Τεχνολογία λογισμικού στην πράξη
Έλεγχος λογισμικού

Διομήδης Σπινέλλης
Τμήμα Διοικητικής Επιστήμης και Τεχνολογίας
Οικονομικό Πανεπιστήμιο Αθηνών

dds@aueb.gr
http://www.dmst.aueb.gr/dds
@CoolSWEng

2019-11-17

Άσκηση (Έλεγχος λογισμικού)

- Σε ένα σημαντικό-δημοφιλές έργο λογισμικού:
  - να αναγνωρίσετε τα επίπεδα υλοποίησης ελέγχων,
  - να εξετάσετε τις τεχνικές ελέγχου που χρησιμοποιεί,
  - να αναδείξετε μετρικές που αξιολογούν την πληρότητα των ελέγχων για το συγκεκριμένο έργο με βάση κατάλληλα χαρακτηριστικά του έργου (π.χ. μέγεθος κώδικα, καταχωρημένα λάθη, αριθμός σεναρίων ελέγχου κ.α.), και
  - να προτείνετε συμπληρωματικές τεχνικές ελέγχου και κατάλληλες κατηγορίες εργαλείων για το συγκεκριμένο έργο.

Ορισμός

Δυναμική επαλήθευση ότι το πρόγραμμα έχει την *αναμενόμενη* συμπεριφορά σε ένα κατάλληλα *επιλεγμένο πεπερασμένο* σύνολο περιπτώσεων ελέγχου

Ορολογία

- Επαλήθευση (verification): Υλοποιούμε σωστά το λογισμικό;
- Επικύρωση (validation): Υλοποιούμε το σωστό λογισμικό;
- Ελάττωμα (fault) (οδηγεί σε)
- Αποτυχία (failure)

Επίπεδα ελέγχου

- Έλεγχος μονάδας (unit testing)
- Έλεγχος συνένωσης (integration testing)
- Έλεγχος συστήματος (system testing)

Στόχοι ελέγχου

- Έλεγχος αποδοχής (acceptance testing)
- Έλεγχος εγκατάστασης (installation testing)
- Έλεγχος A/B (A/B testing)
- Έλεγχος παλινδρόμησης (regression testing)
- Έλεγχος απόδοσης (performance testing)
- Δοκιμές καταπόνησης (stress testing)
- Έλεγχος ανάκτησης (recovery testing)
- Έλεγχος διεπαφών (interface testing)
- Έλεγχος σχηματισμών (configuration testing)
- Έλεγχος χρηστικότητας (usability testing)

Παράδειγμα: Έλεγχος A/B

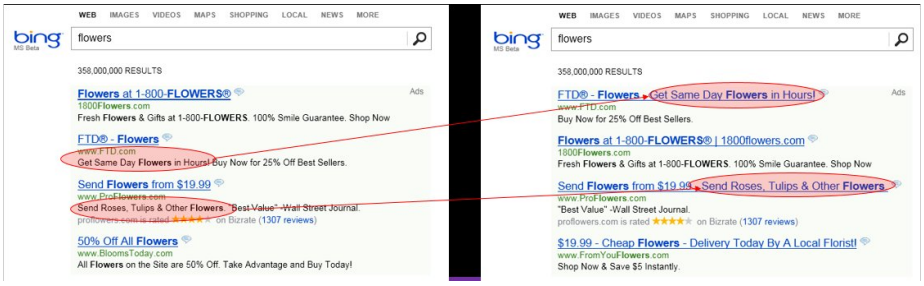

Figure 1: A/B testing example

(Από παρουσίαση του Ronny Kohavi.)

Χρήσιμες τεχνικές ελέγχου

- Ad hoc
- Διερευνητικά
- Ακραίες τιμές
- Τυχαίες τιμές
- Κάλυψη των ροών ελέγχου
- Κάλυψη των ροών δεδομένων
- Εικασία λαθών

Χρήσιμα δεδομένα ελέγχου

- Ιστορικό ελαττωμάτων
- Κατηγοριοποίηση ελαττωμάτων
- Αριθμός ελαττωμάτων σε σχέση με κώδικα
- Προσδιορισμός αξιοπιστίας από τις αποτυχίες

- Τερματισμός ελέγχου με βάση την αξιοπιστία

Αποτίμηση προόδου των ελέγχων

- Με βάση την κάλυψη
- Με βάση την εύρεση τεχνητών ελαττωμάτων

Πρακτικές ελέγχου

- Αποδοχή ελαττωμάτων ως επιτυχία
- Διεργασία και οδηγός ελέγχου
- Τεκμηρίωση
- Ανάπτυξη οδηγούμενη από ελέγχους (TDD)
- Εσωτερική ή εξωτερική ομάδα
- Απόφαση ολοκλήρωσης με βάση αποτίμηση κινδύνου

Ενέργειες ελέγχου

- Σχέδιο ελέγχου
- Δημιουργία περιπτώσεων ελέγχου
- Περιβάλλον ελέγχου
- Εκτέλεση ελέγχου
- Αποτίμηση αποτελεσμάτων
- Αναφορές
- Παρακολούθηση λαθών

Παράδειγμα: Σχέδιο ελέγχου

- Ταυτότητα
- Εισαγωγή
- Στοιχεία προς έλεγχο
- Χαρακτηριστικά που θα ελεγχθούν
- Χαρακτηριστικά που δε θα ελεγχθούν
- Μεθοδολογία
- Κριτήρια αποδοχής
- Κριτήρια αναστολής και απαιτήσεις επανάληψης
- Παραδοτέα
- Εργασίες ελέγχου
- Περιβάλλον ελέγχου
- Ευθύνες (ανά ομάδα)
- Προσωπικό και εκπαίδευση
- Χρονοδιάγραμμα
- Αντιμετώπιση προβλημάτων
- Έγκριση

Παράδειγμα τεκμηρίωσης

- Προδιαγραφές σχεδίου ελέγχου
- Προδιαγραφές περίπτωσης ελέγχου
- A
- T
- K
- K
- A
- Δ
- E
- Προδιαγραφές διαδικασίας διεκπεραίωσης ελέγχου
- Έκθεση διαβίβασης οντοτήτων προς έλεγχο
- Ημερολόγιο ελέγχου
- Περιληπτική έκθεση ελέγχου

Εργαλεία ελέγχου

- Ζυγός ελέγχου (test harness)
- Γεννήτρια ελέγχου
- Εργαλεία εγγραφής και επανάληψης
- Προγράμματα σύγκρισης
- Συστήματα εκτέλεσης ελέγχου
- Εργαλεία ελέγχου κάλυψης
- Εργαλεία εκτέλεσης ελέγχου μονάδας
- Εργαλεία ιχνηλάτησης
- Εργαλεία ελέγχου παλινδρόμησης

Παράδειγμα: έλεγχος κάλυψης με gcov

```
     1    if (*++argv && !strcmp(*argv, "-n")) {
######            ++argv;
######            nflag = 1;
          }
          else
     1            nflag = 0;
     3    while (*argv) {
     2            (void)printf("%s", *argv);
     2            if (*++argv)
```

```
1                       putchar(' ');
        }
1       if (!nflag)
1               putchar('\n');
1       exit(0);
```

Παράδειγμα: Κάλυψη εντολών (Perl)

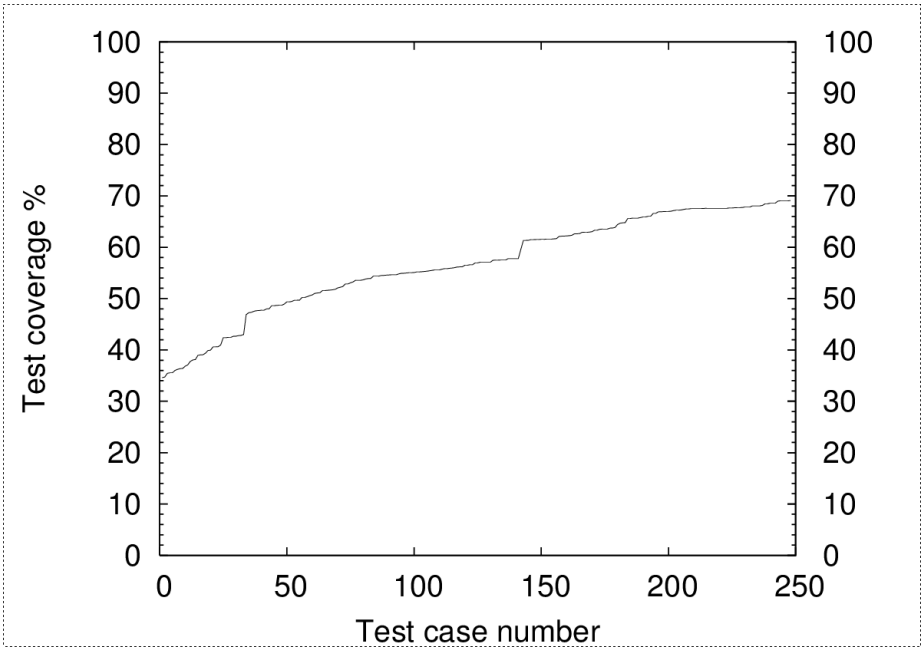

Figure 2: Perl statement coverage

Παράδειγμα: Κάλυψη διακλαδώσεων (Perl)

Παράδειγμα: Έλεγχοι μονάδας

```
public class MoneyTest extends TestCase {
    private Money f12CHF;
    private Money f14CHF;
    private Money f28USD;

    protected void setUp() {
        f12CHF= new Money(12, "CHF");
        f14CHF= new Money(14, "CHF");
        f28USD= new Money(28, "USD");
```

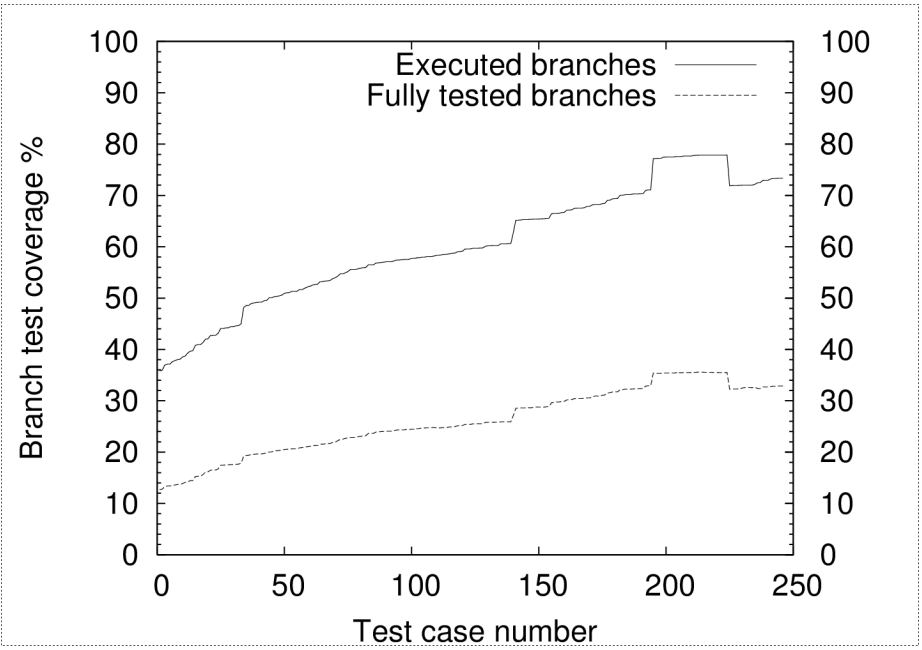

Figure 3: Perl branch coverage

```
    }
}

public void testMoneyMoneyBag() {
    // [12 CHF] + [14 CHF] + [28 USD] == {[26 CHF][28 USD]}
    Money bag[]= { f26CHF, f28USD };
    MoneyBag expected= new MoneyBag(bag);
    assertEquals(expected, f12CHF.add(f28USD.add(f14CHF)));
}

public static Test suite() {
    TestSuite suite= new TestSuite();
    suite.addTest(new MoneyTest("testMoneyEquals"));
    suite.addTest(new MoneyTest("testSimpleAdd"));
    return suite;
}
```

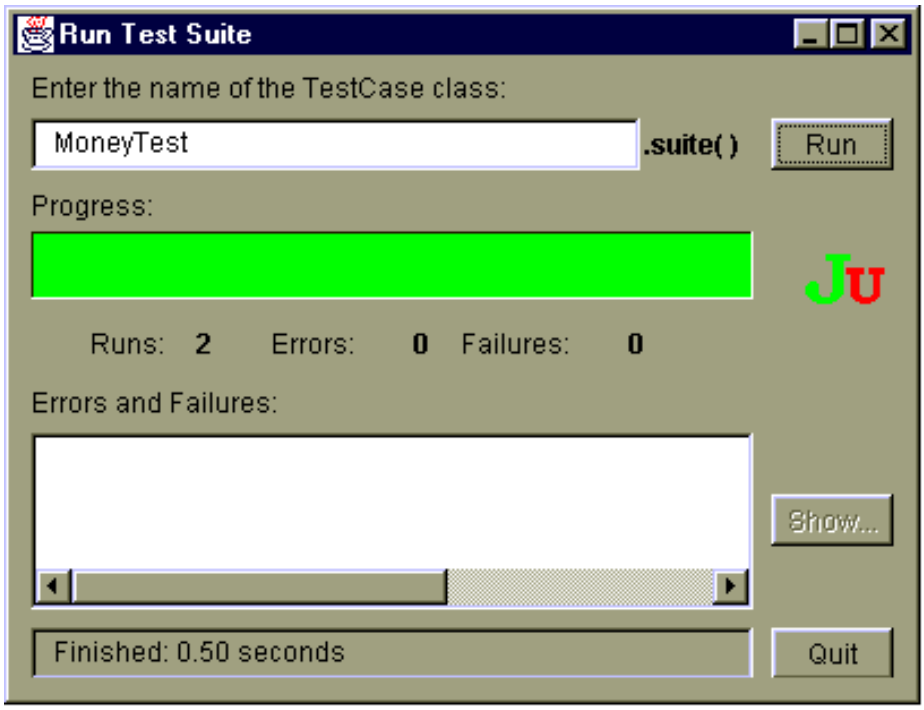

Figure 4: JUnit execution

Παράδειγμα: JUnit

Άσκηση (Συντήρηση λογισμικού)

- Επιλέξτε ένα δημοφιλές-σημαντικό έργο ανοιχτού λογισμικού και μελετήστε τη συντήρησή του:
  - αναγνωρίστε ενέργειες συντήρησης και αντίστοιχες τεχνικές,
  - προτείνετε κατάλληλες μετρικές για την αξιολόγηση της παρούσας συντήρησης του έργου,
  - αναγνωρίστε ανάγκες για περαιτέρω συντήρηση και αντίστοιχες τεχνικές, και
  - ταξινομήστε στις κατηγορίες συντήρησης τις ενέργειες και ανάγκες που αναγνωρίσατε.

Πρόσθετες πηγές

- Kent Beck: Test Desiderata