Tampereen yliopisto
Tampere University

Ohjelmistojen testaus

10.9.2019

Tampereen yliopisto
Tampere University

Mitä testaus on?

- Testaus on **tiedon tuottamista**:
  - testaamaton koodi oletetaan rikkinäiseksi
  - testauksesta saadaan tietoa koodin laadusta päätöksentekoa varten
- Miksi ohjelma pitää testata?

Miksi testataan?

- Testaamalla voidaan osoittaa, että
  - Ohjelma tekee, mitä sen ei pitäisi
  - Ohjelma ei tee, mitä sen pitäisi
  - Ohjelma toimii määrittelyn vastaisella tavalla (kumpi on väärässä?)
  - Ohjelmisto on hankala ymmärtää, vaikeakäyttöinen, hidas tai toimii käyttäjän mielestä väärin
- Kaikkea ei voi testata

Tampereen yliopisto
Tampere University

Tavoite: virheen paljastaminen

- Oletus: ohjelmassa on aina virheitä, ne pitää vain löytää
- Lähtökohta: onnistunut testiajo aiheuttaa häiriön ohjelman toiminnassa
  - Virheen poistaminen parantaa laatua
  - Voidaan selvittää, mistä virheet johtuvat: Juurisyyt ja tekninen velka

Keskeisiä termejä

- **Virhe** (error): poikkeama spesifikaatiosta
- **Vika** (fault, defect) aiheutuu virheellisen kohdan suorituksesta tai kun pitäisi suorittaa jotakin, mitä ei ole toteutettu
- **Häiriö** (failure): viasta johtuva näkyvä tapahtuma ulkoisessa toiminnassa

- **Bugi** (bug) voi tarkoittaa mitä tahansa edellisistä

Kuva: Joeks (**CC BY-NC 2.0**)

Tapoja testata

- **Dynaaminen testaus**: ohjelmaa ajetaan sopivilla syötteillä
- **Staattinen testaus**: virheitä yritetään löytää tutkimalla lähdekoodia tai dokumentaatiota
- **Positiivinen testaus:** “happy case” –tyyppisiä testitapauksia; yritetään varmistaa, että testattava järjestelmä tekee mitä pitäisi
- **Negatiivinen testaus:** **“**unhappy case” –testejä eli tilanteita, joista vaatimukset eivät sano (juuri) mitään, virhetilanteita, yms

Miten testaat?

Kuva: Zhao! **(CC BY 2.0)**

Hyvä testitapaus

- Pieni testi ohjelman käyttäytymiselle:
  - Mitä testataan? Esim. laskimen jakolasku
  - Mikä on hyvä syöte? Esim. jako nollalla
  - Odotettu tulos? Esim. virheilmoitus, ohjelma ei kaadu, jotain muuta
- Suunnitellaan joko ennen suoritusta tai “lennosta”

Tampereen yliopisto
Tampere University

Testitapauksen rakenne

1. Alustus
2. Suorittaminen
3. Tuloksen evaluointi
4. Puhdistus

Yksikkötestaus

Ohjelman testaaminen

- Tavoite: itsensä testaava koodi
  - Kattavat testit osa toimivaa ohjelmaa
  - Jatkuva varmuus siitä, että bugit koodissa löytyvät
  - Regression välttäminen

Kuva: Martin Fowler

Yksikkötestaaminen käytännössä

- Mitä testataan?
- Miten testataan?
- Kuka testaa?

Kuva: sleepymyf (CC BY-NC-ND 2.0)

Tampereen yliopisto
Tampere University

Koodari testaajana

- Hyvä koodari osaa testata omat koodinsa
  - Yksikön koodarin vastuulla on yleensä testata oma toteutuksensa
  - Usein lisäksi laadunvarmistustehtäviä muiden koodiin

Kuva: sleepymyf (CC BY-NC-ND 2.0)

Yksikkötestaus

- Osa yksikön toteutusvaihetta: Mitä pikemmin toteutus testataan, sen parempi.
- Rajapinta usein sopiva näkymä (kapselointi)
- Test Driven Development:
  - Luodaan automaattisesti ajettava testi (ja ajetaan se)
  - Koodataan toiminnallisuus ja ajetaan testi
  - Korjataan ja refaktoroidaan koodia

Refaktorointi

- Koodin toiminnallisuus ei muutu
- Rakennetta ja koodin ei-toiminnallista laatua kehitetään
  - tekninen velka pienenee
  - saattaa ratkaista piilossa olevia ongelmia
- Kommentointi

Kommentointi

```
/* if allocation flag is zero */
if ( alloc_flag == 0 )
```

```
/* if allocating new member */
if ( alloc_flag == 0 )
```

```
/* if allocating new member */
if ( alloc_flag == NEW_MEMBER
```

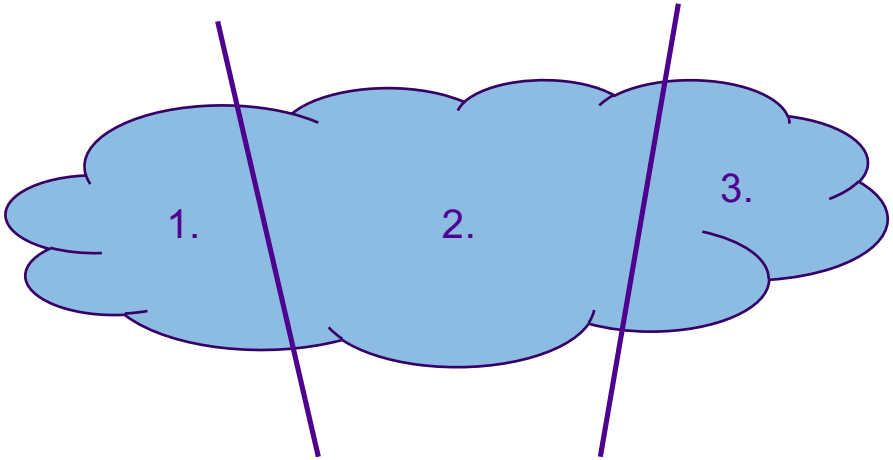
(turha)

Reuna-arvojen testaaminen

- Testataan:
  - Parametrien ja paluuarvojen reuna-arvot
  - Silmukoiden pyörimiskertojen reuna-arvot
  - Tietorakenteisiin liittyvät reuna-arvot

Ekvivalenssiositus

- Syöte jaetaan ekvivalenssiluokkiin
  - Valitse ehto syötteestä
  - Jaa luokkiin
  - Perusjako: sallitut ja kielletyt arvot



Yksikkötestauksen toteuttaminen

- Lyhyitä selkeitä testifunktioita ja niissä yksinkertaisia tarkistuksia
  - Testikehyksen assertiot **testikoodiin**

```
TEST_TYPE test_square_root() {
    double result = my_sqrt(x);
    ASSERT_TRUE((result * result) == x);
    // Pieni bugi testikoodissa (mikä?)
    // yksinkertaistuksen vuoksi...
}
```

Tampereen yliopisto
Tampere University

Yksikkötestauksen toteuttaminen

- Testikoodi on koodia
  - Dokumentoitava
  - Testattava
  - Ylläpidettävä

Yksikkötestaus

1. Mikä on metodille kaikkein tärkeintä
2. Testaa kaikkein yleisimmät tapaukset
3. Ole luova
4. Keskity rajapintaan
5. Tee testeistä niin yksikertaisia kuin mahdollista
6. Käytä testikehystä

Tampereen yliopisto
Tampere University

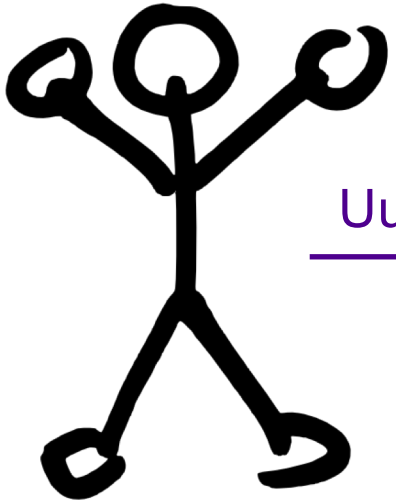
Integrointitestaus

- Yksikkötestauksen jälkeen yksiköt integroidaan isommiksi kokonaisuuksiksi
- Versionhallinnan kautta kehittäjällä koko ohjelma käytettävissä
- Testiautomaatio ja jatkuva integraatio

Tampereen yliopisto
Tampere University

Jatkuva integraatio

Kuva: Jenkins



Created by Swen-Peter Ekkebus
from Noun Project

Uusi koodi

Version-
hallinta

Autom.

CI

- Build
- Unit
- Integration

Tampereen yliopisto
Tampere University

Mitä testataan?

- Ekvivalenssiositus
- Raja-arvoanalyysi
- Staattinen analyysi

OLE LUOVA!

Tampereen yliopisto
Tampere University

Millaisia virheitä löydetään?

- Syntaksivirheet
- Alustamattomat muuttujat
- Käyttämättömät paluuarvot
- Virheellinen osoitinten käyttö
- Samaa koodia useammassa kuin yhdessä paikassa, kuollut koodi
- Koodin ylläpidettävyys- ja siirrettävyysongelmia