Testaus ja laadunvarmistus

- Laadunvarmistuksen keinoja
- Testauskäsitteistöä
  - vika, virhe, häiriö; testitapaus; testausstrategia; testaustyypit: black & white boxes
- Menetelmiä testitapausten määräämiseen
- Vuorovaikutteisen järjestelmän testaus
- Olioiden testaus
- Testaussuunnitelma
- Katselmus ja tarkastus (review, inspection)

Laadunvarmistustekniikat

- testaus (testing)
  - ohjelman suorittamista tapauksissa joilla todennäköisimmin löydetään ohjelmistossa piileviä virheitä
- tarkastukset (inspections, reviews)
  - asiantuntijoiden suorittamia dokumentin (voi olla myös koodi) läpikäyntejä tarkistuskohdin joilla todennäköisimmin löydetään virheitä

Testas - havaintoja

- ammattiohjelmoija tuottaa ~1,2 virhettä / 200 riviä ohjelmakoodia
- uudessa 200000 rivin ohjelmatuotteessa on ~1200 virhettä
- pitkäänkin käytössä olleissa ohjelmistoissa on noin 1 virhe / 1000 riviä koodia
- yhden virheen poistamiseen kuluu noin 12 henklötyötuntia

Testaus - virhejakaumia

- Beizer: Software testing techniques, 2 ed, van Nostrand Reinhold, 1990
  - ohjelmakoodin lauseissa 25%
  - ohjelmakoodin tietorakenteissa 22%
  - spesifikaation toteutustavassa 16%
  - moduulien/komponenttien integroinnissa 9%
  - huolimattomuusvirheitä 9% ...
  - spesifikaatioissa 8%

Testaus

- Hyvä testitapaus on sellainen, joka suurella todennäköisyydellä paljastaa aiemmin havaitsemattoman virheen
  - virheen löytyessää testaaja iloitsee, ohjelmoija puree hammasta … destruktiivinen luonne ?
- Testaus on näytteitä -- ei täydellistä
  - ei voi todistaa ohjelmaa oikeaksi
  - tukee kuitenkin hyviä uskomuksia ohjelmiston yleisestä käyttäytymisestä
    MIKÄLI TESTAUS NÄYTTÄÄ USKOTTAVALTA

Käsitteitä

- virhe (error, mistake, bug):
  - ohjelmiston poikkeaminen sen spesifikaatista (=määrittelystä)
- vika (fault)
  - virheellisen ohjelmakohdan suoritus - vika esiintyy siis vain jos virheellinen ohjelmakohta suoritetaan
- häiriö (failure)
  - viasta aiheutuva ulkoisesti havaittava poikkeama ohjelmiston spesifikaatiosta
  - kaikki virheet ja viat eivät välttämättä johda häiriöön

Testaus - käsitteitä

- Testitapaus:
  - tietty ohjelmiston syötearvojen joukko
- Mieluummin:
  - syötearvot + odotetut tulostiedot

Testaus - käsitteitä

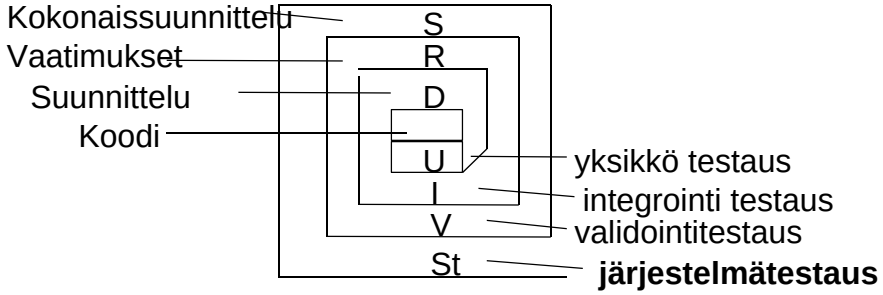
- Rakenteellinen eli sisäinen eli lasilaatikkotestaus (structural / glass box / white box t)
  - **testitapaukset valitaan siten, että ne aiheuttavat ohjelmakoodin mahdollisten suoritusreittien läpikäyntiä**
  - ohjelmasilmukoista johtuen kaikkien mahdollisten suoritusreittien läpikäynti ei ole yleensä mahdollista järkevässä ajassa

Testaus - käsitteitä

- Toiminnallinen eli funktionaalinen eli ulkoinen eli mustalaatikkotestaus (functional / black box testing)
  - perustuu ohjelmiston ulkoisesti havaittavan toiminnan tarkasteluun
  - syötteet -> komponentti -> tulokset ok?
  - Suuresta syöteavaruudesta johtuen (esim. R: real) kaikkien mahdollisten syötteiden testaus ei ole mahdollista järkevässä ajassa

Käsitteitä: testausstrategia

- Spiraalikuvaus ohjelmistotuotantoprosessista



tuotanto sisäänpäin - testaus ulospäin

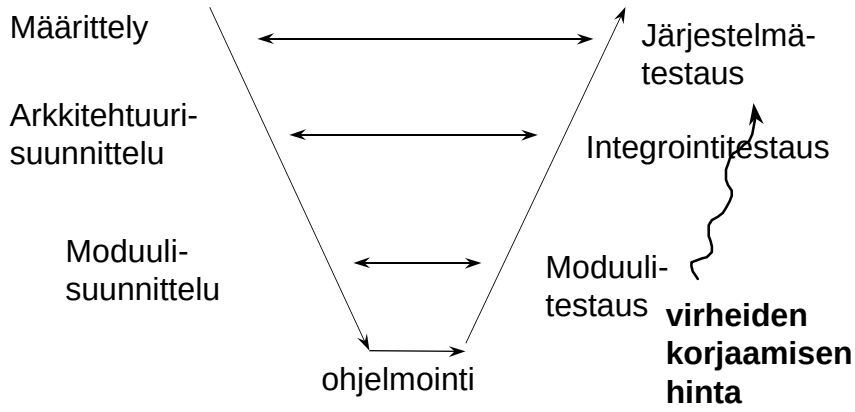
Käsitteitä: testausstrategia...

- yksikkötestaus (unit test):
  - testataan ohjelmiston jokainen komponentti
    - (moduuli, aliohjelma, luokka, …)
  - tyypillisesti white box

Käsitteitä: testausstrategia ...

- integrointitestaus (integration test)
  - testataan komponenttien yhteistoiminta
- validointitestaus (validation test)
  - testataan ohjelmiston ja vaatimusmäärittelyn vastaavuus
- järjestelmätestaus (system test)
  - testataan järjestelmä kokonaisuudessaan (mukana laitteistot)
- nämä tyypillisesti black box

Testausstrategia V-malli

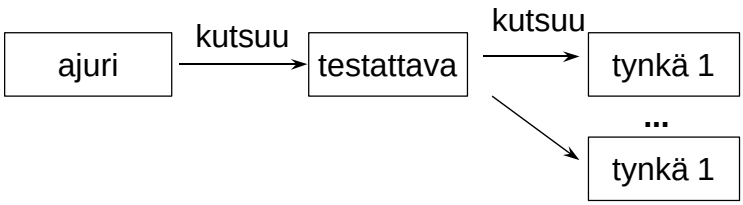


Testausstrategia V-malli

- V-malli kytkee rakentamisvaiheen ja testausvaiheen
- Testaussuunnitelma laaditaan testausvaihetta vastaavassa rakentamisvaiheessa
- Kussakin testausvaiheessa ohjelmiston toimintaa verrataan vastaavan rakentamisvaiheen tuottamaan spesifikaatioon

Käsitteitä: testaustyypit

- yksikkötestaus
  - toimiiko moduuli irrallisena kokonaisuutena?
  - toimivatko paikalliset tietorakenteet ja sisäiset suoritusreitit
  - voidaan tarvita erityisiä ‘tynkiä’ (stub) ja ‘ajureita’ (driver) korvaamaan testattavan kutsumia ja sitä kutsuvia moduuleja



Testaustyypit

- Integrointitestaus
  - toimiiko moduulijoukko integroituna kokonaisuutena (kun yksittäiset moduulit on ensin testattu erikseen)
  - toimiiko tiedonvälitys moduulien rajapinnoilla?
  - toimiiko moduulien kytkentä?

- Tapa integroida vaikuttaa testaukseen
  - big bang - kaikki valmiiksi, sitten kasaan ja testataan
  - poluittain (threads)- suorituspolku kerrallaan
  - top down (tarvitaan tynkiä), bottom up (tarvitaan ajureita)

Testaustyypit

- Validointitestaus / järjestelmätestaus (usein yhdistetty)
  - toimiiko kokonaisjärjestelmä niin kuin tuotantoprosessin alussa on määritelty?
  - millainen on ulkoisesti havaittava toiminta, suorituskyky, varmuus, siirrettävyys, virheensietokyky, ylläpidettävyys,..?
  - toimivatko ohjelmisto ja laitteisto yhdessä?

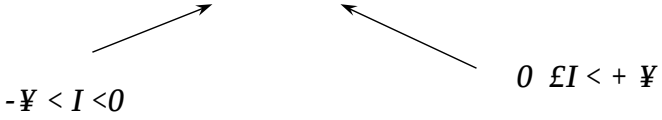
- testilajeja:
  - **alfa-testaus**
    - asiakkaan suorittama tuottajan tiloissa ja laitteilla
  - **beta-testaus**
    - asiakkaan suorittama omassa todellisessa käyttöympäristössään

Testausmenetelmiä

- Toiminnallinen testaus - black box
  - arvoalueisiin perustuva
  - arvoalueiden ekvivalenssiluokkiin p.
- Rakenteellinen testaus - white box
  - polkutestaus
    - pyrkimyksiä: polku-, lause-, päätös- jne. kattavuus
  - tietovuotestaus
    - tiedon tila: määritelty - käytetty - jne

Toiminnallinen testaus (black box)

- perustuu useimmiten syötemuuttujien arvoalueen (domain) osittamiseen => arvoaluetestaus (domain testing)
- parametrin tyyppi kertoo arvoalueen
  var I: integer; I kokonaisluku (välillä -¥ < I < + ¥)
- ohjelma synnyttää arvoalueita
  esim: *if I<0 then … else ….*



Toiminnallinen testaus (black box)

- arvoaluetestauksen periaate:
  - valitaan yksi testitapaus arvoalueen sisältä
  - valitaan arvoalueen rajat
  - valitaan yksi testitapaus välittömästi arvoalueen rajojen ulkopuoleta

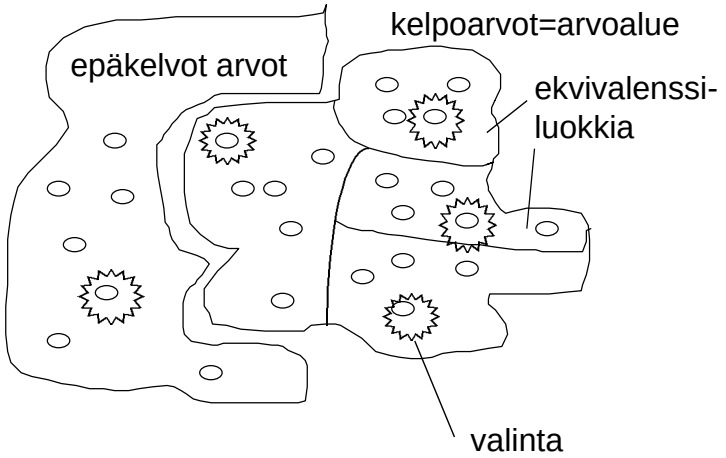
- esim: spesifikaatiossa arvoalue: $0 \leq I \leq 100$, testiaineisto:
  - I=50 (sisällä, voisi olla yhtä hyvin vaikka 78)
  - I=0 (alarajalla)
  - I=100 (ylärajalla)
  - I= -1 (alarajan ulkopuolella)
  - I= 101 (ylärajan ulkopuolella)

Toiminnallinen testaus (black box)

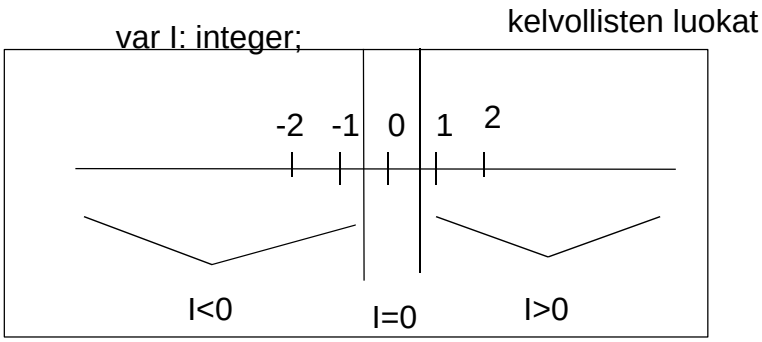
- Arvoalueiden ekvivalenssiluokat
  - T.J.Ostrand, M.J.Balcer: The Category-Partition Method for Specifying and Generating Functional Tests, CACM, 31, 6, 1988, pp 676-686

- jokainen tiettyyn ekvivalenssiluokkaan kuuluva arvo käyttäytyy testauksen kannalta ‘samalla tavalla’
  - jokainen luokan arvo paljastaa virheen yhtä suurella todennäköisyydellä
  - jos toiminta on virheellistä arvolla α, se on virheellistä muillakin ko. luokan arvoilla
  - jos toiminta on oikein arvolla α, se on oikein myös muilla ko. luokan arvoilla
- testitapauksiksi kustakin luokasta 1 (tai muutama) arvo

Toiminnallinen testaus (black box)



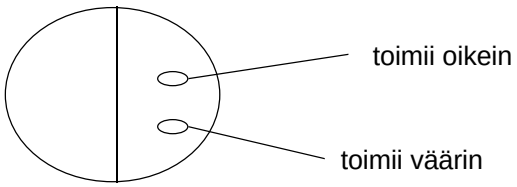
Toiminnallinen testaus (black box)



kelvottomat	3.14
	abc

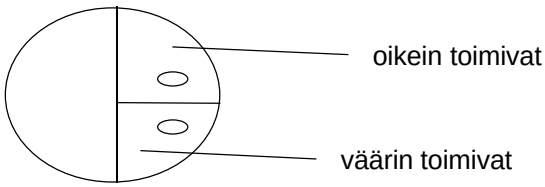
Toiminnallinen testaus (black box)

- testauksen aikana ekvivalenssihypoteesi saattaa osoittautua pätemättömäksi



Toiminnallinen testaus (black box)

- Luokkarajoja on tällöin muutettava



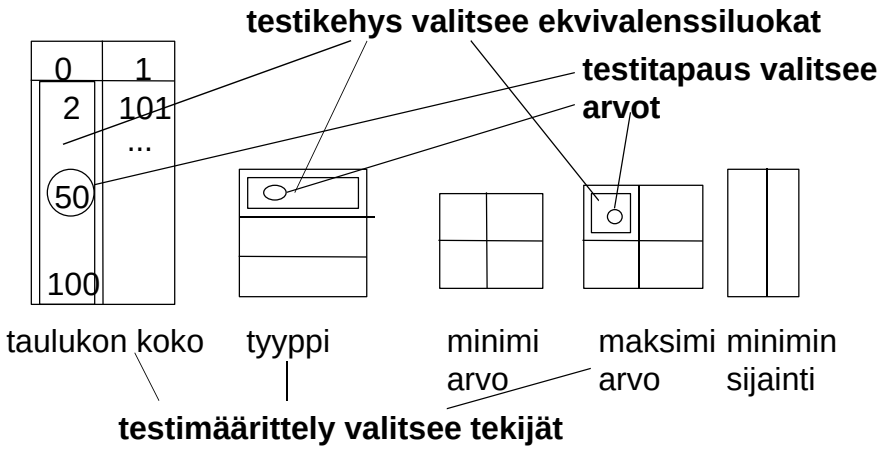
Category-Partition tekniikan vaiheet:

- Jaa ohjelmisto spesifikaation perusteella erikseen testattaviksi osajärjestlmiksi
- Tunnista jokaisen osajärjestelmän syötemuuttujat
  - testitapaus on syötemuuttujien alustusarvojen joukko

- Määrittele jokaiselle syötemuuttujalle sen tekijät
  - tekijä, kategoria (category) on muuttujan keskeinen ominaisuus
  - esim. järjestettävä taulukko
    - tekijöitä: taulukon koko, alkioiden tyyppi, alkioiden minimiarvo, alkioiden maksimiarvo, minimiarvon sijainti, maksimiarvon sijainti - nämä ovat oleellisia järjestämisen toimimisen kannalta

- Määrää jokaiselle tekijälle arvojen ekvivalenssiluokat (choice)
  - esim. järjestettävä taulukko / taulukon koko:
    - {koko=0} {koko=1} {2<=koko<=100} {koko>100}

- Tee testimäärittely jokaiselle osajärj:lle
  - **tekijälista**
  - **kunkin tekijän ekvivalenssiluokat**
  - näiden perusteella voidaan määrätä **testikehykset** (test frame)
    - testikehys muodostetaan valitsemalla joukko tekijöitä
    - tarkoituksena on muodostaa testitapaus valitsemalla ensin yksi ekvivalenssiluokka kuhunkin testikehyksen tekijään liittyen ja sitten yksi arvo kustakin valitusta luokasta



- Lisää testimäärittelyihin rajoitukset, jotka koskevat tekijöiden ja ekvivalenssiluokkien mukaanottamista testikehyksiin
  - esim, jotkin ekvivalenssiluokat eivät voi olla samaan aikaan voimassa, kuten {taulukon koko=0} ja {minimin sijainti>100}
  - tällä vähennetään testikehysten ja -tapausten lukumäärää

- Tuota testikehykset automaattisesti testaustyökalulla
- Tuota testikehyksiin liittyvät testitapaukset (työkalulla)
- Yhdistä loogisesti yhteenkuuluvat testitapaukset testijonoiksi (test script)

- Testaa osajärjestelmät suorittamalla testijonot
- Ylläpidä testimäärittelyjä ja -tuloksia testitietokannassa
  - jos saman ekvivalenssiluokan tapaukset aiheuttavat ristiriitaista toimintaa, muuta luokkajakoa.

Ekvivalenssiluokista

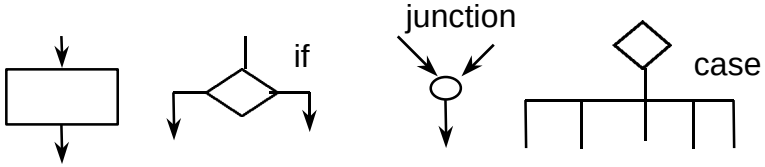
- Luokittelu pyrittävä tekemään niiden ominaisuuksien mukaan, jotka todennäköisimmin aiheuttavat virheitä
- Testitapauksia valittava luokan koon mukaan:
  - iso luokka, enemmän tapauksia
- Huono luokittelu paljastaa virheitä yhtä tehottomasti kuin puhtaasti satunnainen testaus, missä koko arvoalue tutkitaan yhdeksi ekvivalenssiluokaksi ja testitapaukset valitaan satunnaisesti

Testausmenetelmiä

- Toiminnallinen testaus - black box
  - arvoalueisiin perustuva
  - arvoalueiden ekvivalenssiluokkiin p.
- Rakenteellinen testaus - white box
  - polkutestaus
    - pyrkimyksiä: polku-, lause-, päätös- jne. kattavuus
  - tietovuotestaus
    - tiedon tila: määritelty - käytetty - jne

Rakenteellinen testaus (white box)

- perustuu ohjelman rakenteen hyväksikäyttöön - tieto ja kontrollivuoesityksiin
  - tietovuo (data flow) - tiedon kulku
  - kontrollivuo (control flow) - kontrollin kulku
- Kontrollin kulkua voidaan kuvata perinteisellä kulkukaaviolla (flowchart)



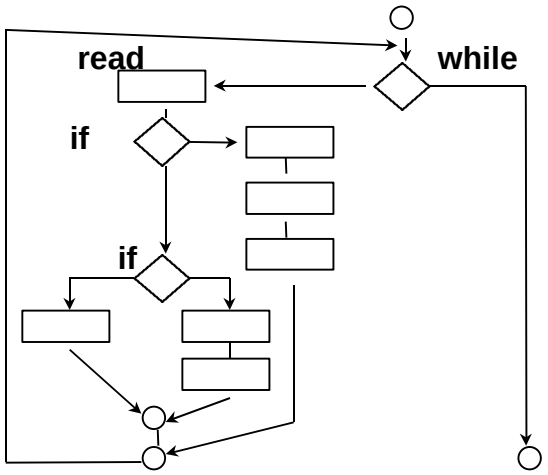
Kontrollin kulku

- vuokaavio
  - 1-1 vastaavuus ohjelmakoodin kanssa
  - jokainen lause näkyy kaaviossa
  - liian yksityiskohtainen testauksen kannalta
- vuoverkko (flowgraph)
  - abstraktio vuokaaviosta
  - kontrollin haarautumis- ja yhtymiskohdat esitetään verkon solmuina (node)
  - peräkkäiset suoritettavat lauseet yhdistyvät yhdeksi prosessiksi

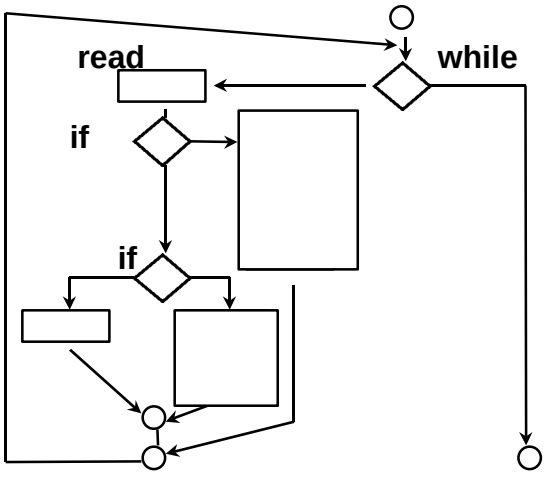
ohjelmasta vuoverkoksi

```
procedure sort begin
   while records_remain begin
        read record;
        if record.field1 = 0 then begin
                process record;
                store in buffer;
                Increment counter;
        end  else begin
                if record.field2=0 then begin
                        reset counter;
                end else begin
                        process record;
                        store in file;
                end;
        end;
   end while;
end;
```

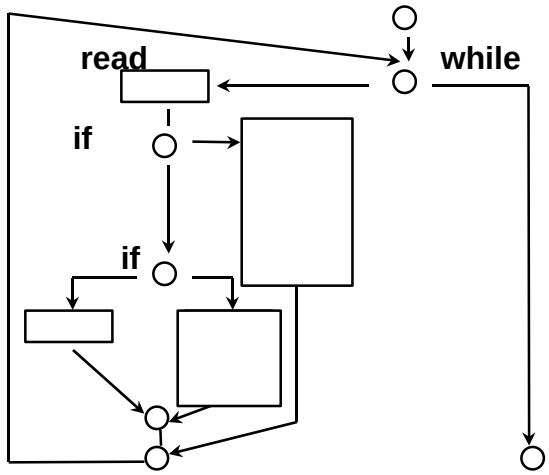
ohjelmaa vastaava kulkukaavio



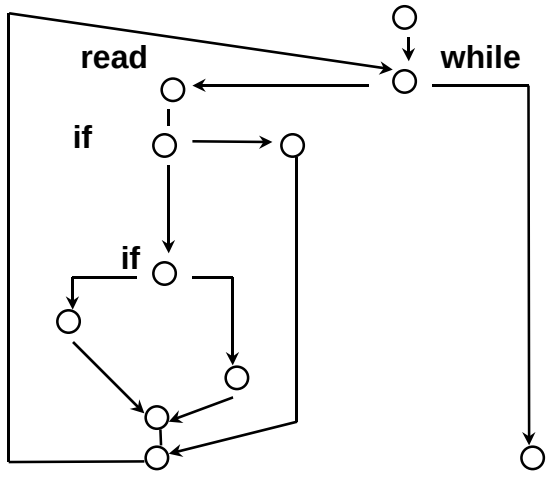
kohti vuoverkkoa
- lauseet prosesseiksi



kohti vuoverkkoa - haarautumasolmut

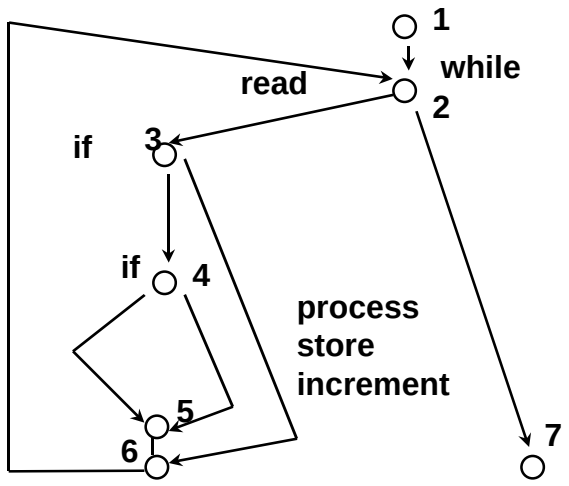


kohti vuoverkkoa - prosessisolmut



jotkut esitykset esim. Pressman jättävät vuoverkon tällaiseksi

kohti abstraktia vuoverkkoa - prosessit särmiin

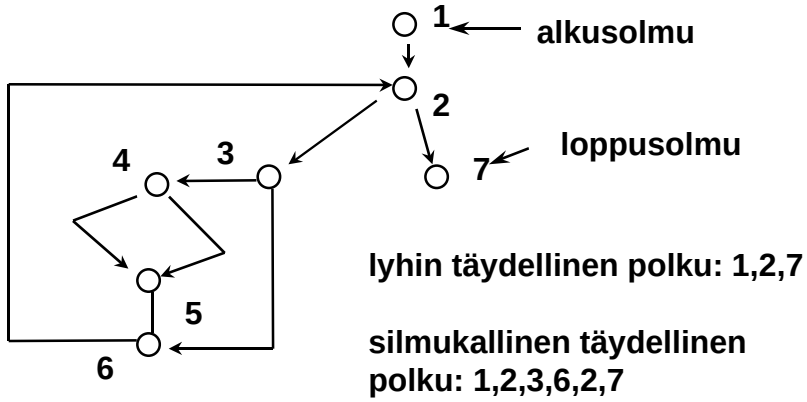


vuoverkon käsitteitä

- vuoverkon polku (path), reitti:
  - jono linkkien yhdistämiä solmuja
  - sama solmu voi esiintyä polulla useaan kertaan (silmukat)
- segmentti:
  - polun osajono
- segmentin pituus:
  - segmentin solmujen lukumäärä

- täydellinen polku (complete path):
  - polku, joka alkaa vuoverkon alkusolmusta ja päättyy sen loppusolmuun
  - alkusolmu: ei sisääntulevia linkkejä (esimerkissä 1)
  - loppusolmu: ei ulosmeneviä linkkejä (esimerkissä 7)

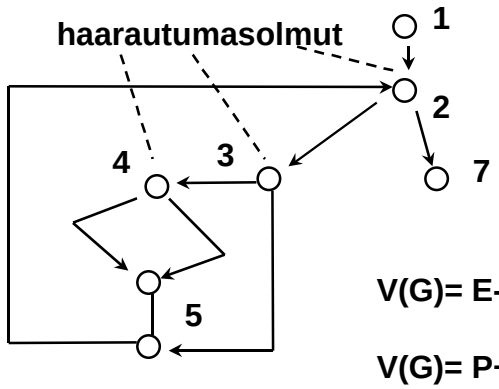
vuoverkon käsitteitä



vuoverkon käsitteitä

- McCaben kompleksisuusmitta (cyclomatic complexity) merk. V(G) verkolle G
  - ohjelman vuoverkon alueiden määrä
  - V(G)= E-N+2, missä E=särmien määrä ja N=solmujen määrä
  - V(G)= P+1, missä P on verkon haarautumasolmujen lukumäärä
  - käytetään mittaamaan ohjelman monimutkaisuutta
  - käytetään laskettaessa tarvittavien testien lukumäärää

vuoverkon käsitteitä - McCaben komplesisuusmitta



V(G)= E-N+2= 9-7+2 =4

V(G)= P+1 = 3+1 =4

Polkutestaus

- Polkutestauksessa on tarkoitus valita testiaineistot siten, että ne aiheuttavat tiettyjen vuoverkon polkujen suorituksen
- On määritelty erilaisia kattavuusmittoja polkutestaukselle:
  - **polkukattavuus (path coverage)**
    - käydään läpi kaikki mahdolliset täydelliset polut (polku/ testiajo)
    - usein mahdotonta koska polkuja on liian paljon

Polkutestaus

  - **lausekattavuus (statement coverage)**
    - käydään jokaisessa ohjelman lauseessa, eli jokaisessa vuoverkon solmussa ja prosessisärmässä, vähintään kerran testiajojen yhteydessä

- **haara- eli päätöskattavuus (branch/decision coverage)**
  - käydään ohjelman jokaisen ehdon kaikki arvot (kaikki vuoverkon särmät) läpi vähintään kerran testiajojen aikana
  - McCaben kompleksisuusmitta antaa tällöin tarvittavien testiajojen vähimmäismäärän
  - yleisesti tavoitelty kattavuus

Polkutestaus

- **ehtokattavuus (condition coverage)**
  - ohjelman jokaisen päätöksen kaikkien osaehtojen on saatava kaikki arvonsa testiajojen aikana
- **moniehtokattavuus (multiple condition coverage)**
  - testaus on suoritettava ohjelman jokaisen paatöksen osaehtojen kaikilla arvoyhdistelmillä

– **silmukkatestaus (loop testing)**
  - suoritetaan ohjelman toistolauseita (vuoverkon silmukallisia polkuja) useampaan kertaan saman testiajon aikana (koska virheet kasautuvat silmukoihin)
  - 0 kertaa, 1 kerta, tyypillinen määrä, maksimimäärä, mäksimi+1 kertaa
  - tarkentaa haarakattavaa testausta

Polkutestaus

- täydellistä kattavuutta voi olla mahdoton saavuttaa

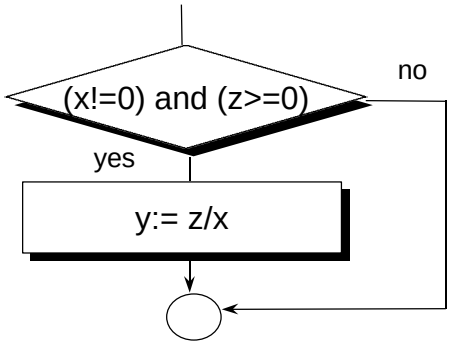
```
if a<100 then begin
  b:=a;
  ----
  ----      ei sijoituksia b:lle
  ----
  if b>100 then begin -------  end;
end;
```

saavuttamaton kohta

Polkutestaus

- Esimerkki:

  if (x!=0) and (z>=0) then y:= z/x; endif;



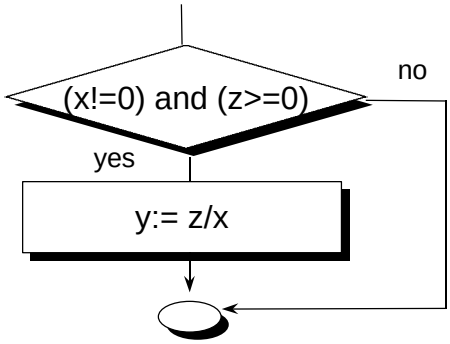
Lausekattavuus:
x=1, z=0: **yes**
1 testiajo

Päätöskattavuus:
x=1, z=0: **yes**
x=1, z=-1: **no**
2 testiajoa

Polkutestaus

- Esimerkki:

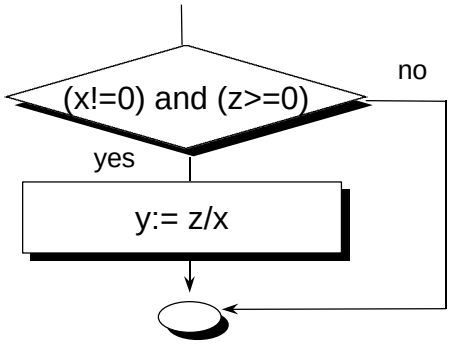
  if (x!=0) and (z>=0) then y:= z/x; endif;



Ehtokattavuus:
x=0: no, z=0: yes: **no**
x=1: yes, z=-1: no**: no**
2 testiajoa
mutta yes-haara jää
testaamatta

Polkutestaus

- Esimerkki:
  if (x!=0) and (z>=0) then y:= z/x; endif;



Moniehtokattavuus:
x=0: no, z=0: yes: **no**
x=0: no, z=-1: no**: no**
x=1: yes, z=0:yes**: yes**
x=1: yes, z=-1**:** no**: no**
4 testiajoa

Silmukoiden testaamisesta

- Yksinkertainen looppi: ohitus, 1 kierros, 2 kierrosta, tyypillinen lkm, maksimimäärä, maksimi+-1
- Sisäkkäisien silmukoiden testaus
  - aloita sisimmästä yksink. silmukan tapaan, kiinnitä ylemmät minimiarvoihinsa
  - siirtyessäsi keskimmäisiin, pidä sisemmät silmukat tyyppiarvoissa

Silmukoiden testaamisesta

- Peräkkäiset silmukat
  - riippuvuus huonoa suunnittelua
    - riippuvuus: esim kierroslaskuri “peritään” edelliseltä silmukalta
  - jos riippuvuus pakollinen, käytä sisäkkäisten silmukoiden testaustekniikkaa
  - riippumattomat silmukat -> yksinkertaisten silmukoiden testityyli

Tietovuotestaus

- Tarkastellaan tietorakenteiden tilamuutoksia ohjelman suorituksen aikana
  - tarkentaa polkutestausta
  - tilamuutokset
    - arvon asettaminen (definition)
    - arvon hyväksikäyttö (use)
- perustuu ohjelmaa vastaavan tietovuoverkon (data flowgraph) systemaattiseen läpikäyntiin

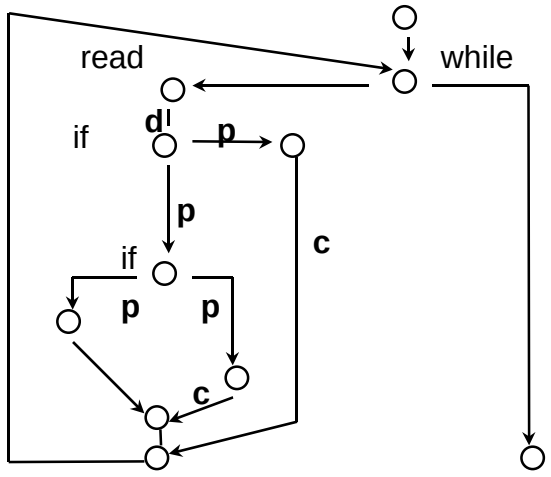
- tietovuoverkko on vuoverkko, jota on täydennetty tietyn tietorakenteen tilamuutosten kuvauksilla

Tietovuotestaus

- tietorakenteet tilat
  - **d**: tietorakenne saa uuden arvon
  - **u**: tietorakennetta käytetään
    - **c**: laskennassa
    - **p**: päätöksenteossa (haarautumisehdossa)
  - **k**: tietorakenteen tuhoaminen, rakenne tulee määrittelemättömäksi (killed)

```
procedure sort begin
    while records_remain begin
        read record;
        if record.field1 = 0 then begin
            process record;
            store in buffer;
            Increment counter;
        end  else begin
            if record.field2=0 then begin
                reset counter;
            end else begin
                process record;
                store in file;
            end;
        end;
    end while;
end;
```

Tietovuotestaus

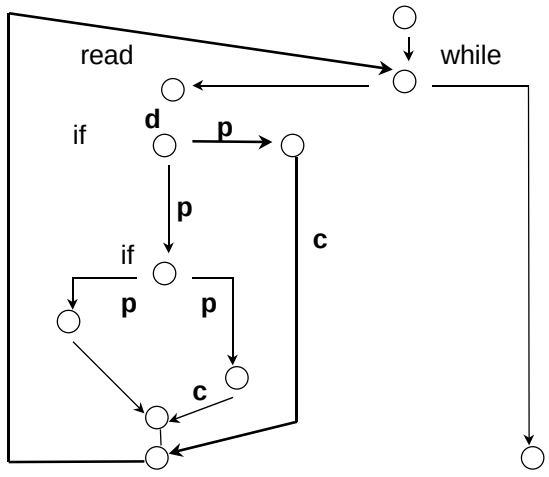


muuttujan **record** tilamuutokset merkittynä vuoverkkoon

Tietovuotestaus

- Määrittelytön segmentti muuttujan x suhteen:
  - x:lle arvon asettavasta linkistä alkava segmentti, joka ei sisällä muita x:lle arvon asettavia tai x:n tuhoavia linkkejä
- Tietovuotestauksessa ohjelmaa käydään läpi tilamuutosten d, u,c,p ja k ohjaamana

Tietovuotestaus



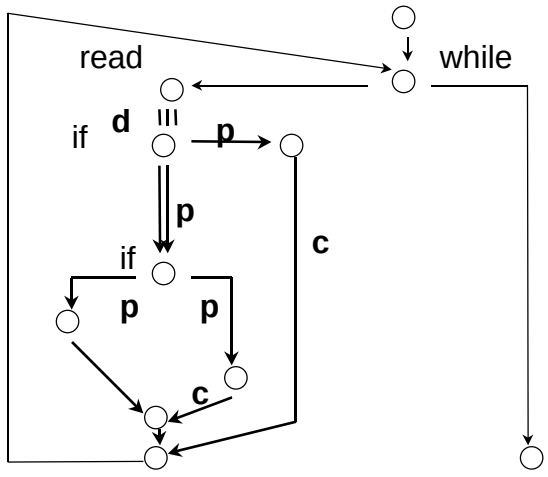
leveä viiva: esimerkki määrittelyttömästä segmentistä

Tässä esimerkissä uusi silmukan kierros aloittaa uuden määrittelyttömän segmentin

Tietovuotestaus

- kaikki käytöt strategia (all uses)
  - jokaisen muttujan x suhteen käydään läpi vähintäin yksi jokaisesta x:n määrittelystä (d) alkava määrittelytön segmentti jokaiseen kyseisen määrittelyn käyttöön (c,p)

Tietovuotestaus



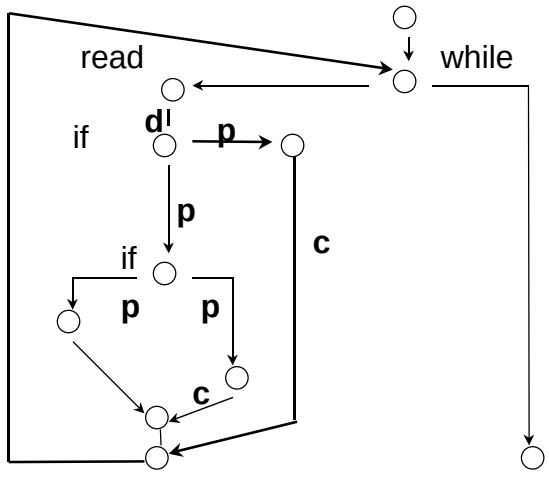
kaikki record käytöt

tarvitaan ainakin 3 testitietuetta

Tietovuotestaus

- kaikki määrittelyt strategia (all definitions)
  - jokaisen muuttujan suhteen käydään läpi vähintäin yksi jokaisesta x:n asetuksesta johonkin x:n käyttöön johtava segmentti
  - lievempi kuin kaikki käytöt

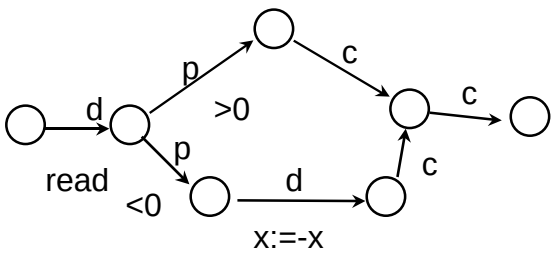
Tietovuotestaus



muuttujan record
kaikki määrittelyt
- riittää testata
yksi polku

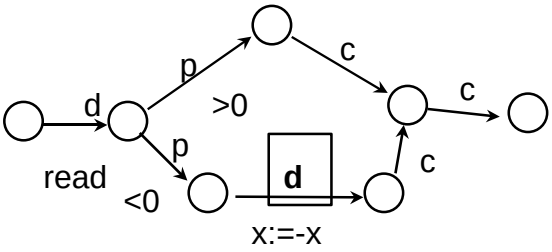
```
program example(input,output)
   var x, itseisarvo:integer;
begin
   read(x);
   if x>0 then
      itseisarvo:=x
   else begin
      x.=-x; itseisarvo:=x;
   end;
   write(x, itseisarvo);
end.
```

Tietovuotestaus- x:n tietovuoverkko



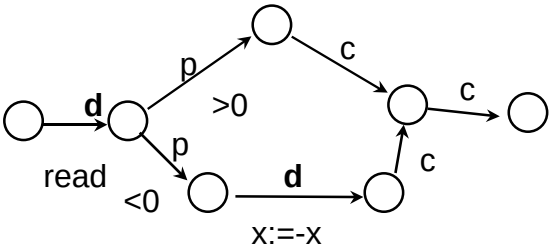
Tietovuotestaus- x:n tietovuoverkko

jos valitaan vaikkapa arvo 5 tarvitaan vielä toinenkin testiajo jotta kaikki määrittelyt toteutuisi



Tietovuotestaus- x:n tietovuoverkko

jos valitaan arvo -5 tarvitaan vain yksi testiajo kaikki määrittelyt strategialla



Tietovuotestaus

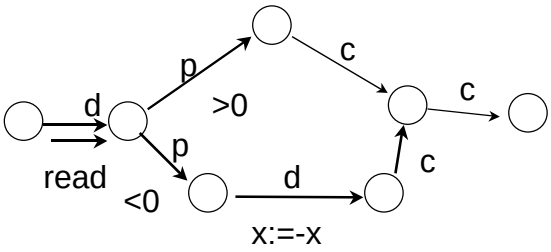
- kaikki du-polut strategia (all du-paths)
  - jokaisen muttujan x suhteen käydään läpi kaikki mahdolliset x:n asetuksesta alkavat ja sen käyttöön johtavat määrittelyttömät segmentit
  - tiukempi

- kaikki p käytöt, jotkin c-käytöt startegia
  - jokaisen muuttujan suhteen käydään läpi vähintään yksi joikaisesta x:n asetuksesta alkava määrittelytön segmentti jokaiseen kyseisen määrittelyn p-käyttöön, ellei tällaista ole valitaan vähintään yksi c-käyttöön johtava segmentti

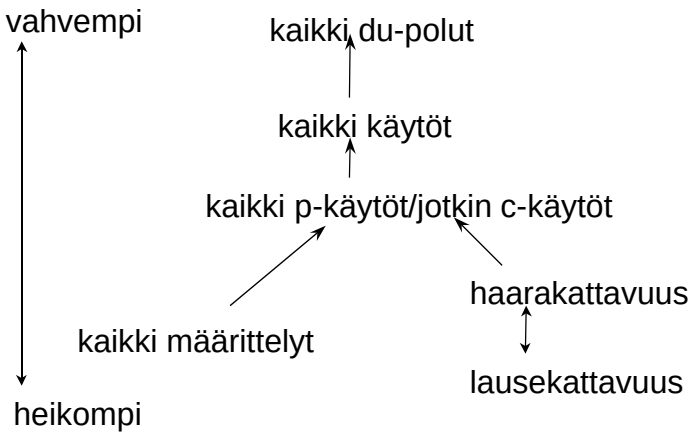
Tietovuotestaus- x:n tietovuoverkko

all du-paths
tarvitaan kaksi testiajoa

all p - some c
tarvitaan kaksi testiajoa



Testauskattavuuksien suhteita



Testaus ja virheen paikantaminen

- black box
  - moduuli tiedetään - ei tarkempaa sijaintia
- white box - polkutestaus
  - virhe sijaitsee jossain testatulla polulla
- white box - tietovuotestaus
  - jokin kuljetun polun d-linkki on virheellinen tai d-linlkki puuttuu

Esimerkki: HYTKY/Tietovuotoimitin

```
Testauksen kohde:   Perustoiminnot
Tavoite:        Osoittaa, että perustoiminnot
                toimivat määritysten mukaisesti.
Hyväksyminen:   Kaikki testit hyväksytään
                kummassakin testiympäristössä.
Testiympäristöt:
                Targa / Intel 486DX/33
                MS-Windows 3.11 ja
                IBM PS2/70 /Intel 386DX/25
                MS-Windows 3.1
```

Esimerkki: HYTKY/Tietovuotoimitin

Testit: Perustoiminnot oheisen käyttötapauskaavion mukaisesti

Käyttötapaus	Testi
Tallennus	Syntaksitestit - eri kokoisilla ja tyyppisillä aineistoilla Latautuvuustestit - eri kokoisilla ja tyyppisillä aineistoilla
Lataus	Latautuvuustestit
Kohteen valinta	Valintatestit
Prosessin luonti	Tasotestit Sijaintipaikkatestit jne ....

Testikuvaus: Tallennus / Syntaksitestit

Tarkoitus:	Varmistua, että malli tallentuu oikean muotoisena
Kriteeri:	Muoto määritelty suunnitelman liitteessä 3.
	Varmistus silmämääräisesti vertaamalla tulostiedostoa muotomäärittelyyn .
Riippuvuudet:	Kohteiden luontitestit tehtävä ennen näitä.
	Tehtävä ennen latautuvuustestejä.

Testitapaukset: Laaditaan vuorovaikutteisesti ohjelmistolla:

1. Tyhjä malli
   Tarkistetaan mallitietojen tallentuminen.
2. Yhden prosessin malli
   Alikaaviotietojen ja prosessitietojen tallentuminen.
3. Yhden tason usean kohteen malli.
   Kohteiden tiedot tallentuvat
4. 2 tason usean kohteen malli.
   Kaavioiden välisten riippuvuuksien tallentuminen.
5. 3 tason usean kohteen malli - yksipuolinen ja kaksipuolinen tarkennos
   Huom. Iisätasot samanlaisia, joten ei tarpeen testata.

Testikuvaus: Tallennus/Lataus Latautuvuustestit

Tarkoitus:	Varmistua siitä, että lataus ja tallennustoiminnot ovat yhteensopivia.
Kriteeri:	Tallennettu malli latautuu ohjelmistoon samanmuotoisena kuin se oli tallennettaessa ja kohteen valinta sekä sukellusoperaatiot toimivat ladatussa mallissa.
Riippuvuudet:	Kohteiden valintatestit ja sukellustestit tehtävä ennen näitä.Tallennuksen syntaksitestit tehtävä ennen.
Testitapaukset:	Testataan tallennuksen syntaksitestin yhteydessä luoduilla tallennustiedostoilla.

Oliorakenteen ja olio-ohjelman testaus

- Suunnitelmatasolla: käyttötapauspohjaisten testitapausten läpikäynti olioroolipelinä
  - (CRC-kortit)
    - olioiden ominaisuudet ja niiden palvelut kuvaavat kortit

- Perintä ja polymorfismi tuovat mukanaan testausongelmia (mutta poistavat myös joitakin)
  - voi olla vaikea hahmottaa minkä luokan palvelu varsinaisesti suoritetaan
  - ohjelmoijalla on voinut olla väärä käsitys siitä miten palvelu kehittyy luokkahierarkiasssa
  - palvelun valinta perustuu polymorfismiin, jolloin virheelliseen valintaehtoon liittyviä ongelmia ei esiinny

Oliorakenteen ja olio-ohjelman testaus

- Käyttötapauspohjaiset testitapaukset - testaavat yhteistyötä
  - testitapaukset käyttötapausten ilmentymiä

- Yksikkötestausta vastaa luokan testaus
  - metodien satunnaiset suoritusjärjestykset
  - elinkaaritestaus - tilakaavion pohjalta
  - ositus : testataan vain osaa luokasta
    - olion tilaa muuttavat & olion tilan säilyttävät metodit
    - attribuuttiperustainen ositus
    - toimintojen kategorisointiin pohjaustuva jaottelu (alustus, laskenta, kysely, …., lopetus)

Vuorovaikutteisten järjestelmien testauksesta:

- Käyttäjä ohjaa toimintaa
- Vaihtoehtoisia polkuja runsaasti
- Jos syötteet voidaan antaa vain vuorovaikutteisesti, tilanteen toisto hankalaa
  - nauhoitus, komentojono
- Pohjaksi tarvitaan malli käyttäjän toiminnoista esim. käyttötapausmalliin perustuva malli
- Ositetaan toiminta riittävän pieniin osiin

Vuorovaikutteisten järjestelmien testauksesta:

- Määritellään testitapaukset osakohtaisesti
  - missä tilassa oltava, voidaan käyttää apuna tila-automaattia
  - mitä tehdään
  - mitä pitäisi tapahtua
  - mistä nähdään, että se mitä piti tapahtua todella tapahtui

- Ohjelmat pitäisi toteuttaa siten, että toiminta on riippumatonta toimintaketjuista
  - riittää testata perustoiminnot - moodittomuus

Vuorovaikutteisten järjestelmien testauksesta:

- Tyypillisiä testattavia tilanteita ikkunatasolla:
  - ikkunoiden avautuminen
  - voiko ikkunoiden kokoa muuttaa
  - pystyykö tietoa käsittelemään kaikilla määritellyillä tavoin
  - uudistuuko ikkuna kun se tulee esiin toisen takaa tai kun se palautetaan

- ovatko kaikki toiminnot käytettävissä
- ovatko kaikki kontrollit esillä
- onko ikkunat oikein nimetty
- onko aktiivinen ikkuna selkesti merkitty
- päivittyvätkö ikkunat
- aiheuttavatko toistuvat tai virheelliset hiirivalinnat sivuvaikutuksia
- sulkeutuuko ikkuna oikein

Vuorovaikutteisten järjestelmien testauksesta:

- Tyypillistä testattavaa valikoihin ja hiirioperaatioihin liittyen
  - ovatko oikeat valikkovaihtoehdot tarjolla kussakin yhteydessä
  - toimivatko valikot asianmukaisesti
  - toimivatko vetovalikot kunnolla
  - ovatko valikkovaihtoehdot valittavissa hiirellä

- ovatko fontit oikein
- onko yhteyteen sopimattomat toiminnot estetty (harmaannutettu)
- ovatko vaihtoehtojen nimet kuvaavia
- toimiiko avustus
- tunnistaako järjestelmä hiirioperaatiot oikein eri tilanteessa
- muuttuuko hiirikursori tilanteen mukaisesti

Virheenjäljitys

- Varsin huonosti systematisoitu alue
  - virhe voi olla kaukana ulkoisesta ilmenemispaikastaan
  - virheen korjaaminen voi tilapäisesti estää muiden virheiden ilmenemisen
  - virheen syy saattaa olla käyttäjässä
  - virheellistä toimintaa voi olla vaikea toistaa
  - virhe voi johtua ohjelmointikielen tai laitteistoarkkitehtuurin erityispiirteistä

Virheenjäljitys

- ongelmanratkaisutilanne, johon auttaa
  - kokemus
  - näkökulman vaihto
- Jäljitysmenetelmiä
  - raaka työ
    - työkalujen mekaaninen käyttö:
    - muistivedokset (dump)
    - suoritusjäljitin (trace, debugger)
    - suuri määrä tulosteita
    - vaikea löytää tarvittavaa tietoa

Virheenjäljitys

  - peruutus
    - lähdetään havaitusta häiriöstä ja peruutetaan mahdollisia toimintapolkuja pitkin
    - tietovuoviipaloijaa käyttämällä (esim .HyperSoft)
      - ohjelma näyttää miten virhekohtaan johtavat tietovuot ja mahdollistaa niiden seuraamisen

- syiden eliminointi
  - induktio:
    - kerää virheeseen liittyvät tiedot
    - hypoteesi virheen syystä
    - hypoteesin testaus
  - deduktio:
    - kerää mahdolliset syyt
    - eliminoi syitä havaintojen perusteella

Virheiden korjaus

- Esiintyykö samantyyppinen virhe myös muualla ohjelmassa?
  - jos esiintyy, niin korjataan
- Aiheuttaako muutos mahdollisesti uusia virheitä ohjelmaan?
  - jäljitetään sivuvaikutukset
- Mitä olisi pitänyt tehdä toisin, ettei virhettä olisi syntynyt
  - vältetään jatkossa, oppiva tuotantoprosessi

Testaussuunnitelma

- Scope of testing
- test plan
  - test phases and builds
  - schedule
  - overhead software
  - environment and resources

- Test procedure for build n
  - order of integration
    - purpose
    - modules to be tested
  - unit tests for modules in build
    - description of test for module n
    - overhead software description
    - expected results
  - test environment
    - special tools and techniques
    - overhead software description

- Test case data
- Expected results for build n

- Actual test results

Testaussuunnitelma

- In all test phases:
  - interface integrity
  - functional validity
  - information contents
  - performance

Testien dokumentoinnista

- All tests traced to customer requirements
- tests should be planned long before testing begins
- Pareto principle: 80 % of uncovered errors caused by 20 % of modules
- testing in the small -> in the large
- exhaustive testing is not possible
- effective testing conducted by an independent third party

Testien valitsemisesta

- A good test
  - has a high probablity of finding an error
  - is not redundant
  - is neither too simple nor too complex
    - combinatory
    - masking of errors

Ohjelmiston laatukriteeri: testattavuus

- Operability - Toimivuus
- Observability - tarkkailtavuus
- Controllability - hallittavuus
- Decomposability - modularisuus
- Simplicity - suppeus
- Stability - testausaikaisten muutosten vähyys
- Understandability - ymmärrettävyys

Tarkastukset (inspection, review)

- Asiantuntijoiden suorittama ohjelmistodokumentin (myös koodi on dokumentti) manuaalinen läpikäynti, jolla pyritään löytämään virheitä
- tavoite:
  - laadunvarmistus tuotantoprosessin aikana, ennen ohjelmakoodin testausta

- tarkastukset tehokas tapa parantaa laatua
  - kun yksikkötestaus löytää n. 15-50% ja järjestelmätestaus 25-55% virheistä, löytyy määrämuotoisessa suunnittelun tarkastuksessa 45-65% ja määrämuotoisessa koodin tarkastuksessa 45-70% virheistä
    - (Jones: Software defect removal, (IEEE) Computer, April, 1996, pp 94-95)

Tarkastukset - roolit

- puheenjohtaja (moderator)
  - valvoo, että tarkastus sujuu sääntöjen mukaan
  - huolehtii siitä, että jokainen hoitaa oman tehtävänsä
- alustaja (reader)
  - johtaa dokumentin tarkastusta
  - toimii esilukijana, selittää/tulkitsee dokumentin sisältöä muulle ryhmälle, valitsee etenemisjärjestyksen

- tekijä (author)
  - dokumentin laatija
  - selittää kohdat, joita alustaja tai muut ryhmäläiset eivät ymmärrä
- sihteeri (recorder)
  - kirjaa löytyneet viat
  - auttaa puheenjohtajaa raportoinnissa
- tarkastaja (inspector)
  - etsii virheitä dokumentista
  - kaikki ryhmän jäsenet ovat tarkastajia

Tarkastukset (inspection, review)

- Voidaan käyttää määrämuotoisia lomakkeita
  - vikojen raportointiin
  - pöytäkirjaksi
  - yhteenvetoihin

Tarkastukset (inspection, review)

- Suunnittelu
  - onko dokumentti tarkastuskunnossa
  - roolien jako
  - ajan ja paikan varaus
- Toimitus/ esittely
  - tekijä esittelee dokumentin muille tarkastajille, jos tarpeen
  - dokumentti toimitetaan tarkastajille
- Tutustuminen
  - huolellinen tutustuminen dokumenttiin
  - virheiden merkintä

Tarkastukset (inspection, review)

- Tarkastustilaisuus
  - dokumentti käydään läpi alustajan johdolla
  - sihteeri kirjaa viat
  - parempi, että tekijä ei ole sihteerinä
  - virheitä ei korjata eikä syitä selvitetä kokouksessa
  - vian vakavuusaste kirjataan

- Jälkitoimet
  - virheet korjataan
  - mahdollinen uusintatarkastus
  - puheenjohtaja voi hyväksyä korjaukset