Ειδικές Οδηγίες του ΟΔΥΣΣΕΑ (για τους προγραμματιστές)

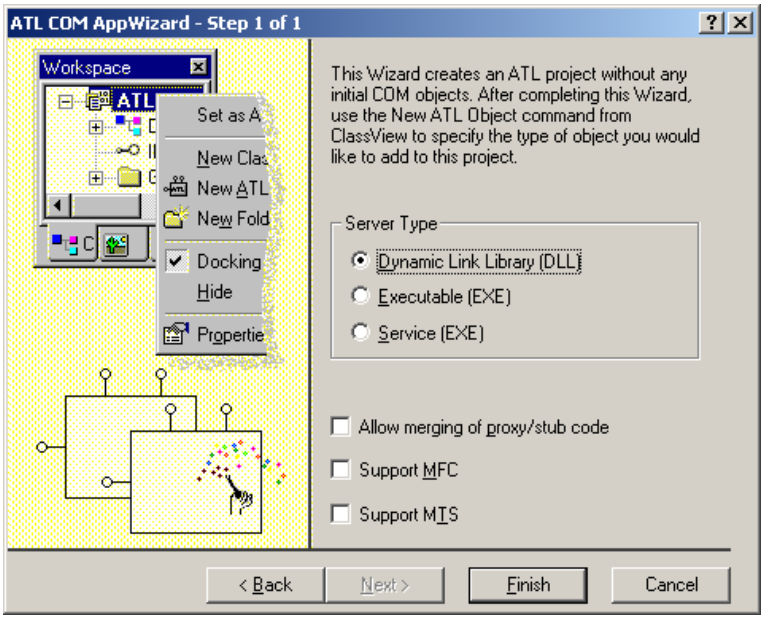
1. Μορφή Συστατικών – DLL

Βασικότερη από όλες τις προδιαγραφές που θέτει ο ΟΔΥΣΣΕΑΣ στους κατασκευαστές των συστατικών του Βοηθήματος Διαπροσωπικής Επικοινωνίας είναι τα προϊόντα τους να έρχονται σε μορφή αρχείων DLL (Βιβλιοθήκες Δυναμικής Σύνδεσης - Dynamic Link Libraries) [26]. Ο κύριος λόγος για τη θέσπιση αυτής της προδιαγραφής είναι ότι, σύμφωνα με το σχεδιασμό του ΟΔΥΣΣΕΑ, όλα τα συστατικά που πρόκειται να ενσωματωθούν σε μια ολοκληρωμένη εφαρμογή, δηλώνονται στο Component Services των Windows 2000. Για να δηλωθεί οποιοδήποτε συστατικό (component) ως μέρος μιας εφαρμογής στο Component Services πρέπει να είναι σε μορφή DLL.

Σχήμα 1: Πλαίσιο διαλόγου για την εκκίνηση της δημιουργίας ενός ActiveX DLL στη Microsoft Visual Basic 6.0.

Φυσικά ο προγραμματισμός κάθε τέτοιου συστατικού πρέπει να γίνει με τα μοντέρνα εργαλεία και γλώσσες προγραμματισμού που μπορούν να υποστηρίξουν τη δημιουργία τέτοιας μορφής αρχείων, αλλά και παράγουν αρχεία συμβατά με τις τυποποιήσεις του COM και του COM+. Αυτή τη συμβατότητα μπορεί να την ελέγξει κάποιος προγραμματιστής δοκιμάζοντας να δηλώσει ως νέο συστατικό σε μια εφαρμογή στα Component Services. Κατά τη διάρκεια αυτής της διαδικασίας φαίνεται αν το σύστημα αναγνωρίζει το συστατικό ως ένα DLL συμβατό με COM+ και αν υπάρχει κάποιο σοβαρό πρόβλημα συμβατότητας το

σύστημα ειδοποιεί το χρήστη με μηνύματα σφάλματος. Ένα περαιτέρω βήμα για τον έλεγχο των συστατικών με το πλαίσιο ΟΔΥΣΣΕΑΣ είναι να προμηθευτεί ο κατασκευαστής το δοκιμαστικό πρόγραμμα φόρτωσης των συστατικών και τα δοκιμαστικά προγράμματα αποστολής και λήψης COM+ Events που έχουν ήδη περιγραφεί και να ακολουθήσει τις διαδικασίες φόρτωσης και εκκίνησης του Βοηθήματος Διαπροσωπικής Επικοινωνίας, όπως θα έκανε και σε πραγματικές συνθήκες ένας πωλητής ενός τέτοιου συστήματος. Όλα αυτά τα προγράμματα είναι ελεύθερα προς διανομή στο Διαδικτυακό τόπο του έργου.



Σχήμα 2: Πλαίσιο διαλόγου για την κατασκευή ενός DLL εφαρμογής ATL COM στη Microsoft Visual C++ 6.0.

2. Διεπαφή χρήσης – η κλάση `Activate`

Κάθε συστατικό που είναι συμβατό με το Πλαίσιο Ανάπτυξης Εφαρμογών ΟΔΥΣΣΕΑΣ και πρέπει να εμφανίζει μια διεπαφή χρήσης κατά τη λειτουργία του, θα πρέπει οπωσδήποτε να περιέχει μια κλάση με το όνομα `Activate`. Το όνομα της κλάσης αυτής είναι αυστηρά καθορισμένο από τις προδιαγραφές του ΟΔΥΣΣΕΑ και πρέπει να χρησιμοποιείται μόνο για τη λειτουργία που περιγράφεται εδώ. Επίσης πρέπει να σημειωθεί ότι είναι case-sensitive, που σημαίνει ότι το πρώτο γράμμα πρέπει οπωσδήποτε να είναι κεφαλαίο και τα υπόλοιπα πεζά όπως ακριβώς γράφηκε παραπάνω.

Αν και κάθε αντικείμενο που συμμετέχει στην «ορχήστρα» των συστατικών ενός Βοηθήματος Διαπροσωπικής Επικοινωνίας βασισμένου στον ΟΔΥΣΣΕΑ έχει τον τυπικό τρόπο ενεργοποίησης μέσω της υπηρεσίας γεγονότων των Windows 2000, τα αντικείμενα που αντιπροσωπεύουν τις αρχικές διεπαφές χρήσης των συστατικών δεν μπορούν να ξεκινούν με τον ίδιο αυτόματο τρόπο. Αυτό είναι λογικό μια και κατά τον αυτόματο τρόπο ενεργοποίησης των επιθυμητών αντικειμένων που συμμετέχουν κατά οποιοδήποτε τρόπο στις υπηρεσίες γεγονότων (είτε ως Εκδότες είτε ως Συνδρομητές) χρειάζεται κάποιο ερέθισμα από τον χρήστη για να εκκινηθούν οι διαδικασίες του συστήματος. Και βέβαια για μπορέσει να δώσει ο χρήστης είσοδο στο Βοήθημα Διαπροσωπικής Επικοινωνίας θα πρέπει να υπάρχει

ήδη κάποια ενεργοποιημένη διεπαφή χρήσης. Επίσης, αυτή η διεπαφή χρήσης θα πρέπει να είναι ολοκληρωμένη μπροστά στον χρήστη κατά τη διάρκεια όλης της αλληλεπίδρασής του με το σύστημα. Το πρόβλημα ήταν λοιπόν πως θα ξεδίπλωναν όλα τα συστατικά το καθένα τη δική του διεπαφή χρήσης, συνθέτοντας έτσι την ολοκληρωμένη διεπαφή χρήσης του Βοηθήματος. Ακόμη θα έπρεπε η εμφάνιση όλων αυτών των διεπαφών χρήσης να γίνεται ταυτόχρονα και αυτόματα κατά την εκκίνηση του Βοηθήματος, χωρίς βέβαια να απαιτείται από τον πιθανώς μειωμένων δυνατοτήτων χρήστη να ενεργοποιήσει το κάθε συστατικό που χρειάζεται ξεχωριστά. Αυτό το πρόβλημα ήρθε να λύσει η κλάση `Activate`.

Κατασκευάστηκε ένα πρόγραμμα εκκίνησης του Βοηθήματος το οποίο μεταξύ άλλων λειτουργιών ανέλαβε και το καθήκον της ενεργοποίησης των αρχικών διεπαφών χρήσης όλων των χρησιμοποιούμενων συστατικών. Αυτό το πρόγραμμα, το οποίο γράφτηκε στη γλώσσα Microsoft Visual Basic 6.0 (Service Pack 4), ενεργοποιεί την κλάση `Activate` με την εξής διαδικασία:

- Ανοίγει το Component Catalog του λειτουργικού συστήματος.
- Αναζητά την εφαρμογή με το όνομα AENEAS.
- Μέσα στη συγκεκριμένη εφαρμογή αναζητά όλα τα συστατικά (αντικείμενα) που το όνομά τους καταλήγει σε `.Activate`.

  Σημείωση: Κάθε DLL που εγκαθίσταται στα Component Services των Windows 2000 μπορεί να έχει πολλές κλάσεις και κατ' επέκταση πολλά αντικείμενα ενσωματωμένα. Αυτά φαίνονται και κατά την εγκατάστασή του αλλά και στο Component Catalog. Υπάρχουν εκεί καταχωρήσεις με την εξής μορφή:

  `ΌνομαDLL.ΌνομαΚλάσης`

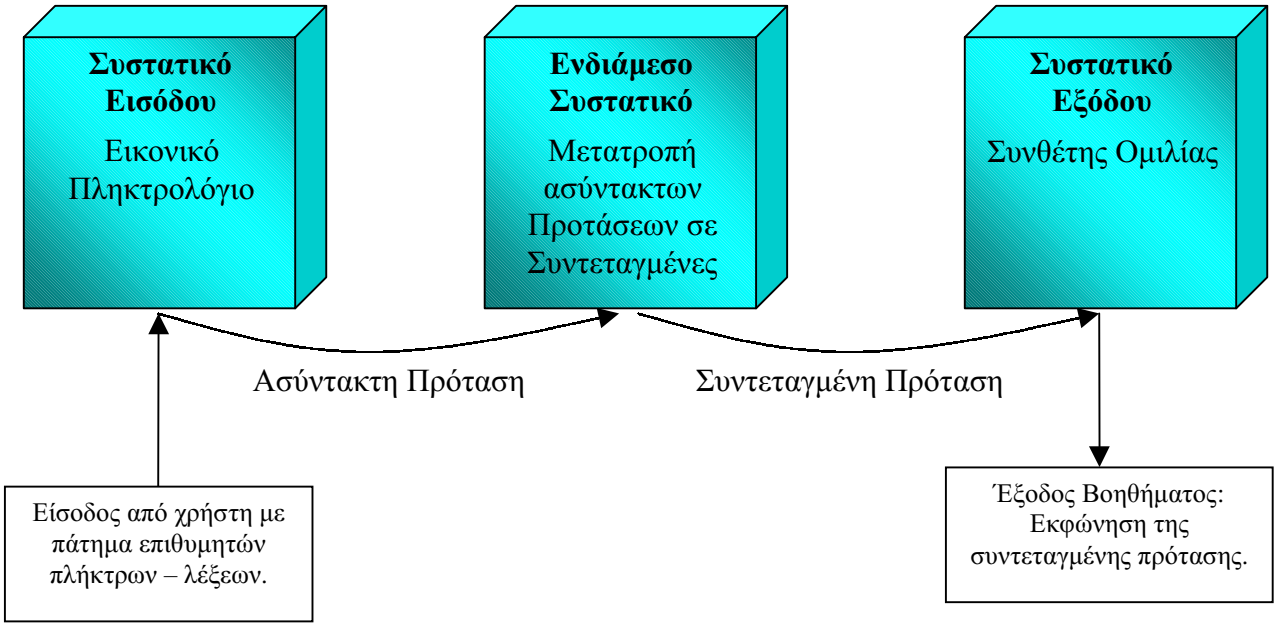
- Ενεργοποιεί όλα αυτά τα αντικείμενα (μέσω της `CreateObject`).

Αυτό είναι και το πρόγραμμα που πρέπει να εκτελέσει ο τελικός χρήστης για να ξεκινήσει το Βοήθημα Διαπροσωπικής Επικοινωνίας του. Το αποτέλεσμα της παραπάνω διαδικασίας είναι να εμφανιστούν όλες οι διεπαφές χρήσης των συστατικών που είναι εγκατεστημένα στην εφαρμογή AENEAS.

Η τεχνική για να ενεργοποιείται η διεπαφή χρήσης ενός συστατικού κατά την ενεργοποίηση μιας κλάσης του είναι να εισάγεται μέσω προγραμματισμού η εμφάνιση της επιθυμητής φόρμας στη μέθοδο αρχικοποίησης της συγκεκριμένης κλάσης (για τη Visual Basic είναι η `Class_Initialize`).

Βέβαια, μπορεί να υπάρχουν και συστατικά τα οποία δεν απαιτούν την ύπαρξη μιας διεπαφής χρήσης κατά τη λειτουργία τους. Για παράδειγμα, ένα συστατικό Μετατροπής Ασύντακτων Προτάσεων σε συντεταγμένες (π.χ. από «θέλω βλέπω τηλεόραση βράδυ» σε «θέλω να δω τηλεόραση το βράδυ») μπορεί να λειτουργεί στο παρασκήνιο, παρεμβαλλόμενο μεταξύ ενός συστατικού εισόδου (π.χ. ένα Εικονικό Πληκτρολόγιο) και ενός συστατικού εξόδου (π.χ. ένας Συνθέτης Ομιλίας). Το συγκεκριμένο λοιπόν συστατικό μπορεί να λειτουργεί χωρίς να κάνει ποτέ την εμφάνισή του με τη μορφή μιας διεπαφής χρήσης και δε χρειάζεται να έχει την κλάση `Activate` (βλέπε Σχήμα 3). Η έναρξη της λειτουργίας του, όπου αυτή χρειάζεται γίνεται από το λειτουργικό σύστημα μέσω των υπηρεσιών γεγονότων. Φυσικά πρέπει να έχουν τηρηθεί όλες οι άλλες προδιαγραφές που αφορούν στην λειτουργία ενός συστατικού με αυτόν τον τρόπο. Όταν μια ασύντακτη πρόταση «εκδίδεται» από ένα εικονικό πληκτρολόγιο, δημιουργείται ένα αντικείμενο του συγκεκριμένου συστατικού (μπορεί να συναντηθεί και με το όνομα Parser) το οποίο είναι Συνδρομητής με Συνδρομή στον Εκδότη Εικονικό Πληκτρολόγιο. Στο συγκεκριμένο παράδειγμα, ούτε και το συστατικό εξόδου, δηλαδή ο Συνθέτης Ομιλίας είναι απαραίτητο να έχει την κλάση `Activate`. Μπορεί και αυτό να

λειτουργεί στο παρασκήνιο και να μη χρειάζεται διεπαφή χρήσης κατά την εκκίνησή του. Βέβαια το Εικονικό πληκτρολόγιο χρειάζεται διεπαφή χρήσης για να δίνει ο χρήστης την είσοδο στο Βοήθημα πατώντας τα πλήκτρα του.



Σχήμα 3: Παράδειγμα Βοηθήματος Διαπροσωπικής Επικοινωνίας αποτελούμενου από τρία συστατικά.

3. Μοντέλο νημάτων – `Single Threaded`

ΠΡΟΣΟΧΗ: Αυτή η οδηγία αναφέρεται σε όσα συστατικά διαθέτουν User Interface κατά το χρόνο εκτέλεσης και ειδικότερα σε όσα η διεπαφή χρήσης τους συμπίπτει με την κλάση Activate.

Η οδηγία αυτή αφορά το μοντέλο νημάτων (threading model) [25], [46] των συστατικών που πρόκειται να ενσωματωθούν σε ένα Βοήθημα Διαπροσωπικής Επικοινωνίας που υποστηρίζεται από το πλαίσιο ΟΔΥΣΣΕΑΣ.

Τα μοντέλα νημάτων στο COM παρέχουν το μηχανισμό για να συνεργάζονται συστατικά που χρησιμοποιούν διαφορετικές αρχιτεκτονικές νημάτων. Επίσης παρέχουν υπηρεσίες συγχρονισμού στα συστατικά που τις χρειάζονται. Για παράδειγμα, ένα συγκεκριμένο αντικείμενο μπορεί να είναι σχεδιασμένο να καλείται μόνο από ένα μονό νήμα και μπορεί να μη συγχρονίζει ταυτόχρονες κλήσεις από πελάτες. Αν ένα τέτοιο αντικείμενο καλεστεί ταυτόχρονα από πολλαπλά νήματα, καταρρέει ή προκαλεί σφάλματα. Το COM παρέχει τους μηχανισμούς για τη διαχείριση της διαλειτουργικότητας των αρχιτεκτονικών των νημάτων.

Ακόμα και συστατικά που υποστηρίζουν τα νήματα, συχνά χρειάζονται υπηρεσίες συγχρονισμού. Για παράδειγμα, συστατικά που έχουν μια γραφική διεπαφή χρήσης (graphical user interface – GUI), όπως τα OLE/ActiveX controls, τα in-place active embeddings, και τα ActiveX documents, απαιτούν συγχρονισμό και σειριακή διάταξη των κλήσεων του COM και των μηνυμάτων των παραθύρων. Το COM παρέχει αυτές τις υπηρεσίες συγχρονισμού έτσι

ώστε αυτά τα COM συστατικά να μπορούν να γραφτούν χωρίς πολύπλοκο κώδικα συγχρονισμού.

Ένα "διαμέρισμα" ("apartment") έχει αρκετές συσχετιζόμενες όψεις. Πρώτον, είναι μια λογική κατασκευή για να μας βοηθά να σκεφτόμαστε την ταυτόχρονη συνεργασία (concurrency), όπως το πώς τα νήματα σχετίζονται με ένα σύνολο από COM αντικείμενα. Δεύτερον, είναι ένα σύνολο από κανόνες που πρέπει να ακολουθούν οι προγραμματιστές για να επιτυγχάνουν τη συμπεριφορά της ταυτόχρονης συνεργασίας που περιμένουν από το περιβάλλον του COM. Τέλος, είναι κώδικας που παρέχεται από το σύστημα και βοηθά τους προγραμματιστές να διαχειριστούν την ταυτόχρονη συνεργασία σε σχέση με τα αντικείμενα του COM.

Ο όρος ¨διαμέρισμα" ("apartment") προέρχεται από μια μεταφορά στην οποία μια διεργασία θεωρείται μια εντελώς διακριτή οντότητα, όπως ένα "κτίριο" που χωρίζεται σε ένα σύνολο από συσχετιζόμενες αλλά διαφορετικές "τοποθεσίες" που λέγονται ¨διαμερίσματα". Ένα διαμέρισμα είναι ένα "λογικό δοχείο" που δημιουργεί μια συσχέτιση μεταξύ αντικειμένων και, σε μερικές περιπτώσεις, νημάτων (threads). Τα νήματα δεν είναι διαμερίσματα, αν και μπορεί να υπάρχει ένα μονό νήμα που είναι λογικά συσχετισμένο με ένα διαμέρισμα στο μοντέλο STA. Τα αντικείμενα δεν είναι διαμερίσματα, αν και κάθε αντικείμενο είναι συσχετισμένο με ένα και μοναδικό διαμέρισμα. Όμως τα διαμερίσματα είναι κάτι περισσότερο από μια λογική κατασκευή: οι κανόνες τους περιγράφουν τη συμπεριφορά του συστήματος COM. Αν δεν ακολουθούνται οι κανόνες των μοντέλων διαμερισμάτων, τα αντικείμενα του COM δεν λειτουργούν σωστά.

Τα COM αντικείμενα μπορούν να χρησιμοποιηθούν σε πολλαπλά νήματα μιας διεργασίας. Οι όροι "Διαμέρισμα Μονού Νήματος – Single Threaded Apartment – STA" και "Διαμέρισμα Πολλαπλών Νημάτων – Multi Threaded Apartment – MTA" χρησιμοποιούνται για τη δημιουργία ενός εννοιολογικού πλαισίου για την περιγραφή της σχέσης μεταξύ των αντικειμένων και των νημάτων, τις σχέσεις συνεργασίας μεταξύ των αντικειμένων, τις μεθόδους με τις οποίες οι κλήσεις μεθόδων παραδίδονται στα αντικείμενα και των κανόνων για το πέρασμα δεικτών διεπαφών μεταξύ νημάτων. Τα συστατικά και οι πελάτες τους επιλέγουν μεταξύ των δύο μοντέλων που υποστηρίζονται από το COM:

1) **Single-threaded Apartment model (STA)**: Ένα ή περισσότερα νήματα σε μια διεργασία χρησιμοποιούν το COM και οι κλήσεις στα αντικείμενα του COM συγχρονίζονται από το COM. Οι διεπαφές γίνονται marshaled μεταξύ των νημάτων. Μια εκφυλισμένη περίπτωση του μοντέλου διαμερισμάτων μονού νήματος, όπου μόνο ένα νήμα μέσα σε μια δεδομένη διεργασία χρησιμοποιεί COM, ονομάζεται μοντέλο μονού νήματος (Single-Threaded Model).
2) **Multi-threaded Apartment Model (MTA)**: Ένα ή περισσότερα νήματα χρησιμοποιούν το COM και οι κλήσεις στα COM αντικείμενα που υπακούουν στο MTA γίνονται απ' ευθείας από όλα τα νήματα που υπακούουν στο MTA χωρίς καμία παρέμβαση του κώδικα του συστήματος μεταξύ του καλούντα και του αντικειμένου. Επειδή μπορεί πολλαπλοί ταυτόχρονοι πελάτες να καλούν αντικείμενα λίγο πολύ ταυτόχρονα (ταυτόχρονα σε συστήματα πολλαπλών επεξεργαστών), τα αντικείμενα πρέπει να συγχρονίζουν την εσωτερική τους κατάσταση από μόνα τους. Οι διεπαφές δεν γίνονται marshaled μεταξύ των νημάτων.
3) **Το μοντέλο STA και το μοντέλο MTA** μπορούν να χρησιμοποιηθούν **μαζί** στην ίδια διεργασία. Αυτές οι διεργασίες αναφέρονται και ως διεργασίες "μικτού μοντέλου".

Ένα συστατικό μπορεί να επιλέξει να υποστηρίξει το μοντέλο STA, το μοντέλο MTA, ή ένα συνδυασμό των δύο χρησιμοποιώντας το μοντέλο των μικτών νημάτων. Για παράδειγμα, ένα αντικείμενο που κάνει μεγάλη χρήση λειτουργιών εισόδου/εξόδου μπορεί να επιλέξει να

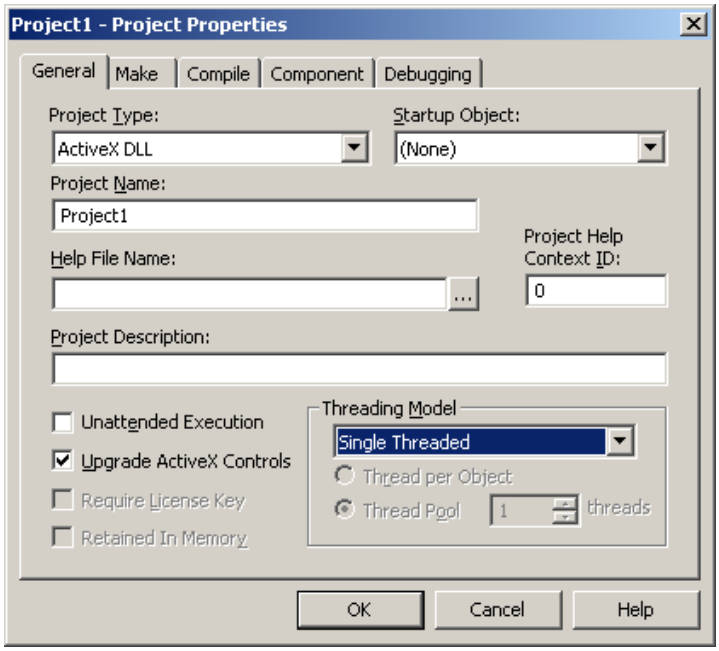
υποστηρίξει το MTA για να παρέχει τη βέλτιστη απόκριση στους πελάτες επιτρέποντας στις κλήσεις διεπαφών να γίνονται κατά τη διάρκεια της λανθάνουσας κατάστασης της εισόδου/εξόδου. Σε άλλη περίπτωση, ένα αντικείμενο που αλληλεπιδρά με το χρήστη σχεδόν πάντα , επιλέγει να υποστηρίξει το STA για να συγχρονίζουν τις εισερχόμενες κλήσεις του COM με τις ενέργειες της γραφικής διεπαφής χρήσης.

Η υποστήριξη του μοντέλου STA είναι ευκολότερη επειδή το COM παρέχει το συγχρονισμό. Η υποστήριξη του μοντέλου MTA δυσκολότερη επειδή το αντικείμενο θα πρέπει να υλοποιεί το συγχρονισμό, αλλά η απόκριση στους πελάτες είναι καλύτερη επειδή ο συγχρονισμός χρησιμοποιείται για μικρότερα τμήματα κώδικα, αντί για όλη την κλήση της διεπαφής που προσφέρει το COM.

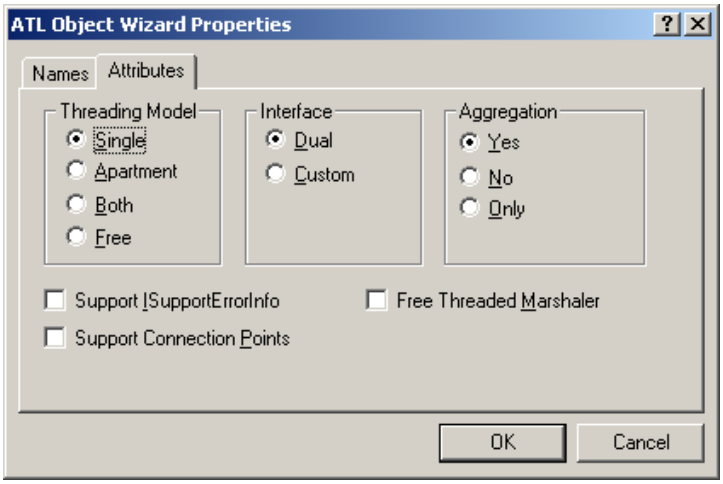
Στην περίπτωση του ΟΔΥΣΣΕΑ, επιλέχτηκε η χρήση του μοντέλου μονού νήματος (single-threaded model). Ο κύριος λόγος που έγινε αυτό είναι ότι η χρήση οποιουδήποτε άλλου μοντέλου που χρησιμοποιεί διαμερίσματα δεν επιτρέπει την εκκίνηση των παραθύρων των διεπαφών χρήσεων των συστατικών, ως non-modal forms. Modal forms είναι τα παράθυρα ή πλαίσια διαλόγου, τα οποία όταν εμφανίζονται, παίρνουν τον έλεγχο της εφαρμογής και απαιτούν κάποια ενέργεια του χρήστη για να κλείσουν ή να δώσουν τον έλεγχο σε άλλα παράθυρα. Τα modal forms δεν αφήνουν κανένα άλλο παράθυρο ή φόρμα να γίνει ενεργή πριν τα ίδια κλείσουν. Η λειτουργικότητα του Βοηθήματος Διαπροσωπικής Επικοινωνίας απαιτεί να υπάρχουν στην οθόνη πάνω από ένα παράθυρα ταυτόχρονα και να υπάρχει η δυνατότητα καθένα από αυτά να μπορεί να γίνει ενεργό και να κάνει κάποια εργασία. Η φιλοσοφία των modal forms δεν θα επέτρεπε κάτι τέτοιο.

Η αναγκαιότητα χρησιμοποίησης no-modal forms λοιπόν, αναγκάζει το σχεδιασμό του συστήματος να γίνει με βάση το μοντέλο μονού νήματος ώστε να μπορεί το πρόγραμμα εκκίνησης του Βοηθήματος Διαπροσωπικής Επικοινωνίας να μπορεί να ανοίξει πολλά λειτουργικά παράθυρα ταυτόχρονα.

Η κατάλληλη ρύθμιση γίνεται στη Visual Basic και στη Visual C++ σε ειδικά πλαίσια διαλόγου, όπως φαίνεται στα Σχήματα 4 και 5 αντίστοιχα.



Σχήμα 4: Πλαίσιο διαλόγου για τον ορισμό του Threading Model του DLL στη Microsoft Visual Basic 6.0 (μενού Project>>Properties)



Σχήμα 5: Πλαίσιο διαλόγου για τον ορισμό του Threading Model ενός ATL Object στη Microsoft Visual C++ 6.0 (wizard εισαγωγής νέου απλού ATL αντικειμένου σε μία ATL εφαρμογή.

4. Βιβλιοθήκη διεπαφής – `Interface.dll`

Η βιβλιοθήκη διεπαφών του ΟΔΥΣΣΕΑ είναι ένα αρχείο βιβλιοθήκης δυναμικής σύνδεσης (Dynamic Link Library – DLL) που ονομάζεται `Interface.dll`. Πρόκειται για μια υλοποίηση σε Microsoft Visual Basic 6.0 μιας κλάσης γεγονότων, όπως αυτή περιγράφεται στις τυποποιήσεις του COM+ [27], [37]. Συγκεκριμένα, για τη λειτουργία της υπηρεσίας γεγονότων (event service) απαιτείται μία διεπαφή που λέγεται κλάση γεγονότων (Event Class) που τυποποιεί την επικοινωνία μεταξύ των Εκδοτών και των Συνδρομητών.

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 υλοποιήθηκε στα πλαίσια του ΟΔΥΣΣΕΑ, αποτελείται από τέσσερις κλάσεις, οι οποίες εξυπηρετούν την επικοινωνία με τέσσερις αντίστοιχους τύπους δεδομένων. Ο ΟΔΥΣΣΕΑΣ έχει σχεδιαστεί ώστε να υποστηρίζει τους εξής τύπους δεδομένων:

1. **Χαρακτήρας**: χρησιμοποιείται για την επικοινωνία συστατικών ανά χαρακτήρα. Για παράδειγμα, θα μπορούσε να στέλνει ένα συστατικό επεξεργαστή κειμένου τον κάθε χαρακτήρα που πληκτρολογεί ο χρήστης, σε ένα συστατικό πρόβλεψης λέξεων, μέσω της κλάσης `Interface.Character`.

2. **Λέξη**: χρησιμοποιείται για την επικοινωνία των συστατικών ανά λέξη. Για παράδειγμα, θα μπορούσε ένα συστατικό εικονικού πληκτρολογίου συμβόλων BLISS να στέλνει τη λέξη που αντιστοιχεί σε κάθε σύμβολο που επιλέγει ο χρήστης, σε ένα συστατικό σύνθεσης ομιλίας για να εκφωνηθεί, μέσω της κλάσης `Interface.Word`.
3. **Πρόταση**: χρησιμοποιείται για την επικοινωνία μεταξύ των συστατικών με ολόκληρες φράσεις. Για παράδειγμα, ένα συστατικό ελέγχου της σύνταξης και της γραμματικής, θα απαιτούσε από ένα συστατικό εικονικού πληκτρολογίου λέξεων, να του αποστέλλει ολόκληρη την πρόταση αφού πατήσει ο χρήστης ένα κουμπί που δηλώνει ότι τελείωσε με την σύνταξή της, μέσω της κλάσης `Interface.Sentence`.
4. **Έγγραφο**: χρησιμοποιείται για την επικοινωνία μεταξύ των συστατικών με ολοκληρωμένα έγγραφα. Για παράδειγμα, ένα συστατικό αποστολής ηλεκτρονικών μηνυμάτων μέσω e-mail, θα απαιτούσε από ένα συστατικό σύνταξης μηνυμάτων, να του αποστέλλει ολόκληρη το έγγραφο προς αποστολή αφού πατήσει ο χρήστης ένα κουμπί που δηλώνει ότι τελείωσε με την σύνταξή του, μέσω της κλάσης `Interface.Document`.

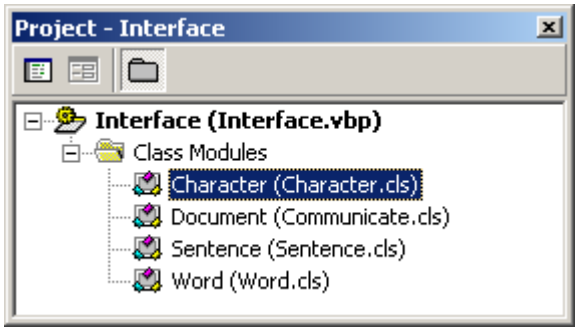
Έτσι τα δεδομένα που ανταλλάσσονται μεταξύ των συστατικών του ΟΔΥΣΣΕΑ είναι πάντα αλφαριθμητικές σειρές (strings), σε τέσσερις διαφορετικές διεπαφές, δηλαδή τέσσερις διαφορετικές κλάσεις ανάλογα με το μέγεθος και τη λειτουργικότητα των δεδομένων. Προγραμματιστικά δεν υπάρχει καμία διαφορά μεταξύ των 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.

Παρακάτω παρατίθεται ο κώδικας και η δομή για τη διεπαφή του Interface.dll, μια σχηματική απεικόνιση της λειτουργικότητας της διεπαφής και οδηγίες για τη χρήση της.

```
'••••••• ••• ••• •••••••••• •••• ••• •••••••• •••••••••
Public Sub Communicate(ByVal Data As String, ByVal PubID As String)
End Sub

'••••••• ••• ••• •••••••••• •••• ••• •••••••• •••••
Public Sub Communicate(ByVal Data As String, ByVal PubID As String)
End Sub

'••••••• ••• ••• •••••••••• •••• ••• •••••••• ••••••••
Public Sub Communicate(ByVal Data As String, ByVal PubID As String)
End Sub

'••••••• ••• ••• •••••••••• •••• ••• •••••••• ••••••••
Public Sub Communicate(ByVal Data As String, ByVal PubID As String)
End Sub

'••••••• ••• ••• •••••••••• •••• ••• •••••••• •••••••••
```

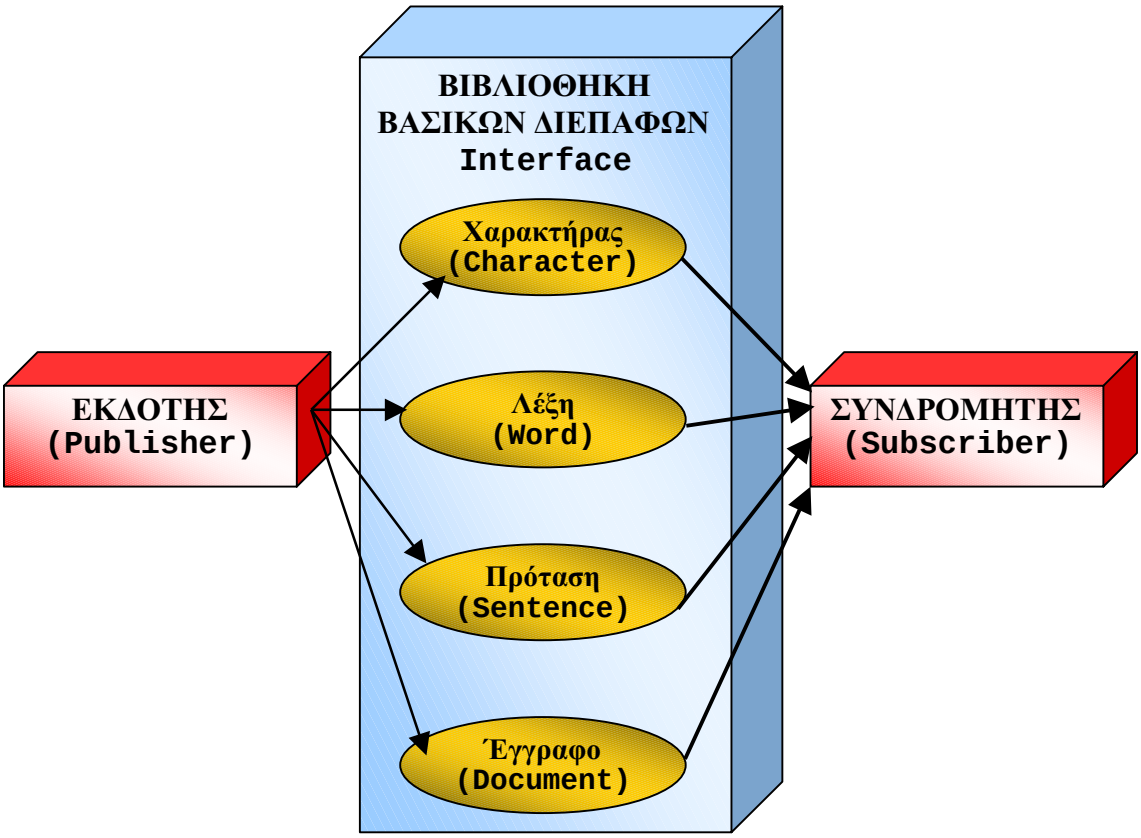
```
Public Sub Communicate(ByVal Data As String, ByVal PubID As String)
End Sub
```

Κώδικας 1: Η κλάση γεγονότων Interface.dll



Σχήμα 6: Η δομή της κλάσης γεγονότων Interface.dll

Η χρήση της βιβλιοθήκης αυτής είναι πολύ απλή για τους προγραμματιστές: αρκεί να την προμηθευτούν (διανέμεται ελεύθερα) και να την ενσωματώσουν στο περιβάλλον προγραμματισμού τους. Μπορούν να την προσθέσουν στα References στη Microsoft Visual Basic ή να την κάνουν include στη Microsoft Visual C++. Στη συνέχεια, ανάλογα με τη λειτουργικότητα του συστατικού τους μπορούν να χρησιμοποιήσουν τις κλάσεις και τις μεθόδους της. Είναι φανερό ότι αυτή η βιβλιοθήκη του ΟΔΥΣΣΕΑ χρησιμοποιείται για την επικοινωνία μεταξύ των Συστατικών. Έτσι αν ο προγραμματιστής κατασκευάζει ένα συστατικό που θα έχει το ρόλο του Εκδότη, αυτό που πρέπει να κάνει είναι να δημιουργήσει το κατάλληλο αντικείμενο από τη βιβλιοθήκη Interface, ανάλογα με τον τύπο δεδομένων που πρέπει να εκδοθούν. Στη συνέχει αρκεί να εκτελέσει τη μέθοδο `Communicate` με τα κατάλληλα ορίσματα `data` και `PubID` για να εκδώσει τα δεδομένα του στη διεπαφή της υπηρεσίας γεγονότων που επέλεξε. Ένα απλό παράδειγμα για το πως γίνεται αυτό στη Visual Basic δίνεται στον παρακάτω Κώδικα.



Σχήμα 7: Η βασική διεπαφή του ΟΔΥΣΣΕΑ: Το Event Class `Interface.dll`.

Η χρήση της βιβλιοθήκης αυτής είναι πολύ απλή για τους προγραμματιστές: αρκεί να την προμηθευτούν (διανέμεται ελεύθερα) και να την ενσωματώσουν στο περιβάλλον προγραμματισμού τους. Μπορούν να την προσθέσουν στα References στη Microsoft Visual Basic ή να την κάνουν include στη Microsoft Visual C++. Στη συνέχεια, ανάλογα με τη λειτουργικότητα του συστατικού τους μπορούν να χρησιμοποιήσουν τις κλάσεις και τις μεθόδους της. Είναι φανερό ότι αυτή η βιβλιοθήκη του ΟΔΥΣΣΕΑ χρησιμοποιείται για την επικοινωνία μεταξύ των Συστατικών. Έτσι αν ο προγραμματιστής κατασκευάζει ένα συστατικό που θα έχει το ρόλο του Εκδότη, αυτό που πρέπει να κάνει είναι να δημιουργήσει το κατάλληλο αντικείμενο από τη βιβλιοθήκη Interface, ανάλογα με τον τύπο δεδομένων που πρέπει να εκδοθούν. Στη συνέχει αρκεί να εκτελέσει τη μέθοδο `Communicate` με τα κατάλληλα ορίσματα `data` και `PubID` για να εκδώσει τα δεδομένα του στη διεπαφή της υπηρεσίας γεγονότων που επέλεξε. Ένα απλό παράδειγμα για το πως γίνεται αυτό στη Visual Basic δίνεται στον παρακάτω Κώδικα.

ΟΝΟΜΑΤΟΛΟΓΙΚΗ ΟΔΗΓΙΑ: Ο κώδικας για τον Εκδότη θα πρέπει να βρίσκεται μέσα σε μια κλάση που λέγεται `Publisher`. Με αυτόν τον τρόπο θα είναι πιο εύκολη η αναγνώριση της δυνατότητας έκδοσης ενός συστατικού διαχειριστικά (στα Component Services). Αν απαιτείται από τη διάρθρωση του προγράμματος να υπάρχει ειδική μέθοδος για την Έκδοση δεδομένων αυτή θα πρέπει να λέγεται `Publish`.

```
'..... .... ............
Dim objEventClass As Object
'....... ... ........... .... ............ ...... ... Interface, .•.•. •••
'Character
Set objEventClass = CreateObject("Interface.Character")
```

```
‘•••••••••••••• ••• ••••••• Communicate ••• ••••••••••••, ••• ••• •••••• •••
‘••••••••• data
objEventClass.Communicate data, PubID
‘•••••••• ••• •••••••••••• ••• •• ••••• (•••• ••••••••• ••• ••••••• ••
‘•••••••)
Set objEventClass = Nothing
```

Κώδικας 2: Υλοποίηση για έναν Εκδότη στη Visual Basic

Αντίστοιχα, για την κατασκευή ενός Συνδρομητή, η διαδικασία είναι εξίσου απλή: Μετά την ενσωμάτωση της βιβλιοθήκης στο περιβάλλον προγραμματισμού, αρκεί η δήλωση ότι πρόκειται να υλοποιηθεί η κατάλληλη διεπαφή και η υλοποίησή της. Στη συνέχεια μέσω της υλοποίησης της μεθόδου Communicate, γίνονται διαθέσιμα στο πρόγραμμα τα δεδομένα που έχουν εκδοθεί στην υπηρεσία γεγονότων και μπορεί να κατασκευαστεί η επιθυμητή λειτουργικότητα και η διαδικασία επεξεργασίας των δεδομένων. Παράδειγμα τέτοιας υλοποίησης στη Visual Basic, δίνεται στον παρακάτω Κώδικα.

ΟΝΟΜΑΤΟΛΟΓΙΚΗ ΟΔΗΓΙΑ: Ο κώδικας για τον Συνδρομητή θα πρέπει να βρίσκεται μέσα σε μια κλάση που λέγεται `Subscriber`. Με αυτόν τον τρόπο θα είναι πιο εύκολη η αναγνώριση της δυνατότητας συνδρομών ενός συστατικού διαχειριστικά (στα Component Services). Αν απαιτείται από τη διάρθρωση του προγράμματος να υπάρχει ειδική μέθοδος για την υλοποίηση της συνδρομής (ανάκτηση δεδομένων) αυτή θα πρέπει να λέγεται `Subscribe`.

```
‘•••••• ••• ••••••••• •• ••••••••• • ••••••• Interface.Word
Implements Interface.Word
‘•••••••••••••• ••• ••••••• Communicate ••• •••••••••••••• •••••• ••• ••
‘•••••••••• ••• •••••••••
Private Sub Word_Communicate(ByVal Data As String, ByVal PubID As String)
‘••• ••••••• • ••••••• ••• ••• •••••••••• ••• •••••••••
End Sub
```

Κώδικας 3: Υλοποίηση για έναν Συνδρομητή στη Visual Basic.

Βέβαια, μπορούν να χρησιμοποιηθούν στο ίδιο συστατικό, είτε αυτό είναι Εκδότης είτε Συνδρομητής, πολλαπλές διεπαφές. Δηλαδή μπορεί ένα συστατικό να Εκδίδει και χαρακτήρες και λέξεις και προτάσεις και έγγραφα, ή οποιονδήποτε συνδυασμό από όλα αυτά. Αντίστοιχα ένας Συνδρομητής μπορεί να λαμβάνει οποιονδήποτε συνδυασμό ή και όλους τους τύπους δεδομένων, να «ακούει» δηλαδή, όλα τα γεγονότα. Τέλος υποστηρίζεται και η δυνατότητα να κατασκευάζονται «Ενδιάμεσα συστατικά» τα οποία είναι ταυτόχρονα και Εκδότες και Συνδρομητές. Σε αυτήν την περίπτωση, μπορούν να χρησιμοποιούνται και πολλαπλές διεπαφές για την έκδοση και να υλοποιούνται πολλαπλές διεπαφές για την επεξεργασία των δεδομένων. Είναι επίσης δυνατό, ένα συστατικό να λαμβάνει γεγονότα σαν Συνδρομητής σε μία διεπαφή, να τα επεξεργάζεται και να τα Εκδίδει σε μία διαφορετική διεπαφή.

5. Διεπαφής Χρήσης – θέση και μέγεθος παραθύρων

Συστατικά τρίτων κατασκευαστών που διαθέτουν διεπαφή χρήσης θα πρέπει να «θυμούνται» τη θέση και το μέγεθος του παραθύρου ή πλαισίου διαλόγου της διεπαφής χρήσης μεταξύ διαδοχικών ανοιγμάτων. Αυτό σημαίνει ότι όταν ο τελικός χρήστης του Βοηθήματος

Επικοινωνίας, μόνος του ή με τη βοήθεια του θεραπευτή του, ρυθμίζει το μέγεθος και τη θέση του παραθύρου της διεπαφής χρήσης του (ή των πολλαπλών διεπαφών χρήσης), θα πρέπει την επόμενη φορά που θα χρησιμοποιήσει το Βοήθημα Επικοινωνίας του να βρει τη διεπαφή χρήσης με τις ίδιες ρυθμίσεις που είχε κάνει.

Πρέπει λοιπόν οι κατασκευαστές να σχεδιάζουν έτσι τη διεπαφή χρήσης των συστατικών τους ώστε:

- Να υπάρχει η δυνατότητα προσαρμογής του μεγέθους (πλάτος και ύψος του παραθύρου) της διεπαφής χρήσης σύμφωνα με τις προτιμήσεις του χρήστη.
- Να υπάρχει η δυνατότητα τοποθέτησης μετακίνησης του παραθύρου της διεπαφής χρήσης στο σημείο της οθόνης που θέλει ο χρήστης.
- Να παραμένει η διεπαφή χρήσης στο μέγεθος και τη θέση που την προτιμά ο χρήστης μεταξύ διαδοχικών ανοιγμάτων του παραθύρου και, αν γίνονται νέες ρυθμίσεις, να τις διατηρεί.
- Αν είναι δυνατό, θα πρέπει και όλα τα στοιχεία που περιέχει το παράθυρο της διεπαφής χρήσης (κουμπιά, εικόνες, πλαίσια διαλόγου κλπ), να προσαρμόζουν το μέγεθος και τη θέση τους μέσα στο παράθυρο, ώστε να είναι πάντα όλα ορατά από το χρήστη, ακόμα και σε διαφορετικές ρυθμίσεις μεγέθους του γενικού παραθύρου.

Για να γίνουν όλα αυτά, οι κατασκευαστές μπορεί να επιλέξουν διάφορες τεχνικές. Μπορούν να κρατούν αρχεία ρυθμίσεων (ini files) που περιέχουν πληροφορίες για την αρχικοποίηση κάθε φορά της διεπαφής χρήσης, ή μπορούν να χρησιμοποιούν και το registry του συστήματος για να αποθηκεύουν αυτές τις πληροφορίες. Γενικά ο τρόπος υλοποίησης αυτών των απαιτήσεων είναι ελεύθερος και εδώ δίνονται απλά προτάσεις. Συνήθως είναι αρκετές τέσσερις μεταβλητές για να κρατήσουν τις πληροφορίες που χρειάζονται. Μία κρατά το πλάτος του παραθύρου, μια άλλη το ύψος, μια Τρίτη την κάθετη συνιστώσα της απόστασης της επάνω αριστερής γωνίας του παραθύρου από την πάνω αριστερή γωνία της οθόνης και η τελευταία κρατά την αντίστοιχη οριζόντια συνιστώσα. Πιο πολύπλοκη είναι η ικανοποίηση της προδιαγραφής που αφορά τα στοιχεία που περιέχει κάθε παράθυρο. Αν και τα περιεχόμενα του παραθύρου πρέπει να προσαρμόζονται στο μέγεθος του παραθύρου, χρειάζεται λίγος περισσότερος κώδικας για να υπολογίζεται κάθε φορά το μέγεθος και η θέσης που πρέπει να έχουν για να χωρούν μέσα σε αυτό. Και αυτή η υλοποίηση αφήνετε στην κρίση των προγραμματιστών.

6. Κώδικας Προγράμματος Εκκίνησης Βοηθημάτων Διαπροσωπικής Επικοινωνίας

Ο κώδικας του Προγράμματος Εκκίνησης Βοηθημάτων Διαπροσωπικής Επικοινωνίας είναι απαραίτητο να είναι γνωστός στους Προγραμματιστές, για να ξέρουν τη διαδικασία ενεργοποίησης η οποία θα εφαρμοστεί και στα δικά τους συστατικά (ειδικά στην περίπτωση που αυτά έχουν διεπαφή χρήσης.

```
Option Explicit
'•• ••••••• •••• ••••• •• •••••• "••••••";
Private Sub btnExit_Click()

'•••••••••••• • •••••
Unload Me
'••••••• • Communicator
```

```
End Sub
'•• ••••••• •••• •••••••••• • •••••;
Private Sub Form_Load()

    '••••• •••••••••• ••• •• •••••••••••••• ••• Component Catalog
    Dim oCat As COMAdminCatalog
    '••••• ••••••••••o oApp ••• •• ••••• •• ••••• ••• COM+ •••••••••
    Dim oApps As COMAdminCatalogCollection
    '••••• •••••••••• oAeneas ••• •• ••••• ••• •••••••• AENEAS
    Dim oAeneas As COMAdminCatalogObject
    '••••• •••••••••• oComs ••• •• ••••• •• ••••• ••• Components •••
Communicator
    Dim oComs As COMAdminCatalogCollection
    '••••• •••••••••• oCom ••• •••••••••••••• ••• component
    Dim oCom As COMAdminCatalogObject
    '••••• •••••• ••• Objects ••• •• ••••••• •• •••••••••• ••• component
    Dim component As New Collection

    '•••••• •• COM+ Catalog •• ••• •••••••••• oCat
    Set oCat = New COMAdminCatalog

    '••••••• •• ••••• ••• COM+ •••••••••
    Set oApps = oCat.GetCollection("Applications")

    '•••••• ••• •••••• (•••••••••••• aApp)•• ••• COM+ ••••••••• •••
••••••••.
    oApps.Populate

    '••••• •••• •••••• ••• ••• •••••••• AENEAS ••• •• •• ••• ••• •••••••
••• •••••••••• oAeneas
    For Each oAeneas In oApps
        If oAeneas.Name = "AENEAS" Then Exit For
    Next

    '•• ••• ••• ••• ••••• ••••• ••• •••••••• ••• Communicator
    If oAeneas Is Nothing Then
        Set oAeneas = oApps.Add
        oAeneas.Value("Name") = "AENEAS"
        oAeneas.Value("Activation") = 0
        oAeneas.Value("CreatedBy") = "ODYSSEAS Framework"
        oAeneas.Value("Description")   =   "Alternative   and   Augmentative
Communication Application based on ODYSSEAS framework"
        oAeneas.Value("RunForever") = True
        oApps.SaveChanges
        oCat.InstallEventClass "AENEAS", "Interface.dll", "", ""
        oApps.SaveChanges
```

```
        MsgBox "H •••••••• ••••••• ••••• ••••••••••••." + vbCrLf + vbCrLf
+ "•••• •••••••• •• ••••••••• •• components ••• •• •••••••••" + vbCrLf +
"•• ••••••• •••••••••••• ••••••••••••"
        Unload Me
        End
    End If

    '••••••• •• ••••• ••• components ••• ••••••••• AENEAS
    Set oComs = oApps.GetCollection("Components", oAeneas.Key)

    '•••••• ••• •••••• (•••••••••••• oCom) •• •• components ••• ••••••••
    oComs.Populate

    '•••••• •• Objects ••• •• •••••••••••• ••• •••• component (•••• •••
••••••••• •• ".Activate"
    For Each oCom In oComs
        If Right(oCom.Name, 9) = ".Activate" Then
        component.Add Item:=CreateObject(oCom.Name)
        End If
    Next

End Sub
```

Κώδικας 4: Το Πρόγραμμα Εκκίνησης Βοηθημάτων Επικοινωνίας και Ενεργοποίησης Συστατικών