ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Προηγμένες τεχνικές ενορχήστρωσης Υπηρεσιών Ιστού σε σενάρια BPEL

Ευάγγελος Καρελιώτης
it 21115

Αθήνα, 2018

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Προηγμένες τεχνικές ενορχήστρωσης Υπηρεσιών Ιστού σε σενάρια BPEL

Τριμελής Εξεταστική Επιτροπή

Τσερπές Κωνσταντίνος (Επιβλέπων)

Νικολαίδη Μάρα

Μιχαήλ Δημήτρης

Ο Ευάγγελος Καρελιώτης

δηλώνω υπεύθυνα ότι:

- Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλει τα πνευματικά δικαιώματα τρίτων.

- Αποδέχομαι ότι η ΒΚΠ μπορεί, χωρίς να αλλάξει το περιεχόμενο της εργασίας μου, να τη διαθέσει σε ηλεκτρονική μορφή μέσα από τη ψηφιακή Βιβλιοθήκη της, να την αντιγράψει σε οποιοδήποτε μέσο ή/και σε οποιοδήποτε μορφότυπο καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.

ΠΕΡΙΛΗΨΗ

Η προσανατολισμένη στις υπηρεσίες αρχιτεκτονική (SOA) είναι ένα σύνολο αρχών και μεθοδολογιών για το σχεδιασμό και την ανάπτυξη υπηρεσιών που μπορούν να επικοινωνούν ανεξαρτήτως περιβάλλοντος εκτέλεσης και γλώσσας προγραμματισμού, παρέχοντας κατ' αυτό τον τρόπο διαλειτουργικότητα (interoperability). Οι υπηρεσίες έχουν επιπλέον το σημαντικό πλεονέκτημα ότι μπορούν να επαναχρησιμοποιηθούν καθώς δεν περιορίζονται από υποκείμενες τεχνολογίες υλοποίησης. Η SOA με Υπηρεσίες Ιστού (Web Services) συνδυάζει βέλτιστες πρακτικές των παλαιότερων αρχιτεκτονικών λογισμικού, ευθυγραμμίζει καλύτερα την επιχειρησιακή λογική με την υπάρχουσα τεχνολογία και επιλύει σύγχρονα επιχειρησιακά προβλήματα όπως την ολοκλήρωση (integration) και την επαναχρησιμοποίηση (reusability) των υπαρχόντων συστημάτων αλλά και την αυτοματοποίηση των επιχειρησιακών διαδικασιών. Σκοπός της παρούσας εργασίας είναι να παρουσιάσει πως η υιοθέτηση της SOA με υπηρεσίες Ιστού μπορεί να είναι μια αποδοτική λύση σε περιπτώσεις εκτός του στενού πλαισίου της επιχείρησης, όπου είναι επιθυμητή η αυτοματοποίηση διαδικασιών, η αλληλεπίδραση και η ολοκλήρωση ετερογενών συστημάτων. Για να επιτευχθεί η σύνθεση των υπηρεσιών και ο συντονισμός τους σε μια αυτοματοποιημένη επιχειρησιακή διαδικασία, δηλαδή η ενορχήστρωση των Υπηρεσιών Ιστού, χρησιμοποιήθηκε η γλώσσα BPEL. Για την αποθήκευση του συνόλου των δεδομένων χρησιμοποιήθηκε μια σχεσιακή βάση δεδομένων. Στα πλαίσια της εργασίας μελετήθηκαν οι προδιαγραφές, η τεχνολογία της SOA και τα θέματα διαχείρισης δεδομένων, αξιοπιστίας, ανταλλαγής μηνυμάτων, οι συναλλαγές καθώς και η ανάγκη ενορχήστρωσης και διαχείρισης επιχειρησιακών διεργασιών που προκύπτουν με την χρησιμοποίηση της SOA.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Κατανεμημένα Συστήματα

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Υπηρεσιοστρεφής αρχιτεκτονική, Υπηρεσίες Ιστού, ενορχήστρωση επιχειρησιακών διεργασιών, BPEL

ABSTRACT

Service-oriented architecture (SOA) is a set of principles and methodologies concerning design and development of services that can communicate independently of their execution environment and implementation's programming language, thus offering interoperability. Services have also the important advantage of being able to be reused because they are not limited by underlying implementation technologies. SOA with Web Services combines best practices of earlier software architectures and aligns more efficiently business logic with existing technology. Therefore, it solves modern business problems such as integration and reusability of both existing and legacy systems and automation of business processes. The purpose of this thesis is to show how SOA with Web Services adoption can be an effective solution in study cases outside the narrow business framework, where processes' automation is desired along with interaction and integration of heterogeneous systems. To achieve services' composition and coordination in an automated business process, in other words Web Services' orchestration, the BPEL language was used. Additionally, a relational database was used to store the complete set of data needed. This thesis studies SOA's specifications and technologies and issues of data management, reliability, messaging, transactions, orchestration and Business Process Management (BPM) arising from the use of SOA. Furthermore, we examine protocols and technologies used in Web Services to emphasize how they facilitate reuse and loose coupling of services. Last but not least, we display experimental measurements of execution time and throughput and also fault handling techniques for system exceptions as well as business level exceptions that can be thrown during BPEL scenarios' execution.

SUBJECT AREA: Distributed Systems

KEYWORDS: SOA, Web Services, business process orchestration, BPEL

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ 8

1. ΕΙΣΑΓΩΓΗ 9

1.1 Οι υπηρεσίες και η σημασία τους στη SOA 9

1.2 Σύνθεση υπηρεσιών σε επιχειρησιακές διεργασίες και ενορχήστρωση 10

1.3 Αντικείμενο της πτυχιακής εργασίας 12
1.4 Layers 12

2. ΤΕΧΝΟΛΟΓΙΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΠΡΟΣΑΝΑΤΟΛΙΣΜΕΝΗΣ ΣΤΙΣ ΥΠΗΡΕΣΙΕΣ 15

2.1 Εισαγωγή και ορισμοί .....
2.1.1 Ορισμός της έννοιας της υπηρεσίας 15
2.1.2 Ορισμός της προσανατολισμένης στις υπηρεσίες αρχιτεκτονικής (Service Oriented Architecture - SOA)16

2.2 Η προσανατολισμένη στις υπηρεσίες αρχιτεκτονική διευκολύνει την επαναχρησιμοποίηση μέσω της χαλαρής σύνδεσης (loose coupling) 16

2.3 Προδιαγραφές (Specifications) 17

2.4 Επισκόπηση επιλεγμένων τεχνολογιών που έχουν χρησιμοποιηθεί για την υλοποίηση SOA 18
2.4.1 IBM WebSphere MQ 18
2.4.2 CORBA 19
2.4.3 Python 19
2.4.4 Ruby 20
2.4.5 Apache Groovy 21
2.4.6 Java και J2EE 21
2.4.7 Πλατφόρμες υπηρεσίας προς υπηρεσία (B2B platforms) 22

2.5 Η προσανατολισμένη στις υπηρεσίες δραστηριότητα (Service Oriented Enterprise) 22

2.6 Προσανατολισμένη στις υπηρεσίες ανάπτυξη (Service Oriented Development) 22

2.7 Η σημασία των υπηρεσιών για τις επιχειρήσεις 23

2.8 Αν η προσανατολισμένη στις υπηρεσίες αρχιτεκτονική (SOA) δεν είναι νέα έννοια, τότε τι έχει αλλάξει;. 24

2.9 Διαχείριση μεταδεδομένων (metadata management) 24

2.10 Αξιοπιστία και ανταλλαγή μηνυμάτων 25

2.11 Συναλλαγές (Transactions) 25

2.12 Διακυβέρνηση υπηρεσιών, διαδικασίες, οδηγίες, αρχές και μέθοδοι 26
2.12.1 Πολιτικές διακυβέρνησης και διαδικασίες της προσανατολισμένης σε υπηρεσίες αρχιτεκτονικής .. 26

2.13 Αρχές και κατευθυντήριες γραμμές για τη SOA αρχιτεκτονική 27
2.13.1 Βασικά χαρακτηριστικά υπηρεσίας 27

2.13.2 Χαρακτηριστικά που πρέπει να πληρούν οι YI 28

3. ΤΕΧΝΟΛΟΓΙΑ XML 30

3.1 Εισαγωγή 30

3.2 Δημιουργία ενός Κανονικού Μοντέλου Δεδομένων (Canonical Data Model) 30
3.2.1 Ορισμός του κανονικού μοντέλου δεδομένων 31
3.2.2 Ενημέρωση του κανονικού μοντέλου 32

3.3 Πως η XML βοηθάει να απλοποιηθεί η ανάπτυξη και ολοκλήρωση συστημάτων 32

4. ΣΤΟΙΒΑ ΠΡΩΤΟΚΟΛΛΩΝ ΥΠΗΡΕΣΙΩΝ ΙΣΤΟΥ 33

4.1 Η κλασσική στοίβα πρωτοκόλλων υπηρεσιών ιστού 33

4.2 Μοντελοποίηση της στοίβας πρωτοκόλλων Υπηρεσιών Ιστού σύμφωνα με το μοντέλο αναφοράς OSI 35

4.3 Απλό πρωτόκολλο πρόσβασης αντικειμένου (Simple Object Access Protocol –SOAP) 38
4.3.1 Μορφή μηνύματος SOAP 39
4.3.2 SOAP Πελάτες και Εξυπηρετητές 39
4.3.3 SOAP ρόλοι και SOAP κόμβοι 40
4.3.4 SOAP-RPC 40
4.3.5 Μια περίπτωση χρήσης του Απλού Πρωτόκολλου Πρόσβασης Αντικειμένου (SOAP) 45

4.4 Γλώσσα Περιγραφής Υπηρεσιών Ιστού (Web Services Description Language – WSDL) 46
4.4.1 WSDL Έγγραφα (Documents) 46
4.4.2 WSDL θύρες (ports) 48
4.4.3 WSDL δέσμευση (binding) 48

4.5 Παγκόσμια περιγραφή, ανακάλυψη και ενσωμάτωση (Universal Description, Discovery and Integration) 48
4.5.1 Εισαγωγή 49
4.5.2 Τα οφέλη του UDDI και η σταδιακή εγκατάλειψή του 49

5. ΤΕΧΝΟΛΟΓΙΑ ΥΠΗΡΕΣΙΩΝ ΙΣΤΟΥ 50

5.1 Ατομικές και σύνθετες υπηρεσίες 50

5.2 Σύγκριση CORBA με τις Υπηρεσίες Ιστού για SOA 50

5.3 Δημιουργία Υπηρεσιών Ιστού με JAX-WS 51
5.3.1 Εισαγωγή 51
5.3.2 Χρήση των ιδιοτήτων ονομάτων της σήμανσης JAX-WS (JAX-WS annotation name properties) 52
5.3.3 Δημιουργία ενός πληρεξούσιου πελάτη (client proxy) 52
5.3.4 Χρήση υπηρεσίας Ιστού από κάποιο Servlet ή EJB 53

5.4 Παροχή υπηρεσιών Ιστού βασισμένων σε SOAP 53
5.4.1 Δημιουργία μιας βασικής Υπηρεσίας Ιστού 53
5.4.2 Καθορισμός χώρου ονομάτων (namespace) 53
5.4.3 Δημιουργία μιας λειτουργίας υπηρεσίας Ιστού 53
5.4.4 Προσδιορισμός του μέρους μηνύματος της υπηρεσίας Ιστού (Web Service message part) και καθορισμός επιστρεφόμενης τιμής λειτουργίας 54

5.5 RESTful Υπηρεσίες Ιστού 54
5.5.1 Οι αρχές του REST 54
5.5.2 Πλεονεκτήματα του REST 55

6. ΤΕΧΝΟΛΟΓΙΑ ΕΠΙΧΕΙΡΗΣΙΑΚΩΝ ΔΙΑΔΙΚΑΣΙΩΝ / BPEL ΚΑΙ SOA 56

6.1 Διαχείριση Επιχειρησιακών Διαδικασιών (Business Process Management – BPM) 56

6.2 Συστήματα διαχείρισης επιχειρησιακών διαδικασιών (Business Process Management Systems – BPMS) . 57
6.2.1 Μοντελοποίηση διαδικασίας 58
6.2.2 Εκτέλεση Διαδικασίας 58
6.2.3 Παρακολούθηση διαδικασίας και επιχειρησιακής δραστηριότητας (Process Monitoring and Business Activity monitoring) 58
6.2.4 Υποδομή 58

6.3 Γλώσσα εκτέλεσης επιχειρησιακών διεργασιών (Business Process Execution Language – BPEL) 60
6.3.1 BPEL για αυτοματοποίηση της διαδικασίας 60
6.3.2 Ενορχήστρωση και Χορογραφία (Orchestration and Choreography) 60
6.3.3 Βασικές έννοιες 63
6.3.4 Κλήση υπηρεσιών 63
6.3.5 Σύνδεσμοι συνεργατών (partner links) 64
6.3.6 Μεταβλητές 65
6.3.7 Οι δραστηριότητες `<invoke>`, `<receive>` και `<reply>` 66
6.4 Οφέλη BPEL 66

7. ΜΕΘΟΔΟΛΟΓΙΑ – ΥΛΟΠΟΙΗΣΗ 67

7.1 Στάδια ανάπτυξης εφαρμογής 67
7.1.1 Ανάδειξη Προβλήματος (Problem identification) 67

7.2 Τεχνολογία και ανάπτυξη πλαισίου εργασίας 68

7.3 Τεχνική Ανάλυση Τεχνολογίας που χρησιμοποιήθηκε 69
7.3.1 Ανάλυση του κώδικα 69
7.3.2 Ανάλυση BPEL PROCESS 74
7.3.3 Ανάλυση της βάσης δεδομένων 93

8. ΣΥΜΠΕΡΑΣΜΑΤΑ 98

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ 99

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ 100

ΑΝΑΦΟΡΕΣ 102

ΠΡΟΛΟΓΟΣ

Η παρούσα πτυχιακή εργασία εκπονήθηκε στα πλαίσια του Προπτυχιακού Προγράμματος Σπουδών του Τμήματος Πληροφορικής και Τηλεματικής του Χαροκόπειου Πανεπιστημίου Αθηνών. Στόχος της εργασίας είναι να δείξουμε πως μπορούμε να διαχειριστούμε πολλά Web services μαζί, με τη βοήθεια της γλώσσας BPEL που λειτουργεί ως ενορχηστρωτής και συντονίζει όλο το project, το οποίο αναπτύχθηκε με χρήση τεχνικών ενορχήστρωσης Υπηρεσιών Ιστού στο πλαίσιο της προσανατολισμένης σε υπηρεσίες αρχιτεκτονικής (SOA). Στην εργασία μελετώνται οι τεχνολογίες της SOA, οι τεχνολογίες XML και τα πρωτόκολλα που χρησιμοποιούν οι Υπηρεσίες Ιστού ως τεχνική ενορχήστρωσης Υπηρεσιών Ιστού.

1. ΕΙΣΑΓΩΓΗ

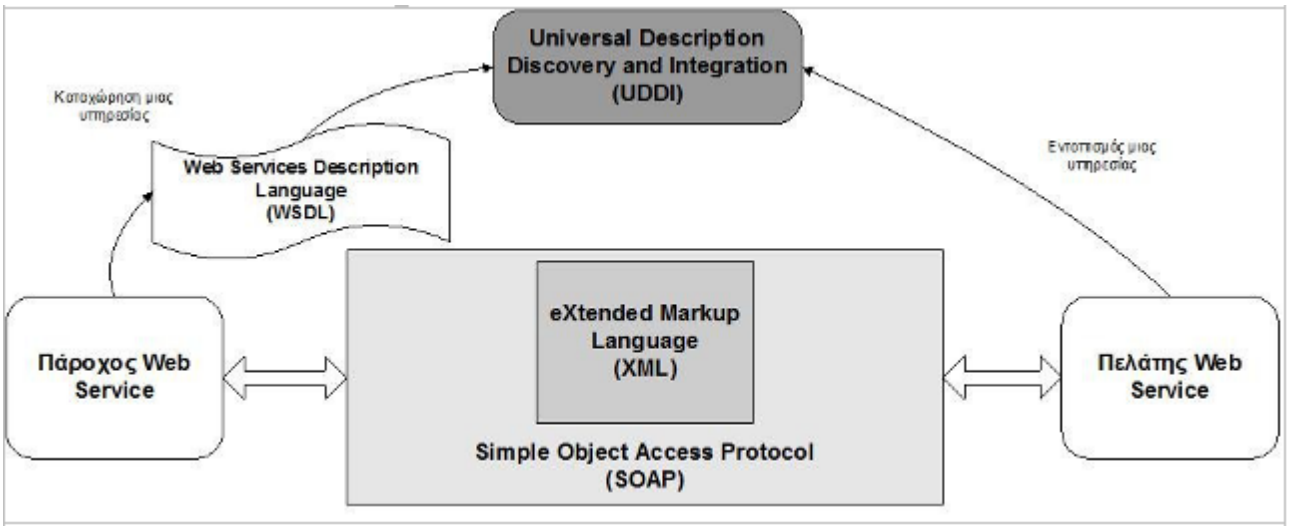
Κατά το παρελθόν, όταν η χρήση υπολογιστικών συστημάτων εντάχθηκε στις επιχειρήσεις, η κυρίαρχη πρόκληση ήταν η ανάπτυξη συστημάτων που θα αυτοματοποιούσαν τις λειτουργίες που μέχρι πρότινος διεκπεραίωνε το ανθρώπινο δυναμικό, όπως για παράδειγμα η τιμολόγηση, η λογιστική ή η διαχείριση παραγγελιών. Σε αυτό το πλαίσιο, η ανταλλαγή λογικής της εφαρμογής (application logic) και δεδομένων μεταξύ πολλαπλών μηχανημάτων δε θεωρούνταν ιδιαίτερα σημαντική. Επιπροσθέτως, η ιδέα μιας κοινής, επαναχρησιμοποιήσιμης αρχιτεκτονικής ήταν κάτι που λίγες μόνο επιχειρήσεις θα μπορούσαν να υλοποιήσουν.
Στη σύγχρονη εποχή, οι πιο πολλές επιχειρησιακές λειτουργίες έχουν αυτοματοποιηθεί και το ερώτημα που τίθεται είναι πως θα βελτιωθεί η ικανότητα των συστημάτων να ανταποκρίνονται στις σύγχρονες απαιτήσεις που επιβάλλουν επαναχρησιμοποίηση (reusability), ευελιξία (flexibility) και ολοκλήρωση (integration) συστημάτων Τεχνολογίας Πληροφοριών (ΤΠ). Στη λύση του προβλήματος αυτού αναγνωρίζεται η καθοριστική συμβολή της προσανατολισμένης σε υπηρεσίες αρχιτεκτονικής (SOA). Η SOA είναι ένα σχεδιαστικό στυλ που καθοδηγεί τον οργανισμό κατά τη διάρκεια όλων των σταδίων δημιουργίας και χρήσης επιχειρησιακών υπηρεσιών, από τη σύλληψη της ιδέας για κάποια υπηρεσία μέχρι την κατάργηση της υπηρεσίας.
Η χρήση της SOA σε επιχειρήσεις και οργανισμούς αυξάνει την επαναχρησιμοποίηση, μειώνει το συνολικό κόστος και βελτιώνει την ικανότητα εξέλιξης και προσαρμογής στις αλλαγές των συστημάτων ΤΠ.

1.1 Οι υπηρεσίες και η σημασία τους στη SOA

Μια υπηρεσία διαφέρει από ένα αντικείμενο ή μια διαδικασία επειδή καθορίζεται από τα μηνύματα που ανταλλάσσει με άλλες υπηρεσίες. Η χαλαρή συνδεσιμότητα (loose coupling) της υπηρεσίας με τις εφαρμογές που τη φιλοξενούν επιτρέπει την ευκολότερη ανταλλαγή δεδομένων μεταξύ των τμημάτων της επιχείρησης μέσω τοπικών δικτύων ή μέσω του Διαδικτύου. Η αρχιτεκτονική SOA καθορίζει τον τρόπο με τον οποίο θα εγκατασταθούν οι υπηρεσίες και πως θα τις διαχειριστούν οι επιχειρήσεις.

Όταν μια υπηρεσία είναι διαθέσιμη μέσω του Διαδικτύου ή ιδιωτικού τοπικού δικτύου (intranet), χρησιμοποιεί τυποποιημένο σύστημα ανταλλαγής μηνυμάτων που βασίζεται σε XML (π.χ. SOAP[1]) και δεν σχετίζεται ούτε δεσμεύεται από συγκεκριμένο λειτουργικό σύστημα ή γλώσσα προγραμματισμού, αποτελεί μια Υπηρεσία Ιστού (ΥΙ). Οι ΥΙ είναι αυτό-περιγραφόμενες με την έννοια ότι πρέπει να δημοσιοποιούν τη διεπαφή τους στην οποία περιγράφονται οι λειτουργίες που παρέχει η υπηρεσία καθώς και το πώς θα κληθούν αυτές και ποιες είναι οι επιστρεφόμενες τιμές τους.

[1] http://www.w3.org/TR/2007/REC-soap12-part0-20070427/



Τα ευρέως υιοθετημένα και υλοποιημένα βασικά πρότυπα Υπηρεσιών Ιστού έχουν επιτύχει πρωτοφανή διαλειτουργικότητα μεταξύ πολύ διαφορετικών συστημάτων λογισμικού. Ως επακόλουθο, έχουν προταθεί νέα πρότυπα ΥΙ που επεκτείνουν τη χρήση των ΥΙ σε μεγάλο εύρος νέων εφαρμογών. Τα πρότυπα αυτά καλύπτουν θέματα ασφάλειας, αξιοπιστίας, συναλλαγών, διαχείρισης μεταδεδομένων και ενορχήστρωσης στα οποία γίνεται αναφορά στην παρούσα εργασία.
Η SOA με υπηρεσίες Ιστού είναι μια καλή επιλογή για την επίλυση των προβλημάτων ολοκλήρωσης των συστημάτων ΤΠ και αυτοματοποίησης των επιχειρησιακών διεργασιών στις επιχειρήσεις και οργανισμούς επειδή ευθυγραμμίζει την ΤΠ με τις επιχειρησιακές απαιτήσεις. Οι υπηρεσίες αναπτύσσονται ώστε να αποκρύπτουν τις λεπτομέρειες του περιβάλλοντος εκτέλεσής τους. Συγκεκριμένα οι Υπηρεσίες Ιστού έχουν τη δυνατότητα να αποστέλλουν και να χρησιμοποιούν δεδομένα με XML αναπαράσταση, ανεξαρτήτως της πλατφόρμας ανάπτυξης, του ενδιάμεσου λογισμικού, του τύπου του υλικού ή του λειτουργικού συστήματος.

1.2 Σύνθεση υπηρεσιών σε επιχειρησιακές διεργασίες και ενορχήστρωση

Το σύνολο των σχετιζόμενων, δομημένων δραστηριοτήτων που συνθέτουν κάποια υπηρεσία καλείται επιχειρησιακή διεργασία (business process). Ένα από τα βασικά πρότυπα που επιταχύνουν την υιοθέτηση της SOA από τις επιχειρήσεις είναι η γλώσσα εκτέλεσης επιχειρησιακών διεργασιών (WS-BPEL[2]) η οποία καθορίζει τις ενέργειες που πραγματοποιούνται μέσα στις επιχειρησιακές διεργασίες με τις Υπηρεσίες Ιστού. Η BPEL επιτρέπει την αυτοματοποίηση της διεργασίας με την ενορχήστρωση των υπηρεσιών που συμμετέχουν σε αυτές. Η ενορχήστρωση είναι σύνθεση των υπηρεσιών σε μια λογική ροή, καθορίζοντας μια εκτελέσιμη διεργασία που περιλαμβάνει ανταλλαγές μηνυμάτων με άλλα συστήματα. Οι ακολουθίες ανταλλαγής μηνυμάτων που προκύπτουν κατά την εκτέλεση ελέγχονται κεντρικά από το σχεδιαστή της ενορχήστρωσης.

[2] http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html

1.3 Αντικείμενο της πτυχιακής εργασίας

Η ανάλυση των υπηρεσιών που προσφέρει μια εταιρεία (ταξιδιωτικός πράκτορας) σε συνδυασμό με τις υπηρεσίες που χρειάζονται από αυτήν οι χρήστες, οδήγησαν στη διαπίστωση για δημιουργία αυτοματοποιημένων διαδικασιών. Οι ανάγκες αυτές αφορούν ένα ευρύ φάσμα λειτουργιών που περιλαμβάνει λειτουργίες, όπως είναι η εύρεση μιας ξενοδοχειακής μονάδας, με αναζήτηση κόστους και ημερομηνιών, η εύρεση κάποιου οχήματος για την διευκόλυνση του πελάτη και τέλος η εύρεση ακτοπλοικών εισητηρίων αναλογα με τον προορισμό, την διαθεσιμότητα και την πληρότητα. Ο σκοπός της παρούσας εργασίας είναι να παρουσιάσει πώς ένα σύνολο τέτοιων διαδικασιών, μπορεί να μοντελοποιηθεί και να ελεγχθεί σε ένα ολοκληρωμένο πλαίσιο.

Η επιχειρησιακή διεργασία, η οποία αναπτύχθηκε στο πλαίσιο της εργασίας, παρουσιάζει ένα σενάριο παροχής εξατομικευμένων υπηρεσιών στους χρήστες για πακέτα διακοπών. Η χρήση της BPEL στην παρούσα εργασία αποσκοπεί στην επιτυχημένη διαχείριση των συντονισμένων δραστηριοτήτων με την κατάλληλη σειρά προκειμένου να επιτευχθεί ο στόχος της διάθεσης των εξατομικευμένων υπηρεσιών στο χρήστη με αυτοματοποιημένο τρόπο.

Οι υπηρεσίες Ιστού δεν ενσωματώνουν κλήσεις η μία στην άλλη μέσα στον πηγαίο κώδικα τους αλλά χρησιμοποιούν, όπως αναφέρθηκε, πρωτόκολλα για να περιγράψουν το πώς θα γίνει η ανταλλαγή μηνυμάτων μεταξύ τους και επεξεργάζονται τα μηνύματα με χρήση μεταδεδομένων (δηλαδή με πληροφορίες που αφορούν τα δεδομένα). Η SOA κάνει εκτεταμένη χρήση της XML για την δόμηση των δεδομένων. Η επικοινωνία μεταξύ διαφορετικών μορφών δεδομένων (data formats) επιτυγχάνεται με ένα πρότυπο σχεδίασης γνωστό ως κανονικό μοντέλο δεδομένων (canonical data model). Τόσο τα οφέλη και οι αρχές της SOA όσο και οι τεχνολογίες XML και οι προδιαγραφές που χρησιμοποιεί εξετάζονται στο δεύτερο και τρίτο κεφάλαιο της παρούσας πτυχιακής εργασίας.

Η SOA χρησιμοποιεί την WSDL για την περιγραφή των ίδιων των υπηρεσιών και το πρωτόκολλο SOAP για να περιγράψει τα πρωτόκολλα επικοινωνίας. Η σημασία της XML για τις YI τονίζεται από το γεγονός ότι αποτελεί τη βάση του πρωτοκόλλου SOAP, της γλώσσας WSDL[3] και του μητρώου UDDI[4] που συνθέτουν τη στοίβα πρωτοκόλλων Υπηρεσιών Ιστού. Η στοίβα πρωτοκόλλων και τα μέλη της αναλύονται στο τέταρτο κεφάλαιο της εργασίας.

Στο πέμπτο κεφάλαιο εξετάζεται η δημιουργία Υπηρεσιών Ιστού με το Java API for XML Web Services (JAX-WS[5]) που χρησιμοποιήθηκε στην υλοποίηση των υπηρεσιών Ιστού που αναπτύχθηκαν και θέματα που αφορούν τις βασισμένες σε SOAP Υπηρεσίες Ιστού. Επίσης, γίνεται αναφορά στις εναλλακτικές των SOAP υπηρεσιών, τις RESTful[6] υπηρεσίες.
Η αυτοματοποίηση της επιχειρησιακής διεργασίας θέτει το θέμα της διαχείρισης επιχειρησιακών διεργασιών (Business Process Management – BPM) και των συστημάτων BPM για τη διασφάλιση επίτευξης των στόχων προσφέροντας λειτουργίες που αφορούν τη

---

3 http://www.w3.org/TR/wsdl20/
4 http://www.uddi.org/pubs/uddi_v3.htm
5 http://jax-ws.java.net/
6 http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

μοντελοποίηση, την εκτέλεση αλλά και την παρακολούθηση της διεργασίας. Το πρώτο μισό του έκτου κεφαλαίου αφιερώνεται στα θέματα αυτά, ενώ στη συνέχεια εξετάζεται ο ρόλος της γλώσσας BPEL στην αυτοματοποίηση των επιχειρησιακών διεργασιών και βασικές έννοιες και δραστηριότητες της BPEL που χρησιμοποιήθηκαν στην πράξη κατά την ανάπτυξη του ολοκληρωμένου πλαισίου εργασίας.

Το έβδομο κεφάλαιο αφορά την μεθοδολογία που ακολουθήθηκε για την ανάπτυξη της εφαρμογής, περιγράφεται λεπτομερώς τι υλοποιήθηκε και πώς (χρησιμοποιούμενες τεχνολογίες, περιβάλλον και γλώσσα υλοποίησης) καθώς και η επιχειρησιακή λογική της διεργασίας. Παρατίθεται ο σχετικός κώδικας και σχήματα από το IDE.

1.4 Layers

Λογικές ομαδοποιήσεις των στοιχείων λογισμικού που απαρτίζουν την εφαρμογή ή την υπηρεσία. Βοηθούν να γίνει η διάκριση μεταξύ των διαφόρων ειδών των εργασιών που εκτελούνται από τα συστατικά.

- Presentation services : Υπηρεσίες προσανατολισμένες στο χρήστη υπεύθυνος για τη διαχείριση της αλληλεπίδρασης χρήστη με το σύστημα και γενικά αποτελούνται από συστατικά που βρίσκονται μέσα στο στρώμα παρουσίασης.

- Business services : Εφαρμόζουν τη βασική λειτουργικότητα του συστήματος και ενσωματώνουν τη σχετική επιχειρηματική λογική. Αυτοί γενικά αποτελούνται από συστατικά που βρίσκονται μέσα στο στρώμα των επιχειρήσεων οι οποίες ενδέχεται να εκθέσουν διεπαφές υπηρεσιών που άλλοι καλούντες μπορούν να χρησιμοποιήσουν.

- Data services : Παρέχουν πρόσβαση σε δεδομένα που φιλοξενείται εντός των ορίων του συστήματος και τα δεδομένα που εκτίθενται από άλλα back-end συστήματα. Το στρώμα δεδομένων εκθέτει τα δεδομένα με το στρώμα των επιχειρήσεων μέσω της γενικής διασύνδεσης και έχει σχεδιαστεί ώστε να είναι βολικό για κάθε χρήστη από επιχειρηματικές υπηρεσίες.

Presentation Layer Components

- Εφαρμόζει την λειτουργικότητα που απαιτείται για να επιτρέπει στους χρήστες να αλληλεπιδρούν με την αίτηση.

- User interface (UI) components : Αυτά τα συστατικά παρέχουν το μηχανισμό για τους χρήστες να αλληλεπιδρούν με την αίτηση.

- UI process components : Συγχρονίζουν και ενορχηστρώνουν τις αλληλεπιδράσεις του χρήστη. Μπορεί να είναι χρήσιμη για να οδηγεί στη διαδικασία χρησιμοποιώντας ξεχωριστά συστατικά διαδικασίας UI. Αυτό εμποδίζει τη ροή διαδικασίας και τη λογική διαχείρισης κατάστασης, από το να είναι κωδικοποιημένες στα στοιχεία UI οι ίδιοι. Έτσι επιτρέπεται να χρησιμοποιούνται ξανά τα ίδια βασικά πρότυπα αλληλεπίδρασης χρήστη σε άλλες διεπαφές χρήστη.

Business Layer Components

- Εφαρμόζουν τη βασική λειτουργικότητα του συστήματος και ενσωματώνουν τη σχετική επιχειρηματική λογική.

- Application Facade : Προαιρετική δυνατότητα που μπορείτε να χρησιμοποιήσετε για το συνδυασμό πολλαπλών επιχειρηματικών δραστηριοτήτων σε μειωμένες εργασίες (single message-based). Αυτή η λειτουργία είναι χρήσιμη όταν εντοπίζεται το στρώμα παρουσίασης σε μία φυσική ξεχωριστή βαθμίδα που μας επιτρέπει να βελτιώσουμε τη χρήση της μεθόδου επικοινωνίας που τους συνδέει.

- Business components : Εφαρμόζουν την επιχειρηματική λογική της εφαρμογής. Ανεξάρτητα από το αν μία επιχειρηματική διαδικασία αποτελείται από μόνο ένα βήμα η μία ενορχηστρωμένη ροή εργασίας, η αίτηση σας είναι πιθανό να απαιτεί τα στοιχεία που εφαρμόζουν τους κανόνες των επιχειρήσεων και να εκτελέσει τα καθήκοντά των επιχειρήσεων.

- Business workflows : Μετά τα στοιχεία UI components των αιτούμενων στοιχείων από το χρήστη, η εφαρμογή μπορεί να χρησιμοποιήσει αυτά τα δεδομένα για την εκτέλεση μιας επιχειρηματικής διαδικασίας η οποία περιλαμβάνει πολλά βήματα. Αυτά τα βήματα πρέπει να εκτελούνται με σωστή σειρά και να μπορούν να αλληλεπιδρούν μέσω ενός ενορχηστρωτή. Τα Business workflows components καθορίζουν και συντονίζουν τις long-Running και multi-step επιχειρηματικές διαδικασίες και μπορεί να εφαρμοστεί χρησιμοποιώντας τα εργαλεία διαχείρισης επιχειρηματικών διαδικασιών.

- Business entity components : Επιχειρηματικές οντότητες που χρησιμοποιούνται για να περάσει τα δεδομένα μεταξύ συσκευών. Τα δεδομένα αντιπροσωπεύουν επιχειρηματικές οντότητες του πραγματικού κόσμου όπως τα προϊόντα ή οι παραγγελίες. Οι επιχειρηματικές οντότητες που η εφαρμογή χρησιμοποιεί εσωτερικά είναι συνήθως δομές δεδομένων, αλλά μπορεί επίσης να υλοποιηθεί με τη χρήση custom object-oriented κατηγοριών.

Data Layer Components

➔ Παρέχουν πρόσβαση σε δεδομένα που φιλοξενούνται εντός των ορίων του συστήματος και τα δεδομένα που εκτείθενταιαπό άλλα back-end συστήματα.

- Data access components : Αυτά τα abstract components είναι η λογική που

απαιτείται γαι την πρόσβαση των υποκείμενων που αποθηκεύει δεδομένα. Κάτι τέτοιο κάνει την εφαρμογή πιο εύκολη να ρυθμίζεται και να διατηρείται.

- Data helper and utility components : μειώνει την πολυπλοκότητα των συστατικών πρόσβασης σε δεδομένα και συγκεντρώνει την λογική, η οποία απλοποιεί την συντήρηση. Επίσης, τα συστατικά αυτού μπορούν συχνά να επαναχρησιμοποιηθούν σε άλλες εφαρμογές.
- Service agents : Όταν ένα στοιχείο των επιχειρήσεων πρέπει να χρησιμοποιεί την λειτουργικότητα που παρέχεται από μια ξεχωριστή υπηρεσία, μπορεί να χρειαστεί να εφαρμόσει κώδικα για την διαχείρηση της επικοινωνίας με την συγκεκριμένη υπηρεσία.

Cross – Cutting components

- Components for implementing security : Εκτελούν πιστοποίηση, εξουσιοδότηση και την επικύρωση.
- Components for implementing operational management tasks : Περιλαμβάνουν συστατικά που εφαρμόζουν πολιτικές με εξαίρεση το χειρισμό, την υλοτομία, μετρητές επιδόσεων, την διαμόρφωση και την ανίχνευση.
- Components for implementing communication : Περιλαμβάνουν συστατικά που επικοινωνούν με άλλες υπηρεσίες και εφαρμογές.

Service Layer Components

- Παρέχονται σε άλλους πελάτες και εφαρμογές με ένα τρόπο για να αποκτήσουν πρόσβαση σε επιχειρηματική λογική της εφαρμογής και να κάνουν χρήση της λειτουργικότητας της εφαρμογής με το πέρασμα μηνυμάτων.

- Service interfaces : Υπηρεσίες εκθέτουν μία διεπαφή στην οποία αποστέλλονται όλα τα εισερχόμενα μηνύματα. Ορισμός των μηνυμάτων συνιστούν σύμβαση.

- Message Types : Κατά την ανταλλαγή δεδομένων μέσω ενός service layer οι δομές δεδομένων είναι τυλιγμένες από message structure που υποστηρίζονται από διαφορετικούς τύπους ενεργειών.

Tiers

- Βαθμίδες που αντιπροσωπεύουν το φυσικό διαχωρισμό των λειτουργιών παρουσίασης, των επιχειρήσεων, τις υπηρεσίες και όλα τα στοιχεία του σχεδιασμού μας, σε άλλους υπολογιστές και συστήματα.

- Two – tier **:** Αντιπροσωπεύει μία βασική δομή με δύο κύριες συνιστώσες ενός πελάτη και ενός εξυπηρετητή. Σε αυτό το σενάριο ο πελάτης και ο διακομιστής μπορεί να υπάρχουν στο ίδιο μηχάνημα ή μπορεί να βρίσκονται σε διαφορετικά

μηχανήματα. Αυτή η βαθμίδα περιέχει presentation layer logic και κάθε απαιτούμενη business layer logic. Η εφαρμογή web επικοινωνεί με ξεχωριστό μηχάνημα που φιλοξενεί τη βαθμίδα βάσης δεδομένων η οποία περιέχει τα data layer logic.

- Three – tier **:** Ο πελάτης αλληλεπιδρά με το λογισμικό εφαρμογής σε ξεχωριστό server και ο διακομιστής-εφαρμογή με τη βάση δεδομένων που βρίσκεται επίσης σε ξεχωριστό διακομιστή.

- N – tier **:** Σε αυτό το σενάριο ο διακομιστής web server είναι φυσικά διαχωρισμένα από το application server που αποτελεί την επιχειρηματική λογική. Αυτό συμβαίνει για λόγους ασφαλείας, όπου ο web server έχει αναπτυχθεί σε περιμετρικό δίκτυο και ο application server βρίσκεται σε διαφορετικό υποδίκτυο μέσω ενός τείχους προστασίας. Είναι επίσης κοινό να εφαρμόσει ένα firewall μεταξύ client και web-tier.

2. ΤΕΧΝΟΛΟΓΙΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΠΡΟΣΑΝΑΤΟΛΙΣΜΕΝΗΣ ΣΤΙΣ ΥΠΗΡΕΣΙΕΣ

2.1 Εισαγωγή και ορισμοί

2.1.1 Ορισμός της έννοιας της υπηρεσίας

Μια αρχική δυσκολία στην προσέγγιση SOA είναι ότι δεν είναι πάντοτε εφικτό να βρεθεί ένας κοινά αποδεκτός ορισμός για το θεμελιώδες δομικό στοιχείο της, **την υπηρεσία**. Πολλοί από τους ορισμούς της υπηρεσίας αναφέρονται σε πολύ αφηρημένες έννοιες, όπως «προσαρμοστικότητα» και «ευελιξία», που είναι πάρα πολύ ασαφείς για να είναι χρήσιμες για τον προγραμματιστή.
Μια υπηρεσία μπορεί να ορίζεται βάσει των σχημάτων ανταλλαγής μηνυμάτων που υποστηρίζει. Συνεπώς, η υπηρεσία είναι μια τοποθεσία στο δίκτυο που έχει μια αναγνώσιμη από μηχανή (machine-readable) περιγραφή των μηνυμάτων που λαμβάνει και προαιρετικά επιστρέφει. Ένα σχήμα για τα δεδομένα που περιέχονται στο μήνυμα χρησιμοποιείται ως το κύριο μέρος μιας συμφωνίας ή ενός συμβολαίου που έχει συσταθεί ανάμεσα σε έναν αιτούντα μιας υπηρεσίας (service requester) και σε έναν πάροχο της (service provider). Μεταδεδομένα περιγράφουν τη διεύθυνση δικτύου για την υπηρεσία, τις λειτουργίες που υποστηρίζονται και τις απαιτήσεις για την αξιοπιστία, την ασφάλεια και τις συναλλαγές.
Μια υπηρεσία μπορεί να οριστεί επίσης με βάση τις ιδιότητες της:

- Καθορίζεται από μια διεπαφή (service interface) πιθανώς ανεξάρτητης της πλατφόρμας υλοποίησης.
- Είναι διαθέσιμη σε ένα δίκτυο.
- Οι λειτουργίες που καθορίζονται στη διεπαφή φέρουν εις πέρας επιχειρησιακές λειτουργίες χρησιμοποιώντας επιχειρησιακά αντικείμενα.
- Η διεπαφή και η υλοποίησή της μπορούν να εμπλουτιστούν με επεκτάσεις που

χρησιμοποιούνται κατά το χρόνο εκτέλεσης.

Φαίνεται ότι μια υπηρεσία αξίζει τον κόπο να καθοριστεί και να δημιουργηθεί μόνο αν θα επαναχρησιμοποιηθεί. Ένας κύριος στόχος της SOA είναι οι συνιστώσες της να είναι αυτόνομες και να μην παρουσιάζουν επικάλυψη λειτουργικότητας (overlapping functionality). Οι συνιστώσες της θα πρέπει να παρέχουν ένα βαθμό επαναχρησιμοποίησης (reusability). Δηλαδή, δεν πρέπει να προορίζονται για ένα πολύ περιορισμένο πλαίσιο της επιχείρησης, που έχει μικρή ή καμία σημασία σε άλλα σημεία της επιχείρησης.

2.1.2 Ορισμός της προσανατολισμένης στις υπηρεσίες αρχιτεκτονικής (Service Oriented Architecture - SOA)

Η SOA αρχιτεκτονική είναι ένα είδος σχεδίασης, που ρυθμίζει τα θέματα της δημιουργίας και χρήσης υπηρεσιών καθ' όλη τη διάρκεια της ζωής τους (από τη σύλληψη της ιδέας για την υπηρεσία μέχρι την κατάργηση της υπηρεσίας). Ειδικότερα, τα στάδια του κύκλου ζωής μιας υπηρεσίας, όπου συμμετέχει η SOA, είναι τα ακόλουθα:

- Σύλληψη (Conception)
- Μοντελοποίηση (Modeling)
- Σχεδίαση (Design)
- Ανάπτυξη (Development)
- Εγκατάσταση (Deployment)
- Διαχείριση (Management)
- Διαχείριση εκδόσεων (Versioning)
- Κατάργηση (Retirement)

Μια υπηρεσιοστρεφής αρχιτεκτονική είναι επίσης ένας τρόπος να καθοριστεί και να προβλεφθεί μια υποδομή ΤΠ που θα επιτρέπει την ανταλλαγή δεδομένων σε εφαρμογές υλοποιημένες σε διαφορετικά περιβάλλοντα, καθώς και τη συμμετοχή σε επιχειρησιακές διαδικασίες, ανεξάρτητα από το λειτουργικό σύστημα και τις γλώσσες προγραμματισμού που χρησιμοποιούνται. Για την αρχιτεκτονική αυτή οι επιχειρησιακές υπηρεσίες είναι η κύρια οργανωτική αρχή που χρησιμοποιείται για να ανταποκρίνονται τα συστήματα ΤΠ στις ανάγκες της επιχείρησης. Αντίθετα, οι προηγούμενες προσεγγίσεις για την ανάπτυξη συστημάτων ΤΠ όπως:

- ο προσανατολισμός στα αντικείμενα (object-oriented),
- ο προσανατολισμός στις διαδικασίες (process-oriented) και
- ο προσανατολισμός στα μηνύματα (message-oriented)

συνήθιζαν να χρησιμοποιούν συγκεκριμένα περιβάλλοντα υλοποίησης με αποτέλεσμα συστήματα άρρηκτα συνδεδεμένα με τα χαρακτηριστικά και τις συναρτήσεις της τεχνολογίας ενός συγκεκριμένου περιβάλλοντος εκτέλεσης όπως CICS, IMS, CORBA, J2EE και COM/DCOM.

Ένας πολύ σύντομος, αλλά ορθός, ορισμός του τι είναι η προσανατολισμένη στις υπηρεσίες αρχιτεκτονική είναι ο εξής:

«είναι ένα είδος αρχιτεκτονικής που χρησιμοποιεί υπηρεσίες ως δομικά στοιχεία για τη διευκόλυνση της επιχειρησιακής ολοκλήρωσης (enterprise integration) και την επαναχρησιμοποίηση των συνιστωσών μέσω χαλαρής σύνδεσης.»

2.2 Η προσανατολισμένη στις υπηρεσίες αρχιτεκτονική διευκολύνει την επαναχρησιμοποίηση μέσω της χαλαρής σύνδεσης (loose coupling)

Η προσανατολισμένη στις υπηρεσίες αρχιτεκτονική εισάγει δύο διακριτές ιδέες: κατ' αρχάς διευκολύνει την επαναχρησιμοποίηση και το καταφέρνει αυτό δημιουργώντας χαλαρά

συνδεδεμένα συστατικά μέρη (loosely coupled components).

- Επαναχρησιμοποίηση

Στο παρελθόν οι προσπάθειες που γίνονταν με τους πρώτους παρόχους υπηρεσιών δεν εστίαζαν στην ιδέα της επαναχρησιμοποίησης. Πλέον τα νέα πρότυπα και η καθιέρωση μιας γενικής μορφής όπως η XML για τις ανταλλαγές μηνυμάτων συνέβαλλαν στην κατανόηση των επιχειρησιακών υπηρεσιών με έναν αρκετά γενικό τρόπο ώστε να είναι εφικτή η επαναχρησιμοποίηση τους.
Η επαναχρησιμοποίηση αυτή μπορεί να έρθει με τη μορφή της διαλειτουργικότητας (interoperability). Αν δύο συστατικά στοιχεία λογισμικού (software elements) είναι διαλειτουργικά, αυτό σε καμία περίπτωση δεν σημαίνει ότι είναι χαλαρά συνδεδεμένα. Οι χαλαρά συνδεδεμένες συνιστώσες όμως, έχουν μεγαλύτερη πιθανότητα να είναι διαλειτουργικές.
Ωστόσο, υπάρχει σαφής διαφορά των υπηρεσιών στη SOA αρχιτεκτονική από τις τυπικές προσπάθειες ολοκλήρωσης, που στοχεύουν συνήθως σε ένα μοναδικό σύνολο γνωστών διεπαφών και όχι σε υπηρεσίες που θα πρέπει να λειτουργούν αρμονικά με άγνωστα ή απρόβλεπτα συστατικά μέρη (components).

- Χαλαρή σύνδεση υπηρεσιών

Η χαλαρή σύνδεση είναι στην πραγματικότητα μια ευρεία έννοια που αναφέρεται σε διαφορετικά στοιχεία μιας υπηρεσίας, στην υλοποίηση της και στη χρήση της.
Η «σύνδεση διεπαφής» αναφέρεται στη διασύνδεση ανάμεσα στους αιτούντες κάποια υπηρεσία και στους παρόχους υπηρεσίας. Είναι μέτρο των εξαρτήσεων που επιβάλλει ο πάροχος στον αιτούντα. Ιδανικά, ο πελάτης που ζητάει υπηρεσία πρέπει να είναι ικανός να χρησιμοποιήσει μια υπηρεσία βασιζόμενος μόνο στο δημοσιευμένο συμβόλαιο υπηρεσίας και τη συμφωνία επιπέδου υπηρεσίας και σε καμία περίπτωση δεν πρέπει να απαιτεί τη γνώση πληροφοριών για την εσωτερική υλοποίησή της. Με άλλα λόγια, η διεπαφή πρέπει να συμπυκνώνει όλες τις πληροφορίες υλοποίησης και να τις αποκρύπτει από τους αιτούντες την υπηρεσία.

Η «σύνδεση της τεχνολογίας» μετράει το βαθμό στον οποίο μια ορισμένη υπηρεσία εξαρτάται από μια συγκεκριμένη τεχνολογία, ένα προϊόν και μια πλατφόρμα ανάπτυξης. Η «σύνδεση με τις διαδικασίες» μετράει το βαθμό στον οποίο μια υπηρεσία είναι συνδεδεμένη με μια συγκεκριμένη επιχειρησιακή διαδικασία (business process). Ιδανικά, μια υπηρεσία μπορεί να επαναχρησιμοποιηθεί σε πολλές διαφορετικές διαδικασίες και εφαρμογές.

2.3 Προδιαγραφές (Specifications)

Υπάρχουν δεκάδες προδιαγραφές που εμφανίζονται στο προσκήνιο στον κόσμο της SOA αρχιτεκτονικής και των βασισμένων σε Java υπηρεσιών ιστού. Η σύνδεση με τις προδιαγραφές μπορεί να εξασφαλίσει ότι ο καθένας θα μπορεί εύκολα να βρει την καθοριστική απάντηση σε μια ερώτηση και έμμεσα καθοδηγεί τους ανθρώπους στο τι πρέπει να κατανοήσουν για να είναι παραγωγικοί σε ένα περιβάλλον SOA. Οι σχετικές Java προδιαγραφές ενδέχεται να περιλαμβάνουν τα ακόλουθα:

- JAX-WS
- JAXB
- SAAJ
- JAXR
- Web Services
- WS-Annotations
- DOM, SAX, StAX

Αυτές είναι μόνο οι προδιαγραφές που σχετίζονται με την Java. Οι πιο σημαντικές SOAP και WS-* προδιαγραφές περιλαμβάνουν τις:

- XML Schema
- XSLT
- SOAP
- WSDL
- OASIS UDDI
- WS-Policy
- BPEL
- WS-Security
- WS-Trust
- WS-ReliableMessaging
- WS-Transaction
- WS-MetadataExchange

2.4 Επισκόπηση επιλεγμένων τεχνολογιών που έχουν χρησιμοποιηθεί για την υλοποίηση SOA

Με την πάροδο του χρόνου, πλήθος διαφορετικών τεχνολογιών έχουν χρησιμοποιηθεί για την υλοποίηση SOAs. Στη συνέχεια αναφέρονται κατηγορίες τέτοιων τεχνολογιών και παραδείγματα για την καθεμία.

- Κατανεμημένα αντικείμενα (distributed objects) – CORBA, J2EE , COM/DCOM
- Ενδιάμεσο λογισμικό (mediation software) προσανατολισμένο στην ανταλλαγή μηνυμάτων (Message-oriented middleware - MOM) – IBM WebSphere MQ, IBM Tibco Rendezvous
- Ελεγκτές επεξεργασίας συναλλαγών (Transaction Processing monitors) – CICS, IMS, Encinia, Tuxedo
- Ενδιάμεσο λογισμικό που αναπτύσσεται κατ' οίκον
- Πλατφόρμες επιχείρηση προς επιχείρηση (B2B)

2.4.1 IBM WebSphere MQ

Καθώς το WebSphere MQ είναι απλά ενδιάμεσο λογισμικό προσανατολισμένο στην ανταλλαγή μηνυμάτων, ο πελάτης ΤΠ απαιτείται να παρέχει τα περισσότερα από την παρακάτω λίστα στην πλατφόρμα υπηρεσίας:

- Καθορισμός του πως ορίζονται τα συμβόλαια (συνήθως ένα έγγραφο Word ή Excel).
- Ορισμός των μορφών μηνυμάτων (π.χ. συμβολοσειρές που χωρίζονται με κόμμα, μηνύματα προκαθορισμένης μορφής, και πιο πρόσφατα XML).
- Ορισμός του πώς οι υπηρεσίες θα καταχωρούνται και θα ανακαλύπτονται (συνήθως ένα μητρώο).
- Παροχή μηχανισμών ασφαλείας επιπέδου υπηρεσίας (συμπεριλαμβανομένων αυθεντικοποίησης, κρυπτογράφησης, εξουσιοδότησης κ.α.)
- Παροχή διευκολύνσεων διαχείρισης επιπέδου υπηρεσίας.

Οι λόγοι χρήσης του WebSphere MQ ως βάση για μια αρχιτεκτονική προσανατολισμένη στις υπηρεσίες είναι:

- η συνάφεια προμηθευτή τεχνολογιών πληροφορίας (IT vendor affinity) (καθώς οι πιο πολλοί οργανισμοί που το χρησιμοποιούν είναι παλιοί πελάτες της IBM),
- η συνάφεια με το κεντρικό υπολογιστικό σύστημα (mainframe affinity) αφού χρησιμοποιείται για πρόσβαση συναλλαγών υλοποιημένων σε CICS ή IMS,
- η διαθεσιμότητα σε πολλές διαφορετικές πλατφόρμες,

- η ενίσχυση της χαλαρής σύνδεσης,
- η πολύ απλή διεπαφή προγραμματισμού εφαρμογών (API) του που μπορούν να μάθουν εύκολα οι προγραμματιστές,
- η εξασφάλιση της αξιόπιστης παράδοσης μηνυμάτων και τέλος
- η διαθεσιμότητα προγραμμάτων για την ασφάλεια και τη διαχείριση του WebSphere MQ.

2.4.2 CORBA

Η χρήση της CORBA (Common Object Request Broker Architecture) στην υλοποίηση μιας SOA προϋποθέτει να γίνει κατανοητό ότι πέρα από τον παραδοσιακό ρόλο αυτής στην τεχνολογία κατανεμημένων αντικειμένων, παρέχει πολλά από τα βασικά συστατικά μιας πλατφόρμας υπηρεσίας συμπεριλαμβανομένων των ακόλουθων:

- Είναι ένα ανοιχτό πρότυπο.
- Υποστηρίζει την απομακρυσμένη κλήση μεθόδων (Remote Method Invocation - RMI), την ασύγχρονη ανταλλαγή μηνυμάτων (asynchronous communication) και επικοινωνία για δημοσίευση ή συνδρομή.
- Παρέχει ολοκληρωμένη ασφάλεια, υπηρεσίες ονοματοδοσίας, διαχείριση συναλλαγών και αξιόπιστη ανταλλαγή μηνυμάτων.
- Υποστηρίζει πολλές προγραμματιστικές γλώσσες.
- Παρέχει τη δική της γλώσσα ορισμού υπηρεσιών (CORBA IDL).
- Τα αντικείμενα CORBA μπορούν να εκτεθούν ως υπηρεσίες ιστού, λόγω του ότι έχει καθοριστεί αντιστοίχιση μεταξύ CORBA IDL και WSDL.

Ωστόσο υπάρχουν κάποιοι περιορισμοί που θέτει αυτή η αρχιτεκτονική, οι οποίοι είναι ότι γενικά θεωρείται πολύπλοκη, τόσο ο πελάτης που ζητάει την υπηρεσία όσο και ο πάροχος πρέπει να χρησιμοποιούν CORBA και τέλος δεν υποστηρίζεται η XML, η χαλαρή σύνδεση και η ασύγχρονη ανταλλαγή επιχειρησιακών εγγράφων μέσω του Διαδικτύου.

2.4.3 Python

Η Python είναι μια ερμηνευμένη, αντικειμενοστρεφής γλώσσα προγραμματισμού υψηλού επιπέδου με δυναμική σημασιολογία. Οι υψηλού επιπέδου ενσωματωμένες δομές δεδομένων σε συνδυασμό με τη δυναμική πληκτρολόγηση και τη δυναμική σύνδεση καθιστούν την εφαρμογή πολύ ελκυστική για την Ανάπτυξη Rapid Application, καθώς και για τη χρήση ως scripting ή γλώσσας κόλλας για να συνδέσει τα υπάρχοντα στοιχεία μαζί. Η απλή και εύκολη σύνταξη της Python δίνει έμφαση στην αναγνωσιμότητα και επομένως μειώνει το κόστος συντήρησης του προγράμματος. Η Python υποστηρίζει δομοστοιχεία και πακέτα, τα οποία ενθαρρύνουν τη διαμόρφωση του προγράμματος και την επαναχρησιμοποίηση του κώδικα. Ο διερμηνέας της Python και η εκτεταμένη τυποποιημένη βιβλιοθήκη είναι διαθέσιμα σε πηγή ή σε δυαδική μορφή χωρίς χρέωση για όλες τις μεγάλες πλατφόρμες και μπορούν να διανεμηθούν ελεύθερα.
Συχνά, οι προγραμματιστές ερωτεύονται την Python λόγω της αυξημένης παραγωγικότητας που παρέχει. Δεδομένου ότι δεν υπάρχει κανένα βήμα σύνταξης, ο κύκλος επεξεργασίας-δοκιμής-εντοπισμού σφαλμάτων είναι απίστευτα γρήγορος. Τα προγράμματα εντοπισμού σφαλμάτων Python είναι εύκολα: ένα σφάλμα ή κακή είσοδος δεν θα προκαλέσει ποτέ σφάλμα κατάτμησης. Αντίθετα, όταν ο διερμηνέας ανακαλύψει ένα σφάλμα, δημιουργεί μια εξαίρεση. Όταν το πρόγραμμα δεν καλύψει την εξαίρεση, ο διερμηνέας εκτυπώνει ένα ίχνος στοίβας. Ένα εργαλείο εντοπισμού σφαλμάτων σε επίπεδο πηγής επιτρέπει την επιθεώρηση των τοπικών και παγκόσμιων μεταβλητών, την αξιολόγηση των αυθαίρετων εκφράσεων, τη ρύθμιση των σημείων διακοπής, τη μετάβαση από τον κώδικα σε μια γραμμή τη φορά και ούτω καθεξής. Το πρόγραμμα εντοπισμού

σφαλμάτων είναι γραμμένο στην ίδια την Python, μαρτυρώντας την ενδοσκοπική δύναμη του Python. Από την άλλη πλευρά, συχνά ο ταχύτερος τρόπος για να διορθώσετε ένα πρόγραμμα είναι να προσθέσετε μερικές εκτυπώσεις στην πηγή: ο γρήγορος κύκλος επεξεργασίας-δοκιμής-εντοπισμού καθιστά την απλή αυτή προσέγγιση πολύ αποτελεσματική.

Μειωνεκτήματα :

- Κατανάλωση μνήμης

Η Python δεν είναι μια καλή επιλογή για εργασίες που απαιτούν μνήμη. Λόγω της ευελιξίας των τύπων δεδομένων, η κατανάλωση μνήμης της Python είναι επίσης υψηλή.

- Πρόσβαση σε βάσεις δεδομένων

Η Python έχει περιορισμούς με την πρόσβαση σε βάση δεδομένων. Σε σύγκριση με τις δημοφιλείς τεχνολογίες όπως το JDBC και το ODBC, το επίπεδο πρόσβασης της βάσης δεδομένων της Python βρίσκεται λίγο ανεπτυγμένο και πρωτόγονο. Ωστόσο, δεν μπορεί να εφαρμοστεί στις επιχειρήσεις που χρειάζονται ομαλή αλληλεπίδραση των σύνθετων δεδομένων παλαιού τύπου.

- Σφάλματα χρόνου εκτέλεσης

Οι προγραμματιστές της Python ανέφεραν πολλά θέματα με το σχεδιασμό της γλώσσας. Επειδή η γλώσσα πληκτρολογείται δυναμικά, απαιτείται περισσότερη δοκιμή και παρουσιάζει σφάλματα που εμφανίζονται μόνο κατά το χρόνο εκτέλεσης.

2.4.4 Ruby

Ο Ruby σχεδιάστηκε αρχικά με στόχο να κάνει διασκέδαση προγραμματισμού, και στην Ιαπωνία, από όπου προέρχεται, ο Ruby συνηθίζει να κάνει παιχνίδια. Ο Ruby είναι σύντομος γεγονός που καθιστά τον κώδικα εύκολο στην κατανόηση για αρχάριους κωδικοποίησης.Δεδομένου ότι θα είστε σε θέση να δημιουργήσετε γρήγορα πρωτότυπα με το Ruby on Rails, πολλοί βρίσκουν στην Ruby μια ικανοποιητική εμπειρία.

- Flexibility

Ως μια δυναμικά πληκτρολογημένη γλώσσα, ο Ruby δεν έχει σκληρούς κανόνες για τον τρόπο κατασκευής χαρακτηριστικών και είναι πολύ κοντά στις προφορικές γλώσσες. Θα έχετε μεγαλύτερη ευελιξία για την επίλυση προβλημάτων χρησιμοποιώντας διαφορετικές μεθόδους. Επιπλέον, ο Ruby είναι επίσης πιο συγχωριασμένος από σφάλματα, έτσι θα είστε σε θέση να συντάξετε και να εκτελέσετε το πρόγραμμά σας μέχρι να βρεθείτε στο προβληματικό μέρος.

- Scalability

Δεν είναι εύκολο να διατηρηθεί, επειδή το Ruby είναι μια δυναμικά πληκτρολογημένη γλώσσα, το ίδιο πράγμα μπορεί εύκολα να σημαίνει κάτι διαφορετικό ανάλογα με το πλαίσιο. Καθώς η εφαρμογή Ruby μεγαλώνει και είναι πιο περίπλοκη, αυτό μπορεί να γίνει δύσκολο να διατηρηθεί καθώς τα σφάλματα θα είναι δύσκολο να εντοπιστούν και να διορθωθούν, οπότε θα χρειαστεί εμπειρία και διορατικότητα για να γνωρίζετε πώς να σχεδιάσετε τον κώδικα ή να γράψετε δοκιμές μονάδας για να διευκολύνετε τη συντηρησιμότητα. Ωστόσο, μπορείτε να μάθετε πώς να σχεδιάζετε καλύτερα τον κώδικα δουλεύοντας με έμπειρο μέντορα του Ruby.

- Slow

Ως μια δυναμικά πληκτρολογημένη γλώσσα, ο Ruby είναι αργός επειδή είναι πολύ ευέλικτος και το μηχάνημα θα χρειαστεί να κάνει πολλές αναφορές για να βεβαιωθεί τι είναι ο ορισμός του κάτι και αυτό επιβραδύνει την απόδοση του Ruby. Το Rails είναι γενικά πιο πεινασμένο από πόρους.
Εν πάση περιπτώσει, υπάρχουν εναλλακτικές λύσεις όπως το JRuby, το οποίο είναι ταχύτερη εφαρμογή του Ruby. Παρόλο που αυτό πιθανότατα δεν είναι τόσο γρήγορο όσο η Java, για παράδειγμα, είναι ακόμα μια τεράστια βελτίωση.

- Community

Πρώτα απ 'όλα, το μέγεθος της κοινότητας είναι σημαντικό, διότι όσο μεγαλύτερη είναι η κοινότητα των γλωσσών προγραμματισμού, τόσο περισσότερη υποστήριξη θα είναι πιθανό να πάρετε. Καθώς μπαίνετε στον κόσμο του προγραμματισμού, σύντομα θα καταλάβετε πόσο σημαντική είναι η υποστήριξη, καθώς η κοινότητα προγραμματιστών έχει να κάνει με την παροχή βοήθειας. Επιπλέον, όσο μεγαλύτερη είναι η κοινότητα, τόσο περισσότεροι άνθρωποι θα δημιουργήσουν χρήσιμα εργαλεία για να διευκολύνουν την ανάπτυξη της συγκεκριμένης γλώσσας. Από σήμερα, υπάρχουν πάνω από 600 αξιοσημείωτες γλώσσες προγραμματισμού παγκοσμίως.
Έτσι, με αυτό το πλαίσιο στο μυαλό, ας πάρουμε τις λεπτομέρειες του μεγέθους της κοινότητας Ruby.

2.4.5 Apache Groovy

Το Apache Groovy είναι μια ισχυρή, προαιρετικά δακτυλογραφημένη και δυναμική γλώσσα, με δυνατότητες στατικής πληκτρολόγησης και στατικής συλλογής, για την πλατφόρμα Java που στοχεύει στη βελτίωση της παραγωγικότητας των προγραμματιστών χάρη σε μια σύντομη, οικεία και εύκολη σύνταξη. Ενσωματώνεται ομαλά με οποιοδήποτε πρόγραμμα Java και παρέχει άμεσα στην εφαρμογή σας δυνατότητες, όπως δυνατότητες δέσμης ενεργειών, συγγραφή γλωσσών για το συγκεκριμένο τομέα, μετα-προγραμματισμό χρόνου εκτέλεσης και μεταγλωττισμών και λειτουργικό προγραμματισμό.

2.4.6 Java και J2EE

Οι τεχνολογίες Java και J2EE έχουν πολλά από τα πλεονεκτήματα και μειονεκτήματα που έχουν και οι παραπάνω γλώσσες όσον αφορά την υλοποίηση μιας SOA αρχιτεκτονικής. Μερικές από τις ομοιότητες αυτές είναι:

- Είναι ανοιχτά πρότυπα.
- Είναι τεχνολογίες κατανεμημένων αντικειμένων που παρέχουν τέλεια υποστήριξη για απομακρυσμένη κλήση μεθόδων

- Απαιτούν ο πάροχος και ο αιτών την υπηρεσία να χρησιμοποιούν την ίδια τεχνολογία
- Παρέχουν ενσωματωμένη ασφάλεια, υπηρεσίες ονοματοδοσίας, διαχείρισης συναλλαγών και αξιόπιστης παράδοσης μηνυμάτων.

Κάποιες διαφορές που σχετίζονται με την SOA είναι:

- Η Java Community Process έχει καθορίσει μια σειρά API για το χειρισμό XML.
- Η J2EE έχει μεγαλύτερη και πιο εύρωστη προγραμματιστική κοινότητα.
- J2EE υλοποιήσεις είναι διαθέσιμες από τους περισσότερους προμηθευτές ΤΠ.

2.4.7 Πλατφόρμες υπηρεσίας προς υπηρεσία (B2B platforms)

Αν και συχνά παραλείπονται όταν γίνεται συζήτηση για SOA, οι πλατφόρμες αυτές (όπως η ebXML και η RosettaNet) είναι ιδανικές για υπηρεσίες ιστού γιατί:

- Είναι ανοιχτά πρότυπα
- Είναι χαλαρά συνδεδεμένες
- Βασίζονται σε XML
- Βασίζονται στην ασύγχρονη ανταλλαγή επιχειρησιακών εγγράφων

Παρέχουν ολοκληρωμένους μηχανισμούς για την καταχώρηση υπηρεσιών, την ασφάλεια, την παρακολούθηση και τη διεύθυνση υπηρεσιών και επιχειρησιακών διαδικασιών, την επαναφορά συναλλαγών και την αξιόπιστη ανταλλαγή μηνυμάτων.

2.5 Η προσανατολισμένη στις υπηρεσίες δραστηριότητα (Service Oriented Enterprise)

Με την χρήση υπηρεσιών ιστού, η προσανατολισμένη στις υπηρεσίες δραστηριότητα υπόσχεται να βελτιώσει σημαντικά την εταιρική ευελιξία, να επιταχύνει το χρόνο διάθεσης στην αγορά νέων προϊόντων και υπηρεσιών, να μειώσει το κόστος της τεχνολογίας πληροφοριών (Information Technology - IT) και να βελτιώσει την αποδοτικότητα. Διάφορες τεχνολογικές τάσεις συγκλίνουν για να οδηγήσουν στις θεμελιώδεις αλλαγές της τεχνολογίας που αφορούν τις έννοιες και την υλοποίηση του προσανατολισμού στις υπηρεσίες. Οι κύριες τεχνολογίες που συμμετέχουν σ' αυτήν τη σύγκλιση είναι:

- Η Επεκτάσιμη Γλώσσα Σήμανσης (eXtensible Markup Language – XML): μια κοινή, εντός και εκτός της επιχείρησης, μορφή δεδομένων που παρέχει πρότυπους τύπους και δομές ανεξάρτητα από γλώσσα προγραμματισμού, περιβάλλον ανάπτυξης και λειτουργικό σύστημα. Επίσης, παρέχει τεχνολογία για ορισμό επιχειρησιακών αρχείων και ανταλλαγές επιχειρησιακών πληροφοριών καθώς και λογισμικό για χειρισμό λειτουργιών σε XML.
- Οι Υπηρεσίες Ιστού (Web Services): βασισμένες σε XML τεχνολογίες για ανταλλαγή μηνυμάτων, περιγραφή υπηρεσιών, ανακάλυψη υπηρεσιών και επιπλέον χαρακτηριστικά που παρέχουν μεταξύ άλλων ανεξαρτησία από την υποκείμενη τεχνολογία και τις πλατφόρμες εφαρμογής και επεκτασιμότητα προκειμένου να πληρούν τις εταιρικές ποιότητες υπηρεσίας (qualities of service).
- Η προσανατολισμένη στις υπηρεσίες ή υπηρεσιοστρεφής αρχιτεκτονική (Service Oriented Architecture – SOA): μια μεθοδολογία για την επίτευξη διαλειτουργικότητας εφαρμογών και επαναχρησιμοποίησης της υπάρχουσας τεχνολογίας.
- Η διαχείριση επιχειρησιακών διεργασιών (Business Process Management - BPM): αφορά μεθοδολογίες και τεχνολογίες που αυτοματοποιούν τις επιχειρησιακές διεργασίες.

Κάθε μια από τις τεχνολογίες αυτές έχει έντονη επίδραση σε τουλάχιστον ένα τομέα της επιχειρησιακής χρήσης υπολογιστών. Όταν συνδυαστούν οι ανωτέρω τεχνολογίες παρέχουν μια περιεκτική πλατφόρμα, που αποκομίζει τα οφέλη του προσανατολισμού στις

υπηρεσίες (service orientation) και συμβάλλει στην εξέλιξη των συστημάτων ΤΠ (IT systems).

2.6 Προσανατολισμένη στις υπηρεσίες ανάπτυξη (Service Oriented Development)

Η προσανατολισμένη στις υπηρεσίες ανάπτυξη συμπληρώνει τις αντικειμενοστραφείς (object-oriented), διαδικαστικές (procedure-oriented), προσανατολισμένες στα μηνύματα (message-oriented), και προσανατολισμένες στις βάσεις δεδομένων (database-oriented) προσεγγίσεις ανάπτυξης οι οποίες είχαν προηγηθεί.
Η προσανατολισμένη στις υπηρεσίες ανάπτυξη παρέχει τα ακόλουθα οφέλη:

- Επαναχρησιμοποίηση (reusability): Όπως αναλύθηκε στην ενότητα 0, είναι δυνατόν να δημιουργηθούν υπηρεσίες που μπορούν να ξαναχρησιμοποιηθούν σε πολλαπλές εφαρμογές.
- Αποδοτικότητα (throughput): Η ικανότητα να δημιουργηθούν εύκολα και γρήγορα νέες υπηρεσίες και εφαρμογές χρησιμοποιώντας ένα συνδυασμό παλιών και νέων υπηρεσιών, μαζί με την δυνατότητα της εστίασης στα δεδομένα που θα διαμοιραστούν και όχι στο πως θα υλοποιηθεί η υπηρεσία.
- Χαλαρή σύνδεση της τεχνολογίας (loosely coupling): Όπως επίσης αναλύθηκε στην ενότητα 0, αφορά τη δυνατότητα να μοντελοποιηθούν οι υπηρεσίες ανεξάρτητα από το περιβάλλον εκτέλεσης και η δημιουργία μηνυμάτων που μπορούν να αποσταλούν σε κάθε υπηρεσία.
- Διαχωρισμός των ευθυνών (responsibility discrimination): Είναι πιο εύκολο οι υπάλληλοι της επιχείρησης να επικεντρωθούν στα επιχειρησιακά ζητήματα, οι τεχνικοί να αφοσιωθούν σε θέματα τεχνολογίας και να συνεργαστούν οι δύο αυτές ομάδες βάσει του συμβολαίου της υπηρεσίας (SLA).

Αξίζει να τονισθεί ότι οι υπηρεσίες σχεδιάστηκαν για να λύνουν προβλήματα διαλειτουργικότητας ανάμεσα στις ετερογενείς εφαρμογές και για τη σύνθεση νέων εφαρμογών ή συστημάτων εφαρμογών, αλλά όχι για σχεδιασμό μιας λεπτομερούς επιχειρησιακής λογικής (business logic) για τις εφαρμογές. Η δυνατότητα σύνθεσης επαναχρησιμοποιήσιμων υπηρεσιών σε μεγαλύτερες υπηρεσίες εύκολα και γρήγορα είναι αυτή που παρέχει στους οργανισμούς τα οφέλη της αυτοματοποίησης της διαδικασίας (process automation) και την ευελιξία (flexibility) να ανταποκρίνονται στις διαρκώς μεταβαλλόμενες συνθήκες.

2.7 Η σημασία των υπηρεσιών για τις επιχειρήσεις

Πριν γίνει αναφορά στην τεχνολογία SOA, θα εξεταστεί η έννοια των υπηρεσιών και διαδικασιών από την οπτική γωνία της επιχείρησης (business-wise). Οι περισσότεροι οργανισμοί (είτε εμπορικοί είτε κυβερνητικοί) παρέχουν υπηρεσίες σε πελάτες, πολίτες, υπαλλήλους ή συνεταίρους τους.
Οι υπηρεσίες σχεδιάζονται, υλοποιούνται και εγκαθίστανται (design-implement-deploy) ώστε να ανταποκρίνονται στις υπηρεσίες που χρειάζονται οι πελάτες. Χρησιμοποιώντας το παράδειγμα της λειτουργίας μιας τράπεζας, ο ορισμός των υπηρεσιών λογισμικού πρέπει να ευθυγραμμίζεται με τις επιχειρησιακές υπηρεσίες (software to business alignment) που μια τράπεζα προσφέρει για να εξασφαλίσει την ομαλή της λειτουργία και για να συμβάλλει στην αναγνώριση των στρατηγικών στόχων, όπως η παροχή τραπεζικών υπηρεσιών τόσο στα υποκαταστήματα της τράπεζας όσο και μέσω των ATM και της ιστοσελίδας της.
Η ανάπτυξη υπηρεσιών στο πλαίσιο της SOA αρχιτεκτονικής διευκολύνει τη σύνθεση υπηρεσιών σε απλές ή πολύπλοκες εφαρμογές (atomic or composite) που με τη σειρά τους μπορούν να προσφερθούν ως υπηρεσίες προσπελάσιμες από ανθρώπους και συστήματα ΤΠ.

Η χρήση των υπηρεσιών δεν μειώνει μόνο το πλήθος κώδικα που αναπτύσσεται αλλά επιπλέον μειώνει το φόρτο διαχείρισης, συντήρησης και υποστήριξης. Ωστόσο, πρέπει να εξεταστεί ενδελεχώς η επιρροή (performance impact) στην απόδοση που επιβάλλει η εγκαθίδρυση της τεχνολογίας αυτής, που αντικαθιστά την κλασσική χρήση εσωτερικών συναρτήσεων. Η υιοθέτηση της SOA αρχιτεκτονικής ενδεχομένως να αυξάνει την ανάγκη παροχής υπολογιστικών πόρων από ότι μια επαναχρησιμοποιήσιμη βιβλιοθήκη κώδικα. Αυτό οφείλεται στην πρόσθεση του ενδιάμεσου επιπέδου (mediation layer), ώστε να γίνει δυνατή η σύνδεση και προσφορά νέων πολύπλοκων υπηρεσιών σε εξωτερικά συστήματα.

2.8 Αν η προσανατολισμένη στις υπηρεσίες αρχιτεκτονική (SOA) δεν είναι νέα έννοια, τότε τι έχει αλλάξει;

Η ιδέα της προσανατολισμένης στις υπηρεσίες αρχιτεκτονικής δεν είναι καινούργια – η καινοτομία που εισάγει είναι η δυνατότητα της ολοκλήρωσης διαφορετικών περιβαλλόντων εκτέλεσης, διαχωρίζοντας ξεκάθαρα τη διεπαφή της υπηρεσίας από την τεχνολογία εκτέλεσης (interface and execution environment). Στο πρόσφατο παρελθόν όλες οι υλοποιήσεις της αρχιτεκτονικής βασίζονταν σε μια μοναδική τεχνολογία του περιβάλλοντος εκτέλεσης.
Αυτή η ιδέα του διαχωρισμού διεπαφής και υλοποίησης έχει υλοποιηθεί (realization) στα J2EE, CORBA, COM και ακόμα και στο DCE που είναι προγενέστερο. Ωστόσο, η δυνατότητα να διαχωριστεί ολοκληρωτικά η περιγραφή της υπηρεσίας από το περιβάλλον εκτέλεσης είναι νέα προοπτική. Σε πολλές περιπτώσεις το θέμα της απόδοσης (που μέχρι τώρα δρούσε ανασταλτικά στον ουσιαστικό διαχωρισμό) είναι λιγότερο σημαντικό από τη δυνατότητα επίτευξης της διαλειτουργικότητας.
Πλέον οι επιχειρήσεις θα μπορούν να σκέφτονται και να σχεδιάζουν επενδύσεις στον τομέα της τεχνολογίας πληροφοριών, όπως αυτές εκφράζονται από την περιγραφή τους. Σε αυτή την περίπτωση η περιγραφή γίνεται ενός είδους ελάχιστος κοινός παρονομαστής – δηλαδή ένα σύνολο από λειτουργίες και συναρτήσεις που μπορούν να υποστηριχθούν ανεξάρτητα της χρησιμοποιούμενης τεχνολογίας και του περιβάλλοντος υλοποίησης. Η SOA που χρησιμοποιεί υπηρεσίες ιστού υποστηρίζει τη διαφορετικότητα και δίνει τη δυνατότητα δημιουργίας συστημάτων, που αντιστοιχούν σε επιχειρησιακές λειτουργίες καλύτερα από οτιδήποτε πριν από αυτά. Ωστόσο, αυτό απαιτεί μια σημαντική αλλαγή στον τρόπο σκέψης όχι μόνο για τα τμήματα ΤΠ, αλλά για ολόκληρη τη βιομηχανία λογισμικού. Οι επιχειρήσεις θα χρειαστεί να μάθουν όχι μόνο να σκέφτονται τις υπηρεσίες ανεξάρτητα από το περιβάλλον εκτέλεσης, αλλά επίσης το πώς να συνενώνουν τα διάφορα κομμάτια μιας εφαρμογής που θα προέρχονται από διαφορετικούς παρόχους (service providers/business partners).

2.9 Διαχείριση μεταδεδομένων (metadata management)

Η ολοκλήρωση ετερογενών συστημάτων και η επικοινωνία τους, επιτυγχάνεται συνήθως με την ανταλλαγή μηνυμάτων που περιγράφονται από μετα-γλώσσες όπως η XML. Ο ενδιαφερόμενος αναγνώστης παραπέμπεται στο αντίστοιχο εδάφιο της παρούσας εργασίας. Η διαχείριση των μεταδεδομένων περιλαμβάνει κυρίως τη διαχείριση των πληροφοριών περιγραφής για μια υπηρεσία ιστού, οι οποίες είναι απαραίτητες για την κατασκευή του σώματος ενός μηνύματος (message body) και της επικεφαλίδας (header) του μηνύματος έτσι ώστε ένας αιτούμενος την υπηρεσία (service requester) να μπορεί να την καλέσει. Ο πάροχος (provider) της υπηρεσίας δημοσιεύει (publish) τα μεταδεδομένα έτσι ώστε ο αιτών να μπορεί να τα ανακαλύψει (discover) και να τα χρησιμοποιήσει για να κατασκευάσει μηνύματα που μπορεί να επεξεργαστεί επιτυχώς ο πάροχος.
Όταν καλείται μια υπηρεσία είναι σημαντικό να γίνει κατανοητό όχι μόνο ποιοι τύποι δεδομένων και δομές θα σταλούν αλλά και τα ποιοτικά χαρακτηριστικά (Quality of Service

– QoS characteristics) όπως είναι η ασφάλεια και η αξιοπιστία πάνω στα οποία θα πραγματοποιηθεί μια επικοινωνία.
Οι προδιαγραφές μεταδεδομένων περιλαμβάνουν:

- XML Schema – Κυρίως για τύπους και δομές δεδομένων.
- Γλώσσα Περιγραφής Υπηρεσιών Ιστού (WSDL) – Για τη συσχέτιση μηνυμάτων και σχεδίων ανταλλαγής μηνυμάτων με ονόματα υπηρεσιών και διευθύνσεις δικτύου.
- Διευθυνσιοδότηση υπηρεσιών ιστού (WS-Addressing) – Για τη συμπερίληψη διευθυνσιοδότησης τερματικού σημείου (endpoint) και ιδιότητες αναφορών που σχετίζονται με τερματικά σημεία.
- Πολιτική υπηρεσιών ιστού (WS-Policy) – Για τη συσχέτιση των απαιτήσεων ποιότητας υπηρεσίας με ένα WSDL ορισμό. Είναι ένα πλαίσιο που περιλαμβάνει δηλώσεις πολιτικής/τακτικής για διάφορα θέματα ασφάλειας, τις συναλλαγές, και την αξιοπιστία.
- Ανταλλαγή μεταδεδομένων υπηρεσιών ιστού (WS-Metadata Exchange) – Για την επερώτηση και ανακάλυψη μεταδεδομένων που σχετίζονται με μια υπηρεσία ιστού. Περιλαμβάνει τη δυνατότητα της ανάκτησης ενός WSDL αρχείου και των σχετικών ορισμών πολιτικής υπηρεσιών ιστού. 0

2.10 Αξιοπιστία και ανταλλαγή μηνυμάτων

Η βιομηχανία της πληροφορικής δεν έχει καταλήξει ομόφωνα σε ένα μοναδικό, ενοποιημένο σύνολο από προδιαγραφές για προηγμένη ανταλλαγή μηνυμάτων.
Η ανταλλαγή μηνυμάτων για την εγκαθίδρυση μιας επικοινωνίας σε SOA αρχιτεκτονική πραγματοποιείται συνήθως με τη χρήση SOAP μηνυμάτων, που είναι ένα XML-βασισμένο σχήμα. Στο υποκεφάλαιο 0 γίνεται περιγραφή της προδιαγραφής των SOAP μηνυμάτων.
Γενικά, η αξιόπιστη ανταλλαγή μηνυμάτων είναι ο μηχανισμός που εγγυάται ότι ένα ή περισσότερα SOAP μηνύματα λήφθηκαν όσες φορές έπρεπε σε ένα πιθανώς αναξιόπιστο δίκτυο όπως τα HTTP δίκτυα, όπως ο παγκόσμιος ιστός. Οι προδιαγραφές της αξιόπιστης ανταλλαγής μηνυμάτων περιλαμβάνουν τα: WS-Reliability και WS-ReliableMessaging που αρχικά υποστηρίχθηκαν από την IBM και τη Microsoft. Το πρωτόκολλο αυτό εγγυάται την παράδοση των μηνυμάτων, τη μη ύπαρξη διπλότυπων μηνυμάτων, και τη σωστή τους ταξινόμηση. Αυτό είναι εφικτό με την ομαδοποίηση των μηνυμάτων με το ίδιο αναγνωριστικό, τη συλλογή μηνυμάτων σε ομάδες βασιζόμενοι στον αριθμό μηνύματος και την ταξινόμηση με βάση τον αριθμό ακολουθίας.
Η αξιόπιστη ανταλλαγή μηνυμάτων αυτοματοποιεί την ανάκαμψη (recovery) από συγκεκριμένα λάθη επιπέδου μεταφοράς (transportation level) στα μηνύματα, που σε άλλες αρχιτεκτονικές θα έπρεπε να τα αντιμετωπίσει η εφαρμογή. Επίσης υποστηρίζεται η ιδέα της γεφύρωσης (bridging) δύο ανεξάρτητων πρωτοκόλλων ανταλλαγής μηνυμάτων στο Διαδίκτυο.
Στο γενικό τομέα της ανταλλαγής μηνυμάτων εντάσσονται και προδιαγραφές για εκτεταμένα σχήματα ανταλλαγής μηνυμάτων (Message Exchange Patterns - MEPs) όπως είναι οι ειδοποιήσεις γεγονότων (events notification) και η δημοσίευση (publishing) που επεκτείνουν τη δυνατότητα ασύγχρονης ανταλλαγής μηνυμάτων (asynchronous transactions) των υπηρεσιών ιστού. Οι προδιαγραφές στον τομέα αυτό περιλαμβάνουν τα WS-Eventing και WS-Notification.

2.11 Συναλλαγές (Transactions)

Οι συναλλαγές επιτρέπουν σε πολλαπλές λειτουργίες να αποτύχουν ή να επιτύχουν ως μια ενότητα, όπως επεξεργασία μιας παραγγελίας και μεταφορά χρημάτων. Ένα από τα πιο

σημαντικά θέματα των τεχνολογιών επεξεργασίας συναλλαγών είναι η δυνατότητα τους να επαναφέρουν μια εφαρμογή σε μια γνωστή κατάσταση μετά από μια αποτυχία του λογισμικού ή του υλικού (latest safe state roll-back).
Πολλές εφαρμογές υπηρεσιών ιστού πιθανόν να απαιτούν μόνο τις δυνατότητες επεξεργασίας συναλλαγών που κληρονομούνται από το υποκείμενο περιβάλλον εκτέλεσης, όπως αυτές που παρέχονται από εξυπηρετητές εφαρμογών και βάσεις δεδομένων. Άλλες εφαρμογές απαιτούν πολλαπλές κλήσεις υπηρεσιών ιστού να ομαδοποιούνται σε μια μεγαλύτερη ενότητα συναλλαγής, που περιλαμβάνει ένα πλαίσιο συναλλαγής μέσα στις SOAP επικεφαλίδες έτσι ώστε η συναλλαγή να συντονίζεται σε πολλαπλά περιβάλλοντα εκτέλεσης.
Ο συντονισμός (coordination) είναι ένας μηχανισμός για τον καθορισμό ενός συνεπούς αποτελέσματος για τις σύνθετες εφαρμογές Υπηρεσιών Ιστού. Όταν συμβεί κάποιο σφάλμα ο συντονιστής είναι υπεύθυνος για το πρωτόκολλο επαναφοράς. Οι προδιαγραφές συναλλαγών υπηρεσιών ιστού επεκτείνουν την έννοια του συντονιστή συναλλαγών (transaction coordinator), υιοθετούν το γνωστό πρωτόκολλο εκτέλεσης δύο – φάσεων (two–phase commit) για τις υπηρεσίες ιστού και ορίζουν νέα επεκταμένα πρωτόκολλα συναλλαγών για πιο χαλαρά συνδεδεμένες υπηρεσίες ιστού. Επί του παρόντος, υπάρχουν συντονιστές συναλλαγών στα περισσότερα περιβάλλοντα εκτέλεσης, περιλαμβανομένων των J2EE, .NET Framework και CORBA .
Ο τομέας των συναλλαγών πρέπει να βασίζεται στις παρακάτω προδιαγραφές:

- Οικογένεια Συναλλαγών Υπηρεσιών Ιστού (WS-Transactions) που συντηρείται από τις BEA (που πλέον ανήκει στον όμιλο Oracle), IBM και Microsoft
  - WS-AtomicTransactions
  - WS-BusinessActivity
  - WS-Coordination
- Πλαίσιο Σύνθετης Εφαρμογής Υπηρεσιών Ιστού (WS-Composite Application Framework – WS-CAF) από την πρωτοβουλία OASIS
  - WS-Context
  - WS-CoordinationFramework
  - Ws-TransactionManagement

Και τα δύο σύνολα προδιαγραφών επικεντρώνονται σε έναν εκτεταμένο συντονιστή στον οποίο μπορούν να ενσωματωθούν πρωτόκολλα συναλλαγών. Επίσης και τα δύο σύνολα καθορίζουν ατομικές συναλλαγές (atomic transactions) (π.χ. two-phase commit[7]) και περιλαμβάνουν τη στρατηγική της αποζημίωσης συναλλαγών (compensation transactions)[8] [9].

2.12 Διακυβέρνηση υπηρεσιών, διαδικασίες, οδηγίες, αρχές και μέθοδοι

Πολλοί οργανισμοί πιστεύουν ότι έχουν υιοθετήσει την SOA αρχιτεκτονική απλά και μόνο επειδή χρησιμοποιούν τεχνολογίες Υπηρεσιών Ιστού, όπως το απλό πρωτόκολλο πρόσβασης αντικειμένων (Simple Object Access Protocol -SOAP), τη γλώσσα περιγραφής υπηρεσιών ιστού (Web Service Description Language -WSDL), και/ή την προδιαγραφή παγκόσμιας περιγραφής, ανακάλυψης και ολοκλήρωσης (Universal Description, Discovery and Integration – UDDI). Ωστόσο, η SOA αρχιτεκτονική είναι σύνολο μεθοδολογιών και νοοτροπία, όχι απλά μια εφαρμογή ή τεχνολογία και επηρεάζει σχεδόν κάθε δραστηριότητα ΤΠ συμπεριλαμβανομένων των διαδικασιών, των μεθόδων και των εργαλείων που χρησιμοποιούνται για σχεδίαση, ανάπτυξη, διαχείριση και συντήρηση των συστημάτων ΤΠ.

---

7 http://publib.boulder.ibm.com/infocenter/idshelp/v10/index.jsp?topic=/com.ibm.admin.doc/admin704.htm
8 http://www.soapatterns.org/compensating_service_transaction.php
9 http://goo.gl/G2Xjt

2.12.1 Πολιτικές διακυβέρνησης και διαδικασίες της προσανατολισμένης σε υπηρεσίες αρχιτεκτονικής

Η SOA διακυβέρνηση (SOA governance) είναι μια επέκταση της διακυβέρνησης της τεχνολογίας πληροφοριών, η οποία εστιάζει στη διαχείριση υπηρεσιών και των σχετικών αφηρημένων τύπων επιπέδου υπηρεσίας. Ο σκοπός της είναι το ταίριασμα διακυβέρνησης λογισμικού και επιχειρησιακής διακυβέρνησης ώστε να επιτευχθεί η μέγιστη ευελιξία.

Οι υπηρεσίες πρέπει να διευθύνονται καθ' όλη τη διάρκεια ζωής τους. Οι υπηρεσίες παρέχουν ένα κοινό σύνολο διαχείρισης απαιτήσεων, συμφωνιών επιπέδου υπηρεσίας (Service Level Agreements - SLAs), επιχειρησιακής και τεχνικής επίδοσης και χρησιμοποίησης πόρων.

2.13 Αρχές και κατευθυντήριες γραμμές για τη SOA αρχιτεκτονική

Οι αρχές και οι κατευθυντήριες γραμμές καθοδηγούν τους σχεδιαστές/αρχιτέκτονες και τους προγραμματιστές όταν αυτοί ορίζουν επιχειρησιακές και τεχνικές υπηρεσίες.
Ο πιο σημαντικές από αυτές τις αρχές και οδηγίες αναφέρονται παρακάτω:

- Αρχή 1: Οι υπηρεσίες είναι η κεντρική οργανωτική έννοια μιας SOA αρχιτεκτονικής
- Αρχή 2: Κάθε υπηρεσία ορίζεται από ένα τυπικό συμβόλαιο που ξεχωρίζει με σαφήνεια την παρεχόμενη λειτουργικότητα από την τεχνική υλοποίηση.
- Αρχή 3: Οι υπηρεσίες πρέπει να αλληλεπιδρούν με άλλες υπηρεσίες μόνο μέσω των καλά ορισμένων διεπαφών τους (well defined interfaces).
- Αρχή 4: Οι υπηρεσίες πρέπει να είναι προσπελάσιμες από τυποποιημένες τεχνολογίες που είναι διαθέσιμες σε ευρεία γκάμα περιβαλλόντων. Αυτό επίσης υπονοεί ότι οι μηχανισμοί κλήσης (invocation) όπως τα πρωτόκολλα, οι μεταφορές, οι μηχανισμοί ανακάλυψης και οι περιγραφές υπηρεσίας πρέπει να συμμορφώνονται με τα ευρέως αποδεκτά βιομηχανικά πρότυπα. Αυτό σημαίνει χρήση των SOAP, WSDL, XML, XML Schema, HTTP και UDDI.
- Αρχή 5: Οι υπηρεσίες πρέπει να καθορίζονται σε ένα αφηρημένο επίπεδο που ανταποκρίνεται στις επιχειρησιακές δραστηριότητες του αληθινού κόσμου και σε αναγνωρίσιμες επιχειρησιακές λειτουργίες έτσι ώστε να ταυτίζονται όσο γίνεται περισσότερο οι ανάγκες της επιχείρησης με τις τεχνολογικές δυνατότητες.
- Αρχή 6: Οι υπηρεσίες πρέπει να είναι πλήρως κατανοητές στον αιτούντα την υπηρεσία, δηλαδή στον πελάτη. Τα παραδοσιακά συστήματα (legacy systems) και οι λεπτομέρειες υλοποίησης δεν θα πρέπει να υπαγορεύουν τις υπηρεσίες και τα συμβόλαια υπηρεσιών.
- Αρχή 7: Οι υπηρεσίες πρέπει να είναι χαλαρά συνδεδεμένες (δηλαδή να μοντελοποιούνται όσο γίνεται ανεξάρτητα από το περιβάλλον ανάπτυξης).
- Αρχή 8: Συλλογές από συσχετιζόμενες υπηρεσίες πρέπει να χρησιμοποιούν τους ίδιους XML τύπους αρχείου για να διευκολύνουν την ανταλλαγή πληροφοριών ανάμεσα σε σχετικές υπηρεσίες. Επίσης η δομή και η σημασιολογία των εγγράφων πρέπει να έχει συμφωνηθεί και να είναι κατανοητή.
- Αρχή 9: Οι υπηρεσίες πρέπει να πραγματοποιούν διακριτές εργασίες και να παρέχουν απλές διεπαφές για πρόσβαση στη λειτουργικότητα τους προκειμένου να ενθαρρύνουν την επαναχρησιμοποίηση και τη χαλαρή σύνδεση της τεχνολογίας.
- Αρχή 10: Οι υπηρεσίες πρέπει να παρέχουν μεταδεδομένα που καθορίζουν τις δυνατότητες και τους περιορισμούς τους και αυτά τα μεταδεδομένα πρέπει να είναι διαθέσιμα σε ένα αποθετήριο (repository), το οποίο μπορεί να προσπελαστεί από τις υπηρεσίες. Αυτό επιτρέπει να ανακαλύπτονται οι ορισμοί των υπηρεσιών χωρίς

να χρειάζεται πρόσβαση στην ίδια την υπηρεσία.

2.13.1 Βασικά χαρακτηριστικά υπηρεσίας

Οι υπηρεσίες παρέχουν πολυάριθμα τεχνικά και επιχειρησιακά οφέλη.
Τα κύρια χαρακτηριστικά που πρέπει να διακρίνουν το σχεδιασμό, την υλοποίηση και τη διαχείριση των υπηρεσιών είναι τα ακόλουθα:

- Χαλαρά συνδεδεμένες υπηρεσίες (loosely coupled services)
- Καλά ορισμένα συμβόλαια υπηρεσίας (SLAs)

Κάθε υπηρεσία πρέπει να έχει μια καλά ορισμένη διεπαφή που ονομάζεται το συμβόλαιο υπηρεσίας (service contract) και ορίζει σαφώς τις δυνατότητες της υπηρεσίας και το πώς θα κληθεί η υπηρεσία με τρόπο που να ενισχύει τη διαλειτουργικότητα και να διαχωρίζει την εξωτερικά προσπελάσιμη διεπαφή από την τεχνική υλοποίηση της υπηρεσίας. Είναι σημαντικό το συμβόλαιο να καθορίζεται βάσει της γνώσης του επιχειρησιακού τομέα και να μη προκύπτει απλά από την υλοποίηση της υπηρεσίας. Η WSDL περιγραφή παρέχει τη βάση για τον καθορισμό συμβολαίων, ωστόσο το συμβόλαιο μπορεί να περιλαμβάνει επιπλέον μεταδεδομένα ασφάλειας και πολιτικής χρησιμοποιώντας την οικογένεια προδιαγραφών WS-Policy.

- Κατανοητές υπηρεσίες για τους αιτούντες (requesters)

Οι υπηρεσίες και τα συμβόλαιά τους πρέπει να ορίζονται με έναν αφαιρετικό τρόπο που έχει νόημα για τους αιτούντες τις υπηρεσίες. Μια αφηρημένη διεπαφή προωθεί την υποκατάσταση (substitutability), δηλαδή η διεπαφή αναπαριστά ένα επιχειρησιακό θέμα και είναι ανεξάρτητη της συγκεκριμένης υλοποίησης, γεγονός το οποίο επιτρέπει σε ένα πάροχο μιας υπηρεσίας να την αντικαταστήσει χωρίς να επηρεαστούν οι αιτούντες υπηρεσίες. Με αυτό τον τρόπο ενισχύεται η χαλαρή σύνδεση και η επαναχρησιμοποίηση.

- Βασισμένες σε (ανοιχτά) πρότυπα

Η χρήση των ανοιχτών προτύπων παρέχει πολλά πλεονεκτήματα όπως για παράδειγμα

- η ελαχιστοποίηση χρήσης τεχνολογιών και διεπαφών ενός αποκλειστικού προμηθευτή,
- περισσότερες ευκαιρίες για ένα πάροχο να υποστηρίζει πολλούς διαφορετικούς αιτούντες και
- ευκαιρίες ένας οργανισμός να επωφεληθεί από τις κοινότητες χρηστών και προγραμματιστών που αναπτύσσονται για τις υλοποιήσεις ανοιχτού κώδικα.

2.13.2 Χαρακτηριστικά που πρέπει να πληρούν οι ΥΙ

Μια υπηρεσία πρέπει επίσης να διαθέτει όσο περισσότερα από τα παρακάτω δευτερεύοντα χαρακτηριστικά είναι δυνατόν προκειμένου να παρέχει τα καλύτερα επιχειρησιακά και τεχνικά οφέλη.

- Προβλέψιμες συμφωνίες επιπέδου υπηρεσίας (SLAs)
- Δυναμικές υπηρεσίες, ανιχνεύσιμες και οδηγούμενες από τα μεταδεδομένα
- Ανεξαρτησία υλοποίησης από άλλες Υπηρεσίες Ιστού
- Πρόβλεψη (provisioning) για επιβολή μηχανισμού επαναφοράς των συναλλαγών
- Σχεδιασμός για πολλαπλά στυλ κλήσεων (interface bindings)
- Υπηρεσίες μη διατήρησης κατάστασης (stateless)
- Σχεδιασμός υπηρεσιών με πρόβλεψη μέγιστης απόδοσης

<u>Κατευθυντήριες γραμμές της SOA αρχιτεκτονικής ως προς τους αιτούντες τις υπηρεσίες</u>

- Οι αιτούντες τις υπηρεσίες πρέπει -όποτε είναι δυνατόν- να καλούν τους παρόχους

υπηρεσιών χρησιμοποιώντας ένα διακομιστή μεσολάβησης υπηρεσίας (service proxy) υψηλού επιπέδου. Ο διακομιστής μεσολάβησης υπηρεσίας πρέπει να παρέχει εύχρηστη διεπαφή (usable interface) ειδικά υλοποιημένη για το προγραμματιστικό μοντέλο του αιτούντα (για παράδειγμα, για πελάτες JAVA, ο διακομιστής υπηρεσίας πιθανόν να παρέχει μια JAX-RPC διεπαφή).

- Επίσης, πρέπει να ενσωματώνει τις λεπτομέρειες της SOAP σύνδεσης (SOAP binding) ώστε να μπορούν να αλλάξουν στο μέλλον αν χρειαστεί.
- Τέλος, πρέπει να είναι δυνατός ο έλεγχος της συμπεριφοράς του διακομιστή μεσολάβησης υπηρεσίας με χρήση δεδομένων ρύθμισης έτσι ώστε να μπορεί εύκολα να προσαρμοστεί στις αλλαγές στη μορφή δεδομένων και στο ενδιάμεσο πρωτόκολλο μεταφοράς χωρίς να επηρεάζεται ο αιτούμενος την υπηρεσία.

Κατευθυντήριες γραμμές της SOA αρχιτεκτονικής σχετικά με τα παλαιού τύπου συστήματα και υπηρεσίες (legacy systems and services)

Οι υπηρεσίες παλαιού τύπου πρέπει να αντιμετωπίζονται όπως οι άλλες υπηρεσίες, δηλαδή να καθορίζονται από ένα επίσημο συμβόλαιο υπηρεσίας (που περιγράφεται με χρήση WSDL) και να δημοσιεύονται στο μητρώο υπηρεσιών (UDDI).
Αν η υπηρεσία παλαιού τύπου είναι προσπελάσιμη μέσα από ένα ενδιάμεσο λογισμικό (mediation – ESB)[10], τότε πρέπει να υλοποιεί το υποκείμενο ενδιάμεσο λογισμικό (π.χ. JMS, IBM WebSphere MQ) .

[10] http://naveenbalani.com/index.php/2010/05/esb-frameworks/

3. ΤΕΧΝΟΛΟΓΙΑ XML

3.1 Εισαγωγή

Η εκτεταμένη γλώσσα σήμανσης (eXtensible Markup Language – XML) είναι η κοινή διάλεκτος στην προσανατολισμένη στις υπηρεσίες αρχιτεκτονική (SOA). Χρησιμοποιείται για το περιεχόμενο των μηνυμάτων, για τις ρυθμίσεις εφαρμογών, για την ανάπτυξη και την ανακάλυψη των υπηρεσιών, για τις πολιτικές/τακτικές κατά το χρόνο εκτέλεσης και όλο και συχνότερα για την αναπαράσταση εκτελέσιμων γλωσσών όπως η BPEL. Οι διεπαφές υπηρεσιών ιστού αναπαρίστανται με XML στη μορφή της WSDL και η XML χρησιμοποιείται ως ο κύριος μηχανισμός μεταφοράς (transportation mechanism) στη SOA αρχιτεκτονική. Η XML είναι άρρηκτα συνδεδεμένη με την ανάπτυξη υπηρεσιών ιστού και πολύ ισχυρή λόγω της εγγενούς ευελιξίας (flexibility) και εκφραστικότητας (expressiveness) που παρέχει.
Στο παρελθόν, οι προγραμματιστές και οι προμηθευτές εφαρμογών (vendors) κατασκεύαζαν εφαρμογές και συστήματα εγκατεστημένα σε μια επιχείρηση τα οποία επεξεργάζονταν δεδομένα που μπορούσαν να ανταλλάσουν με το δικό τους ιδιωτικό τρόπο. Αλλά καθώς η ανταλλαγή πληροφορίας μεταξύ εφαρμογών και συστημάτων στις επιχειρήσεις επικρατούσε, έγινε πολύ δύσκολο να ανταλλάσσονται δεδομένα διότι τα συστήματα δε σχεδιάστηκαν ώστε να δέχονται δεδομένα από εξωτερικά, άγνωστα συστήματα. Η XML λύνει το πρόβλημα αυτό παρέχοντας μια πρότυπη και κοινή δομή για την αποστολή δεδομένων μεταξύ ανόμοιων συστημάτων και επιπλέον εγγυάται ότι η δομή των δεδομένων που λαμβάνεται είναι έγκυρη. Τα XML Schemas έχουν μεγάλες δυνατότητες αλλά δεν είναι αντικειμενοστραφή και προορίζονται για να αναπαριστούν μοντέλα δεδομένων και όχι μοντέλα αντικειμένων. Τα σχήματα επιτρέπουν στους σχεδιαστές να δημιουργήσουν μοντέλα που μπορεί να μοιάζουν πολύ με τα αντικείμενα. Η JAXB, για παράδειγμα, επιτρέπει τη δημιουργία τύπων αρίθμησης της Java 5. Ουσιαστικά, το νόημα είναι ότι οι υπηρεσίες πρέπει να είναι αρκετά γενικές ώστε να μπορούν να επικοινωνούν ανεξαρτήτως γλώσσας υλοποίησης και ο σχεδιασμός μοντέλων δεδομένων (data models) με XML προσφέρει αυτή τη δυνατότητα.

3.2 Δημιουργία ενός Κανονικού Μοντέλου Δεδομένων (Canonical Data Model)

Η ανάλυση και η δημιουργία του σχήματος πρέπει να πραγματοποιείται πριν και ανεξάρτητα από την ανάλυση της υπηρεσίας. Οι υπηρεσίες θα ανταλλάσουν επιχειρησιακά έγγραφα και αυτά τα επιχειρησιακά έγγραφα θα αποτελούνται από διατμηματικές οντότητες (cross-domain entities). Για παράδειγμα, μια σειρά από υπηρεσίες σε διαφορετικούς τομείς των επιχειρήσεων ενδεχομένως θα χρειαστεί να χρησιμοποιήσει ορισμένους βασικούς τύπους, όπως ο Πελάτης ή το Προϊόν ή το Τιμολόγιο.
Σε πρακτικό επίπεδο το γεγονός αυτό θέτει άμεσα το πρόβλημα της επαναχρησιμοποίησης του σχήματος. Οι επιλογές για την αντιμετώπισή του είναι οι ακόλουθες:

- Μεγιστοποίηση της επαναχρησιμοποίησης των τύπων και εξάλειψη των ορισμών περιττών τύπων. Τα σχήματα Πελάτη, Προϊόντος και Τιμολογίου πρέπει να ορίζονται με ένα σχέδιο σχήματος (schema pattern) που να επιτρέπει την ανασύνθεση. Πρέπει να γράφονται σχήματα που αναπαριστούν βασικούς ή γενικούς τύπους που ίσως χρησιμοποιηθούν σε πολλά μέρη όπως π.χ. η διεύθυνση, ή ο αριθμός πιστωτικής κάρτας. Στη συνέχεια γράφονται σχήματα για κάθε συγκεκριμένη υπηρεσία που επαναχρησιμοποιούν αυτούς τους βασικούς τύπους. Το πρόβλημα όμως θα δημιουργηθεί όταν οι απαιτήσεις για μια υπηρεσία αλλάξουν ώστε να αλλάξει το σχήμα οπότε και θα πρέπει να ξανά αναπτυχθούν και να επανελεγχθούν πολλές υπηρεσίες.

- Μεγιστοποίηση της ευελιξίας της υπηρεσίας και ελαχιστοποίηση της εξάρτησης. Σ' αυτή την επιλογή οι τύποι Πελάτης, Προϊόν και Τιμολόγιο καθορίζονται χωριστά για κάθε υπηρεσία χωρίς συμβιβασμούς. Αυτό μπορεί να σημαίνει πολλή πλεονάζουσα πληροφορία (επίσης γνωστό με την ονομασία «απομαλοποίηση σχήματος» (schema denormalization)) αλλά επιτρέπει στις υπηρεσίες να εξελίσσονται ανεξάρτητα. Επίσης επιτρέπει μια περισσότερο ενθυλακωμένη υπηρεσία (encapsulated). Και σε αυτή την περίπτωση αν οι απαιτήσεις αλλάξουν για ένα βασικό τύπο που ορίζεται σε πολλά διαφορετικά σχήματα οι προγραμματιστές θα έχουν αρκετή χειρωνακτική και επιρρεπή σε λάθη δουλειά.

Συνεπώς, καμία από αυτές τις επιλογές δεν είναι ιδιαίτερα ελκυστική. Υπάρχει όμως τρόπος να συνδυαστούν τα καλά στοιχεία των 2 αυτών μεθόδων χρησιμοποιώντας το επίπεδο της αφαίρεσης. Αυτό εφαρμόζεται στο *κανονικό μοντέλο δεδομένων ή κανονικό πρότυπο σχήμα* (*canonical data model* or *canonical schema pattern*).

3.2.1 Ορισμός του κανονικού μοντέλου δεδομένων

Το κανονικό μοντέλο δεδομένων αναφέρεται στην πρακτική του καθορισμού σχημάτων που αναπαριστούν τύπους για ολόκληρη την εφαρμογή και στην επαναχρησιμοποίηση των τύπων αυτών με αναφορά στα σχήματα που σχεδιάζονται τοπικά για την υπηρεσία. Με τον τρόπο που ένα ESB (Enterprise Service Bus) δρα ως μεσολαβητής, επιτρέποντας σε οποιοδήποτε αριθμό υπηρεσιών να συνομιλούν μεταξύ τους χωρίς να χρειάζεται η δημιουργία συγκεκριμένης σχέσης σημείου προς σημείο ανάμεσα στην κάθε μια, έτσι ένα κανονικό μοντέλο δεδομένων επιτρέπει τον καθορισμό τελικών σχημάτων για τους τύπους στην επιχείρηση: κάθε υπηρεσία που χρειάζεται να επικοινωνήσει μπορεί να χρησιμοποιήσει το ESB ή κάποιον άλλο μηχανισμό που θα μεταφράσει την έκδοση των τύπων της υπηρεσίας στην κανονική μορφή.
Το "κανονικό σχήμα" (canonical schema) λειτουργεί ως πρότυπο. Μπορεί να αναπαραστήσει τα πάντα για μια οντότητα. Κάθε υπηρεσία που αναφέρεται σε μια οντότητα ίσως χρειαστεί να την προσαρμόσει με κάποιο τρόπο για να εκτελεί τη συγκεκριμένη εργασία που πρέπει. Αλλά αυτές οι διαφορές είναι συγκεκριμένες στο σχήμα για αυτήν την υπηρεσία και κάθε συγκεκριμένο σχήμα μπορεί να μετατραπεί στο κανονικό σχήμα. Το σχήμα της υπηρεσίας και το κανονικό σχήμα είναι διαφορετικά έγγραφα και έτσι μεγιστοποιείται και η ελευθερία και η επαναχρησιμοποίηση.
Το κανονικό σχήμα αντιπροσωπεύει τον τρόπο που η εταιρεία θέλει να εκπροσωπεί π.χ. τη διεύθυνση, το όνομα και άλλα στοιχεία. Το κανονικό σχήμα είναι πιο ευέλικτο και γενικό από το τοπικό σχήμα οντότητας. Είναι πιο ελαστικό, επειδή ο κάθε περιορισμός διεύθυνσης για κάθε εφαρμογή που χρειάζεται να ενσωματωθεί δε θα ορίζεται σύμφωνα με τους ίδιους περιορισμούς. Μπορούμε να φτάσουμε στην κανονική αναπαράσταση από κάθε τοπικό σχήμα οντότητας. Και από το κανονικό σχήμα, μπορούμε να μεταβούμε σε οποιαδήποτε άλλη αναπαράσταση. Με το ESB, μπορεί να χρησιμοποιηθεί το XSLT μέσα σε μια ενορχήστρωση ή ενδιάμεση υπηρεσία για τη μετάφραση από τον ένα τύπο στον άλλο ανάλογα με τις ανάγκες, να συμπληρωθούν τα μη καθορισμένα πεδία με προεπιλογές, και ούτω καθεξής.

3.2.2 Ενημέρωση του κανονικού μοντέλου

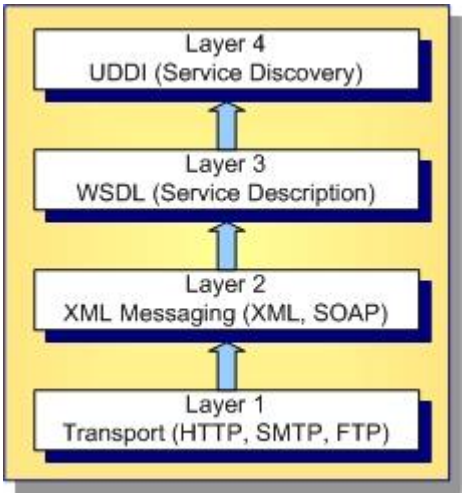
Οι απόψεις για το κανονικό μοντέλο διίστανται. Κάποιοι ερευνητές της SOA το ονομάζουν πρότυπο προς αποφυγή για το οποίο είναι καλό να γνωρίζουν. Αλλά πολλοί άλλοι αναφέρονται σε αυτό όχι απλά σαν θετικό πρότυπο αλλά ως ένα από τα πιο σημαντικά και θεμελιώδη στοιχεία της δημιουργίας μιας καλής αρχιτεκτονικής SOA. Το κανονικό μοντέλο δε θα πρέπει να θεωρείται κακό πρότυπο αν γίνεται αρκετή ανάλυση και διευκρίνιση των εννοιών. Κατά τον ορισμό των υπηρεσιών πρέπει να ορίζεται ένα σχήμα τοπικά για την υπηρεσία. Οι τύποι του κανονικού μοντέλου πρέπει να χρησιμοποιούνται όσο το δυνατόν περισσότερο με χρήση import.
Όλες οι αλλαγές στο κανονικό μοντέλο είναι σωστό να προκύπτουν μέσω της SOA διακυβέρνησης (“governance”) και αυτές θα πρέπει να είναι σπάνιες. Οι υπηρεσίες αναπαριστούν επιχειρησιακές οντότητες, κάτι που δεν αλλάζει κάθε μέρα. Όταν τελικά χρειαστεί αλλαγή δημιουργείται ένα νέο σχήμα και αλλάζει μόνο η υπηρεσία που θα χρησιμοποιεί τη νέα έκδοση του κανονικού μοντέλου. Φυσικά η ανανέωση και των υπόλοιπων υπηρεσιών θα πρέπει να γίνει σύντομα γιατί αν υπάρχουν παραπάνω από 2 εκδόσεις του κανονικού σχήματος τότε παύει να είναι κανονικό.

3.3 Πως η XML βοηθάει να απλοποιηθεί η ανάπτυξη και ολοκλήρωση συστημάτων

Με τη χρήση της XML στις υπηρεσίες ιστού παρέχεται ένας σαφής διαχωρισμός ανάμεσα στον ορισμό μιας υπηρεσίας και στην εκτέλεσή της. Αυτός ο διαχωρισμός είναι σκόπιμος έτσι ώστε οι υπηρεσίες ιστού να μπορούν να δουλέψουν με κάθε λειτουργικό σύστημα. Το XML Schema μιας υπηρεσίας προσφέρει μια XML αναπαράσταση των τύπων και των δομών δεδομένων επιτρέποντας στον προγραμματιστή να μην ασχολείται με λεπτομέρειες μιας συγκεκριμένης υλοποίησης. Το γεγονός αυτό εκφράζει μια αλλαγή στη φύση του προβλήματος ολοκλήρωσης που προέκυπτε όταν προκειμένου να γίνει επικοινωνία με την υπηρεσία έπρεπε να είναι γνωστή η υλοποίηση της. Πλέον δεν έχει σημασία αν το περιβάλλον εκτέλεσης της υπηρεσίας είναι αντικείμενο, ουρά μηνυμάτων ή αποθηκευμένη διαδικασία αφού η Υπηρεσία Ιστού περιλαμβάνει ένα επίπεδο που αντιστοιχίζει την YI σε οποιοδήποτε περιβάλλον εκτέλεσης υλοποιεί την υπηρεσία.
Η σημασία της XML για τις YI, τονίζεται επιπροσθέτως από το γεγονός ότι αποτελεί τη βάση του πρωτοκόλλου SOAP, της γλώσσας WSDL και του μητρώου UDDI που απαρτίζουν τη στοίβα πρωτοκόλλων υπηρεσιών Ιστού, η οποία εξετάζεται στο επόμενο κεφάλαιο.

4. ΣΤΟΙΒΑ ΠΡΩΤΟΚΟΛΛΩΝ ΥΠΗΡΕΣΙΩΝ ΙΣΤΟΥ

Εξετάζοντας τις υπηρεσίες ιστού ως στοίβα πρωτοκόλλων, φυσιολογικά, υπονοούνται δύο παραδοχές.



Σχήμα 1. Στοίβα πρωτοκόλλων Υπηρεσιών Ιστού

Το Σχήμα 1 παρουσιάζει την πρώτη παραδοχή που αφορά την περιγραφή της στοίβας πρωτοκόλλων Υπηρεσιών Ιστού. Σύμφωνα με αυτήν:

- στην κορυφή της στοίβας τοποθετείται η Καθολική, Περιγραφή, Ανακάλυψη και Ολοκλήρωση (UDDI) για την αποθήκευση των περιγραφών των μηνυμάτων,
- στο αμέσως κατώτερο επίπεδο ακολουθεί η Γλώσσα Περιγραφής Υπηρεσιών Ιστού (WSDL) για την περιγραφή των μηνυμάτων και
- στο υποκείμενο επίπεδο βρίσκεται το Απλό Πρωτόκολλο Πρόσβασης Αντικειμένων (SOAP) για τη μεταφορά των μηνυμάτων.

Όλα τα υπόλοιπα πρωτόκολλα σχετίζονται σε κάποιο βαθμό με αυτά τα τρία αλλά δεν τίθενται σε ένα σαφές πλαίσιο.

Η δεύτερη παραδοχή αφορά το SOAP, που διαισθητικά τοποθετείται πάνω από το υποκείμενο επίπεδο μεταφοράς, το οποίο συνήθως είναι το Πρωτόκολλο Μεταφοράς Υπερκειμένου (Hypertext transfer Protocol - HTTP).

Το πλαίσιο των υπηρεσιών ιστού αποτελείται από ένα κύκλο *δημοσίευση-εύρεση-δέσμευση* (publish – bind – find), σύμφωνα με τον οποίο οι πάροχοι υπηρεσιών καθιστούν τα δεδομένα, το περιεχόμενο ή τις υπηρεσίες διαθέσιμες σε εγγεγραμμένους αιτούντες υπηρεσιών, οι οποίοι καταναλώνουν πόρους από τον εντοπισμό και τη δέσμευση των υπηρεσιών. Οι αιτούσες εφαρμογές μπορούν να συντονίζονται με τις υπηρεσίες Web, χρησιμοποιώντας τη WSDL (γλώσσα περιγραφής υπηρεσιών Ιστού), η οποία παρέχει χαμηλού επιπέδου τεχνικές πληροφορίες σχετικά με την επιθυμητή υπηρεσία, εγγυάται στις εφαρμογές πρόσβαση σε πληροφορίες XML Schema για την κωδικοποίηση δεδομένων και εξασφαλίζει ότι οι σωστές λειτουργίες καλούνται με τα σωστά πρωτόκολλα. Οι μηχανισμοί δημοσίευσης – εύρεσης – δέσμευσης αντιστοιχούν στα τρία διαφορετικά πρωτόκολλα που αποτελούν τη στοίβα πρωτοκόλλων υπηρεσιών Ιστού: UDDI, SOAP και WSDL.

4.1 Η κλασσική στοίβα πρωτοκόλλων υπηρεσιών ιστού

- Επίπεδο μεταφοράς (Transport Layer)

Ξεκινώντας από το χαμηλότερο επίπεδο βρίσκουμε το επίπεδο μεταφοράς που είναι το θεμέλιο της στοίβας των υπηρεσιών ιστού. Οι υπηρεσίες πρέπει να καλούνται από έναν πελάτη, ώστε να μπορούν να είναι προσπελάσιμες σε ένα δίκτυο. Υπηρεσίες Ιστού που είναι δημόσια διαθέσιμες τρέχουν μέσω του Διαδικτύου. Μόνο οι εξουσιοδοτημένοι χρήστες μέσα σε μια εσωτερική οργάνωση μπορούν να δουν υπηρεσίες ιστού διαθέσιμες σε εσωτερικά δίκτυα intranet, ενώ οι μη εξουσιοδοτημένοι χρήστες στον έξω κόσμο δεν μπορούν. Τα πρωτόκολλα διαδικτύου που υποστηρίζονται είναι το Πρωτόκολλο Μεταφοράς Υπερκειμένου (HTTP) και σε μικρότερο βαθμό το Απλό Πρωτόκολλο Μεταφοράς Αλληλογραφίας (Simple Mail Transfer Protocol - SMTP) και το Πρωτόκολλο Μεταφοράς Αρχείων (File Transfer Protocol - FTP).

- Επίπεδο ανταλλαγής XML μηνυμάτων (Messaging Exchange Layer)
  Σε αυτό το επίπεδο το Απλό Πρωτόκολλο Πρόσβασης Αντικειμένων (Simple Object Access Protocol - SOAP) είναι το πρωτόκολλο ανταλλαγής μηνυμάτων για την SOA. Βασίζεται στο χαμηλότερο επίπεδο της μεταφοράς κάτι που σημαίνει ότι το SOAP χρησιμοποιείται σε συνδυασμό με άλλα πρωτόκολλα μεταφοράς. Το SOAP περιλαμβάνει τρία μέρη:
  - το εξωτερικό “περιτύλιγμα” (envelope) που περιγράφει το περιεχόμενο του μηνύματος,
  - ένα σύνολο κανόνων κωδικοποίησης και
  - ένα μηχανισμό παροχής απομακρυσμένων κλήσεων διαδικασιών (remote procedure calls) και αποκρίσεων.
- Επίπεδο περιγραφής υπηρεσίας (Description Layer)

Η περιγραφή της υπηρεσίας παρέχει τα μέσα για την κλήση υπηρεσιών ιστού. Η Γλώσσα Περιγραφής Υπηρεσιών Ιστού (Web Service Description Language – WSDL) είναι το βασικό πρότυπο για τον καθορισμό, σε μορφή XML, των υλοποιήσεων και των διεπαφών των υπηρεσιών.

- Επίπεδο δημοσίευσης υπηρεσίας (Publish Layer)

Σε αυτό το επίπεδο ο πάροχος υπηρεσίας μπορεί να αποστείλει ένα WSDL έγγραφο κατευθείαν σε ένα πελάτη της υπηρεσίας (π.χ. με ένα μήνυμα ηλεκτρονικού ταχυδρομείου) ή να αποστείλει τη διεύθυνσή του http://MyWebservice/MyService?WSDL. Αυτή η πράξη είναι γνωστή ως άμεση δημοσίευση υπηρεσίας. Ο πάροχος της υπηρεσίας μπορεί επίσης να επιλέξει να δημοσιεύσει το έγγραφο WSDL σε ένα μητρώο τοπικά στον εξυπηρετητή ή σε ένα δημόσιο ή ιδιωτικό μητρώο UDDI.

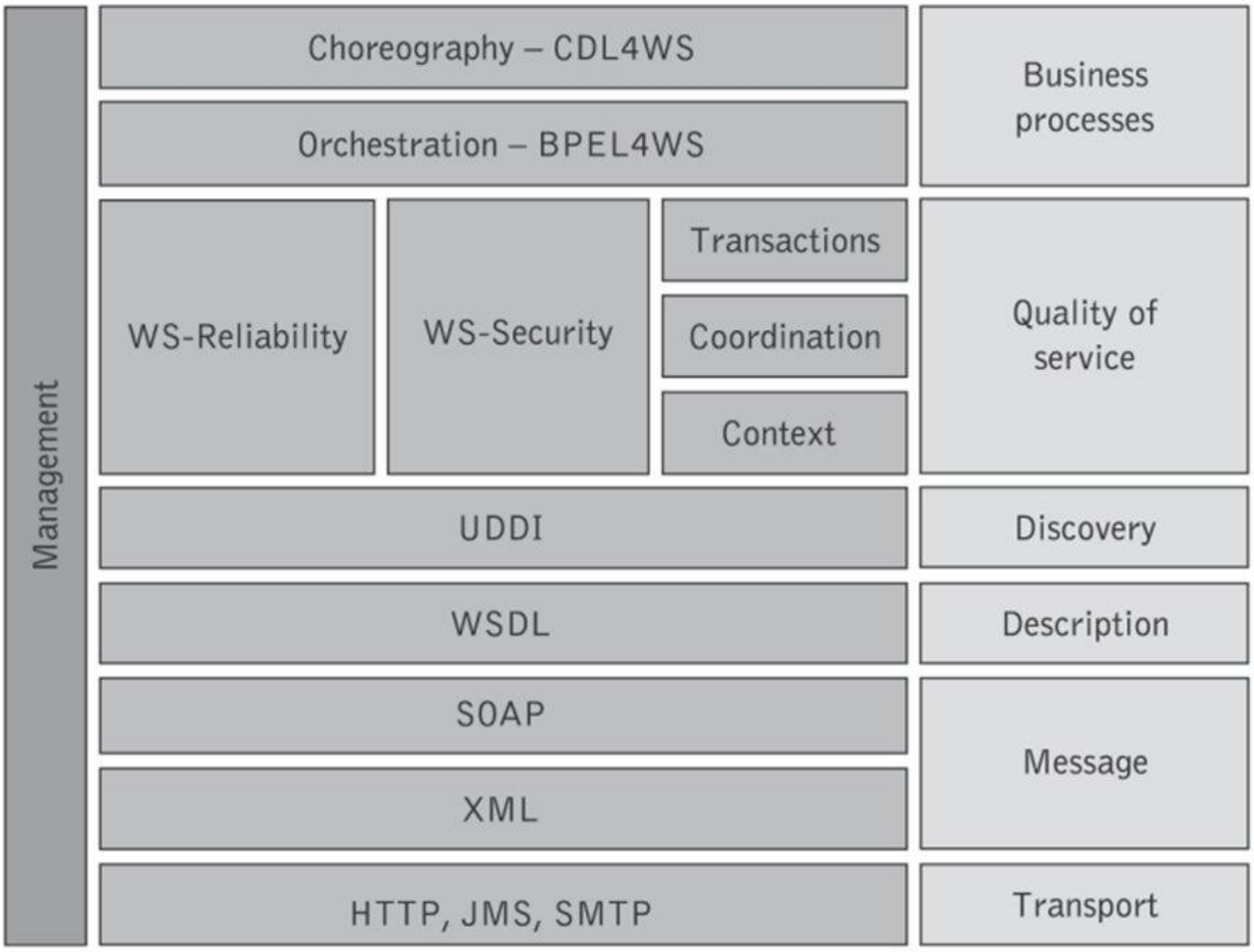
- Επίπεδο ανακάλυψης υπηρεσίας (Discovery Layer)

Η ανακάλυψη της υπηρεσίας επηρεάζεται από τη δημοσίευση της υπηρεσίας. Ο πελάτης της υπηρεσίας μπορεί να κάνει διαθέσιμη την περιγραφή της υπηρεσίας σε μια εφαρμογή κατά το χρόνο εκτέλεσης. Για παράδειγμα ο πελάτης της υπηρεσίας μπορεί να ανακτήσει ένα έγγραφο WSDL από ένα τοπικό αρχείο που αποκτήθηκε με άμεση δημοσίευση. Αυτή η ενέργεια είναι γνωστή ως στατική ανακάλυψη.

- Επίπεδο επιχειρησιακών διεργασιών

Στην κορυφή της στοίβας τοποθετείται η πρότυπη εκτελέσιμη γλώσσα BPEL, ο ρόλος της οποίας είναι ο προσδιορισμός εκτελέσιμων και αφηρημένων (abstract) επιχειρησιακών διεργασιών με υπηρεσίες ιστού. Η γλώσσα αυτή, πρόγονοι της οποίας είναι η WSFL και XLANG, παρέχει υποστήριξη επιχειρησιακών συναλλαγών (business transactions). Είναι βασικά μια γλώσσα ενορχήστρωσης που ελέγχει την ακολουθία ανταλλαγής μηνυμάτων της επιχειρησιακής διεργασίας με άλλα συστήματα χρησιμοποιώντας τις υπηρεσίες ιστού (και ειδικά τις WSDL περιγραφές τους) ως εξωτερικό μηχανισμό επικοινωνίας. Η BPEL και η σημασία της στην ενορχήστρωση υπηρεσιών και την αυτοματοποιημένη εκτέλεση επιχειρησιακών διεργασιών εξετάζεται

αναλυτικότερα σε επόμενο κεφάλαιο.



Σχήμα 2. Η στοίβα πρωτοκόλλων Υπηρεσιών Ιστού

4.2 Μοντελοποίηση της στοίβας πρωτοκόλλων Υπηρεσιών Ιστού σύμφωνα με το μοντέλο αναφοράς OSI

Το σύνολο πρωτοκόλλων που αποτελούν τη στοίβα Υπηρεσιών Ιστού μπορεί να μοντελοποιηθεί παρόμοια με το μοντέλο αναφοράς Ανοικτής Διασύνδεσης Συστημάτων (Open Systems Interconnection - OSI reference model). Με αυτό τον τρόπο παρέχεται ένα ορθό και τεκμηριωμένο πλαίσιο που θα περιλαμβάνει τα υπάρχοντα και τυχόν νέα πρωτοκόλλα υπηρεσιών ιστού.

	Internet	Web Services
Presentation Layer (Layer 6)	various (application dependent)	XML Schema
Session Layer (Layer 5)		various (e.g., BPEL4WS, WS-Choreography)
Transport Layer (Layer 4)	TCP or UDP	WSDL MEP
Network Layer (Layer 3)	IP	WS-Addressing, WS-Routing, ...
Data Link Layer (Layer 2)	Ethernet	SOAP, URI, XML Infoset
Physical Layer (Layer 1)	various transport media	HTTP, TCP, BEEP, ...

Σχήμα 3. Τα πρωτόκολλα ΥΙ με τα ισοδύναμά τους πρωτόκολλα της στοίβας OSI

- Επίπεδο 1: Επίπεδο υποκείμενου πρωτοκόλλου (Underlying Protocol Layer)

Στη στοίβα πρωτοκόλλων OSI το πρώτο επίπεδο είναι το φυσικό, το οποίο καθορίζει τις ηλεκτρικές και φυσικές προδιαγραφές της επικοινωνίας. Η τεχνολογία του επιπέδου 1 για τις υπηρεσίες Ιστού χρησιμοποιείται για να κάνει εφικτή τη σύνδεση και την ανταλλαγή δεδομένων μεταξύ των συμμετεχόντων στην επικοινωνία. Η διασύνδεση αυτή για τις υπηρεσίες Ιστού μπορεί να είναι σχεδόν οτιδήποτε, καθώς οι υπηρεσίες ιστού μπορούν να βασίζονται σε διαφορετικούς μηχανισμούς μεταφοράς. Έτσι περιλαμβάνονται πρωτόκολλα που ανήκουν στο πέμπτο ή έκτο επίπεδο του Διεθνή Οργανισμού Τυποποίησης – (International Organization for Standardization – ISO) όπως το Πρωτόκολλο Μεταφοράς Υπερκειμένου (HyperText Transfer Protocol - HTTP). Ο κυριότερος λόγος για αυτό είναι ότι το HTTP είναι το ευρύτερα χρησιμοποιούμενο πρωτόκολλο του Διαδικτύου που υποστηρίζεται σε όλες τις πλατφόρμες και λόγω της αναπαράστασής του επιτρέπεται η διέλευσή του από τα τείχη προστασίας (firewalls) και άλλους μηχανισμούς φιλτραρίσματος. Ένα άλλο είδος διασύνδεσης που είναι εφικτό και χρησιμοποιείται στη στοίβα πρωτοκόλλων υπηρεσιών ιστού είναι το Απλό Πρωτόκολλο Μεταφοράς Αλληλογραφίας (Simple Mail Transfer Protocol - SMTP). Επίσης, τα πρωτόκολλα του επιπέδου μεταφοράς του Διεθνή Οργανισμού Τυποποίησης όπως το ζεύγος TCP/IP και το πρωτόκολλο δεδομενογράμματος χρήστη (User Datagram Protocol – UDP), όταν δεν απαιτείται αξιόπιστη μεταφορά, μπορούν να αναπτυχθούν για τη μεταφορά περιεχομένου σχετικού με υπηρεσίες ιστού.

- Επίπεδο 2: Επίπεδο URI, XML και SOAP (URI, XML and SOAP Layer)

Το επίπεδο 2 είναι υπεύθυνο για την αξιόπιστη μεταφορά δεδομένων ανάμεσα σε δύο ομότιμα μέρη που επικοινωνούν και συνδέονται με την υποδομή του επιπέδου 1. Σύμφωνα με το μοντέλο αναφοράς Ανοικτής Διασύνδεσης Συστημάτων το επίπεδο αυτό χωρίζεται σε δύο υπό-επίπεδα. Στη στοίβα πρωτοκόλλων υπηρεσιών ιστού το χαμηλότερο υπο-επίπεδο είναι το Ενιαίο Αναγνωριστικό Πόρου – Uniform Resource Identifier (URI). Μια από τις βασικές αρχές του Ιστού είναι ότι οι πόροι παίρνουν κάποια διεύθυνση μέσω των URIs και εφόσον μια υπηρεσία ιστού είναι απλά ένας πόρος είναι προσπελάσιμος μέσω ενός αναγνωριστικού ενιαίου πόρου. Το ενιαίο αναγνωριστικό πόρου περιέχει ένα μέρος που ονομάζεται σχήμα (scheme) που καθορίζει τη μέθοδο προσπέλασης με την οποία μπορεί να προσπελαστεί κάποιος συγκεκριμένος πόρος. Το άλλο υπο-επίπεδο χρησιμοποιεί το απλό πρωτόκολλο πρόσβασης αντικειμένου (SOAP) για την ομαδοποίηση των μηνυμάτων που ανταλλάσσονται μεταξύ των συμμετεχόντων. Ενώ η προδιαγραφή του SOAP 1.1 βασιζόταν στην προδιαγραφή XML 1.0 για το μοντέλο δεδομένων (ο παραλήπτης υποτίθεται ότι λάμβανε το ίδιο ακριβώς έγγραφο XML 1.0 που έστελνε ο αποστολέας), η SOAP 1.2 άλλαξε στο Σύνολο Πληροφοριών (Information Set - Infoset), το οποίο είναι μια γενίκευση της XML. Με το Σύνολο Πληροφοριών να αποτελεί τη νέα βάση του SOAP, η κωδικοποίηση και αποκωδικοποίηση μηνυμάτων μπορεί να γίνει με πιο ευέλικτο τρόπο επειδή το σύνολο πληροφοριών (Infoset) αγνοεί κάποιες συντακτικές ιδιαιτερότητες της XML. Ειδικότερα, με το μηχανισμό βελτιστοποίησης μετάδοσης μηνυμάτων (Message Transmission Optimization Mechanism - MTOM) υπάρχουν κάποια επιπλέον πρότυπα που μπορούν να χρησιμοποιηθούν για να επιτευχθεί διαλειτουργικότητα στο επίπεδο κωδικοποίησης των ανταλλασσόμενων δεδομένων.

- Επίπεδο 3: Δρομολόγηση και Διευθυνσιοδότηση (Addressing and Routing Layer)

Με βάση τη φυσική σύνδεση που πραγματοποιείται από τα πρωτόκολλα του επιπέδου 2, το ισοδύναμο του επιπέδου δικτύου της στοίβας OSI για τις υπηρεσίες ιστού επίσης παρέχει λογικές συνδέσεις. Δύο από τις κύριες λειτουργίες που παρέχονται από το επίπεδο δικτύου είναι η δρομολόγηση (routing) και η απόδοση διεύθυνσης (addressing). Στη στοίβα πρωτοκόλλου του διαδικτύου αυτές υλοποιούνται βάσει του Πρωτοκόλλου Διαδικτύου (Internet Protocol – IP). Η απόδοση διεύθυνσης με τρόπο παρόμοιο της IP παρέχεται από το πρωτόκολλο Διευθυνσιοδότησης Υπηρεσιών Ιστού

(WS-Addressing) το οποίο λειτουργεί πάνω από το SOAP. Το εν λόγω πρωτόκολλο είναι ουσιαστικά ένα σύνολο στοιχείων XML με μια καθορισμένη σημασιολογία. Πιο συγκεκριμένα καθορίζει στοιχεία όπου αποθηκεύονται τα ενιαία αναγνωριστικά πόρων (URIs) του αποστολέα και του παραλήπτη του μηνύματος. Επίσης, ένα στοιχείο καθορίζει την υπηρεσία στην οποία θα δρομολογηθεί η κλήση.
Η οικογένεια πρωτοκόλλων υπηρεσιών ιστού επανασχεδιάζει τα δεδομένα που αποθηκεύονται σε κομμάτια υλικού της υποδομής δικτύου. Ένα χαρακτηριστικό παράδειγμα είναι η Δρομολόγηση υπηρεσιών ιστού (WS-Routing). Επιτρέπει τον ρητό καθορισμό του μονοπατιού που πρέπει να ακολουθήσει το μήνυμα από τον αποστολέα-πελάτη μέχρι τον τελικό παραλήπτη. Ουσιαστικά, το WS-Routing αποτελεί ένα πρωτόκολλο χωρίς καταστάσεις για την περιγραφή αυθαίρετων διαδρομών από τον πελάτη μέχρι τον τελικό προορισμό μέσω ενός πιθανόν κενού συνόλου μεσαζόντων. Τέλος, το επίπεδο δικτύου πρέπει να παρέχει μια συγκεκριμένη ποιότητα υπηρεσίας η οποία απαιτείται για την εκτέλεση συγκεκριμένων εργασιών στο δίκτυο. Το πρωτόκολλο WS-Policy αντιμετωπίζει αυτή την απαίτηση θέτοντας το πλαίσιο για να εκφράζονται οι κατάλληλες πολιτικές.

- Επίπεδο 4: Επίπεδο Ασφάλειας και Μορφής Μηνύματος (Security and Message Pattern Layer)

Μέσα στη στοίβα πρωτοκόλλου των υπηρεσιών ιστού, το επίπεδο μεταφοράς αντιπροσωπεύει το πιο σημαντικό τμήμα ολόκληρης της στοίβας πρωτοκόλλων. Η πιο αξιοσημείωτη διάκριση μεταξύ των πρωτοκόλλων που έχουν αναπτυχθεί στο πλαίσιο του κλασικού περιβάλλοντος δικτύωσης και της στοίβας πρωτοκόλλων υπηρεσιών ιστού είναι ο τρόπος με τον οποίο υποστηρίζονται μοτίβα μηνυμάτων. Συνήθως, τα πρωτόκολλα της ISO / OSI στοίβας, αλλά και τα πρωτόκολλα του Διαδικτύου ενώνουν τα πρωτόκολλα μεταφοράς όπως το TCP, UDP, RTP και τα μοτίβα των μηνυμάτων που αποστέλλονται και λαμβάνονται. Για παράδειγμα, το Πρωτόκολλο Ελέγχου Μεταφοράς (Transport Control Protocol - TCP) συνεπάγεται ένα αυστηρό μοτίβο αίτησης – απάντησης στο οποίο κάθε συνομιλία αποτελείται από ακριβώς δύο μηνύματα στην σειρά και κάθε εισερχόμενο μήνυμα ενός κόμβου πρέπει να ακολουθείται από ένα εξερχόμενο μήνυμα. Η αλλαγή του φυσικού σχήματος αίτησης – απάντησης του πρωτοκόλλου ελέγχου μεταφοράς (TCP) σε ένα σχήμα κοινοποίησης μιας κατεύθυνσης επιτυγχάνεται με τη χρήση του Πρωτοκόλλου Δεδομενογράμματος Χρήστη (User Datagram Protocol – UDP).
Στη στοίβα πρωτοκόλλων των υπηρεσιών ιστού το πραγματικό στυλ μιας αλληλεπίδρασης βασισμένης σε ανταλλαγή μηνυμάτων δεν είναι προκαθορισμένο από το πρωτόκολλο επικοινωνίας, π.χ. το SOAP. Αντίθετα η γλώσσα περιγραφής υπηρεσιών ιστού WSDL διαθέτει ένα σύνολο σχημάτων μηνυμάτων που εφαρμόζονται σε κάθε πρωτόκολλο επικοινωνίας και τα οποία καθορίζουν την ακολουθία και την πληθικότητα των μηνυμάτων. Για να παρακάμψει τα εμπόδια που προκύπτουν από την εξάρτηση του μηχανισμού μεταφοράς από το στυλ ή άλλα χαρακτηριστικά επικοινωνίας όπως η ασφάλεια, η στοίβα υπηρεσιών ιστού διαχωρίζει σαφώς τα θέματα ασφάλειας από το υποκείμενο πρωτόκολλο . Ως εκ τούτου SOAP-κωδικοποιημένα μηνύματα μπορούν να εξασφαλιστούν απλά και μόνο με το XML επίπεδο, με την υπογραφή και/ή κρυπτογράφηση τμημάτων ενός μηνύματος ή ολόκληρου του μηνύματος (WS-Security).

- Επίπεδο 5: Επίπεδο συντονισμού (Coordination Layer)

Εξ’ αιτίας των διαφορετικών απαιτήσεων που διέπουν την διαχείριση συνόδου δεν υπάρχει προς το παρόν κάποιο μοναδικό αποδεκτό πρότυπο για την παροχή λογικής συσχέτισης των μηνυμάτων που στέλνονται ξεχωριστά. Οι εφαρμογές που λειτουργούν πάνω από το πρωτόκολλο Διαδικτύου τυπικά παρέχουν συντονισμό και διαχείριση συνόδου με ανεξάρτητο και συνεπώς μη διαλειτουργικό τρόπο.
Για τις υπηρεσίες ιστού έως τώρα δεν έχει γίνει κοινά αποδεκτή μια μοναδική

προσέγγιση. Στα επικρατέστερα –ωστόσο- πρότυπα συγκαταλέγονται η Γλώσσα Εκτέλεσης Επιχειρησιακών Διεργασιών για Υπηρεσίες Ιστού (Business Process Execution Language for Web Services – BPEL4WS) και επίσης η Χορογραφία Υπηρεσιών Ιστού (WS-Choreography). Με το BPEL4WS θα ασχοληθεί η παρούσα πτυχιακή, με το οποίο θα υλοποιηθεί και μελετηθεί ένα σενάριο ενορχήστρωσης.

- Επίπεδο 6: Επίπεδο λεξιλογίου (Vocabulary Layer)

Το επίπεδο αυτό αντιστοιχίζει το μοντέλο δεδομένων της εφαρμογής σε μια φόρμα που μπορεί να μεταδοθεί ανάμεσα σε πελάτες και παρόχους υπηρεσίας. Αυτό επιτυγχάνεται με τη χρήση XML. Ωστόσο, οι εφαρμογές πρέπει να συμφωνούν σε ένα συγκεκριμένο σχήμα XML εγγράφων που θα ανταλλάσσονται και αυτό επιτυγχάνεται με το XML Schema. Το XML Schema καθορίζει με έναν αφαιρετικό τρόπο την σύνταξη της XML, ενώ η XML αποτελεί τον κανόνα κωδικοποίησης που χρησιμοποιείται για την κωδικοποίηση στιγμιότυπων δεδομένων. Ο ορισμός του XML Schema είναι μέρος του ορισμού της WSDL, που περιέχει ένα τμήμα που προσδιορίζει τους τύπους που χρησιμοποιούνται για την Υπηρεσία Ιστού. Εννοιολογικά, η WSDL δεν περιορίζεται στο XML και μπορεί να χρησιμοποιηθεί για την υποστήριξη άλλων μετά-γλωσσών επίσης. Ωστόσο, κάθε υλοποίηση WSDL απαιτείται να υποστηρίζει XML.

4.3 Απλό πρωτόκολλο πρόσβασης αντικειμένου (Simple Object Access Protocol – SOAP)

Η ονομασία SOAP προέκυψε από τα αρχικά του Simple Object Access Protocol (Απλό Πρωτόκολλο Πρόσβασης Αντικειμένου). Η διεθνής κοινοπραξία W3C όμως αποφάσισε να μη διατηρήσει αυτόν τον ορισμό, κυρίως επειδή στην πραγματικότητα η προδιαγραφή δεν υποχρεώνει τη χρήση αντικειμένων. Ωστόσο ο όρος SOAP είχε ήδη εδραιωθεί στον προγραμματιστικό τομέα και έτσι η W3C δεν προχώρησε στον ορισμό νέου ονόματος, αλλά το SOAP δεν αποτελεί πλέον ακρωνύμιο.
Το πρωτόκολλο αυτό επιτρέπει σε αντικείμενα Java και αντικείμενα COM (Component Object Model) να επικοινωνούν σε ένα κατανεμημένο, αποκεντρωμένο, βασισμένο στον Ιστό περιβάλλον. Γενικά, το πρωτόκολλο επιτρέπει σε αντικείμενα ή κώδικα κάθε είδους, σε κάθε πλατφόρμα και γλώσσα να επικοινωνούν μεταξύ τους. Προς το παρόν, το SOAP έχει υλοποιηθεί σε περισσότερες από 60 γλώσσες και σε περισσότερες από 20 πλατφόρμες. Πλέον, αντικείμενα οπουδήποτε, τοπικά και απομακρυσμένα, μεγάλα και μικρά, έχουν τη δυνατότητα να αλληλεπιδρούν. Το SOAP είναι τεχνολογία που προέκυψε από ένα προγενέστερο βασισμένο σε XML πρότυπο με όνομα XML-RPC και προηγείται ενός ανερχόμενου προτύπου με όνομα XML ηλεκτρονικής επιχείρησης (electronic business XML - ebXML). Το ebXML είναι έργο σε εξέλιξη που στόχο έχει να παρέχει ένα κατανοητό ορισμό των επιχειρησιακών μηνυμάτων που ανταλλάσσονται μεταξύ εμπορικών εταίρων. Το SOAP είναι πιο περιορισμένο σε έκταση και λιγότερο περίπλοκο στην εφαρμογή.
Σύμφωνα με τον ορισμό της διεθνούς κοινοπραξίας World Wide Web Consortium (W3C) η έκδοση 1.2 του SOAP είναι ένα ελαφρύ πρωτόκολλο που προορίζεται για την ανταλλαγή δομημένων πληροφοριών σε ένα κατανεμημένο περιβάλλον. Το πρωτόκολλο αυτό χρησιμοποιεί τις XML τεχνολογίες για να καθορίσει ένα επεκτάσιμο πλαίσιο ανταλλαγής μηνυμάτων που μπορεί να πραγματοποιηθεί σε ένα πλήθος διαφορετικών υποκείμενων πρωτοκόλλων. Το πλαίσιο σχεδιάστηκε να είναι ανεξάρτητο από κάθε συγκεκριμένο προγραμματιστικό μοντέλο και κάθε άλλη συγκεκριμένη σημασιολογία υλοποίησης.
Το πρωτόκολλο SOAP καταργεί την απαίτηση δύο συστήματα να εκτελούνται στην ίδια πλατφόρμα ή να είναι γραμμένα στην ίδια προγραμματιστική γλώσσα. Αντί για την κλήση των μεθόδων μέσα από ένα δυαδικό ιδιόκτητο πρωτόκολλο, ένα πακέτο SOAP χρησιμοποιεί XML, ένα συντακτικό με βάση το κείμενο για την πραγματοποίηση κλήσεων μεθόδων (method invocation) και λόγω αυτού του γεγονότος διαφοροποιείται από τα πλαίσια CORBA, DCOM και Java RMI που έχουν παρόμοια λειτουργικότητα. Όλες οι

πληροφορίες μεταξύ της αιτούσας εφαρμογής και του παραλήπτη αντικειμένου αποστέλλονται ως δεδομένα που επισημαίνονται με ετικέτες (tags) σε μια XML ροή μέσω HTTP. Από τη σκοπιά των υπηρεσιών Ιστού, το SOAP μπορεί να υλοποιηθεί είτε ως πελάτης είτε ως διακομιστής.

4.3.1 Μορφή μηνύματος SOAP

Ένα μήνυμα μιας κατεύθυνσης (one-way message), ένα αίτημα από πελάτη ή μια απάντηση από εξυπηρετητή ονομάζονται επίσημα SOAP μηνύματα. Το κύριο μέρος του μηνύματος έχει ένα MIME (επέκταση διαδικτυακού ταχυδρομείου πολλαπλών χρήσεων - Multipurpose Internet Mail Extensions) τύπου "text/xml" και περιέχει το φάκελο SOAP. Αυτός ο φάκελος (envelope) είναι ένα έγγραφο XML. Ο φάκελος περιέχει μια κεφαλίδα (header) προαιρετικά και ένα σώμα (body) υποχρεωτικά. Το μέρος του σώματος του φακέλου είναι πάντα προοριζόμενο για τον τελικό παραλήπτη του μηνύματος, ενώ οι καταχωρήσεις κεφαλίδας μπορούν να αφορούν τους κόμβους που εκτελούν ενδιάμεση επεξεργασία. Συνημμένα, δυαδικής μορφής ή άλλης, είναι δυνατόν να επισυνάπτονται στο σώμα του μηνύματος. Το σώμα συμπεριλαμβάνει το κύριο ωφέλιμο φορτίο (payload) του SOAP μηνύματος.
Το SOAP παρέχει έναν τρόπο για τον πελάτη να προσδιορίσει ποιος από τους ενδιάμεσους κόμβους επεξεργασίας πρέπει να ασχοληθεί με κάποια ορισμένη καταχώρηση κεφαλίδας. Επειδή οι κεφαλίδες είναι ανεξάρτητες του κύριου περιεχομένου του μηνύματος SOAP, είναι χρήσιμες για την προσθήκη πληροφοριών στο μήνυμα που δεν επηρεάζει την επεξεργασία του σώματος του μηνύματος.

4.3.2 SOAP Πελάτες και Εξυπηρετητές

Ένας SOAP πελάτης είναι ένα πρόγραμμα που δημιουργεί ένα XML έγγραφο που περιέχει τις αναγκαίες πληροφορίες για την απομακρυσμένη κλήση μιας μεθόδου σε ένα κατανεμημένο σύστημα. Ένας SOAP εξυπηρετητής είναι απλά ειδικός κώδικας που λαμβάνει SOAP μηνύματα και δρα ως διανεμητής και διερμηνέας SOAP εγγράφων. Εξωτερικές υπηρεσίες Ιστού μπορούν να αλληλεπιδρούν με εξυπηρετητές εφαρμογών βασισμένους σε τεχνολογία J2EE που επεξεργάζεται SOAP αιτήματα από διάφορους πελάτες. Οι SOAP εξυπηρετητές εξασφαλίζουν ότι τα έγγραφα που λαμβάνονται σε μια HTTP σύνδεση μετατρέπονται σε μια γλώσσα κατανοητή από το αντικείμενο στο άλλο άκρο της σύνδεσης.
Καθώς όλες οι επικοινωνίες γίνονται υπό τη μορφή XML, τα αντικείμενα σε μια γλώσσα μπορούν να επικοινωνήσουν μέσω SOAP με αντικείμενα σε οποιαδήποτε άλλη γλώσσα. Οι εφαρμογές πελάτη μπορούν να αλληλεπιδράσουν σε χαλαρά συνδεδεμένα (loosely coupled) περιβάλλοντα για να ανακαλύψουν και να συνδεθούν δυναμικά σε υπηρεσίες χωρίς να έχει πραγματοποιηθεί κάποια συμφωνία εκ των προτέρων ανάμεσά τους. Το πρωτόκολλο SOAP είναι επεκτάσιμο, επειδή οι πελάτες που το χρησιμοποιούν, οι εξυπηρετητές και το ίδιο το πρωτόκολλο μπορούν να εξελίσσονται χωρίς να αχρηστεύονται οι ήδη υπάρχουσες εφαρμογές. Επιπλέον το SOAP υποστηρίζει μεσάζοντες και αρχιτεκτονικές επιπέδων. Αυτό σημαίνει ότι μπορεί να υπάρχουν κόμβοι επεξεργασίας ανάμεσα στον πελάτη και στον εξυπηρετητή, δηλαδή στη διαδρομή την οποία ακολουθεί το αίτημα. Αυτοί οι ενδιάμεσοι κόμβοι επεξεργάζονται μέρη του μηνύματος που καθορίζονται από το SOAP με τη χρήση επικεφαλίδων (headers) που επιτρέπουν στους πελάτες να ταυτοποιούν ποιος κόμβος επεξεργάζεται κάθε μέρος του μηνύματος. Το SOAP παρέχει ένα γνώρισμα «mustUnderstand» για τις επικεφαλίδες που επιτρέπει στον πελάτη να καθορίζει αν η επεξεργασία είναι προαιρετική ή υποχρεωτική. Αν το γνώρισμα

«mustUnderstand» έχει τιμή 1, ο εξυπηρετητής είτε πραγματοποιεί την ενδιάμεση επεξεργασία που καθορίζεται στην επικεφαλίδα είτε ενημερώνει με ένα σφάλμα σε περίπτωση αποτυχίας επεξεργασίας.

4.3.3 SOAP ρόλοι και SOAP κόμβοι

Κατά την επεξεργασία ενός SOAP μηνύματος, ένας SOAP κόμβος λέγεται ότι δρα σύμφωνα με έναν ή περισσότερους SOAP ρόλους, ο καθένας από τους οποίους καθορίζεται από ένα ενιαίο αναγνωριστικό πόρου (URI) γνωστό ως το όνομα του SOAP ρόλου. Οι ρόλοι ενός κόμβου πρέπει να μένουν αναλλοίωτοι κατά την επεξεργασία ενός συγκεκριμένου μηνύματος. Τρία ονόματα ρόλων καθορίστηκαν να έχουν ειδική σημασία σε ένα SOAP μήνυμα:

- next: Σύμφωνα με αυτό το ρόλο, πρέπει να δράσει κάθε SOAP ενδιάμεσος και ο τελευταίος SOAP παραλήπτης.
- none: Οι SOAP κόμβοι δεν πρέπει να δρουν, σύμφωνα με αυτόν τον ρόλο.
- ultimateReceiver: Ο τελικός παραλήπτης πρέπει να δράσει, σύμφωνα με αυτόν τον ρόλο.

Ενώ ο σκοπός ενός ονόματος SOAP ρόλου είναι να καθορίσει ένα SOAP κόμβο ή κόμβους δεν υπάρχει σημασιολογία δρομολόγησης ή ανταλλαγής μηνυμάτων που να σχετίζεται με το όνομα του SOAP ρόλου.

4.3.4 SOAP-RPC

Το SOAP παρέχει ένα κατανεμημένο μοντέλο επεξεργασίας που υποθέτει ότι ένα SOAP μήνυμα προέρχεται από ένα SOAP αποστολέα και στέλνεται σε κάποιον τελικό SOAP παραλήπτη μέσω μηδέν ή περισσότερων SOAP ενδιάμεσων. Το μοντέλο επεξεργασίας SOAP μπορεί να υποστηρίξει πολλά σχήματα ανταλλαγής μηνυμάτων περιλαμβανομένων των μεταδόσεων μιας κατεύθυνσης, των αλληλεπιδράσεων αίτησης/απόκρισης και συνομιλιών μεταξύ ομότιμων (peer-to-peer conversations). Για να γίνουν απομακρυσμένες κλήσεις διαδικασιών (RPC) με τη χρήση SOAP, πρέπει να τηρούνται κάποιες συμβάσεις. Καταρχάς, τα μηνύματα αίτησης και απόκρισης πρέπει να κωδικοποιούνται ως δομές. Για κάθε παράμετρο εισόδου μιας λειτουργίας πρέπει να υπάρχει ένα στοιχείο (ή μέλος της δομής εισόδου) με το ίδιο όνομα όπως η παράμετρος. Επίσης, για κάθε παράμετρο εξόδου, πρέπει να υπάρχει ένα στοιχείο (ή μέλος της δομής εξόδου) με το αντίστοιχο όνομα.

ονόματος μιας λειτουργίας Request. Αυτή η δομή εξόδου περιέχει ένα στοιχείο που ονομάζεται τιμή (price), η οποία επιστρέφει τα αποτελέσματα της κλήσης της μεθόδου, πιθανώς ως πραγματικό αριθμό. Είναι σημαντικό να σημειωθεί ότι πουθενά στο φάκελο SOAP δεν καθορίζονται ρητά οι τύποι δεδομένων, επομένως δεν ξέρουμε πραγματικά τον τύπο του συμβόλου ή τον τύπο της επιστρεφόμενης παραμέτρου που ονομάζεται price βλέποντας μόνο το SOAP μήνυμα.

Ακολουθούν παραδείγματα SOAP μηνυμάτων:

- Soap_Request για την getHotelListByDate

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://services.thesis.vaggelis/">
   <soapenv:Header/>
   <soapenv:Body>
      <ser:getHotelListByDate>
         <!--Optional:-->
         <arg0>2018-01-01</arg0>
         <!--Optional:-->
         <arg1>2018-01-10</arg1>
         <!--Optional:-->
         <arg2>Crete</arg2>
      </ser:getHotelListByDate>
   </soapenv:Body>
</soapenv:Envelope>
```

- Soap_Response για την getHotelListByDate

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:getHotelListByDateResponse xmlns:ns2="http://services.thesis.vaggelis/">
      <return>
        <Hotel_Name>Villa Costas</Hotel_Name>
        <Hotel_price>280.0</Hotel_price>
      </return>
      <return>
        <Hotel_Name>Sea Hotel</Hotel_Name>
        <Hotel_price>156.0</Hotel_price>
      </return>
      <return>
        <Hotel_Name>Blue Hotel</Hotel_Name>
        <Hotel_price>150.0</Hotel_price>
      </return>
    </ns2:getHotelListByDateResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

- Soap_Request για την getHotelListByPrice

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://services.thesis.vaggelis/">
   <soapenv:Header/>
   <soapenv:Body>
      <ser:getHotelListByPrice>
         <!--Optional:-->
         <arg0>Crete</arg0>
         <arg1>2</arg1>
         <!--Optional:-->
         <arg2>high</arg2>
         <!--Optional:-->
         <arg3>500.0</arg3>
      </ser:getHotelListByPrice>
   </soapenv:Body>
</soapenv:Envelope>
```

- Soap_Response για την getHotelListByPrice

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:getHotelListByPriceResponse xmlns:ns2="http://services.thesis.vaggelis/">
      <return>
        <Hotel_Name>Sea Hotel</Hotel_Name>
        <Hotel_price>156.0</Hotel_price>
      </return>
    </ns2:getHotelListByPriceResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

- Soap_Request για την getListOfCarsByPrice

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://services.thesis.vaggelis/">
   <soapenv:Header/>
   <soapenv:Body>
      <ser:getListOfCarsByPrice>
         <!--Optional:-->
         <arg0>2018-01-01</arg0>
         <!--Optional:-->
         <arg1>2018-01-10</arg1>
         <!--Optional:-->
         <arg2>300.0</arg2>
         <!--Optional:-->
         <arg3>Crete</arg3>
      </ser:getListOfCarsByPrice>
   </soapenv:Body>
</soapenv:Envelope>
```

- Soap_Response για την getListOfCarsByPrice

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:getListOfCarsByPriceResponse xmlns:ns2="http://services.thesis.vaggelis/">
      <return>
        <brand>**Suzuki**</brand>
        <model>**Swift**</model>
        <colour>**red**</colour>
        <Insurance>@**nytime**</Insurance>
        <price>**120.0**</price>
      </return>
    </ns2:getListOfCarsByPriceResponse>
  </soapenv:Body>
</soapenv:Envelope>

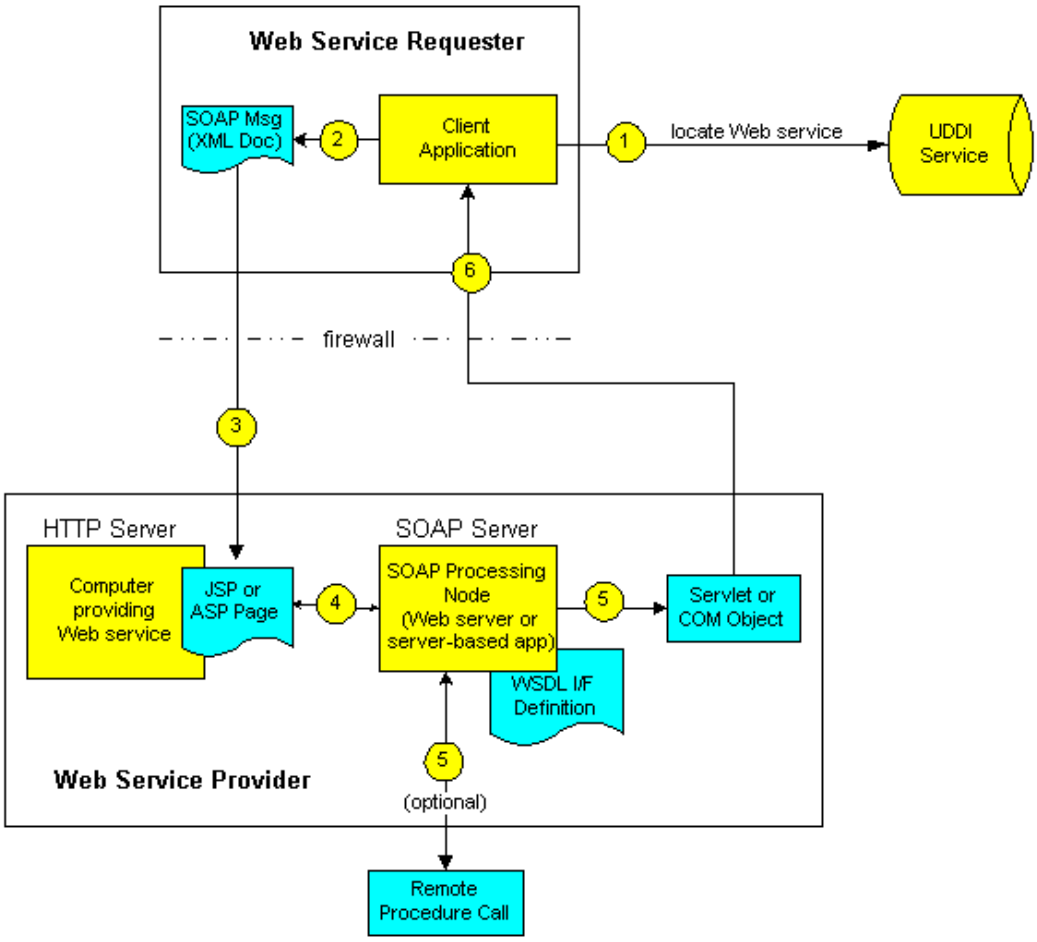
- Soap_Request για την getShipListByDate

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://services.thesis.vaggelis/">
   <soapenv:Header/>
   <soapenv:Body>
      <ser:getShipListByDate>
         <!--Optional:-->
         <arg0>2018-01-01</arg0>
         <!--Optional:-->
         <arg1>Crete</arg1>
         <!--Optional:-->
         <arg2>300.0</arg2>
      </ser:getShipListByDate>
   </soapenv:Body>
</soapenv:Envelope>
```

- Soap_Response για την getShipListByDate

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:getShipListByDateResponse xmlns:ns2="http://services.thesis.vaggelis/">
      <return>
        <Name>Elpida</Name>
        <capacity>2000</capacity>
        <companyName>Blue Star Ferries</companyName>
        <portName>Crete's Port</portName>
        <ShipBooking_price>140.0</ShipBooking_price>
      </return>
    </ns2:getShipListByDateResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

4.3.5 Μια περίπτωση χρήσης του Απλού Πρωτόκολλου Πρόσβασης Αντικειμένου (SOAP)



- Ένας SOAP πελάτης χρησιμοποιεί το μητρώο UDDI για να προσδιορίσει την τοποθεσία μιας υπηρεσίας Ιστού. Αντί να χειριστεί τη Γλώσσα Περιγραφής Υπηρεσιών Ιστού (WSDL) άμεσα, στις περισσότερες περιπτώσεις μια εφαρμογή SOAP θα είναι προγραμματισμένη να χρησιμοποιήσει ένα συγκεκριμένο τύπο θύρας και στυλ δέσμευσης (binding) και αυτό θα ρυθμίσει δυναμικά τη διεύθυνση της υπηρεσίας που θα κληθεί ώστε να ταιριάζει με αυτή που ανακαλύφθηκε με χρήση UDDI. (Βήμα 1 στο Σχήμα)
- Η εφαρμογή πελάτη σχηματίζει ένα SOAP μήνυμα, το οποίο είναι ένα XML έγγραφο ικανό να πραγματοποιήσει την επιθυμητή λειτουργία αίτησης/απόκρισης. (Βήμα 2)
- Ο πελάτης στέλνει το SOAP μήνυμα σε μια σελίδα JSP σε ένα εξυπηρετητή ιστού που περιμένει να λάβει SOAP αιτήσεις. (Βήμα 3)
- Ο SOAP εξυπηρετητής αναλύει το μήνυμα και καλεί την κατάλληλη μέθοδο του αντικειμένου στον τομέα του (domain), περνώντας τις παραμέτρους που περιλαμβάνονται στο SOAP έγγραφο. Προαιρετικά, κάποιοι ενδιάμεσοι κόμβοι επεξεργασίας ίσως να είχαν εκτελέσει ειδικές λειτουργίες όπως υποδεικνύεται από τις SOAP κεφαλίδες πριν την παραλαβή του μηνύματος από τον SOAP εξυπηρετητή. (Βήμα 4)
- Το αντικείμενο αίτησης (request object) πραγματοποιεί την κατάλληλη λειτουργία και επιστρέφει δεδομένα στον SOAP εξυπηρετητή ο οποίος περικλείει την απάντηση με SOAP φάκελο (envelope) που με τη σειρά του ενσωματώνεται σε ένα αντικείμενο απόκρισης (response object) π.χ. ένα servlet. (Βήμα 5)
- Ο πελάτης λαμβάνει το αντικείμενο, αφαιρεί το SOAP φάκελο και στέλνει το έγγραφο απάντησης στο πρόγραμμα που το ζήτησε εξ' αρχής ολοκληρώνοντας τον κύκλο αίτησης/απόκρισης. (Βήμα 6)

4.4 Γλώσσα Περιγραφής Υπηρεσιών Ιστού (Web Services Description Language – WSDL)

Η WSDL είναι μια προδιαγραφή που ορίζει πως μπορούν να περιγραφούν οι υπηρεσίες Ιστού σε μια κοινή XML γραμματική. Η WSDL περιγράφει τα ακόλουθα σημαντικά δεδομένα:

- Πληροφορίες διεπαφής που περιγράφουν όλες τις δημόσια διαθέσιμες λειτουργίες
- Πληροφορίες τύπου δεδομένων για όλα τα μηνύματα αίτησης και τα μηνύματα απόκρισης
- Πληροφορίες δέσμευσης για το πρωτόκολλο μεταφοράς που θα χρησιμοποιηθεί
- Πληροφορίες διεύθυνσης για την εύρεση τοποθεσίας της καθορισμένης υπηρεσίας

Συνοπτικά, η WSDL αναπαριστά ένα συμβόλαιο ανάμεσα στον αιτούμενο μια υπηρεσία πελάτη και τον πάροχο της υπηρεσίας με παρόμοιο τρόπο με αυτόν με τον οποίο μια διεπαφή Java αναπαριστά ένα συμβόλαιο ανάμεσα στον κώδικα του πελάτη και στο πραγματικό Java αντικείμενο. Η θεμελιώδης διαφορά είναι ότι η WSDL είναι ανεξάρτητη γλώσσας και πλατφόρμας και χρησιμοποιείται κυρίως (αλλά όχι αποκλειστικά) για την περιγραφή υπηρεσιών SOAP.
Με τη χρήση της WSDL ένας πελάτης μπορεί να βρει την τοποθεσία μιας υπηρεσίας ιστού και να καλέσει οποιαδήποτε από τις δημόσια διαθέσιμες συναρτήσεις της. Με τα κατάλληλα εργαλεία αυτή η διαδικασία αυτοματοποιείται επιτρέποντας στις εφαρμογές να ενσωματώνουν εύκολα νέες υπηρεσίες με την απαίτηση συγγραφής ελάχιστου ή καθόλου κώδικα. Κατ' αυτόν τον τρόπο η WSDL αποτελεί θεμέλιο λίθο της αρχιτεκτονικής υπηρεσιών Ιστού επειδή παρέχει μια κοινή γλώσσα για την περιγραφή υπηρεσιών και μια πλατφόρμα για την αυτόματη ενσωμάτωση αυτών των υπηρεσιών.
Μια περιγραφή υπηρεσίας δείχνει πως πρόκειται να αλληλεπιδράσουν με την υπηρεσία οι πιθανοί πελάτες. Διαβεβαιώνει ότι η περιγραφόμενη υπηρεσία υλοποιεί πλήρως και ανταποκρίνεται σε αυτό που περιγράφει το έγγραφο WSDL 2.0. Μια διεπαφή WSDL 2.0 περιγράφει πιθανές αλληλεπιδράσεις με μια υπηρεσία Ιστού, όχι απαιτούμενες αλληλεπιδράσεις. Η δήλωση μιας λειτουργίας σε διεπαφή WSDL 2.0 δεν αποτελεί διαβεβαίωση ότι η περιγραφόμενη από τη λειτουργία αλληλεπίδραση θα συμβεί οπωσδήποτε. Εξασφαλίζει όμως ότι αν μια τέτοια αλληλεπίδραση ξεκινήσει με κάποιο τρόπο, τότε η δηλωμένη λειτουργία περιγράφει πως θα συμβεί αυτή.

4.4.1 WSDL Έγγραφα (Documents)

Ένα WSDL έγγραφο είναι απλά ένα XML έγγραφο. Περιέχει ένα σύνολο ορισμών που περιγράφουν μια υπηρεσία Ιστού. Όσον αφορά τη δομή του εγγράφου, αυτό αποτελείται από τα ακόλουθα κύρια στοιχεία:

- definitions: πρέπει να αποτελεί το στοιχείο ρίζα όλων των εγγράφων WSDL. Καθορίζει το όνομα της υπηρεσίας ιστού, δηλώνει πολλαπλά πεδία ονομάτων (namespaces) που χρησιμοποιούνται στο υπόλοιπο έγγραφο και περιέχει όλα τα υπόλοιπα στοιχεία της υπηρεσίας που περιγράφονται στη συνέχεια.

```
<?xml version="1.0" encoding="UTF-8"?><!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net.
RI's version is JAX-WS RI 2.1.1 in JDK 6. -->
<definitions name="HotelBrokerService" targetNamespace="http://services.thesis.vaggelis/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://services.thesis.vaggelis/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

- types: το στοιχείο αυτό περιγράφει όλους τους τύπους δεδομένων που χρησιμοποιούνται ανάμεσα στον πελάτη και τον εξυπηρετητή. Δε σχετίζεται αποκλειστικά με ένα συγκεκριμένο σύστημα τύπων αλλά χρησιμοποιεί την προδιαγραφή XML Schema της W3C ως προεπιλογή. Αν η υπηρεσία χρησιμοποιεί μόνο απλούς τύπους που περιέχονται στο XML Schema, όπως συμβολοσειρές και ακεραίους, το στοιχείο `types` δε χρειάζεται.

```
<types>
  <xsd:schema>
    <xsd:import namespace="http://services.thesis.vaggelis/" schemaLocation="HotelBrokerService_schema1.xsd"/>
  </xsd:schema>
</types>
```

- message: το στοιχείο περιγράφει ένα μήνυμα μιας κατεύθυνσης είτε είναι ένα μόνο μήνυμα αίτησης είτε απόκρισης. Καθορίζει το όνομα του μηνύματος και περιέχει μηδέν ή περισσότερα στοιχεία message part που αναφέρονται στις παραμέτρους ή τις επιστρεφόμενες τιμές του μηνύματος. Συνεπώς, προσδιορίζει τι μηνύματα μεταδίδονται.

```
<message name="getHotelListByPrice">
  <part element="tns:getHotelListByPrice" name="parameters"/>
</message>
<message name="getHotelListByPriceResponse">
  <part element="tns:getHotelListByPriceResponse" name="parameters"/>
</message>
<message name="getHotelListByDate">
  <part element="tns:getHotelListByDate" name="parameters"/>
</message>
<message name="getHotelListByDateResponse">
  <part element="tns:getHotelListByDateResponse" name="parameters"/>
</message>
```

- portType: το στοιχείο portType συνδυάζει πολλαπλά στοιχεία message για να σχηματίσει μια ολοκληρωμένη λειτουργία και περιγράφει ποιες λειτουργίες υποστηρίζονται. Για παράδειγμα, ένα στοιχείο portType μπορεί να συνδυάζει ένα μήνυμα αίτησης και ένα μήνυμα απόκρισης σε μια λειτουργία αίτησης/απόκρισης. Το αντίστοιχο του portType σε μια παραδοσιακή προγραμματιστική γλώσσα είναι μια βιβλιοθήκη συναρτήσεων ή κλάση.

```
<portType name="HotelBrokerDelegate">
  <operation name="getHotelListByPrice">
    <input message="tns:getHotelListByPrice"/>
    <output message="tns:getHotelListByPriceResponse"/>
  </operation>
  <operation name="getHotelListByDate">
    <input message="tns:getHotelListByDate"/>
    <output message="tns:getHotelListByDateResponse"/>
  </operation>
</portType>
```

- binding: το στοιχείο binding περιγράφει πως θα υλοποιηθεί η μετάδοση της υπηρεσίας δηλαδή τα πρωτόκολλα επικοινωνίας που θα χρησιμοποιήσει και καθορίζει τη μορφή των μηνυμάτων.

```
<binding name="HotelBrokerPortBinding" type="tns:HotelBrokerDelegate">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="getHotelListByPrice">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
  <operation name="getHotelListByDate">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>
```

4.4.2 WSDL θύρες (ports)

Μια θύρα WSDL περιγράφει τις διεπαφές, δηλαδή τις νόμιμες λειτουργίες που δημοσιοποιούνται από μια υπηρεσία ιστού. Η θύρα καθορίζει το σημείο σύνδεσης σε μια υπηρεσία ιστού. Η WSDL καθορίζει 4 τύπους λειτουργιών από τους οποίους ο πιο κοινός είναι ο τύπος αίτησης-απόκρισης. Σ' αυτόν η λειτουργία λαμβάνει μια αίτηση και επιστρέφει μια απάντηση.

4.4.3 WSDL δέσμευση (binding)

Οι WSDL δεσμεύσεις καθορίζουν τη μορφή των μηνυμάτων και τις λεπτομέρειες του πρωτοκόλλου για μια υπηρεσία ιστού. Το στοιχείο binding έχει δύο γνωρίσματα: όνομα και τύπο. Το γνώρισμα «τύπος» αναφέρεται στο portType που ορίστηκε νωρίτερα στο έγγραφο.

4.5 Παγκόσμια περιγραφή, ανακάλυψη και ενσωμάτωση (Universal Description, Discovery and Integration)

Το έργο UDDI παρέχει μια τυποποιημένη μέθοδο για τη δημοσίευση και την ανακάλυψη πληροφοριών για τις υπηρεσίες ιστού. Ουσιαστικά πρόκειται για έναν κατάλογο διεπαφών υπηρεσιών ιστού που περιγράφονται σε WSDL. Για αυτό το λόγο η σημασία του UDDI στην ανερχόμενη στοίβα πρωτοκόλλων υπηρεσιών ιστού είναι αυξημένη, καθώς δίνει τη δυνατότητα στους οργανισμούς να δημοσιεύουν αλλά και να βρίσκουν τις υπηρεσίες ιστού που χρειάζονται.

4.5.1 Εισαγωγή

Το UDDI αποτελείται από δύο μέρη. Κατ' αρχάς είναι μια τεχνική προδιαγραφή για τη δημιουργία ενός κατανεμημένου καταλόγου επιχειρήσεων και υπηρεσιών ιστού. Τα δεδομένα αποθηκεύονται σε μία συγκεκριμένη μορφή XML και η προδιαγραφή UDDI περιλαμβάνει λεπτομέρειες Διεπαφής Προγραμματισμού Εφαρμογών (Application Programming Interface – API) για την αναζήτηση υπαρχόντων δεδομένων και τη δημοσίευση νέων. Επιπροσθέτως, το μητρώο επιχειρήσεων UDDI (UDDI Business Registry ή UDDI "cloud services") είναι μια πλήρως λειτουργική υλοποίηση της προδιαγραφής UDDI.
Μια επιχείρηση μπορεί να καταχωρήσει τρεις τύπους πληροφορίας σε ένα μητρώο UDDI. Αυτοί οι τρεις τύποι συνοψίζουν τι μπορεί να αποθηκεύσει το UDDI για μια επιχείρηση.

- Λευκές σελίδες (White pages): Περιλαμβάνουν βασικές πληροφορίες επικοινωνίας και αναγνωριστικά για μια επιχείρηση, τα οποία περιλαμβάνουν το όνομα της επιχείρησης, την περιγραφή της επιχείρησης, τη διεύθυνση και το τηλέφωνο αλλά και μοναδικά αναγνωριστικά της επιχείρησης.
- Κίτρινες σελίδες (Yellow pages): Περιλαμβάνουν πληροφορίες που περιγράφουν μια υπηρεσία Ιστού χρησιμοποιώντας διαφορετικές κατηγοριοποιήσεις (taxonomies) που επιτρέπουν την ανακάλυψη της υπηρεσίας με βάση συγκεκριμένες κατηγορίες όπου ανήκει.
- Πράσινες σελίδες (Green pages): Περιέχουν τεχνικές πληροφορίες για μια υπηρεσία ιστού. Δηλαδή περιγράφουν τη συμπεριφορά και τις υποστηριζόμενες λειτουργίες μιας υπηρεσίας ιστού. Γενικά, αυτό περιλαμβάνει ένα δείκτη σε πληροφορίες ομαδοποίησης υπηρεσιών ιστού και μια διεύθυνση για την κλήση της υπηρεσίας ιστού.

4.5.2 Τα οφέλη του UDDI και η σταδιακή εγκατάλειψή του

Πριν από το UDDI δεν υπήρχε πρότυπο του Διαδικτύου που να επιτρέπει στις επιχειρήσεις να προσεγγίσουν τους πελάτες και τους συνεργάτες τους με πληροφορίες για τα προϊόντα και τις υπηρεσίες τους. Επίσης δεν υπήρχε τρόπος να ενσωματωθούν οι υπηρεσίες αυτές στις διαδικασίες και στα συστήματα των άλλων επιχειρήσεων. Η προδιαγραφή UDDI θα μπορούσε να βοηθήσει μια επιχείρηση με τους εξής τρόπους:

- Δίνει τη δυνατότητα ανακάλυψης της κατάλληλης επιχείρησης από τις εκατομμύρια που είναι διαθέσιμες στο διαδίκτυο
- Καθορίζει πως θα γίνει η συναλλαγή μόλις ανακαλυφθεί η κατάλληλη επιχείρηση
- Προσέγγιση περισσότερων πελατών και αύξηση της πρόσβασης στους τωρινούς πελάτες
- Αύξηση του μεριδίου της επιχείρησης στην αγορά
- Περιγραφή επιχειρησιακών διαδικασιών και υπηρεσιών προγραμματιστικά σε ένα μοναδικό, ανοιχτό και ασφαλές περιβάλλον

Παρά τις πολλές προσπάθειες για την προαγωγή του ως μέρος των SOA πλατφορμών διακυβέρνησης, το UDDI έχει αποδειχθεί ότι είναι ένας εξαιρετικά αναποτελεσματικός μηχανισμός προκειμένου να επιτρέψει τη δημοσίευση των υπηρεσιών και την ανακάλυψή τους. Αυτό που κάνει τη διαδικασία αυτή υπερβολικά δύσκολη είναι η πολυπλοκότητα της διεπαφής προγραμματισμού εφαρμογών του UDDI (UDDI API) και του μοντέλου. Για παράδειγμα ένας πελάτης γραμμένος σε μια δυναμική γλώσσα όπως η Ruby, που είναι γνωστό ότι δεν ανταποκρίνεται τόσο επιτυχημένα στην προδιαγραφή SOAP, είναι αρκετά δύσκολο να αλληλεπιδράσει με ένα αποθετήριο (repository) UDDI. Τέλος, τα μοντέλα που δημιουργήθηκαν με UDDI είναι πολύ πολύπλοκα για να υλοποιηθούν και να χρησιμοποιηθούν και καταλήγουν στην πράξη να γίνουν εμπόδιο στην προσανατολισμένη στις υπηρεσίες αρχιτεκτονική.

5.ΤΕΧΝΟΛΟΓΙΑ ΥΠΗΡΕΣΙΩΝ ΙΣΤΟΥ

5.1 Ατομικές και σύνθετες υπηρεσίες

Ορισμός : *«Οι ατομικές υπηρεσίες δεν βασίζονται σε άλλες υπηρεσίες και συνήθως σχετίζονται με απλές επιχειρησιακές συναλλαγές ή με εκτέλεση επερωτήσεων δεδομένων και ενημερώσεις δεδομένων.»*
Ορισμός: *«Οι σύνθετες υπηρεσίες χρησιμοποιούν άλλες υπηρεσίες. Κατά τα άλλα είναι σαν κανονικές υπηρεσίες, δηλαδή έχουν ένα καλά ορισμένο συμβόλαιο υπηρεσίας (service contract), είναι καταχωρημένες στο μητρώο υπηρεσιών (service registry), μπορούν να βρεθούν και να κληθούν όπως κάθε πάροχος υπηρεσιών.»*
Επίσης, αναπτύσσονται, διευθύνονται και ασφαλίζονται όπως οι άλλες υπηρεσίες, μπορούν να ψάξουν και να χρησιμοποιήσουν άλλες υπηρεσίες, απλές ή σύνθετες. Μπορεί να μοντελοποιούν μια μόνο συναλλαγή ή μια διαδικασία που χρειάζεται μεγάλο διάστημα για να ολοκληρωθεί. Για τον έλεγχο της υπηρεσίας αποθηκεύονται πληροφορίες κατάστασης της διαδικασίας.
Οι σύνθετες υπηρεσίες ορίζονται είτε δηλωτικά με χρήση WS-BPEL ή προγραμματιστικά με μια γλώσσα όπως Python, Perl, Java, C++ ή C#. Η δηλωτική προσέγγιση (declarative) είναι πιο ευέλικτη καθώς είναι πιο εύκολο να αλλάξει. Η προγραμματιστική προσέγγιση είναι καταλληλότερη όταν απαιτείται πιο εξειδικευμένη επεξεργασία.

5.2 Σύγκριση CORBA με τις Υπηρεσίες Ιστού για SOA

Άτομα οικεία με το πρότυπο CORBA συχνά παρατηρούν ότι οι Υπηρεσίες Ιστού είναι απλά υλοποιήσεις CORBA χρησιμοποιώντας XML και τονίζουν ότι συγκριτικά με την CORBA οι υπηρεσίες Ιστού δεν έχουν πολλά από τα θετικά χαρακτηριστικά της. Πράγματι οι αρχικοί στόχοι της CORBA είναι πολύ παρόμοιοι με τους στόχους των Υπηρεσιών Ιστού.

- Το κυριότερο ελάττωμα της CORBA σε σχέση με τις υπηρεσίες ιστού είναι ότι δεν καθορίζει ένα πρότυπο για διαλειτουργικότητα (interoperability). Οι υπηρεσίες Ιστού όμως ξεκίνησαν με το πρωτόκολλο SOAP που είναι πρότυπο διαλειτουργικότητας.
- Μια δεύτερη διαφορά μεταξύ των δύο είναι ότι είναι αδύνατο να χρησιμοποιηθεί η μεταφορά διαλειτουργικότητας στην CORBA χωρίς τη γλώσσα ορισμού διεπαφών (Interface Definition Language – IDL), ενώ το πρωτόκολλο SOAP χρησιμοποιείται άνετα χωρίς τη γλώσσα περιγραφής υπηρεσιών Ιστού (WSDL).

- Από τεχνική άποψη η CORBA φαίνεται να υπερέχει των υπηρεσιών Ιστού αλλά από την οπτική γωνία του χρήστη οι υπηρεσίες Ιστού δημοσιοποιούνται (publish) και χρησιμοποιούνται πιο εύκολα. Επίσης η υποστήριξη της διαλειτουργικότητας τις κάνει να υπερισχύουν.

5.3 Δημιουργία Υπηρεσιών Ιστού με JAX-WS

5.3.1 Εισαγωγή

Η Java Διεπαφή Προγραμματισμού Εφαρμογών για XML Υπηρεσίες Ιστού (Java API for XML Web Services (JAX-WS) είναι μια διεπαφή προγραμματισμού εφαρμογών υψηλού επιπέδου για την «κατανάλωση» (consume) και παροχή (provide) Υπηρεσιών Ιστού (ΥΙ). Η JAX-WS αντικατέστησε την παλαιότερη JAX-RPC API και υποστηρίζει εξίσου ΥΙ που χρησιμοποιούν μηνύματα (message-oriented) αλλά και απομακρυσμένες κλήσεις διαδικασίας (RPC-oriented). Η JAX-WS χρησιμοποιεί σήμανση (annotations), για να απλοποιήσει την ανάπτυξη και εγκατάσταση στον εξυπηρετητή των πελατών ΥΙ και των τερματικών σημείων (endpoints) π.χ. java servlet που λαμβάνουν τα SOAP μηνύματα, τα επεξεργάζονται και επιστρέφουν κάποιο αποτέλεσμα.
Σε αντίθεση με τη διεπαφή SOAP with Attachments API for Java (SAAJ), η JAX-WS δεν απαιτεί εκτεταμένη γνώση XML και WSDL και ολόκληρο το επίπεδο XML μπορεί να αποκρύπτεται από τους προγραμματιστές παρέχοντας ευκολότερη συντήρηση των πελατών (client maintenance). Ουσιαστικά, η JAX-WS στηρίζεται στην SAAJ την οποία χρησιμοποιεί για την επεξεργασία και την επικοινωνία. Μια υπηρεσία ιστού αναπαρίσταται σε Γλώσσα Περιγραφής ΥΙ (WSDL), η οποία καθορίζει XML τύπους για κάθε μέρος μηνύματος (message part) που χρησιμοποιείται σε αιτήσεις και αποκρίσεις υπηρεσιών. Προκειμένου να δημιουργηθούν αναπαραστάσεις αντικειμένων αυτών των τύπων, είναι αναγκαία μια γλώσσα δέσμευσης (binding language) για τη μετατροπή (marshal) από Java σε XML και αντίστροφα. Στην παλαιότερη API, JAX-RPC, αυτό γινόταν απευθείας εσωτερικά με το δικό της μηχανισμό μετατροπής. Λόγω της πολυπλοκότητας της προδιαγραφής όμως, η JAX-WS χρησιμοποιεί την ανεξάρτητη προδιαγραφή JAXB 2.0 (Java API for XML Binding specification).

Μια υπηρεσία ιστού είναι μια μονάδα διαχειριζόμενου κώδικα που μπορεί να ενεργοποιηθεί εξ αποστάσεως με τη χρήση HTTP, δηλαδή μπορεί να ενεργοποιηθεί χρησιμοποιώντας αιτήσεις HTTP. Οι υπηρεσίες Web σάς επιτρέπουν να εκθέσετε τις λειτουργίες του υπάρχοντος κώδικα σας μέσω του δικτύου. Αφού εκτεθεί στο δίκτυο, άλλη εφαρμογή μπορεί να χρησιμοποιήσει τη λειτουργικότητα του προγράμματος.

- Διαλειτουργικότητα

Οι υπηρεσίες Web επιτρέπουν σε διάφορες εφαρμογές να συζητούν μεταξύ τους και να μοιράζονται δεδομένα και υπηρεσίες μεταξύ τους. Άλλες εφαρμογές μπορούν επίσης να χρησιμοποιούν τις υπηρεσίες ιστού. Οι υπηρεσίες Web χρησιμοποιούνται για να κάνουν

την πλατφόρμα εφαρμογής και την τεχνολογία ανεξάρτητη.

- Τυποποιημένο πρωτόκολλο

Οι υπηρεσίες Web χρησιμοποιούν τυποποιημένο πρότυπο πρωτόκολλο για την επικοινωνία. Όλα τα τέσσερα επίπεδα (Service Transport, XML Messaging, Service Description και Service Discovery layers) χρησιμοποιούν καλά καθορισμένα πρωτόκολλα στη στοίβα πρωτοκόλλου υπηρεσιών web. Αυτή η τυποποίηση της στοίβας πρωτοκόλλων δίνει στην επιχείρηση πολλά πλεονεκτήματα, όπως ένα ευρύ φάσμα επιλογών, μείωση του κόστους λόγω ανταγωνισμού και αύξηση της ποιότητας.

- Επικοινωνία χαμηλού κόστους

Οι υπηρεσίες Web χρησιμοποιούν το πρωτόκολλο SOAP μέσω πρωτοκόλλου HTTP, ώστε να μπορείτε να χρησιμοποιήσετε το υπάρχον χαμηλού κόστους διαδίκτυο για την υλοποίηση υπηρεσιών ιστού. Οι υπηρεσίες Ιστού μπορούν επίσης να εφαρμοστούν σε άλλους αξιόπιστους μηχανισμούς μεταφοράς όπως το FTP.

5.3.2 Χρήση των ιδιοτήτων ονομάτων της σήμανσης JAX-WS (JAX-WS annotation name properties)

WebService.targetNamespace: Η σήμανση @WebService δείχνει ότι μια Java κλάση υλοποιεί μια υπηρεσία Ιστού. Αν χρησιμοποιείται σε μια διεπαφή, αντιπροσωπεύει τον ορισμό διεπαφής μιας υπηρεσίας Ιστού. Μια διεπαφή τερματικού σημείου υπηρεσίας (service endpoint) αντιστοιχεί σε ένα στοιχείο wsdl:portType. Η ιδιότητα targetNamespace αλλάζει το πεδίο ονομάτων στο οποίο καθορίζεται η υπηρεσία.
WebService.name: η ιδιότητα αυτή αλλάζει το όνομα θύρας (port) στις παραγόμενες @WebServiceClient κλάσεις. Είναι σημαντικό να γίνει διάκριση μεταξύ του παραγόμενου στελέχους, το οποίο είναι μια διεπαφή σχολιασμένη με @WebService (και το οποίο παίρνει το όνομα του από την τιμή της ιδιότητας name στο σχολιασμό @WebService στην κλάση που πραγματικά υλοποιεί την επιχειρησιακή λογική της υπηρεσίας Ιστού) και της πραγματικής υλοποίησης αν ο πάροχος και ο πελάτης της υπηρεσίας Ιστού βρίσκονται στην ίδια JVM, όπως ίσως συμβαίνει με ένα java servlet.
WebService.serviceName: αυτή η ιδιότητα ορίζει το URL της deployed υπηρεσίας. Μπορεί επίσης να τροποποιήσει το όνομα της παραγόμενης κλάσης υπηρεσίας.
WebService.portName: Αυτή η ιδιότητα αλλάζει το όνομα της μεθόδου που επιστρέφει το στέλεχος που υλοποιεί την υπηρεσία από την παραγόμενη κλάση υπηρεσίας.

5.3.3 Χρήση υπηρεσίας Ιστού από κάποιο Servlet ή EJB

Εναλλακτικά, αντί να δημιουργηθεί αυτόματα ο πελάτης που θα καλεί την υπηρεσία με βάση την περιγραφή WSDL της ΥΙ υπάρχει η δυνατότητα να δράσει ως πελάτης κάποιο EJB, servlet ή άλλος διαχειριζόμενος από τον container πόρος (container-managed resources). Στην περίπτωση αυτή χρησιμοποιείται η σήμανση @WebServiceRef για να ενσωματώσει μια αναφορά για την υπηρεσία που πρέπει να καλεστεί. Τα βήματα που απαιτούνται για να γραφτεί ένα servlet που θα καλεί την υπηρεσία Ιστού με χρήση του σχολιασμού @WebServiceRef είναι:

- Δημιουργία διεπαφής την οποία θα υλοποιεί η υπηρεσία Ιστού και στην οποία θα αναφέρονται οι πελάτες.
- Υλοποίηση της υπηρεσίας Ιστού.
- Προσθήκη της σήμανσης αναφοράς (reference annotation) στο java servlet: ένα στιγμιότυπο της παραγόμενης διεπαφής τερματικού σημείου υπηρεσίας (SEI) θα παρεμβληθεί.

Υπάρχουν δύο τρόποι να χρησιμοποιηθεί η σήμανση @WebServiceRef:

- Πιθανόν η σήμανση να αναφέρεται σε μια κλάση παραγόμενη από υπηρεσία. Αν η προκαθορισμένη τιμή δε μπορεί να εξαχθεί από το πεδίο ή τη μέθοδο την οποία τροποποιεί η σήμανση, τότε πρέπει να παρέχεται τουλάχιστον μια τιμή για την ιδιότητα type, που αναφέρεται στον τύπο της παραγόμενης κλάσης.
- Μπορεί η σήμανση να αναφέρεται σε μια διεπαφή τερματικού σημείου υπηρεσίας (SEI). Μ' αυτήν την επιλογή πρέπει να παρέχεται ένα αντικείμενο της κλάσης τύπου υπηρεσίας για την ιδιότητα value.

5.3.4 Χρήση ενός σχολιασμένου με JAXB στιγμιότυπου σε ένα SOAP μήνυμα

Αν είναι επιθυμητό να χρησιμοποιηθεί μια αντικειμενοστραφής όψη των περιεχομένων της SOAP αίτησης, αντί για το χαμηλό επίπεδο της XML για να φτιαχτούν μηνύματα και υπάρχει ένα java αντικείμενο που δημιουργήθηκε από μια JAXB σχολιασμένη κλάση, είναι δυνατόν το αντικείμενο αυτό να μετατραπεί σε XML και να χρησιμοποιηθεί ως στοιχείο στο σώμα της SOAP αίτησης (payload).
Ένα σημαντικό χαρακτηριστικό που φαίνεται από αυτή τη δυνατότητα είναι ότι με τη JAX-WS είναι εύκολο να χρησιμοποιηθούν βασικοί καθώς και σύνθετοι τύποι (complex types) ως παράμετροι των λειτουργιών της υπηρεσίας.

5.4 Παροχή υπηρεσιών Ιστού βασισμένων σε SOAP

5.4.1 Δημιουργία μιας βασικής Υπηρεσίας Ιστού

Σ' αυτή την ενότητα περιγράφεται η δημιουργία μιας βασικής αλλά ταυτόχρονα ρεαλιστικής υπηρεσίας ιστού σε Java EE 5.
Η διαδικασία που ακολουθείται δεν είναι πολύπλοκη:

- Πρώτα δημιουργείται ένα EJB ή servlet το οποίο επισημαίνεται με τη σήμανση @WebService και ανήκει σε κάποιο πακέτο όπως θα συνέβαινε σε οποιαδήποτε κανονική εφαρμογή.
- Το EJB εγκαθίσταται στον εξυπηρετητή και η JAX-WS αναλαμβάνει τη δημιουργία της περιγραφής WSDL και τις απαραίτητες αντιστοιχίες κατά το χρόνο εγκατάστασης (deploy time).

5.4.2 Καθορισμός χώρου ονομάτων (namespace)

Μπορεί να γίνει με την ιδιότητα targetNamespace της σήμανσης @WebService για να δηλώσει το χώρο ονομάτων για την υπηρεσία. Η ιδιότητα αυτή είναι προαιρετική και επιτρέπει να επανακαθοριστεί το πεδίο ονομάτων με τιμή διαφορετική της προκαθορισμένης τιμής. Επίσης οι σημάνσεις WebResult και WebParam μπορούν να

λάβουν αυτή την ιδιότητα και να μην πάρουν την προκαθορισμένη τιμή που κληρονομείται από τον χώρο ονομάτων της υπηρεσίας.

5.4.3 Δημιουργία μιας λειτουργίας υπηρεσίας Ιστού

Προσθέτοντας μια μέθοδο με δημόσια ορατότητα στην κλάση που υλοποιεί την υπηρεσία (και που ορίζεται από τη σήμανση @WebService), αυτή προστίθεται αυτόματα και στο παραγόμενο WSDL αρχείο. Ο ρητός προσδιορισμός μιας Java μεθόδου ως λειτουργίας υπηρεσίας γίνεται με τη σήμανση με @WebMethod. Ο σχολιασμός αυτός επιτρέπει τον καθορισμό συγκεκριμένων ιδιοτήτων που αφορούν την αναπαράσταση και τη συμπεριφορά της λειτουργίας στο WSDL. Ο σχολιασμός @WebMethod χρησιμοποιείται στην προσέγγιση που ξεκινάει από την Java για να δείξει τι θα πρέπει να
να δημιουργηθεί για το WSDL ως το όνομα λειτουργίας: αν το σχόλιο δεν χρησιμοποιείται, το όνομα της λειτουργίας θα ταιριάζει με το όνομα της μεθόδου. Όμως αν χρησιμοποιούμε την προσέγγιση που ξεκινάει από το WSDL, η ιδιότητα operationName δίνει τη δυνατότητα να ονομαστεί η Java μέθοδος όπως θέλουμε αλλά να αντιστοιχεί στην WSDL λειτουργία.

5.4.4 Προσδιορισμός του μέρους μηνύματος της υπηρεσίας Ιστού (Web Service message part) και καθορισμός επιστρεφόμενης τιμής λειτουργίας

Η σήμανση @WebParam χρησιμοποιείται για να προσδιορίσει την αντιστοιχία του ονόματος της παραμέτρου κάποιας μεθόδου με κάποιο όνομα wsdl:part. Παρέχει τρόπους να προσδιοριστεί το πεδίο ονομάτων προορισμού στο οποίο ορίζεται αυτό το part, αν αυτή η παράμετρος θα αποσπαστεί από την επικεφαλίδα (header), ποιο θα είναι το όνομα της παραμέτρου στο WSDL και την κατεύθυνση της ροής της παραμέτρου (in, out, in/out). Η απλούστερη και πλέον ορατή αλλαγή που μπορεί να γίνει είναι η ιδιότητα WebParam.name.
Η σήμανση @WebResult αντιστοιχεί μια επιστρεφόμενη τιμή κάποιας Java μεθόδου σε ένα αποτέλεσμα WSDL (wsdl:output).

5.5 RESTful Υπηρεσίες Ιστού

Η SOAP είναι μια προδιαγραφή, το ίδιο ισχύει για τη γλώσσα περιγραφής υπηρεσιών Ιστού (WSDL) και το XML Schema. Παρόλο που σχεδιαστές και προγραμματιστές ίσως διαφωνούν στο πώς ακριβώς πρέπει να γράφονται και να χρησιμοποιούνται υλοποιήσεις των βασισμένων σε SOAP υπηρεσιών Ιστού δεν υπάρχει αμφισβήτηση ότι αυτά είναι συγκεκριμένα και υπαρκτά.
Η SOA αντιθέτως είναι εκ φύσεως πιο εννοιολογική. Οι αρχές της είναι ξεκάθαρες αλλά στην πραγματικότητα αναπαριστούν στυλ και στρατηγικές αρχιτεκτονικής. Έτσι, ενώ γενικά οι περισσότεροι συμφωνούν στις βασικές αρχές της προσανατολισμένης στις υπηρεσίες αρχιτεκτονικής (π.χ. χαλαρή σύνδεση – loose coupling) οι λεπτομέρειες είναι περισσότερο αμφιλεγόμενες.
Το REST είναι πνευματικό παιδί του Roy T. Fielding, συνιδρυτή του έργου Apache HTTP server και ενός εκ των συγγραφέων των προτύπων HTTP και URI. Ο Fielding όρισε το REST σε μια διδακτορική διατριβή που έγραψε το 2000 με θέμα «Αρχιτεκτονικά στυλ και σχεδιασμός αρχιτεκτονικών λογισμικού βασισμένων σε δίκτυο». Ο ισχυρισμός του Fielding είναι ότι «το αρχιτεκτονικό στυλ REST χρησιμοποιήθηκε για να καθοδηγήσει το σχεδιασμό και την ανάπτυξη της αρχιτεκτονικής του μοντέρνου Ιστού». Το REST δεν είναι ένας εναλλακτικός τρόπος να γίνουν υπηρεσίες ιστού που ελκύει περισσότερο όσους προτιμούν τις PHP και Rails από τις Java και .NET. Ο Ιστός ουσιαστικά δομήθηκε με βάση τις αρχές RESTful αλλά αυτές ονομάστηκαν έτσι μερικά χρόνια αργότερα.

Οι αναλυτές του κλάδου έχουν πρόσφατα επινοήσει τον όρο WOA, ή Web Oriented Αρχιτεκτονική. Αυτή συνήθως χρησιμοποιείται για να δηλώσει μια αρχιτεκτονική υπηρεσίας που χρησιμοποιεί τις XML, JSON (JavaScript Object Notation), και HTTP. Ο όρος γενικά έχει σχεδιαστεί ώστε να συνεπάγεται δύο πράγματα: μια αρχιτεκτονική που 1) δεν χρησιμοποιεί WS-* προδιαγραφές και 2) μια εφαρμογή των RESTful αρχών που είναι πολύ πιο χαλαρή από ό,τι η διατριβή του Fielding επιτρέπει στην πραγματικότητα.

5.5.1 Οι αρχές του REST

Το REST είναι ακρωνύμιο των λέξεων REpresentational State Transfer, δηλαδή Αντιπροσωπευτική Μεταφορά Κατάστασης. Επειδή είναι αρχιτεκτονικό στυλ υπάρχει μια ελευθερία στο πώς κατανοούνται συγκεκριμένοι περιορισμοί που θέτει. Ακολουθούν τα βασικά γνωρίσματα του REST όπως παρουσιάζονται στη διατριβή του Fielding:

- *Οι REST υπηρεσίες δεν έχουν κατάσταση (stateless)*. Δε θα πρέπει να απαιτείται η φύλαξη κατάστασης της υπηρεσίας στον εξυπηρετητή. Όπως αναφέρει ο Fielding στη διατριβή του: *«Κάθε αίτημα από πελάτη σε εξυπηρετητή πρέπει να περιέχει όλες τις απαραίτητες πληροφορίες για να γίνει κατανοητό αίτημα και δεν μπορεί να χρησιμοποιεί για το σκοπό αυτό συναφές περιεχόμενο αποθηκευμένο στον εξυπηρετητή»*.
- Συνεπώς, δεν χρειάζονται σύνοδοι εξυπηρετητή (server sessions) και οτιδήποτε είναι αναγκαίο για την επεξεργασία του αιτήματος θα περιλαμβάνεται στο αίτημα.
- Οι REST υπηρεσίες έχουν μια ομοιόμορφη διεπαφή. Δεν υπάρχει WSDL στο REST. Αυτός ο περιορισμός συνήθως εκλαμβάνεται ως τη διεπαφή που παρέχεται από τις πρότυπες HTTP μεθόδους αλλά στην πραγματικότητα το πρωτόκολλο είναι ανεξάρτητο παρά τη δημοφιλή εξάρτηση από το HTTP.
- Ο χειρισμός των πόρων γίνεται μέσω αναπαραστάσεων. Οι συνιστώσες του συστήματος ανταλλάσουν δεδομένα (συνήθως XML έγγραφα) που αναπαριστούν τους πόρους. Μια αναπαράσταση είναι απλά μια ακολουθία από bytes και μεταδεδομένα στη μορφή ζευγών ονόματος/δεδομένων που περιγράφουν τον πόρο.
- Τα μηνύματα είναι αυτό-περιγραφικά. Δεν απαιτείται διαπραγμάτευση για το πώς θα γίνει η επικοινωνία με την υπηρεσία.
- Οι RESTful αρχιτεκτονικές δομούνται με πόρους. Καθένας από τους πόρους έχει το δικό του μοναδικό ενιαίο αναγνωριστικό πόρου (URI). Κάθε αντικείμενο πληροφορίας που μπορεί να ονομαστεί μπορεί να είναι πόρος π.χ. μια τιμή μετοχής σε κάποια συγκεκριμένη χρονική στιγμή, μια παραγγελία αγοράς κ.λπ.
- Τα υπερμέσα (hypermedia) ως μηχανή κατάστασης της εφαρμογής. Κάθε έγγραφο που επιστρέφεται από τον εξυπηρετητή θα περιλαμβάνει όλα τα URIs για κάθε επόμενο βήμα. Δηλαδή όλες οι πιθανές καταστάσεις εφαρμογής όπου μπορεί να μεταβεί ο χρήστης από την τρέχουσα κατάσταση αναπαρίστανται ως URIs πόρων (σύνδεσμοι υπερμέσων). Η κατάσταση της εφαρμογής οδηγείται σε μια νέα κατάσταση με την επιλογή ενός URI.

5.5.2 Πλεονεκτήματα του REST

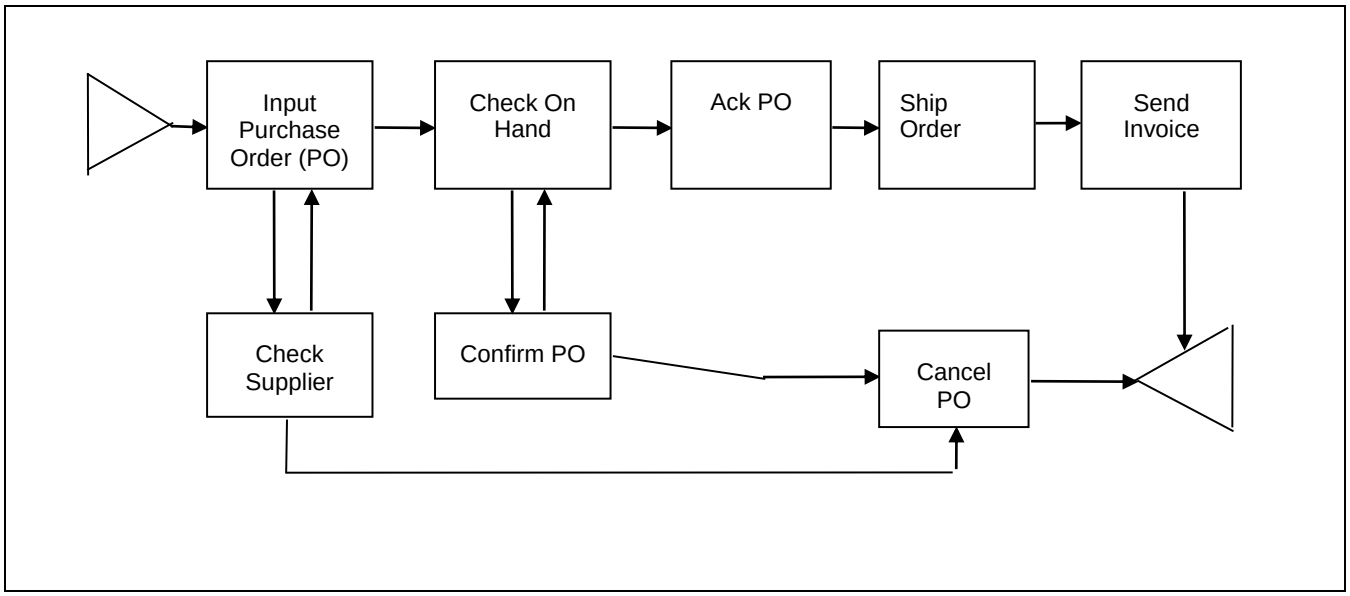
- Αυτή η αρχιτεκτονική παρέχει αρκετά πλεονεκτήματα στην πλευρά του εξυπηρετητή. Αυτά περιλαμβάνουν την επιπρόσθετη δυνατότητα της οριζόντιας κλιμάκωσης, ένα σαφή μηχανισμό για απόκρυψη (caching) και μια στρατηγική ανάκαμψης. Αυτά τα οφέλη προκύπτουν από τη δόμηση με βάση την αρχιτεκτονική του Ιστού στη στατική του μορφή. Υπάρχουν όμως πλεονεκτήματα και από τη χρήση REST στην πλευρά του πελάτη. Αυτά περιλαμβάνουν την ικανότητα να αποκρύπτονται ή να

φυλάσσονται αναπαραστάσεις και η ευέλικτη επιλογή μορφών δεδομένων που είναι καταλληλότερες για κάποια συγκεκριμένη περίπτωση χρήσης. Επίσης, οι βασισμένες σε SOAP υπηρεσίες ιστού είναι συχνά άσκοπα περίπλοκες, ενώ το REST είναι απλούστερο και αναπαριστά ένα μικρό και ευρέως κατανοητό σύνολο περιορισμών στον Ιστό.

6. ΤΕΧΝΟΛΟΓΙΑ ΕΠΙΧΕΙΡΗΣΙΑΚΩΝ ΔΙΑΔΙΚΑΣΙΩΝ / BPEL ΚΑΙ SOA

6.1 Διαχείριση Επιχειρησιακών Διαδικασιών (Business Process Management – BPM)

Μια επιχειρησιακή διαδικασία είναι μια δραστηριότητα του πραγματικού κόσμου που αποτελείται από ένα σύνολο λογικά συνδεδεμένων εργασιών, που όταν πραγματοποιούνται στην κατάλληλη σειρά και σύμφωνα με σωστούς επιχειρησιακούς κανόνες παράγουν ένα αποτέλεσμα που έχει νόημα για την επιχείρηση π.χ. συμπλήρωση παραγγελίας αγορών ή επεξεργασία αίτησης ασφάλισης.
Στο ακόλουθο σχήμα (**Error! Reference source not found.**) απεικονίζεται ένα παράδειγμα ροής επιχειρησιακής διεργασίας για κάποια παραγγελία αγοράς.



Σχήμα 4. Παράδειγμα ροής επιχειρησιακής διαδικασίας για παραγγελία αγοράς

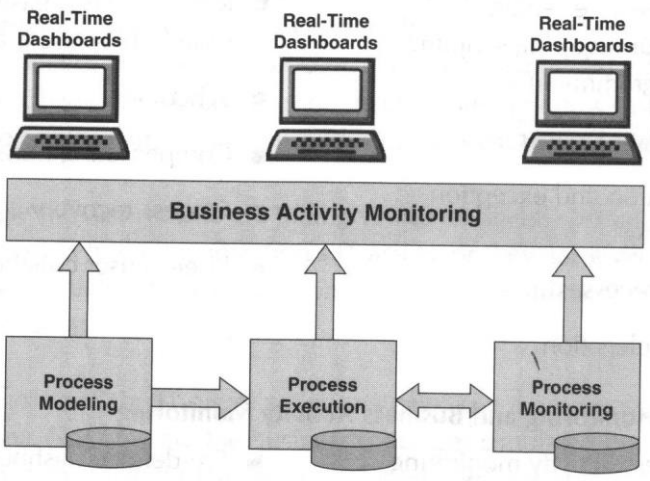
Διαχείριση επιχειρησιακών διαδικασιών (Business Process Management - BPM)[11] *ονομάζεται ένα σύνολο συστημάτων λογισμικού, εργαλείων και μεθοδολογιών που διευθετούν το πώς οι οργανισμοί ταυτοποιούν, μοντελοποιούν, αναπτύσσουν και διαχειρίζονται τέτοιες επιχειρησιακές διαδικασίες.*

Μια επιχειρησιακή διαδικασία μπορεί να περιλαμβάνει τόσο συστήματα ΤΠ όσο και ανθρώπινη διάδραση (human interaction). Διάφορες BPM λύσεις χρησιμοποιούνται εδώ και αρκετό καιρό ξεκινώντας με τα πρώιμα συστήματα ροής εργασίας (workflow systems) και φτάνοντας μέχρι τη μοντέρνα ενορχήστρωση υπηρεσιών Ιστού. Ένας σημαντικός λόγος επένδυσης στη SOA αρχιτεκτονική με Υπηρεσίες Ιστού είναι ότι οι τεχνικές

[11] http://www.bpm.com/

διαχείρισης επιχειρησιακών διαδικασιών εκμεταλλεύονται τη βάση της προσανατολισμένης σε υπηρεσίες αρχιτεκτονικής για την καλύτερη αυτοματοποίηση της επιχειρησιακής διαδικασίας.
Τα συστήματα διαχείρισης επιχειρησιακών διαδικασιών σχεδιάστηκαν για να συμβάλλουν στην ευθυγράμμιση (align) των επιχειρησιακών διαδικασιών με τα επιθυμητά αποτελέσματα και να διασφαλίσουν ότι τα συστήματα ΤΠ θα υποστηρίζουν αυτές τις διαδικασίες. Κάθε σύστημα ΤΠ υποστηρίζει και υλοποιεί επιχειρησιακές διεργασίες με κάποια μορφή. Αυτό που καθιστά μοναδική τη BPM είναι ο ρητός διαχωρισμός της επιχειρησιακής λογικής (business logic) από τον υπόλοιπο κώδικα της εφαρμογής . Η BPM απλοποιεί το πρόβλημα της ενορχηστρωμένης εκτέλεσης πολλαπλών υπηρεσιών Ιστού για την επίλυση ενός συγκεκριμένου επιχειρησιακού προβλήματος. Μπορούμε να θεωρήσουμε το επίπεδο διαχείρισης επιχειρησιακών διεργασιών ως σημείο σύνδεσης πολλαπλών υπηρεσιών σε μια ροή εκτέλεσης για να ολοκληρωθεί κάποια λειτουργία. Βγάζοντας τη ροή της διαδικασίας εκτός του κώδικα σε επίπεδο εφαρμογής (application layer) η επιχειρησιακή διαδικασία θα μπορεί να αλλάζει ευκολότερα και να ενημερώνεται με νέα γνωρίσματα και συναρτήσεις (reusability and refactoring).
Οι ροές διαδικασίας τυπικά συνίστανται σε ατομικές εργασίες που καλούν περισσότερες από μία υπηρεσίες ιστού. Επίσης, οι ροές περιλαμβάνουν ελέγχους που διακλαδώνουν σε διαφορετικά μονοπάτια εκτέλεσης (execution paths) τη ροή με βάση τα αποτελέσματα της εκτέλεσης κάποιας εργασίας. Επίσης κάθε «μονοπάτι» εκτέλεσης, στα οποία μοιράζεται η ροή, μπορεί να χειριστεί σφάλματα (workflow fault handling).

6.2 Συστήματα διαχείρισης επιχειρησιακών διαδικασιών (Business Process Management Systems – BPMS)



Εικόνα 1. Σύστημα διαχείρισης επιχειρησιακών διεργασιών

Ενώ η BPM σχετίζεται με

- τον καθορισμό,
- τη διαχείριση και
- την εκτέλεση επιχειρησιακών διεργασιών ως εταιρικό κεφάλαιο (asset)

τα συστήματα διαχείρισης επιχειρησιακών διαδικασιών (BPMS) παρέχουν την *τεχνολογία* που υλοποιεί μια ή περισσότερες από αυτές τις βασικές BPM λειτουργίες.
Τα περισσότερα συστήματα διαχείρισης επιχειρησιακών διαδικασιών εκφράζονται συνήθως ως

- ροή εργασίας (workflow),
- αυτοματοποίηση διαδικασίας,

- ολοκλήρωση διαδικασίας,
- B2B – Business2Business,
- σύνθεση υπηρεσιών,
- ενορχήστρωση
- χορογραφία

Operation Modeling
Σε όλες τις παραπάνω μορφές παρέχεται ένα εργαλείο μοντελοποίησης διαδικασίας που επιτρέπει σε αυτές να ορίζονται ως γράφοι, με τους κόμβους να αναπαριστούν τις εργασίες που πρέπει να γίνουν και τις ακμές του γράφου να αναπαριστούν εξαρτήσεις ροών ελέγχου (control flow) ή ροών δεδομένων (data flow) ανάμεσα στις εργασίες. Οι ακμές του γράφου πολλές φορές ακολουθούνται και από κανόνες που ορίζουν μεταξύ άλλων

- προϋποθέσεις,
- λογική δρομολόγησης (routing logic),
- χρονικές καθυστερήσεις και
- προθεσμίες

Οι παρακάτω λίστες δείχνουν το ευρύ φάσμα λειτουργιών που πρέπει να κατέχει ένα πλήρες σύστημα BPM.

6.2.1 Μοντελοποίηση διαδικασίας

- Σχεδιαστής επιχειρησιακής διαδικασίας
- Σχεδιαστής τεχνικής ροής
- Σχεδίαση και παρακολούθηση επιχειρησιακών μετρήσεων
- Συνεργατικός σχεδιασμός
- Σχεδίαση και υποστήριξη ευέλικτων φορμών
- Τεκμηρίωση διαδικασιών
- Επιχειρησιακός αναλυτής και δημιουργός αναφορών (business analyzer and report generator)
- Διαχείριση χρήστη / ρόλου (user/role administration)
- Διαχείριση πολιτικής ασφαλείας

6.2.2 Εκτέλεση Διαδικασίας

- Μηχανή Διαδικασίας
- Μηχανή επιχειρησιακών κανόνων (χωρίς απαίτηση προγραμματισμού)
- Διαχειριστής λιστών εργασίας (worklist manager)
- Κλιμάκωση και χειρισμός εξαιρέσεων
- Υποδιαδικασίες
- Ουρές και ομάδες
- Δρομολόγηση – Βασισμένη στο χρήστη, στους ρόλους και ad hoc
- Χρονοπρογραμματιστής (Scheduler)
- Αποζημίωση συναλλαγής (Compensating transactions)
- Επαναφορά διαδικασίας
- Συνεργασία χρήστη προς χρήστη

6.2.3 Παρακολούθηση διαδικασίας και επιχειρησιακής δραστηριότητας (Process

Monitoring and Business Activity monitoring)

- Παρακολούθηση *επιχειρησιακής δραστηριότητας και διαχείριση γεγονότων (event management)*
- Παρακολούθηση και διαχείριση επιχειρησιακών διαδικασιών
- Λειτουργία ελέγχου και καταγραφής λαθών

6.2.4 Υποδομή

- Ολοκλήρωση με εταιρικά συστήματα διαχείρισης
- Ολοκλήρωση υπηρεσιών ιστού (web services integration)
- Επεξεργασία συναλλαγών
- Αρχιτεκτονική Βασισμένη στον Ιστό (web-based)
- Ανακατεύθυνση / χρήση εφεδρικού συστήματος (failover)
- Κατανεμημένη διαχείριση χρήστη
- Ολοκλήρωση με εταιρικά συστήματα ασφάλειας
- Ολοκλήρωση επιχειρησιακών εφαρμογών (Enterprise Application Integration) και πράκτορες εφαρμογών (application agents)
- Συνδεσιμότητα βάσης δεδομένων
- Επεκτασιμότητα
- Εξισορρόπηση φορτίου
- Οργανωτικό διάγραμμα και ολοκλήρωση καταλόγου (organization chart and directory integration)

Operation Model

Ο στόχος της μοντελοποίησης επιχειρησιακών διαδικασιών είναι να αναγνωριστούν οι απαιτήσεις στο αρχικό στάδιο του σχεδιασμού και στη συνέχεια να είναι διαθέσιμες στα υπόλοιπα στάδια ανάπτυξής τους. Το μοντέλο διαδικασίας ορίζει

- τις εξαρτήσεις ανάμεσα στις εργασίες,
- την ακολουθία εκτέλεσης των εργασιών,
- το πότε και πώς θα ενεργοποιηθούν οι εργασίες και
- ποιος μπορεί να τις πραγματοποιήσει και

άλλους συγκεκριμένους επιχειρησιακούς κανόνες για τη διαδικασία . Επίσης, τα μοντέλα διαδικασιών μπορεί να χρησιμοποιηθούν από τους αναλυτές της επιχείρησης για να τρέξουν προσομοιώσεις π.χ. για να εντοπίσουν *πιθανά σημεία συμφόρησης* (bottlenecks) στις διαδικασίες.

Deployment and Process Execution

Μετά τη μοντελοποίηση, την προσομοίωση και την αντιστοίχιση της επιχειρησιακής διαδικασίας στους ήδη υπάρχοντες, αλλά και στους νέους πόρους, αυτή είναι έτοιμη να εγκατασταθεί προς λειτουργία (deploy). Τα πακέτα/σουίτες διαχείρισης επιχειρησιακών διαδικασιών περιλαμβάνουν μηχανές εκτέλεσης διαδικασιών (process execution engines) που εισάγουν τα μοντέλα διαδικασιών (που ορίζονται τυπικά με WS-BPEL) και στη συνέχεια εκτελούν και διαχειρίζονται όσα στιγμιότυπα (process instances) επιχειρησιακών διεργασιών είναι αναγκαίο για να υποστηρίξουν τις λειτουργικές απαιτήσεις του οργανισμού.

Η μηχανή εκτέλεσης διαδικασιών είναι υπεύθυνη για την εκτέλεση των μοντέλων διαδικασιών και για τη διασφάλιση των επιχειρησιακών κανόνων που σχετίζονται με τη διαδικασία π.χ. για την κλήση και εκτέλεση των εργασιών με τη σωστή σειρά ή για την ανίχνευση της τρέχουσας κατάστασης της διαδικασίας και την προσπέλαση τοπικών ή απομακρυσμένων συστημάτων ΤΠ για την ανάκτηση πληροφοριών που χρησιμοποιούνται από την διαδικασία.

Οι σουίτες BPM περιλαμβάνουν εργαλεία εποπτείας διεργασιών (monitoring) που επιτρέπουν στους χρήστες και τους διαχειριστές ΤΠ να παρακολουθούν και να ελέγχουν τη

διαδικασία. Τα εργαλεία αυτά παρέχουν διάφορες δυνατότητες όπως ιστορικό ολοκληρωμένων και εκτελούμενων διαδικασιών (summary of all completed and executing processes), επίβλεψη της κατάστασης μιας διαδικασίας (viewing the status of a process), αλλαγή προτεραιότητας διαδικασιών (altering the priority of processes), αναστολή και συνέχιση διαδικασιών (suspending and resuming processes) και εκχώρηση εκ νέου των διαδικασιών (re-assigning processes)

BAM

Η παρακολούθηση επιχειρηματικής δραστηριότητας (Business Activity Monitoring - BAM) αναλύει γεγονότα που δημιουργούνται από επιχειρησιακές διεργασίες και πληροφορίες που συλλέγονται για αυτές για να παρέχει ανατροφοδότηση (feedback) σε πραγματικό χρόνο στις υψηλότερου επιπέδου συναρτήσεις και επιχειρησιακές μετρήσεις απόδοσης.

6.3 Γλώσσα εκτέλεσης επιχειρησιακών διεργασιών (Business Process Execution Language – BPEL)

Ένας από τους πιο σημαντικούς στόχους της προσανατολισμένης σε υπηρεσίες αρχιτεκτονικής (SOA) είναι να παρέχει από άκρο εις άκρον αυτοματοποίηση των επιχειρησιακών διεργασιών. Για να επιτευχθεί η αυτοματοποίηση εισήχθηκε μια μετα-γλώσσα η οποία είναι η γλώσσα εκτέλεσης επιχειρησιακών διεργασιών Business Process Execution Language (BPEL).

Η BPEL σχεδιάστηκε για να εκτελεί επιχειρησιακές διεργασίες χρησιμοποιώντας τον εξυπηρετητή διεργασιών (process server). Η BPEL υπόσχεται να παρέχει ένα περιβάλλον για την εύκολη και αποδοτική ανάπτυξη των επιχειρησιακών διεργασιών καθώς και την άμεση εκτέλεσή τους, την παρακολούθηση και την γρήγορη προσαρμογή στις διαρκώς μεταβαλλόμενες ανάγκες των εταιρειών.

Οι υπηρεσίες είναι το κύριο δομικό στοιχείο των διεργασιών BPEL. Είναι τα εκτελέσιμα συστατικά μέρη και ονομάζονται επιχειρησιακές υπηρεσίες (business services). Οι λειτουργίες στις επιχειρησιακές υπηρεσίες συνήθως αναπαριστούν *διακριτές επιχειρησιακές δραστηριότητες*. Αυτές χρησιμοποιούνται σε επιχειρησιακές διεργασίες, πιο συγκεκριμένα, η BPEL χρησιμοποιεί επιχειρησιακές υπηρεσίες για να εκτελέσει επιχειρησιακές δραστηριότητες.

"*Δηλαδή οι επιχειρησιακές υπηρεσίες παρέχουν τη λειτουργικότητα (functionality) ενώ οι διεργασίες BPEL περιέχουν την ροή της διαδικασίας* (process flow)."

Η BPEL παρέχει:

- Υποστήριξη σχέσεων και αλληλεπιδράσεων υπηρεσιών ιστού που ασχολούνται τόσο με βραχυπρόθεσμες όσο και με μακροπρόθεσμες επιχειρηματικές συναλλαγές. Η BPEL παρέχει τα θεμέλια για την αυτοματοποίηση των επιχειρηματικών διαδικασιών.
- Συσχέτιση ανταλλαγής μηνυμάτων για ανταλλαγές μηνυμάτων μεγάλης διάρκειας, όχι περισσότερο από ένα λεπτό ή δύο, αλλά σε ημέρες, εβδομάδες ή μήνες. Η BPEL παρέχει υποστήριξη σε βιομηχανικά πρότυπα για διαδικασίες που απαιτούν πολύ μεγάλες χρονικές περιόδους για να ολοκληρωθούν.
- Εφαρμογή για την παράλληλη επεξεργασία των δραστηριοτήτων, η οποία επιτρέπει ταυτόχρονα την εκτέλεση μη εξαρτώμενων ενεργειών για τη βελτίωση της απόδοσης της διαδικασίας.
- Για τη χαρτογράφηση δεδομένων μεταξύ των αλληλεπιδράσεων εταίρων, έτσι είναι δυνατόν, για παράδειγμα, να λάβουμε το αποτέλεσμα από μια υπηρεσία ιστού και να το χρησιμοποιήσουμε για να καλέσουμε άλλη υπηρεσία ιστού.

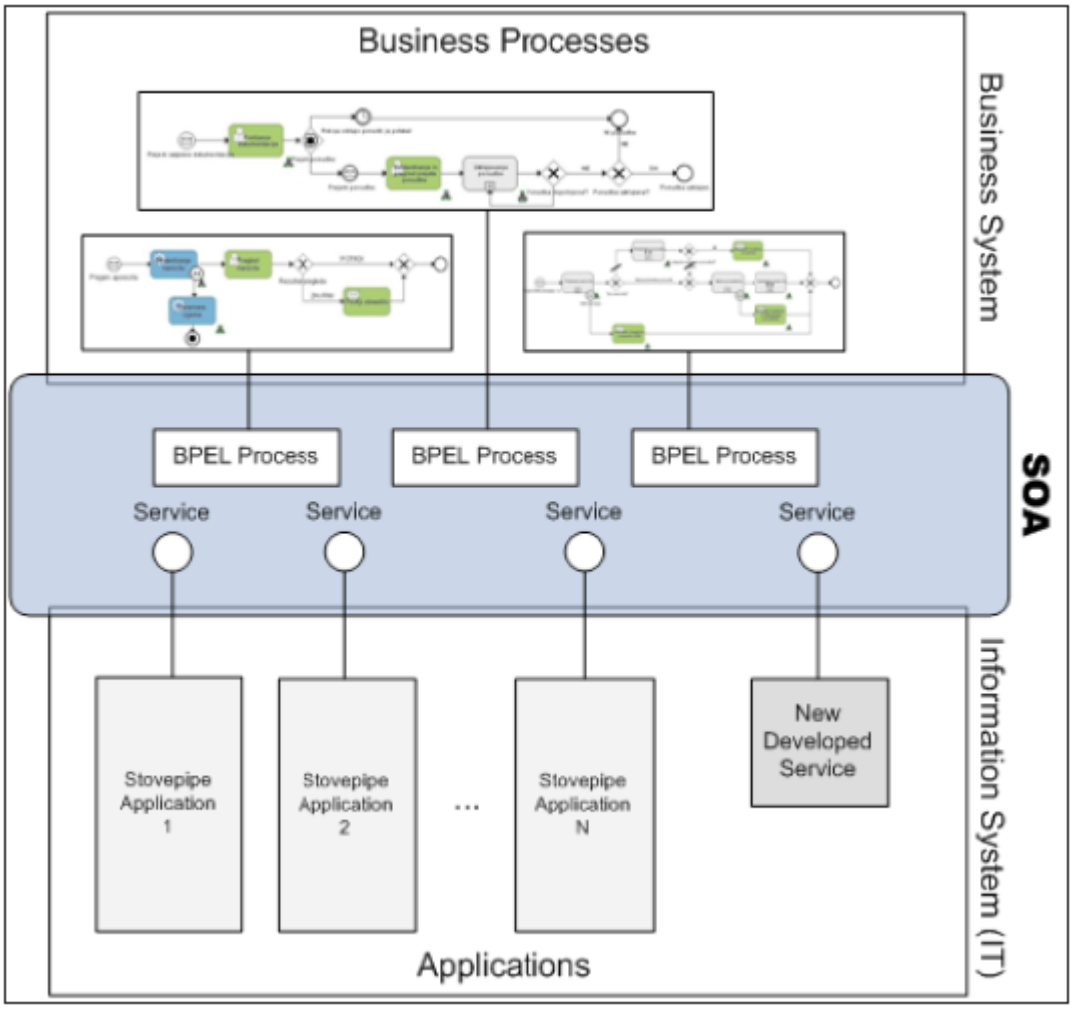
Το πρότυπο BPEL παρέχει συνεπή χειρισμό εξαίρεσης και ανάκτησης για αναπτυσσόμενες επιχειρηματικές διαδικασίες.

6.3.1 BPEL για αυτοματοποίηση της διαδικασίας

Οι υπηρεσίες στη SOA αρχιτεκτονική συντίθενται σε συναθροιστικές υπηρεσίες (aggregate services) μέχρι αυτές να υποστηρίζουν ολόκληρες επιχειρησιακές διεργασίες. Για αυτόν το λόγο, οι επιχειρησιακές διεργασίες ορίζονται ως *μια συλλογή από δραστηριότητες μέσω των οποίων καλούνται οι υπηρεσίες*. Τα συνηθισμένα είδη επιχειρησιακών διεργασιών είναι δύο:

(α) αυτές που περιέχουν υπηρεσίες μόνο μέσα από την ίδια επιχείρηση και
(β) αυτές που χρησιμοποιούν υπηρεσίες που παρέχονται από διάφορες εταιρείες.

Η γενική υιοθέτηση των λύσεων αυτοματοποίησης επιχειρησιακών διαδικασιών απαιτεί ένα πρότυπο ως βάση και μια εξειδικευμένη γλώσσα για τη σύνθεση των υπηρεσιών στις επιχειρησιακές διεργασίες, που να παρέχουν τη δυνατότητα να εκφράζονται οι επιχειρησιακές διεργασίες με τυποποιημένο τρόπο, χρησιμοποιώντας μια κοινώς αποδεκτή γλώσσα. Η BPEL είναι μια τέτοια γλώσσα και εκτιμάται ότι γρήγορα θα γίνει το κυρίαρχο πρότυπο. Ο κύριος στόχος της BPEL είναι η τυποποίηση της διαδικασίας της αυτοματοποίησης μεταξύ των υπηρεσιών ιστού.
Στο ακόλουθο σχήμα η SOA αρχιτεκτονική απεικονίζεται ως ο μεσάζοντας που ευθυγραμμίζει (aligns) τις επιχειρήσεις και τις ΤΠ καλύτερα. Η από-άκρο-εις-άκρο αυτοματοποίηση των επιχειρησιακών διεργασιών με τη BPEL καλύπτει το κενό από την πλευρά του επιχειρησιακού συστήματος, ενώ οι υπηρεσίες γεφυρώνουν το χάσμα από την πλευρά των συστημάτων ΤΠ.



Μέσα στις επιχειρήσεις, η BPEL χρησιμοποιείται για την προτυποποίηση της ολοκλήρωσης (integration) επιχειρησιακών εφαρμογών και την επέκταση της ολοκλήρωσης σε μέχρι πρότινος απομονωμένα συστήματα (isolated system). Μεταξύ διαφορετικών επιχειρήσεων, η BPEL προσφέρει ευκολότερη και αποδοτικότερη ολοκλήρωση με χρήση εταιρικών συνεργατών (partner links). Η BPEL δίνει το κίνητρο στις επιχειρήσεις να επανακαθορίσουν τις επιχειρησιακές διεργασίες τους. Αυτή η διαδικασία οδηγεί στη βελτιστοποίηση των επιχειρησιακών διεργασιών, την αναδιοργάνωση και επιλογή πιο κατάλληλων διεργασιών. Οι ορισμοί των επιχειρησιακών διεργασιών που περιγράφονται στη BPEL δεν επηρεάζουν τα υφιστάμενα συστήματα. Η BPEL είναι το κλειδί της τεχνολογίας σε περιβάλλοντα όπου οι λειτουργίες ήδη έχουν εκτεθεί, ή θα εκτεθούν, μέσω των υπηρεσιών ιστού.
Η BPEL αποτελεί σύμπτυξη δύο πρώιμων γλωσσών ροής εργασίας, της WSFL (Web Services Flow Language) της IBM η οποία βασίζεται στους κατευθυνόμενους γράφους και της XLANG της Microsoft που είναι γλώσσα δομημένη σε μπλοκ. Η BPEL συνδυάζει και τις δύο αυτές προσεγγίσεις και χρησιμοποιεί λεξιλόγιο βασισμένο σε XML για να προσδιορίσει και να περιγράψει τις επιχειρησιακές διαδικασίες. Σε ένα ορισμένο βαθμό, η BPEL είναι παρόμοια με τις παραδοσιακές γλώσσες προγραμματισμού. Προσφέρει κατασκευές, όπως βρόχους, κλάδους ανάλογους με τις εντολές επιλογής, μεταβλητές και αναθέσεις που επιτρέπουν τον καθορισμό επιχειρησιακών διεργασιών με ένα αλγοριθμικό τρόπο. Η BPEL είναι μια εξειδικευμένη γλώσσα εστιασμένη στον ορισμό και την εκτέλεση των επιχειρησιακών διεργασιών και όχι μια γλώσσα μοντελοποίησης. Ως

εκ τούτου προσφέρει τις κατασκευές που διευκολύνουν τον ορισμό των διεργασιών και παράλληλα είναι λιγότερο περίπλοκη από τις παραδοσιακές γλώσσες προγραμματισμού απλοποιώντας την εκμάθησή της.
Τα πιο σημαντικά χαρακτηριστικά της BPEL είναι αυτά που σχετίζονται με την κλήση υπηρεσιών. Η κλήση λειτουργιών είναι δυνατόν να γίνει είτε σύγχρονα είτε με ασύγχρονο τρόπο, σειριακά ή παράλληλα. Επίσης, η BPEL παρέχει λεξιλόγιο για το χειρισμό σφαλμάτων και εξαιρέσεων και υποστηρίζει μακροπρόθεσμες διαδικασίες που περιέχουν συναλλαγές και διαθέτει μηχανισμούς «αποζημίωσης» (compensation) που αφορούν την αναίρεση των ενεργειών μιας διαδικασίας όταν αυτή δεν καταφέρνει να ολοκληρωθεί επιτυχώς.

6.3.2 Ενορχήστρωση και Χορογραφία (Orchestration and Choreography)

Ανάλογα με τις απαιτήσεις η σύνθεση υπηρεσιών μπορεί να αφορά ιδιωτικές ή δημόσιες διαδικασίες, για τις οποίες χρησιμοποιούνται δύο όροι: ενορχήστρωση και χορογραφία.

"*Στην ενορχήστρωση μια κεντρική διαδικασία ελέγχει τις εμπλεκόμενες υπηρεσίες και συντονίζει την εκτέλεση διαφορετικών λειτουργιών στις υπηρεσίες που χρησιμοποιούνται στη λειτουργία*."

Οι εμπλεκόμενες υπηρεσίες δεν χρειάζεται να ξέρουν ότι είναι μέρος μιας σύνθετης επιχειρησιακής διεργασίας. Ο συντονιστής (coordinator) παρέχει ρητούς ορισμούς των λειτουργιών και της σειράς κλήσης των υπηρεσιών. Οι επιχειρησιακές διεργασίες των οποίων οι ακριβείς λεπτομέρειες προσδιορίζονται καλούνται *εκτελέσιμες επιχειρησιακές διεργασίες* και μπορούν να εκτελεστούν από μηχανές ενορχήστρωσης.
Η χορογραφία, αντιθέτως, δε βασίζεται σε έναν κεντρικό συντονιστή. Σε αυτή την περίπτωση κάθε υπηρεσία που εμπλέκεται στην χορογραφία ξέρει ακριβώς πότε να εκτελέσει τις λειτουργίες της και με ποιον να αλληλεπιδράσει. Πρόκειται για μια συνεργατική προσπάθεια που εστιάζει στην ανταλλαγή μηνυμάτων σε δημόσιες επιχειρησιακές διεργασίες. Όλοι οι συμμετέχοντες της χορογραφίας πρέπει να γνωρίζουν

- την επιχειρησιακή διεργασία,
- τις λειτουργίες προς εκτέλεση,
- τα μηνύματα που ανταλλάσσονται,
- καθώς και τη χρονική στιγμή της ανταλλαγής μηνυμάτων.

Διαδικασίες στις οποίες προσδιορίζεται μόνο η δημόσια ανταλλαγή μηνυμάτων μεταξύ των διαφόρων μερών καλούνται αφαιρετικές (abstract) επιχειρησιακές διεργασίες και ακολουθούν το παράδειγμα της χορογραφίας.
Οι εκτελέσιμες επιχειρησιακές διαδικασίες αποτελούνται από ένα σύνολο υπηρεσιών. Για να περιγραφεί μια επιχειρησιακή διεργασία σε BPEL ουσιαστικά καθορίζεται μια νέα υπηρεσία ως σύνθεση υπαρχόντων υπηρεσιών. Η διεπαφή (WSDL αρχείο) της νέας BPEL σύνθετης υπηρεσίας χρησιμοποιεί ένα σύνολο από τύπους θύρας (port types) μέσω των οποίων παρέχει λειτουργίες όπως κάθε άλλη απλή υπηρεσία. Σε ένα τυπικό σενάριο, η επιχειρησιακή διαδικασία BPEL λαμβάνει μια αίτηση. Για να την εκπληρώσει, η διαδικασία καλεί στη συνέχεια τις εμπλεκόμενες υπηρεσίες και τέλος απαντά στον αρχικό καλούντα. Επειδή η διαδικασία BPEL επικοινωνεί με άλλες υπηρεσίες, στηρίζεται σε μεγάλο βαθμό από την περιγραφή WSDL των υπηρεσιών που καλεί η σύνθετη υπηρεσία BPEL. Οι επιχειρησιακές διαδικασίες είναι ουσιαστικά γραφήματα δραστηριοτήτων, οπότε μερικές φορές είναι χρήσιμο να τις εκφράσουμε με τη χρήση σημειογραφίας μοντελοποίησης, όπως BPMN (Σημειογραφία Μοντελοποίησης Επιχειρησιακών Διεργασιών - Business Process Modeling Notation) ή UML (Ενοποιημένη Γλώσσα Μοντελοποίησης - Unified Modeling Language) διαγράμματα δραστηριότητας.
"Η BPEL δεν είναι μια γλώσσα μοντελοποίησης για διεργασίες, αλλά μια γλώσσα εκτέλεσης

διεργασιών και ενορχήστρωσης των υπηρεσιών.”

6.3.3 Βασικές έννοιες

Μια BPEL διαδικασία αποτελείται από βήματα που ονομάζονται δραστηριότητες (activity) και διακρίνονται σε βασικές και δομημένες δραστηριότητες. Οι βασικές δραστηριότητες αναπαριστούν βασικές κατασκευές και χρησιμοποιούνται για συνηθισμένες εργασίες όπως η κλήση άλλων υπηρεσιών ιστού (με χρήση του <invoke>), αναμονή για τον πελάτη να καλέσει τις επιχειρησιακές διαδικασίες στέλνοντας ένα μήνυμα (χρησιμοποιώντας το <receive>), δημιουργία απάντησης για σύγχρονες λειτουργίες (<reply>), χειρισμός μεταβλητών δεδομένων (<assign>), υπόδειξη σφαλμάτων και εξαιρέσεων (<throw>) και τερματισμός ολόκληρης της διαδικασίας (<wait>).
Είναι δυνατό να συνδυαστούν αυτές και άλλες βασικές δραστηριότητες και να οριστούν πολύπλοκες ροές που καθορίζουν ακριβώς τα βήματα μιας επιχειρησιακής διεργασίας. Για να συνδυαστούν οι βασικές δραστηριότητες, η BPEL υποστηρίζει διάφορες δομημένες δραστηριότητες. Οι πιο σημαντικές είναι:

- Η ακολουθία (<sequence>) για τον ορισμό του συνόλου των δραστηριοτήτων που θα κληθούν σειριακά.
- Ροή (<flow>) για τον ορισμό του συνόλου των δραστηριοτήτων που θα κληθούν παράλληλα.
- Δομές συνθήκης (<if>) για την υλοποίηση διακλαδώσεων.
- Οι <while>, <repeatUntil> και <forEach> για τον ορισμό βρόχων.
- Η δυνατότητα να επιλέγεται ένα από πολλά εναλλακτικά μονοπάτια με χρήση της <pick>.

Κάθε BPEL διεργασία επίσης θα ορίζει συνδέσμους συνεργατών (partner links) με χρήση της <partnerLinks> και θα δηλώνει μεταβλητές με χρήση της <variables>.

6.3.4 Κλήση υπηρεσιών

Ο ορισμός μιας BPEL διεργασίας γράφεται ως ένα έγγραφο XML χρησιμοποιώντας το στοιχείο <process>. Μέσα στο στοιχείο αυτό, μια διαδικασία BPEL συνήθως θα έχει το στοιχείο του ανώτερου επιπέδου (top-level element) <sequence>. Μέσα σ’ αυτή την ακολουθία η διαδικασία αρχικά αναμένει το εισερχόμενο μήνυμα για να ξεκινήσει. Αυτή η αναμονή μοντελοποιείται με το στοιχείο <receive>. Μετά η διαδικασία καλεί τις σχετιζόμενες υπηρεσίες με χρήση της <invoke>. Οι κλήσεις μπορούν να γίνουν σειριακά η παράλληλα με χρήση της <flow>.
Οι λειτουργίες των υπηρεσιών χωρίζονται σε δύο μεγάλες κατηγορίες οι οποίες είναι:

- Λειτουργίες υπηρεσιών σύγχρονης αίτησης/απόκρισης (synchronous invocations): Σε αυτές ο πελάτης στέλνει μια αίτηση και περιμένει απάντηση. Συνήθως η επεξεργασία τέτοιων αιτήσεων είναι σύντομη και είναι λογικό ο αποστολέας να περιμένει την απάντηση για να συνεχίσει.
- Ασύγχρονες λειτουργίες (asynchronous invocations): Συνήθως η πραγματοποιούμενη επεξεργασία στις λειτουργίες αυτές απαιτεί μεγαλύτερο χρονικό διάστημα για να ολοκληρωθεί έτσι δεν σταματούν τον αποστολέα κατά τη διάρκεια της λειτουργίας. Σε περίπτωση που απαιτείται να επιστραφεί κάποιο αποτέλεσμα στον πελάτη, οι λειτουργίες αυτές πραγματοποιούν επανακλήσεις (callbacks). Μια επανάκληση πρέπει συνήθως να σχετίζεται με την αρχική αίτηση. Αυτό ονομάζεται συσχέτιση μηνυμάτων (message correlation).

Αντίστοιχα με τις λειτουργίες και η ίδια η διαδικασία μπορεί να είναι σύγχρονη ή ασύγχρονη. Μια σύγχρονη BPEL διαδικασία επιστρέφει μια απάντηση στον πελάτη (με χρήση <reply>) αμέσως μετά την επεξεργασία και ο πελάτης δεσμεύεται για όλη τη

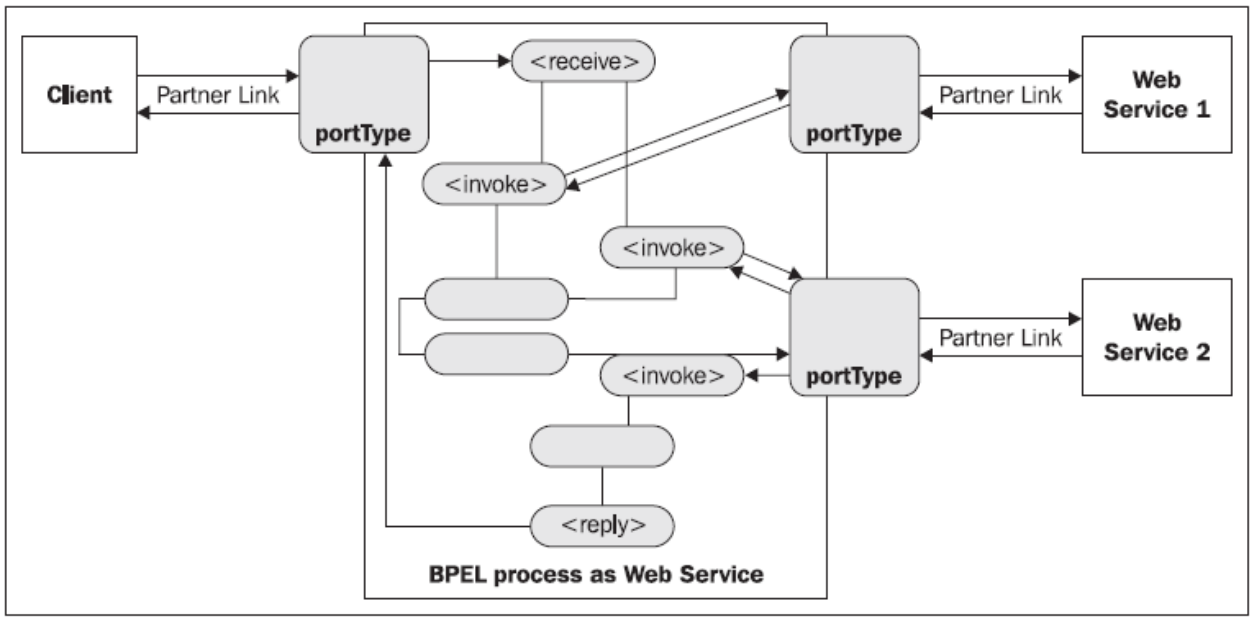
διάρκεια της εκτέλεσης της διαδικασίας BPEL.

6.3.5 Σύνδεσμοι συνεργατών (partner links)

Οι συνδέσεις προς όλα τα μέρη με τα οποία αλληλεπιδρά η BPEL ονομάζονται σύνδεσμοι συνεργατών και μπορούν να είναι σύνδεσμοι προς υπηρεσίες τις οποίες καλεί η BPEL διεργασία ή σύνδεσμοι προς πελάτες που καλούν τη διεργασία. Κάθε διεργασία BPEL διαθέτει τουλάχιστον ένα σύνδεσμο συνεργάτη-πελάτη, επειδή πρέπει να υπάρχει πάντα ένας πελάτης που καλεί τη διεργασία BPEL.
Η BPEL αντιμετωπίζει τους πελάτες ως συνδέσμους συνεργατών για δύο λόγους:

- Ο προφανής λόγος είναι η υποστήριξη των ασύγχρονων αλληλεπιδράσεων. Στην περίπτωση ασύγχρονης αλληλεπίδρασης η διεργασία χρειάζεται να καλέσει λειτουργίες στους πελάτες της.
- Ο δεύτερος λόγος βασίζεται στο γεγονός ότι η BPEL διαδικασία προσφέρει υπηρεσίες. Αυτές οι υπηρεσίες είναι διαθέσιμες μέσω τύπων θύρας και μπορούν να χρησιμοποιηθούν από περισσότερους από έναν πελάτες. Η διαδικασία θα έχει τη δυνατότητα να διακρίνει τους διαφορετικούς πελάτες και να τους προσφέρει τη λειτουργικότητα την οποία είναι εξουσιοδοτημένοι να έχουν. Για παράδειγμα, μια ασφαλιστική διαδικασία θα προσφέρει διαφορετικές λειτουργίες σε πελάτες που πραγματοποιούν ασφάλειες αυτοκινήτων και σε πελάτες που πραγματοποιούν ασφάλειες ακινήτων.

Οι συνεργάτες μπορούν να έχουν τόσο το ρόλο του πελάτη (που καλεί τη BPEL διεργασία) όσο και το ρόλο της χρησιμοποιούμενης υπηρεσίας (που καλείται από τη διεργασία).
Περιγράφοντας καταστάσεις όπου η υπηρεσία έχει κληθεί από τη διεργασία, και αντίστροφα, απαιτείται επιλογή μιας ορισμένης προοπτικής. Μπορούμε να επιλέξουμε την προοπτική της διεργασίας και να περιγράψουμε τη διεργασία ως διεργασία που απαιτεί το portType A από την υπηρεσία και που παρέχει το portType B στην υπηρεσία. Εναλλακτικά, επιλέγουμε την προοπτική της υπηρεσίας και περιγράφουμε την υπηρεσία που προσφέρει portType A στη διεργασία BPEL και απαιτεί portType B από τη διεργασία. Για να ξεπεραστεί αυτός ο περιορισμός, η BPEL εισάγει τύπους συνδέσμου συνεργάτη. Αυτοί μας επιτρέπουν να μοντελοποιήσουμε τέτοιες σχέσεις ως τρίτο μέρος. Δεν είμαστε υποχρεωμένοι να ακολουθήσουμε μια συγκεκριμένη προοπτική: αντί για αυτό, ορίζουμε μόνο ρόλους. Ένας τύπος συνδέσμου συνεργάτη πρέπει να διαθέτει τουλάχιστον ένα ρόλο και μπορεί να έχει το πολύ δύο ρόλους. Η τελευταία είναι η συνηθισμένη περίπτωση. Για κάθε ρόλο, θα πρέπει να καθορίζεται ένα portType που χρησιμοποιείται για την αλληλεπίδραση. Μερικές φορές, δεν χρειάζεται να καθοριστούν δύο ρόλοι. Τυπικό παράδειγμα είναι όταν χρησιμοποιούνται λειτουργίες σύγχρονης αίτησης-απόκρισης.
Στο παρακάτω σχήμα απεικονίζεται μια BPEL διεργασία και οι συνδέσεις της με YI, δηλαδή οι σύνδεσμοι συνεργατών (partner links).



Οι σύνδεσμοι συνεργατών είναι συγκεκριμένες αναφορές σε υπηρεσίες με τις οποίες αλληλεπιδρά μια BPEL επιχειρησιακή διεργασία. Καθορίζονται κοντά στην αρχή του εγγράφου ορισμού της BPEL διεργασίας, αμέσως μετά την ετικέτα <process>. Μέσα στο στοιχείο <partnerLinks> μπορούν να είναι εμφωλευμένοι αρκετοί ορισμοί <partnerLink>. Για κάθε σύνδεσμο συνεργάτη προσδιορίζονται τα εξής:

- Όνομα (name): Χρήση ως αναφορά για αλληλεπιδράσεις μέσα στον σύνδεσμο συνεργάτη.
- partnerLinkType: Ορίζει τον τύπο του συνδέσμου συνεργάτη.
- myRole: Ο ρόλος της ίδιας της BPEL διεργασίας.
- partnerRole: Ο ρόλος του συνεργάτη.
- initializePartnerRole: Δείχνει αν η BPEL μηχανή πρέπει να αρχικοποιήσει την τιμή του ρόλου του συνδέσμου συνεργάτη.

6.3.6 Μεταβλητές

Οι BPEL επιχειρησιακές διαδικασίες μοντελοποιούν την ανταλλαγή μηνυμάτων ανάμεσα στις εμπλεκόμενες υπηρεσίες. Μηνύματα ανταλλάσσονται όταν καλούνται λειτουργίες. Όταν η επιχειρησιακή διεργασία καλεί μια λειτουργία και λαμβάνει το αποτέλεσμα, αυτό συχνά πρέπει να αποθηκευτεί για διαδοχικές κλήσεις είτε για να χρησιμοποιηθεί εξ' ολοκλήρου είτε να εξαχθούν από αυτό συγκεκριμένα δεδομένα. Η BPEL παρέχει μεταβλητές για την αποθήκευση και διατήρηση της κατάστασης.
Οι μεταβλητές μπορούν να αποθηκεύσουν μηνύματα WSDL, στοιχεία XML Schema ή απλούς τύπους XML Schema. Κάθε μεταβλητή δηλώνεται πριν να χρησιμοποιηθεί και πρέπει να προσδιορίζεται το όνομα και ο τύπος της, δηλαδή τι στοιχεία μπορεί να αποθηκεύσει. Οι μεταβλητές μπορεί να έχουν τοπική ή καθολική εμβέλεια, δηλαδή να ισχύουν για όλη την επιχειρησιακή διεργασία.

6.3.7 Οι δραστηριότητες <invoke>, <receive> και <reply>

Και οι τρεις δραστηριότητες χρησιμοποιούν τα ίδια τρία βασικά γνωρίσματα:

- partnerLink: Καθορίζει ποιος σύνδεσμος συνεργάτη θα χρησιμοποιηθεί
- portType: Προσδιορίζει τον χρησιμοποιούμενο τύπο θύρας
- operation: Καθορίζει το όνομα της λειτουργίας που θα κληθεί, που περιμένει να καλεστεί ή που έχει ήδη κληθεί αλλά είναι σύγχρονη και απαιτεί μια απάντηση (περιπτώσεις <invoke>, <receive> και <reply> αντίστοιχα).

Η λειτουργία <invoke> υποστηρίζει δύο άλλα σημαντικά γνωρίσματα. Όταν η επιχειρησιακή διεργασία καλεί μια λειτουργία στέλνει ένα σύνολο παραμέτρων. Αυτές οι παράμετροι μοντελοποιούνται ως μηνύματα εισόδου. Για να καθοριστεί το εισερχόμενο μήνυμα για την κλήση χρησιμοποιείται το γνώρισμα inputVariable. Αν η λειτουργία που καλείται είναι σύγχρονης αίτησης/απόκρισης και επιστρέφει αποτέλεσμα, αυτό μοντελοποιείται ως εξερχόμενο μήνυμα και για να καθοριστεί αυτό χρησιμοποιείται το γνώρισμα outputVariable. Η δραστηριότητα <reply> σχετίζεται με την αρχική <receive> δραστηριότητα μέσω της οποίας ξεκίνησε η BPEL δραστηριότητα. Με χρήση της <reply> μπορεί να επιστραφεί κανονικά η απάντηση είτε ένα μήνυμα σφάλματος.

6.4 Οφελη BPEL

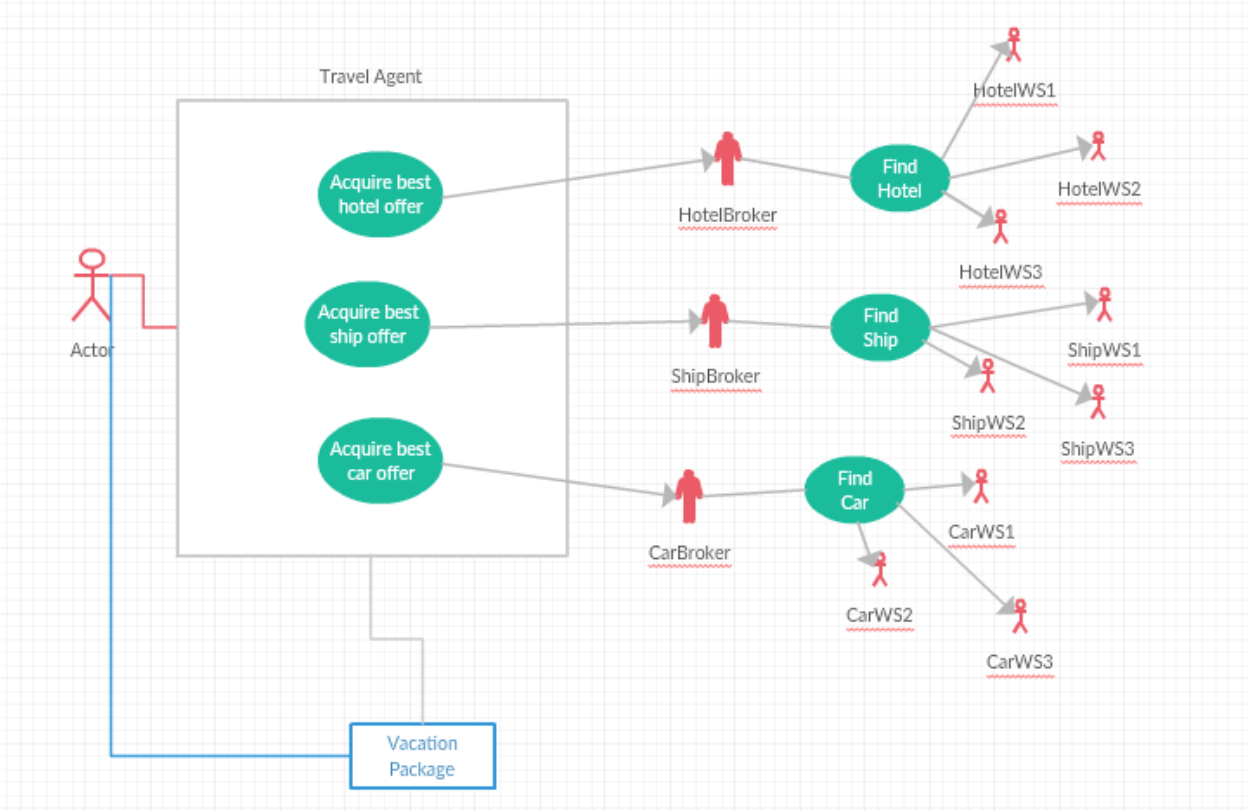
Μερικά από τα οφέλη από τη χρήση του BPEL περιλαμβάνουν:

- Το BPEL είναι SOA (Service Oriented Architecture), που σημαίνει ότι βασίζεται σε υπηρεσίες ιστού, οι οποίες αποτελούν το σύνολο πρωτοκόλλων με τα οποία οι υπηρεσίες αυτές μπορούν να δημοσιευθούν, να ανακαλυφθούν και να χρησιμοποιηθούν σε τεχνολογικά ουδέτερη, τυποποιημένη μορφή.
- Η BPEL μας επιτρέπει να αξιοποιήσουμε τα υπάρχοντα πρότυπα και σύνολα ικανοτήτων, όλα σε μια κοινή γλώσσα.
- Οι ενοποιημένες ενορχηστρώσεις της BPEL είναι οι ίδιες οι υπηρεσίες ιστού και κατά συνέπεια προσαρμόζονται φυσικά στην υπάρχουσα στοίβα Web Services.
- Το BPEL εκφράζεται εξ ολοκλήρου σε XML, χρησιμοποιεί και επεκτείνει τους ορισμούς του WSDL 1.1 και χρησιμοποιεί το XML Schema 1.0 για το μοντέλο δεδομένων.
- Η BPEL είναι πλατφόρμα και πωλητής agnostic, και έτσι μια διαδικασία BPEL θα τρέξει σε κάθε κινητήρα που είναι συμβατό με BPEL.
- Οι διαδικασίες BPEL είναι διαλειτουργικές μεταξύ των υφιστάμενων / τρέχουσων υπηρεσιών ιστού και μεταξύ αυτών, επειδή είναι οι ίδιες υπηρεσίες διαδικτύου.

7.ΜΕΘΟΔΟΛΟΓΙΑ – ΥΛΟΠΟΙΗΣΗ

7.1 Στάδια ανάπτυξης εφαρμογής

7.1.1 Ανάδειξη Προβλήματος (Problem identification)



Στα πλαίσια της παρούσας εργασίας θα δείξουμε πως με διαφορετικά σενάρια μέσω της γλώσσας BPEL, φτάνουμε στην κράτηση πακέτων διακοπών.
Μια εταιρεία (ταξιδιωτικός πράκτορας) θέλει να προσφέρει στους ανθρώπους τη δυνατότητα να κλείσουν πλήρη πακέτα διακοπών: ακτοπλοικά, ξενοδοχεία, ενοικιάσεις αυτοκινήτων.
Οι πάροχοι υπηρεσιών (ακτοπλοικές εταιρίες, εταιρείες ενοικιαζόμενων αυτοκινήτων, ξενοδοχειακές αλυσίδες κ.λπ.) παρέχουν υπηρεσίες Web για να αναζητήσουν τις προσφορές τους και να πραγματοποιήσουν κρατήσεις.

- Ταξιδιωτικός πράκτορας (travel agent): παρέχει ένα σύστημα που θα δίνει στο χρήστη τις επιλογές για τις διακοπές του. Κερδίζει χρήματα χρεώνοντας εν τέλη για κάθε πακέτο που αγοράστηκε και ικανοποιεί τους πελάτες.
- Παροχείς υπηρεσιών (Service Providers): πωλούν υπηρεσίες.
- Καταναλωτής : καλύτερος συνδυασμός υπηρεσιών και τιμών για τις ανάγκες του / της.

Σενάριο / Βήματα

Ο καταναλωτής επικοινωνεί με την υπηρεσία ταξιδιωτικών πρακτόρων για την επιλογή διαμονής του.

Η υπηρεσία ταξιδιωτικών πρακτόρων στέλνει την αίτηση.
Η απάντηση δείχνει επιτυχία με έναν αριθμό εξουσιοδότησης, υπογεγραμμένο από την αρχή πληρωμής.
Η υπηρεσία ταξιδιωτικών πρακτόρων κάνει κράτηση στο δωμάτιο του ξενοδοχείου.
Η υπηρεσία ταξιδιωτικών πρακτόρων απαιτεί μια περιγραφή του τρόπου κράτησης ενός δωματίου στην επιλεγμένη υπηρεσία ξενοδοχείου.
Η υπηρεσία ταξιδιωτικών πρακτόρων αποστέλλει την αίτηση ανάλογα, τον υπογεγραμμένο αριθμό εξουσιοδότησης από αυτήν την υπηρεσία.
Η υπηρεσία ταξιδιωτικών πρακτόρων επιβεβαιώνει την κράτηση των ακτοπλοικών:
Η υπηρεσία ταξιδιωτικών πρακτόρων αποστέλλει την αίτηση ανάλογα, τον υπογεγραμμένο αριθμό εξουσιοδότησης από αυτήν την υπηρεσία.
Η υπηρεσία ταξιδιωτικών πρακτόρων επιβεβαιώνει την κράτηση των ενοικιαζόμενων αυτοκινήτων.
Η υπηρεσία παρέχει στον χρήστη με διάφορους αριθμούς επιβεβαίωσης και επιθυμεί τον χρήστη να έχει καλές διακοπές.

Εάν δεν είναι δυνατή η επιβεβαίωση της κράτησης ξενοδοχείου και κατεπέκταση η κράτηση των ακτοπλοικών και των ενοικιαζόμενων αυτοκινήτων, αυτόματα παρουσιάζεται ένα σφάλμα.

Αυτό το σενάριο δείχνει πώς ένα πρόγραμμα, η υπηρεσία ταξιδιωτικών πρακτόρων, μπορεί να αλληλεπιδρά δυναμικά με τις υπηρεσίες ακτοπλοικών εταιρειών, τις ξενοδοχειακές υπηρεσίες και των υπηρεσιών των ενοικιαζόμενων αυτοκινήτων, χωρίς να τους γνωρίζει εκ των προτέρων ή με τον τρόπο που εργάζονται. Χάρη στις οντολογίες που χρησιμοποιούνται, το πρόγραμμα μπορεί να προσαρμοστεί στις παραλλαγές των μορφών που μια υπηρεσία μπορεί να χρησιμοποιεί και να προσαρμόζεται στην εισαγωγή νέων προϊόντων.

Το αποτέλεσμα της ανάλυσης ανέδειξε την ανάγκη δημιουργίας αυτοματοποιημένων διαδικασιών (automated procedures) ως προς τις παρακάτω ανάγκες:

- **εύρεσης ξενοδοχειακής μονάδας**
- **εύρεσης ακτοπλοικών**
- **εύρεσης ενοικιαζόμενων αυτοκινήτων**

7.2 Τεχνολογία και ανάπτυξη πλαισίου εργασίας

Η παραπάνω ανάλυση οδήγησε στην αναγκαιότητα ανάπτυξης ενός πλαισίου εργασίας, το οποίο:

- **Προσφέρει υπηρεσία εύρεσης ξενοδοχειακής μονάδας, ανάλογα με την ποιότητα και την διαθεσιμότητα.**
- **Προσφέρει υπηρεσία εύρεσης ακτοπλοικών, ανάλογα με την διαθεσιμότητα.**
- **Προσφέρει υπηρεσία εύρεσης ενοικιαζόμενων αυτοκινήτων, ανάλογα με μια επιθυμητή τιμή.**

Η τεχνολογία που υιοθετήθηκε για την υλοποίηση του παραπάνω πλαισίου εργασίας βασίζεται στην SOA αρχιτεκτονική.Η εφαρμογή που αναπτύχθηκε στα πλαίσια της πτυχιακής εργασίας πρέπει να χρησιμοποιεί μια ποικιλία από υπηρεσίες ιστού, που θα

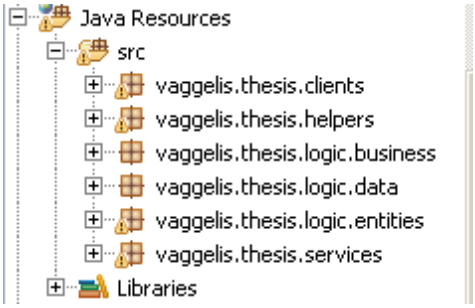
υλοποιούν τους συνεργάτες (partners) στη λύση που προτείνεται. Η εφαρμογή που αναπτύχθηκε ικανοποιεί τις αρχές επεκτασιμότητας και παραμετροποίησης, έτσι ώστε να μπορεί να παρέχει εξατομικευμένες υπηρεσίες στους χρήστες με βάση την διαθεσιμότητα , την τιμή και τον προορισμό.

7.3 Τεχνική Ανάλυση Τεχνολογίας που χρησιμοποιήθηκε

Όπως αναφέρθηκε σε προηγούμενη ενότητα μια υπηρεσία ιστού υλοποιείται ουσιαστικά ως μια Java κλάση. Αυτό που τη διαφοροποιεί είναι η χρήση ειδικής σήμανσης (annotations). Τα annotations έχουν ουσιώδη ρόλο στην JAX-WS 2.0. Καταρχάς χρησιμοποιούνται στην αντιστοίχιση της Java σε WSDL διεπαφές και Schema's. Επίσης χρησιμοποιούνται κατά το χρόνο εκτέλεσης για να ελέγξουν πώς το JAX-WS εκτελέσιμο θα επεξεργαστεί και θα απαντήσει σε κλήσεις υπηρεσιών ιστού. Τα annotations ανήκουν στην κατηγορία των μεταδεδομένων υπηρεσιών ιστού . Το πρώτο είναι το javax.jws.WebService, του οποίου ο σκοπός είναι να είτε να επισημάνει ότι μια συγκεκριμένη υλοποίηση τερματικού σημείου (endpoint reference), ουσιαστικά υλοποιεί μια υπηρεσία ιστού, είτε να δείξει ότι μια διεπαφή (interface) υπηρεσίας ορίζει μια διεπαφή υπηρεσίας ιστού. Συνεπώς, όλες οι κλάσεις που υλοποιούν υπηρεσίες ιστού στην εφαρμογή θα πρέπει να έχουν αυτό το WebService annotation. Η παράμετρος operationName είναι το όνομα της wsdl:operation που αντιστοιχεί στη μέθοδο αυτή και από προεπιλογή είναι το όνομα της Java μεθόδου.
Επίσης, τα πεδία των κλάσεων που υλοποιούν υπηρεσίες ιστού στην εφαρμογή είναι σταθερές, ίδιες σε όλες τις σχετικές κλάσεις. Οι τιμές των σταθερών αυτών αποτελούν:

- τη διεύθυνση της βάσης δεδομένων,
- το όνομα των κλάσεων του JDBC driver,
- το όνομα χρήστη της βάσης για τον οποίο γίνεται η σύνδεση και τον κωδικό του χρήστη. Τα πεδία αυτά χρησιμοποιούνται για να καταστήσουν δυνατό το χειρισμό δεδομένων στον MySQL εξυπηρετητή από το Java πρόγραμμα και για να επιτευχθεί σύνδεση με τη βάση δεδομένων και δεν αφορούν άμεσα τις υπηρεσίες ιστού.

7.3.1 <u>ΑΝΑΛΥΣΗ ΤΟΥ ΚΩΔΙΚΑ</u>



<u>Package</u>: <u>vaggelis.thesis.helpers</u>

<u>id</u>	<u>Class</u>	<u>methods</u>	<u>input</u>	<u>output</u>	<u>Description</u>

1	CheckHotelAvailability	Public: CheckAvailbleHotel()	Date dIn, DatedOut, String location	ResultSet	Η κλαση αυτη υλοποιεί τη διαθεσιμοτητα των Ξενοδοχείων μεταξυ των ημερομηνιών dIn και dOut και επιστρέφει ένα Object java.sql.ResultSet που είναι οι εγγραφές με τις λεπτομέρειες των ξενοδοχείων που ικανοποιούν τις συνθήκες. Οι εγγραφές αποτελούνται από HotelName και RoomPrice.
2	FindHotelByPrice	Public: getHotelData()	String location, int room_size, String quality, double price	ResultSet	Η κλάση αυτή υλοποιεί τη διαθεσιμότητα των ξενοδοχείων ανάλογα με μια επιθυμητή τιμή και επιστρέφει ένα Object java.sql.ResultSet που είναι οι εγγραφές με τις λεπτομέρειες των ξενοδοχείων που ικανοποιούν τις συνθήκες. Οι εγγραφές αποτελούνται από HotelName και RoomPrice.
3	CheckShip Availability	Public: CheckShip Availability()	Date DT, String Destination, double price	ResultSet	Η κλάση αυτή υλοποιεί τη διαθεσιμότητα των ακτοπλοικών ανάλογα με την DT(datetime) και επιστρέφει ένα Object java.sql.ResultSet που είναι οι εγγραφές με τις λεπτομέρειες των πλοίων αυτών που ικανοποιούν τις συνθήκες. Οι εγγραφές αποτελούνται από ShipName,ShipCompany, PortName,Capacity και BookPrice.
4	FindCarRentalByPriceAndDate	Public: CarRental()	Date dIn, Dated Out, double price, String location	ResultSet	Η κλάση αυτή υλοποιεί τη διαθεσιμότητα των ενοικιαζόμενων αυτοκινήτων μεταξυ των ημερομηνιών dIn, dOut και μιας επιθυμητής τιμής και επιστρέφει ένα Object java.sql.ResultSet που είναι οι εγγραφές με τις λεπτομέρειες των αυτοκινήτων αυτών που ικανοποιούν τις συνθήκες. Οι εγγραφές αποτελούνται από BrandName,CarColour,CarModel,Insurance και ReservationPrice.

Package: vaggelis.thesis.logic.business

id	Class	methods	input	output	Description
5	HotelList	Public: 1)getHotelL() 2)addHotelsInList()	1) - 2) hotel h	1)ArrayList <Hotel> 2) -	Η κλάση αυτή υλοποιεί ένα ArrayList από Hotels και προσθέτει ξενοδοχεία στη λίστα αυτή.
6	CarList	Public: 1)getCarL() 2)addCarInList()	1) - 2)car c	1)ArrayList <Car> 2) -	Η κλάση αυτή υλοποιεί ένα ArrayList από Cars και προσθέτει αυτοκίνητα στη λίστα αυτή.
7	ShipList	Public: 1)getShipS() 2)addShipsInList()	1) - 2)ship s	1)ArrayList <Ship> 2) -	Η κλάση αυτή υλοποιεί ένα ArrayList από Ships και προσθέτει πλοία στη λίστα αυτή.

```
/*
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "HotelList")
@XmlSeeAlso(Hotel.class)

public class HotelList implements java.io.Serializable
*/

/*
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "CarList")
@XmlSeeAlso(Car.class)

public class CarList implements java.io.Serializable
*/

/*
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "ShipList")
@XmlSeeAlso(Ship.class)

public class ShipList implements java.io.Serializable
*/
```

***Serializable σημαινει ότι η κλαση γίνεται serialize, δηλαδή μεταφράζει τις δομές δεδομένων ή την κατάσταση των αντικειμένων σε μορφή που μπορεί να μεταδοθεί σε μια σύνδεση δικτύου.

Package: vaggelis.thesis.logic.data

id	Class	methods	input	output	Description
8	DBConnectionSingleton	1)Public: DBConnectionSingleton() Public static: 2)DBConnectionSingleton getInstance() 3)getConnection()	1) - 2) - 3) -	1) - 2)DBConnectionSingleton instance 3)Connection conn	Η κλάση αυτή υλοποιεί την σύνδεσή μας στη βάση. Η δημιουργία της μας δίνει την δυνατότητα να ανοίγουμε μόνο μια φορά σύνδεση και όχι συνέχεια.
9	DBExecuteQuery	Public: 1)execute() 2) update()	1)String query 2)String query	1)ResultSet 2) -	Η κλάση αυτή υλοποιείται για να εκτελούμε τα query μας μέσω της execute για μεγαλύτερη διευκόλυνση.(Η execute περιέχει και μεθόδους από id 8)

```
/*
public DBConnectionSingleton() {
            String dbDriver = "com.mysql.jdbc.Driver";
            String url =
"jdbc:mysql://192.168.160.1:3306/thesis?autoReconnect=true&useSSL=fals
e&requireSSL=false";
            String username = "root";
            String password = "kare";
*/



/*
public ResultSet execute(String query) {
            DBConnectionSingleton.getInstance();
            Connection conn = DBConnectionSingleton.getConnection();
            ResultSet rs = null;
            try {
                  Statement stmt = conn.createStatement();
                  rs = stmt.executeQuery(query);
            } catch (Exception e) {
                  System.out.println(e);
            }
            return rs;
      }
*/
```

Package: vaggelis.thesis.logic.entities

id	Class	definitions	methods	input	output	Description
10	Hotel	1)Name 2)Hotel_value	Getters()& Setters()			Η κλάση αυτή υλοποιεί μία οντότητα με όνομα Hotel.
11	Car	1)Brand 2)Colour 3)Model 4)Insurance_Name 5)Price	Getters()& Setters()			Η κλάση αυτή υλοποιεί μία οντότητα με όνομα Car.
12	Ship	1)Name 2)Capacity 3)Company_Name 4)Port_Name 5)Price	Getters()& Setters()			Η κλάση αυτή υλοποιεί μία οντότητα με όνομα Ship.

```
/*
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "Hotel")

public class Hotel implements java.io.Serializable
*/

/*
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "Car")

public class Car implements java.io.Serializable
*/

/*
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "Ship")

public class Ship implements java.io.Serializable
*/
```

***Serializable σημαινει ότι η κλαση γίνεται serialize, δηλαδή μεταφράζει τις δομές δεδομένων ή την κατάσταση των αντικειμένων σε μορφή που μπορεί να μεταδοθεί σε μια σύνδεση δικτύου.

id	Class	methods	input	output	Description
13	HotelApp	Public: 1)AppAddHotelList() 2)returnHL()	1)Hotel h 2) -	1) - 2)HotelList	Η κλάση αυτή υλοποιείται για να εμφανίζονται τα Hotels με τα στοιχεία τους στοιβαγμένα σε λίστα. `public HotelList returnHL(){ return hList; }` Η returnHL επιστρέφει μία λίστα τύπου HotelList(id 5)
14	CarApp	Public: 1)AppAddCarList() 2)returnCL()	1)Car c 2) -	1) - 2)CarList	Η κλάση αυτή υλοποιείται για να εμφανίζονται τα Cars με τα στοιχεία τους στοιβαγμένα σε λίστα. `public CarList ResultCL(){ return cList; }` Η returnCL επιστρέφει μία λίστα τύπου CarList(id 6)
15	ShipApp	Public: 1)AppAddShipList() 2)returnSL()	1)Ship sh 2) -	1) - 2)ShipList	Η κλάση αυτή υλοποιείται για να εμφανίζονται τα Ships με τα στοιχεία τους στοιβαγμένα σε λίστα. `public ShipList ResultSL(){ return sList; }`

					H returnSL επιστρέφει μία λίστα τύπου ShipList(id 7)

Package: vaggelis.thesis.services

id	Class	methods	input	output	Description

16	HotelBroker	Public:
		1)getHotelListByPrice()
		
		2)getHotelListByDate()
			1)String location, int room_size, String quality, double price
			
			2)Date dIn, DatedOut, String location
				1)ArrayList <Hotel>
				
				2)ArrayList <Hotel>
					Στην κλάση αυτή γίνεται η υλοποίηση του Web-Service για τα hotels. Χρησιμοποιούνται μέθοδοι από τις κλάσεις με id 1,2,13.
					`/*ResultSet listOfHotels = HB.getHotelData(location,room_size,quality,price);`
					`/*ResultSet listOfHotels = CheckHAvail.CheckAvailableHotel(dIn, dOut, location);`
					`/*HL = HA.returnHL();`

17	CarBroker	getListOfCarsByPrice()	Date dIn, Dated Out, double price, String location	ArrayList <Car>	Στην κλάση αυτή γίνεται η υλοποίηση του Web-Service για τα cars. Χρησιμοποιούνται μέθοδοι από τις κλάσεις με id 4,14.
					`/*ResultSet ListOfCars = CarRent.CarRental(dIn, dOut, price, location);`
					`/*CL = CA.ResultCL();`

18	ShipBroker	getShipListByDate()	Date DT, String Destination, double price	ArrayList <Ship>	Στην κλάση αυτή γίνεται η υλοποίηση του Web-Service για τα cars. Χρησιμοποιούνται μέθοδοι από τις κλάσεις με id 3,15.

					/*ResultSet ListOfShips = CheckSAvail.CheckShipAvailability (DT, Destination,Price); /*SL= SA.ResultSL();

id	Class	methods	input	output	Description
19	HotelBrokerDelegate	Public: 1)getHotelList ByPrice() 2)getHotelList ByDate() 3)HotelBrokerDelegate()	1)String location, int room_size, String quality, double price 2)Date dIn, DatedOut, String location 3) -	1)ArrayList <Hotel> 2)ArrayList <Hotel> 3)Constructor	Η κλάση αυτή είναι το Web-Service για τα hotels εξαιτίας αυτών των annotations: @javax.jws.WebService (targetNamespace="http://services.thesis.vaggelis/", serviceName="HotelBrokerService", portName="HotelBrokerPort", wsdlLocation="WEB-INF/wsdl/HotelBrokerService.wsdl")
20	CarBroker Delegate	1)getListOfCars ByPrice() 2)CarBroker Delegate()	1)Date dIn, Dated Out, double price, String location 2) -	1)ArrayList <Car> 2)Constructor	Η κλάση αυτή είναι το Web-Service για τα cars εξαιτίας αυτών των annotations: @javax.jws.WebService (targetNamespace="http://services.thesis.vaggelis/", serviceName="CarBrokerService", portName="CarBrokerPort", wsdlLocation="WEB-INF/wsdl/CarBrokerService.wsdl")

21	ShipBrokerDelegate	1)getShipList ByDate() 2)ShipBroker Delegate()	1)Date DT, String Destination, double price 2) -	1)ArrayList <Ship> 2)Constructor	Η κλάση αυτή είναι το Web-Service για τα ships εξαιτίας αυτών των annotations: `@javax.jws.WebService (targetNamespace="http://services.thesis.vaggelis/", serviceName="ShipBrokerService", portName="ShipBrokerPort", wsdlLocation="WEB-INF/wsdl/ShipBrokerService.wsdl")`

7.3.2 ΑΝΑΛΥΣΗ BPEL PROCESS

- PartnerLink : Οι σύνδεσμοι συνεργατών ορίζονται ως ανταλλαγές επικοινωνιών μεταξύ όλων των μερών με τα οποία αλληλεπιδρά η διαδικασία BPEL. Πρόκειται για τις αναφορές στις πραγματικές εφαρμογές, μέσω των οποίων η διαδικασία BPEL αλληλεπιδρά με τον εξωτερικό κόσμο.

Ο Επεξεργαστής Ιδιότητας Link Partner μας επιτρέπει να δημιουργήσουμε συνδέσμους συνεργατών για τις διαδικασίες BPEL. Με το Εργαλείο επεξεργασίας ιδιοτήτων συνδέσμου συνεργατών, μπορείτε να ορίσετε τα ακόλουθα :

- Name - Καθορίζει το όνομα του στοιχείου Invoke.
- WSDL File - Υποδεικνύει το αρχείο WSDL που σχετίζεται με το Link Partner.
- Partner Link Type - Υποδεικνύει τον τύπο συνδέσμου συνεργατών που ορίζεται στο WSDL.
- My Role - Υποδηλώνει το ρόλο της ίδιας της επιχειρηματικής διαδικασίας.
- Partner Role - Υποδηλώνει το ρόλο του εταίρου.
- Documentation - Έχετε πρόσβαση στο παράθυρο Ιδιότητες.

Οι σύνδεσμοι συνεργατών καθορίζονται στο αρχείο .bpel.

Ένα BPEL μπορεί να αλληλεπιδράσει με τις υπηρεσίες με τους ακόλουθους τρεις τρόπους :

- Υπηρεσίες που επικαλούνται μια διαδικασία BPEL
- Υπηρεσίες που χρησιμοποιούνται από τη διαδικασία BPEL
- Υπηρεσίες που ενεργούν με δύο τρόπους

```
/*
      <bpws:partnerLinks>
    <bpws:partnerLink myRole="Interface"
name="BookServiceInterface"
partnerLinkType="ns:BookServiceInterfacePLT" wpc:id="2"/>
    <bpws:partnerLink name="CarBrokerServiceDelegatePartner"
partnerLinkType="ns:PartnerPLT" partnerRole="Reference"
wpc:id="79"/>
    <bpws:partnerLink name="HotelBrokerServiceDelegatePartner"
partnerLinkType="ns:HotelBrokerSeriviceDelegatePartnerPLT"
partnerRole="Reference" wpc:id="80"/>
    <bpws:partnerLink name="ShipBrokerServiceDelegatePartner"
partnerLinkType="ns:ShipBrokerDelegatePartnerPLT"
partnerRole="Reference" wpc:id="81"/>
  </bpws:partnerLinks>
```

- Variables: Οι μεταβλητές στον προγραμματισμό BPEL λειτουργούν όπως και σε άλλες γλώσσες προγραμματισμού: διατηρούν προσωρινές τιμές, αποτελούν μέρη εκφράσεων ή μεταβιβάζονται ως παράμετροι σε εξωτερικούς συνεργάτες. Κανονικά, χρειάζεστε μια μεταβλητή για κάθε μήνυμα που αποστέλλεται ή λαμβάνεται από μια υπηρεσία συνεργατών.Οι μεταβλητές που ορίζονται στη ρίζα διεργασίας είναι παγκόσμιες μεταβλητές, οι οποίες έχουν παγκόσμια προβολή σε όλη τη διαδικασία. Οι μεταβλητές που ορίζονται σε ένα συγκεκριμένο Πεδίο εφαρμογής είναι ορατές μόνο μέσα σε αυτό το πεδίο εφαρμογής και σε όλα τα ένθετα πεδία. Αυτές οι μεταβλητές ονομάζονται τοπικές μεταβλητές. Μια μεταβλητή που ορίζεται για ένα εσωτερικό στοιχείο Scope μπορεί να κρύψει μια ανώτερη καθορισμένη μεταβλητή με το ίδιο όνομα.

Το όνομα μιας μεταβλητής πρέπει να είναι μοναδικό μεταξύ των ονομάτων όλων των μεταβλητών που ορίζονται στο ίδιο πεδίο εφαρμογής.

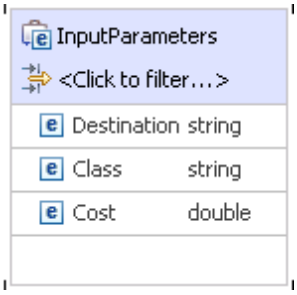
```
<bpws:variables>
    <bpws:variable element="ns0:InputParameters"
name="InputParameters" wpc:id="5"/>
    <bpws:variable name="InputParams" type="ns0:InputItem"
wpc:id="16"/>
    <bpws:variable name="InputtoHotel"
type="ns2:getHotelListByPrice" wpc:id="17"/>
    <bpws:variable name="OutputfromHotel"
type="ns2:getHotelListByPriceResponse" wpc:id="18"/>
    <bpws:variable name="TotalOutput"
type="ns0:TravelScenarioData" wpc:id="33"/>
    <bpws:variable name="InputtoCar"
type="ns2:getListOfCarsByPrice" wpc:id="41"/>
    <bpws:variable name="OutputfromCar"
type="ns2:getListOfCarsByPriceResponse" wpc:id="42"/>
    <bpws:variable name="InputtoShip"
type="ns2:getShipListByDate" wpc:id="44"/>
    <bpws:variable name="OutputfromShip"
type="ns2:getShipListByDateResponse" wpc:id="45"/>
  </bpws:variables>
```

- Receive Activity : Αυτή η δραστηριότητα καθορίζει τον σύνδεσμο συνεργάτη από τον οποίο λαμβάνεται η πληροφορία, τον τύπο θύρας και τη λειτουργία για την κλήση του συνδέσμου συνεργατών. Αυτή η δραστηριότητα περιμένει ένα μήνυμα ασύγχρονης απάντησης επανάκλησης από μια υπηρεσία, όπως μια υπηρεσία εγκρίσεων αιτήσεων δανείου. Ενώ η διαδικασία BPEL περιμένει (συμπιέζεται και αποθηκεύεται) μέχρι να φτάσει το μήνυμα επανάκλησης. Τα περιεχόμενα αυτής της απόκρισης αποθηκεύονται σε μια μεταβλητή απόκρισης στη διαδικασία.

```
<bpws:receive createInstance="yes" name="Receive"
operation="getBookResults"
partnerLink="BookServiceInterface"
portType="ns1:BookServiceInterface"
wpc:displayName="Receive" wpc:id="4"
wpc:transactionalBehavior="commitAfter">

      <wpc:output>
        <wpc:parameter name="InputParameters"
variable="InputParameters"/>
      </wpc:output>
    </bpws:receive>
```

- `PartnerLink = BookServiceInterface`
- `operation = getBookResults`
- `Variable = InputParameters`



- Assign Activity : Αυτή η δραστηριότητα παρέχει μια μέθοδο για τον χειρισμό δεδομένων, όπως η αντιγραφή των περιεχομένων μιας μεταβλητής στην άλλη. Οι λειτουργίες αντιγραφής σας επιτρέπουν να μεταφέρετε πληροφορίες μεταξύ μεταβλητών, εκφράσεων, τελικών σημείων και άλλων στοιχείων.

```
<bpws:assign name="Assign" wpc:displayName="Assign"
wpc:id="15">
      <bpws:copy>
        <bpws:from variable="InputParameters">
          <bpws:query
queryLanguage="http://www.w3.org/TR/1999/REC-xpath-
19991116"><![CDATA[Destination]]></bpws:query>
        </bpws:from>
        <bpws:to variable="InputtoHotel">
          <bpws:query
queryLanguage="http://www.w3.org/TR/1999/REC-xpath-
19991116"><![CDATA[arg0]]></bpws:query>
        </bpws:to>
      </bpws:copy>
      <bpws:copy>
        <bpws:from variable="InputParameters">
          <bpws:query
queryLanguage="http://www.w3.org/TR/1999/REC-xpath-
19991116"><![CDATA[Cost]]></bpws:query>
        </bpws:from>
        <bpws:to variable="InputtoHotel">
          <bpws:query
queryLanguage="http://www.w3.org/TR/1999/REC-xpath-
19991116"><![CDATA[arg3]]></bpws:query>
        </bpws:to>
      </bpws:copy>
      <bpws:copy>
        <bpws:from variable="InputParameters">
          <bpws:query
queryLanguage="http://www.w3.org/TR/1999/REC-xpath-
19991116"><![CDATA[Class]]></bpws:query>
        </bpws:from>
        <bpws:to variable="InputtoHotel">
          <bpws:query
queryLanguage="http://www.w3.org/TR/1999/REC-xpath-
19991116"><![CDATA[arg2]]></bpws:query>
        </bpws:to>
      </bpws:copy>
      <bpws:copy>
```

Αντιγραφή δεδομένων (Destination,Cost,Class,room_size = 3) της μεταβλητής “InputParameters” στην μεταβλητή “InputtoHotel”.

Assign From		Assign To
InputParameters Destination		InputtoHotel arg0
InputParameters Cost		InputtoHotel arg3
InputParameters Class		InputtoHotel arg2
3		InputtoHotel arg1

- Invoke Activity : Αυτή η δραστηριότητα μας δίνει τη δυνατότητα να καθορίσουμε μια ενέργεια που θέλουμε να καλέσουμε για την υπηρεσία (αναγνωρισμένη από το σύνδεσμο συνεργάτη). Η λειτουργία μπορεί να είναι μονόδρομη ή απάντηση στο αίτημα σε μια θύρα που παρέχεται από την υπηρεσία. Μπορείτε επίσης να δημιουργήσετε αυτόματα μεταβλητές σε μια δραστηριότητα invoke. Μια δραστηριότητα invoke επικαλείται μια σύγχρονη υπηρεσία ιστού ή εκκινεί μια ασύγχρονη υπηρεσία ιστού.

```
<bpws:invoke name="Invoke"
operation="getHotelListByPrice"
partnerLink="HotelBrokerServiceDelegateP
artner"
portType="ns2:HotelBrokerServiceDelegate
" wpc:continueOnError="inherit"
wpc:displayName="InvokeHotelService"
wpc:id="14">
      <wpc:input>
        <wpc:parameter name="parameters"
variable="InputtoHotel"/>
      </wpc:input>
      <wpc:output>
        <wpc:parameter name="parameters"
variable="OutputfromHotel"/>
      </wpc:output>
    </bpws:invoke>
```

Καλείται το operation “getHotelListByPrice” με παραμέτρους του variable “InputtoHotel”. Τα αποτελέσματα τους καταχωρούνται μέσω της “getHotelListByPriceResponse” στο variable “OutputfromHotel” .

	Name	Type	Read from Variable
Inputs	parameters	getHotelListByPrice	InputtoHotel
	Name	Type	Store into Variable
Outputs	parameters	getHotelListByPriceResponse	OutputfromHotel

- Snippet : Μπορείτε να χρησιμοποιήσετε μεθόδους Java ™ σε snippet επεξεργασία

για να εκτελέσετε διάφορες λειτουργίες.

Οι μέθοδοι κωδικοποίησης Java χρησιμοποιούνται σε snippet επεξεργασία για την εκτέλεση ποικίλων λειτουργιών.

Όταν χρησιμοποιείτε ένα snippet σε μια διαδικασία BPEL, ο κώδικας Java κάνει ό, τι κανονικά κάνετε σε ένα επιχειρηματικό bean και χρησιμοποιώντας τους ίδιους περιορισμούς.

```
<bpws:invoke name="Snippet1" operation="null"
partnerLink="null" portType="wpc:null"
wpc:continueOnError="inherit"
wpc:displayName="Print Hotel List" wpc:id="19">
      <wpc:script>

<wpc:javaCode><![CDATA[System.out.println("After
Hotel Invocation");
/*
com.ibm.websphere.bo.BOFactory factory =
(com.ibm.websphere.bo.BOFactory) new
com.ibm.websphere.sca.ServiceManager().locateServi
ce("com/ibm/websphere/bo/BOFactory");
commonj.sdo.DataObject response =
factory.create("http://services.thesis.vaggelis/",
"getHotelListByPriceResponse");
commonj.sdo.DataObject HotelName = null;
java.util.ListIterator<commonj.sdo.DataObject>
HotelList = response.getList(0).listIterator();

while(HotelList.hasNext()){
      HotelName = HotelList.next();
      System.out.println("Hotel
Name:"+HotelName.getString("Hotel_Name"));
}
*/
```

- For-Each Activity : Αυτή η δραστηριότητα μας δίνει τη δυνατότητα να επεξεργαζόμαστε πολλαπλά σύνολα δραστηριοτήτων διαδοχικά ή παράλληλα. Η δραστηριότητα forEach εκτελεί την περιεχόμενη δραστηριότητα (παιδιού) ακριβώς N + 1 φορές, όπου N ισούται με την τελική τιμή του μετρητή μείον την τιμή μετρητή εκκίνησης που καθορίζετε στην Counter Values του παραθύρου For Each.

```
<bpws:forEach counterName="Index" name="ForEach" parallel="no"
wpc:counterVariableId="22" wpc:displayName="ForEach" wpc:id="21">
      <bpws:startCounterValue
expressionLanguage="http://www.w3.org/TR/1999/REC-xpath-
19991116"><![CDATA[1]]></bpws:startCounterValue>
      <bpws:finalCounterValue
expressionLanguage="http://www.w3.org/TR/1999/REC-xpath-
19991116"><![CDATA[count($OutputfromHotel/return)]]></bpws:finalCounter
```

- Choice : Το structure choice (HotelCheck) στο bpel process μας, ελέγχει εάν υπάρχουν διαθέσιμα hotels.

- Snippet2 : "`System.out.println("Hotel exists\n" +OutputfromHotel.getList("return").size());`"

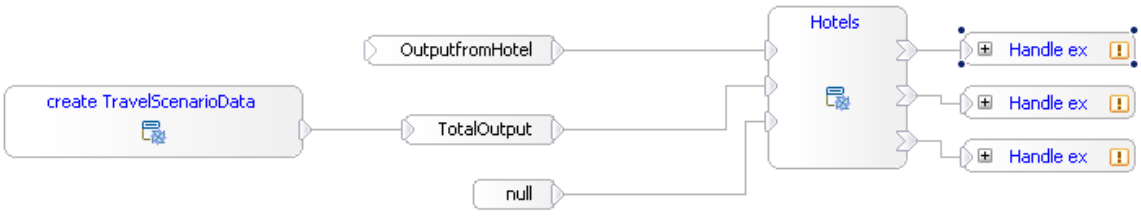
- No Hotels : "`System.out.println ("Empty Hotel List... Stay at Home.");`"

- Terminate : Μια δραστηριότητα τερματισμού μας δίνει τη δυνατότητα να τερματίσουμε τις εργασίες μιας δραστηριότητας.

➔ Εφόσον δεν υπάρχουν διαθέσιμες ξενοδοχιακές μονάδες χρησιμοποιούμε το συγκεκριμένο activity και ολοκληρώνεται η διεργασία.

➔ Εφόσον υπάρχουν διαθέσιμες ξενοδοχιακές μονάδες χρησιμοποιούμε τα Data Maps.

- Data Map : Για να μετατρέψουμε δεδομένα μεταξύ του αντικειμένου επιχειρησιακής πηγής και του αντικειμένου επιχείρησης προορισμού, χρησιμοποιούμε έναν Data Map. Υπάρχουν δύο υποστηριζόμενες χάρτες δεδομένων: XML maps and business object maps.

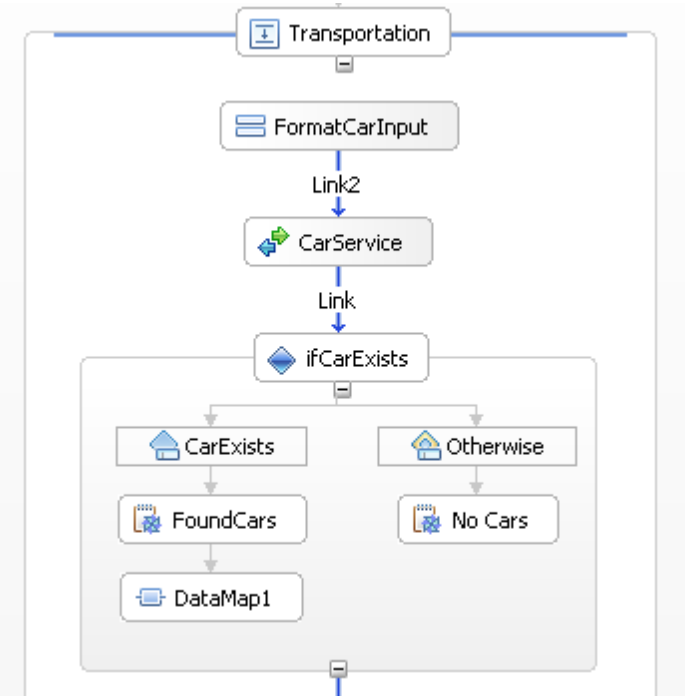


```
try {
      {// Hotels
            java.util.HashMap inputMap = new java.util.HashMap();
            java.util.HashMap outputMap = new java.util.HashMap();
            inputMap.put("getHotelListByPriceResponse",
OutputfromHotel);
            outputMap.put("TravelScenarioData", TotalOutput);
            com.ibm.wbiserver.map.MapService _serv =
                  (com.ibm.wbiserver.map.MapService)new
com.ibm.websphere.sca.ServiceManager().locateService("com/ibm/wbiserver
/map/MapService");
            _serv.transform("http://BookProcess/DataTypes", "Hotels",
inputMap, outputMap,
((com.ibm.wbiserver.relationshipservice.common.ExecutionContext)__resul
t__4));
      }
}
catch(com.ibm.wbiserver.map.exceptions.WBIMapFailureException ex){
      com.ibm.wbiserver.map.exceptions.WBIMapFailureException
__result__7;
      ex.printStackTrace();
}
catch(com.ibm.wbiserver.map.exceptions.WBIMapNotFoundException ex){
      com.ibm.wbiserver.map.exceptions.WBIMapNotFoundException
__result__10;
      ex.printStackTrace();
}
catch(com.ibm.wbiserver.map.exceptions.WBIMapServiceException ex){
      com.ibm.wbiserver.map.exceptions.WBIMapServiceException
__result__13;
      ex.printStackTrace();
}
```

Τα αποτελέσματα της “getHotelListByPriceResponse()”, που βρίσκονται στο variable “OutputfromHotel” , μετατρέπονται και περνούν στο variable “TotalOutput” μέσω της “createTravelScenarioData()”.

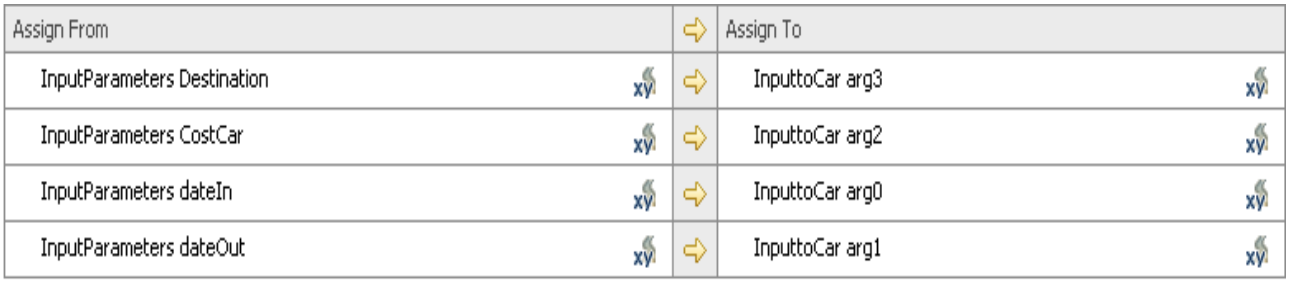
```
create TravelScenarioData
      com.ibm.websphere.bo.BOFactory factory =
         (com.ibm.websphere.bo.BOFactory) new
com.ibm.websphere.sca.ServiceManager().locateService("com/ibm/websphere
/bo/BOFactory");
        __result__2 =
factory.create("http://BookProcess/DataTypes","TravelScenarioData");
}
TotalOutput = __result__2;
java.lang.Object   result  4 = null;
```

- Parallel Activities : είναι πολύ ευέλικτες και μπορούν να προσθέσουν ένα βάθος σε μια μακροχρόνια διαδικασία BPEL. Μπορούμε να τα χρησιμοποιήσουμε για να εκτελέσουμε μερικές απλές δραστηριότητες ταυτόχρονα ή να ενσωματώσουμε ολόκληρες δευτερεύουσες διεργασίες. Η διαδικασία ολοκληρώνεται αφού κάθε μία από τις δραστηριότητες είτε εκτελεστεί είτε παραλειφθεί σε περιπτώσεις όπου η κατάσταση ενεργοποίησης εκτιμάται ως ψευδής.



```
<bpws:assign name="Assign1" wpc:displayName="FormatCarInput" wpc:id="58">
                  <bpws:sources>
                    <bpws:source linkName="Link2"/>
                  </bpws:sources>
                  <bpws:copy>
                    <bpws:from variable="InputParameters">
                      <bpws:query queryLanguage="http://www.w3.org/TR/1999/REC-
xpath-19991116"><![CDATA[Destination]]></bpws:query>
                    </bpws:from>
                    <bpws:to variable="InputtoCar">
                      <bpws:query queryLanguage="http://www.w3.org/TR/1999/REC-
xpath-19991116"><![CDATA[arg3]]></bpws:query>
                    </bpws:to>
                  </bpws:copy>
                  <bpws:copy>
                    <bpws:from variable="InputParameters">
                      <bpws:query queryLanguage="http://www.w3.org/TR/1999/REC-
xpath-19991116"><![CDATA[CostCar]]></bpws:query>
                    </bpws:from>
                    <bpws:to variable="InputtoCar">
                      <bpws:query queryLanguage="http://www.w3.org/TR/1999/REC-
xpath-19991116"><![CDATA[arg2]]></bpws:query>
                    </bpws:to>
                  </bpws:copy>
                  <bpws:copy>
                    <bpws:from variable="InputParameters">
                      <bpws:query queryLanguage="http://www.w3.org/TR/1999/REC-
xpath-19991116"><![CDATA[dateIn]]></bpws:query>
                    </bpws:from>
                    <bpws:to variable="InputtoCar">
                      <bpws:query queryLanguage="http://www.w3.org/TR/1999/REC-
xpath-19991116"><![CDATA[arg0]]></bpws:query>
                    </bpws:to>
                  </bpws:copy>
                  <bpws:copy>
                    <bpws:from variable="InputParameters">
                      <bpws:query queryLanguage="http://www.w3.org/TR/1999/REC-
xpath-19991116"><![CDATA[dateOut]]></bpws:query>
                    </bpws:from>
                    <bpws:to variable="InputtoCar">
                      <bpws:query queryLanguage="http://www.w3.org/TR/1999/REC-
xpath-19991116"><![CDATA[arg1]]></bpws:query>
                    </bpws:to>
                  </bpws:copy>
                </bpws:assign>
```

Αντιγραφή δεδομένων (Destination,CostCar,dateIn,dateOut) της μεταβλητής “InputParameters” στην μεταβλητή “InputtoCar”.



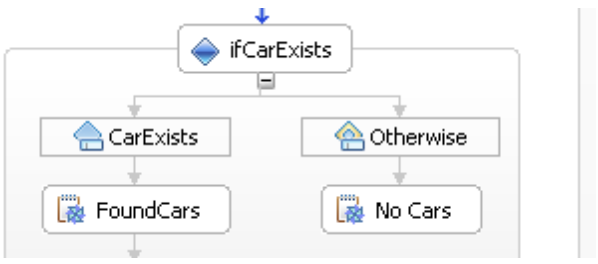

```
<bpws:flow name="ParallelActivities" wpc:displayName="Transportation"
wpc:id="38">
                <bpws:links>
                  <bpws:link name="Link1" wpc:displayName="Link1"
wpc:id="52"/>
                  <bpws:link name="Link2" wpc:displayName="Link2"
wpc:id="59"/>
                  <bpws:link name="Link4" wpc:displayName="Link4"
wpc:id="62"/>
                  <bpws:link name="Link3" wpc:displayName="Link3"
wpc:id="77"/>
                  <bpws:link name="Link" wpc:displayName="Link"
wpc:id="78"/>
                </bpws:links>
```

```
<bpws:invoke name="Invoke1" operation="getListOfCarsByPrice"
partnerLink="CarBrokerServiceDelegatePartner"
portType="ns2:CarBrokerServiceDelegate" wpc:displayName="CarService"
wpc:id="39">
                    <wpc:input>
                      <wpc:parameter name="parameters"
variable="InputtoCar"/>
                    </wpc:input>
                    <wpc:output>
                      <wpc:parameter name="parameters"
variable="OutputfromCar"/>
                    </wpc:output>
                    <bpws:targets>
                      <bpws:target linkName="Link2"/>
                    </bpws:targets>
                    <bpws:sources>
                      <bpws:source linkName="Link"/>
                    </bpws:sources>
                  </bpws:invoke>
```

Καλείται το operation “getListOfCarsByPrice” με παραμέτρους του variable “InputtoCar”. Τα αποτελέσματα τους καταχωρούνται μέσω της “getListofCarsByPriceResponse” στο variable “OutputfromCar” .

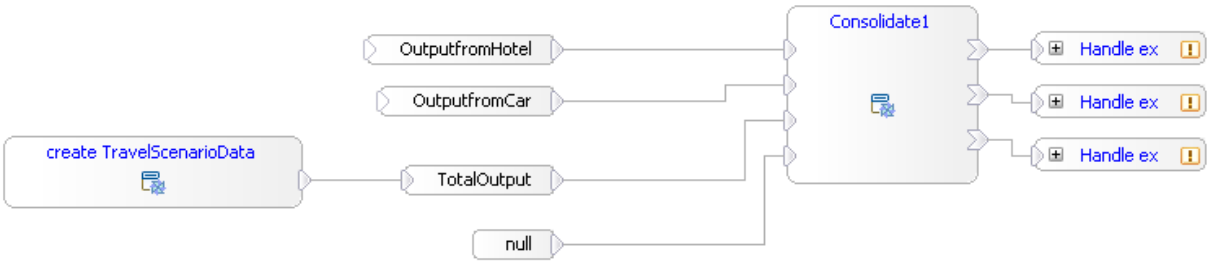
	Name	Type	Read from Variable
Inputs	parameters	getListOfCarsByPrice	InputtoCar
	Name	Type	Store into Variable
Outputs	parameters	getListOfCarsByPriceResponse	OutputfromCar

- Choice : Το structure choice (ifCarExists) στο bpel process μας, ελέγχει εάν υπάρχουν διαθέσιμα cars.



- FoundCars : “`System.out.println("Cars Found");`”
- No Cars : “`System.out.println("No Cars Found");`”

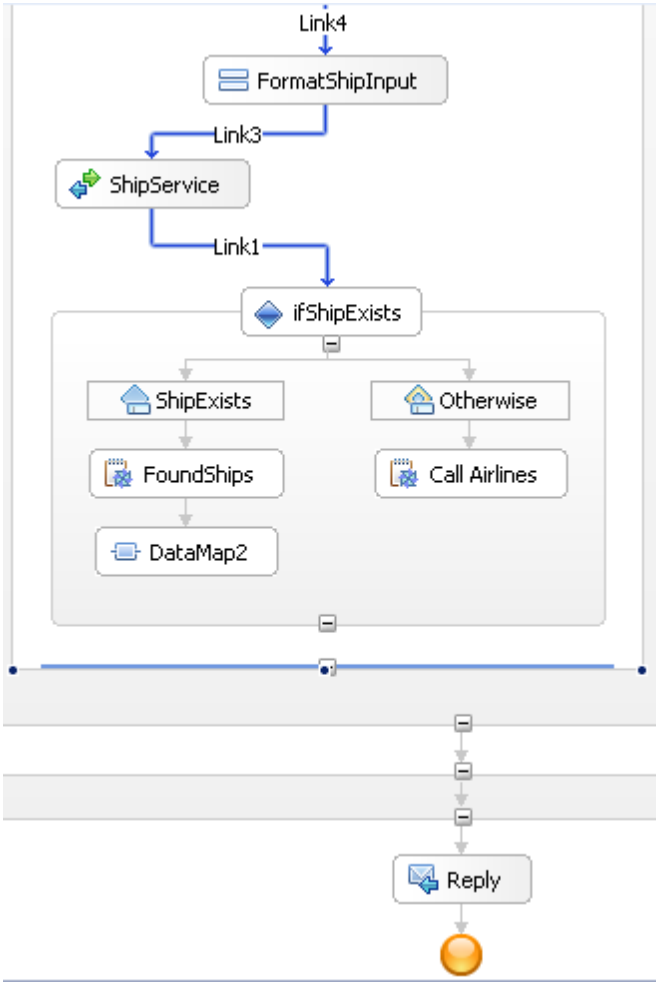
Data Map1 :




```
Consolidate1
            java.util.HashMap inputMap = new
java.util.HashMap();
            java.util.HashMap outputMap = new
java.util.HashMap();
            inputMap.put("getHotelListByPriceResponse",
OutputfromHotel);
            inputMap.put("getListOfCarsByPriceResponse",
OutputfromCar);
            outputMap.put("TravelScenarioData", TotalOutput);
            com.ibm.wbiserver.map.MapService _serv =
                  (com.ibm.wbiserver.map.MapService)new
com.ibm.websphere.sca.ServiceManager().locateService("com/ibm/
wbiserver/map/MapService");
            _serv.transform("http://BookProcess/DataTypes",
"Consolidate1", inputMap, outputMap,
((com.ibm.wbiserver.relationshipservice.common.ExecutionContex
t)__result__5));
      }
}
catch(com.ibm.wbiserver.map.exceptions.WBIMapFailureException
ex){
      com.ibm.wbiserver.map.exceptions.WBIMapFailureException
__result__8;
      ex.printStackTrace();
}
catch(com.ibm.wbiserver.map.exceptions.WBIMapNotFoundException
ex){
      com.ibm.wbiserver.map.exceptions.WBIMapNotFoundException
__result__11;
      ex.printStackTrace();
}
catch(com.ibm.wbiserver.map.exceptions.WBIMapServiceException
ex){
      com.ibm.wbiserver.map.exceptions.WBIMapServiceException
__result__14;
      ex.printStackTrace();
}
```

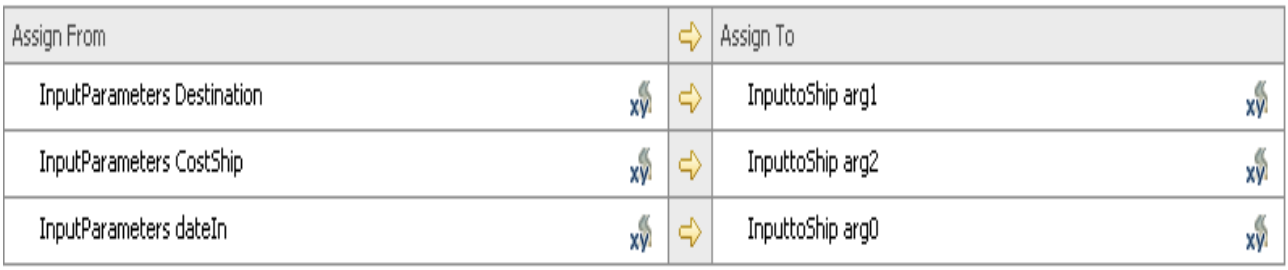
Τα αποτελέσματα της “getHotelListByPriceResponse()”, που βρίσκονται στο variable “OutputfromHotel” και Τα αποτελέσματα της “getListOfCarsByPriceResponse()”, που βρίσκονται στο variable “OutputfromCar” μετατρέπονται και περνούν στο variable “TotalOutput” μέσω της “createTravelScenarioData()”.

```
create TravelScenarioData
      com.ibm.websphere.bo.BOFactory
factory =
         (com.ibm.websphere.bo.BOFactory)
new
com.ibm.websphere.sca.ServiceManager().loc
ateService("com/ibm/websphere/bo/BOFactory
");
       __result__3 =
factory.create("http://BookProcess/DataTyp
es","TravelScenarioData");
}
TotalOutput = __result__3;
java.lang.Object   result  5 = null;
```


Link4
FormatShipInput
Link3
ShipService
Link1
ifShipExists
ShipExists
Otherwise
FoundShips
Call Airlines
DataMap2
Reply

```
<bpws:assign name="Assign2" wpc:displayName="FormatShipInput" wpc:id="60">
                  <bpws:targets>
                    <bpws:target linkName="Link4"/>
                  </bpws:targets>
                  <bpws:sources>
                    <bpws:source linkName="Link3"/>
                  </bpws:sources>
                  <bpws:copy>
                    <bpws:from variable="InputParameters">
                      <bpws:query queryLanguage="http://www.w3.org/TR/1999/REC-
xpath-19991116"><![CDATA[Destination]]></bpws:query>
                    </bpws:from>
                    <bpws:to variable="InputtoShip">
                      <bpws:query queryLanguage="http://www.w3.org/TR/1999/REC-
xpath-19991116"><![CDATA[arg1]]></bpws:query>
                    </bpws:to>
                  </bpws:copy>
                  <bpws:copy>
                    <bpws:from variable="InputParameters">
                      <bpws:query queryLanguage="http://www.w3.org/TR/1999/REC-
xpath-19991116"><![CDATA[CostShip]]></bpws:query>
                    </bpws:from>
                    <bpws:to variable="InputtoShip">
                      <bpws:query queryLanguage="http://www.w3.org/TR/1999/REC-
xpath-19991116"><![CDATA[arg2]]></bpws:query>
                    </bpws:to>
                  </bpws:copy>
                  <bpws:copy>
                    <bpws:from variable="InputParameters">
                      <bpws:query queryLanguage="http://www.w3.org/TR/1999/REC-
xpath-19991116"><![CDATA[dateIn]]></bpws:query>
                    </bpws:from>
                    <bpws:to variable="InputtoShip">
                      <bpws:query queryLanguage="http://www.w3.org/TR/1999/REC-
xpath-19991116"><![CDATA[arg0]]></bpws:query>
                    </bpws:to>
                  </bpws:copy>
                </bpws:assign>
```

Αντιγραφή δεδομένων (Destination,CostShip,dateIn) της μεταβλητής “InputParameters” στην μεταβλητή “InputtoShip”.

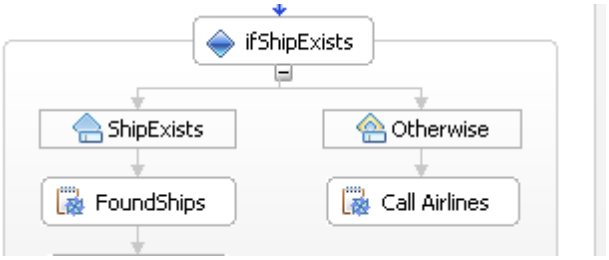



```
<bpws:invoke name="Invoke2" operation="getShipListByDate"
partnerLink="ShipBrokerServiceDelegatePartner"
portType="ns2:ShipBrokerServiceDelegate" wpc:displayName="ShipService"
wpc:id="43">
                  <wpc:input>
                    <wpc:parameter name="parameters"
variable="InputtoShip"/>
                  </wpc:input>
                  <wpc:output>
                    <wpc:parameter name="parameters"
variable="OutputfromShip"/>
                  </wpc:output>
                  <bpws:targets>
                    <bpws:target linkName="Link3"/>
                  </bpws:targets>
                  <bpws:sources>
                    <bpws:source linkName="Link1"/>
                  </bpws:sources>
                </bpws:invoke>
```

Καλείται το operation “getShipListByDate” με παραμέτρους του variable “InputtoShip”. Τα αποτελέσματα τους καταχωρούνται μέσω της “getShipListByDateResponse” στο variable “OutputfromShip” .

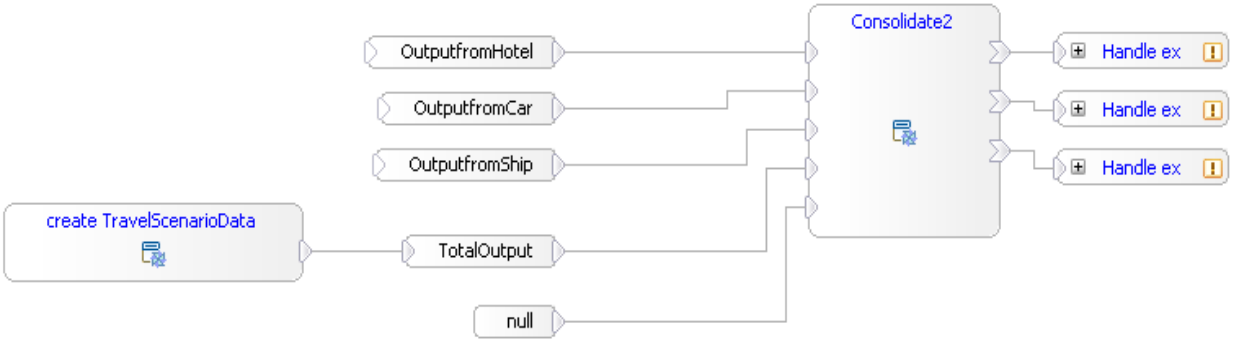
	Name	Type	Read from Variable
Inputs	parameters	getShipListByDate	InputtoShip
	Name	Type	Store into Variable
Outputs	parameters	getShipListByDateResponse	OutputfromShip

- Choice : To structure choice (ifShipExists) στο bpel process μας, ελέγχει εάν υπάρχουν διαθέσιμα ships.



- FoundShips : “`System.out.println("Ships Found");`”
- Call Airlines : “`System.out.println("No Ships Found, Check Airlines");`”

Data Map 2 :



```
try {
      {// Consolidate2
            java.util.HashMap inputMap = new java.util.HashMap();
            java.util.HashMap outputMap = new java.util.HashMap();
            inputMap.put("getHotelListByPriceResponse",OutputfromHotel);
            inputMap.put("getListOfCarsByPriceResponse", OutputfromCar);
            inputMap.put("getShipListByDateResponse", OutputfromShip);
            outputMap.put("TravelScenarioData", TotalOutput);
            com.ibm.wbiserver.map.MapService _serv =
                  (com.ibm.wbiserver.map.MapService)new
com.ibm.websphere.sca.ServiceManager().locateService("com/ibm/wbiserver/
map/MapService");
            _serv.transform("http://BookProcess/DataTypes",
"Consolidate2", inputMap, outputMap,
((com.ibm.wbiserver.relationshipservice.common.ExecutionContext)__result
__6));
      }
}
catch(com.ibm.wbiserver.map.exceptions.WBIMapFailureException ex){
      com.ibm.wbiserver.map.exceptions.WBIMapFailureException
__result__9;
      ex.printStackTrace();
}
catch(com.ibm.wbiserver.map.exceptions.WBIMapNotFoundException ex){
      com.ibm.wbiserver.map.exceptions.WBIMapNotFoundException
__result__12;
      ex.printStackTrace();
}
catch(com.ibm.wbiserver.map.exceptions.WBIMapServiceException ex){
      com.ibm.wbiserver.map.exceptions.WBIMapServiceException
__result__15;
      ex.printStackTrace();
}
```

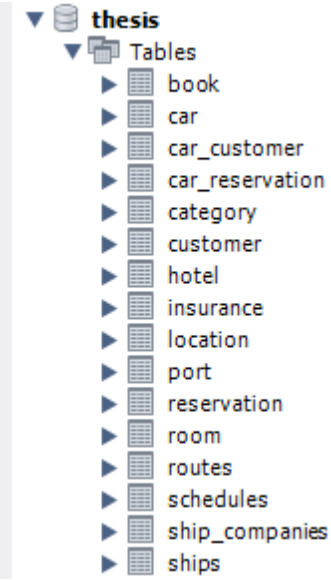
Τα αποτελέσματα της “getHotelListByPriceResponse()”, που βρίσκονται στο variable “OutputfromHotel” , τα αποτελέσματα της “getListOfCarsByPriceResponse()”, που βρίσκονται στο variable “OutputfromCar” και α αποτελέσματα της “getShipListByPriceResponse()”, που βρίσκονται στο variable “OutputfromShip” μετατρέπονται και περνούν στο variable “TotalOutput” μέσω της “createTravelScenarioData()”.

```
create TravelScenarioData
      com.ibm.websphere.bo.BOFactory factory =
         (com.ibm.websphere.bo.BOFactory) new
com.ibm.websphere.sca.ServiceManager().locateService("com/ibm
/websphere/bo/BOFactory");
       __result__4 =
factory.create("http://BookProcess/DataTypes","TravelScenario
Data");
}
TotalOutput = __result__4;
java.lang.Object   result  6 = null;
```

- Reply Activity : Αυτή η δραστηριότητα επιτρέπει στη διαδικασία να στείλει ένα μήνυμα σε απάντηση σε ένα μήνυμα που ελήφθη μέσω μιας receive activity. Ο συνδυασμός μιας receive activity και μιας reply activity αποτελεί μια λειτουργία απάντησης-αίτησης στον τύπο θύρας WSDL για τη διαδικασία.

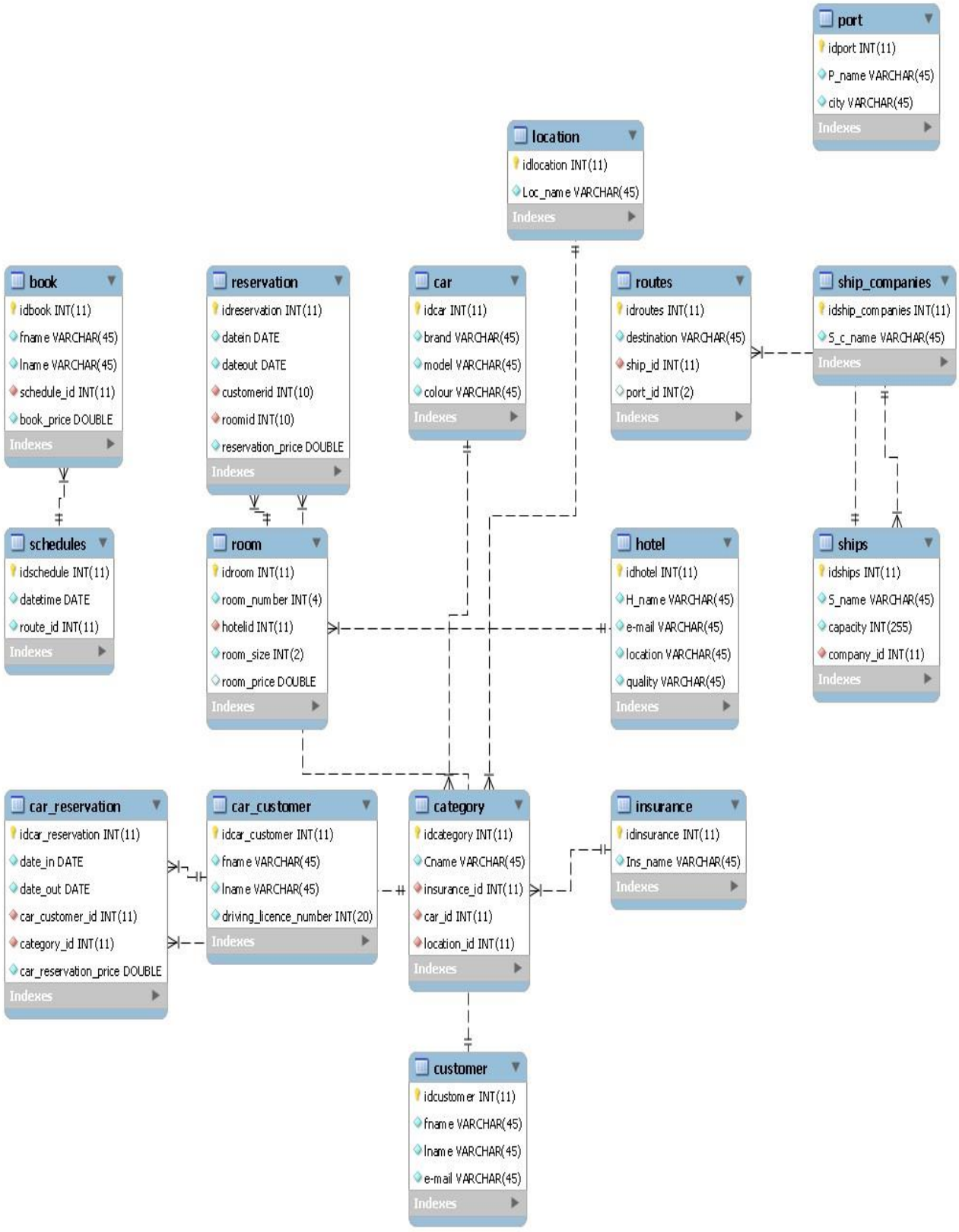

```
<bpws:reply name="Reply"
operation="getBookResults"
partnerLink="BookServiceInterface"
portType="ns1:BookServiceInterface"
wpc:displayName="Reply" wpc:id="34">
      <wpc:input>
        <wpc:parameter name="TotalBookingOutput"
variable="TotalOutput"/>
      </wpc:input>
    </bpws:reply>
```

7.3.3 ΑΝΑΛΥΣΗ ΤΗΣ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ



TABLES	COLUMNS	PRIVATE KEY	FOREIGN KEY	DESCRIPTION
book	idbook, fname, lname, schedule_ id, book_price	idbook	schedule_id	Σ αυτόν το πίνακα γινονται οι κρατήσεις για τα ακτοπλοικά εισητήρια.Στο table book περιέχονται τα προγραμματισμένα δρομολόγια, που συνδέονται με το foreign key (schedule_id).
car	idcar, brand, model, colour	idcar	-	
car _customer	idcar_customer,f name, lname, driving_licence_ number	idcar_customer	-	
car_reserva tion	idcar_reservation , date_in, date_out, car_customer_id, category_id car_reservation_ price	idcar_reservation	car_customer_id, category_id	Σ αυτόν το πίνακα γινονται οι κρατήσεις για τα ενοικιαζόμενα αυτοκίνητα .Στο table car_reservation περιέχονται οι κατηγορίες των αυτοκινήτων και οι πελάτες , που συνδέονται με τα foreign key (car_customer_id, category_id).
category	Idcategory, Cname, insurance_id, car_id, location_id	idcategory	insurance_id, car_id, location_id	Στο table category περιέχονται τα αυτοκίνητα, οι τοποθεσίες και οι ασφάλειες, που συνδέονται με τα foreign keys (insurance_id,

				car_id, location_id)
customer	idcustomer , fname, lname, e-mail	idcustomer	-	
hotel	idhotel, H_name,email, location, quality	idhotel	-	
insurance	Idinsurance, Ins_name	idinsurance	-	
location	idlocation, Loc_name	idlocation	-	
port	idport, P_name, city	idport	-	
reservation	idreservation, datein, datout, customerid, roomid, reservation _price	idreservation	customerid, roomid	Σ αυτόν το πίνακα γινονται οι κρατήσεις για τις ξενοδοχειακές μονάδες. Στο table reservation περιέχονται οι πελάτες και τα κλεισμένα δωμάτια, που συνδέονται με τα foreign keys (customerid, roomid).
room	idroom, room_number, hotelid, room_size, room_price	idroom	hotelid	Στο table room περιέχονται τα ξενοδοχεία, που συνδέονται με το foreign key (hotelid).
routes	idroutes, destination, ship_id, port_id	idroutes	ship_id, port_id	Στο table routes περιέχονται τα ονόματα των πλοίων και των λιμανιών, που συνδέονται με τα foreign keys (ship_id, port_id).
schedule	idschedule, datetime, route_id	idschedule	route_id	Στο table schedule περιέχονται τα δρομολογια, που συνδέονται με τοforeign key (route_id).
ship _companies	dship_companies , S_c_name	idship_companie s		
ships	idships, S_name, capacity, company_id	idships	company_id	Στο table ships περιέχονται οι ακτοπλοικές εταιρίες, που συνδέονται με το foreign key (company_id).



8. ΣΥΜΠΕΡΑΣΜΑΤΑ

Η αρχιτεκτονική SOA μπορεί να βοηθήσει τις επιχειρήσεις (τους οργανισμούς ή ακόμα και δημόσιες υπηρεσίες) να ανταποκριθούν πιο γρήγορα και αποδοτικά από άποψη κόστους

στις διαρκώς μεταβαλλόμενες συνθήκες της αγοράς. Συνδυάζει πολλές από τις βέλτιστες πρακτικές των παλαιότερων αρχιτεκτονικών λογισμικού και παράλληλα προωθεί την επαναχρησιμοποίηση υπηρεσιών, ενισχύει τη διαλειτουργικότητα και απλοποιεί την διασύνδεση και την χρήση των υπαρχόντων συστημάτων ΤΠ. Με την απόκρυψη των λεπτομερειών υλοποίησης των υπηρεσιών από τους πελάτες οι υπηρεσίες μπορούν να εκτελούνται σε κατανεμημένα συστήματα και να είναι προσπελάσιμες μέσω Διαδικτύου ή ιδιωτικού δικτύου (intranet).
Στην παρούσα εργασία, μετά την μελέτη των αρχών και των τεχνολογιών της SOA, παρουσιάστηκε μια εφαρμογή της με Υπηρεσίες Ιστού και ενορχήστρωση αυτών σε μια επιχειρησιακή διεργασία, προκειμένου να επιτευχθεί ο στόχος αυτοματοποιημένης παροχής εξατομικευμένων υπηρεσιών σε καταναλωτές για την λήψη πακέτων διακοπών. Εξετάστηκε επίσης κατά την εκτέλεση BPEL σεναρίων, πως είναι εύκολο με αυτόν τον τρόπο να χρησιμοποιούμε διάφορα web services χωρίς να αλλάζουμε τον BPEL κώδικα. Ένα έμμεσο πλεονέκτημα της χρήσης SOA, που διαπιστώθηκε και κατά την ανάπτυξη της εφαρμογής, είναι η ευκολία να γίνει έλεγχος για την ορθότητα των υπηρεσιών καθώς είναι αυτόνομες και προσδιορίζονται πλήρως από τις διεπαφές τους. Από την άλλη πλευρά, για πιο σύνθετες εφαρμογές είναι πιθανό να χρειαστεί ο οργανισμός να επενδύσει σε συστήματα ελέγχου προκειμένου να εξασφαλίζει την ορθή λειτουργία των υπηρεσιών. Παρά τα οφέλη της, η υιοθέτηση SOA στυλ στο σχεδιασμό και την ανάπτυξη λογισμικού θέτει κάποιες σημαντικές προκλήσεις που μπορεί να δράσουν ανασταλτικά στο να επιλεγεί η χρήση της. Μια προφανής πρόκληση που αντιμετωπίζεται αφορά τη διαχείριση μεταδεδομένων καθώς σε ένα περιβάλλον που χρησιμοποιεί SOA μια και μόνο εφαρμογή μπορεί να παράγει εκατομμύρια μηνύματα. Έτσι η πολυπλοκότητα του να καθοριστούν οι αλληλεπιδράσεις των υπηρεσιών αυξάνεται και η ανάγκη για SOA διακυβέρνηση φαίνεται επιτακτική. Επίσης, πρέπει να διευκρινιστεί ότι η SOA υπόσχεται σημαντικά οφέλη αλλά δεν μπορεί να τα εγγυηθεί: η επιτυχημένη υλοποίηση της SOA είναι αυτή που θα καθορίσει σε τι βαθμό θα επωφεληθεί ο οργανισμός από την υιοθέτηση του υπηρεσιοστρεφούς στυλ αρχιτεκτονικής.
Οι τεχνολογίες για την εφαρμογή της SOA είναι βέβαιο ότι θα εξελιχθούν για την αντιμετώπιση των αναδυόμενων αναγκών, αλλά οι έννοιες της θα παραμείνουν. Για να αντιμετωπιστούν αυτές οι ανησυχίες ότι η SOA είναι πιθανό να ξεπεράσει τα όριά της, ένα συντονισμένο ερευνητικό πρόγραμμα είναι απαραίτητο. Αν η SOA πρόκειται να χρησιμοποιηθεί με προηγμένες μεθόδους, έρευνα πρέπει να γίνει σε τομείς όπως ο σχεδιασμός για την κατανόηση πλαισίου (context awareness), τη χρηστικότητα των υπηρεσιών, την αυτοματοποιημένη διαχείριση, τη runtime παρακολούθηση και προσαρμογή, τη δυναμική ανακάλυψη και σύνθεση των υπηρεσιών, την καθοδηγούμενη από γεγονότα (event-driven) SOA και τις real time εφαρμογές.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	Ελληνικός Όρος

Scope	Εμβέλεια
Specification/Spec	Προδιαγραφή
Recommendations	Υποδείξεις
Application Server	Εξυπηρετητής Εφαρμογών
Deployment	Εγκατάσταση στον εξυπηρετητή εφαρμογών
Loose coupling	Χαλαρή συνδεσιμότητα
Integration	Ολοκλήρωση
Endpoint	Τερματικό Σημείο
Binding	Συσχέτιση
Messaging	Ανταλλαγή μηνυμάτων
Service Provider	Πάροχος υπηρεσίας
Service Requestor	Αιτούντας/πελάτης υπηρεσίας
Middleware	Ενδιάμεσο λογισμικό
Invocation	Κλήση
Annotation	Επισήμανση
Compensation	Αποζημίωση
Metadata	Μεταδεδομένα
Namespace	Πεδίο ονομάτων

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

ΤΠ	Τεχνολογία Πληροφοριών/Πληροφορικής
ΥΙ	Υπηρεσίες Ιστού
SOA	Service Oriented Architecture
BPEL	Business Process Execution Language
ISO	International Organization for Standardization
W3C	World Wide Web Consortium
SOAP	Simple Object Access Protocol
WSDL	Web Service Description Language
UDDI	Universal Description, Discovery and Integration
XML	eXtensible Markup Language
HTTP	HyperText Transfer Protocol
RPC	Remote Procedure Calls
IDE	Integrated Development Environment
JAX-WS	Java API for XML Web Services
IT	Information Technology
CICS	Customer Information Control System (IBM)
IMS	Integrated Management System
CORBA	Common Object Request Broker Architecture
J2EE	Java 2 Platform, Enterprise Edition
COM	Component Object Model
DCOM	Microsoft Distributed Component Object Model
JAXB	Java API for XML Binding
SAAJ	SOAP with Attachments for Java
JAXR	Java API for XML Registries
DOM	Document Object Model
SAX	Simple API for XML
XSLT	eXtensible Stylesheet Language Transformations
OASIS	Organization for the Advancement of Structured Information Standards
SLA	Service Level Agreement
DCE	Distributed Computing Environment
WS-CAF	WS-Composite Application Framework

JAX-RPC	Java Api for Xml-Based Remote Procedure Call
ESB	Enterprise Service Bus
POJO	Plain Old Java Object
SMTP	Simple Mail Transfer Protocol
WSFL	Web Services Flow Language
OSI	Open Systems Interconnection
URI	Uniform Resource Identifier
MTOM	Message Transmission Optimization Mechanism
ebXML	electronic business XML
Java RMI	Java Remote Method Invocation
URL	Uniform Resource Locator
IDL	Interface Definition Language
API	Application Programming Interface
SEI	Service Endpoint Interface
EJB	Enterprise Java Bean
REST	REpresentational State Transfer
WOA	Web Oriented Architecture
JSON	JavaScript Object Notation
BPMS	Business Process Management Systems
BPMN	Business Process Modeling Notation
UML	Unified Modeling Language
TCP/IP	Transfer Control Protocol/Internet Protocol
MEP	Message Exchange Patterns
JBI	Java Business Integration
QoS	Quality of Service

ΑΝΑΦΟΡΕΣ

E. Newcomer and G. Lomow, Understanding SOA with Web Services, Addison-Wesley, 2005, p. 8

E. Hewitt, Java SOA Cookbook, O'Reilly Media, 2009, p. 31

E. Hewitt, Java SOA Cookbook, O'Reilly Media, 2009, p. 35

E. Hewitt, Java SOA Cookbook, O'Reilly Media, 2009, p. 27

E. Newcomer and G. Lomow, Understanding SOA with Web Services, Addison-Wesley, 2005, p. 151-159

E. Newcomer and G. Lomow, Understanding SOA with Web Services, Addison-Wesley, 2005, p. 5

E. Newcomer and G. Lomow, Understanding SOA with Web Services, Addison-Wesley, 2005, p. 39

E. Newcomer and G. Lomow, Understanding SOA with Web Services, Addison-Wesley, 2005, p. 45

E. Newcomer and G. Lomow, Understanding SOA with Web Services, Addison-Wesley, 2005, p. 72-73

E. Hewitt, Java SOA Cookbook, O'Reilly Media, 2009, p. 39

E. Hewitt, Java SOA Cookbook, O'Reilly Media, 2009, p. 48

E. Hewitt, Java SOA Cookbook, O'Reilly Media, 2009, p. 69

M. Jeckle and E. Wilde, Identical Principles, Higher Layers: Modeling Web Services as Protocol Stack; http://www.jeckle.de/files/xmleu04/index.html#S4. [Προσπελάστηκε 27/7/11]

T. Clements, Overview of SOAP, Oracle Sun Developer Network (SDN), January 2002, http://java.sun.com/developer/technicalArticles/xml/webservices/ [Προσπελάστηκε 3/8/11]

M. Gudgin et al., SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), W3C Recommendation, 27 April 2007; http://www.w3.org/TR/soap12-part1/ [Προσπελάστηκε 8/8/11]

E. Cherami, Web Services Essentials, O'Reilly and Associates, 2002, p. 119

R. Chinnici et al., Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, W3C Recommendation, 26 June 2007; http://www.w3.org/TR/wsdl20/ [Προσπελάστηκε 18/6/11]

E. Cherami, Web Services Essentials, O'Reilly and Associates, 2002, p. 157

T. Jewell, D. Chappell, Java Web Services, Chapter 6- UDDI: Universal Description, Discovery and Integration, O'Reilly Online Catalog, March 2002, http://oreilly.com/catalog/javawebserv/chapter/ch06.html [Προσπελάστηκε 17/8/11]

E. Hewitt, Java SOA Cookbook, O'Reilly Media, 2009, p. 231 – 233

E. Hewitt, Java SOA Cookbook, O'Reilly Media, 2009, p. 238 – 241

E. Hewitt, Java SOA Cookbook, O'Reilly Media, 2009, p. 249 – 253

E. Hewitt, Java SOA Cookbook, O'Reilly Media, 2009, p. 379 – 384

E. Newcomer and G. Lomow, Understanding SOA with Web Services, Addison-Wesley, 2005, p. 124 – 126

E. Newcomer and G. Lomow, Understanding SOA with Web Services, Addison-Wesley, 2005, p. 29

E. Newcomer and G. Lomow, Understanding SOA with Web Services, Addison-Wesley, 2005, p. 223 – 228

M. Juric et al., WS-BPEL 2.0 for SOA Composite Applications with IBM WebSphere 7, Packt Publishing, 2010, p. 13 – 41

M. Juric et al., WS-BPEL 2.0 for SOA Composite Applications with IBM WebSphere 7, Packt Publishing, 2010, p. 59 – 63

M. Juric et al., WS-BPEL 2.0 for SOA Composite Applications with IBM WebSphere 7, Packt Publishing, 2010, p. 63 – 68

M. Juric et al., WS-BPEL 2.0 for SOA Composite Applications with IBM WebSphere 7, Packt Publishing, 2010, p. 71 – 73

The Java API for XML based Web Services (JAX-WS) 2.0 specification, Final Release, Sun Microsystems Inc., April 2006