Δοκιμασία και πλάνο δοκιμασίας

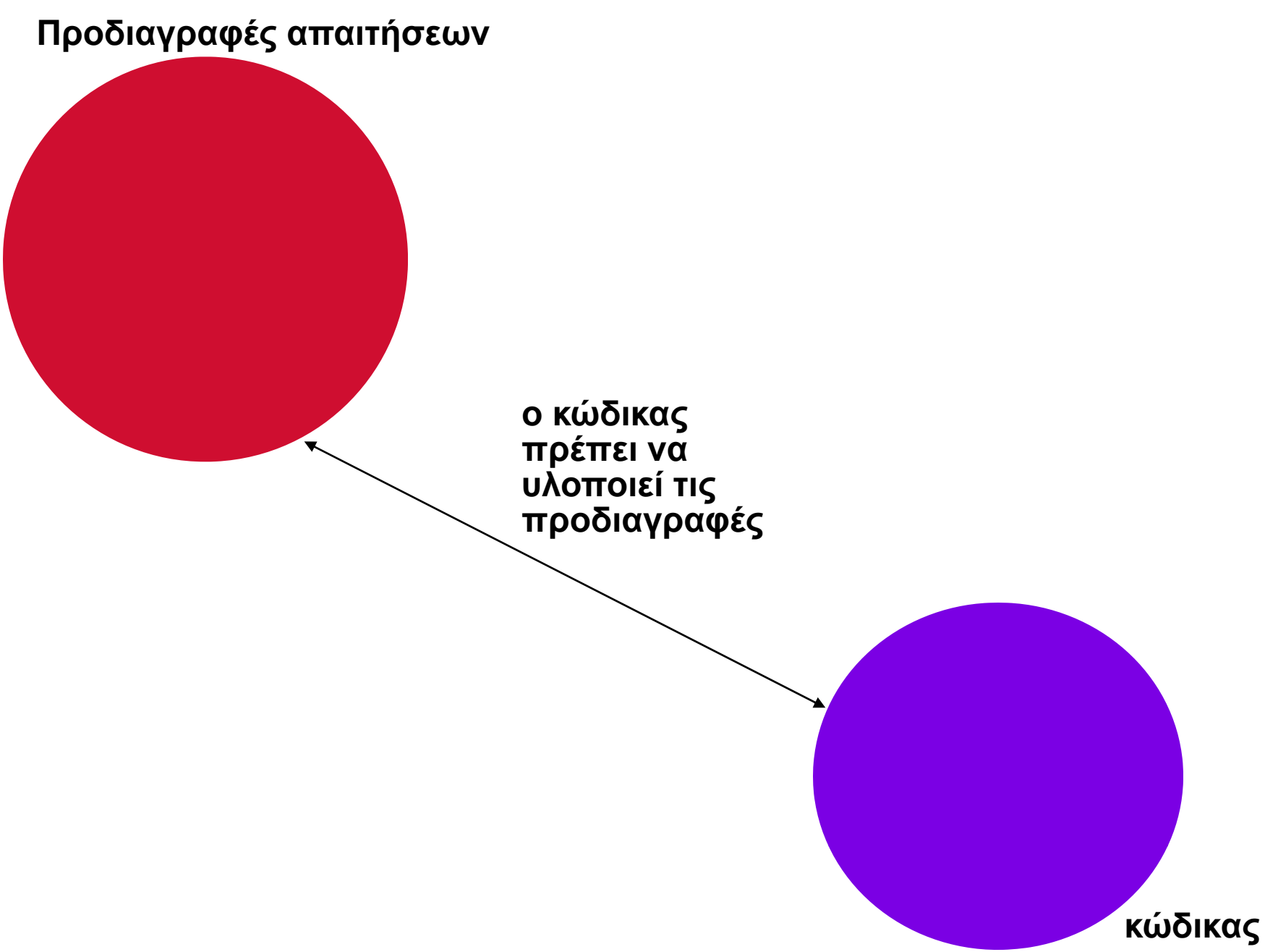
Γιάννης Σμαραγδάκης

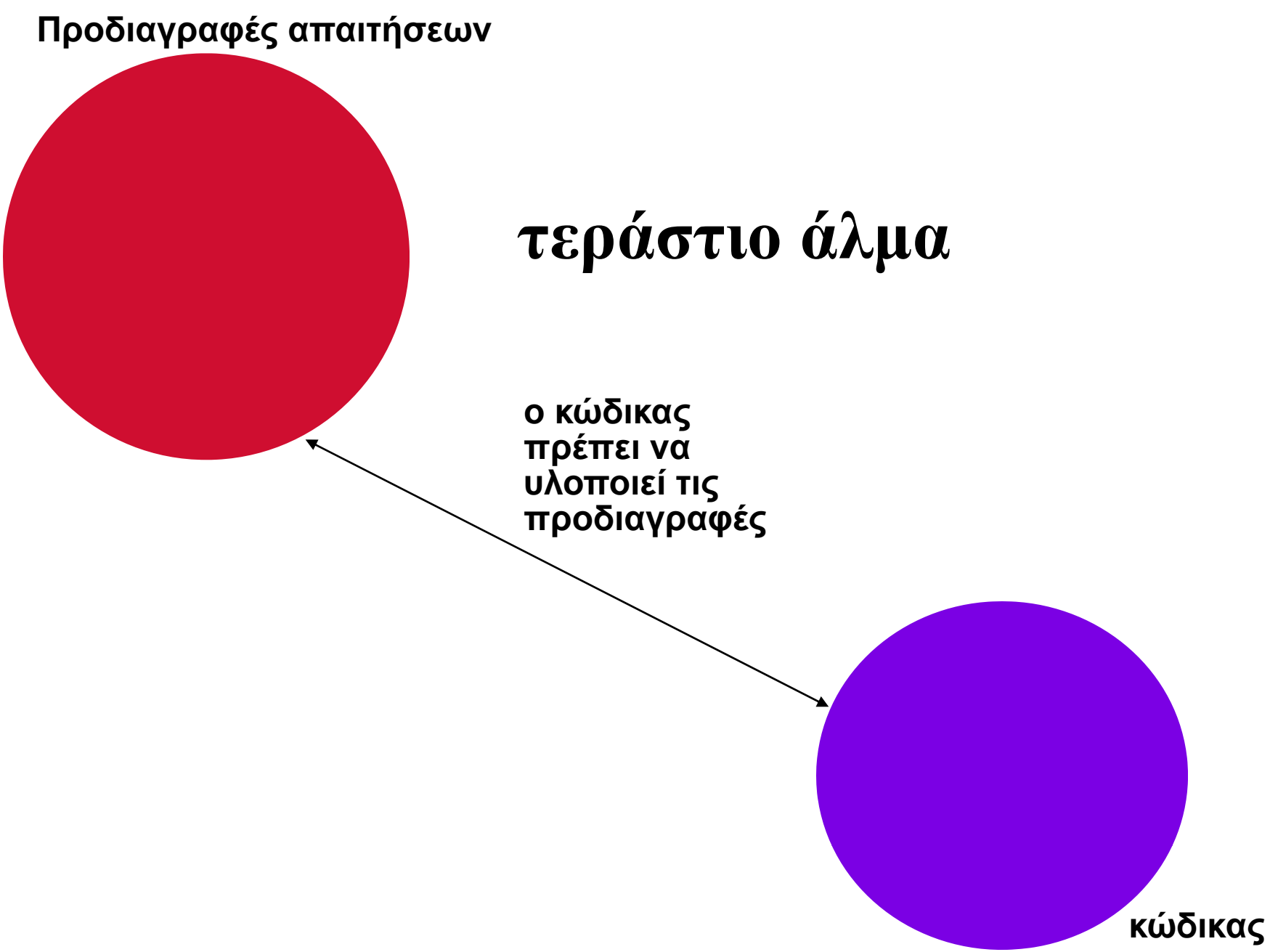
Η δοκιμασία επιχειρεί να απαντήσει

- **Κάνει το λογισμικό αυτό που υποτίθεται;**
- **Πότε μπορεί να έχει πρόβλημα;**
- **Πόσο γρήγορα τρέχει;**
- **Πόσο ακριβή είναι τα αποτελέσματα;**
- **Όταν έχει πρόβλημα, τι κάνει;**
- **Σε τι μπορώ να στηρίζομαι;**
- **Ποια είναι τα δυνατά/αδύναμα του στοιχεία;**

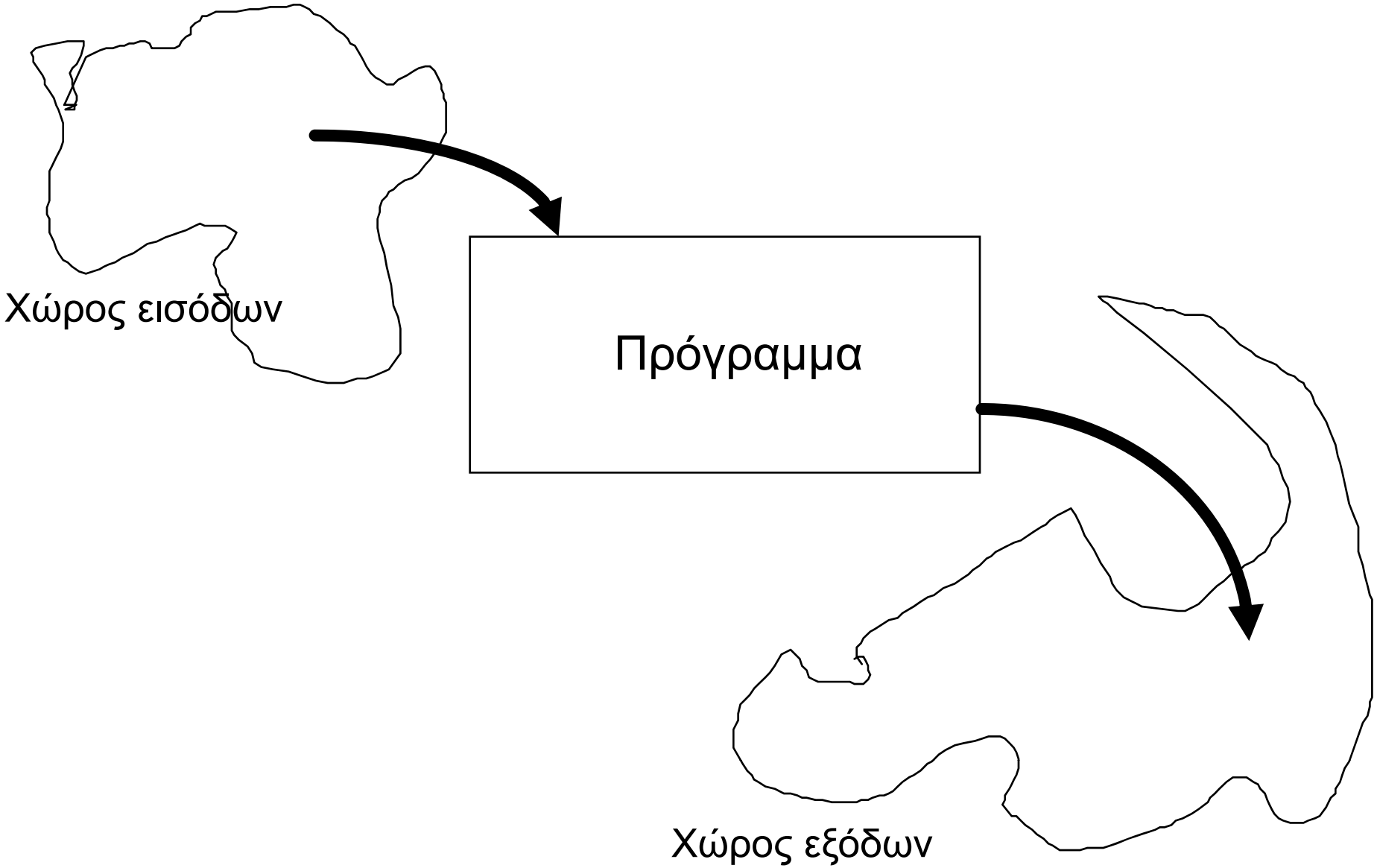
Βασικοί ορισμοί

- **Αστοχία συστήματος (failure): ασυνέπεια μεταξύ συμπεριφοράς και προδιαγραφής**
- **Σφάλμα (error): διαφορά στην εσωτερική κατάσταση του προγράμματος από την αναμενόμενη. Μπορεί να οδηγήσει σε αστοχία ή όχι**
- **Ελάττωμα (fault): αιτία πιθανού σφάλματος μέσα στο λογισμικό**
- **Δοκιμασία: συστηματική (;) έρευνα στο χώρο εισόδων του προγράμματος σε αναζήτηση αστοχίας**
- **Debugging: ή ερεύνα για το ελάττωμα που προκάλεσε την αστοχία**

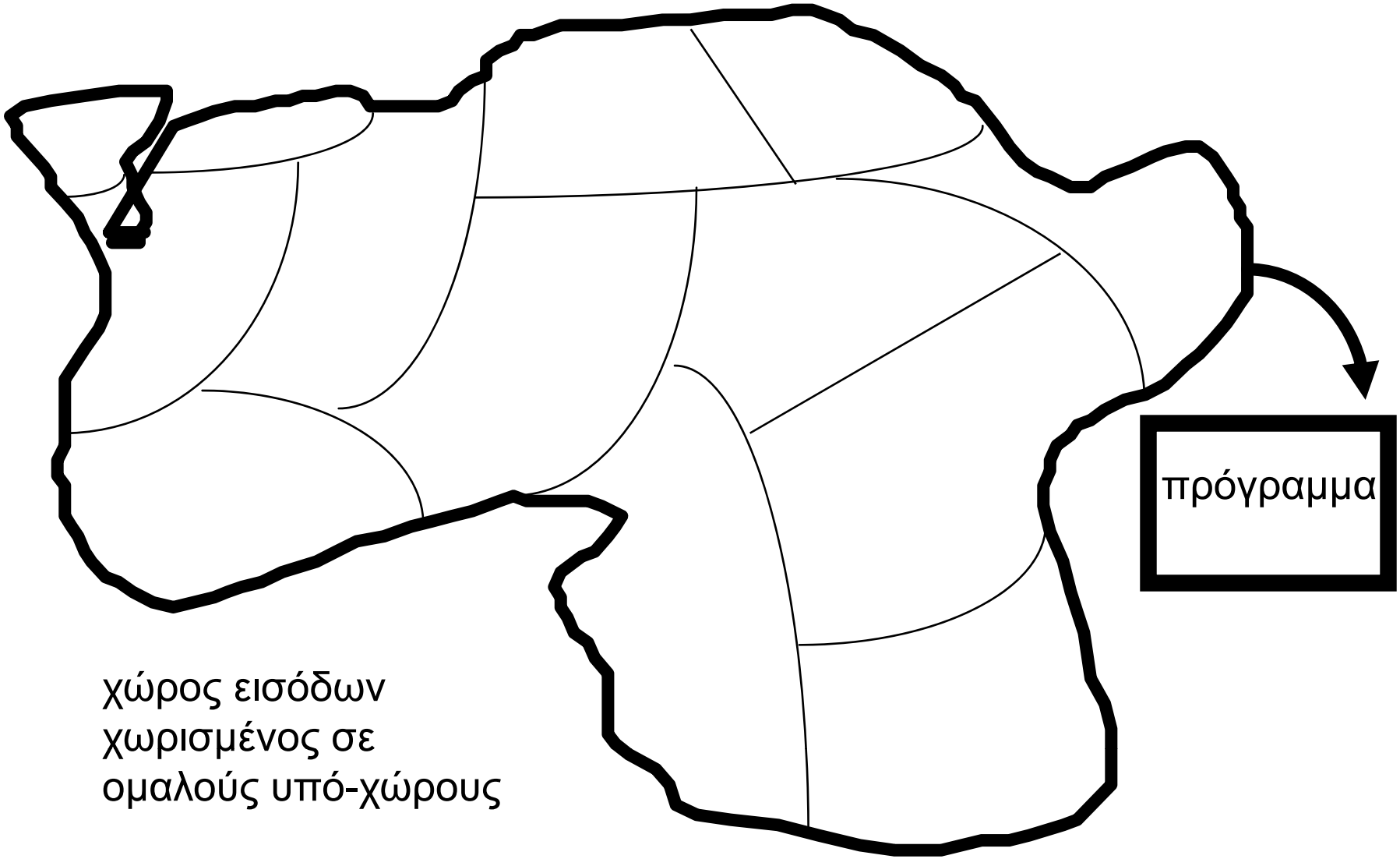

Προδιαγραφές απαιτήσεων
ο κώδικας πρέπει να υλοποιεί τις προδιαγραφές
κώδικας


Προδιαγραφές απαιτήσεων
τεράστιο άλμα
ο κώδικας
πρέπει να
υλοποιεί τις
προδιαγραφές
κώδικας

Απλοϊκή θεώρηση



Πιο ρεαλιστική θεώρηση



Η δοκιμασία είναι δειγματοληψία του χώρου εισόδων

- Πρόβλημα: ποιος είναι ο χώρος εισόδων;
  - και ποιες οι επιθυμητές έξοδοι;
- Υποπρόβλημα: Ο χώρος εισόδων είναι τεράστιος
  - μια διάσταση για κάθε είσοδο
  - κάθε διάσταση έχει πολλά στοιχεία (π.χ. 2**32 ακεραίους)
  - Πόσο διαφορετικά; Ποιες διαφορές έχουν σημασία;
- Βασικό πρόβλημα: πώς να πάρουμε δείγματα από το χώρο εισόδων;
- Πότε σταματάμε; Είναι καλό ή κακό να βρίσκουμε ολοένα και άλλα ελαττώματα;

Ποιος είναι ο χώρος εισόδων; Πόσες είσοδοι παράγουν αστοχία;

Προδιαγραφή

Το sum_of_roots παίρνει μια αυθαίρετα μεγάλη ακολουθία πραγματικών αριθμών και υπολογίζει το άθροισμα των τετραγωνικών τους ριζών. Η ακολουθία τελειώνει με τον αριθμό 9999.99

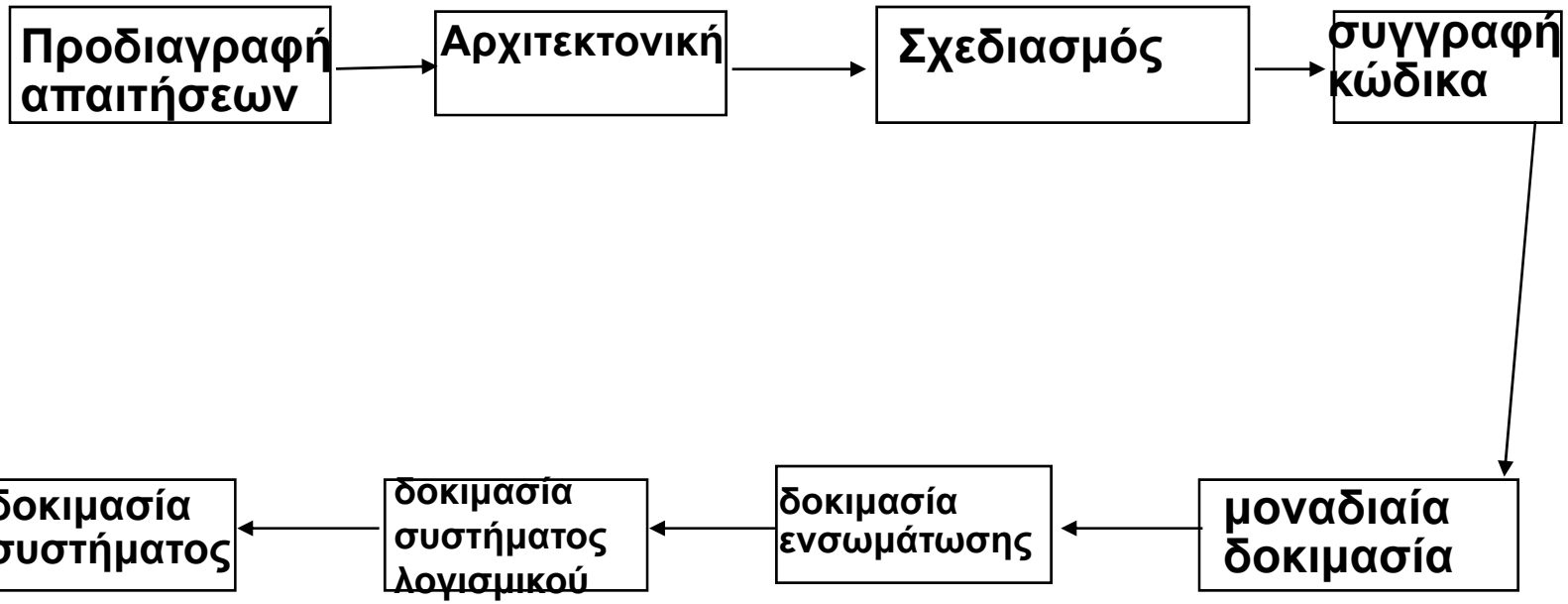
Υλοποίηση

```
Program sum_of_roots;
Real sum, x, r;
sum := 0;
Do forever
        input x;
        if x = 9999.99 then exit
                else
                r := sqrt(x);
                sum := sum + r;
        end do;
print sum;
end;
```

Φάσεις δοκιμασίας

- **Μοναδιαία δοκιμασία/δοκιμασία τμημάτων (unit/module testing)**
  - **σύγκριση με το σχεδιασμό του τμήματος**
- **Δοκιμασία ενσωμάτωσης (integration testing)**
  - **δοκιμασία του συνδυασμού τμημάτων ώστε να επιβεβαιώσουμε τη συνέπειά του**
- **Δοκιμασία συστήματος λογισμικού (software system testing)**
  - **σύγκριση ενός ολοκληρωμένου συστήματος λογισμικού με τις προδιαγραφές απαιτήσεων**
- **Δοκιμασία συστήματος (system testing)**
  - **εκτίμηση ενός ολοκληρωμένου συστήματος λογισμικού και υλικού**

Φάσεις ανάπτυξης

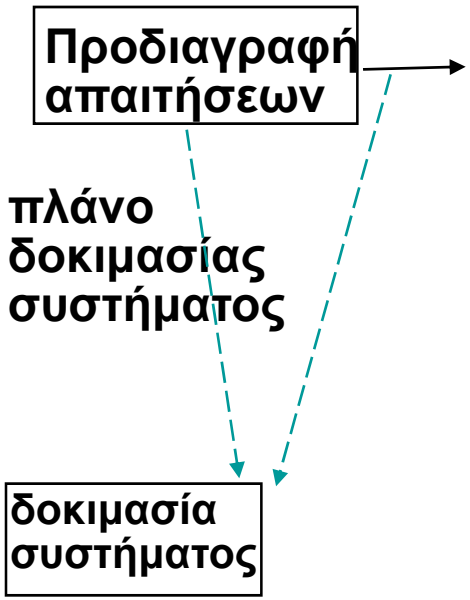


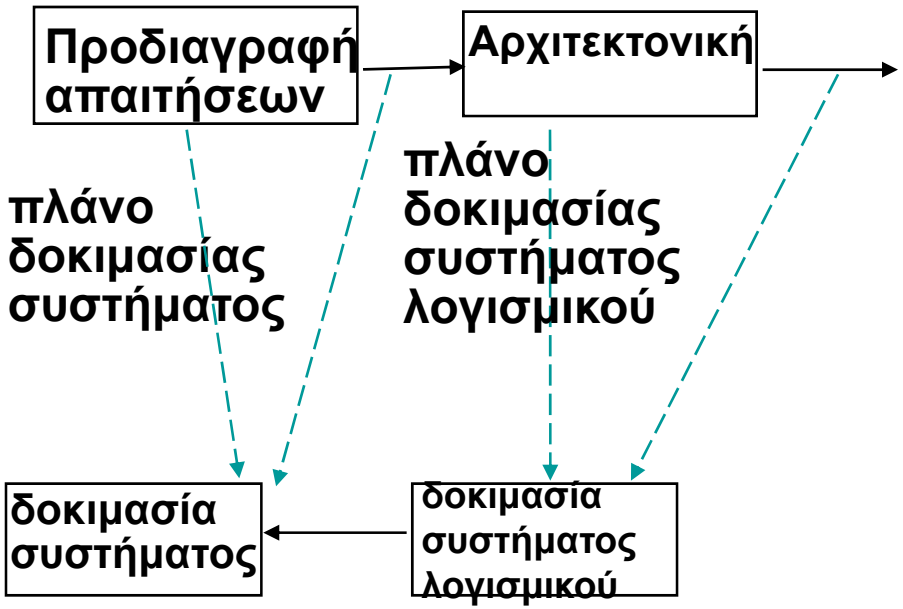
Φάσεις δοκιμασίας

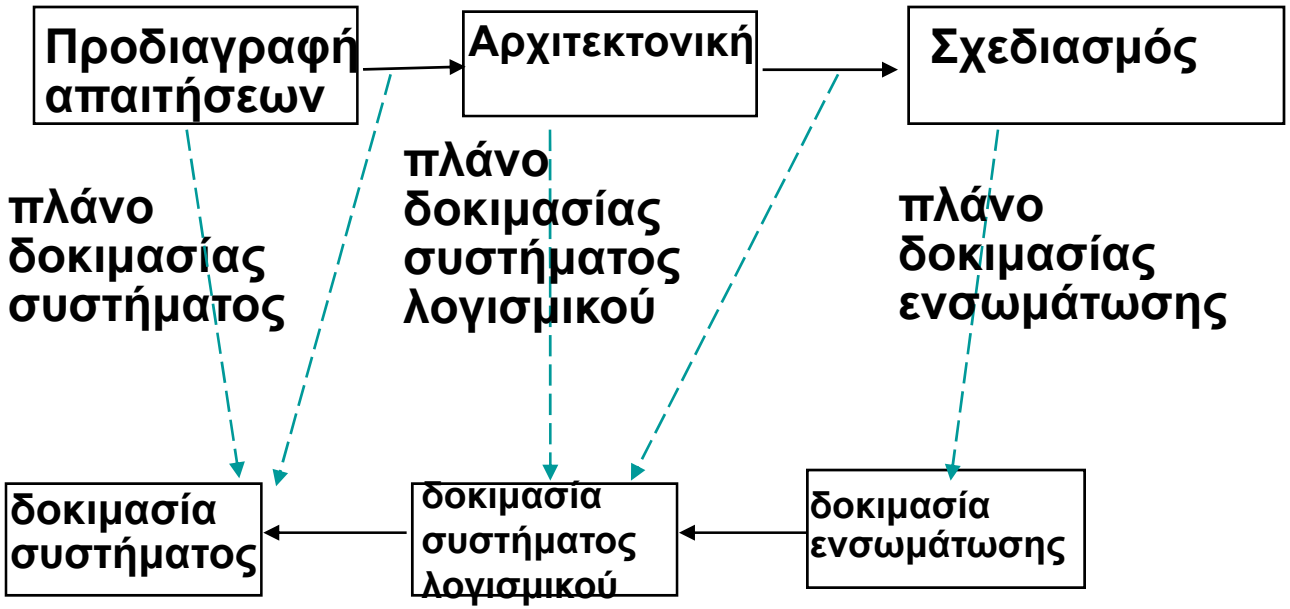
Φτιάξτε πλάνα δοκιμασίας το συντομότερο δυνατόν

- Οι επιθυμητές συμπεριφορές πρέπει να ορίζονται από τις προδιαγραφές και όχι από τον κώδικα
- Η δοκιμασία βοηθάει να αποσαφηνίσουμε τις προδιαγραφές
  - και αργότερα να βρούμε εύκολα τα σφάλματα στον κώδικα

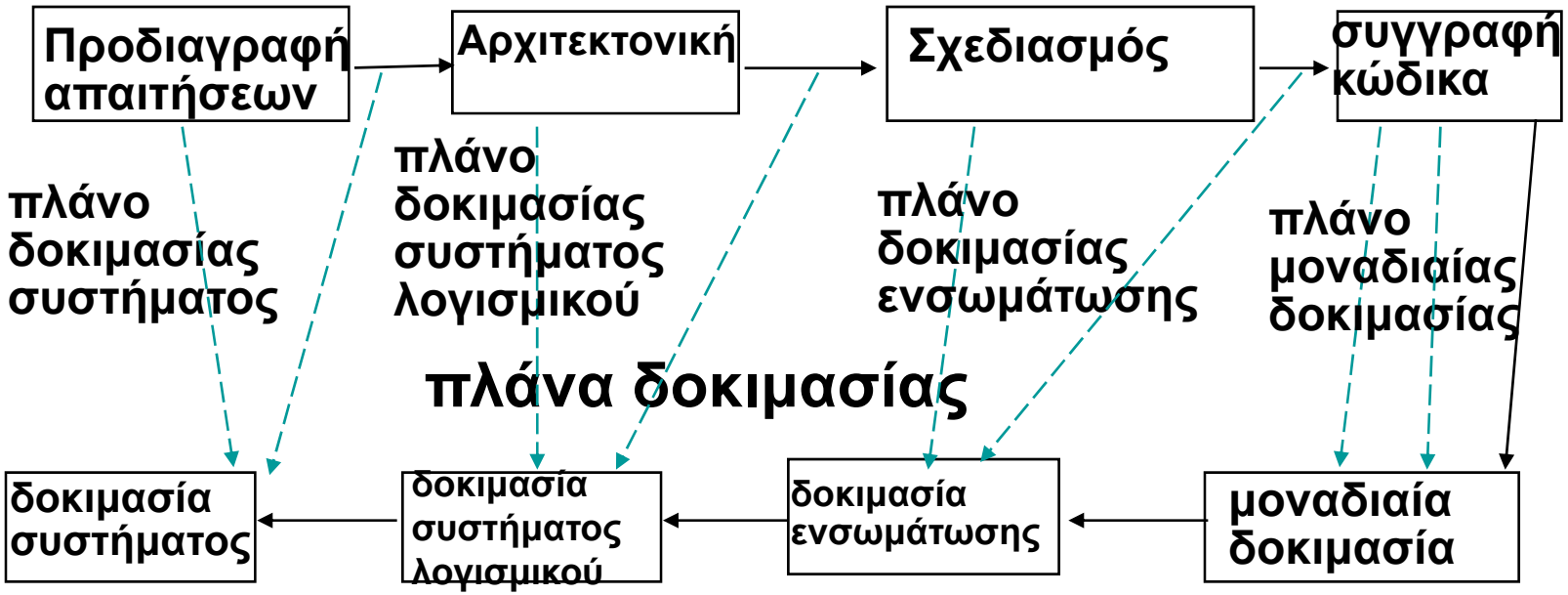
Ξεκινάμε με:


Προδιαγραφή απαιτήσεων
πλάνο δοκιμασίας συστήματος
δοκιμασία συστήματος


Προδιαγραφή απαιτήσεων
Αρχιτεκτονική
πλάνο δοκιμασίας συστήματος
πλάνο δοκιμασίας συστήματος λογισμικού
δοκιμασία συστήματος
δοκιμασία συστήματος λογισμικού


Προδιαγραφή απαιτήσεων
Αρχιτεκτονική
Σχεδιασμός
πλάνο δοκιμασίας συστήματος
πλάνο δοκιμασίας συστήματος λογισμικού
πλάνο δοκιμασίας ενσωμάτωσης
δοκιμασία συστήματος
δοκιμασία συστήματος λογισμικού
δοκιμασία ενσωμάτωσης

Φάσεις ανάπτυξης



Φάσεις δοκιμασίας

Η δοκιμασία μας δίνει γνώση με αντίτιμο κόστος

- Χρειαζόμαστε προσπάθεια ανάπτυξης δοκιμασιών, χρόνο εκτέλεσης και εξέτασής τους
- Αξίζει η γνώση το κόστος;
- Το πλάνο δοκιμασίας είναι κι αυτό μέρος του λογισμικού
- Αναπτύσεται και εκτιμάται με τον ίδιο τρόπο όπως το τελικό πρόγραμμα

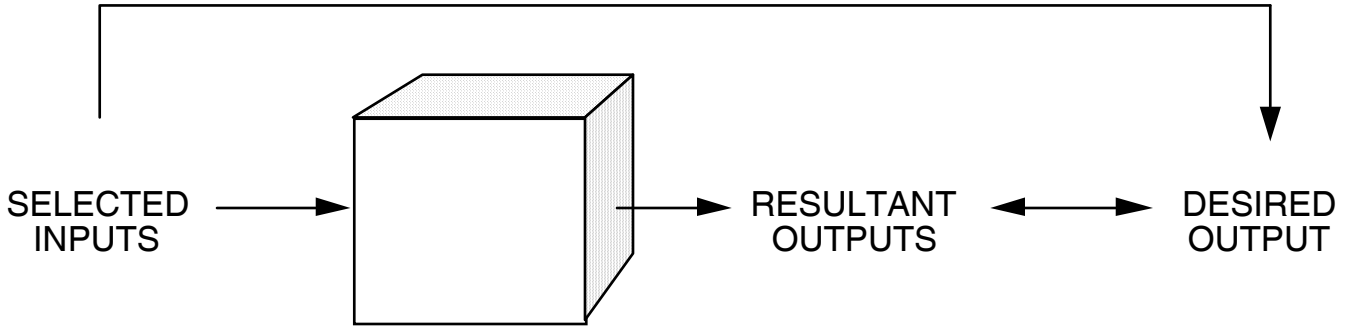
Ανατομία πλάνου δοκιμασίας

- Συλλογή (πιθανόν ιεραρχική) από στοιχεία δοκιμασίας
- Στοιχεία δοκιμασίας (προδιαγραφή-υλοποίηση test cases)
  - στόχοι, απαιτήσεις
  - τι δοκιμάζεται
  - τι πόροι χρειάζονται για τη δοκιμασία (π.χ. βάσεις δεδομένων, χρήστες, κλπ.)
  - πώς στήνεται η δοκιμασία
  - δεδομένα εισόδου
  - αναμενόμενα αποτελέσματα
  - χειρισμός λαθών
  - ...
- Παράδειγμα: ας φτιάξουμε πλάνο για τη συνάρτηση SQRT (περιπτώσεις, στήσιμο, αποτελέσματα, κτλ.)
  - πιο δύσκολο: για σενάριο επιλογής μαθημάτων

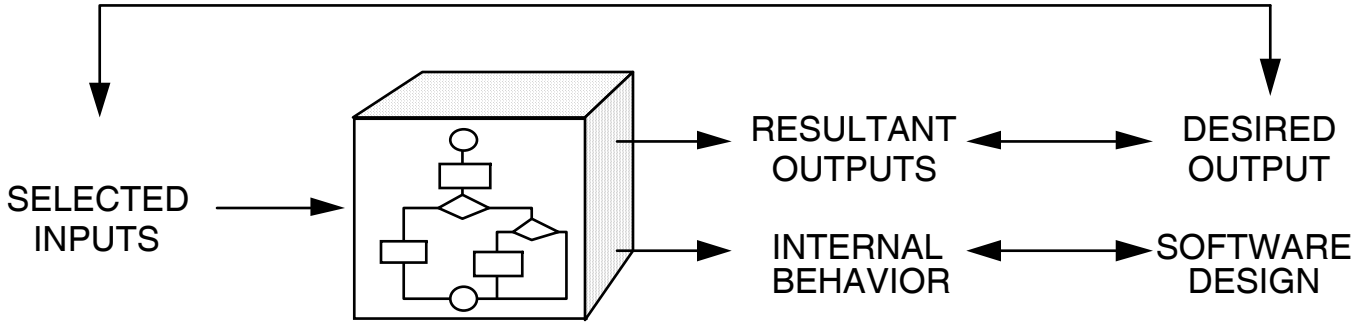
Διαφανής/αδιαφανής δοκιμασία (white-/black-box testing)

- Αδιαφανής δοκιμασία
  - καμμία γνώση εσωτερικών χαρακτηριστικών της υλοποίησης
  - η δοκιμασία βασίζεται στην παρατήρηση εξωτερικής συμπεριφοράς
  - ταιριάζει σε χρήστες, πελάτες, κτλ.
- Διαφανής δοκιμασία
  - χρησιμοποιεί γνώση της δομής του προγράμματος
  - εξετάζει την εσωτερική κατάσταση
  - συχνά χρησιμοποιείται από προγραμματιστές

Διαφανής/αδιαφανής δοκιμασία (white-/black-box testing)



“BLACK BOX” TESTING



“WHITE BOX” TESTING
(CLEAR BOX TESTING)

Ανάλυση

Η δοκιμασία μπορεί να τεκμηριώσει μόνο την ύπαρξη ελαττωμάτων, όχι την απουσία τους

- **Η στατική ανάλυση δεν χρειάζεται εκτέλεση**
  - **μαθηματικό μοντέλο του λογισμικού**
  - **επεξεργασία του με τυπικές μεθόδους**
  - **μπορεί να δείξει την απουσία λαθών**
- **Λύνει τα προβλήματα της δοκιμασίας**
  - **δεν χρειάζεται να κάνουμε δειγματοληψία**
  - **δεν χρειάζεται να ερμηνεύσουμε αποτελέσματα ή να αποφασίσουμε πότε να σταματήσουμε**
- **Περιορισμοί**
  - **τα θεωρήματα που θέλουμε να αποδείξουμε είναι συνήθως πολύ δύσκολα**

Επαλήθευση και Επιβεβαίωση (verification and validation)