Web Services

και

SOAP

Πίνακας Περιεχομένων

1 Εισαγωγή στα web services

1.1 Τι είναι τα web services

Υπάρχουν πολλοί ορισμοί για το τι είναι web services, περίπου όσοι και οι εταιρίες πληροφορικής που αναπτύσουν εργαλεία για τα web services.

Ένας πολύ καλός ορισμός έρχεται από την IBM[1]:

> *Τα web services είναι μια τεχνολογία που επιτρέπει στις εφαρμογές να επικοινωνούν μεταξύ τους ανεξαρτήτως πλατφόρμας και γλώσσας προγραμματισμού. Ένα web service είναι μια διεπαφή λογισμικού (software interface) που περιγράφει μια συλλογή από λειτουργίες οι οποίες μπορούν να προσεγγιστούν από το δίκτυο μέσω πρότυπων μηνυμάτων XML. Χρησιμοποιεί πρότυπα βασισμένα στη γλώσσα XML για να περιγράψει μία λειτουργία (operation) προς εκτέλεση και τα δεδομένα προς ανταλλαγή με κάποια άλλη εφαρμογή. Μια ομάδα από web services οι οποίες αλληλεπιδρούν μεταξύ τους καθορίζει μια εφαρμογή web services.*

Η Microsoft μέσα από το MSDN της καταλήγει ότι όλα τα web services έχουν τρία (3) κοινά χαρακτηριστικά[2]:

- *Τα web services εκθέτουν χρήσιμη λειτουργικότητα σε χρήστες του διαδικτύου μέσα από ένα πρότυπο δικτυακό πρωτόκολο. Στις περισσότερες περιπτώσεις αυτό το πρωτόκολλο είναι το SOAP (Simple Object Access Protocol).*
- *Τα web services παρέχουν ένα τρόπο να περιγράψουν τις διεπαφές τους με αρκετή λεπτομέρεια ώστε να επιτρέψουν στο χρήστη τους να χτίσει μια εφαρμογή πελάτη η οποία να επικοινωνήσει μαζί τους. Η περιγραφή συνήθως παρέχεται σε ένα έγγραφο XML το οποίο ονομάζεται έγγραφο WSDL (Web Services Description Language).*
- *Τα web services καταχωρούνται ώστε οι δυνητικοί χρήστες να μπορούν να τα βρουν εύκολα. Αυτό γίνεται με το UDDI (Universal Discovery Description and Integration).*

Τα web services λοιπόν αποτελούν μία αρχιτεκτονική κατανεμημένων συστημάτων κατασκευασμένη από πολλά διαφορετικά υπολογιστικά συστήματα τα οποία επικοινωνούν μέσω του δικτύου ώστε να δημιουργήσουν ένα σύστημα. Αποτελούνται από ένα σύνολο από πρότυπα τα οποία επιστρέπουν στους υπεύθυνους για την ανάπτυξη (προγραμματιστές - developers) να υλοποιήσουν κατανεμημένες εφαρμογές -χρησιμοποιώντας διαφορετικά εργαλεία από διαφορετικούς προμηθευτές- ώστε να κατασκευάσουν εφαρμογές που χρησιμοποιούν ένα συνδυασμό από ενότητες λογισμικού (software modules) οι οποίες καλούνται από συστήματα που ανήκουν σε διαφορετικά τμήματα ενός οργανισμού ή σε διαφορετικούς οργανισμούς[3].

1.2 Πλεονεκτήματα σε σχέση με παλαιότερες κατανεμημένες τεχνολογίες

Μία εύλογη ερώτηση είναι τι διαφορετικό έχει να προσφέρει αυτή η τεχνολογία σε σχέση με προηγούμενες κατανεμημένες τεχνολογίες.

1.2.1 Ευκολότερος χειρισμός δεδομένων

Παραδοσιακά το κυριότερο πρόβλημα στις κατανεμημένες τεχνολογίες ήταν το λεγόμενο tight-coupling ή στα ελληνικά η ισχυρή συνδεδιμότητα. Μια εφαρμογή που καλούσε μια άλλη απομακρυσμένη ήταν αυστηρά δεμένη με αυτή από την κλήση λειτουργίας (function call) που εκτελούσε και τις παραμέτρους που περνούσε. Στα περισσότερα συστήματα πριν από την έλευση των web services ο τρόπος επικοινωνίας ήταν μια σταθερή διεπαφή με λίγη έως καθόλου ευελεξία ή προσαρμοστικότητα στα περιβάλλοντα ή τις ανάγκες που μεταβάλλονται συνεχώς.

Τα web services χρησιμοποιούν τη γλώσσα XML η οποία μπορεί να περιγράψει οποιαδήποτε δεδομένα σε ένα πραγματικά ανεξάρτητο από πλατφόρμα τρόπο για ανταλλαγή αυτών των δεδομένων μεταξύ συστημάτων. Με αυτόν τον τρόπο οδηγούμαστε σε εφαρμογές με χαλαρή συνδεσιμότητα (loosely-coupled). Επιπλέον τα web services μπορούν να λειτουργήσουν σε πιο αφηρημένο επίπεδο στο οποίο μπορούν να επαναξιολογήσουν, να τροποποιήσουν ή να χειριστούν τύπους δεδομένων δυναμικά κατά περίπτωση. Έτσι σε τεχνικό επίπεδο τα web services μπορούν να χειριστούν δεδομένα πολύ ευκολότερα και να επιτρέψουν στο λογισμικό να επικοινωνεί πιο ελεύθερα[1].

1.2.2 Απλότητα πρωτοκόλλου επικοινωνίας

Τα web services χρησιμοποιούν ως πρωτόκολλο επικοινωνίας το SOAP. Το πρωτόκολλο αυτό είναι πολύ πιο απλό από πρωτόκολλα παλαιότερων τεχνολογιών όπως αυτά που χρησιμοποούνταν από τα κατανεμημένα περιβάλλοντα CORBA , DCOM, RPC. Έτσι το να δημιουργήσει κανείς μια υλοποίηση SOAP που υπόκειται στα πρότυπα (standards-compliant) είναι πολύ πιο εύκολο. Σήμερα μπορεί να βρει κανείς υλοποιήσεις του SOAP από τις μεγαλύτερες εταιρίες πληροφορικής αλλά ακόμη και από μεμονομένους προγραμματιστές, πράγμα αδιανόητο για παλαιότερες κατανεμημένες τεχνολογίες[2].

1.2.3 Απλότητα υποδομής

Τα web services λειτουργούν με πρότυπες γλώσσες και πρωτόκολλα όπως η XML , το HTTP και το TCP/IP. Η πλειονότητα των εταιριών έχουν ήδη την δικτυακή υποδομή και τους ανθρώπους με γνώσεις και εμπειρία που τη

συντηρούν. Έτσι το κόστος για την εφαρμογή των web services είναι σημαντικά μικρότερο από αυτό των προηγούμενων τεχνολογιών[2].

1.2.4 Ευκολία στην επικοινωνία

Με τις προηγούμενες τεχνολογίες η συνεργασία μεταξύ εταιριών ήταν ένα θέμα διότι κατανεμημένες τεχνολογίες όπως CORBA και DCOM χρησιμοποιούσαν μη πρότυπες πόρτες. Σαν αποτέλεσμα η συνεργασία σήμαινε άνοιγμα «οπών» στα τείχη προστασίας (firewalls) κάτι που πολλές φορές δεν ήταν αποδεκτό από τους ανθρώπους της πληροφορικής σε μια εταιρία αφού έθετε σε κίνδυνο στην ασφάλεια των συστημάτων. Το γεγονός αυτό δεν επέτρεπε δυναμική συνεργασία λόγω του ότι απαιτούσε μια χειροκίνητη διαδικασία για τη συνεργασία μιας εταιρίας με τους συνεργάτες της. Τα web services μπορούν να χρησιμοποιήσουν (μεταξύ άλλων) το HTTP ως πρωτόκολλο μεταφοράς και τα περισσότερα τείχη προστασίας επιτρέπουν την πρόσβαση μέσω της θύρας 80 (πρότυπη θύρα για το HTTP). Με αυτόν τον τρόπο οδηγούμαστε σε ευκολότερες και δυναμικές συνεργασίες μεταξύ των συστημάτων των εταιριών[4].

1.2.5 Διαλειτουργικότητα και ευκολία ανάπτυξης νέων εφαρμογών

Οι προηγούμενες κατανεμημένες τεχνολογίες υπέφεραν από ζητήματα διαλειτουργικότητας διότι κάθε προμηθευτής (vendor) υλοποιούσε το δικό του πρότυπο για distributed object messaging. Με την XML σαν το μόνο πρότυπο στα web services, συστήματα φτιαγμένα από διαφορετικές τεχνολογίες όπως η Java και το .Net μπορούν να επικοινωνήσουν μεταξύ τους[4]. Επιπλέον λόγω της απλότητας της XML είναι πολύ πιο εύκολο να γραφτούν νέες εφαρμογές σε μικρό χρονικό διάστημα.

1.3 Τα web services από την επιχειρηματική σκοπιά

Σε ένα υψηλότερο εννοιολογικό επίπεδο, μπορούμε να δούμε τα web services σαν μονάδες εργασίας (units of work). Ένα βήμα παραπέρα, αυτές οι μονάδες μπορούν να συνδυαστούν σε εργασίες επιχειρησιακού προσανατολισμού για να χειριστούν συγκεκριμένες επιχειρησιακές λειτουργίες. Αυτό με τη σειρά του επιτρέπει σε μη τεχνικούς ανθρώπους να σχεδιάσουν εφαρμογές οι οποίες μπορούν να χειριστούν τα επιχειρησιακά ζητήματα συνδυάζοντας τα web services σε ροές εργασίας. Σε μία αναλογία από τον χώρο των αυτοκινήτων, ο αρχιτέκτονας των επιχειρησιακών διαδικασιών μπορεί να συνδυάσει ολόκληρο τον κινητήρα με το πλαίσιο, τη μετάδοση και τα υπόλοιπα υποσυστήματα παρά να ασχοληθεί με τα εσωτερικά κομμάτια του κινητήρα. Επιπλέον η δυναμική πλατφόρμα σημαίνει ότι ο κινητήρας μπορεί να συνεργαστεί με τη μετάδοση ή άλλα υποσυστήματα διαφορετικών κατασκευαστών.

Αυτό που προκύπτει από αυτή την πτυχή είναι ότι τα web services βοηθούν στη γεφύρωση του κενού που υπάρχει ανάμεσα στους ανθρώπους του επιχειρείν και τους ανθρώπους της πληροφορικής σε ένα οργανισμό. Οι άνθρωποι του επιχειρείν μπορούν να περιγράψουν γεγονότα και δραστηριότητες και οι τεχνικοί μπορούν να τα συνδέσουν με τις κατάλληλες υπηρεσίες.

Με καθολικά καθορισμένες διεπαφές και καλά σχεδιασμένες λειτουργίες, γίνεται επίσης εύκολο να επαναχρησιμοποιηθούν αυτές οι λειτουργίες άρα και οι εφαρμογές που αντιπροσωπεύουν. Η επαναχρησιμοποιήση μιας εφαρμογής λογισμικού σημαίνει καλύτερη απόδοση της επένδυσης (return on investment) διότι μπορεί να παράγει περισσότερα με τους ίδιους πόρους. Επιτρέπει στους ανθρώπους να εξετάσουν το ενδεχόμενο χρησιμοποίησης μιας ήδη υπάρχουσας εφαρμογής με ένα διαφορετικό τρόπο ή το ενδεχόμενο προσφοράς αυτής σε ένα συνεργάτη με διαφορετικό τρόπο, αυξάνοντας ενδεχομένως τις επιχειρησιακές συναλλαγές μεταξύ των συνεργατών.

Επομένως, τα κύρια ζητήματα που τα web services προσπαθούν να λύσουν είναι τα ζητήματα ολοκλήρωσης (integration) δεδομένων και εφαρμογών και αυτά της μεταμόρφωσης των τεχνικών λειτουργιών σε υπολογιστικές λειτουργίες επιχειρησιακού προσανατολισμού[4].

1.4 Τα web services από την τεχνική σκοπιά

Ενώ τα web services προσφέρουν όλα αυτά τα δυναμικά χαρακτηριστικά ώστε να συνδυάσουμε υπηρεσίες σε εφαρμογές, πρέπει πρώτα να χτίσουμε αυτές τις υπηρεσίες. Οι γλώσσες προγραματισμού στην πληροφορική συνεχώς εξελίσονται. Ξεκινήσαμε δεκαετίες πριν με την ιδέα της function στην οποία παρέχουμε μερικές παραμέτρους, εκτελεί μια λειτουργία με αυτές τις παραμέτρους, και επιστρέφει μια τιμή βασισμένη στους υπολογισμούς που έγιναν. Τελικά, αυτή η πρώτη έννοια εξελίχθηκε σε αντικείμενο (object) όπου κάθε αντικείμενο είχε όχι απλώς έναν αριθμό από λειτουργίες (functions) που μπορεί να εκτελέσει αλλά και τις δικές του ιδιωτικές μεταβλητές (private data variables), αντί να στηρίζεται σε εξωτερικές μεταβλητές του συστήματος που προηγουμένως έκανα πιο περίπλοκη την ανάπτυξη εφαρμογών. Δεδομένου ότι οι εφαρμογές άρχισαν να επικοινωνούν μεταξύ τους, η έννοια του καθορισμού καθολικών διεπαφών (universal interfaces) για αντικείμενα έγινε σημαντική, επιτρέποντας αντικείμενα σε διαφορετικές πλατφόρμες να επικοινωνούν ακόμη και αν είχαν αναπτυχθεί σε διαφορετικές γλώσσες προγραμματισμού και εκτελούνταν σε διαφορετικά λειτουργικά συστήματα[1].

Στο πιο πρόσφατο βήμα, τα web services προχώρησαν μπροστά με την έννοια των διεπαφών και επικοινωνιών καθορισμένων με XML, ενώνοντας τελικά κάθε είδους εφαρμογή με οποιαδήποτε άλλη, όπως και παρέχοντας την ελευθερία στις εφαρμογές αν αλλάξουν και να εξελιχθούν με το χρόνο, αρκεί να είναι σχεδιασμένες σύμφωνα με την κατάλληλη διεπαφή. Η μεταβλητότητα της

XML είναι αυτό που κάνει τα web services διαφορετικά από τεχνολογίες προηγούμενων γενεών. Επιτρέπει το διαχωρισμό της γραμματικής δομής (syntax) και της γραμματικής έννοιας (semantics), και του πώς αυτά υποβάλλονται σε επεξεργασία και κατανοούνται από μία υπηρεσία και το περιβάλλον μέσα στο οποίο υπάρχει. Έτσι λοιπόν τώρα, τα αντικείμενα μπορούν να καθοριστούν σαν υπηρεσίες, οι οποίες επικοινωνούν με άλλες υπηρεσίες σε γραμματική καθορισμένη σε XML, με την οποία κάθε υπηρεσία μεταφράζει και αναλύει το μήνυμα σύμφωνα με μην τοπική της υλοποίησης και το περιβάλλον της. Κατά συνέπεια μια διακτυακή εφαρμογή μπορεί πραγματικά να συντεθεί από πολλαπλές οντότητες διαφόρων υλοποιήσεων και σχεδιασμών εφόσον προσαρμόζονται στους κανόνες που καθορίζονται από την προσανατολισμένη στις υπηρεσίες αρχιτεκτονική τους[1].

Κατά συνέπεια, με αυτά στο μυαλό, τα web services μας επιτρέπουν[1]:

- Την αλληλεπίδραση μεταξύ υπηρεσιών σε οποιαδήποτε πλατφόρμα, γραμμένες σε οποιαδήποτε γλωσσα προγραμματισμού.
- Να αντιληφθούμε λειτουργίες εφαρμογών ως εργασίες, οδηγούμενοι σε ανάπτυξη και ροές εργασιών προσανατολισμένες σε εργασίες. Αυτό επιτρέπει μια υψηλότερη αφαίρεση του λογισμικού το οποίο μπορεί να υιοθετηθεί από λιγότερο τεχνικά καταρτισμένους χρήστες.
- Τη χαλαρή συνδεσιμότητα μεταξύ εφαρμογών, πράγμα που σημαίνει ότι αλληλεπιδράσεις μεταξύ υπηρεσιών δε θα χαλάνε κάθε φορά που υπάρχει κάποια αλλαγή το πώς μία ή περισσότερες υπηρεσίες σχεδιάζονται ή υλοποιούνται.
- Την προσαρμογή ήδη υπάρχουσων εφαρμογών στις μεαβαλλόμενες επιχειρησιακές συνθήκες και ανάγκες των πελατών.
- Να παρέχουμε υπάρχουσες εφαρμογές λογισμικού με διαπαφές υπηρεσιών χωρίς να αλλάξουμε τις αρχικές εφαρμογές, επιτρέποντάς τους να λειτουργούν πλήρως στο περιβάλλον των υπηρεσιών.
- Να εισάγουμε άλλες διοικητικές λειτουργίες ή λειτυοργίες διαχείρισης διαδικασιών όπως η αξιοπιστία, υπευθυνότητα, ασφάλεια, κ.λπ., ανεξάρτητες της αρχικής λειτουργίας μιας εφαρομογής, αυξάνοντας κατά συνέπεια τη μεταβλητότητα και τη χρησιμότητά της στο επιχειρησιακό περιβάλλον.

1.5 Εφαρμογές των web services

Τα πρώτα web services σκόπευαν να είναι πηγές πληροφορίας τις οποίες μπορεί κανείς πολύ εύκολα να ενσωματώσει στις εφαρμογές του : τιμές μετοχών, προβλέψεις καιρού, αποτελέσματα αθλητικών παινιδιών κλπ. Έιναι εύκολο να φανταστεί κανείς μια ολόκληρη κατηγορία εφαρμογών που μπορεί να κατασκευάσει ώστε να αναλύει και να συνδυάζει πληροφορία που τον ενδιαφέρει και να την παρουσιάζει με ποικίλους τρόπους. Για παράδειγμα, θα μπορούσαμε να έχουμε ένα λογιστικό φύλλο (spreadsheet) το οποίο συνοψίζει όλη την οικονομική μας εικόνα : μετοχές, τραπεζικούς λογαριασμούς, δάνεια κλπ. Αν αυτή η πληροφορία ήταν διαθέσιμη μέσω web services το λογιστικό

φύλλο θα μπορούσε να ενημερώνεται συνεχώς. Οι περισσότερες από αυτές τις πληροφορίες είναι ήδη διαθέσιμες στον παγκόσμιο ιστό αλλά τα web services θα κάνουν την προγραμματιστική πρόσβαση σε αυτές πιο εύκολη και πιο αξιόπιστη[2].

Εκθέτοντας ήδη υπάρχουσες εφαρμογές σαν web services θα επιτρέψει στους χρήστες να κατασκευάσουν νέες πιο ισχυρές εφαρμογές οι οποίες χρησιμοποιούν τα web services σαν δομικά στοιχεία. Για παράδειγμα, ένας χρήστης θα μπορούσε να αναπτύξει μια εφαρμογή προμηθειών η οποία να παίρνει αυτόματα τιμές από προμηθευτές, να επιστρέπει στο χρήστη να επιλέξει προμηθευτή, να υποβάλει την παραγγελία και να παρακολουθεί την αποστολή έως ότου να γίνει η παραλαβή της. Η εφαρμογή του προμηθευτή, εκτός από το να εκθέτει τις υπηρεσίες της στον ιστό, θα μπορούσε να χρησιμοποιήσει άλλα web services για να ελέγξει την πιστωληπτική ικανότητα του πελάτη, να χρεώσει τον τραπεζικό λογαριασμό του πελάτη και να καθορίσει την αποστολή με μια εταιρία μεταφορών[2].

Στο άμεσο μέλλον, μερικά από τα πιο ενδιαφέροντα web services θα υποστηρίζουν εφαρμογές που χρησιμοποιούν τον ιστό για να κάνουν πράγματα που δεν μπορούν να γίνουν σήμερα. Για παράδειγμα, μία από τις υπηρεσίες που τα web services θα κάνουν δυνατή είναι η υπηρεσία ημερολογίου. Αν ο οδοντίατρος ή ο μηχανικός σας εξέθεταν τα ημερολόγιά τους μέσω μιας τέτοιας web service, θα μπορούσατε να προγραμματίσετε τα ραντεβού σας με αυτούς ή θα μπορούσαν να προγραμματίσουν αυτοί τα ραντεβού κατευθείαν στο δικό σας ημερολόγιο άν θέλατε. Με λίγη φαντασία, μπορούμε να οραματιστούμε εκατοντάδες εφαρμογές οι οποίες μπορούν να κατασκευαστούν μόλις έχουμε τη δυνατότητα να προγραμματίσουμε τον ιστό[2].

2 Αρχιτεκτονική και δομικά στοιχεία των web services

Για να καταλάβουμε τις τεχνολογίες που απαιτούνται για τα web services πρέπει πρώτα να καταλάβουμε μία τυπική αλληλεπίδραση των web services.

2.1 Μία αναλογία

Ας εξετάσουμε ένα σενάριο στο οποίο χρειάζεται να εντοπίσουμε ένα συγκεκριμένο φαρμακείο στην περιοχή μας το οποίο έχει και ιστοσελίδα στο διαδίκτυο. Δεν θα βγαίναμε στο δρόμο και θα ρωτούσαμε κάθε άτομο που θα συναντούσαμε για την ταχυδρομική διεύθυνση του φαρμακείου. Αντί για αυτό θα μπορούσαμε να αναφερθούμε στην ιστοσελίδα του φαρμακείου στο διαδίκτυο. Αν γνωρίζαμε την ηλεκτρονική διεύθυνση της ιστοσελίδας θα πλοηγούμασταν κατευθείαν σε αυτή και θα βρίσκαμε την ταχυδρομική διεύθυνση για την οποία ενδιαφερόμαστε. Αν δεν γνωρίζαμε την ηλεκτρονική διεύθυνση του φαρμακείου θα πλοηγούμασταν σε μια μηχανή αναζήτησης και θα πληκτρολογούσαμε το όνομα του φαρμακείου στη γλώσσα την οποία η μηχανή αναζήτησης αναγνωρίζει. Αφού βρίσκαμε την ιστοσελίδα, θα πλογούμασταν σε αυτήν και στη συνέχεια θα βρίσκαμε την ταχυδρομική διεύθυνση.

Η δομή των web services είναι παραπλήσια. Αν κοιτάξουμε προσεκτικά το προηγούμενο παράδειγμα θα δούμε ότι υπάρχει ένας αιτούντας ή καταναλωτής : εμείς. Υπάρχει επίσης μία υπηρεσία : το φαρμακείο. Η κεντρική βάση δεδομένων είναι το διαδίκτυο, μέσω της οποίας θα βρούμε την τοποθεσία του φαρμακείου. Στο παράδειγμα, όταν εκτελούμε μία αναζήτηση στη μηχανή αναζήτησης, η αίτησή μας τυλίγεται σε μία δομή, της οποίας η γλώσσα είναι προκαθορισμένη και τοπική, και τότε περνιέται στον εξυπηρετητή της μηχανής αναζήτησης.

Στα web services το SOAP, UDDI και WSDL αντιπροσωπεύουν τους ρόλους που αναφέρθηκαν στα παραπάνω βήματα.

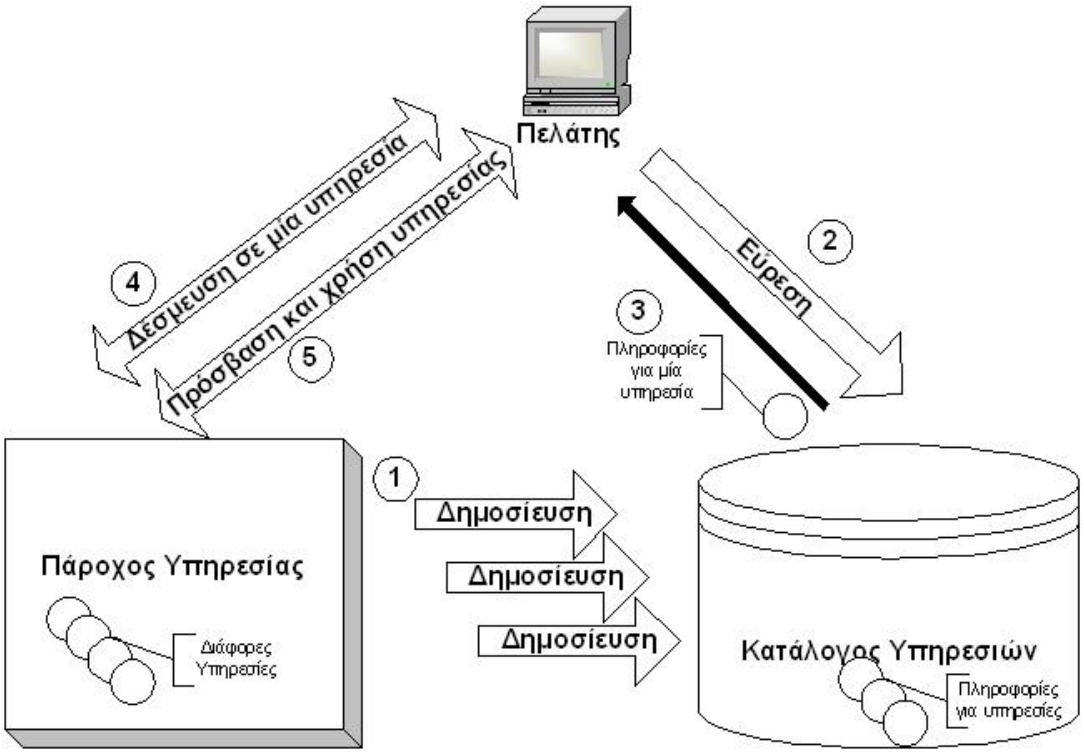
Το SOAP (Simple Object Access Protocol) είναι η μέθοδος με την οποία μπορούμε να στέλνουμε μηνύματα σε διαφορετικές ενότητες λογισμικού (software modules). Είναι παρόμοιο με το πώς επικοινωνούμε με τη μηχανή αναζήτησης.

Το UDDI (Universal Description, Discovery and Integration) είναι η καθολική βάση αναζήτησης για τον εντοπισμό των υπηρεσιών. Στο προηγούμενο παράδειγμα, είναι ανάλογη της υπηρεσίας ευρετηρίου της μηχανής αναζήτησης, στην οποία όλες οι ιστοσελίδες καταχωρούνται και σχετίζονται με λέξεις κλειδιά. Διατηρεί ένα κατάλογο με όλα τα φαρμακεία της χώρας.

Η WSDL (Web Services Definition Language) είναι ο τρόπος με τον οποίο διαφορετικές υπηρεσίες περιγράφονται στο UDDI. Αυτή αντιστοιχεί στην πραγματική μηχανή αναζήτησης[5].

2.2 Το μοντέλο των web services

Το μοντέλο των web services ακολουθεί το παράδειγμα ***δημοσίευση (publish), εύρεση(find) και σύνδεση(bind)***. Στο πρώτο βήμα, ο προμηθευτής της υπηρεσίας δημοσιεύει την υπηρεσία σε ένα κατάλογο υπηρεσιών. Στο δεύτερο βήμα, ο πελάτης ο οποίος ψάχνει για μία υπηρεσία η οποία να καλύπτει τις απαιτήσεις του την αναζητεί στον κατάλογο. Αφού επιτυχημένα βρει πολλαπλές υπηρεσίες επιλέγει μία βάσει των προτιμήσεών του. Τότε μεταφορτώνει την περιγραφή της υπηρεσίας και συνδέεται (δεσμεύεται) με αυτήν ώστε να μπορέσει να καλέσει και να εκτελέσει την υπηρεσία[4].

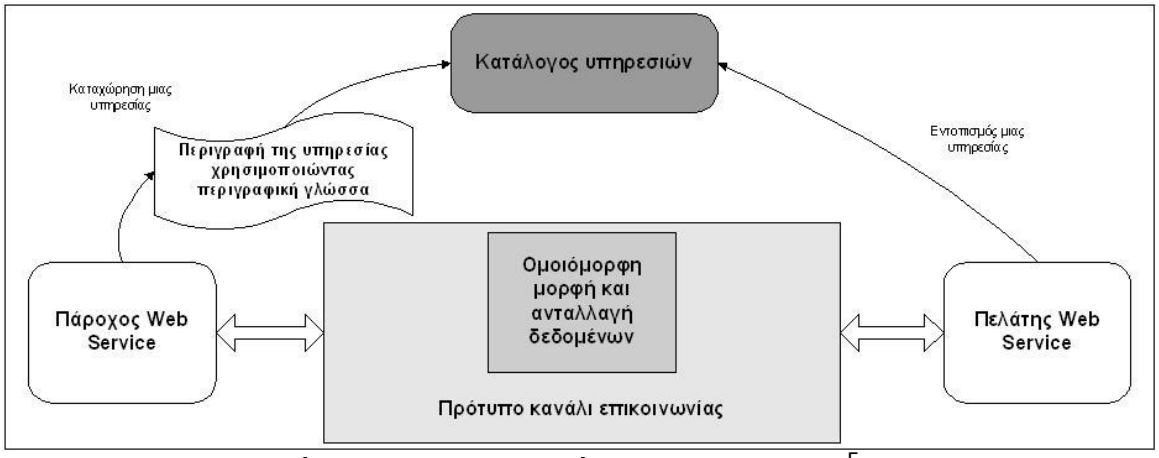


Σχήμα 1. Μοντέλο των web services [6]

Όταν μιλάμε λοιπόν για μία αρχιτεκτονική προσανατολισμένη στις υπηρεσίες προκύπτουν ορισμένα ζητήματα. Η εφαρμογή που παρέχει την υπηρεσία και η εφαρμογή-πελάτης η οποία χρησιμοποιεί την υπηρεσία μιλάνε μεταξύ τους σε μια κοινή γλώσσα. Έπειτα οι δύο εφαρμογές χρειάζονται ένα τρόπο να εντοπίζουν η μία την άλλη πριν ξεκινήσουν να μιλούν μεταξύ τους. Αυτό αληθεύει ακόμη παραπάνω για τις κατανεμημένες εφαρμογές όπου μία εφαρμογή δεν έχει καμία γνώση της θέσης της άλλης.

Ως εκ τούτου, μπορούμε να πούμε ότι μια βασική αρχιτεκτονική για web services πρέπει να παρέχει[5]:

- Έναν πρότυπο τρόπο για επικοινωνία.
- Ένα ομοιόμορφο μηχανισμό για περιγραφή και ανταλλαγή των δεδομένων.
- Μια πρότυπη περιγραφική γλώσσα (meta language) για να περιγράψει τις υπηρεσίες που προσφέρονται.
- Ένα μηχανισμό για να καταχωρούνται και να εντοπίζονται οι εφαρμογές που βασίζονται σε web services.



Σχήμα 2. Αρχιτεκτονική των web services [5]

2.3 Διαφορές από προηγούμενες τεχνολογίες

Τίποτα από τα παραπάνω δεν είναι καινούργιο. Στην πραγματικότητα τα SUN/RPC, DCE/RPE, DCOM και EJB όλα παρέχουν παρόμοιες υπηρεσίες. Η διαφορά, εντούτοις, είναι στο πώς παρέχονται αυτές οι υπηρεσίες. Οι παραδοσιακές RPC υπηρεσίες απαιτούν παρόμοια αρχιτεκτονική υποδομής, μορφή δεδομένων, κλπ. Για παράδειγμα, δύο διαφορετικές υλοποιήσεις RPC προκειμένου να επικοινωνήσουν, και οι δύο πρέπει να παράσχουν μηχανισμούς επικοινωνίας. Ιστορικά τέτοια επικοινωνία καθορίστηκε φτωχά και απαιτούσε μεγάλη φροντίδα για να μπορέσουν τα δύο συστήματα να επικοινωνήσουν[6].

Τα web services διαφέρουν από τους παραδοσιακούς RPC μηχανισμούς από πολλές απόψεις[6]:

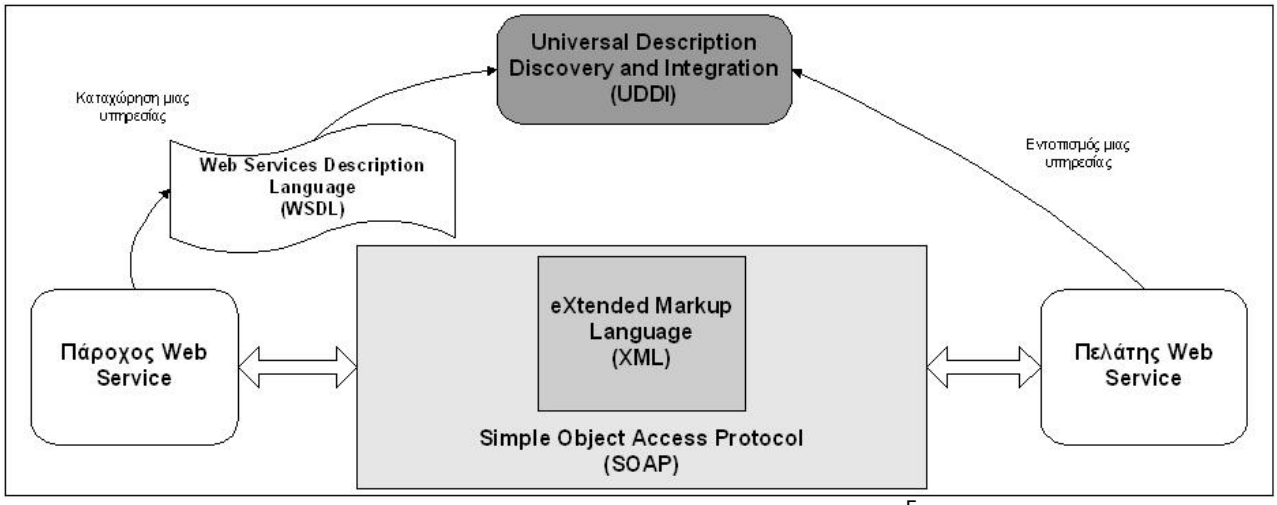
- Τα δεδομένα είναι μορφοποιημένα για μεταφορά χρησιμοποιώντας XML, βελτιώνοντας ή εξαλείφοντας το marshalling, το unmarshalling και άλλες σχετικά με τη μετάφραση απαιτήσεις που συνήθως προγραμματίζονταν από τον ίδιο τον προγραμματιστή.
- Τα δεδομένα ανταλλάσσονται χρησιμοποιώντας προτυποποιημένα πρωτόκολλα όπως το HTTP ή το SMTP, τα οποία έχουν δημοσιευμένα καλά καθορισμένα πρότυπα.
- Η προς έκθεση υπηρεσία είναι καλά καθορισμένη χρησιμοποιώντας ένα γνωστό και αποδεκτό μηχανισμό, την WSDL.
- Οι υπηρεσίες ανευρίσκονται χρησιμοποιώντας ένα καλά καθορισμένο πρότυπο, το UDDI, και το πιο εξελογμένο ebXML.

2.4 Βασικές Τεχνολογίες των web services

Οι βασικές τεχνολογίες στις οποίες βασίονται τα web services συνοψίζονται στον παρακάτω πίνακα[5]:

Επίπεδο	**Τεχνολογία**	**Περιγραφή**
Ομοιόμορφος ορισμός και ανταλλαγή δεδομένων	XML	Η Extended Markup Language (XML) είναι μια μέτα-γλώσσα (περιγραφική γλώσσα) η οποία έχει καλή καθορισμένη σύνταξη και σημασιολογία. Τα «αυτοπεριγραφικά» χαρακτηριστικά της XML την κάνουν απλό, αλλά δυνατό, μηχανισμό για τη σύλληψη και την ανταλλαγή των στοιχείων μεταξύ των διαφορετικών εφαρμογών.
Πρότυπο κανάλι επικοινωνίας	SOAP	Το Simple Object Access Protocol (SOAP) είναι το κανάλι που χρησιμοποιείται για επικοινωνία μετξύ μιας εφαρμογής – προμηθευτή web services και μιας εφαρμογής-πελάτη. Η απλότητα του SOAP είναι το ότι δεν καθορίζει κανένα νέο πρωτόκολλο μεταφοράς. Αντίθετα, επαναχρησιμοποιεί μεταξύ άλλων το Hyper Text Transfer Protocol (HTTP) ή το Simple Mail Transfer Protocol (SMTP) για μεταφορά δεδομένων σαν μηνύματα. Αυτή η χρήση του HTTP ή του SMTP σαν πρωτόκολλο μεταφοράς εξασφαλίζει ότι οι εφαρμογές – προμηθευτές με τις εφαρμογές – πελάτες μπορούν να επικοινωνήσουν χρησιμοποιώντας το διαδίκτυο σαν ραχοκοκαλιά. Είναι η χρήση του SOAP που πολλαπλασιάζει τις ικανότητες των web services.
Πρότυπη περιγραφική	WSDL	Οι εφαρμογές που παρέχουν web services διαφημίζουν τις διάφορες υπηρεσίες που

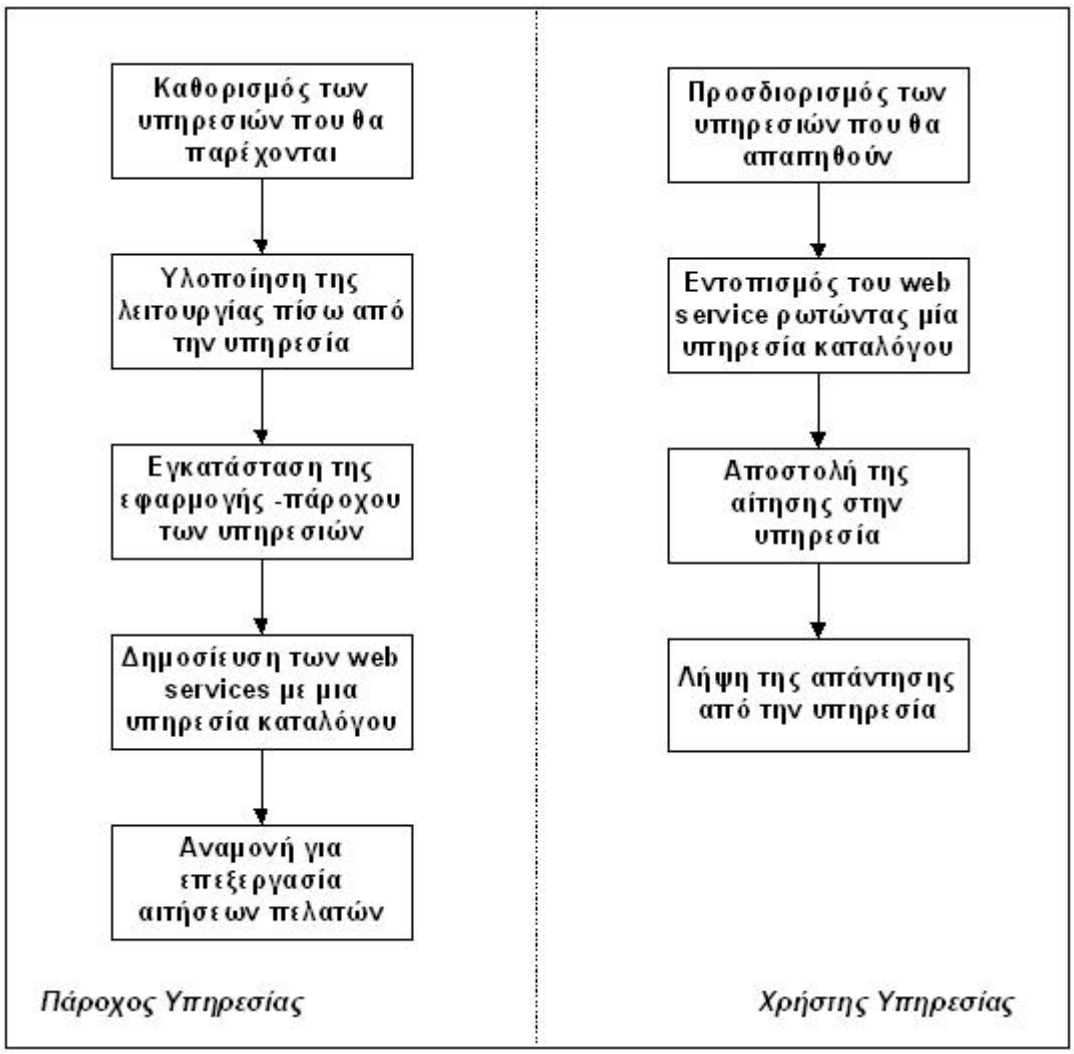
γλώσσα για την περιγραφή των παρεχόμενων υπηρεσιών		παρέχουν χρησιμοποιώντας μια πρότυπη περιγραφική γλώσσα που ονομάζεται Web Services Description Language (WSDL). Κατά τρόπο ενδιαφέροντα, η WSDL βασίζεται στην XML και χρησιμοποιεί ένα ειδικό σύνολο ετικετών (tags) για να περιγράψει ένα web service, τις υπηρεσίες που παρέχονται, που να εντοπιστεί και ούτω καθ'εξής. Οι εφαρμογές-πελάτες λαμβάνουν πληροφορίες για ένα web service πριν από την πρόσβασή τους σε αυτό και τελικά τη χρήση του.
Καταχώρηση και εντοπισμός των παρεχόμενων υπηρεσιών	UDDI	Ο «χρυσός οδηγός» των web services είναι το Universal Description Discovery and Integration (UDDI). Οι εφαρμογές που παρέχουν web services παρατίθενται σε ένα κατάλογο από πάροχους υπηρεσιών χρησιμοποιώντας το UDDI. Παρόμοια, οι εφαρμογές-πελάτες εντοπίζουν τους παρόχους εφαρμογών web services χρησιμοποιώντας UDDI. Όπως και στην περίπτωση της WSDL, και το UDDI βασίζεται στην XML.



Σχήμα 3. Τεχνολογίες των web services [5]

2.5 Βασικά βήματα για την ανάπτυξη εφαρμογών με web services

Τα βασικά βήματα για την ανάπτυξη μίας εφαρμογής web services είναι:



Σχήμα 4. Βήματα ανάπτυξης εφαρμογών [5]

Αυτά τα βήματα θα είναι τα ίδια ανεξαρτήτου τεχνολογίας και γλώσσας προγραμματισμού η οποία θα χρησιμοποιηθεί για την υλοποίηση των web services[5].

3 XML

Η eXtensible Markup Language (XML) είναι μια γλώσσα ανεξάρτητη από σύστημα και υλικό για την αναπαράσταση δεδομένων και της μορφής τους σε ένα έγγραφο XML (XML document). Ένα έγγραφο XML στην πιο απλή του μορφή είναι ένα αρχείο κειμένου το οποίο περιέχει δεδομένα μαζί με σήμανση η οποία καθορίζει τη δομή των δεδομένων[7].

Η XML είναι μια παγκοσμίως συμφωνημένη μεταγλώσσα σήμανσης που χρησιμοποιείται πρώτιστα για την ανταλλαγή πληροφοριών. Η ομορφιά της XML βρίσκεται στο γεγονός ότι είναι επεκτάσιμη. Απλά, η XML είναι ένα σύνολο προκαθορισμένων κανόνων (συντακτικό πλαίσιο) που πρέπει να ακολουθήσουμε κατά τη δόμηση των δεδομένων μας[5].

Για ένα μεγάλο χρονικό διάστημα, οι προγραμματιστές και οι προμηθετές εφαρμογών κατασκεύαζαν εφαρμογές και συστήματα εγκατεστημένα σε μια επιχείρηση τα οποία επεξεργάζονταν δεδομένα τα οποία μπορούσαν να με το δικό τους ιδιωτικό τρόπο. Αλλά καθώς η ανταλλαγή πληροφορίας μεταξύ εφαρμογών και συστημάτων στις επιχειρήσεις επικρατούσε, έγινε πολύ δύσκολο να αναταλλάσεις δεδομένα διότι τα συστήματα δε σχεδιάστηκαν ώστε να δέχονται δεδομένα από εξωτερικά, άγνωστα συστήματα[5].

Η XML παρέχει μία πρότυπη και κοινή δομή για τη διανομή δεδομένων μεταξύ ανόμοιων συστημάτων. Επιπλέον, η XML έχει ενσωματωμένο ένα μηχανισμό επικύρωσης δεδομένων, ο οποίος εγγυάται ότι η δομή των δεδομένων που λαμβάνεται είναι έγκυρη[5].

3.1 Παράδειγμα αναπαράστασης δεδομένων με XML

Ας δούμε πώς αναπαριστούμε δεδομένα με τη βοήθεια της XML[5]:

```
<employee>
        <shift id= "counter" time="8-12">
                <phone id = "1"> All phone information
                        <number>3444333</number >
                </phone>
        </shift >
        <shift id="help_desk" time="1-5">
                <phone id = "2"> All phone information
                        <number>332333</number >
                </phone>
        </shift >
        ...
        <home-address>
                <street>3434 Norwalk street</street>
                <city>New York</city>
                <state>NY</state>
        </home-address>
</employee>
```

Στο παραπάνω παράδειγμα, αναπαριστούμε τις προσωπικές πληροφορίες και τις πληροφορίες σχετικά με τις βάρδιες ενός υπαλλήλου σε ένα οργανισμό.
Βλέπουμε πώς η XML χρησιμοποιεί τις διακριτικές ετικέτες "<>" και "</>" παρόμοια με τις ετικέτες που χρησιμοποιούνται στην HTML. Αυτό συμβαίνει γιατί η XML είναι μια γλώσσα σήμανσης σαν την HTML.
Η κύρια διαφορά της XML με την HTML είναι ως προς τον σκοπό της κάθε μίας:

- Η XML σχεδιάστηκε για να περιγράφει δεδομένα και να εστιάσει στο τί είναι αυτά τα δεδομένα.
- Η HTML σχεδιάστηκε για να προβάλει δεδομένα και να εστιάσει στο πώς φαίνονται αυτά τα δεδομένα.

Στο παραπάνω παράδειγμα βλέπουμε ότι έχουμε να κάνουμε με καθαρά δεδομένα : ότι ένας υπάλληλος (employee) έχει παραπάνω από μία βάρδιες (shift), για παράδειγμα το πρωί εργάζεται στο ταμείο (counter) και το μεσημέρι στο γραφείο βοήθειας (help desk) και διευθύνσεις.
Οι δύο αρχικές δομικές μονάδες XML που χρησιμοποιούνται στο προηγούμενο παράδειγμα είναι τα elements (στοιχεία) και τα attributes (ιδιότητες)[5].

3.2 Elements ή Στοιχεία

Τα στοιχεία (elements) είναι ετικέτες, όπως και στην HTML, και περιέχουν τιμές. Επιπλέον τα elements είναι δομημένα σαν δένδρο. Ώς εκ τούτου έχουμε τα στοιχεία οργανωμένα σε ένα ιεραρχικό τρόπο με ένα στοιχείο-πατέρα και στοιχεία-παιδιά. Τα στοιχεία-παιδιά μπορούν να περιέχουν και αυτά άλλα στοιχεία-παιδιά και ούτω καθ'εξής[5].

Στο προηγούμενο παράδειγμα, το στοιχείο <employee> είναι το γονικό στοιχείο και έχει το στοιχείο <shift> σαν στοιχείο-παιδί. Παρακάτω το στοιχείο <phone> είναι στοιχείο-παιδί του γονικού στοιχείου <shift>.

Τα στοιχεία έχουν συγκεκριμένα χαρακτηριστικά. Ορισμένα από αυτά είναι[5]:

- Τα στοιχεία μπορεί να περιέχουν δεδομένα, οπως το στοιχείο <number> στο παράδειγμα.
- Αντίστροφα, τα στοιχεία μπορεί να μην περιέχουν δεδομένα αλλά μόνο ιδιότητες, όπως το στοιχείο <shift>.
- Εναλλακτικά, τα στοιχεία μπορεί να περιέχουν ταυτόχρονα και ιδιότητες αλλά και δεδομένα, αλλά επίσης και στοιχεία-παιδιά, όπως το στοιχείο <phone>.

Τα στοιχεία έχουν κάποιους κανόνες[8]:

- Όλα τα στοιχεία πρέπει να έχουν ετικέτα κλεισίματος αντίθετα με την HTML όπου υπάρχουν και ετικέτες που δε χρειάζονται κλείσιμο όπως για παράδειγμα η
.
- Οι ετικέτες των στοιχείων είναι case sensitive δηλαδή υπάρχει διαχωρισμός μεταξύ κεφαλαίων και πεζών και τα ονόματά τους υπακούουν σε κανόνες ονοματολογίας.
- Τα στοιχεία πρέπει να είναι τοποθετημένα σωστά αντίθετα με την HTML.
  HTML : This text is bold and italic
  XML : This text is bold and italic
- Τα έγγραφα της XML πρέπει να έχουν ακριβώς ένα αρχικό στοιχείο (root element).

3.3 Attributes ή Ιδιότητες

Οι ιδιότητες (attributes) μας βοηθούν να δώσουμε περισσότερο νόημα και να περιγράψουμε τα στοιχεία μας πιο αποτελεσματικά και με σαφήνεια. Στο προηγούμενο παράδειγμα, το στοιχείο <shift> έχει μία ιδιότητα "id" με τιμές "counter" και "help_desk" . Με τη χρήση τέτοιων ιδιοτήτων, μπορούμε να ξέρουμε άν ένας υπάλληλος εργα΄ζεται στο ταμείο ή στο γραφείο βοήθειας. Αυτό βοηθάει στο να κάνουμε τα δεδομένα σε ένα έγγραφο XML αυτοπεριγραφικά. Πρέπει πάντα να θυμόμαστε ότι ο κύριος σκοπός των ιδιοτήτων είναι να παρέχουν περισσότερη πληροφορία σχετική με ένα στοιχείο και δεν πρέπει να χρησιμοποιούνται για να περιέχουν τα ίδια τα δεδομένα[5].

Όπως και τα στοιχεία έτσι και οι ιδιότητες έχουν κάποιους κανόνες[5]:

- Οι τιμές των ιδιοτήτων πρέπει να είναι εσωκλείωνται σε '"' ή σε ''.
- Τα ονόματα των ιδιοτήτων ακολουθούν τους ίδιους κανόνες με αυτά των ετικετών.

3.4 Κανόνες ονομασίας

Τα στοιχεία και οι ιδιότητες στην XML πρέπει να ακολουθούν στους παρακάτω κανόνες[8]:

- Τα ονόματα μπορούν να περιέχουν γράμματα, αριθμούς και άλλους χαρακτήρες.
- Τα ονόματα δεν πρέπει να ξεκινούν με αριθμό ή χαρακτήρα στίξης.
- Τα ονόματα δεν πρέπει να ξεκινούν με τα γράμματα xml (ή XML, ή Xml κλπ.).
- Τα ονόματα δεν μπορούν να περιέχουν κενά.

3.5 Καλά διαμορφωμένα έγγραφα (well formed documents)

Ένα «καλά διαμορφωμένο» έγγραφο XML είναι ένα έγγραφο που υπακούει στους κανόνες σύνταξης της XML που αναφέραμε προηγουμένως[8]:

- Τα έγγραφα XML πρέπει να περιέχουν ένα αρχικό στοιχείο.
- Τα στοιχεία XML πρέπει να έχουν ετικέτες κλεισίματος.
- Στις ετικέτες XML υπάρχει διαχωρισμός κεφαλαίων και πεζών.
- Τα στοιχεία XML πρέπει να είναι σωστα τοποθετημένα.
- Οι ιδιότητες XML πρέπει να εσωκλείονται πάντα σε "" ή ''.

3.6 DTD

Όπως σε μία γλώσσα προγραμματισμού πρέπει να ξέρουμε τις προδιαγραφές της γλώσσας, με παρόμοιο τρόπο το Document Type Definition (DTD) είναι μία προδιαγραφή, η οποία πρέπει να ακολουθηθεί όταν δημιουργούμε ένα έγγραφο XML. Επίσης, όπως μία από τις εργασίες του μεταγλωττιστή για κάθε γλώσσα προγραμματισμού είναι να ελέγξει αν η προδιαγραφές ακολουθήθηκαν, με παρόμοιο τρόπο υπάρχουν XML parsers οι οποίοι χρησιμοποιούν το DTD για να ελέγξουν την εγκυρότητα ενός εγγράφου XML[5].

Ένα DTD μας βοηθάει να καθορίσουμε τη δομή ενός εγγράφου XML. Μας παρέχει ένα αυστηρό πλαίσιο και κανόνες οι οποίοι θα ακολουθηθούν όταν δημιουργούμε έγγραφα XML. Επιπρόσθετα, το DTD μπορεί να χρησιμοποιηθεί για τον έλεγχο της εγκυρότητας και της ακεραιότητας των δεδομένων που περιέχονται σε ένα έγγραφο XML.

Μερικά χαρακτηριστικά του DTD είναι τα παρακάτω[5]:

- Το DTD χρησιμοποιείται για να καθορίσει έγκυρα στοιχεία και ιδιότητες που μπορούν να χρησιμοποιηθούν σε ένα έγγραφο XML.
- Με ένα DTD μπορούμε να καθορίσουμε μια ιεραρχική δομή στοιχείων.
- Σε ένα DTD μπορεί επίσης να καθοριστεί η διαδοχική οργάνωση μιας συλλογής στοιχείων-παιδιών τα οποία μπορούν να υπάρχουν σε ένα έγγραφο XML.

Ένα DTD μπορεί να χρησιμοποιηθεί απευθείας μέσα σε ένα έγγραφο XML ή μπορεί να υπάρχει εκτός του εγγράφου XML. Στη δεύτερη περίπτωση θα αναφέρεται με ένα δεσμό μέσα στο έγγραφο XML που δείχνει σε αυτό το DTD.

Βασικά το DTD αποτελείται από τα παρακάτω στοιχεία[5]:

Στοιχείο	Περιγραφή
DTD Element	Μεταδεδομένα για ένα στοιχείο. Καθορίζει τι είδους δεδομένα θα έχει το στοιχείο, τον αριθμό των περιστατικών κάθε στοιχείου, τις σχέσεις μεταξύ των στοιχείων και ούτω καθ'εξής.
DTD Attributes	Καθορίζει διάφορους κανόνες και ορισμούς που σχετίζονται με τα δεδομένα.
DTD Entities	Χρησιμοποιείται για να αναφέρει ένα εξωτερικό αρχείο ή για να παρέχει συντομεύσεις σε κοινό κείμενο.

Παράδειγμα : <!ELEMENT employee(shift+,home-address, hobbies*)>
Ένα στοιχείο **employee** μπορεί να περιέχει ένα ή περισσότερα στοιχεία **shift** και πρέπει να έχει ένα στοιχείο **home-address** και μπορεί να έχει μηδέν ή περισσότερα στοιχεία **hobbies**.

Παράδειγμα : <!ATTLIST shift id CDATA #REQUIRED>
Το στοιχείο **shift** πρέπει να έχει μία ιδιότητα **id**.

Εν ολίγοις, ένα DTD χρησιμοποιείται για να καθορίσει μια δομή εγγράφων με τη διευκρίνιση των λεπτομερειών σχετικά με όλα τα στοιχεία και τις ιδιότητες που πρόκειται να χρησιμοποιηθούν σε ένα έγγραφο XML. Ως εκ τούτου μπορεί να χρησιμοποιηθεί για να ελέγξει την εγκυρότητα ενός εγγράφου XML που υποτίθεται ότι ακολουθεί τους κανόνες που καθορίζονται από αυτό το DTD[5].

3.7 XML Schema

Το XML Schema είναι μια πιο προηγμένη έκδοση του DTD. Το DTD έχει πολλά μειονεκτήματα σε σχέση με το schema, όπως το ότι δεν υποστηρίζει ισχυρούς τύπους δεδομένων, έχει σύνταξη διαφορετική από την XML και δεν είναι επεκτάσιμο. Το XML Schema παρουσιάστηκε για να υπερνικήσει αυτά τα μειονεκτήματα[5].

Οι δύο κύριοι στόχοι του W3c XML Schema working group κατά τη διάρκεια του σχεδιασμού του προτύπου του XML Schema ήταν[4]:

- Να μπορέσουν να εκφράσουν μέσα στο πρότυπο αρχές αντικειμενοστραφούς σχεδιασμού οι οποίες μπορούν να βρεθούν σε όλες τις αντικειμενοστραφείς γλώσσες προγραμματισμού.
- Να παρέχουν υποστήριξη για σύνθετους τύπους δεδομένων παρόμοια με την υποστήριξη που υπάρχει στις περισσότερες σχεσιακές βάσεις δεδομένων.

Τα κυριότερα χαρακτηριστικά του XML Schema είναι τα παρακάτω[5]:

- Η σύνταξη είναι όμοια με της XML. Αυτό σημαίνει ότι μπορούμε να επεξεργαστούμε το schema με οποιοδήποτε επεξεργαστή XML.
- Δεν καθορίζουμε μόνο βασικούς τύπους δεδομένων όπως αλφαρηθμητικό, ακέραιος, πραγματικός και ούτω καθ'εξής αλλά μπορούμε επίσης να καθορίσουμε δικούς μας τύπους δεδομένων. Για παράδειγμα:

  <xs:element name="name" type="xs:string" />

  Οι νέοι τύποι που μπορούμε να καθορίσουμε μπορεί να είναι απλοί ή σύνθετοι. Οι σύνθετοι τύποι μπορεί να περιέχουν και άλλα στοιχεία ή και ιδιότητες, ενώ οι απλοί τύποι όχι. Αντίθετα μπορούν να περιέχουν μόνο δεδομένα.

- Το XML Schema παρέχει επικύρωση βασισμένη στο περιεχόμενο (content-based validation) δηλαδή μπορεί να ορίσει την σειρά με την οποία τα στοιχεία-παιδιά εμφανίζονται. Επίσης παρέχει επικύρωση στους ίδιους τους τύπους δεδομένων.

  Για παράδειγμα μπορούμε να ορίσουμε έναν απλό τύπο "year" με τιμές μεταξύ 2000 και 2100:

```
<xsd:simpleType name="year">
        <xsd:restriction base="xsd:integer">
                <xsd:minInclusive value="2000"/>
                <xsd:maxInclusive value="2100"/>
    </xsd:restriction>
</ xsd:simpleType>
```

  Παρομοίως οι σύνθετοι τύποι μπορεί να ορίσουν τη σειρά με την οποία τα στοιχεία-παιδιά θα εμφανίζονται:

```
<xsd:complexType name="Employee">
        <xsd:sequence>
                <xsd:element name="Name" type="xsd:string" />
                <xsd:element name="Address" type="xsd:string" />
                <xsd:element name="Phone" type="xsd:string" />
        </sd:sequence>
</xsd:complexType>
```

- Το XML Schema μας παρέχει τη δυνατότητα να επεκτείνουμε άλλα έγγραφα το οποίο δεν είναι τίποτα άλλο παρά κληρονομικότητα. Αυτό σημαίνει ότι μπορούμε να παράγουμε νέους τύπους δεδομένων βάσει παλαιών τύπων.
- Το XML Schema παρέχει υποστήριξη για Namespaces (χρησιμοποιώντας URI). Παρέχει σε κάθε στοιχείο ένα μοναδικό

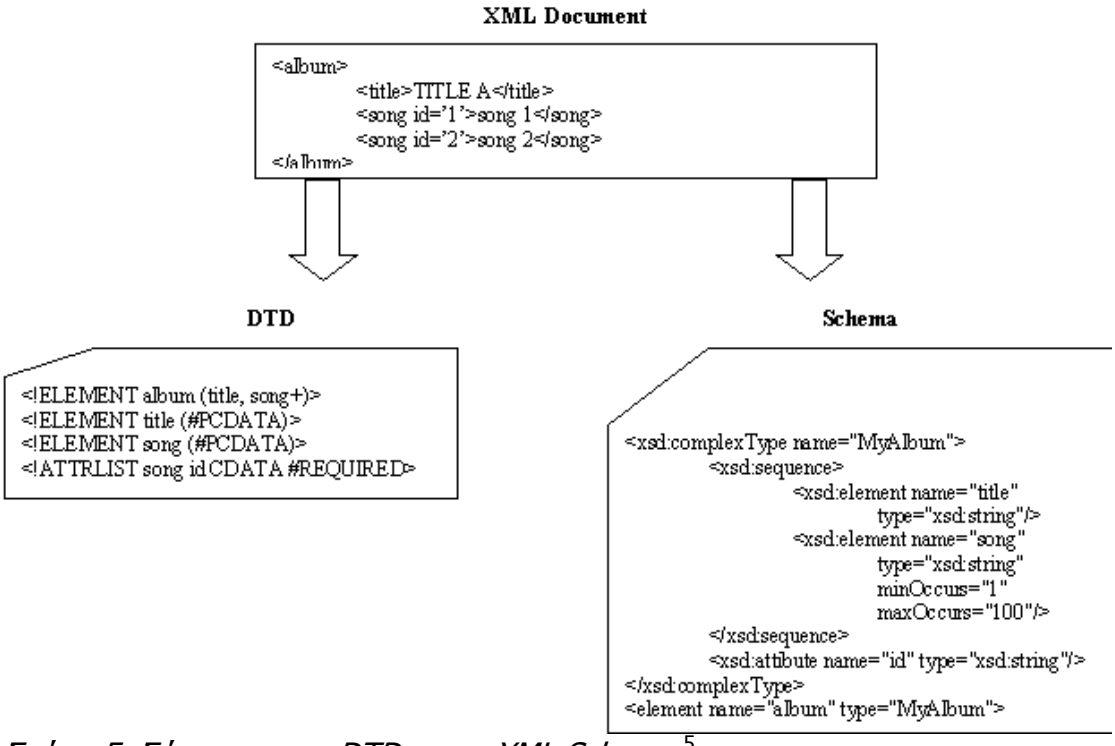
αναγνωριστικό, με το οποίο αποφεύγονται συγκρούσεις ονομάτων μεταξύ των στοιχείων. Αυτό θα μπορούσε να συμβεί, για παράδειγμα, όταν δύο έγγραφα συγχωνεύονταν, και περιείχαν και τα δύο στοιχεία με όνομα "name" τα οποία όμως είχαν διαφορετικό νόημα σε κάθε έγγραφο. Εν ολίγοις βοηθάει στο να ξεχωρίζουν στοιχεία και ιδιότητες με ίδιο όνομα και διαφορετικό νόημα. Μία απλή αναλογία στις γλώσσες προγραμματισμού είναι η χρήση καθολικών και τοπικών μεταβλητών. Μια τοπική μεταβλητή είναι μοναδική μέσα στο πεδίο ισχύος της ενώ μια καθολική μεταβλητή πρέπει να είναι μοναδική σε ολόκληρο το πρόγραμμα. Παρομοίως, με το namespace έχουμε την ελευθερία να να ορίσουμε τύπους χωρίς να ανησυχούμε για συγκρούσεις ονομάτων[4].

- Τέλος το XML Schema είναι εύκολα επεκτάσιμο για να ενσωματώσει και άλλες λειτουργίες στο μέλλον.

3.8 Σύγκριση του XML Schema με το DTD

Οι κυριότερες διαφορές συνοψίζονται ως εξής[5]:

- Το XML Schema είναι επέκταση του DTD.
- Το XML Schema υποστηρίζει namespaces ενώ το DTD όχι.
- Το XML Schema χρησιμοποιεί σύνταξη XML η οποία είναι εύκολη να την κατανοήσεις ενώ το DTD χρησιμοποιεί ειδική σύνταξη.
- Το XML Schema υποστηρίζει πρότυπους τύπους δεδομένων καθώς επίσης και τύπους ορισμένους από το χρήστη (user-defined) ενώ το DTD παρέχει μόνο τύπους κειμένου.
- Το XML Schema υποστηρίζει κληρονομικότητα ενώ το DTD όχι.



Σχήμα 5. Σύγκριση του DTD με το XML Schema [5]

3.9 Παράδειγμα XML Schema

Ας δούμε ένα παράδειγμα ενός XML Schema για τον καθορισμό της δομής ενός εγγράφου XML στο οποίο θα τηρούμε πληροφορίες για ένα βιβλιοπωλείο (BookStore) το οποίο περιέχει πολλά βιβλία (Book)[4]:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.books.org"
                xmlns=http://www.books.org>  A
<xsd:element name="BookStore">  B
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="Book">  C
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="Title" type="xsd:string"/>  D
<xsd:element name="Author" type="xsd:string"/>
<xsd:element name="Date" type="xsd:string"/>
<xsd:element name="ISBN" type="xsd:string"/>
<xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

Είναι εύκολο να καταλάβουμε τα αναγνωριστικά. Ένα στοιχείο αναπαριστάται με την ετικέτα "element". Η ετικέτα "comlexType" μας παρέχει το μηχανισμό με τον οποίο μπορούμε να ορίσουμε σύνθετους τύπους δεδομένων.

Για εύκολη αναφορά στο παραπάνω έγγραφο XML, το έγγραφο έχει χαρακτηριστεί σε διάφορα τμήματα. Στο τμήμα A του εγγράφου, η πρώτη γραμμή περιέχει μια τιμή "xsd" σαν ένα namespace. Κάθε αναφορά σε namespace απεικονίζεται χρησιμοποιώντας αυτή την τιμή σαν πρόθεμα και η σύνταξη για να αναφερθούμε σε ένα στοιχείο είναι namespace:elementname (για παράδειγμα xsd:element). Η ιδιότητα targetNamespace καθορίζει το namespace στο οποίο θα ανήκουν οι νέοι καθορισμένοι τύποι. Στο παραπάνω παράδειγμα, οι καθορισμένοι τύποι BookStore, Book, Title, Author, ISBN και Publisher ανήκουν στο namespace http://www.books.org.

Η ιδιότητα xmlns ορίζει τον προεπιλεγμένο namespace. Με άλλα λόγια, από αυτή τη θέση και έπειτα αν βρεθεί κάποιο στοιχείο στο οποίο δεν ορίζεται πρόθεμα για namespace τότε αυτό ανήκει στο default namespace.

Στο τμήμα B ένας νέος τύπος BookStore, ορίζεται σαν complexType (σύνθετος τύπος), ο οποίος περιέχει μια sequence (ακολουθία) από στοιχεία τύπου Book. Ένας complexType χρησιμοποιείται για να ορίσουμε ένα τύπο καθορισμένο από τον χρήστη, ο οποίος μπορεί να περιέχει πολλαπλά στοιχεία και ιδιότητες. Το τμήμα C ορίζει τον τύπο Book, ο οποίος είναι ο ίδιος ένας σύνθετος τύπος και περιέχει μια ακολουθία από τα στοιχεία Title, Author, Date, ISBN και Publisher. Οι ιδιότητες minOccur και maxOccur καθορίζουν τον περιορισμό στην εμφάνιση των στοιχείων. Η σειρά των στοιχείων που εμφανίζονται μέσα σε ένα στοιχείο Book πρέπει να ακολουθεί τη σειρά της προδιαγραφής. Με άλλα λόγια, ο τύπος Book θα περιέχει ένα στοιχείο Title ακολουθούμενο από ένα στοιχείο Author και ούτω καθ'εξής.

Το τμήμα D ορίζει τους τύπους των στοιχείων του Book. Κάθε ένα από αυτά τα στοιχεία είναι ένας simpleType (απλός τύπος) string.

Μπορούμε να ορίσουμε δικούς μας απλούς τύπους, αλλά ένας απλός τύπος δεν αποτελείται από στοιχεία ή ιδιότητες. Ένας απλός τύπος συνήθως καθορίζεται για να αναπαραστήσει ένα περιορισμό σε ένα βασικό τύπο. Για παράδειγμα, παρακάτω έχουμε τον ορισμό ενός απλού τύπου με όνομα elevation ο οποίος μπορεί να έχει έγκυρες τιμές μεταξύ 50 και 12000.

```
<xsd:simpleType name="elevation">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="50"/>
    <xsd:maxInclusive value="12000"/>
  </xsd:restriction>
</xsd:simpleType>
```

Έχοντας εξετάσει το έγγραφο του XML Schema παραπάνω, ας δούμε ένα έγγραφο XML που ακολουθεί το παραπάνω schema.

```
<?xml version="1.0"?>
<BookStore xmlns ="http://www.books.org"  1
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  2
        xsi:schemaLocation="http://www.books.org/
                        BookStore.xsd">  3
   <Book>  4
         <Title>Web Services Security</Title>
         <Author>Ravi Trivedi</Author>
         <Date>Dec, 2002</Date>
         <ISBN>1861007655</ISBN>
         <Publisher>Wrox Publishing</Publisher>
      </Book>
</BookStore>
```

Ο πρώτος ορισμός του xmlns χρησιμοποιεί τη δήλωση του προεπιλεγμένου namespace, και αυτό υπαγορεύει στον ελεγκτή του schema ότι όλα τα

στοιχεία που χρησιμοποιούνται σε αυτό το έγγραφο έρχονται από το namespace Book.
Ο δεύτερος ορισμός λέει στον ελεγκτή του schema ότι η ιδιότητα schemaLocation έρχεται από το namespace XMLSchema.
Ο τρίτος ορισμός, με το schemaLocation, λέει στον ελεγκτή ότι το namespace http://www.books.org είναι ορισμένο στο BookStore.xsd. Αυτό το αρχείο θα ανατρεχτεί από τον ελεγκτή του schema για να επικυρώσει την ορθότητα του τρέχοντος εγγράφου XML.
Τέλος, στον τέταρτο ορισμό, δηλώνεται ένα στοιχείο Book το οποίο περιέχει τις λεπτομέρειες ενός βιβλίου. Άξιο προσοχής είναι το γεγονός ότι η σειρά των στοιχείων-παιδιών του διατηρείται όπως στον ορισμό του complexType Book.

Ένα από τα μεγαλύτερα πλεονεκτήματα του XML Schema είναι ότι χρησιμοποιεί XML για τη σύνταξή του. Χρησιμοποιώντας XML, υπάρχοντες XML parsers μπορούν να χρησιμοποιηθούν σε συνδυασμό με ελεγκτές schema για να παρέχουν υπηρεσίες ελέγχου καλής διαμόρφωσης (well-formedness) και επικύρωσης (validation).

Το XML Schema προσφέρει έναν αυτοματοποιημένο μηχανισμό για επικύρωση των εγγράφων XML. Έχει παρατηρηθεί ότι σε τυπικό πρόγραμμα, μέχρι και 60% του κώδικα ξοδεύεται σε έλεγχο δεδομένων. Αν τα δεδομένα μας είναι δομημένα σαν XML, και υπάρχει και ένα schema, μπορούμε να περάσουμε τον έλεγχο των δεδομένων σε ένα ελεγκτή schema (schema validator). Κατά συνέπεια, μπορούμε να μειώσουμε τον κώδικά μας μέχρι και 60%. Επίσης, την επόμενη φορά που θα αλλάξουν οι περιορισμοί των δεδομένων μας ή προσθέσουμε και άλλα στοιχεία, δεν θα χρειαστεί να γράψουμε νέο κώδικα για τον έλεγχο των δεδομένων μας αλλά απλώς να αλλάξουμε το schema.

Η προδιαγραφή του XML Schema παίζει σημαντικό ρόλο στο σχεδιασμό και την υλοποίηση των web services. Τα αρχεία WSDL είναι επίσης κατασκευασμένα χρησιμοποιώντας τη σύνταξη του XML Schema.

4 WSDL και UDDI

4.1 Ορισμός και σκοπός της WSDL

Ο ορισμός που δίνει το W3C για την Web Services Description Language είναι ο παρακάτω[9]:

"*Η WSDL είναι ένα σχήμα XML για την περιγραφή δικτυακών υπηρεσιών σαν ένα σύνολο από τελικά σημεία που λειτουργούν σε μηνύματα τα οποία περιέχουν πληροφορία είτε προσανατολισμένη στα έγγραφα είτε πρσανατολισμένη στις διαδικασίες.*
Οι λειτουργίες και τα μηνύματα περιγράφονται περιληπτικά, και τότε δένονται σε ένα συγκεκριμένο πρωτόκολλο δικτύων και μορφή μηνυμάτων για να καθορίσουν ένα τελικό σημείο. Πολλά σχετικά τελικά σημεία συνδυάζονται σε υπηρεσίες (services).
Η WSDL είναι επεκτάσιμη στο να επιτρέπει την περιγραφή τελικών σημείων και των μηνυμάτων τους άσχετα από τη μορφή των μηνυμάτων και των πρωτοκόλλων δικτύων που χρησιμοποιούνται για την επικοινωνία. Παρόλα αυτά, αυτή τη στιγμή στην προδιαγραφή της WSDL οι μόνες συνδέσεις που περιλαμβάνονται περιγράφουν πώς μπορούμε να χρησιμοποιήσουμε την WSDL σε συνδυασμό με το SOAP 1.1, το HTTP GET/POST και το MIME."

Με πιο απλά λόγια η WSDL μας βηθάει να περιγράψουμε ένα σύνολο από μηνύματα και το πώς αυτά τα μηνύματα ανταλλάσσονται.
Για να καταλάβουμε την αξία της WSDL, μπορούμε να υποθέσουμε ότι θέλουμε να καλέσουμε ένα web service μέσω SOAP το οποίο μας παρέχεται από έναν συνεργάτη μας. Θα μπορούσαμε να του ζητήσουμε μερικά παραδείγματα μηνυμάτων SOAP και να γράψουμε κώδικα στην εφαρμογή μας για να παράγουμε και να καταναλώνουμε μηνύματα SOAP που μοιάζουν με αυτά τα παραδείγματα. Κάτι τέτοιο όμως είναι επιρρεπές σε λάθη. Για παράδειγμα, θα βλέπαμε ένα ID πελάτη με την τιμή 2837 και θα υποθέταμε ότι είναι ακέραιος ενώ στην πραγματικότητα θα ήταν αλφαρηθμητικό. Η WSDL καθορίζει τι πρέπει να περιέχει ένα μήνυμα και πώς πρέπει να είναι ένα μήνυμα απάντησης με σαφή σήμανση[2].

Η σήμανση που χρησιμοποιείται σε ένα αρχείο WSDL για να περιγράψει μορφές μηνυμάτων βασίζεται στο πρότυπο του XML Schema το οποίο σημαίνει ότι είναι ταυτόχρονα ανεξάρτητη από γλώσσα προγραμματισμού και βασισμένη σε πρότυπα. Αυτό το γεγονός την κάνει κατάλληλη για να περιγράψει διεπαφές web services οι οποίες είναι προσβάσιμες από μία μεγάλη ποικιλία πλατφορμών και γλωσσών προγραμματισμού. Επιπλέον, εκτός του ότι περιγράφει τα περιεχόμενα των μηνυμάτων, η WSDL ορίζει πού είναι διαθέσιμη μία υπηρεσία και ποιά πρωτόκολλα επικοινωνίας χρησιμοποιούνται για να επικοινωνήσουμε με αυτή την υπηρεσία. Αυτό σημαίνει ότι ένα αρχείο WSDL ορίζει όλα όσα χρειάζονται για να γράψουμε ένα πρόγραμμα το οποίο να λειτουργεί με ένα web service[2].

4.2 Τεχνική περιγραφή της WSDL

Η WSDL παρέχει ένα τρόπο στους παροχείς υπηρεσιών να περιγράψουν τη βασική μορφή των αιτήσεων και απαντήσεων των υπηρεσιών πάνω από διαφορετικά πρωτόκολλα και κωδικοποιήσεις.
Η WSDL χρησιμοποιείται για να περιγράψει ***τί*** μπορεί να κάνει ένα web service, ***πού*** βρίσκεται και ***πώς*** να το καλέσει κανείς.
Οι κατάλογοι UDDI περιγράφουν πολλές πτυχές των web services, συμπεριλαμβανομένων και των λεπτομερειών σύνδεσης μίας υπηρεσίας. Η WSDL ταιριάζει απόλυτα σε μια τέτοια περιγραφή μιας υπηρεσίας του UDDI[10].

Η WSDL ορίζει υπηρεσίες σαν συλλογές από τελικά σημεία δικτύου ή αλλιώς ***ports***. Στην WSDL ο περιγραφικός ορισμός των τελικών σημείων και των μηνυμάτων διαχωρίζεται από συγκεκριμένα διακτυακά πρωτόκολλα ή μορφές δεδομένων. Αυτό επιτρέπει την επαναχρησιμοποίηση των περιγραφικών ορισμών : των μηνυμάτων (***messages***).
Τα μηνύματα είναι αόριστες περιγραφές των δεδομένων που ανταλλάσσονται και των τύπων τελικών σημείων (***port types***).
Οι τύποι τελικών σημείων είναι συλλογές λειτουργιών.
Το συγκεκριμένο πρωτόκολλο και ο ορισμός της μορφής των δεδομένων για ένα συγκεκριμένο τύπο τελικών σημείων δημιουργεί μία επαναχρησιμοποιούμενη σύνδεση (***binding***).
Ένα τελικό σημείο (***port***) ορίζεται συνδέοντας μια διεύθυνση δικτύου με μία επαναχρησιμοποιούμενη σύνδεση (binding), και μία συλλογή τελικών σημείων ορίζουν μία υπηρεσία (***service***).

Ως εκ τούτου, ένα έγγραφο WSDL χρησιμοποιεί τα παρακάτω στοιχεία για τον ορισμό δικτυακών υπηρεσιών[10] :

- ***Types -*** ένα περίβλημα για ορισμούς τύπων δεδομένων χρησιμοποιώντας ένα σύστημα τύπων (όπως για παράδειγμα το XML Schema).
- ***Message -*** ένας περιγραφικός ορισμός των δεδομένων που ανταλλάσσονται.
- ***Operation -*** μία περιγραφή μίας λειτουργίας που υποστηρίζεται από μία υπηρεσία
- ***Port Type -*** ένα περιγραφικό σύνολο από λειτουργίες που υποστηρίζονται από ένα ή περισσότερα τελικά σημεία.
- ***Binding -*** ένα συγκεκριμένο πρωτόκολλο και μορφή δεδομένων για ένα συγκεκριμένο τύπο τελικών σημείων (port type).
- ***Port -*** ένα μοναδικό τελικό σημείο που ορίζεται σαν συνδυασμός μίας σύνδεσης (binding) και μιας διεύθυνσης δικτύου.
- ***Service*** - μία συλλογή από σχετικά τελικά σημεία.

4.3 Παράδειγμα WSDL

Το παρακάτω παράδειγμα είναι ένα έγγραφο WSDL που περιγράφει μια υπηρεσία για τις τιμές μετοχών[10]:

```
<?xml version="1.0"?>
<definitions name="StockQuote"

targetNamespace="http://example.com/stockquote.wsdl"
        xmlns:tns="http://example.com/stockquote.wsdl"
        xmlns:xsd1="http://example.com/stockquote.xsd"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns="http://schemas.xmlsoap.org/wsdl/">

   <types>
     <schema targetNamespace="http://example.com/stockquote.xsd"
          xmlns="http://www.w3.org/2000/10/XMLSchema">
       <element name="TradePriceRequest">
         <complexType>
            <all>
               <element name="tickerSymbol" type="string"/>
            </all>
         </complexType>
       </element>
       <element name="TradePrice">
         <complexType>
            <all>
               <element name="price" type="float"/>
            </all>
         </complexType>
       </element>
     </schema>
   </types>

   <message name="GetLastTradePriceInput">
      <part name="body" element="xsd1:TradePriceRequest"/>
   </message>

   <message name="GetLastTradePriceOutput">
      <part name="body" element="xsd1:TradePrice"/>
   </message>

   <portType name="StockQuotePortType">
      <operation name="GetLastTradePrice">
        <input message="tns:GetLastTradePriceInput"/>
        <output message="tns:GetLastTradePriceOutput"/>
      </operation>
   </portType>

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

```
            <soap:body use="literal"/>
          </output>
        </operation>
    </binding>

    <service name="StockQuoteService">
        <documentation>My first service</documentation>
        <port name="StockQuotePort" binding="tns:StockQuoteBinding">
          <soap:address location="http://example.com/stockquote"/>
        </port>
    </service>

</definitions>
```

Και ακολουθεί ένα μία αίτηση σε SOAP και η απάντηση της υπηρεσίας:

Αίτηση SOAP μέσω HTTP

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml;
charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
    <m:GetLastTradePrice
     xmlns:m="Some-URI">
      <symbol>MOT</symbol>
    </m:GetLastTradePrice>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Απάντηση της υπηρεσίας με SOAP μέσω HTTP

```
HTTP/1.1 200 OK Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
 <SOAP-ENV:Body>
  <m:GetLastTradePriceResponse
   xmlns:m="Some-URI">
   <Price>14.5</Price>
  </m:GetLastTradePriceResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

4.4 Περιγραφή και σκοπός του UDDI

Ο σκοπός που δίνει το OASIS για το Universal Description, Discovery and Integration είναι ο παρακάτω[11]:

"*Τα web services έχουν νόημα μόνο όταν δυνητικοί χρήστες μπορούν να βρουν πληροφορίες ικανές ώστε να επιτρέψουν την εκτέλεσή τους. Το Universal Description Discovery & Integration (UDDI) εστιάζει στον καθορισμό ενός συνόλου από υπηρεσίες που θα υποστηρίζουν την περιγραφή και την ανακάλυψη :*

1) *Των εταιριών, των οργανισμών και άλλων παρόχων web services*
2) *Των web services, που είναι διαθέσιμες*
3) *Και των τεχνικών διεπαφών οι οποίες μπορούν να χρησιμοποιηθούν ώστε να έχει κάποιος χρήστης πρόσβαση σε αυτές τις υπηρεσίες.*

Βασισμένο σε ένα κοινό σύνολο από βιομηχανικά πρότυπα, συμπεριλαμβανομένων των HTTP, XML, XML Schema και SOAP το UDDI παρέχει μία διαλειτουργική, θεμελιώδη υποδομή για ένα περιβάλλον λογισμικού προσανατολισμένο στις υπηρεσίες τόσο για δημόσια διαθέσιμες υπηρεσιές όσο και για υπηρεσίες που εκτίθενται μόνο εσωτερικά ενός οργανισμού."

Με πιο απλά λόγια το UDDI είναι ο «χρυσός οδηγός» των web services. Όπως σε ένα χρυσό οδηγό, μπορούμε να αναζητήσουμε μια εταιρία που προσφέρει τις υπηρεσίες που χρειαζόμαστε, να διαβάσουμε για μια προσφερόμενη υπηρεσία και να επικοινωνήσουμε με κάποιον για περισσότερες λεπτομέρειες. Φυσικά μπορούμε να προσφέρουμε ένα web service χωρίς να το καταχωρίσουμε στο UDDI, όπως αν ανοίγαμε μία επιχείρηση στο υπόγειο του σπιτιού μας και βασιζόμασταν στη διαφήμιση από στόμα σε στόμα. Αλλά αν θέλουμε να αγγίξουμε το ευρύ κοινό, θα χρειαστούμε το UDDI ώστε οι δυνητικοί μας πελάτες να μπορέσουν να μας βρούν[2].

4.5 Υπηρεσίες του UDDI

Το UDDI παρέχει ένα μηχανισμό στους πελάτες να βρίσκουν δυναμικά άλλα web services. Χρησιμοποιώντας μία διεπαφή UDDI, οι επιχειρήσεις μπορούν να συνδεθούν δυναμικά με υπηρεσίες που παρέχονται από εξωτερικούς συνεργάτες. Ένας κατάλογος UDDI (UDDI registry) είναι όμοιος μέ ένα CORBA trader, ή θα μπορούσαμε να τον παρομοιάσουμε με μία υπηρεσία DNS αλλά για εφαρμογές.
Ένας κατάλογος UDDI έχει δύο ειδών πελάτες : επιχειρήσεις που θέλουν να δημοσιεύσουν μια υπηρεσία (και τις διεπαφές της), και πελάτες που θέλουν να χρησιμοποιήσουν συγκεκριμένες υπηρεσίες και συνδέονται προγραμματιστικά με αυτές[10].

Ο παρακάτω πίνακας περιγράφει περιληπτικά τις προσφερόμενες υπηρεσίες του UDDI[10]:

Πληροφορία	**Λειτουργίες**	**Λεπτομέρειες (που υποστηρίζονται από το API)**
White pages: Πληροφορίες όπως το όνομα, η διεύθυνση, το τηλέφωνο και άλλες πληροφορίες επικοινωνίας για μία επιχείρηση.	**Publish**: Πώς ο προμηθευτής ενός web service καταχωρεί των εαυτό του.	**Business Information**: Περιλαμβάνεται σε ένα αντικείμενο ***BusinessEntity***, το οποίο με τη σειρά του περιλαμβάνει πληροφορίες για υπηρεσίες, κατηγορίες, επαφές, URLs, και άλλα αναγκαία στοιχεία για να αλληλεπιδράσουμε με μία επιχείρηση.
Yellow Pages: Πληροφορίες που κατηγοριοποιούν επιχειρήσεις. Βασίζονται σε υπάρχοντα πρότυπα κατηγοριοποίησης (μη ηλεκτρονικά).	**Find**: Πώς μία εφαρμογή βρίσκει ένα συγκεκριμένο web service.	**Service Information**: Περιγράφει μία ομάδα από web services. Αυτές περιλαμβάνονται σε ένα αντικείμενο ***BusinessService***.
Green Pages: Τεχνικές πληροφορίες για τα web services που παρέχονται από μία επιχείρηση.	**Bind**: Πώς μία εφαρμογή συνδέεται, και αλληλεπιδρά με ένα web service αφού αυτό βρεθεί.	**Binding Information**: Οι απαραίτητες τεχνικές λεπτομέρειες για την κλήση ενός web service. Περιλαμβάνουν τα URLs, πληροφορίες για ονόματα μεθόδων, τύπους ορισμάτων και ούτω καθ'εξής. Το αντικείμενο ***BindingTemplate*** αναπαριστά αυτά τα δεδομένα. **Service Specification Detail**: Πρόκειται για μεταδεδομένα για διάφορες προδιαγραφές που υλοποιούνται από ένα web service. Αυτά καλούνται ***tModels***.

Το UDDI αναμένεται να αποτελέσει τη βάση για υψηλότερου επιπέδου υπηρεσίες που θα υποστηρίζονται από άλλα πρότυπα.

4.6 Παράδειγμα UDDI

Στο παρακάτω παράδειγμα φαίνεται ένα ερώτημα σε SOAP και η απάντηση από ένα κατάλογο UDDI[10]:

Ερώτημα : το παρακάτω ερώτημα, τοποθετημένο σε ένα φάκελο SOAP, επιστρέφει λεπτομέρειες για τη Microsoft:

```
<find_business generic="1.0" xmlns="urn:uddi-org:api">
<name>Microsoft</name>
</find_business>
```

Αποτέλεσμα : Λεπτομερείς καταχωρήσεις για τη Microsoft, οι οποίες περιλαμβάνουν και πληροφορίες για το ίδιο το UDDI.

```
<businessList generic="1.0"
 operator="Microsoft Corporation"
 truncated="false"
 xmlns="urn:uddi-org:api">
 <businessInfos>
  <businessInfo
     businessKey="0076B468-EB27-42E5-AC09-9955CFF462A3">
   <name>Microsoft Corporation</name>
   <description xml:lang="en">
        Empowering people through great software -
        any time, any place and on any device is Microsoft's
        vision. As the worldwide leader in software for personal
        and business computing, we strive to produce innovative
        products and services that meet our customer's
   </description>
    <serviceInfos>
     <serviceInfo
       businessKey="0076B468-EB27-42E5-AC09-9955CFF462A3"
       serviceKey="1FFE1F71-2AF3-45FB-B788-09AF7FF151A4">
      <name>Web services for smart searching</name>
     </serviceInfo>
    <serviceInfo
       businessKey="0076B468-EB27-42E5-AC09-9955CFF462A3"
       serviceKey="8BF2F51F-8ED4-43FE-B665-38D8205D1333">
      <name>Electronic Business Integration Services</name>
    </serviceInfo>
    <serviceInfo
       businessKey="0076B468-EB27-42E5-AC09-9955CFF462A3"
       serviceKey="611C5867-384E-4FFD-B49C-28F93A7B4F9B">
            <name>Volume Licensing Select Program</name>
    </serviceInfo>
          <serviceInfo
       businessKey="0076B468-EB27-42E5-AC09-9955CFF462A3"
       serviceKey="A8E4999A-21A3-47FA-802E-EE50A88B266F">
            <name>UDDI Web Sites</name>
    </serviceInfo>
    </serviceInfos>
  </businessInfo>
 </businessInfos>
</businessList>
```

5 SOAP

5.1 Ορισμός και σκοπός του SOAP

Το W3C από τον Ιούνιο του 2003 έχει συντάξει μία σύσταση (recommendation) η οποία είναι ό,τι πιο κοντινό υπάρχει στην προδιαγραφή (specification) του Simple Object Access Protocol v1.2.
Μέσα στη σύσταση αυτή βρίσκεται ο παρακάτω ορισμός[12]:

"*Το SOAP στην έκδοση 1.2 είναι ένα ελαφρύ πρωτόκολλο προορισμένο για την ανταλλαγή δομημένων πληροφοριών σε ένα αποκεντρωμένο, διανεμημένο περιβάλλον. Χρησιμοποιεί τεχνολογίες XML για να καθορίσει ένα επεκτάσιμο πλαίσιο παρέχοντας μια δομή μηνυμάτων η οποία μπορεί να ανταλλαχθεί πάνω από ποικίλα δικτυακά πρωτόκολλα. Το πλαίσιο έχει σχεδιαστεί να είναι ανεξάρτητο από οποιοδήποτε προγραμματιστικό μοντέλο και σημασιολογία υλοποίησης.*"

Δύο βασικοί στόχοι του σχεδιασμού του SOAP είναι η απλότητα και η επεκτασιμότητα. Το SOAP προσπαθεί να πετύχει αυτούς τους στόχους παραλείποντας, από το πλαίσιο μηνυμάτων, χαρακτηριστικά γνωρίσματα τα οποία συνήθως συναντούνται σε κατανεμημένα συστήματα. Μερικά από αυτά τα γνωρίσματα είναι η «αξιοπιστία» (reliability),η «ασφάλεια» (security), ο «συσχετιμός» (correlation),η «δρομολόγηση» (routing) και τα «σχέδια ανταλλαγής μηνυμάτων» (Message Exchange Patterns - MPEs).
Ενώ αναμένεται ότι πολλά χαρακτηριστικά θα καθοριστούν, η παρούσα προδιαγραφή παρέχει τις λεπτομέρειες μόνο για δύο MEPs : το HTTP και το SMTP. Τα υπόλοιπα χαρκατηριστικά γνωρίσματα αφήνονται να καθοριστούν από άλλες προδιαγραφές[12].

Με πιο απλά λόγια το SOAP είναι ένα πρωτόκολλο βασισμένο στην XML το οποίο επιτρέπει στις εφαρμογές να ανταλλάσουν πληροφορία πάνω από κοινώς χρησιμοποιούμενα πρωτόκολλα του διαδικτύου.

5.2 SOAP και άλλα πρωτόκολλα ανταλλαγής μηνυμάτων

Πριν ανακαλυφθεί το SOAP είχαμε και άλλες κατανεμημένες τεχνολογίες τόσο για ανταλλαγή μηνυμάτων όσο και για απομακρυσμένη κλήση διαδικασιών (remote procedure call). Για να καταλάβουμε καλύτερα τα πλεονεκτήματα του SOAP καλό θα ήταν να εξετάσουμε, έστω και επιφανειακά, αυτές τις τεχνολογίες ώστε να καταλάβουμε τα πλεονεκτήματα και κύριως τα μειονεκτήματα της κάθε μίας τα οποία και μας οδήγησαν στη δημιουργία αυτού του πρωτοκόλλου[13]:

5.2.1 EDI

Το κυριότερο πρωτόκολλο ανταλλαγής μηνυμάτων στον χώρο των επιχειρήσεων ήταν και είναι ακόμη το Electronic Data Interchange (EDI) το οποίο όμως δε χρησιμοποιεί το διαδίκτυο αλλά ιδωτικά δίκτυα γεγονός που το καθιστά μια ακριβή τεχνολογία από άποψη υποδομής. Τώρα πλέον είναι πολύ φθηνότερο να δημιουργήσουμε ιδεατά ιδιωτικά δίκτυα (Virtual Private Networks - VPNs) πάνω από το διαδίκτυο για να ανταλλάσουμε μηνύματα EDI. Ένα ακόμη μειονέκτημα του EDI είναι ότι η μορφή των μηνυμάτων του είναι πολύ ειδική για κάθε βιομηχανία οπότε δεν είναι τόσο ευέλικτο όσο το SOAP. Το EDI θεωρείται μια δύσκολη και αρκετά ακριβή τεχνολογία για να την υλοποιήσει μια επιχείρηση.

5.2.2 Επικοινωνούντα αντικείμενα

Οι πιο σημαντικές σημαντικές τεχνολογίες, που δε βασίζονται στην XML, για το πέρασμα μηνυμάτων και για απομακρυσμένη κλήση διαδικασίας (remote procedure call) είναι οι CORBA και DCOM. Μερικοί σημαντικοί όροι που αφορούν και τις δύο τεχνολογίες είναι οι παρακάτω :

Interface Definition Language (IDL) : Η γλώσσα που χρησιμοποιείται για να καθορίσει την διεπαφή που ένα αντικείμενο χρησιμοποιεί για την επικοινωνία με τον έξω κόσμο.

Marshalling ή Serializing : Η διαδικασία μετατροπής της δομής ενός αντικειμένου ενός προγράμματος σε ένα ρεύμα από ψηφιολέξεις (bytes).

Unmarshalling ή Deserializing : Η διαδικασία μετατροπής ενός ρεύματος από ψηφιολέξεις στην αρχική δομή του αντικειμένου.

Encoding : Η μετατροπή των δεδομένων σε μία μορφή η οποία μπορεί να μεταφερθεί από ένα συγκεκριμένο πρωτόκολλο. Για παράδειγμα, για να περιλάβουμε τα δυαδικά δεδομένα που σχηματίζουν μία εικόνα σε ένα μήνυμα XML, τα bytes πρέπει να κωδικοποιηθούν σαν χαρακτήρες συμβατοί με τις ετικέτες της XML.

5.2.2.1 CORBA

Μία πρωτοποριακή προσπάθεια για να έχουμε αξιόπιστη επικοινωνία αντικειμένων μεταξύ διαφορετικών συστημάτων ήταν το Common Object Request Broker Architecture (CORBA). Το Object Management Group (OMG) ήθελε να δημιουργήσει ένα πρωτόκολλο ικανό να παρέχει επικοινωνία μεταξύ αντικειμένων σε εντελώς διαφορετικά λειτουργικά συστήματα και γραμμένα σε διαφορετικές γλώσσες προγραμματισμού. Ένα Object Request Broker (ORB) παρέχει τη αναγκαία διεπαφή ώστε ένα πρόγραμμα-πελάτης να μιλήσει με απομακρυσμένα αντικείμενα.
Λόγω του ότι η αρχική προδιαγραφή του CORBA δημοσιεύτηκε το 1992, είχε αρκετό καιρό για βελτίωση σε σχέση με άλλες τεχνολογίες. Πολλοί μεγάλοι

προμηθευτές λογισμικού έχουν δημιουργήσει προϊόντα για αυτή την τεχνολογία. Το CORBA θεωρείται μια δύσκολη κατανεμημένη τεχνολογία.

Το OMG δημιούργησε μια πρότυπη γλώσσα την Interface Description Language (IDL) η οποία επιτρέπει τον καθορισμό της διεπαφής ενός αντικειμένου σε μία μορφή ανεξάρτητη από γλώσσα προγραμματισμού.
Το OMG δημιούργησε επίσης και ένα πρωτόκολλο το
Internet Inter-ORB Protocol (IIOP) για επικοινωνία μέσω μηνυμάτων πάνω από το Internet. Το OMG τώρα εργάζεται για τη δημιουργία ενός προτύπου διεπαφής μεταξύ SOAP και CORBA.

5.2.2.2 COM και DCOM

Το Common Object Model (COM) είναι μια προδιαγραφή της Microsoft για ενοποίηση συστατικών (components integration) μέσα σε μια εφαρμογή. Η επικοινωνία είναι χαμηλού επιπέδου και επιτρέπει συστατικά γραμμένα σε διαφορετικές γλώσσες προγραμματισμού να αλληλεπιδρούν. Το Distributed COM (DCOM) είναι μια πιο πρόσφατη προσπάθεια που επιτρέπει συστατικά να αλληλεπιδρούν πάνω από ένα δίκτυο. Η μορφή επικοινωνίας είναι ουσιαστικά μια κλήση απομακρυσμένης διαδικασίας.
Τα COM και DCOM θεωρούνται όριμες τεχνολογίες και παρότι οι κύριοι χρήστες του DCOM είναι Microsoft Windows συστήματα θα μπορούσε να επεκταθεί και σε άλλα λειτουργικά συστήματα.

5.2.2.3 RMI

Η πιο απλή μορφή επικοινωνίας στη Java μεταξύ αντικειμένων σε κατανεμημένα συστήματα παρέχεται από το Remote Method Invocation (RMI). Το RMI επιτρέπει στους προγραμματιστές να χειριστούν ένα απομακρυσμένο αντικείμενο σαν να βρίσκεται μέσα στην τοπική εφαρμογή.

Αυτό που πρέπει να κάνει ο προγραμματιστής είναι να καθορίσει τη δημόσια διεπαφή που πρόκειται να εκτεθεί για ένα αντικείμενο. Με το βοηθητικό πρόγραμμα *rmic* εξετάζει τη διεπαφή και δημιουργεί κλάσεις γνωστές ως *stub* και *skeleton*.
Στην πλευρά του διακομιστή η εφαρμογή δηλώνει τη διεπαφή στο *rmiregistry* και περιμένει νέες συνδέσεις.
Μια εφαρμογή-πελάτης χρησιμοποιεί το *rmiregistry* το οποίο «ζει» στο δίκτυο σε μία πρότυπη πόρτα, για να λάβει ένα στιγμιότυπο της stub κλάσης η οποία υλοποιεί την επιθυμητή διεπαφή. Οι κλήσεις που γίνονται στο στιγμιότυπο της stub κλάσης συμπεριφέρονται σαν να βρίσκεται το απομαρκυσμένο αντικείμενο στην τοπική Java Virtual Machine.
Τις λεπτομέρειες του serializing και deserializing των αντικειμένων, και της επικοινωνίας μεταξύ των συστημάτων τις χειρίζονται οι κλάσεις stub και skeleton.

Το πιο προφανές πλεονέκτημα του RMI είναι η ευκολία χρήσης και η απόλυτη συμβατότητα των τύπων δεδομένων. Για τους προγραμματιστές, όλες τις λεπτομέρειες του serialization και deserialization τις αναλαμβάνουν οι κλάσεις του RMI. Από τη στιγμή που ένα αντικείμενο είναι serializable, μπορεί να χρησιμοποιηθεί σε μια απομακρυσμένη κλήση μεθόδου. Για αυτούς τους λόγους το RMI είναι η ουσιαστική τεχνολογία για κατανεμημένες εφαρμογές σε Java.

Παρόλα αυτά, το RMI έχει και μειονεκτήματα. Φυσικά μπορεί να χρησιμοποιηθεί μόνο μεταξύ εφαρμογών Java. Επίσης, και οι δύο πλευρές πρέπει να έχουν πρόσβαση στις κλάσεις της διεπαφής. Ακόμη ένα μειονέκτημα είναι ότι οι πόρτες (ports) που χρησιμοποιούνται συνήθως μπλοκάρονται από firewalls και proxy servers. Για αυτούς τους λόγους το RMI είναι περισσότερο επιτυχημένο στα τοπικά δίκτυα (intranets).

Για τη πλατφόρμα της Java 2, η Sun δημιούργησε εργαλεία ικανά να γεφυρώσουν το κενό ανάμεσα σε αντικείμενα RMI και αντικείμενα CORBA γραμμένα σε άλλες γλώσσες προγραμματισμού. Αυτό το πακέτο ονομάζεται *RMI-IIOP* και αν πρέπει να επικοινωνήσουμε με απομακρυσμένα αντικείμενα σε παλαιά συστήματα αυτή η επλογή είναι η πιο κατάλληλη.

5.2.3 XML-RPC

Το πρωτόκολλο XML-RPC δημιουργήθηκε από τον Dave Winer της UserLand το 1998 και είναι ο πρόδρομος του πρωτοκόλλου SOAP.
Το XML-RPC εκτελεί απομακρυσμένες κλήσεις διαδικασίας μέσω μιας αίτησης HTTP POST με κωδικοποίηση XML 1.
Χρησιμοποιώντας το ευρέως υποστηριζόμενο πρωτόκολλο HTTP κάνει δυνατή την προσθήκη επεξεργασίας του XML-RPC σε κάθε εξυπηρετητή ιστού που υποστηρίζει CGI (Common Gateway Interface).
Επίσης χρησιμοποιώντας αυτό το πρότυπο πρωτόκολλο επιτρέπει το πέρασμα των τυχών προστασίας, κάνοντας εύκολη την εγκατάσταση ενός εξυπηρετητή XML-RPC χωρίς συμβιβασμούς στην ασφάλεια.
Το XML-RPC είναι πολύ πιο απλό από το SOAP αλλά δεν έχει και τόσες δυνατότητες όσες το SOAP. Υποστηρίζει βασικούς τύπους δεδομένων αλλά και σύνθετους (μέσω των τύπων <struct> και <array>).
Η πρώτη προσπάθεια για τη δημιουργία του SOAP (έκδοση 1.1) είχε σαν βάση της αυτό το πρωτόκολλο.

5.3 Τι προσφέρει το SOAP

Αν εξετάσουμε τις παραπάνω τενολογίες θα δούμε ότι σε κάθε περίπτωση έχουμε κάποια από τα παρακάτω μειονεκτήματα:

- Κόστος υποδομής (EDI, CORBA)
- Κόστος και πολυπλοκότητα υλοποίησης (EDI, CORBA)
- Χρήση μόνο σε ιδιωτικά δίκτυα (EDI, CORBA, DCOM, RMI)
- Χρήση μόνο σε εφαρμογές που εκτελούνται στην ίδια πλατφόρμα (DCOM)
- Χρήση μόνο σε εφαρμογές γραμμένες στην ίδια γλώσσα προγραμματισμού (RMI)
- Μειωμένες λειτουργίες (XML-RPC)
- Έλειψη ευελιξίας (EDI, CORBA, DCOM, RMI)

Στην προσπάθεια δημιουργίας του SOAP λήφθηκαν υπόψιν όλα τα παραπάνω και δημιουργήθηκε ένα πρωτόκολλο με τα εξής χαρακτηριστικά:

- Το SOAP είναι απλό. Άρα το κόστος και η πολυπλοκότητα υλοποίησης μειώνονται αισθητά.
- Το SOAP είναι ανεξάρτητο από πλατφόρμα και γλώσσα προγραμματισμού οπότε μπορεί να χρησιμοποιηθεί για επικοινωνία μεταξύ εφαρμογών γραμμένων για διαφορετικές πλατφόρμες και σε διαφορετικές γλώσσες προγραμματισμού.
- Το SOAP είναι ευέλικτο. Χρησιμοποιεί πρότυπα πρωτόκολλα όπως το HTTP και το SMTP ως μέσα μεταφοράς οπότε μπορεί να χρησιμοποιηθεί στο διαδίκτυο και να διαπερνά τύχη προστασίας χωρίς συμβιβασμούς στην ασφάλεια της υποδομής μιας επιχείρησης. Αυτό αυτομάτως μειώνει και σε ορισμένες περιπτώσεις εξαλήφει το κόστος υποδομής αφού οι περισσότερες επιχειρήσεις σήμερα έχουν και τον εξοπλισμό και την τεχνογνωσία για τη χρήση του διαδικτύου.
- Το SOAP είναι επεκτάσιμο. Αν και δεν προσφέρει τόσες πολλές λειτουργίες όσο άλλες τεχνολογίες όπως το CORBA και το DCOM επιτρέπει σε άλλα πρότυπα να το επεκτείνουν παρέχοντας υπηρεσίες που λείπουν από αυτό. Αυτό το χαρακτηριστικό αποδείχθηκε ίσως το σημαντικότερο γιατί επάνω του βασίζονται πολλές αναπτυσσόμενες τεχνολογίες των web services που προσφέρουν υπηρεσίες όπως «αξιοπιστία» (reliability), «δρομολόγηση» (routing) και «ασφάλεια» (security).

5.4 Δομή ενός μηνύματος SOAP

Ένα μήνυμα SOAP είναι ένα συνηθισμένο έγγραφο XML το οποίο περιέχει τα παρακάτω στοιχεία (elements)[14]:

- Ένα απαιτούμενο στοιχείο ***Envelope*** από το οποίο αναγνωρίζεται το έγγραφο XML ως μήνυμα SOAP.
- Ένα προαιρετικό στοιχείο ***Header*** που περιλαμβάνει βοηθητικές πληροφορίες.
- Ένα απαιτούμενο στοιχείο ***Body*** το οποίο περιλαμβάνει την κύρια πληροφορία του μηνύματος.
- Ένα προαιρετικό στοιχείο ***Fault*** το οποίο παρέχει πληροφορίες για σφάλματα που προκλήθηκαν κατά την επεξεργασία ενός μηνύματος.

Όλα τα παραπάνω στοιχεία δηλώνονται στο προεπιλεγμένο namespace για το SOAP envelope :

http://www.w3.org/2003/05/soap-envelope

Το προεπιλεγμένο namespace για την κωδικοποίηση του SOAP και τους τύπους δεδομένων είναι ο :

http://www.w3.org/2001/12/soap-encoding

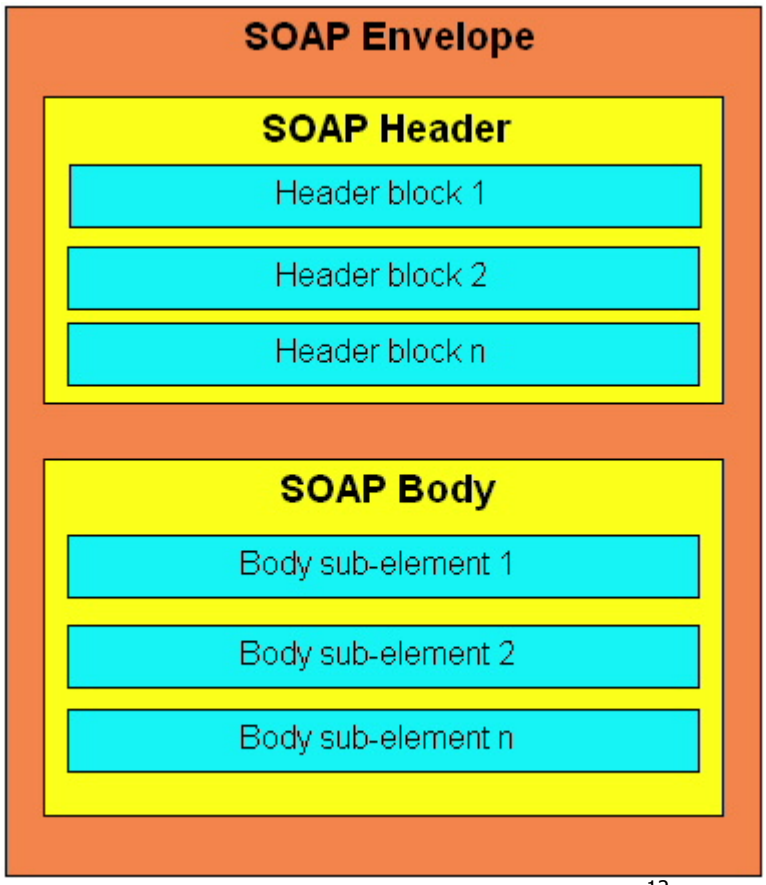
Μερικοί σημαντικοί κανόνες σύνταξης είναι οι παρακάτω[14]:

1. Ένα μήνυμα SOAP ΠΡΕΠΕΙ να είναι κωδικοποιημένο χρησιμοποιώντας XML.
2. Ένα μήνυμα SOAP ΠΡΕΠΕΙ να χρησιμοποιεί το SOAP Envelope namespace.
3. Ένα μήνυμα SOAP ΠΡΕΠΕΙ να χρησιμοποιεί το SOAP Encoding namespace.
4. Ένα μήνυμα SOAP ΔΕΝ ΠΡΕΠΕΙ να περιέχει αναφορά σε DTD.
5. Ένα μήνυμα SOAP ΔΕΝ ΠΡΕΠΕΙ να περιέχει XML Processing Instructions.

Ο σκελετός ενός μηνύματος SOAP φαίνεται παρακάτω[14]:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
soap:encodingStyle=" http://www.w3.org/2001/12/soap-encoding">
<soap:Header>
  ...
  ...
</soap:Header>
<soap:Body>
  ...
  ...
</soap:Body>
</soap:Envelope>
```

Σχηματικά θα μπορούσε να παρασταθεί ως εξής:

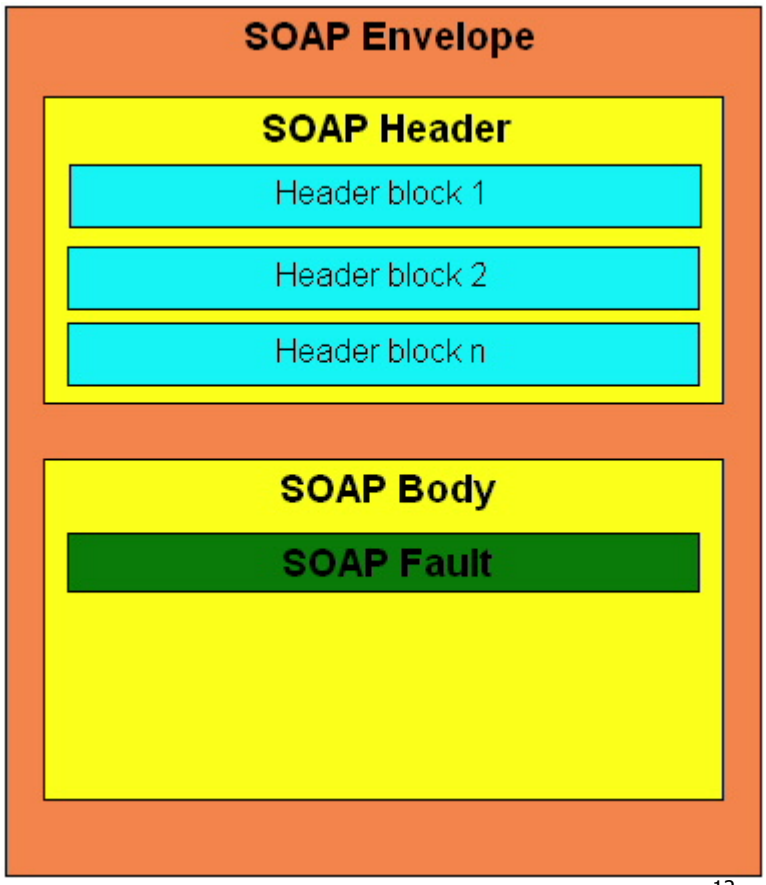


Σχήμα 7. Δομή ενός μηνύματος SOAP [12]

Όταν ένα μήνυμα SOAP περιέχει το στοιχείο Fault έχει την παρακάτω μορφή[14]:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
soap:encodingStyle=" http://www.w3.org/2001/12/soap-encoding">
<soap:Header>
  ...
  ...
</soap:Header>
<soap:Body>
  <soap:Fault>
    ...
    ...
  </soap:Fault>
</soap:Body>
</soap:Envelope>
```

Σχηματικά θα μπορούσε να παρασταθεί ως εξής:



Σχήμα 8. Δομή ενός μηνύματος SOAP με Fault [12]

5.5 SOAP Envelope [12]

Το SOAP Envelope είναι το στοιχείο που περικλείει όλα τα υπόλοιπα στοιχεία σε ένα μήνυμα SOAP. Η ύπαρξη του ως αρχικού στοιχείου (root element) ενός εγγράφου XML σηματοδοτεί ότι πρόκειται για μήνυμα SOAP.

Για το SOAP Envelope είναι απαραίτητα τα εξής:

- Το όνομα του στοιχείου πρέπει να είναι Envelope.
- Το όνομα του namespace του να είναι http://www.w3.org/2003/05/soap-envelope.
- Να περιέχει μηδέν ή περισσότερες ιδιότητες ορισμένες με namespace.
- Να περιέχει ένα ή δύο στοιχεία-παιδιά με την εξής σειρά :
    - Ένα προαιρετικό στοιχείο Header
    - Ένα απαραίτητο στοιχείο Body

5.6 SOAP – «ειδικές» ιδιότητες XML (XML attributes) [12]

Οι παρακάτω ιδιότητες XML έχουν ειδικό νόημα για την επεξεργασία ενός μηνύματος SOAP:

- ***encodingStyle*** : υποδεικνύει τους κανόνες κωδικοποίησης που χρησιμοποιούνται για το serialization των διάφορων τμημάτων ενός μηνύματος SOAP.
- ***role*** : χρησιμοποιείται για να υποδείξει τον κόμβο επεξεργασίας (SOAP node) για τον οποίο προορίζεται το SOAP header block.
- ***mustUnderstand*** : χρησιμοποιείται για να υποδείξει ότι η επεξεργασία ενός SOAP header block είναι υποχρεωτική ή προαιρετική.
- ***relay*** : χρησιμοποιείται για να υποδείξει αν το SOAP header block που προορίζεται για ένα κόμβο επεξεργασίας πρέπει να αναμεταδοθεί αν δεν υποστεί επεξεργασία.

5.7 SOAP Header [12]

Το SOAP Header παρέχει ένα μηχανισμό για επέκταση ενός μηνύματος SOAP με ένα αποκεντρωμένο και δομημένο τρόπο.

Για το SOAP Header είναι απαραίτητα τα εξής:

- Το όνομα του στοιχείου πρέπει να είναι Header.
- Το όνομα του namespace του να είναι "http://www.w3.org/2003/05/soap-envelope".
- Μπορεί να περιέχει μηδέν ή περισσότερες ιδιότητες ορισμένες με namespace.
- Μπορεί να περιέχει μηδέν ή περισσότερα στοιχεία-παιδιά ορισμένα με namespace.

Κάθε στοιχείο-παιδί του SOAP Header ονομάζεται *SOAP header block*.

5.7.1 SOAP header block

Για κάθε SOAP header block είναι απαραίτητα τα εξής:

- Το όνομα του στοιχείου πρέπει να είναι ορισμένο με namespace.
- Μπορεί να περιέχει μηδέν ή περισσότερους κόμβους-παιδιά χαρακτήρων.
- Μπορεί να περιέχει μηδέν ή περισσότερα στοιχεία-παιδιά τα οποία μπορεί να είναι ορισμένα με namespace.
- Μπορεί να περιέχει μηδέν ή περισσότερες ιδιότητες. Ανάμεσα σε αυτές μπορεί να υπάρχουν κάποιες ή όλες από τις παρακάτω που έχουν ιδιαίτερο νόημα για την επεξεργασία του μηνύματος SOAP:
    - Η ιδιότητα *encodingStyle*.
    - Η ιδιότητα *role*.
    - Η ιδιότητα *mustUnderstand*.
    - Η ιδιότητα *relay*.

5.8 SOAP Body [12]

Το SOAP Body παρέχει ένα μηχανισμό για μετάδοση πληροφοριών στον τελικό αποδέκτη ενός μηνύματος SOAP.

Για το SOAP Body είναι απαραίτητα τα εξής:

- Το όνομα του στοιχείου πρέπει να είναι Body.
- Το όνομα του namespace του να είναι "http://www.w3.org/2003/05/soap-envelope".
- Μπορεί να περιέχει μηδέν ή περισσότερες ιδιότητες ορισμένες με namespace.
- Μπορεί να περιέχει μηδέν ή περισσότερα στοιχεία-παιδιά ορισμένα με namespace.
- Μπορεί να περιέχει μηδέν ή περισσότερους κόμβους-παιδιά χαρακτήρων.

Κάθε στοιχείο-παιδί του SOAP Body ονομάζεται SOAP Body child Element.

5.8.1 SOAP Body child Element

Για κάθε στοιχείο-παιδί του SOAP Body είναι απαραίτητα τα εξής:

- Το όνομα του στοιχείου καλό θα ήταν να είναι ορισμένο με namespace.
- Μπορεί να περιέχει μηδέν ή περισσότερους κόμβους-παιδιά χαρακτήρων.
- Μπορεί να περιέχει μηδέν ή περισσότερα στοιχεία-παιδιά τα οποία μπορεί να είναι ορισμένα με namespace.
- Μπορεί να περιέχει μηδέν ή περισσότερες ιδιότητες. Ανάμεσα σε αυτές μπορεί να υπάρχει η παρακάτω που έχει ιδιαίτερο νόημα για την επεξεργασία του μηνύματος SOAP:
    - Η ιδιότητα encodingStyle.

5.9 SOAP Fault [12]

Το SOAP Fault χρησιμοποιείται για να κουβαλάει πληροφορίες σφαλμάτων μέσα σε ένα μήνυμα SOAP.

Για το SOAP Fault είναι απαραίτητα τα εξής:

- Το όνομα του στοιχείου πρέπει να είναι Fault.
- Το όνομα του namespace του να είναι "http://www.w3.org/2003/05/soap-envelope".
- Πρέπει να περιέχει δύο ή περισσότερα στοιχεία-παιδιά με την παρακάτω σειρά:
    - Ένα υποχρεωτικό στοιχείο Code.
    - Ένα υποχρεωτικό στοιχείο Reason.
    - Ένα προαιρετικό στοιχείο Node.
    - Ένα προαιρετικό στοιχείο Role.
    - Ένα προαιρετικό στοιχείο Detail.

Για να αναγνωριστεί ότι ένα μήνυμα SOAP κουβαλάει πληροφορίες σφάλματος πρέπει να περιέχει ένα στοιχείο Fault σαν το μοναδικό στοιχείο-παιδί του SOAP Body.

5.9.1 Στοιχείο SOAP Code

Για το στοιχείο SOAP Code ισχύουν τα παρακάτω :

- Το όνομα του στοιχείου πρέπει να είναι Code.
- Το όνομα του namespace του να είναι "http://www.w3.org/2003/05/soap-envelope".
- Πρέπει να περιέχει ένα ή και τα δύο από τα στοιχεία-παιδιά με την παρακάτω σειρά:
    - Ένα υποχρεωτικό στοιχείο Value.
    - Ένα υποχρεωτικό στοιχείο Subcode.

5.9.1.1 Στοιχείο SOAP Value (με πατέρα το στοιχείο Code)

Για το στοιχείο SOAP Code ισχύουν τα παρακάτω :

- Το όνομα του στοιχείου πρέπει να είναι Code.
- Το όνομα του namespace του να είναι "http://www.w3.org/2003/05/soap-envelope".

Ο τύπος αυτού του στοιχείου είναι ***env:faultCodeEnum***. Το SOAP καθορίζει ένα μικρό σύνολο από πιθανές τιμές καλύπτωντας σφάλματα SOAP υψηλού επιπέδου.

Οι τιμές που ορίζει το SOAP είναι οι εξής :

Όνομα	**Νόημα**
VersionMismatch	Βρέθηκε λάθος στοιχείο στη θέση του στοιχείου Envelope. Το όνομα, το namespace ή και τα δύο δεν ταιριάζουν με αυτά του προτύπου για το στοιχείο Envelope.
MustUnderstand	Υπάρχει SOAP header block με τιμή "true" στην ιδιότητα mustUnderstand του που όμως ο κόμβος δεν μπορεί να επεξεργαστεί.
DataEncodingUnknown	Ένα SOAP header block ή ένα SOAP body child που προοριζόταν για αυτόν τον κόμβο είχε κωδικοποίηση δεδομένων που δεν υποστηρίζει ο κόμβος.
Sender	Το μήνυμα δεν ήταν σωστά δομημένο ή δεν περιείχε τη σωστή πληροφορία για να επιτύχει η επεξεργασία του. Για παράδειγμα, το μήνυμα θα μπορούσε να μην περιείχε πληροφορίες αυθεντικοποίησης ή πληρωμής. Είναι ενδεικτικό ότι το μήνυμα δεν μπορεί να

	ξανασταλεί χωρίς αλλαγές.
Receiver	Δεν μπόρεσε να γίνει η επεξεργασία του μηνύματος για λόγους που είχαν να κάνουν με την ίδια τη διαδικασία παρά με τα περιεχόμενα του μηνύματος. Για παράδειγμα η επεξεργασία θα μπορούσε να περιέχει επικοινωνία με άλλο κόμβο η οποία να μην πέτυχε. Το μήνυμα θα μπορούσε να επιτύχει αν αποστελόταν κάποια στιγμή αργότερα.

5.9.1.2 Στοιχείο SOAP Subcode (με πατέρα το στοιχείο Code)

Για το στοιχείο SOAP Subcode ισχύουν τα παρακάτω :

- Το όνομα του στοιχείου πρέπει να είναι Subcode.
- Το όνομα του namespace του να είναι "http://www.w3.org/2003/05/soap-envelope".
- Πρέπει να περιέχει ένα ή και τα δύο από τα στοιχεία-παιδιά με την παρακάτω σειρά:
    - Ένα υποχρεωτικό στοιχείο Value.
    - Ένα υποχρεωτικό στοιχείο Subcode (ανδρομικά).

5.9.1.2.1 Στοιχείο SOAP Value (με πατέρα το στοιχείο Subcode)

Για αυτό το στοιχείο ισχύουν τα παρακάτω :

- Το όνομα του στοιχείου πρέπει να είναι Value.
- Το όνομα του namespace του να είναι "http://www.w3.org/2003/05/soap-envelope".

Ο τύπος αυτού του στοιχείου είναι **xs:QName**. Η τιμή αυτού του στοιχείου είναι μια υποκατηγορία της τιμής του στοιχείου Value με πατέρα το στοιχείο Subcode. Η τιμή αυτή καθορίζεται από την ίδια την εφαρμογή κάθε φορά.

5.9.2 Στοιχείο SOAP Reason

Το στοιχείο Reason προορίζεται για να παρέχει μία εξήγηση του σφάλματος κατανοήσιμη από τον άνθρωπο.

Για αυτό το στοιχείο ισχύουν τα παρακάτω :

- Το όνομα του στοιχείου πρέπει να είναι Reason.
- Το όνομα του namespace του να είναι "http://www.w3.org/2003/05/soap-envelope".
- Πρέπει να περιέχει ένα ή περισσότερα στοιχεία-παιδιά *Text*. Κάθε στοιχείο Text θα ήταν καλό να περιέχει διαφορετική τιμή για κάθε γλώσσα.

5.9.2.1 Στοιχείο SOAP Text

Το στοιχείο Text προορίζεται να κουβαλάει το κείμενο της εξήγησης σφάλματος.

Για αυτό το στοιχείο ισχύουν τα παρακάτω :

- Το όνομα του στοιχείου πρέπει να είναι Text.
- Το όνομα του namespace του να είναι "http://www.w3.org/2003/05/soap-envelope".
- Πρέπει να περιέχει υποχρεωτικά μία ιδιότητα με όνομα *lang* και namespace "http://www.w3.org/XML/1998/namespace". Σημειωτέον ότι ο ορισμός της ιδιότητας lang απαιτεί το πρόθεμα "*xml*" (ανεξαρτήτα από κεφαλαία-πεζά).
- Οποιοδήποτε αριθμό χαρακτήρων.

Αυτό το στοιχείο θα πρέπει να περιέχει πληροφορία που εξηγεί τη φύση του σφάλματος και δεν προορίζεται για αλγοριθμική επεξεργασία.

5.9.3 Στοιχείο SOAP Node

Το στοιχείο Node προορίζεται να παρέχει πληροφορίες για το ποιός κόμβος επεξεργασίας SOAP προκάλεσε το σφάλμα.

Για αυτό το στοιχείο ισχύουν τα παρακάτω :

- Το όνομα του στοιχείου πρέπει να είναι Node.
- Το όνομα του namespace του να είναι "http://www.w3.org/2003/05/soap-envelope".

Ο τύπος αυτού του στοιχείου είναι xs:URI. Κάθε κόμβος επεξεργασίας αναγνωρίζεται από ένα URI. Η τιμή αυτού του στοιχείου είναι το URI του κόμβου επεξεργασίας στον οποίο προκλήθηκε το σφάλμα.

Στην περίπτωση που το σφάλμα προκλήθηκε από την επεξεργασία του μηνύματος σε έναν ενδιάμεσο κόμβο τότε ο κόμβος αυτός πρέπει οπωσδήποτε να περιλάβει στην απάντησή του αυτό το στοιχείο για να δηλώσει ότι το σφάλμα προκλήθηκε σε αυτόν. Αν πρόκειται για τον τελικό παραλήπτη του μηνύματος τότε αυτός μπορεί προεραιτικά να περιλάβει αυτό το στοιχείο.

5.9.4 Στοιχείο SOAP Role

Το στοιχείο Role υποδεικνύει το ρόλο τον οποίο είχε ο κόμβος στον οποίο προκλήθηκε το σφάλμα.

Για αυτό το στοιχείο ισχύουν τα παρακάτω :

- Το όνομα του στοιχείου πρέπει να είναι Role.
- Το όνομα του namespace του να είναι "http://www.w3.org/2003/05/soap-envelope".

Το SOAP καθορίζει τρεις από αυτούς που έχουν ειδικό νόημα και είναι οι εξής :

Όνομα	**Περιγραφή**
next	Κάθε ενδιάμεσος κόμβος ή ο τελικός παραλήπτης πρέπει να δράσει.
none	Κανένας κόμβος δεν πρέπει να δράσει.
ultimateReceiver	Ο τελικός παραλήπτης πρέπει να δράσει.

5.9.5 Στοιχείο SOAP Detail

Το στοιχείο Detail προορίζεται για να κουβαλάει πληροφορίες για σφάλματα που σχετίζονται με το SOAP Body.

Για αυτό το στοιχείο ισχύουν τα παρακάτω :

- Το όνομα του στοιχείου πρέπει να είναι Detail.
- Το όνομα του namespace του να είναι "http://www.w3.org/2003/05/soap-envelope".
- Μπορεί να περιέχει μηδέν ή περισσότερες ιδιότητες.
- Μπορεί να περιέχει μηδέν ή περισσότερα στοιχεία-παιδιά τα οποία μπορεί να είναι ορισμένα με namespace.

Το στοιχείο Detail μπορεί να περιέχει χαρακτήρες.

Το στοιχείο Detail μπορεί να είναι παρόν σε ένα SOAP Fault και στην περίπτωση αυτή κουβαλάει επιπλέον πληροφορίες σχετικές με τους κωδικούς σφάλματος περιγράφοντας έτσι το σφάλμα. Για παράδειγμα, το στοιχείο Detail θα μπορούσε να περιέχει πληροφορίες σχετικές με το ότι ένα μήνυμα δεν περιλαμβάνει κατάλληλα πιστοποιητικά.
Η παρουσία του στοιχείου Detail δεν έχει καμία σημασία στο ποια μέρη του μηνύματος έχουν υποστεί επεξεργασία.

Όλα τα στοιχεία-παιδιά αυτού του στοιχείου ονομάζονται *detail entries*.

5.9.5.1 SOAP Detail Entry

Για κάθε detail entry ισχύουν τα παρακάτω:

- Το όνομα του στοιχείου μπορεί να είναι ορισμένο με namespace.
- Μπορεί να περιέχει μηδέν ή περισσότερα στοιχεία-παιδιά.
- Μπορεί να περιέχει μηδέν ή περισσότερους κόμβους-παιδιά χαρακτήρων.
- Μπορεί να περιέχει μηδέν ή περισσότερες ιδιότητες. Ανάμεσα σε αυτές μπορεί να υπάρχει η παρακάτω που έχει ιδιαίτερο νόημα για την επεξεργασία του μηνύματος SOAP:
    - Η ιδιότητα encodingStyle.

Αν είναι παρούσα, η ιδιότητα encodingStyle υποδεικνύει τον τρόπο κωδικοποίησης για αυτό το detail entry.

5.10 Μοντέλα ανταλλαγής μηνυμάτων [12]

Το SOAP είναι ένα απλό πλαίσιο μηνυμάτων για ανταλλαγή μηνυμάτων σε μορφή XML μεταξύ ενός αρχικού αποστολέας και ενός τελικού αποδέκτη. Τα πιο ενδιαφέροντα σενάρια περιλαμβάνουν πολλαπλές ανταλλαγές μηνυμάτων μεταξύ των δύο αυτών κόμβων. Η πιο απλή από αυτές τις περιπτώσεις είναι η μορφή αίτηση-απάντηση (request-response).

5.11 Απομακρυσμένες κλήσεις διαδικασιών (RPC) [12]

Οι αρχικές χρήσεις του SOAP έδιναν έμφαση στο μοντέλο αίτησης-απάντησης για τη χρήση του σε απομακρυσμένες κλήσεις δαιδικασίας που ήταν και ένας από τους στόχους κατά το σχεδιασμό του πρωτοκόλλου.

Το SOAP έχει καθορίσει μία ομοιόμορφη αναπαράσταση μηνύματος για κλήσεις RPC και απαντήσεις.

Στα παρακάτω παραδείγματα φαίνονται η αίτηση SOAP και η απάντηση SOAP για μία κλήση RPC και αφορούν την απομακρυσμένη κλήση μιας διαδικασίας με όνομα GetStockPrice η οποία δέχεται ένα όρισμα με όνομα StockName τύπου αλφαριθμητικού και επιστρέφει μία τιμή τύπου πραγματικού.

5.11.1 Παράδειγμα SOAP Request

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
soap:encodingStyle=" http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPrice>
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

5.11.2 Παράδειγμα SOAP Response

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
soap:encodingStyle=" http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPriceResponse>
      <m:Price>34.5</m:Price>
    </m:GetStockPriceResponse>
  </soap:Body>
</soap:Envelope>
```

5.12 Συνομιλητικές ανταλλαγές μηνυμάτων [12]

Ένα ακόμη μεγαλύτερο σύνολο από σενάρια χρήσης τα οποία μπορούν να καλυφθούν από το μοντέλο αίτησης-απάντησης μπορούν να μοντελοποιηθούν απλά σαν ανταλλαγή ολόκληρων εγγράφων XML με τη μορφή μίας μπρος-πίσω «συνομιλίας», όπου η σημασιολογία εξαρτάται από τις ίδιες τις εφαρμογές.
Αυτό το ενδεχόμενο μας δίνει πληρη ευελειξία για εφαρμογές web services αν και μέχρι στιγμής οι περισσότερες εφαρμογές web services βασίζονται σε σενάρια RPC, ίσως και λόγω του γεγονότος ότι οι προγραμματιστές είναι πιο κοντά νοητικά σε αυτό το σενάριο.

5.13 Πλαίσιο Συνδέσεων Binding Framework [12]

Τα μηνύματα SOAP μπορούν να ανταλλάσσονται χρησιμοποιώντας μια πληθώρα από πρωτόκολλα δικτύου συμπεριλαμβανομένων και πρωτοκόλλων επιπέδου εφαρμογών (application layer protocols).
Η προδιαγραφή για το πώς μηνύματα SOAP μπορούν να περνιούνται από ένα κόμβο σε ένα άλλο χρησιμοποιώντας ένα πρωτόκολλο ονομάζεται «σύνδεση SOAP» (SOAP binding).

5.14 HTTP Binding [12]

Το HTTP έχει ένα ευρέως γνωστό μοντέλο σύνδεσης και μορφή ανταλλαγής μηνυμάτων. Ο πελάτης αναγνωρίζει τον εξυπηρετητή μέσω ενός URI, συνδέεται σε αυτόν χρησιμοποιώντας το TCP/IP, εκτελεί μία αίτηση HTTP και λαμβάνει μία απάντηση HTTP πάνω από την ίδια σύνδεση TCP. Το HTTP συνδέει το μήνυμα αίτησης με το μήνυμα απάντησης. Για αυτό το λόγο μια εφαρμογή που χρησιμοποιεί αυτή τη σύνδεση μπορεί να επιλέξει να συνδέσει ένα μήνημα SOAP που στάλθηκε στο σώμα μιας αίτησης HTTP με ένα μήνυμα SOAP που επιστρέφηκε στο σώμα μιας απάντησης HTTP.

Η απλότητα χρήσης του HTTP είναι και ο λόγος για τον οποίο είναι το πιο ευρέως χρησιμοποιούμενο πρωτόκολλο για την αποστολή μηνυμάτων SOAP.

Παρακάτω παρατίθεται η αίτηση και η απάντηση HTTP που πειρλαμβάνουν τα μηνύματα SOAP που είδαμε προηγουμένως στο RPC σενάριο.

5.14.1 Παράδειγμα SOAP Request με HTTP Binding

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
soap:encodingStyle=" http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPrice>
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

5.14.2 Παράδειγμα SOAP Response με HTTP Binding

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
soap:encodingStyle=" http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPriceResponse>
      <m:Price>34.5</m:Price>
    </m:GetStockPriceResponse>
  </soap:Body>
</soap:Envelope>
```

5.15 Σύνοψη για το SOAP

Από τα προαναφερθέντα γύρω από το πρωτόκολλο SOAP θα πρέπει να έχει ήδη γίνει αντιληπτό ότι οι σχεδιαστές του πέτυχαν τους στόχους τους. Το Simple Object Access Protocol είναι ένα απλό, ευέλικτο και επεκτάσιμο πρωτόκολλο ανταλλαγής μηνυμάτων σε μορφή XML και μπορεί να χρησιμοποιήσει άλλα πρότυπα πρωτόκολλα σα μέσα μεταφοράς.

Ίσως το λιγότερο εμφανές σε όλα τα παραπάνω είναι η επεκτασιμότητά του η οποία όμως είναι το σημαντικότερο πλεονέκτημά του αφού πάνω σε αυτό το χαρακτηριστικό βασίζονται πολλές νέες τεχνολογίες των web services.

6 Αναπτυσσόμενες τεχνολογίες

Οι τρεις βασικές τεχνολογίες των web services : SOAP, WSDL, και UDDI μαζί με την XML στην οποία βασίζονται παρέχουν μόνο τις βασικές λειτουργίες των web services. Άλλα χαρακτηριστικά τα οποία είναι αναγκαία για μία ολοκληρωμένη κατανεμημένη τεχνολογία είναι η αξιοπιστία (reliability),η ασφάλεια (security) και οι συναλλαγές (transactions).
Η παροχή αυτών των υπηρεσιών στο περιβάλλον των web services είναι ευθύνη κάποιων άλλων προδιαγραφών που βρίσκονται αυτή τη στιγμή ακόμη υπό ανάπτυξη.

6.1 Web services και ασφάλεια

Τα web services παρέχουν πολλά πλεονεκτήματα στις εφαρμογές αλλά επίσης εκθέτουν σημαντικούς νέους κινδύνους στην ασφάλεια. Η δημιουργία και η διατήρηση ενός ασφαλούς περιβάλλοντος web services περιλαμβάνει και τη διαχείριση διαφόρων μηχανισμών για το διαδίκτυο, την XML και τα ίδια τα web services.

Η γενική αντιμετώπιση αφορά τα παρακάτω[15] :

- Ασφάλεια σε επίπεδο μεταφοράς όπως τα τύχη προστασίας (firewalls), τα ιδεατά ιδιωτικά δίκτυα (VPNs), η επικύρωση (authentication), η μη αποκήρυξη ευθύνης (non-repudiation), και η κρυπτογράφηση (encryption).

- Ασφάλεια σε επίπεδο μηνύματος όπως τα στοιχεία επικύρωσης (authentication tokens) για να επικυρώσουν την ταυτότητα του αιτούντα και οι δηλώσεις έγκρισης (authorization assertions) για να ελέγξουν την πρόσβαση στις υπηρεσίες του παρόχου.

- Ασφάλεια σε επιπεδο δεδομένων όπως είναι η κρυπτογράφηση (encryption) και η ηλεκτρονική υπογραφή (digital signature) για την προστασία ενάντια στις αθέμητες αλλαγές των δεδομένων.

- Ασφάλεια σε επίπεδο περιβάλλοντος όπως είναι η διαχείριση (management), η αναγραφή (logging) και ο έλεγχος (auditing) για την αναγνώριση των προβλημάτων που πρέπει να διορθωθούν.

Για την αντιμετώπιση όλων των παραπάνω δημιουργήθηκε μια ομόδα από προδιαγραφές που είναι ακόμη υπό ανάπτυξη. Οι κυριότερες από αυτές είναι οι παρακάτω[15]:

- ***WS-Security***: καθορίζει μια αρχιτεκτονική για ασφαλή επικοινωνία.
- ***WS-Policy***: και οι σχετικές με αυτήν προδιαγραφές καθορίζουν μια πολιτική από κανόνες για το πώς οι υπηρεσίες θα αλληλεπιδρούν μεταξύ τους.
- ***WS-Trust***: καθορίζει το μοντέλο εμπιστοσύνης για ασφαλείς συναλλαγές.
- ***WS-Privacy***: καθορίζει πώς τηρείται η ιδιοτικότητα στις πληροφορίες.
- ***WS-Secure Conversation***: καθορίζει πώς θα επιτευχθεί μια ασφαλή συνεδρία μεταξύ υπηρεσιών που ανταλλάσουν δεδομένα με κανόνες ορισμένους στα WS-Policy, WS-Trust, WS-Privacy.
- ***WS-Federation***: καθορίζει τους κανόνες σχετικά με την ταυτότητα σε κατανεμημένο περιβάλλον.
- ***WS-Authorization***: χειρίζεται την επεξεργασία για την επικύρωση που αφορά στην πρόσβαση και την ανταλλαγή δεδομένων.

Όλα τα παραπάνω χρησιμοποιούν το SOAP Header και έτσι εκμεταλλεύονται στο έπακρο την επεκτασιμότητα του πρωτοκόλλου SOAP.

6.2 Web services και μηνυματοδότηση (messaging)

Υπάρχει ακόμη μία ομάδα προδιαγραφών που πραγματεύεται ζητήματα που έχουν να κάνουν με την αξιόπιστη αποστολή μηνυμάτων (reliable messaging) και τις ειδοποιήσεις (notification) [15]:

- ***WS-ReliableMessaging και WS-Reliability***: καθορίζουν μηχανισμούς για την εξασφάλιση της παράδοσης των μηνυμάτων, με τη σωστή σειρά, και την εξάλειψη των διπλότυπων.
- ***WS-Eventing και WS-Notification***: παρέχουν ένα μηχανισμό σχετικά με τα γεγονότα (events) ώστε οι συνδρομητές να ειδοποιούνται ανεξάρτητα από τη σχέση μεταξύ του παρόχου και του αιτούντα.

Και οι παραπάνω τεχνολογίες χρησιμοποιούν το SOAP Header.

6.3 Web services και συναλλαγές (transactions)

Οι συναλλαγές αποτελούν ένα πολύ σημαντικό χαρακτηριστικό το οποίο έχει αποδειχθεί πολύ χρήσιμο στο πέρασμα των χρόνων.
Οι συναλλαγές στο περιβάλλον των web services μας εξασφαλίζουν ότι μία ομάδα από web services πετυχαίνει ένα κοινό αποτέλεσμα. Τα web services συχνά εξαρτώνται το ένα από το άλλο για να ολοκληρώσουν πολύπλοκες αιτήσης σε μία εφαρμογή όπως η ενημέρωση μιας εγγραφής πελάτη (η οποία μπορεί να ενημερώνει πολλαπλές βάσεις δεδομένων με στοιχεία πελάτη) ή η επεξεργασία μίας παραγγελίας (η οποία μπορεί να ενημερώνει πολλαπλές βάσεις δεδομένων αποθηκών) [15].

Οι πιο σημαντικές προσπάθειες που σχετίζονται με τις συναλλαγές στο περιβάλλον των web services είναι οι παρακάτω[15]:

- ***WS-Transaction***: μία οικογένεια προδιαγραφών που περιλαμβάνει:
    - ***WS-AtomicTransactions (WS-AT)***: Ένα πρωτόκολλο two-phase commit για διαλειτουργικότητα των web services.
    - **WS-BusinessActivity (WS-BA)**: Ένα ανοικτό ενσωματωμένο πρωτόλλο για μακροπρόθεσμες επιχειρισιακές διαδικασίες.
    - ***WS-Coordination (WS-C)***: Ένα πλαίσιο συντονισμού που υποστηρίζει τα WS-AT και WS-C.
- ***WS-Composite Application Framework***: μία οικογένεια προδιαγραφών που περιλαμβάνει:
    - ***WS-Context (WS-CTX)***: Ένας γενικός μηχανισμός διαχείρισης πλαισίου.
    - ***WS-CoordinationFramework (WS-CF)***: Ένα πλαίσιο συντονισμού που υποστηρίζει τα WS-AT, WS-BA, και τα τρία πρωτόκολλα στο WS-TXM.
    - ***WS-TransactionManagement (WS-TXM)***: Ένα πρωτόκολλο για two-phase commit για διαλειτουργικότητα των Web Services (ACID), ένα πρωτόκολλο βασισμένο στη διόρθωση (compensation) για μακροπρόθεσμες διαδικασίες (LRA), και ένα πρωτόκολλο διαχείρισης επιχειρησιακών διαδικασιών (BP).

7 Συμπέρασμα

Τα τελευταία χρόνια η XML έχει επιτρέψει σε διαφορετικά υπολογιστικά περιβάλλοντα να μοιράζονται πληροφορίες μέσω του παγκόσιμου ιστού. Τώρα προσφέρει ένα απλοποιημένο τρόπο με τον οποίο μπορούν να μοιράζονται και επεξεργασία.

Από τεχνικής σκοπιάς, η άνθηση των Web Services δεν είναι μια επανάσταση στα κατανεμημένα συστήματα. Αντίθετα είναι μια φυσική εξέλιξη της εφαρμογής της XML από δομημένη αναπαράσταση πληροφορίας σε δομημένη αναπαράσταση μηνυμάτων μεταξύ των εφαρμογών[3].

Πριν την άφιξη των Web Services, η ολοκλήρωση επιχειρηματικών συστημάτων ήταν πολύ δύσκολη εξαιτίας των διαφορών στις γλώσσες προγραμματισμού και του υλικολογισμικού (middleware) που χρησιμοποιούνταν μέσα στις επιχειρήσεις. Αυτό οδήγησε σε μια κατάσταση όπου η διαλειτουργικότητα ήταν δύσκολη και επίπονη. Με την άφιξη των Web Services κάθε εφαρμογή μπορεί να ολοκληρωθεί αρκεί να είναι διαδικτυακή.

Είναι δύσκολο να αποφύγει κανείς τη δημοσιότητα και τη διαφημιστική εκστρατεία γύρω από τα Web Services. Κάθε μεγάλος προμηθευτής λογισμικού έχει κάποια πρωτοβουλία σχετική με τα Web Services και υπάρχει πάντα μια θεωρεία για το μέλλον της αγοράς. Όπως και να έχει, οι αρχιτεκτονικές Web Services παρέχουν ένα πολύ διαφορετικό τρόπο σκέψης για την ανάπτυξη λογισμικού. Από το μοντέλο πελάτη-διακομιστή στα συστήματα ν-επιπέδων, στα κατανεμημένα συστήματα, οι εφαρμογές Web Services αντιπροσωπεύουν το αποκορύφωμα κάθε μίας από αυτές τις αρχιτεκτονικές σε συνδυασμό με το διαδίκτυο[3].

8 Χρήσιμοι Δεσμοί

Πρωτόκολλα μεταφοράς

HTTP

- **RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1**
  http://www.ietf.org/rfc/rfc2616.txt

- **RFC 617: HTTP Authentication: Basic and Digest Access Authentication**
  http://www.ietf.org/rfc/rfc2617.txt

- **Wikipedia – HTTP**
  http://en.wikipedia.org/wiki/HTTP

SMTP

- **RFC 821: Simple Mail Transfer Protocol**
  http://www.faqs.org/rfcs/rfc821.html

- **Wikipedia – SMTP**
  http://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol

XML

Γλώσσα XML

- **W3C - Extensible Markup Language**
  http://www.w3.org/XML/

- **XML.org**
  http://www.xml.org/

- **XML.com: A Technical Introduction to XML**
  http://www.xml.com/lpt/a/316

- **Wikipedia – XML**
  http://en.wikipedia.org/wiki/XML

DTD

- **W3C XML Specification DTD**
  http://www.w3.org/XML/1998/06/xmlspec-report-19980910.htm

- **W3CSchools - DTD Tutorial**
  http://www.w3schools.com/dtd/default.asp

XML Schema

- **W3C XML Schema**
  http://www.w3.org/XML/Schema

- **W3CSchools - XML Schema Tutorial**
  http://www.w3schools.com/schema/default.asp

- **XML.com - Using W3C XML Schema**
  http://www.xml.com/pub/a/2000/11/29/schemas/part1.html

Βασικές Τεχνολογίες των web services

UDDI

- **UDDI.org**
  http://www.uddi.org/

- **UDDI Specification v3**
  http://uddi.org/pubs/uddi_v3.htm

- **UDDI Best Practices**
  http://www.oasis-open.org/committees/uddi-spec/doc/bps.htm

- **UDDI Technical Notes**
  http://www.oasis-open.org/committees/uddi-spec/doc/tns.htm

- **Cover Pages – UDDI**
  http://xml.coverpages.org/uddi.html

WSDL

- **W3C - Web Services Description Language**
  http://www.w3.org/TR/wsdl

- **W3CSchools – WSDL Tutorial**
  http://www.w3schools.com/wsdl/default.asp

- **Cover Pages – WSDL**
  http://xml.coverpages.org/wsdl.html

- **Wikipedia – WSDL**
  http://en.wikipedia.org/wiki/Web_Services_Description_Language

SOAP

- **W3C - Web Services Description Language**
  http://www.w3.org/TR/2006/PER-soap12-part0-20061219/
  (Part 0: Primer)

  http://www.w3.org/TR/soap12-part1/
  (Part 1: Messaging Framework)

  http://www.w3.org/TR/2003/REC-soap12-part2-20030624/
  (Part 2: Adjuncts)

9 Βιβλιογραφία

1. IBM (2007), *"New to SOA and Web services"*
http://www-128.ibm.com/developerworks/webservices/newto/websvc.html
[6 Μαρτίου 2007]

2. MSDN (2001), "*XML Web Services Basics*"
http://msdn2.microsoft.com/en-us/library/ms996507.aspx
[6 Μαρτίου 2007]

3. Roseindia.net (2007), "*Web Services*"
http://www.roseindia.net/webservices/webservices.shtml
[6 Μαρτίου 2007]

4. Developer.com (2003), "*Web Services Tutorial: Understanding XML and XML Schema*"
"Part 1"
http://www.developer.com/services/print.php/2195981
"Part 2"
http://www.developer.com/services/print.php/10928_2195981_2
[6 Μαρτίου 2007]

5. Developer.com (2002), "*Introduction to Web Services*"
"Part 1"
http://www.developer.com/services/article.php/1485821
"Part 2: Architecture"
http://www.developer.com/services/article.php/1495091
"Part 3: Understanding XML"
http://www.developer.com/services/article.php/1557871
[6 Μαρτίου 2007]

6. ONJava.com (2001), "*Java and Web Services Primer*"
http://www.onjava.com/lpt/a/1025
[6 Μαρτίου 2007]

7. Horton I. (2003) *"Beginning Java 2"* : Wrox

8. W3schools.com (2007), "*XML Tutorial*"
http://www.w3schools.com/xml/default.asp
[6 Μαρτίου 2007]

9. W3C (2001), "*Web Service Definition Language (WSDL)*"
http://www.w3.org/TR/wsdl
[6 Μαρτίου 2007]

10. XML.com (2001), "*A Web Services Primer*"
http://webservices.xml.com/lpt/a/760
[6 Μαρτίου 2007]

11. UDDI.org (2004), "*UDDI Version 3.0.2*"
http://uddi.org/pubs/uddi_v3.htm
[6 Μαρτίου 2007]

12. W3C (2006), "*SOAP Version 1.2*"
"Part 0: Primer"
http://www.w3.org/TR/2006/PER-soap12-part0-20061219/
"Part 1: Messaging Framework"
http://www.w3.org/TR/soap12-part1/
"Part 2: Adjuncts"
http://www.w3.org/TR/2003/REC-soap12-part2-20030624/
[6 Μαρτίου 2007]

13. Borgen B. (2001) *"SOAP Programming with Java"* : Sybex Inc

14. W3schools.com (2007), "*SOAP Tutorial*"
http://www.w3schools.com/soap/default.asp
[6 Μαρτίου 2007]

15. Newcomer E., Lomow G (2005) *"Understanding SOA with Web Services"* : Addison-Wesley Professional