ΜΕΘΟΔΟΛΟΓΙΑ ΑΝΑΠΤΥΞΗΣ ΕΜΠΟΡΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

Η μεθοδολογία ανάπτυξης μιας εμπορικής εφαρμογής δίνει την δυνατότητα στην ομάδα εργασίας να έχει τον πλήρη έλεγχο όλων των φάσεων ανάπτυξης της εφαρμογής και να διασφαλίσει τον σωστό συντονισμό των ενεργειών που θα οδηγήσουν στην επιτυχία του έργου. Για να ολοκληρωθεί ένα έργο, το οποίο περιλαμβάνει την ανάπτυξη μιας εμπορικής εφαρμογής, είναι απαραίτητο να ακολουθηθούν κάποιες συγκεκριμένες διαδικασίες, που τοποθετούνται μέσα σε σαφώς καθορισμένα στάδια. Τα βασικά στάδια-φάσεις για την ολοκλήρωση μιας εμπορικής εφαρμογής παρουσιάζονται στην συνέχεια.

1. Εισαγωγή στην Ανάπτυξη Εφαρμογών

1.1. Στάδια Ανάπτυξης Εφαρμογών

Στάδιο 1: Ανάλυση αναγκών και απαιτήσεων
(Πρόκειται για τον ορισμό του προβλήματος)
Καταγράφεται ένα σύνολο απαιτήσεων που συμφωνούνται μεταξύ του πελάτη και της ομάδας που παράγει το λογισμικό. Συνήθως η καταγραφή γίνεται σε φυσική γλώσσα και το αποτέλεσμα είναι ένα κείμενο που περιγράφει σε γενικές γραμμές τι θα είναι το προϊόν αλλά και πώς θα δουλεύει ακριβώς ή πώς θα είναι δομημένο εσωτερικά.

Στάδιο 2: Προδιαγραφές
(Συμβόλαιο Πελάτη-Ομάδα Προγραμματιστών)
Το τμήμα που αναπτύσσει τις τυπικές προδιαγραφές δηλ.
- Λεπτομέρειες τρόπου λειτουργίας
- Περιγραφή συνδεομένων τμημάτων
- Απαιτήσεις σε ζητήματα λειτουργίας

Στάδιο 3: Πλάνο εργασιών – Χρονοπρογραμματισμός
Σε αυτό το στάδιο το έργο χωρίζεται σε πακέτα εργασίας, καθένα από τα οποία μπορεί να χωρισθεί σε μικρότερα κομμάτια, τις εργασίες. Οι εργασίες αυτές κατανέμονται χρονικά σε όλο το έργο και μπορεί κάποιος να τις διαχειρισθεί ανεξάρτητα. Οι εργασίες αποτελούν τα δομικά στοιχεία της διαχείρισης του έργου. Ο υπολογισμός και η κατανομή του χρόνου που θα απαιτήσει κάθε εργασία μπορεί να αναπαρασταθεί οπτικά πάνω σε ένα ημερολόγιο. Αυτό θα αποτελέσει και το πλάνο εργασιών του έργου (project plan). Στον χρονοπρογραμματισμό θα πρέπει πάντα να υπολογίζεται και ο χρόνος έγκρισης του αποτελέσματος των διάφορων φάσεων του έργου από τον πελάτη. Ο χρόνος αυτός είναι συνήθως αρκετά μεγάλος και οδηγεί σε επανασχεδιασμό της εργασίας.

Στάδιο 4: Σχεδιασμός
(Επιλογή δομών δεδομένων, σύνδεση μεταξύ των διαφόρων τμημάτων του συστήματος και ροή ελέγχου μέσα στα τμήματα αυτά).
- Δίνει απάντηση στο ερώτημα του πώς θα δουλεύει το πρόγραμμα.
- Από το σωστό σχεδιασμό εξαρτάται η ευκολία υλοποίησης, ελέγχου και συντήρησης της εφαρμογής.
- Μερικές φορές το στάδιο αυτό περιλαμβάνει και την επιλογή γλώσσας προγραμματισμού ή εργαλείων.

Στάδιο 5: Υλοποίηση
(Ανάπτυξη κώδικα)

- Δεν πρέπει να γράφεται γραμμή κώδικα εάν δεν έχουν λυθεί όλα τα σχεδιαστικά προβλήματα.
- Η στιγμή ανάπτυξης του κώδικα είναι και η στιγμή για να γραφτεί η αντίστοιχη τεκμηρίωση.

Στάδιο 6: Έλεγχος και εγκατάσταση εφαρμογής.
(Απαραίτητο στάδιο πριν την παράδοση του προϊόντος στον πελάτη)
Προτεινόμενη ανάλυση ελέγχων:

1. Προσεκτική ανάγνωση του κώδικα (από κάποιο μέλος της ομάδας ανάπτυξης)
2. Έλεγχος υπορουτινών (κάθε πρόγραμμα ελέγχεται ανεξάρτητα).
3. Έλεγχος ενοτήτων (modules). Κάθε ομάδα υποπρογραμμάτων που ανήκουν σε ένα μεγαλύτερο ανεξάρτητο τμήμα ελέγχεται για σωστή επικοινωνία μεταξύ των υπορουτινών.
4. Έλεγχος υποσυστήματος (Πολλά modules ελέγχονται μαζί).
5. Έλεγχος συστήματος (Όλο το προϊόν ελέγχεται στο σύνολό του για να είναι βέβαιο ότι πληροί τις προδιαγραφές λειτουργίας του).
6. Έλεγχος αποδοχής (Έλεγχος παρουσία του πελάτη με χρήση πραγματικών πλέον δεδομένων).
7. Έλεγχος εγκατάστασης (Επιβεβαιώνεται η λειτουργία του συστήματος στο περιβάλλον που θα δουλεύει).

Στάδιο 7: Συντήρηση
(Ακολουθεί αναπόφευκτα την παράδοση του συστήματος).
Μπορεί να είναι 3 ειδών:

1. Συντήρηση για διόρθωση (λάθη ή παραλείψεις που έχουν διαφύγει από προηγούμενους ελέγχους ή δεν είχαν προβλεφθεί).
2. Συντήρηση για προσαρμογή (για να ανταποκριθεί σε αλλαγές του επιχειρηματικού περιβάλλοντος).
3. Συντήρηση για τελειοποίηση (για να προσφέρει νέες δυνατότητες).

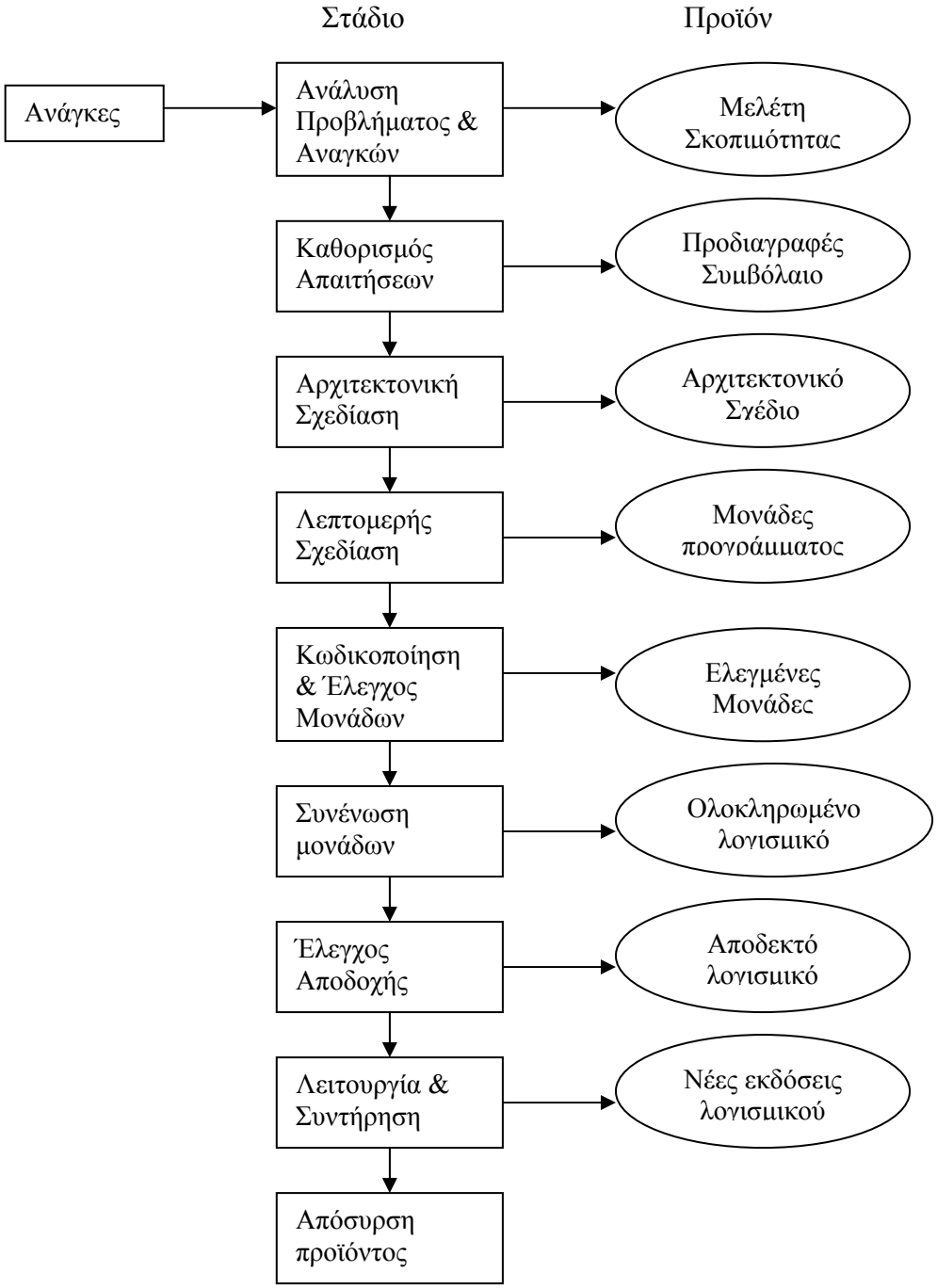
1.2. Γενικές Μέθοδοι Σχεδιασμού

1. Μέθοδος bottom-up
   - Αποφασίζουμε ποιες είναι οι μικρότερες μονάδες που αποτελούν το πρόγραμμα και το συνθέτουμε προχωρώντας προς τα ανώτερα επίπεδα του ιεραρχικού διαγράμματος HIPO.
   - Συρραφή της εφαρμογής με βάση τα κομμάτια του κώδικα που έχουν γραφτεί.
2. Μέθοδος Top-Down
   Παίρνουμε το πρόβλημα σαν ένα γενικό αφηρημένο θέμα και προχωρούμε σιγά σιγά στις λεπτομέρειες κάνοντας ένα βήμα την φορά.

1.3 Μοντέλα Ανάπτυξης Λογισμικού

1. *Μοντέλο Καταρράκτη*

Παρέχει ένα σύνολο εργασιών που πρέπει να εκτελεσθούν ιεραρχικά. Κάθε στάδιο-ενέργεια μας δίνει ένα «προϊόν»-παραδοτέο (artifact). Κάθε φάση ολοκληρώνεται και μετά «παγώνει» (δεν μπορεί να τροποποιηθεί το προϊόν της).



2. *Μοντέλο πρωτοτυποποίησης (rapid prototyping)*

Ετοιμάζεται ένα «πρωτότυπο» (prototype) δηλαδή ένα δείγμα-demo της εφαρμογής που περιέχει τις πιο κρίσιμες λειτουργίες του συστήματος. Δίνεται στους χρήστες για δοκιμή, γίνονται παρατηρήσεις και βελτιώσεις που υιοθετούνται στην ανάπτυξη του συστήματος, ξαναδίνεται στους χρήστες κ.ο.κ. μέχρι να ικανοποιηθούν οι προδιαγραφές του συστήματος.

1.4. ΑΠΑΙΤΗΣΕΙΣ

Οι απαιτήσεις είναι μια περιγραφή του τι μπορεί να κάνει το σύστημα ώστε να ικανοποιεί τον σκοπό για τον οποίο αναπτύσσεται.

Οι απαιτήσεις διακρίνονται σε Λειτουργικές & Μη λειτουργικές Απαιτήσεις.

1. *Λειτουργικές απαιτήσεις*

Περιγράφουν την συμπεριφορά του συστήματος λογισμικού σε κάποια δεδομένα εισόδου. Είναι ανεξάρτητες από την υλοποίηση του συστήματος (Σ). Π.χ. να μπορεί το (Σ) να βγάζει μισθοδοσία κάθε μέρα/εβδομάδα/μήνα.

2. *Μη Λειτουργικές απαιτήσεις*

Περιορισμοί που τίθενται κατά την ανάπτυξη του λογισμικού και περιορίζουν τις επιλογές λύσης. Π.χ. σε ποια γλώσσα θα αναπτυχθεί η εφαρμογή, να τρέχει η εφαρμογή σε UNIX κλπ.

Στάδια Προσδιορισμού απαιτήσεων Λογισμικού

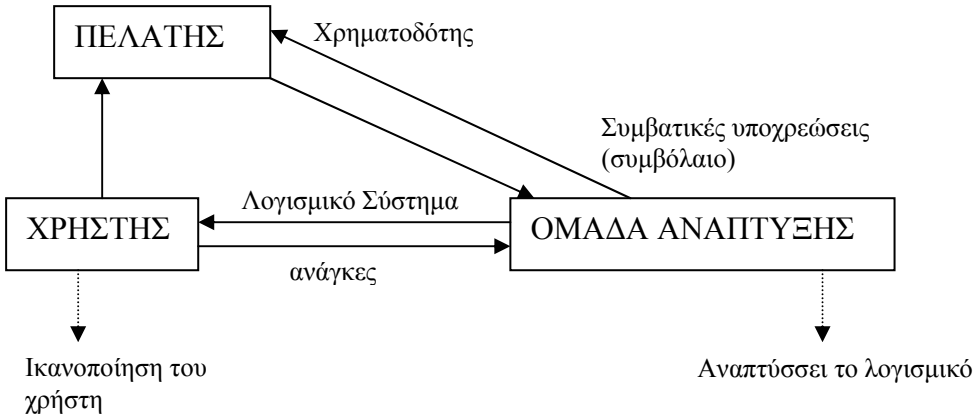
1ο βήμα: Διατύπωση από τον πελάτη του προβλήματος σε φυσική γλώσσα (Ανάλυση Συστήματος)

2ο βήμα: Πλάνο έργου (οργανώνεται το έργο ανάπτυξης)

3ο βήμα: Προσδιορισμός των απαιτήσεων του συστήματος (τι θα κάνει, καταμερισμός λειτουργιών κλπ.)

4ο βήμα: Προσδιορισμός απαιτήσεων S/W.

- Ανάλυση απαιτήσεων S/W
- Προσδιορισμός απαιτήσεων S/W
- Τεκμηρίωση απαιτήσεων S/W
- Επανεξέταση / Επικύρωση απαιτήσεων
- Σχεδίαση S/W



2. Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον

Η επεξεργασία και διαχείριση αρχείων αποτελεί το κεντρικό και βασικό στοιχείο τόσο στον προγραμματισμό συστημάτων όσο και στον προγραμματισμό των εφαρμογών. Με τον όρο επεξεργασία αρχείων εννοούμε ένα ή περισσότερα αρχεία με μια σειρά προγραμμάτων που διαχειρίζονται αυτά τα αρχεία. Σε αντίθεση με τα συστήματα αρχείων τα συστήματα βάσεων δεδομένων είναι πολύπλοκα συστήματα με ολοκληρωμένη υποστήριξη για πολλές δραστηριότητες επεξεργασίας, όπως για παράδειγμα επεξεργασία ερωτήσεων (query processing), παραγωγή αναφορών, εισαγωγή στοιχείων κτλ. Η βασική διαφορά μεταξύ των όρων οργάνωση αρχείων και βάσεων δεδομένων είναι ότι ο όρος οργάνωση αρχείων χρησιμοποιείται συνήθως για να δηλώσει την διαχείριση πληροφοριών στο φυσικό επίπεδο, δηλαδή τον τρόπο αποθήκευσης των πληροφοριών στα μέσα, ενώ ο όρος βάση δεδομένων χρησιμοποιείται στο λογικό επίπεδο το οποίο ουσιαστικά αποτελεί μια αφαίρεση (abstraction) του τρόπου αποθήκευσης των πληροφοριών.

Η κλασική προσέγγιση ανάπτυξης εφαρμογών ήταν προσανατολισμένη προς τις ανάγκες των προγραμμάτων και στηριζόταν στην δημιουργία αρχείων που βρίσκονταν σε εξάρτηση και από την γλώσσα προγραμματισμού και από το ίδιο το πρόγραμμα. Η μια εφαρμογή ήταν ανεξάρτητη από την άλλη, χωρίς μεταξύ τους επικοινωνία, και καθεμία χρησιμοποιούσε τα δικά της αρχεία που είχαν διαφορετικές προδιαγραφές (μήκος και πεδία εγγραφών), ανάλογα με τις δικές της απαιτήσεις.

Η σύγχρονη προσέγγιση ανάπτυξης εφαρμογών όμως είναι προσανατολισμένη στα δεδομένα και στηρίζεται στην χρήση των συστημάτων βάσεων δεδομένων. Όλες οι εφαρμογές υλοποιούνται και λειτουργούν μέσω ενός συστήματος βάσεων δεδομένων. Τα δεδομένα αντιμετωπίζονται ως αυθύπαρκτες οντότητες, και υπάρχει προτυποποίησή τους μέσα σε μια εφαρμογή. Κάθε τύπος εγγραφής ορίζεται μόνο μια φορά και στην συνέχεια απλώς καλείται κατά την εκτέλεση του προγράμματος της εφαρμογής που τον χρειάζεται. Όλα τα προγράμματα χρησιμοποιούν τώρα τα ίδια δεδομένα από την βάση, χωρίς να υπάρχουν διπλότυπες εγγραφές σε πολλά αρχεία.

2.1 Δομή ενός Αρχείου Δεδομένων και λειτουργίες σε αυτό

Αρχείο είναι μια συλλογή από δεδομένα που σχετίζονται μεταξύ τους και βρίσκονται αποθηκευμένα μέσα σε ένα μέσο αποθήκευσης. Η χρήση των αρχείων είναι αναπόφευκτη για μια εμπορική εφαρμογή, οι αναφορές όμως σε αυτά θα πρέπει να είναι όσο το δυνατόν λιγότερες. Για το λόγο αυτό η επιλογή της δομής ή οργάνωσης ενός αρχείου η οποία καθορίζει και την επίδοση των αλγορίθμων επεξεργασίας του, αποτελεί την πιο δύσκολη και κρίσιμη απόφαση του αναλυτή ενός προγράμματος εμπορικής εφαρμογής.

Φυσικό Αρχείο είναι μια συλλογή από ψηφιοσυλλαβές (bytes) πάνω σε ένα μέσο αποθήκευσης.

Λογικό Αρχείο είναι μια συλλογή από συσχετιζόμενες πληροφορίες όπως αυτές φαίνονται μέσα από ένα πρόγραμμα. Το λειτουργικό σύστημα αναλαμβάνει να κάνει την σύνδεση μεταξύ λογικού και φυσικού αρχείου με κατάλληλες εντολές που δίνονται μέσα από το πρόγραμμα. Η χρήση λογικών αρχείων επιτρέπει στο πρόγραμμα να χρησιμοποιήσει διάφορες πράξεις (λειτουργίες) για την επεξεργασία του αρχείου χωρίς καμιά άλλη γνώση του αντίστοιχου φυσικού αρχείου. Το πρόγραμμα μπορεί έτσι να επεξεργαστεί διάφορα αρχεία τα οποία όμως έχουν την ίδια ακριβώς δομή.

Κάθε ένα από τα στοιχεία που αποτελούν ένα αρχείο χαρακτηρίζεται από τα εξής γνωρίσματα: μια μονάδα (π.χ. εργαζόμενος), ένα πλήθος ιδιοτήτων της μονάδας αυτής (π.χ. Αριθμός Μητρώου, Ονοματεπώνυμο, Διεύθυνση, Μισθός κτλ.) και ένα πλήθος τιμών για τις ιδιότητες αυτές (π.χ. 1132, Κίμων Καραϊσκάκης, Αχαρνών 23, 1000 €). Κάθε στοιχείο μονάδα μαζί με τις ιδιότητες που την χαρακτηρίζουν αποτελεί μια εγγραφή (record). Κάθε μια από τις ιδιότητες της εγγραφής αναφέρεται και ως πεδίο της εγγραφής. Το πεδίο μιας εγγραφής είναι η μικρότερη λογική μονάδα με νόημα σε ένα αρχείο.

Συνήθως σε ένα αρχείο οι εγγραφές είναι όλες του ίδιου τύπου, δηλαδή έχουν τα ίδια πεδία, την ίδια διάταξη πεδίων και το ίδιο μήκος. Τέτοιες εγγραφές ονομάζονται *εγγραφές σταθερού μήκους* (fixed length records). Γενικά μια εγγραφή χαρακτηρίζεται σταθερού μήκους είτε με βάση το μήκος της σε bytes είτε με βάση τα πεδία της. Το ότι όλες οι εγγραφές ενός αρχείου έχουν το ίδιο μήκος δεν σημαίνει ότι και τα μεγέθη ή το πλήθος των πεδίων τους είναι πάντα σταθερά. Σε πολλές εφαρμογές το μέγεθος των εγγραφών είναι *μεταβλητό* (variable length records). Αυτό οφείλεται σε διάφορους λόγους, είτε γιατί τα πεδία των εγγραφών είναι μεταβλητού μήκους (ονοματεπώνυμα), είτε γιατί υπάρχουν διαφορετικού τύποι πεδίων (ωρομίσθιοι και μόνιμοι υπάλληλοι μιας επιχείρησης) είτε τέλος γιατί σε μια εγγραφή υπάρχουν επαναλαμβανόμενα πεδία (πληροφορίες για τα παιδιά κάθε υπαλλήλου).

Το πεδίο που χρησιμοποιούμε για να κάνουμε επιλογή (selection) μιας εγγραφής ή για να ταξινομήσουμε (sort) ένα αρχείο λέγεται <u>*κλειδί*</u> (key). Θεωρητικά όλα τα πεδία μιας εγγραφής μπορούν να χρησιμοποιηθούν ως κλειδιά. Συνήθως ένα από τα κλειδιά χαρακτηρίζεται ως πρωτεύον κλειδί (primary key). Αυτό σημαίνει ότι το κλειδί αυτό μπορεί να χρησιμοποιηθεί για την μονοσήμαντη επιλογή μιας εγγραφής από το αρχείο (ξεχωρίζει την εγγραφή αυτή από όλες τις άλλες), π.χ. σε ένα αρχείο μισθοδοσίας ο αριθμός μητρώου του ΙΚΑ μπορεί να αποτελέσει το πρωτεύον κλειδί. Όλα τα άλλα κλειδιά λέγονται δευτερεύοντα.

<u>*Λειτουργίες – Επεξεργασία αρχείων σε Εμπορικές Εφαρμογές*</u>

- Δημιουργία Αρχείου
- Εισαγωγή Εγγραφής σε αρχείο
- Διαγραφή Εγγραφής από αρχείο
- Τροποποίηση Εγγραφής σε αρχείο
- Ανάγνωση (Αναζήτηση) Εγγραφής σε αρχείο
- Εκτύπωση Εγγραφών αρχείου

Ένα *αρχείο κειμένου* (text file) αποτελείται από ψηφιοσυλλαβές (bytes) όπου κάθε ψηφιοσυλλαβή αντιστοιχεί σε ένα χαρακτήρα από το σύνολο των χαρακτήρων του συστήματος. Για τους περισσότερους υπολογιστές το σύνολο των χαρακτήρων είναι το ASCII. Πολλές φορές λέμε ότι ένα αρχείο κειμένου είναι εκείνο το αρχείο που περιέχει μόνο εκτυπώσιμους χαρακτήρες συν ορισμένους άλλους χαρακτήρες (whitespace characters) όπως είναι οι χαρακτήρες: newline, space, tab, backspace.
Ένα *δυαδικό αρχείο* (binary file) αποτελείται από ψηφιοσυλλαβές που μπορεί να αντιστοιχούν σε ένα χαρακτήρα ή σε μια αριθμητική τιμή ή σε ένα χαρακτήρα ελέγχου (control character). Το εκτελέσιμο αρχείο που δημιουργείται από τη μεταγλώττιση ενός προγράμματος είναι χαρακτηριστικό παράδειγμα ενός δυαδικού αρχείου.

Ένα αρχείο κειμένου μπορεί να επεξεργασθεί από έναν επεξεργαστή κειμένου, να εκτυπωθεί κανονικά μέσω μιας απλής εντολής (όπως print), και είναι αναγνώσιμο

από τον άνθρωπο (human-readable) ενώ το ίδιο δεν συμβαίνει για τα δυαδικά αρχεία. Οι περισσότερες γλώσσες προγραμματισμού παρέχουν και τους δύο τύπους αρχείων.

2.3. Συσχετίσεις Αρχείων Δεδομένων

Οι συσχετίσεις μεταξύ των αρχείων δεδομένων γίνονται μέσω των πρωτευόντων και δευτερευόντων (εξωτερικών) πεδίων-κλειδιών τους.
Στο παράδειγμα του φροντιστηρίου:
Συσχέτιση Αρχείων Μαθημάτων-Δηλώσεων: ως προς το πεδίο «Κωδικός Μαθήματος».
Συσχέτιση Αρχείων Σπουδαστών- Δηλώσεων: ως προς το πεδίο «Κωδικός Φοιτητή».

2.4. Οργάνωση αρχείων

Οι συνήθεις μέθοδοι οργάνωσης αρχείων είναι οι εξής:

1. *Σειριακή ή Ακολουθιακή (Sequential).*
   Είναι η απλούστερη μέθοδος οργάνωσης ενός αρχείου και χρησιμοποιείται όταν δεν μας ενδιαφέρει η σειρά (διάταξη) με την οποία θα επεξεργαστούν οι εγγραφές του αρχείου. Οι εγγραφές (records) γράφονται στο αρχείο αλλά και επεξεργάζονται ακολουθιακά, με τη σειρά η μία μετά την άλλη. Προσθήκες εγγραφών σε ένα τέτοιο αρχείο γίνονται μόνο στο τέλος του. Αν χρειασθεί να γίνουν προσθήκες εγγραφών οπουδήποτε αλλού ή μεταβολές που επηρεάζουν το μήκος τους, τότε απαιτείται η επαναγραφή όλου του αρχείου.
2. *Άμεση (Direct ή Relative).*
   Μια οργάνωση αρχείου άμεσης προσπέλασης επιτρέπει την άμεση προσπέλαση στη ζητούμενη εγγραφή. Τα διάφορα στοιχεία ανακαλούνται ή απ' ευθείας, αν είναι γνωστές οι διευθύνσεις τους ή μέσω των κλειδιών τους. Συνήθως το κλειδί μιας εγγραφής με την βοήθεια κάποιου αλγόριθμου μετατρέπεται στην ακριβή διεύθυνση της εγγραφής στον δίσκο, ή στην διεύθυνση ενός μεγαλύτερου χώρου που περιέχει την εγγραφή και στην συνέχεια με σειριακή αναζήτηση, η εγγραφή ανευρίσκεται. Υπάρχει δηλαδή μια συνάρτηση η οποία δέχεται σαν όρισμα το κλειδί της εγγραφής (την λογική της διεύθυνση) και επιστρέφει την πραγματική (φυσική) διεύθυνση της στο δίσκο.
3. *Σειριακή με δείκτες.*
   Η σειριακή οργάνωση με δείκτες επιτρέπει την άμεση και σειριακή προσπέλαση των εγγραφών ενός αρχείου. Δίνεται η δυνατότητα στον προγραμματιστή να προσαρτήσει, να ενθέσει αλλά και να διαγράψει εγγραφές διατηρώντας την σειριακή δομή του αρχείου. Παρέχεται επίσης η δυνατότητα επιλογής της άμεσης ή σειριακής προσπέλασης. Υπάρχει ουσιαστικά ένα ευρετήριο (index) με δείκτες οι οποίοι αντιστοιχίζουν τα πρωτεύοντα κλειδιά των ορισμένων εγγραφών-ορόσημων (π.χ. ανά 10 εγγραφές) στις φυσικές θέσεις τους στο δίσκο. Για τις επόμενες εγγραφές η αναζήτηση γίνεται σειριακά.

2.5. Βασικές αρχές ανάπτυξης μιας Εμπορικής Εφαρμογής χρησιμοποιώντας στοιχεία Τεχνολογίας Λογισμικού

Βήμα 1ο: Απόφαση για τις απαιτήσεις και την γραμμογράφηση των αρχείων της εφαρμογής.

Π.χ. Εφαρμογή διαχείρισης φροντιστηρίου

Απαιτήσεις: διαχείριση σπουδαστών
διαχείριση καθηγητών
διαχείριση μαθημάτων
διαχείριση δηλώσεων μαθημάτων
διαχείριση απουσιών
διαχείριση βαθμών

Γραμμογράφηση αρχείων: Αποφασίζουμε ποια πεδία θα περιλαμβάνει κάθε αρχείο δεδομένων της εφαρμογής. (Προσοχή! Κατά την τήρηση των αρχείων δεν πρέπει τα ίδια στοιχεία να κρατούνται άνω της μίας φοράς).

αρχείο σπουδαστών	Κωδικός σπουδαστή
	Ονοματεπώνυμο
	Τάξη
	Διεύθυνση
	Τηλέφωνο

αρχείο καθηγητών	Κωδικός καθηγητή
	Ονοματεπώνυμο
	Διεύθυνση
	Τηλέφωνο

αρχείο μαθημάτων	Κωδικός μαθήματος
	Τίτλος
	Κωδικός καθηγητή

αρχείο δηλώσεων μαθημάτων	Κωδικός μαθήματος
	Κωδικός σπουδαστή

αρχείο απουσιών	Κωδικός σπουδαστή
	Αριθμός απουσιών

αρχείο βαθμών	Κωδικός μαθήματος
	Κωδικός σπουδαστή
	Βαθμός

Βήμα 2ο : Απόφαση για την υλοποίηση των πράξεων διαχείρισης στα αρχεία

Δυνατές Επιλογές:

A. Δήλωση όλων των υποπρογραμμάτων διαχείρισης στο κυρίως πρόγραμμα που είναι δομημένο με βάση τις αρχές του δομημένου προγραμματισμού (top-down design, bottom-up υλοποίηση). (Για το προηγούμενο παράδειγμα 24 υποπρογράμματα, 6 αρχεία * 4 πράξεις για το καθένα)
B. Δήλωση των υποπρογραμμάτων διαχείρισης που αφορούν κάθε αρχείο σε ξεχωριστά προγράμματα διαχείρισης του καθενός και κατασκευή ενός κύριου προγράμματος που καλεί τα υποπρογράμματα αυτά. (7 προγράμματα για το προηγούμενο παράδειγμα).
C. Δήλωση όλων των υποπρογραμμάτων διαχείρισης σε μια βιβλιοθήκη προγραμμάτων και σύνδεση της βιβλιοθήκης με το κυρίως πρόγραμμα μέσω εντολών της γλώσσας προγραμματισμού.
D. Ως επέκταση του (C) και εφόσον το επιτρέπει η γλώσσα προγραμματισμού, στην βιβλιοθήκη μπορούν να υπάρχουν υποπρογράμματα γενικού σκοπού που

να λειτουργούν ανεξαρτήτως της γραμμογράφησης των αρχείων. (4 υποπρογράμματα, ένα για κάθε πράξη σε αρχείο).

Βήμα 3ο: Απόφαση για την σχεδίαση και την υλοποίηση των διεπαφών με το χρήστη.
Δυνατές Επιλογές:

A. Δήλωση τόσων υποπρογραμμάτων όσα τα αρχεία και οι καταστάσεις για την κατασκευή οθονών με εντολές της γλώσσας προγραμματισμού.
B. Κατασκευή βιβλιοθήκης με παράθυρα στη γλώσσα ανάπτυξης της εμπορικής εφαρμογής ή σε άλλη γλώσσα και σύνδεση με το κυρίως πρόγραμμα.
C. Κατασκευή οθονών μέσω χρήσης προγραμμάτων Screen Generator.

Βήμα 4ο : Απόφαση για την σχεδίαση και την υλοποίηση των εκτυπωτικών καταστάσεων.
Δυνατές Επιλογές:

A. Δήλωση στο κυρίως πρόγραμμα τόσων διαδικασιών όσες και οι εκτυπωτικές καταστάσεις.
B. Κατασκευή βιβλιοθήκης για την δημιουργία καταστάσεων.
C. Χρήση εργαλείου Report Generator αν είναι διαθέσιμο από την γλώσσα προγραμματισμού για εύκολη κατασκευή καταστάσεων.

Βήμα 5ο: Σχεδιασμός του συστήματος ασφαλείας της εφαρμογής και των χρηστών.
Δυνατές Επιλογές:

A. Δήλωση κωδικών ασφαλείας με την μορφή σταθερών τιμών μέσα στο κυρίως πρόγραμμα και έλεγχος των προσπελάσεων στις επιλογές της εφαρμογής μέσω εντολών ελέγχου και ροής (υποτυπώδης παροχή ασφάλειας)
B. Σχεδίαση προγράμματος δημιουργίας και συντήρησης του συστήματος ασφαλείας. Κατασκευή αρχείου στο δίσκο με περιεχόμενα τα κωδικοποιημένα αρχικά passwords. Το καλύτερο κλείδωμα γίνεται με ενσωμάτωση στο σύστημα ασφάλειας διαδικασιών σε γλώσσα προσανατολισμένη προς την μηχανή.

Βήμα 6ο: Σχεδιασμός του συστήματος μεταφερσιμότητας της εφαρμογής.

2.6 ΔΟΜΗ ΕΦΑΡΜΟΓΗΣ

Μπορεί να αποτελείται από:

1. Διαδικασίες (procedures) και συναρτήσεις (functions)
2. Ανεξάρτητες ενότητες-προγράμματα (modules) που καλούνται από ένα γενικό κύριο πρόγραμμα.

Οι εμπορικές εφαρμογές αποτελούνται από ανεξάρτητες προγραμματιστικές ενότητες.
Παράδειγμα:
Ενότητα 1: Διαχείριση Αρχείων
Ενότητα 2: Έλεγχος ορθότητας εισαγωγής δεδομένων στην εφαρμογή
Ενότητα 3: Διαχείριση οθονών
Ενότητα 4: Βοηθητικά προγράμματα
Ενότητα 5: Εξασφάλιση μεταφερσιμότητας εφαρμογής