Τεχνολογικό Εκπαιδευτικό Ίδρυμα Πάτρας
Σχολή Διοίκησης Οικονομίας
Τμήμα Διοίκησης Επιχειρήσεων

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

"Σύγκριση των λογισμικών πακέτων (SOFTWARE) που υπάρχουν για την υποστήριξη δομημένης σχεδίασης αυτόματης παραγωγής λογισμικού (COMPUTER AIDED SOFTWARE ENGINEERING)"

Εισηγητής: Χαλκιόπουλος Αλέξανδρος

Λαΐου Μαρία
Χατζηπαναγιώτη Βασιλεία

Πάτρα, 2002

Περιεχόμενα

Εισαγωγή 5

ΚΕΦΑΛΑΙΟ 1 - Λογισμικό 6

1.1 Τι είναι λογισμικό 6

1.2 Συχνά λάθη και η πραγματικότητα 7
1.2.1 Μύθοι του management 7
1.2.2 Μύθοι του πελάτη 8
1.2.3 Μύθοι του προγραμματιστή 10

1.3 Οι ανθρώπινοι παράγοντες 12

1.4 Ο κύκλος ζωής του λογισμικού 13

1.5 Ο ρόλος της τεχνολογίας λογισμικού 16

1.6 Τεχνολογία συστημάτων 18

1.7 Συμπεράσματα 19

ΚΕΦΑΛΑΙΟ 2-Δομημένη Ανάλυση και Σχεδίαση Συστήματος ή Λογισμικού (Μεθοδολογία) 20

2.1 Ανάλυση προβλήματος (Problem Analysis) 20

2.2 Τεχνικές Συλλογής Γεγονότων (Fact Collection Techniques) 22
2.2.1 Μελέτη εγγράφων τεκμηρίωσης (Documentation Review) 22
2.2.2 Η μέθοδος της συνέντευξης (Interviewing) 23
2.2.3 Η μέθοδος του ερωτηματολογίου (Questioning) 24
2.2.4 Προσωπική Παρατήρηση (Observation) 26
2.2.5 Καταμέτρηση και Εκτίμηση Δεδομένων (Measuring) 28

2.3 Ανάλυση Συστημάτων 29
2.3.1 Ανάλυση Διαδικασιών (Process Analysis) 32
2.3.1.1 Διαγράμματα Ροής Δεδομένων (Data Flow Diagrams) 32
2.3.1.2 Λεξικά Δεδομένων (Data Dictionaries) 39
2.3.1.3 Τεχνικές περιγραφής διαδικασιών 41
2.3.1.3.1 Δέντρα Αποφάσεων 42
2.3.1.3.2 Πίνακες Αποφάσεων 43
2.3.1.3.3 Δομημένα Κείμενα 46
2.3.1.3.4 Σύγκριση των Τεχνικών Περιγραφής Διαδικασιών 47
2.3.2 Ανάλυση Δεδομένων (Data Analysis) 49
2.3.2.1 Τα τμήματα ενός Διαγράμματος Οντοτήτων-Συσχετίσεων 50
2.3.2.2 Το Μοντέλο Οντοτήτων – Συσχετίσεων 53
2.3.2.3 Σύγχρονες Μέθοδοι Μοντελοποίησης (Advanced Modeling Methods) 55
2.3.2.4 Σύνδεση των Δεδομένων με τις Διαδικασίες 57
2.3.2.5 Ανάλυση Λειτουργικής Εξάρτησης 58

2.3.3 Ανάλυση της «ΣΥΜΠΕΡΙΦΟΡΑΣ» ως προς το χρόνο (Time – dependent behavior analysis) 59
2.3.3.1 Κατασκευάζοντας το Διάγραμμα Μετάβασης – Κατάστασης 61
2.3.4 Ισορρόπηση των μοντέλων που εξετάσθηκαν (Balancing the described models) 65
2.3.4.1 Ισορρόπηση του Διαγράμματος Ροής Δεδομένων με το Λεξικό Δεδομένων και με τις Περιγραφές των Διαδικασιών 66
2.3.4.2 Ισορρόπηση των Περιγραφών των Διαδικασιών με το Διάγραμμα Ροής Δεδομένων και το Λεξικό Δεδομένων 67
2.3.4.3 Ισορρόπηση του Διαγράμματος Οντοτήτων – Συσχετίσεων με το Διάγραμμα Ροής Δεδομένων και τις Περιγραφές των Διαδικασιών 67
2.3.4.4. Ισορρόπηση του Διαγράμματος Ροής Δεδομένων με το Διάγραμμα Μετάβασης – Κατάστασης 68
2.4.1 Αρχές Σχεδίασης Συστημάτων 72
2.4.2 Αντικειμενικοί Στόχοι της Σχεδίασης 74
2.4.3 Ανάλυση των σταδίων Σχεδίασης Συστημάτων 76
2.4.4 Ανάπτυξη Σχεδίασης Συστημάτων (Developing a System Design) 78

ΚΕΦΑΛΑΙΟ 3 - Computer Aided Software Engineering (CASE) 80

3.1 Τι είναι τα CASE Tools 80
3.2 Η ιστορία των CASE Tools 80
3.3 Κατηγορίες CASE Tools 83
3.3 Διάφορα Είδη CASE Tools 87
3.3.1 E32 blue river (ένα εργαλείο ανάπτυξης λογισμικού για ενσωματωμένα συστήματα) (www.blue-river-software.com) 87
3.3.2 GeneXus (www.genexus.com) 88
3.3.3 Resolution xCASE (www.xcase.com) 89
3.3.4 DA-C Project Manager της RistanCASE (www.ristancase.com) 90
3.3.5 AxiomSys & AxiomDsn της STG (www.stgcase.com) 91

ΚΕΦΑΛΑΙΟ 4 – Oracle Developer Suite 2000 93

4.1 Oracle Designer 2000 93
4.1.1 Περιγραφή του Oracle Designer 2000 93
4.1.1.1 Μοντελοποίηση Oracle βάσεων δεδομένων 93
4.1.1.2 Δημιουργία βάσεων δεδομένων βασισμένη σε συγκεκριμένα μοντέλα 94
4.1.1.3 Απεικόνιση του σχεδιασμού των υπαρχόντων βάσεων δεδομένων 95
4.1.1.4 Δημιουργία πολυ-λειτουργικών εφαρμογών 95
4.1.1.5 Ανάλυση εξάρτησης 96
4.1.2 Τα εργαλεία του Oracle Designer 2000 97
4.1.2.1 Μοντελοποίηση των απαιτήσεων του συστήματος 97

4.1.2.2 Μετασχηματισμός των προκαταρκτικών σχεδίων εφαρμογών 99
4.1.2.3 Σχεδιασμός και δημιουργία βάσεων δεδομένων και εφαρμογών 100
4.1.2.4 Αποθήκευση και διαχείριση δεδομένων εφαρμογών 101
4.1.3 Τα πλεονεκτήματα του Oracle Designer 2000 102
4.1.3.1 Κεντρική διαχείριση των δεδομένων των εφαρμογών 102
4.1.3.2 Ακριβής ανάλυση των απαιτήσεων του συστήματος 102
4.1.3.3 Ισχυροί μετασχηματιστές βάσεων δεδομένων και εφαρμογών 102
4.1.3.4 Γρήγορη δημιουργία βάσεων δεδομένων και εφαρμογών με τη χρήση γεννητριών 103
4.1.3.5 Αποτελεσματική απεικόνιση των υπαρχόντων μηχανογραφικών εφαρμογών 103
4.1.3.6 Ολοκληρωμένος επαναληπτικός σχεδιασμός και ανάπτυξη εφαρμογών 104
4.1.3.7 Αποτελεσματικά εργαλεία για τη διαχείριση των αποθηκών 104
4.2 Oracle Developer 2000 106
4.2.1 Δυνατότητες και Πλεονεκτήματα 106
4.2.1.1 Παραγωγικότητα 106
4.2.1.2 Διανομή Εφαρμογών μέσω του Web 107
4.2.1.3 Επέκταση σε Ευρύτερη Κλίμακα 108
4.2.1.4 Ανοικτές Εφαρμογές 108
4.2.2 Εισαγωγή στο Project Builder 110
4.2.2.1 Σύνοψη 111
4.2.2.2 Η Ορολογία του Project Builder 111
4.2.2.3 Πώς Επηρεάζει το Project Builder τους Υπάρχοντες Ρόλους Ανάπτυξης 114
4.2.2.4 Τα Πλεονεκτήματα του Project Builder 115
4.2.2.5 Δημιουργία ενός Έργου 120
4.2.3 Εισαγωγή στο Form Builder 122
4.2.3.1 Σύνοψη 122
4.2.3.2 Βασικές Έννοιες 122
4.2.3.3 Διαδικασία 124
4.2.4 Εισαγωγή στο Graphics Builder 125
4.2.4.1 Σύνοψη 125
4.2.4.2 Βασικές Έννοιες 125
4.2.4.3 Διαδικασία 126
4.2.5 Εισαγωγή στο Report Builder 128
4.2.5.1 Σύνοψη 128
4.2.5.2 Βασικές Έννοιες 128
4.2.5.3 Διαδικασία 129
4.2.6 Εισαγωγή στο Procedure Builder 130
4.2.6.1. Σύνοψη 130
4.2.6.2 Βασικές Έννοιες 130
4.2.6.3 Διαδικασία 131
4.2.7 Εισαγωγή στο Query Builder 133
4.2.7.1 Σύνοψη 133

4.2.7.2 Βασικές Έννοιες 133
4.2.7.3 Διαδικασία 134
4.2.8 Εισαγωγή στο Schema Builder 135
4.2.8.1 Σύνοψη 135
4.2.8.2 Βασικές Έννοιες 135
4.2.8.3 Διαδικασία 136
4.2.9 Εισαγωγή στο Translation Builder 137
4.2.9.1 Σύνοψη 137
4.2.9.2 Βασικές Έννοιες 137
4.2.9.3 Διαδικασία 138

ΚΕΦΑΛΑΙΟ 5 – Συμπεράσματα 140

Ελληνική Βιβλιογραφία 142

Ξένη Βιβλιογραφία 142

Παραρτήματα 144

1. Διάγραμμα Οντοτήτων – Συσχετίσεων (Entity Relationship Diagram) 145
2. Σχηματικό Διάγραμμα Δεδομένων (Data Schema Diagram) 146
3. SQL Code (Κώδικας SQL) 147

Εισαγωγή

Στη σημερινή εποχή, στους συντελεστές παραγωγής (φύση, εργασία, κεφάλαιο, τεχνολογία, κλπ) προστέθηκε ένας νέος συντελεστής γνωστός ως πληροφοριακός πόρος ή πληροφορία. Αυτό συνέβη, διότι οι λειτουργίες της διοίκησης των επιχειρήσεων βασίζονται όλο και περισσότερο στην αποτελεσματική χρησιμοποίηση της πληροφορίας και στα συστήματα που την παρέχουν, δηλαδή τα Πληροφοριακά Συστήματα Διοίκησης.

Για παράδειγμα, η επιχείρηση για τον προγραμματισμό της χρησιμοποιεί πληροφορίες από το εξωτερικό και εσωτερικό της περιβάλλον, ώστε να μπορέσει να εξισορροπήσει με τον καλύτερο δυνατό τρόπο τις παρουσιαζόμενες ευκαιρίες και απειλές του εξωτερικού περιβάλλοντός της με τις δυνάμεις και αδυναμίες της.

Η πληροφορία είναι επίσης στενά συνδεδεμένη με την οργανωσιακή δομή, αφού τα σπουδαιότερα προβλήματα που παρουσιάζονται στην επικοινωνία οφείλονται στην φτωχή ροή πληροφοριών. Όμως και η διεύθυνση βασίζεται στην πληροφορία, με τη βοήθεια της οποίας επικοινωνούν οι εργαζόμενοι, ώστε να κατανοήσουν καλύτερα τους στρατηγικούς στόχους της επιχείρησης.

Τέλος, τα διευθυντικά στελέχη δεν θα μπορούσαν να ελέγξουν και να συντονίσουν τα διάφορα τμήματα της επιχείρησης χωρίς ακριβή, έγκυρη και επίκαιρη πληροφόρηση.

Η εργασία που ακολουθεί ξεκινά με τον ορισμό και τη γενική περιγραφή του Λογισμικού και συνεχίζει αναπτύσσοντας τη μεθοδολογία της Δομημένης Ανάλυσης και Σχεδίασης Συστήματος ή Λογισμικού. Ακολουθεί εκτενής αναφορά στα εργαλεία Computer Aided Software Engineering (CASE TOOLS) που υποστηρίζουν τη Δομημένη Ανάλυση, Σχεδίαση και Υλοποίηση Συστημάτων και ολοκληρώνεται με την ανάλυση ενός τέτοιου εργαλείου, του Oracle Developer Suite 2000, το οποίο αποτελεί ένα από τα πιο παραγωγικά και αποτελεσματικά εργαλεία CASE.

ΚΕΦΑΛΑΙΟ 1 - Λογισμικό

1.1 Τι είναι λογισμικό

Με τα νέα δεδομένα στη βιομηχανία των ηλεκτρονικών υπολογιστών ο ρυθμός ανάπτυξης της τεχνολογίας τους είναι εξίσου εντυπωσιακός με τον ρυθμό που εισβάλουν τόσο στην καθημερινή ζωή όσο και στα μεγάλα κέντρα αποφάσεων. Η προοπτική μιας βιομηχανοποιημένης οικονομίας εξαρτάται σε μεγάλο βαθμό από την αποτελεσματική χρήση της τεχνολογίας λογισμικού.

Με τον όρο λογισμικό εννοούμε :

- *Σύνολο εντολών* προς τον υπολογιστή (πρόγραμμα), που όταν εκτελεστούν παρέχουν επιθυμητές λειτουργίες και αποδόσεις
- *Δομές δεδομένων* που επιτρέπουν την ικανοποιητική διαχείριση πληροφοριών
- *Έγγραφα* τα οποία περιγράφουν τη λειτουργία και χρήση των προγραμμάτων

Όσον αφορά στον όρο *τεχνολογία λογισμικού (software engineering)*, πρωτοεμφανίστηκε στα τέλη του 1960, κατά τη διάρκεια μίας σύσκεψης η οποία πραγματευόταν την αυξανόμενη ανάγκη για νέο λογισμικό που είχε παρατηρηθεί μετά την εμφάνιση των υπολογιστών 3ης γενιάς.

Οι πρώτες εμπειρίες σχετικά με την παραγωγή μεγάλων πακέτων λογισμικού έδειχναν ότι οι υπάρχουσες μέθοδοι δεν ήταν ικανοποιητικές. Τεχνικές που χρησιμοποιούνταν για την παραγωγή μικρών πακέτων λογισμικού δεν μπορούσαν να αναχθούν στα μεγάλα πακέτα. Η ανάγκη εύρεσης νέων μεθόδων ήταν επιτακτική. Σήμερα, παρ' όλες τις προόδους που έχουν παρατηρηθεί, η κρίση που υπάρχει στον τομέα παραγωγής λογισμικού δεν έχει λυθεί. Πολλά από τα λάθη που είχαν παρατηρηθεί στις αρχές του 1970 παρατηρούνται ακόμα. Έτσι, θα έλεγε κανείς ότι απαιτούνται νέα εργαλεία, μέθοδοι και τεχνικές και κυρίως, καλύτερη εκπαίδευση

προσωπικού, ώστε να μπορεί η βιομηχανία παραγωγής λογισμικού να ανταποκριθεί στις ανάγκες της αγοράς.

1.2 Συχνά λάθη και η πραγματικότητα

Πολλά από τα προβλήματα με το λογισμικό μπορούν να αποδοθούν σε μία μυθολογία που προέκυψε κατά τη διάρκεια της ιστορίας των πρώτων χρόνων της ανάπτυξης του λογισμικού. Δυστυχώς τέτοιοι μύθοι σχετιζόμενοι με το λογισμικό προωθούν τη λανθασμένη πληροφόρηση και τη σύγχυση.

Σήμερα, οι περισσότεροι ειδικοί ερμηνεύουν τους μύθους αυτούς σαν συμπεριφορές που αποπροσανατολίζουν και που έχουν προκαλέσει σοβαρά προβλήματα για τους managers και τους τεχνικούς. Παρ' όλα αυτά, επειδή είναι δύσκολο να αλλάζουν παλιές συμπεριφορές και συνήθειες γίνονται ακόμα πιστευτοί μύθοι για το λογισμικό αν και διανύουμε την 5η δεκαετία του λογισμικού.

1.2.1 Μύθοι του management

Οι managers που είναι υπεύθυνοι για το λογισμικό, όπως κάθε manager, βρίσκονται συχνά κάτω από μεγάλη πίεση για να διατηρούν προϋπολογισμούς, να κρατούν τις προθεσμίες μέσα στα όρια και να βελτιώνουν την ποιότητα. Σε δύσκολες καταστάσεις ο manager λογισμικού συχνά κρατιέται από την πίστη κάποιου από τους μύθους λογισμικού, ιδιαίτερα όταν αυτή η πίστη μειώνει την πίεση (έστω και προσωρινά).

- *Μύθος* : υπάρχουν στην εταιρεία μου βιβλία γεμάτα με πρότυπα και διαδικασίες για το χτίσιμο λογισμικού. Αυτό δε θα παρέχει στους ανθρώπους μου όλα όσα θα πρέπει να ξέρουν ;
- *Πραγματικότητα* : τα βιβλία αυτά υπάρχουν αλλά χρησιμοποιούνται ; Ξέρουν όλοι για την ύπαρξή τους ; Αντανακλά τη σύγχρονη πρακτική ανάπτυξης λογισμικού ; Είναι πλήρη ; Στις πιο πολλές περιπτώσεις η απάντηση σε όλα αυτά τα ερωτήματα είναι «όχι».

- *Μύθος* : οι δικοί μου άνθρωποι έχουν state of the art εργαλεία ανάπτυξης λογισμικού. Τους αγοράζουμε πάντα τους πιο σύγχρονους υπολογιστές.
- *Πραγματικότητα* : χρειάζεται περισσότερο από τον τελευταίο τύπο εξοπλισμού σε υλικό για να γίνει υψηλής ποιότητας ανάπτυξη λογισμικού. Εργαλεία *CASE (Computer Aided Software Engineering)* είναι πιο σημαντικά από το υλικό για την επίτευξη καλής ποιότητας και παραγωγικότητας. Η πλειοψηφία των ανθρώπων που κάνουν ανάπτυξη λογισμικού ακόμη δεν τα χρησιμοποιούν.

- *Μύθος* : αν βγούμε από τον χρονοπρογραμματισμό μας, τότε μπορούμε να προσθέσουμε περισσότερους προγραμματιστές και να προλάβουμε.
- *Πραγματικότητα* : η ανάπτυξη λογισμικού δεν είναι μία μηχανιστική διαδικασία παραγωγής. Καθώς προσθέτουμε νέους ανθρώπους θα πρέπει να καταναλωθεί χρόνος για την εκπαίδευσή τους από τους ανθρώπους που ήδη δουλεύουν. Αυτό μειώνει το συνολικό χρόνο που δαπανάται για την παραγωγική ανάπτυξη. Σε τελική ανάλυση μπορούν να προστεθούν άνθρωποι, αλλά μόνο με ένα σχεδιασμένο και καλά συντονισμένο τρόπο.

1.2.2 Μύθοι του πελάτη

Ο πελάτης που απαιτεί λογισμικό υπολογιστών μπορεί να είναι μία ομάδα τεχνικών, το τμήμα πωλήσεων/marketing, ή μία άλλη εταιρεία που έχει απαιτήσει λογισμικό κάτω από ένα συμβόλαιο. Στις περισσότερες περιπτώσεις, ο πελάτης πιστεύει στους μύθους που αφορούν το λογισμικό γιατί οι υπεύθυνοι πολλές φορές δεν κάνουν πολλά για να αποκαταστήσουν την παραπληροφόρηση. Οι μύθοι οδηγούν σε λανθασμένες απαιτήσεις (από τον πελάτη) και τελικά σε δυσαρέσκεια με τον άνθρωπο που έκανε την ανάπτυξη.

- *Μύθος* : μια γενική διατύπωση των στόχων είναι επαρκής για την αρχή της συγγραφής προγραμμάτων, οι λεπτομέρειες υλοποιούνται αργότερα.
- *Πραγματικότητα* : η μεγαλύτερη αιτία για αποτυχημένες προσπάθειες στην ανάπτυξη λογισμικού είναι ο ανεπαρκής αρχικός ορισμός. Είναι απαραίτητη μία λεπτομερής και δομημένη περιγραφή των πληροφοριών, των διαδικασιών, της απόδοσης, των περιορισμών σχεδιασμού και του κριτηρίου εγκυρότητας. Αυτά τα χαρακτηριστικά μπορούν να καθοριστούν μόνο μετά από μία ολοκληρωμένη επικοινωνία μεταξύ του πελάτη και του ανθρώπου που κάνει την ανάπτυξη.

- *Μύθος* : οι απαιτήσεις ενός έργου αλλάζουν. Οι αλλαγές μπορεί να αντιμετωπιστούν λόγω της ευελιξίας που έχει το λογισμικό.
- *Πραγματικότητα* : είναι αληθινό ότι οι απαιτήσεις του λογισμικού αλλάζουν, όμως η επίδραση που έχει η αλλαγή διαφέρει ανάλογα με το χρόνο που εφαρμόστηκε η αλλαγή αυτή. Αν δοθεί ιδιαίτερη προσοχή στον αρχικό ορισμό, τότε μπορούν εύκολα να αντιμετωπιστούν και να διεκπεραιωθούν αλλαγές που απαιτούνται σχετικά νωρίς. Ο πελάτης μπορεί να επανεξετάσει τις απαιτήσεις και να προτείνει αλλαγές με σχετική μικρή επίδραση στο κόστος. Όταν οι αλλαγές απαιτούνται να γίνουν κατά τη διάρκεια της σχεδίασης του λογισμικού, τότε η επίδραση στο κόστος μεγαλώνει ραγδαία. Οι πόροι έχουν ανατεθεί και έχει επιτευχθεί ένα σχεδιαστικό πλαίσιο. Η συλλογή μπορεί να προκαλέσει αναταραχή η οποία απαιτεί επιπλέον πόρους και σημαντικές σχεδιαστικές τροποποιήσεις π.χ. επιπλέον κόστος. Αλλαγές στη λειτουργία, την απόδοση, των διαμέσων ή άλλων χαρακτηριστικών στη διάρκεια της εφαρμογής (κώδικας και έλεγχος) έχουν σοβαρή επίδραση στο κόστος. Η αλλαγή που απαιτείται στα τελευταία στάδια ενός έργου, μπορεί να κοστίζει μία τάξη μεγέθους πιο

πολύ από την ίδια αλλαγή η οποία απαιτείται να γίνει στα πρωταρχικά στάδια.

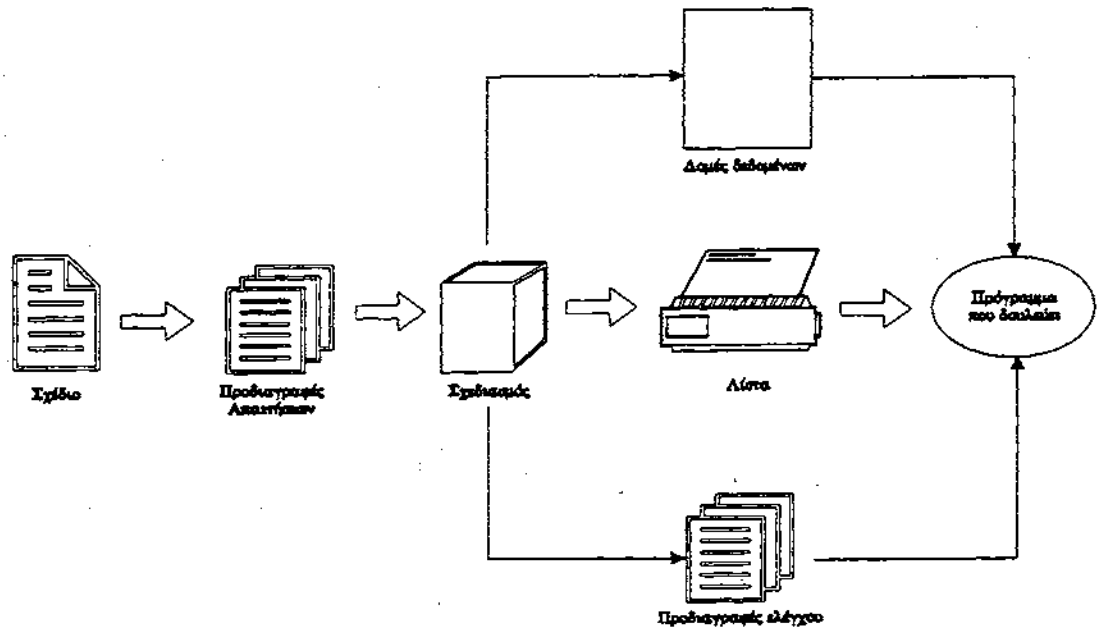
1.2.3 Μύθοι του προγραμματιστή

Στις πρώτες μέρες ανάπτυξης του λογισμικού ο προγραμματισμός είχε θεωρηθεί σαν μία μορφή τέχνης.

- *Μύθος* : Η δουλειά μας έχει τελειώσει από τη στιγμή που γράψαμε το πρόγραμμα και το τρέχουμε.
- *Πραγματικότητα* : Έχει ειπωθεί κάποτε ότι «όσο πιο γρήγορα ξεκινήσει να γράφει κανείς κώδικα, τόσο πιο πολύ θα κάνει να τελειώσει». Βιομηχανικά δεδομένα δείχνουν ότι μεταξύ 50 και 70 τοις εκατό από την όλη προσπάθεια που καταναλώνεται πάνω σε ένα πρόγραμμα θα καταναλωθεί μετά την πρώτη παράδοση λογισμικού στον πελάτη.

- *Μύθος* : Δεν μπορώ να καθορίσω την ποιότητα μέχρι να τρέξει το πρόγραμμα.
- *Πραγματικότητα* : Ένας από τους πιο αποτελεσματικούς μηχανισμούς εγγύησης της ποιότητας του λογισμικού μπορεί να εφαρμοσθεί από την έναρξη ενός έργου τυπική, τεχνική ανασκόπηση. Οι ανασκοπήσεις του λογισμικού αποτελούν ένα φίλτρο ποιότητας και έχουν βρεθεί ότι είναι περισσότερο αποτελεσματικές από τον έλεγχο για την ανίχνευση ορισμένων κλάσεων ελαττωμάτων λογισμικού.

- *Μύθος* : Το μόνο παραδοτέο για ένα πετυχημένο έργο είναι το πρόγραμμα που τρέχει.
- *Πραγματικότητα* : Το πρόγραμμα που τρέχει είναι ένα μέρος από την εγκατάσταση λογισμικού η οποία περιλαμβάνει :



Οι φόρμες τεκμηρίωσης αποτελούν τη βάση για μία επιτυχημένη ανάπτυξη και, κυρίως, μας παρέχουν κατευθυντήριες γραμμές για το έργο της συντήρησης του λογισμικού.

1.3 Οι ανθρώπινοι παράγοντες

Οι ανθρώπινοι παράγοντες συνήθως αγνοούνται από τους μηχανικούς λογισμικού. Η γνώση στο παραπάνω θέμα παρέχει μια πιο ολοκληρωμένη αντίληψη σε πολλές περιοχές της τεχνολογίας λογισμικού και μας βοηθά στις τεχνικές διαδικασίες του σχεδιασμού του συστήματος και στο management.

Υπάρχει ανωριμότητα στο συγκεκριμένο πεδίο μελέτης και αβεβαιότητα στα συμπεράσματα άλλων ψυχολογικών ερευνών. Η μελέτη ανθρωπίνων παραγόντων είναι σημαντική για τους εξής λόγους :

- Είναι σημαντική μία αποτελεσματική διαχείριση του λογισμικού αν έχουμε στόχο μεγάλα προγραμματιστικά έργα που πρέπει να ολοκληρωθούν σε συγκεκριμένα χρονικά πλαίσια και να βρίσκονται μέσα στον προϋπολογισμό. Η αποτελεσματικότητα του manager λογισμικού συνίσταται από την ικανότητα να κατανοεί κάθε έναν από το προσωπικό ξεχωριστά και να εκλαμβάνει τον τρόπο με τον οποίο τα ξεχωριστά αυτά άτομα αλληλεπιδρούν μεταξύ τους.
- Τα συστήματα υπολογιστών χρησιμοποιούνται από ανθρώπους. Αν κατά τη διάρκεια του σχεδιασμού του συστήματος δε ληφθούν υπόψη οι ικανότητες και οι περιορισμοί των ανθρώπων αυτών, τότε το τελικό σύστημα δε θα χρησιμοποιηθεί με τον καλύτερο δυνατό τρόπο από τους ανθρώπους.
- Η παραγωγικότητα του προγραμματιστή είναι κρίσιμος παράγοντας κόστους στην τεχνολογία λογισμικού. Η κατανόηση των ανθρωπίνων παραγόντων βοηθά στο να βρεθούν τρόποι για την αύξηση της παραγωγικότητας με μικρό σχετικά κόστος.

Οι ανθρώπινοι παράγοντες που έχουν σχέση με τη τεχνολογία λογισμικού περιλαμβάνουν :

- τον τρόπο με τον οποίο επεξεργαζόμαστε την πληροφορία
- τα ιδιαίτερα χαρακτηριστικά του ατόμου και της ομάδας, και

- την εργονομία του χώρου εργασίας.

1.4 Ο κύκλος ζωής του λογισμικού

Όπως όλα τα μεγάλα συστήματα, έτσι και τα μεγάλα συστήματα λογισμικού απαιτούν ένα μεγάλο χρονικό διάστημα για την ανάπτυξή τους και στην πλειοψηφία τους χρησιμοποιούνται για πολύ περισσότερο. Σε όλο αυτό το χρονικό διάστημα ανάπτυξης και χρήσης μπορούν να διαχωριστούν κάποια βασικά στάδια τα οποία όπως έχει δείξει η πείρα είναι κοινά αλλά και θεμελιώδη σε όλα τα συστήματα. Όλα αυτά τα στάδια μαζί αποτελούν τον *«κύκλο ζωής λογισμικού»*.

Η ιδέα του κύκλου ζωής του λογισμικού, που αναφέρεται και σαν «μοντέλο καταρράκτη», προτάθηκε για πρώτη φορά το 1970 από τον Royce αλλά από τότε έχουν εμφανιστεί πολλές εκλεπτύνσεις και παραλλαγές. Ένας βασικός κύκλος ζωής παρουσιάζεται στο παρακάτω σχήμα και αποτελείται από 6 στάδια :

1ο Στάδιο : Ανάλυση του περιβάλλοντος συστήματος

Το λογισμικό σχεδόν ποτέ δεν υπάρχει αυτόνομο αλλά είναι μέρος ενός μεγαλύτερου συστήματος γι' αυτό και πρέπει να ξεκινήσει η ανάπτυξή του.

2ο Στάδιο : Ανάλυση των απαιτήσεων του λογισμικού

Για να μπορέσει ο μηχανικός λογισμικού να κατανοήσει τη φύση των προγραμμάτων που πρέπει να κατασκευαστούν πρέπει να συγκεντρώσει τις απαιτήσεις που έχουν οι τελικοί χρήστες από το σύστημα. Έτσι, θα κατανοήσει τις υπηρεσίες που θα πρέπει να προσφέρει το σύστημα, τους περιορισμούς του και το πεδίο ορισμού των πληροφοριών με τις οποίες θα τροφοδοτείται το σύστημα στην είσοδό του. Επίσης, εξετάζει θέματα απόδοσης (π.χ. αν είναι απαραίτητη η ταχύτητα λόγω on – line υπηρεσιών) και τρόπους επικοινωνίας του λογισμικού με τον τελικό χρήστη.

Αυτό το στάδιο απαιτεί συνεργασία και συζήτηση με τους μελλοντικούς χρήστες, με σκοπό τη σαφή διατύπωση των απαιτήσεων τους από το λογισμικό και καταγραφή τους σε μορφή κατανοητή από τον πελάτη.

3ο Στάδιο : Σχεδιασμός (Design)

Στο στάδιο αυτό γίνεται η αναπαράσταση των λειτουργιών του συστήματος με τρόπο που είναι δυνατή η μετατροπή τους σε ένα ή περισσότερα προγράμματα. Γίνεται δηλαδή μία μετατροπή των αποτελεσμάτων της προηγούμενης φάσης σε μία πιο αυστηρή μορφή, της οποίας μπορεί να ελεγχθεί η ποιότητα πριν περάσουμε στο στάδιο της υλοποίησης.

Συνήθως ο σχεδιασμός διαχωρίζει 4 βασικές ιδιότητες του λογισμικού.

1. Δομές δεδομένων. Είναι ο τρόπος αναπαράστασης της εξωτερικής πληροφορίας σε μορφή κατάλληλη για επεξεργασία.
2. Αρχιτεκτονική του λογισμικού. Είναι η φιλοσοφία σχεδίασης του όλου συστήματος που μπορεί να γίνεται με τμηματοποίηση του συστήματος ή με προσανατολισμό στα αντικείμενα κ.ο.κ.
3. Ανάλυση σε καθορισμένο βαθμό λεπτομέρειας των λειτουργιών και διατύπωσή τους σε διαδικαστική γλώσσα (ίσως ψευδοκώδικα) ώστε να είναι εύκολη η μετάβαση σε μία γλώσσα προγραμματισμού.
4. Ανάλυση του τμήματος επικοινωνίας με τον χρήστη που θα χρειαστεί και ταξινόμησή του σε κάποια από τις ήδη γνωστές κατηγορίες.

4ο Στάδιο : Υλοποίηση – Προγραμματισμός

Μεταγραφή της φάσης σχεδιασμού σε μορφή κατανοητή από το hardware. Το εργαλείο γι' αυτό είναι μία γλώσσα προγραμματισμού υψηλού επιπέδου όπως οι Pascal, C, Fortran αλλά αρκετά συχνά χρησιμοποιούνται *CASE Tools*. Ο τρόπος με τον οποίο θα γίνει αυτό εξαρτάται από τη σχεδίαση.

Με λεπτομερή σχεδίαση ο προγραμματισμός μπορεί να επιτευχθεί σχεδόν μηχανικά ενώ αντίθετα μπορεί η σχεδίαση να αφήνει εσκεμμένα πολλά περιθώρια στον προγραμματιστή. Για παράδειγμα, αν πρέπει υποχρεωτικά ένα μέρος του λογισμικού να υλοποιηθεί σε κώδικα μηχανής, τότε η σχεδίαση σίγουρα θα δώσει μία αυστηρή περιγραφή αλλά βέβαια δεν μπορεί να μπει σε λεπτομέρειες υλοποίησης που εξαρτώνται από το υλικό.

Μία καλή σχεδίαση σπάει το σύστημα σε επιμέρους κομμάτια (modules) καθένα από τα οποία υλοποιείται ξεχωριστά. Μετά την υλοποίηση του, το κάθε κομμάτι πρέπει να ελέγχεται διεξοδικά για τυχόν λάθη, ώστε να επιβεβαιωθεί ότι η συμπεριφορά του είναι αυτή που έχει οριστεί από τη φάση της σχεδίασης.

5ο Στάδιο : Έλεγχος συστήματος (Testing)

Ενώ τα διάφορα κομμάτια έχουν ελεγχθεί δεν είναι σίγουρο ότι συνεργάζονται αρμονικά μεταξύ τους. Σε αυτό το στάδιο όλα τα κομμάτια διασυνδέονται και το σύστημα ελέγχεται στην πλήρη και λειτουργική μορφή του. Ελέγχονται όλες οι πιθανές είσοδοι για να διαπιστωθεί κατά πόσο παράγονται οι επιθυμητές έξοδοι.

6ο Στάδιο : Συντήρηση

Μετά τη παράδοση του τελικού προϊόντος είναι σίγουρο ότι θα χρειαστεί αλλαγές, είτε λόγω των λαθών που θα έχει είτε λόγω των αλλαγών που συμβαίνουν στο σύστημα (νέο λειτουργικό, νέα περιφερειακά), είτε ακόμα λόγω απαίτησης του πελάτη για αυξημένη απόδοση και λειτουργικότητα. Η συντήρηση επαναλαμβάνει τα προηγούμενα βήματα πάνω όμως σε ολοκληρωμένο σύστημα.

1.5 Ο ρόλος της τεχνολογίας λογισμικού

Το μοντέλο του κύκλου ζωής του λογισμικού δεν είναι το μόνο που χρησιμοποιείται. Υπάρχουν πολλοί διαδεδομένοι τρόποι αντιμετώπισης του θέματος. Όποια προσέγγιση κι αν ακολουθηθεί υπάρχουν τρεις γενικές φάσεις στην ανάπτυξη του λογισμικού. Είναι η φάση του *ορισμού*, της *ανάπτυξης* και της *συντήρησης*.

Η φάση του *ορισμού* περιλαμβάνει τα εξής βήματα :

- την ανάλυση του συστήματος,
- την ανάλυση του έργου και ανάθεση υποέργων και τέλος
- την ανάλυση των απαιτήσεων.

Η φάση της *ανάπτυξης* περιλαμβάνει :

- το σχεδιασμό του λογισμικού,
- την κωδικοποίηση και
- τον έλεγχο του λογισμικού.

Η φάση της *συντήρησης* περικλείει δραστηριότητες όπως :

- διόρθωση λαθών,
- προσαρμογή στην εξέλιξη του συστήματος και
- λειτουργικές βελτιώσεις

Πολλές φορές χρησιμοποιείται η μέθοδος της «αναστροφής της ανάπτυξης» κατά την οποία με ειδικά προγράμματα (CASE TOOLS) ανακαλύπτουμε τη δομή και τον τρόπο λειτουργίας ενός παλιού συστήματος και του λογισμικού του ώστε να είναι δυνατή η εφαρμογή των παραπάνω βημάτων.

Οι φάσεις και τα βήματα που προαναφέρθηκαν, συμπληρώνονται από παράλληλες διαδικασίες. Αυτές μπορεί να είναι σύνταξη αναφορών για την εξασφάλιση της ποιότητας, τεκμηρίωση για το σύστημα και το λογισμικό ή και ένα σύστημα ελέγχου των αλλαγών πάνω στο λογισμικό ώστε οι αλλαγές κατά την ανάπτυξη αλλά και ύστερα να είναι δυνατό να ελέγχονται και να αρχειοθετούνται.

Αν όλα τα παραπάνω τελικά γίνουν δε σημαίνει ότι ο τρόπος αυτός είναι καθορισμένος. Ο ρόλος της τεχνολογίας λογισμικού είναι να δώσει μία προσέγγιση στην ανάπτυξη του λογισμικού η οποία θα τονίζει την πειθαρχία και τις καλά οργανωμένες μεθοδολογίες κατά την εκτέλεση των παραπάνω φάσεων και διαδικασιών.

1.6 Τεχνολογία συστημάτων

Η τεχνολογία λογισμικού και υλικού ανήκουν σε μία ευρύτερη κατηγορία της τεχνολογίας συστημάτων. Οι δύο παραπάνω κατευθύνσεις αντιπροσωπεύουν την προσπάθεια να μπει σε τάξη η ανάπτυξη συστημάτων βασιζόμενα σε υπολογιστές.

Τα στοιχεία ενός τέτοιου συστήματος είναι :

- **Λογισμικό** : προγράμματα, δομές δεδομένων και σχετιζόμενη τεκμηρίωση τα οποία έχουν σα στόχο να επηρεάζουν τη λογική μέθοδο, διαδικασία ή τον έλεγχο που απαιτείται.
- **Υλικό** : ηλεκτρονικές συσκευές (π.χ. Κεντρική Μονάδα Επεξεργασίας, Μνήμη) οι οποίες παρέχουν υπολογιστική ικανότητα και ηλεκτρομηχανικές συσκευές (π.χ. αισθητήρες, μοτέρ, αντλίες) που παρέχουν απαραίτητα σήματα.
- **Άνθρωποι** : χρήστες και χειριστές του υλικού και λογισμικού
- **Βάση δεδομένων** : μία μεγάλη και οργανωμένη συλλογή από πληροφορίες.
- **Τεκμηρίωση** : εγχειρίδια, φόρμες και οποιαδήποτε άλλη περιγραφική πληροφορία που μας δείχνει τη χρήση ή τη λειτουργία του συστήματος.
- **Διαδικασίες** : τα βήματα που ορίζουν την ειδική χρήση του κάθε στοιχείου του συστήματος.

1.7 Συμπεράσματα

Λόγω του μεγάλου και συνεχώς αυξανόμενου όγκου των πληροφοριών που βρίσκονται στη διάθεση της διοίκησης και της συνεχώς αυξανόμενης πολυπλοκότητας, αλλά και του ρυθμού με τον οποίο γίνονται οι αλλαγές που χαρακτηρίζουν τη σημερινή κοινωνία, η διοίκηση των οικονομικών μονάδων είναι όλο ένα και περισσότερο υποχρεωμένη να χρησιμοποιεί Συστήματα Πληροφοριών Διοίκησης όπου είναι δυνατό για να μπορέσει να ανταποκριθεί στον επιταχυνόμενο ρυθμό των κοινωνικών και τεχνολογικών αλλαγών που έχουν συντομεύσει δραστικά τον κύκλο ζωής των προϊόντων, των μεθόδων παραγωγής, των μεθόδων και πρακτικών διοίκησης προσωπικού, των οργανωτικών σχέσεων και σχημάτων κλπ.

Το ζητούμενο για τη διοίκηση της σύγχρονης επιχείρησης είναι να μπορεί να έχει πρόσβαση στην κατάλληλη πληροφόρηση την κατάλληλη στιγμή και στην κατάλληλη μορφή, ώστε να μπορεί να λάβει αποφάσεις που θα συμβάλλουν ουσιαστικά στο στρατηγικό σχεδιασμό.

Για το λόγο αυτό το σύγχρονο Πληροφοριακό Σύστημα Διοίκησης πρέπει να διαθέτει τα ακόλουθα χαρακτηριστικά:

- Να είναι ευμετάβλητο – να μπορεί δηλαδή με εύκολο τρόπο να μεταβάλλεται έτσι ώστε να μπορεί να εξάγει τα δεδομένα που έχει ανάγκη η διοίκηση χωρίς να απαιτείται εκ νέου σχεδιασμός του και χρονοβόρος ανάπτυξη
- Να είναι αξιόπιστο – δηλαδή η πληροφόρηση να είναι έγκυρη και να στηρίζεται σε αξιόπιστα και καλώς ορισμένα δεδομένα τα οποία είναι οργανωμένα και δομημένα
- Να είναι τυποποιημένο – δηλαδή να μην αποτελεί προϊόν εξειδικευμένης ανάπτυξης συγκεκριμένης ομάδας τεχνικών πληροφορικής, αλλά να είναι ανεπτυγμένο σύμφωνα με συγκεκριμένες αρχές τυποποίησης και με τυποποιημένα και ευρέως γνωστά εργαλεία ανάπτυξης

Στην επίτευξη των παραπάνω συμβάλλει ουσιαστικά η *Δομημένη Ανάλυση και Σχεδίαση Συστημάτων ή Λογισμικού*.

ΚΕΦΑΛΑΙΟ 2-Δομημένη Ανάλυση και Σχεδίαση Συστήματος ή Λογισμικού (Μεθοδολογία)

2.1 Ανάλυση προβλήματος (Problem Analysis)

Κάνοντας μια προσπάθεια για καλύτερη προσέγγιση του κύκλου ζωής του συστήματος διαπιστώνουμε ότι είχε δοθεί πολύ λίγη προσοχή στο στάδιο της ανάλυσης προβλημάτων. Αυτό είχε ως αποτέλεσμα να δοθεί περισσότερος χρόνος στην ανάπτυξη των άλλων σταδίων της ανάπτυξης software όπως επίσης και την αύξηση του κόστους διόρθωσης των διαπραγμένων λαθών.

Για να ελαχιστοποιηθεί ο χρόνος που απαιτείται, αλλά και το κόστος διόρθωσης λαθών, αυτοί που ασχολούνται με την ανάπτυξη του software, θα πρέπει να έχουν περιεκτική και κατανοητή εικόνα του απαιτούμενου συστήματος των αναγκών και απαιτήσεων του, πριν ακόμα αρχίσει η ανάπτυξη του software και πριν αρχίσουν να καταβάλλονται μεγάλες προσπάθειες προς αυτό τον τομέα.

Η αλήθεια είναι ότι είναι πολύ δύσκολο να αποκτήσουμε αυτή την εικόνα και να κατανοήσουμε πλήρως το σύστημα. Κι αυτό κυρίως γιατί ο αναλυτής θα πρέπει να αντιμετωπίσει και να συνδιαλλαγεί με τους ανθρώπους που υπάρχουν μέσα στο υπό μελέτη σύστημα. Όπως είναι πολύ γνωστό, η συμπεριφορά αυτών των ανθρώπων είναι κυμαινόμενη, μη λογική και πολλές φορές αναξιόπιστη.

Τα πιο συχνά και πιο σημαντικά προβλήματα που αντιμετωπίζει ένας αναλυτής είναι :

- η αντίληψη των εργαζομένων ότι απειλείται η θέση τους
- η ήδη διαμορφωμένη άποψη για το σύστημα και τον οργανισμό στον οποίο το σύστημα θα τοποθετηθεί
- η ασυνέπεια και η αντιφατικότητα των σκέψεων αυτών

Η διαδικασία ανάλυσης προβλημάτων πρέπει στην αρχή να ασχοληθεί με την κατανόηση του συστήματος, τα προβλήματά του, καθώς και τις ανάγκες του. Έχοντας υπόψη τα προβλήματα αυτής της φάσης, ο αναλυτής για να μπορέσει να αποκτήσει μια γενική εικόνα της δομής και λειτουργίας του οργανισμού, πρέπει να ακολουθήσει μια διαδικασία πέντε σταδίων :

- προσδιορισμός της δομής, οργάνωσης και του προορισμού του οργανισμού
- προσδιορισμός των προβληματικών περιοχών του οργανισμού
- προσδιορισμός του εξωτερικού περιβάλλοντος του οργανισμού και εξακρίβωση των ορίων των προβλημάτων
- καταγραφή των αναλυτικών χαρακτηριστικών του προβλήματος
- εξακρίβωση της εγκυρότητας των πληροφοριών / δεδομένων που συλλέχθηκαν

2.2 Τεχνικές Συλλογής Γεγονότων (Fact Collection Techniques)

Οι τεχνικές που χρησιμοποιούνται για την απόκτηση των απαραίτητων πληροφοριών για την εκτέλεση των παραπάνω εργασιών είναι : η μελέτη των εγγράφων τεκμηρίωσης, οι συνεντεύξεις, η μέθοδος του ερωτηματολογίου, η προσωπική παρατήρηση και επαφή με τα γεγονότα, όπως επίσης και η καταμέτρηση μαζί με την εκτίμηση των δεδομένων.

Ο αναλυτής πρέπει να προσδιορίσει τις προβληματικές περιοχές των δραστηριοτήτων ενός οργανισμού και κατόπιν να φτιάξει και να παρουσιάσει μία απλή και κατανοητή σε όλους κατάσταση των προβλημάτων που υπάρχουν.

2.2.1 Μελέτη εγγράφων τεκμηρίωσης (Documentation Review)

Η μελέτη των υπαρχόντων συστημάτων έχει τα εξής χαρακτηριστικά :

- είναι χρήσιμη για το σχεδιασμό μελλοντικών περιοχών έρευνας και ανάλυσης δίνοντας μία γενική εικόνα, περίληψη όλων των πληροφοριών, καθώς και δυνατότητα επαλήθευσης.
- είναι εύκολα προσπελάσιμη
- είναι φορητή

Υπάρχουν όμως και μειονεκτήματα, όπως :

- έλλειψη ακρίβειας,
- η συχνά περίπλοκη και μεγάλη σε μέγεθος τεκμηρίωση και
- η πρόχειρη ή ατελής τεκμηρίωση.

Συνεπώς, συμπεραίνουμε ότι η τεκμηρίωση μόνο σε μια πολύ γενική εικόνα της μπορεί να αναφέρεται στις δραστηριότητες ενός οργανισμού.

2.2.2 Η μέθοδος της συνέντευξης (Interviewing)

Στις συνεντεύξεις συνήθως, παίρνουν μέρος δύο άτομα : ο αναλυτής και ο άνθρωπος από τον οποίο ο αναλυτής παίρνει τη συνέντευξη. Ο αναλυτής πρέπει να σχεδιάσει από πριν όλη τη διαδικασία της συνέντευξης, να ξεκαθαρίσει γιατί θα γίνει η συνέντευξη, τι ερωτήσεις θα τεθούν και ποιοι παράγοντες θα καθορίσουν στο τέλος, αν η συνέντευξη ήταν επιτυχημένη ή όχι.

Είναι σημαντικό, να σκεφτούμε όλα τα προβλήματα που τυχόν θα δημιουργηθούν κατά τη διάρκεια της συνέντευξης, καθώς και τους εναλλακτικούς τρόπους, με τους οποίους μπορούμε να τα υπερπηδήσουμε ή να τα αποφύγουμε.

Οι συνεντεύξεις αποτελούν το πλέον σημαντικό και χρήσιμο εργαλείο που μπορεί να χρησιμοποιήσει ένας αναλυτής, στην προσπάθεια να κατανοήσει κάθε πλευρά ενός οργανισμού.

Τα κρίσιμα σημεία, που μπορούν να εξασφαλίσουν την επιτυχία αυτού του εργαλείου είναι :

- η πολύ καλή προετοιμασία του ανθρώπου που διενεργεί τη συνέντευξη
- ο τρόπος που διεξάγεται η συνέντευξη και
- η προσήλωση στο θέμα, αλλά ταυτόχρονα και
- η ελευθερία ομιλίας στον συνεντευξιαζόμενο

Ο αναλυτής πρέπει να προσπαθήσει έτσι ώστε η συνέντευξη να έχει τρία στάδια :

- το αρχικό στάδιο (άνοιγμα της συνέντευξης),
- το κύριο στάδιο (συλλογή των δεδομένων / πληροφοριών) και
- το τελικό στάδιο (συμπεράσματα).

Ένα επιπλέον στάδιο, αλλά σίγουρα όχι με λιγότερη σπουδαιότητα, είναι *η τελική σύνοψη της συζήτησης*, όταν πλέον η συνέντευξη έχει τελειώσει. Σ' αυτό το τελευταίο στάδιο, για την αποφυγή παρεξηγήσεων και παρερμηνειών προτείνεται να είναι ο αναλυτής αυτός, που κάνει τη σύνοψη.

Μετά τη συνέντευξη, ο αναλυτής θα πρέπει να συντάξει μία κατάσταση αναφέροντας όλα τα κύρια σημεία που συζητήθηκαν καθώς επίσης και τις εντυπώσεις του για τα διάφορα θέματα που θίχτηκαν και γενικότερα, κάθε τι που θεωρεί σημαντικό από τη συνέντευξη.

Τα κύρια σημεία της κατάστασης είναι εκείνα που καθορίζουν το σκοπό της συνέντευξης, το αν αυτός ο σκοπός επιτεύχθηκε, ποια ήταν τα κύρια σημεία της συνέντευξης κατά τη γνώμη του αναλυτή για την απάντηση που δόθηκε σε κάθε ένα απ' αυτά τα κύρια σημεία.

2.2.3 Η μέθοδος του ερωτηματολογίου (Questioning)

Τα ερωτηματολόγια είναι μία ακόμη τεχνική συλλογής πληροφοριών, που είναι διαθέσιμη στον αναλυτή και χρησιμοποιείται για να συλλέξει πληροφορίες γύρω από :

- τη στάση (attitude),
- τα πιστεύω (belief),
- τη συμπεριφορά-την ασχολία (behavior) και
- τα χαρακτηριστικά (characteristics)

μερικών ανθρώπων-κλειδιών μέσα στον οργανισμό, που επηρεάζονται από το τρέχον και το μελλοντικό σύστημα.

Τα ερωτηματολόγια είναι χρήσιμα και μπορούν να βοηθήσουν στη συλλογή σημαντικών πληροφοριών αν :

- οι άνθρωποι μέσα στον οργανισμό είναι πλατιά διασκορπισμένοι
- ασχολούνται πολλοί άνθρωποι με την ανάπτυξη συστήματος

- είναι απαραίτητη η διερευνητική εργασία πριν από την υποβολή εναλλακτικών προτάσεων και
- υπάρχει ανάγκη βολιδοσκόπησης των προβλημάτων πριν από τη διεξαγωγή των συνεντεύξεων.

Η χρήση ερωτηματολογίου είναι μία ακόμη τεχνική συλλογής πληροφοριών στην οποία οι ερωτήσεις δεν είναι προφορικές, αλλά γραπτές, συγκεκριμένες και έχουν μία προκαθορισμένη μορφή. Αυτή η τεχνική πρέπει να θεωρείται συμπληρωματική των άλλων τεχνικών συλλογής πληροφοριών/ δεδομένων.

Σε πολύ γενικές γραμμές, η τεχνική της χρήσης ερωτηματολογίου έχει τα παρακάτω χαρακτηριστικά :

- είναι δημοφιλής
- δεν είναι εύκολο να εφαρμοστεί σωστά
- απαιτεί πολύ προσεχτική σχεδίαση για να αποφευχθούν παρανοήσεις, αοριστίες, ασάφειες και αμφιβολίες
- συμπληρώνει άλλες μεθόδους
- μπορεί να χρησιμοποιηθεί σα φίλτρο πριν από τις συνεντεύξεις ή τις παρατηρήσεις

Κατά τη σχεδίαση ενός ερωτηματολογίου θα πρέπει να δοθεί μεγάλη προσοχή στα παρακάτω ζητήματα :

- συνοδευτικές οδηγίες
- τύποι ερωτήσεων
- περιεχόμενο ερωτήσεων
- διατύπωση, απλότητα, συνοπτικότητα, περιεκτικότητα και ασάφειες ή διφορούμενες έννοιες, μνήμη και ικανότητα στο να απαντά ο ερωτώμενος

- η σειρά των ερωτήσεων

Πέρα όμως από τα τεχνικής φύσεως θέματα, υπάρχει ένα ακόμη που έχει να κάνει με την *επιλογή των ανθρώπων* που θα απαντήσουν τις ερωτήσεις οι οποίες παρατίθενται στο ερωτηματολόγιο. Η επιλογή αυτή είναι πολύ σημαντική γιατί αν γίνει λάθος σ' αυτό το σημείο, τότε το αποτέλεσμα μπορεί να μην ανταποκρίνεται στις ανάγκες και τις απαιτήσεις του συστήματος. Αυτοί που θα επιλεγούν για να απαντήσουν σ' αυτές τις ερωτήσεις θα πρέπει να ανήκουν σε όλες (αν είναι δυνατόν) τις ομάδες των εργαζομένων που έχουν σχέση με το αντικείμενο της έρευνας.

Τέλος, μία μεγάλη αδυναμία της τεχνικής του ερωτηματολογίου είναι ότι ο αναλυτής δεν είναι ποτέ σίγουρος για την αξιοπιστία των στοιχείων του κάθε ερωτηματολογίου. Λείπει το στοιχείο της προσωπικής επαφής που τον στερεί από ένα μεγάλο πλεονέκτημα : να διαπιστώσει, κατά πόσο η απάντηση είναι αξιόπιστη.

2.2.4 Προσωπική Παρατήρηση (Observation)

Η προσωπική παρατήρηση μπορεί να χρησιμοποιηθεί για να επαληθεύσει ο αναλυτής όσα γνώρισε κατά τη διάρκεια των συνεντεύξεων ή ακόμη νωρίτερα. Είναι επίσης πολύ χρήσιμη τεχνική για την αποκάλυψη των σχέσεων που υπάρχουν μεταξύ των ανθρώπων του συστήματος και είναι ίσως δύσκολο να ανακαλυφθούν με οποιαδήποτε άλλη τεχνική συλλογής πληροφοριών.

Για να μεγιστοποιήσει τις πληροφορίες που θα καταφέρει να αποσπάσει ο αναλυτής χρησιμοποιώντας αυτή τη μέθοδο, θα πρέπει να ακολουθήσει τις παρακάτω οδηγίες :

(α) Προετοιμασίες πριν από την παρατήρηση

Ο αναλυτής πρέπει να προσδιορίσει με ακρίβεια τι πρέπει να παρατηρήσει, να υπολογίσει το χρόνο που θα χρειαστεί, να εξηγήσει στη

διοίκηση του τομέα που πρόκειται να παρατηρήσει τι πρόκειται να γίνει και να αποκτήσει την έγκρισή της για να αρχίσει η παρατήρηση.

(β) Διενεργώντας την παρατήρηση

Ο αναλυτής πρέπει να εξοικειωθεί με το περιβάλλον, μέσα στο οποίο θα κάνει τις παρατηρήσεις του, θα πρέπει να βλέπει μια ενέργεια σε σχέση με το χρόνο που αυτή διεξάγεται και να συγκεντρώνει την προσοχή του στις ενέργειες που τον ενδιαφέρουν χωρίς να σχολιάζει και να κρίνει τους ανθρώπους που συνδιαλέγεται.

(γ) Μετά την παρατήρηση

Οι πληροφορίες που συλλέχθηκαν θα πρέπει να οργανωθούν και να καταγραφούν. Τα συμπεράσματα του αναλυτή πρέπει να συζητηθούν και πάλι με τους ανθρώπους που παρατηρήθηκαν καθώς και με τους ανωτέρους για να εξετασθεί η ακρίβεια και η αντικειμενικότητα τους.

Η ανάλυση με βάση την τεχνική της παρατήρησης μπορεί να αποκαλύψει :

- διακοπές / παρεμβολές στη ροή εργασιών
- άτυπους τρόπους επικοινωνιών
- κακομεταχείριση αρχείων και τεκμηρίωσης
- ανισορροπίες στην κατανομή της εργασίας
- λειτουργικές ανεπάρκειες

Οι αδυναμίες της τεχνικής αυτής είναι ότι απαιτεί πολύ χρόνο και η φερεγγυότητα των γεγονότων που παρατηρήθηκαν είναι χαμηλή λόγω της ψυχολογικής επίδρασης που υπάρχει στους εργαζόμενους.

2.2.5 Καταμέτρηση και Εκτίμηση Δεδομένων (Measuring)

Είναι παραπλήσια της τεχνικής της παρατήρησης. Η διαφοροποίηση μεταξύ των δύο τεχνικών έγινε για να διακρίνει την παρατήρηση στην τεχνική που προσδιορίζει / αναγνωρίζει πρότυπα, υποδείγματα, τάσεις ενώ η τεχνική της καταμέτρησης και εκτίμησης των δεδομένων υποδηλώνει αριθμητικές δραστηριότητες, στις οποίες συγκεκριμένες τιμές ανατίθενται σε ορισμένα χαρακτηριστικά κάποιων δραστηριοτήτων.

2.3 Ανάλυση Συστημάτων

Αφού έχει ολοκληρωθεί το στάδιο συλλογής δεδομένων, το επόμενο βήμα είναι η ανάλυση συστήματος.

Ο Hawryszkiewyck (1988) με τον όρο *ανάλυση συστήματος*, εννοεί την έρευνα και ανάλυση της λειτουργίας του συστήματος και τον προσδιορισμό των πιθανών αλλαγών που μπορούν να επέλθουν σ' αυτό. Γενικά, μπορούμε να πούμε ότι «ανάλυση» είναι μία ακόμα ονομασία της μελέτης.

Στα πλαίσια της ανάπτυξης ενός πληροφοριακού συστήματος διοίκησης η εταιρεία εξετάζεται / μελετάται σα σύστημα. Σ' αυτή τη φάση γίνεται μια πιο λεπτομερής ανάλυση του συστήματος, ώστε οι σχεδιαστές να εξοικειωθούν με τη λειτουργία του υπάρχοντος ή ενός προτεινόμενου καινούριου συστήματος. Η ανάλυση είναι μια λογική διαδικασία / εργασία. Ο αντικειμενικός σκοπός αυτού του σταδίου δεν είναι να δώσει λύσεις στα προβλήματα, αλλά να προσδιορίσει με ακρίβεια αυτό που πρέπει να γίνει για να λυθεί το πρόβλημα. Ο αναλυτής αποτελεί το συνδετικό κρίκο μεταξύ του χρήστη και της εφαρμογής του συστήματος.

Κατά τη διάρκεια της ανάλυσης ο αναλυτής συνεργάζεται πολύ στενά με το χρήστη για να αναπτύξει ένα λογικό μοντέλο του συστήματος. Λογικό, είναι το μοντέλο στο οποίο ο μόνος περιορισμός που λαμβάνεται υπόψη είναι οι απαιτήσεις των χρηστών, χωρίς να γίνεται λόγος για το φυσικό περιβάλλον του συστήματος.

Για μια επιτυχημένη ανάλυση συστήματος, ο αναλυτής πρέπει να έχει δύο προσόντα :

- τεχνική επιδεξιότητα, αλλά και
- κοινωνικότητα, δηλαδή να μπορεί να επικοινωνεί εύκολα και αποτελεσματικά με άλλους ανθρώπους

Μέχρι σήμερα, δινόταν βαρύτητα μόνο στο πρώτο χαρακτηριστικό ενώ το δεύτερο δεν θεωρούνταν σημαντικό.

Διάφορες μελέτες, που έχουν γίνει κατά καιρούς έχουν υποδείξει πέντε κύριους ρόλους για τον αναλυτή :

- παράγοντα αλλαγής
- ερευνητής
- συνδετικός κρίκος, εφόσον κρατά το ρόλο του συνδέσμου μεταξύ της γενικής και αφηρημένης σχεδίασης των απαιτήσεων του τελικού χρήστη και της αναλυτικής φυσικής σχεδίασης
- ακροατής και
- πολιτικός

Το κύριο πρόβλημα που σχετίζεται με τους αναλυτές συστημάτων είναι ότι μερικές φορές παρασύρονται και κινούνται πολύ γρήγορα στη σχεδίαση του προγράμματος, με αποτέλεσμα τον πρόωρο φυσικό σχεδιασμό (σε αντίθεση με το λογικό σχεδιασμό, ο φυσικός σχεδιασμός λαμβάνει υπόψη και το φυσικό περιβάλλον του συστήματος, δηλαδή την τεχνολογία, τους πόρους που μπορούν να διατεθούν κ.α.). Αυτή η τάση πρέπει οπωσδήποτε να αποφεύγεται γιατί σε διαφορετική περίπτωση υπάρχει κίνδυνος να παραχθεί κακοσχεδιασμένο σύστημα. Πώς, όμως, μπορεί ο αναλυτής να αποφύγει μια τέτοια περίπτωση ; Απλώς, μπορεί να βοηθηθεί με τη χρησιμοποίηση διαφόρων δομημένων μεθοδολογιών.

Ο Tom De Marco ορίζει ως *Δομημένη Ανάλυση* τη χρησιμοποίηση Διαγραμμάτων Ροής Δεδομένων, Λεξικών Δεδομένων, Δομημένων Κειμένων, Πινάκων Αποφάσεων και των Δέντρων Αποφάσεων με άμεσο σκοπό την τεκμηρίωση του συστήματος, η οποία θα έχει τη μορφή της δομημένης περιγραφής του συστήματος.

Τα επιθυμητά χαρακτηριστικά της κάθε περιγραφής είναι τα εξής :

- να έχει συνοχή και συνέπεια στην περιγραφή,
- να κάνει λειτουργική ανάλυση και αποσύνθεση,
- να είναι γραφική (παραστατική) και συντηρήσιμη.

Ο αναλυτής πρέπει να συγκεντρώνει την προσοχή του σε τρεις διαφορετικές πλευρές / απόψεις του συστήματος :

- στη διαδικαστική (λειτουργική) πλευρά. Η *Ανάλυση των διαδικασιών (Process analysis)* περιλαμβάνει τη χρησιμοποίηση κάποιων τεχνικών, όπως Διαγράμματα Ροής Δεδομένων, Λεξικά Δεδομένων και άλλες
- στα δεδομένα. Η *Ανάλυση Δεδομένων (Data Analysis)* περιλαμβάνει το Διάγραμμα Οντοτήτων -Συσχετίσεων (Entity – Relationship Diagram) και το Μοντέλο Γεγονότων (Fact – based model).
- το χρόνο. *Η εξέταση του συστήματος με βάση το χρόνο* γίνεται με τη βοήθεια του Διαγράμματος Μετάβασης – Κατάστασης (State – transition Diagram).

Κάθε διαγραμματικό μοντέλο στοχεύει στην ανάπτυξη του συστήματος από μια διαφορετική κάθε φορά πλευρά του συστήματος. Τα Διαγράμματα Ροής Δεδομένων παρουσιάζουν τις λειτουργίες, τα Διαγράμματα Οντοτήτων – Συσχετίσεων τονίζουν τις σχέσεις των δεδομένων και τα Διαγράμματα Μετάβασης – Κατάστασης δίνουν έμφαση στη χρονική εξάρτηση της συμπεριφοράς του συστήματος.

Εξαιτίας της πολυσύνθετης κατάστασης που βρίσκεται το κάθε σύστημα, είναι απαραίτητο να μελετήσουμε κάθε μία διαφορετική πλευρά του συστήματος ξεχωριστά. Από την άλλη πλευρά, αυτές οι τρεις διαφορετικές σκοπιές, από τις οποίες εξετάζουμε το σύστημα, πρέπει να είναι σταθερές, συνεπείς, να συμβιβάζεται η μία με την άλλη και επίσης, να σχεδιάζονται παράλληλα (ταυτόχρονα).

2.3.1 Ανάλυση Διαδικασιών (Process Analysis)

Αφού περάσει η φάση της συλλογής των δεδομένων και αφού οι απαιτήσεις των χρηστών έχουν διυλιστεί από τα γεγονότα και όλες τις άλλες πληροφορίες που έχουν αποκτηθεί ο αναλυτής πρέπει να κινηθεί από τη φάση του ορισμού (καθορισμού) του προβλήματος στη φάση της απλής περιγραφής του συστήματος.

Αυτό μπορεί να επιτευχθεί σε δύο στάδια :

- Αρχικά, δομώντας μία σειρά από μεμονωμένες απόψεις χρηστών για το σύστημα, και
- κατόπιν, μέσω της ενσωμάτωσης όλων αυτών των απόψεων σε μία και μοναδική ολοκληρωμένη άποψη του συστήματος.

2.3.1.1 Διαγράμματα Ροής Δεδομένων (Data Flow Diagrams)

Ο Tom De Marco (1978) δίνει τον εξής ορισμό για το Διάγραμμα Ροής Δεδομένων :

" Ένα Διάγραμμα Ροής Δεδομένων είναι μία σύνθετη διαγραμματική αναπαράσταση του δικτύου ενός συστήματος. Το σύστημα αυτό μπορεί να είναι αυτόματο, χειροκίνητο ή μεικτό. "

Το Διάγραμμα Ροής Δεδομένων, συνεπώς, απεικονίζει το σύστημα υπό μορφή κομματιών, τα οποία είναι μέρη του συστήματος και με καθορισμένους όλους τους συνδετικούς κρίκους που υπάρχουν ανάμεσα στα επιμέρους κομμάτια που αποτελούν το όλο σύστημα.

Σύμφωνα, επίσης με τον Yourdon (1989) το Διάγραμμα Ροής Δεδομένων είναι ένα εργαλείο μοντελοποίησης (modeling tool), το οποίο μας επιτρέπει να περιγράψουμε διαγραμματικά ένα σύστημα σαν ένα δίκτυο λειτουργικών διαδικασιών, η κάθε μία από τις οποίες συνδέεται με μια άλλη «αγωγών» και διατηρεί μία «δεξαμενή» δεδομένων.

Άλλα συνώνυμα του Διαγράμματος Ροής Δεδομένων που συναντάμε στη βιβλιογραφία είναι :

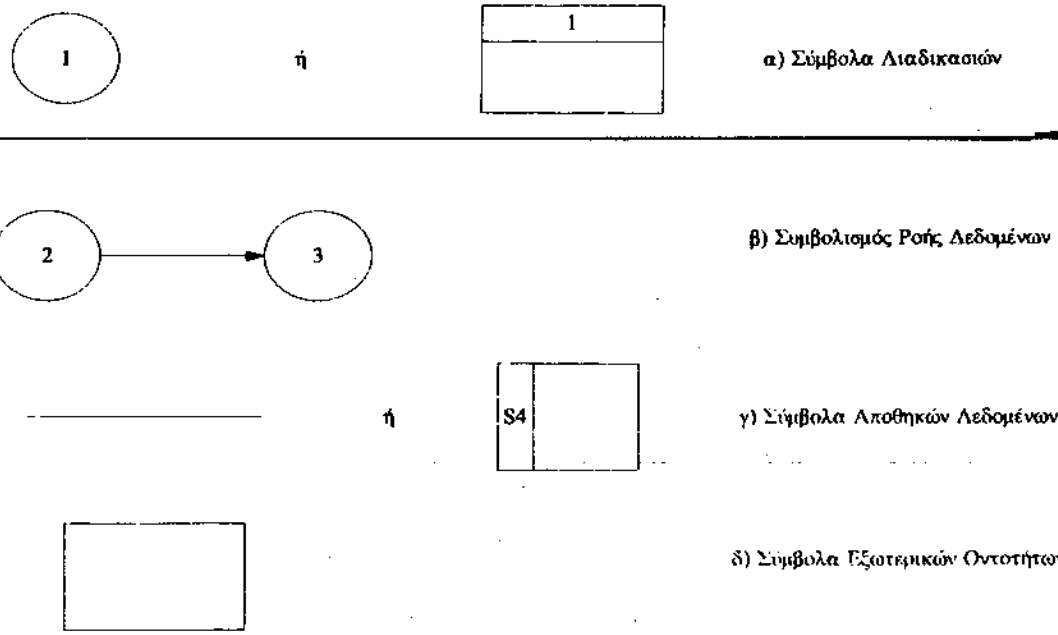
- Πίνακας Φυσαλίδων (Bubble Chart),
- Διάγραμμα Φυσαλίδων (Bubble Diagram),
- Μοντέλο Διαδικασιών (Process model),
- Μοντέλο Ροής Εργασιών (Work Flow model) και
- Μοντέλο Λειτουργιών (Function model).

Οι Martin και Estrin (1967), Whitehouse (1975) και Ross (1977) περιέγραψαν έναν πολύ γενικό τρόπο για τη γραφική απεικόνιση ενός συστήματος. Όμως, η χρήση των Διαγραμμάτων Ροής Δεδομένων, ως εργαλείων μοντελοποίησης, έγινε ευρύτερα γνωστή από τον Tom De Marco (1978) και τους Gane και Sarson (1970) μέσω μεθοδολογιών ανάλυσης δομημένων συστημάτων που παρουσίασαν. Αυτοί απέδειξαν ότι όταν είμαστε στο λογικό στάδιο, το Διάγραμμα Ροής Δεδομένων είναι το εργαλείο – κλειδί για την κατανόηση και το χειρισμό συστημάτων οποιασδήποτε πολυπλοκότητας.

Το Διάγραμμα Ροής Δεδομένων παρέχει μόνο μία άποψη του συστήματος, αυτήν που είναι προσανατολισμένη στις λειτουργίες και είναι μόνο ένα από τα πολλά εργαλεία μοντελοποίησης που είναι διαθέσιμα στον αναλυτή συστημάτων.

Επομένως, αν κάποιος αναπτύσσει ένα σύστημα, για το οποίο οι σχέσεις μεταξύ των δεδομένων ή ακόμη της συμπεριφορά τους σε εξάρτηση με το χρόνο, είναι πολύ πιο σημαντικά θέματα απ' ότι οι λειτουργίες, τότε θα ήταν καλύτερο να αναπτύξει ένα σύνολο Διαγραμμάτων Οντοτήτων – Συσχετίσεων ή ένα σύνολο Διαγραμμάτων Μετάβασης – Καταστάσεων αντίστοιχα.

Όπως μπορούμε να δούμε και στο ακόλουθο σχήμα, τα διαγράμματα ροής δεδομένων αποτελούνται από τέσσερα βασικά στοιχεία / μέρη :

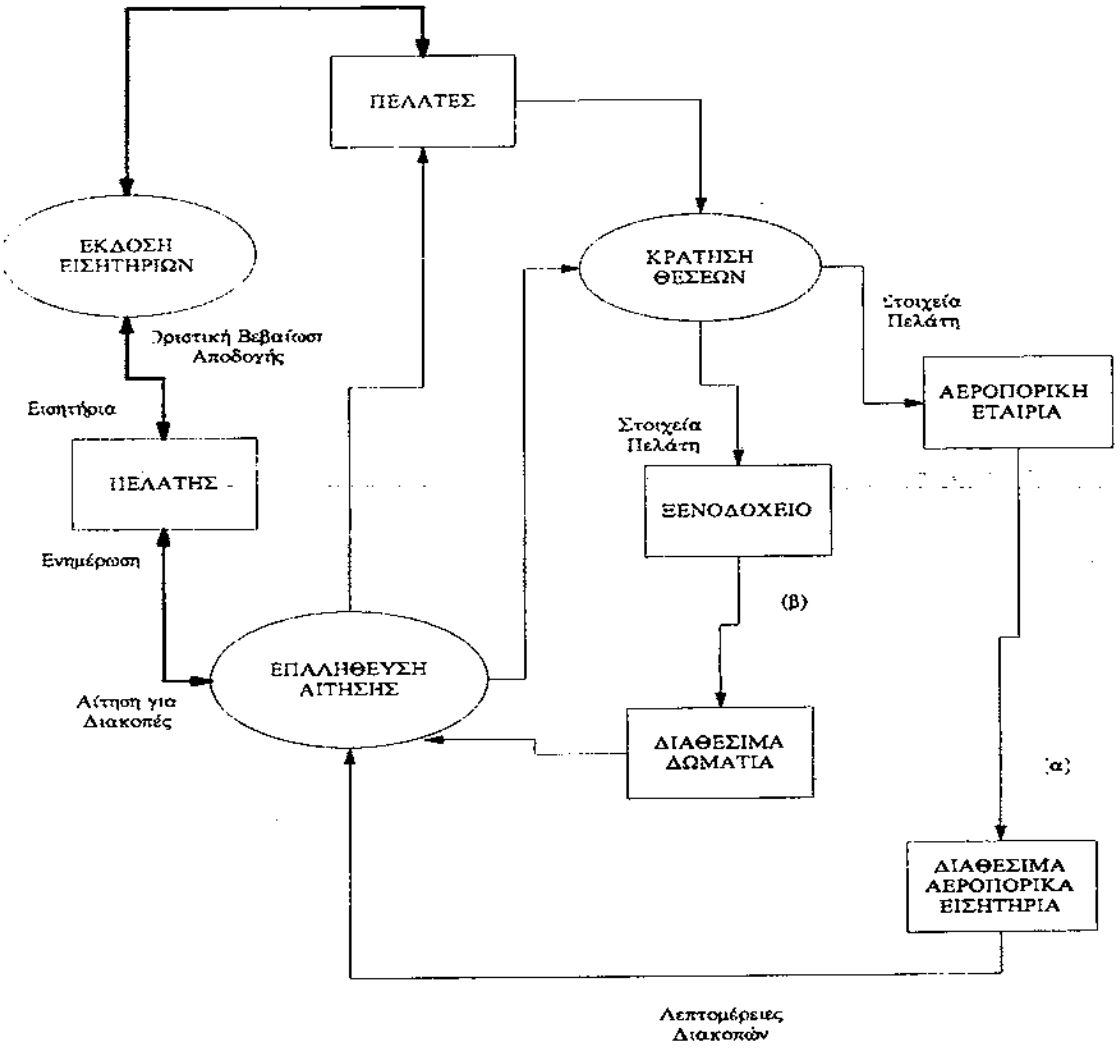


1. *Ροές Δεδομένων (Data Flows)*, οι οποίες απεικονίζονται με γραμμές που έχουν όνομα και κατεύθυνση. Η ροή χρησιμοποιείται για να δείχνει την (μετα)κίνηση των πληροφοριών από ένα μέρος του συστήματος σ' ένα άλλο. Κατ' αυτή την έννοια, οι ροές απεικονίζουν δεδομένα σε κίνηση, ενώ τα αρχικά απεικονίζουν δεδομένα σε στάση. Είναι σημαντικό και πρέπει να τονιστεί ότι οι ροές δεδομένων απεικονίζουν ελέγχους επί των δεδομένων και ότι καμιά από αυτές δεν έχει το ίδιο όνομα με κάποια άλλη.
2. *Διαδικασίες (Processes)*, που απεικονίζονται με κύκλους ή φυσαλίδες. Κοινά συνώνυμά τους είναι φυσαλίδα, λειτουργία ή μετατροπή. Κάθε διαδικασία έχει μία ή περισσότερες εισαγωγές δεδομένων (data inputs) και παράγει μία ή περισσότερες εξαγωγές (data outputs). Κάθε διαδικασία έχει ένα μοναδικό όνομα και αριθμό που είναι αποκλειστικός.
3. *Αρχεία (Files)*, τα οποία απεικονίζονται με ένα ορθογώνιο παραλληλόγραμμο που έχει ανοικτή τη μία πλάγια πλευρά. Ένα αρχείο αποτελεί αποθήκη δεδομένων. Οι διαδικασίες μπορούν να

εισάγουν δεδομένα σ' αυτή την αποθήκη ή να επανακτήσουν δεδομένα από αυτή.

4. *Πηγές Δεδομένων ή Νεροχύτες (Data Sources or Sinks)*, που απεικονίζονται με ορθογώνια παραλληλόγραμμα. Μία πηγή ή νεροχύτης είναι ένα άτομο ή τμήμα ενός οργανισμού έξω από τα όρια του εξεταζόμενου συστήματος, που επικοινωνεί μ' αυτό το σύστημα. Με άλλα λόγια, είναι οντότητες, στις οποίες ο σχεδιαστής δεν μπορεί να ασκήσει κανέναν έλεγχο.

Στο επόμενο σχήμα γίνεται μια γραφική παρουσίαση του τρόπου, με τον οποίο μοντελοποιούνται τα συστήματα, χρησιμοποιώντας τα τέσσερα βασικά στοιχεία κατασκευής των διαγραμμάτων ροής δεδομένων.



Τα γενικά βήματα που πρέπει να ακολουθηθούν προκειμένου να σχεδιαστεί ένα διάγραμμα ροής δεδομένων για ένα ήδη υπαρκτό ή προτεινόμενο σύστημα είναι:

- Προσδιορισμός των εξωτερικών οντοτήτων που σχετίζονται με το υπό εξέταση σύστημα
- Προσδιορισμός των εισερχομένων (πληροφοριών, δεδομένων) και των εξερχόμενων (αποτελεσμάτων) που κάποιος μπορεί να περιμένει και προσεκτική μελέτη και καταγραφή της λειτουργίας της επιχείρησης κάτω από κανονικές συνθήκες
- Προσδιορισμός των μελλοντικών αναγκών για πληροφορίες που μπορεί να προκύψουν

- Προσεκτική επιλογή των ονομάτων των διαδικασιών, των όρων, των αποθηκών και των σημείων τερματισμού. Τα ονόματα αυτά πρέπει να είναι λιτά, περιεκτικά και να προσδιορίζουν επακριβώς το ρόλο του κάθε συγκεκριμένου σημείου
- Αρίθμηση των διαδικασιών
- Σχεδιασμός του διαγράμματος ροής δεδομένων, όσες φορές κι αν χρειαστεί. Ένα ωραίο και καθαρό σχέδιο διαβάζεται ευκολότερα και γρηγορότερα και παρέχει στον παρατηρητή όλες εκείνες τις πληροφορίες που είναι απαραίτητες.
- Προσπάθεια αποφυγής, με οποιονδήποτε τρόπο, σχεδίασης πολύπλοκων διαγραμμάτων ροής δεδομένων
- Επιβεβαίωση ότι το διάγραμμα ροής δεδομένων που σχεδιάστηκε είναι σύμφωνο με τη λογική

Οι βασικές οδηγίες για να επιτευχθεί αυτή η συνέπεια είναι οι εξής :

- αποφυγή άπειρων νεροχυτών (sinks) και φυσαλίδων (bubbles) που έχουν εισερχόμενα στοιχεία, όχι όμως και εξερχόμενα
- αποφυγή δημιουργίας αυθόρμητων φυσαλίδων
- προσοχή στις διαδικασίες και ροές που δεν έχουν όνομα. Πρέπει να δοθεί κάποιο όνομα αμέσως
- προσοχή στις αποθήκες, όπου επιτρέπεται μόνο το διάβασμα (read-only) ή μόνο το γράψιμο (write – only).

Αφού ολοκληρωθεί το διάγραμμα ροής δεδομένων, πρέπει να σιγουρευτούμε ότι το διάγραμμα αυτό αποτελεί πραγματικά ένα μοντέλο του υπό μελέτη συστήματος. Το μυστικό είναι ότι το διάγραμμα πρέπει να ελεγχθεί σε συνεργασία με το χρήστη.

Πρώτα απ' όλα, ο αναλυτής πρέπει να ελέγχει εάν οι αποθήκες δεδομένων περιέχουν κάθε τμήμα δεδομένου που εισέρχεται και εξέρχεται

απ' αυτές. Αν π.χ. διαπιστωθεί ότι μια αποθήκη δεδομένων, που ανήκει στο σύστημα, δίνει δεδομένα χωρίς ποτέ να δέχεται, τότε είναι σίγουρο ότι κάτι λείπει. Σ' αυτή την περίπτωση, ίσως ο αναλυτής να ξέχασε να υποδείξει τη διασύνδεση που υπάρχει μεταξύ αυτής της αποθήκης και άλλου συστήματος ή ακόμη να έχει ξεχάσει μία διαδικασία του συστήματος (ή και να αγνοεί την ύπαρξή της).

Από την άλλη μεριά, μπορεί να έχουμε μια αποθήκη όπου γίνεται εισροή δεδομένων, χωρίς αυτά να βγαίνουν ποτέ από εκεί. Αυτό σημαίνει ότι τα δεδομένα αυτά είναι πλεονάζοντα ή άχρηστα, οπότε δεν υπάρχει λόγος να τα κρατάμε. Μπορεί επίσης να έχει παραληφθεί ή να μην έχει ολοκληρωθεί κάποια λειτουργία που χρησιμοποιεί αυτά τα δεδομένα.

Άλλη μια καλή ενέργεια, που πρέπει να γίνει από τον αναλυτή, είναι να συγκρίνει διάφορες αποθήκες δεδομένων μεταξύ τους, για να διαπιστώσει αν υπάρχει πλεονασμός δεδομένων. Αν όντως συμβαίνει κάτι τέτοιο σημαίνει ότι οι δύο ή τρεις αποθήκες με κοινά δεδομένα μπορούν να συγχωνευθούν. Μέχρι τώρα αναφερόμαστε σε αποθήκες δεδομένων και όχι σε αρχεία. Αυτό γίνεται γιατί η συγκεκριμένη αποθήκη δεδομένων ίσως χρειάζεται για λειτουργικούς και τεχνικούς λόγους να διασπαστεί σε περισσότερα του ενός αρχεία.

Το Διάγραμμα Ροής Δεδομένων χρησιμεύει στην επίτευξη πολλών στόχων. Βοηθά τον αναλυτή να οργανώσει (γραφικά) τις πληροφορίες που διαθέτει για το σύστημα. Η δημιουργία του Διαγράμματος Ροής Δεδομένων αναγκάζει τον αναλυτή να συγκεντρώσει τις πληροφορίες του, να αντλήσει από αυτές τα σημεία – κλειδιά και να σκεφθεί τη σχέση μεταξύ διαφόρων τμημάτων αυτών των πληροφοριών. Λόγω της δομής του διαγράμματος τμήματα που λείπουν και που πιθανόν να έχουν παραβλεφθεί σε μια μεγάλη αφηγηματική περιγραφή του συστήματος, εντοπίζονται πολύ συχνά. Επιπλέον, το περιεχόμενο των Ροών Δεδομένων και των Αποθηκών Δεδομένων, αποτελεί τη βάση για την ανάπτυξη των Λεξικών Δεδομένων. Στα πρώτα στάδια της ανάπτυξης του συστήματος, ένα πρόχειρο διάγραμμα ροής δεδομένων μπορεί να χρησιμεύσει, ώστε να συνοψίσουμε τα αποτελέσματα των συνεντεύξεων, των ερωτηματολογίων ή ακόμη και της

τυπικής περιγραφής του συστήματος. Αργότερα, ένα ολοκληρωμένο διάγραμμα ροής δεδομένων μπορεί να χρησιμοποιηθεί, για να εκφράσει την άποψη του αναλυτή για το σύστημα.

Το Διάγραμμα Ροής Δεδομένων μπορεί να χρησιμοποιηθεί και ως βοήθημα για τη σχεδίαση του συστήματος. Χρησιμοποιώντας σαν οδηγό το χρόνο που χρειάζεται για να ολοκληρωθεί μία ενέργεια, είναι πιθανό να χαράξουμε στο διάγραμμα ένα σημαντικό αριθμό ανύπαρκτων «ορίων». Κάθε όριο θα χωρίζει το σύστημα σε μικρότερα κομμάτια που το καθένα από αυτά θα υποδηλώνει την πιθανή ύπαρξη ενός διαφορετικού φυσικού συστήματος.

Τέλος, το Διάγραμμα Ροής Δεδομένων μπορεί να χρησιμοποιηθεί, για να ελέγξουμε την αρχική φυσική σχεδίαση του συστήματος. Κάθε τμήμα του συστήματος πρέπει να έχει το λογικό αντίστοιχο του στο διάγραμμα ροής δεδομένων. Αν δεν υπάρχει, τότε κατά τη διάρκεια της σχεδίασης έχει συμβεί κάποιο λάθος, το οποίο πρέπει να εντοπισθεί και να διορθωθεί.

2.3.1.2 Λεξικά Δεδομένων (Data Dictionaries)

Ένα δεύτερο και πολύ σημαντικό εργαλείο μοντελοποίησης είναι αυτό των Λεξικών Δεδομένων. Ένα Λεξικό Δεδομένων είναι μία αποθήκη δεδομένων.

Ο Yourdon (1989) ορίζει το Λεξικό Δεδομένων σαν ένα οργανωμένο κατάλογο, με ακριβείς και αυστηρούς ορισμούς, όλων των στοιχείων των δεδομένων που έχουν σχέση με το σύστημα, έτσι ώστε ο χρήστης, μαζί με τον αναλυτή συστημάτων, να κατανοούν από κοινού όλα τα εισερχόμενα (inputs), εξερχόμενα (outputs), τα περιεχόμενα των αποθηκών και τους ενδιάμεσους υπολογισμούς.

Έτσι, η βασική ιδέα είναι η χρησιμοποίηση πληροφοριών για τον ορισμό, τη δομή και τη χρήση κάθε στοιχείου δεδομένων που χρησιμοποιεί ο οργανισμός (το σύστημα).

Ένα Λεξικό Δεδομένων ορίζει τα στοιχεία των δεδομένων με τους παρακάτω τρόπους :

- Περιγράφει την έννοια και τη σημασία των ροών και των αποθηκών, οι οποίες απεικονίζονται στα διαγράμματα ροής δεδομένων.
- Περιγράφει τη σύνθεση όλων των ομαδικών πακέτων δεδομένων που μεταφέρονται από τις ροές δεδομένων.
- Περιγράφει τη σύνθεση των πακέτων δεδομένων που βρίσκονται στις αποθήκες.
- Αναφέρει λεπτομερώς όλες τις σχετικές τιμές και μονάδες των στοιχειωδών πληροφοριών στις ροές δεδομένων και στις αποθήκες δεδομένων και
- Περιγράφει τις λεπτομέρειες των σχέσεων μεταξύ των αποθηκών, οι οποίες τονίζονται από το Διάγραμμα Οντοτήτων-Συσχετίσεων.

Πριν ασχοληθούμε με την τεχνική του προσδιορισμού και της περιγραφής των διαδικασιών, αξίζει να προσδιορίσουμε τα χαρακτηριστικά που πρέπει να έχει κάθε μορφή απεικόνισης.

Τα χαρακτηριστικά αυτά είναι τα εξής :

- Οι εγγραφές στα λεξικά δεδομένων πρέπει να είναι ακριβείς, συνοπτικές και περιεκτικές ροές δεδομένων, τμήματα δεδομένων και αρχεία. Οι περιγραφές των διαδικασιών πρέπει επίσης να επισημαίνουν τα ακριβή χαρακτηριστικά κάθε τύπου του τμήματος, χωρίς να αφήνουν καθόλου χώρο για ασάφειες. Γι' αυτό το λόγο άλλωστε, αποκλείονται οι φυσικές γλώσσες, ως ανταγωνιστής των γλωσσών που πρέπει να χρησιμοποιηθούν από τα λεξικά δεδομένων.
- Οι εγγραφές στα λεξικά δεδομένων δεν πρέπει να περιέχουν πλεονάζοντα στοιχεία που πολλές φορές είναι και περιττά, γιατί αυτό οδηγεί σε μια αύξηση των προσπαθειών, σε περιπτώσεις ενημέρωσης των λεξικών.

Σήμερα, το λεξικό δεδομένων σχεδόν πάντα υλοποιείται σαν ένα μέρος από ένα εργαλείο CASE «σχεδιασμού δομημένης ανάλυσης».

Αν και το σχήμα των λεξικών διαφέρει από εργαλείο σε εργαλείο, τα περισσότερα περιέχουν την ακόλουθη πληροφορία :

- Όνομα - το κυρίως όνομα του είδους δεδομένου ή ελέγχου της αποθήκης δεδομένων ή από μία εξωτερική οντότητα
- Ψευδώνυμο – κάποια άλλα ονόματα για την πρώτη εγγραφή
- Πού χρησιμοποιείται / πως χρησιμοποιείται
- Περιγραφή περιεχομένου
- Συμπληρωματική πληροφορία

2.3.1.3 Τεχνικές περιγραφής διαδικασιών

Μια λειτουργικά πρωτογενής διαδικασία είναι μια διαδικασία που δεν μπορεί να αναλυθεί περισσότερο και η οποία είναι προσδιορισμένη μέσα στο λεξικό δεδομένων. Πρόκειται για μια περιγραφή αυτού που συμβαίνει μέσα σε κάθε ακρότατο επίπεδο, την πρωτογενή φυσαλίδα, ενός διαγράμματος ροής δεδομένων. Οι De Marco (1978), Gane και Sarson (1977) και Weinberg (1978) χρησιμοποιούν σαν εναλλακτικό όρο τη λέξη minispec, που είναι σύντμηση του miniature specification.

Σκοπός αυτής της τεχνικής είναι να καθορίσει αυτό που πρέπει να γίνει, έτσι ώστε να μετατρέψουμε τα εισαγόμενα δεδομένα σε εξαγόμενα αποτελέσματα (transform inputs into outputs).

Αυτά τα χαρακτηριστικά (προδιαγραφές) μπορούν να απεικονισθούν με τη βοήθεια μιας σειράς τεχνικών, όπως π.χ. τα Δέντρα Αποφάσεων (Decision Trees), τους Πίνακες Αποφάσεων (Decision Tables), τα Δομημένα Κείμενα, τα Διαγράμματα Nassi-Shneiderman, τις Παραθέσεις Συνθηκών και πολλές άλλες.

2.3.1.3.1 Δέντρα Αποφάσεων

Ένα Δέντρο Αποφάσεων είναι ένα διάγραμμα, το οποίο απεικονίζει συνθήκες και ενέργειες, καθώς επίσης και τη σχέση που υπάρχει ανάμεσα στην κάθε συνθήκη και τις ενέργειες που απαιτούνται να γίνουν κάτω απ' αυτές τις συνθήκες.

Η ανάπτυξη ενός δέντρου αποφάσεων ωφελεί τον αναλυτή για δύο λόγους. Πρώτα απ' όλα η ανάγκη περιγραφής των συνθηκών και των ενεργειών αναγκάζει τον αναλυτή να καθορίσει την ενέργεια που απαιτείται να γίνει. Έτσι, είναι σχεδόν αδύνατο να παραβλεφθεί ένα σύνολο βημάτων της διαδικασίας λήψης αποφάσεων, ανεξάρτητα με το αν αυτή εξαρτάται από ποσοτικές ή ποιοτικές μεταβλητές. Τα δέντρα αποφάσεων επίσης, αναγκάζουν τους αναλυτές να μελετήσουν τη σειρά των αποφάσεων. Πολύ εύκολα μπορεί κάποιος να εξακριβώσει ότι μια συνθήκη δεν μπορεί να υπάρξει, παρά μόνο αν υπάρχει ήδη κάποια άλλη συνθήκη και έχει διευθετηθεί με μια απόφαση. Έτσι επιτυγχάνουμε τον έλεγχο όλων των δυνατών συνθηκών και αποφάσεων (ενεργειών) που μπορεί να υπάρχουν σε μια διαδικασία αποφάσεων αλλά ακόμη και του χρόνου και της σειράς που θα λάβει χώρα κάθε συνθήκη και θα ληφθεί απόφαση.

Εκείνο που θα πρέπει να προσεχθεί ιδιαίτερα είναι ο τρόπος παρουσίασης των συνθηκών και των ενεργειών. Κατά τη σχεδίαση των δέντρων αποφάσεων, είναι πάρα πολύ χρήσιμο να γίνεται διαχωρισμός των συνθηκών και των ενεργειών, ιδιαίτερα όταν συμβάλλουν μέσα σε ένα ορισμένο χρονικό διάστημα. Επίσης, η σειρά με την οποία συμβάλλουν είναι σημαντικός παράγοντας. Γι' αυτό το λόγο, η χρήση ενός τετράγωνου συμβόλου που θα υποδηλώνει μια ενέργεια και ενός κύκλου που θα απεικονίζει μια συνθήκη, είναι ένας ενδεδειγμένος τρόπος διαχωρισμού των συνθηκών από τις ενέργειες. Η χρησιμοποίηση, λοιπόν αυτών των συμβόλων κάνει ένα δέντρο αποφάσεων πιο ευκολοδιάβαστο, αν μεταφράσουμε κάθε κύκλο σαν IF (εάν) και κάθε τετράγωνο σαν THEN (τότε).

Κατά τη σχεδίαση ενός δέντρου θα πρέπει να καθοριστούν όλες οι συνθήκες και ενέργειες, καθώς επίσης η σειρά και ο χρόνος τους.

Μια ακόμη ιδιότητα των δέντρων αποφάσεων είναι ότι αναγνωρίζουν όλα εκείνα τα δεδομένα που απαιτούνται από τους managers και χρησιμοποιούνται για να μπορέσουν αυτοί να καταλήξουν σε ορισμένα συμπεράσματα και να πάρουν μια απόφαση. Έτσι, ο αναλυτής θα πρέπει να προσδιορίσει και να καταγράψει όλα τα δεδομένα που χρησιμοποιούνται από αυτή τη διαδικασία, ακόμη κι αν δεν περιγράφονται στο δέντρο αποφάσεων. Εάν το δέντρο αποφάσεων κατασκευάζεται μετά την ολοκλήρωση της ανάλυσης ροής δεδομένων, τα δεδομένα ίσως ήδη να περιγράφονται από το Λεξικό Δεδομένων (Data Dictionary), το οποίο περιγράφει τα δεδομένα του συστήματος και τον τρόπο που αυτά χρησιμοποιούνται. Αν όμως, παραδόξως, το δέντρο αποφάσεων χρησιμοποιείται μόνο του, ο αναλυτής θα πρέπει να αναγνωρίσει και να περιγράψει με σαφήνεια και ακρίβεια, όλα εκείνα τα τμήματα δεδομένων που χρησιμοποιούνται από τη διαδικασία λήψης αποφάσεων.

Παρατηρούμε ότι τα δέντρα αποφάσεων είναι πολύ χρήσιμο εργαλείο στα χέρια του αναλυτή. Αυτό δεν σημαίνει ότι μπορεί να ασχοληθεί παντού και πάντα και να δώσει σημαντικά αποτελέσματα. Πολλές φορές, λόγω της ύπαρξης περίπλοκων συστημάτων με πολλά διαδοχικά βήματα και συνδυασμούς συνθηκών, η χρησιμοποίηση των δέντρων αποφάσεων αντί να βοηθήσει τον αναλυτή, τον αποπροσανατολίζει έτσι ώστε στο τέλος αυτός να μην είναι σε θέση να προσδιορίσει τους κανόνες και τις πολιτικές εκείνες, που οδηγούν στη λήψη μιας συγκεκριμένης απόφασης.

Όταν λοιπόν αντιμετωπίζουμε τέτοιου είδους προβλήματα είναι χρησιμότερο να καταφεύγουμε σε μια άλλη μέθοδο, τους Πίνακες Αποφάσεων (Decision Tables).

2.3.1.3.2 Πίνακες Αποφάσεων

Οι Layzell και Loucopoulos (1987) ορίζουν έναν Πίνακα Αποφάσεων ως μία συνοπτική απεικόνιση των συνθηκών και ενεργειών και επίσης ως μία

ένδειξη του ποιες ενέργειες θα πρέπει να γίνουν κάτω από ορισμένες συνθήκες.

Ένας Πίνακας Αποφάσεων αποτελείται από τέσσερα τμήματα στη μορφή που φαίνονται στον επόμενο πίνακα :

ΣΥΝΘΗΚΕΣ	ΚΑΝΟΝΕΣ							
	1	2	3	4	5	6	7	8
Πάνω από 3 είδη	Y	Y	N	N	Y	N	Y	N
Αξία>=50.000 δρχ.	Y	Y	Y	Y	N	N	N	N
Πληρωμή τοις μετρητοίς	Y	N	Y	N	Y	Y	N	N
Εκπτωση 15%	X							
Εκπτωση 10%		X	X					
Εκπτωση 7%				X	X			
Εκπτωση 3%						X	X	
Εκπτωση 0%								X

Πίνακας Αποφάσεων – Καθορισμός Ποσοστού Έκπτωσης

(α) *Το τμήμα των συνθηκών* περιλαμβάνει διάφορες συνθήκες οι οποίες επηρεάζουν την τελική απόφαση και τοποθετείται στο πάνω αριστερό τμήμα του πίνακα,

(β) *Το τμήμα ενεργειών* περιέχει διάφορες ενέργειες, οι οποίες πρόκειται να γίνουν και τοποθετείται στο κάτω αριστερό τμήμα του πίνακα,

(γ) *Το τμήμα των κανόνων* περιλαμβάνει επιλογείς, οι οποίοι προσδιορίζουν τους διαφορετικούς συνδυασμούς των πιθανών συνθηκών και τοποθετείται στο πάνω δεξιό τμήμα του πίνακα και

(δ) *Το τμήμα εισαγωγής ενεργειών* περιλαμβάνει επιλογείς οι οποίοι επιλέγουν τις ενέργειες που πρέπει να εκτελεσθούν και τοποθετείται στο κάτω δεξιό τμήμα του πίνακα

Με δεδομένη την παραπάνω δομή των πινάκων αποφάσεων τρεις παραλλαγές είναι πιθανές :

- Ένας πίνακας αποφάσεων περιορισμένων εισαγωγών περιέχει μόνο τους δυαδικούς επιλογείς Υ και Ν στο τμήμα των κανόνων και το σύμβολο επιλογής Χ στο τμήμα επιλογής ενεργειών.
- Ένας πίνακας αποφάσεων μικτών εισαγωγών εμφανίζει στο τμήμα εισαγωγής ενεργειών σύμβολα διάφορα του Χ, και
- Ένας πίνακας αποφάσεων εκτενών εισαγωγών. Στο τμήμα των κανόνων δεν υπάρχουν απλώς μόνο οι δυαδικοί επιλογείς Υ και Ν, αλλά και ορισμένες τιμές ή εύρος τιμών.

Για να κατασκευάσει ο αναλυτής έναν πίνακα αποφάσεων πρέπει :

- να προσδιορίσει το μέγιστο πιθανό μέγεθος του,
- να απαλείψει κάθε περίπτωση αδύνατης κατάστασης, ασυνέπειας ή πλεονασμού και τέλος,
- να απλοποιήσει τον πίνακα όσο περισσότερο είναι δυνατό

Η προφανής αξία των πινάκων αποφάσεων έγκειται στο ότι αυτοί βοηθούν τον αναλυτή (και τον σχεδιαστή) να ξεδιαλύνει πολύπλοκες και όχι τόσο εμφανείς στρατηγικές, για να διαπιστώσει εάν και που υπάρχει έλλειψη συνοχής σ' αυτές.

Μια όχι και τόσο φανερή επίπτωση των πινάκων αποφάσεων είναι το μήνυμα που αυτοί μπορεί να περιέχουν για τον manager. Αυτοί ανακαλύπτουν συχνά ότι η πολιτική που ακολουθείται, δεν είναι ξεκαθαρισμένη και ότι σ' ένα μεγάλο μέρος της, ακολουθείται χωρίς να υπάρχει η απαραίτητη γνώση όχι μόνο όλων των μεταβλητών που την επηρεάζουν, αλλά και του τρόπου που την επηρεάζουν.

Ένα ακόμη πλεονέκτημα των πινάκων αποφάσεων είναι ότι προσγειώνουν τον αναλυτή και τον manager στην πραγματικότητα, κάνοντάς τους να δουν το πράγμα από τη σωστή τους πλευρά.

Ένα άλλο πλεονέκτημα είναι ότι όταν ο αναλυτής παραδώσει στο σχεδιαστή ή στον προγραμματιστή τον Πίνακα Αποφάσεων (μαζί με τα Διαγράμματα Ροής Δεδομένων και όλα τα άλλα απαραίτητα στοιχεία) υπάρχει μία πληθώρα επιλογών όσον αφορά τη στρατηγική εφαρμογής. Οι πίνακες αποφάσεων μπορούν να μεταφρασθούν σε προγράμματα, χρησιμοποιώντας IF δηλώσεις, δομή CASE ή ακόμη δομή GO TO DEPENDING ON (στην Cobol).

Συνοπτικά, οι πίνακες αποφάσεων έχουν τα εξής πλεονεκτήματα :

- μπορούν να κατανοηθούν πολύ εύκολα
- οι εναλλακτικές προτάσεις βρίσκονται η μία δίπλα στην άλλη και μπορούν να συγκριθούν εύκολα
- η σχέση συνθήκης – αποτελέσματος είναι προφανής και έτσι επιτυγχάνεται μεγαλύτερη εγκυρότητα
- μπορεί να γίνει εύκολα ο έλεγχος για το αν έχουν ληφθεί υπόψη όλοι οι πιθανοί συνδυασμοί

2.3.1.3.3 Δομημένα Κείμενα

Τα Δομημένα Κείμενα ή αλλιώς Γλώσσα Σχεδιασμού Προγράμματος (program design language – POL) είναι ένα υποσύνολο της αγγλικής γλώσσας με ορισμένους σοβαρούς περιορισμούς στα είδη των προτάσεων που μπορούν να χρησιμοποιηθούν.

Τα Δομημένα Κείμενα αναγνωρίζουν τα παρακάτω τρία κατασκευάσματα ελέγχου :

- ***Σειρές (Sequences)*** : απεικονίζουν μία ή περισσότερες ενέργειες οι οποίες λαμβάνουν μέρος διαδοχικά χωρίς διακοπή
- ***Επιλογές (Selections)*** : μόνο μία πολιτική επιλέγεται από μία σειρά εναλλακτικών πολιτικών

- ***Επαναλήψεις (Iterations)*** : επαναλήψεις έχουμε εκεί, όπου μία πολιτική ή μία σειρά ενεργειών επαναλαμβάνεται με μια συχνότητα

2.3.1.3.4 Σύγκριση των Τεχνικών Περιγραφής Διαδικασιών

Τα Δέντρα Αποφάσεων έχουν τρία κύρια πλεονεκτήματα σε σύγκριση με τους Πίνακες Αποφάσεων :

- Πρώτον, εκμεταλλευόμενοι τη σειριακή δομή των Δέντρων Αποφάσεων, η σειρά του ελέγχου των συνθηκών και της εκτέλεσης των αποφάσεων είναι άμεσα ορατή.
- Δεύτερον, ορισμένες συνθήκες και ενέργειες βρίσκονται μόνο σε ορισμένα τμήματα / κλαδιά του Δέντρου Αποφάσεων, ενώ σε άλλα παραλείπονται. Αντίθετα, στους Πίνακες Αποφάσεων υπάρχουν όλα μαζί στον ίδιο πίνακα. Έτσι, μπορούμε να καταλάβουμε ότι τα Δέντρα Αποφάσεων δεν πρέπει αναγκαστικά να είναι συμμετρικά.
- Τρίτον, σε σύγκριση με τους Πίνακες Αποφάσεων, τα Δέντρα Αποφάσεων είναι περισσότερο κατανοητά από άλλους που βρίσκονται μέσα στο υπό μελέτη σύστημα. Έτσι, μπορεί να θεωρηθεί ως καταλληλότερο εργαλείο επικοινωνίας και κατανόησης των καταστάσεων μεταξύ των μελών του συστήματος.

Κάθε μία τεχνική έχει τα δικά της πλεονεκτήματα, αλλά και μειονεκτήματα. Όμως μία απ' αυτές μπορεί να χρησιμοποιηθεί σε κάθε περίπτωση.

Συγκρίνοντας τα Δομημένα Κείμενα με τα Δέντρα Αποφάσεων και τους Πίνακες Αποφάσεων, μπορούμε να πούμε ότι τα Δομημένα Κείμενα μπορούν να χρησιμοποιηθούν καλύτερα, οπουδήποτε το πρόβλημα περιλαμβάνει σύνθετες σειρές ενεργειών με αποφάσεις ή βρόγχους.

Στον ακόλουθο πίνακα γίνεται μια αναλυτικότερη σύγκριση μεταξύ των τριών αυτών εργαλείων.

ΧΡΗΣΗ	ΔΕΝΤΡΑ ΑΠΟΦΑΣΕΩΝ	ΠΙΝΑΚΕΣ ΑΠΟΦΑΣΕΩΝ	ΔΟΜΗΜΕΝΑ ΚΕΙΜΕΝΑ
Λογική Επαλήθευση	Μέτρια	Πολύ Καλή	Καλή
Παρουσίαση Λογικής Δομής	Πολύ Καλή (μόνο αποφάσεων)	Μέτρια (μόνο αποφάσεων)	Καλή
Απλότητα και Ευκολία στην Χρήση	Πολύ Καλή	Πολύ φτωχή προς φτωχή	Μέτρια
Επιβεβαίωση Χρήστη	Καλή	Φτωχή (εκτός εάν ο χρήστης είναι έμπειρος)	Φτωχή προς Μέτρια (ελαφρά ακατάληπτη)
Περιγραφή Προγράμματος	Μέτρια	Πολύ Καλή	Πολύ Καλή
Κατανόηση Μηχανισμού	Φτωχή	Πολύ Καλή	Πολύ Καλή
Επιμέλεια Μηχανισμού	Φτωχή	Πολύ Καλή	Πολύ Καλή Μέτρια (χρειάζεται καλύτερη σύνταξη)
Ευμεταβλητότητα	Μέτρια	Φτωχή	----

Σύγκριση Εργαλείων Περιγραφής Διαδικασιών

Πηγή : Gane και Sarson, Structured Systems Analysis :

Εργαλεία και Τεχνικές, 1986, σελ.162.

2.3.2 Ανάλυση Δεδομένων (Data Analysis)

Η ανάλυση δεδομένων είναι, σύμφωνα με τον Travis (1987) μία βασική τεχνική τεκμηρίωσης των πόρων των δεδομένων.

Οι Layzell και Loucopoulos (1987) ορίζουν την Ανάλυση Δεδομένων ως τεχνική που λαμβάνει ένα άμορφο πλήθος γεγονότων γύρω από τα δεδομένα που χρησιμοποιούνται από το σύστημα και μετατρέπεται σ' ένα σύνολο από ακριβείς, σαφείς, περιεκτικές και όχι πλεονασματικές περιγραφές δεδομένων. Αυτή τη μέθοδο την έχουν αναπτύξει και έχουν δώσει κάποιες οδηγίες χρησιμοποιήσεως της, οι Cobol (1970,1972), Chen (1976,1977) και επίσης η επιτροπή (ANSI/SPARC) για το σχεδιασμό των Βάσεων Δεδομένων (ANSI,1975).

Η μοντελοποίηση δεδομένων αποτελείται από περισσότερα του ενός βήματα-επίπεδα, εκ των οποίων το καθένα χρησιμοποιεί δικές του ιδέες-εργαλεία.

Αυτά τα βήματα-επίπεδα είναι τα εξής :

- καθορισμός των περιοχών δεδομένων που θα αναλυθούν
- κατασκευή του Μοντέλου Οντοτήτων-Συσχετίσεων (E-R modeling)
- λεπτομερής ανάλυση οντοτήτων (detailed entity analysis)
- ανάλυση λειτουργικής εξάρτησης (functional dependency analysis)
- ομαλοποίηση δεδομένων (data normalization)
- ανάλυση της ιστορίας των οντοτήτων του συστήματος (entity life history analysis)
- ενοποίηση των περιοχών δεδομένων
- συνέπεια με τη μοντελοποίηση των διαδικασιών

2.3.2.1 Τα τμήματα ενός Διαγράμματος Οντοτήτων-Συσχετίσεων

Η ανάλυση Οντοτήτων-Συσχετίσεων χρησιμοποιεί τρεις κύριες υποθέσεις – «αφαιρέσεις» για να περιγράψει τα δεδομένα.

Αυτές είναι οι εξής:

- ***Οντότητες ή Γεγονότα (Entities)*** οι οποίες είναι ξεχωριστά πράγματα στην επιχείρηση
- ***Σχέσεις (Relationships)*** οι οποίες είναι νοηματικές αλληλεπιδράσεις μεταξύ των αντικειμένων και
- ***Χαρακτηριστικά (Attributes)*** που είναι οι ιδιότητες των οντοτήτων και σχέσεων

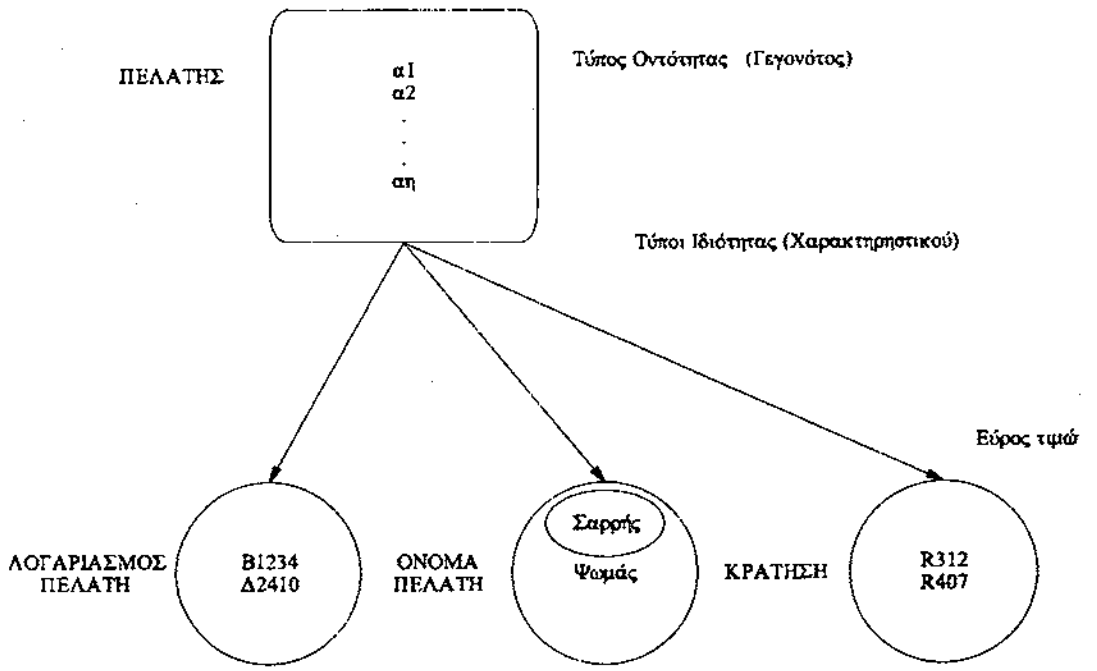
Μια οντότητα μπορεί να οριστεί ως ένα αντικείμενο, που η ταυτότητα του προσδιορίζεται εύκολα και καθαρά και για το οποίο αποθηκεύονται διάφορες πληροφορίες. Αρκετά περιστατικά οντοτήτων ομαδοποιούνται σε τύπους οντοτήτων (ή τύπους γεγονότων). Μία συλλογή από οντότητες καλείται ομάδα (σειρά) οντοτήτων.

Κάθε οντότητα που ανήκει σε μια συγκεκριμένη ομάδα οντοτήτων έχει τα παρακάτω χαρακτηριστικά :

- η ταυτότητα της κάθε μιας μπορεί κατά κάποιο τρόπο να προσδιοριστεί με μοναδικότητα
- κάθε μία παίζει έναν απαραίτητο ρόλο μέσα στο σύστημα που κατασκευάζεται
- κάθε μία μπορεί να περιγραφεί από ένα ή περισσότερα τμήματα δεδομένων

Τα χαρακτηριστικά μιας οντότητας εκφράζονται με το ζευγάρι «χαρακτηριστικό – τιμή». Κάθε χαρακτηριστικό μπορεί να έχει ένα όνομα (attribute name) και επίσης σε αντιστοιχία μ' αυτό το όνομα, θα υπάρχουν μία

ή περισσότερες τιμές (attribute values). Όλες αυτές οι τιμές για ένα δεδομένο χαρακτηριστικό ονομάζονται χώρος αρμοδιοτήτων ή εύρος τιμών ή πεδίο ορισμού (domain). Η σχέση μεταξύ των τύπων οντοτήτων, των τύπων ιδιοτήτων και του χώρου αρμοδιοτήτων παρουσιάζεται στο ακόλουθο σχήμα :



Μερικές φορές είναι απαραίτητο να δίνεται σε ορισμένες μεμονωμένες οντότητες ένα όνομα-διεύθυνση. Αυτό μπορεί να επιτευχθεί με τη χρήση ενός δείκτη, του αναγνωριστή οντότητας (entity identifier). Ένας αναγνωριστής οντότητας είναι ένα σύνολο από ιδιότητες, οι οποίες προσδιορίζουν με μοναδικό τρόπο μία οντότητα. Έτσι, εάν ένας τύπος οντότητας δεν έχει από μόνος του ένα φυσικό αναγνωριστή, τότε θα πρέπει να δοθεί ένας τεχνητός (δηλαδή από εμάς), ο οποίος θα επιλέγει ανάμεσα σε ένα ή περισσότερα από τα ήδη υπάρχοντα χαρακτηριστικά / ιδιότητες ή και ακόμη, αν χρειασθεί θα εισάγει ένα καινούριο.

Οι οντότητες συνδέονται μεταξύ τους με διάφορες σχέσεις. Αυτή η σχέση που υπάρχει ανάμεσα σε δύο ή περισσότερους τύπους οντοτήτων είναι γνωστή με το όνομα συγγένειας ή συσχέτισης (relationship type).

Είναι πολύ σημαντικό να αναγνωρίσουμε ότι μια σχέση απεικονίζει κάτι, το οποίο πρέπει να θυμάται το σύστημα, κάτι που δεν μπορεί να υπολογισθεί ή να προέλθει μηχανικά. Επίσης, ο αναγνωριστής συγγένειας ή συσχέτισης (relationship identifier) πρέπει να χρησιμοποιείται για να προσδιορίσουμε μία συγκεκριμένη σχέση που υπάρχει ανάμεσα σε δύο οντότητες. Αυτός ο αναγνωριστής συγγένειας είναι η μίξη-σύνθεση των αναγνωριστών οντοτήτων μέσα στην ίδια συγγένεια-σχέση.

Ο τρόπος συμμετοχής των οντοτήτων στις συγγένειες των Διαγραμμάτων Οντοτήτων-Συσχετίσεων είναι ακόμη ένα πολύ σημαντικό θέμα.

Γενικά, μπορούμε να πούμε, ότι υπάρχουν τρεις τρόποι, με τους οποίους δύο τύποι οντοτήτων (Α και Β) μπορούν να συσχετισθούν.

- *Απλή σύνδεση ή Τύπος 1* ανάμεσα στην Α και Β υπάρχει, όταν κάθε αξία της Α προσδιορίζει μοναδικά ένα Β.
- *Περίπλοκη σύνδεση ή Τύπος Μ* ανάμεσα στην Α και Β υπάρχει, όταν κάθε Α μπορεί να συσχετισθεί με οποιονδήποτε αριθμό των Β.
- *Υπό συνθήκη σύνδεση ή Τύπος C* ανάμεσα στην Α και Β υπάρχει, όταν κάθε Α συσχετίζεται με ένα ή κανένα Β.

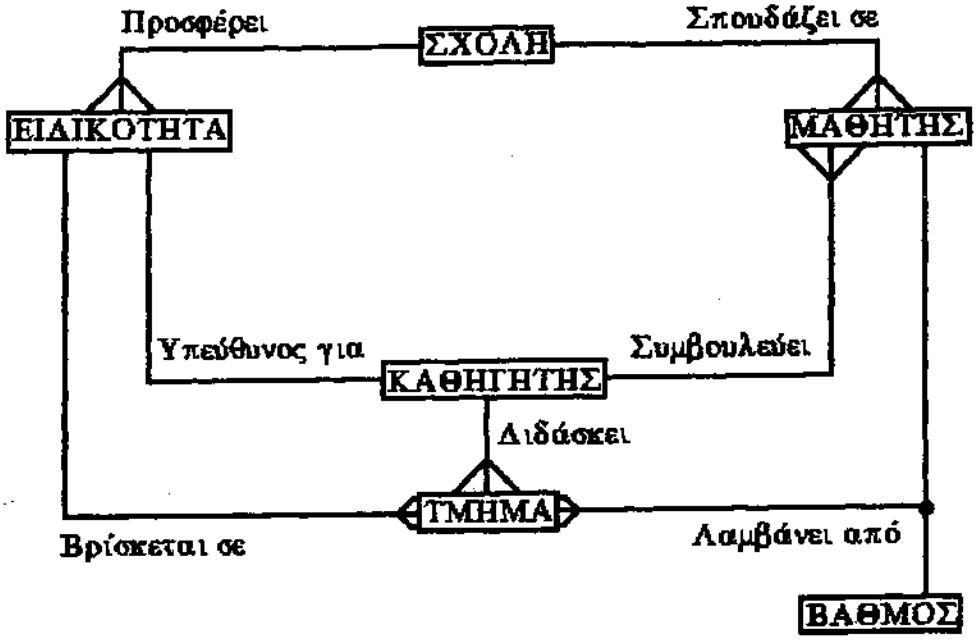
Ο τρόπος συσχέτισης των οντοτήτων μας φέρνει αντιμέτωπους με ένα άλλο σημαντικό θέμα, αυτό της «σχέσης προτεραιότητας» (relationship cardinality). Ο απόλυτος αριθμός της σχέσης / συγγένειας είναι οι εμφανίσεις των οντοτήτων σε μία αντιστοίχηση.

Μία αντιστοίχηση Α και Β είναι η συσχέτιση από το Α στο Β και το αντίθετο και μπορεί να ερμηνευθεί ως εξής :

- Αντιστοίχηση ένα προς ένα (1:1) : κάθε ένα μέλος του Α μπορεί να σχετίζεται με ένα μέλος του Β και αντίστροφα κάθε ένα μέλος του Β μπορεί να σχετίζεται με ένα μέλος του Α.
- Αντιστοίχηση ένα προς πολλά (1:m) : κάθε ένα μέλος του Α μπορεί να σχετίζεται με πολλά μέλη του Β αλλά κάθε ένα μέλος του Β μπορεί να σχετίζεται με ένα μόνο μέλος του Α.
- Αντιστοίχηση πολλά προς ένα (m:1) : είναι η αντίστροφη της προηγούμενης. Δηλαδή ενώ κάθε μέλος του Α μπορεί να σχετισθεί με ένα μόνο μέλος του Β κάθε μέλος του Β μπορεί να σχετισθεί με πολλά μέλη του Α και
- Αντιστοίχηση πολλά προς πολλά (m:m) : κάθε μέλος του Α μπορεί να σχετισθεί με πολλά μέλη του Β αλλά και κάθε μέλος του Β μπορεί να σχετισθεί με πολλά μέλη του Α.

2.3.2.2 Το Μοντέλο Οντοτήτων – Συσχετίσεων

Το Διάγραμμα Οντοτήτων-Συσχετίσεων (Entity Relationship Diagram ή ERD ή E-R Diagram) (Παράρτημα-Entity Relationship Diagram), είναι ένα δικτυωτό μοντέλο το οποίο περιγράφει τον τρόπο αποθήκευσης των δεδομένων του συστήματος σε ένα υψηλό επίπεδο αφαίρεσης (Yourdon, 1989). Αυτή η τεχνική έχει αναπτυχθεί από τον Chen (1976) και χρησιμοποιείται πάρα πολύ στη σχεδίαση των Βάσεων Δεδομένων. Παράδειγμα ενός τέτοιου διαγράμματος Οντοτήτων-Συσχετίσεων δίνεται στο επόμενο σχήμα :



Το παράδειγμα του προηγούμενου σχήματος αναφέρεται σε μια σχολή, η οποία προσφέρει κάποιες ειδικότητες. Παρατηρούμε επίσης ότι στο διάγραμμα αυτό, εκτός από τις οντότητες / γεγονότα και τις σχέσεις τους, απεικονίζονται και οι προτεραιότητες (cardinalities) των σχέσεων αυτών.

Αν θέλαμε να μεταφράσουμε το διάγραμμα αυτό θα λέγαμε ότι η σχολή προσφέρει πολλές ειδικότητες, στις οποίες σπουδάζουν πολλοί μαθητές. Κάθε ειδικότητα έχει έναν υπεύθυνο καθηγητή και πολλά τμήματα. Όπως είναι φυσικό, ο κάθε καθηγητής διδάσκει σε πολλά τμήματα και έχει πολλούς μαθητές, εκ των οποίων ο καθένας μπορεί να λαμβάνει τους βαθμούς του από πολλά τμήματα της σχολής, στην οποία σπουδάζει.

Η μετάφραση αυτή άρχισε από την οντότητα ΣΧΟΛΗ και διαγράφοντας κυκλική τροχιά από δεξιά προς τα αριστερά, έφθασε πάλι στην σχολή. Θα μπορούσε όμως αρχίζοντας από τη σχολή, να διαγράψει την αντίστροφη πορεία και να καταλήξει πάλι στη σχολή. Σε μια τέτοια περίπτωση θα εξετάζαμε τις ίδιες οντότητες, αλλά οι σχέσεις θα ήταν διαφορετικές.

Θα πρέπει να τονιστεί, ότι η Μετάφραση ενός Διαγράμματος Οντοτήτων-Συσχετίσεων μπορεί να αρχίσει από οποιοδήποτε σημείο (οντότητα) και να διαγράψει οποιαδήποτε πορεία. Αρκεί βέβαια να

εξετασθούν οι σχέσεις της κάθε μιας οντότητας ξεχωριστά με όλες τις σχετιζόμενες μ' αυτήν οντότητες.

Για την κατασκευή ενός Διαγράμματος Οντοτήτων-Συσχετίσεων τα βήματα που πρέπει να ακολουθηθούν είναι τα εξής :

1. Παράπεμψε τα δεδομένα στις διαδικασίες
2. Φτιάξε αρκετά Διαγράμματα Οντοτήτων-Συσχετίσεων, ανάλογα με τις απόψεις διαφορετικών χρηστών

   (α) Για κάθε οντότητα καθόρισε τους τύπους ιδιοτήτων της

   (β) Ζωγράφισε το Διάγραμμα Οντοτήτων-Συσχετίσεων
3. Ένωσε τις διαφορετικές απόψεις των χρηστών σ' ένα γενικό εννοιολογικό σχήμα.
4. Καθόρισε και βελτίωσε τους τύπους συγγένειας
5. Καθόρισε τους γενικούς περιορισμούς του γενικού εννοιολογικού σχήματος
6. Σχεδίασε την δομή των record

2.3.2.3 Σύγχρονες Μέθοδοι Μοντελοποίησης (Advanced Modeling Methods)

Το μοντέλο Οντοτήτων-Συσχετίσεων βασίζεται σε τρεις αφαιρέσεις :

- των ομάδων οντοτήτων (entity sets),
- των ομάδων συγγένειας (relationship sets) και
- των ιδιοτήτων (attributes).

Πρόσφατες μελέτες όμως έχουν επεκτείνει το μοντέλο Οντοτήτων-Συσχετίσεων, έτσι ώστε να περιλαμβάνει μερικά νεότερα στοιχεία όπως τις *εξαρτώμενες οντότητες (dependent entities)* και τις *υποομάδες (subsets).*

Α. Εξαρτώμενες Οντότητες

Οι ομάδες οντοτήτων ή συγγένειας αποτελούνται από αντικείμενα, τα οποία έχουν τα ίδια χαρακτηριστικά. Αυτό ισχύει επίσης και για τα αντικείμενα της ίδιας ομάδας εξαρτώμενων οντοτήτων (dependent entity set). Επιπρόσθετα, στη δεύτερη περίπτωση υπάρχει και ένα ακόμη κοινό χαρακτηριστικό : η ύπαρξή τους εξαρτάται από την ύπαρξη της οντότητας-γονέα (parent entity) σε μια άλλη ομάδα. Αυτή η εξάρτηση εμφανίζεται παραστατικά σαν ένα βέλος, το οποίο ξεκινά από την πατρική ομάδα οντότητας (parent entity set) και καταλήγει στην ομάδα εξαρτώμενων οντοτήτων (dependent entity set).

Ένα πολύ σημαντικό χαρακτηριστικό των εξαρτώμενων οντοτήτων είναι ότι αυτές έχουν σύνθετους αναγνωριστές (composite identifiers). Αυτοί οι αναγνωριστές αποτελούνται από τους αναγνωριστές της οντότητας-γονέα και ένα ακόμη χαρακτηριστικό που προσδιορίζει μοναδικά την εξαρτώμενη οντότητα μέσα στο γονέα.

Β. Υποομάδες

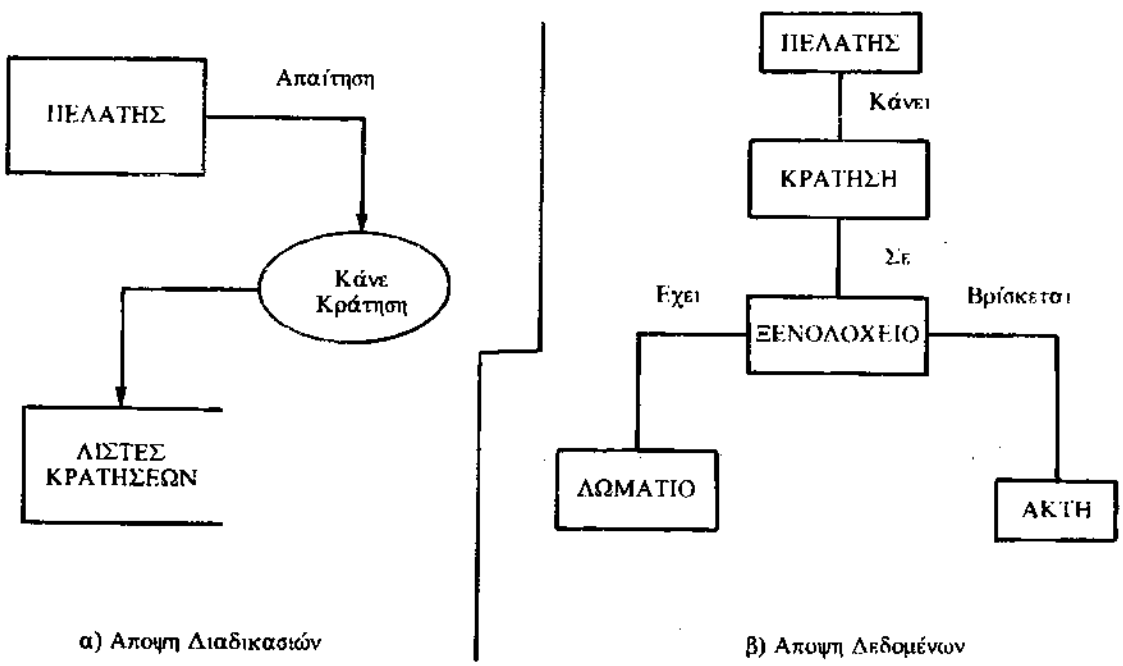
Οι υποομάδες είναι τμήματα των ομάδων και αυτές επίσης αποτελούνται από οντότητες. Οι υποομάδες χρησιμοποιούνται όταν είναι αναγκαίο να μοντελοποιήσουμε διαφορετικές μεθόδους μεταχείρισης οντοτήτων μέσα σε μια ομάδα οντοτήτων. Έτσι οντότητες μιας ομάδας μπορούν να είναι οντότητες σε υποομάδες της ίδιας ομάδας.

Σε οντότητες διαφορετικών υποομάδων είναι συνηθισμένο να έχουν μερικές διαφορετικές και μερικές κοινές ιδιότητες. Οι κοινές ιδιότητες παρουσιάζονται σαν ιδιότητες μιας υποομάδας παρουσιάζοντας μόνο σαν ιδιότητες αποκλειστικά δικές της.

Στις σύγχρονες μεθόδους μοντελοποίησης, για να προσδιορίσουμε τα μέλη μιας υποομάδας, το απλούστερο πράγμα που έχουμε να κάνουμε είναι να χρησιμοποιήσουμε τον ίδιο αναγνωριστή στην υποομάδα όπως και στο γονέα της υποομάδας.

2.3.2.4 Σύνδεση των Δεδομένων με τις Διαδικασίες

Η σχέση μεταξύ δεδομένων και διαδικασιών είναι πολύ σημαντική στα πρώτα στάδια της ανάλυσης δεδομένων. Η ανάλυση των δεδομένων και των διαδικασιών πρέπει να εκτελεσθεί παράλληλα έτσι ώστε οι γνώσεις που αποκομίζονται από το ένα μοντέλο να χρησιμοποιούνται για την κατασκευή και τη βελτίωση του άλλου. Τα αποτελέσματα αυτού του τρόπου κατασκευής των μοντέλων εγγράφονται σε έναν πίνακα, του οποίου οι γραμμές απεικονίζουν τις διαδικασίες και οι στήλες τα δεδομένα. Ένα παράδειγμα του τρόπου με τον οποίο μπορεί να κατασκευασθεί ένας τέτοιος πίνακας, δίνεται στο ακόλουθο σχήμα :



Στο σχήμα (α) απεικονίζεται πολύ απλά ένα σύστημα κράτησης θέσεων σ' ένα ξενοδοχείο. Το σύστημα απεικονίζεται από την άποψη των διαδικασιών, δηλ. παρουσιάζονται οι διαδικασίες που υπάρχουν σ' ένα τέτοιο σύστημα. Παρατηρούμε ότι ο πελάτης έχει μια απαίτηση, την οποία διαβιβάζει στον ταξιδιωτικό πράκτορα, ο οποίος με τη σειρά του προσπαθεί

να ικανοποιήσει την απαίτηση αυτή και να κάνει την κράτηση που επιθυμεί ο πελάτης.

Από την άλλη, στο σχήμα (β) απεικονίζεται το ίδιο σύστημα, αλλά από την άποψη των δεδομένων. Βλέπουμε δηλ. ότι κάποιος πελάτης κάνει μία κράτηση σ' ένα δωμάτιο ξενοδοχείου που βρίσκεται σε μια ορισμένη ακτή.

2.3.2.5 Ανάλυση Λειτουργικής Εξάρτησης

Η ανάλυση λειτουργικής εξάρτησης ασχολείται με τον προσδιορισμό και τη μοντελοποίηση των κανόνων που διέπουν τις σχέσεις των χαρακτηριστικών / ιδιοτήτων του συστήματος. Το Β είναι λειτουργικά εξαρτώμενο του Α, εάν μια τιμή του Α προσδιορίζει μοναδικά την τιμή του Β. Με άλλα λόγια, αν εμείς γνωρίζουμε την τιμή του Α, μπορούμε να προσδιορίσουμε μια μοναδική τιμή του Β. Σ' αυτήν την περίπτωση, το χαρακτηριστικό Α ονομάζεται «προσδιορίζον» χαρακτηριστικό. Ένα προσδιορίζον χαρακτηριστικό αποτελείται μερικές φορές από περισσότερα του ενός τμήματα Α και Γ. Αν συμβεί αυτό τότε το χαρακτηριστικό είναι λειτουργικά εξαρτώμενο του Α και του Γ ταυτόχρονα.

Η παγίδα που υπάρχει στην ανάλυση της Λειτουργικής Εξάρτησης είναι ότι μια σχέση λειτουργικής εξάρτησης μπορεί να προέλθει μερικές φορές από άλλες σχέσεις λειτουργικών εξαρτήσεων. Σε μια τέτοια περίπτωση, η ανάλυση πρέπει να συνεχισθεί μέχρι το σημείο εκείνο, όπου ο αναλυτής ή ο σχεδιαστής θα είναι βέβαιος ότι δεν υπάρχουν άλλες λειτουργικές εξαρτήσεις τέτοιου είδους στο μοντέλο του.

2.3.3 Ανάλυση της «ΣΥΜΠΕΡΙΦΟΡΑΣ» ως προς το χρόνο (Time – dependent behavior analysis)

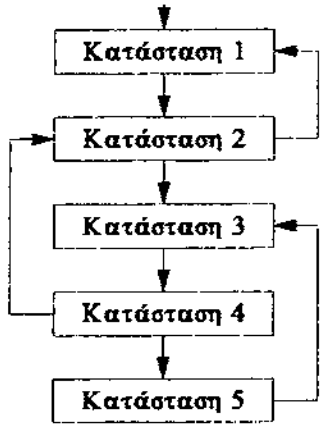
Η Ανάλυση της Συμπεριφοράς ως προς το Χρόνο και το κύριο εργαλείο μοντελοποίησης που χρησιμοποιεί το Διάγραμμα Μετάβασης – Κατάστασης (State – transition Diagram, STD) επικεντρώνονται στη συμπεριφορά του συστήματος ως προς το χρόνο.

Μέχρι πρόσφατα το μοντέλο αυτού του είδους ήταν σημαντικό μόνο για την ανάπτυξη ειδικών κατηγοριών συστημάτων, γνωστών ως Συστήματα Επεξεργασίας Πραγματικού Χρόνου (Real time systems) αλλά στις μέρες μας, είναι πολύ χρήσιμο εργαλείο για την ανάπτυξη μερικών μεγάλων και πολύπλοκων, προσανατολισμένων στο έργο συστημάτων.

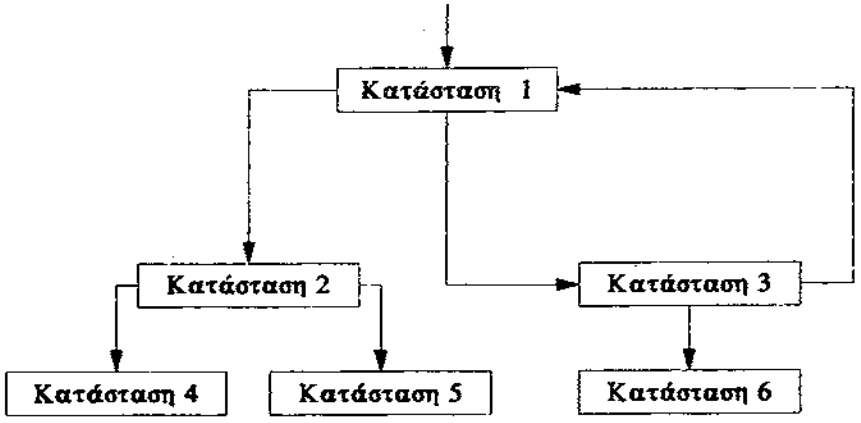
Τα κύρια τμήματα που απαρτίζουν ένα τυπικό Διάγραμμα Μετάβασης – Κατάστασης, είναι οι καταστάσεις και τα βέλη που απεικονίζουν τις διαδοχικές αλλαγές αυτών των καταστάσεων.

«Κατάσταση» ορίζεται (από το Λεξικό Webster's New Word Dictionary) «ένα σύνολο γεγονότων / περιστάσεων ή ιδιοτήτων που χαρακτηρίζουν ένα άτομο ή ένα πράγμα, σε μια δεδομένη χρονική στιγμή». Στην πραγματικότητα όμως δεν υπάρχει στατική κατάσταση. Αλλαγές πραγματοποιούνται σε όλες τις καταστάσεις. Αυτές τις αλλαγές προσπαθεί να απεικονίσει σε μια διαγραμματική μορφή και το Διάγραμμα Μετάβασης – Κατάστασης.

Παράδειγμα ενός τέτοιου διαγράμματος υπάρχει στο επόμενο σχήμα :



α) Αρχικές και τελικές καταστάσεις



β) Σύστημα πολλαπλών τελικών καταστάσεων

Σ' αυτό το σχήμα, παρατηρούμε ότι κάθε σύστημα έχει μια αρχική και μία ή περισσότερες τελικές καταστάσεις. Θα πρέπει όμως να είμαστε πολύ προσεχτικοί, γιατί οι διάφορες τελικές καταστάσεις είναι αλληλοαποκλειόμενες (mutually exclusive). Αυτό σημαίνει ότι μόνο μία από αυτές μπορεί να πραγματοποιηθεί κατά τη διάρκεια οποιασδήποτε εκτελέσεως του συστήματος.

Τι είναι αυτό που προκαλεί αυτές τις αλλαγές και αναγκάζει το σύστημα να μεταβαίνει από τη μία κατάσταση στην άλλη; Αυτό γίνεται εξαιτίας κάποιων «σημάτων» ή γεγονότων που λαμβάνουν χώρα κυρίως στο εξωτερικό περιβάλλον του συστήματος. Μετά από την έλευση του μηνύματος, το σύστημα θα πραγματοποιήσει τυπικά μία ή περισσότερες ενέργειες. Οι ενέργειες αυτές, που απεικονίζονται στο Διάγραμμα Μετάβασης – Κατάστασης, είναι είτε απαντήσεις που στέλνονται πίσω στο εξωτερικό περιβάλλον του συστήματος, είτε υπολογισμοί των οποίων τα αποτελέσματα καταγράφονται για να χρησιμοποιηθούν στο μέλλον.

2.3.3.1 Κατασκευάζοντας το Διάγραμμα Μετάβασης – Κατάστασης

Για την κατασκευή του Διαγράμματος Μετάβασης – Κατάστασης μπορούν να ακολουθηθούν δύο διαφορετικές προσεγγίσεις :

- Προσδιόρισε κατ' αρχήν όλες τις πιθανές καταστάσεις του συστήματος και διερεύνησε όλες τις αλλαγές καταστάσεων με κάποια σημασία για το σύστημα και κατόπιν, ανάπτυξε ολόκληρο το σύστημα.
- Άρχισε με την ανάπτυξη της πρώτης κατάστασης και κατόπιν μεθοδικά, βρες τον τρόπο να φτάσεις στα επόμενα στάδια και να τα αναπτύξεις επίσης.

Όταν το αρχικό Διάγραμμα Μετάβασης – Κατάστασης θα έχει τελειώσει, τότε σύμφωνα με τον Yourdon (1989) πρέπει να εκτελεσθούν οι παρακάτω οδηγίες ελέγχου της συνέπειας του Διαγράμματος :

(α) Έχουν καθορισθεί όλες οι καταστάσεις ;

(β) Μπορεί μια οποιαδήποτε κατάσταση να λάβει χώρα, αν υπάρχουν όλες οι κατάλληλες συνθήκες;

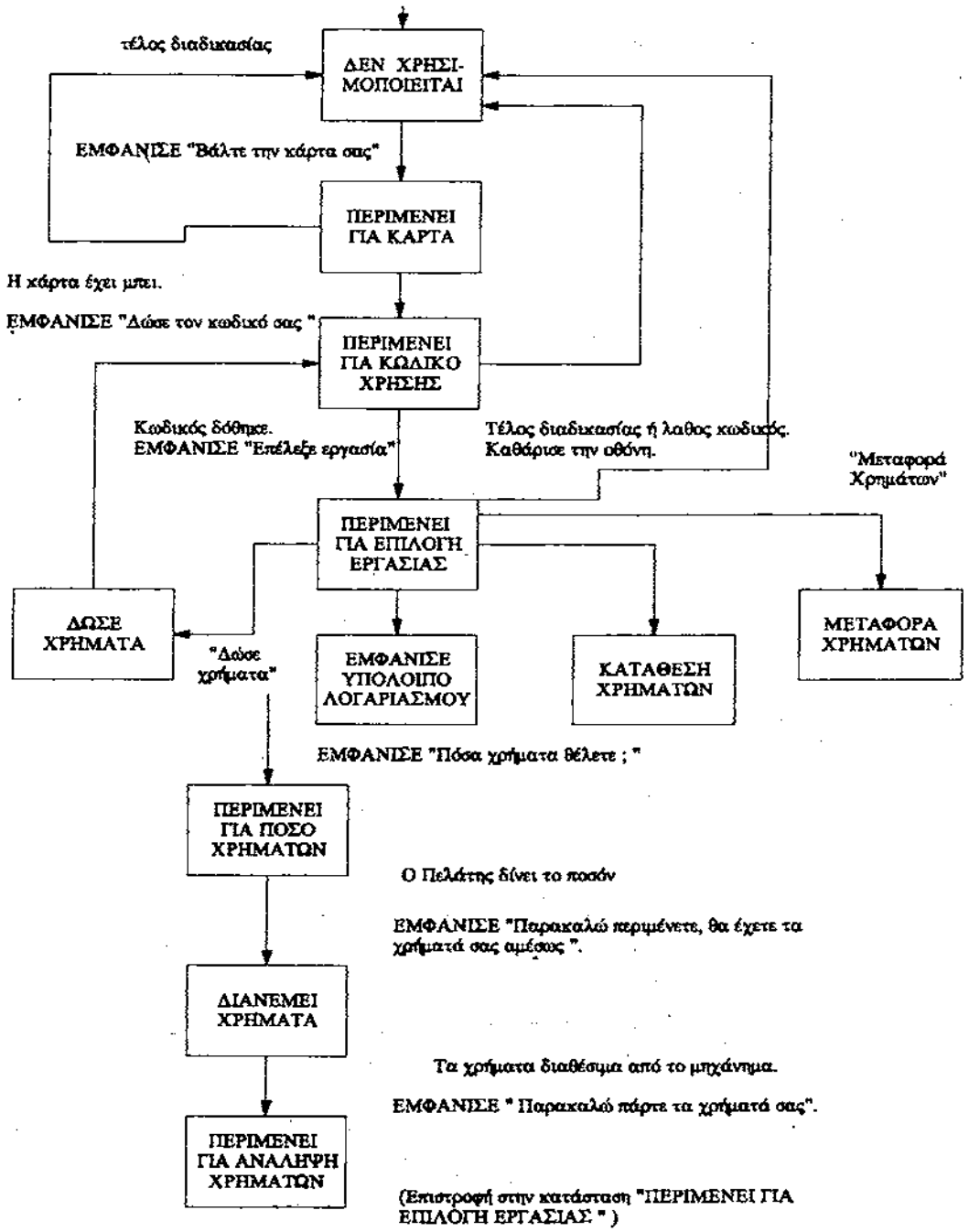
(γ) Μπορεί κάποιος να ξεφύγει απ' όλες τις καταστάσεις (αν υπάρχουν ιδιαίτεροι λόγοι) ;

(δ) Ανταποκρίνεται το σύστημα κατάλληλα σε κάθε κατάσταση και σ' όλες τις ενδεχόμενες συνθήκες ;

Αν η απάντηση σ' όλα αυτά τα ενδεχόμενα ερωτήματα είναι «ΝΑΙ» τότε το σύστημα που κατασκευάστηκε είναι καλό και ανταποκρίνεται στην πραγματικότητα.

Τέλος, το Διάγραμμα Μετάβασης – Κατάστασης είναι ένα πολύ ισχυρό εργαλείο μοντελοποίησης για την περιγραφή της συμπεριφοράς ενός συστήματος επεξεργασίας πραγματικού χρόνου (και όχι μόνο) αλλά επίσης και για την παρουσίαση του ανθρώπινου μεσολαβητικού παράγοντα πολλών συστημάτων άμεσης επεξεργασίας (on line systems).

Στο επόμενο σχήμα δίνεται ένα απλό παράδειγμα για το πως περιγράφει ένα Διάγραμμα Μετάβασης – Κατάστασης ένα σύστημα.



Το σχήμα αυτό απεικονίζει τη διαδικασία χρησιμοποίησης των αυτόματων μηχανών (ATM) για ανάληψη χρημάτων ή τη διαδικασία χρησιμοποίησης άλλων υπηρεσιών. Υποθέτουμε στην αρχή ότι το μηχάνημα δεν χρησιμοποιείται. Στην οθόνη υπάρχει το μήνυμα «βάλτε την κάρτα σας» και το μηχάνημα είναι έτοιμο να δεχθεί αυτή την κάρτα. Μόλις εμφανισθεί κάποιος πελάτης που θέλει να χρησιμοποιήσει το μηχάνημα και βάλει την

κάρτα του στο χώρο υποδοχής, τότε εμφανίζεται στην οθόνη το μήνυμα «δώστε τον κωδικό σας» και το μηχάνημα είναι έτοιμο να δεχθεί τον κωδικό. Για να φθάσουμε όμως σ' αυτό το στάδιο, η κάρτα που έβαλε ο πελάτης θα πρέπει να κριθεί έγκυρη από το μηχάνημα, διαφορετικά η κάρτα επιστρέφεται και επανερχόμαστε στην αρχική κατάσταση μη χρησιμοποίησης του μηχανήματος. Αφού λοιπόν δοθεί ο κωδικός από τον πελάτη και κριθεί ότι όντως είναι ο σωστός κωδικός της κάρτας, τότε εμφανίζεται στην οθόνη το μήνυμα «επέλεξε εργασία» και ο πελάτης πρέπει να δηλώσει στο μηχάνημα τι είδους υπηρεσία θέλει να του παρασχεθεί. Φυσικά, αν ο κωδικός που δόθηκε δεν είναι ο σωστός, το μηχάνημα επανέρχεται στην αρχική του κατάσταση. Ο πελάτης επιλέγει μια υπηρεσία και αφού αυτή εκτελεσθεί, τότε το μηχάνημα επανέρχεται στην κατάσταση, όπου περιμένει από τον πελάτη καινούριες διαταγές γι' αυτό που θέλει ο πελάτης να κάνει. Αν αυτός θέλει και κάποια άλλη υπηρεσία, δίνει την ανάλογη επιλογή. Αν όμως δεν επιθυμεί τίποτα, τότε δηλώνει ότι δεν θέλει τίποτε άλλο, του επιστρέφεται η κάρτα και το μηχάνημα επανέρχεται στην αρχική του κατάσταση, όπου περιμένει να εξυπηρετήσει έναν νέο πελάτη.

2.3.4 Ισορρόπηση των μοντέλων που εξετάσθηκαν (Balancing the described models)

Ο αναλυτής δεν είναι υποχρεωμένος, κατά τη διάρκεια της φάσης της ανάλυσης, να χρησιμοποιήσει όλα αυτά τα μοντέλα και τις τεχνικές, αλλά μόνο εκείνο το μοντέλο (ή εκείνα τα μοντέλα) που αυτός κρίνει ότι τον εξυπηρετούν καλύτερα στη συγκεκριμένη περίπτωση. Τα προβλήματα όμως αρχίζουν όταν εμείς θέλουμε γιατί το κρίνουμε αναγκαίο να χρησιμοποιήσουμε πολλά μοντέλα ταυτόχρονα.

Για παράδειγμα όταν μοντελοποιούμε τρεις διαφορετικές πλευρές / απόψεις του συστήματος (λειτουργίες, δεδομένα, χρονικό ορίζοντα) είναι πολύ εύκολο, χρησιμοποιώντας διαφορετικά μοντέλα, ένα για την ανάπτυξη κάθε πλευράς, να δημιουργήσουμε αρκετές διαφορετικές πολύ ασυνεπείς και αντιφατικές τους μεταξύ τους ερμηνείες του συστήματος. Αυτού του είδους τα δια-μοντελικά λάθη είναι συνήθως τα πλέον ύπουλα και αυτά που πολύ δύσκολα διορθώνονται.

Ισορροπημένη (balanced), ονομάζεται σύμφωνα με τον Yourdon (1989), μια δομημένη περιγραφή του συστήματος, όπου κάθε εργαλείο μοντελοποίησης έχει ελεγχθεί, σε σχέση με όλα τα υπόλοιπα μοντέλα για να διαπιστωθεί η συνέπεια και η συνοχή του.

Το πιο συχνά εμφανιζόμενο λάθος ισορροπίας, είναι οι ορισμοί που λείπουν δηλαδή κάτι που είναι κατάλληλα ορισμένο σ' ένα μοντέλο, δεν αναφέρεται σ' ένα άλλο.

Ο δεύτερος πιο συχνά εμφανιζόμενος τύπος λαθών είναι αυτός της ασυνέπειας και της αντιφατικότητας, δηλαδή η ίδια πραγματικότητα περιγράφεται με διαφορετικό, αντιφατικό και αντικρουόμενο τρόπο από δύο διαφορετικά μοντέλα.

2.3.4.1 Ισορρόπηση του Διαγράμματος Ροής Δεδομένων με το Λεξικό Δεδομένων και με τις Περιγραφές των Διαδικασιών

Οι αναλυτές συστημάτων πρέπει να επιθεωρούν συχνά, τόσο τα Διαγράμματα Ροής Δεδομένων, όσο και τα Λεξικά Δεδομένων, για να βεβαιώνονται ότι πράγματι υπάρχει ισορροπία μεταξύ των δύο αυτών τεχνικών.

Οι κανόνες που διέπουν τη σχέση ισορροπίας τους είναι πολύ απλοί. Κάθε ροή δεδομένων και κάθε αποθήκη δεδομένων, πρέπει να ορίζεται από το Λεξικό Δεδομένων. Σε αντίθετη περίπτωση, θεωρούνται μη ορισμένα. Το ίδιο πρέπει να συμβαίνει και αντίστροφα. Δηλαδή, κάθε στοιχείο δεδομένων και κάθε αποθήκη δεδομένων, που ορίζεται στο Λεξικό Δεδομένων, πρέπει να εμφανίζεται κάπου μέσα στο Διάγραμμα Ροής Δεδομένων.

Μια διαφορετική περίπτωση είναι η ισορροπία ανάμεσα στο Διάγραμμα Ροής Δεδομένων και στις Περιγραφές των Διαδικασιών.

Εδώ οι κανόνες που διέπουν τη σχέση ισορροπίας είναι οι παρακάτω :

- Κάθε φυσαλίδα του Διαγράμματος Ροής Δεδομένων πρέπει να σχετίζεται με ένα χαμηλότερου επιπέδου Διάγραμμα Ροής Δεδομένων ή με την περιγραφή μιας Διαδικασίας, αλλά οπωσδήποτε όχι και με τα δύο, γιατί σε μια τέτοια περίπτωση το μοντέλο θα είναι πλεονασματικό δηλαδή θα έχει παραπάνω στοιχεία απ' ότι πρέπει.
- Κάθε χαρακτηριστικό μιας Διαδικασίας πρέπει να έχει σχέση με φυσαλίδα τελευταίου επιπέδου μέσα στο Διάγραμμα Ροής Δεδομένων.
- Εισαγωγές και εξαγωγές πρέπει να ταιριάζουν.

Πρέπει οπωσδήποτε να διευκρινισθεί ότι τα παραπάνω αναφέρονται ειδικά σε φυσαλίδες επεξεργασίας (διαδικαστικές). Για τις φυσαλίδες ελέγχου των Διαγραμμάτων Ροής Δεδομένων, υπάρχει αντιστοιχία μεταξύ των φυσαλίδων και των σχετιζόμενων Διαγραμμάτων Μετάβασης – Κατάστασης.

2.3.4.2 Ισορρόπηση των Περιγραφών των Διαδικασιών με το Διάγραμμα Ροής Δεδομένων και το Λεξικό Δεδομένων

Για να ισορροπήσουμε τις Περιγραφές των Διαδικασιών με το Διάγραμμα Ροής Δεδομένων και το Λεξικό Δεδομένων κάθε αναφορά των δεδομένων στις Περιγραφές των Διαδικασιών πρέπει να ικανοποιεί έναν από τους παρακάτω (Yourdon, 1989) :

- να ταυτίζει τα ονόματα των ροών δεδομένων ή των αποθηκών δεδομένων που σχετίζονται με τις φυσαλίδες όπως περιγράφονται από τις περιγραφές των διαδικασιών ή
- να είναι ένας τοπικός όρος, κατηγορηματικά ορισμένος στις Περιγραφές των Διαδικασιών ή
- να εμφανίζεται σαν συστατικό μέρος μιας εισαγωγής στο Λεξικό Δεδομένων για ένα Διάγραμμα Ροής Δεδομένων ή Αποθήκη Δεδομένων που είναι συνδεδεμένη με τη φυσαλίδα.

2.3.4.3 Ισορρόπηση του Διαγράμματος Οντοτήτων – Συσχετίσεων με το Διάγραμμα Ροής Δεδομένων και τις Περιγραφές των Διαδικασιών

Ακόμη κι αν το Διάγραμμα Οντοτήτων – Συσχετίσεων και το Διάγραμμα Ροής Δεδομένων, παρουσιάζουν δύο τελείως διαφορετικές πλευρές του συστήματος, υπάρχουν κάποιες σχέσεις που πρέπει να διατηρούνται έτσι ώστε όλο το σύστημα να είναι ολοκληρωμένο, σωστό και συνεπές.

Μερικές απ' αυτές παρουσιάζονται παρακάτω :

- Κάθε αποθήκη του Διαγράμματος Ροής Δεδομένων πρέπει να ανταποκρίνεται σ' έναν τύπο συστημάτων ή σε μια σχέση (relationship) ή σ' ένα συνδυασμό ενός τύπου οντοτήτων και μιας σχέσης.

- Τα ονόματα των οντοτήτων του Διαγράμματος Οντοτήτων Συσχετίσεων και τα ονόματα των αποθηκών δεδομένων του Διαγράμματος Ροής Δεδομένων πρέπει να ταιριάζουν (ταυτίζονται).
- Τα εισαγόμενα στο Λεξικό Δεδομένων πρέπει να ισχύουν για το Διάγραμμα Ροής Δεδομένων, όπως επίσης και για το Διάγραμμα Οντοτήτων – Συσχετίσεων.

Ομοίως, υπάρχουν κανόνες που βεβαιώνουν ότι το Διάγραμμα Οντοτήτων – Συσχετίσεων είναι συνεπές με τις Περιγραφές των Διαδικασιών. Αυτοί οι κανόνες συνίστανται στο ότι μια συνδυασμένη ομάδα όλων των διαδικασιών πρέπει στην ολότητά τους :

- να δημιουργούν και να εξαφανίζουν παραδείγματα του κάθε τύπου οντοτήτων και σχέσεων που εμφανίζονται σ' ένα Διάγραμμα Οντοτήτων – Συσχετίσεων
- μερικές φυσαλίδες ορίζουν τιμές για κάθε τμήμα δεδομένων που αποδίδεται σε κάθε παράδειγμα κάθε τύπου οντοτήτων και έτσι σώζουν διαδικασίες του Διαγράμματος Ροής Δεδομένων, χρησιμοποιώντας τιμές του κάθε τμήματος δεδομένων.

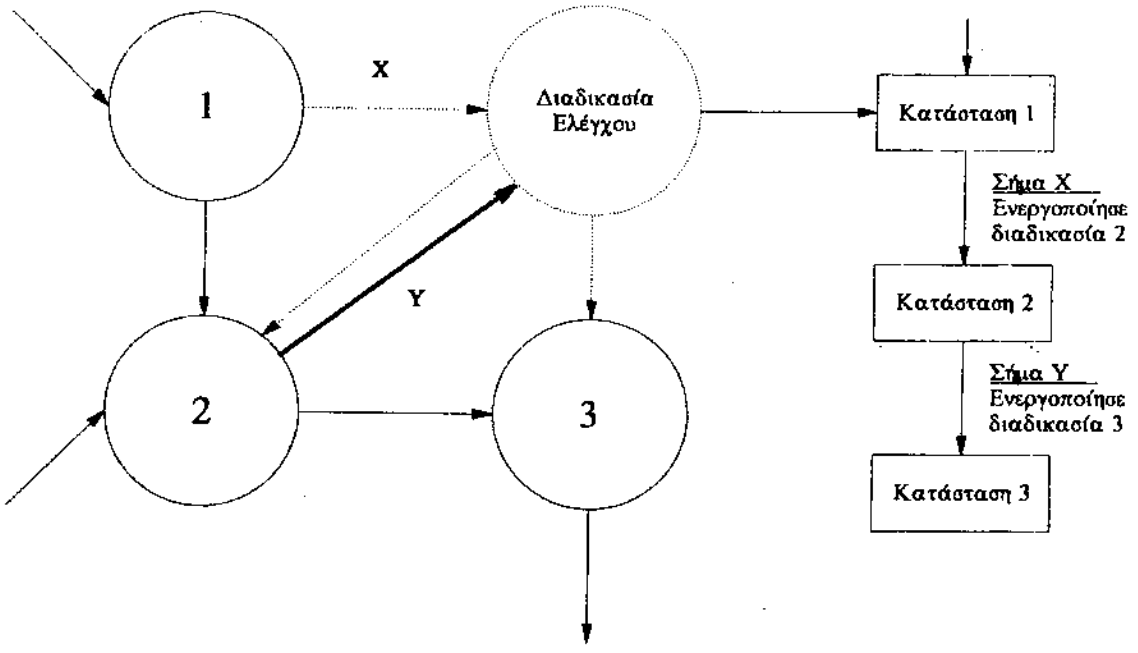
2.3.4.4. Ισορρόπηση του Διαγράμματος Ροής Δεδομένων με το Διάγραμμα Μετάβασης – Κατάστασης

Σύμφωνα με τον Yourdon (1989) μπορεί να θεωρηθεί ότι υπάρχει ισορροπία μεταξύ του Διαγράμματος Ροής Δεδομένων και του Διαγράμματος Μετάβασης Κατάστασης, όταν ισχύουν οι παρακάτω κανόνες :

- Κάθε φυσαλίδα ελέγχου του Διαγράμματος Ροής Δεδομένων πρέπει να συνδέεται με το Διάγραμμα Μετάβασης Κατάστασης όπως στις περιγραφές των Διαδικασιών. Το ίδιο πρέπει να συμβαίνει και αντίστροφα.
- Κάθε συνθήκη του Διαγράμματος Μετάβασης – Κατάστασης πρέπει να ανταποκρίνεται (αντιστοιχεί) σε μια νέα εισερχόμενη ροή ελέγχου

μέσα στη διαδικασία ελέγχου που σχετίζεται με το Διάγραμμα Μετάβασης – Κατάστασης. Το ίδιο πρέπει να συμβαίνει και αντίστροφα.

- Κάθε ενέργεια μέσα στο Διάγραμμα Μετάβασης – Κατάστασης πρέπει να ανταποκρίνεται (αντιστοιχεί) σε μια εξερχόμενη ροή ελέγχου μέσα στη διαδικασία ελέγχου που σχετίζεται με το Διάγραμμα Μετάβασης – Κατάστασης. Το ίδιο πρέπει να συμβαίνει και αντίστροφα. Οι αντιστοιχίες μεταξύ του Διαγράμματος Ροής Δεδομένων και του Διαγράμματος Μετάβασης – Κατάστασης μπορεί να κατανοηθούν καλύτερα, αν παρατηρήσουμε το επόμενο σχήμα. Η μετάβαση από τη μια κατάσταση στην άλλη γίνεται όταν ληφθεί ένα σήμα ελέγχου και είναι ανάλογη με το σήμα.



2.4 Σχεδίαση Συστημάτων (Systems Design)

Ο *Σχεδιασμός του Λογισμικού (Software Design)* είναι η διαδικασία εκείνη που βάσει διαφόρων αρχών και τεχνικών μεταφέρει το πρόβλημα από την αρχική μορφή των απαιτήσεων που το περιγράφουν σε μία καλώς ορισμένη και επαρκώς περιεκτική αναπαράσταση λογισμικού, πριν το τελικό στάδιο της υλοποίησής του. Προηγείται της δημιουργίας κώδικα και του ελέγχου του λογισμικού και ξεκινά αμέσως μετά την ανάλυση και τον καθορισμό των απαιτήσεων.

Οι αποφάσεις που παίρνονται κατά το στάδιο του σχεδιασμού είναι καθοριστικές για την επιτυχία της υλοποίησης του λογισμικού και της ευκολίας συντήρησής του αργότερα. Ακόμα περισσότερο συμβάλλει στην ποιότητά του.

Ο σχεδιαστής χρησιμοποιώντας το ένστικτο και τη δική του κρίση, την εμπειρία του από προηγούμενες εφαρμογές, καθώς και ένα σύνολο αρχών που τον οδηγούν στη σωστή ανάπτυξη της φάσης του σχεδιασμού και κριτηρίων που εξασφαλίζουν την ποιότητά του, στοχεύει στη δημιουργία ενός μοντέλου πάνω στο οποίο θα χτιστεί κατόπιν όλο το σύστημα.

Μέχρι πριν λίγα χρόνια, δεν υπήρχε κάποιο ιδιαίτερο εργαλείο για να ξεχωρίσουμε το στάδιο της ανάλυσης (ΤΙ πρέπει να κάνει το σύστημα) από το στάδιο της σχεδίασης (ΠΩΣ πρόκειται να γίνει). Ευτυχώς, πραγματοποιήθηκαν αρκετές προσπάθειες προς αυτή την κατεύθυνση και έτσι τα τελευταία χρόνια έκαναν την εμφάνισή τους αρκετές τεχνικές σχεδίασης. Μερικές από αυτές έχουν προέλθει από τους L.Constantine (1974), Myers (1975) και Yourdon (1976).

Μερικοί ενδεικτικοί ορισμοί των όρων Σχεδίαση Συστημάτων (System Design) και Δομημένη Σχεδίαση (Structured Design) όπως αυτοί έχουν χρησιμοποιηθεί κατά καιρούς από διάφορους συγγραφείς είναι :

- Οι Stevens, Myers και Constantine (1974) ορίζουν τη Δομημένη Σχεδίαση ως «ένα σύνολο από προτεινόμενα γενικά ζητήματα και τεχνικές σχεδίασης προγραμμάτων που κάνουν τις διαδικασίες κωδικοποίησης (coding), εξάλειψης λαθών (debugging) και

τροποποίησης (modification) ευκολότερες , γρηγορότερες και λιγότερο ακριβές, εφ' όσον μειώνουν την πολυπλοκότητα του συστήματος.

- Η «Δομημένη Σχεδίαση μπορεί να οριστεί ως μία διαδικασία που έχει σαν εισαγόμενα στοιχεία τη Δομημένη Περιγραφή των Προδιαγραφών του Συστήματος, από το στάδιο της Δομημένης Ανάλυσης, καθώς επίσης και τις Προδιαγραφές του Υλικού (hardware) και του Λογισμικού (software) και παράγει σαν εξαγόμενο κάτι που ονομάζεται Ολοκληρωμένο Σχέδιο, το οποίο με τη σειρά του είναι το εισαγόμενο στοιχείο της διαδικασίας εφαρμογής (implementation) του νέου συστήματος» (Connor, 1985).
- Οι G.Gane και T. Sarson (1986) ορίζουν τη Σχεδίαση ως «την επαναληπτική διαδικασία που χρησιμοποιεί το λογικό μοντέλο ενός συστήματος μαζί με μια ομάδα ισχυρά καθορισμένων αντικειμενικών στόχων αυτού του συστήματος και παράγει τα χαρακτηριστικά (προδιαγραφές) του Φυσικού Συστήματος (physical system) το οποίο θα ικανοποιήσει αυτούς τους στόχους».
- Οι Layzell και Loucopoulos (1987) τονίζουν ότι «η Σχεδίαση είναι μια διαδικασία συναρμολόγησης, με την οποία συγκεντρώνονται ομάδες λειτουργιών και δεδομένα μέσα από τον καθορισμό της δομής των διαδικασιών και δεδομένων αντίστοιχα».

2.4.1 Αρχές Σχεδίασης Συστημάτων

Η διαδικασία σχεδιασμού του λογισμικού ξεκινά από μια γενικότερη και αφηρημένη αναπαράσταση της λύσης του προβλήματος και προχωρά προς τη βαθμιαία συγκεκριμενοποίησή του και περιγραφή των εσωτερικών λειτουργιών του. Αυτό γίνεται βάσει μιας top-down μεθόδου που παρουσιάζει ο Wirth και η οποία «χτίζει» το λογισμικό εμβαθύνοντας διαδοχικά στις διαδικαστικές λεπτομέρειες κάθε επιπέδου αυτού. Η λογική είναι η εξής : Σε κάθε βήμα μία ή περισσότερες οδηγίες διασπώνται σε περισσότερες και λεπτομερέστερες μέχρι την απλή αναπαράσταση τους σε όρους μιας γλώσσας προγραμματισμού. Σε κάθε φάση εκλέπτυνσης (refinement) παίρνονται οι αντίστοιχες σχεδιαστικές αποφάσεις. Ο σχεδιαστής πρέπει να γνωρίζει καλά τα εκάστοτε κριτήρια που τις προσδιορίζουν και τις εναλλακτικές λύσεις που προσφέρουν.

Με την *τμηματοποίηση (modulation)* του λογισμικού, δηλ. τη διάσπαση του σε ξεχωριστά προσπελάσιμα συστατικά όπως υπορουτίνες, συναρτήσεις, διαδικασίες, ο σχεδιαστής είναι σε θέση να αντιμετωπίσει πιο άμεσα, εύκολα και εύστοχα τα επιμέρους τμήματα του λογισμικού. Η ολοκλήρωση τους θα οδηγήσει αργότερα στην εκπλήρωση των απαιτήσεων.

Η επιλογή και ο προσδιορισμός των κατάλληλων δομών δεδομένων (data structures) που θα χρησιμοποιηθούν από το λογισμικό είναι σημαντικά για τον τελικό σχεδιασμό των διαδικασιών αυτού. Οι δομές δεδομένων αναπαριστούν τη λογική σχέση των διαφόρων δεδομένων μεταξύ τους. Υπαγορεύουν, δε την οργάνωση, τις μεθόδους προσπέλασης, το βαθμό συσχέτισης και τους τρόπους επεξεργασίας της πληροφορίας.

Οι βασικές δομές δεδομένων που χρησιμοποιούνται και για το χτίσιμο πολύπλοκων δομών δεδομένων είναι:

- το αντικείμενο,
- το ακολουθιακό διάνυσμα και το διάνυσμα και
- ο – η διάστατος χώρος

Προϋποθέτοντας την αρχή της τμηματοποίησης (modulation), *η αρχή της Απόκρυψης της πληροφορίας (Information Hiding)* συνιστά το σχεδιασμό των τμημάτων που θα προκύψουν από την τμηματοποίηση του λογισμικού με τρόπο ώστε η πληροφορία που θα περιέχεται στο καθένα να είναι απρόσιτη από τα υπόλοιπα που δεν τη χρειάζονται.

Η Αρχή της Ανεξαρτησίας των Λειτουργιών (Functional Independence) που επίσης προϋποθέτει την τμηματοποίηση του σχεδιασμού του λογισμικού, πολύ απλά συνιστά την ανάθεση της υλοποίησης μιας συγκεκριμένης λειτουργίας σε κάθε τμήμα και ενός απλού περιβάλλοντος διασύνδεσης-επικοινωνίας (interface) με τα άλλα τμήματα του προγράμματος. Έτσι, γίνεται ευκολότερος ο έλεγχος και η συντήρησή τους, αφού περιορίζεται η επέκτασή των αλλαγών που θα γίνουν σε ένα σημείο του κώδικα στα άλλα τμήματα, μειώνεται η διάδοση του λάθους και τα διακεκριμένα τμήματα μπορούν να ξαναχρησιμοποιηθούν.

Η ανεξαρτησία των λειτουργιών τονίζει ο Pressman είναι κλειδί για τον καλό σχεδιασμό, που ο ίδιος είναι κλειδί στην ποιότητα του λογισμικού. Μέτρο της ανεξαρτησίας των λειτουργιών αποτελούν η Συνοχή και η Σύνδεση των τμημάτων μεταξύ τους.

2.4.2 Αντικειμενικοί Στόχοι της Σχεδίασης

Οι πλέον σημαντικοί στόχοι της Σχεδίασης είναι να παραδώσει στους χρήστες ένα σύστημα που θα ικανοποιεί τις απαιτήσεις και τις ανάγκες τους, διαθέτοντας όλες τις ζητούμενες απ' αυτούς λειτουργίες. Αν συλλογιστούμε όμως ότι υπάρχουν πολλά σωστά σχέδια, τότε πως μπορεί να κατασκευασθεί το καλύτερο δυνατό σχέδιο;

Υπάρχουν τρεις κύριοι στόχοι / κριτήρια, τα οποία πρέπει να έχει ο σχεδιαστής στο μυαλό του, ενώ αναπτύσσει και αξιολογεί ένα σχέδιο. Αυτά είναι τα παρακάτω :

- ***Λειτουργία / Απόδοση (performance)*** : είναι το πόσο γρήγορα το σχέδιο θα είναι σε θέση να εκτελεί τις καθορισμένες λειτουργίες. Εκφράζεται με τους όρους :
  1. *Throughput,* που είναι ο αριθμός εντολών, οι οποίες εκτελούνται ανά δευτερόλεπτο ή κάποια άλλη μέτρηση της αποδοτικότητας εκτέλεσης εντολών (π.χ. υπολογισμοί ανά ώρα).
  2. *Run time*, που είναι ο χρόνος εκτέλεσης ενός προγράμματος
  3. *Response time*, που είναι ο χρόνος ανταπόκρισης του προγράμματος
- ***Έλεγχος (Control)*** : είναι το πόσο ασφαλές είναι το σχέδιο σε σχέση με τα ανθρώπινα και μηχανικά λάθη
- ***Ικανότητα για αλλαγή ή ευκαμψία (changeability)*** : είναι ένα μέτρο του χρόνου που απαιτείται, για να γίνουν τυχόν αλλαγές στο σύστημα.

Αν και δεν συμβαίνει πάντοτε, αυτοί οι τρεις παράγοντες, μετρητές βρίσκονται σε αντίθεση ο ένας με τους άλλους. Έτσι, γίνεται πλέον κατανοητό ότι είναι πολύ δύσκολο να βρεθεί ένα τέλειο σχέδιο.

Για να αναπτυχθεί ένα σχέδιο που θα ικανοποιεί τον καλύτερο συνδυασμό αυτών των στόχων, θα πρέπει να ακολουθηθούν τα παρακάτω βήματα :

1. εξασφάλιση από την τυχόν ύπαρξη φυσικών ζητημάτων στο σχέδιο
2. προσδιορισμός της εγκυρότητας του σχεδίου
3. καθορισμός της αρχιτεκτονικής του συστήματος
4. προσδιορισμός των ζητημάτων ασφαλείας και προστασίας
5. προσδιορισμός των λειτουργικών αναγκών
6. καθορισμός και αξιολόγηση των εναλλακτικών λύσεων
7. ανάπτυξη του σχεδίου του συστήματος και
8. εξέταση και έγκριση του σχεδίου του συστήματος

2.4.3 Ανάλυση των σταδίων Σχεδίασης Συστημάτων

Στα δύο πρώτα βήματα, ο σχεδιαστής πρέπει να συνεργάζεται πολύ στενά με τον αναλυτή, για να απομακρύνουν κάθε φυσικό ζήτημα από το σύνολο των Διαγραμμάτων Ροής Δεδομένων και επίσης για να επικυρώσουν το γεγονός, ότι όλες οι ανάγκες και απαιτήσεις των χρηστών έχουν ήδη προσδιοριστεί.

Οι διαδικασίες της επικύρωσης πρέπει να αφορούν τα παρακάτω θέματα :

- Δυνατότητα αποδοχής (acceptability)
- Ακρίβεια (accuracy)
- Πληρότητα (completeness)
- Τμηματοποίηση (δόμηση) και έλεγχος κόστους

Το τρίτο βήμα είναι αυτό στο οποίο πρέπει να προσδιοριστούν τα όρια αυτοματοποίησης του συστήματος και επίσης να προετοιμασθούν οι φυσικές εναλλακτικές λύσεις που θα διαμορφώσουν τη βάση για τον προσδιορισμό του τελικού (ενδεχομένως) κόστους του συστήματος. Σ' αυτήν την προσπάθεια το κύριο εργαλείο που θα χρησιμοποιηθεί είναι το Διάγραμμα Ροής Δεδομένων. Πρέπει να τονισθεί εδώ ότι ανάλογα με τα καθορισμένα όρια για αυτοματοποίηση των διαδικασιών, θα σχεδιασθούν αρκετά διαφορετικά μοντέλα Οντοτήτων Συσχετίσεων. Όταν, λοιπόν, αποφασίσουμε σε ποιο βαθμό (ποιοτικά) θα αυτοματοποιηθεί το σύστημα, τότε πρέπει να αποφασίσουμε και ποιος θα είναι ο τύπος του συστήματος που θα χρησιμοποιηθεί.

Υπάρχουν οι ακόλουθες εναλλακτικές προτάσεις :

- Μαζική (batch) ή Σύγχρονη (on line) Επεξεργασία,
- Κεντρική (central) ή Αποκεντρωμένη,
- Κατανεμημένη (distributed) Επεξεργασία

Η τελική απόφαση θα εξαρτάται πάντα από τους στόχους που έχουν τεθεί και από τους σκοπούς για τους οποίους προορίζεται το σύστημα.

Όσον αφορά στο τέταρτο βήμα, που σχετίζεται με τα ζητήματα της ασφάλειας και της προστασίας, ο σχεδιασμός του συστήματος πρέπει να λάβει υπόψη του την προστασία του συστήματος και των δεδομένων ενάντια οποιουδήποτε μη εξουσιοδοτημένου χρήστη.

Σε γενικές γραμμές, λοιπόν, ο σχεδιαστής θα πρέπει να λάβει υπόψη του τα θέματα:

- της Φυσικής Ασφάλειας (Physical Security),
- της Ασφάλειας Πρόσβασης και Χρήσης (Access Security) και τέλος
- της Ασφάλειας Μεταβίβασης (Transition Security).

Στο πέμπτο βήμα, θα πρέπει να προσδιορισθούν οι τύποι των εσωτερικών ελέγχων που μπορούν να βρεθούν σ' ένα πληροφοριακό σύστημα.

Αυτοί διαιρούνται σε 2 κατηγορίες :

- ***Λειτουργικοί Έλεγχοι (Operational Controls)***

  Τμηματοποίηση Εργασίας

  Έλεγχος Εισόδου (Input controls)

  Έλεγχοι Επεξεργασίας (Processing controls)

  Έλεγχοι Εξόδου (Output controls)

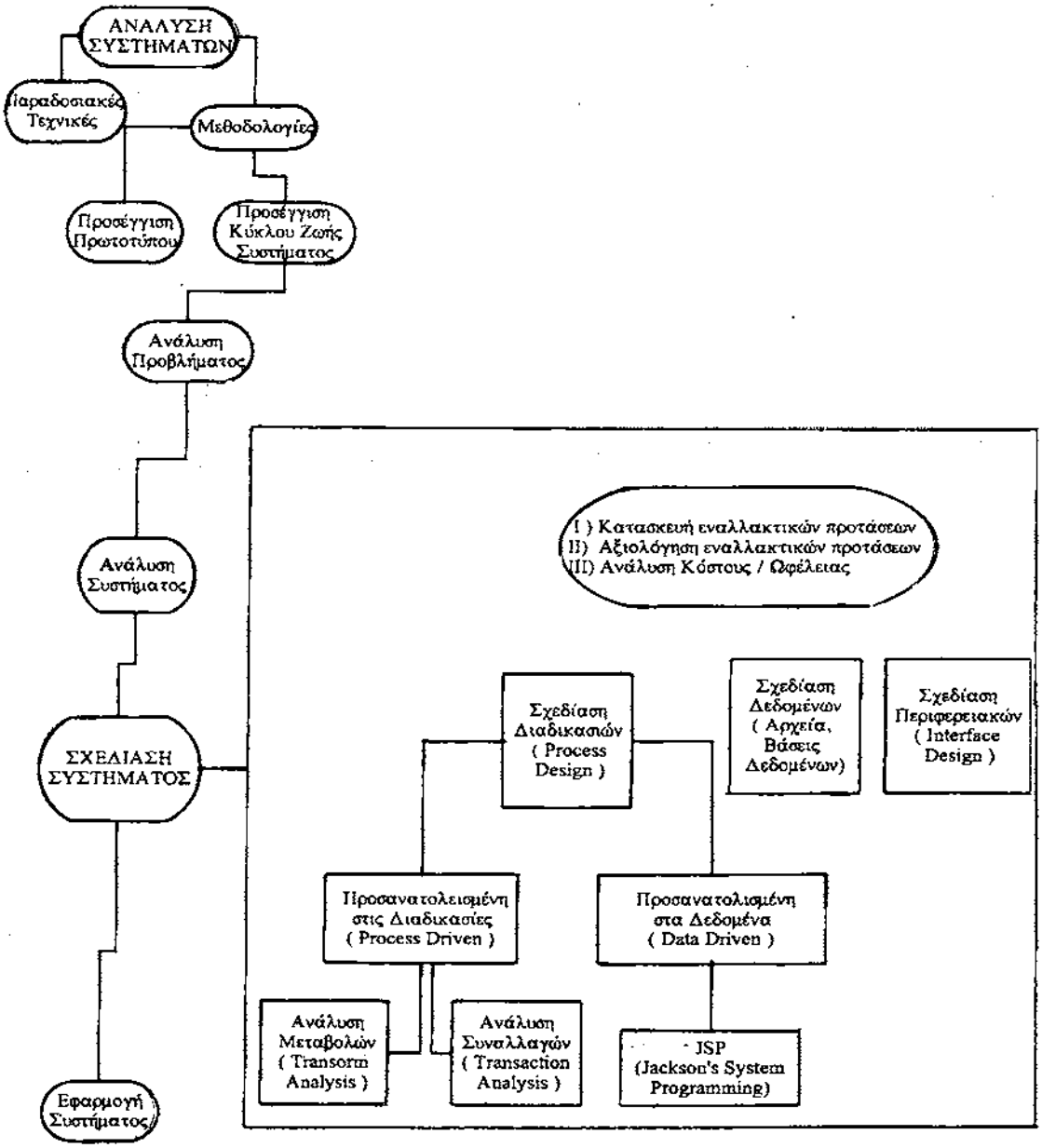
- ***Διοικητικοί / Διαχειριστικοί Έλεγχοι (Administrative Controls)***

Έχοντας προσεγγίσει και το πέμπτο βήμα ο σχεδιαστής του συστήματος πρέπει να προσδιορίσει αρκετές πιθανές εναλλακτικές λύσεις και να αποτιμήσει το κόστος και τα οφέλη της κάθε μία από αυτές.

2.4.4 Ανάπτυξη Σχεδίασης Συστημάτων (Developing a System Design)

Στο στάδιο της σχεδίασης ενός συστήματος ο σχεδιαστής πρέπει να συγκεντρώσει την προσοχή του σε τρία διαφορετικά θέματα :

- Σχεδίαση Διαδικασιών (Process Design)
- Σχεδίαση Δεδομένων (Data Design)
- Σχεδίαση Ενδιάμεσων και Περιφερειακών (Interface Design)


ΑΝΑΛΥΣΗ ΣΥΣΤΗΜΑΤΩΝ
Παραδοσιακές Τεχνικές
Μεθοδολογίες
Προσέγγιση Πρωτοτύπου
Προσέγγιση Κύκλου Ζωής Συστήματος
Ανάλυση Προβλήματος
Ανάλυση Συστήματος
ΣΧΕΔΙΑΣΗ ΣΥΣΤΗΜΑΤΟΣ
Εφαρμογή Συστήματος
I) Κατασκευή εναλλακτικών προτάσεων
II) Αξιολόγηση εναλλακτικών προτάσεων
III) Ανάλυση Κόστους / Ωφέλειας
Σχεδίαση Διαδικασιών (Process Design)
Σχεδίαση Δεδομένων (Αρχεία, Βάσεις Δεδομένων)
Σχεδίαση Περιφερειακών (Interface Design)
Προσανατολεισμένη στις Διαδικασίες (Process Driven)
Προσανατολισμένη στα Δεδομένα (Data Driven)
Ανάλυση Μεταβολών (Transorm Analysis)
Ανάλυση Συναλλαγών (Transaction Analysis)
JSP (Jackson's System Programming)

ΚΕΦΑΛΑΙΟ 3 - Computer Aided Software Engineering (CASE)

3.1 Τι είναι τα CASE Tools

Ένα εργαλείο CASE (CASE Tool) είναι ένα βασιζόμενο σε υπολογιστή προϊόν το οποίο έχει σαν στόχο την υποστήριξη μιας ή περισσοτέρων δραστηριοτήτων κατασκευής λογισμικού μέσα σε μία γενικότερη διαδικασία ανάπτυξης λογισμικού.

3.2 Η ιστορία των CASE Tools

Σήμερα τα πάντα κινούνται με γρηγορότερους ρυθμούς. Λόγω της αυξανόμενης ταχύτητας με την οποία αλλάζουν οι απαιτήσεις της αγοράς, τα νέα προϊόντα αντικαθιστούν τα παλιά πολύ πιο νωρίς, πράγμα που σημαίνει ότι η ανάπτυξη νέων προϊόντων πρέπει να κινείται με ταχύτερους ρυθμούς. Συνεπώς, οι γραμμές παραγωγής πρέπει επίσης να αναπτύσσονται γρηγορότερα.

Πολύ σημαντικό ρόλο σ' αυτή την ανάπτυξη παίζει η κατασκευή λογισμικού (software engineering), επειδή πολλές διαδικασίες παραγωγής είναι υποβοηθούμενες από τον υπολογιστή, πράγμα που σημαίνει ότι πρέπει να σχεδιαστεί λογισμικό για το σύστημα παραγωγής. Είναι εξαιρετικά σημαντικό η κατασκευή του λογισμικού να γίνει σωστά και γρήγορα.

Στο παρελθόν, τα συστήματα λογισμικού κατασκευάζονταν με «παραδοσιακές» τεχνικές οι οποίες βασίζονταν στη συγγραφή του κώδικα από τους προγραμματιστές. Οι μηχανικοί λογισμικού ήταν υποχρεωμένοι να σχεδιάζουν λογισμικό χωρίς την βοήθεια των υπολογιστών, προγραμματίζοντας κάθε βήμα με την σειρά. Ωστόσο, η προσέγγιση αυτή ήταν πολύ ακριβή και χρονοβόρα.

Για να επιταχυνθεί αυτή η διαδικασία, πρέπει να εντοπιστούν τα σημεία «συμφόρησης» κατά την κατασκευή συστημάτων λογισμικού. Αυτό δεν είναι εύκολο να γίνει, λόγω του αυξανόμενου ρόλου που παίζουν οι

υπολογιστές στην κοινωνία μας. Η τεχνολογία αναπτύσσεται όλο και περισσότερο, μέρα με τη μέρα, με αποτέλεσμα να παρουσιάζονται διαρκώς γρηγορότεροι και μεγαλύτεροι υπολογιστές. Το λογισμικό που τρέχει σ' αυτούς τους υπολογιστές μπορεί να είναι πιο ισχυρό, επειδή μπορούν να χειρίζονται περισσότερες πληροφορίες στον ίδιο χρόνο, οπότε αυξάνεται επίσης και το ποσό των απαιτούμενων δεδομένων. Η εύρεση των σωστών δεδομένων από το διαρκώς αυξανόμενο ποσό πληροφοριών γίνεται ολοένα και πιο δύσκολη, πράγμα το οποίο με τη σειρά του δυσχεραίνει τον εντοπισμό των σημείων συμφόρησης.

Για την επιτάχυνση της διαδικασίας κατασκευής συστημάτων λογισμικού, στην δεκαετία του 70 παρουσιάστηκε μία νέα προσέγγιση στην σχεδίαση λογισμικού, *η Υποβοηθούμενη από Υπολογιστή Κατασκευή Λογισμικού (Computer Aided Software Engineering, CASE).*

Ο όρος *CASE* χρησιμοποιείται για μία νέα γενιά εργαλείων τα οποία εφαρμόζουν αυστηρές αρχές στην ανάπτυξη και ανάλυση των προδιαγραφών λογισμικού. Με απλά λόγια οι υπολογιστές αναπτύσσουν λογισμικό για άλλους υπολογιστές γρήγορα χρησιμοποιώντας ειδικά εργαλεία. Όταν ενσωματώνεται σε ένα περιβάλλον «παράλληλης ανάπτυξης» (Concurrent Engineering) η διαδικασία αυτή μπορεί να λαμβάνει χώρα κατά τη διάρκεια που τρέχουν ήδη ορισμένα άλλα μέρη του αναπτυσσόμενου λογισμικού. Είναι περίπου σαν μία on line διαδικασία κατασκευής λογισμικού. Υπάρχουν πολλά προβλήματα μ' αυτά τα είδη συστημάτων, επειδή είναι ιδιαίτερα πολύπλοκα και ευπαθή και δεν συντηρούνται εύκολα.

Ορισμένα από τα εργαλεία τα οποία υπήρξαν πολύ προοδευτικά εκείνη την εποχή είναι εντελώς απαρχαιωμένα σήμερα, οπότε πρέπει να ανανεωθούν, πράγμα το οποίο μπορεί να αποτελέσει πρόβλημα επειδή η διαδικασία κατασκευής λογισμικού είναι προσαρμοσμένη πάνω σ' αυτά τα εργαλεία. Τα εργαλεία που αναπτύσσονται σήμερα αποτελούν την μετεξέλιξη προγενέστερων εργαλείων.

Τα πρώτα εργαλεία, τα οποία αναπτύχθηκαν στα μέσα της δεκαετίας του 70, είχαν κυρίως σαν στόχο την αυτοματοποίηση της παραγωγής και την

συντήρηση δομημένων διαγραμμάτων. Όταν αυτή η αυτοματοποίηση καλύπτει ολόκληρο τον κύκλο ζωής της ανάπτυξης λογισμικού, μιλάμε για «ενοποιημένη υποβοηθούμενη από τον υπολογιστή κατασκευή λογισμικού» (Integrated Computer Aided Software Engineering, I-CASE). Όταν καλύπτεται μόνο ένα συγκεκριμένο τμήμα του κύκλου ανάπτυξης, μιλάμε απλά για «υποβοηθούμενη από τον υπολογιστή κατασκευή λογισμικού» (Computer Aided Software Engineering, CASE).

Αργότερα παρουσιάστηκαν ενοποιημένα προϊόντα CASE (I-CASE). Αυτό ήταν ένα σημαντικό βήμα, επειδή τα προϊόντα I-CASE έχουν την δυνατότητα να χρησιμοποιούνται για την παραγωγή ολόκληρων εφαρμογών από τις προδιαγραφές σχεδίασης.

Πρόσφατα, τα εργαλεία CASE εισήλθαν σε μία τρίτη φάση : την παρουσίαση νέων μεθοδολογιών βασιζόμενων στις δυνατότητες των εργαλείων I-CASE. Αυτές οι νέες μεθοδολογίες χρησιμοποιούν τεχνικές ταχείας δημιουργίας πρωτοτύπων (Rapid Prototyping) για την γρηγορότερη ανάπτυξη των εφαρμογών με μικρότερο κόστος και υψηλότερη ποιότητα. Χρησιμοποιώντας αυτές τις τεχνικές μπορεί να κατασκευαστεί γρήγορα ένα πρωτότυπο, πράγμα που σημαίνει ότι το αναπτυσσόμενο σύστημα λογισμικού μπορεί να ελέγχεται πιο συχνά μεταξύ των διαφόρων φάσεων της ανάπτυξης, επειδή η δημιουργία ενός πρωτοτύπου δεν κοστίζει πολύ σε χρόνο. Έτσι, τα σφάλματα μπορούν να ανιχνεύονται και να διορθώνονται σχετικά νωρίς. Όσο νωρίτερα μπορεί να γίνεται αυτό, τόσο το καλύτερο για το αναπτυσσόμενο λογισμικό, επειδή η διόρθωση των ίδιων σφαλμάτων γίνεται δυσκολότερη και πιο ακριβή όταν το σύστημα φτάσει σ' ένα επόμενο επίπεδο ανάπτυξης. Συνεπώς, οι μεθοδολογίες ταχείας προτυποποίησης μπορούν να μειώσουν σημαντικά τον χρόνο και το κόστος.

Όπως αναφέραμε παραπάνω, είναι πλέον απαραίτητα νέα εργαλεία. Τα εργαλεία αυτά θα πρέπει να αυτοματοποιούν κάθε φάση του κύκλου ανάπτυξης και να συσχετίζουν στενότερα την ανάπτυξη της εφαρμογής με τις στρατηγικές διεργασίες της επιχείρησης για την οποία προορίζεται. Τα τελευταία χρόνια παρουσιάστηκαν διάφορα εργαλεία και συνεχώς

αναπτύσσονται νέα. Υπάρχουν πλέον τόσα πολλά εργαλεία, που εύκολα θα μπορούσε να μπερδευτεί κανείς.

3.3 Κατηγορίες CASE Tools

Για μία συνοπτική εξέταση των διαθέσιμων εργαλείων CASE θα επιχειρήσουμε να τα κατατάξουμε στις ακόλουθες κατηγορίες :

- ***Προϊόντα για την κατασκευή και υποστήριξη πληροφοριών.*** Αντιστοιχούν σε διαδικασίες οι οποίες προκύπτουν από τα στρατηγικά πλάνα της επιχείρησης και παρέχουν έναν αποθηκευτικό χώρο (repository) για την δημιουργία και συντήρηση των επιχειρηματικών μοντέλων, των μοντέλων δεδομένων και των μοντέλων διεργασιών.
- ***Προϊόντα για την δημιουργία δομημένων διαγραμμάτων.*** Αυτά προέρχονται από αρκετές μεθοδολογίες ανάπτυξης όπως οι Gane-Sarson και Jackson. Τα προϊόντα αυτά υποστηρίζουν τουλάχιστον τρεις βασικούς τύπους δομημένων διαγραμμάτων που χρησιμοποιούνται στην κατασκευή λογισμικού : διαγράμματα ροής δεδομένων (data flow), ροής ελέγχου (control flow) και ροής οντοτήτων (entity flow).
- ***Προϊόντα τα οποία παρέχουν βοηθήματα για την δομημένη ανάπτυξη.*** Τα προϊόντα αυτά είναι ιδιαίτερα κατάλληλα για τους αναλυτές συστημάτων, επειδή οι δομημένες διεργασίες μπορούν να αναλύονται γρηγορότερα και με μεγαλύτερη ακρίβεια.
- ***Προϊόντα παραγωγής κώδικα.*** Αυτά είναι προϊόντα τα οποία παράγουν κώδικα για έναν συγκεκριμένο στόχο, καθοριζόμενο από τον σχεδιαστή της εφαρμογής. Τα περισσότερα προϊόντα αυτής της κατηγορίας χρησιμοποιούν μία «γεννήτρια» COBOL – ένα εργαλείο το οποίο παράγει κώδικα σε μία συγκεκριμένη γλώσσα προγραμματισμού, από τις προδιαγραφές που έχει καθορίσει ο σχεδιαστής του συστήματος.

Ο πυρήνας ενός καλοσχεδιασμένου συστήματος CASE είναι ένας κεντρικός αποθηκευτικός χώρος (repository) ο οποίος χρησιμοποιείται σαν «βάση γνώσεων» (knowledge base) για την αποθήκευση πληροφοριών σχετικών με τον οργανισμό, την δομή του, το επιχειρησιακό μοντέλο, τις λειτουργίες, τις διαδικασίες, τα μοντέλα δεδομένων, κ.λ.π. Τα διαγράμματα και οι λεπτομέρειές τους αποθηκεύονται στο repository. Το repository συσσωρεύει σταθερά πληροφορίες σχετιζόμενες με την προετοιμασία, την ανάπτυξη, την σχεδίαση, την κατασκευή και την συντήρηση συστημάτων. Με δύο λόγια, το repository είναι ο πυρήνας ενός συστήματος CASE.

Έχουν χρησιμοποιηθεί δύο τύποι μηχανισμών σε λογισμικό CASE για την αποθήκευση πληροφοριών σχεδίασης :

1. Ένα ***λεξικό (dictionary)*** το οποίο περιέχει τα ονόματα και τις περιγραφές των στοιχείων δεδομένων, των διεργασιών, κ.λ.π.
2. Ένας ***αποθηκευτικός χώρος (repository)*** ο οποίος περιέχει τις πληροφορίες του λεξικού και μία πλήρη, κωδικοποιημένη αναπαράσταση των πλάνων, των μοντέλων και των σχεδίων, με εργαλεία ελέγχου (cross-checking), ανάλυσης συσχετισμών (correlation analysis) και επικύρωσης (validation).

Πριν χρησιμοποιηθούν εργαλεία CASE και σχεδίασης, θα πρέπει να ολοκληρωθούν τα ακόλουθα βήματα :

1. Κατάρτιση μιας μελέτης η οποία θα καθορίζει πως πρέπει να αλλάζουν οι βασικές επιχειρησιακές διεργασίες του οργανισμού ώστε να μεγιστοποιούνται οι ευκαιρίες και τα οφέλη που προκύπτουν από την γρήγορη αλλαγή της τεχνολογίας.
2. Εξέταση του τρόπου με τον οποίο θα πρέπει να αναδιαρθρωθεί ο οργανισμός ώστε να εκμεταλλεύεται τη νέα τεχνολογία.
3. Καθιέρωση ενός προγράμματος για την αντικατάσταση των παλαιών συστημάτων με νέες αποτελεσματικότερες τεχνολογίες.
4. Επιλογή μιας συνολικά ενοποιημένης αρχιτεκτονικής.

5. Επιλογή της μεθοδολογίας ανάπτυξης.
6. Επιλογή ενός εργαλείου CASE.
7. Καθιέρωση μιας πολιτικής η οποία θα ευνοεί την επαναχρησιμοποίηση.
8. Αναζήτηση ενός περιβάλλοντος το οποίο θα επιτρέπει την διασύνδεση ανοικτών συστημάτων και τη δυνατότητα μεταφοράς του λογισμικού σε όλα τα συστήματα του οργανισμού.
9. Δημιουργία συνδέσεων δικτύου μεταξύ του οργανισμού και των πιο σημαντικών συνεργατών του.
10. Εύρεση του τρόπου με τον οποίο θα μεταφερθεί όλη η τεχνογνωσία στα στελέχη του οργανισμού.
11. Καθορισμός των αλλαγών που πρέπει να γίνουν στη δομή της διοίκησης ώστε να καταστεί δυνατή η πλήρης εκμετάλλευση νέων, πρωτοποριακών συστημάτων αρχιτεκτονικών, μεθοδολογιών και εργαλείων.

Όταν ένας οργανισμός κάνει τα παραπάνω βήματα πριν από τον ανταγωνισμό του, αποκτά ένα σημαντικό πλεονέκτημα. Ένας οργανισμός θα πρέπει να είναι διαρκώς ενήμερος για τις εξελίξεις της τεχνολογίας των υπολογιστών, επειδή αυτό του επιτρέπει να διατηρεί το πλεονέκτημα έναντι των ανταγωνιστών του.

Ορισμένες από τις σημαντικές τάσεις στην ανάπτυξη νέων συστημάτων είναι :

1. MIPS χαμηλού κόστους – η τιμή των γρήγορων επεξεργαστών μειώνεται με διαρκώς αυξανόμενο ρυθμό, κάθε ημέρα.
2. Περιβάλλοντα κατανεμημένης επεξεργασίας – οι τελικοί χρήστες κινούνται προς μία πολυεπίπεδη αρχιτεκτονική υπολογιστών με στόχο την κατανεμημένη επεξεργασία.

3. Εργαλεία CASE και I-CASE – ενοποιημένα, βασιζόμενα σε repository εργαλεία κατασκευής λογισμικού τα οποία καθιστούν εφικτή την παραγωγή του κώδικα μιας εφαρμογής απευθείας από τις προδιαγραφές της.
4. Εργαλεία μηχανικής / αντίστροφης μηχανικής – εργαλεία τα οποία επιτρέπουν στους αναλυτές να μετατρέπουν τον χαμηλού επιπέδου κώδικα ορισμού δεδομένων και μη – δομημένων διεργασιών σε προτυποποιημένα στοιχεία δεδομένων και δομημένο κώδικα.
5. Νέες μεθοδολογίες ανάπτυξης – αποτελεσματικότερες τεχνικές οι οποίες καθιστούν εφικτή την ταχύτερη ανάπτυξη εφαρμογών σε πιο στενή συνεργασία με τους τελικούς χρήστες.
6. Ανάπτυξη προτύπων – πρότυπα τα οποία θα διέπουν τη μελλοντική εξέλιξη του υλικού (hardware) και του λογισμικού.

3.3 Διάφορα Είδη CASE Tools

3.3.1 E32 blue river (ένα εργαλείο ανάπτυξης λογισμικού για ενσωματωμένα συστήματα) (www.blue-river-software.com)

Το E32 ανήκει στη νέα γενιά 32-bit εργαλείων και είναι ειδικά προσαρμοσμένο για χρήση κατά την ανάπτυξη εφαρμογών για συστήματα micro controller. Συμβαδίζοντας με τις μοντέρνες τάσεις, αυτό το προϊόν υλοποιεί την αρχή "όλα σε ένα".

Το E32 είναι ένα περιβάλλον ανάπτυξης για όλο τον κύκλο ζωής των μοντέρνων εφαρμογών λογισμικού. Διαθέτει γραφικό σύστημα επικοινωνίας με τον χρήστη για "δομημένη ανάλυση και σχεδίαση" και βασίζεται στην μέθοδο SA/RT των "Ward & Mellor". Πρόκειται για ένα εργαλείο το οποίο μπορεί να χρησιμοποιηθεί ακόμη και από χρήστες με μικρή εμπειρία και γνώση των μεθόδων.

Τα προαναφερθέντα χαρακτηριστικά όχι μόνο μειώνουν σημαντικά τον απαιτούμενο χρόνο εκπαίδευσης, αλλά είναι επίσης ζωτικής σημασίας για την επικοινωνία μεταξύ των ατόμων που ασχολούνται με διαφορετικούς τομείς του ίδιου έργου.

Το γραφικό σύστημα επικοινωνίας με τον χρήστη για την φάση των προδιαγραφών και της σχεδίασης παρέχει αυξημένη ευελιξία και ένα μεγάλο πλεονέκτημα: ακόμη και οι μεγάλες λειτουργικές μονάδες και συναρτήσεις της εφαρμογής παραμένουν σαφείς και κατανοητές.

Ο μηχανισμός αυτόματης παραγωγής κώδικα μεταφράζει το μοντέλο προδιαγραφών σε κώδικα της ANSI C και τον καθιστά διαθέσιμο σε μορφή δομικού διαγράμματος (structogram).

Το περιβάλλον υλοποίησης (βασίζεται στην μέθοδο Nassi-Shneiderman) επιτρέπει τον χειρισμό του πηγαίου κώδικα με μορφή δομικού διαγράμματος και παρέχει την σύνδεση με ενσωματωμένα συστήματα.

Το E32 είναι διαθέσιμο για τις γλώσσες προγραμματισμού C/C++ και για τα λειτουργικά συστήματα MS-WIN NT 4.0 και μεταγενέστερες εκδόσεις, καθώς και για τα MS-WIN 95/98.

3.3.2 GeneXus (www.genexus.com)

Το GeneXus σας δίνει την δυνατότητα να αναπτύξετε εφαρμογές στρατηγικής σημασίας, αποκρύπτοντας την πολυπλοκότητά τους με την προσθήκη ευφυϊών χαρακτηριστικών. Είναι ένα προηγμένο εργαλείο το οποίο επιτρέπει την γρήγορη δημιουργία εφαρμογών βάσεων δεδομένων για μικρές ή μεγάλες επιχειρήσεις.

Τα χαρακτηριστικά και οι δυνατότητες του GeneXus είναι :

- *Η Απλότητα*

  Με το GeneXus είναι απλή υπόθεση η δημιουργία εφαρμογών βάσεων δεδομένων οποιουδήποτε μεγέθους, δεδομένου ότι το GeneXus διαχειρίζεται με έξυπνο τρόπο την πολυπλοκότητα της εφαρμογής.

- *Οι Έξυπνες δυνατότητες*

  Όταν είναι απαραίτητη η τροποποίηση της εφαρμογής, το GeneXus "αναμεταδίδει" αυτόματα τις αλλαγές που γίνονται, τόσο στην βάση δεδομένων, όσο και στα προγράμματα.

- *Η Διαισθητική χρήση*

  Ο δημιουργός κατασκευάζει το σύστημα με διαισθητικό τρόπο και κατόπιν το GeneXus το παράγει για οποιαδήποτε πλατφόρμα ή αρχιτεκτονική.

Ανόμοια με τα υπόλοιπα εργαλεία, το GeneXus έχει βαθιά γνώση των σχέσεων που υπάρχουν μεταξύ της βάσης δεδομένων και των προγραμμάτων εφαρμογών. Αυτό το μοναδικό χαρακτηριστικό διαφοροποιεί

το GeneXus από τα άλλα εργαλεία και του δίνει δυνατότητες όπως οι ακόλουθες:

- Δημιουργία κανονικοποιημένων βάσεων δεδομένων (πινάκων, ευρετηρίων, κ.λ.π.)
- Αυτόματη παραγωγή των προγραμμάτων
- Ανάλυση του τρόπου με τον οποίο επηρεάζουν οι αλλαγές την δομή της βάσης δεδομένων στα προγράμματα, και αυτόματη αναπαραγωγή τους

Με βάση όλα τα παραπάνω, το GeneXus μειώνει σημαντικά το κόστος της ανάπτυξης εφαρμογών και της συντήρησης υπολογιστικών συστημάτων. Το GeneXus επιτρέπει την ανάπτυξη εφαρμογών ανεξάρτητα από την πλατφόρμα, την βάση δεδομένων, την γλώσσα, την αρχιτεκτονική και το λειτουργικό σύστημα. Επιτυγχάνει χαμηλότερο κόστος συντήρησης και παρέχει δυνατότητα εύκολης μεταφοράς σε άλλες πλατφόρμες.

3.3.3 Resolution xCASE (www.xcase.com)

Η Resolution Ltd. ιδρύθηκε το 1988 και ειδικεύεται στην ανάπτυξη του xCase, μιας τεχνολογικά προηγμένης λύσης για την μοντελοποίηση και συντήρηση δεδομένων, ειδικά προσαρμοσμένης στις τρέχουσες ευρέως χρησιμοποιούμενες εφαρμογές βάσεων δεδομένων. Αναγνωρίζοντας από νωρίς την στρατηγική σπουδαιότητα της σωστής σχεδίασης και συντήρησης της βάσης δεδομένων, η Resolution αφιέρωσε τα δέκα τελευταία χρόνια στην ανάπτυξη του xCase με στόχο να βοηθήσει τους δημιουργούς εφαρμογών στην εκτέλεση οποιασδήποτε σχετιζόμενης με την βάση δεδομένων εργασίας, για ολόκληρο τον κύκλο ζωής της εφαρμογής.

Η αποστολή της Resolution είναι να παρέχει μία εκτενή και πλήρη, ενοποιημένη, εξαιρετικά προσαρμόσιμη και ταυτόχρονα εύχρηστη λύση για την σχεδίαση και συντήρηση βάσεων δεδομένων.

Σύμφωνα με τον Les Pinter, έναν ειδικό στον τομέα της ανάπτυξης, το 90% των προβλημάτων που συναντά οφείλονται στον φτωχό σχεδιασμό των βάσεων δεδομένων. Η απώλεια σε χρόνο και χρήμα - τόσο για τον πελάτη, όσο και για τον δημιουργό της εφαρμογής - είναι πολύ σημαντική, δεδομένου ότι οποιαδήποτε προβλήματα υπάρχουν στην σχεδίαση μεγεθύνονται και διαδίδονται σε όλη την έκταση της εφαρμογής. Η σωστή σχεδίαση και συντήρηση της βάσης δεδομένων, συμπεριλαμβανομένων των δομών και του σχετιζόμενου κώδικα, είναι κρίσιμης σημασίας.

Ο πελάτης ενδιαφέρεται για την αποθήκευση με ακρίβεια, την συντήρηση και την προσπέλαση των δεδομένων που είναι ζωτικής σημασίας για την αποτελεσματική διαχείριση της επιχείρησής του, και όχι για την απόκτηση αυτής καθ' εαυτής της εφαρμογής. Η ευθύνη του δημιουργού της εφαρμογής είναι να κατασκευάσει ένα πληροφοριακό σύστημα το οποίο θα δίνει στον πελάτη του αυτή την δυνατότητα. Με δεδομένη την μεγάλη διαθεσιμότητα αυτοματοποιημένων εργαλείων και υποδομών ανάπτυξης, ο δημιουργός μπορεί να επικεντρωθεί στην σχεδίαση και συντήρηση της βάσης δεδομένων - δύο στοιχεία τα οποία αποτελούν την ουσία της επιτυχημένης ανάπτυξης εφαρμογών βάσεων δεδομένων.

Το xCase, ένα εργαλείο CASE τρίτης γενιάς, είναι μία εύχρηστη, πλήρως ενοποιημένη λύση Upper CASE και Lower CASE, ειδικά προσαρμοσμένη στις μεγάλες εφαρμογές βάσεων δεδομένων που κυκλοφορούν σήμερα στην αγορά.

3.3.4 DA-C Project Manager της RistanCASE (www.ristancase.com)

Το Project Manager παρέχει τρεις δυνατότητες για την οργάνωση και παρουσίαση των δεδομένων που σχετίζονται με ένα έργο:

- *Explorer View (προβολή Εξερεύνησης).* Αντικαθιστά τον Explorer των Windows και παρέχει στον χρήστη επιπλέον πληροφορίες για τους καταλόγους που περιέχουν τα δεδομένα του έργου. Διευκολύνει την δημιουργία νέων έργων και βελτιώνει την παρακολούθηση πολύπλοκων συστημάτων αρχείων.

- *Folder View (προβολή Φακέλων).* Εμφανίζει μόνο τους φακέλους (καταλόγους) που σχετίζονται με ένα έργο, εξαλείφοντας την ανάγκη αναζήτησης και εντοπισμού τους σε ολόκληρο το σύστημα αρχείων. Από την άλλη, αφού παραχθεί το έργο, δεν υπάρχει λόγος να διέρχεστε από το σύστημα αρχείων για να αναζητήσετε τα σχετιζόμενα με το έργο αρχεία, επειδή τα πάντα είναι σε "απόσταση βολής" και συνοδεύονται από πολύ χρήσιμες πληροφορίες. Σ' αυτή την προβολή μπορείτε επίσης να ορίσετε ψευδώνυμα σαν μεγάλα ονόματα αρχείων ή περιγραφές αρχείων.
- *Logical View (Λογική προβολή).* Πρόκειται για την σημαντικότερη καινοτομία, επειδή διευκολύνει την καλύτερη οργάνωση υπαρχόντων έργων χωρίς να απαιτεί αλλαγές στην βασική δομή τους. Πολλά από τα έργα που συντηρούνται ή συνεχίζουν να αναπτύσσονται σήμερα προέρχονται από την εποχή όπου επικρατούσε ο περιορισμός "8.3" για τα μεγέθη των ονομάτων των αρχείων. Είναι δυνατό να αποφύγετε τα προβλήματα που αναφέραμε παραπάνω εφαρμόζοντας την Λογική προβολή (Logical View).

  Εν συντομία, η Λογική προβολή είναι ένα φίλτρο το οποίο τοποθετείται ανάμεσα στον χρήστη και στο πραγματικό σύστημα αρχείων, και μεταφράζει την εικόνα που έχει ο χρήστης για την ιδανική οργάνωση στην πραγματική, υπάρχουσα οργάνωση, η οποία δεν χρειάζεται να αλλάξει (ή δεν μπορεί να αλλάξει, λόγω περιορισμών του συστήματος ελέγχου εκδόσεων). Το φίλτρο αυτό μπορεί να είναι εξαρτώμενο από το εκάστοτε έργο, ή εξαρτώμενο από τον χρήστη.

3.3.5 AxiomSys & AxiomDsn της STG (www.stgcase.com)

Η STG διαθέτει εργαλεία δομημένης ανάλυσης (Structured Analysis tools) για ανάλυση συστημάτων και απαιτήσεων λογισμικού, καθώς και εργαλεία διαδικαστικών γλωσσών (Procedural Language tools) για την σχεδίαση λογισμικού.

Τα υψηλής απόδοσης προϊόντα της σειράς Axiom απολαμβάνουν ευρείας αποδοχής μεταξύ των μηχανικών λογισμικού, οι οποίοι αναγνωρίζουν την αξία της σχολαστικής ανάλυσης απαιτήσεων, της επικύρωσης και της τεκμηρίωσης.

Τα AxiomSys και AxiomDsn χρησιμοποιούνται σε τομείς της βιομηχανίας όπως η αεροναυτική, η άμυνα, οι επικοινωνίες, οι μεταφορές, ο βιομηχανικός έλεγχος και η κατασκευή ιατρικών οργάνων.

Το AxiomSys είναι μία πλήρης, φιλική προς τον χρήστη υλοποίηση της μεθόδου δομημένης ανάλυσης (Structured Analysis), συμπεριλαμβανομένων των επεκτάσεων Hatley-Phirbai Real-Time. Περιλαμβάνει επίσης τις δυνατότητες μοντελοποίησης Hatley-Phirbai Architecture. Το AxiomSys είναι ιδανικό για την μοντελοποίηση απαιτήσεων σε επίπεδο λογισμικού και συστημάτων. Το AxiomSys έκδοση 6.0 κυκλοφόρησε τον Απρίλιο του 1999 και είναι διαθέσιμο για MS-Windows NT, MS-Windows 95 και MS-Windows 98.

Το AxiomDsn είναι ένα πλήρες, φιλικό για τον χρήστη εργαλείο σχεδίασης λογισμικού (CASE) το οποίο υποστηρίζει διαδικαστικές γλώσσες προγραμματισμού όπως η C και η Ada. Επί του παρόντος, το AxiomDsn είναι διαθέσιμο μόνο για τα MS-Windows 3.1.

Στο Κεφάλαιο 4 αναλύεται το λογισμικό **Oracle Developer Suite 2000**, το οποίο περιλαμβάνει τον *Oracle Designer 2000*, για το σχεδιασμό και δημιουργία των βάσεων δεδομένων και τον *Oracle Developer 2000*, για τη δημιουργία των φορμών και των εκτυπωτικών αρχείων που αξιοποιούνται από τον τελικό χρήστη του Πληροφοριακού Συστήματος Διοίκησης

ΚΕΦΑΛΑΙΟ 4 – Oracle Developer Suite 2000

Στο κεφάλαιο αυτό θα αναφερθούμε στα πλεονεκτήματα, τις δυνατότητες και τα συστατικά του Oracle Developer Suite 2000.

4.1 Oracle Designer 2000

4.1.1 Περιγραφή του Oracle Designer 2000

Ο *Oracle Designer 2000* παρέχει ένα ολοκληρωμένο σετ εργαλείων για την ανάπτυξη client server εφαρμογών άμεσα και αποτελεσματικά. Ο *Oracle Designer 2000* περιλαμβάνεται στα εργαλεία ανάπτυξης λογισμικού *Oracle Developer Suite*.

Ο *Oracle Designer 2000* διαθέτει όλα τα χαρακτηριστικά που απαιτούνται για την ολοκληρωμένη ανάπτυξη των εφαρμογών, όπως:

- σχεδιασμό βάσεων δεδομένων και εφαρμογών που υποστηρίζουν λύσεις ηλεκτρονικού εμπορίου
- αξιοποίηση των υπαρχόντων μηχανογραφικών εφαρμογών με ανασχεδιασμό τους και
- δημιουργία σύνθετων εφαρμογών client/server

Τα παραπάνω χαρακτηριστικά τα υποστηρίζει το *Oracle Repository*, το οποίο αποθηκεύει και διαχειρίζεται τα δεδομένα των εφαρμογών. Το *Oracle Repository* επιτρέπει τη διαχείριση πολλαπλών εκδόσεων του λογισμικού υποστηρίζοντας με τον τρόπο αυτό την ολοκληρωμένη διαχείριση έργου και την ομαδική εργασία.

4.1.1.1 Μοντελοποίηση Oracle βάσεων δεδομένων

Ο *Oracle Designer 2000* διαθέτει τρεις βασικές μεθόδους μοντελοποίησης μιας βάσεως δεδομένων δημιουργώντας άμεσα τη φυσική βάση δεδομένων και τα αντικείμενά της. Ο σχεδιασμός μπορεί να βασιστεί σε

πραγματικές επιχειρησιακές διαδικασίες, σχεδιασμός εκ του μηδενός ή απεικόνιση του σχεδιασμού μιας υπάρχουσας μηχανογραφικής εφαρμογής.

Η γραφική απεικόνιση της βάσεως δεδομένων μέσω του *Design Editor* γίνεται άμεσα και εύκολα. Οι οργανισμοί πλέον μπορούν να εστιάσουν στις ανάγκες του πελάτη έχοντας μία υγιή βάση διαδικασιών την οποία μπορούν να τροποποιήσουν εάν απαιτηθεί. Παρέχεται, επομένως, η δυνατότητα σε μηχανογράφους και χρήστες να συνεργαστούν και να ανακαλύψουν τα στοιχεία των διαδικασιών τα οποία θα βελτιστοποιήσουν την ανταπόκριση της επιχείρησης στις ανάγκες των πελατών. Το τελικό μοντέλο είναι έτοιμο προς παραγωγή μέσω του *Server Generator.*

4.1.1.2 Δημιουργία βάσεων δεδομένων βασισμένη σε συγκεκριμένα μοντέλα

Ένα ισχυρότατο εργαλείο του *Design Editor* είναι ο *Server Generator*, ο οποίος χτίζει τη νέα βάση δεδομένων και ενσωματώνει υπάρχουσες βάσεις δεδομένων μέσα στην αποθήκη, δίνοντας τη δυνατότητα στις επιχειρήσεις να ανασχεδιάζουν τη βάση δεδομένων διαρκώς σύμφωνα με τις ανάγκες τους. Η συγκεκριμένη λειτουργία απεικονίζει τις σύγχρονες, πραγματικές επιχειρησιακές διεργασίες και διασφαλίζει ότι η εφαρμογή απεικονίζει και λειτουργεί σύμφωνα με τις απαιτήσεις τις επιχείρησης.

Ο Server Generator δημιουργεί βάσεις δεδομένων για κάθε πλατφόρμα από τα δεδομένα που βρίσκονται στην αποθήκη, όπως:

- Βάσεις δεδομένων Oracle ή τρίτων κατασκευαστών.
- Αντικείμενα που καθορίζονται στην αποθήκη άμεσα πάνω στη βάση δεδομένων (π.χ. πίνακες).
- Data Definition Language Scripts για τη δημιουργία φυσικής βάσεως δεδομένων ή για τα αντικείμενά της.
- Το PL/SQL Server API Interface που καλούν οι εφαρμογές για να διαχειριστούν τα δεδομένα των πινάκων.

4.1.1.3 Απεικόνιση του σχεδιασμού των υπαρχόντων βάσεων δεδομένων

Ο *Server Generator* απεικονίζει το σχέδιο μιας υπάρχουσας Oracle ή τρίτου κατασκευαστή βάσης δεδομένων και την ενσωματώνει στην αποθήκη.

Αυτή η διαδικασία μπορεί να χρησιμοποιηθεί για το σχεδιασμό των βάσεων δεδομένων των μηχανογραφικών εφαρμογών και την μετατροπή τους σε Oracle βάσεις δεδομένων, ή με τη χρήση του *Oracle Migration Manager* να αναβαθμιστεί μια προηγούμενη Oracle βάση δεδομένων. Ουσιαστικά ο *Server Generator* εξάγει τους ορισμούς της υπάρχουσας βάσης και δημιουργεί αντίστοιχους νέους ορισμούς στην αποθήκη.

4.1.1.4 Δημιουργία πολυ-λειτουργικών εφαρμογών

Ο *Server API* είναι ένα ισχυρότατο σύστημα επικοινωνίας με εφαρμογές που εκτελούν DML λειτουργίες σε πίνακες. Ο *Server API* μεταφράζει την εντολή της εφαρμογής προς τον αντίστοιχο πίνακα και αντίστοιχα επιστρέφει την απαιτούμενη πληροφόρηση στην αρχική εφαρμογή. Ο *Server API* είναι αναγκαίος στις επιχειρήσεις για να μπορέσουν να εγκαταστήσουν εφαρμογές σε υπολογιστές που δεν διαθέτουν τη σχετική λογική.

Ο *Server API* αποτελείται από:

- Τον Table API και
- Τον Module Component API

Το API μπορεί να εξειδικευτεί ώστε να εκτελεί ενέργειες σχετικές με τους επιχειρησιακούς κανόνες και τον σχεδιασμό των εφαρμογών.

4.1.1.5 Ανάλυση εξάρτησης

Με τη χρήση του *Dependency Manager* οι επιχειρήσεις μπορούν να εντοπίσουν και να αναλύσουν τις εξαρτήσεις ανάμεσα σε δομημένα και μη δομημένα δεδομένα. Υπάρχουν δύο είδη εξαρτήσεων:

- Οτιδήποτε χρησιμοποιεί ένα αντικείμενο, για παράδειγμα μία φόρμα χρησιμοποιεί έναν EMP πίνακα ή
- ότι χρησιμοποιείται από ένα αντικείμενο, για παράδειγμα ένα menu που χρησιμοποιείται από μία φόρμα

Η συγκεκριμένη δυνατότητα είναι απαραίτητη για τη διαχείριση των διαφόρων εκδόσεων των εφαρμογών. Σε ένα σύνθετο περιβάλλον είναι συνήθως δύσκολο να εντοπιστούν οι επιπτώσεις μιας αλλαγής στο σύνολο των εφαρμογών. Ο *Dependency Manager* διασφαλίζει ότι αυτές οι εξαρτήσεις και οι σχέσεις παραμένουν ενεργές.

4.1.2 Τα εργαλεία του Oracle Designer 2000

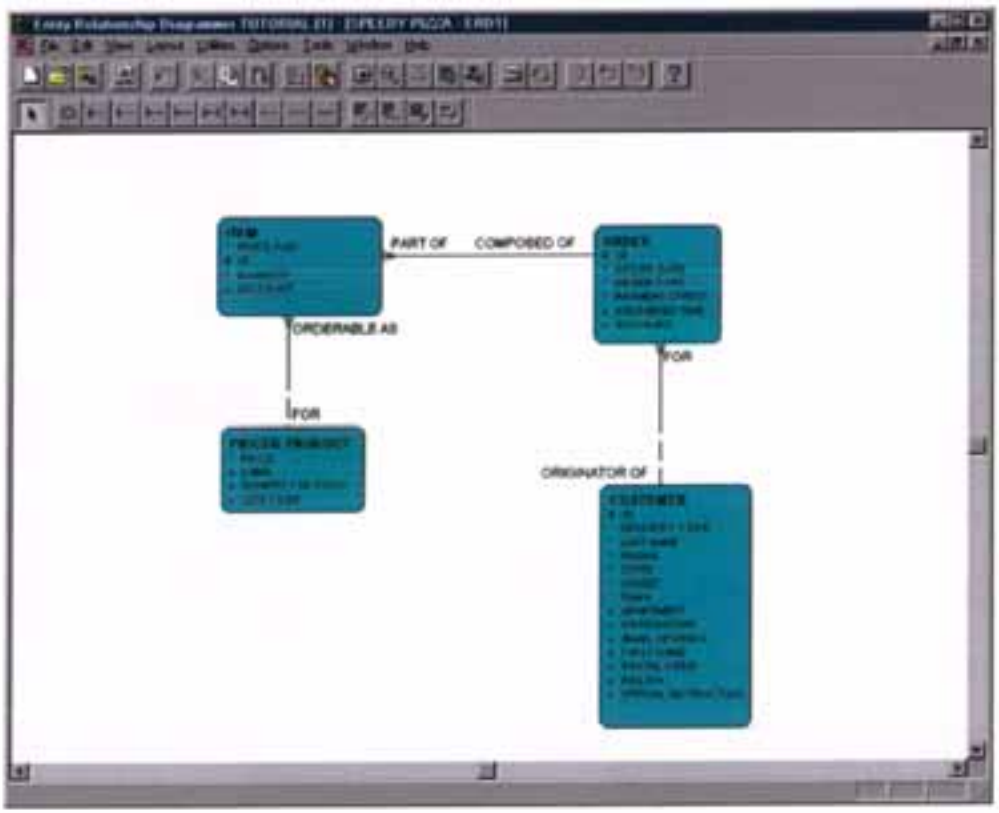
Η πρόσβαση σε όλα τα εργαλεία μοντελοποίησης σχεδιασμού και δημιουργίας εφαρμογών του *Oracle Designer 2000* καθώς και τα εργαλεία του *Oracle Repository* παρέχεται από την ακόλουθη, βασική οθόνη εκκίνησης του *Oracle Designer 2000*.

Οι λειτουργίες που υποστηρίζονται από τα εργαλεία του *Oracle Designer 2000* αναλύονται στη συνέχεια.

4.1.2.1 Μοντελοποίηση των απαιτήσεων του συστήματος

Ο *Process Modeler* απεικονίζει τις υπάρχουσες και τις μελλοντικές επιχειρησιακές διαδικασίες, συμπεριλαμβάνοντας μετρήσεις όπως ο χρόνος, το κόστος και η παραγωγικότητα. Επίσης, παρέχει ευέλικτη υποστήριξη για τις τεχνικές ανασχεδιασμού επιχειρησιακών διαδικασιών (Business Process Reengineering).

Ο *Entity Relationship Diagrammer* (Παράρτημα-Entity Relationship Diagram), απεικονίζει τα συγκεντρωμένα χαρακτηριστικά της εφαρμογής και παρέχει πρόσβαση στον *Database Design Transformer* για τη δημιουργία των αρχικών σχεδίων της βάσης δεδομένων. Χρησιμοποιεί τυποποιημένες τεχνολογίες πληροφοριών για να απεικονίσει τις οντότητες, τις ιδιότητες και τις σχέσεις μεταξύ των οντοτήτων. Ένα παράδειγμα οθόνης του *Entity Relationship Diagrammer* παρουσιάζεται στο ακόλουθο σχέδιο :

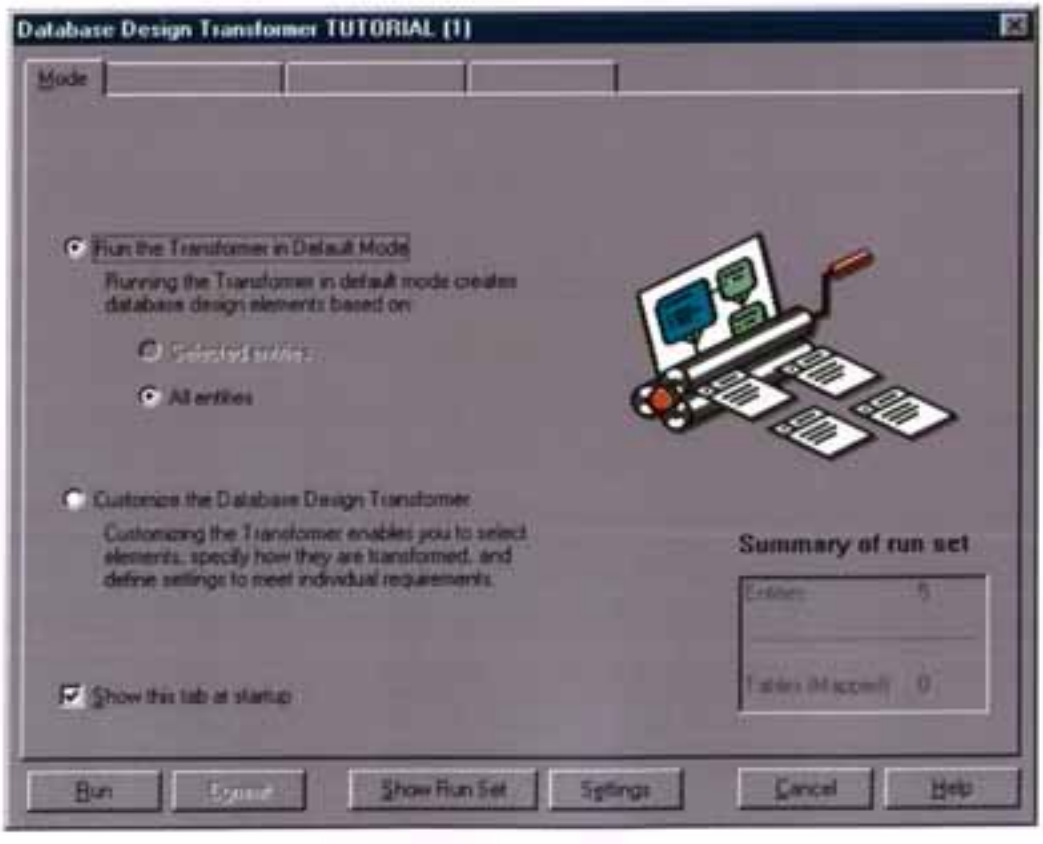


Ο *Function Hierarchy Diagrammer* απεικονίζει τις στοιχειώδεις επιχειρησιακές λειτουργίες που επεξηγούν τη χρήση των οντοτήτων και των ιδιοτήτων τους ως βασικό στοιχείο για τον σχεδιασμό εφαρμογών. Υποστηρίζει *Function Point Analysis* στη διαχείριση έργου και παρέχει πρόσβαση στο *Application Design Transformer* για τη δημιουργία του βασικού σχεδίου της εφαρμογής.

Ο *Dataflow Diagramer* παρουσιάζει τη ροή των δεδομένων στην επιχείρηση για να ελεγχθούν οι τυχόν υπάρχουσες εξαρτήσεις. Η λειτουργία αυτή επιτρέπει άμεσο έλεγχο του λειτουργικού μοντέλου και παρέχει πρόσβαση στον *Application Design Transformer*.

4.1.2.2 Μετασχηματισμός των προκαταρκτικών σχεδίων εφαρμογών

Ο *Database Design Transformer* δημιουργεί τα αρχικά σχέδια της βάσης δεδομένων (Παράρτημα – Data Schema Designs), που βασίζονται στα *Entity Relationship Diagrams* που είχαν δημιουργηθεί στην αρχική δήλωση των απαιτήσεων του συστήματος. Δημιουργεί για παράδειγμα πίνακες για την καταγραφή των οντοτήτων, στήλες για την αποθήκευση των ιδιοτήτων και δείκτες που υποστηρίζουν τα ξένα κλειδιά. Με τη μέθοδο αυτή ελέγχεται πλήρως ο σχεδιασμός των διαδικασιών καθώς αναπτύσσονται αναλυτικά τα μοντέλα ή τα βασικά μέρη αυτών. Η κεντρική οθόνη του *Database Design Transformer* είναι η παρακάτω :



Ο *Application Design Transformer* δημιουργεί τα αρχικά σχέδια της εφαρμογής που στηρίζονται στις λειτουργίες και τους ορισμούς που είχαν αρχικώς δηλωθεί σύμφωνα με τις απαιτήσεις του συστήματος. Δημιουργεί τις ενότητες και τα menu που στη συνέχεια εφαρμόζονται μέσω του *Oracle Forms* και του *Oracle Reports* με τη χρήση του *Design Editor*. Αυτή η ευέλικτη προσέγγιση υποστηρίζει λεπτομερώς τις ανάγκες του σχεδιασμού και ανταποκρίνεται άμεσα στις αλλαγές των απαιτήσεων.

4.1.2.3 Σχεδιασμός και δημιουργία βάσεων δεδομένων και εφαρμογών

Ο *Design Editor* είναι η βασική εφαρμογή για το σχεδιασμό βάσεων δεδομένων και εφαρμογών, ανάπτυξη, παραγωγή και απεικόνιση σχεδίων. Διαθέτει χαρακτηριστικά όπως:drag-and-drop λειτουργία, πολλαπλούς τύπους διαγραμμάτων, επιλογές χαρακτηριστικών, βιβλιοθήκες αντικειμένων και βοηθούς ανάπτυξης (wizards) που δημιουργούν μία ολοκληρωμένη πλατφόρμα εργασίας για τον ορισμό των βασικών προτύπων επικοινωνίας τα οποία χρησιμοποιούνται κατά την ανάπτυξη της εφαρμογής.

Αυτή η προσέγγιση του *Design Editor* απεικονίζει γραφικά τους επιχειρησιακούς κανόνες, υποστηρίζοντας πλήρως το σχεδιασμό και τη δημιουργία των δομών των βάσεων δεδομένων:

- Δημιουργία εφαρμογών για κάθε πλατφόρμα από τα δεδομένα που είναι αποθηκευμένα στην αποθήκη.
- Δημιουργία Oracle Forms, Oracle Reports, Web PL/SQL από ένα διάγραμμα.
- Δημιουργία λογικής για PL/SQLή Java Script.
- Δημιουργία και σχεδιασμός βάσεων δεδομένων (Oracle ή άλλου κατασκευαστή).
- Απεικόνιση του σχεδίου εφαρμογών με σκοπό την ανάλυση, τον ανασχεδιασμό και την εκ νέου δημιουργία.

4.1.2.4 Αποθήκευση και διαχείριση δεδομένων εφαρμογών

Το *Repository Administration Utility* είναι το κέντρο ελέγχου του *Oracle Repository* που χρησιμοποιείται για τη διαχείριση των δεδομένων που περιλαμβάνονται στην αποθήκη. Το γραφικό αυτό εργαλείο είναι ο μηχανισμός για την εγκατάσταση νέας αποθήκης του *Oracle Designer 2000* ή την αναβάθμιση μιας αποθήκης που προέρχεται από προηγούμενη έκδοση. Στο σημείο αυτό προστίθενται εξειδικευμένα αντικείμενα ανάπτυξης λογισμικού που συμβάλουν στην αντιμετώπιση των ειδικών επιχειρησιακών λειτουργιών.

Με τη χρήση του *Repository Administration Utility* για τη διαχείριση των περιεχομένων της αποθήκης αναδεικνύεται το πραγματικό multi-user περιβάλλον του *Oracle Repository*.

4.1.3 Τα πλεονεκτήματα του Oracle Designer 2000

4.1.3.1 Κεντρική διαχείριση των δεδομένων των εφαρμογών

Ο *Oracle Designer 2000* διαθέτει τη δυνατότητα πρόσβασης στα δεδομένα μίας κεντρικής αποθήκης, όπως για παράδειγμα τον ορισμό ενός πίνακα. Η δυνατότητα αυτή επιτρέπει την παραγωγική και αποτελεσματική ανάπτυξη εφαρμογών σε διαδικτυακό και σε client/server περιβάλλον.

4.1.3.2 Ακριβής ανάλυση των απαιτήσεων του συστήματος

Ο *Oracle Designer 2000* παρέχει ενσωματωμένα γραφικά εργαλεία μοντελοποίησης που απεικονίζουν τις απαιτήσεις των νέων ή των υπαρχόντων εφαρμογών με ακρίβεια, αμεσότητα και απλότητα.

Τα εργαλεία αυτά συνεργάζονται στενά με το *Oracle Repository*, επιτρέποντας την πολλαπλή πρόσβαση σε ένα περιβάλλον πολλών χρηστών όπου οι απαιτήσεις της εφαρμογής είναι διαθέσιμες άμεσα σε όλους τους χρήστες. Τα εργαλεία μοντελοποίησης του *Oracle Designer 2000* υποστηρίζουν τις κλασικές τεχνικές μοντελοποίησης για την ανάλυση των απαιτήσεων των σχεσιακών βάσεων δεδομένων.

4.1.3.3 Ισχυροί μετασχηματιστές βάσεων δεδομένων και εφαρμογών

Ο *Oracle Designer 2000* διαθέτει ισχυρότατους μετασχηματιστές για τη δημιουργία των αρχικών σχεδίων βάσεων δεδομένων και εφαρμογών που καλύπτουν τις προκαθορισμένες απαιτήσεις του συστήματος. Ο *Database Design Transformer* χτίζει τη βάση δεδομένων με πίνακες, κολόνες και δείκτες (Παράρτημα – Data Schema Designs), ενώ ο *Application Design Transformer* χτίζει τους πλήρεις ορισμούς των οθονών, των εκτυπώσεων και των menu.

Τα παραπάνω αρχικά σχέδια εφαρμογών και βάσεων δεδομένων μετά από την απαραίτητη επεξεργασία και ανάπτυξη οδηγούν στην ολοκληρωμένη εφαρμογή.

Οι μετασχηματιστές παρέχουν τα βασικά σχέδια και με τη χρήση τους οι αρχικές απαιτήσεις του συστήματος αναλύονται αποτελεσματικότερα αυξάνοντας την παραγωγικότητα και βελτιώνοντας την ποιότητα της τελικής εφαρμογής.

4.1.3.4 Γρήγορη δημιουργία βάσεων δεδομένων και εφαρμογών με τη χρήση γεννητριών

Οι γεννήτριες του *Oracle Designer 2000* δημιουργούν υψηλής ποιότητας εφαρμογές αξιοποιώντας τα δεδομένα που βρίσκονται αποθηκευμένα. Παρέχουν τη δυνατότητα για την ανάπτυξη όλων των βασικών συστατικών των εφαρμογών client server, για παράδειγμα την Oracle Database, την Oracle Forms, PL/SQL APIs και Web PL/SQL (Παράρτημα – SQL Code).

Οι γεννήτριες του *Oracle Designer 2000* είναι η ακριβής, παραγωγική και αποτελεσματική μέθοδος για τη δημιουργία των σωστών εφαρμογών στην κατάλληλη πλατφόρμα, καθώς βοηθούν σημαντικά στα αρχικά στάδια της ανάλυσης και του σχεδιασμού μιας ολοκληρωμένης εφαρμογής Πληροφοριακού Συστήματος Διοίκησης.

4.1.3.5 Αποτελεσματική απεικόνιση των υπαρχόντων μηχανογραφικών εφαρμογών

Ο *Oracle Designer 2000* απεικονίζει όλες τις σχεδιαστικές πληροφορίες των υπαρχόντων μηχανογραφικών εφαρμογών. Οι εφαρμογές που έχουν αναπτυχθεί με τη χρήση του *Oracle Forms* και όσες χρησιμοποιούνε διαφορετική βάση δεδομένων μπορούν επίσης να απεικονιστούν, συμπεριλαμβάνοντας τη λογική της εφαρμογής που έχει γραφτεί σε βασικό scripting περιβάλλον.

Η μετάβαση από ένα υπάρχον μηχανογραφικό περιβάλλον σε μία νέα βάση δεδομένων γίνεται απλή καθώς ο *Oracle Designer 2000* μπορεί να απεικονίσει το σχεδιασμό κάθε βάσης που είναι σύμφωνη με οποιοδήποτε ODBC. Παρέχει μία ολοκληρωμένη εικόνα όλων των διαθέσιμων επιχειρησιακών δεδομένων και δέχεται τα δεδομένα των μηχανογραφικών εφαρμογών τα οποία αργότερα τροποποιεί και αναπαράγει σύμφωνα με τις νέες απαιτήσεις.

4.1.3.6 Ολοκληρωμένος επαναληπτικός σχεδιασμός και ανάπτυξη εφαρμογών

Ο *Oracle Designer 2000* υποστηρίζει τις αλλαγές στη βασική εφαρμογή οι οποίες ενσωματώνονται στις νέες εκδόσεις που παράγονται. Με την έννοια αυτή ο *Oracle Designer 2000* υποστηρίζει τις πιο πρόσφατες δυνατότητες λειτουργιών του Oracle Forms και του *Oracle Reports*.

Η αναβάθμιση του σχεδίου των εφαρμογών είναι βασικό στοιχείο για κάθε ολοκληρωμένο παραγωγικό περιβάλλον σχεδιασμού και ανάλυσης νέων εφαρμογών ή μετάβασης από ένα υπάρχον μηχανογραφικό περιβάλλον.

4.1.3.7 Αποτελεσματικά εργαλεία για τη διαχείριση των αποθηκών

Για να διευκολύνει την γρήγορη, παραγωγική και αποτελεσματική σχεδίαση και ανάπτυξη εφαρμογών, ο *Oracle Designer 2000* χρησιμοποιεί το *Oracle Repository* που είναι ένα ολοκληρωμένο σετ εργαλείων για την αποθήκευση και διαχείριση των δεδομένων των εφαρμογών.

Το *Oracle Repository* υποστηρίζει κάθε φάση της ανάπτυξης του λογισμικού σε ένα ασφαλές και ελεγχόμενο περιβάλλον. Ο *Oracle Designer 2000* επιτρέπει εύκολη αλλαγή και αναβάθμιση των εφαρμογών καθώς το *Oracle Repository* συγχρονίζει αυτόματα την εφαρμογή και τα δεδομένα της.

Η πρόσβαση των χρηστών στην αποθήκη είναι απόλυτα ελεγχόμενη καθώς καταγράφονται όλες οι αλλαγές και η ιστορικότητα στην ανάπτυξη των

εφαρμογών. Επίσης, παρέχεται η δυνατότητα της ανάλυσης των σχέσεων μεταξύ των δομημένων και των μη δομημένων δεδομένων καθώς ελέγχονται τα κοινά δεδομένα και η πολυπλοκότητα των εφαρμογών, μειώνοντας τον κίνδυνο των διπλών εγγραφών και διασφαλίζοντας την ενιαία πηγή δεδομένων.

Το *Oracle Repository* διαθέτει ανοιχτό API και υποστηρίζει την ανάπτυξη εξειδικευμένου λογισμικού που ανταποκρίνεται σε συγκεκριμένες δραστηριότητες και επιχειρησιακές ανάγκες.

4.2 Oracle Developer 2000

4.2.1 Δυνατότητες και Πλεονεκτήματα

Το Developer 2000 είναι ένα εργαλείο ανάπτυξης Web εφαρμογών πολλαπλών βαθμίδων (multi-tier Web applications) και εφαρμογών αρχιτεκτονικής client/server οι οποίες μπορούν να τρέχουν σε όλες τις "μεγάλες" πλατφόρμες υπολογιστών.

Περιλαμβάνει ένα ενοποιημένο σύνολο εργαλείων δημιουργίας :

- (builders) φορμών,
- αναφορών,
- γραφικών,
- ερωτημάτων και
- διαδικασιών.

Οι δυνατότητες αυτών των συστατικών επιτρέπουν τη δημιουργία εφαρμογών από ορισμούς βάσεων δεδομένων (database definitions) χωρίς να χρειάζεται η γραφή κώδικα.

Το Developer 2000 παρέχει τις ακόλουθες δυνατότητες και πλεονεκτήματα:

- Βελτιώσεις στην παραγωγικότητα
- Διανομή εφαρμογών μέσω του Web
- Επέκταση σε ευρύτερη κλίμακα
- Δημιουργία "ανοικτών" εφαρμογών

4.2.1.1 Παραγωγικότητα

Το Developer 2000 χρησιμοποιεί μεθόδους ταχείας σχεδίασης εφαρμογών (Rapid Application Design, RAD), αντικειμενοστραφείς ((object oriented) τεχνικές, ενοποιημένη αρχιτεκτονική client/server και δυνατότητες

βασιζόμενης στον υπολογιστή εκπαίδευσης για την βελτίωση της παραγωγικότητας. Η δημιουργία εφαρμογών απλοποιείται με την χρήση ειδικών οδηγών (wizards) και μέσω της κληρονομικότητας (inheritance) των συστατικών με δυνατότητα επαναχρησιμοποίησης. Το Developer 2000 μπορεί επίσης να χρησιμοποιείται για την ανάπτυξη εφαρμογών απευθείας από μοντέλα δημιουργημένα με το Oracle Designer/2000 (CASE).

Το Developer 2000 παρέχει ένα περιβάλλον κατάλληλο για την ανάπτυξη έργων οποιουδήποτε μεγέθους. Το νέο ενοποιημένο Project Builder διαχειρίζεται όλα τα συστατικά της εφαρμογής, καθώς και τα εξωτερικά συστατικά (π.χ. περιεχόμενο σε μορφή πολυμέσων) και επιτρέπει το άνοιγμα κάθε συστατικού με το κατάλληλο εργαλείο. Το Project Builder αυξάνει την παραγωγικότητα, διευκολύνοντας την ολοκληρωμένη ανάπτυξη των εφαρμογών. Το Project Builder μπορεί να χρησιμοποιηθεί για να "πακεταριστούν" οι εφαρμογές σ' ένα .TAR ή .ZIP αρχείο, καθώς και για να δημιουργηθούν scripts εγκατάστασης για χρήση με το Oracle Installer. Για την διαχείριση της εφαρμογής παρέχονται λειτουργίες οι οποίες βασίζονται σε δημοφιλή πακέτα ελέγχου πηγαίου κώδικα και εκδόσεων.

4.2.1.2 Διανομή Εφαρμογών μέσω του Web

Το Developer 2000 επιτρέπει στους δημιουργούς εφαρμογών να διανέμουν νέες και υπάρχουσες εφαρμογές εκμεταλλευόμενοι τις τεχνολογίες του Web, είτε σ' ένα δίκτυο intranet είτε στο ίδιο το Internet. Το Developer 2000 εκμεταλλεύεται την ευκολία πρόσβασης στο Web, προάγοντάς το από έναν στατικό μηχανισμό δημοσίευσης πληροφοριών σ' ένα περιβάλλον ικανό να υποστηρίξει πολύπλοκες εφαρμογές με "δυναμική" συμπεριφορά.

Οι φόρμες και οι οθόνες που δημιουργούνται (με εργαλεία όπως τα Form/Graphics Builder) για το Web έχουν το ίδιο πλούσιο και "διαλογικό" περιβάλλον όπως και αυτές που παράγονται για client/server εφαρμογές, σε συνδυασμό με την επεκτασιμότητα που είναι απαραίτητη σε οποιαδήποτε κρίσιμης σημασίας, βασιζόμενη σε συναλλαγές εφαρμογή. Οι εφαρμογές που αναπτύσσονται για το Web παρέχουν τον βαθμό της ποιότητας και της

ακρίβειας που απαιτεί ένας οργανισμός για την δημοσίευση των πληροφοριών του.

4.2.1.3 Επέκταση σε Ευρύτερη Κλίμακα

Η επέκταση σε ευρύτερη κλίμακα είναι ένα χαρακτηριστικό εγγενές στην αρχιτεκτονική πολλαπλών βαθμίδων (multi-tiered architecture) του Developer 2000. Είναι προαπαιτούμενο όσον αφορά στην υποστήριξη για λειτουργίες server όπως οι εντολές διαχείρισης δεδομένων (array DML), οι cursors βάσεων δεδομένων, ο συσχετισμός μεταβλητών (variable bind), τα σημεία αποθήκευσης (save points) και η επιστροφή αποτελεσμάτων (result sets).

Είναι επίσης καθοριστικό για τον διαχωρισμό (partitioning) των αντικειμένων, πράγμα το οποίο μειώνει σημαντικά την κυκλοφορία του δικτύου μεταξύ του server και των client συστημάτων. Η ευέλικτη αρχιτεκτονική πολλαπλών βαθμίδων του Developer 2000 υποστηρίζει server εφαρμογών (application servers) της Oracle 7, διασφαλίζοντας την δυνατότητα επέκτασης από μεμονωμένους σταθμούς εργασίας σε επίπεδο οργανισμού.

4.2.1.4 Ανοικτές Εφαρμογές

Η επικοινωνία του Developer 2000 με άλλες εφαρμογές και εργαλεία βασίζεται σε πρότυπα και υλοποιείται μέσω των μηχανισμών OCX/ActiveX OLE (Object Linking and Embedding, διασύνδεση και ενσωμάτωση αντικειμένων) και της DDE (Dynamic Data Exchange, δυναμική ανταλλαγή δεδομένων). Η υποστήριξη για μία ποικιλία μορφών πολυμέσων συμπληρώνεται από μία ανοικτή αρχιτεκτονική, δίνοντας στους δημιουργούς εφαρμογών την ευελιξία να επεκτείνουν τις εφαρμογές του Developer 2000 και να ενσωματώσουν άλλα συστατικά σ' αυτές.

Το Developer 2000 παρέχει τη δυνατότητα ενοποίησης των εφαρμογών με βάσεις δεδομένων της Oracle, καθώς και "διαφανή"

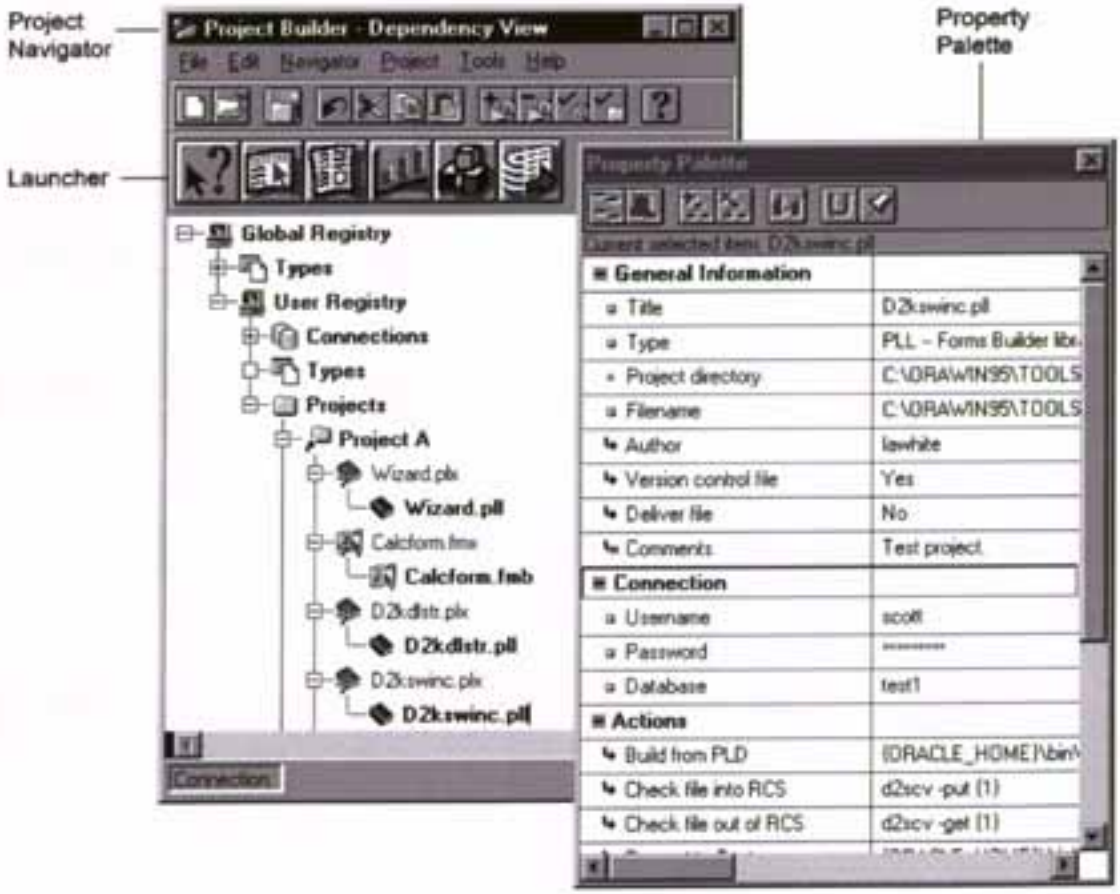
πρόσβαση σε όλες τις "μεγάλες" βάσεις δεδομένων, συμπεριλαμβανομένων των Rdb, SQL Server, Informix, Sybase και DB/2. Η πρόσβαση σε βάσεις δεδομένων παρέχεται μέσω προγραμμάτων οδήγησης ODBC, καθώς επίσης και μέσω ειδικών πυλών της Oracle (Oracle Gateways).

Είναι διαθέσιμα πολλά συστήματα (interfaces) για την επικοινωνία με προϊόντα τρίτων κατασκευαστών, από τα μέλη της πρωτοβουλίας Open Tools Initiative. Τα συστήματα αυτά περιλαμβάνουν εργαλεία για την διαχείριση της διαμόρφωσης, εργαλεία CASE και μοντελοποίησης, μηχανισμούς ελέγχου ροής (workflow engines), μηχανισμούς παρακολούθησης των συναλλαγών (transaction processing monitors) και πολλά άλλα.

4.2.2 Εισαγωγή στο Project Builder

Με την έκδοση 2.0. το Developer 2000 παρουσίασε για πρώτη φορά το Project Builder, ένα ισχυρό εργαλείο το οποίο δίνει τη δυνατότητα στους χρήστες να απλοποιήσουν τις εργασίες διαχείρισης της διαμόρφωσης του λογισμικού, έτσι ώστε να μπορούν να επικεντρωθούν στον πρωτεύοντα στόχο : <u>τον σχεδιασμό και τον έλεγχο της εφαρμογής</u>.

Η κεντρική οθόνη του Project Builder είναι η ακόλουθη :



Επειδή το Project Builder είναι ένα νέο και ισχυρό εργαλείο θα το αναλύσουμε περισσότερο απ' τα υπόλοιπα συστατικά του Developer 2000.

4.2.2.1 Σύνοψη

Το Project Builder βοηθά στην οργάνωση και συντήρηση των πολλών διαφορετικών αρχείων που απαρτίζουν μία εφαρμογή. Μέσα από ένα έργο μπορεί να παρακολουθηθεί οτιδήποτε - από το υλικό τεκμηρίωσης της εφαρμογής που έχει δημιουργηθεί χρησιμοποιώντας οποιοδήποτε επεξεργαστή κειμένων, μέχρι τις φόρμες, τις αναφορές και τα γραφήματα που περιέχουν, και τις διάφορες ρουτίνες που καλούνται. Το Project Builder μπορεί να παρέχει ένα μεμονωμένο "σημείο επαφής" με την εφαρμογή , ενοποιώντας όλα τα εργαλεία, τα αρχεία και τις εργασίες που περιλαμβάνει το έργο ανάπτυξης μιας εφαρμογής.

4.2.2.2 Η Ορολογία του Project Builder

Το Project Builder βασίζεται σε δύο έννοιες: την έννοια του έργου (project) και την έννοια του υπο-έργου (subproject).

- Τα ***έργα*** είναι συλλογές δεικτών (pointers) προς λειτουργικές μονάδες και αρχεία τα οποία είναι μέρος της εφαρμογής.
- Τα ***υπό-έργα*** είναι έργα τα οποία περιέχονται μέσα σε άλλα έργα, παρέχοντας ένα επιπλέον επίπεδο οργάνωσης.

Επιπρόσθετα με τα έργα και υπό-έργα, οι ακόλουθοι όροι είναι επίσης ιδιαίτερα σημαντικοί για την κατανόηση της λειτουργίας του Project Builder:

- ***Τύποι (Types).*** Ένας τύπος είναι η βάση κάθε καταχώρισης και ελέγχει τις ενέργειες που είναι διαθέσιμες στο Project Builder. Οι τύποι του Project Builder αναγνωρίζουν τους τύπους αρχείων με τους οποίους συσχετίζονται κυρίως μέσω των προεπιλεγμένων επεκτάσεων - για παράδειγμα, η επέκταση.TXT για τα αρχεία απλού κειμένου. Το Project Builder περιλαμβάνει προκαθορισμένους τύπους για πολλούς κοινά χρησιμοποιούμενους τύπους αρχείων, όπως έγγραφα φορμών (FMB), αρχεία κειμένου, και αρχεία πηγαίου κώδικα της C.

- ***Καταχωρίσεις (Entries).*** Τα συστατικά που απαρτίζουν ένα έργο αναφέρονται με τον όρο "καταχωρίσεις". Μία καταχώριση είναι απλώς μία περιγραφή ενός αρχείου το οποίο είναι μέρος ενός έργου. Κάθε καταχώριση περιγράφεται πλήρως στην σχετιζόμενη Παλέτα Ιδιοτήτων (Property Palette), η οποία παρουσιάζει τον τύπο της καταχώρισης, την θέση της στο σύστημα αρχείων, το μέγεθός της και την ημερομηνία τελευταίας τροποποίησής της. Ορίζονται επίσης οι ενέργειες και οι μακροεντολές (δείτε παρακάτω) για την καταχώριση. Μία καταχώριση δεν είναι ένα αρχείο - αντίθετα, είναι μία περιγραφή του αρχείου. Συνεπώς, όταν διαγράφετε μία καταχώριση από ένα έργο, ο Project Builder καταλαβαίνει ότι το αρχείο στο οποίο "δείχνει" αυτή η καταχώριση δεν αποτελεί πλέον μέρος του έργου. Το ίδιο το αρχείο δεν διαγράφεται.
- ***Ενέργειες (Actions).*** Οι ενέργειες είναι ακολουθίες εντολών (command strings) οι οποίες εφαρμόζονται σε αρχεία ενός συγκεκριμένου τύπου - για παράδειγμα, η ενέργεια Edit για ένα αρχείο κειμένου μπορεί να είναι η ακολουθία εντολών που καλεί το Notepad ή το WordPad για την επεξεργασία του.
- ***Μακροεντολές (Macros)***. Οι μακροεντολές είναι μεταβλητές οι οποίες μπορούν να χρησιμοποιηθούν για την τροποποίηση των ενεργειών. Μία μακροεντολή μπορεί να είναι είτε μία σταθερά (constant), είτε μία απλή έκφραση (η οποία, με την σειρά της, μπορεί να περιέχει σταθερές και / ή άλλες εκφράσεις). Οι πληροφορίες που περιέχονται στην μακροεντολή εισάγονται κατόπιν στην ενέργεια, έτσι ώστε να μπορεί να συνδεθεί αυτόματα. Ακριβώς όπως χρησιμοποιούνται οι μεταβλητές περιβάλλοντος σε script ή batch αρχεία για να προσαρμόζουν τις ενέργειες που εκτελούν χωρίς να απαιτείται η τροποποίηση των ίδιων των αρχείων, κατά τον ίδιο τρόπο μπορούν να χρησιμοποιηθούν οι μακροεντολές για να προσαρμόζονται οι ενέργειες χωρίς να πρέπει να τροποποιούνται οι ίδιες ενέργειες.

- ***Γενικό (Global) Registry.*** Το Γενικό Registry περιέχει τους προκαθορισμένους τύπους του Project Builder.
- ***Registry Χρήστη (User Registry).*** Κάθε χρήστης έχει το δικό του Registry, στο οποίο μπορεί να ορίζει νέους τύπους, να επαναπροσδιορίζει υπάρχοντες τύπους και να τροποποιεί ή να δημιουργεί ενέργειες ή μακροεντολές.
- ***Registry Έργου (Project registry).*** Το Registry έργου είναι ένα αρχείο το οποίο περιέχει τις πληροφορίες που απαιτούνται για την παρακολούθηση ενός έργου, συμπεριλαμβανομένων των δεικτών προς τις λειτουργικές μονάδες που περιέχει το έργο, των προεπιλεγμένων εντολών σύνδεσης και ενός δείκτη προς τον "βασικό" (home) κατάλογο του έργου.
- ***Στοιχεία Έργου (Project items).*** Τα στοιχεία έργου είναι όλα τα αντικείμενα που περιέχονται σ' ένα έργο, όπως οι τύποι, οι ενέργειες, οι μακροεντολές και οι λειτουργικές μονάδες.

Το Project Builder παρέχει τρία εργαλεία για την διαχείριση των στοιχείων που απαρτίζουν ένα έργο:

- Το ***Project Navigator*** διαθέτει ένα περιβάλλον το οποίο διευκολύνει την εξέταση των έργων και μπορεί να χρησιμοποιηθεί για την εξέταση των λειτουργικών μονάδων μιας εφαρμογής. Παρέχει επίσης τη δυνατότητα εκκίνησης των εργαλείων επεξεργασίας απευθείας από το Project Navigator.
- Η ***Παλέτα Ιδιοτήτων (Property Palette)*** επιτρέπει την εξέταση και την τροποποίηση των ιδιοτήτων των επιλεγμένων στοιχείων.
- Η ***γραμμή εργαλείων Launcher*** παρέχει έναν ακόμη τρόπο για την προσπέλαση των εργαλείων ανάπτυξης. Δίνει επίσης την δυνατότητα πρόσθεσης κουμπιών στην γραμμή εργαλείων Launcher και της συσχέτισής τους με άλλα εργαλεία.

4.2.2.3 Πώς Επηρεάζει το Project Builder τους Υπάρχοντες Ρόλους Ανάπτυξης

Θα πρέπει να οριστούν συγκεκριμένοι ρόλοι έτσι ώστε ο κύκλος ανάπτυξης μιας εφαρμογής να ολοκληρωθεί χωρίς προβλήματα.

Ορισμένοι ρόλοι, όπως :

- ο ρόλος του επόπτη έργου (project manager),
- του επόπτη ανάπτυξης (development manager) και
- του επόπτη ομάδας (team leader)

είναι κοινοί στις περισσότερες ομάδες ανάπτυξης και δεν απαιτούν ειδικό ορισμό.

Ωστόσο, το Project Builder εισάγει έναν νέο ρόλο - αυτόν του *διαχειριστή έργου (project administrator).*

Ένας διαχειριστής έργου είναι το άτομο που έχει την ευθύνη για την δημιουργία έργων και τα καθιστά διαθέσιμα στους προγραμματιστές. Ο διαχειριστής έργου διατηρεί το Γενικό Registry και το τροποποιεί όπως απαιτείται, εξάγοντας τις αλλαγές που γίνονται στα μέλη της ομάδας εργασίας. Το άτομο αυτό μπορεί επίσης να εξάγει τις πληροφορίες έργου σε διαφορετικά περιβάλλοντα (π.χ. περιβάλλοντα ελέγχου / δοκιμών), ή σε άλλες πλατφόρμες, για την ανάπτυξη της εφαρμογής σε πολλαπλές πλατφόρμες.

Η εργασία που εκτελεί ο διαχειριστής έργου μπορεί να επηρεάζει τους ρόλους των ακόλουθων μελών της ομάδας:

- Προγραμματιστές
- Υπεύθυνος για τον έλεγχο του πηγαίου κώδικα
- Δοκιμαστές (QA)
- Υπεύθυνος για την τελική έκδοση (Releaser)

Φυσικά, τα καθήκοντα κάθε μέλους μπορεί να είναι διαφορετικά σε κάθε ομάδα εργασίας. Ένα μέλος της ομάδας εργασίας μπορεί επίσης να αναλάβει περισσότερους από έναν ρόλους - για παράδειγμα, ένας επόπτης ομάδας μπορεί επίσης να είναι ο διαχειριστής έργου, ή ένας προγραμματιστής μπορεί να είναι υπεύθυνος για τον έλεγχο του πηγαίου κώδικα.

4.2.2.4 Τα Πλεονεκτήματα του Project Builder

(α) Συσχετισμός Λειτουργικών Μονάδων με μία Εφαρμογή

Μπορούν να συσχετιστούν όλες οι λειτουργικές μονάδες μιας εφαρμογής με την ίδια την εφαρμογή, αν αυτές προστεθούν απλώς στο ίδιο έργο. Αυτό επιτρέπει την παρακολούθηση μιας μεγάλης εφαρμογής σαν μία μεμονωμένη οντότητα, τον καθορισμό των εξαρτήσεων μεταξύ των λειτουργικών μονάδων, κ.λ.π.

(β) Αυτοματοποίηση Ενεργειών με Βάση τους Τύπους Αρχείων

Το Project Builder διαθέτει μία εκτενή λίστα τύπων στους οποίους έχουν ανατεθεί προεπιλεγμένες ενέργειες, όπως οι :

- Open (άνοιγμα),
- Edit (επεξεργασία), ή
- Print (εκτύπωση)

Όταν γίνει η επιλογή μιας λειτουργικής μονάδας από τον χρήστη και κάνει κλικ με το δεξί πλήκτρο του ποντικιού, εμφανίζεται ένα μενού συντόμευσης με τις ενέργειες οι οποίες σχετίζονται μ' αυτό τον συγκεκριμένο τύπο. Οι ενέργειες που περιλαμβάνονται σ' έναν ορισμό τύπο εφαρμόζονται σε όλες τις λειτουργικές μονάδες αυτού του τύπου, σε όλη την έκταση του

έργου. Μπορεί επίσης να τροποποιήσει τις ενέργειες, ή να προσθέσει νέες ενέργειες.

Οι ενέργειες είναι απλώς ακολουθίες εντολών. Ένα πλεονέκτημα που παρέχει η χρήση των πραγματικών ακολουθιών εντολών για τον ορισμό των ενεργειών (εκτός φυσικά από την απλότητα) είναι ότι μία ενέργεια μπορεί να σχετίζεται εννοιολογικά με αρκετούς διαφορετικούς τύπους.

Για παράδειγμα, η επεξεργασία ενός εγγράφου του Word απαιτεί ένα διαφορετικό εργαλείο απ' ότι η επεξεργασία ενός εγγράφου σε μορφή απλού κειμένου - ωστόσο, σε εννοιολογικό επίπεδο, οι δύο αυτές ενέργειες επεξεργασίας είναι πολύ παρόμοιες. Το Project Builder μπορεί να συσχετίσει την ίδια ενέργεια Edit με δύο διαφορετικούς τύπους χωρίς να υπάρξει οποιαδήποτε σύγχυση, επειδή η ενέργεια Edit έχει διαφορετική ακολουθία εντολών για κάθε τύπο. Κατ' αυτό τον τρόπο, μία ενέργεια εκτελεί τις κατάλληλες εντολές ανεξάρτητα από τον τύπο της λειτουργικής μονάδας με την οποία δουλεύει ο χρήστης.

(γ) Δημιουργία Εξαρτήσεων Μεταξύ Λειτουργικών Μονάδων

Το να γνωρίζει ο χρήστης ποιες λειτουργικές μονάδες εξαρτώνται από ποιες άλλες είναι απαραίτητο για να εξακριβώσει σε ποιες περιπτώσεις απαιτείται εκ νέου μεταγλώττιση όλων ή συγκεκριμένων λειτουργικών μονάδων, όταν κάνει οποιεσδήποτε αλλαγές. Η γνώση αυτή είναι επίσης το κλειδί για την διαχείριση της επίδρασης των αλλαγών - για παράδειγμα, όταν αλλάζει μία βιβλιοθήκη, ποιες φόρμες καθίστανται παρωχημένες.

Το Project Builder περιλαμβάνει τις εξαρτήσεις για τύπους λειτουργικών μονάδων στους ορισμούς των τύπων τους. Συνεπώς, μπορεί να αναγνωρίζει τις εξαρτήσεις μεταξύ υπαρχουσών λειτουργικών μονάδων ενός έργου. Επειδή μπορεί επίσης να παρακολουθεί τις τροποποιήσεις που γίνονται στις λειτουργικές μονάδες, εκτελεί εκ νέου την μεταγλώττιση αυτόματα για τις τροποποιημένες λειτουργικές μονάδες και για τις λειτουργικές μονάδες που εξαρτώνται από αυτές.

Στην πραγματικότητα, το Project Builder μπορεί να αναγνωρίζει εξαρτήσεις οι οποίες δεν υπάρχουν ακόμη στο έργο και να δημιουργεί ενδείκτες γι' αυτές. Αυτοί οι ενδείκτες αποκαλούνται "έμμεσα" ή "υπονοούμενα" στοιχεία (implied items). Για παράδειγμα, αν θεωρηθεί ότι το έργο περιέχει ένα .FMB αρχείο, αυτό ορίζεται με τον τύπο "Form Builder document" του Project Builder. Το εκτελέσιμο αρχείο του Form Builder ("Form Builder executable", ένα.FMX αρχείο) μπορεί να μην υπάρχει - μπορεί να μην έχει παραχθεί ακόμη. Αλλά το Project Builder θεωρεί ότι η ύπαρξη αυτού του .FMX αρχείου υπονοείται, λόγω της ύπαρξης του .FMB αρχείου, και δημιουργεί ένα έμμεσο ή υπονοούμενο στοιχείο σαν ενδείκτη γι' αυτό.

Για να καθορίσει την ύπαρξη ενός υπονοούμενου στοιχείου, το Project Builder συσχετίζει την τιμή της ιδιότητας Deliverable Type για κάθε καθοριζόμενο τύπο με τα στοιχεία εισόδου (input items ή source) που απαιτούνται για την ενέργεια Build From <τύπος>, για κάθε καθοριζόμενο τύπο. Στο παραπάνω παράδειγμα, η ιδιότητα Deliverable Type για τον τύπο "Form Builder document" ορίζεται σαν "Form Builder executable" ή .FMX. Η ενέργεια Build From <τύπος> που ορίζεται για ένα εκτελέσιμο του Form Builder είναι "Build From FMB". Αυτό σημαίνει ότι τα .FMB αρχεία είναι τα στοιχεία εισόδου για την δημιουργία .FMX αρχείων - αντίστροφα, τα .FMX αρχεία είναι το τελικό προϊόν των .FMB αρχείων.

Η αλυσίδα των υπονοούμενων στοιχείων μπορεί να αποτελείται από πολλαπλά αρχεία. Για παράδειγμα, θεωρηθεί ότι ο χρήστης προσθέσει ένα αρχείο πηγαίου κώδικα C σ' ένα αρχείο βιβλιοθήκης, τότε το Project Builder προσθέτει λειτουργικές μονάδες οποιωνδήποτε άλλων τύπων είναι απαραίτητες για να δημιουργηθεί μία πλήρης διαδρομή ενεργειών Build From <τύπος> από τον έναν τύπο αρχείου έως τον άλλο (όμοια μ' ένα object αρχείο).

Αν και το Project Builder ανιχνεύει τις εξαρτήσεις που υπάρχουν μόνο μεταξύ μεταγλωττιζόμενων λειτουργικών μονάδων και των παραγόμενων εκτελέσιμων, μπορεί ο χρήστης να ορίσει εξαρτήσεις "χειροκίνητα", προσθέτοντας λειτουργικές μονάδες σ' ένα έργο, κάτω από το στοιχείο που

εξαρτάται από αυτές. Για παράδειγμα, εάν ένα .FMB αρχείο εξαρτάται από μία βιβλιοθήκη PL/SQL μπορεί να προσθέσει το αρχείο της βιβλιοθήκης (ένα.PLL αρχείο) στο έργο, κάτω από το .FMB αρχείο, και το Project Builder θα αναγνωρίσει αυτή την εξάρτηση.

Για να εμφανίσει ο χρήστης τις εξαρτήσεις που περιέχονται στο έργο του επιλέγει Navigator->Dependency View. Η εντολή αυτή εμφανίζει τις λειτουργικές μονάδες στο Project Navigator με την σειρά με την οποία εξαρτώνται η μία από την άλλη. Οι κόμβοι έργων (project nodes) εμφανίζονται στο υψηλότερο επίπεδο της ιεραρχίας και ακολουθούνται από κόμβους "προορισμού" (target nodes) οι οποίοι με την σειρά του ακολουθούνται από τα "στοιχεία εισόδου".

(δ) Αντιστοίχηση Προκαθορισμένων Εντολών Σύνδεσης σε Λειτουργικές Μονάδες

Με το Project Builder ο χρήστης μπορεί να ορίσει όλες τις συχνά χρησιμοποιούμενες εντολές σύνδεσης (connection strings) και να αποθηκεύσει τους ορισμούς τους στον κόμβο Connections (συνδέσεις) στο Registry χρήστη (user registry). Μπορεί κατόπιν να αντιστοιχίσει μία σύνδεση σε μία λειτουργική μονάδα σέρνοντας την σύνδεση από τον κόμβο Connections και αποθέτοντάς την πάνω στην λειτουργική μονάδα. Όταν θέλει να τροποποιήσει αυτή την λειτουργική μονάδα - π.χ. μία φόρμα - μπορεί να επιλέξει την φόρμα από το Project Navigator και κατόπιν να επιλέξει Edit από το μενού συντόμευσης. Το Project Builder ανοίγει αυτόματα το Form Builder και ο χρήστης συνδέεται στην βάση δεδομένων.

(ε) Επιλογή των Λειτουργικών Μονάδων που θα Συμπεριληφθούν στο Τελικό Πακέτο Εγκατάστασης της Εφαρμογής

Το Project Builder βοηθάει τον χρήστη να καθορίσει και να παρακολουθεί τις λειτουργικές μονάδες που θα συμπεριληφθούν στο τελικό πακέτο εγκατάστασης (π.χ..EXE αρχεία,.DLL αρχεία και.HLP αρχεία). Για να

μαρκάρει ένα αρχείο έτσι ώστε να συμπεριληφθεί στο τελικό πακέτο εγκατάστασης, ορίζει την ιδιότητά του Deliver File σε Yes. Όταν είναι έτοιμος να δημιουργήσει ένα πακέτο εγκατάστασης (install package), η ενέργεια Deliver "πακετάρει" όλες τις λειτουργικές μονάδες για τις οποίες η ιδιότητα Deliver File είναι ορισμένη σε Yes σε μία μεμονωμένη μονάδα.

Σημείωση: Μπορεί να ορίσει την ιδιότητα Deliver File για έναν τύπο ή για μεμονωμένα στοιχεία του έργου.

(στ) Κοινή Χρήση και Μεταφορά των Αρχείων του Registry για Έργα και Υπό-Έργα

Το Project Builder επιτρέπει στον χρήστη να εξάγει (export) τις πληροφορίες για ένα έργο σε άλλα μέλη της ομάδας εργασίας και σε άλλες πλατφόρμες υπολογιστών. Οι πληροφορίες για τους τύπους, τις ενέργειες, τις μακροεντολές και τα αρχεία του Registry του έργου - συμπεριλαμβανομένων όλων των προσαρμογών που έχει κάνει - μπορούν να αποθηκευτούν σε ένα αρχείο απλού κειμένου το οποίο μπορεί κατόπιν να εισάγει (import) σε άλλα περιβάλλοντα και άλλες πλατφόρμες. Η δυνατότητα αυτή επιτρέπει την ανάπτυξη και τον έλεγχο μιας εφαρμογής σε πολλαπλές πλατφόρμες.

(ζ) Χρήση Άλλων Εργαλείων του Developer 2000 και Εργαλείων Τρίτων

Ο χρήστης μπορεί να προσπελάσει άλλα εργαλεία από το περιβάλλον του Project Builder με αρκετούς τρόπους:

- ***Οι ενέργειες (actions),*** τις οποίες προσπελάζει επιλέγοντας μία λειτουργική μονάδα από το Project Navigator και κάνοντας κλικ με το δεξιό πλήκτρο του ποντικιού. Εμφανίζεται ένα μενού συντόμευσης το οποίο παρουσιάζει όλες τις ενέργειες που σχετίζονται με το επιλεγμένο στοιχείο- οι ενέργειες αυτές καλούν τα οποιαδήποτε εργαλεία καθορίζονται στις ακολουθίες εντολών τους (command strings). Μπορεί επίσης να κάνει διπλό κλικ σε μία καταχώριση στο Project Navigator για να καλέσει την προεπιλεγμένη ενέργειά της.

- ***Οι ενέργειες Compile και Deliver***, καθώς και οι ενέργειες ελέγχου πηγαίου κώδικα (source control), μπορούν να επιλεχθούν από το μενού File->Administration. Οι ενέργειες αυτές εκκινούν τα εργαλεία με τα οποία ο χρήστης τα έχει συσχετίσει.
- ***Η γραμμή εργαλείων Launcher***, η οποία εκκινεί πολλά συστατικά του Developer 2000 όπως τα Form Builder, Report Builder και Graphics Builder.

Ο χρήστης μπορεί επίσης να προσθέσει δικά του κουμπιά στην γραμμή εργαλείων Launcher και να τα συσχετίσει με άλλα εργαλεία τρίτων κατασκευαστών τα οποία θέλει να χρησιμοποιήσει.

(η) Χρήση Πακέτων Ελέγχου Πηγαίου Κώδικα με το Developer 2000

Το Developer 2000 μπορεί να συνεργάζεται με τα ακόλουθα πακέτα ελέγχου πηγαίου κώδικα:

- PVCS της Intersolv
- Clear case της Pure Atria
- Versions, το συστατικό ελέγχου πηγαίου κώδικα του Star Team, της Star Base

Ο χρήστης μπορεί επίσης να χρησιμοποιεί άλλα εργαλεία ελέγχου πηγαίου κώδικα, προσαρμόζοντας τις ενέργειες ελέγχου πηγαίου κώδικα (source control actions) που παρέχει το Project Builder ώστε να "δείχνουν" σ' αυτά.

4.2.2.5 Δημιουργία ενός Έργου

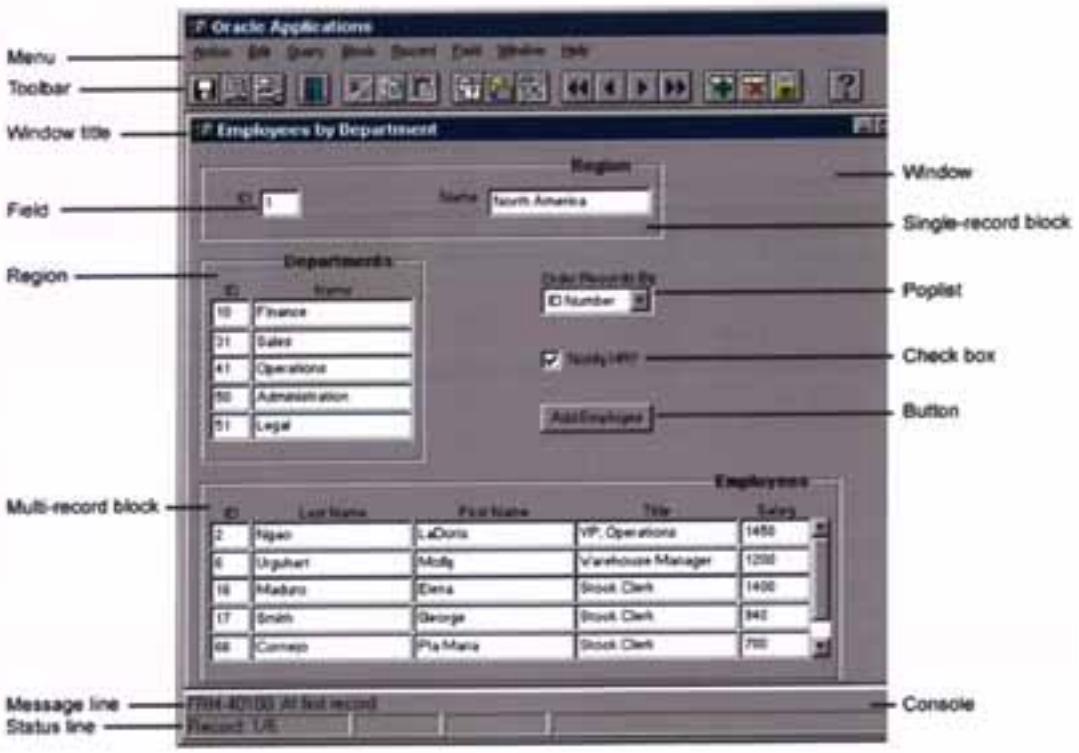
Για την δημιουργία ενός έργου με το Project Builder θα πρέπει να ακολουθούνται από το χρήστη τα παρακάτω βήματα:

- Δημιουργία του έργου με τον οδηγό Project Wizard.
- Προσθήκη στοιχειών (τύπων, ενεργειών, μακροεντολών και αρχείων) στο νέο έργο.
- Παραγωγή (build) των τελικών αρχείων του έργου και παράδοση της εφαρμογής.

4.2.3 Εισαγωγή στο Form Builder

4.2.3.1 Σύνοψη

Το Form Builder είναι ένα εργαλείο ανάπτυξης για την δημιουργία εφαρμογών οι οποίες παρέχουν στους χρήστες πρόσβαση σε πληροφορίες αποθηκευμένες σε μία βάση δεδομένων. Οι λειτουργικές μονάδες που δημιουργεί το Form Builder μπορούν να τρέχουν αυτόνομες, και μπορούν επίσης να περιλαμβάνουν ενσωματωμένες αναφορές ή γραφικά. Ένα τυπικό παράθυρο του Form Builder παρουσιάζεται στο παρακάτω σχήμα :



4.2.3.2 Βασικές Έννοιες

Όταν δημιουργηθεί μία εφαρμογή με το Form Builder ο χρήστης δουλεύει με τρεις τύπους λειτουργικών μονάδων:

- φόρμες (forms),
- μενού (menus) και

- βιβλιοθήκες (libraries).

Μία λειτουργική μονάδα φόρμας (form module) είναι μία συλλογή αντικειμένων και δεδομένων με τα οποία αλληλεπιδρούν οι χρήστες για να κάνουν αλλαγές στους πίνακες μιας βάσης δεδομένων. Αποτελούν τον κορμό μιας εφαρμογής του Form Builder και μπορούν να περιέχουν οποιοδήποτε από τα συστατικά που περιλαμβάνει ένα σύστημα επικοινωνίας με τον χρήστη (user interface) μαζί με τους ορισμούς της "πηγής" δεδομένων και την συμπεριφορά της.

Μία λειτουργική μονάδα μενού (menu module) αποτελείται από μενού εντολών και τον κώδικα για την χρήση τους. Οι χρήστες επιλέγουν στοιχεία (εντολές) από τα μενού για να εκτελέσουν τις λειτουργίες της εφαρμογής.

Μία λειτουργική μονάδα βιβλιοθήκης (library module) είναι μία συλλογή ρουτινών οι οποίες τρέχουν στην πλευρά του client συστήματος και μπορούν να είναι κοινόχρηστες για πολλαπλές λειτουργικές μονάδες και εφαρμογές.

Το *Object Navigator* παρουσιάζει με ιεραρχικό τρόπο όλα τα αντικείμενα της εφαρμογής.

Ο *οδηγός Data Block Wizard* επιτρέπει στον χρήστη να δημιουργήσει ή να τροποποιήσει μπλοκ δεδομένων για χρήση με την εφαρμογή του. Μπορεί να χρησιμοποιήσει τον οδηγό Data Block Wizard για την τροποποίηση ενός υπάρχοντος μπλοκ δεδομένων, ακόμη κι αν αυτό δεν δημιουργήθηκε αρχικά με τον οδηγό.

Οι *Βιβλιοθήκες Αντικειμένων (Object Libraries)* παρέχουν μία εύκολη μέθοδο για την επαναχρησιμοποίηση αντικειμένων και την καθιέρωση / επιβολή προτύπων σε επίπεδο οργανισμού.

Ο οδηγός Layout Wizard επιτρέπει στον χρήστη να καθορίσει γρήγορα και εύκολα την διάταξη των στοιχείων ενός μπλοκ δεδομένων. Ο οδηγός εμφανίζει τα στοιχεία σ' ένα πλαίσιο, πάνω σε μία "επιφάνεια σχεδίασης" (canvas), και μπορεί να τα διευθετήσει με αρκετά διαφορετικά στυλ διάταξης. Αφού διευθετήσει τα στοιχεία ο οδηγός ο χρήστης μπορεί να προσαρμόσει μόνος του την διάταξή τους, ανάλογα με τις ανάγκες του. Μπορεί επίσης να

χρησιμοποιήσει τον οδηγό Layout Wizard για την τροποποίηση μιας υπάρχουσας ομάδας στοιχείων, ακόμη κι αν αυτή δεν δημιουργήθηκε αρχικά με τον οδηγό.

Η *Παλέτα Ιδιοτήτων (Property Palette)* επιτρέπει στον χρήστη να καθορίσει τις ιδιότητες των αντικειμένων που δημιούργησε σε λειτουργικές μονάδες φορμών και μενού. Όταν επιλέγει ένα αντικείμενο σ' έναν συντάκτη ή στο Object Navigator, η Παλέτα Ιδιοτήτων παρουσιάζει τις ιδιότητές του. Όταν θέλει να συγκρίνει τις ιδιότητες δύο διαφορετικών αντικειμένων, μπορεί να ανοίγει επιπλέον Παλέτες Ιδιοτήτων.

Ο *ενσωματωμένος Συντάκτης PL/SQL* επιτρέπει στον χρήστη να γράψει κώδικα PL/SQL μέσα από το Form Builder. Παρέχει επίσης ένα ειδικό περιβάλλον για την επεξεργασία και την αποσφαλμάτωση των διαδικασιών και συναρτήσεων τόσο στην πλευρά του client συστήματος, όσο και στην πλευρά του server.

4.2.3.3 Διαδικασία

Για να κατασκευάσει μία εφαρμογή ο χρήστης με το Form Builder θα πρέπει να ακολουθήσει τα παρακάτω βήματα:

- Δημιουργία λειτουργικών μονάδων με το Object Navigator.
- Δημιουργία μπλοκ δεδομένων και στοιχείων είτε "χειροκίνητα", είτε με τον οδηγό Data Block Wizard.
- Χρήση αντικειμένων από την Βιβλιοθήκη Αντικειμένων και εφαρμογή των απαιτούμενων προτύπων για το σύστημα επικοινωνίας της εφαρμογής με τον χρήστη.
- Βελτίωση της διάταξης των στοιχείων με τον οδηγό Layout Wizard, ή με τον Συντάκτη Διάταξης (Layout Editor).
- Ορισμός ιδιοτήτων με την Παλέτα Ιδιοτήτων.
- Προσθήκη κώδικα με τον Συντάκτη PL/SQL.

- Δοκιμή της εφαρμογής χρησιμοποιώντας το συστατικό Forms Runtime και το ενσωματωμένο εργαλείο αποσφαλμάτωση (Debugging).

4.2.4 Εισαγωγή στο Graphics Builder

4.2.4.1 Σύνοψη

Το Graphics Builder επιτρέπει στον χρήστη να παράγει οθόνες οι οποίες παρουσιάζουν τα δεδομένα με "οπτικό" τρόπο και αλληλεπιδρούν με τον χρήστη. Μία οθόνη είναι ένας συνδυασμός συμπεριφορών και δεδομένων, τα οποία ενθυλακώνονται σε μία αυτόνομη εφαρμογή. Αυτό δίνει τη δυνατότητα στο χρήστη να επαναχρησιμοποιεί τις οθόνες και να τις ενσωματώνει σε φόρμες ή αναφορές.

4.2.4.2 Βασικές Έννοιες

Μία εφαρμογή δημιουργημένη με το Graphics Builder αποκαλείται "οθόνη" (display). Μία οθόνη περιέχει όλα τα συστατικά που χρησιμοποιούνται στην εφαρμογή, συμπεριλαμβανομένων των ορισμών για την "πηγή" δεδομένων, των οπτικών στοιχείων και της συμπεριφοράς. Κάθε οθόνη περιέχει μία διάταξη (layout) βάσει της οποίας εμφανίζονται τα διάφορα γραφικά αντικείμενά της. Η διάταξη αποτελείται από ένα ή περισσότερα επίπεδα τα οποία περιέχουν τα μεμονωμένα οπτικά στοιχεία. Κατά τον χρόνο εκτέλεσης μπορούν να αποκρύψουν, να επανεμφανίσουν, ή να αναδιατάξουν τα επίπεδα για να παρουσιαστούν διαφορετικές απόψεις στον χρήστη. Όλα τα άλλα συστατικά της εφαρμογής, όπως τα ερωτήματα επιλογής δεδομένων και οι δομές PL/SQL, είναι επίσης μέρος της οθόνης.

Υπάρχουν πάνω από πενήντα προκαθορισμένα κοινά στυλ γραφημάτων, και μπορούν επίσης να χρησιμοποιηθούν πλήρη σετ εργαλείων σχεδίασης για την δημιουργία εξειδικευμένων οθονών, όπως π.χ. χάρτες με χαρακτηριστικά διαλογικότητας.

Δίνετε επίσης η δυνατότητα εισαγωγής (import) εικόνων από μία ευρεία ποικιλία μορφών αρχείων. Μπορεί να χρησιμοποιηθεί ο συντάκτης

Layout Editor για την δημιουργία και τροποποίηση των γραφικών αντικειμένων.

Το Object Navigator παρουσιάζει με ιεραρχικό τρόπο όλα τα αντικείμενα μιας οθόνης.

Ο οδηγός Chart Wizard δίνει την δυνατότητα δημιουργίας γρήγορων και εύκολων γραφημάτων, επιλέγοντας μία πηγή δεδομένων και δημιουργώντας ένα ερώτημα για την επιλογή των δεδομένων που θα απεικονίζει το γράφημα. Μπορεί επίσης να χρησιμοποιηθεί ο οδηγός Chart Wizard για την τροποποίηση υπαρχουσών οθονών, ακόμη κι αν αυτές δεν δημιουργήθηκαν αρχικά με τον οδηγό.

Η Παλέτα Ιδιοτήτων (Property Palette) επιτρέπει τον ορισμό ιδιοτήτων των αντικειμένων που περιλαμβάνονται σε μία οθόνη. Όταν επιλεγεί ένα αντικείμενο στο Object Navigator, η Παλέτα Ιδιοτήτων εμφανίζει τις ιδιότητες αυτού του αντικειμένου. Αν χρειασθεί να συγκριθούν οι ιδιότητες διαφορετικών αντικειμένων, μπορούν να ανοίξουν επιπλέον Παλέτες Ιδιοτήτων.

Ο ενσωματωμένος Συντάκτης PL/SQL επιτρέπει τη γραφή κώδικα PL/SQL μέσα από το περιβάλλον του Graphics Builder. Παρέχει ένα γραφικό σύστημα επικοινωνίας με τον χρήστη για την τροποποίηση και αποσφαλμάτωση των διαδικασιών και συναρτήσεων τόσο στην πλευρά του client συστήματος, όσο και στην πλευρά του server.

4.2.4.3 Διαδικασία

Για να δημιουργηθεί ένα αντικείμενο γραφήματος χρησιμοποιώντας ένα από τα προκαθορισμένα στυλ γραφημάτων του Graphics Builder, πρέπει να ακολουθηθούν τα παρακάτω βήματα:

- Χρησιμοποίηση του οδηγού Chart Wizard για να δημιουργηθεί ένα αντικείμενο γραφήματος, επιλογή μιας πηγής δεδομένων και δημιουργία ενός ερωτήματος για την επιλογή των δεδομένων που θα απεικονίζει το γράφημα.

- Επιλογή ενός τύπου γραφήματος από το παράθυρο Chart Properties.
- Τροποποίηση του γραφήματος και των ιδιοτήτων των στοιχείων του γραφήματος, όπως απαιτείται.
- Πρόσθεση κωδικού χρησιμοποιώντας τον Συντάκτη PL/SQL.
- Δοκιμή της εφαρμογής με χρήση του Report Builder ή του συστατικού Form Builder Runtime και του ενσωματωμένου εργαλείου αποσφαλμάτωσης.
- Συνδυασμός της οθόνης με μία λειτουργική μονάδα φόρμας.

4.2.5 Εισαγωγή στο Report Builder

4.2.5.1 Σύνοψη

Το Report Builder είναι ένα εργαλείο το οποίο δίνει την δυνατότητα δημιουργίας ποιοτικών αναφορών σε περιβάλλοντα εφαρμογών client / server ή εφαρμογών Web. Οι εφαρμογές μπορούν να είναι αυτόνομες, ή ενσωματωμένες σε φόρμες και οθόνες δημιουργημένες με το Graphics Builder.

4.2.5.2 Βασικές Έννοιες

Μία αναφορά (report) είναι μία συλλογή αντικειμένων τα οποία ορίζουν τα δεδομένα που εμφανίζει η αναφορά, την διάταξή της, και το σύστημα επικοινωνίας της με τον χρήστη (interface) κατά τον χρόνο εκτέλεσης της εφαρμογής.

Για την γρήγορη και εύκολη δημιουργία αναφορών μπορεί να χρησιμοποιηθεί ο οδηγός δημιουργίας αναφορών, Report Wizard. Ο οδηγός απλοποιεί την δημιουργία αναφορών, κατευθύνοντας όλα τα βήματα της διαδικασίας στην επιλογή τύπου αναφοράς (report type), στον ορισμό ενός μοντέλου δεδομένων (data model) και στην παρουσίαση των δεδομένων με συγκεκριμένη διάταξη. Μπορεί επίσης να καλεί τον οδηγό Report Wizard για να τροποποιήσει μια υπάρχουσα αναφορά, ανεξάρτητα από το εάν αυτή έχει δημιουργηθεί με τον οδηγό ή όχι.

Αφού ολοκληρωθεί η διαδικασία του οδηγού, η αναφορά εμφανίζεται στο παράθυρο Live Previewer του Report Editor. Το Live Previewer είναι ένα WYSIWYG περιβάλλον επεξεργασίας, μέσα από το οποίο μπορούν να γίνουν απλές αλλαγές στην διάταξη της αναφοράς.

Για πολυπλοκότερες ενέργειες επεξεργασίας μπορούν να χρησιμοποιηθούν οι προβολές (views) Data Model, Layout Model και Parameter Form του Report Editor (μπορεί να επιλεγεί οποιαδήποτε προβολή από το μενού View του Report Editor). Για την παραγωγή HTML ή

PDF αρχείων από τις αναφορές σας θα χρησιμοποιηθεί ο οδηγός Web Wizard.

Για την παραγωγή αναφορών σε "απομακρυσμένους" servers, χρησιμοποιείτε το Reports Multi-tier Server (το οποίο από εδώ και στο εξής θα αναφέρεται με το όνομα Reports Server). Το Reports Server δέχεται αιτήσεις εργασιών από σταθμούς εργασίας (client συστήματα) και διατηρεί μία "ουρά" αιτήσεων. Για την εμφάνιση των περιεχομένων της ουράς και την διαχείρισή της χρησιμοποιείται το πρόγραμμα Reports Queue Manager (r30rqm32.exe).

Για την "δυναμική" παραγωγή αναφορών μέσα από το περιβάλλον μιας εφαρμογής Web browser, χρησιμοποιείται το Reports Server σε συνδυασμό με το Reports Web Cartridge ή το Web CGI. Το Reports Web Cartridge ή το Web CGI επιτρέπει στον Web server σας να στέλνει τις σχετιζόμενες με αναφορές αιτήσεις στο Reports Server για την διεκπεραίωσή τους, και το αποτέλεσμα επιστρέφει στην εφαρμογή Web browser.

4.2.5.3 Διαδικασία

Για την δημιουργία μιας αναφοράς με το Report Builder θα πρέπει να ακολουθηθούν τα παρακάτω βήματα:

- "Χρήση του οδηγού Report Wizard για την δημιουργία μιας αναφοράς.
- Προσαρμογή της αναφοράς στις προβολές Live Previewer, Data Model, Layout Model και Parameter Form του Report Editor.
- Αποστολή της αναφοράς με την κατάλληλη μορφή, στον κατάλληλο «προορισμό», όπως π.χ. ένας εκτυπωτής PostScript, ένα HTML αρχείο, ή ένα PDF αρχείο.

4.2.6 Εισαγωγή στο Procedure Builder

4.2.6.1. Σύνοψη

Το Procedure Builder είναι ένα ενοποιημένο περιβάλλον για την ανάπτυξη και συντήρηση κώδικα εφαρμογών client/server.

4.2.6.2 Βασικές Έννοιες

Το Procedure Builder παρέχει ένα γραφικό σύστημα επικοινωνίας με τον χρήστη για την δημιουργία, τροποποίηση και μεταγλώττιση PL/SQL κώδικα - συμπεριλαμβανομένων μονάδων προγραμμάτων, βιβλιοθηκών και triggers βάσεων δεδομένων - ο οποίος μπορεί να τρέχει στην πλευρά του client συστήματος ή στην πλευρά του server.

Το Object Navigator παρουσιάζει με ιεραρχικό τρόπο αυτά τα αντικείμενα και επιτρέπει την εκκίνηση οποιονδήποτε συντακτών ή παραθύρων διαλόγου του Procedure Builder. Μπορεί να χρησιμοποιηθεί η τεχνική μεταφοράς και απόθεσης (drag and drop) στο Object Navigator για την αλλαγή των σχέσεων μεταξύ των αντικειμένων. Μπορούν επίσης να διαιρεθούν (partition) οι εφαρμογές μεταφέροντας διαδικαστικό κώδικα μεταξύ του client και του server.

Επειδή το Procedure Builder είναι ενσωματωμένο στα Form Builder, Report Builder και Graphics Builder, μπορεί επίσης να χρησιμοποιηθεί η διαδικασία μεταφοράς και απόθεσης PL/SQL κώδικα σ' αυτά τα εργαλεία.

Το Program Unit Editor παρέχει πλήρεις δυνατότητες επεξεργασίας κειμένου για την εισαγωγή και τροποποίηση του πηγαίου κώδικα του προγράμματός. Μπορεί επίσης να χρησιμοποιηθεί το Program Unit Editor για την μεταγλώττιση μονάδων προγραμμάτων και την διόρθωση των συντακτικών σφαλμάτων του PL/SQL κώδικα, με την βοήθεια ενός παραθύρου εμφάνισης μηνυμάτων σφάλματος.

Το Database Trigger Editor παρέχει ένα γραφικό σύστημα επικοινωνίας με τον χρήστη για την δημιουργία triggers βάσεων δεδομένων.

Ο Διερμηνευτής (Interpreter) είναι ο κεντρικός χώρος αποσφαλμάτωσης, στον οποίο ορίζονται οι ενέργειες αποσφαλμάτωσης και δοκιμάζονται οι μονάδες προγραμμάτων σε προσομοίωση χρόνου εκτέλεσης (runtime simulation). Ένα σημείο διακοπής (breakpoint) αναστέλλει την εκτέλεση μιας μεταγλωττισμένης μονάδας προγράμματος, πράγμα το οποίο επιτρέπει την εκτέλεση βήμα προς βήμα του κώδικα και την λύση οποιονδήποτε προβλημάτων που προκύπτουν κατά τον χρόνο εκτέλεσης. Λόγω του ενοποιημένου client-server περιβάλλοντος, του Procedure Builder, για την αποσφαλμάτωση αποθηκευμένων μονάδων προγραμμάτων απαιτείται η ίδια διαδικασία όπως και για την αποσφαλμάτωση τοπικών μονάδων προγραμμάτων.

Οι βιβλιοθήκες είναι συλλογές μονάδων προγραμμάτων οι οποίες αποθηκεύονται είτε στο client σύστημα, είτε στον server, για εύκολη επαναχρησιμοποίηση σε πολλές εφαρμογές. Οι βιβλιοθήκες παρέχουν επίσης πλεονεκτήματα στον τομέα της απόδοσης, επειδή οι μονάδες προγραμμάτων που περιλαμβάνονται σε μία βιβλιοθήκη δεν φορτώνονται στην μνήμη παρά μόνο την στιγμή που καλούνται. Οι μονάδες προγραμμάτων που περιέχονται σε βιβλιοθήκες φορτώνονται στην μνήμη μόνο την στιγμή που είναι απαραίτητες στην εφαρμογή, και η αναζήτησή τους γίνεται με την σειρά που υπαγορεύει η ιεραρχία αντικειμένων στο Object Navigator.

4.2.6.3 Διαδικασία

Για να δημιουργηθεί μία μονάδα προγράμματος με το Procedure Builder πρέπει να ακολουθηθούν τα παρακάτω βήματα:

- Χρησιμοποίηση του συντάκτη για τη δημιουργία και τη μεταγλώττιση των μονάδων προγραμμάτων PL/SQL.

- Τρέξιμο δοκιμαστικά της μονάδας προγράμματος σε προσομοίωση του χρόνου εκτέλεσης.
- Εκτέλεση των απαιτούμενων ενεργειών αποσφαλμάτωσης, έλεγχος της ροής εκτέλεσης και εξέταση της κατάστασης του προγράμματος.
- Επανάληψη του κύκλου ανάπτυξης αυξητικά. Τρέξιμο δοκιμαστικά κάθε μονάδα προγράμματος αφού γίνουν οι απαιτούμενες αλλαγές.

4.2.7 Εισαγωγή στο Query Builder

4.2.7.1 Σύνοψη

Το Query Builder παρέχει έναν διαισθητικό τρόπο για την προσπέλαση των πληροφοριών από τις βάσεις δεδομένων του οργανισμού σας, για σκοπούς ανάλυσης και παραγωγής αναφορών. Το Query Builder εμφανίζει τους πίνακες με γραφικό τρόπο, παρέχοντάς οπτική βοήθεια καθώς γίνεται χρησιμοποίηση των μηχανισμών του για την δημιουργία ερωτημάτων.

4.2.7.2 Βασικές Έννοιες

Κάθε πίνακας από τον οποίο επιλέγει δεδομένα ένα ερώτημα εμφανίζεται σαν ένα πλαίσιο στο παράθυρο Query. Οι στήλες που περιλαμβάνει ο πίνακας παρουσιάζονται στο παράθυρο Query κάτω από το όνομα του πίνακα.

Το παράθυρο Query χωρίζεται σε δύο περιοχές - την περιοχή Conditions (συνθήκες) και την περιοχή Datasource (πηγή δεδομένων). Στην περιοχή Datasource καθορίζονται οι πίνακες που πρέπει να συμπεριληφθούν στο ερώτημά και οι στήλες που θα εξεταστούν από αυτούς τους πίνακες. Το Query Builder κατανοεί τις σχέσεις μεταξύ των πινάκων και τις απεικονίζει με γραφικό τρόπο, χρησιμοποιώντας γραμμές. Μπορούν να οριστούν συνθήκες για την σύγκριση ή αναζήτηση τιμών, και μπορούν επίσης να χρησιμοποιηθούν πολλαπλές συνθήκες σε ένα ερώτημα.

Μπορούν να ταξινομηθούν τα αποτελέσματα ενός ερωτήματος, να παραχθούν συγκεντρωτικά αποτελέσματα και να χρησιμοποιηθούν δεδομένα από διαφορετικά μέρη της βάσης δεδομένων. Μπορούν επίσης να εκτελεστούν υπολογισμοί των δεδομένων για τη δημιουργία σεναρίων "τι θα συμβεί εάν" (what-if) και προβλέψεων, να αλλαχθούν οι γραμματοσειρές και οι άλλες ρυθμίσεις μορφοποίησης, να εκτυπωθούν και να αποθηκευτούν τα

δεδομένα που ανακτήθηκαν, καθώς και να εξαχθούν σε άλλες εφαρμογές, όπως το Lotus 1-2-3 και το Microsoft Excel.

Το Query Builder εμφανίζει τα ζητούμενα δεδομένα στο παράθυρο Results (αποτελέσματα).

Το Data Editor επιτρέπει την εισαγωγή, την ενημέρωση και τη διαγραφή εγγραφών από την βάση δεδομένων. Εκτελεί τις εντολές DML (διαχείρισης δεδομένων) που απαιτούνται για την επεξεργασία των δεδομένων της βάσης δεδομένων.

Σημείωση: Επειδή το Data Editor επιτρέπει την αλλαγή ή την διαγραφή δεδομένων, στην αρχική διαμόρφωση του Query Builder το Data Editor είναι απενεργοποιημένο. Για την προσπέλαση του θα πρέπει να υπάρχουν τα κατάλληλα δικαιώματα στην βάση δεδομένων και ο επόπτης της βάσης δεδομένων θα πρέπει να έχει ενεργοποιήσει το Data Editor στην πλευρά του server.

4.2.7.3 Διαδικασία

Για να δημιουργηθεί ένα ερώτημα με τη χρησιμοποίηση του Query Builder, θα πρέπει να ακολουθηθούν τα παρακάτω βήματα:

- Στην περιοχή Datasource καθορίζονται οι πίνακες που θα συμπεριληφθούν στο ερώτημά και οι στήλες από τις οποίες θα επιλεγούν τα δεδομένα.
- Καθορισμός συγκρίσεων και οποιεσδήποτε άλλες ρυθμίσεις χρειάζονται για το ερώτημα, όπως η σειρά ταξινόμησης ή οι συγκεντρωτικοί υπολογισμοί.
- Τρέξιμο του ερωτήματος.
- Χρησιμοποίηση του Data Editor για την εισαγωγή, την ενημέρωση και τη διαγραφή εγγραφών στη βάση δεδομένων.

4.2.8 Εισαγωγή στο Schema Builder

4.2.8.1 Σύνοψη

Το Schema Builder είναι ένα βασιζόμενο σε γραφικό περιβάλλον εργαλείο το οποίο επιτρέπει τη δημιουργία, την αντιγραφή, την τροποποίηση και τη διαγραφή των αντικειμένων και των σχέσεων τους από την βάση δεδομένων.

4.2.8.2 Βασικές Έννοιες

Ένα σχήμα (schema) είναι μία συλλογή λογικών αντικειμένων της βάσης δεδομένων, όπως οι πίνακες, οι όψεις, τα ευρετήρια και οι σχέσεις, τα οποία ορίζονται σύμφωνα με τις ανάγκες του χρήστη.

Το Schema Builder παρέχει έναν τρόπο για να εμφανίζονται διάφοροι συνδυασμοί αντικειμένων της βάσης δεδομένων, ανάλογα με τα αντικείμενα με τα οποία θα δουλευτούν. Συνήθως τα αντικείμενα της βάσης δεδομένων που περιλαμβάνονται στο Schema Builder είναι εκείνα στα οποία αναμένετε ότι θα εκτελούνται λειτουργίες διαχείρισης, όπως η αντιγραφή, η τροποποίηση και η διαγραφή.

Ένα σχήμα είναι μία λογική δομή, σχεδιασμένη κυρίως για την ευκολία του χρήστη. Θα χρησιμοποιηθεί το Schema Builder για να οριστεί μία σειρά λειτουργιών SQL μέσα στο σχήμα. Μία εντολή Commit εκτελεί όλες αυτές τις λειτουργίες στην βάση δεδομένων.

Το περιβάλλον του Schema Builder είναι ειδικά σχεδιασμένο ώστε να δείχνει παρόμοιο με το παράθυρο Query του Query Builder. Ωστόσο, αντί να τρέχουν τα ερωτήματα σε πίνακες, το Schema Builder επιτρέπει να εκτελούνται οι λειτουργίες της γλώσσας ορισμού δεδομένων (DDL). Μπορούν να τροποποιούνται τα λεξικά δεδομένων (data dictionaries) και να επαναπροσδιορίζονται τα αντικείμενα της βάσης δεδομένων μέσα από ένα περιβάλλον το οποίο δίνει την δυνατότητα εξέτασης των αλλαγών με οπτικό τρόπο πριν οριστικοποιηθούν στη βάση δεδομένων.

Σημείωση: Επειδή το Schema Builder επιτρέπει την αλλαγή ή διαγραφή πινάκων ζωτικής σημασίας, δεν μπορεί να προσπελαστεί παρά μόνο εάν υπάρχουν τα κατάλληλα δικαιώματα στην βάση δεδομένων. Για περισσότερες πληροφορίες, θα πρέπει να υπάρξει επικοινωνία με τον επόπτη του συστήματός.

4.2.8.3 Διαδικασία

Για να δημιουργηθεί ένα σχήμα χρησιμοποιώντας το Schema Builder, πρέπει να ακολουθηθούν τα παρακάτω βήματα:

- Δημιουργία ενός νέου σχήματος.
- Επιλογή πινάκων δεδομένων για το σχήμα.
- Δημιουργία και ενεργοποίηση των σχέσεων.

4.2.9 Εισαγωγή στο Translation Builder

4.2.9.1 Σύνοψη

Το Oracle Translation Builder υποστηρίζει και διαχειρίζεται μεταφράσεις κειμένου το οποίο εξάγεται από αρχεία πόρων (resource files) της Oracle, όπως εφαρμογές του Developer 2000, καθώς και αρχεία πόρων τα οποία προέρχονται από άλλες πηγές, όπως τα.rc αρχεία των Microsoft Windows και τα HTML αρχεία.

4.2.9.2 Βασικές Έννοιες

Το Translation Builder μπορεί να χρησιμοποιηθεί για την οργάνωση έργων και γλωσσάριων και την διαχείριση της μετάφρασης των λειτουργικών μονάδων του έργου. Επίσης, επιτρέπει την επαναχρησιμοποίηση προηγούμενων μεταφράσεων.

Για την εκτέλεση μιας μετάφρασης, το Translation Builder εξάγει τα αλφαριθμητικά που πρόκειται να μεταφραστούν από τα αρχεία πόρων και τα εισάγει σε μία ειδική "περιοχή μετάφρασης" (translation repository).

Το Project Navigator διαχειρίζεται συνολικά την διαδικασία και διασφαλίζει την ακεραιότητα των δεδομένων. Από το Project Navigator μπορεί να γίνει εκκίνηση του Συντάκτη (Editor) ο οποίος χρησιμοποιείται για την μετάφραση του συγκεκριμένου συνόλου αλφαριθμητικών και την προσθήκη των μεταφράσεων στο Repository. Αφού ολοκληρωθεί η μετάφραση, τα αλφαριθμητικά εξάγονται από το Repository και κατόπιν τοποθετούνται στα αρχεία πόρων. Εάν ο συγκεκριμένος τύπος πόρου υποστηρίζει πολλαπλές γλώσσες, οι μεταφράσεις προστίθενται στο αρχικό αρχείο πόρου. Αλλιώς δημιουργείται ένα αντίγραφο του πόρου, στο οποίο τα βασικά αλφαριθμητικά αντικαθίστανται από τις μεταφράσεις τους.

Μία από τις βασικές δυνατότητες του Translation Builder είναι ότι αποθηκεύει τις προηγούμενες μεταφράσεις στο Repository μεταφράσεων. Το σύνολο των προηγούμενων μεταφράσεων και γλωσσάριων αναφέρεται

συλλογικά σαν "μνήμη μεταφράσεων" (Translation Memory). Οι αναβαθμίσεις και οι αλλαγές που γίνονται σε μία εφαρμογή δεν απαιτούν την επανάληψη της διαδικασίας μετάφρασης από την αρχή. Κάθε νέα μετάφραση προστίθεται στην μνήμη μεταφράσεων (Translation Memory), καθιστώντας γρηγορότερη και ευκολότερη την εκτέλεση μελλοντικών μεταφράσεων. Η μνήμη μεταφράσεων παρέχει επίσης δυνατότητα ελέγχου εκδόσεων για το κείμενο και τις μεταφράσεις.

Οι λειτουργικές μονάδες μπορούν να μεταφράζονται από οποιαδήποτε γλώσσα σε οποιαδήποτε άλλη γλώσσα. Αφού ολοκληρωθεί μία συγκεκριμένη μετάφραση, η μετάφραση αυτή μπορεί να χρησιμοποιηθεί σαν βάση για μελλοντικές μεταφράσεις. Για παράδειγμα, εάν μία λειτουργική μονάδα πρέπει να μεταφραστεί από τα Αγγλικά στα Παραδοσιακά και Απλοποιημένα Κινέζικα, μπορεί να μεταφραστεί πρώτα στα Παραδοσιακά Κινέζικα και κατόπιν σε Απλοποιημένα Κινέζικα, αντί να ξεκινήσει τη μετάφραση από την αρχική Αγγλική έκδοση. Αυτό επιταχύνει τις μεταφράσεις και μειώνει το κόστος τους.

Το Translation Builder μπορεί να χρησιμοποιηθεί σε αυτόνομα συστήματα και σε περιβάλλοντα client-server, και οι μεταφράσεις που παράγει είναι μεταφερτές σε πολλαπλές πλατφόρμες.

4.2.9.3 Διαδικασία

Για να μεταφραστεί ένας πόρος με τη χρήση του Translation Builder, θα πρέπει να ακολουθηθούν τα παρακάτω βήματα:

- Εξαγωγή των αλφαριθμητικών που πρέπει να μεταφραστούν από τα αρχεία πόρων (είτε της Oracle, είτε άλλων εφαρμογών) και εισαγωγή αυτών στο Translation Repository.
- Υπόδειξη της γλώσσας ή των γλωσσών στις οποίες θα μεταφραστεί μία συγκεκριμένη λειτουργική μονάδα.
- Χρησιμοποίηση του Translation Editor για την εκτέλεση της μετάφρασης.

- Εξέταση του αποτελέσματος της μετάφρασης και πραγματοποίηση των απαιτούμενων προσαρμογών.
- Εξαγωγή των μεταφρασμένων αλφαριθμητικών από το Translation Repository και ενσωμάτωσή τους ξανά στο πρωτότυπο αρχείο πόρων ή σ' ένα αντίγραφό του, με τη χρήση του βοηθήματος Export.

ΚΕΦΑΛΑΙΟ 5 – Συμπεράσματα

Το μέλλον των επιχειρήσεων καθορίζεται αποφασιστικά από τις στρατηγικές τους αποφάσεις. Ένα καλά δομημένο **Πληροφοριακό Σύστημα Διοίκησης** είναι ένα **Στρατηγικό Εργαλείο** για την επιχείρηση, το οποίο την βοηθά σημαντικά να αξιοποιήσει τις ευκαιρίες και να αντιμετωπίσει του κινδύνους που θα της παρουσιαστούν. Χρειάζεται, επομένως, οι επιχειρήσεις να επενδύουν στα Πληροφοριακά Συστήματα Διοίκησης και να προσλαμβάνουν διοικητικά στελέχη με επιχειρησιακές γνώσεις, αλλά και γνώσεις πληροφορικής.

Παλαιότερα η λειτουργία των Πληροφοριακών Συστημάτων ήταν κάτω από τον έλεγχο του τμήματος της μηχανογράφησης μέσα σε μια επιχείρηση. Αυτό είχε σαν αποτέλεσμα την αντίδραση των χρηστών. Με την πάροδο του χρόνου οι χρήστες αναζητούσαν την υποστήριξη της μηχανογράφησης, για να αντιμετωπιστούν οι μεταβαλλόμενες απαιτήσεις τους. Παρά το γεγονός ότι η μηχανογράφηση δεν ήταν πάντοτε σε θέση να αντιμετωπίσει αποτελεσματικά τις απαιτήσεις των χρηστών, η κατάσταση αυτή οδήγησε στην ανάγκη δημιουργίας φιλικότερων προγραμμάτων για το χρήστη. Αυτό είχε σαν συνέπεια την εμφάνιση του *end-user computing*, δηλαδή εκείνων των τελικών χρηστών, οι οποίοι παρέχουν τόσο τις τεχνικές, όσο και τις λειτουργικές γνώσεις για την ανάπτυξη των συστημάτων.

Το *end-user computing* επιτρέπει στο διευθυντικό στέλεχος να αλληλεπιδρά απ' ευθείας με το Πληροφοριακό Σύστημα με τη χρήση φιλικών προς αυτόν εργαλείων ανάπτυξης λογισμικού. Αυτό φυσικά δε σημαίνει ότι οι ειδικοί των Η/Υ δεν είναι πλέον απαραίτητοι, αλλά απλά ότι τώρα περισσότερο υποστηρίζουν παρά εκτελούν τις ενέργειες των χρηστών. Δηλαδή στα καθήκοντά τους συμπεριλαμβάνεται και η απόκτηση ή η παροχή του λογισμικού, που είναι απαραίτητο για το end-user computing, η παροχή βοήθειας στους χρήστες κυρίως για την επανάκτηση των πληροφοριών, καθώς επίσης και η επίλυση των τεχνικών προβλημάτων που εμφανίζονται.

Αξίζει να σημειωθεί ότι όσες εφαρμογές μιας επιχείρησης δεν μπορούν να αναπτυχθούν με τη βοήθεια του end-user computing, παραμένουν στη δικαιοδοσία της μηχανογράφησης. Σ' αυτή την κατηγορία ανήκουν κυρίως εφαρμογές, οι οποίες είναι μεγάλες και σημαντικές για την επιχείρηση και που συνήθως αφορούν τις βασικές λειτουργίες της. Για παράδειγμα, η επεξεργασία της μισθοδοσίας των εργαζομένων σε μια επιχείρηση θα πρέπει να παραμείνει στην αρμοδιότητα του τμήματος της μηχανογράφησης. Αντίθετα, εφαρμογές όπως είναι η επεξεργασία κειμένου, η ανάλυση των τάσεων της αγοράς, κ.α., ανήκουν στη δικαιοδοσία των χρηστών και μπορούν να εκτελεσθούν καλύτερα με τη βοήθεια του end-user computing.

Για τη σχεδίαση ενός δομημένου Πληροφοριακού Συστήματος Διοίκησης πολύ σημαντικό ρόλο κατέχουν τα σύγχρονα εργαλεία ανάπτυξης (*CASE TOOLS – Computer Aided Software Engineering*). Με τη χρήση αυτών, οι επιχειρήσεις επιτυγχάνουν τα ακόλουθα:

- Ευελιξία στη σχεδίαση των νέων εφαρμογών,
- Απλή και γρήγορη δημιουργία νέων λειτουργικών εφαρμογών,
- Ανάπτυξη εφαρμογών στρατηγικής σημασίας, χτίζοντας πάνω σε παλιές μηχανογραφικές εφαρμογές, ενσωματώνοντας άμεσα τα υπάρχοντα δεδομένα,
- Εύκολη τροποποίηση των εφαρμογών που έχουν αναπτυχθεί, προκειμένου να μπορούν να αποδώσουν τα αναγκαία, επίκαιρα στοιχεία,
- Αποθήκευση και συντήρηση των δεδομένων σε ενιαία Βάση Δεδομένων, που συνεπάγεται αξιοπιστία των δεδομένων και εύκολη τροποποίησή τους,
- Αύξηση της παραγωγικότητας των τελικών χρηστών μέσω της βελτίωσης της ποιότητας των τελικών εφαρμογών,

Επομένως, οι σύγχρονες επιχειρήσεις πρέπει να επενδύουν στα CASE Tools και στη σχετική τεχνογνωσία προκειμένου να υλοποιούν αξιόπιστα Πληροφοριακά Συστήματα Διοίκησης.-

Ελληνική Βιβλιογραφία

Οικονόμου Γεώργιος, Γεωργόπουλος Νικόλαος, (1995), «Πληροφοριακά Συστήματα για τη Διοίκηση Επιχειρήσεων»

Χατζόγλου Πρόδρομος, (1994), «Τεχνικές Ανάλυσης & Σχεδίασης Πληροφοριακών Συστημάτων», Εκδόσεις ΙΩΝ

Χριστοδουλάκης Δημήτρης, Ξένος Μιχάλης, (1994), «Τεχνολογία Λογισμικού», Εκδόσεις Πανεπιστημίου Πατρών

Ξένη Βιβλιογραφία

ANSI, (1975), "Interim Report of ANSI/X3/SPARC Group on Data Base Management Systems", ANSI, Febr. 1975

Chen P.P., (1976), "The Entity Relationship Model: Towards a Unified View of Data", in ACM Transactions on Data Base Systems, vol.1 No.1, March 1976, pp 9-36

Chen P.P., (1977), "The Entity Relationship Model: A Basis for the Enterprise View of Data", National Computer Conference

Connor D., (1985), "Information System Specification & Design Road Map", Prentice-Hall International

DeMarco T., (1978), "Structured Analysis & System Specification", Yourdon Press

DeMarco T., (1979), "Structured Analysis & System Specification", Prentice-Hall

Gane S. & Sarson T., (1970), "Structured Analysis", Prentice-Hall, Englewood Cliffs, N.J.

Gane S. & Sarson T., (1986), "Structured Systems Analysis Tools & Techniques", Improved System Technologies Databooks

Hawryszkiewyck I.T., (1988), "Introduction to System Analysis & Design", Prentice-Hall

Jackson M.A., (1975), "Principles of Program Design", Academic Press

Jackson M.A., (1983), "Systems Development", Prentice-Hall

Layzell P.J. & Loucopoulos P., (1987), "Systems Analysis & Development", 2nd ed., Chartwell Bratt, Student litteratur

Martin D. & Estrin G., (1967), Models of Computations & Systems Evaluation of Vertex Probabilities in Graph Models of Computations", Journal of the ACM, vol.14, No.2, April 1967

Myers G.J., (1975), "Relable Software through Composite Design", Petrocelli/Charter

Peters L., (1988), "Advanced Structured Analysis & Design", Prentice-Hall International

Ross D., (1977), "Structured Analysis (SA): A Language for Communicating Ideas", IEEE Transactions on Software Engineering, January 1977

Stevens W.P., Myers G.J. & Constantine L.L., (1974), "Structured Design", IBM Systems Journal, vol.13, No.2, 1974, pp. 115-139

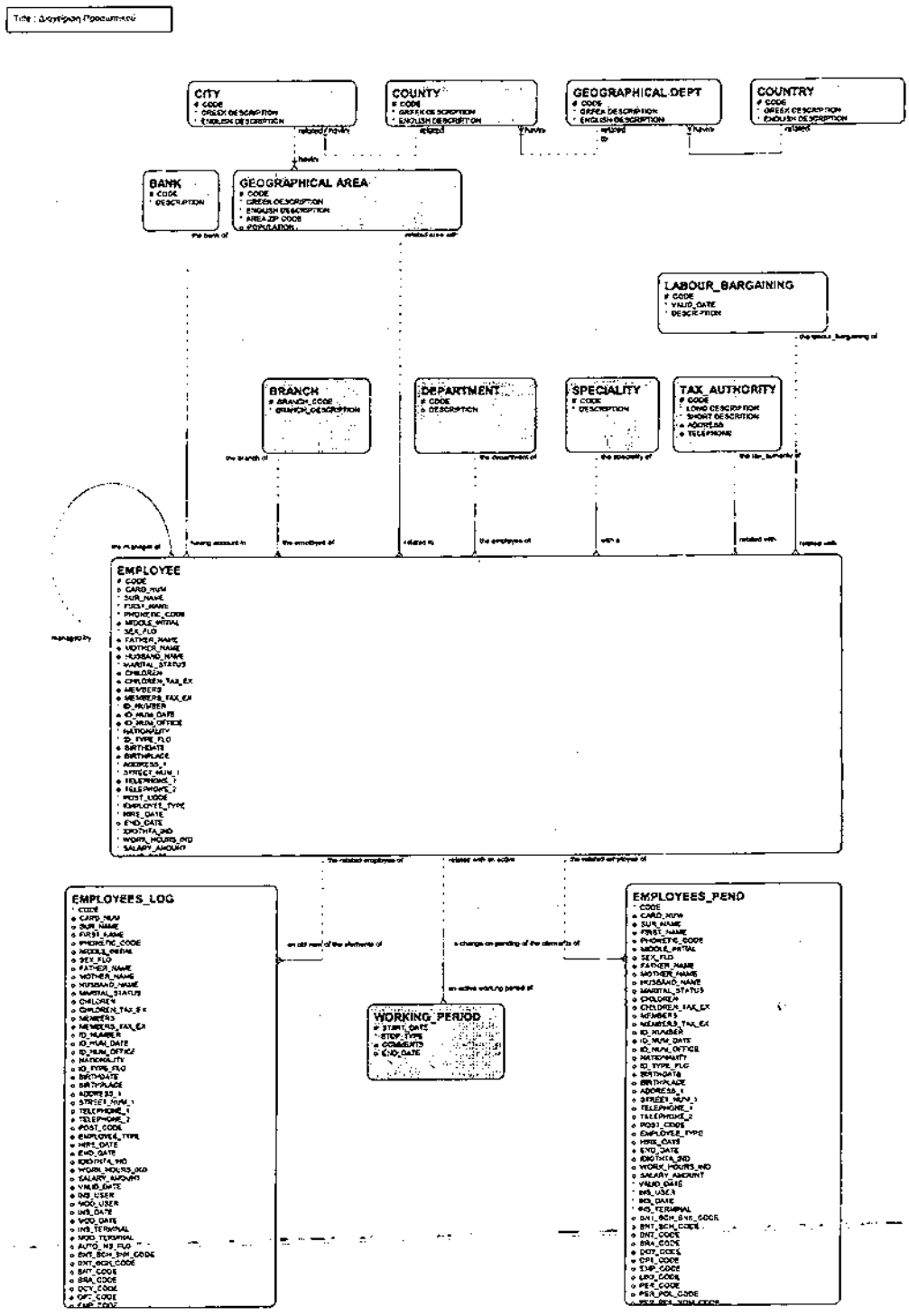
Travis B.J., (1987), "Auditing the Development of Computing Systems", Butterworths

Weinberg V., (1978), "Structured Analysis", Prentice-Hall, Englewood Cliffs, N.J.

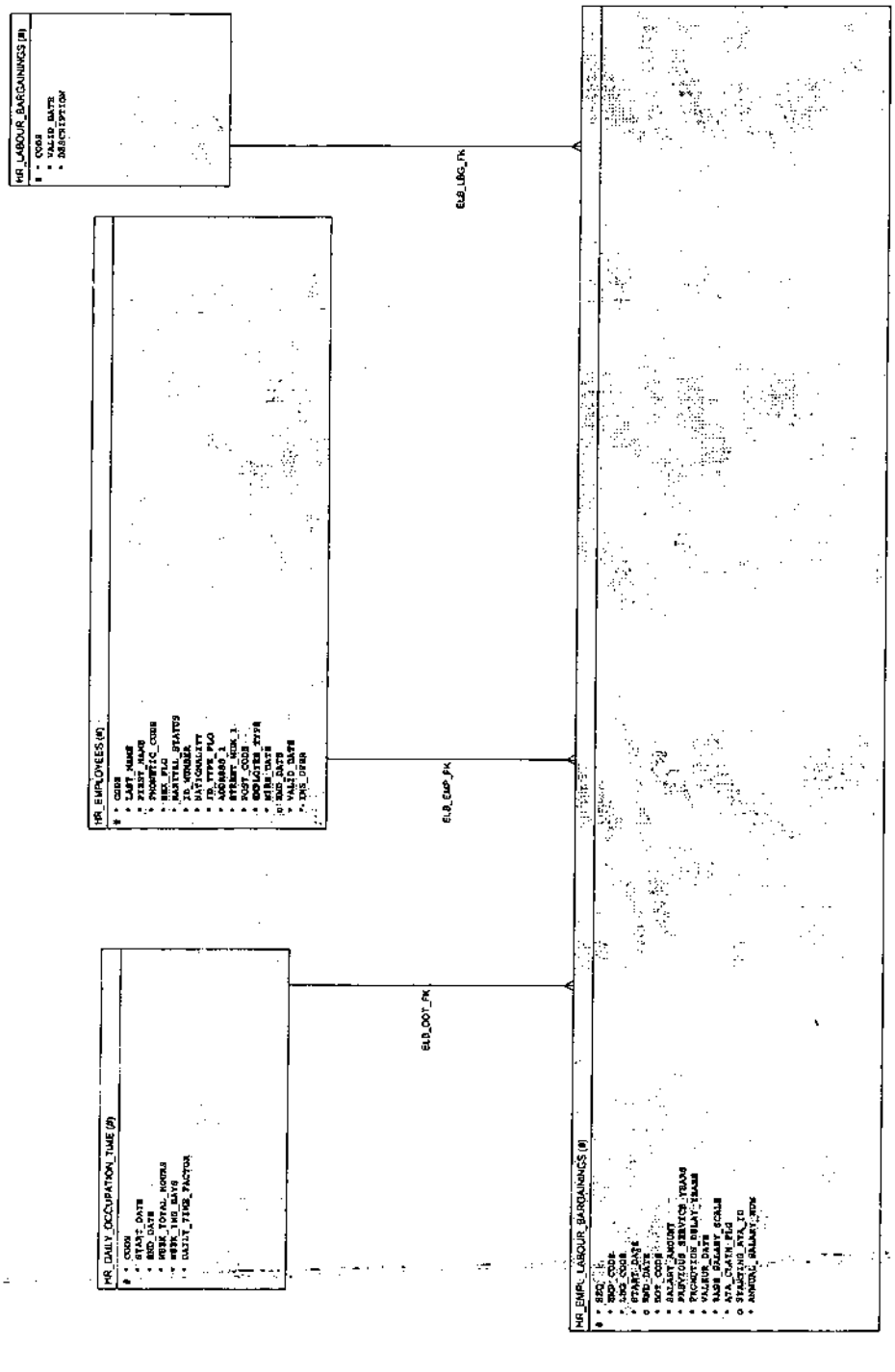
Yourdon E., (1989), "Modern Structured Analysis", Prentice-Hall International

Παραρτήματα

1. Διάγραμμα Οντοτήτων – Συσχετίσεων (Entity Relationship Diagram)



2. Σχηματικό Διάγραμμα Δεδομένων (Data Schema Diagram)



3. SQL Code (Κώδικας SQL)

EPD.SQL

```
REM
REM  This ORACLE command file was generated by Oracle
REM  Server Generator Version 5.5.8.2.0 on 10-JUN-02
REM
REM  For application HR version 1 database V72
REM
SET SCAN OFF

SPOOL EPD.lst

REM TABLE CREATION
start  EPD.tab

REM INDEX CREATION
start  EPD.ind

REM CONSTRAINT CREATION
start  EPD.con

REM
REM  End of command file
REM
SPOOL OFF
```

EPD.TAB

```
REM
REM  This ORACLE command file was generated by Oracle
REM  Server Generator Version 5.5.8.2.0 on 10-JUN-02
REM
REM  For application HR version 1 database V72
REM
REM  TABLE
REM  HR_EMPLOYEES_PENDS
REM
PROMPT
PROMPT Creating Table HR_EMPLOYEES_PENDS
CREATE TABLE hr_employees_pends(
 seq                    NUMBER(10,0)          NOT NULL,
 sex_flg                   VARCHAR2(1)        NULL
     CHECK ( sex_flg IN ( '1' , '2' )  ),
 manager_code              VARCHAR2(14)       NULL,
```

```
card_num                    VARCHAR2(15)        NULL,
last_name                   VARCHAR2(30)        NULL,
first_name                  VARCHAR2(20)        NULL,
phonetic_code               VARCHAR2(15)        NULL,
middle_initial              VARCHAR2(10)        NULL,
father_name                 VARCHAR2(20)        NULL,
mother_name                 VARCHAR2(20)        NULL,
husband_name                VARCHAR2(20)        NULL,
marital_status              VARCHAR2(4)         NULL
    CHECK ( marital_status IN ( '0', '1', '2', '3', '4',
          '5', '6', '7', '8', '9')  ),
Children                    NUMBER(3,0)         NULL,
children_tax_ex             NUMBER(3,0)         NULL,
members                     NUMBER(3,0)         NULL,
members_tax_ex              NUMBER(3,0)         NULL,
id_number                   VARCHAR2(15)        NULL,
id_num_date                 DATE                NULL,
id_num_office               VARCHAR2(30)        NULL,
nationality                 VARCHAR2(20)        NULL,
id_type_flg                 VARCHAR2(4)         NULL
    CHECK ( id_type_flg IN ( '1', '2', '3', '4', '5') ),
birthdate                   DATE                NULL,
birthplace                  VARCHAR2(20)        NULL,
address_1                   VARCHAR2(30)        NULL,
street_num_1                VARCHAR2(10)        NULL,
telephone_1                 VARCHAR2(30)        NULL,
telephone_2                 VARCHAR2(30)        NULL,
post_code                   VARCHAR2(15)        NULL,
employee_type               CHAR(1)             NULL
    CHECK ( employee_type IN ( '0' , '1' , '2' )  ),
hire_date                   DATE                NULL,
end_date                    DATE                NULL,
property                    VARCHAR2(4)         NULL
    CHECK ( property IN ( '4', '3', '2', '1', '5' )  ),
per_code                    NUMBER(3)           NULL,
per_pol_code                NUMBER(3)           NULL,
per_pol_nom_code            NUMBER(3)           NULL,
per_pol_nom_gdp_code        NUMBER(3)           NULL,
per_pol_nom_gdp_xwr_code    NUMBER(3)           NULL,
bnt_account_code            VARCHAR2(20)        NULL,
bra_code                    VARCHAR2(4)         NULL,
dpt_code                    VARCHAR2(14)        NULL,
sps_code                    VARCHAR2(14)        NULL,
doy_code                    NUMBER(4,0)         NULL,
bnt_bch_bnk_code            NUMBER(5,0)         NULL,
bnt_bch_code                NUMBER(5,0)         NULL,
valid_date                  DATE                NOT NULL,
ins_user                    VARCHAR2(14)        NOT NULL,
ins_date                    DATE                NOT NULL,
```

```
 ins_terminal                VARCHAR2(14)       NOT NULL,
 emp_code                    VARCHAR2(14)       NOT NULL,
 category                    VARCHAR2(4)        NULL
     CHECK ( category IN ( '1', '2', '3', '4', '5')  ),
 tax_id                      VARCHAR2(15)       NULL,
 education_level             VARCHAR2(1)        NULL,
 education_major             VARCHAR2(60)       NULL,
 training                    VARCHAR2(60)       NULL,
 experties                   VARCHAR2(60)       NULL
)
;

COMMENT ON TABLE hr_employees_pends IS
       'Κάθε εγγραφή απεικονίζει τα στοιχεία που
        πρόκειται να αλλάξουν στον εργαζόμενο σε
        δεδομένη χρονική στιγμή.'

;
```

EPD.IND

```
REM
REM  This ORACLE command file was generated by Oracle
REM  Server Generator Version 5.5.8.2.0 on 10-JUN-02
REM
REM  For application HR version 1 database V72
REM
REM  INDEX
REM  EPD_EMP_FK_I


REM
REM
REM
PROMPT
PROMPT    Creating    Index    EPD_EMP_FK_I    on    Table
HR_EMPLOYEES_PENDS
CREATE INDEX EPD_EMP_FK_I ON HR_EMPLOYEES_PENDS
(emp_code )
PCTFREE  40
;
```

EPD.CON

```
REM
REM  This ORACLE command file was generated by Oracle
REM  Server Generator Version 5.5.8.2.0 on 10-JUN-02
REM
REM  For application HR version 1 database V72
REM
REM  CONSTRAINT

PROMPT Adding PRIMARY Constraint To HR_EMPLOYEES_PENDS
Table

ALTER TABLE HR_EMPLOYEES_PENDS ADD (
      CONSTRAINT EPD_PK
      PRIMARY KEY (SEQ)
USING INDEX
PCTFREE  10
)
/

PROMPT Adding FOREIGN Constraint To HR_EMPLOYEES_PENDS
Table

ALTER TABLE HR_EMPLOYEES_PENDS ADD (
      CONSTRAINT EPD_EMP_FK
      FOREIGN KEY (EMP_CODE)
      REFERENCES  HR_EMPLOYEES (CODE)
)
/
```