ΚΕΦΑΛΑΙΟ 4

ΣΧΕΔΙΑΣΜΟΣ ΛΟΓΙΣΜΙΚΟΥ
(SOFTWARE DESIGN)

Σχεδιασμός Λογισμικού

- Ο Σχεδιασμός Λογισμικού χωρίζεται σε δύο φάσεις:

1) Εξωτερικό Σχεδιασμό (External Design).

2) Εσωτερικό Σχεδιασμό (Internal Design).

Εξωτερικός Σχεδιασμός

- Περιλαμβάνει τη σύλληψη και τον προσδιορισμό εξωτερικών χαρακτηριστικών ενός προϊόντος λογισμικού.
- Αυτά περιλαμβάνουν τις παρουσιάσεις στον χρήστη (user displays) και τυποποιήσεις αναφορών, τις εξωτερικές πηγές δεδομένων και εκβολές και τα λειτουργικά χαρακτηριστικά, τις απαιτήσεις εκτέλεσης και τη δομή επεξεργασίας σε υψηλό επίπεδο.
- Ο εξωτερικός σχεδιασμός αρχίζει κατά τη φάση της **ανάλυσης** και συνεχίζει στη φάση του **σχεδιασμού.**
- Ο εξωτερικός σχεδιασμός έχει σκοπό την εκλέπτυνση του προσδιορισμού απαιτήσεων και τον καθορισμό της δομής του συστήματος σε υψηλό επίπεδο.
- Ο διαχωρισμός μεταξύ του ορισμού των απαιτήσεων και του εξωτερικού σχεδιασμού δεν είναι σαφής. Απλώς η έμφαση περνάει από το **λεπτομερές "τι"** στο **υψηλού επιπέδου "πώς".**

Εσωτερικός Σχεδιασμός

- Οι δραστηριότητες του εσωτερικού σχεδιασμού χωρίζονται στις εξής κατηγορίες:

α) Αρχιτεκτονικός σχεδιασμός:
Γίνεται αναγνώριση όλων των **υποσυστημάτων** που συνθέτουν το σύστημα καθώς και των **σχέσεών** τους.

β) Περιληπτικός προσδιορισμός:
Για κάθε υποσύστημα προσδιορίζονται περιληπτικά οι υπηρεσίες που παρέχει και οι περιορισμοί κάτω από τους οποίους λειτουργεί.

γ) Σχεδιασμός διασύνδεσης (Interface design):
Σχεδιάζεται και τεκμηριώνεται η διασύνδεση (interface) κάθε υποσυστήματος με άλλα υποσυστήματα. Ο προσδιορισμός διασύνδεσης πρέπει να είναι σαφής και να επιτρέπει την χρήση του υποσυστήματος χωρίς γνώση της λειτουργίας του υποσυστήματος.

Εσωτερικός Σχεδιασμός

δ) Σχεδιασμός στοιχείων:

Οι υπηρεσίες που παρέχει ένα υποσύστημα χωρίζονται στα στοιχεία (μέρη) που συνθέτουν το υποσύστημα.

ε) Σχεδιασμός δομών δεδομένων:

Οι δομές δεδομένων σχεδιάζονται με λεπτομέρεια και προσδιορίζονται.

στ) Σχεδιασμός αλγορίθμων:

Οι αλγόριθμοι που χρησιμοποιούνται για να παρέχουν υπηρεσίες σχεδιάζονται με λεπτομέρεια και προσδιορίζονται.

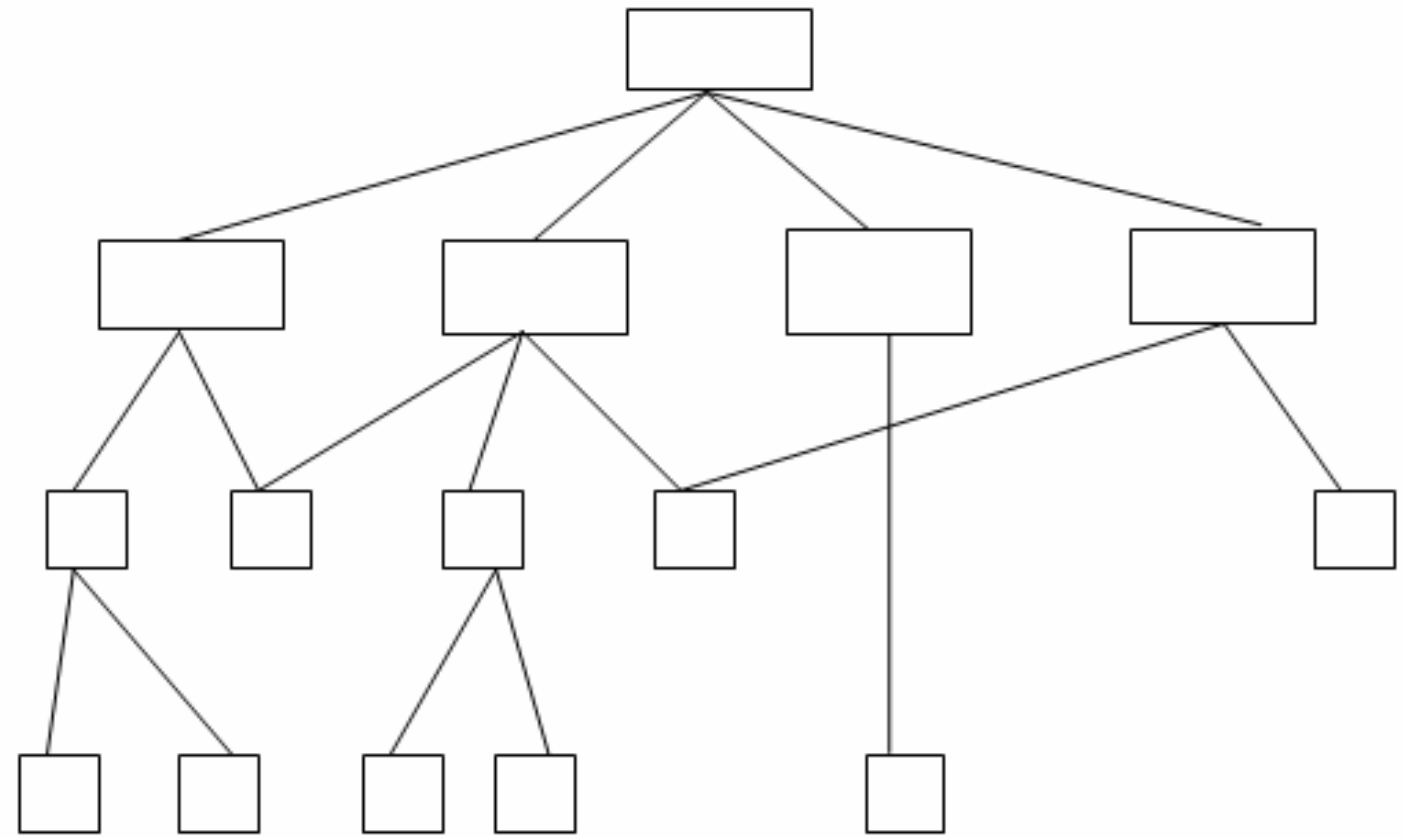
Προσεγγίσεις στο σχεδιασμό(1)

- **Από πάνω προς τα κάτω** (Top-down).

  Είναι μια προσέγγιση που θεωρείται πολύ καλή καθώς ένα **πρόβλημα χωρίζεται σε μικρότερα προβλήματα** μέχρις ότου φτάσουμε σε υποπροβλήματα που λύνονται.

  Η γενική μορφή αυτού του σχεδιασμού είναι **ιεραρχική:**

Ιεραρχική Μορφή



Προσεγγίσεις στο σχεδιασμό(2)

- **Από κάτω προς τα πάνω** (Bottom-up).

  Καμιά φορά χρησιμοποιείται και αυτή η προσέγγιση σε συνδιασμό με την **από πάνω προς τα κάτω.**

  Για παράδειγμα όταν ο σχεδιαστής χρησιμοποιεί την πείρα του από προηγούμενους σχεδιασμούς μπορεί να μην επεκταθεί στην πλήρη ανάλυση κάποιου μέρους του σχεδιασμού γιατί ίσως γνωρίζει τις λεπτομέρειες αυτού του μέρους από άλλο έργο.

Προσεγγίσεις στο σχεδιασμό(3)

- **<u>Αντικειμενοστρεφής προσέγγιση</u>.**

  Εάν υπάρχουν αρκετά αντικείμενα για επαναχρησιμοποίηση τότε ο σχεδιαστής χρησιμοποιεί τα **υπάρχοντα αντικείμενα** σαν ένα **σκελετό σχεδιασμού** και προσπαθεί να σχεδιάσει γύρω από αυτά. Δεν υπάρχει δηλαδή η έννοια της μοναδικής κορυφής.

Μερικές σημαντικές Έννοιες Σχεδιασμού

1. **Αφαίρεση** (abstraction).
2. **Ενθυλάκωση δεδομένων** (encapsulation).
3. **Συνεκτικότητα** (cohesion).
4. **Σύζευξη** (coupling).
5. **Απόκρυψη πληροφοριών** (information hiding).
6. **Τμηματικότητα** (modularity).

Αφαίρεση(1)
(abstraction)

- Η αφαίρεση είναι το διανοητικό εργαλείο που μας επιτρέπει να χειριστούμε **έννοιες** μακριά από **συγκεκριμένα παραδείγματα των εννοιών** αυτών.

- Στη διάρκεια της ανάλυσης και του σχεδιασμού η αφαίρεση επιτρέπει το διαχωρισμό των εννοιολογικών πλευρών του συστήματος από τις λεπτομέρειες υλοποίησης (που δεν έχουν προσδιοριστεί ακόμα).

Παράδειγμα:

Η ιδιότητα FIFO (First In First Out) της ουράς και η LIFO (Last In First Out) του σωρού μπορούν να χρησιμοποιηθούν χωρίς αναφορά στην υλοποίηση της ουράς και του σωρού.

Αφαίρεση(2)

- Κάτα τη διάρκεια του **αναλυτικού σχεδιασμού** ο **αρχιτεκτονικός σχεδιασμός** οδηγείται μέσω **εκλέπτυνσης** σε **λεπτομέρειες υλοποίησης**.

- Ο **σχεδιασμός** χρησιμοποιεί την **αφαίρεση** με τρόπο αντίθετο από τα μαθηματικά.

  Στο σχεδιασμό: από αφηρημένες έννοιες προχωράμε στη δημιουργία συγκεκριμένων εννοιών.

  Στα μαθηματικά: από πολλές συγκεκριμένες καταστάσεις οδηγούμαστε αφαιρετικά σε βασικές αρχές.

Μηχανισμοί Αφαίρεσης(1)

- **Έχουμε τρία είδη μηχανισμών αφαίρεσης.**

α) Λειτουργική αφαίρεση (functional abstraction).

β) Αφαίρεση δεδομένων (data abstraction).

γ) Αφαίρεση ελέγχου (control abstraction).

Μηχανισμοί Αφαίρεσης(2)

- **α) Λειτουργική αφαίρεση.**
  Η **λειτουργική αφαίρεση** περιλαμβάνει τη χρήση υποροutινών με παραμέτρους. Μια **ομάδα** υποπρογραμμάτων παρέχει λειτουργική αφαίρεση όπου οι **ορατές** ρουτίνες επικοινωνούν με άλλες ομάδες και οι **κρυμμένες** ρουτίνες υποστηρίζουν τις ορατές.

- **β) Αφαίρεση δεδομένων.**
  Περιλαμβάνει τον προσδιορισμό ενός **τύπου δεδομένων** (data type) ή **αντικειμένου** (object) με το να προσδιορίζει τις **λειτουργίες** στα αντικείμενα.

  π.χ. Ο **τύπος:** στοίβα μπορεί να προσδιοριστεί αφαιρετικά σαν ένας μηχανισμός LIFO όπου υπάρχουν οι **λειτουργίες: Νέα, Βάλε, Βγάλε, Κορυφή, Άδεια.**

Μηχανισμοί Αφαίρεσης(3)

γ) Αφαίρεση ελέγχου.

Χρησιμοποιείται για να δηλώνει ένα **επιθυμητό αποτέλεσμα** χωρίς να δηλώνει τον ακριβή μηχανισμό ελέγχου.

Για παράδειγμα: η WHILE εντολή στις πιο πρόσφατες γλώσσες είναι μια αφαίρεση των υλοποιήσεων σε γλώσσα μηχανής των εντολών διακλάδωσης υπό συνθήκη (jump διεύθυνση).

Ενθυλάκωση δεδομένων

- Χρησιμοποιείται για να δηλώσει ένα **μεμονωμένο δείγμα** ενός αντικειμένου δεδομένων που ορίζεται με βάση τις λειτουργίες που μπορούν να γίνουν πάνω του.
- Η ενθυλάκωση δεδομένων διαφέρει από τον αφηρημένο τύπο δεδομένων ο οποίος χρησιμοποιείται για να δηλώσει έναν τύπο δεδομένων (όπως ο σωρός) από τον οποίο φτιάχνονται πολλαπλά δείγματα.
- Η ενθυλάκωση δεδομένων περιλαμβάνει την **πακετοποίηση** μιας **δομής δεδομένων** και των **ρουτινών της πρόσβασης** σε ένα τμήμα (module).

Συνεκτικότητα – Βαθμίδες Συνεκτικότητας

- Η συνεκτικότητα είναι ένα μέτρο για το πόσο καλά "δένουν" τα κομμάτια του σχεδίου.

- Οι Constantine και Yourdon (1979) έχουν ορίσει εφτά βαθμίδες συνεκτικότητας ξεκινώντας από τη χαμηλότερη προς την υψηλότερη βαθμίδα.

1) Συμπτωματική συνεκτικότητα:

Όταν τα μέρη ενός τμήματος δεν έχουν σχέση μεταξύ τους αλλά απλώς βρίσκονται στο ίδιο τμήμα (ίσως λόγω χώρου μνήμης κ.λ.π.)

2) Λογική συσχέτιση:

Όταν τα μέρη ενός τμήματος εκτελούν παρόμοιες δουλειές όπως είσοδο δεδομένων, διαχείριση λαθών κ.λ.π. και γι' αυτό βρίσκονται μαζί

Βαθμίδες Συνεκτικότητας

3) Χρονική συνεκτικότητα:

Όταν όλα τα κομμάτια που ενεργοποιούνται την ίδια στιγμή βρίσκονται μαζί (π.χ. χρονική στιγμή έναρξης ή κλεισίματος).

4) Διαδικαστική συνεκτικότητα:

Τα στοιχεία του τμήματος αποτελούν μια σειρά ελέγχου.

5) Επικοινωνιακή συνεκτικότητα:

Όλα τα στοιχεία ενός τμήματος λειτουργούν πάνω στην ίδια εισαγωγή δεδομένων (input) ή παράγουν την ίδια εξαγωγή (output).

6) Σειριακή συνεκτικότητα:

Η έξοδος από ένα στοιχείο του τμήματος χρησιμοποιείται σαν είσοδος σε άλλο στοιχείο.

7) Λειτουργική συνεκτικότητα:

Κάθε μέρος του τμήματος είναι αναγκαίο για την εκτέλεση μιας μόνο λειτουργίας.

Συνεκτικότητα

- Η συνεκτικότητα είναι ένα επιθυμητό χαρακτηριστικό και για κάθε **αντικείμενο** σε έναν αντικειμενοστρεφή σχεδιασμό.

- Μία ακόμα κατηγορία συνεκτικότητας προτείνεται από τον Sommervile.

Συνεκτικότητα αντικειμένου:

Κάθε λειτουργία (operation) στο αντικείμενο προσφέρει λειτουργικότητα η οποία επιτρέπει στα χαρακτηριστικά του αντικειμένου να αλλαχθούν, να ελεγχθούν ή να χρησιμοποιηθούν για να παρέχουν κάποια υπηρεσία.

Σύζευξη

- Η σύζευξη σχετίζεται με την συνεκτικότητα.
- **Υψηλή σύζευξη** σε συστήματα υπάρχει όταν οι **ενότητες των προγραμμάτων εξαρτώνται η μία από την άλλη.**
- **Χαλαρή σύζευξη** υπάρχει όταν το σύστημα αποτελείται από ενότητες που είναι ανεξάρτητες ή σχεδόν ανεξάρτητες μεταξύ τους.
- Ο καλός σχεδιασμός προϋποθέτει έναν ελάχιστο βαθμό σύζευξης

Περιπτώσεις Υψηλής Σύζευξης

1) Οι ενότητες προγραμμάτων έχουν υψηλή σύζευξη εάν **ανταλλάσσουν πληροφορίες ελέγχου.** (Αυτό ονομάζεται "**σύζευξη ελέγχου**" κατά τους Yourdon-Constantine).

2) Υψηλή σύζευξη υπάρχει επίσης εάν οι ενότητες κάνουν **χρήση μεταβλητών που τις μοιράζονται μεταξύ τους**. (Αυτό ονομάζεται "**κοινή σύζευξη**" κατά τους Yourdon-Constantine).

3) Προβλήματα σύζευξης υπάρχουν επίσης όταν **ονόματα μεταβλητών δένονται με τιμές** αρκετά νωρίς κατά την ανάπτυξη του σχεδίου.

Ένα παράδειγμα
Προβλήματα Σύζευξης

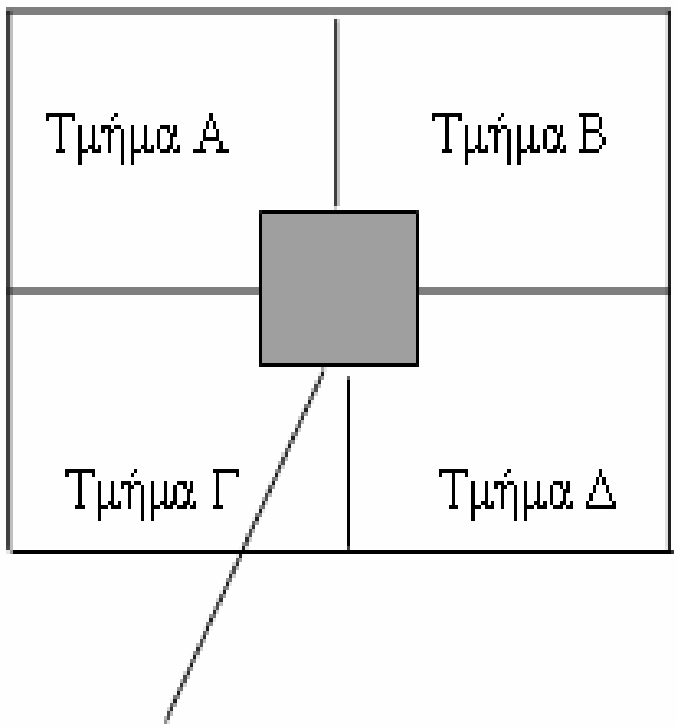
Εάν ένα πρόγραμμα ασχολείται με υπολογισμούς φόρων και κωδικοποιούμε ένα **ποσοστό φόρου** 30% σαν νούμερο μέσα στο πρόγραμμα, τότε το **πρόγραμμα έχει σύζευξη με το ποσοστό φόρου.**

Δηλαδή για να αλλάξουμε το ποσοστό φόρου θα πρέπει να αλλάξουμε το ίδιο το πρόγραμμα.

Το **σωστό** θα ήταν το πρόγραμμα **να διαβάζει το ποσοστό φόρου** κατά την διάρκεια της εκτέλεσης οπότε είναι εύκολο να γίνονται αλλαγές στο ποσοστό φόρου.

Σχηματική παράσταση σύζευξης

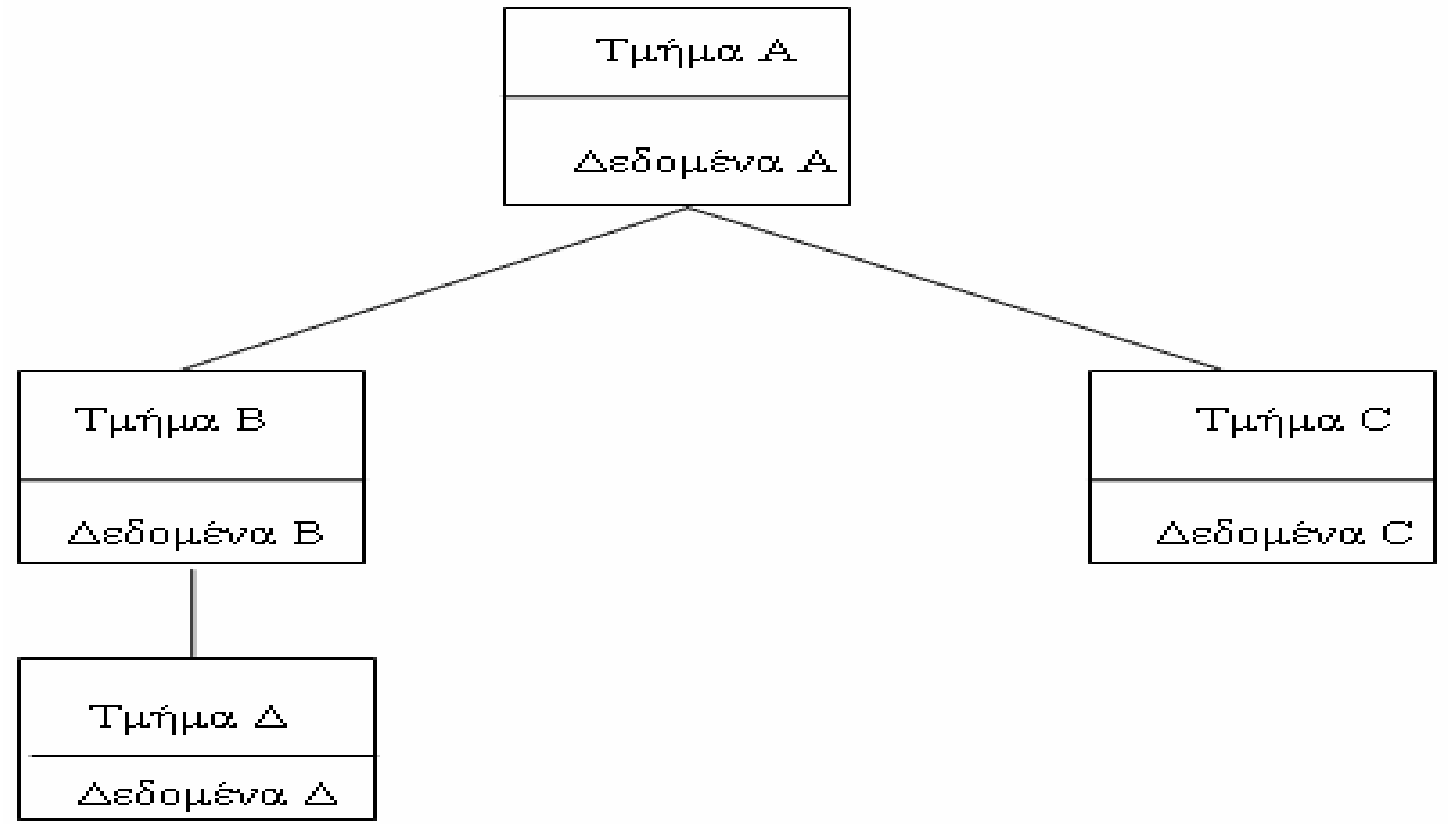
1) Υψηλή σύζευξη:



Περιοχή δεδομένων που μοιράζονται τα τμήματα Α, Β, Γ, Δ μεταξύ τους.

Σχηματική παράσταση σύζευξης

2) **Χαλαρή σύζευξη:** Χαλαρή σύζευξη επιτυγχάνεται όταν το κάθε τμήμα περιέχει όλες τις πληροφορίες αναπαράστασης που του χρειάζονται και η διασύνδεση δεδομένων (data interface) με άλλα τμήματα γίνεται μέσω της **λίστας παραμέτρων.**



Σύζευξη στον Αντικειμενοστρεφή Σχεδιασμό

- **Πλεονεκτήματα** του Αντικειμενοστρεφούς Σχεδιασμού:
  - Η φύση των αντικειμένων οδηγεί σε συστήματα με χαλαρή σύζευξη.
  - Θεμελιώδης αρχή του αντικειμενοστρεφούς σχεδιασμού είναι η αναπαράσταση του αντικειμένου να περιορίζεται στο αντικείμενο και να μην φαίνεται από άλλα εξαρτήματα.
  - Το σύστημα δεν χρειάζεται να έχει κοινές καταστάσεις και κάθε αντικείμενο μπορεί να αντικατασταθεί από άλλο αντικείμενο με την ίδια διασύνδεση (interface).

Σύζευξη στον Αντικειμενοστρεφή Σχεδιασμό

- **Μειονεκτήματα** του Αντικειμενοστρεφούς Σχεδιασμού:
  - Η **κληρονομικότητα** σε Αντικειμενοστρεφή συστήματα οδηγεί σε άλλης μορφής σύζευξη.
  - Τα **αντικείμενα** τα οποία κληρονομούν χαρακτηριστικά και λειτουργίες έχουν **σύζευξη** με τις υπέρ-τάξεις τους.
  - Αλλαγές στις υπερ-τάξεις πρέπει να γίνονται με προσοχή γιατί αυτές οι αλλαγές **διαδίδονται** σε όλες τις τάξεις που κληρονομούν.

Απόκρυψη Πληροφοριών (Information hiding)

- Είναι θεμελιώδης έννοια σχεδιασμού Software.
- Η αρχή της απόκρυψης δημιουργήθηκε από τον Parnas (PAR 71).
- Σύμφωνα με τον Parnas:

  Ο σχεδιασμός πρέπει να αρχίζει δημιουργώντας μια λίστα από δύσκολες **αποφάσεις σχεδιασμού** (design decision) και αποφάσεις σχεδιασμού που μπορεί να αλλάξουν. Ο σχεδιασμός των τμημάτων πρέπει να γίνεται έτσι ώστε να **κρύβονται οι αποφάσεις σχεδιασμού.**

Απόκρυψη Πληροφοριών

- Άλλα σημεία που προσφέρονται για απόκρυψη πληροφοριών:

  1)**Δομές δεδομένων:**

  η εσωτερική τους σύνδεση, οι λεπτομέρειες υλοποίησης (αφαίρεση δεδομένων).

  2)Οι **κώδικες χαρακτήρων**:

  και άλλες λεπτομέρειες υλοποίησης.κ.λ.π.

- Η απόκρυψη πληροφοριών χρησιμοποιείται:

  1) σαν βασική αρχή σχεδιασμού του **αρχιτεκτονικού σχεδίου**.

  2) σαν κριτήριο για **τμηματοποίηση** (modularization).

Τμηματικότητα (modularity)

- Υπάρχουν πολλοί ορισμοί για τον όρο "τμήμα" (module).

π.χ.

1) "Τμήμα" είναι μια υπορουτίνα στη FORTRAN.
2) "Τμήμα" είναι η ανάθεση εργασίας σε έναν προγραμματιστή στα πλαίσια ενός έργου.

- Οι ορισμοί αυτοί είναι σωστοί με την έννοια ότι τα **τμηματικά συστήματα** αποτελούνται από καλά-ορισμένες ενότητες με καλά-ορισμένη διασύνδεση (interface) ανάμεσα στις ενότητες

Ιδιότητες Τμηματικού Συστήματος

1) Κάθε **αφαίρεση** επεξεργασιών είναι ένα καλά-ορισμένο υποσύστημα (πιθανώς χρήσιμο και σε άλλες εφαρμογές).
2) Κάθε λειτουργία σε κάθε αφαίρεση έχει ένα μόνο σκοπό.
3) Κάθε λειτουργία διαχειρίζεται μόνο μία σημαντική δομή δεδομένων.
4) Οι **λειτουργίες** που διαχειρίζονται παραδείγματα (instances) των αφηρημένων τύπων δεδομένων είναι **ενθυλακωμένες** με την δομή δεδομένων υπό επεξεργασία.

Χαρακτηριστικά Τμήματος Λογισμικού

1) Ένα τμήμα περιέχει εντολές, λογική επεξεργασίας και δομές δεδομένων.

2) Τα τμήματα μπορούν να μεταφραστούν ξεχωριστά και να αποθηκευτούν σε βιβλιοθήκη.

3) Τα τμήματα μπορούν να συμπεριληφθούν σε ένα πρόγραμμα.

4) Τα τμήματα μπορούν να χρησιμοποιούν άλλα τμήματα.

Στρατηγικές Σχεδιασμού

- Αυτή τη στιγμή υπάρχουν κυρίως δύο στρατηγικές σχεδιασμού:

1) **Λειτουργικός σχεδιασμός** και

2) **Αντικειμενοστρεφής σχεδιασμός**

Λειτουργικός Σχεδιασμός

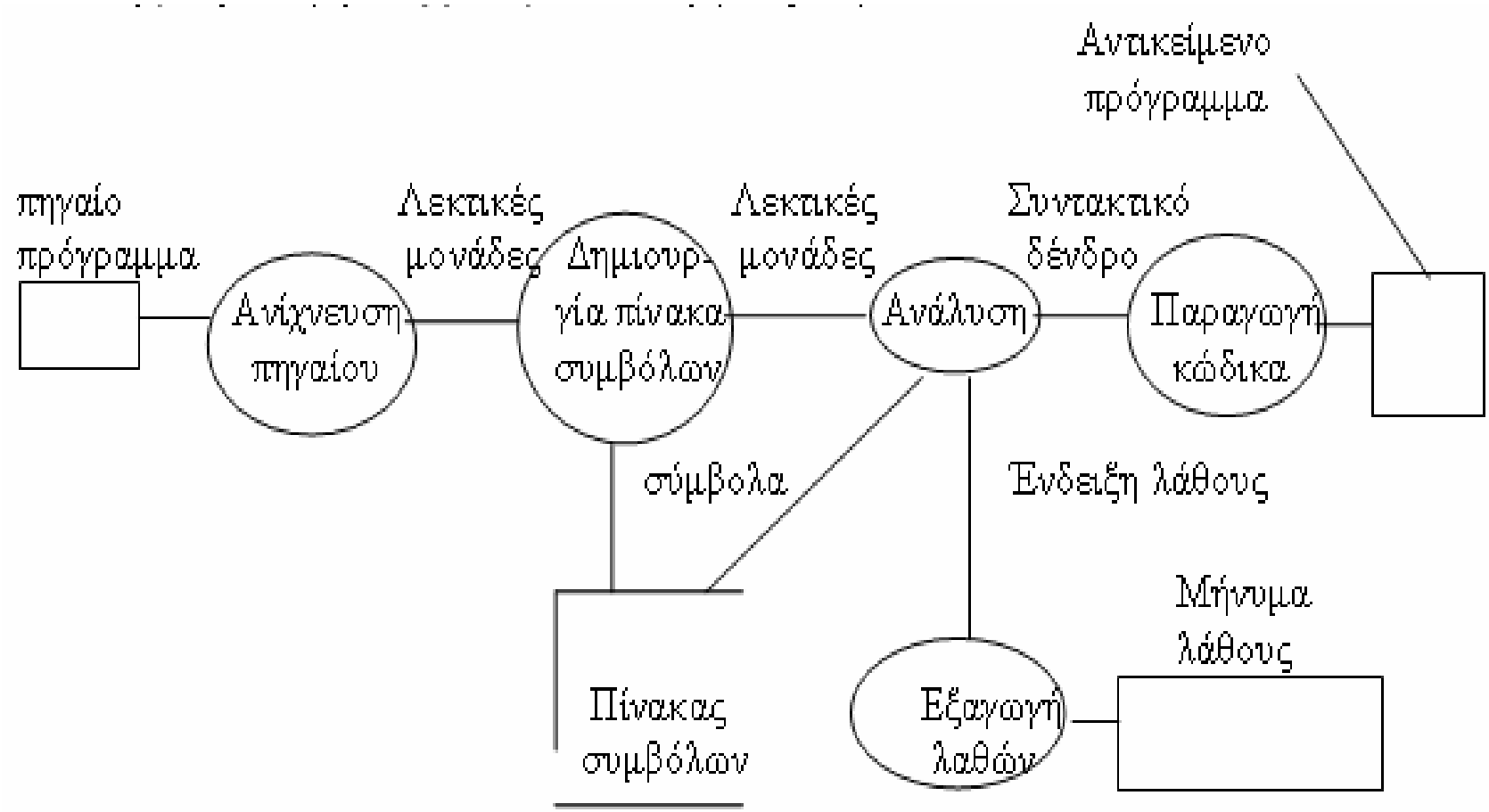
- Το σύστημα σχεδιάζεται από μια λειτουργική άποψη
  - ξεκινώντας από μια άποψη υψηλού επιπέδου και με προοδευτική εκλέπτυνση καταλήγουμε σε λεπτομερή σχεδιασμό.

- Αυτή η στρατηγική έχει αναλυθεί από το **δομημένο σχεδιασμό** (Constantine και Yourdon 1979) και τη **βηματική εκλέπτυνση** (Wirth 1971, 1976).

Αντικειμενοστρεφής Σχεδιασμός

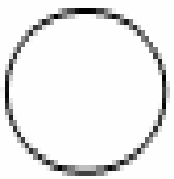
- Το σύστημα θεωρείται σαν μια συλλογή αντικειμένων αντί μιας συλλογής λειτουργιών.
- Ο αντικειμενοστρεφής σχεδιασμός βασίζεται στην ιδέα της απόκρυψης πληροφοριών (Parnas 1972).
- Ο σχεδιασμός Jackson (JSD) είναι κάτι μεταξύ των δύο μεγάλων στρατηγικών.

Λειτουργική άποψη ενός μεταγλωττιστή (compiler)



Λειτουργική άποψη ενός μεταγλωττιστή

Όπου:

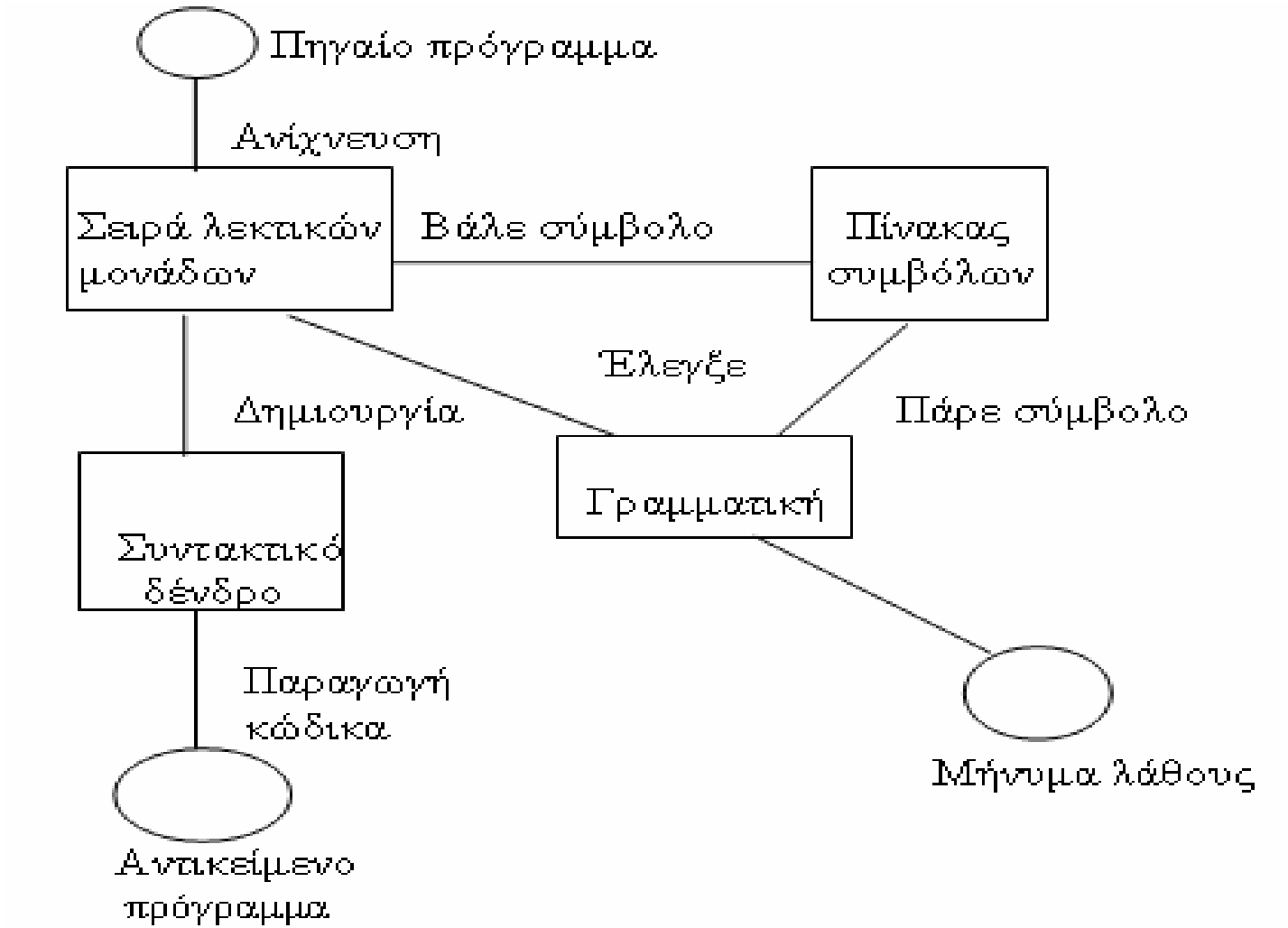


: Κόμβος επεξεργασίας

: Πηγές και εκβολές δεδομένων

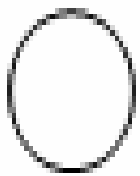
: Αρχείο ή Βάση δεδομένων

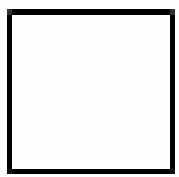
Αντικειμενοστρεφής άποψη ενός μεταγλωττιστή (compiler)

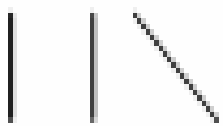


Αντικειμενοστρεφής άποψη ενός μεταγλωττιστή

Όπου:

 : Πηγές / εκβολές δεδομένων

 : Αντικείμενο

 : Μηνύματα μεταξύ αντικειμένων

Δομημένος Σχεδιασμός
(Structured Design)

- Είναι μια μέθοδος που αναπτύχθηκε από τον Constantine σαν μια τεχνική από πάνω προς τα κάτω (top-down) για τον **αρχιτεκτονικό σχεδιασμό** λογισμικού.

- Η βασική προσέγγιση της μεθόδου:

  Η συστηματική μετατροπή των ΔΡΔ (DFD) σε **διαγράμματα δομής** (structure charts).

- Ο σχεδιασμός ακολουθεί κάποια κριτήρια ποιότητας όπως χαμηλή σύζευξη και υψηλή συνεκτικότητα.

Βήματα δομημένου σχεδιασμού

- 1) Εποπτική θεώρηση και εκλέπτυνση των ΔΡΔ που αναπτύχθηκαν στη φάση της Ανάλυσης Απαιτήσεων.

- 2) α. Προσδιορίζουμε αν το σύστημα είναι **επικεντρωμένο σε μεταβολή**(transform-centered) ή εάν είναι **οδηγούμενο από διεξαγωγή** (transaction-driven).

  β. Κατασκευάζουμε ένα **διάγραμμα δομής** ανάλογα με την περίπτωση.

- 3) Αποσύνθεση του κάθε υποσυστήματος χρησιμοποιώντας οδηγίες σχεδιασμού όπως τα κριτήρια συνεκτικότητας, σύζευξης κ.λ.π.

Επεξήγηση 2ου Βήματος

- Πρέπει να καθορίσουμε αν το σύστημα είναι:

  α) **επικεντρωμένο σε μεταβολή** ή

  β) **οδηγούμενο από διεξαγωγή**.

Χαρακτηριστικά των 2 κατηγοριών(1)

α) **Επικεντρωμένο σε μεταβολή** (transform-centered)

- Εδώ το ΔΡΔ περιέχει Εισαγωγή Δεδομένων, Επεξεργασία και Εξαγωγή δεδομένων τα οποία μετατρέπονται στα αντίστοιχα υποσυστήματα στο **διάγραμμα δομής**.

- Τα όρια μεταξύ των τριών υποσυστημάτων αναγνωρίζονται με το να καθορίζουμε το σημείο της πιο αφηρημένης εισαγωγής δεδομένων και της πιο αφηρημένης εξαγωγής δεδομένων.

Χαρακτηριστικά των 2 κατηγοριών(2)

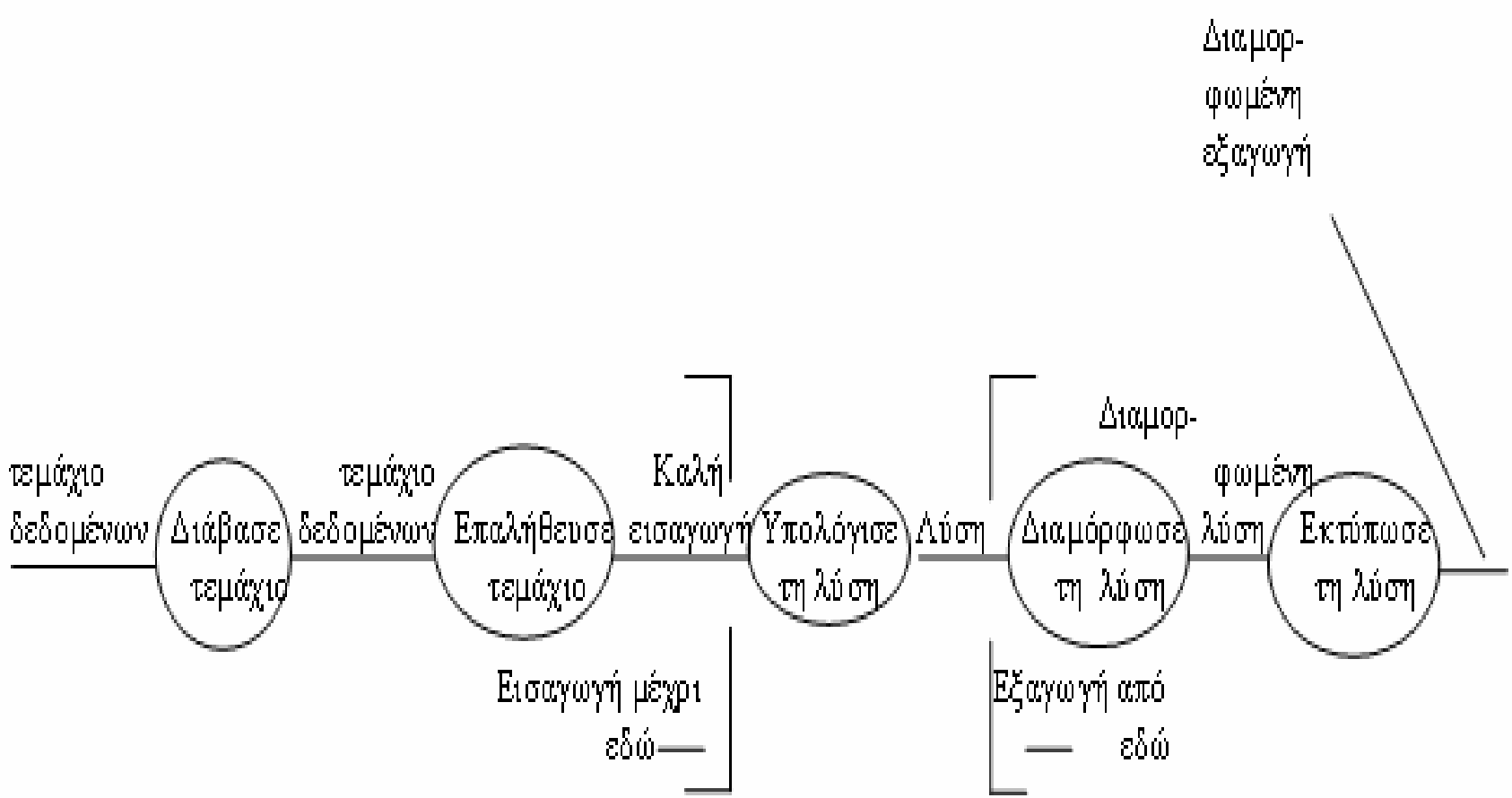
- **Σημείο της πιο αφηρημένης εισαγωγής δεδομένων**:

  Το σημείο εκείνο στο ΔΡΔ όπου η ροή εισαγωγής δεν μπορεί πλέον να αναγνωριστεί.

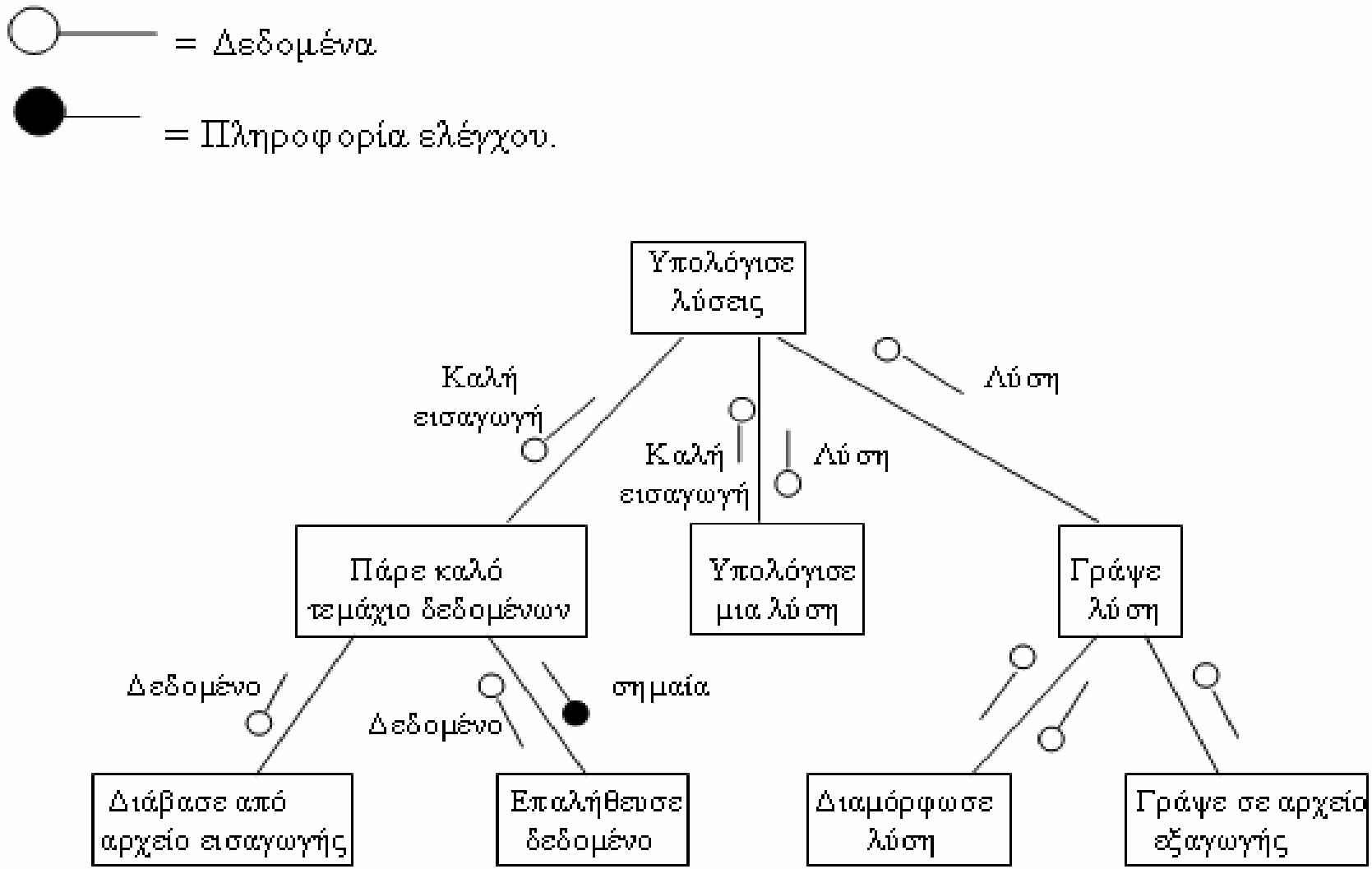
- **Σημείο της πιο αφηρημένης εξαγωγής δεδομένων**:

  Το σημείο εκείνο στο ΔΡΔ όπου αναγνωρίζονται για πρώτη φορά κάποια συστατικά εξαγωγής δεδομένων.

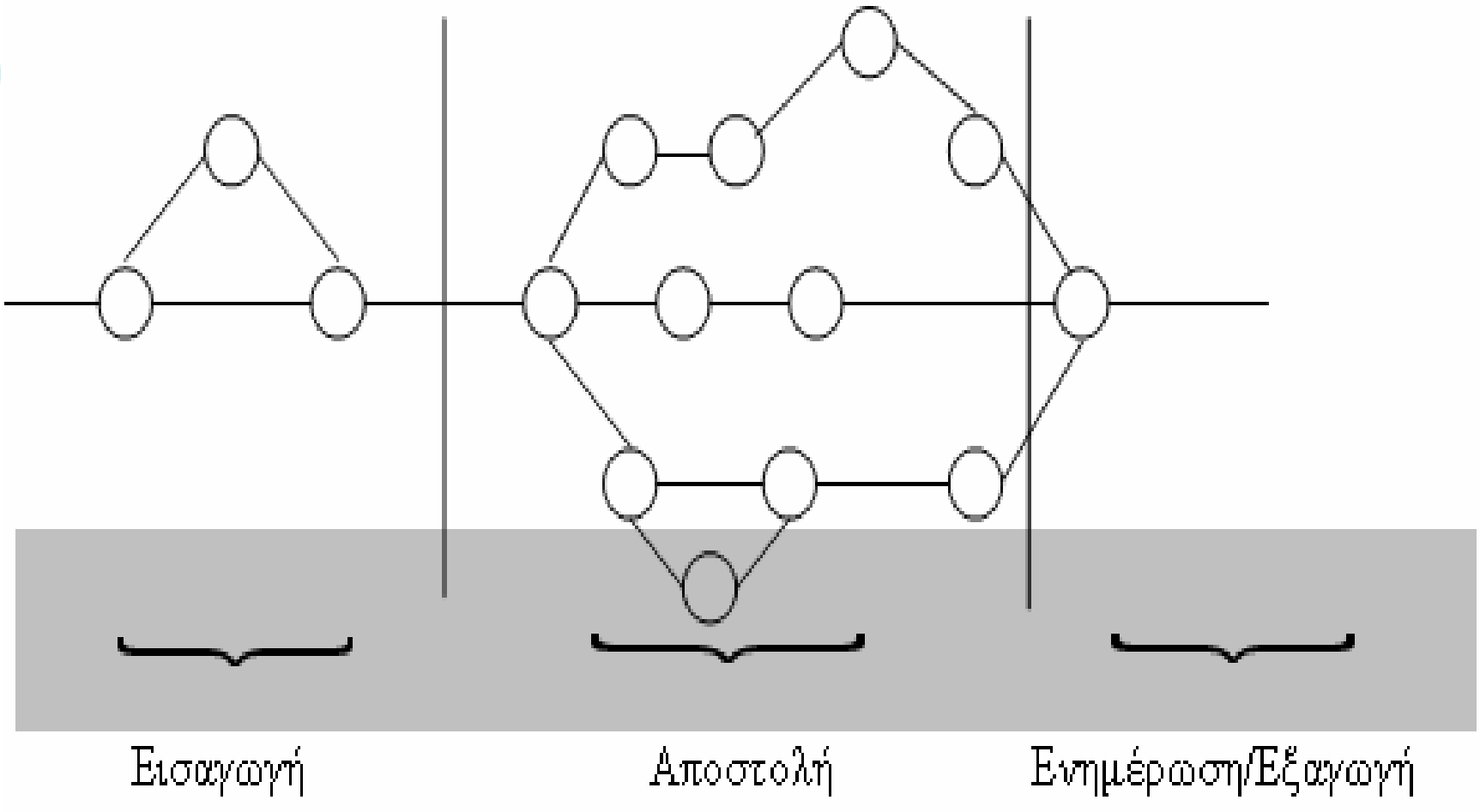
Παράδειγμα ενός ΔΡΔ επικεντρωμένου σε μεταβολή



Μετατροπή του ΔΡΔ σε διάγραμμα δομής



Παράδειγμα ΔΡΔ που είναι οδηγούμενο από διεξαγωγή



Μετατροπή του ΔΡΔ σε διάγραμμα δομής

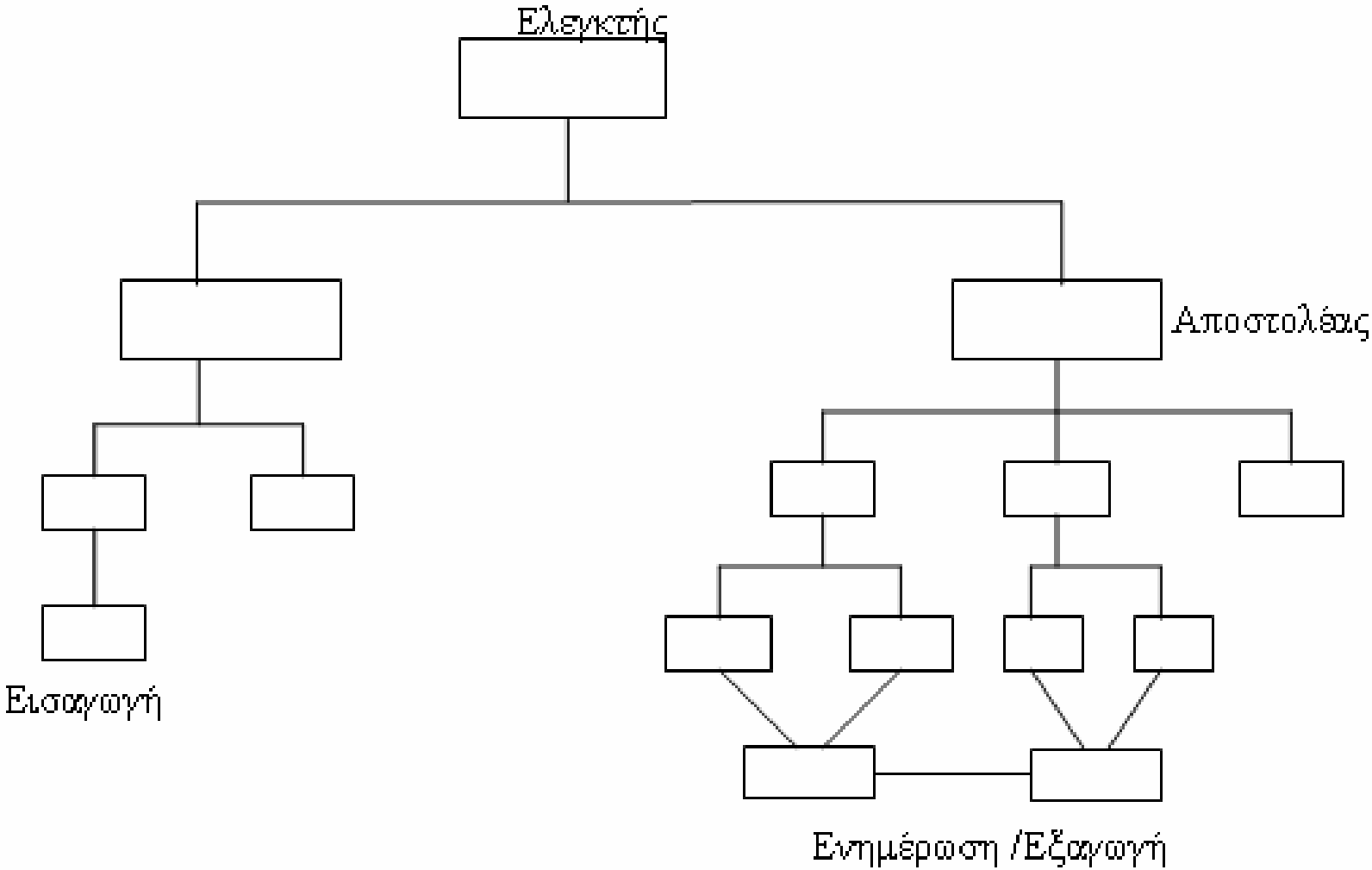
- Τέτοιου είδους ΔΡΔ μετατρέπεται σε διάγραμμα δομής έχοντας τα υποσυστήματα:
  1) Εισαγωγή (Input)
  2) Ελεγκτής (Controller)
  3) Αποστολέας (Dispatcher)
  4) Ενημέρωση/Εξαγωγή (Update/Output)

- Σε ένα ΔΡΔ οδηγούμενο-από-διεξαγωγή υπάρχουν **πολλά πιθανά μονοπάτια** (ένα για κάθε διαφορετική διεξαγωγή).

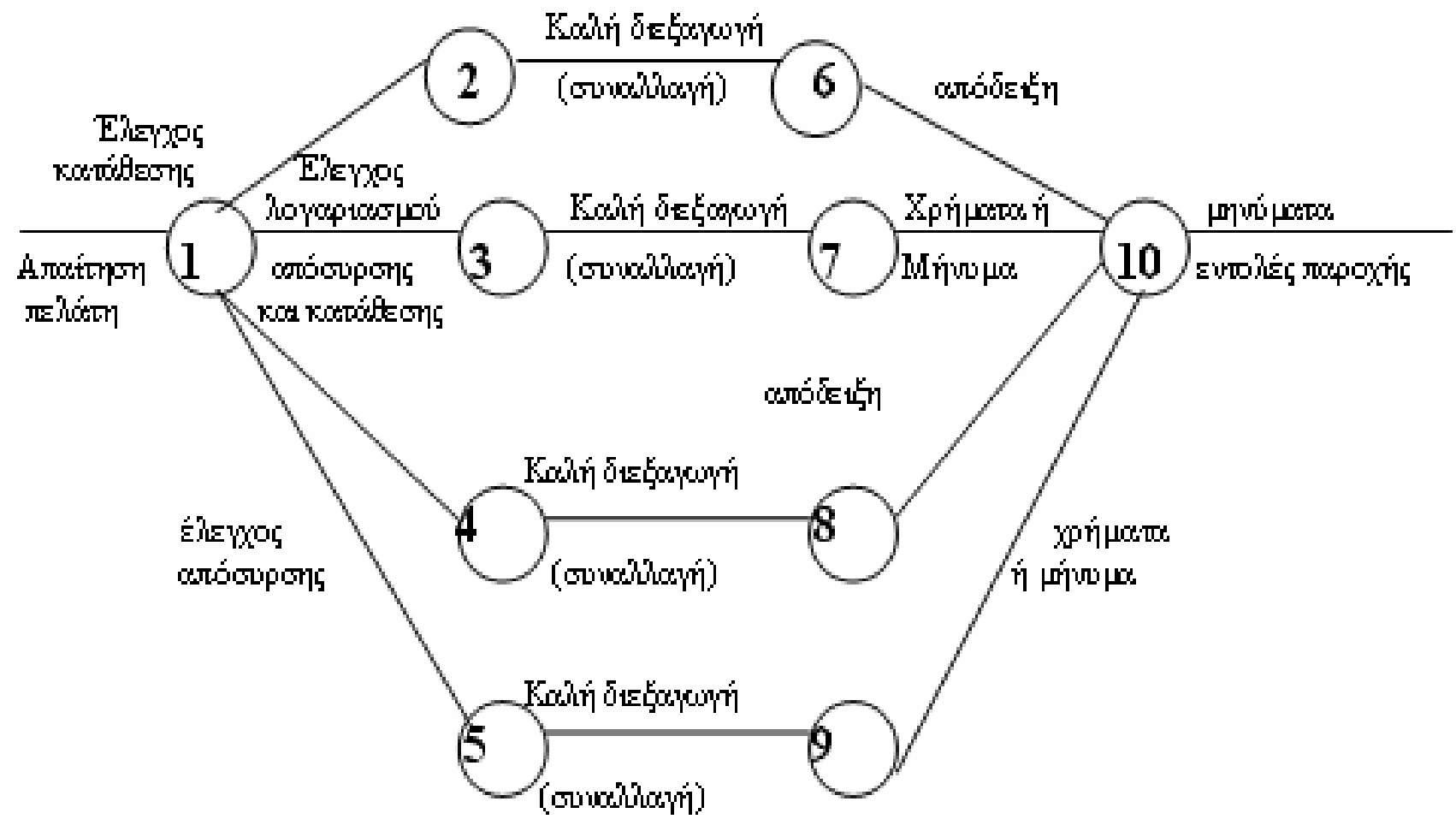
- Το μονοπάτι που επιλέγεται καθορίζεται από τις εντολές του χρήστη.

- Συχνά τα υποσυστήματα που αποτελούν το οδηγούμενο-από-διεξαγωγή ΔΡΔ είναι ΔΡΔ επικεντρωμένα-σε-μεταβολή τα οποία έχουν δομή εισαγωγή-επεξεργασία-εξαγωγή.

Τελικό Διάγραμμα Δομής



Παράδειγμα ΔΡΔ σε αυτόματο τραπεζικό ταμείο



Παράδειγμα ΔΡΔ σε αυτόματο τραπεζικό ταμείο

- Όπου:

  1: Καθορισμός τύπου διεξαγωγής.

  2, 3, 4, 5: Αποκωδικοποίηση και επαλήθευση απαίτησης.

  6: Απόδειξη παραλαβής κατάθεσης και ενημέρωση λογαριασμού.

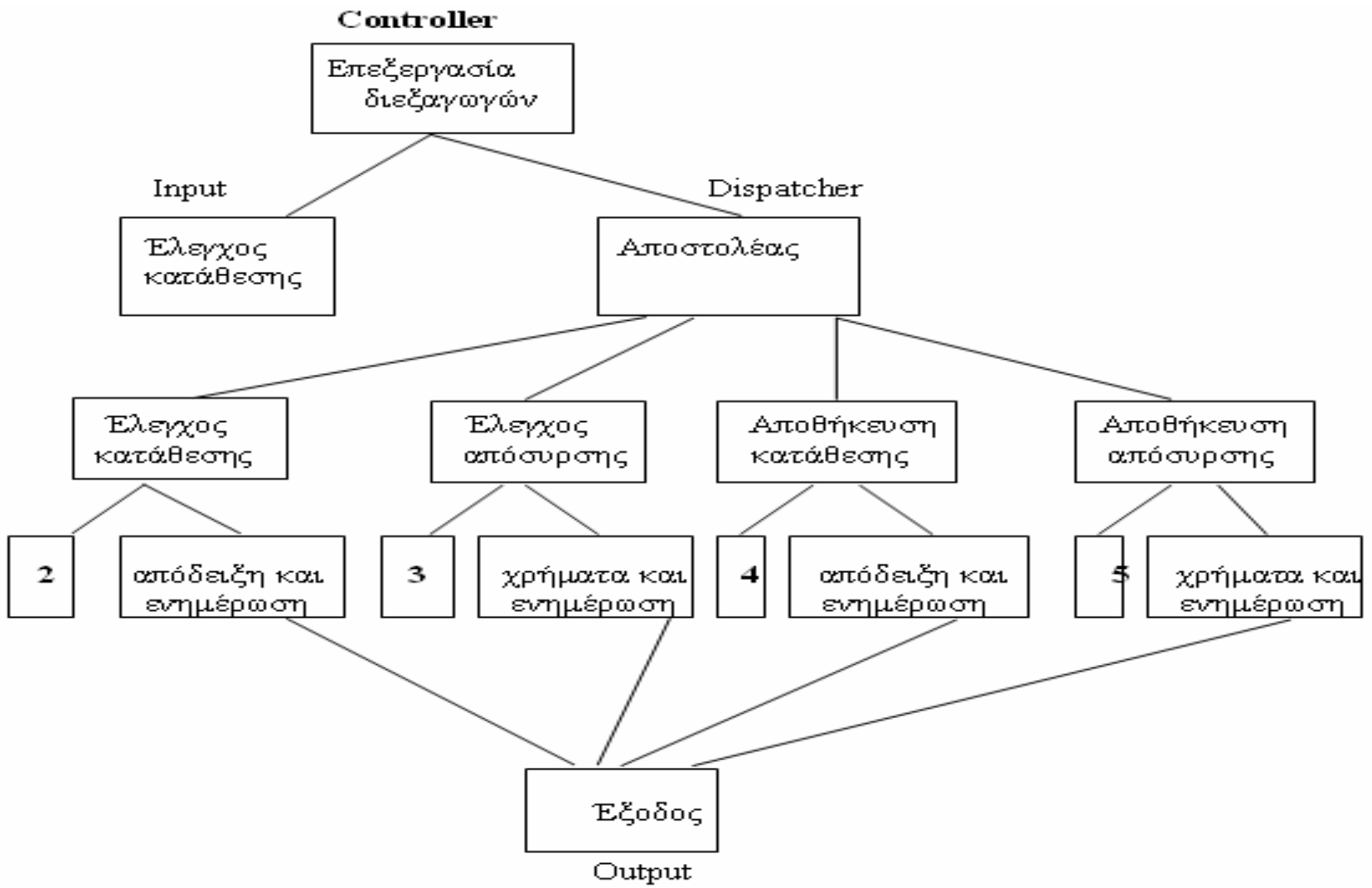
  7: Παροχή χρημάτων ή μηνύματος και ενημέρωση λογαριασμού.

  8: Απόδειξη κατάθεσης και ενημέρωση λογαριασμού.

  9: Παροχή χρημάτων ή μηνύματος, ενημέρωση λογαριασμού.

  10: Εκτύπωση εξαγωγής δεδομένων, παροχή χρημάτων.

Παράδειγμα Διαγράμματος Δομής σε αυτόματο ταμείο



Δομημένος προγραμματισμός Jackson (Jackson Structured Programming JSP)

- **Βήματα:**

1) Το πρόβλημα μοντελοποιείται με τον προσδιορισμό των δομών δεδομένων εισόδου και εξόδου σε δενδροειδές διάγραμμα.

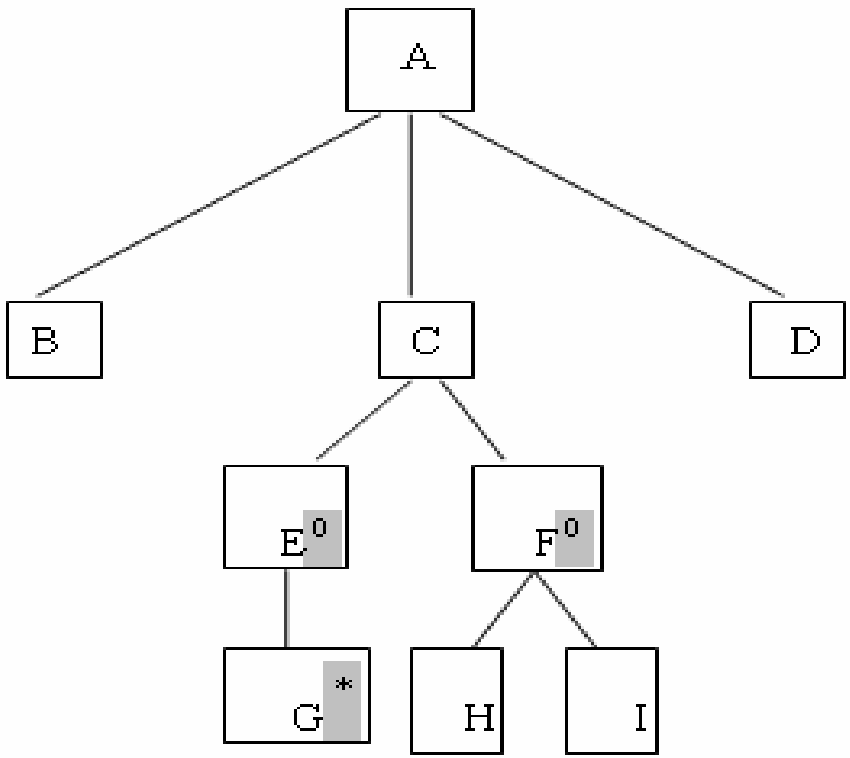
2) Το μοντέλο εισόδου-εξόδου μετατρέπεται σε δομικό μοντέλο για το πρόγραμμα με την αναγνώριση των σημείων επικοινωνίας μεταξύ κόμβων εισόδου και δένδρων εξόδου.

3) Το δομικό μοντέλο του προγράμματος αναπτύσσεται σε λεπτομερειακό μοντέλο σχεδιασμού που περιέχει τις λειτουργίες που απαιτούνται για τη λύση του προβλήματος.

Βήμα 1ο

- Οι δομές εισόδου/εξόδου προσδιορίζονται με ένα γραφικό συμβολισμό που προσδιορίζει ιεραρχία δεδομένων, επανάληψη δεδομένων και εναλλακτικά δεδομένα.

Παράδειγμα προσδιορισμού δεδομένου A (ή αντικειμένου A)



Αντιστοιχία σε BNF:

A ::= BCD

C ::= E/F

E ::= å/GE

F ::= HI

Βήμα 2ο και 3ο

- Το βήμα 2ο, επιτυγχάνεται με την αναγνώριση σημείων που είναι κοινά στις δομές εισόδου και εξόδου και με τον συνδυασμό των δύο δομών σε μια δομή προγράμματος.

- Ενώ το βήμα 3ο αποτελείται από τρία ακόμη βήματα:

  α) Αναπτύσσεται μια **λίστα από λειτουργίες** που απαιτούνται για να εκτελέσουν την επεξεργασία.

  β) Οι λειτουργίες σχετίζονται με τη δομή του προγράμματος.

  γ) Η δομή του προγράμματος και οι λειτουργίες εκφράζονται σε ένα συμβολισμό που ονομάζεται **σχηματική λογική**. Στην αναπαράσταση της σχηματικής λογικής προσδιορίζεται και η **ροή ελέγχου**.

Παράδειγμα μεθόδου Jackson

- Ένα αρχείο εισαγωγής (input file) αποτελείται από μια συλλογή **εγγράφων απογραφής** ταξινομημένης κατά τον **αριθμό εξαρτήματος**.
- Κάθε εγγραφή περιέχει έναν **αριθμό εξαρτήματος** και τον **αριθμό των τεμαχίων** αυτού του εξαρτήματος που δίνονται ή λαμβάνονται σε μια συναλλαγή.
- Στην έξοδο (output) πρέπει να παραχθεί μία αναφορά η οποία περιέχει επικεφαλίδα και μια γραμμή για την "καθαρή κίνηση" για κάθε **αριθμό εξαρτήματος** του αρχείου εισαγωγής.

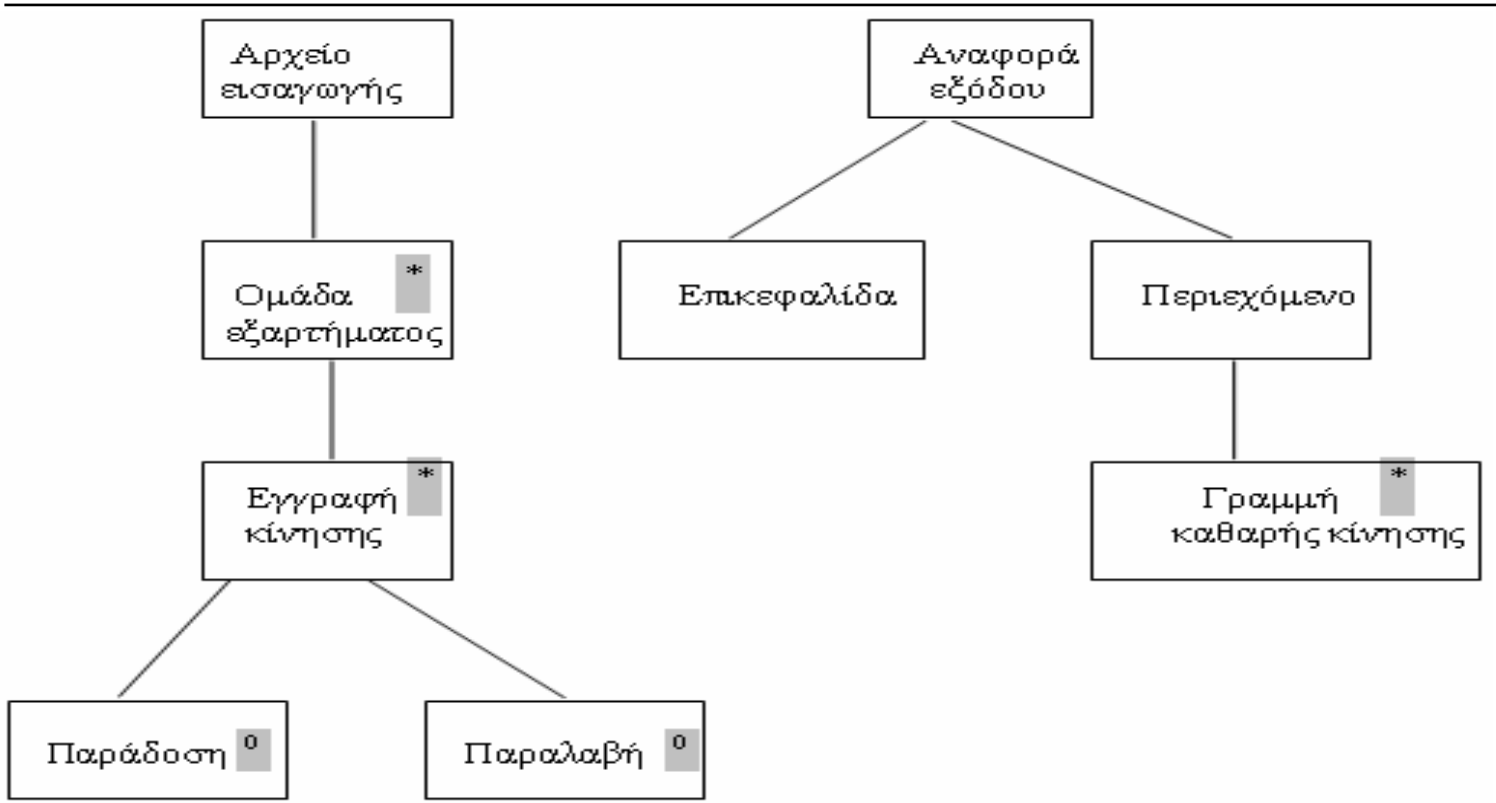
Παράδειγμα μεθόδου Jackson

- Επειδή το αρχείο εισαγωγής είναι ταξινομημένο κατά αριθμό εξαρτήματος, οι αποδείξεις **παράδοσης** και **παραλαβής** ενός **δοθέντος εξαρτήματος** βρίσκονται στο αρχείο εισαγωγής.

- Το μέρος όπου βρίσκονται αυτές οι αποδείξεις λέγεται **ομάδα εξαρτήματος**.

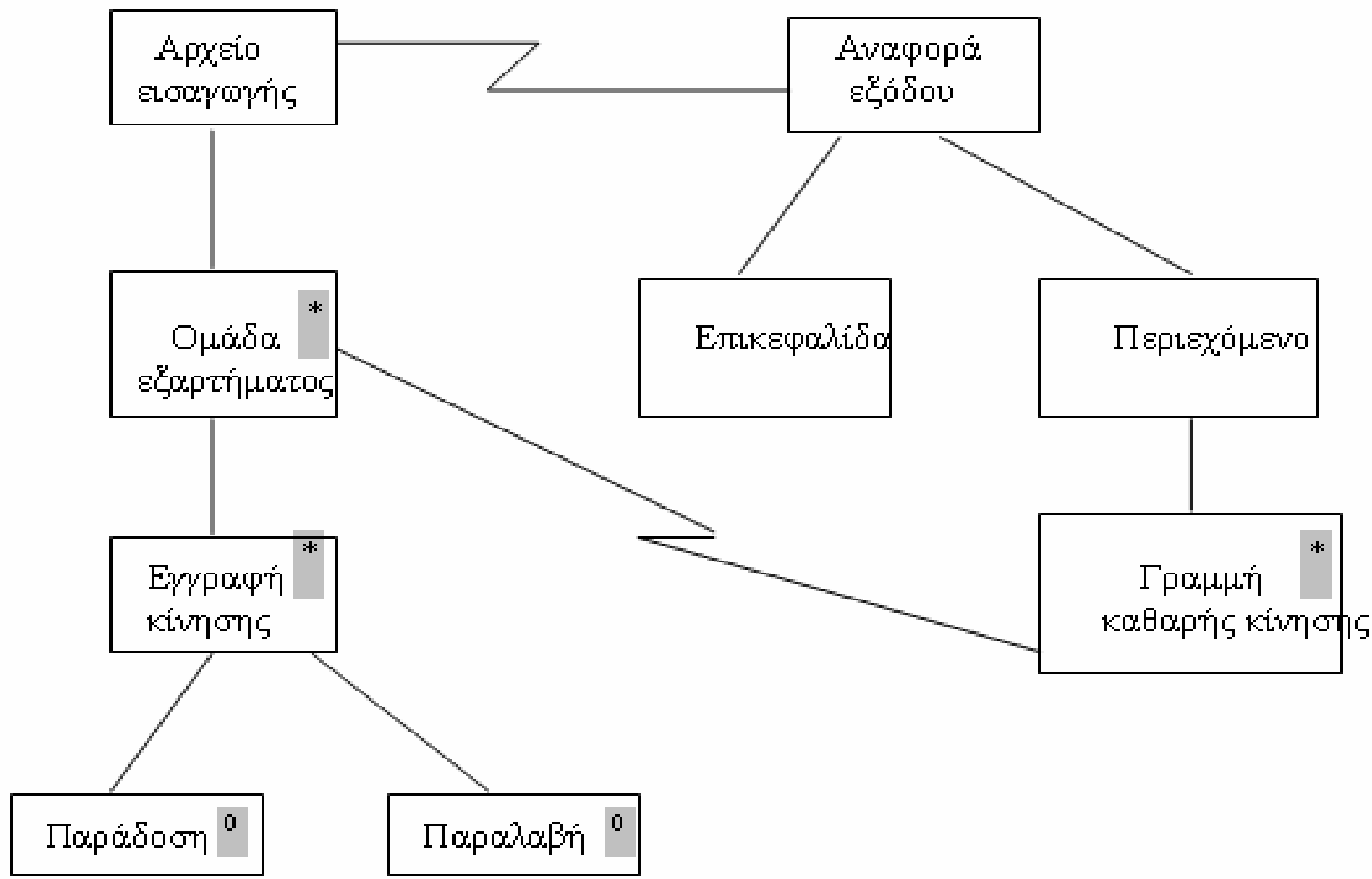
- Κάθε εγγραφή στην ομάδα εξαρτήματος ονομάζεται **εγγραφή κίνησης**.

Οι δομές εισόδου και εξόδου του παραδείγματος:

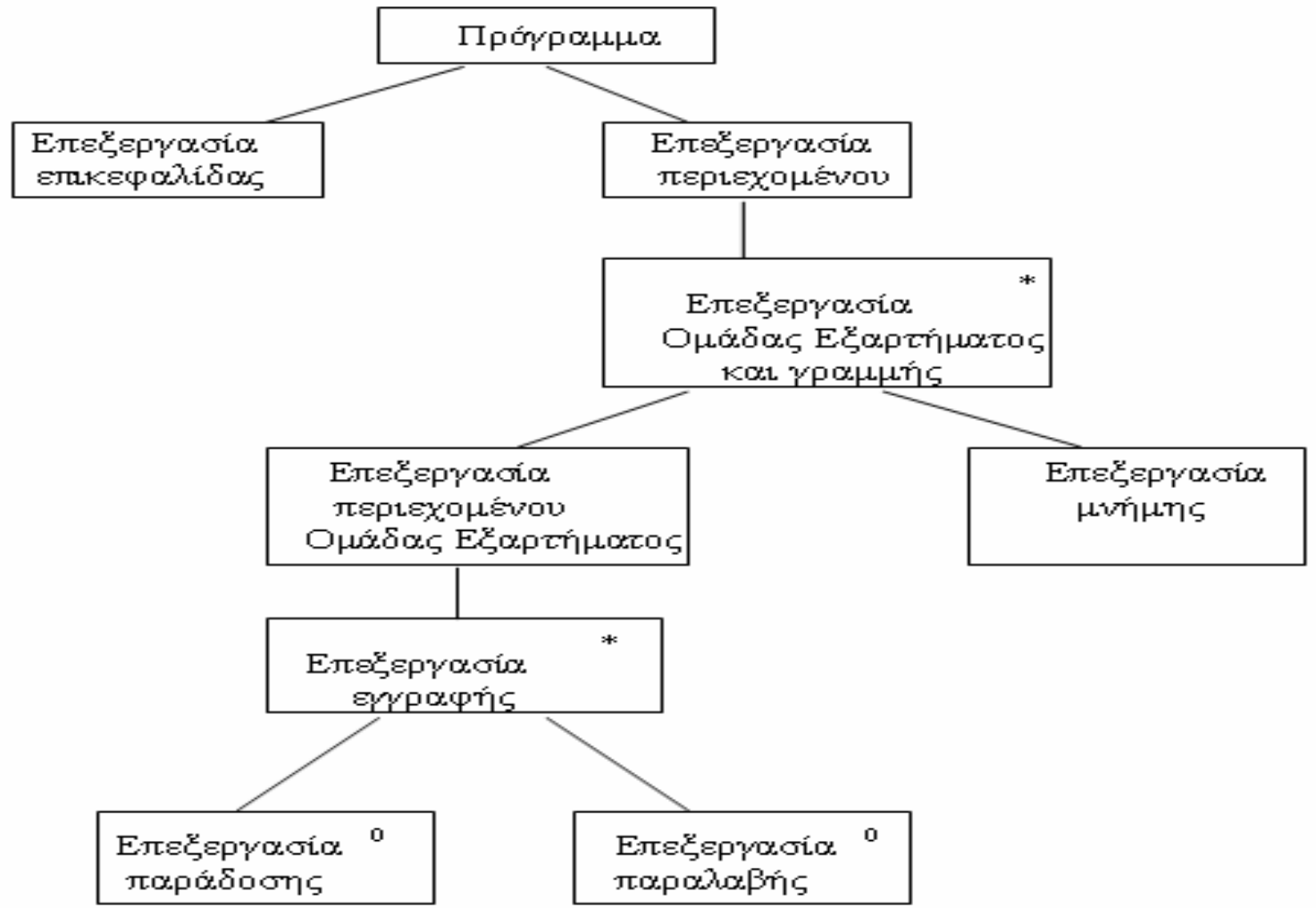


- Το αρχείο εισαγωγής αντιστοιχεί στην αναφορά εξόδου και **κάθε ομάδα εξαρτήματος** αντιστοιχεί σε μια **γραμμή καθαρής κίνησης**.
- Η δομή του προγράμματος βρίσκεται εάν συγκρίνουμε τη δομή του αρχείου εισαγωγής με τη δομή της αναφοράς εξόδου και μετά εάν ενώσουμε τους κόμβους που αντιστοιχούν μεταξύ τους σε κάθε γράφημα.

Αντιστοιχία μεταξύ δομών εισόδου-εξόδου για το παράδειγμα



Η δομή προγράμματος που προκύπτει :



Ερμηνεία της δομής προγράμματος

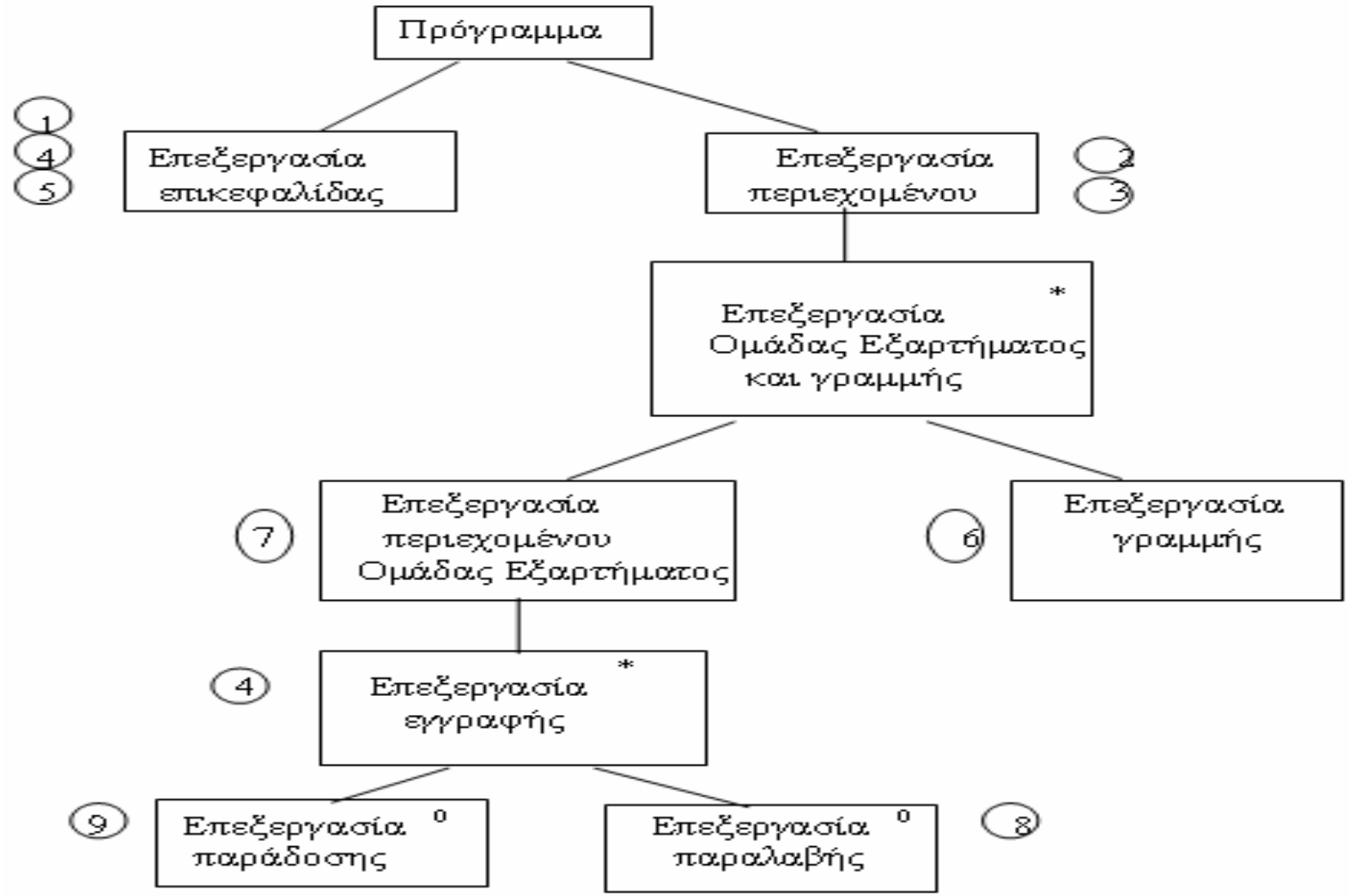
- Το πρόγραμμα αποτελείται από έναν αριθμό βημάτων επεξεργασίας.
- Υπάρχει ένα βήμα για την **εκτύπωση της επικεφαλίδας** της αναφοράς. Αυτό ακολουθείται από ένα βήμα για την εκτύπωση του **περιεχομένου** της **αναφοράς.**
- Το **περιεχόμενο της αναφοράς** αποτελείται από **επαναληπτική κλήση** της **ομάδας εξαρτήματος και γραμμής**. (Μια κλήση για κάθε **ομάδα εξαρτήματος** στο αρχείο εισαγωγής).
- Η **ομάδα εξαρτήματος και γραμμή** περιέχει ένα βήμα επεξεργασίας για μια **ομάδα εξαρτήματος** και ένα βήμα για την εκτύπωση της **γραμμής καθαρής κίνησης** για αυτήν την ομάδα εξαρτήματος.
- Το περιεχόμενο της ομάδας εξαρτήματος αποτελείται από ένα βήμα επεξεργασίας το οποίο ενεργοποιείται μία φορά για κάθε εγγραφή εξαρτήματος σε κάθε ομάδα εξαρτήματος. Κάθε ενεργοποίηση μιας επεξεργασίας εγγραφής επεξεργάζεται μια **παράδοση** ή μια **παραλαβή**.

Αναλυτικός Σχεδιασμός

- Πρέπει να αναπτύξουμε μια λίστα από λειτουργίες που χρειάζονται για το πρόγραμμα **συσχετίζοντας** τις **λειτουργίες** με τη **δομή προγράμματος** και μεταφράζοντας το αποτέλεσμα σε **σχηματική λογική** (ψευδοκώδικα).

- Α) **Λίστα από λειτουργίες.**
  1. Άνοιξε αρχεία.
  2. Κλείσε αρχεία.
  3. Σταμάτα την εκτέλεση.
  4. Επεξεργασία εγγραφής PART-NUM, MOVMNT.
  5. Γράψε επικεφαλίδα.
  6. Γράψε γραμμή NET-MOVMNT.
  7. Βάλε NET-MOVMNT στο ΜΗΔΕΝ.
  8. Πρόσθεσε MOVMNT στο NET-MOVMNT.
  9. Αφαίρεσε MOVMNT από NET-MOVMNT.

Συσχέτιση λειτουργιών με δομή προγράμματος



Σχηματική λογική του παραδείγματος

```
BEGIN PROGRAM
    OPEN FILES
    READ PART-NUM, MOVMNT
    WRITE HEADING
    ITERATE WHILE NOT (END-OF-FILE)
            SET NET-MOVMNT TO ZERO
             ITERATE WHILE SAME PART-NUMBER
                  IF (MOVMNT = PARADOSI) THEN
                     SUBTRACT MOVMNT FROM NET-MOVMNT
                  ELSE IF (MOVMNT = PARALAVI) THEN
                      ADD MOVMNT TO NET-MOVMNT
                  END IF
                  READ PART-NUM, MOVMNT
             END WHILE
             CLOSE FILES
             STOP
END PROGRAM
```