Συστήματα Πραγματικού Χρόνου

Σχεδιασμός Συστημάτων Πραγματικού Χρόνου

Κεφάλαιο 3

Σχεδιασμός συστημάτων

- Η διαδικασία ανάλυσης που παρουσιάστηκε στο κεφάλαιο 2 παράγει ένα βασικό μοντέλο το οποίο περιγράφει ακριβώς τι πρέπει να κάνει το σύστημα
  - Τίθενται και περιορισμοί στην υλοποίηση
- Ο σκοπός του σχεδιασμού είναι ο μετασχηματισμός του βασικού μοντέλου σε πραγματικό μοντέλο, τηρώντας τους ανωτέρω περιορισμούς υλοποίησης
  - Λαμβάνονται επίσης υπ' όψιν οι περιορισμοί της τεχνολογίας και λαμβάνονται αποφάσεις για την τεχνολογία που θα υιοθετηθεί
  - Οι περιορισμοί αυτοί δεν θα πρέπει να τροποποιούν τη βασική λειτουργικότητα ή να την αλλάζουν όσο το δυνατόν λιγότερο
- Θα εστιάσουμε στη δομημένη σχεδίαση, παρέχοντας και μία επισκόπηση της αντικειμενοστρεφούς

Περιορισμοί τεχνολογίας

Τρεις περιοχές τεχνολογίας λαμβάνονται υπόψη:

- *επεξεργαστές – περιορισμοί υλικού*. Τίθενται περιορισμοί από:
  (α) το πλήθος των επεξεργαστών που διατίθενται
  (β) το μέγεθος και τις δυνατότητες αυτών των επεξεργαστών
  (γ) τις πιθανές διεπαφές (interfaces) ανάμεσα στους επεξεργαστές
- *επεξεργαστές – περιορισμοί λογισμικού*. Τίθενται περιορισμοί από το είδος των ευκολιών που είναι διαθέσιμες για κάθε επεξεργαστή:
  (α) το λειτουργικό σύστημα
  (β) υπορουτίνες συστήματος και βιβλιοθηκών
  (γ) απαιτούμενα πακέτα λογισμικού
- *γλώσσες προγραμματισμού*.
  - δυνατότητες και περιορισμοί που υπάρχουν σε κάθε μία από τις διαθέσιμες γλώσσες.

Δομημένη σχεδίαση

Δομικά διαγράμματα

- Παρουσιάζουν την εσωτερική δομή ενός προγράμματος μέσω μίας ιεραρχίας αυτοτελών μονάδων κώδικα
  - Σε αυτά βασίζεται η ανάπτυξη του κώδικα
- Συμπληρώνεται το *λεξικό δεδομένων*
  - περιέχει τις προδιαγραφές (συνήθως σε μορφή κειμένου), όλων των διαγραμματικών στοιχείων του μοντέλου που περιγράφονται στο δομικό διάγραμμα μαζί με τις προδιαγραφές διαδικασιών και μονάδων κώδικα
  - Περιέχει επίσης τις προδιαγραφές χαμηλού επιπέδου, οι οποίες δεν μπορούν να παρουσιαστούν εύκολα στο μοντέλο, επιτρέπουν ωστόσο τον έλεγχο της ορθότητάς του

Διαδικασία οργάνωσης κώδικα σε Σ.Π.Χ.

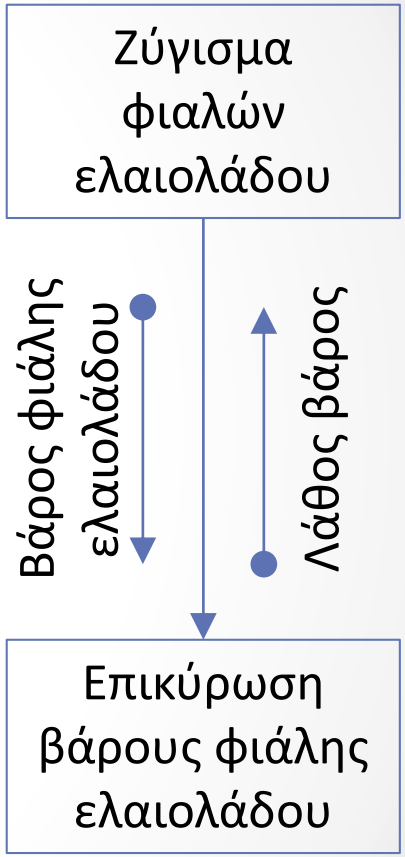
1. Θεωρούμε ως είσοδο τα αντικείμενα της ανάλυσης, δηλ. τα διαγράμματα DFD, STD, ERD και το λεξικό δεδομένων
   - Με έμφαση στα DFD και STD διότι εστιάζουμε σε λειτουργίες ενώ το ERD και το λεξικό δεδομένων περιγράφουν δεδομένα
2. Αναδεικνύουμε τα προγράμματα που χρειάζονται για την υλοποίηση της επιθυμητής λειτουργικότητας
3. Για κάθε πρόγραμμα δημιουργείται ένα δομικό διάγραμμα (structure chart) που παρουσιάζει την εσωτερική δομή του μέσω μίας ιεραρχίας αυτοτελών μονάδων κώδικα
   - Παράλληλα δημιουργείται το *λεξικό δεδομένων* (data / project dictionary), το οποίο περιέχει τις προδιαγραφές όλων των διαγραμματικών στοιχείων του μοντέλου που περιγράφονται στο δομικό διάγραμμα μαζί με τις προδιαγραφές διαδικασιών και μονάδων κώδικα
   - Περιέχει ακόμη προδιαγραφές χαμηλού επιπέδου, που δεν μπορούν να παρουσιαστούν εύκολα στο μοντέλο, επιτρέπουν όμως τον έλεγχο της ορθότητάς του
4. Τέλος, τα δομικά διαγράμματα βελτιώνονται

Εσωτερική δομή δομικών διαγραμμάτων

- Το δομικό διάγραμμα αντιστοιχεί σε ένα πρόγραμμα, συνεπώς διαθέτει ένα «κύριο πρόγραμμα» το οποίο ελέγχει αυτοτελείς μονάδες κώδικα χαμηλότερου επιπέδου
  - καλεί διαδικασίες-συναρτήσεις
  - αναμένει την ολοκλήρωσή τους και την επιστροφή αποτελεσμάτων
- Σε αντιδιαστολή, τα DFD, STD δεν έχουν αντίστοιχη δομή
  - Τα στοιχεία τους συνδέονται χωρίς ιεραρχία, σε δομή δικτύου
- Η ανάδειξη/δημιουργία της ιεραρχικής δομής είναι ουσιαστικό στοιχείο στην κατασκευή των δομικών διαγραμμάτων

Στοιχεία και σημειολογία δομικών διαγραμμάτων

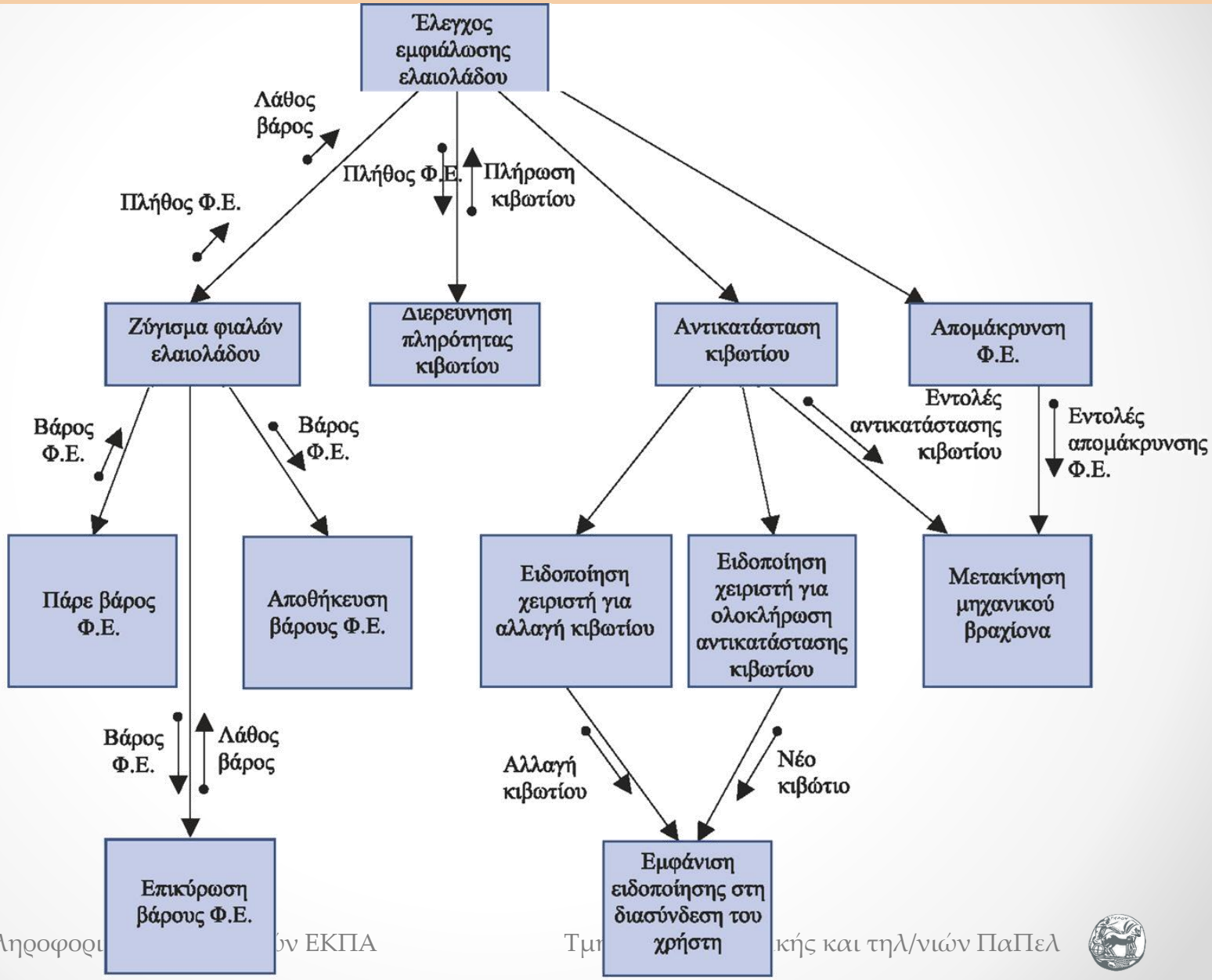
- Τα δομικά διαγράμματα παρουσιάζουν:
  - Τις αυτόνομες μονάδες κώδικα
    - συμβολίζονται με ορθογώνια
  - Την κλήση μιας μονάδας κώδικα χαμηλότερου επιπέδου από μία μονάδα κώδικα υψηλότερου επιπέδου
    - συμβολίζεται με μία κατευθυνόμενη γραμμή από την καλούσα προς την καλούμενη μονάδα
  - Τη μεταβίβαση παραμέτρων από την καλούσα προς την καλούμενη μονάδα και την επιστροφή αποτελεσμάτων στην αντίθετη κατεύθυνση
    - Συμβολίζεται με ένα βέλος, παράλληλο προς την ακμή που συμβολίζει την κλήση, που επιγράφεται με την περιγραφή των δεδομένων



Διάταξη και ανάγνωση δομικού διαγράμματος

- Η μονάδα-ρίζα αναπαριστά το κύριο πρόγραμμα
- Οι κατωτέρου επιπέδου αυτόνομες μονάδες κώδικα τοποθετούνται πιο χαμηλά στο δομικό διάγραμμα σε σχέση με τις ανωτέρου επιπέδου
- Κάθε αυτόνομη μονάδα κώδικα μπορεί να καλείται από περισσότερες από μία μονάδες κώδικα ανώτερου επιπέδου
- Το διάγραμμα διαβάζεται από πάνω προς τα κάτω και από αριστερά προς τα δεξιά
- Κάθε αυτόνομη μονάδα κώδικα έχει τις αντίστοιχες προδιαγραφές (στο *λεξικό δεδομένων* - data / project dictionary), οι οποίες αναλύουν επακριβώς τι συμβαίνει στο εσωτερικό της

Παράδειγμα δομικού διαγράμματος



Δημιουργία προκαταρκτικού δομικού διαγράμματος από την εκτελέσιμη μονάδα

- Θα μελετήσουμε δύο τρόπους:
  1. με χρήση των *πινάκων μετάβασης καταστάσεων*.
     - Για την προσέγγιση αυτή θα αναλύσουμε πώς φτιάχνονται οι πίνακες, τι ισχύει γι' αυτούς και θα δώσουμε ένα παράδειγμα δημιουργίας κώδικα
  2. με άμεση κωδικοποίηση
     - Η προσέγγιση αυτή είναι πιο άμεση στην εφαρμογή της, ωστόσο δημιουργεί κάποια προβλήματα όταν η λογική του διαγράμματος μετάβασης καταστάσεων (STD) δεν είναι απλοϊκή
- Θα μελετήσουμε επίσης τη διαδικασία ανάδειξης ενός *ελεγκτικού μετασχηματισμού* από το DFD, ο οποίος θα αποτελέσει και το κύριο πρόγραμμα

Δημιουργία προκαταρκτικού δομικού διαγράμματος από STD

- Δύο μέθοδοι:
  - Η πρώτη μέθοδος περιλαμβάνει πίνακες μετάβασης καταστάσεων (state transition tables - STTs)
    - το διάγραμμα STD δίνει μία μορφή «αναγνώσιμου προγράμματος» που ενσωματώνεται στο κύριο πρόγραμμα
    - Επιτυγχάνεται εύκολη συντήρηση και ιχνηλασιμότητα
  - Παρόμοια με την πρώτη μέθοδο, η λογική του STD κωδικοποιείται εντός του κυρίου προγράμματος
    - εδώ ωστόσο ο κώδικας του διαγράμματος STD παράγεται απ' ευθείας μέσω της εξέτασης του STD.
    - Μεγαλύτερη απλότητα, αμεσότητα, και αποδοτικότητα, δύσκολη συντήρηση και ιχνηλασιμότητα

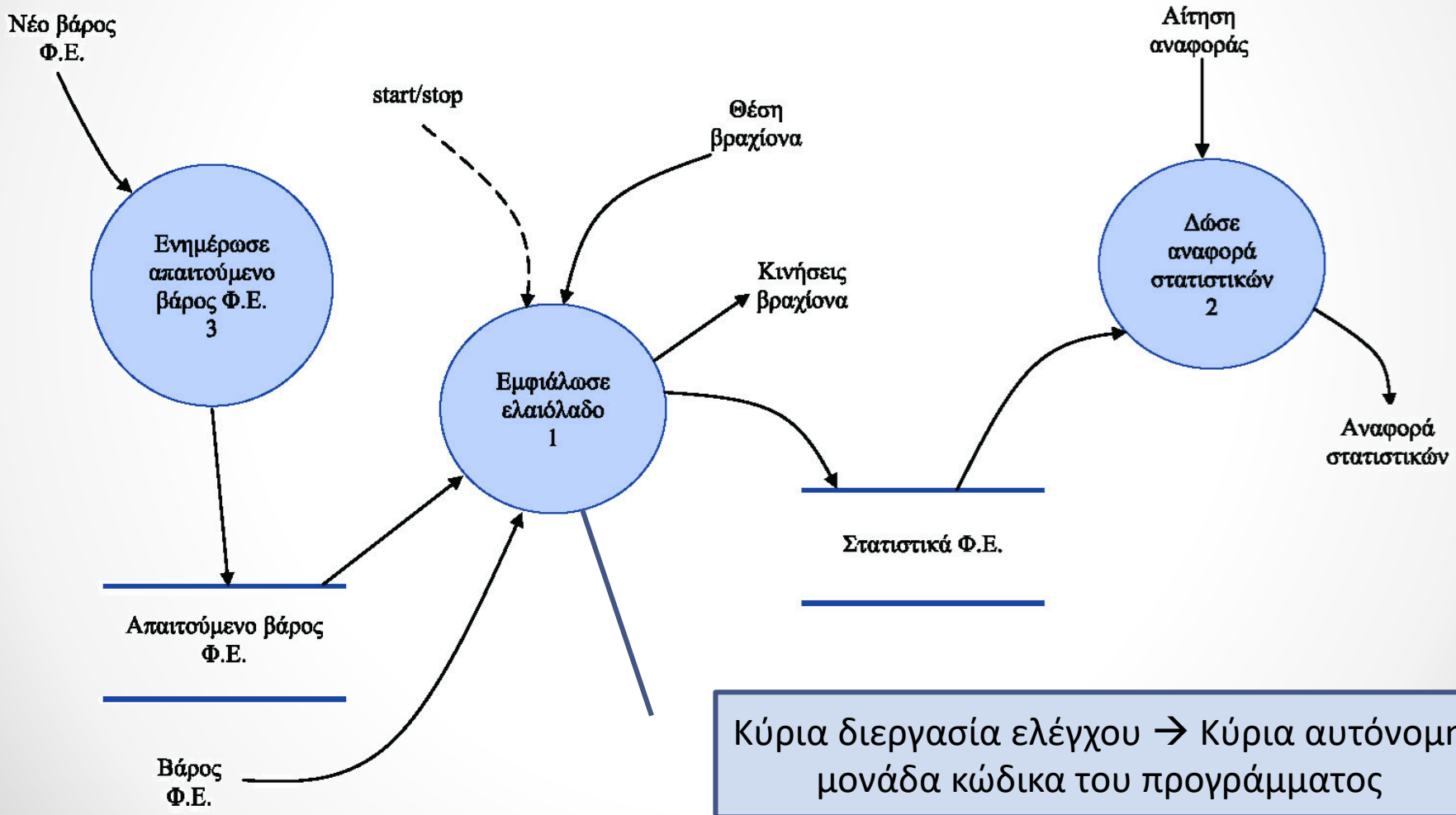
Μετατροπή διαγράμματος STD

Στόχος: μετατροπή διαγραμμάτων DFD & STD σε δομικό διάγραμμα

1. Η κύρια διεργασία ελέγχου του STD γίνεται η κύρια αυτόνομη μονάδα κώδικα του προγράμματος και ονομάζεται με τρόπο που να περιγράφει τη λειτουργία του συνολικού προγράμματος.
   - Η κορυφαία μονάδα κώδικα θα πρέπει να περιέχει μόνο τη λογική του προγράμματος
   - Οι επιμέρους εργασίες θα γίνονται στα χαμηλότερα επίπεδα
2. Οι ενέργειες χαμηλότερου επιπέδου δεν αντιστοιχούν όλες σε μετασχηματισμούς δεδομένων του διαγράμματος DFD
   - Μερικές ενέργειες προκύπτουν από το STD
3. Όλες οι ενέργειες του STD θα απεικονιστούν σε μονάδες στον κώδικα του προγράμματος
4. Οι προδιαγραφές των μονάδων μπορούν να προκύψουν από τα χαρακτηριστικά των μετασχηματισμών δεδομένων

Μετατροπή διαγράμματος STD: παράδειγμα

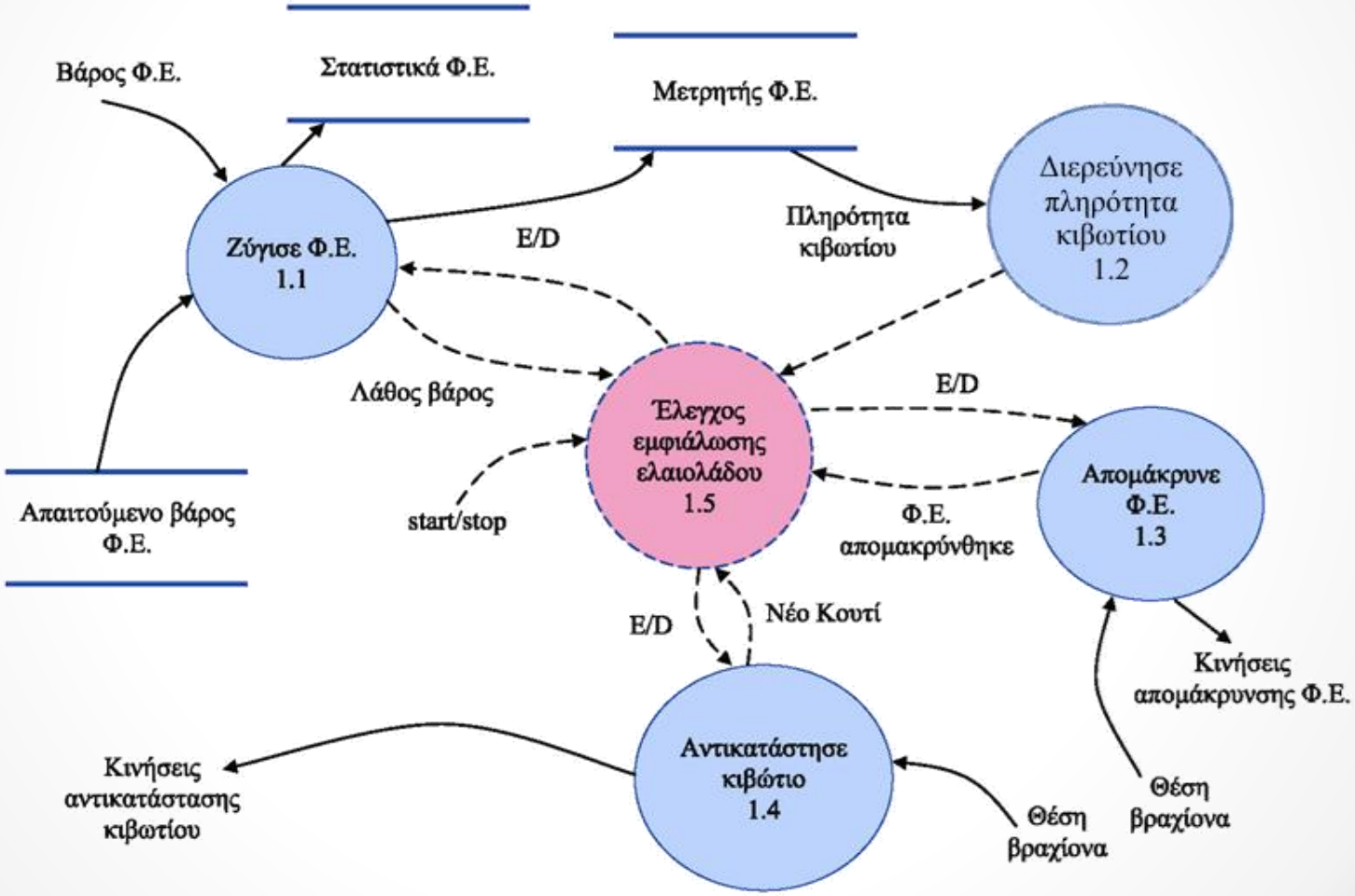
1. DFD πρώτου επιπέδου για σύστημα εμφιάλωσης ελαιολάδου



Κύρια διεργασία ελέγχου → Κύρια αυτόνομη μονάδα κώδικα του προγράμματος

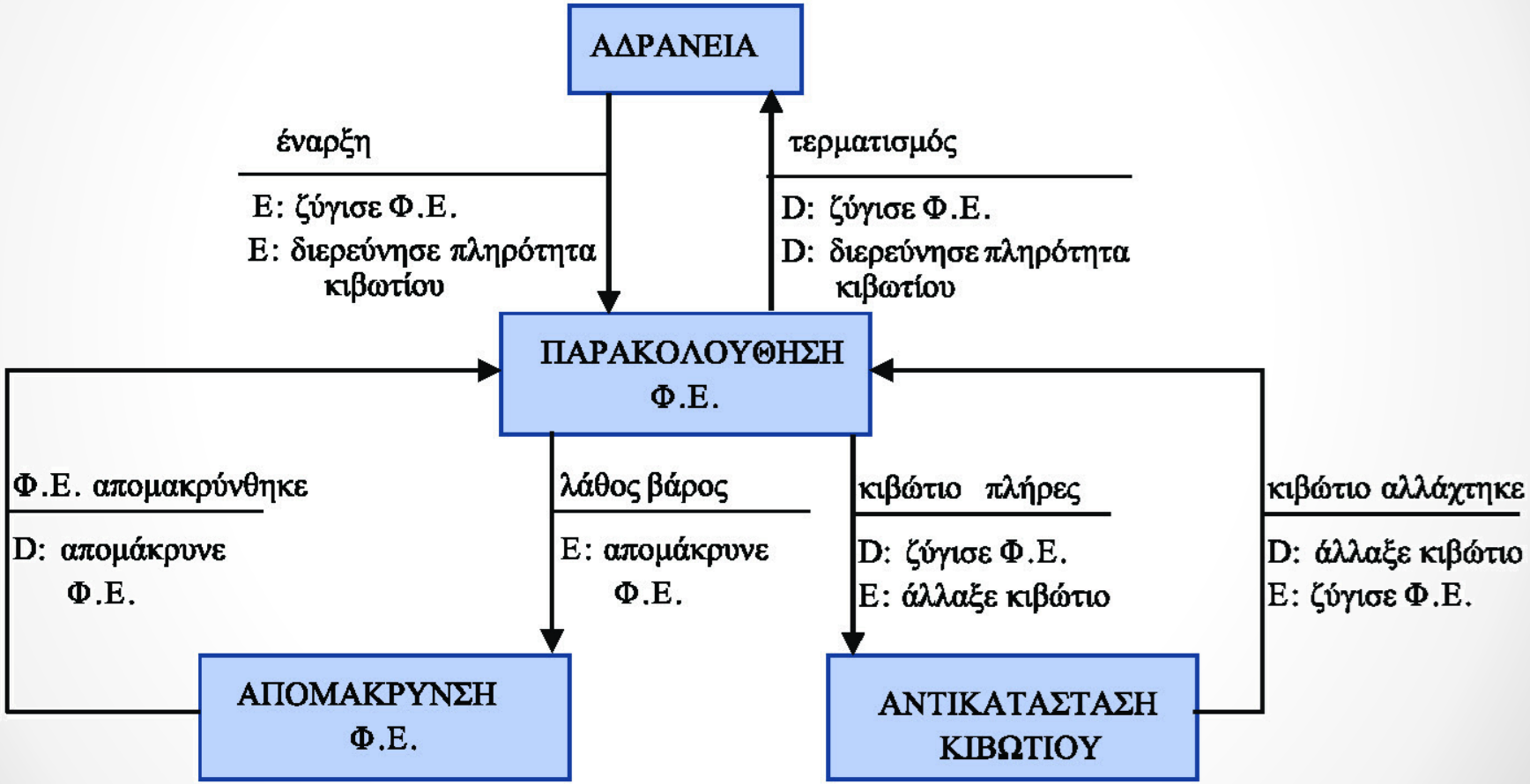
Μετατροπή διαγράμματος STD: παράδειγμα

2. DFD δευτέρου επιπέδου για σύστημα εμφιάλωσης ελαιολάδου



Μετατροπή διαγράμματος STD: παράδειγμα

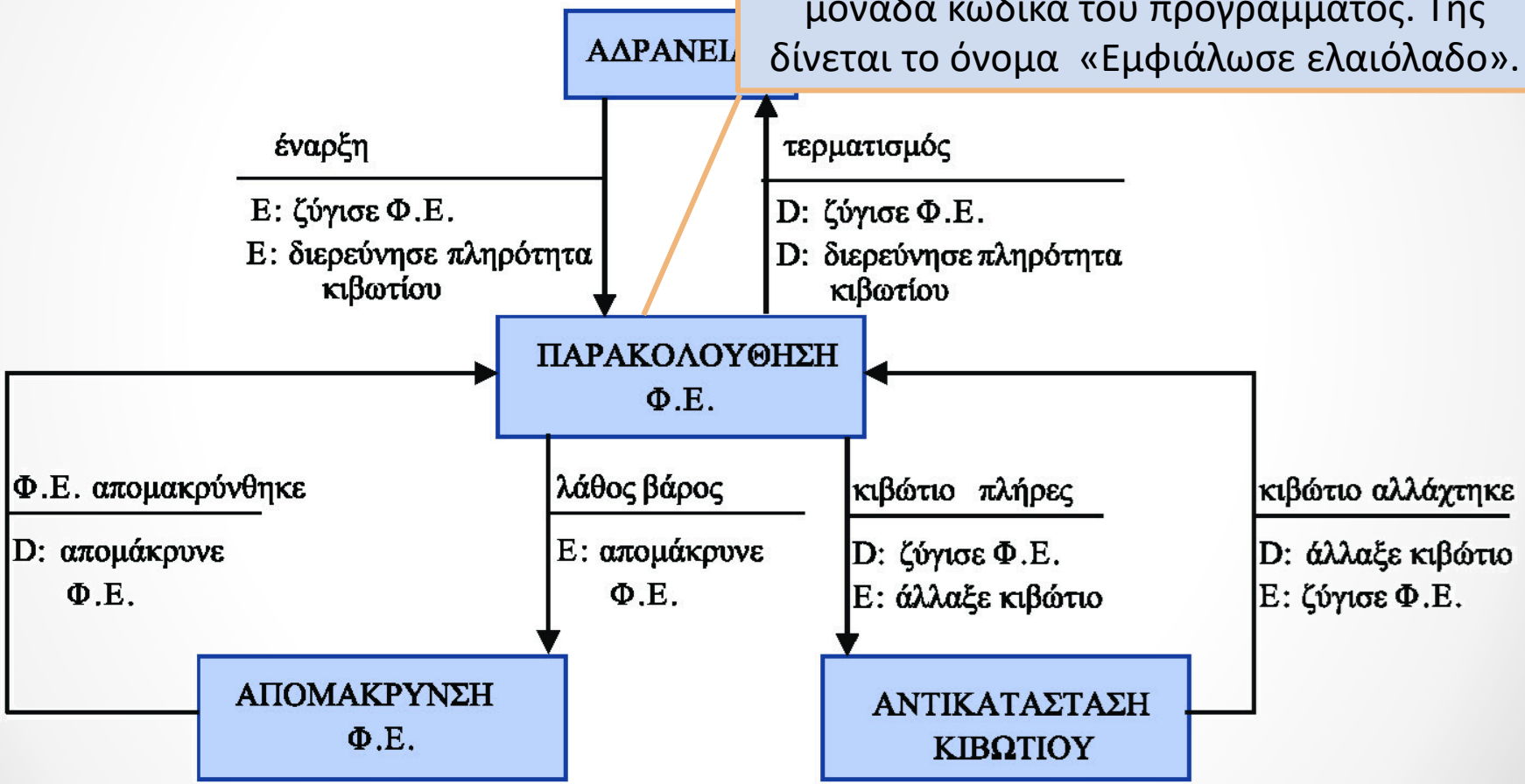
3. STD συστήματος εμφιάλωσης ελαιολάδου



Μετατροπή διαγράμματος STD: παράδειγμα

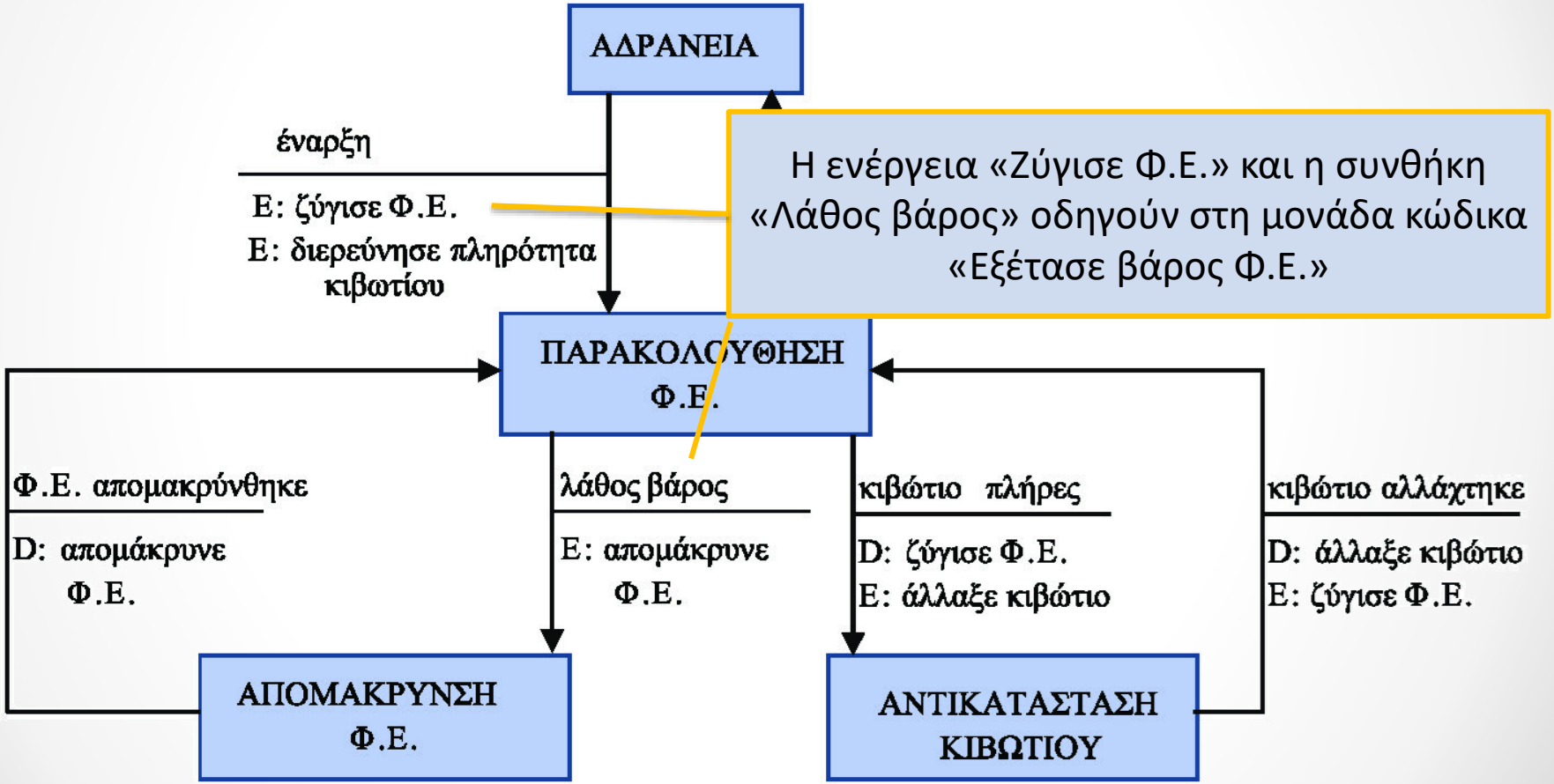
3. STD συστήματος εμφιάλωσης ελαιολάδου

Κύρια διεργασία ελέγχου → Κύρια αυτόνομη μονάδα κώδικα του προγράμματος. Της δίνεται το όνομα «Εμφιάλωσε ελαιόλαδο».



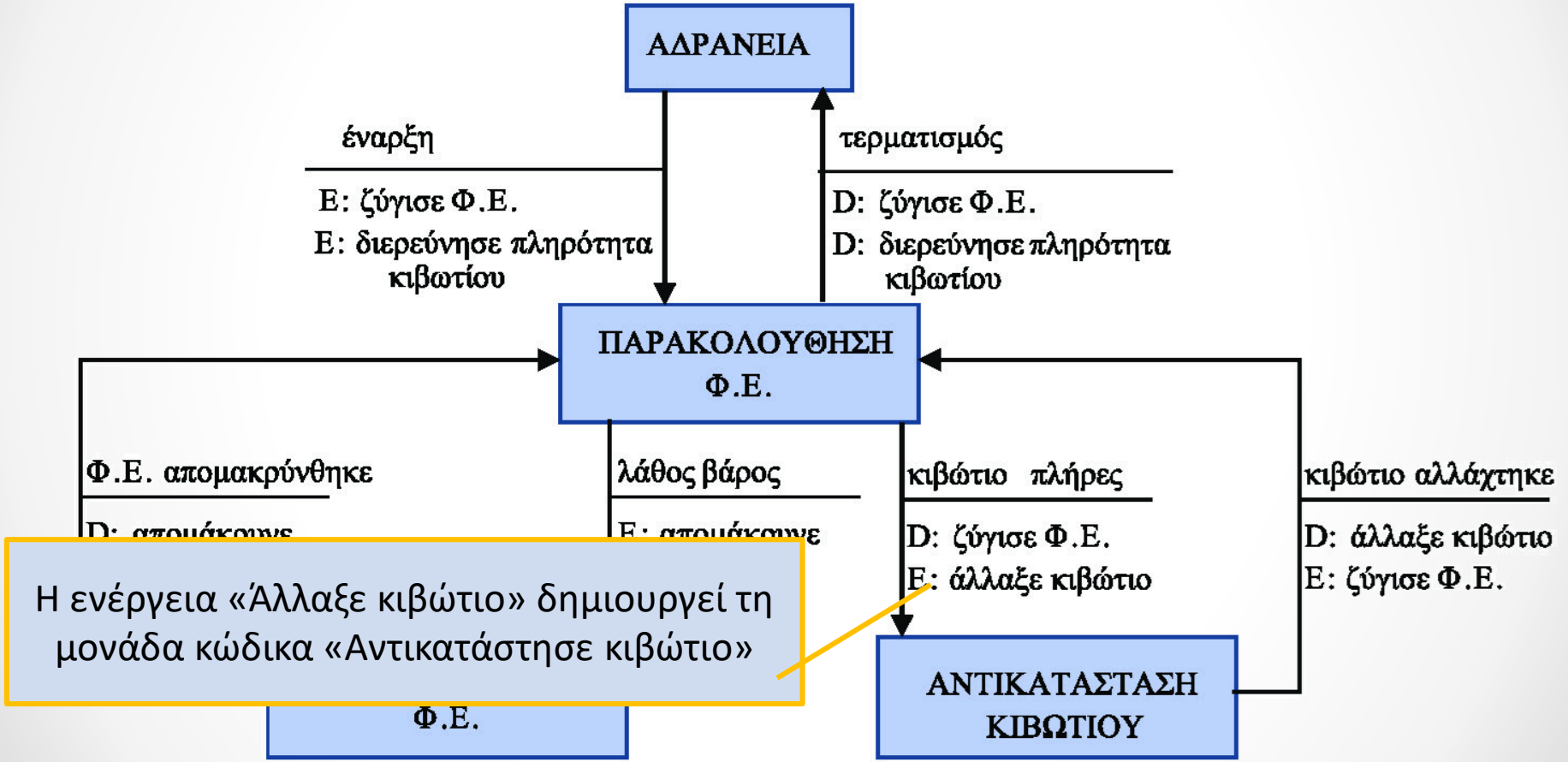
Μετατροπή διαγράμματος STD: παράδειγμα

3. STD συστήματος εμφιάλωσης ελαιολάδου



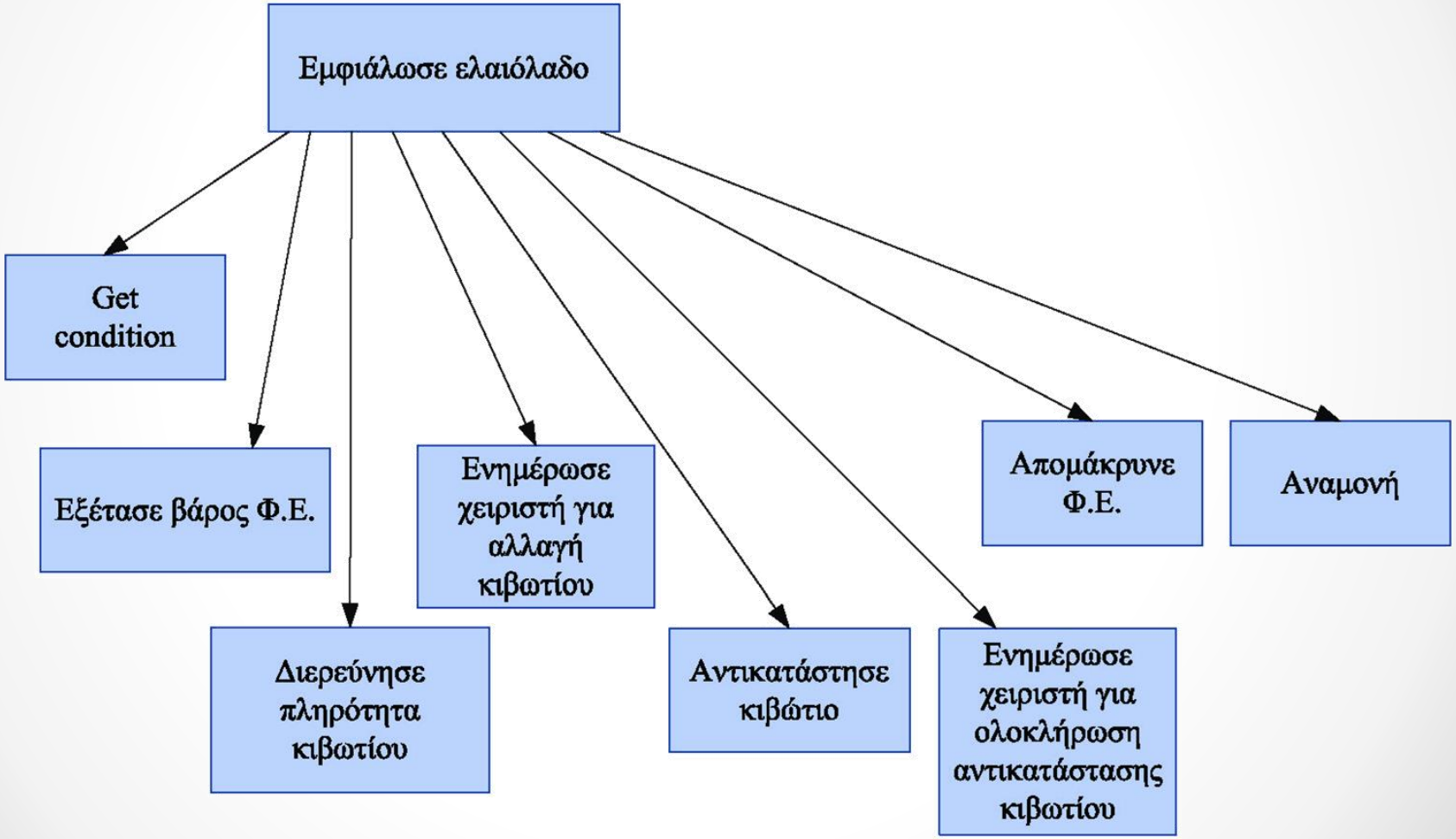
Μετατροπή διαγράμματος STD: παράδειγμα

3. STD συστήματος εμφιάλωσης ελαιολάδου



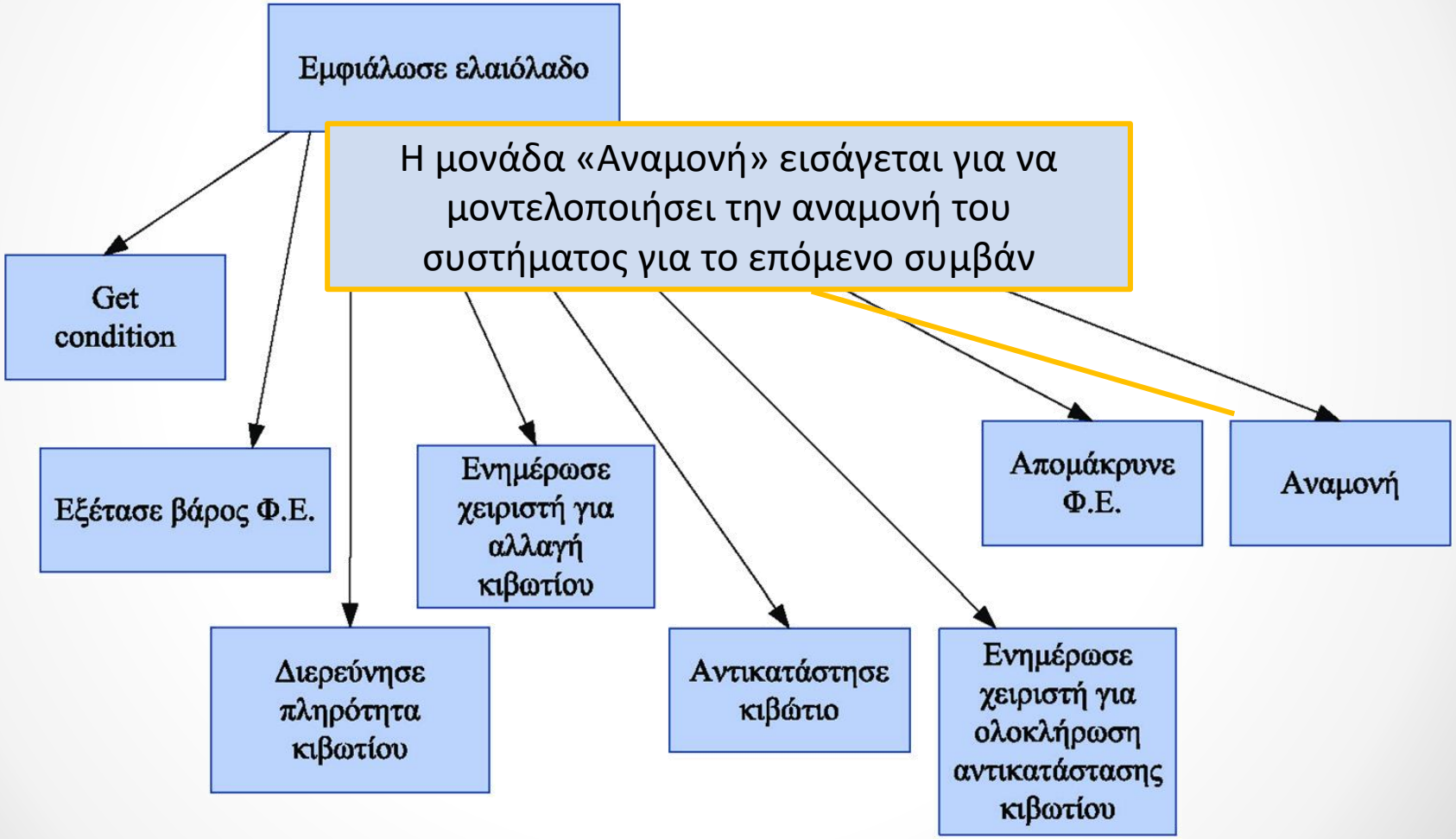
Μετατροπή διαγράμματος STD: παράδειγμα

4. Δομικό διάγραμμα συστήματος εμφιάλωσης ελαιολάδου



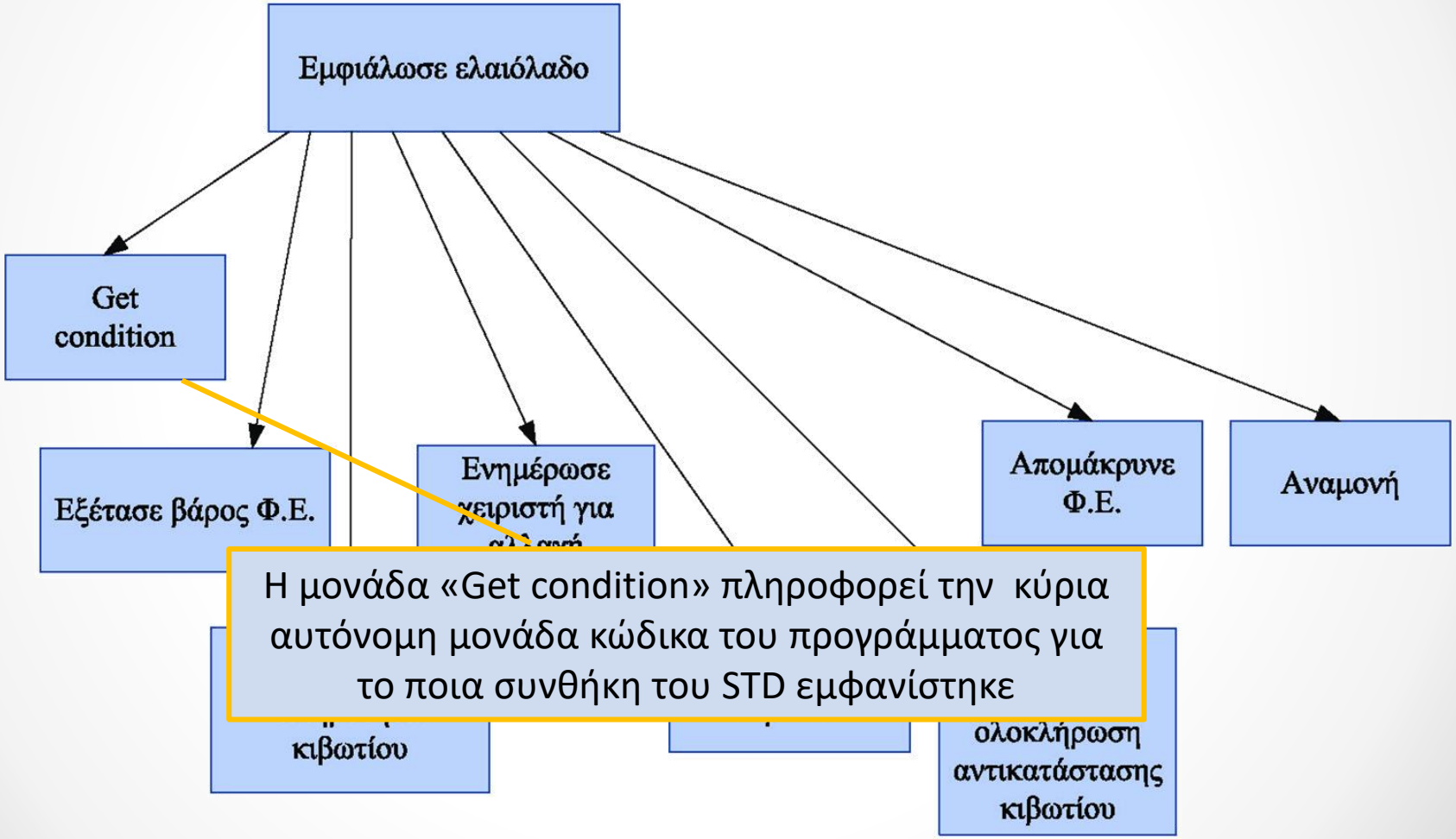
Μετατροπή διαγράμματος STD: παράδειγμα

4. Δομικό διάγραμμα συστήματος εμφιάλωσης ελαιολάδου



Μετατροπή διαγράμματος STD: παράδειγμα

4. Δομικό διάγραμμα συστήματος εμφιάλωσης ελαιολάδου

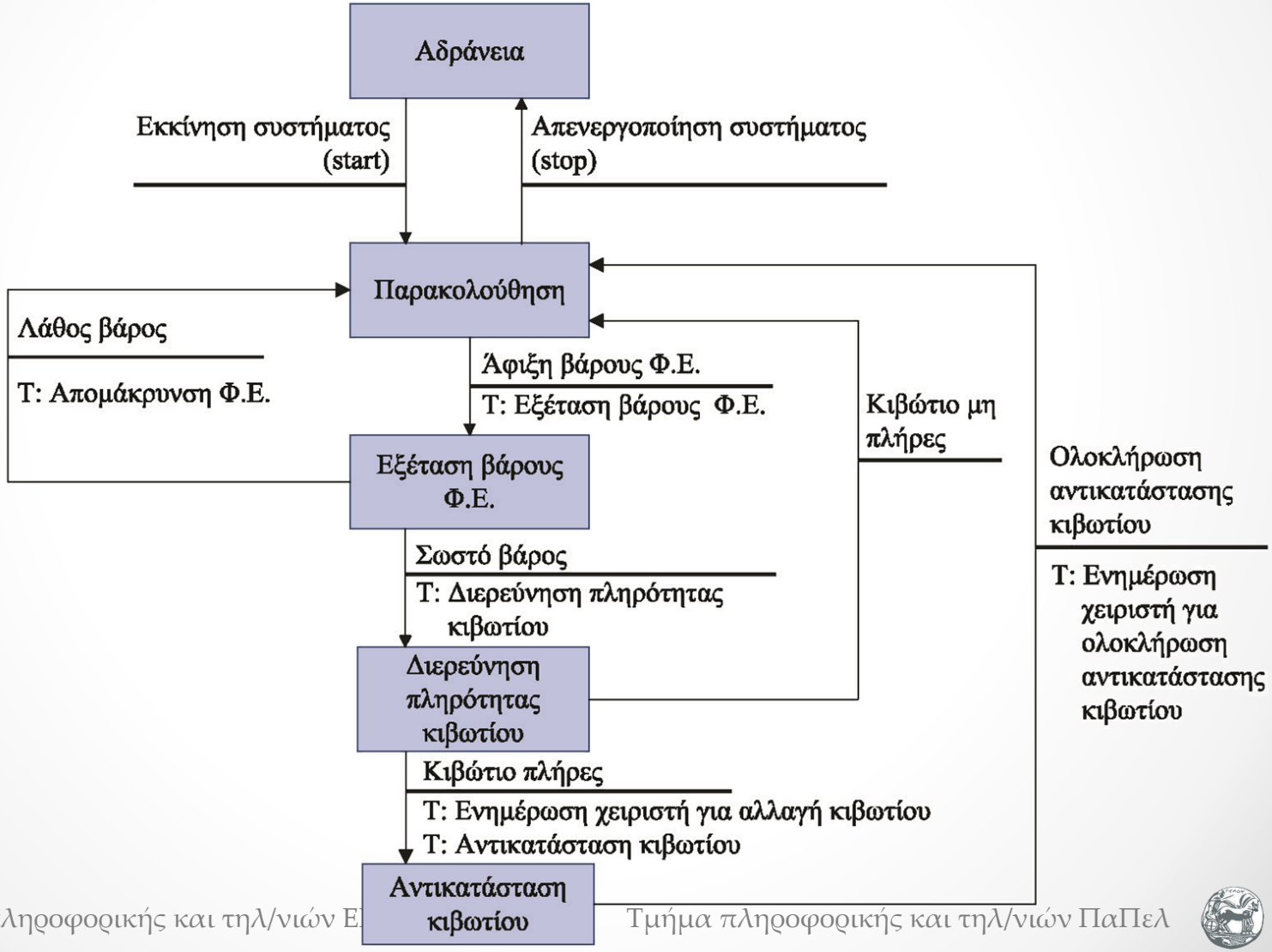


Πίνακες μετάβασης καταστάσεων

- Ο πίνακας μετάβασης καταστάσεων (state transition table – STT) είναι ένας πίνακας όπου:
  - οι συνθήκες του STD είναι οι στήλες του πίνακα
  - οι καταστάσεις του STD είναι οι γραμμές του
  - Τα κελιά συμπληρώνονται με τις μεταβάσεις του STD. Σε κάθε κελί καταγράφονται οι ενέργειες και η νέα κατάσταση
- Η κατασκευή του βασίζεται σε ένα μετασχηματισμένο STD, όπου όλες οι μεταβατικές καταστάσεις έχουν μετατραπεί σε πραγματικές καταστάσεις
  - Αυτό είναι απαραίτητο γιατί *κάθε* μετάβαση στην πραγματική υλοποίηση χρειάζεται κάποιον μη μηδενικό χρόνο για να εκτελεστεί και κάποιες ενέργειες από πλευράς συστήματος
- Η χρήση του συνίσταται όταν η λογική εντός του STD διαγράμματος δεν είναι απλοϊκή και η γλώσσα προγραμματισμού που θα χρησιμοποιήσουμε επιτρέπει την αξιοποίησή του

Δημιουργία STT: μετασχηματισμός STD

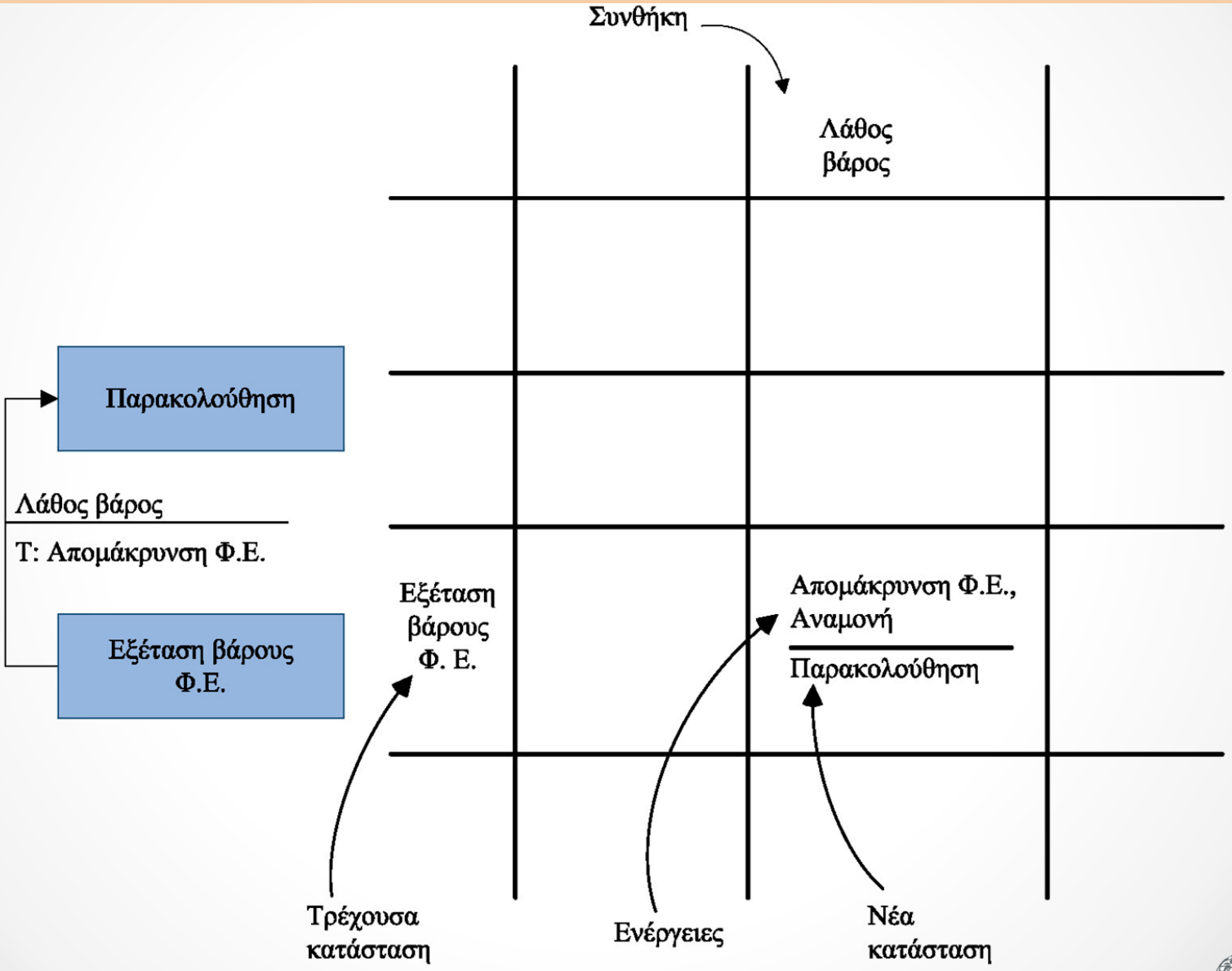
- STD με ανάλυση των μεταβατικών καταστάσεων



Δημιουργία STT: συνθήκες και καταστάσεις

- Πέντε καταστάσεις που αντιστοιχούν σε ισάριθμες γραμμές:
  1. Αδράνεια
  2. Παρακολούθηση
  3. Έλεγχος βάρους Φ.Ε.
  4. Διερεύνηση πληρότητας κιβωτίου
  5. Αντικατάσταση κιβωτίου
- Οκτώ συνθήκες οι οποίες αντιστοιχούν σε ισάριθμες στήλες:
  1. Εκκίνηση συστήματος
  2. Απενεργοποίηση συστήματος
  3. Άφιξη βάρους Φ.Ε.
  4. Σωστό Βάρος
  5. Λάθος βάρος
  6. Κιβώτιο πλήρες
  7. Κιβώτιο μη πλήρες
  8. Ολοκλήρωση αντικατάστασης κιβωτίου

Δημιουργία STT: περιεχόμενα κελιών



STT συστήματος εμφιάλωσης ελαιολάδου

	Εκκίνηση συστήματος	Απενεργο-ποίηση συστήματος	Άφιξη βάρους Φ.Ε.	Λάθος βάρος	Σωστό βάρος	Κιβώτιο πλήρες	Κιβώτιο μη πλήρες	Ολοκλήρωση αντικατάστασης κιβωτίου
Αδράνεια	Αναμονή							
	Παρακολού-θηση							
Παρακο-λούθηση			Εξέταση βάρους Φ.Ε.					
		Αδράνεια	Εξέταση βάρους Φ.Ε.					
Εξέταση βάρους Φ. Ε.				Απομά-κρυνση Φ.Ε., Αναμονή	Διερεύ-νηση πλη-ρότητας κιβωτίου			
				Παρακο-λούθηση	Διερεύ-νηση πλη-ρότητας κιβωτίου			
Εξέταση πληρό-τητας κιβωτίου						Ενημέρωση χειριστή για αλ-λαγή κιβωτίου, Αντικατάσταση κιβωτίου	Αναμονή	
						Αντικατάσταση κιβωτίου	Παρακο-λούθηση	
Αντικα-τάσταση κιβωτίου								Ενημέρωση χειρι-στή για ολοκλή-ρωση αντικατά-στασης κιβωτίου, Αναμονή
								Παρακολούθηση

Απεικόνιση STT σε κώδικα: διαδικασία

- Με βάση το STT είναι εύκολο να συγγράψουμε κώδικα που να υλοποιεί τη λειτουργικότητα:
  - Λαμβάνουμε κάθε φορά την επόμενη συνθήκη
  - Εξετάζουμε την τρέχουσα κατάσταση
  - Εκτελούμε τις ενέργειες που υποδεικνύονται από το κελί (τρέχουσα κατάσταση, συνθήκη) του STT
  - Μεταβαίνουμε στη νέα κατάσταση που υποδεικνύει το κελί (τρέχουσα κατάσταση, συνθήκη) του STT
  - Αν κάποιο κελί (τρέχουσα κατάσταση, συνθήκη) του STT είναι κενό, τότε δεν πραγματοποιείται καμία ενέργεια και δεν μεταβάλλεται και η τρέχουσα κατάσταση

Απεικόνιση STT σε κώδικα: παράδειγμα

```
BEGIN
        State := Αδράνεια /* Αρχική κατάσταση = αρχική κατάσταση του STD */
        FOREVER DO
        BEGIN
                CALL get condition(:condition)
                action := action-table(state, condition)
                new_state := state-table(state, condition)
                IF (new_state <> null) then
                        state = new_state
                END IF
                CASE action OF
                        null : /* do nothing */
                        Αναμονή :
                                CALL Αναμονή
                        Διαδικασία εξέτασης βάρους Φ.Ε.:
                                CALL Διαδικασία εξέτασης βάρους Φ.Ε.
                        Διαδικασία απομάκρυνσης Φ.Ε.:
                                CALL Διαδικασία απομάκρυνσης Φ.Ε.
                                CALL Αναμονή
                        …
                END CASE
        END /* FOREVER */
END
```

Επιθεώρηση STT: πληρότητα πίνακα

- Κατά την ανάπτυξη του STD, υποθέτουμε ότι τα πράγματα συμβαίνουν μόνο όταν εμείς θέλουμε να συμβούν, όταν δηλαδή έχουμε μία κατάσταση που τις αναμένει
  - Θεωρούμε δηλαδή ιδανικές συνθήκες
  - Π.χ. στην κατάσταση «Αδράνεια» δεν αναμένουμε τη συνθήκη «Κιβώτιο πλήρες» και δεν τη μοντελοποιούμε
  - Έτσι στο STT το σχετικό κελί θα είναι κενό
- Σε αυτή τη φάση θα πρέπει να εξετάσουμε ποιες συνθήκες είναι δυνατόν να εμφανιστούν σε μη επιθυμητές για τις συγκεκριμένες συνθήκες καταστάσεις και να προβλέψουμε αντίστοιχες ενέργειες
  - Π.χ. ποιες ενέργειες θα κάνει ένα ATM αν στη διάρκεια πληκτρολόγησης του PIN αφαιρεθεί η κάρτα; (μη φυσιολογική ροή/εμφάνιση συνθήκης σε μη επιθυμητή κατάσταση)
  - Με αυτόν τον τρόπο καλύπτουμε τις περιπτώσεις σφαλμάτων και οδηγούμαστε σε εύρωστο σύστημα

Επιθεώρηση STT: πληρότητα πίνακα (2)

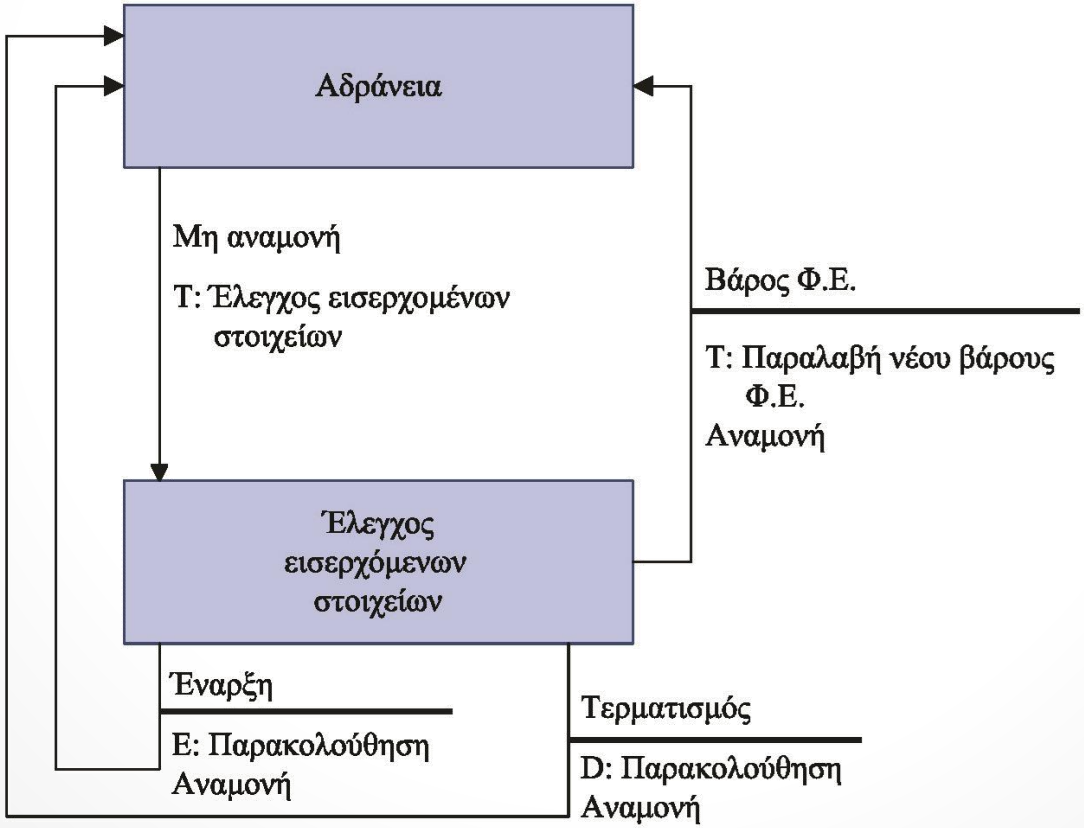
- Εκτός από τις ανεπιθύμητες συνθήκες που εμφανίζονται και πρέπει να τύχουν χειρισμού σε συγκεκριμένες καταστάσεις, υπάρχουν και αυτές που μπορεί να εμφανιστούν σε οποιαδήποτε κατάσταση
  - Π.χ. διακοπή ρεύματος, λήψη σήματος υπερθέρμανσης, πάτημα πλήκτρου ακύρωσης κ.ο.κ.
- Για τις συνθήκες αυτές πολλές φορές παραλείπουμε την απεικόνιση στο STD για να αποφύγουμε την υπέρμετρη αύξηση της πολυπλοκότητας και τις μοντελοποιούμε απ΄ ευθείας στο STT, προσθέτοντας μία στήλη

Μετατροπή STD σε κώδικα με άμεση κωδικοποίηση

- Αν η λογική εντός του STD είναι πολύ απλή, οι επιβαρύνσεις της δημιουργίας, αποθήκευσης και χρήσης του STT μπορεί να είναι δυσανάλογα υψηλές
- Ένα STD με πολύ απλή λογική μπορεί εύκολα να μετατραπεί άμεσα σε κώδικα χωρίς τη «μεσολάβηση» του STT
- Η προσέγγιση αυτή καλείται «Άμεση κωδικοποίηση», βασίζεται στην απλή ανάγνωση του STD, την κατανόησή του και τη συγγραφή σχετικού κώδικα
  - Η εμπειρία στη συγγραφή κώδικα και στη συγκεκριμένη τεχνική μετατροπής είναι καθοριστικοί παράγοντες για την ποιότητα του κώδικα

Μετατροπή STD σε κώδικα με άμεση κωδικοποίηση: παράδειγμα απλού STD

- Θεωρούμε το απλό STD του σχήματος
  - Δείχνει ότι θα πρέπει να αναμένουμε για εισερχόμενα δεδομένα, να εξετάσουμε τα δεδομένα και να εκτελέσουμε μία από τις τρεις ενέργειες ανάλογα με το περιεχόμενο των δεδομένων



Μετατροπή STD σε κώδικα με άμεση κωδικοποίηση: προκύπτων κώδικας

- Με άμεση κωδικοποίηση μπορούμε να καταλήξουμε στον εξής κώδικα:

```
FOREVER DO
    BEGIN
        CALL έλεγχος εισερχομένων στοιχείων (:data-type)
        CASE data-type OF
            Έναρξη :
                παρακολούθηση := true
            Τερματισμός:
                παρακολούθηση:= false
            Παραλαβή νέου βάρους Φ.Ε.:
                CALL παραλαβή νέου βάρους Φ.Ε.
        END CASE
        CALL αναμονή
    END
```

- Ο κώδικας αυτός είναι πολύ απλούστερος σε σχέση με τον κώδικα που θα καταλήγαμε ακολουθώντας μία τυποποιημένη διαδικασία παραγωγής κώδικα μέσω του STT

Άμεση κωδικοποίηση: σημεία για προσοχή

- Αν η λογική του STD δεν είναι απλοϊκή, προκύπτει ένας αριθμός προβλημάτων σχετικών με τη χρήση της άμεσης κωδικοποίησης, όπως:
  - όσο πιο σύνθετη είναι η λογική, τόσο πιο εύκολο είναι να κάνουμε λάθη κατά τη μετατροπή της σε κώδικα.
  - είναι πολύ πιο δύσκολο να επιβεβαιώσουμε τον κώδικα σε συνάρτηση με το αρχικό STD, σε σχέση με την κωδικοποίηση με χρήση STT, ιδίως κατά τη φάση της συντήρησης.
    - Μία αλλαγή στο STD μπορεί εύκολα να αποτυπωθεί στο STT, ενώ μπορεί να απαιτεί μη προφανείς αλλαγές στον κώδικα.
  - απρόσμενες συνθήκες, π.χ. διαχείριση σφαλμάτων, δεν λαμβάνονται υπόψη ρητά σε αυτή τη μέθοδο, πράγμα που σημαίνει ότι είναι πιο εύκολο να τις αγνοήσουμε και πιο δύσκολο να τις χειριστούμε.

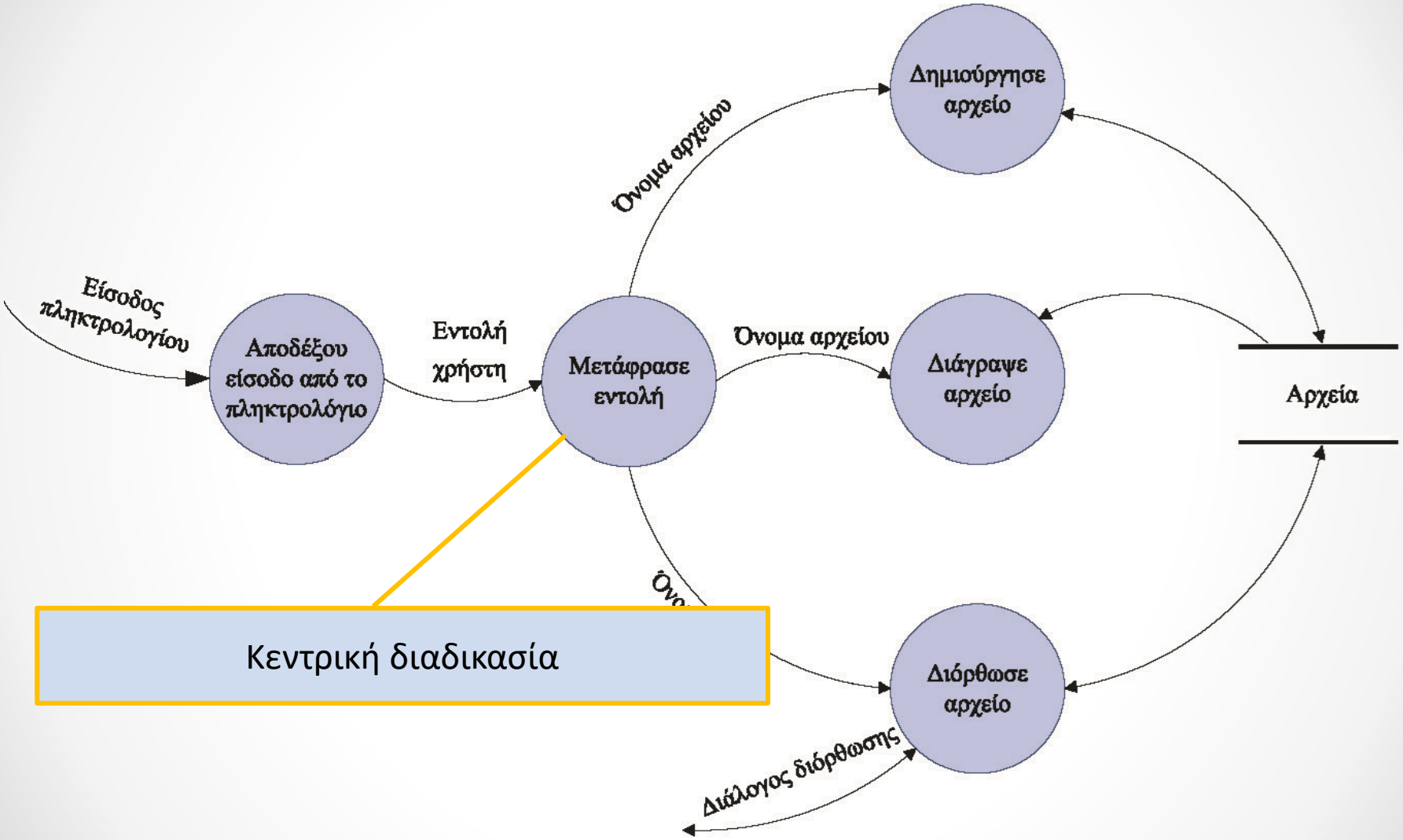
Δημιουργία προκαταρκτικού δομικού διαγράμματος από DFD

- Μπορούμε να προχωρήσουμε στη δημιουργία προκαταρκτικού δομικού διαγράμματος με βάση το DFD
- Το πρώτο βήμα είναι να εντοπίσουμε μία διαδικασία στο DFD που θα αποτελέσει την κύρια μονάδα του προγράμματος
  - Αυτό θα αποτελέσει το «κύριο πρόγραμμα» και αποτελεί τη βάση για την κατασκευή του δομικού διαγράμματος
  - Συνήθως υπάρχει μία διαδικασία στο DFD που, σε εννοιολογικό επίπεδο, είτε ελέγχει, είτε είναι ενήμερη για τη λειτουργία των υπόλοιπων διαδικασιών του διαγράμματος DFD: αυτή η διαδικασία θα επιλεγεί
  - Ο εντοπισμός γίνεται με βάση το αν το DFD είναι προσανατολισμένο σε συναλλαγές (transaction-oriented) ή σε μετασχηματισμούς (transform-oriented)

Εισαγωγή ιεραρχίας σε ένα DFD προσανατολισμένο σε συναλλαγές

- Σε ένα DFD προσανατολισμένο σε συναλλαγές τυπικά έχουμε συλλογή εισόδων, εξέτασή τους και εκτέλεση ενεργειών, ανάλογα με τις εισόδους
- Τυπικά, η διαδικασία που συλλέγει τις εισόδους και αποφασίζει ποιες ενέργειες θα εκτελεστούν αναδεικνύεται ως κύρια μονάδα του προγράμματος

Εισαγωγή ιεραρχίας σε ένα DFD προσανατολισμένο σε συναλλαγές: παράδειγμα

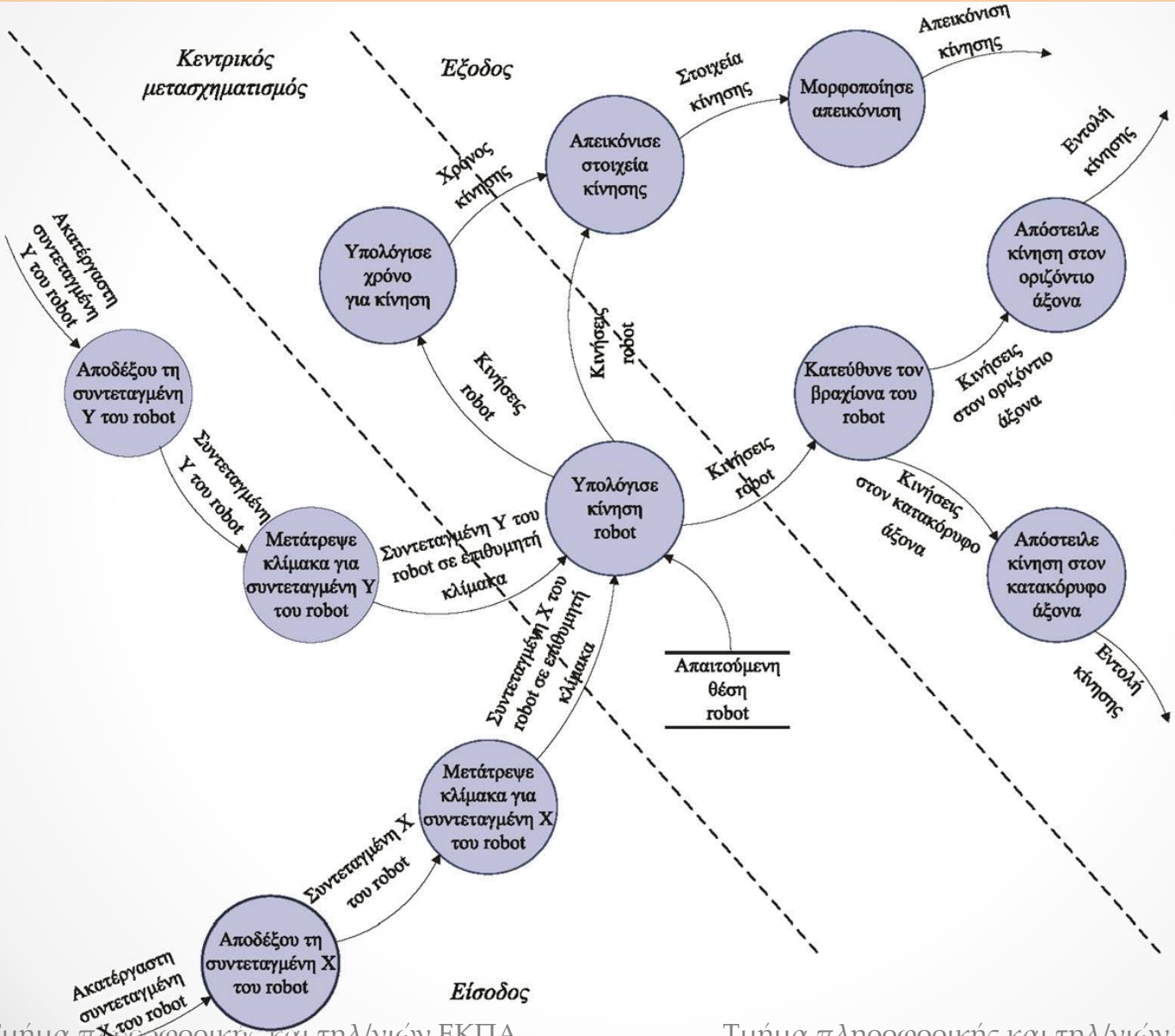


Εισαγωγή ιεραρχίας σε ένα DFD προσανατολισμένο σε μετασχηματισμούς

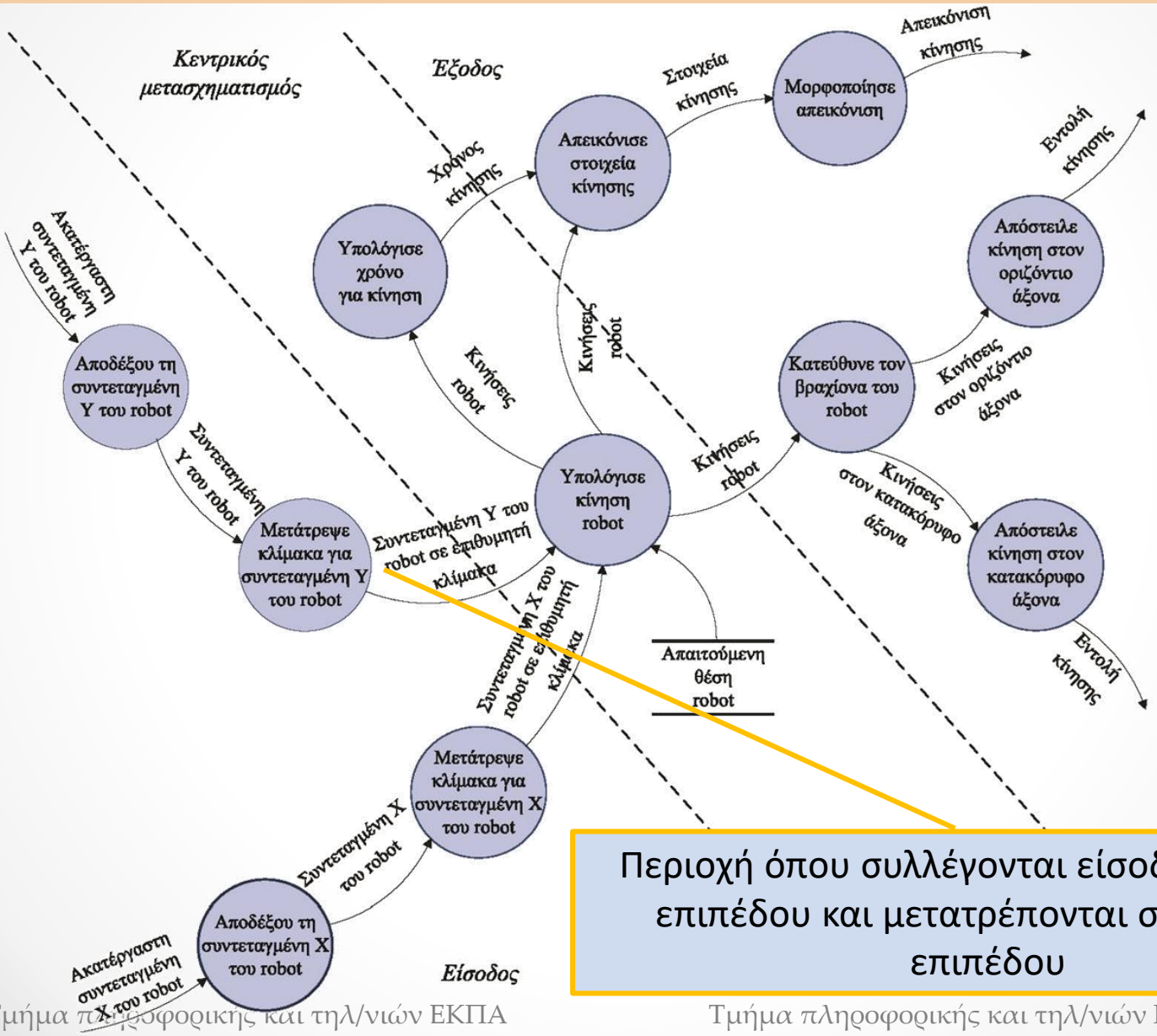
- Σε ένα DFD προσανατολισμένο σε μετασχηματισμούς τυπικά έχουμε:
  - Διαδικασίες που συλλέγουν εισόδους χαμηλού επιπέδου αφαίρεσης και τις συνθέτουν σε εισόδους υψηλού επιπέδου αφαίρεσης[1]
  - Διαδικασίες που εκτελούν τη βασική επεξεργασία-μετασχηματισμούς, παράγοντας εξόδους υψηλού επιπέδου αφαίρεσης
  - Διαδικασίες που μετατρέπουν τις υψηλού επιπέδου εξόδους σε εξόδους χαμηλού επιπέδου αφαίρεσης
- Τυπικά, η διαδικασία που μετασχηματίζει τις εισόδους υψηλού επιπέδου σε εξόδους υψηλού επιπέδου αναδεικνύεται ως κύρια μονάδα του προγράμματος

- [1] Τα πιο υψηλά επίπεδα αφαίρεσης εκφράζουν την πληροφορία με λιγότερες λεπτομέρειες, ενώ τα χαμηλά επίπεδα έχουν περισσότερες λεπτομέρειες. Για παράδειγμα, το «διάβασε τη θερμοκρασία» είναι πιο υψηλού επιπέδου περιγραφή από το «διάβασε έναν ακέραιο 16 bit από τη θύρα εισόδου 3245 και μετάτρεψέ τον σε πραγματικό στην περιοχή [0, 100]».

Εισαγωγή ιεραρχίας σε ένα DFD προσανατολισμένο σε μετασχηματισμούς: παράδειγμα

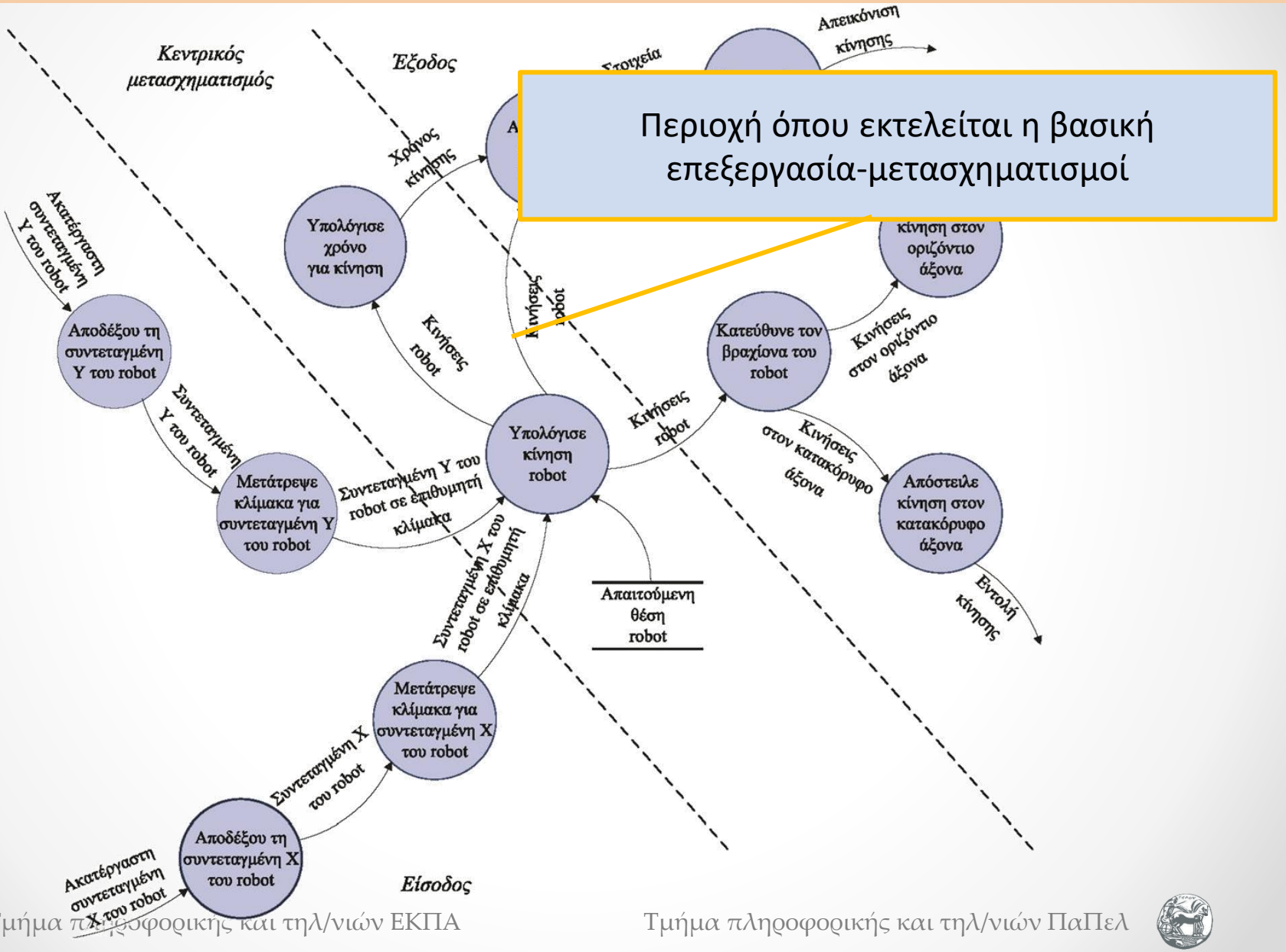


Εισαγωγή ιεραρχίας σε ένα DFD προσανατολισμένο σε μετασχηματισμούς: παράδειγμα

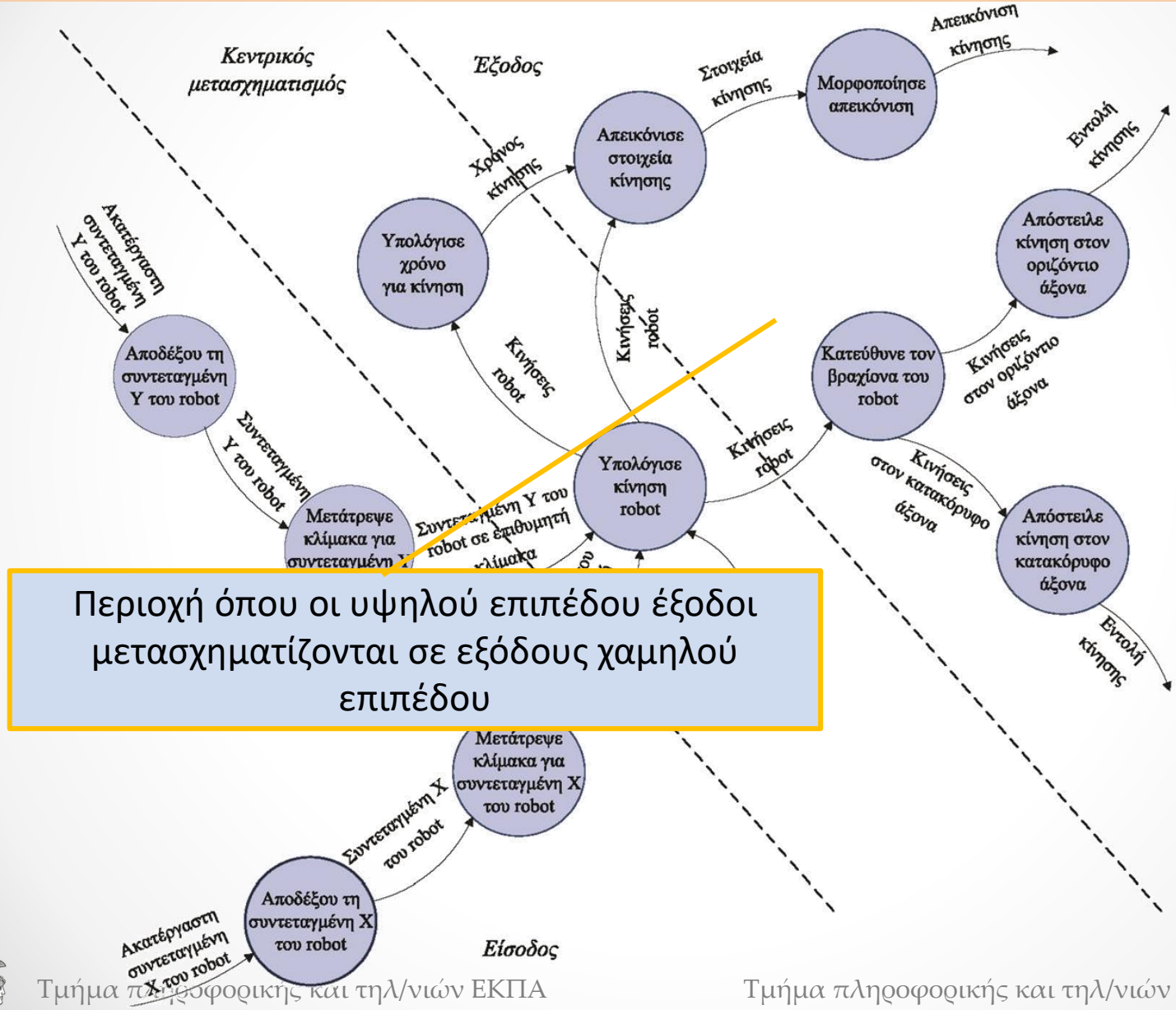


Περιοχή όπου συλλέγονται είσοδοι χαμηλού επιπέδου και μετατρέπονται σε υψηλού επιπέδου

Εισαγωγή ιεραρχίας σε ένα DFD προσανατολισμένο σε μετασχηματισμούς: παράδειγμα



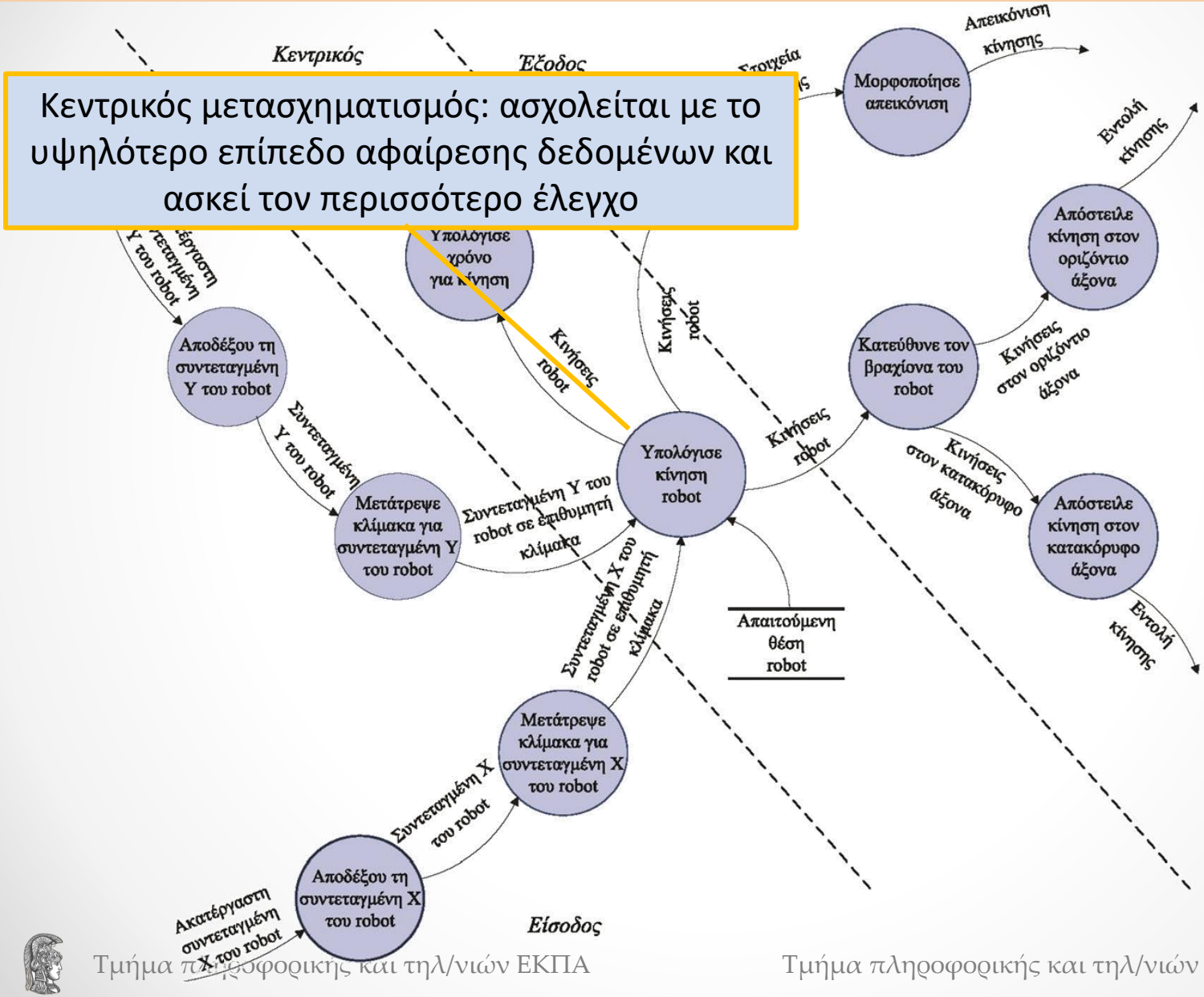
Εισαγωγή ιεραρχίας σε ένα DFD προσανατολισμένο σε μετασχηματισμούς: παράδειγμα



Περιοχή όπου οι υψηλού επιπέδου έξοδοι μετασχηματίζονται σε εξόδους χαμηλού επιπέδου

Εισαγωγή ιεραρχίας σε ένα DFD προσανατολισμένο σε μετασχηματισμούς: παράδειγμα

Κεντρικός μετασχηματισμός: ασχολείται με το υψηλότερο επίπεδο αφαίρεσης δεδομένων και ασκεί τον περισσότερο έλεγχο

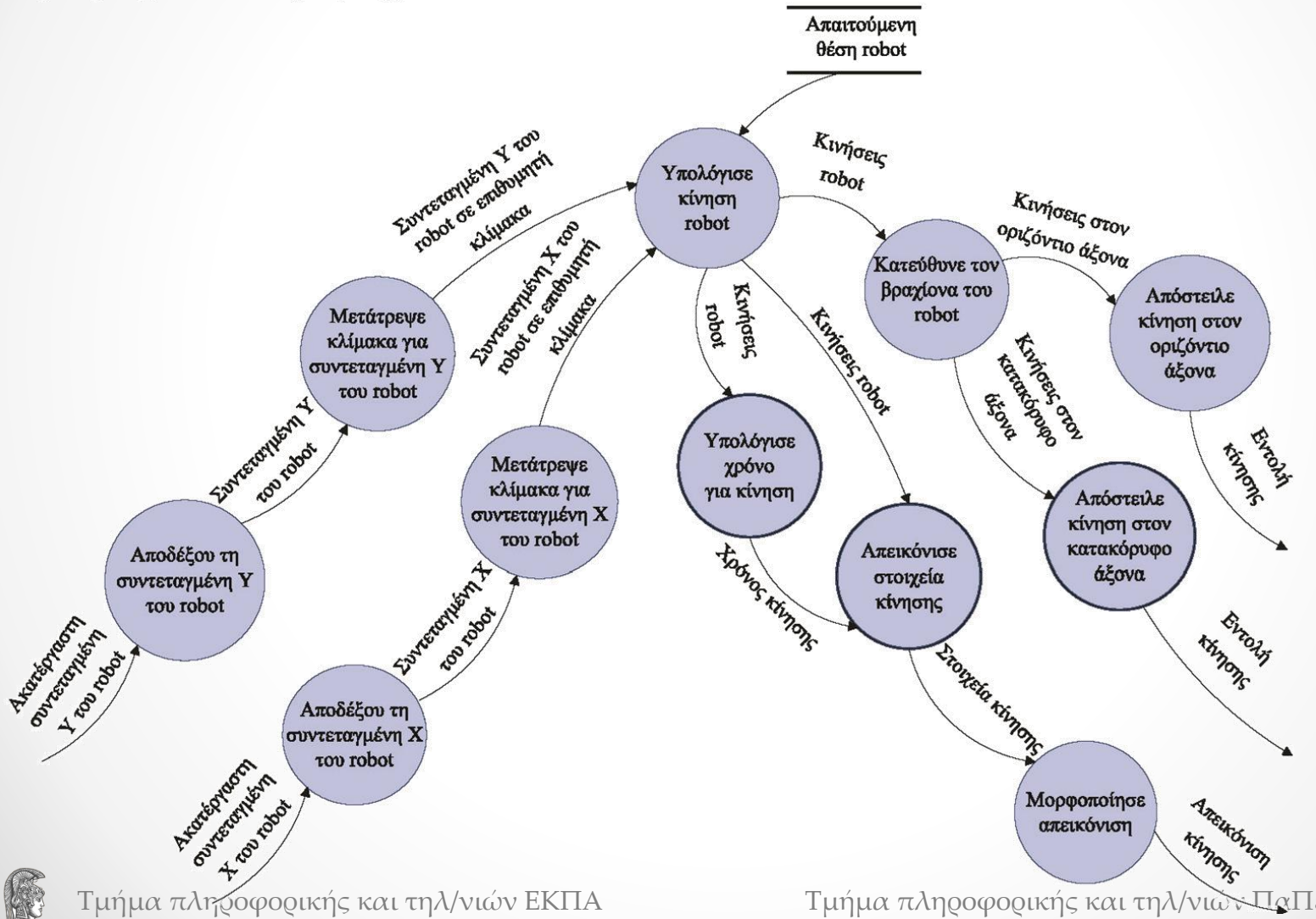


Η μετατροπή ενός διαγράμματος DFD

- Αφού αναδείξουμε τον ελεγκτικό μετασχηματισμό δεδομένων, εκτελούμε τα εξής βήματα:
  - Μεταφέρουμε τον ελεγκτικό μετασχηματισμό δεδομένων στην κορυφή του διαγράμματος
  - Αφήνουμε τους υπόλοιπους μετασχηματισμούς δεδομένων να «κρέμονται» κάτω από αυτό
  - Μετατρέπουμε τους μετασχηματισμούς δεδομένων σε αυτόνομες μονάδες κώδικα και τις ροές δεδομένων σε κλήσεις των αυτόνομων μονάδων κώδικα μαζί με τα αντίστοιχα ζεύγη δεδομένων
  - Οι μονάδες ονομάζονται ανάλογα με το τι πιστεύει η καλούσα ενότητα ότι κάνουν, παρά το τι πιστεύουν ότι κάνουν οι ίδιες οι μονάδες.

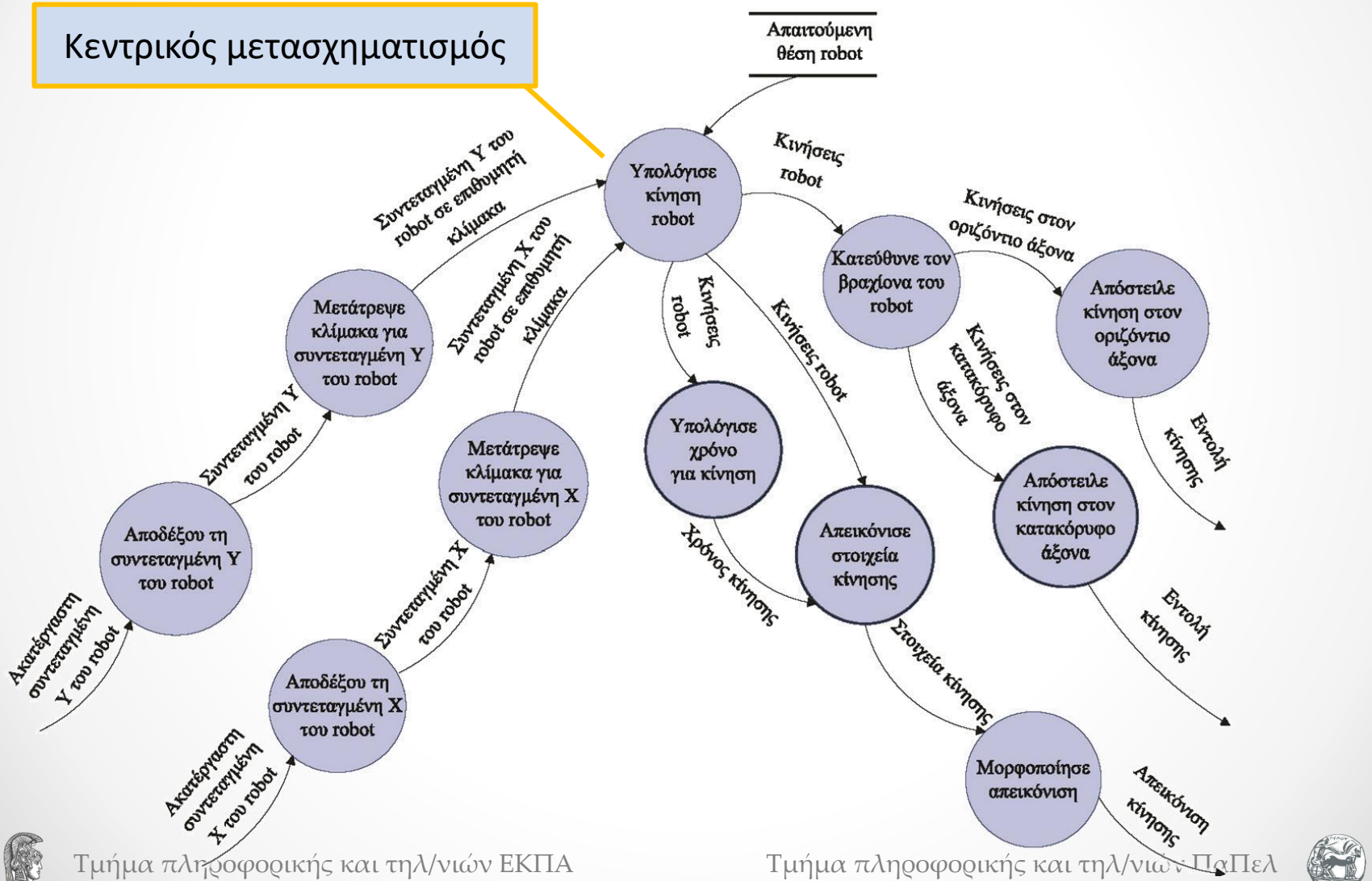
Η μετατροπή ενός διαγράμματος DFD: παράδειγμα

1. Μετακίνηση του κεντρικού μετασχηματισμού στην κορυφή



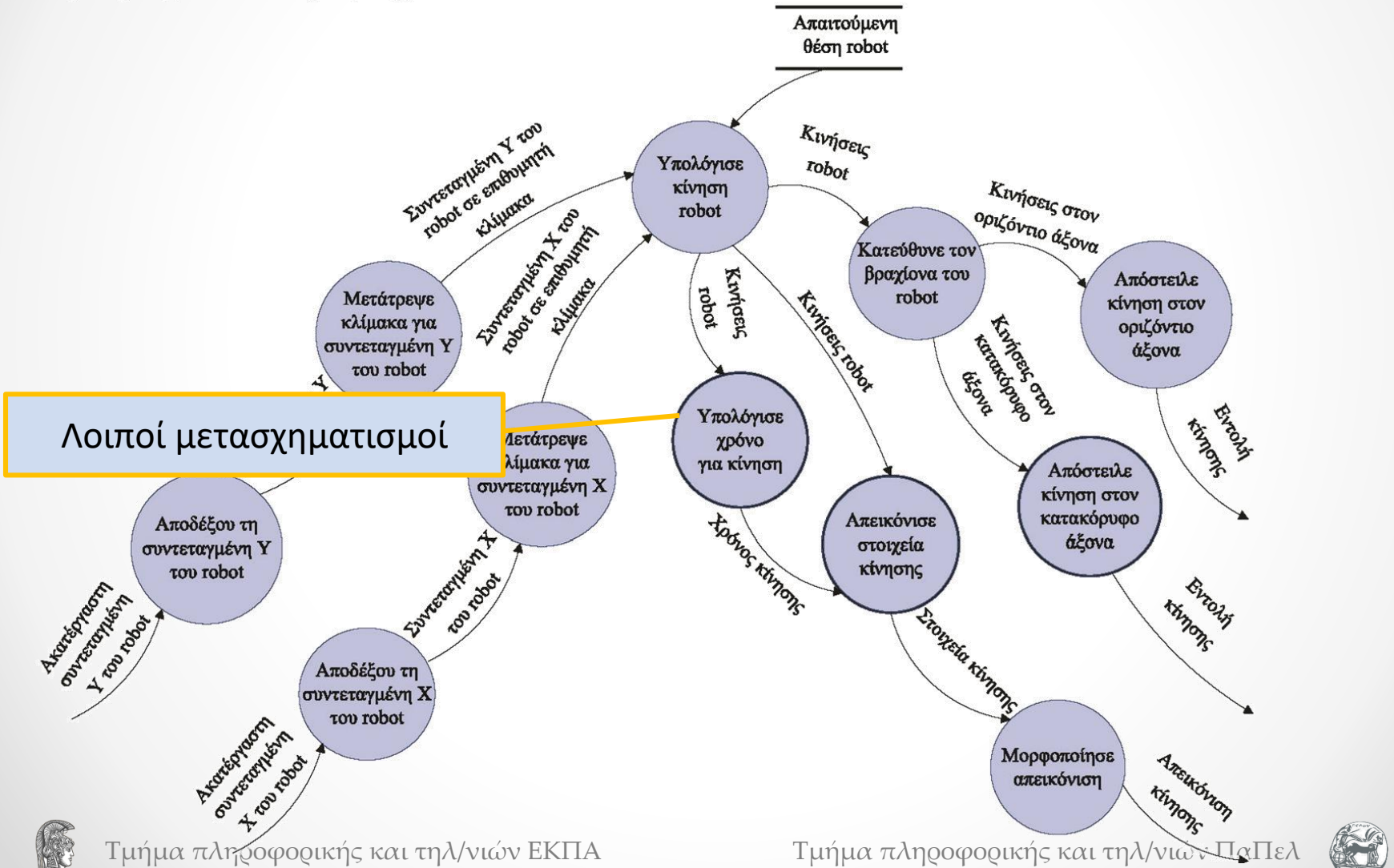
Η μετατροπή ενός διαγράμματος DFD: παράδειγμα

1. Μετακίνηση του κεντρικού μετασχηματισμού στην κορυφή



Η μετατροπή ενός διαγράμματος DFD: παράδειγμα

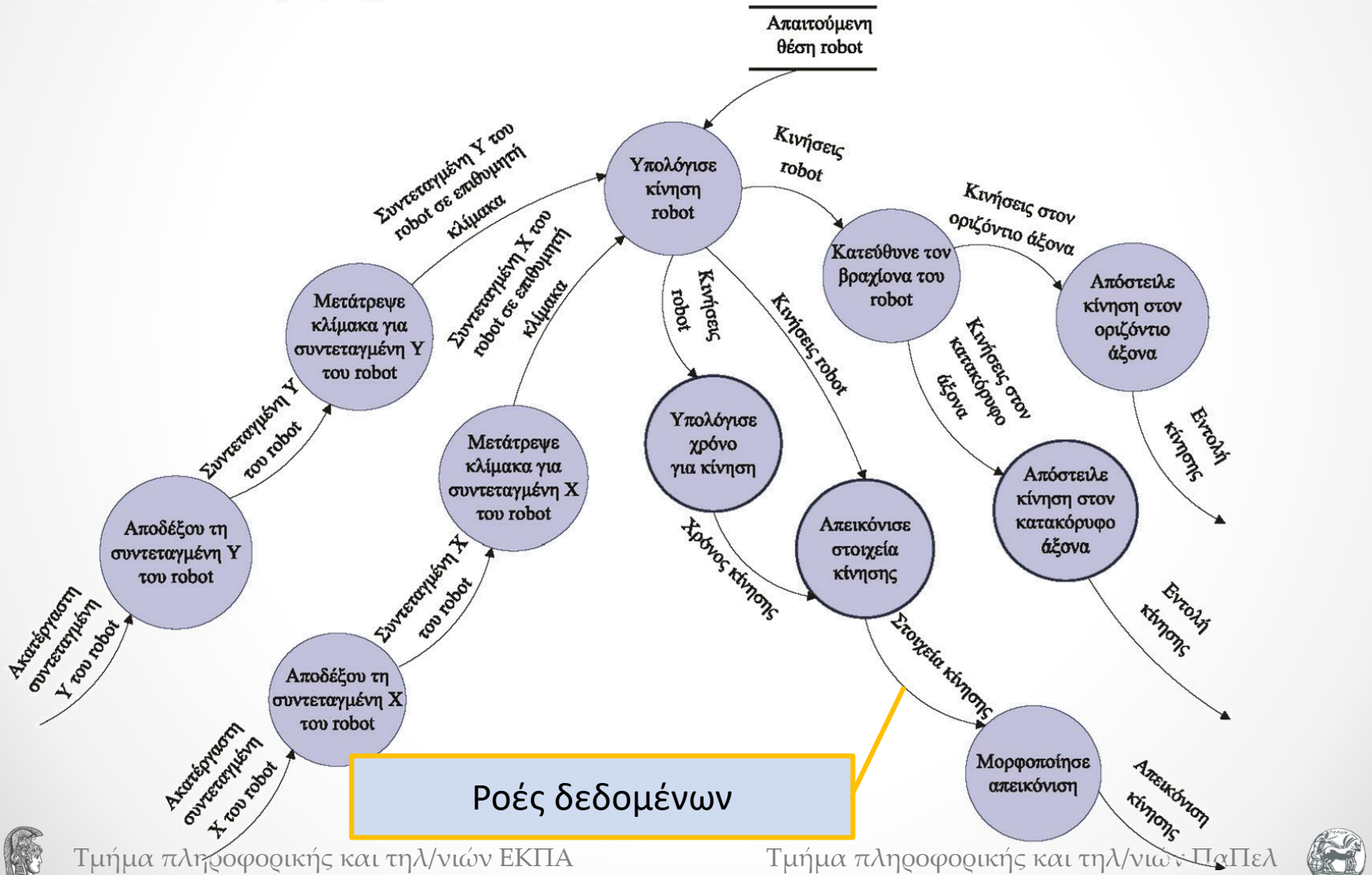
1. Μετακίνηση του κεντρικού μετασχηματισμού στην κορυφή



Λοιποί μετασχηματισμοί

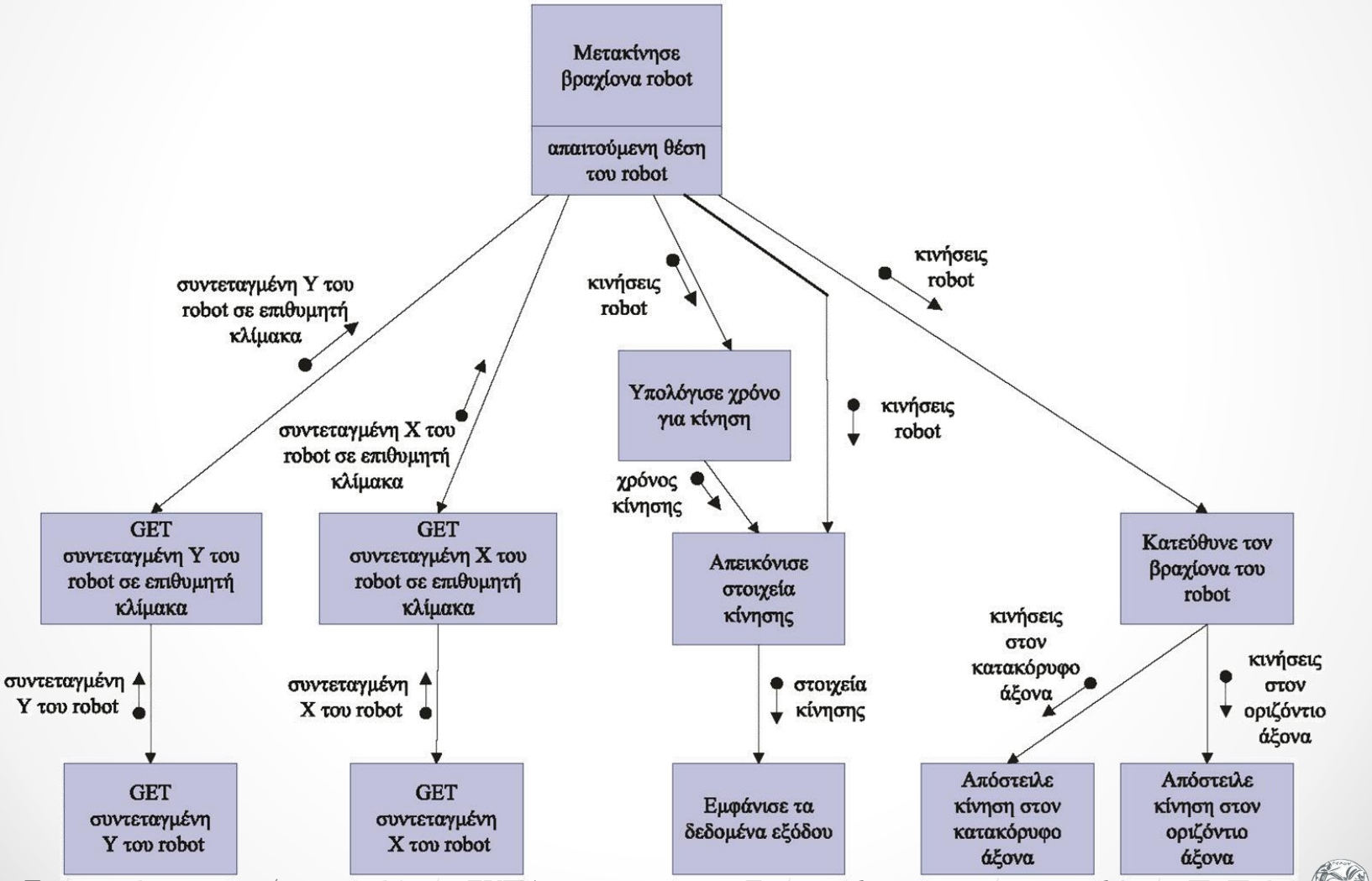
Η μετατροπή ενός διαγράμματος DFD: παράδειγμα

1. Μετακίνηση του κεντρικού μετασχηματισμού στην κορυφή



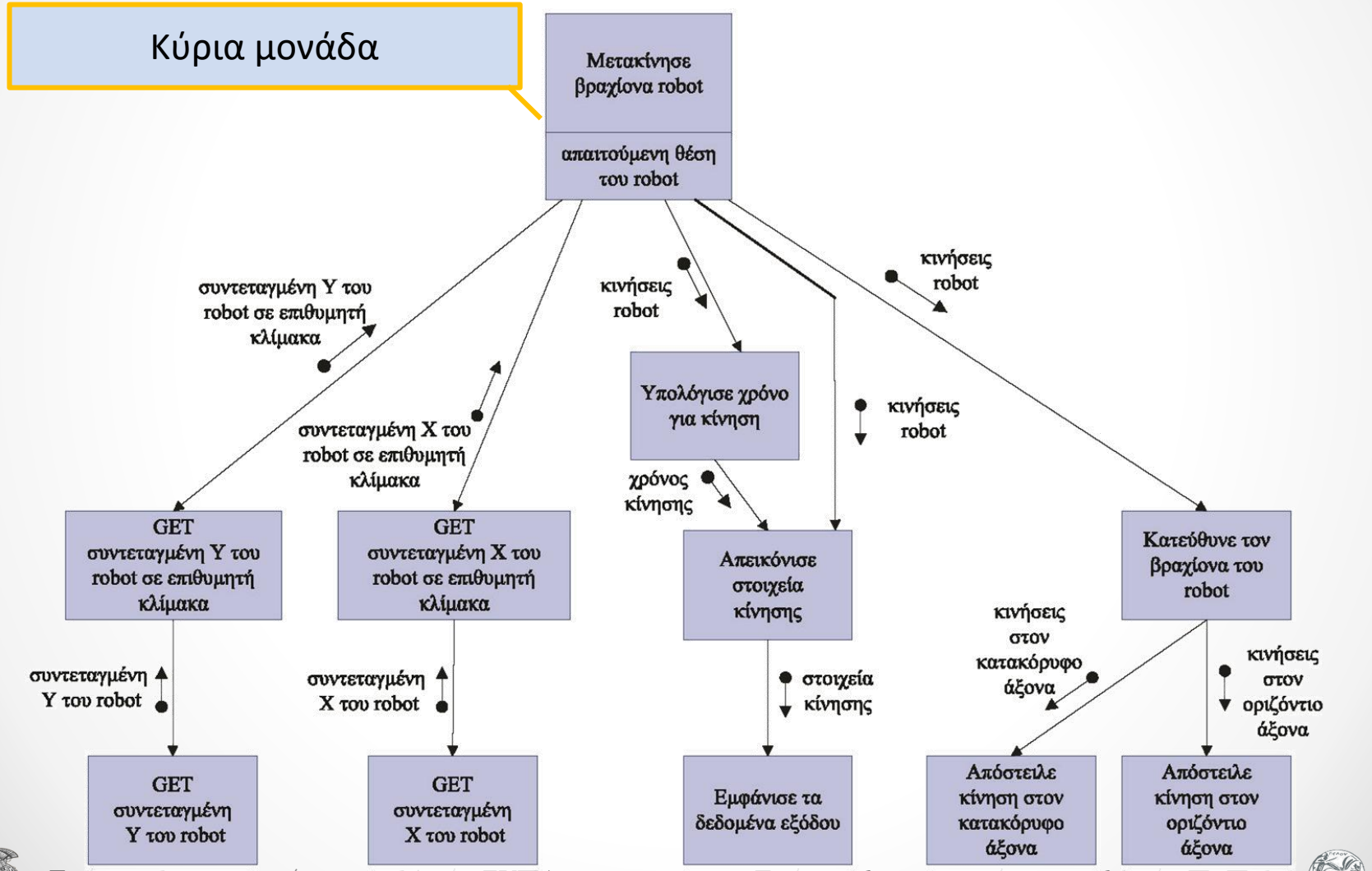
Η μετατροπή ενός διαγράμματος DFD: παράδειγμα

2. Εξαγωγή μονάδων από μετασχηματισμούς



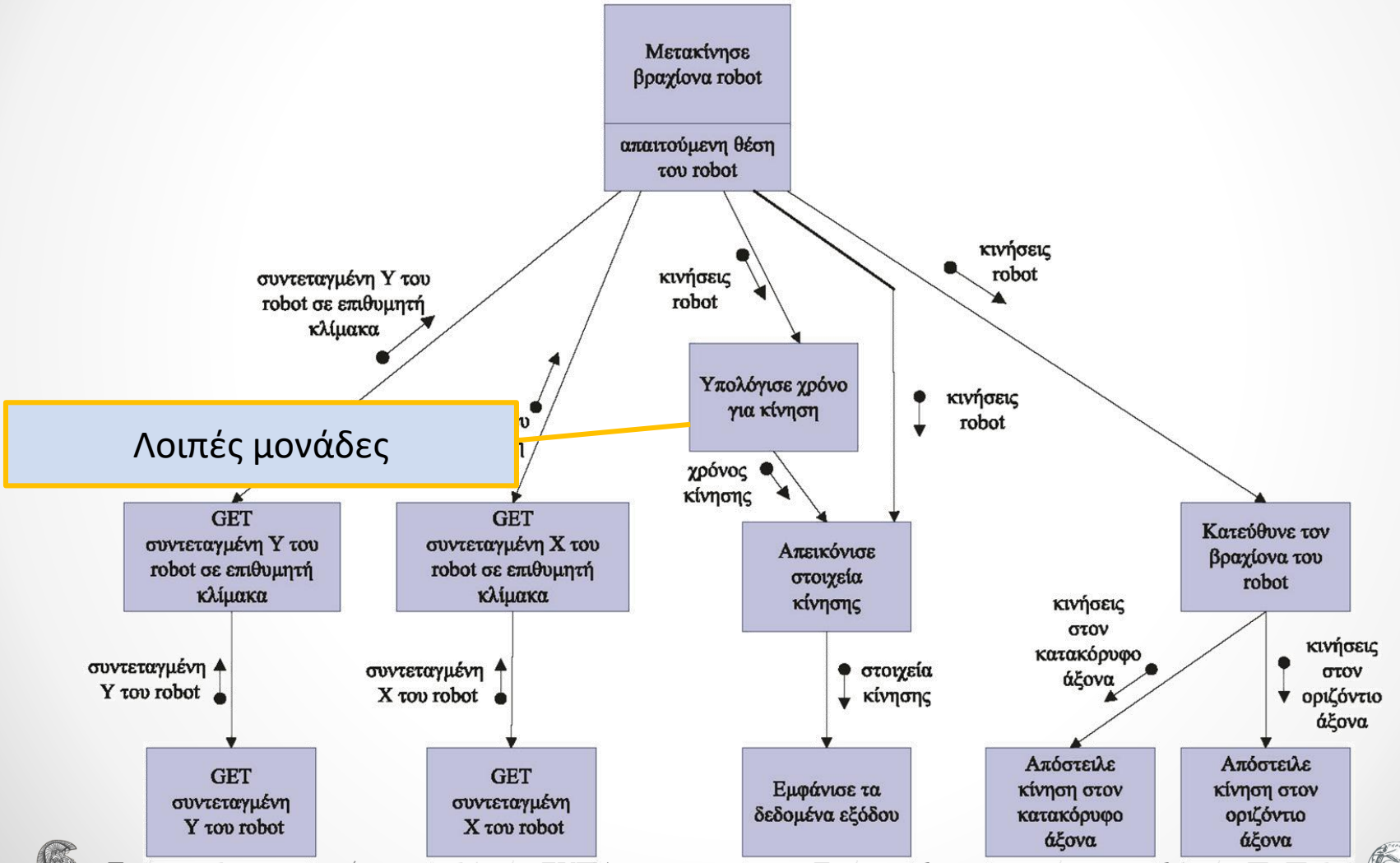
Η μετατροπή ενός διαγράμματος DFD: παράδειγμα

2. Εξαγωγή μονάδων από μετασχηματισμούς



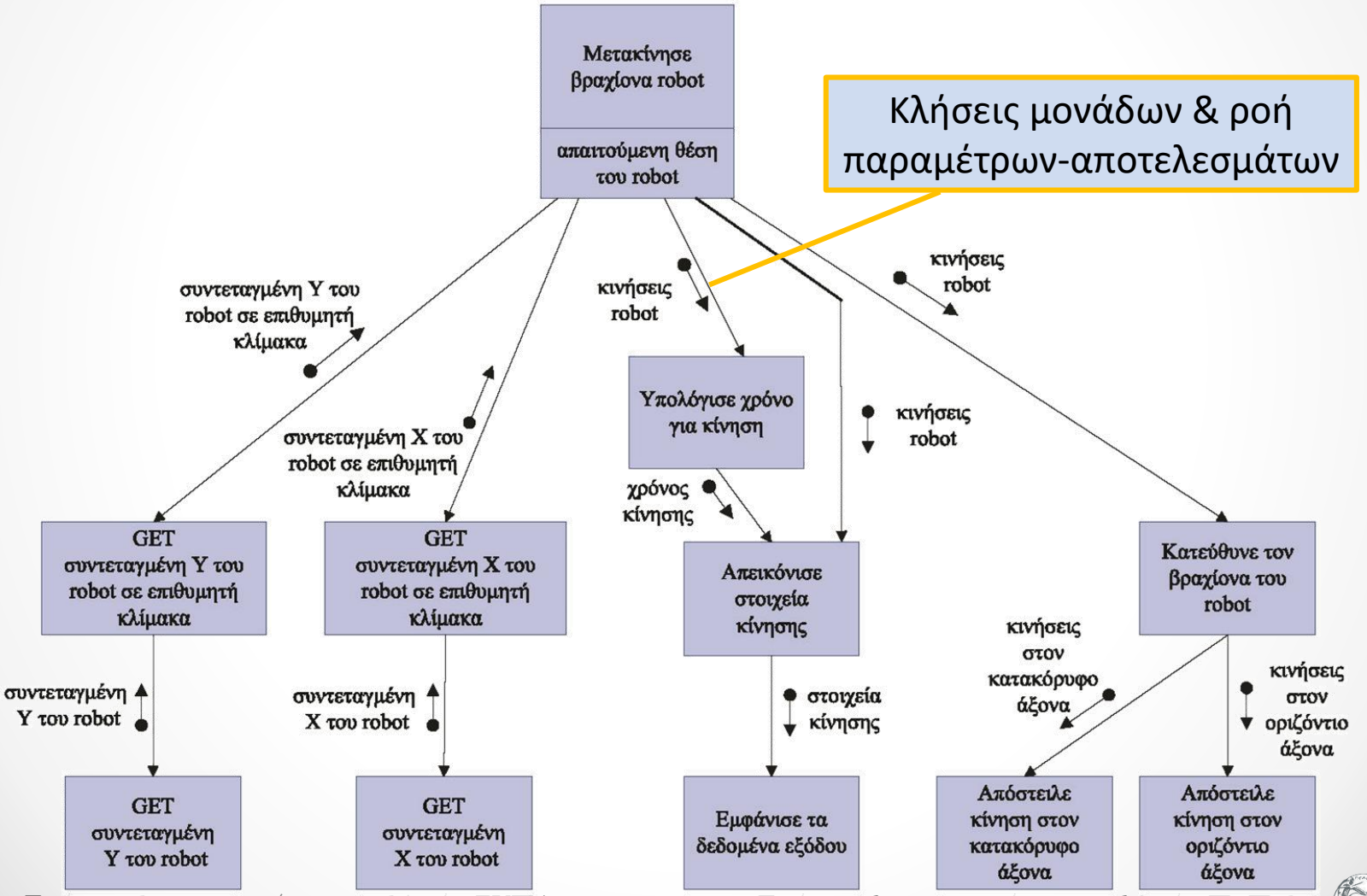
Η μετατροπή ενός διαγράμματος DFD: παράδειγμα

2. Εξαγωγή μονάδων από μετασχηματισμούς



Η μετατροπή ενός διαγράμματος DFD: παράδειγμα

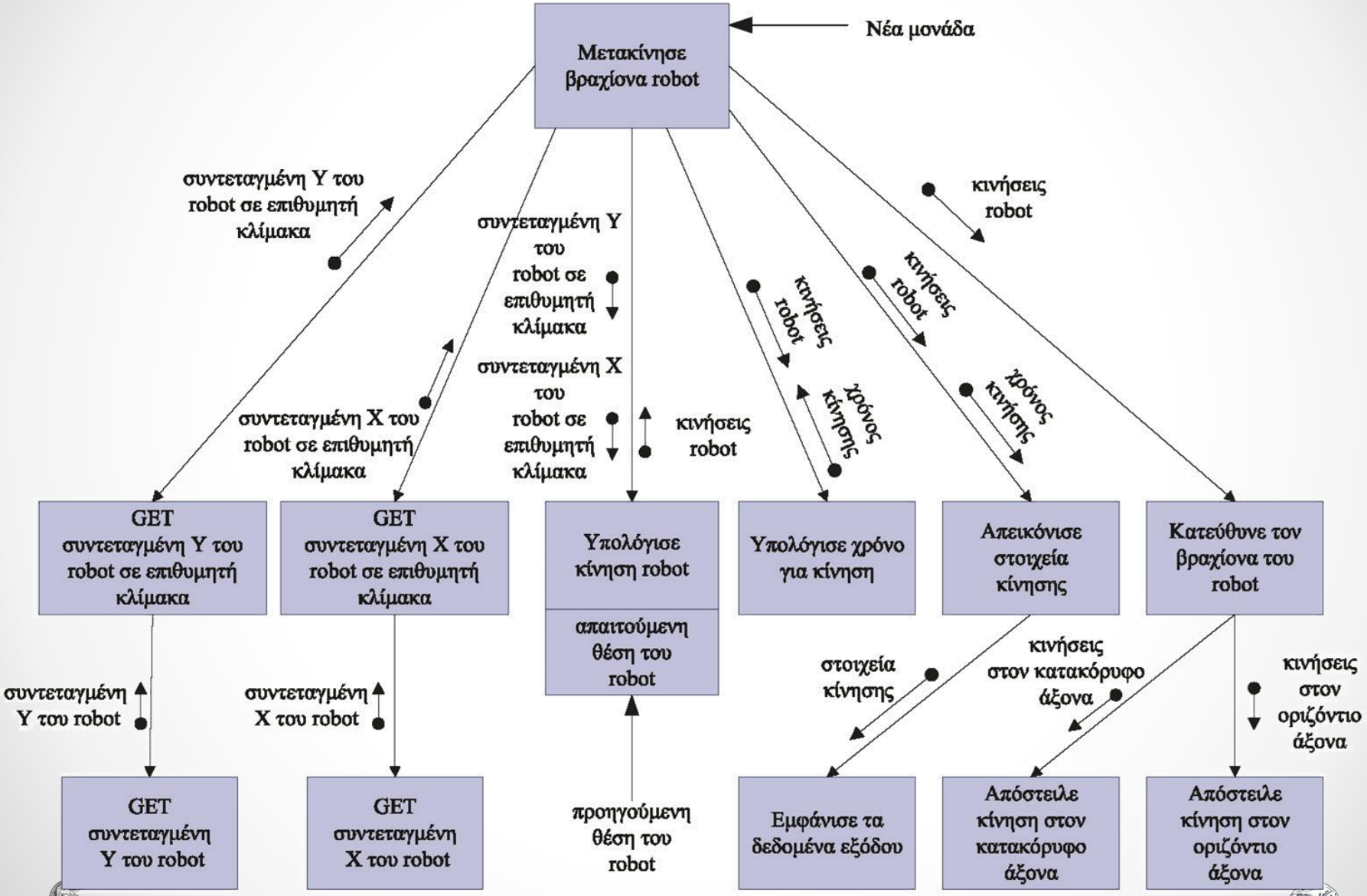
2. Εξαγωγή μονάδων από μετασχηματισμούς



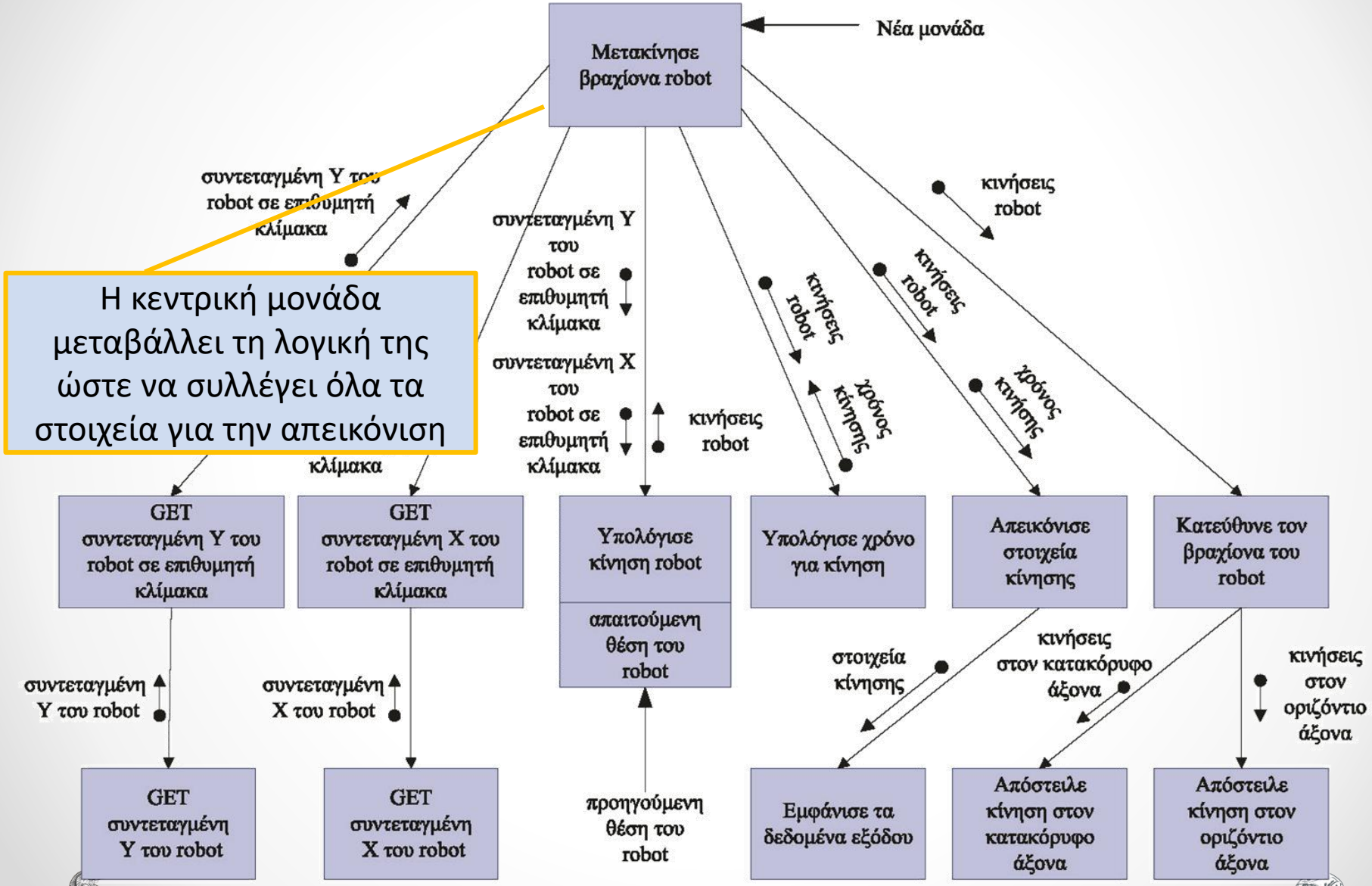
Βελτίωση του δομικού διαγράμματος

- Στο προηγούμενο διάγραμμα η ενότητα «Απεικόνισε στοιχεία κίνησης» καλείται από δύο διαφορετικές μονάδες με διαφορετικούς στόχους (απεικόνιση κινήσεων και απεικόνιση χρόνου κίνησης)
  - Αυτή η πρακτική δεν είναι συνιστώμενη, καθώς αυξάνει την πολυπλοκότητα και την πιθανότητα σφαλμάτων, μειώνει δε τη συντηρισιμότητα
- Το διάγραμμα μπορεί να ανασχεδιαστεί ώστε η κάθε μονάδα να επιτελεί έναν ακριβώς σκοπό, αντιμετωπίζοντας έτσι τα ανωτέρω προβλήματα

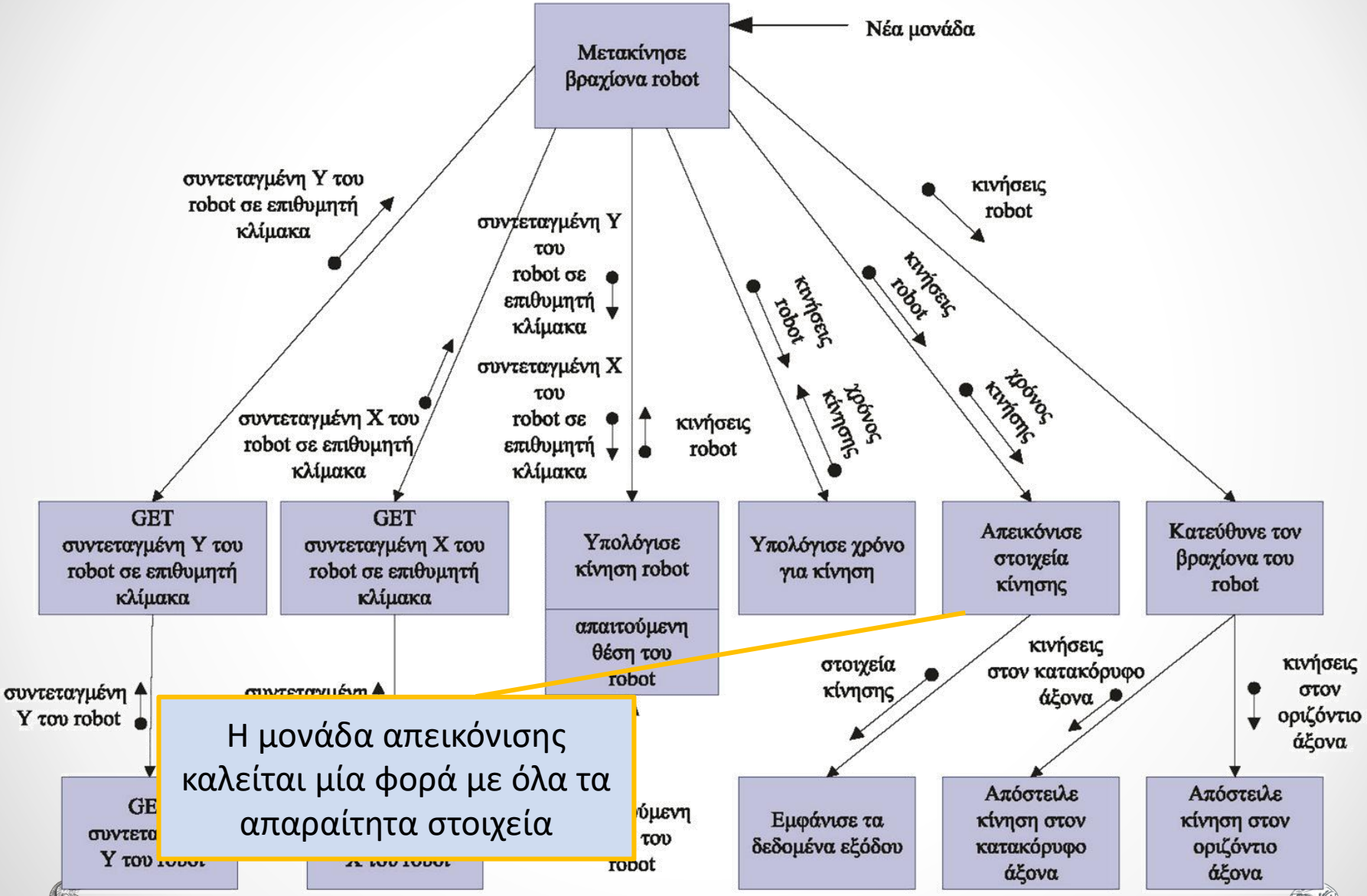
Ανασχεδιασμένο διάγραμμα



Ανασχεδιασμένο διάγραμμα



Ανασχεδιασμένο διάγραμμα



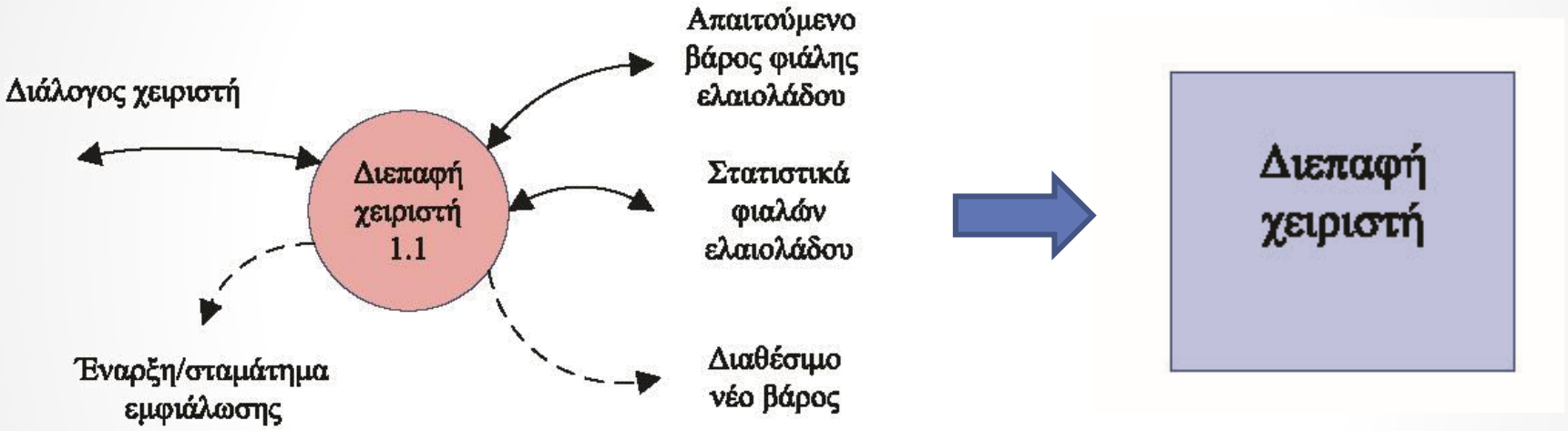
Η μονάδα απεικόνισης καλείται μία φορά με όλα τα απαραίτητα στοιχεία

Μετατροπή της ιεραρχίας εκτελέσιμης μονάδας σε ένα προκαταρκτικό δομικό διάγραμμα

- Μία εκτελέσιμη ενότητα συχνά περιγράφεται από περισσότερα του ενός DFD
  - Π.χ. στο σύστημα εμφιάλωσης ελαιολάδου, η λειτουργικότητα «διασύνδεση χρήστη» μπορεί να αναλύεται σε περισσότερες επί μέρους λειτουργικότητες
- Μέσα στην ιεραρχία των DFD, όλα τα DFD μετατρέπονται σε δομικά διαγράμματα, που συνδέονται σε ένα μοναδικό δομικό διάγραμμα για ολόκληρη την εκτελέσιμη μονάδα
  - Τα δομικά διαγράμματα που παράγονται από τα DFD χαμηλότερου επιπέδου, παρέχουν τη συνολική λειτουργικότητα της εκάστοτε μονάδας
  - Το κάθε δομικό διάγραμμα που παράγεται από ένα DFD χαμηλότερου επιπέδου, μπορεί να αντικαταστήσει την «γονική» του μονάδα που παρήχθη από το DFD του αμέσως υψηλότερου επιπέδου

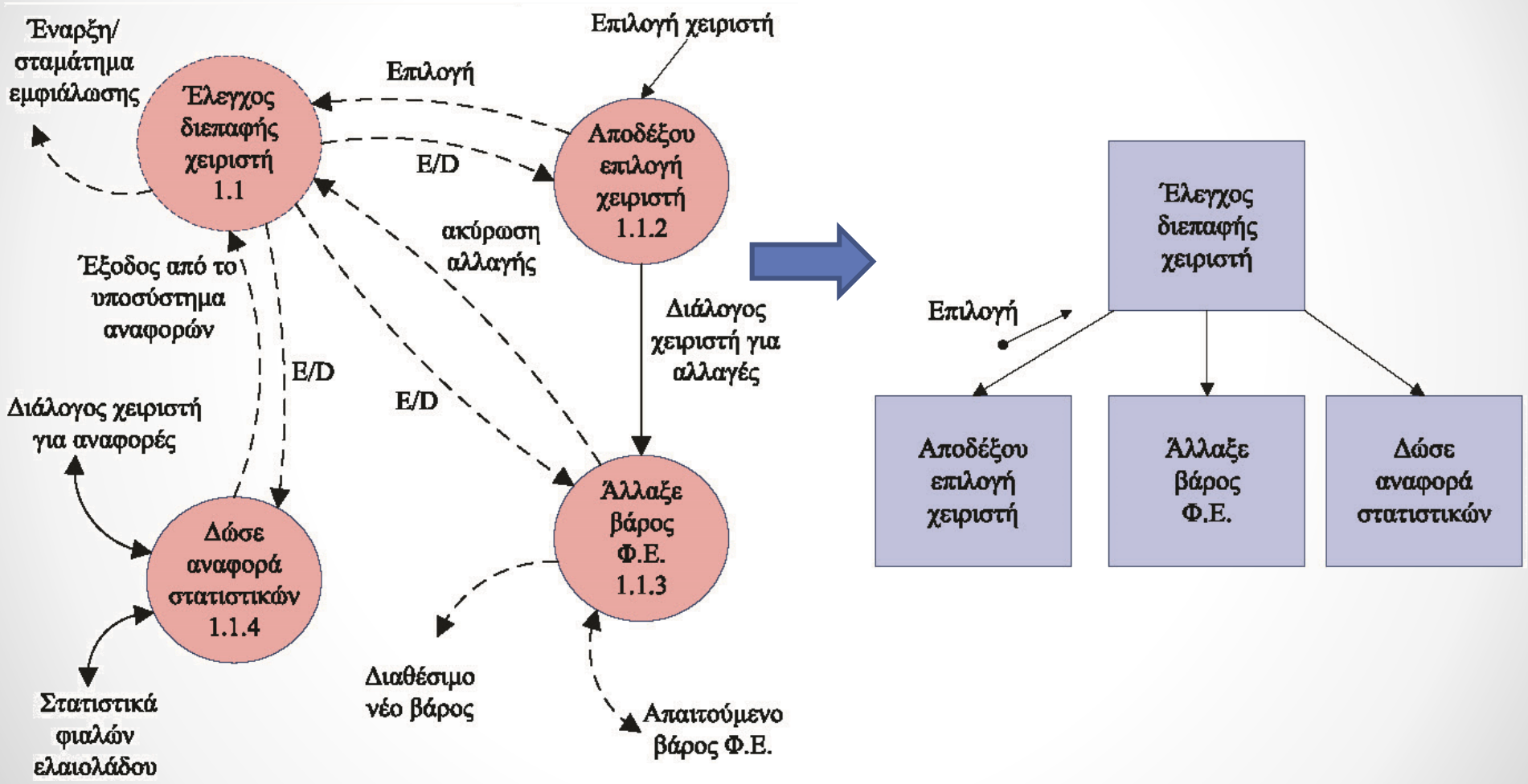
Μετατροπή της ιεραρχίας εκτελέσιμης μονάδας σε ένα προκαταρκτικό δομικό διάγραμμα: παράδειγμα

- 1ο επίπεδο



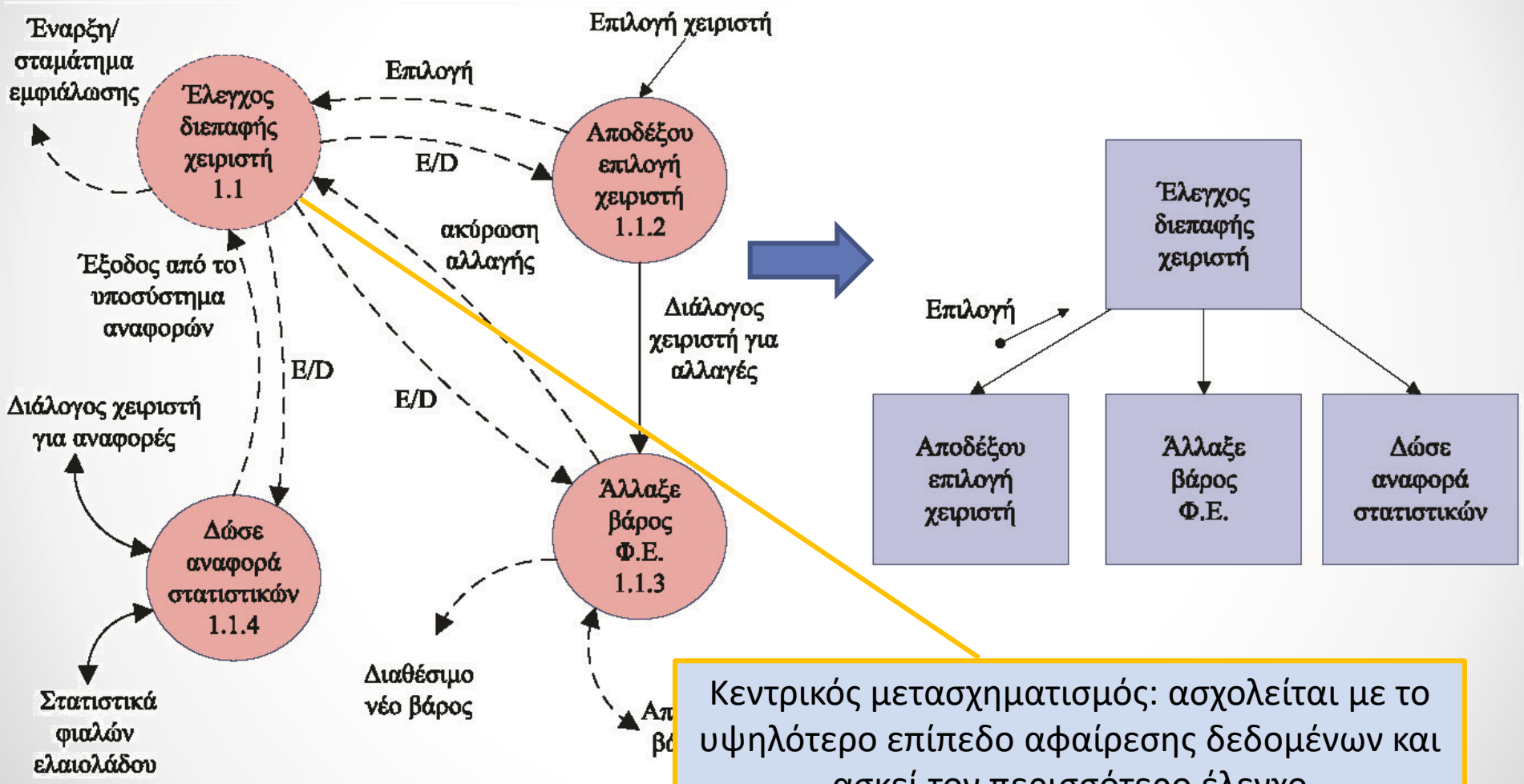
Μετατροπή της ιεραρχίας εκτελέσιμης μονάδας σε ένα προκαταρκτικό δομικό διάγραμμα: παράδειγμα

- 2ο επίπεδο



Μετατροπή της ιεραρχίας εκτελέσιμης μονάδας σε ένα προκαταρκτικό δομικό διάγραμμα: παράδειγμα

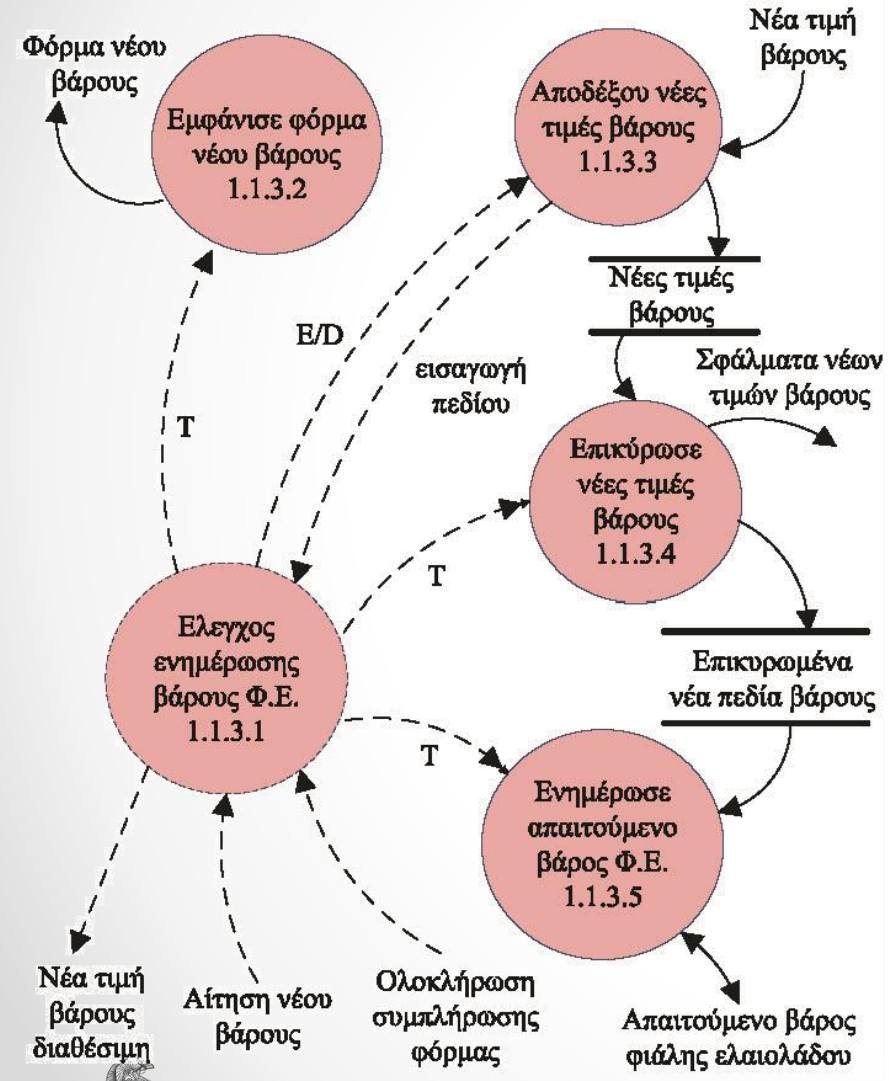
- 2ο επίπεδο

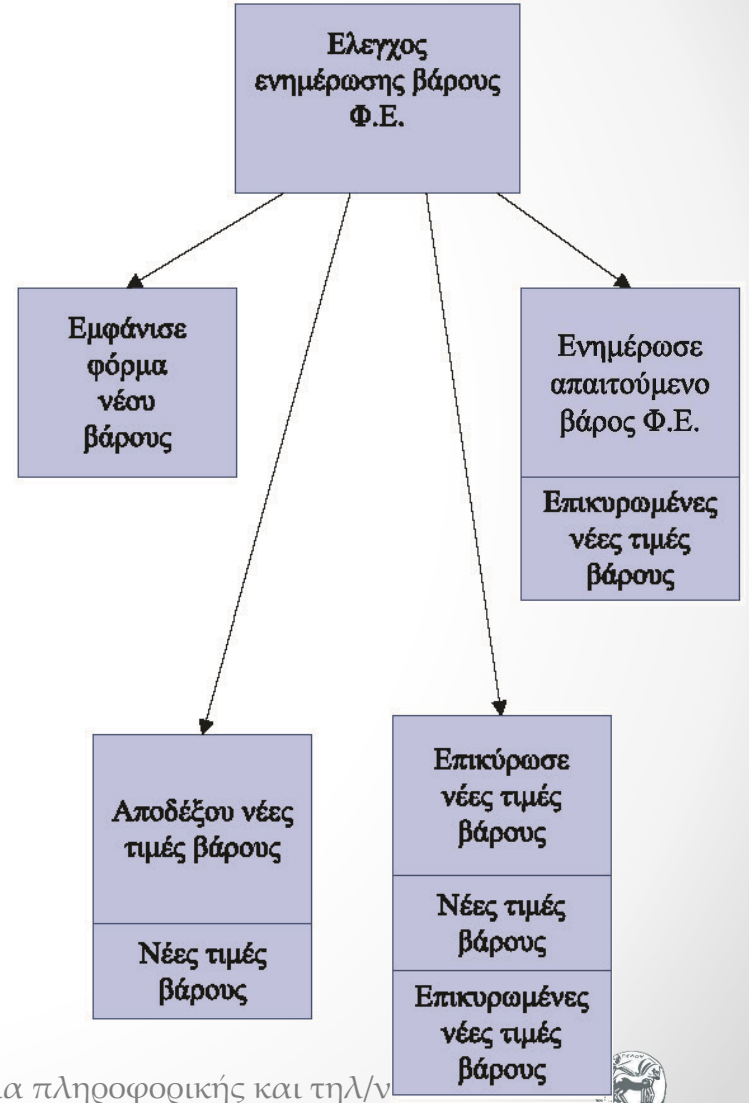


Κεντρικός μετασχηματισμός: ασχολείται με το υψηλότερο επίπεδο αφαίρεσης δεδομένων και ασκεί τον περισσότερο έλεγχο

Μετατροπή της ιεραρχίας εκτελέσιμης μονάδας σε ένα προκαταρκτικό δομικό διάγραμμα: παράδειγμα

- 3ο επίπεδο

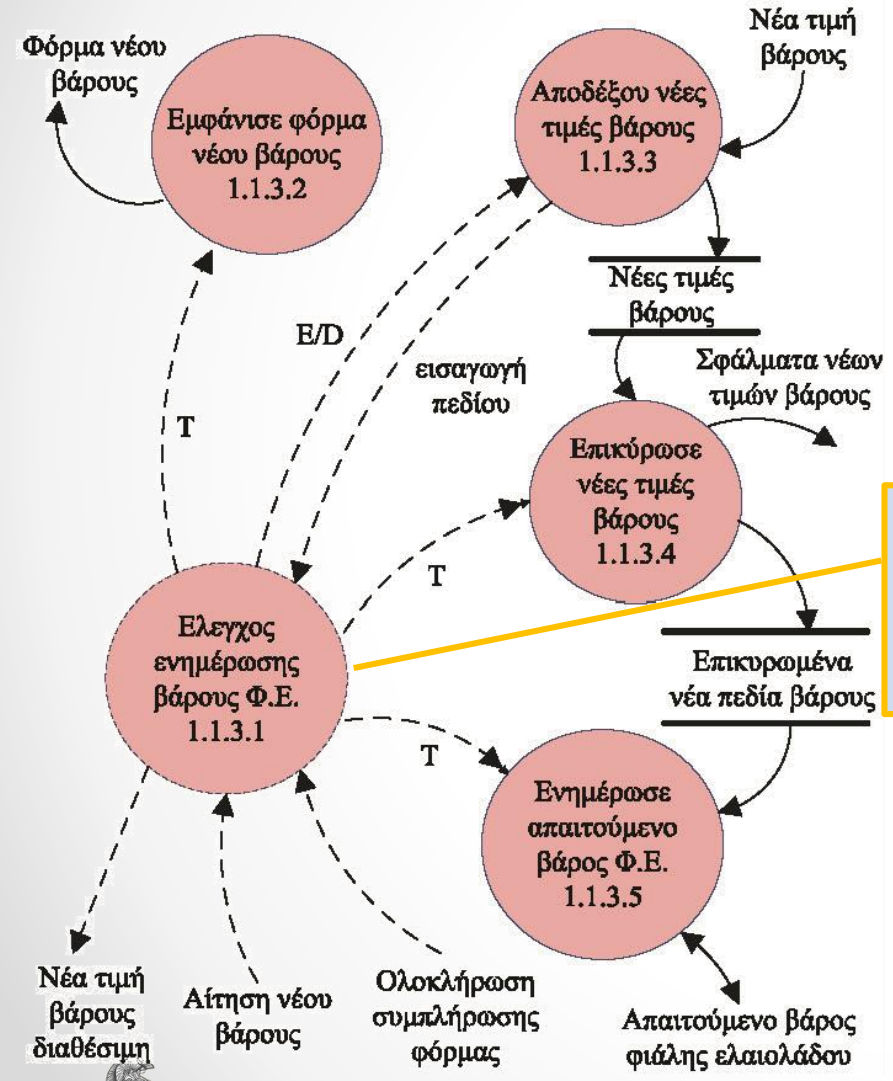


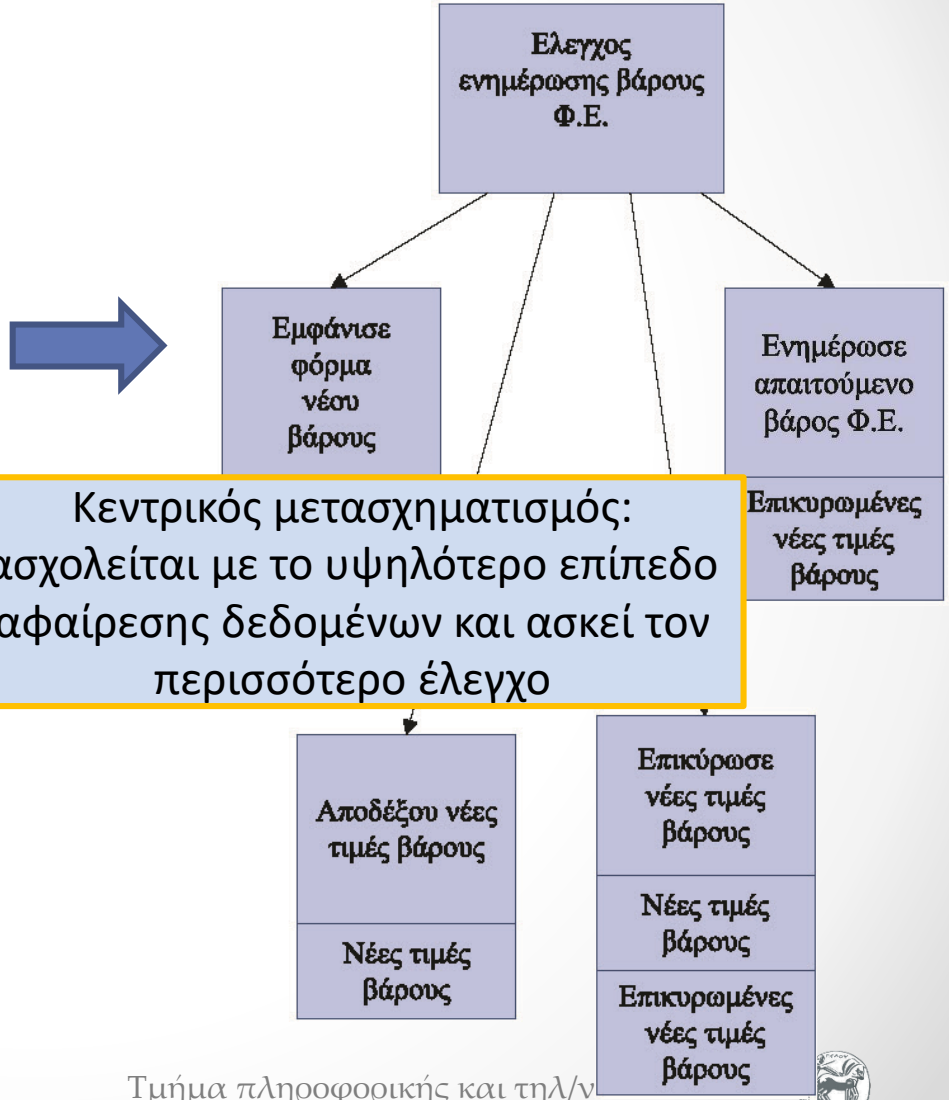


Τμήμα πληροφορικής και τηλ/ν

Μετατροπή της ιεραρχίας εκτελέσιμης μονάδας σε ένα προκαταρκτικό δομικό διάγραμμα: παράδειγμα

- 3ο επίπεδο



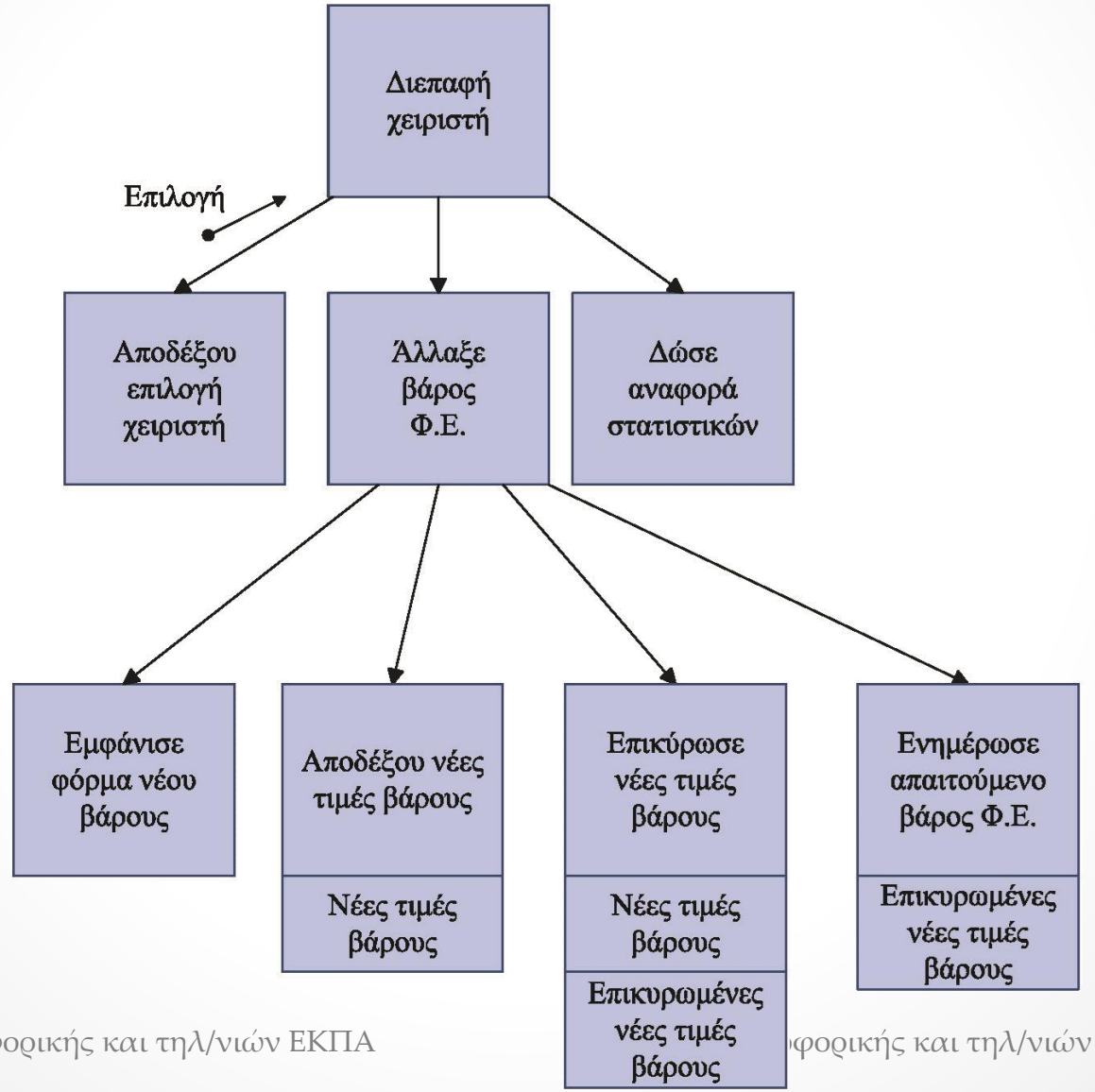


Κεντρικός μετασχηματισμός: ασχολείται με το υψηλότερο επίπεδο αφαίρεσης δεδομένων και ασκεί τον περισσότερο έλεγχο

Τμήμα πληροφορικής και τηλ/ν

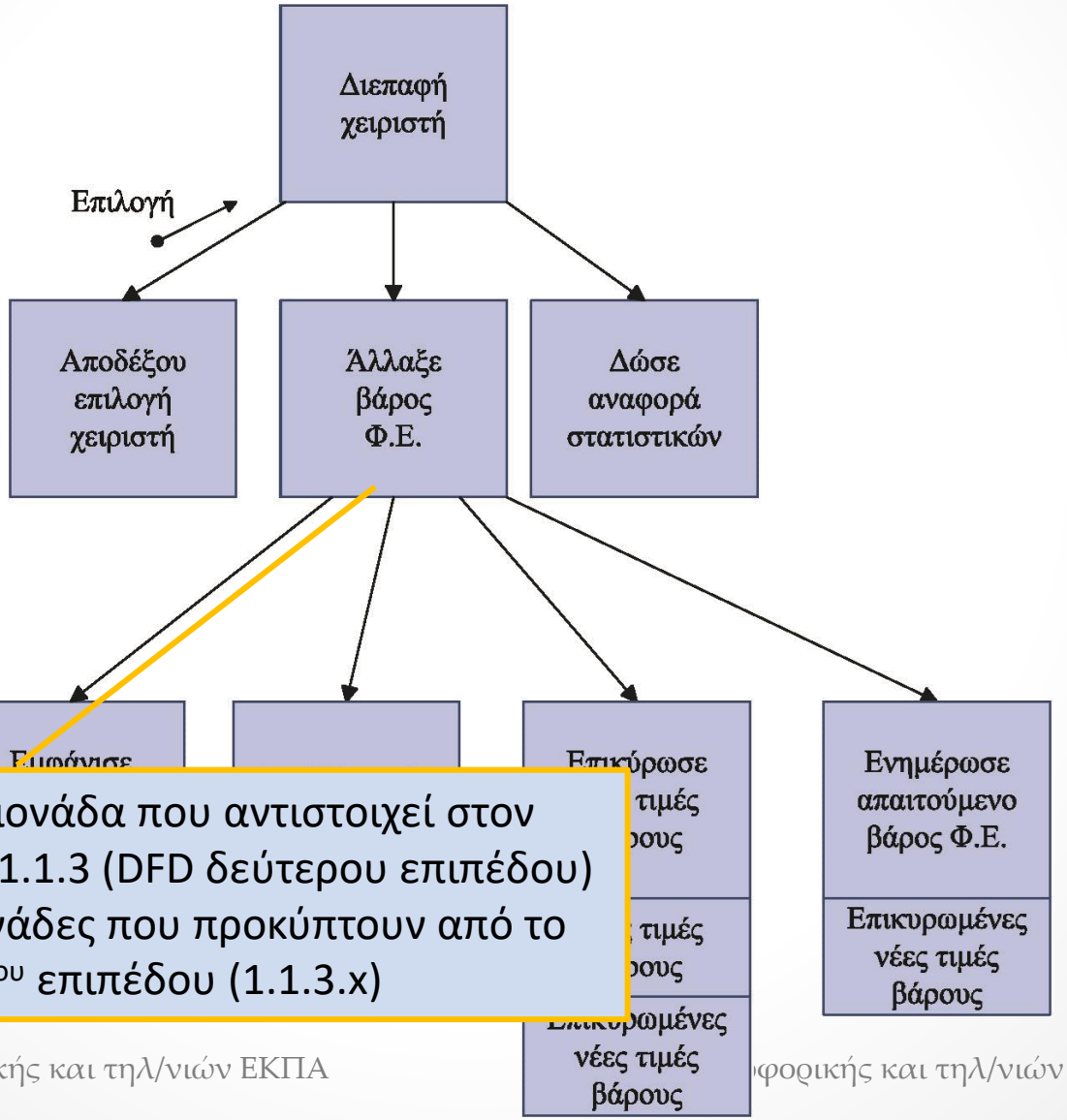
Μετατροπή της ιεραρχίας εκτελέσιμης μονάδας σε ένα προκαταρκτικό δομικό διάγραμμα: παράδειγμα

- Σύνθεση 2ου και 3ου επιπέδου



Μετατροπή της ιεραρχίας εκτελέσιμης μονάδας σε ένα προκαταρκτικό δομικό διάγραμμα: παράδειγμα

- Σύνθεση 2ου και 3ου επιπέδου



Κάτω από τη μονάδα που αντιστοιχεί στον μετασχηματισμό 1.1.3 (DFD δεύτερου επιπέδου) μπαίνουν οι μονάδες που προκύπτουν από το DFD 3ου επιπέδου (1.1.3.x)

Πίνακας ιχνηλάτησης

- Κατά τη διάρκεια των μετασχηματισμών, πρέπει να τηρείται ο πίνακας ιχνηλάτησης (traceability table), προκειμένου να παρέχεται η δυνατότητα ελέγχου υλοποίησης από τον κώδικα των απαιτήσεων του συστήματος
- Για το σύστημα εμφιάλωσης ελαιολάδου, ο πίνακας έχει ως εξής:

	Βασικό μοντέλο	**Μοντέλο οργάνωσης κώδικα**
1.1.1	Έλεγχος διεπαφής χειριστή	Διεπαφή χειριστή
1.1.2	Αποδέξου επιλογή χειριστή	Αποδέξου επιλογή χειριστή
1.1.3.1	Έλεγχος ενημέρωσης βάρους Φ.Ε.	Άλλαξε βάρος Φ.Ε.
1.1.3.2	Εμφάνισε φόρμα νέου βάρους	Εμφάνισε φόρμα νέου βάρους
1.1.3.3	Αποδέξου νέες τιμές βάρους	Αποδέξου νέες τιμέςβάρους
1.1.3.4	Επικύρωσε νέες τιμές βάρους	Επικύρωση νέες τιμές βάρους
1.1.3.5	Ενημέρωσε απαιτούμενο βάρος Φ.Ε.	Ενημέρωσε απαιτούμενο βάρος Φ.Ε.
1.1.4	Δώσε αναφορά στατιστικών	Δώσε αναφορά στατιστικών

Πίνακας ιχνηλάτησης: παρατηρήσεις

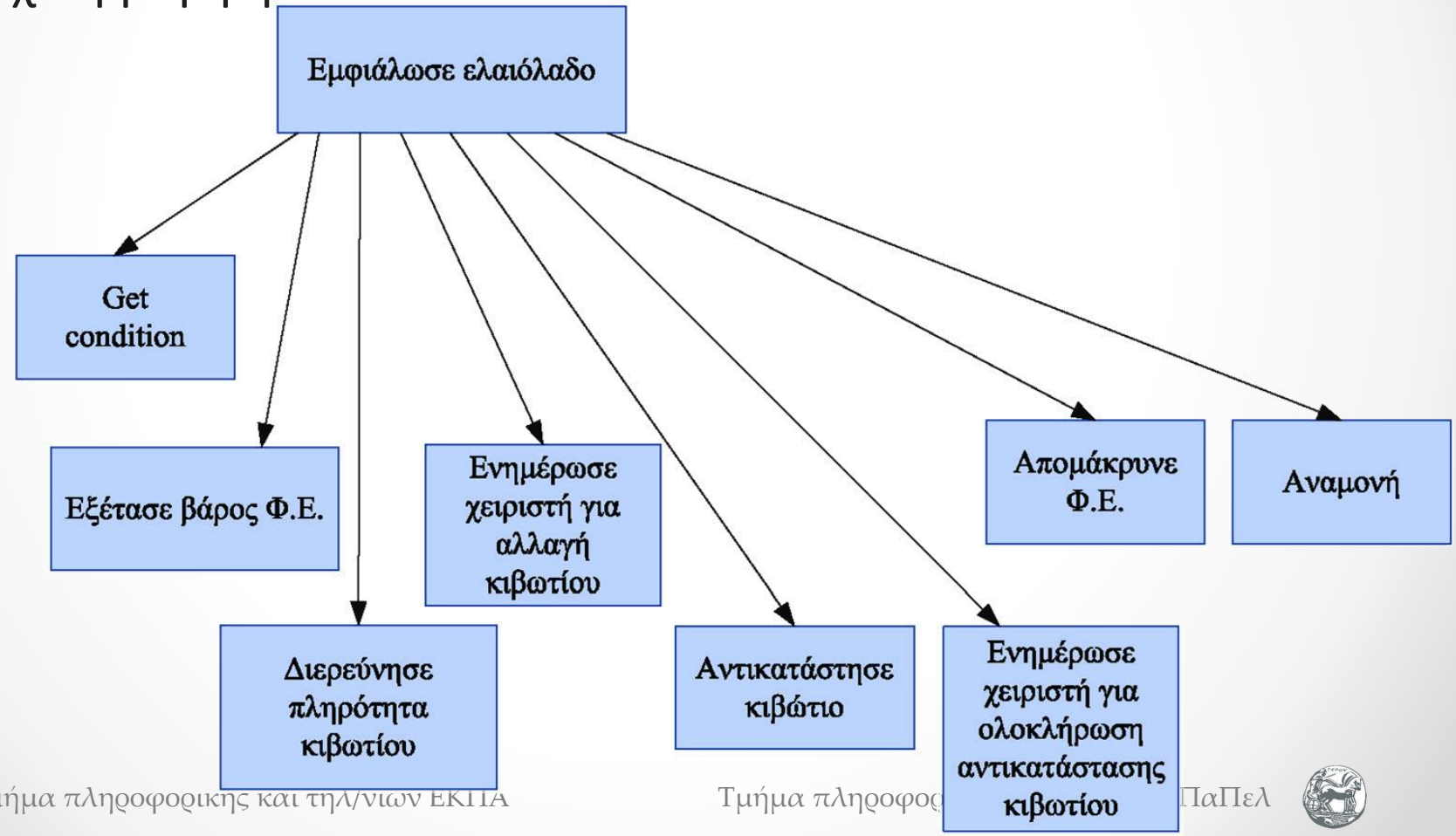
- Οι αυτόνομες μονάδες κώδικα δεν έχουν αριθμούς, καθώς οι υπορουτίνες του κώδικα επίσης δεν είναι αριθμημένες.
- Μόνο οι μετασχηματισμοί δεδομένων στο τέλος της ιεραρχίας του DFD και οι διεργασίες ελέγχου παράγουν αυτόνομες μονάδες κώδικα.
  - Οι μετασχηματισμοί δεδομένων υψηλότερου επιπέδου DFD δεν παράγουν αυτόνομες μονάδες κώδικα, καθώς ομαδοποιούν αυτές του χαμηλότερου επιπέδου, για ευκολία κατανόησης.

Βελτίωση δομικού διαγράμματος

- Τα δομικά διαγράμματα που παράγονται απευθείας από το DFD έχουν γενικά μία καλή δομή, αλλά μπορεί να μην παράγουν πολύ καλά προγράμματα
- Έως τώρα δόθηκε έμφαση κυρίως στη λειτουργικότητα, ωστόσο οι παραγόμενες μονάδες:
  - Μπορεί να είναι πολύ μεγάλες ή πολύ μικρές ή πολύ σύνθετες
  - Μπορεί να εκτελούν πανομοιότυπες ή παραπλήσιες εργασίες που θα πρέπει να συγχωνευθούν
- Συνακόλουθα το δομικό διάγραμμα χρήζει βελτίωσης
  - Αυτό γίνεται μέσω (i) *ανάλυσης του διαγράμματος* και (ii) *διάσπαση* και *συγχώνευση* των αυτόνομων μονάδων
  - Καθώς το βήμα αυτό μπορεί να καταστρέψει την καλή αρχική δομή, ακολουθείται από ένα βήμα ελέγχου *συνεκτικότητας* και *σύζευξης* των μονάδων και *έλεγχο ισορροπίας του διαγράμματος*

Βελτίωση δομικού διαγράμματος: παράδειγμα

- Θα βελτιώσουμε το προκαταρκτικό δομικό διάγραμμα της εκτελέσιμης μονάδας της εμφιάλωσης ελαιολάδου
- Αρχική μορφή:



Βελτίωση δομικού διαγράμματος: παράδειγμα

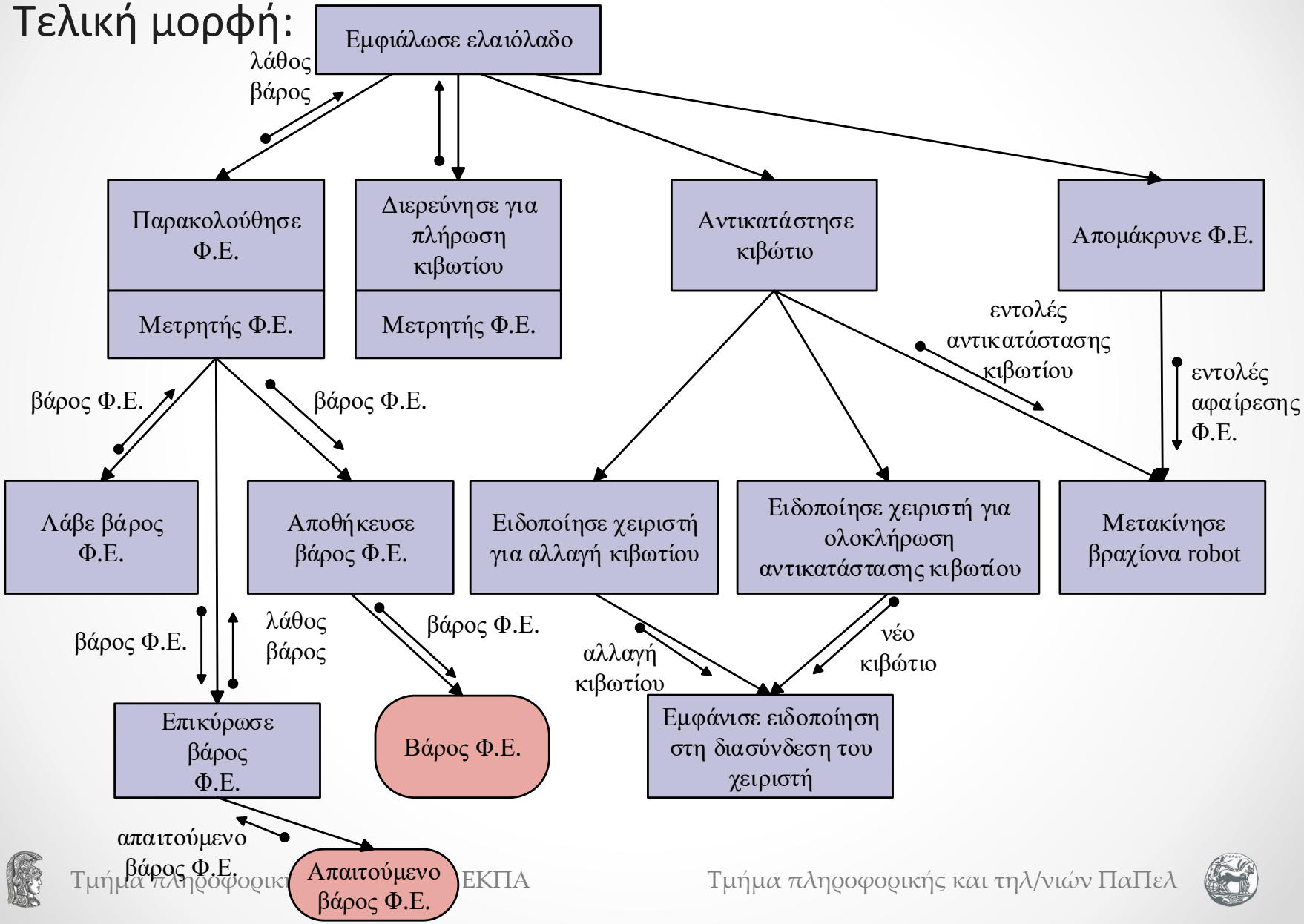
- Αρχικά, αναλύουμε εκείνες τις αυτόνομες μονάδες κώδικα που είναι είτε πολύ μεγάλες, είτε πολύπλοκες, ή μοιράζονται περιοχές λειτουργικότητας
  - η μονάδα «Εξέτασε βάρος Φ. Ε.» τροποποιείται σε «Παρακολούθησε Φ. Ε.», η οποία καλεί τις ενότητες «Λάβε βάρος Φ.Ε.», «Επικύρωσε βάρος Φ.Ε.» και «Αποθήκευσε βάρος Φ.Ε.» *(ανάλυση μιας σύνθετης λειτουργίας σε μικρότερες)*
  - η κοινή εργασία της διεπαφής του χειριστή αφαιρείται από τις «Ενημέρωσε χειριστή για αλλαγή κιβωτίου» και «Ενημέρωσε χειριστή για ολοκλήρωσε αντικατάστασης κιβωτίου» και έτσι οδηγούμαστε σε δύο υψηλότερου επιπέδου ενότητες, τις «Ειδοποίησε χειριστή για αλλαγή κιβωτίου» και «Ειδοποίησε χειριστή για ολοκλήρωση αντικατάστασης κιβωτίου», που και οι δύο καλούν τη χαμηλότερου επιπέδου ενότητα «Εμφάνισε ειδοποίηση στη διασύνδεση του χρήστη» *(αποφυγή επανάληψης κοινής λειτουργικότητας)*
  - η κοινή εργασία της μετακίνησης του ρομποτικού βραχίονα αφαιρείται από τις «Απομάκρυνε Φ.Ε.» και «Αντικατάστησε κιβώτιο», που τώρα καλούν και οι δύο τη χαμηλότερου επιπέδου μονάδα «Μετακίνησε βραχίονα robot». *(αποφυγή επανάληψης κοινής λειτουργικότητας)*

Βελτίωση δομικού διαγράμματος: παράδειγμα

- Οι μονάδες έχουν καλή σύζευξη και συνεκτικότητα
- Ωστόσο η μονάδα «Εμφιάλωσε ελαιόλαδο» έχει αυξημένες αρμοδιότητες. Αυτό το αντιμετωπίζουμε με αναδόμηση γύρω από τις «Αντικατάστησε κιβώτιο», «Ενημέρωσε χειριστή για αλλαγή κιβωτίου» και «Ενημέρωσε χειριστή για ολοκλήρωση αντικατάστασης κιβωτίου» *(μεταφορά αρμοδιοτήτων κλήσης σε χαμηλότερο επίπεδο)*
- Η «Get condition» απορροφάται από την «Εμφιάλωσε ελαιόλαδο» *(εξάλειψη υπερβολικά μικρών μονάδων)*
- Η μονάδα «Αναμονή» απορροφάται από την «Εμφιάλωσε ελαιόλαδο» *(αφαίρεση στοιχείων που δεν προωθούν την κατανόηση του διαγράμματος και εξάλειψη υπερβολικά μικρών μονάδων)*

Βελτίωση δομικού διαγράμματος: παράδειγμα

- Τελική μορφή:



Βελτίωση δομικού διαγράμματος: παράδειγμα

- Το δομικό διάγραμμα περιέχει ένα διαχωρισμό αποφάσεων. Η μονάδα «Παρακολούθησε Φ.Ε.» αποφασίζει ότι η φιάλη ελαιολάδου είναι με «Λάθος Βάρος», όμως τίποτα δεν γίνεται έως ότου η ένδειξη «Λάθος Βάρος» φθάσει στη μονάδα «Εμφιάλωσε ελαιόλαδο».
  - Ο διαχωρισμός της απόφασης από την ενέργεια μπορεί να αρθεί με τη κλήση της μονάδας «Επικύρωσε βάρος Φ.Ε.» από την «Εμφιάλωσε ελαιόλαδο» και όχι από την «Παρακολούθησε Φ.Ε.», κάτι που θα αύξανε πολύ τις αρμοδιότητες της μονάδας «Εμφιάλωσε ελαιόλαδο».
- Είναι συνήθως αδύνατο να παράγουμε ένα δομικό διάγραμμα που να είναι απόλυτα τέλειο σε όλη του τη δομή, καθώς μερικές φορές η επίλυση μιας πλευράς του προβλήματος θα προκαλέσει προβλήματα σε άλλες πλευρές της δομής.
  - Σε τέτοιες περιπτώσεις, θα πρέπει να χρησιμοποιήσουμε τη λύση που κάνει πιο κατανοητό το δομικό διάγραμμα.

Αντικειμενοστρεφής σχεδιασμός

Σχεδιασμός στο αντικειμενοστρεφές μοντέλο

- Ο σχεδιασμός στο αντικειμενοστρεφές μοντέλο αφορά στον προσδιορισμό και τη βελτιστοποίηση των συστατικών του λογισμικού, έτσι ώστε να καλύπτουν τις απαιτήσεις που προκύπτουν από την ανάλυση
- Μία ευρέως διαδεδομένη διαδικασία που καλύπτει τη διαδικασία σχεδιασμού συστημάτων πραγματικού χρόνου (αλλά και τον πλήρη κύκλο ανάπτυξής τους) είναι η διαδικασία ROPES
- Στη διαδικασία ROPES η φάση του σχεδιασμού χωρίζεται σε τρεις μείζονες υποφάσεις:
  - *αρχιτεκτονικός σχεδιασμός*
  - *μηχανιστικός σχεδιασμός*
  - *λεπτομερής σχεδιασμό*

Αρχιτεκτονικός σχεδιασμός

- Ο αρχιτεκτονικός σχεδιασμός ορίζει τον τρόπο με τον οποίο τα αντικείμενα που έχουν προσδιοριστεί στη φάση της ανάλυσης οργανώνονται σε δομές ευρύτερης κλίμακας
- Στον αρχιτεκτονικό σχεδιασμό ορίζουμε θέματα που αφορούν είτε ολόκληρο το εύρος του συστήματος, είτε θεωρούνται σε επίπεδο επεξεργαστή και αφορούν στα ακόλουθα:
  - υποσυστήματα και συστατικά,
  - ταυτοχρονισμός και διαχείριση πόρων,
  - κατανομή σε πολλαπλούς χώρους διευθύνσεων,
  - διαχείριση ασφάλειας και αξιοπιστίας και
  - θέση σε λειτουργία (deployment) του λογισμικού σε στοιχεία υλικού

Τύποι αρχιτεκτονικών

- Στη διαδικασία ROPES θεωρούνται δύο τύποι αρχιτεκτονικών:
  1. *λογική αρχιτεκτονική*
     - αναφέρεται στην οργάνωση των στοιχείων που θεωρείται μόνο κατά τη φάση του σχεδιασμού
     - αφορά τις κλάσεις και τους τύπους δεδομένων
     - γίνεται κατά τέτοιο τρόπο ώστε να επιτυγχάνονται καλύτερα χαρακτηριστικά στον κώδικα, όπως π.χ. υψηλή συνεκτικότητα, χαλαρή σύζευξη, αποφυγή υπερφόρτωσης αντικειμένων με αρμοδιότητες κ.τ.λ
     - προτείνεται ο διαχωρισμός των αντικειμένων του λογισμικού σε *πεδία* (domains), τα οποία ορίζονται ανάλογα με την εφαρμογή
       - Π.χ. στο μοντέλο του ανελκυστήρα που παρουσιάστηκε στο κεφάλαιο 2 θα μπορούσαμε να θεωρήσουμε τα πεδία (i) των στοιχείων υλικού, (ii) τα στοιχεία χειρισμού του ανελκυστήρα από τους χρήστες και (iii) τα στοιχεία ελέγχου του ανελκυστήρα

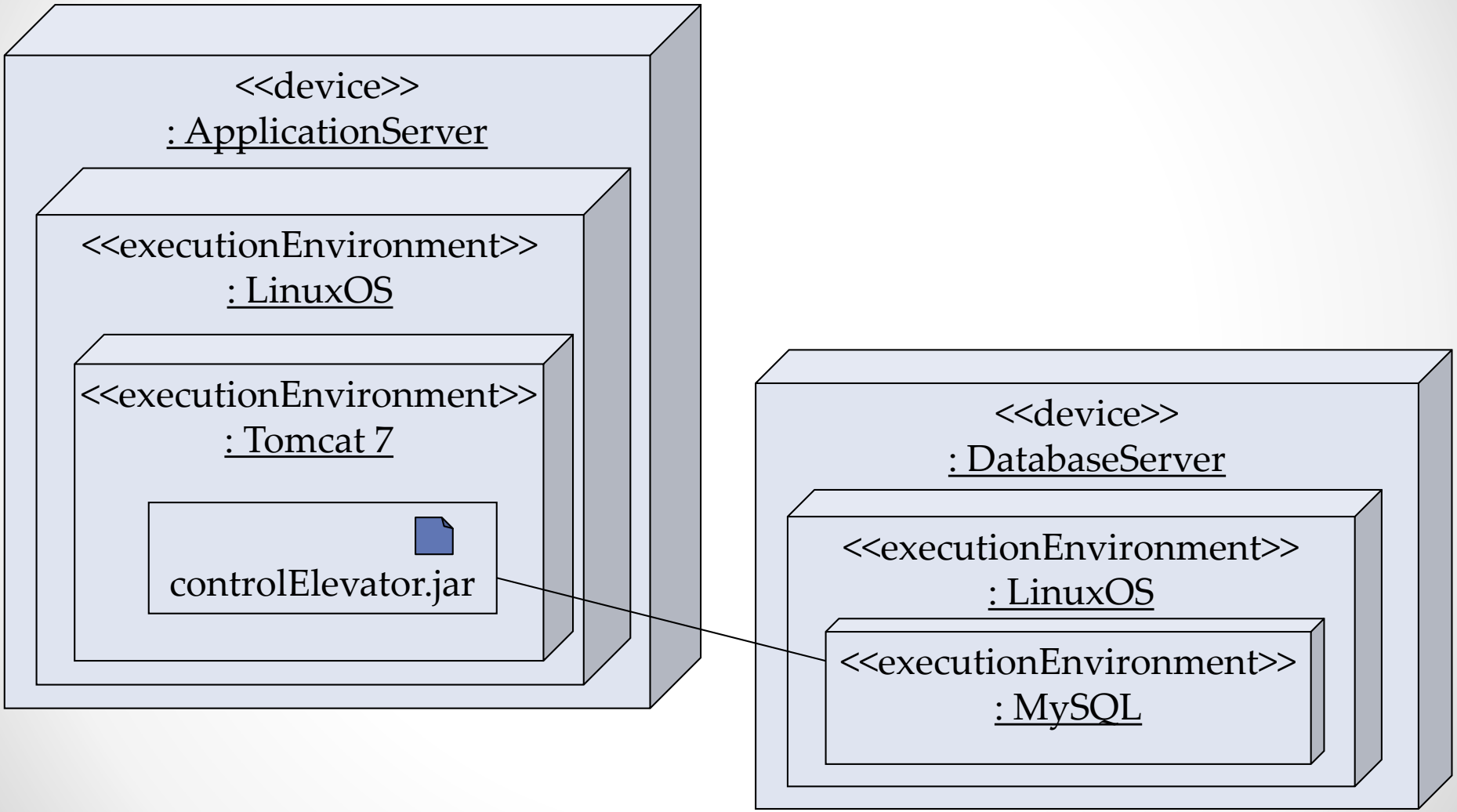
Τύποι αρχιτεκτονικών (2)

- Στη διαδικασία ROPES θεωρούνται δύο τύποι αρχιτεκτονικών:
  2. *φυσική αρχιτεκτονική*
     - απεικονίζει το πώς τα στοιχεία λογισμικού αντιστοιχίζονται με φυσικές δομές, π.χ. εκτέλεση σε φυσικούς επεξεργαστές, αποθήκευση σε συγκεκριμένα φυσικά μέσα, παροχή ηλεκτρικού ρεύματος
     - Συνηθισμένες έννοιες στη φυσική αρχιτεκτονική είναι τα *υποσυστήματα*, τα *στοιχεία* (components), οι *διεργασίες* (tasks ή processes) και τα *αντικείμενα* (objects).
     - Σε ό,τι αφορά τα αντικείμενα, η φυσική αρχιτεκτονική ορίζει πώς θα δημιουργηθούν ή αποθηκευθούν τα στιγμιότυπα των διάφορων κλάσεων
     - Η φυσική αρχιτεκτονική μπορεί να μην ταιριάζει απόλυτα στη λογική αρχιτεκτονική: π.χ. στο παράδειγμα του ανελκυστήρα το υποσύστημα της παροχής ηλεκτρικού ρεύματος μπορεί να περιλαμβάνει κλάσεις που εννοιολογικά περιλαμβάνονται στα λογικά πεδία του υλικού, των στοιχείων χειρισμού και των στοιχείων ελέγχου του ανελκυστήρα

Απεικόνιση λογικής αρχιτεκτονικής σε φυσική

- Για να αποτυπώσουμε το πώς απεικονίζονται τα στοιχεία του λογισμικού σε φυσικές συσκευές, χρησιμοποιούμε τα διαγράμματα *θέσης σε λειτουργία* (ή *παράταξης*) (deployment diagrams) της UML
  - Π.χ. ο υπολογιστής 1 διαθέτει ένα περιβάλλον εκτέλεσης “Linux” το οποίο φιλοξενεί έναν εξυπηρέτη εφαρμογών (application server) «Tomcat» και εντός αυτού εκτελείται το “controlElevator.war”. Το “controlElevator.war” επικοινωνεί με ένα στοιχείο λογισμικού «MySQL» που εκτελείται στον υπολογιστή 2, ο οποίος διαθέτει λειτουργικό σύστημα Linux.

Απεικόνιση λογικής αρχιτεκτονικής σε φυσική



Αρχιτεκτονική λογισμικού και ποιοτικά χαρακτηριστικά

- Τα ποιοτικά χαρακτηριστικά του λογισμικού, είναι σαφές ότι δεν είναι ανεξάρτητα μεταξύ τους, π.χ.:
  - αν επιθυμούμε καλύτερη απόδοση, πιθανώς να πρέπει να έχουμε μικρότερη ευελιξία, διαλειτουργικότητα και μεταφερσιμότητα
  - αν θέλουμε μεγαλύτερη διαλειτουργικότητα, ίσως να υποβαθμιστεί η απόδοση, αλλά θα έχουμε βελτίωση σε ό,τι αφορά στην ευελιξία και τη μεταφερσιμότητα
- Πρέπει να αποφασίσουμε ποια ποιοτικά χαρακτηριστικά είναι πιο σημαντικά και ποιο είναι το κατάλληλο μείγμα για το σύστημά μας.

Αρχιτεκτονικά πρότυπα

- Είναι τυποποιημένες μορφές διάρθρωσης των συστατικών του λογισμικού που εφαρμόζονται στη φάση του αρχιτεκτονικού σχεδιασμού
- Συνηθισμένα αρχιτεκτονικά πρότυπα:
  - *διαστρωματωμένη αρχιτεκτονική*. Κάποιες μονάδες ασχολούνται με λειτουργίες υψηλού επιπέδου ενώ κάποιες άλλες ασχολούνται με λειτουργίες χαμηλού επιπέδου. Οι μονάδες διαχωρίζονται σε *στρώματα* ανάλογα με το επίπεδό τους. Τυπικές περιπτώσεις: π.χ. δύο στρώματα υπάρχουν → μοντέλο *εξυπηρέτη-εξυπηρετούμενου*, τρία στρώματα (*παρουσίασης*, *πεδίου* και *πηγών δεδομένων)* → τριεπίπεδη (three-tier) αρχιτεκτονική..
  - *Μοντέλο-Όψη-Ελεγκτής (model-view-controller – MVC)* με τρεις συνιστώσες:
    1. το *μοντέλο* περιέχει τη βασική λειτουργικότητα και αντιστοιχεί στα αντικείμενα της λογικής του πεδίου
    2. η *όψη* αναλαμβάνει την παρουσίαση που είναι αντικείμενο της διεπαφής χρήστη (π.χ. παράθυρο ή σελίδα HTML) και
    3. *ο ελεγκτής* λαμβάνει τις επιλογές του χρήστη και ενεργοποιεί τη συμπεριφορά του μοντέλου.
    - Το πρότυπο *Μοντέλο-Όψη-Ελεγκτής* επιτυγχάνει τον διαχωρισμό της αρμοδιότητας της παρουσίασης της πληροφορίας από την ίδια την πληροφορία καθώς επίσης και την ανάθεση της μετάφρασης των ενεργειών και των εντολών του τελικού χρήστη σε ένα εξειδικευμένο αντικείμενο, που είναι ο ελεγκτής.

Μηχανιστικός σχεδιασμός

- Ο μηχανιστικός σχεδιασμός αφορά στην προσθήκη και την οργάνωση κλάσεων για να βελτιστοποιηθεί κάποια συνεργασία
  - δηλαδή η αλληλεπίδραση μεταξύ των κλάσεων ώστε να επιτευχθεί ένας συγκεκριμένος στόχος
- Στα πλαίσια του μηχανιστικού σχεδιασμού επιθεωρείται το μοντέλο που προκύπτει από την ανάλυση και προστίθενται ή αναθεωρούνται κλάσεις έτσι ώστε να βελτιωθεί η ποιότητα του κώδικα
- Η βελτίωση επιτυγχάνεται αναγνωρίζοντας ορισμένες τυποποιημένες περιπτώσεις συνεργασίας που υπάρχουν στο λογισμικό και εφαρμόζοντας *μοτίβα σχεδιασμού* (design patterns), τα οποία είναι γενικές λύσεις που μπορούν να επαναχρησιμοποιηθούν στα πλαίσια της λύσης κάποιου συχνά εμφανιζόμενου προβλήματος

Συνήθη σχεδιαστικά μοτίβα

Όνομα μοτίβου	Σκοπός
Παρατηρητής (observer)	Επιτρέπει σε πολλαπλούς εξυπηρετούμενους να μοιράζονται αποτελεσματικά έναν εξυπηρέτη και να ενημερώνονται αυτόνομα
Αντιπρόσωπος (proxy)	Υλοποιεί ένα μοτίβο παρατηρητή σε ένα κατανεμημένο περιβάλλον
Αξιόπιστη δοσοληψία (reliable transaction)	Ελέγχει την επικοινωνία μεταξύ αντικειμένων, υλοποιώντας διάφορα επίπεδα αξιοπιστίας
Έξυπνος δείκτης (smart pointer)	Βοηθάει στο να αποφεύγονται προβλήματα που προκαλούνται από τους συνήθεις δείκτες
Προφυλαγμένη κλήση (guarded call)	Παρέχει εύρωστη και εντός χρονικών ορίων πρόσβαση σε υπηρεσίες και δεδομένα
Περιέκτης (container)	Αποκρύπτει μέσω αφαίρεσης έννοιες που αφορούν τη δομή των δεδομένων από τις κλάσεις της εφαρμογής για να απλοποιηθεί το μοντέλο και να διευκολυνθεί η επαναχρησιμοποίηση
Διεπαφή (interface)	Αποκρύπτει μέσω αφαίρεσης τους τύπους των δεδομένων από την υλοποίηση, για να υποστηριχθούν πολλαπλές υλοποιήσεις και πολλαπλοί τύποι με κοινή εσωτερική δομή
Ραντεβού (rendezvous)	Παρέχει έναν ευέλικτο μηχανισμό για χαμηλής επιβάρυνσης επικοινωνία μεταξύ διεργασιών

Λεπτομερής σχεδιασμός

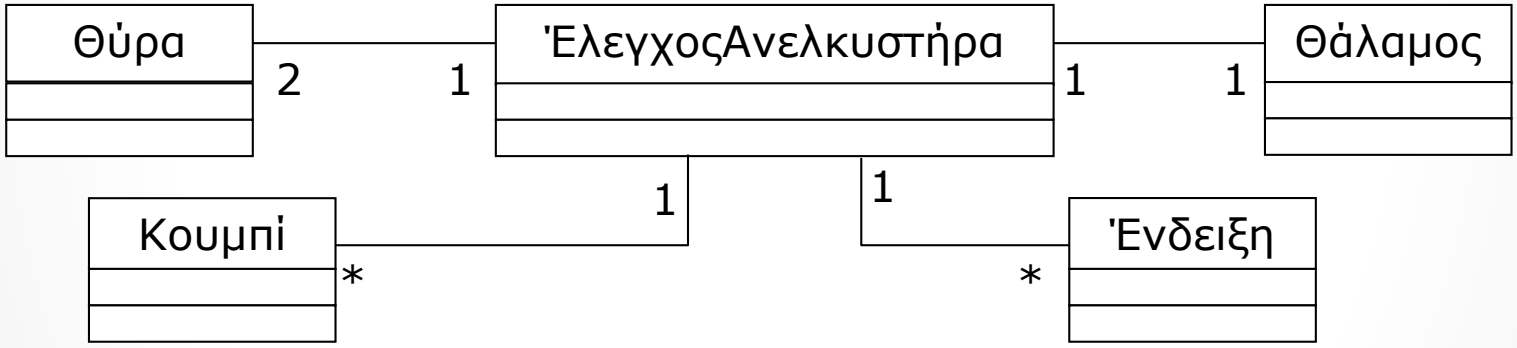
- Ο λεπτομερής σχεδιασμός ασχολείται με την εσωτερική δομή των μονάδων, τις δομές δεδομένων και τους αλγορίθμους που αυτές χρησιμοποιούν
  - η λεπτομερής αναπαράσταση των δεδομένων με δομές (π.χ. λίστες έναντι πινάκων, αραιοί πίνακες έναντι κανονικών)
  - οι συσχετίσεις μεταξύ κλάσεων – εξετάζεται η πληθικότητα του κάθε άκρου και αποφασίζεται το πώς θα αναπαρασταθούν οι αναφορές.
    - Όταν τα αντικείμενα που πρέπει να συσχετισθούν βρίσκονται σε διαφορετικά μηχανήματα, τότε πρέπει να χρησιμοποιηθούν κατάλληλες τεχνικές για κατανεμημένα αντικείμενα όπως π.χ. RMI
    - Για τις συσχετίσεις καθορίζουμε επίσης και την *πλοηγησιμότητα*
  - η *ορατότητα* δηλ. αν οι λειτουργίες που υλοποιεί η κάθε μονάδα θα είναι ορατές και σε ποιους
  - η περιγραφή της επεξεργασίας των αλγορίθμων (π.χ. επιλογή αλγόριθμου ταξινόμησης ή κρυπτογράφησης)
  - οι *εξαιρέσεις* (exceptions), δηλ. οι «αναφορές σφάλματος» που παράγονται από τις διάφορες μονάδες και τις λειτουργίες που αυτές υλοποιούν

Ορισμός και βελτιστοποίηση αντικειμένων λογισμικού

- Η αντικειμενοστρεφής ανάλυση μπορεί να δώσει ένα μοντέλο αντικειμένων που δεν είναι καλό ποιοτικά. Τα πιο συνηθισμένα προβλήματα είναι τα ακόλουθα:
  - *υπερφορτώνεται το κεντρικό αντικείμενο*, το οποίο πρακτικά αναλαμβάνει όλες τις αρμοδιότητες να δέχεται όλες τις πληροφορίες και να εκτελεί όλη την επεξεργασία
  - *ύπαρξη άεργων αντικείμενων*. Σε αντιδιαστολή με το κεντρικό αντικείμενο, άλλα αντικείμενα έχουν πολύ ελαττωμένο ρόλο στο σύστημα, π.χ. αντικείμενα ενδείξεων, τα οποία απλώς δέχονται εντολές
  - *ανταγωνισμός για υπολογιστικούς πόρους*. Όσα αντικείμενα ζητάνε από το κεντρικό αντικείμενο να «ελεγχθούν» (ή στέλνουν πληροφορίες σ' αυτό), πρακτικά ζητάνε να χρησιμοποιήσουν υπολογιστικούς πόρους του αντικειμένου αυτού. Έτσι, υπάρχει η πιθανότητα κάποια αιτήματα να μην διεκπεραιωθούν εντός των χρονικών ορίων που πρέπει, κάτι που μπορεί με τη σειρά του να οδηγήσει σε σοβαρά προβλήματα.
  - *χαμηλή αποτελεσματικότητα του όλου συστήματος*. Ακόμη κι αν οι υπολογιστικοί πόροι του κεντρικού αντικειμένου είναι απεριόριστοι, η συγκεκριμένη προσέγγιση υποχρησιμοποιεί τους λοιπούς πόρους (αυτούς των άλλων αντικειμένων) και είναι ακατάλληλος για κατανεμημένα συστήματα.

Αντιμετώπιση ζητημάτων

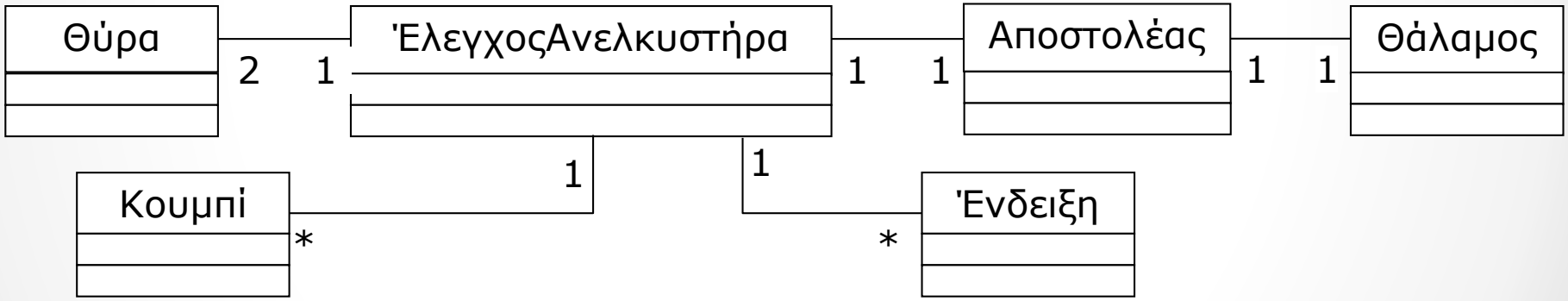
- Μία προσέγγιση είναι να εισαχθούν *αντικείμενα ελέγχου*, τα οποία θα αναλάβουν τον έλεγχο των επιμέρους στοιχείων
  - Το κεντρικό αντικείμενο διατηρεί μόνο συντονιστικό ρόλο στην όλη λειτουργία
- Π.χ.: στην αρχική δομή συστήματος ανελκυστήρα (απόσπασμα)



εισάγεται το στοιχείο *Αποστολέας* (επόμενη διαφάνεια)

Αντιμετώπιση ζητημάτων

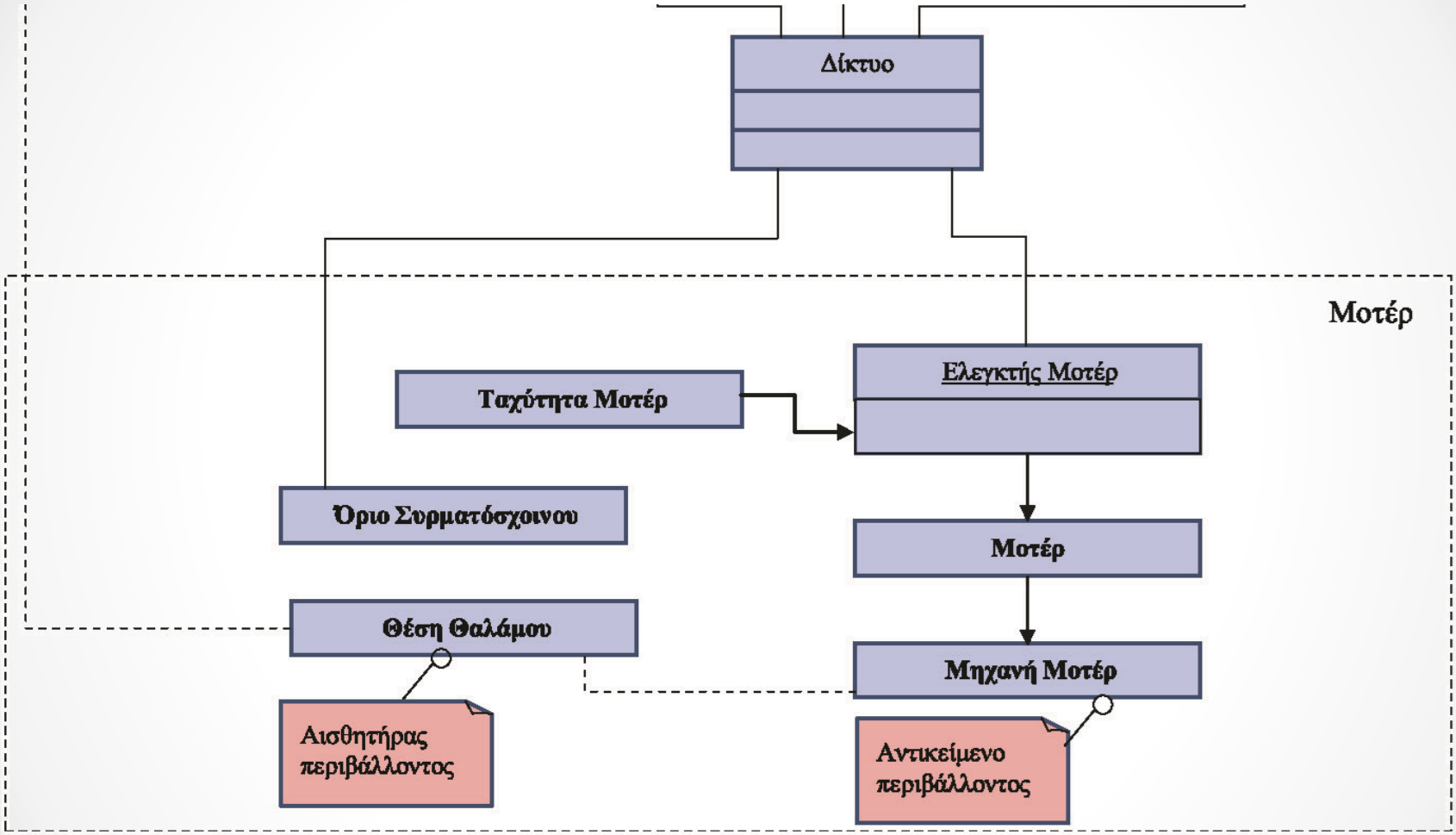
- Ο αποστολέας μεριμνά για να μεταβαίνει ο θάλαμος στομ σωστό όροφο και να ανοιγοκλείνουν οι θύρες, αλληλεπιδρώντας με τα αντίστοιχα αντικείμενα (αφαιρώντας αυτές τις αρμοδιότητες από τον υπερφορτωμένο «ΈλεγχοΑνελκυστήρα»)



Πρόσθετα ζητήματα και αντιμετώπισή τους

- Πρόσθετα ζητήματα που υφίστανται:
  - πώς ελέγχονται τα αντικείμενα του φυσικού περιβάλλοντος, όπως π.χ. να αντιληφθούμε που βρίσκεται ο θάλαμος του ανελκυστήρα;
  - πώς το κάθε αντικείμενο λαμβάνει την απαραίτητη πληροφορία από τα άλλα αντικείμενα;
  - πώς μοντελοποιείται το δίκτυο;
- Λύσεις για τα ζητήματα:
  - τα αντικείμενα ελέγχου συνδέονται με αισθητήρες (sensors) και ενεργοποιητές (actuators) από τους οποίους λαμβάνουν πληροφορίες για το περιβάλλον και εκτελούν εντολές στα αντικείμενα του περιβάλλοντος, αντίστοιχα.
  - το δίκτυο μοντελοποιείται ως ξεχωριστή οντότητα και όλα τα αντικείμενα ελέγχου αλληλεπιδρούν με αυτό, λαμβάνοντας μηνύματα από άλλα αντικείμενα ή αποστέλλοντας μηνύματα σε αυτά

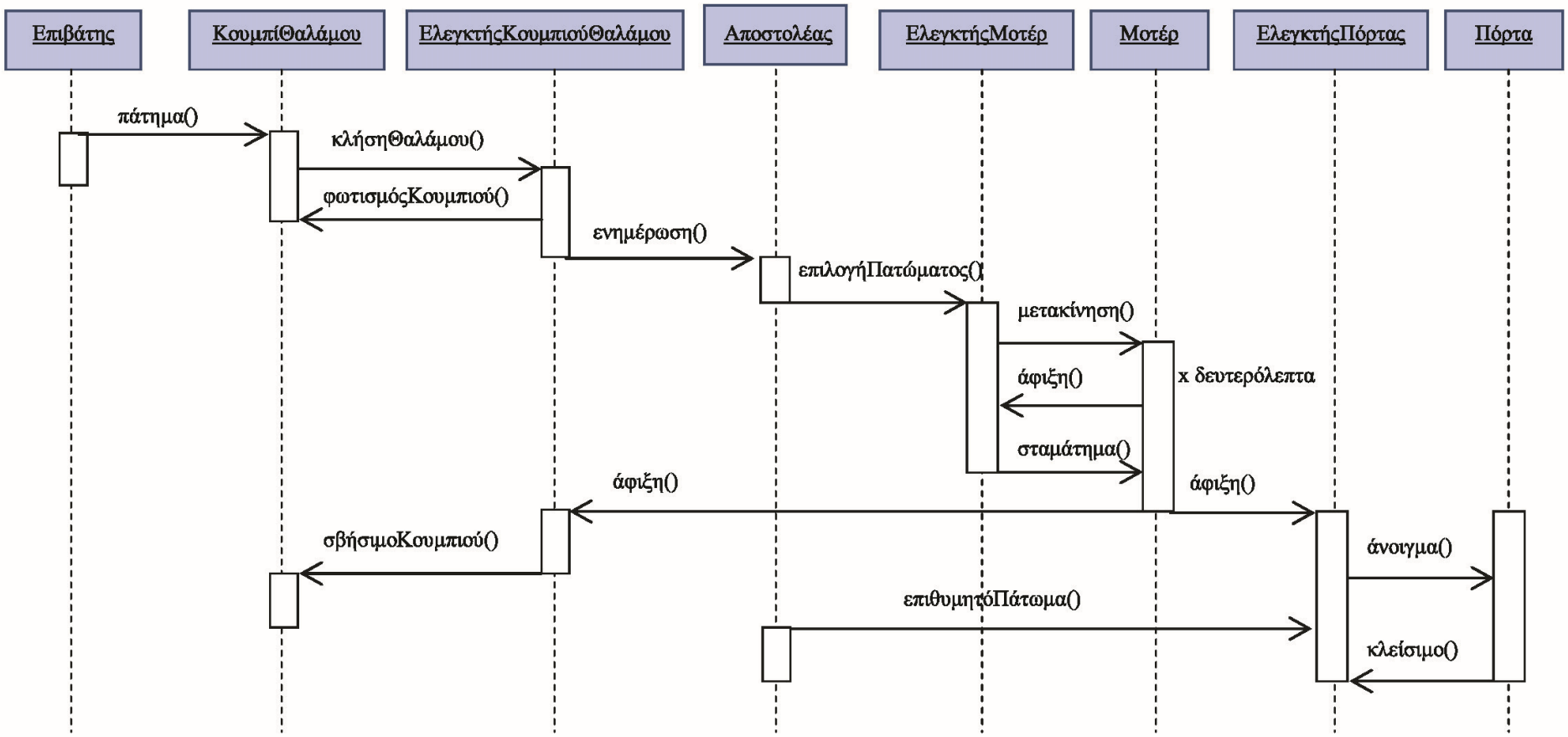
Παράδειγμα μοντελοποίησης με επίλυση ζητημάτων



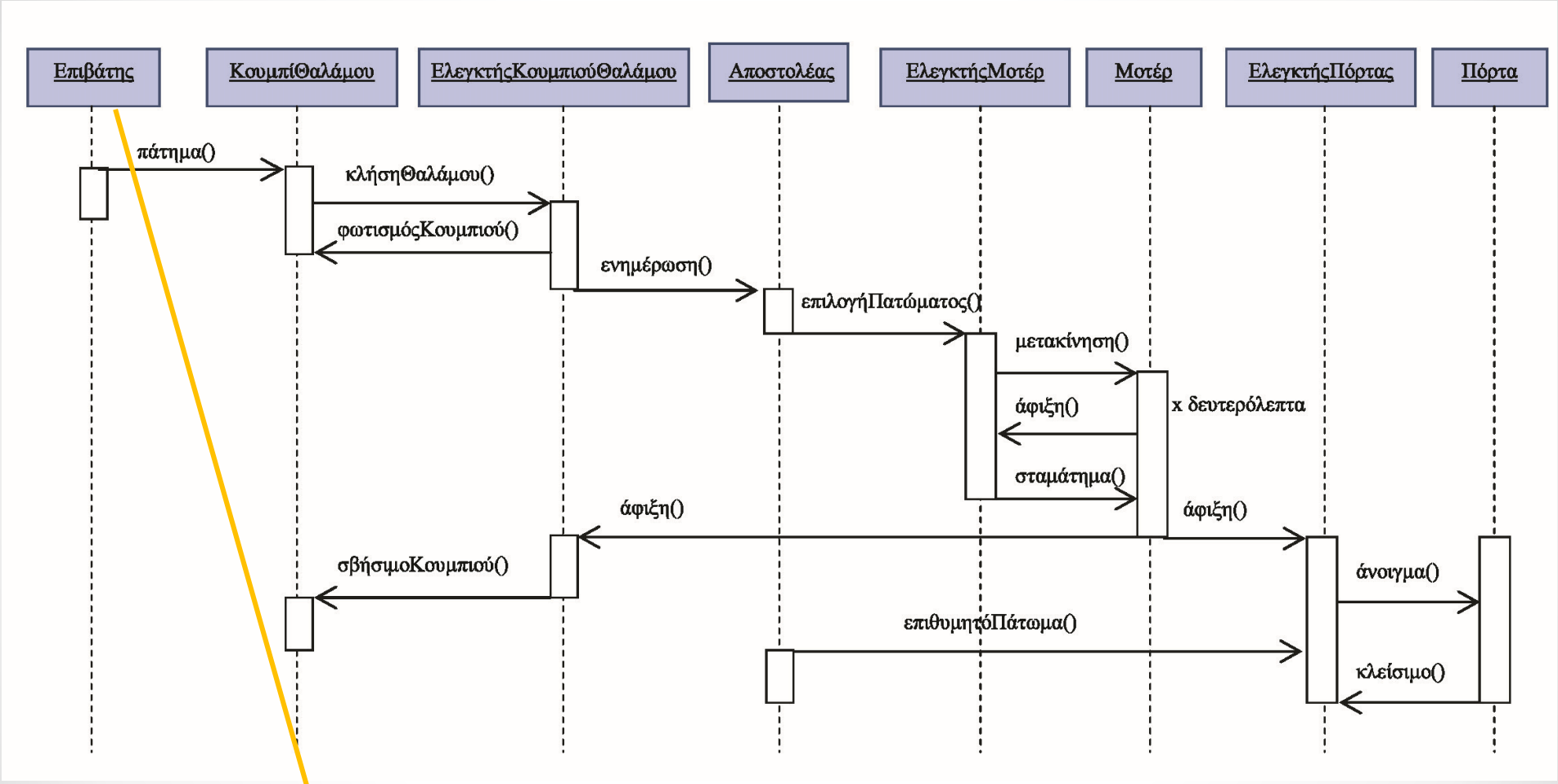
Αποτύπωση δυναμικής συμπεριφοράς συστήματος στον αντικειμενοστρεφή σχεδιασμό

- Ο συνηθέστερος τρόπος είναι η χρήση διαγραμμάτων ακολουθίας
- Τα διαγράμματα ακολουθίας απεικονίζουν το πως επικοινωνούν τα αντικείμενα μεταξύ τους μέσω μηνυμάτων προκειμένου να ολοκληρώσουν τη λειτουργία που αιτείται ο χρήστης του συστήματος
  - Κάθε τέτοιο μήνυμα αντιστοιχεί σε μία μέθοδο που πρέπει να υλοποιεί το αντικείμενο που λαμβάνει το μήνυμα
- Με τα διαγράμματα ακολουθίας επαληθεύεται ότι οι κλάσεις και οι μέθοδοι που έχουμε επιλέξει είναι κατάλληλες για την υλοποίηση του συστήματος, και όταν διαπιστωθεί πρόβλημα γίνονται οι κατάλληλες αλλαγές

Παράδειγμα διαγράμματος ακολουθίας

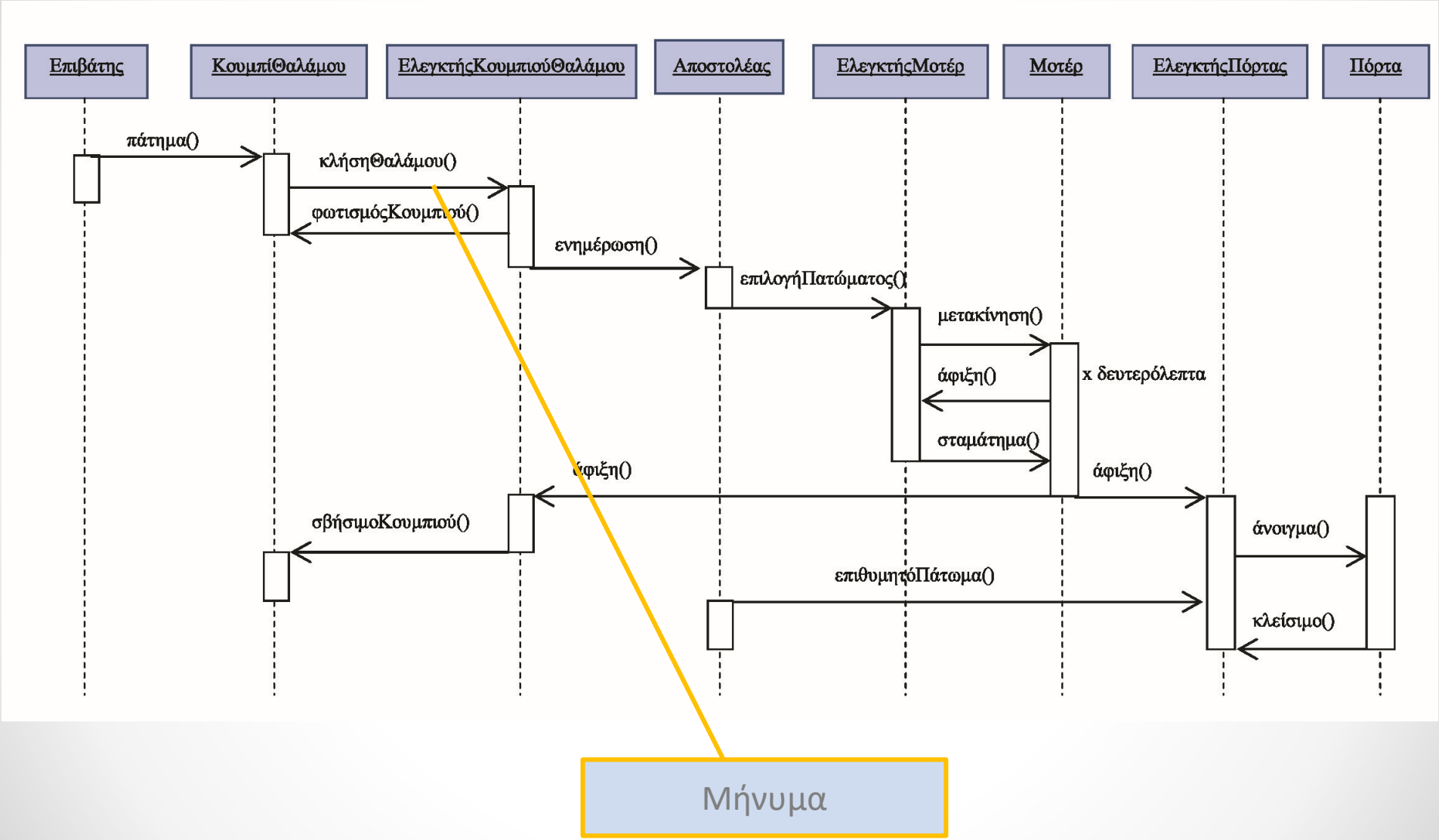


Παράδειγμα διαγράμματος ακολουθίας

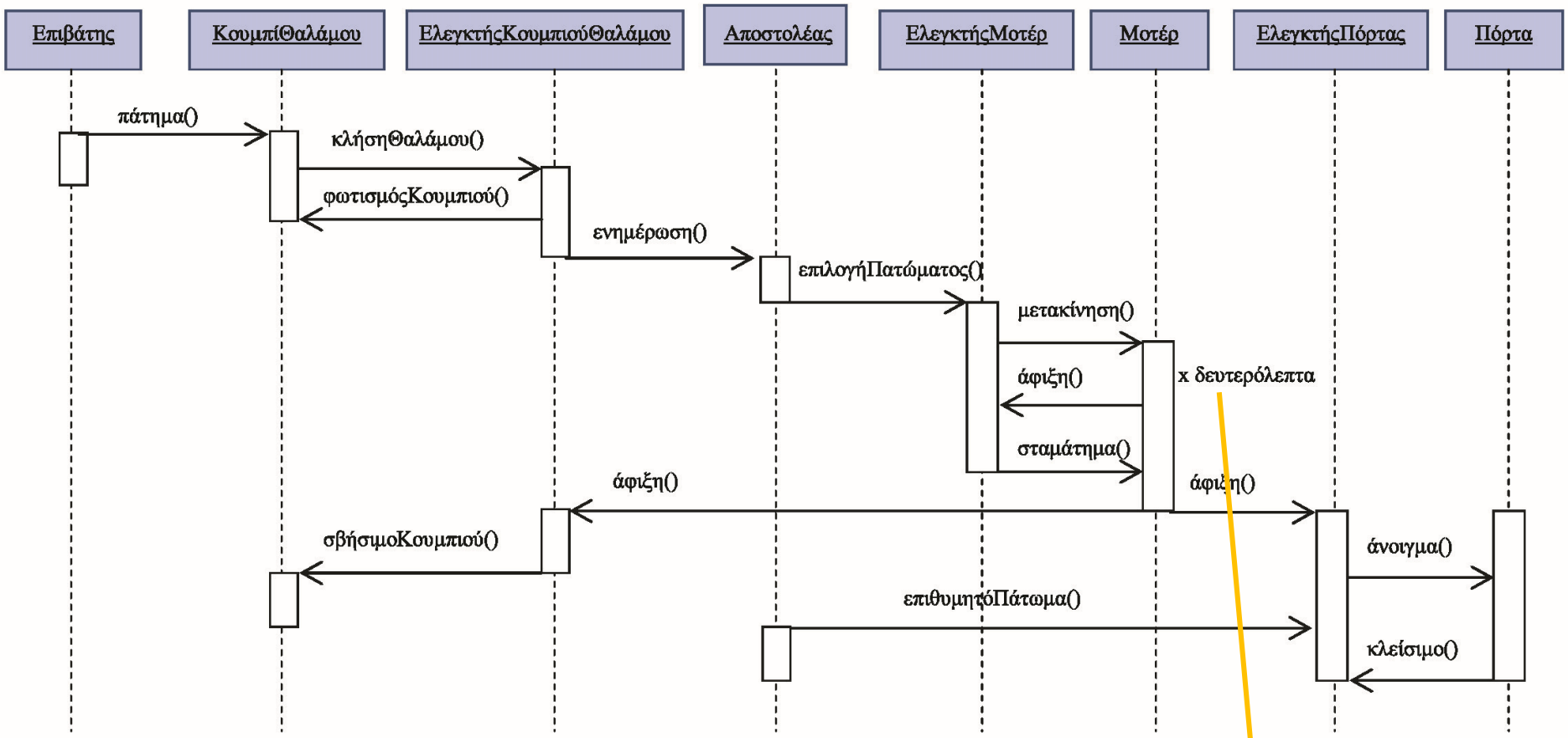


Χρήστης που αιτείται τη λειτουργία

Παράδειγμα διαγράμματος ακολουθίας



Παράδειγμα διαγράμματος ακολουθίας



Χρονική συνθήκη/περιορισμός

Επιλογή Επεξεργαστών

- Η επιλογή επεξεργαστών μπορεί να γίνεται από τον σχεδιαστή ή από τον πελάτη
  - στη 2η περίπτωση, αποτελεί περιορισμό στην υλοποίηση του συστήματος
- Αν έχουμε να επιλέξουμε επεξεργαστές, πρέπει να επιλεγεί ο κατάλληλος ώστε να διαθέτει επαρκή επεξεργαστική ισχύ με το μικρότερο κόστος
- Αν οι επεξεργαστές είναι δεδομένοι, πρέπει να *επαληθεύσουμε* ότι διαθέτουν επαρκή ισχύ για τις ανάγκες του συστήματος

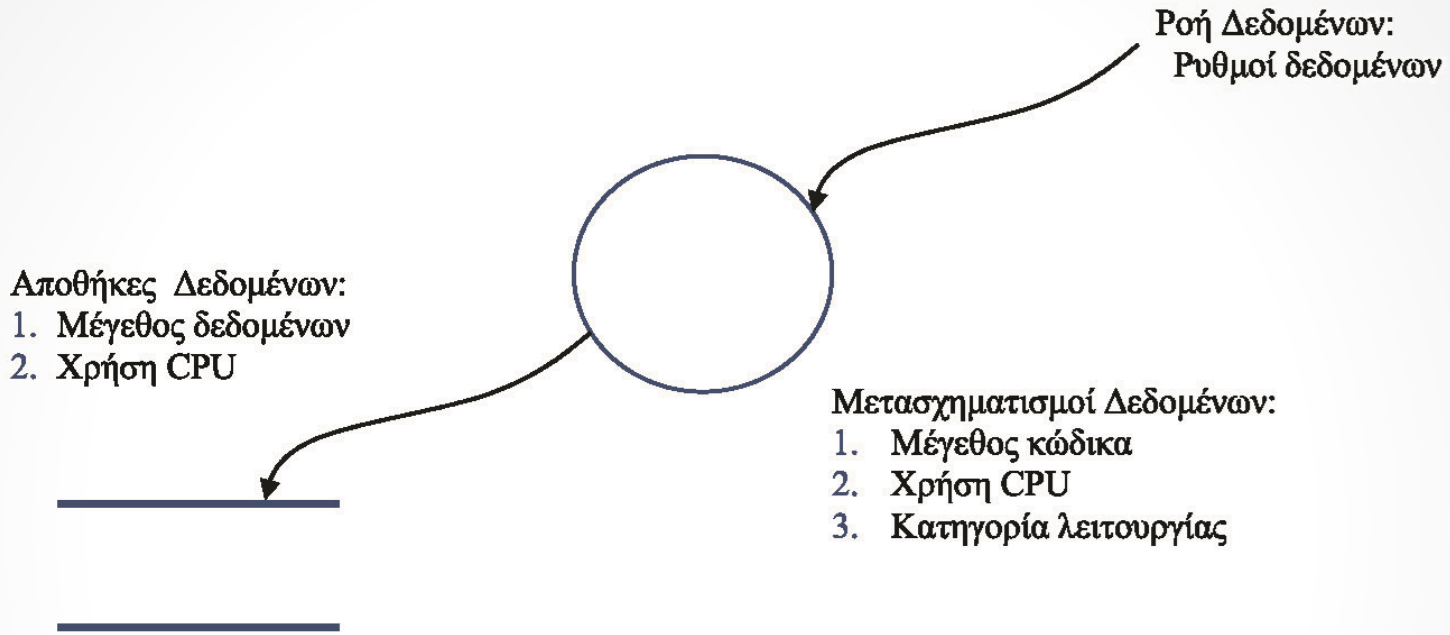
Επιλογή Επεξεργαστών

- Μπορεί σε αυτή τη φάση να προταθούν αλλαγές στις αρχικές απαιτήσεις ώστε οι προεπιλεγμένοι επεξεργαστές να ανταποκριθούν σε αυτές ή για να μειωθεί το κόστος
  - Αυτό μπορεί να έχει επιπτώσεις και στις μη λειτουργικές απαιτήσεις, π.χ. να επιλεγεί η συγγραφή κώδικα με χειρότερη δομή ώστε να έχουμε βελτίωση στην απόδοση
  - Ωστόσο, η χειρότερη δομή του κώδικα θα τον καταστήσει λιγότερο επαληθεύσιμο και συντηρήσιμο
- Δεν είναι απαραίτητο να ορίσουμε συγκεκριμένο επεξεργαστή, αλλά θα πρέπει να ορίσουμε το εύρος των δυνατοτήτων του
  - Επεξεργαστική ισχύ, μνήμη, χειρισμό σημάτων διακοπής, δυνατότητες πολυνηματισμού, κ.λπ.
  - Με αυτόν τον τρόπο προάγουμε και τη μεταφερσιμότητα του συστήματος

Εκτίμηση των απαιτήσεων επεξεργασίας

- Πριν την επιλογή επεξεργαστών θα πρέπει να αξιολογήσουμε την απόδοση και τις ικανότητες που πρέπει να έχουν στα ακόλουθα:
  - το βασικό μοντέλο συμπεριφοράς.
  - τους περιορισμούς του περιβάλλοντος και της υλοποίησης,
  - οποιουσδήποτε περαιτέρω περιορισμούς του σχεδιασμού και
  - δυνατότητες και ανάγκες παραλληλισμού.
- Αρχικά, εξετάζουμε το βασικό μοντέλο συμπεριφοράς για να προσδιορίσουμε την «ποσότητα» που απαιτείται για κάποιο πόρο
  - Για πιο ακριβή εκτίμηση χρησιμοποιούμε αρχικά τα διαγράμματα ροής δεδομένων των κατώτερων επιπέδων του βασικού μοντέλου συμπεριφοράς και εκτιμούμε κάθε μετασχηματισμό ξεχωριστά

Χρήση διαγραμμάτων ροής χαμηλού επιπέδου για εκτίμηση αναγκών επεξεργασίας



- Μέγεθος κώδικα: υποστηριζόμενη RAM
- Χρήση CPU: απαραίτητη υπολογιστική ισχύς ανά επεξεργασία
- Ρυθμοί δεδομένων: για τον υπολογισμό της συνολικής υπολογιστικής ισχύος που είναι απαραίτητη
- Κατηγορία λειτουργίας: για ομαδοποίηση των λειτουργιών ανά επεξεργαστή
- Για τις αποθήκες δεδομένων, το μέγεθος, το είδος (πτητική έναντι μη πτητικής) και η ταχύτητα της μνήμης είναι τα πιο σημαντικά στοιχεία

Εκτίμηση απαιτούμενων πόρων

- Είναι πολύπλοκη και δύσκολη διαδικασία
  - Υπάρχουν τεχνικές για την υποβοήθησή της, ωστόσο η εμπειρία παίζει πολύ σημαντικό ρόλο και δεν υποκαθίσταται από αυτές
  - Αν έχουμε υλοποιήσει παρόμοιο σύστημα, μπορούμε να αξιοποιήσουμε τις γνώσεις που αποκομίσαμε από την υλοποίησή του
- Εξετάζονται οι περιορισμοί υλοποίησης
- Διερευνούμε ενδεχόμενους περιορισμούς που τίθενται από τη γνώση του τι ακολουθεί αργότερα στο σχεδιασμό, π.χ.:
  - *επιβεβλημένη χρήση συγκεκριμένης αρχιτεκτονικής λογισμικού.*
    - Π.χ. MVC, τριεπίπεδη (three-tier) αρχιτεκτονική, συγκεκριμένο Λ.Σ. κ.λπ.
  - *προβληματικές περιοχές*: τι –κατά την εκτίμησή μας– μπορεί να προκαλέσει προβλήματα, π.χ. ζητήματα θερμοκρασίας στο προβλεπόμενο περιβάλλον λειτουργίας
- Εξετάζουμε ανάγκες για παραλληλία, για ανάδειξη απαιτήσεων σε πολυνηματισμό, χειρισμό σημάτων διακοπής, ύπαρξη εντολών κλειδώματος κ.λπ.

Αξιολόγηση επεξεργαστών του πραγματικού κόσμου

- Οι προδιαγραφές που έχουν συγκεντρωθεί αντιπαραβάλλονται με τις δυνατότητες των επεξεργαστών που διατίθενται στο εμπόριο
- Εξετάζουμε αν διαθέτουν:
  - τα λειτουργικά χαρακτηριστικά που πρέπει
  - επαρκή ποσότητα πόρων
  - εξειδικευμένα χαρακτηριστικά που χρειαζόμαστε όπως π.χ. εντολές πολυμέσων (MMX) ή εντολές εικονικοποίησης (virtualization - VT-x)
- Ο επεξεργαστής πρέπει να έχει επαρκή ποσότητα πόρων
  - Αύξηση διαθέσιμων πόρων → αύξηση κόστους, άρα επιλέγουμε τον επεξεργαστή που *μόλις* υπερκαλύπτει τις απαιτήσεις, και όχι μοντέλα πιο πλούσια σε πόρους
  - Αν έχουμε αμφιβολία για το αν ένα μοντέλο επεξεργαστή είναι επαρκές, καλύτερα να επιλέξουμε το μεγαλύτερο μοντέλο (άλλωστε η ποσότητα των απαιτούμενων πόρων είναι *εκτίμηση* άρα υπάρχει περιθώριο απόκλισης)
- Παρέχουμε «συμβουλευτική απαιτήσεων» στους πελάτες
  - Οι απαιτήσεις τους σε χρόνους και επιδόσεις μπορεί να είναι υπερβολικά και αχρείαστα αυστηρές, ειδικά όταν δεν έχουν εικόνα για την επίπτωση στο κόστος!
  - Πιθανόν οι πελάτες να προσαρμόσουν-εκλογικεύσουν τις απαιτήσεις τους