

# Τεχνολογία Λογισμικού

## Έλεγχος

Μιχάλης Ξένος, Καθηγητής



Μιχάλης Ξένος, 2025

1



2

Διαδικαστικά  
Μαθήματος

2




3

TMHYΠ  
ΤΕΧΝΙΚΗ ΠΑΝΕΠΙΣΤΗΜΙΟΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

Μιχάλης Ξένος, 2025

3

# eClass

Γραφτείτε άμεσα στο μάθημα

password: **softeng202425**

Θα δίνουμε το password κάθε βδομάδα μέχρι να μην έχει πλέον νόημα!

4

TMHYΠ  
ΤΕΧΝΙΚΗ ΠΑΝΕΠΙΣΤΗΜΙΟΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

Μιχάλης Ξένος, 2025

4



5

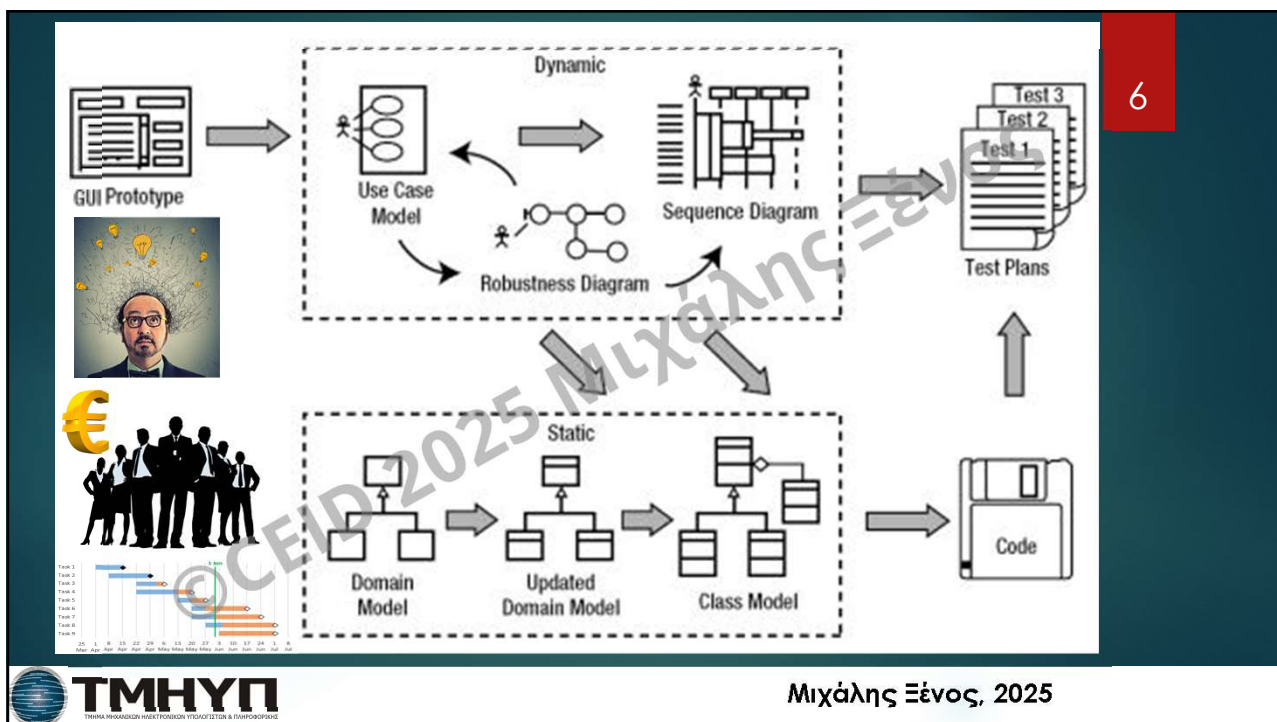
# Επανάληψη

©CEID 2025 Μιχάλης Ξένος

**ΤΜΗΥΠ**  
ΤΕΧΝΙΚΗ ΠΡΟΒΛΕΨΗ ΚΑΙ ΕΚΤΙΜΗΣΗ ΤΕΧΝΟΛΟΓΙΑΣ ΚΑΙ ΠΡΟΓΡΑΜΜΑΤΩΝ

Μιχάλης Ξένος, 2025

5



6

## Στις πρώτες σας δουλειές ως Junior Software Engineer

7

- ▶ Η εμπειρία μετράει
  - ▶ Εντός πλαισίου μαθημάτων
  - ▶ Και κυρίως εκτός πλαισίου μαθημάτων
  - ▶ (π.χ. αυτό είναι το project στο software engineering στο GitHub)
- ▶ Η γλώσσες που έχετε εμπειρία όχι και τόσο!
- ▶ Η βασική γνώση αρχών Τεχνολογίας Λογισμικού για να ενταχθείτε άμεσα σε ομάδες μετράει και μάλιστα πολύ!
- ▶ Η ικανότητα εργασίας σε ομάδες μετράει πολύ!

## Υλοποίηση σε ομάδες...

8

- ▶ Καθοδήγηση των Junior Engineers από Senior Engineers
  - ▶ Coaching
  - ▶ Seminars / Training
- ▶ Διαδικασίες για version control και team working
  - ▶ Code repository
  - ▶ Version control
  - ▶ Revert changes
- ▶ Σίγουρα code reviews
- ▶ Πιθανότατα code inspections



## Επαγγελματική και ηθική ευθύνη

9

- ▶ Η τεχνολογία λογισμικού δεν αφορά μόνο την εφαρμογή τεχνικών δεξιοτήτων, αλλά ενέχει και ευρύτερες ευθύνες.
- ▶ Οι μηχανικοί λογισμικού πρέπει επίσης να συμπεριφέρονται με τρόπο **δεοντολογικό και ηθικά υπεύθυνο** προκειμένου να γίνουν σεβαστοί ως επαγγελματίες.
- ▶ Η ηθική υπευθυνότητα υπερβαίνει τα στενά όρια της έννοιας της τήρησης της νομοθεσίας.

## Δεοντολογικά διλήμματα

10

### Ας προβληματιστούμε...

- ▶ Διαφωνία επί της αρχής με τις πολιτικές της διεύθυνσης της εταιρείας.
- ▶ Π.χ. ο εργοδότης σας ενεργεί με αντιδεοντολογικό τρόπο και θέτει σε κυκλοφορία ένα σύστημα κρίσιμο ως προς την ασφάλεια χωρίς να ολοκληρώσει τις δοκιμές του.

## Ας συνεχίσουμε με τη διάλεξη...

11



11

## Καλύτερα Έλεγχος (και όχι δοκιμή)

12

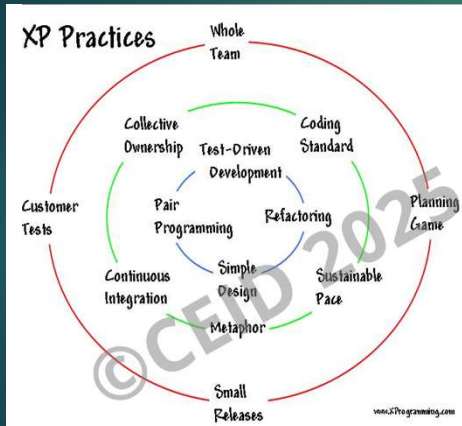


ΑΝ και στη βιβλιογραφία χρησιμοποιούνται το ίδιο (π.χ. το βιβλίο σας μιλάει για «δοκιμές»)

12

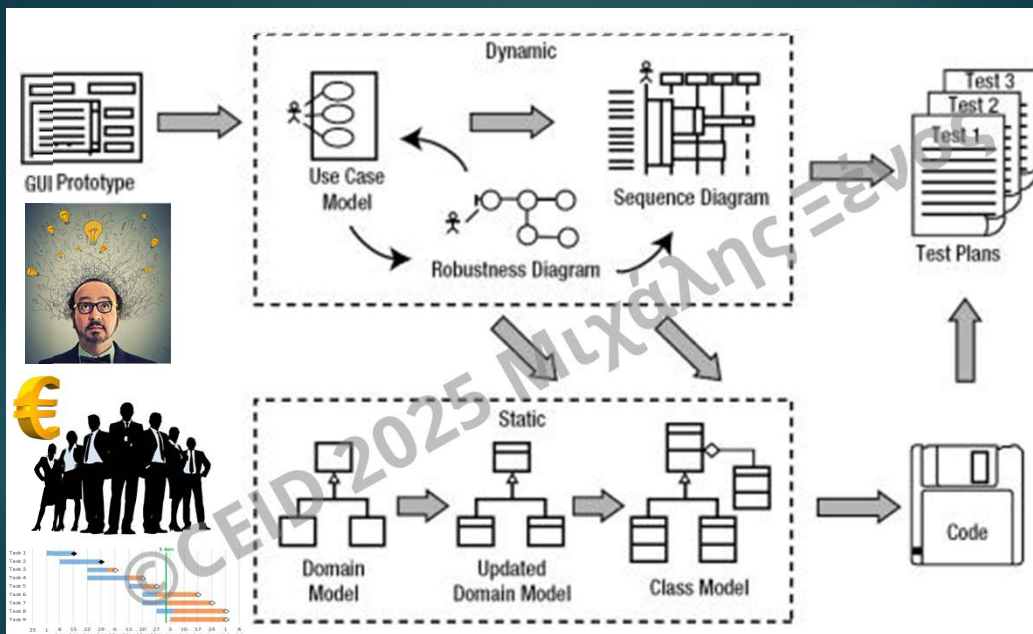
# Ο Έλεγχος στις 'μοντέρνες' μεθόδους

13



Ο έλεγχος δεν είναι πια μια QA διαδικασία, αλλά βασική πρακτική κάθε σύγχρονης μεθόδου!

13



14

14

## Εγκυροποίηση Λογισμικού

15



## Έλεγχος και Αποσφαλμάτωση

16

- ▶ Κατά τον έλεγχο εντοπίζουμε λάθη...
- ▶ Τα οποία διορθώνονται στη φάση της αποσφαλμάτωσης (debugging)
- ▶ Η διαδικασία ελέγχου είναι συστημική διαδικασία
- ▶ Η διαδικασία αποσφαλμάτωσης είναι νοητική διαδικασία



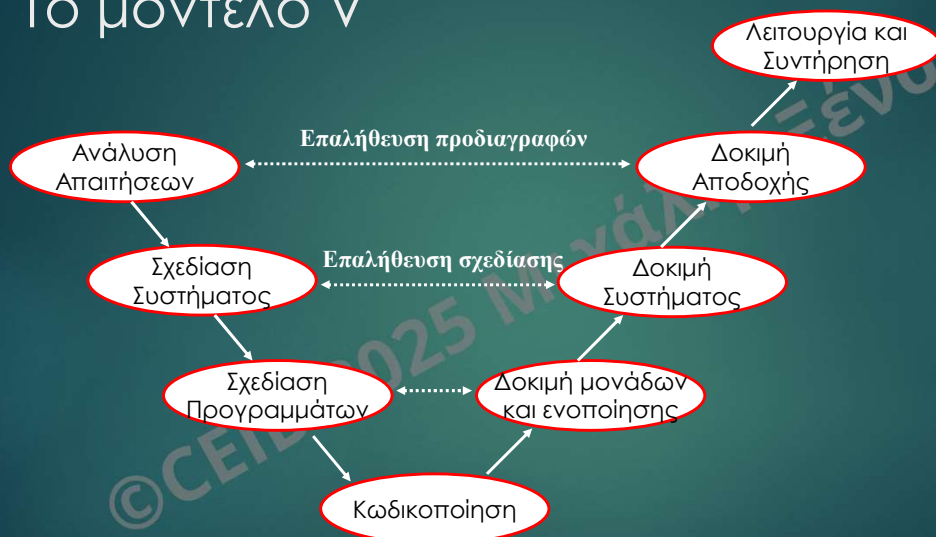
## Παράδειγμα V&V

17

- ▶ Πείτε μου τι διαδικασίες θα κάνετε μετά την υποβολή της Χ<sup>ης</sup> παράδοσης του project σας στο eClass;
- ▶ Τι είναι verification και τι validation;

## Το μοντέλο V

18



## Διαδικασίες Εγκυροποίησης

19

- ▶ Αποσκοπούν στον έγκαιρο εντοπισμό και διόρθωση των δυσλειτουργιών και ατελειών του λογισμικού.
- ▶ Πρέπει να εφαρμόζονται καθ' όλη τη διάρκεια του κύκλου ζωής.
  - ▶ Επαλήθευση: κατά τη διάρκεια κάθε φάσης του κύκλου ζωής.
  - ▶ Επικύρωση: μετά το πέρας μίας φάσης του κύκλου ζωής.
- ▶ Οι εξαντλητικοί έλεγχοι **δεν** πιστοποιούν ότι το λογισμικό δεν έχει σφάλματα ή ατέλειες.

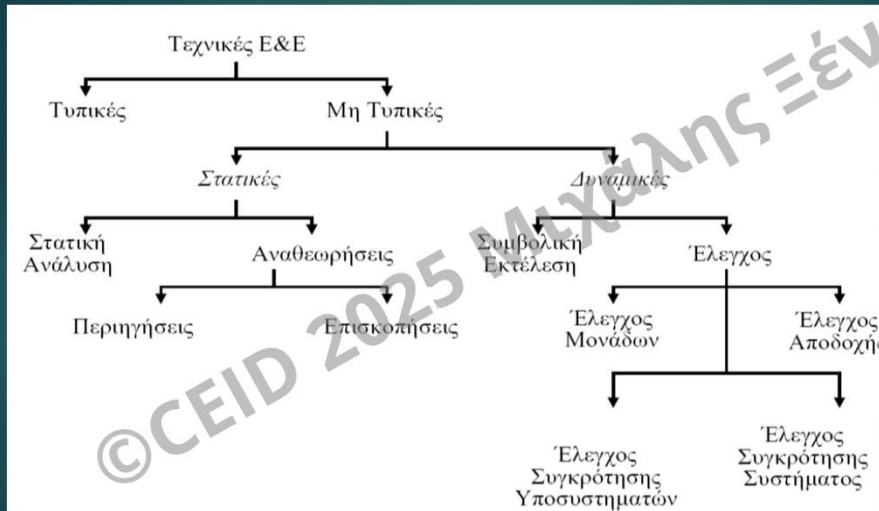
## Κατηγορίες Τεχνικών Εγκυροποίησης

20

- ▶ **Τυπικές (formal):** παράγουν μαθηματική απόδειξη ορθότητας. Πιστοποιούν ότι το λογισμικό λειτουργεί σύμφωνα με τις προδιαγραφές του.
- ▶ **Μη Τυπικές (informal):** Επικεντρώνονται στον εντοπισμό σφαλμάτων. Επαναλαμβανόμενη εφαρμογή, μέχρις ότου επιτευχθεί ικανό επίπεδο εμπιστοσύνης στο λογισμικό.
  - ▶ **Στατικές τεχνικές (static techniques):** εξέταση αναπαραστάσεων λογισμικού (προδιαγραφών, σχεδίου, κώδικα) χωρίς εκτέλεση.
  - ▶ **Δυναμικές τεχνικές (dynamic techniques):** απαιτούν την εκτέλεση του λογισμικού χρησιμοποιώντας δεδομένα εισόδου.

## Κατηγορίες Τεχνικών Εγκυροποίησης

21



## Στατικές Τεχνικές Εγκυροποίησης

22

- ▶ Είναι **φθηνότερες και ταχύτερες** από τις δυναμικές τεχνικές.
  - ▶ Συνήθως γίνονται **πριν** τον δυναμικό έλεγχο ώστε να ανακαλυφθούν όσο το δυνατό περισσότερα σφάλματα και να μειωθεί το κόστος του δυναμικού ελέγχου.
  - ▶ Δεν ελέγχουν τη δυναμική συμπεριφορά του λογισμικού (τη λειτουργία με δεδομένα εισόδου).
  - ▶ Επικρατέστερες τεχνικές:
    - ▶ Στατική Ανάλυση (Static Analysis).
    - ▶ Επιθεωρήσεις και αναθεωρήσεις (Reviews).
- ©CEID 2025 Μιχάλης Ξένος

## Στατική Ανάλυση

23

- ▶ Χρησιμοποιείται κυρίως στην επαλήθευση (verification) κώδικα.
- ▶ Συνήθως γίνεται αυτοματοποιημένα με τη χρήση **στατικών αναλυτών** (static analyzers):
  - ▶ Δέχονται ως είσοδο τον πηγαίο κώδικα.
  - ▶ Κάνουν λεκτική ανάλυση (parse) και ανακαλύπτουν πιθανά σφάλματα.
- ▶ Πολύ αποτελεσματική τεχνική, που πρέπει να χρησιμοποιείται σε συνδυασμό με την τεχνική των αναθεωρήσεων.

## Στατική Ανάλυση - Στάδια

24

- ▶ Ανάλυση Ροής Ελέγχου: βρόχοι με πολλαπλά σημεία εισόδου και εξόδου, κώδικας που δεν εκτελείται ποτέ, κλπ.
- ▶ Ανάλυση Χρήσης Δεδομένων: μεταβλητές χωρίς αρχικοποίηση, μεταβλητές που δεν χρησιμοποιούνται, μεταβλητές στις οποίες ανατίθενται δύο διαφορετικές τιμές χωρίς να χρησιμοποιούνται ενδιάμεσα, κλπ.
- ▶ Ανάλυση Διεπαφών: συνέπεια στην δήλωση και κλήση υπορουτινών (διαδικασιών και συναρτήσεων), μη χρησιμοποιούμενες υπορουτίνες, μη χρησιμοποιούμενα αποτελέσματα υπορουτινών.



## Στατική Ανάλυση - Στάδια

25

- ▶ Ανάλυση Ροής Πληροφορίας: μεταβλητές εισόδου καθώς και συνθήκες από τις οποίες εξαρτάται κάθε μεταβλητή εξόδου.
- ▶ Ανάλυση Μονοπατιού Εκτέλεσης: εντοπισμός όλων των πιθανών μονοπατιών εκτέλεσης και καταγραφή των εντολών που θα εκτελεστούν για κάθε μονοπάτι.
- ▶ Τα δύο αυτά στάδια δεν εντοπίζουν λάθη αλλά δίνουν πληροφορίες που χρησιμοποιούνται κατά τις αναθεωρήσεις.

## Στατική Ανάλυση - Χρησιμότητα

26

- ▶ Ιδιαίτερα χρήσιμη σε γλώσσες που αφήνουν σφάλματα κατά τη μεταγλώττιση – weakly typed – όπως η C.
- ▶ Δεν χρειάζεται χρήση στατικών αναλυτών σε γλώσσες που εντοπίζουν σφάλματα όπως αυτά που περιγράφηκαν – strongly typed – όπως η Java (που κάνει από μόνη της ελέγχους κατά τη μεταγλώττιση).
- ▶ **Οι στατικοί αναλυτές δεν αναγνωρίζουν λογικά λάθη** (π.χ. λάθος αρχικοποίηση, πέρασμα λανθασμένων μεταβλητών ορθού τύπου σε υπορουτίνα, κλπ.)

## Επανάληψη κεφαλαίου 22 βιβλίου

- ▶ Λύστε/απαντήστε, ή έστω προβληματιστείτε στις ασκήσεις του βιβλίου: 22.6, 22.7 και 22.9
- ▶ Αναζητήστε εναλλακτικές πηγές

## Δυναμικές Τεχνικές Εγκυροποίησης

28

- ▶ Απαιτούν την ύπαρξη του συνόλου ή τμήματος του λογισμικού προς έλεγχο.
- ▶ Εξετάζουν τη **δυναμική** συμπεριφορά του λογισμικού με δεδομένα εισόδου (πραγματικά ή τεχνητά για τις ανάγκες προσομοίωσης).
- ▶ Συνήθως περιορίζεται στην εγκυροποίηση εκδόσεων του λογισμικού (π.χ. πρωτότυπα και τμήματα κώδικα του λογισμικού).
- ▶ Έπονται των στατικών τεχνικών, που αν γίνουν επιτυχώς, περιορίζουν το κόστος του δυναμικού ελέγχου (απαιτούνται λιγότεροι έλεγχοι).
- ▶ **Απόδειξη ύπαρξης και όχι απουσίας σφαλμάτων.**
- ▶ Οι μόνες τεχνικές επικύρωσης μη-λειτουργικών απαιτήσεων (performance, usability) μιας και απαιτούν την εκτέλεση του λογισμικού (ή τμήματος αυτού).

## Δυναμικές Τεχνικές Εγκυροποίησης

29

- ▶ **Δοκιμές λογισμικού** (με πραγματικά δεδομένα)
- ▶ **Συμβολική εκτέλεση** (με συμβολικά δεδομένα)
- ▶ **Προσομοίωση** (simulation):
  - ▶ Χρήση τεχνητών δεδομένων εισόδου που προσομοιώνουν την πραγματικότητα, σε πραγματικές συνθήκες λειτουργίας.
  - ▶ Προσομοίωση του τρόπου λειτουργίας τμημάτων του λογισμικού που δεν έχουν ακόμη ολοκληρωθεί.
  - ▶ Ιδιαίτερα δαπανηρή μέθοδος.
- ▶ **Ανάλυση ευαισθησίας** (sensitivity analysis).

## Συμβολική Εκτέλεση

30

- ▶ Έλεγχος καταστάσεων εξόδου με δεδομένες τις καταστάσεις εισόδου (μεταβλητές) και τις συνθήκες ελέγχου.
- ▶ Έλεγχος επίδρασης μεταβλητών εισόδου στην έξοδο του προγράμματος.
- ▶ Χρήσιμη τεχνική για εύρεση περιπτώσεων ελέγχου.
- ▶ Διαδικασία:
  - ▶ Ανάθεση σε κάθε μεταβλητή εισόδου μίας **συμβολικής τιμής**.
  - ▶ Διάδοση των συμβολικών τιμών στις υπόλοιπες μεταβλητές του προγράμματος.

## Συμβολική Εκτέλεση – Παράδειγμα

31

ΑΛΓΟΡΙΘΜΟΣ test

ΑΡΧΗ

ΔΙΑΒΑΣΕ (x, y)

ΕΑΝ (x &gt; y) ΤΟΤΕ

x = x + y;

y = x - y;

x = x - y;

ΕΑΝ (x - y &gt; 0)

ΕΚΤΕΛΕΣΕ ΤΜΗΜΑ ΚΩΔΙΚΑ 1;

ΑΛΛΙΩΣ

ΕΚΤΕΛΕΣΕ ΤΜΗΜΑ ΚΩΔΙΚΑ 2;

ΕΑΝ-ΤΕΛΟΣ

ΕΑΝ-ΤΕΛΟΣ

ΤΕΛΟΣ

 $x \leftarrow X, y \leftarrow Y$  $X > Y$  $x \leftarrow X + Y$  $y \leftarrow X + Y - Y = X$  $x \leftarrow X + Y - X = Y$  $Y - X > 0 \Rightarrow Y > X$  $Y - X \leq 0 \Rightarrow Y \leq X$ 

Μιχάλης Ξένος, 2025

31

## Συμβολική Εκτέλεση – Παράδειγμα

32



Μιχάλης Ξένος, 2025

32



## Ανάλυση Ευαισθησίας (Sensitivity Analysis)

33

- ▶ Συμπεριφορά λογισμικού σε σχέση με τη πιθανότητα ύπαρξης λαθών.
- ▶ Επαναλαμβανόμενη εκτέλεση προγράμματος και λανθασμένων παραλλαγών αυτού.
- ▶ Βασίζεται στις υποθέσεις:
  - ▶ Μοναδικού σφάλματος (σε όλο τον κώδικα!).
  - ▶ Απλού σφάλματος (σε μία μόνο θέση στον κώδικα!).
- ▶ Πιθανότητα αποτυχίας εξαιτίας μοναδικού σφάλματος σε ένα σημείο του κώδικα (υπολογίζεται για όλα τα σημεία).
- ▶ Υπολογίζεται **εισάγοντας εσκεμμένα σφάλματα** στον κώδικα.

## Ανάλυση Ευαισθησίας (Sensitivity Analysis)

34

- ▶ Τρεις διαδικασίες ανάλυσης για κάθε σημείο του κώδικα (στοχεύουν να καλύψουν την πιθανότητα εκτέλεσης, πιθανότητα λάθους στην κατάσταση και πιθανότητα εκτέλεσης λανθασμένου τμήματος κώδικα – εξόδου):
  - ▶ **Ανάλυση εκτέλεσης (execution analysis)**: πιθανότητα εκτέλεσης λανθασμένου σημείου σε ένα σύνολο δεδομένων ελέγχου.
  - ▶ **Ανάλυση μόλυνσης (infection analysis)**: πιθανότητα τροποποίησης κατάστασης δεδομένων σε σχέση με τις αναμενόμενες τιμές σε ένα σύνολο δεδομένων ελέγχου.
  - ▶ **Ανάλυση διάδοσης (propagation analysis)**: πιθανότητα τροποποίησης κατάστασης εξόδου (π.χ. τμήμα κώδικα που εκτελείται) αν υπάρχει τροποποίηση στην κατάσταση των δεδομένων μετά την εκτέλεση της εσφαλμένης εντολής.

## Ανάλυση Ευαισθησίας - Παράδειγμα

35

```
void test()
{
  int x,y,z;
  scanf ("%d%d", &x, &y);
  if (x > 3)
  {
    z = x + y;
    y += x;
    if ( 2 * z == y)
      ΤΜΗΜΑ ΚΩΔΙΚΑ 1
    else
      ΤΜΗΜΑ ΚΩΔΙΚΑ 2
  }
}
```

Ανάλυση ευαισθησίας για  
την εντολή

με δεδομένα ελέγχου

x	y
6	-6
8	8

Αντικατάσταση εντολής με

$z=x-y;$

## Ανάλυση Ευαισθησίας - Παράδειγμα

36

- ▶ Η εντολή θα εκτελεστεί για όλα τα δεδομένα ελέγχου άρα  $P_e = 1$ .
- ▶ Περίπτωση 1
  - ▶  $x = 6$   $y = -6$
  - ▶ Αντί για  $z = y = 0 \rightarrow z = 12$   $y = 0$
  - ▶ Εκτελείται το Τμήμα Κώδικα 2 αντί για το Τμήμα Κώδικα 1.
- ▶ Περίπτωση 2
  - ▶  $x = 8$   $y = 8$
  - ▶ Αντί για  $z = 16$   $y = 16 \rightarrow z = 0$   $y = 16$
  - ▶ Εκτελείται (όπως θα έπρεπε) το Τμήμα Κώδικα 2.
- ▶ Σε όλες τις περιπτώσεις έχουμε σφάλμα στην κατάσταση δεδομένων, άρα  $P_\mu = 1$ .
- ▶ Μόνο στη μία περίπτωση υπήρξε διάδοση της μόλυνσης (εκτελέστηκε άλλο τμήμα κώδικα), άρα  $P_\delta = 0.5$ .
- ▶ Πιθανότητα αστοχίας  $= P_e * P_\mu * P_\delta = 1 * 1 * 0.5 = 0.5 = 50\%$

## Έλεγχος Συστήματος Λογισμικού – Software Testing

37

- ▶ Πιο διαδεδομένη και χρήσιμη μη τυπική, δυναμική τεχνική εγκυροποίησης.
- ▶ Δοκιμαστική εκτέλεση συστήματος με δεδομένα εισόδου που μοιάζουν (κατά το δυνατό) με πραγματικά.
- ▶ Περίπτωση Ελέγχου (test case): κάθε δοκιμαστική εκτέλεση του λογισμικού.
- ▶ Δεδομένα ελέγχου (test data): δεδομένα εισόδου περιπτώσεων ελέγχου.
- ▶ Για κάθε περίπτωση ελέγχου υπάρχει προδιαγραφή των δεδομένων εισόδου και αναμενόμενης εξόδου του υπό δοκιμή προγράμματος. Ο έλεγχος εξετάζει τη συμφωνία με την προδιαγραφή.

## Έλεγχος Συστήματος Λογισμικού – Software Testing

38

- ▶ Το μόνο είδος ελέγχου που παρέχει απόλυτη βεβαιότητα για την ορθότητα του λογισμικού είναι ο εξαντλητικός έλεγχος (exhaustive testing) – έλεγχος με όλα τα δυνατά δεδομένα εισόδου.
- ▶ Πολύ χρονοβόρα και πρακτικά αδύνατη διαδικασία.
- ▶ Επομένως σχεδιάζεται **στρατηγική ελέγχου (software testing strategy)** που ορίζει
  - ▶ τα κριτήρια επιλογής σημαντικών ελέγχων (η σημαντικότητα όμως δεν μπορεί να καθοριστεί με τυπικό τρόπο) και επίσης
  - ▶ πότε πρέπει να τερματίζεται ο έλεγχος.

## Φάσεις Ελέγχου Λογισμικού

39

- ▶ **Έλεγχος Μονάδων** (unit testing): Έλεγχος κάθε μονάδας ανεξαρτήτως των άλλων τμημάτων του λογισμικού
- ▶ **Έλεγχος Τμημάτων** (module testing): δοκιμή συσχετιζόμενων μονάδων που αποτελούν μία ενότητα.
- ▶ **Έλεγχος Υπο-συστημάτων** (sub-system testing) ή **Έλεγχος Ολοκλήρωσης**: Μετά την ολοκλήρωση των τμημάτων σε υποσύστημα, ελέγχεται η συγκρότηση και η διεπαφή μεταξύ των τμημάτων.
- ▶ **Έλεγχος Συστήματος** (system testing): Έλεγχος συγκρότησης υποσυστημάτων σε ένα σύστημα. Έλεγχος ορθότητας συστήματος και επικύρωση του συστήματος αναφορικά με τις απαιτήσεις που καθορίστηκαν και προδιαγράφηκαν.
- ▶ **Έλεγχος Αποδοχής** (acceptance testing): Έλεγχος χρήσης υπό πραγματικές συνθήκες (δεδομένα και συχνά χώρος λειτουργίας). Τελευταίο στάδιο δοκιμών πριν την παράδοση του συστήματος.

## Δοκιμές Λογισμικού - Στόχος

40

- ▶ Ανακάλυψη όσο το δυνατό περισσότερων σφαλμάτων και ατελειών.
- ▶ **Ιδανικό σύνολο ελέγχων**: σύνολο που αν υπάρχει ένα σφάλμα στο πρόγραμμα, αυτό αποκαλύπτεται με βάση τουλάχιστον μία περίπτωση δεδομένων ελέγχου. Πρακτικά αδύνατο να σχεδιαστεί ιδανικό σύνολο ελέγχων (ειδικά για σύνθετα προγράμματα).
- ▶ Πολλές φορές χρησιμοποιείται **ανεξάρτητη ομάδα ελέγχου** αντί για τους δημιουργούς ενός προγράμματος για τις δοκιμές.





# Κατηγορίες Ελέγχου Λογισμικού

41

- ▶ **Λειτουργικός Έλεγχος** (functional testing) ή **Έλεγχος Αδιαφανούς Κουτιού** (black-box testing):
  - ▶ Άγνοια εσωτερικής δομής & λειτουργίας
  - ▶ Για κάθε είσοδο, έλεγχος συμφωνίας της παραγόμενης σε σχέση με την αναμενόμενη έξοδο
  - ▶ Κατάλληλη μέθοδος για έλεγχο συστήματος.
- ▶ **Δομικός Έλεγχος** (structural testing) ή **Έλεγχος Διαφανούς Κουτιού** (white-box testing):
  - ▶ Επιλογή δεδομένων ελέγχου με βάση τη γνώση της εσωτερικής δομής και λειτουργίας.
  - ▶ Κατάλληλη μέθοδος για έλεγχο μονάδων και τμημάτων (λόγω μικρού μεγέθους).
- ▶ **Έλεγχος Διεπαφών** (interface testing):
  - ▶ έλεγχος συμπεριφοράς τμημάτων σε σχέση με συσχετιζόμενα τμήματα.

# Κατηγορίες Ελέγχου Λογισμικού

42

Κατηγορία τεχνικής	Η όψη του ελεγκτή	Πηγή πληροφοριών	Μέθοδοι
Αδιαφανούς κουτιού		<ul style="list-style-type: none"> <li>- Έγγραφο απαιτήσεων</li> <li>- Προδιαγραφές</li> <li>- Γνώση πεδίου</li> <li>- Απευθείας ανάλυση δεδομένων</li> </ul>	<ul style="list-style-type: none"> <li>- Κλάσεις ισοδυναμίας</li> <li>- Έλεγχος οριακών τιμών</li> <li>- Γράφος αιτίου – αποτελέσματος</li> <li>- Υποθέσεις</li> </ul>
Διαφανούς κουτιού		<ul style="list-style-type: none"> <li>- Σχέδιο λογισμικού</li> <li>- Γράφος ελέγχου</li> <li>- Γράφος κυκλωματικής πολυπλοκότητας</li> </ul>	<ul style="list-style-type: none"> <li>- Έλεγχος εντολών</li> <li>- Έλεγχος διακλαδώσεων</li> <li>- Έλεγχος μονοπατιών</li> <li>- Έλεγχος ροής δεδομένων</li> <li>- Έλεγχος επαναλήψεων</li> </ul>

## Κατηγορίες Ελέγχου Λογισμικού

43



## Κατηγορίες Ελέγχου Λογισμικού

44

- ▶ Έλεγχος ως γκρι κουτί, λαχανί κουτί, και λαχανί κουτί με ροζ βουλίτσες
- ▶ ... και άλλα πολύχρωμα κουτιά

## Έλεγχος Μαύρου Κουτιού

45



Δεν έχουμε γνώση του κώδικα (και δεν θέλουμε να έχουμε), αλλά βασιζόμαστε μόνο στις προδιαγραφές.

## Πριν ξεκινήσουμε...

46

- ▶ Έστω 2 προγράμματα που υλοποιούν τις εξής συναρτήσεις (τα  $x$ ,  $f(x)$ ,  $g(x)$  είναι ακέραιοι):
  - ▶  $f(x) = |x|$
  - ▶  $g(x) = x^2$
- ▶ Ποιες περιπτώσεις ελέγχου θα σχεδιάζατε;

## Έλεγχος Αδιαφανούς Κουτιού (black-box)

47

- ▶ Σχεδίαση περιπτώσεων ελέγχου με βάση τις απαιτήσεις ή τις προδιαγραφές.
- ▶ Εξέταση συμπεριφοράς με βάση το συσχετισμό δεδομένων εισόδου με δεδομένα εξόδου.
- ▶ Οι περιπτώσεις ελέγχου σχεδιάζονται με σκοπό να βρεθούν τα δεδομένα εισόδου που είναι πιθανότερο να προκαλέσουν μη επιθυμητή συμπεριφορά.
- ▶ Βασικές τεχνικές:
  - ▶ Τυχαίος έλεγχος.
  - ▶ **Διαμέριση σε κλάσεις ισοδυναμίας** (equivalence partitioning).
  - ▶ **Ανάλυση οριακών τιμών** (boundary value analysis).

## Διαμέριση σε Κλάσεις Ισοδυναμίας

48

- ▶ Διαμερισμός δυνατών δεδομένων εισόδου σε **κλάσεις δεδομένων** από τις οποίες θα προκύψουν οι περιπτώσεις ελέγχου
  - ▶ Το λογισμικό αναμένεται να συμπεριφέρεται με τον ίδιο τρόπο για όλα τα δεδομένα μίας κλάσης.
  - ▶ Αγνοούνται οι εξαρτήσεις μεταξύ των κλάσεων.
  - ▶ Οι κλάσεις ισοδυναμίας προκύπτουν από τους περιορισμούς επί των δεδομένων εισόδου ή του τρόπου που το πρόγραμμα αντιμετωπίζει τα δεδομένα εισόδου.



## Διαμέριση σε Κλάσεις Ισοδυναμίας

49

- Ορισμός κλάσεων ισοδυναμίας για ένα δεδομένο εισόδου:

- **Μεταβλητή = Τιμή:** 3 κλάσεις ισοδυναμίας.

- Μεταβλητή = Τιμή

- Μεταβλητή > Τιμή

- Μεταβλητή < Τιμή

- **Κάτω όριο < Μεταβλητή < Άνω όριο:** 3 κλάσεις

- Κάτω όριο < Μεταβλητή < Άνω όριο

- Μεταβλητή ≤ Κάτω όριο

- Μεταβλητή ≥ Άνω όριο

- **Μεταβλητή IN Σύνολο τιμών:** 2 κλάσεις ισοδυναμίας

- Μεταβλητή IN Σύνολο τιμών

- Μεταβλητή NOT IN Σύνολο τιμών

- **Boolean τιμές:** 2 κλάσεις ισοδυναμίας

- Αληθής

- Ψευδής

## Διαμέριση σε Κλάσεις Ισοδυναμίας

50

Για κάθε κλάση σχεδιάζουμε **περιπτώσεις ελέγχου (test cases)**. Μια μέθοδος με ικανοποιητικά αποτελέσματα:

- Επιλέγουμε δεδομένα εισόδου **κοντά στο μέσο της κλάσης**
- Επιλέγουμε δεδομένα εισόδου **κοντά στα όρια της κλάσης** (εφαρμογή μιας άλλης τεχνικής που θα παρουσιαστεί στη συνέχεια)

## Διαμέριση σε Κλάσεις Ισοδυναμίας – Παράδειγμα 1

51

- ▶ Έστω ότι πρέπει να υπολογιστεί το παραγοντικό ενός ακέραιου αριθμού ως εξής:
  - ▶ Αν η τιμή εισόδου  $n$  είναι  $< 0$  η είσοδος δεν γίνεται αποδεκτή εμφανίζοντας κατάλληλο μήνυμα.
  - ▶ Αν  $0 \leq n \leq 20$  εκτυπώνεται η ακριβής τιμή του  $n!$
  - ▶ Αν  $20 < n \leq 200$  εκτυπώνεται μία προσεγγιστική τιμή του  $n!$  σε επιστημονικό συμβολισμό. Το αποδεκτό σφάλμα είναι 0.1% της ακριβούς τιμής.
  - ▶ Αν  $n > 200$  η είσοδος δεν γίνεται αποδεκτή εμφανίζοντας κατάλληλο μήνυμα.

## Διαμέριση σε Κλάσεις Ισοδυναμίας – Παράδειγμα 1

52

- ▶ Κλάσεις ισοδυναμίας:
  - ▶  $n < 0$  (άκυρες τιμές)
  - ▶  $0 \leq n \leq 20$
  - ▶  $20 < n \leq 200$
  - ▶  $n > 200$  (άκυρες τιμές)
- ▶ Περιπτώσεις ελέγχου:
  - ▶  $n < 0 \rightarrow n = -10$
  - ▶  $0 \leq n \leq 20 \rightarrow n = 10$
  - ▶  $20 < n \leq 200 \rightarrow n = 110$
  - ▶  $n > 200 \rightarrow n = 300$

## Ανάλυση Οριακών Τιμών

1/2

53

- ▶ Χρησιμοποιείται **σε συνδυασμό με την τεχνική διαμέρισης** σε κλάσεις ισοδυναμίας.
- ▶ Αρχικά ορίζουμε τις κλάσεις ισοδυναμίας (όμοια με την τεχνική διαμέρισης σε κλάσεις ισοδυναμίας).
- ▶ Σχεδίαση περιπτώσεων ελέγχου **στα όρια τιμών κάθε κλάσης** (όπου γίνεται η υπόθεση ότι παρατηρούνται οι περισσότερες δυσλειτουργίες).
- ▶ Έλεγχος έγκυρων ΚΑΙ άκυρων κλάσεων.
- ▶ Περιλαμβάνει και περιπτώσεις ελέγχου του πεδίου τιμών (έξοδοι του λογισμικού).

## Ανάλυση Οριακών Τιμών

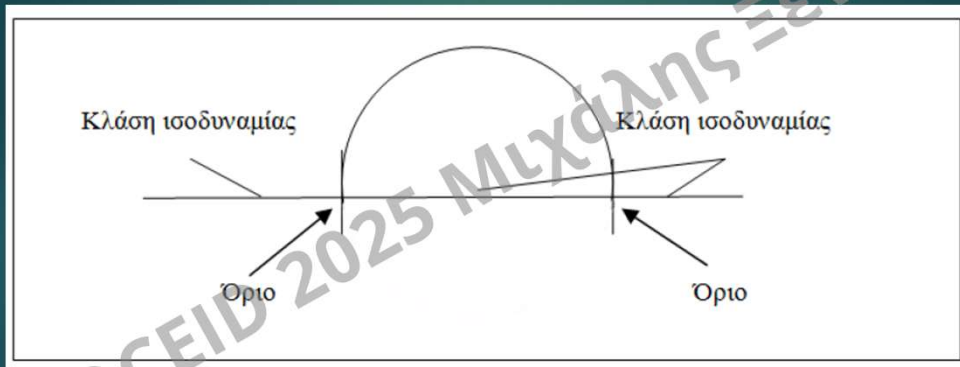
2/2

54

- ▶ Περιλαμβάνει και περιπτώσεις ελέγχου του πεδίου τιμών (έξοδοι του λογισμικού).
- ▶ **Επιλέγουμε δεδομένα εισόδου τα οποία παράγουν δεδομένα σε κάθε κλάση δεδομένων εξόδου**
  - ▶ Δηλ. για κάθε δεδομένο εξόδου βρίσκουμε (όπως και πριν) τις κλάσεις ισοδυναμίας, αλλά για τις περιπτώσεις ελέγχου θα πρέπει να καθορίσουμε τις εισόδους που παράγουν δεδομένα σε κάθε κλάση δεδομένων εξόδου

## Ανάλυση Οριακών Τιμών

55



55

## Ανάλυση Οριακών Τιμών

56

- ▶ Οδηγίες σχεδίασης περιπτώσεων ελέγχου:
  - ▶ **Μεταβλητή in Σύνολο τιμών:**  
Περιπτώσεις ελέγχου για όλες τις έγκυρες τιμές αν είναι λίγες, αλλιώς για τις πιο ενδεικτικές και ειδικά τις ακραίες αν υπάρχουν.
  - ▶ **Κάτω όριο < Μεταβλητή < Άνω όριο:**  
Περιπτώσεις ελέγχου για κάτω και άνω όριο καθώς και για τιμές λίγο μεγαλύτερες από το κάτω όριο και λίγο μικρότερες από το άνω όριο. (θα μπορούσαν να προστεθούν και οριακά άκυρες τιμές)
  - ▶ **Δομές δεδομένων με καθορισμένα όρια (π.χ. πίνακες):** Περιπτώσεις ελέγχου κοντά στα όρια της δομής (π.χ. πίνακας με ένα μόνο στοιχείο).
  - ▶ Εφαρμογή των δύο πρώτων οδηγιών για τους περιορισμούς που ισχύουν για **τα δεδομένα εξόδου**.

56



## Ανάλυση Οριακών Τιμών – Παράδειγμα

57

- ▶ Κλάσεις ισοδυναμίας:
  - ▶  $n < 0$
  - ▶  $0 \leq n \leq 20$
  - ▶  $20 < n \leq 200$
  - ▶  $n > 200$
- ▶ Περιπτώσεις ελέγχου για  $n < 0$ 
  - ▶  $n = -1, n = -2, n = 0$  μικρότερος δυνατός αρνητικός,  $n = 0$  μικρότερος δυνατός αρνητικός + 1
- ▶ Περιπτώσεις ελέγχου για  $0 \leq n \leq 20$ 
  - ▶  $n = 0, n = 1, n = 20, n = 19$
- ▶ Περιπτώσεις ελέγχου για  $20 < n \leq 200$ 
  - ▶  $n = 21, n = 22, n = 200, n = 199$
- ▶ Περιπτώσεις ελέγχου για  $n > 200$ 
  - ▶  $n = 201, n = 202, n = 0$  μεγαλύτερος δυνατός θετικός,  $n = 0$  μεγαλύτερος δυνατός θετικός - 1

## Ανάλυση Οριακών Τιμών – Παράδειγμα

58

- ▶ Περιπτώσεις για δεδομένα εξόδου
  - ▶ Κλάσεις εξόδου:
    - ▶ ο μικρότερος δυνατός αρνητικός  $\leq \text{fact} \leq 0$
    - ▶  $1 \leq \text{fact} \leq 0$  μεγαλύτερος δυνατός θετικός
    - ▶  $\text{fact} > 0$  μεγαλύτερος δυνατός θετικός
  - ▶ Για κάθε κλάση θα πρέπει να προσδιοριστούν δοκιμαστικές τιμές για το  $n$ , όπου το  $\text{fact} = n!$  θα ανήκει σε αυτή την κλάση.
    - ▶ δεν υπάρχει περίπτωση ελέγχου που να παράγει τέτοια έξοδο, καθώς δεν ορίζεται το παραγοντικό για αρνητικούς ακεραίους.
    - ▶  $n = 15$
    - ▶  $n = 45$  εδώ θέλει λίγο προσοχή γιατί από υπολογιστή σε υπολογιστή αλλάζει η τιμή του μεγαλύτερου δυνατού θετικού (π.χ. σε συστήματα 32-bit και 64-bit).
  - ▶ Παρατηρούμε ότι έχουμε ήδη καλύψει τις παραπάνω περιπτώσεις ελέγχου από τις προηγούμενες. Έτσι δεν προσθέτουμε κάποια επιπλέον. Αυτό όμως μπορεί να μην ισχύει σε άλλες ασκήσεις.

## Διαμέριση σε Κλάσεις Ισοδυναμίας & Ανάλυση Οριακών Τιμών – Παράδειγμα 2

59

- ▶ Λογισμικό υπολογισμού τόκων σε λογαριασμό τραπεζής, σύμφωνα με τον ακόλουθο πίνακα κεφαλαίου - επιτοκίων.

$< 500 \text{ €}$	0%
$< 1000 \text{ €}$	2%
$\geq 1000 \text{ €}$	4%
$\geq 5000 \text{ €}$	5%

- ▶ Ορισμός κλάσεων ισοδυναμίας και σχεδίαση περιπτώσεων ελέγχου με βάση τις δύο τεχνικές.
- ▶ Κεφάλαιο: πραγματικός αριθμός, μηδέν ή θετικός.

## Διαμέριση σε Κλάσεις Ισοδυναμίας & Ανάλυση Οριακών Τιμών – Παράδειγμα 2

60

- ▶ Ορισμός κλάσεων ισοδυναμίας:
  - ▶ Κεφάλαιο  $< 0$  (άκυρες τιμές)
  - ▶  $0 \leq \text{Κεφάλαιο} < 500$
  - ▶  $500 \leq \text{Κεφάλαιο} < 1000$
  - ▶  $1000 \leq \text{Κεφάλαιο} < 5000$
  - ▶ Κεφάλαιο  $\geq 5000$
- ▶ Το Επιτόκιο δεν είναι είσοδος, αλλά εσωτερική μεταβλητή του λογισμικού, άρα δεν την λαμβάνω υπόψη στην ανάλυση μου.

## Διαμέριση σε Κλάσεις Ισοδυναμίας & Ανάλυση Οριακών Τιμών – Παράδειγμα 2

61

- ▶ Κεφάλαιο  $< 0$ 
  - ▶ Κεφάλαιο = -500
  - ▶ Κεφάλαιο = -0.01
  - ▶ Κεφάλαιο = ο μικρότερος δυνατός αρνητικός πραγματικός
- ▶  $0 \leq \text{Κεφάλαιο} < 500$ 
  - ▶ Κεφάλαιο = 0
  - ▶ Κεφάλαιο = 0.01
  - ▶ Κεφάλαιο = 499.99
  - ▶ Κεφάλαιο = 500
- ...
- ▶ Κεφάλαιο  $\geq 5000$ 
  - ▶ Κεφάλαιο = 5000
  - ▶ Κεφάλαιο = 5000.01
  - ▶ Κεφάλαιο = ο μεγαλύτερος δυνατός θετικός πραγματικός

## Τέλος 10<sup>ης</sup> διάλεξης

62

- ▶ Σε αυτό το σημείο η διάλεξη τελείωσε
- ▶ Και μόλις τώρα ξεκινά το σημαντικότερο μέρος της δουλειάς σας για να μην πάει χαμένο το 2ωρο ++ που αφιερώσατε!