

# ΑΛΓΟΡΙΘΜΟΙ ΠΡΑΓΜΑΤΙΚΟΥ ΧΡΟΝΟΥ

ΚΕΦΑΛΑΙΟ 13<sup>ο</sup>: ΔΙΑΓΡΑΜΜΑΤΑ ΔΟΜΗΣ  
(STRUCTURE CHARTS)

ΣΥΝΟΠΤΙΚΗ ΑΝΑΦΟΡΑ

Χάρης Γεωργίου, Μ-177

Αθήνα, Μάιος 1998.

### 13.1 ΕΙΣΑΓΩΓΗ

Μέχρι στιγμής, έχουμε μοντελοποιήσει τη συμπεριφορά στα πλαίσια των DFDs, STDs και ERDs. Αν και τα διαγράμματα αυτά είναι οργανωμένα ιεραρχικά, κάθε ένα από αυτά αποτελεί ένα δίκτυο, κάτι που τα κάνει ιδανικά για την μοντελοποίηση της φυσικής συμπεριφοράς.

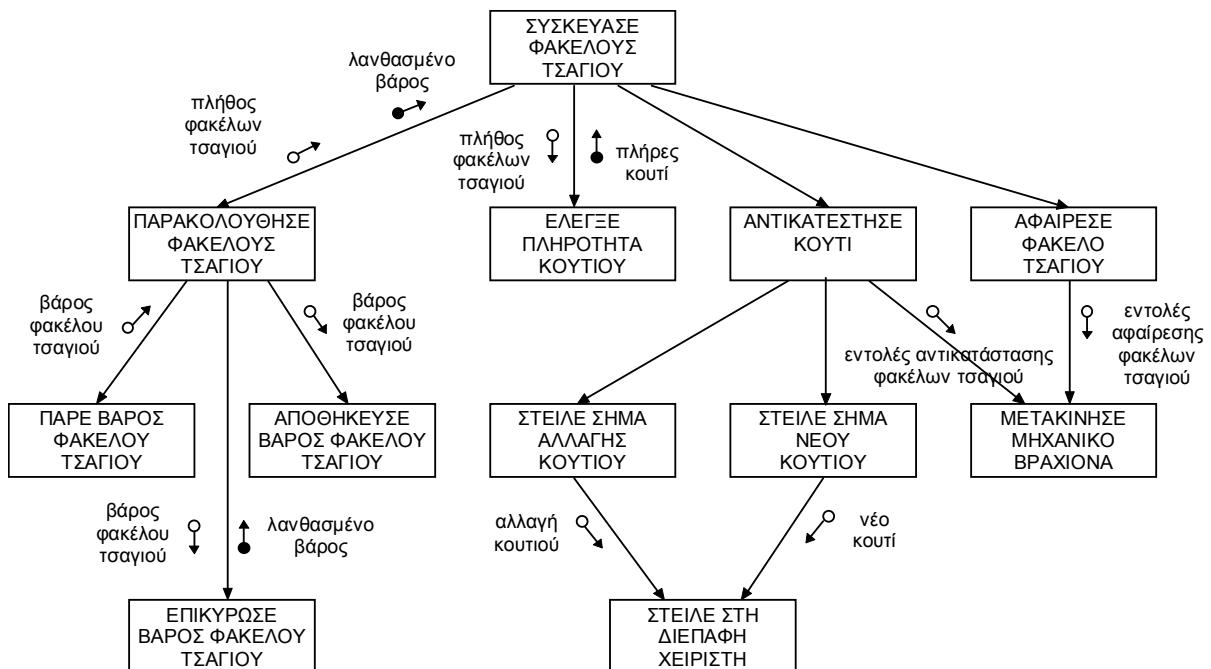
Το διάγραμμα δομής έχει την ίδια συνολική ιεραρχική δομή όπως ένα πρόγραμμα, κάνοντας εύκολη τη μετάφραση ενός διαγράμματος δομής σε κώδικα. Για κάθε μια μονάδα εκτέλεσης στο σύστημα, κατασκευάζεται ένα αντίστοιχο διάγραμμα δομής. Δίνει μια συνοπτική περιγραφή του σκοπού και της οργάνωσης του προγράμματος.

### 13.2 ΣΥΣΤΑΤΙΚΑ ΔΙΑΓΡΑΜΜΑΤΩΝ ΔΟΜΗΣ

Τα διαγράμματα δομής χρησιμοποιούνται εκτενώς στην ανάπτυξη και τεκμηρίωση προγραμμάτων. Δεν υπάρχει λόγος παρουσίασης στοιχείων στα διαγράμματα δομής τα οποία δεν μπορούν να υλοποιηθούν στη γλώσσα προγραμματισμού.

#### 13.2.1 Βασική σημειογραφία

Μια μονάδα (module) παριστάνεται ως ορθογώνιο κουτί, με όνομα που περιγράφει τι κάνει η συγκεκριμένη μονάδα. Οι μονάδες συνδέονται μέσω κλήσεων (module calls), οι οποίες μπορεί να έχουν συσχετισμούς που αναπαριστούν τις παραμέτρους που μεταβιβάζονται μεταξύ των μονάδων.



**Σχήμα 13.1:** Διάγραμμα δομής για το σύστημα συσκευασίας φακέλων τσαγιού.

### 13.2.2 Μονάδες (modules)

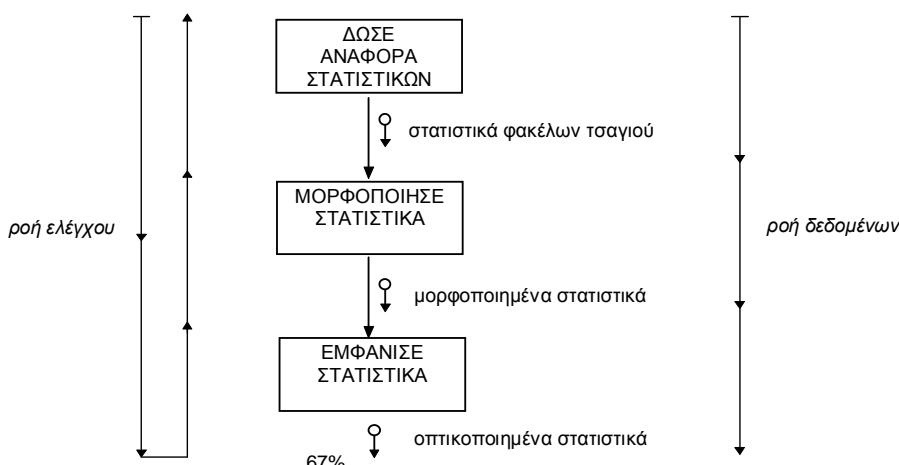
Μια μονάδα είναι μια συλλογή από οδηγίες που αντιμετωπίζονται ως ενότητα από τη γλώσσα προγραμματισμού που χρησιμοποιείται - θα μπορούσε να είναι μια διαδικασία, μια υπορούτινα ή μια συνάρτηση. Σε μια καλά δομημένη γλώσσα προγραμματισμού κάθε μονάδα θα περιλαμβάνει μια απλή συνάρτηση. Υπό αυτή την άποψη, μια μονάδα μοιάζει με ένα μετασχηματισμό δεδομένων και θα πρέπει να της δίνεται ένα απλό, περιεκτικό όνομα που αντικατοπτρίζει τι ακριβώς κάνει όταν καλείται.

Αν και η λειτουργία κάθε μονάδας μπορεί να εξεταστεί από το άμεσα διάγραμμα δομής (από το όνομα και τις παραμέτρους της), τα διαγράμματα αυτά δεν έχουν ως σκοπό την περιγραφή του τρόπου λειτουργίας της κάθε μονάδας. Η πληροφορία αυτή (αλγόριθμος και εσωτερικά δεδομένα μονάδας) εμπεριέχονται στη συνοδευτική τεκμηρίωση της μονάδας.

### 13.2.3 Κλήσεις μονάδων (module calls)

Όπως ακριβώς μια μονάδα εκτέλεσης (execution unit) περιλαμβάνει ένα και μόνο νήμα ελέγχου (control thread) που εκτελείται στο εσωτερικό της, κατά συνέπεια σε κάθε διάγραμμα δομής υπάρχει επίσης ένα μοναδικό νήμα ελέγχου, που σημαίνει ότι μονάχα μία μονάδα (module) θα είναι ενεργή σε κάθε χρονική στιγμή.

Οι ροές των δεδομένων και του ελέγχου διαμέσου του διαγράμματος δομής συχνά διαφέρουν, ενώ στα DFD διαγράμματα οι ροές δεδομένων και ελέγχου συνήθως ταυτίζονται.



**Σχήμα 13.2:** Ροή ελέγχου και ροή δεδομένων

Μια μονάδα μπορεί να καλείται από πολλές άλλες μονάδες ανώτερων επιπέδων. Παρόλα αυτά, θα πρέπει να εμφανίζεται μία και μοναδική φορά στο διάγραμμα δομής. Στην ενότητα 13.2.5 θα εξεταστούν ειδικοί τρόποι διασύνδεσης μονάδων, για την αποφυγή μπερδεμάτων που

προκαλούν μονάδες οι οποίες καλούνται από πολλές άλλες που βρίσκονται διεσπαρμένες στο διάγραμμα δομής.

Πληροφορίες σχετικά με το πότε και γιατί οι μονάδες αυτές καλούνται περιλαμβάνονται στη συνοδευτική τεκμηρίωση της μονάδας «συσκεύασε φακέλους τσαγιού».

Τέλος, δεν ισχύει πάντα η από αριστερά προς δεξιά διαδοχή των κλήσεων μονάδων. Όπου είναι δυνατό, το διάγραμμα δομής θα πρέπει να κατασκευάζεται με τέτοιο τρόπο, ώστε οι μονάδες να εμφανίζονται με τη σωστή διαδοχή από αριστερά προς τα δεξιά, όπως ακριβώς διαβάζονται με φυσικό τρόπο από τους ανθρώπους.

### 13.2.4 Δεδομένα

Υπάρχουν δύο λόγοι για την εμφάνιση των δεδομένων που χρησιμοποιούνται κοινά από τις μονάδες, είτε ως διαμοιραζόμενα μεταξύ των μονάδων δεδομένα, είτε ως παράμετροι κλήσεων, στο διάγραμμα δομής. Πρώτον, βοηθούν στην κατανόηση του τι ακριβώς κάνει το πρόγραμμα. Δεύτερον, βοηθούν στην εκτίμηση του αν το πρόγραμμα είναι σωστά δομημένο.

#### 13.2.4.1 Σύνδεσμοι (*couples*)

Οι μονάδες επικοινωνούν μεταξύ τους χρησιμοποιώντας συνδέσμους. Οι σύνδεσμοι μοιάζουν πολύ με τις ροές δεδομένων ή γεγονότων, όμως καλούνται σύνδεσμοι αφού επιτρέπουν να παρουσιαστεί η σύνδεση ή αλλιώς σύζευξη (το μέγεθος και την πολυπλοκότητα της διεπαφής) μεταξύ των μονάδων.

Το βέλος στο σύμβολο του συνδέσμου φανερώνει την κατεύθυνση κατά την οποία μεταβιβάζεται ο σύνδεσμος. Ο σύνδεσμος δεδομένων (data couple), όπου ο κύκλος στο άκρο του βέλους δεν είναι γεμισμένος, δείχνει τα δεδομένα που μεταβιβάζονται.

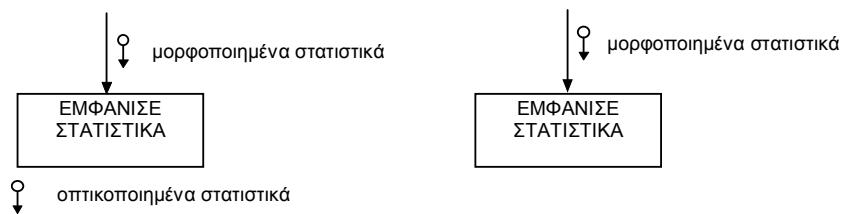
Οι σύνδεσμοι ελέγχου (control couples) διαφέρουν από τις ροές γεγονότων, μια και είναι δυνατό να περιλαμβάνουν πολλές διαφορετικές τιμές κατάστασης, ενώ οι ροές γεγονότων περιορίζονται σε αληθές ή ψευδές.

Όπως υποδηλώνει και το όνομα, ο σύνδεσμος ελέγχου χρησιμοποιείται για τον έλεγχο του προγράμματος. Αν μια μονάδα αποφασίζει τι θα κάνει στη συνέχεια βασιζόμενη στην πληροφορία που λαμβάνει, τότε η πληροφορία αυτή μεταβιβάζεται μέσω ενός συνδέσμου ελέγχου:

```
CALL VALIDATE TEABAG WEIGHT (teabag weight : wrong weight)
IF wrong weight = FALSE
    THEN teabag count := teabag count + 1
```

Από την άλλη πλευρά, αν μια μονάδα λαμβάνει πληροφορία και την επεξεργάζεται με τον ίδιο τρόπο, ανεξάρτητα από την τιμή της, η πληροφορία αυτή μεταβιβάζεται μέσω ενός συνδέσμου δεδομένων.

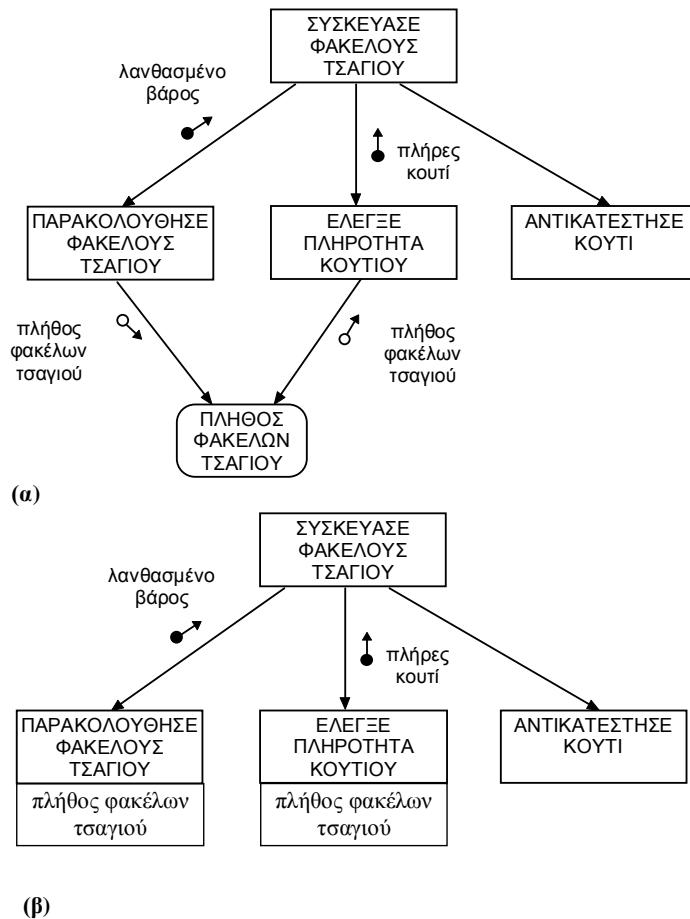
Το αν παρουσιάζονται σύνδεσμοι εισερχόμενοι ή εξερχόμενοι από το κάτω μέρος του διαγράμματος δομής, δηλαδή αποστέλλοντας ή λαμβάνοντας δεδομένα εκτός των ορίων του προγράμματος, εξαρτάται από τον σχεδιαστή. Και οι δύο εκδόσεις της μονάδας «εμφάνισε στατιστικά» που παρουσιάζονται στο σχήμα 13.3 είναι το ίδιο ορθές. Σε εφαρμογές όπου οι διεπαφές συσκευών είναι επιρρεπείς σε αλλαγές, ή το υλικό (hardware) αναπτύσσεται παράλληλα με το λογισμικό, συχνά είναι χρήσιμο να εμφανίζονται οι φυσικές είσοδοι και έξοδοι που διαχειρίζεται το πρόγραμμα.



**Σχήμα 13.3:** Σύνδεσμοι που μεταβιβάζονται εκτός του διαγράμματος δομής

#### 13.2.4.2 Διαμοιραζόμενα δεδομένα (shared data)

Όπως οι σύνδεσμοι χρησιμοποιούνται για τη μεταβίβαση παραμέτρων, μπορούν επίσης να αναπαριστάνουν διαμοιραζόμενα δεδομένα.



**Σχήμα 13.4:** Διαμοιραζόμενα δεδομένα: (α) κοινά/καθολικά δεδομένα, (β) ιδιαίτερα διαμοιραζόμενα δεδομένα.

Το σχήμα 13.4(β) παρουσιάζει τα ιδιαίτερα δεδομένα (private data), τα οποία μπορεί να υποδηλώνουν δυνατότητες απόκρυψης δεδομένων (data hiding) που πιθανόν να προσφέρονται από τη γλώσσα προγραμματισμού ή την αρχιτεκτονική του λογισμικού. Εδώ, η μονάδα «παρακολούθησε φακέλους τσαγιού» και η μονάδα «έλεγχε πληρότητα κουτιού» έχουν εξίσου πρόσβαση στην παράμετρο «πλήθος φακέλων τσαγιού», αλλά αφού τα δεδομένα είναι προσβάσιμα μόνο από αυτές, καμία άλλη από τις μονάδες δεν μπορεί να χρησιμοποιήσει ή να καταστρέψει τα δεδομένα.

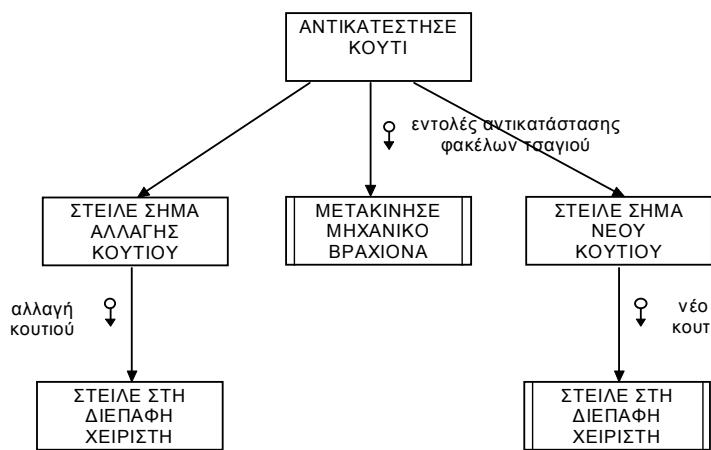
### 13.2.5 Περαιτέρω σημειογραφία

Πέρα από τα σύμβολα που εξετάζονται παρακάτω, υπάρχουν και άλλα σύμβολα που χρησιμοποιούνται σε διαγράμματα δομής και που, λόγω του μεγάλου πλήθους τους, δεν αναφέρονται - αυτό δε σημαίνει πως δεν θα πρέπει να χρησιμοποιούνται, εφ' όσον είναι κατανοητά και βοηθούν τη σχεδίαση.

#### 13.2.5.1 Μονάδα βιβλιοθήκης (*library module*)

Το σύμβολο μονάδας βιβλιοθήκης υποδηλώνει κλήση σε κάποια διαδικασία βιβλιοθήκης. Αφού οι μονάδες βιβλιοθήκης υπάρχουν ήδη, δεν θα υπάρχουν μονάδες που θα καλούνται από το σύμβολο μονάδας βιβλιοθήκης και δεν θα υπάρχει συνοδευτική τεκμηρίωση.





(β)

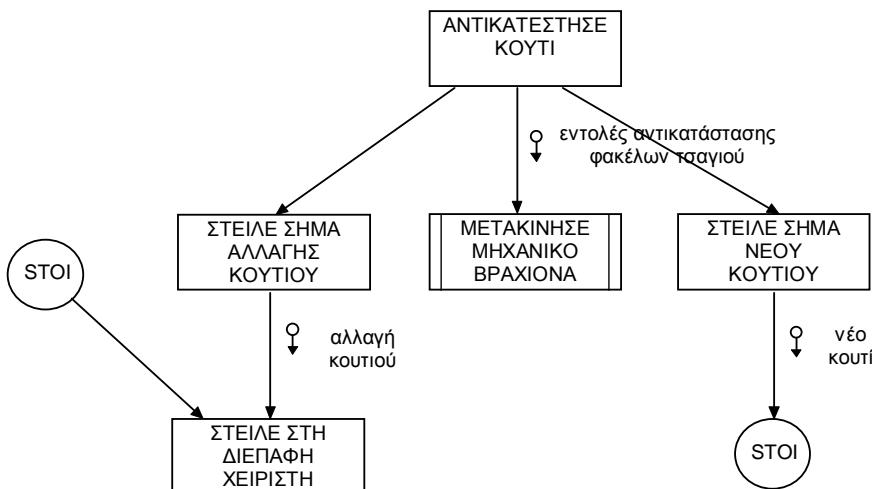
**Σχήμα 13.5:** Χρήση μονάδων βιβλιοθήκης: (α) ως αρχιτεκτονική λογισμικού, (β) ως επαναλαμβανόμενη μονάδα.

Το σύμβολο αυτό έχει δύο ακόμη χρήσεις:

1. Το σύμβολο μονάδας βιβλιοθήκης μπορεί να δείξει κλήσεις στα πλαίσια της αρχιτεκτονικής του λογισμικού.
2. Το σύμβολο μονάδας βιβλιοθήκης μπορεί να παριστάνει την ίδια μονάδα, η οποία καλείται πολλές φορές στο ίδιο διάγραμμα δομής, όπως παρουσιάζεται στο διάγραμμα 13.5(β).

#### 13.2.5.2 Συνδέτης (connector)

Εάν δεν είναι δυνατό να συνδεθούν όλες οι καλούμενες μονάδες μεταξύ τους, είτε γιατί εμφανίζονται σε διαφορετικά φύλλα του σχεδίου, είτε γιατί οι συνδέσεις θα προκαλούσαν διασταυρούμενες γραμμές, χρησιμοποιείται το σύμβολο του συνδέτη (connector) που παρουσιάζεται στο σχήμα 13.6.



**Σχήμα 13.6:** Σύμβολο συνδέτη

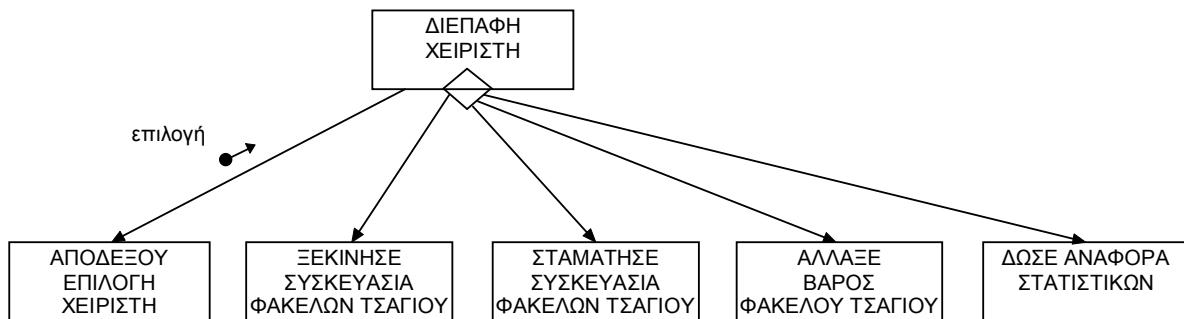
#### 13.2.5.3 Κέντρο επιλογής/συναλλαγής (selection/transaction centre)

Το σύμβολο του διαμαντιού καλείται σύμβολο επιλογής ή κέντρο συναλλαγής και υποδηλώνει ότι οι μονάδες που συνδέονται σε αυτό καλούνται με αμοιβαίως αποκλειστικό τρόπο. Το σύμβολο κέντρου επιλογής/συναλλαγής πάντα υποδηλώνει κάποια εντολή πολλαπλής επιλογής («case statement» ή ισοδύναμη δομή) στον κώδικα.

```

CALL accept operator choice (:choice)
CASE choice OF
    start : CALL start boxing teabags
    stop  : CALL stop boxing teabags
    change : CALL change teabag weight
    report : CALL provide statistics report
END CASE

```



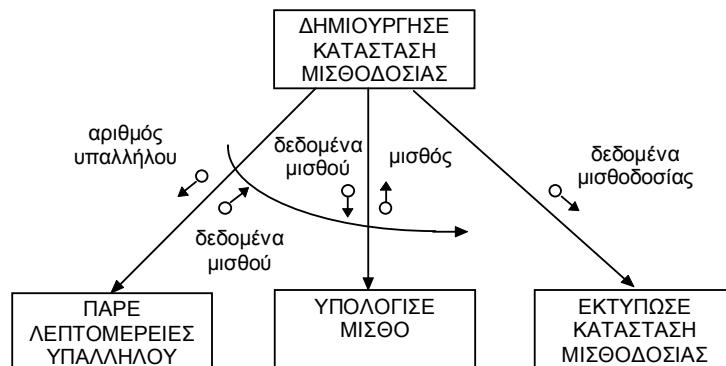
**Σχήμα 13.7:** Σύμβολο κέντρου επιλογής/συναλλαγής.

#### 13.2.5.4 Επανάληψη (repetition)

```

FOR each employee
BEGIN
    CALL get employee details (employee number : salary data)
    CALL calculate salary (salary data : salary)
END
CALL print payroll (payroll data:)

```



**Σχήμα 13.8:** Σύμβολο επανάληψης.**13.2.5.5 Αναδρομή (recursion)**

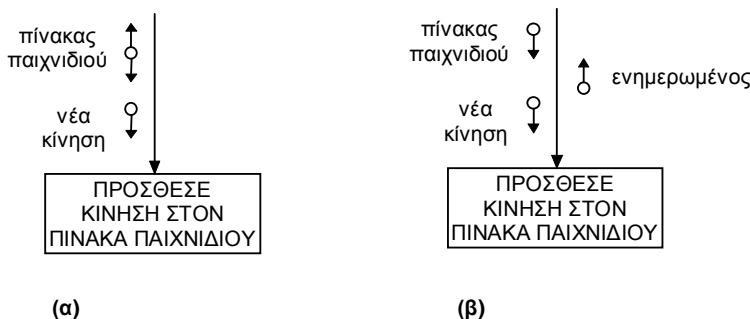
Αν η γλώσσα προγραμματισμού που χρησιμοποιείται επιτρέπει κάτι τέτοιο, μια μονάδα μπορεί να καλεί τον εαυτό της, όπως παρουσιάζεται από το σχήμα 13.9.

**Σχήμα 13.9:** Σύμβολο αναδρομής.**13.2.5.6 Συμπερίληψη (inclusion)**

Το «καπέλο» στη μονάδα «πάρε επικυρωμένο βάρος φακέλου τσαγιού» στο σχήμα 13.10 υποδηλώνει ότι δεν αποτελεί ξεχωριστή υπορουτίνα η οποία μπορεί να κληθεί από τη μονάδα «παρακολούθησε φακέλους τσαγιού», αλλά είναι ένα τμήμα κώδικα στο εσωτερικό της μονάδας.

**Σχήμα 13.10:** Σύμβολο συμπερίληψης.**13.2.5.7 Σύνδεσμος διαλόγου (dialogue couple)**

Ένας σύνδεσμος που κατευθύνεται και προς τα δύο μέρη υποδηλώνει ότι κάποια παράμετρος μεταβιβάζεται στη μονάδα του χαμηλότερου επιπέδου, τροποποιείται και στη συνέχεια επιστρέφεται στην αρχική μονάδα.

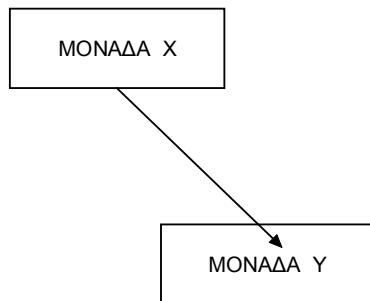


**Σχήμα 13.11:** Σύμβολο συνδέσμου διαλόγου.

#### 13.2.5.8 Σύνδεσμος περιεχομένου (content couple)

MODULE X	MODULE Y
^ ^ ^	^ ^ ^
GOTO L1	^ ^ ^
^ ^ ^	LABEL 1:
^ ^ ^	^ ^ ^

Φυσικά, δεν αναμένεται ο κώδικας του προγράμματος να γραφτεί με παρόμοιο τρόπο. Παρόλα αυτά, αν για οποιοδήποτε λόγο ο κώδικας συμπεριλαμβάνει διακλαδώσεις αυτής της μορφής, είναι καλή ιδέα να εμφανίζονται με αυτό τον τρόπο. Διαφορετικά, θα είναι αδύνατο να κατανοηθεί ο κώδικας.



**Σχήμα 13.12:** Σύμβολο συνδέσμου περιεχομένου.

### 13.3 ΣΥΝΟΔΕΥΤΙΚΗ ΤΕΚΜΗΡΙΩΣΗ

Καθώς το διάγραμμα δομής έχει σαν σκοπό να βοηθήσει την ανάπτυξη, όπως επίσης να παρουσιάσει μια γενική εικόνα του προγράμματος, θα πρέπει να υπάρχει συνοδευτική τεκμηρίωση προδιαγραφών για όλα τα συστατικά του.

#### 13.3.1 Προδιαγραφές δεδομένων

Οι σύνδεσμοι (οι οποίοι μεταβιβάζονται άμεσα ή μέσω διαμοιραζόμενων χώρων δεδομένων) εμπεριέχουν την ίδια πληροφορία με τις DFD ροές και προσδιορίζονται με τον ίδιο τρόπο.

$$\text{Θέση βραχίονα} = \text{arm } x + \text{arm } y + \text{arm } z + \text{claw}$$

*arm x = ELEMENTAL*

*σημασία = δίνει τη θέση του βραχίονα σε όρους κίνησης πάνω και κάτω.*

*πεδίο τιμών = -100 ως +100*

*Η τιμή 0 δίνει τη θέση που είναι οριζόντια ως προς τη βάση του μηχανικού βραχίονα, αρνητικές τιμές είναι κάτω από το επίπεδο βάσης, θετικές τιμές είναι πάνω από το επίπεδο βάσης.*

*1 μονάδα = 10 mm.*

Στη συνέχεια η παραπάνω δομή καθορίζεται λεπτομερώς ως μια δομή εγγραφής, με πεδία τα arm x, arm y, arm z και claw.

ARM POSITION = RECORD

ARM X : BYTE

ARM Y : BYTE

ARM Z : BYTE

CLAW : BOOLEAN

### 13.3.2 Προδιαγραφές μονάδων

Οι προδιαγραφές των μονάδων εξυπηρετούν δύο σκοπούς και μπορούν να συγγραφούν σε δύο μέρη. Πρώτον, επιτρέπουν στον προγραμματιστή να γράψει την κώδικα, και δεύτερον επιτρέπουν τη συντήρηση του κώδικα αυτού.

Πριν να υλοποιηθεί η μονάδα, η τεκμηρίωση προδιαγραφών καθορίζει τι ακριβώς θα πρέπει να κάνει η μονάδα, χωρίς πολλές λεπτομέρειες σχετικά με το πώς θα το κάνει.

Η αρχικές προδιαγραφές δεν θα είναι αρκετά λεπτομερείς για να επαρκούν για τη συντήρηση του κώδικα. Μετά την υλοποίηση, η τεκμηρίωση θα πρέπει να επεκταθεί ώστε να επεξηγεί τον τρόπο με τον οποίο η μονάδα υλοποιήθηκε.

Με τρόπο παρόμοιο με τις προδιαγραφές μετασχηματισμών, υπάρχουν πολλαπλές επιλογές για τη μέθοδο τεκμηρίωσης προδιαγραφών για τη μονάδα:

1. ψευδοκώδικας
2. κείμενο
3. διαγράμματα
4. γλώσσα προσδιορισμού προγράμματος (program definition language)
5. πίνακες
6. κάθε άλλη κατάλληλη μέθοδος

Η πλήρης τεκμηρίωση της μονάδας θα πρέπει να περιλαμβάνει τα παρακάτω:

1. περιγραφή των εισόδων και εξόδων της μονάδας.
2. περιγραφή του αλγορίθμου που χρησιμοποιείται στη μονάδα.
3. περιγραφή των εσωτερικών δεδομένων που χρησιμοποιούνται στη μονάδα.

### 13.4 ΚΑΤΑΣΚΕΥΑΖΟΝΤΑΣ ΕΝΑ ΔΙΑΓΡΑΜΜΑ ΔΟΜΗΣ

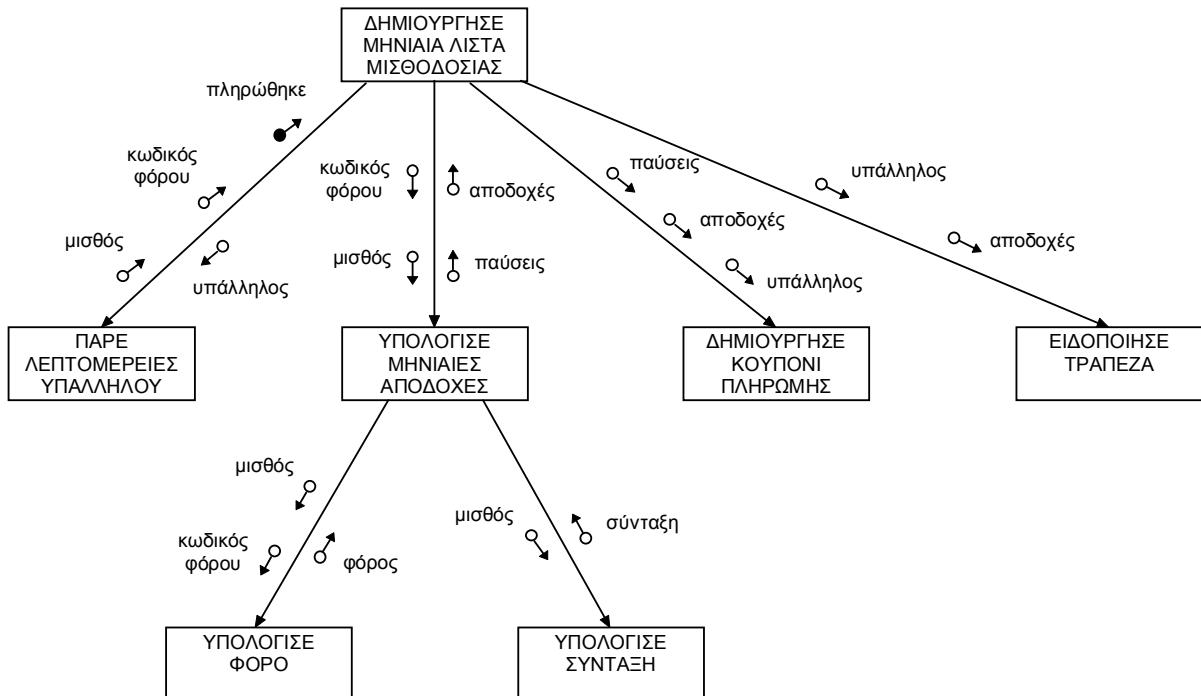
Όταν αναπτύσσεται ένα νέο σύστημα, το διάγραμμα δομής δημιουργείται χρησιμοποιώντας την ιεραρχία των διαγραμμάτων ροής δεδομένων. Η περίπτωση αυτή εξετάζεται στο επόμενο κεφάλαιο. Στην περίπτωση συντήρησης ενός υπάρχοντος συστήματος, το αρχικό διάγραμμα δομής μοντελοποιεί τον υπάρχοντα κώδικα, και κατά συνέπεια δημιουργείται από τον ίδιο τον κώδικα.

Για μερικές γλώσσες προγραμματισμού, υπάρχουν διαθέσιμα πακέτα τα οποία κατασκευάζουν διαγράμματα δομής με αυτόματο τρόπο από ένα τμήμα κώδικα.

```

PRODUCE MONTHLY PAYROLL
begin
    for employee := 1 to no_of_staff do
        begin
            GET EMPLOYEE DETAILS (employee : salary,tax_code,paid)
            if paid = monthly then
                begin
                    CALCULATE MONTHLY WAGES (salary,tax_code : stops,wages)
                    PRODURE PAYSLIP (employee,stops,wages)
                    NOTIFY BANK (employee,wages)
                end
            end
        end
    end
end

```



**Σχήμα 13.13:** Δημιουργία μηνιαίας λίστας μισθοδοσίας - διάγραμμα δομής.

Μερικές πραγματικές γλώσσες προγραμματισμού θέτουν περιορισμούς σχετικά με το πλήθος των χαρακτήρων στα ονόματα υπορουτινών ή παραμέτρων, κάνοντας αδύνατη την ονομασία κάποιας υπορουτίνας ως «get employee details» («πάρε λεπτομέρειες υπαλλήλου»). Σε αυτή την περίπτωση υπάρχουν δύο επιλογές:

1. Διατήρηση των «όχι και τόσο κατανοητών» ονομάτων απευθείας από τον κώδικα στο διάγραμμα δομής, έτσι ώστε να είναι εύκολη η αναφορά μεταξύ του διαγράμματος δομής και του κώδικα, παρόλο που μπορεί το ίδιο το διάγραμμα μπορεί να μην είναι απόλυτα κατανοητό.
2. Εφαρμογή διαφορετικής ονοματολογίας για τις μονάδες και τους συνδέσμους στο διάγραμμα δομής και στον κώδικα, και ενός πίνακα που να δείχνει την αντιστοιχία, έτσι ώστε το διάγραμμα δομής να είναι πλέον κατανοητό, αλλά πιο δύσκολο να αναφερθεί άμεσα στον κώδικα.

Η επιλογή της κατάλληλης μεθόδου αφήνεται στο σχεδιαστή.

### 13.5 ΕΛΕΓΧΟΝΤΑΣ ΕΝΑ ΔΙΑΓΡΑΜΜΑ ΔΟΜΗΣ

#### 13.5.1 Έλεγχος συντακτικής ορθότητας (syntactic check)

Εφόσον το διάγραμμα δομής είναι ένα πολύ απλό διάγραμμα, δεν υπάρχει σχεδόν κανένας συντακτικός έλεγχος. Παρόλα αυτά, αξίζει να σημειωθεί πως κάποια συγκεκριμένη γλώσσα προγραμματισμού μπορεί να μην υποστηρίζει οτιδήποτε μπορεί να μοντελοποιηθεί σε ένα διάγραμμα δομής.

#### 13.5.2 Έλεγχος σαφήνειας (clarity check)

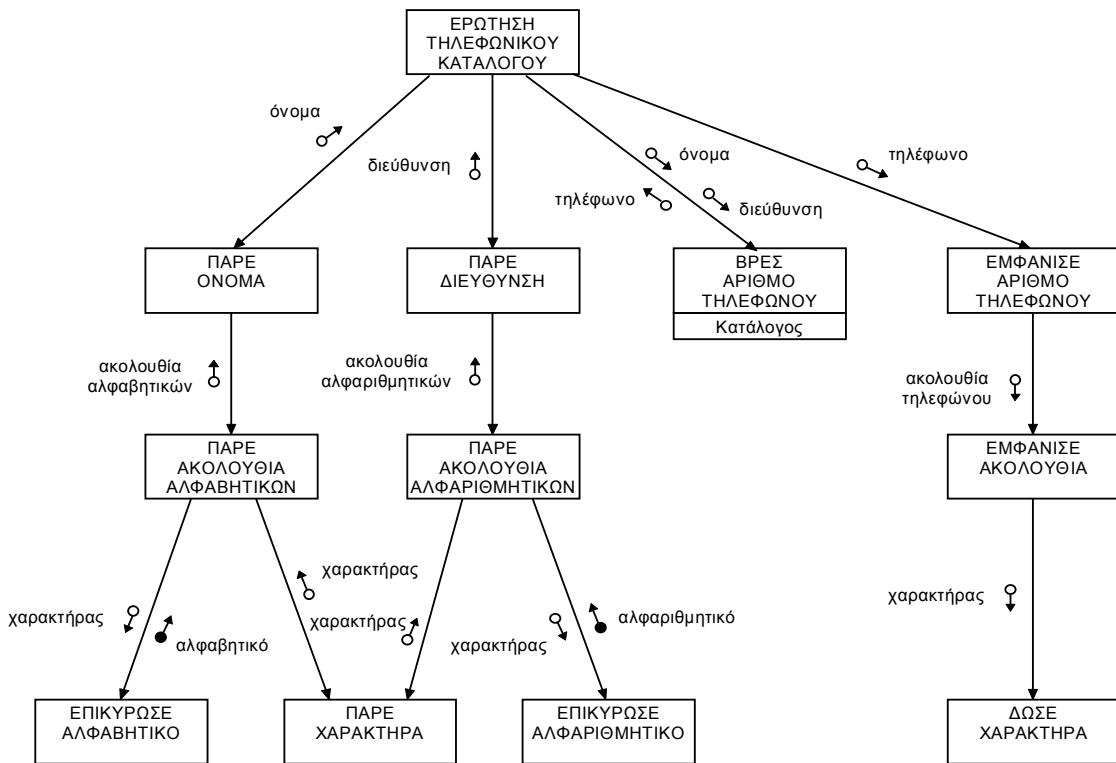
Όπως όλα τα άλλα εργαλεία μοντελοποίησης, η προσεκτική ονοματολογία και τμηματοποίηση θα κάνει το διάγραμμα δομής εύκολα κατανοητό.

##### 13.2.5.1 Ονοματολογία διαγράμματος δομής

Καθώς οι μονάδες και οι σύνδεσμοι είναι συστατικά παρόμοια με τους μετασχηματισμούς δεδομένων και τις ροές, η ονοματολογία και η τμηματοποίηση θα είναι επίσης ανάλογη.

Μερικές φορές το τι ακριβώς κάνει μια μονάδα δεν είναι το ίδιο με αυτό που κάποια μονάδα, που την καλεί, νομίζει ότι κάνει.

Ένα καλά δομημένο σύστημα έχει ένα ζυγισμένο διάγραμμα δομής, όπως αυτό που παρουσιάζεται στο σχήμα 13.14. Το διάγραμμα αυτό έχει τον έλεγχο στην κορυφή, τη λειτουργικότητα και τα δεδομένα υψηλού επιπέδου (ή αναγκαία) στη μέση και τη λειτουργικότητα και τα δεδομένα χαμηλού επιπέδου στο κάτω μέρος.



**Σχήμα 13.14: Ζυγισμένο διάγραμμα δομής.**

Στο πάνω μέρος του διαγράμματος δομής τα ονόματα είναι περισσότερο σημαντικά και κατά συνέπεια πιο κατανοητά σε σχέση με την εφαρμογή.

Το ζύγισμα ενός διαγράμματος δομής μπορεί επίσης να εξεταστεί στα πλαίσια του ελέγχου. Οι μονάδες χαμηλού επιπέδου εκτελούν την εργασία, ενώ ταυτόχρονα οι μονάδες υψηλού επιπέδου εξασφαλίζουν ότι αυτές καλούνται με τη σωστή σειρά. Επομένως, το μεγαλύτερο τμήμα του ελέγχου βρίσκεται στο ανώτερο επίπεδο του διαγράμματος δομής.

### 13.5.2.2 Διαμερισμός (partitioning) διαγράμματος δομής

Μερικές φορές υπάρχει διαφορά μεταξύ των μονάδων που είναι θεωρητικά καλά τμηματοποιημένες και ενός διαγράμματος δομής διαμερισμένου έτσι ώστε να παράγει ένα λειτουργικό κομμάτι κώδικα.

#### Συνεκτικότητα (Cohesion)

Μια καλή μονάδα περιλαμβάνει μια μοναδική εννοιολογικά λειτουργία, την οποία εκτελεί κάθε φορά που καλείται, ανεξάρτητα από ποιο σημείο καλείται. Η συνεκτικότητα κάποιας μονάδας καθορίζει πόσα διαφορετικά πράγματα κάνει και αποτελεί ένα θεωρητικό μέτρο καλού διαμερισμού.

Μονάδες με κακή συνεκτικότητα είναι ανεπιθύμητες για πολλούς λόγους.

1. Αν κάποια μονάδα εκτελεί πολλές διαφορετικές λειτουργίες, είναι δύσκολο να βρεθεί ονομα που να περιγράφει ξεκάθαρα οιδήποτε κάνει. Μονάδες με κακή συνεκτικότητα συνήθως έχουν ονόματα χωρίς νόημα ή, ακόμη χειρότερα, παραπλανητικά.

2. Μια μονάδα η οποία εκτελεί πολλές μικρές διαφορετικές λειτουργίες υποδηλώνει πως μια ενιαία λειτουργία έχει διασπαστεί σε πολλές ανεξάρτητες μονάδες στο διάγραμμα δομής.
3. Λειτουργίες που πρόκειται να εκτελεστούν ταυτόχρονα συχνά ομαδοποιούνται σε μια μονάδα. Αν η αρχική μονάδα είχε γραφτεί ως δύο επιμέρους μονάδες, τότε αυτές θα μπορούσαν να χρησιμοποιηθούν οποιαδήποτε στιγμή σε όλη την εφαρμογή. Μονάδες με καλή συνεκτικότητα είναι πιο πιθανό να χρησιμοποιηθούν ξανά.

Σε ένα πραγματικό πρόγραμμα η έννοια της συνεκτικότητας μπορεί να επεκταθεί περισσότερο από ότι πρέπει.

Κάθε φορά που δημιουργείται μια μονάδα, θα πρέπει γίνεται έλεγχος αν παρόμοιες λειτουργίες απαιτούνται κάπου αλλού στο πρόγραμμα. Θα πρέπει να ισοσταθμίζεται η επιλογή μεταξύ μονάδων που δεν κάνουν αρκετές ενέργειες και μονάδων που κάνουν υπερβολικά πολλές.

Το μέτρο του πόσο «χρήσιμη» είναι μια μονάδα καλείται «fan-in». Στο διάγραμμα δομής του συστήματος συσκευασίας φακέλων τσαγιού, που παρουσιάζεται στο σχήμα 13.1, η μονάδα «στείλε στη διεπαφή χειριστή» έχει fan-in ίσο με 2, καθώς καλείται από δύο μονάδες ανώτερων επιπέδων. Μια μονάδα θα πρέπει να έχει πάντα καλή συνεκτικότητα - με υψηλό fan-in έχει το επιπλέον όφελος ότι μια συγκεκριμένη περιοχή εργασίας, που έχει αναπτυχθεί και υλοποιηθεί μια φορά, χρησιμοποιείται από πολλές άλλες. Επίσης, υπό συντήρηση, αντί να αλλαχθούν πολλά μικρά κομμάτια κώδικα, μονάχα μια μονάδα χρειάζεται να τροποποιηθεί.

### Σύζευξη (coupling)

Το ποσό της πληροφορίας που διαμοιράζεται μεταξύ των μονάδων καλείται σύζευξη. Μονάδες που είναι στενά συνδεδεμένες («συζευγμένες») μεταβιβάζουν πολλές παραμέτρους ή διαμοιράζονται πολλά δεδομένα. Η περισσότερο χαλαρή είναι η σύζευξη όπου δεν μεταβιβάζονται ή διαμοιράζονται καθόλου δεδομένα. Η χαμηλή σύζευξη παρουσιάζει ένα πλήθος από πλεονεκτήματα:

1. Όσο λιγότερα δεδομένα μεταβιβάζονται, τόσο λιγότερα είναι τα στοιχεία που πρέπει να ληφθούν υπόψη και τόσο ευκολότερη είναι η κατανόηση της μονάδας.
2. Μονάδες οι οποίες διαμοιράζονται πολλά δεδομένα τείνουν να εξαρτώνται η μία από την άλλη σε μεγάλο βαθμό. Αυτό υποδηλώνει πως δεν θα είναι εύκολη η κατανόηση ή η χρήση της μιας μονάδας χωρίς την άλλη.
3. Όσο περισσότερα δεδομένα μεταβιβάζονται μεταξύ των μονάδων, τόσο περισσότερες μονάδες έχουν πρόσβαση σε συγκεκριμένα δεδομένα και τόσο πιθανότερο είναι τα δεδομένα να καταστραφούν.
4. Όσο περισσότερες μονάδες διαπερνά κάποιο δεδομένο, τόσο περισσότερες μονάδες θα πρέπει να τροποποιηθούν ανάλογα σε περίπτωση που το δεδομένο αυτό τροποποιηθεί.

Η σύζευξη των μονάδων μπορεί να παρατηρηθεί από το διάγραμμα δομής, και, αν και θα πρέπει προφανώς να υπάρχει κάποια διακίνηση δεδομένων ώστε το σύστημα να κάνει κάτι, η χαμηλή σύζευξη μεταξύ των μονάδων είναι επιθυμητή.

Υπάρχουν δύο σχολές σκέψης σχετικά με το αν είναι καλύτερο τα δεδομένα να μεταβιβάζονται μέσω παραμέτρων ή διαμοιραζόμενων περιοχών. Το πέρασμα παραμέτρων είναι πολύ πιο ασφαλές. Παρόλα αυτά, αν οι μονάδες μεταβιβάζουν μεγάλο πλήθος παραμέτρων, είναι επίσης εύκολο να γίνουν σφάλματα στις κλήσεις των μονάδων. Αν χρησιμοποιούνται περιοχές διαμοιραζόμενων δεδομένων, θα πρέπει να χρησιμοποιείται κάποια μέθοδος εξασφάλισης των δεδομένων και την οργάνωσή τους στη διαμοιραζόμενη περιοχή με κατανοητό τρόπο.

Οι σύνδεσμοι ελέγχου είναι γενικά λιγότερο επιθυμητοί από τους συνδέσμους δεδομένων. Οι σύνδεσμοι ελέγχου χωρίζονται σε δύο κατηγορίες:

1. Αυτούς που δίνουν πληροφορίες κατάστασης.
2. Αυτούς που δίνουν εντολές.

Οι σύνδεσμοι ελέγχου που δίνουν πληροφορίες κατάστασης είναι δικαιολογημένοι. Παρόλα αυτά, αυτοί που δίνουν εντολές είναι συνήθως ένδειξη ότι το πρόγραμμα δεν είναι σωστά δομημένο.

Εντολές που κινούνται προς τα πάνω στο διάγραμμα δομής υποδηλώνουν ότι η μονάδα ελέγχου πληροφορείται τι να κάνει από τις μονάδες των κατώτερων επιπέδων. Αυτό υποδηλώνει πως υπάρχει υπερβολικός έλεγχος στα κατώτερα επίπεδα του διαγράμματος δομής, έτσι ώστε αυτό να είναι ζυγισμένο, και συχνά προκύπτει ως αποτέλεσμα μιας «διάσπασης απόφασης» (decision split). Στην περίπτωση αυτή, μια απόφαση λαμβάνεται σε κάποια μονάδα, αλλά τα αποτελέσματα της απόφασης υλοποιούνται σε κάποια άλλη μονάδα, όπως παρουσιάζεται στο σχήμα 13.15.

Οι διασπάσεις ελέγχου είναι μη επιθυμητές καταστάσεις, αφού κάνουν το διάγραμμα δομής και τον κώδικα δύσκολα κατανοήσιμο.

Η έκταση ελέγχου (span of control) μιας μονάδας είναι η περιοχή μονάδων που η συγκεκριμένη μονάδα ελέγχει άμεσα. Η έκταση αποτελέσματος (span of effect) μιας μονάδας είναι η περιοχή μονάδων που η συγκεκριμένη μονάδα ελέγχει έμμεσα, λαμβάνοντας αποφάσεις οι οποίες επηρεάζουν τις μονάδες αυτές.

Όταν η έκταση αποτελέσματος είναι μεγαλύτερη από την έκταση ελέγχου, κάτι τέτοιο υποδηλώνει κάποια διάσπαση απόφασης και κατά συνέπεια είναι επιθυμητό η έκταση αποτελέσματος να μην είναι μεγαλύτερος από την έκταση ελέγχου.

### 13.5.3 Ελέγχοντας την υλοποίηση

Η συνεκτικότητα και η σύζευξη είναι θεωρητικά μέτρα της δομής ενός συστήματος και αποτελούν τεχνικούς όρους, οι οποίοι περιγράφουν την ιδέα του «διαμερισμού για την ελαχιστοποίηση των διεπαφών» που εξετάζεται από την αρχή της φάσης της ανάλυσης. Σε

πραγματικό κώδικα θα πρέπει να εξεταστεί επιπλέον το πραγματικό μέγεθος των μονάδων. Αν οι μονάδες περιλαμβάνουν υπερβολικά μεγάλο όγκο κώδικα, γίνονται πολύπλοκες και δύσκολα κατανοητές. Επίσης, καθώς οι κλήσεις διαδικασιών αντιπροσωπεύουν μια επιπλέον επιβάρυνση (overhead) στο χρόνο επεξεργασίας, πολλές μικρές μονάδες μπορεί να δίνουν μια αδικαιολόγητα αργή υλοποίηση.

### Μονάδα 1

```

1 x := largest
2 y := smallest
3 total := largest + smallest
4 medium := total DIV 2
5 if value > medium
6 then over := over + 1
7 else under := under + 1

```

### Μονάδα 2

```

1 if x > 5
2 then for z := 1 to x
3     a := lookup_table(z) - 42
4 else if y < 10 or flag = false
5     then c := x mod 5
6 else if flag = true
7     then c := 0

```

Αν και περιέχουν το ίδιο πλήθος γραμμών κώδικα, η μονάδα 1 είναι πολύ ευκολότερη στην κατανόηση από τη μονάδα 2.

Η έννοια του fan-out βοηθά στη μέτρηση της πολυπλοκότητας της μονάδας, καθορίζοντας το πλήθος των μονάδων χαμηλότερου επιπέδου τις οποίες η συγκεκριμένη μονάδα καλεί. Στο διάγραμμα δομής του συστήματος συσκευασίας φακέλων τσαγιού, που παρουσιάζεται στο σχήμα 13.1, το fan-out της μονάδας «παρακολούθησε φακέλους τσαγιού» είναι ίσο με 3. Για την πλήρη κατανόηση του τι ακριβώς κάνει η μονάδα, θα πρέπει να εξεταστούν οι υφιστάμενες μονάδες. Σε προηγούμενη ενότητα, το όριο της ανθρώπινης αντίληψης ορίστηκε επτά. Αν μια μονάδα καλεί περισσότερες από επτά μονάδες κατώτερων επιπέδων, δεν μπορεί να γίνει ταυτόχρονα αντιληπτό οτιδήποτε κάνει η μονάδα αυτή - συνεπώς δεν είναι απόλυτα κατανοητή. Μια μονάδα η οποία καλεί περισσότερες από επτά κατώτερες μονάδες μπορεί επίσης να είναι υπερβολικά μεγάλη και πολύπλοκη. Αν μια μονάδα είναι υπερβολικά μεγάλη ή υπερβολικά πολύπλοκη, θα πρέπει να διασπαστεί σύμφωνα με την από πάνω προς τα κάτω αποσύνθεση (top-down decomposition) - μερικές φορές καλείται παραγοντοποίηση (factoring).

Μονάδες με μικρότερες λειτουργικές περιοχές έχουν επίσης την τάση να είναι επαναχρησιμοποιήσιμες.

## 13.6 ΠΕΡΙΛΗΨΗ

Εφ' όσον το διάγραμμα δομής χρησιμοποιείται για τη δημιουργία του κώδικα:

1. Θα πρέπει να είναι καλά δομημένο
2. Θα πρέπει να περιγράφει μια λογική υλοποίηση.

Μερικές φορές θα πρέπει να γίνεται κάποιος συμβιβασμός μεταξύ του θεωρητικού και του πρακτικού όταν κατασκευάζεται το διάγραμμα δομής. Κάθε δομή, όμως, που καταστρέφεται στο όνομα της αποδοτικότητας θα στοιχίσει αργότερα στη συντήρηση και την ευκολία κατανόησης.

Αν και ο μη δομημένος κώδικας δεν είναι αναγκαστικά πιο γρήγορος και πιο μικρός σε όγκο από τον σωστά δομημένο κώδικα, ο σωστά δομημένος κώδικας έχει την τάση να είναι ογκωδέστερος και πιο αργός από ότι χρειάζεται να είναι, αφού δεν περιλαμβάνει έξυπνα τρυκ και περισσότερο κώδικα για να βοηθήσει στην κατανόηση του τι ακριβώς συμβαίνει. Αυτό σημαίνει ότι στην πραγματικότητα είναι ευκολότερη η βελτιστοποίηση του δομημένου κώδικα, σε σχέση με τον μη δομημένο, αφού υπάρχουν περισσότερο προφανή σημεία που ενδείκνυται για βελτιστοποίηση. Κατά συνέπεια, θα πρέπει πάντοτε η αρχή να γίνεται με δομημένο κώδικα και στη συνέχεια αν κρίνεται υπερβολικά αργός ή μεγάλος σε μέγεθος, να γίνεται βελτιστοποίησή του.

## ΠΑΡΑΡΤΗΜΑ

Το παραπάνω κείμενο αποτελεί συνοπτική παρουσίαση του κεφαλαίου 13 και ταυτόχρονα μια πλήρης καταγραφή της σχετικής παρουσίασης, με εκτενή σχόλια στα κυριότερα σημεία. Επειδή η προσοχή επικεντρώθηκε στην ίδια την παρουσίαση του θέματος και λιγότερο στην ακριβής δομή του σχετικού (μεταφρασμένου από το πρωτότυπο) κειμένου, χρησιμοποιήθηκε ορολογία και έννοιες από σχετική βιβλιογραφία, η οποία αναφέρεται παρακάτω.

Το κείμενο αυτό, μαζί με το πλήρες κείμενο (μετάφραση πρωτότυπου) και τις διαφάνειες που χρησιμοποιήθηκαν στην παρουσίαση, υπάρχει σε ηλεκτρονική μορφή (αρχεία MS-Word 7.0) στην παρακάτω διεύθυνση (WWW site):

<a href="http://www.di.uoa.gr/~grad0177/rts_ch13">http://www.di.uoa.gr/~grad0177/rts_ch13</a>	
• rts_ch13.doc	Πλήρες κείμενο (μετάφραση πρωτότυπου) του κεφαλαίου 13.
• rts_dist.doc	Συνοπτική αναφορά & σχολιασμός παρουσίασης (το παρόν κείμενο)
• rts_pres.doc	Διαφάνειες παρουσίασης

## ΒΙΒΛΙΟΓΡΑΦΙΑ

- [1] «Τεχνολογία Λογισμικού - Τόμος Β'»  
Εμμ. Α. Γιακουμάκης - Εκδ. Σταμούλης, 1994.
- [2] «Τεχνολογία Λογισμικού - Τόμος Α'»

- Εμμ. Α. Γιακουμάκης - Εκδ. Σταμούλης, 1993.
- [3] «Τεχνικές για την Ανάλυση και Σχεδίαση Συστημάτων»
- Ευγενία Μανωλοπούλου - Εκδ. Anubis, 1994.