ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΔΙΠΛΩΜΑ ΕΙΔΙΚΕΥΣΗΣ (MSc)
στα ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Συνεργαζόμενες Ηλεκτρονικές Υπηρεσίες στο Διαδίκτυο»

Κοκκαλάκης Δημήτρης

Μ3990002

ΑΘΗΝΑ, ΦΕΒΡΟΥΑΡΙΟΣ 2001

ΜΕΤΑΠΤΥΧΙΑΚΟ ΔΙΠΛΩΜΑ ΕΙΔΙΚΕΥΣΗΣ (MSc)
στα ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Συνεργαζόμενες Ηλεκτρονικές Υπηρεσίες στο Διαδίκτυο»

Κοκκαλάκης Δημήτρης
Μ3990002

Επιβλέπων Καθηγητής: Εμμανουήλ Γιακουμάκης
Εξωτερικός Κριτής: Νικόλαος Μαλεύρης

ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΘΗΝΑ, ΦΕΒΡΟΥΑΡΙΟΣ 2001

Περιεχόμενα

Στο σημείο αυτό, κλείνοντας ένα κύκλο ακαδημαϊκής δραστηριοποίησης θα ήθελα να ευχαριστήσω θερμά την οικογένεια μου για την βοήθεια που μου πρόσφερε ενισχύοντας με ποικιλοτρόπως όλα αυτά τα χρόνια. Επίσης, τους δασκάλους μου όλων των βαθμίδων καθώς συνέβαλλαν αποφασιστικά στη διαμόρφωση της προσωπικότητας μου καθώς και στην τωρινή επαγγελματική μου εξέλιξη. Τέλος θερμές ευχαριστίες στη Μένια Γιαννάκη για την πολύπλευρη συνδρομή της τόσο στην παρούσα εργασία όσο και σε άλλα εξίσου σημαντικά ζητήματα της ζωής μου.

Δημήτρης Κοκκαλάκης

ΚΕΦΑΛΑΙΟ 1ο

ΣΥΝΟΨΗ

1 ΣΥΝΟΨΗ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Το νέο όραμα που οδηγεί τις εξελίξεις αφορά το μοντέλο που πυρήνα έχει την "υπηρεσία" ως υπολογιστική μονάδα. Μιλώντας για ηλεκτρονικές υπηρεσίες αναφερόμαστε σε υπηρεσίες διαθέσιμες μέσω του Internet, οι οποίες φέρουν εις πέρας διάφορα καθήκοντα, επιλύουν προβλήματα ή ολοκληρώνουν συναλλαγές. Μελλοντικά, σε μια ιδεατή κατάσταση θα μπορούμε να ισχυριστούμε ότι κάθε αγαθό από S/W και H/W έως επιχειρηματικές διαδικασίες, δεδομένα και εμπειρική γνώση μπορεί να γίνουν διαθέσιμα σαν ηλεκτρονικές υπηρεσίες, οδηγώντας σε νέα ρεύματα επιχειρηματικής δραστηριοποίησης. Το Web ήδη τείνει να μετασχηματιστεί σε μια τεράστια αγορά όπου τέτοιου είδους υπηρεσίες μπορούν να αναπτυχθούν, να ανακαλυφθούν δυναμικά και να μετασχηματιστούν συνεργαζόμενες σε ανωτέρου επιπέδου υπηρεσίες.

Το μοντέλο των υπηρεσιών διαμορφώνει ένα κόσμο όπου η λειτουργικότητα και οι δυνατότητες θα προσφέρονται και θα υποστηρίζονται από αυτόνομες, ανεξάρτητες οντότητες. Παρέχοντας διαρκώς μια ποικιλία υπηρεσιών σε όλο το εύρος των κλάδων της βιομηχανίας (όχι κατ' ανάγκην της πληροφορικής βιομηχανίας μόνο), θα προκαλέσει άμεσες επιδράσεις στη ζωή μας. Βασικό στοιχείο της αλλαγής αποτελεί η μεταφορά από την κεφαλαιουχική αντίληψη των αγαθών, σε μια αντίληψη όπου η χρέωση διαμορφώνεται σύμφωνα με την χρήση που γίνεται. Έτσι, με βάση αυτή την αρχή, στο μέλλον αναμένεται να δούμε σε μεγαλύτερη έκταση χρήση και χρέωση υπολογιστικής ισχύος, αποθήκευσης κ.τ.λ, όπως ακριβώς συμβαίνει σήμερα με τη χρήση και τη χρέωση του ηλεκτρικού ρεύματος.

Οι ηλεκτρονικές υπηρεσίες αναμένεται να διαμορφώσουν μια γέφυρα μετάβασης από τη κλασσική έννοια της λειτουργίας της I/T, σε μια επόμενη γενιά εφαρμογών προσανατολισμένες στην προς-χρήση αντιμετώπιση. Δεν είναι καθόλου παράδοξο να δούμε τα πάντα, από υπολογιστική ισχύ έως web-sites να διαμορφώνονται, να διαφημίζονται και να παραδίδονται σαν ηλεκτρονικές υπηρεσίες. Άλλα παραδείγματα εφαρμογής περιλαμβάνουν την απομακρυσμένη αποθήκευση αρχείων, ή την βασισμένη στο δίκτυο προετοιμασία και υποβολή της φορολογικής δήλωσης. Πιο συγκεκριμένα, το πεδίο που όπως αναφέραμε διαμορφώνεται, θα έχει σημαντική επίδραση πάνω σε τρεις τομείς: Βασικές λειτουργίες I/T, application delivery και επόμενης γενιάς συνδυαζόμενες ηλεκτρονικές υπηρεσίες.

Ο απώτερος στόχος των ηλεκτρονικών υπηρεσιών βρίσκεται πέρα από τη παροχή των συμβατικών IT λειτουργιών που αναφέραμε. Αφορά στη διαμόρφωση υπηρεσιών ανώτερου επιπέδου, συνδυάζοντας τις ήδη υπάρχουσες. Ας πάρουμε για παράδειγμα την επόμενη γενιά των υπηρεσιών δημοπρασίας. Αντί να ταιριάζουν απλά τις προσφορές για κάποιο αντικείμενο, αυτές θα ενεργοποιούν ορισμένους δημοπράτες οι οποίοι θα ορίζουν ένα εύρος τιμών και προτιμήσεων και σε πραγματικό χρόνο, σε συνδυασμό πάντα με το αντίστοιχο χρηματοπιστωτικό ίδρυμα, θα πλειοδοτούν για το καλύτερο αποτέλεσμα. Τα όρια που θέτουμε στο παραπάνω σενάριο είναι εύκολο να ξεπεραστούν προχωρώντας σε πολύπλοκες υποθέσεις τις οποίες μπορούν να διαμορφώσουν μόνο κάποιες συνεργαζόμενες υπηρεσίες.

Με τις συνεργαζόμενες ηλεκτρονικές υπηρεσίες μπορούμε να έχουμε κατά νου μια τελείως διαφορετική οργάνωση του επιχειρηματικού μοντέλου, όπου τα διάφορα λειτουργικά τμήματά του μπορούν να θεωρηθούν ως αυτόνομες υπηρεσίες. Τις υπηρεσίες αυτές μπορούμε να τις καλέσουμε ανάλογα με τις ανάγκες. Έτσι, με την

πλήρη μοντελοποίηση και αυτονόμηση των επιμέρους τμημάτων, αν μερικά από αυτά δεν ανταποκρίνονται στον επιθυμητό βαθμό μπορούν σχετικά εύκολα να γίνουν outsourcing. Με αυτόν τον τρόπο μεταφέρουμε εκτός της επιχείρησης τις αδυναμίες της, στοχεύοντας στην βελτίωση τους. Από τη στιγμή που η επιχείρηση στηρίζεται σε πλήρης ανεξάρτητες υπηρεσίες καμία διάσπαση δεν μπορεί να πραγματοποιηθεί

Για τους πελάτες/ χρήστες του νέου μοντέλου υπηρεσιών τα οφέλη που θα προκύψουν αναμένονται να είναι λόγω της εκτεταμένης χρήσης τους, η αξιοπιστία, η περικοπή του κόστους, καθώς και η ασφάλεια της πληροφοριακής υποδομής. Στο νέο περιβάλλον, οι χρήστες έχουν πρόσβαση σε εφαρμογές (applications) οι οποίες συνεχώς βελτιώνονται. Έτσι για παράδειγμα, νέες εκδόσεις λογισμικού που έχουν ελεγχθεί μπορούν να παραδοθούν άμεσα στους χρήστες χωρίς να υπάρχει η ανάγκη της αναμονής για τη νέα έκδοση. Αλλά η λειτουργικότητα του νέου μοντέλου υπηρεσιών δεν μετράται μόνο από τον αριθμό των προσφερόμενων υπηρεσιών, αλλά και από τη διαθεσιμότητά τους. Έχοντας αυτή την παράμετρο υπ' όψιν, αναμένεται να καλυφθούν πλήρως οι προσδοκίες των χρηστών ικανοποιώντας τα υψηλά επίπεδα απόδοσης τα οποία αναμένονταν από την I/T.

Βασιζόμενοι στα δυνατά τους σημεία οι παροχείς υπηρεσιών, μπορούν να παραχωρήσουν στους πελάτες τα πλεονεκτήματα της διαμοιραζόμενης αρχιτεκτονικής, όπως και της συνεχούς υποστήριξης. Έχοντας την δυνατότητα επαναχρησιμοποίησης βασικών συστατικών στοιχείων παρέχουν οικονομίες κλίμακας στο περιβάλλον του I/T. Οι ASP's που ήδη αναφέραμε έχουν κάνει την εμφάνισή τους ακόμη και στην παροχή ERP υπηρεσιών μέσω δικτύου.

Η εφαρμογή του μοντέλου των υπηρεσιών είναι δυνατή και στο εσωτερικό των οργανισμών. Ο μετασχηματισμός του ισχύοντος μοντέλου προς αυτό των υπηρεσιών είναι κάτι που θα απασχολήσει σε μεγάλη έκταση τους διοικούντες έναν οργανισμό. Βέβαια, η απόφαση για το αν ορισμένες εσωτερικές διαδικασίες μπορούν να μετασχηματιστούν σε νέα βάση, αποτελεί παλιό δίλημμα του χώρου. Σε αυτό το κεφάλαιο θα αναζητήσουμε τις λεπτομέρειες μιας τέτοιας εφαρμογής, όπως επίσης τα θετικά και τα αρνητικά της υλοποίησης της.

Μιλώντας για παράδειγμα για τα τμήματα πληροφορικής των διαφόρων οργανισμών, συχνά αυτά αντιμετωπίζονται ως κέντρα παραγωγής κόστους παρά την σημαντική προσφορά τους. Η ίδια υπόθεση μπορεί να γίνει και για κάποιο άλλο τμήμα (π.χ. λογιστήριο). Ο ανταγωνισμός του συγκεκριμένου τομέα με αυτόν άλλων παροχέων υπηρεσιών είναι πολύ μικρός ως ανύπαρκτος και κυρίως για θέματα ασφαλείας. Η απομόνωση αυτή οδηγεί σε κλειστά διοικητικά τμήματα όπου παρατηρείται έντονα η έλλειψη καινοτομίας, η καθυστέρηση στην ανάδραση των ερεθισμάτων που υπάρχουν, αποφυγή του δημιουργικού ρίσκου και τέλος υψηλό κόστος.

Το ερώτημα που προκύπτει είναι σχετικά με το ποιος θα είναι ο νέος ρόλος των IT τμημάτων στο εσωτερικό των οργανισμών. Αυτό που πιθανολογείται ότι θα συμβεί είναι ότι θα μετασχηματιστούν άμεσα σε ενδιάμεσους (brokers) ή σε εσωτερικούς συμβούλους των εφαρμογών, κατευθύνοντας τους οργανισμούς ανάλογα.

Στο μέλλον όπου η απόφαση για εσωτερική ανάπτυξη ή αγορά θα κλίνει υπέρ του δεύτερου, ο ρόλος του ενδιάμεσου-συμβούλου κρίνεται ιδιαίτερα σημαντικός για τις αποφάσεις που θα παίρνονται. Καθώς το σενάριο που θέλει τους εσωτερικούς

χρήστες να επιλέγουν άμεσα τις υπηρεσίες που κρίνουν απαραίτητες, κρίνεται ανεδαφικό, ο ρόλος της επιλογής και της αξιολόγησης θα περάσει στο προσωπικό των τμημάτων της πληροφορικής. Η ηγεσία είναι αυτή που θα χαρακτηρίζει πλέον, μιας και είναι απαραίτητη για την υπόδειξη των τρόπων ανάληψης και υλοποίησης των επιχειρηματικών με τις κατάλληλες τεχνολογίες και μέσα.

Το Μοντέλο αγοράς ηλεκτρονικών υπηρεσιών στοχεύει στο να προσδιορίσει τα βασικά στοιχεία μιας γενικότερης αγοράς ηλεκτρονικών υπηρεσιών. Αυτά τα στοιχεία αντιπροσωπεύουν ένα μοναδικό σύνολο συνεισφορών, οι οποίες όλες μαζί σχηματίζουν τα θεμέλια για τη δημιουργία και διεξαγωγή ηλεκτρονικών υπηρεσιών. Τα στοιχεία του μοντέλου αγοράς ορίζονται ως ακόλουθα:

- **Οι συμμετέχοντες στη Συναλλαγή**, οι οποίοι αντιπροσωπεύουν τις ομάδες οι οποίες θα πρωταγωνιστούν σε μία συναλλαγή. Μια συναλλαγή μπορεί να περιπλέκει δύο ή περισσότερες ομάδες.
- **Οι Μεσάζοντες,** οι οποίοι αντιπροσωπεύουν τις οντότητες οι οποίες φέρνουν σε επαφή τους Αγοραστές με τους Παροχείς Υπηρεσιών, έτσι ώστε να επιτευχθεί κάποια συμφωνία.
- **Οι Δημιουργοί Αγοράς,** οι οποίοι αντιπροσωπεύουν τις οντότητες που διασφαλίζουν ότι η συναλλαγή διεκπεραιώνεται όπως ακριβώς είχε προσυμφωνηθεί. Αυτό αποτελεί ένα βασικό στοιχείο του μοντέλου, δεδομένου ότι δημιουργεί ένα κλίμα εμπιστοσύνης, το οποίο απαιτείται για τη διεξαγωγή εμπορικών συναλλαγών.
- **Οι Οικονομικοί Διευθυντές,** οι οποίοι αντιπροσωπεύουν τις οντότητες που υποστηρίζουν τις οικονομικές πτυχές μιας ηλεκτρονικής συναλλαγής.
- **Οι Διευθυντές Κύκλου ζωής,** οι οποίοι αντιπροσωπεύουν τις οντότητες που διοικούν όλες τις φάσεις του κύκλου ζωής, από τη γένεση ως και τη χρήση μιας υπηρεσίας. Το έργο τους περιλαμβάνει τη διαχείριση λειτουργιών σύνθεσης, εφοδιασμού, ανάπτυξης, και πραγματοποίησης.

Κάθε ηλεκτρονική υπηρεσία δεν είναι απαραίτητο να εμπεριέχει όλα τα προαναφερθέντα στοιχεία του μοντέλου αγοράς. Μια ηλεκτρονική υπηρεσία αγοράς μπορεί να είναι τόσο απλή, όσο μια απλή συναλλαγή χαμηλού ρίσκου μεταξύ ενός ελάχιστου αριθμού συμμετεχόντων η οποία δεν απαιτεί καν κάποιο συμβόλαιο, ή τόσο πολύπλοκη, όσο ένα σύνολο συναλλαγών το οποίο απαιτεί ένα εκτεταμένο σύστημα υποστήριξης και στο οποίο εμπλέκονται πολλοί ανώνυμοι συμμετέχοντες.

Η αναφορά στο μοντέλο των ηλεκτρονικών υπηρεσιών δημιουργεί τις κατάλληλες προϋποθέσεις για εφαρμογή του σε συγκεκριμένα επιχειρηματικά μοντέλα. Η παγκόσμια βιβλιογραφία που σχετίζεται με το ηλεκτρονικό εμπόριο δεν είναι συνεπής όταν αναφέρεται στον όρο "επιχειρηματικό" μοντέλο. Παρόλα αυτά, μπορούμε να εντοπίσουμε περιπτώσεις εφαρμογής ηλεκτρονικών υπηρεσιών, στα πλαίσια επιχειρηματικών μοντέλων που έχουν προταθεί μέχρι σήμερα. Μεταξύ αυτών διακρίνουμε τα παρακάτω επιχειρηματικά μοντέλα στα οποία μπορούν να εφαρμοσθούν οι ηλεκτρονικές υπηρεσίες:

- **E-procurement**
- **E-auction**
- **Third Party MarketPlaces**

- **Virtual Communities**

Στο εξής και με την σταδιακή εισαγωγή του νέου μοντέλου των υπηρεσιών αρχίζει να διαφαίνεται μια νέα πραγματικότητα με τη δεύτερη φάση την οποία διέρχεται το Internet, η οποία σε πολλά σημεία διαφέρει πλήρως από τα μέχρι τώρα δεδομένα. Οι εταιρίες θα ανταγωνίζονται πλέον σε μια διαφορετική βάση όπου η σχέση προσφερομένων-τιμής, η ποιότητα και η ταχύτητα παράδοσης θα αποτελούν τα προς σύγκριση τεκμήρια. Από την πλευρά τους οι πελάτες θα έχουν την δυνατότητα πρόσβασης σε αυτές τις υπηρεσίες μέσω πολλών διαφορετικών συσκευών (π.χ κινητών τηλεφώνων, palmtops, τηλεοράσεων κ.λ.π.) και όχι μόνο από τον κλασσικό προσωπικό ηλεκτρονικό υπολογιστή, ο οποίος αποτέλεσε τον θεμέλιο λίθο της πρώτης φάσης. Τέλος, οι παροχείς υπηρεσιών όπως ISP's, ASP's τηλεπικοινωνιακοί οργανισμοί, εκμεταλλευόμενοι τις τεχνολογικές προτάσεις - όπως αυτές που θα αναφερθούν συγκρινόμενες στη συνέχεια- θα υιοθετήσουν μια ξεχωριστή οπτική για την αγορά.

Βασικοί ανταγωνιστές του χώρου είναι, η Sun Microsystems, η HP και η Microsoft, των οποίων περιγράφεται η οπτική που έχουν υιοθετήσει για τον χώρο των συνεργαζόμενων ηλεκτρονικών υπηρεσιών, ενώ στη συνέχεια θα παρατεθούν πλεονεκτήματα και μειονεκτήματα βάσει αυτών τα οποία έχουν προτείνει μέχρι στιγμής.

- **HP E-speak**
- **SUN JINI**
- **Microsoft BizTalk**

Μεταξύ των τεχνολογικών προτάσεων που αναφέρθηκαν εντοπίσαμε ορισμένες εγγενείς αδυναμίες, περιορισμούς ή και πλεονεκτήματα αντίστοιχα για κάθε μια από αυτές. Συνυπολογίζοντας όλες αυτές τις παραμέτρους και έχοντας σαν στόχο την επιλογή της καταλληλότερης για την πρακτική εφαρμογή του μοντέλου των ηλεκτρονικών υπηρεσιών καταλήξαμε σε αυτή του e-speak. Η συγκεκριμένη πλατφόρμα καλύπτει σε μεγάλο βαθμό τις ιδιαιτερότητες του μοντέλου ενώ παρέχει και την δυνατότητα ανάπτυξης εφαρμογών σε τοπικό επίπεδο και με διαθέσιμα μέσα (υλικό, λογισμικό, υπολογιστική ισχύς).

Στο πρακτικό σκέλος της διπλωματικής περιγράφεται με αρκετή προσέγγιση ένας νέος τρόπος λειτουργίας των υπαρχόντων χρηματοοικονομικών μοντέλων μέσα από το μοντέλο των υπηρεσιών. Πιο συγκεκριμένα, η εφαρμογή αναφέρεται σε ένα χρηματιστηριακό παράδειγμα, όπου εμπλέκονται όλοι οι πραγματικοί φορείς. Συμμετοχή έχουν οι πελάτες, οι χρηματιστές, οι τράπεζες και η έμπιστη τρίτη οντότητα, για παράδειγμα η χρηματαγορά (Χ.Α.Α), όπου καταγράφονται όλες οι συναλλαγές που λαμβάνουν χώρα μεταξύ των προαναφερόμενων μελών. Έχουμε τη δυνατότητα αφενός να χρησιμοποιήσουμε ένα ελάχιστο σύνολο ρόλων ώστε να υλοποιηθεί μια συναλλαγή (δύο πελάτες, μια τράπεζα, ένας χρηματιστής), έως πολύπλοκες δομές πολλών πελατών, πολλών τραπεζών, πολλών χρηματιστών, εξαρτώμενοι πάντα από τους υπολογιστικούς πόρους που είναι διαθέσιμοι.

Όλες οι οντότητες που συμμετέχουν έχουν την μορφή υπηρεσίας. Κοινό υπόβαθρο αποτελεί η πλατφόρμα του e-speak, η οποία επιτρέπει την καταχώρηση νέων υπηρεσιών, την αναζήτηση-ανακάλυψη, την διαπραγμάτευση και την σύνθεση

των υπηρεσιών. Στην παρούσα φάση υποστηρίζονται όλες οι λειτουργίες εκτός από αυτή της σύνθεσης, καθώς δεν υποστηρίζεται από την τρέχουσα έκδοση της πλατφόρμας του e-speak.

Περιγραφή της εφαρμογής

Τα βασικά συστατικά της εφαρμογής αποτελούν οι συμμετέχουσες οντότητες, αλλά και οι σχέσεις που διαμορφώνονται μεταξύ τους. Συνοπτικά, ως γνωστόν ο πελάτης είναι αυτός που επιδιώκει να συνδιαλλαγεί με άλλους πελάτες προβαίνοντας με αυτόν τον τρόπο σε συναλλαγές. Άμεση είναι η σχέση του με κάποιο τραπεζικό ίδρυμα - την Τράπεζα- η οποία είναι υπεύθυνη για την αποδοχή ή αποστολή χρημάτων σε άλλες τράπεζες ή σε λογαριασμούς της ίδιας, προκειμένου να εισπραχθούν/αποδοθούν τα ποσά από την αγορά ή πώληση μετοχών. Στην τράπεζα κατατίθενται επίσης και χρηματικά ποσά των πελατών, προκειμένου να χρησιμοποιηθούν ως έναντι για την αγορά-προμήθεια νέων μετοχών. Επίσης, το ποσό το οποίο προκύπτει από την πώληση μετοχών του σχετικού καταθέτη, πιστώνεται αυτόματα στον λογαριασμό του.

Ο χρηματιστής είναι μια άλλη οντότητα, εξίσου σημαντική. Έρχεται σε επαφή με τους πελάτες και στο εξής δρα ως πληρεξούσιος τους. Ενημερώνεται από την χρηματαγορά για τις διαθέσιμες προς διαπραγμάτευση μετοχές, αναλαμβάνοντας εν συνεχεία την εκπλήρωση της όποιας συναλλαγής επιλέξει ο πελάτης του. Η οντότητα-υπηρεσία "Χρηματιστής" δέχεται εντολές ώστε να πουλήσει ή να αγοράσει μετοχές, κλείνοντας επίσης την συμφωνία με την τράπεζα. Έντονη είναι η σχέση του χρηματιστή με την οντότητα χρηματαγορά, από όπου πληροφορείται τις κινήσεις άλλων αντίστοιχων υπηρεσιών- χρηματιστών.

θεωρούμε την ύπαρξη δύο πελατών που εκφράζουν την επιθυμία συναλλαγής ενεργοποιώντας την αντίστοιχη-υπηρεσία πελάτη. Μετά την ενεργοποίηση της γίνεται άμεσα η επιλογή τράπεζας και χρηματιστή από τους διαθέσιμους. Η διαθεσιμότητα τράπεζας/ων ή του χρηματιστή/ων, γίνεται μέσα από την περιγραφή των ιδιοτήτων τους μέσω XML και την σύνδεση τους στην υποκείμενη πλατφόρμα του e-speak. Στην συνέχεια ο πελάτης (δύο στη συγκεκριμένη περίπτωση), ανοίγει έναν λογαριασμό στην τράπεζα την οποία έχει επιλέξει. Στον λογαριασμό του καταθέτει όσα χρήματα επιθυμεί. Η ύπαρξη του λογαριασμού είναι απαραίτητη προκειμένου σε αυτόν να αποθηκεύονται τα αποτελέσματα των συναλλαγών του κατόχου. Επίσης, ο πελάτης έχει την δυνατότητα προσθήκης στον λογαριασμό του όλων των μεριδίων μετοχών που έχει στην κατοχή του. Η ίδια διαδικασία ακολουθείται από όλους τους πελάτες που μπαίνουν στην αγορά. Μετά την επιλογή τράπεζας ακολουθεί η επιλογή χρηματιστή με τις ίδιες προϋποθέσεις.

Οι πελάτες στην συνέχεια έχουν την δυνατότητα να πληροφορηθούν για τις διαθέσιμες μετοχές που υπάρχουν για πώληση/αγορά στα πλαίσια της χρηματαγοράς, μέσω του χρηματιστή τους. Κατόπιν, θέτουν τις εντολές τους στον χρηματιστή τους, για παράδειγμα πώληση 100 μετοχών της εταιρείας Χ. Ο χρηματιστής με τη σειρά του αναζητά άλλες αντίστοιχες αιτήσεις για πώληση των ίδιων μετοχών από πελάτες που έχουν εκφράσει το αίτημα τους είτε στον ίδιο, είτε σε άλλους χρηματιστές στα πλαίσια της χρηματαγοράς. Αυτό που πρέπει να σημειωθεί είναι η έντονη παρουσία της διαμεσολάβησης, τόσο στα πλαίσια της χρηματαγοράς η οποία λειτουργεί ως

διαμεσολαβητής αιτημάτων μεταξύ χρηματιστών, όσο και στα πλαίσια των χρηματιστών οι οποίοι λειτουργούν ως εκπρόσωποι και διαμεσολαβητές για τους πελάτες τους. Αν το αίτημα του πελάτη δεν καλυφθεί αμέσως, μπαίνει σε λίστα αναμονής, εγείροντας την ένδειξη της πώλησης, περιμένοντας για σχετικές αιτήσεις.

Οι υπόλοιποι χρηματιστές λαμβάνουν γνώση για το αίτημα πώλησης του συγκεκριμένου πακέτου μετοχών. Ακολούθως ένας άλλος πελάτης θέτει ένα αίτημα αγοράς όλου ή ενός μέρους του προαναφερθέντος πακέτου. Ο χρηματιστής με τη σειρά του ανακαλύπτει την ταύτιση των αιτημάτων που υπάρχει και πληροφορεί τον πελάτη του θέτοντας του αίτημα για διαπραγμάτευση με τον κάτοχο της προσφοράς. Αν το αίτημα γίνει αποδεκτό, ενεργοποιείται η φάση της διαπραγμάτευσης της μετοχής ανάμεσα σε πωλητή και αγοραστή. Κατά τη διάρκεια της διαπραγμάτευσης υπάρχει η ευχέρεια πρότασης και αντιπρότασης τιμών.

Η εξέλιξη της διαδικασίας οδηγεί όπως και κάθε διαπραγμάτευση σε συμφωνία ή διαφωνία. Αν η διαπραγμάτευση αποτύχει, τότε το καθεστώς παραμένει ως έχει χωρίς αλλαγές στους λογαριασμούς των πελατών, ενώ το αίτημα εξακολουθεί να υφίσταται αναμένοντας μία νέα διαδικασία διαπραγμάτευσης εξ' αρχής, με όρους και συνθήκες όμοιες με αυτές που προαναφέρθηκαν. Σε περίπτωση συμφωνίας, ο αγοραστής έχει αυτόματα στον λογαριασμό του τις μετοχές που μόλις αγόρασε με αντίστοιχη μείωση του διαθέσιμου χρηματικού ποσού στον λογαριασμό του, ενώ ο πωλητής έχει και αυτός με τη σειρά του πιστωμένο στον λογαριασμό του το αντίστοιχο συμφωνηθέν ποσό. Η συναλλαγή κλείνει και το αίτημα που την εξέφρασε αποσύρεται από την διαπραγμάτευση.

ΚΕΦΑΛΑΙΟ 2ο

EXECUTIVE SUMMARY

2 EXECUTIVE SUMMARY

The new vision focuses on a model where "service" in terms of a computer unit is treated as the kernel.. Speaking of e-services, we refer to services available through Internet accomplishing various tasks, resolve problems, or integrate transactions. In virtual situation in the future, one could assert that every commodity, starting from S/W and H/W to business processes, data and experience (empirical knowledge) could be available as an e-service, leading in new streams of business activation. Internet already tends to change in a global market where such kind of services can be developed, automatically detected and transformed into more enhanced services.

The e-services model forms an environment where functionality and alternative possibilities will be offered and supported by separate and independent entities. The increase in offering a wide range of services that cover all branches of industry (not only the computer science industry) will directly affect our lives. A basic feature of this change is the transmission from the capital perception of goods into one where charge is proportional to usage. According to this concept, we are likely to have in future much more usage/charge of computing power, storage, e.t.a., as it happens up to now with usage/charge of the electricity.

E-services are likely to form a bridge uniting the typical meaning of an I/T function with the next generation of usage-oriented applications. It will not come as a surprise if everything- from computing power to web-sites is developed, advertised and delivered as e-services. Other examples of application could include faraway file storage or electronic preparation and submission of tax returns. In particular, the environment that is being developed will strongly affect three basic sectors: Basic I/T functions, application delivery and next generation enhanced e-services.

The ulterior e-services object lies beyond the provision (supply) of the conventional IT functions mentioned above on the combination of the existing IT functions in order to configure new, more enhanced services. We could mention for example the next generation of auction services. Instead of simply matching the offers for an item, in future they will "activate" some auctioneers, who will define a price and preference width and will bid for the best result in co-operation with the proper finance institute. We can easily top off the limits of the above-mentioned scenario by moving on to more complicated cases that only certain services under combination can configure.

With co-operating e-services we can envisage a totally different structure of business model, where different functional parts can be considered as separate services. We can use these services according to our needs. By fully separating and giving autonomy to separate segments, if some of these do not come up to our expectations, they can easily be outsourced. if By this way, we take problems out of a given business aiming to improve it. Since a company depends on independent services, no disruption can take place.

Expected benefits for customers/users of the new model of services include liability, cost cutback, and security of the informative infrastructure. Within the new environment, users have access in applications that are continuously improving. For example, recently checked software versions can be delivered to users with no need to

wait for the new version. The functionality of the new model is measured not only by the number of the offered services but by their availability as well. The I/T's high level of performance is going to meet the expectations of the users.

Based on their strong points, service providers can grant their customers the advantages of the distributed architecture and continuous support as well. Being able to re-use basic elements, they offer scale economy in I/T environment. ASP's that have been already mentioned are even present in offering ERP services through Internet.

The implementation of services model is possible even in the interior of an organisation. The transition from the current model to services model will keep executive officers of organisations preoccupied. The decision whether some internal procedures can be re-organised constitutes an old dilemma. We will present the details of such an application as well as the advantages and disadvantages of such an implementation.

Concerning IT departments within different organisations, these are often considered to be more cost production centres despite their important contribution. The same assumption stands for some other departments, such as the Finance department. The competition of the particular sector with other service providers is either little or non-existentand mainly on security issues. This isolation, results in closed executive teams that lack innovation, give feedback in delay, evade creative risk and produce high costs.

The question that arises is what is going to be the new role for the IT departments within organisations. What is expected to happen is that IT departments be transformed directly in brokers or internal application consultants, managing organisations accordingly.

In future where decisions concerning internal development or purchase will tend to be in favour of the latter, the role of the broker/consultant will be very important. Seeing that the scenario according to which the internal users choose directly the services that consider to be necessary is unrealistic, the task of selection and evaluation will now fall in the hands of the staff of IT departments. Leadership will now be necessary for suggesting ways of undertaking and implementing business plans using the proper means and technology.

The e-services market model aims to define the basic elements of a more general e-services market. These elements represent a unique set of contributors that form the foundation for e-services creation and materialisation. These elements are:

- **Transaction Participants:** they represent the parties that participate in a transaction. A transaction may implicate two or more teams.
- **Brokers:** they represent the entities that bring in contact Byers and Service Providers, so that an agreement takes place.
- **Market Makers:** they represent the entities that ensure that the transaction takes place as agreed. This is a basic element of the transaction, given that it builds confidence which is necessary for the fulfilment of commercial transactions.
- **Finance Managers:** they represent the entities supporting the finance aspects of an electronic transaction.

- **Lifecycle Managers:** they represent the entities that manage all phases of a service lifecycle, right from the point that a service is being created up to the point that it is being used. Their duties include management of composition, supply, development and realisation functions.

Every e-service does not necessarily contain all the above mentioned elements. An electronic transaction can be as simple as a low risk transaction among a minimum number of participants demanding no contract or so complicated that it may demand an extensive support system with many anonymous participants.

Reference to the e-services model creates the necessary basis for its application in particular business models. The term "business model" is not consistent in global e-commerce bibliography. However, we can detect cases where e-services have been applied within terms of current business models. We can refer to the following business models where e-services can be applied.

- E- Procurement
- E-auction
- Third Party MarketPlaces
- Virtual Communities

From now on and with the step-by-step use of the new model of services is starting to face a new reality. This is the second phase that Internet passes and has major changes from the previous one. The companies will be antagonize in a different base, where the relation between offer-price, the quality and the speed of delivery are the main issues. From their side, the customers will have the availability of using all these services through several devices (mobile phones, palmtops, tv etc) and only through the classical desktop computer. Final the service providers like ISP's, ASP's and Telco's taking advantage from the new technological proposals -like the next above- will adopt a new vision for the market.

Main competitors of this area are Sun Microsystems, HP and Microsoft. The main thoughts and proposals will be follow and also the main advantages and disadvantages. Namely we have:

- **HP E-speak**
- **SUN JINI**
- **Microsoft BizTalk**

Between them we distinguished some internal weaknesses, constraints and advantages also. Making a summary of them and having as a main target the selection of the best solution, I decided to select HP e-speak. The specific platform supports a great part of the service model since also gives the opportunity of developing applications in a local space with all available means (Hardware, Software, computational power).

At the practical scale of thesis is described with quite great approximation a new way of work for the existing stock-financial models through the service models. In more details, this application refers to a stock exchange example and are included all the real conveyors. Participate the customers, the stock brokers, banks and the trust

third party-entity where all the exchanges between customers are written down. We have the opportunity to use a short part of roles in order to proceed an exchange (two customers, one bank, one stock broker) or multiple structures of many customers, many banks many stock exchangers, depending on the available computational resources.

All the entities which participate are services. Common ground is the e-speak platform that allows the registration, the detection-discovery, the brokering and the composition of new services. In this phase all the functions are available except the composition part.

Application description

The main elements of this application are the participated entities and the relations between then. Synoptically, customer wants to trade with other customers setting down some exchanges. There is and an other direct relation with bank which is responsible for the acceptance or the remittent of money in accounts of the same bank or others one in order to give/collect the amount of money from the buy/sell of stocks. In bank are deposited the money in accounts that are used for the buy of stocks. Also the amount of the sell of stocks is credit directly to the account of the relevant customer.

Stock Trader is another entity with great meaning. He has relation with his customers and he acts as a proxy for them. He is all time informed for the available stocks in the marketplace taking the responsibility of carrying out the fulfillment of his customer's exchange. The entity-service "Broker" accepts orders for sell/buy closing also the exchange with banks. There is also strong relationship with the enity "Stock Market" from where he is informed for the movements of the other brokers.

We consider the existence of two customers, that they want to start an exchange starting up the customer service. After the activation follows the selection of the relevant bank and broker. The availability of the banks and brokers is described with an XML file which is registered in e-speak platform. Continuously customer opens an account and deposits as much money as he wants and also his stocks(if he has any). The same process is followed by all customers that insert in the market.

After this customers are informed for the available stocks in the market for buy/sell. After that they set their orders for sell/buy through their brokers. For example sell of 100 stocks of company X. Stock broker tries to find other appropriate orders that are expressed to him or to other stock brokers in the market. We must insist on the strong appearance of brokering as in the marketplace which acts like a proxy between brokers, as also between brokers that act like a proxy between customers. If the customer order doesn't fulfilled it is getting in list of requests starting a sell flag waiting for the right order.

The rest of the brokers take knowledge of the sell order. After that an other customer sets an order for buy of all/part of the packet. The broker finds the relevance between sell and buy order and informs his customer starting a negotiation flag. If the request is accepted the negotiation phase between buyer and seller is activated. This phase can has may steps until a common price founded.

The developing of the process stops with an agreement or disagreement. If we have at the end a disagreement the status is the same as it was before the negotiation without changes in accounts. Also the request is still existing waiting for a new negotiation process. Otherwise buyer has the stocks in his account with the reduction of his money. Seller looses his stocks but he gets the negotiated amount of money. The exchange closes and the request stops to be valid.

ΚΕΦΑΛΑΙΟ 3ο

ΠΕΡΙΓΡΑΦΗ ΤΩΝ ΣΥΝΕΡΓΑΖΟΜΕΝΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΗΡΕΣΙΩΝ

3 ΠΕΡΙΓΡΑΦΗ ΤΩΝ ΣΥΝΕΡΓΑΖΟΜΕΝΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΗΡΕΣΙΩΝ

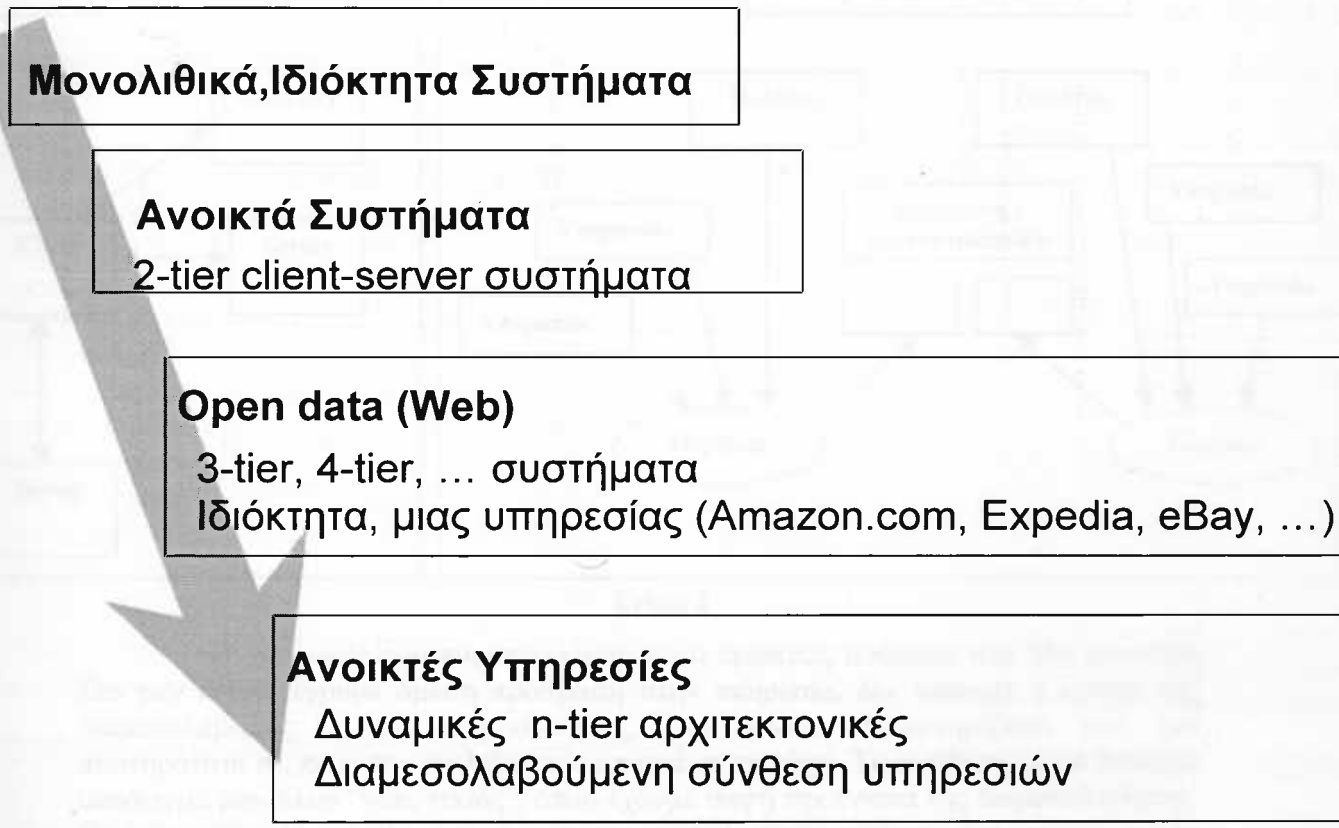
Η επανάσταση αποτελεί σήμερα έννοια συνώνυμη με τον χώρο της πληροφορικής. Μόλις 20 χρόνια πριν, ο κόσμος κινούνταν στην εποχή των mainframes συστημάτων. Ελάχιστοι χρήστες είχαν πρόσβαση σε αυτούς τους υπολογιστές, ενώ παράλληλα περιορίζονταν η πρόσβαση στον πλησιέστερο χώρο εργασίας. Η εποχή άρχισε σταδιακά να διαφοροποιείται με αποτέλεσμα τα PC's και η γραφική διεπαφή τους να εκδημοκρατικοποιήσουν τον χώρο των υπολογιστών κοινοποιώντας τις "αρετές" τους σε δεκάδες εκατομμυρίων ατόμων ανά τον κόσμο. Στην συνέχεια, ο κόσμος των επιχειρήσεων διαπίστωσε ότι τα δίκτυα των υπολογιστών και η Client-Server αρχιτεκτονική θα άλλαζε τον τρόπο λειτουργίας τους. Ακολούθησε το Internet το οποίο άλλαξε καθολικά τον τρόπο με τον οποίο επικοινωνούμε, αποτελώντας ένα νέο μέσο παροχής πληροφορίας αλλά και ψυχαγωγίας. Το σημαντικότερο όμως που κατόρθωσε, βάζοντας υποθήκη για το μέλλον, ήταν η προσθήκη του "e" στο commerce, διαμορφώνοντας νέα επιχειρηματικά μοντέλα, αλλά και εργαλεία υποστήριξης αυτών. Σήμερα σύμφωνα με επίσημες στατιστικές μελέτες, 300 εκατομμύρια άνθρωποι είναι συνδεδεμένοι στο διαδίκτυο, ενώ υπολογίζονται σε 250 δις δολάρια οι συναλλαγές που θα πραγματοποιηθούν κατά το τρέχον έτος.

Πέρα όμως από τα όσα θαυμαστά αναφέρθηκαν για τα άλματα που παρατηρήθηκαν, υπάρχει αρκετός χώρος για βελτίωση. Σήμερα το Ιντερνετ εξακολουθεί να αποτελεί ένα μεγαλύτερο καθρέπτη του αρχικού μοντέλου των mainframe συστημάτων. Παρά το μεγάλο συγκριτικά εύρος ζώνης, η πληροφορία εξακολουθεί να παραμένει κλειδωμένη σε μεγάλες κεντρικές βάσεις δεδομένων πίσω από πολλά μέτρα ασφαλείας. Οι χρήστες πρέπει να βασίζονται σε ένα Web Server προκειμένου να πραγματοποιήσουν καθετί, προσομοιάζοντας σε μεγάλο βαθμό το παλαιό μοντέλο διαμοίρασης χρόνου (time-sharing model). Επίσης σήμερα μέσω των HTML σελίδων, δίνεται μεγαλύτερη έμφαση στην εμφάνιση της πληροφορίας παρά στο περιεχόμενο της, ενώ παράλληλα οι browsers παίζουν τον απλό ρόλο ενός read-only τερματικού, χωρίς να δίνουν τη δυνατότητα στους χρήστες να αναλύουν και να διαχειρίζονται την πληροφορία. Όλα όσα αναφέρθηκαν, αποτελούν περιορισμούς που μπορούμε να διαγνώσουμε στην παρούσα φάση. Στη συνέχεια, θα αναφερθούμε στις προσεγγίσεις οι οποίες αναμένονται να ακολουθήσουν.

Το νέο όραμα που οδηγεί τις εξελίξεις αφορά το νέο μοντέλο που πυρήνα έχει την "υπηρεσία" ως υπολογιστική μονάδα. Μιλώντας για ηλεκτρονικές υπηρεσίες αναφερόμαστε σε υπηρεσίες διαθέσιμες μέσω του Internet, οι οποίες φέρουν εις πέρας διάφορα καθήκοντα, επιλύουν προβλήματα ή ολοκληρώνουν συναλλαγές. Μελλοντικά, σε μια ιδεατή κατάσταση θα μπορούμε να ισχυριστούμε ότι κάθε αγαθό από S/W και H/W έως επιχειρηματικές διαδικασίες, δεδομένα και εμπειρική γνώση μπορεί να γίνουν διαθέσιμα σαν ηλεκτρονικές υπηρεσίες, οδηγώντας σε νέα ρεύματα επιχειρηματικής δραστηριοποίησης. Το Web ήδη τείνει να μετασχηματιστεί σε μια τεράστια αγορά όπου τέτοιου είδους υπηρεσίες μπορούν να αναπτυχθούν, να ανακαλυφθούν δυναμικά και να μετασχηματιστούν συνεργαζόμενες σε ανωτέρου επιπέδου υπηρεσίες.

Σύμφωνα με τις προσδοκίες του Joel Birnbaunn - ερευνητή, όπως αυτές εκφράστηκαν στην Αμερικανική ένωση Φυσικών το 1999, το όραμα θα εκπληρωθεί όταν "η σύγχρονη τεχνολογία της πληροφορικής θα γίνει τόσο ευρεία και εύκολη, όσο η χρήση του ηλεκτρικού ρεύματος, του νερού, ή του τηλεφώνου στο σπίτι". ([4])

Το μοντέλο των υπηρεσιών διαμορφώνει ένα κόσμο όπου η λειτουργικότητα και οι δυνατότητες θα προσφέρονται και θα υποστηρίζονται από αυτόνομες, ανεξάρτητες οντότητες. Παρέχοντας διαρκώς μια ποικιλία υπηρεσιών σε όλο το εύρος των κλάδων της βιομηχανίας (όχι κατ' ανάγκην της πληροφορικής βιομηχανίας μόνο), θα προκαλέσει άμεσες επιδράσεις στη ζωή μας. Βασικό στοιχείο της αλλαγής αποτελεί η μεταφορά από την κεφαλαιουχική αντίληψη των αγαθών, σε μια αντίληψη όπου η χρέωση διαμορφώνεται σύμφωνα με την χρήση που γίνεται. Έτσι, με βάση αυτή την αρχή, στο μέλλον αναμένεται να δούμε σε μεγαλύτερη έκταση χρήση και χρέωση υπολογιστικής ισχύος, αποθήκευσης κ.τ.λ, όπως ακριβώς συμβαίνει σήμερα με τη χρήση και τη χρέωση του ηλεκτρικού ρεύματος.



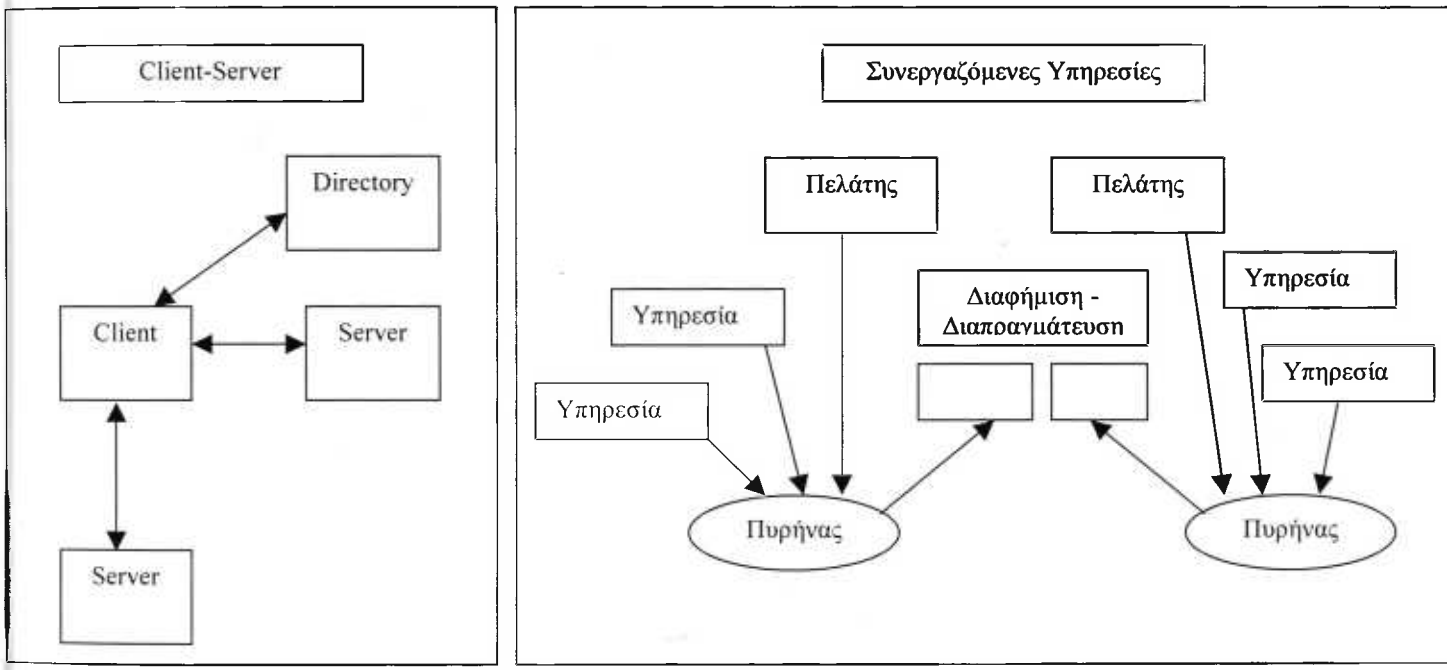
Σχήμα 1

Κάνοντας μια σύνοψη παρατηρούμε ότι αρχικά υπήρχαν τα μονολιθικά και αυστηρώς ιδιόκτητα συστήματα μεγάλων οργανισμών που είχαν και την αποκλειστική χρήση τους. Στην συνέχεια έχουμε την είσοδο της Client-Server αρχιτεκτονικής όπου τα συστήματα ανοίγονται σε ευρύτερο κοινό. Το συγκεκριμένο μοντέλο εξακολουθεί ακόμη και σήμερα να παρέχει κάλυψη σε αρκετές εφαρμογές. Τα πράγματα όμως έφεραν στο προσκήνιο τις 3-tier, 4-tier αρχιτεκτονικές μέσω του WEB. Στα πλαίσια αυτά έχουν κάνει την εμφάνιση τους ιστοσελίδες που παρέχουν συνήθως ενός είδους υπηρεσίες. Αυτό που πλέον προτείνεται από το μοντέλο των συνεργαζόμενων υπηρεσιών είναι η ύπαρξη πολυεπίπεδων αρχιτεκτονικών όπου οι υπηρεσίες διαπραγματεύονται και συντίθενται δυναμικά.

Μια λογική εξέλιξη του μοντέλου του Internet προς ένα μοντέλο υπηρεσιών, περιλαμβάνει επίσης μια συνεχόμενη μετατόπιση από τις βαριές Client-Server

αρχιτεκτονικές σε πιο ευέλικτα υπολογιστικά μοντέλα. Στα πλαίσια αυτών των μοντέλων υποστηρίζεται μια δομή όπου διάφοροι παροχείς υπηρεσιών μικροί ή μεγάλοι, ανταγωνίζονται για να παραδώσουν στους πελάτες τις υπηρεσίες που αυτοί επιθυμούν.

Το ιδεατό που αναμένεται (μετά από πολλά χρόνια εξέλιξης) είναι το γεγονός διάφορες υπηρεσίες να συντίθενται από "υλικά-υπηρεσίες" πολλών διαφορετικών παροχέων. Μια σχηματική σύγκριση ανάμεσα στην κλασσική client-server αρχιτεκτονική και αυτής του νέου προτεινόμενου μοντέλου είναι η εξής:



Σχήμα 2

Οι διαφορές που παρατηρούνται είναι εμφανείς ανάμεσα στα δύο μοντέλα. Στο μεν πρώτο έχουμε άμεση πρόσβαση στην υπηρεσία, δεν υπάρχει η έννοια της διαμεσολάβησης ή της εκπροσώπησης, ενώ γενικά χαρακτηρίζεται από μια αυστηρότητα ως προς την ευελιξία που μπορεί να επιτύχει. Σε αντίθεση με το δεύτερο υπόδειγμα μοντέλου "νέας γενιάς", όπου έχουμε σαφή την έννοια της διαμεσολάβησης. Οι υπηρεσίες (όπως θα δούμε και πιο κάτω) παρουσιάζουν ένα ενιαίο τρόπο ανάπτυξης. Η ύπαρξη των υπηρεσιών επισφραγίζεται με την καταχώρηση - δήλωση στον πυρήνα, ο οποίος παίζει και τον ρόλο του διαμεσολαβητή. Η δήλωση των υπηρεσιών στον πυρήνα γίνεται βάσει των ιδιοτήτων που προσφέρει η κάθε υπηρεσία και συνήθως γίνεται μέσω λεξιλογίου- παρεχόμενων ιδιοτήτων σε XML μορφή.
Κατά συνέπεια, η αναζήτηση υπηρεσιών από τους πελάτες γίνεται βάσει συγκεκριμένων ιδιοτήτων και όχι στηριζόμενη σε εξωτερικά δεδομένα, όπως για παράδειγμα το όνομα.

3.1 Χώρος που διαμορφώνεται

Οι ηλεκτρονικές υπηρεσίες αναμένεται να διαμορφώσουν μια γέφυρα μετάβασης από τη κλασσική έννοια της λειτουργίας της I/T, σε μια επόμενη γενιά εφαρμογών προσανατολισμένες στην προς-χρήση αντιμετώπιση. Δεν είναι καθόλου παράδοξο να δούμε τα πάντα, από υπολογιστική ισχύ έως web-sites να διαμορφώνονται, να διαφημίζονται και να παραδίδονται σαν ηλεκτρονικές υπηρεσίες.

Άλλα παραδείγματα εφαρμογής περιλαμβάνουν την απομακρυσμένη αποθήκευση αρχείων, ή την βασισμένη στο δίκτυο προετοιμασία και υποβολή της φορολογικής δήλωσης. Πιο συγκεκριμένα, το πεδίο που όπως αναφέραμε διαμορφώνεται, θα έχει σημαντική επίδραση πάνω σε τρεις τομείς: Βασικές λειτουργίες I/T, application delivery και επόμενης γενιάς συνδυαζόμενες ηλεκτρονικές υπηρεσίες.

Το παρακάτω σχήμα περιγράφει το εύρος χρήσης, παρέχοντας παραδείγματα υπηρεσιών τα οποία υπάρχουν ή θα υπάρξουν στο μέλλον.

Βασικές λειτουργίες I/T
- Εγκατάσταση υποδομής
- Εξειδικευμένη υπολογιστική ισχύς
- Αποθήκευση& Backup
- Ασφάλεια
- Διαχείριση συστημάτων και παρακολούθηση
- Εργαλεία ανάπτυξης

Παράδοση Υπηρεσιών
- Μετάδοση μηνυμάτων (e-mail κ.τ.λ)
- Χρονοπρογραμματισμός
- Ανθρώπινοι Πόροι
- Οικονομικές Υπηρεσίες
- Παιχνίδια κ.τ.λ.

Συνδυαζόμενες Υπηρεσίες
- Συνδεδεμένες Κάθετες Αγορές
- Συμβουλευτικές Υπηρεσίες
- Προσωποποιημένες Κάθετες Υπηρεσίες
- Διαχείριση και παράδοση Content
- Διαμεσολάβηση για content ή εμπόριο

Οι βασικές λειτουργίες I/T αποτελούν έναν από τους πρωταρχικούς στόχους των υπηρεσιών. Κατευθυνόμενοι προς αυτήν την αντιμετώπιση, παρέχεται μεγάλη ευελιξία, αξιοπιστία και μείωση κόστους για τους περισσότερους πελάτες αυτών των υπηρεσιών. Η μετάβαση έχει αρχίσει ήδη να συντελείται. Η ύπαρξη των ASP's – Application Service Providers (κυρίως στο εξωτερικό), είναι πλέον γεγονός. Η παροχή εκ μέρους τους χώρου (φυσικού και «υπολογιστικού») για την απομακρυσμένη εγκατάσταση της πληροφοριακής υποδομής μιας επιχείρησης, είναι πραγματικότητα. Επιπλέον, πολλά είναι εκείνα τα Websites τα οποία εδρεύουν και λειτουργούν βασισμένα σε τέτοιες υπηρεσίες. Όσο περισσότερες επιχειρήσεις και πελάτες τείνουν να γίνουν όλο και περισσότερο δικτυωμένοι, τόσο περισσότερες κλασσικές λειτουργίες IT θα ενσωματωθούν στο πρότυπο των υπηρεσιών.

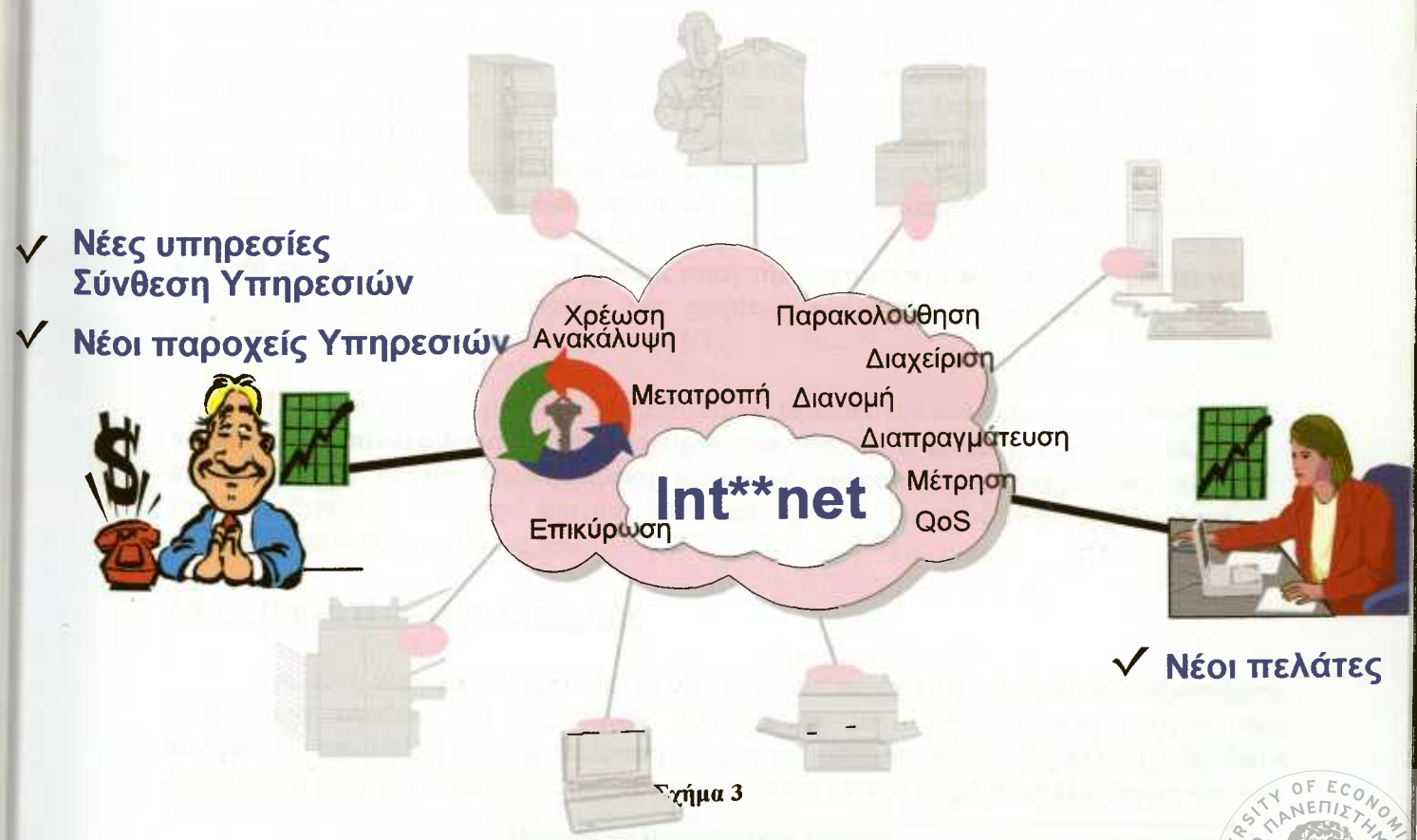
Η Παράδοση υπηρεσιών αποτελεί ένα δεύτερο κατά σειρά στόχο του μοντέλου των υπηρεσιών. Σήμερα, η παραγωγή βασίζεται κυρίως στο desktop περιβάλλον των εργαζομένων, ενώ πλήρη είναι τα εταιρικά συστήματα διαχείρισης τα οποία χρησιμοποιούνται από τις επιχειρήσεις. Οι περισσότερες από αυτές τις εφαρμογές είναι τοπικά εγκατεστημένες και συντηρούνται από το προσωπικό πληροφορικής της επιχείρησης. Το μοντέλο των υπηρεσιών παρέχει ένα εναλλακτικό μοντέλο το οποίο έχει αρχίσει να υιοθετείται σε πολλές περιπτώσεις. Για παράδειγμα, μικρές επιχειρήσεις χωρίς μεγάλη πληροφοριακή υποδομή και ταυτόχρονα με διάθεση να κρατήσουν χαμηλά την πολυπλοκότητα, ενσωματώνουν σταδιακά την ιδέα για εφαρμογές εξωτερικά παρεχόμενες (π.χ. ηλεκτρονικό ταχυδρομείο κ.τ.λ) με τη μορφή υπηρεσιών. Έτσι, πληροφοριακές υπηρεσίες, όπως e-mail, ημερολογιακός προγραμματισμός, παιχνίδια κ.τ.λ, βαίνουν προς τη μετάπτωσή τους στο νέο μοντέλο.

Για ορισμένους όμως τομείς της αγοράς, οι υπάρχουσες επενδύσεις σε υλικό και λογισμικό και η προσαρμογή των ανωτέρω στις ανάγκες τους αποτελεί ένα σημαντικό εμπόδιο προς την αλλαγή. Για τις περιπτώσεις αυτές έχουν διαμορφωθεί

διάφορα στάδια μετάπτωσης όπου έμφαση για είσοδο στο νέο μοντέλο δίνεται στις νεοεμφανιζόμενες ανάγκες. Οπωσδήποτε, η μετάβαση αυτή σε ένα πλήρες μοντέλο υπηρεσιών χαρακτηρίζεται από κόστος αλλά και από σταδιακά αργές διαδικασίες μετάβασης.

Ο απώτερος στόχος των ηλεκτρονικών υπηρεσιών βρίσκεται πέρα από τη παροχή των συμβατικών IT λειτουργιών που αναφέραμε. Αφορά στη διαμόρφωση υπηρεσιών ανώτερου επιπέδου, συνδυάζοντας τις ήδη υπάρχουσες. Μια πιο προσεχτική ματιά στους αντικειμενικούς στόχους κάθε περίπτωσης, ξεκαθαρίζει απόλυτα τις διαφορές ανάμεσα στις συνεργαζόμενες κ.τ.λ και στις κλασσικές υπηρεσίες των εφαρμογών. Ας πάρουμε για παράδειγμα την επόμενη γενιά των υπηρεσιών δημοπρασίας. Αντί να ταιριάζουν απλά τις προσφορές για κάποιο αντικείμενο, αυτές (στο μέλλον) θα ενεργοποιούν ορισμένους δημοπράτες (όπως θα δούμε και στη συνέχεια) οι οποίοι θα ορίζουν ένα εύρος τιμών και προτιμήσεων και σε πραγματικό χρόνο, σε συνδυασμό πάντα με το αντίστοιχο χρηματοπιστωτικό ίδρυμα, θα πλειοδοτούν για το καλύτερο αποτέλεσμα. Τα όρια τα οποία θέτουμε στο παραπάνω σενάριο είναι εύκολο να ξεπεραστούν προχωρώντας σε πολύπλοκες υποθέσεις τις οποίες μπορούν να διαμορφώσουν μόνο κάποιες συνεργαζόμενες υπηρεσίες.

Οι ηλεκτρονικές υπηρεσίες παρουσιάζουν ένα αξιοσημείωτο κύκλο ζωής. Σύμφωνα με αυτόν τον κύκλο, οι παροχείς υπηρεσιών δημιουργούν νέες ή προβαίνουν σε συνδυασμούς από ήδη υπάρχουσες υπηρεσίες. Το ποιες υπηρεσίες δημιουργούνται πρέπει να αναζητηθεί στους μηχανισμούς της αγοράς, οι οποίοι καθορίζουν όλα όσα εμφανίζονται και εξαφανίζονται στα πλαίσια του ελεύθερου ανταγωνισμού.



Σχήμα 3

Εκτός των νέων υπηρεσιών εμφανίζονται και νέοι προμηθευτές υπηρεσιών, ερχόμενοι να καλύψουν όποιες αδυναμίες ή κενά υπήρχαν. Η "παραγωγή" τους διαφημίζεται στα πλαίσια ενός στενού (εσωτερικό του οργανισμού) ή ευρύτερου πλαισίου (Internet), ή και στα δύο ταυτόχρονα, ανάλογα με το μοντέλο που θα ακολουθηθεί. Στον χώρο όπου οι υπηρεσίες διαφημίζονται, προκύπτει μια σειρά αλληλοσυναρτόμενων θεμάτων που έχουν σχέση με μια σειρά πολύ σοβαρών θεμάτων. Μερικά από αυτά είναι η ανακάλυψη των υπηρεσιών αυτών από τους εν δυνάμει πελάτες τους, η διαπραγμάτευση που οφείλει να γίνει ανάμεσα στα ενδιαφερόμενα μέρη, η εξασφάλιση της ποιότητας της παρεχόμενης υπηρεσίας καθ' όλη τη διάρκεια εφαρμογής της, η χρέωση σύμφωνα με τη χρήση που έγινε κ.α..

Με τις συνεργαζόμενες ηλεκτρονικές υπηρεσίες μπορούμε να έχουμε κατά νου μια τελείως διαφορετική οργάνωση του επιχειρηματικού μοντέλου, όπου τα διάφορα λειτουργικά τμήματά του μπορούν να θεωρηθούν ως αυτόνομες υπηρεσίες. Τις υπηρεσίες αυτές μπορούμε να τις καλέσουμε ανάλογα με τις ανάγκες. Έτσι, με την πλήρη μοντελοποίηση και αυτονόμηση των επιμέρους τμημάτων, αν μερικά από αυτά δεν ανταποκρίνονται στον επιθυμητό βαθμό μπορούν σχετικά εύκολα να γίνουν outsourcing. Με αυτόν τον τρόπο μεταφέρουμε εκτός της επιχείρησης τις αδυναμίες της, στοχεύοντας στην βελτίωση τους. Από τη στιγμή που η επιχείρηση στηρίζεται σε πλήρης ανεξάρτητες υπηρεσίες καμία διάσπαση δεν μπορεί να πραγματοποιηθεί.

3.2 Λόγοι Επιτυχίας

3.2.1 Οφέλη Πελατών / Χρηστών

Για τους πελάτες/ χρήστες του νέου μοντέλου υπηρεσιών τα οφέλη που θα προκύψουν αναμένονται να είναι λόγω της εκτεταμένης χρήσης τους, η αξιοπιστία, η περικοπή του κόστους, καθώς και η ασφάλεια της πληροφοριακής υποδομής. Στο νέο περιβάλλον, οι χρήστες έχουν πρόσβαση σε εφαρμογές (applications) οι οποίες συνεχώς βελτιώνονται. Έτσι για παράδειγμα, νέες εκδόσεις λογισμικού που έχουν ελεγχθεί μπορούν να παραδοθούν άμεσα στους χρήστες χωρίς να υπάρχει η ανάγκη της αναμονής για τη νέα έκδοση. Αλλά η λειτουργικότητα του νέου μοντέλου υπηρεσιών δεν μετράται μόνο από τον αριθμό των προσφερόμενων υπηρεσιών, αλλά και από τη διαθεσιμότητά τους. Έχοντας αυτή την παράμετρο υπ' όψιν, αναμένεται να καλυφθούν πλήρως οι προσδοκίες των χρηστών ικανοποιώντας τα υψηλά επίπεδα απόδοσης τα οποία αναμένονταν από την I/T.

Το συγκεκριμένο μοντέλο αναμένεται να μειώσει την πολυπλοκότητα ιδίως σε περιπτώσεις επανακαθορισμού του configuration. Επίσης ο πελάτης δεν είναι πλέον εγκλωβισμένος σε μια συγκεκριμένη εφαρμογή ενός παροχέα υπηρεσιών, αλλά αντίθετα έχει στη διάθεση του μια λίστα παροχέων, οι οποίοι είναι πρόθυμοι να ικανοποιήσουν τις απαιτήσεις του.

3.2.2 Οφέλη Παροχέων Υπηρεσιών

Για τους παροχείς υπηρεσιών το νέο μοντέλο παρέχει την δυνατότητα αξιοποίησης υπαρχόντων αγαθών και δεξιοτήτων προκειμένου να αυξήσουν τους στόχους των προσφορών τους, αλλά και την επίδραση τους σε νέες αγορές. Μέσω της αλλαγής δίνεται η δυνατότητα παροχής νέων υπηρεσιών δημιουργώντας με την σειρά τους νέες πηγές εσόδων.

Βασιζόμενοι στα δυνατά τους σημεία οι παροχείς υπηρεσιών, μπορούν να παραχωρήσουν στους πελάτες τα πλεονεκτήματα της διαμοιραζόμενης αρχιτεκτονικής, όπως και της συνεχούς υποστήριξης. Έχοντας την δυνατότητα επαναχρησιμοποίησης βασικών συστατικών στοιχείων παρέχουν οικονομίες κλίμακας στο περιβάλλον του I/T. Οι ASP's που ήδη αναφέραμε έχουν κάνει την εμφάνισή τους ακόμη και στην παροχή ERP υπηρεσιών μέσω δικτύου.

3.3 Επιπτώσεις

Οι επιπτώσεις από μια τέτοιου εύρους αλλαγή είναι προφανείς. Οι εκάστοτε ηγεσίες των οργανισμών οφείλουν να κατανοήσουν ότι ο συνδυασμός της δικτυωμένης οικονομίας με το μοντέλο των υπηρεσιών, επιτάσσει την διαφοροποίηση των επιλογών τους για το που και το πως θα ανταγωνιστούν. Για τα στελέχη τα οποία είναι αποστασιοποιημένα από την τεχνολογία και από τον ρόλο της στην νέα επιχειρηματικότητα, η πρόκληση είναι προφανής.

Προκειμένου να υπάρξει πρόοδος, η διοίκηση οφείλει να θέσει ορισμένα βασικά ερωτήματα για το πως ο οργανισμός θα ανταποκριθεί στους νέους στόχους:

1. Τί επιπτώσεις θα έχει η δικτυωμένη οικονομία και το μοντέλο υπηρεσιών στη δομή της επιχείρησης
2. Πως η ηγεσία οραματίζεται τα προϊόντα και τις αγορές σε ένα περιβάλλον όπου τα πάντα μπορούν να προσφερθούν σαν μέρος ή σαν όλον μιας υπηρεσίας
3. Ποια στάση θα πρέπει να τηρηθεί απέναντι στις συμμαχίες και στις συνεργασίες σε ένα περιβάλλον όπου οι υπηρεσίες θα έχουν τη δυνατότητα να συντίθενται επανειλημμένα για την ικανοποίηση των ξεχωριστών αναγκών του πελάτη;
4. Πως οι επιχειρήσεις πρέπει να διαμορφώσουν την πολιτική τους σχετικά με τις επενδύσεις και τα αγαθά της πληροφορικής σε ένα μέλλον όπου όλοι οι πόροι θα είναι διαθέσιμοι από τρίτους σαν αυτόνομες υπηρεσίες

Αναμφίβολα οι εταιρείες για να επιβιώσουν στις νέες συνθήκες οφείλουν να αναπτύξουν αντισώματα σε πολλές επιρροές.

3.3.1 Αυξανόμενη σημασία της ταχύτητας υλοποίησης καθώς και πιο ευέλικτες δομές.

Το μέλλον παρουσιάζει ένα νέο ευρύ σύνολο ευκαιριών για τις επιχειρήσεις. Νέοι δρόμοι ανοίγονται για την προσέγγιση και την ικανοποίηση του πελάτη, νέες προσεγγίσεις δίνουν την δυνατότητα στις επιχειρήσεις να επισπεύσουν την υιοθέτηση των αλλαγών της αγοράς, ενώ νέες ευκαιρίες δημιουργούνται με χρήση εναλλακτικών δομών για την επέκταση των επιχειρήσεων. Εντούτοις, για να εκμεταλλευτούν αυτό το περιβάλλον οι διοικητικές ομάδες των επιχειρήσεων θα πρέπει να συμπεριφέρονται διαφορετικά και κυρίως να παραμείνουν λιτές και να ενεργούν ταχύτερα.

Καταρχήν, τα διευθυντικά στελέχη των επιχειρήσεων θα πρέπει να συνηθίσουν να λαμβάνουν τις αποφάσεις πιο σύντομα. Ένα μέλλον επικεντρωμένο στις υπηρεσίες, αλλάζει ολόκληρη τη φύση της διαδικασίας αγορών. Ο σχεδιασμός ενός προϊόντος, η ανάπτυξη του, η προώθηση του και η εξάπλωση του στην αγορά είναι διαδικασίες οι οποίες μεταμορφώνονται δυναμικά από το Internet και τις ηλεκτρονικές υπηρεσίες. Επιπλέον, συχνά τα χαρακτηριστικά του προϊόντος που "διαφημίζεται" είναι γνωστά στους ενδιαφερόμενους. Κατά συνέπεια, η ανταγωνιστική ζωή ενός προϊόντος ή

υπηρεσίας συνεχώς μειώνεται. Γενικότερα, σε ένα περιβάλλον εύκολου κέρδους, γρήγορης ανάπτυξης, άμεσης διανομής και γρήγορης αποτίμησης, δεν επαρκεί ο παραδοσιακός κύκλος σχεδιασμού και λήψης αποφάσεων. Για τις εταιρείες οι οποίες καταβάλλουν έντονες προσπάθειες στον επαγγελματικό χώρο και έχουν βαθιά επίγνωση ή εκτελεστική ικανότητα, η διαδικασία ανάπτυξης των προϊόντων δεν αποτελεί πρόβλημα. Αντιθέτως, έξω από τις εξελίξεις μένουν ορισμένες εταιρείες οι οποίες χαρακτηρίζονται από μια απροθυμία να αναλάβουν νέα δράση.

Το σκεπτικό των ηλεκτρονικών υπηρεσιών αποτελεί πρόκληση για τον κόσμο των επιχειρήσεων. Τα ατελείωτα πλάνα σχεδιασμού και οι απαιτήσεις δεδομένων, πρέπει να δώσουν τη θέση τους στην δράση. Το δύσκολο είναι ότι μέχρι πρότινος η έννοια του πειθαρχημένου σχεδιασμού αποτελούσε το παν. Κάτι τέτοιο όμως στις μέρες μας δεν ισχύει και θα πρέπει να γίνει κατανοητό ότι το κόστος και το ρίσκο της αδράνειας θα είναι πιο σοβαρά από τα ενδεχόμενα μιας περιορισμένης αποτυχίας. Στο άμεσο μέλλον, το γεγονός ότι ένα προσεκτικά σχεδιασμένο πρόγραμμα δράσης, το οποίο όμως γίνεται προκαταβολικά δεν θα αποτελεί αποδοτική στρατηγική.

Κατά δεύτερον, τα διευθυντικά στελέχη θα πρέπει να επανεκτιμήσουν τα θεμελιώδη όρια των δραστηριοτήτων τους και σκόπιμα να ορίσουν τον σκοπό των επιχειρήσεων τους. Η ευκαιρία που παρουσιάζεται είναι ξεκάθαρη. Η διαδικασία διαμόρφωσης της αγοράς είναι προφανής, το κόστος συνεργασίας είναι εφικτό και το μεγαλύτερο τμήμα των εσωτερικών διεργασιών - σε μια δικτυωμένη οικονομία- δεν είναι απαραίτητο να εκτελεστεί τοπικά. Από την πλευρά της επιχείρησης τα οφέλη από τη διείσδυση της αγοράς σε αυτήν είναι σημαντικά - βελτιωμένη ικανότητα εστίασης, αυξημένη προσαρμοστικότητα και άμεση ανταγωνιστικότητα από άποψη κόστους.

Οι εταιρείες θα περιορίσουν την εμβέλεια του παραδοσιακά ενοποιημένου οργανισμού και θα συνενωθούν γύρω από τις βασικές, προσανατολισμένες στο προϊόν και στην εξυπηρέτηση του πελάτη, υπηρεσίες με απώτερο στόχο την ανάπτυξη. Οι παραδοσιακές ικανότητες λειτουργίας και υποστήριξης πρέπει συνεχώς να επαναπροσδιορίζουν τον ρόλο τους μέσα στα πλαίσια της επιχείρησης, ενώ συχνά θα περιορίζονται και θα αντικαθιστώνται από τις νέες συμβαλλόμενες σχέσεις και λειτουργίες.

3.3.2 Η ανάπτυξη και προώθηση προϊόντων και εφαρμογών απαιτεί νέους τομείς γνώσης

Για να υπάρξει επιτυχία, κάθε κλάδος διοίκησης, από τις πωλήσεις έως και το τμήμα διαχείρισης ανθρώπινου δυναμικού, θα πρέπει να εκτιμήσει τις επιδράσεις που θα έχει πάνω του ένα νέο μοντέλο προσανατολισμένο στις υπηρεσίες. Κατά κύριο λόγο θα πρέπει να εξετάσουν τι μπορεί να κάνουν προκειμένου να ενσωματώσουν και να "εμφυσήσουν" στον τομέα/τμήμα τους, τον τρόπο σκέψης των ηλεκτρονικών υπηρεσιών. Αυτό αποτελεί επιτακτική ανάγκη κυρίως για τα διευθυντικά στελέχη μιας εταιρείας, τα οποία είναι υπεύθυνα για την ανάπτυξη και προώθηση των προϊόντων και των υπηρεσιών αυτής.

Εντούτοις, οι διευθυντές τμημάτων ανάπτυξης και προώθησης προϊόντων, αντιμετωπίζουν ένα ιδιαίτερα πολύπλοκο έργο. Δεδομένου ότι μέχρι τώρα αυτοί είναι συχνά υπεύθυνοι για τα φυσικά προϊόντα και υπηρεσίες της εταιρείας τους, θα πρέπει να εξετάσουν πολύ σοβαρά τα χαρτοφυλάκια τους και να προσδιορίσουν εάν και με

ποιον τρόπο θα μπορέσουν να ενσωματώσουν στον τρόπο λειτουργίας της εταιρείας τους τα υπολογιστικά και επικοινωνιακά δεδομένα, τα οποία στοιχειοθετούν/συγκροτούν μια ηλεκτρονική υπηρεσία. Η διαχείριση τέτοιων "ολοκληρωμένων" προϊόντων θα απαιτήσει ένα εκτεταμένο σύνολο τομέων γνώσης σχετικά με την διαδικασία ανάπτυξης και προώθησης προϊόντων.

Παράλληλα με τις βασικές προσπάθειες ανάπτυξης θα πρέπει να αναπτυχθούν και να οργανωθούν νέες τεχνικές ικανότητες σχετικές με την προσανατολισμένη στις ηλεκτρονικές υπηρεσίες φύση των προϊόντων. Κατά τον σχεδιασμό των προϊόντων λοιπόν, οι μηχανικοί θα πρέπει να επικεντρώσουν την προσοχή τους σε χαρακτηριστικά όπως η προσαρμοστικότητα, δεδομένου οι ηλεκτρονικές υπηρεσίες τείνουν να δημιουργήσουν πολλές επαναλήψεις στη πορεία του κύκλου ζωής ενός προϊόντος. Παρόμοια, οι συνδεδεμένες ηλεκτρονικές υπηρεσίες θα χρειαστεί να συμβιβάσουν μια ευρείας κλίμακας βάση ετερογενών περιφερειακών συστημάτων H/W.

Κατά τον σχεδιασμό του κύκλου ζωής ενός προϊόντος οι σχεδιαστές θα πρέπει να δώσουν έμφαση στην "επίδραση" που έχει σε ένα προϊόν η χρησιμοποίηση του από μια ηλεκτρονική υπηρεσία, αν και εφόσον υπάρχει, προκειμένου οι αναβαθμίσεις των "ολοκληρωμένων" προϊόντων να μπορούν να επιτευχθούν χωρίς προβλήματα. Επιπλέον, εντελώς διαφορετικές ικανότητες/δεξιότητες θα απαιτηθούν για την ολοκλήρωση του επιχειρησιακού μοντέλου που στηρίζεται σε αυτό το νεοεισηγμένο ορισμό του προϊόντος. Η καταμέτρηση, η παρακολούθηση και η χρέωση είναι λειτουργίες (δυνατότητες) άμεσα συνυφασμένες με τον κλάδο των επικοινωνιών. Στο μοντέλο των ηλεκτρονικών υπηρεσιών, κάθε μία από αυτές τις διαδικασίες μπορεί να είναι ένα ολοκληρωμένο κομμάτι εξασφάλισης κέρδους από τη διάθεση ενός προϊόντος.

Η πρόκληση μεγαλώνει από δύο ακόμη παράγοντες:

1) αυτά τα προϊόντα και οι ηλεκτρονικές υπηρεσίες επόμενης γενιάς συχνά θα αποτελούνται από πολλές δυνατότητες οι οποίες θα παρέχονται από διάφορα μέρη και
2) από τη φύση τους, αυτά τα προϊόντα και οι υπηρεσίες μπορούν να συνεπάγονται λιγότερα λειτουργικά έξοδα για τους πελάτες, μικρότερο επενδεδυμένο κεφάλαιο και μεγαλύτερη δυνατότητα επίτευξης ανταγωνιστικής τιμής.

Για να παραμείνουν ανταγωνιστικοί, οι επαγγελματίες της ανάπτυξης και προώθησης προϊόντων θα πρέπει επίσης να επικεντρώσουν την προσοχή τους και σε άλλους τομείς, όπως η αναζήτηση και οικοδόμηση αποτελεσματικών συμμαχιών και η ενσωμάτωση στην προσφορά των προϊόντων της λογικής της πίστης του πελάτη.

Η αποτελεσματική ανάπτυξη και προώθηση των ηλεκτρονικών υπηρεσιών θα απαιτήσει τη συνεργασία πολλών ομάδων. Αυτό βέβαια που δεν είναι ακόμα ξεκάθαρο είναι η βέλτιστη τακτική για τη δημιουργία συνεργασιών. Βασικά ερωτήματα που τίθενται είναι αν θα πρέπει μια οντότητα να συγχωνευθεί με μια άλλη εταιρεία, ή να αναπτύξει μια δεσμευτική συνεργασία Υπό αυτή την έννοια, η αναζήτηση και η δόμηση αποτελεσματικών συνεργασιών αποτελεί μια κρίσιμη διαδικασία. Οι παροχείς υπηρεσιών οι οποίοι επιθυμούν να μείνουν βιώσιμοι και αποτελεσματικοί, θα πρέπει να έχουν επάρκεια ως προς ένα μεγάλο εύρος δραστηριοτήτων επιχειρηματικής ανάπτυξης, από το να εκτιμούν γρήγορα το ενδεχόμενο μιας συνεργασίας, μέχρι το να

επιτυγχάνουν την κατάλληλη συμφωνία και να διαχειρίζονται σε συνεχή βάση την σχέση που απορρέει από αυτήν.

Επιπλέον όλων αυτών, οι επιχειρήσεις αντιμετωπίζουν μια επιπλέον πρόκληση: σε έναν κόσμο όλο και πιο ολοκληρωμένης πληροφόρησης, ενεργών αγορών και μειωμένων λειτουργικών εξόδων, οι μηχανικοί σχεδίασης και προώθησης προϊόντων θα πρέπει να σκεφτούν σοβαρά για το πως θα ενσωματώσουν κατευθείαν στο σχεδιασμό και την προώθηση των προϊόντων/υπηρεσιών κίνητρα για την πίστη των πελατών. Αυτές οι προσεγγίσεις θα πρέπει να αποτελούν ένα αναπόσπαστο τμήμα της διαδικασίας σχεδιασμού και παράδοσης του προϊόντος/υπηρεσίας.

3.4 Μετασχηματισμός της IT λειτουργίας

Η εμφάνιση του προσανατολισμένου στις υπηρεσίες υπολογιστικού μοντέλου έχει σοβαρές επιδράσεις στη I/T λειτουργία μιας επιχείρησης. Κατά πρώτον, οι διευρυμένες ηλεκτρονικές υπηρεσίες αυξάνουν τον ρόλο της τεχνολογίας των πληροφοριών στη δημιουργία και υποστήριξη της αύξησης των εσόδων αυτή την περίοδο. Επιπλέον, η προσανατολισμένη στις υπηρεσίες πληροφορική στρέφει σε μια διαφορετική αντιμετώπιση όσον αφορά την τιμολόγηση και κοστολόγηση εσωτερικών I/T υπηρεσιών, οδηγώντας σε μια πιο δυναμική και προσανατολισμένη στην αγορά προσέγγιση διαδικασία λήψης αποφάσεων για θέματα λειτουργίας.

Το τελικό αποτέλεσμα θα είναι η αλλαγή σκηνικού η οποία θα αποτελέσει σοβαρή πρόκληση για μια I/T λειτουργία, αν όχι θα επιφέρει την οριστική ανατροπή της. Υπό το πρίσμα αυτής της προοπτικής, οι επιχειρήσεις θα πρέπει να πάρουν στρατηγικές αποφάσεις σχετικά με:

- το πεδίο ευθύνης τους
- την προσέγγιση των δομών προέλευσης και οργάνωσης
- τις τεχνολογικές αρχιτεκτονικές οι οποίες σχετίζονται και υποστηρίζονται από την I/T λειτουργία τους.

Προκειμένου να προχωρήσουν σε οποιαδήποτε από αυτές τις διαστάσεις, οι επιχειρήσεις θα πρέπει πρώτα να συνειδητοποιήσουν ότι μπορούν να διαχωρίσουν την I/T λειτουργία σε ένα πλήθος διαφορετικών τομέων, καθένας από τους οποίους υποστηρίζει μια συγκεκριμένη ιδιότητα. Για κάθε προσδιοριζόμενη ιδιότητα, τα διευθυντικά στελέχη θα πρέπει να δώσουν απάντηση σε μερικά βασικά ερωτήματα σχετικά με το:

- αν αποτελεί αυτή η ιδιότητα πηγή ανταγωνιστικής διαφοροποίησης
- ποιο είναι το καταλληλότερο υπολογιστικό μοντέλο για την ικανοποίηση αυτής της ιδιότητας
- ποιος μπορεί να την παρέχει πιο αποτελεσματικά

Οι επιχειρήσεις θα πρέπει να προσδιορίσουν τις I/T ιδιότητες οι οποίες παρέχουν μια ανταγωνιστική διαφοροποίηση. Πολλές τέτοιες ιδιότητες μπορεί να είναι σημαντικές (πχ. e-mail), αλλά όχι διαφοροποιητικές. Ένα πρώτο βήμα είναι η ανάπτυξη ενός συγκεκριμένου περιβάλλοντος όπου σε ένα υψηλό επίπεδο θα προσδιορίζονται τα κατάλληλα I/T αποκτήματα και θα οργανώνονται σε κατηγορίες οι I/T ιδιότητες της εταιρείας. Οι πραγματικά οριζόντιες ιδιότητες είναι τα βιομηχανικά πρότυπα τα οποία παρέχουν πολύ γνωστές και διαδεδομένες λειτουργίες (π.χ

messaging). Οι ημι-οριζόντιες ιδιότητες βασίζονται σε προκαθορισμένες επιχειρηματικές διαδικασίες, αλλά είναι ειδικά σχεδιασμένες για ένα συγκεκριμένο λειτουργικό περιβάλλον. Αυτές οι εφαρμογές περιλαμβάνουν κυρίως διαστάσεις ERP, Οικονομικές, Ανθρωπίνων Πόρων, και παραγωγής. Τέλος, υπάρχουν I/T εφαρμογές οι οποίες είναι προσαρμοσμένες σε μεγάλο βαθμό και υποστηρίζουν κατά κύριο λόγο τον συγκεκριμένο τρόπο με τον οποίο μια εταιρεία προσφέρει τις υπηρεσίες της.

Αν και κάθε τομέας θα έχει διαφορετικά θέματα μετάβασης, θα υιοθετήσουν το νέο πληροφοριακό μοντέλο με διαφορετικούς ρυθμούς. Τα οριζόντια συστήματα θα είναι και τα πρώτα που θα μετατραπούν με γνώμονα το προσανατολισμένο στις υπηρεσίες μοντέλο. Παρά την προκαθορισμένη φύση τους, αυτά τα στοιχεία και οι εφαρμογές απαιτούν σημαντικούς διοικητικούς πόρους για τη λειτουργία και τη συντήρηση τους. Μια δεύτερη κατηγορία, η οποία θα πρέπει να αποτελέσει αντικείμενο σκέψης, περιλαμβάνει τα υψηλού βαθμού προσαρμοσμένα συστήματα, τα οποία όμως προσφέρουν διαφοροποίηση σε πολύ μικρό βαθμό. Ειδικότερα, αυτά τα συστήματα απαιτούν υψηλό κόστος συντήρησης και αυξάνουν το επιχειρηματικό ρίσκο, χωρίς να προσφέρουν πολλά ανταλλάγματα.

Το επόμενο βήμα για τις εταιρίες είναι να προσδιορίσουν το πληροφοριακό μοντέλο που τους ταιριάζει καλύτερα προκειμένου να αναπτύξουν αυτές τις δυνατότητες. Μια από τις βασικές επιλογές αποτελεί το αν θα αναπτύξουν αυτές τις ικανότητες σαν μια παραδοσιακή I/T λειτουργία, ή αν θα έχουν μια άλλη προσέγγιση προσανατολισμένη στη χρησιμότητα τους. Σε ορισμένες περιπτώσεις, οι απαιτήσεις της εφαρμογής και η αρχιτεκτονική μπορεί να μεροληπτούν υπέρ της παραδοσιακής I/T ανάπτυξης. Για προσαρμοσμένα συστήματα, ή/και συστήματα υψηλής στρατηγικής μπορεί να μην υφίσταται καν η δυνατότητα ανάπτυξης υπό το πρίσμα του προσανατολισμένου στις υπηρεσίες μοντέλου, ενώ για άλλες εφαρμογές το μοντέλο αυτό μπορεί να είναι το πλέον κατάλληλο.

Ακόμα και για αυτές τις δυνατότητες πρέπει να επικρατήσει η στρατηγική της βαθμιαίας μετάβασης. Για παράδειγμα πολλές εταιρίες έχουν αναπτύξει σημαντικές ικανότητες/χαρακτηριστικά γύρω από μια βάση επικοινωνίας (π.χ. e-mails). Η βασική πλατφόρμα μηνυμάτων συνδέεται πολύπλοκα με άλλες εφαρμογές όπως η οργάνωση ομάδων, η δυναμική συζήτηση (αλληλεπίδρασης), το διάγραμμα ροής εργασιών, οι βάσεις και η διεξαγωγή των συναλλαγών. Συνεπώς η φάση μετάβασης θα πρέπει να είναι βαθμιαία και άκρως προσεκτική.

Καθώς οι εταιρίες σκέφτονται αυτά τα θέματα, θα πρέπει έχουν υπ' όψιν τους μια πολύ σημαντική ιδέα - η βασισμένη στις υπηρεσίες τεχνολογία της πληροφορικής μπορεί εύκολα να συνυπάρξει με τις τρέχουσες τεχνολογικές εφαρμογές. Σε μερικές περιπτώσεις, οι χρήστες ίσως να μην αναγνωρίσουν καν τη διαφορά. Σε άλλες, οι χρήστες εύκολα θα μεταβούν από τις παραδοσιακές εφαρμογές σε ένα άλλο κόσμο προσανατολισμένο στις υπηρεσίες. Έτσι, στις περισσότερες περιπτώσεις, η αφορμή για την αλλαγή αυτή, αλλά και το μονοπάτι της μετάβασης θα πρέπει να εξεταστούν από τις εταιρίες και να αναχθούν σε μακροχρόνιους στόχους.

Τελικά, η διοίκηση πρέπει να προσδιορίσει τους εσωτερικούς ή εξωτερικούς παροχείς οι οποίοι μπορούν να παραδώσουν τις καλύτερες υπηρεσίες. Πληροφορική προσανατολισμένη στις υπηρεσίες μπορεί να παραδοθεί και από ιδιόκτητες οντότητες. Παράγοντες κλειδιά που θα πρέπει να λάβουμε υπ' όψη μας σε αυτή την απόφαση είναι

το κόστος της υπηρεσίας, η ποιότητά της και τα προσφερόμενα ιδιαίτερα χαρακτηριστικά της. Όπως ακριβώς στις παραδοσιακές πρακτικές, είναι πιθανόν ότι για τις περισσότερες κλάσεις εφαρμογών ή τα I/T χαρακτηριστικά, εξωτερικοί παροχείς υπηρεσιών θα μειώσουν το κόστος και θα αυξήσουν τα πλεονεκτήματα της υπηρεσίας, μέσα από μια επιθετική προσέγγιση και οικονομίες κλίμακας. Άλλοι παράγοντες που θα πρέπει να λάβουμε υπ' όψη μας είναι το επίπεδο ελέγχου και η προσαρμοστικότητα στην αλλαγή. Η εκ των έσω διαχείριση θα παρέχει τυπικά μεγαλύτερα επίπεδα ελέγχου, γρηγορότερη απόκριση και προσαρμογή στις διαδικασίες τροποποίησης ενός συστήματος, αλλά επίσης μπορεί να συνεπάγεται υψηλότερο κόστος και δέσμευση της αυξανόμενης τεχνολογίας.

Το όραμα των συνεργαζόμενων ηλεκτρονικών υπηρεσιών βασίζεται σε ορισμένες αρχές. Ορισμένες από αυτές θα λάβουν χώρα στο επιχειρηματικό πεδίο, ενώ κάποιες άλλες στο τεχνολογικό.

- Οι ηλεκτρονικές υπηρεσίες θα είναι πλήρως μοντελοποιημένες και θα στηρίζονται σε ένα ανοικτό πρότυπο, έτσι ώστε να μπορούν να συνδυαστούν ευκολότερα προσφέροντας νέου τύπου υπηρεσίες.
- Δεν θα λαμβάνουν χώρα όλες οι συναλλαγές μέσω των WebSites. Οι περισσότερες θα πραγματοποιούνται δυναμικά μέσω συνεργαζόμενων υπηρεσιών που θα βρίσκονται στο παρασκήνιο.
- Το Web θα μετασχηματιστεί από το υπάρχον μοντέλο όπου ο χρήστης λαμβάνει μέριμνα για οτιδήποτε, σε ένα νέο όπου οι υπηρεσίες θα δρουν για λογαριασμό του.
- Το θέμα των υπηρεσιών αφορά κατά τι παραπάνω από τους προσωπικούς υπολογιστές. Συσκευές διαφόρων τύπων θα αποτελέσουν πλατφόρμες για την παράδοση υπηρεσιών όλων των τύπων.
- Το Internet θα χρησιμοποιηθεί για να καλύψει τις ανάγκες τόσο των ανθρώπων, όσο και των πραγμάτων. Εκατομμύρια συσκευές θα συνδεθούν στο δίκτυο, αλλάζοντας τον τρόπο με τον οποίο οι υπηρεσίες δομούνται, παραδίδονται και χρησιμοποιούνται.

ΚΕΦΑΛΑΙΟ 4ο

Η ΕΠΙΔΡΑΣΗ ΤΟΥ ΜΟΝΤΕΛΟΥ ΤΩΝ ΥΠΗΡΕΣΙΩΝ ΣΤΟ ΕΣΩΤΕΡΙΚΟ ΤΩΝ ΟΡΓΑΝΙΣΜΩΝ

4 Η ΕΠΙΔΡΑΣΗ ΤΟΥ ΜΟΝΤΕΛΟΥ ΤΩΝ ΥΠΗΡΕΣΙΩΝ ΣΤΟ ΕΣΩΤΕΡΙΚΟ ΤΩΝ ΟΡΓΑΝΙΣΜΩΝ

Η εφαρμογή του μοντέλου των υπηρεσιών είναι δυνατή και στο εσωτερικό των οργανισμών. Ο μετασχηματισμός του ισχύοντος μοντέλου προς αυτό των υπηρεσιών είναι κάτι που θα απασχολήσει σε μεγάλη έκταση τους διοικούντες έναν οργανισμό. Βέβαια, η απόφαση για το αν ορισμένες εσωτερικές διαδικασίες μπορούν να μετασχηματιστούν σε νέα βάση, αποτελεί παλιό δίλημμα του χώρου. Σε αυτό το κεφάλαιο θα αναζητήσουμε τις λεπτομέρειες μιας τέτοιας εφαρμογής, όπως επίσης τα θετικά και τα αρνητικά της υλοποίησης της.

Μιλώντας για παράδειγμα για τα τμήματα πληροφορικής των διαφόρων οργανισμών, συχνά αυτά αντιμετωπίζονται ως κέντρα παραγωγής κόστους παρά την σημαντική προσφορά τους. Η ίδια υπόθεση μπορεί να γίνει και για κάποιο άλλο τμήμα (π.χ. λογιστήριο). Ο ανταγωνισμός του συγκεκριμένου τομέα με αυτόν άλλων παροχέων υπηρεσιών είναι πολύ μικρός ως ανύπαρκτος και κυρίως για θέματα ασφαλείας. Η απομόνωση αυτή οδηγεί σε κλειστά διοικητικά τμήματα όπου παρατηρείται έντονα η έλλειψη καινοτομίας, η καθυστέρηση στην ανάδραση των ερεθισμάτων που υπάρχουν, αποφυγή του δημιουργικού ρίσκου και τέλος υψηλό κόστος ([1]).

Η εισαγωγή του μοντέλου των υπηρεσιών στο εσωτερικό των οργανισμών θα διαμορφώσει αναντίρρητα ατμόσφαιρα ανταγωνισμού. Διάφοροι έξυπνοι "Agents" επιτρέπουν στους καταναλωτές υπηρεσιών να ψάξουν στο Web και να επιτύχουν την καταλληλότερη συμφωνία που ικανοποιεί τις ανάγκες τους. Το μοντέλο των υπηρεσιών διευκολύνει ακόμη και τους εσωτερικούς καταναλωτές υπηρεσιών ενός οργανισμού να καταφύγουν σε εξωτερικούς παροχείς. Οι υπό διαμόρφωση συνθήκες σαφώς πριμοδοτούν ένα ανταγωνιστικό περιβάλλον στο εσωτερικό του οργανισμού, όμοιο με αυτό που οι οργανισμοί επιδιώκουν με τους εξωτερικούς τους πελάτες. Πιο συγκεκριμένα, ο ανταγωνισμός οδηγεί σε μείωση του κόστους, μεγαλύτερη ποικιλία υπηρεσιών, καθώς και ανάδειξη των καλύτερων υπηρεσιών που είναι σε θέση να δώσουν επιπλέον καινοτομία και ανταπόκριση.

4.1 Εσωτερικός Ανταγωνισμός

Η ιδέα βρίσκεται στην ύπαρξη πολλών εσωτερικών μονάδων που ανταγωνίζονται μεταξύ τους για την διαμόρφωση της καλύτερης υπηρεσίας. Σαφώς, για να υπάρχει εσωτερικός ανταγωνισμός απαραίτητη προϋπόθεση είναι η ύπαρξη μεγάλων οργανισμών που συμπεριλαμβάνουν τις επιμέρους μονάδες.

Η υιοθέτηση ενός τέτοιου οργανωτικού σχήματος ενδεχομένως δεν είναι άμεσα υλοποιήσιμη, κι αυτό γιατί σπαταλώνται πολλαπλάσιες δυνάμεις για την επίτευξη ενός στόχου. Παρόλα αυτά, το μοντέλο των υπηρεσιών δημιουργεί και αυτή την εναλλακτική αφήνοντας τις αποφάσεις στους έχοντες την ευθύνη διαμόρφωσης της εσωτερικής δομής, για το εύρος και το είδος των αλλαγών. Αν και ο εσωτερικός ανταγωνισμός θεωρείται μάλλον υποβαθμισμένος, μεγάλη σημασία έχει ο εξωτερικός ανταγωνισμός και η σύγκριση εσωτερικά παραγόμενων υπηρεσιών με αυτές των εξωτερικών παροχέων.

4.2 Εξωτερικός Ανταγωνισμός

Σαφώς ο ανταγωνισμός που θα υπάρξει μεταξύ εσωτερικών και εξωτερικών παροχέων είναι και ο ουσιαστικότερος που μπορεί να υπάρξει. Πολλές από τις εσωτερικές υπηρεσίες που μέχρι πρόσφατα θεωρούνταν πρωτεύουσας σημασίας όπως το e-mail, το www hosting, το δίκτυο, το remote access, η υποστήριξη, αρχίζουν και παρέχονται από εξωτερικούς παροχείς υπηρεσιών σε χαμηλότερες τιμές, καλύτερη ποιότητα, αλλά και ταχύτερη παράδοση. Όλα αυτά οφείλονται βέβαια στις οικονομίες κλίμακας που έχουν επιτύχει οι διάφοροι προμηθευτές εξαιτίας της εξειδίκευσης τους με κάποιο συγκεκριμένο πεδίο. Σε όλα αυτά μπορεί να αντιπαρατεθεί ο αντίλογος των εσωτερικών παροχέων που υποστηρίζουν ότι η από μέρους τους γνώση του εσωτερικού περιβάλλοντος, των αναγκών, των σχέσεων και η εξειδίκευση τους αποτελούν εξίσου σημαντικά πλεονεκτήματα. Συγκριτικά όμως το κόστος, η ταχύτητα και η ευελιξία είναι ικανά στοιχεία για να γείρουν το ζυγό της στάθμισης των θετικών και των αρνητικών στοιχείων που αντιπαρατίθενται.

4.3 Ο νέος **ρόλος των εσωτερικών IT τμημάτων**

Είναι βέβαιο ότι τα εσωτερικά IT τμήματα θα μπουν σε μια νέα εποχή, όπου η δυναμική διαμεσολάβηση θα είναι το κυρίαρχο στοιχείο. Οι βασικοί ρόλοι που καλύπτουν τα τμήματα αυτά, όπως η διαχείριση των servers του δικτύου και πολλών άλλων, θα απλοποιηθούν ακόμη περισσότερο τα προσεχή χρόνια, με αποτέλεσμα να αφήνεται ανοικτή η πιθανότητα κάλυψης αυτών των αναγκών από εξωτερικούς παροχείς, όπως ήδη αναφέρθηκε. Συνεπώς, το ερώτημα που προκύπτει είναι σχετικά με το ποιος θα είναι ο νέος ρόλος των IT τμημάτων στο εσωτερικό των οργανισμών. Αυτό που πιθανολογείται ότι θα συμβεί είναι ότι θα μετασχηματιστούν άμεσα σε ενδιάμεσους (brokers) ή σε εσωτερικούς συμβούλους των εφαρμογών, κατευθύνοντας τους οργανισμούς ανάλογα.

Στο μέλλον όπου η απόφαση για εσωτερική ανάπτυξη ή αγορά θα κλίνει υπέρ του δεύτερου, ο ρόλος του ενδιάμεσου-συμβούλου κρίνεται ιδιαίτερα σημαντικός για τις αποφάσεις που θα παίρνονται. ([20]) Καθώς το σενάριο που θέλει τους εσωτερικούς χρήστες να επιλέγουν άμεσα τις υπηρεσίες που κρίνουν απαραίτητες, κρίνεται ανεδαφικό, ο ρόλος της επιλογής και της αξιολόγησης θα περάσει στο προσωπικό των τμημάτων της πληροφορικής. Η ηγεσία είναι αυτή που θα χαρακτηρίζει πλέον, μιας και είναι απαραίτητη για την υπόδειξη των τρόπων ανάληψης και υλοποίησης των επιχειρηματικών με τις κατάλληλες τεχνολογίες και μέσα.

Πιο αναλυτικά ο ρόλος του διαμεσολαβητή περιλαμβάνει την κατανόηση των αναγκών που ανακύπτουν, τον συντονισμό των εξωτερικών παροχέων, την εποπτεία των προτεινόμενων τιμών και επιπέδου υπηρεσιών, την εξασφάλιση διαδικασιών ολοκλήρωσης και διαβεβαίωση ότι η υπηρεσία που εκμισθώνεται καλύπτει απόλυτα τις ανάγκες και ταιριάζει με τις εσωτερικές επιχειρηματικές διαδικασίες.([2]) Μια άλλη πολύ σημαντική αρμοδιότητα είναι αυτή της διαχείρισης των συμβολαίων που περιλαμβάνει περαιτέρω διαπραγματεύσεις επί των τιμών, εισαγωγή στην υπό διαμόρφωση συμφωνία και άλλων παραμέτρων, δημιουργία μιας στρατηγικής συνεργασίας κ.λ.π. Τέλος, κλείνοντας την παράγραφο αυτή, πρέπει να αναφερθούμε στην αυξανόμενη σημασία που θα δοθεί στο μέλλον στις οικονομικές παραμέτρους, τις οποίες πρέπει να κατέχουν οι ασχολούμενοι με το διαμεσολαβητικό ρόλο.

ΚΕΦΑΛΑΙΟ 5ο

ΑΝΑΛΥΤΙΚΟ ΜΟΝΤΕΛΟ ΤΩΝ ΣΥΝΕΡΓΑΖΟΜΕΝΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΗΡΕΣΙΩΝ

5 ΑΝΑΛΥΤΙΚΟ ΜΟΝΤΕΛΟ ΤΩΝ ΣΥΝΕΡΓΑΖΟΜΕΝΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΗΡΕΣΙΩΝ

Το όραμα για τις ηλεκτρονικές υπηρεσίες, μετατρέπει τα υπάρχοντα πρότυπα ηλεκτρονικού εμπορίου, αντικαθιστώντας τις υποθέσεις των B2C και B2B σχέσεων, με ένα πολλά προς πολλά B2B μοντέλο. Μετατρέπει το παράδειγμα της απομονωμένης βιτρίνας καταστήματος, δίνοντας της την δυνατότητα να αγοράζει από και να πουλάει σε άλλες ομότιμες ηλεκτρονικές υπηρεσίες. Μετατρέπει το παράδειγμα των "φιλοξενούμενων" εφαρμογών, επιτρέποντας σε μια ηλεκτρονική υπηρεσία να ανακαλύψει και να συνδεθεί δυναμικά με άλλες ηλεκτρονικές υπηρεσίες οι οποίες φιλοξενούνται σε άλλους παροχείς (providers). Μετατρέπει τα παραδείγματα out-sourcing, αυτοματοποιώντας τη διαδικασία σύνδεσης ετερογενών ηλεκτρονικών υπηρεσιών, καθιστώντας δυνατή την ηλεκτρονική διεξαγωγή ολόκληρης της διαδικασίας, ενδεχομένως χωρίς καμία ανθρώπινη παρέμβαση. Οι υπηρεσίες ηλεκτρονικού εμπορίου θα συνδέονται με τις πολυάριθμες ηλεκτρονικές υπηρεσίες, οι οποίες διευθύνουν τις άλλες επιχειρησιακές δραστηριότητες (π.χ. κέρδη επιχειρήσεων ή διαχείριση αλυσίδας προμηθειών). Οι συναλλαγές και η ροή εργασιών θα εξαπλωθούν σε όλες αυτές τις υπηρεσίες και ένα γεγονός σε μία υπηρεσία θα είναι ικανό να προκαλεί αντιδράσεις σε κάποια άλλη.

Αυτό το όραμα παρουσιάζει ένα νέο μοντέλο για τις ηλεκτρονικές επιχειρήσεις. Καταρχήν, προβλέπει μια αγορά η οποία υποστηρίζει διαδικασίες δημοπρασιών καθώς και χρηματιστηριακές, και που επιπλέον καθιστά δυνατή τη δυναμική δημιουργία και επισφράγιση συμφωνιών για τις ηλεκτρονικές υπηρεσίες. Κατά δεύτερον, προκειμένου οι ηλεκτρονικές υπηρεσίες να ολοκληρώνονται δυναμικά (χωρίς ανθρώπινη παρέμβαση), πρέπει να είναι δυνατή η οικοδόμηση δυναμικών σχέσεων μεταξύ των ηλεκτρονικών υπηρεσιών. Κατά τρίτον, οι ηλεκτρονικές υπηρεσίες θα πρέπει με κάποιο τρόπο να συμμετέχουν σε "αλληλεπιδράσεις χαλαρής σύνδεσης" (π.χ. σε έναν πλειστηριασμό)([9]). Δηλαδή, το όραμα για τις ηλεκτρονικές υπηρεσίες προβλέπει μια οριοθετημένη ηλεκτρονική αγορά, στην οποία οι συμβάσεις μπορούν αυτόματα να διαπραγματευτούν και οι ηλεκτρονικές υπηρεσίες μπορούν να οργανωθούν δυναμικά.

Ο στόχος του αρχιτεκτονικού σχεδίου για τις ηλεκτρονικές υπηρεσίες, είναι η ανάπτυξη των αρχιτεκτονικών προδιαγραφών μιας πλατφόρμας η οποία υποστηρίζει την γρήγορη και εύκολη ανάπτυξη βασικών αλλά και πιο σύνθετων ηλεκτρονικών υπηρεσιών, αξιοποιώντας παράλληλα τεχνολογίες υποδομής όπως το E-speak, το BizTalk, ή το Jini όπως θα δούμε στην συνέχεια. Στο παρόν κεφάλαιο προτείνεται ένα μοντέλο για την αγορά ηλεκτρονικών υπηρεσιών. Προσδιορίζουμε τα βασικά στοιχεία, τους ρόλους, τα αντικείμενα και συζητάμε τις ευθύνες και τους συνεργάτες για κάθε ρόλο και αντικείμενο.

5.1 Μοντέλο ηλεκτρονικών υπηρεσιών αγοράς

Το Μοντέλο αγοράς ηλεκτρονικών υπηρεσιών στοχεύει στο να προσδιορίσει τα βασικά στοιχεία μιας γενικότερης αγοράς ηλεκτρονικών υπηρεσιών. Αυτά τα στοιχεία αντιπροσωπεύουν ένα μοναδικό σύνολο συνεισφορών, οι οποίες όλες μαζί σχηματίζουν τα θεμέλια για τη δημιουργία και διεξαγωγή ηλεκτρονικών υπηρεσιών ([3]). Τα στοιχεία του μοντέλου αγοράς ορίζονται ως ακόλουθα:

- **Οι συμμετέχοντες στη Συναλλαγή**, οι οποίοι αντιπροσωπεύουν τις ομάδες οι οποίες θα πρωταγωνιστούν σε μία συναλλαγή. Μια συναλλαγή μπορεί να περιπλέκει δύο ή περισσότερες ομάδες.
- **Οι Μεσάζοντες**, οι οποίοι αντιπροσωπεύουν τις οντότητες οι οποίες φέρνουν σε επαφή τους Αγοραστές με τους Παροχείς Υπηρεσιών, έτσι ώστε να επιτευχθεί κάποια συμφωνία.
- **Οι Δημιουργοί Αγοράς**, οι οποίοι αντιπροσωπεύουν τις οντότητες που διασφαλίζουν ότι η συναλλαγή διεκπεραιώνεται όπως ακριβώς είχε προσυμφωνηθεί. Αυτό αποτελεί ένα βασικό στοιχείο του μοντέλου, δεδομένου ότι δημιουργεί ένα κλίμα εμπιστοσύνης, το οποίο απαιτείται για τη διεξαγωγή εμπορικών συναλλαγών.
- **Οι Οικονομικοί Διευθυντές**, οι οποίοι αντιπροσωπεύουν τις οντότητες που υποστηρίζουν τις οικονομικές πτυχές μιας ηλεκτρονικής συναλλαγής.
- **Οι Διευθυντές Κύκλου ζωής**, οι οποίοι αντιπροσωπεύουν τις οντότητες που διοικούν όλες τις φάσεις του κύκλου ζωής, από τη γένεση ως και τη χρήση μιας υπηρεσίας. Το έργο τους περιλαμβάνει τη διαχείριση λειτουργιών σύνθεσης, εφοδιασμού, ανάπτυξης, και πραγματοποίησης.

Κάθε ηλεκτρονική υπηρεσία δεν είναι απαραίτητο να εμπεριέχει όλα τα προαναφερθέντα στοιχεία του μοντέλου αγοράς. Μια ηλεκτρονική υπηρεσία αγοράς μπορεί να είναι τόσο απλή, όσο μια απλή συναλλαγή χαμηλού ρίσκου μεταξύ ενός ελάχιστου αριθμού συμμετεχόντων η οποία δεν απαιτεί καν κάποιο συμβόλαιο, ή τόσο πολύπλοκη, όσο ένα σύνολο συναλλαγών το οποίο απαιτεί ένα εκτεταμένο σύστημα υποστήριξης και στο οποίο εμπλέκονται πολλοί ανώνυμοι συμμετέχοντες ([13]). Σε κάθε περίπτωση, υπάρχει ένα γενικότερο μοντέλο για τη πραγματοποίηση συναλλαγών και είναι η φύση της συναλλαγής ο παράγοντας αυτός ο οποίος καθορίζει πόσα στοιχεία του μοντέλου απαιτούνται κάθε φορά. Ο ελάχιστος αριθμός συμμετεχόντων σε κάθε περίπτωση, είναι δύο.

Παρακάτω, δίνεται περισσότερη έμφαση σε κάθε ένα από αυτά τα βασικά στοιχεία, σε συνδυασμό με τους βαθύτερους ρόλους και τα αντικείμενα που τα υποστηρίζουν. Το παρακάτω διάγραμμα παρέχει μια περίληψη του μοντέλου αγοράς με όλους τους βαθύτερους ρόλους και τα αντικείμενα:

Ρόλος, είναι ένα λογικά συσχετιζόμενο σύνολο λειτουργιών, το οποίο συμβολίζει ένα δυναμικό σημείο αλληλεπίδρασης. Ο ορισμός κάποιου ρόλου είναι ανεξάρτητος από τη βαθύτερη εφαρμογή του ρόλου αυτού. Κατά κάποιο τρόπο, η έννοια του ρόλου μπορεί να θεωρηθεί ταυτόσημη με τις διεπαφές λογισμικού.

Αντικείμενο, είναι ένα λογικά συσχετιζόμενο σύνολο δεδομένων, με ενδεχομένως κάποια επιχειρησιακή λογική η οποία συνδέεται με αυτά τα δεδομένα. Αυτός είναι ο ορισμός ενός αντικειμένου υπό την οπτική γωνία του αντικειμενοστραφούς προγραμματισμού.

Συνιστώσα, είναι μια οντότητα λογισμικού η οποία παρέχει υλοποίηση ενός ή περισσότερων ρόλων. Εφαρμόζει ένα σημαντικό τμήμα επιχειρησιακής λογικής και αποτελεί τη βασική μονάδα διαμόρφωσης, επεκτασιμότητας, αντικατάστασης και κατανομής. Η εφαρμογή κάποιας συνιστώσας είναι εντελώς ανεξάρτητη από την εφαρμογή οποιασδήποτε άλλης.

Στο παρόν έγγραφο περιγράφονται ρόλοι και αντικείμενα. Ο τρόπος με τον οποίο οι ρόλοι συνδέονται με τις συνιστώσες προσδιορίζεται κατά τη σχεδίαση των πραγματικών συνιστωσών λογισμικού, ή ακόμα καλύτερα, σε μία χαμηλού επιπέδου αρχιτεκτονική συνιστωσών, η οποία καθοδηγεί το πώς αυτά τα βασικά στοιχεία οργανώνονται μέσα σε εκτελέσιμα συστήματα.

Μοντέλο ηλεκτρονικών υπηρεσιών αγοράς

Συμμετέχοντες στη συναλλαγή
- Αγοραστής
- Παροχέας Υπηρεσίας
- *Προδιαγραφές Αγοραστή*
- *Προδιαγραφές Παροχέα Υπηρεσιών*

Κανονική γραφή = Ρόλος
Πλάγια Γραφή = Αντικείμενο

Μεσάζοντες
- Διαφημιστής
- Διαμεσολαβητής
- Ανιχνευτής αντιπρόσωπος
- Δημοπράτης
- Μεταφραστής
- Διαπραγματευτής
- *Συμβόλαιο*
- *Προσφορά*
- *Καταχώρηση προσφοράς*
- *Αίτηση για προσφορά*
- *Αίτηση για καταχώρηση προσφοράς*

Διαμορφωτές Αγοράς
- Δημιουργοί Πολιτικής
- Σύμβουλος
- Ελεγκτής
- Ασφαλιστής
- Τριτεγγυητής
- Συμβολαιογράφος
- Ενδυναμωτής Συμφωνίας
- Καταχώρηση Συμβολαίου

Διαχειριστές Κύκλου Ζωής
- Διαχειριστής Συμφωνίας
- Διαχειριστής Υποστήριξης
- Διαχειριστής Εκτέλεσης

Οικονομικοί Διαχειριστές
- Διαχειριστές Προσφοράς
- Διαχειριστές Χρέωσης
- Διαχειριστές Πληρωμής
- Τριτεγγυητής

5.2 Πρωτεύοντες Ρόλοι και Αντικείμενα

5.2.1 Συμμετέχοντες στη συναλλαγή

ΡΟΛΟΙ	ΑΝΤΙΚΕΙΜΕΝΑ
· Παροχέας Υπηρεσιών	· Προδιαγραφές Παροχέα Υπηρεσιών
· Αγοραστής	· Προδιαγραφές Αγοραστή

Οι συμμετέχοντες στη συναλλαγή εκπροσωπούν τους συμμετέχοντες σε οποιαδήποτε συναλλαγή (είτε αναφέρονται σε κάποιο συμβόλαιο, είτε όχι). Στο μοντέλο αυτό, κάθε συναλλαγή θα έχει ένα τουλάχιστον αγοραστή και ένα τουλάχιστον παροχέα υπηρεσιών. Κάθε εμπλεκόμενος ο οποίος είτε αγοράζει, είτε παρέχει κάποια υπηρεσία (δωρεάν ή μη), θα πρέπει να συμμορφώνεται ως προς κάποιο ελάχιστο σύνολο ευθυνών, στα πλαίσια του μοντέλου Ηλεκτρονικών Υπηρεσιών αγοράς. Αυτά τα ελάχιστα σύνολα ευθυνών, προδιαγράφονται στους ρόλους αγοραστών και παροχέων υπηρεσιών.

Μια συνιστώσα λογισμικού μπορεί να υλοποιήσει πολλούς ρόλους. Κάθε ρόλος απλά προσδιορίζει ένα συγκεκριμένο σύνολο ευθυνών. Οι συμμετέχοντες στη διαδικασία ρόλοι είναι σημαντικοί γιατί προσδιορίζουν το ελάχιστο σύνολο απαιτήσεων για τη συμμετοχή στο μοντέλο αγοράς ηλεκτρονικών υπηρεσιών. Ακολουθεί ένα παράδειγμα εφαρμογής ηλεκτρονικών υπηρεσιών το οποίο προϋποθέτει πολλαπλούς ρόλους.

Μια ηλεκτρονική υπηρεσία υποθήκης, έχει τρεις συμμετέχοντες. Μια μεσιτική ηλεκτρονική υπηρεσία υποθηκών (Mortage Broker E-Service), μια ηλεκτρονική τραπεζική υπηρεσία και μια ηλεκτρονική πιστοληπτική υπηρεσία.

- Η μεσιτική ηλεκτρονική υπηρεσία υποθηκών αναλαμβάνει τον ρόλο του πελάτη στη σχέση της με την ηλεκτρονική τραπεζική υπηρεσία. Αναλαμβάνει επίσης τον ρόλο του μεσίτη.
- Η ηλεκτρονική τραπεζική υπηρεσία αναλαμβάνει τον ρόλο του παροχέα υπηρεσιών στη σχέση της με την ηλεκτρονική υπηρεσία υποθήκης, αλλά αναλαμβάνει τον ρόλο του αγοραστή στη σχέση της με την ηλεκτρονική πιστοληπτική υπηρεσία.

Από το παραπάνω παράδειγμα συνάγεται το συμπέρασμα ότι οι συμμετέχοντες σε μια συναλλαγή μπορεί να διαδραματίζουν πολλούς διαφορετικούς ρόλους, αλλά όλες οι σχέσεις καταλήγουν σε έναν αγοραστή και σε έναν παροχέα υπηρεσιών.

5.2.2 Παροχέας Υπηρεσιών

Ο παροχέας υπηρεσιών είναι η οντότητα η οποία απευθείας παρέχει/πουλά μια υπηρεσία. Έτσι, ο παροχέας υπηρεσιών έχει τη τελική ευθύνη για τη διασφάλιση του γεγονότος ότι η υπηρεσία παραδόθηκε όπως ακριβώς είχε προσυμφωνηθεί. Έχει απόλυτη εξουσία σε όλες τις φάσεις του κύκλου ζωής μιας ηλεκτρονικής υπηρεσίας, συμπεριλαμβανομένης της δημιουργίας της, της σύνθετης υπηρεσίας συντονισμού, της διαπραγμάτευσης συμφωνίας, της εκπλήρωσης των υπηρεσιών κ.τ.λ. Ο παροχέας

υπηρεσιών ενδεχομένως να μεταβιβάσει κάποιες από αυτές τις ευθύνες, αλλά θα είναι ο αποκλειστικός υπεύθυνος για οποιαδήποτε διατάραξη της υπηρεσίας.

5.2.2.1 Ευθύνες

- Δημιουργεί προσφορές
- Ίσως εξουσιοδοτήσει διαφημιστή να διαφημίσει τη προσφορά
- Ίσως εξουσιοδοτήσει αντιπρόσωπο για να εντοπίζει βιώσιμες αιτήσεις για προσφορά
- Κάνει εκτίμηση των αιτήσεων για προσφορά
- Διαπραγματεύεται τα συμβόλαια για τις υπηρεσίες
- Εμβαθύνει στα συμβόλαια παροχής υπηρεσιών
- Διαπραγματεύεται και εμβαθύνει στα συμβόλαια παροχής χρηματιστηριακών υπηρεσιών και υπηρεσιών δημοπράτησης
- Συμμετέχει σε συμφωνίες υπηρεσιών συντήρησης
- Παρέχει στους αγοραστές προδιαγραφές των υπηρεσιών
- Συμμετέχει στην έναρξη, την αλληλεπίδραση και τον τερματισμό της υπηρεσίας
- Ενδεχόμενα να παρέχει ειδοποιήσεις στους αγοραστές
- Δέχεται και ενεργεί επί των παρατηρήσεων των πελατών
- Κυρίως παρέχει αυτοματοποιημένες υπηρεσίες για προγράμματα

5.2.2.2 Συνεργάτες

- Αγοραστής
- Παροχέας Υπηρεσιών
- Διαφημιστικός Πράκτορας
- Συνεργάτης αναζητήσεων (Finder Agent)
- Διαπραγματευτής
- Χρηματιστής
- Μεταφραστής
- Δημοπράτης

5.2.3 Αγοραστής

Ο αγοραστής είναι η οντότητα η οποία απευθείας καταναλώνει/αγοράζει την ηλεκτρονική υπηρεσία. Ο αγοραστής μπορεί να συμβολίζει ανθρώπους ή προγράμματα. Ο αγοραστής χρησιμοποιεί αιτήσεις-για-προσφορές προκειμένου να προσδιορίσει την επιθυμητή ηλεκτρονική υπηρεσία.

5.2.3.1 Ευθύνες

- Δημιουργεί αιτήσεις για προσφορές
- Ίσως εξουσιοδοτήσει διαφημιστή να διαφημίσει τις αιτήσεις για προσφορές
- Ίσως εξουσιοδοτήσει αντιπρόσωπο για να εντοπίζει βιώσιμες προσφορές
- Κάνει εκτίμηση των προσφορών
- Διαπραγματεύεται τα συμβόλαια για τις υπηρεσίες
- Εμβαθύνει στα συμβόλαια παροχής υπηρεσιών
- Διαπραγματεύεται και εμβαθύνει στα συμβόλαια παροχής χρηματιστηριακών υπηρεσιών και υπηρεσιών δημοπράτησης
- Συμμετέχει σε συμφωνίες υπηρεσιών συντήρησης

- Διαπραγματεύεται την πρόσβαση στις υπηρεσίες
- Διαπραγματεύεται τη χρήση των υπηρεσιών
- Δέχεται τις προδιαγραφές των υπηρεσιών από τους παροχείς υπηρεσιών
- Αποδέχεται την ανακατεύθυνση προς τους παροχείς υπηρεσιών
- Συμμετέχει στην έναρξη, την αλληλεπίδραση και τον τερματισμό της υπηρεσίας
- Παρέχει ειδοποιήσεις στους παροχείς υπηρεσιών
- Δέχεται και ενεργεί επί των παρατηρήσεων των παροχέων υπηρεσιών
- Συμμετέχει εκ μέρους ανθρώπων ή προγραμμάτων

5.2.3.2 Συνεργάτες

- Παροχέας Υπηρεσιών
- Διαφημιστής
- Συνεργάτης αναζητήσεων (Finder Agent)
- Διαπραγματευτής
- Χρηματιστής
- Μεταφραστής
- Δημοπράτης

5.2.4 Προδιαγραφές παροχέα υπηρεσιών

Οι προδιαγραφές ενός παροχέα υπηρεσιών είναι μια ολοκληρωμένη περιγραφή των χαρακτηριστικών του παροχέα υπηρεσιών και της λειτουργικότητας την οποία είναι ικανός να προσφέρει. Συνήθως χρησιμοποιείται ένα προκαθορισμένο λεξιλόγιο και μορφοποίηση, έτσι ώστε οι βασικές πλευρές των προδιαγραφών να είναι παγκόσμια κατανοητές. Επιπρόσθετα, εξειδικευμένα λεξιλόγια μπορούν να χρησιμοποιηθούν για να ορίσουν συγκεκριμένα τοπολογικά χαρακτηριστικά.

5.2.4.1 Ευθύνες

- Προσδιορισμός Διεπαφής
- Προσδιορισμός Πρωτοκόλλου αρχικοποίησης
- Προσδιορισμός Πρωτοκόλλου αλληλεπίδρασης
- Προσδιορισμός Πρωτοκόλλου ολοκλήρωσης
- Περιγραφή χαρακτηριστικών διαμόρφωσης
- Περιγραφή στατικών υποθέσεων
- Αποδοχή παρατηρήσεων γενόμενες από τους αγοραστές
- Αποδοχή παρατηρήσεων γενόμενες από τους παροχείς υπηρεσιών
- Παρακολούθηση ευθυνών Πελατών
- Προσδιορισμός Λεξιλογίου
- Προσδιορισμός χαρακτηριστικών
- Προσδιορισμός σύνθεσης

5.2.5 Προδιαγραφές Αγοραστή

Οι προδιαγραφές ενός αγοραστή περιγράφουν τα χαρακτηριστικά του, συμπεριλαμβανομένης της λειτουργικότητάς του. Για παράδειγμα, ο αγοραστής θα ανακαλύψει τη λειτουργικότητα που θα του επιτρέψει να απαντήσει σε ειδοποιήσεις. Δεν περιγράφει την υπηρεσία την οποία ο αγοραστής ζητάει από τον παροχέα¨ αυτός ο

ορισμός καλύπτεται από την αίτηση για προσφορά. Η λειτουργικότητα την οποία ο αγοραστής επιθυμεί να εκθέσει στον παροχέα υπηρεσιών, μπορεί να προσδιορίζεται στο συμβόλαιο.

5.2.5.1 Ευθύνες

- Περιγράφει τα χαρακτηριστικά και τη λειτουργικότητα των αγοραστών
- Προσδιορίζει τις διεπαφές
- Περιγράφει στατικές υποθέσεις
- Ειδοποιήσεις παραγόμενες από τους αγοραστές
- Προσδιορισμός Πρωτοκόλλου

5.2.6 Ενδιάμεσοι

5.2.6.1 Ρόλοι

- Διαφημιστής
- Συνεργάτης αναζητήσεων (Finder Agent)
- Διαπραγματευτής
- Χρηματιστής
- Μεταφραστής
- Δημοπράτης

5.2.6.2 Αντικείμενα

- Συμβόλαιο
- Προσφορά
- Αίτηση για προσφορά
- Καταχώρηση προσφοράς
- Αίτηση για καταχώρηση προσφοράς
- Καταχώρηση συμβολαίου

Οι μεσάζοντες επικεντρώνονται στην επίτευξη συμφωνίας συνεργασίας μεταξύ των συμμετεχόντων. Ο ρόλος των μεσαζόντων τείνει να είναι σύντομος, δεδομένου ότι είναι συγκεκριμένος σε κάθε συναλλαγή. Οι ευθύνες των μεσαζόντων μπορεί να ποικίλουν, από το να βοηθήσουν τους συμμετέχοντες στη συναλλαγή να έρθουν σε επαφή, μέχρι και στις διαπραγματεύσεις μιας συμφωνίας.

5.2.7 Πράκτορας Αναζήτησης

Ο ρόλος ενός πράκτορα αναζήτησης επικεντρώνεται στην εύρεση επιλογών οι οποίες ικανοποιούν τις ανάγκες του πελάτη του. Ο πελάτης μπορεί να είναι είτε κάποιος Αγοραστής, είτε κάποιος Παροχέας Υπηρεσιών. Ο διαφημιστής θα διαφημίσει μια αίτηση για προσφορά (Request-for-Offer RFO) για λογαριασμό ενός αγοραστή, ενώ την ίδια στιγμή μπορεί θα διαφημίζει μια προσφορά για λογαριασμό ενός Παροχέα Υπηρεσιών.

Ο ρόλος ενός πράκτορα αναζήτησης είναι σύντομος γιατί ερευνά ειδικά, για λογαριασμό ενός συγκεκριμένου πελάτη (είτε είναι Παροχέας Υπηρεσιών, είτε Αγοραστής) για μια συγκεκριμένη συναλλαγή. (Μια συγκεκριμένη μορφοποίηση των παραμέτρων εκτέλεσης ενός ρόλου, μπορεί να χρησιμοποιηθεί σε πολλές συναλλαγές, αλλά κάθε ρόλος τελειώνει μετά από κάθε συγκεκριμένη συναλλαγή). Άπαξ και βρεθεί κάποια συμφέρουσα επιλογή και επιτευχθεί κάποια συμφωνία, τελειώνει και ο ρόλος του πράκτορα αναζήτησης.

Ο πράκτορας αναζήτησης διασυνδέεται με άλλους ρόλους και αντικείμενα προκειμένου να εντοπίσει τις δυνατές επιλογές. Μπορεί να αναζητήσει καταχωρημένες Προσφορές και Αιτήσεις -για-Προσφορές, ή να επικοινωνήσει απευθείας με μεσίτες, δημοπράτες, αγοραστές, παροχείς υπηρεσιών, ή άλλους πράκτορες αναζήτησης. Οι πράκτορες αναζήτησης μπορεί να διαφέρουν ως προς τη φιλοσοφία, την ικανότητα, ή την αποτελεσματικότητα τους.

Ο πράκτορας αναζήτησης εκτιμά τις επιλογές προκειμένου να προσδιορίσει αυτές που ικανοποιούν περισσότερο τις απαιτήσεις του πελάτη, και του παρουσιάζει μόνο αυτές. Ο πράκτορας αναζήτησης δεν πραγματοποιεί καμία διαπραγμάτευση για λογαριασμό του πελάτη του.

Από την στιγμή που έχουν προσδιοριστεί οι δυνατές επιλογές, ο Αγοραστής και ο Παροχέας Υπηρεσιών πραγματοποιούν τις διαπραγματεύσεις, βοηθούμενοι ενδεχομένως από κάποιον διαπραγματευτή.

5.2.7.1 Ευθύνες

- Τηρεί παραπομπές σε λίστες Αιτήσεων-για-Προσφορές και Προσφορές Ηλεκτρονικών Υπηρεσιών, Μεσίτες, Δημοπράτες, Αγοραστές, παροχείς υπηρεσιών, ή άλλους πράκτορες αναζήτησης.
- Ερευνά για Προσφορές ή Αιτήσεις για Προσφορές
- Επιλέγει Προσφορές ή Αιτήσεις για Προσφορές
- Παρουσιάζει τις επιλογές στον πελάτη
- Δέχεται ειδοποιήσεις για αλλαγή των απαιτήσεων από τους πελάτες

5.2.7.2 Συνεργάτες

- Παροχείς υπηρεσιών
- Αγοραστές
- Μεσίτες
- Δημοπράτες
- Μεταφραστής
- Καταγραφή Προσφοράς
- Καταγραφή Αίτησης για προσφορά

5.2.8 Διαφημιστής

Ο διαφημιστής δημιουργεί και δημοσιεύει διαφημίσεις για λογαριασμό του πελάτη του. Ο πελάτης μπορεί να είναι είτε κάποιος Αγοραστής, είτε κάποιος

Παροχέας Υπηρεσιών. Το επίκεντρο της διαφήμισης εξαρτάται από τον τύπο του πελάτη ο οποίος εκπροσωπείται. Ο διαφημιστής θα διαφημίσει μια αίτηση για προσφορά (RFO) για λογαριασμό ενός αγοραστή, ενώ θα διαφημίσει μια προσφορά για λογαριασμό ενός Παροχέα Υπηρεσιών.

Όπως και ένας πράκτορας αναζήτησης, έτσι και ο ρόλος του διαφημιστή είναι σύντομος. Δημιουργεί διαφημίσεις για λογαριασμό ενός συγκεκριμένου πελάτη (είτε είναι Παροχέας Υπηρεσιών, είτε Αγοραστής). Άπαξ και βρεθεί κάποια συμφέρουσα επιλογή και επιτευχθεί κάποια συμφωνία, τελειώνει και ο ρόλος του διαφημιστή.

Ο διαφημιστής διασυνδέεται με άλλους ρόλους και αντικείμενα προκειμένου να δημιουργήσει διαφήμιση για λογαριασμό του πελάτη του. Μπορεί να διαφημίσει σε Λίστες Προσφορών και Αιτήσεων -για-Προσφορές, ή να επικοινωνήσει απευθείας με μεσίτες, δημοπράτες, αγοραστές, παροχείς υπηρεσιών, ή πράκτορες αναζήτησης.

Είναι σημαντικό να σημειωθεί ότι ένας Αγοραστής ή Παροχέας Υπηρεσιών μπορεί να υλοποιήσει τον ρόλο ενός Αγοραστή ή/και Πράκτορα Αναζήτησης. Σε αυτή την περίπτωση, αναλαμβάνει και διεκπεραιώνει τις ευθύνες οι οποίες συνεπάγονται από τον ρόλο αυτό.

5.2.8.1 Ευθύνες

- Τηρεί παραπομπές σε λίστες Αιτήσεων-για-προσφορές και Προσφορές Ηλεκτρονικών Υπηρεσιών, Μεσίτες, Δημοπράτες, Αγοραστές, παροχείς υπηρεσιών, ή άλλους πράκτορες αναζήτησης.
- Διαφημίζει Προσφορές ή Αιτήσεις για Προσφορές
- Δέχεται ειδοποιήσεις για αλλαγή των απαιτήσεων από τους πελάτες

5.2.8.2 Συνεργάτες

- Παροχείς υπηρεσιών
- Αγοραστές
- Μεσίτες
- Δημοπράτες
- Πράκτορας Αναζήτησης
- Καταγραφή Προσφοράς
- Καταγραφή Αίτησης για προσφορά

5.2.9 Διαμεσολαβητής

Ο Διαμεσολαβητής διαχειρίζεται και παρουσιάζει σύνολα προσφορών. Σε αντίθεση με τον Διαφημιστή ή τον Πράκτορα αναζήτησης, δεν ενεργεί για λογαριασμό κανενός Αγοραστή ή Παροχέα Υπηρεσιών. Απλά προσφέρει μια μοναδική συλλογή προσφορών συγκεκριμένου τύπου και παρέχει μερικά πρωτόκολλα που βοηθούν τον Αγοραστή να επιλέξει τη βέλτιστη προσφορά.

Για παράδειγμα, ένας Διαμεσολαβητής μπορεί να εξειδικεύεται σε ασφάλειες αυτοκινήτου για άτομα τα οποία είναι επιρρεπείς σε δυστυχήματα. Οι Πράκτορες αναζήτησης οι οποίοι αναζητούν τέτοιου τύπου προσφορές μπορούν να

εκμεταλλευτούν την προεργασία προσφορών τέτοιου τύπου την οποία έχει κάνει ο Διαμεσολαβητής. Η βασική έννοια που περικλείεται είναι περίπου ισοδύναμη με τη γνωστή ιδέα της "Πύλης" (Portal) ηλεκτρονικών υπηρεσιών, στο ότι παρέχει μια τοποθεσία όπου οι Αγοραστές μπορούν να καταφύγουν για να βρουν ένα σύνολο σχετικών υπηρεσιών, συνήθως οργανωμένες και φιλτραρισμένες με έναν πολύ χρήσιμο τρόπο. Η μεγαλύτερη διαφορά είναι αυτή του πεδίου δράσης. Μια "πύλη" παρέχει στον συνδρομητή μια λίστα από καταστήματα, καθένα από τα οποία περιέχει πολλά αντικείμενα, ο "Διαμεσολαβητής" παρέχει μια λίστα ηλεκτρονικών υπηρεσιών σε έναν Αγοραστή (electronic byer).

Ο Διαμεσολαβητής μπορεί να χρεώσει αμοιβή εύρεσης και/ή αμοιβή παροχής. Η αμοιβή εύρεσης χρεώνεται στον Αγοραστή με βάση την επιτυχημένη επιλογή σε μια αίτηση για προσφορά (RFO). Η αμοιβή παροχής χρεώνεται σε έναν παροχέα υπηρεσιών για την καταχώρηση των προσφορών του στο μητρώο του Μεσίτη. Κατά περίπτωση μπορούν να υπάρξουν και άλλοι μηχανισμοί χρέωσης.

Ο Διαμεσολαβητής δεν συμμετέχει στην διαδικασία των διαπραγματεύσεων. Βοηθάει συγκεκριμένα στην επιλογή της βέλτιστης προσφοράς σε σχέση με μια αντίστοιχη αίτηση-για-προσφορά.

5.2.9.1 Ευθύνες

- Αποδέχεται ενός ορισμένου τύπου προσφορές
- Διατηρεί συλλογές από Προσφορές
- Αντιπαραθέτει Προσφορές με Αιτήσεις-για-Προσφορές
- Παρουσιάζει τις επιλογές (αντιπαραθέσεις) στον πελάτη
- Δέχεται ανακοινώσεις

5.2.9.2 Συνεργάτες

- Πράκτορας Αναζήτησης (Finder Agent)
- Διαφημιστές
- Μητρώο Προσφορών
- Μητρώο Αιτήσεων-για-Προσφορά

5.2.10 Πλειστηριαστής

Ο Πλειστηριαστής είναι ένας άλλος ρόλος ο οποίος βοηθά στη διαδικασία αναζήτησης. Χειρίζεται τις καταστάσεις στις οποίες υπάρχουν πολλοί Αγοραστές για μια συγκεκριμένη υψηλών προδιαγραφών υπηρεσία. Ο Πλειστηριαστής προσφέρει μια μοναδική συλλογή αιτήσεων για προσφορές συγκεκριμένου τύπου και παρέχει μερικά πρωτόκολλα που βοηθούν τον Παροχέα υπηρεσιών να επιλέξει τη βέλτιστη αίτηση για προσφορά.

Όπως και ο Διαμεσολαβητής, ο Πλειστηριαστής μπορεί να χρεώσει αμοιβή εύρεσης (για λογαριασμό του Παροχέα Υπηρεσιών αυτή τη φορά), αμοιβή Αγοραστών (για τους αγοραστές οι οποίοι εγγράφονται για μια συγκεκριμένη δημοπράτηση, ή άλλων ειδών χρεώσεις.

Όπως και ο Διαμεσολαβητής, ο Πλειστηριαστής δεν συμμετέχει στην διαδικασία των διαπραγματεύσεων. Βοηθάει συγκεκριμένα στην επιλογή της βέλτιστης προσφοράς σε σχέση με μια αντίστοιχη αίτηση-για-προσφορά.

5.2.10.1 Ευθύνες

- Αποδέχεται ενός ορισμένου τύπου αιτήσεις για προσφορές
- Διατηρεί συλλογές από Αιτήσεις-για-Προσφορές
- Αντιπαραθέτει Προσφορές με Αιτήσεις-για-Προσφορές
- Παρουσιάζει τις αντιπαραθέσεις στον πελάτη
- Δέχεται ανακοινώσεις (π.χ. αλλαγές κατάστασης Registry (μητρώου, γραμματείας))

5.2.10.2 Συνεργάτες

- Finder Agent
- Διαφημιστές
- Μητρώο Προσφορών
- Μητρώο Αιτήσεων-για-Προσφορά

5.2.11 Διαπραγματευτής

Ο Διαπραγματευτής φέρνει δύο πλευρές σε συμφωνία, η οποία ενσαρκώνεται σε ένα Συμβόλαιο, το οποίο ικανοποιεί και τις δύο πλευρές. Ενώ ο Αγοραστής και ο Παροχέας Υπηρεσιών διαπραγματεύονται για λογαριασμό τους ο καθένας, ο Διαπραγματευτής παίζει ουδέτερο ρόλο. Ο Διαπραγματευτής χρησιμοποιεί τους όρους και τις συνθήκες που προσφέρονται από έναν παροχέα (όπως εκφράζονται σε μία Προσφορά) και τους όρους και τις συνθήκες τις οποίες επιθυμεί ο αγοραστής (όπως εκφράζονται στην Αίτηση για προσφορά), για να δει αν μπορεί να επιτευχθεί μία συμφωνία. Ο Διαπραγματευτής μπορεί να ενεργεί ως ενδιάμεσος, παρέχοντας ευκολίες και διαπραγματεύσεις μεταξύ του Αγοραστή και του Παροχέα Υπηρεσιών.

Οι διαπραγματευτές μπορούν να ειδικεύονται σε διαπραγματεύσεις που επικεντρώνονται σε ένα συγκεκριμένο σύνολο χαρακτηριστικών και τα ανταλλάγματά τους. Για παράδειγμα, ένας διαπραγματευτής Τιμής-QOS μπορεί να συμβάλει στην επίτευξη συμφωνίας ενός Αγοραστή και ενός Παροχέα Υπηρεσιών με μία αποδεκτή αρμονία Τιμής και QOS. Ένα συμβόλαιο δύναται να είναι αποτέλεσμα διαπραγματεύσεων οι οποίες προήχθησαν από πολλούς εξειδικευμένους Διαπραγματευτές.

5.2.11.1 Ευθύνες

- Διαχειρίζεται τις φάσεις της Πληροφόρησης, Συμφωνίας και Διακανονισμού της διαπραγμάτευσης.

5.2.11.2 Συνεργάτες

- Παροχέας Υπηρεσιών
- Αγοραστής

- Διαπραγματευτές
- Προδιαγραφές Υπηρεσιών
- Προδιαγραφές Αγοραστών
- Προσφορές
- Μεταφραστής

5.2.12 Μεταφραστής

Ένας Μεταφραστής ενεργεί σαν ένας μεταφραστής μονάδων, λεξιλογίων, κτλ μεταξύ δύο οντοτήτων. Για παράδειγμα, ένας Μεταφραστής μπορεί να κληθεί για να κάνει την "μετάφραση" μεταξύ Ευρωπαϊκής και Αμερικανικής αρίθμησης εμβαδού.

5.2.12.1 Ευθύνες

- Μετάφραση χαρακτηριστικών.

5.2.13 Συμβόλαιο

Ένα συμβόλαιο είναι μία δεσμευτική συμφωνία μεταξύ ενός Αγοραστή και ενός Παροχέα Υπηρεσιών. Καθορίζει τους όρους και τις συνθήκες τις οποίες θα πρέπει κάθε πλευρά να ικανοποιεί. Δεν πρέπει να υπερβαίνει τα όρια της Προδιαγραφής Αγοραστή ή την ίδια την Προδιαγραφή. Η τελική έκδοση ενός Συμβολαίου μπορεί να κρατηθεί και από τους δύο Συμμετέχοντες στην Συναλλαγή, ή μπορεί να καταχωρηθεί σε μία πολύ γνωστή "θέση" (π.χ. σε ένα Οδηγό αφιερωμένο για Συμβόλαια).

5.2.13.1 Ευθύνες

- Διατηρεί προδιαγραφές υπηρεσιών
- Διατηρεί προδιαγραφές Αγοραστή
- Διατηρεί όρους και συνθήκες συμβολαίου
- Όροι (προτιμώμενοι, απαιτούμενοι, διαπραγματεύσιμοι και αποδεκτοί):
- Μέθοδος μέτρησης χρήσης υπηρεσιών
- Μέθοδος χρέωσης
- Μέθοδος πληρωμής
- Μηχανισμός κοστολόγησης
- Διάρκεια ή σύνολο χρήσης
- Εξαιρέσεις (προαιρετικό): συνθήκες υπό τις οποίες το συμβόλαιο δεν ισχύει
- Καθορίζει την πρόσβαση στις υπηρεσίες
- Καθορίζει την χρήση των υπηρεσιών
- Πολιτική διεξόδου από παραβίαση
- Εμπλεκόμενα μέρη
- Συνθήκες (προτιμώμενες, διαπραγματεύσιμες, και αποδεκτές):
- Πρωτόκολλο χρήσης
- Περιορισμοί στο πρωτόκολλο χρήσης
- Περιορισμοί στην συχνότητα χρήσης
- Χαρακτηριστικά σύνθεσης συμφωνημένα με συμβόλαιο για την Υπηρεσία και τον Αγοραστή

- Πολιτική διαφυγής
- Καταγραφή εμπλεκομένων μερών

5.2.14 Προσφορά

Μια Προσφορά είναι μία απεικόνιση ενός συνόλου όρων και συνθηκών που παρέχει ένας Παροχέας Υπηρεσιών. Οι Προσφορές είναι μονάδες επικοινωνίας που χρησιμοποιούνται κατά την διάρκεια διαπραγμάτευσης μεταξύ ενός αγοραστή και ενός πωλητή. Οι Προσφορές μπορούν να χρησιμοποιηθούν και σαν μονάδες διαφημίσεως. Δεν πρέπει να υπερβαίνει τα όρια της Προδιαγραφής.

5.2.14.1 Ευθύνες

- Παρέχει περιγραφή της υπηρεσίας
- Παρέχει διαφήμιση της υπηρεσίας
- Παρέχει συμβόλαιο προσφερόμενης υπηρεσίας
- Παρέχει αναφορά υπηρεσίας
- Υποδεικνύει λήξη προσφοράς

5.2.14.2 Συνεργάτες

- Παροχέας Υπηρεσιών-Αγοραστή

5.2.15 Αίτηση -για- Προσφορά

Ένας Αγοραστής χρησιμοποιεί μία Αίτηση -για- Προσφορά για να ανακοινώσει ότι ψάχνει για μία υπηρεσία. Η Αίτηση -για- Προσφορά περιέχει τις ακριβείς λεπτομέρειες για την απαιτούμενη υπηρεσία. Ένας Αντιπρόσωπος, ένας Χρηματιστής ή Δημοπράτης μπορεί να αντιπαραθέσουν Προσφορές Παροχέων Υπηρεσιών με Αιτήσεις -για- Προσφορές ενός Αγοραστή.

5.2.15.1 Ευθύνες

- Υποδεικνύει απαιτούμενα χαρακτηριστικά
- Υποδεικνύει λήξη της αίτησης

5.2.16 Μητρώο Προσφορών

Μία Προσφορά παρέχει πρόσβαση σε ένα αποθετήριο Προσφορών. Ένα Μητρώο Προσφορών μπορεί ή όχι να παρέχει διάρκεια.

Ευθύνες

- Δέχεται Προσφορές
- Διαχειρίζεται, ταξινομεί και φιλτράρει Προσφορές
- Απαντά σε ερωτηματολόγια Προσφορών
- Παρουσιάζει Προσφορές σε αιτούντες
- Μπορεί να παρέχει διάρκεια

5.2.16.1 Συνεργάτες

- Προσφορές

5.2.17 Μητρώο Αιτήσεων -για- Προσφορές

Ένα Μητρώο Αιτήσεων -για- Προσφορές παρέχει πρόσβαση σε ένα αποθετήριο Αιτήσεων -για- Προσφορές. Ένα Μητρώο Αιτήσεων -για- Προσφορές μπορεί ή όχι να παρέχει διάρκεια.

5.2.17.1 Ευθύνες

- Δέχεται Αιτήσεις - για - Προσφορές
- Διαχειρίζεται, ταξινομεί και φιλτράρει Αιτήσεις - για - Προσφορές
- Απαντά σε ερωτηματολόγια Αιτήσεων -για- Προσφορές
- Παρουσιάζει Αιτήσεις - για - Προσφορές σε αιτούντες
- Μπορεί να παρέχει διάρκεια

5.2.17.2 Συνεργάτες

- Αιτήσεις - για – Προσφορές

5.2.18 Μητρώο Συμβολαίων

Ένα Μητρώο Συμβολαίων παρέχει πρόσβαση σε ένα αποθετήριο Συμβολαίων. Ένα Μητρώο Συμβολαίων μπορεί να παρέχει διάρκεια.

5.2.18.1 Ευθύνες

- Δέχεται Αιτήσεις - για - Προσφορές
- Διαχειρίζεται, ταξινομεί και φιλτράρει Αιτήσεις - για - Προσφορές
- Απαντά σε ερωτηματολόγια Αιτήσεων -για- Προσφορές
- Παρουσιάζει Αιτήσεις - για - Προσφορές σε αιτούντες
- Μπορεί να παρέχει διάρκεια

5.2.18.2 Συνεργάτες

- Συμβόλαιο

5.3 Δημιουργοί Αγοράς (Market Makers)

Ρόλοι

- Δημιουργός Πολιτικής
- Σύμβουλος
- Ελεγκτής
- Ασφαλιστής

- Τριτεγγυητής
- Συμβολαιογράφος
- Υποστηρικτής Συμβολαίου
- Καταχωρητής Συμβολαίων

Το Βασικό Στοιχείο του Δημιουργού Αγοράς εδραιώνει την εμπιστοσύνη στην αγορά. Αυτό είναι σημαντικό, διότι χωρίς εμπιστοσύνη δεν θα υπάρξει καθόλου συμμετοχή. Καθένας από τους Ρόλους εντός του βασικού στοιχείου του Δημιουργού Αγοράς απευθύνεται σε μία συγκεκριμένη πλευρά της εμπιστοσύνης. Αυτό επιτρέπει στην αγορά να προσαρμόζεται στις καταστάσεις. Απλές ηλεκτρονικές υπηρεσίες μπορούν να μην χρησιμοποιήσουν Δημιουργούς Αγοράς ή μπορούν να επηρεάσουν τα οφέλη ενός απλού Συμβούλου. Από την άλλη, πιο σύνθετες συναλλαγές ηλεκτρονικών υπηρεσιών μπορούν να εδραιώσουν υψηλότερα επίπεδα εμπιστοσύνης θέτοντας σε ισχύ τα περισσότερα από τα αντικείμενα και τους ρόλους ενός Market Maker.

Ο ρόλος των Market Makers είναι πιο μακροχρόνιος από αυτόν των Μεσαζόντων, δεδομένου ότι εξυπηρετούν περισσότερες συναλλαγές. Οι Market Makers μπορούν να αναπτυχθούν υπερωριακά, έτσι ώστε να μπορούν να κάνουν πιο αποτελεσματικές συστάσεις.

5.3.1 Δημιουργοί Πολιτικής (Policy Makers)

Οι Δημιουργοί Πολιτικής προσδιορίζουν τις πολιτικές τις οποίες ακολουθεί μια κοινότητα. Αυτό μπορεί να περιλαμβάνει πρότυπα (standards) και πρακτικές εργασίας. Επιτρέπει στους Αγοραστές και στους Παροχείς Υπηρεσιών να αναπτύξουν δραστηριότητες/συναλλαγές με βάση ένα σύνολο πολιτικών, αυξάνοντας το βασικό επίπεδο προσδοκιών των προτύπων πολιτικής.

5.3.1.1 Ευθύνες

- Παροχή και τήρηση πολιτικών κοινότητας

5.3.1.2 Συνεργάτες

- Αγοραστής
- Παροχέας Υπηρεσιών

5.3.2 Σύμβουλος

Ο Σύμβουλος μπορεί να παρέχει πληροφορίες αξιολόγησης και συστάσεις για Αγοραστές και Παροχείς Υπηρεσιών. Το στοιχείο αυτό προσδίδει έναν χαρακτήρα ποιοτικής βελτίωσης στη διαδικασία αναζήτησης, αφού παρέχει στους Αγοραστές και στους Παροχείς Υπηρεσιών τη δυνατότητα να περιορίσουν τις επιλογές οι οποίες στατιστικά είχαν ιστορικό κακής απόδοσης. Οι Σύμβουλοι συγκεντρώνουν τις πληροφορίες αυτές από διάφορες πηγές συμπεριλαμβανομένων Αγοραστών, Πωλητών και Ελεγκτών.

5.3.2.1 Ευθύνες

- Παρέχουν και τηρούν πληροφορίες απόδοσης για Αγοραστές και Παροχείς Υπηρεσιών

5.3.2.2 Συνεργάτες

- Αγοραστής
- Παροχέας Υπηρεσιών
- Ελεγκτές

5.3.3 Ελεγκτές

Οι ελεγκτές συγκεντρώνουν στοιχεία από την απόδοση Αγοραστών και Πωλητών. Οι ελεγκτές δεν ενεργούν σε σχέση με αυτά τα δεδομένα, αλλά απλά συγκεντρώνουν τα στοιχεία απόδοσης και τα παρουσιάζουν με έναν αποτελεσματικό τρόπο. Οι Υποστηρικτές συμβολαίων συγκρίνουν αυτά τα δεδομένα με το συμβόλαιο για να προσδιορίσουν αν τηρούνται οι όροι του. Ο διευθυντής υποστήριξης μπορεί να αξιοποιήσει αυτή τη πληροφορία προκειμένου να προσδιορίσει ή να προβλέψει θέματα απόδοσης τα οποία χρίζουν επαναρρύθμισης.

5.3.3.1 Ευθύνες

- Καταγράφουν την απόδοση Αγοραστών και Πωλητών
- Διαχειρίζονται και παρουσιάζουν τα δεδομένα απόδοσης με έναν αποτελεσματικό τρόπο.

5.3.3.2 Συνεργάτες

- Αγοραστής
- Παροχέας Υπηρεσιών

5.3.4 Ασφαλιστής

Ο ασφαλιστής παρέχει εγγυήσεις για το αποτέλεσμα, παρέχοντας εχέγγυα αν οι όροι ενός συμβολαίου δεν εκπληρωθούν. Αυτά τα εχέγγυα μπορεί να περιλαμβάνουν αποζημίωση ή άλλους τρόπους αποπληρωμής. Το κόστος για τέτοια κάλυψη μπορεί να ποικίλει.

5.3.4.1 Ευθύνες

- Παρέχουν εγγυήσεις για το αποτέλεσμα

5.3.4.2 Συνεργάτες

- Αγοραστής
- Παροχέας Υπηρεσιών
- Συμβόλαιο

5.3.5 Τριτεγγυητής

Ο Τριτεγγυητής παρέχει μια ουδέτερη τοποθεσία όπου φυλάσσονται ποσά μέχρι να παραδοθούν οι υπηρεσίες. Το γεγονός αυτό προσθέτει ένα στοιχείο εγγύησης για το ότι η υπηρεσία θα εξοφληθεί, λόγω του ότι τα ποσά έχουν δεδμευτεί στον τριτεγγυητή.

5.3.5.1 Ευθύνες

- Διαχειρίζεται τα εχέγγυα ποσά

5.3.5.2 Συνεργάτες

- Αγοραστής
- Παροχέας Υπηρεσιών

5.3.6 Συμβολαιογράφος

Ο Συμβολαιογράφος πιστοποιεί τις υπογραφές των συμβολαίων, γεγονός που δεσμεύει τη συμφωνία και για τις δύο πλευρές.

5.3.6.1 Ευθύνες

- Πιστοποιεί τις υπογραφές ενός συμβολαίου

5.3.6.2 Συνεργάτες

- Αγοραστής
- Παροχέας Υπηρεσιών
- Συμβόλαιο

5.3.7 Υποστηρικτής Συμβολαίου

Ο Υποστηρικτής Συμβολαίου συγκεντρώνει από τους Ελεγκτές τα στοιχεία απόδοσης και τα συγκρίνει με τους όρους του συμβολαίου για να διαπιστώσει αν αυτοί τηρούνται. Αν το συμβόλαιο "σπάσει" προειδοποιεί τους ενδιαφερόμενους. Έγκειται τότε στον Αγοραστή ή στον Παροχέα Υπηρεσιών να κάνει προσφυγή για τις παραβιάσεις του συμβολαίου. Η προσφυγή μπορεί να περιλαμβάνει επαναδιαπραγμάτευση των όρων του συμβολαίου, αποζημίωση, ή είσπραξη ασφάλειας.

5.3.7.1 Ευθύνες

- Συγκεντρώνει από τους Ελεγκτές τα στοιχεία απόδοσης και τα συγκρίνει με τους όρους του συμβολαίου και ειδοποιεί τους ενδιαφερόμενους αν αυτοί δεν τηρούνται.

5.3.7.2 Συνεργάτες

- Αγοραστές
- Παροχείς Υπηρεσιών
- Συμβόλαιο
- Ελεγκτές

5.3.8 Μητρώο/Καταγραφή Συμβολαίου (Contract registry)

Δες παρ. 3.2.12 για ορισμό

5.4 Διαχειριστές Κύκλου Ζωής (Lifecycle Managers)

Ρόλοι

- Διαχειριστής συγκρότησης
- Διαχειριστής συντήρησης
- Διαχειριστής ολοκλήρωσης

Οι Διαχειριστές κύκλου ζωής έργου επικεντρώνονται στη διαχείριση της γενικότερης δημιουργίας και συντήρησης μιας ηλεκτρονικής υπηρεσίας. Αυτό περιλαμβάνει τη διαχείριση ευθυνών σύνθεσης, εφοδιασμού, ανάπτυξης και λειτουργικότητας, οι οποίες απαιτούνται για τη διασφάλιση του ότι η ηλεκτρονική υπηρεσία θα πραγματοποιηθεί όπως αναμενόταν.

5.4.1 Διαχειριστής Συγκρότησης

Ο διαχειριστής συγκρότησης επικεντρώνεται σε θέματα σύνθεσης, ενίσχυσης και ανάπτυξης μιας ηλεκτρονικής υπηρεσίας. Καταγράφει τη ροή εργασιών, τη διαχείριση των γεγονότων και των συναλλαγών μεταξύ ετερογενών ηλεκτρονικών υπηρεσιών για να διασφαλίσει ότι λειτουργεί ακέραια. Πρωταρχικός του στόχος είναι να διαχειρίζεται την ηλεκτρονική υπηρεσία με τέτοιο τρόπο έτσι ώστε τα θέματα σύνθεσης να μην είναι ορατά στον Αγοραστή.

5.4.1.1 Ευθύνες

- Γνωρίζει τις απαραίτητες/απαιτούμενες υπηρεσίες
- Διαχειρίζεται τη συγκρότηση/αποσυγκρότηση
- Τροφοδοτεί την ηλεκτρονική υπηρεσία
- Διαχειρίζεται τον κύκλο εργασιών, τα γεγονότα και τις συναλλαγές για τις υπηρεσίες
- Χειρίζεται θέματα απευθείας αποζημίωσης και εξαιρέσεων

5.4.1.2 Συνεργάτες

- Παροχείς Υπηρεσιών
- Συμβόλαια

5.4.2 Διαχειριστής Υποστήριξης

Ο Διαχειριστής Υποστήριξης διασφαλίζει ότι η ηλεκτρονική υπηρεσία λειτουργεί κανονικά. Χρησιμοποιεί δεδομένα από τον Ελεγκτή για να προσδιορίσει ή να προβλέψει θέματα απόδοσης. Κατόπιν χρησιμοποιεί τα δεδομένα αυτά για να εγείρει προειδοποιήσεις στα πλαίσια της συγκροτούμενης υπηρεσίας, ή να επανορθώσει την κατάσταση.

5.4.2.1 Ευθύνες

- Αναλύει δεδομένα συγκεντρωμένα από τον Ελεγκτή για να προσδιορίσει ή να προβλέψει θέματα απόδοσης
- Εγείρει προειδοποιήσεις αν προκύψουν θέματα απόδοσης
- Διορθώνει καταστάσεις απόδοσης

5.4.2.2 Συνεργάτες

- Παροχείς Υπηρεσιών
- Συμβόλαια
- Ελεγκτής

5.4.3 Διαχειριστής Ολοκλήρωσης

Ο Διαχειριστής ολοκλήρωσης διαβεβαιώνει ότι η υπηρεσία ολοκληρώθηκε και παραδόθηκε όπως είχε προσυμφωνηθεί.

5.4.3.1 Ευθύνες

- Πιστοποιεί την παράδοση της υπηρεσίας

5.4.3.2 Συνεργάτες

- Ελεγκτής
- Συμβόλαιο

5.5 Οικονομικοί Διευθυντές

Ρόλοι

- Διευθυντής Προσφορών
- Διευθυντής Χρέωσης
- Διευθυντής Πληρωμής
- Τριτεγγυητής

Το βασικό στοιχείο του Οικονομικού Διευθυντή παρέχει τους υποστηρικτικούς ρόλους οι οποίοι σχετίζονται με τις οικονομικές πλευρές μιας ηλεκτρονικής υπηρεσίας. Αυτό περιλαμβάνει τις διαδικασίες της χρέωσης, της πληρωμής και της χρηματοδότησης.

5.5.1 Διαχειριστής Προσφοράς (BID Manager)

Ο διαχειριστής προσφοράς κάνει μια εκτίμηση του κόστους για τη συγκρότηση της ηλεκτρονικής υπηρεσίας. Δεν είναι δυνατόν να δοθεί ένα εγγυημένο ποσό, δεδομένου ότι το κόστος συχνά εξαρτάται από τη χρήση. Σε απάντηση του αιτήματος του παροχέα υπηρεσιών για να έχει μια εκτίμηση κόστους, ο διαχειριστής προσφοράς

ετοιμάζει μια προσφορά για την ηλεκτρονική υπηρεσία. Αυτή η προσφορά γίνεται χρησιμοποιώντας πρωτόκολλο που εκτιμά το κόστος από τις διάφορες υπό συγκρότηση υπηρεσίες.

5.5.1.1 Ευθύνες

- Αναπτύσσει γενική προσφορά της συγκροτούμενης υπηρεσίας
- Παρουσιάζει τη προσφορά στον Αγοραστή

5.5.1.2 Συνεργάτες

- Παροχέας Υπηρεσίας
- Διαχειριστής Προσφοράς
- Συμβόλαιο

5.5.2 Διαχειριστής Χρέωσης

Ο διαχειριστής χρέωσης χειρίζεται όλα τις διαδικασίες χρέωσης για μια ηλεκτρονική υπηρεσία. Συγκεντρώνει, οργανώνει και επεξεργάζεται τα δεδομένα χρέωσης από όλες τις συγκροτούμενες υπηρεσίες σε έναν μόνο λογαριασμό, τον οποίο παρουσιάζει στον Αγοραστή.

5.5.2.1 Ευθύνες

- Συγκεντρώνει τα δεδομένα χρέωσης από όλες τις συγκροτούμενες υπηρεσίες
- Οργανώνει και επεξεργάζεται τα δεδομένα χρέωσης σε έναν μόνο λογαριασμό
- Παρουσιάζει τον λογαριασμό στον Αγοραστή

5.5.2.2 Συνεργάτες

- Ελεγκτής
- Συμβόλαιο

5.5.3 Διαχειριστής Πληρωμής

Ο διαχειριστής πληρωμής χειρίζεται την ασφαλή εξόφληση του αγοραστή και παροχέα υπηρεσιών.

5.5.3.1 Ευθύνες

- Χειρίζεται τις πληρωμές μεταξύ Αγοραστή και Παροχέα Υπηρεσιών.

5.5.3.2 Συνεργάτες

- Διαχειριστής Χρέωσης
- Συμβόλαιο
- Υποστηρικτής συμβολαίου

5.5.4 Τριτεγγυητής (Escrow Manager)

Δες παρ 3.3.5

5.6 Διαγράμματα Χρήσης

Τα παρακάτω διαγράμματα αναπαριστούν τις πέντε πρώτες φάσεις του κύκλου ζωής μιας ηλεκτρονικής υπηρεσίας:

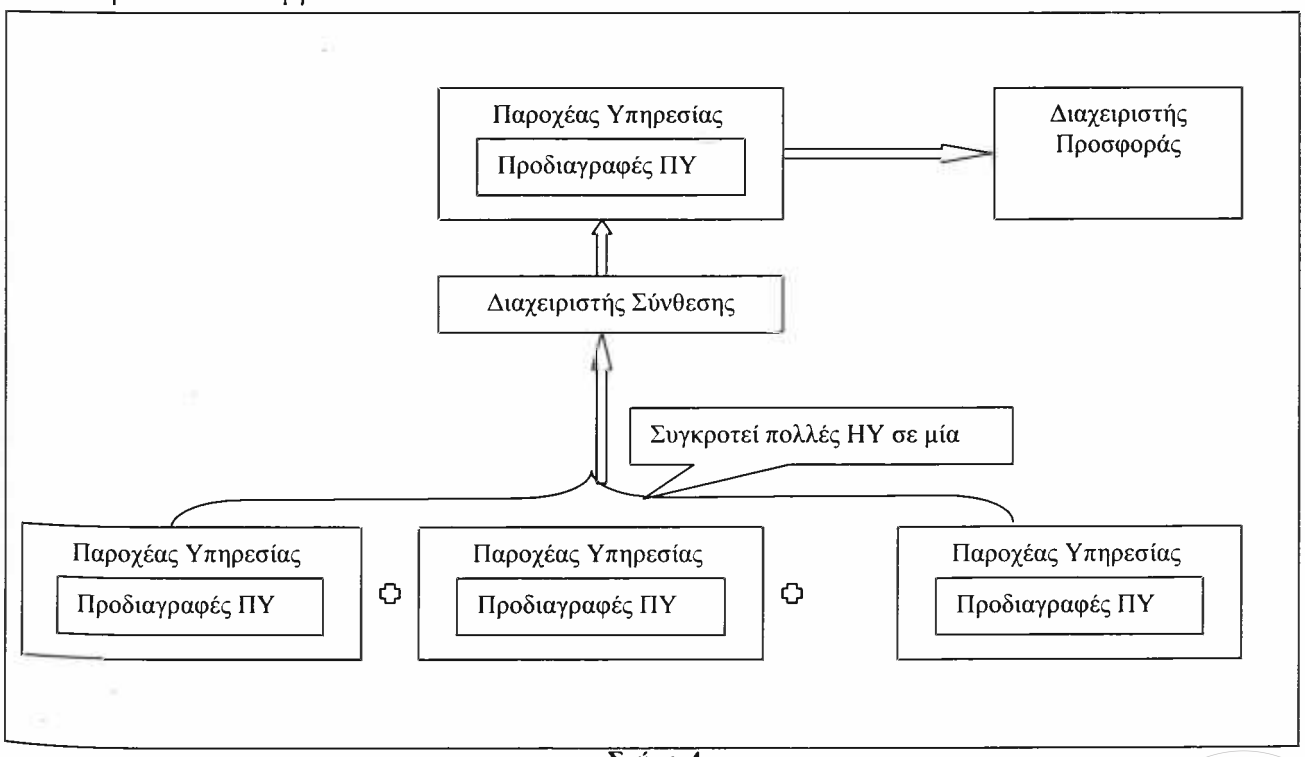
- Φάση Δημιουργίας
- Φάση Ανακάλυψης
- Φάση Διαπραγμάτευσης και Συμφωνίας
- Φάση Καταγραφής και Διαχείρισης
- Φάση ολοκλήρωσης και Διευθέτησης

Αυτά τα διαγράμματα είναι σχεδιασμένα να παραστήσουν μια υψηλού επιπέδου απεικόνιση της σχέσης μεταξύ των ορισμένων ρόλων και αντικειμένων.

5.6.1.1 Φάση Δημιουργίας

Η φάση δημιουργίας επικεντρώνεται κυρίως στον Παροχέα υπηρεσιών. Αυτή είναι η φάση κατά την οποία οι ηλεκτρονικές υπηρεσίες συγκροτούνται, εφοδιάζονται και αναπτύσσονται. Η διαδικασία της συγκρότησης προϋποθέτει ότι οι διαπραγματεύσεις και τα συμβόλαια έχουν ήδη προσδιοριστεί για όλους τους συμμετέχοντες παροχείς υπηρεσιών. Η διαδικασία της συγκρότησης επικεντρώνεται στο γενικότερο διάγραμμα ροής εργασιών, τη διαχείριση γεγονότων και συναλλαγών για τις συγκροτούμενες ηλεκτρονικές υπηρεσίες.

Σημείωση: Η φάση της δημιουργίας δεν απορρέει απαραίτητα από τη φάση των διαπραγματεύσεων και της συμφωνίας. Για δυναμικές ηλεκτρονικές υπηρεσίες, η φάση της δημιουργίας μπορεί να ακολουθήσει αμέσως μετά αφού οι απαιτήσεις έχουν οριστεί στο συμβόλαιο.

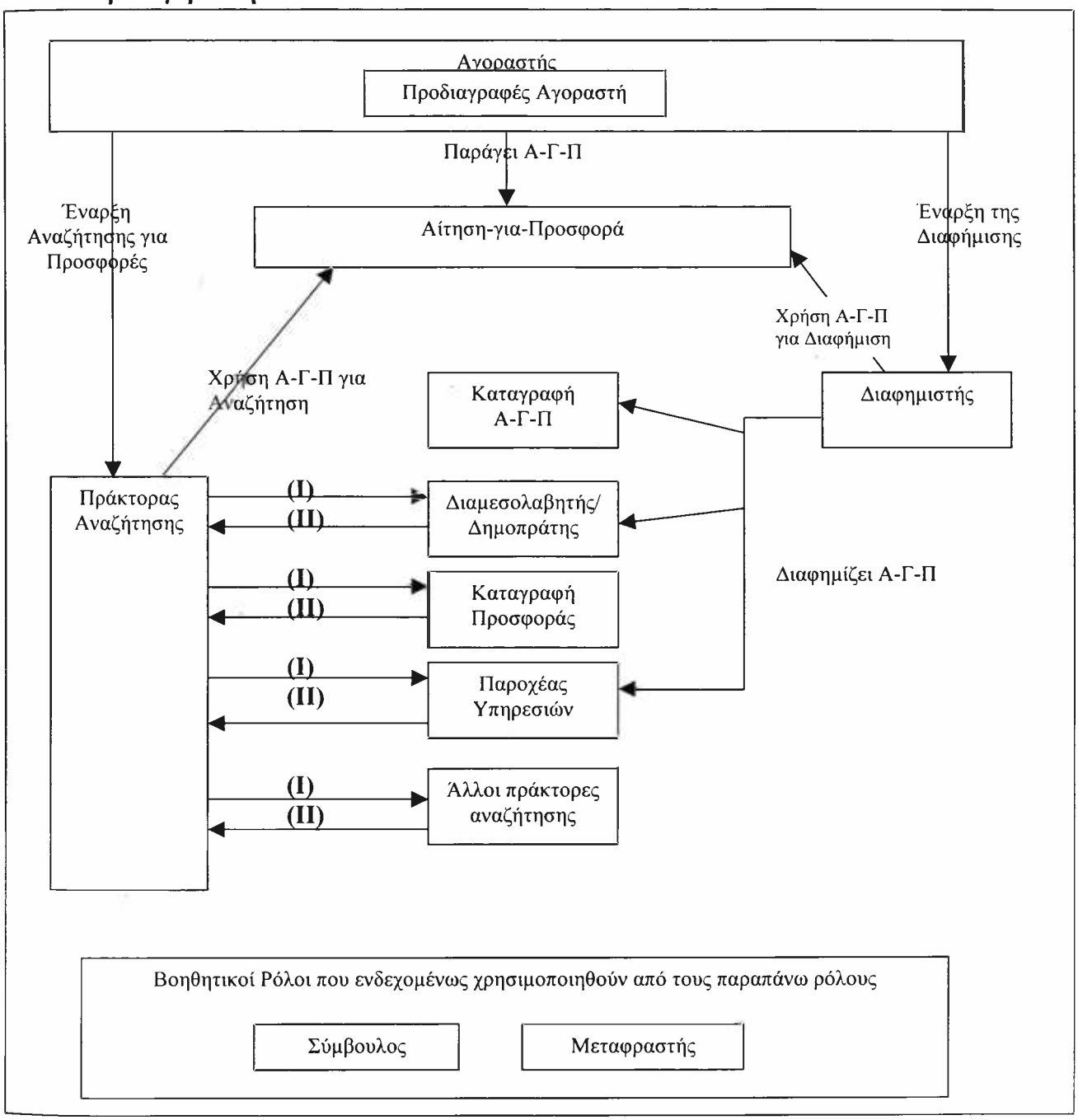


Σχήμα 4

5.6.2 Φάση Ανακάλυψης

Ο στόχος της φάσης ανακάλυψης είναι να εντοπίσει και να εδραιώσει τον καλύτερο δυνατό συνδυασμό μεταξύ μιας αίτησης για προσφορά ηλεκτρονικής υπηρεσίας ενός πελάτη και την τελική προσφορά ενός παροχέα υπηρεσιών. Τα παρακάτω διαγράμματα διαχωρίζονται σε φάση ανακάλυψης/αναζήτησης για τον Αγοραστή και τον Παροχέα Υπηρεσιών.

Πλευρά Αγοραστή

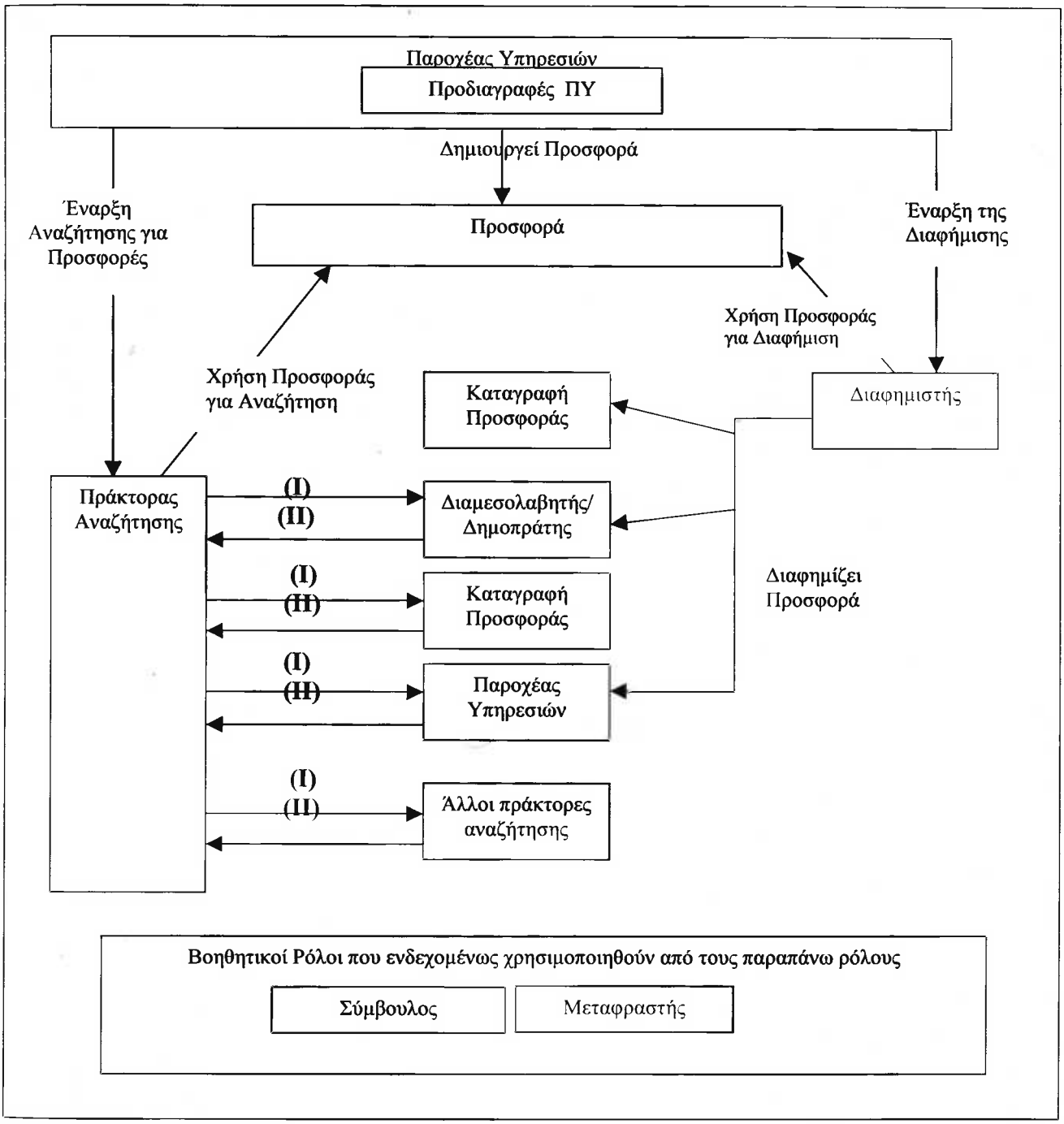


Σχήμα 5

Α-Γ-Π: Αίτηση για Προσφορά

(I) : Αναζήτηση Ταύτισης για Αιτούμενη Προσφορά
(II) : Απάντηση Ταύτισης για Αιτούμενη Προσφορά

Πλευρά του Παροχέα Υπηρεσιών

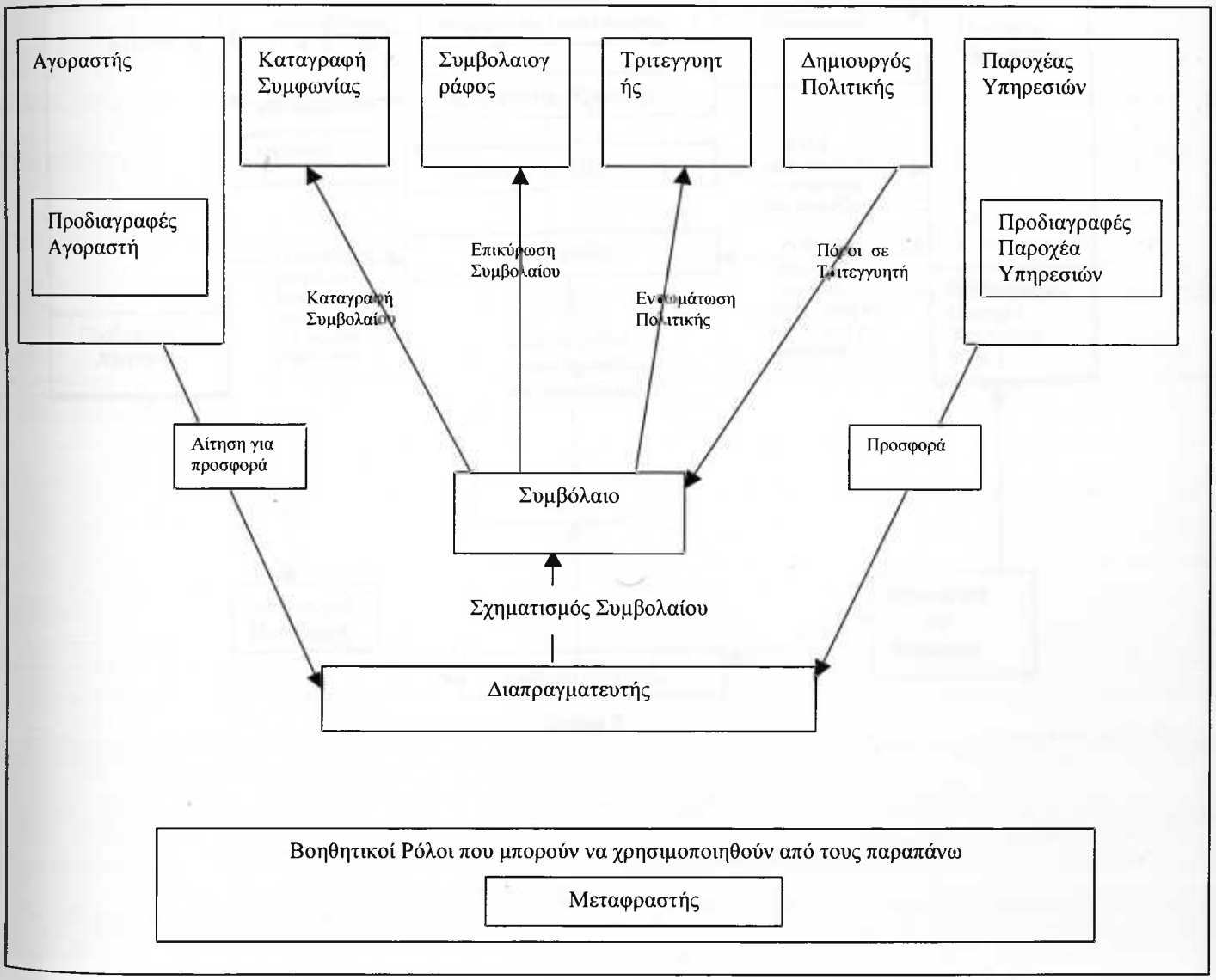


Σχήμα 6

(I): Αναζήτηση Ταύτισης για Προσφορές
(II): Απάντηση Ταύτισης για Προσφορές

5.6.3 Φάση Διαπραγμάτευσης και Συμφωνίας

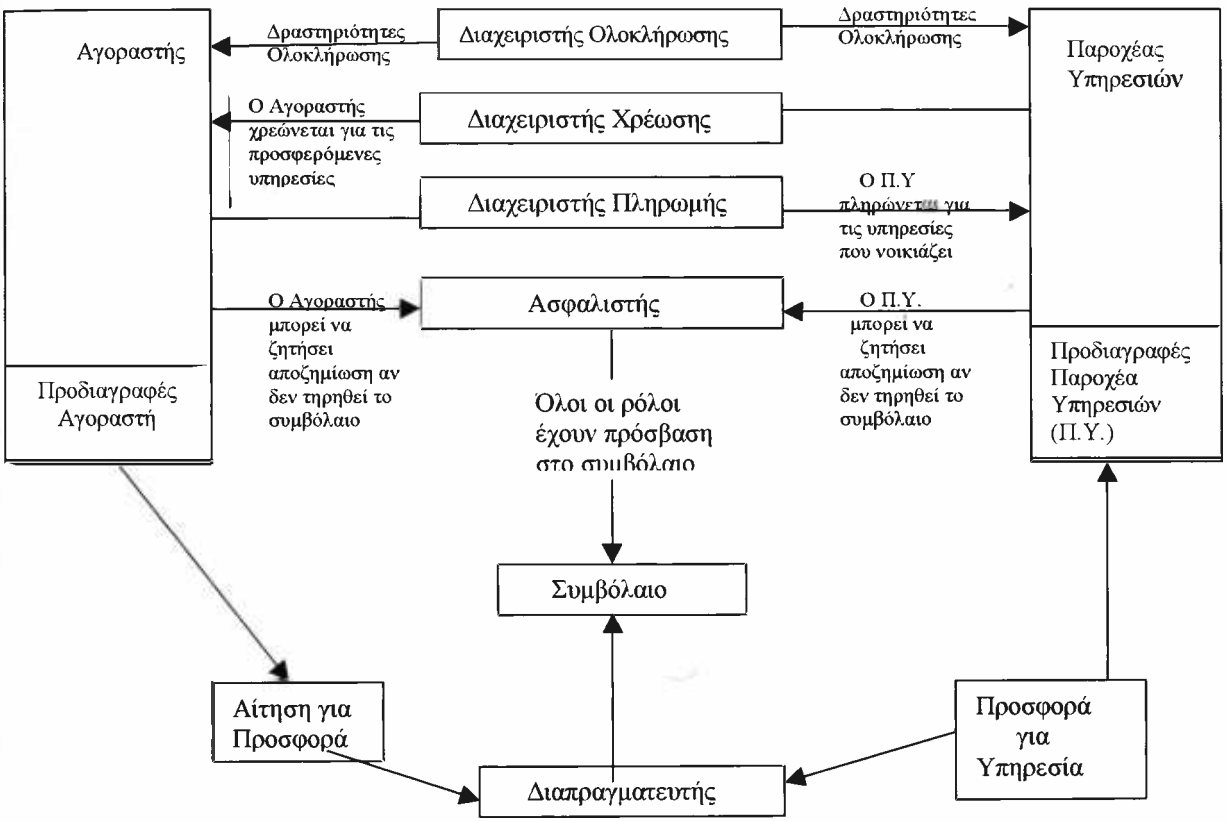
Τη φάση της διαπραγμάτευσης και της συμφωνίας εδραιώνει ένα συμβόλαιο μεταξύ ενός **Παροχέα Υπηρεσιών** και ενός **Αγοραστή**. Σε αυτή τη φάση του κύκλου ζωής της διαδικασίας υποτίθεται ότι έχει γίνει μια ταύτιση μεταξύ της αρχικής **αίτησης για προσφορά** και της τελικής **προσφοράς**. Παρόλα αυτά, μια τέτοια ταύτιση δείχνει μόνο συμβατότητα μεταξύ μιας αίτησης για προσφορά και της προσφοράς Υπηρεσίας. Οι διαπραγματεύσεις στη φάση διαπραγμάτευσης θα καθορίσουν εάν ο Παροχέας Υπηρεσιών και ο Αγοραστής δεσμευτούν εν όψη κάποιου νομικού συμβολαίου, το οποίο προσδιορίζει τις ευθύνες του Αγοραστή και του Παροχέα Υπηρεσιών.



Σχήμα 7

5.6.5 Φάση Ολοκλήρωσης και Διευθέτησης

Σε αυτή τη φάση, και ο Αγοραστής και ο Παροχέας Υπηρεσιών ολοκληρώνουν και διευθετούν τους όρους του συμβολαίου τους. Αν προκύψουν εξαιρέσεις και οι δύο μεριές δεν μπορούν να ολοκληρώσουν το συμβόλαιο, τότε μπορεί να ξαναρχίσει μια επαναδιαπραγμάτευση των όρων του συμβολαίου. Στο τέλος του συμβολαίου, η σχέση ολοκληρώνεται.



Σχήμα 9

ΚΕΦΑΛΑΙΟ 6ο

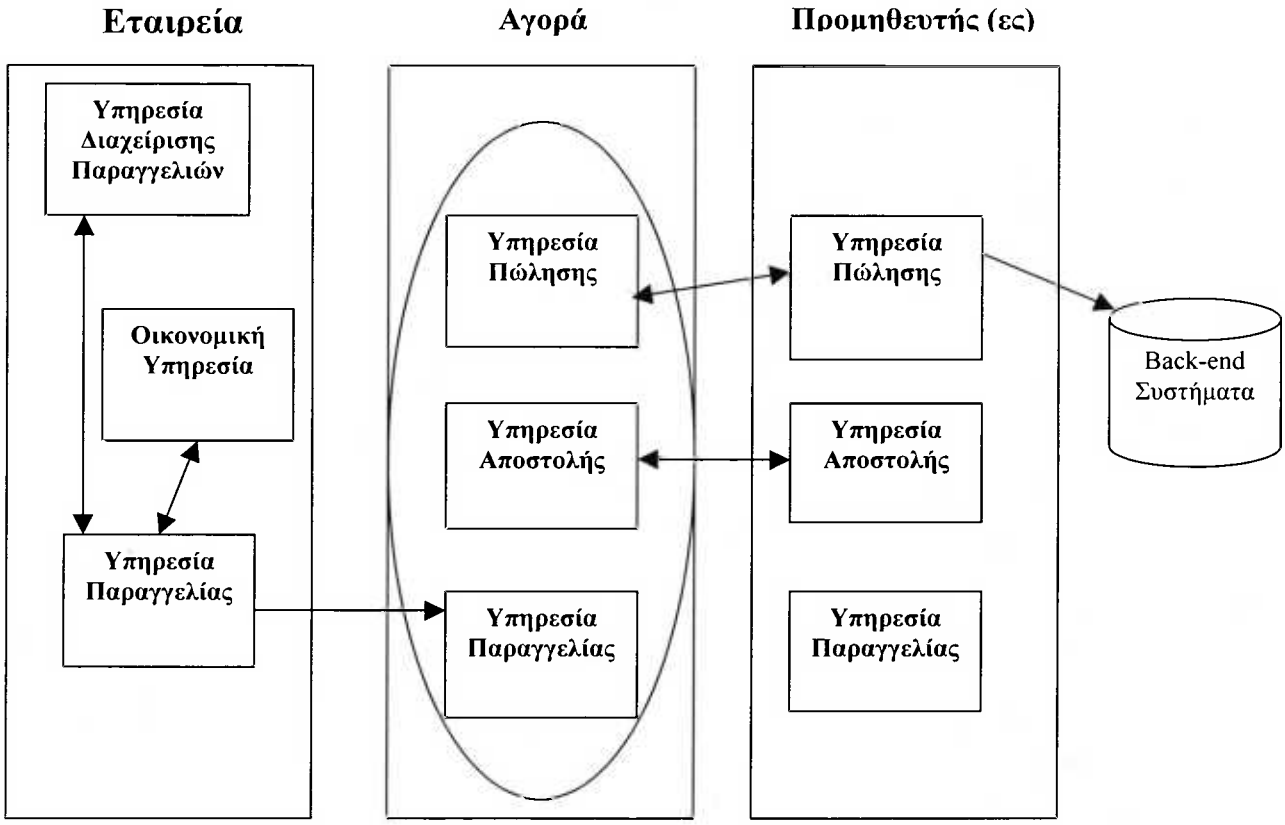
ΕΠΙΧΕΙΡΗΜΑΤΙΚΑ ΜΟΝΤΕΛΑ

6 ΕΠΙΧΕΙΡΗΜΑΤΙΚΑ ΜΟΝΤΕΛΑ

Η περιγραφή του μοντέλου των ηλεκτρονικών υπηρεσιών που προηγήθηκε δημιουργεί τις κατάλληλες προϋποθέσεις για εφαρμογή του σε συγκεκριμένα επιχειρηματικά μοντέλα. Η παγκόσμια βιβλιογραφία που σχετίζεται με το ηλεκτρονικό εμπόριο δεν είναι συνεπής όταν αναφέρεται στον όρο "επιχειρηματικό" μοντέλο.([11]) Παρόλα αυτά, μπορούμε να εντοπίσουμε περιπτώσεις εφαρμογής ηλεκτρονικών υπηρεσιών, στα πλαίσια επιχειρηματικών μοντέλων που έχουν προταθεί μέχρι σήμερα.

6.1 E-procurement

Αρχικά θα αναφερθούμε στο **e-procurement** μοντέλο. Το συγκεκριμένο αναφέρεται στη μαζική προμήθεια αγαθών και υπηρεσιών από ορισμένους προμηθευτές. Μέσω της υιοθέτησης του μοντέλου των υπηρεσιών επιτυγχάνεται μεγαλύτερος αριθμός προμηθευτών, γεγονός που οδηγεί σε χαμηλότερο κόστος προμήθειας, καλύτερη ποιότητα, όπως επίσης και βελτιωμένη διαδικασία παράδοσης ([10]). Η ηλεκτρονική διαπραγμάτευση και επισημοποίηση της συμφωνίας μέσω συμβολαίου, μπορεί να συμβάλλει δραματικά στη μείωση του χρόνου περάτωσης της συναλλαγής και στην αξιοπιστία. Για τους προμηθευτές, τα οφέλη που προκύπτουν αφορούν την αύξηση των ευκαιριών ακόμη και σε διεθνή κλίμακα καθώς και στην επίτευξη μικρότερου κόστους παράδοσης μιας προσφοράς.



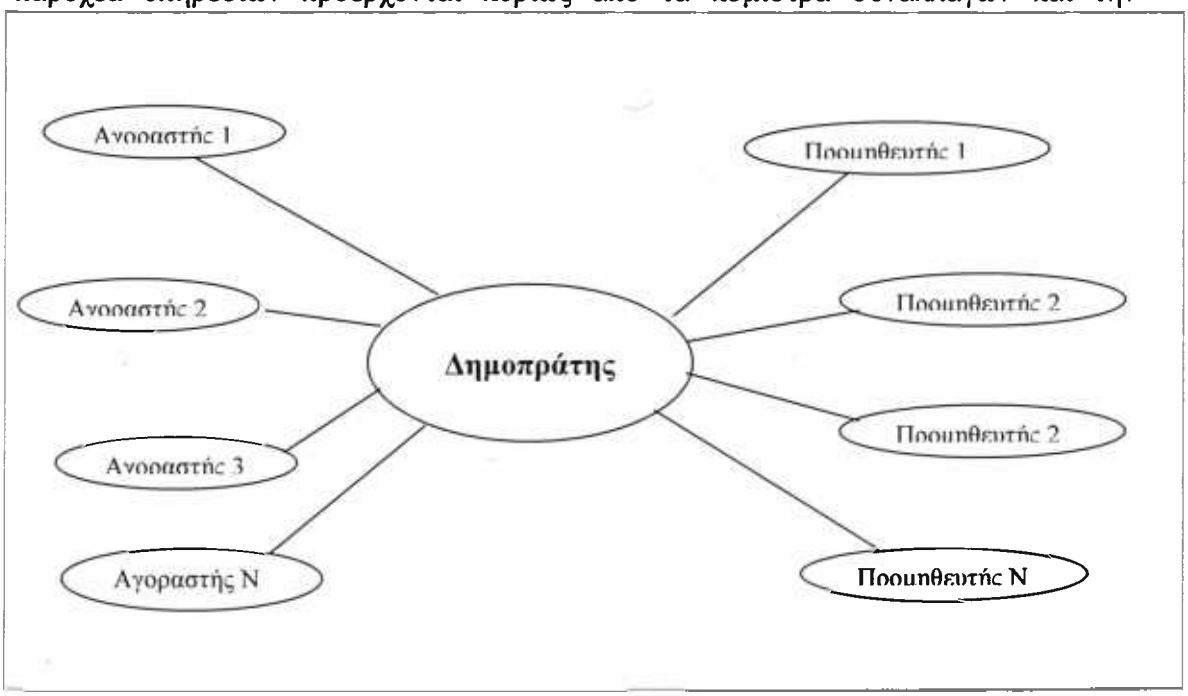
Σχήμα 9

Στο παραπάνω σχήμα παρατηρούμε τα εξής:
Στο ενδιάμεσο υπάρχει ένας πυρήνας στον οποίο καταχωρούνται όλες οι υπηρεσίες. Σε ποιον ανήκει ο συγκεκριμένος πυρήνας εξαρτάται από την εκάστοτε υλοποίηση. Στον πυρήνα, ο (οι) προμηθευτής (ες) εκχωρούν τις προσφερόμενες από αυτούς υπηρεσίες. Από την άλλη, η ενδιαφερόμενη επιχείρηση υποβάλλει ένα σχετικό αίτημα ζητώντας την κάλυψη κάποιων αναγκών βάσει συγκεκριμένων κριτηρίων. Το αίτημα μεταβιβάζεται στον ενδιάμεσο όπου και λαμβάνεται η σχετική απόφαση για το αν υπάρχει και ποια υπηρεσία του προμηθευτή θα καλύψει τις αιτούμενες ανάγκες του.

Με την άφιξη της υπηρεσίας Παραγγελίας, ανακαλύπτεται η υπηρεσία Πώλησης. η οποία διαφημίζεται στην ίδια περιοχή. Η υπηρεσία Πώλησης επικοινωνεί με τα back-end συστήματα του προμηθευτή ζητώντας επιβεβαίωση ή άρνηση στο αίτημα που εκφράζεται από την υπηρεσία Παραγγελίας. Στη συνέχεια, η υπηρεσία Πώλησης ανακαλύπτει και πάλι την υπηρεσία Παραγγελίας επιβεβαιώνοντας ή μη τη συμφωνία.

6.2 E-auction

Μια δεύτερη περίπτωση εφαρμογής μπορεί να θεωρηθεί το **e-auction** Οι ηλεκτρονικές δημοπρασίες προσφέρουν μια ηλεκτρονική υλοποίηση του παραδοσιακού όρου ([15]). Συνήθως, δεν περιορίζονται μόνο στην λειτουργία της δημοπρασίας, αλλά παρέχουν μια ολοκληρωμένη διαδικασία που περιλαμβάνει την επικύρωση της συμφωνίας, τις πληρωμές και την παράδοση. Οι πηγές εσόδων για τον παροχέα υπηρεσιών προέρχονται κυρίως από τα κόμιστρα συναλλαγών και την

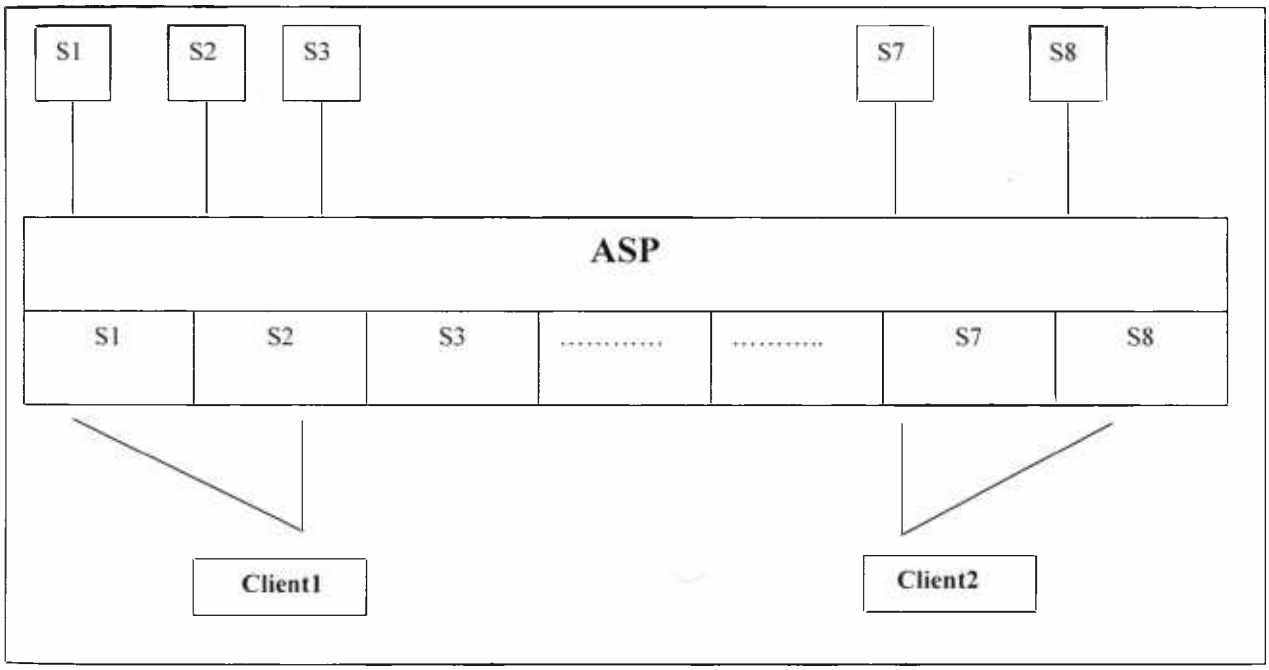


Σχήμα 10

Εξαιτίας της σύμφυτης ιδιότητας των συνδυαζόμενων ηλεκτρονικών υπηρεσιών να "διαπραγματεύονται", θεωρούμε αυτονόητη τη δυνατότητα υλοποίησης του προ-αναφερθέντος μοντέλου. Η αλληλουχία των ενεργειών μετά την υποβολή-διαφήμιση μιας υπηρεσίας θα μπορούσε να θεωρηθεί όμοια με αυτή που αναφέρθηκε στο προηγούμενο επιχειρηματικό μοντέλο της ηλεκτρονικής προμήθειας.

6.3 Third Party MarketPlaces

Το μοντέλο των **Third Party MarketPlaces** είναι αυτό στο οποίο επιδιώκεται να μετατραπεί η πλειονότητα των εφαρμογών στο Internet. Άμεση υλοποίηση το συγκεκριμένο μοντέλο βρίσκει στους ASP's.



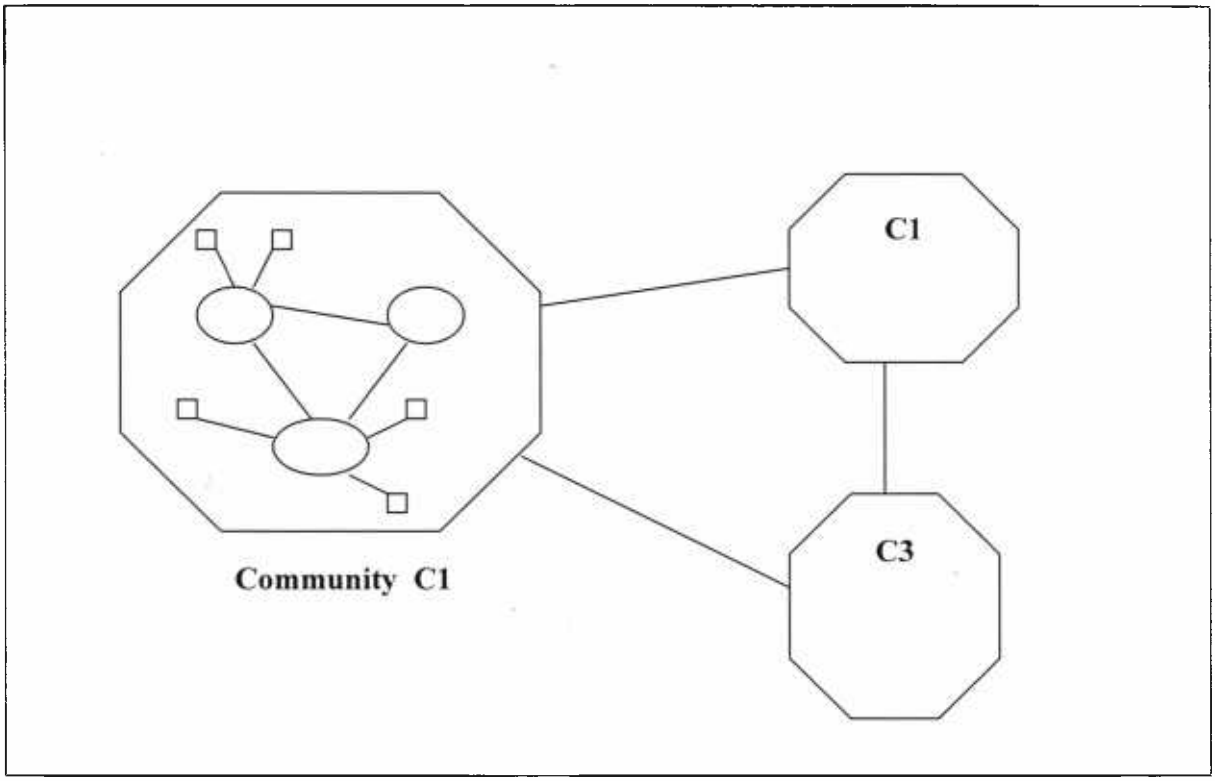
Σχήμα 11

Ο ASP στη συγκεκριμένη περίπτωση, παίζει τον ρόλο του διαμεσολαβητή (broker) ανάμεσα στην αγορά των υπηρεσιών που βρίσκεται από την μία πλευρά και στις απαιτήσεις των πελατών για συγκεκριμένες υπηρεσίες ([8]). Στην περίπτωση αυτή εφαρμόζεται το μοντέλο της χρέωσης σύμφωνα με την χρήση, ή σε συνδρομητική βάση εξασφαλίζοντας τα απαραίτητα έσοδα.

6.4 Virtual Communities

Το μοντέλο στο οποίο μπορούν να διαδραματίσουν ρόλο οι συνεργαζόμενες ηλεκτρονικές υπηρεσίες είναι αυτό των Virtual Communities. Σύμφωνα με αυτό, έχουμε την δυνατότητα συνασπισμού πολλών διαφορετικών υπηρεσιών σε πολλούς πυρήνες οι οποίοι συνδυαζόμενοι -συνεργαζόμενοι, συναποτελούν μια εικονική κοινότητα. Πολλές κοινότητες μαζί μπορούν να δημιουργήσουν ευρύτερους σχηματισμούς. Οι εικονικές κοινότητες δίνουν την ευκαιρία δημιουργίας νέων

υπηρεσιών μέσα από την συνεργασία κοινοτήτων που υιοθετούν μερικά από τα διαφορετικά επιχειρηματικά μοντέλα που αναφέρθηκαν.

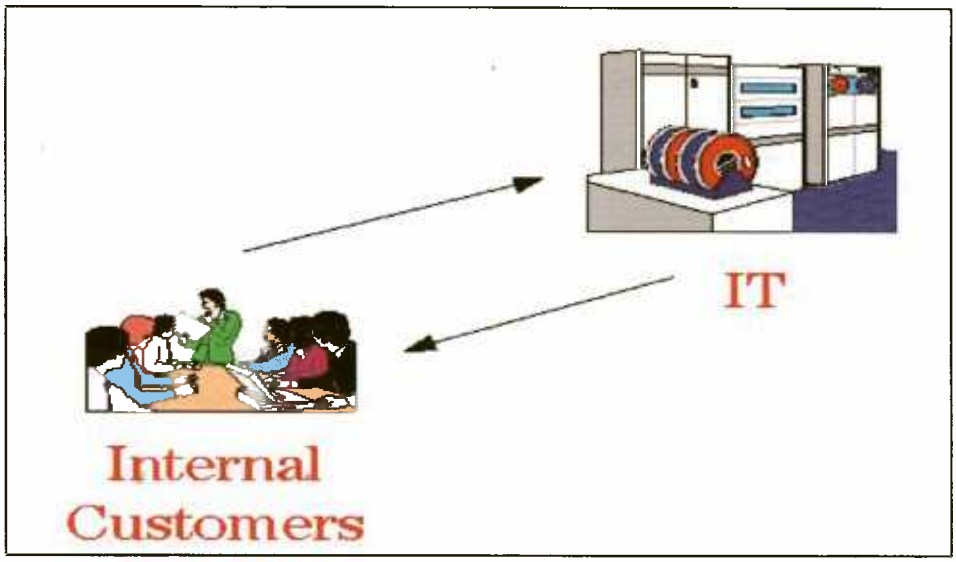


Σχήμα 12

Εκτός των επιδράσεων που θα προκύψουν εξαιτίας του μοντέλου των υπηρεσιών στα δια-επιχειρησιακά σχήματα, αξιοσημείωτες θα είναι και οι εξελίξεις στον εσωτερικό τρόπο οργάνωσης των οργανισμών ([8]). Στην συνέχεια θα αναφερθούμε σε πέντε πιθανά σενάρια σχετικά με τις επιπτώσεις που θα παρουσιαστούν στον χώρο των τμημάτων πληροφορικής. (Γενικεύσεις και σε άλλους χώρους μπορούν να γίνουν, αναλόγως βέβαια τη διάδοση της πληροφορικής στον αντίστοιχο χώρο).
Τα σενάρια περιλαμβάνουν:

6.5 Έλεγχος στο τμήμα Πληροφορικής (αποκλειστικά)

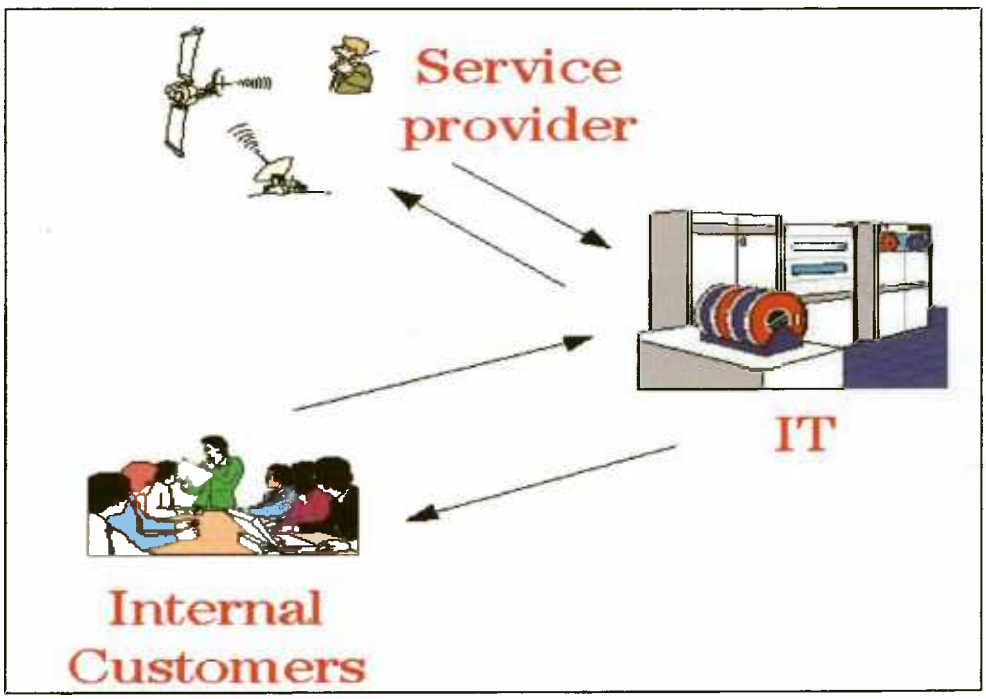
Σε ένα τέτοιο περιβάλλον τα διάφορα τμήματα ζητούν και λαμβάνουν υπηρεσίες σχεδόν αποκλειστικά από το τμήμα πληροφορικής. Από αυτή την οπτική γωνία, μπορούμε να πούμε ότι με αυτόν τον τρόπο διασφαλίζεται η σταθερότητα, η προγνωσιμότητα και η αξιοπιστία στην υποδομή. Σε μερικές περιπτώσεις η εξοικονόμηση οικονομικών πόρων είναι εφικτή, αλλά δεν υπάρχει ανταγωνισμός μεταξύ των παροχέων υπηρεσιών, ώστε να ενθαρρυνθεί η καινοτομία. Από την οπτική της επιχειρηματικότητας αυτό το μοντέλο παρέχει την ελάχιστη δυνατή ευελιξία, καθώς το μεγαλύτερο μέρος των υπηρεσιών και των λύσεων πληροφορικής εντάσσονται σε ένα ομογενοποιημένο περιβάλλον λύσεων που προσπαθεί να διατηρήσει την αξιοπιστία και την συνέπεια ([21]).



Σχήμα 13

6.6 Έλεγχος στο τμήμα Πληροφορικής σε συνδυασμό με εξωτερικούς παροχείς

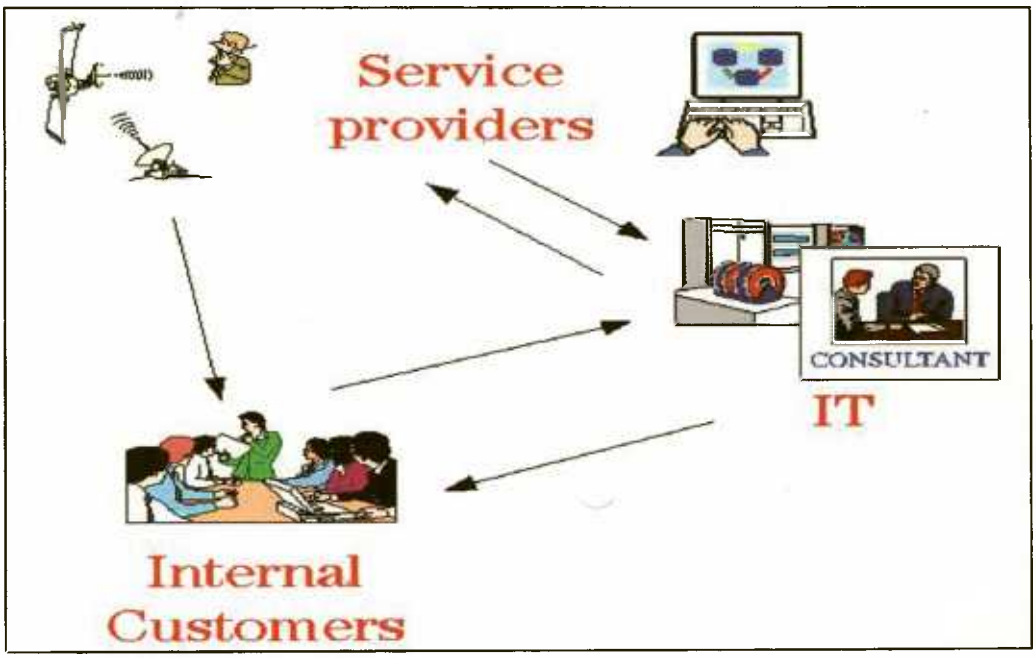
Και σε αυτήν τη περίπτωση το τμήμα πληροφορικής είναι αυτό που παρέχει το πλείστον των υπηρεσιών. Παρά όμως αυτό το καθεστώς, προβαίνει και σε αξιολόγηση κάποιων άλλων εξωτερικών προμηθευτών εμπλουτίζοντας την ποικιλία των παρεχόμενων υπηρεσιών. Στα αρνητικά ακόμη συγκαταλέγονται ο αργός χρόνος απόκρισης, η έλλειψη καινοτομίας και η πεποίθηση της ύπαρξης μιας λύσης για όλα τα προβλήματα.



Σχήμα 14

6.7 Το τμήμα Πληροφορικής ως διαμεσολαβητής-σύμβουλος

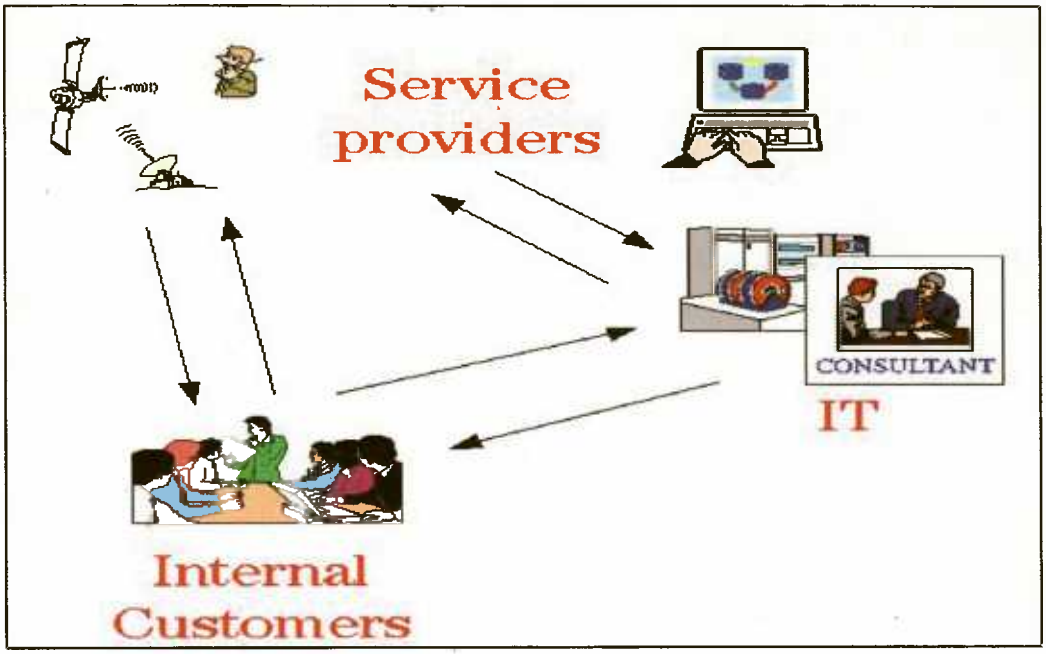
Στο σενάριο αυτό το τμήμα Πληροφορικής δρα ως διαμεσολαβητής ή έμπιστος σύμβουλος για τις ανάγκες των χρηστών. Μερικές υπηρεσίες παρέχονται εσωτερικά, ενώ κάποιες άλλες από εξωτερικούς προμηθευτές, ενώ σε ορισμένες περιπτώσεις οι εσωτερικοί χρήστες δέχονται απ' ευθείας τις υπηρεσίες των εξωτερικών προμηθευτών. Ο ρόλος του τμήματος πληροφορικής αφορά στην αξιολόγηση και στην διαχείριση των σχέσεων με τους εξωτερικούς προμηθευτές υπηρεσιών, στην ολοκλήρωση των λύσεων που παρέχουν. Το μοντέλο αυτό παρέχει λιγότερη υποδομή ελέγχου και προσωποποίησης, παρόλα αυτά έχει την δυνατότητα παραχώρησης κατευθύνσεων και αξιολογήσεων προς τις εξωτερικές πηγές υπηρεσιών.



Σχήμα 15

6.8 Το Τμήμα Πληροφορικής σαν διαμεσολαβητής με εξωτερικό ανταγωνισμό

Το συγκεκριμένο μοντέλο ενισχύει το προηγούμενο προσθέτοντας στοιχεία της αγοράς. Το τμήμα πληροφορικής εξακολουθεί να δρα σαν διαμεσολαβητής ή σύμβουλος. Μερικές υπηρεσίες παρέχονται εσωτερικά ενώ κάποιες άλλες από εξωτερικούς προμηθευτές. Αναλαμβάνει τους ίδιους ρόλους που αναφέραμε και προηγουμένως, παρόλα αυτά το βάρος στρέφεται στην αγορά καθώς εκεί υπάρχει μεγαλύτερη ευελιξία και επιλογή για δημιουργία νέων υπηρεσιών.



Σχήμα 16

6.9 Το τμήμα Πληροφορικής σαν ένας από τους πολλούς παροχείς

Στο μοντέλο αυτό το τμήμα πληροφορικής μετατρέπεται σε ένα από τους πολλούς προμηθευτές της επιχείρησης. Παρέχεται έτσι ένας τελείως ανοικτός τρόπος επιλογής. Αυτό πάντως που θα πρέπει να τονίσουμε είναι ότι το πιο δόκιμο σενάριο είναι αυτό του μετασχηματισμού του τμήματος πληροφορικής σε διαμεσολαβητή-σύμβουλο που θα ρυθμίζει την εύρεση, αξιολόγηση, σύμπραξη συμβολαίων για τις νέες υπηρεσίες.

ΚΕΦΑΛΑΙΟ 7ο

ΣΥΓΚΡΙΤΙΚΗ ΑΝΑΛΥΣΗ

7 ΣΥΓΚΡΙΤΙΚΗ ΑΝΑΛΥΣΗ

Στο πεδίο που τείνει να διαμορφωθεί, υποθήκη έχει τεθεί από διάφορους μεγάλους παίκτες της αγοράς, οι οποίοι με τις τεχνολογικές τους προτάσεις αλλά και τις συμμαχίες επιχειρούν να προσποριστούν το μεγαλύτερο κομμάτι της αγοράς. Κάνοντας μια σύνδεση με τα προηγούμενα θα μπορούσαμε να πούμε ότι το Internet βρίσκεται στο μεταίχμιο μεταξύ δύο διαφορετικών φάσεων. Στη μεν πρώτη και μέχρι πρότινος παρατηρούσαμε τις συνεχείς προσπάθειες των διαφόρων οργανισμών να εδραιώσουν την παρουσία τους στο δίκτυο συνδέοντας σε αυτό τις ιστοσελίδες τους. Σε αυτό το περιβάλλον, η εταιρεία που έμπαινε πρώτη σε κάποιο κλάδο θα ήταν η επικρατέστερη και για τη συνέχεια. (βλ. amazon.com). Η ελκυστικότητα, η φρεσκάδα και η ευκολία-χρήσης ήταν μερικά από τα θέλγητρα τα οποία κρατούσαν τον κόσμο σε κάποιες ιστοσελίδες, παρέχοντας διαφημίσεις κυρίως για την αντιμετώπιση του κόστους, αλλά παράλληλα και ορισμένες δωρεάν υπηρεσίες (π.χ. αναζητήσεις κ.τ.λ). Τέλος, κάποια οριζόντια Portals (Yahoo, Lycos, AOL) έκαναν την εμφάνιση τους, εδραιώνοντας την παρουσία τους.

Στο εξής και με την σταδιακή εισαγωγή του νέου μοντέλου των υπηρεσιών αρχίζει να διαφαίνεται μια νέα πραγματικότητα με τη δεύτερη φάση την οποία διέρχεται το Internet, η οποία σε πολλά σημεία διαφέρει πλήρως από τα μέχρι τώρα δεδομένα. Οι εταιρίες θα ανταγωνίζονται πλέον σε μια διαφορετική βάση όπου η σχέση προσφερομένων-τιμής, η ποιότητα και η ταχύτητα παράδοσης θα αποτελούν τα προς σύγκριση τεκμήρια. Από την πλευρά τους οι πελάτες θα έχουν την δυνατότητα πρόσβασης σε αυτές τις υπηρεσίες μέσω πολλών διαφορετικών συσκευών (π.χ κινητών τηλεφώνων, palmtops, τηλεοράσεων κ.λ.π.) και όχι μόνο από τον κλασσικό προσωπικό ηλεκτρονικό υπολογιστή, ο οποίος αποτέλεσε τον θεμέλιο λίθο της πρώτης φάσης. Τέλος, οι παροχείς υπηρεσιών όπως ISP's, ASP's τηλεπικοινωνιακοί οργανισμοί, εκμεταλλευόμενοι τις τεχνολογικές προτάσεις - όπως αυτές που θα αναφερθούν συγκρινόμενες στη συνέχεια- θα υιοθετήσουν μια ξεχωριστή οπτική για την αγορά.

Στην συνέχεια θα αναφερθούμε σε τρεις βασικούς ανταγωνιστές του χώρου, στην Sun Microsystems, την HP και την Microsoft, περιγράφοντας την οπτική που έχουν υιοθετήσει για τον χώρο των συνεργαζόμενων ηλεκτρονικών υπηρεσιών, ενώ στη συνέχεια θα παρατεθούν πλεονεκτήματα και μειονεκτήματα βάσει αυτών τα οποία έχουν προτείνει μέχρι στιγμής.

7.1 SUN Microsystems

Το όραμα της SUN για το θέμα εμφανίστηκε γύρω στα τέλη της δεκαετίας του 1980. Με την έλευση της ιδέας για ένα δικτυωμένο κόσμο υπηρεσιών συνεχίζει το όραμα της και στο νέο αιώνα. Όντας περισσότερο ένα όραμα περιορισμένο στον υπολογιστή περιγράφουν ένα WEB κόσμο όπου κάθε συσκευή (όχι μόνο υπολογιστής) θα έχει την δυνατότητα να συνδέεται στο δίκτυο με τρόπο παρόμοιο με αυτόν του τηλεφώνου. Η SUN οραματίζεται έναν πληροφοριακό κόσμο ανεξάρτητο πλατφόρμας, όπου τα προγράμματα και οι εφαρμογές θα γράφονται μία φορά και θα είναι άμεσα χρησιμοποιήσιμα, ανεξαρτήτως λειτουργικού συστήματος, ή δικτυακού ή άλλου λογισμικού.

Σύμφωνα με τις προβλέψεις της για το μέλλον υποστηρίζει ότι:

1. Το WEB θα είναι διαθέσιμο σε όλους με αξιοπιστία που προσεγγίζει αυτή ενός τηλεφώνου.
2. Η καινοτομία στη νέα οικονομία θα προκύψει σε ένα μετά-ιδιοκτησιακό κόσμο.
3. Σχεδόν κάθε θέμα επιχειρηματικής δραστηριοποίησης θα είναι θέμα υπηρεσιών πλέον.

Συμπληρώνοντας το όραμα της για το μέλλον η SUN θέτει τις παρακάτω προβλέψεις, οι οποίες συγκλίνουν με το αρχικό μοντέλο των ηλεκτρονικών υπηρεσιών:
1. Οι υπηρεσίες και όχι οι εφαρμογές θα είναι το κλειδί στη νέα οικονομία.
2. Η αγορά του Internet με όλες της τις εκδηλώσεις θα είναι πολύ μεγαλύτερη από την αγορά των PC.
3. Τα Portals θα είναι οι βασικές εφαρμογές της νέας εποχής.
4. Τα extranets και τα intranets θα συγκλίνουν σε ένα κοινό δίκτυο.
5. Η αυριανή αρχιτεκτονική της IT θα προσομοιάζει με το σημερινό τηλεπικοινωνιακό μοντέλο.
6. Οι εφαρμογές οι οποίες θα υλοποιούν τις υπηρεσίες θα είναι σχεδιασμένες για το δίκτυο.
7. Αυτό που σήμερα θεωρείται υψηλής απόδοσης υπολογιστική ισχύς θα αποτελέσει στο μέλλον δεσπόζουσα τάση ενώ παράλληλα θα είναι οικονομικότερη, χωρίς να αποτελεί το επίκεντρο της επιστήμης και του σχεδιασμού.
8. Νέοι παροχείς υπηρεσιών θα ανακύψουν ταχύτατα.
9. Η απλότητα και η ευκολία χρήσης θα είναι το χαρακτηριστικό για πολλούς νέους χρήστες.

Σύμφωνα με τα όσα αναφέρθηκαν είναι φανερό ότι το όραμα της SUN κινείται στο μοντέλο των υπηρεσιών και των portals. Για να υλοποιήσει σε κάποιο βαθμό τα οράματα της η εταιρεία προσπάθησε να παρουσιάσει μια σειρά εφαρμογών επεξεργασίας κειμένου, spreadsheets και επεξεργασίας γραφικών με τη μορφή υπηρεσίας στο δίκτυο, βασισμένες σε Java (Web Based Star Office) ([7]). Επίσης εξακολουθεί να προωθεί το περιβάλλον της Java σαν το απόλυτο στοιχείο συγγραφής εφαρμογών για το δίκτυο. Με την αναγγελία της τεχνολογίας Jini η οποία θα στηρίζεται στη Java, φιλοδοξεί να δημιουργήσει ένα περιβάλλον που θα επιτρέπει την εύκολη σύνδεση και λειτουργία των συσκευών και των υπηρεσιών στο δίκτυο. Εισάγεται επίσης η ιδέα για τη μη ιδιοκτησιακή αντίληψη των αρχιτεκτονικών όπου η κατοχή μονολιθικών λειτουργικών συστημάτων και εφαρμογών δεν είναι απαραίτητη. Τέλος, το Internet σε συνδυασμό με τα ήδη υπάρχοντα πρότυπα, (TCP/IP, HTML, HTTP) χρειάζεται την εισαγωγή νέων όπως το LDAP (για τη δομή καταλόγων), το Kerberos/PM (για την ασφάλεια), το Voice over IP, το XML κ.λ.π, για την επιτυχή εξέλιξη του.

7.2 HP

Τρεις είναι οι βασικοί τομείς στους οποίους φαίνεται να δραστηριοποιείται η εταιρεία υλοποιώντας το όραμα της:
1. Εφαρμογές έτοιμες προς χρήση (με τη μορφή υπηρεσιών)
2. Portals νέας γενιάς
3. Δυναμικό μοντέλο διαμεσολάβησης

Πιο αναλυτικά και για κάθε ένα από τα παραπάνω παρατηρούμε τα εξής:

7.2.1 Εφαρμογές έτοιμες προς χρήση (με τη μορφή υπηρεσιών)

Αποτελεί έναν από τους πρωταρχικούς στόχους της εταιρείας η παράδοση επιχειρηματικών εφαρμογών στο δίκτυο με βάση το μοντέλο της πληρωμής σύμφωνα με την χρήση, οδηγώντας με αυτό τον τρόπο στη μείωση του πληροφοριακού κόστους. Οι επιχειρήσεις έχουν την δυνατότητα να αναπτύξουν την δική τους επιχειρηματική στρατηγική μεταφέροντας οτιδήποτε δεν είναι άμεσα εφικτό ή οικονομικά σύμφορο μέσω μίσθωσης στις ηλεκτρονικές υπηρεσίες που είναι έτοιμες προς χρήση.

Απώτερο όραμα της HP σχετικά με το μοντέλο υπηρεσιών είναι η ύπαρξη ενοικιαζόμενων υπηρεσιών κάπου εγκατεστημένων στο δίκτυο, αντί της ανάγκης για ύπαρξη αντιγράφων της εφαρμογής στον πελάτη. Η κίνηση αυτή δίνει νέα ώθηση στον ρόλο της διαμεσολάβησης. Οι ASP's (Application Service Providers) δρουν πλέον ως ενδιάμεσοι, μεταξύ των κατασκευαστών λογισμικού και των αντίστοιχων χρηστών. Υπάρχουν διάφορες δυνάμεις οι οποίες ενισχύουν την παραπάνω φιλοσοφία. Μεταξύ αυτών, η διαπίστωση του περιορισμένου αριθμού καταρτισμένου προσωπικού για την λειτουργία πολύπλοκων συστημάτων, ιδίως σε μικρές και μεσαίες επιχειρηματικές μονάδες. Επίσης το αυξημένο κόστος, η πολυπλοκότητα και οι χρονοβόρες διαδικασίες έχουν οδηγήσει τους ασχολούμενους με την διοίκηση της πληροφορικής σε νέες κατευθύνσεις, πέρα από τις κλασσικές.

Υπάρχουν βέβαια και οι επιφυλάξεις για τις νέες προτάσεις σχετικά με την ασφάλεια, το εύρος ζώνης, αλλά και την ψυχολογία του να μην έχεις υπό την ιδιοκτησία σου την εφαρμογή που σε εξυπηρετεί. Παράγοντες που πρέπει να ληφθούν υπ' όψιν για την μετάβαση στο μοντέλο των υπηρεσιών είναι οι ακόλουθοι:

1. Οι υπηρεσίες πρέπει να είναι σταθερές και σύμφωνες με τις ανάγκες των οργανισμών.
2. Όλοι οι παροχείς πρέπει να έχουν πιστοποιητικά φερεγγυότητας.
3. Η ασφάλεια είναι το επίμαχο πρωτεύον χαρακτηριστικό.
4. Το κόστος από την χρήση της υπηρεσίας πρέπει να είναι λογικό και χαμηλότερο από το ήδη υπάρχον.
5. Η χρέωση πρέπει να είναι άμεση και ακριβής.
6. Οι υπηρεσίες πρέπει να ενσωματώνονται εύκολα στις λειτουργίες της επιχείρησης.
7. Οι υπηρεσίες πρέπει να είναι αξιόπιστες και η ποιότητά τους εγγυημένη.

Βασικό κοινό αυτών των προσδοκιών αποτελούν οι μικρές και μεσαίες επιχειρήσεις οι οποίες αδυνατούν να υιοθετήσουν την απαραίτητη σήμερα πληροφοριακή υποδομή (π.χ. ERP συστήματα). Αν και αναμένεται μια εξίσου θεαματική στροφή των μεγάλων οργανισμών σε τέτοιου είδους αντιμετώπιση των πληροφοριακών αναγκών τους.

7.2.2 Portals νέας γενιάς

Το επόμενο σημείο αναφοράς της HP αποτελούν τα Portals νέας γενιάς. Καθώς τα Portals κατά την πρώτη φάση του Internet χαρακτηρίζονταν από μια ευρύτητα, ήταν/είναι πλέον στραμμένα κυρίως προς τον καταναλωτή ενώ αναφέρονται σε πολλά διαφορετικά θέματα (οριζόντια Portals) και αρχίζουν πλέον σταδιακά να μετασχηματίζονται. Έτσι, έχουν κάνει ήδη την εμφάνιση τους business-to-business (B2B) portals με έμφαση σε συγκεκριμένες περιοχές ενδιαφέροντος, καθώς και για συγκεκριμένες κάθετες βιομηχανίες. Στα πλαίσια των τελευταίων έχουν σταδιακά αρχίσει να εισάγονται έννοιες, όπως το κόστος των συναλλαγών με τους ενδιάμεσους κ.λ.π.

7.2.3 Δυναμική Διαμεσολάβηση

Το θέμα αυτό είναι ίσως και το σημαντικότερο μιας και είναι αυτό που πλησιάζει σε μεγάλο βαθμό το όραμα της HP για ένα πλήρως εφαρμόσιμο μοντέλο συνεργαζόμενων ηλεκτρονικών υπηρεσιών. Η ιδέα που βρίσκεται πίσω από την δυναμική διαμεσολάβηση, αφορά την προμήθεια των οργανισμών με ηλεκτρονικές υπηρεσίες σχεδόν άμεσα, προσδιορίζοντας συγκεκριμένες ιδιότητες (όπως τιμή, διαθεσιμότητα, ποιότητα κ.λ.π.). Στα πλαίσια ενός τέτοιου μοντέλου ο ενδιάμεσος θα πρέπει:

- Να ανακαλύψει τις υπηρεσίες που καλύπτουν στον μεγαλύτερο βαθμό τις προσδιοριζόμενες από τον πελάτη ανάγκες.
- Να διαπραγματευτεί τους όρους της συναλλαγής.
- Να συνθέσει την τελική λύση, πιθανόν από διαφορετικούς παροχείς υπηρεσιών (εάν αυτό είναι απαραίτητο).
- Να παρουσιάσει το αποτέλεσμα στον πελάτη του.
- Να παρακολουθήσει το αποτέλεσμα της συναλλαγής για λόγους εξασφάλισης ποιότητας, αλλά και χρέωσης.
- Να διαφυλάξει την ανωνυμία του πελάτη και του παροχέα, αν αυτό απαιτείται.

Ασφαλώς η δυναμική διαμεσολάβηση αποτελεί μια ριζοσπαστική τεχνολογία. Οι ενδεχόμενες επιπτώσεις στους οργανισμούς που θα υιοθετήσουν το νέο τρόπο οργάνωσης δεν είναι ακόμη σίγουρο ότι έχουν γίνει απόλυτα κατανοητοί.

Σε γενικές γραμμές οι αναμενόμενες επιπτώσεις είναι οι ακόλουθες:

- Το να είναι ένας οργανισμός πρώτος σε μία συγκεκριμένη αγορά, καθίσταται δευτερεύουσας σημασίας μιας και η απόφαση για χρησιμοποίηση μιας συγκεκριμένης υπηρεσίας, εξαρτάται από το ποιος παροχέας υπηρεσιών μπορεί να ικανοποιήσει καλύτερα τις ανάγκες τους αγοραστή την συγκεκριμένη στιγμή.
- Οι παραδοσιακές αλυσίδες αξίας θα επηρεαστούν άμεσα, καθώς οι ενδιάμεσοι μπορούν να οδηγήσουν συγκεκριμένους παροχείς να καλύψουν μια ανάγκη, ενώ λίγα λεπτά αργότερα το αίτημα για εξυπηρέτηση να μπορεί να καλυφθεί από ένα διαφορετικό συνδυασμό προμηθευτών.
- Οι παροχείς υπηρεσιών είναι σίγουρο ότι θα αναγκαστούν να γίνουν πιο δραστήριοι και οικονομικά ανταγωνιστικοί, καθώς το ενδιαφέρον θα μετατοπιστεί από την εμπιστοσύνη της φίρμας και της μακρόχρονης σχέσης, στην δυνατότητα αυτών να παραδώσουν άμεσα τις υπηρεσίες.
- Νέες ευκαιρίες, κυρίως οικονομικές, ανοίγονται για τους ενδιάμεσους, καθώς θα υπάρχει επίσης μια μετατόπιση από τα οριζόντια portals προς τα portals κάθετης λειτουργίας, τους ISP's, ASP's και τους τηλεπικοινωνιακούς οργανισμούς.
- Οι αγοραστές είναι σίγουρο ότι δεν θα συνεχίσουν να είναι εγκλωβισμένοι σε μακροχρόνια συμβόλαια, αλλά αντίθετα θα έχουν την δυνατότητα να επιλέξουν την πιο συμφέρουσα υπηρεσία την στιγμή που την χρειάζονται, ενώ παράλληλα να αποτελεί εγγύηση ότι η τιμή που θα καταβάλουν θα είναι η πλέον ανταγωνιστική εξαιτίας της ανταγωνιστικής φύσης της δυναμικά διαμεσολαβούμενης αγοράς.

Παρά τα όσα θετικά υποστηρίζουν ότι θα κομίσουν οι ηλεκτρονικές υπηρεσίες, υπάρχουν και κάποια γκρίζα σημεία τα οποία πρέπει να μελετηθούν. Λόγοι νομικοί, τεχνολογικοί, πολιτικοί κ.α., αποτελούν εμπόδια στην πλήρη υιοθέτηση του μοντέλου. Ήδη σήμερα στις περιπτώσεις όπου εφαρμόζεται το outsourcing, συνήθως συνοδεύεται από αυστηρές ρήτρες για τα εμπλεκόμενα μέρη. Συνεπώς το ερώτημα το οποίο τίθεται είναι το αν και κατά πόσο οι οργανισμοί (και ιδιαίτερα οι μεγάλοι) θα εμπιστευτούν τις

λειτουργίες τους σε άγνωστους παροχείς υπηρεσιών. Επίσης, μια ακόμη σημαντική παράμετρο αποτελεί το γεγονός του ελέγχου των υπηρεσιών από τους παροχείς.

Δεν πρέπει όμως να είμαστε τόσο απαισιόδοξοι για το μέλλον του μοντέλου το οποίο προτείνεται, μιας και οι ίδιες επιφυλάξεις και ερωτηματικά υπήρχαν και πριν τη γιγάντωση του παγκόσμιου ιστού, στις διαστάσεις που εκτείνεται σήμερα. Παρόλα αυτά, η αποδοχή ανοικτών προτύπων για ανοικτές αγορές μπορεί να δημιουργήσει πρόσφορο έδαφος για νέες τεχνολογίες. Όπως τα TCP/IP, HTML, HTTP,FTP αποτέλεσαν θεμέλια για την εξέλιξη του WWW, έτσι και τα νέα πρότυπα XML, JINI, E-SPEAK, καθώς και οι βελτιώσεις στο εύρος ζώνης μπορούν να αποδειχτούν οι ενισχυτές στην προσπάθεια για δυναμική διαμεσολάβηση.

7.3 Microsoft

Η Microsoft είναι και αυτή από τις εταιρίες οι οποίες δραστηριοποιούνται στον χώρο των ηλεκτρονικών υπηρεσιών. Στόχος της είναι να προσελκύσει μικρομεσαίες κυρίως επιχειρήσεις, χωρίς ιδιαίτερη υποδομή και με προθέσεις που δεν βασίζονται σε υπέρογκους προϋπολογισμούς. Κύριο άξονα της πολιτικής της Microsoft αποτελεί το πλαίσιο εργασίας BizTalk, το οποίο θα αναλυθεί σε μεγαλύτερη λεπτομέρεια αργότερα. Το συγκεκριμένο περιβάλλεται από μια σειρά άλλων προϊόντων που αποκαθιστούν τον "διάλογο" μεταξύ διαφορετικών επιχειρήσεων. Το BizTalk είναι ένα προϊόν προσανατολισμένο στην XML, το οποίο επιτρέπει την δημιουργία ενός λεξιλογίου επικοινωνίας μεταξύ των επιχειρηματικών διαδικασιών δύο ή περισσότερων επιχειρήσεων.

Όσον αφορά το χώρο των έτοιμων υπό μορφή υπηρεσιών- εφαρμογών, η Microsoft ανακοίνωσε την διάθεση του γνωστού προϊόντος της Microsoft Office 2000, ως υπηρεσία εγκαταστημένη στο δίκτυο. Στο χώρο των Portals, η εταιρεία δίνει ιδιαίτερη έμφαση ιδίως μέσω του δικού της portal MSN (Microsoft Network), το οποίο φιλοδοξεί να μετασχηματίσει σε μια ενεργή αγορά ηλεκτρονικού εμπορίου.

Παρόλα αυτά, η Microsoft εξακολουθεί να τηρεί μια οπτική της οποίας βασικός άξονας είναι το PC, έχοντας κατά νου και την εξυπηρέτηση άλλων συσκευών (π.χ. palmtops, κινητών τηλεφώνων κ.λ.π.). Το όραμα που εκφράζει η Microsoft θα μπορούσε να χαρακτηριστεί απλούστερο και γι' αυτό εύκολα υλοποιήσιμο, άρα και αποδεκτό.
Στην συνέχεια θα αναφερθούμε σε περισσότερες λεπτομέρειες κάνοντας περαιτέρω σχόλια στις τεχνολογικές προτάσεις των κατασκευαστών που αναφέραμε.

7.4 ΒΑΣΙΚΕΣ ΤΕΧΝΟΛΟΓΙΚΕΣ ΠΡΟΤΑΣΕΙΣ

7.4.1 HP E-speak

Το e-speak αποτελεί τη βασική τεχνολογική πρόταση της HP για τον χώρο των συνεργαζόμενων ηλεκτρονικών υπηρεσιών. Πιο συγκεκριμένα, το e-speak είναι μια ανοικτή πλατφόρμα βασισμένη σε πρότυπα, με σκοπό την δημιουργία, σύνθεση, μεσολάβηση, διαχείριση και πρόσβαση σε υπηρεσίες ευρισκόμενες στο Internet. Το e-speak επιτρέπει την εσωτερική ανάπτυξη αλλά και τις ασφαλείς αλληλεπιδράσεις πίσω από firewalls. Η αρχιτεκτονική του επιτρέπει την συνεργασία με τις περισσότερες πλατφόρμες λογισμικού και υλικού, ενώ έχει την δυνατότητα διαχείρισης μεγάλου

αριθμού πόρων ([5]). Το e-speak έχει επίσης τη δυνατότητα να λειτουργήσει σε ένα απαιτητικό περιβάλλον όπως αυτό του Internet, συχνά ανάμεσα σε πολλές διαφορετικές ασύμβατες τεχνολογίες. (*Η περιγραφή που αφορά το e-speak, αλλά και τις υπόλοιπες τεχνολογικές προτάσεις που θα αναφερθούν στη συνέχεια, δεν είναι σε τεχνικό επίπεδο. Αντίθετα, γίνεται προσπάθεια για ανώτερη προσέγγιση σε επίπεδο εφαρμογής).*

Τα βασικά λειτουργικά στοιχεία του e-speak είναι τα ακόλουθα:

- Διαμεσολάβηση: Αναφέρεται στην ύπαρξη μιας κεντρικής οντότητας, όπου οι υπηρεσίες καταχωρούνται από τους παραγωγούς τους και στην συνέχεια "δημοπρατούνται" ανάλογα με τις αιτήσεις των πελατών.
- Δυναμική ανακάλυψη: Είναι η δυνατότητα εύρεσης μιας ηλεκτρονικής υπηρεσίας, χρησιμοποιώντας τις ιδιότητες της υπηρεσίας και όχι το όνομά της. Κατά την καταχώρηση της υπηρεσίας ο παροχέας δημιουργεί μια περιγραφή της , αναφέροντας τις ιδιότητές της. Προσδιορίζοντας ο χρήστης της υπηρεσίας ιδιότητες όπως το κόστος, η ποιότητα υπηρεσιών, οι προτιμώμενοι παροχείς, η ανακάλυψη καθίσταται ταχύτερη και σαφέστερη.
- Διαπραγμάτευση: Μετά την ανακάλυψη της υπηρεσίας το e-speak παρέχει την δυνατότητα περαιτέρω διαπραγμάτευσης ανάμεσα στον παροχέα και τον χρήστη της υπηρεσίας ακόμη και πέρα από τα κριτήρια τα οποία είχαν τεθεί αρχικά.
- Παρακολούθηση: Όταν ένας χρήστης χρησιμοποιήσει μια υπηρεσία, το e-speak παρέχει την δυνατότητα ελέγχου της συναλλαγής εξασφαλίζοντας την απ'-άκρη-σε-άκρη ασφαλή παράδοση της υπηρεσίας. Επίσης, λειτουργίες όπως η χρέωση σύμφωνα με τη χρήση εξασφαλίζονται μέσα από την συγκεκριμένη λειτουργία.
- Σύνθεση: Η σύνθεση είναι η δυνατότητα συνδυασμού υπαρχόντων ηλεκτρονικών υπηρεσιών, με σκοπό την δημιουργία νέων βελτιωμένων υπηρεσιών εξειδικευμένων προδιαγραφών. (Σημ. Η συγκεκριμένη λειτουργία δεν είναι άμεσα διαθέσιμη από την τεχνολογική πλατφόρμα του e-speak).

Στην προηγούμενη ενότητα αναφερθήκαμε σε τρεις βασικές παραμέτρους που χαρακτηρίζουν τα οράματα των κατασκευαστών για τον χώρο των συνεργαζόμενων υπηρεσιών. Στην συνέχεια θα αναφερθούμε στον τρόπο με τον οποίο το e-speak καλύπτει τις ανάγκες αυτών των παραμέτρων ([14]):

α) **Έτοιμες Ηλεκτρονικές Υπηρεσίες**: Το e-speak μπορεί να δράσει ως βασικός ενδιάμεσος στα πλαίσια ενός ASP (Application Service Provider) φέρνοντας σε επαφή πελάτες με εφαρμογές, συγκεντρώνοντας πληροφορίες τιμολόγησης αλλά και ποιότητας της υπηρεσίας κατά τη σύνδεση, ενώ τέλος μπορεί να προβεί άμεσα σε προσθήκη απαιτούμενων υπηρεσιών.

β) **Portals**: Το e-speak μπορεί να παρέχει τη βασική λειτουργία της αναζήτησης υπηρεσιών τόσο σε κάθετα, όσο και σε οριζόντια portals. Η αναζήτηση γίνεται με βάση τις ιδιότητες της υπηρεσίας και όχι το όνομά της. Επίσης, πέρα από την δυνατότητα αναζήτησης βάσει συγκεκριμένων κριτηρίων του πελάτη, βασικό χαρακτηριστικό είναι ότι αυτή γίνεται με τον βέλτιστο τρόπο, βοηθώντας στην χρέωση των υπηρεσιών.

γ) **Δυναμική Διαμεσολάβηση**: Είναι το αντικείμενο για το οποίο σχεδιάστηκε το e-speak, αποτελώντας το συγκριτικό πλεονέκτημα της HP σε σχέση με τους ανταγωνιστές της. Το e-speak έχει την δυνατότητα να φέρει σε πέρας ηλεκτρονικές συναλλαγές υπηρεσιών από την αρχή ως το τέλος, μέσω της διαφήμισης, της

ανακάλυψης, της διαμεσολάβησης, της σύνθεσης, της παρακολούθησης και της χρέωσης που μπορεί να επιτύχει. Ένα σύγχρονο επιχειρηματικό μοντέλο το οποίο μπορεί να υποστηριχθεί μέσω αυτής της τεχνολογίας είναι το outsourcing, στο οποίο μπορούν να καταφύγουν διάφοροι οργανισμοί εξοικονομώντας μεγάλα κονδύλια.

Τα προβλήματα τα οποία ενδεχομένως να αντιμετωπίσει η συγκεκριμένη πρόταση αφορούν κυρίως τον πολύ φιλόδοξο χαρακτήρα της και κατ' επέκταση τον μεγάλο κύκλο υιοθέτησης σαν τρέχουσα τεχνολογία, άμεσα χρησιμοποιήσιμη. Πολλά είναι εκείνα τα στοιχεία τα οποία πρέπει να ρυθμιστούν πριν πούμε με βεβαιότητα, ότι η καθόλα πολλά υποσχόμενη λύση που προαναφέραμε είναι σε θέση να εκπληρώσει τα υποσχόμενα. Επίσης, πολύ έντονος είναι ο ανταγωνισμός από αντίστοιχα προϊόντα άλλων εταιριών, ενώ σημαντικός είναι και ο παράγοντας της υιοθέτησης που θα λάβει από τους ανθρώπους της αγοράς.

7.4.2 SUN JINI

Το JINI της Sun είναι μια τεχνολογία που φιλοδοξεί να δώσει λύσεις στις πολυπλοκότητες του σύγχρονου υπολογιστικού περιβάλλοντος. Η ιδέα πίσω από αυτή την τεχνολογία είναι ότι συσκευές και υπηρεσίες μπορούν να ανακαλυφθούν στο δίκτυο χρησιμοποιώντας το JINI ([6]). Οι συσκευές έχουν την δυνατότητα να αυτο-εγκατασταθούν χωρίς την παρέμβαση των χρηστών και την απαίτηση για πολύπλοκες διαδικασίες εγκατάστασης επιπρόσθετου λογισμικού. Μέσω του JINI, η Sun φιλοδοξεί να δημιουργήσει έναν κόσμο όπου οι συσκευές και οι υπολογιστές, τρέχοντας το Java Virtual Machine, έχουν τη δυνατότητα να διασυνδεθούν και να εντοπιστούν μεταξύ τους στο δίκτυο.
Βασικά στοιχεία της τεχνολογίας είναι οι ακόλουθες βασικές υπηρεσίες:

- Αναζήτηση: Η Sun περιγράφει την υπηρεσία αυτή σαν ένα τηλεφωνικό πίνακα που βοηθά τη σύνδεση πελατών με υπηρεσίες. Οι υπηρεσίες καταχωρούνται μέσω της υπηρεσίας, ενώ οι πελάτες μπορούν με τη σειρά τους να τους ανακαλύψουν.
- Ανακάλυψη: Οι συσκευές χρησιμοποιούν την υπηρεσία ανακάλυψης για να βρουν τις διαδικασίες αναζήτησης.
- Συμμετοχή: Οι συσκευές χρησιμοποιούν την υπηρεσία για να ενταχθούν στο δίκτυο.

Αν και η Sun υποστηρίζει ότι είναι παρόν για να παράξει σύνδεση υπηρεσιών στο Internet, βασικός στόχος κατά την εμφάνιση του ήταν η μονάδα εργασίας και ο οικιακός χρήστης. Τα περισσότερα παραδείγματα της χρήσης του, αναφέρονται στα πλεονεκτήματα της διασύνδεσης συσκευών, όπως η TV, το DVD, οι κάμερες, το ραδιόφωνο, οι εκτυπωτές κ.τ.λ. Το JINI εξαρτά την διάδοση και την επιτυχία του από την επιτυχία της JAVA.

7.4.3 Microsoft BizTalk

Πολύ πρόσφατα (μέσα του 1999), η Microsoft ανακοίνωνε το πλαίσιο εργασίας BizTalk. Τα βασικά πλεονεκτήματα που εισάγει σύμφωνα πάντα με τη Microsoft, είναι ότι ενισχύει την ολοκλήρωση των εφαρμογών εντός του οργανισμού, βοηθά στην καλύτερη επικοινωνία μεταξύ επιχειρήσεων, ενώ παρέχει πλουσιότερο περιεχόμενο για τις on-line αγορές ([24]).

Το BizTalk είναι βασισμένο σε ορισμένα XML σχήματα (schemas) ([29]). Λεξικά δημιουργούνται βάσει αυτών των σχημάτων με σκοπό την αντιστοίχηση των διαδικασιών μεταξύ των εταιριών. Στις προθέσεις του περιλαμβάνεται η ελπίδα για δημιουργία λεξικών που καλύπτουν τόσο τις κάθετες (τράπεζες, ενέργεια κ.τ.λ), όσο και τις οριζόντιες αγορές. Οι προσπάθειες γύρω από το BizTalk επικεντρώνονται γύρω από τις αλληλεπιδράσεις μεταξύ των επιχειρήσεων ([26]). Περισσότερο δε στρέφονται προς τις μικρές και μεσαίες επιχειρήσεις οι οποίες είχαν απορρίψει το EDI ως μέσο επικοινωνίας, κυρίως λόγω του υψηλού κόστους υιοθέτησης.

Σύμφωνα με εκτιμήσεις, το BizTalk αναμένεται να υιοθετηθεί από την αγορά κυρίως λόγω της εύκολης χρήσης του, αλλά και του απλού στόχου που κομίζει. Και αυτό με τη σειρά του βρίσκεται σε συνεχή βελτίωση.

7.5 Σύγκριση : Επικαλυπτόμενες ή αλληλοσυμπληρούμενες υπηρεσίες

Στο σημείο αυτό θα προβούμε σε μια σύγκριση των τεχνολογιών που αναφέρθηκαν, σημειώνοντας ομοιότητες και διαφορές μεταξύ των προσεγγίσεων της HP, της SUN και Microsoft. Επίσης θα αναφερθούμε σε ενδεχόμενους τρόπους σύγκλισης των ανωτέρω τεχνολογιών.

Σε ανώτερο επίπεδο και οι τρεις συγκλίνουν υποστηρίζοντας ότι οι προτάσεις τους αποτελούν σημείο αναφοράς για τον χώρο των ηλεκτρονικών υπηρεσιών.

- Η Sun υποστηρίζει ότι η Java είναι η κοινή γλώσσα που συνδέει τα πάντα στον χώρο του Internet.
- Η Microsoft υποστηρίζει ότι κατάφερε να δημιουργεί λογισμικό, το οποίο μιλάει την γλώσσα των επιχειρήσεων.
- Τέλος, η HP με το e-speak υποστηρίζει ότι δημιούργησε ένα κοινό πλαίσιο όπου οι ηλεκτρονικές υπηρεσίες μπορούν να επικοινωνούν μεταξύ τους χρησιμοποιώντας κοινή γλώσσα.

Συγκρίνοντας, πέρα από τα όποια φραστικά παιχνίδια, τα οράματα των ανταγωνιστών παρατηρούμε ότι περισσότερο μιλούν για έτοιμες εφαρμογές (υπό μορφή υπηρεσιών) και για τα portals. Εκεί όπου το όραμα της HP υπερισχύει έναντι των άλλων είναι στο χώρο της διαμεσολάβησης έναντι των υπηρεσιών, στην ιδέα της άμεσης προμήθειας υπηρεσιών καθορίζοντας συγκεκριμένες ιδιότητες. Καμία από τις άλλες ανταγωνίστριες εταιρίες δεν συζητά σε λεπτομέρεια το συγκεκριμένο θέμα. Η Sun ενδεχομένως έρχεται πιο κοντά στην ιδέα της υπηρεσίας όταν αναφέρεται στο Jini, παρόλα αυτά τα περισσότερα παραδείγματα που αναφέρει σχετίζονται περισσότερο με υπηρεσίες που αφορούν το hardware, παρά σε υπηρεσίες Software χαρακτηριζόμενες από ιδιότητες.

Η Microsoft φαίνεται να στοχεύει κυρίως στο B2B ηλεκτρονικό εμπόριο και στον χώρο του καταναλωτή. Οι προσπάθειες της στρέφονται κυρίως γύρω από απτά προϊόντα και υπηρεσίες, ενώ μόλις πρόσφατα άρχισε να αναφέρεται στο λογισμικό σαν υπηρεσία.

ΚΕΦΑΛΑΙΟ 8ο

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΤΩΝ ΣΥΝΕΡΓΑΖΟΜΕΝΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΗΡΕΣΙΩΝ

8 ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΤΩΝ ΣΥΝΕΡΓΑΖΟΜΕΝΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΗΡΕΣΙΩΝ

Στόχος του κεφαλαίου αυτού αποτελεί η εισαγωγή του αναγνώστη σε πρακτικότερα θέματα ανάπτυξης βοηθώντας τον να κατανοήσει βαθύτερα πλέον πως δημιουργούνται αυτές οι υπηρεσίες και στη συνέχεια πως συνεργάζονται μεταξύ τους. Επίσης αποτελεί ένα σημαντικό εισαγωγικό βοήθημα για την κατανόηση του κώδικα από τον οποίο αποτελείται η εφαρμογή που ακολουθεί.

Μεταξύ των τεχνολογικών προτάσεων που αναφέρθηκαν σε προηγούμενο κεφάλαιο εντοπίσαμε ορισμένες εγγενείς αδυναμίες, περιορισμούς ή και πλεονεκτήματα αντίστοιχα για κάθε μια από αυτές ([25]). Συνυπολογίζοντας όλες αυτές τις παραμέτρους και έχοντας σαν στόχο την επιλογή της καταλληλότερης για την πρακτική εφαρμογή του μοντέλου των ηλεκτρονικών υπηρεσιών καταλήξαμε σε αυτή του e-speak. Η συγκεκριμένη πλατφόρμα καλύπτει σε μεγάλο βαθμό τις ιδιαιτερότητες του μοντέλου ενώ παρέχει και την δυνατότητα ανάπτυξης εφαρμογών σε τοπικό επίπεδο και με διαθέσιμα μέσα (υλικό, λογισμικό, υπολογιστική ισχύς).

Πριν όμως αναφερθούμε στον προγραμματισμό των υπηρεσιών κρίνουμε σκόπιμη την αναφορά σε ορισμένες έννοιες που αφορούν το e-speak βοηθώντας στην μετέπειτα κατανόηση.

8.1 Μεσολάβηση και πρόσβαση.

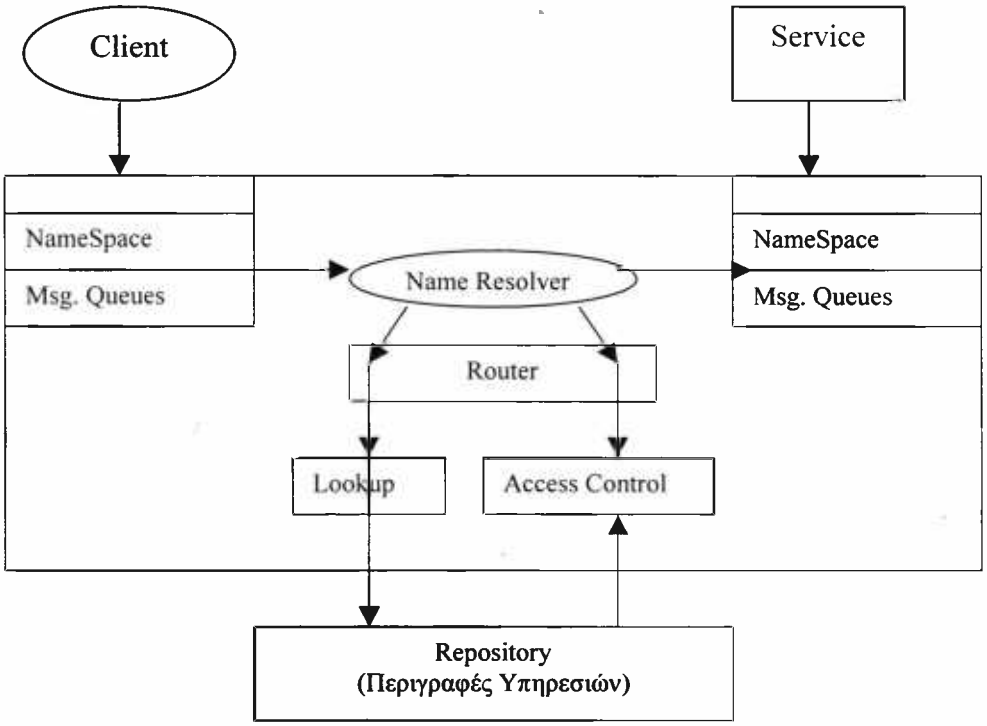
Για την επιτυχία μιας δυναμικής αγοράς ηλεκτρονικών υπηρεσιών οι μηχανισμοί εντοπισμού των υπηρεσιών αυτών απαιτείται να είναι πιο γενικοί από μια απλή αναζήτηση. Το e-speak το επιτυγχάνει αυτό διαφοροποιώντας την περιγραφή της υπηρεσίας από την λειτουργική διεπαφή της. Η περιγραφή της υπηρεσίας πραγματοποιείται μέσω ενός λεξιλογίου ή ενός σχήματος (XML). Με την περιγραφή αυτή άλλες υπηρεσίες μπορούν πλέον να την ανακαλύψουν, να την χρησιμοποιήσουν και να την επεκτείνουν. Το βασικό που πρέπει να επισημανθεί είναι ότι ο πυρήνας της αρχιτεκτονικής του e-speak παρέχει αλληλεπίδραση μεταξύ χρηστών που βρίσκονται πίσω από firewalls. Αυτό θα αποτελέσει κοινή απαίτηση για ταχύτατη ανάπτυξη δυναμικών υπηρεσιών στο Internet ([14]).

8.2 Σύνοψη του πυρήνα

Το e-speak παρέχει την υποδομή πάνω στην οποία οι υπηρεσίες μπορούν να δημιουργηθούν, ανακαλυφθούν και να επικοινωνήσουν. Ο πυρήνας αποτελεί ένα λεπτό στρώμα λογισμικού που επικάθεται στην κορυφή του προϋπάρχοντος λειτουργικού συστήματος και παρέχει τα ακόλουθα:

- Ανακάλυψη
- Δρομολόγηση
- Ασφάλεια
- Ονομασία
- Παρακολούθηση

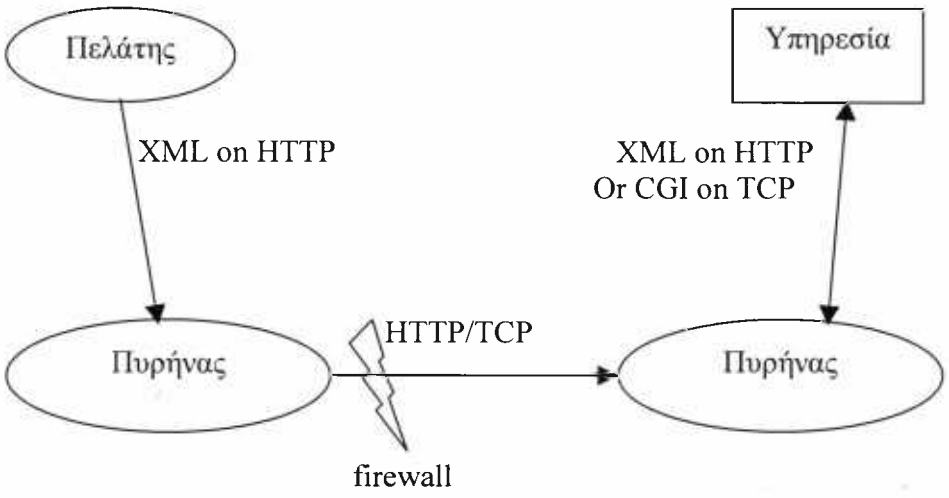
Στο παρακάτω σχήμα περιγράφονται τα βασικά στοιχεία του πυρήνα. Αυτός περιέχει περιοχές ασφαλείας (protection domains) για κάθε χρήστη. Οι χρήστες μπορούν να έχουν



Σχήμα 17

πρόσβαση στην περιοχή τους και εκεί να ανακαλύψουν υπάρχουσες υπηρεσίες ή να δηλώσουν τις δικές τους. Κάθε περιοχή ασφαλείας παρέχει στους χρήστες μια ομάδα από ουρές μηνυμάτων (Message Queues) και ένα τοπικό ιεραρχικό κατάλογο (namespace) όπου μπορούν να αποθηκεύσουν προσωρινά κάποιες συνδέσεις σε υπηρεσίες που έχουν ανακαλύψει. Οι χρήστες χρησιμοποιούν συμβολικά τοπικά ονόματα για να προσδιορίσουν τις υπηρεσίες προσπαθώντας να επιτύχουν μια χαλαρή σύνδεση. Ένας name-resolver μεταφράζει κάθε όνομα σε μια καθολική διεύθυνση. Η διεύθυνση αυτή χρησιμοποιείται για την παράδοση του μηνύματος στον αποδέκτη. Όλες οι περιγραφές των υπηρεσιών βρίσκονται αποθηκευμένες στο repository. Η καταγραφή των υπηρεσιών στον πυρήνα διαφημίζεται σε όλους τους πυρήνες με τους οποίους ο συγκεκριμένος έχει επικοινωνία. Οι κατασκευαστές των υπηρεσιών μπορούν να ορίσουν σε ποιους πυρήνες επιθυμούν να διαφημίζονται οι υπηρεσίες τους δίνοντας κατ' αυτόν τον τρόπο άμεσο έλεγχο της πρόσβασης. Οι χρήστες των υπηρεσιών συνήθως δεν γνωρίζουν που είναι φυσικά εγκατεστημένες. Το e-speak μπορεί να δρομολογήσει μηνύματα μεταξύ πυρήνων από και προς τις αντίστοιχες ουρές μηνυμάτων τους.

Στο παρακάτω σχήμα παρουσιάζονται τα πρωτόκολλα που χρησιμοποιούνται κατά τις αλληλεπιδράσεις. Οι χρήστες μπορούν να επικοινωνήσουν με τον πυρήνα στέλνοντας XML έγγραφα μέσω HTTP ή χρησιμοποιώντας το JESI (όπως θα δούμε στην συνέχεια) μέσω TCP ή HTTP αντίστοιχα. Η τεχνική του tunneling δίνει την δυνατότητα διαπέρασης των firewalls και την επικοινωνία πυρήνων πίσω από αυτά



Σχήμα 18

8.3 Τι είναι οι υπηρεσίες στο e-speak

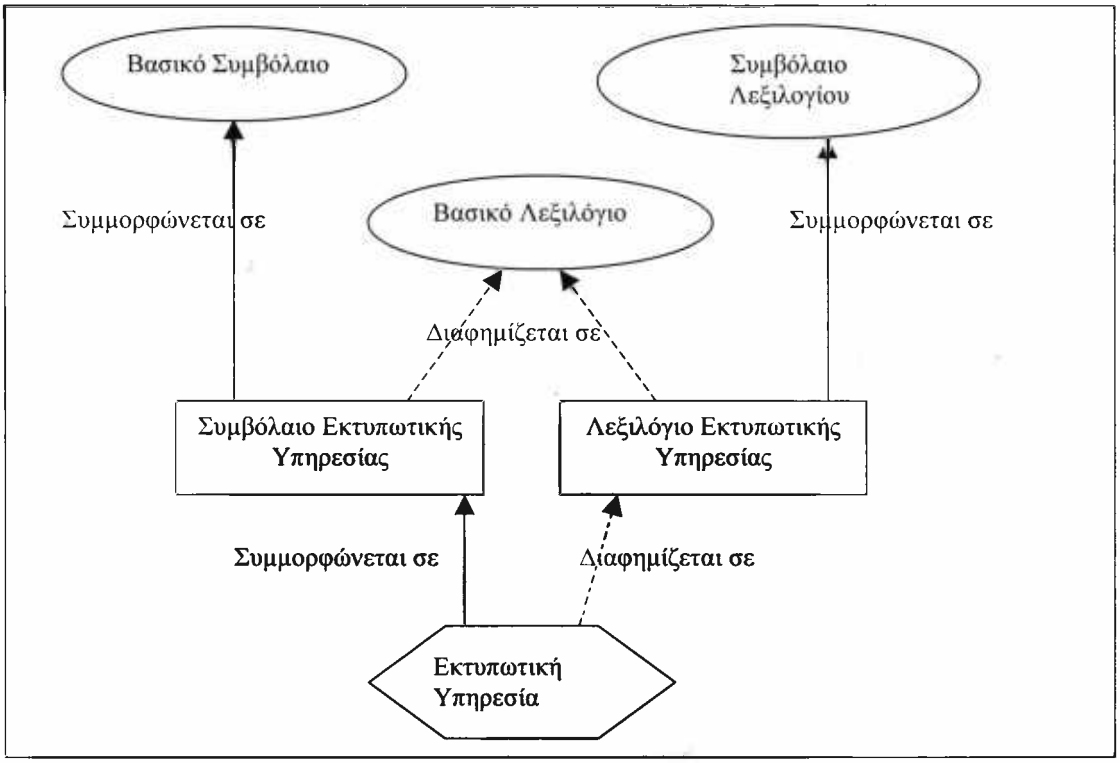
Όλες οι υπηρεσίες χαρακτηρίζονται από τον τύπο της υπηρεσίας που προσφέρουν. Παραδοσιακά ο τύπος των υπηρεσιών ορίζονταν μέσω του τύπου των αντικειμένων που υλοποιούσαν την υπηρεσία. Αυτό καθίσταται ιδιαίτερα περιοριστικό καθώς δεν δίνεται έμφαση σ' αυτό που η υπηρεσία πραγματικά προσφέρει. Στο e-speak γίνεται πλήρης διαχωρισμός ανάμεσα στην περιγραφή της υπηρεσίας και στο κομμάτι που την υλοποιεί. Έτσι όλες οι υπηρεσίες γίνονται γνωστές μέσω ενός γνωστού λεξιλογίου. Η διαφήμιση αυτή έχει την μορφή *ιδιότητα=τιμή* και αποθηκεύεται στο repository που είναι συνδεδεμένο με τον πυρήνα. Από την άλλη ένα συμβόλαιο (contract) χρησιμοποιείται για τον προσδιορισμό της διεπαφής της υπηρεσίας και κατ' επέκταση της πραγματικής λειτουργικότητας της. Μια προγραμματιστική οντότητα σχετίζεται με την προαναφερθείσα διεπαφή και την υλοποιεί στην ουσία της.

Όλες οι υπηρεσίες οφείλουν να "υπακούουν" στις εξής σχέσεις:

1. **Σχέση συμμόρφωσης:** Η σχέση αυτή καθορίζει το συμβόλαιο που πρέπει να υποστηρίζει η υπηρεσία.
2. **Σχέση διαφήμισης:** Η σχέση αυτή καθορίζει το λεξιλόγιο που περιγράφει την προσφερόμενη υπηρεσία

Όπως φαίνεται και στο ακόλουθο σχήμα η υπηρεσία εκτύπωσης για παράδειγμα ικανοποιεί και τις δύο σχέσεις. Συμμορφώνεται με το συμβόλαιο εκτύπωσης αλλά ταυτόχρονα διαφημίζεται και στο σχετικό λεξικό. Το συμβόλαιο της εκτυπωτικής υπηρεσίας περιγράφει όλη την λειτουργικότητα της υπηρεσίας καθώς το λεξικό ταυτόχρονα περιγράφει τα χαρακτηριστικά της υπηρεσίας. Τόσο τα λεξιλόγια όσο και τα συμβόλαια αποτελούν υπηρεσίες από μόνα τους, γι΄ αυτό το σκοπό διαφημίζονται σε άλλα ανώτερα λεξιλόγια ή συμμορφώνονται σε άλλα ανώτερα συμβόλαια αντίστοιχα. Στο συγκεκριμένο παράδειγμα της εκτυπωτικής υπηρεσίας το λεξιλόγιο της εκτύπωσης διαφημίζεται στο βασικό λεξιλόγιο ενώ συμμορφώνεται σε ένα λεξιλόγιο συμβολαίου το

οποίο υποστηρίζεται από τον πυρήνα. Η αναδρομική αυτή σχέση τερματίζεται με την χρήση του βασικού λεξιλογίου και του βασικού συμβολαίου που και τα δύο υποστηρίζονται από τον πυρήνα. Το βασικό λεξιλόγιο αποτελείται από ένα σύνολο καλά ορισμένων ιδιοτήτων, όπου όλοι οι πυρήνες οφείλουν να γνωρίζουν. Όλοι οι χρήστες των υπηρεσιών έχουν πρόσβαση στις υπηρεσίες αυτές μέσω της σύνδεσης που πραγματοποιούν με τον πυρήνα , ενώ δεν χρειάζεται να διαφημίσουν μέσω της παραπάνω διαδικασία την ύπαρξη τους.



Σχήμα 19

8.4 Βασικές και επεκτεινόμενες υπηρεσίες

Παράλληλα με τις βασικές αφηρημένες έννοιες του Λεξιλογίου, του Συμβολαίου και της Υπηρεσίας ο πυρήνας του e-speak υποστηρίζει ένα σύνολο από βασικές και επεκτεινόμενες υπηρεσίες, οι οποίες κάνουν ευκολότερη την χρήση των υπηρεσιών μέσω του Internet. Οι βασικές υπηρεσίες περιλαμβάνουν διαχείριση σύνδεσης, συμβόλαια, λεξιλόγια και ευρέτες υπηρεσιών. Οι επεκτεινόμενες περιλαμβάνουν κοινότητες, γεγονότα και καταλόγους.

8.4.1 Βασικές Υπηρεσίες

Αυτές είναι οι απαραίτητες για την σύνδεση με την πλατφόρμα του e-speak και στόχο έχουν την δημιουργία ή την ανακάλυψη υπηρεσιών. Πιο συγκεκριμένα:

- **Σύνδεση:** Η υπηρεσία σύνδεσης χρησιμοποιείται για την σύνδεση και την αποσύνδεση από την υποδομή του e-speak.
- **Λεξιλόγιο:** Η υπηρεσία λεξιλογίου επιτρέπει την δημιουργία νέων λεξιλογίων καθώς επίσης και τον ορισμό ερωτημάτων προς ήδη υπάρχοντα λεξιλόγια.
- **Συμβόλαιο:** Η υπηρεσία συμβολαίου επιτρέπει την χρήση και την δημιουργία συμβολαίων

- **Ευρέτες:** Με τους ευρέτες ανακαλύπτονται οι υπηρεσίες που έχουν καταχωρηθεί στον πυρήνα.

8.4.2 Επεκτεινόμενες υπηρεσίες

- **Κοινότητα:** Η υπηρεσία αυτή επιτρέπει την ανακάλυψη των υπηρεσιών στα πλαίσια ενός ευρύτερου σχηματισμού πυρήνων που συνδέονται μεταξύ τους. Οι κοινότητες αποτελούν λογικές αφαιρέσεις ομάδων που διαμορφώνουν ένα χώρο αναζήτησης υπηρεσιών
- **Γεγονότα:** Αυτή η υπηρεσία καθορίζει ένα πλαίσιο καταχώρησης - δημοσιοποίησης γεγονότων που πραγματοποιούνται στα πλαίσια της κοινότητας. Για παράδειγμα γεγονός αποτελεί η είσοδος μια νέας υπηρεσίας στην κοινότητα για την οποία πρέπει να λάβει γνώση η κοινότητα.
- **Κατάλογοι :** Η υπηρεσία καταλόγου δίνει την δυνατότητα στους χρήστες της να διαχειριστούν τις υπηρεσίες τους (πιθανώς απομακρυσμένες) σαν να πρόκειται για απλή διαχείριση αρχείων στο υπάρχον λειτουργικό σύστημα. Μόνιμοι κατάλογοι διατηρούν την σύνδεση που έχει πραγματοποιήσει ο χρήστης με την μορφή bookmarks.

8.5 Η σημασία του λεξιλογίου

Το λεξιλόγιο αποτελεί σημαντικό στοιχείο στην αρχιτεκτονική διάρθρωση του e-speak. Δύο είναι τα βασικά στοιχεία που εμπλέκονται στην περιγραφή του λεξιλογίου ([22]). Το πρώτο εξαιτίας του ότι το λεξιλόγιο αποτελεί υπηρεσία που μπορεί να ανακαλυφθεί τόσο από τους πελάτες όσο και από τους παροχείς υπηρεσιών. Κατά δεύτερο οι ιδιότητες του λεξιλογίου που περιγράφονται σε αυτό πρέπει να οριστούν επαρκώς έτσι ώστε κάθε υπηρεσία που χρησιμοποιεί ένα συγκεκριμένο λεξιλόγιο να έχει ακριβώς τις ίδιες ιδιότητες.

Τα λεξιλόγια μπορεί να θεωρηθούν ως κάτι ανάλογο με το σχήμα ενός πίνακα σε μια σχεσιακή βάση δεδομένων. Οι ιδιότητες του λεξιλογίου βρίσκονται σε ευθεία αναλογία με τις στήλες του πίνακα, ενώ οι υπηρεσίες που χρησιμοποιούν ένα συγκεκριμένο λεξιλόγιο μπορούν κάλλιστα ν θεωρηθούν γραμμές σε πίνακα βάσης. Οι χρήστες που δημιουργούν λεξιλόγια μπορούν να τα ορίσουν χρησιμοποιώντας την μέθοδο addAttribute() της κλάσης ESVocabularyDescription όπως θα δούμε και στην συνέχεια.

Το ακόλουθο παράδειγμα αναφέρεται σε ένα λεξιλόγιο με τρεις βασικές ιδιότητες-γνωρίσματα : Manufacturer, Modelname, DPI.

```
<?xml version="1.0"?>
<ESpeak version="E-Speak 1.0beta" operation="CreateVocab" xmlns="http://
localhost/e:/Esxml/Schemas/espeak.xsd">
<resource xmlns="">
    <resourceDes xmlns="">
      <pattern>
          <Name xmlns="">printervocab</Name>
        <Type>Vocabulary</Type>
      </pattern>
    </resourceDes>
    <resourceData xmlns="">
    <attrGroup name="my printer Vocabulary" xmlns="">
    <attrDecl xmlns="" name="Manufacturer" required="no">
```

```
        <datatypeRef xmlns="" name="string"/>
      </attrDecl>
      <attrDecl xmlns="" name="Modelname" required="no">
        <datatypeRef xmlns="" name="string"/>
      </attrDecl>
      <attrDecl xmlns="" name="DPI" required="no">
        <datatypeRef xmlns="" name="integer"></datatypeRef>
      </attrDecl>
</attrGroup>
</resourceData>
</resource>
</ESpeak>
```

Όλα τα λεξιλόγια που ορίζονται στο e-speak έχουν σαν αρχικό element το <Espeak> όπου προσδιορίζονται διάφορες πληροφορίες έκδοσης και αναφοράς. Για να μπορέσουν οι χρήστες του λεξιλογίου να το εντοπίσουν βασική προϋπόθεση αποτελεί ο ορισμός του. Στην ετικέτα (tag) resourceDes περιγράφεται το όνομα του νέου λεξιλογίου καθώς και ο τύπος. Αμέσως μετά ορίζονται οι ιδιότητες του λεξιλογίου.

8.6 Μοντέλα επικοινωνίας ηλεκτρονικών υπηρεσιών

Για την δημιουργία εφαρμογών στο χώρο του Internet απαραίτητη είναι η απόφαση σε υψηλό σχεδιαστικό επίπεδο για τον τρόπο επικοινωνίας των εφαρμογών μεταξύ τους. Αυτή τη στιγμή υπάρχουν δυο βασικά μοντέλα επικοινωνίας:

- **Network Object Model (NOM):** αποτελεί ένα στενά συνδεδεμένο μεταξύ των εφαρμογών τρόπο επικοινωνίας. Επίσης αποτελεί ένα σύγχρονο τρόπο επικοινωνίας το οποίο μεταφράζεται στο εξής: όταν μια εφαρμογή αποστέλλει ένα μήνυμα σε μια άλλη αναμένει απάντηση προτού προβεί σε νέα αποστολή ή προχωρήσει στην εκτέλεση της.
- **Document Exchange Model (DEM):** αποτελεί ένα ασύγχρονο και χαλαρά συνδεδεμένο μοντέλο επικοινωνίας. Στην συγκεκριμένη περίπτωση μια εφαρμογή αποστέλλει ένα XML αρχείο και κάποια στιγμή στο μέλλον ο αποδέκτης απαντά με ένα αντίστοιχο μήνυμα. Το μοντέλο αυτό είναι περισσότερο αποδοτικό σε B2B εφαρμογές όπου δεν απαιτείται για τον αποστολέα να λάβει άμεσα απάντηση.

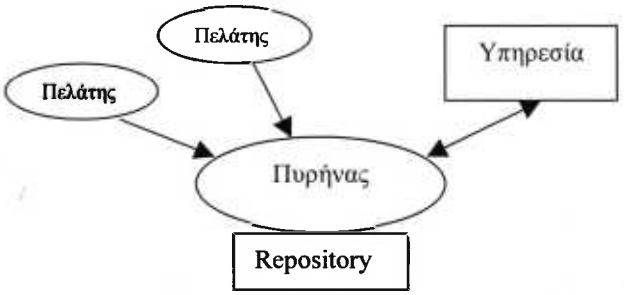
Το e-speak υποστηρίζει και τα δύο μοντέλα που αναφέραμε με τα εξής:

- **NOM:** Μέσω των **J-ESI APIs** (Java E-Service Interface) μπορούν να έρθουν σε επικοινωνία πυρήνες e-speak ή εφαρμογές που βρίσκονται εγκατεστημένες πάνω στους πυρήνες αυτούς. Το συγκεκριμένο είναι ταχύ κατά την εκτέλεση του, ενώ παράλληλα απαιτεί στενή σύζευξη μεταξύ των μερών που αποτελούν την εφαρμογή. Συνίσταται κυρίως για χρήση Intranet. Επίσης για εφαρμογές όπου η σύζευξη εφαρμογών βάσει χρόνου είναι ιδιαίτερα σημαντική.
- **Document Exchange Model (DEM):** Η δεύτερη προσέγγιση καλείται WEB Access, είναι ευκολότερη σε υλοποίηση μιας και χρησιμοποιεί το Web για την ενεργοποίηση και χρήση των υπηρεσιών Μέσω του WEB Access οι εφαρμογές μπορούν απλά να ανταλλάσουν XML αρχεία (όπως συμβαίνει στο SOAP) μέσω του e-speak service bus. Στόχος της είναι οι B2B εφαρμογές, μιας και ένα από τα κύρια χαρακτηριστικά της είναι η δυνατότητα διαπέρασης των Firewalls και η επικοινωνία εφαρμογών πίσω από αυτά.

8.7 Διαφήμιση των υπηρεσιών

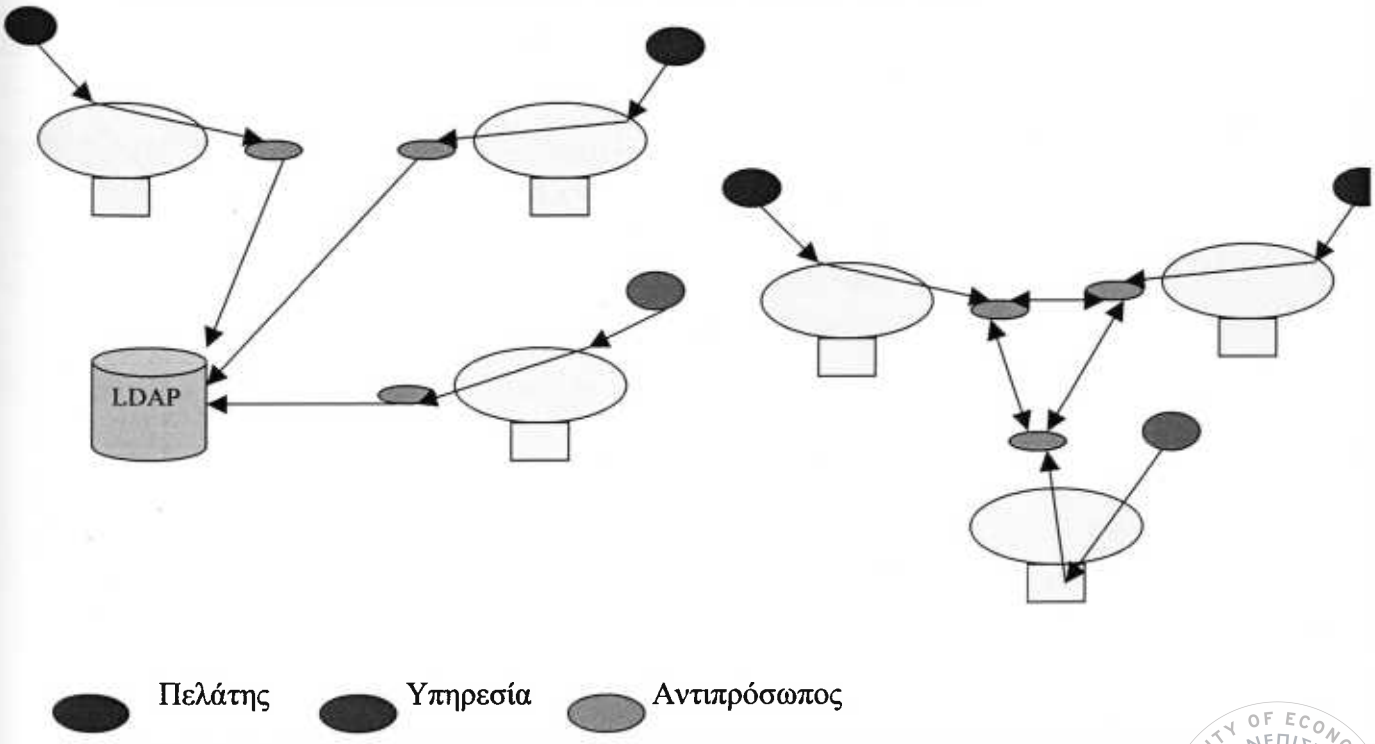
Κάθε υπηρεσία που έχει καταχωρηθεί σε έναν e-speak πυρήνα μπορεί να γίνει ευρύτερα γνωστή και σε άλλους πυρήνες. Οι πιθανότερες περιοχές όπου μπορεί να γίνει γνωστή και κατ΄ επέκταση χρησιμοποιήσιμη είναι:

- **Στον ίδιο πυρήνα:** Αυτό είναι το κλασσικό μοντέλο client-server ανάπτυξης όπου η υπηρεσία είναι ορατή μόνο στους χρήστες που συνδέονται στον συγκεκριμένο πυρήνα. Κάθε πυρήνας έχει στην διάθεση του ένα repository όπου βρίσκονται καταχωρημένες όλες οι περιγραφές των υπηρεσιών ου βρίσκονται καταχωρημένες σε αυτόν.



Σχήμα 20

- **Σε μια ομάδα πυρήνων:** Μια ομάδα πυρήνων είναι ένα repository υπηρεσιών διαμοιραζόμενων σε διαφορετικούς πυρήνες. Η ομάδα χαρακτηρίζεται από ένα κοινό όνομα και αποτελείται από ένα group-server με ένα σύνολο από αντιπροσώπους (proxies) που αναλαμβάνουν την διαφήμιση των υπηρεσιών στον group-server.Η διαφήμιση των υπηρεσιών στον group-server επιτρέπει την ανακάλυψη τους από πελάτες που είναι συνδεδεμένοι σε διαφορετικούς πυρήνες. Οι group-servers είναι ανάλογοι με τους servers τύπου καταλόγου όπως είναι ο LDAP. Οι διαχειριστές των αρχιτεκτονικών μπορούν να χρησιμοποιήσουν για τον σκοπό αυτό έναν LDAP Server ή τους αντιπροσώπους(proxies) επιτυγχάνοντας το ίδιο αποτέλεσμα.



Σχήμα 21

- **Μια κοινότητα e-speak:** Η κοινότητα αποτελεί επέκταση της ομάδας με περισσότερα μέλη με όλα τα πλεονεκτήματα που εντοπίζουμε σε αυτές.

8.8 Προγραμματισμός των υπηρεσιών

Πιο συγκεκριμένα, το χτίσιμο και η ενεργοποίηση μιας υπηρεσίας προϋποθέτουν τρία βασικά βήματα ([16]):

- Προσδιορισμό της διεπαφής της υπηρεσίας
- Συγγραφή του σχετικού κώδικα που αποτελεί την υλοποίηση της προαναφερθείσας διεπαφής
- Ενεργοποίηση της υπηρεσίας

Στο εξής θα χρησιμοποιηθεί ένα απλό παράδειγμα που θα κάνει ευκολότερη την κατανόηση των επεξηγήσεων που παρατίθενται σε κάθε βήμα. Αντικείμενο του παραδείγματος αποτελεί η δημιουργία μιας υπηρεσίας που θα χρησιμοποιεί μια υπάρχουσα εφαρμογή που αναζητά συγκεκριμένα άτομα (όνομα και επίθετο σε μια ιεραρχική δομή μέσα στον LPAP Server). Η νέα υπηρεσία θα χρησιμοποιεί τα αποτελέσματα της αναζήτησης προκειμένου να εξαγάγει τα e-mail τους, σχηματίζοντας μια λίστα αποτελεσμάτων σε XML μορφή. Επίσης θα αναφερθεί ο τρόπος με τον οποίο δημιουργείται μια εφαρμογή πελάτης που καλεί την υπηρεσία πραγματοποιώντας μια αναζήτηση

8.8.1 Προσδιορισμός της Διεπαφής

Η διεπαφή της υπηρεσίας καθορίζεται ως e-speak IDL, το οποίο προγραμματιστικά είναι όμοιο με το JAVA RMI IDL. Το αρχείο IDL πρέπει να έχει την κατάληξη .esidl για τον IDL Compiler. Η εφαρμογή RecordFinder έχει μια μέθοδο την Search_by_Name, η οποία και θα χρησιμοποιηθεί. Στη συνέχεια δημιουργείται ένα αρχείο LookupServiceIntf.esidl που περιέχει μόνο την μέθοδο Search_by_Name:

```
public interface Lookup ServiceIntf {
        public String search_by_Name (String Name) };
```

Το IDL μεταγλωττίζεται χρησιμοποιώντας τον IDL Compiler:

Java net.espeak.util.esidl.IDLCompiler LookupServiceIntf.esidl.

Ο IDL Compiler παράγει τα ακόλουθα αρχεία:

- Lookup ServiceIntf.java
- Lookup ServiceStub.java
- Lookup Service MessageRegistry.java

- Το πρώτο είναι αντίγραφο του Lookup ServiceIntf.java με μικρές διαφοροποιήσεις. Το δεύτερο αποτελεί μια κλάση Stub που η υπηρεσία αναζήτησης επιστρέφει στον πελάτη όταν ανακαλύπτει την Lookup υπηρεσία. Το τελευταίο χρησιμοποιείται προκειμένου το J-ESI να καταγράψει τα αντικείμενα που προκύπτουν.

8.8.2 Συγγραφή του Κώδικα υλοποίησης της Διεπαφής

Μετά τον προσδιορισμό της διεπαφής απαραίτητη είναι η υλοποίηση των μεθόδων που ορίστηκαν σε αυτή. Στην συνέχεια περιγράφεται πως η LookupServiceImpl υλοποιεί την μέθοδο Search_by_Name, η οποία πραγματοποιεί τις ακόλουθες ενέργειες:

- Χρησιμοποιεί το αντικείμενο JavaRecorderFinder για να ελέγξει τις εγγραφές των ατόμων με ένα συγκεκριμένο όνομα από τον LDAP Server.
- Εξάγει τις διευθύνσεις e-mail των αντίστοιχων ατόμων και σχηματίζει την απάντηση σε XML μορφή.
- Επιστρέφει την λίστα με την μορφή String στον πελάτη

```
import java.util.Vector;
import netscape.ldap.LDAPException;
import net.espeak.jesi.ESService;
import net.espeak.infra.cci.exception.ESInvocationException;

/* Υλοποίηση της μεθόδου "searchByName" */
public class LookupServiceImpl implements LookupServiceIntf {
public String searchByName(String name) throws ESInvocationException {
Vector list;
String emails = "";
String results = "";

/* Δημιουργεί ένα στιγμυότυπο του RecordFinder, πραγματοποιεί μια αναζήτηση
με βάσει το όνομα και επιστρέφει το αποτέλεσμα υπό μορφή XML */
RecordFinder finder = new RecordFinder();
try {
results += "<list name=\"" + name + "\">\n";
list = finder.searchByName(name);
int size = list.size();
if (size > 0) {
Person p;
for (int i = 0; i < size; ++i) {
p = (Person)list.elementAt(i);
emails += (" <email>" + p.getEmail().trim() + "</email>\n");
}
}
else
emails += " <empty/>\n";
results = "<list name=\"" + name + "\">\n" + emails + "</list>\n";
} catch (LDAPException e) {
System.out.println("ERROR >> " + e.getMessage());
e.printStackTrace();
results += "<error>";
results += e.getMessage();
results += "</error>";
throw(new ESInvocationException(results));
}
System.out.println("Search for " + name + " is completed.");
return results;
```

```
}
}
```

Η υλοποίηση του Interface δεν περιέχει καθόλου όπως παρατηρήθηκε εξειδικευμένο e-speak κώδικα. Παρόμοιες ή διάφορες υπηρεσίες μπορούν να αναπτυχθούν, πρώτα με τον προσδιορισμό των διεπαφών και ακολούθως με την συγγραφή της υλοποίησης.

8.8.3 Ενεργοποίηση της Υπηρεσίας

Στη φάση αυτή γίνεται η περιγραφή της υπηρεσίας σε ένα επιλεχθέν λεξικό καθώς επίσης η καταγραφή και η ενεργοποίηση της υπηρεσίας , έτσι ώστε οι πελάτες να μπορούν να τη ανακαλύψουν και να την χρησιμοποιήσουν.
Βασικά βήματα σε αυτή την διαδικασία είναι τα εξής:

1. Δημιουργία σύνδεσης με τον πυρήνα
2. Δημιουργία της περιγραφής της υπηρεσίας
3. Δημιουργία της υλοποιημένης διεπαφής
4. Καταγραφή, Διαφήμιση και έναρξη της υπηρεσίας
5. Αναμονή για λήψη ερωτήσεων και άμεση απάντηση

```
import net.espeak.jesi.*;
import net.espeak.infra.cci.exception.*;
import java.lang.reflect.Array;
public class LookupServer
{
public static void main(String[] args) {

/* 'Ελεγχος παραμέτρων εισόδου*/
int argSize = Array.getLength(args);
if (argSize != 1) {
System.out.println("Usage: java LookUpServer "+ "<property file>");
System.exit(1);
}
try {

/*  Προσπάθεια σύνδεσης στον πυρήνα βάσει των  παραμέτρων εισόδου */
ESConnection connection = new ESConnection(args[0]);

/*  Δημιουργία περιγραφής της υπηρεσίας βάσει λεξιλογίου */
ESServiceDescription desc = new ESServiceDescription();
desc.addAttribute("Name", "LookupService");
desc.addAttribute("Description", "E-mail lookup");

/* Δημιουργία ενός αντικειμένου της υπηρεσίας και έναρξη υλοποίησης της */
ESServiceElement element = new ESServiceElement(connection, desc);
element.setImplementation(new LookupServiceImpl());

/* Καταχώρηση, διαφήμιση και έναρξη της υπηρεσίας */
```

```
element.register();
element.advertise();
element.start();
System.out.println("LookupService is Ready . . .");
}
catch (Exception e) {
e.printStackTrace();
}
}
}
```

8.8.4 Δημιουργία Διασύνδεσης με τον πυρήνα e-speak

Στο J-ESI API, το κανάλι στο οποίο γίνεται η επικοινωνία μιας εφαρμογής με το e-speak είναι το ESConnection. Ένα στιγμιότυπο της σύνδεσης γίνεται μέσω της κλήσης:

ESConnection.connection = new ESConnection (argc [0]);

Οι πελάτες μιας εφαρμογής εισάγουν τα απαραίτητα στοιχεία τα οποία απαιτούνται για τη διασύνδεσή τους με το e-speak σε ένα configuration File, το οποίο παρέχεται σαν όρισμα στη γραμμή εντολών, στον constructor του ESConnection. Τα στοιχειώδη που μπορεί να περιέχει το αρχείο αυτό είναι τα ακόλουθα:

- AccountName (Guest Connection)
- UserName (guest)
- Password (null)
- Protocol (tcp)
- HostName (LocalHost)
- Postnumber (12346)
- Community (null)
- EventControl (0)
- Esurl (tcp: localhost: 12346);

Για παράδειγμα, αν ο πυρήνας «τρέχει» στην τοπική μηχανή, το αντίστοιχο property αρχείο θα διαμορφωθεί ως εξής:

HostName = LocalHost
PostNumber = 23456

Όταν ένα Property αρχείο χρησιμοποιείται, τότε παρατηρούνται τα ακόλουθα:

- Όταν το accountname δεν προσδιορίζεται, δίνεται ένας προσωρινός αριθμός επισκέπτη ο οποίος χάνεται με τη λήξη της σύνδεσης.
- Όταν ο αριθμός υπάρχει ήδη και οι πιστοποιήσεις του χρήστη ελεγχθούν, το κομμάτι του πυρήνα που ασχολείται με τις συνδέσεις επιστρέφει μια σύνδεση στην οποία περιγράφονται τα τελευταία έγκυρα πιστοποιημένα στοιχεία του χρήστη.

8.8.5 Δημιουργία περιγραφής για την Υπηρεσία

Χρησιμοποιείται το αντικείμενο ESService Description για να προσδιοριστεί το λεξικό για την υπηρεσία, προσθέτοντας του τις τιμές των γνωρισμάτων που την περιγράφουν.

ESServiceDescription desc = new ESServiceDescription();
desc.addAttribute("Name", "LookupService");

Στην συγκεκριμένη περίπτωση το γνώρισμα "Name" έχει τη τιμή "LookupService". Η πληροφορία αυτή συμβάλλει στην καταχώρηση της υπηρεσίας και την εύρεση της από τους πελάτες. Ο πυρήνας της αρχιτεκτονικής έχει ενσωματωμένο ένα βασικό λεξικό. Οι δημιουργοί νέων υπηρεσιών έχουν τη δυνατότητα να χρησιμοποιήσουν το λεξικό αυτό ή να προχωρήσουν στην κατασκευή νέων. Ο δημιουργός του λεξικού είναι υπεύθυνος για την «προβολή» της ύπαρξης του λεξικού του στους ενδεχόμενους χρήστες.

8.8.6 Δημιουργία της υλοποιημένης διεπαφής

Στην συνέχεια δημιουργείται ένα νέο αντικείμενο ESServiceElement, μεταφέροντας στον δημιουργό του (Constructor) το αντικείμενο της σύνδεσης και της περιγραφής της υπηρεσίας:

ESServiceElement element = new ESServiceElement (connection, desc);
Η υλοποίηση αυτής της διεπαφής για την υπηρεσία γίνεται με την ενεργοποίηση της μεθόδου SetImplementation με όρισμα ένα στιγμιότυπο της LookupServiceImpl
Element.setImplementation(new LookupServiceImpl());

8.8.7 Καταγραφή, Διαφήμιση και έναρξη της υπηρεσίας

Η μέθοδος Register του EEServiceElement, είναι αυτή η οποία καταγράφει την υπηρεσία στο Repository της αρχιτεκτονικής. Η διαφήμιση της υπηρεσίας ενεργοποιείται από την advertise. Αυτό είναι απαραίτητο ώστε «γειτονικές» υπηρεσίες ευρισκόμενες σε γειτονικούς πυρήνες να μπορέσουν να ανιχνεύσουν τη νεοεισελθείσα υπηρεσία. Τέλος, μέσω της start(), η υπηρεσία δηλώνει την ετοιμότητα της να εξυπηρετήσει πελάτες:

element.register();
element.advertise();
element.start();

8.8.8 Πρόσβαση στην Υπηρεσία

Ο πελάτης ο οποίος επιθυμεί να αποκτήσει πρόσβαση σε μια υπηρεσία πρέπει να:

- Δημιουργήσει μια σύνδεση με τον πυρήνα
- Ανακαλύψει την υπηρεσία μέσω της κλάσης EEServiceFinder
- Ενεργοποιήσει την υπηρεσία μέσω της διεπαφής της

Ο πελάτης αποκτά ένα stub στην συγκεκριμένη υπηρεσία αμέσως μετά την εύρεση της και μέσω αυτού, αποκτά πρόσβαση σε αυτήν.
Το ακόλουθο παράδειγμα αναφέρει διεξοδικά τα παραπάνω:

```
import net.espeak.jesi.*;
import net.espeak.infra.cci.exception.*;
import java.lang.reflect.Array;

public class LookupClient
{
private String propFile;
public LookupClient(String pFile) {
this.propFile = pFile;
}
public String search(String name) {
String result = "";

try {
/* Προσπάθεια σύνδεσης στον πυρήνα χρησιμοποιώντας παραμέτρους που
περιγράφονται στο αρχείο. */
ESConnection connection = new ESConnection(propFile);

/*Προσδιορισμός της διεπαφής που θα χρησιμοποιηθεί. */
String intfName = LookupServiceIntf.class.getName();

/* Δημιουργία στυγμιοτύπου της ESServiceFinder για την αναζήτηση της
διεπαφής */
ESServiceFinder finder = new ESServiceFinder(connection, intfName);

/* Προσδιορισμός των ιδιοτήτων του λεξιλογίου που θα αναζητηθεί. */
ESQuery queryString = new ESQuery("Name == 'LookupService'");

/* Εύρεση της υπηρεσίας που ικανοποιεί τα κριτήρια της αναζήτησης και
επιστροφή του stub για την χρησιμοποίηση των μεθόδων. */
LookupServiceIntf lookupService =
(LookupServiceIntf)finder.find(queryString);

/* Πραγματοποίηση αναζήτησης ενός συγκεκριμένου ονόματος */
System.out.println("Searching for: " + name + "\n");
result = lookupService.searchByName(name);
}
catch (Exception e) {
result = e.getMessage();
}
return result;
}
public static void main(String[] args) {
String name = null;

/* Έλεγχος παραμέτρων εισόδου*/
int argSize = Array.getLength(args);
if (argSize == 3) {
name = args[1] + " " + args[2];
```

```
}
else {
System.out.println("Usage: java LookUpClient <property file> "
+ "<first name> <last name>");
System.exit(1);
}

/* Πραγματοποίηση αναζήτησης και επιστροφή αποτελεσμάτων */
LookupClient lc = new LookupClient(args[0]);
String result = lc.search(name);
System.out.println(result);
}
}
```

8.8.9 Δημιουργία Σύνδεσης

Η σύνδεση στην πλευρά του πελάτη δημιουργείται με την εντολή:

ESConnection connection = new ESConnection (propfile);

Το Configuration File το οποίο αναφέρθηκε προηγουμένως, μπορεί να χρησιμοποιηθεί και στον Server και στον Client, από τη στιγμή που και οι δύο εφαρμογές είναι συνδεδεμένες στον ίδιο πυρήνα. Αν ο Client βρισκόταν σε διαφορετικό πυρήνα, τότε θα ήταν απαραίτητο ένα διαφορετικό configuration file, στο οποίο θα περιγραφόταν η κοινότητα στην οποία ανήκει.

8.9 Εύρεση της Υπηρεσίας

Ο Πελάτης χρησιμοποιεί το όνομα της διεπαφής της υπηρεσίας και με αυτόν τον τρόπο δημιουργεί ένα στιγμιότυπο της υπηρεσίας Finder:

```
String intfName = LookupServiceIntf.class.getName();
ESServiceFinder Finder = new ESServiceFinder(connection,intfName);
```

Η κλάση ESServiceFinder χρησιμοποιεί την ESQuery για να προσδιορίσει τους περιορισμούς της αναζήτησης. Έτσι αν ένας πελάτης αναζητά μια υπηρεσία που έχει διαφημιστεί έχουσα το γνώρισμα "Name" = "LookupService" τότε τα ακόλουθα μπορούν να χρησιμοποιηθούν:

```
ESQuery queryString = new ESQuery("Name == 'LookupService'");
LookupServiceIntf lookupService =(LookupServiceIntf)finder.find(queryString);
```

Οι πελάτες ανακαλύπτουν τις υπηρεσίες μέσω των λεξικών. Η επιστρεφόμενη τιμή της μεθόδου find επιστρέφει ένα stub για τον πελάτη της υπηρεσίας, σε μια διεπαφή της υπηρεσίας αυτής. Στην συνέχεια, μετά την εύρεση της υπηρεσίας ο πελάτης μπορεί να χρησιμοποιήσει τις μεθόδους της μέσω της διεπαφής της. Στο παράδειγμα μας η LookupClient χρησιμοποιεί την μέθοδο searchByName της υπηρεσίας LookupService προκειμένου να αναζητήσει τα στοιχεία ενός συγκεκριμένου ατόμου.

```
result = lookupService.searchByName(name);
```

ΚΕΦΑΛΑΙΟ 9ο

ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

9 ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Το σκέλος αυτό της διπλωματικής αποτελεί ένα έμπρακτο δείγμα του μοντέλου των υπηρεσιών, τηρουμένων πάντα των αναλογιών. Περιγράφει με αρκετή προσέγγιση ένα νέο τρόπο λειτουργίας των υπαρχόντων χρηματοοικονομικών μοντέλων μέσα από το μοντέλο των υπηρεσιών. Πιο συγκεκριμένα, η εφαρμογή αναφέρεται σε ένα χρηματιστηριακό παράδειγμα, όπου εμπλέκονται όλοι οι πραγματικοί φορείς. Συμμετοχή έχουν οι πελάτες, οι χρηματιστές, οι τράπεζες και η έμπιστη τρίτη οντότητα, για παράδειγμα η χρηματαγορά (Χ.Α.Α), όπου καταγράφονται όλες οι συναλλαγές που λαμβάνουν χώρα μεταξύ των προαναφερόμενων μελών. Το πλεονέκτημα της εφαρμογής είναι ότι μπορούμε να την κάνουμε όσο απλή ή όσο πολύπλοκη επιθυμούμε. Έχουμε δηλαδή τη δυνατότητα αφενός να χρησιμοποιήσουμε ένα ελάχιστο σύνολο ρόλων ώστε να υλοποιηθεί μια συναλλαγή (δύο πελάτες, μια τράπεζα, ένας χρηματιστής), έως πολύπλοκες δομές πολλών πελατών, πολλών τραπεζών, πολλών χρηματιστών, εξαρτώμενοι πάντα από τους υπολογιστικούς πόρους που είναι διαθέσιμοι.

Όλες οι οντότητες που συμμετέχουν έχουν την μορφή υπηρεσίας. Κοινό υπόβαθρο αποτελεί η πλατφόρμα του e-speak, η οποία ως γνωστόν επιτρέπει την καταχώρηση νέων υπηρεσιών, την αναζήτηση-ανακάλυψη, την διαπραγμάτευση και την σύνθεση των υπηρεσιών. Στην παρούσα φάση υποστηρίζονται όλες οι λειτουργίες εκτός από αυτή της σύνθεσης, καθώς δεν υποστηρίζεται από την τρέχουσα έκδοση της πλατφόρμας του e-speak.

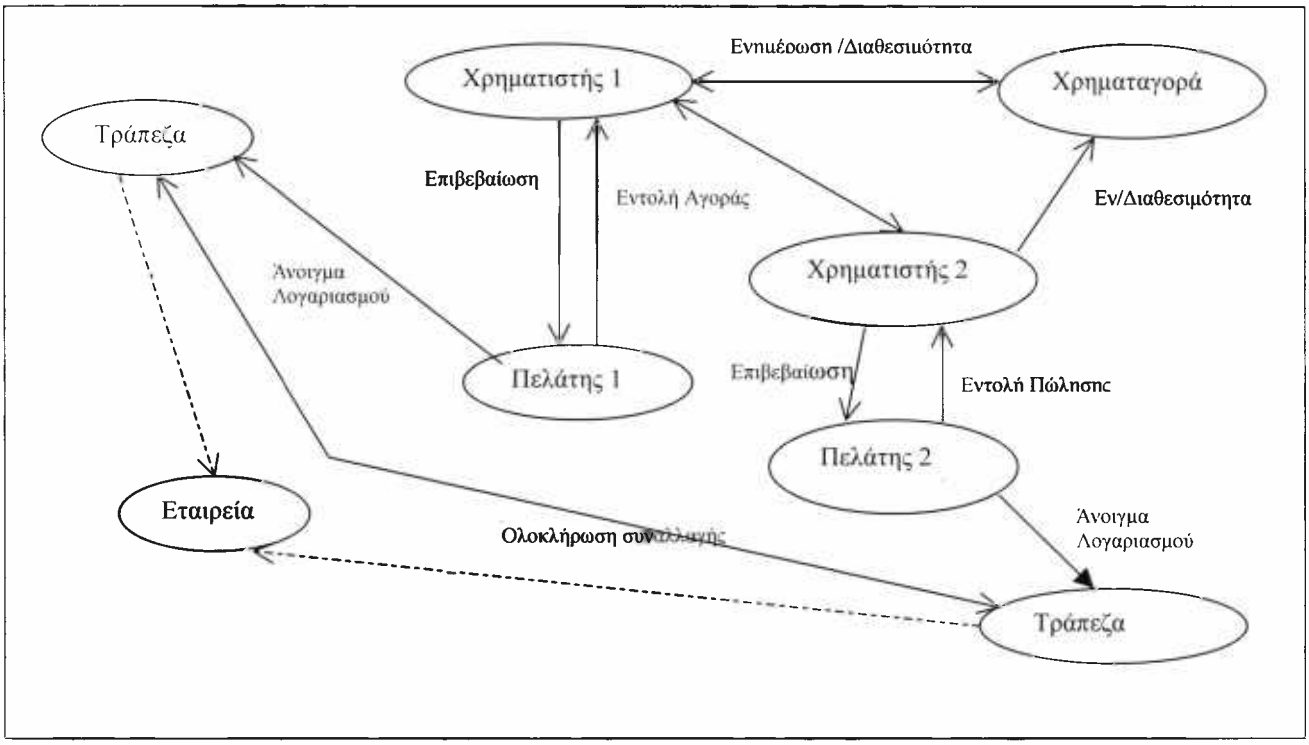
9.1 Περιγραφή της εφαρμογής

Τα βασικά συστατικά της εφαρμογής αποτελούν οι συμμετέχουσες οντότητες, αλλά και οι σχέσεις που διαμορφώνονται μεταξύ τους. Συνοπτικά, ως γνωστόν ο πελάτης είναι αυτός που επιδιώκει να συνδιαλλαγεί με άλλους πελάτες προβαίνοντας με αυτόν τον τρόπο σε συναλλαγές. Άμεση είναι η σχέση του με κάποιο τραπεζικό ίδρυμα - την Τράπεζα- η οποία είναι υπεύθυνη για την αποδοχή ή αποστολή χρημάτων σε άλλες τράπεζες ή σε λογαριασμούς της ίδιας, προκειμένου να εισπραχθούν/αποδοθούν τα ποσά από την αγορά ή πώληση μετοχών. Στην τράπεζα κατατίθενται επίσης και χρηματικά ποσά των πελατών, προκειμένου να χρησιμοποιηθούν ως έναντι για την αγορά-προμήθεια νέων μετοχών. Επίσης, το ποσό το οποίο προκύπτει από την πώληση μετοχών του σχετικού καταθέτη, πιστώνεται αυτόματα στον λογαριασμό του.

Ο χρηματιστής είναι μια άλλη οντότητα, εξίσου σημαντική. Έρχεται σε επαφή με τους πελάτες και στο εξής δρα ως πληρεξούσιος τους. Ενημερώνεται από την χρηματαγορά για τις διαθέσιμες προς διαπραγμάτευση μετοχές, αναλαμβάνοντας εν συνεχεία την εκπλήρωση της όποιας συναλλαγής επιλέξει ο πελάτης του. Η οντότητα-υπηρεσία "Χρηματιστής" δέχεται εντολές ώστε να πουλήσει ή να αγοράσει μετοχές, κλείνοντας επίσης την συμφωνία με την τράπεζα. Έντονη είναι η σχέση του χρηματιστή με την οντότητα χρηματαγορά, από όπου πληροφορείται τις κινήσεις άλλων αντίστοιχων υπηρεσιών- χρηματιστών.

Με σκοπό την όσο το δυνατόν μεγαλύτερη ρεαλιστικότητα της εφαρμογής, ενεργοποιούμε δύο υπηρεσίες-χρηματιστές. Αντίστοιχοι πελάτες χρησιμοποιούν τις υπηρεσίες των χρηματιστών και πραγματοποιούνται συναλλαγές. Τα αποτελέσματα

κατευθύνονται στους λογαριασμούς των πελατών, ή απευθείας στην τράπεζα. Σχηματικά έχουμε:



Σχήμα 22

Το ενδιαφέρον είναι ότι οι πελάτες, είτε αυτοί είναι αγοραστές, είτε πωλητές, συναλλάσσονται μέσω των χρηματιστών τους όπως και υπό πραγματικές συνθήκες. Θεωρητικά, όλες οι εταιρίες των οποίων οι μετοχές διαπραγματεύονται στη χρηματιστηριακή αγορά, θα μπορούσαν μέσω των τραπεζών να πληροφορηθούν για το ύψος του πακέτου μετοχών το οποίο κατέχει κάθε μέτοχος.
Στη συνέχεια, θα περιγράψουμε ένα πλήρες σενάριο, όπου αναφέρονται όλες οι δυνατές συσχετίσεις μεταξύ των υπηρεσιών.

θεωρούμε την ύπαρξη δύο πελατών που εκφράζουν την επιθυμία συναλλαγής ενεργοποιώντας την αντίστοιχη-υπηρεσία πελάτη. Μετά την ενεργοποίηση της γίνεται άμεσα η επιλογή τράπεζας και χρηματιστή από τους διαθέσιμους. Η διαθεσιμότητα τράπεζας/ων ή του χρηματιστή/ων, γίνεται μέσα από την περιγραφή των ιδιοτήτων τους μέσω XML και την σύνδεση τους στην υποκείμενη πλατφόρμα του e-speak. Στην συνέχεια ο πελάτης (δύο στη συγκεκριμένη περίπτωση), ανοίγει έναν λογαριασμό στην τράπεζα την οποία έχει επιλέξει. Στον λογαριασμό του καταθέτει όσα χρήματα επιθυμεί. Η ύπαρξη του λογαριασμού είναι απαραίτητη προκειμένου σε αυτόν να αποθηκεύονται τα αποτελέσματα των συναλλαγών του κατόχου. Επίσης, ο πελάτης έχει την δυνατότητα προσθήκης στον λογαριασμό του όλων των μεριδίων μετοχών που έχει στην κατοχή του. Η ίδια διαδικασία ακολουθείται από όλους τους πελάτες που μπαίνουν στην αγορά. Μετά την επιλογή τράπεζας ακολουθεί η επιλογή χρηματιστή με τις ίδιες προϋποθέσεις.

Οι πελάτες στην συνέχεια έχουν την δυνατότητα να πληροφορηθούν για τις διαθέσιμες μετοχές που υπάρχουν για πώληση/αγορά στα πλαίσια της χρηματαγοράς, μέσω του χρηματιστή τους. Κατόπιν, θέτουν τις εντολές τους στον χρηματιστή τους, για παράδειγμα πώληση 100 μετοχών της εταιρείας Χ. Ο χρηματιστής με τη σειρά του αναζητά άλλες αντίστοιχες αιτήσεις για πώληση των ίδιων μετοχών από πελάτες που έχουν εκφράσει το αίτημα τους είτε στον ίδιο, είτε σε άλλους χρηματιστές στα πλαίσια της χρηματαγοράς. Αυτό που πρέπει να σημειωθεί είναι η έντονη παρουσία της διαμεσολάβησης, τόσο στα πλαίσια της χρηματαγοράς η οποία λειτουργεί ως διαμεσολαβητής αιτημάτων μεταξύ χρηματιστών, όσο και στα πλαίσια των χρηματιστών οι οποίοι λειτουργούν ως εκπρόσωποι και διαμεσολαβητές για τους πελάτες τους. Αν το αίτημα του πελάτη δεν καλυφθεί αμέσως, μπαίνει σε λίστα αναμονής, εγείροντας την ένδειξη της πώλησης, περιμένοντας για σχετικές αιτήσεις.

Οι υπόλοιποι χρηματιστές λαμβάνουν γνώση για το αίτημα πώλησης του συγκεκριμένου πακέτου μετοχών. Ακολούθως ένας άλλος πελάτης θέτει ένα αίτημα αγοράς όλου ή ενός μέρους του προαναφερθέντος πακέτου. Ο χρηματιστής με τη σειρά του ανακαλύπτει την ταύτιση των αιτημάτων που υπάρχει και πληροφορεί τον πελάτη του θέτοντας του αίτημα για διαπραγμάτευση με τον κάτοχο της προσφοράς. Αν το αίτημα γίνει αποδεκτό, ενεργοποιείται η φάση της διαπραγμάτευσης της μετοχής ανάμεσα σε πωλητή και αγοραστή. Κατά τη διάρκεια της διαπραγμάτευσης υπάρχει η ευχέρεια πρότασης και αντιπρότασης τιμών.

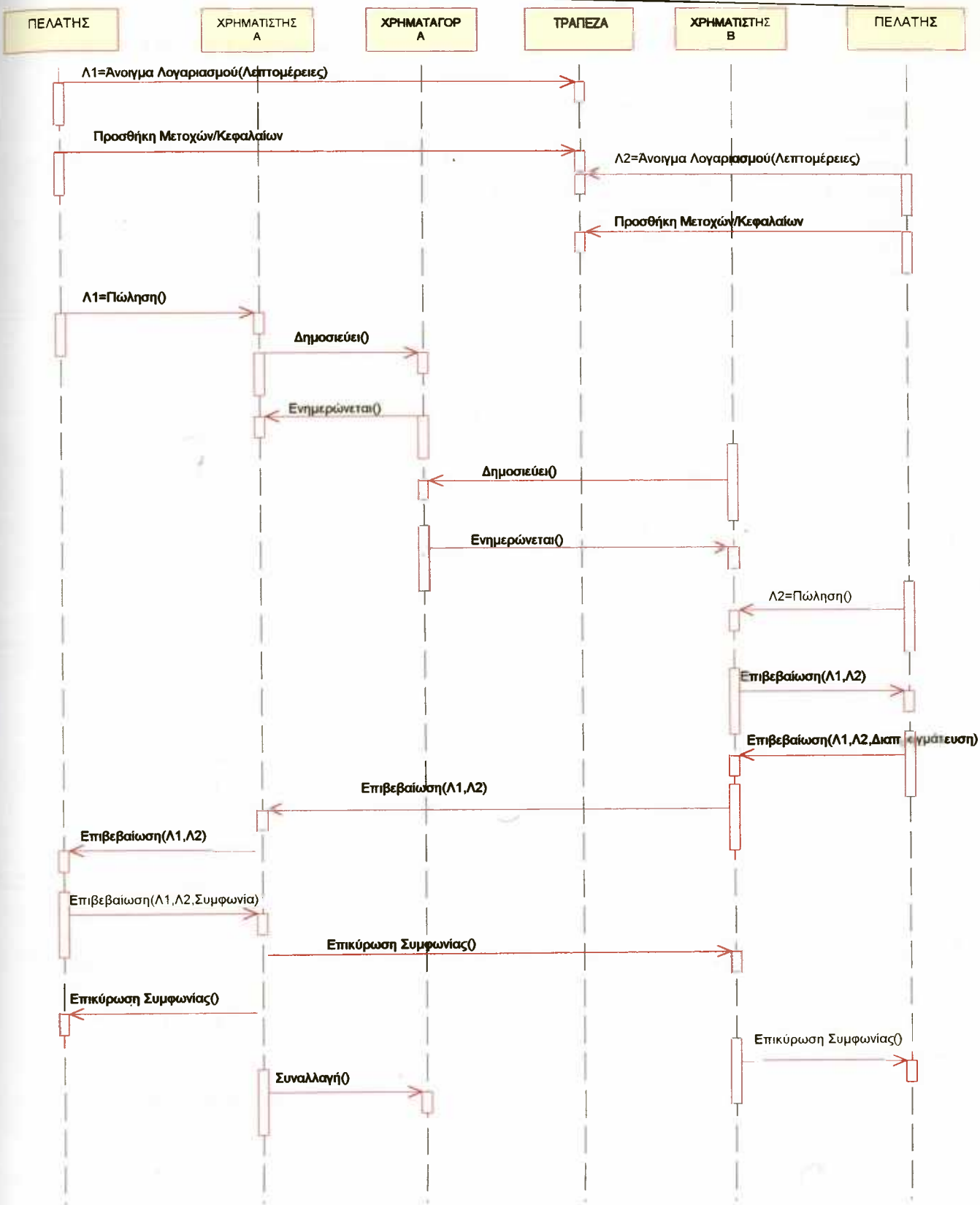
Η εξέλιξη της διαδικασίας οδηγεί όπως και κάθε διαπραγμάτευση σε συμφωνία ή διαφωνία. Αν η διαπραγμάτευση αποτύχει, τότε το καθεστώς παραμένει ως έχει χωρίς αλλαγές στους λογαριασμούς των πελατών, ενώ το αίτημα εξακολουθεί να υφίσταται αναμένοντας μία νέα διαδικασία διαπραγμάτευσης εξ' αρχής, με όρους και συνθήκες όμοιες με αυτές που προαναφέρθηκαν. Σε περίπτωση συμφωνίας, ο αγοραστής έχει αυτόματα στον λογαριασμό του τις μετοχές που μόλις αγόρασε με αντίστοιχη μείωση του διαθέσιμου χρηματικού ποσού στον λογαριασμό του, ενώ ο πωλητής έχει και αυτός με τη σειρά του πιστωμένο στον λογαριασμό του το αντίστοιχο συμφωνηθέν ποσό. Η συναλλαγή κλείνει και το αίτημα που την εξέφρασε αποσύρεται από την διαπραγμάτευση.

Η διαδικασία αυτή αποτελεί την συνηθισμένη οδό ικανοποίησης ενός αιτήματος. Υπάρχουν και οι διάφορες υποπεριπτώσεις, στις οποίες τα αιτήματα αγοραστών και πωλητών δεν καλύπτονται απόλυτα. Πιο συγκεκριμένα:

- Ο πωλητής ή ο αγοραστής εκφράζει την επιθυμία ακύρωσης της εντολής του. Τότε οι αντίστοιχοι χρηματιστές ενημερώνουν την χρηματαγορά για την αλλαγή που προέκυψε και διαγράφουν το συγκεκριμένο αίτημα από τις λίστες τους.
- Το αίτημα για αγορά είναι μικρότερο από το προσφερόμενο. Τότε η συναλλαγή πραγματοποιείται κανονικά με τη διαδικασία που αναφέρθηκε στην αρχή. Το υπόλοιπο του πακέτου των μετοχών αφήνεται στην χρηματαγορά για περαιτέρω διαπραγμάτευση με άλλους εν δυνάμει αγοραστές.
- Το υπόλοιπο του αγοραστή είναι μικρότερο από το απαιτούμενο για την αγορά των μετοχών που επιθυμεί. Η συναλλαγή επίσης χρησιμοποιείται, ο πωλητής λαμβάνει το κανονικό ποσό που έπρεπε να λάβει, ενώ ο αγοραστής χρεώνεται με αρνητικό υπόλοιπο οφειλόμενο στην τράπεζα του.
- Ύπαρξη ενδιαφέροντος για αγορά άνω των πελατών. Στην περίπτωση αυτή ξεκινά η διαπραγμάτευση με τον πρώτο χρονικά που θα θέσει το αίτημα του. Αν η διαπραγμάτευση μεταξύ των δύο καταλήξει θετικά, ισχύουν ότι και αρχικά, ενώ το

αίτημα του ενδιαφερόμενου παραμένει στην χρηματαγορά αναμένοντας είτε ικανοποίηση από άλλη σχετική αντιπροσφορά, είτε απόσυρση από τον κάτοχο του.

Ακολουθεί η αναπαράσταση των βασικών λειτουργιών μέσω της UML.



9.2 Τεχνική Επεξήγηση

Στο κεφάλαιο αυτό θα αναφερθούμε σε διάφορες τεχνικές λεπτομέρειες που έχουν να κάνουν με την λειτουργικότητα της εφαρμογής αφενός, αλλά και τις τεχνικές ιδιαιτερότητες του e-speak ως πλατφόρμας, αφετέρου. Βασικό συστατικό στη δήλωση των υπηρεσιών στον πυρήνα της εφαρμογής αποτελεί η XML([30]). Στη συγκεκριμένη περίπτωση λεξιλόγιο ιδιοτήτων στην XML χρησιμοποιήθηκε για τον ορισμό της τράπεζας και των χρηματιστών. Το λεξιλόγιο για να είναι ορατό και προσβάσιμο από τους χρήστες πρέπει αφότου δημιουργηθεί και καταχωρηθεί στον πυρήνα, να δημιουργηθεί μια δήλωση αντίστοιχη με την Java-RMI-IRL ([17],[18])(RMI: Remote Method Invocation).

Για την δημιουργία-ορισμό-χρήση του λεξιλογίου απαραίτητη είναι μια προδιαγεγραμμένη διαδικασία, η οποία απαιτεί τα ακόλουθα:
1) Σύνδεση στον πυρήνα του e-speak.
2) Αναζήτηση αν υπάρχει καταχωρημένο άλλο αντίστοιχο λεξιλόγιο (π.χ. τράπεζας).
3) Εάν δεν υπάρχει αντίστοιχο λεξιλόγιο, δημιουργείται ένα νέο και καταχωρείται στον πυρήνα του e-speak. Στο εξής θα είναι διαθέσιμο και άμεσα χρησιμοποιήσιμο από οποιονδήποτε χρήστη. Τα λεξιλόγια αντιμετωπίζονται και αυτά ως υπηρεσίες.
4) Δημιουργία του σχετικού espeak IDL-contruct που δίνει την δυνατότητα χρήσης του λεξικού ως πόρου (resourse) σε ένα κατανεμημένο περιβάλλον. Πιο συγκεκριμένα:

9.2.1 Δήλωση - Χρήση Λεξιλογίου στο e-speak

Για την δήλωση της τράπεζας χρησιμοποιούμε το παρακάτω XML σχήμα. Το e-speak παρέχει ένα βασικό μοντέλο XML σχήματος, το οποίο αν δεν καλύπτει πλήρως τις ανάγκες μπορεί να επεκταθεί προσθέτοντας συγκεκριμένα χαρακτηριστικά. Για την περιγραφή της τράπεζας χρησιμοποιήθηκε αυτή η δυνατότητα προσθέτοντας επιπλέον στοιχεία για την ολοκληρωμένη περιγραφή (Bankname, Street, City, Country - στα Αγγλικά για την αποφυγή δυσλειτουργιών, λόγω των ελληνικών χαρακτήρων). Το συγκεκριμένο XML αρχείο περιγράφει τα γνωρίσματα που πρέπει να έχει ένα οποιοδήποτε λεξιλόγιο που αναφέρεται σε μια τραπεζική υπηρεσία

```
<?xml version="1.0"?>
<E-speak version="E-speak 1.0beta" operation="CreateVocab"
xmlns="http://localhost/e:/Esxml/Schemas/e-speak.xsd">
<resource xmlns="">
<!-- Begin: Specify the vocabulary description -->
<resourceDes xmlns="">
<pattern>
<Name>
        BankVocabulary
</Name>
<Type>
        ESVocabulary
</Type>
</pattern>
</resourceDes>
<!-- End: Specify the vocabulary description -->
<resourceData xmlns="">
<!-- Begin: Specify attribute property set -->
```

```
<attrGroup name="BankVocabulary" xmlns="">
   <attrDecl xmlns="" name="BankName">
       <datatypeRef name="string"/>
       </attrDecl>
    <attrDecl xmlns="" name="Street">
       <datatypeRef name="string"/>
   </attrDecl>
   <attrDecl xmlns="" name="City">
       <datatypeRef name="string"/>
   </attrDecl>
   <attrDecl xmlns="" name="Country">
       <datatypeRef name="string"/>
   </attrDecl>
</attrGroup>
</resourceData>
<!-- End: Specify attribute property set -->
</resource>
</E-speak>
```

Ακολούθως θα περιγράψουμε τα βασικά στοιχεία της κλάσης που υλοποιεί τα προηγούμενα τέσσερα (4) βήματα για την δήλωση του λεξιλογίου ως υπηρεσία. Η βασική κλάση που χρησιμοποιείται είναι η BankVocabulary (περιέχεται στο αρχείο BankVocabulary.java). Στην κλάση αυτή υπάρχει η μέθοδος :

public static void main (String[] args),

η οποία δέχεται σαν ορίσματα ένα αρχείο περιγραφής του λεξιλογίου (π.χ. Bankvocab.xml). Στην συνέχεια πραγματοποιεί μια σύνδεση στον πυρήνα χρησιμοποιώντας ορισμένα προκαθορισμένα ports. Οι λεπτομέρειες της σύνδεσης με τον πυρήνα περιγράφονται στο αρχείο προσδιορισμού των παραμέτρων αυτών (π.χ. bank.pr). Ακολουθεί η σύνδεση :

ESConnection session = new ESConnection ("bank.pr");

Το λεξιλόγιο δημιουργείται, χρησιμοποιώντας τον δημιουργό (constructor):

BankVocabulary bv = new BankVocabulary (session, args[0], args[1]);

όπου arg[0] το αντίστοιχο xml αρχείο και arg[1] το αντίστοιχο contract name που χρησιμοποιείται για τον ορισμό της υπηρεσίας όπως θα δούμε και στην συνέχεια.
Ο δημιουργός:

public BankVocabulary (ESConnection session, string vocabXML File, string contractName)

αναζητά στο εσωτερικό repository των υπηρεσιών το συγκεκριμένο λεξιλόγιο. Εάν αυτό δεν υπάρχει, δημιουργείται άμεσα με τις προδιαγραφές που υπάρχουν στο XML αρχείο περιγραφής ιδιοτήτων. Ο κώδικας είναι ο εξής:

```
// Εύρεση του λεξιλογίου εάν αυτό είναι διαθέσιμο ειδάλλως δημιουργείται ένα νέο
ESVocabulary bankVocab = null;
ESVocabularyFinder vf = new ESVocabularyFinder(session);
try
 {
   bankVocab = vf.find( new ESQuery( "Name == 'BankVocabulary'") );
```

```
  }
catch (LookupFailedException e)
{
  // Δημιουργία  νέου λεξιλογίου
  ESVocabularyDescription esvd = new ESVocabularyDescription(
  new ESXMLFile(vocabXMLFile), session);
  ESVocabularyElement esvc = new ESVocabularyElement(session, esvd);
  vocab = esvc.register();
  ((ESVocabularyStub)vocab).getAccessor();

  // Δημιουργία του συμβολαίου που περιγράφει επακριβώς την υλοποίηση της
υπηρεσίας
  // και αναφορά στο esidl αρχείο που θα χρησιμοποιηθεί.
  ESContractDescription escd = new ESContractDescription();
  escd.setInterfaceName(contractName);
  escd.addAttribute(ESConstants.SERVICE_NAME, "BankContract");
  escd.setInterfaceDefinition("BankServiceIntf.esidl");
  ESContractElement escc =new ESContractElement(session, escd);
  ESContract cont = escc.register();
}
```

Η υπηρεσία του λεξιλογίου της τράπεζας αποτελεί τον ορισμό μιας οντότητας "τράπεζα". Οι εμφανίσεις της οντότητας αποτελούν την συγκεκριμενοποίηση της σε πραγματικά ονόματα τραπεζών.

9.2.2 Δημιουργία - Δήλωση - Χρήση της Υπηρεσίας

Τα προαναφερθέντα αναφέρονται στην περιγραφή της υπηρεσίας με ένα συγκεκριμένο τρόπο (χρήση XML) με σκοπό την διαφήμιση της υπηρεσίας σε συγκεκριμένες ομάδες ή στο Internet γενικότερα. Η υλοποίηση της υπηρεσίας αποτελεί κάτι το διαφορετικό.
Αναφερόμενοι στο συγκεκριμένο παράδειγμα της τραπεζικής υπηρεσίας στην συνέχεια θα αναφερθούμε στα συγκεκριμένα βήματα που απαιτούνται για τον προσδιορισμό της υπηρεσίας αλλά και του πελάτη (client) που θα την χρησιμοποιήσει. Πιο αναλυτικά έχουμε για το καθένα τα εξής βήματα:

Υπηρεσία (Τραπεζική):
1. Σύνδεση στον πυρήνα της αρχιτεκτονικής
2. Έλεγχος ύπαρξης του κατάλληλου λεξιλογίου που περιγράφει την συγκεκριμένη τραπεζική υπηρεσία
3. Υλοποίηση της υπηρεσίας και καταχώρησή της στον πυρήνα
4. Εκκίνηση της υπηρεσίας

Πελάτης Υπηρεσίας(Τραπεζική):
1. Σύνδεση στον πυρήνα της αρχιτεκτονικής
2. Ανακάλυψη του λεξιλογίου για το οποίο ενδιαφέρεται
3. Χρήση του λεξιλογίου, μορφοποίηση ερωτήματος και ανακάλυψη υπηρεσίας

Προκειμένου να περιγραφεί επακριβώς η συγκεκριμένη τραπεζική υπηρεσία απαραίτητη είναι η δήλωση της μέσω του XML σχήματος αρχείου που πληροί τις

προϋποθέσεις του γενικού XML αρχείου που περιγράψαμε προηγουμένως. Συγκεκριμένα έχουμε:

```
<?xml version="1.0"?>
<E-speak version="E-speak 1.0beta" operation="RegisterService"
xmlns="http://localhost/e:/Esxml/Schemas/e-speak.xsd">
<resource>
<resourceDes xmlns="" name="Bank Description">
<!-- Begin: Specify Bank Vocabulary in a query-->
<query xmlns="">
<queryBlock xmlns="">
<WHERE xmlns="">
<condition xmlns="">
<IN xmlns="">
<pattern xmlns="">
<ResourceName xmlns="">BankVocabulary</ResourceName>
<ResourceType xmlns="">ESVocabulary</ResourceType>
</pattern>
</IN>
</condition>
</WHERE>
</queryBlock>
</query>
<!-- End: Specify Bank Vocabulary -->
<!-- Begin: the attribute list -->
<attrSet xmlns="">
<!-- End: Use bank vocabulary -->
<attr xmlns="" name="BankName" required="true">
<value xmlns="">Ethniki Bank</value>
</attr>
<attr xmlns="" name="City" required="true">
<value xmlns="">Athens</value>
</attr>
</attrSet>
<!-- End: the attribute list -->
</resourceDes>
</resource>
</E-speak>
```

Το XML σχήμα προσδιορίζει στο <queryBlock>την προέλευση του αρχείου που θεωρείται ως κέντρο αναφοράς. Στην συνέχεια προσδιορίζουμε τιμές για δύο μόνο χαρακτηριστικά την ονομασία και την έδρα της τράπεζας. Για την δημιουργία την τραπεζικής υπηρεσίας (BankService) απαραίτητα είναι τα ακόλουθα:

java BankService BankVocabulary.xml BankServiceIntf BankService.xml

Η κλάση BankService έχει την μέθοδο main() που δημιουργεί και καταχωρεί την υπηρεσία ως ακολούθως:

```
ESConnection session =new ESConnection("bank.pr");
```

```
BankVocabulary bv = new BankVocabulary(session, args[0], args[1]);
BankService bs = new BankService(session, args[2], args[1]);
```

Η δημιουργία του λεξιλογίου επεξηγήθηκε προηγουμένως. Ο κώδικας που ακολουθεί αναφέρεται στην δήλωση της τραπεζικής υπηρεσίας. Ο τρόπος είναι παρόμοιος με αυτόν που χρησιμοποιήθηκε για το λεξιλόγιο.

```
// Δημιουργία μιας περιγραφής της υπηρεσίας χρησιμοποιώντας XML
ESServiceDescription essd =
new ESServiceDescription(new ESXMLFile(svcXMLFile), session);
// Προσθήκη του συμβολαίου που χρησιμοποιείται για την
δήλωσηessd.addAttribute( ESConstants.SERVICE_NAME, contractName );
//Δημιουργία ενός  service element και συσχέτιση του με ένα  implementation object
ESServiceElement servElem = new ESServiceElement(session, essd);
servElem.setImplementation(new BankServiceImpl());
// Καταχώρηση και έναρξη της υπηρεσίας
ESAccessor accessor = servElem.register();
servElem.start();
```

Παίρνοντας ως παράδειγμα την τραπεζική υπηρεσία, απαραίτητη είναι η δημιουργία μιας διεπαφής (Interface) για αυτήν. Αυτό επιτυγχάνεται με το e-speak IDL, το οποίο είναι παρόμοιο με το Java RMI IDL. Το IDL αρχείο πρέπει να έχει την "*.esidl*" κατάληξη, απαραίτητη για την μεταγλώττιση του από τον IDL Compiler, ο οποίος υπάρχει ενσωματωμένος στο e-speak. Το BankServiceIntf.esidl είναι της μορφής:

```
public interface BankServiceIntf extends ESService
{
        public String openAccount(String user, Address address)
          throws ESInvocationException;
        public boolean changePassword(String accountNumber,String
        oldPassword,String newPassword)
          throws ESInvocationException;
        public void addShare(String accountNumber, ShareDetailParam share)
          throws ESInvocationException;
        public void addCash(String accountNumber, float cash)
          throws ESInvocationException;
        public float listBalance(String accountNumber)
          throws ESInvocationException;
        public String getAccountName(String accountNumber)
          throws ESInvocationException;
        public ESArray listShares(String accountNumber)
          throws ESInvocationException;
        public ESArray listTransactions(String accountNumber)
          throws ESInvocationException;
        public void closeTransaction(OrderDetailParam order,OrderDetailParam
        matchWith)
          throws ESInvocationException;
        public String login(String accountNumber, String password)
          throws ESInvocationException;
```

}

- public String openAccount(String user, Address address);

Η μέθοδος αυτή ανοίγει ένα λογαριασμό με τις λεπτομέρειες που προσδιορίζονται

- boolean changePassword(String accountNumber, String oldPassword,String newPassword);

Η μέθοδος αυτή αλλάζει το password για ένα συγκεκριμένο λογαριασμό.

- void addShare(String accountNumber, ShareDetailParam share);

Η μέθοδος αυτή προσθέτει τη μετοχή στο λογαριασμό

- void addCash(String accountNumber, float cash);

Η μέθοδος αυτή προσθέτει μετρητά στο λογαριασμό

- float listBalance(String accountNumber);.

Η μέθοδος αυτή επιστρέφει το υπόλοιπο του λογαριασμού

- String getAccountName(String accountNumber);

Η μέθοδος αυτή επιστρέφει το όνομα του κατόχου ενός λογαριασμού

- ESArray listShares(String accountNumber);

Η μέθοδος αυτή επιστρέφει την λίστα των μετοχών ενός λογαριασμού

- ESArray listTransactions(String accountNumber);

Η μέθοδος αυτή επιστρέφει την λίστα των συναλλαγών ενός λογαριασμού

- String login(String accountNumber, String password);

Η μέθοδος αυτή επιτρέπει την είσοδο του χρήστη στο λογαριασμό

Μετά την μεταγλώττιση του **BankServiceIntf.esidl** παράγονται τα εξής αρχεία:

- **BankServiceIntf.java**

Το οποίο αποτελεί σχεδόν αντίγραφο του **BankServiceImpl.esidl** και παίζει τον ρόλο του κοινού σημείου μεταξύ της υλοποίησης της τραπεζικής υπηρεσίας (BankService) και του client side κομματιού που κάνει κλήση της συγκεκριμένης υπηρεσίας.

- **BankServiceStub.java**

Το οποίο αποτελεί την απάντηση που επιστρέφεται σε έναν πελάτη που επιθυμεί να χρησιμοποιήσει την υπηρεσία BankService (ή οποιαδήποτε άλλη) όταν αυτή ανακαλυφθεί. Για κάθε μέθοδο που υπάρχει στο Java Interface, το *stub αρχείο περιέχει τον κώδικα δημιουργίας μηνυμάτων, marchal παραμέτρων (εσωτερική κωδικοποίηση και αρίθμηση αντικειμένων) από και προς τις δύο εφαρμογές.

- **BankServiceIntfFinder.java**

Σκοπός της κλάσης αυτής είναι η μέθοδος είναι η εύρεση μιας τραπεζικής υπηρεσίας (BankService) που έχει καταχωρηθεί στον πυρήνα της εφαρμογής. Η υλοποίηση της αναζήτησης των τραπεζών στη συγκεκριμένη περίπτωση πραγματοποιείται με τη συγγραφή του κατάλληλου κώδικα και τον προσδιορισμό συγκεκριμένων κριτηρίων.

```
static ESConnection session = null;
static ESVocabulary vocab = null;
public static BankServiceIntf bsi=null;
String contractName = null;
String vocabxmlFile = null;
{
  bsi=null;
  try
   {
      session=new ESConnection("bank.pr");
```

```
        ESVocabularyFinder vf = new ESVocabularyFinder(session);
        vocab = vf.find( new ESQuery("Name == 'BankVocabulary'" ) );
        String queryString = "(BankName == '" + "Ethniki Bank" + "' and "+
              "City == '" + "Athens" + "')";
        String interfaceName= "BankServiceIntf";
        ESQuery query = new ESQuery(vocab,queryString);
        ESServiceFinder sf = new ESServiceFinder(session,interfaceName);
        bsi = (BankServiceIntf) sf.find(query);
    }
}
```

Η υλοποίηση του πελάτη της υπηρεσίας γίνεται μέσα από την κλάση **BankClient**. Μεταξύ των άλλων μεθόδων, η βασική μέθοδος της κλάσης είναι η :

public static void main (String args[]).

Η συγκεκριμένη παίρνει ως παραμέτρους το λεξιλόγιο που είναι απαραίτητο, το όνομα της τράπεζας και την πόλη που δημιουργεί μια σύνδεση με τον πυρήνα αναζητώντας την υπηρεσία που περιγράφεται:

public BankClient(ESConnection session, String vocabName, String bankName,String city);

Ο δημιουργός ανακαλύπτει λεξιλόγιο και υπηρεσία ως εξής:

```
//Εύρεση του λεξιλογίου
ESVocabularyFinder vf = new ESVocabularyFinder(session);
try
  {
    bankVocab = vf.find(new ESQuery( "Name == '" +vocabName + "'" ) );
  }
//Εύρεση της υπηρεσίας
String queryString = "(BankName == '" + bankName + "' and " +
"City == '" + city + "')";
System.out.println("Looking for a bank where :"+ queryString + " \n");
String interfaceName= "BankServiceIntf";
ESQuery query = new ESQuery(bankVocab,queryString);
ESServiceFinder sf = new ESServiceFinder(session,interfaceName);
try
{
   bsif = (BankServiceIntf) sf.find(query);
   System.out.println("Found the service");
   Address ad = new Address();
   sb.openAccount(sb,ad);
 }
```

Παράλληλα με την τραπεζική υπηρεσία και με τον ίδιο τρόπο ορίζουμε και τις υπόλοιπες υπηρεσίες που συμμετέχουν στην εφαρμογή (πελάτης, τράπεζα, χρηματιστής). Στην συνέχεια θα αναφερθούμε στις σημαντικότερες κλάσεις που πλαισιώνουν την εφαρμογή με μια πολύ συνοπτική περιγραφή του τι ακριβώς

επιτελούν. Για λεπτομερέστερη μελέτη ο αναγνώστης μπορεί να ανατρέξει στο παράρτημα της εφαρμογής και στα σχόλια που αντιστοιχούν στα αντίστοιχα τμήματα.

9.3 Βασικές κλάσεις

- **StockTrade.java**

Είναι η κλάση που ενεργοποιεί το βασικό παράθυρο της εφαρμογής με την επιγραφή "Χρηματαγορά". Είναι υπεύθυνη για το γραφικό τμήμα της εφαρμογής και για την λειτουργικότητα των επιμέρους τμημάτων, περιλαμβάνοντας τη δημιουργία και τη συντήρηση των λογαριασμών, την αγοραπωλησία μετοχών, την επιλογή χρηματιστή-τράπεζας κ.λ.π. Η StockTrade ελέγχει άμεσα τα παρακάτω αρχεία:
AddFundsGUI.java, AddSharesGUI.java, AuthorizationWindow.java, BuySharesGUI.java, ListSharesGUI.java, SellSharesGUI.java OpenAccountGUI.java, ListAccountGUI.java,LoginGUI.java,

- **TraderIntfFinder**

Δίνει την δυνατότητα εντοπισμού των διαθέσιμων πελατών στο repository. Διαθέτει την ακόλουθη μέθοδο η οποία χρησιμοποιείται από την οντότητα 'Χρηματιστής' για τον εντοπισμό των πελατών.

public static TraderIntf find (TraderQuerySpec *query*Spec)

- **TraderVocabulary.java**

Δημιουργεί και καταχωρεί ένα λεξιλόγιο Πελάτη και το αντίστοιχο συμβόλαιο στον πυρήνα. Χρησιμοποιεί την εξής μέθοδο:

public TraderVocabulary(.ESConnection session, String vocabXMLFile, String contractName)

- **StartShareBrokerService.java**

Η κλάση αυτή εκκινεί την οντότητα 'Χρηματιστής'. Χρησιμοποιεί τον εξής δημιουργό:

public StartShareBrokerService(String BrokerName)

- **ShareBrokerIntfFinder.java**

Η συγκεκριμένη κλάση παρέχει μια μέθοδο εύρεσης των 'Χρηματιστών' που είναι καταχωρημένοι στον πυρήνα του e-speak. Κυρίως χρησιμοποιείται από την κλάση StockTrade.

public static ShareBrokerIntfFinder()

- **ShareBrokerImpl.java**

Η κλάση υλοποιεί τις μεθόδους που περιγράφονται στο ShareBrokerIntf θεωρητικά. Στην πράξη γίνεται μια αναστροφή προς την κλάση BrokerLogic που πραγματοποιεί και την ουσιαστική επεξεργασία σε όλη την εφαρμογή.

- **BrokerLogic.java**

Η κλάση αυτή πραγματοποιεί όπως είπαμε την πραγματική επεξεργασία που περιγράφεται στην προηγούμενη κλάση. Κρατά όλη εκείνη την πληροφορία για τις

αγορές και τις πωλήσεις μετοχών από διαφορετικούς πελάτες. Παρέχει επίσης μεθόδους για την αποδοχή εντολών, την ταύτιση τους, την πληροφόρηση των πελατών κλπ.
Μερικές από τις μεθόδους της κλάσης είναι:

public synchronized void generateOrderNumber(TraderQuerySpec query, OrderDetail order)
Η μέθοδος παράγει έναν αριθμό παραγγελίας για τον πελάτη που μπορεί να το χρησιμοποιήσει σαν αναφορά

public String buyShares(TraderQuerySpec query, OrderDetail order)
Η μέθοδος διαχειρίζεται την διαδικασία αγοράς μετοχών

public void adjustAndStartDeal(OrderDetail order, OrderDetail matchWith)
Η μέθοδος υπολογίζει πόσες μετοχές μπορούν να μετακινηθούν από την μία εντολή στην άλλη. Στη συνέχεια ακολουθεί η φάση της διαπραγμάτευσης.

public synchronized void match(OrderDetail order)
Είναι η μέθοδος που προσπαθεί να ταυτοποιήσει εντολές πού προκύπτουν με αυτές που υπάρχουν στους χρηματιστές. Αμέσως μετά την αρχική ταύτιση προσπαθούμε να εντοπίσουμε αν μπορεί να υπάρξει έναρξη της διαπραγμάτευσης. Αν η διαφορά μεταξύ εντολής και προσφοράς είναι μεγαλύτερη από την μεταβλητή PRICE_DIFF_PCNT τότε η διαπραγμάτευση δεν ξεκινά.

public void closeTheDeal(OrderDetail order, OrderDetail matchWith)
Η μέθοδος αυτή πληροφορεί την τράπεζα για μεταφορά χρημάτων από τον ένα λογαριασμό στον άλλο.

Στα πλαίσια της κλάσης BrokerLogic πρέπει να τονίσουμε την μεγάλη σημασία της διαπραγμάτευσης μεταξύ των οντοτήτων της εφαρμογής. Σε αυτήν δίνεται η δυνατότητα χρήσης πολλαπλών χρηματιστών, οι οποίοι διαμορφώνουν μια κοινότητα και επιτρέπουν τις συναλλαγές μέσα στα πλαίσια της. Όλες οι τράπεζες και οι χρηματιστές είναι ορατές και ορατοί αντίστοιχα στους χρήστες. Έτσι αυτοί έχουν το δικαίωμα επιλογής των κατάλληλων για αυτούς παροχέων υπηρεσιών προκειμένου να συνδιαλλαγούν στα πλαίσια της χρηματαγοράς.

Κάθε χρηματιστής "δημοσιεύει" όλες τις κλήσεις των πελατών του για αγορά ή πώληση μετοχών και όλοι οι υπόλοιποι χρηματιστές λαμβάνουν γνώση αυτών των κλήσεων. Υπάρχει ένας κεντρικός διανομέας αυτών των μηνυμάτων. Η διαθεσιμότητα των μετοχών πρέπει επίσης να είναι ενήμερη κάθε στιγμή για να αποτυπώνει την πραγματική κατάσταση. Οι πελάτες συνδεόμενοι στους δικούς τους χρηματιστές υποβάλλουν τα αιτήματα τους. Αυτοί με την σειρά τους προκειμένου να καλύψουν την ανάγκη που παρουσιάζεται αναζητούν τρόπο κάλυψης του αιτήματος στο εσωτερικό τους. Αν η αναζήτηση αποτύχει τότε το αίτημα μεταβιβάζεται προς τους άλλους χρηματιστές της χρηματαγοράς.

Ένα δείγμα του τρόπου με τον οποίο δημοσιοποιούνται τα αιτήματα περιγράφεται στην κλάση **ShareBrokerEventDistributor.java.** Η main() μέθοδος περιγράφει τα εξής:

public static void main(String[] argv)

```
{
try
{
//Κλάση του e-speak που αναφέρεται στην "δημοσιοποίηση" των αιτημάτων προς
//τις άλλες οντότητες. Οι παράμετροι αναφέρονται στον πυρήνα αλλά και στο port
//όπου δημοσιοποιούνται αυτά τα αιτήματα.
ESDistributor dist = new ESDistributor(argv[0],(new
Integer(argv[1])).intValue());
dist.addEvent("SellEvent".toString());
dist.addEvent("BuyEvent".toString());
dist.addEvent("CancelEvent".toString());
dist.start();
dist.setDistType(ESDistributor.ADVERTISED);
}
}
```

Η διανομή αιτημάτων προς άλλους χρηματιστές, αλλά και αποδοχή πληροφόρησης από αυτούς, στα πλαίσια της κοινότητας περιγράφεται με τα παρακάτω αποσπάσματα κώδικα από τις κλάσεις BrokerLogic και ShareBrokerImpl.

Δημοσιοποίηση Αιτημάτων

```
public ShareBrokerImpl(ESConnection conn, String brokerName)
{
...
ESPublisher publ = new ESPublisher(conn);
publ.addEvent("SellEvent".toString());
publ.addEvent("BuyEvent".toString());
publ.addEvent("CancelEvent".toString());
publ.publish();
...
}
```

Πληροφόρηση Αιτημάτων

```
public ShareBrokerImpl(ESConnection conn, String brokerName)
{
...
impl = new BrokerLogic(publ,brokerName);
subs.setImplementation(impl);
ESSubscriber subs = new ESSubscriber(conn);
subs.addEvent("BuyEvent".toString());
subs.addEvent("SellEvent".toString());
subs.addEvent("CancelEvent".toString());
subs.subscribe();
...
}
```

Όταν ο χρηματιστής αποτύχει στην εσωτερική ικανοποίηση του αιτήματος εγείρει ένα αίτημα προς τους άλλους χρηματιστές.

```
public void raiseEvent(OrderDetailParam order)
{
try
{
   Event evt;
   if(order.option==1)
     {
      evt = new Event("BuyEvent");
     }
   else
     {
      evt = new Event("SellEvent");
     }
}
....
}
```

Τέλος κλείνοντας το κομμάτι αυτό θα αναφέρουμε την κλάση

- **ShareBrokerIntfFinder.java**

Η συγκεκριμένη συνδέεται στον πυρήνα με σκοπό την ανακάλυψη χρηματιστών που είναι συνδεδεμένοι σ ' αυτόν.

public static ShareBrokerIntf find(String brokerName)
public static Object[][] findBrokers()

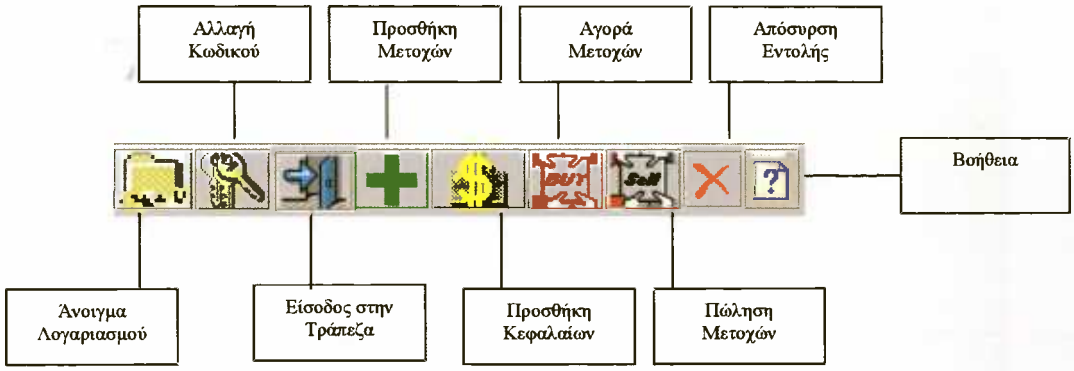
Η μεν πρώτη βρίσκει ένα συγκεκριμένο χρηματιστή που είναι συνδεδεμένος στον πυρήνα ή δε δεύτερη επιστρέφει το σύνολο όλων.

9.4 Γραφική Περιγραφή

Στην συνέχεια θα κάνουμε μια γραφική αναπαράσταση του τρόπου λειτουργίας της εφαρμογής βοηθώντας αποτελεσματικότερα τον αναγνώστη στην πληρέστερη κατανόηση της. Στόχος είναι η παράλληλη αναφορά σε οθόνες και σε συμβάντα που πραγματοποιούνται πίσω από αυτές περιγράφοντας έτσι με μεγαλύτερη σαφήνεια τα συμβαίνοντα στο μοντέλο των συνεργαζόμενων ηλεκτρονικών υπηρεσιών.

9.4.1 Επεξήγηση χειριστηρίων

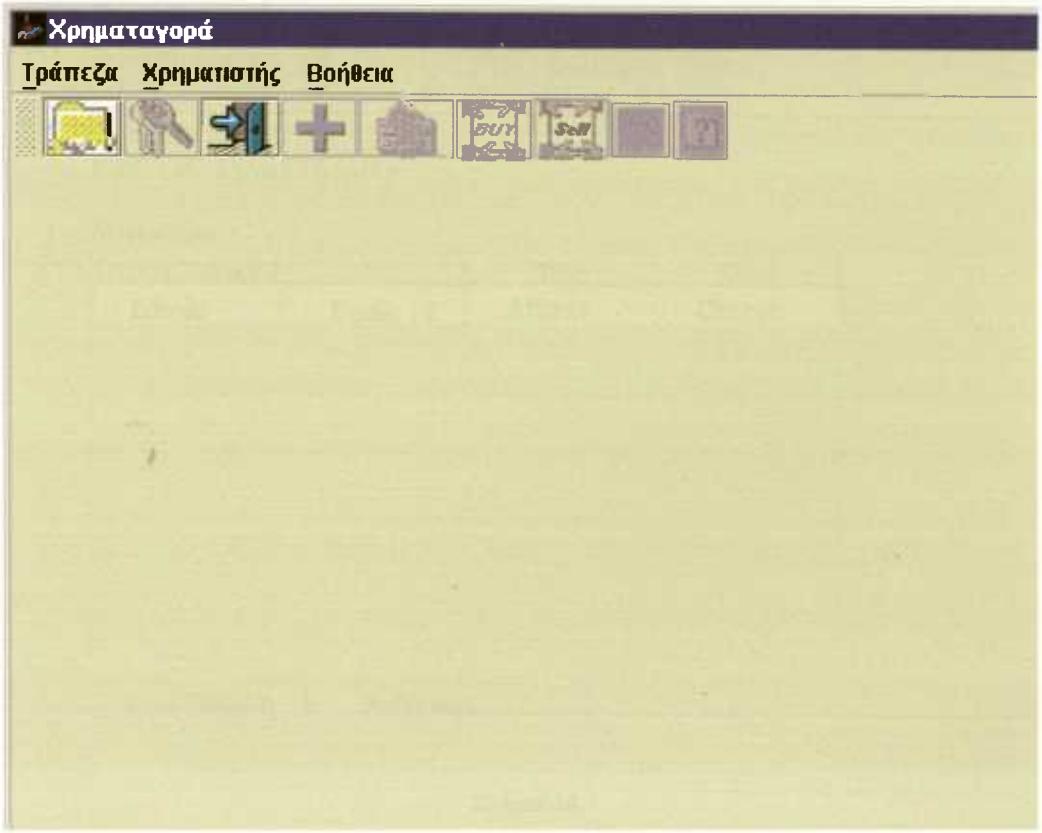
Η λειτουργικότητα της εφαρμογής εμπλουτίζεται με δύο βασικά μέσα ενεργοποίησης. Την ομάδα των κομβίων ενεργοποίησης και τα μενού επιλογών. Αναφερόμενοι στην πρώτη κατηγορία έχουμε τις εξής δυνατότητες.



- **Άνοιγμα Λογαριασμού:** Δίνεται η δυνατότητα στον χρήστη της εφαρμογής να ανοίξει ένα νέο λογαριασμό στην Τράπεζα που έχει επιλεγεί όπως θα δούμε και στην συνέχεια. (Οθόνη 1)
- **Αλλαγή Κωδικού:** Ο χρήστης έχει την δυνατότητα να αλλάξει τον τρέχοντα κωδικό πού του έχει δοθεί από την εφαρμογή με ένα νέο της δικής του επιλογής. (Οθόνη 1)
- **Είσοδος στην Τράπεζα:** Είναι η πρώτη ενέργεια που απαιτείται για την είσοδο στην χρηματαγορά Απαραίτητη για την είσοδο είναι η καταχώρηση του ορθού αριθμού πελάτη και του αντίστοιχου κωδικού πρόσβασης (password).
- **Προσθήκη Μετοχών:** Αφορά στην προσθήκη μετοχών στον λογαριασμό τραπέζης του πελάτη και οι οποίες έχουν αποκτηθεί εκτός διαπραγμάτευσης στην χρηματαγορά.
- **Προσθήκη Κεφαλαίων:** Αφορά στην προσθήκη κεφαλαίου κίνησης στον λογαριασμό τραπέζης του πελάτη προκειμένου να θεωρηθεί σαν αντιστάθμισμα για την αγορά μετοχών.
- **Απόσυρση Εντολής:** Απόσυρση εντολή πραγματοποιείται όταν είτε πωλητής είτε αγοραστής μετοχών αποφασίσει την ακύρωση της συγκεκριμένης εντολής.
- **Βοήθεια:** Επεξηγήσεις και συμβουλές προς τον χρήστη της εφαρμογής.

9.4.2 Επεξήγηση Οθονών

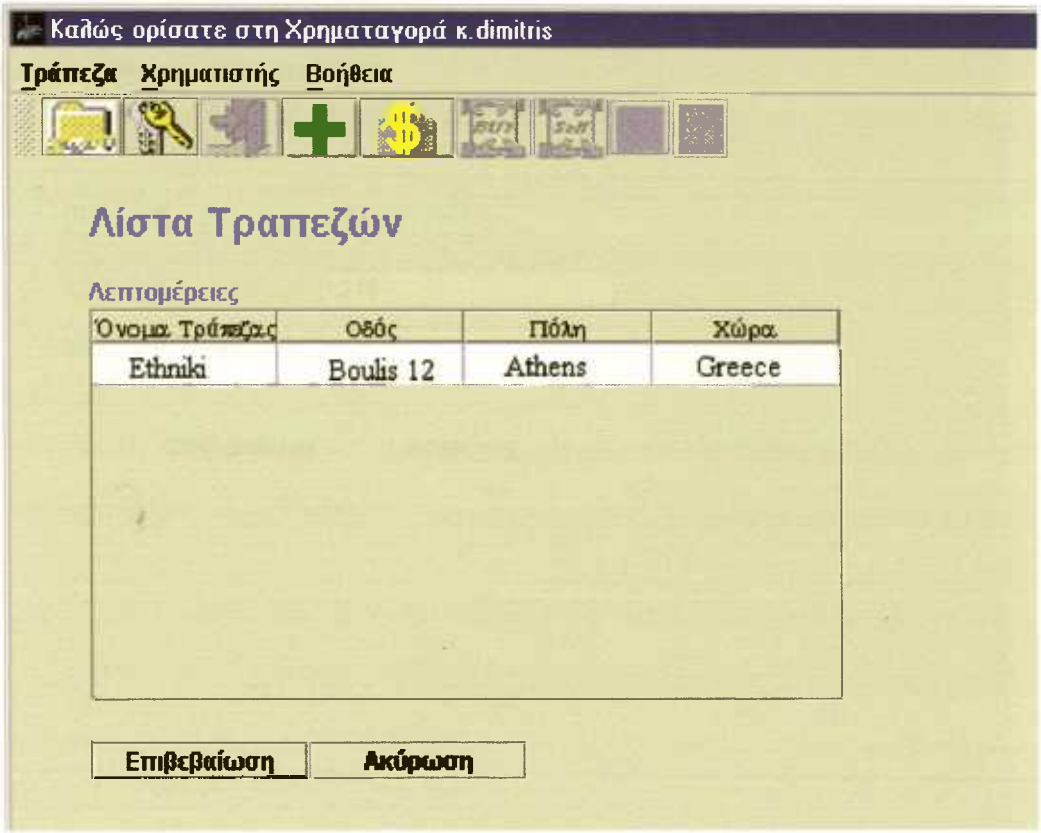
Οθόνη 1η



Σχήμα 23

Η παραπάνω οθόνη είναι η πρώτη που συναντά ο χρήστης κάνοντας την είσοδο του στην εφαρμογή. Η γραμμή των εργαλείων όπως φαίνεται είναι μερικώς απενεργοποιημένη. Για την ενεργοποίηση της και κατά συνέπεια την είσοδο του χρήστη στη χρηματαγορά απαραίτητη προϋπόθεση αποτελεί η ύπαρξη λογαριασμού στην Τράπεζα, εάν αυτός υπάρχει, ειδάλλως έχουμε την δημιουργία ενός νέου. Οι παραπάνω ενέργειες είναι δυνατό να πραγματοποιηθούν και από τις δύο κατηγορίες χειριστηρίων που εμπεριέχονται στην εφαρμογή.

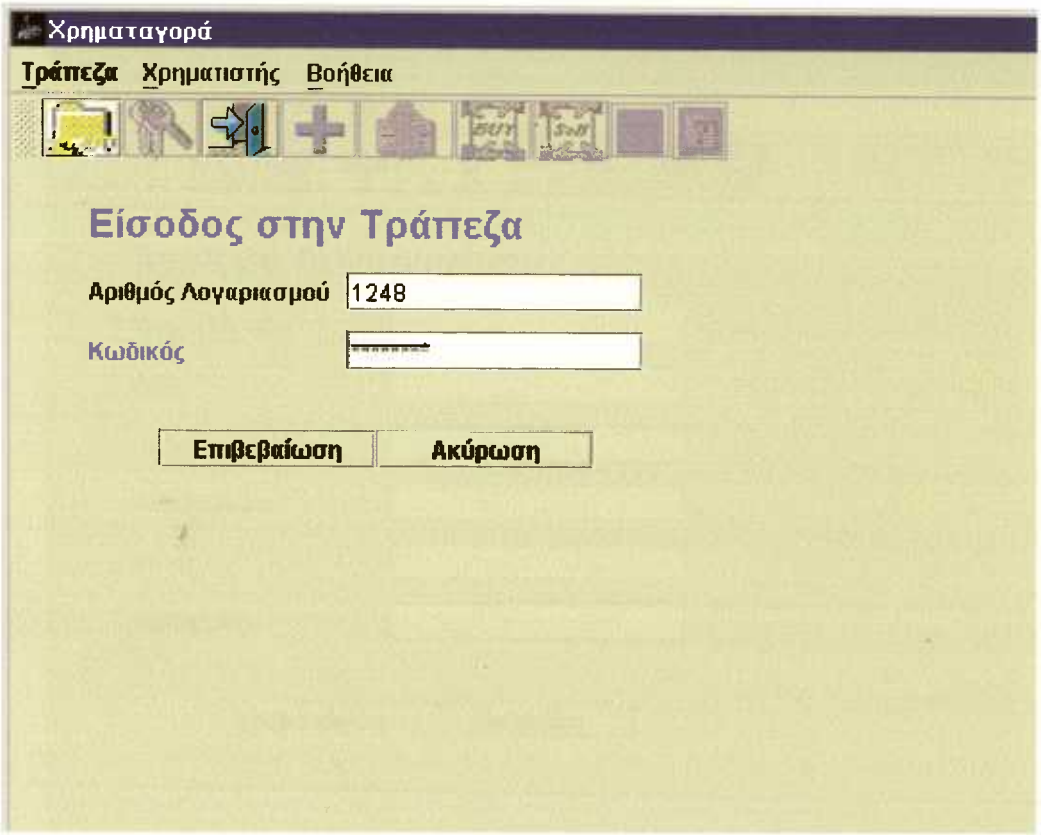
Οθόνη 2η



Σχήμα 24

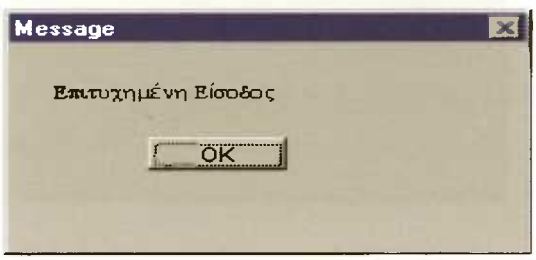
Ο χρήστης επιλέγει την τράπεζα στην οποία έχει ήδη λογαριασμό ή αυτή στην οποία ενδιαφέρεται να δημιουργήσει. Όπως θα δούμε η Τράπεζα παίζει το ρόλο του διεκπεραιωτή της συναλλαγής αναλαμβάνοντας την εκκαθάριση του ποσού της συναλλαγής. Οι τράπεζα(ες) που εμφανίζονται ως διαθέσιμες προς επιλογή έχουν δηλωθεί μέσω XML στην πλατφόρμα που χρησιμοποιούμε (e-speak) προσφέροντας την αντίστοιχη υπηρεσία στους πελάτες της.

Οθόνη 3η



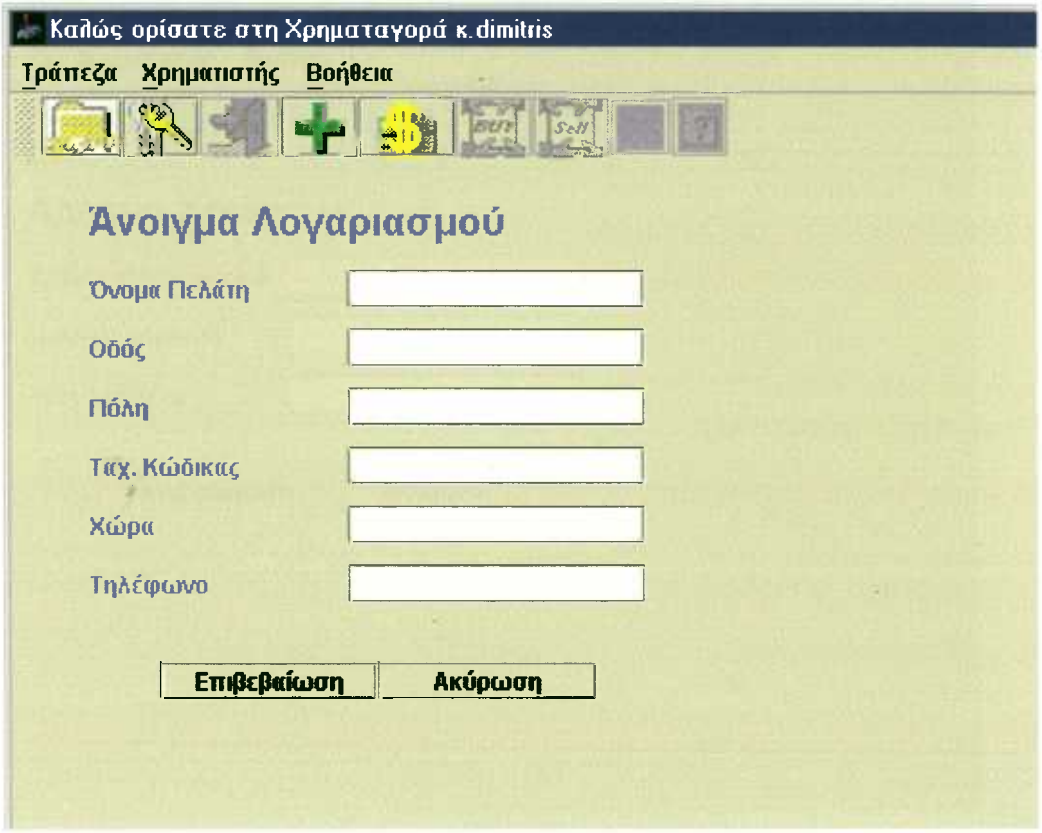
Σχήμα 25

Για την είσοδο του χρήστη στην τράπεζα απαραίτητη είναι η επιτυχημένη καταγραφή του Αριθμού Λογαριασμού του χρήστη και του αντίστοιχου κωδικού του. Ο Αριθμός Λογαριασμού δίνεται στον χρήστη κατά την πρώτη του είσοδο στην εφαρμογή και δεν του δίνεται η δυνατότητα αλλαγής. Αντίθετα ο κωδικός εισόδου που δίνεται για πρώτη φορά στον χρήστη είναι το όνομα του (χρήση καλυμμένου πεδίου) και έχει την δυνατότητα αλλαγής όπως θα δούμε κι στην συνέχεια. Αν η είσοδος είναι επιτυχημένη ο χρήστης λαμβάνει την ακόλουθη επιβεβαίωση, ενώ η εφαρμογή τον υποδέχεται με το όνομα του. Στο εξής το συγκεκριμένο στυγμιότυπο της εφαρμογής θα βρίσκεται υπό την κατοχή του εξουσιοδοτημένου χρήστη (μέσω κωδικού πρόσβασης).



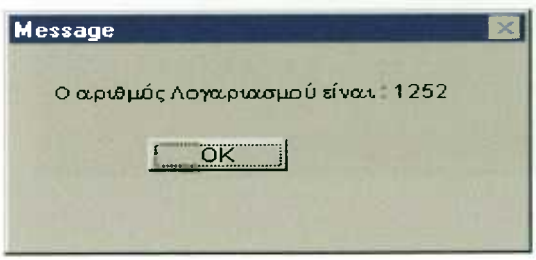
Σχήμα 26

Οθόνη 4η



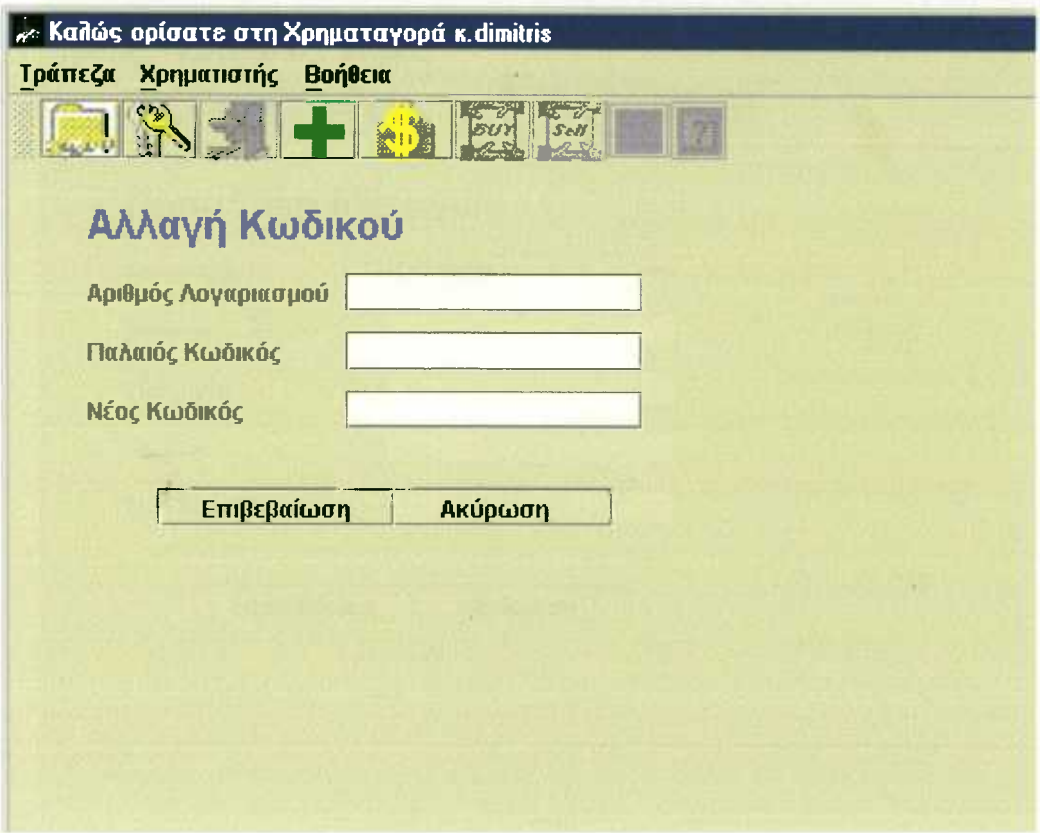
Σχήμα 27

Όπως αναφέρθηκε και προηγουμένως στην περίπτωση που ο χρήστης δεν έχει ήδη λογαριασμό απαραίτητη είναι η δημιουργία ενός νέου. Τα στοιχεία που δίνονται αφορούν το Όνομα, τη Διεύθυνση και την Πόλη κατοικίας, τον Ταχυδρομικό Κώδικα την Χώρα Διαμονής και το Τηλέφωνο του χρήστη. Όλα τα πεδία είναι τύπου String. Με την καταχώρηση που γίνεται δημιουργείται αυτόματα ένας αριθμός λογαριασμού. Σαν εξ' ορισμού κωδικός πρόσβασης θεωρείται το Όνομα του Πελάτη. Η Επιβεβαίωση που δέχεται ο πελάτης είναι της ακόλουθης μορφής.



Σχήμα 28

Οθόνη 5η

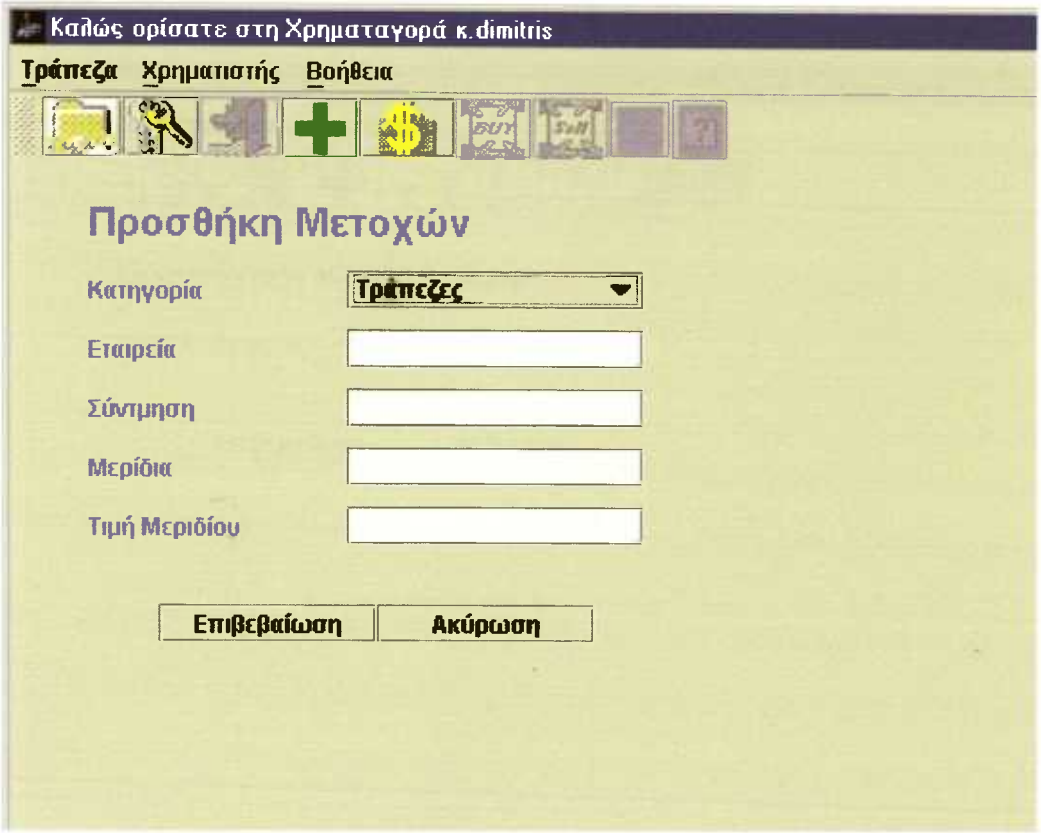


Σχήμα 29

Στην περίπτωση επιθυμίας αλλαγής του εξ' ορισμού κωδικού δίνεται η δυνατότητα αυτή στο χρήστη. Απαραίτητη είναι η συμπλήρωση όλων των πεδίων της φόρμας. Αν όλες οι προϋποθέσεις πληρούνται τότε ο χρήστης λαμβάνει την ακόλουθη επιβεβαίωση.

Σχήμα 30

Οθόνη 6η

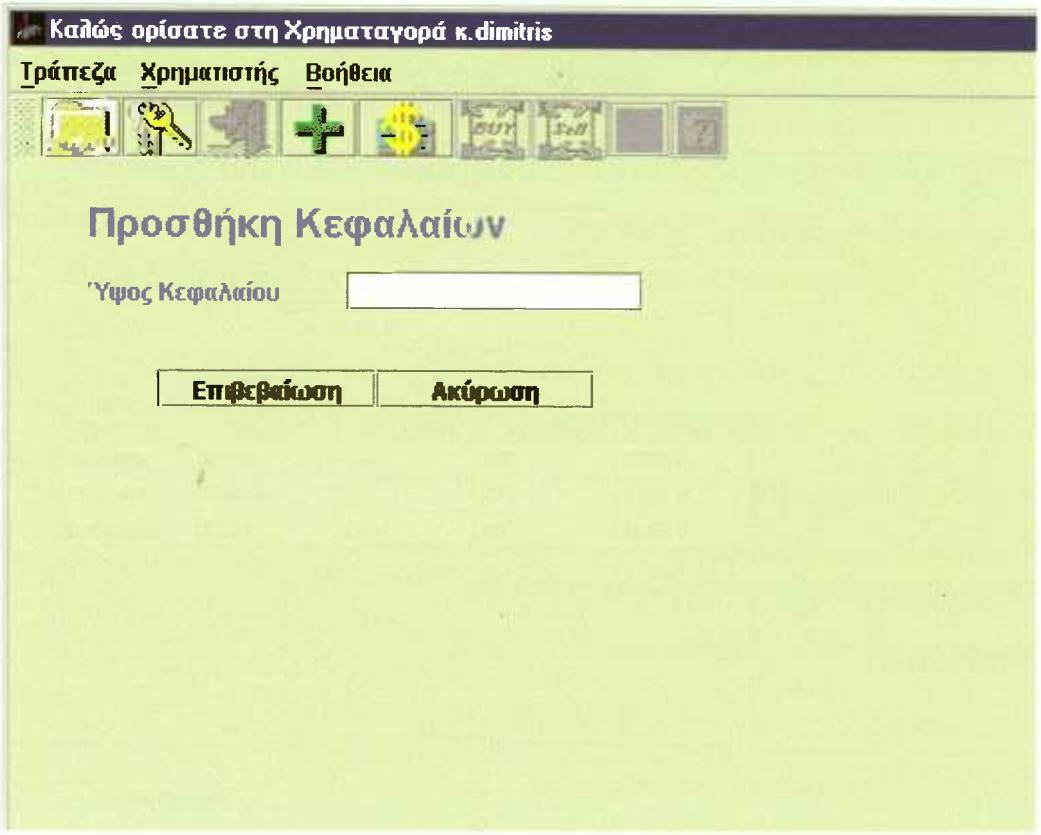


Σχήμα 31

Ο λογαριασμός του χρήστη μπορεί να εμπλουτισθεί με Μετοχές, οι οποίες προϋπήρχαν του λογαριασμού. Το πεδίο 'Κατηγορία' αποτελεί μια λίστα όλων των διαθέσιμων κατηγοριών μετοχών που διαπραγματεύονται στη χρηματαγορά. Απαραίτητη είναι η συμπλήρωση όλων των πεδίων. Τα πεδία 'Εταιρεία' και 'Σύντμηση' είναι τύπου String, ενώ τα πεδία 'Μερίδια' και 'Τιμή Μεριδίου' είναι τύπου Integer. Μετά την επιτυχή προσθήκη των μετοχών ο χρήστης λαμβάνει το ακόλουθο μήνυμα.

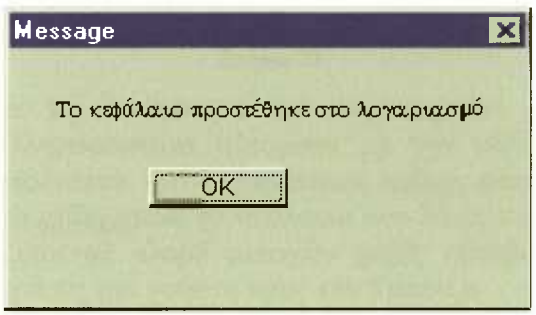
Σχήμα 32

Οθόνη 7η



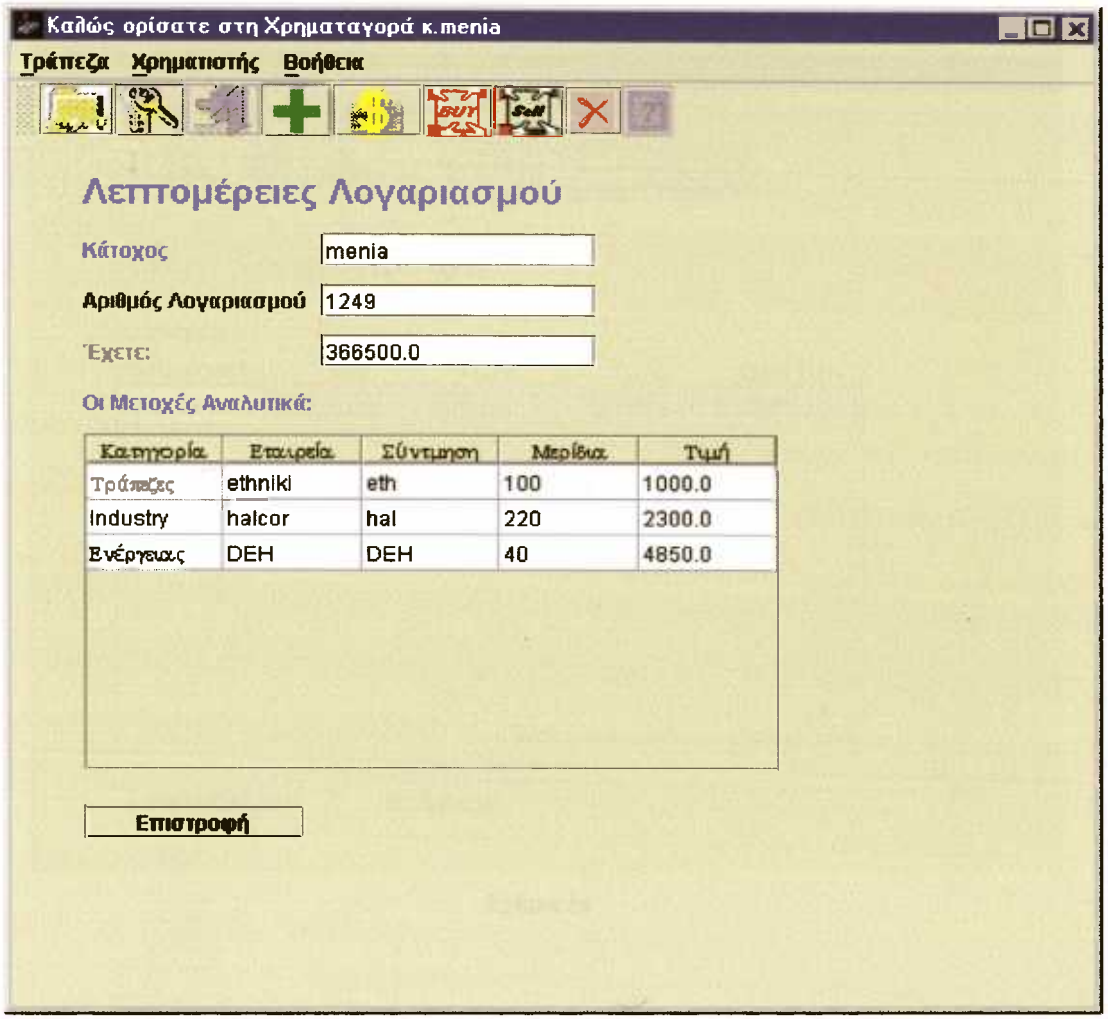
Σχήμα 33

Το ίδιο σκεπτικό ισχύει και για την προσθήκη κεφαλαίων. Ο χρήστης συμπληρώνει το ποσο που τον ενδιαφέρει να εισαγάγει στο λογαριασμο του προκειμένου αυτός να αγοράσει μετοχές στα πλαίσια της χρηματαγοράς. Η επιβεβαίωση ακολουθεί.



Σχήμα 34

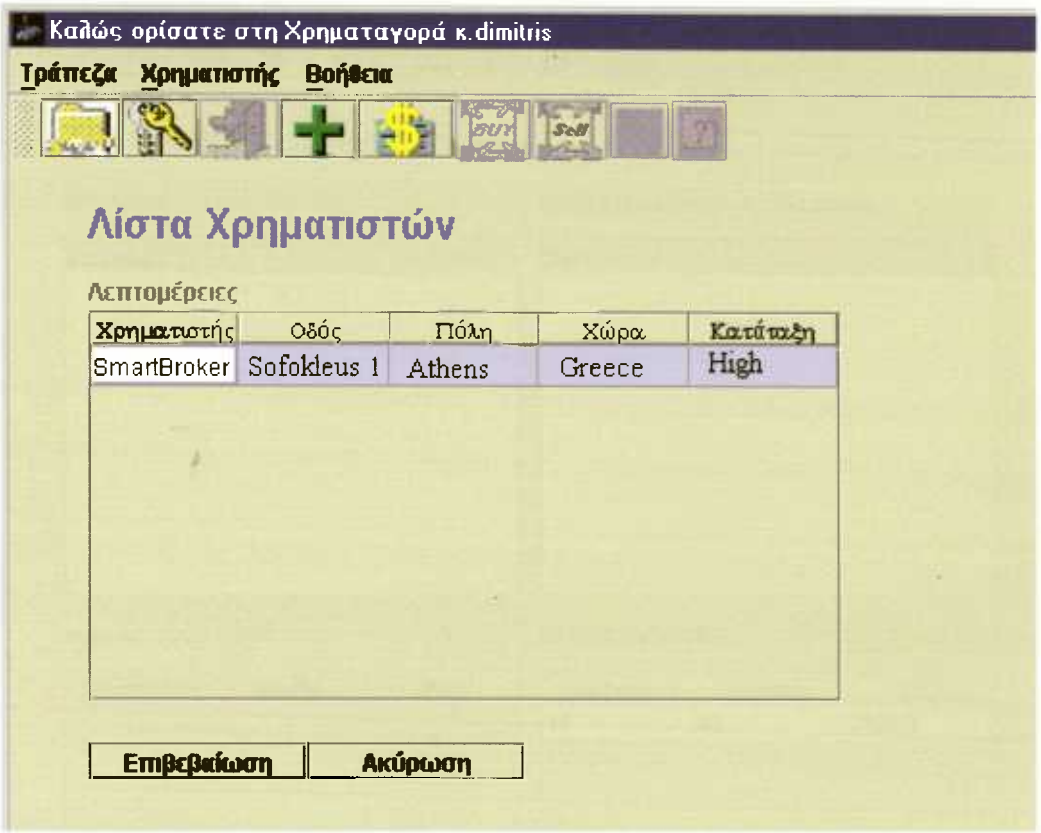
Οθόνη 8η



Σχήμα 35

Ο χρήστης έχει τη δυνατότητα να πληροφορηθεί για την σύνθεση του χαρτοφυλακίου του. Πληροφορείται αυτόματα με την κλήση της συγκεκριμένης οθόνης το ύψος του υπολοίπου του λογαριασμού καθώς επίσης αναλυτικά όλες τις μετοχές του. Υπάρχει το ενδεχόμενο το υπόλοιπο του λογαριασμού να είναι αρνητικό γεγονός που μεταφράζεται σε αγορά μετοχών χωρίς καταβολή του αντιτίμου και ταυτόχρονη εμφάνιση χρέους του χρήστη προς την Τράπεζα.

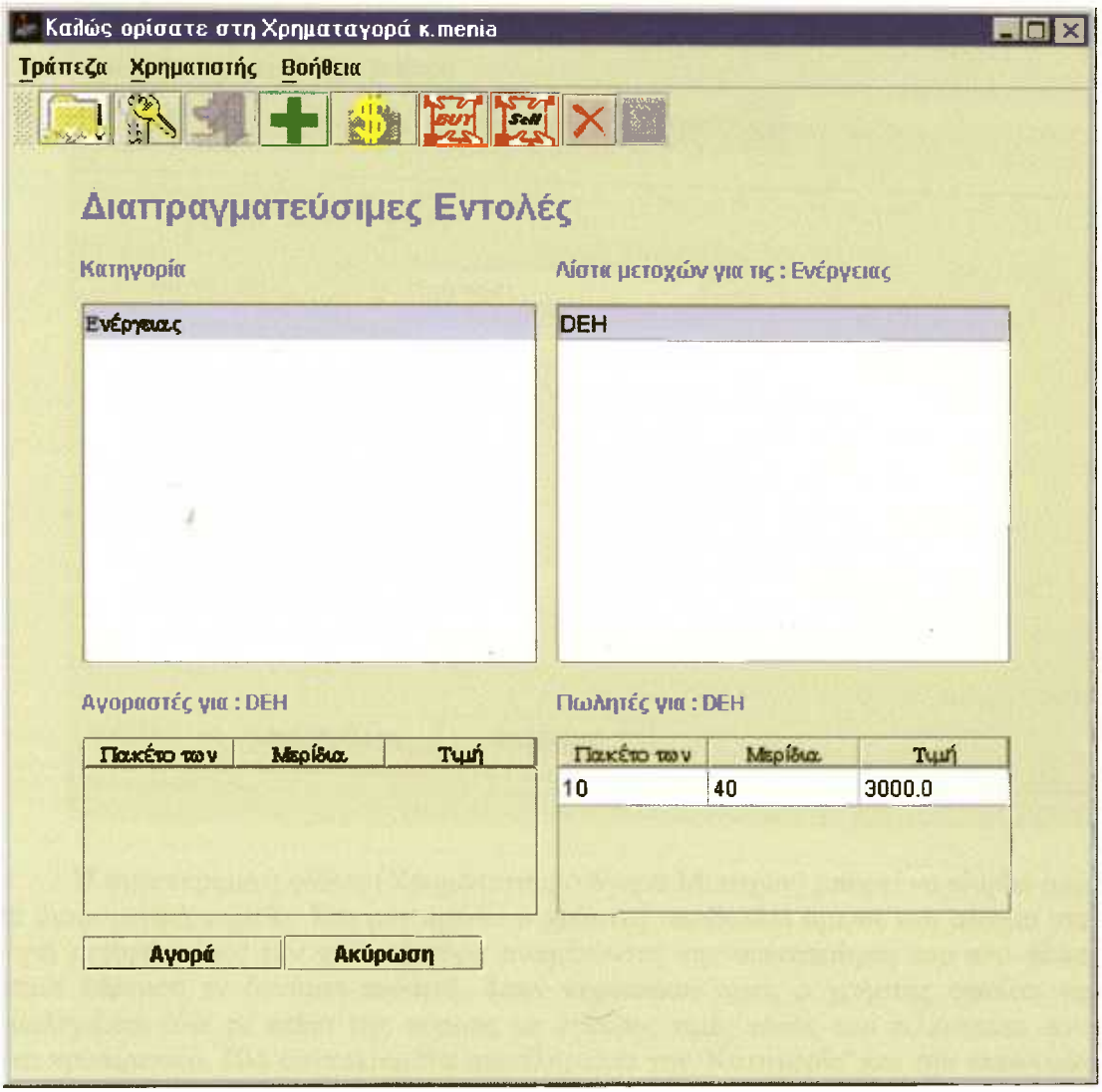
Οθόνη 9η



Σχήμα 36

Μέχρι στιγμής οι οθόνες που περιγράφηκαν αναφέρονται στην είσοδο του χρήστη στη χρηματαγορά και στην σύνδεση του με την τράπεζα. Στο εξής θα αναφερθούμε στην διαπραγμάτευση μετοχών και αξιών. Απαραίτητη προϋπόθεση είναι η επιλογή ενός από τους διαθέσιμους χρηματιστές, οι οποίοι όπως και οι τράπεζες έχουν κοινοποιήσει τα χαρακτηριστικά τους μέσω XML στον πυρήνα της εφαρμογής. Η επιλογή χρηματιστή ενεργοποιεί και τα χειριστήρια αγοραπωλησιών μετοχών.

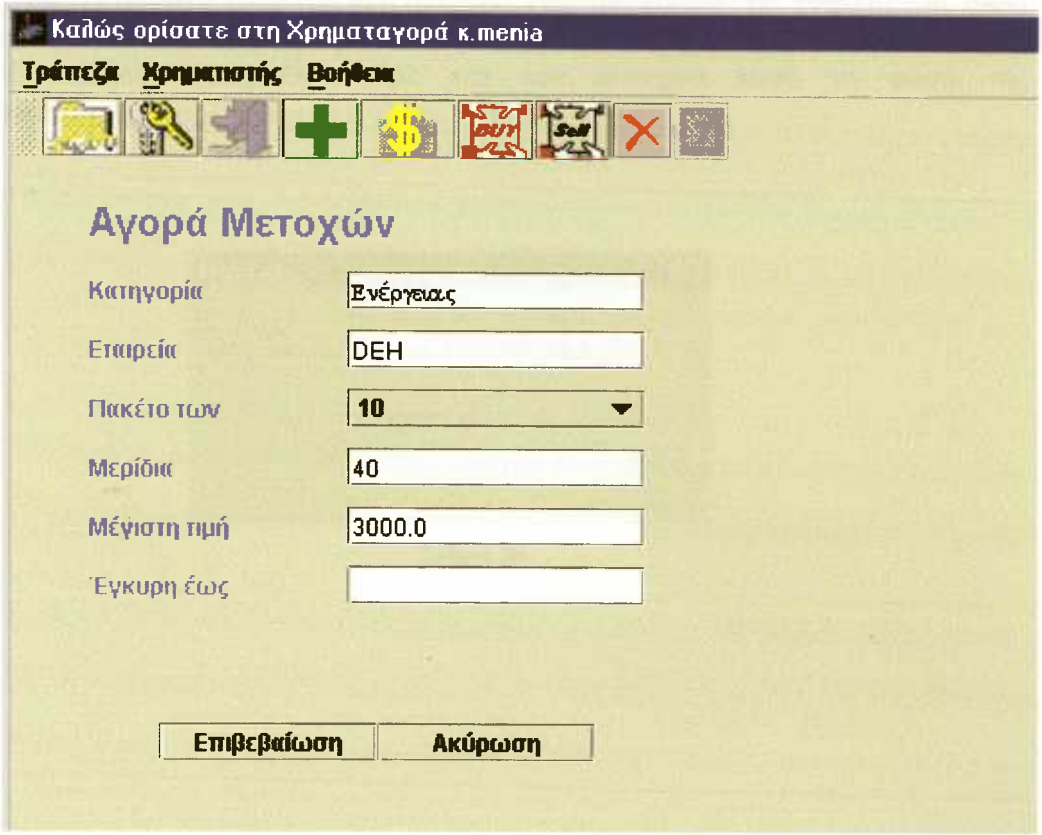
Οθόνη 10η



Σχήμα 37

Την κατάσταση που επικρατεί στην χρηματαγορά ο χρήστης της πληροφορείται από την παραπάνω οθόνη (Χρηματιστής -> Λίστα Μετοχών). Σε αυτή εκτίθενται όλες οι προσφερόμενες μετοχές είτε προς πώληση είτε προς αγορά. Υπάρχει μια κατηγοριοποίηση των μετοχών όπως αυτές προσφέρονται. Ο χρήστης πρέπει πρώτα να επιλέξει Κατηγορία. Αμέσως μετά επιλέγει μία από τις διαθέσιμες μετοχές της κατηγορίας . Με την επιλογή έχουμε την εμφάνιση των λεπτομερειών της μετοχής στο σχετικό πλαίσιο ανάλογα με το αν πρόκειται για αγορά ή για πώληση αντίστοιχα. Στο σημείο αυτό ο χρήστης μπορεί είτε να αποσυρθεί, είτε να προχωρήσει σε αγορά κάποιας μετοχής. Αν ισχύσει το δεύτερο σενάριο τότε ο χρήστης με το πάτημα του κομβίου 'Αγορά' μεταφέρεται αυτόματα στην οθόνη 'Αγορά Μετοχών' που περιγράφεται στη συνέχεια με συμπληρωμένα τα πεδία της συναλλαγής έτοιμος προς επιβεβαίωση ή μη της αυτής.

Οθόνη 11η



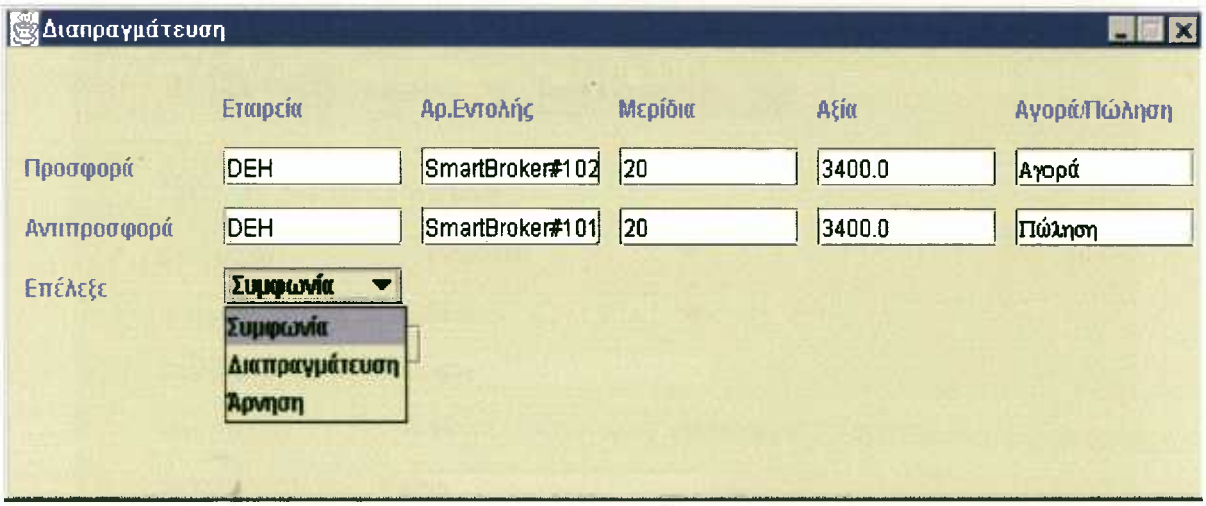
Η συγκεκριμένη οθόνη (Χρηματιστής->Αγορά Μετοχών) μπορεί να κληθεί από δυο διαφορετικά σημεία. Στο μεν πρώτο ο χρήστης υποβάλλει άμεσα ένα αίτημα για αγορά μετοχών προς την χρηματαγορά αναμένοντας την ικανοποίηση του από άλλο αίτημα κάποιου εν δυνάμει πωλητή. Στην περίπτωση αυτή ο χρήστης οφείλει να συμπληρώσει όλα τα πεδία της φόρμας με έγκυρες τιμές εκτός του τελευταίου που είναι προαιρετικό. Πιο συγκεκριμένα συμπληρώνει την 'Κατηγορία' και την επωνυμία της 'Εταιρείας' για την οποία ενδιαφέρεται, το εύρος του 'Πακέτου' στο οποίο θα υποδιαιρεθούν τα 'Μερίδια' του καθώς και την προσφερόμενη 'Μέγιστη Τιμή' που θα τεθεί σε διαπραγμάτευση με τις προσφορές των πωλητών. Τέλος συμπληρώνει το πεδίο 'Έγκυρη έως' για να δηλώσει το χρονικό περιθώριο μέσα στο οποίο θα ισχύει το ανωτέρω αίτημα. Χάριν λειτουργικότητας το πεδίο αυτό είναι προαιρετικό. Τα πεδία 'Κατηγορία', 'Εταιρεία' είναι τύπου String. Τα πεδία 'Μερίδια' και 'Μέγιστη Τιμή' είναι τύπου Integer και Float αντίστοιχα. Το πεδίο 'Έγκυρη έως' είναι τύπου date.

Το δεύτερο σημείο απ' όπου μπορεί να κληθεί η συγκεκριμένη οθόνη αποτελεί συνέχεια της προηγούμενης (Οθόνη 10). Ο χρήστης στην περίπτωση αυτή έχει ενημερωθεί για τα συμβαίνοντα στην χρηματαγορά και κινητοποιείται να αγοράσει απευθείας. Οι λεπτομέρειες της μετοχής που περιγράφονται στην οθόνη 10 μεταφέρονται στα αντίστοιχα πεδία της τρέχουσας φόρμας. Ο χρήστης μόνο συμπληρώνει το τελευταίο πεδίο 'Έγκυρη έως', αν φυσικά συμφωνεί με τα χαρακτηριστικά της μετοχής (συγκεκριμένα την τιμή της). Ο χρήστης μπορεί να θέσει τη δική του προσφορά για την μετοχή αρκεί αυτή να μην ξεπερνά ένα προκαθορισμένο όριο απόκλισης (0,15 της αρχικής προσφοράς). Μετά την αποστολή της προσφοράς και εφόσον υπάρξει ταύτιση με την προσφερόμενη μετοχή ενεργοποιείται η επόμενη

οθόνη η οποία διεξάγει την διαπραγμάτευση ανάμεσα στον αγοραστή και στον πωλητή της μετοχής. Ο χρήστης λαμβάνει στη συνέχεια ένα "απόκομμα" της εντολής του, όπου περιλαμβάνεται ο αύξων αριθμός της και το όνομα του Χρηματιστή που έχει επιλέξει. Αυτό διευκολύνει όπως θα δούμε και στη συνέχεια κατά τη φάση της διαπραγμάτευσης.

Σχήμα 38

Οθόνη 12η



Σχήμα 39

Η συγκεκριμένη οθόνη είναι επιφορτισμένη με την διαδικασία της διαπραγμάτευσης μεταξύ των εμπλεκομένων μερών, αγοραστών και πωλητών. Με τον εντοπισμό ταύτισης μεταξύ προσφοράς και ζήτησης εγείρεται αυτόματα ένα αίτημα διαπραγμάτευσης που εκφράζεται μέσα από την συγκεκριμένη οθόνη. Στην πρώτη σειρά εμφανίζεται η εντολή που ενέκυψε τελευταία. Σε ένα αποκεντρωμένο περιβάλλον πολλαπλών πυρήνων κάθε εμπλεκόμενος χρήστης έχει στη διάθεση του την συγκεκριμένη οθόνη. Ο χρήστης της "πρώτης γραμμής" έχει την δυνατότητα να αλλάξει τα μερίδια αλλά και την αξία τους, και στη συνέχεια να επιλέξει μεταξύ "Συμφωνίας", "Διαπραγμάτευσης" και "Άρνησης" στέλνοντας τις προθέσεις του στον πωλητή/αγοραστή της διαπραγματευόμενης μετοχής. Η απάντηση φθάνει στον αποδέκτη της με την ίδια μορφή παραθύρου με ανεστραμμένους όρους πλέον. Αυτός που πρέπει να απαντήσει βρίσκεται πλέον στην πρώτη γραμμή θέτοντας τις δικές του προτιμήσεις και αποστέλλοντας τες δηλώνοντας πρόθεση "Συμφωνίας", "Διαπραγμάτευσης" ή "Άρνησης". Αν η διαπραγμάτευση στεφθεί με επιτυχία τότε η διαδικασία της διαπραγμάτευσης τερματίζεται και οι δύο αποκομίζουν τα αναμενόμενα από την συναλλαγή. Ο Αγοραστής τις μετοχές στο χαρτοφυλάκιο του με ταυτόχρονη μείωση του αποθεματικού του, ενώ ο Πωλητής το συμφωνηθέν ποσό και την αντίστοιχη μείωση του χαρτοφυλακίου του. Αν η διαπραγμάτευση εξακολουθήσει να υφίσταται η διαδικασία θα συνεχιστεί έως ότου επέλθει συμφωνία ή άρνηση.

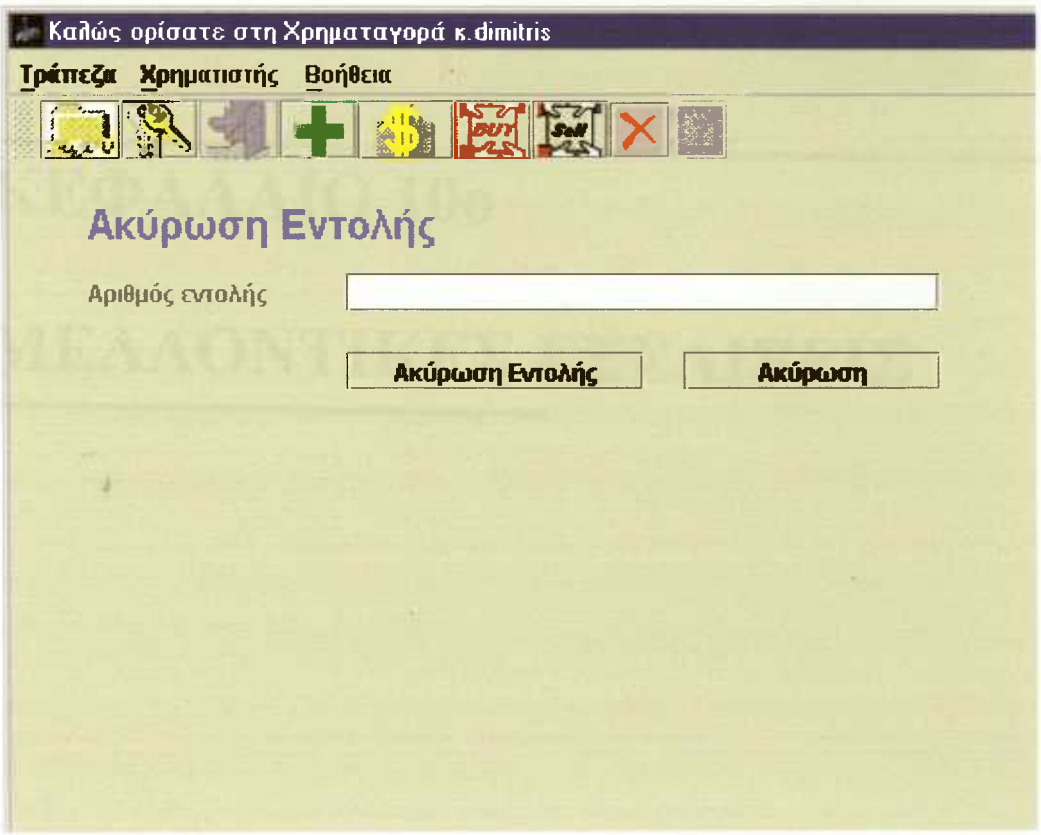
Οθόνη 13η

Σχήμα 40

Μέσω της συγκεκριμένης οθόνης ο χρήστης προβαίνει σε πώληση μετοχών που βρίσκονται στην κατοχή του. Επιλέγει από ένα προεπιλεγμένο σύνολο την κατηγορία μετοχής που τον ενδιαφέρει, καθώς και την εταιρεία. Οι εταιρείες που εμφανίζονται στην λίστα είναι αυτές που κατέχει στο χαρτοφυλάκιο του. Για να ολοκληρωθεί το αίτημα πώλησης ο χρήστης περιγράφει το εύρος του πακέτου πώλησης, τα μερίδια που θέτει σε διαπραγμάτευση καθώς και την ελάχιστη τιμή που είναι αποδεκτή. Τέλος αναφέρει την ημερομηνία μέχρι την οποία το αίτημα θα βρίσκεται σε ισχύ. Το πεδίο αυτό δεν είναι υποχρεωτικό. Με την επιβεβαίωση της πώλησης ο χρήστης λαμβάνει στη συνέχεια ένα "απόκομμα" της εντολής του, όπου περιλαμβάνεται ο αύξων αριθμός της και το όνομα του Χρηματιστή που έχει επιλέξει. Αυτό διευκολύνει όπως θα δούμε και στη συνέχεια κατά τη φάση της διαπραγμάτευσης.

Σχήμα 41

Οθόνη 14η



Σχήμα 42

Τέλος κλείνοντας την παρουσίαση του τμήματος αυτού, με την παρούσα οθόνη έχουμε την δυνατότητα ακύρωσης κάποιας από τις εντολές διαπραγμάτευσης που ο χρήστης έχει υποβάλλει. Μοναδικό πεδίο που πρέπει να συμπληρωθεί είναι ο αριθμός της εντολής που ο χρήστης λαμβάνει στο απόκομμα της συναλλαγής είτε πρόκειται για πώληση είτε για αγορά.

ΚΕΦΑΛΑΙΟ 10ο

ΜΕΛΛΟΝΤΙΚΕΣ ΕΞΕΛΙΞΕΙΣ

10 ΜΕΛΛΟΝΤΙΚΕΣ ΕΞΕΛΙΞΕΙΣ

Τα όσα αναφέραμε μέχρι στιγμής προδιαγράφουν ένα πολύ έντονο μέλλον μιας και η φάση που διερχόμαστε ως μεταβατική μπορεί να χαρακτηριστεί. Βρισκόμαστε μεταξύ δύο φάσεων που αλληλοδιαδέχονται σταδιακά η μία την άλλη. Έως τώρα, η επέκταση και η επισημοποίηση του Internet ως μέσου, αποτέλεσε τη βασική επιδίωξη των δυνάμεων που το μετασχημάτισαν και το ενδυνάμωσαν ταυτόχρονα. Στο εξής και αφού η γνωριμία επιτευχθεί, το μέσο αλλάζει σταδιακά μορφή μετεξελισσόμενο σε αδιαμφισβήτητο μοχλό επιχειρηματικής δράσης. Λόγω του μεταιχμίου στο οποίο βρίσκεται, εύλογο είναι να εμφανίζονται κλυδωνισμοί έως ότου το τοπίο να ξεκαθαρίσει και νέα πρότυπα, πρωτόκολλα, τεχνολογικές προτάσεις υπερισχύσουν.

Χωρίς να φιλοδοξώ να παίξω τον ρόλο του μάντη μιας και κάτι τέτοιο στον χώρο της πληροφορικής είναι σχεδόν ακατόρθωτο, στη συνέχεια με βάση τις συνθήκες που έχουν διαμορφωθεί αλλά και με το μοντέλο των υπηρεσιών, θα προσπαθήσω να αναδείξω κάποια λεπτά σημεία στην περαιτέρω εξέλιξη στον χώρο των επιχειρήσεων-οργανισμών και του Internet.

Το όραμα για μια παγκοσμιοποιημένη αγορά στα πλαίσια του Internet, χωρίς περιορισμούς για ελεύθερη επικοινωνία ανάμεσα στους πάντες και τα πάντα, πιστεύω ότι θα αναβληθεί για το μέλλον. Αυτό που θα μας απασχολήσει άμεσα είναι η δημιουργία κάθετων Portals, στραμμένα στις ανάγκες ενημέρωσης και συναλλαγής συγκεκριμένων ομάδων. Ήδη έχει αρχίσει να παρατηρείται μια μεταστροφή στα υπό διαμόρφωση portals - κυρίως στο εξωτερικό, όπου η ιδέα είναι ωριμότερη- από γενικού ενδιαφέροντος και ποικίλης ύλης, σε εξειδικευμένου ενδιαφέροντος Portals. Έτσι, μεγάλοι παίκτες της αγοράς στους τομείς της υγείας, της τεχνολογίας, της μεταποίησης και των οικονομικών υπηρεσιών θα στραφούν στη προσπάθεια εγκαθίδρυσης κάθετων portals.

Άμεση θα είναι η μετεξέλιξη αυτών των portals σε B2B αγορές, οι οποίες θα προσπαθήσουν να φέρουν κοντά αγοραστές και πωλητές, συγκεκριμένων προϊόντων και συγκεκριμένων γεωγραφικών περιοχών, με σκοπό το εμπόριο. Συνήθως ο ρόλος τους, τουλάχιστον στην αρχή, θα είναι διαμεσολαβητικός χωρίς πάντα να υποστηρίζουν συναλλαγές. Σαφώς θα υπάρξει εξειδίκευση, όσον αφορά την υιοθέτηση του B2B ή του B2C μοντέλου. Η από κοινού δράση και στα δύο, θεωρείται μάλλον απίθανη. Αυτό που αναμένεται να είναι το επιπλέον στοιχείο αυτών των αγορών είναι η διαμόρφωση ενός δυναμικού πεδίου στο χώρο των τιμών. Σταδιακά θα αρχίσουν να διαμορφώνονται συνθήκες δια-επιχειρησιακών και ενδο-επιχειρησιακών συναλλαγών (logistics, οικονομικές υπηρεσίες κ.λ.π) τις οποίες οι αγορές θα κληθούν να ικανοποιήσουν.

Αποτέλεσμα όλων αυτών θα είναι η αλλαγή των σχέσεων μεταξύ των επιχειρήσεων από τις κλασσικές παραδοσιακές σχέσεις, σε νέες που θα πλησιάζουν την έννοια του οικοσυστήματος των αγορών. Η αλλαγή στα πρώτα της στάδια θα αφήσει εκτός πολλούς οργανισμούς οι οποίοι δεν θα κατανοήσουν έγκαιρα το νέο σκηνικό που διαμορφώνεται με αξιοσημείωτες επιπτώσεις. Μακροπρόθεσμα, αυτό θα γίνει αντιληπτό με αποτέλεσμα όλο και περισσότερες επιχειρήσεις να συμμετέχουν σε τέτοιου είδους ηλεκτρονικές αγορές, είτε ως πωλητές, είτε ως αγοραστές. Μια άμεση επίδραση της αλλαγής θα είναι η μεγάλη αύξηση της δραστηριοποίησης των εταιριών

παραγωγής λογισμικού για την ικανοποίηση των αναγκών της αυξανόμενης ζήτησης για ηλεκτρονικές αγορές με αλυσιδωτές συνέπειες (οικονομικές, τεχνολογικές κ.λ.π).

Μεγάλο ρόλο αναμένεται να παίξουν οι ASP's, στο υπό διαμόρφωση σκηνικό. Οι Application Service Providers, θα αποτελέσουν βασικό στοιχείο για την εξέλιξη ή μη του μοντέλου των υπηρεσιών. Η συγκεκριμένη δομή ανάπτυξης θα είναι από τις βασικότερες τα προσεχή χρόνια. Η καθετοποίηση στην οποία στοχεύουν οι ASP's σε συνδυασμό με τις ηλεκτρονικές αγορές που προαναφέραμε, θα έχουν σαν αποτέλεσμα την παροχή εξειδικευμένων υπηρεσιών για κάθε τομέα. Παρόλα αυτά, μόνο λίγοι ASP's θα είναι αυτοί που θα κυριαρχήσουν, μιας και τα ρευστά υπό διαμόρφωση μοντέλα που υπάρχουν σήμερα δεν ξεκαθαρίζουν απόλυτα το τοπίο. Τέλος, το μόνο σίγουρο είναι ότι:

- μεγάλα ποσά πρόκειται να επενδυθούν από πολλούς
- μόνο λίγοι θα κατορθώσουν να επιτύχουν τους στόχους τους, ιδίως αυτοί που θα στραφούν στη προτυποποίηση των διαδικασιών και την προσαρμογή στις ανάγκες των πελατών.

Το μοντέλο που σχετίζεται με τους ASP's είναι αυτή τη στιγμή υπό διαμόρφωση. Κύριο στοιχείο αποτελεί η ηλεκτρονική υπηρεσία. Άμεσα θα δούμε τη χρήση πλέον του λογισμικού ως υπηρεσία. Έτσι, πολλοί οργανισμοί αντί να προμηθεύονται το λογισμικό όπως συμβαίνει μέχρι στιγμής, θα το μισθώνουν και θα το χρησιμοποιούν ως υπηρεσία. Η υπηρεσία αυτή θα έχει πάντα προς παραχώρηση στους πελάτες τις τελευταίες εκδόσεις με όλες τις διορθώσεις ή αναβαθμίσεις που συνεχίζουν να γίνονται μετά την αρχική παράδοση του λογισμικού. Παρά την τρέχουσα αβεβαιότητα, το μοντέλο που μόλις αναφέρθηκε αντιπροσωπεύει μια πλήρως διαφορετική υπόσταση στον χώρο της πληροφορικής. Για την επικράτησή του απαιτείται σαφώς μακρύς χρονικός ορίζοντας, μιας και οι αλλαγές που απαιτούνται τόσο τεχνολογικά, όσο και επιχειρηματικά είναι πολλές και πολυδάπανες.

Δύο είναι οι βασικοί τομείς στους οποίους θα αναπτύξουν δράση οι ASP's μακροπρόθεσμα.. Αρχικά και όπως έχει αρχίσει να διαφαίνεται θα λειτουργήσουν ως παροχείς υποδομής για πλήρως outsourced web sites. Το γεγονός αυτό δίνει στους εκμισθωτές αυτών των υπηρεσιών με πολύ μικρότερο κόστος τη δυνατότητα χρήσης καινοτομικών εφαρμογών με παράλληλη αποφυγή της επιπλέον πολυπλοκότητας. Το κόστος μιας τέτοιας υποδομής θα είναι υψηλό και κατ' επέκταση οι χρεώσεις θα είναι και αυτές υψηλές. Παρόλα αυτά, με την εμφάνιση νέων παικτών, το κόστος αυτό σταδιακά θα μειωθεί. Ένα άλλο στοιχείο που επιτυγχάνει μια επιχείρηση προσφεύγοντας στη λύση αυτή είναι η αποφυγή του μεγάλου ρίσκου που συνεπάγεται η εσωτερική ανάπτυξη ενός ERP συστήματος για παράδειγμα, που στο εξής θα μπορεί να μισθώνεται σαν υπηρεσία.

Ο δεύτερος τομέας δράσης αναμένεται να είναι ο χώρος παροχής ERP υπηρεσιών για μικρές και μεγάλες επιχειρήσεις. Το κόστος ανάπτυξης ενός ERP συστήματος στα πλαίσια μιας μικρής ή μεσαίας επιχείρησης είναι ομολογουμένως μια δαπανηρή διαδικασία στο σύνολο της, γεγονός που απέτρεπε μέχρι τώρα πολλές επιχειρήσεις στο να προβούν σε μια τέτοια απόφαση εισαγωγής ERP συστημάτων στην παραγωγική τους διαδικασία. Αυτό που επιχειρείται μέσω του μοντέλου των υπηρεσιών, είναι ο μετασχηματισμός των κατάλληλων ERP εφαρμογών σε μορφή υπηρεσιών άμεσα ενεργοποιήσιμων και χρησιμοποιήσιμων μέσω δικτύου. Παράλληλα με αυτή τη προσπάθεια υλοποίησης, αναγκαία είναι η ανάπτυξη κατάλληλων

μοντέλων, αλλά και λογισμικού χρέωσης αυτού του είδους των υπηρεσιών. Σύμφωνα με τις εκτιμήσεις αυτές, στο προσεχές μέλλον οι περισσότερες από τις μικρές ή μεσαίες επιχειρήσεις θα κινηθούν προς τους αντίστοιχους ASP's κυρίως εξαιτίας του μεγάλου κόστους εσωτερικής ανάπτυξης εφαρμογών, καθώς επίσης και λόγω της έλλειψης εξειδικευμένου προσωπικού για τη λειτουργία των ηλεκτρονικών εφαρμογών που υπάρχουν.

Ένας χώρος όπου υπάρχουν μεγάλα περιθώρια δραστηριοποίησης κυρίως λόγω της μεγάλης οικονομικής σημασίας που παρουσιάζει από πλευράς εταιριών λογισμικού, είναι αυτός του λεγόμενου JIT (Just-In-Time) λογισμικού. Παραδοσιακά το λογισμικό αντιμετωπίζεται σαν ένα τυποποιημένο προϊόν το οποίο απαιτεί εγκατάσταση, συντήρηση και αναβάθμιση βάση ενός προδιαγεγραμμένου προγράμματος. Εξαιτίας όμως του μεγάλου κόστους διανομής, αλλά πολύ περισσότερο εξαιτίας των διαφυγόντων εσόδων λόγω πειρατείας κυρίως, οι εταιρίες παραγωγής λογισμικού στρέφονται στην υιοθέτηση του μοντέλου των υπηρεσιών χρεώνοντας τους πελάτες τους σε μία συνδρομητική βάση.
Αυτό που αναμένεται είναι μια έκρηξη στον χώρο του λογισμικού, αλλάζοντας ριζικά το τοπίο το οποίο έχει διαμορφωθεί. Η παρεχόμενη υποδομή του WEB θα αποτελέσει τη βασική πλατφόρμα διάθεσης και συντήρησης λογισμικού.

Μια άλλη οντότητα που θα κάνει χρήση του μοντέλου τών υπηρεσιών είναι αυτή του Management Service Provider (MSP). Ο MSP παρέχει υπηρεσίες management για δίκτυα, συστήματα, εφαρμογές καθώς και σε υποδομές ηλεκτρονικού εμπορίου, δια μέσου δικτύου σε πολλαπλές επιχειρήσεις ταυτόχρονα, χρησιμοποιώντας το μοντέλο της πληρωμής σύμφωνα με την χρήση. Σαν οντότητα εμφανίστηκε για πρώτη φορά φέτος στην Αμερική, ενώ αναμένεται να συναντήσει μεγάλο ενδιαφέρον καθώς οι χρηματοδότες των Venture Capital τους συμπεριλαμβάνουν στις βασικές προτεραιότητες τους. Η αγορά των MSP's είναι σαφώς ανώριμη προς το παρόν, παρόλα αυτά στοχεύει στη διαχείριση δικτύων και συστημάτων επόμενης γενιάς. Η εξάπλωση τους αναμένεται να είναι ραγδαία, μιας και η εξάπλωση των ASP's προϋποθέτει την ύπαρξη MSP's. Ένας από τους λόγους οι οποίοι επιβάλλουν την υιοθέτηση τέτοιων μοντέλων (ASP's και MSP's) σύμφωνα με μελέτες, είναι η έλλειψη καταρτισμένου προσωπικού για εξειδικευμένες εφαρμογές, έλλειψη η οποία θα διογκωθεί στα επόμενα χρόνια.

Άμεσες θα είναι οι επιπτώσεις που θα παρατηρηθούν στην αλυσίδα αξίας. Η συνεργασία μεταξύ των μελών μιας B2B αγοράς θα αναδειχθεί σε βασικό στοιχείο. Σύμφωνα με μια επίσημη στατιστική, αναμένεται να δημιουργηθούν περίπου 3000 B2B ηλεκτρονικές αγορές. Από αυτές το 5% θα επιτύχει, το 15% θα συγχωνευθεί, ενώ το υπόλοιπο 80% θα αποτύχει. Αυτό που θα μας απασχολήσει προσεχώς, είναι το ονομαζόμενο c-commerce (collaborative commerce), το οποίο είναι ένα σύνολο συνεργατικών αλληλεπιδράσεων μεταξύ μιας επιχείρησης, των πελατών της, των εταίρων, των προμηθευτών και των εργαζομένων της. Αυτή η προσέγγιση του c-commerce διαφέρει από αυτή του e-commerce στο σημείο θεώρησης ως κέντρου του μοντέλου την συνεργασία και όχι την συναλλαγή αγοράς και πώλησης προϊόντων που πρεσβεύει το e-commerce. Το νέο που αναμένεται να προσδώσει το c-commerce είναι η έμφαση στην εξάπλωση των ευκαιριών και όχι στη βελτιστοποίηση της συναλλαγής. Το c-commerce, όχι μόνο θα αλλάξει την φύση των δια-επιχειρησιακών σχέσεων, αλλά θα επαναπροσδιορίσει τις επιχειρηματικές διαδικασίες και τους λειτουργικούς ρόλους.

Οι επιχειρήσεις που θα υιοθετήσουν πιο γρήγορα αυτά τα νέα οργανωτικά σχήματα, θα αποκτήσουν και το ανταγωνιστικό πλεονέκτημα στην αγορά.

Τα ενδο-επιχειρησιακά portals είναι μια από τις παραμέτρους όπου το μοντέλο των υπηρεσιών βρίσκει εφαρμογή. Το μεγαλύτερο μέρος των εταιριών έχουν μπει στη διαδικασία δημιουργίας ενός τέτοιου portal με σκοπό την προσπελασιμότητα του από όλο το φάσμα της επιχείρησης, αλλά και σαν δίοδος για την πληροφόρηση μέσω Internet.

Η έννοια της δυναμικής διαπραγμάτευσης είναι ένα άλλο σημαντικό στοιχείο που χαρακτηρίζει το μοντέλο των συνεργαζόμενων ηλεκτρονικών υπηρεσιών και που θα αποτελέσει καθοριστικό παράγοντα στο μέλλον. Η δυναμική διαπραγμάτευση θα δώσει περαιτέρω ώθηση στους αγοραστές να βρουν την καταλληλότερη συμφωνία σε όρους τιμής, παράδοσης και ποιότητας δημιουργώντας μια μετακίνηση δυνάμεων από τον πωλητή στον αγοραστή υπηρεσιών. Το μοντέλο των υπηρεσιών θα παραχωρήσει επιπλέον διευκολύνσεις στις διάφορες επιχειρήσεις να κάνουν outsource πολλές από τις εσωτερικές λειτουργίες πληροφορικής, επιτυγχάνοντας έτσι όλα τα παρεχόμενα οφέλη από τους εξωτερικούς παροχείς υπηρεσιών. Ένα περιβάλλον όπου η έννοια της υπηρεσίας θα είναι πρωτεύουσας σημασίας, θα δημιουργήσει την ανάγκη για νέα προσόντα στον χώρο της πληροφορικής που έχουν να κάνουν με τη διαμεσολάβηση, την αξιολόγηση, την διαχείριση συμβολαίων με προμηθευτές, οικονομικές γνώσεις κ.α. Επίσης, μέσω της δυναμικής διαμεσολάβησης θα επικρατήσει ένα μοντέλο ισορροπίας τιμών στηριζόμενο κυρίως στον ανταγωνισμό, επιτρέποντας την εισαγωγή ευέλικτων οικονομικών μοντέλων και στον χώρο του IT.

ΒΙΒΛΙΟΓΡΑΦΙΑ

11 ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Arthur B. Increasing returns and the world of Business - Harvard Business Review (14-AUG-1996)

[2] Bailey J. and Bakos Y. An Explanatory Study of the emerging Role of electronic intermediaries. International journal of Electronic Commerce (Spring 1997)

[3] ([A model for the e-service marketplace Anna Durante, David Bell, Louis Goldstein HP Laboratories Palo Alto])
[4] Joel Birnbaum "Physics and the Information Revolution" speech to American Physical Society's 1999Contennial Conference.

[5] "E-speak Hewlett Packard' s Open Internet Services Platform An IDC WhitePaper" by Frank Gens and Anna Giraldo.

[6] Assesing the Impact of Jini in the Enterprise. Yankee Group February 1999

[7] Sun unbottles Jini. Dataquest March 15,1999

[8] The Emerging Role of Electronic Marketplaces on the Internet. Yannis Bakos-Communications of the ACM Aug 1998/Vol.41 No8

[9] Interorganizational Information Systems. Strategic Implications for Competition and coperation. Ph.D dissernation Sloan Scholl of Management Massachusetts Institute of Technology 1987

[10] "Business-to-Business Electronic Commerce: Exploiting Market Opportunities in the Extranet Age", Datamonitor 1997

[11] Jelassi T. Lai H-S "CitiusNet: The Emergence of a global Electronic market", Society for Information Management 1996.

[12] Mougayar W. "Opening Digital Markets, Advanced strategies for Internet-driven Commerce", Cybermanagment Publications 1997 pp201-211

[13] Electronic Service Markets by Michael Merz Univercity of Hambourg Germany

[14] E-speak Architecture Spesifications

[15] Brokering Strategies in Electronic Markets. Arte Sagev, Garrie Bearni

[16] E-speak Programming Guide

[17] Java 1.2 Developers Guide. Jamie Jaworski

[18] Thinking in Java,2 nd Edition, Release 11by Prentice-Hall mid-June, 2000
Bruce Eckel, President,MindView, Inc.

[19] Andersen Consulting, 2000. Understanding e-Markets.Research Note. ECommerce Networks, Issue Five. Institute of strategic change

[20] Robertson Stephens, 2000 B2B: Building Technology Bridges Outside the Four Wall of the Enterprice

[21]Rangaswamy A et al., 2000 Strategic Thinking of the Digital Economy

[22] Webber D., 1998. Introduction XML/EDI Frameworks. Elumen Solutions. Electronic Markets Vol .8. No.1.

ΧΡΗΣΙΜΑ URLs

[24] Web page: *BizTalk* URL: http://biztalk.org

[25] Web page: *Hewlett-Packard's E-Service*s. URL: http://www.hp.com/e-services/

[26] http://research.microsoft.com/research/os/millennium/mgoals.html

[27] Web page: *XML, Java, and the future of the We*b. URL: http://metalab.unc.edu/pub/sun-info/standards/xml/why/xmlapps.htm

[29] Gillmor, S., Get a JumpStart on BizTalk Server *Enterprise Developmen*t, http://www.enterprisedev.com/upload/free/features/entdev/1999/12dec99/sg1299/sg1299.asp

[30] Extended Mark-up Language XML, http://www.xml.org

[31] Simple Object Protocol, SOAP, http://msdn.microsoft.com./xml/general/soap_v09.asp

[32] http://b2b.ebizq.net/ebiz_integration/lawton_1.html

[33] http://e-commerce.com

[34] http:// fuji.stanford.edu/jguide

[35] www.b2business.net

[36] www netacadeny.org/netacademy/publications.nsf

[37] www commerce.net/resources

[38] www internetwk.com

[39] www cio.com

[40] www epoly.polymtl.ca/internet/fpublicat.html

[41] www ispo.cec.be/ecommerce/answears/introductions

[42] www.electronicmarkets.org/netacademy
[43] users.pandora.be/paul.timmers

ΠΑΡΑΡΤΗΜΑ

Παράρτημα

1. BankServiseIntf.esidl

```
package shareBroker;
import net.espeak.jesi.ESService;
import net.espeak.jesi.*;
import net.espeak.infra.cci.exception.ESInvocationException;
import net.espeak.util.*;
import java.util.*;

public interface BankServiceIntf extends ESService
{
   public String openAccount(String user, Address address)
      throws ESInvocationException;
   public boolean changePassword(String accountNumber,
      String oldPassword,String newPassword)
      throws ESInvocationException;
   public void addShare(String accountNumber, ShareDetailParam share)
      throws ESInvocationException;
   public void addCash(String accountNumber, float cash)
      throws ESInvocationException;
   public float listBalance(String accountNumber)
      throws ESInvocationException;
   public String getAccountName(String accountNumber)
      throws ESInvocationException;
   public ESArray listShares(String accountNumber)
      throws ESInvocationException;
   public ESArray listTransactions(String accountNumber)
      throws ESInvocationException;
   public void closeTransaction(OrderDetailParam order,
      OrderDetailParam matchWith) throws ESInvocationException;
   public String login(String accountNumber, String password)
      throws ESInvocationException;
}
```

2. BankServiseIntf.java

```
// Αρχείο παραγόμενο από την μεταγλώτισση του BankServiceIntf.esidl μέσω του IDL compiler
package shareBroker;
import net.espeak.jesi.ESService;
import net.espeak.jesi.*;
import net.espeak.infra.cci.exception.ESInvocationException;
import net.espeak.util.*;
import java.util.*;
import net.espeak.jesi.ESService;
import net.espeak.infra.cci.exception.ESInvocationException;


public interface BankServiceIntf
extends ESService {
   public String openAccount(String user, Address address) throws  ESInvocationException;
   public boolean changePassword(String accountNumber, String oldPassword, String newPassword)
throws  ESInvocationException;
   public void addShare(String accountNumber, ShareDetailParam share) throws
ESInvocationException;
   public void addCash(String accountNumber, float cash) throws  ESInvocationException;
   public float listBalance(String accountNumber) throws  ESInvocationException;
```

```
   public String getAccountName(String accountNumber) throws  ESInvocationException;
   public ESArray listShares(String accountNumber) throws  ESInvocationException;
   public ESArray listTransactions(String accountNumber) throws  ESInvocationException;
   public void closeTransaction(OrderDetailParam order, OrderDetailParam matchWith) throws
ESInvocationException;
   public String login(String accountNumber, String password) throws  ESInvocationException;
}
```

3. BankServiseStub.java

```
// Αρχείο παραγόμενο από την μεταγλώτισση του BankServiceIntf.esidl μέσω του IDL compiler
import net.espeak.jesi.ESConnection;
import net.espeak.infra.cci.messaging.MessageRegistry;
import net.espeak.infra.cci.messaging.MessageRegistry.SecondaryRegistry;
import net.espeak.infra.client.exception.UnexpectedExceptionException;
import net.espeak.jesi.ESServiceStub;
import net.espeak.infra.cci.messaging.MessageOutputStream;
import sharebroker.OrderDetailParam;
import sharebroker.Address;
import java.io.IOException;
import net.espeak.infra.cci.exception.ESInvocationException;
import net.espeak.infra.cci.exception.ESException;
import net.espeak.util.ESArray;
import sharebroker.ShareDetailParam;
import net.espeak.jesi.ESAccessor;
import net.espeak.infra.client.util.ParameterList;
import net.espeak.infra.cci.messaging.MessageInputStream;


public class BankServiceStub
extends ESServiceStub
implements BankServiceIntf{
   private static final short SECONDARY_ABI_VERSION = 10264;
   public BankServiceStub() {
      super();
      this.secondaryAbiVersion__ = SECONDARY_ABI_VERSION;
   }
   public BankServiceStub(ESConnection connection, ESAccessor accessor) {
      super(connection, accessor);
      this.secondaryAbiVersion__ = SECONDARY_ABI_VERSION;
   }
   public String openAccount(String arg0, Address arg1) throws  ESInvocationException {
      String result = null;
      ParameterList params = new ParameterList();
      params.addObject( arg0, "java.lang.String" );
      params.addObject( arg1, "shareBroker.Address" );
      try {
         Object retObj = this.invokeSynchronous("shareBroker.BankServiceIntf", "openAccount",
params);

            result = ( String ) retObj;

         }
      catch( ESInvocationException ex ) { throw ex; }
      catch( ESException ex) {
         throw new UnexpectedExceptionException(ex);

      }
      return result;
   }
```

```
  public boolean changePassword(String arg0, String arg1, String arg2) throws  ESInvocationException
{
    boolean result = true;
    ParameterList params = new ParameterList();
    params.addObject( arg0, "java.lang.String" );
    params.addObject( arg1, "java.lang.String" );
    params.addObject( arg2, "java.lang.String" );
    try {
      Object retObj = this.invokeSynchronous("shareBroker.BankServiceIntf", "changePassword",
params);
      result = ((java.lang.Boolean) retObj).booleanValue();
      }
    catch( ESInvocationException ex ) { throw ex; }
    catch( ESException ex) {
      throw new UnexpectedExceptionException(ex);
    }
    return result;
  }

  public void addShare(String arg0, ShareDetailParam arg1) throws  ESInvocationException {
    ParameterList params = new ParameterList();
    params.addObject( arg0, "java.lang.String" );
    params.addObject( arg1, " shareBroker.ShareDetailParam" );
    try {
      Object retObj = this.invokeSynchronous("shareBroker.BankServiceIntf", "addShare", params);

      }
    catch( ESInvocationException ex ) { throw ex; }
    catch( ESException ex) {
      throw new UnexpectedExceptionException(ex);

    }
  }

  public void addCash(String arg0, float arg1) throws  ESInvocationException {
    ParameterList params = new ParameterList();
    params.addObject( arg0, "java.lang.String" );
    params.addObject( new java.lang.Float(arg1), "float" );
    try {
      Object retObj = this.invokeSynchronous("shareBroker.BankServiceIntf", "addCash", params);

      }
    catch( ESInvocationException ex ) { throw ex; }
    catch( ESException ex) {
      throw new UnexpectedExceptionException(ex);

    }
  }

  public float listBalance(String arg0) throws  ESInvocationException {
    float result = 0.0f;
    ParameterList params = new ParameterList();
    params.addObject( arg0, "java.lang.String" );
    try {
      Object retObj = this.invokeSynchronous("shareBroker.BankServiceIntf", "listBalance", params);

        result = ((java.lang.Float) retObj).floatValue();
       }
    catch( ESInvocationException ex ) { throw ex; }
```

```
    catch( ESException ex) {
      throw new UnexpectedExceptionException(ex);
    }
    return result;
  }

  public String getAccountName(String arg0) throws  ESInvocationException {
    String result = null;
    ParameterList params = new ParameterList();
    params.addObject( arg0, "java.lang.String" );
    try {
      Object retObj = this.invokeSynchronous("shareBroker.BankServiceIntf", "getAccountName",
params);

        result = ( String ) retObj;

      }
    catch( ESInvocationException ex ) { throw ex; }
    catch( ESException ex) {
      throw new UnexpectedExceptionException(ex);

    }
    return result;
  }

  public ESArray listShares(String arg0) throws  ESInvocationException {
    ESArray result = null;
    ParameterList params = new ParameterList();
    params.addObject( arg0, "java.lang.String" );
    try {
      Object retObj = this.invokeSynchronous("shareBroker.BankServiceIntf", "listShares", params);

        result = ( ESArray ) retObj;

      }
    catch( ESInvocationException ex ) { throw ex; }
    catch( ESException ex) {
      throw new UnexpectedExceptionException(ex);

    }
    return result;
  }

  public ESArray listTransactions(String arg0) throws  ESInvocationException {
    ESArray result = null;
    ParameterList params = new ParameterList();
    params.addObject( arg0, "java.lang.String" );
    try {
      Object retObj = this.invokeSynchronous("shareBroker.BankServiceIntf", "listTransactions",
params);
        result = ( ESArray ) retObj;

      }
    catch( ESInvocationException ex ) { throw ex; }
    catch( ESException ex) {
      throw new UnexpectedExceptionException(ex);

    }
    return result;
  }
```

```
   public void closeTransaction(OrderDetailParam arg0, OrderDetailParam arg1) throws
ESInvocationException {
      ParameterList params = new ParameterList();
      params.addObject( arg0, " shareBroker.OrderDetailParam" );
      params.addObject( arg1, " shareBroker.OrderDetailParam" );
      try {
         Object retObj = this.invokeSynchronous("BankServiceIntf", "closeTransaction", params);

         }
      catch( ESInvocationException ex ) { throw ex; }
      catch( ESException ex) {
         throw new UnexpectedExceptionException(ex);

      }
   }

   public String login(String arg0, String arg1) throws  ESInvocationException {
      String result = null;
      ParameterList params = new ParameterList();
      params.addObject( arg0, "java.lang.String" );
      params.addObject( arg1, "java.lang.String" );
      try {
         Object retObj = this.invokeSynchronous("shareBroker.BankServiceIntf", "login", params);

            result = ( String ) retObj;

         }
      catch( ESInvocationException ex ) { throw ex; }
      catch( ESException ex) {
         throw new UnexpectedExceptionException(ex);

      }
      return result;
   }

   public static Object receiveObject(MessageInputStream in)
      throws IOException
   {
      ESAccessor x = ESAccessor.receiveESAccessor(in);
      BankServiceStub stub = new BankServiceStub(x.getConnection(), x);
      stub.secondaryAbiVersion__ = in.readShort();
      return stub;
   }

   static {
      (new BankServiceIntfMessageRegistry()).initialize();
   }
}
```

4. BankServiseStub.java

```
// Αρχείο παραγόμενο από την μεταγλώτισση του BankServiceIntf.esidl μέσω του IDL compiler
package sharebroker;
import net.espeak.jesi.ESMessageRegistryIntf;
import net.espeak.infra.cci.messaging.MessageInputStream;
import net.espeak.infra.cci.messaging.MessageRegistry;
import net.espeak.infra.cci.messaging.MessageRegistry.SecondaryRegistry;
import net.espeak.infra.cci.messaging.MessagableFactory;
import java.io.IOException;
```

```
public class BankServiceIntfMessageRegistry implements ESMessageRegistryIntf {
   private static boolean initialized = false;
   private static final short SECONDARY_ABI_VERSION = 10264;
   public void initialize() {
      synchronized(BankServiceIntfMessageRegistry.class) {
         if (initialized == false) {
            initialized = true;
         } else {
            return;
         }
      }
      MessageRegistry mr = MessageRegistry.getMessageRegistry();
      MessageRegistry.SecondaryRegistry r = mr.new
    SecondaryRegistry(SECONDARY_ABI_VERSION);
      r.register(new shareBroker.Address(),(short) 256,Address.class);
      r.register(new shareBroker.Company(),(short) 257,Company.class);
      r.register(new shareBroker.OrderDetailParam(),(short) 258, OrderDetailParam.class);
      r.register(new shareBroker.ShareDetailParam(),(short) 259,ShareDetailParam.class);
      r.register(new MessagableFactory() {
            public Object receiveObject(MessageInputStream in)throws IOException {
               return BankServiceStub.receiveObject(in);
            } }, (short) 260, BankServiceStub.class);
         }
}
```

5. BankService.java

```
package sharebroker;
import net.espeak.jesi.*;
import net.espeak.infra.cci.exception.*;


/**
* Αυτή η κλάση υλοποιεί την πλευρά του server της υπηρεσίας μιας συγκεκριμένης τράπεζας
* Αρχικά καταχωρεί το λεξιλόγιο της τράπεζας (Bank Vocabulary)στον πυρήνα. Στη συνέχεια
καταχωρεί *την ίδια την υπηρεσία στον πυρήνα και κατόπιν ξεκινάει την τραπεζική υπηρεσία
*
*/
public class BankService {
   private static String bankName = null;
```

```
/**
 *Κύρια Μέθοδος της κλάσης
 */
public static void main(String[] args) {
   if (args.length < 3) {
      System.out.println("Χρήση: java BankService <vocab xml file>"
         + " <contractName> <service xml File>");
      return;
   }
   try {

      // Σύνδεση στον πυρήνα του E-SPEAK
      ESConnection session = new ESConnection("bank.pr");
      BankVocabulary bv = new BankVocabulary(session, args[0], args[1]);
      BankService bs = new BankService(session, args[2], args[1]);
   } catch (Exception e) {
      e.printStackTrace();
   }
}


/**
 *Παράμετροι:
 * Η ESConnection χρησιμοποιείται για την επικοινωνία με τον πυρήνα
 *
 * Η svcXMLFile  χρησιμοποιείται για την περιγραφή του λεξιλογίου που περιγράφει την τράπεζα
 *
 * Η contractName αναφέρεται στο συμβόλαιο που χρησιμοποιείται για την καταχώρηση της
υπηρεσίας
 *
 */
public BankService(ESConnection session, String svcXMLFile,
   String contractName) {
   try {

      // Δημιουργία της υπηρεσίας
      String tempStr;
      ESServiceDescription essd =
         new ESServiceDescription(new ESXMLFile(svcXMLFile), session);

      essd.addAttribute(ESConstants.SERVICE_NAME, contractName);
      ESServiceElement servElem = new ESServiceElement(session, essd);

      servElem.setImplementation(new BankServiceImpl());
      ESAccessor accessor = servElem.register();

      //Εναρξη της Υπηρεσίας
      servElem.start();
      tempStr = (String)accessor.getAttribute("BankName");
      bankName = new String(tempStr);
      MessageClient.out(bankName + " : Έναρξη Υπηρεσίας");
      servElem.advertise();
   } catch (ESServiceException e) {
      MessageClient.out(bankName+ " : Η Υπηρεσία διαφήμισης δεν λειτουργεί ");
   } catch (Exception e) {
      e.printStackTrace();
   }
}
```

```
  /**
   * Επιστρέφει το όνομα της τράπεζας
   */
  public static String getBankName() {
    return bankName;
  }
}
```

6. BankServiceImpl.java

```
package sharebroker;
import net.espeak.util.*;
import net.espeak.infra.cci.exception.*;
import net.espeak.jesi.*;
import java.util.*;
import java.lang.*;
import java.io.*;
import net.espeak.infra.client.impl.*;
import net.espeak.infra.client.util.*;


/**
 * Η κλάση αυτή υλοποιεί στην πράξη την τραπεζική υπηρεσία
 *Περιέχει όλες τις υλοποιήσεις των μεθόδων  στις οποιές αναφέρονται όλες οι κλάσεις της εφαρμογής
 * μέσω της διεπαφής BankServiceIntf.
*/
public class BankServiceImpl implements BankServiceIntf {
  String bankName;
  private Hashtable accountHash;
  private BankService service;
  private int count;


  /**
   * Δημιουργός
   */
  BankServiceImpl() {
    count = 0;
    File f = new File("bankdata.per");

    if (f.exists()) {
      try {
        FileInputStream fis = new FileInputStream("bankdata.per");
        ObjectInputStream ois = new ObjectInputStream(fis);

        accountHash = (Hashtable)(ois.readObject());
        ois.close();
        fis.close();
      } catch (Exception e) {
        accountHash = new Hashtable();
      }
    } else {
      accountHash = new Hashtable();
    }
    PersistThread p = new PersistThread(3, accountHash);
    Thread t = new Thread(p);

    t.start();
```

```
}


/**
 *
 *Η μέθοδος αυτή δημιουργεί ένα νέο λογαρισμό βάσει των δεδομένων που λαμβάνει από την
 * OpenAccountGUI. Επιστρέφει ένα νέο λογαριασμό τράπέζης στον χρήστη
 *
 *Παράμετροι:
 *Η  userName αναφέρεται στο όνομα του χρήστη
 *Η  address αποτελεί κλάση η οποία περιέχει τις λεπτομέρειες του κάθε χρήστη
 *
 */
public String openAccount(String userName, Address address) {
   String tempString = BankService.getBankName();
   UserAccount userAccount = new UserAccount();
   AccountNumberGenerator accGenerator =
      new AccountNumberGenerator("account.dat");
   try {
      userAccount.accountNumber = accGenerator.nextNumber();
      userAccount.userName = userName;
      userAccount.passWord = userAccount.userName;
      userAccount.address = address;
      userAccount.balance = 0.0f;
      userAccount.sharesFolder = new ESArray();
      userAccount.transactionFolder = new ESArray();
       accountHash.put(userAccount.accountNumber, userAccount);
   } catch (Exception e) {
      e.printStackTrace();
   }
   return userAccount.accountNumber;
}


/**
 *
 *Η μέθοδος αυτή αλλάζει τον κωδικό εισόδου ενός δοθέντος λογαριασμού.
 *Παράμετροι:
 *  accountNumber  Ο αριθμός του λογαριασμού που θα αλλάξει κωδικό
 *oldPassword Ο παλαιός κωδικός εισόδου
 *newPassword Ο νέος κωδικός εισόδου
 *
 */
public boolean changePassword(String accountNumber, String oldPassword,
   String newPassword) {
   String passwd;
   UserAccount userAccount;

   try {
      String tempString = BankService.getBankName();

      userAccount = (UserAccount)accountHash.get(accountNumber);
          if (userAccount.passWord.equals(oldPassword)) {
         userAccount.passWord = newPassword;
         accountHash.put(accountNumber, userAccount);
         return true;
      } else {
         return false;
      }
   } catch (Exception e) {
```

```
            e.printStackTrace();
            return false;
        }
    }


    /**
     *Η μέθοδος αυξάνει τις ήδη υπάρχουσες  μετοχές που βρίσκονται στο χαρτοφυλάκιο ενός πελάτη
     *Παράμετροι
     *  userAccount: Λογαριασμός του χρήστη
     *  share: Μετοχή που προστίθεται
     *
     */
    public boolean addShareToFolder(UserAccount userAccount,
        ShareDetailParam share) {
        ShareDetailParam matchWith;
        int size = userAccount.sharesFolder.size();
    for (count = 0; count < size; count++) {
            matchWith =
                (ShareDetailParam)userAccount.sharesFolder.elementAt(count);
            if (matchWith.company.companyName.equals(share.company.companyName)) {
                if (matchWith.companyReference.equals(share.companyReference)) {
                    if (matchWith.categoryName.equals(share.categoryName)) {
                        matchWith.numShares += share.numShares;
                        matchWith.pricePerShare = share.pricePerShare;
                        userAccount.sharesFolder.setElementAt((ShareDetailParam)matchWith,
                            count);
                        return true;
                    }
                }
            }
        }
        return false;
    }


    /**
    *Η μέθοδος προσθέτει μια νέα μετοχή (τίτλο) στο χαρτοφυλάκιο του πελάτη. Σε αντίθεση με την
    * προηγούμενη μέθοδο  η οποία αυξάνει μια ήδη υπάρχουσα.
    *Παράμετροι:
    *accountNumber :Ο αριθμός λογαριασμού
    *
    *  share :Η μετοχή που θα προστεθεί
    *
    */
    public void addShare(String accountNumber, ShareDetailParam share) {
        try {
            UserAccount userAccount =
                (UserAccount)accountHash.get(accountNumber);

            if (!addShareToFolder(userAccount, share)) {
                userAccount.sharesFolder.addElement((ShareDetailParam)share);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }


    /**
```

```
 * Η μέθοδος επιστρέφει το υπόλοιπο ενός συγκεκριμένου λογαριασμού
 * Παράμετροι
 * accountNumber : Ο αριθμός λογαριασμού
 *
 */
public float listBalance(String accountNumber) {
   try {
      UserAccount userAccount =
         (UserAccount)accountHash.get(accountNumber);
       return (userAccount.balance);
   } catch (Exception e) {
      e.printStackTrace();
      return ((float)0.0);
   }
}


/**
 *Η μέθοδος επιστρέφει το όνομα του κατόχου του λογαριασμού
 *Παράμετροι:
* accountNumber : Ο αριθμός λογαριασμού
*/
public String getAccountName(String accountNumber) {
   try {
         UserAccount userAccount =
        (UserAccount)accountHash.get(accountNumber);
    return (userAccount.userName);
   } catch (Exception e) {
      e.printStackTrace();
      return null;
   }
}


/**
 * Η μέθοδος επιστρέφει την λίστα των μετοχών που σχετίζονται με ένα συγκεκριμένο λογαριασμό.
 *Παράμετροι:
 * accountNumber :Ο αριθμός λογαριασμού
 *
 */
public ESArray listShares(String accountNumber) {
   try {
      UserAccount userAccount =
         (UserAccount)accountHash.get(accountNumber);
      return (userAccount.sharesFolder);
   } catch (Exception e) {
      e.printStackTrace();
      return ((ESArray)null);
   }
}


 /**
  * Η μέθοδος επιστρέφει την λίστα των συναλλαγών που σχετίζονται με ένα συγκεκριμένο
λογαριασμό.
  *Παράμετροι:
  * accountNumber :Ο αριθμός λογαριασμού
  *
  */
  public ESArray listTransactions(String accountNumber) {
```

```
        try {
            UserAccount userAccount =
                (UserAccount)accountHash.get(accountNumber);
                return (userAccount.transactionFolder);
        } catch (Exception e) {
            e.printStackTrace();
            return ((ESArray)null);
        }
    }

    /**
     *Η μέθοδος αυτή χρησιμοποιείται για την μεταφορά χρημάτων από τον λογαριασμό του αγοραστή σε
     *αυτόν του πωλητή, καθώς και για την μεταφορά τον μετοχών από το χαρτοφυλάκιο του πωλητή σε
     *αυτό του αγοραστή
     * Παράμετροι:
     * order1: Η πρώτη εντολή (μπορεί να είναι πωλητή ή αγοραστή)
     *
      * order2: Η δεύτερη εντολή (μπορεί να είναι πωλητή ή αγοραστή)
*/
    public void closeTransaction(OrderDetailParam order1,
        OrderDetailParam order2) {

        OrderDetailParam sellOrder, buyOrder;
        UserAccount sellersAccount, buyersAccount;

        if (order1.option == 1) {
            buyOrder = (OrderDetailParam)order1;
            sellOrder = (OrderDetailParam)order2;
        } else {
            buyOrder = (OrderDetailParam)order2;
            sellOrder = (OrderDetailParam)order1;
        }
        sellersAccount =
            (UserAccount)accountHash.get(sellOrder.accountNumber);
        buyersAccount = (UserAccount)accountHash.get(buyOrder.accountNumber);
       ESArray allShares = sellersAccount.sharesFolder;
        int size = allShares.size();
        int count;
        ShareDetail share;

        for (count = 0; count < size; count++) {
            share = (ShareDetail)(allShares.elementAt(count));
            if (share.companyReference.equals(sellOrder.companyReference)) {
                if (share.numShares == sellOrder.no) {
                    allShares.removeElementAt(count);
                }
                ShareDetail newShare = splitShare(share, sellOrder.no);

                newShare.pricePerShare = buyOrder.price;
                newShare.companyReference =
                    new String(sellOrder.companyReference);
                if (!addShareToFolder(buyersAccount, newShare)) {
                    buyersAccount.sharesFolder.addElement(newShare);
                }
                buyersAccount.balance = buyersAccount.balance
                    - buyOrder.no * buyOrder.price;
                sellersAccount.balance = sellersAccount.balance
                    + buyOrder.no * buyOrder.price;
                break;
            }
```

```
    }
  }


  /**
   *Η μέθοδος προσθέτει χρήματα στο λογαριασμό του πελ
   * Παράμετροι:
   *accountNumber : Ο αριθμός λογαριασμού
   *
   * cash: Το ποσό που πρέπει να προστεθεί
   *
   */
  public void addCash(String accountNumber, float cash) {
    UserAccount userAccount;
    float balance;

    try {
      userAccount = (UserAccount)accountHash.get(accountNumber);
      userAccount.balance += cash;
      return;
    } catch (Exception e) {
      e.printStackTrace();
      return;
    }
  }


  /**
   * Η μέθοδος επιτρέπει την πρόσβαση του χρήστη σε ένα συγκεκριμένο λογαρισμό
   * Παράμετροι:
   * accountNumber :Ο αριθμός λογαριασμού
   *  password: Ο κωδικός πρόσβασης του λογαριασμού
   *
   */
  public String login(String accountNumber, String password) {
    String userName = null;
    String passwd;
    UserAccount userAccount;

    try {
      userAccount = (UserAccount)accountHash.get(accountNumber);
      if (userAccount.passWord.equals(password)) {
        return (new String(userAccount.userName));
      } else {
        return null;
      }
    } catch (Exception e) {
      return null;
    }
  }
}


/**
 * Η ακόλουθη κλάση χρησιμοποιείται για την διατήρηση της λειτουργίας της τραπεζικής υπηρεσίας
 *  καθόλη την διάρκεια της χρήσης. Μέσω της μεθόδου persistify() τα δεδομένα των πελατών
*αποθηκεύονται σε ένα αρχείο και για μελλοντική χρήση
 *
 */
class PersistThread implements Runnable {
```

```
   private int sleepInterval;
   private Hashtable accountHash;

public PersistThread(int i, Hashtable acc) {
      this.sleepInterval = i * 60 * 1000;
      this.accountHash = acc;
   }


public void run() {
      while (true) {
         persistify();
         try {
            Thread.sleep(sleepInterval);
         } catch (InterruptedException e) {}
      }
   }


public synchronized void persistify() {
      try {
         FileOutputStream fos = new FileOutputStream("bankdata.per");
         ObjectOutputStream oos = new ObjectOutputStream(fos);

         oos.writeObject(accountHash);
         oos.flush();
         oos.close();
         fos.close();
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

7. BankServiceIntfFinder.java

```
package sharebroker;
import java.util.Hashtable;
import java.util.Enumeration;
import net.espeak.jesi.*;
import net.espeak.infra.cci.exception.*;


/**
 * Σκοπός της κλάσης αυτής είναι η εύρεση των διεπαφών των τραπεζικών υπηρεσιών
*/
public class BankServiceIntfFinder {
   static ESVocabulary vocab = null;
   static BankServiceIntf bsi = null;

  /**
    * Η μέθοδος βρίσκει την διεπαφή μιας συγκεκριμένης υπηρεσίας
  */
   public static BankServiceIntf find(ESConnection connection) {
      bsi = null;
      try {
         if (connection == null) {
            connection = new ESConnection("bank.pr");
         }
         ESVocabularyFinder vf = new ESVocabularyFinder(connection);
```

```
        vocab = vf.find(new ESQuery("Name == 'BankVocabulary'"));
        String queryString = "(BankName == '" + "Ethniki Bank" + "' and "
          + "City == '" + "Athens" + "')";
        String interfaceName ="shareBroker.BankServiceIntf";
        ESQuery query = new ESQuery(vocab, queryString);
        ESServiceFinder sf = new ESServiceFinder(connection,
          interfaceName);
        bsi = (BankServiceIntf)sf.find(query);
      } catch (Exception e) {
        e.printStackTrace();
      }
      return bsi;
  }


  /**
   * Η μέθοδος βρίσκει τις διεπαφές  όλων των διαφορετικών τραπεζικών υπηρεσιών που είναι
   * συνδεδεμένες στον πυρήνα
  */
  public static Object[][] findBankProperties(ESConnection connection) {
      try {
        ESVocabularyFinder vf = new ESVocabularyFinder(connection);
        vocab = vf.find(new ESQuery("Name == 'BankVocabulary'"));
      } catch (Exception e) {}
      String queryString = "(Identifier == '" + "bank" + "')";
      ESQuery query = new ESQuery(vocab, queryString);
      String interfaceName = "shareBroker.BankServiceIntf";
      ESServiceFinder sf = new ESServiceFinder(connection, interfaceName);
      ESService[] services = null;
      Object[][] result = null;
      int j;

      try {
        services = sf.findAll(query);
        if (services != null && services.length != 0) {
          result = new Object[services.length][4];
          String tempStr;
          for (j = 0; j < services.length; j++) {
            ESServiceStub stub = (ESServiceStub)(services[j]);
            tempStr = (String)stub.getAccessor().getAttribute("BankName");
            if (tempStr != null) {
              result[j][0] = new String(tempStr);
            } else {
              result[j][0] = new String(" ");
            }
            tempStr =
              (String)stub.getAccessor().getAttribute("Street");
            if (tempStr != null) {
              result[j][1] = new String(tempStr);
            } else {
              result[j][1] = new String(" ");
            }
            tempStr = (String)stub.getAccessor().getAttribute("City");
            if (tempStr != null) {
              result[j][2] = new String(tempStr);
            } else {
              result[j][2] = new String(" ");
            }
            tempStr =
              (String)stub.getAccessor().getAttribute("Country");
```

```
                if (tempStr != null) {
                    result[j][3] = new String(tempStr);
                } else {
                    result[j][3] = new String(" ");
                }
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return result;
  }
}
```

8. AddFundsGui.java

```
package sharebroker;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;


/**
* Η AddFundsGUI δημιουργεί την οθόνη προσθήκης κεφαλαίου στον λογαριασμό του πελάτη *
*/
public class AddFundsGUI extends JPanel {
  JLabel titleLabel, amountLabel;
  JTextField amountTextField;
  JButton okayButton, cancelButton;
  AddListener addListener = new AddListener();
  StockTrade stockTrade;


  /**
   * Ο δημιουργός  παίρνει σαν παράμετρο το αντικείμενο StockTrade object. Στη συνέχεια δημιουργεί
    *την οθόνη και ενεργοποιεί τα χειριστήρια της μέσω του event listener
   */
  public AddFundsGUI(StockTrade stockTrade) {
    this.stockTrade = stockTrade;
    setLayout(null);
    setVisible(false);
    titleLabel = new JLabel("Προσθήκη Κεφαλαίων");
    titleLabel.setFont(new Font("Times New Roman", Font.BOLD, 20));
    titleLabel.setBounds(10, 20, 250, 30);
    add(titleLabel);
    amountLabel = new JLabel("Ύψος Κεφαλαίου");
    amountLabel.setBounds(10, 60, 150, 20);
    add(amountLabel);
    amountTextField = new JTextField();
    amountTextField.setBounds(140, 60, 150, 20);
    add(amountTextField);
    okayButton = new JButton("Επιβεβαίωση");
    okayButton.setBounds(45, 110, 110, 20);
    add(okayButton);
    okayButton.addActionListener(addListener);
    cancelButton = new JButton("Ακύρωση");
    cancelButton.setBounds(155, 110, 110, 20);
    add(cancelButton);
```

```
    cancelButton.addActionListener(addListener);
}


/**
 * Η AddListener είναι κλάση , η οποία με τις μεθόδους της χειρίζεται τα 'γεγονότα' που προκύπτουν
 * από τα χειριστήρια
 * Έαν το γεγονός είναι η επιβεβαίωση τότε το κεφάλαιο προστίθενται στο λογαριασμό
  */
class AddListener implements ActionListener {
    public void actionPerformed(ActionEvent event) {
        Object object = event.getSource();

        if (object == cancelButton) {
            hideAddFundsPanel();
            stockTrade.repaint();
        } else if (object == okayButton) {

            // Κλήση της μεθόδου που κάνει την προσθήκη
            try {
                BankServiceIntfFinder.find(stockTrade.connection).addCash(stockTrade.accountNumber,
                    (new Float(amountTextField.getText().trim()).floatValue()));
            } catch (NumberFormatException e) {
                JOptionPane.showMessageDialog(null,"Αναμένονται αριθμητικά δεδομένα"
                    + " για το πεδίο ύψος κεφαλαίου");
                showAddFundsPanel();
                stockTrade.repaint();
                return;
            } catch (Exception e) {
                e.printStackTrace();
            }
            ShowMessageDialog("Το κεφάλαιο προστέθηκε στο λογαριασμό");
             hideAddFundsPanel();
            stockTrade.repaint();
        }
    }
}


/**
 * Αρχικοποιεί το σχετικό πεδίο
 */
public void showAddFundsPanel() {
    amountTextField.setText("");
    setVisible(true);
}


/**
 * Εξαφανίζει από το προσκήνιο την οθόνη
 */
public void hideAddFundsPanel() {
    setVisible(false);
}


/**
 * Απεικονίζει το μύνημα με το οποίο αρχικοποιήθηκε*
 */
void ShowMessageDialog(String message) {
```

```
        (new MessageDialog(stockTrade, true, message)).show();
    }
}
```

9. AddSharesGUI.java

```
package sharebroker;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;


/**
 * Η AddSharesGUI δημιοργεί την οθόνη προσθήκης μετοχών στο χαρτοφλάκιο του συγκεκριμένου
 *  πελάτη .
*/
public class AddSharesGUI extends JPanel {

JLabel titleLabel, categoryLabel, companyLabel, companyReferenceLabel,
        numSharesLabel, priceLabel;
   JTextField categoryTextField, companyTextField,
      companyReferenceTextField, numSharesTextField, priceTextField;
   JComboBox CategoryCombo;
   JButton okayButton, cancelButton;
   AddListener addListener = new AddListener();
   StockTrade stockTrade;

 /**
    * Ο δημιουργός  παίρνει σαν παράμετρο το αντικείμενο StockTrade object. Στη συνέχεια δημιουργεί
    *την οθόνη και ενεργοποιεί τα χειριστήρια της μέσω του event listener
 */
public AddSharesGUI(StockTrade stockTrade) {
      this.stockTrade = stockTrade;
      setLayout(null);
      setVisible(false);
      titleLabel = new JLabel("Προσθήκη Μετοχών");
      titleLabel.setFont(new Font("Times New Roman", Font.BOLD, 20));
      titleLabel.setBounds(10, 20; 250, 30);
      add(titleLabel);
      categoryLabel = new JLabel("Κατηγορία");
      categoryLabel.setBounds(10, 60, 120, 20);
      add(categoryLabel);

      String[] Categories = {"Τράπεζες", "Ασφάλειες", "Επενδύσεων", "Συμμετοχών",
"Τηλεπικοινωνίες","Ενέργειας"};
      CategoryCombo = new JComboBox(Categories);
      CategoryCombo.setBounds(140,60,150,20);
      add(CategoryCombo);
      companyLabel = new JLabel("Εταιρεία");
      companyLabel.setBounds(10, 90, 150, 20);
      add(companyLabel);
      companyTextField = new JTextField();
      companyTextField.setBounds(140, 90, 150, 20);
      add(companyTextField);
      companyReferenceLabel = new JLabel("Σύντμηση");
      companyReferenceLabel.setBounds(10, 120, 150, 20);
      add(companyReferenceLabel);
      companyReferenceTextField = new JTextField();
      companyReferenceTextField.setBounds(140, 120, 150, 20);
```

```
        add(companyReferenceTextField);
        numSharesLabel = new JLabel("Μερίδια");
        numSharesLabel.setBounds(10, 150, 150, 20);
        add(numSharesLabel);
        numSharesTextField = new JTextField();
        numSharesTextField.setBounds(140, 150, 150, 20);
        add(numSharesTextField);
        priceLabel = new JLabel("Τιμή Μεριδίου");
        priceLabel.setBounds(10, 180, 150, 20);
        add(priceLabel);
        priceTextField = new JTextField();
        priceTextField.setBounds(140, 180, 150, 20);
        add(priceTextField);
        okayButton = new JButton("Επιβεβαίωση");
        okayButton.setBounds(45, 230, 110, 20);
        add(okayButton);
        okayButton.addActionListener(addListener);
        cancelButton = new JButton("Ακύρωση");
        cancelButton.setBounds(155, 230, 110, 20);
        add(cancelButton);
        cancelButton.addActionListener(addListener);
    }

/**
     * Η AddListener είναι κλάση , η οποία με τις μεθόδους της χειρίζεται τα 'γεγονότα' που προκύπτουν
     * από τα χειριστήρια
     * Έαν το γεγονός είναι η επιβεβαίωση τότε οι μετοχές προστίθενται στο λογαριασμό
*/
class AddListener implements ActionListener {
public void actionPerformed(ActionEvent event) {
        Object object = event.getSource();
        if (object == cancelButton) {
            hideAddSharesPanel();
            stockTrade.repaint();
        } else if (object == okayButton) {

            // Καλεί την μέθοδο που προσθέτει τις μετοχές στο λογαριασμό
            ShareDetailParam share = new ShareDetail();
            share.categoryName = new String(CategoryCombo.getSelectedItem().toString());
            share.company = new Company();
            share.company.companyName =
                new String(companyTextField.getText());
            share.companyReference =
                new String(companyReferenceTextField.getText());
            try {
                share.numShares =Integer.parseInt(numSharesTextField.getText().trim());
                share.pricePerShare = new Float(priceTextField.getText().trim()).floatValue();
            } catch (NumberFormatException e) {
                JOptionPane.showMessageDialog(null, "Αναμένονται αριθμητικά δεδομένα "
                    + "για τα ακόλουθα πεδία\n Μερίδια \n" + "Τιμή Μεριδίου");
                showAddSharesPanel();
                stockTrade.repaint();
                return;
            }
            try {
                BankServiceIntfFinder.find(stockTrade.connection).addShare(stockTrade.accountNumber,
                    share);
            } catch (Exception e) {
                e.printStackTrace();
            }
```

```
            ShowMessageDialog("Οι μετοχές προστέθηκαν στο λογαριασμό");
            hideAddSharesPanel();
            stockTrade.repaint();
          }
        }
    }


    /**
     *Αρχικοποιεί τα πεδία της οθόνης
     */
    public void showAddSharesPanel() {
      //categoryTextField.setText("");
        companyTextField.setText("");
        companyReferenceTextField.setText("");
        numSharesTextField.setText("");
        priceTextField.setText("");
        setVisible(true);
    }


    /**
     * Εξαφανίζει από το προσήνιο την οθόνη
     */
    public void hideAddSharesPanel() {
        setVisible(false);
    }


    /**
     * Εμφανίζει το μύνημα με το οποίο αρχικοποιήθηκε
     */
    void ShowMessageDialog(String message) {
        (new MessageDialog(stockTrade, true, message)).show();
    }
}
```

10. BuySharesGUI.java

```
package sharebroker;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;


/**
* Η BuySharesGUI δημιουργεί την βασική οθόνη αγοράς μετοχών για έναν πελάτη
*/
public class BuySharesGUI extends JPanel {
JLabel titleLabel, buyCategoryLabel, buyCompanyLabel,
        buyCompanyReferenceLabel, buyNumSharesLabel, buyMaxValueLabel,
        buyValidTillLabel, buyLotSizeLabel;
  JTextField buyCategoryTextField, buyCompanyTextField,
      buyCompanyReferenceTextField, buyNumSharesTextField,
      buyMaxValueTextField, buyValidTillTextField;
    JComboBox buyLotSize;
    JButton buyOkayButton, buyCancelButton;
    BuyListener buyListener = new BuyListener();
```

```
  StockTrade stockTrade;
  TraderQuerySpec trader;


/**
   * Ο δημιουργός  παίρνει σαν παράμετρο το αντικείμενο StockTrade object. Στη συνέχεια δημιουργεί
   *την οθόνη και ενεργοποιεί τα χειριστήρια της μέσω του event listener
 */
  public BuySharesGUI(StockTrade stockTrade) {
     this.stockTrade = stockTrade;
     trader = new TraderQuerySpec();
     trader.traderName = new String(stockTrade.userName);
     setLayout(null);
     setVisible(false);
     titleLabel = new JLabel("Αγορά Μετοχών");
     titleLabel.setFont(new Font("Times New Roman", Font.BOLD, 20));
     titleLabel.setBounds(10, 20, 250, 30);
     add(titleLabel);
     buyCategoryLabel = new JLabel("Κατηγορία");
     buyCategoryLabel.setBounds(10, 60, 120, 20);
     add(buyCategoryLabel);
     buyCategoryTextField = new JTextField();
     buyCategoryTextField.setBounds(140, 60, 150, 20);
     add(buyCategoryTextField);
     buyCompanyLabel = new JLabel("Εταιρεία");
     buyCompanyLabel.setBounds(10, 90, 120, 20);
     add(buyCompanyLabel);
     buyCompanyTextField = new JTextField();
     buyCompanyTextField.setBounds(140, 90, 150, 20);
     add(buyCompanyTextField);
     buyLotSizeLabel = new JLabel("Πακέτο των");
     buyLotSizeLabel.setBounds(10, 120, 120, 20);
     add(buyLotSizeLabel);
     String[] lotItems = {
        "10", "50", "100", "500", "1000"
     };

     buyLotSize = new JComboBox(lotItems);
     buyLotSize.setBounds(140, 120, 150, 20);
     add(buyLotSize);
     buyNumSharesLabel = new JLabel("Μερίδια");
     buyNumSharesLabel.setBounds(10, 150, 120, 20);
     add(buyNumSharesLabel);
     buyNumSharesTextField = new JTextField();
     buyNumSharesTextField.setBounds(140, 150, 150, 20);
     add(buyNumSharesTextField);
     buyMaxValueLabel = new JLabel("Μέγιστη τιμή");
     buyMaxValueLabel.setBounds(10, 180, 120, 20);
     add(buyMaxValueLabel);
     buyMaxValueTextField = new JTextField();
     buyMaxValueTextField.setBounds(140, 180, 150, 20);
     add(buyMaxValueTextField);
     buyValidTillLabel = new JLabel("Έγκυρη έως");
     buyValidTillLabel.setBounds(10, 210, 120, 20);
     add(buyValidTillLabel);
     buyValidTillTextField = new JTextField();
     buyValidTillTextField.setBounds(140, 210, 150, 20);
     add(buyValidTillTextField);
     buyOkayButton = new JButton("Επιβεβαίωση");
     buyOkayButton.setBounds(45, 290, 110, 20);
```

```
        add(buyOkayButton);
        buyOkayButton.addActionListener(buyListener);
        buyCancelButton = new JButton("Ακύρωση");
        buyCancelButton.setBounds(155, 290, 110, 20);
        add(buyCancelButton);
        buyCancelButton.addActionListener(buyListener);
    }

/**
    * Η BuyListener είναι κλάση , η οποία με τις μεθόδους της χειρίζεται τα 'γεγονότα' που προκύπτουν
    * από τα χειριστήρια
    * Έαν το γεγονός είναι η επιβεβαίωση τότε ενεργοποιείται η διαδικασία πώλησης μετοχών
*/

class BuyListener implements ActionListener {
public void actionPerformed(ActionEvent event) {
        Object object = event.getSource();

        if (object == buyCancelButton) {
            hideBuyPanel();
            stockTrade.repaint();
        } else if (object == buyOkayButton) {

            // Συλλογή των στοιχείων τησ πώλησης από τα αντίστοιχα πεδία
            String companyName = buyCompanyTextField.getText();
            String categoryName = buyCategoryTextField.getText();
            OrderDetailParam order = new OrderDetailParam();
            order.company = new String(companyName);
            order.categoryName = new String(categoryName);
            order.companyReference = new String("");
            if (order.company.equals("") || order.categoryName.equals("")) {
                JOptionPane.showMessageDialog(null,"Αναμένονται έγκυρα δεδομένα για τα πεδία"
                    + "\n Κατηγορία \n Εταιρεία  \n ");
                showBuyPanel();
                stockTrade.repaint();
                return;
            }
            try {
                int numShares = Integer.parseInt(buyNumSharesTextField.getText());
                order.no = numShares;
                int lotSize = Integer.parseInt(buyLotSize.getSelectedItem().toString());
                order.lotSize = lotSize;
                order.price =Float.valueOf(buyMaxValueTextField.getText()).floatValue();
                order.accountNumber =
                    new String(stockTrade.accountNumber);
               String temp = ShareBrokerIntfFinder.find(stockTrade.brokerName
stockTrade.connection).buy(trader, order);
                ShowMessageDialog("Η εντολή σας είναι  " + temp);
            } catch (NumberFormatException e) {
                JOptionPane.showMessageDialog(null,
                    "Αναμένονται αριθμητικά δεδομένα για τα πεδία"+ " \n Μερίδια \n Μέγιστη τιμή  \n
Έγκυρη έως");
                showBuyPanel();
                stockTrade.repaint();
                return;
            } catch (Exception e) {
                e.printStackTrace();
            }
            hideBuyPanel();
            stockTrade.repaint();
```

```
        }
      }
    }

    /**
     * Αρχικοποιεί τα πεδία της οθόνης
     */
    public void showBuyPanel() {
      buyCategoryTextField.setText("");
      buyCompanyTextField.setText("");
      buyNumSharesTextField.setText("");
      buyMaxValueTextField.setText("");
      buyValidTillTextField.setText("");
      setVisible(true);
    }

    /**
     * Εξαφανίζει την οθόνη από το προσκήνιο
     */
    public void hideBuyPanel() {
      setVisible(false);
    }

    /**
     * Γεμίζει τα πεδία της οθόνης όταν έχει επιλεγεί "Αγορά" από την οθόνη ListSharesGUI
     */
    public void fillData(String categoryName, String companyName,
      String lotSize, String numShares, String price) {
      buyCategoryTextField.setText(categoryName);
      buyCompanyTextField.setText(companyName);
      buyNumSharesTextField.setText(numShares);
      buyMaxValueTextField.setText(price);
      buyLotSize.setSelectedItem(lotSize);
      setVisible(true);
    }

    /**
     *Εμφανίζει το μύνημα με το οποίο αρχικοποιήθηκε
     */
    void ShowMessageDialog(String message) {
      (new MessageDialog(stockTrade, true, message)).show();
    }
}
```

11. SellSharesGUI.java

```
package sharebroker;
import net.espeak.util.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
```

```
/**
 * Η SellSharesGUI δημιουργεί την οθόνη που φροντίζει για την πώληση των μετοχών
 *του πελάτη
 */
public class SellSharesGUI extends JPanel {
   TraderQuerySpec trader;
JLabel titleLabel, sellCategoryLabel, sellCompanyLabel,
        sellCompanyReferenceLabel, sellNumSharesLabel, sellMinValueLabel,
        sellValidTillLabel, sellLotSizeLabel;
   JTextField
      sellCompanyReferenceTextField, sellNumSharesTextField,
      sellMinValueTextField, sellValidTillTextField;
   JComboBox sellLotSize,sellCategoryCombo,sellCompanyCombo;
   JButton sellOkayButton, sellCancelButton;
   SellListener sellListener = new SellListener();
   ESArray stocks = null;
   ShareDetailParam share;
   String Name;
   StockTrade stockTrade;

  /**
    * Ο δημιουργός  παίρνει σαν παράμετρο το αντικείμενο StockTrade object. Στη συνέχεια δημιουργεί
    *την οθόνη και ενεργοποιεί τα χειριστήρια της μέσω του event listener
  */
public SellSharesGUI(StockTrade stockTrade) {
       this.stockTrade = stockTrade;
       trader = new TraderQuerySpec();
       trader.traderName = new String(stockTrade.userName);
       setLayout(null);
       setVisible(false);
       titleLabel = new JLabel("Πώληση Μετοχών");
       titleLabel.setFont(new Font("Times New Roman", Font.BOLD, 20));
       titleLabel.setBounds(10, 20, 250, 30);
       add(titleLabel);
       sellCategoryLabel = new JLabel("Κατηγορία");
       sellCategoryLabel.setBounds(10, 60, 120, 20);
       add(sellCategoryLabel);
       String[] Categories = {
          "Τράπεζες", "Ασφάλειες", "Επενδύσεων", "Συμμετοχών", "Τηλεπικοινωνίες","Ενέργειας"};
       sellCategoryCombo = new JComboBox(Categories);
       sellCategoryCombo.setBounds(140,60,150,20);
       add(sellCategoryCombo);
       sellCompanyLabel = new JLabel("Εταιρεία");
       sellCompanyLabel.setBounds(10, 90, 120, 20);
       add(sellCompanyLabel);
       sellCompanyCombo = new JComboBox();
        try{
       stocks =
(BankServiceIntfFinder.find(stockTrade.connection)).listShares(stockTrade.accountNumber);
       } catch (Exception e){
          e.printStackTrace();
         }

      for (int i=0; i<stocks.size(); i++){
         share = (ShareDetailParam)(stocks.elementAt(i));
         Name = new String(share.company.companyName);
        sellCompanyCombo.addItem(Name);
      }
       stockTrade.repaint();
       sellCompanyCombo.setBounds(140,90,150,20);
```

```
        add(sellCompanyCombo);
        sellCompanyReferenceLabel = new JLabel("Σύντμηση");
        sellCompanyReferenceLabel.setBounds(10, 120, 120, 20);
        add(sellCompanyReferenceLabel);
        sellCompanyReferenceTextField = new JTextField();
        sellCompanyReferenceTextField.setBounds(140, 120, 150, 20);
        add(sellCompanyReferenceTextField);
        sellLotSizeLabel = new JLabel("Πακέτο των");
        sellLotSizeLabel.setBounds(10, 150, 120, 20);
        add(sellLotSizeLabel);
        String[] lotItems = {
           "10", "50", "100", "500", "1000"
        };
        sellLotSize = new JComboBox(lotItems);
        sellLotSize.setBounds(140, 150, 150, 20);
        add(sellLotSize);
        sellNumSharesLabel = new JLabel("Μερίδια");
        sellNumSharesLabel.setBounds(10, 180, 120, 20);
        add(sellNumSharesLabel);
        sellNumSharesTextField = new JTextField();
        sellNumSharesTextField.setBounds(140, 180, 150, 20);
        add(sellNumSharesTextField);
        sellMinValueLabel = new JLabel("Ελάχιστη Τιμή");
        sellMinValueLabel.setBounds(10, 210, 120, 20);
        add(sellMinValueLabel);
        sellMinValueTextField = new JTextField();
        sellMinValueTextField.setBounds(140, 210, 150, 20);
        add(sellMinValueTextField);
        sellValidTillLabel = new JLabel("Εγκυρη έως");
        sellValidTillLabel.setBounds(10, 240, 120, 20);
        add(sellValidTillLabel);
        sellValidTillTextField = new JTextField();
        sellValidTillTextField.setBounds(140, 240, 150, 20);
        add(sellValidTillTextField);
        sellOkayButton = new JButton("Επιβεβαίωση");
        sellOkayButton.setBounds(45, 290, 110, 20);
        add(sellOkayButton);
        sellOkayButton.addActionListener(sellListener);
        sellCancelButton = new JButton("Ακύρωση");
        sellCancelButton.setBounds(155, 290, 110, 20);
        add(sellCancelButton);
        sellCancelButton.addActionListener(sellListener);
    }


/**
    * Η SellListener είναι κλάση , η οποία με τις μεθόδους της χειρίζεται τα 'γεγονότα' που προκύπτουν
    * από τα χειριστήρια
    * Έαν το γεγονός είναι η επιβεβαίωση τότε ενεργοποιείται η διαδικασία πώλησης μετοχών
*/
class SellListener implements ActionListener {
public void actionPerformed(ActionEvent event) {
        Object object = event.getSource();
        if (object == sellCancelButton) {
           hideSellPanel();
           stockTrade.repaint();
        } else if (object == sellOkayButton) {
           // Συλλογή των στοιχείων της πώλησης από τα πεδία της οθόνης
           String companyName = sellCompanyCombo.getSelectedItem().toString();
           String categoryName = sellCategoryCombo.getSelectedItem().toString();
```

```
            String companyReference = sellCompanyReferenceTextField.getText();
            OrderDetailParam order = new OrderDetailParam();
            order.accountNumber = new String(stockTrade.accountNumber);
            order.company = new String(companyName);
            order.categoryName = new String(categoryName);
            order.companyReference = new String(companyReference);
            try {
               int numShares = Integer.parseInt(sellNumSharesTextField.getText());
               order.no = numShares;
               int lotSize =Integer.parseInt(sellLotSize.getSelectedItem().toString());
               order.lotSize = lotSize;
               order.price = Float.valueOf(sellMinValueTextField.getText()).floatValue();
               String temp = ShareBrokerIntfFinder.find(stockTrade.brokerName,
                  stockTrade.connection).sell(trader, order);
               ShowMessageDialog("Η εντολή σας είναι " + temp);
            } catch (NumberFormatException e) {
               JOptionPane.showMessageDialog(null,"Αναμένονται αριθμητικά δεδομένα"
                  + "για τα ακόλουθα πεδία\n" + " Μερίδια\n Ελάχιστη τιμή \n Έγκυρη έως");
               showSellPanel();
               stockTrade.repaint();
               return;
            } catch (Exception e) {
               e.printStackTrace();
            }
            hideSellPanel();
            stockTrade.repaint();
         }
      }
   }


   /**
    * Αρχικοποιεί τα πεδία της οθόνης
    */
   public void showSellPanel() {
      sellNumSharesTextField.setText("");
      sellMinValueTextField.setText("");
      sellValidTillTextField.setText("");
      setVisible(true);
   }

  /**
    * Εξαφανίζει την οθόνη από το προσκήνιο
    */
   public void hideSellPanel() {
      setVisible(false);
   }

   /**
    * Εμφανίζει το μύνημα με το οποίο αρχικοποιήθηκε
    */
   void ShowMessageDialog(String message) {
      (new MessageDialog(stockTrade, true, message)).show();
   }
}
```

12. CancelOrderGUI.java

```
package sharebroker;
```

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
import javax.swing.table.*;


/**
*Δημιουργεί την οθόνη με την οποιά ο χρήστης ακυρώνει μια εντολή που έχει θέσει προηγουμένως
*/
public class CancelOrderGUI extends JPanel {
   JLabel titleLabel, orderNumberLabel;
   JTextField orderNumberTextField;
   JButton cancelOrderButton, cancelButton;
   CancelOrderListener cancelOrderListener = new CancelOrderListener();
   StockTrade stockTrade;


/**
    * Ο δημιουργός  παίρνει σαν παράμετρο το αντικείμενο StockTrade object. Στη συνέχεια δημιουργεί
    *την οθόνη και ενεργοποιεί τα χειριστήρια της μέσω του event listener
 */
   public CancelOrderGUI(StockTrade stockTrade) {

      // this.bankClient = bankClient;
      this.stockTrade = stockTrade;
      setLayout(null);
      setVisible(false);
      titleLabel = new JLabel("Ακύρωση Εντολής");
      titleLabel.setFont(new Font("Times New Roman", Font.BOLD, 20));
      titleLabel.setBounds(10, 20, 250, 30);
      add(titleLabel);
      orderNumberLabel = new JLabel("Αριθμός εντολής");
      orderNumberLabel.setBounds(10, 60, 120, 20);
      add(orderNumberLabel);
      orderNumberTextField = new JTextField();
      orderNumberTextField.setBounds(140, 60, 300, 20);
      add(orderNumberTextField);
      cancelOrderButton = new JButton("Ακύρωση Εντολής");
      cancelOrderButton.setBounds(140, 100, 150, 20);
      cancelOrderButton.addActionListener(cancelOrderListener);
      add(cancelOrderButton);
      cancelButton = new JButton("Ακύρωση");
      cancelButton.setBounds(310, 100, 130, 20);
      add(cancelButton);
      cancelButton.addActionListener(cancelOrderListener);
   }


/**
    * Η CancellListener είναι κλάση , η οποία με τις μεθόδους της χειρίζεται τα 'γεγονότα' που
προκύπτουν
    * από τα χειριστήρια
    * Έαν το γεγονός είναι η επιβεβαίωση τότε ενεργοποιείται η διαδικασία ακύρωσης εντολής
*/
   class CancelOrderListener implements ActionListener {
   public void actionPerformed(ActionEvent event) {
         Object object = event.getSource();

         if (object == cancelButton) {
```

```
            hideCancelOrderPanel();
            stockTrade.repaint();
        } else if (object == cancelOrderButton) {
            try {
                ShareBrokerIntfFinder.find(stockTrade.brokerName,
                    stockTrade.connection).cancelOrder(orderNumberTextField.getText());
                hideCancelOrderPanel();
                stockTrade.repaint();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
  }

  /**
   * Αρχικοποιεί το πεδίο της οθόνης
  */
  void showCancelOrderPanel() {
    orderNumberTextField.setText("");
    setVisible(true);
  }

/**
   *Εξαφανίζει την οθόνη
   */
  void hideCancelOrderPanel() {
    setVisible(false);
  }

/**
  Εμφανίζει το αντίστοιχο μύνημα με το οποίο αρχιοποιήθηκε
   */
  void ShowMessageDialog(String message) {
    (new MessageDialog(stockTrade, true, message)).show();
  }
}
```

13. LoginGUI.java

```
package sharebroker;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
import javax.swing.table.*;

/**
 * Η LoginGUI δημιουργεί την οθόνη που διαχειρίζεται την είσοδο του πελάτη στην εφαρμογή
 */
public class LoginGUI extends JPanel {
  JLabel titleLabel, accountNumberLabel, passwordLabel;
  JTextField accountNumberTextField;
  JPasswordField passwordTextField;
  JButton loginButton, cancelButton;
  LoginListener loginListener = new LoginListener();
  StockTrade stockTrade;
```

```
/**
   * Ο δημιουργός  παίρνει σαν παράμετρο το αντικείμενο StockTrade object. Στη συνέχεια δημιουργεί
   *την οθόνη και ενεργοποιεί τα χειριστήρια της μέσω του event listener
 */
  public LoginGUI(StockTrade stockTrade) {
      this.stockTrade = stockTrade;
     setLayout(null);
     setVisible(false);
     titleLabel = new JLabel("Είσοδος στην Τράπεζα");
     titleLabel.setFont(new Font("Times New Roman", Font.BOLD, 20));
     titleLabel.setBounds(10, 20, 250, 30);
     add(titleLabel);
     accountNumberLabel = new JLabel("Αριθμός Λογαριασμού");
     accountNumberLabel.setBounds(10, 60, 150, 20);
     add(accountNumberLabel);
     accountNumberTextField = new JTextField();
     accountNumberTextField.setBounds(140, 60, 150, 20);
     add(accountNumberTextField);
     passwordLabel = new JLabel("Κωδικός");
     passwordLabel.setBounds(10, 90, 150, 20);
     add(passwordLabel);
     passwordTextField = new JPasswordField();
     passwordTextField.setBounds(140, 90, 150, 20);
     passwordTextField.setEchoChar('*');
     add(passwordTextField);
     loginButton = new JButton("Επιβεβαίωση");
     loginButton.setBounds(45, 140, 110, 20);
     loginButton.addActionListener(loginListener);
     add(loginButton);
     cancelButton = new JButton("Ακύρωση");
     cancelButton.setBounds(155, 140, 110, 20);
     add(cancelButton);
     cancelButton.addActionListener(loginListener);
  }


/**
   * Η LoginListener είναι κλάση , η οποία με τις μεθόδους της χειρίζεται τα 'γεγονότα' που προκύπτουν
   * από τα χειριστήρια
   * Έαν το γεγονός είναι η επιβεβαίωση τότε ενεργοποιείται η διαδικασία εισόδου
*/
  class LoginListener implements ActionListener {
  public void actionPerformed(ActionEvent event) {
       Object object = event.getSource();
       String tempStr = null;
       if (object == cancelButton) {
          hideLoginPanel();
          stockTrade.repaint();
       } else if (object == loginButton) {
          String b = null;
          try {
             b =
BankServiceIntfFinder.find(stockTrade.connection).login(accountNumberTextField.getText(),
                passwordTextField.getText());
          } catch (Exception e) {
             e.printStackTrace();
          }
          if (b != null) {
             stockTrade.accountNumber =
```

```
                new String(accountNumberTextField.getText());
            tempStr = stockTrade.accountNumber;
            stockTrade.setTraderName(tempStr);
            stockTrade.userName = new String(b);

            // Χειριστήρια άλλων οθονών που ενεργοποιούνται με την επιτυχημένη είσοδο του πελάτη
            stockTrade.miListShareBroker.setEnabled(true);
            stockTrade.miChangePassword.setEnabled(true);
            stockTrade.changePasswordButtonTB.setEnabled(true);
            stockTrade.miAddShares.setEnabled(true);
            stockTrade.miAddFunds.setEnabled(true);
            stockTrade.miListAccount.setEnabled(true);
            stockTrade.addSharesButtonTB.setEnabled(true);
            stockTrade.addFundsButtonTB.setEnabled(true);
            stockTrade.miLogin.setEnabled(false);
            stockTrade.loginButtonTB.setEnabled(false);
            new StartTraderService(stockTrade.userName);
            ShowMessageDialog("Επιτυχημένη Είσοδος");
            hideLoginPanel();
            stockTrade.repaint();
            stockTrade.setTitle("Καλώς ορίσατε στη Χρηματαγορά κ."+ stockTrade.userName);
          } else {
            ShowMessageDialog("Άκυρος Κωδικός");
            showLoginPanel();
            stockTrade.repaint();
          }
        }
      }
    }

  /**
    * Αρχικοποίηση των πεδίων
  */
    void showLoginPanel() {
      accountNumberTextField.setText("");
      passwordTextField.setText("");
      setVisible(true);
    }

/**
    * Εξαφάνιση της οθόνης
*/
    void hideLoginPanel() {
      setVisible(false);
    }

/**
    *Εμφάνιση μυνήματος
    */
    void ShowMessageDialog(String message) {
      (new MessageDialog(stockTrade, true, message)).show();
    }
}
```

14. ChangePasswordGUI.java

```
package sharebroker;
import java.awt.*;
import java.awt.event.*;
```

```
import javax.swing.*;

/**
*Δημιουργεί την οθόνη με την οποιά ο χρήστης μπορεί να αλλάξει τον κωδικό εισόδου του που του έχει
*δωθεί αρχικά
*/
public class ChangePasswordGUI extends JPanel {
    JLabel titleLabel, accountNumberLabel, oldPasswordLabel, newPasswordLabel;
   JTextField accountNumberTextField;
   JPasswordField oldPasswordTextField, newPasswordTextField;
   JButton changeOkButton, changeCancelButton;
   ChangeListener changeListener = new ChangeListener();
   StockTrade stockTrade;


/**
    * Ο δημιουργός  παίρνει σαν παράμετρο το αντικείμενο StockTrade object. Στη συνέχεια δημιουργεί
    *την οθόνη και ενεργοποιεί τα χειριστήρια της μέσω του event listener
 */
   public ChangePasswordGUI(StockTrade stockTrade) {
      this.stockTrade = stockTrade;
      setLayout(null);
      setVisible(false);
      titleLabel = new JLabel("Αλλαγή Κωδικού");
      titleLabel.setFont(new Font("Times New Roman", Font.BOLD, 20));
      titleLabel.setBounds(10, 20, 250, 30);
      add(titleLabel);
      accountNumberLabel = new JLabel("Αριθμός Λογαριασμού");
      accountNumberLabel.setBounds(10, 60, 150, 20);
      add(accountNumberLabel);
      accountNumberTextField = new JTextField();
      accountNumberTextField.setBounds(140, 60, 150, 20);
      add(accountNumberTextField);
      oldPasswordLabel = new JLabel("Παλαιός Κωδικός");
      oldPasswordLabel.setBounds(10, 90, 120, 20);
      add(oldPasswordLabel);
      oldPasswordTextField = new JPasswordField();
      oldPasswordTextField.setBounds(140, 90, 150, 20);
      oldPasswordTextField.setEchoChar('*');
      add(oldPasswordTextField);
      newPasswordLabel = new JLabel("Νέος Κωδικός");
      newPasswordLabel.setBounds(10, 120, 150, 20);
      add(newPasswordLabel);
      newPasswordTextField = new JPasswordField();
      newPasswordTextField.setBounds(140, 120, 150, 20);
      newPasswordTextField.setEchoChar('*');
      add(newPasswordTextField);
      changeOkButton = new JButton("Επιβεβαίωση");
      changeOkButton.setBounds(45, 170, 120, 20);
      add(changeOkButton);
      changeOkButton.addActionListener(changeListener);
      changeCancelButton = new JButton("Ακύρωση");
      changeCancelButton.setBounds(155, 170, 120, 20);
      add(changeCancelButton);
      changeCancelButton.addActionListener(changeListener);
   }

/**
```

```
    * Η ChangeListener είναι κλάση , η οποία με τις μεθόδους της χειρίζεται τα 'γεγονότα' που
προκύπτουν
    * από τα χειριστήρια
    * Έαν το γεγονός είναι η επιβεβαίωση τότε ενεργοποιείται η διαδικασία αλλαγής κωδικού
*/
   class ChangeListener implements ActionListener {
   public void actionPerformed(ActionEvent event) {
         Object object = event.getSource();
         if (object == changeCancelButton) {
            hideChangePWDPanel();
            stockTrade.repaint();
         } else if (object == changeOkButton) {
            boolean b = false;
            try {
              b =
BankServiceIntfFinder.find(stockTrade.connection).changePassword(stockTrade.accountNumber,
                  oldPasswordTextField.getText(),
                  newPasswordTextField.getText());
            } catch (Exception e) {
               e.printStackTrace();
            }
            if (b) {
               ShowMessageDialog("Επιτυχής Αλλαγή Κωδικού!!!");
               stockTrade.miLogin.setEnabled(true);
               stockTrade.loginButtonTB.setEnabled(true);
            } else {
               ShowMessageDialog("Εσφαλμένη Αλλαγή Κωδικού!!!");
            }
            hideChangePWDPanel();
            stockTrade.repaint();
         }
      }
   }


   /**
    * Αρχικοποίηση των πεδίων της οθόνης
    */
   public void showChangePWDPanel() {
      accountNumberTextField.setText("");
      oldPasswordTextField.setText("");
      newPasswordTextField.setText("");
      titleLabel.setVisible(true);
      accountNumberLabel.setVisible(true);
      accountNumberTextField.setVisible(true);
      oldPasswordLabel.setVisible(true);
      oldPasswordTextField.setVisible(true);
      newPasswordLabel.setVisible(true);
      newPasswordTextField.setVisible(true);
      changeOkButton.setVisible(true);
      changeCancelButton.setVisible(true);
      setVisible(true);
   }


   /**
    * Εξαφανίζει την οθόνη
    */
   public void hideChangePWDPanel() {
      titleLabel.setVisible(false);
```

```
      accountNumberLabel.setVisible(false);
      accountNumberTextField.setVisible(false);
      oldPasswordLabel.setVisible(false);
      oldPasswordTextField.setVisible(false);
      newPasswordLabel.setVisible(false);
      newPasswordTextField.setVisible(false);
      changeOkButton.setVisible(false);
      changeCancelButton.setVisible(false);
      setVisible(false);
    }


    /**
     *Εμφάνιση  του μυνήματος που έχει ως παράμετρο
*/
    void ShowMessageDialog(String message) {
      (new MessageDialog(stockTrade, true, message)).show();
    }
}
```

15. ListAccountGUI.java

```
package sharebroker;
import net.espeak.util.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
import javax.swing.table.*;


/**
*
 *Είναι η οθόνη που εμφανίζει τον αναλυτικό λογαριασμό ενός πελάτη περιέχοντας το χρηματικό
υπόλοιπο
 *αλλά και και την λίστα των μετοχών του χαρτοφυλακίου του
*/
public class ListAccountGUI extends JPanel {
JLabel titleLabel, accountNameLabel, accountNumberLabel,
        balanceAmountLabel, sharesListLabel;
    JTextField accountNameTextField, accountNumberTextField,
      balanceAmountTextField;
    JScrollPane sharesPane;
    JButton cancelButton;
    ListListener listListener = new ListListener();
    StockTrade stockTrade;
    String[] columnNames = {
      "Κατηγορία", "Εταιρεία", "Σύντμηση", "Μερίδια", "Τιμή"
    };
    JTable sharesTableView;
    Object[][] sharesData = null;
    AbstractTableModel sharesDataModel = new AbstractTableModel(){
/* Οι μέθοδοι που ακολουθούν είναι απαραίτητες για την αρχικοποίηση της abstract μεθόδου
      AbstractTableModel */

     public int getColumnCount() {
        return columnNames.length;
      }

    public int getRowCount() {
```

```
        if (brokerData == null) {
            return 0;
        } else {
            return sharesData.length;
        }
    }

  public Object getValueAt(int row, int col) {
        return sharesData[row][col];
    }

  public String getColumnName(int column) {
        return columnNames[column];
    }

    public Class getColumnClass(int c) {
        if (sharesData == null) {
            return null;
        } else {
            return getValueAt(0, c).getClass();
        }
    }
public boolean isCellEditable(int row, int col) {
        return col == 0;
    }

public void setValueAt(Object aValue, int row, int column) {
        sharesData[row][column] = aValue;
    }
  };

/**
   * Ο δημιουργός  παίρνει σαν παράμετρο το αντικείμενο StockTrade object. Στη συνέχεια δημιουργεί
   *την οθόνη και ενεργοποιεί τα χειριστήρια της μέσω του event listener
 */
  public ListAccountGUI(StockTrade stockTrade) {
    this.stockTrade = stockTrade;
    setLayout(null);
    setVisible(false);
    titleLabel = new JLabel("Λεπτομέρειες Λογαριασμού");
    titleLabel.setFont(new Font("Times New Roman", Font.BOLD, 20));
    titleLabel.setBounds(10, 20, 270, 30);
    add(titleLabel);
    accountNameLabel = new JLabel("Κάτοχος");
    accountNameLabel.setBounds(10, 60, 150, 20);
    add(accountNameLabel);
    accountNameTextField = new JTextField();
    accountNameTextField.setBounds(140, 60, 150, 20);
    add(accountNameTextField);
    accountNumberLabel = new JLabel("Αριθμός Λογαριασμού");
    accountNumberLabel.setBounds(10, 90, 150, 20);
    add(accountNumberLabel);
    accountNumberTextField = new JTextField();
    accountNumberTextField.setBounds(140, 90, 150, 20);
    add(accountNumberTextField);
    balanceAmountLabel = new JLabel("Έχετε:");
    balanceAmountLabel.setBounds(10, 120, 150, 20);
    add(balanceAmountLabel);
    balanceAmountTextField = new JTextField();
    balanceAmountTextField.setBounds(140, 120, 150, 20);
```

```
        add(balanceAmountTextField);
        sharesListLabel = new JLabel("Οι Μετοχές Αναλυτικά:");
        sharesListLabel.setBounds(10, 150, 150, 20);
        add(sharesListLabel);
        sharesTableView = new JTable(sharesDataModel);
        sharesTableView.setRowHeight(20);
        sharesPane = new JScrollPane(sharesTableView);
        sharesPane.setBounds(10, 180, 380, 200);
        add(sharesPane);
        cancelButton = new JButton("Επιστροφή");
        cancelButton.setBounds(10, 400, 120, 20);
        add(cancelButton);
        cancelButton.addActionListener(listListener);
    }


/**
     * Η ListListener είναι κλάση , η οποία με τις μεθόδους της χειρίζεται τα 'γεγονότα' που προκύπτουν
     * από τα χειριστήρια
     * Έαν το γεγονός είναι η επιβεβαίωση τότε ενεργοποιείται η διαδικασία αλλαγής κωδικού
*/
    class ListListener implements ActionListener {
    public void actionPerformed(ActionEvent event) {
            Object object = event.getSource();
            if (object == cancelButton) {
              hideListAccountPanel();
              stockTrade.repaint();
            }
        }
    }



    /**
     * Δημιουργεί μια λίστα με τις μετοχές και το υπόλοιπο του κάθε πελάτη
     */
    void showListAccountPanel() {
      ESArray shareVector = null;

      accountNameTextField.setText(stockTrade.userName);
      try {
        accountNumberTextField.setText(stockTrade.accountNumber);
        String balAmount = new
Float((BankServiceIntfFinder.find(stockTrade.connection).listBalance(stockTrade.accountNumber))).toS
tring();
        balanceAmountTextField.setText(balAmount);
        shareVector =
BankServiceIntfFinder.find(stockTrade.connection).listShares(stockTrade.accountNumber);
      } catch (Exception e) {
        e.printStackTrace();
      }
      ShareDetailParam share;
      String str;
      String categoryName, companyName, companyRef, numShare, pricePerShare;
      sharesData = new Object[shareVector.size()][5];
      for (int i = 0; i < shareVector.size(); i++) {
        share = (ShareDetailParam)(shareVector.elementAt(i));
        System.out.println(share.categoryName);
        categoryName = new String(share.categoryName);
        System.out.println(categoryName);
```

```
            companyName = new String(share.company.companyName);
            System.out.println(companyName);
            companyRef = new String(share.companyReference);
            System.out.println(companyRef);
            numShare = new Integer(share.numShares).toString();
            System.out.println(numShare);
            pricePerShare = new Float(share.pricePerShare).toString();
            System.out.println(pricePerShare);
            sharesData[i][0] = new String(categoryName);
            sharesData[i][1] = new String(companyName);
            sharesData[i][2] = new String(companyRef);
            sharesData[i][3] = new String(numShare);
            sharesData[i][4] = new String(pricePerShare);
        }
        sharesTableView.updateUI();
        setVisible(true);
    }

    /**
     * Εξαφανίζει την οθόνη
     */
    void hideListAccountPanel() {
        setVisible(false);
    }

    /*
     * Εμφάνιση του μηνύματος που υπάρχει σαν παράμετρος
     */
void ShowMessageDialog(String message) {
        (new MessageDialog(stockTrade, true, message)).show();
    }
}
```

16. ListBrokersGUI.java

```
package sharebroker;
import net.espeak.util.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
import javax.swing.table.*;


/**
 * Η συγκεκριμένη κλάση χρησιμοποιείται για παρουσίαση και την επιλογή των χρηματιστών
 */
public class ListBrokersGUI extends JPanel {
    JLabel titleLabel, brokerListLabel;
    JScrollPane brokersPane;
    JButton okButton, cancelButton;
    ListListener listListener = new ListListener();
    StockTrade stockTrade;
    String[] columnNames = {
        "Χρηματιστής", "Οδός", "Πόλη", "Χώρα", "Κατάταξη"
    };
    JTable brokerView;
```

```
Object[][] brokerData = null;
AbstractTableModel brokerDataModel = new AbstractTableModel(){
/* Οι μέθοδοι που ακολουθούν είναι απαραίτητες για την αρχικοποίηση της abstract μεθόδου
  AbstractTableModel */

 public int getColumnCount() {
     return columnNames.length;
   }

public int getRowCount() {
     if (brokerData == null) {
       return 0;
     } else {
       return brokerData.length;
     }
   }

 public Object getValueAt(int row, int col) {
       return brokerData[row][col];
    }

 public String getColumnName(int column) {
       return columnNames[column];
    }

   public Class getColumnClass(int c) {
     if (brokerData == null) {
       return null;
     } else {
       return getValueAt(0, c).getClass();
     }
   }
public boolean isCellEditable(int row, int col) {
     return col == 0;
   }

public void setValueAt(Object aValue, int row, int column) {
     brokerData[row][column] = aValue;
   }
};

/**
  * Ο δημιουργός  παίρνει σαν παράμετρο το αντικείμενο StockTrade object. Στη συνέχεια δημιουργεί
  *την οθόνη και ενεργοποιεί τα χειριστήρια της μέσω του event listener
 */
  public ListBrokersGUI(StockTrade stockTrade) {
    this.stockTrade = stockTrade;
    setLayout(null);
    setVisible(false);
    titleLabel = new JLabel("Λίστα Χρηματιστών");
    titleLabel.setFont(new Font("Times New Roman", Font.BOLD, 20));
    titleLabel.setBounds(10, 20, 270, 30);
    add(titleLabel);
    brokerListLabel = new JLabel("Λεπτομέρειες");
    brokerListLabel.setBounds(10, 60, 150, 20);
    add(brokerListLabel);
    brokerView = new JTable(brokerDataModel);
    brokerView.setRowHeight(20);
    brokersPane = new JScrollPane(brokerView);
    brokersPane.setBounds(10, 80, 380, 200);
```

```
        add(brokersPane);
        okButton = new JButton("Επιβεβαίωση");
        okButton.setBounds(10, 300, 110, 20);
        add(okButton);
        okButton.addActionListener(listListener);
        cancelButton = new JButton("Ακύρωση");
        cancelButton.setBounds(120, 300, 110, 20);
        add(cancelButton);
        cancelButton.addActionListener(listListener);
    }


/**
     * Η ListListener είναι κλάση , η οποία με τις μεθόδους της χειρίζεται τα 'γεγονότα' που προκύπτουν
     * από τα χειριστήρια
     * Έαν το γεγονός είναι η επιβεβαίωση τότε ενεργοποιείται η διαδικασία επιλογής χρηματιστή
*/
    class ListListener implements ActionListener {
    public void actionPerformed(ActionEvent event) {
          Object object = event.getSource();
          if (object == cancelButton) {
            hideListBrokersPanel();
            stockTrade.repaint();
          } else if (object == okButton) {
            stockTrade.brokerName =
              (String)brokerDataModel.getValueAt(brokerView.getSelectedRow(),0);
            if (stockTrade.brokerName != null) {
              stockTrade.enableBuyAndSell();
              stockTrade.miBrokerList.setEnabled(true);
              stockTrade.miBuyShares.setEnabled(true);
              stockTrade.miSellShares.setEnabled(true);
              stockTrade.buySharesButtonTB.setEnabled(true);
              stockTrade.sellSharesButtonTB.setEnabled(true);
              stockTrade.miCancelOrder.setEnabled(true);
              stockTrade.cancelOrderButtonTB.setEnabled(true);
            }
            hideListBrokersPanel();
            stockTrade.repaint();
          }
        }
    }

/*
*Εμφανίζει την λίστα των χρηματιστών
*/
void showListBrokersPanel() {
  try {
        brokerData =
          ShareBrokerIntfFinder.findBrokers(stockTrade.connection);
      } catch (Exception e) {
        e.printStackTrace();
      }
      brokerView.updateUI();
      setVisible(true);
    }

    /**
     * Εξαφανίζει την οθόνη
     */
    void hideListBrokersPanel() {
```

```
        setVisible(false);
    }

/**
*Εμφανίζει το μήνυμα που έχει σαν παράμετρο
*/
    void ShowMessageDialog(String message) {
        (new MessageDialog(stockTrade, true, message)).show();
    }
}
```

17. ListBanksGUI.java

```
package sharebroker;
import net.espeak.util.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
import javax.swing.table.*;


/**
 *Σκοπός της κλάσης αυτής είναι η παρουσίαση και η επιλογή των διαθέσιμων υπηρεσιών τραπέζας από
 *τον πελάτη
 */

public class ListBanksGUI extends JPanel {
    JLabel titleLabel, bankListLabel;
    JScrollPane banksPane;
    JButton okButton, cancelButton;
    ListListener listListener = new ListListener();
    StockTrade stockTrade;
    String[] columnNames = {
        "Όνομα Τράπεζας", "Οδός", "Πόλη", "Χώρα"
    };
    JTable bankView;
    Object[][] bankData = null;
    AbstractTableModel bankDataModel = new AbstractTableModel() {
/* Οι μέθοδοι που ακολουθούν είναι απαραίτητες για την αρχικοποίηση της abstract μεθόδου
     AbstractTableModel */

public int getColumnCount() {
        return columnNames.length;
    }

public int getRowCount() {
        if (bankData == null) {
            return 0;
        } else {
            return bankData.length;
        }
    }


public Object getValueAt(int row, int col) {
        return bankData[row][col];
    }
```

```
public String getColumnName(int column) {
        return columnNames[column];
    }

public Class getColumnClass(int c) {
        if (bankData == null) {
          return null;
        } else {
          return getValueAt(0, c).getClass();
        }
    }

public boolean isCellEditable(int row, int col) {
        return col == 0;
    }

public void setValueAt(Object aValue, int row, int column) {
        bankData[row][column] = aValue;
    }
  };


/**
   * Ο δημιουργός  παίρνει σαν παράμετρο το αντικείμενο StockTrade object. Στη συνέχεια δημιουργεί
   *την οθόνη και ενεργοποιεί τα χειριστήρια της μέσω του event listener
 */
  public ListBanksGUI(StockTrade stockTrade) {
    this.stockTrade = stockTrade;
    setLayout(null);
    setVisible(false);
    titleLabel = new JLabel("Λίστα Τραπεζών");
    titleLabel.setFont(new Font("Times New Roman", Font.BOLD, 20));
    titleLabel.setBounds(10, 20, 250, 30);
    add(titleLabel);
    bankListLabel = new JLabel("Λεπτομέρειες");
    bankListLabel.setBounds(10, 60, 150, 20);
    add(bankListLabel);
    bankView = new JTable(bankDataModel);
    bankView.setRowHeight(20);
    banksPane = new JScrollPane(bankView);
    banksPane.setBounds(10, 80, 380, 200);
    add(banksPane);
    okButton = new JButton("Επιβεβαίωση");
    okButton.setBounds(10, 300, 110, 20);
    add(okButton);
    okButton.addActionListener(listListener);
    cancelButton = new JButton("Ακύρωση");
    cancelButton.setBounds(120, 300, 110, 20);
    add(cancelButton);
    cancelButton.addActionListener(listListener);
  }


/**
   * Η ListListener είναι κλάση , η οποία με τις μεθόδους της χειρίζεται τα 'γεγονότα' που προκύπτουν
   * από τα χειριστήρια
   * Έαν το γεγονός είναι η επιβεβαίωση τότε ενεργοποιείται η διαδικασία επιλογής τράπεζας
*/
```

```
class ListListener implements ActionListener {
 public void actionPerformed(ActionEvent event) {
        Object object = event.getSource();

        if (object == cancelButton) {
           hideListBanksPanel();
           stockTrade.repaint();
        } else if (object == okButton) {
           hideListBanksPanel();
           stockTrade.repaint();
        }
     }
   }


   /**
    * Δημιουργεί την λίστα με τις διαθέσιμες τράπεζες
    */
   void showListBanksPanel() {
      try {
         bankData =
            BankServiceIntfFinder.findBankProperties(stockTrade.connection);
      } catch (Exception e) {
         e.printStackTrace();
      }
      bankView.updateUI();
      setVisible(true);
   }

/*Εξαφανίζει την οθόνη*/
void hideListBanksPanel() {
      setVisible(false);
   }

   /**
      *Εμφανίζει το μήνυμα της παραμέτρου
      */
   void ShowMessageDialog(String message) {
      (new MessageDialog(stockTrade, true, message)).show();
   }
}
```

18. StockTrade.java

```
package sharebroker;
import java.awt.*;
import java.io.*;
import java.util.*;
import java.awt.image.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import net.espeak.jesi.*;


/**
*Η κλάση αυτή δημιουργεί την αρχική οθόνη της εφαρμογής και διαχειρίζεται όλα τα χειριστήρια που
*περιέχει, καθώς επίσης ενεργοποιεί όλες τις κλάσεις οθονών που περιγράφηκαν παραπάνω
 *
```

```
public class StockTrade extends JFrame {
   JMenuItem miSelectBank, miOpenAccount, miChangePassword, miLogin,
      miAddShares, miAddFunds, miListAccount, miListFunds, MiListShares,
      miListTransactions, miBrokerList, miBuyShares, miSellShares,
      miCancelOrder, miRegisterInterest, miCreateVocabulary, miHelp,
      miAbout, miListShareBroker;
   JButton openAccountButtonTB, changePasswordButtonTB, loginButtonTB,
      addSharesButtonTB, addFundsButtonTB;
   JButton buySharesButtonTB, sellSharesButtonTB, folderButtonTB,
      cancelOrderButtonTB;
   JButton registerInterestButtonTB, createVocabularyButtonTB, helpButtonTB;
   String[] shareBrokerNames;
   int index = 0;
   private static String traderName = null;
   BankListener bankListener = new BankListener();
   BrokerListener brokerListener = new BrokerListener();
   HelpListener helpListener = new HelpListener();
   WindowListener windowListener = new WindowListener();
   ListBanksGUI selectBankPanel;
   OpenAccountGUI openAccountPanel;
   ChangePasswordGUI changePasswordPanel;
   LoginGUI loginPanel;
   AddSharesGUI addSharesPanel;
   AddFundsGUI addFundsPanel;
   BuySharesGUI buySharesPanel;
   SellSharesGUI sellSharesPanel;
   ListSharesGUI listSharesPanel;
   ListAccountGUI listAccountPanel;
   CancelOrderGUI cancelOrderPanel;
   ListBrokersGUI listShareBrokersPanel;
   int currentscreen = -1;
   String accountNumber;
   String userName;
   String brokerName;
   ESConnection connection = null;


   /**
    * Η κλάση WindowListener χειρίζεται το κλείσιμο του παραθύρου
    */
   class WindowListener extends WindowAdapter {
         public void windowClosing(WindowEvent e) {
         System.exit(0);
      }
   }

public static String getTraderName() {
      return traderName;
   }

public static void setTraderName(String userName) {
      traderName = new String(userName);
   }


/**
* Η BankListener ελέγχει τα χειριστήρια της πρώτης λίστας επιλογών που ασχολείται με την τράπεζα
*/
   class BankListener implements ActionListener {
```

```
public void actionPerformed(ActionEvent event) {
  Object object = event.getSource();

  switch (currentscreen) {
  case 0:
    selectBankPanel.hideListBanksPanel();
    repaint();
    break;
  case 1:
    openAccountPanel.hideOpenAccountPanel();
    repaint();
    break;
  case 2:
    changePasswordPanel.hideChangePWDPanel();
    repaint();
    break;
  case 3:
    loginPanel.hideLoginPanel();
    repaint();
    break;
  case 4:
    addSharesPanel.hideAddSharesPanel();
    repaint();
    break;
  case 5:
    addFundsPanel.hideAddFundsPanel();
    repaint();
    break;
  case 6:
    listAccountPanel.hideListAccountPanel();
    repaint();
    break;
  case 7:
    listSharesPanel.hideQueryPanel();
    repaint();
    break;
  case 8:
    buySharesPanel.hideBuyPanel();
    repaint();
    break;
  case 9:
    sellSharesPanel.hideSellPanel();
    repaint();
    break;
  case 10:
    listShareBrokersPanel.hideListBrokersPanel();
    repaint();
    break;
  case 11:
    cancelOrderPanel.hideCancelOrderPanel();
    repaint();
    break;
  default:
    break;
  }
  if (object == miSelectBank) {
    System.out.println("Επιλογή Τράπεζας");
    ShowSelectBankGUI();
    currentscreen = 0;
  }
```

```
        if ((object == miOpenAccount)
            || (object == openAccountButtonTB)) {
            System.out.println("Άνοιγμα Λογαριασμού");
            ShowOpenAccountGUI();
            currentscreen = 1;
        } else if ((object == miChangePassword)
    || (object == changePasswordButtonTB)) {
            System.out.println("Αλλαγή Κωδικού");
            ShowChangePasswordGUI();
            currentscreen = 2;
        } else if ((object == miLogin) || (object == loginButtonTB)) {
            System.out.println("Είσοδος");
            ShowLoginGUI();
            currentscreen = 3;
        } else if ((object == miAddShares)
    || (object == addSharesButtonTB)) {
            System.out.println("Προσθήκη Μετοχών");
            ShowAddSharesGUI();
            currentscreen = 4;
        } else if ((object == miAddFunds)
    || (object == addFundsButtonTB)) {
            System.out.println("Προσθήκη Κεφαλαίων");
            ShowAddFundsGUI();
            currentscreen = 5;
        } else if (object == miListAccount) {
            System.out.println("Λίστα Λογαριασμού");
            ShowListAccountPanel();
            currentscreen = 6;
        }
      }
    }


/**
* Η BrokerListener ελέγχει τα χειριστήρια της δεύτερης λίστας επιλογών που ασχολείται με τον
*Χρηματιστή
*/

class BrokerListener implements ActionListener {
public void actionPerformed(ActionEvent event) {
        Object object = event.getSource();

        switch (currentscreen) {
        case 0:
            selectBankPanel.hideListBanksPanel();
            repaint();
            break;
        case 1:
            openAccountPanel.hideOpenAccountPanel();
            repaint();
            break;
        case 2:
            changePasswordPanel.hideChangePWDPanel();
            repaint();
            break;
        case 3:
            loginPanel.hideLoginPanel();
            repaint();
            break;
        case 4:
```

```
            addSharesPanel.hideAddSharesPanel();
            repaint();
            break;
         case 5:
            addFundsPanel.hideAddFundsPanel();
            repaint();
            break;
         case 6:
            listAccountPanel.hideListAccountPanel();
            repaint();
            break;
         case 7:
            listSharesPanel.hideQueryPanel();
            repaint();
            break;
         case 8:
            buySharesPanel.hideBuyPanel();
            repaint();
            break;
         case 9:
            sellSharesPanel.hideSellPanel();
            repaint();
            break;
         case 10:
            listShareBrokersPanel.hideListBrokersPanel();
            repaint();
            break;
         case 11:
            cancelOrderPanel.hideCancelOrderPanel();
            repaint();
            break;

         default:
            break;
         }
         if (object == miBrokerList) {
            System.out.println("Λίστα Μετοχών");
            ShowListSharesPanel();
            currentscreen = 7;
         } else if ((object == miBuyShares)
   || (object == buySharesButtonTB)) {
            System.out.println("Αγορά Μετοχών");
            ShowBuySharesPanel();
            currentscreen = 8;
         } else if ((object == miSellShares)
   || (object == sellSharesButtonTB)) {
            System.out.println("Πώληση Μετοχών");
            ShowSellSharesPanel();
            currentscreen = 9;
         } else if ((object == miCancelOrder)
   || (object == cancelOrderButtonTB)) {
            System.out.println("Ακύρωση Εντολής");
            CancelOrderPanel();
            currentscreen = 11;
         } else if ((object == miListShareBroker)) {
            ShowListBrokersPanel();
            currentscreen = 10;
         }
      }
   }
```

```
/**
* Η HelpListener ελέγχει τα χειριστήρια της τρίτης λίστας επιλογών που ασχολείται με την Βοήθεια
*/
   class HelpListener implements ActionListener {
   public void actionPerformed(ActionEvent event) {
         Object object = event.getSource();

         if ((object == miHelp) || (object == helpButtonTB)) {
            System.out.println("Βοήθεια");
         }
         if (object == miAbout) {
            System.out.println("Για την εφαρμογή");
         }
      }
   }


public void ShowSelectBankGUI() {
      selectBankPanel.showListBanksPanel();
   }


public void ShowOpenAccountGUI() {
      openAccountPanel.showOpenAccountPanel();
   }


public void ShowChangePasswordGUI() {
      changePasswordPanel.showChangePWDPanel();
   }


public void ShowLoginGUI() {
      loginPanel.showLoginPanel();
   }


public void ShowAddSharesGUI() {
      addSharesPanel.showAddSharesPanel();
   }


public void ShowAddFundsGUI() {
      addFundsPanel.showAddFundsPanel();
   }


public void ShowListAccountPanel() {
      listAccountPanel.showListAccountPanel();
   }


public void ShowBuySharesPanel() {
      buySharesPanel.showBuyPanel();
   }
```

```
public void ShowSellSharesPanel() {
      sellSharesPanel.showSellPanel();
   }

public void ShowListSharesPanel() {
      listSharesPanel.showQueryPanel();
   }

public void ShowListBrokersPanel() {
      listShareBrokersPanel.showListBrokersPanel();
   }

public void CancelOrderPanel() {
     cancelOrderPanel.showCancelOrderPanel();
   }


   /**
    *Ο δημιουργός  δημιουργεί την βασική οθόνη αρχικά κα στη συνέχει προσθέτει όλα τα απαραίτητα
    * μενού,υπομενού, κουμπιά με τα αντίστοιχα χειριστήρια
    */
   public StockTrade() {
      JMenuBar mainMenuBar;
      JMenu bankMenu, brokerMenu, adminMenu, helpMenu, quitMenu;
      JToolBar toolBar;

      try {
         connection = new ESConnection("stocktrade.pr");
         (connection.getDefaultServiceHandler()).setNumThreads(4);
      } catch (Exception e) {}
      getContentPane().setLayout(null);
      addWindowListener(windowListener);
      Image titleImage = Toolkit.getDefaultToolkit().getImage("Anim.gif");

      setIconImage(titleImage);
      toolBar = new JToolBar();
      addButtons(toolBar);
      toolBar.setBounds(0, 0, 500, 35);
      getContentPane().add(toolBar);
      mainMenuBar = new JMenuBar();
      setJMenuBar(mainMenuBar);

      // Το πρώτα  menu της  εφαρμογής
      bankMenu = new JMenu("Τράπεζα");
      mainMenuBar.add(bankMenu);
      brokerMenu = new JMenu("Χρηματιστής");
      mainMenuBar.add(brokerMenu);
      helpMenu = new JMenu("Βοήθεια");
      mainMenuBar.add(helpMenu);

      // Στοιχεία του μενού της τράπεζας
      miSelectBank = new JMenuItem("Επιλογή Τράπεζας");
      bankMenu.add(miSelectBank);
      miSelectBank.addActionListener(bankListener);
      miOpenAccount = new JMenuItem("Άνοιγμα Λογαριασμού");
      bankMenu.add(miOpenAccount);
      miOpenAccount.addActionListener(bankListener);
```

```
miChangePassword = new JMenuItem("Αλλαγή Κωδικού");
bankMenu.add(miChangePassword);
miChangePassword.addActionListener(bankListener);
miChangePassword.setEnabled(false);
miLogin = new JMenuItem("Είσοδος");
bankMenu.add(miLogin);
miLogin.addActionListener(bankListener);
miAddShares = new JMenuItem("Προσθήκη Μετοχών");
bankMenu.add(miAddShares);
miAddShares.addActionListener(bankListener);
miAddShares.setEnabled(false);
miAddFunds = new JMenuItem("Προσθήκη Κεφαλαίων");
bankMenu.add(miAddFunds);
miAddFunds.addActionListener(bankListener);
miAddFunds.setEnabled(false);
miListAccount = new JMenuItem("Λίστα Λογαριασμού");
bankMenu.add(miListAccount);
miListAccount.addActionListener(bankListener);
miListAccount.setEnabled(false);

//Στοιχεία του μενού του χρηματιστή
miListShareBroker = new JMenuItem("Επιλογή Χρ/στή");
brokerMenu.add(miListShareBroker);
miListShareBroker.addActionListener(brokerListener);
miListShareBroker.setEnabled(false);
miBrokerList = new JMenuItem("Λίστα Μετοχών");
brokerMenu.add(miBrokerList);
miBrokerList.addActionListener(brokerListener);
miBrokerList.setEnabled(false);
miBuyShares = new JMenuItem("Αγορά Μετοχών");
brokerMenu.add(miBuyShares);
miBuyShares.addActionListener(brokerListener);
miBuyShares.setEnabled(false);
miSellShares = new JMenuItem("Πώληση Μετοχών");
brokerMenu.add(miSellShares);
miSellShares.addActionListener(brokerListener);
miSellShares.setEnabled(false);
miCancelOrder = new JMenuItem("Ακύρωση Εντολής");
brokerMenu.add(miCancelOrder);
miCancelOrder.addActionListener(brokerListener);
miCancelOrder.setEnabled(false);


// Στοιχεία του μενού της Βοήθειας
miHelp = new JMenuItem("Βοήθεια");
helpMenu.add(miHelp);
miHelp.addActionListener(helpListener);
miHelp.setEnabled(false);
miAbout = new JMenuItem("Για την εφαρμογή");
helpMenu.add(miAbout);
miAbout.addActionListener(helpListener);
miAbout.setEnabled(false);
selectBankPanel = new ListBanksGUI(this);
selectBankPanel.setBounds(30, 30, 500, 500);
getContentPane().add(selectBankPanel);
openAccountPanel = new OpenAccountGUI(this);
openAccountPanel.setBounds(30, 30, 500, 500);
getContentPane().add(openAccountPanel);
changePasswordPanel = new ChangePasswordGUI(this);
```

```
        changePasswordPanel.setBounds(30, 30, 500, 500);
        getContentPane().add(changePasswordPanel);
        loginPanel = new LoginGUI(this);
        loginPanel.setBounds(30, 30, 500, 500);
        getContentPane().add(loginPanel);
        addSharesPanel = new AddSharesGUI(this);
        addSharesPanel.setBounds(30, 30, 500, 500);
        getContentPane().add(addSharesPanel);
        addFundsPanel = new AddFundsGUI(this);
        addFundsPanel.setBounds(30, 30, 500, 500);
        getContentPane().add(addFundsPanel);
        listAccountPanel = new ListAccountGUI(this);
        listAccountPanel.setBounds(30, 30, 500, 500);
        getContentPane().add(listAccountPanel);
        listSharesPanel = new ListSharesGUI(this);
        listSharesPanel.setBounds(30, 30, 550, 500);
        getContentPane().add(listSharesPanel);
        listShareBrokersPanel = new ListBrokersGUI(this);
        listShareBrokersPanel.setBounds(30, 30, 550, 500);
        getContentPane().add(listShareBrokersPanel);

        cancelOrderPanel = new CancelOrderGUI(this);
        cancelOrderPanel.setBounds(30,30,550,500);
        getContentPane().add(cancelOrderPanel);
    }


    /**
     * Η μέθοδος αυτή αρχικοποιεί το παράθυρο διαπραγμάτευσης
     */
    public void enableBuyAndSell() {
        buySharesPanel = new BuySharesGUI(this);
        buySharesPanel.setBounds(30, 30, 500, 500);
        getContentPane().add(buySharesPanel);
        sellSharesPanel = new SellSharesGUI(this);
        sellSharesPanel.setBounds(30, 30, 500, 500);
        getContentPane().add(sellSharesPanel);
    }


    /**
    Αρχικοποιεί όλη την εφαρμογή
*/
    public static void main(String[] args) {
        StockTrade window = new StockTrade();

        window.setTitle("Χρηματαγορά");
        window.setSize(600, 600);
        window.setVisible(true);
    }



    /**
     * Προσθέτει κουμπία και χειριστήρια
     */
    protected void addButtons(JToolBar toolBar) {
        openAccountButtonTB = new JButton(new ImageIcon("folder.gif"));
        openAccountButtonTB.setToolTipText("Άνοιγμα Λογαριασμού");
        openAccountButtonTB.addActionListener(bankListener);
```

```
        toolBar.add(openAccountButtonTB);
        changePasswordButtonTB = new JButton(new ImageIcon("cpwd.gif"));
        changePasswordButtonTB.setToolTipText("Αλλαγή Κωδικού");
        changePasswordButtonTB.addActionListener(bankListener);
        changePasswordButtonTB.setEnabled(false);
        toolBar.add(changePasswordButtonTB);
        loginButtonTB = new JButton(new ImageIcon("login.gif"));
        loginButtonTB.setToolTipText("Είσοδος");
        loginButtonTB.addActionListener(bankListener);
        toolBar.add(loginButtonTB);
        addSharesButtonTB = new JButton(new ImageIcon("Create.gif"));
        addSharesButtonTB.setToolTipText("Προσθήκη Μετοχών");
        addSharesButtonTB.addActionListener(bankListener);
        addSharesButtonTB.setEnabled(false);
        toolBar.add(addSharesButtonTB);
        addFundsButtonTB = new JButton(new ImageIcon("addfunds.gif"));
        addFundsButtonTB.setToolTipText("Προσθήκη Κεφαλαίων");
        addFundsButtonTB.addActionListener(bankListener);
        addFundsButtonTB.setEnabled(false);
        toolBar.add(addFundsButtonTB);
        buySharesButtonTB = new JButton(new ImageIcon("buy.gif"));
        buySharesButtonTB.setToolTipText("Αγορά Μετοχών");
        buySharesButtonTB.addActionListener(brokerListener);
        buySharesButtonTB.setEnabled(false);
        toolBar.add(buySharesButtonTB);
        sellSharesButtonTB = new JButton(new ImageIcon("sell.gif"));
        sellSharesButtonTB.setToolTipText("Πώληση Μετοχών");
        sellSharesButtonTB.addActionListener(brokerListener);
        sellSharesButtonTB.setEnabled(false);
        toolBar.add(sellSharesButtonTB);
        cancelOrderButtonTB = new JButton(new ImageIcon("cancelord.gif"));
        cancelOrderButtonTB.setToolTipText("Ακύρωση Εντολής");
        cancelOrderButtonTB.addActionListener(brokerListener);
        cancelOrderButtonTB.setEnabled(false);
        toolBar.add(cancelOrderButtonTB);
          helpButtonTB = new JButton(new ImageIcon("help.gif"));
        helpButtonTB.setToolTipText("Βοήθεια");
        helpButtonTB.addActionListener(helpListener);
        helpButtonTB.setEnabled(false);
        toolBar.add(helpButtonTB);
    }
}
```

19. ListSharesGUI.java

```
package sharebroker;
import net.espeak.util.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
import javax.swing.table.*;


/**
 * Η ListSharesGUI δημιουργεί την οθόνη όπου εμφανίζονται όλες οι μετοχές που είναι διαθέσιμες στην
*χρηματαγορά μέσω των αντίστοιχων χρηματιστών.
 */

public class ListSharesGUI extends JPanel {
```

```
JLabel titleLabel, categoryListLabel, companyListLabel, buyersListLabel,sellersListLabel;
   JList categoryList, companyList;
   JScrollPane categoryPane, companyPane, buyersPane, sellersPane;
   DefaultListModel categoryModel, companyModel;
   JButton queryCancelButton, buyButton, fileButton;
   QueryListener queryListener = new QueryListener();
   DoubleClickListener doubleClickListener = new DoubleClickListener();
   StockTrade stockTrade;
   Vector shareDetailsVector = new Vector(0, 1);
   String selectedCategoryName = null;
   String selectedCompanyName = null;
   String[] columnNames = {"Πακέτο των", "Μερίδια", "Τιμή"};
   JTable buyersTableView, sellersTableView;
   Object[][] buyersData = null;
   Object[][] sellersData = null;
   String[] orderNumberArray;
   AbstractTableModel buyersDataModel = new AbstractTableModel() {
   /* Οι μέθοδοι που ακολουθούν είναι απαραίτητες για την αρχικοποίηση της abstract μεθόδου
     AbstractTableModel */

public int getColumnCount() {
        return columnNames.length;
     }


public int getRowCount() {
        if (buyersData == null) {
           return 0;
        } else {
           return buyersData.length;
        }
     }


public Object getValueAt(int row, int col) {
        return buyersData[row][col];
     }


public String getColumnName(int column) {
        return columnNames[column];
     }


public Class getColumnClass(int c) {
        return getValueAt(0, c).getClass();
     }


public boolean isCellEditable(int row, int col) {
        return col == 0;
     }


public void setValueAt(Object aValue, int row, int column) {
        buyersData[row][column] = aValue;
     }
   };
   AbstractTableModel sellersDataModel = new AbstractTableModel() {
/* Οι μέθοδοι που ακολουθούν είναι απαραίτητες για την αρχικοποίηση της abstract μεθόδου
```

```
    AbstractTableModel */

public int getColumnCount() {
        return columnNames.length;
    }

public int getRowCount() {
        if (sellersData == null) {
            return 0;
        } else {
            return sellersData.length;
        }
    }


public Object getValueAt(int row, int col) {
        return sellersData[row][col];
    }


public String getColumnName(int column) {
        return columnNames[column];
    }


public Class getColumnClass(int c) {
        return getValueAt(0, c).getClass();
    }


public boolean isCellEditable(int row, int col) {
        return col == 0;
    }


public void setValueAt(Object aValue, int row, int column) {
        sellersData[row][column] = aValue;
    }
  };


  /**
   * Ο δημιουργός  παίρνει σαν παράμετρο το αντικείμενο StockTrade object. Στη συνέχεια δημιουργεί
   *την οθόνη και ενεργοποιεί τα χειριστήρια της μέσω του event listener
 */

  public ListSharesGUI(StockTrade stockTrade) {
  this.stockTrade = stockTrade;
    setLayout(null);
    setVisible(false);
    titleLabel = new JLabel("Διαπραγματεύσιμες Εντολές");
    titleLabel.setFont(new Font("Times New Roman", Font.BOLD, 20));
    titleLabel.setBounds(10, 20, 290, 30);
    add(titleLabel);
    categoryListLabel = new JLabel("Κατηγορία");
    categoryListLabel.setBounds(10, 60, 150, 20);
    add(categoryListLabel);
    categoryModel = new DefaultListModel();
    categoryList = new JList(categoryModel);
```

```
        categoryPane = new JScrollPane(categoryList);
        categoryPane.setBounds(10, 90, 250, 200);
        categoryList.addMouseListener(doubleClickListener);
        add(categoryPane);
        companyListLabel = new JLabel("Εταιρεία");
        companyListLabel.setBounds(270, 60, 300, 20);
        add(companyListLabel);
        companyModel = new DefaultListModel();
        companyList = new JList(companyModel);
        companyPane = new JScrollPane(companyList);
        companyPane.setBounds(270, 90, 250, 200);
        companyList.addMouseListener(doubleClickListener);
        add(companyPane);
        buyersListLabel = new JLabel("Αγοραστές");
        buyersListLabel.setBounds(10, 300, 150, 20);
        add(buyersListLabel);
        buyersTableView = new JTable(buyersDataModel);
        buyersTableView.setRowHeight(20);
        buyersPane = new JScrollPane(buyersTableView);
        buyersPane.setBounds(10, 330, 250, 100);
        add(buyersPane);
        sellersListLabel = new JLabel("Πωλητές");
        sellersListLabel.setBounds(270, 300, 150, 20);
        add(sellersListLabel);
        sellersTableView = new JTable(sellersDataModel);
        sellersTableView.setRowHeight(20);
        sellersPane = new JScrollPane(sellersTableView);
        sellersPane.setBounds(270, 330, 250, 100);
        add(sellersPane);
        buyButton = new JButton("Αγορά");
        buyButton.setBounds(10, 440, 110, 20);
        add(buyButton);
        buyButton.addActionListener(queryListener);
        queryCancelButton = new JButton("Ακύρωση");
        queryCancelButton.setBounds(120, 440, 110, 20);
        add(queryCancelButton);
        queryCancelButton.addActionListener(queryListener);
    }

/**
    * Η QueryListener είναι κλάση , η οποία με τις μεθόδους της χειρίζεται τα 'γεγονότα' που προκύπτουν
    * από τα χειριστήρια
    * Έαν το γεγονός είναι η αγορά τότε ενεργοποιείται η διαδικασία αγοράς της αντίστοιχης μετοχής
*/

class QueryListener implements ActionListener {
public void actionPerformed(ActionEvent event) {
        Object object = event.getSource();
        String sellItem, token, orderNumber, lotSize, numShares, price,
            buySell;
        StringTokenizer toks;

        if (object == queryCancelButton) {
            hideQueryPanel();
            stockTrade.repaint();
        } else if (object == buyButton) {
            if (sellersData == null) {
                JOptionPane.showMessageDialog(null,
                    "Δεν επελέγη καμμία μετοχή!");
                showQueryPanel();
```

```
            stockTrade.repaint();
            return;
        }
        hideQueryPanel();
        stockTrade.repaint();
        int index = sellersTableView.getSelectedRow();
        orderNumber = orderNumberArray[index];
        lotSize = (String)sellersData[index][0];
        numShares = (String)sellersData[index][1];
        price = (String)sellersData[index][2];
        stockTrade.buySharesPanel.fillData(selectedCategoryName,
          selectedCompanyName, lotSize, numShares, price);
    }
  }
}

/**
 * Η DoubleClickListener υλοποιεί τις ενέργειες εκείνες που είναι απαραίτητες και συμβαίνουν με το
 *διπλό πάτημα του ποντικιού πάνω στα αντικείμενα  "Κατηγορία Μετοχών" και των "Μετοχών"
 */
class DoubleClickListener extends MouseAdapter {
public void mouseClicked(MouseEvent event) {
    String selItem, token, orderNumber, lotSize, numShares, price,
        buySell;
    StringTokenizer toks;

    if (event.getClickCount() == 2) {
      Object object = event.getSource();

      if (object == categoryList) {
        String categoryName =
          categoryModel.getElementAt(categoryList.getSelectedIndex()).toString();

        companyListLabel.setText("Λίστα μετοχών για τις : "
          + categoryName);
        Vector companyVector = getCompanyByCategory(categoryName,
          shareDetailsVector);

        if (companyModel.size() != 0) {
          companyModel.clear();
        }
        for (int i = 0; i < companyVector.size(); i++) {
          companyModel.addElement(companyVector.elementAt(i).toString());
        }
        companyList.setSelectedIndex(0);
      } else if (object == companyList) {
        selectedCategoryName =
          categoryModel.elementAt(categoryList.getSelectedIndex()).toString();
        selectedCompanyName =
          companyModel.elementAt(companyList.getSelectedIndex()).toString();
        buyersListLabel.setText("Αγοραστές για : "
          + selectedCompanyName);
        sellersListLabel.setText("Πωλητές για : "
          + selectedCompanyName);
        Vector buyerVector = getBuySellList(shareDetailsVector,
          selectedCategoryName, selectedCompanyName, "Buy");
        Vector sellerVector = getBuySellList(shareDetailsVector,
          selectedCategoryName, selectedCompanyName, "Sell");
```

```
                buyersData = new Object[buyerVector.size()][3];
                orderNumberArray = new String[buyerVector.size()];
                for (int i = 0; i < buyerVector.size(); i++) {
                    sellItem = buyerVector.elementAt(i).toString();
                    toks = new StringTokenizer(sellItem, ",");
                    orderNumber = toks.nextToken();
                    lotSize = toks.nextToken();
                    numShares = toks.nextToken();
                    price = toks.nextToken();
                    buyersData[i][0] = new String(lotSize);
                    buyersData[i][1] = new String(numShares);
                    buyersData[i][2] = new String(price);
                    orderNumberArray[i] = new String(orderNumber);
                    buyersTableView.updateUI();
                }
                sellersData = new Object[sellerVector.size()][3];
                orderNumberArray = new String[sellerVector.size()];
                for (int i = 0; i < sellerVector.size(); i++) {
                    sellItem = sellerVector.elementAt(i).toString();
                    toks = new StringTokenizer(sellItem, ",");
                    orderNumber = toks.nextToken();
                    lotSize = toks.nextToken();
                    numShares = toks.nextToken();
                    price = toks.nextToken();
                    sellersData[i][0] = new String(lotSize);
                    sellersData[i][1] = new String(numShares);
                    sellersData[i][2] = new String(price);
                    orderNumberArray[i] = new String(orderNumber);
                    sellersTableView.updateUI();
                }
            }
        }
    }
}

/*
*Βρίσκει όλες τις εντολές (προς πώληση ή αγορά) που έχουν όλοι οι χρηματιστές στη διάθεση τους
*/
    void showQueryPanel() {
     try {
        ESArray detailsVector = ShareBrokerIntfFinder.find(stockTrade.brokerName,
            stockTrade.connection).listOrderDetails();

        if (shareDetailsVector.size() != 0) {
            shareDetailsVector.removeAllElements();
        }
        OrderDetail order;
        String str;

        for (int i = 0; i < detailsVector.size(); i++) {
            order = (OrderDetail)detailsVector.elementAt(i);
            str = new String(order.categoryName + "," + order.company
                + "," + order.orderNumber + "," + order.lotSize + ","
                + order.no + "," + order.price + ",");
            if (order.option == 1) {
                str = str + "Buy";
            } else {
                str = str + "Sell";
            }
            shareDetailsVector.addElement(str);
```

```
            System.out.println(str);
        }
        Vector categoryVector = getCategories(shareDetailsVector);

        if (categoryModel.size() != 0) {
            categoryModel.clear();
        }
        for (int i = 0; i < categoryVector.size(); i++) {
            categoryModel.addElement(categoryVector.elementAt(i).toString());
        }
        if (companyModel.size() != 0) {
            companyModel.clear();
        }
        sellersListLabel.setText("Αγοραστές");
        buyersListLabel.setText("Πωλητές");
        companyListLabel.setText("Λίστα Μετοχών");

        buyersData = null;
        buyersTableView.updateUI();
        sellersData = null;
        sellersTableView.updateUI();
        setVisible(true);
    } catch (Exception e) {
        e.printStackTrace();
    }
}


/**
 * Εξαφανίζει την σχετική οθόνη
 */
void hideQueryPanel() {
    setVisible(false);
}


/**
 *Επιστρέφει την λίστα με τις κατηγορίες των μετοχών που περιέχονται στις εντολές προς
 *διαπραγμάτευση όλων των χρηματιστών
 */
Vector getCategories(Vector shareVector) {
    Vector categories = new Vector(0, 1);
    StringTokenizer toks;
    String token;

    for (int i = 0; i < shareVector.size(); i++) {
        toks = new StringTokenizer(shareVector.elementAt(i).toString(), ",");
        token = toks.nextToken();
        if (categories.indexOf(token) == -1) {
            categories.addElement(token);
        }
    }
    return categories;
}


/**
 * Επιστρέφει την λίστα των εταιρειών ανά κατηγορία μετοχών
 */
Vector getCompanyByCategory(String categoryName, Vector shareVector) {
```

```
    Vector companies = new Vector(0, 1);
    StringTokenizer toks;
    String token;

    for (int i = 0; i < shareVector.size(); i++) {
      toks = new StringTokenizer(shareVector.elementAt(i).toString(),
        ",");
      token = toks.nextToken();
      if (token.equals(categoryName)) {
        token = toks.nextToken();
        if (companies.indexOf(token) == -1) {
          companies.addElement(token);
        }
      }
    }
    return companies;
  }

  /**
   *Επιστρέφει τις μετοχές οι οποίες είναι προς αγορά ή προς πώληση
   */
  Vector getBuySellList(Vector shareVector, String catName, String coName,
    String type) {
    Vector output = new Vector(0, 1);
    StringTokenizer toks;
    String token;
    String orderNumber, lotSize, numShares, price, buySell;

    for (int i = 0; i < shareVector.size(); i++) {
      toks = new StringTokenizer(shareVector.elementAt(i).toString(),",");
      token = toks.nextToken();
      if (token.equals(catName)) {
        token = toks.nextToken();
        if (token.equals(coName)) {
          orderNumber = toks.nextToken();
          lotSize = toks.nextToken();
          numShares = toks.nextToken();
          price = toks.nextToken();
          buySell = toks.nextToken();
          if (buySell.equals(type)) {
            output.addElement(orderNumber + "," + lotSize + ","
              + numShares + "," + price + "," + type);
          }
        }
      }
    }
    return output;
  }

  /**
   * Εμφανίζει το μήνυμα που έχει σαν παράμετρο
   *
   */
  void ShowMessageDialog(String message) {
    (new MessageDialog(stockTrade, true, message)).show();
  }
}
```

20. AuthorizationWindow.java

```
package sharebroker;
import java.awt.*;
import java.lang.*;
import javax.swing.*;


/**
 * <dl>
 * <dt>Purpose:
 * <dd>
 * Η κλάση AuthorizationWindow χρησιμοποιείται για την δημιουργία της οθόνης που χειρίζεται την
 * διαπραγμάτευση της αγοράς ή της πώλησης των μετοχών ενός πελάτη
 *
 */
public class AuthorizationWindow extends JFrame {
   JLabel optionLabel, yourOrderLabel, matchingOrderLabel, companyNameLabel,
        numSharesLabel, orderNoLabel, valueLabel, buySellLabel;
   JTextField yourCompanyTextField, matchingCompanyTextField,
      yourOrderNoTextField, matchingOrderNoTextField;
   JTextField yourNumSharesTextField, matchingNumSharesTextField,
      yourValueTextField, matchingValueTextField;
   JTextField yourBuySellTextField, matchingBuySellTextField;
   JComboBox optionChoice;
   JButton okayButton;
   ShareBrokerIntf sbi;
   OrderDetailParam yourOrder, matchingOrder;


 /**
    * Ο δημιουργός  παίρνει σαν παραμέτρους  το αντικείμενο ShareBrokerIntf, τις λεπτομέρειες της
    * εντολής του πελάτη, όπως επίσης και τις λεπτομέρειες της εντολής με την οποία ταιριάζει η
    * εντολή του πελάτη. Η οθόνη αυτή βοηθάει στην διαπραγμάτευση/συμφωνία/ακύρωση μιας
    * διαπραγμάτευσης και στον χειρισμό των κατάλληλων χειριστηρίων.
*/
   public AuthorizationWindow(ShareBrokerIntf temp,
      OrderDetailParam yourOrder, OrderDetailParam matchingOrder) {

      this.yourOrder = yourOrder;
      this.matchingOrder = matchingOrder;
      sbi = temp;
      getContentPane().setLayout(null);
      setSize(670, 250);
      setTitle("Διαπραγμάτευση");
      yourOrderLabel = new JLabel("Προσφορά");
      yourOrderLabel.setBounds(10, 50, 100, 20);
      getContentPane().add(yourOrderLabel);
      matchingOrderLabel = new JLabel("Αντιπροσφορά");
      matchingOrderLabel.setBounds(10, 80, 100, 20);
      getContentPane().add(matchingOrderLabel);
      companyNameLabel = new JLabel("Εταιρεία");
      companyNameLabel.setBounds(120, 20, 100, 20);
      getContentPane().add(companyNameLabel);
      yourCompanyTextField = new JTextField(yourOrder.company);
      yourCompanyTextField.setBounds(120, 50, 100, 20);
      getContentPane().add(yourCompanyTextField);
      matchingCompanyTextField = new JTextField(matchingOrder.company);
      matchingCompanyTextField.setBounds(120, 80, 100, 20);
```

```
getContentPane().add(matchingCompanyTextField);
orderNoLabel = new JLabel("Αρ.Εντολής");
orderNoLabel.setBounds(230, 20, 100, 20);
getContentPane().add(orderNoLabel);
yourOrderNoTextField = new JTextField(yourOrder.orderNumber);
yourOrderNoTextField.setBounds(230, 50, 100, 20);
getContentPane().add(yourOrderNoTextField);
matchingOrderNoTextField = new JTextField(matchingOrder.orderNumber);
matchingOrderNoTextField.setBounds(230, 80, 100, 20);
getContentPane().add(matchingOrderNoTextField);
numSharesLabel = new JLabel("Μερίδια");
numSharesLabel.setBounds(340, 20, 100, 20);
getContentPane().add(numSharesLabel);
yourNumSharesTextField = new JTextField();
yourNumSharesTextField.setText(Integer.toString(yourOrder.no));
yourNumSharesTextField.setBounds(340, 50, 100, 20);
getContentPane().add(yourNumSharesTextField);
matchingNumSharesTextField = new JTextField();
matchingNumSharesTextField.setText(Integer.toString(matchingOrder.no));
matchingNumSharesTextField.setBounds(340, 80, 100, 20);
getContentPane().add(matchingNumSharesTextField);
valueLabel = new JLabel("Αξία");
valueLabel.setBounds(450, 20, 100, 20);
getContentPane().add(valueLabel);
yourValueTextField = new JTextField();
yourValueTextField.setText(Float.toString(yourOrder.price));
yourValueTextField.setBounds(450, 50, 100, 20);
getContentPane().add(yourValueTextField);
matchingValueTextField = new JTextField();
matchingValueTextField.setText(Float.toString(matchingOrder.price));
matchingValueTextField.setBounds(450, 80, 100, 20);
getContentPane().add(matchingValueTextField);
buySellLabel = new JLabel("Αγορά/Πώληση");
buySellLabel.setBounds(560, 20, 100, 20);
getContentPane().add(buySellLabel);
String buysell = new String();

if (yourOrder.option == 1) {
   buysell = "Αγορά";
} else {
   buysell = "Πώληση";
}
yourBuySellTextField = new JTextField(buysell);
yourBuySellTextField.setBounds(560, 50, 100, 20);
getContentPane().add(yourBuySellTextField);
if (matchingOrder.option == 1) {
   buysell = "Αγορά";
} else {
   buysell = "Πώληση";
}
matchingBuySellTextField = new JTextField(buysell);
matchingBuySellTextField.setBounds(560, 80, 100, 20);
getContentPane().add(matchingBuySellTextField);
optionLabel = new JLabel("Επέλεξε");
optionLabel.setBounds(10, 110, 100, 20);
getContentPane().add(optionLabel);
String[] optionItems = {
   "Συμφωνία", "Διαπραγμάτευση", "Άρνηση"
};
```

```
        optionChoice = new JComboBox(optionItems);
        optionChoice.setBounds(120, 110, 100, 20);
        getContentPane().add(optionChoice);
        okayButton = new JButton("Επιβεβαίωση");
        okayButton.setBounds(120, 140, 110, 20);
        getContentPane().add(okayButton);
        setResizable(false);

        /**
     * Ενεργοποιούνται οι listeners που χειρίζονται το παράθυρο και το κουμπί επικύρωσης της
     * διαπραγμάτευσης*/
        SymWindow aSymWindow = new SymWindow();
        this.addWindowListener(aSymWindow);
        SymAction lSymAction = new SymAction();
        okayButton.addActionListener(lSymAction);
}


    /**
     * Η υλοποίηση του listener παραθύρου
     */
    class SymWindow extends java.awt.event.WindowAdapter {
    public void windowClosing(java.awt.event.WindowEvent event) {
          Object object = event.getSource();

          if (object == AuthorizationWindow.this) {
            Authorization_WindowClosing(event);
          }
        }
    }


void Authorization_WindowClosing(java.awt.event.WindowEvent event) {
        setVisible(false);
    }


    /**
     *Υλοποίηση του Listener που χειρίζεται το κουμπί επικύρωσης της συναλλαγής
     */
    class SymAction implements java.awt.event.ActionListener {
         public void actionPerformed(java.awt.event.ActionEvent event) {
          Object object = event.getSource();

          if (object == okayButton) {
            OkayButton_Clicked(event);
          }
        }
    }


void OkayButton_Clicked(java.awt.event.ActionEvent event) {
        OrderDetailParam modYourOrder, modMatchingOrder;

        modYourOrder = (OrderDetailParam)(new OrderDetail(yourOrder));
        modMatchingOrder = (OrderDetailParam)(new OrderDetail(matchingOrder));
        modYourOrder.companyReference =
          new String(yourOrder.companyReference);
        modMatchingOrder.companyReference =
          new String(matchingOrder.companyReference);
```

```
        modYourOrder.company = yourCompanyTextField.getText();
        modMatchingOrder.company = matchingCompanyTextField.getText();
        modYourOrder.orderNumber = yourOrderNoTextField.getText();
        modMatchingOrder.orderNumber = matchingOrderNoTextField.getText();
        modYourOrder.no = Integer.parseInt(yourNumSharesTextField.getText());
        modMatchingOrder.no =
          Integer.parseInt(matchingNumSharesTextField.getText());
        modYourOrder.price = new Float(yourValueTextField.getText()).floatValue();
        modMatchingOrder.price =
          new Float(matchingValueTextField.getText()).floatValue();
        if (yourBuySellTextField.getText().equals("Αγορά")) {
          modYourOrder.option = 1;
        } else {
          modYourOrder.option = -1;
        }
        if (matchingBuySellTextField.getText().equals("Αγορά")) {
          modMatchingOrder.option = 1;
        } else {
          modMatchingOrder.option = -1;
        }
        int option = optionChoice.getSelectedIndex();
        String selected = null;

        if (option == 0) {
          selected = new String("2");
        } else if (option == 1) {
          selected = new String("1");
        } else {
          selected = new String("0");
        }

        /*Ενεργοποίηση της μεθόδου στο BrokerLogic για το κλείσιμο της διαπραγμάτευσης*/
        try {
          sbi.authorizeResp(modYourOrder, modMatchingOrder, selected);
        } catch (Exception e) {
          e.printStackTrace();
        }
        setVisible(false);
      }
    }
```

21. AccountNumberGenerator.java

```
package sharebroker;
import java.io.*;


/**
 * Σκοπός της AccountNumberGenerator είναι η δημιουργία νέων αριθμών λογαριασμού για τους νέους
 *χρήστες της εφαρμογής. Η λειτουργία του στηρίζεται στη ανάγνωση ενός αριθμού από ένα
 *προκαθορισμένο αρχείο και την αύξηση του κατά ένα για κάθε περίπτωση
*/
public class AccountNumberGenerator {
  int currentNumber;
  String line;
  String fileName;


  /**
   * Ο δημιουργός διαβάζει τον τρέχοντα αριθμό από ένα δοθέν αρχείο, τον αυξάνει και τον
```

```
     * επανατοποθετεί στο αρχείο ως τρέχοντα πλέον αριθμό
     */
    AccountNumberGenerator(String fileName) {
      try {
        DataInputStream fin = new DataInputStream(new FileInputStream(fileName));
        line = fin.readLine();
        currentNumber = Integer.parseInt(line);
        this.fileName = fileName;
      } catch (IOException e) {
        System.println.out(BankService.getBankName() + ":Το αρχείο δεν βρέθηκε");
      }
    }

    /**
     * Αυξάνει τον τρέχοντα αριθμό
     */
    public String nextNumber() {
      currentNumber++;
      PrintStream fout = null;

      try {
        fout = new PrintStream(new FileOutputStream(fileName));
      } catch (Exception e) {
        e.printStackTrace();
      }
      line = new Integer(currentNumber).toString();
      fout.println(line);
      return line;
    }
}
```

22. BrokerLogic.java

```
package sharebroker;
import java.util.*;
import java.lang.Math.*;
import net.espeak.infra.client.impl.*;
import net.espeak.infra.client.util.*;
import net.espeak.util.*;
import net.espeak.infra.cci.events.*;
import net.espeak.jesi.event.*;
import java.awt.Dialog;
import java.awt.Frame;


/**
* Η κλάση BrokerLogic είναι η βασικότερη της εφαρμογής. Διαχειρίζεται όλη την φιλοσοφία της
εύρεσης *εντολών που ταυτίζονται μεταξύ τους, πληροφορεί του συναλλασόμενους, υλοποιεί την
διαπραγμάτευση *και το κλείσιμο της συμφωνίας. Επίσης τηρεί όλες της πληροφορίες για τις εντολές
πώληση ή αγοράς *όλων των πελατών.
*/
public class BrokerLogic implements ESListenerIntf {
   ESArray orderTable;                    // Πίνακας με τις λεπτομέρειες της συναλλαγής
   boolean CancelOrder = false;
   Hashtable reservedShares;              // Εντολές υπό διαπραγμάτευση
   Hashtable matchCountMap;               // Αριθμός διαπραγματεύσεων σε εξέλιξη
   Hashtable callersHash;                 // Εντολές ενός πελάτη
   Hashtable accountNumberHash;           // maps order num to account num of trader
   int latest;                            // Τελευταία εντολή που προέκυψε
```

```
Semaphore sem;                    // Σηματοφορέας για τον συγχρονισμό των threads
ESPublisher publ;                 // Με το  publ μπορεί να γνωστοποιήσει αιτήματα(εντολές)
String brokerName;
String from = "broker";
String traderName = new String();
static int DENIED = 0;
static int NEGOTIATE = 1;
static int AGREED = 2;
static float PRICE_DIFF_PCNT = (float)0.15;   // Ποσοστό διαφοροποίησης μεταξύ των εντολών


/**
 * Κλάση για την τήρηση του αριθμού των διαπραγματεύσεων που γίνονται
 */
class MatchCount {
  public MatchCount(int value) {
     count = value;
   }
   int count;
}


/**
 * Ο δημιουργός αρχικοποιεί τις μεταβλητές που χρησιμοποιούνται
 */
BrokerLogic(ESPublisher publ, String brokerName) {
   orderTable = new ESArray();
   reservedShares = new Hashtable();
   matchCountMap = new Hashtable();
   callersHash = new Hashtable();
   accountNumberHash = new Hashtable();
   latest = 100;
   sem = new Semaphore();
   this.publ = publ;
   this.brokerName = brokerName;
   CancelOrder = false;
}


/**
 * Δημιουργεί τον αριθμό της τρέχουσας εντολής, η οποία μπορεί να χρησιμοποιηθεί σαν σημείο
 * αναφοράς από τον χρήστη στη φάση της διαπραγμάτευσης.
 */
public synchronized void generateOrderNumber(TraderQuerySpec query,
   OrderDetail order) {
   latest++;
   order.orderNumber = (brokerName + "#"+ new Integer(latest)).toString().trim();
   accountNumberHash.put(new String(order.orderNumber),
   new String(order.accountNumber));
   callersHash.put(order.orderNumber, TraderIntfFinder.find(query, null));
   return;
}


/**
 * Η μέθοδος αυτή αποθηκεύει τις εντολές αγοράς των πελατών στον σχετικό πίνακα εντολών προς
 * διαπραγμάτευση, ενώ επίσης αρχικοποιεί ένα Thread  που ελέγχει συνεχώς για την ύπαρξη εντολής
 * ταύτισης .
 */
public String buyShares(TraderQuerySpec query, OrderDetail order) {
```

```
        if (query != null) {
            generateOrderNumber(query, order);
            order.option = 1;   // Αγορά
            traderName = query.traderName;
        }
        orderTable.add(order);
        MatchThread mathThread = new MatchThread(this, order);
        mathThread.start();
        return (order.orderNumber);
    }

/**
     * Η μέθοδος αυτή αποθηκεύει τις εντολές πώλησης των πελατών στον σχετικό πίνακα εντολών προς
     * διαπραγμάτευση, ενώ επίσης αρχικοποιεί ένα Thread που ελέγχει συνεχώς για την ύπαρξη εντολής
     * ταύτισης .
     */
    public String sellShares(TraderQuerySpec query, OrderDetail order) {
        if (query != null) {
            System.out.println("Query is not null");
            generateOrderNumber(query, order);
            order.option = -1;   // Πώληση
            traderName = query.traderName;
        }
        orderTable.add(order);
        MatchThread matchThread = new MatchThread(this, order);
        matchThread.start();
        return (order.orderNumber);
    }


    /**
     * Η μέθοδος υπολογίζει το πλήθος των μετοχών που μπορούν να μετακινηθούν από τον ένα
λογαριασμό
     *σε κάποιο άλλο προτού ξεκινήσει η διαπραγμάτευση. Ο αριθμός των μετοχών που προκύπτει
     *δεσμεύεται. Η μέθοδος matchCount ενημερώνεται. Μετά ξεκινά η διαπραγμάτευση.
     */
    public void adjustAndStartDeal(OrderDetail order, OrderDetail matchWith) {
        int min;   // αριθμός μετοχών που μπορούν να μετακινηθούν
        if (matchWith.no < order.no) {
            min = matchWith.no;
        } else {
            min = order.no;
        }
        OrderDetail lorder = lock(order, matchWith, min);
        OrderDetail lmatchWith = lock(matchWith, order, min);
        incrementMatchCount(lorder);
        incrementMatchCount(lmatchWith);
        startDeal(lorder, lmatchWith);
    }


    /**
     *Η μέθοδος αυτή προσπαθεί να ταυτίσει μια νέα εντολή που ανακύπτει με μια ήδη υπάρχουσα στον
     *χρηματιστή. Με την ύπαρξη της ταύτισης ελέγχεται αν μπορεί να ξεκινήσει μια διαπραγμάτευση.
     * Εάν η προσφορά είναι πολύ μεγαλύτερη σε ποσοστό από αυτή που προσδιορίζεται στη μεταβλητή
     * PRICE_DIFF_PCNT , δεν υπάρχει σημείο επαφής και έτσι η διαπραγμάτευση ματαιώνεται. Εάν
     * δεν βρεθεί κατάλληλη εντολή εγείρεται ένα αίτημα αγοράς ή πώλησης αναλόγως της εντολής.
     */
    public synchronized void match(OrderDetail order) {
      sem.down();
```

```
        int count = 0;
        OrderDetail matchWith;
        int size = orderTable.size();
        int origNo = order.no;

        for (count = 0; count < size; count++) {
          matchWith = (OrderDetail)(orderTable.elementAt(count));
          if (matchWith != null && matchWith.no != 0) {
            // Ελέγχονται οι εντολές που ανήκουν στην ίδια κατηγορία
            if (matchWith.categoryName.equals(order.categoryName)) {
              // Ελέγχονται οι μετοχές βάσει του ονόματος τους
              if (matchWith.company.equals(order.company)) {
                //  Ελέγχεται αν είναι αγοραστής και πωλητής
                if ((matchWith.option + order.option)== 0) {
                  // Ελέγχεται αν η προσφορά βρίσκεται στα προκαθορισμένα όρια
                  if (java.lang.Math.abs(order.price - matchWith.price)
                    <= PRICE_DIFF_PCNT * order.price) {
                    // Δεσμεύει τον αριθμό των μετοχών
                    adjustAndStartDeal(order, matchWith);
                  }
                }
              }
            }
          }
          sem.up();
          if (order.no == origNo &&!remoteOrder(order)) {
            this.raiseEvent(order);
          }
        }


        /**
         * Ελέγχει αν μια εντολή βρίσκεται στον τρέχοντα χρηματιστή ή βρίσκεται σε κάποιο άλλο.
Επιστρέφει
         * TRUE αν η εντολή βρίσκεται σε διαφορετικό χρηματιστή ειδάλλως επιστρέφει FALSE.
         */
        public boolean remoteOrder(OrderDetail order) {
          return !(order.orderNumber.substring(0,
            order.orderNumber.indexOf('#')).equals(brokerName));
        }


        /**
         * Επιστρέφει τον χρηματιστή που έχει στη κατοχή του μια συγκεκριμένη μετοχή.
         */
        public ShareBrokerIntf findRemoteBroker(OrderDetail order) {
return ShareBrokerIntfFinder.find(order.orderNumber.substring(0,
order.orderNumber.indexOf('#')),null);
        }
        /**
         * This remote method is invoked by a remote broker as a request
         * for authorization from the this broker. This broker will inturn
         * contact the a local trader. If matchWith is not in order table add it
         * to the table.
         * @param  order   order corresponding to the trader.
         * @param  matchWith the other party's order.
         */
        public synchronized void authorizeReq(OrderDetail order,
          OrderDetail matchWith) {
          try {
```

```
            if (!search(matchWith)) {
               add(matchWith);
            }
            TraderIntf abc = (TraderIntf)(callersHash.get(order.orderNumber));

            abc.authorizeReq(order, matchWith);
         } catch (Exception e) {
            e.printStackTrace();
         }
      }


      /**
       * Η μέθοδος αυτή ενεργοποιείται από τον πελάτη σαν απάντηση στο αίτημα του χρηματιστή για
       * έγκριση. Ανάλογα με την απάντηση  η διαπραγμάτευση συνεχίζεται ή διακόπτεται. Επίσης ανάλογα
       * με την περίπτωση πρέπει να αποδεσμευθούν και οι μετοχές που είχαν δεσμευθεί πρίν τη φάση της
       *διαπραγμάτευσης
       *
       */
      public synchronized void authorizeResp(OrderDetail order,
         OrderDetail matchWith, String opt) {
         int option = Integer.parseInt((new String(opt)).trim());
         boolean remote = remoteOrder(order);
if (remote) {
            try {
               if (!search(order)) {
                  add(order);
                  int min;
                  if (matchWith.no < order.no) {
                     min = matchWith.no;
                  } else {
                     min = order.no;
                  }
                  OrderDetail savedOrder, savedMatchWith;
                  savedOrder = findSavedOrder(order);
                  savedMatchWith = findSavedOrder(matchWith);
                  OrderDetail lorder = lock(savedOrder, savedMatchWith, min);
                  OrderDetail lmatchWith = lock(savedMatchWith, savedOrder, min);
                  incrementMatchCount(lorder);
                  incrementMatchCount(lmatchWith);
                  order = lorder;
                  matchWith = lmatchWith;
               } else {
                  System.println.output(brokerName + ": " + order.orderNumber + " allready exists");
               }
            } catch (Exception e) {
               e.printStackTrace();
               return;
            }
         }
         if (option == DENIED)    // Εάν υπάρχει άρνηση διαπραγμέτευσης
          {
            unlock(order, matchWith, false);
            unlock(matchWith, order, false);
            invokeNotifyDeal(order, matchWith, false);
            return;
         }
         if (order.option == 1) {
            ReqAuthThread reqForAuth = new ReqAuthThread(this, matchWith,order.nclone(), opt);
            reqForAuth.start();
```

```
        return;
    }
    if (option == NEGOTIATE) {
        ReqAuthThread reqForAuth = new ReqAuthThread(this, matchWith, order.nclone(), opt);
        reqForAuth.start();
        return;
    }
    closeTheDeal(order, matchWith);
    unlock(order, matchWith, true);
    unlock(matchWith, order, true);
    invokeNotifyDeal(order, matchWith, true);
}

/**
 * Η μέθοδος αυτή ενημερώνει την τράπεζα λα κλείσει τη συμφωνία μεταξύ δύο λογαριασμών.
 */
public void closeTheDeal(OrderDetail order, OrderDetail matchWith) {
try {
        BankServiceIntfFinder.find(null).closeTransaction(order,
            matchWith);
    } catch (Exception e) {
    }
}

/**
 *Η μέθοδος αυτή αποδεσμεύει της μετοχές που είχαν δεσμευθεί πρίν τη διαπραγμάτευση. Εάν η
 εντολή
 *δεν κατέληξε επιτυχώς επανέρχεται στην αρχική κατάσταση, διαφορετικά διαγράφεται από τον
 * χρηματιστή.
 */
public void unlock(OrderDetail order, OrderDetail matchWith,boolean dealDone) {
    sem.down();
    reservedShares.remove(genhash(order, matchWith));
    MatchCount temp = decrementMatchCount(order);
    if (dealDone == false) {
        add(order);
    } else if (temp.count== 0) {
        tryPurgeOrder(order);
    }
    sem.up();
}

/**
  *Ανακαλύπτει μια συγκεκριμένη μετοχή στον πίνακα των μετοχών
 */
public OrderDetail findSavedOrder(OrderDetail order) throws Exception {
    int count = 0;
    OrderDetail temp = null;
    int size = orderTable.size();

    for (count = 0; count < size; count++) {
        temp = (OrderDetail)(orderTable.elementAt(count));
        if (order.orderNumber.equals(temp.orderNumber)) {
```

```
            return temp;
        }
    }
    throw (new Exception("Order not found"));
}


/**
 * Επιστρέφει TRUE ή FALSE ανάλογα με το αν βρέθηκε ή όχι η μετοχή στον πίνακα
 *
 * @param order
 * @return
 */
public boolean search(OrderDetail order) {
    int count = 0;
    OrderDetail temp = null;
    int size = orderTable.size();

    for (count = 0; count < size; count++) {
        temp = (OrderDetail)(orderTable.elementAt(count));
        if (order.orderNumber.equals(temp.orderNumber)) {
            return true;
        }
    }
    return false;
}


/**
 * Προσθέτει ή ενημερώνει μια εντολή στον πίνακα των εντολών. Εάν η εντολή υπάρχει ενημερώνεται
 *αλλάζοντας τον αριθμό των μετοχών, ειδάλλως προστίθεται στον πίνακα.
 */
public void add(OrderDetail order) {
    int count = 0;
    int flag = 0;
    OrderDetail temp = null;
    int size = orderTable.size();

    for (count = 0; count < size; count++) {
        temp = (OrderDetail)(orderTable.elementAt(count));
        if (order.orderNumber.equals(temp.orderNumber)) {
            temp.no += order.no;
            flag = 1;
            break;
        }
    }
    if (flag == 0) {
        orderTable.addElement(order);
    }
}


/**
 * Ξεκινάει την διαδικασία συμφωνίας ζητώντας την έγκριση του αγοραστή
 */
public void startDeal(OrderDetail order, OrderDetail matchWith) {

 if (order.option == 1)    // Εντολή αγοράς
    {
      ReqAuthThread reqForAuth = new ReqAuthThread(this, order,matchWith, "1");
```

```
        reqForAuth.start();
    } else {
        ReqAuthThread reqForAuth = new ReqAuthThread(this, matchWith,order, "1");
        reqForAuth.start();
    }
}


/**
 * Η μέθοδος αυτή δεσμεύει τις μετοχές που περιέχονται σε μια εντολή έως ότου λήξει η
 *διαπραγμάτευση
 */
public OrderDetail lock(OrderDetail order, OrderDetail matchWith,
    int number) {
    order.no = order.no - number;
    OrderDetail newOrder = (OrderDetail)order.nclone();

    newOrder.no = number;
    reservedShares.put(genhash(order, matchWith), newOrder);
    return (newOrder);
}


/**
 * Επιστρέφει την λίστα των διαθέσιμων μετοχών
 */
public ESArray listOrderDetails() {
   return (orderTable);
}


public void cancelOrder(String orderNo) {
    CancelOrder = true;
    raiseCancelOrderEvent(orderNo);
}


/**
 * Χρησιμοποιείται για την ακύρωση των εντολών
 */
public void CanceltheOrder(String Orderno) {
    OrderDetail temp;
    int size = orderTable.size();
    for (int count = 0; count < size; count++) {
        temp = (OrderDetail)(orderTable.elementAt(count));
        if (temp != null && temp.orderNumber.equals(Orderno)) {
            accountNumberHash.remove(temp.orderNumber);
            orderTable.removeElementAt(count);
            break;
        }
    }
    Enumeration e = reservedShares.elements();
    Enumeration ekeys = reservedShares.keys();
    for (; e.hasMoreElements(); ) {
        temp = (OrderDetail)e.nextElement();
        if (temp.orderNumber.equals(Orderno)) {
            reservedShares.remove(ekeys.nextElement());
            break;
        }
        ekeys.nextElement();
```

```
    }
    matchCountMap.remove(Orderno);
    callersHash.remove(Orderno);
  }

  /**
   * Διαγράφει μια εντολή  από τον πίνακα εντολών. */
  private void tryPurgeOrder(OrderDetail order) {
    OrderDetail temp;
    int size = orderTable.size();
    for (int count = 0; count < size; count++) {
      temp = (OrderDetail)(orderTable.elementAt(count));
      if (temp != null && temp.orderNumber.equals(order.orderNumber)
        && temp.no == 0) {
        accountNumberHash.remove(temp.orderNumber);
        orderTable.removeElementAt(count);
        break;
      }
    }
  }
}
```