ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΕΡΕΥΝΑΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

ΕΠΕΤ ΙΙ

ΕΡΓΟ: 98ΑΜΕΑ 19

ΑΙΝΕΙΑΣ:ΑΝΑΠΤΥΞΗ ΕΥΕΛΙΚΤΩΝ ΣΥΣΤΗΜΑΤΩΝ ΕΝΑΛΛΑΚΤΙΚΗΣ ΚΑΙ ΕΠΑΥΞΗΤΙΚΗΣ ΕΠΙΚΟΙΝΩΝΙΑΣ ΜΕΣΩ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΤΟΥ ΔΙΑΔΙΚΤΥΟΥ

Παραδοτέο Π3.1

Ανάλυση και Τεχνικές Προδιαγραφές Αντικειμενοστραφούς Πλαισίου Ανάπτυξης Εφαρμογών «ΟΔΥΣΣΕΑΣ»

Αλέξανδρου Πίνο και Γεωργίου Κουρουπέτρογλου

ΑΘΗΝΑ

15 ΣΕΠΤΕΜΒΡΙΟΥ 2000

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΕΡΕΥΝΑΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ
ΕΠΕΤ ΙΙ

ΕΡΓΟ: 98ΑΜΕΑ 19

ΑΙΝΕΙΑΣ: ΑΝΑΠΤΥΞΗ ΕΥΕΛΙΚΤΩΝ ΣΥΣΤΗΜΑΤΩΝ ΕΝΑΛΛΑΚΤΙΚΗΣ ΚΑΙ ΕΠΑΥΞΗΤΙΚΗΣ ΕΠΙΚΟΙΝΩΝΙΑΣ ΜΕΣΩ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΤΟΥ ΔΙΑΔΙΚΤΥΟΥ

Ανάδοχος φορέας: Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Συνεργαζόμενοι φορείς: 01 ΠΛΗΡΟΦΟΡΙΚΗ

Κέντρο Αποκατάστασης Ατόμων με Ειδικές Ανάγκες (Κ.Α.Α.Μ.Ε.Α.) Σερρών

Επιστημονικός Υπεύθυνος: Γεώργιος Κουρουπέτρογλου
Πανεπιστήμιο Αθηνών, Τμήμα Πληροφορικής,
Πανεπιστημιούπολη, Ιλίσια,
Αθήνα 15784

Τηλέφωνο: (01) 7275305
Fax: (01) 6018677

Ηλεκτρονικό Ταχυδρομείο: koupe@di.uoa.gr

Περιεχόμενα

1. Εισαγωγή

Το πλαίσιο Σχεδιασμού και Ανάπτυξης Βοηθημάτων Διαπροσωπικής Επικοινωνίας του έργου ΑΙΝΕΙΑΣ [1], που εν συντομία ονομάζεται «ΟΔΥΣΣΕΑΣ» και θα περιγραφεί εδώ, έχει ως στόχο την μελέτη και ανάπτυξη μιας νέας σταθερής πλατφόρμας για την γρήγορη και με μικρό κόστος υλοποίηση από διάφορους κατασκευαστές ευέλικτων και βελτιωμένων επικοινωνιακών βοηθημάτων που βασίζονται σε ηλεκτρονικό υπολογιστή. Αυτό θα επιτευχθεί με την υιοθέτηση συγκεκριμένων τυποποιήσεων, τη θέσπιση αυστηρών προδιαγραφών και την ανάπτυξη μιας αρθρωτής αρχιτεκτονικής και των σχετικών βοηθητικών εργαλείων ανάπτυξης λογισμικού. Όλα όσα αναφέρονται στην σύντομη παρουσίαση του πλαισίου ΟΔΥΣΣΕΑΣ στην εισαγωγή αυτή, αναλύονται με λεπτομέρεια στα επιμέρους κεφάλαια της παρούσας Τεχνικής Έκθεσης.

Τα Συστήματα Εναλλακτικής και Επαυξητικής Επικοινωνίας ή Βοηθήματα Διαπροσωπικής Επικοινωνίας απευθύνονται σε χρήστες Άτομα Με Ειδικές Ανάγκες (ΑΜΕΑ) [2]. Τα Βοηθήματα Διαπροσωπικής Επικοινωνίας που μας απασχολούν σε αυτό το έργο βασίζονται σε ηλεκτρονικούς υπολογιστές και είναι συχνά πολύπλοκα συστήματα που πολλές φορές έχουν σαν έξοδο φωνή και επικεντρώνονται στις ανάγκες κάποιου συγκεκριμένου χρήστη. Μετά από μια αρχική εκτίμηση των επικοινωνιακών αναγκών του χρήστη και λαμβάνοντας υπ' όψη την κατάσταση των νευρολογικών του λειτουργιών, τις δυνατότητες κινήσεών του καθώς και τις γλωσσικές του ικανότητες, καταρτίζεται ένα πρόγραμμα επανένταξης το οποίο συχνά περιλαμβάνει τη χρήση εξοπλισμού. Ο εξοπλισμός αυτός δεν είναι ευρέως διαδεδομένος στη χώρα μας και τα συστήματα της αγοράς του εξωτερικού παρουσιάζουν δυο βασικά μειονεκτήματα: έχουν μεγάλο κόστος και δεν υποστηρίζουν την Ελληνική γλώσσα. Τα βοηθήματα αυτά πρέπει να παρουσιάζουν μεγάλη ευελιξία και προσαρμοστικότητα γιατί συνήθως πριν ολοκληρωθεί το πρόγραμμα επανένταξης μεταβάλλουν τα χαρακτηριστικά τους για να ικανοποιήσουν τις αυξημένες επικοινωνιακές ανάγκες του χρήστη. Για εκείνους που δεν μπορούν αν χρησιμοποιήσουν το πληκτρολόγιο υπάρχουν εναλλακτικές μέθοδοι όπως η χρήση του ποδιού, του κεφαλιού ή ακόμα και της κίνησης των βλεφάρων μέσω ειδικών συσκευών και διακοπτών. Οι εναλλακτικοί τρόποι πρόσβασης στα Βοηθήματα Διαπροσωπικής Επικοινωνίας παρουσιάζονται σε χωριστή Τεχνική Έκθεση [61]. Το επικοινωνιακό βοήθημα θα πρέπει να παρέχει άμεση πρόσβαση σε προτάσεις ζωτικές για την ικανοποίηση καθημερινών αναγκών καθώς και σε προτάσεις χρήσιμες στην καθημερινή επικοινωνία. Ένας από τους σημαντικούς παράγοντες σε ένα σύστημα επικοινωνίας είναι η ταχύτητα επικοινωνίας που μπορεί να επιτευχθεί με αυτό. Θα πρέπει να τονίσουμε από την αρχή ότι ο ΟΔΥΣΣΕΑΣ **δεν είναι** ένα Βοήθημα Επικοινωνίας.

Το μονοπάτι της πληροφορίας ανάμεσα στον άνθρωπο και τον υπολογιστή που είναι από τη φύση του χαμηλού ρυθμού επιβραδύνεται ακόμα περισσότερο από την αναπηρία του χρήστη. Αυτό είναι ίσως και το σημαντικότερο τεχνικό πρόβλημα στην ανάπτυξη της συνεργασίας του ανθρώπου με τη μηχανή μιας συνεργασίας αρκετά καλής, ώστε να μπορέσει να αποδώσει αρκετά, ώστε ο χρήστης να μπορεί να πάρει μέρος σε μια συζήτηση. Χρειάζεται εντατική έρευνα για να μπορέσει να βελτιωθεί ή διεπαφή του χρήστη ώστε να μειωθεί στο ελάχιστο ή απαιτούμενη ροή πληροφορίας ανάμεσα στον άνθρωπο και τη μηχανή, καθώς και για τη δημιουργία εναλλακτικών καναλιών διεπαφής, τα οποία θα δώσουν στον χρήστη τη δυνατότητα να κωδικοποιεί την πληροφορία πιο εύκολα και γρήγορα.

Ένα αρθρωτό επικοινωνιακό βοήθημα πρέπει να μπορεί να αντιμετωπίσει επαρκώς χρήστες με προβλήματα κίνησης, αντίληψης και διανόησης. Τα ειδικά προβλήματα του ατόμου πρέπει να αντιμετωπιστούν όταν δημιουργείται μια ειδική λύση, ενώ παράλληλα το επικοινωνιακό βοήθημα που χρησιμοποιείται θα πρέπει να ταιριάζει με την διανοητική ικανότητα του ατόμου που το χρησιμοποιεί. Είναι λοιπόν αδύνατο να χρησιμοποιηθούν συγκεκριμένοι απλοί κανόνες στην διαδικασία αυτή. Απαιτείται συχνά η χρησιμοποίηση μιας μεθόδου δοκιμής και σφάλματος και έτσι η χρησιμοποιούμενη τεχνολογία πρέπει να επιτρέπει τις γρήγορες αλλαγές. Πολλά διαφορετικά χαρακτηριστικά χρειάζονται για να ικανοποιηθούν οι φυσικές ανάγκες, τα νοητικά και γλωσσικά επίπεδα και οι επικοινωνιακές ανάγκες των ανθρώπων που χρησιμοποιούν αυτά τα βοηθήματα. Επιπλέον, οι χρήστες υποβοηθητικής τεχνολογίας (assistive technology) συχνά κριτικάρουν την έλλειψη συμβατότητας και ολοκλήρωσης διαφόρων συσκευών. Η χρήση ενός συνδυασμού από συσκευές οδηγεί σε μεγάλες δυσκολίες στην καθημερινή χρήση.

Η σχεδίαση και ανάπτυξη των ήδη υπαρχόντων επικοινωνιακών βοηθημάτων πάσχει από την έλλειψη μιας γενικής τεχνικής λύσης που θα βοηθούσε την ανάπτυξη προσαρμοσμένων στο χρήστη και λειτουργικά ευέλικτων συστημάτων με βάση γενικών κριτηρίων και εργαλείων.

Η ιδανική τεχνολογική λύση θα ήταν αυτή που θα μπορούσε να επικεντρώσει στον συγκεκριμένο χρήστη, χρησιμοποιώντας της ειδικές του ικανότητες στο μέγιστο και παράλληλα να έχει μόνο τα ειδικά χαρακτηριστικά, τα οποία εκείνος χρειάζεται. Όμως δεν είναι πρακτικό για τους προγραμματιστές να αναπτύσσουν πολλά ειδικά για τον χρήστη συστήματα. Έτσι μπορούν να ακολουθηθούν οι ακόλουθοι δρόμοι:

- Η κατασκευή περιορισμένων συστημάτων τα οποία ικανοποιούν ανάγκες συγκεκριμένων ομάδων πελατών.
- Η ανάπτυξη παραμετρικών συστημάτων για όλες τις περιπτώσεις στις παραμέτρους των οποίων δίδονται τιμές ώστε να καλύψουν τις συγκεκριμένες ανάγκες κάποιου χρήστη.
- Η ανάπτυξη αρθρωτής αρχιτεκτονικής η οποία υποστηρίζει εξειδικευμένα τμήματα - συστατικά (components) για μικρότερες ομάδες πιθανών χρηστών.

Η πρώτη προσέγγιση είναι η πιο εύκολη να επιτευχθεί τεχνικά, ενώ επιτυγχάνεται ευκολία χρήσης και αποτελεσματικότητα. Η απαιτήσεις μπορούν να ικανοποιηθούν και να επανεπεξεργαστούν όπως συνήθως χρειάζεται, μέσα από μια διαρκή διαδικασία αξιολόγησης με κάποια ομάδα σταθερών πελατών. Το πρόβλημα είναι ότι παράγονται ακριβά προϊόντα λόγω των μικρών αριθμών παραγωγής. Επίσης, θα υπάρχουν πάντα μερικοί πελάτες οι οποίοι παραβλέπονται λόγω τις ειδικότητας των αναγκών τους.

Οι τελευταίες δύο προσεγγίσεις έχουν τη δυνατότητα να καλύψουν μεγαλύτερες ομάδες χρηστών. Η κάθε μια από αυτές έχει τα δικά της πλεονεκτήματα και μειονεκτήματα. Στην δεύτερη προσέγγιση ο υπεύθυνος για την ανάπτυξη μπορεί να διατηρήσει τον έλεγχο για όλα τα θέματα που αφορούν το σύστημα και έτσι να εγγυηθεί έλεγχο της ποιότητας. Παραταύτα, ένα τέτοιο σύστημα τείνει να γίνει πολύπλοκο και ο πελάτης πρέπει να αγοράσει το όλο πακέτο. Ο μεγάλος αριθμός παραμέτρων προσαρμογής επίσης μπορεί να οδηγήσει τον πελάτη σε αδιέξοδο μιας και θα είναι δύσκολο γι' αυτόν να καταλάβει ποιές παράμετροι θα πρέπει να ρυθμιστούν και ποιές όχι.

Η αρθρωτή προσέγγιση είναι η καλύτερη για να ακολουθηθεί για ένα σύνολο από λόγους. Ο πιο σημαντικός είναι το γεγονός ότι οι χρήστες θα μπορούσαν στις περισσότερες

περιπτώσεις να αγοράσουν μόνο τις λειτουργίες εκείνες που τους είναι απαραίτητες. Αυτό θα μπορούσε να μεταφραστεί σε μικρότερο κόστος γιατί είναι ευκολότερο να αναπτυχθούν πολλά μικρά τμήματα λογισμικού και υλικού με περιορισμένες δυνατότητες. Επίσης η δυσκολία και το κόστος προσαρμογής είναι μικρότερο γιατί μόνο σχετικοί με το χρήστη παράμετροι πρέπει να κατανοηθούν και να ρυθμιστούν. Αφού οι χρήστες δεν έχουν πρόσβαση σε χαρακτηριστικά που δεν τους είναι γνωστά, είναι λιγότερο πιθανό να μπερδευτούν από αυτά. Μια αξιόπιστη αρθρωτή σχεδίαση θα μπορούσε να λύσει το πρόβλημα των αναγκών του χρήστη καθώς αυτές αλλάζουν με τον χρόνο. Δυστυχώς, τα αρθρωτά συστήματα δεν είναι εύκολο να υλοποιηθούν. Αυτό συμβαίνει γιατί οι υπεύθυνοι ανάπτυξης δεν μπορούν να γνωρίζουν πώς διάφορα τμήματα μπορούν να συνδυαστούν ή ακόμα μπορεί και να μην είναι γνωστή σε αυτούς η ύπαρξη κάποιου ήδη έτοιμου τμήματος. Έτσι τα διάφορα τμήματα θα πρέπει να καταγραφούν και να χωριστούν σε κατηγορίες με κοινούς στόχους. Σε κάθε τέτοιο συστατικό (component) ή τμήμα ενός βοηθήματος χρειάζεται μια λογική ισορροπία ανάμεσα στην ευελιξία και στα τυποποιημένα χαρακτηριστικά.

Ο ΟΔΥΣΣΕΑΣ φιλοδοξεί λοιπόν να δώσει λύσεις στα εξής σημαντικά προβλήματα Βοηθημάτων Επικοινωνίας που σχετίζονται με:

- Την έλλειψη ποικιλίας σε Βοηθήματα Επικοινωνίας στην αγορά, ιδιαίτερα την Ελληνική
- Την υψηλή τιμή αγοράς τέτοιων προϊόντων.
- Τις μεγάλες ανάγκες παραμετροποίησης.
- Τις μεταβαλλόμενες απαιτήσεις των χρηστών.
- Τις γρήγορες αλλαγές στην τεχνολογία λογισμικού και υλικού.
- Την έλλειψη επαναχρησιμοποίησης κώδικα και συστατικών.
- Τον κατακερματισμό της αγοράς.
- Την έλλειψη συμβατότητας μεταξύ των διαφόρων συστατικών ενός Βοηθήματος Επικοινωνίας όταν αυτά προέρχονται από διαφορετικούς κατασκευαστές.
- Την αδυναμία συνεργασίας των κατασκευαστών.
- Την έλλειψη βοηθημάτων που να υποστηρίζουν την Ελληνική γλώσσα.

Ο ΟΔΥΣΣΕΑΣ είναι προϊόν έρευνας και εξέλιξης

Το πλαίσιο ΟΔΥΣΣΕΑΣ δεν είναι η πρώτη προσπάθεια για την αντιμετώπιση των προβλημάτων της αγοράς λογισμικού Βοηθημάτων Διαπροσωπικής Επικοινωνίας, μέσω της καθιέρωσης νέων κατάλληλων αρχιτεκτονικών και τυποποιήσεων. Γενικά για τον τομέα της τεχνολογίας λογισμικού έχουν υπάρξει προτάσεις και λύσεις όπως μοντέλα αντικειμενοστραφούς προγραμματισμού - OMT [51], [52], μοντέλα προγραμματισμού με συστατικά - COM [36], [39], [44] και συνδυασμοί των προηγουμένων με εφαρμογή στον κατανεμημένο προγραμματισμό - DCOM [26], [40], CORBA [31], [29]. Αυτά τα μοντέλα και οι αρχιτεκτονικές θα μελετηθούν στο πλαίσιο της ανάπτυξης του ΟΔΥΣΣΕΑ, και θα συγκριθούν μεταξύ τους. Ιδιαίτερη έμφαση στην ανάλυση αυτή, θα δοθεί στο τεχνολογικά νεώτερο, αλλά και αρτιότερο μοντέλο, δηλαδή το COM+ [27], [28].

Συγκεκριμένα για το χώρο των Βοηθημάτων επικοινωνίας υπήρξαν προτάσεις στο παρελθόν, όπως αυτές του έργου ACCESS, το οποίο χρηματοδοτήθηκε από το Πρόγραμμα TIDE της Ευρωπαϊκής Ένωσης, και της αρχιτεκτονικής ATIC που προέκυψε από αυτό [4], [5], [6], [7], [8], [9], [10], [11]. Στα πλαίσια αυτού του έργου αναπτύχθηκε ένα πλαίσιο που περιελάμβανε μια αυτοσχέδια (proprietary) αρχιτεκτονική, πρωτόκολλο επικοινωνίας μεταξύ συστατικών και έναν αυτοσχέδιο διαχειριστή των συστατικών και των μηνυμάτων που έπρεπε να ανταλλάσσονται μεταξύ τους. Όλα αυτά μαζί κάλυπταν τις ανάγκες για μια νέα προσέγγιση στην υποβοήθηση, τυποποίηση και υποβοήθηση της δημιουργίας Βοηθημάτων Επικοινωνίας με βάση τις τότε (1995) δυνατότητες της τεχνολογίας λογισμικού. Ο ΟΔΥΣΣΕΑΣ δεν απορρίπτει τα επιτεύγματα τέτοιων παλαιότερων προσπαθειών, αλλά προχωρά ένα βήμα παραπάνω. Προσαρμόζει τις βασικές αρχές τους στη σημερινή τεχνολογία, εξελίσσει και απλοποιεί τις αρχιτεκτονικές και τις τυποποιήσεις που προτάθηκαν. Μια επίσης σημαντική προσπάθεια του παρελθόντος που μελετήθηκε στα πλαίσια του ΟΔΥΣΣΕΑ είναι και το ονομαζόμενο Comspec [16], [19]. Επρόκειτο για μια διαφορετική προσέγγιση που βασιζόταν στην φιλοσοφία της γεννήτριας εφαρμογών, ενός πακέτου λογισμικού που προσφέρει ένα πλήρες περιβάλλον ανάπτυξης Βοηθημάτων Επικοινωνίας με κατάλληλα εργαλεία και μεθοδολογίες. Αυτή η προσέγγιση θεωρήθηκε πολύ περιοριστική ως προς τις δυνατότητές της, μια και παρήγαγε πολύ συγκεκριμένους τύπους Βοηθημάτων και πολύ πολύπλοκη και χρονοβόρα για την δημιουργία μιας παρεμφερούς υποδομής. Προτιμήθηκε να αφεθεί μεγαλύτερη ελευθερία στους κατασκευαστές που σε τελική ανάλυση πρέπει να πεισθούν για την αποτελεσματικότητα, τις δυνατότητες και την ευκολία χρήσης ενός πλαισίου ανάπτυξης Βοηθημάτων Επικοινωνίας.

Ο ΟΔΥΣΣΑΣ βασίζεται στις νέες τεχνολογίες.

Το πλαίσιο «ΟΔΥΣΣΕΑΣ» θα αναπτυχθεί έτσι, ώστε να υποστηρίζει τις νέες τεχνολογίες αιχμής που κυριαρχούν στον τομέα της τεχνολογίας λογισμικού. Έτσι, θα σχεδιαστεί ώστε να υποστηρίζει την πλατφόρμα των προσωπικών υπολογιστών που είναι εφοδιασμένοι με το λειτουργικό σύστημα Windows 2000. Θα πρέπει να εκμεταλλεύεται πλήρως τις τεχνολογικές καινοτομίες που προσφέρει αυτό το λειτουργικό σύστημα σε όλες τους τις εκδόσεις, όπως το COM+ [24], [37] και τα Component Services [24], [43]. Θα πρέπει να ενθαρρύνει την ανάπτυξη συστατικών σε μοντέρνα εργαλεία ή γλώσσες προγραμματισμού όπως η Microsoft Visual Basic [33], [41] [48], [49], η Microsoft Visual C++ και η Java. Σημαντικό είναι το γεγονός ότι θα πρέπει να προτείνει τη χρησιμοποίηση μοντέρνων και ευρέως διαδεδομένων τεχνικών προγραμματισμού όπως ο αντικειμενοστραφής προγραμματισμός [28], ο προγραμματισμός που βασίζεται σε συστατικά [21] και ο κατανεμημένος προγραμματισμός [30]. Κυρίως όμως, θα πρέπει να είναι έτσι σχεδιασμένος ώστε να έχει ως εγγενές χαρακτηριστικό του την υποστήριξη του **Διαδικτύου** και των υπηρεσιών του.

Ο ΟΔΥΣΣΕΑΣ επικεντρώνεται στην Τεχνολογία Λογισμικού.

Το πλαίσιο ΟΔΥΣΣΕΑΣ στην πραγματικότητα θα προτείνει μια σειρά από οδηγίες που απευθύνονται στους κατασκευαστές Βοηθημάτων Επικοινωνίας ή τμημάτων τέτοιων εφαρμογών, ώστε να κατασκευάζονται αποδοτικά, διαδραστικά, αρθρωτά, επεκτάσιμα και επαναχρησιμοποιούμενα τμήματα λογισμικού.

Επίσης, θα πρέπει να προσφέρει υλοποιημένα τμήματα λογισμικού για τη δοκιμή των προτεινόμενων τεχνολογιών και αρχιτεκτονικών, τα οποία χρησιμεύουν και ως πρότυπα υποδείγματα για τον τρόπο που θα πρέπει να κατασκευάζονται τμήματα Βοηθημάτων Επικοινωνίας.

Το πλαίσιο θα πρέπει να περιλαμβάνει πλήρη περιγραφή των τεχνικών προγραμματισμού και των τυποποιήσεων που πρέπει να ακολουθηθούν, ώστε να παραχθούν από ανεξάρτητους κατασκευαστές τμήματα λογισμικού που θα μπορούν να συνεργάζονται μεταξύ τους με σκοπό να ολοκληρώνονται σε ένα ενιαίο και λειτουργικό Βοήθημα Επικοινωνίας. Αυτά τα τμήματα λογισμικού λέγονται **συστατικά (components)** [59].

Τέλος, ο ΟΔΥΣΣΕΑΣ θα πρέπει να προσφέρει ένα εκτελέσιμο πρόγραμμα το οποίο χρησιμεύει για την εκκίνηση του βοηθήματος Διαπροσωπικής Επικοινωνίας και τον συγχρονισμό και την ταξινόμηση (ή την «ενορχήστρωση») των συστατικών που το αποτελούν.

Όλα αυτά που αναφέρθηκαν και θα περιγραφούν με λεπτομέρεια στα επόμενα κεφάλαια, αφορούν την κύρια ομάδα χρηστών στην οποία απευθύνεται το «πλαίσιο ΟΔΥΣΣΕΑΣ» και είναι οι **προγραμματιστές ή οι κατασκευαστές λογισμικού**. Υπάρχει όμως και μια δεύτερη ομάδα χρηστών στην οποία απευθύνεται ο ΟΔΥΣΣΕΑΣ και αυτή είναι οι **πωλητές ή ολοκληρωτές** Βοηθημάτων Διαπροσωπικής Επικοινωνίας. Για αυτή την ομάδα χρηστών, το πλαίσιο ΟΔΥΣΣΕΑΣ θα πρέπει προσφέρει ένα «Εγχειρίδιο Χρήσης» [53], όπου θα περιγράφονται με λεπτομέρεια οι διαδικασίες που πρέπει να ακολουθηθούν για την σύνθεση και το συγχρονισμό των έτοιμων συστατικών ενός Βοηθήματος Επικοινωνίας, καθώς και όλες οι διαχειριστικές ενέργειες που πρέπει να γίνουν για τη σωστή του λειτουργία στο στάδιο της εγκατάστασής του στον ηλεκτρονικό υπολογιστή του ΑΜΕΑ χρήστη.

Ο ΟΔΥΣΣΕΑΣ και το Διαδίκτυο

Η σχέση του πλαισίου ΟΔΥΣΣΕΑΣ με το Διαδίκτυο δεν θα πρέπει να περιορίζεται στην εγγενή υποστήριξη του Διαδικτύου από την αρχιτεκτονική του. Το πλαίσιο χρησιμοποιεί το Διαδίκτυο ως ένα πολύτιμο εργαλείο. Τα προϊόντα της υλοποίησης του ΟΔΥΣΣΕΑ [53], πρέπει να βρίσκονται διαθέσιμα στο Διαδίκτυο για να χρησιμοποιούνται από τους κατασκευαστές και τους πωλητές Βοηθημάτων Διαπροσωπικής Επικοινωνίας. Συγκεκριμένα θα διανέμονται ελεύθερα από το Διαδίκτυο τμήματα λογισμικού (βιβλιοθήκες, έτοιμα συστατικά δοκιμών και συστατικά αναφοράς) που αναπτύχθηκαν και έχουν σκοπό:

- την τυποποίηση της επικοινωνίας μεταξύ των συστατικών
- τον έλεγχο της συμβατότητας των συστατικών (υλοποιημένα συστατικά εισόδου, εξόδου και ενδιάμεσα συστατικά)
- την υποβοήθηση συγγραφής κώδικα για συστατικά του
- την εκκίνηση ενός βοηθήματος Διαπροσωπικής Επικοινωνίας και των συστατικών του

Φυσικά τα συστατικά και οι βιβλιοθήκες που αναφέρθηκαν θα είναι διαθέσιμα και ως δυαδικά αρχεία, αλλά και ως κώδικας. Δεν είναι όμως μόνο αυτό. Διαθέσιμα στο Διαδίκτυο θα βρίσκονται και όλες οι τυποποιήσεις και οι οδηγίες που δίνει ο ΟΔΥΣΣΕΑΣ προς του κατασκευαστές, αλλά και το εγχειρίδιο χρήσης για την Εγκατάσταση Βοηθημάτων Διαπροσωπικής Επικοινωνίας που βασίζονται στο πλαίσιο ΟΔΥΣΣΕΑΣ.

Τέλος ο ρόλος του Διαδικτύου είναι σημαντικότατος γιατί εκεί υπολογίζεται ότι θα συγκεντρώνεται κάθε νέα πληροφορία για νέα διαθέσιμα συστατικά Βοηθημάτων Διαπροσωπικής Επικοινωνίας, είτε αυτά κατασκευαστούν στα πλαίσια του έργου ΑΙΝΕΙΑΣ, είτε ανεξάρτητα από τρίτους κατασκευαστές. Θα είναι η πηγή για τους πωλητές

και για κάθε ενδιαφερόμενο, όπου μπορούν να δουν τι υπάρχει διαθέσιμο για τη σύνθεση ενός πλήρους βοηθήματος, αλλά και τον τρόπο που θα γίνει η σύνθεση αυτή.

Η δομή της τεχνικής έκθεσης

Στην παρούσα τεχνική έκθεση σκοπεύουμε να περιγράψουμε τι είναι το πλαίσιο ΟΔΥΣΣΕΑΣ, και να εκθέσουμε τις προδιαγραφές και τα χαρακτηριστικά του. Επιπλέον, θα παρουσιαστεί η διαδικασία έρευνας και μελέτης που ακολουθήθηκε ή τουλάχιστο μέρος αυτής για να καταλήξουμε στη λύση που προτείνουμε. Επίσης περιγράφονται λεπτομερώς οι τεχνολογίες που προτείνονται και αναπτύσσεται τι μας έκανε να τις επιλέξουμε. Η παρούσα τεχνική έκθεση αποτελείται από επτά κεφάλαια, των οποίων το πρώτο αποτελεί αυτήν την Εισαγωγή.

Στο δεύτερο κεφάλαιο αναλύεται το πρόβλημα που φιλοδοξεί να αντιμετωπίσει το Πλαίσιο Σχεδιασμού και Ανάπτυξης Βοηθημάτων Διαπροσωπικής Επικοινωνίας ΟΔΥΣΣΕΑΣ. Επίσης δίδεται μια γενική περιγραφή της λύσης που προτείνεται και των προδιαγραφών του πλαισίου. Στη συνέχεια αναλύονται τα περιεχόμενα, οι προτάσεις και τα χαρακτηριστικά του ΟΔΥΣΣΕΑ σε έξι διαφορετικούς άξονες:

- Τυποποιήσεις και οδηγίες
- Λογισμικό υποβοήθησης υλοποίησης συστατικών
- Λογισμικό ελέγχου συμβατότητας συστατικών
- Λογισμικό σύνδεσης συστατικών
- Ολοκλήρωση του πλαισίου
- Υποστήριξη Διαδικτύου

Στο τρίτο κεφάλαιο παρουσιάζονται διάφορες προσεγγίσεις για ένα βασικό πρόβλημα Τεχνολογίας Λογισμικού που απασχολεί το πλαίσιο ΟΔΥΣΣΕΑΣ και είναι αυτό τις Επικοινωνίας μεταξύ Διεργασιών. Αναφέρονται τεχνικές όπως η χρησιμοποίηση του Clipboard των Windows, του OLE που βασίζεται στο COM, της τεχνολογίας Dynamic Data Exchange (DDE) [54], [55], του File Mapping, των Mailslots, των Σωληνώσεων (pipes) [56], των Remote Procedure Calls κλπ. Όλα αυτά αναλύονται ως λύσεις για την επικοινωνία μεταξύ των συστατικών και ως μια αναφορά για τους τρόπους επίτευξης αυτής της επικοινωνίας που χρησιμοποιούνταν πριν τον ΟΔΥΣΣΕΑ. Επιλέγεται ως καταλληλότερη για τις ανάγκες του έργου η τεχνολογία που βασίζεται σε COM και αναλύεται στο επόμενο κεφάλαιο με λεπτομέρεια.

Το τέταρτο κεφάλαιο παρουσιάζει με λεπτομέρεια το Μοντέλο Συστατικών και Αντικειμένων (Component Object Model – COM) [36], [39], το οποίο αποτελούσε την ευρύτερα διαδεδομένη τεχνολογία στο χώρο των μοντέλων, τεχνικών και αρχιτεκτονικών προγραμματισμού, όταν ξεκίνησε η μελέτη για το πλαίσιο ΟΔΥΣΣΕΑΣ. Είναι το μοντέλο που χρησιμοποιείται από τους περισσότερους προγραμματιστές σήμερα και ικανοποιεί κατά μεγάλο μέρος τις απαιτήσεις του ΟΔΥΣΣΕΑ. Πραγματεύεται τις έννοιες των αντικειμένων και των πολύ σημαντικών για τον ΟΔΥΣΣΕΑ **διεπαφών**, την επαναχρησιμοποίηση των αντικειμένων και το μοντέλο πελάτη/εξυπηρέτη (client/server) για την επικοινωνία. Αυτό το μοντέλο και η ενσωματωμένη υποδομή που παρέχει στο λειτουργικό σύστημα των Windows 95 και 98 θα μας ήταν αρκετό, αν δεν είχαμε την επιθυμία για εγγενή υποστήριξη απομακρυσμένων διεργασιών και του Διαδικτύου. Στο επόμενο λοιπόν κεφάλαιο εξετάζουμε την επέκτασή του δηλαδή το DCOM.

Το πέμπτο κεφάλαιο ασχολείται με την αρχιτεκτονική και τα χαρακτηριστικά της «δικτυακής» επέκτασης του COM που ονομάζεται Κατανεμημένο Μοντέλο Συστατικών και Αντικειμένων (Distributed Component Object Model – DCOM). Υπάρχει ένας βασικός αντίπαλος αυτού του μοντέλου που ονομάζεται CORBA [31], [29] και ήταν επιβεβλημένη η λεπτομερής σύγκριση των δύο για να καταλήξουμε ότι το πρώτο είναι αυτό που ταιριάζει καλύτερα στις απαιτήσεις του ΟΔΥΣΣΕΑ. Η σύγκριση έγινε σε τρία επίπεδα που περιγράφονται λεπτομερώς και είναι τα εξής:

- Κορυφαίο Επίπεδο: Βασική Αρχιτεκτονική Προγραμματισμού
- Μεσαίο Επίπεδο: Αρχιτεκτονική Απομακρυσμένων Διαδικασιών
- Χαμηλό Επίπεδο: Αρχιτεκτονική Πρωτοκόλλου Επικοινωνίας

Επίσης και στο σημείο αυτό, θα μας ήταν αρκετά όσα μελετήσαμε και αξιολογήσαμε μέχρι τώρα ως υποδομή για το πλαίσιο ΟΔΥΣΣΕΑΣ, αν δεν μας προλάβαινε πάλι η τεχνολογία, και τις λίγες ελλείψεις που υπήρχαν δεν τις κάλυπτε μια νέα τεχνολογία που εδραιώθηκε κατά τη διάρκεια της μελέτης και ήταν το COM+ που περιγράφεται στο επόμενο κεφάλαιο.

Το έκτο κεφάλαιο ασχολείται με την «ψυχή» του πλαισίου «ΟΔΥΣΣΕΑΣ». Πρόκειται για τη νεότερη τεχνολογία στο χώρο των μοντέλων προγραμματισμού και ονομάζεται COM+ Component Services (Υπηρεσίες Συστατικών). Είναι μια αρχιτεκτονική που καλύπτει όλα τα κενά και τις αδυναμίες που είχαν οι προηγούμενες προσεγγίσεις που αντιμετώπιζαν τα προβλήματα που φιλοδοξεί να λύσει ο ΟΔΥΣΣΕΑΣ. Στην ουσία αποτελεί μια εξέλιξη όλων των προηγούμενων μοντέλων που περιγράφτηκαν και επιλέχτηκαν, δηλαδή του COM και του DCOM σε συνδυασμό με ένα σύνολο από πολύτιμες λειτουργίες και υπηρεσίες ενσωματωμένες στο λειτουργικό σύστημα των Windows 2000. Ενσωματώνει επίσης και την εξέλιξη του Microsoft Transaction Server, μιας υπηρεσίας που προσφερόταν μέχρι τώρα στα λειτουργικά συστήματα με τεχνολογία NT (Windows NT) και εξυπηρετούσε πολλαπλές ανάγκες επικοινωνίας μεταξύ διεργασιών ανεξάρτητα με το που βρίσκονται (στο ίδιο μηχάνημα, στο δίκτυο ή στο Διαδίκτυο). Το νέο μοντέλο δεν καταργεί ό’τι έχει εδραιωθεί μέχρι τώρα, αλλά το βελτιώνει και το απλοποιεί. Παρουσιάζεται μια ανασκόπηση για την εξέλιξη του μοντέλου, όπου φαίνεται και η σχέση του με τα προηγούμενα και στη συνέχεια αναλύονται λεπτομερώς οι επιμέρους υπηρεσίες που προσφέρει το μοντέλο και συγκεκριμένα αυτές στις οποίες στηρίζεται το πλαίσιο ΟΔΥΣΣΕΑΣ.

Το κεφάλαιο 7 ασχολείται με λεπτομέρειες και αποσαφηνίσεις του σχεδιασμού του ΟΔΥΣΣΕΑ, όπως αυτός περιγράφηκε γενικά στο κεφάλαιο 2. Συγκεκριμενοποιούνται οι ομάδες χρηστών του πλαισίου και αναφέρεται τι προσφέρει η υλοποίηση του ΟΔΥΣΣΕΑ σε αυτούς τους χρήστες, όπως επίσης περιγράφεται και ο σχεδιασμός του πρωτοκόλλου επικοινωνία μεταξύ των συστατικών ενός Βοηθήματος Επικοινωνίας και της διεπαφής που χρησιμοποιεί ο ΟΔΥΣΣΕΑΣ για την επικοινωνίας αυτή. Στο έβδομο κεφάλαιο περιλαμβάνεται επίσης ο σχεδιασμός όλων των προγραμμάτων που αναπτύχθηκαν στα πλαίσια του ΟΔΥΣΣΕΑ και παρουσιάζονται οι προδιαγραφές και τα χαρακτηριστικά τους. Παρουσιάζονται τα συστατικά δοκιμών και το πρόγραμμα εκκίνησης του Βοηθήματος Διαπροσωπικής επικοινωνίας. Επειδή δεν πρόκειται για ολοκληρωμένες εφαρμογές αλλά για μικρά τμήματα ενός συστήματος, δεν θα ακολουθηθεί η παραδοσιακή διαδικασία παρουσίασης του σχεδιασμού με ειδικές γλώσσες μοντελοποίησης όπως η Ενοποιημένης Γλώσσα Μοντελοποίησης (Unified Modeling Language – UML) [60], ούτε θα χρησιμοποιηθούν πολύπλοκα εργαλεία όπως το Microsoft Visual Modeler [34]. Όλα αυτά

χρησιμοποιούνται συνήθως σε μοντελοποιήσεις εφαρμογών και συστημάτων μεγαλύτερης κλίμακας από αυτά που θα σχεδιαστούν στο κεφάλαιο 7. Έτσι προτιμήθηκε η προσέγγιση των κατατοπιστικών γραφικών απεικονίσεων και σχημάτων για την περιγραφή των συστατικών. Επίσης, σε αυτό το κεφάλαιο περιλαμβάνεται και ο σχεδιασμός της αλληλεπίδρασης των πωλητών Βοηθημάτων Επικοινωνίας με το διαχειριστικό περιβάλλον με το οποίο τους φέρνει σε επαφή ο ΟΔΥΣΣΕΑΣ.

Η σύνθεση όλων των αποτελεσμάτων της μελέτης των μοντέλων που αναφέρθηκαν και η υλοποίηση του πλαισίου ΟΔΥΣΣΕΑΣ είναι το αντικείμενο της συνέχειας της παρούσας Τεχνικής Έκθεσης και αποτελεί το Παραδοτέο 3.2 του έργου ΑΙΝΕΙΑΣ με τίτλο «Υλοποίηση του Πλαισίου ΟΔΥΣΣΕΑΣ και Εγχειρίδιο Χρήσης» [53]. Σε αυτήν την Τεχνική Έκθεση περιλαμβάνεται ο κώδικας βάσει του οποίου υλοποιήθηκαν τα συστατικά του ΟΔΥΣΣΕΑ. Επίσης περιλαμβάνονται οι τυποποιήσεις και οι οδηγίες που απευθύνονται στους προγραμματιστές συστατικών, καθώς και το πλήρες εγχειρίδιο χρήσης που απευθύνεται στους πωλητές Βοηθημάτων Διαπροσωπικής Επικοινωνίας.

2. Το Πλαίσιο «ΟΔΥΣΣΕΑΣ»

Ξεκινώντας την περιγραφή του Πλαισίου Σχεδιασμού και Ανάπτυξης Βοηθημάτων Διαπροσωπική Επικοινωνίας «ΟΔΥΣΣΕΑΣ» παρουσιάζουμε τα προβλήματα που φιλοδοξεί να λύσει αυτό το πλαίσιο. Στη συνέχεια δίνεται μια πρώτη περιγραφή της λύσης που προτείνεται για τα προβλήματα αυτά. Το επόμενο βήμα είναι να γίνει μια γενική περιγραφή του τι είναι ένα πλαίσιο ανάπτυξης εφαρμογών και να επιλεχτεί η κατεύθυνση που θα ακολουθηθεί κατά το σχεδιασμό του ΟΔΥΣΣΕΑ. Το παρόν κεφάλαιο θα ολοκληρωθεί με την ανάλυση της μεθοδολογίας ανάπτυξης του ΟΔΥΣΣΕΑ και με τη λεπτομερή περιγραφή του πλαισίου στους άξονες που σχεδιάστηκε.

2.1. Το πρόβλημα

Το πρόβλημα επικοινωνίας που έχουν αρκετά Άτομα Με Ειδικές Ανάγκες (ΑΜΕΑ) είναι σοβαρότατο [2]. Λόγω κινητικών ή και νοητικών προβλημάτων υπάρχουν στη χώρα μας πολλοί άνθρωποι που αδυνατούν να ζήσουν μια ζωή φυσιολογική. Βοηθήματα Επαυξητικής και Εναλλακτικής Επικοινωνίας (ΕΕΕ) μπορούν να δώσουν λύση σε αυτό το πρόβλημα και η τεχνολογία είναι σε θέση να βελτιώσει δραματικά την ποιότητα ζωής και την ένταξη στην κοινωνία των ΑΜΕΑ με ειδικές επικοινωνιακές ανάγκες [3].

Σήμερα, αν και υπάρχουν παγκοσμίως πολλές μεμονωμένες λύσεις για βοηθήματα επικοινωνίας που απευθύνονται σε ΑΜΕΑ [11], είναι έντονο το πρόβλημα της διαθεσιμότητας των βοηθημάτων αυτών στους Έλληνες χρήστες από τη μία, αλλά και της συμβατότητας των προϊόντων των διάφορων εταιριών μεταξύ τους.

Ένα ΑΜΕΑ στην Ελλάδα είναι πολύ πιθανό να μην γνωρίζει ή να μην έχει πρόσβαση σε πληροφορίες που θα μπορούσαν να του προσφέρουν μία έτοιμη λύση για το επικοινωνιακό του πρόβλημα. Βέβαια θα περίμενε κανείς οι θεραπευτές του ατόμου αυτού να είναι σε θέση και να έχουν τα εφόδια να ψάξουν και να βρουν στην παγκόσμια αγορά κάποιο προϊόν που θα έκανε την καθημερινή ζωή του πιο εύκολη στον τομέα της επικοινωνίας του. Οποιοσδήποτε ενδιαφερόμενος θα μπορούσε διενεργώντας μια σχετικά σύντομη έρευνα στον Παγκόσμιο Ιστό, να βρει κάποια λύση βασισμένη στην τεχνολογία των υπολογιστών, η οποία θα προσφέρει στο ΑΜΕΑ νέες δυνατότητες διαπροσωπικής επικοινωνίας. Δυστυχώς, ακόμη και οι θεραπευτές ή οι άνθρωποι που βοηθούν τα ΑΜΕΑ στην καθημερινή τους ζωή δεν βρίσκουν εύκολη τη διαδικασία αυτή, και είναι πολύ πιθανό να μη γνωρίζουν την ύπαρξη στην αγορά προϊόντων που οι ίδιοι και τα ΑΜΕΑ έχουν φανταστεί τη λειτουργικότητά τους και εύχονται να ήταν διαθέσιμα σε αυτούς. Σε αυτές τις δυσκολίες θα πρέπει να προσθέσουμε και το γεγονός ότι το σύνολο σχεδόν των Διαδικτυακών τόπων που ασχολούνται με τέτοια θέματα είναι ξενόγλωσσοι. Επίσης, ξενόγλωσσα είναι και τα περισσότερα σχετικά προϊόντα που κυκλοφορούν στην αγορά, και το ίδιο φυσικά ισχύει και για τα εγχειρίδια χρήσης τους.

Οι εταιρίες κατασκευής τέτοιων προϊόντων από την άλλη πλευρά, ίσως προσφέρουν ανακούφιση και βελτίωση της ποιότητας ζωής στις χώρες που εδρεύουν ή στις συγκεκριμένες τοπικές αγορές που απευθύνονται, αλλά δεν γίνονται τόσο «διάσημες» όσο θα έπρεπε στην παγκόσμια κοινότητα των ΑΜΕΑ και η διασπορά των προϊόντων τους κάνει δύσκολο τον εντοπισμό τους. Ακόμη, όσον αφορά στα προϊόντα που βασίζονται στην τεχνολογία των ηλεκτρονικών υπολογιστών, μια καλύτερη διαχείρισή τους ή ένας βελτιωμένος τρόπος ανάπτυξης και διανομής τους, θα μπορούσε να μειώσει δραματικά την

τιμή τους και συγχρόνως να αυξήσει τους αγοραστές τους. Η αγορά Βοηθημάτων Διαπροσωπικής Επικοινωνίας είναι κατακερματισμένη. Οι εταιρίες αναπτύσσουν τα προϊόντα τους «κατά βούληση» και ανεξάρτητα η μία από την άλλη με αποτέλεσμα η κάθε εταιρία να μην είναι σε θέση να καλύψει τον αριθμό των περιπτώσεων επικοινωνιακών αναγκών που θα ήθελε στον χρόνο που θα έπρεπε, αλλά και να υπάρχει μεγάλη επικάλυψη στη διαδικασία ανάπτυξης βοηθημάτων επικοινωνίας με επιβαρυντικές επιπτώσεις στο χρόνο ανάπτυξης και στην τελική τιμή του προϊόντος.

Τέλος, αναπτύσσονται πολύ εξειδικευμένες λύσεις βοηθημάτων επικοινωνίας με καμία συμβατότητα μεταξύ τους, οι οποίες δεν υπακούουν σε μια κοινά αποδεκτή τυποποίηση ή σε κάποιους κανόνες ανάπτυξης και προδιαγραφών των βοηθημάτων. Οι εξειδικευμένες αυτές λύσεις είναι ακριβές και δύσκολα προσαρμόσιμες σε διαφορετικές επικοινωνιακές ανάγκες από αυτές για τις οποίες σχεδιάστηκαν και υλοποιήθηκαν. Αναγκαστικά λοιπόν μένει μερίδα της αγοράς των ΑΜΕΑ έξω από τα ενδιαφέροντα των εταιριών, μια και το εξειδικευμένο προϊόν που θα απευθυνόταν σε μικρή αγορά θα κατέληγε να είναι πανάκριβο.

Από την άλλη πλευρά, αν η διαδικασία ανάπτυξης και σχεδιασμού των βοηθημάτων βασιζόταν στη φιλοσοφία ότι το τελικό προϊόν θα πρέπει να απευθύνεται στο μεγαλύτερο δυνατό κομμάτι της αγοράς ή των επικοινωνιακών αναγκών, να είναι πλήρως παραμετροποιήσιμο και προσαρμόσιμο, αυτό ίσως να οδηγούσε πάλι σε μεγάλο χρόνο και κόπο ανάπτυξης αλλά και σε υψηλή τιμή του προϊόντος.

Παρατηρείται στην αγορά ότι αν και υπάρχουν «κομμάτια» ενός βοηθήματος επικοινωνίας διαθέσιμα και φτηνά, αυτά τα «κομμάτια» αναπτύσσονται εκ νέου στα πλαίσια ενός «ολοκληρωμένου» βοηθήματος. Αν η μία εταιρία για παράδειγμα δραστηριοποιείται στο χώρο της σύνθεσης ομιλίας και έχει έτοιμα προϊόντα (συνθέτες ομιλίας) και η άλλη εταιρία κατασκευάζει ένα βοήθημα επικοινωνίας που κάνει χρήση του συνθέτη ομιλίας, το πιθανότερο είναι η δεύτερη εταιρία να αναπτύξει από την αρχή το δικό της συνθέτη ομιλίας. Αυτό φαίνεται σαν διπλός κόπος με συνέπεια διπλό χρόνο και διπλό κόστος ανάπτυξης. Επίσης όσον αφορά την ποιότητα του τελικού προϊόντος, είναι πιο πιθανό το αποτέλεσμα να ήταν καλύτερο αν για παράδειγμα στην ανάπτυξη ενός υπολογιστικού συστήματος που αποτελεί βοήθημα επικοινωνίας, μπορούσαν να συνεισφέρουν στον τομέα της σύνθεσης ομιλίας οι ειδικοί όπως επίσης και στον τομέα της εργονομίας και τέλος στον τομέα της πρόβλεψης λέξεων για τη γρηγορότερη επικοινωνία πάλι οι ειδικοί του χώρου αυτού.

Μεγάλο πρόβλημα θεωρείται και η αδυναμία υποστήριξης των μεταβαλλόμενων αναγκών των χρηστών Βοηθημάτων Διαπροσωπικής Επικοινωνίας. Αυτό το στοιχείο δημιουργεί την απαίτηση για βοηθήματα πλήρως παραμετροποιήσιμα, εύκολα τροποποιήσιμα, με δυνατότητες προσθαφαίρεσης σε αυτά λειτουργικοτήτων και υπηρεσιών. Με τους παραδοσιακούς τρόπους κατασκευής Βοηθημάτων τέτοιες δυνατότητες θεωρούνται «εξωτικές».

Μιλώντας πιο συγκεκριμένα για την τεχνολογία των υπολογιστών και της πληροφορικής, η συνεχείς καινοτομίες στο hardware και το software έχουν φέρει στις επιφάνειες εργασίας και τα δίκτυα των χρηστών μια πλειάδα δυνατών και σύγχρονων εφαρμογών. Όμως αυτή η ανάπτυξη έχει φέρει και τα ανάλογα προβλήματα στους προγραμματιστές, τους πωλητές λογισμικού και τους χρήστες:

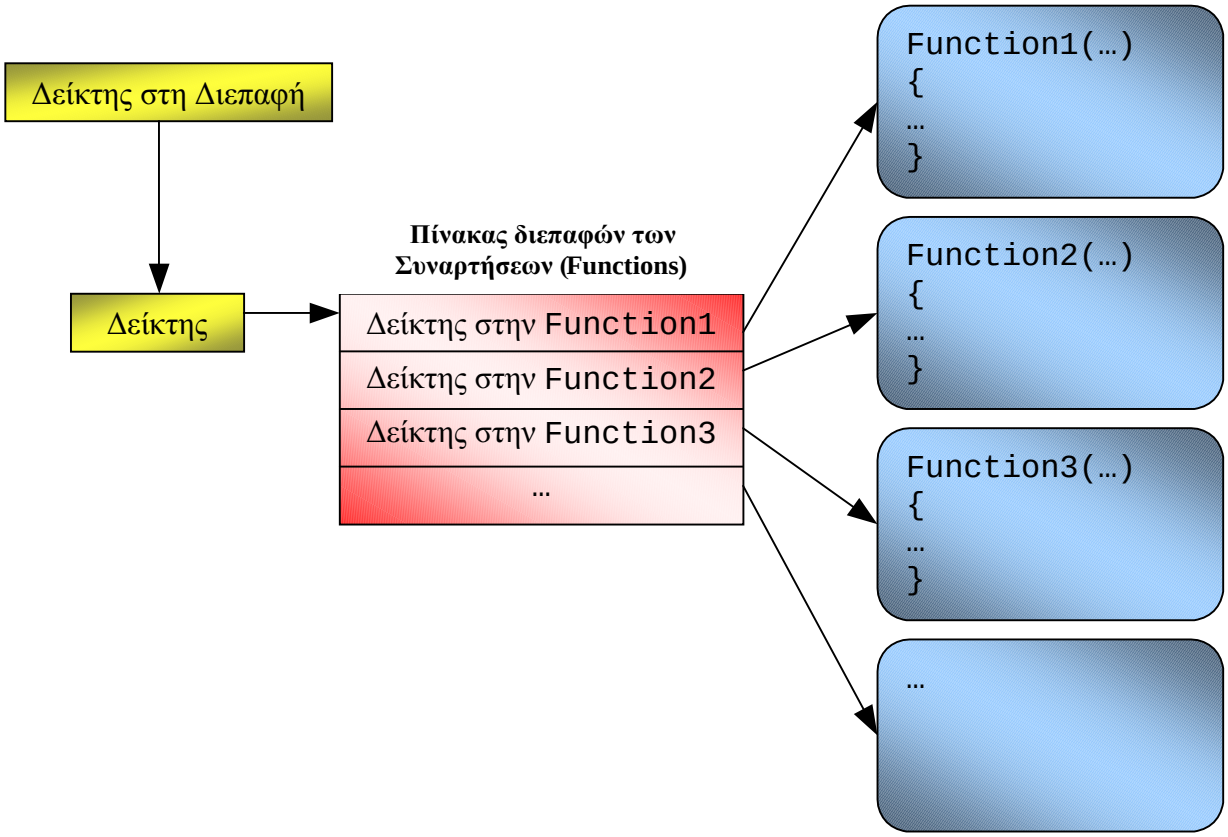
- Οι σημερινές εφαρμογές είναι μεγάλες και πολύπλοκες. Παίρνουν πολύ χρόνο για την ανάπτυξή τους, είναι δύσκολη και ακριβή η συντήρησή τους και παρακινδυνευμένη η επέκτασή τους με πρόσθετες λειτουργίες.
- Οι εφαρμογές είναι μονολιθικές. Έρχονται πακεταρισμένες με μεγάλη ποικιλία χαρακτηριστικών, αλλά τα πιο πολλά χαρακτηριστικά δεν μπορούν να αφαιρεθούν, να αναβαθμιστούν ανεξάρτητα, ή να αντικατασταθούν από εναλλακτικά.
- Οι εφαρμογές δεν ολοκληρώνονται εύκολα. Τα δεδομένα και η λειτουργικότητα μιας εφαρμογής δεν είναι εύκολα προσβάσιμες σε άλλες εφαρμογές, ακόμη κι αν οι εφαρμογές είναι γραμμένες στην ίδια γλώσσα προγραμματισμού και τρέχουν στον ίδιο υπολογιστή.
- Τα λειτουργικά συστήματα έχουν επίσης τα σχετικά προβλήματα. Δεν είναι αρκετά αρθρωτά και είναι δύσκολο να παρακαμφθούν, να αναβαθμιστούν, ή να αντικατασταθούν οι λειτουργίες που παρέχουν με έναν καθαρό και ευέλικτο τρόπο.
- Τα μοντέλα προγραμματισμού είναι ασυνεπή χωρίς να υπάρχει ιδιαίτερος λόγος. Ακόμα και όταν οι εφαρμογές έχουν τη δυνατότητα συνεργασίας, οι υπηρεσίες τους προσφέρονται σε άλλες εφαρμογές με διαφορετικό τρόπο από τις υπηρεσίες που προσφέρονται από το λειτουργικό σύστημα ή το δίκτυο. Ακόμα, τα προγραμματιστικά μοντέλα διαφέρουν πολύ ανάλογα με το αν η υπηρεσία προέρχεται από κάποιον προμηθευτή στον ίδιο χώρο διευθύνσεων με αυτόν του προγράμματος πελάτη (με δυναμική σύνδεση dynamic linking), από μια ξεχωριστή διεργασία στον ίδιον υπολογιστή, από το λειτουργικό σύστημα, ή από έναν παροχέα που τρέχει σε διαφορετικό υπολογιστή (ή μια ομάδα ξεχωριστών υπολογιστών) στο δίκτυο.

Επιπροσθέτως, ως αποτέλεσμα των τάσεων για μείωση του μεγέθους του hardware και αύξησης της πολυπλοκότητας του software, δημιουργείται η ανάγκη για ένα νέο στυλ προγραμματισμού που είναι αρθρωτό και βασίζεται στην αρχιτεκτονική πελάτη/εξυπηρέτη (client/server) και στα συστατικά (components) [13], [15]. Αυτό το στυλ απαιτεί:

- Δυνατότητες για την εύρεση και χρήση των παροχέων υπηρεσιών, για τη διαπραγμάτευση των υπηρεσιών με αυτούς και για την επέκταση και βελτίωση των παροχέων με ένα τρόπο που δεν θα αχρηστεύει τις ήδη υπάρχουσες Εκδόσεις των υπηρεσιών που μπορεί να έχουν οι καταναλωτές.
- Χρήση αντικειμενοστραφούς (object-oriented) αρχιτεκτονικής των υπηρεσιών και του συστήματος, για να ταιριάζουν καλύτερα με τα εργαλεία object-oriented ανάπτυξης νέας γενιάς, για να αντιμετωπίζεται η αυξανόμενη πολυπλοκότητα του λογισμικού μέσω της αύξησης της δυνατότητας άρθρωσης (modularity), για να επαναχρησιμοποιούνται ήδη υπάρχουσες λύσεις και για να διευκολύνεται η σχεδίαση πιο αυτόνομων συστατικών (components) λογισμικού.
- Κατανεμημένο σχεδιασμό για να παρέχεται μια μοναδική εικόνα στους χρήστες και τις εφαρμογές και για να επιτρέπεται η χρήση των υπηρεσιών σε ένα δικτυωμένο περιβάλλον ανεξάρτητα από την θέση, την αρχιτεκτονική ή το περιβάλλον υλοποίησης.

Σαν παράδειγμα για την επεξήγηση των θεμάτων αυτών, μπορούμε να θεωρήσουμε το πρόβλημα της δημιουργίας ενός API (Application Programming Interface) μιας υπηρεσίας του συστήματος που λειτουργεί με πολλαπλούς παροχείς με έναν «πολυμορφικό» τρόπο. Αυτό σημαίνει ότι ένας πελάτης της υπηρεσίας μπορεί με διαφάνεια να χρησιμοποιήσει οποιονδήποτε παροχέα της υπηρεσίας χωρίς να γνωρίζει ποιος συγκεκριμένος παροχέας – ή υλοποίηση – χρησιμοποιείται. Στα παραδοσιακά συστήματα υπάρχει ένα κεντρικό τμήμα

κώδικα το οποίο καλείται από όλες τις εφαρμογές για να προσπελάσουν λειτουργίες όπως η επιλογή ενός αντικειμένου και η σύνδεση με αυτό. Στην ουσία ο διαχειριστής υπηρεσιών είναι ένα είδος «διαχειριστή αντικειμένων». Όμως, αφού οι εφαρμογές έχουν χρησιμοποιήσει αυτές τις λειτουργίες του διαχειριστή αντικειμένων και είναι συνδεδεμένες σε έναν παροχέα υπηρεσιών, ο διαχειριστής αντικειμένων απλά παραμένει και επιβάλλει επιβάρυνση σε όλες τις εφαρμογές όπως φαίνεται στο Σχήμα 1.



Σχήμα 1: Τα παραδοσιακά APIs υπηρεσιών συστήματος απαιτούν από όλες τις υπηρεσίες να επικοινωνούν μέσω ενός κεντρικού διαχειριστή με την αντίστοιχη επιβάρυνση (overhead).

Εκτός από την επιβάρυνση που περιγράφηκε, ένα άλλο σημαντικό πρόβλημα με τα παραδοσιακά μοντέλα υπηρεσιών είναι ότι είναι αδύνατο για τον παροχέα να γνωστοποιήσει νέες, βελτιωμένες δυνατότητες στους πιθανούς χρήστες με έναν τυποποιημένο τρόπο. Μια καλά σχεδιασμένη παραδοσιακή αρχιτεκτονική υπηρεσίας μπορεί να παρέχει την έννοια των διαφορετικών επιπέδων υπηρεσίας (το Microsoft Open Database Connectivity (ODBC) είναι τέτοιο API). Οι εφαρμογές μπορούν να υπολογίζουν στο ελάχιστο επίπεδο υπηρεσίας και μπορούν να διαπιστώσουν κατά τη διάρκεια της εκτέλεσης αν ο παροχέας υποστηρίζει υψηλότερα επίπεδα υπηρεσίας, αλλά οι παροχείς είναι αναγκασμένοι να παρέχουν τα επίπεδα υπηρεσιών που καθορίζονται από το API. Δεν μπορούν να παράσχουν εύκολα νέες δυνατότητες και μετά να ειδοποιήσουν τους καταναλωτές, ώστε να τις προσπελάσουν φθηνά και με έναν τρόπο που ταιριάζει με το τυπικό μοντέλο. Αν πάρουμε σαν παράδειγμα το ODBC, ο πωλητής ενός παροχέα βάσης

δεδομένων που κάνει περισσότερα πράγματα από αυτά που επιτρέπει η τρέχουσα τυποποίηση ODBC, πρέπει να πείσει τη Microsoft να αναθεωρήσει το στάνταρ έτσι ώστε να υποστηρίζει τις πρόσθετες δυνατότητες του πωλητή. Έτσι, οι παραδοσιακές αρχιτεκτονικές υπηρεσιών δεν μπορούν εύκολα να επεκταθούν ή να συμπληρωθούν με έναν αποκεντρωμένο τρόπο.

Οι παραδοσιακές αρχιτεκτονικές έχουν επίσης την τάση να είναι περιορισμένες ως προς την τάση να εξελίσσονται συμπαγώς καθώς οι υπηρεσίες αναθεωρούνται ή αλλάζουν έκδοση. Το πρόβλημα με την παραδοσιακή διαχείριση των Εκδόσεων ενός λογισμικού με αυτόν τον τρόπο είναι δύσκολο για τον κώδικα να δείχνει πως *ακριβώς* διαφέρει από μια προηγούμενη έκδοση, και ακόμα χειρότερα, για τους πελάτες αυτού του κώδικα να προσαρμόζονται κατάλληλα στις νέες Εκδόσεις – ή να μην προσαρμόζονται καθόλου αν θέλουν να υποστηρίζουν μόνο την προηγούμενη έκδοση. Αυτό το πρόβλημα μπορεί να αντιμετωπιστεί λογικά σε ένα παραδοσιακό σύστημα όταν (i) υπάρχει μόνο ένας παροχέας μιας συγκεκριμένης υπηρεσίας, (ii) ο αριθμός της έκδοσης της υπηρεσίας ελέγχεται από τον καταναλωτή όταν δεσμεύεται με την υπηρεσία, (iii) η υπηρεσία επεκτείνεται με μόνο προς τα πίσω συμβατότητα – για παράδειγμα, μπορούν μόνο να προστεθούν και ποτέ να αφαιρεθούν δυνατότητες (σημαντικός περιορισμός) – έτσι ώστε ο παροχέας της έκδοσης N να μπορεί επίσης να λειτουργήσει με καταναλωτές της έκδοσης 1 έως N-1 και (iv) δεν μεταφέρονται από καταναλωτή σε καταναλωτή αναφορές σε ένα στιγμιότυπο μιας υπηρεσίας που τρέχει. Αλλά τέτοιοι περιορισμοί είναι προφανώς απαράδεκτοι σε ένα κατανεμημένο, αρθρωτό σύστημα με πολλαπλούς κατασκευαστές και πολυμορφικούς παροχείς υπηρεσιών.

Αυτά τα προβλήματα της διαχείρισης των υπηρεσιών, της επεκτασιμότητας και της διαχείρισης των Εκδόσεων προκαλούν και τα προβλήματα που αναφέρθηκαν παραπάνω. Η πολυπλοκότητα των εφαρμογών συνεχίζει να αυξάνεται καθώς γίνεται όλο και δυσκολότερο να επεκταθεί η λειτουργικότητα. Οι μονολιθικές εφαρμογές είναι δημοφιλείς επειδή είναι ασφαλέστερο και ευκολότερο να συγκεντρωθούν όλες οι συσχετιζόμενες υπηρεσίες και ο κώδικας που τις χρησιμοποιεί σε ένα πακέτο. Η διαλειτουργικότητα επίσης περιορίζεται καθώς οι μονολιθικές εφαρμογές δεν αφήνουν ανεξάρτητους agents να προσπελάσουν τη λειτουργικότητά τους. Πάντως, επειδή οι τελικοί χρήστες απαιτούν τη διαλειτουργικότητα, οι εφαρμογές αναγκάζονται να την επιχειρήσουν, και οδηγούνται απευθείας στο πρόβλημα της πολυπλοκότητας, κλείνοντας έναν κύκλο προβλημάτων που περιορίζουν την πρόοδο της ανάπτυξης του λογισμικού.

2.2. Η προτεινόμενη λύση

Το Πλαίσιο Σχεδιασμού και Ανάπτυξης Εφαρμογών «ΟΔΥΣΣΕΑΣ» υπόσχεται να προσφέρει λύσεις στα προβλήματα που προαναφέρθηκαν. Στην ενότητα αυτή θα γίνει μια γενική περιγραφή και μια προκαταρκτική τεχνική προσέγγιση των λύσεων που προτείνονται από τον ΟΔΥΣΣΕΑ.

Η λύση που προτείνεται στοχεύει στο να έχει ένας κατασκευαστής τη δυνατότητα, αν τα προϊόντα του έχουν κάποια χρησιμότητα ως τμήματα ενός βοηθήματος επικοινωνίας, να κατασκευάζει λύσεις «συμβατές» με άλλα τμήματα λογισμικού που θα μπορούσαν σε συνδυασμό μεταξύ τους να αποτελέσουν ένα ολοκληρωμένο βοήθημα. Ο τρόπος για να γίνει αυτό είναι ο καθορισμός συγκεκριμένων προδιαγραφών συμβατότητας και αυστηρών οδηγιών. Αυτές οι προδιαγραφές και οι οδηγίες είναι είτε γενικές, που αφορούν για παράδειγμα στο λειτουργικό σύστημα με το οποίο θα είναι συμβατά τα επιμέρους τμήματα

ενός βοηθήματος επικοινωνίας, είτε πιο ειδικές που αναφέρονται για παράδειγμα στον τρόπο επικοινωνίας των τμημάτων αυτών μεταξύ τους.

Επίσης προτείνεται ένας «χώρος» όπου ο κάθε ενδιαφερόμενος θα μπορούσε να πάρει τις πληροφορίες που χρειάζεται για την ανάπτυξη της εφαρμογής του. Εκεί θα είναι διαθέσιμες όλες οι τυποποιήσεις, οι προδιαγραφές και οι οδηγίες που χρησιμοποιούνται από τον ΟΔΥΣΣΕΑ και θα διευκολύνεται τεχνικά η ανάκτησή τους. Στον ίδιο χώρο βέβαια θα εκτίθενται και όλα τα ήδη έτοιμα και συμβατά με τον ΟΔΥΣΣΕΑ προϊόντα με κάθε πληροφορία για τις προδιαγραφές, τη λειτουργικότητα και τη χρήση τους. Ο ίδιος χώρος θα μπορούσε να χρησιμοποιηθεί και από τους κατασκευαστές που ενδιαφέρονται να αναπτύξουν ή να υποβάλλουν ήδη έτοιμα προϊόντα τους συμβατά με τον ΟΔΥΣΣΕΑ, αλλά και από τους χρήστες, είτε πρόκειται για τα ίδια τα ΑΜΕΑ είτε για τους θεραπευτές τους, που ενδιαφέρονται να μάθουν τι υπάρχει στην αγορά. Στην καλύτερη περίπτωση θα μπορούσαν οι χρήστες να συνθέσουν το προϊόν που τους ταιριάζει και να το παραγγείλουν κατευθείαν από το χώρο αυτό. Στην ουσία αυτός ο χώρος θα είναι ένας τόπος στο Διαδίκτυο, όπου οι τελικοί χρήστες και οι πωλητές Βοηθημάτων Διαπροσωπικής επικοινωνίας θα μπορούν εύκολα να ερευνούν τις δυνατότητες που προσφέρουν τα διαθέσιμα συστατικά (components) και οι κατασκευαστές να βρίσκουν οδηγίες και βοηθήματα για την ανάπτυξη των δικών τους συστατικών (components), αλλά και να είναι σε θέση να τα εκθέσουν.

Επικεντρώνοντας πάλι το ενδιαφέρον στις τεχνικές προγραμματισμού, είναι γνωστό ότι ο αντικειμενοστραφής (object-oriented) προγραμματισμός έχει προταθεί ως λύση στα προβλήματα που μας απασχολούν. Όμως, ενώ ο object-oriented προγραμματισμός είναι ισχυρός, δεν είχε φτάσει μέχρι πρόσφατα στις πλήρεις δυνατότητές του, επειδή δεν υπήρχε τυποποιημένο ή κοινά αποδεκτό πλαίσιο μέσω του οποίου τα αντικείμενα λογισμικού (συστατικά) που έχουν δημιουργηθεί από διαφορετικούς κατασκευαστές να μπορούν να αλληλεπιδράσουν το ένα με το άλλο μέσα στον ίδιο χώρο διευθύνσεων, πόσο μάλλον μεταξύ διαφορετικών χώρων διευθύνσεων και μεταξύ συνόρων δικτύου και αρχιτεκτονικών. Το αποτέλεσμα της επανάστασης του object-oriented προγραμματισμού ήταν η παραγωγή «νησιών αντικειμένων» που δεν μπορούν να μιλήσουν μεταξύ τους με επιτυχία διαμέσου της θάλασσας των συνόρων των εφαρμογών.

Η λύση είναι ένα σύστημα στο οποίο αυτοί που αναπτύσσουν τις εφαρμογές δημιουργούν επαναχρησιμοποιούμενα *συστατικά λογισμικού*. Ένα συστατικό (component) είναι ένα επαναχρησιμοποιούμενο τμήμα λογισμικού σε δυαδική μορφή το οποίο μπορεί να «προσκολληθεί» σε άλλα συστατικά από άλλους κατασκευαστές με σχετικά μικρή προσπάθεια. Για παράδειγμα, ένα συστατικό (component) θα μπορούσε εκτελεί μια διαδικασία ορθογραφικού ελέγχου που πωλείται από ένα κατασκευαστή και μπορεί να «προσκολληθεί» σε διάφορους επεξεργαστές κειμένου από διάφορους άλλους κατασκευαστές. Τα συστατικά λογισμικού πρέπει να υπακούουν σε μια εξωτερική δυαδική τυποποίηση, αλλά η εσωτερική τους υλοποίηση είναι εντελώς αδέσμευτη. Μπορούν να κατασκευαστούν χρησιμοποιώντας διαδικαστικές (procedural) γλώσσες όπως επίσης και αντικειμενοστραφείς (object-oriented) γλώσσες και πλαίσια, αν και οι τελευταίες παρουσιάζουν πολλά πλεονεκτήματα στον κόσμο των συστατικών λογισμικού.

Τα συστατικά λογισμικού μοιάζουν αρκετά με τα συστατικά των ολοκληρωμένων κυκλωμάτων (integrated circuits – IC) και το συστατικό λογισμικού είναι το ολοκληρωμένα κύκλωμα του αύριο. Η βιομηχανία λογισμικού σήμερα βρίσκεται εκεί που βρισκόταν η βιομηχανία hardware πριν από 20 χρόνια. Εκείνη την εποχή, οι κατασκευαστές ηλεκτρονικών μάθαιναν πως να συρρικνώνουν τα τρανζίστορ και να τα

βάζουν σε πακέτα έτσι ώστε να μη ξαναχρειαζόταν ποτέ κανείς να ανακαλύψει πως να κατασκευάσει μια διακριτή συνάρτηση – για παράδειγμα μια πύλη NAND. Τέτοιες συναρτήσεις υλοποιούνταν μέσα σε ένα ολοκληρωμένο κύκλωμα, ένα απλό πακέτο που οι σχεδιαστές θα μπορούσαν εύκολα να το αγοράζουν και να σχεδιάζουν γύρω από αυτό. Καθώς οι hardware συναρτήσεις γίνονταν όλο και πιο πολύπλοκες, τα ICs ολοκληρωνόταν για να σχηματίσουν πλακέτες από τσιπ και να παρέχουν πιο πολύπλοκη λειτουργικότητα και αυξημένες δυνατότητες. Καθώς τα ολοκληρωμένα κυκλώματα γινόταν μικρότερα προσφέροντας ακόμα μεγαλύτερη λειτουργικότητα, οι πλακέτες από τσιπ γινόταν απλά μεγαλύτερα τσιπ. Έτσι τώρα η τεχνολογία του hardware χρησιμοποιεί τσιπ για να κατασκευάσει ακόμα μεγαλύτερα τσιπ.

Η βιομηχανία του λογισμικού είναι τώρα σε ένα σημείο όπου οι κατασκευαστές λογισμικού απασχολήθηκαν για πολύ καιρό με την κατασκευή του λογισμικού που είναι ισοδύναμο με τα διακριτά τρανζίστορ – ρουτίνες λογισμικού.

Το μοντέλο συστατικών και αντικειμένων (Component Object Model) [21], [36] επιτρέπει στους παροχείς λογισμικού να πακετάρουν τις συναρτήσεις τους σε επαναχρησιμοποιούμενα συστατικά λογισμικού με παρόμοιο τρόπο με τα ολοκληρωμένα κυκλώματα. Το μοντέλο και τα συστατικά του φέρνει το λογισμικό σε ένα κόσμο όπου ο κατασκευαστής λογισμικού δεν χρειάζεται πια να γράψει, για παράδειγμα, έναν αλγόριθμο ταξινόμησης. Ένας αλγόριθμος ταξινόμησης μπορεί να πακεταριστεί ως ένα δυαδικό αντικείμενο και να σταλεί στην αγορά των αντικειμένων συστατικών. Ο κατασκευαστής που χρειάζεται τον αλγόριθμο ταξινόμησης απλά χρησιμοποιεί οποιοδήποτε αντικείμενο ταξινόμησης οποιουδήποτε τύπου χρειάζεται χωρίς να ανησυχεί πως υλοποιήθηκε. Ο κατασκευαστής του αντικειμένου ταξινόμησης μπορεί να αποφύγει τις ενοχλήσεις και τα θέματα πνευματικής ιδιοκτησίας του κώδικα και να αφοσιωθεί στην παροχή της καλύτερης δυαδικής έκδοσης του αλγορίθμου ταξινόμησης.

Όπως έγινε με τους κατασκευαστές hardware και τα ολοκληρωμένα κυκλώματα, έτσι και οι κατασκευαστές εφαρμογών δεν χρειάζεται να ανησυχούν για το *πως* θα κατασκευάσουν αυτήν την συνάρτηση. Μπορούν απλά να την αγοράσουν. Η κατάσταση είναι παρόμοια με την αγορά των ολοκληρωμένων κυκλωμάτων σήμερα: δεν μπορεί κανείς να αγοράσει τον «κώδικα» για το IC και να κατασκευάσει από μόνος του το IC. Το μοντέλο επιτρέπει την αγορά του συστατικού λογισμικού, όπως γίνεται η αγορά ενός ολοκληρωμένου κυκλώματος. Το συστατικό (component) είναι συμβατό με οτιδήποτε κάποιος «προσκολλήσει» σε αυτό.

Ενεργοποιώντας την ανάπτυξη των συστατικών λογισμικού, το μοντέλο COM (Component Object Model) παρέχει έναν πολύ πιο παραγωγικό τρόπο για τον σχεδιασμό, την κατασκευή, την πώληση, την χρήση και την επαναχρησιμοποίηση του λογισμικού. Τα συστατικά λογισμικού έχουν σημαντικές επιπτώσεις για τους πωλητές λογισμικού, τους χρήστες και τις εταιρίες:

- **Οι κατασκευαστές εφαρμογών** μπορούν να κατασκευάζουν και να διανέμουν εφαρμογές πιο εύκολα από πριν. Τα συστατικά παρέχουν και κλιμάκωση και άρθρωση για την επαναχρησιμοποίηση του κώδικα. Ακόμα, οι κατασκευαστές μπορούν να επιτύχουν υψηλότερη απόδοση αφού μπορούν να μάθουν ένα σύστημα αντικειμένων για πολλές πλατφόρμες.
- **Οι πωλητές** έχουν ένα μοναδικό μοντέλο για την αλληλεπίδραση με άλλες εφαρμογές και το υπολογιστικό περιβάλλον. Ενώ τα συστατικά λογισμικού μπορούν να προστεθούν εύκολα σε ήδη υπάρχουσες εφαρμογές χωρίς να ξαναγραφτεί από τα

θεμέλια και από την αρχή ο κώδικας, προσφέρουν επίσης την ευκαιρία να γίνουν οι εφαρμογές αρθρωτές και να αντικαθιστώνται αποσπασματικά δυνατότητες του συστήματος όπου αυτό χρειάζεται.

- **Οι τελικοί χρήστες** θα έχουν μια πολύ μεγαλύτερη ποικιλία επιλογών λογισμικού, και ταυτόχρονα μεγαλύτερη αποδοτικότητα. Θα έχουν πρόσβαση σε εκατοντάδες αντικειμένων μέσα από πλατφόρμες client/server. Αυτά τα αντικείμενα θα έχουν από πριν κατασκευαστεί από ανεξάρτητους πωλητές λογισμικού και εταιρίες. Ακόμα, καθώς οι χρήστες θα βλέπουν τις δυνατότητες των συστατικών λογισμικού, είναι πολύ πιθανό να αυξηθεί η ζήτηση για εξειδικευμένα συστατικά που μπορούν να αγοραστούν σε τοπικά καταστήματα και να «προσκολληθούν» σε εφαρμογές.

2.3. Πλαίσια ανάπτυξης εφαρμογών

2.3.1. Γεννήτριες εφαρμογών

Η μία προσέγγιση στον ΟΔΥΣΣΕΑ θα μπορούσε να είναι αυτή της γεννήτριας εφαρμογών. Πρόκειται για ένα ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών με λειτουργικότητα που μπορεί να φτάσει σε μεγάλο βάθος. Τα τεράστια πλεονεκτήματα της προσέγγισης αυτής είναι η δραματική μείωση του χρόνου και της πολυπλοκότητας ανάπτυξης μιας εφαρμογής και η βεβαιότητα ότι όλες οι εφαρμογές που αναπτύσσονται με αυτόν τον τρόπο θα είναι συμβατές με οτιδήποτε επιλέξει ο δημιουργός της γεννήτριας εφαρμογών. Μια τέτοια εφαρμογή θα μπορούσε να αποτελεί ένα γραφικό περιβάλλον ανάπτυξης εφαρμογών με τη δυνατότητα να δημιουργεί κώδικα για πολλαπλές πλατφόρμες ή σε πολλές γλώσσες. Τα μειονεκτήματα είναι ότι οι ανεξάρτητοι κατασκευαστές περιορίζονται αρκετά στα προϊόντα που μπορούν να αναπτύξουν. Επίσης είναι κάπως παρακινδυνευμένο να αναγκαστούν να χρησιμοποιούν συγκεκριμένο εργαλείο ανάπτυξης των εφαρμογών τους μια και οι περισσότεροι θα έχουν ήδη έναν τρόπο ανάπτυξης με τα εργαλεία που τους βολεύουν. Εξάλλου ο χρόνος και οι αναγκαίοι πόροι για την κατασκευή μιας τέτοιας γεννήτριας εφαρμογών θα ήταν πολύ μεγαλύτεροι από τους διαθέσιμους στα πλαίσια του έργου ΑΙΝΕΙΑΣ.

Ως γεννήτρια εφαρμογών θα μπορούσαμε να θεωρήσουμε το σύστημα Comspec [16], [19]. Πρόκειται για ένα πρόγραμμα κατασκευής Βοηθημάτων Επικοινωνίας για Άτομα με Ειδικές Ανάγκες. Βασίζεται σε μια γενική σχεδίαση για το πώς πρέπει να είναι ένα Βοήθημα Επικοινωνίας και σε μια ποικιλία από συστατικά που μπορούν να το αποτελούν. Τα συστατικά που υποστηρίζει το Comspec είναι αυτά που το ίδιο το πρόγραμμα περιλαμβάνει, μια και είναι αρκετά δύσκολο για τρίτους κατασκευαστές να αναπτύξουν δικά τους συστατικά που μπορούν να ενσωματωθούν στο Comspec. Η φιλοσοφία της εφαρμογής είναι η χρησιμοποίηση έτοιμων απλών εικονικών πληκτρολογίων ως βάση για την ανάπτυξη πιο σύνθετων Βοηθημάτων προσθέτοντας στοιχεία που είναι διαθέσιμα σε μπάρες εργαλείων, όπως κουμπιά, πλαίσια κειμένου, συσκευές εισόδου/εξόδου κτλ. Το πρόγραμμα βασίζεται σε 16μπιτες εκδόσεις του λειτουργικού Windows και η διεπαφή χρήσης του είναι η τυπική μιας εφαρμογής που τρέχει σε αυτό το λειτουργικό με τα παράθυρα, τα πλαίσια διαλόγου, τα κουμπιά τα πλαίσια κειμένου και όλα τα τυπικά στοιχεία ελέγχου εφαρμογών. Το Comspec αν και είναι πολύ εύκολο στη χρήση και έχει πολλές δυνατότητες παραμετροποίησης των Βοηθημάτων που κατασκευάζονται με αυτό, χαρακτηρίζεται από το μειονέκτημα που αναφέραμε και αφορά στον περιορισμό των χρηστών του ως προς την ποικιλία και τον πλήρη έλεγχο στα Βοηθήματα που μπορούν να κατασκευάσουν.

Φυσικά θα μπορούσε κανείς να κατασκευάσει σχεδόν οποιασδήποτε μορφής Βοήθημα Επικοινωνίας χρησιμοποιώντας μια άλλη εφαρμογή που μπορεί επίσης να θεωρηθεί ως γεννήτρια εφαρμογών. Η Microsoft Visual Basic [33] συχνά αναφέρεται ως εργαλείο προγραμματισμού και γεννήτρια εφαρμογών. Δεν πρόκειται για μια τυπική γλώσσα προγραμματισμού, αλλά βρίσκεται σε ένα πολύ υψηλότερο επίπεδο, ώστε να μπορούμε να την κατατάξουμε στην κατηγορία των πακέτων λογισμικού που μας απασχολεί εδώ. Είναι φανερό ότι με ένα τόσο ισχυρό εργαλείο θα μπορούσε κανείς να επιτύχει οποιαδήποτε χαρακτηριστικά και προδιαγραφές έβαζε ως στόχο για ένα Βοήθημα Επικοινωνίας, αλλά είναι επίσης φανερό πως δεν είναι εφικτό να κινηθούμε προς μια τέτοια κατεύθυνση στα πλαίσια αυτού του έργου μια και έστω και μια απλοποιημένη μορφή, εστιασμένη στα Βοηθήματα Επικοινωνίας ενός τέτοιου πακέτου λογισμικού απαιτεί χρόνια συντονισμένης δουλειά εκατοντάδων ανθρώπων για την υλοποίησή της.

2.3.2. Τυποποιήσεις, οδηγίες και τεχνικές προγραμματισμού

Μια δεύτερη προσέγγιση είναι η υιοθέτηση συγκεκριμένων τεχνικών αντικειμενοστραφούς και επαναχρησιμοποιούμενου προγραμματισμού και η πρόταση ειδικών προδιαγραφών, τυποποιήσεων και οδηγιών για τη δημιουργία προϊόντων συμβατών μεταξύ τους, ανεξάρτητα από τις γλώσσες προγραμματισμού, τα εργαλεία και τις πλατφόρμες που χρησιμοποιούνται για την ανάπτυξή τους. Αυτή η προσέγγιση απαιτεί την προσεκτική μελέτη όλων των διαθέσιμων τυποποιήσεων και τεχνολογιών για την ανάπτυξη εφαρμογών και την απόφαση για το ποιες από αυτές θα υποστηριχθούν. Επίσης προβλέπεται να αναπτυχθούν νέες τυποποιήσεις και οδηγίες σε τομείς που δεν καλύπτονται από τις ήδη υπάρχουσες. Το αποτέλεσμα είναι ένα θεωρητικό περιβάλλον ανάπτυξης και όχι μια γεννήτρια εφαρμογών υλοποιημένη με λογισμικό. Οι ανεξάρτητοι προγραμματιστές θα έχουν την ελευθερία να χρησιμοποιήσουν τα εργαλεία ανάπτυξης εφαρμογών που προτιμούν, αρκεί οι εφαρμογές που δημιουργούνται να πληρούν τις προδιαγραφές του πλαισίου και η ανάπτυξη να συμμορφώνεται με τις αυστηρές οδηγίες που αναφέρονται στη συμβατότητα και στην ενσωμάτωση των εφαρμογών σε ένα ολοκληρωμένο βοήθημα επικοινωνίας. Έτσι ο κώδικας που αναπτύσσεται για τη δημιουργία ενός τμήματος μιας εφαρμογής θα μπορεί να ξαναχρησιμοποιείται και σε άλλες εφαρμογές και δεν θα απαιτείται οι ίδιες λειτουργικότητες να ξανασχεδιάζονται και να αναπτύσσονται από την αρχή για κάθε νέα ολοκληρωμένη εφαρμογή που τις χρειάζεται.

Μια τέτοια προσέγγιση υλοποιήθηκε στα πλαίσια του έργου ACCESS [4], [5], [6],, [7], [8], [9], [10], [11] και της αρχιτεκτονικής ATIC που προέκυψε από αυτό. Το πλαίσιο βασιζόταν στην υιοθέτηση τεχνικών προγραμματισμού βασισμένων στα συστατικά και αντικειμενοστραφών. Περιελάμβανε ένα αυτοσχέδιο (proprietary) πρωτόκολλο επικοινωνίας μεταξύ των συστατικών, έναν επίσης αυτοσχέδιο Διαχειριστή μηνυμάτων που λειτουργούσε ως μεσάζοντας και συντονιστής της επικοινωνίας αυτής και ένα σύνολο ρουτινών και βιβλιοθηκών για χρήση από τους προγραμματιστές και διευκόλυνση της διαδικασίας υλοποίησης Βοηθημάτων επικοινωνίας. Επίσης αναπτύχθηκαν στα πλαίσια εκείνου του έργου και κάποια εργαλεία για υποβοήθηση της αποσφαλμάτωσης και της συντήρησης συστατικών που ήταν συμβατά με την αρχιτεκτονική ATIC. Τέλος αυτή η αρχιτεκτονική βασιζόταν στην υποδομή των 16μπιτων εκδόσεων του λειτουργικού συστήματος των Windows και στις υπηρεσίες Μηνυμάτων Παραθύρων που προσέφεραν. Είναι αυτή η φιλοσοφία πλαισίου ανάπτυξης εφαρμογών που θα ακολουθηθεί και σε αυτό το έργο. Βέβαια στόχος εδώ είναι η υποστήριξη πιο νέων τεχνολογιών, νέων λειτουργικών συστημάτων, αποφυγή μεγάλου όγκου αυτοσχέδιων τυποποιήσεων και υλοποιήσεων και μεγαλύτερη διευκόλυνση των κατασκευαστών από την άποψη του όγκου των

πληροφοριών που θα πρέπει να γνωρίζουν και να χρησιμοποιούν για να κατασκευάζουν συστατικά συμβατά με το πλαίσιο. Επίσης ο ΟΔΥΣΣΕΑΣ θα προτείνει έναν λίγο διαφοροποιημένο κύκλο ζωής των προϊόντων από αυτόν που πρότεινε η αρχιτεκτονική ATIC και θα έχει τις υπηρεσίες του λειτουργικού συστήματος στη θέση του αυτοσχέδιου Διαχειριστή Μηνυμάτων που ήταν και η καρδιά της αρχιτεκτονικής αυτής.

2.4. Γενικά Χαρακτηριστικά του Πλαισίου Ανάπτυξης Εφαρμογών "ΟΔΥΣΣΕΑΣ"

- **Διαλειτουργικότητα**

Οι επιμέρους λύσεις που μπορούν να συμμορφωθούν με το πλαίσιο ΟΔΥΣΣΕΑΣ θα πρέπει να συνεργάζονται μεταξύ τους ώστε να προκύπτουν μεγαλύτερα και πιο ολοκληρωμένα βοηθήματα επικοινωνίας που να χρησιμοποιούν τμήματα ανεπτυγμένα ακόμη και από διαφορετικούς κατασκευαστές.

- **Αρθρωτή σχεδίαση**

Η φιλοσοφία ανάπτυξης εφαρμογών για ΑΜΕΑ θα πρέπει να βασίζεται στην κατάτμησή τους σε μικρότερα κομμάτια που θα μπορούν ευκολότερα να τροποποιηθούν και να συντηρηθούν χωρίς να απαιτείται σε κάθε αλλαγή των απαιτήσεων από την αρχή ανάπτυξη όλης της εφαρμογής. Τα κομμάτια αυτά θα πρέπει να έχουν όσο το δυνατό μεγαλύτερη αυτονομία ώστε να προσφέρουν τον μεγαλύτερο βαθμό επαναχρησιμοποίηση και να οδηγούν στο επόμενο επιθυμητό χαρακτηριστικό.

- **Με βάση τα συστατικά (component based)**

Η επέκταση και το αποτέλεσμα της αρθρωτής σχεδίασης μιας εφαρμογής οδηγεί στον προγραμματισμό με βάση τα συστατικά (component based), στα πλαίσια του οποίου αναπτύσσονται τμήματα της εφαρμογής τα οποία είναι εντελώς αυτόνομα, μπορούν να χρησιμοποιηθούν και από άλλες εφαρμογές και το κάθε ένα από αυτά συμμορφώνεται ξεχωριστά με τις επιθυμητές προδιαγραφές.

- **Επεκτασιμότητα**

Ως αποτέλεσμα των παραπάνω ικανοποιείται και η απαίτηση της επεκτασιμότητας των επιμέρους προϊόντων με βάση την ανεξαρτησία τους από το ολοκληρωμένο βοήθημα επικοινωνίας και της δυνατότητας πλήρους αντικατάστασής τους με άλλα παρεμφερή. Έτσι το τελικό προϊόν θα μπορεί να είναι πολύ ευκίνητο και παραμετροποιήσιμο, ανάλογα με τα διαθέσιμα συστατικά και τις απαιτήσεις των χρηστών. Πάνω από όλα το βοήθημα επικοινωνίας θα μπορεί να δέχεται τροποποιήσεις και επεκτάσεις και να συμβαδίζει ακόμα και με μεταβαλλόμενες απαιτήσεις με το μικρότερο δυνατό κόστος σε πολυπλοκότητα και χρόνο για την επέκτασή του.

- **Ανεξαρτησία από γλώσσα προγραμματισμού**

Αυτό το χαρακτηριστικό μπορεί να υποστηριχθεί μέχρι κάποιο βαθμό από το πλαίσιο ΟΔΥΣΣΕΑΣ και με βάση τα προηγούμενα χαρακτηριστικά, αρκεί η επιλεγμένη από κάθε προγραμματιστή γλώσσα να υποστηρίζει τις τεχνολογίες που θα απαιτούνται για τη λειτουργία μιας εφαρμογής συμβατής με το πλαίσιο αυτό. Στόχος είναι να επιλεχθούν οι τεχνολογίες αυτές που υποστηρίζονται στο παρόν από τις περισσότερες γλώσσες προγραμματισμού έχοντας βέβαια υπ' όψη και την παράμετρο της μελλοντικής επιτυχίας και επικράτησης των τεχνολογιών αυτών στο βαθμό που κάτι τέτοιο είναι δυνατόν να προβλεφθεί.

2.5. Μεθοδολογία Ανάπτυξης του ΟΔΥΣΣΕΑ

Μετά τη μελέτη των αναγκών των χρηστών [2], [3] που έγινε στα πλαίσια του έργου ΑΙΝΕΙΑΣ, μπορούμε να κάνουμε μια γενική περιγραφή του προκαταρκτικού σχεδιασμού του Πλαισίου Σχεδιασμού και Ανάπτυξης Βοηθημάτων Διαπροσωπικής Επικοινωνίας ΟΔΥΣΣΕΑΣ εστιάζοντας σε έξι άξονες:

- Τυποποιήσεις – Οδηγίες
- Λογισμικό υποβοήθησης υλοποίησης συστατικών
- Λογισμικό ελέγχου συμβατότητας συστατικών
- Λογισμικό σύνδεσης συστατικών
- Ολοκλήρωση – Διεπαφή χρήσης
- Υποστήριξη Διαδικτύου

Αυτοί οι έξι άξονες αντιπροσωπεύουν μια κατηγοριοποίηση των χαρακτηριστικών του πλαισίου ΟΔΥΣΣΕΑ και της υλοποίησής του ή έξι διαφορετικές γωνίες από τις οποίες πρέπει να αντιμετωπιστεί το πλαίσιο και αναλύονται στη συνέχεια. Οι λεπτομέρειες του σχεδιασμού παρουσιάζονται στο έβδομο κεφάλαιο, αφού αναλυθούν οι τεχνολογίες και η υποδομή που πρόκειται να χρησιμοποιηθεί.

2.5.1. Τυποποιήσεις – Οδηγίες

Όσον αφορά τον ΟΔΥΣΣΕΑ, η ανάπτυξη θα γίνει σε πλατφόρμα Microsoft Windows 2000. Θα χρησιμοποιηθούν εργαλεία ανάπτυξης εφαρμογών (τοπικών εφαρμογών desktop και εφαρμογών για το Διαδίκτυο) της Microsoft όπως: Microsoft Visual C++ 6.0, Microsoft Visual Basic 6.0, Microsoft Visual Modeler, Microsoft SQL Server, Microsoft Internet Information Server. Οι τεχνολογίες που θα χρησιμοποιηθούν περιλαμβάνουν τον προγραμματισμό με βάση τα συστατικά που προτείνει και χρησιμοποιεί η Microsoft όπως Microsoft Windows DNA-based Applications on Windows 2000, COM+, Queued Components, System Events, Data Access Components, ADO, OLE-DB, Active Server Pages, Dynamic HTML, XML. Είναι δυνατόν να χρησιμοποιηθούν και τεχνολογίες της Sun Microsystems, όπως Java, Java Beans, Java Applets.

Όσον αφορά τα συστατικά του ΟΔΥΣΣΕΑ, δεν καθορίζεται συγκεκριμένη γλώσσα ή πλατφόρμα ανάπτυξης, αλλά ο βασικός όρος είναι να είναι συμβατά με τα Windows 2000. Επίσης, όλες οι τεχνολογίες που αναφέρθηκαν παραπάνω για την ανάπτυξη του ΟΔΥΣΣΕΑ θα μπορούν να χρησιμοποιηθούν και να υποστηριχθούν και από τα συστατικά του. Φυσικά θα υπάρχουν αυστηρές οδηγίες και προδιαγραφές για την κατασκευή κάθε συστατικού ξεχωριστά. Βασικός στόχος είναι τα συστατικά να συνεργάζονται μεταξύ τους.

2.5.2. Λογισμικό υποβοήθησης υλοποίησης συστατικών

Θα αποφευχθεί η φιλοσοφία της γεννήτριας εφαρμογών ή του υλοποιημένου σε λογισμικό περιβάλλοντος ανάπτυξης. Δεν θα δημιουργείται από αυτό το επίπεδο κώδικας διαφορετικός από τον πηγαίο κώδικα κάθε συστατικό, αλλά θα προτιμηθεί η προσέγγιση της εφαρμογής που βοηθά τον κατασκευαστή να βρει τις σωστές οδηγίες (guidelines) και προδιαγραφές (specifications) που χρειάζεται για την κατασκευή ή την προσαρμογή του συστατικού του στα πλαίσια του ΟΔΥΣΣΕΑ. Θα χρησιμοποιηθούν τεχνολογίες υπερκειμένου (hypertext) και πρόσβαση σε βάσεις δεδομένων. Φυσικά θα απαιτηθούν κομμάτια λογισμικού που θα μπορούν να χρησιμοποιηθούν σε οποιοδήποτε συστατικό

κατασκευάζεται ώστε να διευκολύνεται ή να αυτοματοποιείται η διαδικασία ενσωμάτωσής τους στο ολοκληρωμένο Βοήθημα Επικοινωνίας.

2.5.3. Λογισμικό ελέγχου συμβατότητας συστατικών

Αν ακολουθηθεί η ιδέα να υποβάλλουν οι κατασκευαστές τα συστατικά τους σε έναν δικτυακό τόπο όπου θα είναι διαθέσιμα, είτε αυτούσια, είτε ως σύνδεσμοι, θα πρέπει να γίνεται κάποιος έλεγχος σε αυτά τα συστατικά για το αν έχουν συμμορφωθεί με τις οδηγίες και τις προδιαγραφές που τους αντιστοιχούν. Βέβαια όποιος και να είναι ο τρόπος δημοσιοποίησης ή καταχώρησης των συμβατών με τον ΟΔΥΣΣΕΑ συστατικών, θα πρέπει να γίνεται ο ίδιος έλεγχος προσεκτικά, ώστε να μην υπάρχουν προβλήματα στη διασύνδεση και λειτουργία των συνδυασμών των συστατικών. Αυτοί οι έλεγχοι μπορούν να διεξάγονται από ειδικευμένο προσωπικό, βάσει αυστηρών μεθοδολογιών και πρωτοκόλλων, ή, στην καλύτερη περίπτωση να είναι αυτόματοι και να γίνονται από ένα ειδικό software engine. Η δεύτερη βέβαια προσέγγιση είναι αρκετά δύσκολη και δεν είναι εφικτή στα πλαίσια αυτού του έργου. Επίσης δεν καθορίζεται ακόμα σε ποιο βάθος θα προχωρούν αυτοί οι έλεγχοι. Είναι άμεσα συνδεδεμένη η διαδικασία του ελέγχου με την διαδικασία τυποποίησης και επιβολής οδηγιών. Όσο πιο αυστηρή είναι η δεύτερη τόσο πιο εύκολη θα είναι η πρώτη. Έτσι η διαδικασία ελέγχου θα είναι σχετικά ελεύθερη με τους προγραμματιστές να έχουν τη δυνατότητα να ελέγχουν τα συστατικά τους με τις δυνατότητες ελέγχου που θα τους προσφέρει το ίδιο το λειτουργικό σύστημα, αλλά και με ειδικά προγράμματα ελέγχου που θα διατίθενται από το Διαδικτυακό τόπο του ΟΔΥΣΣΕΑ.

2.5.4. Λογισμικό σύνδεσης συστατικών

Αυτό το σημείο είναι και το πιο σημαντικό και ίσως πολύπλοκο για το πλαίσιο ΟΔΥΣΣΕΑΣ. Η επιθυμητή κατάσταση βέβαια θα ήταν όλα τα έτοιμα συστατικά να μπορούν να διασυνδεθούν άμεσα, εφόσον βέβαια οι συγκεκριμένες συνδέσεις των συγκεκριμένων συστατικών προβλέπονται από τον ΟΔΥΣΣΕΑ. Το επιθυμητό είναι να μην απλά και αυτόματα η διαδικασία χωρίς καμία παρέμβαση.

Μια λύση θα μπορούσε να είναι η κατασκευή «πρακτόρων» (agents) που θα διασυνδέουν τα συστατικά σε επίπεδο κώδικα ή σε υψηλότερο επίπεδο χρησιμοποιώντας και το registry του λειτουργικού. Παρεμφερής λύση είναι η κατασκευή message centers που θα αποκαθιστούν της επικοινωνία των συστατικών μέσω ειδικών πρωτοκόλλων και οδών που θα καθορίζονται από τον ΟΔΥΣΣΕΑ και θα χρησιμοποιούν ειδικά αρχεία ρυθμίσεων για τα συστατικά, φτιαγμένα ειδικά για τις προδιαγραφές του ΟΔΥΣΣΕΑ.

Η λύση που θα έλυνε περισσότερα προβλήματα σε λιγότερο χρόνο και με μικρότερη πολυπλοκότητα φαίνεται όμως να είναι αυτή της χρήσης των Windows APIs ή του των υπηρεσιών του λειτουργικού συστήματος για την επικοινωνία και σύνδεση των συστατικών. Μία εφαρμογή που θα μπορούσε να εκμεταλλευτεί έξυπνα τις κατάλληλες υπηρεσίες θα μπορούσε χωρίς να παρεμβαίνει στη λειτουργία του βοηθήματος και χωρίς να περιορίζει ους προγραμματιστές των συστατικών να ενεργοποιεί και να συντονίζει ομαλά τα συστατικά και τις λειτουργίες τους.

2.5.5. Ολοκλήρωση – user interface

Όσον αφορά τη Διεπαφή Χρήστη του ΟΔΥΣΣΕΑ, αυτή μπορεί να υλοποιηθεί σε πολλά επίπεδα, ανάλογα με τη λειτουργικότητα που θα έχει τελικά το πλαίσιο. Αν επιλεγεί το πλαίσιο να είναι προσβάσιμο από το Διαδίκτυο, η διεπαφή χρήστη θα είναι ο browser. Ο browser μπορεί να χρησιμοποιηθεί ως Διεπαφή Χρήστη ακόμη και στην περίπτωση που

δεν μεσολαβεί το Διαδίκτυο στην αλληλεπίδραση με το πλαίσιο ΟΔΥΣΣΕΑΣ. Οι σελίδες HTML σε όλες τις μορφές τους (από την απλή text HTML έως και τις Active Server Pages) είναι ένα βολικό User Interface από την άποψη ότι είναι ανεξάρτητο πλατφόρμας, εύκολα μεταφέρσιμο, ομοιογενές και συνεπές από τη φύση του και υποστηρίζει τοπική ή δικτυακή αλληλεπίδραση.

Η τελική μορφή του Οδυσσέα μπορεί να περιλαμβάνει στην ιδεατή περίπτωση όλα τα παραπάνω, αλλά ρεαλιστικά, μόνο τμήματα του παραπάνω σχεδιασμού θα μπορούσαν να υλοποιηθούν. Στόχος είναι τα λειτουργικά επίπεδα του πλαισίου ΟΔΥΣΣΕΑΣ να λειτουργούν σωστά και να συνεργάζονται αρμονικά μεταξύ τους ώστε και να έχουμε τα ίδια αποτελέσματα στη σύνδεση και λειτουργία των υποστηριζόμενων συστατικών. Στην ιδανική περίπτωση η διεπαφή χρήσης του ΟΔΥΣΣΕΑ θα πρέπει να είναι ανύπαρκτη! Δεν είναι για κάποιο λόγο αναγκαία η πολυπλοκότητα που θα επέβαλε μια ακόμα διεπαφή χρήστη. Ένα έξυπνα και σωστά σχεδιασμένο σύστημα θα μπορούσε να λειτουργεί στο παρασκήνιο χωρίς να αποσπά την προσοχή ούτε του προγραμματιστή ούτε του τελικού βέβαια χρήστη. Ένα λογισμικό τόσο συμπαγές και διάφανο σαν να αποτελεί και αυτό μια ακόμα υπηρεσία του λειτουργικού συστήματος που ο καθένας μπορεί να χρησιμοποιήσει χωρίς να χρειάζεται να μάθει κάτι περισσότερο είναι η προσέγγιση που τελικά ακολουθείται.

2.5.6. Υποστήριξη Διαδικτύου

Η υποστήριξη του Διαδικτύου από το πλαίσιο ΟΔΥΣΣΕΑΣ είναι μία από τις απαιτήσεις μας. Το θέμα εδώ είναι έως ποιο βάθος θα υποστηρίζεται η λειτουργία του ΟΔΥΣΣΕΑ μέσα από το Internet.

Από τη μια πλευρά θα μπορούσε ο ΟΔΥΣΣΕΑΣ να είναι μια καθαρά κατανεμημένη και προσβάσιμη μόνο από το Διαδίκτυο οντότητα, που θα περιλαμβάνει βάσεις δεδομένων με πληροφορίες για τυποποιήσεις και guidelines, με όλα τα διαθέσιμα συστατικά και τις προδιαγραφές τους, με Ιδρύματα και Χρήστες βοηθημάτων επικοινωνίας. Σε αυτόν θα μπορούσαν επίσης να ενσωματωθούν εγγενή συστατικά όπως βάσεις δεδομένων γλωσσών, συμβόλων, γραμματοσειρών, εικόνων πολυμέσων, εκπαιδευτικού υλικού ή υλικού αναφοράς, λεξικών κτλ. Η λειτουργικότητά του θα μπορούσε να περιλαμβάνει την υποστήριξη τεχνικών υπερκειμένου, αναζήτησης πληροφοριών και δεδομένων, πλοήγηση μέσω Internet στον κόσμο των βοηθημάτων επικοινωνίας, σχεσιακούς δεσμούς μεταξύ όλων των στοιχείων και δεδομένων του, διαδικασίες πολύπλοκων ερωτημάτων και απλών ανακτήσεων δεδομένων από τις σχεσιακές βάσεις, μηχανισμούς σύνδεσης συστατικών και σύνθεσης communicators έστω και σε επίπεδο συστάσεων και προτάσεων. Η λειτουργικότητα ενός τέτοιου μοντέλου περιορίζεται μόνο από τις δυνατότητες των εργαλείων προγραμματισμού και των υπολογιστών.

Από την άλλη πλευρά στα πλαίσια αυτού του έργου θα πρέπει να γίνουν κάποια trade-offs στα πλαίσια του διαθέσιμου χρόνου και πόρων, ώστε ο ΟΔΥΣΣΕΑΣ να προσφέρει όσο το δυνατό περισσότερα με όσο το δυνατό λιγότερη πολυπλοκότητα και προγραμματισμό. Έτσι η Υποστήριξη του Διαδικτύου θα μπορούσε στην πιο ρεαλιστική περίπτωση να περιλαμβάνει απλά τη διαθεσιμότητα των προδιαγραφών των τυποποιήσεων και των οδηγιών του ΟΔΥΣΣΕΑ προς τους κατασκευαστές των συστατικών μέσα από το Internet σε μορφή υπερκειμένου.

Τέλος είναι πολύ σημαντικό να σχεδιαστεί έτσι το πλαίσιο ώστε η υποστήριξη του Διαδικτύου από τις λειτουργίες του και τις δυνατότητές του να είναι εγγενής. Δεν πρόκειται να υιοθετηθεί η προσέγγιση της επέκτασης του πλαισίου για υποστήριξη του

Διαδικτύου, αλλά ο εξαρχής σχεδιασμός του ώστε να περιλαμβάνει όλες τις απαραίτητες τυποποιήσεις και χαρακτηριστικά ώστε η υποστήριξη του Διαδικτύου να είναι ενσωματωμένη. Για παράδειγμα ο τρόπος και η διαδικασία ενσωμάτωσης στο Βοήθημα Επικοινωνίας ενός συστατικού που θα στέλνει e-mails μέσω του Διαδικτύου δεν θα πρέπει να διαφέρει δραματικά από την εγκατάσταση ενός συστατικού που λειτουργεί αποκλειστικά σε τοπικό επίπεδο. Ο ΟΔΥΣΣΕΑΣ είναι που πρέπει να το φροντίσει αυτό.

Στη συνέχεια θα αναλυθούν κάποιες τεχνολογίες που μελετήθηκαν και επιλέχθηκαν ώστε να αποτελέσουν την υποδομή του πλαισίου ΟΔΥΣΣΕΑ. Στο κεφάλαιο 7 θα δοθούν περισσότερες λεπτομέρειες για τον τελικό σχεδιασμό του πλαισίου αφού αναφερθούν όλες οι τεχνικές υποδομές που χρησιμοποιούνται και αποτελούν τη βάση του πλαισίου.

3. Επικοινωνία μεταξύ Διεργασιών

Ένα πολύ σημαντικό θέμα που πρέπει να λύσει ο ΟΔΥΣΣΕΑΣ είναι αυτό της επικοινωνίας μεταξύ των συστατικών. Κάτι που στο παρελθόν έγινε από το έργο ACCESS [4], [5], [6], [7], [8], [9], [10], [11] με την αρχιτεκτονική και τα πρωτόκολλα επικοινωνίας του ATIC πρέπει τώρα να βελτιωθεί και να προσαρμοστεί στις τεχνολογίες αιχμής στο χώρο της πληροφορικής. Όπως ήδη αναφέρθηκε, το αδύνατο σημείο του ATIC ήταν ότι απαιτείτο η παρουσία ενός αυτοσχέδιου Διαχειριστή Μηνυμάτων αλλά και ενός αυτοσχέδιου πρωτοκόλλου επικοινωνίας μεταξύ των συστατικών, δύο στοιχεία που επέβαλλαν σοβαρό overhead στην εφαρμογή που θα βασιζόταν σε αυτά, και σοβαρούς περιορισμούς στους υποψήφιους κατασκευαστές συστατικών για Βοηθήματα Διαπροσωπικής Επικοινωνίας.

Το λεπτό σημείο του σχεδιασμού είναι ότι πρέπει να υποστηρίζεται και να υλοποιείται η επικοινωνία μεταξύ συστατικών τα οποία δεν γνωρίζουν τίποτα το ένα για το άλλο και φυσικά δεν έχουν το στη διάθεσή τους προγραμματιστικούς δείκτες ή αναφορές το ένα για το άλλο. Θα πρέπει λοιπό να παρεμβληθεί κάποια οντότητα που να συντονίζει και να διαχειρίζεται την κυκλοφορία των δεδομένων μεταξύ των συστατικών και για να αποφευχθεί η λύση του αυτοσχέδιου Διαχειριστή Μηνυμάτων θα πρέπει να γίνει σωστή και αποτελεσματική εκμετάλλευση των υπηρεσιών του συστήματος που μπορούν να καλύψουν αυτές τις ανάγκες.

Το Microsoft® Win32® API παρέχει μηχανισμούς για τη διευκόλυνση της επικοινωνίας και του διαμοιρασμού δεδομένων μεταξύ των εφαρμογών. Συνολικά, οι δραστηριότητες που ενεργοποιούνται από αυτούς τους μηχανισμούς καλούνται *επικοινωνίες μεταξύ διεργασιών (interprocess communications – IPC)* [57]. Συνήθως, οι εφαρμογές μπορούν να χρησιμοποιήσουν IPCs κατηγοριοποιημένες σε πελάτες (clients) και εξυπηρέτες (servers). *Πελάτης* είναι μια εφαρμογή ή μια διεργασία η οποία ζητάει μια υπηρεσία από κάποια άλλη εφαρμογή ή διεργασία. *Εξυπηρέτης* είναι μια εφαρμογή η διεργασία που απαντά σε μία αίτηση του πελάτη.

Από το Win32 API υποστηρίζονται οι παρακάτω μηχανισμοί IPC:

- Clipboard
- OLE/COM
- DDE
- File Mapping
- Mailslots
- Pipes
- RPC
- Windows Sockets
- WM_COPYDATA

Όλοι αυτοί οι μηχανισμοί [55], [57], περιγράφονται με συντομία στη συνέχεια.

3.1. Clipboard

Το Clipboard παίζει το ρόλο μιας κεντρικής προσωρινής αποθήκης για το διαμοιρασμό δεδομένων μεταξύ εφαρμογών [55]. Όταν ένας χρήστης πραγματοποιεί μια αποκοπή ή αντιγραφή (cut or copy) σε μια εφαρμογή, η εφαρμογή βάζει τα επιλεγμένα δεδομένα στο clipboard σε μία ή περισσότερες τυπικές ή προσαρμοσμένες στην εφαρμογή μορφές (formats). Οποιαδήποτε άλλη εφαρμογή μπορεί να ανακτήσει αυτά τα δεδομένα από το clipboard, επιλέγοντας από τις διαθέσιμες μορφές που καταλαβαίνει. Το Clipboard είναι ένα πολύ χαλαρά συνδεδεμένο (loosely coupled) μέσο συναλλαγής, όπου οι εφαρμογές χρειάζεται μόνο να συμφωνούν στον τύπο των δεδομένων. Οι εφαρμογές μπορούν να βρίσκονται στον ίδιο υπολογιστή ή σε διαφορετικούς υπολογιστές σε ένα δίκτυο.

3.2. COM

Οι εφαρμογές που χρησιμοποιούν OLE διαχειρίζονται *μικτά έγγραφα (compound documents)* [55] δηλαδή, έγγραφα που αποτελούνται από δεδομένα από διαφορετικές εφαρμογές. Το OLE έχει υπηρεσίες που διευκολύνουν τις εφαρμογές να καλέσουν άλλες εφαρμογές για επεξεργασία δεδομένων. Για παράδειγμα, ένας επεξεργαστής κειμένου που χρησιμοποιεί OLE θα μπορούσε να ενσωματώσει ένα γράφημα από μια εφαρμογή λογιστικού φύλλου. Ο χρήστης θα μπορούσε να ξεκινήσει το λογιστικό φύλλο αυτόματα μέσα από τον επεξεργαστή κειμένου επιλέγοντας το ενσωματωμένο γράφημα για επεξεργασία. Το OLE αναλαμβάνει την εκκίνηση του λογιστικού φύλλου και την παρουσίαση του γραφήματος για επεξεργασία. Όταν ο χρήστης κλείσει το λογιστικό φύλλο, το γράφημα θα ενημερωθεί στο αρχικό έγγραφο του επεξεργαστή κειμένου. Το λογιστικό φύλλο φαίνεται σαν να είναι μια επέκταση του επεξεργαστή κειμένου.

Τα θεμέλια του OLE είναι το Component Object Model (COM) [36]. Ένα συστατικό λογισμικού που χρησιμοποιεί COM μπορεί να επικοινωνήσει με διάφορα άλλα συστατικά, ακόμα και με συστατικά που δεν έχουν ακόμη γραφτεί. Τα συστατικά αλληλεπιδρούν ως αντικείμενα (objects) και πελάτες (clients). Το κατανεμημένο (Distributed) COM επεκτείνει το προγραμματιστικό μοντέλο COM έτσι ώστε να λειτουργεί σε δίκτυο. Το COM παρέχει πρόσβαση στα δεδομένα ενός αντικειμένου μέσω ενός ή περισσοτέρων συνόλων σχετικών λειτουργιών (functions), που είναι γνωστά ως *διεπαφές (interfaces).*

3.3. Dynamic Data Exchange (DDE)

Το DDE [54], [55] είναι ένα πρωτόκολλο το οποίο επιτρέπει στις εφαρμογές να ανταλλάσσουν δεδομένα σε διάφορα formats. Οι εφαρμογές μπορούν να χρησιμοποιήσουν το DDE για ανταλλαγές δεδομένων που γίνονται μία φορά ή για συνεχόμενες συναλλαγές κατά τις οποίες οι εφαρμογές ενημερώνονται μεταξύ τους καθώς γίνονται διαθέσιμα τα νέα δεδομένα.

Τα δεδομένα που χρησιμοποιούνται στο DDE είναι τα ίδια με αυτά που χρησιμοποιούνται στο clipboard. Το DDE μπορεί να θεωρηθεί ως μια προέκταση του μηχανισμού του clipboard. Το clipboard χρησιμοποιείται σχεδόν πάντα για μία (φορά) απόκριση σε μια εντολή του χρήστη, όπως στην επιλογή της εντολής επικόλλησης (paste) από έναν κατάλογο επιλογών. Το DDE συνήθως ξεκινά με μια εντολή του χρήστη, αλλά συχνά συνεχίζει να λειτουργεί χωρίς περαιτέρω αλληλεπίδραση με το χρήστη. Μπορούν επίσης να καθοριστούν προσαρμοσμένα formats δεδομένων DDE για ειδικού τύπου IPCs μεταξύ εφαρμογών με πιο σφικτά συνδεδεμένες (tightly coupled) απαιτήσεις επικοινωνίας.

Οι συναλλαγές DDE μπορούν να γίνουν μεταξύ εφαρμογών που τρέχουν στον ίδιο υπολογιστή ή σε διαφορετικούς υπολογιστές σε δίκτυο. Το DDE δεν είναι τόσο αποδοτικό όσο οι νεώτερες τεχνολογίες. Πάντως μπορεί να χρησιμοποιηθεί αν δεν είναι κατάλληλοι οι υπόλοιποι μηχανισμοί IPC ή αν πρέπει να υλοποιηθεί μια διεπαφή με μια εφαρμογή που υποστηρίζει μόνο DDE.

3.4. File Mapping

Το *file mapping* επιτρέπει σε μια διεργασία να χειριστεί τα περιεχόμενα ενός αρχείου σαν να ήταν ένα block μνήμης στο χώρο διευθύνσεων της διεργασίας. Αυτή η διεργασία μπορεί να χρησιμοποιεί απλούς χειρισμούς με δείκτες για να εξετάζει και να τροποποιεί τα περιεχόμενα ενός αρχείου. Αν δύο η περισσότερες διεργασίες προσπελαύνουν το ίδιο file mapping, κάθε διεργασία λαμβάνει έναν δείκτη στη μνήμη στο δικό της χώρο διευθύνσεων τον οποίο μπορεί να χρησιμοποιήσει για να διαβάσει ή να τροποποιήσει τα περιεχόμενα του αρχείου. Οι διεργασίες πρέπει να χρησιμοποιούν ένα αντικείμενο συγχρονισμού, όπως ένας σημαφόρος (semaphore), για να αποτραπεί η καταστροφή των δεδομένων σε ένα πολυδιεργασιακό (multitasking) περιβάλλον.

Το file mapping είναι αρκετά αποδοτικό και επίσης προσφέρει χαρακτηριστικά ασφαλείας υποστηριζόμενα από το λειτουργικό σύστημα που μπορούν να αποτρέψουν την καταστροφή των δεδομένων από κάποιον που δεν έχει δικαίωμα πρόσβασης. Το file mapping μπορεί να χρησιμοποιηθεί μόνο μεταξύ διεργασιών που τρέχουν στον ίδιο υπολογιστή. Βέβαια έχει και το μειονέκτημα ότι ο προγραμματιστής πρέπει να ασχοληθεί με το συγχρονισμό μεταξύ των διεργασιών.

3.5. Mailslots

Τα mailslots παρέχουν μονόδρομη επικοινωνία. Κάθε διεργασία που δημιουργεί ένα mailslot είναι ένας *εξυπηρέτης mailslot*. Άλλες διεργασίες, που καλούνται *πελάτες mailslot*, στέλνουν μηνύματα στον εξυπηρέτη mailslot γράφοντάς τα στο mailslot του. Τα εισερχόμενα μηνύματα προστίθενται πάντα στο mailslot όπου αποθηκεύονται έως ότου ο εξυπηρέτης τα διαβάσει. Μια διαδικασία μπορεί να είναι ταυτόχρονα εξυπηρέτης και πελάτης mailslot, ούτως ώστε να γίνεται δυνατή η αμφίδρομη επικοινωνία με τη χρήση πολλαπλών mailslots.

Τα mailslots παρέχουν έναν εύκολο τρόπο στις εφαρμογές να στέλνουν και να λαμβάνουν σύντομα μηνύματα. Επίσης παρέχουν τη δυνατότητα εκπομπής (broadcast) μηνυμάτων σε όλους τους υπολογιστές ενός τοπικού δικτύου.

3.6. Σωληνώσεις (Pipes)

Το Win32 API παρέχει δύο τύπους σωληνώσεων για αμφίδρομη επικοινωνία: ανώνυμα (anonymous ή unnamed) και επώνυμα (named) pipes [56]. Τα ανώνυμα pipes επιτρέπουν συσχετιζόμενες διεργασίες να μεταφέρουν πληροφορίες μεταξύ τους. Τυπικά, ένα ανώνυμο pipe χρησιμοποιείται για την ανακατεύθυνση της τυπικής εισόδου ή εξόδου (standard input or output) μιας θυγατρικής διεργασίας (child process) έτσι ώστε να μπορεί να ανταλλάσσει δεδομένα με την μητρική διεργασία (parent process). Για την ανταλλαγή δεδομένων και προς τις δύο κατευθύνσεις (λειτουργία duplex), πρέπει να δημιουργηθούν δύο ανώνυμα pipes. Η μητρική διεργασία γράφει δεδομένα στο ένα pipe χρησιμοποιώντας το handle εγγραφής της, ενώ η θυγατρική διεργασία διαβάζει τα δεδομένα από αυτό το Pipe χρησιμοποιώντας το handle ανάγνωσής της. Παρομοίως, η θυγατρική διεργασία

γράφει δεδομένα στο άλλο pipe και η μητρική διεργασία τα διαβάζει από αυτό. Τα ανώνυμα pipes δεν μπορούν να χρησιμοποιηθούν σε δίκτυο, ούτε μπορούν να χρησιμοποιηθούν μεταξύ μη συσχετισμένων διεργασιών.

Τα επώνυμα pipes χρησιμοποιούνται για να μεταφέρουν δεδομένα μεταξύ διεργασιών που δεν είναι συσχετισμένες ή βρίσκονται σε διαφορετικούς υπολογιστές. Τυπικά, μια *διεργασία εξυπηρέτης επωνύμου pipe (named-pipe server process)* δημιουργεί ένα επώνυμο pipe με ένα γνωστό όνομα ή με ένα όνομα το οποίο θα γίνει γνωστό στους πελάτες της. Μια *διεργασία πελάτης επωνύμου pipe (named-pipe client process)* που γνωρίζει το όνομα του pipe μπορεί να ανοίξει την άλλη άκρη του, υποκείμενη στα δικαιώματα πρόσβασης που έχουν δοθεί από τη διεργασία εξυπηρέτη pipe. Αφού και ο εξυπηρέτης και ο πελάτης έχουν συνδεθεί στο pipe μπορούν να ανταλλάξουν δεδομένα εκτελώντας ενέργειες ανάγνωσης και εγγραφής σε αυτό.

3.7. RPC

Το Win32 API παρέχει Remote Procedure Calls (RPC) για να μπορούν οι εφαρμογές να καλούν διεργασίες από μακριά. Έτσι, το RPC κάνει την IPC τόσο εύκολη όσο η κλήση μιας διεργασίας. Το RPC λειτουργεί μεταξύ διεργασιών στον ίδιο υπολογιστή, στο δίκτυο ή στο Διαδίκτυο.

Το RPC που παρέχεται από το Win32 API είναι σύμφωνο με το Open Software Foundation (OSF) Distributed Computing Environment (DCE). Αυτό σημαίνει ότι οι εφαρμογές που βασίζονται στα Win32 και χρησιμοποιούν RPC μπορούν να επικοινωνήσουν με εφαρμογές που τρέχουν σε άλλα λειτουργικά συστήματα που υποστηρίζουν DCE. Το RPC υποστηρίζει αυτόματα την μετατροπή των δεδομένων ανάλογα με τις διάφορες αρχιτεκτονικές hardware και ανάλογα με την σειρά των bytes μεταξύ ανόμοιων περιβαλλόντων.

Οι πελάτες και οι εξυπηρέτες του RPC είναι στενά συνδεδεμένοι (tightly coupled) αλλά διατηρούν την υψηλή απόδοση.

3.8. Windows Sockets

Τα Windows Sockets είναι μια διεπαφή ανεξάρτητη από πρωτόκολλα. Εκμεταλλεύεται τις επικοινωνιακές δυνατότητες των πρωτοκόλλων που βρίσκονται από κάτω. Στα Windows Sockets 2, ένα socket handle μπορεί προαιρετικά να χρησιμοποιηθεί ως handle αρχείου με τις τυπικές λειτουργίες εισόδου/εξόδου αρχείων.

Τα Windows Sockets βασίζονται στα sockets που έγιναν αρχικά δημοφιλή από την Berkeley Software Distribution (BSD). Μια εφαρμογή που χρησιμοποιεί Windows Sockets μπορεί να επικοινωνεί με άλλες υλοποιήσεις sockets σε συστήματα διαφορετικού τύπου.

3.9. WM_COPYDATA

Το μήνυμα WM_COPYDATA επιτρέπει την αποστολή δεδομένων από μια εφαρμογή σε μία άλλη. Όταν τα δεδομένα περνούν από μια εφαρμογή που βασίζεται στα Widows 16 bit σε μια άλλη που βασίζεται στα 32 bit, το σύστημα μεταφράζει τους οποιουσδήποτε δείκτες.

Το βασικότερο μειονέκτημα αυτής της τελευταίας μεθόδου, όπως και των περισσότερων υπολοίπων εκτός των υπηρεσιών του COM είναι ότι θα πρέπει να είναι γνωστές από πριν

οι εφαρμογές ή τα συστατικά που πρέπει να επικοινωνήσουν. Αυτό όμως είναι ανέφικτο σύμφωνα με τις προδιαγραφές του πλαισίου ΟΔΥΣΣΕΑΣ, το οποίο απαιτεί πλήρη απομόνωση μεταξύ των συστατικών. Έτσι το μοντέλο αντικειμένων COM και, όπως θα δειχτεί στη συνέχεια, η εξέλιξή του, το COM+ επιλέχθηκε ως η καλύτερη λύση για την υλοποίηση του πλαισίου ΟΔΥΣΣΕΑΣ.

4. COM (Component Object Model)

Που εφαρμόζεται

Τα αντικείμενα (objects) COM μπορούν να δημιουργηθούν με πολλές γλώσσες προγραμματισμού. Οι object-oriented γλώσσες προγραμματισμού, όπως η C++, παρέχουν μηχανισμούς προγραμματισμού που απλοποιούν την υλοποίηση των αντικειμένων COM. Αυτά τα αντικείμενα μπορούν να βρίσκονται μέσα σε μία μόνο διεργασία (process), σε διαφορετικές διεργασίες, ακόμα και σε διαφορετικούς υπολογιστές.

Σε ποιους προγραμματιστές απευθύνεται

Το COM [25], [26], [36] είναι σχεδιασμένο πρωταρχικά για τους προγραμματιστές της C++ και της Microsoft Visual Basic®.

Απαιτήσεις χρόνου εκτέλεσης (run-time)

Τρέχει σε πολλά λειτουργικά συστήματα.

4.1. Περιγραφή

Το Component Object Model (COM) είναι ένα σύστημα ανεξάρτητο από πλατφόρμα, κατανεμημένο, object-oriented, για τη δημιουργία δυαδικών συστατικών λογισμικού τα οποία μπορούν να αλληλεπιδρούν. Το COM είναι η τεχνολογία θεμελίωσης για τις τεχνολογίες Microsoft OLE (μικτά έγγραφα) και ActiveX® (συστατικά με υποστήριξη Internet), και άλλες [46], [48].

Για την κατανόηση του COM (και κατ' επέκταση όλων των τεχνολογιών που βασίζονται σε αυτό), είναι σημαντικό να διευκρινιστεί ότι δεν πρόκειται για μια object-oriented γλώσσα προγραμματισμού, αλλά για μια τυποποίηση. Επίσης το COM δεν καθορίζει πως θα πρέπει να είναι δομημένη μια εφαρμογή. Η επιλογή της γλώσσας, της δομής και οι λεπτομέρειες της υλοποίησης αφήνεται στον προγραμματιστή της εφαρμογής. Το COM καθορίζει ένα μοντέλο αντικειμένων και τις προγραμματιστικές απαιτήσεις που επιτρέπουν στα αντικείμενα COM (που καλούνται επίσης και συστατικά COM, ή απλά objects) να αλληλεπιδράσουν με άλλα αντικείμενα. Αυτά τα αντικείμενα μπορεί να βρίσκονται μέσα σε μία μόνο διεργασία (process), σε διαφορετικές διεργασίες, ακόμα και σε διαφορετικούς υπολογιστές. Μπορεί να είναι γραμμένα σε διαφορετικές γλώσσες προγραμματισμού και δομικά να μην έχουν καμία ομοιότητα. Γι' αυτόν το λόγο το COM αναφέρεται ως δυαδική τυποποίηση – είναι μια τυποποίηση που εφαρμόζεται αφού ένα πρόγραμμα μεταφραστεί σε δυαδικό κώδικα μηχανής.

Η μόνη απαίτηση του COM ως προς τη γλώσσα προγραμματισμού είναι ότι ο κώδικας θα πρέπει να γράφεται σε μια γλώσσα που να μπορεί να δημιουργεί δομές δεικτών και, είτε ρητά είτε συνεπαγόμενα, να μπορεί να καλεί λειτουργίες (functions) μέσω των δεικτών. Object-oriented γλώσσες προγραμματισμού, όπως η C++ και η Smalltalk, παρέχουν μηχανισμούς προγραμματισμού που απλοποιούν την υλοποίηση των αντικειμένων COM, αλλά και γλώσσες όπως η C, η Pascal, η Ada, η Java, ακόμα και τα προγραμματιστικά περιβάλλοντα BASIC, μπορούν να κατασκευάσουν και να χρησιμοποιήσουν αντικείμενα COM.

Το COM καθορίζει τη θεμελιώδη φύση ενός αντικειμένου COM. Γενικά, ένα αντικείμενο λογισμικού μπορεί να αποτελείται από ένα σύνολο δεδομένων και τις λειτουργίες

(functions) που διαχειρίζονται αυτά τα δεδομένα, Ένα αντικείμενο COM είναι τέτοιο ώστε η πρόσβαση στα δεδομένα του επιτυγχάνεται αποκλειστικά μέσω ενός ή περισσότερων σχετικών συνόλων λειτουργιών. Αυτά τα σύνολα των λειτουργιών καλούνται *διεπαφές (interfaces)* και οι λειτουργίες μιας διεπαφής καλούνται *μέθοδοι (methods)*. Ακόμη, το COM καθορίζει ότι ο μόνος τρόπος πρόσβασης στις μεθόδους μιας διεπαφής είναι μέσω ενός δείκτη στη διεπαφή.

4.2. Αντικείμενα και Διεπαφές του COM

Το COM είναι μια τεχνολογία που επιτρέπει σε αντικείμενα να αλληλεπιδρούν μέσα από τις διεργασίες και τα όρια των υπολογιστών τόσο εύκολα όσο αλληλεπιδρούν τα αντικείμενα μέσα σε μία διεργασία. Αυτό επιτυγχάνεται επειδή σύμφωνα με το COM ο μόνος τρόπος διαχείρισης των δεδομένων που σχετίζονται με ένα αντικείμενο είναι μέσω της *διεπαφής του αντικειμένου*. Όταν χρησιμοποιείται αυτός ο όρος, αναφέρεται σε μια υλοποίηση σε κώδικα μιας διεπαφής σύμφωνης με το δυαδικό COM η οποία είναι συνδεδεμένη με το αντικείμενο.

Το ότι ένα αντικείμενο *υλοποιεί μια διεπαφή* σημαίνει ότι το αντικείμενο χρησιμοποιεί κώδικα που υλοποιεί κάθε μέθοδο της διεπαφής και παρέχει δείκτες σύμφωνους με το δυαδικό COM σε αυτές τις λειτουργίες στη βιβλιοθήκη COM. Μετά το COM αναλαμβάνει τη διάθεση αυτών των λειτουργιών σε οποιονδήποτε πελάτη ζητήσει έναν δείκτη στη διεπαφή, είτε ο πελάτης είναι μέσα είτε έξω από τη διεργασία που υλοποιεί αυτές τις λειτουργίες.

4.2.1. Διεπαφές και υλοποιήσεις διεπαφών

Το COM ενέχει μια θεμελιώδη διαφοροποίηση μεταξύ των ορισμών των διεπαφών (interface definitions) και των υλοποιήσεών τους. Μια *διεπαφή* είναι στην ουσία μια σύμβαση που αποτελείται από μια ομάδα *πρωτοτύπων* λειτουργιών των οποίων ορίζεται η χρήση αλλά όχι η υλοποίηση. Αυτά τα πρωτότυπα λειτουργιών είναι ισοδύναμα με τις καθαρές βασικές κλάσεις της C++ στον προγραμματισμό. Ο ορισμός μιας διεπαφής καθορίζει τις λειτουργίες - μέλη της διεπαφής, δηλαδή τις μεθόδους, τους τύπους δεδομένων που επιστρέφουν, τον αριθμό και τους τύπους των παραμέτρων τους και το τι πρέπει να κάνουν. Καμία υλοποίηση δεν συσχετίζεται με τη διεπαφή.

Μια *υλοποίηση διεπαφής* είναι ο κώδικας που παρέχει ο προγραμματιστής για να πραγματοποιήσει τις ενέργειες που καθορίζονται στον ορισμό μιας διεπαφής. Υλοποιήσεις πολλών διεπαφών που θα μπορούσε να χρησιμοποιήσει ο προγραμματιστής σε μια object-oriented εφαρμογή περιέχονται στις βιβλιοθήκες COM. Πάντως, οι προγραμματιστές είναι ελεύθεροι να αγνοήσουν αυτές τις υλοποιήσεις και να γράψουν τις δικές τους. Μια υλοποίηση διεπαφής συσχετίζεται με ένα αντικείμενο όταν δημιουργείται ένα στιγμιότυπο (instance) αυτού του αντικειμένου και προσφέρει τις υπηρεσίες που προσφέρει το αντικείμενο.

Απλά αντικείμενα μπορεί να υποστηρίζουν μόνο μία διεπαφή. Πιο πολύπλοκα αντικείμενα, όπως αυτά που μπορούν να ενσωματωθούν (embeddable), συνήθως υποστηρίζουν αρκετές διεπαφές. Οι πελάτες έχουν πρόσβαση σε ένα αντικείμενο COM μόνο μέσω ενός δείκτη σε μία από τις διεπαφές του, που με τη σειρά της, επιτρέπει στον πελάτη να καλέσει οποιαδήποτε από τις μεθόδους που αποτελούν τη διεπαφή. Αυτές οι μέθοδοι καθορίζουν πως ο πελάτης μπορεί να χρησιμοποιήσει τα δεδομένα του αντικειμένου.

Στην ουσία, οι διεπαφές καθορίζουν μια σύμβαση μεταξύ του αντικειμένου και των πελατών του. Η σύμβαση καθορίζει τις μεθόδους που συσχετίζονται με κάθε διεπαφή και ποια πρέπει να είναι η συμπεριφορά κάθε μιας από τις μεθόδους όσον αφορά την είσοδο και την έξοδο. Γενικά, η σύμβαση δεν καθορίζει *πως* θα υλοποιηθούν οι μέθοδοι μέσα σε μια διεπαφή. Ένα άλλο σημαντικό χαρακτηριστικό της σύμβασης είναι ότι εάν ένα αντικείμενο υποστηρίζει μια διεπαφή, πρέπει με κάποιο τρόπο να υποστηρίζει όλες τις μεθόδους της. Δεν χρειάζεται σε μια υλοποίηση όλες οι μέθοδοι να κάνουν κάτι – αν ένα αντικείμενο δεν υποστηρίζει τη λειτουργία που περιέχεται σε μια μέθοδο, η υλοποίησή της μπορεί να είναι μια απλή επιστροφή, ή η επιστροφή ενός κατανοητού μηνύματος σφάλματος – αλλά σίγουρα θα πρέπει να υφίστανται.

4.3. Επαναχρησιμοποίηση αντικειμένων

Ένας σημαντικός σκοπός οποιουδήποτε μοντέλου αντικειμένων είναι οι προγραμματιστές των αντικειμένων να μπορούν να επαναχρησιμοποιούν και να επεκτείνουν τα αντικείμενα που παρέχονται από άλλους ως τμήματα των δικών τους υλοποιήσεων. Ένας τρόπος να γίνει αυτό στην C++ και σε άλλες γλώσσες είναι η κληρονομικότητα της υλοποίησης (implementation inheritance), η οποία επιτρέπει σε ένα αντικείμενο να κληρονομήσει κάποιες από τις λειτουργίες του από ένα άλλο αντικείμενο καθώς και να αγνοήσει άλλες λειτουργίες.

Η χρήση της παραδοσιακής κληρονομικότητας της υλοποίησης κατά την αλληλεπίδραση των αντικειμένων σε ένα σύστημα, έχει το πρόβλημα ότι η σύμβαση (η διεπαφή) μεταξύ των αντικειμένων σε μια ιεραρχική υλοποίηση δεν είναι καθαρά ορισμένη. Στην ουσία είναι συνεπαγόμενη και αμφιλεγόμενη. Όταν το μητρικό ή το θυγατρικό αντικείμενο αλλάξει την υλοποίησή του, η συμπεριφορά των συσχετιζόμενων συστατικών μπορεί να γίνει ακαθόριστη, ή ασταθής. Σε μία εφαρμογή, όπου η υλοποίηση μπορεί να διαχειριστεί από μία ομάδα μηχανικών, οι οποίοι ενημερώνουν όλα τα συστατικά την ίδια στιγμή, αυτό δεν είναι μεγάλο πρόβλημα. Όμως, σε ένα περιβάλλον όπου τα συστατικά της μιας ομάδας κατασκευάζονται μετά από επαναχρησιμοποίηση των συστατικών μιας άλλης ομάδας ως μαύρα κουτιά, αυτού του είδους η αστάθεια κάνει την επαναχρησιμοποίηση παρακινδυνευμένη. Ακόμη, η κληρονομικότητα της υλοποίησης, συνήθως λειτουργεί μόνο μέσα στα όρια των διεργασιών. Αυτό κάνει την παραδοσιακή κληρονομικότητα της υλοποίησης μη πρακτική για μεγάλα, επεκτεινόμενα συστήματα που αποτελούνται από συστατικά λογισμικού που κατασκευάζονται από πολλές ομάδες μηχανικών.

Το κλειδί για την κατασκευή επαναχρησιμοποιούμενων συστατικών είναι να υπάρχει η δυνατότητα τα συστατικά να αντιμετωπιστούν ως μαύρα κουτιά. Αυτό σημαίνει ότι το τμήμα του κώδικα που επιχειρεί να επαναχρησιμοποιήσει ένα άλλο αντικείμενο δε γνωρίζει τίποτα, και δε χρειάζεται να γνωρίζει τίποτα, σχετικά με την εσωτερική δομή ή την υλοποίηση του συστατικού που πρόκειται να χρησιμοποιηθεί. Με άλλα λόγια, ο κώδικας που προσπαθεί να χρησιμοποιήσει ένα συστατικό εξαρτάται από την *συμπεριφορά* του αντικειμένου και όχι την ακριβή *υλοποίηση*.

Για την επίτευξη της επαναχρησιμοποίησης τύπου μαύρου κουτιού, το COM υιοθετεί κάποιους άλλους εδραιωμένους μηχανισμούς επαναχρησιμοποίησης που καλούνται *containment/delegation* και *aggregation*. Για την περιγραφή τους το αντικείμενο το οποίο επαναχρησιμοποιείται καλείται *εσωτερικό αντικείμενο* και το αντικείμενο που χρησιμοποιεί αυτό το εσωτερικό αντικείμενο καλείται *εξωτερικό αντικείμενο*.

4.3.1. Containment/Delegation

Αυτός ο τύπος επαναχρησιμοποίησης είναι μια οικεία ιδέα που συναντάται στις περισσότερες object-oriented γλώσσες προγραμματισμού και συστήματα. Το εξωτερικό αντικείμενο συμπεριφέρεται ως πελάτης για το εσωτερικό αντικείμενο. Το εξωτερικό αντικείμενο «περιέχει» το εσωτερικό αντικείμενο και όταν χρειάζεται τις υπηρεσίες του, το εξωτερικό αντικείμενο μεταβιβάζει ρητά την υλοποίηση στις μεθόδους του εσωτερικού αντικειμένου. Έτσι, το εξωτερικό αντικείμενο χρησιμοποιεί τις υπηρεσίες του εσωτερικού αντικειμένου για να υλοποιήσει τον εαυτό του.

Τα δύο αντικείμενα δεν είναι ανάγκη να υποστηρίζουν τις ίδιες διεπαφές, και είναι λογικό να περιέχεται ένα αντικείμενο που υλοποιεί τις διεπαφές που δεν υλοποιεί το εξωτερικό και υλοποιεί τις μεθόδους του εξωτερικού αντικειμένου απλά ως κλήσεις στις αντίστοιχες μεθόδους του εσωτερικού αντικειμένου.

Είναι απλό να υλοποιηθεί το containment για ένα εξωτερικό αντικείμενο. Το εξωτερικό αντικείμενο δημιουργεί το εσωτερικό αντικείμενο που θέλει να χρησιμοποιήσει όπως θα έκανε οποιοσδήποτε πελάτης. Η διεργασία είναι σαν ένα αντικείμενο της C++ το οποίο περιέχει ένα αντικείμενο string της C++ και το χρησιμοποιεί για να διεξάγει ορισμένες διεργασίες με strings, ακόμα και αν το εξωτερικό αντικείμενο δεν θεωρείται από μόνο του ένα string αντικείμενο. Χρησιμοποιώντας το δείκτη του στο εσωτερικό αντικείμενο, μια κλήση σε μια μέθοδο στο εξωτερικό αντικείμενο δημιουργεί μια κλήση σε μια μέθοδο του εσωτερικού αντικειμένου.

4.3.2. Aggregation

Σε αυτόν τον μηχανισμό επαναχρησιμοποίησης, το εξωτερικό αντικείμενο εκθέτει διεπαφές από το εσωτερικό αντικείμενο σαν να ήταν υλοποιημένες στο ίδιο το εξωτερικό αντικείμενο. Αυτό είναι χρήσιμο όταν το εξωτερικό αντικείμενο πάντα αναθέτει κάθε κλήση σε κάποια από τις διεπαφές του στην ίδια διεπαφή του εσωτερικού αντικειμένου. Το aggregation είναι στην ουσία μια ειδική περίπτωση του containment/delegation και προσφέρει τη διευκόλυνση της αποφυγής πρόσθετου overhead υλοποίησης στο εξωτερικό αντικείμενο σε τέτοιες περιπτώσεις.

4.4. Η βιβλιοθήκη COM

Οποιαδήποτε διεργασία χρησιμοποιεί COM πρέπει να αρχικοποιεί (initialize) και να ελευθερώνει (unitialize) τη βιβλιοθήκη COM. Το COM εκτός από το ότι είναι μια τυποποίηση, επίσης υλοποιεί κάποιες σημαντικές υπηρεσίες στη βιβλιοθήκη. Η Βιβλιοθήκη είναι σε μορφή μιας ομάδας από DLLs και EXEs (κυρίως το OLE32.DLL και το RPCSS.EXE) στα Microsoft® Windows®, και περιέχει:

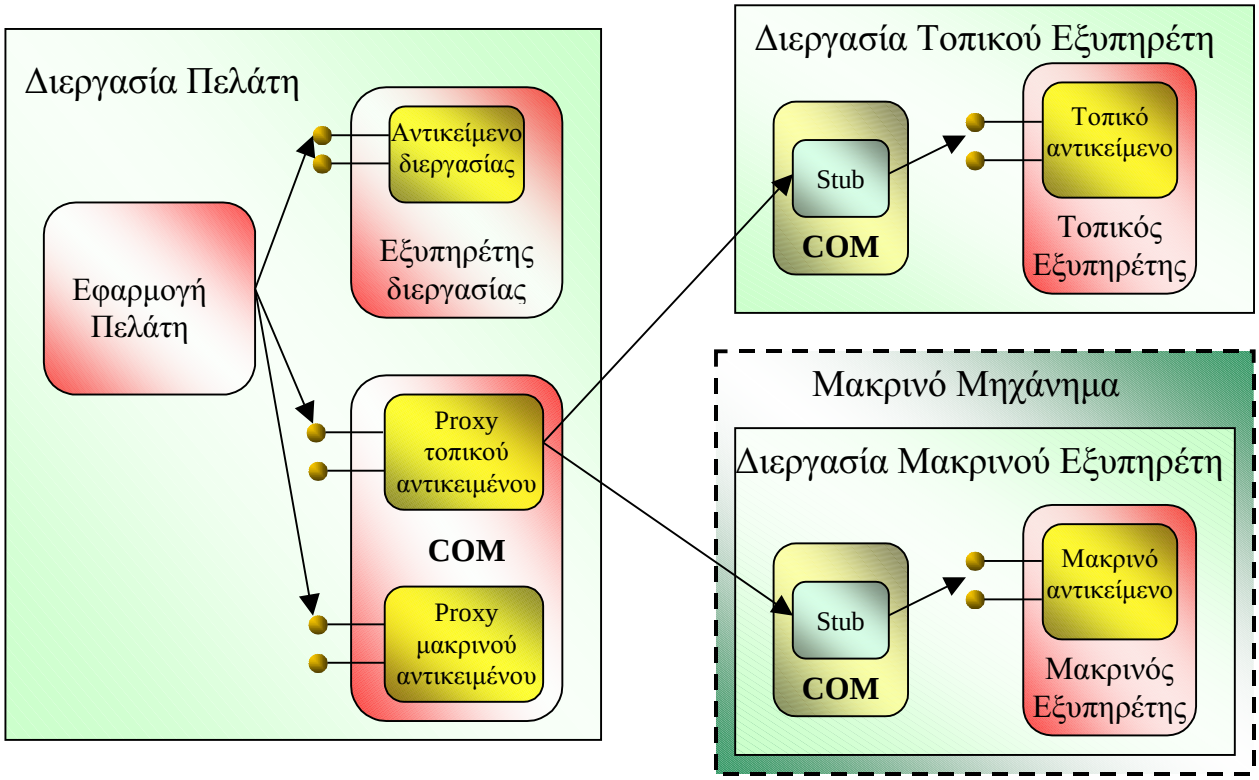
- Ένα μικρό αριθμό από θεμελιώδεις λειτουργίες API οι οποίες διευκολύνουν τη δημιουργία των εφαρμογών COM, είτε είναι πελάτες είτε εξυπηρέτες. Για τους πελάτες το COM δίνει βασικές λειτουργίες για την δημιουργία αντικειμένων. Για τους εξυπηρέτες, το COM δίνει τα μέσα για να εκθέσουν τα αντικείμενά τους.
- Υπηρεσίες ανιχνευτή υλοποίησης μέσα από τις οποίες το OCM προσδιορίζει από έναν μοναδικό δείκτη κλάσης (class identifier CLSID) ποιος εξυπηρέτης υλοποιεί αυτήν την κλάση και που βρίσκεται. Αυτή η υπηρεσία περιλαμβάνει και υποστήριξη για ένα επίπεδο απομόνωσης, συνήθως με ένα registry του συστήματος, μεταξύ της ταυτότητας μιας κλάσης αντικειμένου και του πακεταρίσματος της υλοποίησης. Έτσι ώστε οι

πελάτες να είναι ανεξάρτητοι από το πακετάρισμα, το οποίο μπορεί να αλλάξει στο μέλλον.

- Διάφανες μακρινές κλήσεις διεργασιών (remote procedure calls) όταν ένα αντικείμενο τρέχει σε έναν τοπικό ή μακρινό εξυπηρέτη.
- Έναν τυποποιημένο μηχανισμό για να μπορεί μια εφαρμογή να ελέγχει πόση μνήμη δεσμεύεται στην διεργασία της, και ιδιαίτερα μνήμη που πρέπει να περάσει ανάμεσα από συνεργαζόμενα αντικείμενα, έτσι ώστε να μπορεί να ελευθερωθεί σωστά.

4.5. Το μοντέλο Client/Server του COM

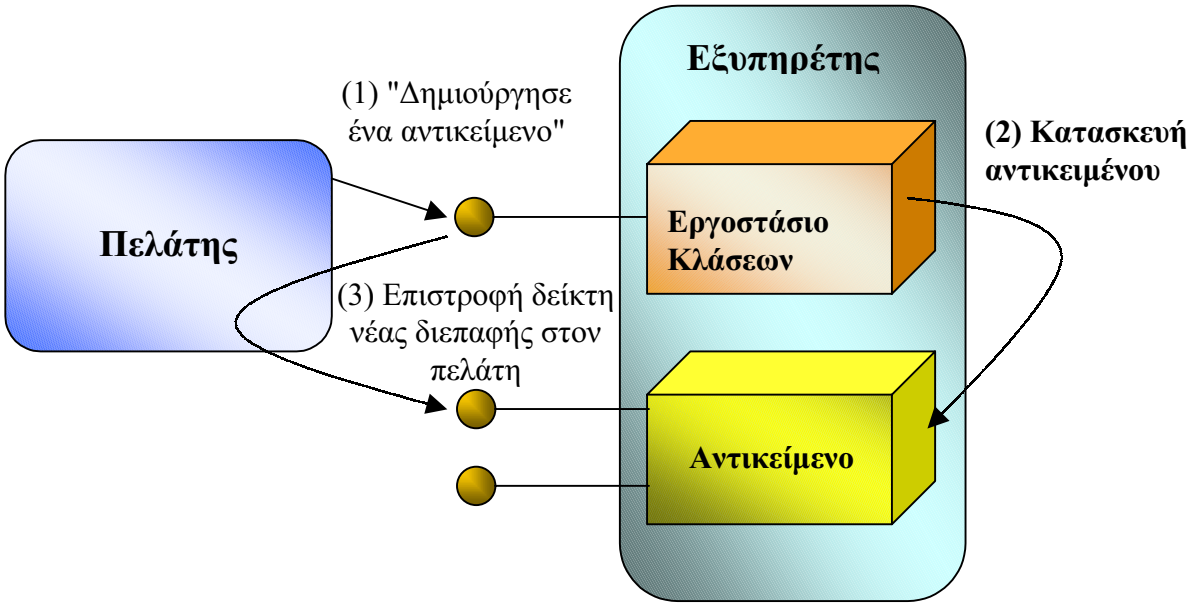
Το Com υποστηρίζει ένα μοντέλο αλληλεπίδρασης τύπου client/server μεταξύ ενός χρήστη των υπηρεσιών ενός αντικειμένου, τον πελάτη (client), και του παροχέα αυτού του αντικειμένου και των υπηρεσιών του, του εξυπηρέτη (server). Για την ακρίβεια, ο πελάτης είναι ένας *οποιοδήποτε* τμήμα κώδικα (όχι αναγκαστικά μια εφαρμογή) που με κάποιο τρόπο αποκτά ένα δείκτη (pointer) μέσω του οποίου μπορεί να έχει πρόσβαση στις υπηρεσίες ενός αντικειμένου και στη συνέχεια καλεί αυτές τις υπηρεσίες όταν είναι αναγκαίο. Ο εξυπηρέτης είναι ένα τμήμα κώδικα που υλοποιεί το αντικείμενο και το δομεί με τέτοιο τρόπο, ώστε η βιβλιοθήκη COM να μπορεί να αντιστοιχίσει αυτήν την υλοποίηση σε έναν αναγνωριστή κλάσης (class identifier), ή CLSID. Αυτό που διαφοροποιεί έναν εξυπηρέτη από μια γενική υλοποίηση ενός αντικειμένου είναι η παρουσία του αναγνωριστή κλάσης.



Σχήμα 2: Οι πελάτες εντοπίζουν και προσπελαύνουν τα αντικείμενα μέσω υπηρεσιών εντοπισμού υλοποίησης του COM. Στη συνέχεια το COM συνδέει τον πελάτη με ένα αντικείμενο σε έναν εξυπηρέτη.

Η βιβλιοθήκη COM χρησιμοποιεί το CLSID για να παρέχει στους πελάτες υπηρεσίες *εντοπισμού της υλοποίησης*. Ένας πελάτης χρειάζεται μόνο να πει στο COM το CLSID που θέλει και τον τύπο του εξυπηρέτη – ενδο-διεργασιακό, τοπικό, ή μακρινό – που επιτρέπει στο COM να φορτωθεί ή να ξεκινήσει. Με τη σειρά του, το COM, εντοπίζει την υλοποίηση αυτής της κλάσης και αποκαθιστά μια σύνδεση μεταξύ αυτής και του πελάτη. Η σχέση μεταξύ του πελάτη, του COM και του εξυπηρέτη απεικονίζεται στο Σχήμα 2. Σημασία έχει επίσης και η ιδέα της διαφάνειας της θέσης, σύμφωνα με την οποία οι πελάτες και οι εξυπηρέτες δεν χρειάζεται να ξέρουν που πραγματικά βρίσκονται, δηλαδή, αν είναι στην ίδια διεργασία, σε διαφορετικές διεργασίες, ή σε διαφορετικούς υπολογιστές.

Ανεξάρτητα από τον τύπο του εξυπηρέτη που χρησιμοποιείται, ένας πελάτης COM πάντα ζητά από το COM να δημιουργήσει στιγμιότυπα αντικειμένων με τον ίδιο ακριβώς τρόπο. Η απλούστερη μέθοδος για τη δημιουργία ενός αντικειμένου είναι η κλήση της συνάρτησης CoCreateInstance. Αυτή δημιουργεί ένα αντικείμενο του δεδομένου CLSID και επιστρέφει έναν δείκτη διεπαφής οποιουδήποτε τύπου έχει ζητηθεί από τον πελάτη. Εναλλακτικά ο πελάτης μπορεί να αποκτήσει έναν δείκτη διεπαφής σε αυτό που λέμε *εργοστάσιο κλάσεων (class factory)* με βάση το CLSID καλώντας την CoGetClassObject. Αυτό το αντικείμενο (εργοστάσιο κλάσεων) υποστηρίζει μια διεπαφή που λέγεται IClassFactory μέσω της οποίας ο πελάτης ζητά από το εργοστάσιο να κατασκευάσει ένα αντικείμενο της κλάσης του. Σε αυτό το σημείο ο πελάτης έχει δείκτες διεπαφών για *δύο ξεχωριστά αντικείμενα*, το εργοστάσιο κλάσεων και ένα αντικείμενο αυτής της κλάσης, που το καθένα έχει το δικό του μετρητή αναφοράς. Είναι ένα σημαντικό σημείο που απεικονίζεται στο Σχήμα 3.



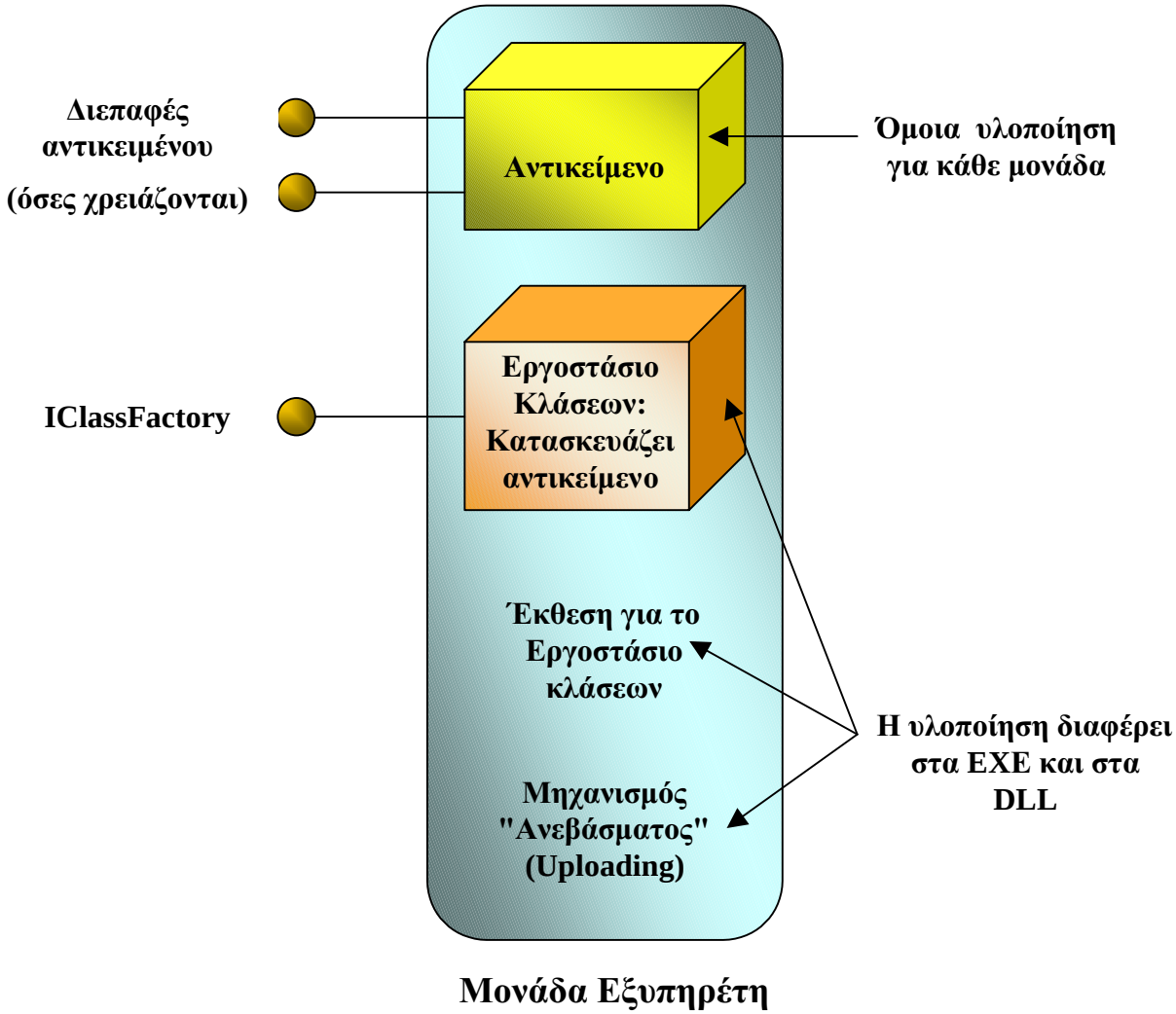
Σχήμα 3: Ένας πελάτης COM δημιουργεί αντικείμενα μέσω ενός εργοστασίου κλάσεων.

Εσωτερικά η συνάρτηση CoCreateInstance καλεί το ίδιο CoGetClassObject. Είναι απλά μια πιο βολική συνάρτηση για πελάτες που θέλουν να δημιουργήσουν ένα αντικείμενο.

Υπάρχουν δύο βασικά είδη εξυπηρετών αντικειμένων:

- **Βασισμένοι σε Dynamic Link Library (.DLL).** Ο εξυπηρέτης υλοποιείται σε ένα module το οποίο μπορεί να φορτωθεί και να εκτελεστεί μέσα από το χώρο διευθύνσεων του πελάτη. (Ο όρος .DLL χρησιμοποιείται για να περιγράψει οποιοδήποτε μηχανισμό διαμοιραζόμενης βιβλιοθήκης που υπάρχει σε μια δεδομένη πλατφόρμα COM) [58].
- **Βασισμένοι σε .EXE.** Αυτός ο εξυπηρέτης υλοποιείται ως ένα stand-alone εκτελέσιμο module.

Αφού το COM υποστηρίζει κατανεμημένα αντικείμενα, επιτρέπει επίσης και στους δυο βασικούς τύπους εξυπηρετών να υλοποιηθούν σε απομακρυσμένους υπολογιστές. Για να μπορούν οι εφαρμογές να ενεργοποιούν απομακρυσμένα αντικείμενα, το COM χρησιμοποιεί τον Service Control Manager (SCM).



Σχήμα 4: Η γενική δομή ενός εξυπηρέτη COM

Όπως ένας πελάτης είναι υπεύθυνος για τη χρήση ενός εργοστασίου κλάσεων και για τη διαχείριση του εξυπηρέτη, ένας εξυπηρέτης είναι υπεύθυνος για την υλοποίηση του

εργοστασίου κλάσεων, την υλοποίηση της κλάσης των αντικειμένων που κατασκευάζει το εργοστάσιο, την έκθεση του εργοστασίου κλάσεων στο COM και την παροχή της δυνατότητας του κλεισίματος (unloading) του εξυπηρέτη κάτω από τις κατάλληλες συνθήκες. Στο Σχήμα 4 φαίνεται ένα διάγραμμα που δείχνει τι υπάρχει μέσα σε ένα module εξυπηρέτη (EXE ή DLL).

Το πως ο εξυπηρέτης ικανοποιεί αυτές τις απαιτήσεις εξαρτάται από το αν ο εξυπηρέτης υλοποιείται ως .DLL ή ως .EXE, αλλά είναι ανεξάρτητο από το αν ο εξυπηρέτης είναι στον ίδιο υπολογιστή με τον πελάτη ή σε έναν απομακρυσμένο υπολογιστή.

5. DCOM και CORBA

Το DCOM (Distributed Component Object Model) και το CORBA (Common Object Request Broker Architecture) είναι δύο δημοφιλή κατανεμημένα μοντέλα αντικειμένων [29], [36], [31]. Θα κάνουμε μια σύγκριση των αρχιτεκτονικών DCOM και CORBA τη βασική προγραμματιστική αρχιτεκτονική, την αρχιτεκτονική απομακρυσμένων διαδικασιών και την αρχιτεκτονική του πρωτοκόλλου επικοινωνίας. Θα δοθεί μια βήμα προς βήμα περιγραφή της απομακρυσμένης ενεργοποίησης αντικειμένων και κλήσης μεθόδων για να φανούν οι ομοιότητες και οι διαφορές των δύο πλαισίων.

5.1. Εισαγωγή

Η ανάπτυξη του Διαδικτύου, η αυξανόμενη δημοτικότητα των προσωπικών υπολογιστών και η ανάπτυξη της πρόσβασης σε δίκτυα με υψηλές ταχύτητες έβαλαν τον προγραμματισμό κατανεμημένων εφαρμογών στην πρώτη γραμμή. Για την απλούστευση του προγραμματισμού των δικτύων και για την υλοποίηση της αρχιτεκτονικής του λογισμικού που βασίζεται σε συστατικά, εμφανίστηκαν ως τυποποιήσεις τα δύο μοντέλα που μελετήσαμε.

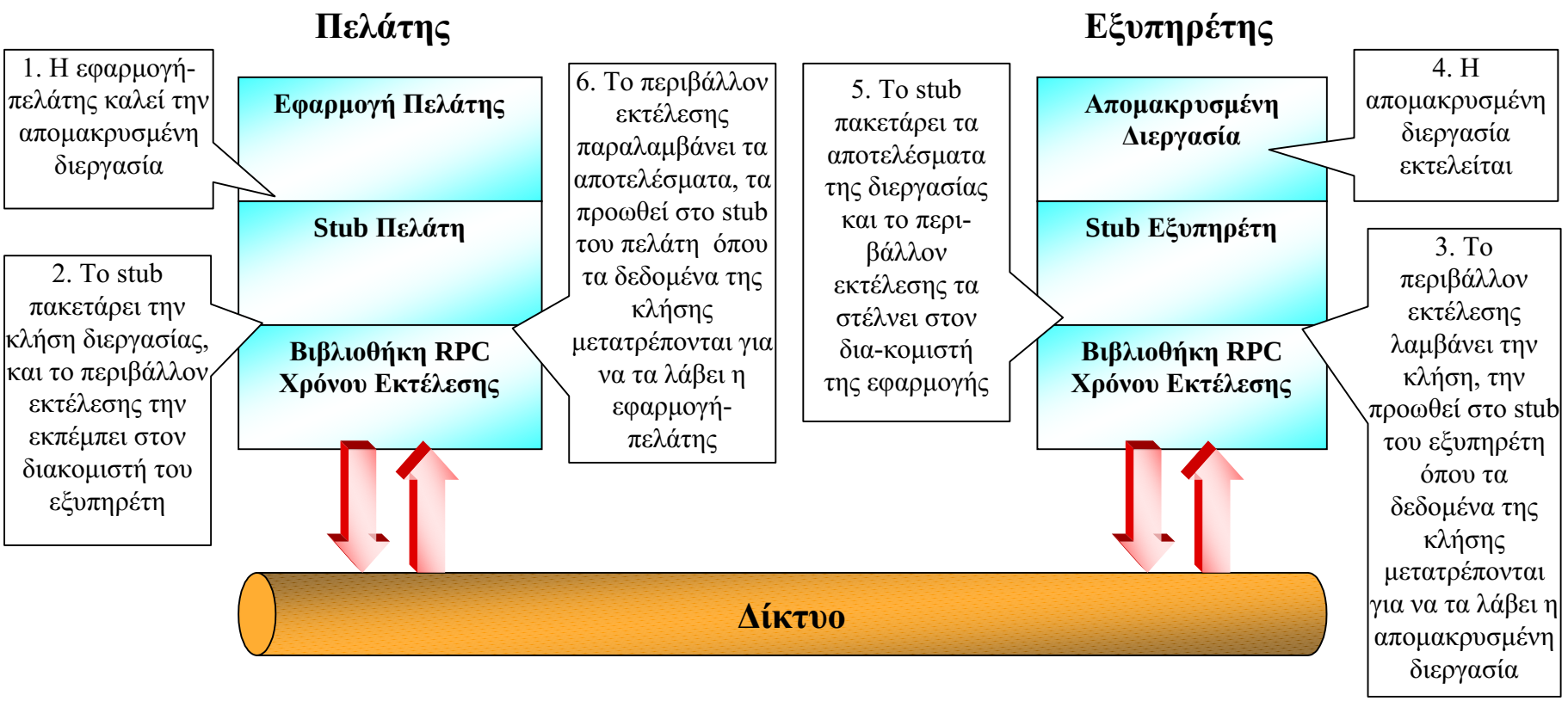
Το DCOM είναι η κατανεμημένη επέκταση του COM (Component Object Model) που δημιουργεί ένα επίπεδο κλήσης απομακρυσμένων διεργασιών αντικειμένων (Object Remote Procedure Call – ORPC) πάνω από το DCE RPC (Distributed Computing Environment) για να υποστηρίξει απομακρυσμένα αντικείμενα. Ένας *εξυπηρέτης COM* μπορεί να δημιουργήσει στιγμιότυπα αντικειμένων πολλαπλών *κλάσεων αντικειμένων*. Ένα αντικείμενο COM μπορεί να υποστηρίζει πολλαπλές **διεπαφές**, που κάθε μία αντιπροσωπεύει μια διαφορετική πλευρά ή συμπεριφορά του αντικειμένου. Μια διεπαφή αποτελείται από ένα σύνολο συσχετιζόμενων λειτουργικά μεθόδων. Ένας πελάτης COM αλληλεπιδρά με ένα αντικείμενο COM αποκτώντας έναν δείκτη σε μία από τις διεπαφές του αντικειμένου και καλώντας μεθόδους μέσω αυτού του δείκτη, σαν να ήταν το αντικείμενο μέσα στο χώρο διευθύνσεων του πελάτη. Το COM καθορίζει ότι οποιαδήποτε διεπαφή πρέπει να ακολουθεί μια τυποποιημένη εμφάνιση στη μνήμη, η οποία είναι η ίδια με το virtual function table της C++. Αφού η τυποποίηση είναι στο δυαδικό επίπεδο, επιτρέπει την ολοκλήρωση δυαδικών συστατικών που πιθανώς να έχουν γραφτεί σε διαφορετικές γλώσσες προγραμματισμού όπως οι C++, Java, Visual Basic.

Το CORBA είναι ένα πλαίσιο κατανεμημένων αντικειμένων που προτάθηκε από μια διεθνή συνεργασία πάνω από 700 εταιριών που λέγεται Object Management Group (OMG). Ο πυρήνας της αρχιτεκτονικής CORBA είναι ο "Μεσίτης Αιτήσεων Αντικειμένων" (**Object Request Broker - ORB**) που δρα ως ο δίαυλος αντικειμένων πάνω στον οποίον τα αντικείμενα αλληλεπιδρούν με διαφάνεια με άλλα αντικείμενα που βρίσκονται τοπικά ή απομακρυσμένα. Ένα αντικείμενο CORBA αντιπροσωπεύεται στον έξω κόσμο από μια διεπαφή με ένα σύνολο μεθόδων. Ένα συγκεκριμένο στιγμιότυπο ενός αντικειμένου αναγνωρίζεται από μια αναφορά αντικειμένου (object reference). Το ORB είναι υπεύθυνο για όλους τους μηχανισμούς που απαιτούνται για την εύρεση της υλοποίησης του αντικειμένου, για την προετοιμασία του ώστε να λάβει την αίτηση, για τη μεταφορά της αιτήσεως σε αυτό και για τη μεταφορά της απάντησης (αν υπάρχει) πίσω στον πελάτη. Η υλοποίηση του αντικειμένου αλληλεπιδρά με το ORB είτε μέσω ενός *Προσαρμογέα Αντικειμένου (Object Adapter – OA)* είτε μέσω της διεπαφής του ORB.

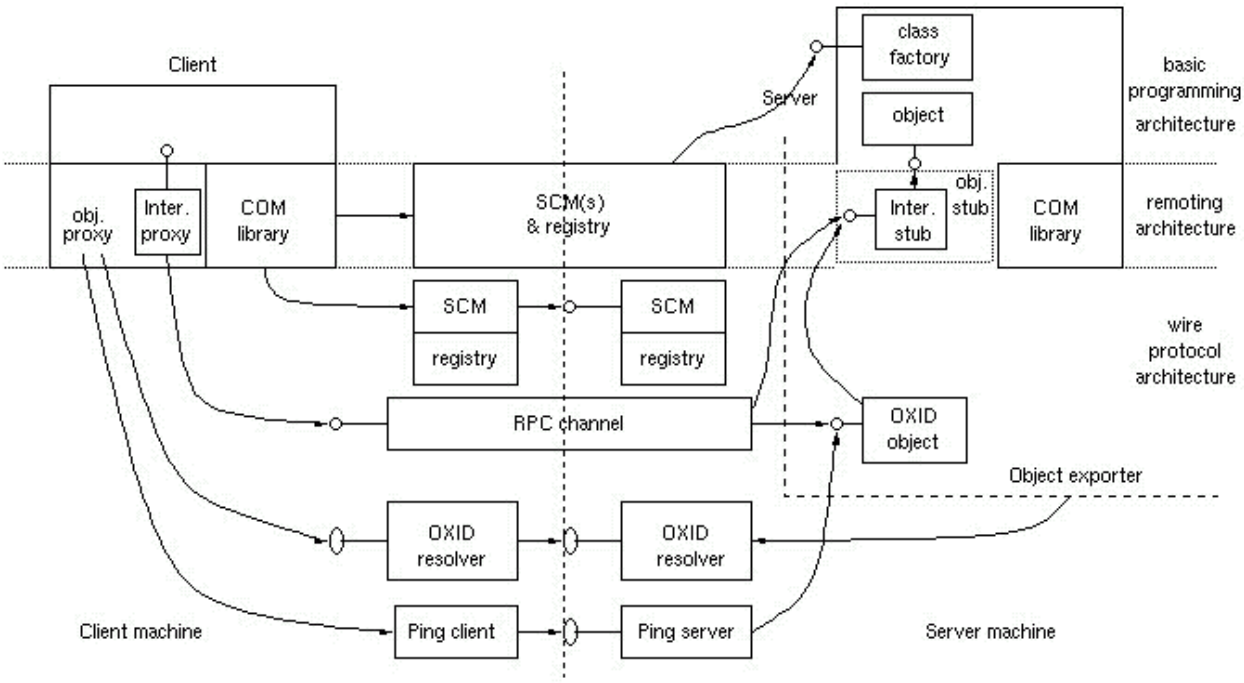
Και τα δύο πλαίσια, το DCOM και το CORBA παρέχουν επικοινωνία τύπου πελάτη-εξυπηρέτη (client-server). Για να ζητήσει μία υπηρεσία, ένας πελάτης καλεί μια μέθοδο που είναι υλοποιημένη από ένα απομακρυσμένο αντικείμενο, το οποίο δρα ως ο εξυπηρέτης στο μοντέλο πελάτη-εξυπηρέτη. Η υπηρεσία που παρέχεται από τον εξυπηρέτη πακετάρεται ως ένα αντικείμενο και η διεπαφή ενός αντικειμένου περιγράφεται σε μία **Γλώσσα Ορισμού Διεπαφών (Interface Definition Language – IDL)**. Οι διεπαφές που ορίζονται από ένα αρχείο IDL έχουν το ρόλο ενός *συμβολαίου* μεταξύ του εξυπηρέτη και των πελατών του. Οι πελάτες αλληλεπιδρούν με έναν εξυπηρέτη καλώντας μεθόδους που περιγράφονται στο IDL. Η πραγματική υλοποίηση του αντικειμένου κρύβεται από τους πελάτες. Στο επίπεδο του IDL υπάρχουν κάποια χαρακτηριστικά του αντικειμενοστραφούς προγραμματισμού, όπως το data encapsulation, ο πολυμορφισμός και η απλή κληρονομικότητα. Η CORBA υποστηρίζει επίσης την πολλαπλή κληρονομικότητα στο επίπεδο του IDL, αλλά το DCOM όχι. Για την επίτευξη του ίδιου στόχου, στο DCOM χρησιμοποιείται η έννοια του αντικειμένου που έχει πολλαπλές διεπαφές.

Και στο DCOM και στο CORBA, οι αλληλεπιδράσεις μεταξύ μιας διεργασίας-πελάτη και ενός αντικειμένου-εξυπηρέτη υλοποιούνται ως αντικειμενοστραφείς επικοινωνίες τύπου RPC. Το Σχήμα 5 δείχνει την τυπική δομή του RPC. Για να κληθεί μια απομακρυσμένη συνάρτηση, ο πελάτης κάνει μια κλήση στο *stub του πελάτη*. Το stub πακετάρει τις παραμέτρους της κλήσης σε ένα μήνυμα αίτησης και καλεί ένα πρωτόκολλο επικοινωνίας για να στείλει το μήνυμα στον εξυπηρέτη. Στην πλευρά του εξυπηρέτη, το πρωτόκολλο επικοινωνίας παραδίδει το μήνυμα στο stub του εξυπηρέτη, το οποίο στη συνέχει ξεπακετάρει το μήνυμα της αίτησης και καλεί την πραγματική συνάρτηση στο αντικείμενο. Στο DCOM, το stub του πελάτη αναφέρεται ως **proxy** και το stub του εξυπηρέτη αναφέρεται ως **stub**. Αντίστοιχα, το stub του πελάτη στο CORBA αναφέρεται ως **stub** και το stub του εξυπηρέτη αναφέρεται ως **skeleton**. Μερικές φορές, ο όρος "proxy" χρησιμοποιείται για την αναφορά σε ένα εκτελούμενο στιγμιότυπο του stub στην CORBA.

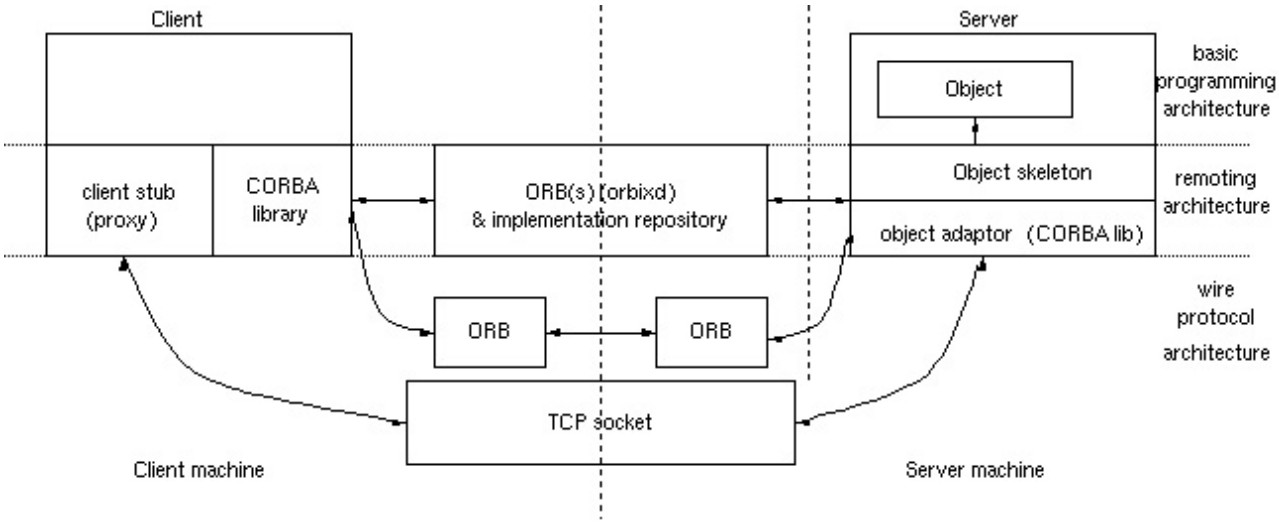
Οι συνολικές αρχιτεκτονικές του DCOM και της CORBA φαίνονται στο Σχήμα 6 και στο Σχήμα 7 αντίστοιχα.



Σχήμα 5: Η δομή του RPC (Remote Procedure Call).



Σχήμα 6: Η συνολική αρχιτεκτονική του DCOM.



Σχήμα 7: Η συνολική αρχιτεκτονική του CORBA.

Στη συνέχεια, θα δοθεί μια βήμα προς βήμα περιγραφή της ενεργοποίησης αντικειμένων και της κλήσης μεθόδων όπως υλοποιούνται στο COM και στο CORBA, στα τρία διαφορετικά επίπεδα που φαίνονται στη Σχήματα. Το κορυφαίο επίπεδο είναι η **βασική προγραμματιστική αρχιτεκτονική (basic programming architecture)**, η οποία είναι ορατή στους προγραμματιστές των προγραμμάτων του πελάτης και του αντικειμένου-εξυπηρέτη. Το μεσαίο επίπεδο είναι η **αρχιτεκτονική απομακρυσμένων διαδικασιών (remoting architecture)**, ή οποία με διαφάνεια κάνει τους δείκτες διεπαφών ή τις αναφορές σε αντικείμενα να έχουν νόημα μεταξύ διαφορετικών διεργασιών. Το χαμηλότερο επίπεδο είναι η **αρχιτεκτονική του πρωτοκόλλου επικοινωνίας (wire**

protocol architecture), η οποία επεκτείνει την αρχιτεκτονική απομακρυσμένων διαδικασιών για να λειτουργεί και διαμέσου διαφορετικών μηχανημάτων.

5.2. Κορυφαίο Επίπεδο: Βασική Αρχιτεκτονική Προγραμματισμού

Στο κορυφαίο επίπεδο φαίνεται η εικόνα που έχει ο προγραμματιστής για το DCOM και το CORBA. Πιο συγκεκριμένα, περιγράφεται πως ένας πελάτης ζητά ένα αντικείμενο και καλεί τις μεθόδους του και πως ένα εξυπηρέτης δημιουργεί ένα στιγμιότυπο ενός αντικειμένου και το κάνει διαθέσιμο στον πελάτη. Το πώς ακριβώς συνδέεται ο πελάτης με τον εξυπηρέτη είναι εντελών κρυμμένο από τους προγραμματιστές. Τα προγράμματα του πελάτη και του εξυπηρέτη αλληλεπιδρούν σαν να βρισκόταν στον ίδιο χώρο διευθύνσεων στο ίδιο μηχάνημα. Οι σημαντικότερες διαφορές μεταξύ του DCOM και του CORBA σε αυτό το επίπεδο περιλαμβάνουν το πώς *ένας πελάτης καθορίζει μία διεπαφή* και τα *εργοστάσια κλάσεων του COM και τις μεθόδους* `IUnknown`. Μια βήμα προς βήμα περιγραφή δίνεται στον Πίνακα 1 και απεικονίζεται στα Σχήματα 8 και 9 για το DCOM και το CORBA, αντίστοιχα.

Αν και ο Πίνακας 1 δείχνει μια κοινή ακολουθία κλήσης του DCOM, πρέπει να σημειωθούν δύο σημεία. Πρώτον, η χρήση των εργοστασίων κλάσεων στο COM είναι προαιρετική. Ένα αντικείμενο του εξυπηρέτη μπορεί να καλέσει την `CoregisterClassObject()` για να δηλώσει οποιονδήποτε δείκτη διεπαφής και οι πελάτες μπορούν να καλέσουν ένα άλλο API του COM που λέγεται `CoGetClassObject()` για να πάρουν αυτόν τον δείκτη. Δεύτερον, η `CoCreateInstance()` δεν δημιουργεί αναγκαία ένα νέο στιγμιότυπο. Μέσα στην `IClassFactory::CreateInstance()`, ο εξυπηρέτης μπορεί να επιλέξει να επιστρέφει συνέχεια τον ίδιο δείκτη διεπαφής, έτσι ώστε διαφορετικοί πελάτες να μπορούν να συνδεθούν στο ίδιο στιγμιότυπο αντικειμένου με μια συγκεκριμένη κατάσταση. Ένας άλλος τρόπος σύνδεσης σε ένα στιγμιότυπο αντικειμένου είναι η χρήση των *monikers* ή/και του *Running Object table*.

DCOM	**CORBA**
Ενεργοποίηση αντικειμένων	
1. • ••••••• ••••• ••• `CoCreateInstance()` ••• •••••••••• ••• COM ••••••••••••• •• Class IDs ••• ••••••••••.	**1.** • ••••••• ••••• •• ::`bind()`, • ••••• ••••• ••• ••••••• ••••••••• ••• stub ••• ••••••.
2. • ••••••• ••• COM ••••••• •••• ••••••••• ••••••••••.	**2.** •• ORB ••••••• •••• ••••••••• • ••••••• ••••••••• ••• •••••••••• ••• •••••••••• •• ••••••••• •••••••.
3. O ••••••••••• •••••••••• ••••••••• ••••••• ••• ••• •• `CLSIDs` ••• ••••••••••••• ••• ••••• ••• `CoRegisterClassObject()` ••• •• ••••••• •• •••• ••••••••••. • ••••••••••• ••••••••• ••• •••• •• •••••• •••• ••• •• ••• •••••••••• ••• ••• ••••••••••• ••• ••. •• •••••••••••• ••••••• ••••••• •• ••••••••••••• •• •••• threads.	**3.** • ••••••••••• ••••••••••• •••••••••• •••• ••• ••••••••••••• ••••••••••••. (•• •••• ••••••••• •••••••••, •••••••• ••••••• ••• •• ••••••••• ••• •• •••••• ••• •••••••• ••• •••••••••••••). • ••••••••• ••••• ••• `CORBA::BOA::impl_is_ready()` ••• •• •••••••••••• •• ORB ••• ••••• ••••••• •• •••••• ••••••••

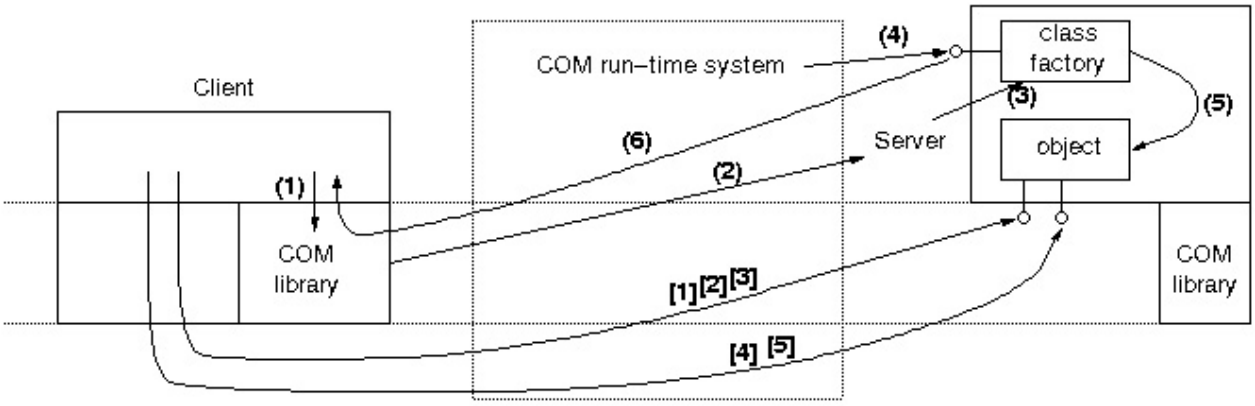
DCOM	**CORBA**
Ενεργοποίηση αντικειμένων	
•• •••• threads. **4.** •• COM ••••••• •••• ••••••• IClassFactory ••• •• CLSIDs ••• ••••••••• ••• •• •••••• ••••• ••• CreateInstance(). **5.** •••• CreateInstance(), • ••••••••• ••••••••• ••• ••••••••••• ••• ••••••••••• ••• ••••• ••• ••••• •••• QueryInterface() ••• •• ••••• •••• •••••• •• ••••••• ••• •• IID ••• ••••••••••. **6.** •• COM ••••••••• •• •••••• ••• ••••••• •••• ••••••.	•••••••. **4.** •• ORB ••••••••• ••• ••••••• ••• •••••••••••• •••• ••••••.
Κλήση Μεθόδων	
1. • ••••••• ••••• ••• ->get() ••• •• ••••••• ••• •••••• • ••••• ••• ••••••• ••••• ••• ::get() ••• ••••••••• •••• •••••••••. **2.** ••• •• ••••• •••• •••••• •• ••• •••• ••••••• ••• ••••• ••••••••••• ••• ••••••••••••, • ••••••• ••••• ••• ->QueryInterface() •••• •••••••, • ••••• ••••• ••• ::QueryInterface() ••• ••••••••••• •••• •••••••••. **3.** •••• ••••••••• •• •• ••••• ••• •••••• •••••••••, • ••••••• ••••• ••• ->Release() (• ••••• •••••• ••• •• ••• ••••••• ••• ::Release() ••• ••••••••••••) [••••••••• 1]. **4.** • ••••••• ••••• ••• ->reset() ••• ••• ••••••• •••••••, • ••••• ••••• ••• ::reset() ••• ••••••••••. **5.** • ••••••• ••••• ••• ->Release() ••• •• ••••••• •••••••, • ••••• ••••• ••• ::Release() ••• ••••••••••••.	**1.** • ••••••• ••••• ••• ->get(), • ••••• ••• ••••••••• ••••• ••• ::get() •••• •••••••••. **2.** • ••••••• ••••• ••• ->reset(), • ••••• ••••• ••• ::reset() •••• •••••••••.

Πίνακας 1: Περιγραφή του κορυφαίου επιπέδου.

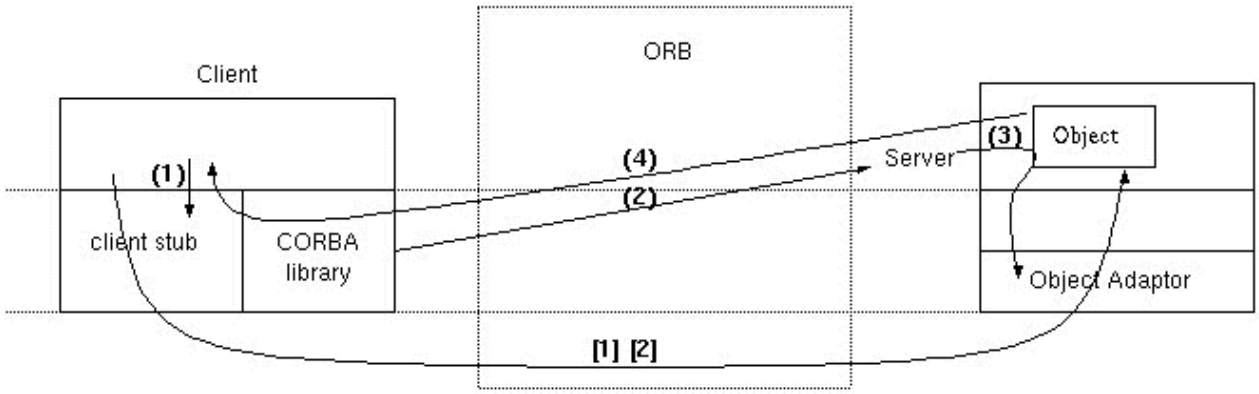
Στο CORBA, ένα αντικείμενο μπορεί να ενεργοποιηθεί καλώντας οποιαδήποτε μέθοδο μιας υπάρχουσας αναφοράς αντικειμένου. Κάποιοι πωλητές παρέχουν ειδικές κλήσεις μεθόδων, όπως η `bind()` στο Orbix [31], για να ενεργοποιήσουν ένα αντικείμενο-

εξυπηρέτη και να πάρουν την αναφορά αντικειμένου του. Ο πελάτης μπορεί να συνδεθεί σε ένα υπάρχον στιγμιότυπο αντί σε ένα νέο, αν βέβαια υπάρχει ένα υπάρχον στιγμιότυπο που να ταιριάζει με τον τύπο που ζητείται. Σημειώνεται ότι ένας πελάτης μπορεί να αποθηκεύσει μια αναφορά αντικειμένου κάνοντάς την αλφαριθμητική σειρά χρησιμοποιώντας την `object_to_string()` και να τη χρησιμοποιήσει αργότερα επαναφέροντάς την με την `string_to_object()`.

Σημείωση για τα Σχήματα: οι αριθμοί σε παρενθέσεις () αφορούν στα βήματα ενεργοποίησης αντικειμένων ενώ οι αριθμοί σε άγκιστρα [] αφορούν σε βήματα κλήσεων μεθόδων.



Σχήμα 8: Βήματα του DCOM στο κορυφαίο επίπεδο.



Σχήμα 9: Βήματα του CORBA στο κορυφαίο επίπεδο.

Μια άλλη διαφορά μεταξύ του DCOM και του CORBA στο προγραμματιστικό επίπεδο είναι ο τρόπος που διαχειρίζονται τις εξαιρέσεις (exceptions) και τα σφάλματα. Το CORBA παρέχει υποστήριξη για συνήθεις εξαιρέσεις της C++ και κάποιες εξαιρέσεις της CORBA. Ακόμη, επιτρέπονται εξαιρέσεις που ορίζονται από τον χρήστη και δηλώνονται στην IDL. Ο IDL compiler αντιστοιχίζει μια εξαίρεση χρήστη σε μία κλάση της C++.

Σε αντίθεση, το DCOM απαιτεί σε αυτό το επίπεδο, όλες οι μέθοδοι να επιστρέφουν ένα 32μπιτο κωδικό σφάλματος που ονομάζεται *HRESULT*. Στο επίπεδο της γλώσσας ή του

εργαλείου προγραμματισμού, μια ομάδα από συμβάσεις και υπηρεσίες παρεχόμενες από το σύστημα (το αντικείμενο `IError`) επιτρέπει τα HRESULTs αποτυχίας να μετατρέπονται σε εξαιρέσεις με έναν τρόπο φυσικό για τη γλώσσα. Το χαμηλότερο επίπεδο του DCOM περιλαμβάνει έναν μηχανισμό γνωστό ως *body extensions*, ο οποίος επιτρέπει τη μεταφορά πληροφοριών εξαιρέσεων (όπως αλφαριθμητικές σειρές που εξηγούν το σφάλμα.

5.3. Μεσαίο Επίπεδο: Αρχιτεκτονική Απομακρυσμένων Διαδικασιών

Το μεσαίο επίπεδο αποτελείται από την υποδομή που χρειάζεται για να δίνει στον πελάτη και τον εξυπηρέτη την ψευδαίσθηση ότι βρίσκονται στον ίδιο χώρο διευθύνσεων. Η περιγραφεί στον Πίνακα 2 δείχνει πως η υποδομή βρίσκει και εκκινεί τον ζητούμενο εξυπηρέτη και τις οντότητες που εμπλέκονται όταν παρουσιάζεται μια κλήση μεθόδου μεταξύ διαφορετικών διεργασιών. Οι ανάλογες απεικονίσεις για το DCOM και το CORBA φαίνονται στα Σχήματα 10 και 11,αντίστοιχα. Οι βασικές διαφορές μεταξύ του DCOM και του CORBA σε αυτό το επίπεδο περιλαμβάνουν το πώς *τα αντικείμενα του εξυπηρέτη δηλώνονται* και *πότε δημιουργούνται τα στιγμιότυπα των proxy/stub/skeleton*.

DCOM	**CORBA**
Ενεργοποίηση αντικειμένων	
1. •• ••• •••••••• ••• •••••• CoCreateInstance(), • ••••••••••• COM •••••••• ••• •••••••• ••• Service Control Manager (SCM). **2.** • SCM ••••••• •• •••• ••••••• ••• ••••••••• ••••••• ••• •• •••••••••••• CLSID. •• •••, • SCM •••• ••• registry •• •••••••••••• •• CLSID ••• •••••••• ••• ••••••••• ••• ••• •••••• ••• •••••••••. **3.** • ••••••••• ••••••• ••• •• •••••••••••• ••••••••• ••••••• •• •••• •••••• ••••••• ••••••••••••. **4.** • SCM ••••••• ••• ••• •••••• ••• •••••• IClassFactory ••• •• ••••••••• ••• •••••••••••• CLSID ••• ••••• •• ••••• ••• CreateInstance(). **5.** •••• • CreateInstance() ••••••••• ••• •••••• IID• •• COM ••••••••• ••• stub •••••••••••• ••• •• ••• •••••••••• ••••••••••••. **6.** •• stub •••••••••••• ••••• marshal ••• •••••• ••• ••••••••, •••• ••• registry •• ••••••• •••	**1.** •• ••• •••••••• ••• •••••• ::bind(), •• stub ••• •••••• •••••••• ••• •••••••• ••• ORB. **2.** •• ORB •••• ••• Implementation Repository •• •••••••••••• ••• ••••• ••• ••••• ••• •••••••• ••• •••••••• ••• •••••••• ••• •••••••• , ••• ••••••••••• ••• ••••••••• (••• Orbix, • orbixd daemon ••••••••••• •• ••••••••••• ••• •••••••••). **3.** • ••••••••• ••••••••• ••••••••••• •••• ••• •••••••••••••• ••••••••••••. •• •••••••• ••••••••••• ••••••••• •••••• •• CORBA::Object ••• •••••• • ••••••••• ••••••••• ••••• •• BOA::create() •• ••• •••••••• ID •••••••• ••• •• ••••• ••• ••••••• ••••••••••••. ••• •••••••• ••••••• ••• ••••••• ••• •••••••••••• •• ORB •••••••• ••• obj_is_ready(). **4.** • ••••••••• ••••••••• ••• ••••••••• •••••• •••••••••••• •••••• ••• ••••••••••• ••• •••••• skeleton.

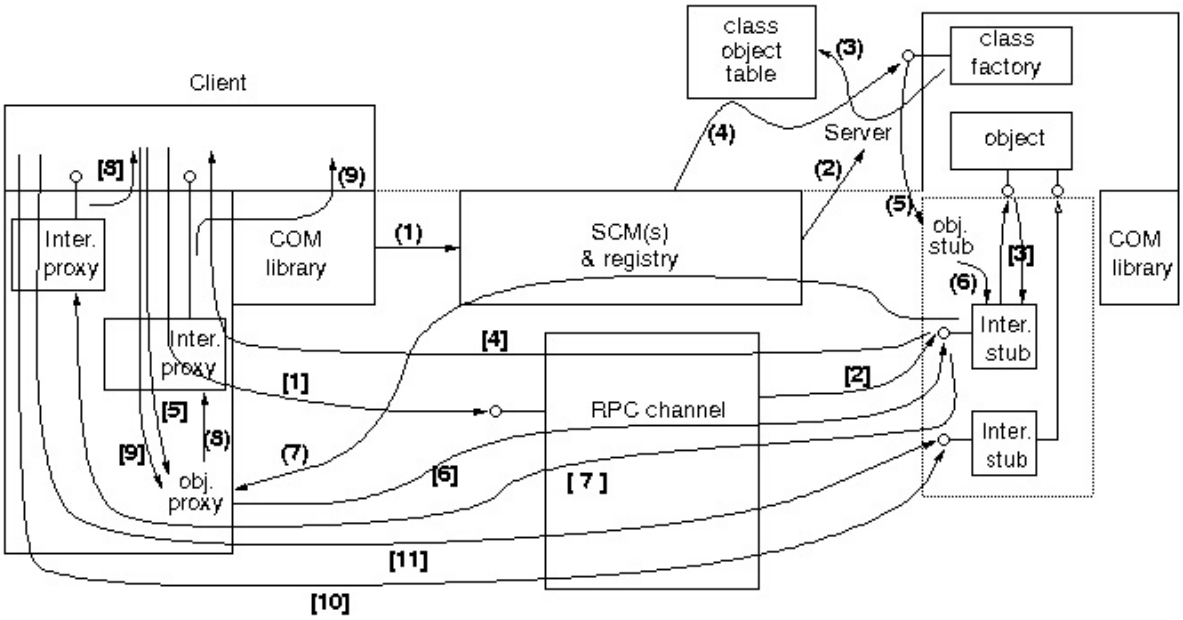
DCOM	**CORBA**
Ενεργοποίηση αντικειμένων	
stub •••••••• ••• •• •••••••••••• IID, ••• •• ••••••••• •• •• ••••••••• IID ••• •••••••• ••• ••••••••••••-•••••••••.	
7. •••• • SCM ••••••••• ••• marshaled •••••• •••• •••• •••••• ••• •••••, •• COM ••••••••• ••• proxy •••••••••••• ••• •• ••••••••••• ••• ••••••••••••.	**5.** •••• •• ORB ••••••••• ••• ••••••• •••••••••••• •••• •••• •••••• ••• ••••••, ••••••••• ••• ••••••••••• ••• proxy ••••••• ••• •• ••••••• •••• •••••• proxy •••••••••••• •• ••• ••••••••• ••••••• ••••••••••••.
8. To proxy •••••••••••• ••••• unmarshal ••• •••••, •••• ••• registry •• •••••••••••• ••• proxy •••••••• ••• •• •••••••••••• IID ••• •• ••••••••• •• •• •••••••••• •••••••• RPC ••• ••••• ••••••••••• ••• stub.	
9. • ••••••••••• COM ••••••••••• •••• •••••• •••• •••••• IID ••• proxy ••••••••.	**6.** •• stub ••• •••••• ••••••••• •••• •••••• ••• ••••••• ••••••••••••.

DCOM	**CORBA**
Κλήση μεθόδων	
1. •• ••• •••••••• ••• •••••• ->get(), •• proxy •••••••• ••••• marshal ••• ••••••••• ••••••••••• ••• ••••• •• •••••• SendReceive() ••• •••••••••••• ••• RPC •••••••• ••• •• ••••••• ••• ••••••.	**1.** •• ••• •••••••• ••• •••••• ->get, •• proxy ••••••••••• ••• ••••••••••••••• Request, ••••• marshal ••• •••••••••• ••••••••••• •• ••••, ••• ••••• ••• Request::invoke(), • ••••• ••••• ••• CORBA::request::send() ••• •• ••••• •• •••••• ••• •••••• ••• ••••••••• •••• CORBA::Request::get_response( ) ••• ••• ••••••••.
2. •• RPC •••••• ••••••• ••• •••••• •••• •••••• ••• •••••••••, ••••••• •• stub •••••••• ••• •• ••••••••• IID ••• ••••• •• •••• ••• •••••• Invoke().	**2.** •••• •• •••••• •••••• •••• •••••••••, •• BOA ••••••• ••• ••••• skeleton, •••••••••••• •• •••••••••• Request ••• •• ••••••• ••• skeleton.
3. To stub •••••••• ••••• unmarshal ••• ••••••••••••, ••••• •• •••••• (••• ••••••••••• ••• •••• •••••• •••••••) ••• ••••••••••••, ••••• marshal ••• ••••• ••••••••••• ••• ••••••••• ••• •• •••••• Invoke.	**3.** •• skeleton ••••• unmarshal ••• •••••••••••• ••• •• •••••••••• request, ••••• •• •••••• 9••• •••••••••••••• ••• •4•• ••••• •••••••) ••• ••••••••••••, ••••• marshal ••• •••••••••• ••••••••••• ••• ••••••••••• ••• •••

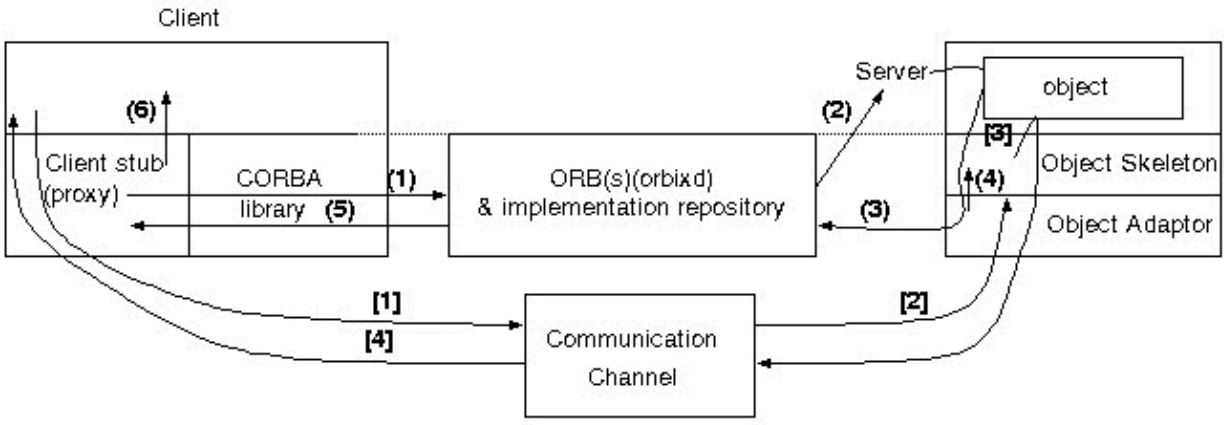
DCOM	**CORBA**
Κλήση μεθόδων	
4. •••• •• RPC •••••• ••••••••• ••• marshaled ••••• ••••••••• •••• •••• •••••• ••• ••••••, •• stub ••••••• ••••••••• ••• ••• ••••• ••• SendReceive(), ••••• unmarshal ••• ••••• ••••••••• ••• ••• ••••••••• •••• •••••• ••• •• ••••••••• ••• ••••• ->get().	•••••• ••• skeleton. •• ORB •••••••••••• ••• •••••• ••••••••• ••• •• ••••••••• ••• buffer •••••••••.
5. •• ••• ••••••• ••• •••••• ->QueryInterface(), •• proxy ••••••• ••••••• ••• •••••• ••• ••••••• IUnknown() ••• proxy ••••••••••••.	**4.** •••• • ••••••• •••••• •••• •••••• ••• ••••••, • ••••• CORBA::Request::get_response() ••••••••• •••• ••••••• •• •••••• ••••••••• ••• •• buffer •••••••••. ••• ••••••• •• proxy ••••• unmarshal ••• ••••• •••••••••, ••••••• ••• ••••••••• ••• ••• ••••••••• •••• •••••• ••• •• ••••••••• ••• ••••• ••• ->get().
6. •• proxy •••••••••••• ••••• •••••••••••• ••• ••••••••• ••••• QueryInterface() ••• ••••••••• •••••••••• •••• ••• ••••• •••••••••• ••• ••••••••• ••••••••.	
7. •• ••• ••••••••• ••• •••• •••••• ••••••••• ••• •• ••• IID, •• COM ••••••••• •• stub ••••••• ••• •• proxy ••• •••• (•• ••••• ••••••••• •• •••• stub •••••••••••• ••• proxy •• •• stub ••••••• ••• •• proxy ••• •• ••••• IID, ••••••••••).	
8. •• proxy ••••••• ••• •• ••••• IID ••••••••• •••• •••••• •••• •••••• •• •• ••• IID ••• •• ••• proxy ••••••••.	
9. •• ••• ••••••• ••• •••••• ->Release(), •• proxy ••••••• ••• •••••• IID ••••••• ••• •••••• ••• proxy ••••••••••••.	
10. •• ••• ••••••• ••• •••••• ->reset() ••• •• ••••••• ••••••••••, •• proxy ••••••• ••• •• ••••••• IID ••••• ••• •••••••••••• ••••• •••• •••••••.	
11. •• ••• ••••••• ••• •••••• ->Release ••• •• ••••••• ••••••••••, •• proxy ••••••• ••• ••••••• IID ••••••• ••• •••••• ••• proxy •••••••••••• ••	

DCOM	**CORBA**
Κλήση μεθόδων	
••••• ••• •••••••• ••••• ••• ••••••••••••• ••••• ••• •• ••••••••••••• •• ••••••• •••••••••••• (••• •••••• ••• •• •••••).	

Πίνακας 2: Περιγραφή του μεσαίου επιπέδου.



Σχήμα 10: Βήματα του DCOM στο μεσαίο επίπεδο.



Σχήμα 11: Βήματα του CORBA στο μεσαίο επίπεδο.

Για την αποστολή δεδομένων διαμέσου διαφορετικών χώρων διευθύνσεων απαιτεί μια διαδικασία που λέγεται marshaling και unmarshaling. Το *marshaling* πακετάρει τις

παραμέτρους μιας κλήσης μεθόδου (στο χώρο του πελάτη) ή τις τιμές επιστροφής (στο χώρο του εξυπηρέτη) σε μία τυπική μορφή για μετάδοση. Το *unmarshaling*, η αντίστροφη διαδικασία, ξεπακετάρει την τυπική μορφή σε μια κατάλληλη μορφή παρουσίασης των δεδομένων στο χώρο διευθύνσεων της διεργασίας που παραλαμβάνει. Σημειώνεται ότι η διαδικασία marshaling που περιγράφεται εδώ λέγεται *standard marshaling* στην ορολογία του DCOM. Το DCOM παρέχει επίσης έναν μηχανισμό *custom marshaling* για να παρακάμψει τη διαδικασία του standard marshaling. Υλοποιώντας μία διεπαφή `IMarshal`, ένα αντικείμενο του εξυπηρέτη ξεκαθαρίζει ότι θέλει να ελέγχει πως και ποια δεδομένα γίνονται marshal και unmarshal και πως θα πρέπει να επικοινωνεί ο πελάτης με τον εξυπηρέτη. Ως αποτέλεσμα, το custom marshaling παρέχει μια **επεκτάσιμη αρχιτεκτονική** για τη σύνδεση σε υποδομές επικοινωνίας που είναι ειδικές για συγκεκριμένες εφαρμογές. Αυτό μπορεί να είναι χρήσιμο για data caching στην πλευρά του πελάτη, για έλεγχο ανοχής σφαλμάτων, κλπ.

Θα περιγράψουμε εδώ κάποιους από τους πρόσθετους όρους τις CORBA που χρησιμοποιούνται στον Πίνακα 2. Όπως ειπώθηκε στην εισαγωγή, το ORB συμπεριφέρεται ως δίαυλος των αντικειμένων. Το Object Adaptor (OA) κάθεται πάνω από το ORB και είναι υπεύθυνο για τη σύνδεση της υλοποίησης του αντικειμένου με το ORB. Τα Object Adaptors παρέχουν υπηρεσίες όπως η δημιουργία των αναφορών των αντικειμένων (object references), της κλήσης μεθόδων, ενεργοποίηση και απενεργοποίηση αντικειμένων, αντιστοίχηση αναφορών αντικειμένων σε υλοποιήσεις. Διαφορετικά στυλ υλοποίησης αντικειμένων έχουν διαφορετικές απαιτήσεις και χρειάζονται υποστήριξη από διαφορετικούς προσαρμογείς αντικειμένων, δηλαδή αντικειμενοστραφείς προσαρμογείς βάσεων δεδομένων για αντικείμενα μέσα σε μια βάση δεδομένων. Το Basic Object adapter (BOA) ορίζει έναν προσαρμογέα αντικειμένων ο οποίος μπορεί να χρησιμοποιηθεί για τις περισσότερες συμβατικές υλοποιήσεις αντικειμένων. Η τυποποίηση CORBA δεν επιβάλλει τον τρόπο που η λειτουργικότητα ORB/BOA θα υλοποιηθεί. Το σύστημα Orbix κατασκεύασε τη λειτουργικότητα ORB/BOA σε δύο βιβλιοθήκες και μια διεργασία daemon (`orbixd`). Ο daemon είναι υπεύθυνος για την θέση και την ενεργοποίηση των αντικειμένων. Οι δύο βιβλιοθήκες, μια στην πλευρά του εξυπηρέτη και μία στην πλευρά του πελάτη, συνδέονται (linked) κατά τη διάρκεια της μεταγλώττισης (compile time) με τις υλοποιήσεις του εξυπηρέτη και του πελάτη, αντίστοιχα, για να παρέχουν τη λειτουργικότητα.

Πρέπει να σημειωθεί ότι το πρόσφατο POA αντικαθιστά το BOA. Η τυποποίηση POA παρέχουν μεταφερσιμότητα για τον κώδικα εξυπηρέτη της CORBA και εισάγει επίσης κάποια νέα χαρακτηριστικά στον Object Adapter.

5.4. Χαμηλό Επίπεδο: Αρχιτεκτονική Πρωτοκόλλου Επικοινωνίας

Το χαμηλότερο επίπεδο καθορίζει το πρωτόκολλο των καλωδίων για την υποστήριξη της εκτέλεσης του πελάτη και του εξυπηρέτη σε διαφορετικά μηχανήματα. Η περιγραφή στον Πίνακα 3 δείχνει πως δημιουργούνται τα αντικείμενα σε ένα απομακρυσμένο μηχάνημα και περιγράφει τις οντότητες που εμπλέκονται όταν μια κλήση μεθόδου μεταφέρεται διαμέσου μηχανημάτων. Τα Σχήματα 12 και 13 απεικονίζουν τα βήματα για το DCOM και το CORBA, αντίστοιχα. Οι βασικές διαφορές μεταξύ του DCOM και του CORBA σε αυτό το επίπεδο περιλαμβάνουν το *πώς οι δείκτες των απομακρυσμένων διεπαφών ή οι αναφορές των αντικειμένων αντιπροσωπεύονται για να μεταδώσουν τις πληροφορίες από τον εξυπηρέτη στον πελάτη* και την *τυπική μορφή με την οποία τα δεδομένα γίνονται marshaled για τη μεταφορά τους σε ένα ετερογενές περιβάλλον*. Σημειώνεται ότι το CORBA δεν

καθορίζει ένα πρωτόκολλο για την επικοινωνία μεταξύ ενός πελάτη και ενός αντικειμένου εξυπηρέτη που τρέχουν σε ORBs του ίδιου κατασκευαστή. Το πρωτόκολλο για την επικοινωνία μεταξύ των ORBs ενός κατασκευαστή εξαρτάται από τον κατασκευαστή. Πάντως, για την υποστήριξη της διαλειτουργικότητας μεταξύ διαφορετικών ORBs, έχει καθοριστεί ένα γενικό πρωτόκολλο το *General Inter-ORB Protocol (GIOP)*. Έχει επίσης καθοριστεί μια αντιστοίχηση του GIOP στις συνδέσεις TCP/IP και είναι γνωστή ως *Internet Inter-ORB Protocol (IIOP)*.

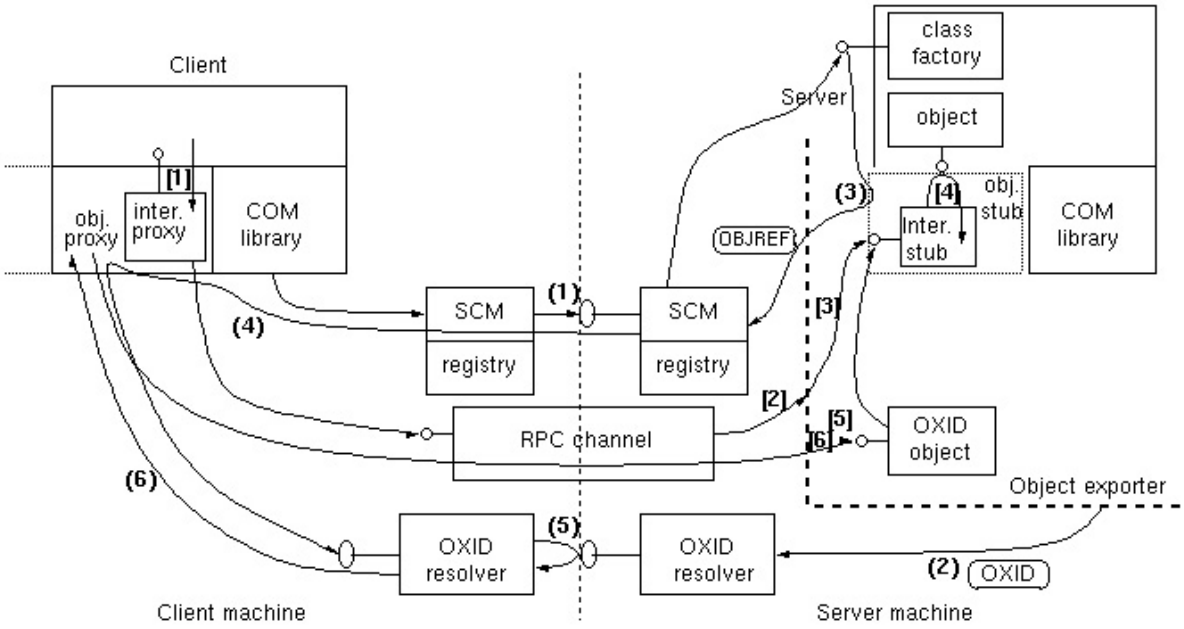
DCOM	**CORBA**
Ενεργοποίηση αντικειμένων	
1. •• ••• •••••••• ••• ••••••• CoCreateInstance, •• •• SCM ••• ••••••• ••• •••••• ••• ••• •••••• registry ••• •• •••••••••• •• •••••• •• •••••••• •• ••• •••• •••••••• •••••••••, ••••• ••• •••••• ••• •••••••• RPC IremoteActivation ••• SCM ••• •••••• ••• •••••••••.	**1.** •• ••• •••••••• ••• ••••••• ::bind(), •• ORB ••• ••••••• ••• •••••• •••• •• ••• •••••• locator •• •••••••• ••• •••••••• ••• •••••••••• •• •••••••••••• •••••••••• ••• •••••••••• ••• •••••• ••• ORB ••• ••••••• ••• •••••••• •••• ••• TCP/IP.
2. •••• • •••••••••• ••••••••• ••• •• SCM ••• ••••••• ••• •••••••••, •••••••••••• •• •••• •••••••• •••••••••••• ••• ••• ••••••••• •••• •••••••••••• ••••••••• •••••••••••• (Object Exporter ID – OXID). • •••••••••••• ••• OXID •• •• ••••••• RPC ••• •••••• •• •••••••••••• ••• ••• ••••••••• •••• ••••••••• ••••••••• •••• OXID resolver ••• ••••••• ••• •••••••••.	**2.** •••• • •••••••••• ••••••••• ••• •• ORB ••• ••••••• ••• •••••••••, •••••••••••• ••• ••• ••••••••• ••• •••••••••• ••• ••••••••••••, •••••••• • •••••••••••• CORBA::Object ••• • ••••••• BOA::create(). •••• •••• BOA::create(), •• BOA •••••••••• ••• socket endpoint, ••• •••••••••• ••••••••• ••• object ID, •••••••• •••• •••••••••, •••••••••••• ••• ••••••• •••••••••• •• ••••••• ••• •••••••• ••• ••• ••••••••••, •• ID ••• •••••••• ••• •• ••••••••• ••• endpoint. ••• ••••••• ••• •••••• ••• ••••••••• IIOP, • ••••••••• ••••••••• ••• **interoperable Object reference (IOR)** ••• •••••••• •• ••••• •••• ••••••••••, •••• •••••• ••••• TCP/IP, ••• ••• key ••••••••••••. To BOA ••••••• ••• ••••••• •••••••••••• ••• ORB.
	3. •••• • ••••••• ••• •••••••••• ••••••••• •••• •••••• •••

DCOM	CORBA
Ενεργοποίηση αντικειμένων	
••••••••••• •••• •••••• ••• •••••• •••• ••• SCM ••• ••••••• ••• ••••••••• ••• ••• ••••••, •• proxy •••••••••••• •••••• •• OXID ••• ••• •••••••••••• ••• OXID resolvers ••• •• OBJREF ••• ••••• •• •••••• IOXIDResolver:ResolveOXID ••• ••••••• OXID resolver. **5.** • OXID resolver ••• ••••••• ••• •••••• ••••••• •• •••• ••• cached •••••••••••• ••• OXID, •• •••, ••••• •• •••••• IOXIDResolver:ResolveOXID ••• OXID resolver ••• ••••••• ••• •••••••••, • ••••• ••••••••• •• •••••••• ••••••• RPC. **6.** • resolver ••• ••••••• ••• •••••• ••••••••• (caches) ••• •••••••••••• ••• ••••••••• •• ••••••• RPC ••• proxy ••••••••••••. •••• ••••••••• ••• proxy •••••••••••• •• •••••••• •• •••• ••• •• proxies •••••••• ••• ••••••••• •• ••• RPC •••••• •• ••••• ••••• •••••••••• •••• •••••••• ••••••••••••.	••••••••••, •• proxy •••••• •• ••••••••• ••• endpoint ••• ••••••• ••• socket ••••••• •• ••• •••••••••.
Κλήση μεθόδων	
1. •• ••• •••••••• ••• •••••• ->get(), •• proxy •••••••• ••••• marshal ••• •••••••••• ••• ••••• **Network Data Representation (NDR)**. **2.** •• •••••• RPC ••••••• ••• •••••• ••• ••••• •••••••• •••••••••••• ••• •••••••••• ••• ••• RPC ••••••• (OXID). **3.** • ••••••• RPC ••• •••••• ••• ••••••••• ••••••• •• ••••• stub •••••••• ••••••••• ••• IPID ••• ••••••••• •••• RPC •••••••••• (header). **4.** •••• ••• ••••• ••• •••••••••• ••••••• ••• •••••••••• ••• •••••••••, •• stub •••••••• ••••• marshal ••• ••••• ••••••••• ••• ••••• NDR. **5.** •• ••• •••••••• ••• •••••• ->QueryInterface(), •• proxy •••••••••••• ••••• •• ••••••	**1.** •• ••• •••••••• ••• •••••• ->get(), •• proxy ••••• marshal ••• •••••••••• ••• ••••• **Common Data Representation (CDR)**. **2.** • •••••• •••••••••••• •••• ••••••••• ••••• •••• ••• socket •••••••• **3.** • skeleton •••••• •••••••••••• •••• ••• •• reference ID •••• ••• •• object_key. **4.** •••• ••• ••••• ••• •••••••••• ••••••• ••• •••••••••• ••• •••••••••, •• skeleton ••••• marshal ••• ••••• ••••••••• ••• ••••• CDR.

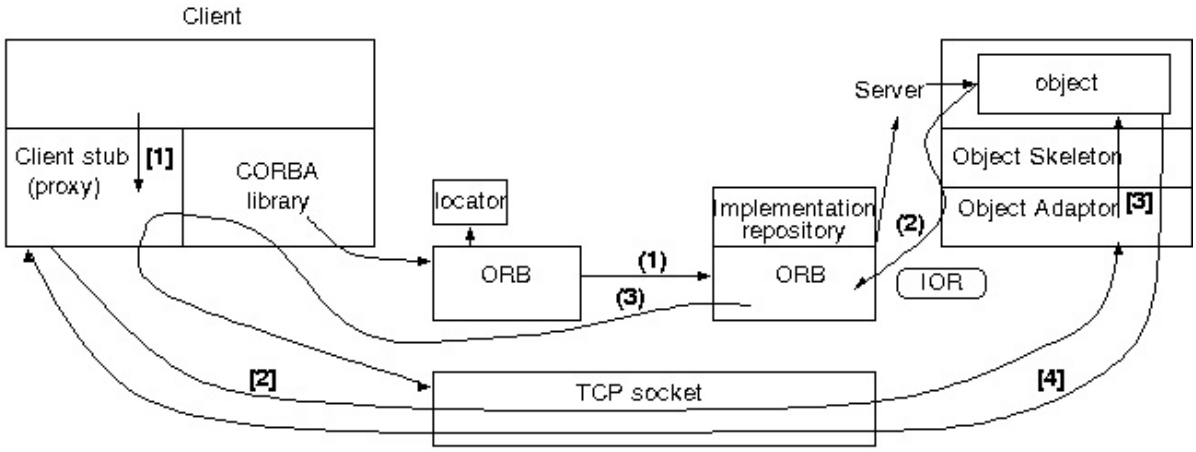
DCOM	CORBA
Ενεργοποίηση αντικειμένων	
IRemUnknown::RemQueryInterface ••• •••••••••• OXID ••• ••••• ••••••• ••••••••••. ••• ••••••• •• •••••••••• OXID ••••• ••• •• •••••• queryInterface() •• (•••• ••••••••) ••••••• •••• •••• •••••••. 6. •• ••• ••••••• ••• •••••• ->Release(), •• proxy •••••••••••• ••••• ••• •••••• IRemUnknown::RemRelease() ••• •••••••••• OXID ••• ••••• ••••••• ••••••••••. •• •••••••••• OXD ••• ••••••• ••••• •• •••••• Release() •• (•••• •••••••••) ••••••• •••• •••• ••••••••.	

Πίνακας 3: Περιγραφή του χαμηλότερου επιπέδου

Το πρωτόκολλο επικοινωνίας του DCOM βασίζεται κυρίως στην τυποποίηση OSF DCE RPC με κάποιες επεκτάσεις. Αυτές περιλαμβάνουν τις απομακρυσμένες αναφορές αντικειμένων, μία διεπαφή `IRemUnknown` για την βελτιστοποίηση της απόδοσης των απομακρυσμένων κλήσεων μεθόδων της `IUnknown` και ένα *pinging πρωτόκολλο*. Η διαδικασία του pinging επιτρέπει σε ένα αντικείμενο εξυπηρέτη να ακυρώνει απομακρυσμένες αναφορές αντικειμένων όταν ένας απομακρυσμένος πελάτης σταματά ανώμαλα. Όταν ένας πελάτης αποκτά έναν δείκτη διεπαφής για ένα απομακρυσμένο αντικείμενο για πρώτη φορά, ο κώδικας ping στη μηχανή του πελάτη προσθέτει το αντικείμενο σε μια ομάδα ping και στέλνει περιοδικά ένα ping στο μηχάνημα του εξυπηρέτη για του γνωστοποιήσει ότι ο πελάτης τρέχει ακόμα. Αν χαθεί ένας προκαθορισμένος αριθμός από συνεχόμενα pings, σημαίνει ότι ο πελάτης έχει σταματήσει ανώμαλα και ότι οι αναφορές αντικειμένων που έχει κρατήσει μπορούν να απελευθερωθούν. Για τη βελτιστοποίηση της απόδοσης, τα pings μπορεί να καταργηθούν όταν χρειάζεται για να μειωθεί η κίνηση στο δίκτυο.



Σχήμα 12: Βήματα του DCOM στο χαμηλότερο επίπεδο.



Σχήμα 13: Βήματα του CORBA στο χαμηλότερο επίπεδο.

5.5. Συμπέρασμα

Οι βήμα προς βήμα περιγραφές των τριών επιπέδων έδειξαν ότι οι αρχιτεκτονικές του DCOM και του CORBA/Orbix είναι παρόμοιες. Και οι δύο παρέχουν μια υποδομή κατανεμημένων αντικείμενων για διαφανείς ενεργοποιήσεις και πρόσβαση σε απομακρυσμένα αντικείμενα. Ο Πίνακας 4 συνοψίζει τους όρους και τις οντότητες των δύο αρχιτεκτονικών. Πρέπει να σημειωθεί ότι πολλές από τις αντιστοιχίες είναι κατά προσέγγιση. Οι κύριες διαφορές τους επίσης συνοψίζονται παρακάτω. Πρώτον, το DCOM υποστηρίζει αντικείμενα με πολλαπλές διεπαφές και παρέχει μια στάνταρ μέθοδο `QueryInterface()` για την περιήγηση μεταξύ των διεπαφών. Αυτό επίσης εισάγει και την ιδέα ένα proxy/stub αντικειμένου να φορτώνει δυναμικά πολλαπλά proxies/stubs διεπαφών στο επίπεδο των απομακρυσμένων διαδικασιών. Τέτοιες έννοιες δεν υπάρχουν στο CORBA. Δεύτερον, κάθε διεπαφή του CORBA κληρονομεί από το

`CORBA::Object`, ο κατασκευαστής του οποίου κάνει κοινές ενέργειες, όπως τη δήλωση των αντικειμένων, τη δημιουργία αναφορών αντικειμένων, δημιουργία στιγμιοτύπων του skeleton, κτλ. Στο DCOM, τέτοιες ενέργειες γίνονται είτε από τα προγράμματα του εξυπηρέτη ή διαχειρίζονται δυναμικά από το σύστημα του περιβάλλοντος χρόνου εκτέλεσης. Τρίτον, το πρωτόκολλο επικοινωνίας του DCOM είναι στενά συνδεδεμένο με το RPC, ενώ του CORBA δεν είναι. Τέλος πρέπει να σημειωθεί ότι η τυποποίηση του DCOM περιέχει πολλές λεπτομέρειες που θεωρούνται ως θέματα υλοποίησης και δεν τυποποιούνται στο CORBA. Ως αποτέλεσμα, πρέπει να χρησιμοποιείται σε πολλά σημεία μια υλοποίηση του CORBA (όπως είναι το Orbix) για επιτευχθούν λειτουργικότητες που το DCOM έχει ενσωματωμένες.

	DCOM	**CORBA**
Κορυφαίο Επίπεδο: Αρχιτεκτονική Προγραμματισμού		
Κοινή βασική κλάση	IUnknown	CORBA::Object
Αναγνωριστής διεπαφής	CLSID	Όνομα διεπαφής
Ενεργοποίηση αντικειμένου στην πλευρά του πελάτη	CoCreateInstance()	Κλήση μεθόδου / bind()
Handle αντικειμένου	Δείκτης αντικειμένου	Αναφορά αντικειμένου
Μεσαίο Επίπεδο: Αρχιτεκτονική Απομακρυσμένων Διαδικασιών		
Όνομα αντιστοίχησης της υλοποίησης	Registry	Implementation Repository
Τύπος πληροφοριών για τις μεθόδους	Type library	Interface Repository
Εύρεση υλοποίησης	SCM	ORB
Ενεργοποίηση υλοποίησης	SCM	OA
Stub πελάτη	proxy	Stub/proxy
Stub εξυπηρέτη	Stub	Skeleton
Χαμηλό Επίπεδο: Αρχιτεκτονική Πρωτοκόλλου Επικοινωνίας		
Resolver του endpoint του εξυπηρέτη	OXID resolver	ORB
Endpoint του εξυπηρέτη	Εξαγωγέας αντικειμένων	OA
Αναφορά αντικειμένου	OBJREF	IOR (ή αναφορά αντικειμένου)
Δημιουργία αναφοράς αντικειμένου	Εξαγωγέας αντικειμένων	OA
Μορφή marshaling δεδομένων	NDR	CDR
Αναγνωριστής στιγμιοτύπου διεπαφής	IPID	Object_key

Πίνακας 4: Σύνοψη αντίστοιχων όρων και οντοτήτων

6. COM+ (Component Services)

Περιγραφή

Το COM+ είναι η εξέλιξη του Component Object Model (COM) και του Microsoft® Transaction Server (MTS) [28], [37]. Το COM+ μπορεί να επεκτείνει εφαρμογές που υλοποιήθηκαν χρησιμοποιώντας τα COM, MTS και άλλες τεχνολογίες που βασίζονται σε COM. Το COM+ διαχειρίζεται πολλές από τις λειτουργίες διαχείρισης πόρων τις οποίες μέχρι τώρα οι προγραμματιστές έπρεπε να αναπτύξουν, όπως η κατανομή των νημάτων (thread allocation) και η ασφάλεια.

Που εφαρμόζεται

Το COM+ μπορεί να χρησιμοποιηθεί για την ανάπτυξη εταιρικών, κρίσιμων, κατανεμημένων εφαρμογών που βασίζονται στο λειτουργικό σύστημα Microsoft® Windows® 2000.

Σε ποιους προγραμματιστές απευθύνεται

Το COM+ είναι σχεδιασμένο πρωταρχικά για τους προγραμματιστές της C++ και της Microsoft Visual Basic®. Μπορεί επίσης να χρησιμοποιηθεί για να διαχειριστεί και να εγκαταστήσει εφαρμογές.

Απαιτήσεις χρόνου εκτέλεσης (run -time)

Είναι ενσωματωμένο στο λειτουργικό σύστημα και απαιτεί τα Windows 2000. Οι εφαρμογές COM+ μπορούν επίσης να τρέξουν σε πελάτες (clients) Windows 98 και Microsoft Windows NT®.

6.1. Εισαγωγή

Η κατασκευή λογισμικού που βασίζεται σε συστατικά είναι πια πολύ πιο απλή χάρη στην επέκταση του COM, το COM+. Το COM+ παρέχει ένα περιβάλλον χρόνου εκτέλεσης (run-time) και υπηρεσίες που χρησιμοποιούνται εύκολα από οποιαδήποτε γλώσσα ή εργαλείο προγραμματισμού και επιτρέπει εκτεταμένη διαλειτουργικότητα μεταξύ συστατικών, ανεξάρτητα από τον τρόπο που έχουν υλοποιηθεί.

6.2. Ανασκόπηση

Η βασική ιδέα του αντικειμενοστραφούς (object-oriented) λογισμικού είναι ότι ένα πρόγραμμα μπορεί να εκφραστεί με βάση αντικείμενα και ενέργειες που μπορούν να εκτελέσουν αυτά τα αντικείμενα. Σε ένα αφαιρετικό επίπεδο τα αντικείμενα είναι έννοιες που μπορεί να καταλάβει ο χρήστης. Για παράδειγμα σε ένα Βοήθημα Διαπροσωπικής Επικοινωνίας, τα κουμπιά, τα παράθυρα, οι μπάρες, οι λέξεις, τα μηνύματα και τα πλαίσια κειμένου είναι αντικείμενα. Καθώς προχωρά ο σχεδιασμός και η υλοποίηση, ορίζονται πρόσθετοι τύποι αντικειμένων που είναι πιο κατανοητοί σε έναν προγραμματιστή.

Μια κλάση (class) είναι ένας τύπος αντικειμένου που ορίζεται βάσει της κατάστασης και της συμπεριφοράς του. Η κατάσταση εκφράζεται από ένα σύνολο ιδιοτήτων (properties). Οι τιμές αυτών των ιδιοτήτων αποτελούν την κατάσταση ενός αντικειμένου, αλλά το σύνολο των ιδιοτήτων είναι το ίδιο για όλα τα αντικείμενα μιας συγκεκριμένης κλάσης. Η συμπεριφορά καθορίζεται από δημόσιες μεθόδους (public methods) τις οποίες μπορούν να

καλέσουν οι πελάτες (clients) του αντικειμένου για να μεταβάλλουν την κατάστασή του. Ο μόνος τρόπος για να αλληλεπιδράσουν οι πελάτες με το αντικείμενο είναι μέσω των ιδιοτήτων και των μεθόδων του. Πρέπει να σημειωθεί ότι οι ιδιότητες και οι μέθοδοι δεν δίνουν πληροφορία για το πως έχει υλοποιηθεί η κλάση και, στην ιδανική περίπτωση, οι πελάτες δεν θα χρειαστεί ποτέ να ξέρουν τίποτα για την υλοποίηση.

Πολλές φορές κάποιες ομάδες από μεθόδους χρησιμοποιούνται από περισσότερες από μία κλάσεις. Μπορεί κανείς να εκμεταλλευτεί αυτήν την κοινή συμπεριφορά ορίζοντας διεπαφές (interfaces). Μια διεπαφή είναι απλά ένας ορισμός από μία ομάδα συσχετιζόμενων ιδιοτήτων και μεθόδων, χωρίς την υλοποίηση. Οι διεπαφές υλοποιούνται από κλάσεις και είναι ένα δυνατό εργαλείο για την επαναχρησιμοποίηση. Ας πούμε ότι έχουμε δύο κλάσεις, την Α και τη Β, που υλοποιούν την ίδια διεπαφή IAddress. Επειδή η αλληλεπίδραση με τα αντικείμενα γίνεται μέσω public διεπαφών, ένας πελάτης που έχει έναν δείκτη στην IAddress, δεν χρειάζεται να γνωρίζει αν μιλά σε ένα στιγμιότυπο της κλάσης Α ή της κλάσης Β, ούτε καν την ύπαρξη των κλάσεων Α και Β. Επίσης οι κλάσεις Α και Β μπορούν να αγνοούν την ύπαρξη η μία της άλλης. Αυτή η δυνατότητα να αντιμετωπίζονται πολλαπλές κλάσεις αντικειμένων σαν να ήταν ο ίδιος τύπος είναι γνωστή ως πολυμορφισμός.

Μια σημαντική σχέση μεταξύ των κλάσεων είναι η κληρονομικότητα (inheritance) που χρησιμοποιείται για την κατασκευή ιεραρχίας κλάσεων. Σε αυτήν την σχέση μία κλάση μπορεί να κληρονομεί και τη διεπαφή και την υλοποίηση από μία άλλη. Άλλες σχέσεις μεταξύ των κλάσεων χρησιμοποιούνται για τον καθορισμό σύνθετων τύπων αντικειμένων. Τέλος, υπάρχει και η δυνατότητα της οργάνωσης των κλάσεων σε μονάδες που μπορούν να μεταφερθούν.

Αντικειμενοστραφείς γλώσσες προγραμματισμού όπως η C++ και η Java προσφέρουν δυνατότητες υλοποίησης κλάσεων και διεπαφών. Οι γλώσσες διαφέρουν στο βαθμό που επιτρέπουν την πρόσβαση σε αντικείμενα ή σε public διεπαφές και που οι διεπαφές είναι ξεχωριστές οντότητες. Πάντως τα βασικά μπορούν γενικά να χρησιμοποιηθούν. Αν και οι αντικειμενοστραφείς γλώσσες προγραμματισμού προσφέρουν πολλά πλεονεκτήματα, δεν είναι από μόνες τους αρκετές για την ανάπτυξη κατανεμημένων εφαρμογών μεγάλης κλίμακας.

Οι περισσότερες γλώσσες προγραμματισμού επικεντρώνονται στη δημιουργία μονοδιεργασιακών και μονογλωσσικών εφαρμογών. Όμως, μπορεί να μην είναι πάντα πρακτικό να γράφεται όλος ο κώδικας μιας εφαρμογής σε μία μόνο γλώσσα. Για παράδειγμα, θα μπορούσαμε να θέλουμε να γράψουμε τη διεπαφή χρήσης χρησιμοποιώντας ένα εργαλείο RAD (γρήγορης ανάπτυξης) και τις διεργασίες διαχείρισης δεδομένων σε C++ για να εκμεταλλευτούμε μια βιβλιοθήκη μαθηματικών συναρτήσεων. Θα πρέπει τότε να μάθει κανείς κάποιους μηχανισμούς επικοινωνίας μεταξύ διεργασιών ή μεταξύ μηχανημάτων για να κάνει την εφαρμογή να λειτουργεί σε διαφορετικά υπολογιστικά συστήματα. Επίσης οι περισσότερες γλώσσες ενθαρρύνουν την χρήση της κληρονομικότητας της υλοποίησης, κάτι που μπορεί να δημιουργεί προβλήματα όταν βασικές κλάσεις τροποποιούνται.

Μοντέλα αντικειμένων του επιπέδου του συστήματος όπως το COM απευθύνονται σε τέτοια προβλήματα. Το COM παρέχει ένα απλό, ισχυρό μοντέλο για την κατασκευή συστημάτων λογισμικού από αλληλεπιδρώντα αντικείμενα. Το COM ορίζει μια δυαδική τυποποίηση για τα αντικείμενα και για την επικοινωνία μεταξύ των αντικειμένων. Όλη η επικοινωνία με ένα αντικείμενο πρέπει να γίνεται μέσω διεπαφών και όλες οι επικοινωνίες πρέπει να φαίνονται ως απλές κλήσεις μεθόδων, ακόμα και αν το αντικείμενο προορισμού

βρίσκεται σε μια άλλη διεργασία ή σε ένα άλλο υπολογιστικό σύστημα. Υλοποιήσεις του COM παρέχουν τις βασικές υπηρεσίες που απαιτούνται για τον εντοπισμό, την ενεργοποίηση και την πρόσβαση σε αντικείμενα και συστατικά που εκθέτουν τις διεπαφές τους. Επειδή το COM ορίζει μια δυαδική τυποποίηση για το πως είναι τα αντικείμενα και οι διεπαφές στη μνήμη, το COM είναι ανεξάρτητο από γλώσσες προγραμματισμού. Οι πελάτες δεν ενδιαφέρονται (και κανονικά δε γνωρίζουν) ποια γλώσσα προγραμματισμού έχει χρησιμοποιηθεί για να γραφτούν τα συστατικά που χρησιμοποιούν.

Μαζί, το COM και οι αντικειμενοστραφείς γλώσσες προγραμματισμού, παρέχουν απλοποίηση της διαδικασίας ανάπτυξης λογισμικού. Μπορούμε να κατασκευάσουμε συστήματα αλληλεπιδρώντων αντικειμένων COM και να γράψουμε συστατικά COM χρησιμοποιώντας αντικειμενοστραφείς γλώσσες προγραμματισμού. Όμως, αν και το μοντέλο προγραμματισμού COM φαίνεται απλό, η κατασκευή συστατικών και εφαρμογών βασισμένων σε συστατικά είναι συχνά δυσκολότερη από ότι θα έπρεπε.

Ένα πρόβλημα είναι οι πολλές ασυμβατότητες μεταξύ του τι θεωρούν αντικείμενο τα εργαλεία και οι γλώσσες προγραμματισμού και του τι θεωρεί αντικείμενο το COM. Εννοιολογικά, αυτό κάνει πιο δύσκολη για τους προγραμματιστές την εκμάθηση της ανάπτυξης με βάση τα συστατικά. Επίσης κάνει πολύ δύσκολη την ανάπτυξη συστατικών και υπηρεσιών του συστήματος που μπορούν πραγματικά να χρησιμοποιηθούν από οποιαδήποτε γλώσσα ή εργαλείο προγραμματισμού. Για παράδειγμα, πολλά εργαλεία και γλώσσες προγραμματισμού υποστηρίζουν μόνο ένα υποσύνολο από τα χαρακτηριστικά και τις δυνατότητες του COM, κι έτσι τα γενικώς προσβάσιμα συστατικά περιορίζονται στο κοινό υποσύνολο που υποστηρίζεται από τα εργαλεία (όπως το Automation) και πρέπει να εκθέτουν τη λειτουργικότητά τους με πολλαπλούς τρόπους. Ιδανικά, το COM θα έπρεπε να παρέχει συνεπή αντικείμενα που είναι εύκολα προσβάσιμα και συμβατά με την έννοια του αντικειμένου που ορίζεται από τις μοντέρνες αντικειμενοστραφείς γλώσσες και εργαλεία προγραμματισμού. Θα έπρεπε επίσης να διασφαλίζει ότι τα νέα APIs και υπηρεσίες θα μπορούσαν να καλεστούν εύκολα από οποιαδήποτε μεταγλωττισμένη ή μεταφρασμένη γλώσσα προγραμματισμού χωρίς πρόσθετη ανάπτυξη, δοκιμή ή τεκμηρίωση. Τέλος, όλα τα αντικείμενα, ανεξάρτητα από την προέλευσή τους, θα έπρεπε να έχουν το χαρακτηριστικό της διαλειτουργικότητας.

Σε ένα συνηθισμένο συστατικό ή εφαρμογή γραμμένη σε C++, μεγάλα τμήματα του κώδικα δεν έχουν καμία σχέση με το προς επίλυση πρόβλημα. Υπάρχει κώδικας για την αρχικοποίηση υπηρεσιών, κώδικας για τη συνεργασία με το λειτουργικό σύστημα, κώδικας για την κυκλοφορία των πληροφοριών του συστήματος... Το μεγαλύτερο κομμάτι αυτού του κώδικα παραμένει το ίδιο από εφαρμογή σε εφαρμογή και από συστατικό σε συστατικό. Εργαλεία όπως ή Visual Basic αντλούν πολλή από τη χρησιμότητά τους κρύβοντας αυτόν τον κοινό κώδικα σε ένα περιβάλλον χρόνου εκτέλεσης (συγκεκριμένο για κάθε εργαλείο) και αφήνοντας τον προγραμματιστή να εστιάσει στον κώδικα που λύνει το πρόβλημα που τον απασχολεί. Πολλά πλαίσια και γεννήτριες εφαρμογών έχουν αναπτυχθεί για να παρέχουν παρόμοιες υπηρεσίες στους προγραμματιστές της C++.

Υπάρχουν μειονεκτήματα στα περιβάλλοντα χρόνου εκτέλεσης και στα πλαίσια (frameworks) του κάθε εργαλείου: περιορίζουν τον προγραμματιστή στις δυνατότητες που υποστηρίζονται από αυτό το εργαλείο. Επίσης, υπάρχει μια σημαντική χρονική καθυστέρηση για να γίνει προσβάσιμη από τα περισσότερα εργαλεία μία νέα δυνατότητα του λειτουργικού συστήματος. Όμως, αν το λειτουργικό σύστημα παρέχει το περιβάλλον χρόνου εκτέλεσης, θα είναι δυνατό να γίνουν τα νέα χαρακτηριστικά διαθέσιμα σε όλα τα εργαλεία αμέσως. Επίσης όταν χρησιμοποιούνται συστατικά γραμμένα σε πολλαπλά

εργαλεία, πιθανώς να είναι φορτωμένα στη μνήμη πολλαπλά περιβάλλοντα χρόνου εκτέλεσης, επιβαρύνοντας άσκοπα το χώρο που καταλαμβάνει στη μνήμη η εφαρμογή. Ένα περιβάλλον χρόνου εκτέλεσης που προσφέρεται από το λειτουργικό σύστημα θα πρέπει να είναι τελικά το μόνο περιβάλλον χρόνου εκτέλεσης που χρειάζεται. Με τα περιβάλλοντα χρόνου εκτέλεσης των διαφόρων εργαλείων προγραμματισμού, πρέπει να διασφαλίζεται ότι το κάθε ένα από αυτά τα περιβάλλοντα είναι εγκατεστημένο οπουδήποτε γίνεται χρήση κάποιου συστατικού που βασίζεται σε αυτό το περιβάλλον. Όμως ένα περιβάλλον εκτέλεσης που παρέχεται από το σύστημα θα εγκαθίσταται πάντα αυτόματα.

Εκτός από την απλοποίηση της ανάπτυξης συστατικών, πολλά εργαλεία προγραμματισμού εισήγαγαν καινοτομικές δυνατότητες για να γίνει η ανάπτυξη εφαρμογών βασισμένων στο COM πιο εύκολη. Κάποια από αυτά τα χαρακτηριστικά ταιριάζουν καλύτερα στο επίπεδο του συστήματος. Για παράδειγμα η Active Template Library (ATL – Βιβλιοθήκη Ενεργών προτύπων) εισήγαγε έναν μηχανισμό βασισμένο σε script για την απλοποίηση της δήλωσης (registration) των συστατικών. Η Visual Basic εισήγαγε εύκολους στη χρήση μηχανισμούς πρόσβασης δεδομένων μέσω data-bound controls. Αυτά είναι χαρακτηριστικά τα οποία είναι χρήσιμα σε πολλά συστατικά.

Ο Microsoft Transaction Server (MTS) εισήγαγε πολλά χαρακτηριστικά για να κάνει πιο εύκολη τη συγγραφή κλιμακούμενων και κατανεμημένων εφαρμογών. Εκτός από την υποστήριξη των συναλλαγών (transactions), ο MTS παρέχει και ένα περιβάλλον εκτέλεσης με υπηρεσίες διαχείρισης νημάτων (threads) και αντικειμένων. Οι προγραμματιστές μπορούν να γράφουν συστατικά υποθέτοντας ότι τα αντικείμενά τους θα προσπελαύνονται μόνο από έναν πελάτη τη φορά – ο MTS αναλαμβάνει τα υπόλοιπα. Αυτό απλοποιεί κατά πολύ την ανάπτυξη των συστατικών.

Ο MTS φανερώνει έναν περιορισμό της αρχιτεκτονικής του COM: Δεν υπάρχει τυποποιημένος μηχανισμός για την πρόσθεση εξωτερικών υπηρεσιών στο COM. Βέβαια, μπορεί κανείς να ορίζει διεπαφές και να γράφει συστατικά αλλά το πρόβλημα προκύπτει όταν θελήσει να παρέμβει στη διαδικασία ενεργοποίησης των αντικειμένων ή να παρακολουθήσει τις κλήσεις μεθόδων. Ο MTS διαχειρίζεται το registry έτσι ώστε να καλείται όταν δημιουργείται ένα αντικείμενο, και στη συνέχεια δημιουργεί ένα περιτύλιγμα που τοποθετείται μεταξύ των πελατών και των πραγματικών αντικειμένων. Αυτό δουλεύει καλά με τον MTS, αλλά δημιουργεί πρόβλημα όταν εμφανίζεται μία υπηρεσία που θέλει να κάνει το ίδιο πράγμα.

6.3. Γιατί COM+;

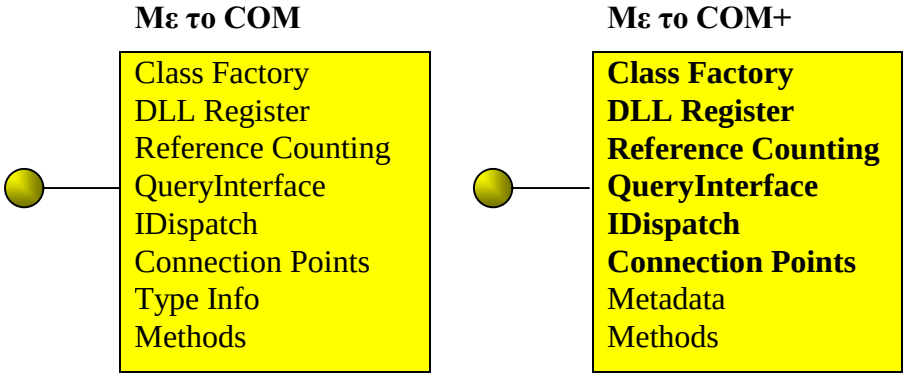
Το COM είναι ένα αρκετά επιτυχημένο μοντέλο αντικειμένων, με υποστήριξη από μια μεγάλη γκάμα εργαλείων προγραμματισμού και μια γρήγορα αναπτυσσόμενη αγορά τρίτων κατασκευαστών για τα συστατικά. Όμως υπάρχουν περιθώρια και περιοχές που παίρνουν βελτίωση και το COM+ προτίθεται να καλύψει αυτές τις περιοχές. Οι πρωταρχικοί στόχοι του COM+ είναι:

- Να είναι ευκολότερη η κατασκευή των συστατικών COM.
- Να αντιμετωπιστούν θέματα-κλειδιά στην ανάπτυξη και την μεταφορά εφαρμογών βασισμένων στο COM.
- Να παρασχεθούν νέες υπηρεσίες στους προγραμματιστές του COM.
- Να κατοχυρωθεί ένας τυποποιημένος μηχανισμός επεκτασιμότητας για την ενσωμάτωση νέων καινοτομιών.

Τα δύο πρώτα σημεία ικανοποιούνται από το περιβάλλον χρόνου εκτέλεσης του COM+. Όπως φαίνεται στο Σχήμα 14, το περιβάλλον εκτέλεσης εξελίχθηκε από τη Microsoft για τη διευκόλυνση της συγγραφής αντικειμένων COM με συγκεκριμένες γλώσσες προγραμματισμού. Τα δύο τελευταία σημεία ικανοποιούνται από υπηρεσίες που έχουν βασιστεί στο περιβάλλον χρόνου εκτέλεσης του COM+, το οποίο εξελίχθηκε από το COM και τον MTS. Μαζί, το περιβάλλον χρόνου εκτέλεσης και οι υπηρεσίες είναι γνωστά ως COM+.

Υπηρεσίες Συστατικών	**Κατανεμημένες Υπηρεσίες**
COM+ (1998) Πλουσιότερο περιβάλλον χρόνου εκτέλεσης και υπηρεσίες για όλες τις γλώσσες	
Java VM (1997) Διάφανο COM για Java	**IIS 4.0 / MTS 2.0 (1997)** Συναλλαγές στο Internet, διαλειτουργικότητα
ATL (1996) Διάφανο COM για C++	**MTS 1.0 (1996)** Συναλλαγές, pooling
Visual Basic 4.0 (1995) Διάφανο COM για Visual Basic	**DCOM (1996)** Κατανομή, ασφάλεια
COM (1992)	

Σχήμα 14: Η εξέλιξη του COM+



Κανονική γραφή: Γράφονται από τον προγραμματιστή
Έντονη γραφή: Το σύστημα παρέχει προκαθορισμένη υλοποίηση

Σχήμα 15: Συγγραφή συστατικών

Σήμερα ένας προγραμματιστής αντικειμένων βασισμένων στο COM (ή ο δημιουργός ενός εργαλείου ανάπτυξης σε COM), πρέπει να ανησυχεί για πολλά θέματα που δεν έχουν σχέση με την ουσιαστική λειτουργικότητα των συστατικών (βλέπε Σχήμα 15). Κάθε συστατικό θα πρέπει να περιέχει μια υλοποίηση της `IUnknown` για να παράσχει την αναφορά (reference counting) και τις υπηρεσίες της `QueryInterface`. Οι πελάτες ενός αντικειμένου πρέπει να χρησιμοποιούν σωστά το reference counting για διαχειρίζονται τον κύκλο ζωής (lifetime) του αντικειμένου και πρέπει να χρησιμοποιούν το `QueryInterface` για να έχουν πρόσβαση στα χαρακτηριστικά του αντικειμένου.

Επίσης κάθε συστατικό χρειάζεται ένα εργοστάσιο κλάσεων (class factory) που γνωρίζει πώς να δημιουργεί αντικείμενα ενός συγκεκριμένου τύπου, και ο απαιτούμενος κώδικας πρέπει να πακετάρεται έτσι ώστε το συστατικό να αρχικοποιείται σωστά στο σύστημα κατά τη διάρκεια της εκτέλεσης. Επίσης, τα συστατικά πρέπει να παρέχουν πληροφορίες δήλωσης (registration). Οι νέες διεπαφές πρέπει να περιγράφονται μέσω της IDL (Interface Definition Language – Γλώσσα Ορισμού Διεπαφών), η οποία κατασκευάζει proxy/stub DLLs και type libraries. Αν ένα συστατικό πρέπει να προσπελαστεί από scripting γλώσσες, πρέπει να υλοποιεί την υποστήριξη Automation μέσω του `IDispatch` και άλλων διεπαφών. Τα συστατικά που θέλουν να πυροδοτήσουν γεγονότα (fire events) και πελάτες που θέλουν να λάβουν γεγονότα πρέπει να υλοποιούν τις διεπαφές `IConnectionPoint`. Πρέπει ακόμη κάποιος να σκεφτεί αν το εργαλείο προγραμματισμού που χρησιμοποιεί συμμορφώνεται στη δυαδική τυποποίηση για τα αντικείμενα στη μνήμη.

Ο περισσότερος κώδικας για την υλοποίηση των `IUknown`, `IDispach`, των γεγονότων (events), των εργοστασίων κλάσεων (class factories) και του πακεταρίσματος των συστατικών είναι σχεδόν όμοιος για όλα τα συστατικά. Έτσι ο πρώτος στόχος του COM+ είναι να παράσχει ένα περιβάλλον χρόνου εκτέλεσης που θα προσφέρει μια προκαθορισμένη υλοποίηση για τη διαχείριση αυτών των θεμάτων για τα πιο κοινά σενάρια (βλέπε Σχήμα 15). Οι προγραμματιστές υλοποιούν μερικά κλάσεις για να διαχειριστούν τη λογική της εφαρμογής και να παράσχουν πληροφορίες για τα χαρακτηριστικά των κλάσεων. Το περιβάλλον εκτέλεσης του COM+ κάνει τα υπόλοιπα. Δε χρειάζεται ειδικός κώδικας για τη δήλωση των κλάσεων στο registry, για την περιγραφή της λειτουργικότητάς τους στους πελάτες, για την παροχή ενός εργοστασίου κλάσεων για τη δημιουργία των αντικειμένων, για τη διαχείριση του κύκλου ζωής των αντικειμένων ή για οτιδήποτε δεν είναι άμεσα συσχετισμένο με τη λογική της εφαρμογής που περιλαμβάνει το συστατικό. Αν η υλοποίηση του περιβάλλοντος εκτέλεσης δεν ικανοποιεί τις ανάγκες του προγραμματιστή, μπορεί πάντα να χρησιμοποιήσει τις παραδοσιακές τεχνικές του COM για την κατασκευή προσαρμοσμένων υλοποιήσεων.

Διάφορα εργαλεία προγραμματισμού και ειδικά η Visual Basic ήδη έχει αντιμετωπίσει αυτά τα θέματα για τον κατασκευαστή του συστατικού και για τον πελάτη του συστατικού. Το πλεονέκτημα ενός περιβάλλοντος εκτέλεσης που παρέχεται από το σύστημα είναι ότι το ίδια περιβάλλον μπορεί να χρησιμοποιηθεί από οτιδήποτε, άσχετα από τη γλώσσα ανάπτυξης. Αυτό προσφέρει βελτιώσεις στην απόδοση και πιο συνεπή συμπεριφορά των συστατικών. Αλλά τα πραγματικά πλεονεκτήματα του COM+ για την ανάπτυξη με εργαλεία όπως η Visual Basic είναι οι υπηρεσίες και οι μηχανισμοί επέκτασης που θα περιγραφούν αργότερα.

Ένας δεύτερος στόχος του COM+ είναι να διευθετηθούν βασικά θέματα στην ανάπτυξη και μεταφορά εφαρμογών βασισμένων σε COM. Πολλές δυσκολίες που συναντώνται κατά τη δημιουργία των συστατικών σχετίζονται με τον ορισμό νέων διεπαφών μέσω της IDL. Με το COM+ οι διεπαφές ορίζονται χρησιμοποιώντας κοινές γλώσσες προγραμματισμού. Δεν χρειάζεται να είναι κάποιος ειδικός στην IDL ή να διαχειρίζεται ξεχωριστά αρχεία ορισμού διεπαφών. Τα εργαλεία προγραμματισμού χρησιμοποιούν το περιβάλλον εκτέλεσης του COM+ για να εξάγουν metadata που περιγράφουν τις διεπαφές. Τα metadata είναι αρκετά για την αυτόματη δημιουργία proxies και stubs, κάτι που υποβοηθά τη διαδικασία κατασκευής και εγκατάστασης των συστατικών.

Επίσης το COM+ ορίζει ένα απλούστερο και πιο συμπαγές μοντέλο για τη δήλωση, την εγκατάσταση και τις Εκδόσεις των συστατικών. Αυτό το μοντέλο στηρίζεται στις

υπηρεσίες downloading του Win32, στα πακέτα του MTS, στο class store των Windows NT και στα διαχειριστικά εργαλεία του COM, και συνδυάζει όλα αυτά σε μια συνεπή και εύκολη στη χρήση υπηρεσία.

Το πακέτο (package) είναι μια απλή, εγκαταστάσιμη μονάδα κώδικα που περιέχει μία ή περισσότερες κλάσεις. Ένα πακέτο μπορεί επίσης να αντιπροσωπεύει μια έννοια όπως ένας χρήστης ή ένα μηχάνημα. Όλη η πληροφορία μπορεί να συντηρηθεί από τα εργαλεία του προγραμματιστή ή από το διαχειριστή του συστήματος. Δε χρειάζεται η συγγραφή κώδικα για τη δημιουργία πληροφορίας δηλώσεων (registration).

Η υπηρεσία δηλώσεων παρέχει έναν μηχανισμό για την παράκαμψη της έκδοσης μιας κλάσης που βρίσκεται σε ένα πακέτο. Αυτό βοηθά στην επίλυση ενός από τα πιο ενοχλητικά θέματα των εφαρμογών των συστατικών. Ας πούμε για παράδειγμα, ότι η εφαρμογή Α χρησιμοποιεί την έκδοση 1.0 του συστατικού Ψ και λειτουργεί καλά, αλλά η εφαρμογή Β εγκαθίσταται με την έκδοση 2.0 του συστατικού Ψ και επίσης λειτουργεί καλά, αλλά η εφαρμογή Α σταματά να δουλεύει. Με το περιβάλλον εκτέλεσης του COM+, μπορούν να χρησιμοποιηθούν και οι δύο Εκδόσεις του συστατικού Ψ, η έκδοση 1.0 για την εφαρμογή Α και η έκδοση 2.0 για την εφαρμογή Β.

Οι προγραμματιστές μπορούν εύκολα να εκμεταλλευτούν αυτές τις υπηρεσίες. Για καινούρια συστατικά θα πρέπει απλά να γραφούν οι κλάσεις, να δοθούν τα χαρακτηριστικά των κλάσεων και να αφεθεί το περιβάλλον εκτέλεσης του COM+ να κάνει τα υπόλοιπα. Η περισσότερη δουλειά για τη μεταφορά των παλιών συστατικών στο μοντέλο COM+ θα είναι στην ουσία η διαγραφή τμημάτων κώδικα. Τα συστατικά που θα προκύψουν θα είναι συστατικά COM και θα λειτουργούν με κάθε εργαλείο που μπορεί να χρησιμοποιήσει συστατικά COM, αλλά αυτά τα νέα συστατικά θα είναι πιο εύκολο να γραφτούν και να εγκατασταθούν από τα παραδοσιακά συστατικά COM.

6.4. Επικοινωνία με γεγονότα

Ένας **ΕΚΔΟΤΗΣ** (Publisher) είναι οποιοδήποτε πρόγραμμα κάνει τις COM κλήσεις που ξεκινούν γεγονότα και ένας **ΣΥΝΔΡΟΜΗΤΗΣ** (Subscriber) είναι ένα συστατικό COM+ που λαμβάνει τις COM κλήσεις που αντιπροσωπεύουν γεγονότα από έναν Εκδότη. Ο Συνδρομητής υλοποιεί μία διεπαφή ως ένας εξυπηρέτης COM. Ο Εκδότης κάνει κλήσεις σε αυτόν σαν ένας πελάτης COM [43].

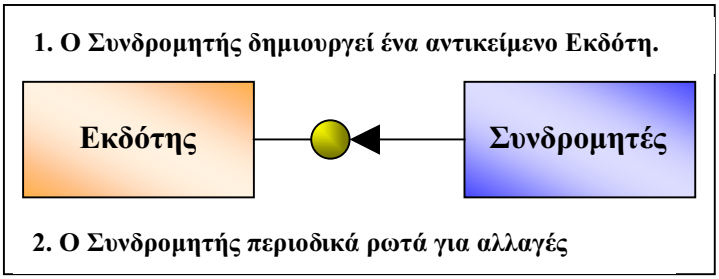
Η σύνδεση προγραμμάτων που παρέχουν πληροφορίες που αλλάζουν με το χρόνο (Εκδότες) με προγράμματα που θέλουν να λαμβάνουν ειδοποιήσεις για αυτές τις (Συνδρομητές) ήταν για καιρό μια πρόκληση στην κατασκευή κατανεμημένων εφαρμογών. Η υπηρεσία γεγονότων του COM+ παρέχει μια υποδομή που κάνει εύκολη την έκδοση και τη συνδρομή σε δεδομένα.

6.4.1. Έκδοση (Publishing) και Συνδρομή (Subscribing)

Ένα πρόγραμμα ανιχνεύει μια αλλαγή στον κόσμο που θεωρεί ότι τα άλλα προγράμματα πρέπει να τη γνωρίζουν. Ένα Εικονικό Πληκτρολόγιο βλέπει το πάτημα ενός πλήκτρου, ένα πρόγραμμα παρακολούθησης του χρηματιστηρίου βλέπει την αλλαγή σε μια τιμή μετοχής, ένα πρόγραμμα παρακολούθησης του καιρού βλέπει αλλαγές στις βαρομετρικές μετρήσεις από ένα μακρινό αισθητήρα ή ένα πρόγραμμα ιατρικής παρακολούθησης βλέπει ότι η πίεση του αίματος ενός ασθενή έχει υπερβεί την αποδεκτή τιμή. Κάπου αλλού óôïõ κόσμο υπάρχουν άλλα προγράμματα που θα ήθελαν να ξέρουν για αυτές τις αλλαγές: Ένας συνθέτης ομιλίας θέλει να εκφωνήσει τη λέξη που αντιστοιχεί στο πλήκτρο του Εικονικού

Πληκτρολογίου που πατήθηκε, ένα πρόγραμμα χαρτοφυλακίου που πρέπει να αγοράσει μια μετοχή όταν αυτή φτάσει σε μια συγκεκριμένη τιμή, ένα πρόγραμμα συναγερμού που λέει στις ψαρόβαρκες να επιστρέψουν στο λιμάνι ή ένα πρόγραμμα παρακολούθησης του ασθενή που δίνει σήμα στο σταθμό των νοσοκόμων ότι ο ασθενής χρειάζεται φάρμακο.

Οι έννοιες του πελάτη και του εξυπηρέτη γίνονται ομιχλώδεις σε σενάρια σαν αυτά κι έτσι εισάγουμε τη νέα ονοματολογία. Προγράμματα που παρέχουν ειδοποιήσεις σε άλλα προγράμματα, όπως το Εικονικό Πληκτρολόγιο, τα καλούμε "Εκδότες". Εφαρμογές όπως ο Συνθέτης Ομιλίας που λαμβάνουν δεδομένα από Εκδότες και δρουν ανάλογα με αυτά, τις καλούμε "Συνδρομητές". Όταν ο Εκδότης ανιχνεύει μια αλλαγή που αφορά τον Συνδρομητή, προκύπτει το πρόβλημα της ειδοποίησης του Συνδρομητή. Ο απλούστερος τρόπος είναι να "ρωτά" κάθε τόσο (polling) ο Συνδρομητής τον Εκδότη. Στην ορολογία του COM+, ο Εκδότης θα μπορούσε να δώσει στο Συνδρομητή μία διεπαφή και ο Συνδρομητής θα μπορούσε να καλεί περιοδικά μια μέθοδο σε αυτήν τη διεπαφή για να δει αν έχουν συμβεί αλλαγές, όπως φαίνεται στο Σχήμα 16.



Σχήμα 16: "Ερωτήσεις" (polling) του Συνδρομητή

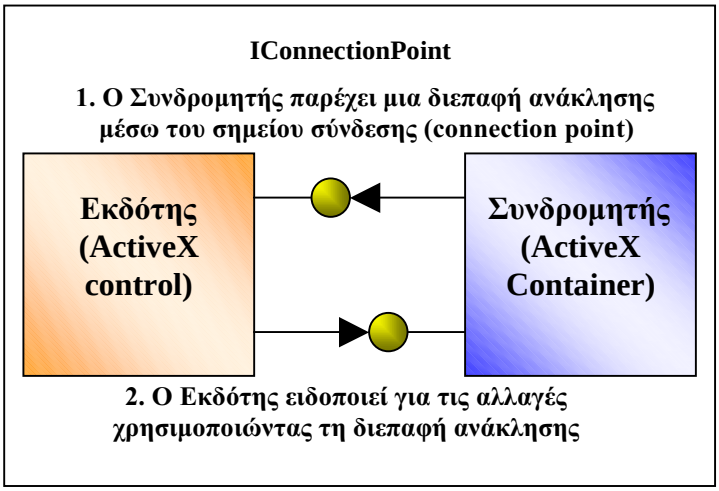
Αυτή η στρατηγική είναι απλή óôïõ προγραμματισμό, αλλά δεν είναι σωστή για πολλούς λόγους. Πρώτον ο Συνδρομητής ξοδεύει πολύ χρόνο και ενέργεια ρωτώντας "Υπάρχουν αλλαγές;". Ο Εκδότης επίσης σπαταλά χρόνο και προσπάθεια απαντώντας "Όχι δεν υπάρχουν". Κάτι τέτοιο είναι απαράδεκτο από άποψη απόδοσης και ταχύτητας για την εφαρμογή.

Δεύτερον, η τεχνική του polling εισάγει κάποια αναπόφευκτη καθυστέρηση μεταξύ του χρόνου που λαμβάνει χώρα η αλλαγή και του χρόνου που ο Συνδρομητής ρωτά. Κατά μέσο όρο, αυτή η καθυστέρηση είναι ίση με το μισό του μεσοδιαστήματος του polling (μεταξύ των ερωτήσεων). Καθώς μεγαλώνει των μεσοδιάστημα των ερωτήσεων για να ξοδεύονται λιγότεροι κύκλοι του επεξεργαστή, αυξάνει η καθυστέρηση. Αυτή η καθυστέρηση δεν είναι μόνο ένα μειονέκτημα από μόνη της, αλλά και το γεγονός ότι δεν είναι ντετερμινιστική (διαφέρει μεταξύ των κλήσεων) εισάγει ένα πρόβλημα στο σχεδιασμό των συστημάτων.

Θα θέλαμε ο Εκδότης να ξεκινά τη διαδικασία ειδοποίησης όταν ανιχνεύει ενδιαφέρουσες αλλαγές óôïõ κόσμο. Στην ορολογία του COM, ο Συνδρομητής εφοδιάζει τον Εκδότη με μία διεπαφή και ο Εκδότης καλεί μια μέθοδο σε αυτή όταν συμβαίνει κάτι ενδιαφέρον. Αυτή είναι η προσέγγιση που χρησιμοποιούν τα ActiveX controls για να πυροδοτήσουν γεγονότα στις οντότητες που τα περιέχουν (container), όπως φαίνεται στο Σχήμα 17. Εδώ το control είναι ο Εκδότης και η μονάδα που το περιέχει (container) είναι ο Συνδρομητής.

Αυτό καλείται στενά συνδεδεμένο γεγονός (tightly coupled event). Ο Συνδρομητής ξέρει ακριβώς από ποιον Εκδότη να ζητήσει ειδοποιήσεις (η μονάδα που τον περιέχει γνωρίζει

το CLSID, ή τον αναγνωριστή της κλάσης ή το control) και το μηχανισμό για να συνδεθεί σε αυτό (τις διεπαφές `IConnectionPointContainer` και `IConnectionPoint` που εκτίθενται από το control). Ένα στενά συνδεδεμένο γεγονός δεν ταιριάζει όσο θα θέλαμε στη φιλοσοφία και το σχεδιασμό του ΟΔΥΣΣΕΑ μια και στη δική μας εφαρμογή απαιτείται να μην ξέρουν τα συστατικά την ύπαρξη το ένα του άλλου. Θα πρέπει να υπάρχει μεγαλύτερη ελευθερία και λιγότερες απαιτήσεις για την επικοινωνία μεταξύ τους.



Σχήμα 17: Ανακλήσεις ActiveX

Για να λειτουργήσει ο μηχανισμός των γεγονότων, ένα στενά συνδεδεμένο γεγονός απαιτεί να τρέχουν συνέχεια και ο Εκδότης και ο Συνδρομητής. Και οι δύο πλευρές πρέπει να τρέχουν όταν ο Συνδρομητής (container) δίνει στον Εκδότη (control) τη διεπαφή ανάκλησης και επίσης όταν ο Εκδότης καλεί τη μέθοδο στη διεπαφή του Συνδρομητή. Στη δική μας περίπτωση, οι Συνδρομητές και οι Εκδότες δεν θα πρέπει να είναι τόσο στενά συνδεδεμένοι μια και κανείς δεν εγγυάται ούτε καν τη ύπαρξή τους σε κάθε στιγμή. Θα χρειάζεται Συνδρομητές και Εκδότες να μπορούν να αντικαθίστανται ή να καταργούνται χωρίς να απαιτείται επαναμεταγλώτισση ή επανεγκατάσταση όλης της εφαρμογής. Επίσης θα παρουσιαστούν και ανάγκες ασύγχρονης επικοινωνίας συστατικών, κάτι που δεν είναι συμβατό με την απαίτηση να τρέχουν ταυτόχρονα οι Εκδότες και οι Συνδρομητές.

Ένα άλλο πρόβλημα με τα στενά συνδεδεμένα γεγονότα είναι ότι ο Συνδρομητής πρέπει να ξέρει τον ακριβή μηχανισμό που ένας Εκδότης απαιτεί για να εγκαθιδρύει συνδρομές και αυτός ο μηχανισμός μπορεί να διαφέρει ριζικά από τον έναν Εκδότη όôïõ άλλον. Για παράδειγμα, τα ActiveX controls χρησιμοποιούν το μηχανισμό του IConnectionPoint για να στήσουν το κύκλωμα ανακλήσεων και να παραδίδουν ειδοποιήσεις για τα γεγονότα τους. Ένας OLE server χρησιμοποιεί τη μέθοδο Advise στη διεπαφή IOleObject για να φτιάξει ένα κύκλωμα ανακλήσεων και να παραδίδει ειδοποιήσεις γεγονότων που σχετίζονται με embedding. Θα ήταν πολύ καλό να τυποποιηθεί ένας μηχανισμός σύνδεσης για τους Εκδότες και τους Συνδρομητές και να μπορεί κανείς να χρησιμοποιεί αυτόν το μηχανισμό σύνδεσης διαχειριστικά αντί να πρέπει να γράφει κώδικα για να έχει πρόσβαση σε αυτόν.

Το τρίτο πρόβλημα του κλασσικού μηχανισμού γεγονότων είναι ότι δεν περιλαμβάνει μηχανισμούς για φιλτράρισμα ή αναχαίτιση (interception). Για παράδειγμα, θα μπορούσαμε να έχουμε πολλά συστατικά μιας εφαρμογής να λειτουργούν παράλληλα και να πρέπει να μοιράζονται τον ίδιο χώρο μηνυμάτων και τον ίδιο μηχανισμό για την

επικοινωνία μεταξύ τους. Αν ένα Εικονικό Πληκτρολόγιο ως Εκδότης ανακοίνωνε τη λέξη που ο χρήστης επέλεξε, και αυτή η λέξη αφού επεξεργαζόταν από ένα Συστατικό Μετάφρασης έπρεπε να παραδοθεί στον Συνθέτη Ομιλίας, πως θα ήξερε αυτό το τελευταίο συστατικό αν η λέξη είναι η πρωτογενής ή η μεταφρασμένη για να την εκφωνήσει; Επίσης το Συστατικό Μετάφρασης πως θα απέφευγε να μεταφράσει την ίδια τη έξοδό του αφού αυτή θα την έβρισκε πάλι σαν είσοδο όôïõ κοινό χώρο μηνυμάτων επικοινωνίας; Θα έπρεπε κάθε συστατικό να έχει την ικανότητα φιλτραρίσματος των γεγονότων που δέχεται ή που στέλνει και ιδανικά αυτή η διαδικασία θα έπρεπε να ρυθμίζεται διαχειριστικά.

Μια λύση σε αυτό το πρόβλημα θα ήταν η αποθήκευση της πληροφορίας για το ταίριασμα των Εκδοτών και των Συνδρομητών εξωτερικά, αντί αυτή να αποθηκεύεται μέσα στα ίδια τα προγράμματα. ο Εκδότης θα διατηρούσε μια εξωτερική βάση δεδομένων που θα περιείχε μια λίστα με τα διάφορα γεγονότα για τα οποία ξέρει πώς να στέλνει ειδοποιήσεις. Οι Συνδρομητές θα διάβαζαν αυτή τη λίστα και θα επέλεγαν τα γεγονότα για τα οποία θέλουν να ειδοποιούνται. Ο Εκδότης θα διατηρούσε επίσης ένα είδος βάσης δεδομένων συνδρομών, κάτι σαν τις λίστες ηλεκτρονικής αλληλογραφίας, που θα περιείχε τα CLSIDs των Συνδρομητών που θέλουν να ειδοποιούνται για το κάθε γεγονός. Θ έπρεπε για όλα αυτά να υπάρχουν διαχειριστικά εργαλεία ή τα ίδια τα προγράμματα θα έπρεπε να ξέρουν πώς να διαχειρίζονται τις εγγραφές αυτών των βάσεων δεδομένων. Όταν λοιπό ο Εκδότης θέλει να πυροδοτήσει ένα γεγονός, συνδέεται με τη βάση δεδομένων, βρίσκει τα CLSIDs όλων των ενδιαφερόμενων Συνδρομητών, δημιουργεί ένα νέο αντικείμενο για κάθε μια από τις ενδιαφερόμενες κλάσεις και καλεί μια μέθοδο σε αυτό το αντικείμενο. Ένα τέτοιο σύστημα θα ονομαζόταν σύστημα χαλαρά συνδεδεμένων γεγονότων αντί για στενά συνδεδεμένων γιατί η πληροφορία για το ποιος Συνδρομητής θέλει να ακούει ποιον Εκδότη θα εσυντηρείτο σε μια κεντρική βάση δεδομένων αντί να είναι κλεισμένη στα ίδια τα προγράμματα.

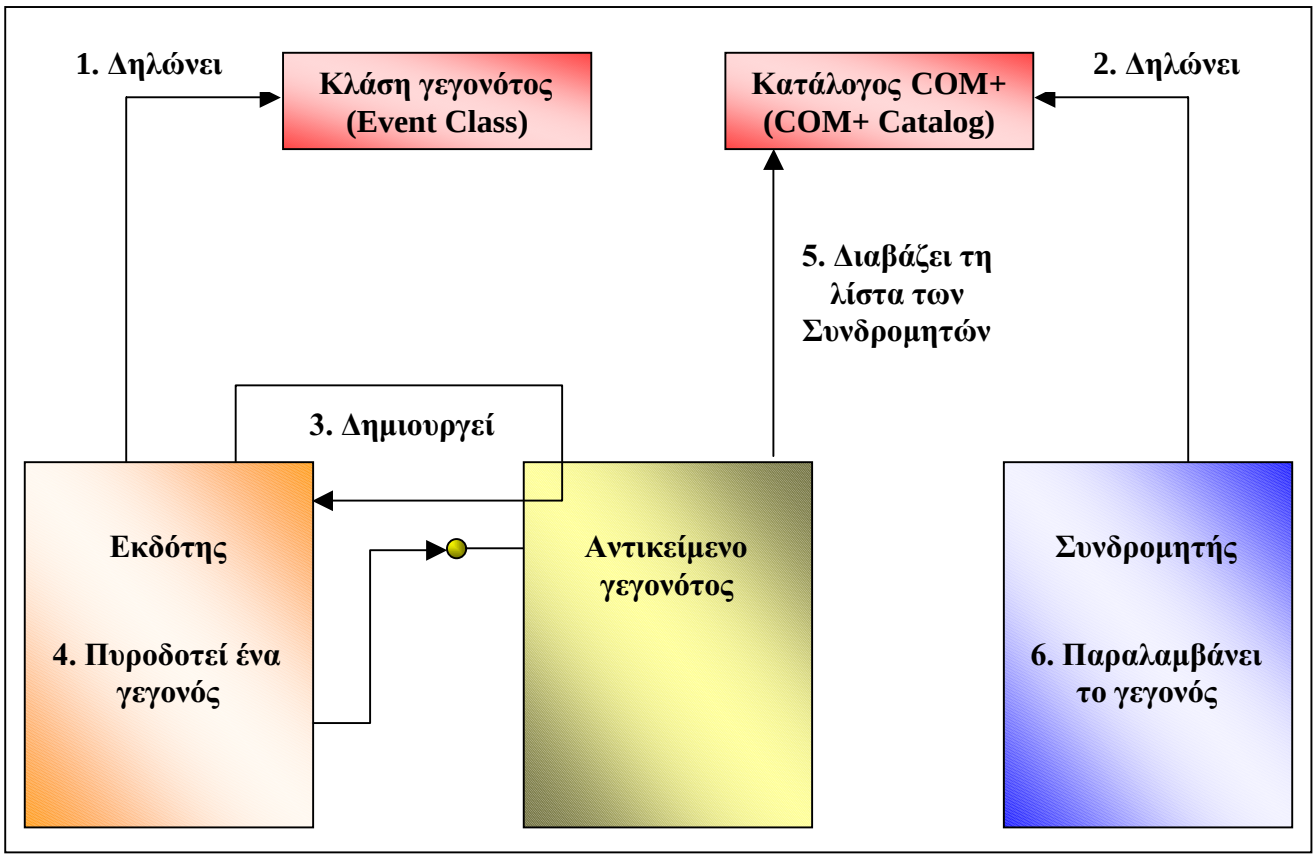
Αυτή η πολλά υποσχόμενη σχεδιαστική προσέγγιση έχει δύο παγίδες. Πρώτα, θα έπρεπε να αναπτύξουμε και να συντηρήσουμε τη βάση των γεγονότων και των συνδρομών και να γράψουμε όλον τον κώδικα για τον μηχανισμό πυροδότησης γεγονότων από την πλευρά του Εκδότη και για τα εργαλεία διαχείρισης. Δεύτερον, ακόμα και αν αναπτύσσαμε αυτήν την υποδομή η διαδικασία συνδρομών μας θα ήταν ακόμη διαφορετική από οποιουδήποτε άλλου κατασκευαστή. Οι Συνδρομητές θα έπρεπε να ξέρουν όχι μόνο τις συγκεκριμένες τεχνικές που απαιτούνται για να έχουν συνδρομή στα γεγονότα μας, αλλά και τους διάφορους μηχανισμούς που απαιτούνται από οποιονδήποτε άλλον Εκδότη τον οποίο θέλουν να ακούν και να είναι συμβατοί με αυτόν. Αυτό που θα θέλαμε πραγματικά θα ήταν να εκμεταλλευτούμε έναν τέτοιο μηχανισμό που θα τον προσέφερε το λειτουργικό σύστημα. Με αυτόν τον τρόπο θα είχαμε να γράψουμε πολύ λίγο κώδικα και η διαδικασία συνδρομών για κάθε κατασκευαστή θα ήταν η ίδια.

6.5. Υπηρεσίες γεγονότων του COM+

Η υπηρεσία γεγονότων του COM+ [43], είναι η υπηρεσία του λειτουργικού συστήματος που ασχολείται με το ταίριασμα και τη σύνδεση Εκδοτών και Συνδρομητών. Η αρχιτεκτονική της φαίνεται στο Σχήμα 18.

Χρησιμοποιώντας την ορολογία του Σχήματος, ένα γεγονός αντιπροσωπεύει μία κλήση σε μία μέθοδο σε μία COM διεπαφή, που ξεκίνησε από έναν Εκδότη και παραδόθηκε από την υπηρεσία γεγονότων στον σωστό Συνδρομητή ή Συνδρομητές. Ο Εκδότης είναι οποιοδήποτε πρόγραμμα κάνει τις κλήσεις που ξεκινούν γεγονότα και ο Συνδρομητής είναι ένα συστατικό COM+ που παραλαμβάνει τις COM κλήσεις που αντιστοιχούν σε γεγονότα

από έναν Εκδότη. Ο Συνδρομητής υλοποιεί μια διεπαφή ως ένας εξυπηρέτης COM. Ο Εκδότης κάνει κλήσεις σε αυτήν ως πελάτης COM. Η μόνη αλλαγή από το κλασσικό COM είναι η υπηρεσία γεγονότων στη μέση, η οποία παρακολουθεί ποιοι Συνδρομητές θέλουν να παραλαμβάνουν τις κλήσεις και κατευθύνει τις κλήσεις σε αυτούς τους Συνδρομητές χωρίς να απαιτείται οποιαδήποτε γνώση για τον Εκδότη.



Σχήμα 18: Αρχιτεκτονική της υπηρεσίας Γεγονότων του COM+

Η πραγματοποίηση μιας απλής κλήσης γεγονότος λέγεται πυροδότηση του γεγονότος. Μπορούμε επίσης να χρησιμοποιούμε και τον όρο "Έκδοση" ως συνώνυμο της πυροδότησης.

Η σύνδεση μεταξύ ενός Εκδότη και ενός Συνδρομητή αντιπροσωπεύεται από μία κλάση γεγονότος (event class). Μια κλάση γεγονότος είναι ένα συστατικό COM+ που συντίθεται από το σύστημα γεγονότων και περιέχει τις διεπαφές και τις μεθόδους που θα καλέσει ένας Εκδότης για να πυροδοτήσει γεγονότα και που ο Συνδρομητής πρέπει να υλοποιήσει αν θέλει να παραλαμβάνει γεγονότα. Οι διεπαφές και οι μέθοδοι που παρέχονται από μια κλάση γεγονότων λέγονται διεπαφές γεγονότων και μέθοδοι γεγονότων. Πληροφορούμε το COM+ ποιες διεπαφές και μεθόδους θέλουμε να περιέχει μια κλάση γεγονότων, δίνοντάς του μια βιβλιοθήκη τύπων (type library). Οι κλάσεις γεγονότων αποθηκεύονται στον κατάλογο του COM+ (COM+ Catalog), και τοποθετούνται εκεί είτε από τους ίδιους τους Εκδότες, είτε από διαχειριστικά εργαλεία.

Ένας Συνδρομητής εκφράζει την επιθυμία του να παραλαμβάνει γεγονότα από έναν Εκδότη, δηλώνοντας στην υπηρεσία γεγονότων του COM+ μια συνδρομή. Η συνδρομή

είναι μια δομή δεδομένων που δίνει στην υπηρεσία γεγονότων πληροφορίες σχετικά με τον παραλήπτη ενός γεγονότος. Καθορίζει για τον Συνδρομητή, από ποια κλάση γεγονότων, και από ποια διεπαφή ή μέθοδο μέσα σε αυτήν την κλάση γεγονότων, θα λαμβάνει κλήσεις. Οι συνδρομές αποθηκεύονται στον κατάλογο του COM+, και τοποθετούνται εκεί είτε από τους ίδιους τους Συνδρομητές, είτε από διαχειριστικά εργαλεία. Οι μόνιμες (persistent) συνδρομές επιζούν μετά από μια επανεκκίνηση του συστήματος, ενώ οι προσωρινές (transient) δεν επιζούν.

Όταν ένας Εκδότης θέλει να πυροδοτήσει ένα γεγονός, χρησιμοποιεί τις τυπικές συναρτήσεις δημιουργίας αντικειμένων, όπως η `CoCreateInstance` ή η `CreateObject`, για να δημιουργήσει ένα αντικείμενο από την επιθυμητή κλάση γεγονότων. Αυτό το αντικείμενο, γνωστό ως αντικείμενο γεγονότος, περιέχει την υλοποίηση του συστήματος γεγονότων για τη ζητούμενη διεπαφή. Στη συνέχεια, ο Εκδότης καλεί τη μέθοδο του γεγονότος που θέλει να πυροδοτήσει προς τους Συνδρομητές. Μέσα στην υλοποίηση της διεπαφής από το σύστημα γεγονότων, το σύστημα γεγονότων κοιτάζει τον κατάλογο του COM+ και βρίσκει όλους τους Συνδρομητές που έχουν δηλώσει συνδρομές σε αυτήν τη διεπαφή και μέθοδο. Όταν αυτή η διαδικασία ολοκληρωθεί, το σύστημα γεγονότων συνδέεται με κάθε Συνδρομητή (χρησιμοποιώντας οποιοδήποτε συνδυασμό άμεσης δημιουργίας, monikers ή queued components) και καλεί τη ζητούμενη μέθοδο.

Επειδή πολύ συχνά περισσότεροι από έναν Συνδρομητές θέλουν ειδοποιήσεις για κάθε γεγονός, οι μέθοδοι των γεγονότων δεν μπορούν να χρησιμοποιούν οποιουδήποτε τύπου παραμέτρους εξόδου, αλλά πρέπει να επιστρέφουν μόνο `HRESULT`s επιτυχίας ή αποτυχίας. Με αυτόν τον τρόπο, κάθε πελάτης COM μπορεί ουσιαστικά να γίνει Εκδότης και κάθε συστατικό COM+ μπορεί να γίνει Συνδρομητής. Κανείς από τους δύο δεν χρειάζεται να ξέρει τίποτα για τις διαδικασίες που θα διεξαχθούν από το σύστημα των γεγονότων για να γίνει η σύνδεση.

6.6. Συνδρομές

Η συνδρομή είναι μια δομή δεδομένων που βρίσκεται στον κατάλογο COM+. Οι ιδιότητες μιας συνδρομής είναι διαθέσιμες μέσω της διεπαφής `I•ventSubscription`. Πολλές από τις ιδιότητες μπορούν να τροποποιηθούν μέσω των Component Services, αλλά άλλες δεν είναι διαθέσιμες μέσω αυτής της διεπαφής χρήστη. Πρέπει να γράψει κανείς το δικό του διαχειριστικό πρόγραμμα για να προσπελάσει αυτές τις ιδιότητες.

Οι συνδρομές συναντώνται σε δύο τύπους, μόνιμες και προσωρινές. Οι μόνιμες συνδρομές βρίσκονται στον κατάλογο COM+ και συνεχίζουν να υπάρχουν μετά από επανεκκινήσεις του συστήματος. Υπάρχουν ανεξάρτητα από τον κύκλο ζωής του αντικειμένου – Συνδρομητή. Ένα πρόγραμμα Συνδρομητής δημιουργεί συχνά μια μόνιμη συνδρομή όταν εγκαθίσταται, και αφαιρεί τη συνδρομή όταν απεγκαθίσταται.

Όταν ένας Εκδότης κάνει μια κλήση σε ένα αντικείμενο γεγονότος, αυτό ψάχνει όλες τις μόνιμες συνδρομές στον κατάλογο και δημιουργεί ένα καινούριο στιγμιότυπο για κάθε κλάση Συνδρομητή. Η διαδικασία δημιουργίας μπορεί να γίνει είτε άμεσα είτε μέσω moniker. Το ποιο αντικείμενο Συνδρομητή θα δημιουργηθεί καθορίζεται από τη ρύθμιση της ιδιότητας της συνδρομής `SubscriberCLSID` ή την `SubscriberMoniker`. Το αντικείμενο Συνδρομητή που δημιουργείται από μια μόνιμη συνδρομή απελευθερώνεται πάντα μετά από κάθε κλήση γεγονότος, ανεξάρτητα από την επιτυχία ή αποτυχία της κλήσης και από τον αν ο Εκδότης απελευθερώνει το αντικείμενο γεγονότος.

Μια προσωρινή συνδρομή απαιτεί οι κλήσεις να γίνονται σε ένα συγκεκριμένο υπάρχον αντικείμενο Συνδρομητή. Οι προσωρινές συνδρομές αποθηκεύονται επίσης στον κατάλογο COM+, αλλά δεν συνεχίζουν να υπάρχουν μετά από μια επανεκκίνηση του συστήματος. Η διεπαφή χρήσης του Component Services δεν προβλέπει τη δημιουργία προσωρινών συνδρομών. Πρέπει το πρόγραμμα του Συνδρομητή να δημιουργήσει μόνο του τη συνδρομή χρησιμοποιώντας τις διαχειριστικές διεπαφές του COM+.

Μια προσωρινή συνδρομή εγκαθίσταται προσθέτοντας μια νέα συνδρομή στο σύστημα γεγονότων και θέτοντας στην ιδιότητα `SubscriberInterface` τη διεπαφή `IUnknown` του αντικειμένου Συνδρομητή. Σε αυτήν την περίπτωση, ο μηχανισμός των γεγονότων δεν δημιουργεί ένα νέο στιγμιότυπο του αντικειμένου Συνδρομητή όταν πυροδοτεί ένα γεγονός, αλλά χρησιμοποιεί αυτό που έχει δοθεί. Το σύστημα γεγονότων θα κρατά ένα reference count για το αντικείμενο Συνδρομητή έως ότου ένα διαχειριστικό πρόγραμμα ή το ίδιο το αντικείμενο Συνδρομητή αφαιρέσει τον εαυτό του από το σύστημα γεγονότων.

Επειδή δεν περιλαμβάνουν επαναλαμβανόμενες δημιουργίες και καταστροφές αντικειμένων, οι προσωρινές συνδρομές είναι πιο αποδοτικές από τις μόνιμες συνδρομές, αν και αίρουν όλα τα θέματα που έχουν να κάνουν με τον κύκλο ζωής και αποφεύγονται με τις μόνιμες συνδρομές.

Οποιαδήποτε συνδρομή μπορεί να απενεργοποιηθεί. Αυτό γίνεται θέτοντας την ιδιότητα Enabled της συνδρομής στην τιμή FALSE. Μια απενεργοποιημένη συνδρομή δεν καλείται ποτέ από το σύστημα γεγονότων.

7. Λεπτομερής Σχεδιασμός

Έχοντας μελετήσει και επιλέξει τις τεχνολογίες αιχμής που θα αποτελέσουν τη βάση του Πλαισίου Σχεδιασμού και Ανάπτυξης Βοηθημάτων Διαπροσωπικής Επικοινωνίας "ΟΔΥΣΣΕΑΣ", και έχοντας υπ' όψη τη γενική περιγραφή του πλαισίου που προηγήθηκε, ακολουθούν κάποιες λεπτομέρειες του σχεδιασμού της υλοποίησής του. Μέθοδοι σχεδιασμού λογισμικού και εφαρμογών όπως η Ενοποιημένη Γλώσσα Μοντελοποίησης (Unified Modeling Language – UML) [60], και το εργαλείο μοντελοποίησης Microsoft Visual Modeler [34] που είναι ενσωματωμένο στο περιβάλλον προγραμματισμού Visual Studio της Microsoft, ήταν από τις μεθόδους που εξετάστηκαν ως υποψήφιες για την τεχνική (formal) περιγραφή των συστατικών που επρόκειτο να υλοποιηθούν. Τελικά, λόγω του μικρού μεγέθους των συστατικών αυτών αποφασίστηκε ότι δεν υπάρχει η ανάγκη χρησιμοποίησης τέτοιων μεθόδων σχεδιασμού και μοντελοποίησης. Τέτοιες μέθοδοι χρησιμοποιούνται μάλλον σε πιο μεγάλα προγράμματα και ολοκληρωμένες εφαρμογές όπου ο σχεδιασμός και η τεχνική περιγραφή δεν είναι δυνατό να γίνει με απλά λόγια ή σχήματα και απαιτείται πιο τεχνική (formal) μέθοδος για να δοθεί η ακριβής εικόνα του τι υλοποιείται. Σε τέτοιες περιπτώσεις τα εργαλεία αυτά επιτρέπουν την ακριβή περιγραφή, την αποφυγή ασαφειών, την απλοποίηση και την τυποποίηση της ορολογίας για την περιγραφή των λειτουργικοτήτων και της δομής των προγραμμάτων. Σε περιπτώσεις όμως σαν τη δική μας, όπου πρέπει να περιγραφούν μικρά και απλά τμήματα λογισμικού, θεωρείται ασύμφορη η χρήση τέτοιων εργαλείων. Η περιγραφή για παράδειγμα ενός δοκιμαστικού πρότυπου συστατικού, όπως αυτά που υλοποιούνται στα πλαίσια του ΟΔΥΣΣΕΑ, μπορεί να γίνει με ακρίβεια, σαφώς και χωρίς πολυπλοκότητα με τη χρήση απλού κειμένου και σχημάτων που αποσαφηνίζουν τη λειτουργικότητα και τη δομή του.

Εκτός από τα συστατικά που παράγονται στα πλαίσια του ΟΔΥΣΣΕΑ, ο σχεδιασμός του πλαισίου αφορά και σε θέματα όπως ο καθορισμός των ομάδων χρηστών του ΟΔΥΣΣΕΑ, ο κύκλος ζωής του Βοηθήματος και η διαδικασία εγκατάστασής του. Όλα αυτά περιγράφονται στις ενότητες που ακολουθούν.

7.1. Ομάδες Χρηστών του ΟΔΥΣΣΕΑ

Σχεδιαστές – Κατασκευαστές – Προγραμματιστές Βοηθημάτων Διαπροσωπικής επικοινωνίας και Συστατικών τους

Πωλητές – Συναρμολογητές – Ολοκληρωτές Βοηθημάτων Διαπροσωπικής Επικοινωνίας

Σχήμα 19: Ομάδες χρηστών του πλαισίου ΟΔΥΣΣΕΑΣ.

Το Πλαίσιο Σχεδιασμού και Ανάπτυξης Εφαρμογών ΟΔΥΣΣΕΑΣ σχεδιάστηκε για δύο συγκεκριμένες κύριες ομάδες χρηστών: τους **Προγραμματιστές** και τους **Πωλητές** (Σχήμα 19). Για την τρίτη ομάδα χρηστών που είχε προταθεί από την αρχιτεκτονική ATIC του έργου ACCESS [5], [6], τους **Θεραπευτές** ή βοηθούς των ΑΜΕΑ, το πλαίσιο ΟΔΥΣΣΕΑΣ είναι εντελώς **διάφανο**. Αν και οι Θεραπευτές απολαμβάνουν τα ευεργετικά αποτελέσματα του ΟΔΥΣΣΕΑ, όπως την ποικιλία των συστατικών, τις δυνατότητες προσαρμογής των Βοηθημάτων Επικοινωνίας και το χαμηλό κόστος τους, δεν έρχονται σε άμεση με το ίδιο το πλαίσιο και τις υπηρεσίες του και δε θεωρούνται άμεσοι χρήστες του.

Θα αναλυθεί στη συνέχεια πως ακριβώς ο ΟΔΥΣΣΕΑΣ απευθύνεται στις δύο κύριες ομάδες χρηστών και ποιες είναι οι υπηρεσίες που τους προσφέρει. Σε επίπεδο Διαδικτύου, ο σχεδιασμός των υπηρεσιών του ΟΔΥΣΣΕΑ για αυτούς τους χρήστες, αλλά και για όλους τους ενδιαφερόμενους περιγράφεται στην ενότητα "ο ΟΔΥΣΣΕΑΣ στο Διαδίκτυο".

7.1.1. Προγραμματιστές

Αυτή η ομάδα χρηστών είναι τα πρόσωπα που σχεδιάζουν και υλοποιούν συστατικά Βοηθημάτων Διαπροσωπικής Επικοινωνίας, Μπορεί να πρόκειται για μεμονωμένα πρόσωπα ή για ολόκληρες εταιρίες ανάπτυξης λογισμικού ή ολοκληρωμένων συστημάτων επικοινωνίας.

Ο ΟΔΥΣΣΕΑΣ αρχικά πρέπει να παρέχει σε αυτή την ομάδα χρηστών μια σειρά από τυποποιήσεις σχετικές με την ανάπτυξη λογισμικού. Πρόκειται να προταθούν τρεις τυποποιήσεις της Microsoft και είναι οι εξής:

- Τυποποίηση για την κατασκευή εφαρμογών των Windows 2000, που αφορά σε τεχνικές απαιτήσεις των εφαρμογών αυτών και απατήσεις για τη διεπαφή χρήσης της.
- Τυποποίησης του Μοντέλου Αντικειμένων και Συστατικών (COM) που αφορά στην τεχνική προγραμματισμού συστατικών.
- Τυποποίηση των Υπηρεσιών Συστατικών (COM+) που αφορά στις υπηρεσίες υποστήριξης της ανάπτυξης εφαρμογών με συστατικά και είναι ενσωματωμένες στο λειτουργικό σύστημα.

Όλες αυτές οι τυποποιήσεις αναλύονται περισσότερο στο παραδοτέο Π3.1 που αφορά την Υλοποίηση του πλαισίου ΟΔΥΣΣΕΑΣ [53].

Στη συνέχεια ο ΟΔΥΣΣΕΑΣ πρέπει δίνει στους Προγραμματιστές πιο συγκεκριμένες τεχνικές οδηγίες για την κατασκευή συστατικών. Αυτές οι οδηγίες δεν είναι τόσο γενικές όσο οι τυποποιήσεις που προαναφέρθηκαν, αλλά είναι συγκεκριμένα για το πλαίσιο ΟΔΥΣΣΕΑΣ. Κύριος στόχος αυτών των οδηγιών είναι να παράγονται συστατικά που είναι συμβατά μεταξύ τους και μπορούν να επικοινωνήσουν ακολουθώντας συγκεκριμένο πρωτόκολλο επικοινωνίας και συγκεκριμένες προδιαγραφές. Μεγάλο μέρος αυτών των προδιαγραφών καθώς και το πρωτόκολλο επικοινωνίας καθορίζονται από τον ΟΔΥΣΣΕΑ και περιλαμβάνονται στην Υλοποίησή του. Παραδείγματα τέτοιων οδηγιών είναι ότι τα συστατικά θα πρέπει να είναι αρχεία βιβλιοθηκών δυναμικής σύνδεσης (Dynamic Link Libraries – DLLs) [58], ώστε να είναι δυνατή η ενσωμάτωσή τους στον κατάλογο συστατικών των Windows 2000, να είναι φυσικά συμβατά με το λειτουργικό σύστημα Windows 2000, να χρησιμοποιούν συγκεκριμένο μοντέλο νημάτων (threading model), να χρησιμοποιούν καθορισμένη δομή και ονοματολογία στις κλάσεις και τα αντικείμενα που περιέχουν, καθώς και να υλοποιούν συγκεκριμένες διεπαφές για την επικοινωνία τους με τα άλλα συστατικά.

Οι διεπαφές επικοινωνίας που αναφέρθηκαν, ο σχεδιασμός και η υλοποίησή τους, είναι μία ακόμη υποχρέωση του πλαισίου ΟΔΥΣΣΕΑΣ. Ενσωματώνουν το προκαθορισμένο πρωτόκολλο επικοινωνίας μεταξύ των συστατικών του ΟΔΥΣΣΕΑ και επιτρέπουν τη χρήση της υποδομής και των υπηρεσιών του λειτουργικού συστήματος για τη διεκπεραίωση της επικοινωνίας αυτής. Πρέπει να αναπτυχθεί ένα αρχείο dll, το οποίο να περιέχει τη διεπαφή αυτή και να είναι τέτοιο ώστε να μπορεί να δηλωθεί ως μία κλάση γεγονότων (Event Class) στην υπηρεσία συστατικών των Windows 2000.

Τέλος, στα πλαίσια του ΟΔΥΣΣΕΑ πρέπει να υλοποιηθούν συστατικά δοκιμών και αναφοράς για τους προγραμματιστές. Αυτά χρησιμεύουν ως πρότυπα για τον τρόπο που πρέπει να εφαρμόζονται οι τυποποιήσεις και να ακολουθούνται οι οδηγίες του ΟΔΥΣΣΕΑ και πρέπει να διατίθεται ελεύθερα ο κώδικάς τους ως δείγμα. Επίσης, τα συστατικά αυτά πρέπει να εξυπηρετούν σκοπούς δοκιμών και ελέγχου της συμβατότητας των συστατικών που αναπτύσσουν οι προγραμματιστές με το πλαίσιο ΟΔΥΣΣΕΑΣ.

7.1.2. Πωλητές

Αυτή η ομάδα χρηστών περιλαμβάνει τους ανθρώπους που διαθέτουν τα Βοηθήματα Διαπροσωπικής Επικοινωνίας στους τελικούς τους χρήστες ΑΜΕΑ. Οι χρήστες αυτοί του ΟΔΥΣΣΕΑ δεν είναι απλά "Πωλητές" αλλά πρέπει να διαθέτουν τα κατάλληλα προσόντα και γνώσεις ώστε να μπορούν να αναγνωρίζουν σωστά τις ανάγκες των τελικών χρηστών των Βοηθημάτων, να τις αντιστοιχούν σε υπηρεσίες του συστήματος και να βρίσκουν τα κατάλληλα συστατικά του Βοηθήματος που παρέχουν αυτές τις υπηρεσίες. Τέλος πρέπει να έχουν και γνώσεις χρήστη ηλεκτρονικού υπολογιστή, μια και θα πρέπει να εκτελούν διαχειριστικές ενέργειες για την εγκατάσταση και ρύθμιση των Βοηθημάτων.

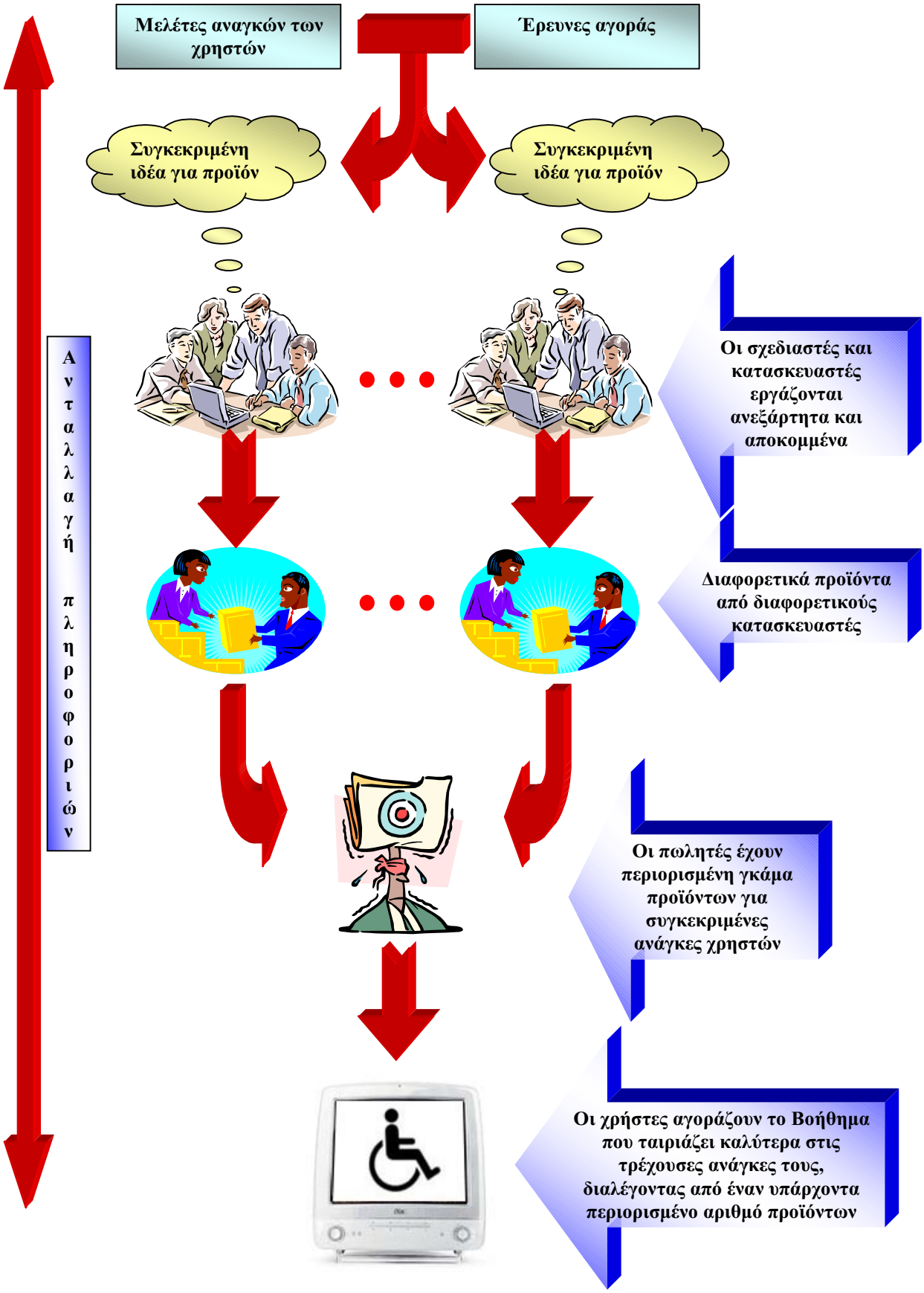
Ο ΟΔΥΣΣΕΑΣ για τους Πωλητές έχει τη μορφή ενός Εγχειριδίου Χρήσης για τις Υπηρεσίες των Συστατικών (Component Services) των Windows 2000 και τις συγκεκριμένες δραστηριότητες που πρέπει αυτοί να εκτελούν, ώστε να εγκαθίστανται και να λειτουργούν σωστά τα Βοηθήματα Επικοινωνίας. Το Εγχειρίδιο Χρήσης περιλαμβάνεται στην Τεχνική Έκθεση για την Υλοποίηση του Πλαισίου [53].

7.2. Κύκλος ζωής του Βοηθήματος Επικοινωνίας

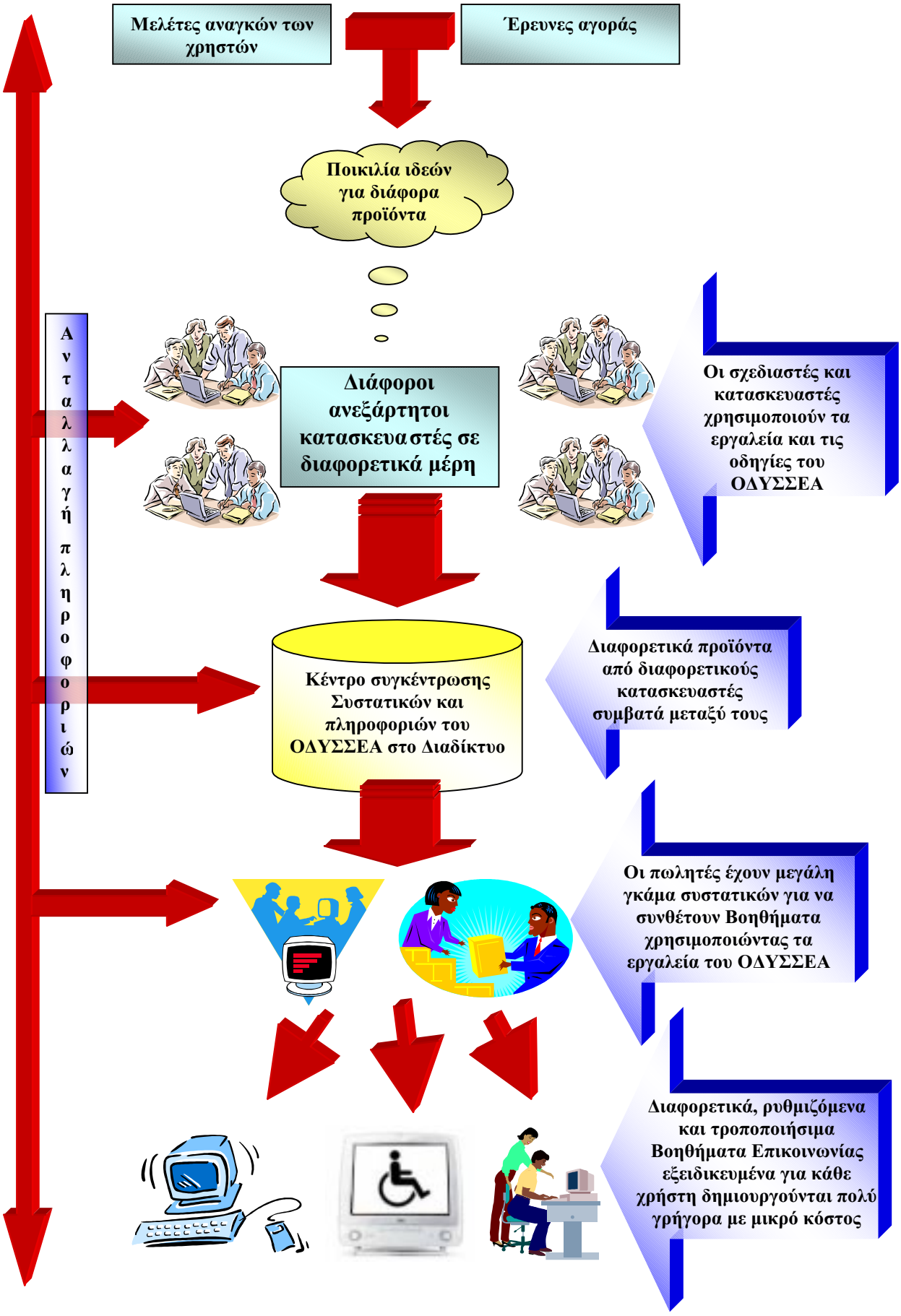
Μέχρι τώρα, ο παραδοσιακός τρόπος ανάπτυξης Βοηθημάτων Διαπροσωπικής Επικοινωνίας και γενικά εφαρμογών για υπολογιστές, ήταν οι προγραμματιστές να εργάζονται απομονωμένοι και χωριστά. Κάθε κατασκευαστής είχε μια ομάδα σχεδιαστών και προγραμματιστών οι οποίοι υλοποιούσαν μια ιδέα (ή ιδέες) για προϊόντα που βασίζονταν στην κατάλληλη έρευνα αγοράς και τη μελέτη των αναγκών των χρηστών. Η Υλοποίηση των προϊόντων προχωρούσε μέχρι το τέλος για την παραγωγή ολοκληρωμένων Βοηθημάτων και μονολιθικών εφαρμογών. Το εξειδικευμένο υλικό (συσκευές εισόδου εξόδου, μηχανήματα επικοινωνίας) και το λογισμικό προέρχονταν για κάθε σύστημα από τον ίδιο κατασκευαστή με μικρή ή καθόλου ευελιξία και προβλήματα συμβατότητας με προϊόντα άλλων κατασκευαστών (Σχήμα 20).

Οι κατασκευαστές υλοποιούν τα προϊόντα τους που στη συνέχεια, φτάνουν στον τελικό χρήστη μέσω ενός πωλητή. Ο χρήστης με ειδικές ανάγκες πρέπει να αγοράσει το προϊόν που ταιριάζει καλύτερα στις ανάγκες του. Υπάρχει δυνατότητα ρυθμίσεων μόνο στο επίπεδο του τελικού χρήστη και στο βαθμό που προσφέρεται αυτή η δυνατότητα από τον κατασκευαστή. Οποιαδήποτε τροποποίηση και ανάγκη αλλαγής συνήθως σημαίνει αγορά ενός νέου προϊόντος ή ανάπτυξή του από την αρχή. Τέλος η ανταλλαγή πληροφοριών

σχετικά με το σχεδιασμό του προϊόντος (feedback) γίνεται μεταξύ τελικών πελατών και κατασκευαστών. Ο ρόλος των πωλητών είναι υποβαθμισμένος στο επίπεδο του απλού μεσάζοντα για τη διάθεση των προϊόντων.



Σχήμα 20: Παραδοσιακή διαδικασία ανάπτυξης και διάθεσης Βοηθημάτων Επικοινωνίας



Σχήμα 21: Η προσέγγιση του ΟΔΥΣΣΕΑ για τον κύκλο ζωής των Βοηθημάτων

Από την άλλη πλευρά, στην προσέγγιση του ΟΔΥΣΣΕΑ (Σχήμα 21), ενώ οι αρχικές ιδέες αφήνονται στους κατασκευαστές, ενθαρρύνεται ένας μεγαλύτερος βαθμός συνεργασίας που στοχεύει στη μεγαλύτερη ευελιξία και επαναχρησιμοποίηση των σχεδιασμών και των προϊόντων. Επίσης διευκολύνεται η μεγαλύτερη προσαρμοστικότητα και συμβατότητα των διαφόρων υλοποιήσεων. Η Προσέγγιση του ΟΔΥΣΣΕΑ, σχετικά με τον κύκλο ζωής των προϊόντων έχει αρκετές ομοιότητες με την προσέγγιση της αρχιτεκτονικής ATIC του έργου ACCESS [5], [6], αλλά έχει και μια βασική διαφορά: τον ενεργό ρόλο του Διαδικτύου στον κύκλο ζωής. Στην ουσία η αρχιτεκτονική ATIC προέβλεπε στη θέση του Κέντρου Συγκέντρωσης Συστατικών και Πληροφοριών του ΟΔΥΣΣΕΑ στο Διαδίκτυο, μια πιο παραδοσιακή «δεξαμενή» συστατικών που θα μπορούσε να είναι είτε το ίδιο το κατάστημα του πωλητή, είτε ένα κέντρο συγκέντρωσης συστατικών που θα ελεγχόταν από ειδικούς διαχειριστές (απαιτούσε προσωπικό) και δεν είχε καμία σχέση με το Διαδίκτυο. Η εισαγωγή του Διαδικτύου στον κύκλο ζωής των προϊόντων που προτείνει ο ΟΔΥΣΣΕΑΣ κάνει εξαιρετικά πιο ευέλικτη τη διαδικασία συλλογής των συστατικών, αλλά και τη διαδικασία εντοπισμού τους. Τέλος μια άλλη διαφορά που ήδη αναφέρθηκε είναι ότι, ενώ οι θεραπευτές ήταν χρήστες του ATIC, και επομένως περιλαμβάνονταν στον κύκλο ζωής των προϊόντων, στο πλαίσιο ΟΔΥΣΣΕΑΣ δεν περιλαμβάνονται. Το αποτέλεσμα της διαφάνειας του ΟΔΥΣΣΕΑ για τους θεραπευτές-βοηθούς των ΑΜΕΑ είναι η μείωση των γνώσεων που χρειάζεται να έχουν σχετικά με το πλαίσιο και δυνατότητα εκμετάλλευσής του χωρίς να απαιτείται η «χρήση» του.

Οι κατασκευαστές υλοποιούν συστατικά, τα οποία συναρμολογούν οι πωλητές και τα ολοκληρώνουν σε πλήρως λειτουργικά Βοηθήματα Επικοινωνίας σύμφωνα με τις ανάγκες των συγκεκριμένων χρηστών. Οι χρήστες ΑΜΕΑ έχουν πολύ μεγαλύτερες πιθανότητες να αποκτήσουν ένα σύστημα κομμένο και ραμμένο στα μέτρα τους. Οι δυνατότητες ρυθμίσεων αυξάνονται μια και αυτές δεν γίνονται πια μόνο από τους κατασκευαστές και τους τελικούς χρήστες σύμφωνα με τις δυνατότητες που τους δίνονται από τους κατασκευαστές, αλλά επεκτείνονται και στους πωλητές συστημάτων. Ο σχεδιασμός γίνεται με βάση την επαναχρησιμοποίηση και ενθαρρύνεται η επικοινωνία και η ανταλλαγή πληροφοριών (feedback), μεταξύ τόσο των χρηστών και των κατασκευαστών όσο και μεταξύ των πωλητών και του πλαισίου ΟΔΥΣΣΕΑΣ και όλων των συνδυασμών των προηγουμένων. Μεγάλη διαφορά υπάρχει στην προσπάθεια που χρειάζεται για τη μεταβολή του σχεδιασμού του Βοηθήματος και την τροποποίησή του. Μπορεί ο κατασκευαστή να μη χρειαστεί καν να παρέμβει για να επιτευχθεί κάτι τέτοιο μια και υπάρχουν οι δυνατότητες προσθαφαίρεσης λειτουργιών και υπηρεσιών από το Βοήθημα με την επέμβαση μόνο του πωλητή ή και του ίδιου του χρήστη και του βοηθού/θεραπευτή του. Σε καμία περίπτωση δεν χρειάζεται επανασχεδιασμός ολόκληρου του Βοηθήματος και κάθε νέα ανάγκη μπορεί να αντιμετωπιστεί με κατασκευή νέων συστατικών ή πρόσθεση/αντικατάσταση ήδη υπαρχόντων.

7.3. Συστατικά Βοηθήματος Επικοινωνίας

Τα συστατικά ενός Βοηθήματος Επικοινωνίας που παράγεται από το πλαίσιο ΟΔΥΣΣΕΑΣ πρέπει να έχουν κάποια βασικά χαρακτηριστικά εκτός από αυτά που προκύπτουν από τη συμμόρφωση στις γενικές τυποποιήσεις και οδηγίες που δίνει ο ΟΔΥΣΣΕΑΣ. Πρώτον πρέπει να υλοποιούν τη λειτουργικότητα είτε του Εκδότη δεδομένων, είτε του Συνδρομητή σε Εκδότες είτε ένα συνδυασμό των δύο.

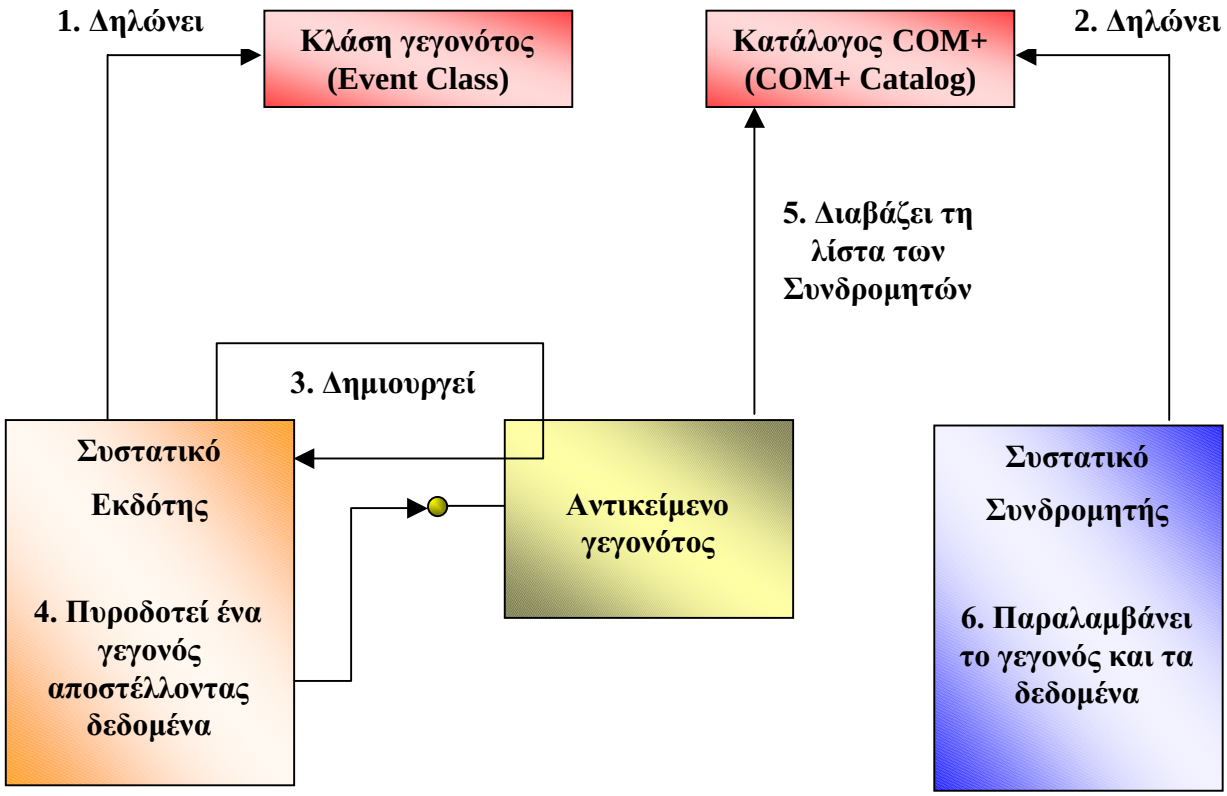
Πολύ βασικό είναι το γεγονός ότι κατά τον προγραμματισμό αυτών των συστατικών πρέπει να λαμβάνεται η κατάλληλη μέριμνα για να μπορούν στη συνέχεια αυτά τα συστατικά να επικοινωνήσουν μεταξύ τους και να εκμεταλλευτούν την υποδομή του λειτουργικού συστήματος για την επικοινωνία και τη λειτουργία τους. Η Επικοινωνία βασίζεται σε συγκεκριμένο πρωτόκολλο και διεπαφές που περιγράφονται στη συνέχεια.

Η λειτουργία των συστατικών σύμφωνα με όλα όσα έχουν περιγραφεί μέχρι τώρα βασίζεται στην υπηρεσία γεγονότων του COM+ που περιγράφτηκε στο σχετικό κεφάλαιο και συνοψίζεται στο παρακάτω Σχήμα 22.

Συστατικά Εκδότες θεωρούνται τα συστατικά "εισόδου". Μπορούν να είναι Εικονικά Πληκτρολόγια με Συμβατικά Πλήκτρα (γράμματα αλφαβήτου), Εικονικά Πληκτρολόγια με Εναλλακτικά Πλήκτρα (σύμβολα BLISS, εικόνες, λέξεις), Επεξεργαστές Κειμένου, Αναγνωριστές Ομιλίας, Προγράμματα Παραλαβής Ηλεκτρονικής Αλληλογραφίας.

Συστατικά Συνδρομητές θεωρούνται τα συστατικά "εξόδου". Μπορούν να είναι Συνθέτες Ομιλίας, Απεικονίσεις Κειμένου στην Οθόνη, Προγράμματα Αποστολής Ηλεκτρονικής Αλληλογραφίας.

Συστατικά που υλοποιούν και τους δύο ρόλους (του Εκδότη και του Συνδρομητή) θεωρούνται τα "ενδιάμεσα" συστατικά που έχουν και είσοδο και έξοδο. Τέτοια μπορεί να είναι Μεταφραστές, Συστατικά Πρόβλεψης Λέξεων, Επεξεργαστές Προτάσεων (μετατροπή ασύντακτων προτάσεων σε συντεταγμένες) και γενικά συστατικά που εκτελούν οποιαδήποτε επεξεργασία σε δεδομένα που λαμβάνουν και τα αποστέλλουν προς την έξοδο.



Σχήμα 22: Μοντέλο λειτουργίας των συστατικών του Βοηθήματος Επικοινωνίας

7.4. Πρωτόκολλο επικοινωνίας των συστατικών

Το πρωτόκολλο επικοινωνίας μεταξύ των συστατικών ενός βοηθήματος Διαπροσωπικής Επικοινωνίας που βασίζεται στο πλαίσιο ΟΔΥΣΣΕΑΣ, περιλαμβάνει μηνύματα που αποστέλλονται και λαμβάνονται με τη μεσολάβηση του λειτουργικού συστήματος Windows 2000 και συγκεκριμένα με την εκμετάλλευση της τεχνολογίας COM+ και της υπηρεσίας γεγονότων.

Τα μηνύματα αυτά αποστέλλονται και παραλαμβάνονται μέσω της υλοποίησης από τα συστατικά συγκεκριμένων διεπαφών της Υπηρεσίας Γεγονότων (Event Classes) [27] που θα περιγραφούν στην επόμενη ενότητα. Το βασικό που πρέπει να περιέχουν αυτά τα μηνύματα είναι τα δεδομένα του χρήστη. Το πλαίσιο ΟΔΥΣΣΕΑΣ προβλέπει για τα δεδομένα του χρήστη ένα μόνο τύπο δεδομένων και αυτός είναι οι αλφαριθμητικές σειρές (strings). Αυτό θέτει σε όλα τα συστατικά την προδιαγραφή ότι, άσχετα με τη λειτουργικότητα του καθενός, την επεξεργασία που εκτελεί το καθένα στα δεδομένα και τους εσωτερικούς τύπους δεδομένων που μπορεί να υποστηρίζουν το καθένα για τον εαυτό του, θα πρέπει υποχρεωτικά να επικοινωνούν μεταξύ τους μόνο με βάση αλφαριθμητικές σειρές.

Αυτές οι αλφαριθμητικές σειρές προβλέπεται να περιέχουν την καθαρή πληροφορία του χρήστη, η οποία μπορεί να είναι ένας απλός χαρακτήρας, μια λέξη, μία πρόταση ή ένα ολόκληρο έγγραφο. Άσχετα με το αν ο χρήστης επικοινωνεί με συμβατική γλώσσα, με συμβατικούς αλφαριθμητικούς χαρακτήρες ή με εικόνες, με εναλλακτικές γλώσσες επικοινωνίας, όπως οι εικονικές γλώσσες, τα συστατικά (εισόδου) έχουν την υποχρέωση να μετατρέπουν την οποιαδήποτε είσοδο του χρήστη σε αλφαριθμητικές σειρές (για παράδειγμα, να αντιστοιχίζουν ένα σύμβολο στην έννοια του) πριν τη στείλουν σε επόμενα συστατικά.

Λόγω της διαφορετικής επεξεργασίας που εφαρμόζεται στα είδη αλφαριθμητικών σειρών που αναφέρθηκαν, δηλαδή σε χαρακτήρες, σε λέξεις, προτάσεις και έγγραφα, ο ΟΔΥΣΣΕΑΣ διαχωρίζει ρητά τα είδη αυτά, σαν να ήταν διαφορετικοί τύποι δεδομένων. Δεν είναι ένας διαχωρισμός που θα έχει κάποιο προγραμματιστικό αντίκτυπο, όπως για παράδειγμα ο διαχωρισμός σε γλώσσες προγραμματισμού μεταξύ τύπων δεδομένων, όπως οι αριθμοί και οι αλφαριθμητικές σειρές. Είναι μάλλον ένας λογικός και εννοιολογικός διαχωρισμός που οφείλουν να σεβαστούν οι κατασκευαστές συστατικών για να επιτευχθεί η σωστή και αποτελεσματική επεξεργασία των δεδομένων του χρήστη. Ακολουθεί ένας πίνακας (Πίνακας 5) που περιλαμβάνει αυτό το διαχωρισμό στο επίπεδο πρωτοκόλλου επικοινωνίας και διαφωτιστικά παραδείγματα που δείχνουν γιατί γίνεται αυτός ο διαχωρισμός.

Τύπος δεδομένων	**Μπορεί να είναι έξοδος από**	**Μπορεί να είναι είσοδος σε**
Χαρακτήρας	• Εικονικό πληκτρολόγιο χαρακτήρων • Επεξεργαστή κειμένου	• Συστατικό πρόβλεψης λέξεων • Συστατικό απεικόνισης κειμένου • Συστατικό για σύγχρονη απομακρυσμένη συνομιλία

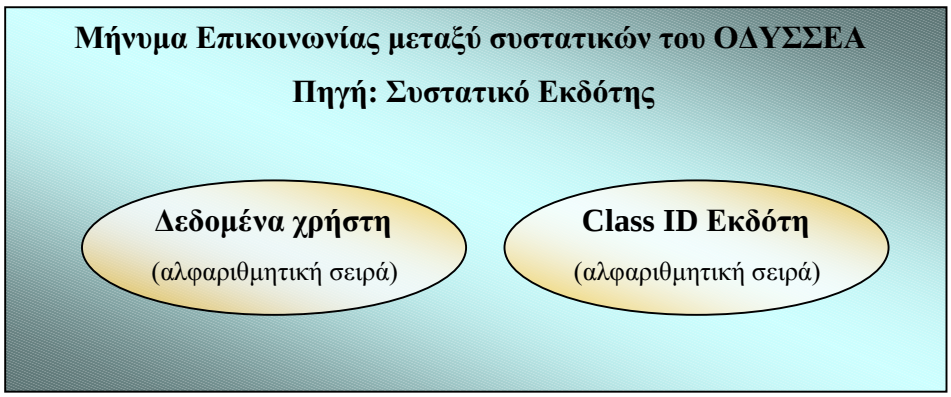
Λέξη	• Εικονικό πληκτρολόγιο λέξεων • Εικονικό πληκτρολόγιο συμβόλων • Επεξεργαστή κειμένου • Συστατικό αναγνώρισης ομιλίας • Συστατικό μετάφρασης λέξεων	• Συστατικό απεικόνισης κειμένου • Συστατικό για σύγχρονη απομακρυσμένη συνομιλία • Συνθέτη ομιλίας • Συστατικό μετάφρασης λέξεων
Πρόταση	• Εικονικό πληκτρολόγιο συμβόλων • Επεξεργαστή κειμένου • Συστατικό μετάφρασης • Συστατικό μετάφρασης προτάσεων	• Συστατικό απεικόνισης κειμένου • Συστατικό για σύγχρονη απομακρυσμένη συνομιλία • Συνθέτη ομιλίας • Συστατικό μετάφρασης προτάσεων
Έγγραφο	• Επεξεργαστή κειμένου • Συστατικό σύνθεσης μηνυμάτων ηλεκτρονικής αλληλογραφίας • Συστατικό μετάφρασης κειμένων	• Συνθέτη ομιλίας • Συστατικό αποστολής ηλεκτρονικών μηνυμάτων • Συστατικό μετάφρασης κειμένων

Πίνακας 5: Πίνακας τύπων δεδομένων μηνυμάτων επικοινωνίας μεταξύ συστατικών

Τα δεδομένα του χρήστη δεν είναι το μόνο συστατικό των μηνυμάτων που πρέπει να ανταλλάσσονται μεταξύ των συστατικών ενός Βοηθήματος που βασίζεται στον ΟΔΥΣΣΕΑ. Πρέπει να ανταλλάσσεται και κάποια πληροφορία για το συγχρονισμό και την τοποθέτηση των συστατικών σε σειρά. Αυτό συμβαίνει γιατί έτσι όπως σχεδιάζεται η επικοινωνία μεταξύ των συστατικών σαν ένα είδος "Πίνακα Ανακοινώσεων" στον οποίο μπορούν να αφήνουν μηνύματα οι Εκδότες και από τον οποίο μπορούν να διαβάζουν μηνύματα οι Συνδρομητές, υπάρχει ο κίνδυνος "μπερδέματος" των μηνυμάτων. Σίγουρα σε διάφορες περιπτώσεις υπάρχει οι ανάγκη κάποιοι Συνδρομητές να λαμβάνουν Μηνύματα συγκεκριμένων Εκδοτών και όχι άλλων εκτός από αυτούς. Επίσης μπορεί να απαιτείται μηνύματα κάποιον Εκδοτών να λαμβάνονται μόνο από μερικούς ή έναν συγκεκριμένο Συνδρομητή. Παρουσιάζεται λοιπόν η ανάγκη για "φιλτράρισμα" των μηνυμάτων, μια δυνατότητα που δίνεται από τις Υπηρεσίες Γεγονότων του COM+ [27], [28]. Για να επιτευχθεί όμως αυτό το φιλτράρισμα χρειάζεται να είναι γνωστό κάποιο στοιχείο που αποκαλύπτει την ταυτότητα τουλάχιστο των Εκδοτών δεδομένων. Αυτό, από τη φύση του ΟΔΥΣΣΕΑ δεν υποστηρίζεται μια και ένα βασικό χαρακτηριστικό του πλαισίου είναι η "απομόνωση" των συστατικών μεταξύ τους και η έλλειψη γνώσης της ύπαρξης του ενός για το άλλο, πόσο μάλιστα της ταυτότητάς του.

Έτσι λοιπόν πάρθηκε η απόφαση τα μηνύματα μεταξύ των συστατικών να περιέχουν εκτός από τα δεδομένα του χρήστη και κάποιο στοιχείο για την ταυτότητα του αποστολέα. Αυτό το στοιχείο πρέπει να χαρακτηρίζει με μοναδικό τρόπο τον κάθε Εκδότη. Τέτοιο στοιχείο είναι το Class ID που έχει κάθε κλάση συστατικού. Είναι μια σειρά αριθμών και χαρακτήρων που δημιουργείται βάση κατάλληλου αλγορίθμου από το λειτουργικό σύστημα κατά τη δήλωση μιας κλάσης σε αυτό και ανήκει στην κλάση για όλη τη διάρκεια της ζωής της, μια και ενσωματώνεται στο αρχείο που την περιέχει. Όταν σε ένα εργαλείο ή γλώσσα προγραμματισμού, ο προγραμματιστής δημιουργεί μια κλάση, αυτόματα ανατίθεται σε αυτήν αυτό το μοναδικό Class ID. Όταν η κλάση ή το πρόγραμμα που την περιέχει εγκατασταθεί ακόμη και σε ένα άλλο σύστημα από αυτό στο οποίο δημιουργήθηκε, το Class ID της παραμένει το ίδιο στο νέο σύστημα. Είναι λοιπόν ένα στοιχείο της ταυτότητας μιας κλάσης Εκδότη, που μπορεί να το γνωρίζει και ο προγραμματιστής που δημιουργεί την κλάση, αλλά και ο Πωλητής που την ενσωματώνει μαζί με το συστατικό που την περιέχει σε ένα Βοήθημα Επικοινωνίας.

Έτσι, το μήνυμα επικοινωνίας μεταξύ δύο συστατικών περιέχει δύο πεδία δεδομένων. Και τα δύο είναι αλφαριθμητικές σειρές και το ένα πρέπει να περιέχει τα δεδομένα του χρήστη σε οποιονδήποτε τύπο από αυτούς που αναφέρθηκαν και το άλλο να περιέχει το Class ID του αποστολέα ή του Εκδότη (Σχήμα 23). Η μορφή του μηνύματος πέρα από τη δομή των πεδίων (παραμέτρων) που περιέχει δεν μας ενδιαφέρει από σχεδιαστική άποψη, μια και όλα τα αναλαμβάνει το λειτουργικό και η μορφή δε γίνεται φανερή ούτε καν στον προγραμματιστή.



Σχήμα 23: Δομή ενός μηνύματος επικοινωνίας μεταξύ συστατικών του ΟΔΥΣΣΕΑ

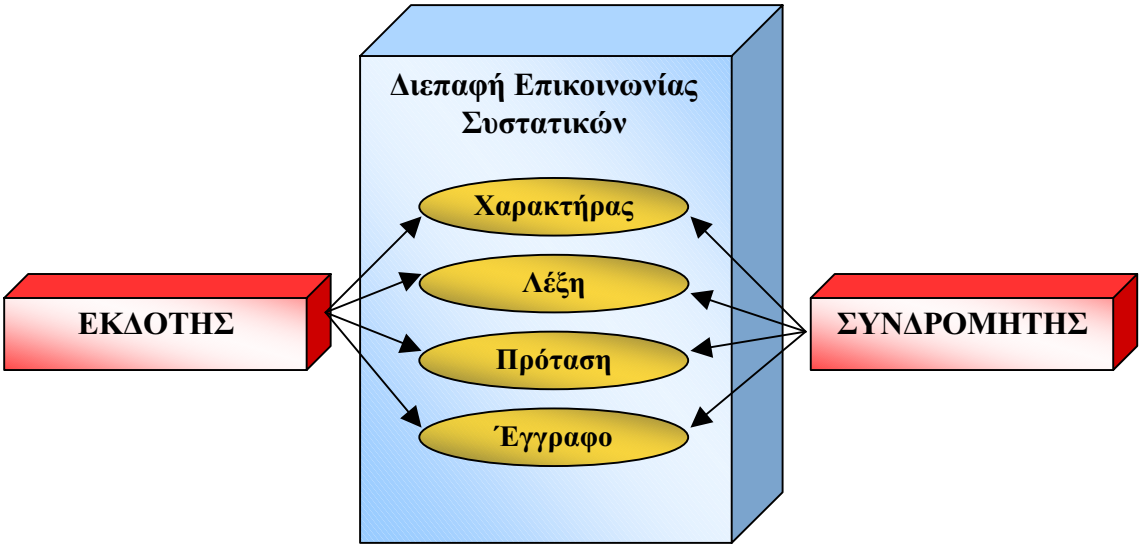
7.5. Διεπαφή επικοινωνίας των συστατικών

Όπως έχει περιγραφεί, η κλάση γεγονότων είναι απλά μια δήλωση διεπαφών και μεθόδων, χωρίς καμία υλοποίηση. Στην ουσία περιέχει άδειες κλάσεις και ρουτίνες, στις οποίες καθορίζονται μόνο οι μεταβλητές (arguments) και τα ονόματα των κλάσεων και των ρουτινών. Η υλοποίηση στα πλαίσια της υπηρεσίας γεγονότων γίνεται από τον εκάστοτε Συνδρομητή που έχει δηλώσει ότι υλοποιεί την συγκεκριμένη διεπαφή. Από τη μεριά του Εκδότη, η σχέση του με την κλάση γεγονότων είναι απλά ότι καλεί κάποιες από τις μεθόδους της. Η βασική λειτουργικότητα της κλάσης γεγονότων σε συνδυασμό με την υπηρεσία γεγονότων είναι ότι, όταν ένας Εκδότης καλεί μια μέθοδο σε ένα αντικείμενο της διεπαφής, τότε η υπηρεσία γεγονότων αναλαμβάνει να καλέσει την αντίστοιχη

υλοποιημένη μέθοδο των συνδρομητών σε αυτήν τη διεπαφή, απομονώνοντας έτσι τους Εκδότες από τους Συνδρομητές.

Η συγκεκριμένη διεπαφή που σχεδιάστηκε στα πλαίσια του ΟΔΥΣΣΕΑ, αποτελείται από τέσσερις κλάσεις, οι οποίες εξυπηρετούν την επικοινωνία με τέσσερις αντίστοιχους τύπους δεδομένων (Σχήμα 24). Ο ΟΔΥΣΣΕΑΣ έχει σχεδιαστεί ώστε να υποστηρίζει τους εξής τύπους δεδομένων: **Χαρακτήρας**, **Λέξη, Πρόταση, Έγγραφο.**

Έτσι τα δεδομένα που ανταλλάσσονται μεταξύ των συστατικών του ΟΔΥΣΣΕΑ είναι πάντα αλφαριθμητικές σειρές (strings), σε τέσσερις διαφορετικές διεπαφές, δηλαδή τέσσερις διαφορετικές κλάσεις γεγονότων, ανάλογα με το μέγεθος και τη λειτουργικότητα των δεδομένων. Προγραμματιστικά δεν υπάρχει καμία διαφορά μεταξύ των τεσσάρων τύπων δεδομένων εκτός από το όνομα των αντίστοιχων κλάσεων. Ο διαχωρισμός είναι καθαρά σε λογικό και λειτουργικό επίπεδο. Αυτό σημαίνει ότι τίποτα δεν εμποδίζει κάποιον να στείλει μια ολόκληρη πρόταση στη διεπαφή του χαρακτήρα, αλλά δίνεται αυστηρή οδηγία να μη γίνεται κάτι τέτοιο. Επίσης είναι φανερό ότι τέτοιος έλεγχος δεν θα ήταν εφικτός προγραμματιστικά, γιατί δεν θα επιτρεπόταν, για παράδειγμα στο άρθρο «ο» να αποσταλεί ως λέξη μια που αποτελεί έναν μόνο χαρακτήρα. Σε λογικό επίπεδο όμως μπορεί να αποτελεί και ολόκληρη λέξη.

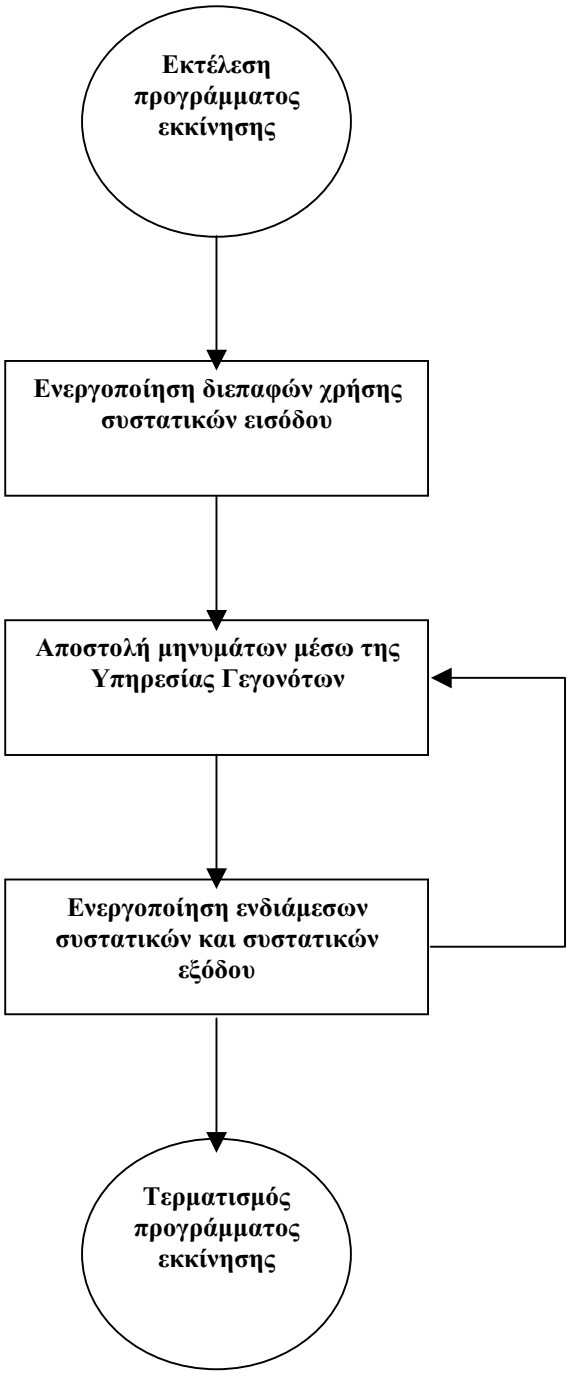


Σχήμα 24: Η διεπαφή για την επικοινωνία μεταξύ των συστατικών του ΟΔΥΣΣΕΑ.

7.6. Πρόγραμμα εκκίνησης του Βοηθήματος Επικοινωνίας

Όπως αναφέρθηκε, προβλέπεται από τον ΟΔΥΣΣΕΑ όλα τα συστατικά που συνθέτουν ένα Βοήθημα Διαπροσωπικής Επικοινωνίας, πακετάρονται σε μορφή αρχείων βιβλιοθηκών δυναμικής σύνδεσης (DLLs) [58]. Είναι γνωστό ότι τέτοια αρχεία δεν είναι εκτελέσιμα, δεν μπορούν δηλαδή να ενεργοποιηθούν εκτός αν κάποιο άλλο πρόγραμμα τα καλέσει. Η μέθοδος ενεργοποίησης πολλών από αυτά τα αρχεία είναι η κλήση τους από την υπηρεσία γεγονότων του λειτουργικού συστήματος, κάτι που γίνεται αυτόματα και με διαφάνεια μόλις Εκδοθούν δεδομένα. Υπάρχουν όμως και συστατικά που θα πρέπει να έχουν κάποια

διεπαφή χρήστη που θα πρέπει να ενεργοποιείται κατά την εκκίνηση του Βοηθήματος πριν την έκδοση οποιονδήποτε μηνυμάτων και θα πρέπει τέλος πάντων να ενεργοποιείται κάποιο αρχικό συστατικό Εκδότης για να μπορούν να εκδοθούν μηνύματα που θα ενεργοποιήσουν και τα υπόλοιπα συστατικά.



Σχήμα 25: Βασική Λειτουργία Βοηθήματος Επικοινωνίας

Θα πρέπει λοιπόν στα πλαίσια του ΟΔΥΣΣΕΑ να αναπτυχθεί ένα αρχικό πρόγραμμα εκκίνησης του Βοηθήματος Διαπροσωπικής Επικοινωνίας, το οποίο θα εγκαθίσταται μαζί με τα συστατικά του Βοηθήματος στον υπολογιστή του τελικού χρήστη. Αυτό το

πρόγραμμα θα πρέπει να είναι φυσικά σε εκτελέσιμη μορφή ώστε να μπορεί να ενεργοποιηθεί με μια απλή ενέργεια του χρήστη (για παράδειγμα με διπλό κλικ σε ένα εικονίδιο) ή αυτόματα (αν εισαχθεί στο μενού προγραμμάτων εκκίνησης – Start up menu των Windows).

Λειτουργίες που θα πρέπει να εκτελεί αυτό το πρόγραμμα είναι οι εξής:

- Εκκίνηση Βοηθήματος Επικοινωνίας.
- Εύρεση των συστατικών που πρέπει να ενεργοποιηθούν κατά την εκκίνηση του Βοηθήματος.
- Ενεργοποίηση αυτών των συστατικών (διεπαφών χρήσεων Εκδοτών).
- Τερματισμός της λειτουργίας του Βοηθήματος Διαπροσωπικής Επικοινωνίας.

Η λειτουργία του Βοηθήματος διαπροσωπικής Επικοινωνίας με την προσθήκη αυτού του προγράμματος εκκίνησης συνοψίζεται στο Σχήμα 25.

7.7. Διαδικασία εγκατάστασης Βοηθήματος Επικοινωνίας

Βασικό προαπαιτούμενο για την εγκατάσταση ενός Βοηθήματος Διαπροσωπικής Επικοινωνίας είναι προφανώς να είναι γνωστή η σύνθεση και η επιθυμητή λειτουργικότητα του συστήματος. Η εγκατάσταση αυτή γίνεται μετά από προσεκτική επιλογή των διαθεσίμων συστατικών που ικανοποιούν τις απαιτήσεις του εκάστοτε τελικού χρήστη δηλαδή του Ατόμου με Ειδικές Ανάγκες [2], [3]. Χρειάζεται λοιπόν, από την πλευρά του προσώπου που εγκαθιστά την εφαρμογή, πολύ καλή γνώση τόσο των αναγκών του χρήστη, όσο και των χαρακτηριστικών των συστατικών που έχει στη διάθεσή του. Πρέπει να γίνει η αντιστοίχηση των απαιτήσεων των χρηστών σε υπηρεσίες του συστήματος και να εντοπιστούν τα συστατικά που προσφέρουν μέσω της λειτουργικότητάς τους αυτές τις υπηρεσίες είτε μόνα τους είτε σε συνεργασία με άλλα.

Η εγκατάσταση του Βοηθήματος Διαπροσωπικής επικοινωνίας έχει σχεδιαστεί έτσι ώστε να διενεργείται από τους ΠΩΛΗΤΕΣ των συστημάτων αυτών. Όλη η διαδικασία βασίζεται στη χρήση της Υπηρεσίας Συστατικών των Windows 2000 (Component Services). Όπως ήδη αναφέρθηκε, πρόκειται για μια διεπαφή χρήσης του λειτουργικού συστήματος όπου δίνεται η ευκαιρία στο χρήστη της να εγκαταστήσει και να ρυθμίσει διαχειριστικά εφαρμογές COM+ και να επέμβει στα περιεχόμενα του Καταλόγου των Συστατικών (Component Catalog). Με βάση την τεχνολογία και τις υπηρεσίες του συστήματος που παρέχονται από την ενσωμάτωση του COM+, ο Κατάλογος αυτός των Συστατικών (Component Catalog), παρέχει τη δυνατότητα να ικανοποιηθούν κάποιες από τις βασικότερες αρχικές απαιτήσεις που έπρεπε να καλύψει το πλαίσιο ΟΔΥΣΣΕΑΣ. Οι δυνατότητες που δίνει ο Κατάλογος των Συστατικών και αφορούν άμεσα το πλαίσιο ΟΔΥΣΣΕΑΣ είναι οι εξής:

- **Δυνατότητα οπτικής ομαδοποίησης των συστατικών μέσα σε μία "εφαρμογή".** Ο όρος εφαρμογή εδώ αναφέρεται σε μία σύνθεση από ένα ή περισσότερα συστατικά που λειτουργούν συνδυασμένα και σε συνεργασία για να επιτύχουν την επιθυμητή λειτουργικότητα ενός συστήματος λογισμικού. Η "εφαρμογή" είναι στην ουσία όλη η υποδομή που προσφέρει το λειτουργικό σύστημα και η τεχνολογία COM+, ώστε να επιτευχθεί η διαλειτουργικότητα του Μοντέλου Συστατικών. Μόλις στον Κατάλογο Συστατικών δηλωθεί από τον Πωλητή μια νέα εφαρμογή με το "ΑΕΝΕΑΣ", προετοιμάζεται από το λειτουργικό σύστημα όλη αυτή η υποδομή για να παρασχεθούν

στα συστατικά της εφαρμογής αυτή που θα προστεθούν στη συνέχεια όλες οι υπηρεσίες που απαιτούνται για την επικοινωνία και τη συνεργασία τους.

- **Δυνατότητα πρόσθεσης συστατικών λογισμικού στην εφαρμογή.** Τα συστατικά λογισμικού προστίθενται επίσης με διαχειριστικό τρόπο στην εφαρμογή συνθέτοντας έτσι το ολοκληρωμένο σύστημα. Είναι μεγάλο πλεονέκτημα ότι και σε ένα ήδη υπάρχον σύστημα μπορούν να προστεθούν συστατικά εκ των υστέρων. Αυτό σημαίνει ότι μπορεί να μεταβληθεί διαχειριστικά η λειτουργικότητα του Βοηθήματος Διαπροσωπικής Επικοινωνίας προσθέτοντας υπηρεσίες και λειτουργικότητες που αρχικά δεν υπήρχαν.
- **Δυνατότητα αφαίρεσης συστατικών από την εφαρμογή.** Η απαίτηση του ΟΔΥΣΣΕΑ για δυνατότητα και ευκολία στην μετατροπή του Βοηθήματος Επικοινωνίας συμπληρώνεται από αυτό το χαρακτηριστικό. Εκτός λοιπόν από τη δυνατότητα πρόσθεσης λειτουργικοτήτων στην εφαρμογή μπορεί ο πωλητής διαχειριστικά να αφαιρέσει και συστατικά με τις αντίστοιχες λειτουργικότητες όταν αυτές δεν χρειάζονται πια.
- **Δυνατότητα ρύθμισης της επικοινωνίας των συστατικών.** Μέσα από τη διαχείριση των Συνδρομών στην Υπηρεσία Γεγονότων του COM+, ο πωλητή έχει τη δυνατότητα να προσδιορίσει ποια συστατικά θα επικοινωνούν με ποια. Κάποια από τα συστατικά, σύμφωνα με το σχεδιασμό του ΟΔΥΣΣΕΑ είναι Εκδότες δεδομένων, κάποια άλλη είναι Συνδρομητές σε γεγονότα και λαμβάνουν τα δεδομένα αυτά και τέλος υπάρχουν συστατικά του Βοηθήματος Επικοινωνίας που έχουν τη λειτουργικότητα του Εκδότη και του Συνδρομητή ταυτόχρονα. Ο πωλητής έχει τη δυνατότητα μέσα από τις ρυθμίσεις των φίλτρων των Συνδρομητών να επιλέξει ποιον ή ποιους Εκδότες ενός συγκεκριμένου γεγονότος θα "ακούει" κάθε Συνδρομητής σε αυτό.

Για να εγκαταστήσει ένας Πωλητής ένα σύστημα Βοηθήματος Διαπροσωπικής Επικοινωνίας θα πρέπει να αλληλεπιδράσει με την Υπηρεσία Συστατικών των Windows 2000 και να χρησιμοποιήσει τις δυνατότητες που αναφέρθηκαν. Η διαδικασίες εγκατάστασης μιας εφαρμογής COM+, προσθαφαίρεσης συστατικών και ρύθμισης της επικοινωνίας μεταξύ τους είναι αυτές που θα πρέπει να εκτελεί ένας πωλητής για να μπορεί να εγκαθιστά με επιτυχία ένα Βοήθημα Επικοινωνίας. Η πλήρης διαδικασία περιλαμβάνεται στο Εγχειρίδιο Χρήσης των Πωλητών που είναι ένα από τα θέματα του Παραδοτέου Π3.2.

7.8. Δοκιμαστικά συστατικά ελέγχου και αναφοράς

Για λόγους δοκιμών και ελέγχων, θα πρέπει ο ΟΔΥΣΣΕΑΣ να παρέχει δοκιμαστικά συστατικά στους Προγραμματιστές. Με τη βοήθεια αυτών των συστατικών θα μπορούν να ελέγχουν αν τα δικά τους συστατικά είναι συμβατά με το πλαίσιο ΟΔΥΣΣΕΑΣ. Θα πρέπει να δίνουν την δυνατότητα στους προγραμματιστές να ελέγχουν αν τα δικά τους συστατικά μπορούν να Εκδώσουν δεδομένα και να παραλάβουν δεδομένα μέσα από την Υπηρεσία Γεγονότων και ακολουθώντας τις τυποποιήσεις και οδηγίες του ΟΔΥΣΣΕΑ. Έτσι, θα πρέπει να κατασκευαστούν δυνατότητες συστατικά με έκδοσης σε γεγονότα, και συστατικά με δυνατότητες Συνδρομής σε γεγονότα. Επίσης χρειάζεται και η δημιουργία ενός συστατικού με συνδυασμένες δυνατότητες Έκδοσης και Συνδρομής για να δοκιμάζονται σενάρια συγχρονισμού και τοποθέτησης συστατικών σε σειρά (Πίνακας 6).

Εκτός από τη λειτουργικότητά τους ως δοκιμαστικά συστατικά, τα τρία αυτά συστατικά θα πρέπει να αποτελούν και προγραμματιστικά υποδείγματα για τη χρήση όλων των

υπηρεσιών του ΟΔΥΣΣΕΑ. Θα πρέπει ο κώδικάς τους να είναι ελεύθερα διαθέσιμος, ώστε να χρησιμεύει σαν δείγμα σωστής και αποτελεσματικής εφαρμογής όλων των τυποποιήσεων και οδηγιών του ΟΔΥΣΣΕΑ και των υπηρεσιών του συστήματος που θέτει ο ΟΔΥΣΣΕΑΣ ως υποδομή σε ένα Βοήθημα Επικοινωνίας.

Δοκιμαστικό Συστατικό Εισόδου (Εκδότης)	• Δυνατότητα έκδοσης γεγονότων • Αποστολή του δικού του Class ID σε κάθε μήνυμα • Υποστήριξη όλων των διεπαφών του ΟΔΥΣΣΕΑ • Δυνατότητα επιλογής από τον χρήστη των δεδομένων που Εκδίδονται
Δοκιμαστικό Συστατικό Εξόδου (Συνδρομητής)	• Δυνατότητα συνδρομής σε γεγονότα • Δυνατότητα απεικόνισης των δεδομένων που παραλαμβάνονται • Δυνατότητα απεικόνισης του Class ID του Εκδότη • Υποστήριξη όλων των διεπαφών του ΟΔΥΣΣΕΑ
Δοκιμαστικό Ενδιάμεσο Συστατικό	• Δυνατότητα Έκδοσης γεγονότων σε όλες τις διεπαφές του ΟΔΥΣΣΕΑ • Δυνατότητα Συνδρομής σε γεγονότα σε όλες τις διεπαφές του ΟΔΥΣΣΕΑ • Στοιχειώδη επεξεργασία των δεδομένων που έρχονται στην είσοδο πριν την αποστολή τους στην έξοδο • Δυνατότητα απεικόνισης των δεδομένων πριν και μετά την επεξεργασία τους

Πίνακας 6: Προδιαγραφές δοκιμαστικών συστατικών του ΟΔΥΣΣΕΑ

7.9. Ο ΟΔΥΣΣΕΑΣ στο Διαδίκτυο

Ο ΟΔΥΣΣΕΑΣ θα έχει φυσικά το δικό του τόπο στο Διαδίκτυο. Σε αυτόν τον τόπο θα είναι διαθέσιμες όλες οι τυποποιήσεις και οι οδηγίες που θέτει το πλαίσιο και αναλύονται στην Τεχνική Έκθεση που αφορά στην Υλοποίησή του [53]. Εκτός από αυτά θα είναι διαθέσιμα και τα παραδοτέα του παρόντος έργου που αφορούν το πλαίσιο ΟΔΥΣΣΕΑΣ και φυσικά τα δοκιμαστικά συστατικά του πλαισίου και όλα τα εργαλεία που αναπτύσσονται κατά την υλοποίησή του.

Τέλος ο τόπος του ΟΔΥΣΣΕΑ στο Διαδίκτυο είναι μέρος του ίδιου του πλαισίου. Εκεί έχει σχεδιαστεί να συγκεντρώνονται τα συστατικά που κατασκευάζονται από τους συνεργάτες του έργου ΑΙΝΕΙΑ, αλλά και από ανεξάρτητους κατασκευαστές. Θα περιλαμβάνεται περιγραφή τους, προδιαγραφές τους και οδηγίες χρήσης για το καθένα από αυτά. Θα

λειτουργεί αυτός ο τόπος ως μία βάση δεδομένων για τα συστατικά που είναι συμβατά με το πλαίσιο ΟΔΥΣΣΕΑΣ. Θα είναι μια πηγή πληροφοριών για τους προγραμματιστές, τους κατασκευαστές και τους χρήστες Βοηθημάτων Διαπροσωπικής Επικοινωνίας και για οποιονδήποτε άλλον ενδιαφερόμενο.

Έτσι, σχεδιάζεται ένας τόπος στον οποίον θα φτάνει κανείς από έναν σύνδεσμο της κεντρικής σελίδας του έργου ΑΙΝΕΙΑΣ και θα περιέχει τα εξής:

- Το παρόν παραδοτέο Π3.1 που αφορά το σχεδιασμό του πλαισίου ΟΔΥΣΣΕΑΣ.
- Το παραδοτέο Π3.2 που αφορά την περιγραφή της Υλοποίησης του πλαισίου ΟΔΥΣΣΕΑΣ.
- Βάση δεδομένων με τα συστατικά Βοηθημάτων Διαπροσωπικής Επικοινωνίας που είναι συμβατά με τον ΟΔΥΣΣΕΑ.
- Περιγραφή και τεχνικές εκθέσεις Ολοκληρωμένων Βοηθημάτων Διαπροσωπικής Επικοινωνίας που κατασκευάστηκαν σύμφωνα με το πλαίσιο ΟΔΥΣΣΕΑΣ.
- Τα ελεύθερα διανεμόμενα δοκιμαστικά συστατικά του ΟΔΥΣΣΕΑ, καθώς και τον κώδικά τους.
- Τις διεπαφές που χρησιμοποιούνται στον ΟΔΥΣΣΕΑ για την επικοινωνία των συστατικών και τον κώδικά τους.
- Το πρόγραμμα εγκατάστασης Βοηθημάτων Διαπροσωπικής Επικοινωνίας και το Εγχειρίδιο Χρήσης του.

Φυσικά οποιαδήποτε συστατικά περιέχονται στον τόπο αυτό, θα συνοδεύονται με πλήρη τεκμηρίωσή τους που θα περιλαμβάνει:

- Γενική περιγραφή τους
- Ανάλυση της λειτουργικότητας και των υπηρεσιών που προσφέρουν
- Δυνατότητες συνεργασίας τους με άλλα συστατικά
- Διεπαφές που υλοποιούν
- Οδηγίες χρήσης τους
- Οδηγίες για την εγκατάστασή τους

ΑΝΑΦΟΡΕΣ

1. Γ. Κουρουπέτρογλου: «ΑΙΝΕΙΑΣ - Ανάπτυξη Ευέλικτων Συστημάτων Εναλλακτικής και Επαυξητικής Διαπροσωπικής Επικοινωνίας μέσω Υπολογιστών και του Διαδικτύου», Τεχνικό Δελτίο, ΕΠΕΤ ΙΙ, Αθήνα, 1998.

2. Γ. Κουρουπέτρογλου και Σ. Λιάλιου: «Τεχνική Περιγραφή και Ανάλυση των Χαρακτηριστικών και των Επικοινωνιακών Αναγκών των ΑΜΕΑ-Στόχου του Έργου – Θεωρητική Προσέγγιση», Τεχνική Έκθεση Π1.1.α., Έργο ΑΙΝΕΙΑΣ, ΕΠΕΤ ΙΙ, Αθήνα, 2000.

3. Κ. Βίγλα, Γ. Κουρουπέτρογλου, Ρ. Πίτα και Μ. Αθουσάκη: «Τεχνική Περιγραφή και Ανάλυση των Χαρακτηριστικών και των Επικοινωνιακών Αναγκών των ΑΜΕΑ-Στόχου του Έργου – Πειραματική Προσέγγιση», Έργο ΑΙΝΕΙΑΣ, Τεχνική Έκθεση Π1.1.β, ΕΠΕΤ ΙΙ, Αθήνα, 2000.

4. E. Kaasinen, J. Ahonen, M. Koskinen, C. Stephanidis, A. Paramythis, D. Gramenos, G. Paparoylis, G. Kouroupetroglou, C. Viglas and C. Stamatis: "Report on the implementation of the user interfaces for the two demonstrators", Technical Report D.2.3, part I, TIDE Project 1001 - ACCESS, 1996.

5. G. Kouroupetroglou, A. Anagnostopoulos, C. Viglas and H. Frangouli: "Preliminary design of the modular architecture for interpersonal communication aids", Technical Report 2.2.1, TIDE Project 1001 - ACCESS, 1995.

6. Kouroupetroglou, A. Anagnostopoulos, C. Viglas, C. Stamatis and F. Pentaris: "Report on the modular architecture and the related software Design Tools for Interpersonal communication aids, Part A - Implementation of the modular architecture for interpersonal communication aids", Technical Report D.2.5, TIDE Project 1001 - ACCESS, 1996.

7. G. Kouroupetroglou, A. Anagnostopoulos, C. Viglas, C. Stamatis and F. Pentaris: "Report on the modular architecture and the related software Design Tools for Interpersonal communication aids, Part B – Software design tools accompanying the modular architecture for interpersonal communication aids", Deliverable Technical Report M.2.2.4, TIDE Project 1001 - ACCESS, 1996.

8. G. Kouroupetroglou, C. Viglas, C. Stamatis, E. Kaasinen, J. Ahonen and M. Koskinen: "Report on the implementation of the demonstrator interpersonal communication aids", Technical Report 2.6, TIDE Project 1001 - ACCESS, 1997.

9. P. Majaranta, J. Ahonen, E. Kaasinen and M. Koskinen: "Evaluation of the modular architecture for interpersonal communicators", Technical Report 2.6.3, TIDE Project 1001 - ACCESS, 1996.

10. E. Kaasinen, J. Ahonen and M. Koskinen: "Draft specifications of the demonstrators for interpersonal communication; Language-cognitive impaired users – Part II – Architectural design document", Technical Record M.2.5.1, Project 1001 - ACCESS, 1995.

11. E. Kaasinen, J. Leikas, V. Räty, A. Tammela, A. Anagnostopoulos, G. Kouroupetroglou, P. Syros, C. Viglas, C. Stephanidis, A. Paramythis, M. Antona and

A. Savidis: “Preliminary technical and functional descriptions of currently available communication aids for speech-motor and language cognitive impaired users”, Technical Report M.2.1.2, Project 1001 - ACCESS, 1994.

12. C. Stephanidis: “User interfaces for all”, ERCIM News No. 39, pp. 15-17, 1999.

13. C. Viglas, C. Stamatis and G. Kouroupetroglou: “Remote assistive interpersonal communication exploiting component based development”, Proceedings of the XV IFIP World Computer Congress, Nienna, pp.487-496, 1998.

14. M. Antona, C. Stephanidis and G. Kouroupetroglou: “Vocabulary management in modular interpersonal communication aids”, Assistive Technology Research Series Vol.3, IOS Press, ISBN 90 5199361 7, pp. 200-205, 1997.

15. G. Kouroupetroglou, C. Viglas, C. Stamatis and F. Pentaris: “Towards the next generation of computer-based interpersonal communication aids”, Proceedings of AATE 97, Porto Carras, Greece, pp. 110-114, 1997.

16. S. Tyvard, L. Morton, H. Stegavik and E. Stav: “Technical Specification of Comspec – Release 1”, 1995.

17. M. Lansdale and T. Ormerod: “Understanding interfaces – A handbook of human-computer dialogue”, Academic Press, Computers and people series, ISBN 0-12-528390-3, 1994.

18. P. Thagard: “Mind – Introduction to cognitive science”, The MIT Press, ISBN 0-262-20106-2, 1996.

19. P. Odor and C. Centre: “Functional Specification (and user handbooks)”, Release 1, Comspec, 1995.

20. M. Antona, C. Stephanidis and G. Kouroupetroglou: “Access to lexical knowledge in modular interpersonal communication aids”, Journal of AAC, Vol. 15, pp. 269-279, 1999.

21. A. Brown: “Component-based software engineering”, IEEE Computer Society Press, ISBN 0-8186-7718-X, 1996.

22. Microsoft Developer Days ’99, Conference CD, 1999.

23. Microsoft Developer Days ’99, COM+ Resource CD, Version 1.1, 1999.

24. The MSDN Show, “COM+ Services”, 2000:

http://msdn.microsoft.com/theshow/#

25. J. Smith: “Understanding and Using COM Threading Models”, Web Workshop, Components Development, Microsoft Corporation, 1998:

http://msdn.microsoft.com/workshop/components/com/comthread.asp

26. A. Elswify: “Transform Asynchronous DLL to Synchronous ASP COM”, Microsoft Visual C++ Developer, Pinnacle Publishing, 2000:

http://msdn.microsoft.com/library/periodic/period00/asynch.html

27. D. Platt: “The COM+ Event Service Eases the Pain of Publishing and Subscribing to Data”, Microsoft Systems Journal, September 1999:

http://msdn.microsoft.com/library/periodic/period99/com+event.htm

28. M. Kirtland: “Object-Oriented Software Development Made Simple with COM+ Runtime Services”, Microsoft System Journal, November 1997:

http://www.microsoft.com/msj/1197/complus.htm

29. P. Chung, Y. Huang, S. Yajnik, D. Liang, J. Shih, C. Wang, Y. Wang: "DCOM and CORBA Side by Side, Step by Step, and Layer by Layer", 1977: http://www.cs.wustl.edu/~schmidt/submit/Paper.html

30. "The Microsoft Strategy for Distributed Computing and DCE Services", *A White Paper from the Business Systems Technology Series*, Microsoft Technet, 2000: http://www.microsoft.com/technet/Analpln/dce.asp

31. Orbix 2000 Documentation: http://www.iona.com/docs/orbix2000.html

32. Microsoft Developers Network Online Library: http://msdn.microsoft.com/library/default.htm.

33. MSDN Library/Visual Studio 6.0 Documentation/Visual Basic Documentation, http://msdn.microsoft.com/library/default.asp?URL=/library/devprods/vs6/vbasic/vbcon98/vbstartpage.htm

34. MSDN Library/Visual Studio 6.0 Documentation/Visual Studio Documentation/Component, Design and Analysis Tools/Visual Modeler Reference, http://msdn.microsoft.com/library/default.asp?URL=/library/devprods/vs6/vstudio/vstool2/veovrvisualmodelerreference.htm

35. MSDN Library/Visual Studio 6.0 Documentation/Visual Studio Documentation/Component, Design and Analysis Tools/Visual Studio Installer Documentation, http://msdn.microsoft.com/library/default.asp?URL=/library/devprods/vs6/vstudio/vsinstal/veovrvisualstudioinstalleroverview.htm

36. MSDN Library/Platform SDK/Component Services/ COM, http://msdn.microsoft.com/library/default.asp?URL=/library/psdk/com/comportal_3qn9.htm

37. MSDN Library/Platform SDK/Component Services/ COM+ (Component Services), http://msdn.microsoft.com/library/default.asp?URL=/library/psdk/cossdk/betaintr_6qan.htm

38. MSDN Library/Specifications/Application Specification for Windows 2000 for Desktop Applications http://msdn.microsoft.com/library/default.htm

39. MSDN Library/Specifications/Component Object Model (COM) Specification 0.9, http://msdn.microsoft.com/library/default.asp?URL=/library/specs/w2kcli.htm

40. MSDN Library/Specifications/Distributed Component Object Model Protocol -- DCOM/1.0, http://msdn.microsoft.com/library/default.asp?URL=/library/specs/distributedcomponentobjectmodelprotocoldcom10.htm

41. T. Pattison, “COM+ Overview for Visual Basic Programmers”, MSDN Library/Technical Articles/Component Object Model/COM+/ COM+ Overview for Visual Basic Programmers, 2000, http://msdn.microsoft.com/library/default.asp?URL=/library/techart/complus4vb.htm

42. E. Jeziersky, “COM+ Application Guidelines for Visual Basic Development”, MSDN Library/Technical Articles/ Component Object Model/COM+/COM+ Application Guidelines for Visual Basic Development, 2000, http://msdn.microsoft.com/library/default.asp?URL=/library/techart/complus.htm

43. MSDN Library/Technical Articles/ Component Object Model/COM+/COM+ Technical Series: Loosely Coupled Events, 1999, http://msdn.microsoft.com/library/default.asp?URL=/library/techart/compluscouple.htm

44. S. Williams, C, Kindel, “The Component Object Model: A Technical Overview”, MSDN Library/Technical Articles/Component Object Model/The Component Object Model: A Technical Overview, 1994, http://msdn.microsoft.com/library/default.asp?URL=/library/techart/msdn_comppr.htm

45. D. Souza, BJ. Whalen, P, Wilson, “Implementing Side-by-Side Component Sharing in Applications (Expanded)”, MSDN Library/Technical Articles/Windows Platform/Windows 2000/Implementing Side-by-Side Component Sharing in Applications, 1999, http://msdn.microsoft.com/library/default.asp?URL=/library/techart/sidebyside.htm

46. Microsoft Knowledge Base, "Descriptions and Workings of OLE Threading Models", Article ID Q150777, 1999.

47. Neil Allain, " Agility in Server Components", MSDN Web Workshop, 1997: http://msdn.microsoft.com/workshop/server/components/agility.asp.

48. A. Roftail, T. Martin: “Building N-Tier Applications with COM and Visual Basic 6.0”, Wiley, ISBN 0471295493, 1999.

49. G. Eddon, H. Eddon: “Programming Components with Microsoft Visual Basic 6.0 Second Edition”, Microsoft Press, ISBN 1572319666, 1998.

50. D. Kurata: “Doing Objects in Visual Basic 6.0”, Sams, ISBN 1562765779, 1998.

51. P. Coad, E. Yourdon: “Object-Oriented Design”, Yourdon Press Computing Series, ISBN 0136300707, 1991.

52. P. Coad, E. Yourdon: “Object-Oriented Analysis”, Yourdon Press Computing Series, ISBN 0136299814, 1991.

53. Α. Πίνο, Γ. Κουρουπέτρογλου: «Υλοποίηση του Πλαισίου ΟΔΥΣΣΕΑΣ και Εγχειρίδιο Χρήσης», Τεχνική Έκθεση Π3.2., Έργο ΑΙΝΕΙΑΣ, ΕΠΕΤ ΙΙ, 2000.

54. S. Staab, “Raw DDE”, MSDN Library/Technical Articles/Windows Platform/Base Services/DDE/Raw DDE, 1992 http://msdn.microsoft.com/library/default.asp?URL=/library/techart/msdn_rawdde.htm

55. H. Rodent, “Supporting the Clipboard, DDE, and OLE Applications”, MSDN Library/Technical Articles/Windows Platform/Base Services/DDE/ Supporting the Clipboard, DDE, and OLE Applications, 1992 http://msdn.microsoft.com/library/default.asp?URL=/library/techart/msdn_rawdde.htm

56. MSDN Library/Platform SDK/ Base Services/Interprocess Communication/Pipes, http://msdn.microsoft.com/library/default.asp?URL=/library/psdk/winbase/pipes_94xf.htm

57. MSDN Library/Platform SDK/ Base Services/Interprocess Communication, http://msdn.microsoft.com/library/default.asp?URL=/library/psdk/winbase/ipc_57qr.htm

58. MSDN Library/Platform SDK/ Base Services/DLLs, Processes, and Threads/Dynamic Link Libraries, http://msdn.microsoft.com/library/default.asp?URL=/library/psdk/winbase/dll_512r.htm

59. S. Robinson, A. Krassel, "COMponents", MSDN Library/Technical Articles/Component Object Model/COMponents, 1997, http://msdn.microsoft.com/library/default.asp?URL=/library/techart/msdn_components.htm

60. P. Muller. "Instant UML", Wrox Press Ltd., ISBN 186100871, 1997

61. Γ. Κουρουπέτρογλου, Κ. Ξιπτερίδης, «Τεχνικές Αλληλεπίδρασης με Υπολογιστικά Περιβάλλοντα», Έργο ΑΙΝΕΙΑΣ, ΕΠΕΤ ΙΙ, 2000.