ΤΕΙ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ

ΤΜΗΜΑ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ

ΠΜΣ Διοίκηση Εκπαίδευσης

Τεχνολογία Πληροφοριακών Συστημάτων – Μεθοδολογίες, Μοντέλα, Μοντελοποίηση Συστημάτων (Μέρος 2ο)

Δρ Χρήστος Πιερρακέας

Ανάλυση Απαιτήσεων

Σκοπός του μαθήματος

Σκοπός του μαθήματος είναι

- είναι ο ορισμός της έννοιας της «απαίτησης από το πληροφοριακό σύστημα / λογισμικό» και
- η παρουσίαση διαδικασιών προσδιορισμού και μοντελοποίησης αυτών.

Απαίτηση από το λογισμικό

- Μια απαίτηση από το λογισμικό είναι μια λειτουργία που αυτό θα πρέπει να επιτελεί ή μια συνθήκη που θα πρέπει να ικανοποιεί, όταν θα έχει ολοκληρωθεί η κατασκευή του.
- Συνήθως ο πελάτης εκφράζει τις απαιτήσεις του με μεγάλο βαθμό γενικότητας, ενώ ο κατασκευαστής τις διατυπώνει με μεγαλύτερη λεπτομέρεια και σαφήνεια, ώστε να μπορεί να κάνει τη δουλειά του.

Απαίτηση από το λογισμικό

Υπάρχουν τρεις κατηγορίες απαιτήσεων:

- αυτές που πρέπει να τηρηθούν
- αυτές που είναι εξαιρετικά επιθυμητές
- αυτές που μπορούν και να αποκλειστούν

Οι απαιτήσεις καθορίζουν το ***τι*** πρέπει να κάνει το σύστημα και όχι το ***πως***

Ερώτημα - Άσκηση

Μία απαίτηση από τον πελάτη καταγράφει χαρακτηριστικά ότι:

Αν ο πελάτης έχει ηλικία μικρότερη των 18 ετών και παίζει τένις, να του αποστέλλεται επιστολή σχετική με προϊόντα τένις. Αν ο πελάτης έχει ηλικία 18 ή μεγαλύτερη ή έχει άδεια οδήγησης αυτοκινήτου, να του αποστέλλεται επιστολή σχετικά με προϊόντα αυτοκινήτου. Αν συμβαίνουν και τα δυο, να προστεθεί το όνομα του στην λίστα με τα γενικά προϊόντα αθλητισμού.

Παρατηρείτε κάποιο πρόβλημα;

Απάντηση

Η απαίτηση είναι ΛΑΘΟΣ γιατί δεν μπορούν ταυτόχρονα να συμβαίνουν τα 2 πρώτα οπότε η τρίτη εκδοχή δεν πραγματοποιείται ποτέ.

Απαιτείται διευκρίνιση από τον πελάτη.

Ταξινόμηση απαιτήσεων λογισμικού

Οι απαιτήσεις από το λογισμικό διακρίνονται σε δύο μεγάλες κατηγορίες. Στις **«λειτουργικές»** και στις **«μη λειτουργικές»**.

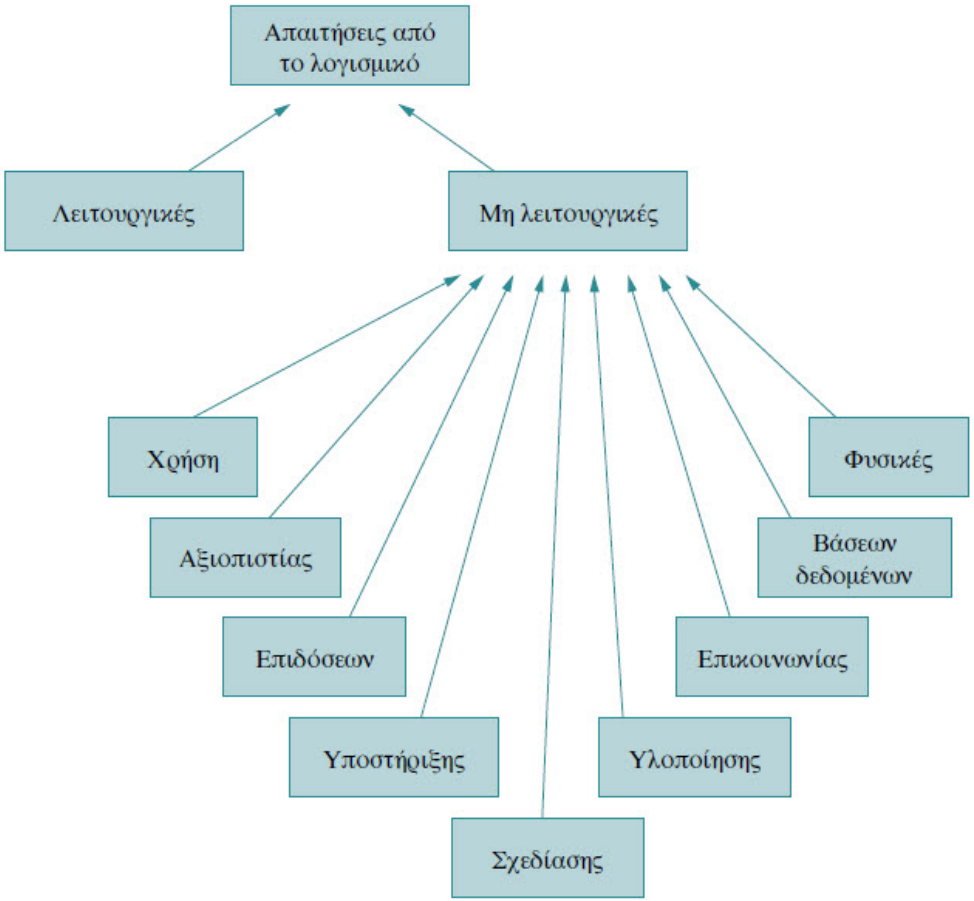
Λειτουργικές απαιτήσεις

Οι λειτουργικές απαιτήσεις περιγράφουν τις εργασίες (λειτουργίες) που θα πρέπει να εκτελεί το λογισμικό.

Μη λειτουργικές απαιτήσεις

Οι μη λειτουργικές απαιτήσεις περιγράφουν χαρακτηριστικά που πρέπει να έχει το λογισμικό τα οποία δεν αφορούν την εκτέλεση κάποιας λειτουργίας από αυτό.

Ταξινόμηση απαιτήσεων λογισμικού



Μη λειτουργικές απαιτήσεις

Οι μη λειτουργικές απαιτήσεις μπορούν να ταξινομηθούν σε επιμέρους κατηγορίες, οι οποίες αναφέρονται ακολούθως.

Απαιτήσεις χρήσης: Καθορίζουν τα χαρακτηριστικά της χρήσης του συστήματος, την διεπιφάνεια χρήστη (user interface), καθώς και το υλικό τεκμηρίωσης και εκπαίδευσης που θα έχει ο τελικός χρήστης.

Παράδειγμα: Το λογισμικό θα πρέπει να ελέγχεται με τη χρήση του ποντικιού ή του πληκτρολογίου και να συνοδεύεται από αναλυτικό εγχειρίδιο χρήστη και εγχειρίδιο εκμάθησης.

Απαιτήσεις αξιοπιστίας: Καθορίζουν τη συμπεριφορά του λογισμικού σε καταστάσεις ενδογενών ή εξωγενών σφαλμάτων, τη διαδικασία αποκατάστασης, την πρόβλεψη τέτοιων καταστάσεων, καθώς και την επιθυμητή διαθεσιμότητα του λογισμικού.

Παράδειγμα: Σε περίπτωση απρόβλεπτου τερματισμού της λειτουργίας του λογισμικού θα πρέπει να επιχειρείται επανεκκίνηση με την ελάχιστη δυνατή απώλεια δεδομένων για το χρήστη (ύπαρξη auto-save, ή διαδικασία auto-recovery κλπ).

Μη λειτουργικές απαιτήσεις

Απαιτήσεις επιδόσεων: Εισάγουν περιορισμούς σχετικά με το χρόνο εκτέλεσής τους και με τη χρήση πόρων, όπως η μνήμη και οι μονάδες επεξεργασίας.

Παράδειγμα: Ο χρόνος αναζήτησης και ανάκτησης από τη βάση δεδομένων μιας εγγραφής με κλειδί το ονοματεπώνυμο δε θα πρέπει να ξεπερνά το 1 δευτερόλεπτο.

Απαιτήσεις υποστήριξης: Καθορίζουν τα επιθυμητά χαρακτηριστικά για τον έλεγχο και τη συντήρηση του λογισμικού.

Παράδειγμα: Κατά την εγκατάσταση θα πρέπει να καταγράφεται σε αρχείο μη ορατό από το χρήστη η έκδοση όλων των αρχείων που εγκαταστάθηκαν.

Απαιτήσεις σχεδίασης: Καθορίζουν τον τρόπο με τον οποίο θα πρέπει να γίνει η σχεδίαση του λογισμικού.

Μη λειτουργικές απαιτήσεις

Απαιτήσεις υλοποίησης: Καθορίζουν τον τρόπο με τον οποίο θα πρέπει να γίνει η συγγραφή του πηγαίου κώδικα (source code) του λογισμικού.

Παράδειγμα: Θα πρέπει να χρησιμοποιηθεί η τάδε γλώσσα ή ότι η διαθέσιμη μνήμη θα είναι τόση.

Απαιτήσεις επικοινωνίας με άλλα συστήματα - Διαλειτουργικότητα: Καθορίζουν τα εξωτερικά συστήματα, λογισμικού ή άλλα, με τα οποία το λογισμικό θα επικοινωνεί, καθώς και τον τρόπο (λ.χ. πρότυπα, φυσική σύνδεση) πραγματοποίησης της επικοινωνίας αυτής.

Μη λειτουργικές απαιτήσεις

Απαιτήσεις Βάσεων Δεδομένων: Καθορίζουν τις οντότητες που θα διαχειριστεί το πληροφοριακό σύστημα, καθώς και τα πεδία αυτών, όπως αυτά είναι αναγνωρίσιμα στην παρούσα φάση της ανάπτυξης.

Παράδειγμα: Το λογισμικό θα πρέπει να διατηρεί αρχείο πελατών με τα εξής στοιχεία: ονοματεπώνυμο, διεύθυνση, τηλέφωνο, ΑΦΜ.

Φυσικές απαιτήσεις: Καθορίζουν τα επιθυμητά φυσικά χαρακτηριστικά του λογισμικού και του συστήματος.

Παράδειγμα: Καθορισμός λειτουργικού συστήματος και προδιαγραφές υπολογιστή όπου θα τρέχει το λογισμικό, περιγραφή απαιτούμενων δικτυακών συνδέσεων.

Βήματα στον προσδιορισμό απαιτήσεων

1ο Η πρώτη πηγή για τον καθορισμό των απαιτήσεων από μια εφαρμογή λογισμικού είναι, ασφαλώς, ο πελάτης, ο οποίος περιγράφει στον κατασκευαστή τις εργασίες που θεωρεί απαραίτητο να εκτελούνται από το λογισμικό. Η περιγραφή αυτή συνήθως γίνεται με τη μορφή μιας έκθεσης, η οποία ενίοτε δεν είναι πλήρης και περιέχει ασάφειες και διφορούμενα.

2ο Η **ανάλυση των απαιτήσεων,** η οποία στοχεύει στη δημιουργία μοντέλων που περιγράφουν διαφορετικές πλευρές του λογισμικού. Τα μοντέλα αυτά παριστάνονται με τη βοήθεια:

Διαγραμμάτων ροής δεδομένων,

Διαγραμμάτων οντοτήτων – συσχετίσεων και

Διαγραμμάτων μετάβασης καταστάσεων, καθώς και με

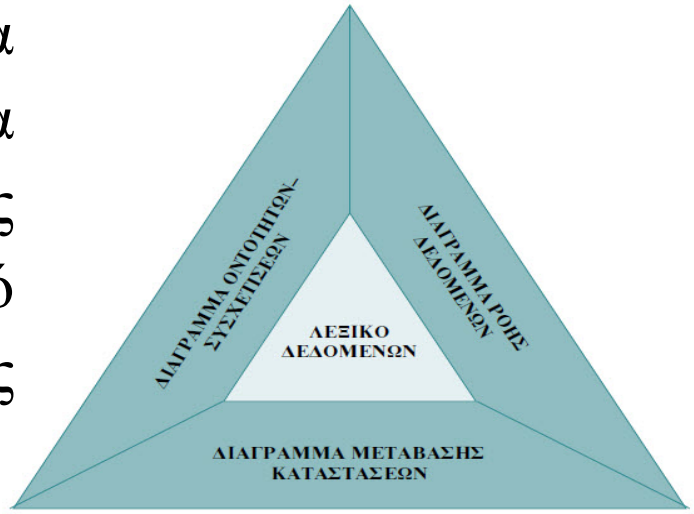
Χρήση ενός λεξικού δεδομένων

Μοντέλο παράστασης λογισμικού

Ένα μοντέλο παράστασης λογισμικού είναι ένα διάγραμμα ή ένα σύνολο από ομοειδή διαγράμματα, το οποίο περιγράφει το λογισμικό από μια συγκεκριμένη οπτική γωνία.

Είναι μια αφαιρετική (abstract) περιγραφή κάποιων επιλεγμένων χαρακτηριστικών του και μόνο.

Υπάρχουν πολλά μοντέλα παράστασης λογισμικού, τα οποία προτείνονται από διαφορετικές μεθοδολογίες ανάπτυξης ή και από διαφορετικές εκδοχές της ίδιας μεθοδολογίας.



Μοντέλο Παράστασης Λογισμικού

ΔΡΔ (ροής δεδομένων**)**

Λογική αναπαράσταση του συστήματος που δείχνει ΠΩΣ τα δεδομένα μετασχηματίζονται από το λογισμικό, αναπαριστώντας τους μετασχηματισμούς/διαδικασίες, τις εισόδους και εξόδους τους.

ΔΟΣ (οντοτήτων – συσχετίσεων**)**

Παράσταση των δεδομένων (ΤΙ δεδομένα επεξεργάζεται το σύστημα) και των συσχετίσεων μεταξύ τους, σύμφωνα με το σχεσιακό μοντέλο δεδομένων (γνωστική περιοχή Βάσεων Δεδομένων)

ΔΜΚ (μετάβασης καταστάσεων**)**

Περιγράφουμε τη δυναμική συμπεριφορά του λογισμικού, τη χρονική σειρά εκτέλεσης (το ΠΟΤΕ) των μετασχηματισμών

Λεξικό Δεδομένων

Αναλυτική περιγραφή όλων των σχετιζόμενων με δεδομένα στοιχείων των μοντέλων παράστασης λογισμικού

Διαγράμματα ροής δεδομένων

Ο κατασκευαστής του λογισμικού, χρειάζεται το **διάγραμμα ροής δεδομένων**, το οποίο περιέχει τις απαιτήσεις του πελάτη με τη μορφή ενός δικτύου στο οποίο «ρέουν» δεδομένα τα οποία μετασχηματίζονται σε νέα δεδομένα από μονάδες λογισμικού.

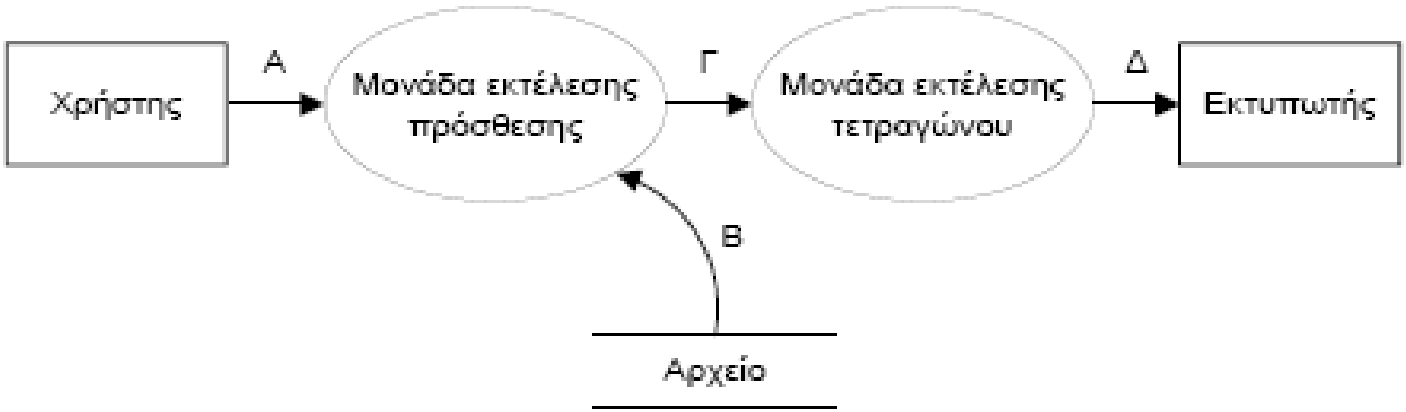
Κάθε **μονάδα λογισμικού θεωρείται ως μετασχηματισμός** που εφαρμόζεται επί κάποιων δεδομένων εισόδου προκειμένου να δημιουργήσει νέα δεδομένα εξόδου.

Τα διαγράμματα ροής δεδομένων αποτελούν τη βάση για αρκετά από τα επόμενα βήματα της ανάπτυξης λογισμικού.

Διαγράμματα ροής δεδομένων

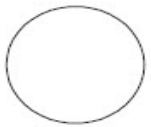
Στο Σχήμα φαίνεται ένα διάγραμμα ροής δεδομένων το οποίο περιγράφει την υλοποίηση της αριθμητικής πράξης $(A+B)^2$, όπου Α, Β πραγματικοί αριθμοί, εκ των οποίων ο Α δίνεται από το χρήστη και ο Β διαβάζεται από κάποια αποθήκη δεδομένων (αρχείο).

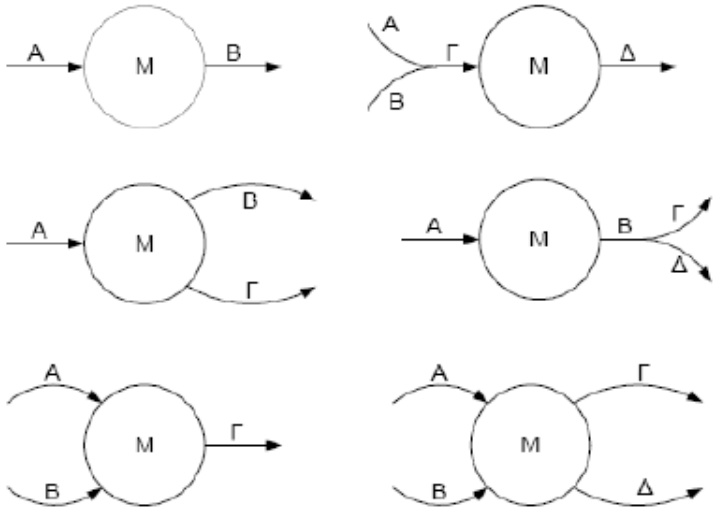
Η «μονάδα εκτέλεσης πρόσθεσης» διαβάζει τους δύο αριθμούς (δεδομένα) και δημιουργεί έναν νέο, ο οποίος αντιστοιχεί στο άθροισμά τους: Γ = Α+Β. Αυτό εισάγεται στη «μονάδα εκτέλεσης τετραγώνου», η οποία δημιουργεί το δεδομένο Δ, που αντιστοιχεί στην ύψωση του Γ στο τετράγωνο: $\Delta = \Gamma^2 = (A+B)^2$. Το δεδομένο αυτό στέλνεται στον εκτυπωτή.



Διαγράμματα ροής δεδομένων

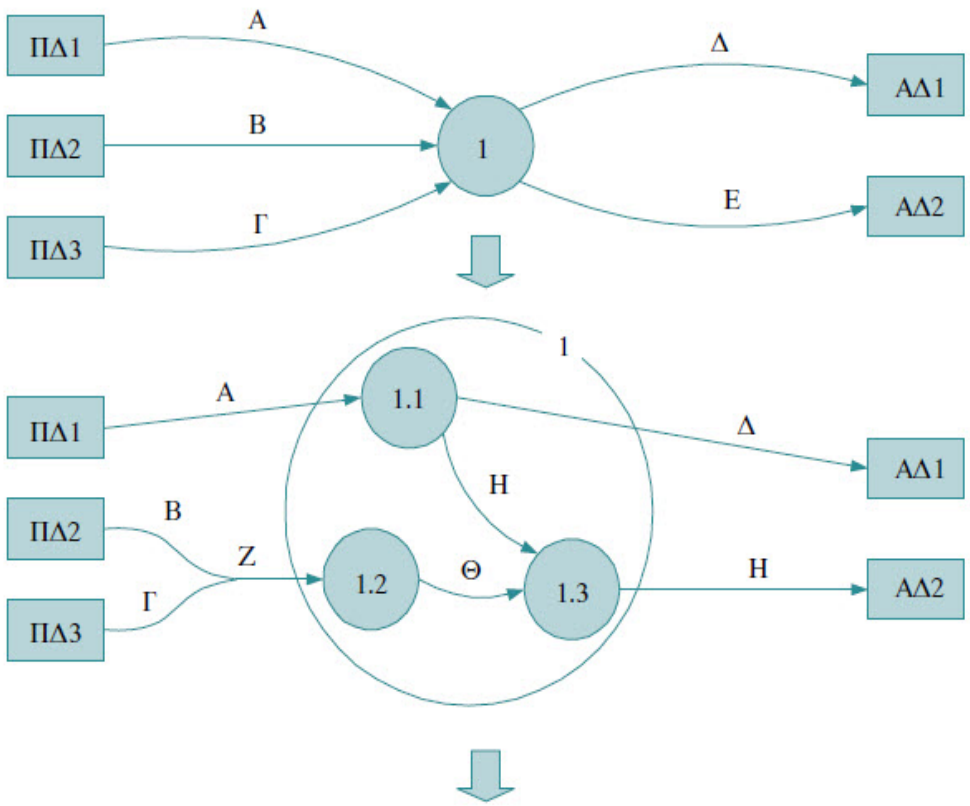
Κάθε υπολογιστική μονάδα-μετασχηματισμός δεδομένων παριστάνεται με έναν κύκλο (για ευκολία στη σχεδίαση χρησιμοποιείται και το ελλειψοειδές σχήμα). Κάθε παραγωγός ή αποδέκτης δεδομένων παριστάνεται με ένα παραλληλόγραμμο. Ως **αποδέκτες ή πηγές** / παραγωγοί δεδομένων νοούνται οντότητες εξωτερικές προς το σύστημα λογισμικού, όπως ο άνθρωπος (χρήστης), ένας εκτυπωτής ή ένα ανεξάρτητο σύστημα λογισμικού. Μια ροή δεδομένων περιγράφεται με ένα βέλος που ενώνει μετασχηματισμούς μεταξύ τους ή με αποδέκτες/παραγωγούς δεδομένων. Πάνω από το βέλος σημειώνεται μια ονομασία για τα δεδομένα που αφορά η ροή.

Συμβολισμοί διαγραμμάτων ροής δεδομένων	
	Διαδικασία/ μετασχηματισμός δεδομένων
	Εξωτερική πηγή ή αποδέκτης δεδομένων
	Ροή δεδομένων
	Αποθήκη δεδομένων



Διαγράμματα ροής δεδομένων – Διαδοχικά επίπεδα λεπτομέρειας

Το διάγραμμα ροής δεδομένων παριστάνεται σε διαφορετικά επίπεδα λεπτομέρειας. Στο πρώτο, λιγότερο λεπτομερές επίπεδο, ολόκληρη η εφαρμογή λογισμικού παριστάνεται ως ένας μετασχηματισμός σύνθετων δεδομένων.



Μεθοδολογία Σχεδίασης ΔΡΔ

Βήμα 1 Πηγές και Αποδέκτες

Διαβάζουμε την εκφώνηση προσεκτικά υπογραμμίζοντας τις πηγές και τους αποδέκτες.

Πηγές είναι οντότητες οι οποίες εισάγουν δεδομένα στο σύστημα

Π.χ. Παρέχει τη δυνατότητα σε άτομα που αναζητούν εργασία (υποψηφίους) να υποβάλλουν το βιογραφικό τους.

Αποδέκτες είναι οι οντότητες οι οποίες λαμβάνουν δεδομένα από το σύστημα

Έμμεση διατύπωση:

Π.χ. Ο υποψήφιος πρέπει να καταχωρίσει τα στοιχεία του (Ονοματεπώνυμο και Διεύθυνση Ηλ. Ταχυδρομείου). Το σύστημα απαντά με ένα Κωδικό Χρήστη .....

Άμεση διατύπωση:

Π.χ. Το Τμήμα Συμβούλων της Χ λαμβάνει για κάθε μια θέση εργασίας που είναι καταχωρημένη στο σύστημα μια λίστα υποψηφίων .....

Μεθοδολογία Σχεδίασης ΔΡΔ

Βήμα 2 Διακρίνουμε δεδομένα από τις πηγές

Τα υπογραμμίζουμε

Π.χ. Ονοματεπώνυμο, Διεύθυνση Ηλ. Ταχυδρομείου, Password, Περιγραφές θέσεων εργασίας.

Μάθετε να ανακαλύπτετε και τις ασαφείς διατυπώσεις:

Το τμήμα data entry (εισόδου δεδομένων) της Χ καταχωρεί με τη βοήθεια του συστήματος **περιγραφές θέσεων εργασίας** μετά από επικοινωνία με εργοδότες. Το σύστημα επίσης αποθηκεύει στοιχεία επικοινωνίας για τους εργοδότες.

Μεθοδολογία Σχεδίασης ΔΡΔ

Βήμα 3 Διακρίνουμε δεδομένα προς Αποδέκτες

Τα υπογραμμίζουμε

πχ. Το σύστημα απαντά με Κωδικό Χρήστη

Το σύστημα δείχνει τα αποτελέσματα της αναζήτησης

Διαφορετικά βγάζει μήνυμα έλλειψης αποτελεσμάτων.

Εμφανίζεται Λίστα υποψηφίων

Βήμα 4 Σχεδιασμός 0/κού επιπέδου (Context Diagram)

Φαίνονται όλες οι πηγές, αποδέκτες

Περιληπτικά (σε σχέση με το επίπεδο 1) αν γίνεται επιγράφουμε τις ροές από και προς τις πηγές και αποδέκτες αντίστοιχα

Μεθοδολογία Σχεδίασης ΔΡΔ

Μεθοδολογία Σχεδίασης ΔΡΔ

Βήμα 5 Διακρίνουμε τις βασικές λειτουργίες ή ομάδες λειτουργιών για το 1ο επίπεδο

Πιθανές περιπτώσεις

Διαχωρισμός ομάδων ανάλογα με τις πηγές ή αποδέκτες

- Υπηρεσίες υποψηφίων
- Υπηρεσίες τμήματος Εισόδου Δεδομένων
- Υπηρεσίες τμήματος Συμβούλων

Διαχωρισμός βάσει της εκφώνησης

Πιο έντεχνος διαχωρισμός βάσει της εκφώνησης και της δικής μας ομαδοποίησης.

Μεθοδολογία Σχεδίασης ΔΡΔ

Βήμα 6 Διακρίνουμε τις αποθήκες δεδομένων

Πχ.

Και τα στοιχεία του υποψηφίου αποθηκεύονται σε σχετικό αρχείο που διατηρεί το λογισμικό.

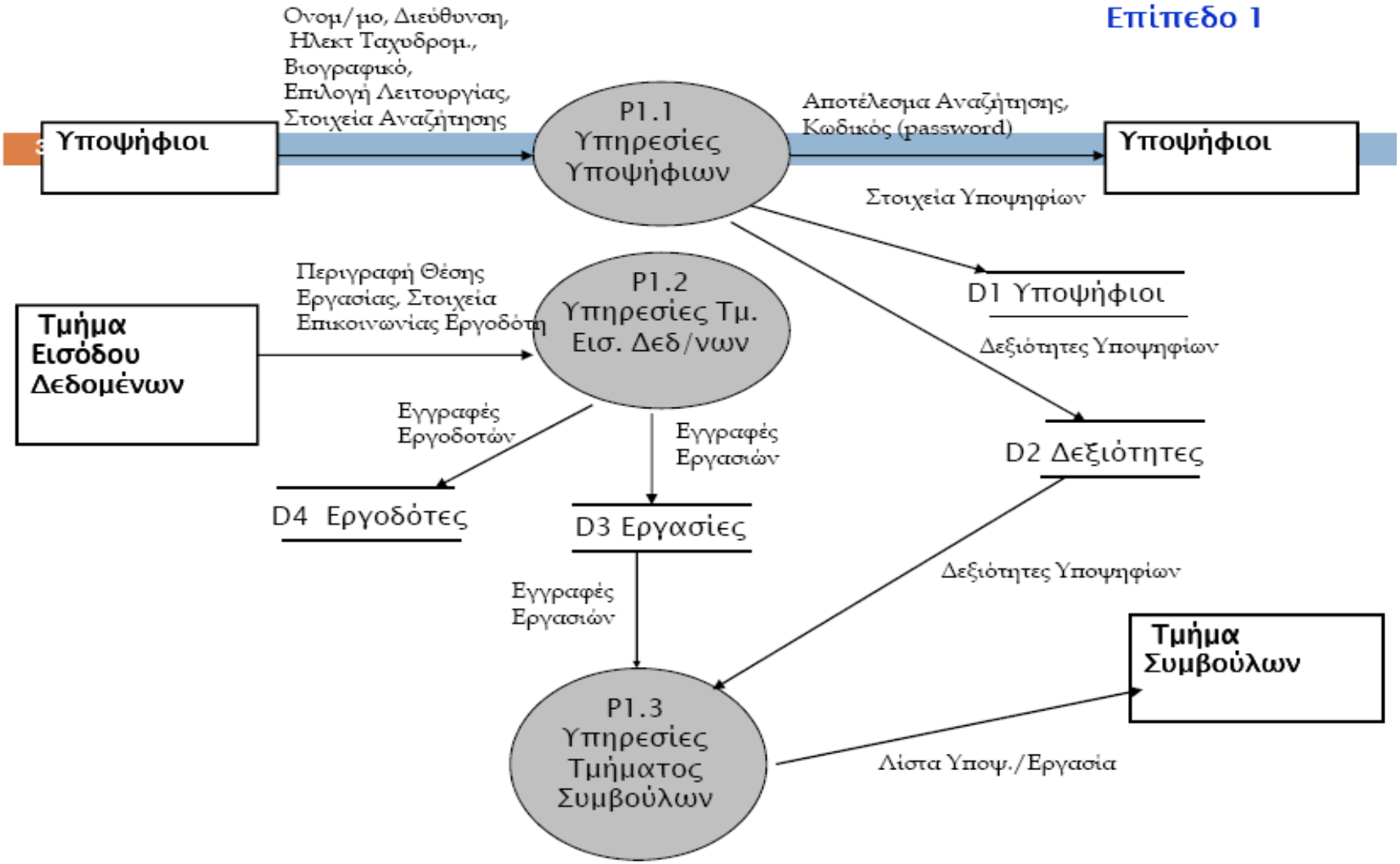
Κατά την οποία λέξεις-κλειδιά σχετικές με τις δεξιότητες των υποψηφίων αποθηκεύονται στο σύστημα.

Μερικές φορές θα χρειαστούμε να βρούμε μια αντιπροσωπευτική ονομασία για την αποθήκη, άλλες φορές μας τη δίνει η εκφώνηση.

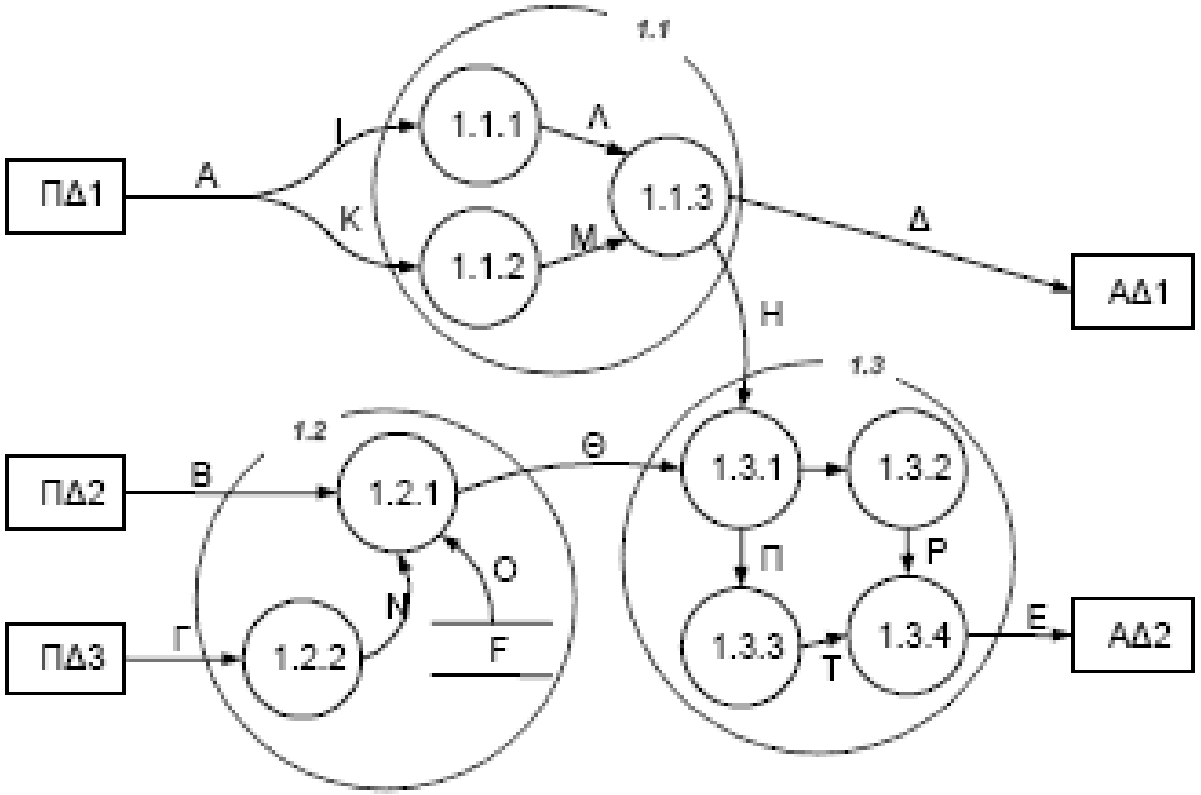
Βήμα 7 Σχεδιασμός 2ου επιπέδου

!!! Προσοχή στη συνέπεια μεταξύ των ιεραρχικών επιπέδων

Μεθοδολογία Σχεδίασης ΔΡΔ



Διαγράμματα ροής δεδομένων – Διαδοχικά επίπεδα λεπτομέρειας



ΔΡΔ – Χρήσιμες Συμβουλές

Σημασία στα κρίσιμα δεδομένα της εφαρμογής

- Κατά τη μετάβαση σε νέο επίπεδο λεπτομέρειας, αναλύονται και οι μετασχηματισμοί και τα δεδομένα
- Να μη συγχέεται με παράσταση αλγορίθμου
- Δεν παριστάνεται πληροφορία χρονισμού (χρονική αλληλουχία)
- Ισορροπία μεταξύ ουσιώδους και κατασκευαστικής λεπτομέρειας
- Χρήση κατάλληλων εργαλείων

Διαγράμματα οντοτήτων – συσχετίσεων

Ως **οντότητα (entity)** νοείται ένα σύνολο από αντικείμενα, πρόσωπα ή γεγονότα του πραγματικού κόσμου τα οποία βρίσκονται εντός του πεδίου ενδιαφέροντος της εφαρμογής λογισμικού η οποία κατασκευάζεται.

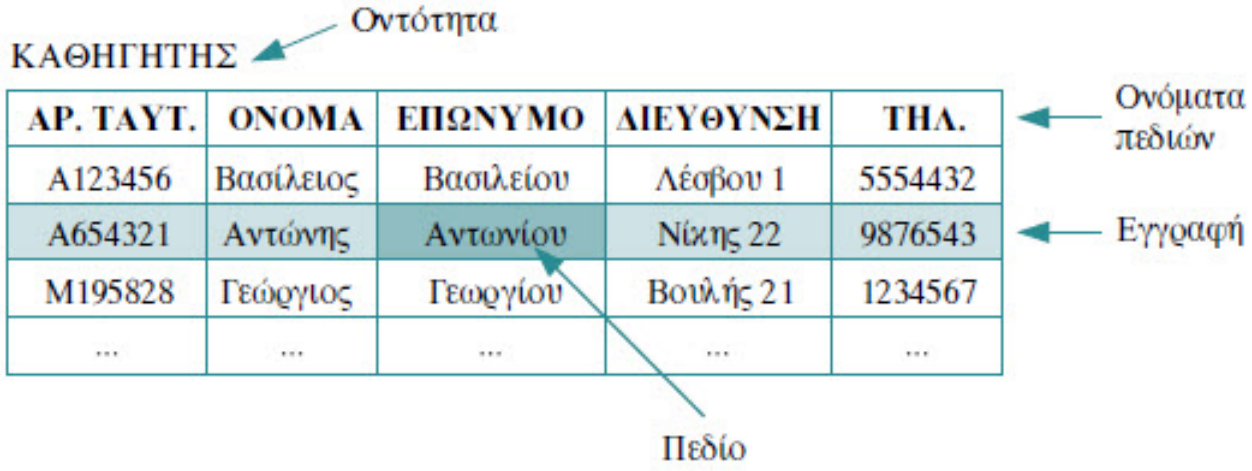
Κάθε οντότητα χαρακτηρίζεται από ένα σύνολο στοιχείων, τα οποία ονομάζονται **πεδία (fields)** και περιέχουν τις τιμές ορισμένων χαρακτηριστικών ιδιωμάτων της οντότητας.

Κάθε πεδίο περιέχει μια συγκεκριμένη πληροφορία που αφορά μια οντότητα. Το σύνολο των πεδίων που αφορούν μια συγκεκριμένη οντότητα ονομάζεται **εγγραφή (record)**.

Το σύνολο των εγγραφών αποθηκεύεται με τη βοήθεια ενός **πίνακα (table)**.

Διαγράμματα οντοτήτων – συσχετίσεων

Για παράδειγμα, η οντότητα «Καθηγητής» μπορεί να χαρακτηρίζεται από τα ιδιώματα / πεδία «Αρ. ταυτότητας», «Όνομα», «Επώνυμο», «Διεύθυνση» και «Τηλέφωνο», όπως φαίνεται στο σχήμα.



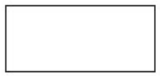
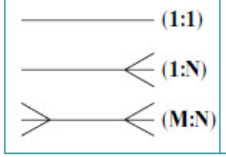
Διαγράμματα οντοτήτων – συσχετίσεων

Στο σχεσιακό μοντέλο δεδομένων υπάρχουν τρία είδη σχέσεων:

- Η σχέση «ένα προς ένα (1:1)», στην οποία ένα μέλος του πληθυσμού μιας οντότητας Α συσχετίζεται με / αντιστοιχεί σε ακριβώς ένα μέλος του πληθυσμού μιας οντότητας Β.
- Η σχέση «ένα προς πολλά (1:N)», στην οποία ένα μέλος του πληθυσμού μιας οντότητας Α συσχετίζεται με / αντιστοιχεί σε τουλάχιστον ένα μέλος του πληθυσμού μιας οντότητας Β.
- Η σχέση «πολλά προς πολλά (M:N)», στην οποία ένα ή περισσότερα μέλη του πληθυσμού μιας οντότητας Α συσχετίζονται με/ αντιστοιχούν σε ένα ή περισσότερα μέλη του πληθυσμού μιας οντότητας Β.

Διαγράμματα οντοτήτων – συσχετίσεων

Με τη βοήθεια του διαγράμματος οντοτήτων-συσχετίσεων καταγράφονται, στη φάση της προδιαγραφής των απαιτήσεων από το λογισμικό, οι απαιτήσεις σε μόνιμη αποθήκευση δεδομένων, χωρίς να ενδιαφέρει ιδιαίτερα η κατασκευαστική λεπτομέρεια. Οι συμβολισμοί που χρησιμοποιούνται στα διαγράμματα οντοτήτων -συσχετίσεων φαίνονται στο σχήμα.

Συμβολισμοί διαγραμμάτων οντοτήτων–συσχετίσεων	
	Οντότητα δεδομένων
	Συσχέτιση μεταξύ οντοτήτων (α)
	Ορισμός πολλαπλότητας συσχέτισης (α)
	Συσχέτιση και ορισμός πολλαπλότητας (β)

Διαγράμματα οντοτήτων – συσχετίσεων Χρήσιμες συμβουλές

Εντοπισμός των οντοτήτων και των σχέσεων. Όχι λεπτομέρειες.

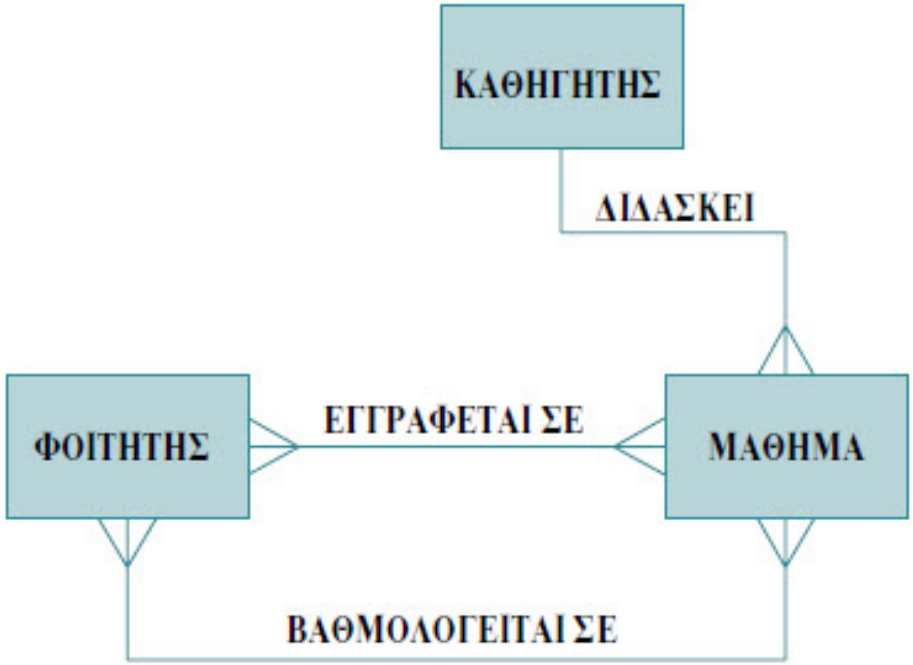
Ένα καλό διάγραμμα οντοτήτων – συσχετίσεων συνήθως είναι απλό και αναγνώσιμο.

Βελτιστοποιήσεις κλπ καλό είναι να γίνουν αργότερα, κατά τη σχεδίαση.

Ανεξαρτησία από το περιβάλλον υλοποίησης, ακόμη και αν αυτό είναι γνωστό.

Διαγράμματα οντοτήτων – συσχετίσεων

Στο σχήμα φαίνεται το διάγραμμα οντοτήτων-συσχετίσεων ενός λογισμικού εκπαιδευτικού ιδρύματος.



Διαγράμματα μετάβασης καταστάσεων

Με τα μοντέλα παράστασης λογισμικού που έχουμε παρουσιάσει μέχρι τώρα δεν παριστάνεται η δυναμική συμπεριφορά του λογισμικού, δηλαδή η χρονική σειρά εκτέλεσης εργασιών ανάλογα με τα εξωτερικά γεγονότα τα οποία προκαλεί ο χρήστης ή άλλες εξωτερικές πηγές.

Ένα χρήσιμο για το σκοπό αυτό εργαλείο είναι το διάγραμμα μετάβασης καταστάσεων (state transition(chart) diagram).

Ανάλογα με την κατάσταση που βρίσκεται το σύστημα και με το μήνυμα που λαμβάνει, εκτελείται μια εργασία και το σύστημα μεταβαίνει, ενδεχομένως, σε μια άλλη κατάσταση. Αυτή ακριβώς η συμπεριφορά είναι που περιγράφεται με το διάγραμμα μετάβασης καταστάσεων.

Διαγράμματα μετάβασης καταστάσεων

Γεγονός Ένα γεγονός (event) είναι μια στιγμιαία μεταβολή στο περιβάλλον λειτουργίας του λογισμικού, η οποία προκαλείται από εξωτερικούς παράγοντες (χρήστης, λειτουργικό σύστημα, άλλες εφαρμογές λογισμικού).

Απόκριση Μια λειτουργία που εκτελεί το λογισμικό όταν προκαλείται ένα γεγονός ονομάζεται απόκριση (response).

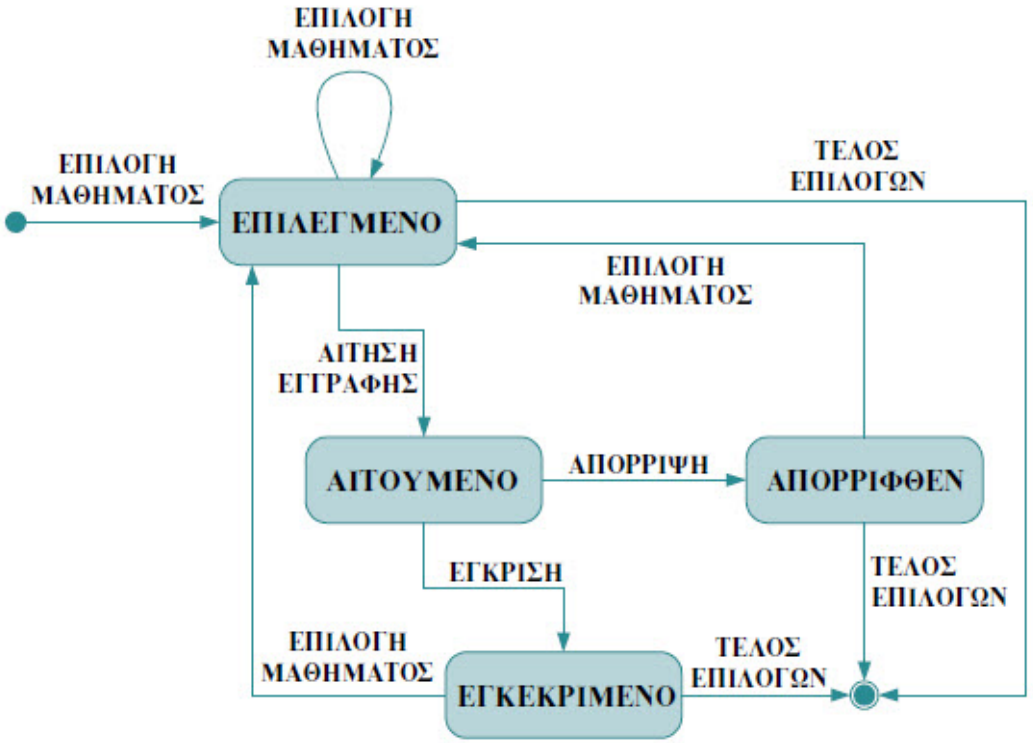
Κατάσταση Όταν το λογισμικό αναμένει («αφουγκράζεται») γεγονότα, τότε λέμε ότι βρίσκεται σε μια κατάσταση. Με τη λήψη ενός γεγονότος, το λογισμικό μπορεί να εκτελεί μια λειτουργία και να μεταβαίνει σε μια άλλη κατάσταση (παραλληλόγραμμο με στρογγυλεμένες γωνίες).

Οι καταστάσεις **έναρξης και τέλους** παριστάνονται με μια μεγάλη μαύρη κουκκίδα εκτός ή εντός κύκλου, αντίστοιχα.

Οι **μεταβάσεις** μεταξύ καταστάσεων περιγράφονται με ένα βέλος από την αρχική στην τελική κατάσταση. Πάνω στο βέλος σημειώνεται το γεγονός που προκαλεί τη μετάβαση.

Διαγράμματα μετάβασης καταστάσεων

Το διάγραμμα περιγράφει τη συμπεριφορά της μονάδας εκείνης του λογισμικού η οποία είναι υπεύθυνη για την εγγραφή φοιτητών σε αριθμό μαθημάτων.



Το λεξικό δεδομένων

Το λεξικό δεδομένων είναι μια οργανωμένη ταξινόμηση όλων των σχετιζομένων με δεδομένα στοιχείων των μοντέλων παράστασης λογισμικού με όσο το δυνατό μεγαλύτερη σαφήνεια και πληρότητα, έτσι ώστε να γίνονται κατανοητά τόσο από τον αναλυτή του συστήματος όσο και από το χρήστη αυτού.

Το λεξικό δεδομένων

Μια συνηθισμένη δομή ενός λεξικού δεδομένων είναι ένας πίνακας που περιλαμβάνει τα παρακάτω πεδία:

Ονομασία: Το κύριο αναγνωριστικό της οντότητας, πεδίου ή ροής δεδομένων.

Βοηθητικές ονομασίες: Δευτερεύουσες ονομασίες που χρησιμοποιούνται χάριν συντομίας ή ισοδύναμα.

Πού χρησιμοποιείται: Αναφορά στους μετασχηματισμούς, οντότητες κτλ. που χρησιμοποιούν το εν λόγω στοιχείο.

Πώς χρησιμοποιείται: Αναφορά στον τρόπο με τον οποίο χρησιμοποιείται το εν λόγω στοιχείο (ως στοιχείο εισόδου, ως αποτέλεσμα, πεδίο κ.ά.)

Τι περιέχει: Περιγραφή του είδους και της μορφής της πληροφορίας που αποθηκεύεται σε αυτό.

Όρια τιμών: Καθορισμός των επιτρεπτών τιμών που μπορεί να πάρει (αν απαιτείται).

Αρχική τιμή: Καθορισμός της αρχικής τιμής του στοιχείου (αν απαιτείται).

Λοιπά στοιχεία: Υπόλοιπες χρήσιμες πληροφορίες.

Ολοκληρώνοντας τη Φάση της Ανάλυσης

Θα πρέπει να συμπληρώσουμε την αναλυτική περιγραφή των προδιαγραφών απαιτήσεων έτσι ώστε για κάθε λειτουργική απαίτηση να έχουμε αναλυτικές πληροφορίες τουλάχιστον για την:

Περιγραφή της

Τις εισόδους της

Τι κάνει (τι επεξεργασία επιτελεί) και

Τι εξόδους έχει

Ερώτημα – Άσκηση

Για τις παρακάτω απαιτήσεις που αρχικά καταγράφηκαν για την προς δημιουργία εφαρμογή να γραφεί η αναλυτική περιγραφή των προδιαγραφών απαιτήσεων:

07. Επιτρέπεται η διαγραφή σπουδαστή μόνο αν δεν έχει εγγραφεί σε κανένα μάθημα.

13. Ζητείται αλφαβητική εκτύπωση της βαθμολογίας σε κάθε μάθημα.

Απάντηση (07. Επιτρέπεται η διαγραφή σπουδαστή μόνο αν δεν έχει εγγραφεί σε κανένα μάθημα. 13. Ζητείται αλφαβητική εκτύπωση της βαθμολογίας σε κάθε μάθημα.)

3.2.4 Λειτουργική απαίτηση Α4

Διαγραφή σπουδαστή. Η εφαρμογή εμφανίζει φόρμα στην οποία ο χρήστης δίνει τα στοιχεία του σπουδαστή που επιθυμεί να διαγράψει. **Είσοδος:** στοιχεία σπουδαστή προς διαγραφή. **Επεξεργασία:** έλεγχος ύπαρξης σπουδαστή, έλεγχος εγγραφής του σε μάθημα, έλεγχος συμμετοχής του σε εξέταση. **Έξοδοι:** αρχείο με διαγραμμένη εγγραφή σπουδαστή ή μήνυμα λάθους.

3.2.5 Λειτουργική απαίτηση Α5

Εκτύπωση βαθμολογίας σε μάθημα. Η εφαρμογή εμφανίζει διάλογο ερώτησης του κωδικού του μαθήματος, δέχεται τον κωδικό από το πληκτρολόγιο και ετοιμάζει την εκτύπωση, την οποία στέλνει στον εκτυπωτή. **Είσοδος:** κωδικός μαθήματος. **Επεξεργασία:** έλεγχος ύπαρξης μαθήματος και ύπαρξης εγγραφών βαθμολογίας, ετοιμασία και μορφοποίηση της εκτύπωσης. **Έξοδοι:** η εκτύπωση στον εκτυπωτή του συστήματος ή μήνυμα λάθους.

Προβλήματα στον προσδιορισμό απαιτήσεων

A man is flying in a hot air balloon and realizes he is lost. He reduces height and spots a man down below. He lowers the balloon further and shouts:

"Excuse me, can you tell me where I am?"

The man below says: "Yes you're in a hot air balloon, hovering 30 feet above this field."

"You must be a software developer," says the balloonist.

"I am," replies the man. "How did you know?"

"Well," says the balloonist, "everything you have told me is technically correct, but it's of no use to anyone."

The man below says,

"You must work in business as a manager."

"I do," replies the balloonist, "but how did you know?"

"Well," says the man, "you don't know where you are or where you are going, but you expect me to be able to help. You're in the same position you were before we met but now it's my fault."

Προβλήματα προτύπων / φυσική γλώσσα

Τα όποια μοντέλα παράστασης λογισμικού που συνήθως χρησιμοποιούνται στην ανάλυση και σχεδίαση λογισμικού πάντα θα πρέπει να συμπληρωθούν από κάποιου είδους κείμενο γραμμένο σε φυσική γλώσσα.

Η φυσική γλώσσα παραμένει ένα ισχυρό εργαλείο και για την περιγραφή των απαιτήσεων από το λογισμικό.

Προσοχή: Η ευστοχία και η σαφήνεια της γλώσσας είναι πολύ σημαντική και καθορίζει την ακρίβεια και τη σαφήνεια της περιγραφής των απαιτήσεων, είτε αυτή χρησιμοποιείται ως τίτλος σε μετασχηματισμό, ροή δεδομένων, οντότητα κτλ. είτε ως συνταγμένο κείμενο κάποιας παραγράφου του εγγράφου προδιαγραφών των απαιτήσεων από το λογισμικό.

Ο σωστός χειρισμός της γλώσσας, λοιπόν, είναι απαραίτητη προϋπόθεση για την περιγραφή των απαιτήσεων από το λογισμικό.

Σχεδίαση Συστήματος

Σκοπός του μαθήματος

Σκοπός του μαθήματος είναι

- η παρουσίαση μεθοδολογιών κατάστρωσης και τρόπων περιγραφής του σχεδίου του λογισμικού.
- η αναφορά στη σχεδίαση καλύπτει τα σχέδια διάταξης, την αρχιτεκτονική σχεδίαση, τη σχεδίαση διαπροσωπειών, τη λεπτομερή σχεδίαση μονάδων, καθώς και τη σχεδίαση σχεσιακών βάσεων δεδομένων.

Αντικείμενο και αποτέλεσμα της σχεδίασης

Η δημιουργία ενός **σχεδίου λογισμικού** δηλαδή η περιγραφή των μονάδων που αποτελούν το λογισμικό, των συσχετίσεων μεταξύ τους, της διάταξής τους, καθώς και της εσωτερικής τους λεπτομέρειας.

Αρχιτεκτονική σχεδίαση: Αφορά τον καθορισμό του ποιες είναι οι μονάδες που συγκροτούν το σύστημα λογισμικού και πώς αυτές ανατίθενται για εκτέλεση (διατάσσονται) στις υπολογιστικές μονάδες που είναι διαθέσιμες.

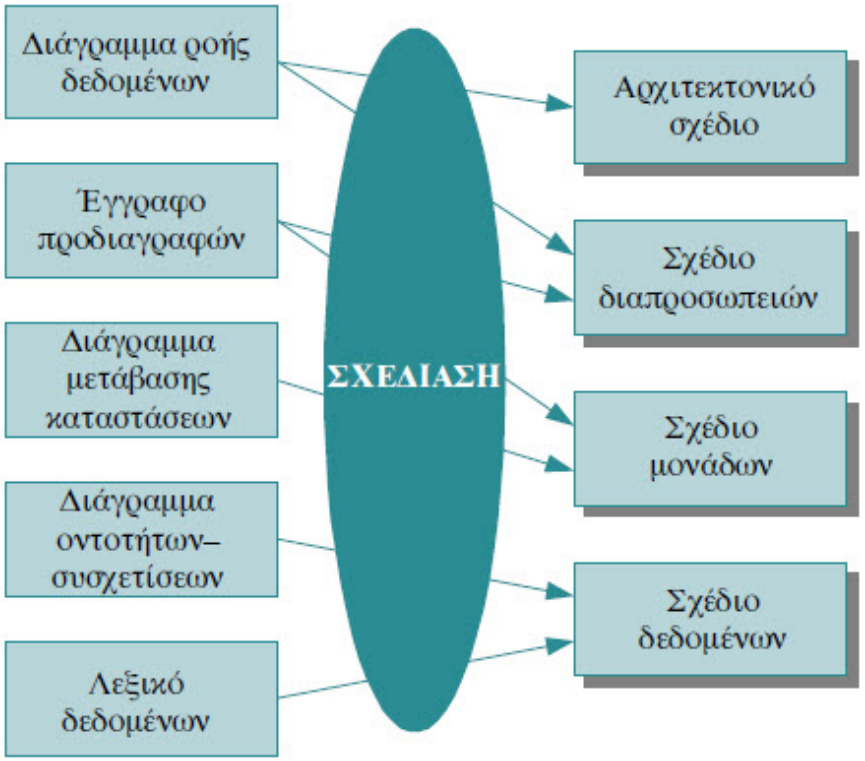
Σχεδίαση διαπροσωπειών (interfaces): Αφορά τον καθορισμό της επικοινωνίας των μονάδων μεταξύ τους, με άλλα συστήματα λογισμικού, με άλλες συσκευές, καθώς και με τον άνθρωπο.

Λεπτομερής σχεδίαση μονάδων: Αφορά τον καθορισμό της εσωτερικής δομής κάθε μονάδας λογισμικού προκειμένου αυτή να ικανοποιεί, αφενός, τις λειτουργικές απαιτήσεις και, αφετέρου, να συνεργάζεται με τις άλλες μονάδες όπως έχει καθοριστεί κατά την αρχιτεκτονική και τη σχεδίαση δεδομένων.

Σχεδίαση δεδομένων: Πρόκειται για τη λεπτομερή σχεδίαση των δεδομένων, η τήρηση των οποίων αποτελεί απαίτηση από το λογισμικό, η οποία έχει τεθεί στη φάση προδιαγραφής των απαιτήσεων

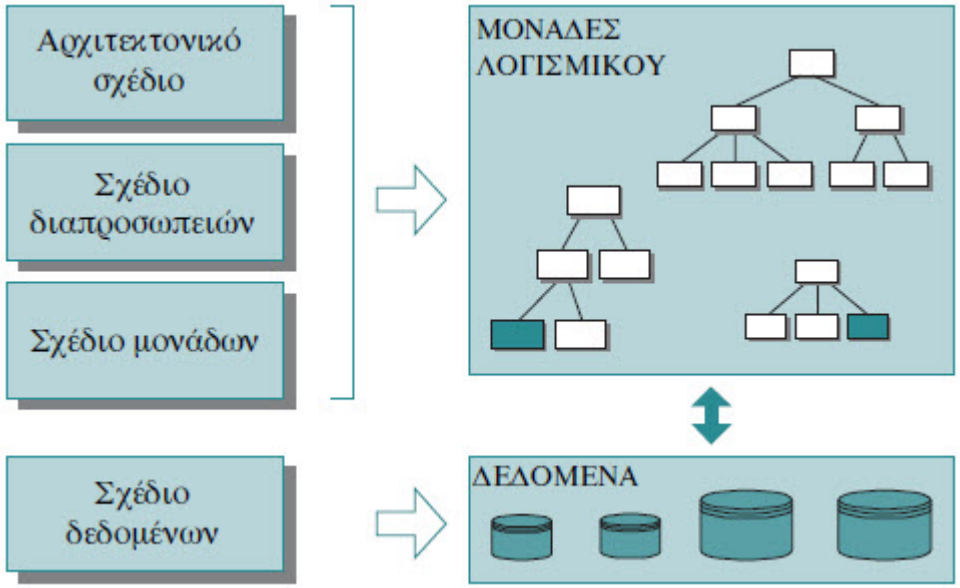
Αντικείμενο και αποτέλεσμα της σχεδίασης

Παρακάτω φαίνεται η συσχέτιση μεταξύ των προϊόντων σχεδίασης, που προκύπτουν από την εκτέλεση των εργασιών στα τέσσερα επίπεδα που αναφέρθηκαν, με τα προϊόντα της φάσης προσδιορισμού των προδιαγραφών.



Αντικείμενο και αποτέλεσμα της σχεδίασης

Από το αρχιτεκτονικό σχέδιο, το σχέδιο διαπροσωπειών και το λεπτομερές σχέδιο μονάδων, τελικά θα κατασκευαστεί μια δομή ενεργών συστατικών (μονάδων) λογισμικού, τα οποία θα επιδρούν πάνω σε κάποια ανεξάρτητα δεδομένα. Οι συναρτήσεις (functions), οι υπορουτίνες (subroutines), οι μονάδες (units), οι διαδικασίες (procedures) είναι οι συνηθέστεροι όροι που χρησιμοποιούνται από τις γλώσσες προγραμματισμού για να περιγράψουν μονάδες λογισμικού.



Σχεδίαση διαπροσωπειών (interfaces)

Κατά τη σχεδίαση διαπροσωπειών (interface design) καθορίζονται οι λεπτομέρειες του περάσματος των παραμέτρων αυτών σε όλα τα επίπεδα επικοινωνίας.

Συγκεντρώνοντας το σύνολο των απαιτήσεων από τη σχεδίαση διαπροσωπειών, καταλήγουμε ότι κατά την εργασία αυτή θα πρέπει να καθοριστούν τα ακόλουθα:

–Ο αριθμός και ο τύπος των παραμέτρων κατά την κλήση μονάδων λογισμικού.

–Το είδος και οι λεπτομέρειες της επικοινωνίας με άλλα συστήματα λογισμικού.

–Το είδος και οι λεπτομέρειες της επικοινωνίας με εξωτερικές συσκευές.

–Η επικοινωνία του λογισμικού με τον άνθρωπο

Η επικοινωνία του λογισμικού με τον άνθρωπο (user interface)

Το user interface πρέπει να είναι επικεντρωμένο στον χρήστη (user-centred). Πρέπει να είναι λογικό και να βοηθά στην επίλυση σφαλμάτων.

Στο φιλικό προς τον χρήστη (user-friendly) περιβάλλον βασίζεται κατά πολύ η εμπορική επιτυχία του λογισμικού.

Να χρησιμοποιούνται γραφικές παραστάσεις, όπου είναι δυνατόν (όταν δεν απαιτείται μεγάλη ακρίβεια).

Συνεχής και εκτεταμένη (όχι υπερβολική) χρήση χρώματος.

Το σύστημα να προσφέρει on-line βοήθεια.

Τα μηνύματα λαθών πρέπει να έχουν θετικό αντί για αρνητικό χαρακτήρα.

Πρέπει να προσφέρεται μια σειρά βοηθητικών εγγράφων στο χρήστη.

Σχεδίαση δεδομένων

Προκειμένου να μπορεί να κατασκευαστεί με πληρότητα το λεπτομερές σχέδιο μονάδων λογισμικού, είναι απαραίτητο να διατίθεται το λεπτομερές σχέδιο δεδομένων.

Έχοντας, λοιπόν, υπόψη το λεξικό δεδομένων και το διάγραμμα οντοτήτων -συσχετίσεων, ο σχεδιαστής λογισμικού:

- Πραγματοποιεί βελτιστοποιήσεις στο μοντέλο οντοτήτων - συσχετίσεων, με σκοπό τη βελτιστοποίηση των επιδόσεων, την πληροφοριακή πληρότητα και την εξάλειψη πλεονάζουσας (redundant) πληροφορίας.
- Καθορίζει τη δομή κάθε πίνακα (πεδία και τύποι αυτών) στο φυσικό επίπεδο.
- Καθορίζει τις δυνατές όψεις των δεδομένων στο λογικό επίπεδο (views).

Αρχιτεκτονικό σχέδιο συστήματος – Σχεδίαση συστήματος

Η μέθοδος της δομημένης σχεδίασης βασίζεται στα διαγράμματα ροής δεδομένων, από τα οποία, με κάποιο συστηματικό αλλά όχι αυτόματο τρόπο, παράγεται το αρχιτεκτονικό σχέδιο του λογισμικού. Το σχέδιο αυτό αποτυπώνεται με τη βοήθεια ενός διαγράμματος που ονομάζεται «**διάγραμμα δομής προγράμματος**», όπως αυτό που φαίνεται στο παρακάτω σχήμα

Αρχιτεκτονικό σχέδιο συστήματος

Στο διάγραμμα δομής προγράμματος οι μονάδες λογισμικού παριστάνονται με ένα παραλληλόγραμμο. Η κλήση της μονάδας Β από τη μονάδα Α υποδηλώνεται με ένα βέλος που συνδέει την Α με τη Β.

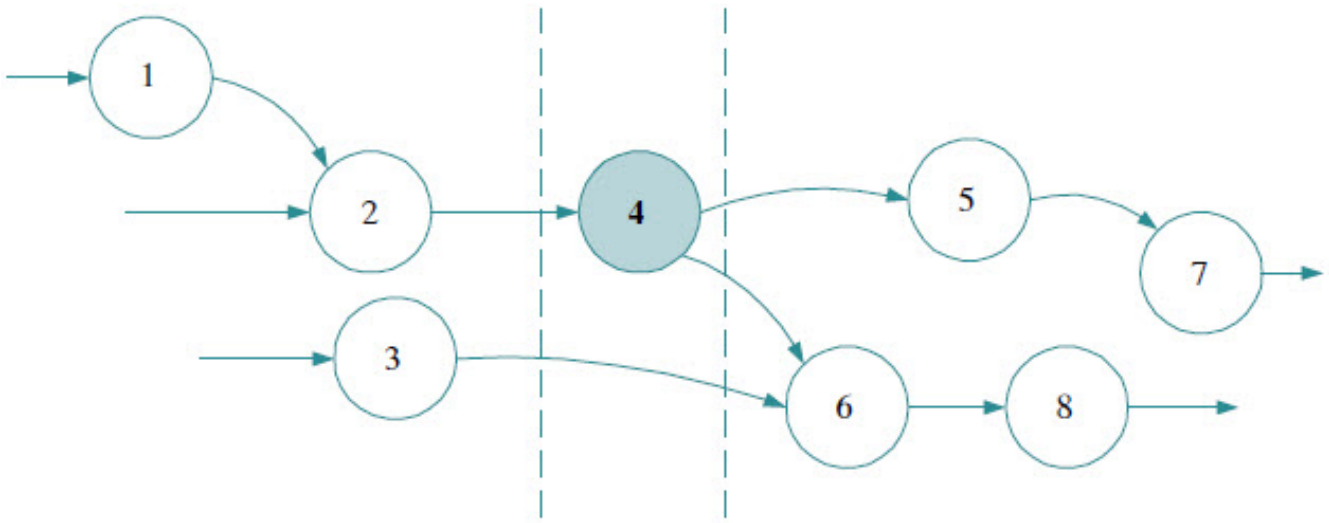
Το πέρασμα παραμέτρων υποδηλώνεται με ένα βέλος με έναν κύκλο στο άκρο της αρχής του, το οποίο φέρει το όνομα της παραμέτρου.

Στα διαγράμματα αυτά εντοπίζονται δύο τύποι χαρακτηριστικών περιοχών, οι **κεντρικοί μετασχηματισμοί** και τα **κέντρα δοσοληψιών**.

Όταν εντοπιστεί μια τέτοια περιοχή, δημιουργείται ή συμπληρώνεται ένα επίπεδο του διαγράμματος δομής προγράμματος και η διαδικασία επαναλαμβάνεται μέχρις ότου να εξαντληθεί το διάγραμμα ροής δεδομένων.

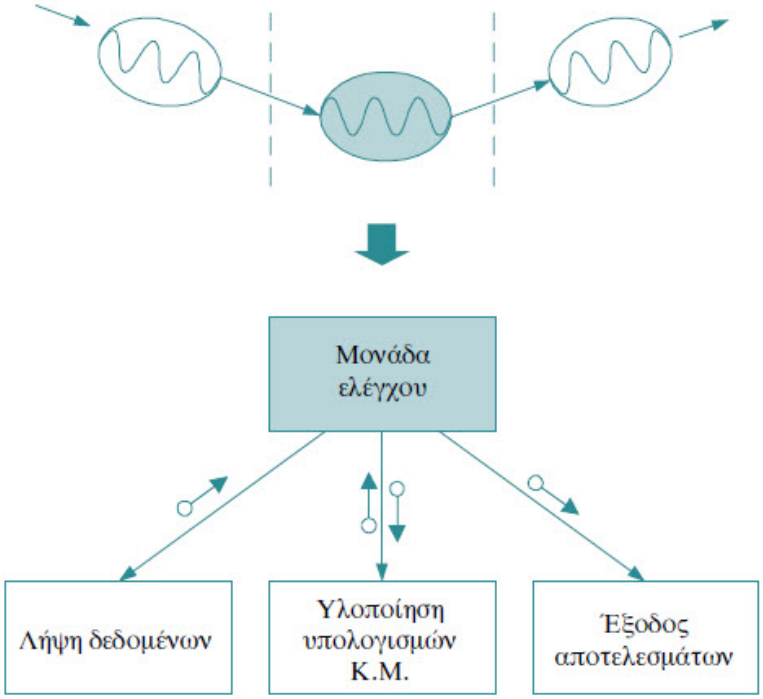
Κεντρικός Μετασχηματισμός

Στο Σχήμα φαίνεται ένα διάγραμμα ροής δεδομένων στο οποίο θεωρείται ότι οι μετασχηματισμοί 1, 2 και 3 λαμβάνουν και προετοιμάζουν τα δεδομένα εισόδου. Ο μετασχηματισμός 4 κάνει την κύρια δουλειά της δημιουργίας νέων δεδομένων και χαρακτηρίζεται ως κεντρικός μετασχηματισμός, ενώ οι μετασχηματισμοί 5, 6, 7 και 8 χρησιμοποιούν τα δεδομένα εξόδου.



Απεικόνιση κεντρικού μετασχηματισμού

Ένας κεντρικός μετασχηματισμός απεικονίζεται σε διάγραμμα δομής, όπως φαίνεται στο παρακάτω σχήμα. Η μονάδα ελέγχου εκτέλεσης καλεί τις μονάδες που απαιτείται για να λάβει τα δεδομένα εισόδου, τις μονάδες που απαιτείται να εκτελέσουν τον κεντρικό μετασχηματισμό και, τέλος, τις μονάδες στις οποίες θα διαθέσει τα δεδομένα εξόδου.

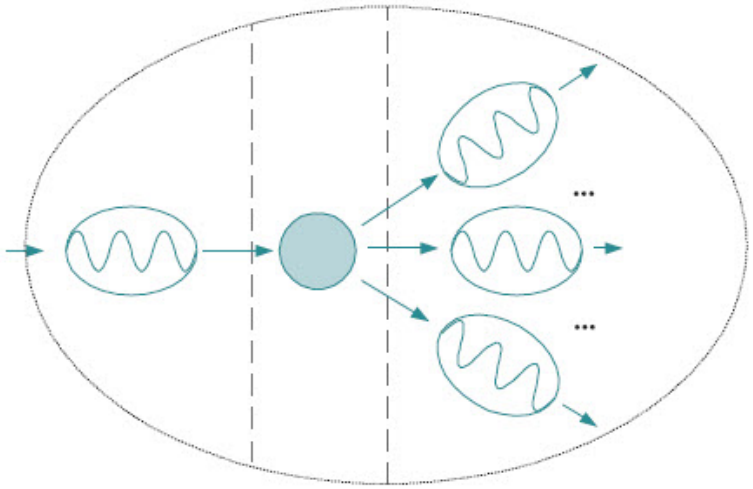


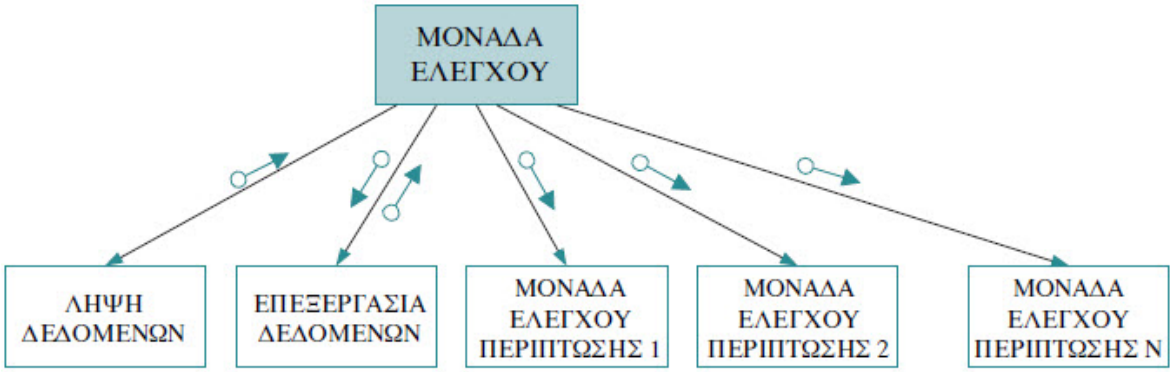
Κέντρο δοσοληψιών

Ως κέντρο δοσοληψιών (transaction centre) ορίζεται ένας μετασχηματισμός του διαγράμματος ροής δεδομένων, ο οποίος δέχεται κάποια δεδομένα εισόδου και παράγει ένα σύνολο δεδομένων εξόδου ανάλογα με την τιμή των δεδομένων εισόδου.

Παράδειγμα: Χαρακτηριστική περίπτωση κέντρου δοσοληψιών είναι ένας μετασχηματισμός ελέγχου ροής προγράμματος, όπου, ανάλογα με την επιλογή του χρήστη, η ροή μεταφέρεται σε διαφορετικούς μετασχηματισμούς, όπως η περίπτωση ενός μενού.

Απεικόνιση κέντρου δοσοληψιών





Βήματα κατασκευής διαγραμμάτων δομής

Η μετάβαση από το διάγραμμα ροής δεδομένων στο διάγραμμα δομής γίνεται με διαδοχική επανάληψη κάποιων βημάτων, μέχρι να έχουν προσδιοριστεί μονάδες για όλους τους μετασχηματισμούς που περιέχονται στα διαγράμματα ροής δεδομένων της εφαρμογής. Τα βήματα αυτά είναι:

1. **Εντοπισμός κεντρικού μετασχηματισμού.** Για κάθε τμήμα του διαγράμματος ροής δεδομένων εντοπίζεται ο κεντρικός μετασχηματισμός και διακρίνονται οι μετασχηματισμοί εισόδου και εξόδου σε αυτόν.

2. **Απεικόνιση του κεντρικού μετασχηματισμού σε διάγραμμα δομής.** Δημιουργείται ένα επίπεδο του διαγράμματος δομής που αντιστοιχεί στον κεντρικό μετασχηματισμό.

Βήματα κατασκευής διαγραμμάτων δομής

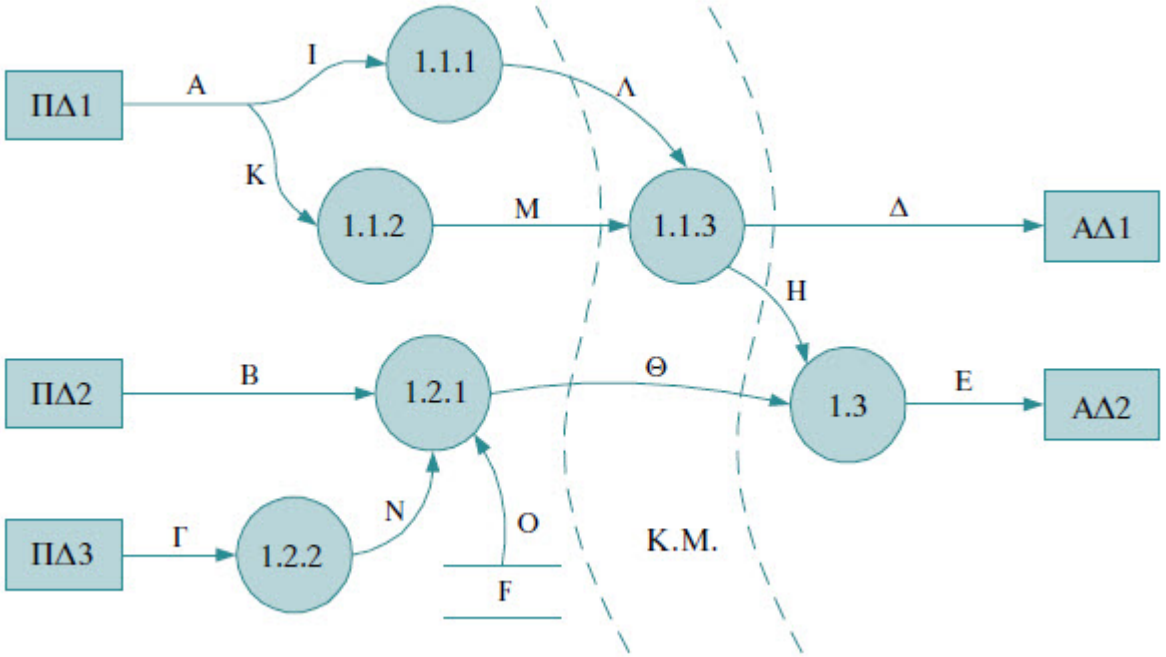
3. **Παραγοντοποίηση (factoring).** Για το αριστερό και το δεξί τμήμα του κεντρικού μετασχηματισμού (είσοδοι και έξοδοι) δημιουργούνται τα διαγράμματα δομής, που αντιστοιχούν στους μετασχηματισμούς που περιέχονται σε αυτά. Κάθε τέτοιος μετασχηματισμός απεικονίζεται σε μια μονάδα ελέγχου και σε δύο άλλες μονάδες.

Η διαδικασία επαναλαμβάνεται μέχρις ότου να φτάσουμε στις πηγές και τους αποδέκτες των δεδομένων, δηλαδή το χρήστη, εξωτερικά συστήματα, συσκευές ή αρχεία.

4. **Συνένωση.** Η τελευταία εργασία που πρέπει να γίνει είναι αυτή της συνένωσης. Για τις περιπτώσεις όπου τα δεδομένα δε λαμβάνονται από εξωτερική πηγή ή δεν καταλήγουν σε εξωτερικό αποδέκτη, η μονάδα που τα εισάγει στον κεντρικό μετασχηματισμό αντικαθίσταται από τη μονάδα ελέγχου του μετασχηματισμού που τα παρέχει.

Παράδειγμα

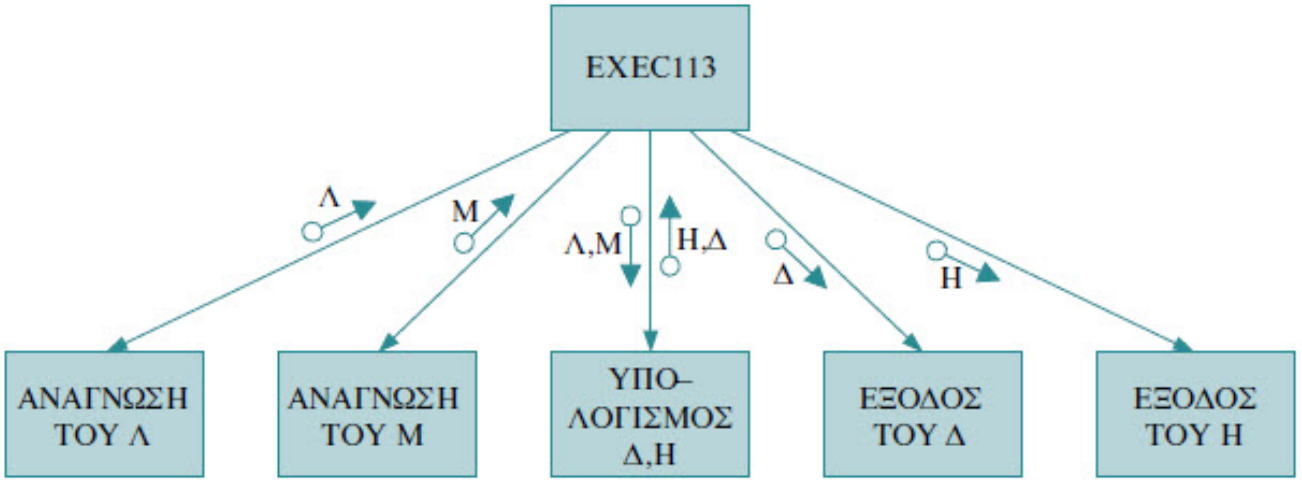
Έστω το ακόλουθο διάγραμμα ροής δεδομένων.



Επιλέγοντας το μετασχηματισμό 1.1.3 ως τον κεντρικό μετασχηματισμό να κατασκευαστεί το διάγραμμα δομής στο πρώτο επίπεδο.

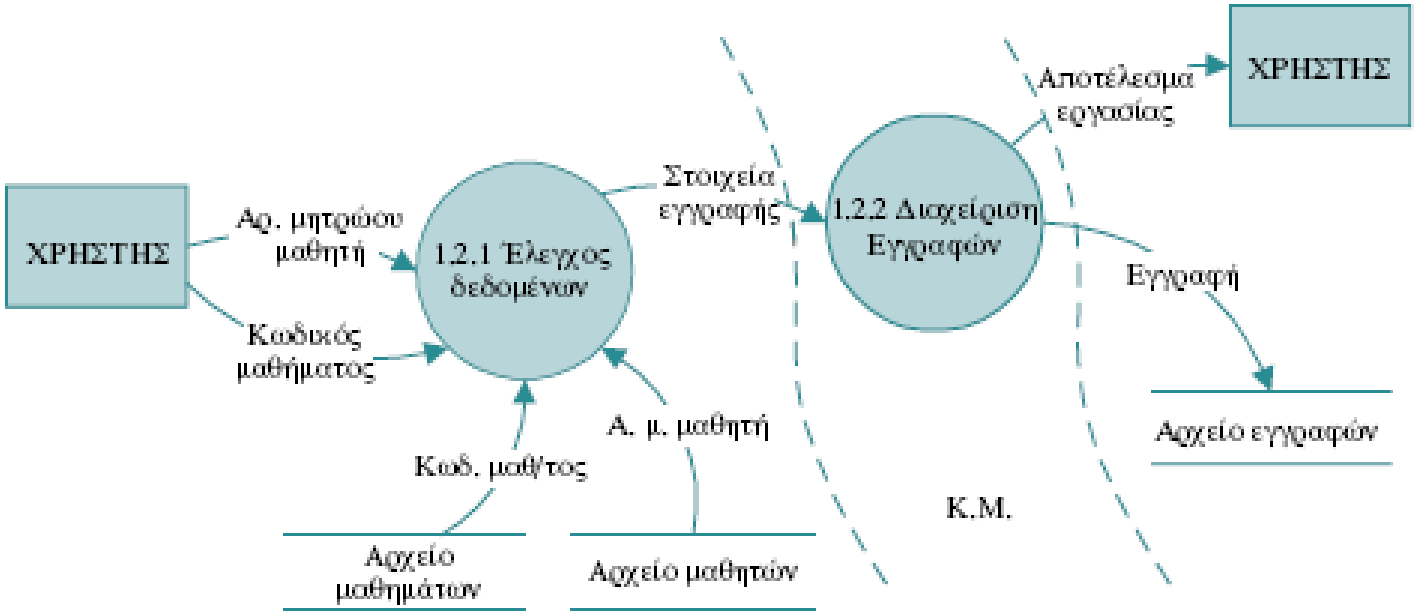
Απάντηση

Με βάση την επιλογή αυτή, κατασκευάζουμε το διάγραμμα δομής που φαίνεται στο Σχήμα (εφαρμόζοντας το βήμα 2 που περιγράφηκε). Στον κεντρικό μετασχηματισμό εισέρχονται δύο ροές δεδομένων, οι Λ και Μ. Αυτές αντιστοιχούν στις μονάδες «ΑΝΑΓΝΩΣΗ ΤΟΥ Λ» και «ΑΝΑΓΝΩΣΗ ΤΟΥ Μ». Τα δεδομένα αυτά περνάνε στη μονάδα «ΥΠΟΛΟΓΙΣΜΟΣ Δ, Η», τα οποία είναι τα δεδομένα εξόδου από τον κεντρικό μετασχηματισμό. Αυτά διοχετεύονται προς δύο μονάδες, τις «ΕΞΟΔΟΣ ΤΟΥ Δ» και «ΕΞΟΔΟΣ ΤΟΥ Η», αντίστοιχα.



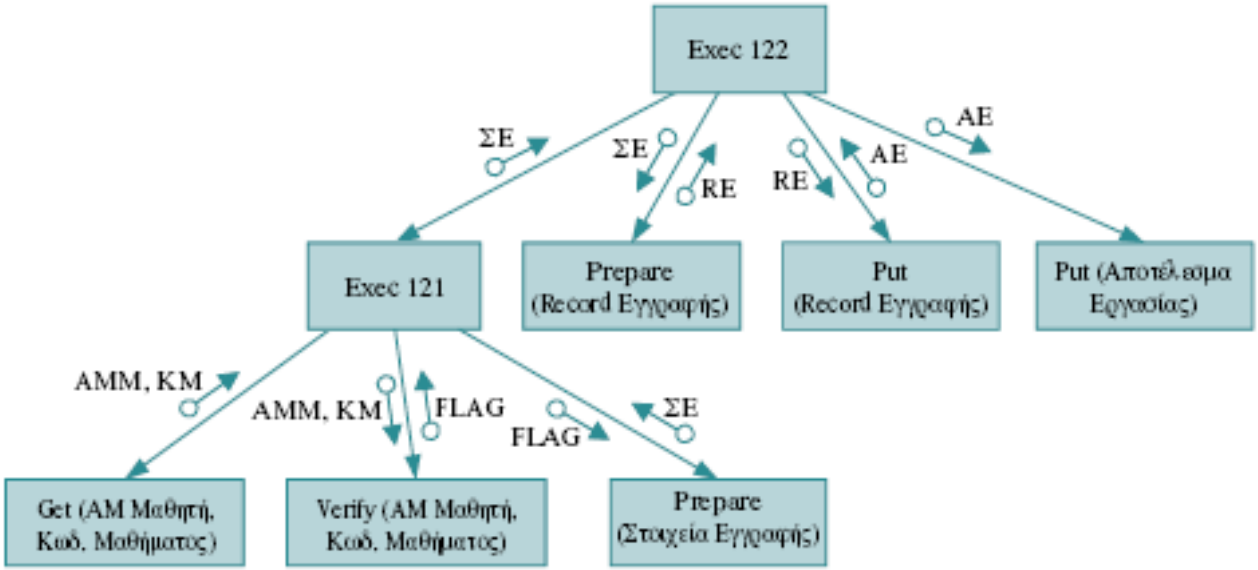
Άσκηση

Να δημιουργηθεί το διάγραμμα δομής για το παρακάτω τμήμα διαγράμματος ροής δεδομένων.



Απάντηση

Ως κεντρικός μετασχηματισμός επιλέγεται ο μετασχηματισμός 1.2.2, διότι ο μετασχηματισμός 1.2.1 ετοιμάζει τα δεδομένα, ο 1.2.2 κάνει την κύρια εργασία και τα αποτελέσματα καταναλώνονται από το χρήστη και το αρχείο. Στο διάγραμμα δομής φαίνεται και η παραγοντοποίηση του μετασχηματισμού 1.2.1.



Λεπτομερής σχεδίαση μονάδων

Έχοντας διαθέσιμο το αρχιτεκτονικό σχέδιο, είναι δυνατή η λεπτομερής σχεδίαση των μονάδων λογισμικού που περιλαμβάνονται σε αυτό. Κατά τη λεπτομερή σχεδίαση θα προσδιοριστεί η εσωτερική δομή κάθε μονάδας, δηλαδή θα δοθεί μια περιγραφή του πηγαίου κώδικα.

Υπάρχουν αρκετοί τρόποι για να περιγράφονται τα αποτελέσματα της λεπτομερούς σχεδίασης. Ο επικρατέστερος είναι με χρήση μιας γλώσσας σχεδίασης προγράμματος (PDL: program description language) που μοιάζει με γλώσσα προγραμματισμού.

Οι γλώσσες σχεδίασης προγράμματος (ψευδογλώσσες – απλοϊκή C, απλοϊκή Pascal) περιέχουν τις βασικές δομές ελέγχου που απαντώνται στις γλώσσες προγραμματισμού.

Λεπτομερής σχεδίαση μονάδων

Απλές εκφράσεις	Επαναληπτική εκτέλεση

/*σχόλιο */
μεταβλητή:–τιμή /* ανάθεση */
φραστική περιγραφή ενέργειας
+ – * / ^ /* μαθηματικές εκφράσεις */
	FOR μτβλ FROM τιμή1 TO τιμή2 STEP τιμή3 DO
	(ενέργειες)
	END FOR

Εκτέλεση με επιλογή περίπτωσης	**Εκτέλεση υπό συνθήκη**

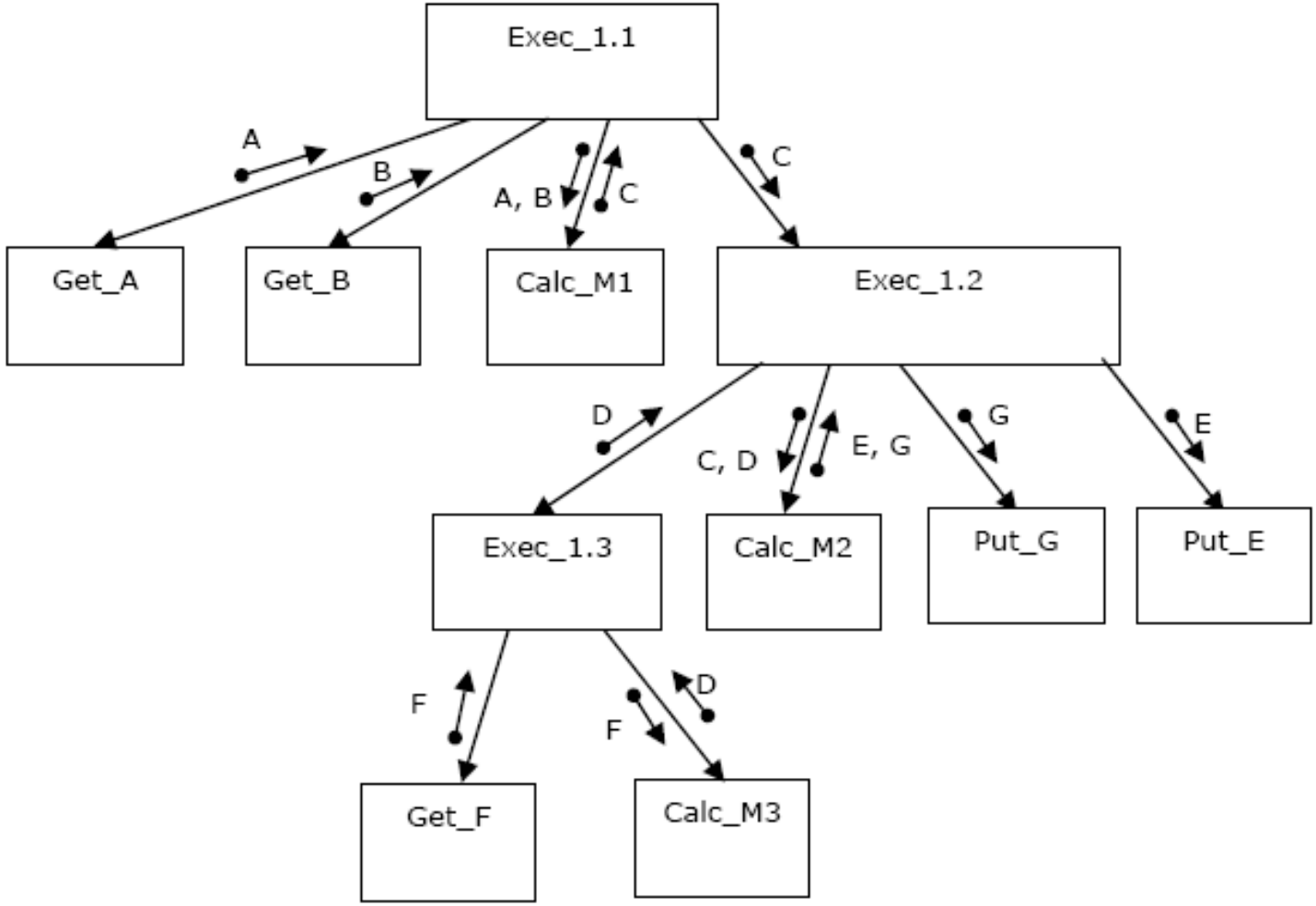
CASE έκφραση OF
(τιμή 1) : (ενέργειες)
(τιμή 2) : (ενέργειες)
…
(τιμή N) : (ενέργειες)
OTHERWISE
(εντολές αν η έκφραση έχει άλλη τιμή)
END CASE
	IF συνθήκη THEN
	(ενέργειες αν η συνθήκη είναι αληθής)
	ELISE
	(εντολή αν η συνθήκη είναι ψευδής)
	END IF

Επαναληπτική εκτέλεση με συνθήκη (1)	**Επαναληπτική εκτέλεση με συνθήκη (2)**

REPEAT
(ενέργειες)
UNTIL συνθήκη
	WHILE συνθήκη DO
	(ενέργειες)
	END WHILE

Ορισμός διαδικασιών	**Ορισμός συναρτήσεων**
PROCEDURE όνομα (παράμετρος: IN/OUT, …) GLOBAL VAR όνομα 1, όνομα 2, … LOCAL VAR όνομα1, όνομα 2, … … (ενέργειες) … CALL όνομα διαδικασίας (παραμ1, παραμ2, …) … (ενέργειες) … END PROCEDURE	FUNCTION όνομα συνάρτησης (παράμετρος, …) GLOBAL VAR όνομα 1, όνομα 2, … LOCAL VAR όνομα 1, όνομα 2, … … (ενέργειες) … όνομα συνάρτησης: = τιμή … (ενέργειες) … END FUNCTION

Παράδειγμα – Άσκηση



Να γίνει η λεπτομερής σχεδίαση των μονάδων: Exec_1.1, Get_A, Exec_1.2

Απάντηση

```
PROCEDURE Exec_1.1(A:IN/OUT, B: IN/OUT, C: IN/OUT)
  LOCAL VAR A, B, C;
  Αρχικοποίησε A, B, C;
  CALL Get_A(A);
  CALL Get_B(B);
  CALL Calc_M1(A, B, C);
  CALL Exec_1.2(C);
END PROCEDURE
```

```
PROCEDURE Get_B (B:IN)
  ΔΙΑΒΑΣΕ B;
END PROCEDURE
```

```
PROCEDURE Exec_1.2
   LOCAL VAR D, C, E, G;
   Αρχικοποίησε D, C, E, G;
   CALL Exec_1.3(D, F);
   CALL Calc_M2(C, D, E, G);
   CALL Put_G(G);
   CALL Put_E(E)
END PROCEDURE
```

Σχεδίαση Δεδομένων

Εργασίες που πρέπει οπωσδήποτε να εκτελούνται κατά τη λεπτομερή σχεδίαση δεδομένων:

Τροποποίηση του μοντέλου οντοτήτων – συσχετίσεων, την αρχική μορφή του οποίου κατασκευάσαμε κατά τη συγγραφή των προδιαγραφών με σκοπό αυτό να περιέχει ακριβώς όση πληροφορία απαιτείται, κατανεμημένη σωστά μεταξύ των οντοτήτων.

Η διαδικασία αυτή ονομάζεται «κανονικοποίηση».

Καθορισμός των πεδίων που περιέχει κάθε πίνακας στο επίπεδο της φυσικής αποθήκευσης δεδομένων.

Καθορισμός των εναλλακτικών μορφών εμφάνισης της οργάνωσης των δεδομένων πάνω από το φυσικό επίπεδο (ορισμός views).

Βελτιστοποιήσεις με σκοπό τη βελτιστοποίηση των επιδόσεων και, γενικά, την επίτευξη κριτηρίων επάρκειας.

Από τη Σχεδίαση στην Κωδικοποίηση

Σκοπός του μαθήματος

Σκοπός του μαθήματος είναι:

- η παρουσίαση των εργασιών και των προβλημάτων κατά τη φάση συγγραφής του πηγαίου κώδικα του λογισμικού

Η παρουσίαση γίνεται αφαιρετικά χωρίς αναφορά σε συγκεκριμένη γλώσσα προγραμματισμού

Σε τι θα αναφερθούμε

Λογισμικό χωρίς σφάλματα

Εργαλεία κωδικοποίησης

Χαρακτηριστικά πηγαίου κώδικα

Βασικές αιτίες σφαλμάτων στον πηγαίο κώδικα

Παραγόμενο λογισμικό

Η μετάβαση από τη λεπτομερή σχεδίαση στον πηγαίο κώδικα πρέπει να γίνεται με τρόπο που να παράγεται λογισμικό χωρίς σφάλματα.

Λογισμικό χωρίς σφάλματα

Είναι το λογισμικό που πληροί τις προδιαγραφές βάσει των οποίων κατασκευάστηκε, χωρίς να παράγει σφάλματα κατά την εκτέλεσή του (runtime), δηλαδή το λογισμικό που κάνει ακριβώς αυτό για το οποίο προορίζεται και το κάνει σωστά.

Εργαλεία κωδικοποίησης

Συντάκτες προγραμμάτων (program editors)

Μεταφραστικά περιβάλλοντα γλωσσών προγραμματισμού (programming language implementations) (παλαιότερα εργαλεία)

- Μεταγλωττιστές (compilers)
- Διερμηνείς (interpreters)
- Συμβολομεταφραστές (assemblers)

Εντοπιστές σφαλμάτων (debuggers)

Γεννήτορες προγραμμάτων (program generators) (πολλοί εξειδικευμένοι)

Συστήματα υποστήριξης λογισμικού (software support systems) και

Ολοκληρωμένα περιβάλλοντα προγραμματισμού (integrated programming environments)

Χαρακτηριστικά πηγαίου κώδικα

Επάρκεια (efficiency) δηλαδή το λογισμικό λειτουργεί σωστά και χωρίς σφάλματα

Επίδοση (performance) δηλαδή παρέχει υψηλή ταχύτητα εκτέλεσης συγκεκριμένων λειτουργιών (**πολυπλοκότητα αλγορίθμων**) και χαμηλές απαιτήσεις σε πόρους, όπως μνήμη και χωρητικότητα δίσκου

Αναγνωσιμότητα (readability) (Εσωτερική τεκμηρίωση)

- Απλότητα
- Πλεονασμός χάριν σαφήνειας (begin .. end, { .. }, (..) κλπ
- Επιλογή κατάλληλων ονομάτων
- Χρήση σχολίων
- Στοίχιση (εσοχές) και αριθμός εντολών ανά γραμμή κώδικα

Τεκμηρίωση (documentation) (εξωτερική)

Μεταφερσιμότητα (portability)

Δυνατότητα επαναχρησιμοποίησης (reusability)

Αναγνωσιμότητα κώδικα (επιλογή καταλλήλων ονομάτων – εσοχές)

Τι κάνει το παρακάτω κομμάτι κώδικα;

```
if (x > 50000)
 y = (x – 50000)* 0.2;
Else
 y = 0;
```

Αναγνωσιμότητα κώδικα (επιλογή καταλλήλων ονομάτων – εσοχές)

Αυτό κάνει

```
const double NOT_TAXABLE = 50000;
const double TAX_RATE = 0.2;

if (income > NOT_TAXABLE)
 tax = (income – NOT_TAXABLE) * TAX_RATE;
else
 tax = 0;
```

Τεκμηρίωση (εξωτερική)

Συνοδευτικά έντυπα γραμμένα συνήθως σε φυσική γλώσσα που αποσκοπούν στην:

Περιγραφή προβλήματος

Περιγραφή αλγορίθμων

Περιγραφή δεδομένων

Επεξήγηση των πιο πολύπλοκων σημείων του πηγαίου κώδικα, όταν αυτό δεν είναι εύκολο να γίνει με σχόλια μέσα σε αυτόν

Μεταφερσιμότητα (portability)

Με τον όρο εννοούμε την ιδιότητα του πηγαίου κώδικα να μπορεί να εκτελεστεί χωρίς αλλαγές σε διαφορετικά περιβάλλοντα (λειτουργικά συστήματα, τύπος υπολογιστή) στα οποία διατίθενται εκδόσεις της γλώσσας προγραμματισμού που χρησιμοποιείται.

Για να επιτευχθεί αυτό, είναι τις περισσότερες φορές αρκετό:

- να μη χρησιμοποιούνται συστατικά της γλώσσας που υποστηρίζονται μόνο από συγκεκριμένες υλοποιήσεις και
- να μη γίνονται αυθαίρετες παραδοχές για τη συμπεριφορά των υλοποιήσεων, δηλαδή παραδοχές που δεν περιγράφονται ρητά στο πρότυπο (standard) της γλώσσας (πράξεις με ακεραίους στη C – ακρίβεια τελεστέων).

Άσκηση

Αναφέρατε με συντομία τα έξι επιθυμητά χαρακτηριστικά του πηγαίου κώδικα, δίνοντας έναν τίτλο και μια περιγραφή 1–2 γραμμών για το καθένα.

Απάντηση

Τα επιθυμητά χαρακτηριστικά του πηγαίου κώδικα μπορούν να συνοψιστούν στα ακόλουθα:

Επάρκεια, δηλαδή λειτουργία του λογισμικού χωρίς σφάλματα.

Επίδοση, δηλαδή μεγάλη ταχύτητα εκτέλεσης του πηγαίου κώδικα.

Αναγνωσιμότητα, δηλαδή δυνατότητα κατανόησης του πηγαίου κώδικα από άλους.

Τεκμηρίωση, δηλαδή συνοδεία του πηγαίου κώδικα με έγγραφα που συμβάλλουν στην κατανόησή του.

Μεταφερσιμότητα, δηλαδή δυνατότητα του πηγαίου κώδικα να μπορεί να εκτελεστεί χωρίς αλλαγές σε διαφορετικά περιβάλλοντα (λειτουργικά συστήματα, υπολογιστές).

Δ**υνατότητα επαναχρησιμοποίησης**, δηλαδή χρήσης του πηγαίου κώδικα και σε άλλα συστήματα λογισμικού που θα κατασκευαστούν στο μέλλον (ΣΧΟΛΙΟ).

Πολλά χαρακτηριστικά θεωρούνται περιττά (ΣΧΟΛΙΟ)

Άσκηση

Αναφέρετε δύο τουλάχιστον λόγους για τους οποίους η αναγνωσιμότητα είναι ιδιαίτερα επιθυμητό χαρακτηριστικό του πηγαίου κώδικα.

Απάντηση

Δύο πολύ σημαντικοί λόγοι είναι:

(α) η δυνατότητα κατανόησης του πηγαίου κώδικα που έχει συγγράψει κάποιος και από έναν άλλο προγραμματιστή και (β) η δυνατότητα κατανόησης του κώδικα από τον ίδιο το συγγραφέα του μετά από λίγο καιρό.

οι προγραμματιστές συχνά αρνούνται να πειράξουν κώδικα που δεν είναι δικός τους

Έλεγχος και διόρθωση σφαλμάτων

Σκοπός του μαθήματος

Σκοπός του μαθήματος είναι

- η ανάδειξη της σημασίας του ελέγχου κατά την ανάπτυξη λογισμικού, καθώς και η παρουσίαση των ενεργειών που γίνονται κατά τη φάση του ελέγχου και των αποτελεσμάτων που παράγονται

Σε τι θα αναφερθούμε

- Στρατηγική του μαύρου κουτιού
- Στρατηγική του άσπρου ή γυάλινου κουτιού
- Δοκιμαστικά δεδομένα
- Πλάνο ελέγχου
- Περίπτωση ελέγχου
- Αναφορές ελέγχου

Έλεγχος Λογισμικού

Έλεγχος είναι η διαδικασία κατά την οποία εξετάζεται το λογισμικό με χρήση ειδικά σχεδιασμένων τεχνικών και με σκοπό την εύρεση και διόρθωση σφαλμάτων στην υλοποίησή του.

Συνταχτικά λάθη (Syntax errors) Παραβίαση syntax rules

Λογικά λάθη (Σημασιολογικά λάθη) Bugs – Debugging

Εκτελέσιμα λάθη (Run-time errors)

Τα σφάλματα που αναζητούνται σε αυτό το στάδιο ανάπτυξης κατηγοριοποιούνται σε δύο ομάδες.

Στην **πρώτη ομάδα** διακρίνουμε τα σφάλματα που παρουσιάζονται λόγω αντίφασης των αποτελεσμάτων της λειτουργίας του λογισμικού (σωστό πρόβλημα – λάθος λύση) με συγκεκριμένες απαιτήσεις και προδιαγραφές του συστήματος (**επικύρωση (validation)**).

Στη **δεύτερη ομάδα** διακρίνουμε τα σφάλματα που παρουσιάζονται κατά την εκτέλεση συγκεκριμένων μονάδων του λογισμικού **(επαλήθευση (verification))**.

Προγραμματισμός του ελέγχου

Μια δομή του εγγράφου **πλάνο ελέγχου**, βασισμένη σε πρότυπο του IEEE:

Πλάνο ελέγχου

01. Ταυτότητα του εγγράφου
02. Εισαγωγή
03. Οντότητες που θα ελεγχθούν
04. Χαρακτηριστικά που θα ελεγχθούν
05. Χαρακτηριστικά που δε θα ελεγχθούν
06. Μέθοδος
07. Κριτήρια επιτυχίας/αποτυχίας ελέγχου οντοτήτων
08. Κριτήρια ακύρωσης και προδιαγραφές επανάληψης ελέγχου
09. Παραδοτέα έγγραφα
10. Εργασίες που πρέπει να γίνουν
11. Αναγκαίοι πόροι περιβάλλοντος
12. Κατανομή ευθυνών για την εκτέλεση του ελέγχου
13. Ανάγκες στελέχωσης και εκπαίδευσης προσωπικού
14. Χρονοπρογραμματισμός ελέγχου
15. Κίνδυνοι και απρόοπτα
16. Εγκρίσεις

Τεχνικές ελέγχου – Ορισμοί

Ο έλεγχος μιας εφαρμογής λογισμικού στηρίζεται στην αρχή ότι εκτελείται ένα τμήμα αυτής με ένα σύνολο από δεδομένα εισόδου για τα οποία τα αποτελέσματα είναι γνωστά και, αν τα αποτελέσματα που λαμβάνονται από την εκτέλεση δεν είναι ίδια με τα αναμενόμενα, τότε το τμήμα αυτό έχει σφάλματα.

Η εκτέλεση μιας μονάδας προγράμματος με ένα σύνολο από δεδομένα για τα οποία τα αποτελέσματα είναι γνωστά λέγεται **δοκιμή (test)** της μονάδας. Δοκιμές που γίνονται με διαφορετικά δεδομένα θεωρούνται διαφορετικές.

Τα δεδομένα εισόδου και εξόδου που χρησιμοποιούνται κατά τις δοκιμές λέγονται **δοκιμαστικά δεδομένα (test data)**.

Μια **περίπτωση ελέγχου (test case)** είναι το σύνολο των δοκιμαστικών δεδομένων, των συνθηκών εκτέλεσης και των αναμενόμενων αποτελεσμάτων που έχουν σχεδιαστεί με ένα συγκεκριμένο σκοπό, όπως το να καλύψουν ένα μονοπάτι εκτέλεσης του λογισμικού ή να επικυρώσουν τη συμφωνία με μια συγκεκριμένη απαίτηση από αυτό.

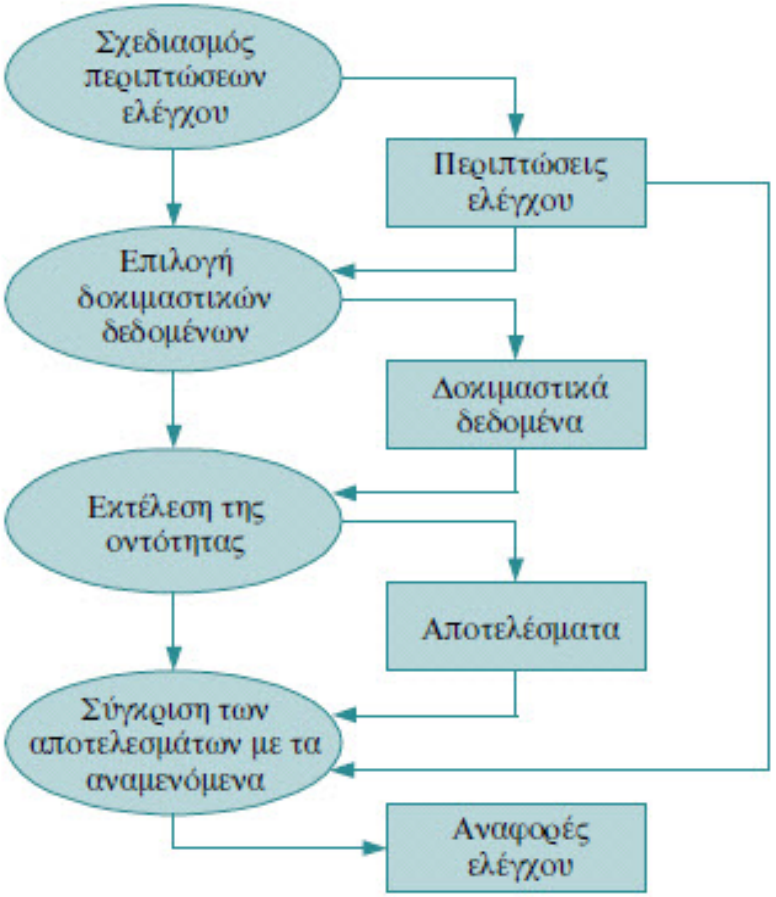
Περίπτωση ελέγχου

Στο Σχήμα φαίνεται η γενική ροή ελέγχου μιας μονάδας λογισμικού.

Σχεδιάζονται οι περιπτώσεις ελέγχου, (ορίζονται τα δοκιμαστικά δεδομένα, οι συνθήκες εκτέλεσης και τα αναμενόμενα αποτελέσματα για κάθε δοκιμή).

Εκτελείται το ελεγχόμενο λογισμικό με τα δοκιμαστικά δεδομένα κάθε περίπτωσης ελέγχου και λαμβάνονται κάποια αποτελέσματα, τα οποία συγκρίνονται με τα αναμενόμενα αποτελέσματα.

Σύνταξη αναφορών ελέγχου.



Στρατηγικές ελέγχου

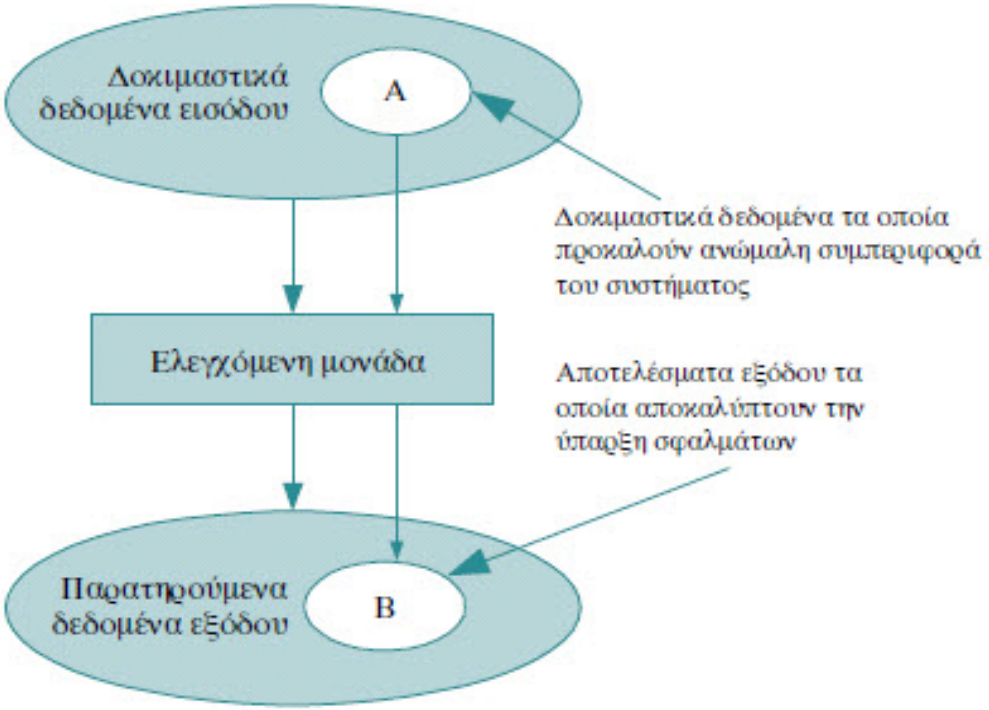
Δεν είναι εφικτό να δοκιμαστούν όλες οι δυνατές περιπτώσεις ελέγχου, καθώς ο αριθμός τους είναι απαγορευτικά μεγάλος.

Επομένως,θα πρέπει να γίνεται μια επιλογή ορισμένων μόνο από αυτές, η οποία θα πρέπει να είναι τέτοια ώστε να ικανοποιείται ο σκοπός του ελέγχου.

Υπάρχουν δύο βασικές στρατηγικές που ακολουθούνται στην πράξη για τη λύση του προβλήματος αυτού, οι οποίες διακρίνονται από την άποψη από την οποία παρατηρούν το λογισμικό: η **στρατηγική του μαύρου κουτιού (black box)** και η **στρατηγική του άσπρου ή γυάλινου κουτιού (white or glass box)**.

Στρατηγική του μαύρου κουτιού

Η **στρατηγική του μαύρου κουτιού (black-box testing)** βασίζεται στις προδιαγραφές της μονάδας λογισμικού που πρόκειται να ελεγχθεί, οι οποίες θεωρούνται γνωστές, ενώ ο τρόπος κατασκευής της θεωρείται άγνωστος. Αντιμετωπίζεται δηλαδή ως ένα «μαύρο κουτί» του οποίου η συμπεριφορά μπορεί να μελετηθεί μόνο παρατηρώντας τις εισόδους και τα αποτελέσματα που αντιστοιχούν σε αυτές.



Στρατηγική του μαύρου κουτιού

Άρα τι ελέγχεται?????

Τα δεδομένα εισόδου σε ένα λογισμικό σύστημα μπορούν συνήθως να κατηγοριοποιηθούν σε έναν αριθμό διαφορετικών κλάσεων. Οι κλάσεις αυτές έχουν κοινά χαρακτηριστικά (π.χ. θετικοί αριθμοί, αρνητικοί αριθμοί, συμβολοσειρές χωρίς κενά κτλ.). Μια μονάδα λογισμικού συνήθως συμπεριφέρεται με παρόμοιο τρόπο για όλα τα μέλη καθεμιάς απ' αυτές τις κλάσεις.

Δηλαδή ????

Εξαιτίας αυτής της «ισοδύναμης» συμπεριφοράς, οι κλάσεις αυτές ονομάζονται κλάσεις **ισοδύναμων τιμών (equivalence partitions)**.

■ **Κλάση ισοδύναμων τιμών**

Μια κλάση ισοδύναμων τιμών είναι ένα σύνολο με μέλη είτε δοκιμαστικά δεδομένα εισόδου είτε δεδομένα εξόδου, τα οποία προσδιορίζουν μια ισοδύναμη συμπεριφορά του συστήματος.

Στρατηγική του μαύρου κουτιού

Δηλαδή ????

Ορίζονται έγκυρες και άκυρες περιπτώσεις ελέγχου.

Άρα τι ελέγχεται?????

Μία τιμή για κάθε έγκυρη κλάση και μία τιμή για κάθε άκυρη κλάση ισοδύναμων τιμών.

Τι άλλο ????

Οι συνοριακές τιμές (boundary values) και μια ή δύο τιμές εκατέρωθεν αυτών.

Τι άλλο ????

Επιλογή τιμών με τη μέθοδο του «μαντέματος».

Δηλαδή ????

Άσκηση

Ας θεωρήσουμε την ακόλουθη μονάδα προγράμματος, γραμμένη σε Pascal, η οποία διαβάζει μια ημερομηνία από το πληκτρολόγιο και την επιστρέφει ως παράμετρο στη μονάδα που την καλεί. Να κατασκευαστούν περιπτώσεις ελέγχου σύμφωνα με τις προσεγγίσεις της «ισοδύναμης διαμέρισης» και των «συνοριακών τιμών». Μεταβλητές εισόδου είναι οι ακέραιες μεταβλητές d, m, y.

```
Procedure GetDate(var d,m,y: integer; var
     flag: integer);
    Var day, month, year: integer;
1. Begin
2.  day:=1;month:=1; year:=1900;flag:=0;
3.  Write('Δώσε ημερομηνία (η/μ/ε):');
4.  Read(day); Read(month); Readln(year);
5.  If ((month<1) or ((month>12) then
6.      Flag:=1;
7.  If ((day<1) or (day>31) then
8.      Flag:=1;
9.  If Flag=0 then
10.  Begin
11.    If month in [[1,3,5,7,8,10,12] then
12.       UpperDayValue:=31;
13.    If month in [[4,6,9,11] then
14.       UpperDayValue:=30;
15.    If month=2 then
16.       If leap(year) then
17.          UpperDayValue:=29
18.       Else
19.          UpperDayValue:=28;
20.    If day>UpperDayValue Then
21.      Flag:=1
22.    Else
23.       Begin
24.          D:=day;
25.          M:=month;
26.          Y:=year
27.       End;
28.   End;
29. End;
```

Απάντηση

Λόγω της έννοιας που αποδίδεται στις μεταβλητές αυτές (π.χ. στη μεταβλητή «d» αποδίδεται η έννοια της «ημέρας»), προκύπτουν κάποιες συνθήκες εισόδου, δηλαδή προδιαγράφονται ορισμένα διαστήματα τιμών. Σύμφωνα με αυτά τα διαστήματα τιμών, καταγράφουμε τις έγκυρες αλλά και τις άκυρες κλάσεις ισοδύναμων τιμών. Επίσης προκύπτουν και οι συνοριακές τιμές. Έτσι, έχουμε τα εξής:

Μεταβλητή	Συνθήκη εισόδου	Κλάσεις ισοδύναμων τιμών	
		Έγκυρες	Άκυρες
day	$1 \leq day \leq 31$	$1 \leq day \leq 31$	$day < 1$ $day > 31$
month	$1 \leq month \leq 12$	$1 \leq month \leq 12$	$month < 1$ $month > 12$

Συνοριακές τιμές: 1, 31, 1, 12.

Απάντηση

Προσέγγιση ισοδύναμης διαμέρισης

Για την κατασκευή περιπτώσεων ελέγχου αρκεί να επιλέξουμε για κάθε μεταβλητή μια τιμή από κάθε κλάση ισοδύναμων τιμών της. Οπότε, ένα ενδεικτικό σύνολο περιπτώσεων ελέγχου είναι το παρακάτω:

Περίπτωση ελέγχου	**Δοκιμαστικά δεδομένα εισόδου**			**Αναμενόμενα αποτελέσματα**
Id#	day	month	year	flag
1	5	3	1950	0
2	5	13	1950	1
3	5	–2	1950	1
4	36	3	1950	1
5	–7	3	1950	1
6	–7	13	1950	1
7	–7	–2	1950	1

Απάντηση

Προσέγγιση συνοριακών τιμών

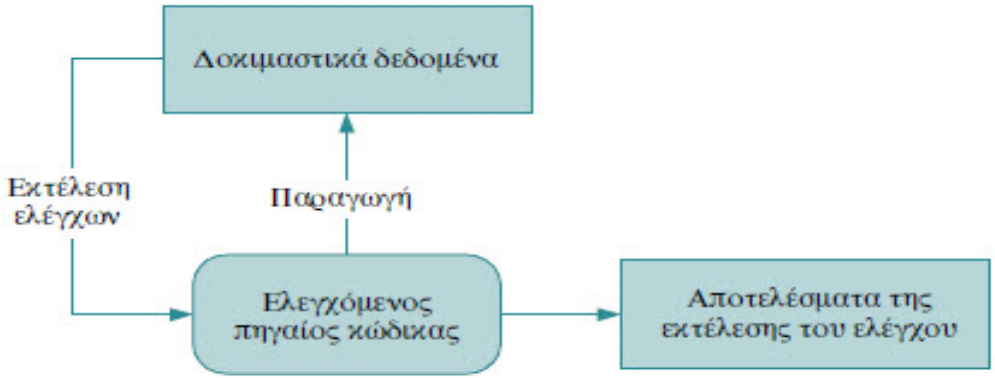
Σύμφωνα με την προσέγγιση «συνοριακών τιμών», για την κατασκευή των περιπτώσεων ελέγχου επιλέγουμε έγκυρες περιπτώσεις για τα άκρα του κάθε διαστήματος και άκυρες για τιμές ακριβώς έξω από τα άκρα. Οπότε, ένα ενδεικτικό σύνολο περιπτώσεων ελέγχου είναι το παρακάτω:

Περίπτωση ελέγχου	**Δοκιμαστικά δεδομένα εισόδου**			**Αναμενόμενα αποτελέσματα**
Id#	day	month	year	flag
1	1	1	1900	0
2	1	12	1900	0
3	31	1	1900	0
4	31	12	1900	0
5	1	0	1900	1
6	1	13	1900	1
7	31	0	1900	1

Στρατηγική του άσπρου κουτιού

Στη στρατηγική αυτή το λογισμικό που ελέγχεται αντιμετωπίζεται ως ένα **άσπρο κουτί (white box)**, ή καλύτερα ως ένα **γυάλινο κουτί (glass box)**. Αυτό, γιατί θεωρείται γνωστή (είναι ορατή) η δομή και ο τρόπος λειτουργίας του και δεν αποτελεί «μαύρο κουτί», όπως στην προηγούμενη περίπτωση.

Η επιλογή των περιπτώσεων ελέγχου γίνεται μετά από μελέτη του κώδικα και της δομής τής υπό έλεγχο οντότητας. Απαραίτητα για την εργασία αυτή είναι το λεπτομερές σχέδιο της υπό έλεγχο μονάδας λογισμικού, καθώς και ο πηγαίος της κώδικας.



Στρατηγική του άσπρου κουτιού

Για την επιλογή περιπτώσεων ελέγχου, στη στρατηγική αυτή χρησιμοποιούνται διάφορα κριτήρια, τα οποία λέγονται «κριτήρια κάλυψης». Βέβαια, αυτονόητο είναι ότι όσο περισσότερες περιπτώσεις ελέγχου επιλέγονται για το κάθε κριτήριο, με τόσο περισσότερη ασφάλεια μπορεί να εγγυηθεί κάποιος ότι το έχει καλύψει.

Μερικά χρήσιμα κριτήρια κάλυψης είναι τα ακόλουθα:

Όλα τα ανεξάρτητα μονοπάτια εκτέλεσης του ελεγχόμενου κώδικα πρέπει να εκτελούνται τουλάχιστον μια φορά. Δηλαδή.

Στρατηγική του άσπρου κουτιού

Δεν πρέπει να υπάρχουν γραμμές πηγαίου κώδικα των οποίων η εκτέλεση να μην ελέγχεται:

Όλες οι διακλαδώσεις ροής με εντολές τύπου if-then-else πρέπει να εκτελούνται τουλάχιστον μια φορά, τόσο για την περίπτωση αληθούς συνθήκης όσο και ψευδούς.

Όλοι οι βρόχοι επανάληψης θα πρέπει να εκτελούνται και για οριακές τιμές επαναλήψεων, αλλά και για αριθμό επαναλήψεων εντός των ορίων. Για παράδειγμα, ένας βρόγχος με μέγιστο αριθμό επαναλήψεων ν θα πρέπει να εκτελεστεί για 0, 1, 2, μ (όπου μ < ν), ν - 1, ν και ν + 1 επαναλήψεις.

Γενικά, η εμπειρία δείχνει ότι ο εξαντλητικός έλεγχος του «μαύρου κουτιού» είναι καλύτερος απ' αυτόν του «άσπρου κουτιού».

Παράδειγμα (άσπρο κουτί)

Η μονάδα που ακολουθεί λύνει τη δευτεροβάθμια εξίσωση $ax2+ bx + c = 0$.
Ο κώδικας της μονάδας, γραμμένος σε γλώσσα Pascal, φαίνεται παρακάτω:

```
   Procedure equ (a,b,c:real; var x1,x2:real; var flag:integer);
       var d:real;
       begin
1              d=b*b–4*a*c;
2              if (d<0) or (a=0) then flag:=2
3              else if d>0 then flag:=0
4              else flag:=1;
5              if flag=0 then
6                 begin
7                    x1:=(–b+sqrt(d))/(2*a);  x2:=(–b–sqrt(d))/(2*a);
8                    writeln(“Η εξίσωση έχει 2 διαφορετικές λύσεις”)
9                 end
10             else if flag=1 then
11                begin
12                   x1:=(–b)/(2*a);   x2:=x1;
13                   writeln(“Η εξίσωση έχει μια διπλή λύση”)
14                end
15             else writeln(“Η εξίσωση δεν έχει λύση στο R ή δεν έχει διπλή λύση”);
       end;
```

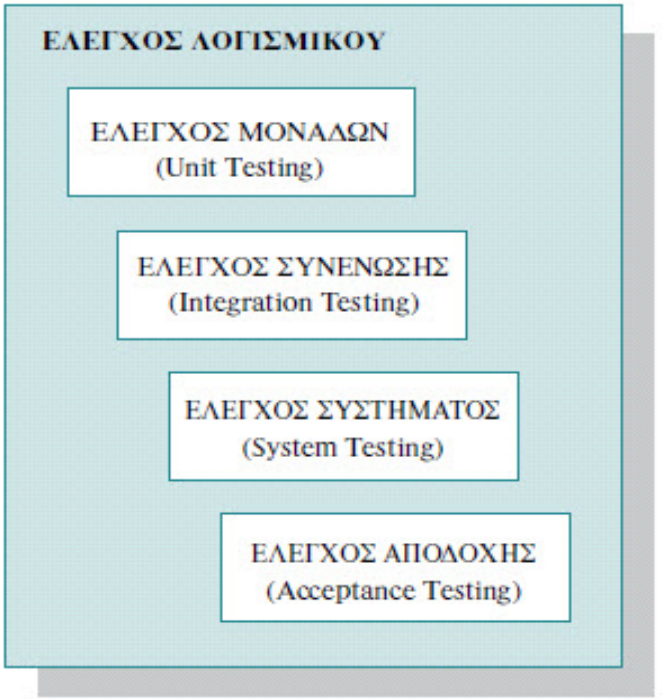
Απάντηση

Δοκιμαστικά δεδομένα			Κάλυψη εντολών (μονοπάτι εκτέλεσης)	Κάλυψη συνθηκών (Εντολές απόφασης – τιμές συνθηκών)
a	b	c		
0	1	2	1, 2, 3, 5, 10, 15	2-T+T, 3-F, 5-F, 10-F, 15-T
1	3	1	1, 2, 3, 5, 6, 7, 8, 9	2-F+F, 3-T, 5-T
4	–4	1	1, 2, 3, 4, 5, 10, 11, 12, 13, 14	2-F+F, 3-F, 4-T, 5-F, 10-T

Εκτέλεση / Στάδια ελέγχου

Ο έλεγχος μιας εφαρμογής λογισμικού πραγματοποιείται σε τέσσερα στάδια.

Τα στάδια αυτά, από το κατώτερο επίπεδο στο ανώτερο, είναι ο έλεγχος μονάδας (unit testing), ο έλεγχος συνένωσης (integration testing), ο έλεγχος συστήματος (system testing) και ο έλεγχος αποδοχής (acceptance testing).

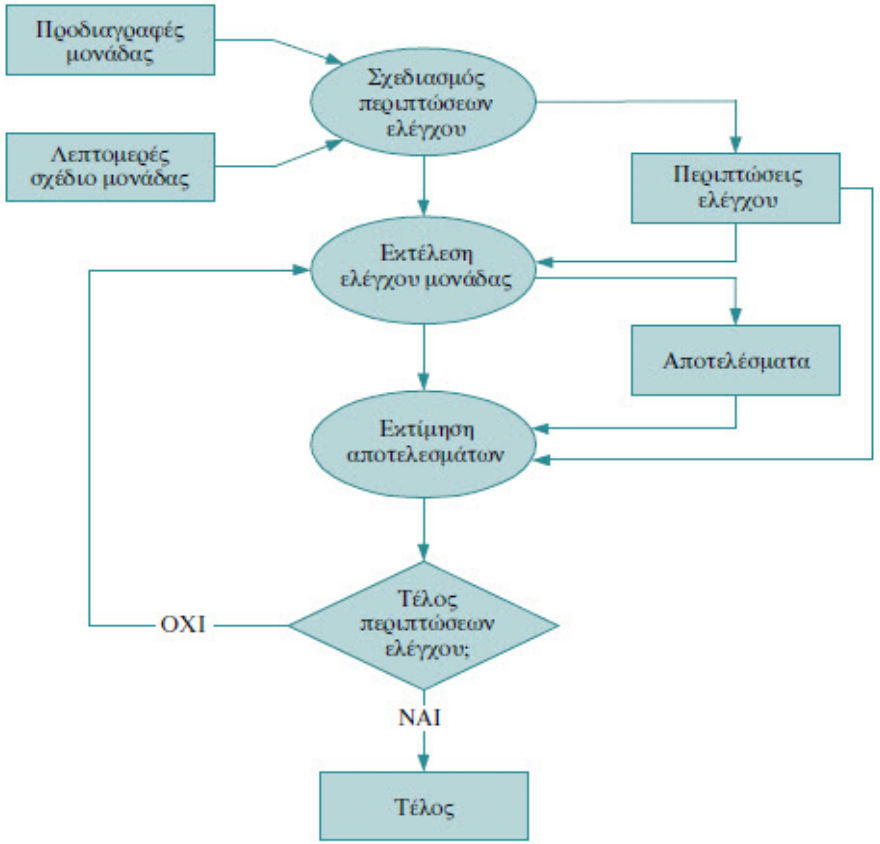


Έλεγχος μονάδας

Κατά τον έλεγχο μονάδας κάθε μονάδα του λογισμικού δοκιμάζεται ανεξάρτητα από τις άλλες με σκοπό να διαπιστωθεί αν πληροί τις προδιαγραφές της (Σχήμα).

Για τη διεκπεραίωση αυτού του επιπέδου ελέγχου μπορεί να χρησιμοποιηθεί οποιαδήποτε από τις δύο στρατηγικές ελέγχου ή και οι δύο, προκειμένου να ελεγχθούν επιπλέον περιπτώσεις με σκοπό την εξασφάλιση της ορθής λειτουργίας της.

Για την εκτέλεση του «ελέγχου μονάδας» χρησιμοποιούνται διάφορες τεχνικές.



Μη αυξητική τεχνική

Στη **μη αυξητική τεχνική** κάθε μονάδα ελέγχεται μεμονωμένα και εντελώς ανεξάρτητα από τις άλλες. Για τη διεκπεραίωση του ελέγχου μονάδας με αυτή την τεχνική, είναι απαραίτητη η δημιουργία καινούριων βοηθητικών μονάδων.

Για κάθε μονάδα που ελέγχεται θα πρέπει να κατασκευάζεται μια βοηθητική μονάδα, η οποία θα την καλεί, θα της παρέχει τα δοκιμαστικά δεδομένα εισόδου και θα τυπώνει τα αποτελέσματα της εκτέλεσης σε κάποια έξοδο. Αυτή η βοηθητική μονάδα ονομάζεται **οδηγός (driver)**.

Στην περίπτωση που η ελεγχόμενη μονάδα περιέχει κλήσεις σε άλλες μονάδες, τότε απαιτούνται και άλλες βοηθητικές μονάδες, οι οποίες θα πρέπει να κατασκευάζονται εκ των προτέρων, έτσι ώστε να κωδικοποιείται η επικεφαλίδα τους, να περνάνε τα ορίσματα και να επιστρέφουν κάποιες τυπικές τιμές. Αυτές οι καινούριες μονάδες ονομάζονται **στελέχη (stubs)**.

Έλεγχος συνένωσης

Ο έλεγχος συνένωσης (integration testing) είναι η διαδικασία κατά την οποία ελέγχονται σύνθετα τμήματα (ολόκληρη ενότητα), μέρη του υπό ανάπτυξη λογισμικού. Ένα τέτοιο τμήμα περιλαμβάνει περισσότερες από μια μονάδες. Αυτό που ενδιαφέρει σε αυτό το στάδιο του ελέγχου είναι να διαπιστωθεί αν οι μονάδες λογισμικού συνεργάζονται ομαλά κατά την ικανοποίηση κάποιας λειτουργικής απαίτησης.

Η επιλογή των απαιτούμενων περιπτώσεων ελέγχου είναι απαραίτητο να εξασφαλίζει την εκτέλεση όλων των μονάδων της ελεγχόμενης ενότητας, καθώς και όλων των εντολών κλήσεων που περιέχονται στις μονάδες αυτές, τουλάχιστον μια φορά.

Άσκηση

Στην περίπτωση που ο έλεγχος μονάδας έχει γίνει σωστά με τη μη αυξητική τεχνική, θεωρείται ότι χρειάζεται να γίνει «έλεγχος συνένωσης»;

Απάντηση

Προφανώς όχι.

Αφού ο έλεγχος μονάδας υποθέτει ότι έχουν συγγραφεί οδηγοί (drivers) και στελέχη (stubs) οπότε δεν τίθεται ζήτημα συνεργασίας εκάστης μονάδας με τις συνδεόμενες σε αυτή μονάδες.

Αυξητική τεχνική

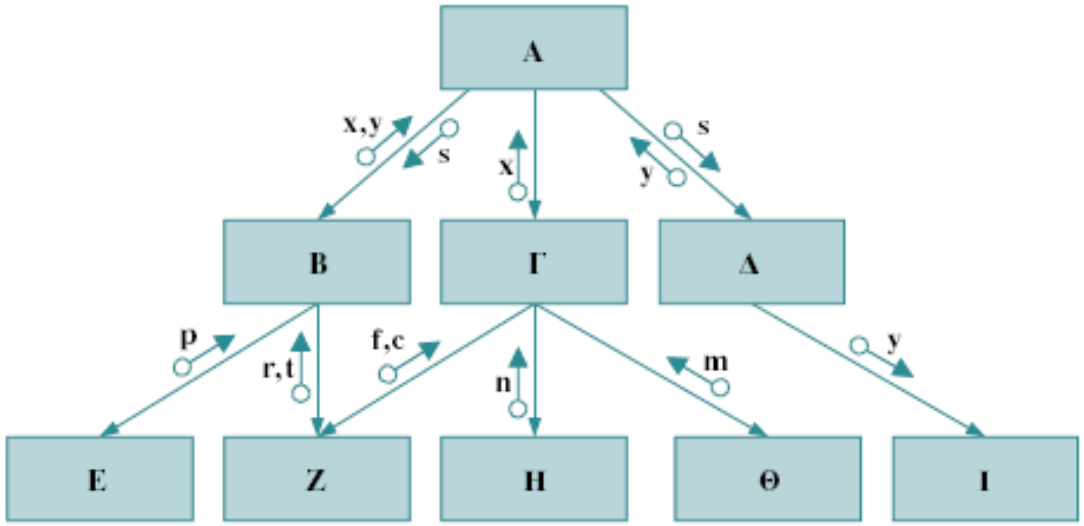
Σύμφωνα με την αυξητική τεχνική, αφού κωδικοποιηθεί μια μονάδα, εκτελείται το πρώτο επίπεδο ελέγχου για αυτή (έλεγχος μονάδας). Στη συνέχεια ενώνεται με την ενότητα στην οποία ανήκει και εκτελείται ο «έλεγχος συνένωσης».

Η διαδικασία επαναλαμβάνεται για κάθε καινούρια μονάδα που κωδικοποιείται, μέχρι να κατασκευαστεί ολόκληρο το λογισμικό.

Η αυξητική τεχνική μπορεί να εφαρμοστεί με δύο τρόπους: από **πάνω προς τα κάτω (top–down incremental)** ή από **κάτω προς τα πάνω (bottom–up incremental)**, ανάλογα με τη σειρά με την οποία επιλέγεται να ελεγχθούν οι μονάδες του λογισμικού.

Άσκηση

Ας υποθέσουμε το παρακάτω διάγραμμα δομής. Με ποιόν τρόπο θεωρείτε ότι θα γίνει ο έλεγχος συνένωσης με την από πάνω προς τα κάτω (top–down incremental) ή από κάτω προς τα πάνω (bottom–up incremental) αυξητική τεχνική;



Απάντηση

Μια πιθαμή

Top–down incremental

Bottom–up incremental

Βήμα	Υποσύστημα που ελέγχεται
1	Α
2	ΑΒ
3	ΑΒΓ
4	ΑΒΓΔ
5	ΑΒΓΔΕ
6	ΑΒΓΔΕΖ
7	ΑΒΓΔΕΖΗ
8	ΑΒΓΔΕΖΗΘ
9	ΑΒΓΔΕΖΗΘΙ

Βήμα	Υποσύστημα που ελέγχεται
1	Ε
2	ΕΖ
3	ΕΖΗ
4	ΕΖΗΘ
5	ΕΖΗΘΙ
6	ΕΖΗΘΙΒ
7	ΕΖΗΘΙΒΓ
8	ΕΖΗΘΙΒΓΔ
9	ΕΖΗΘΙΒΓΔΑ

Έλεγχος συστήματος

Στον «έλεγχο συστήματος» η οντότητα που ελέγχεται είναι πλέον ολόκληρη η εφαρμογή λογισμικού. Στο επίπεδο αυτό του ελέγχου εκτελούνται δοκιμές και γίνονται διάφορες επιδείξεις και αναλύσεις, με σκοπό να διαπιστωθεί αν το σύστημα που κατασκευάστηκε πληροί τις προδιαγραφές του. Δεν πρέπει να ξεχνάμε ότι πάντα ο στόχος σε αυτή τη φάση ανάπτυξης του συστήματος είναι η αποκάλυψη σφαλμάτων.

Σε αυτό το επίπεδο ελέγχου χρησιμοποιείται η **«στρατηγική του μαύρου κουτιού»** για την επιλογή των περιπτώσεων ελέγχου. Οι περιπτώσεις αυτές θα πρέπει να είναι τέτοιες ώστε να ελέγχονται όλες οι απαιτήσεις που περιγράφονται στο «έγγραφο προδιαγραφών των απαιτήσεων από το λογισμικό», δηλαδή όλες οι λειτουργικές και οι μη λειτουργικές απαιτήσεις.

Έλεγχοι μη λειτουργικών απαιτήσεων

Έλεγχοι επίδοσης (Performance testing). Στις δοκιμές αυτές το σύστημα δοκιμάζεται με ένα σταθερό φορτίο, με σκοπό το συντονισμό και τη βελτιστοποίηση των επιδόσεων του λογισμικού. Οι μετρήσεις που γίνονται συνήθως περιλαμβάνουν τον αριθμό των συναλλαγών, τον αριθμό των χρηστών, το μέγεθος των βάσεων δεδομένων στις οποίες γίνεται πρόσβαση κ.ά.

Έλεγχοι πίεσης (Stress testing). Στις δοκιμές αυτές ελέγχονται διάφορες λειτουργίες του λογισμικού κατά την εμφάνιση ανώμαλων συνθηκών στην εκτέλεση του συστήματος (πχ. η στιγμιαία εμφάνιση ενός υψηλού φορτίου, όπως η ταυτόχρονη εκδήλωση πολλών γεγονότων σ' ένα σύστημα αυτόματου ελέγχου, καθώς και η έλλειψη μνήμης ή διάφορων υλικών πόρων του συστήματος.

Έλεγχοι ασφάλειας (Security testing). Οι δοκιμές αυτές ελέγχουν κατά πόσο οι πληροφορίες που περιέχει ένα λογισμικό σύστημα είναι προσβάσιμες από άτομα στα οποία δε θα πρέπει να παρέχεται δυνατότητα πρόσβασης στο σύστημα ή στις συγκεκριμένες πληροφορίες.

Έλεγχοι μη λειτουργικών απαιτήσεων

Έλεγχοι όγκου δεδομένων (Volume testing). Σε αυτές τις δοκιμές ελέγχεται η δυνατότητα του λογισμικού να δουλεύει με μεγάλους όγκους δεδομένων (είτε δεδομένα εισόδου είτε δεδομένα εξόδου είτε δεδομένα τα οποία είναι ήδη εγκατεστημένα σε κάποια βάση δεδομένων του λογισμικού).

Έλεγχοι ανάκτησης (Recovery testing). Στις δοκιμές αυτές ελέγχεται η δυνατότητα του λογισμικού να ανακτά τη λειτουργία του ύστερα από αιφνίδια σταματήματα.

Έλεγχος αποδοχής

Ο έλεγχος αποδοχής είναι ένα υποσύνολο του ελέγχου συστήματος.

Στην περίπτωση αυτή οι περιπτώσεις ελέγχου είναι ένα υποσύνολο των αντίστοιχων περιπτώσεων του ελέγχου συστήματος και για την επιλογή τους είναι αποκλειστικά υπεύθυνος ο πελάτης, στις εγκαταστάσεις του οποίου μπορεί και να πραγματοποιείται ο «έλεγχος αποδοχής».

Ο στόχος αυτού του επιπέδου ελέγχου είναι να πειστεί ο πελάτης ότι το λογισμικό που κατασκευάστηκε για λογαριασμό του πληροί τις προδιαγραφές που τέθηκαν, η δε διάρκεια αυτού του ελέγχου μπορεί να είναι αρκετά μεγάλη, μέχρις ότου ικανοποιηθεί ο πελάτης.

Στην πράξη δεν είναι λίγες οι περιπτώσεις όπου ο «έλεγχος συστήματος» ταυτίζεται με τον «έλεγχο αποδοχής».

Θεωρείτε το παραπάνω σωστό ή λάθος??

Αναφορές ελέγχου

Οι αναφορές σφαλμάτων είναι ένα σημαντικό τμήμα της τεκμηρίωσης του ελέγχου. Η σύνταξη αυτών των αναφορών αποτελεί το τελευταίο στάδιο της διαδικασίας του ελέγχου.

Ο σκοπός μιας τέτοιας αναφοράς είναι να καταγράψει με λεπτομέρεια ένα γεγονός που συμβαίνει κατά την εκτέλεση του ελέγχου, το οποίο υποδεικνύει την ύπαρξη ενός σφάλματος, ώστε να συμβάλλει στην αποτελεσματική διόρθωσή του. Μια τέτοια αναφορά συντάσσεται για κάθε σφάλμα το οποίο αποκαλύπτεται κατά την εκτέλεση ενός ελέγχου. Μια δομή της αναφοράς σφάλματος προτείνεται παρακάτω:

Αναφορά σφάλματος

1. Ταυτότητα του εγγράφου
2. Ανακεφαλαίωση του ελέγχου που εκτελέστηκε
3. Περιγραφή του ελέγχου και του σφάλματος
4. Επιπτώσεις

Διόρθωση σφαλμάτων

Ο έλεγχος και η διόρθωση των σφαλμάτων είναι δυο πολύ σημαντικά στάδια της ανάπτυξης ενός λογισμικού. **Μελέτες έχουν αποδείξει ότι οι δύο αυτές εργασίες μπορεί να κοστίσουν μέχρι και το 50% του προϋπολογισμού για την κατασκευή μιας εφαρμογής λογισμικού.**

Με σκοπό τη διόρθωση των σφαλμάτων που βρέθηκαν κατά τον έλεγχο χρησιμοποιούνται ειδικές τεχνικές (**debugging techniques**).

Με την **εισαγωγή εντολών για την εκτύπωση διαγνωστικών μηνυμάτων**, στον κώδικα του λογισμικού. Τέτοια μηνύματα δίνουν τις τιμές των μεταβλητών σε ορισμένα κρίσιμα σημεία του προγράμματος και επιβεβαιώνουν ότι η εκτέλεσή του έχει περάσει από τα σημεία αυτά.

Με τη χρησιμοποίηση **ιχνών (traces)**. Τα ίχνη καταγράφουν τις εντολές όπως αυτές εκτελούνται, καθώς και τις τιμές προκαθορισμένων μεταβλητών ανά πάσα στιγμή.

Με την τοποθέτηση **σημείων διακοπής (breakpoints)** στο πρόγραμμα. Στα σημεία αυτά διακόπτεται η εκτέλεση του προγράμματος και ο έλεγχος δίνεται στο τερματικό, από όπου ο χρήστης μπορεί να δει τις τιμές των μεταβλητών, των καταχωρητών,να αλλάξει τις τιμές αυτές κλπ.

Βιβλιογραφία

Τεχνολογία Λογισμικού (2000), Βασίλειος Βεσκούκης, Τόμος Α, ISBN: 960–538–097–8, Εκδόσεις Ελληνικό Ανοικτό Πανεπιστήμιο.

Ανάπτυξη Προηγμένων Πληροφοριακών Συστημάτων, (2007), David Avison, Guy Fitzgerald, Κωδικός Βιβλίου στον Εύδοξο: 1177, ISBN 960-8105-96-X, ΕΚΔΟΣΕΙΣ ΝΕΩΝ ΤΕΧΝΟΛΟΓΙΩΝ ΜΟΝ. ΕΠΕ.

Online Εξ Αποστάσεως Εκπαίδευση – Από τη Θεωρία στην Πράξη, (2015), Σοφός (Λοΐζος) Αλιβίζος, Κώστας Απόστολος, Παράσχου Βασίλειος, Εκδόσεις ΣΕΑΒ, Ελληνικά Ακαδημαϊκά Ηλεκτρονικά Συγγράμματα και Βοηθήματα (www. kallipos .gr).

Σχεδιασμός Εκπαιδευτικού Λογισμικού, (2005), Παναγιωτακόπουλος Χ., Πιερρακέας Χ., Πιντέλας Π., ISBN 978-960-375-579-1, Εκδόσεις Ελληνικό Ανοικτό Πανεπιστήμιο.

Πληροφοριακά Συστήματα, Σύγχρονη Ανάλυση & Σχεδίαση, (2016, 6η εκδ.), Hoffer J., George J., Valacich J., Κωδικός Βιβλίου στον Εύδοξο: 18548910, ISBN: 978-960-418-331-9, Εκδόσεις Τζιόλα.