Ohjelmistotuotanto

Matti Luukkainen ja ohjaajat Jami Kousa, Tero Tapio, Mauri Karlin

syksy 2019

Luento 5

11.11.2019

Ohjelmistojen laadunhallinta

- Laadunhallinnan kaksi oleellista kysymystä
  - verifiointi: *are we building the product right*
  - validointi: *are we building the right product*

Ohjelmistojen laadunhallinta

- Laadunhallinnan kaksi oleellista kysymystä
  - verifiointi: *are we building the product right*
  - validointi: *are we building the right product*
- Verifioinnissa varmistetaan, että ohjelmisto toteuttaa vaatimusmäärittelyn aikana asetetut vaatimukset
  - Testataan toiminnalliset ja ei-toiminnalliset vaatimukset

Ohjelmistojen laadunhallinta

- Laadunhallinnan kaksi oleellista kysymystä
  - verifiointi: *are we building the product right*
  - validointi: *are we building the right product*
- Verifioinnissa varmistetaan, että ohjelmisto toteuttaa vaatimusmäärittelyn aikana asetetut vaatimukset
  - Testataan toiminnalliset ja ei-toiminnalliset vaatimukset
- Validointi varmistetaan, että ohjelmisto täyttää käyttäjän odotukset
  - Vaatimusmäärittelyn aikana kirjatut vaatimukset eivät ole aina se mitä käyttäjä todella tarvitsee

Verifiointi ja Validointi

- Tavoitteena on varmistaa että ohjelma on “riittävän hyvä” käyttötarkoitukseensa
  - Hyvyys on suhteellista ja riippuu ohjelman käyttötarkoituksesta
  - Ohjelman ei yleensä tarvitse olla virheetön ollakseen riittävän hyvä käytettäväksi

Verifiointi ja Validointi

- Tavoitteena on varmistaa että ohjelma on “riittävän hyvä” käyttötarkoitukseensa
  - Hyvyys on suhteellista ja riippuu ohjelman käyttötarkoituksesta
  - Ohjelman ei yleensä tarvitse olla virheetön ollakseen riittävän hyvä käytettäväksi
- Verifioinnin ja validoinnin suorittamista käytetään yleisesti nimitystä *laadunhallinta* (engl. quality assurance, QA)
- Jos laadunhallinta on erillisen tiimin vastuulla, käytetään tästä usein nimitystä *QA-tiimi*

Verifioinnin ja Validoinnin tekniikat

- Verifioinnissa käytetään kahta erilaista tekniikkaa
  - Katselmointeja
  - Testausta

Verifioinnin ja Validoinnin tekniikat

- Verifioinnissa käytetään kahta erilaista tekniikkaa
  - Katselmointeja
  - Testausta
- Katselmoinneissa (review) käydään läpi tuotantoprosessin aikana tehtyjä dokumentteja ja ohjelmakoodia, ja etsitään näistä ongelmia

Verifioinnin ja Validoinnin tekniikat

- Verifioinnissa käytetään kahta erilaista tekniikkaa
  - Katselmointeja
  - Testausta
- Katselmoinneissa (review) käydään läpi tuotantoprosessin aikana tehtyjä dokumentteja ja ohjelmakoodia, ja etsitään näistä ongelmia
- Katselmointi on *staattinen tekniikka*, suorituskelpoista ohjelmakoodia ei tarvita
  - Jos katselmoinnin kohteena on ohjelmakoodi, ei sitä katselmoinnissa suoriteta

Verifioinnin ja Validoinnin tekniikat

- Verifioinnissa käytetään kahta erilaista tekniikkaa
  - Katselmointeja
  - Testausta
- Katselmoinneissa (review) käydään läpi tuotantoprosessin aikana tehtyjä dokumentteja ja ohjelmakoodia, ja etsitään näistä ongelmia
- Katselmointi on *staattinen tekniikka*, suorituskelpoista ohjelmakoodia ei tarvita
  - Jos katselmoinnin kohteena on ohjelmakoodi, ei sitä katselmoinnissa suoriteta
- Testaus on *dynaaminen tekniikka*, joka edellyttää ohjelmakoodin suorittamista
  - tarkkaillaan miten ohjelma reagoi annettuihin testisyötteisiin

Vaatimusten validointi

- Ohjelmistolle määritellyt vaatimukset on validoitava:
  - varmistettava, että määrittelydokumentti määrittelee oikeanlaisen ohjelmiston

Vaatimusten validointi

- Ohjelmistolle määritellyt vaatimukset on validoitava:
  - varmistettava, että määrittelydokumentti määrittelee oikeanlaisen ohjelmiston
- Vesiputousmallissa määrittelydokumentin kirjattujen vaatimusten validointi suoritetaan *katselmoimalla*
  - Vaatimusmäärittelyn lopuksi asiakas tarkastaa määrittelydokumenttiin kirjattujen vaatimuksien vastaavan asiakkaan kuvaa tilattavasta järjestelmästä
  - Katselmoinnin jälkeen määrittelydokumentti jäädytetään ja sen muuttaminen vaatii yleensä monimutkaista prosessia

Vaatimusten validointi

- Ohjelmistolle määritellyt vaatimukset on validoitava:
  - varmistettava, että määrittelydokumentti määrittelee oikeanlaisen ohjelmiston
- Vesiputousmallissa määrittelydokumentin kirjattujen vaatimusten validointi suoritetaan *katselmoimalla*
  - Vaatimusmäärittelyn lopuksi asiakas tarkastaa määrittelydokumenttiin kirjattujen vaatimuksien vastaavan asiakkaan kuvaa tilattavasta järjestelmästä
  - Katselmoinnin jälkeen määrittelydokumentti jäädytetään ja sen muuttaminen vaatii yleensä monimutkaista prosessia
- Ketterässä ohjelmistotuotannossa vaatimusten validointi tapahtuu sprinttien päättävien demonstraatioiden yhteydessä
  - Asiakkaalle näytetään ohjelman toimivaa versiota
  - Asiakas voi itse verrata onko lopputulos haluttu
  - Jos ei, on seuraavassa sprintissä mahdollista ottaa korjausliike

Koodin katselmointi

- Koodin katselmointi eli lukeminen jonkun muun kuin ohjelmoijan toimesta on havaittu tehokkaaksi keinoksi laadun parantamisessa

Koodin katselmointi

- Koodin katselmointi eli lukeminen jonkun muun kuin ohjelmoijan toimesta on havaittu tehokkaaksi keinoksi laadun parantamisessa
- Katselmoinnin avulla voidaan havaita koodista ongelmia, joita testauksella ei välttämättä havaita, esim.
  - noudattaako koodi sovittua tyyliä
  - onko koodi ylläpidettävää
  - onko koodissa tietoturvan kannalta vaarallisia piirteitä

Koodin katselmointi

- Koodin katselmointi eli lukeminen jonkun muun kuin ohjelmoijan toimesta on havaittu tehokkaaksi keinoksi laadun parantamisessa
- Katselmoinnin avulla voidaan havaita koodista ongelmia, joita testauksella ei välttämättä havaita, esim.
  - noudattaako koodi sovittua tyyliä
  - onko koodi ylläpidettävää
  - onko koodissa tietoturvan kannalta vaarallisia piirteitä
- Perinteisesti käyty läpi onko koodissa tiettyjä checklisteissä listattuja riskialttiita piirteitä

V1	Variables are lower case words separated by underscores (`this_is_a_var`)
V2	Constants are upper case words separated by underscores. (`THIS_IS_A_CONST`)
V3	No **int** declarations. (Use **uint8, int8, uint16, int16, uint32, int32** instead.)
V4	One declaration per line. (Exception: Highly coupled variables, i.e. `width` and `height`.)
V5	All declarations have a comment after them explaining the variable.
V6	Units are declared when appropriate.
V7	No hidden variables. That is, a variable defined in an inner block may not have the same name as variable in an outer block.
V8	Never use "O" (Capital O) or "l" (lower case "l") for variable or constant names.

- Joissakin kielissä, esim. Javassa kääntäjän tuki tekee osan näistä tarkistuksista turhaksi

Staattinen analyysi

- Nykyään on tarjolla katselmointia automatisoivia *staattista analyysiä* tekeviä työkaluja esim. Javalla Checkstyle, Javascriptilla ESlint

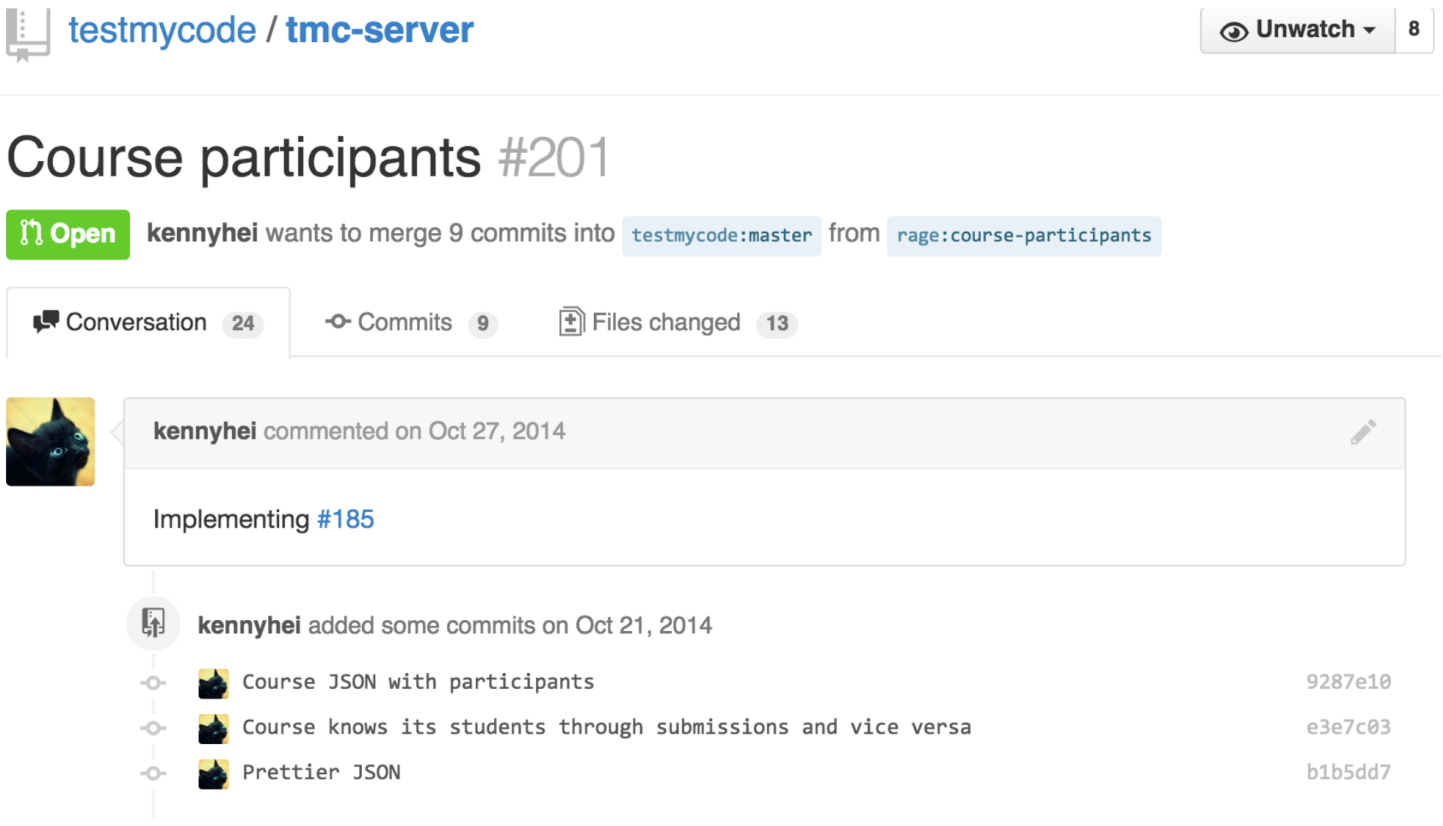
Staattinen analyysi

- Nykyään on tarjolla katselmointia automatisoivia *staattista analyysiä* tekeviä työkaluja esim. Javalla Checkstyle, Javascriptilla ESlint
- Myös pilvipalveluna toimivia staattisen analyysin työkaluja kuten Codeclimate
  - Suorittavat tarkastukset aina kun uutta koodia pushataan GitHubiin
  - Huomaavat koodin laadun muutoksista, esim. jos koodin kompleksisuus kasvaa muutosten yhteydessä

Koodin katselmointi: GitHub ja pull requestit

- GitHubin *pull requestit* tarjoavat hyvän työkalun koodikatselmointiin
- Työn kulku on seuraava
  - Sovelluskehittäjä forkkaa repositorin itselleen, tekee muutokset omaan repositorioon ja tekee pull requestin
  - Joku, esim. *senior developer* tekee katselmoinnin pull requestille
  - Jos koodi ei ole riittävän hyvää, annetaan pull requestin tekijälle parannusehdotuksia
  - Muutosten ollessa hyväksyttäviä, pull request mergetään päärepositorioon

Pullrequest TMC:hen



lib/course_info.rb

View full changes

```
@@ -31,6 +32,17 @@ def course_data(course)
31 32      })
32 33    end
33 34
   35 +  # Course JSON with participants
   36 +  def course_participants_data(course)
   37 +    participants = course.users
   38 +
   39 +    data = {
   40 +      :id => course.id,
   41 +      :name => course.name,
   42 +      :participants => participants.map {|participant| participant_data(participar
```

mpartel added a note on Oct 29, 2014 — Owner

On my desktop, with the mooc production DB dump, this takes around 30 seconds for the k2014-mooc course. I'd really like to avoid adding more really slow queries to TMC.

Would the following make sense?

- Let this only return a list of participants and their newest submission IDs.
- Load a user's exercise statuses on demand, and cache them either on your side or maybe in TMC until the submission ID changes.
- Consider having the per-user URL support ETags.

Koodin katselmointi ketterissä menetelmissä

- Toisin kuin Scrum, eXtreme Programming eli XP määrittelee useita sovelluskehityksen käytänteitä
- Suuri osa XP:n käytänteistä on hyvin tunnettuja *best practiseja*, vietynä äärimmäiseen (extreme) muotoon

Koodin katselmointi ketterissä menetelmissä

- Toisin kuin Scrum, eXtreme Programming eli XP määrittelee useita sovelluskehityksen käytänteitä
- Suuri osa XP:n käytänteistä on hyvin tunnettuja *best practiseja*, vietynä äärimmäiseen (extreme) muotoon
- Osa käytänteistä tähtää laadun maksimoimiseen ja kolmen voidaan ajatella olevan katselmoinnin äärimmilleen vietyjä muotoja

Pariohjelmointi

- *Pariohjelmoinnissa* (pair programming) kaksi ohjelmoijaa työskentelee yhdessä samalla koneella
- Koodia kirjoittava osapuoli toimii *ohjaajana* (driver) ja toinen *navigoijana* (navigator), roolia vaihdetaan sopivin väliajoin
  - Navigoija tekee koodiin jatkuvaa katselmointia

Pariohjelmointi

- *Pariohjelmoinnissa* (pair programming) kaksi ohjelmoijaa työskentelee yhdessä samalla koneella
- Koodia kirjoittava osapuoli toimii *ohjaajana* (driver) ja toinen *navigoijana* (navigator), roolia vaihdetaan sopivin väliajoin
  - Navigoija tekee koodiin jatkuvaa katselmointia
- Etuja:
  - Parantaa ohjelmoijien kuria ja työhön keskittymistä
  - Hyvä oppimisen väline: ohjelmoijat oppivat toisiltaan erityisesti noviisit kokeneimmilta, järjestelmän tietyn osan tuntee aina useampi ohjelmoija
  - Todettu vähentävän bugien määrää 15-50%, kokonaisresurssin kulutus nousee hieman

- Pariohjelmointi tai useamman ihmisen versio siitä, *mob-programming* on melko yleistä
- “Määritelmän” mukaista systemaattista pariohjelmointia tehdään aika harvassa paikassa aamusta iltaan

- Pariohjelmointi tai useamman ihmisen versio siitä, *mob-programming* on melko yleistä
- “Määritelmän” mukaista systemaattista pariohjelmointia tehdään aika harvassa paikassa aamusta iltaan
- Yleensä ohjelmoidaan yksin, mutta spontaania pariutumista ja ryhmäytymistä tapahtuu
  - erityisesti teknisesti haasteellisissa koodin osissa
  - tai jos kyse itselle tuntemattomasta osasta koodia

Koodin katselmointi ketterissä menetelmissä

- *Koodin yhteisomistajuus* (collective code ownership): kukaan yksittäinen ohjelmoija ei hallitse yksin mitään kohtaa koodista
  - Kaikilla lupa tehdä muutoksia mihin tahansa kohtaan koodia

Koodin katselmointi ketterissä menetelmissä

- *Koodin yhteisomistajuus* (collective code ownership): kukaan yksittäinen ohjelmoija ei hallitse yksin mitään kohtaa koodista
  - Kaikilla lupa tehdä muutoksia mihin tahansa kohtaan koodia
- Pariohjelmointi tukee yhteisomistajuutta
- Yhteisomistajuudessa on riskinsä: koodia tuntematon voi saada pahaa jälkeä aikaan
  - XP eliminoi riskejä testauksiin liittyvillä käytänteillä

Koodin katselmointi ketterissä menetelmissä

- *Koodin yhteisomistajuus* (collective code ownership): kukaan yksittäinen ohjelmoija ei hallitse yksin mitään kohtaa koodista
  - Kaikilla lupa tehdä muutoksia mihin tahansa kohtaan koodia
- Pariohjelmointi tukee yhteisomistajuutta
- Yhteisomistajuudessa on riskinsä: koodia tuntematon voi saada pahaa jälkeä aikaan
  - XP eliminoi riskejä testauksiin liittyvillä käytänteillä
- *Ohjelmointistandardi* (coding standards): tiimi määrittelee koodityylin, johon kaikki ohjelmoijat sitoutuvat
  - Tyylillä tarkoitetaan nimeämiskäytäntöä, koodin muotoilua ja myös tiettyjä ohjelman rakenteeseen liittyviä seikkoja
- Noudattamista kontrolloidaan osittain automaattisesti staattisen analyysin työkaluilla kuten checkstylellä

Testaus

- Ohjelmien osoittaminen virheettömäksi on mahdotonta
- Testauksen tarkoituksena vakuuttaa asiakas ja kehittäjät siitä, että ohjelmisto on *tarpeeksi hyvä* käytettäväksi

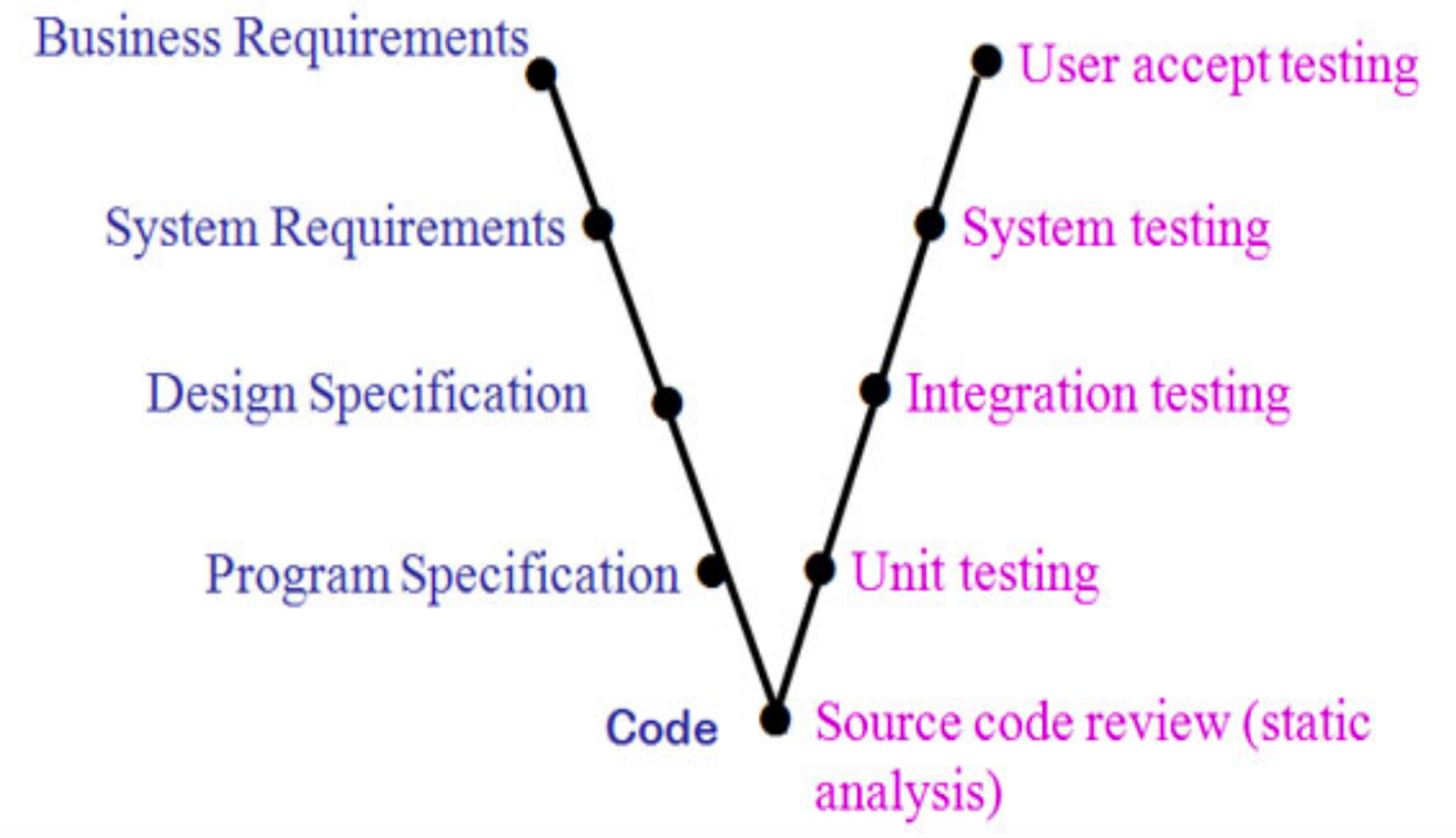
Testaus

- Ohjelmien osoittaminen virheettömäksi on mahdotonta
- Testauksen tarkoituksena vakuuttaa asiakas ja kehittäjät siitä, että ohjelmisto on *tarpeeksi hyvä* käytettäväksi
- Testauksella on kaksi eriävää tavoitetta
  - osoittaa, että ohjelmisto täyttää sille asetetut vaatimukset
  - löytää ohjelmistosta virheitä

Testaus

- Ohjelmien osoittaminen virheettömäksi on mahdotonta
- Testauksen tarkoituksena vakuuttaa asiakas ja kehittäjät siitä, että ohjelmisto on *tarpeeksi hyvä* käytettäväksi
- Testauksella on kaksi eriävää tavoitetta
  - osoittaa, että ohjelmisto täyttää sille asetetut vaatimukset
  - löytää ohjelmistosta virheitä
- Tähtää ohjelman *ulkoisen laadun* (external quality) eli käyttäjän kokeman laadun parantamiseen
  - External Quality is the fitness for purpose of the software. It's most obvious measure is the Functional Tests, and some measure of the bugs that are still loose when the product is released.

Testauksen tasot



- *Yksikkötestaus* (unit testing)
  - Yksittäisten luokkien, metodien ja moduulien testaus erillään muusta kokonaisuudesta

- *Yksikkötestaus* (unit testing)
  - Yksittäisten luokkien, metodien ja moduulien testaus erillään muusta kokonaisuudesta
- *Integraatiotestaus* (integration testing)
  - Yksittäin testattujen komponenttien liittäminen yhteen eli integrointi ja kokonaisuuden testaus

- *Yksikkötestaus* (unit testing)
  - Yksittäisten luokkien, metodien ja moduulien testaus erillään muusta kokonaisuudesta
- *Integraatiotestaus* (integration testing)
  - Yksittäin testattujen komponenttien liittäminen yhteen eli integrointi ja kokonaisuuden testaus
- *Järjestelmätestaus* (system testing)
  - Toimiiko ohjelmisto vaatimuksiin kirjatulla tavalla?
  - Jakautuu useisiin alalajeihin

- *Yksikkötestaus* (unit testing)
  - Yksittäisten luokkien, metodien ja moduulien testaus erillään muusta kokonaisuudesta
- *Integraatiotestaus* (integration testing)
  - Yksittäin testattujen komponenttien liittäminen yhteen eli integrointi ja kokonaisuuden testaus
- *Järjestelmätestaus* (system testing)
  - Toimiiko ohjelmisto vaatimuksiin kirjatulla tavalla?
  - Jakautuu useisiin alalajeihin
- *Käyttäjän hyväksymistestaus* (user acceptance testing)
  - Loppukäyttäjän tuotteelle suorittama testaus

Järjestelmätestaus

- Tarkoitus varmistaa, että järjestelmä toimii vaatimuksiin kirjatulla tavalla
- Kehittäjäorganisaatio suorittaa järjestelmätestauksen

Järjestelmätestaus

- Tarkoitus varmistaa, että järjestelmä toimii vaatimuksiin kirjatulla tavalla
- Kehittäjäorganisaatio suorittaa järjestelmätestauksen

- Testaus tapahtuu ilman tietoa järjestelmän sisäisestä rakenteesta eli kyseessä *black box* -testaus
- Testataan järjestelmää saman rajapinnan kautta, jonka kautta järjestelmää käytetään

Järjestelmätestaus

- Tarkoitus varmistaa, että järjestelmä toimii vaatimuksiin kirjatulla tavalla
- Kehittäjäorganisaatio suorittaa järjestelmätestauksen

- Testaus tapahtuu ilman tietoa järjestelmän sisäisestä rakenteesta eli kyseessä *black box* -testaus
- Testataan järjestelmää saman rajapinnan kautta, jonka kautta järjestelmää käytetään

- Perustuu järjestelmän potentiaalisiin käyttöskenaarioihin
  - jos vaatimukset ilmaistu user storyina, on niistä helppo muotoilla testejä
  - jos storyillä hyväksymiskriteeriot, tilanne on vieläkin parempi
- Kutsutaan myös *toiminnallisiksi testeiksi* (functional test)

Järjestelmätestaus

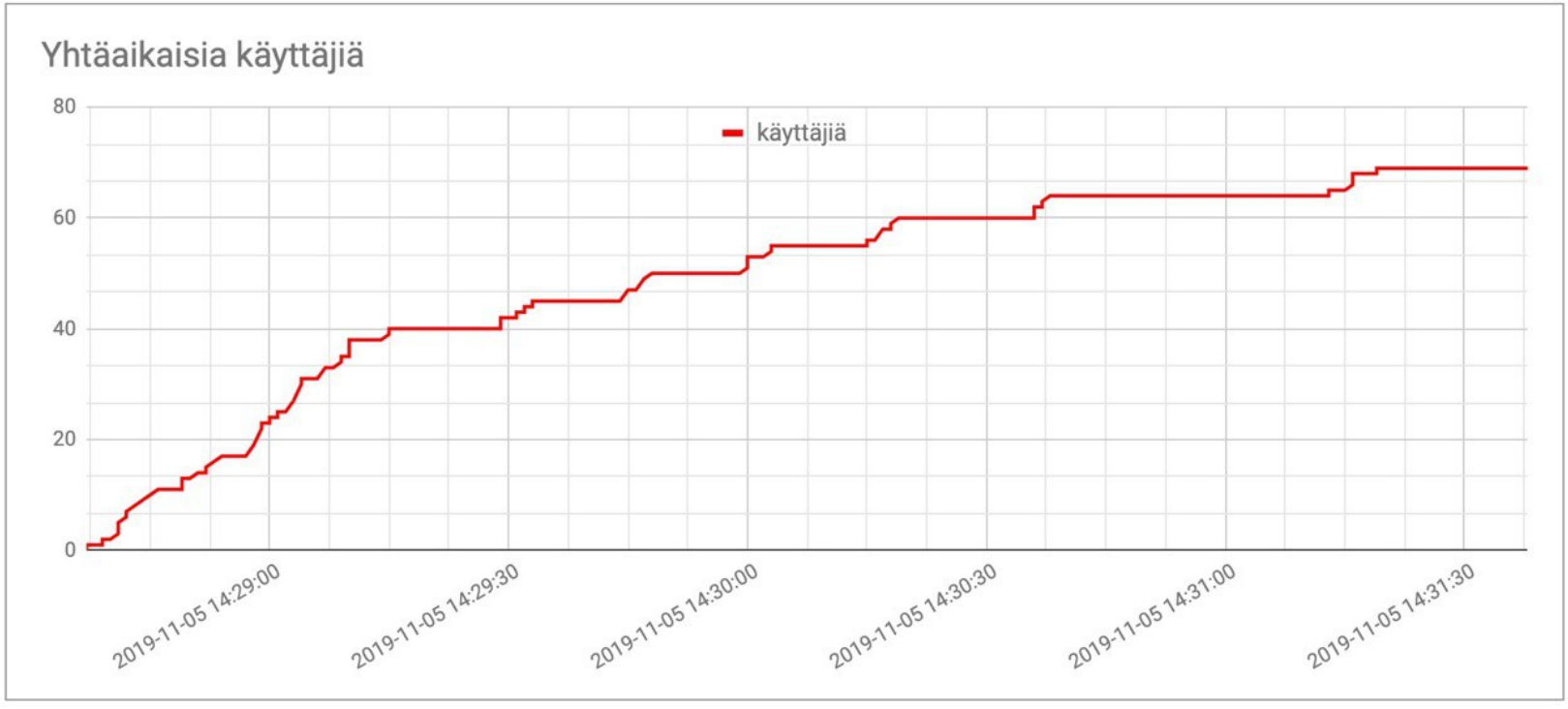
- Tarkoitus varmistaa, että järjestelmä toimii vaatimuksiin kirjatulla tavalla
- Kehittäjäorganisaatio suorittaa järjestelmätestauksen

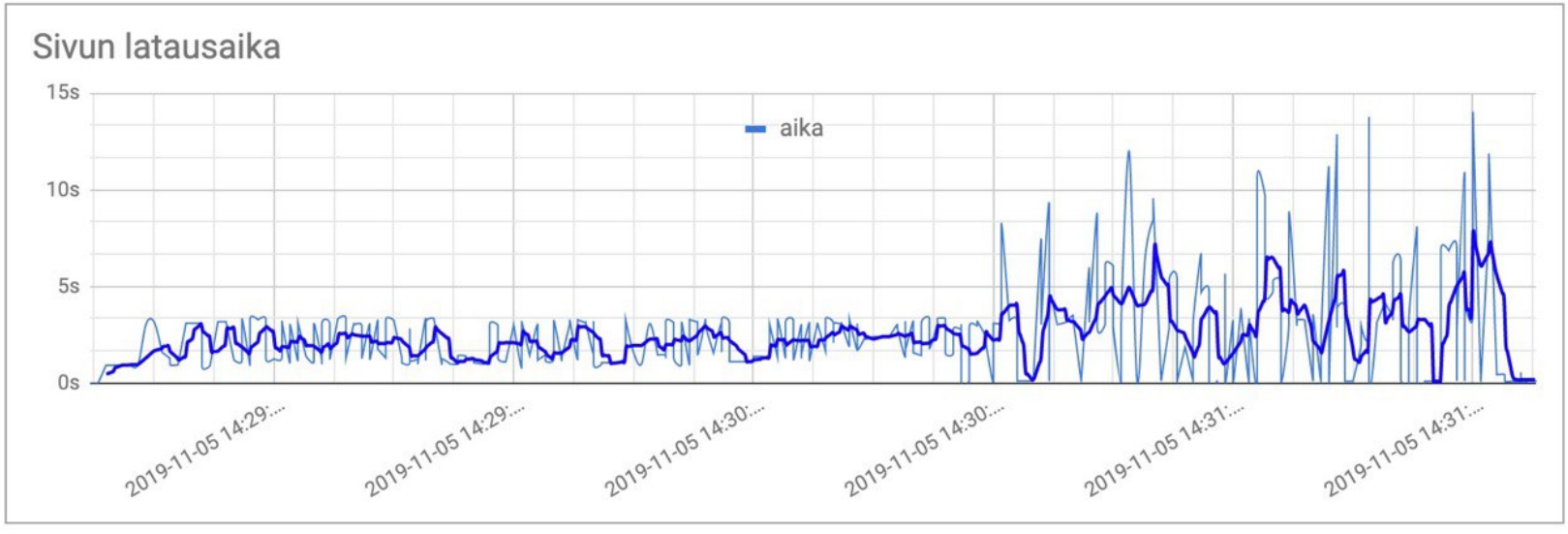
- Testaus tapahtuu ilman tietoa järjestelmän sisäisestä rakenteesta eli kyseessä *black box* -testaus
- Testataan järjestelmää saman rajapinnan kautta, jonka kautta järjestelmää käytetään

- Perustuu järjestelmän potentiaalisiin käyttöskenaarioihin
  - jos vaatimukset ilmaistu user storyina, on niistä helppo muotoilla testejä
  - jos storyillä hyväksymiskriteeriot, tilanne on vieläkin parempi
- Kutsutaan myös *toiminnallisiksi testeiksi* (functional test)

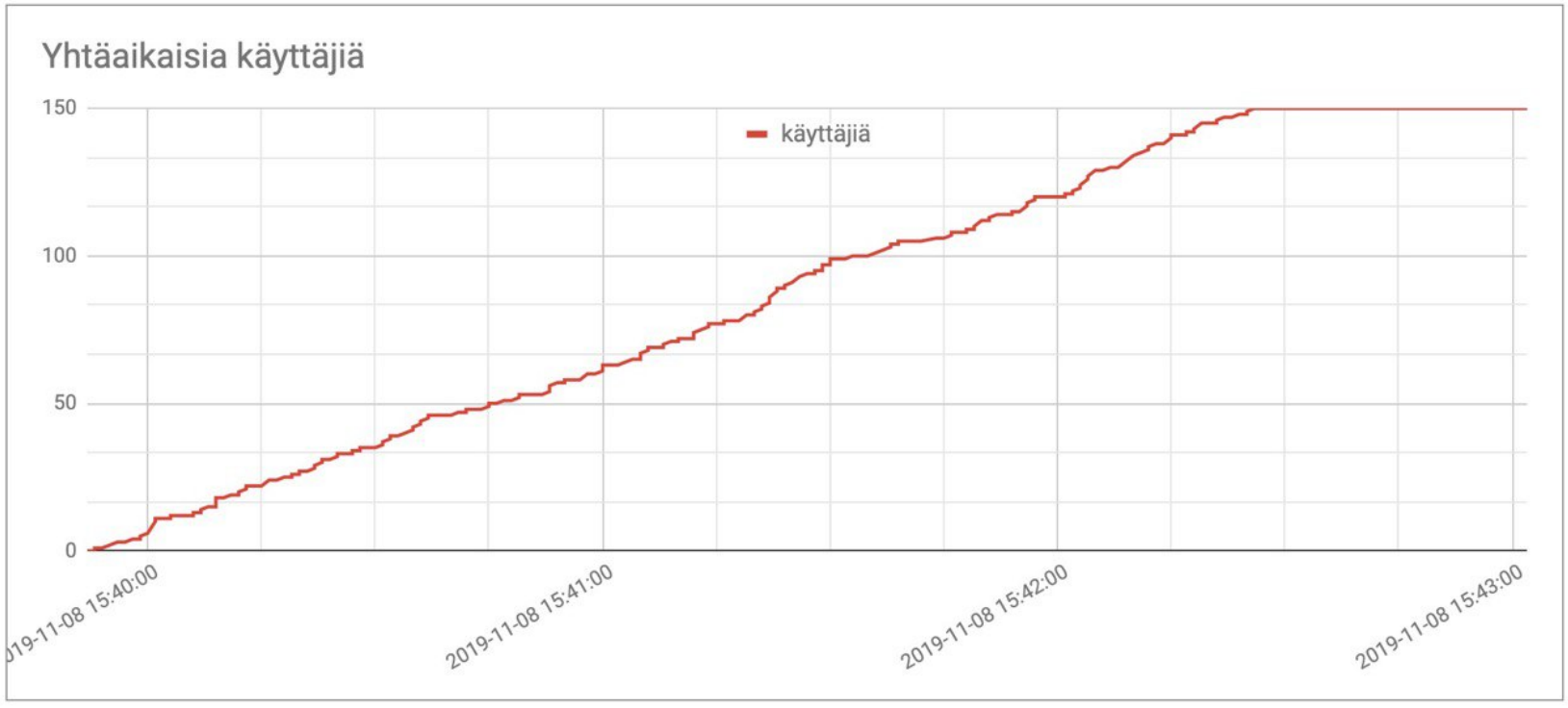
- Toiminnallisen testauksen lisäksi järjestelmätestaukseen kuuluu
  - Käytettävyystestaus
  - Suorituskykytestaus
  - Kuormitustestaus
  - Tietoturvan testaus

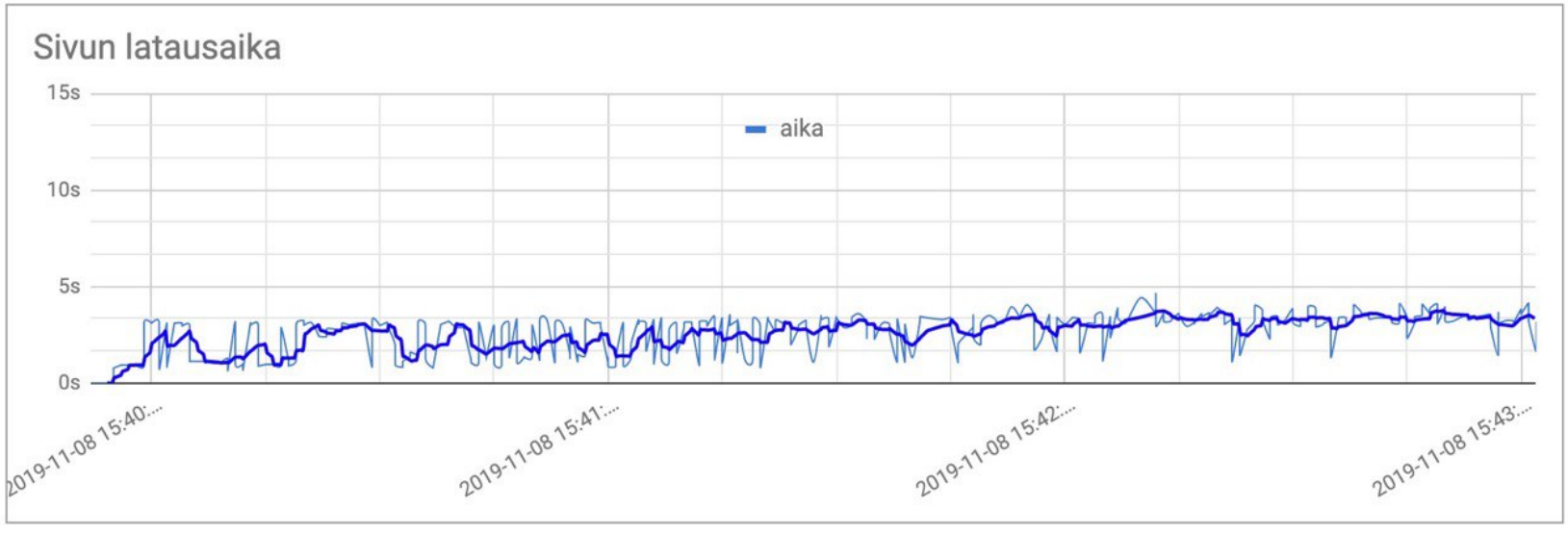
Kuormitustestaus: ennen





Kuormitustestaus: jälkeen





Testitapausten valinta

- Kattava testaaminen on mahdotonta ja testaus työlästä
- Oleellista löytää kohtuullisen kokoinen *testitapausten joukko*, jonka avulla löytyy mahdollisimman suuri määrä virheitä

Testitapausten valinta

- Kattava testaaminen on mahdotonta ja testaus työlästä
- Oleellista löytää kohtuullisen kokoinen *testitapausten joukko*, jonka avulla löytyy mahdollisimman suuri määrä virheitä
- *Testitapaus* testaa ohjelmiston toiminnallisuutta joillakin syötteillä
- Useat syötteet ohjelmiston toiminnan kannalta samanlaisia
  - nämä muodostavat *ekvivalenssiluokan*

Testitapausten valinta

- Kattava testaaminen on mahdotonta ja testaus työlästä
- Oleellista löytää kohtuullisen kokoinen *testitapausten joukko*, jonka avulla löytyy mahdollisimman suuri määrä virheitä
- *Testitapaus* testaa ohjelmiston toiminnallisuutta joillakin syötteillä
- Useat syötteet ohjelmiston toiminnan kannalta samanlaisia
  - nämä muodostavat *ekvivalenssiluokan*
- Jaetaan syötteet ekvivalenssiluokkiin ja tehdään yksi testitapaus kutakin ekvivalenssiluokkaa tai syötteiden ekvivalenssiluokkien kombinaatiota kohti
- Erityisen kiinnostavia syötearvoja ovat ekvivalenssiluokkien väliset *raja-arvot*

Testisyötteiden valinta

- Mitä testitapauksia kannattaisi valita *tekstitv:n sivun valintaikkunan* testaamiseen?
  - Tekstitv:n sivu vastaa lukua väliltä 100-899
  - Osaa välin luvuista vastaavaa sivua ei ole olemassa

Testisyötteiden valinta

- Mitä testitapauksia kannattaisi valita *tekstitv:n sivun valintaikkunan* testaamiseen?
  - Tekstitv:n sivu vastaa lukua väliltä 100-899
  - Osaa välin luvuista vastaavaa sivua ei ole olemassa
- Testisyötteen ekvivalenssiluokkia olisivat ainakin seuraavat
  - Olemassa olevaa sivua vastaavat luvut: 235
  - Validit luvut jotka eivät vastaa mitään sivua: ???
  - Liian pienet ja liian suuret luvut: 99, 900, 1000
  - Syötteet jotka sisältävät kiellettyjä merkkejä: 10X, 100X
  - Tyhjä syöte

Tekstitv:n testitapaukset

Edellinen sivu | Edellinen alasivu | Seuraava alasivu | Seuraava sivu

Teksti-TV

yle.fi/tekstitv 199 PÄÄHAKEMISTO

106 Taksisääntelyn purkaminen etenee

107 Tieto irtisanoo lähes 180

162 Reuters: Kreikan kasvu yllätti

651 MM-karsinta: Turkki nujersi Suomen

221 Janne Keränen jälleen KalPa-sankari

101	UUTISET	160	TALOUS	190	ENGLISH
201	URHEILU	350	RADIOT	470	VEIKKAUS
300	OHJELMAT	400	SÄÄ	575	TEKSTI-TV
799	SVENSKA	500	ALUEET	890	KALENTERI

Sää paikkakunnittain 406-408

Nopea piiras 811

Edellinen sivu | Edellinen alasivu | Seuraava alasivu | Seuraava sivu

Kotimaa | Ulkomaat | Talous | Urheilu | Svenska sidor | Teksti-TV Yle.fi YLE Uutiset

Sivun valinta: OK

Yksikkötestaus

- Kohteena siis yksittäiset metodit ja luokat
- Testattavan koodin rakenne otetaan huomioon testejä laatiessa, *lasilaatikkotestausta* (white box testing)
- *Developer testing*: sovelluskehittäjien vastuulla

Yksikkötestaus

- Kohteena siis yksittäiset metodit ja luokat
- Testattavan koodin rakenne otetaan huomioon testejä laatiessa, *lasilaatikkotestausta* (white box testing)
- *Developer testing*: sovelluskehittäjien vastuulla

- Päätarkoitus *sisäisen laadun* (internal quality) kontrollointi
  - onko virheiden jäljitys ja korjaaminen helppoa
  - onko koodia helppo laajentaa ja jatkokehittää
  - pystytäänkö koodin toiminnallisuuden oikeellisuus varmistamaan muutoksia tehtäessä

Yksikkötestaus

- Iteratiivisessa kehityksessä sisäisellä laadulla suuri merkitys (ohjelmistoa laajennetaan koko ajan)

Yksikkötestaus

- Iteratiivisessa kehityksessä sisäisellä laadulla suuri merkitys (ohjelmistoa laajennetaan koko ajan)
- Edesauttaa myös ulkoista eli asiakkaan kokemaa laatua
  - Yksikkötestit voivat eliminoida joitain asiakkaalle näkyviä virheitä, joita järjestelmätestauksen testitapaukset eivät löydä

Yksikkötestaus

- Iteratiivisessa kehityksessä sisäisellä laadulla suuri merkitys (ohjelmistoa laajennetaan koko ajan)
- Edesauttaa myös ulkoista eli asiakkaan kokemaa laatua
  - Yksikkötestit voivat eliminoida joitain asiakkaalle näkyviä virheitä, joita järjestelmätestauksen testitapaukset eivät löydä
- Bugit on taloudellisesti edullista paikallistaa mahdollisimman aikaisessa vaiheessa
  - yksikkötestauksessa löydetty bugi on halvempi ja nopeampi korjata kuin järjestelmä- tai integraatiotestauksessa löytyvä

Yksikkötestaus

- Iteratiivisessa kehityksessä sisäisellä laadulla suuri merkitys (ohjelmistoa laajennetaan koko ajan)
- Edesauttaa myös ulkoista eli asiakkaan kokemaa laatua
  - Yksikkötestit voivat eliminoida joitain asiakkaalle näkyviä virheitä, joita järjestelmätestauksen testitapaukset eivät löydä
- Bugit on taloudellisesti edullista paikallistaa mahdollisimman aikaisessa vaiheessa
  - yksikkötestauksessa löydetty bugi on halvempi ja nopeampi korjata kuin järjestelmä- tai integraatiotestauksessa löytyvä
- Koska yksikkötestejä joudutaan ajamaan moneen kertaan, tulee niiden suorittaminen ja testien tulosten raportointi *automatisoida*

Mitä tulisi testata yksikkötestein?

- JUnitin kehittäjän Kent Beckin vastaus:
  - “Do I have to write a test for everything?”
  - “No, just test everything that could reasonably break”

Mitä tulisi testata yksikkötestein?

- JUnitin kehittäjän Kent Beckin vastaus:
  - “Do I have to write a test for everything?”
  - “No, just test everything that could reasonably break”
- Optimitilanteessa testitapaukset
  - kaikkien metodit ja niiden kutsukombinaatiot hyväksyttävillä parametrien arvoilla
  - ja virheellisillä parametrien arvoilla

Mitä tulisi testata yksikkötestein?

- JUnitin kehittäjän Kent Beckin vastaus:
  - “Do I have to write a test for everything?”
  - “No, just test everything that could reasonably break”
- Optimitilanteessa testitapaukset
  - kaikkien metodit ja niiden kutsukombinaatiot hyväksyttävillä parametrien arvoilla
  - ja virheellisillä parametrien arvoilla
- Ekvivalenssiluokat ja niiden raja-arvot kannattaa huomioida
- Koska yksikkötestejä tehtäessä ohjelmakoodi on nähtävillä, testien parametrien ekvivalenssiluokat ja raja-arvot pääteltävissä koodista

Ohtuvarasto, ekvivalenssiluokat: tyhjä, puolitäysi, täysi

```
public class Varasto {
  private double tilavuus;
  private double saldo;

  public double otaVarastosta(double maara) {
    if (maara < 0) return 0.0;

    if(maara > saldo) {
      double kaikkiMitaVoidaan = saldo;
      saldo = 0.0;
      return kaikkiMitaVoidaan;
    }

    saldo = saldo - maara;
    return maara;
  }
}
```

Testauskattavuus

- Yksikkötestien (ja toki myös muunkinlaisten testien) hyvyyttä voidaan mitata *testauskattavuuden* (test coverage) käsitteellä

Testauskattavuus

- Yksikkötestien (ja toki myös muunkinlaisten testien) hyvyyttä voidaan mitata *testauskattavuuden* (test coverage) käsitteellä
- Muutamaa eri tyyppiä
  - rivikattavuus
  - haarautumakattavuus
  - ehtokattavuus
  - polkukattavuus

Testauskattavuus

- Yksikkötestien (ja toki myös muunkinlaisten testien) hyvyyttä voidaan mitata *testauskattavuuden* (test coverage) käsitteellä
- Muutamaa eri tyyppiä
  - rivikattavuus
  - haarautumakattavuus
  - ehtokattavuus
  - polkukattavuus
- Rivi- ja haarautumakattavuudelle hyvä työkalutuki, esim JaCoCo

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
Varasto(double, double)		0%		0%	4	4	10	10	1	1
toString()		0%		n/a	1	1	1	1	1	1
otaVarastosta(double)		62%		50%	2	3	4	8	0	1
lisaaVarastoon(double)		77%		50%	2	3	2	6	0	1
Varasto(double)		82%		50%	1	2	1	6	0	1
paljonkoMahtuu()		100%		n/a	0	1	0	1	0	1
getSaldo()		100%		n/a	0	1	0	1	0	1
getTilavuus()		100%		n/a	0	1	0	1	0	1
Total	66 of 126	47%	11 of 16	31%	10	16	18	34	2	8

- Epäkattavasti testattu haarautumiskohta esim. if ilmaistaan keltaisella

```
51.     public void lisaaVarastoon(double maara) {
52.         if (maara < 0) // virhetilanteessa voidaan tehdä
53.         {
54.             return;         // tällainen pikapoistuminenkin!
55.         }
56.         if (maara <= paljonkoMahtuu()) // omia aksessoreita voi kutsua
57.         {
58.             sal                 ;          // ihan suoraan sellaisinaan
59.         } else {
60.             saldo = tilavuus;  // täyteen ja ylimäärä hukkaan!
61.         }
62.     }
```

1 of 2 branches missed.

Mutaatiotestaus

- Pelkkä kattavuus ei kerro paljoa testien laadusta
- Hyvien testien pitäisi huomata (ts. mennä rikki) jos ohjelmaan tulee bugi

Mutaatiotestaus

- Pelkkä kattavuus ei kerro paljoa testien laadusta
- Hyvien testien pitäisi huomata (ts. mennä rikki) jos ohjelmaan tulee bugi
- Mutaatiotestaus (engl. mutation testing) tutkii testien hyvyyttä generoimalla koodiin *mutantteja* eli pieniä "bugeja" ja tarkastamalla hajoavatko testit
- Erilaisia mutanttityyppejä:

```
if ( x<0 )                  if (x <= 0) tai if (true)
x += 1                      x-= 1
return x                    return true
olio = new Olio()           olio = null
```

Mutaatiotestaus

- Pelkkä kattavuus ei kerro paljoa testien laadusta
- Hyvien testien pitäisi huomata (ts. mennä rikki) jos ohjelmaan tulee bugi
- Mutaatiotestaus (engl. mutation testing) tutkii testien hyvyyttä generoimalla koodiin *mutantteja* eli pieniä "bugeja" ja tarkastamalla hajoavatko testit
- Erilaisia mutanttityyppejä:

```
if ( x<0 )                if (x <= 0) tai if (true)
x += 1                    x-= 1
return x                  return true
olio = new Olio()         olio = null
```

- Ongelmana on mutaatioiden suuri määrä ja ns. ekvivalentit mutantit
  - tulos vaatii aina ihmisen tulkintaa

Ohjelmiston integraatio

- Järjestelmän yksittäiset, erillään yksikkötestatut luokat tulee integroida toimivaksi kokonaisuudeksi
- Integroinnin yhteydessä tai sen jälkeen suoritetaan integrointitestaus
  - Toimivatko komponentit yhdessä?

Ohjelmiston integraatio

- Järjestelmän yksittäiset, erillään yksikkötestatut luokat tulee integroida toimivaksi kokonaisuudeksi
- Integroinnin yhteydessä tai sen jälkeen suoritetaan integrointitestaus
  - Toimivatko komponentit yhdessä?
- Kaksi lähestymistapaa: rakenteisiin perustuva ja toiminnallisuuksiin perustuva

Ohjelmiston integraatio

- *Rakenteeseen perustuvassa* integraatiossa keskitytään kerrallaan sovelluksen yksittäisten arkkitehtuurillisten komponenttien integrointiin
  - Verkkokaupassa integroitaisiin sovelluslogiikan luokat, käyttöliittymän luokat ja tietokantarajapinta ensin omina kokonaisuuksinaan

Ohjelmiston integraatio

- *Rakenteeseen perustuvassa* integraatiossa keskitytään kerrallaan sovelluksen yksittäisten arkkitehtuurillisten komponenttien integrointiin
  - Verkkokaupassa integroitaisiin sovelluslogiikan luokat, käyttöliittymän luokat ja tietokantarajapinta ensin omina kokonaisuuksinaan
- *Ominaisuuksiin perustuvassa* integroinnissa liitetään yhteen alikomponentit, jotka toteuttavat järjestelmän loogisen toimintakokonaisuuden
  - Esim. ntegroidaan kerrallaan kaikki storyn *lisää tuote ostoskoriin* toteutukseen liittyvä koodi
  - Tämän jälkeen integroidaan seuraavan storyn koodi

Ohjelmiston integraatio

- Vesiputuoksen toimintatapa
  - Yksittäiset komponentit ohjelmoidaan ja yhsikkötestataan erikseen
  - Tämän jälkeen ne integroidaan, yleensä rakenteeseen perustuen, kerralla yhteen

Ohjelmiston integraatio

- Vesiputuoksen toimintatapa
  - Yksittäiset komponentit ohjelmoidaan ja yhsikkötestataan erikseen
  - Tämän jälkeen ne integroidaan, yleensä rakenteeseen perustuen, kerralla yhteen
- Tämän tyylinen *big bang* -integraatio on osoittautunut todella riskialttiiksi
- Seurauksena usein ns. integraatiohelvetti

Ohjelmiston integraatio

- Vesiputuoksen toimintatapa
  - Yksittäiset komponentit ohjelmoidaan ja yhsikkötestataan erikseen
  - Tämän jälkeen ne integroidaan, yleensä rakenteeseen perustuen, kerralla yhteen
- Tämän tyylinen *big bang* -integraatio on osoittautunut todella riskialttiiksi
- Seurauksena usein ns. integraatiohelvetti
- Moderni ohjelmistotuotanto suosii ns. *jatkuvaa integraatiota*
  - Hyvin tiheässä tahdissa tapahtuvaa ominaisuuksiin perustuvaa integrointia

Regressiotestaus

- Iteratiivisessa ohjelmistotuotannossa, jokainen iteraatio tuottaa ohjelmistoon uusia ominaisuuksia
- Tulee huolehtia, että ei rikota jo toimivia osia

Regressiotestaus

- Iteratiivisessa ohjelmistotuotannossa, jokainen iteraatio tuottaa ohjelmistoon uusia ominaisuuksia
- Tulee huolehtia, että ei rikota jo toimivia osia
- Testit siis on suoritettava uudelleen aina kun ohjelmistoon tehdään muutoksia
- Tätä käytäntöä sanotaan *regressiotestaukseksi*

Regressiotestaus

- Iteratiivisessa ohjelmistotuotannossa, jokainen iteraatio tuottaa ohjelmistoon uusia ominaisuuksia
- Tulee huolehtia, että ei rikota jo toimivia osia
- Testit siis on suoritettava uudelleen aina kun ohjelmistoon tehdään muutoksia
- Tätä käytäntöä sanotaan *regressiotestaukseksi*
- Regressiotestijoukko koostuu kaikista ohjelmistolle tehdyistä testeistä
  - sisältää yksikkö-, integraatio- ja järjestelmätesteistä

Regressiotestaus

- Iteratiivisessa ohjelmistotuotannossa, jokainen iteraatio tuottaa ohjelmistoon uusia ominaisuuksia
- Tulee huolehtia, että ei rikota jo toimivia osia
- Testit siis on suoritettava uudelleen aina kun ohjelmistoon tehdään muutoksia
- Tätä käytäntöä sanotaan *regressiotestaukseksi*
- Regressiotestijoukko koostuu kaikista ohjelmistolle tehdyistä testeistä
  - sisältää yksikkö-, integraatio- ja järjestelmätesteistä
- Testaus on työlästä ja regressiotestauksen tarve tekee siitä entistä työläämpää
- *Testaus kannattaa automatisoida* mahdollisimman suurissa määrin