Τμήμα Πληροφορικής
Σχολή Θετικών Επιστημών
Πανεπιστήμιο Ιωαννίνων

Μεταπτυχιακή Εργασία Ειδίκευσης

Ανακατασκευή συστημάτων καθοδηγούμενη από μοντέλα

Στέφανος – Κωνσταντίνος Πέτσιος

Επιβλέπων καθηγητής: Δημήτριος Ι. Φωτιάδης

Ιωάννινα, Σεπτέμβριος 2004

Ευχαριστίες

Θα ήθελα να ευχαριστήσω όλους όσους βοήθησαν για να διεκπεραιωθεί η εργασία αυτή και ιδιαίτερα τον κ. Δημήτριο Ι. Φωτιάδη Αναπληρωτή Καθηγητή του Τμήματος Πληροφορικής για την υποστήριξη αλλά και την πολύτιμη καθοδήγηση του στη παρούσα εργασία. Επίσης θα ήθελα να ευχαριστήσω τον κ. Απόστολο Ζάρρα Λέκτορα του Τμήματος Πληροφορικής για τις οδηγίες και τις κατευθύνσεις που μου έδωσε πάνω στο θέμα της εργασίας αλλά και για την υπομονή που επέδειξε στα όποια λάθη και αντιρρήσεις προήλθαν από την μεριά μου κατά την διεκπεραίωση της παρούσας εργασίας.

Ευχαριστώ τους γονείς μου Διομήδη - *Odette* και την αδερφή μου Ίρις για την συμπαράστασή τους και την προσπάθεια τους να μου παρέχουν τις καλύτερες δυνατές συνθήκες κατά την διάρκεια των σπουδών μου.

Επίσης ευχαριστώ για τις συμβουλές αλλά και για την βοήθεια που προσέφεραν με τον δικό τους ιδιαίτερο τρόπο τα "παιδιά" του *Instron* και ιδιαίτερα τους Αλέξανδρο, Δημήτρη, Νίκο και Πέτρο.

Εκτός από τους παραπάνω θέλω να ευχαριστήσω ονομαστικά μία σειρά από φίλους οι οποίοι έπαιξαν τον δικό τους καθοριστικό ρόλο κατά την διάρκεια αυτής της εργασίας: Αννίτα (annitoula), Βασίλη (adespotos), Βασίλη (xoriataki), Γιώργο (bexatzis), Γιώργο (roz), Ηλία (d_m_i), Θανάση (thanaxix), Παναγιώτη (maniakos) και τέλος τον I.H.C.

Στέφανος Κωνσταντίνος Πέτσιος

Σεπτέμβριος 2004

Περίληψη

Ένα πρόβλημα που αντιμετωπίζουμε σήμερα με τα υπάρχοντα πληροφοριακά συστήματα *business information systems* (BISs) είναι να τα ανακατασκευάσουμε με την χρήση νέων τεχνολογιών. Στην παρούσα εργασία παρουσιάζουμε μία μεθοδολογία καθώς και εργαλεία, που αυτοματοποιούν την ανακατασκευή παλαιών BIS σε νέα με χρήση νέων μεθόδων και τεχνικών βασισμένων σε διαφορετικές πλατφόρμες. Η προτεινόμενη μεθοδολογία εστιάζει κυρίως σε περιπτώσεις υπάρχοντος λογισμικού όπου έχουμε ελλιπή πληροφορία σχετικά με την αρχιτεκτονική του.

Η μεθοδολογία που παρουσιάζουμε έχει σαν σκοπό να εντοπίσει τα διάφορα υποσυστήματα του συστήματος και τις σχέσεις που ενώνουν αυτά. Για την μεθοδολογία χρειάζεται μόνο ο πηγαίος κώδικας του παλαιού συστήματος για να λειτουργήσει και είναι ανεξάρτητη από τις εμπλεκόμενες τεχνολογίες. Τα στάδια της μεθοδολογίας είναι το δυνατόν αυτοματοποιημένα και γίνεται χρήση γραφικών μοντέλων για την περιγραφή των συστημάτων. Επιτρέπουμε στους χρήστες να επέμβουν στα ενδιάμεσα στάδια για να προσδώσουμε ευελιξία στην μέθοδο. Επιπλέον όταν η μεθοδολογία μας όταν δεν καλύπτει νέες τεχνολογίες υπάρχει τρόπος να εισάγουμε περιγραφές και αναπαραστάσεις οι οποίες θα καλύπτουν το παραπάνω κενό. Είναι δηλαδή κατανοητό ότι μπορούμε να δημιουργούμε τις δικές μας επεκτάσεις σύμφωνα με τις δικές μας απαιτήσεις.

Περιεχόμενα

Κεφάλαιο 1: Εισαγωγή

Ένα πρόβλημα που αντιμετωπίζουμε σήμερα με τα υπάρχοντα πληροφοριακά συστήματα *business information systems* (*BISs*), που λειτουργούν σε περιβάλλον τοπικών κλειστών δικτύων, είναι να τα κάνουμε προσπελάσιμα από το *WEB*. Στην παρούσα εργασία παρουσιάζουμε μία μεθοδολογία καθώς και εργαλεία, που αυτοματοποιούν την ανακατασκευή παλαιών *BIS* σε νέα με χρήση νέων μεθόδων και τεχνικών βασισμένων σε διαφορετικές πλατφόρμες. Η προτεινόμενη μεθοδολογία εστιάζει κυρίως σε περιπτώσεις υπάρχοντος λογισμικού όπου έχουμε ελλιπή πληροφορία σχετικά με την αρχιτεκτονική του. Ειδικότερα μας ενδιαφέρουν οι περιπτώσεις όπου οι πληροφορίες σχετικά με τον τρόπο υλοποίησης των υποσυστημάτων του είναι ανεπαρκείς. Τέτοιες περιπτώσεις με ελλιπή στοιχεία συναντώνται σήμερα αρκετά συχνά. Αυτό οφείλεται κυρίως στις δυνατότητες που προσφέρουν οι διάφορες πλατφόρμες όπου η χρήση *middleware* και *Components Of the Shelves* (*COTS*) συνδυασμένα με *CASE tools* είναι δυνατό να παρασύρει τους σχεδιαστές στο να κατευθυνθούν απευθείας στη φάση της υλοποίησης του έργου, αφήνοντας ημιτελή τη φάση της σχεδίασης αυτού.

Η εργασία αφορά την ανακατασκευή ενός *BIS* το οποίο να κάνει χρήση νέων τεχνολογιών, όπως για παράδειγμα η χρήση του *WEB*. Πολλά συστήματα που χρησιμοποιούνται σήμερα έχουν ολοκληρωθεί με την χρήση *COTS* όπως τα *Borland Delphi, Borland C-Builder, Borland J-Builder, Microsoft Visual Basic*, κτλ και *middleware* πλατφόρμες όπως *CORBA, DCOM* κτλ. Για λόγους συντομίας θα χρησιμοποιούμε τον όρο τεχνολογικές

πλατφόρμες (*technology platforms*) όταν αναφερόμαστε στις πλατφόρμες *middleware* και *COTS* που χρησιμοποιήθηκαν.

1.1 Ανάπτυξη BIS

Η ανάπτυξη ενός *BIS* περιλαμβάνει την συλλογή και την ανάλυση πληροφοριών και απαιτήσεων από τον εμπλεκόμενο οργανισμό. Η ανάλυση απαιτήσεων τις διαχωρίζει σε λειτουργικές και μη λειτουργικές. Οι λειτουργικές περιγράφουν τις διεργασίες που πρέπει να πραγματοποιούνται από το *BIS*, ενώ οι μη λειτουργικές αναφέρονται σε πιο γενικές έννοιες, όπως οι επιδόσεις, η αξιοπιστία και η διαθεσιμότητα του συστήματος. Μετά την ανάλυση απαιτήσεων ακολουθεί ο απαραίτητος αρχιτεκτονικός σχεδιασμός του *BIS*. Στην αρχή ο σχεδιασμός αυτός είναι ανεξάρτητος από την τεχνολογική πλατφόρμα (*platform independent*) που τελικά θα χρησιμοποιηθεί. Για να περιγράψουμε ένα τέτοιο σχέδιο χρειάζεται ένα *Platform Independent Representation* (*PIR*). Το *PIR* αποτελείται από μία συλλογή απεικονίσεων και συσχετίσεων, ικανών να περιγράψουν ένα σύστημα με την χρήση αφηρημένων όρων. Ένα τέτοιο σχέδιο, όπως είναι φυσικό, δεν περιέχει πληροφορία που να συνδέεται άμεσα με την τεχνολογική πλατφόρμα που τελικά θα επιλεγεί. Το μοντέλο, που ακολουθεί τις προδιαγραφές του *PIR* το ονομάζουμε *Platform Independent Model (PIM).* Το επόμενο βήμα είναι η επιλογή της τεχνολογικής πλατφόρμας. Με βάση το *PIR* που έχουμε κατασκευάσει το επαναπροσδιορίζουμε κατάλληλα ώστε τα στοιχεία που το αποτελούν να διαμορφωθούν με σκοπό να περιέχουν και πληροφορία που σχετίζεται με τη τεχνολογική πλατφόρμα (*platform specific*). Σκοπός αυτής της διαδικασίας είναι, να παράγουμε ένα *Platform Specific Model* (*PSM*), το οποίο βασίζεται στο αντίστοιχο *Platform Specific Representation* (*PSR*) νέας τεχνολογικής πλατφόρμας που τελικά επιλέγουμε. Η όλη διαδικασία ανάπτυξης του συστήματος φυσικά ολοκληρώνεται με την παραγωγή τμήματος του πηγαίου κώδικα βασιζόμενη στο παραπάνω *PSM*. Η υλοποίηση μπορεί να στηριχτεί επάνω σε κάποια -*platform specific- CASE tools*, όπως για παράδειγμα η χρήση γραφικού περιβάλλοντος το οποίο να διευκολύνει την επιλογή και την χρήση των κατάλληλων στοιχείων μίας τεχνολογικής πλατφόρμας. Η χρήση *platform specific CASE tools* διευκολύνει την ανάπτυξη *BIS* και συμβάλει θετικά στην επαναχρησιμοποίηση κώδικα, καθώς επίσης επιτρέπει τη βελτίωση της ποιότητας του *BIS* και το χρόνο που απαιτείται για την υλοποίησή του. Η δυνατότητα να μειώσουμε τον χρόνο που απαιτείται για να φτάσει ένα προϊόν στην αγορά είναι πολύ δελεαστική για τους *project*

managers, τους σχεδιαστές και τους προγραμματιστές. Πιο συγκεκριμένα το χαρακτηριστικό της μείωσης του χρόνου παραγωγής, ευνοεί το πέρασμα από την ανάλυση απαιτήσεων κατευθείαν στη φάση υλοποίησης με απώτερο σκοπό την μείωση του χρόνου υλοποίησης και των απαιτούμενων πόρων.

Τα αποτελέσματα μίας τέτοιας προσέγγισης γίνονται άμεσα αντιληπτά όταν ερχόμαστε αντιμέτωποι με μία πιθανή ανακατασκευή (*re-engineering*) ενός υπάρχοντος συστήματος. Το *PSM* που περιγράφει την αρχιτεκτονική ενός *BIS* θα πρέπει τώρα να το εξάγουμε από τον πηγαίο κώδικα του συστήματος, του οποίου το μέγεθος και πολυπλοκότητα τις περισσότερες φορές είναι μεγάλο. Στην συνέχεια θα πρέπει να αφαιρέσουμε από το *PSM* όλη την πληροφορία, η οποία σχετίζεται άμεσα με τη τεχνολογική πλατφόρμα που έχει χρησιμοποιηθεί. Το αποτέλεσμα των παραπάνω θα είναι το *PIM* του συστήματος, το οποίο και θα χρησιμοποιήσουμε για την κατασκευή του νέου *PSM* σύμφωνα με τις νέες τεχνολογικές πλατφόρμες που θα επιλέξουμε και που θα πρέπει να ικανοποιούν τα νέα χαρακτηριστικά που θέλουμε να προσδώσουμε στο σύστημα.

1.2 Περιεχόμενο εργασίας

Στόχος της εργασίας λοιπόν είναι: η κατά το δυνατόν αυτοματοποίηση της ανακατασκευής ενός συστήματος, ανεξάρτητα από τις τεχνολογικές πλατφόρμες (αρχική τελική) και βασιζόμενοι μόνο στην πηγαίο κώδικα του δοθέντος συστήματος. Για τη μεθοδολογία που προτείνουμε κατασκευάσαμε μία συλλογή από εργαλεία. Τα εργαλεία αυτά παρέχουν τη δυνατότητα της αυτόματης παραγωγής *PSM* μοντέλων και *skeleton code* τμημάτων του τελικού συστήματος με χρήση του πηγαίου *Delphi* κώδικα του παλαιού συστήματος.

1.3 Οργάνωση της εργασίας

Αρχικά γίνεται αναφορά στις βασικές αρχές της ανακατασκευή λογισμικού (Κεφάλαιο 2) καθώς και στις σχετικές έρευνες οι οποίες έχουν γίνει στο αντικείμενο αυτό. Στην συνέχεια παρουσιάζεται η μεθοδολογία που προτείνουμε εμείς καθώς και η περιγραφή της αναπαράστασης

που προτείνουμε για την περιγραφή μοντέλων που είναι ανεξάρτητα τεχνολογικής πλατφόρμας (*PIR*). Επίσης περιγράφουμε και δύο αναπαραστάσεις (*PSRs*) που προτείνονται για την περιγραφή μοντέλων συστημάτων που βασίζονται σε *Delphi* και *PhP* αντίστοιχα (Κεφάλαιο 3). Οι δύο αυτές αναπαραστάσεις χρησιμοποιήθηκαν σε ένα σύστημα του οποίου η ανακατασκευή βασίστηκε στην προτεινόμενη μεθοδολογία. Στο κεφάλαιο 4 περιγράφεται η ανάλυση ενός υπαρκτού *BIS* συστήματος που ήταν χτισμένο σε *Delphi* και ανακατασκευάστηκε με βάση τη *PhP*. Τέλος παρουσιάζονται τα αποτελέσματα τα οποία προέκυψαν (Κεφάλαιο 5) από το δικό μας *case study*. Τα αποτελέσματα συνοδεύονται από την κατάλληλη αξιολόγηση, ώστε να διαπιστωθεί η ελαστικότητα και η χρηστικότητα της μεθόδου.

Κεφάλαιο 2: Χρησιμοποιούμενοι μέθοδοι ανακατασκευής

Υπάρχουν πολλά συστήματα λογισμικού τα οποία είναι εγκατεστημένα σε επιχειρήσεις και δημόσιους οργανισμούς και χρησιμοποιούνται εδώ και πολλά χρόνια. Η απλή συντήρηση των συστημάτων αυτών δεν είναι πάντοτε επαρκής και απαιτείται εκσυγχρονισμός και ανακατασκευή. Με τον εκσυγχρονισμό του συστήματος η μελλοντική υποστήριξη και συντήρησή του γίνεται πιο εύκολη. Άλλοι στόχοι, εκτός από τον εκσυγχρονισμό, είναι να παράγουμε συστήματα καλύτερα δομημένα έτσι ώστε η νέα τους δομή να διευκολύνει την χρήση τους, να την κάνει πιο πρακτική, και να καθιστά ευκολότερη την ολοκλήρωση μιας πιθανής μελλοντικής ανακατασκευής τους. Επίσης, ο εκσυγχρονισμός βελτιώνει την ποιότητα του συστήματος.

Ένας τρόπος για τον εκσυγχρονισμό παλαιών συστημάτων είναι η μεταφορά τους σε μία νέα τεχνολογική πλατφόρμα ή γλώσσα. Αυτό είναι απαραίτητο για παράδειγμα εάν το πρωτότυπο σύστημα χρησιμοποιούσε απαρχαιωμένη πλατφόρμα, οπότε λόγω πιθανής έλλειψης υποστήριξης της παλαιάς πλατφόρμας θα είχε επιπτώσεις και στο συγκεκριμένο *BIS*. Στον εκσυγχρονισμό για τη μετατροπή τεχνολογικής πλατφόρμας, χρειαζόμαστε τόσο μεθόδους για την ανακατασκευή λογισμικού (σχετικά με δομές, απαιτήσεις κτλ), όσο και κάποια μέθοδο μετατροπής - μεταφοράς (με *language conversion*) από τη μία πλατφόρμα σε μία άλλη (εάν αυτό είναι δυνατόν), δηλαδή *source-to-source translation*. Εάν μεταφέρουμε ανάμεσα σε διαφορετικά

μοτίβα γλωσσών, παράδειγμα *procedural* σε *object-oriented*, πρέπει να βρούμε τα αντικείμενα από το ακολουθιακό πρόγραμμα. Σε κάθε περίπτωση, είτε παραμένει είτε αλλάζει το μοτίβο της πλατφόρμας θα χρειαστεί να αναπτύξουμε μεθόδους που να ασχολούνται με τη μετατροπή *source-to-source*.

2.1 Βασικές αρχές του της ανακατασκευής λογισμικού

Αυτή η ενότητα ασχολείται γενικότερα με την ανακατασκευή λογισμικού (*re-engineering*) και ειδικότερα με την ανίχνευση δεδομένων. Αρχικά εισάγουμε την απαραίτητη ορολογία. Κατόπιν παρουσιάζουμε μερικές μεθόδους και εργαλεία που χρησιμεύουν στην ανακατασκευή και έπειτα αναφέρουμε την ανίχνευση αντικειμένων από τον πηγαίο κώδικα του παλαιού BIS. Αναφέρουμε κάποια παραδείγματα ανακατασκευής και στη συνέχεια μελετούμε τη σχέση ανάμεσα σε αυτό και στην ανίχνευση αντικειμένων. Τελικά σχολιάζουμε τα προβλήματα αυτών των προσεγγίσεων.

2.1.1 Ορολογία της ανακατασκευής συστημάτων

Δίδονται στην συνέχεια ορισμοί που έχουν δοθεί από τους Chikofsky και Cross [Chikofsky90a].

Συντήρηση Λογισμικού

Ο *ANSI* ορισμός για τη συντήρηση λογισμικού (ANSI/IEEE Std 729-1983) είναι: "*Η τροποποίηση ενός συστήματος μετά την παράδοση του για να διορθωθούν σφάλματα, να βελτιωθούν οι επιδόσεις ή άλλα χαρακτηριστικά του για να προσαρμοσθεί το προϊόν σε ένα διαφοροποιημένο περιβάλλον.*" Το πρώτο βήμα στην συντήρηση λογισμικού είναι να μελετηθεί το σύστημα με σκοπό την κατανόησή του. Οι μέθοδοι αντίστροφης ανακατασκευής μπορούν να χρησιμοποιηθούν για να υποστηρίξουν τη διαδικασία συντήρησης. Έτσι ένα τμήμα της διαδικασίας συντήρησης αποτελεί και η αντίστροφη ανακατασκευή, η οποία διευκολύνει στην κατανόηση του λογισμικού με σκοπό να γίνουν οι επιθυμητές αλλαγές. Η συντήρηση μπορεί να θεωρηθεί ως ανάπτυξη λογισμικού με επαναχρησιμοποίηση [Victor90].

Πρόσθια κατασκευή (Forward engineering)

Η πρόσθια κατασκευή είναι η παραδοσιακή μέθοδος η οποία αρχίζει με την ανάλυση απαιτήσεων του συστήματος, έπειτα γίνεται ο σχεδιασμός του και από το σχεδιασμό οδηγούμαστε στη «συμπαγή» υλοποίηση του συστήματος. Ουσιαστικά ο όρος *forward engineering* σημαίνει το ίδιο με *engineering*. Το επίθετο *forward* προστέθηκε για τον διαχωρισμό από το *reverse engineering*.

Αντίστροφη κατασκευή (Reverse – engineering)

Στην αντίστροφη κατασκευή η υπό εξέταση εξαγόμενη πληροφορία για ένα σύστημα είναι μία σκιαγράφηση - σύνοψη του συστήματος. Για παράδειγμα η διαδικασία αντίστροφης κατασκευής εξετάζει ή βρίσκει αφηρημένες έννοιες ή σχεδιαστικές αποφάσεις, οι οποίες έγιναν στο επίπεδο υλοποίησης. Η αντίστροφη κατασκευή μπορεί να εφαρμοστεί σε οποιοδήποτε επίπεδο – στάδιο της ροής εργασιών που ακολουθούμε για την ολοκλήρωση του συστήματος. Η αντίστροφη κατασκευή δεν περιέχει την μετατροπή του συστήματος. Είναι μία διαδικασία διερεύνησης και δεν γίνονται ούτε σχεδιαστικές αλλαγές ούτε κάποιες αλλαγές στο επίπεδο υλοποίησης.

Αναδόμηση (Restructuring)

Η αναδόμηση είναι η μετατροπή από μία μορφή αναπαράστασης σε μία άλλη χρησιμοποιώντας το ίδιο αφαιρετικό επίπεδο. Η μετατροπή διατηρεί την εξωτερική συμπεριφορά του συστήματος. Η αναδόμηση συνήθως χρησιμοποιείται στη φάση υλοποίησης για τη μετατροπή κώδικα μη δομημένης μορφής σε δομημένη μορφή (για παράδειγμα μετατροπή *go-to* εντολών για τον έλεγχο δομών). Επιπλέον, η αναδόμηση μπορεί να χρησιμοποιηθεί και σε άλλα στάδια, για παράδειγμα για την αναδιοργάνωση σχεδιαστικών ή δομικών αποφάσεων.

Ανακατασκευή (re-engineering)

Η ανακατασκευή είναι η εξέταση και τροποποίηση ενός συστήματος με σκοπό την μετατροπή του σε μία νέα μορφή και στην συνέχεια την υλοποίηση της νέας μορφής. Έτσι η ανακατασκευή γενικά συμπεριλαμβάνει την αντίστροφη κατασκευή (για να επιτύχουμε μία πιο αφηρημένη περιγραφή του συστήματος), ακολουθούμενη από την αναδόμηση. Ενδέχεται να συμπεριλαμβάνει τροποποιήσεις παίρνοντας υπόψη νέες απαιτήσεις που δεν συναντώνται στο

αρχικό σύστημα. Άλλοι όροι που έχουν χρησιμοποιηθεί για την ανακατασκευή είναι *renovation* και *reclamation*.

2.1.2 Η ανακατασκευή ενός συστήματος

Ο πιο εύκολος τρόπος για να αρχίσει κάποιος να χρησιμοποιεί μια νέα πλατφόρμα είναι να "κλείσει τον παλιό κώδικα σε ένα κουτί" και να το τεμαχίσει σε υποομάδες (υποσυστήματα) σύμφωνα με τη νέα πλατφόρμα που θα επιλεγεί. Οι λειτουργίες (*functionality*) του νέου συστήματος αναμένουμε να διατηρηθούν. Το παλαιό σύστημα περιέχει υποσυστήματα τα οποία και στο καινούργιο είναι χρήσιμα. Επειδή συχνά είναι δύσκολο να ξεχωρίσουμε τα υποσυστήματα και να τα χρησιμοποιήσουμε ξεχωριστά, το παλαιό σύστημα το χρησιμοποιούμε ολόκληρο.

Η μετάβαση από ένα σύστημα με ακολουθιακή γλώσσα σε ένα *object-oriented* με απευθείας *source-to-source* μετατροπή δεν είναι εύκολη [Gall95]. Έτσι χρειαζόμαστε μεθόδους αντίστροφης κατασκευής για να αναγνωρίσουμε αντικείμενα από τον πηγαίο κώδικα. Αφού γίνει η αναγνώριση των αντικειμένων η μετατροπή μπορεί να πραγματοποιηθεί με βάση με τον πηγαίο κώδικα και τα αντικείμενα που αναγνωρίστηκαν. Τα αντικείμενα είναι δυστυχώς δύσκολο να αναγνωρισθούν, ιδιαίτερα εάν τα συστήματα δεν είναι σχεδιασμένα σύμφωνα με τις *object-oriented* μεθόδους σχεδιασμού. Επιπλέον οι κλάσεις που αναγνωρίζονται συνήθως δεν σχηματίζουν σχέσεις με υποκλάσεις. Εκτός όμως από την ανίχνευση των αντικειμένων σημαντικό είναι να βρεθούν και άλλα *object-oriented* χαρακτηριστικά όπως τα *inheritance* και *polymorphism* [Harsu00].

Η πραγματική μετατροπή πλατφόρμας συναντά συχνά σοβαρά προβλήματα. Είναι δύσκολο να πραγματοποιηθεί πλήρης *source-to-source* μετατροπή. Συνήθως κάποιες δομές από τις αρχικές πλατφόρμες δεν αναγνωρίζονται και έτσι δεν μετατρέπονται, πιθανότατα επειδή δεν είχαν ακριβή ισοδύναμα στοιχεία στη νέα πλατφόρμα. Συνεπώς τα προγράμματα χρειάζονται κάποια επεξεργασία, είτε πριν γίνει η μετατροπή για αφαίρεση της χρήσης τέτοιων δομών, είτε μετά, για να διορθώσουμε τα λάθη της μετατροπής. Αν και θεωρητικά θα ήταν δυνατό να εκφράσουμε όλες τις αρχές που χρησιμοποιεί η πλατφόρμα του παλαιού συστήματος σε αντίστοιχες αρχές της νέας πλατφόρμας, τα προγράμματα – συστήματα που θα προέκυπταν θα ήταν πολύπλοκα (και

δύσμορφα) και θα ήταν δύσκολο να κατανοηθούν κάτι που θα οδηγούσε στη δύσκολη συντήρησή τους.

Ο εκσυγχρονισμός των παλαιών συστημάτων απαιτεί διάσχιση (*parsing*), όταν τα προγράμματα των συστημάτων εμφανίζονται με τη μορφή δένδρου διάσχισης (*parse-tree*) και συντακτικού δένδρου (*syntax-tree*). Τα παλαιά συστήματα ενδέχεται να είναι μεγάλα και συνεπώς αυτά τα δένδρα περιμένουμε να είναι εξίσου εκτεταμένα, κάτι που μπορεί από μόνο του να δημιουργήσει προβλήματα. Λύση σε τέτοια προβλήματα δίνει μία προσέγγιση γνωστή ως *piecewise translation*, η οποία χωρίζει τις δομές δεδομένων σε μικρότερα τμήματα και τα επεξεργάζεται ξεχωριστά [].

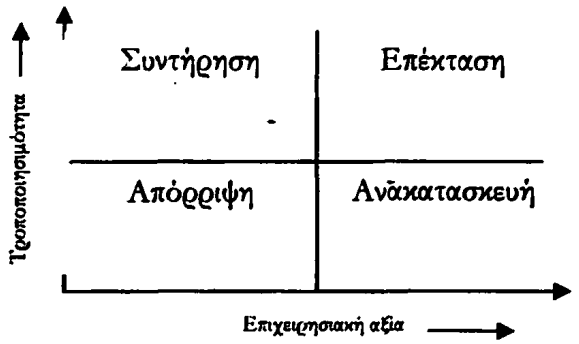
2.1.3 Η ανακατασκευή ενός συστήματος σε σχέση με την συντήρησή του

Η ανακατασκευή βελτιώνει την ποιότητα του λογισμικού και κάνει πιο εύκολη τη μετέπειτα συντήρηση. Αυτή η ιδιότητα είναι πιο φανερή εάν ανιχνευτούν επαναχρησιμοποιήσιμα στοιχεία βάσει των οποίων το σύστημα θα ανακατασκευαστεί σε μία μορφή η οποία υποστηρίζει καλύτερα την επαναχρησιμοποιησιμότητα και τη συντήρηση. Η ποιότητα των συστημάτων που έχουν προκύψει με την εφαρμογή της ανακατασκευής, θεωρείται γενικά καλύτερη από αυτή του αρχικού συστήματος.

Η ανακατασκευή λογισμικού έχει άμεση σχέση με την συντήρηση. Ο *Swanson* [Swanson76] δίνει τις τρεις κατηγορίες προβλημάτων που αντιμετωπίζει η συντήρηση λογισμικού. Πρώτον, τα λάθη στις προδιαγραφές του αρχικού σχεδιασμού και στην υλοποίηση πρέπει να διορθωθούν. Δεύτερον, τα δεδομένα και το περιβάλλον λειτουργίας ενδέχεται να αλλάζουν. Τρίτον, οι επιδόσεις του παλαιού συστήματος πρέπει να διατηρούνται ή να βελτιώνονται. Ο ANSI ορισμός για την συντήρηση λογισμικού καλύπτει αυτές τις τρεις κατηγορίες. Εκτός από τους τρεις παραπάνω λόγους αναγνωρίστηκε και ένας τέταρτος λίγο αργότερα (*preventive maintenance*): το σύστημα προσαρμόζεται για να βελτιώσει μελλοντική συντηρησιμότητα, αξιοπιστία ή για να υπάρχουν καλύτερες συνθήκες για μελλοντικές επεκτάσεις. Η προληπτική συντήρηση είναι στενά συνδεδεμένη με την ανακατασκευή. Επιπλέον όλες οι άλλες μορφές συντήρησης μπορούν να βασισθούν σε εργαλεία ανακατασκευής ενός συστήματος.

Συστήματα τα οποία συντηρούνται συνήθως δεν συνοδεύονται από τα απαραίτητα επεξηγητικά εγχειρίδια ή τα εγχειρίδια που τα συνοδεύουν δεν ανανεώνονται. Γενικά έχουν ελλιπή τεκμηρίωση. Οι απαιτήσεις, ο σχεδιασμός, η συντήρηση, και άλλου τύπου πληροφορία σχετικά με το λογισμικό μπορούν να ανακτηθούν από τους προγραμματιστές, αλλά αυτοί δεν είναι πάντα στη διάθεσή μας. Έτσι η ανακατασκευή μπορεί να χρησιμοποιηθεί στην ανάκτηση πληροφορίας για να κατασκευάσουμε γραφήματα της δομής του συστήματος. Οι μέθοδοι ανακατασκευής μπορούν να χρησιμοποιηθούν για την αναπαραγωγή τέτοιων γραφημάτων και κατά συνέπεια στην συντήρηση.

Παρόλα αυτά η ανακατασκευή δεν είναι πάντα η καλύτερη επιλογή. Το Σχήμα 1 δείχνει σε ποιες περιπτώσεις είναι προτιμητέα η ανακατασκευή ενός συστήματος [Jackson94]. Εάν το υπό εξέταση σύστημα είναι δύσκολο να αλλαχτεί, αλλά έχει μεγάλη αξία για την επιχείρηση, τότε είναι αναγκαίο να γίνουν σημαντικές αλλαγές και το σύστημα αξίζει να ανακατασκευαστεί. Ένα σύστημα με υψηλή τροποποιησιμότητα ή με μικρή αξία για την επιχείρηση, δεν χρειάζεται να ανακατασκευαστεί.



Σχήμα 1: ***Πίνακας απόφασης ανακατασκευής έναντι συντήρησης για ένα σύστημα λογισμικού.***

2.2 Μέθοδοι και εργαλεία που χρησιμοποιούνται στην ανακατασκευή

2.2.1 Μέθοδοι ανακατασκευής

Υπάρχουν πολλοί τρόποι για την ανακατασκευή σε ένα σύστημα. Ο κοινός στόχος των μεθόδων ανακατασκευής είναι η αναζήτηση σχέσεων μεταξύ των υποσυστημάτων (ανάλυση εξαρτήσεων) και να αποδίδουν νόημα στις σχέσεις αυτές.

Η ανάλυση εξαρτήσεων αναδεικνύει τις σχέσεις ανάμεσα σε δομικά στοιχεία γλωσσών (*modules, data objects, procedures, functions*). Η πιο απλή μέθοδος τέτοιου είδους βασίζεται στην αντιστοίχηση όρων που περιγράφουν τα παραπάνω δομικά στοιχεία. Οι όροι αυτοί είναι διαθέσιμοι σε μερικούς μεταφραστές και σχετίζονται με την γραμματική της αντίστοιχης γλώσσας. Η ανάλυση εξαρτήσεων δείχνει πού αναφέρονται διαφορετικά γνωρίσματα και έτσι μειώνει την προσπάθεια να τα αναζητήσουμε εντός ενός συγκεκριμένου (δοθέντος) προγράμματος. Άλλος απλός τρόπος να πάρουμε πληροφορίες εξαρτήσεων είναι να κατασκευάζουμε ένα γράφημα εξαρτήσεων (*dependence graph*) στο οποίο φαίνεται πώς συνδέεται μία διαδικασία με μεταβλητές του συστήματος. Επίσης μία άλλη κοινή μέθοδος στην ανακατασκευή είναι η κατασκευή γράφων ροής (*flow graphs*). Αρχικά τα διαγράμματα ροής χρησιμοποιήθηκαν για μεταφραστές και ειδικότερα για την βελτιστοποίηση του πηγαίου κώδικα. Προφανώς μπορούν να είναι χρήσιμα στην ανακατασκευή, επειδή μπορούν εύκολα να βοηθήσουν στη κατανόηση των προγραμμάτων. Τα διαγράμματα ροής χωρίζονται σε δύο είδη: στα *control flow* και στα *data flow* γραφήματα. Για την κατασκευή ενός *control flow* γραφήματος οι δηλώσεις - αναφορές ενός προγράμματος οργανώνονται σε βασικά *blocks*. Ένα βασικό *block* είναι μία ακολουθία από συνεχόμενες αναφορές όπου η ροή εκτέλεσης ξεκινά από την αρχή του *block* και τελειώνει στο τέλος του χωρίς παύση ή πιθανότητα διακλάδωσης εκτός από του τέλους του. Τα βασικά *block* είναι κόμβοι του *control flow* γραφήματος. Οι ακμές απεικονίζουν μεταφορά ελέγχου ανάμεσα σε βασικά *blocks*. Για παράδειγμα ένα *if-statement* αναπαριστάνεται από ένα υπογράφημα που διακλαδίζεται σε έναν κόμβο συνθήκης σε 2 κόμβους, ο ένας αποτελεί το βασικό *block* για το αληθές ενδεχόμενο της συνθήκης και ο άλλος το βασικό *block* του ψευδούς ενδεχόμενου. *Data flow* εξαρτήσεις επέρχονται ανάμεσα σε *data objects*, όταν η τιμή που ανήκει σε ένα αντικείμενο μπορεί να χρησιμοποιηθεί από ένα άλλο αντικείμενο.

Η μέθοδος *program slicing* [Weiser81] βοηθά στην καλύτερη κατανόηση ενός προγράμματος, έτσι μπορεί και αυτό να χρησιμοποιηθεί στην ανακατασκευή ενός συστήματος. *Program slicing* είναι μία αποσύνθεση βασισμένη στην ανάλυση των *data flow* και *control flow* διαγράμματα. Ένα *programming slice* είναι όλα τα *statements* και *predicates* του προγράμματος τα οποία ενδέχεται να επηρεάσουν την τιμή μίας μεταβλητής. Το *program slicing* έχει παρουσιαστεί σε διάφορες μορφές [Venkatesh91]. Για παράδειγμα το *forward slice* αποτελείται μόνο από *statements* τα οποία επηρεάζονται από τη υπό εξέταση μεταβλητή [Horwitz 90]. Έτσι ο αρχικός

ορισμός για το *program slice* [Weiser81] ονομάζεται ως *backward slice* [Venkatesh91]. Στην ανίχνευση αντικειμένων το *program slicing* μπορεί να χρησιμοποιηθεί για παράδειγμα, εάν μια ακολουθιακή ρουτίνα θα πρέπει να σπάσει σε μεθόδους που θα ανήκουν σε αυτά αντικείμενα ενός αντικειμενοστρεφούς κώδικα. Τα αντικείμενα αυτά θα περιέχουν χαρακτηριστικά που θα αντιστοιχούν στις μεταβλητές του προγράμματος.

2.2.2 Εργαλεία ανακατασκευής συστημάτων

Πολλά διαφορετικά εργαλεία που υποστηρίζουν την ανακατασκευή έχουν αναφερθεί στη βιβλιογραφία. Τα εργαλεία ανακατασκευής έχουν διαφορετικούς σκοπούς και στόχους. Μπορούν να βρουν συγκεκριμένες δομές στον κώδικα του συστήματος [Paul94], να παράγουν νέα κείμενα σχετικά για το σύστημα [Murphy96] ή να οπτικοποιούν τον κώδικα και να παρέχουν ειδικούς επεξεργαστές για την περιήγηση στον κώδικα του συστήματος [Linos94]. Στην ενότητα αυτή δεν θα ξεχωρίσουμε τα εργαλεία ανακατασκευής από αυτά της αντίστροφης κατασκευής.

Μερικά από τα εργαλεία ανακατασκευής βασίζονται στη λεκτική δομή του κώδικα του συστήματος και μερικά άλλα απαιτούν διάσχιση (*parsing*) αυτού. Τα εργαλεία που βασίζονται στη λεκτική δομή είναι συνήθως απλά και εύκολα να υλοποιηθούν. Ωστόσο η εύρεση μερικών δομών μπορεί να απαιτεί διάσχιση του κώδικα. Τα εργαλεία που βασίζονται στη διάσχιση, είναι μεν βολικά για ανίχνευση *programming oriented* εργασιών, δεν μπορούν όμως να χρησιμοποιηθούν για εξόρυξη σημασιολογικής πληροφορίας [Biggerstaff94].

Στην συνέχεια θα αναφερθούμε σε στοιχεία εργαλείων και τις ιδιαιτερότητές τους. Ο *Cordy* δείχνει πώς να γίνει αποκοπή των άχρηστων κομματιών των προγραμμάτων για να γίνουν περισσότερο κατανοητά [Cordy90]. Για παράδειγμα ο κορμός μίας *control structure* μπορεί να αποκοπεί και να δείχνει μόνο την επικεφαλίδα και την κατάληξη της αποκομμένης *structure*. Αυτό κάνει την περιήγηση του προγράμματος ευκολότερη. Παρόμοια εργαλεία χρησιμοποιούνται σε άλλους τομείς εκτός της ανακατασκευής.

Τα εργαλεία της ανακατασκευής τυπικά βοηθούν στο να βρεθούν τα σημεία του συστήματος που θα τροποποιηθούν. Όμως οι πραγματικές τροποποιήσεις θα γίνουν με την

επέμβαση κάποιου ανθρώπου. Ένα σύνολο από εργαλεία που σχετίζονται με την ανακατασκευή παρουσιάζονται στο [FAMOOS].

2.2.3 Ανίχνευση αντικειμένων στον πηγαίο κώδικα

Υπάρχουν πάρα πολλοί τρόποι για την ανίχνευση αντικειμένων από τον πηγαίο κώδικα του παλιού *BIS*. Οι *Jacobson* και *Lindstörm* [Jacobson91] δείχνουν πώς παλιά συστήματα μπορούν βαθμιαία να υποστούν ανακατασκευή σε μία αντικειμενοστρεφή αρχιτεκτονική. Μελετούν διαφορετικές περιπτώσεις ανάλογα για το εάν οι αλλαγές εστιάζονται σε τεχνικές ανάπτυξης ή στις λειτουργίες τους. Ερευνούν τρεις διαφορετικές περιπτώσεις:

- Ριζική αλλαγή της τεχνικής ανάπτυξης και καμία αλλαγή στις λειτουργίες.
- Αλλαγές στη τεχνική ανάπτυξης και καμία αλλαγή στις λειτούργίες.
- Αλλαγές στις λειτουργίες.

Εάν η πλατφόρμα ανάπτυξης αλλάζει μόνο μερικώς, το νέο κομμάτι και το υπόλοιπο του παλιού συστήματος πρέπει να προσαρμοστούν ή να συνδεθούν. Είναι απαραίτητη η ύπαρξη ενδιάμεσου λογισμικού (*interface*) μεταξύ των δύο αυτών τμημάτων για την επίτευξη επικοινωνίας.

Ο *Newcomb* περιγράφει μία αυτοματοποιημένη διεργασία για την ανακατασκευή ακολουθιακών προγραμμάτων σε αντικειμενοστρεφή [Newcomb 95]. Τα αντικειμενοστρεφή μοντέλα που χρησιμοποιεί βασίζονται σε μηχανές καταστάσεων ονομάζονται *hierarchical object-oriented state-machine models.* Ο *Newcomb* περιγράφει διάφορους τρόπους ανάλυσης. Η *Alias analysis* για παράδειγμα εξετάζει εγγραφές και πεδία για να βρει εμφανίσεις εγγραφών που έχουν διαφορετικό όνομα αλλά ίδια δομή. Η φόρμα μιας εγγραφής που έχει διαφορετικές εμφανίσεις ονομάζεται *collision former.* Οι εγγραφές που είναι *collision former* ονομάζονται *aliases.* Ένας χάρτης από *alias* είναι ένα *relation* του οποίου το *domain* είναι *collision former* και το *range* του είναι ένα σύνολο εγγραφών από *alias.* Αυτός κατασκευάζει *control flow* γραφήματα, ένα *data flow* γράφημα και ένα γράφημα που περιγράφει τις μεταβάσεις των διαδικασιών. Το γράφημα καταστάσεων αποτελείται από μία αρχική κατάσταση, μία τελική κατάσταση και ένα σύνολο από ενδιάμεσες καταστάσεις που συνδέονται με μεταβάσεις καταστάσεων. Μία μετάβαση κατάστασης ορίζεται από μία σειρά συνθηκών ελέγχου που συνδέονται από μία σειρά από ενέργειες. Ο πίνακας

μεταβάσεων των καταστάσεων απεικονίζει μία ή περισσότερες καταστάσεις, καθώς και τις συνθήκες και τις ενέργειες που γίνονται κατά τις μεταβάσεις ανάμεσα σε δύο καταστάσεις. Ο *Newcomb* εξάγει κλάσεις με διαφορετικούς τρόπους. Για παράδειγμα *data objects* κλάσεις μπορούν να ανακτηθούν από την *alias analysis.*

2.2.4 Προβλήματα στην εφαρμογή της ανακατασκευής

Υπάρχει ένας αρκετά μεγάλος αριθμός από εργαλεία που υποστηρίζουν την ανακατασκευή. Ωστόσο τα υπάρχοντα εργαλεία είναι συνήθως βολικά μόνο για συγκεκριμένες περιπτώσεις. Επίσης δεν είναι πάντα ευέλικτα ούτε επεκτάσιμα. Ενδέχεται να μην υποστηρίζουν τις ιδιαιτερότητες του συγκεκριμένου συντακτικού της γλώσσας προγραμματισμού. Εργαλεία που βασίζονται στο διάσχιση δεν αποδέχονται προβληματικά προγράμματα, για παράδειγμα, προγράμματα για τα οποία λείπουν κάποια *include files* (απαραίτητα για το συνολικό πρόγραμμα).

Είναι πολύ συνηθισμένο στην ανακατασκευή ενός *BIS* ως μόνη διαθέσιμη πληροφορία να είναι μόνο ο πηγαίος κώδικας. Η έλλειψη πληροφορίας κάνει την ανακατασκευή ακόμη πιο δύσκολη διαδικασία. Έτσι πολλά εργαλεία που εφαρμόζονται στην ανακατασκευή δεν απαιτούν για να λειτουργήσουν άλλη πληροφορία εκτός τον πηγαίο κώδικα του παλαιού *BIS.*

Στην ανακατασκευή συστημάτων, δεν υπάρχουν πλήρως αυτοματοποιημένα εργαλεία που να μπορούν να παράγουν ως αποτέλεσμα καλό τελικό κώδικα. Τα εργαλεία είναι ημιαυτόματα, δηλαδή συνηθισμένες ενέργειες ή επαναλαμβανόμενες διαδικασίες γίνονται αυτόματα, αλλά η επέμβαση του ανθρώπου είναι αναγκαίο σε μερικές αποφάσεις, για παράδειγμα για την απόφαση εάν κάποιο υποψήφιο αντικείμενο θα είναι και στο τελικό σύστημα επίσης αντικείμενο. Από τη μία μεριά είναι θεμιτή η δημιουργία πλήρως αυτοματοποιημένων εργαλείων για τις περισσότερες φάσεις της ανακατασκευής επειδή μπορούν και μειώνουν αρκετά τους απαιτούμενου πόρους που θα χρειαστούμε. Από την άλλη μεριά αυτά τα εργαλεία παράγουν αντικείμενα τα οποία βασίζονται στο τεχνικό μέρος της υλοποίησης του κώδικα και δεν σημαίνει ότι είναι και αντικείμενα στο επίπεδο της εφαρμογής (σε πιο αφαιρετικό επίπεδο) [Deursen99].

Η δική μας μεθοδολογία έχει σαν στόχο την παραγωγή ενός συστήματος κατά το δυνατόν *platform independent* και να απαιτεί το δυνατόν λιγότερο εξωτερική βοήθεια, είτε αυτή προέρχεται από κάποιον άνθρωπο είτε από κάποιο άλλο εργαλείο.

2.3 Χρήση της UML στην ανακατασκευή

Η UML είναι ένα πρότυπο δήλωσης, όπως είναι τα *IDL-based* αντικειμενοστρεφή πρότυπα, οι *Java* διεπαφές και οι *Microsoft IDL*-διεπαφές. Εντούτοις, η *UML* διαφέρει από αυτά τα άλλα είδη πρότυπων δήλωσης σε μερικά σημαντικά σημεία. Μία διαφορά είναι ότι τα *UML-models* μπορούν να εκφραστούν με γραφικές εκφράσεις (*visual representations*). Η σημαντικότερη διαφορά είναι ότι τα *UML-models* είναι σημασιολογικά πολύ πλουσιότερα από τις αντίστοιχες αναπαραστάσεις των άλλων προτύπων. Τα άλλα *declarative* πρότυπα παρέχουν μία σαφή σύνταξη, αλλά πολύ φτωχή σημασιολογία. Μια ακόμη σημαντική διαφορά είναι ότι η *UML* έχει καθοριστεί σαφώς και με τη βοήθεια βασικών αρχών.

Παρουσιάζουμε μία λίστα σημασιολογικών πληροφοριών που μπορούν να εκφραστούν σε ένα *UML-model*:

- Υπάρχουν αμετάβλητοι κανόνες.
- Μπορούν να δηλωθούν συνθήκες πριν και μετά για διαδικασίες.
- Μία λειτουργία μπορεί να έχει *side-effects*.
- Υποκατηγορίες ενός κοινού *super-type* είναι απόλυτα διακριτές μεταξύ τους ή επικαλύπτονται.
- Ένας τύπος μπορεί να είναι αφηρημένος, δηλαδή εάν είναι επιτρεπόμενο να δημιουργηθούν *instances* του τύπου.

Οι αμετάβλητοι κανόνες και pre/post όροι είναι ιδιαίτερα σημαντικά χαρακτηριστικά γνωρίσματα για μία προσέγγιση στην αυστηρή τεχνολογία λογισμικού που ονομάζεται *contract based design*. Η *UML* δεν πρωτοτυπεί στην βασική αρχή του *contract-based-design*, αλλά παρέχει μία πολύ καλή υποστήριξη σε αυτή. Η *UML* παρέχει και μία επίσημη *assertion* γλώσσα που καλείται *Object Constraint Language* (*OCL*) η οποία διευκολύνει την τυποποίηση σημασιολογικών προδιαγραφών, τη στιγμή που δεν μπορεί να εξαλειφθεί η ανάγκη για τη χρήση άτυπων γραπτών

επεξηγήσεων *-informal textual explanations-* της σημασιολογίας, κάτι που μειώνει σημαντικά την ανάγκη για χρήση τους.

Η χρήση *UML* προσφέρει τη δυνατότητα να βασιστούμε στα έτοιμα *Object Management Group (OMG) standards* των οποίων τα *semantics* διευκρινίζονται σήμερα με απλό γραπτό κείμενο. Στην πραγματικότητα μερικές τρέχουσες *OMG-specifications* συμπεριλαμβανομένων των specifications *UML*, *MOF*, *CWM* και *EDOC* (την οποία και εμείς χρησιμοποιούμε) κάνουν χρήση της *UML* και *OCL* για να γίνει ένας τυπικός ορισμός των *semantics*. Ο αυστηρός καθορισμός των *semantics* μειώνει την ασάφεια στις προδιαγραφές των συστημάτων και αυτό κάνει τη ζωή ευκολότερη στους σχεδιαστές και προγραμματιστές όσον αφορά τα ακόλουθα:

- Παρέχει στον προγραμματιστή ακριβέστερες οδηγίες για το τι πρέπει να κάνει και κατ' επέκταση ελαττώνει το βαθμό στον οποίο ο προγραμματιστής πρέπει να υποθέσει ή να μαντέψει ποιες είναι οι προθέσεις του σχεδιαστή.
- Μειώνει το σύνολο της εργασίας που απαιτείται για να κατασκευάσουμε διαφορετικές υλοποιήσεις των συστημάτων, οι οποίες στηρίζονται στις ίδιες προδιαγραφές.

Ένα μοντέλο το οποίο τυποποιεί τα *semantics* σε αυτό το βαθμό, είναι πολύ διαφορετικό σε σχέση με ένα μοντέλο που δεν τα τυποποιεί.

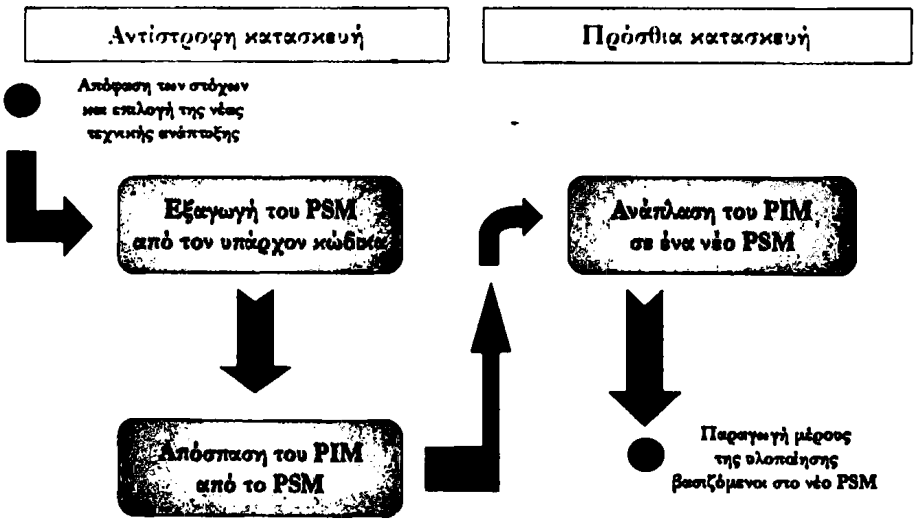
Κεφάλαιο 3: Προτεινόμενη μεθοδολογία ανακατασκευής συστημάτων

Σε αυτό το κεφάλαιο περιγράφεται η μεθοδολογία που ακολουθήθηκε. Αρχικά δίνεται μία γενική περιγραφή της μεθοδολογίας μας. Η μεθοδολογία μας απαιτεί τον ορισμό αναπαραστάσεων για την περιγραφή *Platform Independent Representation* (*PIR*) και το οποίο περιγράφεται αμέσως μετά. Επίσης, στο πλαίσιο του δικού μας *case study*, χρειάστηκε να ορίσουμε δύο *Platform Specific Representations* (*PSR*) για την παλαιά και την νέα τεχνολογική πλατφόρμα που χρησιμοποιήσαμε, που παρουσιάζονται στη συνέχεια.

3.1 Περιγραφή μεθοδολογίας

Η μεθοδολογία που προτείνουμε για την ανακατασκευή ενός *BIS* βασίζεται σε μία γενικευμένη μεθοδολογία για ανάπτυξη συστημάτων λογισμικού, η οποία κάνει χρήση μοντέλων (*Model Driven Architecture MDA*). Η *MDA* έχει προταθεί πρόσφατα από τον *Soley* και το *Object Management Group* (*OMG*) [Soley00]. Η μεθοδολογία που προτείνουμε αποτελείται από δύο βασικές ροές εργασιών για αντίστροφη κατασκευή και πρόσθια κατασκευή αντίστοιχα. Οι ροές εργασιών απεικονίζονται στο Σχήμα 2.

Οι εργασίες που σχετίζονται με την αντίστροφη κατασκευή αρχίζουν με την αναγνώριση των στόχων και την ανάλυση απαιτήσεων από τον σχεδιαστή που έχει αναλάβει την κατασκευή του καινούργιου *BIS*. Σε αυτό το σημείο γίνεται και η επιλογή, από τον σχεδιαστή, της τεχνολογικής πλατφόρμας που θα χρησιμοποιηθεί στο νέο σύστημα. Η πρώτη πρακτική εργασία της αντίστροφης κατασκευής αφορά την εξαγωγή του *PSM* από τον πηγαίο κώδικα του παλαιού συστήματος. Το *PSM* αναπαριστά όλα τα βασικά στοιχεία που συνθέτουν το παλαιό *BIS* και τον τρόπο με τον οποίο τα στοιχεία αυτά σχετίζονται μεταξύ τους. Στη παρούσα φάση μπορεί κάποιος να διακρίνει τον τρόπο με τον οποίο εκμεταλλεύτηκαν οι προγραμματιστές του παλαιού *BIS* τις δυνατότητες αλλά και ιδιαιτερότητες της τεχνολογικής πλατφόρμας που χρησιμοποιήθηκε. Επίσης χρησιμεύει ώστε να πάρουμε μία γενική εικόνα για τα βασικά στοιχεία τα οποία θα χρειαστούν αλλαγή ή εξολοκλήρου ανακατασκευή. Στη συνέχεια γίνεται ένας διαχωρισμός της πληροφορίας που περιέχεται στο *PSM*. Υπάρχει πληροφορία που σχετίζεται άμεσα με την τεχνολογική πλατφόρμα του παλαιού *BIS* η οποία δεν χρησιμοποιείται καθόλου, όσον αφορά την μοντελοποίηση του *BIS*, και πληροφορία που είναι ικανή στο να μας οδηγήσει με κατάλληλους μετασχηματισμούς στην κατασκευή του *PIM*. Με την δημιουργία του *PIM* ολοκληρώνονται και οι εργασίες που σχετίζονται με την αντίστροφη κατασκευή.



Σχήμα 2: *Διαδικασία ανακατασκευής ενός BIS.*

Για την πρόσθια κατασκευή θα αρχίσουμε με το *PIM* που προέκυψε από την αντίστροφη κατασκευή και αποτελεί τον κορμό για την κατασκευή του καινούργιου *BIS* στην νέα τεχνολογική πλατφόρμα που έχει επιλεγεί. Οι εργασίες που γίνονται στην παρούσα φάση αφορούν την

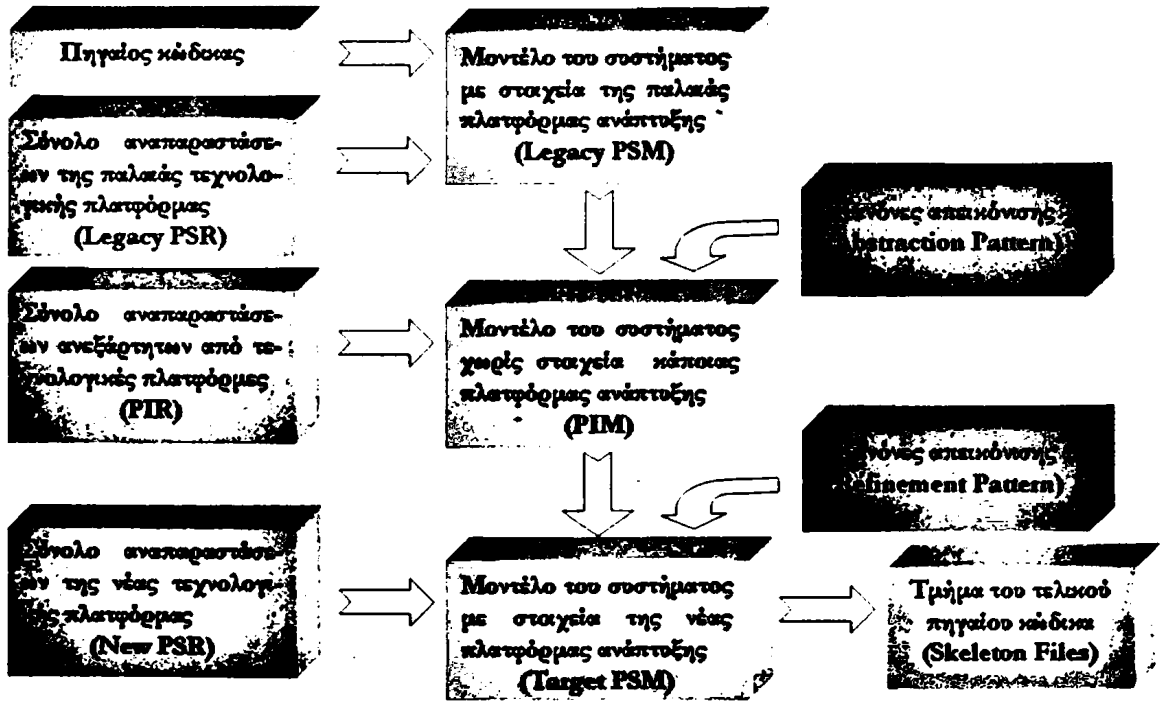
κατασκευή του νέου *PSM*. Στην συνέχεια η ροή εργασιών της πρόσθιας κατασκευής ολοκληρώνεται με την παραγωγή κομματιών της τελικής υλοποίησης του έργου (π.χ. *skeleton files*) και την παραγωγή κατάλληλα δομημένων πληροφοριών που καθοδηγούν τον προγραμματιστή κατά την υλοποίηση του τελικού συστήματος.

Για να περιγράψουμε τα κύρια μοντέλα που απαιτεί η μέθοδός μας και συγκεκριμένα το *PSM* του παλαιού *BIS*, το *PIM* του συστήματος και έπειτα το *PSM* του καινούργιου συστήματος, χρησιμοποιούμε την *UML*. Η *UML* όπως είδαμε και στην παράγραφο 2.3 είναι ένα καθιερωμένο πρότυπο μοντελοποίησης, το οποίο παρέχει βασικά δομικά στοιχεία για τον σχεδιασμό μοντέλων τα οποία αφορούν την περιγραφή της δομής και της συμπεριφοράς του συστήματος που θέλουμε να περιγράψουμε. Παρόλα αυτά τα βασικά δομικά στοιχεία που παρέχονται από την *UML* περιγράφουν πολύ γενικές έννοιες. Αυτό είναι αναμενόμενο, διότι η *UML* έχει σαν στόχο να γίνει η βάση για την ανάπτυξη μίας συλλογής από πιο συγκεκριμένες αναπαραστάσεις - *notations* - οι οποίες ονομάζονται *UML Profiles*. Τα *profiles* αυτά έχουν σαν σκοπό την περιγραφή διαφορετικών τύπων συστημάτων. Παράδειγμα ενός *profile* είμαι αυτό για *Enterprise Distributed Computing (EDOC) systems* στο οποίο και εμείς βασιστήκαμε. Ένα *UML Profile* αποτελείται από τον ορισμό ενός πλήθους από *stereotypes*. Το κάθε *stereotype* αποτελείται από ένα πλήθος περιορισμών *-constraints-* και ιδιοτήτων *-properties-* τα οποία επεκτείνουν τη σημασιολογία των βασικών *UML* δομικών στοιχείων. Στην δική μας μεθοδολογία γίνεται χρήση αναπαραστάσεων που σχετίζονται ή όχι με κάποια τεχνολογική πλατφόρμα (*PIR* και *PSR*). Στην παράγραφο 3.2 ορίζουμε το *PIR*, ενώ στις παραγράφους 3.3 και 3.4 ορίζουμε δύο *PSRs*, τα οποία αντιστοιχίζονται σε διαφορετικού τύπου τεχνολογικές πλατφόρμες.

Ανάλυση εργασιών της αντίστροφης κατασκευής

Η πρώτη εργασία της μεθοδολογίας εστιάζει στην κατασκευή του *PSM* με βάση τον υπάρχοντα κώδικα του παλαιού *BIS*. Αρχικά στον πηγαίο κώδικα του παλαιού συστήματος γίνεται μία προεπεξεργασία με σκοπό να συλλέξουμε πληροφορία η οποία θα περιγράφει στο *PSM*. Η προεπεξεργασία αυτή του πηγαίου κώδικα, που προέρχεται από τα περισσότερα σύγχρονα συστήματα *COTS*, είναι σχετικά εύκολη. Οι λόγοι για τους οποίους ισχύει αυτό είναι δύο:

- Οι περισσότερες τεχνολογικές πλατφόρμες των τελευταίων χρόνων ακολουθούν το αντικειμενοστρεφές μοντέλο, κάτι το οποίο μας διευκολύνει στην αντιστοίχηση αντικειμένων με στοιχεία της εκάστοτε αναπαράστασης στο *PSR*.
- Η κάθε τεχνολογική πλατφόρμα μπορεί να ακολουθεί μία διαφορετική προσέγγιση στον τρόπο με τον οποίο είναι δομημένος ο πηγαίος κώδικας (στον τρόπο και σύνταξη με τον οποίο τον αποθηκεύει σε αρχεία) αλλά σε όλες υπάρχει σαφής διαχωρισμός ανάμεσα στον *Platform Independent* κώδικα (προέρχεται από προγραμματιστή) και στον *Platform Dependent* κώδικα (έχει παραχθεί από την τεχνολογική πλατφόρμα αυτόματα).



Σχήμα 3: ***Διάγραμμα ροής για την μεθοδολογία μας.***

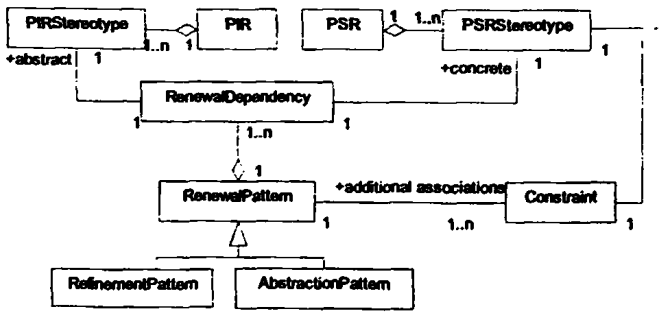
Η δεύτερη εργασία κατά την αντίστροφη κατασκευή έχει σκοπό να συνθέσει το *PSM* του παλαιού *BIS* με βάση το αποτέλεσμα της προεπεξεργασίας. Η σύνθεση του *PSM* γίνεται με βάση το *PSR* που αντιστοιχεί στην τεχνολογική πλατφόρμα που χρησιμοποιήθηκε στο παλαιό *BIS*. Για την σύνθεση του *PSM* αρχικά διασχίζουμε τα δεδομένα που προέρχονται από την προεπεξεργασία και ανιχνεύουμε στοιχεία (π.χ. αντικείμενα). Τα στοιχεία αυτά στη συνέχεια με χρήση κατάλληλης πληροφορίας, η οποία σχετίζεται άμεσα, όπως είναι φυσικό, με τη

τεχνολογική πλατφόρμα που χρησιμοποιήθηκε, αντιστοιχίζονται σε *stereotypes* του *PSR* με διαφορετικούς περιορισμούς. Για κάθε ένα από αυτά τα στοιχεία δημιουργούμε ένα *UML* στοιχείο που τοποθετείται στο *PSM* και χαρακτηρίζεται από το *stereotype*.

Για την ολοκλήρωση της ροής εργασιών της αντίστροφης κατασκευής επεξεργαζόμαστε τα στοιχεία που περιγράφουν το *PSM* με σκοπό την κατασκευή του *PIM*. Η ανάπλαση αυτή του μοντέλου εξαρτάται άμεσα από κάποια *abstraction patterns*. Ο σκοπός αυτών των *abstraction patterns* είναι να καθορίσουμε αυστηρά τον τρόπο αντιστοίχησης ενός *stereotype* X που ανήκει στο *PSR* σε ένα *stereotype* του *PIR*. Η αντιστοίχηση αυτή μπορεί να είναι είτε η απλή περίπτωση της «ένα προς ένα» αντιστοίχησης ενός στοιχείου του *PSR* σε ένα του *PIR* είτε στην αντιστοίχηση μίας αποσύνθεσης. Αποσύνθεση έχουμε όταν ένα στοιχείο X του *PSR* θα αντικατασταθεί από περισσότερα στοιχεία Y_1, Y_2, ... του *PIR*. Μεταξύ των στοιχείων Y_1, Y_2, ... επιτρέπουμε την ύπαρξη συσχετίσεων. Στα *abstraction patterns* είναι αυστηρά καθορισμένες οι νέες συσχετίσεις (όταν υπάρχουν). Τα *attributes* που αφορούν το στοιχείο X ενδέχεται να μοιράζονται στα αντίστοιχα Y_1, Y_2, ... Τέτοιες αντιστοιχήσεις περιγράφονται στην παράγραφο 4.1.2.

Ανάλυση εργασιών της πρόσθιας κατασκευής

Στην πρόσθια κατασκευή πραγματοποιούνται δύο κύριες εργασίες. Αρχικά γίνεται μετατροπή του *PIM* στο *PSM* του νέου *BIS*. Σκοπός είναι να κατασκευαστεί το *PSM* που περιγράφει το *BIS* αλλά με δομικά στοιχεία που σχετίζονται με την νέα τεχνολογική πλατφόρμα που έχουμε επιλέξει. Η ροή εργασιών αρχίζει με την εφαρμογή *refinement patterns* στο υπάρχον μοντέλο που περιγράφει το *PIM*. Με βάση τις εξαρτήσεις που ορίζονται στα *refinement dependencies* πραγματοποιείται η μετατροπή του κάθε *PSR stereotype* X στα αντίστοιχα *platform-specific* στοιχεία.



Σχήμα 4: ***Η σχέση των abstraction και raffinement patterns.***

Παρόμοια εργασία, δηλαδή της ανάπλασης ενός μοντέλου, που εφαρμόστηκε στην περίπτωση της αντίστροφης κατασκευής γίνεται με την χρήση *abstraction patterns* μετατροπή ενός *PSM* σε ένα *PIM* (Σχήμα 4).

Λεπτομέρειες για τα στοιχεία που αποτελούν τα *PIR* και *PSRs*, αλλά και τις σχέσεις που τα ενώνουν, δίνονται στις παραγράφους 3.2, 3.3 και 3.4. Στην παράγραφο 4.1 παρουσιάζουμε *abstraction* και *refinement dependencies* που έχουν χρησιμοποιηθεί για τη δική μας πρακτική εφαρμογή. Τα *abstraction dependencies* αφορούν τη φάση της αντίστροφης κατασκευής και ειδικότερα την μετατροπή του *Delphi PSR* στα αντίστοιχα στοιχεία του *PIR*. Αντίστοιχα για τη φάση της πρόσθιας κατασκευής ορίζονται τα *refinement dependencies* όπου μετατρέπουν το *PIR* σε *PSR* πλατφόρμας που σχετίζεται με την χρήση *PhP*, *HTML* και *JavaScript*.

Το τελικό προϊόν για την πρόσθια κατασκευή εκτός από το *PSM* είναι και η κατασκευή κορμού για την υλοποίηση του καινούργιου *BIS* (*skeleton files*). Η ιδέα είναι, να παρέχουμε όσο το δυνατόν περισσότερη πληροφορία στους προγραμματιστές. Θεωρητικά, με βάση το *PSM* θα μπορούσε κάποιος να φτιάξει ακόμη και το τελικό προϊόν χωρίς την ανάγκη επέμβασης από κάποιον προγραμματιστή. Ο παράγοντας που επηρεάζει το παραπάνω είναι η τεχνολογική πλατφόρμα που είχαμε στην αρχή και σε ποια θέλουμε να καταλήξουμε. Η πιο συχνή περίπτωση είναι να έχουν οριστεί κάποια *stub pattern* για κάθε στοιχείο του *PSR* με βάση τα οποία δημιουργούμε στιγμιότυπα για κάθε συγκεκριμένο στοιχείο του νέου *PSM*.

3.2 Αναλυτική περιγραφή των Platform Independent Representation

Εδώ αναφέρονται όλα τα στοιχεία τα οποία αποτελούν το *PIR*. Ο σχεδιασμός που προτείνουμε αποτελείται από 30 περίπου στοιχεία, τα οποία αντιπροσωπεύουν όσο το δυνατόν καλύτερα τις σύγχρονες τεχνολογικές πλατφόρμες. Για την πραγματοποίησή του εξετάσαμε τις πιο γνωστές πλατφόρμες που χρησιμοποιούνται σήμερα και είναι κατασκευασμένες από τις *Microsoft®* και *Borland®*. Όλες οι πλατφόρμες αυτές παρουσιάζουν μεγάλες ομοιότητες στην ύπαρξη βασικών αντικειμένων, τα οποία σχετίζονται με συνήθεις λειτουργίες σε ένα σύστημα διεπαφής χρήστη καθώς και με την χρήση κάποιας βάσης δεδομένων.

Συχνά χρησιμοποιούμε τον ορισμό *stereotypes* όταν προσαρμόζουμε την *UML* σε κάποια συγκεκριμένη μέθοδο ή στην υλοποίηση ενός συστήματος. Ορισμοί και αρχές που χρησιμοποιούνται σε μία μέθοδο καθώς και κοινά στοιχεία ή σχέσεις που παρουσιάζονται στη φάση της υλοποίησης ενός συστήματος, ενδέχεται να μην υπάρχουν στην *standard UML*. Τα στοιχεία αυτά τα προσθέτουμε στη *UML* με τον ορισμό νέων *stereotypes*. Για τον ορισμό ενός *stereotype* είναι αναγκαίο να γίνουν γνωστά τα ακόλουθα:

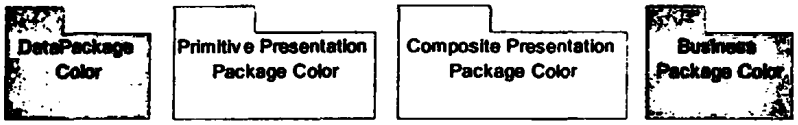
- Σε ποιο *standard UML* στοιχείο θα βασιστεί το νέο *stereotype* (η βασική κλάση).
- Τη νέα σημασιολογία που προσθέτει το νέο *stereotype* και η όποια είναι επέκταση της σημασιολογίας του *standard* στοιχείου.
- Σχέσεις που συσχετίζει το νέο *stereotype* με άλλα που πιθανόν να έχουν ορισθεί για τον τομέα εφαρμογής του συστήματος ή για τη μέθοδο.
- Ένα ή περισσότερα παραδείγματα για το πώς γίνεται η χρήση του νέου *stereotype*.

Τα *stereotypes* τα οποία αποτελούν το *PIR* έχουν χωριστεί σε πακέτα. Ο λογικός διαχωρισμός γίνεται με τρόπο ώστε να έχουμε εποπτεία με τι περίπου σχετίζεται το κάθε *stereotype*. Εδώ θα διαχωρίσουμε τα στοιχεία σε:

- *Business*: Στοιχεία τα οποία σχετίζονται άμεσα με διεργασίες και λειτουργίες στο παρασκήνιο (*background*).
- *Data*: Στοιχεία που σχετίζονται με την διεπαφή με τη βάση δεδομένων.
- *Presentation*: Στοιχεία που καλύπτουν τον σχεδιασμό της διεπαφής με τον χρήστη. Τα στοιχεία που ανήκουν στο *presentation* διαχωρίζονται σε *composite* και *primitive*. Τα *primitive* είναι στοιχεία, τα οποία εμφανίζονται άμεσα στην διεπαφή μας με τον χρήστη. Αντίθετα τα *composite* είναι στοιχεία, τα οποία χρησιμεύουν για να οργανώσουν στοιχεία που ανήκουν στο *primitive presentation* πακέτο (π.χ. καμβάδες και φόρμες).

Ακολουθεί στη συνέχεια η περιγραφή όλων των *stereotypes* που συνθέτουν το *PIR* κατηγοροποιημένα ανά πακέτο. Κατόπιν γίνεται περιγραφή των *metamodels*, τα οποία περιγράφουν τις συσχετίσεις που υπάρχουν ανάμεσα στα *stereotypes*. Κατά την παρουσίαση των *metamodels* χρησιμοποιούμε τέσσερα διαφορετικά χρώματα, που εμφανίζονται στο Σχήμα 5, για

να προσδιορίζουμε με αυτά τα χρώματα σε ποιο πακέτο ανήκουν τα στοιχεία δηλαδή: *data* (βυσσινί), *primitive presentation* (γαλάζιο), *composite presentation* (εκρού) ή *business* (πράσινο) πακέτο.



Σχήμα 5: ***Διαχωρισμός των πακέτων με χρήση χρωμάτων.***

Στην παρούσα παράγραφο αλλά και στις δύο όπου ακολουθούν γίνεται εκτεταμένη αναφορά σε νέα *stereotypes*. Σε προηγούμενη ενότητα αναφέραμε το *specification* του *EDOC* (αναλυτικοί ορισμοί για τις κλάσεις που χρησιμοποιούνται είναι διαθέσιμοι στο παράρτημα) σαν κύριο γνώμονα για τους νέους ορισμούς. Τα νέα στοιχεία που παρουσιάζουμε παρακάτω χρησιμοποιούν βασικές κλάσεις που έχουν οριστεί στο *EDOC* (έγινε σχετική αναφορά αυτού στο 2.3).

Το συχνότερα χρησιμοποιημένο στοιχείο είναι το *ECA:CCA::ProcessComponent*. Είναι το στοιχείο γενικής χρήσεως και θεωρεί τους λιγότερους περιορισμούς. Υπάρχουν στοιχεία τα οποία έχουν ως βασική κλάση την *ECA:CCA::Entity::EntityData* που αντιπροσωπεύει γενικά ένα *data structure*. Αυτό είναι πολλές φορές ισοδύναμο με ένα *entity* στο *data modeling* ή μία σχέση σε μία σχεσιακή βάση δεδομένων. Στοιχεία που συναντάμε στο *data* πακέτο και έχουν έμμεση σχέση με επεξεργασία στοιχείων ή με γενικότερες λειτουργίες της *database* έχουν σαν βασική κλάση την *ECA:Entity::DataManager*.

Stereotypes* που ανήκουν στο πακέτο *Business

- *PIEvent*: Τα *PIEvent* χαρακτηρίζονται διαφορετικά το κάθε ένα με βάση την αιτία που τα προκαλεί. Για παράδειγμα σε ένα στοιχείο θα υπάρχει διαχωρισμός στα *events OnActivate* και *OnResize*. Για κάθε *stereotype* έχουν καθοριστεί ακριβώς ποια *events* μπορεί να υπάρξουν τα οποία τα διαχωρίζουμε σε 2 κύριες κατηγορίες.
    - Τα *PIEvent* που αφορούν τις συνήθεις ενέργειες ενός χρήστη που εμφανίζονται κατά την χρήση ενός συστήματος μέσω της διεπαφής χρήστη *(GUI)*: *OnActivate, OnCanResize, OnClick, OnClose, OnCloseQuery, OnConstrainedResize, OnContext-Popup, OnCreate, OnDblClick, OnDeactivate, OnDestroy, OnDockDrop, OnDockOver,*

(xsl:apply-templates), (xsl:apply-templates/), (xsl:attribute), (xsl:attribute/), (xsl:call-template), (xsl:call-template/), (xsl:choose), (xsl:choose/), (xsl:comment), (xsl:comment/), (xsl:copy), (xsl:copy/), (xsl:copy-of), (xsl:copy-of/), (xsl:decimal-format), (xsl:decimal-format/), (xsl:element), (xsl:element/), (xsl:fallback), (xsl:fallback/), (xsl:for-each), (xsl:for-each/), (xsl:if), (xsl:if/), (xsl:message), (xsl:message/), (xsl:number), (xsl:number/), (xsl:otherwise), (xsl:otherwise/), (xsl:output), (xsl:output/), (xsl:param), (xsl:param/), (xsl:preserve-space), (xsl:preserve-space/), (xsl:processing-instruction), (xsl:processing-instruction/), (xsl:sort), (xsl:sort/), (xsl:strip-space), (xsl:strip-space/), (xsl:stylesheet), (xsl:stylesheet/), (xsl:template), (xsl:template/), (xsl:text), (xsl:text/), (xsl:transform), (xsl:transform/), (xsl:value-of), (xsl:value-of/), (xsl:variable), (xsl:variable/), (xsl:when), (xsl:when/), (xsl:with-param), (xsl:with-param/).

- Τα *PIElements* που σχετίζονται με ενέργειες στη παραπομπή για παράδειγμα τα *xslApplyImports*, *xslApplyImports/*, *xslAttribute*, *xslAttribute/*, *xslCallTemplate*, *xslCallTemplate/*, *xslChoose*, *xslChoose/*, *xslComment*, *xslComment/*, *xslCopy*, *xslCopy/*, *xslCopyOf*, *xslCopyOf/*, *xslDecimalFormat*, *xslDecimalFormat/*, *xslElement*, *xslElement/*, *xslFallback*, *xslFallback/*, *xslForEach*, *xslForEach/*, *xslIf*, *xslIf/*, *xslMessage*, *xslMessage/*, *xslNumber*, *xslNumber/*, *xslOtherwise*, *xslOtherwise/*, *xslOutput*, *xslOutput/*, *xslParam*, *xslParam/*, *xslPreserveSpace*, *xslPreserveSpace/*, *xslProcessingInstruction*, *xslProcessingInstruction/*, *xslSort*, *xslSort/*, *xslStripSpace*, *xslStripSpace/*, *xslStylesheet*, *xslStylesheet/*, *xslTemplate*, *xslTemplate/*, *xslText*, *xslText/*, *xslTransform*, *xslTransform/*, *xslValueOf*, *xslValueOf/*, *xslVariable*, *xslVariable/*, *xslWhen*, *xslWhen/*, *xslWithParam*, *xslWithParam/*.

Τα περισσότερα από τα *stereotype* που παρουσιάζουμε παρακάτω έχουν άμεση σχέση με κάποια από τα παραπάνω *events*. Αυτά υπάρχουν αναλυτικά καθορισμένα για κάθε *stereotype* στο τέλος της παραγράφου στον πίνακα 1. Για τον ορισμό του *stereotype PIElement* σαν βασική κλάση έχει επιλεγεί η *IICAHCAttributeComponent*.

- *PIActivity*: Τα *PIActivity* αντιπροσωπεύουν την πηγαίο κώδικα που έχει προκύψει από έναν προγραμματιστή ή είναι το σημείο στο οποίο θα πρέπει να επέμβει ο προγραμματιστής. Δηλαδή στα στοιχεία *PIActivity* αντιπροσωπεύει *Platform Independent Code* (*PIC*) ενώ όλα τα υπόλοιπα στοιχεία εκφράζουν *Platform Dependent Code* (*PDC*). Για τον ορισμό του *PIActivity* σαν βασική κλάση έχει επιλεγεί η *IICAHCAttributeComponent*.
- *PIConnector*: Ο *PIConnector* χρησιμοποιείται στην μεσολάβηση για να συνδέσει στοιχεία. Βασικό χαρακτηριστικό του είναι ότι μέσω αυτού γίνεται μεταβίβαση πληροφορίας από ή προς ένα αντικείμενο. Για τον ορισμό του *PIConnector* σαν βασική κλάση έχει επιλεγεί η *IICAHCAttributeComponent*.
- *PINetworkConnection*: Τα *PINetworkConnection* είναι στοιχεία τα οποία έχουν ως πατέρα το *PIConnector* και βασική κλάση την *IICAHCAttributeComponent*. Σχετίζονται με την μεταβίβαση πληροφορίας μέσω δικτυακών εφαρμογών. Για παράδειγμα εάν το υποσύστημά μας έχει δικτυακές δυνατότητες και κάποιο στοιχείο του συστήματος ανταλλάσσει στοιχεία με κάποια άλλη μηχανή τότε ενδιάμεσα πρέπει να υπάρχει ένα στοιχείο που χαρακτηρίζεται από το *PINetworkConnection stereotype*.

- *PIDiskConnection*: Τα *PIDiskConnection* είναι στοιχεία τα οποία έχουν ως πατέρα το *PIConnector* και βασική κλάση την *ECA:CCA::ProcessComponent*. Σχετίζονται με την μεταβίβαση πληροφορίας από το τοπικό σύστημα αρχείων στο οποίο λειτουργεί το *BIS*. Για παράδειγμα εάν στο σύστημά μας υπάρχει ένα στοιχείο το οποίο χρειάζεται να ανατρέξει στα δεδομένα ενός τοπικού αρχείου τότε περιμένουμε αυτό να σχετίζεται με ένα στοιχείο που χαρακτηρίζεται από το *PIDiskConnection stereotype*.
- *PIDBConnection*: Τα *PIDBConnection* είναι στοιχεία τα οποία έχουν ως πατέρα το *PIConnector* και βασική κλάση την *ECA:CCA::ProcessComponent*. Σχετίζονται με την επικοινωνία και μεταβίβαση πληροφορίας από μία βάση δεδομένων. Για παράδειγμα εάν το σύστημά μας χρησιμοποιεί κάποιο από τα στοιχεία που παρουσιάζονται παρακάτω στο *data* πακέτο τότε αυτά περιμένουμε να σχετίζονται είτε άμεσα είτε έμμεσα με ένα στοιχείο τύπου *PIDBConnection*. Αυτό προφανώς συμβαίνει διότι το στοιχείο *PIDB-Connection* αναλαμβάνει τον ρόλο να επικοινωνήσουν στοιχεία της εφαρμογής με την βάση δεδομένων, δηλαδή την διεπαφή ανάμεσα σε δύο «ξένες» πλατφόρμες.
- *PITechnologyConnection*: Τα *PITechnologyConnection* είναι στοιχεία τα οποία έχουν ως πατέρα το *PIConnector* και βασική κλάση την *ECA:CCA::ProcessComponent*. Πολλές τεχνολογικές πλατφόρμες έχουν έτοιμα υποσυστήματα τα οποία εξυπηρετούν κάποιο συγκεκριμένο σκοπό. Τις περισσότερες φορές είναι κλειστά κουτιά για τα οποία γνωρίζουμε τις δυνατότητες τους και τις λειτουργίες τους. Επίσης γνωρίζουμε τον τρόπο με τον οποίο εμείς θα τα χρησιμοποιήσουμε (τρόπος που περνάμε τα δεδομένα και τρόπος λήψης δεδομένων) αλλά δεν γνωρίζουμε πως το πετυχαίνουν αυτό. Παράδειγμα τέτοιου έτοιμου υποσυστήματος θα μπορούσε να χαρακτηριστεί ένα αντικείμενο σε ένα *COTS* το οποίο αναλαμβάνει εξ ολοκλήρου την απεικόνιση σχημάτων και σχεδίων στην οθόνη. Εμείς στο *PIR* δεν κάνουμε διαχωρισμό τέτοιων υποσυστημάτων και τα απεικονίζουμε όλα με *PITechnologyConnection*. Στις περιπτώσεις αυτές καλείται ο σχεδιαστής να διαπιστώσει τον ρόλο του κάθε υποσυστήματος με βάση τα συμπληρωματικά στοιχεία που έχουμε συλλέξει (στην περίπτωση που το *PITechnologyConnection* ανήκει σε ένα *PIM* το οποίο έχει προέλθει από κάποιο *PSM* με *abstraction patterns*).
- *PIDataStructure*: Τα δεδομένα που επιστρέφονται συνήθως από το αποτέλεσμα μίας ενέργειας (με τη χρήση ενός *PIConnector*) παρουσιάζονται στο μοντέλο μας με την μορφή ενός στοιχείου *PIDataStructure*. Το στοιχείο αυτό σχετίζεται με την μορφή με την οποία

μεταβιβάζονται δεδομένα. Δηλαδή θα αναθέτει τα δεδομένα με βάση μία συγκεκριμένη δομή, για παράδειγμα σε ακολουθία (σειρά) πλειάδων με καθορισμένο ορισμό των μονάδων που έχει η πλειάδα. Για τον ορισμό του *PIDataStructure* ως βασική κλάση έχει επιλεγεί η *ECA:CCA::ProcessComponent*.

Stereotypes* που ανήκουν στο πακέτο *Data

- *PIDBActivity*: Για εργασίες που σχετίζονται με την αλληλεπίδραση με μία βάση δεδομένων έχει οριστεί το *PIDBActivity*. Η περίπτωση αυτή είναι η γενικότερη που έχουμε στη διάθεσή μας για τον σκοπό αυτό. Επειδή έχει να κάνει με τον χειρισμό δεδομένων σαν βασική κλάση έχει επιλεγεί η *ECA:Entity::DataManager*.
- *PIDBQuery*, *PIDBTable*: Για ενέργειες που αφορούν την αναζήτηση δεδομένων από μία βάση δεδομένων έχουν οριστεί οι *PIDBQuery* και *PIDBTable*. Η *PIDBTable* σχετίζεται άμεσα με την χρήση ενός ολόκληρου πίνακα από μία βάση δεδομένων ενώ η *PIDBQuery* είναι πιο ευέλικτη και ζητάει στήλες από διάφορους πίνακες μίας βάσης δεδομένων (χρήση των *PIEntityData*). Για τον ορισμό του *stereotype* ως πατέρας έχει ορισθεί το *PIDBActivity*. Επειδή έχει να κάνει με τον χειρισμό δεδομένων και σαν βασική κλάση έχει επιλεγεί επίσης η *ECA:Entity::DataManager*.
- *PIDBRawCommand*: Η *PIDBRawCommand* την χρησιμοποιούμε για παραπλήσιες ενέργειες με αυτές που παρουσιάσαμε στις *PIDBQuery* και *PIDBTable*. Οι εντολές προς την βάση δεδομένων είναι οποιασδήποτε φύσης. Για παρά-δείγμα η διαχείριση χρηστών σε μία βάση δεδομένων ή ο ορισμός νέων κλειδιών και πινάκων σε μία βάση δεδομένων. Αντίστοιχα για τον ορισμό του *stereotype* ως πατέρας έχει ορισθεί το *PIDBActivity* και σαν βασική κλάση έχει επιλεγεί επίσης η *ECA:Entity: :DataManager*.
- *PIEntityData*: Τα *PIEntityData* χρησιμεύουν στον αυστηρό καθορισμό πεδίων μίας πηγής δεδομένων με μεταβλητές. Για παράδειγμα στην βάση δεδομένων τα *PIEntityData* καθορίζουν την στήλη ενός πίνακα, σε άλλες περιπτώσεις μπορεί να αντιστοιχούν στο πεδίο μίας δομής. Η βασική κλάση *ECA:CCA::Entity::EntityData* ορίζεται ακριβώς για αυτόν τον σκοπό από το *EDOC*.

Stereotypes* που ανήκουν στο πακέτο *Composite Presentation

- *PICompositePresentation*: Σε κάποιες εφαρμογές τα στοιχεία του *PrimitivePresentation* επικάθονται άμεσα στην κύρια φόρμα της εφαρμογής. Εναλλακτικά είναι οργανωμένα σε διαφορετικού τύπου καμβάδες (*containers*) τα οποία αναπαριστάνονται με την χρήση του *PICompositePresentation*. Δηλαδή οι καμβάδες αυτοί επικάθονται στην κύρια φόρμα της εφαρμογής και τα στοιχεία του *PrimitivePresentation* επικάθονται στους παραπάνω καμβάδες. Σαν βασική κλάση έχει επιλεγεί η *ECA:CCA::ProcessComponent*.
- *PICustomCompositePresentation, PICanvas*: Η ύπαρξη των *PICustomCompositePresentation* και *PICanvas* συνιστούν στην υποστήριξη στοιχείων που ανήκουν στο *Primitive Presentation* πακέτο. Η κύρια διαφορά είναι ότι στα *PICustomCompositePresentation* ο σχεδιαστής μπορεί να προσθέσει δικά του *attributes* και στοιχεία ανάλογα με τις ανάγκες τους ενώ το *PICanvas* θα χρησιμοποιηθεί όταν ο καμβάς αντιστοιχεί σε κάτι πιο συγκεκριμένο. Ο διαχωρισμός αυτός μας ωφελεί στο να γίνεται ευκολότερη η διάκριση στα μοντέλα των απλών περιπτώσεων με των σύνθετων. Δηλαδή κατά την αντίστροφη κατασκευή όταν θα έχουμε την ανάγκη να διασπάσουμε ένα στοιχείο και να χρησιμοποιήσουμε έναν καμβά τότε θα χρησιμοποιήσουμε τον *PICustomCompositePresentation*, αντίθετα όταν στο *PSM* υπάρχει στο μοντέλο κάποιο συγκεκριμένο είδος καμβά τότε χρησιμοποιούμε το *PICanvas* (τέτοιες περιπτώσεις παρουσιάζονται στην ενότητα 4.1.2). Για τον ορισμό τους σαν βασική κλάση έχει επιλεγεί η *ECA:CCA::ProcessComponent* και πατέρας τους είναι το *PICompositePresentation*.
- *PIGroupBox, PIPanel*: Στα *PIGroupBox* και *PIPanel* η κύρια διαφορά ανάμεσα σε αυτά τα δύο στοιχεία εστιάζεται στο ότι το *PIGroupBox* μπορεί να έχει έναν μικρό ρόλο στην διεπαφή με τον χρήστη ενώ το *PIPanel* χρησιμεύει στην απλή ομαδοποίηση των *primitive* στοιχείων. Άμεση συνέπεια των παραπάνω είναι το διαφορετικό σύνολο των *events* που σχετίζονται με αυτά τα δύο *stereotypes* και φαίνονται στις περιγραφές τους στον Πίνακα 1. Για τον ορισμό τους ως πατέρας είναι το *PICanvas* και ως βασική κλάση είναι επίσης η *ECA:CCA::ProcessComponent*.
- *PIForm*: Η *PIForm* είναι κάτι αντίστοιχο με την *PICanvas* αλλά την χρησιμοποιούμε όταν έχουμε την εμφάνιση μίας νέας φόρμας σε ένα νέο παράθυρο. Είναι το βασικό στοιχείο για την μοντελοποίηση του *PIR*. Για τον ορισμό του *PIForm* βασική κλάση έχει επιλεγεί η *ECA:CCA::ProcessComponent*.

Stereotypes* που ανήκουν στο πακέτο *Primitive Presentation

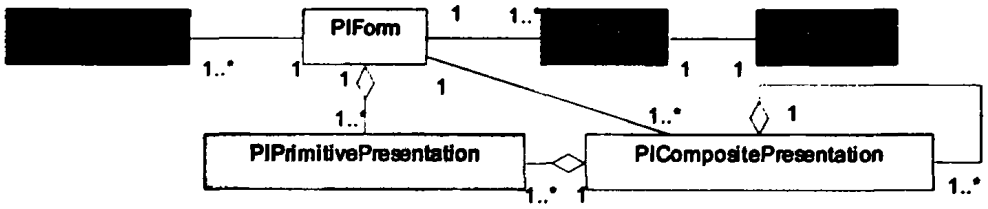
- *PIToolBar, PIToolButton*: Υπάρχουν πλαίσια υπό την μορφή «μπάρας» *PIToolBar* τα οποία περιέχουν συντομεύσεις σε κουμπάκια *PIToolButton*. Παράδειγμα μίας τέτοιας γραφικής υλοποίησης σε μία διεπαφή χρήστη είναι τα γραφικά βελάκια με κατεύθυνση αριστερά και δεξιά σε έναν *web browser* που αντιστοιχούν στις ενέργειες *back* και *forward*. Στην περίπτωση αυτή η μπάρα θα αντιστοιχιστεί σε ένα στοιχείο *PIToolBar* ενώ για το κάθε βελάκι θα έχουμε ένα στοιχείο που χαρακτηρίζεται από το *PIToolButton*. Βασική κλάση για αυτά τα *stereotypes* είναι η *ECA:CCA::ProcessComponent* και πατέρας του *PIToolBar* είναι το *PIPrimitivePresentation*.
- *PIMenu, PIMenuItem*: Σε ένα πρόγραμμα είναι φυσικό να προσφέρουμε στον χρήστη μέσω της διεπαφής με τον χρήστη έναν τρόπο για την εκκίνηση κάποιων ενεργειών με βάση κάποιες «συντομεύσεις». Ο όρος που συνήθως χρησιμοποιείται για κάτι τέτοιο είναι τα "*menu*". Τα *menu* τα εμφανίζουμε με στοιχεία *PIMenu* ενώ τις συντομεύσεις με στοιχεία *PIMenuItem*. Η κύρια διαφορά ανάμεσα στο *toolbar* και στα *PIMenu* είναι στην μορφή και στην οργάνωση που έχουν στην διεπαφή με τον χρήστη. Βασική κλάση για αυτά τα *stereotypes* είναι η *ECA:CCA::ProcessComponent*. Πατέρας του *PIMenu* είναι το *PIPrimitivePresentation*
- *PINewForm, PIMainMenu*: Για την ταξινόμηση «συντομεύσεων» σε μία διεπαφή χρήστη εκτός από τα *PIMenu* και *PIToolBar* ορίζουμε και τα: *PINewForm, PIMainMenu*. Το μεν *toolbar* είναι μόνιμο στην επιφάνεια εργασίας και έχει κάποια κουμπιά συντομεύσεων, το δε *PIMainMenu* αντιστοιχεί σε *menus* της μορφής που συναντάμε σε συνήθη προγράμματα σε μπάρα με επιλογές *File, Edit, View, Window, Help* κτλ και για κάθε επιλογή έχει ένα ιδιαίτερο *dropdown menu*. Το *PINewForm* είναι μία εντελώς καινούργια φόρμα σε ένα ξεχωριστό παράθυρο της οποία ο ρόλος είναι η επιλογή μίας συντόμευσης. Σε πολλές τεχνολογικές πλατφόρμες εμφανίζεται με τον όρο *pop-up window*. Βασική κλάση για αυτά τα *stereotypes* είναι η *ECA:CCA::ProcessComponent* και πατέρας ορίζεται το *PIMenu*.
- *PIImage*: Στοιχεία της διεπαφής με τον χρήστη που εμφανίζονται ως εικόνες τα αναπαριστούμε με το *stereotype PIImage*. Βασική κλάση για αυτό το *stereotype* είναι η *ECA:CCA::ProcessComponent* και πατέρας ορίζεται το *PIPrimitivePresentation*.

- *PIDataField*: Για την περιγραφή παραμέτρων και μεταβλητών (ανεξάρτητου τύπου) που εμφανίζονται στη διεπαφή έχουμε ορίσει το στοιχείο *PIDataField*. Η βασική κλάση είναι η *ECA:CCA::ProcessComponent* και πατέρας ορίζεται το *PIPrimitivePresentation*.
- *PIButton*: Στοιχεία της διεπαφής με τον χρήστη που εμφανίζονται με την μορφή κουμπιών τα αναπαριστούμε με το *stereotype PIButton*. Βασική κλάση για αυτό το *stereotype* είναι η *ECA:CCA::ProcessComponent* και πατέρας ορίζεται το *PIPrimitivePresentation*.
- *PILabel, PIStaticText*: Στοιχεία της διεπαφής με τον χρήστη που εμφανίζονται με την μορφή κειμένου τα αναπαριστούμε με ένα από τα *stereotypes PILabel* και *PIStaticText*. Η διαφορά του έχει να κάνει μόνο στην μορφή που εμφανίζονται στην διεπαφή. Ο διαχωρισμός τους μπορεί να φανεί χρήσιμος κατά την αντίστροφη κατασκευή όταν το *PSM* έχει τέτοιους διαχωρισμούς σε στοιχεία που σχετίζονται με την διεπαφή με τον χρήστη (όπως έγινε και στην περίπτωσή μας με το *Delphi PSM*). Βασική κλάση για αυτά τα *stereotypes* είναι η *ECA:CCA::ProcessComponent* και πατέρας ορίζεται το *PIPrimitivePresentation*.
- *PIDropDownMenu, PIRadioGroup* και *PIComboBox*: Αντικείμενα που σχετίζονται με την επιλογή ενός στοιχείου από μία λίστα αλλά αυτό γίνεται με διαφορετικό γραφικό τρόπο είναι τα: *PIComboBox, PIDropDownMenu, PIRadioGroup*. Βασική κλάση για τα *stereotypes* αυτά είναι η *ECA:CCA::ProcessComponent* και πατέρας τους ορίζεται το *PIPrimitivePresentation*.
- *PICheckBox*: Στοιχεία της διεπαφής με τον χρήστη που εμφανίζονται με την μορφή *checkbox* τα αναπαριστούμε με το *stereotype PICheckBox*. Βασική κλάση για αυτό το *stereotype* είναι η *ECA:CCA::ProcessComponent* και πατέρας ορίζεται το *PIPrimitivePresentation*.
- *PIMemo, PIEditField*: Στοιχεία της διεπαφής με τον χρήστη που εμφανίζονται με την μορφή ενός πλαισίου ειδικό για την εισαγωγή κειμένου, γνωστό ως *edit* και *memo box*, τα αναπαριστούμε με το *stereotype PIEditField* και *PIMemo* αντίστοιχα. Βασική κλάση για αυτό το *stereotype* είναι η *ECA:CCA::ProcessComponent* και πατέρας ορίζεται το *PIPrimitivePresentation*.
- *PITable*: Στοιχεία της διεπαφής με τον χρήστη που εμφανίζονται με την μορφή πίνακα (*grid*) τα αναπαριστούμε με το *stereotype PITable*. Βασική κλάση για αυτό το *stereotype* είναι η *ECA:CCA::ProcessComponent* και πατέρας ορίζεται το *PIPrimitivePresentation*.

- *PICustomPrimitivePresentation*: Για να καλύψουμε περιπτώσεις που θέλουμε να φτιάξουμε στοιχεία που ανήκουν στο γραφικό περιβάλλον και πληρούν ιδιαίτερους κανόνες έχουμε ορίσει την *PICustomPrimitivePresentation* σε αντιστοιχία με το *PICustomCompositePresentation* που ορίσθηκε παραπάνω. Βασική κλάση για αυτό το *stereotype* είναι η *ECA:CCA::ProcessComponent* και πατέρας ορίζεται το *PIPrimitivePresentation*.

Metamodels του *PIR*

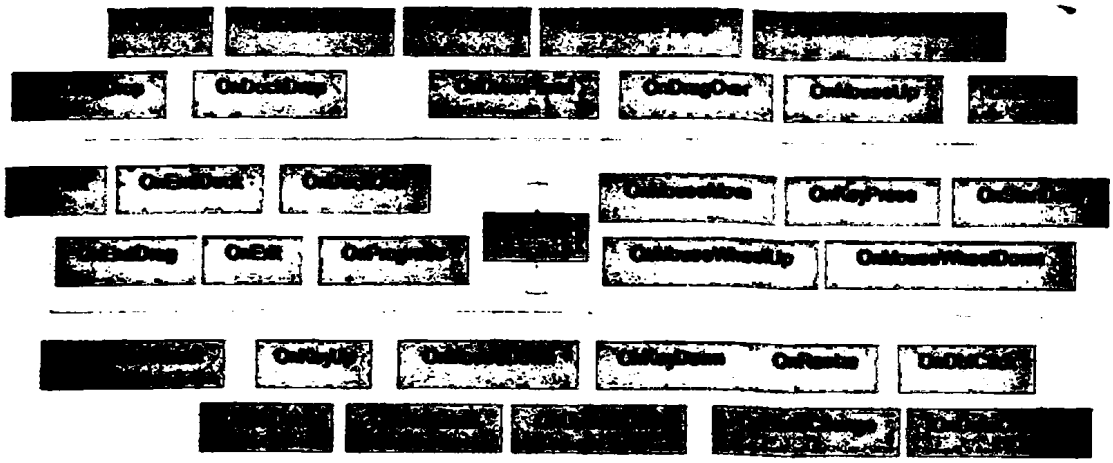
Μέχρι στιγμής έχουμε παρουσιάσει μία συλλογή από *stereotypes* χωρισμένα σε διάφορα πακέτα. Στη συνέχεια παρουσιάζουμε πώς πολλά από αυτά παρουσιάζουν αλληλεξαρτήσεις με τη βοήθεια *metamodels*. Από τεχνική άποψη στο *PIR* το πιο βασικό στοιχείο είναι το *PIForm* του οποίου ο ρόλος είναι να αποτελέσει την βάση πάνω στην οποία θα επικαθίσουν όλα τα υπόλοιπα στοιχεία και το σύνολο τους σχηματίζει το *BIS* (Σχήμα 6).



Σχήμα 6: ***Αναπαράσταση του PIR.***

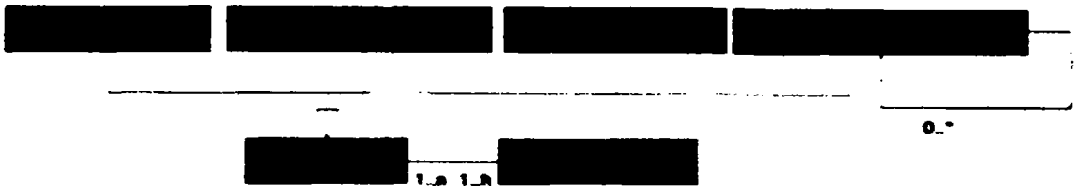
Το στοιχείο *PIForm* αποτελεί όντως το θεμέλιο στοιχείο μίας εφαρμογής και μπορούμε να πούμε ότι αντιστοιχεί στο πλαίσιο του παραθύρου με το οποίο συσχετίζονται τα παρακάτω στοιχεία:

- Ένα ή περισσότερα *PIEvents* τα οποία όταν γίνονται *triggered* προκαλούν την εκκίνηση ενός *PIActivity*. Το *PIActivity* όπως έχει προαναφερθεί αντιπροσωπεύει τις ενέργειες που θα ακολουθήσουν (*PIC*). Τα *PIEvents* εξαρτώνται από τον *event handler* της πλατφόρμας για να λειτουργήσουν. Στο Σχήμα 7 παρουσιάζονται *PIEvents* που σχετίζονται με ενέργειες κατά την χρήση της διεπαφής με τον χρήστη.



Σχήμα 7: PIEvents που σχετίζονται με την διεπαφή του χρήστη και τις ενέργειές του.

- Εκτός από τα *PIEvent* και *PIActivity* στο Σχήμα 6 εμφανίζεται το στοιχείο *PIConnector*. Σε ένα σύστημα μπορούν να υπάρχουν πολλά *PIConnector*.. Το *PIConnector* αποτελεί τον κατέρα για αρκετά στοιχεία που ανήκουν στο *business* πακέτο. Έτσι στο Σχήμα 8 βλέ-

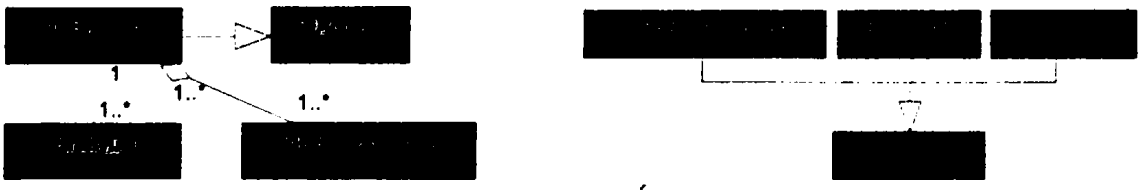


Σχήμα 8: Στοιχεία που ανήκουν στο business πακέτο.

πουμε τα στοιχεία *PIDBConnection*, *PINetworkConnection*, *PIDiskConnection* και *PITechnologyConnection* να κληρονομούν τις ιδιότητες του *PIConnector*. Επίσης με κάθε στοιχείο που χαρακτηρίζεται από το *PIConnector* μπορεί να συσχετίζονται και ένα ή περισσότερα *PIDataStructure*. Αυτό συμβαίνει διότι γενικά το στοιχείο *PIConnector* χρησιμοποιείται για την διαβίβαση δεδομένων. Η δομή των δεδομένων στο μοντέλο απεικονίζονται με την βοήθεια του *PIDataStructure*.

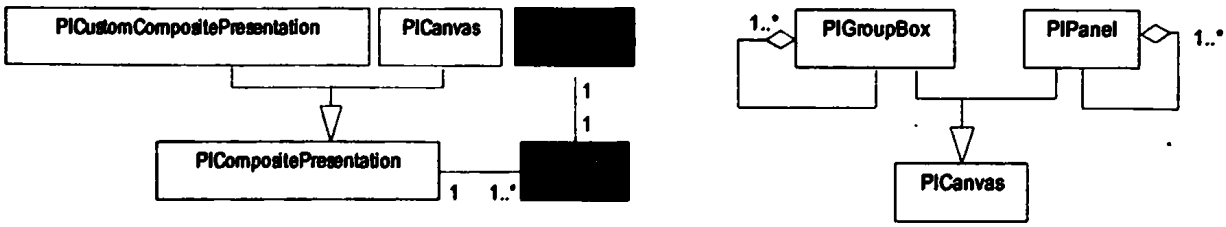
Το στοιχείο *PIDBConnection* χρησιμοποιείται ως το συνδετικό μέσο στοιχείων του *BIS* με κάποια βάση δεδομένων. Στο Σχήμα 9 παρουσιάζουμε τα στοιχεία που έχουν άμεση σχέση με την συνύπαρξη μίας βάσης δεδομένων. Χρήση του *PIDBConnection* όπως είπαμε γίνεται από στοιχεία τα οποία ενεργούν σε μία βάση δεδομένων *PIDBActivity*. Το

κάθε *PIDBActivity* μπορεί να χρησιμοποιεί πολλά *PIDBConnection* αλλά και κάθε *PIDBConnection* μπορεί να εξυπηρετεί διαφορετικά *PIDBActivity*. Το *PIDBActivity* κληρονομεί τις ιδιότητες του *PIActivity*. Όταν οι ενέργειες αφορούν ένα ή περισσότερα πεδία της βάσης δεδομένων τότε αυτά μοντελοποιούνται με τη βοήθεια του *PIEntityData*. Τα στοιχεία *PIDBRawCommand*, *PIDBTable* και *PIDBQuery* κληρονομούν τα χαρακτηριστικά του *PIDBActivity*.



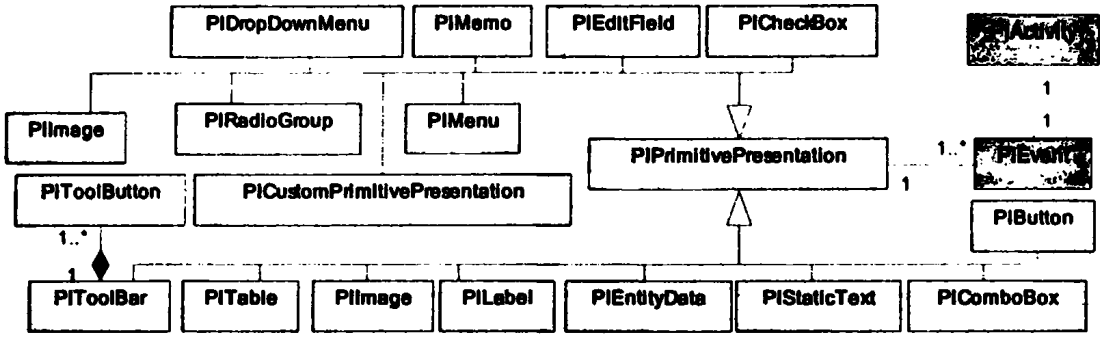
Σχήμα 9: ***Σχέσεις σε στοιχεία τα οποία ανήκουν στο data πακέτο.***

- Στο Σχήμα 6 είδαμε ότι με το *PIForm* μπορούν να συσχετίζονται στοιχεία που ανήκουν στο *composite presentation* πακέτο. Τα *PICompositePresentation* μπορούν είτε να επικάθονται απευθείας στο βασικό στοιχείο *PIForm* είτε σε άλλα στοιχεία του *composite presentation* πακέτου. Σε κάποιες εφαρμογές τα *PrimitivePresentation* βρίσκονται άμεσα στην κύρια φόρμα της εφαρμογής. Εναλλακτικά είναι οργανωμένα σε διαφορετικού τύπου *containers* οι οποίοι εμφανίζονται στο Σχήμα 10 με την μορφή του στοιχείου *PICanvas*. Τα *PICanvas* και *PICustomCompositePresentation* κληρονομούν τις ιδιότητες του *PICompositePresentation*. Αντίστοιχα τα *PIGroupBox* και *PIPanel* που αποτελούν ειδικές περιπτώσεις καμβάδων κληρονομούν το *PICanvas*.



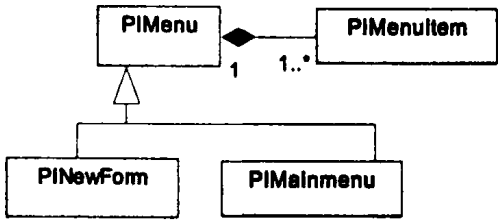
Σχήμα 10: ***Στοιχεία που σχετίζονται με το Composite Presentation.***

- Για την συμπλήρωση του *PIR* έχει απομείνει να περιγράψουμε τη διεπαφή του χρήστη με το σύστημα. Για να το πετύχουμε αυτό χρειαζόμαστε να προσθέσουμε όλα τα στοιχεία όπου στο Σχήμα 6 εμφανίζονται με την μορφή του *PIPrimitivePresentation.*



Σχήμα 11: ***Στοιχεία που σχετίζονται με το Primitive Presentation.***

Στο Σχήμα 11 εμφανίζονται τα στοιχεία της διεπαφής να κληρονομούν τις ιδιότητες του *PIPrimitivePresentation.* Ειδική περίπτωση αποτελούν τα πλαίσια που έχουν την μορφή «μπάρας» *PIToolBar* τα οποία αποτελούνται από στοιχεία *PIToolButton.* Η διεπαφή του χρήστη εκτός από την ύπαρξη μίας μπάρας εργαλείων -*toolbar*- περιμένουμε να έχει κάποιο *menu* το οποίο στο Σχήμα 11 εμφανίζεται με το στοιχείο PIMenu. Το στοιχείο αυτό μπορεί να είναι είτε της μορφής *PINewForm* είτε της μορφής *PIMainMenu,* τα οποία αποτελούνται από ένα πλήθος επιλογών *PIMenuItem* (Σχήμα 12).



Σχήμα 12: ***Σχέσεις στοιχείων που σχετίζονται με τα menus.***

Ο Πίνακας 1 για κάθε *stereotype* που έχουμε ορίσει αναφέρει το αντίστοιχο *Base Class* που προέρχεται από το *EDOC,* το πακέτο στο οποίο έχουμε ταξινομήσει το εκάστοτε στοιχείο και τέλος παραθέτουμε τα *events* που μπορούν να ενεργοποιηθούν από το συγκεκριμένο στοιχείο.

Πίνακας 1: ***Περιγραφή των στοιχείων του PIR.***

Stereotype name	Base Class	Package	Parent	Events
PIEvent	ECA::CCA::Process Component	Busin		

PIActivity	ECA:CCA::Process Component	Busin		
PIConnector	ECA:CCA::Process Component	Busin		
PITechnology Connection	ECA:CCA::Process Component	Busin	PIConnector	OnActivate, OnCanResize, OnClick, OnClose, OnCloseQuery, OnConstrainedResize, OnContextPopup, OnCreate, OnDblClick, OnDeactivate, OnDestroy, OnDockDrop, OnDockOver, OnDragDrop, OnDragOver, OnEndDock, OnGetSiteInfo, OnHelp, OnHide, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnMouseWheel, OnMouseWheelDown, OnMouseWheelUp, OnPaint, OnResize, OnShortCut, OnShow, OnStartDock, OnUnDock
PINetwork Connection	ECA:CCA::Process Component	Busin	PIConnector	OnConnect, OnAccept, OnConnecting, OnDisconnect, OnError, OnLookUp, OnRead, OnWrite OnClientConnect, OnClientDisconnect, OnClientError, OnClientRead, OnClientWrite, OnGetSocket, OnListen, OnThreadEnd, OnThreadStart
PIDiskConnection	ECA:CCA::Process Component	Busin	PIConnector	
PIDataStructure	ECA:CCA::Process Component	Busin		OnDataChange, OnStateChange, OnUpdateData
PIDBConnection	ECA:CCA::Process Component	Busin	PIConnector	AfterDisconnect, BeforeConnect, BeforeDisconnect, OnBeginTransComplete, OnCommitTransComplete, OnConnectComplete, onDisconnect, OnExecuteComplete, OnInfoMessage, OnLogin, OnRollbackTransComplete, OnWillConnect, OnWillExecute
PIDBRawCommand PIDBQuery PIDBTable	ECA:Entity:: Data Manager	Data	PIDB Activity	
PIDBActivity	ECA:Entity:: Data Manager	Data		AfterCancel, AfterClose, AfterDelete, AfterEdit, AfterInsert, AfterOpen, AfterPost, AfterRefresh, AfterScroll, BeforeCancel, BeforeClose, BeforeDelete, BeforeEdit, BeforeInsert, BeforeOpen, BeforePost, BeforeRefresh, BeforeScroll, OnCalcFields, OnDeleteError, OnEndOfRecordset, OnFetchComplete, OnFetchProgress, OnFieldChangeComp, OnFilterRecord, OnMoveComplete, OnNewRecord, OnPostError, OnRecordChangeComplete, OnRecordsetChangeComplete, OnWillChangeField, OnWillChangeRecord, OnWillChangeRecordset, OnWillMove
PIEntityData	ECA:CCA::Entity:: EntityData	Data		
PICanvas	ECA:CCA::Process Component	CPres	PIComposite Presentation	
PIGroupBox	ECA:CCA::Process Component	CPres	PICanvas	OnClick, OnContextPopup, OnDblClick, OnDockDrop, OnDockOver, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnGetSiteInfo, OnMouseDown, OnMouseMove, OnMouseUp, OnStartDock, OnStartDrag, OnUnDock
PIPanel	ECA:CCA::Process Component	CPres	PICanvas	OnCanResize, OnClick, OnConstrainedResize, OnContextPopup, OnDblClick, OnDockDrop, OnDockOver, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnGetSiteInfo, OnMouseDown, OnMouseMove, OnMouseUp, OnResize, OnStartDock, OnStartDrag, OnUnDock
PIForm	ECA:CCA::Process Component	CPres		OnActivate, OnCanResize, OnClick, OnClose, OnCloseQuery,OnConstrainedResize,OnContextPopup, OnCreate, OnDblClick, OnDeactivate, OnDestroy, OnDockDrop, OnDockOver, OnDragDrop, OnDragOver, OnEndDock, OnGetSiteInfo, OnHelp, OnHide, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnMouseWheel, OnMouseWheelDown, OnMouseWheelUp, OnPaint, OnResize, OnShortCut, OnShow, OnStartDock, OnUnDock
PICustomComposite Presentation	ECA:CCA::Process Component	CPres		
PIComposite Presentation	ECA:CCA::Process Component	CPres		
PIPrimitive Presentation	ECA:CCA::Process Component	PPres		

PICustom-Primitive Presentation	ECA:CCA::Process Component	PPres	PIPrimitive Presentation	OnActivate, OnCanResize, OnClick, OnClose, OnCloseQuery,OnConstrainedResize,OnContextPopup, OnCreate, OnDblClick, OnDeactivate, OnDestroy, OnDockDrop, OnDockOver, OnDragDrop, OnDragOver, OnEndDock, OnGetSiteInfo, OnHelp, OnHide, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnMouseWheel, OnMouseWheelDown, OnMouseWheelUp, OnPaint, OnResize, OnShortCut, OnShow, OnStartDock, OnUnDock
PIMenu	ECA:CCA::Process Component	PPres	PIPrimitive Presentation	
PINewForm PIMainMenu	ECA:CCA::Process Component	PPres	PIMenu	
PIMenuItem	ECA:CCA::Process Component	PPres		OnAdvancedDrawItem, OnClick, OnDrawItem, OnMeasureItem
PIToolBar	ECA:CCA::Process Component	PPres		
PIToolButton	ECA:CCA::Process Component	PPres	PIPrimitive Presentation	OnClick, OnContextPopup , OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnMouseDown, OnMouseMove, OnMouseUp, OnStartDock, OnStartDrag
PIImage	ECA:CCA::Process Component	PPres	PIPrimitive Presentation	OnClick
PIDataField	ECA:CCA::Process Component	PPres	PIPrimitive Presentation	
PIButton	ECA:CCA::Process Component	PPres	PIPrimitive Presentation	OnClick, OnContextPopup, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnStartDock, OnStartDrag
PILabel	ECA:CCA::Process Component	PPres	PIPrimitive Presentation	OnClick, OnContextPopup, OnDblClick, OnDragDrop, OnEndDock, OnEndDrag, OnMouseDown, OnMouseMove, OnMouseUp, OnStartDock, OnStartDrag
PIDropDownMenu		PPres	PIPrimitive Presentation	OnChange, OnClick, OnContextPopup, OnDblClick, OnDragDrop, OnDrawItem, OnDropDown, OnEndDock, OnEndDrag, OnEnter, OnExit, OnKeyDown, OnKeyPress, OnKeyUp, OnMeasureMent, OnStartDock, OnStartDrag
PICheckBox	ECA:CCA::Process Component	PPres	PIPrimitive Presentation	OnClick, OnContextPopup, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnStartDock, OnStartDrag
PIEditField	ECA:CCA::Process Component	PPres	PIPrimitive Presentation	OnClick, OnContextPopup, OnDblClick, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnStartDock, OnStartDrag
PIComboBox	ECA:CCA::Process Component	PPres	PIPrimitive Presentation	OnChange, OnClick, OnContextPopup, OnDblClick, OnDragDrop, OnDrawItem, OnDropDown, OnEndDock, OnEndDrag, OnEnter, OnExit, OnKeyDown, OnKeyPress, OnKeyUp, OnMeasureMent, OnStartDock, OnStartDrag
PITable	ECA:CCA::Process Component	PPres	PIPrimitive Presentation	OnClick, OnColumnMoved, OnContextPopup, OnDblClick, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnGetEditMask, OnGetEditText, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnMouseWheelDown, OnMouseWheelUp, OnRowMoved, OnSelectCell, OnSetEditText, OnStartDock, OnStartDrag, OnTopLeftChanged
PIMemo	ECA:CCA::Process Component	PPres	PIPrimitive Presentation	OnChange, OnClick, OnContextPopup, OnDblClick, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnStartDock, OnStartDrag
PIRadioGroup	ECA:CCA::Process Component	PPres	PIPrimitive Presentation	OnClick, OnContextPopup, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnStartDock, OnStartDrag
PIStaticText	ECA:CCA::Process Component	PPres	PIPrimitive Presentation	OnClick, OnContextPopup, OnDblClick, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnMouseDown, OnMousemove, OnMouseUp, OnStartDock, OnStartDrag

3.3 Αναλυτική περιγραφή των αναπαραστάσεων της Delphi

Στην παρούσα ενότητα παρουσιάζουμε τα *stereotypes* μίας αναπαράστασης που μπορεί να χρησιμοποιηθεί για την περιγραφή ενός *BIS* που έχει δομηθεί χρησιμοποιώντας *Borland Delphi* και *ADO*. Η *Delphi* καθώς και άλλες παρόμοιες τεχνολογικές πλατφόρμες προσφέρουν σήμερα δύο ειδών βασικά δομικά στοιχεία:

- Τετριμμένα αλλά αναγκαία αντικείμενα που χρησιμοποιούνται για την υλοποίηση των βασικών λειτουργιών του συστήματος και αποτελούν τον κορμό του.
- Ένα τεράστιο πλήθος από άλλα έτοιμα στοιχεία τα οποία εκτελούν πολύ συγκεκριμένες λειτουργίες. Αυτά τα στοιχεία μπορεί να είναι από μόνα τους μία ολόκληρη συμπληρωματική πλατφόρμα και πολλές φορές στο περιβάλλον *win32* εμφανίζονται με την μορφή έτοιμων *dlls*.

Τα αντικείμενα που παρουσιάζουν ενδιαφέρον και εμείς εξετάζουμε είναι εκείνα τα οποία σχηματίζουν τον κορμό του συστήματος. Έτσι ο παραπάνω διαχωρισμός μας προσέφερε περίπου 60 βασικά αντικείμενα που μπορούν να διαχωριστούν στα παρακάτω πακέτα όπως και στην περίπτωση του *PIR*:

- Στοιχεία του πακέτου *primitive presentation* τα οποία δημιουργούν και συνθέτουν την διεπαφή του χρήστη με το σύστημα (κουμπιά, εικόνες, *menu* κτλ) που αντιστοιχίζονται στο *primitive presentation* πακέτο. Αντικείμενα που σχετίζονται με την υποστήριξη των παραπάνω *primitive presentation*, όπως καμβάδες, *panels* και φόρμες, περιέχονται στο *composite presentation* πακέτο.
- Στοιχεία του *data* πακέτου που σχετίζονται με την χρήση μίας συνηθισμένης βάσης δεδομένων.
- Στοιχεία που ενεργούν στο παρασκήνιο του συστήματος και τα αντιστοιχούμε στο *business* πακέτο.

Για τον ορισμό των *stereotypes* χρησιμοποιούμε την μορφή με την οποία παρουσιάσαμε τα στοιχεία του *PIR* της παραγράφου 3.2.

Stereotypes* που ανήκουν στο πακέτο *Business

- *DelphiEvent, DelphiAction*: Τα δύο αυτά *stereotypes* ορίζονται παρόμοια με τα στοιχεία *PIEvents* και *PIActivity* της παραγράφου 3.2.
- *DelphiBusiness*: Το στοιχείο *DelphiBusiness* χρησιμοποιείται στην μοντελοποίηση για να εκφράσει στοιχεία που ενεργούν στο παρασκήνιο. Χρησιμοποιείται ως πατρικό *stereotype* για τα παρακάτω *stereotypes* που σχετίζονται όλα με εργασίες στο παρασκήνιο. Για τον ορισμό του *DelphiBusiness* σαν βασική κλάση έχει επιλεγεί η *ECA:CCA::ProcessComponent*.
- *DelphiRegValues*: Το *DelphiRegValues* είναι ένα στοιχείο το οποίο χρησιμοποιείται για να μοντελοποιήσει ένα στοιχείο της *Delphi* του οποίου η αρμοδιότητα είναι να διαβάζει στοιχεία από την *registry* των *Windows*. Για τον ορισμό του *DelphiRegValues* ως βασική κλάση έχει επιλεγεί η *ECA:CCA::ProcessComponent* και πατέρας του είναι το *DelphiBusiness*.
- *DelphiClientSocket, DelphiServerSocket*: Η *Delphi* για να υλοποιήσει δικτυακές εφαρμογές με τη βοήθεια *TCP/IP* έχει 2 έτοιμα *APIs* από τα οποία τα ένα χρησιμοποιείται για να δέχεται συνδέσεις (*host*) και το άλλο είναι για να κάνει αίτημα σύνδεσης σε κάποιον *host* (*client*). Η πρώτη περίπτωση του *host* γίνεται με το στοιχείο *DelphiServerSocket* ενώ το δεύτερο με τη βοήθεια του *DelphiClientSocket*. Η βασική λειτουργία, η οποία είναι ίδια και στα δύο, των δύο αντικειμένων αυτών είναι η μεταβίβαση δεδομένων. Η κύρια διαφορά τους αφορά τον τρόπο έναρξης της επικοινωνίας των δύο μηχανών. Η εφαρμογή η οποία έχει το *DelphiServerSocket* έχει την δυνατότητα να εξυπηρετήσει ταυτόχρονα πολλούς *clients* (συνήθως το πετυχαίνει με *threads*). Η διαφορά των δύο στοιχείων αυτών γίνεται άμεσα αντιληπτή από τα εμπλεκόμενα *events* (Πίνακας 2). Για τον ορισμό τους σαν βασική κλάση έχει επιλεγεί η *ECA:CCA::ProcessComponent* και πατέρας τους είναι το *DelphiBusiness*.
- *DelphiDataSource*: Δεδομένα τα οποία επιστρέφονται από το αποτέλεσμα μίας ενέργειας ενός άλλου στοιχείου επιστρέφονται μέσω του στοιχείου *DelphiDataSource*. Για τον ορισμό τους σαν βασική κλάση έχει επιλεγεί η *ECA:CCA::ProcessComponent* και πατέρας τους είναι το *DelphiBusiness*.
- *ADOConnection*: Τα *ADOConnection* είναι στοιχεία που ανήκουν στην τεχνολογική πλατφόρμα *ADO*. Και τα οποία χρησιμοποιούνται για την διεπαφή στοιχείων της *Delphi*

με μία βάση δεδομένων. Για τον ορισμό του *stereotype* αυτού ως βασική κλάση έχουμε επιλέξει την *ECA:CCA::ProcessComponent* ενώ πατέρας είναι το *DelphiBusiness*.

- *DelphiAPI*: Στην *Delphi* υπάρχει μεγάλο πλήθος από έτοιμα προγράμματα τα οποία είτε δίδονται με τη μορφή βιβλιοθήκης είτε χρησιμοποιούνται ως εξωτερικά προγράμματα (π.χ. έτοιμα *dlls*). Τα στοιχεία (στην υλοποίηση) που βασίζονται στα παραπάνω εξωτερικά στοιχεία τα αναπαριστούμε με το *DelphiAPI*. Για τον ορισμό του *stereotype* αυτού ως βασική κλάση έχουμε επιλέξει την *ECA:CCA::ProcessComponent* ενώ πατέρας είναι το *DelphiBusiness*.
- *DelphiWebBrowser*: Είναι στοιχεία, για παράδειγμα έτοιμα *Application Programming Interface (API)*, που έχουν τη δυνατότητα να διαβάζει αρχεία *html* και να τα εμφανίζει σε ένα παράθυρο. Εμείς το μόνο που κάνουμε είναι να του παρέχουμε έναν χώρο στην διεπαφή του χρήστη για την εμφάνιση των σελίδων και σαν είσοδο του δίνουμε ένα αρχείο *html*. Γίνεται φανερό ότι εμείς δεν γνωρίζουμε πως τα πετυχαίνει όλα αυτά. Έτσι το θεωρούμε ως στοιχείο του *business* πακέτου (αν και έχει σχέση με την διεπαφή χρήστη) με πατέρα το *DelphiBusiness* και με βασική κλάση την *ECA:CCA::ProcessComponent*.

Stereotypes* που ανήκουν στο πακέτο *Data

- *DelphDBAction*: Σχετικά με ενέργειες που σχετίζονται με την αλληλεπίδραση με μία βάση δεδομένων έχει οριστεί το *DelphDBAction*. Η περίπτωση αυτή είναι η γενικότερη που έχουμε στη διάθεσή μας για τον σκοπό αυτό. Επειδή έχει να κάνει με τον χειρισμό δεδομένων, σαν βασική κλάση έχει επιλεγεί η *ECA:Entity::DataManager*.
- *ADOTable, ADOQuery*: Για ενέργειες που αφορούν την αναζήτηση δεδομένων από μία βάση δεδομένων με την χρήση *ADO components* έχουμε ορίσει τις *ADOTable* και *ADOQuery*. Η *ADOTable* σχετίζεται άμεσα με την χρήση ενός ολόκληρου πίνακα από τη βάση δεδομένων ενώ η *ADOQuery* είναι πιο ευέλικτη και ζητάει στήλες από διάφορους πίνακες μίας βάσης δεδομένων. Επειδή έχει να κάνει με τον χειρισμό δεδομένων, σαν βασική κλάση έχει επιλεγεί η *ECA:Entity::DataManager* και πατέρας η *DelphDBAction*.
- *DelphiADOCommand*: Η *DelphiADOCommand* χρησιμοποιείται για τις παραπλήσιες ενέργειες με αυτές που παρουσιάσαμε προηγουμένως στις *ADOTable* και *ADOQuery*. Στην προκειμένη περίπτωση οι εντολές προς τη βάση δεδομένων είναι οποιασδήποτε φύσης, δηλαδή δεν περιοριζόμαστε σε εντολές αναζήτησης δεδομένων. Όμοια με τις

παραπάνω περιπτώσεις ορίζεται ως βασική κλάση η *ECA:Entity::DataManager* και πατέρας η *DelphDBAction.*

- *ADODataSet*: Δεδομένα τα οποία επιστρέφονται με την μορφή που έχουν τα *DelphiDataSource* μέσω ενός *ADO component*, δηλαδή έχουν προέλθει από κάποια αίτηση δεδομένων προς κάποια βάση δεδομένων τα αντιστοιχούμε σε στοιχεία του τύπου *ADODataSet.* Έτσι ορίζουμε πατέρα το στοιχείο *DelphiDataSource* και ως βασική κλάση την *ECA:CCA::ProcessComponent.*
- *DelphiDBNavigator*: Το *DelphiDBNavigator* είναι ένα αρκετά σύνθετο αντικείμενο στην *Delphi* που είναι πολύ χρήσιμο και πρακτικό για τους προγραμματιστές. Ο ρόλος του είναι να διαχειρίζεται εύκολα στοιχεία μίας βάσης δεδομένων. Τα στοιχεία είναι οργανωμένα με την μορφή ενός *DelphiDataSource* και αυτά χωρίζονται σε πλειάδες. Αποτελείται από 10 διαφορετικές λειτουργίες οι οποίες χειρίζονται τα παραπάνω δεδομένα και ενεργοποιούνται μέσω της διεπαφής χρήστη από 10 ξεχωριστά κουμπιά.
    - *First* Φέρνει τα στοιχεία της πρώτης πλειάδας του *dataset.*
    - *Prior* Φέρνει τα στοιχεία της προηγούμενης πλειάδας του *dataset.*
    - *Next* Φέρνει τα στοιχεία της επόμενης πλειάδας του *dataset.*
    - *Last* Φέρνει τα στοιχεία της τελευταίας πλειάδας του *dataset.*
    - *Insert* Σε αυτή τη περίπτωση επεμβαίνει στα δεδομένα του *dataset* και εισάγει μία πλειάδα ακριβώς πριν από την θέση στην οποία βρισκόμαστε. Τα νέα στοιχεία είναι όλα κενά. Δηλαδή εάν επεξεργαζόμαστε έναν πίνακα θα προστεθεί μία καινούργια κενή γραμμή – πλειάδα στον πίνακά μας. Κατά κάποιο τρόπο είναι σε *"Insert state"*.
    - *Delete* Σε αυτή τη περίπτωση επεμβαίνει στα δεδομένα του *dataset* και σβήνει την τρέχουσα πλειάδα του στην οποία βρισκόμαστε. Κατά κάποιο τρόπο είναι σε *"Delete state"*.
    - *Edit* Σε αυτή τη περίπτωση επεμβαίνει στα δεδομένα του *dataset* μεταβάλει στοιχεία της τρέχουσας πλειάδας. Κατά κάποιο τρόπο είναι σε *"Edit state"*.
    - *Post* Γράφει τις αλλαγές που έχουμε κάνει στην αντίστοιχη θέση στη βάση δεδομένων (*commit*).
    - *Cancel* Ακυρώνει τις αλλαγές που κάναμε στη τρέχουσα πλειάδα.

- *Refresh* Αρχικά σβήνει τους *"data control display buffers"*, έπειτα ανανεώνει τους *buffers* του με βάση το αντίστοιχο πίνακα ή *query*. Ιδιαίτερα χρήσιμο στις περιπτώσεις που ενδέχεται τα δεδομένα να μεταβάλλονται από κάποια άλλη εφαρμογή ή άλλον χρήστη.

Το *stereotype* ορίζεται με βασική κλάση την *ECA:Entity::DataManager*

- *DelphiDBGuiIO*: Η *Delphi* έχει κάποια αντικείμενα τα οποία εμφανίζονται στην γραφική διεπαφή (ορίζονται στο *primitive presentation* πακέτο στο *DelphiPresentationInput*) με τον χρήστη και τα δεδομένα τους προέρχονται άμεσα και επηρεάζουν άμεσα κάποιο κελί (*cell*) σε μία βάση δεδομένων. Τέτοια στοιχεία παρουσιάζονται παρακάτω και διαχωρίζονται με βάση την μορφή που έχουν αυτά στην διεπαφή του χρήστη, δηλαδή εάν είναι κάποιο *checkbox* ή κάποιος πίνακας. Όλα αυτά οργανώνονται στο *DelphiDBGuiIO*. Για το *stereotype* αυτού ορίζεται ως πατέρας το *DelphiPresentationInput* (ανήκει στο *primitive presentation*) και ως βασική κλάση το *ECA:CCA::ProcessComponent*.
- *DelphiDBImage*: Το *DelphiDBImage* εμφανίζει μία εικόνα στην διεπαφή του χρήστη η οποία προέρχεται από μία βάση δεδομένων. Το *stereotype* έχει σαν πατέρα το *DelphiDBGuiIO* και βασική κλάση το *ECA:CCA::ProcessComponent*.
- *DelphiDBMemo, DelphiDBField*: Τα *DelphiDBMemo* και *DelphiDBField* είναι στοιχεία τα οποία εμφανίζονται στην διεπαφή χρήστη με την μορφή ενός πλαισίου στο οποίο μπορούμε να εισάγουμε κείμενο. Το κείμενο αυτό επηρεάζει άμεσα κάποιο κελί σε μία βάση δεδομένων. Η κύρια διαφορά ανάμεσα στα δύο αυτά στοιχεία είναι το μέγιστο μέγεθος της συμβολοσειράς που επιτρέπουν. Τα *stereotypes* έχουν σαν πατέρα το *DelphiDBGuiIO* και βασική κλάση το *ECA:CCA::ProcessComponent*.
- *DelphiDBText*: Το *DelphiDBText* παρουσιάζεται στη διεπαφή χρήστη ως αμετάβλητο κείμενο το οποίο προέρχεται άμεσα από το κείμενο ενός κελί μίας βάσης δεδομένων. Το *stereotype* έχει σαν πατέρα το *DelphiDBGuiIO* και βασική κλάση το *ECA:CCA::ProcessComponent*
- *DelphiDBCheckBox*: Το *DelphiDBCheckBox* εμφανίζει ένα *checkbox* στην διεπαφή χρήστη η οποία επηρεάζει την τιμή ενός *cell* σε μία βάση δεδομένων η οποία θα είναι *true* ή *false* (*boolean*). Το *stereotype* έχει σαν πατέρα το *DelphiDBGuiIO* και βασική κλάση το *ECA:CCA::ProcessComponent*.

- *DelphiDBGrid*: Το *DelphiDBGrid* είναι ένα στοιχείο το οποίο εμφανίζεται στην γραφική διεπαφή του χρήστη με τη μορφή πίνακα. Τα στοιχεία του πίνακα αυτού γεμίζουν με ένα σύνολο πλειάδων τα οποία προέρχονται από κάποιο *DelphiDataSource* ή από κάποιο *ADODataSet*. Το *stereotype* έχει σαν πατέρα το *DelphiDBGuiIO* και βασική κλάση το *ECA:CCA::ProcessComponent*.
- *DelphiDBLookUpListBox*, *DelphiDBComboBox*, *DelphiDBLookupComboBox*: Στην διεπαφή του χρήστη στις περιπτώσεις που έχουμε αντικείμενα τα οποία σχετίζονται με επιλογή ενός από μία λίστα πολλαπλών επιλογών αντιστοιχεί ένα από τα *DelphiDBLookUpListBox*, *DelphiDBComboBox* και *DelphiDBLookupComboBox*. Η διαφορά είναι στον τρόπο με τον οποίο εμφανίζονται στην διεπαφή του χρήστη. Όπως προδίδουν τα ονόματά τους εμφανίζονται στην διεπαφή χρήστη με την μορφή των *DelphiListBox* και *DelphiComboBox* των οποίων τα πεδία (με τις επιλογές) γεμίζουν με βάση τα στοιχεία μία στήλης σε μία βάση δεδομένων (γίνεται μέσω *query* οπότε μπορούμε να φιλτράρουμε αυτά που εμείς επιθυμούμε). Τα *stereotypes DelphiDBLookUpListBox* και *DelphiDBComboBox* έχουν σαν πατέρα το *DelphiDBGuiIO* και όλα για βασική κλάση το *ECA:CCA::ProcessComponent*.

Stereotypes* που ανήκουν στο πακέτο *Composite Presentation

- *DelphiCompositePresentation*: Είναι το κύριο στοιχείο το οποίο ανήκει στο *composite presentation* πακέτο. Είναι ένα αντικείμενο του οποίου η χρήση εστιάζει στην υποστήριξη αντικειμένων που εμφανίζονται στην διεπαφή με τον χρήστη. Το *stereotype* για τον ορισμό της έχει ως βασική κλάση την *ECA:CCA::ProcessComponent*.
- *DelphiImageList*: Το *DelphiImageList* είναι ένα στοιχείο το οποίο έχει με κάποιο τρόπο δομημένες μία σειρά από εικόνες. Παράδειγμα χρήσης ενός τέτοιου στοιχείου είναι όταν οι συντομεύσεις ενός *toolbar* έχουν εικονίδια τότε αυτά είναι όλα μαζί καταχωρημένα σε ένα *DelphiImageList*. Το *stereotype* για τον ορισμό του έχει ως βασική κλάση την *ECA:CCA::ProcessComponent* και πατέρας είναι το *DelphiCompositePresentation*.
- *DelphiPageControl*, *DelphiCtrlGrid*, *DelphiTabSheet*: Υπάρχουν κάποια στοιχεία τα οποία έχουν την ιδιότητα των καμβάδων αλλά διαφέρουν στον τρόπο με τον οποίο χειρίζονται τα στοιχεία τους. Τα *stereotypes* για τον ορισμό τους έχουν ως βασική κλάση την *ECA:CCA::ProcessComponent* και πατέρας είναι το *DelphiCompositePresentation*.

- *DelphiCanvas, DelphiPanel, DelphiGroupBox, DelphiRadioGroup*: Τα *stereotypes* αυτά χρησιμοποιούνται για την μοντελοποίηση καμβάδων οι οποίοι έχουν κάποια ιδιαίτερη λειτουργικότητα. Το *DelphiRadioGroup* φιλοξενεί ένα πλήθος από *checkboxes* και επιτρέπει για παράδειγμα μόνο ένα να είναι επιλεγμένο (μπορεί να ισχύει οποιοσδήποτε κανόνας). Το *DelphiPanel* είναι ο πιο απλοϊκός καμβάς που διατίθεται στη *Delphi* και περιέχει στο εσωτερικό του οποιαδήποτε στοιχεία. Ένα παράδειγμα για την χρησιμότητά του είναι η διευκόλυνση που παρέχει στον προγραμματιστή όταν θέλει να προβεί σε μία ομαδική ενέργεια στα αντικείμενα του *Panel* π.χ. μετακίνηση κατά 10 *pixels* ή απόκρυψη όλων των στοιχείων. Για τα *stereotypes* αυτά σαν πατέρας έχει οριστεί το *stereotype DelphiCanvas* και έχουν όλα τους σαν βασική κλάση την *ECA:CCA::ProcessComponent.*
- *DelphiForm*: Η *DelphiForm* είναι κάτι αντίστοιχο με την *DelphiCanvas* αλλά χρησιμοποιείται όταν έχουμε την εμφάνιση μίας νέας φόρμας, συνήθως σαν νέο παράθυρο. Είναι το βασικό στοιχείο για την μοντελοποίηση στο *PSR* το οποίο θα φανεί και στην συνέχεια με την περιγραφή των *metamodels*. Βασική κλάση για το *stereotype* αυτό έχει επιλεγεί η *ECA:CCA::ProcessComponent.*
- *DelphiCustomCompositePresentation*: Στη περίπτωση που κάποιο αντικείμενο δεν πληροί τις προδιαγραφές κάποιου από τα παραπάνω στοιχεία και ανήκει στο *composite presentation* πακέτο τότε αυτό μπορεί να απεικονιστεί με το *DelphiCustomCompositePresentation*. Το *stereotype* αυτό έχει ως βασική κλάση του *ECA:CCA::ProcessComponent* και πατέρας είναι το *DelphiCompositePresentation.*

Stereotypes* που ανήκουν στο πακέτο *Primitive Presentation

- *DelphiPrimitivePresentation*: Είναι το κύριο στοιχείο το οποίο ανήκει στο *primitive presentation* πακέτο. Είναι ένα αντικείμενο το οποίο εμφανίζεται στην διεπαφή με τον χρήστη. Το *stereotype* για τον ορισμό του έχει ως βασική κλάση την *ECA:CCA::ProcessComponent.*
- *DelphiPresentationInput*: Το *stereotype DelphiPresentationInput* αντιπροσωπεύει στοιχεία τα οποία ανήκουν στην διεπαφή με τον χρήστη τα οποία χρησιμεύουν στην εισαγωγή παραμέτρων για τον σχηματισμό φορμών. Ως πατέρας του *stereotype* αυτού έχει οριστεί το *DelphiPrimitivePresentation* και βασική κλάση η *ECA:CCA::ProcessComponent.*

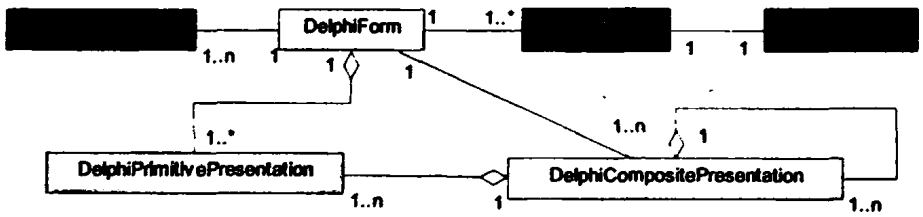
- *DelphiToolBar, DelphiToolButton*: Στις περισσότερες εφαρμογές υπάρχουν για την διευκόλυνση του χρήστη μία μπάρα *DelphiToolBar* από συντομεύσεις *DelphiToolButton*. Οι συντομεύσεις αυτές μοιάζουν με μικρά κουμπιά στα οποία μπορούν να περιέχουν είτε κείμενο είτε κάποιο εικονίδιο (μπορεί να προέρχεται από κάποιο *DelphiImageList*). Βασική κλάση για αυτά έχει επιλεγεί το *ECA:CCA::ProcessComponent* και ως πατέρας για την *DelphiToolBar* ορίζεται το *DelphiPrimitivePresentation*.
- *DelphiMenuItem, DelphiMenu, DelphiMainMenu, DelphiPopUpMenu*: Μία άλλη μορφή για την παρουσίαση στην διεπαφή συντομεύσεων είναι είτε με την χρήση *popup menus* (*DelphiPopUpMenu*), είτε με μορφή που συναντάμε σε συνήθη εφαρμογές στο πάνω μέρος της διεπαφής με τον χρήστη με επιλογές: *File, Edit, View, Help* κτλ και εμφανίζει η κάθε μία τους ένα ιδιαίτερο *dropdown menu*. Για τα *stereotypes* των παραπάνω έχουν οριστεί τα *DelphiPopUpMenu* και *DelphiMainMenu* και ως πατέρας αυτών το *DelphiMenu*. Οι συντομεύσεις έχουν και αυτές αντιστοίχηση στο μοντέλο ως *DelphiMenuItem*. Όλα ως βασική κλάση έχουν το *ECA:CCA::ProcessComponent*.
- *DelphiStatusBar*: Σε πολλές εφαρμογές έχουμε την εμφάνιση της *status bar* συνήθως στο κάτω μέρος της οθόνης και εμφανίζει εξειδικευμένη πληροφορία ανάλογα με την περίσταση και με την επιθυμία του σχεδιαστή του συστήματος. Για τη μοντελοποίηση έχουμε ορίσει το *stereotype DelphiStatusBar*. Βασική κλάση για αυτό έχει επιλεγεί το *ECA:CCA::ProcessComponent* και ως πατέρας ορίζεται το *DelphiPrimitivePresentation*.
- *DelphiImage*: Τα στοιχεία που εμφανίζονται στην διεπαφή χρήστη με τη μορφή μίας εικόνας τα αναπαριστούμε με το *stereotype DelphiImage*. Βασική κλάση για αυτό έχει επιλεγεί το *ECA:CCA::ProcessComponent* και ως πατέρας ορίζεται το *DelphiPrimitivePresentation*.
- *DelphiMemo, DelphiMemoField, DelphiEditField*: Όλα τα στοιχεία της διεπαφής με τον χρήστη που εμφανίζονται με την μορφή ενός ειδικού πλαισίου γα την εισαγωγή κειμένου, γνωστό ως *edit* και *memo box* τα αναπαριστούμε με τα παραπάνω *stereotypes*. Η κύρια διαφορά τους είναι στο μέγιστο πλήθος χαρακτήρων που επιτρέπουν. Βασική κλάση για τα *stereotype* αυτά είναι η *ECA:CCA::ProcessComponent* και ως πατέρας ορίζεται το *DelphiPresentationInput*.
- *DelphiStaticText*: Στοιχεία της διεπαφής με τον χρήστη που εμφανίζονται με την μορφή κειμένου, χωρίς να δίνεται η δυνατότητα το επεξεργαστούμε, τα αναπαριστούμε με το

stereotype DelphiStaticText. Πατέρας του στοιχείου είναι το *DelphiPresentationInput* και βασική κλάση είναι το *ECA:CCA::ProcessComponent*.

- *DelphiCheckBox*: Στοιχεία της διεπαφής με τον χρήστη που εμφανίζονται με την μορφή *checkbox* τα αναπαριστούμε με το *stereotype DelphiCheckBox*. Πατέρας του στοιχείου είναι το *DelphiPresentationInput* και βασική κλάση είναι το *ECA:CCA::ProcessComponent*.
- *DelphiGrid, DelphiStringGrid, DelphiDrawGrid*: Αντιστοιχεί σε στοιχεία τα οποία εμφανίζονται στην διεπαφή με τον χρήστη με την μορφή πίνακα. Υπάρχει περίπτωση αυτόν τον πίνακα αυτόν να μπορεί ο χρήστης να τον επεξεργαστεί. Τα στοιχεία στα κελιά του πίνακα μπορεί να είναι παράμετροι για κάποια φόρμα οπότε σαν πατέρας των *stereotypes* αυτών ορίζεται το *DelphiPresentationInput* και ως βασική κλάση η *ECA:CCA: :ProcessComponent*.
- *DelphiListBox, DelphiComboBox*: Αντικείμενα τα οποία εμφανίζονται στην διεπαφή με τον χρήστη με στόχο την επιλογή ενός στοιχείου από μία λίστα είναι τα *DelphiListBox* και *DelphiComboBox*. Η διαφορά των δύο αυτών στοιχείων εστιάζεται στον διαφορετικό τρόπο με τον οποίο εμφανίζονται στην φόρμα. Για τον ορισμό των *stereotype* τους πατέρας ορίζεται το *DelphiPresentationInput* και βασική κλάση η *ECA:CCA::Process-Component*.
- *DelphiLabel*: Στοιχεία τα οποία εμφανίζονται στην οθόνη με την μορφή απλού κειμένου είναι τα *DelphiLabel*. Η διαφορά με το *DelphiStaticText* είναι ότι το *DelphiStaticText* μπορεί να αποτελέσει το πεδίο μίας μεταβλητής κατά την αίτηση πραγματοποίησης μίας φόρμας, ενώ το *Label* χρησιμοποιείται παθητικά για την εκτύπωση μίας επικεφαλίδας ή ενός κειμένου στην διεπαφή με τον χρήστη. Για τον ορισμό του *stereotype* πατέρας ορίζεται το *DelphiPrimitivePresentation* και βασική κλάση η *ECA:CCA: :ProcessComponent*.
- *DelphiButton*: Στοιχεία της διεπαφής με τον χρήστη που εμφανίζονται με την μορφή κουμπιών τα αναπαριστούμε με το *stereotype DelphiButton*. Για τον ορισμό του *stereotype* πατέρας ορίζεται το *DelphiPrimitivePresentation* και βασική κλάση η *ECA:CCA::Process-Component*.
- *DelphiIntegerField, DelphiWideStringField, DelphiStringField, DelphiBooleanField*: Όλα τα στοιχεία αυτά χρησιμοποιούνται για την περιγραφή μεταβλητών και η τιμή τους μπορεί να εμφανίζεται (μπορεί και όχι) στην διεπαφή με τον χρήστη.

Metamodels του *Delphi PSR*

Όπως και στη περίπτωση του *PIR* έτσι και στο *Delphi PSR* παρουσιάζουμε τις συσχετίσεις μεταξύ των *stereotypes* που ορίσαμε. Στο Σχήμα 13 το στοιχείο *DelphiForm* είναι αυτό που αποτελεί την βάση για το σύστημά μας. Σε αυτό επικάθονται ένα ή περισσότερα στοιχεία που ανήκουν στο *business* πακέτο με την μορφή του *DelphiBusiness*. Η *DelphiForm* μπορεί να προκαλεί την εκκίνηση κάποιου *DelphiAction* μέσω γεγονότων *DelphiEvent*. Στοιχεία τα οποία ανήκουν στην γραφική διεπαφή με τον χρήστη μπορούν να επικάθονται είτε απευθείας στην *DelphiForm* είτε σε καμβάδες *DelphiCompositePresentation* οι οποίοι ανήκουν στο *composite presentation* πακέτο. Οι καμβάδες μπορούν να περιέχουν και άλλους καμβάδες.



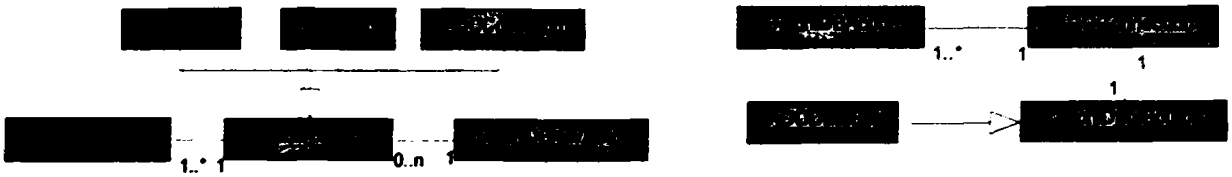
Σχήμα 13: ***Βασική αναπαράσταση του Delphi PSR.***

Στο *business* πακέτο υπάρχει μία σειρά από στοιχεία τα οποία κληρονομούν τις ιδιότητες του *DelphiBusiness* και προσθέτουν κάποιες παραμέτρους σύμφωνα με τον σκοπό τον οποίο επιθυμούν να εξυπηρετήσουν (Σχήμα 14).

Σχήμα 14: ***Στοιχεία του business πακέτου που ανήκουν στο Delphi PSR.***

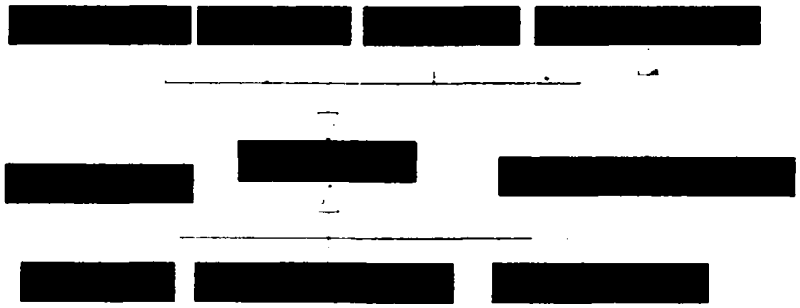
Από τα προαναφερθέντα στοιχεία το *ADOConnection* χρησιμεύει για αρκετά στοιχεία που ανήκουν στο *data* πακέτο. Ένα *ADOConnection* μπορεί να συσχετιστεί με ένα ή περισσότερα *DelphiDBAction*. Εάν επιστρέφονται δεδομένα από μία τέτοια ενέργεια τότε αυτά θα έχουν είτε την μορφή του *DelphiDataSource* είτε του *ADODataSet*. Για τις συνήθης ενέργειες σε μία βάση δεδομένων υπάρχουν τα στοιχεία *DelphiDBNavigator* και *DelphiDBGuiIO* τα οποία διευκολύνουν τις ενέργειες αυτές. Τα στοιχεία *ADOTable*, *ADOQuery* και *ADOCommand* αποτελούν

ειδικότερες περιπτώσεις για πράξεις στα στοιχεία μίας βάσης δεδομένων οπότε και κληρονομούν τις ιδιότητες του *DelphiDBAction* (Σχήμα 15).



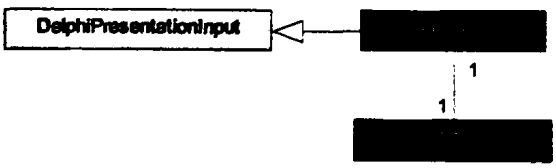
Σχήμα 15: ***Σχέσεις στοιχείων που σχετίζονται με το DelphiDBAction.***

Στο Σχήμα 16 περιγράφονται τα στοιχεία τα οποία κληρονομούν τις ιδιότητες του *DelphiDBGuiIO*.



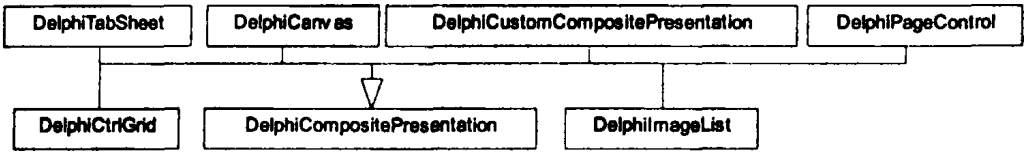
Σχήμα 16: ***Σχέσεις στοιχείων που σχετίζονται με το DelphiDBGuiIO.***

Κάθε ένα από τα στοιχεία του *DelphiDBGuiIO* κληρονομεί τις ιδιότητες του *DelphiPresentationInput* (Σχήμα 20) και μπορεί να συσχετιστεί με ένα *DelphiDBAction*. Για παράδειγμα το *DelphiDBEdit* θα εμφανίζεται στην γραφική διεπαφή του χρήστη με την μορφή ενός *edit field*. Η τιμή που θα έχει το *edit field* θα αλληλεπιδρά έμμεσα με κάποιο κελί κάποιας βάσης δεδομένων μέσω του παραπάνω *DelphiDBAction* (Σχήμα 17).

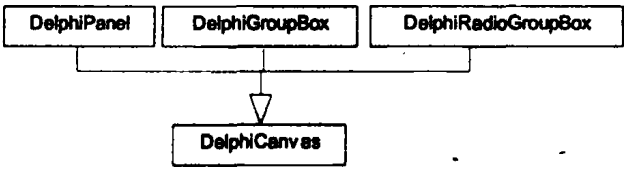


Σχήμα 17: ***Σχέσεις του στοιχείου DelphiGuiIO.***

Στοιχεία τα οποία ανήκουν στο *composite presentation* πακέτο κληρονομούν τις ιδιότητες του *DelphiCompositePresentation*. Όλα τα στοιχεία που περιγράφονται στα Σχήματα 18 και 19 χρησιμεύουν για την οργάνωση στοιχείων που ανήκουν στο *primitive presentation* πακέτο.

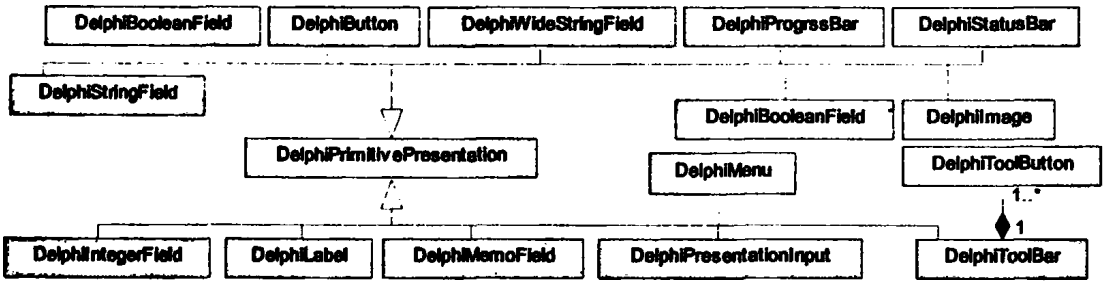


Σχήμα 18: ***Στοιχεία τα οποία κληρονομούν ιδιότητες του στοιχείου DelphiGuiIO.***



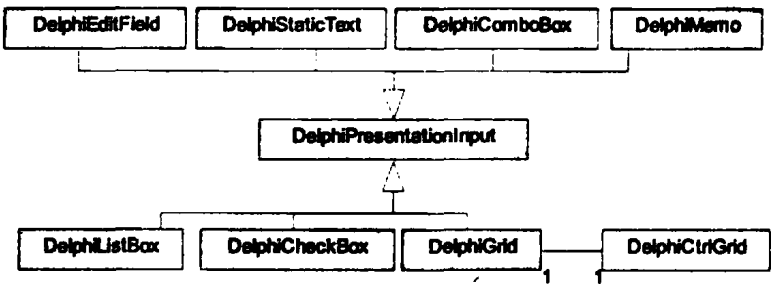
Σχήμα 19: ***Στοιχεία τα οποία κληρονομούν ιδιότητες του στοιχείου DelphiCanvas.***

Τα στοιχεία που σχετίζονται με την διεπαφή του χρήστη μπορούν να επικάθονται είτε απευθείας στην κύρια φόρμα του συστήματος, είτε σε κάποιου είδους καμβά από αυτούς που περιγράψαμε παραπάνω και ανήκει στο πακέτο *composite presentation*. Έτσι στο Σχήμα 20 εμφανίζονται πολλά στοιχεία της διεπαφής με τον χρήστη να κληρονομούν τις ιδιότητες του *PIPrimitivePresentation*. Ειδική περίπτωση αποτελεί το *DelphiToolBar* στο οποίο βλέπουμε να αποτελείται αυστηρά από ένα πλήθος στοιχείων *DelphiToolButton*. Η αυστηρή συσχέτιση οφείλεται στο γεγονός ότι για να τοποθετήσει κάποιος ένα *DelphiToolButton* θα πρέπει αναγκαία να υπάρχει και ένα *DelphiToolBar* στο οποίο θα αντιστοιχιστεί.



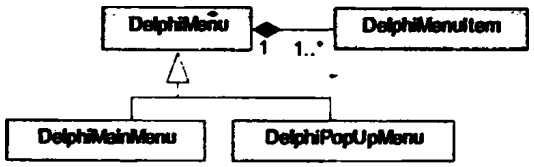
Σχήμα 20: ***Στοιχεία του primitive presentation πακέτου.***

Στο Σχήμα 20 υπάρχει ξεχωριστό στοιχείο το *DelphiPresentationInput* το οποίο το χρησιμοποιούμε για την αναπαράσταση στοιχείων των οποίων ο ρόλος είναι να εισάγουμε δεδομένα σε κάποια φόρμα. Στο Σχήμα 21 βλέπουμε στοιχεία τα οποία κληρονομούν τις ιδιότητες του *DelphiPresentationInput*.



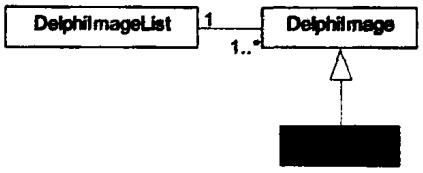
Σχήμα 21: ***Στοιχεία του primitive presentation πακέτου για την εισαγωγή δεδομένων σε μία φόρμα.***

Τα *menus* στην *Delphi* αποτελούνται από ένα πλήθος επιλογών οι οποίες μοντελοποιούνται με την μορφή των *DelphiMenuItem*. Τα *DelphiMainMenu* και *DelphiPopUpMenu*, που έχουν ως σκοπό την κατασκευή ενός *menu*, παρουσιάζουν κάποιες ιδιαιτερότητες. Έτσι κληρονομούν το *DelphiMenu* και κάνουν τις αντίστοιχες επεκτάσεις (Σχήμα 22).



Σχήμα 22: ***Σχέσεις στοιχείων που σχετίζονται με τα menus.***

Για την οργάνωση εικονιδίων στην *Delphi* πολλές φορές χρησιμοποιείται το *DelphiImageList* το οποίο περιέχει μία λίστα από *DelphiImage*. Εικόνες που προέρχονται από κάποια βάση δεδομένων τα αναπαριστάνουμε με το *DBImage* που κληρονομεί τις ιδιότητες του *DelphiImage* (Σχήμα 23).



Σχήμα 23: ***Συσχετίσεις ανάμεσα στα DelphiImage, DelpiImageList και DBImage.***

Πίνακας 2: *Περιγραφή των στοιχείων του Delphi PSR.*

Stereotype name	Base Class	Package	Parent	Events
DelphiBusiness	ECA:CCA::Process Component	Busin		
DelphiEvent	ECA:CCA::Process Component	Busin		
DelphiAction	ECA:CCA::Process Component	Busin		
DelphiAPI	ECA:CCA::Process Component	Busin	Delphi Business	
DelphiRegValues	ECA:CCA::Process Component	Busin	Delphi Business	
DelphiClient Socket	ECA:CCA::Process Component	Busin	Delphi Business	OnConnect, OnConnecting, OnDisconnect, OnError, OnLookUp, OnRead, OnWrite
DelphiServer Socket	ECA:CCA::Process Component	Busin	Delphi Business	OnAccept, OnClientConnect, OnClient Disconnect, OnClientError, OnClient Read, OnClientWrite, OnGetSocket, OnListen, OnThreadEnd, OnThreadStart
DelphiWebBrowser	ECA:CCA::Process Component	Busin	Delphi Component	
DelphiDataSource	ECA:CCA::Process Component	Busin		OnDataChange, OnStateChange, OnUpdateData
ADOConnection	ECA:CCA::Process Component	Busin	Delphi Business	AfterDisconnect, BeforeConnect, BeforeDisconnect, OnBeginTransComplete, OnCommitTransComplete, OnConnectComplete, OnDisconnect, OnExecuteComplete, OnInfoMessage, OnLogin, OnRollbackTransComplete, OnWillConnect, OnWillExecute
DelphiDBAction	ECA:Entity::Data Manager	Data		
DelphiDBGuiIO	ECA:CCA::Process Component	Data	Delphi Presentation Input	
ADOCommand	ECA:Entity::Data Manager	Data	DelphiDB Actions	
ADODataSet	ECA:CCA::Process Component	Data	DelphiData-Source	AfterCancel, AfterClose, AfterDelete, AfterEdit, AfterInsert, AfterOpen, AfterPost, AfterRefresh, AfterScroll, BeforeCancel, BeforeClose, BeforeDelete, BeforeEdit, BeforeInsert, BeforeOpen, BeforePost, BeforeRefresh, BeforeScroll, OnCalcFields, OnDeleteError, OnEndOfRecordset, OnFetchComplete, OnFetchProgress, OnFieldChangeComp, OnFilterRecord, OnMoveComplete, OnNewRecord, OnPostError, OnRecordChangeComplete, OnRecordsetChangeComplete, OnWillChangeField, OnWillChangeRecord, OnWillChangeRecordset, OnWillMove
ADOTable ADOQuery	ECA:Entity::Data Manager	Data	DelphiDB Actions	AfterCancel, AfterClose, AfterDelete, AfterEdit, AfterInsert, AfterOpen, AfterPost, AfterRefresh, AfterScroll, BeforeCancel, BeforeClose, BeforeDelete, BeforeEdit, BeforeInsert, BeforeOpen, BeforePost, BeforeRefresh, BeforeScroll, OnCalcFields, OnDeleteError, OnEndOfRecordset, OnFetchComplete, OnFetchProgress, OnFieldChangeComp, OnFilterRecord, OnMoveComplete, OnNewRecord, OnPostError, OnRecordChangeComplete, OnRecordsetChangeComplete, OnWillChangeField, OnWillChangeRecord, OnWillChangeRecordset, OnWillMove
DelphiDBNavigator	ECA:Entity::Data Manager	Data		OnClick, OnContextPopup, OnDblClick, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnResize, OnStartDock, OnStartDrag

DelphiDBImage	ECA:CCA::Process Component	Data	DelphiImage	OnClick, OnContextPopup, OnDblClick, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnProgress, OnStartDock, OnStartDrag
DelphiDBMemo	ECA:CCA::Process Component	Data	DelphiDB GuiIO	
DelphiDBField	ECA:CCA::Process Component	Data	DelphiDB GuiIO	
DelphiDBText	ECA:CCA::Process Component	Data	DelphiDB GuiIO	
DelphiDBCheckBox	ECA:CCA::Process Component	Data	DelphiDB GuiIO	
DelphiDBGrid	ECA:CCA::Process Component	Data	DelphiDB GuiIO	OnCellClick, OnCellEnter, OnCellExit, OnColumnMoved, OnDblClick, OnDragDrop, OnDragOver, OnDrawColumnCell, OnDrawDataCell, OnEditButtonClick, OnEndDock, OnEndDrag, OnEnter, OnExit, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnStartDock, OnStartDrag, OnTitleClick
DelphiDBLookUpBox	ECA:CCA::Process Component	Data	DelphiDB GuiIO	OnClick, OnContextPopup, OnDblClick, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnKeyDown, OnKeyPress, OnKeyUp, OnMeasureItem, OnMouseDown, OnMouseMove, OnMouseUp, OnStartDock, OnStartDrag
DelphiDBLookup ComboBox	ECA:CCA::Process Component	Data	DelphiDB GuiIO	
DelphiComposite Presentation	ECA:CCA::Process Component	CPres		
DelphiCustom Composite Presentation	ECA:CCA::Process Component	CPres	Delphi Composite Presentation	
DelphiImageList	ECA:CCA::Process Component	GPres	Delphi Composite Presentation	OnClick
DelphiPageControl	ECA:CCA::Process Component	CPres	Delphi Composite Presentation	OnChange, OnChanging, OnContextPopup, OnDockDrop, OnDockOver, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnGetImageIndex, OnGetSiteInfo, OnMouseDown, OnMouseMove, OnMouseUp, OnResize, OnStartDock, OnStartDrag, OnUnDock
DelphiCtrlGrid	ECA:CCA::Process Component	CPres	Delphi Composite Presentation	OnClick, OnDblClick, OnDragDrop, OnDragOver, OnEndDrag, OnEnter, OnExit, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnPaintPanel, OnStartDrag
DelphiTabSheet	ECA:CCA::Process Component	CPres	Delphi Composite Presentation	
DelphiCanvas	ECA:CCA::Process Component	CPres	Delphi Composite Presentation	
DelphiPanel	ECA:CCA::Process Component	CPres	Delphi Canvas	OnCanResize, OnClick, OnConstrainedResize, OnContextPopup, OnDblClick, OnDockDrop, OnDockOver, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnGetSiteInfo, OnMouseDown, OnMouseMove, OnMouseUp, OnResize, OnStartDock, OnStartDrag, OnUnDock

DelphiGroupBox	ECA:CCA::Process Component	CPres	Delphi Canvas	OnClick, OnContextPopup, OnDblClick, OnDockDrop, OnDockOver, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnGetSiteInfo, OnMouseDown, OnMouseMove, OnMouseUp, OnStartDock, OnStartDrag, OnUnDock
DelphiRadioGroup	ECA:CCA::Process Component	CPres	Delphi Canvas	OnClick, OnContextPopup, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnStartDock, OnStartDrag
DelphiForm	ECA:CCA::Process Component	CPres		OnActivate, OnCanResize, OnClick, OnClose, OnCloseQuery, OnConstrainedResize, OnContextPopup, OnCreate, OnDblClick, OnDeactivate, OnDestroy, OnDockDrop, OnDockOver, OnDragDrop, OnDragOver, OnEndDock, OnGetSiteInfo, OnHelp, OnHide, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnMouseWheel, OnMouseWheelDown, OnMouseWheelUp, OnPaint, OnResize, OnShortCut, OnShow, OnStartDock, OnUnDock
DelphiPrimitive Presentation	ECA:CCA::Process Component	PPres		
DelphiPresentation Input	ECA:CCA::Process Component	PPres	Delphi Primitive Presentation	
DelphiMenuItem	ECA:CCA::Process Component	PPres		OnAdvancedDrawItem, OnClick, OnDrawItem, OnMeasureItem
DelphiMenu	ECA:CCA::Process Component	PPres	Delphi Primitive Presentation	
DelphiMainMenu	ECA:CCA::Process Component	PPres	Delphi Menu	OnChange
DelphiPopUpMenu	ECA:CCA::Process Component	PPres	Delphi Menu	OnChange, OnPopup
DelphiStatusBar	ECA:CCA::Process Component	PPres	Delphi Primitive Presentation	OnClick, OnContextPopup, OnDblClick, OnDragDrop, OnDragOver, OnDrawPanel, OnEndDock, OnEndDrag, OnHint, OnMouseDown, OnMouseMove, OnMouseUp, OnResize, OnStartDock, OnStartDrag
DelphiToolBar	ECA:CCA::Process Component	PPres	Delphi Primitive Presentation	
DelphiToolButton	ECA:CCA::Process Component	PPres		OnClick, OnContextPopup , OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnMouseDown, OnMouseMove, OnMouseUp, OnStartDock, OnStartDrag
DelphiImage	ECA:CCA::Process Component	PPres	Delphi Primitive Presentation	OnClick, OnContextPopup, OnDblClick, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnMouseDown, OnMouseMove, OnMouseUp, OnProgress, OnStartDock, OnStartDrag
DelphiMemo	ECA:CCA::Process Component	PPres	Delphi Primitive Input	OnChange, OnClick, OnContextPopup, OnDblClick, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnStartDock, OnStartDrag
DelphiEditField	ECA:CCA::Process Component	PPres	Delphi Primitive Input	OnChange, OnClick, OnContextPopup, OnDblClick, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnStartDock, OnStartDrag
DelphiStaticText	ECA:CCA::Process Component	PPres	Delphi Primitive Input	OnClick, OnContextPopup, OnDblClick, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnMouseDown, OnMousemove, OnMouseUp, OnStartDock, OnStartDrag
DelphiCheckBox	ECA:CCA::Process Component	PPres	Delphi Primitive Input	OnClick, OnContextPopup, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, On-Enter, OnExit, OnKeyDown, On-KeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnStart-Dock, OnStartDrag

DelphiGrid DelphiStringGrid DelphiDrawGrid	ECA:CCA::Process Component	PPres	Delphi Primitive Input	OnClick, OnColumnMoved, OnContextPopup, OnDblClick, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnGetEditMask, OnGetEditText, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnMouseWheelDown, OnMouseWheelUp, OnRowMoved, OnSelectCell, OnSetEditText, OnStartDock, OnStartDrag, OnTopLeftChanged
DelphiListBox	ECA:CCA::Process Component	PPres	Delphi Primitive Input	OnClick, OnContextPopup, OnDblClick, OnDragDrop, OnDragOver, OnDrawItem, OnEndDock, OnEndDrag, OnEnter, OnExit, OnKeyDown, OnKeyPress, OnKeyUp, OnMeasureItem, OnMouseDown, OnMouseMove, OnMouseUp, OnStartDock, OnStartDrag
DelphiComboBox	ECA:CCA::Process Component	PPres	Delphi Primitive Input	OnChange, OnClick, OnContextPopup, OnDblClick, OnDragDrop, OnDrawItem, OnDropDown, OnEndDock, OnEndDrag, OnEnter, OnExit, OnKeyDown, OnKeyPress, OnKeyUp, OnMeasureMent, OnStartDock, OnStartDrag
DelphiLabel	ECA:CCA::Process Component	PPres	Delphi Primitive Presentation	OnClick, OnContextPopup, OnDblClick, OnDragDrop, OnEndDock, OnEndDrag, OnMouseDown, OnMouseMove, OnMouseUp, OnStartDock, OnStartDrag
DelphiButton	ECA:CCA::Process Component	PPres	Delphi Primitive Presentation	OnClick, OnContextPopup, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnStartDock, OnStartDrag
DelphiMemoField	ECA:CCA::Process Component	PPres	Delphi Primitive Presentation	OnChange, OnClick, OnContextPopup, OnDblClick, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnStartDock, OnStartDrag
DelphiIntegerField DelphiFloatField DelphiWide-StringField DelphiStringField DelphiBooleanField	ECA:CCA::Entity:: EntityData	PPres	Delphi Primitive Presentation	

3.4 Αναλυτική περιγραφή των αναπαραστάσεων που έχουν σχέση με το Web

Για την φάση της πρόσθιας κατασκευής της εφαρμογής μας επιλέξαμε την κατασκευή κάποιων τμημάτων σε πλατφόρμες που παράγουν δυναμικά ιστοσελίδες με χρήση *php*. Τα μοντέλα που απαρτίζουν τη *web* αναπαράσταση είναι στοιχεία τα οποία αρχίζουν με το χαρακτηριστικό *php*, όταν τα στοιχεία υλοποιούνται από τεχνικής άποψης με *php*. Αντίστοιχα ισχύουν για τα στοιχεία που υλοποιούνται με *HTML* και *JavaScript* (*JS*).

Stereotypes που ανήκουν στο πακέτο *Business*

- *JSEvent*: Τα *JSEvent* είναι τα *events* τα οποία είναι διαθέσιμα από την *JavaScript*. Αυτά χωρίζονται σε τρεις κατηγορίες. Αυτά που προέρχονται από *event* που προκλήθηκε με τη χρήση του:

- ποντικιού (*mouse*)

  OnMouseDown Ένα κουμπί του ποντικιού έχει πατηθεί.

  OnMouseMove Το ποντίκι μετακινήθηκε.

  OnMouseOut Το ποντίκι έχει εγκαταλείψει την περιοχή ενός στοιχεία.

  OnMouseOver Το ποντίκι έχει εισέλθει στην περιοχή ενός στοιχεία.

  OnMouseUp Ένα κουμπί του ποντικιού έχει αφεθεί.

  OnClick Ένα κουμπί του ποντικιού έχει πατηθεί.

  OndblClick Ένα κουμπί του ποντικιού έχει πατηθεί γρήγορα δύο φορές.

- πληκτρολογίου

  Onkeydown Ένα κουμπί του πληκτρολογίου έχει πατηθεί.

  Onkeyup Ένα κουμπί του πληκτρολογίου έχει αφεθεί.

  Onkeypress Ένα *Onkeydown* ακολουθούμενο από ένα *Onkeyup*.

- γενικότερου είδους *events*.

  Onblur Ένα στοιχείο χάνει το "*focus*".

  Onerror Ένα σφάλμα λαβαίνει χώρα

  Onfocus Ένα στοιχείο αποκτά το "*focus*".

  Onload Το κείμενο (ιστοσελίδα) έχει φορτωθεί πλήρως.

  Onreset Έχει προκληθεί η εντολή *reset* μίας φόρμας.

  Onscroll Το κείμενο ξετυλίγεται (*scrolled*).

  Onselect Η επιλογή ενός στοιχείου έχει διαφοροποιηθεί.

  Onsubmit Έχει προκληθεί η εντολή *submit* μίας φόρμας.

Για τον ορισμό του *stereotype* ως βασική κλάση έχει επιλεγεί η *ECA:CCA::ProcessComponent*.

- *PhPActivity*: Τα *PhPActivity* αναπαριστούν τον πηγαίο κώδικα που έχει προέλθει από έναν προγραμματιστή ή είναι το σημείο στο οποίο θα πρέπει να επέμβει ο προγραμματιστής. Για τον ορισμό του *PlActivity* σαν βασική κλάση έχει επιλεγεί η *ECA:CCA::Process-Component*.
- *PhPConnector*: Ο *PhPConnector* χρησιμοποιείται στην μοντελοποίηση για να συνδέσει ετερογενή στοιχεία. Το κύριο χαρακτηριστικό του είναι ότι μέσω αυτού γίνεται

μεταβίβαση πληροφορίας από ή προς ένα αντικείμενο. Για τον ορισμό του *PhPConnector* ως βασική κλάση έχει επιλεγεί η *ECA:CCA::ProcessComponent.*

- *PhPDBConnection*: Τα *PhPDBConnection* είναι στοιχεία τα οποία έχουν ως πατέρα το *PhPConnector* και βασική κλάση την *ECA:CCA::ProcessComponent.* Σχετίζονται με την επικοινωνία και μεταβίβαση πληροφορίας μίας βάσης δεδομένων.
- *PhPDataSource*: Τα δεδομένα που επιστρέφονται συνήθως από το αποτέλεσμα μίας ενέργειας (με τη χρήση ενός *PhPConnector*) παρουσιάζονται στο μοντέλο μας με την μορφή ενός στοιχείου *PhPDataSource.* Για τον ορισμό του *PhPDataSource* ως βασική κλάση έχει επιλεγεί η *ECA:CCA::ProcessComponent.*
- *PhPComponent*: Στην *PhP* υπάρχει μεγάλο πλήθος από έτοιμα προγράμματα τα οποία είτε δίδονται με τη μορφή βιβλιοθήκης είτε χρησιμοποιούνται ως εξωτερικά προγράμματα (π.χ. έτοιμα *dlls*). Γα αντικείμενα (στην υλοποίηση) που βασίζονται στα παραπάνω εξωτερικά στοιχεία στη μοντελοποίηση τα αντιστοιχούμε σε *PhPComponent.* Για τον ορισμό του *stereotype* αυτού ως βασική κλάση έχουμε επιλέξει την *ECA:CCA::ProcessComponent* ενώ ως πατέρας το *PhPConnector.*

Stereotypes* που ανήκουν στο πακέτο *Data

- *PhPDBActivity*: Για εργασίες που σχετίζονται με την αλληλεπίδραση με μία βάση δεδομένων έχει οριστεί το *PhPDBActivity.* Επειδή έχει να κάνει με τον χειρισμό δεδομένων ως βασική κλάση έχει επιλεγεί η *ECA:Entity::DataManager.* Το *stereotype* αυτό είναι ο πατέρας για πιο συγκεκριμένες ενέργειες που παρουσιάζουμε.
- *PhPDBQuery, PhPDBTable*: Για ενέργειες που αφορούν την αναζήτηση δεδομένων από μία βάση δεδομένων έχουν οριστεί οι *PhPDBQuery* και *PhPDBTable.* Η *PhPDBTable* άμεσα με την χρήση ενός ολόκληρου πίνακα από μία βάση δεδομένων ενώ η *PhPDBQuery* είναι πιο ευέλικτη και ζητάει στήλες από διάφορους πίνακες μίας βάσης δεδομένων. Για τους ορισμούς των *stereotypes* τους ως βασική κλάση έχει οριστεί η *ECA:Entity::DataManager* και πατέρας τα *PhPDBActivity.*
- *PhPDBRawCommand*: Η *PhPDBRawCommand* είναι ένα στοιχείο το οποίο χρησιμοποιείται επίσης για την επικοινωνία με μία βάση δεδομένων. Η διαφορά είναι ότι οι εντολές που πραγματοποιούνται προς την βάση δεδομένων είναι οποιασδήποτε φύσης. Παράδειγμα η αποστολή εντολών που αφορούν την ρύθμιση κλειδιών σε μία σχεσιακή

βάση δεδομένων με την χρήση *raw SQL* εντολών. Για τον ορισμό ως πατέρας του *stereotype* ορίζεται το *PhPDBActivity* και βασική κλάση η *ECA:Entity::DataManager*.

- *PhPDBElement*: Το *PhPDBElement* αντιπροσωπεύει μία στήλη σε κάποια βάση δεδομένων. Όπως και στην περίπτωση του *PIEntityData* ως βασική κλάση έχουμε την *ECA:Entity::EntityData* η οποία ορίζεται για τον σκοπό αυτό.

Stereotypes* που ανήκουν στο πακέτο *Composite Presentation

- *PhPCompositePresentation*: Στις φόρμες από τις οποίες αποτελείται ένα σύστημα που λειτουργεί στο *web* υπάρχουν κάποια στοιχεία τα οποία είναι υπεύθυνα για την οργάνωση της διεπαφής με τον χρήστη και κατά κάποιο τρόπο διενεργούνται στο παρασκήνιο. Σαν βασική κλάση αυτών έχει επιλεγεί η *ECA:CCA::ProcessComponent*.
- *HTMLForm*: Στοιχεία της διεπαφής με τον χρήστη (*HTMLInput* π.χ. κουμπιά, *radio boxes, checkbox* κτλ) τα οποία χρησιμοποιούμε για την συνήθη δημιουργία μίας φόρμας είναι οργανωμένα σε *HTMLForm*. Η *HTML* επιβάλει την οργάνωση σε φόρμες οι οποίες ενεργοποιούνται με την βοήθεια κουμπιών που ανήκουν σε αυτή. Σαν βασική κλάση για τον ορισμό αυτού του *stereotype* είναι το *ECA:CCA::ProcessComponent*.
- *HTMLFrame*: Μία φόρμα μπορεί να διασπαστεί σε μικρότερες υποφόρμες που διαχωρίζονται με βάση την θέση που έχουν στην διεπαφή. Για να δείξουμε ότι υπάρχει τέτοια περίπτωση στο μοντέλο μας εισάγουμε το *stereotype HTMLFrame*. Το συγκεκριμένο στοιχείο στις παραμέτρους του εκτός από τα γνωρίσματα που αφορούν την διεπαφή, π.χ. θέση και διαστάσεις του πλαισίου, θα έχει και μία σαφή αναφορά για την θέση από όπου θα γίνει ανάγνωση της νέας φόρμας.
- *PhPCustomCompositePresentation*: Το στοιχείο *PhPCustomCompositePresentation* υπάρχει για να δώσουμε την δυνατότητα στους σχεδιαστές να προσθέσουν δικά τους στοιχεία τα οποία εκτελούν κάποιες ιδιαίτερες λειτουργίες με την χρήση *php*. Κατά την πρόσθια κατασκευή πολλά από τα *refinement patterns* οδηγούν σε στοιχεία *PhPCustomCompositePresentation*. Αυτό οφείλεται στις περιορισμένες δυνατότητες που έχουν τα βασικά στοιχεία της παρούσας τεχνολογικής πλατφόρμας (*php* και *hmtl*). Πατέρας αυτού του *stereotype* είναι το *PhPCompositePresentation* ενώ βασική κλάση ορίζεται το *ECA:CCA::Process Component*.

Stereotypes* που ανήκουν στο πακέτο *Primitive Presentation

- *PhPPrimitivePresentation*: Είναι το κύριο στοιχείο το οποίο ανήκει στο *primitive presentation* πακέτο. Είναι ένα αντικείμενο το οποίο εμφανίζεται στην διεπαφή με τον χρήστη. Το *stereotype* για τον ορισμό του έχει ως βασική κλάση την *ECA:CCA::Process Component.*
- *HTMLText*: Στοιχεία της διεπαφής με τον χρήστη που εμφανίζονται με την μορφή κειμένου, χωρίς να δύνεται η δυνατότητα να το επεξεργαστεί ο χρήστης, αναπαριστώνται με το *stereotype HTMLText.* Το *stereotype* για τον ορισμό του έχει ως βασική κλάση την *ECA:CCA::ProcessComponent* και πατέρα το *PhPPrimitivePresentation.*
- *HTMLMenus, HTMLNewForm, HTMLCustomMenu, HTMLMenuItem*: Στις εφαρμογές πολύ συχνά παρουσιάζονται *menus* τα οποία περιέχουν συντομεύσεις (κουμπιά και *links* στην *html*). Από πρακτικής άποψης στην *html* δεν έχουμε κάποιο αντικείμενο το οποίο να ομαδοποιεί συντομεύσεις. Αντίθετα, τα στοιχεία τα οποία τις αντιπροσωπεύουν επικάθονται απ' ευθείας στην κύρια φόρμα της ιστοσελίδας ή σε κάποιο καμβά. Στην μοντελοποίηση και συγκεκριμένα όταν το *PSM* είναι αποτέλεσμα της πρόσθιας κατασκευής κάποια *refinement patterns* σχετίζονται με την απεικόνιση *menus* (στο *PIR* υπάρχουν τέτοιες αναπαραστάσεις) είναι χρήσιμο, για τον προγραμματιστή, να υπάρχουν στοιχεία τα οποία καλύπτουν την περίπτωση αυτή. Έτσι για τα *menus* ορίζεται το *stereotype HTMLMenus* και κάθε συντόμευση αυτού αντιστοιχίζεται στο *HTMLMenuItem.* Όταν αυτά τα *menus* έχουν κάποια ειδική μορφή τότε αντιστοιχίζονται σε στοιχεία *HTMLCustomMenu.* Ειδική περίπτωση αποτελεί η χρήση ενός ξεχωριστού παραθύρου (έμμεσα δημιουργείται μία νέα φόρμα) για την αναπαράσταση του *menu* το οποίο το μοντελοποιούμε με το *HTMLNewForm.* Για όλα τα παραπάνω *stereotypes* ως βασική κλάση έχει ορισθεί η *ECA:CCA::ProcessComponent.* Πατέρας των *HTMLNewForm* και *HTMLCustomMenu* είναι το *HTMLMenus.*
- *HTMLImage*: Τα στοιχεία που εμφανίζονται στην διεπαφή χρήστη με τη μορφή μίας εικόνας τα αναπαριστούμε με το *stereotype HTMLImage.* Βασική κλάση για αυτό έχει επιλεγεί το *ECA:CCA::ProcessComponent* και ως πατέρας ορίζεται το *PhpPrimitive-Presentation.*
- *PhPTable*: Πίνακας του οποίου τα δεδομένα έχουν προστεθεί με την βοήθεια της *php.* Δηλαδή γίνεται κάποια εργασία στο παρασκήνιο που αφορά την αναζήτηση δεδομένων από κάποια πηγή και τα εμφανίζουμε στη διεπαφή του χρήστη με τη βοήθεια του

στοιχείου *PhPTable*. Βασική κλάση για το *stereotype* αυτό έχει οριστεί το *ECA:CCA::ProcessComponent*.

- *HTMLInput*: Οι φόρμες στην *html* αποτελούνται από στοιχεία *HTMLInput* για τη διαβίβαση παραμέτρων στην κάθε περίπτωση. Η μορφή των παραμέτρων αυτών ποικίλει. Ο διαχωρισμός τους γίνεται με βάση ένα *attribute* στο οποίο έχουμε προσδώσει το όνομα "*type*". Έτσι όταν η τιμή της παραμέτρου αυτή είναι ίση με:
  - *Text*: Παράγεται πλαίσιο στο οποίο ο χρήστης μπορεί να εισάγει ως το πολύ *maxlength* χαρακτήρες. Η παράμετρος *size* εκφράζει το μέγεθος των χαρακτήρων που φαίνονται στο πλαίσιο, στην περίπτωση που ο χρήστης εισάγει περισσότερους χαρακτήρες τότε γίνεται "*scroll*".
  - *Password*: Το ίδιο με την περίπτωση *type=text*, η διαφορά είναι ότι το κείμενο θα αποκρύπτεται με "*". Στη φόρμα βέβαια τα στοιχεία αποστέλλονται κανονικά και όχι ως "*".
  - *Checkbox*: Παράγει ένα *checkbox*. Έχει δύο καταστάσεις *on* και *off*. Όταν είναι στη κατάσταση *on* και η *form* γίνει *submit* τότε θα σταλθεί ως "*name=on*", αλλιώς θα αγνοηθεί.
  - *Radio*: Παράγει ένα *radio button*. Τα *radio button* πάντα υπάρχουν σε μία ομάδα. Όλα τα μέλη της ίδιας ομάδας θα πρέπει να έχουν τη ίδια τιμή στην παράμετρο *name* και διαφορετική στις *values*. Μόνο η παράμετρος *value* του επιλεγμένου *radio button* θα αποσταλεί κατά το *submit*.
  - *Submit*: Παράγει ένα κουμπί που όταν πατηθεί στέλνει όλα τα δεδομένα της φόρμας στον *server*. Η κάθε φόρμα μπορεί να διαθέτει περισσότερα από ένα κουμπιά. Το κάθε ένα από αυτά θα πρέπει να έχει διαφορετικό *name*. Επίσης το *name* και το *value* του κουμπιού που θα πατηθεί στέλνονται στον *server*.
  - *Reset*: Παράγει επίσης ένα κουμπί το οποίο θα επαναφέρει τις αρχικές τιμές στην φόρμα όταν αυτό πατηθεί.
  - *File*: Χρησιμοποιείται για το *upload* ενός αρχείου. Στην διεπαφή με τον χρήστη παρουσιάζεται με την μορφή ενός *input box* το οποίο με την βοήθεια ενός κουμπιού γίνεται το *browsing* στον τοπικό σκληρό δίσκο. Με την μέθοδο αυτή ο χρήστης μπορεί να καθορίσει ένα ή περισσότερα αρχεία.

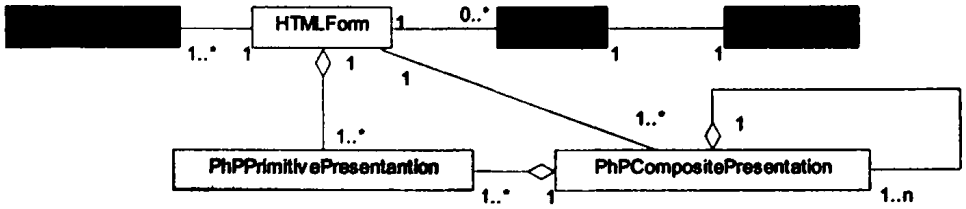
- *Hidden*: Επιτρέπει να προσθέσουμε στην φόρμα κρυφή πληροφορία η οποία δεν επιθυμούμε να υποστεί κάποια διαμόρφωση. Χρησιμοποιείται όταν θέλουμε να αποθηκεύσουμε με κάποιο τρόπο πληροφορία σχετικά με την κατάσταση της φόρμας.
- *Image*: Λειτουργεί ακριβώς όπως η περίπτωση του *submit* αλλά χρησιμοποιείται μία εικόνα για την αναπαράσταση του κουμπιού.

Για τον ορισμό του *stereotype* βασική κλάση είναι η *ECA:CCA::ProcesComponent* και πατέρας ο *PhPPrimitivePresentation.*

- *PhPCustomPrimitivePresentation*: Στην περίπτωση που κάποιο αντικείμενο δεν πληροί τις προδιαγραφές κάποιου από τα παραπάνω στοιχεία και θα πρέπει να ανήκει στο *primitive presentation* πακέτο, τότε αυτό μπορεί να απεικονιστεί με τη χρήση του *PhPCustom-PrimitivePresentation.* Για τον ορισμό του *stereotype* σαν βασική κλάση έχουμε το *ECA:CCA::ProcessComponent* και πατέρα το *PhPPrimitivePresentation.*

Metamodels του *PSR*

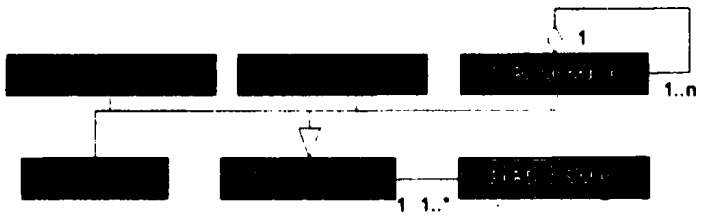
Μέχρι στιγμής έχουμε παρουσιάσει μία συλλογή από *stereotypes* χωρισμένα σε διάφορα πακέτα. Στη συνέχεια παρουσιάζουμε πώς πολλά από αυτά αλληλεξαρτώνται με τη βοήθεια *metamodels.* Οι εφαρμογές στο *web* αποτελούνται από ένα πλήθος ιστοσελίδων, είτε αυτές παράγονται δυναμικά είτε στατικές.



Σχήμα 24: ***Βασική αναπαράσταση του PSR.***

Όπως και στις προηγούμενες αναπαραστάσεις που δόθηκαν, έτσι και εδώ υπάρχει ένα βασικό στοιχείο το *HTMLForm* στο οποίο επικάθονται όλα τα υπόλοιπα στοιχεία και το σύνολό τους σχηματίζει το *BIS* (Σχήμα 24). Στο θεμέλιο στοιχείο *HTMLForm* της κάθε ιστοσελίδας προσαρτώνται τα παρακάτω στοιχεία:

- Ένα ή περισσότερα *JSEvents* τα οποία όταν γίνονται *triggered* προκαλούν την εκκίνηση ενός *PhPActivity*.
- Ένα ή περισσότερα *PhPConnector* το οποίο χρησιμοποιείται για να μοντελοποιήσει στοιχεία που διενεργούνται στο παρασκήνιο. Το *PhPConnector* θα αποτελέσει τον πατέρα για αρκετά στοιχεία που ανήκουν στο *business* πακέτο. Έτσι στο Σχήμα 25 βλέπουμε τα στοιχεία *PhPDBConnection*, *PhPDataSource*, *PhPCustom* και *PhPComponent* να κληρονομούν τις ιδιότητες του *PhPConnector*.



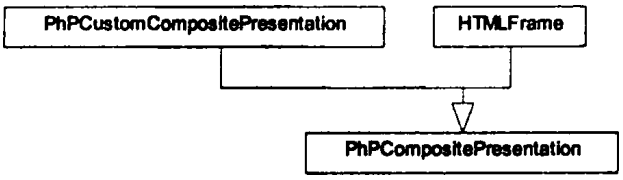
Σχήμα 25: ***Στοιχεία που ανήκουν στο business πακέτο.***

Στο Σχήμα 25 υπάρχει το στοιχείο *PhPDBConnection* το οποίο χρησιμοποιείται ως συνδετικός κρίκος ανάμεσα σε μία βάση δεδομένων και στοιχεία *PhPDBActivity* (Σχήμα 26). Το κάθε *PhPDBActivity* μπορεί και αντιστοιχίζεται σε ένα ή περισσότερα *PhPDBConnection* και αντίστροφα. Το *PhPDBActivity* κληρονομεί τις ιδιότητες του *PhPActivity*. Όταν οι ενέργειες αφορούν ένα ή περισσότερα πεδία της βάσης δεδομένων τότε αυτά μοντελοποιούνται με τη βοήθεια του *PhPDBElement*. Τα στοιχεία *PhPDBTable*, *PhPDBQuery* και *PhPDBRawCommand* κληρονομούν τα χαρακτηριστικά του *PhPDBActivity*.

Σχήμα 26: ***Στοιχεία που ανήκουν στο business πακέτο.***

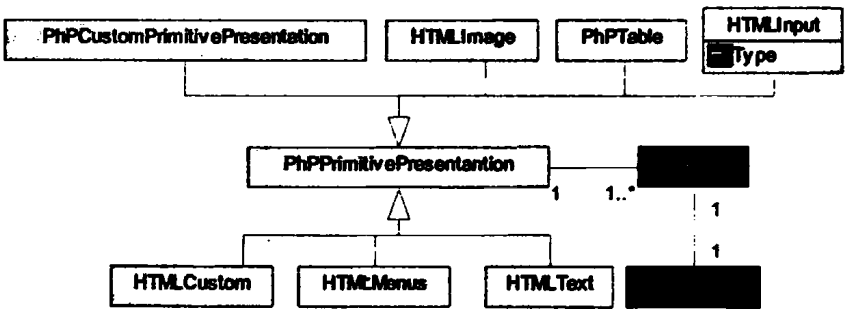
- Ένα ή περισσότερα *PhPCompositePresentation* τα οποία χρησιμοποιούνται για στοιχεία τα οποία ανήκουν στο *composite presentation*. Επίσης μπορούν στοιχεία *PhPCompositePresentation* να προσαρτώνται σε άλλα *PhPCompositePresentation*. Στο Σχήμα

27 βλέπουμε τα στοιχεία *PhPCustomCompositePresentation* και *HTMLFrame* τα οποία κληρονομούν τις ιδιότητες του *PhPCompositePresentation.*



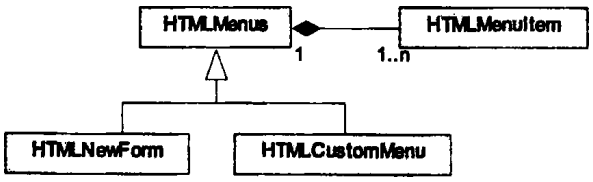
Σχήμα 27: ***Σχέσεις στοιχείων τα οποία ανήκουν στο composite presentation.***

- Για την συμπλήρωση του *PSR* έχει απομείνει να περιγράψουμε την διεπαφή του χρήστη με το σύστημα. Τα στοιχεία αυτά εμφανίζονται στο Σχήμα 24 με την μορφή του *PhPPrimitivePresentation.* Στο Σχήμα 28 εμφανίζονται στοιχεία που κληρονομούν το *PhPPrimitivePresentation.* Το κάθε στοιχείο της διεπαφής σχετίζεται με ένα ή περισσότερα *JSEvent.*



Σχήμα 28: ***Σχέσεις στοιχείων τα οποία ανήκουν στο primitive presentation.***

Στο Σχήμα 28 το στοιχείο *HTMLMenus* σχετίζεται αυστηρά με ένα ή περισσότερα *HTMLMenuItem.* Στις ειδικές περιπτώσεις που το *menu* έχει ειδική μορφή τότε μοντελοποιείται με βάση τα *HMTLNewForm* και *HTMLCustomMenu* τα οποία κληρονομούν τις ιδιότητες του *HTMLMenus* (Σχήμα 29).



Σχήμα 29: ***Σχέσεις στοιχείων που σχετίζονται με menus.***

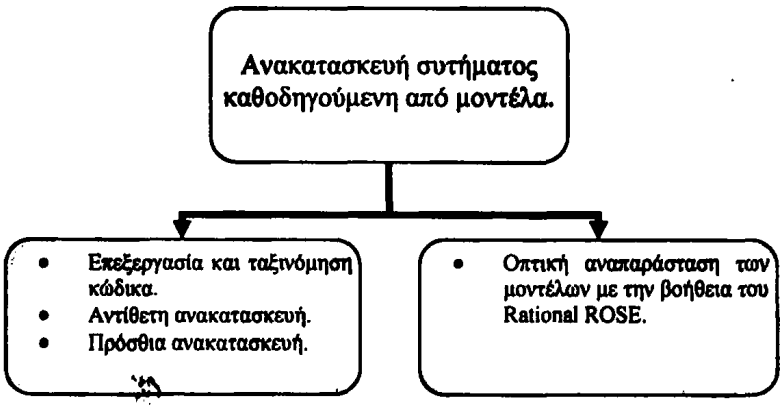
Πίνακας 3: *Περιγραφή των στοιχεία του PhP PSR.*

Stereotype name	Base Class	Package	Parent	Events
JSEvent	ECA:CCA::Process Component	Busin		OnMouseDown, OnMouseMove, OnMouseOut, OnMouseOver, OnMouseUp, OnClick, OndblClick, OnKeyDown, OnKeyUp, OnKeyPress, Onblur, Onerror, Onfocus, Onload, Onreset, Onscroll, OnSelect, OnSubmit
PhPActivity	ECA:CCA::Process Component	Busin		
PhPConnector	ECA:CCA::Process Component	Busin		OnActivate, OnCanResize, OnClick, OnClose, OnCloseQuery, OnConstrainedResize, OnContextPopup, OnCreate, OnDblClick, OnDeactivate, OnDestroy, OnDockDrop, OnDockOver, OnDragDrop, OnDragOver, OnEndDock, OnGetSiteInfo, OnHelp, OnHide, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnMouseWheel, OnMouseWheelDown, OnMouseWheelUp, OnPaint, OnResize, OnShortCut, OnShow, OnStartDock, OnUnDock
PhPDBConnection	ECA:CCA::Process Component	Busin		AfterDisconnect, BeforeConnect, BeforeDisconnect, OnBeginTransComplete, OnCommitTransComplete, OnConnectComplete, OnDisconnect, OnExecuteComplete, OnInfoMessage, OnLogin, OnRollbackTransComplete, OnWillConnect, OnWillExecute
PhPDataSource	ECA:CCA::Process Component	Busin		OnDataChange, OnStateChange, OnUpdateData
PhPComponent	ECA:CCA::Process Component	Busin	PhPConnectot	
PhPDBRawCommand	ECA:CCA::Process Component	Data		
PhPDBQuery PhPDBTable	ECA:Entity::Data Manager	Data	PhPDBActivity	
PhPDBActivity	ECA:Entity::Data Manager	Data		AfterCancel, AfterClose, AfterDelete, AfterEdit, AfterInsert, AfterOpen, AfterPost, AfterRefresh, AfterScroll, BeforeCancel, BeforeClose, BeforeDelete, BeforeEdit, BeforeInsert, BeforeOpen, BeforePost, BeforeRefresh, BeforeScroll, OnCalcFields, OnDeleteError, OnEndOfRecordset, OnFetchComplete, OnFetchProgress, OnFieldChangeComp, OnFilterRecord, OnMoveComplete, OnNewRecord, OnPostError, OnRecordChangeComplete, OnRecordsetChangeComplete, OnWillChangeField, OnWillChangeRecord, OnWillChangeRecordset, OnWillMove
PhPDBElement	ECA:CCA::Entity:: EntityData	Data		
PhPComposite Presentation	ECA:CCA::Process Component	CPres		
HTMLForm	ECA:CCA::Process Component	CPres	**PhPComposite Presentation**	OnActivate, OnCanResize, OnClick, OnClose, OnCloseQuery, OnConstrainedResize, OnContextPopup, OnCreate, OnDblClick, OnDeactivate, OnDestroy, OnDockDrop, OnDockOver, OnDragDrop, OnDragOver, OnEndDock, OnGetSiteInfo, OnHelp, OnHide, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnMouseWheel, OnMouseWheelDown, OnMouseWheelUp, OnPaint, OnResize, OnShortCut, OnShow, OnStartDock, OnUnDock
PhPCustomComposite Presentation	ECA:CCA::Process Component	CPres	PhPComposite Presentation	OnActivate, OnCanResize, OnClick, OnClose, OnCloseQuery, OnConstrainedResize, OnContextPopup, OnCreate, OnDblClick, OnDeactivate, OnDestroy, OnDockDrop, OnDockOver, OnDragDrop, OnDragOver, OnEndDock, OnGetSiteInfo, OnHelp, OnHide, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnMouseWheel, OnMouseWheelDown, OnMouseWheelUp, OnPaint, OnResize, OnShortCut, OnShow, OnStartDock, OnUnDock

PhPPrimitive Presentation	ECA:CCA::Process Component	PPres		
PhPCustomPrimitive Presentation	ECA:CCA::Process Component	PPres	PhPPrimitive Presentation	OnActivate, OnCanResize, OnClick, OnClose, OnCloseQuery, OnConstrainedResize, OnContextPopup, OnCreate, OnDblClick, OnDeactivate, OnDestroy, OnDockDrop, OnDockOver, OnDragDrop, OnDragOver, OnEndDock, OnGetSiteInfo, OnHelp, OnHide, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnMouseWheel, OnMouseWheelDown, OnMouseWheelUp, OnPaint, OnResize, OnShortCut, OnShow, OnStartDock, OnUnDock
HTMLText	ECA:CCA::Process Component	PPres		OnClick, OnContextPopup, OnDblClick, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnMouseDown, OnMousemove, OnMouseUp, OnStartDock, OnStartDrag
HTMLMenus	ECA:CCA::Process Component	PPres		
HTMLNewForm	ECA:CCA::Process Component	PPres	HTMLMenus	OnChange, OnPopup
HTMLCustomMenu HTMLSelect	ECA:CCA::Process Component	PPres	HTMLMenus	OnChange
HTMLMenuItem	ECA:CCA::Process Component	PPres		OnAdvancedDrawItem, OnClick, OnDrawItem, OnMeasureItem
HTMLImage	ECA:CCA::Process Component	PPres		OnClick
PhPTable	ECA:CCA::Process Component	PPres		OnClick, OnColumnMoved, OnContextPopup, OnDblClick, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnGetEditMask, OnGetEditText, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnMouseWheelDown, OnMouseWheelUp, OnRowMoved, OnSelectCell, OnSetEditText, OnStartDock, OnStartDrag, OnTopLeftChanged
HTMLForm	ECA:CCA::Process Component	PPres		OnClick, OnContextPopup, OnDblClick, OnDockDrop, OnDockOver, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnGetSiteInfo, OnMouseDown, OnMouseMove, OnMouseUp, OnStartDock, OnStartDrag, OnUnDock
HTMLFrame	ECA:CCA::Process Component	PPres		OnCanResize, OnClick, OnConstrainedResize, OnContextPopup, OnDblClick, OnDockDrop, OnDockOver, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnGetSiteInfo, OnMouseDown, OnMouseMove, OnMouseUp, OnResize, OnStartDock, OnStartDrag, OnUnDock
HTMLInput Type=submit	ECA:CCA::Process Component	PPres		OnClick, OnContextPopup, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnStartDock, OnStartDrag
HTMLInput Type=text	ECA:CCA::Process Component	PPres		OnChange, OnClick, OnContextPopup, OnDblClick, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnStartDock, OnStartDrag
HTMLInput Type=checkbox	ECA:CCA::Process Component	PPres		OnClick, OnContextPopup, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnStartDock, OnStartDrag
HTMLInput Type=text	ECA:CCA::Process Component	PPres		OnClick, OnContextPopup, OnDblClick, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnKeyDown, OnKeyPress, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnStartDock, OnStartDrag
HTMLInput Type=radio	ECA:CCA::Process Component	PPres		OnClick, OnContextPopup, OnDragDrop, OnDragOver, OnEndDock, OnEndDrag, OnEnter, OnExit, OnStartDock, OnStartDrag

Κεφάλαιο 4: Ανάπτυξη Εργαλείων

Η μεθοδολογία που περιγράφουμε στη παράγραφο 3.1 εφαρμόζεται με την ανάπτυξη δύο βασικών εργαλείων. Το πρώτο εργαλείο περιλαμβάνει δύο κύριες εργασίες, εκ των οποίων η πρώτη είναι να κατασκευάσει με βάση τον πηγαίο κώδικα ένα αρχείο, που να περιέχει πληροφορία για το αρχικό *PSM*, και η δεύτερη με βάση κάποια *patterns* μπορεί και μετασχηματίζει ένα μοντέλο σε ένα νέο. Με το εργαλείο αυτό πετυχαίνουμε την μετατροπή *PSM* σε *PIM* χρησιμοποιώντας *abstraction patterns*, αλλά και του *PIM* σε *PSM* χρησιμοποιώντας *refinement patterns*. Με αυτό τον τρόπο τον τρόπο βλέπουμε ότι πληρείται η απαίτηση για την κατασκευή εργαλείων απαλλαγμένα από την φύση των τεχνολογικών πλατφόρμων. Το δεύτερο εργαλείο αναλαμβάνει την μετατροπή των αρχείων που έχουμε στην διάθεσή μας (περιγράφουν είτε *PSM* είτε *PIM*) σε μία νέα μορφή αρχείων (*mdl*) (Σχήμα 30). Τα παραγόμενα αυτά αρχεία



Σχήμα 30: ***Το σύστημα που υλοποιήσαμε και τα υποσυστήματά του.***

οπτικοποιούνται με τη βοήθεια του προγράμματος *Rational Rose*. Με αυτό τον τρόπο η βασική μεθοδολογία μας που υλοποιείται από το δεύτερο εργαλείο είναι ανεξάρτητη και από το εργαλείο οπτικοποίησης. Στην παράγραφο 4.1 γίνεται αναλυτική αναφορά στα *refinement* και *abstraction patterns* που χρησιμοποιήσαμε στην δική μας εφαρμογή για να περάσουμε από τη παλαιά αναπαράσταση σε μια νέα. Επίσης αναφέρουμε τις εργασίες που γίνονται στην αρχή με σκοπό να συλλέξουμε την απαιτούμενη πληροφορία για την κατασκευή των μοντέλων (4.1.1). Η πληροφορία αυτή διοχετεύεται στο εργαλείο που περιγράφουμε στην παράγραφο 4.2 του οποίου ο σκοπός είναι να κατασκευάσει μία οπτική αναπαράσταση των μοντέλων.

4.1 Εργαλείο εξόρυξης και μετατροπής μοντέλων

Το εργαλείο που παρουσιάζουμε στην ενότητα αυτή χωρίζεται σε τρία κύρια τμήματα:

- Επεξεργασία και ταξινόμηση του πηγαίου κώδικα που διαθέτουμε από το παλιό *BIS* (Τμήμα Α).
- Εργασίες που σχετίζονται με την πρόσθια κατασκευή (Τμήμα Β).
- Εργασίες που σχετίζονται με την αντίστροφη κατασκευή (Τμήμα Γ).

4.1.1 Τμήμα Α: Ανίχνευση αντικειμένων στον πηγαίο κώδικα και δόμηση της πληροφορίας σε αρχεία

Για την ανάπτυξη του δικού μας εργαλείου διερευνήσαμε για τον τρόπο με τον οποίο αποθηκεύονται τα *projects* των διάφορων *COTS*. Κατ' αρχήν όλες οι εν λόγω πλατφόρμες βασίζονται σε αντικειμενοστρεφές περιβάλλον, κάτι που διευκολύνει πολύ την προσέγγισή μας και την πρακτική υλοποίηση της προτεινόμενης μεθοδολογίας.

Όλες οι πλατφόρμες θέτουν ως βάση την ύπαρξη ενός "*project*". Το "*project*" αποτελείται από σύνθεση πολλών αρχείων. Υπάρχει ένα αρχείο, το οποίο περιέχει όλα τα αντικείμενα που συνθέτουν την εκάστοτε φόρμα (*dfm* στη *Delphi* και *frm* στη *Visual Basic*). Στα αρχεία αυτά περιέχονται με κάποια σειρά τα στοιχεία παραθέτοντας όλες τις παραμέτρους τους και αναφορές για τις μεθόδους που καλούν. Οι μέθοδοι υπάρχουν είτε σε ένα μόνο αρχείο, όπως για παράδειγμα κάνει η *Delphi*, είτε σε μία συλλογή από διαφορετικά αρχεία. Τέλος για την λογική

σύνδεση των παραπάνω αρχείων υπάρχουν ένα ή και περισσότερα αρχεία που έχουν αντίστοιχο ρόλο με το *Makefile* της *C*. Στην *Delphi* τον ρόλο αυτό τον έχει ένα αρχείο με προέκταση *dpr*, ενώ στη *Visual Basic* υπάρχουν τουλάχιστον τρία αντίστοιχα αρχεία, τα *scc*, *vbw* και *vbp*.

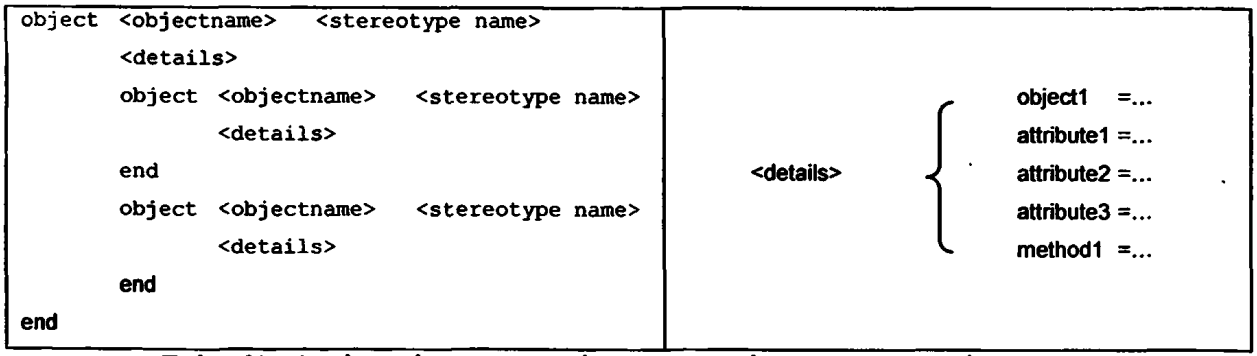
Στη δικιά μας προσέγγιση επεξεργαζόμαστε αρχεία πηγαίου κώδικα που προέρχονται από τη *Borland Delphi 5*. Η τεχνολογική αυτή πλατφόρμα ακολουθεί αντικειμενοστρεφές περιβάλλον. Παραπάνω αναφέρθηκε η ύπαρξη *"projects"*, έτσι για την *Delphi* τα αρχεία που αποτελούν ένα *"project"* είναι: Ένα αρχείο με επέκταση *dpr* στο οποίο περιέχεται μία λίστα από φόρμες, οι οποίες συνθέτουν το σύστημά μας. Οι φόρμες αυτές φυλάσσονται η καθεμιά τους σε ξεχωριστό ζεύγος αρχείων. Τα αρχεία εμφανίζονται σε ζεύγη με προεκτάσεις *dfm* και *pas*. Δηλαδή στο αρχείο *dpr* θα υπάρχει η περιγραφή για την ύπαρξη μίας φόρμας b και ενός αρχείου "b.pas". Σίγουρα θα υπάρχει και το αντίστοιχο αρχείο "b.dfm". Στα αρχεία με επέκταση *dfm* η *Delphi* φυλάσσει με καλά δομημένη μορφή πληροφορία, η οποία σχετίζεται με τα αντικείμενα που έχουν χρησιμοποιηθεί στη φόρμα. Για κάθε αντικείμενο του *dfm* γίνεται αναφορά των παραμέτρων (*attributes*), των *events* που μπορεί να προκληθούν (π.χ. για ένα κουμπί *OnClick*), αλλά και των διαδικασιών (*procedures*) που σχετίζονται με κάθε *event*. Οι παραπάνω διαδικασίες υπάρχουν στα αρχεία με επέκταση *pas*. Όταν κατά τη διάσχιση του *dfm* έχουμε αναφορά μίας διαδικασίας σε σχέση με κάποιο αντικείμενο, τότε αναζητούμε στο αντίστοιχο αρχείο *pas* τον πηγαίο κώδικα που αντιστοιχεί (πρόκειται για κώδικα που έχει εισάγει προγραμματιστής).

Εμείς έχουμε επιλέξει την κατασκευή ενός εργαλείου το οποίο χρησιμεύει μόνο για συστήματα κατασκευασμένα σε *Delphi*. Επιλέξαμε να φτιαχτεί ένα εργαλείο για την ανίχνευση αντικειμένων για κάθε πλατφόρμα ξεχωριστά με την προϋπόθεση ότι όλα τα εργαλεία παράγουν ένα σύνολο από αρχεία με κοινή σύνταξη. Για τις εργασίες που ακολουθούν τα εργαλεία χρησιμοποιούν το σύνολο των αρχείων με κοινή σύνταξη. Η παραπάνω λύση θα πρέπει να τονιστεί ότι δεν επηρεάζει την ανεξαρτησία των επόμενων εργαλείων από την φύση της τεχνολογικής πλατφόρμας.

Για την κοινή σύνταξη των στοιχείων εμείς έχουμε επιλέξει την ύπαρξη ενός αρχείου *A* το οποίο περιέχει έναν κατάλογο από αρχεία (τις σχετικές τους θέσεις στον δίσκο) και έναν φάκελο του συστήματος αρχείων (*directory*) που να αντιστοιχεί στο κάθε ένα αρχείο. Το κάθε αρχείο *B*

από τον παραπάνω κατάλογο του *A* περιγράφει τα διάφορα αντικείμενα μίας φόρμας με κατάλληλη σύνταξη, την οποία και περιγράφουμε παρακάτω. Στον κατάλογο που αντιστοιχεί για το κάθε αρχείο *B* υπάρχουν αρχεία με ονόματα που παραπέμπουν σε μεθόδους που καλούνται και ορίζονται από τα αντικείμενα που έχουν οριστεί στο *B*. Με λίγα λόγια το αρχείο *B* χτίζεται με βάση τον *Platform Dependent Code* (*PDC*), ενώ στα αρχεία που υπάρχουν στον κατάλογο υπάρχει κώδικας, που έχει γραφτεί από τους προγραμματιστές, για να εκπληρώσει κάποιες ενέργειες (*PIC*).

Η δομή του παραπάνω αρχείου *B* μοιάζει με αυτή του Σχήματος 31. Ο λόγος που επιλέξαμε αυτή την αναπαράσταση είναι, ότι όλες οι τεχνολογικές πλατφόρμες που εξετάσαμε (εργαλεία της *Borland* και εργαλεία που ανήκουν στην *MSDN*) έχουν παρόμοια δομή. Πάντα το πρώτο "*object*" είναι η "*form*" της αντίστοιχης πλατφόρμας, και σε αυτό επιδίδονται αντικείμενα και σύμφωνα με τις εξαρτήσεις που υπάρχουν αργότερα τοποθετείται και η αντίστοιχη συσχέτιση. Στο Σχήμα 31 παρατηρούμε να υπάρχουν ενθυλακωμένα "*objects*" το ένα μέσα στο άλλο. Η δομή αυτή έχει να κάνει με την ομαδοποίηση αντικειμένων, που εξαρτώνται άμεσα από το ίδιο αντικείμενο. Ένα παράδειγμα εξαρτήσεων είναι αντικείμενα, που ανήκουν στη γραφική διεπαφή, είναι ομαδοποιημένα και ανήκουν σε κάποιου είδους καμβά, είτε αυτά είναι οργανωμένα σε *panel*, είτε σε *groupbox* (στην παράγραφο 3.2 αυτά θα αναπαρίστανται με στοιχεία *PICanvas, PIPanel* και *PIGroupBox* κτλ). Το *panel* αυτό με την σειρά του μπορεί να ανήκει απ' ευθείας στην κύρια φόρμα του προγράμματος ή να ανήκει σε άλλες φόρμες κτλ. Επιπλέον *constraints* για τα αντικείμενα δύναται να δοθούν στο σημείο <*details*> του Σχήματος 31.



Σχήμα 31: ***Δομή αρχείων που περιγράφουν τα αντικείμενα και τις παραμέτρους τους.***

Στα *details* Σχήμα 31b υπάρχουν μία σειρά από *attributes* που αφορούν άμεσα το αντικείμενο (π.χ. μέγεθος, θέση, χρώμα, επικεφαλίδα κτλ), τα *events* για τα οποία έχουν οριστεί οι

αντίστοιχες μέθοδοι και κάποια *attributes* τα οποία σχετίζονται με χρήση άλλων αντικειμένων. Για παράδειγμα ένα αντικείμενο, που σχετίζεται με ανάκτηση δεδομένων (π.χ. *PIDBQuery*), θα έχει κάποια *attributes* που θα αναφέρουν κάποιο αντικείμενο που σχετίζεται με την σύνδεση με την αντίστοιχη βάση δεδομένων (π.χ. *PIDBConnection*).

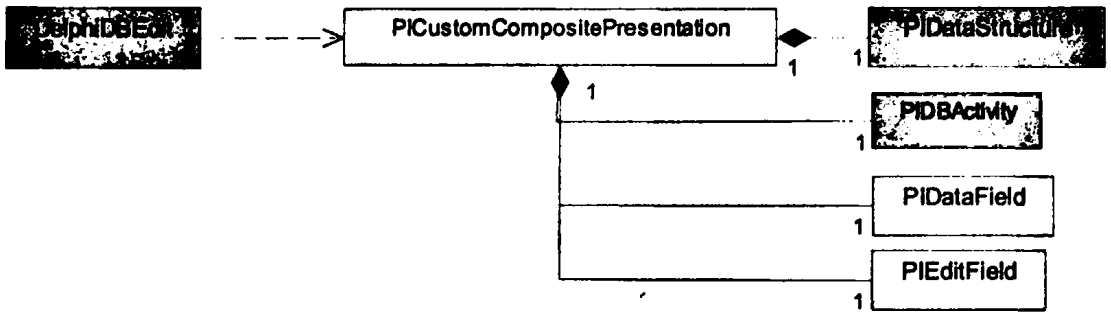
Την παραπάνω δομή δεν την χρησιμοποιούμε μόνο στην αρχή για την περιγραφή του κώδικα, αλλά και για την περιγραφή των μοντέλων των δύο *PSM* και του *PIM*. Οτιδήποτε έχουμε δομήσει με την παραπάνω μορφή, μπορεί με το εργαλείο που παρουσιάζουμε στην ενότητα 4.2 να παρουσιάζεται με γραφικό τρόπο. Λεπτομέρειες για το τι χρειάζονται για να γίνουν τα αναφερθέντα παραθέτουμε μαζί με την παρουσίαση του εργαλείου. Στην συνέχεια παρουσιάζουμε αναλυτικά τις εργασίες κατά την εφαρμογή της αντίστροφης και πρόσθιας κατασκευής.

4.1.2 Τμήμα Β: Αντίστροφη κατασκευή

Στην παράγραφο 3.1 αναφέρουμε την εξαγωγή πληροφορίας σχετικής με το *PSM* από τον πηγαίο κώδικα του παλαιού συστήματος, το οποίο θα περιγράφεται σε ένα αρχείο της μορφής που αναφέρεται στην ενότητα 4.1.1. Έπειτα με ένα σύνολο απεικονίσεων (*abstraction patterns*), το οποίο εξαρτάται από την τεχνολογική πλατφόρμα που χρησιμοποιήθηκε στο παλαιό *BIS*, κατασκευάζεται το αντίστοιχο *PIM*.

Αρχικά το εργαλείο διασχίζει το αρχείο που περιέχει πληροφορία σχετική με το *PSM* και ανιχνεύει αντικείμενα και *APIs*. Για κάθε ένα από αυτά τα στοιχεία αντιστοιχούμε ένα στοιχείο *UML* και τοποθετείται στο *PSM* που εκφράζει το παλαιό *BIS*. Το κάθε *component*, με βάση τα στοιχεία που χρησιμοποιήσαμε για να το εξάγουμε, αντιπροσωπεύεται από ένα *stereotype*. Για την περίπτωση της *Delphi* το κάθε στοιχείο το αντιστοιχούμε σε ένα από τα *stereotypes* της ενότητας 3.3 Αφού έχουμε ολοκληρώσει την δημιουργία του *Delphi PSM*, κατασκευάζουμε το *PIM* με τη βάση ένα σύνολο απεικονίσεων "*abstraction patterns*" που περιγράφουμε παρακάτω. Δηλαδή κάθε ένα *stereotype* X του *Delphi PSM* αντιστοιχίζεται σε ένα *stereotype Y* ή σε ένα σύνολο από *stereotypes* του *PIM*.

Πριν αναφερθούμε αναλυτικά στις απεικονίσεις που εμείς ορίσαμε για το δικό μας *case study* αναφέρουμε για καλύτερη κατανόηση ένα παράδειγμα απεικόνισης. Το παράδειγμα βασίζεται στην περίπτωση του πίνακα 4. Μετατρέπουμε ένα αντικείμενο το οποίο αναπαριστάται με τη χρήση του *DelphiDBEdit stereotype* που έχει οριστεί στο *Delphi PSR* στα αντίστοιχα του *PIR*, όπως εμφανίζεται στο Σχήμα 32.



Σχήμα 32: ***Μετατροπή του DelphiDBEdit στα αντίστοιχα του PIR.***

Ο ρόλος του *DBEdit* στην *Delphi* είναι ένα *edit box*, το οποίο αντιστοιχεί σε ένα πεδίο (σε ένα *cell*) κάποιας βάση δεδομένων. Στο Σχήμα 32 βλέπουμε ότι το στοιχείο *DelphiDBEdit* στο *PIM* θα αντικατασταθεί από ένα στοιχείο που χαρακτηρίζεται από το *PICustomCompositePresentation*, το οποίο θα γίνει το στοιχείο επί του οποίου θα επικαθίσουν κάποια άλλα στοιχεία. Τα στοιχεία που προσαρτούνται στο παραπάνω είναι:

- Ένα *PIEditField* το οποίο αντιστοιχεί σε στοιχείο που ανήκει στην γραφική διεπαφή με τον χρήστη και είναι, όπως προδίδει και το όνομα του, ένα πλαίσιο για την σύνταξη κειμένου (*edit box*).
- Ένα *PIDataField* στοιχείο που σχετίζει τα δεδομένα που υπάρχουν στο παραπάνω πλαίσιο με μία μεταβλητή και μπορεί να χρησιμοποιηθεί για διεργασίες στο παρασκήνιο.
- Ένα *PIDBActivity* το οποίο θα αναλάβει την μεταβολή της τιμής ή των δεδομένων που σχετίζονται με το *PIDataField* στοιχείο.
- Ένα *PIDataStructure* το οποίο επιστρέφει τα δεδομένα που ζητήθηκαν από το παραπάνω *PIDBActivity*. Τα δεδομένα αυτά περιμένουμε έμμεσα ο προγραμματιστής να τα αντιστοιχήσει στο *PIDataField* και να εμφανιστούν στην διεπαφή του χρήστη στο πλαίσιο του *PIEditField*.

Στον πίνακα 4(α) φαίνονται οι παράμετροι που αφορούν ένα *DelphiDBEdit* εμφανίζονται με την μορφή παραμέτρων στα πεδία *DataField* και *DataSource*. Αυτές οι παράμετροι θα πρέπει κατά την

κατασκευή του αρχείου, που περιγράφει το *PIM*, να αντιστοιχηθούν ως παράμετροι στα νέα στοιχεία της απεικόνισης. Έτσι για κάθε μετατροπή (στην περίπτωσή μας τώρα *PSM* σε *PIM*) έχουμε ένα σύνολο από αρχεία, όπως αυτό στον πίνακα 4(β). Για κάθε στοιχείο του *PSM* θα πρέπει να υπάρχει ένα αρχείο που να περιγράφει τα νέα στοιχεία, αλλά και πληροφορία, ώστε να αποδοθούν σωστά οι παράμετροι στα αντίστοιχα στοιχεία (όποτε αυτό είναι αναγκαίο). Στο παράδειγμα του πίνακα 4, βλέπουμε τα *DataField* και *DataSource* του *DelphiDBEdit* να αποδίδονται αντίστοιχα στα *PIDataField* και *PIDataSource*. Παράμετροι που αφορούν τη γραφική διεπαφή, όπως τα: *Left, Top, Width, Height, Color* και παράμετροι που σχετίζονται με το *font*, αποδίδονται στο *PIEditField*. Όσοι παράμετροι δεν αναφέρονται σε τι αντιστοιχούν αποδίδονται στο βασικό στοιχείο της απεικόνισης, το οποίο στην περίπτωσή μας είναι το *PICustomCompositePresentation*.

Πίνακας 4 ***Παράδειγμα για τον τρόπο μετατροπής του Delphi PSM σε PIM (a) (b).***

```
Object DBEditBox1 DelphiDBEdit
        Left = 178
        Top = 8
        Width = 74
        Height = 19
        Color = 12446960
        DataField = 'Ju3A'
        DataSource = Month_ADOQuery_extrads
        Enabled = False
        Font.Charset = DEFAULT_CHARSET
        Font.Color = clWindowFrame
        Font.Height = -8
        Font.Name = 'MS Sans Serif'
        Font.Style = [fsBold]
        ParentFont = False
end
```

```
PICustomCompositePresentation
        object EditField PIEditField
                Left
                Top
                Width
                Height
                Font.Charset
                …
        end
        object DBActivity PIDBActivity
        end
        object DataSource PIDataStructure
                DataSource
        end
        object EditField PIDataField
                DataField
        end
```

Στην παραπάνω δομή δεν παράγεται πληροφορία για το είδος της συσχέτισης που ενώνει τα διάφορα στοιχεία. Την πληροφορία αυτή την παρέχουμε στα στοιχεία που χρειαζόμαστε για να απεικονίσουμε γραφικά το *PIM*, κάτι που σχετίζεται με το εργαλείο της παραγράφου 4.2.

Απεικόνιση στοιχείων που ανήκουν στο *Delphi PSR* σε στοιχεία του *PIR*

Για το δικό μας *case study* χρειάστηκε να μετατρέψουμε μοντέλα που χρησιμοποιούσαν αναπαραστάσεις *Delphi PIR* σε μοντέλα *PIM*. Λόγω του μεγάλου πλήθους στοιχείων τα έχουμε διασπάσει σε αυτά που έχουν ένα προς ένα αντιστοιχία, τα οποία παρουσιάζονται στον πίνακα 5,

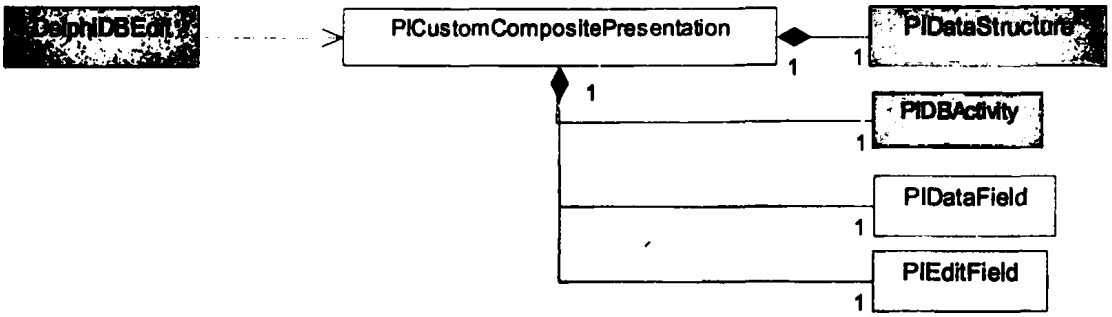
και έπειτα εμφανίζουμε όλα τα στοιχεία, των οποίων η αναπαράστασή τους στο *PIR* ορίζεται από μία σύνθεση στοιχείων (διάσπαση) του *PIR*, σαν αυτό που περιγράψαμε στην προηγούμενη παράγραφο. Αξίζει να αναφερθεί ότι στο δικό μας *case study* δεν είχαμε την περίπτωση συνδυασμός στοιχείων του *Delphi PSR* να απεικονίζεται από ένα στοιχείο του *PIR* (σύνθεση).

Πίνακας 5: ***Οι ένα προς ένα αναπαραστάσεις στοιχείων του Delphi PSR σε στοιχεία του PIR.***

Delphi PSR		***PIR***
DelphiForm	→	*PIForm*
DelphiAction	→	*PIActions*
DelphiRegValues	→	*PITechnologyConnection*
DelphiAPI	→	
DelphiGauge	→	*PICustomComposite-*
DelphiTabSheet	→	*Presentation*
DelphiBevel	→	*PICustomPrimitive Presentation*
ADOConnection	→	*PIDBConnection*
ADOCommand	→	*PIDBRawCommand*
ADODataSet	→	*PIDataStructure*
ADOQuery	→	*PIDBQuery*
ADOTable	→	*PIDBTable*
DelphiDBAction	→	*PIDBActivity*
DelphiComposite-Presentation	→	*PICompositePresentation*
DelphiEditField	→	*PIEditField*
DelphiLabel	→	*PILabel*
DelphiComboBox	→	*PIDropDownMenu*
DelphiMemo	→	*PIMemo*
DelphiMemoField	→	
DelphiGrid	→	*PITable*
DelphiCustomCom-positePresentation	→	*PICustomComposite-Presentation*
DelphiCustomPri-mitivePresentation	→	*PICustomPrimitive-Presentation*
DelphiPanel	→	*PIPanel*
DelphiMainMenu	→	*PINewForm*
DelphiPopUpMenu	→	*PINewForm*

Delphi PSR		***PIR***
DelphiEvent	→	*PIEvent*
DelphiClientSocket	→	*PINetworkConnection*
DelphiServerSocket	→	
DelphiWebBrowser	→	*PITechnologyConnection*
DelphiToolBar	→	*PIToolBar*
DelphiToolButton	→	*PIToolButton*
DelphiDataSource	→	*PIDataStructure*
DelphiBooleanField	→	
DelphiFloatField	→	
DelphiIntegerField	→	*PIDataField*
DelphiStringField	→	
DelphiWideStringField	→	
DelphiDBGuiIO	→	*PIPrimitivePresentaion*
DelphiPrimitive Presentation	→	*PIPrimitivePresentaion*
DelphiStaticText	→	*PIStaticText*
DelphiButton	→	*PIButton*
DelphiGroupBox	→	*PIGroupBox*
DelphiPanel	→	*PIPanel*
DelphiRadioGroup	→	*PIRadioGroup*
DelphiImage	→	*PIImage*
DelphiPageControl	→	*PICustomComposite-*
DelphiCtrlGrid	→	*Presentation*
DephiTabSheet	→	*PICustomComposite-Presentation*
DelphiCanvas	→	*PICanvas*
DelphiMenuItem	→	*PIMenuItem*

Στο *Delphi PSM* υπάρχουν τα στοιχεία, τα οποία εμφανίζονται στη διεπαφή του προγράμματος με τον χρήστη, με τη μορφή κουτιών, πινάκων κτλ. και αλληλεπιδρούν άμεσα με κάποια πηγή δεδομένων π.χ. βάση δεδομένων. Τέτοια στοιχεία είναι τα *DelphiDBEdit, DelphiDBMemo, DelphiDBGrid, DelphiDBText, DelphiDBField, DBLookupListBox, DelphiDB-CheckBox, DelphiDBImage, DelphiDBLookUpBox* και *DelphiDBLookUpComboBox.*



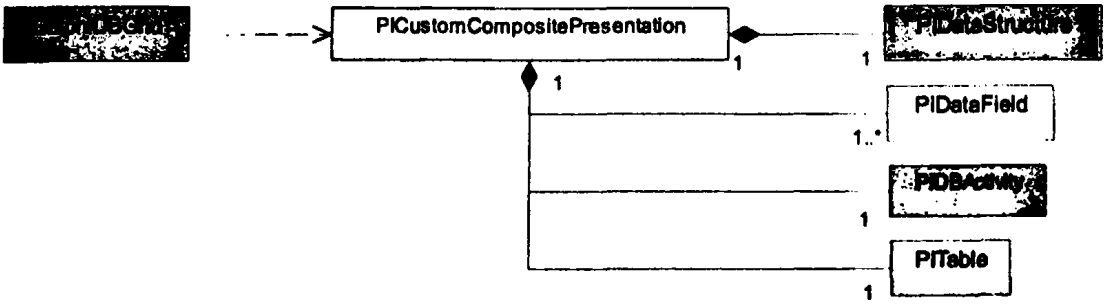
Σχήμα 33: ***Μετατροπή του DelphiDBEdit σε PIR.***

Όλα τα ανωτέρω αντιστοιχίζονται σε ένα *PICustomCompositePresentation,* στοιχείο του *PIR.* Η χρήση *custom* στοιχείων κρίνεται πολλές φορές αναγκαία για τις αντιστοιχίες ανάμεσα σε δύο μοντέλα. Αυτό συμβαίνει, επειδή μας διευκολύνει στην εύκολη αναγνώριση στο νέο μοντέλο ομάδων στοιχείων που προέρχονται από την ίδια διάσπαση. Δηλαδή ο σκοπός ύπαρξής τους εξυπηρετεί έναν κοινό σκοπό. Στο στοιχείο *PICustomCompositePresentation* προσαρτώνται:

- Ένα *PIDataStructure,* από το οποίο θα προέλθουν τα δεδομένα
- Ένα (ή περισσότερα στην περίπτωση του *DelphiDBGrid*) *PIDataField* που σχετίζονται με το πεδίο στο οποίο ανατρέχουμε για την πληροφορία.
- Ένα στοιχείο *PIDBActivity* του οποίου ο ρόλος είναι να κάνει αιτήσεις για την απόκτηση πληροφορίας από κάποια πηγή.
- Ένα στοιχείο που σχετίζεται άμεσα με την γραφική διεπαφή με τον χρήστη (π.χ. στο *DelphiDBCheckBox* θα υπάρχει το στοιχείο *PICheckBox* αντί του *PIEditField* ή του *PITable*).

Ένα τέτοιο παράδειγμα απεικονίζεται στο Σχήμα 33 για την περίπτωση του *DelphiDBEdit,* ενώ για την περίπτωση του *DelphiDBGrid* στο Σχήμα 34 παρατηρούμε την διαφορά στο *multiplicity* της σχέσης ανάμεσα στο *PITable* και *PIDataField.* Αυτό οφείλεται στο

γεγονός ότι το *DelphiDBGrid* χειρίζεται πληροφορία σε πλειάδες οι οποίες για να περιγραφούν χρειάζονται περισσότερα του ενός *PIDataField.*

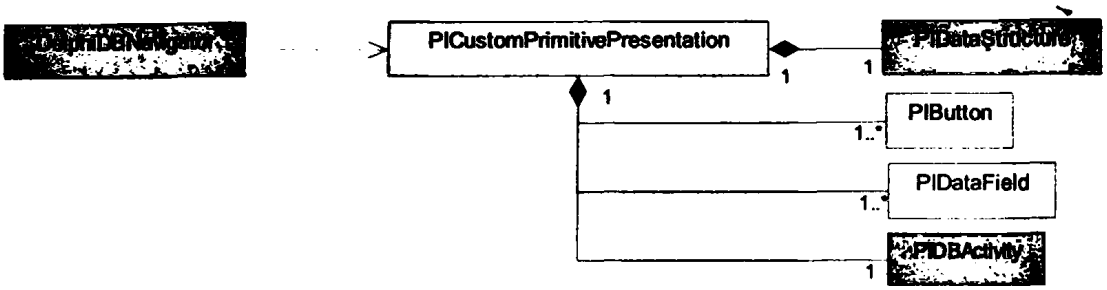


Σχήμα 34: ***Μετατροπή του DelphiDBGrid σε PIR.***

Το *DelphiImageList*, όπως αναφέρουμε στην παράγραφο 3.3, σχετίζεται με την διατήρηση στο *background* μίας λίστας από εικόνες, για παράδειγμα *bitmaps*, που χρησιμεύουν ως εικονίδια σε κουμπιά. Κάτι τέτοιο δεν υπάρχει σε ένα προς ένα αντιστοιχία στο *PIR*, αλλά ούτε μπορεί κάποια σύνθεση από τα στοιχεία του *PIR* να δώσουν ακριβή αντιστοιχία. Έτσι αντιστοιχίζεται, όπως βλέπουμε στο Σχήμα 35, σε ένα πλήθος *PIImage*, τα οποία προσαρτώνται σε ένα *PICustomCompositePresentation.*

Σχήμα 35: Μετατροπή του *DelphiImageList* σε *PIR.*

Αντίστοιχη περίπτωση συναντάμε στο *DelphiDBNavigator* του Σχήματος 36. Όπως περιγράψαμε σε προηγούμενη ενότητα το *DelphiDBNavigator* αποτελείται από διάφορα κουμπιά με τα οποία γίνονται έμμεσα κάποιες ενέργειες στα στοιχεία μίας βάσης δεδομένων. Έτσι στο *PIR* θα αντιστοιχίζεται σε ένα *PICustomPrimitivePresentation* στοιχείο, το οποίο αποτελείται από ένα πλήθος κουμπιών *PIButton* (ένα για κάθε επιλογή). Για να κάνει τις αντίστοιχες λειτουργίες που έκανε το *DelphiDBNavigator*, χρειάζεται ένα στοιχείο για την διεπαφή με μία βάση δεδομένων *PIDBActivity.* Τα δεδομένα αυτού επιστρέφονται μέσω του *PIDataStructure* και τα στοιχεία, τα οποία επηρεάζονται, εμφανίζονται στη διεπαφή με την μεταβολή στοιχείων *PIDataField.*



Σχήμα 36: ***Μετατροπή του DelphiDBNavigator σε PIR.***

4.1.3 Τμήμα Γ: Πρόσθια κατασκευή

Στη πρόσθια κατασκευή χρησιμοποιούμε το εργαλείο που αναφέραμε στην παραπάνω ενότητα, το οποίο μετέτρεπε το *PSM* σε *PIM*. Η παρούσα φάση μετατρέπει το *PIM* στο *PSM* που βασίζεται στην νέα τεχνολογική πλατφόρμα, όπου έχουμε επιλέξει στην αρχή της διαδικασίας για την ανακατασκευή. Αυτή τη φορά το εργαλείο μας παίρνει σαν είσοδο πληροφορία η οποία εμπεριέχει το *PIM* και στην θέση των *abstraction patterns* χρησιμοποιούμε *refinement patterns.*

Για κάθε *stereotype* του *PIM* περιγράφουμε πώς πραγματοποιείται η μετατροπή του στα αντίστοιχα *platform-specific* στοιχεία. Χαρτογραφούμε δηλαδή την μετατροπή ενός *PIR stereotype* σε ένα ή περισσότερα *PSR stereotypes.* Η μετατροπή αυτή μπορεί να συνδέεται με κάποιες επιπλέον συνθήκες οι οποίες δηλώνουν ότι, τα *stereotypes* που θα δημιουργηθούν στο *PSM,* μπορεί να σχετίζονται ή θα πρέπει να συσχετιστούν με κάποια άλλα στοιχεία του μοντέλου. Το εργαλείο μας φροντίζει εκτός από το να ικανοποιήσει τις επιπλέον συνθήκες που ενδέχεται να προσθέσουμε, να παράγει επίσης και κάποια τμήματα από την τελική υλοποίηση.

Για την παραγωγή τμημάτων της τελική υλοποίησης του *BIS* έχουμε φροντίσει, στο εργαλείο αυτό, να υπάρχει τρόπος εμφάνισης των *skeleton files* για κάθε στοιχείο του δοθέντος *PSM.* Τα παραγόμενα κομμάτια κώδικα χρησιμεύουν ως κατευθυντήριες γραμμές. Για την ολοκλήρωση του συστήματος γίνεται κατανοητό, ότι θα πρέπει να υπάρξει ανθρώπινη επέμβαση, ώστε να εκτελεσθεί η κατάλληλη συμπλήρωση και σύνθεση των παραπάνω παραγόμενων στοιχείων.

Ακολούθως παρουσιάζουμε τα κύρια σημεία για τα *refinement patterns* που χρησιμοποιήσαμε στην πρακτική εφαρμογή μας.

Απεικόνιση στοιχείων που ανήκουν στο *PIR* σε στοιχεία του *Web PSR*

Για τις ανάγκες της εφαρμογής μας χρειάστηκε να ορίσουμε την μετατροπή του *PIM* σε *Web PSR*. Σε αυτό το κομμάτι, λόγω της απλότητας που παρουσιάζει το *PSR* του μοντέλου που θέλουμε να καταλήξουμε, όλες οι συσχετίσεις είναι μία προς μία. Δηλαδή, ένα *stereotype* X του *PIR* αντιστοιχίζεται σε ένα *stereotype* Y του *PSR*. Δεν εμφανίζεται η διάσπαση ανάλογη με αυτές στα Σχήματα 33-36.

Στον Πίνακα 6 εμφανίζονται όλες οι αντιστοιχίες. Όλα τα στοιχεία τα οποία αφορούν το *presentation* πακέτο αντιστοιχίζονται σε στοιχεία που πραγματοποιούνται με την βοήθεια της *HTML*. Αντίστοιχα στοιχεία που έχουν να κάνουν με το *event handling* και *triggering*, αντιστοιχίζεται σε στοιχεία της *JavaScript*. Όλα τα υπόλοιπα στοιχεία βασίζονται γενικά στην χρήση της *php*.

Παρατηρούμε, ότι πολλά στοιχεία του *PIR* αντιστοιχήθηκαν στο *HTMLInput*. Η περίπτωση αυτή χρήζει ιδιαίτερης αναφοράς. Για να γίνει διαχωρισμός των παραπάνω *HTMLInput* προσθέτουμε ένα όρισμα, όπως περιγράφεται στα *details* του 30(b), το οποίο και ονομάζουμε *type* και χρησιμεύει στον διαχωρισμό της μορφής που έχει το *HTMLInput*. Περισσότερα για τις τιμές που μπορεί να έχει η παράμετρος αυτή αναφέρονται στη σχετική αναφορά του *stereotype HTMLInput* στην παράγραφο 3.4.

Τα *PIEditField* και *PIMemo*, σαν *type*, θα έχουν τιμή *text*, εφόσον θέλουμε το κείμενο να φαίνεται στον χρήστη, ενώ εάν επιθυμούμε να φαίνεται με την μορφή κρυμμένου κειμένου από '*' τότε το *type* θα έχει την τιμή *password* (αναλυτική περιγραφή γίνεται στη παράγραφο 3.4). Το *PIRadioGroup* θα έχει *type radio* και θα υπάρχει μία επιπλέον παράμετρος, η οποία θα καθορίζει την ομάδα στην οποία ανήκει το εκάστοτε στοιχείο. Αντίστοιχα για το *PICheckBox* θέτουμε τιμή *checkbox* στην παράμετρο *type*. Τέλος, για τα κουμπιά *PIButton* έχουν αντιστοιχισθεί στο

HTMLInput, στο οποίο η παράμετρος *type* παίρνει μία από τις τιμές *submit*, *reset*, *image* και εξαρτάται από τον ρόλο και την μορφή που θα έχει το κουμπί στην διεπαφή με τον χρήστη.

Πίνακας 6: ***Οι ένα προς ένα αναπαραστάσεις στοιχείων του PIR σε στοιχεία του Web representation.***

PIR		***PhP PSR***	***PIR***		***PhP PSR***
PIForm	→	*HTMLForm*	*PIEvent*	→	*JSEvent*
PLActivity	→	*PhPActivity*	*PIDiskConnection*	→	*PhPCustom*
PINetwork			*PITechnology*		
Connection	→	*PhPCustom*	*Connection*	→	*PhPCustom*
PIConnector	→	*PhPConnector*	*PIDBConnection*	→	*PhPDBConnection*
PIDataStructure	→	*PhPDataSource*	*PIDBActivity*	→	*PhPDBActivity*
PICanvas	→	*PhPCustom-CompositePresentation*	*PIComposite-Presentation*	→	*PhPComposite-Presentation*
PIGroupBox	→	*HTMLForm*	*PIMenu*	→	*HTMLMenus*
PIPanel	→	*HTMLFrame*	*PIPopUpMenu*	→	*HTMLNewForm*
PIStaticText	→	*HTMLText*	*PIMainMenu*	→	*HTMLCustomMenu*
PIComposite Presentation	→	*PhPComposite-Presentation*	*PIPrimitive-Presentation*	→	*PhPPrimitive-Presentation*
PIDropDownMenu	→	*HTMLSelect*	*PIToolBar*	→	*HTMLCustom*
PIMemo	→	*HTMLInput Type =text*	*PIToolButton*	→	*HMTLCustom*
PIEditField	→	*HTMLInput Type =text*	*PIComboBox*	→	*HTMLCustom*
PIRadioGroup	→	*HTMLInput Type =radio*	*PIDataField*	→	*HTMLInput Type =hidden*
PICheckBox	→	*HTMLInput Type =checkbox*	*PIImage*	→	*HTMLImage*
PIButton	→	*HTMLInput Type = submit | reset | image*	*PILabel*	→	*HTMLLabel*
PIDBQuery	→	*PhPDBQuery*	*PITable*	→	*HTMLTable*
PIDBTable	→	*PhPDBTable*	*PIDBRawCommand*	→	*PhPDBRawCommand*
PIDBDataSet	→	*PhPDataSource*	*PIDBConnection*	→	*PhPDBConnection*
PICustomCompositePresentation	→	*PhPCustom*	*PIMenuItem*	→	*HTMLMenuItem*
PICustomPrimitivePresentation	→	*PhPCustom*	*PINewForm*	→	*HTMLNewForm*

Τα *events* τα οποία περιγράψαμε για το *PIR* στο Σχήμα 8 δεν μπορούν όλα να απεικονιστούν άμεσα σε κάποιο του τωρινού *PSR*, κάτι το οποίο οφείλεται στους περιορισμούς που έχει η ίδια η πλατφόρμα. Για παράδειγμα ενέργειες, που έχουν να κάνουν με το *drug and drop* με την χρήση του ποντικιού, η *JavaScript* δεν διαθέτει άμεση λύση (στη παράγραφο 3.4 στην περιγραφή του *stereotype JSEvent* γίνεται σχετική αναφορά). Σε τέτοιες περιπτώσεις θα πρέπει ο προγραμματιστής του νέου συστήματος να εξετάσει τις ενέργειες του αντίστοιχου *PIActivity* και να βρει τον καταλληλότερο εναλλακτικό τρόπο βασιζόμενος στην εμπειρία (π.χ. να χρησιμοποιήσει άλλο *event*) ή να βρει κάποιο έτοιμο στοιχείο της *PhP*, το οποίο να εκτελεί αυτό που εμείς ζητάμε.

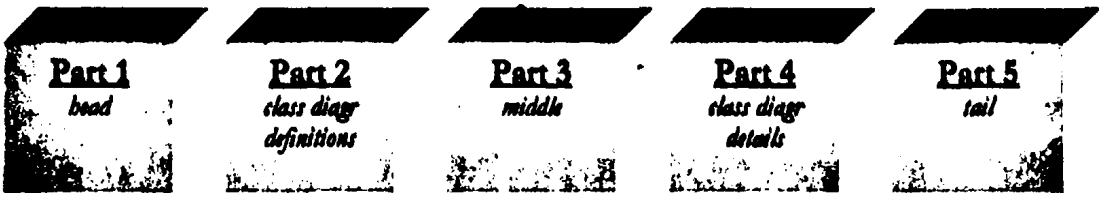
Για το σύνολο των εργασιών που σχετίζονται με διαβίβαση δεδομένων από και προς μία πηγή δεδομένων, είτε αυτή προέρχεται από κάποια αρχεία είτε από μία βάση δεδομένων, γίνονται όλα μέσω της χρήσης *php*. Η μεγάλη υποστήριξη που υπάρχει για την *php* τα τελευταία χρόνια έχει προκαλέσει την ύπαρξη πολλών έτοιμων στοιχείων τα οποία μπορούν να χρησιμοποιηθούν (παράδειγμα μίας τέτοιας συλλογής από στοιχεία παρέχει η *PEAR* [L20]). Από το *PIR* δύο στοιχεία, τα *PIDiskConnection* και *PINetworkConnection*, δεν έχουν κάποια απολύτως συγκεκριμένη απεικόνιση στο *PSR* που θέλουμε να καταλήξουμε, οπότε και αναγκαζόμαστε στην αντιστοίχησή τους με κάποιο *PhPComponent*.

4.2 Εργαλείο αναπαράστασης των μοντέλων με γραφικό τρόπο

Στην παράγραφο 4.1.1 περιγράψαμε μία δομημένη μορφή από αρχεία, τα οποία περιγράφουν διάφορα μοντέλα και χρήζουν να αναπαρασταθούν με γραφικό τρόπο. Για να το πετύχουμε αυτό, αποφασίσαμε την κατασκευή ενός εργαλείου, το οποίο θα παίρνει με κάποιο τρόπο τα παραπάνω αρχεία καθώς και συμπληρωματικές πληροφορίες και θα κατασκευάζει την γραφική τους αναπαράσταση με την βοήθεια κάποιου έτοιμου συστήματος.

Ως εργαλείο απεικόνισης επιλέξαμε την χρησιμοποίηση του πολύ διαδεδομένου *Rational Rose*, το οποίο υποστηρίζεται από την *IBM*. Το *ROSE* χρησιμοποιεί έναν αρκετά πολύπλοκο τρόπο στην αποθήκευση των μοντέλων του σε αρχεία.

Για να παράγουμε αρχεία *mdl*, βάση των οποίων σχεδιάζονται τα αντίστοιχα *class diagrams*, χρειάστηκε να βρούμε την λογική με την οποία είναι χτισμένα τα *mdl* αρχεία. Για τους σκοπούς των εργασιών μας το αρχείο χωρίζεται σε πέντε κομμάτια, όπως εμφανίζονται στο Σχήμα 37.



Σχήμα 37: ***Δομή του αρχείου που περιγράφει το class diagram για το ROSE (mdl).***

Τα *parts* 1,3 και 5 δεν μας απασχολούν ιδιαίτερα για το περιεχόμενό τους και τα χρησιμοποιούμε σε όλες τις περιπτώσεις αυτούσια όπως μας προέκυψαν αυτόματα από το *ROSE* για ένα αρχείο χωρίς κανένα στοιχείο. Στο *part* 2 γίνεται ο ορισμός των στοιχείων που συνθέτουν το μοντέλο μας. Δηλαδή αναφέρουμε όλες τις κλάσεις που θα έχει το μοντέλο μας, αλλά και όλες τις συσχετίσεις που θα χρησιμοποιήσουμε, ενώ λεπτομέρειες για αυτά αναφέρονται στο *part* 4. Δηλαδή στο *part* 4 υπάρχουν πληροφορίες που σχετίζονται με επικεφαλίδες, χρώματα, *fonts* συντεταγμένες και άλλα.

Πίνακας 7: *Ορισμοί για το mdl αρχείο (part 2) του (a) association (b) object.*

```
(object Association "$UNNAMED$-=MY=-"
        quid        "-=ASQUID=-"
        roles       (list role_list
          (object Role "$UNNAMED$-=MY2=-"
                  quid        "-=QUIDMY2=-"
                  supplier    "-=CHILDNAME=-"
                  quidu       "-=CHILDQUID=-"
                  is_navigable            -=ARG1=-
                  is_aggregate            -=ARG2=-)
          (object Role "$UNNAMED$-=MY1=-"
                          quid        "-=QUIDMY1=-"
                  supplier    "-=PARENTNAME=-"
                  quidu       "-=PARENTQUID=-"
                  is_navigable            -=ARG3=-
                  is_aggregate            -=ARG4=-)))
```

```
(object Class "-=OBJECTNAME=-"
        quid        "-=QUID=-"
        stereotype "-=STEREOTYPE=-"
        documentation
        -=DOCUMENTATION=-)
```

Στον πίνακα 7 αναφέρουμε την μορφή που έχουν οι ορισμοί για τα *associations* 7(a) αλλά και για τα *objects* 7(b) στο *part* 2. Όλα τα πεδία που εμφανίζονται στον παραπάνω πίνακα που είναι εντός "-=...=-" κατά την παραγωγή του *mdl* αρχείου αντικαθίστανται από κατάλληλες τιμές. Αντίστοιχα ισχύουν για την περίπτωση της συμπλήρωσης των αντίστοιχων πεδίων στο *part* 4 που αναφέρονται στον παρακάτω πίνακα 8.

Ειδική περίπτωση προκύπτει για τα αντικείμενα τα οποία κάνουν *inherit* από κάποιο άλλο. Σε αυτές τις ειδικές περιπτώσεις το *part* 2 της μορφής που περιγράφεται στον πίνακα 7(b) συμπληρώνεται από κάποια επιπλέον συμπληρωματικά στοιχεία, όπως αυτά φαίνονται στον πίνακα 9(α). Για το *part* 4 στις λεπτομέρειες των αντικειμένων προστίθεται πληροφορία που υπάρχει στον πίνακα 9(b).

Με όλα τα παραπάνω αυτό που μπορούμε να πετύχουμε είναι να τοποθετήσουμε μία κλάση στο μοντέλο μας. Εάν αυτή η κλάση κάνει *inherit* από μία άλλη κλάση, τότε μπορούμε να σχεδιάσουμε το αντίστοιχο *inheritance*. Επίσης ανάμεσα σε οποιεσδήποτε δύο κλάσεις του μοντέλου μπορούμε να σχεδιάσουμε ένα *association* και να του προσδώσουμε επιπλέον περιορισμούς, αν είναι τύπου *aggregate* ή *navigable* ή και τα δύο. Για να καθορίσουμε τους επιπλέον περιορισμούς στις σχέσεις αρκεί να προσδώσουμε κατάλληλες τιμές *true* ή *false* στις παραμέτρους ARG1 έως ARG4 του πίνακα 7(a).

Πίνακας 8: ***Λεπτομέρειες για το mdl αρχείο (part 4) του (a) association (b) object.***

```
(object AssociationViewNew "$UNNAMED$-=MY=-" @-=ID=-
        location    (607, 224)
        font        (object Font
          size      9
          face      "Arial"
          charSet             161
          bold      FALSE
          italics   FALSE
          underline           FALSE
          strike    FALSE
          color     0
          default_color       TRUE)
        stereotype  TRUE
        line_color  3342489
        quidu       "-=ASQUID=-"
        roleview_list       (list RoleViews
        (object RoleView "$UNNAMED$-=MY2=-"
                                    @-=AA2=-
                Parent_View         @-=ID=-
                location    (239, 0)
                stereotype  TRUE
                line_color  3342489
                quidu       "-=QUIDMY2=-"
                client      @-=ID=-
                supplier    @-=CHILDID=-
                line_style  0)
        (object RoleView "$UNNAMED$-=MY1=-" @-
=AA1=-
                Parent_View         @-=ID=-
                location    (239, 0)
                stereotype  TRUE
                line_color  3342489
                quidu       "-=QUIDMY1=-"
                client      @-=ID=-
                supplier    @-=PARENTID=-
                 vertices    (list Points
                (-=X1=-,-=Y1=-)
                (-=X2=-,-=Y2=-)
                (-=X3=-,-=Y3=-))
                line_style  0)))
```

```
(object ClassView "Class"
        "Logical View::-=OBJECTNAME=-" @-=ID=-
        ShowCompartmentStereotypes    TRUE
        SuppressAttribute     TRUE
        SuppressOperation     TRUE
        IncludeAttribute      TRUE
        IncludeOperation      TRUE
        location    (-=X=-, -=Y=-)
        font        (object Font
          size      9
          face      "Arial"
          charSet             161
          bold      FALSE
          italics   FALSE
          underline           FALSE
          strike    FALSE
          color     0
          default_color       TRUE)
        label       (object ItemLabel
          Parent_View         @-=ID=-
          location            (-=X=-, -=Y=-)
          fill_color          13434879
          nlines              1
          max_width           268
          justify   0
          label     "-=LABEL=-")
        stereotype  (object ItemLabel
          Parent_View         @-=ID=-
          location            (-=X=-, -=Y47=-)
          fill_color          13434879
          anchor              10
          nlines              1
          max_width           268
          justify   0
          label     "<<-=STEREOTYPE=->>")
        icon_style  "Icon"
        line_color  3342489
        fill_color  -=FCOLOR=-
        quidu       "-=QUID=-"
        width       286
        height      118
        annotation  8
        autoResize  TRUE
```

Πίνακας 9: ***(a) ορισμός κλάσης που κάνει inherit μία άλλη και (b) συμπληρωματικά στοιχεία για το 7(b).***

```
(object Class "-=OBJECTNAME=-"
        quid        "-=QUID=-"
        superclasses        (list inheritance_relationship_list
        (object Inheritance_Relationship
          quid      "-=QUIDINHERIT=-"
          label     "-=INHERITLABEL=-"
          supplier  "Logical View::-= PARENTOBJNAME=-"
          quidu     "-=QUIDPARENT=-")))
```

```
(object InheritView "-=INHERITLABEL=-" @-=IDH=-
        font        (object Font
          size      9
          face      "Arial"
          charSet             161
          bold      FALSE
          italics   FALSE
          underline           FALSE
          strike    FALSE
          color     0
          default_color       TRUE)
        stereotype  TRUE
        line_color  3342489
        quidu       "-=QUIDINHERIT=-"
        client      @-=ID=-
        supplier    @-=IDPARENT=-
        line_style  0)
```

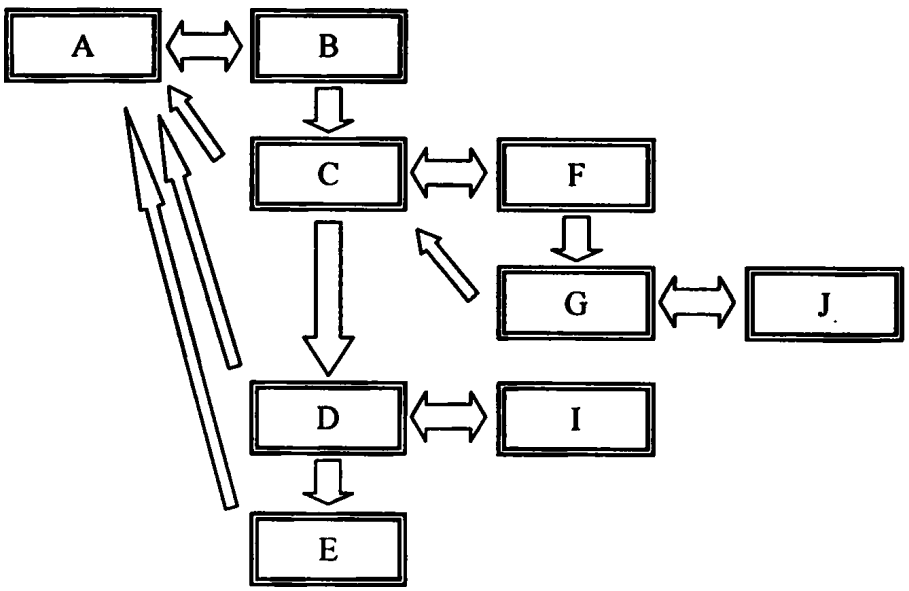
Για την φύση των σχέσεων που θα ενώνει δύο *stereotypes* του εκάστοτε μοντέλου, που θέλουμε να σχεδιάσουμε, το αναφέραμε ήδη ως συμπληρωματική πληροφορία στην περιγραφή της μεθοδολογίας μας. Εμείς επιλέξαμε την πληροφορία αυτή στο μοντέλο μας να την αποθηκεύσουμε σε ξεχωριστό αρχείο για κάθε *stereotype*. Όλα τα αρχεία που αντιστοιχούν για την περιγραφή μίας συγκεκριμένης αναπαράστασης βρίσκονται σε έναν συγκεκριμένο κατάλογο. Η πληροφορία δίνεται από μας την πρώτη φορά που θέλουμε να μοντελοποιήσουμε κάποια τεχνολογική πλατφόρμα. Δίνουμε την δυνατότητα στον σχεδιαστή του συστήματος να επέμβει στην πληροφορία αυτή και να ικανοποιήσει τις δικές του επιθυμίες. Αυτή η δυνατότητα συμβάλει στην ευελιξία της μεθόδου. Αναλυτικότερα, εάν υποθέσουμε την ύπαρξη (τουλάχιστον) δύο *stereotypes* A και B για την περιγραφή κάποιου μοντέλου π.χ. X, τότε θα έχουμε έναν κατάλογο *".\assoc X\"* στον οποίο θα υπάρχουν τα αρχεία A.txt και B.txt (φυσικά για κάθε *stereotype* θα υπάρχει ένα τέτοιο αρχείο). Στο αρχείο A.txt περιμένουμε να υπάρχει μία καταχώρηση της παρακάτω μορφής:

B val arg1 arg2 arg3 arg4

όπου Β είναι το *stereotype* ενός άλλου στοιχείου και η σχέση που θα ενώνει τα στοιχεία τύπου Α και Β θα εξαρτάται από τις τιμές:

- val
  - -1 εάν δεν υπάρχει καμία σχέση ανάμεσα στα Α και Β
  - 0 εάν το Α κάνει *inherit* το στοιχείο Β
  - 1 εάν υπάρχει απλό *association* ανάμεσα στα Α και Β και τα *arg1* έως *arg4* χρειάζονται για τους επιπλέον περιορισμούς
- arg1 θέτουμε 1 αν υπάρχει σχέση *navigable* από το Β προς το Α
- arg2 θέτουμε 1 αν υπάρχει σχέση *aggregate* προς τη μεριά του Β (Β contains Α)
- arg3 θέτουμε 1 αν υπάρχει σχέση *navigable* από το Α προς το Β
- arg4 θέτουμε 1 αν υπάρχει σχέση *aggregate* προς τη μεριά του Α (Α contains Β)

Μέχρι στιγμής έχουμε περιγράψει τον τρόπο με τον οποίο είναι δομημένο ένα αρχείο τύπου *mdl* και τον τρόπο με τον οποίο έχουμε αποθηκεύσει τα στοιχεία για το εκάστοτε *representation*, ώστε να βρίσκουμε την σχέση που θα ενώνει δύο στοιχεία του. Απομένει να παρουσιάσουμε τον τρόπο, με τον οποίο επεξεργαζόμαστε ένα αρχείο της δομής που περιγράφεται στην παράγραφο 4.1.1, για να καταλήξουμε στο ζητούμενο *mdl* αρχείο.



Σχήμα 38: ***Παράδειγμα για την δομή που χρησιμοποιείται για την αναπαράσταση ενός μοντέλου.***

Αρχικά το δοθέν αρχείο με βάση την δομή που δίδεται αποθηκεύεται σε μία δομή δεδομένων όπως αυτή φαίνεται στο Σχήμα 38. Μία τέτοια δομή θα προέκυπτε από ένα αρχείο της μορφής του παρακείμενου πίνακα 10. Η δομή που υπάρχει στο παραπάνω σχήμα, όσον αφορά την θέση των κουτιών, θα εμφανιστεί με την ίδια μορφή και στην τελική οπτική αναπαράσταση του μοντέλου με τη βοήθεια του *Rational ROSE*. Με βάση την πληροφορία που έχουμε για το μοντέλο, θα ελεγχθεί η σχέση που ισχύει ανάμεσα σε κάποιον κόμβο με τους γείτονες του και θα προστεθεί η ανάλογη πληροφορία – συσχέτιση στο αρχείο *mdl*. Στο παράδειγμά μας θα ελεγχθούν οι σχέσεις ανάμεσα στα AB, AC, AD, AE, CF, CG, DI και GF.

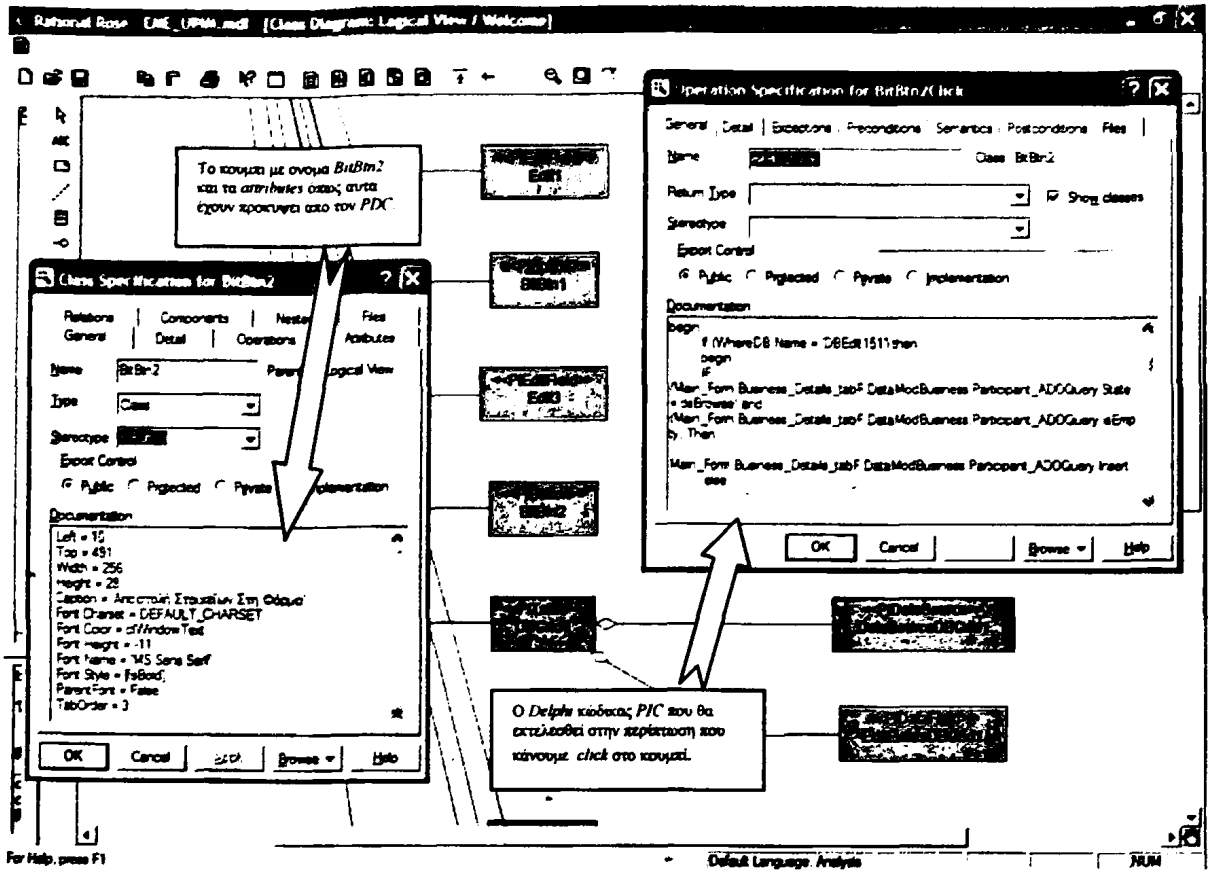
Πίνακας 10: ***Παράδειγμα αρχείου το οποίο περιγράφει κάποιο μοντέλο και αντιστοιχεί στο Σχήμα 38.***

```
object A ...
        object B ...
        end
        object C ...
                object F ...
                end
                object G ...
                        object J ...
                        end
                end
        end
        object D ...
                object F ...
                end
        end
end
```

Αφού ολοκληρωθεί η συμπλήρωση των συσχετίσεων για κάθε αντικείμενο, ελέγχουμε τα "*details*" του Σχήματος 31, όπως αυτά αναλύονται στην παράγραφο 4.1.1. Τα *attributes* τα εμφανίζουμε στο παραγόμενο γράφημα σαν "*documentation*", τα *objects* σαν "*attributes*" (για να είναι πιο εμφανής η εξάρτηση του αντικειμένου αυτού από κάποιο άλλο) και τέλος οι μέθοδοι σαν "*methods*" του αντικειμένου. Στο *documentation* της κάθε μεθόδου παρέχουμε τον πηγαίο κώδικα *PIC*, όπως τον έχουμε εξάγει από τον αρχικό γνωστό σύστημα.

Τέλος, όταν κατασκευάζουμε το *PSM* του τελικού συστήματος, στο πεδίο "*documentation*" του κάθε αντικειμένου, όπου αυτό είναι χρήσιμο, εισάγουμε το αντίστοιχο *skeleton* κομμάτι

πηγαίου κώδικα για το αντίστοιχο *stereotype*. Παράδειγμα του αποτελέσματος που προκύπτει εμφανίζεται στο *screenshot* της Εικόνας 1.



Εικόνα 1: ***Παράδειγμα απεικόνισης στο ROSE για κάποιο PIM το οποίο έχει προέλθει από Delphi PSM.***

Στο παρόν κεφάλαιο παρουσιάστηκε μία συλλογή από εργαλεία τα οποία υλοποιούν πρακτικά την μεθοδολογία που προτάθηκε στο 3° κεφάλαιο. Τα εργαλεία έχουμε φροντίσει να είναι ανεξάρτητα από τις εμπλεκόμενες τεχνολογικές πλατφόρμες που θέλουμε να επεξεργαστούμε. Η μόνη εξάρτηση από τις τεχνολογικές πλατφόρμες είναι στο βήμα του πρώτου εργαλείου που αφορά την ανίχνευση αντικειμένων στον δοθέν πηγαίο κώδικα. Στην περίπτωσή μας εξυπηρετεί μόνο την *Delphi*. Τα εργαλεία είναι πλήρως παραμετροποιήσιμα όσον αφορά την πληροφορία που σχετίζεται με την αναπαράσταση του μοντέλου αλλά και την πληροφορία που προσφέρουμε για την ανάπλαση των μοντέλων.

Κεφάλαιο 5: Case study

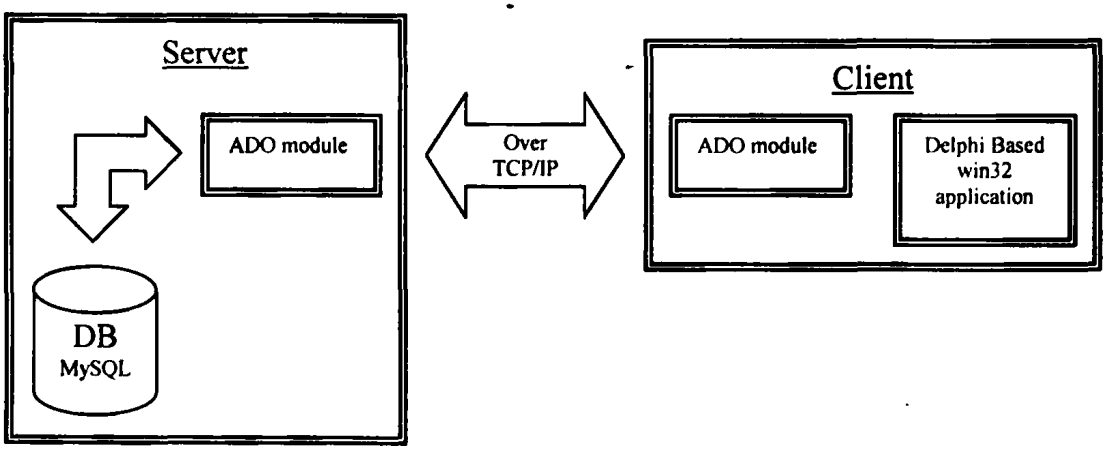
Στο παρόν κεφάλαιο γίνεται εκτενής αναφορά στον τρόπο εφαρμογής της προτεινόμενης μεθόδου στη δική μας εφαρμογή. Κάνουμε μια περιγραφή του συστήματος που έχουμε φτιάξει και αναφερόμαστε στην χρηστικότητα που παρουσιάζει το σύστημά μας από την μεριά του σχεδιαστή αλλά και του προγραμματιστή. Στο τέλος του κεφαλαίου αναφερόμαστε στα τελικά συμπεράσματα της παρούσας εργασίας.

5.1 Παράμετροι του συστήματος

Η μεθοδολογία και οι δοκιμές των εργαλείων μας εφαρμόστηκαν στην ανακατασκευή ενός *BIS* που χρησιμοποιείται από μία ελληνική τράπεζα. Σκοπός αυτού του συστήματος είναι να υποστηρίξει τη μηχανογράφηση των αιτήσεων για δάνεια, αυτόματη παραγωγή εγγράφων που αφορούν γραφειοκρατικά ζητήματα για την έγκριση του δανείου από το υπουργείο και τέλος την εποπτεία του έργου. Το αρχικό σύστημα είχε περατωθεί από το *Medical Technology & Intelligent Information Systems* του Πανεπιστημίου Ιωαννίνων. Σχετικά με την αρχιτεκτονική του παλαιού συστήματος είχαμε ελλιπή γραπτά κείμενα και καταφέραμε όμως να συλλέξουμε βασικά στοιχεία από μερικά άτομα που στελέχωναν την ομάδα ανάπτυξης του παλαιού συστήματος. Η ομάδα αυτή είχε προβεί στην απ' ευθείας υλοποίηση του συστήματος με τη χρήση *Borland Delphi, ADO* και *MySQL database*. Ο κύριος σκοπός της ανακατασκευής ήταν, να προσφέρουμε μία νέα μηχανογραφική υπηρεσία η οποία να επιτρέπει την εισαγωγή στοιχείων μέσω του *WEB*.

Ουσιαστικά η παραπάνω υπηρεσία αρχικά μοιάζει με την κατασκευή κάποιου νέου συστήματος, στην πραγματικότητα όμως ζητάμε την μετατροπή όλων των φορμών που υπάρχουν στο τοπικό περιβάλλον που αναπτύχθηκε με την βοήθεια της *Delphi*. Η τεθείσα προϋπόθεση, ότι το παλαιό σύστημα θα εξακολουθεί να είναι σε χρήση, δηλαδή το καινούργιο θα είναι συμβατό με το παλαιό, ανέβασε αρκετά τον δείκτη της δυσκολίας. Χωρίς την βοήθεια των εργαλείων που αναπτύξαμε οι απαιτούμενοι πόροι που θα απαιτούντο για το έργο αυτό θα ήταν πολύ περισσότεροι και το χρονικό διάστημα ανάπτυξης πολύ μεγαλύτερο. Αυτά οφείλονται στην μεγάλη πολυπλοκότητα της βάσεως δεδομένων, αλλά και στον τρόπο με τον οποίο υλοποιήθηκαν από την ομάδα ανάπτυξης.

Η αρχιτεκτονική του συστήματος για το παλαιό *BIS* αποτελείται από έναν κύριο υπολογιστή, ο οποίος είναι και κάτοχος της βάσης δεδομένων, η οποία λειτουργεί σε περιβάλλον της *MySQL*. Οι χρήστες προσπελαύνουν τις εφαρμογές με την βοήθεια εφαρμογών που υλοποιήθηκαν σε *Borland Delphi* και χρησιμοποιούν έτοιμα *ADO* στοιχεία για την απομακρυσμένη προσπέλαση της εν λόγω βάσης δεδομένων. Φυσικά υπάρχει και μία εφαρμογή, που φιλοξενείται από τον κύριο υπολογιστή, για να εξυπηρετήσει τις υπηρεσίες που χρειάζονται τα αναφερθέντα *ADO* στοιχεία. Όλα τα παραπάνω παρουσιάζονται με γραφικό τρόπο στο Σχήμα 39.

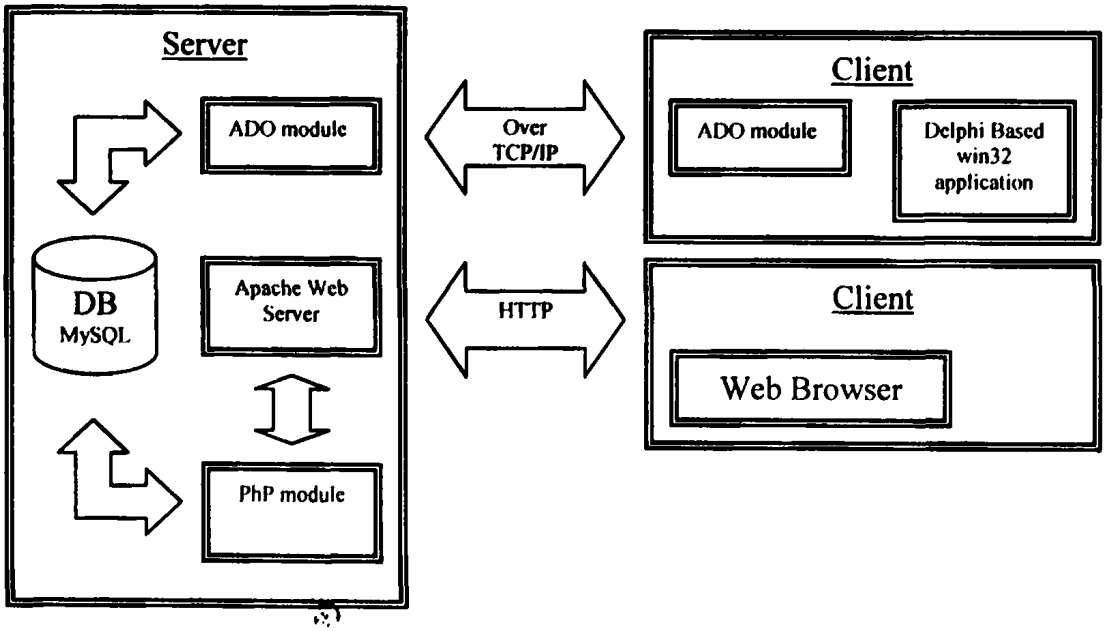


Σχήμα 39: ***Αρχιτεκτονικός σχεδιασμός των πλατφόρμων που χρησιμοποιήθηκαν από το πρωτότυπο σύστημα.***

Η ανακατασκευή είχε σαν στόχο όλες οι δοσοληψίες, που γίνονται ανάμεσα σε κάποιον *client* και της βάσης δεδομένων, να μην εκτελούνται μόνο με την χρήση της ήδη υπάρχουσας εφαρμογής, η οποία ήταν για περιβάλλον *win32*, αλλά να προστεθεί η δυνατότητα να γίνεται προσπέλαση με την χρήση ενός κοινού *web browser*. Οι λειτουργίες που εκτελεί η εφαρμογή του

παλιού συστήματος περιορίζονται στον έλεγχο των δοθέντων στοιχείων, στην εμφάνιση στοιχείων από την βάση δεδομένων καθώς και στην διαχείριση της παραπάνω βάσης. Οι λειτουργίες αυτές μπορούν να αναπτυχθούν με τα μοντέρνα περιβάλλοντα που σχετίζονται με την δυναμική παραγωγή ιστοσελίδων με στοιχεία που προέρχονται από κάποια βάση δεδομένων. Μερικά από τα συστήματα τέτοιων δυνατοτήτων είναι τα *cgi*, *ASP*, *JSP* και *php*.

Για την διεπαφή του χρήστη με την εφαρμογή -GUI- τα *COTS* συνήθως παρουσιάζουν περισσότερες επιλογές για *triggers* από τις δυνατότητες των *web* εφαρμογών. Τα *triggers* των *COTS* αντικατοπτρίζονται από τα *events*, που περιγράψαμε για κάθε *stereotype* του *Delphi PSM* στην ενότητα 3.2. Για να ελαττώσουμε το κενό ανάμεσα στα παραπάνω, στις δυναμικές σελίδες (που στην πιο απλή τους μορφή έχουν μόνο κώδικα HTML) προσθέτουμε και στοιχεία από *JavaScript*. Επειδή όμως τα *events* του συστήματος, που είναι υλοποιημένο με την βοήθεια ενός *COTS*, είναι υπερσύνολο των *events* που μπορούμε να έχουμε από *HTML* και *JavaScript*, οι ικανότητες του προγραμματιστή του νέου συστήματος είναι αυτές που θα πρέπει να λύσουν τέτοια πρακτικά προβλήματα που αφορούν την διεπαφή. Αναφέρουμε την ύπαρξη του τελικού προγραμματιστή ως αναγκαία, κάτι που αναφέρθηκε και στην παράγραφο 2.3. Στην επόμενη παράγραφο 5.2 που περιγράφουμε το εργαλείο μας αναφέρουμε και τον ρόλο του προγραμματιστή. Ο αρχιτεκτονικός σχεδιασμός του νέου συστήματος σύμφωνα με τα παραπάνω έχει την μορφή του Σχήματος 40.



Σχήμα 40: ***Ο νέος αρχιτεκτονικός σχεδιασμός των τεχνολογικών πλατφόρμων.***

5.2 Χρηστικότητα του συστήματός μας για τον σχεδιαστή

Στα προηγούμενα κεφάλαια εκθέσαμε λεπτομερώς τη μέθοδο ανακατασκευής συστημάτων που εμείς προτείνουμε, καθώς και τα εργαλεία που έχουμε κατασκευάσει για το σκοπό αυτό. Αξίζει τώρα να αναφερθούμε στην χρηστικότητα που έχουν όλα τα παραπάνω από την μεριά του σχεδιαστή συστημάτων.

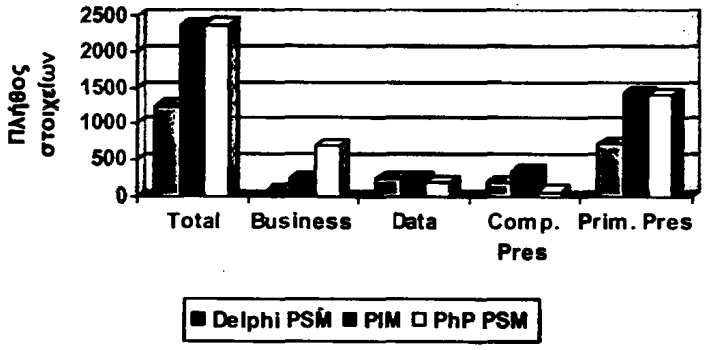
Στην παράγραφο 2.3 παρουσιάσαμε κάποια εργαλεία και μεθόδους ενώ στην παράγραφο 2.2.4 προβλήματα που σχετίζονται με την ανακατασκευή. Υπάρχει, όπως διαπιστώσαμε, μία έλλειψη τόσο σε εργαλεία που βοηθάνε στην ανακατασκευή, όσο και σε μία κοινή πλατφόρμα για την περιγραφή συστημάτων. Για μεγάλης έκτασης συστήματα, τα οποία βασίζονται σε σύνθεση πολλών τεχνολογικών πλατφόρμων, υπάρχει μεγάλη δυσκολία στην ανεύρεση εργαλείου που να βοηθά στην ανακατασκευή. Για παράδειγμα, ενδέχεται να υπάρχει κάποιο εργαλείο που να εφαρμόζεται σε συστήματα που είναι κατασκευασμένα σε τεχνολογικές πλατφόρμες A ή B, αλλά να μην υπάρχει εργαλείο το οποίο να εφαρμόζεται σε συστήματα που κάνουν ταυτόχρονη χρήση της A και της B.

Η δική μας προσέγγιση για τη λύση του προβλήματος είναι η ανακατασκευή να γίνεται με τη βοήθεια μοντέλων. Για την περιγραφή των μοντέλων επιλέξαμε την *UML*. Αρχικά ορίσαμε ένα *PIR*, το οποίο αποτελεί τον κορμό για την διαδικασία της ανακατασκευής. Για το δικό μας *case study* παρουσιάσαμε και δύο *PSR*, τα οποία χρησιμεύουν στην περιγραφή της αρχικής και τελικής πλατφόρμας του συστήματός μας.

Το σύστημά μας για είσοδο απαιτεί μόνο τον πηγαίο κώδικα του παλαιού συστήματος, από τον οποίο παράγει τα τρία μοντέλα που αντιστοιχούν σε δύο *PSM* και ένα *PIM*. Ο σχεδιαστής ως εκ τούτου χρειάζεται μόνο να γνωρίζει τον αντίστοιχο ορισμό των αναπαραστάσεων του εκάστοτε μοντέλου (δηλαδή *PIM* και *PSMs*) για να μπορεί να τα χρησιμοποιήσει. Δηλαδή, δεν είναι αναγκαίο να είναι πλήρης γνώστης της αντίστοιχης τεχνολογικής πλατφόρμας.

Πριν αναφερθούμε στη χρησιμότητα των μοντέλων που παράγει το εργαλείο από πλευράς διαγραμμάτων, μπορεί να γίνει μία ποιοτική μελέτη. Αρχικά μπορεί να γίνει μία εκτίμηση της πολυπλοκότητας που αναμένουμε να έχει η ανακατασκευή του συστήματος με βάση τον αριθμό των στοιχείων που ανήκουν στα μοντέλα. Ο αριθμός αυτός υπολογίζεται είτε συνολικά στα στοιχεία των μοντέλων, είτε τα ξεχωρίζουμε ανά πακέτο. Με βάση τους αριθμούς αυτούς μπορεί να γίνει μία εκτίμηση για την καταλληλότητα της νέας τεχνολογικής πλατφόρμας που επιλέγουμε. Για παράδειγμα στο δικό μας *case study* μετρήσαμε από τα 26 αρχεία, που αποτελούσαν το σύστημα, το πλήθος των στοιχείων του κάθε μοντέλου. Στο παράρτημα παρουσιάζουμε τις μετρήσεις ανά αρχείο ξεχωριστά ενώ τα συνολικά αποτελέσματα τα παρουσιάζουμε στον παρακάτω πίνακα 11.

Πίνακας 11: ***Συνολική κατανομή των στοιχείων των μοντέλων του case study μας.***

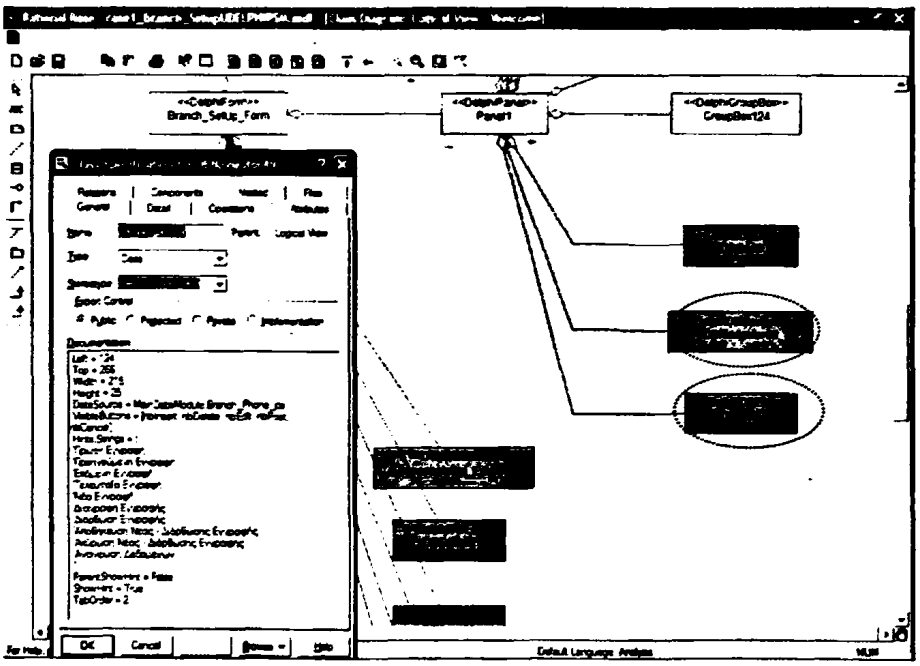


Πρώτη παρατήρηση στο παραπάνω σχήμα είναι η μεγάλη αύξηση των στοιχείων από το *Delphi PSM* στο *PIM*. Έτσι είναι άμεσα αντιληπτό ότι αρκετά στοιχεία του αρχικού μοντέλου διασπώνται σε μία σύνθεση στοιχείων. Από τις μετρήσεις ανά πακέτο αντιλαμβανόμαστε ότι πολλά στοιχεία που προστίθενται αφορούν τη διεπαφή με τον χρήστη. Αυτό που έχει περισσότερο σημασία είναι η μετάβαση από το *PIM* στο *target PSM* της πλατφόρμας που έχουμε επιλέξει. Βλέπουμε ότι ένα μεγάλο μέρος των *composite presentation* στοιχείων του *PIM* αντιστοιχίζονται σε *business* του *PSM*. Ο σχεδιαστής εξετάζοντας τις αντιστοιχίες (*mappings*) τις ενότητας 4.1.3 ώστε να δει τι προκαλεί τη μετακίνηση αυτή, διαπιστώνει ότι η πλατφόρμα που έχει επιλεγεί δεν έχει έτοιμα στοιχεία που να κάνουν τυπικές εργασίες. Αποτέλεσμα αυτού είναι ότι στο κόστος έργου θα υπάρχει μία επιβάρυνση στο πλήθος των προγραμματιστικών ωρών που χρειαζόμαστε. Σε ακραία περίπτωση θα μπορούσε κάποιος να αποφανθεί για την καταλληλότητα

της πλατφόρμας που επιλέγουμε. Μία τέτοια διαπίστωση είναι επικίνδυνη να γίνει. Αυτό οφείλεται για παράδειγμα στην επαναλαμβανόμενη διάσπαση των ίδιων στοιχείων. Δηλαδή είναι πιθανό να κατασκευάσουμε μία φορά κάτι στη νέα πλατφόρμα, και έπειτα να χρησιμοποιηθεί αυτούσια σε όλες τις παρόμοιες περιπτώσεις στη νέα πλατφόρμα.

Στην δική μας περίπτωση παρουσιάζεται αρκετές φορές το ενδεχόμενο της επαναχρησιμοποίησης. Επιλέγουμε να δείξουμε ένα παράδειγμα που προέρχεται από τη δική μας πρακτική εφαρμογή. Θα το χρησιμοποιήσουμε και στην συνέχεια για την χρηστικότητα από την μεριά του προγραμματιστή.

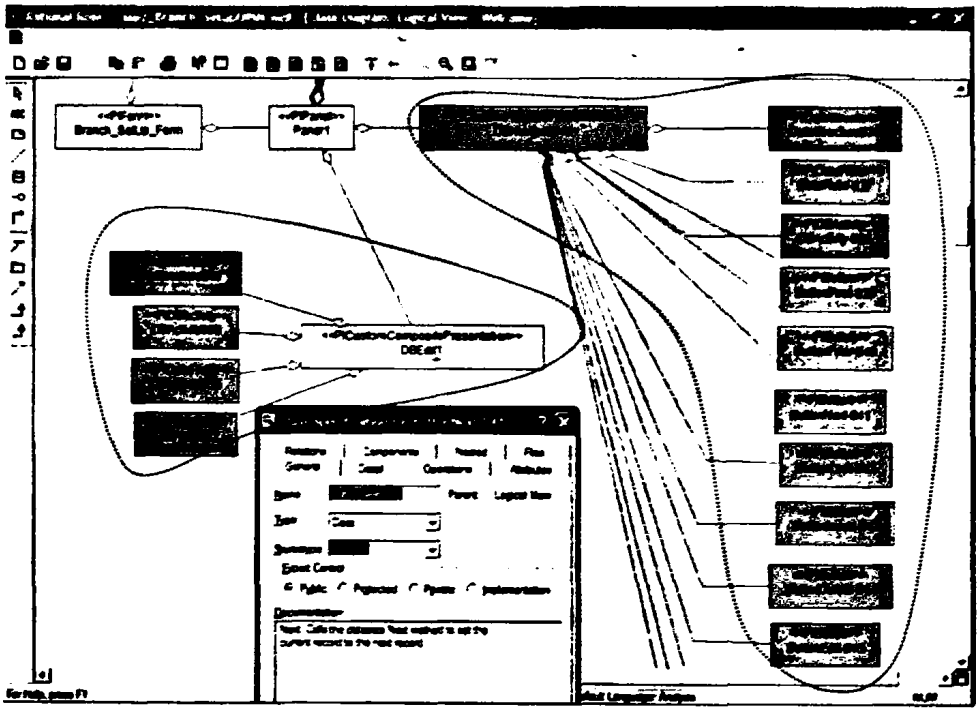
Εάν στην κατοχή μας έχουμε πολλά *PSRs* ικανά να περιγράψουνε το δοθέν παλαιό σύστημα, δηλαδή οι αντίστοιχες τεχνολογικές πλατφόρμες μπορούνε να υλοποιήσουν το δοθέν σύστημα, τότε μπορούμε να συγκρίνουμε ποιοτικά τα παραγόμενα τελικά *PSMs* τους. Μπορεί δηλαδή ο σχεδιαστής να κάνει μία εκτίμηση για την καταλληλότερη τεχνολογική πλατφόρμα για την ανακατασκευή. Μπορεί φυσικά με σχετικά μικρό επιπλέον κόστος να κατασκευάσουμε περισσότερες τελικές υλοποιήσεις.



Εικόνα 2: ***Μοντέλο που περιγράφει Delphi PSM όπως προέκυψε από το case study.***

Παρουσιάζουμε δύο εικόνες 2 και 3 οι οποίες περιγράφουν το ίδιο κομμάτι του *BIS* αλλά σε διαφορετικά μοντέλα. Στην πρώτη περίπτωση είναι σε *Delphi PSM*, ενώ στη δεύτερη σε *PIM*.

Παρατηρούμε ότι υπάρχουν στοιχεία τα οποία όταν χρησιμοποιηθούν για να κατασκευάσουμε το αντίστοιχο *PIM*, διασπώνται και παράγουν το μοντέλο της Εικόνας 3. Συγκεκριμένα βλέπουμε κάποια από τα παραπάνω στοιχεία τα *DBNavigator49* και *DBEdit1* να χαρακτηρίζονται από *stereotypes* τα οποία κατά την εφαρμογή του *abstraction pattern* παράγουν μία πληθώρα από στοιχεία. Πολλές τέτοιες περιπτώσεις παράγουν τα νούμερα του πίνακα 11. Αυτό που αξίζει να τονιστεί εδώ, είναι ότι ο κώδικας που θα κατασκευαστεί στο νέο σύστημα από κάποιον προγραμματιστή για να καλύψει τα στοιχεία που αφορούν το *DBNavigator49* θα επαναχρησιμοποιηθεί, με μικρές αλλαγές, και για άλλα στοιχεία του τελικού μοντέλου. Δηλαδή στοιχεία του *PIM* τα οποία έχουν προέλθει, από την εφαρμογή σε στοιχεία που χαρακτηρίζονται από ίδιο *stereotype*, ίδια *abstraction patterns* (παράδειγμα στο *DelphiDBNavigator*).



Εικόνα 3: ***Μοντέλο που περιγράφει PIM όπως προέκυψε από το μοντέλο της Εικόνας 2.***

Εκτός από την αξιολόγηση της τεχνολογικής πλατφόρμας, μπορεί να ενδιαφέρει τον σχεδιαστή να κάνει μία εκτίμηση κόστους του έργου. Τα μόνα στοιχεία τα οποία μπορεί να χρησιμοποιήσει για το σκοπό αυτό είναι αριθμοί που σχετίζονται άμεσα με τον κώδικα του συστήματος. Τέτοια είναι το πλήθος των μεθόδων και των γραμμών του *platform dependent code* (*PDC*) αλλά και του *platform independent code* (*PIC*). Στο δικό μας *case study* το σύστημα είχε σε 26 αρχεία 20569 γραμμές *PDC* και 11368 γραμμές *PIC*, όπως αναλυτικά παρουσιάζονται στον πίνακα 12. Εν γένη

ένα μεγάλο μέρος του *PDC* (δεν υπάρχει συγκεκριμένο νούμερο) μπορεί να χρησιμεύσει για την παραγωγή των ***skeleton*** αρχείων ή τα ***attributes*** που υπάρχουν στον *PDC* θα χρησιμεύσουν στον προγραμματιστή ως πληροφορία για να συμπληρώσει τα ***skeleton*** αρχεία. Ο προγραμματιστής θα πρέπει επίσης να προβεί στην μετατροπή του *PIC* στη νέα πλατφόρμα.

Πίνακας 12: ***Μέγεθος κώδικα ανά αρχείο και πλήθος μεθόδων.***

Αρχείο	*PDC(lines)*	*Methods*	*PIC(lines)*	Αρχείο	*PDC(lines)*	*Methods*	*PIC(lines)*
1	3491	31	325	14	343	1	105
2	402	7	116	15	198	3	10
3	646	2	19	16	524	8	96
4	234	1	17	17	2038	0	0
5	606	1	8148	18	457	3	66
6	73	3	10	19	574	3	115
7	2084	13	1655	20	577	3	222
8	2131	5	56	21	139	1	8
9	143	1	3	22	478	2	22
10	409	0	0	23	572	2	21
11	719	13	140	24	329	2	87
12	316	3	23	25	63	0	0
13	233	6	73	26	2790	2	31

Η χρησιμότητα του *PSM*, που περιγράφει το σύστημα με την παλαιά τεχνολογική πλατφόρμα, ταιριάζει σε περιπτώσεις που δεν θέλουμε να κάνουμε μεταφορά του νέου συστήματος σε μία άλλη τεχνολογική πλατφόρμα, αλλά απλά να κάνουμε κάποιες αλλαγές πάνω στο ήδη υπάρχον σύστημα. Αρκετές φορές ερχόμαστε αντιμέτωποι με περιπτώσεις, όπου χρειάζεται να συντηρήσουμε ένα σύστημα για το οποίο δεν μας παρέχεται καμιά πληροφορία. Το παραπάνω αναφερθέν *PSM* μπορεί να μας βοηθήσει στον εντοπισμό των αλλαγών που θέλουμε να πραγματοποιήσουμε. Επιπρόσθετα μπορεί να μας βοηθήσει στον εντοπισμό συνεπειών από επικείμενες αλλαγές που αποσκοπούμε να κάνουμε. Σε ακραίες περιπτώσεις μπορεί επίσης να χρησιμοποιηθεί για βελτιστοποίηση ενός συστήματος.

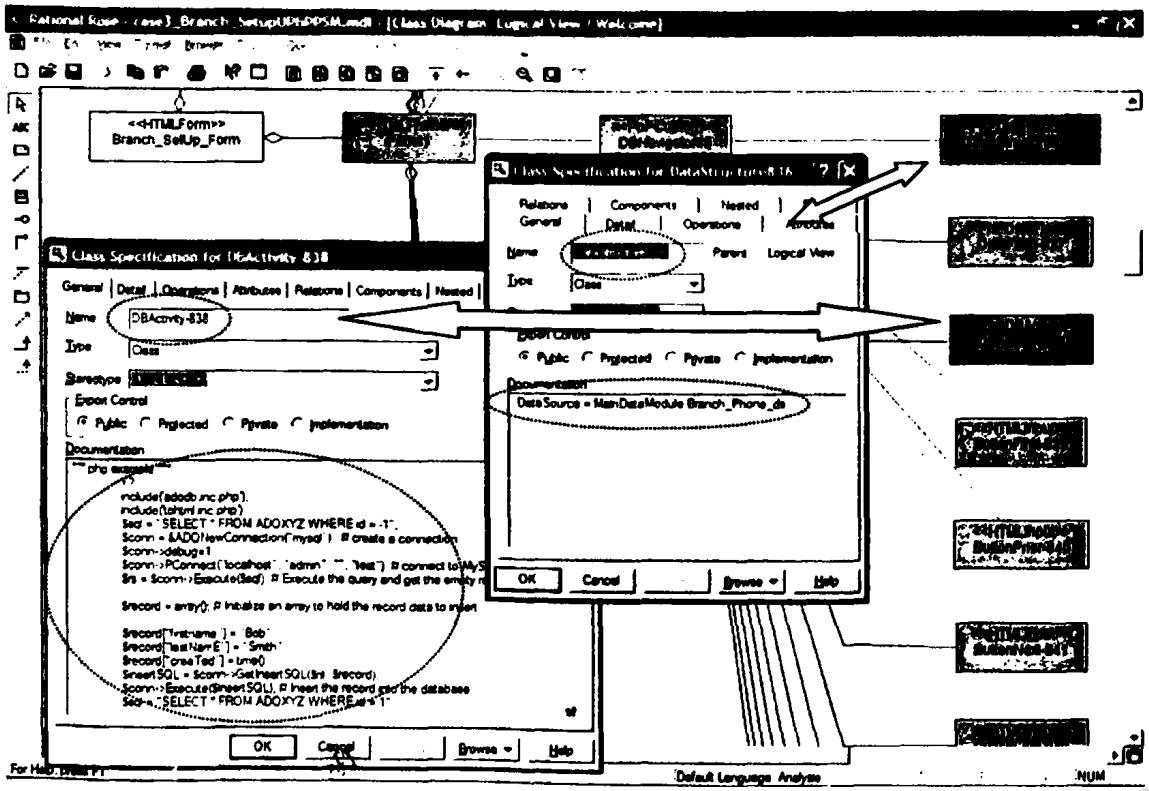
Αυτά που χρησιμοποιούμε περισσότερο για την ανακατασκευή, είναι τα *PIM* και *PSM*. Το *PIM* προσφέρεται περισσότερο για κάποιον σχεδιαστή, ο οποίος επιθυμεί να εξετάσει την αρχιτεκτονική του συστήματος. Αυτό οφείλεται κυρίως στο στάδιο της πρόσθιας κατασκευής. Κατά την πρόσθια κατασκευή πολλά στοιχεία του *PIM* μετατρέπονται σε στοιχεία τα οποία δεν αποκαλύπτουν άμεσα περί τίνος πρόκειται. Για παράδειγμα είδαμε τα *PINetworkConnection, PIDiskConnection, PITechnologyConnection* κ.α. να αναπλάθονται στο ίδιο στοιχείο *PhPCustom* (μία από τις αιτίες που προκάλεσε την αύξηση των ***business*** στοιχείων στο τελικό *PSM* του πίνακα 11).

Βέβαια, εάν είναι άμεσα εξοικειωμένος με την τεχνολογική πλατφόρμα θα μπορεί να έχει μία σαφώς καλύτερη εικόνα για τις ιδιαιτερότητες του συστήματος.

Συνήθως οι σχεδιαστές βρίσκουν δυσκολίες να κατανοήσουν τα *PSRs* και αυτό διότι μπορούν μεν να διαβάσουν τις απεικονίσεις, αλλά το πιθανότερο είναι πως τους λείπει η εμπειρία σε θέματα που σχετίζονται με τον τρόπο υλοποίησης μίας τεχνολογικής πλατφόρμας. Τα *PSRs* είναι κυρίως χρήσιμα για τους προγραμματιστές, που θα αναλάβουν την υλοποίηση του νέου συστήματος.

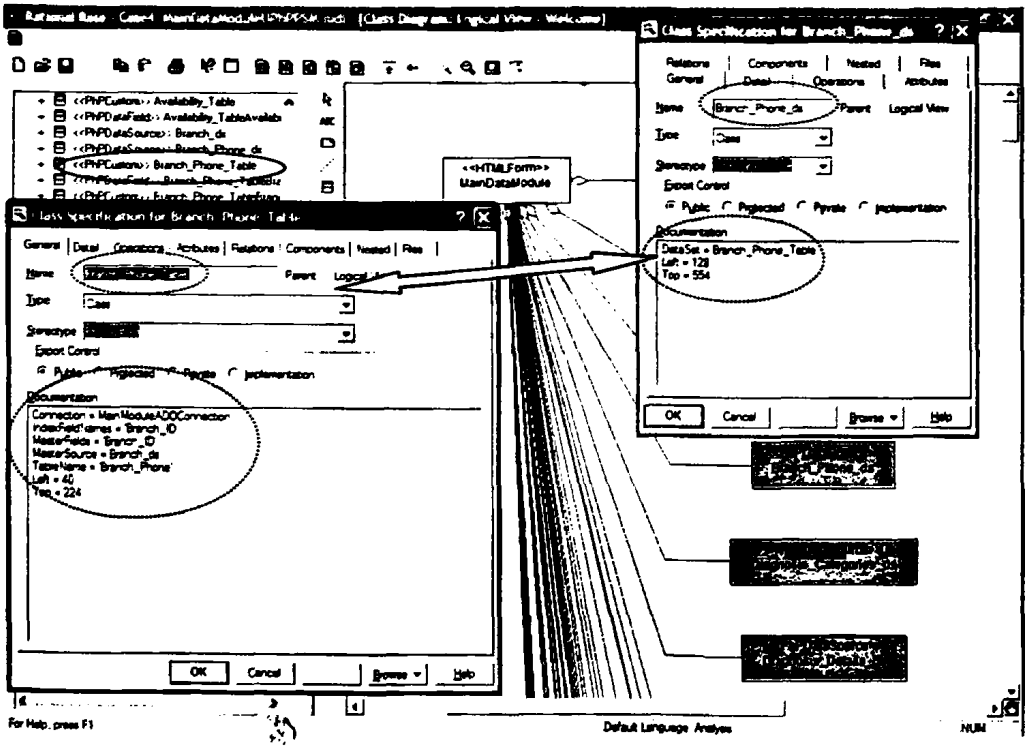
5.3 Χρηστικότητα του συστήματός μας για τον προγραμματιστή

Οι προγραμματιστές χρησιμοποιούν μόνο τα παραγόμενα μοντέλα και την πληροφορία που προσθέσαμε, υπό τη μορφή χαρακτηριστικών (*attributes*) σε στοιχεία αλλά και σε *skeleton files*, κατά τη ροή εργασιών της πρόσθιας κατασκευής. Στη δική μας περίπτωση επιλέξαμε να προσφέρουμε τα *skeleton* αρχεία ενσωματωμένα στο μοντέλο που εμφανίζεται από το *Rational Rose*.



Εικόνα 4: ***Μοντέλο που περιγράφει PhP –HTML PSM όπως προέκυψε από το μοντέλο της Εικόνας 3.***

Σαν παράδειγμα για την χρήση του συστήματος συνεχίζουμε αυτό της προηγούμενης παραγράφου, δηλαδή της Εικόνας 3. Ο προγραμματιστής έχει την εμπειρία να καταλάβει από τεχνικής άποψης και ένα *PIM* της Εικόνας 3 αλλά και ένα *PSM* όπως αυτό της Εικόνας 4. Επιλέγοντας ο προγραμματιστής να δει τις ιδιότητες ενός στοιχείο του μοντέλου μπορεί να δει στοιχεία τα οποία έχουν προέλθει από τον αρχικό *PDC* αλλά και στοιχεία που έχουν να κάνουν με ***skeleton files***. Στο παράδειγμα της Εικόνας 4 βλέπουμε ένα στοιχείο με όνομα *DBActivity-838* να είναι τύπου *PhPDBActivity* και να δίδεται πληροφορία για το συγκεκριμένο *stereotype* υπό την μορφή "*Documentation*". Στην Εικόνα 4 έχουμε επίσης το στοιχείο *DataStructure836* το οποίο στο πεδίο "*Documentation*" έχει μία παράμετρο η οποία έχει προέλθει από τον *PDC* του αρχικού συστήματος. Η παράμετρος αναφέρει "DataSource = MainDataModule.Branch_Phone_ds", το οποίο σημαίνει ότι τα δεδομένα του *DataStructure836* θα προέλθουν από το στοιχείο "Branch_Phone_ds" που βρίσκεται στο αρχείο "MainDataModule". Έτσι ανοίγοντας το αντίστοιχο αρχείο εμφανίζεται το μοντέλο της Εικόνας 5. Στις παραμέτρους του "Branch_Phone_ds" βλέπουμε ότι το στοιχείο βλέπουμε ότι το "Branch_Phone_Table" είναι αυτό που προκαλεί την άντληση δεδομένων από κάποια πηγή. Εξετάζοντας τις παραμέτρους του στοιχείου αυτού, διαπιστώνουμε έμμεσα ότι πρόκειται για μία αίτηση των δεδομένων ενός πίνακα με όνομα "Branch_Phone" και γίνεται χρήση της σύνδεσης "MainModuleADOConnection".



Εικόνα 5: ***Από το μοντέλο της Εικόνας 4 εξετάζουμε τις ιδιότητες του MainDataModule.Branch_Phone_ds.***

Ο προγραμματιστής μπορεί να θεωρήσει σκόπιμο να ανατρέξει στο *Delphi PSM* για περισσότερες πληροφορίες, όπου θα διαπιστώσει ότι το "Branch_Phone_Table" προήλθε από ένα *ADOTable* στοιχείο. Όταν το στοιχείο έχει κάποια *events* τα οποία προκαλούν την κλήση κάποιων μεθόδων, ο προγραμματιστής αρκεί να πάει στην επιλογή *"Operations"* του στοιχείου και να δει το αντίστοιχο "*Documentation*" της μεθόδου που τον ενδιαφέρει, όπως έγινε στην Εικόνα 1 (παράγραφος 4.2).

Σύνοψη

Το σύστημά μας και οι δυνατότητες του

Για το σύστημα που περιγράψαμε στην παράγραφο 5.1 κατασκευάσαμε μία συλλογή από εργαλεία. Τα εργαλεία αυτά παρέχουν τη δυνατότητα της αυτόματης παραγωγής *PSM* μοντέλων και *skeleton code* κομμάτια του τελικού συστήματος με χρήση του πηγαίου *Delphi* κώδικα του παλαιού συστήματος.

Για την γραφική αναπαράσταση εκμεταλλευτήκαμε ένα από τα εργαλεία μοντελοποίησης που χρησιμοποιούνται σήμερα, το *Rational Rose* [L1]. Εμείς παράγουμε αρχεία τύπου *mdl* (το *format* των αρχείων που χρησιμοποιεί το *Rose*) για κάθε ένα από τα μοντέλα ξεχωριστά. Δηλαδή τα αντίστοιχα *PSM* και *PIM* για κάθε φόρμα.

Το σύστημά μας αποτελείται από τα εξής υποσυστήματα:

- Ανάγνωση των αρχείων και διαχωρισμός τους σε *Platform depended* και *Platform independent* τμήματα. Μετατροπή και διάσπασή τους σε μορφή η οποία έχει οριστεί για την χρήση των παραπάνω από τα επόμενα εργαλεία.
- Κατασκευή του μοντέλου *PSM* που περιγράφει το αρχικό *BIS*.
- Με βάση τις αντιστοιχήσεις, που έχουν οριστεί, παράγουμε το πιο αφηρημένο μοντέλο του *PSM* δηλαδή το *PIM*
- Παράγουμε το μοντέλο *PSM* με βάση το *PIM*. Το μοντέλο αυτό περιέχει έννοιες της τεχνολογικής πλατφόρμας που έχουμε επιλέξει για την τελική υλοποίηση του συστήματος.
- Πρόσθεση στο *PSM* τμήματα του πηγαίου κώδικα της τελικής υλοποίησης του συστήματος.

Το σύστημα κατασκευάστηκε έτσι ώστε να μπορούμε εύκολα να προσαρμόζουμε τις απαιτήσεις μας σύμφωνα με τις εκάστοτε τεχνολογικές πλατφόρμες που επιλέγουμε προς χρησιμοποίηση. Στο κεφάλαιο 4 αναφέραμε ήδη λεπτομέρειες για το πως μπορεί κάποιος ορίσει τις παραμέτρους που τον ικανοποιούν. Στην επόμενη παράγραφο αναφερόμαστε λεπτομερειακά στην χρήση και στους περιορισμούς που έχει η τωρινή μορφή του συστήματός μας.

Χρήση και περιορισμοί του συστήματος

Όπως τα περισσότερα *COTS* έτσι και η *Delphi* για κάθε φόρμα, η οποία αποτελεί το κύριο αντικείμενο στο οποίο τοποθετούνται όλα τα υπόλοιπα στοιχεία, έχει ένα ζεύγος αρχείων εκ των οποίων το πρώτο έχει περιγραφές των αντικειμένων που χρησιμοποιεί, ενώ το δεύτερο περιέχει τον κώδικα που έχει γραφεί από τους προγραμματιστές μαζί με κάποια συμπληρωματικά στοιχεία για τα αντικείμενα. Έτσι όλο το *platform independent code* βρίσκεται στο δεύτερο αρχείο και τα υπόλοιπα συνθέτουν το *platform dependent code.*

Η εφαρμογή μας, όπως την περιγράψαμε στην παράγραφο 4.1.2, διαβάζει ένα αρχείο τύπου *dpr*, το οποίο περιέχει τα στοιχεία που αφορούν ένα συγκεκριμένο *project* υλοποιημένο σε *Delphi* και τα αρχεία που το συνθέτουν. Έπειτα, για κάθε τέτοιο αρχείο, κατασκευάζουμε αυτόματα τα αρχεία που περιγράφουν τα *PSM* και *PIM* καθώς και τα αντίστοιχα *mdl* αρχεία.

Η παραμετροποίηση μπορεί να γίνει στις συσχετίσεις που θα υπάρχουν ανάμεσα σε δύο αντικείμενα, δηλαδή εάν το αντικείμενο Α θα συνδέεται με το Β με μία απλή συσχέτιση ή θα κληρονομεί η μία την άλλη, αλλά και στους πίνακες που θα χρησιμοποιήσουμε για την μετατροπή του ενός μοντέλου σε ένα άλλο μοντέλο. Περισσότερα για τον τρόπο παραμετροποίησης, τα οποία είναι τεχνικής φύσης, αναφέραμε στο κεφάλαιο 4.

Όσον αφορά τους περιορισμούς οι βασικοί είναι δύο. Οι εξαρτήσεις των στοιχείων στα μοντέλα μας μπορούν να σημειωθούν μόνο σε όσες ανήκουν στην ίδια φόρμα. Μπορεί να υπάρχουν και εξαρτήσεις σε αντικείμενα τα οποία βρίσκονται σε διαφορετικές φόρμες, δεν φαίνονται όμως άμεσα από τα μοντέλα που παράγουμε (σε επόμενη παράγραφο δίνεται ένα παράδειγμα τέτοια περίπτωσης). Επίσης τα *PDC* έχουν διασπαστεί σε αντικείμενα τα οποία συνθέτουν το *business* πακέτο των αντικειμένων. Τα αντικείμενα αυτά στα μοντέλα μας εμφανίζονται ως εξαρτώμενα από κάποιο αντικείμενο του *presentation* ή *data* πακέτου. Ενδέχεται όμως, στις λειτουργίες που αυτά διενεργούν κατά την εκτέλεσή τους (στο κώδικα που έχει εισάγει ο προγραμματιστής δηλ *pdc*), να χρησιμοποιούν άλλα στοιχεία. Οι εξαρτήσεις αυτές δεν εμφανίζονται στο τελικό μοντέλο.

Συμπεράσματα και μελλοντική εξέλιξη

Στην παρούσα εργασία εξετάσαμε λεπτομερώς το γενικό πρόβλημα της μετατροπής ενός *business information system* σε ένα καινούργιο σύστημα, το οποίο να κάνει χρήση διαφορετικής τεχνολογικής πλατφόρμας. Η κύρια συνεισφορά μας στην μελέτη αυτή είναι η πρόταση μιας μεθοδολογίας, αλλά και κατασκευή ενός εργαλείου το οποίο να πραγματοποιήσει την αναφερθείσα ανακατασκευή. Η προτεινόμενη μεθοδολογία αποτελείται από μία ροή εργασιών της αντίστροφης κατασκευής, η οποία στοχεύει στην αυτόματη ανάκτηση από τον πηγαίο κώδικα του υπάρχοντος συστήματος ενός μοντέλου. Το μοντέλο αυτό περιγράφει μέρος της αρχιτεκτονικής του συστήματός μας. Τα στοιχεία του μοντέλου είναι ανεξάρτητα της τεχνολογικής πλατφόρμας που έχουμε χρησιμοποιήσει (*PIM*). Προς επίτευξη του σκοπού αυτού έχουμε ορίσει μία συλλογή από *stereotypes*, τα οποία έχουν σαν βάση το *specification Enterprise Distributed Computing (EDOC) systems*. Τα *stereotypes* αυτά είναι ικανά να περιγράψουν τα βασικά δομικά στοιχεία για την περιγραφή των διάφορων πλατφόρμων, οι οποίες κάνουν χρήση των πιο διαδεδομένων *COTS* (σε εργαλεία της *Borland* όπως *C++*, *Delphi*, *J++* και της *Microsoft Visual Basic*, *Visual C++* κτλ).

Για να αξιολογήσουμε την παραπάνω προτεινόμενη μέθοδο κατασκευάστηκε ένα πρότυπο εργαλείο και το εφαρμόσαμε σε ένα υπάρχον σύστημα (*BIS*). Τα μοντέλα του εργαλείου μπορούν να είναι χρησιμεύσουν και σε ένα νέο *project manager* που έχει αναλάβει την ανακατασκευή, καθώς επίσης και για τους προγραμματιστές που θα περατώσουν το νέο έργο. Ο *project manager*, μπορεί πλέον, γνωρίζοντας μόνο τους ορισμούς του *PIM*, να έχει μία πολύ καλή εικόνα της έκτασης και της αρχιτεκτονικής του παλαιού συστήματος. Δηλαδή, ο *project manager* πλέον δεν χρειάζεται να γνωρίζει ο ίδιος ή κάποιος άλλος τις λεπτομέρειες για όλες τις εμπλεκόμενες τεχνολογικές πλατφόρμες. Αντίστοιχα, στους προγραμματιστές που θα πραγματοποιήσουν την ανακατασκευή, τους προσφέρεται όλη η διαθέσιμη πληροφορία σε μία δομημένη γραφική μορφή. Η πληροφορία αυτή εμφανίζεται στις παραμέτρους και στα σχόλια των διάφορων στοιχείων στα μοντέλα που παράγουμε. Σαν σχόλια θα επισυνάπτονται κομμάτια κώδικα του παλαιού συστήματος, όπως και κάποια *skeleton files* για το νέο σύστημα.

Εκτός αυτών, είναι δυνατό από τον νέο σχεδιαστή, να ανιχνευθούν πιο εύκολα και να διορθωθούν λάθη σχεδιαστικά του προϋπάρχοντος συστήματος. Θα μπορούσε επίσης να προβεί στην βελτιστοποίηση κάποιων υποσυστημάτων και να διακρίνει τα προβλήματα που θα προκύψουν από κάποιες πιθανές αλλαγές που θέλουμε να επιφέρουμε. Τέλος κατά την φάση της πρόσθιας κατασκευής μπορούν να ανιχνευθούν (όχι αυτόματα αλλά από τον προγραμματιστή εμπειρικά) παρόμοια μοντέλα και με αυτόν τον τρόπο να βρεθούν τα σημεία στα οποία μπορεί να προκύψει η επαναχρησιμοποίηση κώδικα.

Μέχρι σήμερα έχουν προταθεί διάφορες μέθοδοι οι οποίες εστιάζουν στην ανακατασκευή συστημάτων με την βοήθεια μοντέλων. Όμως έτοιμα εργαλεία που να εφαρμόζονται σε βαθμό που να συμπεριλαμβάνουν και *source-to-source* μετατροπή για τέτοιες τεχνολογικές πλατφόρμες δεν υπάρχουν. Για να καλυφθούν οι ανάγκες, τις περισσότερες φορές, η ανακατασκευή γίνεται με τον συνδυασμό εργαλείων που εκτελούν επιμέρους εργασίες. Μεγάλες εταιρίες που κατασκευάζουν λογισμικό και πλατφόρμες για την ανάπτυξη εφαρμογών, όπως για παράδειγμα *Microsoft, SUN* και *Borland*, δεν φαίνονται πρόθυμες να παρουσιάσουν μελλοντικά τέτοια εργαλεία. Θα ήταν άλλωστε παράλογο να περιμέναμε κάποιον από τους παραπάνω να δημιουργήσει εργαλεία που θα εξυπηρετούσαν ανταγωνιστές τους.

Η αναπτυχθείσα μέθοδός μας θα μπορούσε να γίνει αιτία για μελέτη για το πώς μπορούν να προσαρτηθούν υπάρχουσες τεχνικές, με σκοπό το καλύτερο παραγόμενο αποτέλεσμα. Για παράδειγμα να κατασκευαστεί μια γλώσσα που να είναι ικανή να περιγράψει τους πηγαίους κώδικες ή πως να χρησιμοποιηθεί κάποια από τις ήδη υπάρχουσες γλώσσες που έχουν προταθεί για αυτόν τον σκοπό. Τέτοιες προσεγγίσεις θα ήταν χρήσιμες για την περιγραφή των εργασιών που γίνονται στα αντικείμενα που ανήκουν στο ***business*** πακέτο και ειδικότερα των στοιχείων που περιέχουν ***platform independent*** κώδικα.

Ενδιαφέρον θα παρουσίαζε να ερευνηθεί, εάν θα ωφελούσε (δηλαδή θα είχε πρακτικά αποτελέσματα), να οριστεί ένα συντακτικό για ένα πιο ελεύθερο *pattern recognition*, κατά την ανάκτηση του *PSM* από τον πηγαίο κώδικα, το οποίο να είναι ικανό να καλύπτει όλες τις πιθανές περιπτώσεις.

Βιβλιογραφία

[Biggerstaff94] Ted J. Biggerstaff, Bharat G. Mitbander, and Dallas Webster. *Program Understanding and the Concept Assignment Problem.* Communications of the ACM, 37(5):72-83, May 1994.

[Canfora96] Canfora, G. et al., 1996. *An Improved Algorithm for Identifying Objects in Code.* In Software Practice and Experience, Vol. 26, No. 1, pp. 25-48.

[Chidamber 94] Shyam R. Chidamber and Chris F. Kemerer. *A metrics suite for object-oriented design.* IEEE Transactions on Software Engineering, 20(6):476{493, June 1994.

[Chikofsky90a] Chikofsky, Cross. *Reverse Engineering and Design Recovery - a Taxonomy*, IEEE Software, January 1990

[Chikofsky90b] Chikofsky. *Using CASE to leverage existing systems.* Computing Canada, October 25, 1990

[Cimitile99] Cimitile, A. et al., 1999. *Identifying Objects in Legacy Systems Using Design Metrics.* In Journal of Systems and Software, Vol. 44, No. 3, pp. 199-211.

[Cordy90] Cordy, J.R. Eliot, N.L. Robertson, M.G. *TuringTool: a user interface to aid in the software maintenance task.* This paper appears in: Software Engineering, IEEE Transactions on Publication Date: Mar 1990 On page(s): 294-301 Volume: 16, Issue: 3

[DeLucia97] De Lucia, A. et al., 1997. *Migrating Legacy Systems Towards Object-Oriented Platforms.* In Proceedings if the IEEE Conference on Software Maintenance (ICSM'97), pp. 222-229.

[DeMarco82] Tom DeMarco. *Controlling Software Projects : Management, Measurement and Estimation.* Prentice-Hall, 1982.

[Deursen99] A. van Deursen, P. Klint and C. Verhoef. *Research Issues in Software Renovation.* In J.-P. Finance, editor, Proceedings Fundamental Approaches to Software Engineering (FASE99), pages 1-23. Lecture Notes in Computer Science, Springer-Verlag, 1999. Invited paper.

[FAMOOS] Holger Bar, Markus Bauer, Oliver Ciupke, Serge Demeyer, Stephane Ducasse, Michele Lanza, Radu Marinescu, Robb Nebbe, Oscar Nierstrasz, Michael Przybilski, Tamar Richner, Matthias Rieger, Claudio Riva, Anne-Marie Sassen, Benedikt Schulz, Patrick Steyaert, Sander Tichelaar, Joachim Weisbrod, The FAMOOS *Object-Oriented Reengineering Handbook*, 1999
[http://iamwww.unibe.ch/~scg/Archive/famoos/handbook/4handbook.pdf]

[Fernando94] Fernando B. Abreu and Rogerio Carapuca. *Candidate metrics for object-oriented software within a taxonomy framework.* The Journal of Systems and Software, 26(1):87{96, 1994.

[Gall95] Gall, H., Klosch, R, Mittermeir, R., *Object-oriented re-architecturing.* Proceedings of the 5th European Softwar Engineering Conference (ESEC'95), Sitges. Spain September 1995, Lecture Notes in Computer Science 989, Springer, 1995, 499-519

[Harsu98] Harsu M., *A Survey Of Object Identification In Software Re-Engineering* (1998)

[Harsu00] Harsu, M., *Identifying object-oriented features from procedural software*. Proceedings of Nordic Workshop on Programming Environment Research (NWPER'2000), Lillehammer, Norway, May 2000, 143-158.

[Horwitz 90] Horwitz, S., Reps, T., and Binkley, D., *Interprocedural slicing using dependence graphs*. ACM Transactions on Programming Languages and Systems 12, 1 (January 1990), 26-60.

[Jackson94] D. Jackson and E. Rollins *A New Model of Program Dependences for Reverse Engineering*. Proceedings of the ACM Symposium on Foundations of Software Engineering, December 1994

[Jacobson91] Jacobson, J. and Lindstrom, F., 1991. *Reengineering of Old Systems to an Object-Oriented Architecture*. In Proceedings of the ACM Conference on Object-Oriented Programming Systems, Languages, and Applications(OOPSLA'91), pp. 340-350.

[Linos94] Linos, P., Aubet, L. Dumas, Y. Helleboid, P. Lejeune, and P. Tulula, *Visualizing Program Dependencies*. Software-Practice and Experience Journal, 1994, Vol. 24(4), 387-403.

[Lionel99] Lionel C. Briand, Sanro Morasca, and Victor R. Basili. *Defining and validating measures of object-based high-level design*. IEEE Transactions of Software Engineering, 25(5):722{743, September/October 1999.

[Liu90] Liu, S., S. and Wilde, N., 1990. *Identifying Objects in a Conventional Procedural Language: An Example of Data Design Recovery*. In Proceedings if the IEEE Conference on Software Maintenance (ICSM'90), pp. 266-271.

[Murphy96] Gail C. Murphy, David Notkin: *Lightweight Lexical Source Model Extraction*. ACM Trans. Softw. Eng. Methodol. 5(3): 262-292 (1996)

[Newcomb 95] Newcomb, P. and Kotik, G. (1995). *Re-engineering Procedural into Object-Oriented Systems*, In Proceedings of the Second Working Conference on Reverse Engineering. Toronto, Canada, July 1995, pp. 237--249.

[Ong93] Ong, C. and Tsai, W., T., 1993. Class and Object Extraction from Imperative Code. In Journal of Object Oriented Programming, Vol. 6, No. 1, pp. 58-68.

[Paul94] S. Paul A. Prakash *Framework for Source Code Search Using Program Patterns* IEEE Transactions on Software Engineering
Volume 20 , Issue 6 (June 1994) table of contents Pages: 463 - 475 Year of Publication: 1994

[Ping03] Ping, Y. et al., 2003. *Migration of Legacy Web Applications to Enterprise Java Environments – Net.Data to JSP Transformation*. In Proceedings of the ACM Conference Centre for Advanced Studies Conference on Collaborative Research, pp. 223-237.

[Rade90] Rade Adamov and Lutz H. Richter. *A proposal for measuring the structural com-*

plexity of programs. Journal of Systems and Software, 23(12):55{70, 1990.

[Scott97] Scott A. Whitmire. *Object-Oriented Design Measurement*. John Wiley & Sons, 1997.

[Shari97] Shari L. Pfleeger, Ross Jeffery, Bill Curtis, and Barbara Kitchenham. *Status report on software measurement*. IEEE Software, 14(2):33{43, 1997.

[Schwanke91] Schwanke, R., W., 1991. *An Intelligent Tool for Reengineering Software Modularity*. In Proceedings of the 13th IEEE-ACM-SIGSOFT International Conference on Software Engineering (ICSE'91), pp. 83-92.

[Soley00] Soley R. and the OMG Staff Strategy Group, 2000, Model-Driven Architecture. White Paper, Object Management Group.

[Swanson76] Swanson, E. B., *The dimensions of maintenance*, Proceedings of the 2nd International Conference on Software Engineering (ICSE '76), San Francisco, California, October 1976, 492-297

[Thomas89] Thomas McCabe and Butler Charles W. *Design complexity measurement and testing*. Communications of the ACM, 32(12):1415{1425, 1989.

[Venkatesh91] G. A. Venkatesh, *Programming language design and implementation*.Conference on Programming Language Design and Implementation. Proceedings of the ACM SIGPLAN 1991 conference on

[Victor90] Victor R. Basili, *Viewing Maintenance as Reuse-Oriented Software Development*, IEEE Software, pp 19-15, January 1990.

[Weiser81] Weiser, Mark. *Program Slicing*. Proc. 5th Int. Conf. on Software Eng. New York: IEEE, 1981, 439-449.

[William88] William G. Bail and Marvin V. Zelkowitz. *Program complexity using hierarchical abstract computers*. Computer Languages, 13(3/4):109{133, March/April 1988.

Ακρωνύμια

BIS	Business Information Systems
COTS	Components Of the Shelves
PIM	Platform Independent Model
PSM	Platform Specific Model
PIR	Platform Independent Representation
PSR	Platform Specific Representation
MDA	Model Driven Architecture
OMG	Object Management Group
EDOC	Enterprise Distributed Computing Systems
API	Application Programming Interface
UML	Unified Modeling Language™

Links

[L1] http://www.rational.com

[L2] http://www.zlw-ima.rwth-aachen.de/forschung/publications/software_re engineering.html

[L3] http://homepages.cwi.nl/~arie/papers/

[L4] http://www.cs.txstate.edu/~gh10/cs5391/se_readings.html

[L5] http://www.iam.unibe.ch/~scg/Archive/famoos/handbook/

[L6] http://www2.umassd.edu/SWPI/slicing/slicing.html

[L7] http://www.omg.org/

[L8] http://www.omg.org /technology/documents/formal/corba_iiop.htm

[L9] http://msdn.microsoft.com/library/default.asp?url=/library/enus/cossdk/htm/ pgservices_events_5x4j.asp

[L10] http://www.omg.org/technology/documents /formal/components.htm

[L11] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cossdk/htm/ pgcontexts_1 p0z.asp

[L12] http://java.sun.com/j2ee/

[L13] http://java.sun.com/products/jsp/

[L14] http://www.php.net/

[L15] http://www.omg.org/technology/documents/modeling_spec _catalog. htm

[L16] http://medlab.cs.uoi.gr/

[L17] http://www.mysql.com/

[L18] http://www.pittarese.com/Auburn/cse625/case.htm (CASE tools)

[L19] http://www.gmu.edu/depts/survey/

[L20] http://pear.php.net/packages.php

[L21] http://www-306.ibm.com/software/rational/mda/index.html

[L22] http://www.swebok.org/

Παράρτημα

Enterprise Distributed Computing (EDOC) systems

Component Collaboration Architecture (CCA) Metamodel

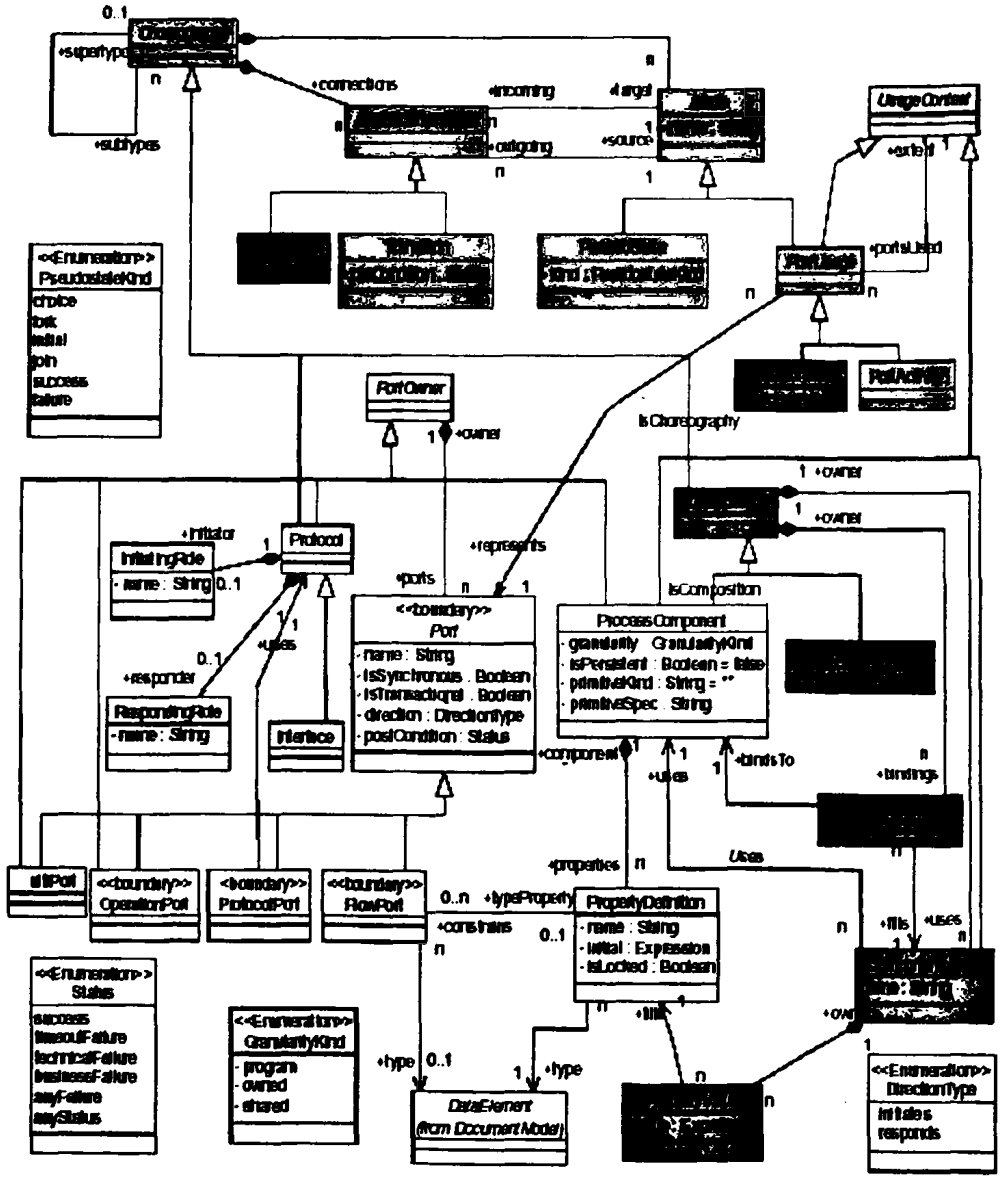


Figure 1: CCA Major Elements

Figure 1 is a combined model of the major elements of the CCA component specification defined below.

Structural Specification

The structural specification represents the physical structure of the component contract, defining the component and its ports.

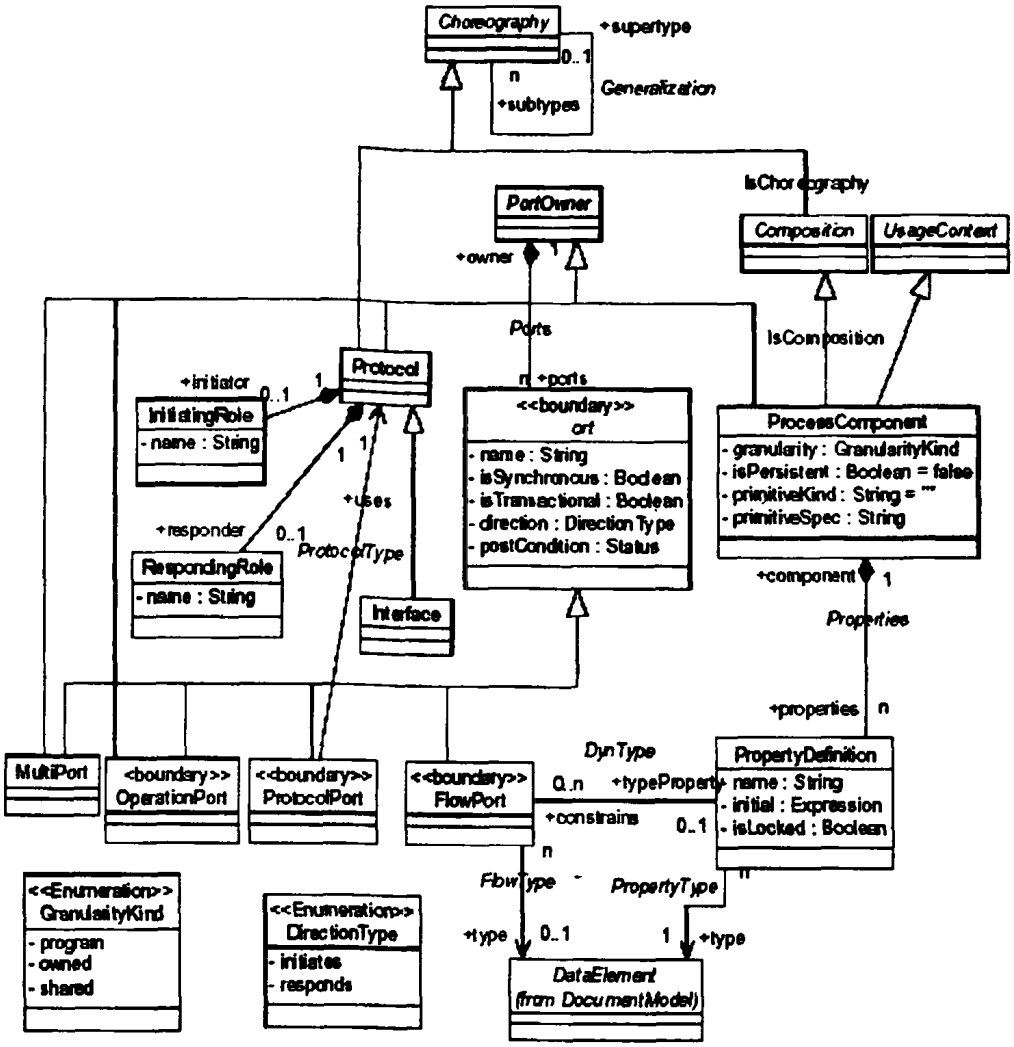


Figure 2 Structural Specification Metamodel

A **ProcessComponent** represents the contract for a component that performs actions it "does something". A ProcessComponent may define a set of **Ports** for interaction with other ProcessComponents. The ProcessComponent defines the external contract of the component in terms of ports and a **Choreography** of port activities (sending or receiving messages or initiating sub-protocols). At a high level of abstraction a ProcessComponent can represent a business partner, other ProcessComponents represent business activities or finer-grain capabilities.

The contract of the ProcessComponent is realized via **ports**. A port defines a point of interaction between ProcessComponents. The simpler form of port is the **FlowPort**, which may produce or consume a single **data type**. More complex interactions between components use a **ProtocolPort**, which refers to a **Protocol**, a complete "conversation" between components. Protocols may also use other protocols as subprotocols. Protocols, like ProcessComponents, are defined in terms of the set of ports they realize and the choreography of interactions across those ports. A protocol may optionally define names for the initiating and responding roles.

ProcessComponents may have **Property Definitions**. A property definition defines a configuration parameter of the component, which can be set, when the component is used.

The behavior of a ProcessComponent may be further specified by its **composition**, the composition shows how other components are used to define and implement the composite component. The specification of the ProcessComponent and protocol may include

Choreography to sequence the actions of multiple ports and their associated actions. The actions of each port may be **Choreographed.** Composition and Choreography are defined in their own sections.

A ProcessComponent may have a **supertype** (derived from Choreography). One common use of supertype is to place abstract ProcessComponents within compositions and then produce separate realizations of those components as subtype composite or primitive components, which can then be substituted for the abstract components when the composition is used, or even at runtime.

An **Interface** represents a standard object interface. It may contain **OperationPorts**, representing call-return semantics, and FlowPorts – representing one-way operations.

A **MultiPort** is a grouping of ports whose actions are tied together. Information must be available on all sub-ports of the MultiPort for any action to occur within an attached component.

An **OperationPort** defines a port which realizes a typical request/response operation and allows ProcessComponents to represent both document oriented (FlowPort) and method oriented (OperationPort) subsystems

ProcessComponent

Semantics
A ProcessComponent represents an active processing unit – it does something. A ProcessComponent may realize a set of Ports for interaction with other ProcessComponents and it may be configured with properties. Each ProcessComponent defines a set of ports for interaction with other ProcessComponents and has a set of properties that are used to configure the ProcessComponent when it is used.

The order in which actions of the Process Component's ports do something may be specified using Choreography. The choreography of a ProcessComponent specifies the external temporal contact of the ProcessComponent (when it will do what) based on the actions of its ports and the ports in protocols of its ports.

UML base element(s) in the Profile and Stereotype
Classifier Stereotyped as <<ProcessComponent>>

Fully Scoped name
ECA::CCA::ProcessComponent

Owned by
Package

Extends
Composition (indicating that the ProcessComponent may be composed of other ProcessComponents and that its ports may be choreographed).

Package (Indicating that a ProcessComponent may own the specification of other elements).

UsageContext (Indicating that the ProcessComponent may be the context for PortUsages representing the activities of its ports).

Properties

Granularity
A *GranularityKind* which defines the scope in which the component operates. The values may be:

Program	the component is local to a program instance (default)
Owned	the component is visible outside of the scope of a particular program but dedicated to a particular task or session which controls its life cycle.
Shared	the component is generally visible to external entities via some kind of distributed infrastructure.

Specializations of CCA may define additional granularity values.

UML Representation
Tagged value

isPersisten
t
Indicates that the component stores session specific state across interactions. The mechanisms for management of sessions are defined outside of the scope of CCA.

UML Representation

Tagged value

primitiveKind
Components implementation includes additional implementation semantics defined elsewhere, perhaps in an action language or programming language. If the component has an implementation specification primitiveKind specifies the implementation specific type, normally the name of a programming language. If primitive kind is blank, the composition is the full specification of the components implantation – the component is not primitive.

UML Representation
Tagged value

primitiveSpec
If primitiveKind has a value, primitiveSpec identifies the location of the implementation. The syntax of primitiveKind is implementation specific.

UML Representation
Tagged value

Related elements

Ports (via "PortOwner")
"Ports" is the set of Ports on the ProcessComponent. Each port provides a connection point for interaction with other components or services and realizes a specific protocol. The protocol may be simple and use a "FlowPort" or the protocol may be complex and use a "ProtocolPort" or an "OperationPort". If allowed by its protocol, a port may send and receive information. UML Representation
Required Aggregation Association from Port (Ports) Supertype (zero or one) , Subtypes (any number) A ProcessComponent may inherit specification elements (ports, properties & states (from Choreography) from a supertype. That supertype must also be a ProcessComponent. A subtype component is bound by the contract of its supertypes but it may add elements, override property values and restrict referenced types. A component may be substituted by a subtype of that component.

UML Representation
Generalization

Properties (Any number)
To make a component capable of being reused in a variety of conditions it is necessary to be able to define and set properties of that component. Properties represents the list of properties defined for this component.

UML Representation
Classifier.feature referencing an attribute.

Constraints
A process component may only inherit from another process component.

Port

Semantics
A port realizes a simple or complex conversation for a ProcessComponent or protocol. All interactions with a ProcessComponent are done via one of its ports. When a component is instantiated, each of its ports is instantiated as well, providing a well-defined connection point for other components.Each port is connected with collaborative components that speak the same protocol. Multi-party conversions are defined by components using multiple ports, one for each kind of party.

Business Example: Flight reservation Port

UML base element(s) in the Profile and Stereotype
Class (abstract)

Fully Scoped name
ECA::CCA::Port

Owned by
ProcessComponent or Protocol via PortOwner

Extends
None

Properties

isTransactional
Indicates that interactions with the component are transactional & atomic (in most implementations this will require that a transaction be started on receipt of a message). Non-transactional components either maintain no state or must execute within a transactional component. The mechanisms for management of transactions are defined outside of the scope of CCA.

UML Representation
Tagged Value

isSynchronous
A port may interact synchronously or asynchronously. A port that is marked as synchronous is required to interact using synchronous messages and return values.

UML Representation
Tagged Value

name
The name of the port. The name will, by default, be the same as the name of the protocol role or document type it realizes.

UML Representation
ModelElement::name

Direction
Indicates that the port will either initiate or respond to the related type. An initiating port will send the first message. Note that by using ProtocolPorts a port may be the initiator of some protocols and the responder to others. The values of DirectionKind may be:

Initiates this port will initiate the conversation by sending the first message.
Responds this port will respond to the initial message and (potentially) continue the conversation.

UML Representation
Tagged Value and stereotype of "Owner" relation.

PostCondition
The status of the conversation indicated by the use of this port. This status may be queried in the postCondition of a transition.

UML Representation
Tagged Value

Related elements

"Owner" ProcessComponent or Protocol (Exactly One via PortOwner)
A Port specifies the realization of protocol by a ProcessComponent. This relation specifies the ProcessComponent that realizes the protocol.

UML Representation
Required aggregate association (Ports). This association will have a stereotype of "initiates" or "responds" to indicate "direction."

Constraints
None

FlowPort

Semantics
A Flow Port is a port which defines a data flow in or out of the port on behalf of the owning component or protocol.

UML base element(s) in the Profile and Stereotype
Class stereotyped as <<FlowPort>>

Fully Scoped name
ECA::CCA::FlowPort

Owned by
PortOwner

Extends

Port

Properties
None

Related elements

type
The type of data element that may flow into our out of the port.
UML Representation
Required relation

TypeProperty
The type of information sent or received by this port as determined by a configurable property. The expression must return a valid type name. This is used to build generic components that may have the type of their ports configured. If *type* and *typeProperty* are both set then the property expression must return the name of a subtype of *type*.

UML Representation
Tagged value containing the name of the property attribute.

Constraints
None

ProtocolPort

Semantics
A protocol port is a port which defines the use of a protocol A protocol port is used for potentially complex two-way interactions between components, such as is common in B2B protocols. Since a protocol has two "roles" (the initiator and responder), the direction is used to determine which role the protocol port is taking on.

UML base element(s) in the Profile and Stereotype
Class stereotyped as <<ProtocolPort>>

Fully Scoped name
ECA::CCA::ProtocolPort

Owned by
PortOwner

Extends
Port

Properties
None

Related elements

uses
The protocol to use, which becomes the specification of this port's behavior.

UML Representation
Generalization – the ProtocolPort inherits the Protocol.

Constraints
None

OperationPort

Semantics
An operation port represents the typical call/return pattern of an operation. The OperationPort is a PortOwner which is constrained to contain only flow ports, exactly one of which must have its direction set to "initiates". The other "responds" ports will be the return values of the operation.

UML base element(s) in the Profile and Stereotype
Operation (no stereotype)

Note1: The type of the "initiates" flow port will be the signature of the operation. Each attribute of the type will be one parameter of the operation.
Note2: Owned flow ports of postCondition==Success and direction=="responds" will be a return value for the operation. All other flow ports where direction=="responds" will correspond to an exception.

Fully Scoped name
ECA::CCA::OperationPort

Owned by
PortOwner (Protocol or ProcessComponent)

Extends
Port and PortOwner

Properties
None

Related elements

Ports (Via PortOwner)
The flow ports representing the call and returns.

UML Representation
Initiates ports – signature of the operation
Responds ports – return values of the operation.

Constraints
As a PortOwner, the OperationPort:

- May only contain FlowPorts.
- Must contain exactly one flow port with direction set to "responds."
- Must contain exactly one flow port with direction set to "initiates" (the call).

MultiPort

Semantics
A MultiPort combines a set of ports which are behaviorally related. Each port owned by the MultiPort will "buffer" information sent to that port until all the ports within the MultiPort have received data, at this time all the ports will send their data.

UML base element(s) in the Profile and Stereotype
Class stereotyped as <<MultiPort>>

Fully Scoped name
ECA::CCA::MultiPort

Owned by
PortOwner

Extends
Port & PortOwner

Properties
None

Related elements

Ports (Via PortOwner)
The flow ports owned by the MultiPort.

UML Representation
Required aggregation association

Constraints
Owned ports will not forward data until all sub-ports have received data.

Protocol

Semantics
A protocol defines a type of conversation between two parties, the initiator and responder. One protocol role is the initiator of the conversation and the other the responder. However, after the conversation has been initiated, individual messages and sub-protocols may by initiated by either party. The ports of a protocol are specified with respect to the responder.
Within the protocol are sub-ports. Each port contained by a protocol defines a subaction of that protocol until, ultimately, everything is defined in terms of FlowPorts.
A Protocol is also a choreography, indicating that activities of its ports (and, potentially their sub-ports) may be sequenced using an activity graph.
A protocol must be used by a two ProtocolPorts to become active.
The protocol specifies the conversation between two ProcessComponents (via their ports). Each component that is using that protocol must use it from the perspective of the "initiating role" or the "responding role." Each of these components will use every port in the protocol, but in complementary directions.
For example, a protocol "X" has a flow port "A" that initiates a message and a flow port "B" that responds to a message. Component "Y" which responds to protocol "X" will also receive "A" and initiate "B". But, Component "Z" which initiates protocol "X" will initiate message "A" and respond to message "B" – thus initiating a protocol will "invert" the directions of all ports in the protocol.

UML base element(s) in the Profile and Stereotype
Class stereotyped as <<Protocol>>

Fully Scoped name
ECA::CCA::Protocol

Owned by
Package

Extends
Choreography – Indicating that the contract of the protocol includes a sequencing of the port activities.
Package – Indicating that the protocol may contain the specification of other model elements (Most probably other protocols or documents).

Properties

None
Ports (Via PortOwner)
The ports which define the sub-actions of the protocol. For example, a "callReturn" protocol may have a "call" FlowPort and a "return" FlowPort.

UML Representation
Required aggregate association

Initiator
The role which sends the first message in the protocol. Note that this is optional, in which case the initiating role name will be Initiator".

UML Representation
Required relation

Responder
The role which receives the first message in the protocol. Note that this is optional, in which case the responding role name will be "Responder".

UML Representation
Required relation

Constraints
None

Interface

Semantics
An interface is a protocol constrained to match the capabilities of the typical object interface. It is constrained to only contain OperationPorts and FlowPorts and all of its ports must respond to the interaction (making interfaces one-way). Each OperationPort or FlowPort in the Interface will map to a method. A ProtocolPort which initiates the Interface will call the interface. A ProtocolPort which Responds will implement the interface.

UML base element(s) in the Profile and Stereotype
Classifier (Usually Interface, but any classifier will do)

Fully Scoped name
ECA::CCA::Interface

Owned by
Package

Extends
Protocol

Properties
None

Related elements
Ports (Via Protocol & PortOwner)
The ports which define the sub-actions of the protocol. For example, a "callReturn" protocol may have a "call" flowport and a "return" port.

Initiator (Via Protocol)
The role which calls the interface. Note that this is optional, in which case the initiating role name will be "Initiator". roles.

Responder (Via Protocol)
The role which implements the interface. Note that this is optional, in which case the responding role name will be "Responder".

Constraints
- The Ports related by the "Ports" association must; be of type OperationPort or FlowPort. have direction == "responds".

InitiatingRole

Semantics
The role of the protocol which will send the first message.

UML base element(s) in the Profile and Stereotype
Class stereotyped as <InitiatingRole>

Fully Scoped name
ECA::CCA::InitiatingRole

Owned by
Protocol

Extends
None

Properties

name
Role name

UML Representation
ModelElement::name

Related elements
Protocol
The protocol for which the role is being defined.
UML Representation
Required relation

Constraints
None

RespondingRole

Semantics
The role in the protocol which will receive the first message.

UML base element(s) in the Profile and Stereotype
Class stereotyped as <RespondingRole>

Fully Scoped name
ECA::CCA::RespondingRole

Owned by
Protocol

Extends
None

Properties
Name

UML Representation
ModelElement::name

Related elements

Protocol
The protocol for which the role is being defined.

UML Representation
Required relation

Constraints
None

PropertyDefinition

Semantics
To allow for greater flexibility and reuse, ProcessComponents may have properties which may be set when the ProcessComponent is used. A **PropertyDefinition** defines that such a property exists, its name and type.

UML base element(s) in the Profile and Stereotype
Attribute (No stereotype)

Fully Scoped name
ECA::CCA::PropertyDefinition

Owned by
ProcessComponent

Extends
None

Properties

name
Name of the property being modeled

UML Representation
ModelElement:name

initial
An expression indicating the initial & default value.

UML Representation
Attribute::initialValue

isLocked
The property may not be changed.

UML Representation
StructuralFeature::changeability

Related elements

component
The owning component
UML Representation
Classifier.feature referencing an attribute.ModelElement::namespace

type
The type of the property

UML Representation
StructuralFeature::type

Constraints
- If the "constrains" relation contains any links, the PropertyValue must contain the fully qualified name of a DataElement.

PortOwner

Semantics
An abstract meta-class used to group the meta-classes that may own ports: Process component, Protocol, OperationPort and MultiPort.

UML base element(s) in the Profile and Stereotype
None (Abstract)

Fully Scoped name
ECA::CCA::PortOwner

Owned by
None

Extends
None

Related elements

ports
The owned ports

UML Representation
Required relation

Constraints
None

Entity Metamodel

This section describes the entity meta-model. This model provides a basis for understanding the modeling concepts and their relationships. The next section describes the implementation of the model in UML.

Overview

The diagram, below, depicts the elements to be considered; those that are part of this profile specification are highlighted. Central to this model are Data Manager and its specializations; these are the core elements of the Entities profile. They encapsulate data and other components, exposing their functionality through ports.

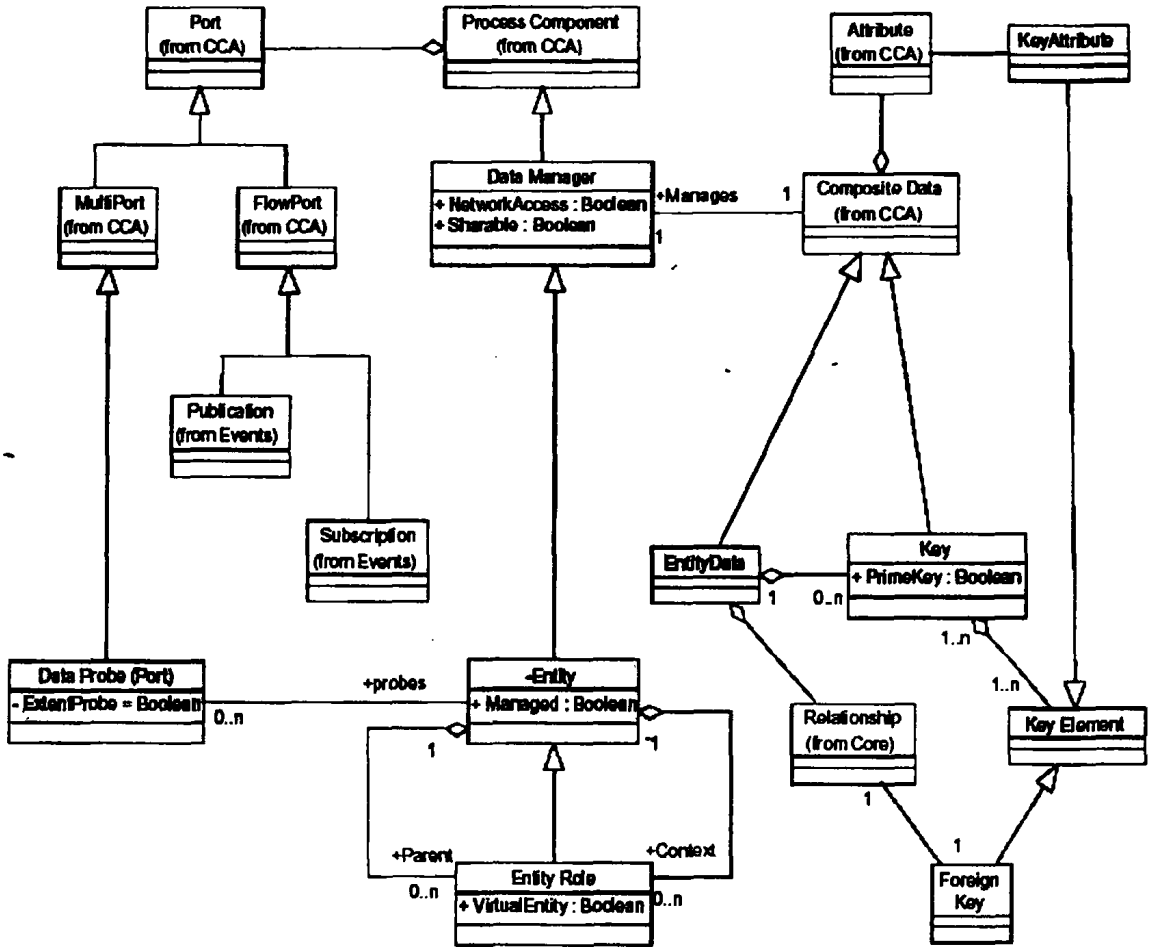


Figure 3 Entity Metamodel

Through ports components receive and respond to messages, publish and subscribe to events and expose state changes response to ad hoc requests to Data Probe ports. Entities represent the application domain. As components they encapsulate the functionality, state and relationships of domain concepts. Entity components incorporate Entity Data structures, which are the core elements of the information model.

Entity Package

This section describes the elements of the Entity metamodel in detail.

DataManager

Semantics

A Data Manager is a functional component that provides access to and may perform operations on its associated Composite Data (i.e., its state). The Data Manager defines ports for access to operations on the state data.

UML base element(s) in the Profile
Class

Fully Scoped name
EDOC::ECA::Entity::Data Manager

Owned by
Package

Properties

Network Access
A Boolean value which indicates if the Data Manager is intended to be accessible over the network.

Sharable
A Boolean value which indicates if the Data Manager can be shared by multiple transactions/sessions. A Data Manager that is not sharable is either transient or depends on a sharable Data Manager that contains it for persistence. For example, an address may not be sharable (although its state may be passed by value), but it can be persistent by association with a Customer that is sharable.

Related elements

Process Component
Data Manager inherits from Process Component and adds the quality of having associated state.

Composite Data
Composite Data defines the data structure that is encapsulated by the Data Manager.

Entity
Entity specializes Data Manager for representation of identifiable application domain things.

Constraints
N/A

EntityData

Semantics
Entity Data is the data structure that represents a concept in the business domain. It is equivalent to an entity in data modeling or a relation in a relational database. In a Data Manager or its specializations, such as Entity, it represents the state of an object. Entity Data has attributes (from Data Element) and relationships. The information viewpoint is a viewpoint on Entity Data elements.

UML base element(s) in the Profile
Class

Fully Scoped name
EDOC::ECA::Entity::EntityData

Owned by
Package

Properties
N/A

Related elements

Composite Data
Entity Data inherits from Composite Data and adds relationships.

Relationship
Describes an association between Entity Data elements.

Data Manager
A Entity Data element is incorporated in a Data Manager which gives it functionality and ports as a component.

Constraints
- Entity Data must have a prime Key that is unique within the extent of the Entity Data type (i.e., the type and all sub-types).
- Entity Data is managed by an Entity Data Manager.

Key

Semantics
A Key is a value that may be used to identify a Data Entity for some purpose. Generally, it will be a unique identifier within some context. A Key designated Prime Key = true is the key intended for unique identity of the Data Entity within the extent of the Data Entity type. A Key is composed of key elements which may be selected attribute values of the associated Data Entity or Foreign Keys. A Foreign Key is the key of a related Date Entity.

UML base element(s) in the Profile
Class

Fully Scoped name
EDOC::ECA::Entity::Key

Owned by
Entity Data

Properties

Prime Key
A Boolean value that indicates if the Key is intended to be the primary unique identity of the associated Entity Data type. If so, the value must be unique within the extent of the identifiable type.

Related elements

Composite Data
A Key is a specialization of Composite Data.

Entity Data
A Key describes an identifier of an Entity Data type.

Key Element
A Key Element is one segment of a Key, which is either a reference to an attribute of the associated Data Entity or a reference to the key of an associated Data Entity.

Constraints

- If Key is Prime Key = true, then the value must be unique within the extent of the associated Entity Data type and its sub-types.
- The attributes that are incorporated into the key must be immutable.
- The Key Elements that comprise the key have an immutable sequence.

Key Element

Semantics
A Key Element is one segment of a Key, which is either a reference to an attribute of the associated Data Entity or a reference to the key of an associated Data Entity.

UML base element(s) in the Profile
Class

Fully Scoped name
EDOC::ECA::Entity::Key Element

Owned by
Key

Properties
N/A

Related elements

Key
The Key in which the Key Element appears.

Key Attribute
A Specialization of Key Element that references an attribute in the associated Entity Data..

Foreign Key
A specialization of Key Element that references the Key of an Entity Data structure that is related to the Entity Data identified by the containing Key.

Constraints
N/A

Foreign Key

Semantics
A Foreign Key is a Key Element that is the value of a related Entity Data structure.The subject Entity Data structure derives its identity, in part, from the related Entity Data structure. For example, the line item of an order may be uniquely identified by the line number and the key of the associated order. The Foreign Key element references the relationship in order to identify the related Entity Data that contains the Foreign Key value.

UML base element(s) in the Profile
Class

Fully Scoped name
EDOC::ECA::Entity::Foreign Key

Owned by
Key

Properties
N/A.

Related elements

Key Element
Foreign Key is a specialization of Key Element.

Relationship
The associated relationship identifies the Entity Data from which the Foreign Key value is obtained..

Constraints
- If the associated Key has PrimeKey = true, then he relationship used to obtain the Foreign Key value must be immutable.

Key Attribute

Semantics
A Key Attribute identifies an attribute of the associated Entity Data that is included as an element of the Entity Data key. The value of the attribute becomes an element of the key of an instance of the Entity Data type.

UML base element(s) in the Profile
Class

Fully Scoped name
EDOC::ECA::Entity::Key Attribute

Owned by
Key

Properties
N/A.

Related elements

Key Element
Key Attribute inherits from Key Element.

Attribute
Attribute is the Attribute of the Entity Data structure that is to be incorporated as an element of the containing Key..

Constraints

- If the containing Key is designated PrimeKey = true, then the Attribute values that are incorporated into the key must be immutable.

Entity

Semantics

An Entity is an object representing something in the real world of the application domain. It incorporates Entity Data that represents the state of the real world thing, and it provides the functionality to encapsulate the Entity Data and provide associated business logic.

An Entity instance has identity derived from the Key of its associated Entity Data.

Entity is the abstract super type of all identifiable application domain elements. This includes Entities that have a collection of rules to operate on the state of related entities. It also includes Entities that incorporate process elements that act on other Entities. The rule set and process specializations introduce additional elements, but have the basic characteristics of being identifiable, having local state (Composite Data) often viewed as their "context," and having relationships to other Entities that they may act upon. If an Entity is managed, all instances of the type and its sub-types are known, each instance has unique identity, and the type can have operations and attributes associated with the extent (i.e., applicable to all instances). This is typically implemented as a type manager or "home" object that represents the extent.

UML base element(s) in the Profile

Class

Fully Scoped name

EDOC::ECA::Entity::Entity

Owned by

Package

Properties

In the list, below, only Managed is introduced as a property by Entity, but NetworkAccess and Sharable, inherited from Data Manager, are also discussed to clarify the implications.

Managed

A Boolean value that indicates if the Entity type is *managed*. If it is managed, then the implementation provides a mechanism for accessing the extent of all instances of the type and its sub-types and may provide a mechanism for dynamically applying rules to all instances. This typically is implemented as a "home" or "type manager."

NetworkAccessible

A Boolean value that indicates if the Entity is expected to be accessed over the network. This implies that it has a network interface (e.g., CORBA IDL). An Entity that is not NetworkAccessible can only be accessed over the network through an associated Entity that is NetworkAccessible.

Sharable

A Boolean value that indicates if the Entity can be shared by multiple, concurrent transactions or users. A Sharable Entity will enforce controls to serialize access by concurrent transactions.

An Entity that is not sharable may be instantiated for use by a particular user or transaction. It generally contains a copy of the primary Entity Data instance representing the real world thing. The primary Entity Data instance may be in a database and the copy is created to perform operations on the Entity Data. Alternatively, the Entity Data may be managed by an Entity that is sharable, but the copy is created so that processing can be localized on another server. In either case, it would be expected that the primary Entity Data would be locked when the copy is taken and released when the copy is deleted. Changes to the copy would likely be applied to the primary instance prior to removing the lock.

Entities that are not sharable may also be implemented as value objects, which are always passed by value over the network. While they may have unique identity by association with an identifiable Entity, they may not have a key that reflects this unique identity and their Entity Data does not carry its unique identity when passed by value.

An Entity that is sharable is expected to be persistent. An Entity that is not sharable may be persistent if it is incorporated in the state of a sharable Entity.

Related elements

DataManager

Entity inherits from DataManager and adds the requirement that its associated Composite Data is Entity Data. It also adds the ability to accept Data Probes and the ability to be Managed.

Entity Role

Entity Role inherits from Entity as a specialized representation of an Entity in a particular context. The Entity Role contains Entity Data that is associated with the parent Entity in the particular context. Entity Role is associated with another Entity that represents the context in which it applies. Thus the parent Entity might be a person, the Entity Role might be the person as an employee, and the context entity might be the employer.

An Entity may have many Entity Roles. Each Entity Role defines characteristics of the Entity in a particular context, such as person in the role of an employee within a corporation. An Entity may be the context for many Role Entities as a corporation is the context of many employees.

Data Probe
A Data Probe port is associated with an Entity that accepts requests to detect changes in the internal state of the Entity and forwards messages or events when the states of interest become true.

Constraints
- An Entity manages Entity Data, which may have a key and relationships.
- A managed Entity must have a Primary Key.
- A network Accessible Entity must have a Primary Key
- An Entity that is Sharable will serialize concurrent transactions that attempt to access its data.

Entity Role

Semantics
An Entity Role extends its parent Entity for participation in a particular context. An Entity may have a number of associated Entity Roles reflecting participation in multiple contexts. The Entity might have several Entity Roles of the same type at the same time, but each should be associated with a different context.
The context of an Entity Role is also represented by an Entity. The context could be a corporation where the parent is a person and the Entity Role is an employee. A context may have many entity roles of the same type or different types representing participation of different parent Entities for different purposes.

UML base element(s) in the Profile
Class

Fully Scoped name
EDOC::ECA::Entity::Entity Role

Owned by
Entity (context)

Properties

VirtualEntity
A Boolean value that indicates if the Entity Role incorporates and extends the primary interface of the parent Entity it represents, i.e., it can be used in place of the primary Entity.

Related elements
Entity
- Inheritance Entity Role inherits from Entity such that it functions as an entity but it derives its unique identity from the Entity it represents (i.e., a Foreign Key).
- Context association An Entity Role represents an Entity in a particular context. This association defines the context.
- Parent association An Entity Role represents an entity in a particular context. This association defines the parent Entity being represented.

Constraints
The parent entity of an entity role cannot be dynamically changed.

DataProbe

Semantics
A Data Probe port is associated with an Entity and accepts ad hoc requests to detect changes in the internal state of the Entity. The Data Probe then forwards messages or events when the states of interest become true until the request is removed. A Data Probe may serve many requests concurrently, producing various messages or events when the appropriate states occur.

UML base element(s) in the Profile
Class

Fully Scoped name
EDOC::ECA::Entity::Data Probe

Owned by

Entity

Properties

ExtentProbe
ExtentProbe = true indicates that requests apply to the extent of the associated entity as opposed to a particular instance. In implementation, an ExtentProbe would be associated with a "home" or "type manager."

Related elements

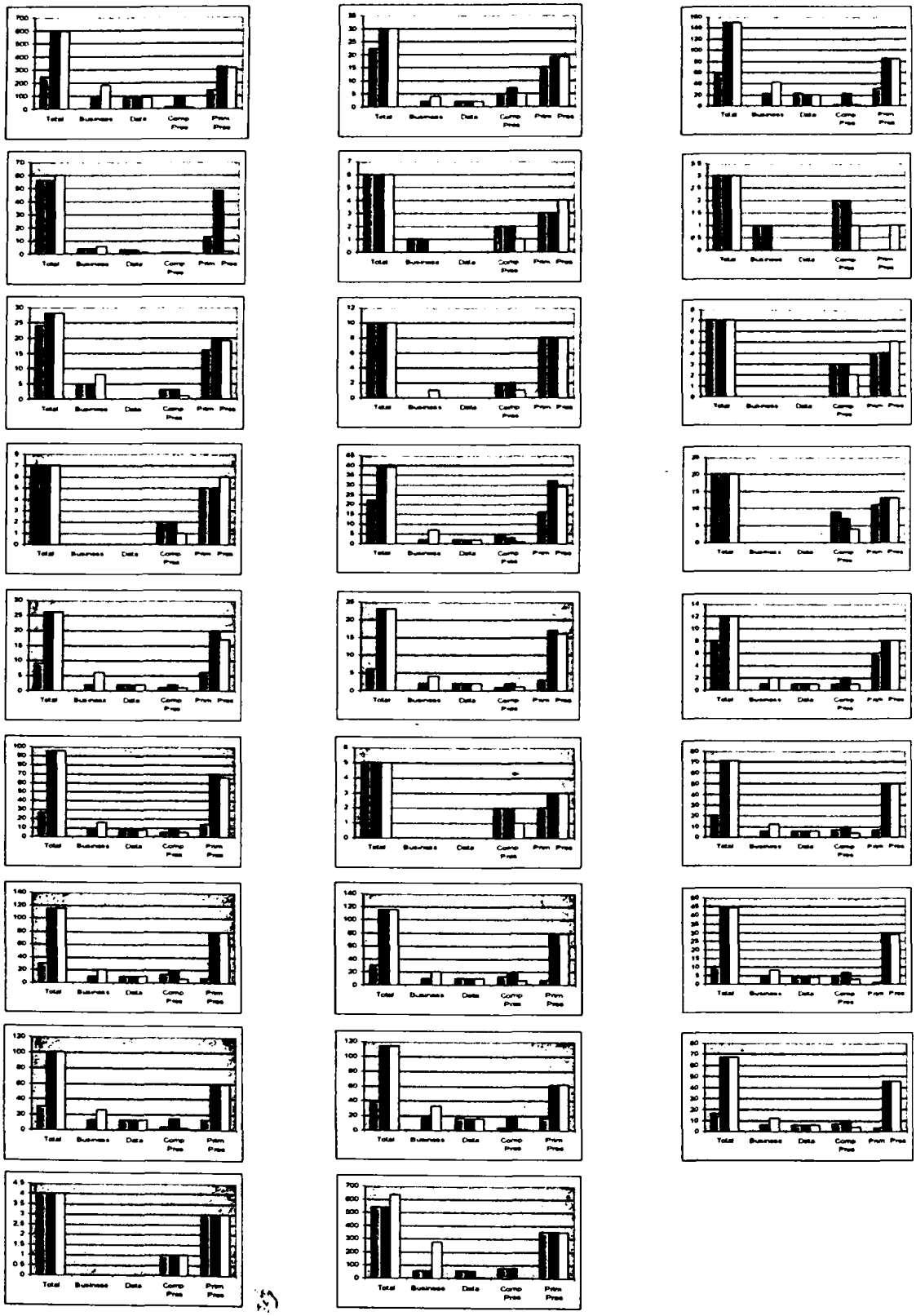
Multi Port
Data Probe inherits from Multi Port.

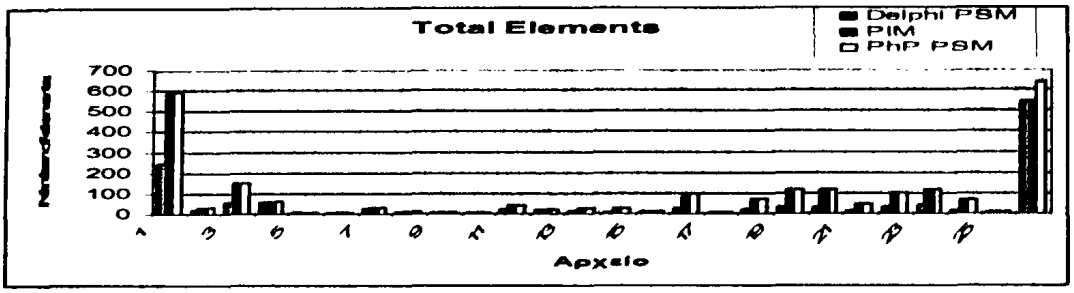
Entity
The Entity that will accept probe requests.

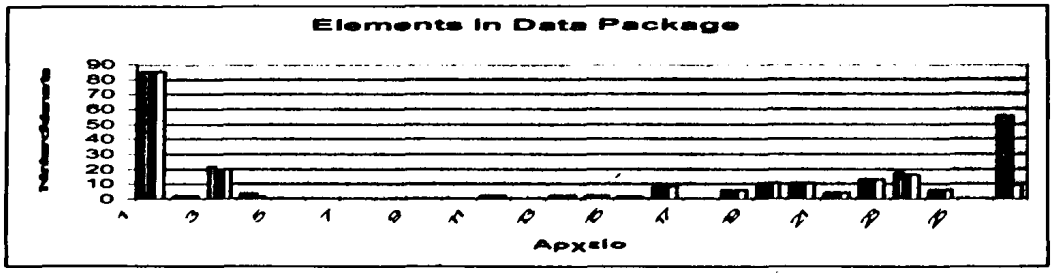
Constraints

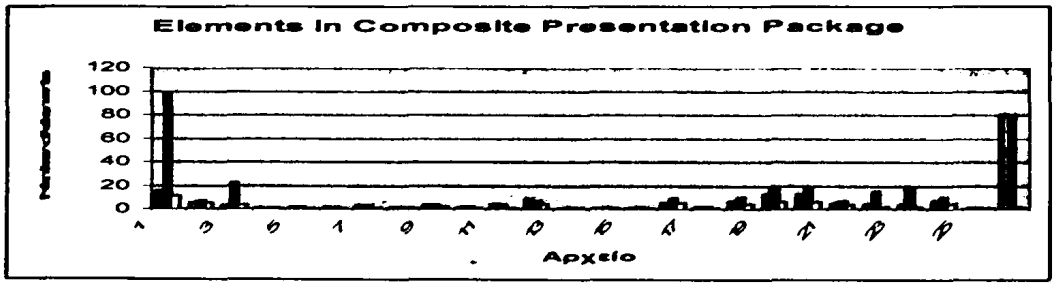
- DataProbes only emit messages (i.e., output only).
- DataProbe can only attach to an Entity with Managed = true..

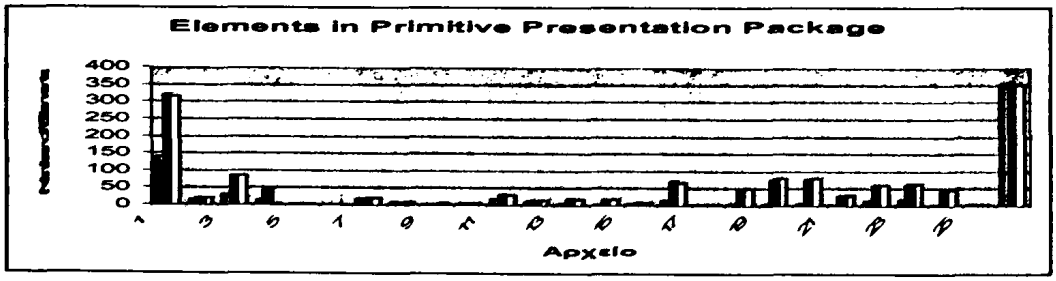
Μετρήσεις

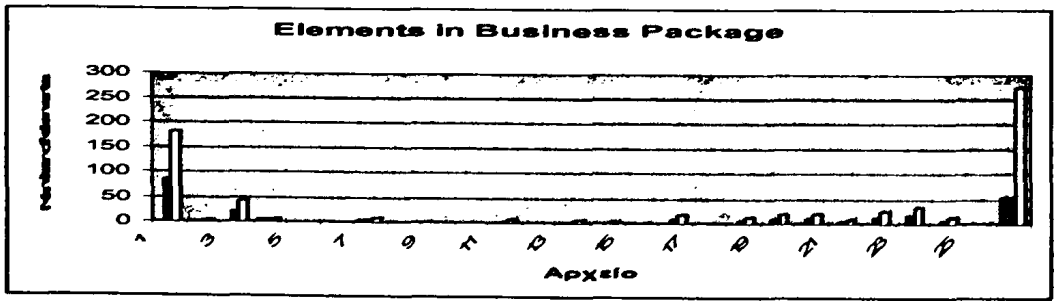












Αρχείο 1

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	241	1	85	15	140
PIM	590	86	85	99	320
PhP PSM	590	180	85	11	314

Αρχείο 2

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	22	0	2	5	15
PIM	30	2	2	7	19
PhP PSM	30	4	2	5	19

Αρχείο 3

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	56	0	22	3	31
PIM	149	22	20	22	85
PhP PSM	149	42	20	3	84

Αρχείο 4

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	56	4	3	1	13
PIM	56	4	3	1	48
PhP PSM	60	6	1	1	2

Αρχείο 5

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	6	1	0	2	3
PIM	6	1	0	2	3
PhP PSM	6	0	0	1	4

Αρχείο 6

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	3	1	0	2	0
PIM	3	1	0	2	0
PhP PSM	3	0	0	1	1

Αρχείο 7

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	24	5	0	3	16
PIM	28	5	0	3	20
PhP PSM	28	8	0	1	19

Αρχείο 8

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	10	0	0	2	8
PIM	10	0	0	2	8
PhP PSM	10	1	0	1	8

Αρχείο 9

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	7	0	0	3	4
PIM	7	0	0	3	4
PhP PSM	7	0	0	2	5

Αρχείο 10

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	7	0	0	2	5
PIM	7	0	0	2	5
PhP PSM	7	0	0	1	6

Αρχείο 11

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	22	0	2	4	16
PIM	39	2	2	3	32
PhP PSM	39	7	2	1	29

Αρχείο 12

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	20	0	0	9	11
PIM	20	0	0	7	13
PhP PSM	20	0	0	4	13

Αρχείο 13

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	9	0	2	1	6
PIM	26	2	2	2	20
PhP PSM	26	6	2	1	17

Αρχείο 14

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	6	0	2	1	3
PIM	23	2	2	2	17
PhP PSM	23	4	2	1	16

Αρχείο 15

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	8	0	1	1	6
PIM	12	1	1	2	8
PhP PSM	12	2	1	1	8

Αρχείο 16

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	27	0	8	5	14
PIM	95	8	8	9	70
PhP PSM	95	16	8	5	66

Αρχείο 17

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	5	0	0	2	2
PIM	5	0	0	2	3
PhP PSM	5	0	0	1	3

Αρχείο 18

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	20	0	6	7	7
PIM	71	6	6	10	49
PhP PSM	71	12	6	4	49

Αρχείο 19

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	30	0	10	13	7
PIM	115	10	10	18	77
PhP PSM	115	20	10	6	79

Αρχείο 20

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	30	0	10	13	7
PIM	115	10	10	18	77
PhP PSM	115	20	10	6	79

Αρχείο 21

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	10	0	4	5	1
PIM	44	4	4	7	29
PhP PSM	44	8	4	3	29

Αρχείο 22

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	30	0	13	4	13
PIM	100	13	13	15	59
PhP PSM	100	26	13	2	59

Αρχείο 23

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	38	0	17	4	17
PIM	113	17	16	19	61
PhP PSM	113	33	16	2	62

Αρχείο 24

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	16	0	6	7	3
PIM	67	6	6	10	45
PhP PSM	67	12	6	4	45

Αρχείο 25

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	4	0	0	1	3
PIM	4	0	0	1	3
PhP PSM	4	0	0	1	3

Αρχείο 26

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	541	54	55	80	352
PIM	541	54	55	80	352
PhP PSM	633	273	9	1	350

Συνολικά Total 5896

Elements	Total	Business	Data	Comp. Pres	Prim. Pres
Delphi PSM	1248	66	248	195	739
PIM	2276	256	245	348	1427
PhP PSM	2372	694	197	70	1411

	Delphi PSM			PIM			PhP PSM		
Δοχεία	Aggregation	Inheritance	Association	Aggregation	Inheritance	Association	Aggregation	Inheritance	Association
1	2	0	238	102	0	487	360	0	229
2	0	0	17	7	0	22	14	0	15
3	0	0	55	21	0	127	82	0	62
4	55	0	0	10	0	45	54	0	5
5	0	0	5	1	0	4	1	0	4
6	1	0	1	1	0	1	1	0	1
7	11	0	12	12	0	15	23	0	2
8	0	0	7	2	0	7	1	0	8
9	0	0	5	2	0	4	1	0	5
10	0	0	6	2	0	4	0	0	6
11	0	0	18	0	0	38	13	0	25
12	0	0	18	3	0	14	6	0	13
13	0	0	7	0	0	24	11	0	14
14	0	0	5	0	0	22	10	0	13
15	0	0	7	2	0	9	6	0	6
16	0	0	26	16	0	78	42	0	53
17	0	0	4	0	0	4	2	0	3
18	0	0	19	9	0	61	30	0	40
19	0	0	29	13	0	101	50	0	64
20	0	0	29	12	0	101	51	0	64
21	0	0	9	3	0	40	21	0	23
22	0	0	29	2	0	96	55	0	45
23	0	0	37	3	0	107	65	0	45
24	0	0	15	2	0	64	30	0	36
25	0	0	3	0	0	3	0	0	3
26	540	0	0	182	0	358	571	0	62

Πηγαίος κώδικας

Table of Contents

Table of contents

Delphi to Rational Rose Ευρετήριο δομών δεδομένων

Delphi to Rational Rose Δομές Δεδομένων

Ακολουθούνε οι δομές δεδομένων με σύντομες περιγραφές:

Delphi to Rational Rose Ευρετήριο Αρχείων

Delphi to Rational Rose Λίστα Αρχείων

Ακολουθεί μια λίστα όλων των αρχείων με σύντομες περιγραφές:

Delphi to Rational Rose Τεκμηρίωση Δομών Δεδομένων

dfmlist Αναφορά Δομής

dfmlistdfmlistdfmlistdfmlistΠεδία Δεδομένων

- **char objectname [MAX_WORD_LENGTH]**
- **char objecttype [MAX_WORD_LENGTH]**
- **dfmlist * parent**
- **dfmlist * next**
- **dfmlist * fson**
- **int x**
- **int y**
- **int quid**
- **int inherit**
- **int arg [4]**

Λεπτομερή Περιγραφή

Στη δομή αυτή αποθηκεύουμε όλα τα στοιχεία του μοντέλου

Ορισμός στη γραμμή 83 του αρχείου dfm2mdl.c.

Τεκμηρίωση Πεδίων

int dfmlist::arg[4]

Ορισμός στη γραμμή 86 του αρχείου dfm2mdl.c.

Αναφορά από append_assoc_to_ab(), και make_relations().struct dfmlist * dfmlist::fson

Ορισμός στη γραμμή 85 του αρχείου dfm2mdl.c.

Αναφορά από append_assoc_to_ab(), append_to_ab(), create_final_mdl(), free_list(), make_a_dfmnode(), make_relations(), και print_dfm_list().int dfmlist::inherit

Ορισμός στη γραμμή 86 του αρχείου dfm2mdl.c.

Αναφορά από append_assoc_to_ab(), make_relations(), write_node_a(), και write_node_b().struct dfmlist * dfmlist::next

Ορισμός στη γραμμή 85 του αρχείου dfm2mdl.c.

Αναφορά από append_assoc_to_ab(), append_to_ab(), free_list(), make_relations(), και print_dfm_list().char dfmlist::objectname[MAX_WORD_LENGTH]

Ορισμός στη γραμμή 84 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), write_assoc_a(), write_node_a(), και write_node_b().char dfmlist::objecttype[MAX_WORD_LENGTH]

Ορισμός στη γραμμή 84 του αρχείου dfm2mdl.c.

Αναφορά από make_relations(), write_node_a(), και write_node_b().struct dfmlist* dfmlist::parent

Ορισμός στη γραμμή 85 του αρχείου dfm2mdl.c.

Αναφορά από append_assoc_to_ab(), make_a_dfmnode(), make_relations(), write_assoc_a(), write_assoc_b(), write_node_a(), και write_node_b().int dfmlist::quid

Ορισμός στη γραμμή 86 του αρχείου dfm2mdl.c.

Αναφορά από write_assoc_a(), write_assoc_b(), write_node_a(), και write_node_b().int dfmlist::x

Ορισμός στη γραμμή 86 του αρχείου dfm2mdl.c.

Αναφορά από make_a_dfmnode(), write_assoc_b(), και write_node_b().int dfmlist::y

Ορισμός στη γραμμή 86 του αρχείου dfm2mdl.c.

Αναφορά από write_assoc_b(), και write_node_b().

Η τεκμηρίωση για αυτή η δομή δημιουργήθηκε απο το ακόλουθο αρχείο:

- **dfm2mdl.c**

line Αναφορά Δομής

linelinelinelineΠεδία Δεδομένων

- int **num_of_words**
- char **words** [MAX_WORDS_PER_LINE][MAX_WORD_LENGTH]

Λεπτομερή Περιγραφή

Η δομή αυτή χρησιμοποιείται για την αποθήκευση μίας γραμμής. Είναι χωρισμένη σε λέξεις

Ορισμός στη γραμμή 77 του αρχείου dfm2mdl.c.

Τεκμηρίωση Πεδίων

int line::num_of_words

- Ορισμός στη γραμμή 78 του αρχείου dfm2mdl.c.

Αναφορά από add_operation_text_old(), και get_next_line().char line::words[MAX_WORDS_PER_LINE][MAX_WORD_LENGTH]

Ορισμός στη γραμμή 79 του αρχείου dfm2mdl.c.

Αναφορά από add_operation_text_old(), aliasparser(), get_listfiles(), get_next_line(), load_dfm(), load_operations(), make_relations(), parser2targetdfm(), prepare_dfm_pas_files(), και stereotype_classify().

Η τεκμηρίωση για αυτή η δομή δημιουργήθηκε απο το ακόλουθο αρχείο:

- **dfm2mdl.c**

listfiles Αναφορά Δομής

listfileslistfileslistfileslistfilesΠεδία Δεδομένων

- char **tag** [MAX_WORD_LENGTH]
- char **filename** [MAX_WORD_LENGTH+10]
- char **pathname** [MAX_WORD_LENGTH+10]
- **listfiles * next**

Λεπτομερή Περιγραφή

Λίστα στην οποία περιέχεται πληροφορία για όλα τα αρχεία του δοθέντος κώδικα Delphi

Ορισμός στη γραμμή 90 του αρχείου dfm2mdl.c.

Τεκμηρίωση Πεδίων

char listfiles::filename[MAX_WORD_LENGTH+10]

Ορισμός στη γραμμή 91 του αρχείου dfm2mdl.c.

Αναφορά από main().struct listfiles* listfiles::next

Ορισμός στη γραμμή 92 του αρχείου dfm2mdl.c.

Αναφορά από get_listfiles(), και main().char listfiles::pathname[MAX_WORD_LENGTH+10]

Ορισμός στη γραμμή 91 του αρχείου dfm2mdl.c.

Αναφορά από get_listfiles(), και main().char listfiles::tag[MAX_WORD_LENGTH]

Ορισμός στη γραμμή 91 του αρχείου dfm2mdl.c.

Αναφορά από prepare_dfm_pas_files(), και write_node_a().

Η τεκμηρίωση για αυτή η δομή δημιουργήθηκε απο το ακόλουθο αρχείο:

- **dfm2mdl.c**

Delphi to Rational Rose Τεκμηρίωση Αρχείων

dfm2mdl.c Αναφορά Αρχείου

dfm2mdl.cdfm2mdl.cdfm2mdl.cdfm2mdl.c#include <stdio.h>
#include <stdlib.h>
#include <string.h>

Δομές Δεδομένων

- struct **line**
- struct **dfmlist**
- struct **listfiles**

Ορισμοί

- #define **DX** 1000
- #define **DY** 300
- #define **MAX_WORDS_PER_LINE** 100
- #define **MAX_WORD_LENGTH** 500
- #define **COLOR_DATA** 10390226
- #define **COLOR_PRESENTATION_COMPOSITE** 13434879
- #define **COLOR_PRESENTATION_PRIMITIVE** 12632730
- #define **COLOR_NA** 5765544
- #define **DIRSEPARATOR** '\\'
- #define **FILEDELPHIPSM** 1
- #define **FILEPIM** 2
- #define **FILEPHPPSM** 3

Συναρτήσεις

- int **get_next_line** (FILE *ftmp, struct **line** *tmp)
- FILE * **Fopen** (char *name, char *mode)
- int **stereotype_classify** (char *stereotype)
- void **append2mdl** (char *filename)
- **dfmlist** * **make_a_dfmnode** (struct **dfmlist** *parent, struct **dfmlist** *next, struct **dfmlist** *fson, int x, int y)
- **dfmlist** * **load_dfm** (FILE *fp, struct **dfmlist** *parent, struct **line** *cline)
- void **print_dfm_list** (struct **dfmlist** *head)
- void **free_list** (struct **dfmlist** *head)
- FILE * **fplaceA** (FILE *from, char *field, char *replace)
- FILE * **fplaceB** (FILE *from, char *field, char *replace)
- int **contains** (char *string, char *pattern)
- void **add_operation_text_old** (FILE *fin, char *opname)
- void **write_node_a** (struct **dfmlist** *head)
- void **write_node_b** (struct **dfmlist** *head)
- void **append_to_ab** (struct **dfmlist** *head)
- void **write_assoc_a** (struct **dfmlist** *head, int arg1, int arg2, int arg3, int arg4)
- void **write_assoc_b** (struct **dfmlist** *head)
- void **append_assoc_to_ab** (struct **dfmlist** *head)
- void **generate_ab_parts** (struct **dfmlist** *head)
- void **create_mdl** (void)

- void **load_operations** (char *file)
- void **make_relations** (struct **dfmlist** *head)
- void **create_final_mdl** (void)
- void **prepare_dfm_pas_files** ()
- void **get_listfiles** (char *dprfile)
- void **replace_word** (char *word)
- void **aliasparser** (void)
- void **parser2targetdfm** (void)
- **main** ()

Μεταβλητές

- FILE * **fmdl**
- FILE * **fdfm**
- FILE * **ftmp**
- FILE * **fout**
- FILE * **f_a**
- FILE * **f_b**
- FILE * **f_results**
- FILE * **f_results1**
- FILE * **f_results2**
- int **curr_id** = 11111
- int **assoc** = 5111111
- int **assoc_1** = 6111111
- int **assoc_2** = 7111111
- int **oper_id** = 9111111
- int **inher_id** = 8111111
- int **nextX**
- int **nextY**
- int **procedurenum**
- int **functionnum**
- int **nodenum**
- int **file2visualize**
- int **aaa** = 0
- char **opWords** [100][MAX_WORD_LENGTH]
- char **Tag** [MAX_WORD_LENGTH]
- char **PathName** [MAX_WORD_LENGTH+10]
- char **FileName** [MAX_WORD_LENGTH+10]
- char **tempdirname** [MAX_WORD_LENGTH+10]
- char **event_tempdirname** [MAX_WORD_LENGTH+10]
- int **results_aggregation**
- int **results_methods**
- int **results_elements**
- int **results_inheritance**
- int **results_relation**
- int **results_bus**
- int **results_data**
- int **results_cpres**
- int **results_ppres**
- int **results_total_php_bus**
- int **results_total_php_data**

- int **results_total_php_cpres**
- int **results_total_php_ppres**
- int **results_total_psm_bus**
- int **results_total_psm_data**
- int **results_total_psm_cpres**
- int **results_total_psm_ppres**
- int **results_total_pim_bus**
- int **results_total_pim_data**
- int **results_total_pim_cpres**
- int **results_total_pim_ppres**
- int **results_pic**
- int **results_pdc**
- int **results_total_pic**
- int **results_total_pdc**
- **dfmlist * ghead**
- **listfiles * filenames**
- **listfiles * filenames_end**
- **listfiles * workingfile**
- char **filenametmp** [MAX_WORD_LENGTH]
- char **b**
- **line flipline**
- int **wlength**
- int **ai**
- FILE * **frel** = NULL

Τεκμηρίωση Ορισμών

#define COLOR_DATA 10390226

Το χρώμα που θα έχουν στο Rational Rose τα στοιχεία του data πακέτου

Ορισμός στη γραμμή 10 του αρχείου dfm2mdl.c.

Αναφορά από stereotype_classify().#define COLOR_NA 5765544

Το χρώμα που θα έχουν στο Rational Rose τα στοιχεία του business πακέτου

Ορισμός στη γραμμή 13 του αρχείου dfm2mdl.c.

Αναφορά από stereotype_classify(), και write_node_b().#define COLOR_PRESENTATION_COMPOSITE 13434879

Το χρώμα που θα έχουν στο Rational Rose τα στοιχεία του composite presentation πακέτου

Ορισμός στη γραμμή 11 του αρχείου dfm2mdl.c.

Αναφορά από stereotype_classify().#define COLOR_PRESENTATION_PRIMITIVE 12632730

Το χρώμα που θα έχουν στο Rational Rose τα στοιχεία του primitive presentation πακέτου

Ορισμός στη γραμμή 12 του αρχείου dfm2mdl.c.

Αναφορά από stereotype_classify().#define DIRSEPARATOR '\\'

Το σύμβολο για τους καταλόγους στο filesystem

Ορισμός στη γραμμή 14 του αρχείου dfm2mdl.c.

Αναφορά από prepare_dfm_pas_files(), και write_node_a().#define DX 1000

Ορισμός στη γραμμή 5 του αρχείου dfm2mdl.c.

Αναφορά από write_assoc_b(), και write_node_b().#define DY 300

Ορισμός στη γραμμή 6 του αρχείου dfm2mdl.c.

Αναφορά από write_assoc_b(), και write_node_b().#define FILEDELPHIPSM 1

Χαρακτηριστικό για Delphi PSM

Ορισμός στη γραμμή 16 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), main(), και make_relations().#define FILEPHPPSM 3

Χαρακτηριστικό για PhP PSM

Ορισμός στη γραμμή 18 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), main(), και make_relations().#define FILEPIM 2

Χαρακτηριστικό για PIM

Ορισμός στη γραμμή 17 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), main(), και make_relations().#define MAX_WORD_LENGTH 500

Το μέγιστο πλήθος χαρακτήρων που επιτρέπουμε να έχει μία λέξη

Ορισμός στη γραμμή 9 του αρχείου dfm2mdl.c.

Αναφορά από aliasparser(), create_final_mdl(), get_listfiles(), get_next_line(), load_dfm(), load_operations(), main(), make_relations(), parser2targetdfm(), prepare_dfm_pas_files(), replace_word(), write_assoc_a(), write_assoc_b(), write_node_a(), και write_node_b().#define MAX_WORDS_PER_LINE 100

Το μέγιστο πλήθος λέξεων ανά γραμμή

Ορισμός στη γραμμή 8 του αρχείου dfm2mdl.c.

Αναφορά από get_next_line(), και load_dfm().

void add_operation_text_old (FILE * *fin*, char * *opname*)

Προσθέτει στο mdl αρχείο για την αναπαράστασης μία μέθοδο ως operation.

Ορισμός στη γραμμή 402 του αρχείου dfm2mdl.c.

```
References b, contains(), f_a, get_next_line(), line::num_of_words, και line::words.402 {
403   struct line line;
404   int count =0,i,j;
405   char n,a,b,c,d;
406
407   while (count != 2){
408     get_next_line(fin, &line);
409     for (i=0;i<line.num_of_words;i++){
410           if (contains( line.words[i], opname)==1 ){
411             if (count == 0){
412                count = 1;
413             }
414             else{
415                count = 2;
416             }
417           }
418     }
419   }
420   while (strncmp(line.words[0], "begin", 5 ) == 0 ) get_next_line(fin, &line);
421   n=a=b=c=d='\0';
422   while (!( n== 'b' && a == 'e' && b== 'g' && c == 'i'&& d == 'n' )){
423    n=a; a=b; b= c; c=d;
424    d = getc(fin);
425   }
426   fprintf(f_a,"begin");
427   n=a=b=c=d='\0';
428   d = getc(fin);
429   while (!( n== '\n' && a == 'e' && b== 'n' && c == 'd'&& d == ';' )){
430    fputc(d,f_a);
431    n=a; a=b; b= c; c=d;
432    d = getc(fin);
433   }
434   fputc(d,f_a);
435 }
```

void aliasparser (void)

Μετά την προεπεξεργασία του δοθέντος κώδικα που γίνεται στο prepare_dfm_pas_files αναλαμβάνει η παρούσα συνάρτηση να κατασκευάσει με την βοήθεια αντιστοιχιών τα αρχεία που περιέχουν το Delphi PSM και PIM.

Ορισμός στη γραμμή 1086 του αρχείου dfm2mdl.c.

References aaa, b, FileName, Fopen(), get_next_line(), MAX_WORD_LENGTH, PathName, replace_word(), results_pdc, και line::words.

```
Αναφορά από main().1086 {
1087   FILE *newPSMf, *newPIMf, *fdfm1, *fdfm2, *fmaptmp;
1088   char nametmp[MAX_WORD_LENGTH], b,c,d,e,f,asdf;
1089   struct line line, lineasdf;
1090   int i, spaces, ftime =0;
1091
1092   for(i=0;i<MAX_WORD_LENGTH;i++) nametmp[i] = '\0';
1093   sprintf(nametmp, "%s%s.dfm\0",PathName,FileName);
1094   fdfm1 = Fopen(nametmp, "r");
1095   fdfm2 = Fopen(nametmp, "r");
1096
1097
1098   for(i=0;i<MAX_WORD_LENGTH;i++) nametmp[i] = '\0';
```

```
1099  sprintf(nametmp, "%s%s.psm\0",PathName,FileName);
1100  newPSMf = Fopen(nametmp,"w");
1101
1102  for(i=0;i<MAX_WORD_LENGTH;i++) nametmp[i] = '\0';
1103  sprintf(nametmp, "%s%s.pim\0",PathName,FileName);
1104  newPIMf = Fopen(nametmp,"w");
1105
1106  b = get_next_line(fdfm1, &line);
1107  results_pdc++;
1108  while(b!=EOF){
1109    if (strncmp( line.words[0],"object",6 ) == 0){
1110       while ((c=getc(fdfm2))== ' '){
1111         fputc(c, newPSMf);
1112         fputc(c, newPIMf);
1113       }
1114       if (c== '\n') printf("paizei prob 1\n");
1115       while ((c=getc(fdfm2))!= '\n');
1116
1117       for (i=0;line.words[1][i] != ':' ;i++);
1118       line.words[1][i]='\0';
1119       if (ftime ==0){
1120         for (i=0;i<MAX_WORD_LENGTH;i++) line.words[2][i] = '\0';
1121         strcpy(line.words[2], "DelphiForm\0");
1122         ftime = 1;
1123       }
1124       replace_word(line.words[2]);
1125       fprintf(newPSMf," %s %s %s\n",line.words[0],line.words[1],line.words[2]);
1126
1127       for(i=0;i<MAX_WORD_LENGTH;i++) nametmp[i] = '\0';
1128       sprintf(nametmp, "obj_spec\\map Delph 2 PIM\\%s.txt\0",line.words[2]);
1129       fmaptmp = fopen (nametmp, "r");
1130       if (fmaptmp == NULL){
1131         fprintf(stderr,"undeclared mapping for: %s\n", line.words[2]);
1132         fprintf(newPIMf," %s %s %s\n",line.words[0],line.words[1],line.words[2]);
1133       }
1134       else{
1135         fprintf(newPIMf," %s %s ",line.words[0],line.words[1]);
1136         while ((c=getc(fmaptmp))!= '\n')  fputc(c, newPIMf);
1137
1138         fputc(c, newPIMf);
1139       }
1140    }
1141    else if (strncmp( line.words[0],"end",3 ) == 0 && line.words[1][0] == '\0'){
1142
1143       if (fmaptmp != NULL){
1144          d=getc(fmaptmp);
1145          e=getc(fmaptmp);
1146          f=getc(fmaptmp);
1147          while ((c=getc(fmaptmp))!= EOF){
1148             if ( d == 'X' && e == 'X' &&  f == '-'){
1149                fputc(c, newPIMf);
1150                while ((c=getc(fmaptmp))!= ' ') fputc(c, newPIMf);
1151                fprintf(newPIMf,"%i",aaa++);
1152
1153                d = c;
1154                e=getc(fmaptmp);
1155                f=getc(fmaptmp);
1156                c=getc(fmaptmp);
1157             }
1158             if ( d != 'X' || e != 'X' ||  f == '-') fputc(d, newPIMf);
1159
1160             d = e;
1161             e = f;
1162             f = c;
1163          }
1164          if (d != '\0' && d != EOF ) fputc(d, newPIMf);
1165          if (e != '\0' && e != EOF ) fputc(e, newPIMf);
1166          if (f != '\0' && f != EOF ) fputc(f, newPIMf);
1167          fclose(fmaptmp);
1168       }
1169
1170       while ((c=getc(fdfm2))!= '\n'){
```

```
1171            fputc(c, newPSMf);
1172            fputc(c, newPIMf);
1173        }
1174        fputc(c, newPSMf);
1175        fputc(c, newPIMf);
1176    }
1177    else{
1178        while ((c=getc(fdfm2))!= '\n'){
1179            fputc(c, newPSMf);
1180            fputc(c, newPIMf);
1181            if ( c == '<'){
1182              while((c=getc(fdfm2))!= '>'){
1183                if (c == '\n'){
1184                    b = get_next_line(fdfm1, &line);
1185                    results_pdc++;
1186                }
1187                fputc(c, newPSMf);
1188                fputc(c, newPIMf);
1189              }
1190              fputc(c, newPSMf);
1191              fputc(c, newPIMf);
1192            }
1193        }
1194        fputc('\n', newPSMf);
1195        fputc('\n', newPIMf);
1196    }
1197    b = get_next_line(fdfm1, &line);
1198    results_pdc++;
1199  }
1200  fclose(fdfm1);
1201  fclose(fdfm2);
1202  fclose(newPSMf);
1203  fclose(newPIMf);
1204 }
```

void append2mdl (char * *filename*)

Προσθέτει τα δεδομένα του αρχείου filename στο αρχείο a (part 2).

Ορισμός στη γραμμή 202 του αρχείου dfm2mdl.c.

References fmdl, Fopen(), και ftmp.

Αναφορά από create_mdl().202

```
203 char tmp;
204   ftmp = Fopen(filename,"r");
205   while( (tmp = fgetc(ftmp) )!= EOF)
206     fputc(tmp,fmdl);
207   fclose(ftmp);
208 }
```

void append_assoc_to_ab (struct dfmlist * *head*)

Με τη βοήθεια της αναδρομής τοποθετούμε τις συσχετίσεις που ανήκουν στο μοντέλο και τα τοποθετούμε στα parts 2 και 4 του mdl αρχείου.

Ορισμός στη γραμμή 707 του αρχείου dfm2mdl.c.

References dfmlist::arg, assoc, assoc_1, assoc_2, dfmlist::fson, ghead, dfmlist::inherit, dfmlist::next, dfmlist::parent, write_assoc_a(), και write_assoc_b().

Αναφορά από generate_ab_parts().707

```
708   if( head == NULL) return;
709   if (head->parent != ghead && head->inherit != 0){
710     assoc++;assoc_1++;assoc_2++;
711     write_assoc_a(head,head->arg[0],head->arg[1],head->arg[2],head->arg[3]);
712     write_assoc_b(head);
713   }
714   append_assoc_to_ab(head->fson);
```

```
715   append_assoc_to_ab(head->next);
716   return;
717 }
```

void append_to_ab (struct dfmlist * *head*)

Με τη βοήθεια της αναδρομής διατρέχουμε όλα τα στοιχεία που ανήκουν στο μοντέλο και τα τοποθετούμε στα parts 2 και 4 του mdl αρχείου.

Ορισμός στη γραμμή 618 του αρχείου dfm2mdl.c.

References dfmlist::fson, ghead, dfmlist::next, write_node_a(), και write_node_b().

```
Αναφορά από generate_ab_parts().618
619   if( head == NULL) return;
620   if (head != ghead) {
621      write_node_a(head);
622      write_node_b(head);
623   }
624   append_to_ab(head->fson);
625   append_to_ab(head->next);
626   return;
627 }
```

int contains (char * *string*, char * *pattern*)

Ελέγχει εάν ένα string περιέχει κάποια συμβολοσειρά.

Ορισμός στη γραμμή 383 του αρχείου dfm2mdl.c.

```
Αναφορά από add_operation_text_old().383
384   char *source=string, *pat=pattern;
385   int wordlength,i;
386   for(wordlength=0;pattern[wordlength]!='\0';wordlength++);
387   while (*source != '\0') {
388      i=0;
389      while (source[i] == *pat) { i++; pat++;}
390      if (i>=wordlength-2) {
391         return(1);
392      }
393      source++;
394      pat=pattern;
395   }
396   return(0);
397 }
```

void create_final_mdl (void)

Κύρια διαδικασία για την παραγωγή ενός τελικού αρχείου mdl το οποίο οπτικοποιείται με τη βοήθεια του Rational ROSE.

Ορισμός στη γραμμή 825 του αρχείου dfm2mdl.c.

References assoc, assoc_1, assoc_2, b, create_mdl(), curr_id, f_results, f_results1, f_results2, fdfm, file2visualize, FILEDELPHIPSM, FileName, filenametmp, FILEPHPPSM, FILEPIM, fmdl, Fopen(), fout, free_list(), dfmlist::fson, generate_ab_parts(), get_next_line(), ghead, inher_id, load_dfm(), load_operations(), make_a_dfmnode(), make_relations(), MAX_WORD_LENGTH, nextX, nextY, nodenum, dfmlist::objectname, oper_id, PathName, print_dfm_list(), results_aggregation, results_bus, results_cpres, results_data, results_elements, results_inheritance, results_methods, results_pdc, results_pic, results_ppres, results_relation, results_total_php_bus, results_total_php_cpres, results_total_php_data, results_total_php_ppres, results_total_pim_bus, results_total_pim_cpres, results_total_pim_data, results_total_pim_ppres, results_total_psm_bus, results_total_psm_cpres, results_total_psm_data, και results_total_psm_ppres.

Αναφορά από main().825

```
826  char b,c,filenametmp[MAX_WORD_LENGTH];
827  struct line cur_line, line2, line3;
828  int i;
829
830  ghead = NULL;
831  curr_id= 11111;
832  assoc = 5111111;
833  assoc_1=6111111;
834  assoc_2=7111111;
835  oper_id=9111111;
836  inher_id=8111111;
837
838  nodenum=0;
839  load_operations("obj_spec\\operations.txt\0");
840
841  results_aggregation= results_methods= results_elements= 0;
842  results_inheritance= results_relation= 0;
843  results_bus= results_data= results_cpres= results_ppres= 0;
844
845  fprintf(f_results,"%s%s\t",PathName,FileName);
846
847  if (file2visualize == FILEDELPHIPSM){
848     printf("Converting %s%s.PSM -> DELPHIPSM .mdl \n",PathName,FileName);
849     for(i=0;i<MAX_WORD_LENGTH;i++) filenametmp[i] = '\0';
850     sprintf(filenametmp, "%s%sDELPHIPSM.mdl\0",PathName,FileName);
851     fmdl = Fopen(filenametmp,"w");
852     for(i=0;i<MAX_WORD_LENGTH;i++) filenametmp[i] = '\0';
853     sprintf(filenametmp, "%s%s.psm\0",PathName,FileName);
854     fdfm = Fopen(filenametmp, "r");
855     fprintf(f_results,"%Delphi\t");
856  }
857  else  if (file2visualize == FILEPIM){
858     printf("Converting %s%s.PIM -> PIM.mdl \n",PathName,FileName);
859     for(i=0;i<MAX_WORD_LENGTH;i++) filenametmp[i] = '\0';
860     sprintf(filenametmp, "%s%sPIM.mdl\0",PathName,FileName);
861     fmdl = Fopen(filenametmp,"w");
862
863     for(i=0;i<MAX_WORD_LENGTH;i++) filenametmp[i] = '\0';
864     sprintf(filenametmp, "%s%s.pim\0",PathName,FileName);
865     fdfm = Fopen(filenametmp, "r");
866     fprintf(f_results,"%PIM\t");
867  }
868  else  if (file2visualize == FILEPHPPSM){
869     printf("Converting %s%s.dfm -> dfm.mdl \n",PathName,FileName);
870     for(i=0;i<MAX_WORD_LENGTH;i++) filenametmp[i] = '\0';
871     sprintf(filenametmp, "%s%sPhPPSM.mdl\0",PathName,FileName);
872     fmdl = Fopen(filenametmp,"w");
873
874     for(i=0;i<MAX_WORD_LENGTH;i++) filenametmp[i] = '\0';
875     sprintf(filenametmp, "%s%sPhP.psm\0",PathName,FileName);
876     fdfm = Fopen(filenametmp, "r");
877     fprintf(f_results,"%PhP\t");
878  }
879  else {
880     return;
881  }
882
883
884  nextX=1;nextY=1;
885  fprintf(fout,"_____Files ready_____\n");
886  ghead =  make_a_dfmnode(NULL,NULL,NULL,nextX,nextY-1);
887  strcpy(ghead->objectname, "ROOT\0");
888  b=get_next_line(fdfm, &cur_line);
889  ghead->fson =load_dfm(fdfm,ghead,&cur_line);
890  print_dfm_list(ghead->fson);
891  make_relations(ghead->fson);
892  fprintf(fout,"_____Generating ab parts_____\n");
893  generate_ab_parts(ghead->fson);
894  fprintf(fout,"_____Creating Final mdl _____\n");
895  create_mdl();
896
897  fclose(fmdl);
```

```
898   fclose(fdfm);
899   free_list(ghead);
900   ghead=NULL;
901
902   fprintf(f_results,"\t%i(%i/%i/%i/%i)", results_elements, results_bus, results_data, results_cpres, results_ppres);
903
904   if (file2visualize == FILEDELPHIPSM){
905      results_total_psm_bus+= results_bus;
906      results_total_psm_data+= results_data;
907      results_total_psm_cpres+= results_cpres;
908      results_total_psm_ppres+= results_ppres;
909   }
910   else  if (file2visualize == FILEPIM){
911      results_total_pim_bus+= results_bus;
912      results_total_pim_data+= results_data;
913      results_total_pim_cpres+= results_cpres;
914      results_total_pim_ppres+= results_ppres;
915   }
916   else  if (file2visualize == FILEPHPPSM){
917      results_total_php_bus+= results_bus;
918      results_total_php_data+= results_data;
919      results_total_php_cpres+= results_cpres;
920      results_total_php_ppres+= results_ppres;
921   }
922   if (file2visualize == FILEDELPHIPSM)
923      fprintf(f_results2,"\n" );
924
925   fprintf(f_results1,"%i\t%i\t%i\t%i\t%i\n", results_elements, results_bus, results_data, results_cpres, results_ppres);
926   fprintf(f_results,"\t%i\t\t%i\t", results_aggregation,results_inheritance);
927   fprintf(f_results,"\t%i\t\t%i\n", results_relation, results_methods);
928   fprintf(f_results2,"%i\t%i\t%i\t", results_aggregation,results_inheritance, results_relation);
929
930   if (file2visualize == FILEDELPHIPSM)
931         fprintf(f_results,"\nPlatform dependent code %i || Platform Independent code %i\n",results_pdc, results_pic);
932
933 }
```

void create_mdl (void)

Συνάρτηση που αναλαμβάνει να κατασκευάσει το τελικό mdl.

Ορισμός στη γραμμή 735 του αρχείου dfm2mdl.c.

References append2mdl().

```
Αναφορά από create_final_mdl().735          {
736   append2mdl("head.txt");
737   append2mdl("a.txt");
738   append2mdl("mid.txt");
739   append2mdl("b.txt");
740   append2mdl("tail.txt");
741 }
```

FILE* Fopen (char * *name*, char * *mode*)

Χρησιμοποιείται για το άνοιγμα αρχείων.

Ορισμός στη γραμμή 142 του αρχείου dfm2mdl.c.

References fout.

```
Αναφορά από aliasparser(), append2mdl(), create_final_mdl(), generate_ab_parts(), get_listfiles(), load_dfm(), load_operations(),
main(), parser2targetdfm(), prepare_dfm_pas_files(), replace_word(), stereotype_classify(), write_assoc_a(), write_assoc_b(),
write_node_a(), και write_node_b().142                    {
143 FILE *t;
144 char tt;
145   if (NULL== (t=fopen(name,mode))){
146      if ( mode[0] == 'r')
147         fprintf(fout,"couldn't read the file name: %s \n",name);
```

```
148     else
149       fprintf(fout,"couldn't write the file name: %s \n",name);
150     exit(0);
151   }
152   return(t);
153 }
```

FILE* fplaceA (FILE * *from*, char * *field*, char * *replace*)

Συνάρτηση που χρησιμεύει στο χτίσιμο του αρχείου που αντιπροσωπεύει το part 2.

Ορισμός στη γραμμή 351 του αρχείου dfm2mdl.c.

References b, και f_a.

```
Αναφορά από write_assoc_a(), και write_node_a().351
352   char a,b,c,buffer;
353   int i;
354   b=getc(from);    c=getc(from);
355   while (b != '-' && c != '='){
356     a=b; b=c; c=getc(from);
357     fputc(a, f_a);
358   }
359   fprintf(f_a,"%s",replace);
360   while ( (c=getc(from)) != '-');
361   return(from);
362 }
```

FILE* fplaceB (FILE * *from*, char * *field*, char * *replace*)

Συνάρτηση που χρησιμεύει στο χτίσιμο του αρχείου που αντιπροσωπεύει το part 4.

Ορισμός στη γραμμή 367 του αρχείου dfm2mdl.c.

References b, και f_b.

```
Αναφορά από write_assoc_b(), και write_node_b().367
368   char a,b,c,buffer;
369   int i;
370   b=getc(from);    c=getc(from);
371   while (b != '-' && c != '='){
372     a=b; b=c; c=getc(from);
373     fputc(a, f_b);
374   }
375   fprintf(f_b,"%s",replace);
376   while ( (c=getc(from)) != '-');
377   return(from);
378 }
```

void free_list (struct dfmlist * *head*)

Αποδεσμεύει την μνήμη που έχουμε δεσμεύσει για την κατακράτηση ενός μοντέλου στη μνήμη.

Ορισμός στη γραμμή 340 του αρχείου dfm2mdl.c.

References dfmlist::fson, και dfmlist::next.

```
Αναφορά από create_final_mdl().340
341   if( head == NULL) return;
342   free_list(head->fson);
343   free_list(head->next);
344   free(head);
345   return;
346 }
```

void generate_ab_parts (struct dfmlist * *head*)

Συνάρτηση που αναλαμβάνει να κατασκευάσει τα parts 2 και 4.

Ορισμός στη γραμμή 722 του αρχείου dfm2mdl.c.

References append_assoc_to_ab(), append_to_ab(), f_a, f_b, Fopen(), και fout.

```
Αναφορά από create_final_mdl().722                    {
723   f_a = Fopen("a.txt","w");
724   f_b = Fopen("b.txt","w");
725   append_to_ab( head);
726   fprintf(fout,"_____\n");
727   append_assoc_to_ab( head);
728   fclose(f_a);
729   fclose(f_b);
730 }
```

void get_listfiles (char * *dprfile*)

Διαβάζει από ένα αρχείο dpr (περιγράφει ένα project) της Delphi όλα τα εμπλεκόμενα αρχεία. Εμείς τα αρχεία αυτά τα μοντελοποιούμε όλα ένα - ένα σαν ξεχωριστή περίπτωση.

Ορισμός στη γραμμή 1011 του αρχείου dfm2mdl.c.

References b, filenames, filenames_end, Fopen(), fout, get_next_line(), MAX_WORD_LENGTH, listfiles::next, listfiles::pathname, line::words, και workingfile.

```
Αναφορά από main().1011                    {
1012   char c, b, bouff[MAX_WORD_LENGTH+10];
1013   int i, k, l, m;
1014   struct line line;
1015   struct listfiles *tmp_f;
1016   FILE *fin;
1017
1018   filenames = workingfile =filenames_end =NULL;
1019   fin = Fopen (dprfile, "r");
1020   line.words[0][0] = '\0';
1021   while (strncmp( line.words[0],"program",7 ) != 0){
1022     b = get_next_line(fin, &line);
1023   }
1024   for(i=0;line.words[1][i] != ';' ; i++);
1025   line.words[1][i] = '\0';
1026   fprintf(fout,"program name *** %s ***\n",line.words[1]);
1027   while (strncmp( line.words[0],"uses",4 ) != 0){
1028     b = get_next_line(fin, &line);
1029   }
1030
1031   while (strncmp( line.words[0],"begin", 5 ) != 0){
1032     if (strncmp( line.words[1],"in", 2 ) == 0){
1033       tmp_f =  malloc( sizeof(struct listfiles ));
1034       tmp_f->next = NULL;
1035       if (filenames_end == NULL) filenames = tmp_f ;
1036       else filenames_end->next = tmp_f;
1037       filenames_end = tmp_f;
1038       tmp_f->pathname[0] = '.';
1039       tmp_f->pathname[1] = '\\';
1040       for(i=1;line.words[2][i] != '.' ; i++) tmp_f->pathname[i+1] = line.words[2][i];
1041       i++;
1042       for(;tmp_f->pathname[i]!= '\\';i--);
1043       for(m=0,k=i+1; tmp_f->pathname[k] != '\0';k++,m++ )
1044             tmp_f->filename[m] = tmp_f->pathname[k];
1045
1046       tmp_f->filename[m] = '\0';
1047
1048       tmp_f->pathname[i+1] = '\0';
1049     }
1050     b = get_next_line(fin, &line);
1051   }
1052   fclose(fin);
1053 }
```

int get_next_line (FILE * *ftmp*, struct line * *tmp*)

Διαβάζει την επόμενη γραμμή του αρχείου ftmp και βάζει τις λέξεις στη παράμετρο tmp.

Ορισμός στη γραμμή 98 του αρχείου dfm2mdl.c.

References ftmp, MAX_WORD_LENGTH, MAX_WORDS_PER_LINE, line::num_of_words, και line::words.

Αναφορά από add_operation_text_old(), aliasparser(), create_final_mdl(), get_listfiles(), load_dfm(), load_operations(), make_relations(), parser2targetdfm(), prepare_dfm_pas_files(), και stereotype_classify().98

```
99  int i,j;
100 char c;
101 tmp->num_of_words=0;
102 for (i=0;i<MAX_WORDS_PER_LINE;i++)
103   for (j=0;j<MAX_WORD_LENGTH;j++) tmp->words[i][j]='\0';
104 j=0;
105 c = fgetc(ftmp);
106 if (c== EOF) return(EOF);
107 while( c  != '\n' && c!=EOF){
108   if( c==' ')
109     if (j==0 && tmp->num_of_words == 0)
110       while (c== ' ') c= fgetc(ftmp);
111     else{
112       tmp->num_of_words++;
113       j=0;
114
115       while (c== ' ') c= fgetc(ftmp);
116     }
117   if ( c == '\"'){
118     tmp->words[tmp->num_of_words][0]='\";
119     c= fgetc(ftmp);j=1;
120     while (c != '\"'){
121       tmp->words[tmp->num_of_words][j]=c;
122       c= fgetc(ftmp);
123       j++;
124     }
125     tmp->words[tmp->num_of_words][j]='\";
126     tmp->num_of_words++;
127     j=0;
128   }
129   else{
130     tmp->words[tmp->num_of_words][j]=c;
131     j++;
132   }
133   c = fgetc(ftmp);
134 }
135 tmp->num_of_words++;
136 return(c);
137 }
```

struct dfmlist* load_dfm (FILE * *fp*, struct dfmlist * *parent*, struct line * *cline*)

Από ένα αρχείο που περιέχει τα στοιχεία για το Delphi PSM ή PIM ή PhP PSM τα τοποθετούμε στην μνήμη με κατάλληλο τρόπο.

Ορισμός στη γραμμή 231 του αρχείου dfm2mdl.c.

References b, Fopen(), get_next_line(), make_a_dfmnode(), MAX_WORD_LENGTH, MAX_WORDS_PER_LINE, nextX, nextY, tempdirname, και line::words.

Αναφορά από create_final_mdl().231

```
232 struct line line;
233 struct dfmlist *head,*tmp1,*lastson=NULL;
234 int next_object_child =1,i,code;
235 char b,filename[MAX_WORD_LENGTH+10];
236
237 FILE *file_tmp=NULL;
238
```

```
239  head = make_a_dfmnode(parent,NULL,NULL,nextX,nextY);
240
241  strcpy(head->objectname, cline->words[1]);
242  strcpy(head->objecttype, cline->words[2]);
243
244  for(i=0;i<MAX_WORD_LENGTH+10;i++)filename[i]='\0';
245  sprintf(filename,"%s\\\\0",tempdirname);
246  strcat(filename,cline->words[1]);
247  file_tmp = Fopen(filename,"w");
248
249  b=get_next_line(fp, &line);
250  while(b !=EOF){
251
252     if ( strncmp(line.words[0],"object",6 ) == 0){
253        fclose(file_tmp);
254        if (next_object_child == 1) {
255           nextX++;
256           tmp1 = load_dfm(fp,head,&line);
257           head->fson=tmp1;
258           next_object_child =0;
259        }
260        else {
261           nextY++;
262           tmp1 = load_dfm(fp,head,&line);
263        }
264
265        if (lastson == NULL)
266           lastson = tmp1;
267        else
268           lastson = lastson->next = tmp1;
269        lastson->next=NULL;
270     }
271     else if ( strncmp(line.words[0],"end",3 ) == 0){
272        if (next_object_child == 1) {
273             fclose(file_tmp);
274             return(head);
275        }
276        else{
277          nextX--;
278          fclose(file_tmp);
279          return(head);
280        }
281     }
282     if ( strncmp(line.words[0],"object",6 ) != 0){
283       for(i=0;i<MAX_WORDS_PER_LINE;i++)
284        if(line.words[i][0] != '\0')
285           fprintf(file_tmp,"%s ",line.words[i]);
286        fprintf(file_tmp,"\n");
287     }
288     b=get_next_line(fp, &line);
289     code =0;
290     for(i=0;i<MAX_WORDS_PER_LINE;i++)
291       if ( strrchr(line.words[i],'<') != NULL) code=1;
292
293     if (code == 1){
294       for(i=0;i<MAX_WORDS_PER_LINE;i++)
295         if ( strrchr(line.words[i],'>') != NULL) code=0;
296       if (code == 0){
297          for(i=0;i<MAX_WORDS_PER_LINE;i++)
298            if(line.words[i][0] != '\0')
299               fprintf(file_tmp,"%s ",line.words[i]);
300            fprintf(file_tmp,"\n");
301          b=get_next_line(fp, &line);
302       }
303     }
304
305     while (code == 1){
306       for(i=0;i<MAX_WORDS_PER_LINE;i++)
307        if(line.words[i][0] != '\0')
308           fprintf(file_tmp,"%s ",line.words[i]);
309        fprintf(file_tmp,"\n");
310       b=get_next_line(fp, &line);
```

```
311         for(i=0;i<MAX_WORDS_PER_LINE;i++)
312          if ( strrchr(line.words[i],'>') != NULL) code=0;
313         if (code == 0){
314          for(i=0;i<MAX_WORDS_PER_LINE;i++)
315            if(line.words[i][0] != '\0')
316               fprintf(file_tmp,"%s ",line.words[i]);
317          fprintf(file_tmp,"\n");
318          b=get_next_line(fp, &line);
319         }
320       }
321    }
322    fclose(file_tmp);
323    return(head);
324 }
```

void load_operations (char * *file*)

Συνάρτηση που διαβάζει από κάποιο αρχείο τις λέξεις κλειδιά που αφορούνε events για κάποιο στοιχείο.

Ορισμός στη γραμμή 746 του αρχείου dfm2mdl.c.

References b, Fopen(), get_next_line(), MAX_WORD_LENGTH, opWords, και line::words.

```
Αναφορά από create_final_mdl().746                    {
747    struct line tmpline;
748    char b;
749    int i,j;
750    FILE *tfile;
751    for(i=0;i<100;i++) for(j=0;j<MAX_WORD_LENGTH;j++) opWords[i][j] = '\0';
752    tfile = Fopen(file, "r");
753    i=0;
754    b=get_next_line(tfile, &tmpline);
755    while(b !=EOF){
756       strcpy(opWords[i], tmpline.words[0]);
757       b=get_next_line(tfile, &tmpline);
758       i++;
759    }
760    fclose (tfile);
761 }
```

main ()

Η main του προγράμματός μας.

Ορισμός στη γραμμή 1279 του αρχείου dfm2mdl.c.

References aliasparser(), create_final_mdl(), f_results, f_results1, f_results2, file2visualize, FILEDELPHIPSM, listfiles::filename, FileName, filenames, filenames_end, FILEPHPPSM, FILEPIM, Fopen(), fout, get_listfiles(), MAX_WORD_LENGTH, listfiles::next, parser2targetdfm(), listfiles::pathname, PathName, prepare_dfm_pas_files(), results_aggregation, results_elements, results_inheritance, results_methods, results_pdc, results_pic, results_relation, results_total_pdc, results_total_php_bus, results_total_php_cpres, results_total_php_data, results_total_php_ppres, results_total_pic, results_total_pim_bus, results_total_pim_cpres, results_total_pim_data, results_total_pim_ppres, results_total_psm_bus, results_total_psm_cpres, results_total_psm_data, results_total_psm_ppres, και workingfile.1279   {

```
1280    char c,dprfile[100];
1281    int i;
1282
1283    results_total_pic= results_total_pdc= 0;
1284    results_total_psm_bus= results_total_psm_data= results_total_psm_cpres= results_total_psm_ppres= 0;
1285    results_total_pim_bus= results_total_pim_data= results_total_pim_cpres= results_total_pim_ppres= 0;
1286    results_total_php_bus= results_total_php_data= results_total_php_cpres= results_total_php_ppres= 0;
1287
1288
1289    fout = Fopen("output_log.txt","w");
1290    f_results = Fopen("results.txt","w");
1291    f_results1 = Fopen("results_lol.txt","w");
1292    f_results2 = Fopen("results_lol2.txt","w");
1293
1294    filenames = filenames_end = NULL;
```

```
1295
1296
1297    strcpy(dprfile, "HELLENIC_BANK.dpr\0");
1298    get_listfiles(dprfile);
1299
1300    fprintf(f_results,"Filename\t\t\t\t");
1301    fprintf(f_results,"elements\taggregation\tinheritance\t", results_elements, results_aggregation, results_inheritance);
1302    fprintf(f_results,"relations\tmethods \n", results_relation, results_methods);
1303
1304    workingfile=filenames;
1305    while(workingfile != NULL){
1306        for(i=0;i<MAX_WORD_LENGTH+10;i++) FileName[i] = '\0';
1307        strcpy(FileName,workingfile->filename);
1308        for(i=0;i<MAX_WORD_LENGTH+10;i++) PathName[i] = '\0';
1309        strcpy(PathName,workingfile->pathname);
1310        fprintf(fout,"path: %s file: %s ",PathName, FileName);
1311
1312        results_pic= results_pdc=0;
1313        prepare_dfm_pas_files();
1314
1315        aliasparser();
1316
1317
1318        parser2targetdfm();
1319        results_total_pic+=results_pic;
1320        results_total_pdc+=results_pdc;
1321
1322        file2visualize = FILEDELPHIPSM;
1323        create_final_mdl();
1324        file2visualize = FILEPIM;
1325        create_final_mdl();
1326        file2visualize = FILEPHPPSM;
1327        create_final_mdl();
1328        fprintf(f_results,"\n");
1329        workingfile=workingfile->next;
1330    }
1331    fprintf(f_results,"\n Total platform dependent code %i\n",results_total_pdc);
1332    fprintf(f_results,"Total platform independent code %i\n",results_total_pic);
1333
1334    fprintf(f_results,"Total PIM business elements %i\n",results_total_pim_bus);
1335    fprintf(f_results,"Total PIM data     elements %i\n",results_total_pim_data);
1336    fprintf(f_results,"Total PIM composite presentation elements %i\n",results_total_pim_cpres);
1337    fprintf(f_results,"Total PIM primitive presentation elements %i\n",results_total_pim_ppres);
1338
1339    fprintf(f_results,"Total PSM business elements %i\n",results_total_psm_bus);
1340    fprintf(f_results,"Total PSM data     elements %i\n",results_total_psm_data);
1341    fprintf(f_results,"Total PSM composite presentation elements %i\n",results_total_psm_cpres);
1342    fprintf(f_results,"Total PSM primitive presentation elements %i\n",results_total_psm_ppres);
1343
1344    fprintf(f_results,"Total PhP business elements %i\n",results_total_php_bus);
1345    fprintf(f_results,"Total PhP data     elements %i\n",results_total_php_data);
1346    fprintf(f_results,"Total PhP composite presentation elements %i\n",results_total_php_cpres);
1347    fprintf(f_results,"Total PhP primitive presentation elements %i\n",results_total_php_ppres);
1348
1349    fclose(f_results);
1350    fclose(f_results1);
1351    fclose(f_results2);
1352    fclose(fout);
1353 }
```

struct dfmlist* make_a_dfmnode (struct dfmlist * *parent*, struct dfmlist * *next*, struct dfmlist * *fson*, int *x*, int *y*)

Η διαδικασία πρόσθεσης ενός στοιχείου στη δομή μας.

Ορισμός στη γραμμή 213 του αρχείου dfm2mdl.c.

References curr_id, dfmlist::fson, nodenum, dfmlist::parent, και dfmlist::x.

Αναφορά από create_final_mdl(), και load_dfm().213

```
214 struct dfmlist *tplace;
215   nodenum++;
216   tplace = malloc(sizeof( struct dfmlist));
217   tplace->next = next;
218   tplace->parent = parent;
219   tplace->fson = fson;
220   tplace->x = x;
221   tplace->y = y;
222   tplace->quid = curr_id++;
223
224   tplace->inherit=0; tplace->arg[0]= tplace->arg[1]= tplace->arg[2]= tplace->arg[3]=0;
225   return(tplace);
226 }
```

void make_relations (struct dfmlist * *head*)

Ορισμός στη γραμμή 775 του αρχείου dfm2mdl.c.

References ai, dfmlist::arg, b, file2visualize, FILEDELPHIPSM, filenametmp, FILEPHPPSM, FILEPIM, flipline, frel, dfmlist::fson, get_next_line(), dfmlist::inherit, MAX_WORD_LENGTH, dfmlist::next, dfmlist::objecttype, dfmlist::parent, results_aggregation, results_elements, results_inheritance, results_relation, wlength, και line::words.

```
Αναφορά από create_final_mdl().775                {
776   if( head == NULL) return;
777
778   results_elements++;
779   for(ai=0;ai<MAX_WORD_LENGTH;ai++) filenametmp[ai] = '\0';
780   if ( file2visualize == FILEDELPHIPSM)
781     sprintf(filenametmp, "obj_spec\\assoc PSM\\%s.txt\0",head->objecttype);
782   if ( file2visualize == FILEPIM)
783     sprintf(filenametmp, "obj_spec\\assoc PIM\\%s.txt\0",head->objecttype);
784   if ( file2visualize == FILEPHPPSM)
785     sprintf(filenametmp, "obj_spec\\assoc PhP\\%s.txt\0",head->objecttype);
786   frel = fopen(filenametmp,"r");
787   for (wlength=0; head->objecttype[wlength] != '\0';wlength++);
788
789   if (head->parent ==NULL || wlength < 2)  printf("pfffffff %i\n",wlength);
790
791   if (frel != NULL){
792       b = get_next_line(frel, &flipline);
793       while (b!=EOF){
794         if( strncmp( flipline.words[0],head->parent->objecttype, wlength-2 ) == 0 ){
795           head->inherit = atoi(flipline.words[1]);
796           head->arg[0]  = atoi(flipline.words[2]);
797           head->arg[1]  = atoi(flipline.words[3]);
798           head->arg[2]  = atoi(flipline.words[4]);
799           head->arg[3]  = atoi(flipline.words[5]);
800           if (head->inherit == 0) results_inheritance++;
801           if (head->arg[3] == 0) results_aggregation++;
802           else results_relation++;
803
804
805           fclose(frel);
806           if( head->fson != NULL)   make_relations(head->fson);
807           if( head->next != NULL)   make_relations(head->next);
808           return;
809         }
810         b = get_next_line(frel, &flipline);
811       }
812       fclose(frel);
813   }
814   head->inherit=1;
815   head->arg[0]= head->arg[1]= head->arg[2]= head->arg[3]= 0;
816   if( head->fson != NULL)   make_relations(head->fson);
817   if( head->next != NULL)   make_relations(head->next);
818   return;
819 }
```

void parser2targetdfm (void)

Με βάση το PIM που έχουμε στη διάθεσή μας κατασκευάζουμε με την βοήθεια αντιστοιχιών το αρχείο που περιέχει το PhP PSM.

Ορισμός στη γραμμή 1209 του αρχείου dfm2mdl.c.

References b, FileName, Fopen(), get_next_line(), MAX_WORD_LENGTH, PathName, και line::words.

Αναφορά από main().1209

```
1210  FILE *newPhPPSMf, *fdfm1, *fdfm2, *fmaptmp;
1211  char nametmp[MAX_WORD_LENGTH], b,c,asdf;
1212  struct line line, lineasdf;
1213  int i, spaces, ftime =0;
1214
1215  for(i=0;i<MAX_WORD_LENGTH;i++) nametmp[i] = '\0';
1216  sprintf(nametmp, "%s%s.pim\0",PathName,FileName);
1217  fdfm1 = Fopen(nametmp, "r");
1218  fdfm2 = Fopen(nametmp, "r");
1219
1220  for(i=0;i<MAX_WORD_LENGTH;i++) nametmp[i] = '\0';
1221  sprintf(nametmp, "%s%sPhP.psm\0",PathName,FileName);
1222  newPhPPSMf = Fopen(nametmp,"w");
1223
1224  b = get_next_line(fdfm1, &line);
1225  while(b!=EOF){
1226    if (strncmp( line.words[0],"object",6 ) == 0){
1227
1228       while ((c=getc(fdfm2))== ' ') fputc(c, newPhPPSMf);
1229
1230       if (c== '\n') printf("paizei prob 1\n");
1231       while ((c=getc(fdfm2))!= '\n');
1232
1233       for(i=0;i<MAX_WORD_LENGTH;i++) nametmp[i] = '\0';
1234       sprintf(nametmp, "obj_spec\\map PIM 2 PhP\\%s.txt\0",line.words[2]);
1235       fmaptmp = fopen (nametmp, "r");
1236       if (fmaptmp == NULL){
1237         fprintf(stderr,"undeclared mapping for: %s\n", line.words[2]);
1238         fprintf(newPhPPSMf," %s %s %s\n",line.words[0],line.words[1],line.words[2]);
1239       }
1240       else{
1241         fprintf(newPhPPSMf," %s %s ",line.words[0],line.words[1]);
1242         while ((c=getc(fmaptmp))!= EOF)  fputc(c, newPhPPSMf);
1243       }
1244    }
1245    else if (strncmp( line.words[0],"end",3 ) == 0 && line.words[1][0] == '\0'){
1246
1247       if (fmaptmp != NULL){
1248          while ((c=getc(fmaptmp))!= EOF)  fputc(c, newPhPPSMf);
1249          fclose(fmaptmp);
1250       }
1251
1252       while ((c=getc(fdfm2))!= '\n') fputc(c, newPhPPSMf);
1253       fputc(c, newPhPPSMf);
1254    }
1255    else{
1256       while ((c=getc(fdfm2))!= '\n'){
1257          fputc(c, newPhPPSMf);
1258          if ( c == '<'){
1259            while((c=getc(fdfm2))!= '>'){
1260              if (c == '\n') b = get_next_line(fdfm1, &line);
1261              fputc(c, newPhPPSMf);
1262            }
1263            fputc(c, newPhPPSMf);
1264          }
1265       }
1266       fputc('\n', newPhPPSMf);
1267    }
1268    b = get_next_line(fdfm1, &line);
1269  }
```

```
1270   fclose(fdfm1);
1271   fclose(fdfm2);
1272   fclose(newPhPPSMf);
1273 }
```

void prepare_dfm_pas_files ()

Προεπεξεργαζόμαστε τον πηγαίο κώδικα μία φόρμας στη delphi από ένα ζεύγος αρχείων dfm και pas.

Ορισμός στη γραμμή 938 του αρχείου dfm2mdl.c.

References b, DIRSEPARATOR, event_tempdirname, FileName, Fopen(), fout, functionnum, get_next_line(), MAX_WORD_LENGTH, PathName, procedurenum, results_pic, Tag, listfiles::tag, tempdirname, line::words, και workingfile.

```
Αναφορά από main().938                {
939   struct line line;
940   int count=0,i,j;
941   char bbb,n,a,b,c,d,bouff[MAX_WORD_LENGTH+10];
942   FILE *fin,*fwrite;
943
944   procedurenum = functionnum = 0;
945   for(i=0;i<MAX_WORD_LENGTH+10;i++) bouff[i] = '\0';
946   sprintf(bouff, "%s%s.pas\0",PathName,FileName);
947   fin = Fopen(bouff, "r");
948
949   for(i=0;i<MAX_WORD_LENGTH+10;i++) tempdirname[i]='\0';
950   sprintf(tempdirname,"%stemp_files_%s%c\0",PathName,FileName,DIRSEPARATOR);
951   mkdir(tempdirname);
952   for(i=0;i<MAX_WORD_LENGTH+10;i++) event_tempdirname[i]='\0';
953   sprintf(event_tempdirname,"%stemp_files_%s_event%c\0",PathName,FileName,DIRSEPARATOR);
954   mkdir(event_tempdirname);
955   bbb = get_next_line(fin, &line);
956   while (strncmp( line.words[0],"type",4 )!= 0){
957     bbb = get_next_line(fin, &line);
958   }
959   bbb = get_next_line(fin, &line);
960   for(i=0;i<MAX_WORD_LENGTH;i++) workingfile->tag[i] ='\0';
961   strcpy(workingfile->tag, line.words[0]);
962   strcpy(Tag,workingfile->tag);
963   fprintf(fout,"%s\n",Tag);
964
965   while (strncmp( line.words[0],"implementation",9 ) != 0){
966     bbb = get_next_line(fin, &line);
967   }
968
969   while(bbb !=EOF){
970     if( strncmp( line.words[0],"procedure",9 ) == 0 || strncmp( line.words[0],"function",8 ) == 0 ){
971           if( strncmp( line.words[0],"procedure",9 ) == 0 ) procedurenum++;
972           else functionnum++;
973           for(i=0;i<MAX_WORD_LENGTH;i++){
974             if (line.words[1][i] == '('){
975               line.words[1][i] = '\0';
976               i=MAX_WORD_LENGTH-2;
977             }
978           }
979           for(i=0;i<MAX_WORD_LENGTH+10;i++) bouff[i] = '\0';
980           sprintf(bouff,"%s%s.txt\0",event_tempdirname,line.words[1] );
981           fwrite = Fopen(bouff, "w");
982
983           while (strncmp(line.words[0], "begin", 5 ) == 0 ) get_next_line(fin, &line);
984           n=a=b=c=d='\0';
985           while (!( n== 'b' && a == 'e' && b== 'g' && c == 'i'&& d == 'n' )){
986             n=a; a=b; b= c; c=d;
987             d = getc(fin);
988           }
989           fprintf(fwrite,"begin");
990           n=a=b=c=d='\0';
991           d = getc(fin);
```

```
992             while (!( n== '\n' && a == 'e' && b== 'n' && c == 'd'&& d == ';' )){
993               if ( d=='\n' ) results_pic++;
994               fputc(d,fwrite);
995               n=a; a=b; b= c; c=d;
996               d = getc(fin);
997             }
998
999             fputc(d,fwrite);
1000             fclose(fwrite);
1001       }
1002       bbb = get_next_line(fin, &line);
1003    }
1004    fclose(fin);
1005 }
```

void print_dfm_list (struct dfmlist * *head*)

Βοηθητική συνάρτηση debuging για την εκτύπωση στην έξοδο τα στοιχεία του μοντέλου.

Ορισμός στη γραμμή 330 του αρχείου dfm2mdl.c.

References dfmlist::fson, και dfmlist::next.

```
Αναφορά από create_final_mdl().330
331    if( head == NULL) return;
332    print_dfm_list(head->fson);
333    print_dfm_list(head->next);
334    return;
335 }
```

void replace_word (char * *word*)

Αντικαθιστά κάποιες λέξεις του PDC με κάποιες άλλες. Οι αντιστοιχίες υπάρχουν στο αρχείο alias.txt.

Ορισμός στη γραμμή 1058 του αρχείου dfm2mdl.c.

References Fopen(), και MAX_WORD_LENGTH.

```
Αναφορά από aliasparser().1058
1059   FILE *aliasfile;
1060   char tmp[MAX_WORD_LENGTH],c;
1061   int i,length;
1062   for(length=0;word[length] != '\0'; length++);
1063   aliasfile = Fopen("obj_spec\\alias.txt", "r");
1064   while(length !=0) {
1065       fscanf(aliasfile,"%s",tmp);
1066       if (strncmp( word,tmp,length-1 ) == 0){
1067           fscanf(aliasfile,"%s",tmp);
1068           for(i=0;i<MAX_WORD_LENGTH;i++) word[i]= '\0';
1069           strcpy(word,tmp);
1070           length =0;
1071           fclose(aliasfile);
1072           return;
1073       }
1074       else  fscanf(aliasfile,"%s",tmp);
1075       c=getc(aliasfile);
1076       if (c == EOF) { length =0;}
1077       else ungetc(c,aliasfile);
1078   }
1079   fprintf(stderr,"\t\t\tno alias for %s\n",word);
1080   fclose(aliasfile);
1081 }
```

int stereotype_classify (char * *stereotype*)

Δίνουμε σαν όρισμα ένα στοιχείο του μοντέλου και συμπληρώνουμε το χαρακτηριστικό το οποίο αφορά το πακέτο στο οποίο ανήκει.

Ορισμός στη γραμμή 158 του αρχείου dfm2mdl.c.

References b, COLOR_DATA, COLOR_NA, COLOR_PRESENTATION_COMPOSITE, COLOR_PRESENTATION_PRIMITIVE, Fopen(), get_next_line(), results_bus, results_cpres, results_data, results_ppres, και line::words.

```
Αναφορά από write_node_b().158                                    {
159   FILE *fcolor;
160   struct line tmpline;
161   char b;
162   int tmp=COLOR_NA;
163   fcolor = Fopen("obj_spec\\data_bussiness_presentation.txt", "r");
164   b=get_next_line(fcolor, &tmpline);
165   while(b !=EOF){
166     if ( strcmp(tmpline.words[0],stereotype) == 0){
167         if ( strncmp(tmpline.words[1],"data",4) == 0){
168             fclose(fcolor);
169             results_data++;
170             return(COLOR_DATA);
171         }
172         if ( strncmp(tmpline.words[1],"composite",9) == 0){
173             fclose(fcolor);
174             results_cpres++;
175             return(COLOR_PRESENTATION_COMPOSITE);
176         }
177         if ( strncmp(tmpline.words[1],"presentation",12) == 0){
178             results_ppres++;
179             fclose(fcolor);
180             return(COLOR_PRESENTATION_PRIMITIVE);
181         }
182         if ( strncmp(tmpline.words[1],"business",8) == 0){
183             fclose(fcolor);
184             results_bus++;
185             return(COLOR_NA);
186         }
187         if ( strncmp(tmpline.words[1],"na",2) == 0){
188             fclose(fcolor);
189             return(COLOR_NA);
190         }
191     }
192     b=get_next_line(fcolor, &tmpline);
193   }
194 fprintf(stderr,"couldn't found colour for: %s \n",stereotype);
195   fclose(fcolor);
196   return (tmp);
197 }
```

void write_assoc_a (struct dfmlist * *head*, int *arg1*, int *arg2*, int *arg3*, int *arg4*)

Τοποθετούμε μία συσχέτιση ανάμεσα σε δύο στοιχεία του μοντέλου (στο part 2 του mdl αρχείου). Ανάλογα με τα ορίσματα arg1-arg4 καθορίζουμε εάν θα είναι απλή συσχέτιση ή aggregtion ή navigable.

Ορισμός στη γραμμή 633 του αρχείου dfm2mdl.c.

References assoc, assoc_1, assoc_2, f_a, Fopen(), fplaceA(), MAX_WORD_LENGTH, dfmlist::objectname, dfmlist::parent, και dfmlist::quid.

```
Αναφορά από append_assoc_to_ab().633                              {
634   struct line tmpline;
635   FILE *fr;
636   char c, buff[MAX_WORD_LENGTH];
637   int i;
638   fr = Fopen("obj_spec\\assoc_a.txt","r");
639   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", assoc);fr= fplaceA(fr, "MY", buff);
640
641   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"51111%i\0", assoc);fr= fplaceA(fr, "ASQUID", buff);
642
```

```
643   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", assoc_2);fr= fplaceA(fr, "MY2", buff);
644   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"71111%i\0", assoc_2);fr= fplaceA(fr, "QUIDMY2",
buff);
645   fr= fplaceA(fr, "CHILDNAME", head->objectname);
646   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"1111111%i\0", head->quid);   fr= fplaceA(fr,
"CHILDQUID", buff);
647   fr= fplaceA(fr, "ARG1", (arg1)? "TRUE" : "FALSE" );
648   fr= fplaceA(fr, "ARG2", (arg2)? "TRUE" : "FALSE" );
649
650
651   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", assoc_1);fr= fplaceA(fr, "MY1", buff);
652   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"61111%i\0", assoc_1);fr= fplaceA(fr, "QUIDMY1",
buff);
653   fr= fplaceA(fr, "PARENTNAME", head->parent->objectname);
654   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"1111111%i\0", head->parent->quid);fr= fplaceA(fr,
"PARENTQUID", buff);
655   fr= fplaceA(fr, "ARG3", (arg3)? "TRUE" : "FALSE" );
656   fr= fplaceA(fr, "ARG4", (arg4)? "TRUE" : "FALSE" );
657
658
659   while( (c = getc(fr)) != EOF ) fputc(c,f_a);
660   fclose(fr);
661 }
```

void write_assoc_b (struct dfmlist * *head*)

Τοποθετούμε τις λεπτομέρειες μίας συσχέτισης ανάμεσα σε δύο στοιχεία του μοντέλου στο part 4 του mdl αρχείου.

Ορισμός στη γραμμή 666 του αρχείου dfm2mdl.c.

References assoc, assoc_1, assoc_2, DX, DY, f_b, Fopen(), fplaceB(), MAX_WORD_LENGTH, dfmlist::parent, dfmlist::quid, dfmlist::x, και dfmlist::y.

Αναφορά από append_assoc_to_ab().666

```
667   FILE *fr;
668   char c, buff[MAX_WORD_LENGTH];
669   int i;
670   fr = Fopen("obj_spec\\assoc_b.txt","r");
671
672   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", assoc);fr= fplaceB(fr, "MY", buff);
673   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", assoc);fr= fplaceB(fr, "ID", buff);
674
675   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"51111%i\0", assoc);fr= fplaceB(fr, "ASQUID", buff);
676
677   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", assoc_2);fr= fplaceB(fr, "MY2", buff);
678   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", assoc_2);fr= fplaceB(fr, "AA2", buff);
679   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", assoc); fr= fplaceB(fr, "ID", buff);
680
681   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"71111%i\0", assoc_2);fr= fplaceB(fr, "QUIDMY2",
buff);
682   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", assoc); fr= fplaceB(fr, "ID", buff);
683   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", head->quid); fr= fplaceB(fr, "CHILDID", buff);
684
685   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", assoc_1);fr= fplaceB(fr, "MY1", buff);
686   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", assoc_1);fr= fplaceB(fr, "AA1", buff);
687   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", assoc); fr= fplaceB(fr, "ID", buff);
688   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"61111%i\0", assoc_1);fr= fplaceB(fr, "QUIDMY1",
buff);
689   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", assoc); fr= fplaceB(fr, "ID", buff);
690   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", head->parent->quid); fr= fplaceB(fr,
"PARENTID", buff);
691
692
693   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0'; sprintf(buff,"%i\0", DX*head->x); fr= fplaceB(fr, "X1",buff);
694   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0'; sprintf(buff,"%i\0", DY*head->y); fr= fplaceB(fr, "Y1",buff);
695   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0'; sprintf(buff,"%i\0", (DX*head->x+DX*head->parent->x)/2); fr=
fplaceB(fr, "X2",buff);
696   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0'; sprintf(buff,"%i\0", DY*head->y); fr= fplaceB(fr, "Y2",buff);
```

```
697   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0'; sprintf(buff,"%i\0", DX*head->parent->x); fr= fplaceB(fr,
"X3",buff);
698   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0'; sprintf(buff,"%i\0", DY*head->parent->y); fr= fplaceB(fr,
"Y3",buff);
699
700   while( (c = getc(fr)) != EOF ) fputc(c,f_b);
701   fclose(fr);
702 }
```

void write_node_a (struct dfmlist * *head*)

Ένα στοιχείο που ανήκει στο μοντέλο το τοποθετούμε κατάλληλα στο part 2 του mdl αρχείου.

Ορισμός στη γραμμή 441 του αρχείου dfm2mdl.c.

References DIRSEPARATOR, event_tempdirname, f_a, Fopen(), fout, fplaceA(), ghead, inher_id, dfmlist::inherit, MAX_WORD_LENGTH, dfmlist::objectname, dfmlist::objecttype, oper_id, dfmlist::parent, dfmlist::quid, results_methods, listfiles::tag, tempdirname, και workingfile.

Αναφορά από append_to_ab().441

```
442   FILE *fr,*document,*foperation, *filepas, *fileinher;
443   char c,t, buff[MAX_WORD_LENGTH+10],bouff[MAX_WORD_LENGTH+10];
444   int i;
445   struct line tmpline;
446   fr = Fopen("obj_spec\\_a.txt","r");
447
448   fr= fplaceA(fr, "OBJECTNAME", head->objectname);
449   sprintf(buff,"1111111%i\0", head->quid);
450   fr= fplaceA(fr, "QUID", buff);
451   fr= fplaceA(fr, "STEREOTYPE", head->objecttype);
452   fr= fplaceA(fr, "DOCUMENTATION", "\0");
453   for(i=0;i<MAX_WORD_LENGTH+10;i++)bouff[i]=buff[i]='\0';
454   sprintf(bouff,"%s\0",tempdirname);
455   strcat(bouff,head->objectname);
456   document = Fopen(bouff, "r");
457   fputc('|', f_a);
458   fputc(' ', f_a);
459
460   while( (c = getc(document)) != EOF ){
461     fputc(c,f_a);
462     if (c == '\n'){
463       fputc('|',f_a);
464       fputc(' ',f_a);
465     }
466   }
467   fclose(document);
468   fprintf(f_a,"\n");
469
470   if (head->inherit == 0 && head->parent != ghead){
471     fileinher = Fopen("obj_spec\\inherit_a.txt","r");
472     fprintf(f_a, "        superclasses   (list inheritance_relationship_list");
473
474     inher_id++;
475     for(i=0;i<MAX_WORD_LENGTH+10;i++) buff[i] = '\0';
476     sprintf(buff, "81%i\0", inher_id);
477     fileinher = fplaceA(fileinher, "QUIDINHERIT", buff);
478     fileinher = fplaceA(fileinher, "INHERITLABEL", "\0");
479     fileinher = fplaceA(fileinher, "PARENTOBJECTNAME", head->parent->objectname);
480     for(i=0;i<MAX_WORD_LENGTH+10;i++) buff[i] = '\0';
481     sprintf(buff, "1111111%i\")\0", head->parent->quid);
482     fileinher = fplaceA(fileinher, "QUIDPARENT", buff);
483
484     fprintf(f_a, ")\n");
485     fclose (fileinher);
486   }
487
488   document = Fopen(bouff, "r");
489   for(i=0;i<MAX_WORD_LENGTH+10;i++) buff[i] = '\0';
490   fscanf(document,"%s", buff);
491   c=buff[0];
```

```
492  while(buff[0] != '\0' ){
493     if ((buff[0] == 'O' && buff[1] == 'n') || (buff[0] == 'A' && buff[1] == 'f' &&
494     buff[2] == 't' &&  buff[3] == 'e' && buff[4] == 'r' ) ||
495     ( buff[0] == 'B' && buff[1] == 'e' && buff[2] == 'f' && buff[3] == 'o' &&
496     buff[4] == 'r' && buff[5] == 'e')){
497           results_methods++;
498           for(i=0;i<MAX_WORD_LENGTH+10;i++)buff[i]='\0';
499           fscanf(document," = %s",buff);
500           for(i=0;i<MAX_WORD_LENGTH+10;i++) boufff[i]='\0';
501           sprintf(bouff,"obj_spec%c_operations_a.txt\0",DIRSEPARATOR);
502           oper_id++;
503           foperation = Fopen(bouff, "r");
504           foperation = fplaceA(foperation, "OPERATIONNAME", buff );
505           sprintf(bouff,"61111%i\0", oper_id);
506           foperation  = fplaceA(foperation, "QUID", bouff);
507           foperation  = fplaceA(foperation, "DOCUMENTATION", "\0");
508           for(i=0;i<MAX_WORD_LENGTH+10;i++) bouff[i] = '\0';
509           sprintf(bouff,"%s%s.%s.txt", event_tempdirname,workingfile->tag,buff );
510           filepas = fopen(bouff, "r");
511           fputc('|',f_a);
512           if (filepas != NULL){
513              while( (t = getc(filepas)) != EOF ){
514                 fputc(t,f_a);
515                 if (t == '\n'){
516                  fputc('|',f_a);
517                  fputc(' ',f_a);
518                 }
519              }
520              fclose(filepas);
521           }
522           else {
523              fprintf(fout,"\ncouldnt read EVENT file: %s\n",bouff);
524              fprintf(f_a," PROBLHMA ME TO FILE: %s",bouff);
525           }
526           while( (t = getc(foperation)) != EOF ) fputc(t,f_a);
527           fclose(foperation);
528
529     }
530     while( (c = getc(document)) != '\n' && c!= EOF);
531     for(i=0;i<MAX_WORD_LENGTH+10;i++)buff[i]='\0';
532     fscanf(document,"%s", buff);
533  }
534  fclose(document);
535
536  fclose(fr);
537  fputc(')',f_a);
538  fputc('\n',f_a);
539 }
```

void write_node_b (struct dfmlist * *head*)

Ένα στοιχείο που ανήκει στο μοντέλο το τοποθετούμε κατάλληλα στο part 4 του mdl αρχείου.

Ορισμός στη γραμμή 544 του αρχείου dfm2mdl.c.

References COLOR_NA, DX, DY, f_b, Fopen(), fplaceB(), ghead, inher_id, dfmlist::inherit, MAX_WORD_LENGTH, dfmlist::objectname, dfmlist::objecttype, dfmlist::parent, dfmlist::quid, stereotype_classify(), dfmlist::x, και dfmlist::y.

Αναφορά από append_to_ab().544

```
545  FILE *fr;
546  char c, buff[MAX_WORD_LENGTH];
547  int i, color=COLOR_NA;
548  fr = Fopen("obj_spec\\_b.txt","r");
549
550  fr= fplaceB(fr, "OBJECTNAME", head->objectname);
551
552  for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", head->quid);fr= fplaceB(fr, "ID", buff);
553
554
```

```
555   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", DX*head->x);
556   fr= fplaceB(fr, "X",buff);
557   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", DY*head->y);
558   fr= fplaceB(fr, "Y",buff);
559
560   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", head->quid);
561   fr= fplaceB(fr, "ID", buff);
562   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", DX*head->x);
563   fr= fplaceB(fr, "X",buff);
564   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", DY*head->y);
565   fr= fplaceB(fr, "Y",buff);
566   fr= fplaceB(fr, "LABEL", head->objectname);
567
568   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", head->quid);
569   fr= fplaceB(fr, "ID", buff);
570   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", DX*head->x);
571   fr= fplaceB(fr, "X",buff);
572   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", DY*head->y-47);
573   fr= fplaceB(fr, "Y47",buff);
574   fr= fplaceB(fr, "STEREOTYPE", head->objecttype);
575
576   color = stereotype_classify(head->objecttype);
577
578   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"%i\0", color);
579   fr= fplaceB(fr, "FCOLOR",buff);
580
581   for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';sprintf(buff,"1111111%i\0", head->quid);
582   fr= fplaceB(fr, "QUID", buff);
583   while( (c = getc(fr)) != EOF ) fputc(c,f_b);
584   fclose(fr);
585
586   if ( head->inherit == 0 && head->parent != ghead){
587     fr = Fopen("obj_spec\\inherit_b.txt","r");
588
589     fr = fplaceB(fr, "INHERITLABEL", "\0");
590
591     for(i=0;i<MAX_WORD_LENGTH+10;i++) buff[i] = '\0';
592     sprintf(buff, "%i\0", inher_id);
593     fr = fplaceB(fr, "IDIH", buff);
594
595     for(i=0;i<MAX_WORD_LENGTH+10;i++) buff[i] = '\0';
596     sprintf(buff, "81%i\0", inher_id);
597     fr = fplaceB(fr, "QUIDINHERIT", buff);
598
599     for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';
600     sprintf(buff,"%i\0", head->quid);
601     fr= fplaceB(fr, "ID", buff);
602
603     for (i=0;i<MAX_WORD_LENGTH;i++) buff[i] = '\0';
604     sprintf(buff,"%i\0", head->parent->quid);
605     fr= fplaceB(fr, "IDPARENT", buff);
606     while( (c = getc(fr)) != EOF ) fputc(c,f_b);
607     fclose(fr);
608
609
610     fputc('\n',f_b);
611   }
612
613 }
```

Τεκμηρίωση Μεταβλητών

int aaa = 0

Αύξων αριθμός και χρησιμοποιείται για την παραγωγή μοναδικού id για κάθε στοιχείο

Ορισμός στη γραμμή 43 του αρχείου dfm2mdl.c.

Αναφορά από aliasparser().int ai

Προσωρινή μεταβλητή int με διάφορες χρήσεις

Ορισμός στη γραμμή 767 του αρχείου dfm2mdl.c.

Αναφορά από make_relations().int assoc = 5111111

Πρόθεμα για την παραγωγή διαφορετικού id που χρειάζεται το αρχείο που περιγράφει το Rational ROSE

Ορισμός στη γραμμή 32 του αρχείου dfm2mdl.c.

Αναφορά από append_assoc_to_ab(), create_final_mdl(), write_assoc_a(), και write_assoc_b().int assoc_1 = 6111111

Πρόθεμα για την παραγωγή διαφορετικού id που χρειάζεται το αρχείο που περιγράφει το Rational ROSE

Ορισμός στη γραμμή 33 του αρχείου dfm2mdl.c.

Αναφορά από append_assoc_to_ab(), create_final_mdl(), write_assoc_a(), και write_assoc_b().int assoc_2 = 7111111

Πρόθεμα για την παραγωγή διαφορετικού id που χρειάζεται το αρχείο που περιγράφει το Rational ROSE

Ορισμός στη γραμμή 34 του αρχείου dfm2mdl.c.

Αναφορά από append_assoc_to_ab(), create_final_mdl(), write_assoc_a(), και write_assoc_b().char b

Προσωρινή μεταβλητή char με διάφορες χρήσεις

Ορισμός στη γραμμή 764 του αρχείου dfm2mdl.c.

Αναφορά από add_operation_text_old(), aliasparser(), create_final_mdl(), fplaceA(), fplaceB(), get_listfiles(), load_dfm(), load_operations(), make_relations(), parser2targetdfm(), prepare_dfm_pas_files(), και stereotype_classify().int curr_id = 11111

Πρόθεμα για την παραγωγή διαφορετικού id που χρειάζεται το αρχείο που περιγράφει το Rational ROSE

Ορισμός στη γραμμή 31 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και make_a_dfmnode().char event_tempdirname[MAX_WORD_LENGTH+10]

Όνομα ενός βοηθητικού directory που αποθηκεύονται μέθοδοι από PIC

Ορισμός στη γραμμή 49 του αρχείου dfm2mdl.c.

Αναφορά από prepare_dfm_pas_files(), και write_node_a().FILE* f_a

Δείκτης αρχείου: αρχείο που αντιστοιχεί στο part 2

Ορισμός στη γραμμή 26 του αρχείου dfm2mdl.c.

Αναφορά από add_operation_text_old(), fplaceA(), generate_ab_parts(), write_assoc_a(), και write_node_a().FILE* f_b

Δείκτης αρχείου: αρχείο που αντιστοιχεί στο part 4

Ορισμός στη γραμμή 27 του αρχείου dfm2mdl.c.

Αναφορά από fplaceB(), generate_ab_parts(), write_assoc_b(), και write_node_b().FILE* f_results

Δείκτης αρχείου: αποτελέσματα

Ορισμός στη γραμμή 28 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και main().FILE* f_results1

Δείκτης αρχείου: αποτελέσματα Μορφή 1

Ορισμός στη γραμμή 29 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και main().FILE* f_results2

Δείκτης αρχείου: αποτελέσματα Μορφή 2

Ορισμός στη γραμμή 30 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και main().FILE* fdfm

Δείκτης αρχείου: dfm από Delphi PDC

Ορισμός στη γραμμή 23 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl().int file2visualize

Παράμετρος που προσδιορίζει εάν κατασκευάζουμε Delphi PSM / PIM / PhP PSM

Ορισμός στη γραμμή 42 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), main(), και make_relations().char FileName[MAX_WORD_LENGTH+10]

Το όνομα του αρχείου

Ορισμός στη γραμμή 47 του αρχείου dfm2mdl.c.

Αναφορά από aliasparser(), create_final_mdl(), main(), parser2targetdfm(), και prepare_dfm_pas_files().struct listfiles * filenames

Λίστα στην οποία περιέχεται πληροφορία για όλα τα αρχεία του δοθέντος κώδικα Delphi

Αναφορά από get_listfiles(), και main().struct listfiles * filenames_end

Λίστα στην οποία περιέχεται πληροφορία για όλα τα αρχεία του δοθέντος κώδικα Delphi

Αναφορά από get_listfiles(), και main().char filenametmp[MAX_WORD_LENGTH]

Προσωρινή μεταβλητή η οποία χρησιμοποιείται για να θυμόμαστε το όνομα ενός προσωρινού αρχείου.

Ορισμός στη γραμμή 763 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και make_relations().struct line flipline

Προσωρινή δομή για την αποθήκευση μία γραμμής σπασμένη σε λέξεις με διάφορες χρήσεις

Ορισμός στη γραμμή 765 του αρχείου dfm2mdl.c.

Αναφορά από make_relations().FILE* fmdl

Δείκτης αρχείου: mdl (Rational ROSE)

Ορισμός στη γραμμή 22 του αρχείου dfm2mdl.c.

Αναφορά από append2mdl(), και create_final_mdl().FILE* fout

Δείκτης αρχείου: η έξοδος των μηνυμάτων λάθους του προγράμματος

Ορισμός στη γραμμή 25 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), Fopen(), generate_ab_parts(), get_listfiles(), main(), prepare_dfm_pas_files(), και write_node_a().FILE* frel = NULL

Με την χρήση αναδρομής διασχίζουμε όλη την δομή στην οποία έχουμε αποθηκεύσει τα στοιχεία του μοντέλου και τοποθετούμε συσχετίσεις. Για να γνωρίζουμε το είδος που ενώνει την κάθε συσχέτιση ανατρέχουμε σε αρχεία που περιγράφουν αναλυτικά τις διάφορες περιπτώσεις. Τα αρχεία αυτά είναι διαχωρισμένα ανά τύπο αναπαράστασης μοντέλου δηλαδή Delphi PSR, PIR και PhP PSR.

Ορισμός στη γραμμή 768 του αρχείου dfm2mdl.c.

Αναφορά από make_relations().FILE* ftmp

Δείκτης αρχείου: προσωρινό αρχείο

Ορισμός στη γραμμή 24 του αρχείου dfm2mdl.c.

Αναφορά από append2mdl(), και get_next_line().int functionnum

Μετράμε το πλήθος των functions στο PIC

Ορισμός στη γραμμή 40 του αρχείου dfm2mdl.c.

Αναφορά από prepare_dfm_pas_files().struct dfmlist * ghead

Στη δομή αυτή αποθηκεύουμε όλα τα στοιχεία του μοντέλου

Αναφορά από append_assoc_to_ab(), append_to_ab(), create_final_mdl(), write_node_a(), και write_node_b().int inher_id = 8111111

Πρόθεμα για την παραγωγή διαφορετικού id που χρειάζεται το αρχείο που περιγράφει το Rational ROSE

Ορισμός στη γραμμή 36 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), write_node_a(), και write_node_b().int nextX

Παράμετρος άξονα X, όπου βοηθά να βρούμε την θέση στην οποία θα τοποθετηθεί το στοιχείο στη οπτικοποίηση του μοντέλου στο Rational Rose

Ορισμός στη γραμμή 37 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και load_dfm().int nextY

Παράμετρος άξονα Ψ, όπου βοηθά να βρούμε την θέση στην οποία θα τοποθετηθεί το στοιχείο στη οπτικοποίηση του μοντέλου στο Rational Rose

Ορισμός στη γραμμή 38 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και load_dfm().int nodenum

Πλήθος των στοιχείων του μοντέλου

Ορισμός στη γραμμή 41 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και make_a_dfmnode().int oper_id = 9111111

Πρόθεμα για την παραγωγή διαφορετικού id που χρειάζεται το αρχείο που περιγράφει το Rational ROSE

Ορισμός στη γραμμή 35 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και write_node_a().char opWords[100][MAX_WORD_LENGTH]

Πίνακας που γεμίζει από ένα αρχείο και είναι τα events που υπάρχουν στην Delphi

Ορισμός στη γραμμή 44 του αρχείου dfm2mdl.c.

Αναφορά από load_operations().char PathName[MAX_WORD_LENGTH+10]

Το path του αρχείου

Ορισμός στη γραμμή 46 του αρχείου dfm2mdl.c.

Αναφορά από aliasparser(), create_final_mdl(), main(), parser2targetdfm(), και prepare_dfm_pas_files().int procedurenum

Μετράμε το πλήθος των procedures στο PDC

Ορισμός στη γραμμή 39 του αρχείου dfm2mdl.c.

Αναφορά από prepare_dfm_pas_files().int results_aggregation

Μεταβλητή απαρίθμησης aggregations του μοντέλου για τα αποτελέσματα

Ορισμός στη γραμμή 50 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), main(), και make_relations().int results_bus

Μεταβλητή απαρίθμησης στοιχείων που ανήκουν στο business πακέτο για τα αποτελέσματα

Ορισμός στη γραμμή 55 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και stereotype_classify().int results_cpres

Μεταβλητή απαρίθμησης στοιχείων που ανήκουν στο composite presentation πακέτο για τα αποτελέσματα

Ορισμός στη γραμμή 57 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και stereotype_classify().int results_data

Μεταβλητή απαρίθμησης στοιχείων που ανήκουν στο data πακέτο για τα αποτελέσματα

Ορισμός στη γραμμή 56 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και stereotype_classify().int results_elements

Μεταβλητή απαρίθμησης στοιχείων του μοντέλου για τα αποτελέσματα

Ορισμός στη γραμμή 52 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), main(), και make_relations().int results_inheritance

Μεταβλητή απαρίθμησης inheritance σχέσεων του μοντέλου για τα αποτελέσματα

Ορισμός στη γραμμή 53 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), main(), και make_relations().int results_methods

Μεταβλητή απαρίθμησης μεθόδων για τα αποτελέσματα

Ορισμός στη γραμμή 51 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), main(), και write_node_a().int results_pdc

Μεταβλητή απαρίθμησης συνολικού πλήθους στοιχείων που ανήκουν στο business πακέτο για τα αποτελέσματα

Ορισμός στη γραμμή 72 του αρχείου dfm2mdl.c.

Αναφορά από aliasparser(), create_final_mdl(), και main().int results_pic

Μεταβλητή απαρίθμησης συνολικού πλήθους στοιχείων που ανήκουν στο business πακέτο για τα αποτελέσματα

Ορισμός στη γραμμή 71 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), main(), και prepare_dfm_pas_files().int results_ppres

Μεταβλητή απαρίθμησης στοιχείων που ανήκουν στο primitive presentation πακέτο για τα αποτελέσματα

Ορισμός στη γραμμή 58 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και stereotype_classify().int results_relation

Μεταβλητή απαρίθμησης απλών σχέσεων (relation) του μοντέλου για τα αποτελέσματα

Ορισμός στη γραμμή 54 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), main(), και make_relations().int results_total_pdc

Μεταβλητή απαρίθμησης συνολικού πλήθους στοιχείων που ανήκουν στο business πακέτο για τα αποτελέσματα

Ορισμός στη γραμμή 74 του αρχείου dfm2mdl.c.

Αναφορά από main().int results_total_php_bus

Μεταβλητή απαρίθμησης συνολικού πλήθους στοιχείων που ανήκουν στο php PSM business πακέτο για τα αποτελέσματα

Ορισμός στη γραμμή 59 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και main().int results_total_php_cpres

Μεταβλητή απαρίθμησης συνολικού πλήθους στοιχείων που ανήκουν στο php PSM composite presentation πακέτο για τα αποτελέσματα

Ορισμός στη γραμμή 61 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και main().int results_total_php_data

Μεταβλητή απαρίθμησης συνολικού πλήθους στοιχείων που ανήκουν στο php PSM data πακέτο για τα αποτελέσματα

Ορισμός στη γραμμή 60 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και main().int results_total_php_ppres

Μεταβλητή απαρίθμησης συνολικού πλήθους στοιχείων που ανήκουν στο php PSM primitive presentation πακέτο για τα αποτελέσματα

Ορισμός στη γραμμή 62 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και main().int results_total_pic

Μεταβλητή απαρίθμησης συνολικού πλήθους στοιχείων που ανήκουν στο business πακέτο για τα αποτελέσματα

Ορισμός στη γραμμή 73 του αρχείου dfm2mdl.c.

Αναφορά από main().int results_total_pim_bus

Μεταβλητή απαρίθμησης συνολικού πλήθους στοιχείων που ανήκουν στο PIM business πακέτο για τα αποτελέσματα

Ορισμός στη γραμμή 67 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και main().int results_total_pim_cpres

Μεταβλητή απαρίθμησης συνολικού πλήθους στοιχείων που ανήκουν στο PIM composite presentation πακέτο για τα αποτελέσματα

Ορισμός στη γραμμή 69 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και main().int results_total_pim_data

Μεταβλητή απαρίθμησης συνολικού πλήθους στοιχείων που ανήκουν στο PIM data πακέτο για τα αποτελέσματα

Ορισμός στη γραμμή 68 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και main().int results_total_pim_ppres

Μεταβλητή απαρίθμησης συνολικού πλήθους στοιχείων που ανήκουν στο PIM primitive presentation πακέτο για τα αποτελέσματα

Ορισμός στη γραμμή 70 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και main().int results_total_psm_bus

Μεταβλητή απαρίθμησης συνολικού πλήθους στοιχείων που ανήκουν στο delphi PSM business πακέτο για τα αποτελέσματα

Ορισμός στη γραμμή 63 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και main().int results_total_psm_cpres

Μεταβλητή απαρίθμησης συνολικού πλήθους στοιχείων που ανήκουν στο delphi PSM composite presentation πακέτο για τα αποτελέσματα

Ορισμός στη γραμμή 65 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και main().int results_total_psm_data

Μεταβλητή απαρίθμησης συνολικού πλήθους στοιχείων που ανήκουν στο delphi PSM data πακέτο για τα αποτελέσματα

Ορισμός στη γραμμή 64 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και main().int results_total_psm_ppres

Μεταβλητή απαρίθμησης συνολικού πλήθους στοιχείων που ανήκουν στο delphi PSM primitive presentation πακέτο για τα αποτελέσματα

Ορισμός στη γραμμή 66 του αρχείου dfm2mdl.c.

Αναφορά από create_final_mdl(), και main().char Tag[MAX_WORD_LENGTH]

Αποθηκεύεται το πρώτο κομμάτι της τοπολογίας του αρχείου που εξετάζουμε στο filesystem

Ορισμός στη γραμμή 45 του αρχείου dfm2mdl.c.

Αναφορά από prepare_dfm_pas_files().char tempdirname[MAX_WORD_LENGTH+10]

Όνομα ενός βοηθητικού directory που βοηθά στο πρώτο στάδιο του parsing ενός dfm σε PSM

Ορισμός στη γραμμή 48 του αρχείου dfm2mdl.c.

Αναφορά από load_dfm(), prepare_dfm_pas_files(), και write_node_a().int wlength

Προσωρινή μεταβλητή int με διάφορες χρήσεις

Ορισμός στη γραμμή 766 του αρχείου dfm2mdl.c.

Αναφορά από make_relations().struct listfiles * workingfile

Λίστα στην οποία περιέχεται πληροφορία για όλα τα αρχεία του δοθέντος κώδικα Delphi

Αναφορά από get_listfiles(), main(), prepare_dfm_pas_files(), και write_node_a().

Ευρετήριο