Ανάπτυξη & Σχεδίαση Λογισμικού (ΗΥ420)

Έλεγχος Λογισμικού

Χρήστος Δ. Αντωνόπουλος
9/1/2014

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Προβλήματα Λογισμικού

- Μια ματιά στα παλιά:
  - Σφάλμα:
    - Ανθρώπινο λάθος (σε προδιαγραφές, τεκμηρίωση κλπ)
  - Ελάττωμα:
    - «Κωδικοποίηση του σφάλματος» στο προϊόν λογισμικού
  - Αστοχία:
    - Απομάκρυνση από την προδιαγεγραμμένη λειτουργία του συστήματος
    - Η εμφάνιση του ελαττώματος
    - Αίτια αστοχίας:
      - Λανθασμένες ή ελλιπείς προδιαγραφές
      - Μη ρεαλιστικές προδιαγραφές
      - Ελαττώματα στη σχεδίαση συστήματος / προγράμματος και στην υλοποίηση (κώδικας)

Έλεγχος Λογισμικού

- Επιβεβαιώνει την ύπαρξη υποπτευόμενης αστοχίας …
- … ή μας βοηθά να προκαλέσουμε νέες αστοχίες
- Πότε είναι επιτυχής;
  - Όταν κατά τη διάρκειά του συμβαίνουν «άσχημα πράγματα»…
  - … όταν δηλαδή μας επιβεβαιώνει ότι δεν κάναμε καλά τη δουλειά μας
    - Ανακαλύπτεται ελάττωμα ☹
    - Συμβαίνει αστοχία ως αποτέλεσμα της διαδικασίας ελέγχου ☹
    - Τελικά καλύτερα τώρα παρά αργότερα…

Έλεγχος Λογισμικού

- Αρχές:
  - Όλοι οι έλεγχοι θα πρέπει να είναι συσχετίσιμοι με τις απαιτήσεις του πελάτη
  - Οι έλεγχοι θα πρέπει να έχουν σχεδιαστεί αρκετά πριν την εφαρμογή τους
    - Σενάριο ελέγχου
    - Αναμενόμενο αποτέλεσμα
  - Ισχύει η αρχή Pareto (80/20)
  - Έλεγχος από το ειδικότερο προς το γενικότερο
  - Δεν είναι δυνατός ο εξαντλητικός έλεγχος
  - Καλό είναι ο έλεγχος να γίνεται από κάποιον «τρίτο»

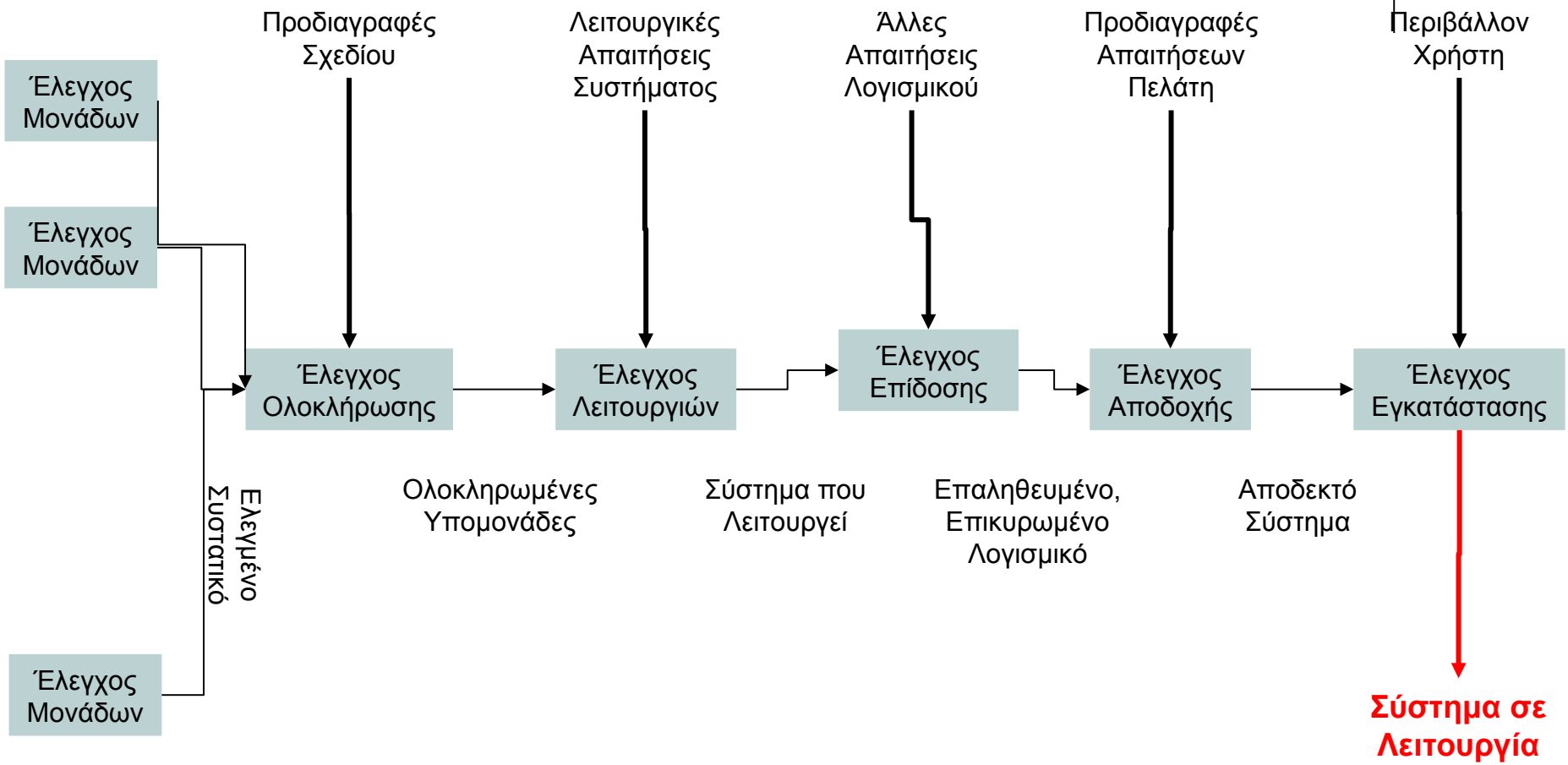
Τι Συμβαίνει κατά τον Έλεγχο;

- Αναγνώριση σφαλμάτων:
  - Ακριβής προσδιορισμός του / των σφάλματος/ων που προκάλεσαν μια αστοχία
- Διόρθωση / εξάλειψη σφαλμάτων
  - Αλλαγές στο σύστημα ώστε να διορθωθούν τα ελαττώματα

Πού μπορώ να Βρω Ελαττώματα;

- Παντού!
  - Αλγόριθμοι
  - Σύνταξη
  - Σφάλματα υπολογισμού ή ακρίβειας
  - Τεκμηρίωση
  - Σφάλματα πίεσης / υπερφόρτωσης
  - Σφάλματα χωρητικότητας / ορίων
  - Σφάλματα χρονισμού / συγχρονισμού
  - Σφάλματα διεκπεραιωτικής ικανότητας
  - Σφάλματα ανάκτησης
  - Σφάλματα υλικού / λογισμικού
  - Σφάλματα προτύπων & διαδικασιών

Οργάνωση Σταδίων Ελέγχου



Τα «Κοινωνικά» Ζητήματα του Ελέγχου…

- Ποιος τον κάνει;
  - Ομάδα ανάπτυξης:
    - Μονάδων & ολοκλήρωσης
  - Ομάδα ελέγχου:
    - Υπόλοιποι έλεγχοι (έλεγχοι συστήματος)
- Τάσεις:
  - Οι άπειροι προγραμματιστές δεν έχουν τη σωστή νοοτροπία για τον έλεγχο ως μέσο ανακάλυψης λαθών
  - Κανείς δε δέχεται την ευθύνη για το σφάλμα
    - Λύση: Μη εγωιστικός προγραμματισμός (egoless programming)
  - Όλοι προσπαθούν να κάνουν τη δουλειά τους
    - Οι ελεγκτές να δείξουν ότι βρίσκουν λάθη
    - Η ομάδα ανάπτυξης να δείξει ότι δεν κάνει
      - Συχνές διενέξεις…

Πώς «Βλέπω» τα Αντικείμενα Ελέγχου;

- Πρόγραμμα:
  - Συνάρτηση απεικόνισης …
  - … από το χώρο όλων των πιθανών εισόδων …
  - … στο χώρο όλων των πιθανών λύσεων
    - Πλήρης έλεγχος είναι συνήθως ανέφικτος

Πώς Αντιμετωπίζω τα Αντικείμενα Ελέγχου;

- Μαύρο κουτί:
  - Ξέρουμε μόνο το interface εισόδου και παρατηρούμε την έξοδο
    - Δε γνωρίζουμε τίποτα για την εσωτερική οργάνωση
    - Δύσκολο να επιλέξουμε αντιπροσωπευτικές περιπτώσεις ελέγχου
    - Μπορούμε μόνο να μοντελοποιήσουμε καλά τον έλεγχο αλληλεπίδρασης με άλλα συστατικά
    - Αν είχαμε δυνατότητα άπειρων ελέγχων;
- Λευκό κουτί:
  - Γνωρίζουμε και την εσωτερική οργάνωση
    - Μπορούμε να επιλέξουμε χαρακτηριστικές περιπτώσεις ελέγχου
    - Πάντως και πάλι δύσκολο να τις ελέγξουμε όλες…
- Συνδυασμοί των δύο

Έλεγχος Μονάδων: Επανεξέταση Κώδικα

- Μελέτη του πηγαίου κώδικα και της τεκμηρίωσης
  - Αναζήτηση ελαττωμάτων, παραλείψεων, σφαλμάτων
  - Ανασκόπηση κώδικα:
    - Παρουσίαση κώδικα και τεκμηρίωσης
    - Περιήγηση στον Κώδικα
    - Συζήτηση και σχολιασμός
  - Επιθεώρηση κώδικα:
    - Ομάδα εξέτασης ελέγχει κώδικα και τεκμηρίωση…
    - … με βάση έναν κατάλογο σημείων προς εξέταση

Επανεξέταση Κώδικα

- Ιδιαίτερα χρήσιμη πρακτική
  - Αποκαλύπτεται μεγάλος αριθμός προβλημάτων
  - Όμως… οι προγραμματιστές συχνά αντιδρούν
    - Δεν αισθάνονται άνετα στην ιδέα της επανεξέτασης του κώδικά τους από τρίτους
  - Πάντως θεωρείται απαραίτητη και εφαρμόζεται από τους περισσότερους μεγάλους οργανισμούς ανάπτυξης λογισμικού

Τυπική Απόδειξη Ορθότητας Κώδικα

- Εξονυχιστική μελέτη με δομημένο τρόπο
  - Ορθό πρόγραμμα: Υλοποιεί λειτουργίες και επεξεργάζεται/εξάγει δεδομένα με τον κατάλληλο τρόπο και παρουσιάζει κατάλληλες διασυνδέσεις με άλλα συστατικά
  - Θεώρηση προγράμματος ως λογική ροή
    - Π.χ. ως σύνολο ισχυρισμών και θεωρημάτων
      - Ισχυρισμοί (συνθήκες εισόδου / εξόδου κάθε συστατικού)
      - Θεωρήματα προς απόδειξη
      - Απόδειξη θεωρημάτων
      - Απόδειξη τερματισμού

Τυπική Απόδειξη

- Πλεονεκτήματα:
  - Αποκάλυψη αλγοριθμικών λαθών
  - Τυπική κατανόηση του προγράμματος
  - Αυστηρότητα
- Μειονεκτήματα:
  - Απαιτητική τεχνική σε χρόνο
  - Όχι ιδιαίτερα κατάλληλη για μεγάλα/πολύπλοκα συστατικά
  - Παράλληλη επεξεργασία;
  - Μη εντοπισμός σφαλμάτων σχεδίου, διασύνδεσης, ερμηνείας προδιαγραφών, σύνταξης/σημειολογίας γλώσσας προγραμματισμού, τεκμηρίωσης
  - Ποιος μας εγγυάται την ορθότητα της απόδειξης;

Άλλες Τεχνικές Απόδειξης

- Συμβολική εκτέλεση:
  - Προσομοιωμένη εκτέλεση κώδικα…
  - … σύμβολα αντί μεταβλητών δεδομένων
    - Λογικό μονοπάτι εκτέλεσης ⇔ ακολουθία αλλαγών κατάστασης των συμβολικών μεταβλητών
  - Παρόμοια πλεονεκτήματα / μειονεκτήματα με τις τυπικές αποδείξεις
    - Μπορεί όμως να αυτοματοποιηθεί
      - Τουλάχιστον μερικώς
    - Δύσκολο αυτόματα συμβολικά εργαλεία να παρακολουθήσουν την εκτέλεση βρόχων

Άλλες Τεχνικές Απόδειξης

- Αυτοματοποιημένη απόδειξη θεωρημάτων
  - Δύσκολο να κατασκευαστούν εργαλεία
  - Trial and error μη αποδεκτή στρατηγική εκτός για πολύ μικρά συστατικά
    - Ανθρώπινη παρέμβαση
      - Τότε το σύστημα ουσιαστικά ελέγχει την απόδειξη που δίνεται από το χρήστη
  - Δεν είναι δυνατό να κατασκευαστεί το ιδανικό σύστημα απόδειξης θεωρημάτων για έλεγχο λογισμικού

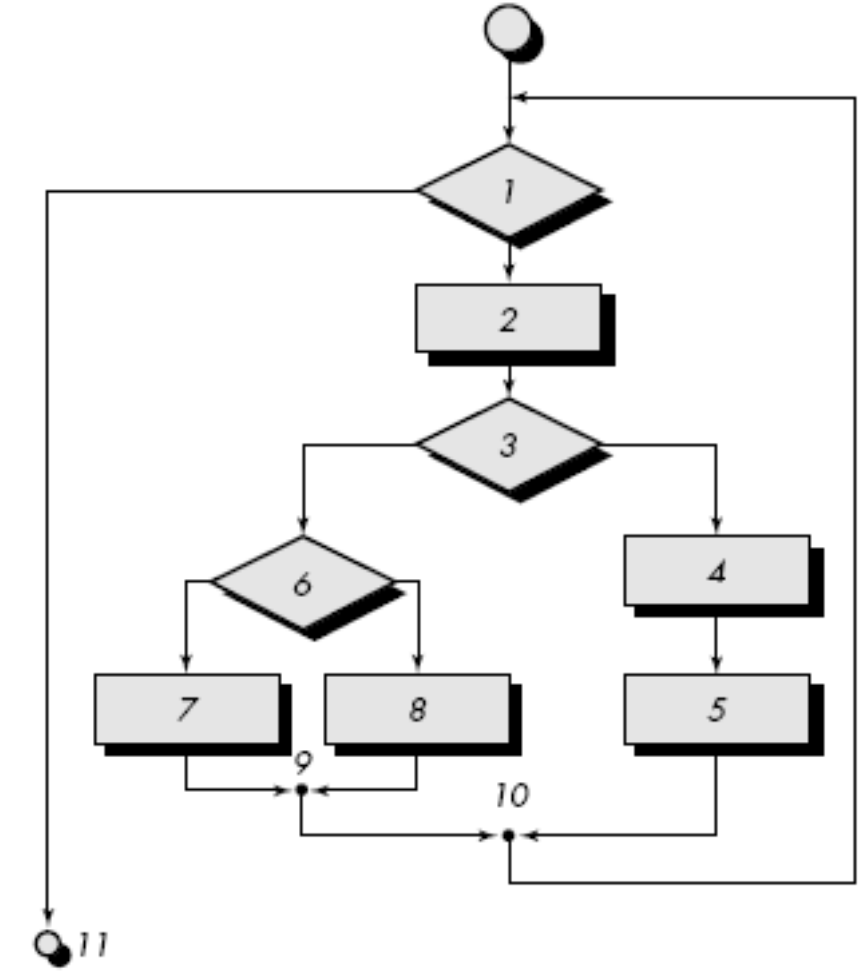
Έλεγχος Συστατικών

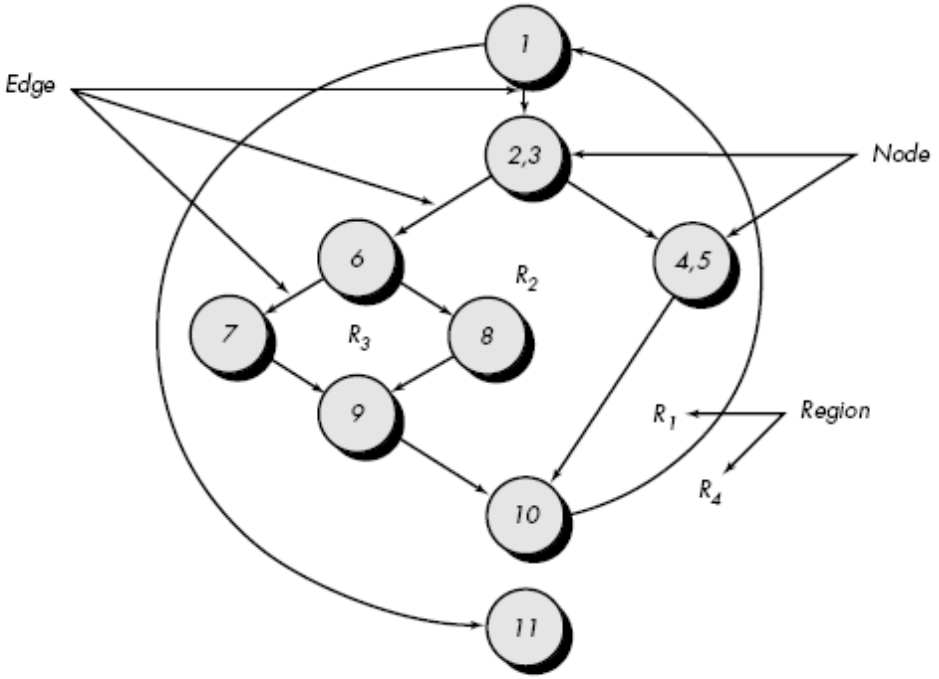
- Πειράματα που δείχνουν πώς συμπεριφέρεται ο κώδικας σε δεδομένες συνθήκες
  - Αυτό που τελικά ζητά και ο πελάτης
- Επιλογή περιπτώσεων ελέγχου
  - Ποια δεδομένα εισόδου θα χρησιμοποιηθούν;
  - Πώς επιλέγουμε τις περιπτώσεις ελέγχου;
    - Ειδικά στην περίπτωση που ακολουθούμε προσέγγιση «λευκού» κουτιού

Πληρότητα Ελέγχου

- Κάλυψη εντολών (C0)
  - Κάθε εντολή θα πρέπει να εκτελεστεί τουλάχιστον μία φορά
- Κάλυψη διακλαδώσεων (C1)
  - Κάθε εντολή διακλάδωσης θα πρέπει να εκτελεστεί τουλάχιστον μία φορά προς κάθε δυνατή κατεύθυνση (απόφαση)
- Κάλυψη μονοπατιών
  - Κάθε διακριτό μονοπάτι εκτελείται μία τουλάχιστον φορά
    - Διακριτό μονοπάτι: Ένα μονοπάτι που βάζει τουλάχιστον μία νέα ακμή σε σχέση με τα προηγούμενα
    - Θυμάστε την κυκλωματική πολυπλοκότητα; (χρησιμοποιήστε τη!)
- Τυχαίος έλεγχος
  - Όχι και τόσο αποδοτικός, δεν προσφέρει εγγυήσεις
- Έλεγχος στα όρια
  - Συχνά εκεί συμβαίνουν σφάλματα

Κάλυψη Μονοπατιών: Παράδειγμα





- Μονοπάτια:
  - 1, 11
  - 1, 2,3, 6, 7, 9, 10, 1, 11
  - 1, 2,3, 6, 8, 9, 10, 1, 11
  - 1, 2,3, 4,5, 10, 1, 11

Παράδειγμα:

```
read hr1 min1 AmOrPm1
read hr2 min2 AmOrPm2
if (hr1 == 12)
      hr1 = 0
if (hr2 == 12)
      hr2 = 0
if (AmOrPm1 == pm)
      hr1 = hr1 + 12
if (AmOrPm2 == pm)
      hr2 = hr2 + 12
if ( min2 < min1)
      min2 = min2 + 60
      hr2 = hr2 – 1
if( hr2 < hr1)
      hr2 = hr2 + 24
elapsed = min2 – min1 + 60* (hr2 – hr1)
print elapsed
```

- Βρείτε περιπτώσεις ελέγχου για τον ακόλουθο κώδικα ώστε να επιτυγχάνεται κάλυψη C0 και C1:

C0		
Χρόνος έναρξης	**Χρόνος Λήξης**	**Αναμενόμενο Elapsed**
12:00 pm	12:40 pm	40
9:57 pm	11:40 pm	103
5:00 pm	4:00 am	660

C1		
Χρόνος έναρξης	**Χρόνος Λήξης**	**Αναμενόμενο Elapsed**
Όλα τα παραπάνω και επιπλέον…		
8:00 am	12:40 pm	280

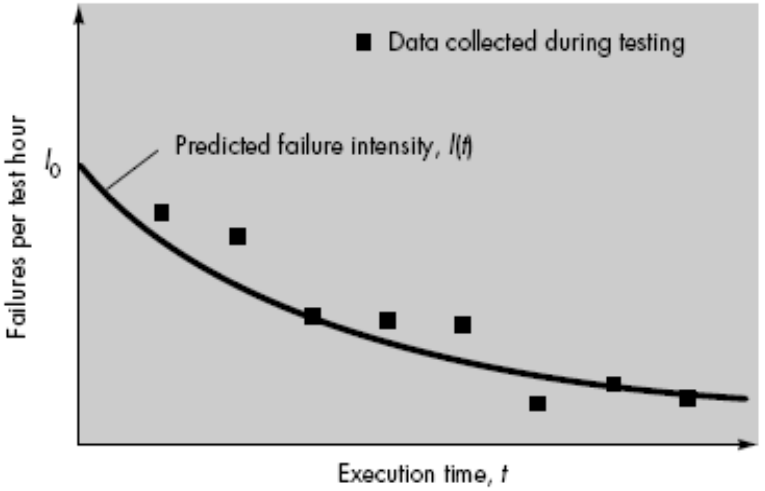
Και Πού να Σταματήσει;

- Πότε τελειώσαμε με τον έλεγχο; Πότε δοκιμάσαμε αρκετά;
  - Ποτέ, απλά μεταφέρεις το βάσανο στον πελάτη
  - Όταν τελειώσει ο χρόνος ή τα χρήματα
- Ναι, υπάρχουν και πιο ρεαλιστικές προσεγγίσεις…

Στατιστικές Μέθοδοι

$$f(t)=(1/p)\ln(l_0 pt+1)$$

- f(t): συνολικός αριθμός προβλημάτων που αναμένεται να έχουν εντοπιστεί μετά από t χρονικές μονάδες ελέγχου
- l_0: η αρχική «ένταση» προβλημάτων (στην αρχή του ελέγχου)
- p: Ο ρυθμός μείωσης της «έντασης» προβλημάτων



$$l(t)=l_0/(l_0 pt+1)$$

- l(t): Στιγμιαία ένταση προβλημάτων
- Μπορώ να σταματήσω όταν η στιγμιαία ένταση προβλημάτων γίνει ικανοποιητικά μικρή

Εμφύτευση Σφαλμάτων

- Συνειδητά εισάγονται σφάλματα στον κώδικα
- Το ποσοστό των εμφυτευμένων σφαλμάτων που παραμένουν υποθέτουμε ότι είναι ίσο με το ποσοστό των εγγενών σφαλμάτων που παραμένουν
  - Είναι όμως τα εμφυτευμένα σφάλματα της ίδιας φύσης και πολυπλοκότητας;
    - Δεν ξέρουμε τι πολυπλοκότητας ή φύσης είναι τα εγγενή σφάλματα
    - Ίσως αν χρησιμοποιήσουμε ιστορική πληροφορία;

Εμφύτευση Σφαλμάτων

$$N = n \frac{S}{s}$$

- N: Αριθμός εγγενών σφαλμάτων
- S: Αριθμός εμφυτευμένων σφαλμάτων
- n: Αριθμός εγγενών σφαλμάτων που βρέθηκαν
- s: Αριθμός εμφυτευμένων σφαλμάτων που βρέθηκαν

Εμπιστοσύνη στο Λογισμικό

- C: Βαθμός (ποσοστό) εμπιστοσύνης στο λογισμικό
  - Πιθανότητα το λογισμικό να μην περιέχει πάνω από R εναπομείναντα σφάλματα

$$C=\begin{cases} 1 & n>R \\ \dfrac{S}{S+R+1} & n\le R \end{cases}$$

- Αν δεν έχουν βρεθεί όλα τα εμφυτευμένα σφάλματα;

$$C=\begin{cases} 1 & n>R \\ \dfrac{\binom{S}{s-1}}{\binom{S+R+1}{R+s}} & n\le R \end{cases}$$

Ανεξάρτητες Ομάδες Ελέγχου

- x: Αριθμός εμφυτευμένων σφαλμάτων που εντοπίστηκαν από την ομάδα ελέγχου 1
- y: Αριθμός εμφυτευμένων σφαλμάτων που εντοπίστηκαν από την ομάδα ελέγχου 2.
- q: Αριθμός κοινών εντοπισθέντων εμφυτευμένων σφαλμάτων
  - q<=x, q<=y

- Αποτελεσματικότητα ομάδων ελέγχου:
  - $E_1 = x/N$, $E_2 = y/N$
  - Όμως μπορούμε να θεωρήσουμε ότι $E_1 = q/y$ και $E_2 = q/x$

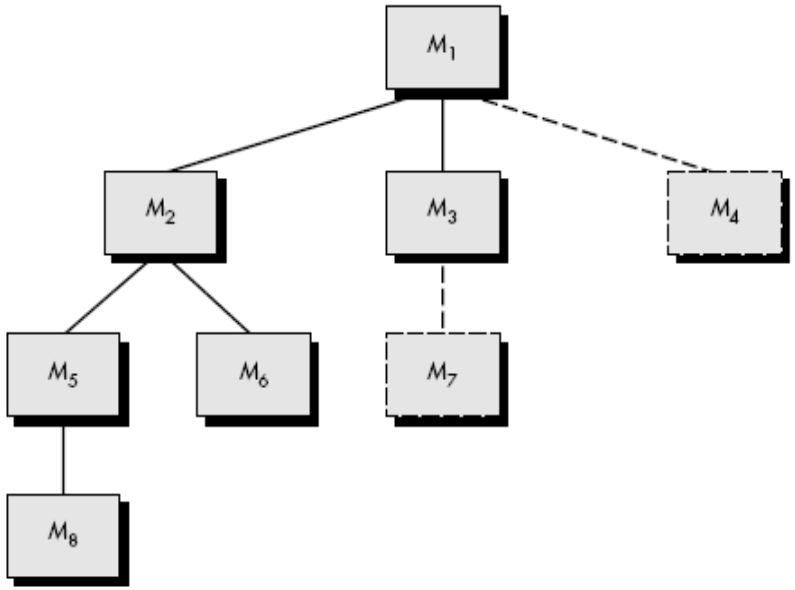
- Τότε, $N = q/(E1*E2)$

Έλεγχος Ολοκλήρωσης

- Γιατί χρειάζεται;
- Μη αυξητική ολοκλήρωση
  - Προσέγγιση Big-Bang
  - Εξέταση συνολικά του προγράμματος
    - Συνήθως χάος
    - Δεν μπορούν να απομονωθούν τα αίτια των προβλημάτων
    - Ενίοτε για κάθε λάθος που διορθώνεται εμφανίζεται ένα άλλο
- Αυξητική ολοκλήρωση
  - Από πάνω προς τα κάτω
  - Από κάτω προς τα πάνω

Προσέγγιση από Πάνω προς τα Κάτω

- Από τη main()…
  - … προς τα υποκείμενα τμήματα
- Κατά βάθος
  - Αρχίζουμε από ένα «ισχυρό» μονοπάτι ελέγχου
    - Ανάλογα με τη δομή του προγράμματος
    - Προσθέτουμε τμήματα
- Ή κατά πλάτος
  - Συμπληρώνουμε κάθε επίπεδο

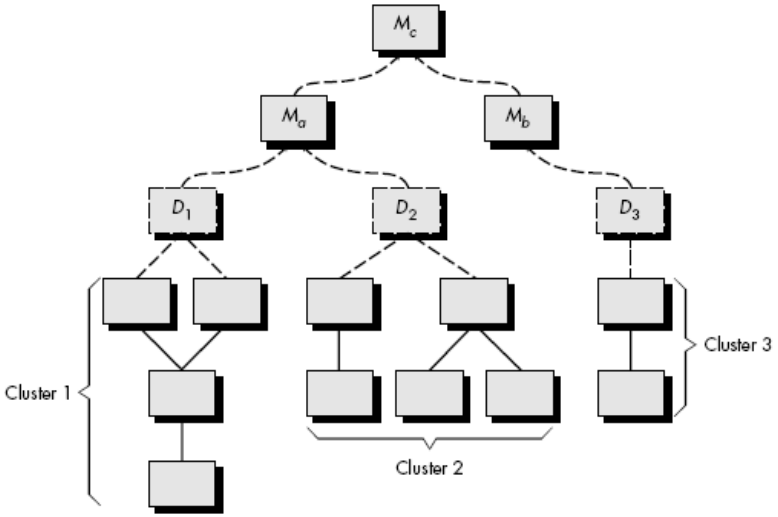


Προσέγγιση από Πάνω προς τα Κάτω: Διαδικασία

1. Βάζουμε το βασικό τμήμα ελέγχου και αντικαθιστούμε τα υπόλοιπα τμήματα με "placebo" (χαζά, πρακτικά άδεια τμήματα)
2. Ανάλογα με τη στρατηγική ολοκλήρωσης (κατά πλάτος ή βάθος) αντικαθιστούμε ένα "placebo" με πραγματικό τμήμα
3. Πραγματοποιούμε ελέγχους κατά την εισαγωγή κάθε τμήματος
4. Πίσω στο 2
5. Παλινδρομικός Έλεγχος!

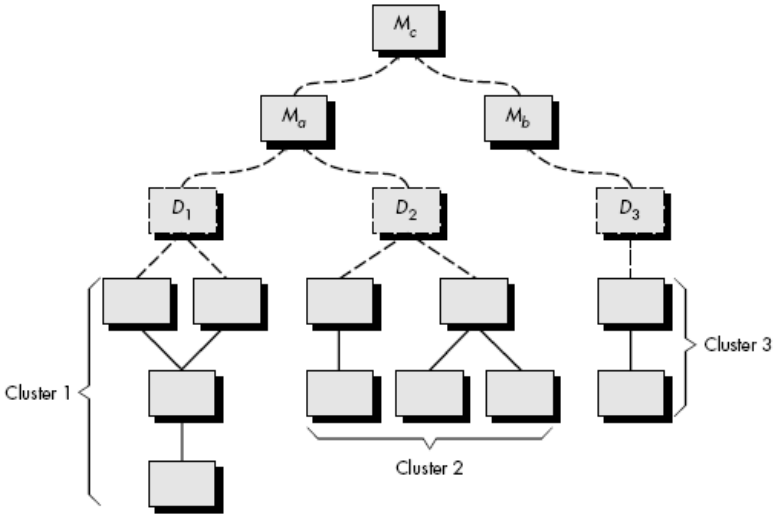
Προσέγγιση από Κάτω προς τα Πάνω

- Ξεκινάμε από τα τμήματα στα χαμηλότερα επίπεδα της δομής του προγράμματος
- … και πηγαίνουμε προς τα πάνω
  - Δε χρειαζόμαστε “placebo”



Προσέγγιση από Κάτω προς τα Πάνω: Διαδικασία

1. Τμήματα χαμηλού επιπέδου οργανώνονται σε «συστάδες» που επιτελούν συγκεκριμένο έργο
2. Εισαγωγή ενός τμήματος «οδηγού» που ελέγχει είσοδο και έξοδο
3. Έλεγχος συστάδας
4. Αφαίρεση οδηγού, εισαγωγή παραπάνω συστάδων



Παλινδρομικός Έλεγχος

- Με την εισαγωγή κάθε τμήματος το λογισμικό αλλάζει
- Αντίστοιχα, αλλαγές με τη διόρθωση σφαλμάτων
- Πιθανά προβλήματα σε τμήματα που πριν δούλευαν σωστά
- Παλινδρομικός έλεγχος:
  - Επανάληψη, σε κάθε βήμα, ενός υποσυνόλου των ελέγχων που έχουν ήδη πραγματοποιηθεί
    - Άσκηση ισορροπίας μεταξύ πλήθους ελέγχων που επανεκτελούνται και αποδοτικότητας.