xsd:all>
      <xsd:element name="userName" type="xsd:string"/>
      <xsd:element name="userPswd" type="xsd:string"/>
      <xsd:element name="UserAFM" type="xsd:string"/>
    </xsd:all>
  </xsd:complexType>
  <xsd:complexType name="declarationType">
    <xsd:all>
      <xsd:element name="userAFM" type="xsd:string"/>
      <part name="loginId" type="xsd:string" />
      <xsd:element name="declaration" type="xsd:string"/>
    </xsd:all>
  </xsd:complexType>

<!-- messages for BPEL WS client call -->
  <message name='userObjMsg'>
    <part name='userObj' type='xsd:userType' />
  </message>
  <message name='userMsg'>
    <part name='userName' type='xsd:string' />
    <part name='userPswd' type='xsd:string' />
    <part name='userAFM' type='xsd:string' />
  </message>

<!-- messages for BPEL WS client response -->
  <message name='userReturnMsg'>
    <part name='userLoginStatus' type='xsd:string' />
  </message>

<!-- messages for WS checking user login -->
  <message name='loginStatusMes'>
    <part name='loginStatus' type='xsd:string' />
  </message>

<!-- messages for BPEL WS client decleration call -->
  <message name='declarationObjMes'>
    <part name='declarationObj' type='xsd:declarationType' />
  </message>
  <message name='userDeclarationMes'>
    <part name='userAFM' type='xsd:string' />
    <part name='loginId' type='xsd:string' />
    <part name='declaration' type='xsd:string' />
  </message>

<!-- messages for WS calculating user tax -->
  <message name='taxMes'>
    <part name='tax' type='xsd:string' />
  </message>

<!-- messages for BPEL WS client decleration response -->
  <message name='userPaymentMes'>
```

```
    <part name='loginId' type='xsd:string' />
    <part name='payment' type='xsd:string' />
  </message>


<!-- port for BPEL WS client call and response -->
  <portType name='userPT'>
    <operation name='userOp'>
      <input message='tns:userMsg'name='userOpInput' />
      <output message='tns:userReturnMsg' name='userReturnOpOutput' />
    </operation>
  </portType>

<!-- port for WS checking user login -->
  <portType name='loginObjPT'>
    <operation name='checLoginObjOp'>
      <input message='tns:userObjMsg' name='userOpInput' />
      <output message='tns:loginStatusMes' name='outputMessage' />
    </operation>
  </portType>

<!-- port for BPEL WS client for user declaration -->
  <portType name='userDeclarationPT'>
    <operation name='userDeclarationObjOp'>
      <input message='tns:userDeclarationMes' name='userDeclarationOpInput'
/>
      <output message='tns:taxMes' name='userPaymentOpOutput' />
    </operation>
   </portType>

<!-- port for WS calculating user tax (OBJ) -->
  <portType name='declarationObjPT'>
    <operation name='declarationObjOp'>
      <input message='tns:declarationObjMes' name='declarationObjOpInput' />
      <output message='tns:taxMes' name='declarationObjOpOutput' />
    </operation>
  </portType>

<!-- binding for BPEL WS client call and response -->
  <binding name='userBnd' type='tns:userPT'>
    <soap:binding style='rpc'
      transport='http://schemas.xmlsoap.org/soap/http'
      xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' />
    <operation name='userOp'>
    <soap:operation soapAction='' style='rpc'
      xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' />
      <input>
        <soap:body encodingStyle='http://schemas.xmlsoap.org/soap/ encoding/'
         use='encoded'
          xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' />
      </input>
      <output>
        <soap:body encodingStyle='http://schemas.xmlsoap.org/soap/ encoding/'
         use='encoded'
         xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' />
      </output>
    </operation>
  </binding>

<!-- binding for WS checking user login -->
  <binding name='loginObjPTBinding' type='tns:loginObjPT'>
    <soap:binding style='rpc'
     transport='http://schemas.xmlsoap.org/soap/http'
```

```
     xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' />
    <operation name='checLoginObjOp'>
      <soap:operation soapAction='' style='rpc'
       xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' />
      <input>
        <soap:body encodingStyle='http://schemas.xmlsoap.org/soap/ encoding/'
          use='encoded' xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' />
      </input>
      <output>
        <soap:body encodingStyle='http://schemas.xmlsoap.org/soap/ encoding/'
          use='encoded' xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' />
      </output>
    </operation>
  </binding>

<!-- binding for BPEL WS client for user declaration -->
  <binding name='userDeclarationPTBinding' type='tns:userDeclarationPT'>
    <soap:binding style='rpc'
     transport='http://schemas.xmlsoap.org/soap/http'
     xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' />
    <operation name='userDeclarationObjOp'>
      <soap:operation soapAction='' style='rpc'
       xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' />
      <input>
        <soap:body encodingStyle='http://schemas.xmlsoap.org/soap/ encoding/'
          use='encoded' xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' />
      </input>
      <output>
        <soap:body encodingStyle='http://schemas.xmlsoap.org/soap/ encoding/'
          use='encoded' xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' />
      </output>
    </operation>
  </binding>

<!-- binding for WS calculating user tax (OBJ) -->
  <binding name='declarationObjBinding' type='tns:declarationObjPT'>
    <soap:binding style='rpc'
     transport='http://schemas.xmlsoap.org/soap/http'
     xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' />
    <operation name='declarationObjOp'>
      <soap:operation soapAction='' style='rpc'
       xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' />
      <input>
        <soap:body encodingStyle='http://schemas.xmlsoap.org/soap/ encoding/'
          use='encoded' xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' />
      </input>
      <output>
        <soap:body encodingStyle='http://schemas.xmlsoap.org/soap/ encoding/'
          use='encoded' xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' />
      </output>
    </operation>
  </binding>

<!-- service for BPEL WS client call and response -->
  <service name='userSer'>
    <port binding='tns:userBnd' name='userPT'>
      <soap:address
       location='http://localhost/MSThesis/v.2.0/bpelServerDemo.php'
       xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'/>
    </port>
  </service>

<!-- service for WS checking user login -->
```

```
  <service name='loginObjSer'>
    <port binding='tns:loginObjPTBinding' name='loginObjPT'>
      <soap:address
       location='http://localhost/MSThesis/v.2.0/WebServices/loginServer.php
       ' xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'/>
    </port>
  </service>

<!-- service for BPEL WS client for user declaration -->
  <service name='userDeclarationSer'>
    <port binding='tns:userDeclarationPTBinding' name='userDeclarationPT'>
      <soap:address
       location='http://localhost/MSThesis/v.2.0/bpelServerDemo.php'
       xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'/>
    </port>
  </service>

<!-- service for WS calculating user tax (OBJ) -->
  <service name='declarationObjSer'>
    <port binding='tns:declarationObjBinding' name='declarationObjPT'>
      <soap:address
       location='http://localhost/MSThesis/v.2.0/WebServices/taxServer.php'
       xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'/>
    </port>
  </service>

</definitions>
```

Το αρχείο WSDL που περιγράφει τα properties είναι:

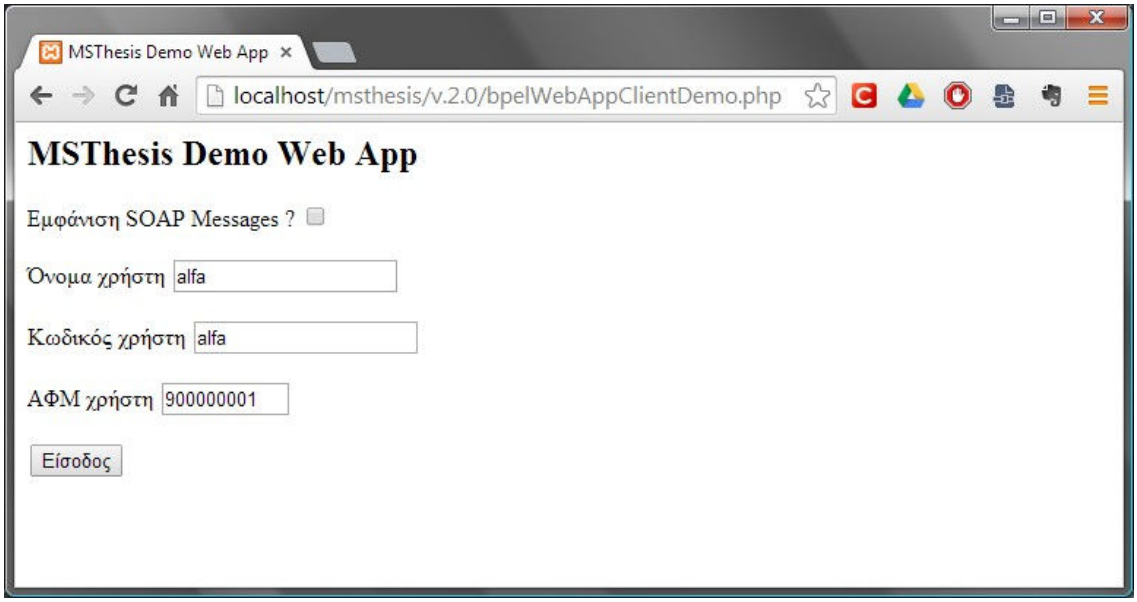
```
<?xml version='1.0' encoding='UTF-8' ?>
<definitions targetNamespace='MSThesisDemo'
       xmlns='http://schemas.xmlsoap.org/wsdl/'
       xmlns:plnk='http://schemas.xmlsoap.org/ws/2003/05/partner-
link/'
       xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
       xmlns:tns='MSThesisDemo'
       xmlns:xsd='http://www.w3.org/2001/XMLSchema'>

   <property name='userProp' />
   <property name='userDecLoginIdProp' />

   <propertyAlias propertyName='userProp'
        messageType='userMsg' part='userAFM' />
   <propertyAlias propertyName='userDecLoginIdProp'
        messageType='userDeclarationMes' part='loginId' />
</definitions>
```

Σημειώνεται ότι όγω της αδυναμίας υπηστήριξης μηνύματα με πολλαπλά <parts> από την επέκταση SOAP της PHP, όταν απαιτήθηκε τα σύνθετα δεδομένα υπέστησαν serialization και μεταδόθηκαν ως ένα αλφαριθμητικό. Προκειμένου όμως να είναι δυνατό αυτό, απαιτήθηκε στο αρχείο WSDL να δηλωθούν οι μεταβλητές αυτών των μηνυμάτων δύο φορές: μία ως xsd:comlexType και μία ως array, όπως φαίνεται παρακάτω:

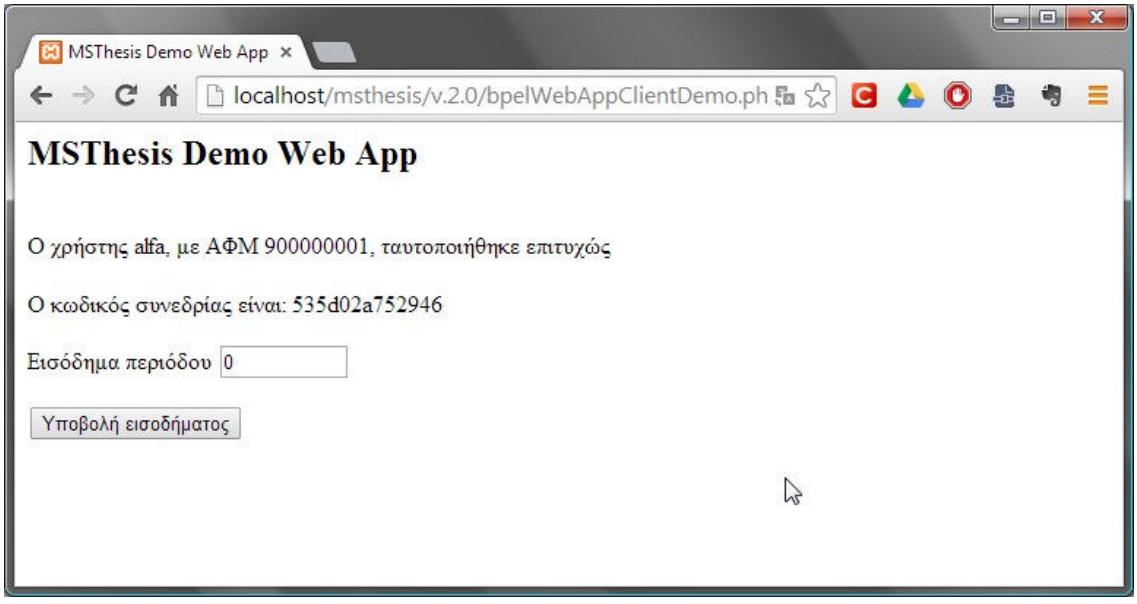
```
  <xsd:complexType name="userType">
    <xsd:all>
      <xsd:element name="userName" type="xsd:string"/>
      <xsd:element name="userPswd" type="xsd:string"/>
      <xsd:element name="UserAFM" type="xsd:string"/>
    </xsd:all>
  </xsd:complexType>
```

```
<message name='userObjMsg'>
  <part name='userObj' type='xsd:userType' />
</message>
<message name='userMsg'>
  <part name='userName' type='xsd:string' />
  <part name='userPswd' type='xsd:string' />
  <part name='userAFM' type='xsd:string' />
</message>
```

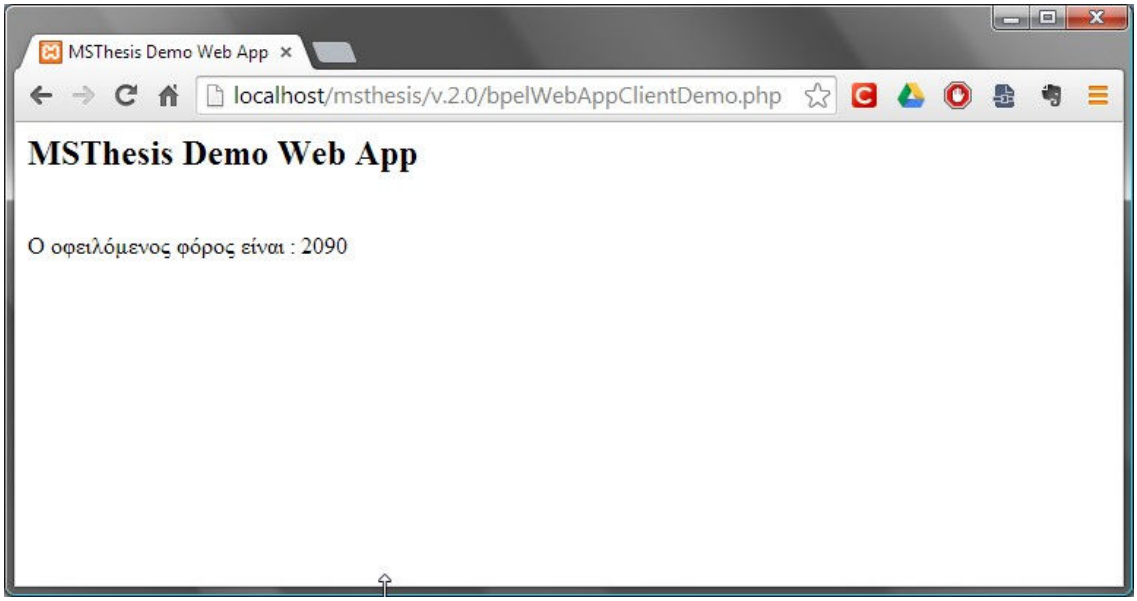
Ενδεικτικά στις επόμενες εικόνες παρουσιάζεται η διεπαφή χρήστη της Web εφαρμογής:



Εικόνα 35 Εισαγωγή στοιχείων ταυτοποίησης χρήστη



Εικόνα 36 Εισαγωγή εισοδήματος



Εικόνα 37 Ενημέρωση για οφειλόμενο ποσό

Ενδεικτικά παρατίθεται το μήνυμα SOAP που καλεί τη διαδικασία:

```
SOAP Request :
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
      xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
 <SOAP-ENV:userOp>
 <userName
xsi:type="xsd:string">a:3:{i:0;s:4:"alfa";i:1;s:4:"alfa";i:2;s:9:"9000
00001";}</userName>
 <userPswd
xsi:nil="true"/>
 <userAFM xsi:nil="true"/>
 </SOAP-ENV:userOp>
 </SOAP-ENV:Body>
 </SOAP-ENV:Envelope>
```

Και το μήνυμα SOAP με το οποί απαντάει η διαδιακσία στη καλούσα Web Service:

```
 SOAP Response :
 <?xml version="1.0" encoding="UTF-8"?>
 <SOAP-ENV:Envelope
       xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
  <SOAP-ENV:userOpResponse>
  <userLoginStatus
 xsi:type="xsd:string">535c3dc740f50</userLoginStatus>
  </SOAP-ENV:userOpResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Συμπεράσματα

1.1 Αξιολόγηση της μηχανής εκτέλεσης BPEL που υλοποιήθηκε

Η μηχανή εκτέλεσης διαδικασιών BPEL που υλοποιήθηκε σε PHP εμφανίζεται να είναι σε θέση να δεχτεί κλήσεις από Web Sevices, να εκτελέσει τις δραστηριότητες που υποστηρίζει (πάντα με τους περιορισμούς της υλοποίησης στο πλαίσιο της μεταπτυχιακής διατριβής), να καλέσει συνεργαζόμενες Wev Services και να επιστρέψει στις καλούσες Wev Services τις απαντήσεις.

Η αρχιτεκτονική υλοποίησης, δηλαδή κώδικας σε PHP για την εκτέλεση της διαδικασίας σε συνδυσμό με MySQL DB για την αποθήκευση δεδομένων και μεταδεδομένων της διαδικασίας, είναι σε θέση να υποστηρίξει διαδικασίες που μπορεί να απαιτούν μακρό χρόνο για να υλοποιηθούν, ή ανθρώπινη παρέμβαση, όπως στο παράδειγμα εφαρμογής.

Είναι απαραίτητο να σημειωθεί ότι, η επιλογή των χαρακτηριστικών του προτύπου WS-BPEL, που θα υποστήριζε η συγκεκριμένη μηχανή, έγινε στο πλαίσιο του βασικού σκοπού της εργασίας, που ήταν η μελέτη αρχιτεκτονικών υπηρεσιο-κεντρικών εφαρμογών και ειδικότερα η το πρότυπο WS-BPEL. Συνεπώς η υποστήριξη εξασφάλισε ένα ελάχιστο επίπεδο λειτουργικότητας των δραστηριοτήτων που υποστηρίζονται, όπως για παράδειγμα της <assign> και δεν καλύπτει το σύνολο της λειτουργικότητας που προβλέπεται στο πρότυπο. Για παράδειγμα όσον αφορά στους τύπος δεδομένων που προβλέπονται στα <messages> δεν υποστηρίζονται xsd: complexTypes. Ο συγκεκριμένος περιορισμός μάλιστα σε συνδυασμό με την αδυναμία υποστήριξης δεδομένων τύπου array σε SOAP μηνύματα από την επέκταση SOAP της PHP που χρησιμοποιήθηκε, δημιούργησε πρόβλημα στο παράδειγμα εφαρμογής, όταν απαιτήθηκε να αποσταλούν μηνύματα που περιείχαν σύνθετα δεδομένα, όπως στη περίπτωση ταυτοποίησης του χρήστη. Ο περιορισμός στο πλαίσιο του παραδείγματος αντιμετωπίσθηκε με serialization των σύνθετων δεδομένων, όμως αυτό είναι ασύμβατο με το πρότυπο.

Επίσης προχωρημένα χαρακτηριστικά, όπως η αντιστάθμιση, επιλέχθηκε να μην υποστηριχθούν, καθότι θα απαιτούσαν μία σαφώς εξαιρετικά πολυπλοκότερη υλοποίηση.

Συνολικά όμως μπορεί να υποστηριχθεί ότι η επιλογή της γλώσσας PHP για την υλοποίηση, παρά τους περιορισμούς που έχει, θα μπορούσε ίσως να αποτελέσει εναλλακτική λύση για μία μηχανή BPEL από την κυρίαρχη τάση χρίσης Java, τουλάχιστον για τον πηρύνα της μηχανής. Επιπρόσθετα χαρακτηριστικά που παρέχουν οι εμπορικές κυρίως υλοποιήσεις, είανι ίσως δυνατόν να ενταχθούν με χρήση διεπαφών μεταξύ PHP και Java ή όποιας άλλης πλατφόρμας προσφέρει καλύτερη και εκτενέστερη λειτουργικότητα. Θα πρέπει να τονιστεί όμως ότι σήμερα η Java και ειδικότερα η Java Platform, Enterpirse Edition παρέχει ένα περιβάλλον που υποστηρίζει την ανάπτυξη και εκτέλεση επιχειρησιακού λογισμικού, περιλαμβανομένων δικτυακών εφαρμογών και Web services, καθώς και άλλων μεγάλου μεθέθους, πολυεπίπεδων (multi-tier), επεκτάσιμων (scalable), αξιόπιστων και ασφαλών δικτυακών εφαρμογών, κάτι που η PHP δεν είναι σε θέση, και ούτε εντάσσεται στους στόχους για τους οποίους αναπτύχθηκε, να υποστηρίξει.

4.2 Προτάσεις για μελλοντικές επεκτάσεις

Η συγκεκριμένη υλοποίηση εκτιμάται ότι δύναται να αποτελέσει τη βάση για μια μηχανή εκτέλεσης διαδικασιών BPEL που θα υποστηρίζει και χαρακτηριστικά του προτύπου όπως, διαχείριση συμβάντων και αντιστάθμιση. Η υποστήριξη της διαχείρισης συμβάντων, μπορεί να βασιστεί στην υπάρχουσα λογική που υποστήριζει τη δραστηριότητα <receive> και τη θέσει της παρουσίας μίας διαδικασίας στις τρεις δυνατές καταστάσεις (idle, running, wating). Η υποστήριξη όμως της αντιστάθμισης θα απαιτήσει ανάπτυξη κώδικα μεγάλου σε μεγεθος και πολυπλοκότητα, καθότι θα απαιτείται να καταγράφονται αναλυτικά οι δραστηριότητες που εκτελούνται και τα αποτελέσματα που επιφέρει η εκτέλεση τους, πιθανότατα σε μία δομή που θα βρίσκεται στη βάση δεδομένων, έτσι ώστε να τηρείται πλήρη ιστορικότητα των ενεργειών που έλεβαν χώρα.

Σημαντική επέκταση θα ήταν η αντιμετώπιση της αδυναμίας υποστήριξης δεδομένων τύπου array σε SOAP μηνύματα από την επέκταση SOAP της PHP, καθότι αυτή είναι ιδιαίτερα κρίσιμη για την επίτευξη συμβατότητας με το πρότυπο. Πιθανές λύσεις θα ήταν ενδεχομένως η αναζήτηση άλλης επέκτασης που θα ικανοποιούσε αυτή την απαίτηση. Η τροποποίηση της συγκεκριμένης επέκτασης θα μπορούσε να είναι επίσης μία εναλλακτική λύση.

Επίσης πρέπει να σημειωθεί ότι, η σύνταξη των αρχείων WSDL και BPEL με το χέρι, χωρίς την υποστήριξη που παρέχει ένα γραφικό περιβάλλον αποδείχθηκε εξαιρετικά επιρεπής σε σφάλματα. Συνεπώς οποιαδήποτε υλοποίηση, για να έχει προστιθέμενη αξία, θα πρέπει να βασίζεται σε ένα γραφικό περιβάλλον που θα απλοποιεί και θα αυτοματοποιεί αντίστοιχα ζητήματα και θα προσφέρει ταυτόχρονα και ένα μηχανισμό ελέγχου της ορθότητας των σχετικών αρχείων.

4.3 Επίλογος

Η συγκεκριμένη διατριβή αποτέλεσε σημαντική άσκηση για τον γράφοντα και παρείχε τη δυνατότητα μελέτης της αρχιτεκτονικής υλοποίησης υπηρεσιο-κεντρικών εφαρμογών και εμβάθυνσης στο πρότυπο WS-BPEL. Ειδικότερα για το τελευταίο, η αντιμετώπιση των προβλημάτων που ανέκυψαν στην υλοποίηση της μηχανής εκτέλεσης επιχειρησιακών διαδικασιών BPEL αποτέλεσε εξαιρετική ευκαιρία για την καλύτερη κατανόηση του, καθότι εξέλειπε η ευκολία που θα παρείχε μία έτοιμη λύση, η οποία θα θα απέκρυπτε λεπτομέριες κάτω από ένα γραφικό περιβάλλον ανάπτυξης, διαχείρισης και εκτέλεσης των διαδικασιών.

Βιβλιογραφία

[1] Anuram Guruge, *Web Services: Theory and Practice*. Elsevier Digital Press, 2004.

[2] Douglas K. Barry, *Web Services and Service-Oriented Architectures - The Savvy Manager's Guide*. Elsevier, 2003.

[3] Lucas Jellema, *Oracle SOA Suite 11g Handbook*. McGraw-Hill, 2010

[4] Matjaz B. Juric, Ramesh Loganathan, Poornachandra Sarang, Frank Jennings, *SOA Approach to Integration*. Packt Publishing, 2007.

[5] Matjaz B. Juric, Marcel Krizevnik, *WS-BPEL 2.0 for SOA Composite Applications with Oracle SOA Suite 11g*. Packt Publishing, 2010

[6] Frank Leymann, Dieter Roller, and Satish Thatte, *Goals of the BPEL4WS Specification*, Διαθέσιμο επιγραμμικά στη διεύθυνση: https://lists.oasis-open.org/archives/wsbpel/200308/doc00000.doc

[7] OASIS, WS-BPEL 2.0 Primer. 2007.

[8] Yuli Vasilief, *SOA and WS-BPEL: Composing Service-Oriented Solutions with PHP and ActiveBPEL.* Pack Publishing, 2007.

[9] Dieter Konig, "Web Service Orchestration and WS-BPEL 2.0", παρουσίαση στο 9th International School on Formal Methods for the Design of Computer, Communication and Software Systems – Web Services – 1-6 Ιουνίου, 2009, Bertinoro

[10] *OASIS Web Services Business Process Execution Language Version 2.0*, OASIS Standard, 2007. Διαθέσιμο επιγραμμικά στη διεύθυνση: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[11] Bjorn Pettersson, An *Introduction to BPEL*. Παρουσίαση διαθέσιμη επιγραμμικά στη διεύθυνση: http://bpmdw.com/library/bpelIntroduction.pdf

[12] Yuli Vasilief, *PHP Oracle Web Development - Data processing, Security, Caching, XML, Web Services, and Ajax*. Publishing, 2007.

[13] Robert Richards, *Pro PHP XML and Web Services*. Apress, 2006.

[14] W3C. *Web Services Activity Statement*. Διαθέσιμο επιγραμμικά στη διεύθυνση: www.w3c.org/2002/ws/Activity

[15] Wikipedia, *Web Services Description Language*. Διαθέσιμο επιγραμμικά στη διεύθυνση: http://en.wikipedia.org/wiki/Web_Services_Description_Language

[16] *W3C Web Services Description Language (WSDL) 1.1*, W3C Standard, 2001

[17] Wikipedia, *Business process*. Διαθέσιμο επιγραμμικά στη διεύθυνση: http://en.wikipedia.org/wiki/Business_process

[18] Wikipedia, *List of BPEL engines*. Διαθέσιμο επιγραμμικά στη διεύθυνση: http://en.wikipedia.org/wiki/List_of_BPEL_engines

[19] *What is Gearman?*. Διαθέσιμο επιγραμμικά στη διεύθυνση: http://gearman.org

[20] *PHP Manual, Function Reference, Web Services, SOAP*. Διαθέσιμο επιγραμμικά στη διεύθυνση: http://www.php.net/manual/en/book.soap.php

[21] *PHP Manual, Function Reference, XML Munipulation, SOAP*. Διαθέσιμο επιγραμμικά στη διεύθυνση: http://www.php.net/manual/en/intro.dom.php

Παραρτήματα

Κώδικας PHP

Το παρακάτα PHP script καλείται από την υπηρεσία πελάτη και χειρίζεται την εκτέλεση της διαδικασίας ακλόντας τις τρεις βασικές κλάσεις.

```
<?php
/**
* This script loads wsdl and calls subsequent scipts that read and construct objects for
 *  wsdl  (class.readWsdl.php),  bpel  (class.execBpel.php)and  finaly execute bpel (class.BpelSec.php)
 *
 */

// comment ob_start(); to enable output in debug
    ob_start();



function  bpelProcessHandler($bpelFile,  $wsdlFile,  $wsdlPropFile, $callingFunction, $serverInputMessage)
{
    date_default_timezone_set("Europe/Athens");
    $bpel = new DOMDocument('1.0', 'ISO-8859-1');
    $bpel->load($bpelFile, LIBXML_NOBLANKS);
    $bpelProcessDom = $bpel->documentElement;
    $wsdl = new DOMDocument('1.0', 'ISO-8859-1');
    $wsdl->load($wsdlFile, LIBXML_NOBLANKS);
    $wsdlProp = new DOMDocument('1.0', 'ISO-8859-1');
    $wsdlProp->load($wsdlPropFile, LIBXML_NOBLANKS);
    $uri = $wsdl->documentElement->lookupnamespaceURI(NULL);
    $wsdlDefinition = $wsdl->documentElement;
    $wsdlPropDefinition = $wsdlProp->documentElement;
    $serverWsdl  =  new  ReadWsdl($wsdlDefinition,  $wsdlPropDefinition, $callingFunction, $serverInputMessage, $wsdl->documentURI, $wsdlFile);
    $serverWsdl->addMessages();
    $serverWsdl->addProperties();
    $serverWsdl->addPropertieAlias();
    $serverWsdl->addPortType();
    $serverWsdl->popInOpMsg2($serverInputMessage);
    $serverWsdl->addBinding();
    $serverWsdl->addService();
    $bpelProcess  =  new  BpelProcess($bpelProcessDom,  $serverWsdl, $serverInputMessage);
    $bpelProcess->addVariables();
    $bpelProcess->addPartnerLinks();
    $bpelProcess->addCorrelationsSets();
```

```
    $bpelProcess->addSequence();
    $bpelProcess->sequence->execSeq();
    $returnValue = $serverWsdl->getOutOpMsg();
    unset($bpelProcess->sequence);
    unset($bpelProcess);
    unset($serverWsdl);
// comment ob_end_clean(); to enable output in debug
    ob_end_clean();
    return $returnValue;
}

?>
```

Βασικές Κλάσεις

Παρακάτω παρατίθεται ο κώδικας των τριών βασικών κλάσεων.

```
<?php

/**
* This script reads the wsdl file and constructs an object for it
*/

class ReadWsdl
{
    public $wsdlId;
    public $definition;
    public $propDefinition;
    public $messages;
    public $parts;
    public $properties;
    public $propertyAlias;
    public $portTypes;
    public $operations;
    public $binding;
    public $service;
    public $serverInputMessage;
    public $callingFunction;
    public $ns;
    public $targetNs;
    public $wsdlDocumentURI;
    public $wsdlFile;


    public function __construct($definition, $propDefinition,
$callingFunction, $serverInputMessage, $wsdlDocumentURI, $wsdlFile)
    {
        update_functionsLogFile(getCaller());
        $this->wsdlDocumentURI = $wsdlDocumentURI;          // com:
the full path to the bpel wsdl file
        $this->wsdlFile = $wsdlFile;                        // com:
wsdl file for DB queries realted to processes
        $this->definition = $definition;
        $this->propDefinition = $propDefinition;
        $this->serverInputMessage = $serverInputMessage;
        $this->ns['tns'] = new NsWsdl();
        $this->ns['tns']->prefix = 'tns';
        $this->ns['tns']->uri = $this->definition-
```

```
>lookupnamespaceURI('tns');
        $this->targetNs = $this->definition-
>getAttribute('targetNamespace');
        $this->callingFunction = $this->targetNs.':'.$callingFunction;
        $this->wsdlId = uniqid();
    }

    function __destruct()
    {
        update_functionsLogFile(getCaller());
    }

    public function addMessages()
    {
        $msgIndex = 0;
        $locMessages = $this->definition-
>getElementsByTagName("message");
        $length = $locMessages->length;
        for ($x = 0; $x < $length; $x++) {
            $locMsgName = $locMessages->item($x)-
>getAttribute('name');
            $qlocMsgName = $this->getPrefix($locMsgName, $this-
>targetNs, $this->ns);
            $this->messages[$qlocMsgName] = new Messages();
            $this->messages[$qlocMsgName]->name = $qlocMsgName;
            $attrName =  $locMessages->item($x)->getAttribute('name');
            if($locMessages->item($x)->hasChildNodes()) {
                $children = $locMessages->item($x)->childNodes;
                foreach ($children as $node) {
                    if($node->nodeName == "part") {
                        $locPartName = $node->getAttribute('name');
                        $qlocPartName = $this->getPrefix($locPartName,
$this->targetNs, $this->ns);
                        $this->parts[$qlocPartName] = new Parts();
                        $this->parts[$qlocPartName]->name =
$qlocPartName;
                        $this->parts[$qlocPartName]->type = $node-
>getAttribute('type');
                    }
                    $this->messages[$qlocMsgName]-
>parts[$qlocPartName] = $this->parts[$qlocPartName];
                    $msgIndex++;
                }
            }
        }
        echoObj($this->messages, __FUNCTION__);
    }

    public function addProperties()
    {
        $propIndex = 0;
        $locProperties = $this->propDefinition-
>getElementsByTagName("property");
        $length = $locProperties->length;
        for ($x = 0; $x < $length; $x++) {
            $locPropName = $locProperties->item($x)-
>getAttribute('name');
            $qlocPropName = $this->getPrefix($locPropName, $this-
>targetNs, $this->ns);
            $this->properties[$qlocPropName] = new Properties();
            $this->properties[$qlocPropName]->name = $qlocPropName;
        }
    }
```

```
    public function addPropertieAlias()
    {
        $propAliasIndex = 0;
        $locPropAlias = $this->propDefinition-
>getElementsByTagName("propertyAlias");
        $length = $locPropAlias->length;
        for ($x = 0; $x < $length; $x++) {
            $locPropAliasName = $locPropAlias->item($x)-
>getAttribute('propertyName');
            $qlocPropAliasName = $this->getPrefix($locPropAliasName,
$this->targetNs, $this->ns);
            $locPropAliasMessage = $locPropAlias->item($x)-
>getAttribute('messageType');
            $qlocPropAliasMessage = $this-
>getPrefix($locPropAliasMessage, $this->targetNs, $this->ns);
            $locPropAliasPart = $locPropAlias->item($x)-
>getAttribute('part');
            $qlocPropAliasPart = $this->getPrefix($locPropAliasPart,
$this->targetNs, $this->ns);
            $this->propertyAlias[$qlocPropAliasName] = new
PropertyAlias();
            $this->propertyAlias[$qlocPropAliasName]->propertyName =
$qlocPropAliasName;
            $this->propertyAlias[$qlocPropAliasName]->messageType =
$qlocPropAliasMessage;
            $this->propertyAlias[$qlocPropAliasName]->part =
$qlocPropAliasPart;
        }
    }

    public function addPortType()
    {
        $opIndex = 0;
        $locPortTypes = $this->definition-
>getElementsByTagName("portType");
        $length = $locPortTypes->length;
        for ($x = 0; $x < $length; $x++) {
            $locPTName = $this->definition-
>getElementsByTagName("portType")->item($x)->getAttribute('name');
            $locPTQName = $this->targetNs.':'.$locPTName;
            $this->portTypes[$locPTQName] = new PortTypes();
            $this->portTypes[$locPTQName]->name = $this->definition-
>getElementsByTagName("portType")->item($x)->getAttribute('name');
            if ($locPortTypes->item($x)->hasChildNodes()) {
                $children = $locPortTypes->item($x)->childNodes;
                foreach ($children as $node) {
                    if($node->nodeName == "operation") {
                        $locOpName = $node->getAttribute('name');
                        $qlocOpName = $this->getPrefix($locOpName,
$this->targetNs, $this->ns);
                        $this->operations[$qlocOpName] = new
Operations();
                        $this->operations[$qlocOpName]->name =
$qlocOpName;
                        $this->operations[$qlocOpName]->inputName =
$node->getElementsByTagName("input")->item(0)->getAttribute("name");
                        $locInMsgName = $node-
>getElementsByTagName("input")->item(0)->getAttribute("message");
                        $qlocInMsgName = $this-
>getPrefix($locInMsgName, $this->targetNs, $this->ns);
                        $this->operations[$qlocOpName]->inputMessage =
$qlocInMsgName;
```

```
                        $this->operations[$qlocOpName]->outputName =
$node->getElementsByTagName("output")->item(0)->getAttribute("name");
                        $locOutMsgName = $node-
>getElementsByTagName("output")->item(0)->getAttribute("message");
                        $qlocOutMsgName = $this-
>getPrefix($locOutMsgName, $this->targetNs, $this->ns);
                        $this->operations[$qlocOpName]->outputMessage
= $qlocOutMsgName;
                    }
                    $this->portTypes[$locPTQName]-
>operations[$qlocOpName] = $this->operations[$qlocOpName]->name;
                    $opIndex++;
                }
            }
        }
    }

    public function popInOpMsg($serverInputMessage)
    {
        $incomingMsg = $this->operations[$this->callingFunction]-
>inputMessage;
        $locParts = $this->messages[$this->operations[$this-
>callingFunction]->inputMessage]->parts;
        foreach($locParts as $inMsg){
            $inMsg->value = $serverInputMessage;
        }
        echoObj($this->messages, __FUNCTION__);
    }

    public function popInOpMsg2($serverInputMessage)
    {
        $incomingMsg = $this->operations[$this->callingFunction]-
>inputMessage;
        $locParts = $this->messages[$this->operations[$this-
>callingFunction]->inputMessage]->parts;
        $msgIndex = 0;
        foreach($locParts as $inMsg){
            if($inMsg->type == "xsd:userType") {
                $inMsg->value = $serverInputMessage;
            }elseif(is_array($serverInputMessage)) {
                $inMsg->value = $serverInputMessage[$msgIndex];
                $msgIndex++;
            }else {
                $inMsg->value = $serverInputMessage;
            }
        }
        echoObj($this->messages, __FUNCTION__);
    }

    public function getOutOpMsg()
    {
        $locParts = $this->messages[$this->operations[$this-
>callingFunction]->outputMessage]->parts;
        foreach($locParts as $outMsg){
            $retValue = $outMsg->value;
        }
        echo "<br/> outMsg->name = " . $outMsg->name;
        echo "<br/> outMsg->value = " . $outMsg->value;
        return $retValue;
    }

    public function addBinding()
    {
```

```
        $locBinding = $this->definition-
>getElementsByTagName("binding");
        $length = $locBinding->length;
        for ($x = 0; $x < $length; $x++) {
            if($locBinding->item($x)->parentNode->isSameNode($this-
>definition)){
                $locBindName = $locBinding->item($x)-
>getAttribute('name');
                $qlocBindName = $this->getPrefix($locBindName, $this-
>targetNs, $this->ns);
                $this->binding[$qlocBindName] = new Binding;
                $this->binding[$qlocBindName]->name = $qlocBindName;
                $locBindType = $locBinding->item($x)-
>getAttribute('type');
                $qlocBindType = $this->getPrefix($locBindType, $this-
>targetNs, $this->ns);
                $this->binding[$qlocBindName]->port[$qlocBindType] =
$this->portTypes[$qlocBindType];
            }
        }
    }

    public function addService()
    {
        $locService = $this->definition-
>getElementsByTagName("service");
        $length = $locService->length;
        for ($x = 0; $x < $length; $x++) {
            $this->service = new Service;
            $locServiceName = $locService->item($x)-
>getAttribute('name');
            $qlocServiceName = $this->getPrefix($locServiceName,
$this->targetNs, $this->ns);
            $this->service->name = $qlocServiceName;
            if($locService->item($x)->hasChildNodes()){
                $children = $locService->item($x)->childNodes;
                foreach ($children as $node) {
                    if($node->nodeName == "port"){
                        // TODO: make services array
                        $locPortName = $node->getAttribute('name');
                        $qlocPortName = $this->getPrefix($locPortName,
$this->targetNs, $this->ns);
                        $this->service->port = new Port;
                        $this->service->port->name = $qlocPortName;
                        $locPBinding = $node->getAttribute('binding');
                        $qlocPBinding = $this->getPrefix($locPBinding,
$this->targetNs, $this->ns);
                        $this->service->port->binding = $this-
>binding[$qlocPBinding];
                    }
                }
            }
        }
    }

    // gets namespace from prefix. If no prefix returns
targetnamespace
    public function getPrefix($name, $targetNs, $ns)
    {
        $namePrefixLen = strcspn($name,':');
        $nameLength = strlen($name);
        if($namePrefixLen == $nameLength){
            $uri = $this->targetNs;
```

```
            $qName = $uri.':'.$name;
        }elseif($namePrefixLen < $nameLength){
            $namePrefix = substr($name,0,$namePrefixLen);
            $uri = $this->ns[$namePrefix]->uri;
            $qName = $uri.':'.substr($name, -($nameLength -
$namePrefixLen - 1));
        }else{
            $uri = '';
            $qName = $name;
        }
        return $qName;
    }

}

class Service
{
    public $name;
    public $port;
}

class Port
{
    public $name;
    public $binding;
}

class Binding
{
    public $name;
    public $port;
}

class Messages
{
    public $name;
    public $parts;
}

class Properties
{
    public $name;
}


class PropertyAlias
{
    public $propertyName;
    public $messageType;
    public $part;
}

class Parts
{
    public $name;
    public $type;
    public $value;
}

class Operations
{
    public $name;
    public $inputName;
```

```
    public $inputMessage;
    public $outputName;
    public $outputMessage;
}

class OperationInputs
{
    public $name;
    public $message;
}

class OperationOutputs
{
    public $name;
    public $message;
}

class PortTypes
{
    public $name;
    public $operations;
}

class NsWsdl
{
    public $prefix;
    public $uri;
}

?>
```

```
<?php

/**
* This script reads the bpel file and constructs an object for it
*/

class BpelProcess
{
    public $bpelId;
    public $process;
    public $variables;
    public $variables2;
    public $partnerLinks;
    public $correlationSets;
    public $sequence;
    public $inPart;
    public $serverWsdl;
    public $serverInputMessage;
    public $ns;
    public $targetNs;
    public $replayMessage;


    function __construct($process, $serverWsdl, $serverInputMessage)
    {
        require_once("classes/class.bpelObjects.php");
        $this->process = $process;
        $this->serverWsdl = $serverWsdl;
        $this->serverInputMessage = $serverInputMessage;
        $this->ns['lns'] = new NsBpel();
        $this->ns['lns']->prefix = 'lns';
```

```
        $this->ns['lns']->uri = $this->process-
>lookupnamespaceURI('lns');
        $this->targetNs = $this->process-
>getAttribute('targetNamespace');
        $this->bpelId = uniqid();
    }

    function __destruct()
    {

    }

    public function addVariables()
    {
        $locVariables = $this->process-
>getElementsByTagName("variables");
        $length = $locVariables->length;
        for ($x = 0; $x < $length; $x++) {
            if ($locVariables->item($x)->hasChildNodes()) {
                $children = $locVariables->item($x)->childNodes;
                foreach ($children as $node) {
                    if($node->nodeName == "variable") {
                        $locVarName = $node->getAttribute('name');
                        $qlocVarName = $this->getPrefix($locVarName);
                        $this->variables[$qlocVarName] = new
Variables();
                        $this->variables[$qlocVarName]->name =
$qlocVarName;
                        $locMsgType = $node-
>getAttribute('messageType');
                        $qlocMsgType = $this->getPrefix($locMsgType);
                        $this->variables[$qlocVarName]->messageType =
$qlocMsgType;
                        $this->variables[$qlocVarName]->varParts =
$this->serverWsdl->messages[$qlocMsgType]->parts;

                    }
                }
            }
        }
    }

    public function addVariables2()
    {
        $locVariables = $this->process-
>getElementsByTagName("variables");
        $length = $locVariables->length;
        for ($x = 0; $x < $length; $x++) {
            if ($locVariables->item($x)->hasChildNodes()) {
                $children = $locVariables->item($x)->childNodes;
                foreach ($children as $node) {
                    if($node->nodeName == "variable") {
                        $locVarName = $node->getAttribute('name');
                        $qlocVarName = $this->getPrefix($locVarName);
                        $this->variables2[$qlocVarName] = new
Variables2();
                        $this->variables2[$qlocVarName]->name =
$qlocVarName;
                        $locMsgType = $node-
>getAttribute('messageType');
                        $qlocMsgType = $this->getPrefix($locMsgType);
                        $this->variables2[$qlocVarName]->messageType =
$qlocMsgType;
```

```
                        $this->variables2[$qlocVarName]->varParts =
$this->serverWsdl->messages[$qlocMsgType]->parts;
                    }
                }
            }
        }
    }

    public function addPartnerLinks()
    {
        $locPartnerLinks = $this->process-
>getElementsByTagName("partnerLinks");
        $length = $locPartnerLinks->length;
        for ($x = 0; $x < $length; $x++) {
            if ($locPartnerLinks->item($x)->hasChildNodes()) {
                $children = $locPartnerLinks->item($x)->childNodes;
                foreach ($children as $node) {
                    $attrName = $node->getAttribute("name");
                    $this->partnerLinks[$attrName]["name"] = $node-
>getAttribute("name");
                    $this->partnerLinks[$attrName]["myRole"] = $node-
>getAttribute("myRole");
                    $this->partnerLinks[$attrName]["partnerLinkType"]
= $node->getAttribute("partnerLinkType");
                    $this->partnerLinks[$attrName]["partnerRole"] =
$node->getAttribute("partnerRole");
                }
            }
        }
    }

    public function addCorrelationsSets()
    {
        $locCorrelationSets = $this->process-
>getElementsByTagName("correlationSets");
        $length = $locCorrelationSets->length;
        for ($x = 0; $x < $length; $x++) {
            if ($locCorrelationSets->item($x)->hasChildNodes()) {
                $children = $locCorrelationSets->item($x)->childNodes;
                foreach ($children as $node) {
                    if($node->nodeName == "correlationSet") {
                        $locCorSetName = $node->getAttribute('name');
                        $qlocCorSetName = $this-
>getPrefix($locCorSetName);
                        $this->correlationSets[$qlocCorSetName] = new
CorrelationSet();
                        $this->correlationSets[$qlocCorSetName]->name
= $qlocCorSetName;
                        $locProperty = $node-
>getAttribute('properties');
                        $qlocProperty = $this-
>getPrefix($locProperty);
                        $this->correlationSets[$qlocCorSetName]-
>properties = $qlocProperty;
                    }
                }
            }
        }
    }

    public function addSequence()
    {
        $locSequences = $this->process-
```

```
>getElementsByTagName("sequence");
        $length = $locSequences->length;
        for ($x = 0; $x < $length; $x++) {
            if ($locSequences->item($x)->parentNode->isSameNode($this-
>process)) {
                $this->sequence = new Sequence($this->bpelId, $this-
>targetNs, $this->ns, $this->variables, $this->correlationSets, $this-
>serverWsdl, $locSequences->item($x));
                $xml =$this->sequence->seq->ownerDocument-
>saveXML($this->sequence->seq);
                $dom = new DOMDocument;
                $rootSequence = new FreezableDOMElement("sequence");
                $dom->appendChild($rootSequence);
                $dom->loadXML($xml);
                $rootSequence = $dom->documentElement;
                $newxml =$rootSequence->ownerDocument-
>saveXML($rootSequence);
                if($xml == $newxml)
                {
                    $succesXml = 1;
                }
            } elseif (!$locSequences->item($x)->parentNode-
>isSameNode($this->process)) {

            }
        }
        $props = get_object_vars($this->sequence);
    }

    public function getPrefix($name)
    {
        $namePrefixLen = strcspn($name,':');
        $nameLength = strlen($name);
        if($namePrefixLen == $nameLength){
            $uri = $this->targetNs;
            $qName = $uri.':'.$name;
        }elseif($namePrefixLen < $nameLength){
            $namePrefix = substr($name,0,$namePrefixLen);
            $uri = $this->ns[$namePrefix]->uri;
            $qName = $uri.':'.substr($name, -($nameLength -
$namePrefixLen - 1));
        }else{
            $uri = '';
            $qName = $name;
        }
        return $qName;
    }

}

?>
```

```
<?php

/**
* This script executes a sequence in the bpel file
*/

class Sequence
{
    public $bpelId;
    public $seqId;
    public $seq;
```

```
    public $targetNs;
    public $ns;
    public $variables;
    public $correlationSets;
    public $recData;
    public $serverWsdl;
    public $fh;
    public $asynch;
    public $bpelAcivitiesArray;

    function __construct($bpelId, $targetNs, $ns, $variables,
$correlationSets, $serverWsdl, $seq)
    {
        $this->bpelId = $bpelId;
        $this->targetNs = $targetNs;
        $this->ns = $ns;
        $this->variables = $variables;
        $this->correlationSets = $correlationSets;
        $this->serverWsdl = $serverWsdl;
        $this->seq = $seq;
        $this->arraySeq = dom2array($seq);                  // TODO:
check if necessary
        $this->serArraySeq = serialize($this->arraySeq);
        $this->seqId = uniqid();
        echoSeqCreation($this->seqId);
        $this->asynch = 0;
        $this->bpelAcivitiesArray = array('invoke', 'receive',
'reply', 'assign', 'flow', 'throw', 'exit', 'wait', 'empty',
'sequence', 'while', 'pick', 'scope', 'compensate');
        $myFile = "../logs/sequenceLog.txt";
    }

    function __destruct()
    {

    }

    function execSeq()
    {
        $seqNode = $this->seq;
        $execNextNode = 0;
        if ($seqNode->hasChildNodes()) {
            $children = $seqNode->childNodes;
            foreach($children as $node) {
                if($node->nodeName == 'receive') {
                    if(is_array($this->serverWsdl-
>serverInputMessage)) {
                        $locServerInputMessage = serialize($this-
>serverWsdl->serverInputMessage);
                    } else {
                        $locServerInputMessage = $this->serverWsdl-
>serverInputMessage;
                    }
                    $this->receiveExctractData($node);      // com:
gets receive node from bpel
                    $locVar = $this->variables[$this-
>recData['qrecVariable']]->value;
                    $dbConn = mysqliConnect();
                    $processId = getRunningProcess($dbConn, $this-
>serverWsdl->wsdlFile);
                    if($processId == FALSE) {
                        return;
                    }
```

```
                    $recProp = $this->correlationSets[$this-
>recData['qrecCorrelation']]->properties;
                    $recCorrMess = $this->serverWsdl-
>propertyAlias[$recProp]->messageType;
                    $recCorrPart = $this->serverWsdl-
>propertyAlias[$recProp]->part;
                    if($recProp <> NULL) {
                        foreach($this->variables as $locVar) {
                            if($locVar->messageType == $recCorrMess) {
                                $locCorVar = new Variables();
                                $locCorVar = $locVar;
                                $locCorVarParts = $locCorVar-
>varParts;
                                $locCorVarPart =
$locCorVarParts[$recCorrPart];
                            }
                        }
                    }  else {
                        $locCorVar = NULL;
                        $locCorVarParts = NULL;
                        $locCorVarPart = NULL;
                    }
                    $processState = getProcessStatusSimple($dbConn,
$processId);
                    if($processState['processStatus'] == 0 or
$processState['processStatus'] == NULL) {
                        if($this->recData['recCreateInstance'] ==
"no") {           // CI? NO, next node
                        } elseif($this->recData['recCreateInstance']
== "yes") {    // CI? YES, instance will be created
                            if($this->serverWsdl->callingFunction ==
$this->recData['qrecOperation']) {
                                $execNextNode = 1;
                            } else {
                                $execNextNode = 0;
                            }
                        }
                    } elseif($processState['processStatus'] == 1) {
// RUNNING
                        if($this->recData['recCreateInstance'] ==
"no") {           // CI? NO, set to wait
                            $locVar = $this->variables[$this-
>recData['qrecVariable']]->value;
                            storeProcessWaitStatus($dbConn,
$processId, $this->bpelId, $this->recData['qrecPortType'], $this-
>recData['qrecOperation'], $locCorVarPart->value, serialize($this-
>variables));
                            $execNextNode = 0;
                            return;
                        } elseif($this->recData['recCreateInstance']
== "yes") {    // CI? YES
                            if($this->recData['qrecCorrelation'] ==
NULL) {         // COR? NULL, create instance
                                if($this->serverWsdl->callingFunction
==    $this->recData['qrecOperation']) {
                                    $execNextNode = 1;
                                } else {
                                    $execNextNode = 0;
                                }
                            } else {
                                $processStatus =
getProcessStatus($dbConn, $processId, $this->recData['qrecPortType'],
$locCorVarPart->value);
```

```
                                    if($processStatus['processStatus'] ==
NULL) {   // COR? NO, create instance
                                        if($this->serverWsdl-
>callingFunction ==    $this->recData['qrecOperation']) {
                                            $execNextNode = 1;
                                        } else {
                                            $execNextNode = 0;
                                        }
                                    } else {
// COR? YES, next node
                                    }
                                }
                            }
                        } elseif($processState['processStatus'] == 2) {
// WAITING
                            if($this->recData['recCreateInstance'] ==
"no") {       // CI? NO
                                $processStatus = getProcessStatus($dbConn,
$processId, $this->recData['qrecPortType'], $locCorVarPart->value);
                                if($this->recData['qrecCorrelation'] ==
NULL) {        // COR? NULL, wake up
                                    if($this->serverWsdl->callingFunction
== $this->recData['qrecOperation']) {
                                        $this->bpelId =
$processStatus['processBpelId'];
                                        $tempVariables = new Variables;
                                        $tempVariables =
unserialize(getVariables($dbConn, $this->bpelId ));
                                        $execNextNode = 1;
                                    } else {
                                        $execNextNode = 0;
                                    }
                                } else {
// COR? NOT NULL
                                    if($processStatus['processStatus'] ==
NULL) {   // COR? NO, next node
                                    } else {
// COR? YES, wake up
                                        echo $style . "Process is waiting"
. $endStyle;
                                        if($this->serverWsdl-
>callingFunction == $this->recData['qrecOperation']) {
                                            $this->bpelId =
$processStatus['processBpelId'];
                                            $tempVariables =
unserialize(getVariables($dbConn, $this->bpelId ));
                                            foreach($this->variables as
$locVar) {
                                                if($locVar->messageType ==
$recCorrMess) {
                                                    $tempVar = new
Variables();
                                                    $tempVar = $locVar;
                                                    $tempVarName =
$locVar->name;
                                                }
                                            }
                                            $this->variables =
$tempVariables;
                                            $this->variables[$tempVarName]
= $tempVar;
                                            $execNextNode = 1;
                                        } else {
```

```
                                        $execNextNode = 0;
                                    }
                                }
                            }
                        } elseif($this->recData['recCreateInstance']
== "yes") { // CI? YES, next node

                        }
                    }
                }       //END of if($node->nodeName == 'receive')
                if($execNextNode == 1) {
                    $this->bpelSeqNodes($node);
                } elseif($execNextNode == 0) {
                }
            }           // END of foreach($children as $node)
        }               // END of if ($seqNode->hasChildNodes())

        $seqNode = $this->seq;                                  // com:
check each node of the (DOMElement) sequence
        if ($seqNode->hasChildNodes()) {
            $children = $seqNode->childNodes;
        }
        $serVariables = getObjRaw($dbConn, $this->seqId . "_var");
        removeProcessFromDb($dbConn, $this->bpelId);
        mysqliClose($dbConn);
    }

    function bpelSeqNodes($seqNode)
    {
        switch($seqNode->nodeName){
            case 'receive':
                $this->bpelReceive($seqNode);
                break;
            case 'invoke':
                $this->bpelInvoke($seqNode);
                break;
            case 'reply':
                $this->bpelReply($seqNode);
                break;
            case 'assign':;
                $this->bpelAssign($seqNode);
                break;
            case 'flow':
                $this->bpelFlow($seqNode);
                break;
            case 'if':
                $this->bpelIf($seqNode);
                break;
            case 'exit':;
                $this->bpelExit($seqNode);
                break;
            case 'wait':
                break;
            case 'sequence':
                break;
        }
    }

    function bpelReceive($receiveNode)
    {
        $this->receiveGetIputMessage($receiveNode);
        $recProp = $this->correlationSets[$this-
>recData['qrecCorrelation']]->properties;
```

```
        $recCorrMess = $this->serverWsdl->propertyAlias[$recProp]-
>messageType;
        $recCorrPart = $this->serverWsdl->propertyAlias[$recProp]-
>part;
        foreach($this->variables as $locVar) {
            if($locVar->messageType == $recCorrMess) {
                $callingMessage = new Variables();
                $callingMessage = $locVar;
            }
        }
        $corrValue = $callingMessage->value;
        $locProperties = $this->serverWsdl->properties;
        $dbConn = mysqliConnect();
        $processId = getRunningProcess($dbConn, $this->serverWsdl-
>wsdlFile);
        $processStatus = getProcessStatus($dbConn, $processId, $this-
>recData['qrecPortType'], $corrValue);
        if($processStatus['processStatus'] == "" and $this-
>recData['recCreateInstance'] == "yes" and $this-
>recData['qrecOperation'] == $this->serverWsdl->callingFunction) {
                storeProcessRunStatus($dbConn, $processId, $this-
>bpelId, $this->recData['qrecPortType'], $this-
>recData['qrecOperation'], $corrValue);
        } else {
            $processStatus = getProcessStatusFromBpelId($dbConn,
$processId, $this->bpelId);
            if($processStatus['processStatus'] == 1) {
                storeProcessWaitStatus($dbConn, $processId, $this-
>bpelId, $qrecPortType, $qrecOperation, $corrValue);
            }
        }
    }

    function receiveExctractData($receiveNode)
    {
        $receiveData['recCreateInstance'] = $receiveNode-
>getAttribute('createInstance');
        $recOperation = $receiveNode->getAttribute('operation');
        $receiveData['qrecOperation'] = $this-
>getPrefix($recOperation);
        $recPortType = $receiveNode->getAttribute('portType');
        $receiveData['qrecPortType'] = $this->getPrefix($recPortType);
        $recVariable = $receiveNode->getAttribute('variable');
        $receiveData['qrecVariable'] = $this->getPrefix($recVariable);
        $recVarMessageName = $this-
>variables[$receiveData['qrecVariable']];
        $recVarMessageType = $recVarMessageName->messageType;
        $locParts = $this->serverWsdl->messages[$recVarMessageType]-
>parts;
        $recCorrSets = $receiveNode-
>getElementsByTagName('correlations');
        $length = $recCorrSets->length;
        for ($x = 0; $x < $length; $x++) {
            $recCorrSetsCorrelation = $recCorrSets->item($x)-
>getElementsByTagName('correlation');
            $lengthInt = $recCorrSetsCorrelation->length;
            for ($y = 0; $y < $lengthInt; $y++) {
                $recCorrelation = $recCorrSetsCorrelation->item($x)-
>getAttribute('set');
                $receiveData['qrecCorrelation'] = $this-
>getPrefix($recCorrelation);
            }
        }
```

```
        $this->recData = $receiveData;
    }

    function receiveGetIputMessage_OLD($receiveNode)
    {
        $recVariable = $receiveNode->getAttribute('variable');
        $receiveData['qrecVariable'] = $this->getPrefix($recVariable);
        $recVarMessageName = $this-
>variables[$receiveData['qrecVariable']];
        $recVarMessageType = $recVarMessageName->messageType;
        $locParts = $this->serverWsdl->messages[$recVarMessageType]-
>parts;
        foreach($locParts as $recMsg){
            $this->variables[$receiveData['qrecVariable']]->value =
$recMsg->value;
        }
        echoVarValues($this->variables);
    }

    function receiveGetIputMessage($receiveNode)
    {
        $recVariable = $receiveNode->getAttribute('variable');
        $receiveData['qrecVariable'] = $this->getPrefix($recVariable);
        $recVarMessageName = $this-
>variables[$receiveData['qrecVariable']];
        $recVarMessageType = $recVarMessageName->messageType;
        $locParts = $this->serverWsdl->messages[$recVarMessageType]-
>parts;
        if(is_array($this->serverWsdl->serverInputMessage)) {
            $locServerInputMessage = serialize($this->serverWsdl-
>serverInputMessage);
        } else {
            $locServerInputMessage = $this->serverWsdl-
>serverInputMessage;
        }
        foreach($locParts as $recMsg){
            $this->variables[$receiveData['qrecVariable']]->value =
$locServerInputMessage;
        }
    }

    function bpelInvoke($invokeNode)
    {
        $invOperation = $invokeNode->getAttribute('operation');
        $qinvOperation = $this->getPrefix($invOperation);
        $invPortType = $invokeNode->getAttribute('portType');
        $qinvPortType = $this->getPrefix($invPortType);
        $invInputVariable = $invokeNode-
>getAttribute('inputVariable');
        $qinvInputVariable = $this->getPrefix($invInputVariable);
        $invVarMessageName = $this->variables[$qinvInputVariable];
        $invVarMessageType = $invVarMessageName->messageType;
        $locVarParts = $this->variables[$qinvInputVariable]->varParts;
        if(count($locVarParts) == 1) {
            $invokeVar = current($locVarParts)->value;
        } else {
            foreach($locVarParts as $parts) {
                $invokeVar[$parts->name] = $parts->value;
            }
        }
        $WSCallResponce = $this->execInvoke($invokeVar,
$invOperation);
        $invOutputVariable = $invokeNode-
```

```
>getAttribute('outputVariable');
        $qinvOutputVariable = $this->getPrefix($invOutputVariable);
        $this->variables[$qinvOutputVariable]->value =
$WSCallResponce;
        $locVarParts = $this->variables[$qinvOutputVariable]-
>varParts;
        if(is_array($WSCallResponce)) {
            reset($WSCallResponce);
            foreach($locVarParts as $varPart) {
                $varPart->value = current($WSCallResponce);
                next($WSCallResponce);
            }
        } else {
             foreach($locVarParts as $varPart) {
                $varPart->value = $WSCallResponce;
             }
        }
    }

    public function execInvoke($wsInMessage, $invOperation)
    {
        $wsdl = $this->serverWsdl->wsdlDocumentURI;
        ini_set("soap.wsdl_cache_enabled", "0");
        $WSclient = new SoapClient($wsdl, array('trace' => 1));
        $request=$wsInMessage;
        try {
            $response = $WSclient->$invOperation($request);
        }catch(SoapFault $fault) {
            echo 'Request : <br/><xmp>',
            $WSclient->__getLastRequest(),
            '</xmp><br/><br/> Error Message : <br/>',
            $fault->getMessage();
        }
        displaySoapReqResp($WSclient, $request, $response);
        return $response;
    }

    function bpelExit($exitNode)
    {
        $dbConn = mysqliConnect();
        removeProcessFromDb($dbConn, $this->bpelId);
        exit();
    }

    function bpelReply($replyNode)
    {
        $repOperation = $replyNode->getAttribute('operation');
        $qrepOperation = $this->getPrefix($repOperation);
        $repPortType = $replyNode->getAttribute('portType');
        $qrepPortType = $this->getPrefix($repPortType);
        $repVariable = $replyNode->getAttribute('variable');
        $qrepVariable = $this->getPrefix($repVariable);
        $this->replayMessage = $this->variables[$qrepVariable]->value;
        $locParts = $this->serverWsdl->messages[$this-
>variables[$qrepVariable]->messageType]->parts;
        foreach($locParts as $replayPart){
            $replayPart->value = $this->variables[$qrepVariable]-
>value;
        }

    }

    function bpelAssign($assignNode)
```

```
    {
        update_functionsLogFile(getCaller(), $this->asynch);
        if($assignNode->hasChildNodes()){
            $assignElements = $assignNode-
>getElementsByTagName("copy");
            $length = $assignElements->length;
            for ($x=0;$x < $length;$x++) {
                $currentCopy = $assignElements->item($x);
                if($currentCopy->hasChildNodes()){
                    $children = $currentCopy->childNodes;
                    foreach($children as $node) {
                        if($node->nodeName == 'from'){
                            if($node->getAttribute('variable') &&
$node->getAttribute('part')){
                                $fromVariable = $node-
>getAttribute('variable');
                                $qfromVariable = $this-
>getPrefix($fromVariable);
                                $fromPart = $node-
>getAttribute('part');
                                $qfromPart = $this-
>getPrefix($fromPart);
                                $fromVarValue = $this-
>variables[$qfromVariable]->value;
                                $fromPartValue = $this-
>variables[$qfromVariable]->varParts[$qfromPart]->value;
                                echo "<br/>" . $fromPartValue .
"<br/>";
                            }elseif($node->getAttribute('partnerLink')
&& $node->getAttribute('endpoinReference')){

                            }elseif($node->getAttribute('variable') &&
$node->getAttribute('property')){

                            }elseif($node-
>getAttribute('expression')){

                            }elseif($node->getAttribute('literal')){

                            }
                        }
                        if($node->nodeName == 'to'){
                            if($node->getAttribute('variable') &&
$node->getAttribute('part')){
                                $toVariable = $node-
>getAttribute('variable');
                                $qtoVariable = $this-
>getPrefix($toVariable);
                                $toPart = $node->getAttribute('part');
                                $qtoPart = $this->getPrefix($toPart);
                                //$this->variables[$qtoVariable]-
>value = $fromVarValue;
                                $this->variables[$qtoVariable]-
>varParts[$qtoPart]->value = $fromPartValue;
                                $this->variables[$qtoVariable]->value
= $this->calcVarValue($this->variables[$qtoVariable]);
                                echo "<pre>";
                                print_r($this-
>variables[$qtoVariable]->value);
                                echo "</pre>";
                            }elseif($node->getAttribute('partnerLink')
&& $node->getAttribute('endpoinReference')){
```

```
                            }elseif($node->getAttribute('variable') &&
$node->getAttribute('property')){

                            }elseif($node-
>getAttribute('expression')){

                            }elseif($node->getAttribute('literal')){

                            }
                        }
                    }
                }
            }
        }

    function bpelIf($ifNode)
    {
        $ifCondition = $ifNode->getAttribute('condition');
        $qifCondition = $this->getPrefix($ifCondition);
        if($ifNode->hasChildNodes()){
            $ifElements = $ifNode->childNodes;
            $ifElement = $ifElements->item(0);
            if($ifElement->nodeName == 'condition') {
                if($ifElement->nodeName == 'condition'){
                    if(strpos($ifElement->nodeValue,
"bpel:getVariableProperty") !== FALSE){
                        if($this->checkCondition($ifElement) == TRUE){
                            $this-
>execConditionalActivity($ifElements);
                        } else {
                            $endOfIf = 0;
                            for ($x=1; $x<=$ifElements->length; $x++)
{
                                if($endOfIf == 0){
                                    $node = $ifElements->item($x);
                                    if($node->nodeName == 'elseif'){
                                        if($this-
>checkElseifCondition($node) == TRUE){
                                            $elseifElements = $node-
>childNodes;
                                            for ($x=1;
$x<=$elseifElements->length; $x++) {
                                                $node =
$elseifElements->item($x);
                                                if (in_array($node-
>nodeName, $this->bpelAcivitiesArray)){
                                                    $this-
>bpelSeqNodes($node);
                                                }
                                            }
                                        }
                                    } elseif($node->nodeName ==
'else') {
                                            $elseElements = $node-
>childNodes;
                                            for ($x=0; $x<=$elseElements-
>length; $x++) {
                                                $node = $elseElements-
>item($x);
                                                if (in_array($node-
>nodeName, $this->bpelAcivitiesArray)){
                                                    $this-
```

```
>bpelSeqNodes($node);
                                                }
                                            }
                                            $endOfIf = 1;
                                        }
                                    }
                                }
                            }
                        }
                    }else {
                        echo "nope!";
                    }
                }
            }
        }

    function bpelFlow($flowNode)
    {
        $timeFlowStart = microtime_float();
        $flowSequence = $flowNode->getElementsByTagName("sequence");
        $length = $flowSequence->length;
        for ($x = 0; $x < $length; $x++) {
                $flowSeqObj = new Sequence($this->targetNs, $this->ns,
$this->variables, $this->serverWsdl, $flowSequence->item($x));
                $serFlowSeqObj = serialize($flowSeqObj);
                $varObj = $flowSeqObj->variables;
                $serVarObj = serialize($varObj);

                $serThis = serialize($this);
                $dbConn = mysqliConnect();
                saveObj($dbConn, $this->seqId, $serThis);
                mysqliClose($dbConn);

                $dbConn = mysqliConnect();
                saveFlowSerSeq($dbConn, $flowSeqObj->seqId,
$flowSeqObj->serArraySeq, basename(__FILE__));
                mysqliClose($dbConn);

                $dbConn = mysqliConnect();
                saveObj($dbConn, $flowSeqObj->seqId, $serFlowSeqObj);
                saveObj($dbConn, $flowSeqObj->seqId . "_var",
$serVarObj);
                mysqliClose($dbConn);

                $value2pass = $flowSeqObj->seqId;
                http_request('GET', '127.0.0.1', 80,
'/MSThesis/v.1.0/includes/asynchHTTPhandlerNewSimple.php',
array('serSeqId' => $value2pass));

            $flowSeqObj->execSeq();
            checkEndOfFlow($flowSeqObj);
            echoVarValues($this->variables);
            unset($flowSeqObj);
        }
    }

    function execSeq2()                                          // com:
executes the flow sequence
    {
        $this->asynch = 1;
        $seqNode = $this->seq;                                  // com:
check each node of the (DOMElement) sequence
        if ($seqNode->hasChildNodes()) {
```

```
            $children = $seqNode->childNodes;
            foreach($children as $node) {
                $this->bpelSeqNodes($node);                 // com:
call to function from execBpel.php (what do I meant to say?)
            }
        }

            $dbConn = mysqliConnect();
            $varObj = $this->variables;
            $serVarObj = serialize($varObj);
            updateObj($dbConn, $this->seqId . "_var", $serVarObj);

            $serVariables = getObjRaw($dbConn, $this->seqId . "_var");
            $phpScriptFunction = basename(__FILE__) . ", function: " .
__FUNCTION__ . ", line: " . __LINE__;
            log_variables_2($this, $serVariables, $phpScriptFunction);

        seqEnd($dbConn, $this->seqId, basename(__FILE__));
        mysqliClose($dbConn);
    }

    public function checkCondition($ifElement)
    {
                        $pos1 = strpos($ifElement->nodeValue, "'");
                        $conditionVariable = substr($ifElement-
>nodeValue, $pos1+1);
                        $pos2 = strpos($conditionVariable, "'");
                        $conditionVariable =
substr($conditionVariable, 0, $pos2);
                        $pos3 = strpos($ifElement->nodeValue, ")");
                        $condition = trim(substr($ifElement-
>nodeValue, $pos3+1));
                        $str1 = substr($condition, 0, 1);
                        $str2 = trim(substr($condition, 1, 1));
                        if($str2 == "="){
                            $conditionSign = $str1 . $str2;
                        } else {
                            $conditionSign = $str1;
                        }
                        $conditionValue = substr($condition,
strlen($conditionSign));
                        $variableValue = $this-
>getVariableValue($conditionVariable);
                        if($conditionSign == "="){
                            if($variableValue == $conditionValue) {
                                return TRUE;
                            }
                        } elseif($conditionSign == "<"){
                            if($variableValue < $conditionValue) {
                                return TRUE;
                            }
                        } elseif($conditionSign == ">"){
                            if($variableValue > $conditionValue) {
                                return TRUE;
                            }
                        } elseif($conditionSign == "<="){
                            if($variableValue <= $conditionValue) {
                                return TRUE;
                            }
                        } elseif($conditionSign == ">="){
                            if($variableValue >= $conditionValue) {
                                return TRUE;
                            }
```

```
                        } elseif($conditionSign == "!="){
                            if($variableValue <> $conditionValue) {
                                return TRUE;
                            }
                        } else {
                            return FALSE;
                        }
    }

    public function execConditionalActivity($ifElements)
    {
            for ($x=1; $x<=$ifElements->length; $x++) {
                $node = $ifElements->item($x);
                if (in_array($node->nodeName, $this-
>bpelAcivitiesArray)){
                    echo "<br/> BPEL ACTIVITY : " . $node->nodeName .
" found <br/>";
                    $this->bpelSeqNodes($node);
                } elseif ($node->nodeName == 'elseif'){
                    echo "<br/> -- ELSEIF FOUND -- <br/>";
                } elseif ($node->nodeName == 'else'){
                    echo "<br/> -- ELSE FOUND -- <br/>";
                }
            }
    }

    public function checkElseifCondition($node)
    {
        $elseifElements = $node->childNodes;
        $elseifElement = $elseifElements->item(0);
        if($elseifElement->nodeName == 'condition') {
            if(strpos($elseifElement->nodeValue,
"bpel:getVariableProperty") !== FALSE){
                if($this->checkCondition($elseifElement) == TRUE){
                    return TRUE;
                } else {
                    return FALSE;
                }
            } else {
                echo "noelse if ";
            }
        }
    }

    public function getPrefix($name)
    {
        $namePrefixLen = strcspn($name,':');
        $nameLength = strlen($name);
        if($namePrefixLen == $nameLength){
            $uri = $this->targetNs;
            $qName = $uri.':'.$name;
        }elseif($namePrefixLen < $nameLength){
            $namePrefix = substr($name,0,$namePrefixLen);
            $uri = $this->ns[$namePrefix]->uri;
            $qName = $uri.':'.substr($name, -($nameLength -
$namePrefixLen - 1));
        }else{
            $uri = '';
            $qName = $name;
        }
        return $qName;
    }
```

```
    public function getVariableValue($var)
    {
        $qVar = $this->getPrefix($var);
        foreach($this->variables as $variable){
            if($variable->name == $qVar) {
                return $variable->value;
            }
        }
    }

    function calcVarValue($var)
    {
        $partLenght = count($var);
        echo "<br/> no of parts " . $partLenght . "<br/>";
        $varPartIndex = 0;
        foreach($var->varParts as $varPart){
            $value[$varPartIndex] = $varPart->value;
            $varPartIndex++;
        }
        if ($partLenght == 1) {
            return $value[0];
        } else {
            return serialize($value);
        }
    }

}

?>
```