7. Verifiointi ja validointi

- Verifiointi ja validointi (V&V) on ohjelmistotuotannon työvaihe, missä varmistetaan, että
  - ohjelmisto täyttää sille asetetut implisiittiset ja eksplisiittiset vaatimukset ja
  - ohjelmisto täyttää sen tilanneen asiakkaan ohjelmistolle asettamat tarpeet.
- V&V:ta tehdään koko ohjelmistoprosessin elinkaaren ajan.

Verifioinnin ja validoinnin ero

- Verifioinnissa varmistetaan, että ohjelmisto vastaa määrityksiään:
  - "Verification: Are we building the product right?"
- Validoinnissa varmistetaan, että ohjelmisto täyttää asiakkaan sille asettamat odotukset:
  - "Validation: Are building the right product?"

Verifiointi- ja validointitekniikat

- V&V:ssa käytetään enimmäkseen kahta tekniikkaa: tarkastuksia ja testausta.
  - Tarkastukset (software inspections).
    - Tarkastuksissa joukko ihmisiä analysoi ja tarkastaa järjestelmäkuvauksia, kuten vaatimusdokumentaatiota, suunnittelukaavioita ja ohjelmakoodia. Tarkastukset ovat staattinen tekniikka. Ne eivät vaadi suorituskelpoista ohjelmaa.

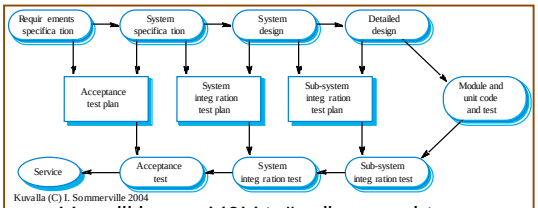
Verifiointi- ja validointitekniikat II

  - Testaus (software testing).
    - Testauksessa ohjelmistoa tai sen osaa suoritetaan tietyillä testitiedoilla ja tuloksia analysoimalla ja ohjelmiston suoritusta seuraamalla selvitetään, että toiminta on odotettua. Testaus on dynaaminen tekniikka, sillä siihen tarvitaan suorituskelpoinen ohjelma.
- Tarkastuksia voidaan tehdä koko ajan, testausta vasta ohjelmakoodin kanssa.
- Molempia tekniikoita tarvitaan V&V:ssa.

Verifioinnin ja validoinnin tavoite

- V&V:n tavoitteena on varmistaa, että ohjelmisto täyttää sille asetetut tavoitteet.
- Ohjelmiston ei tarvitse olla virheetön, eikä se myöskään yleensä ole sitä.
- Ohjelmiston on sen sijaan oltava tarkoitettuun käyttöön "riittävän hyvä".
- Tavoitetaso riippuu sovellusalueesta.

V&V:n hallinnan V-malli



Kuvalla (C) I. Sommerville 2004

- V-malli kuvaa V&V-työvaiheen suhteen muihin prosessin työvaiheisiin.
- Tarkastuksia voidaan pitää missä tahansa työvaiheessa tai niiden välillä.

7.1. Tarkastukset

- Tarkastus (inspection) on kokous, jossa tarkastetaan jonkin työvaiheen tuotos, tai osa siitä, ja yritetään löytää siitä puutteita ja virheitä.
  - Puute = vajaa määritys tai puuttuva toiminta.
  - Virhe = väärin tehty määritys tai ei-toivottu toiminta.

Tarkastukset - II

- Tarkastuksia tehdään kaikissa projektin työvaiheissa. Aina kun projektissa on saatu jotain valmiiksi, tulos kannattaa varmentaa tarkastuksella.
- Tarkastukset parantavat tuotteen laatua, sillä aikaisessa vaiheessa löydetty puute tai virhe on helpompi korjata kuin myöhemmin löydettynä.

Tarkastusten luonne

- Tarkastus on muodollinen tilaisuus, johon osallistuu 3-6 henkilöä.
- Tilaisuudella on tarkka aikataulu.
- Henkilöt edustavat eri sidosryhmiä asiakkaasta projektiryhmän jäseniin.
- Tarkastuksessa kootaan löydetyt puutteet ja virheet.
- Tarkastukseen valmistaudutaan etukäteen noin 2h ajan.

Tarkastukseen osallistujat

- Osallistujien roolit:
  - Puheenjohtaja (moderator): vastaa tarkastuksen aikataulusta ja ohjelmasta.
  - Sihteeri (scribe): kirjaa ylös löyd. asiat.
  - Alustaja (reader): kuvaa esitettävän asian.
  - Kirjoittaja (author/owner): edustaa dokumentin tekijöitä.
  - Tarkastaja (inspector): etsii dokumentista puutteita ja virheitä (kaikkien rooli).

Ennen tarkastusta

- Ennen tarkastustapahtumaa:
  - kaikki tarkastuksessa tarvittavat dokumentit ovat saatavilla,
  - osallistujilla on ollut aikaa tutustua dokumentteihin,
  - käytössä on tarkistuslistat yleisimmistä puutteista ja
  - tarkastettava dokumentti on sellaisella tasolla, että siinä ei ole ilmeisiä virheitä.

Tarkastustapahtuma

- Tarkastustapahtuma saa kestää korkeintaan kaksi tuntia. Siinä keskitytään yksinomaan löytämään puutteita ja virheitä.
- Tarkastukseen osallistujat eivät keskustele löydetyistä puutteista. Kun puute on havaittu, sihteeri kirjaa sen ylös ja siirrytään eteenpäin.

Tarkastuksen päätös

- Tarkastuksen lopuksi ryhmä äänestää tuotoksen hyväksymisestä:
  - Hyväksytään sellaisenaan: ei muutoksia.
  - Hyväksytään muutoksin: löydetyt puutteet ja virheet on korjattava, mutta tuotoksesta ei tarvita enää uutta tarkastusta.
  - Hylätään: löydetyt puutteet ja virheet on korjattava. Korjauksen jälkeen tuotoksesta käydään läpi uusi tarkastus.

Tarkastusten kultaiset säännöt

1. Arvioidaan tuotetta, ei tekijää.
2. Suunnitellaan aikataulu ja pidetään siitä kiinni.
3. Ei väittelyä.
4. Ei ratkota löydettyjä ongelmia.
5. Rajoitetaan osallistujien määrä 3-6 henkeen.
6. Valmistaudutaan huolellisesti tarkastukseen.
7. Käytetään tarkistuslistoja sekä valmistautuessa että tarkastuksessa.
8. Varataan riittävästi aikaa ja resursseja.
9. Koulutetaan osallistujat.
10. Pidetään tarkastuksessa kännykät kiinni!

7.2. Testaus

- Testauksella on kaksi tavoitetta:
  - Osoittaa sekä asiakkaille että projektille, että ohjelmisto täyttää sille asetetut vaatimukset. Tämä on *validointitestausta.*
  - Löytää ohjelmistosta puutteita ja virheitä, joiden johdosta ohjelmisto ei toimi, toimii väärin tai ei vastaa sille asetettuja määrittelyjä. Tämä on *määritysten ja syntaksin testausta* (Sommervillella termi on defect testing).

Täydellinen testaus mahdotonta

- Täydellinen testaus, missä ohjelma testataan kaikilla mahdollisilla syötteillä, syötekombinaatioilla ja ajoituksilla, ei ole käytännössä mahdollista.
  - Jo hyvin yksinkertaisilla ohjelmilla kaikkien testitapausten suoritus veisi vuosia.
- Tämän johdosta testauksessa valitaan osajoukko kaikista mahdollisista testitapauksista.

Testauksen oleellinen kysymys

- Miten valitaan sellainen testitapausten osajoukko, että
  - sen suorittaminen on mahdollista järkevässä ajassa ja että
  - sen avulla ohjelma tai sen osa saadaan testattua *riittävän hyvin*.
- Sommerville suosittelee vastaukseksi yrityskohtaista testauspolitiikkaa, jonka mukaan testitapaukset valitaan.

Testausvaiheet

- Sommerville jakaa testauksen kahtia:
  - *Komponenttitestauksessa* (Component testing) ohjelmaa testataan osina. Testatut osat kootaan yhteen ja testaan koottuina.
    - Komponenttitestaus kuuluu pääosin ohjelmiston kehittäjille (projektiryhmälle).
  - *Järjestelmätestauksessa* (System testing) testataan järjestelmää kokonaisuutena.
    - Järjestelmätestaus kuuluu pääosin ulkopuoliselle testausryhmälle.

Järjestelmätestaus

- Työvaihe on *liittävä:*
  - Kuhunkin osajärjestelmään liitetään komponentteja ja testataan komponenttien yhteistyö. Tämä on *integrointitestausta.*
  - Komponentit lisätään ja testataan yksi kerrallaan, kunnes kaikki osajärjestelmän komponentit on liitetty ja testattu.
  - Kun kaikki osajärjestelmät on koottu ja testattu, testataan vielä, että osajärjestelmät toimivat yhdessä oikein.

Integrointitestaus

- Integrointitestauksessa testataan valmiiden yksittäin toimivien komponenttien yhteistyö osajärjestelmässä.
- Integrointia voidaan tehdä ylhäältä alas:
  - Ensin tehdään osajärjestelmän runko ohjauskomponenteista, minkä jälkeen niihin liitetään toiminnat toteuttavat komponentit yksi kerrallaan.

Integrointitestaus II

- Integrointia voidaan tehdä yhtä lailla alhaalta ylös:
  - aloitetaan toimintakomponenteista ja edetään kohti korkean tason komponentteja.
- Käytännössä tehdään ns. voileipätestausta, missä integrointitestataan sekä alhaalta ylöspäin että ylhäältä alaspäin.

Rasitustestaus

- Integrointitestauksen jälkeen järjestelmän kriittisiä ominaisuuksia voidaan testata:
  - suorituskykyä,
  - luotettavuutta ja vikasietoisuutta,
  - turvallisuutta.
- Rasitustestauksessa ohjelma viedään äärirajoille ja mielellään vielä niiden yli.

Hyväksymistestaus

- *Hyväksymistestaus* (Acceptance testing, Sommervillella Release testing) tehdään sen jälkeen, kun integrointiestaus on saatu valmiiksi.
- Hyväksymistestauksen tarkoituksena on varmistaa, että tuote on sellaisessa kunnossa, että se voidaan antaa asiakkaalle tuotantokäyttöön.

Yksikkötestaus

- Komponenttitestaus, tai *yksikkötestaus,* (Component testing / Unit testing) tehdään ennen järjestelmätestausta.
  - Sommerville halusi esitellä järjestelmätestauksen ensin, koska nykyisin kaikille komponenteille ei tehdä yksikkötestausta.
    - Valmiina ostetut kaupalliset COTS-komponentit (Commercial Off the Shelf) ovat sellaisia, joille ei tehdä lainkaan yksikkötestausta.

Yksikkötestauksen tasot

- Yksikkötestausta tehdään jakamattomille ohjelman osille:
  - *Olioille*.
    - Olioa ei kannata hajottaa erikseen testattaviksi metodeiksi, sillä olion metodit ovat yleensä vahvasti sidoksissa toisiinsa.
  - *Koosteisille olioille tai olioryppäille*.
    - Joskus oliot ovat niin vahvasti sidoksissa toisiinsa, että niitä ei voi testata erikseen.

Yksikkötestauksen tasot II

- Komponenteille.
  - Yleensä komponentin toteuttavat oliot testataan erikseen ja integroidaan sen jälkeen komponentiksi. Tämän jälkeen komponentti voidaan vielä testata koosteisten olioiden tapaan jakamattomana kokonaisuutena.
  - Jos komponentit ovat kovin isoja, niiden toiminnallisuus kannattaa testata osina. Tällöin kannattaa erityisesti keskittyä komponentin rajapintoihin ja integrointiin.

Yksikkötestauksen testejä

- Testauksessa pitää kattaa testattavan yksikön kaikki ominaisuudet:
  - Jokainen operaatio (metodi) testataan erikseen.
  - Jokaisen attribuutin arvon asetus ja käyttö testataan.
  - Kaikki yksikön tilat ja siirtymät testataan. Samalla kaikki mahdolliset ulkoiset tapahtumat generoituvat, sillä ne siirtävät yksikön tilasta toiseen.

Rajapintatestaus

- Komponenttitestauksen oleellinen osa on rajapintatestaus. Siinä etsitään virheitä, jotka johtuvat rajapintojen virheistä tai vääristä rajapintojen oletuksista.
- Rajapinnat ja rajapintatestaus ovat erittäin tärkeitä, sillä sekä olioiden että komponenttien palvelut määritellään niiden rajapintojen kautta.

Rajapinnat

- Rajapintoja:
  - Parametrien välitysrajapinnat.
  - Jaetun resurssin rajapinnat: muisti ym.
  - Proseduraaliset rajapinnat: palvelukutsut
  - Viestinvälitysrajapinnat.
- Havaittavia virheitä:
  - Rajapintaa käytetään väärin.
  - Rajapinnan toimintaa ei ymmärretä.
  - Rajapinnan käytön ajoitus on väärä.

Rajapintatestauksen testejä

- Parametrien arvoalueiden äärirajoilla olevat testiarvot.
- Osoitinparametreille null-osoitin.
- Proseduraaliselle rajapinnalle poikkeukselliset kutsujärjestykset (esim. tiedoston luku ennen sen avaamista).
- Viestinvälitysrajapinnoille rasitustestaus.
- Jaetun resurssin rajapinnalle resurssien rinnakkaisten luku- ja kirjoitusprosessien suoritusjärjestystä vaihtelevat testit.

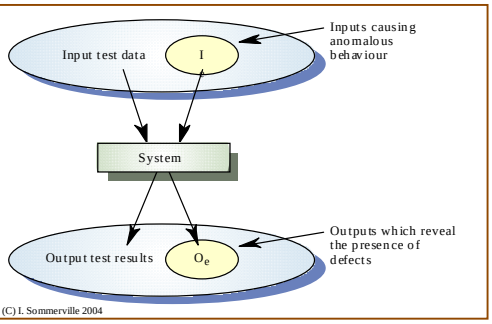
Testausmenetelmiä

- Testauksessa käytetään kahta yleistä menetelmää: *mustalaatikkotestausta* (black-box testing) ja *rakenteellista testausta* (structural testing)
  - Mustalaatikkotestauksessa järjestelmää testataan syötteiden ja tulosteiden avulla. Ohjelmakoodi ei ole näkyvillä.
  - Rakenteellisessa testauksessa järjestelmää testataan ohjelmakoodin avulla.

Mustalaatikkotestaus

- Mustalaatikkotestauksessa (black-box testing) testitapaukset valitaan ohjelman tai komponentin spesifikaation avulla.
- Itse ohjelma on ”musta laatikko”, jonka toiminnan määrittelevät sen saamat syötteet ja sen antamat tulosteet.
- Koska testaus perustuu spesifikaatioon, testauksen suunnittelu voidaan aloittaa heti määrittelyjen valmistuttua.

Mustalaatiikotestauksen anatomia



Kuvalla (C) I. Sommerville 2004

Mustalaatikkotestauksen tavoitteet

- Mustalaatikkotestauksessa pyritään löytämään
  - virheellisiä tai puuttuvia toimintoja,
  - sisäisiin, ulkoisiin tai käyttöliittymään liittyviä virheitä,
  - virheitä yleisissä tietorakenteissa,
  - suorituskykypuutteita ja
  - alustus- ja lopetustoimintojen virheitä.

Ositustestaus

- Mustalaatiikotestauksessa syöteavaruus ositetaan *ekvivalenssiluokiksi*. Yksi luokka kuvaa toiminnallisuuden osajoukkoa, jolle voidaan tehdä yhteisiä testejä.
- Ekvivalenssiluokat voidaan johtaa esimerkiksi ohjelmiston kuvauksista, arkkitehtuurista, käyttäjän vaatimuksista tai järjestelmävaatimuksista.

Testitapausten valinta

- Jokaisesta ekvivalenssiluokasta valitaan muutama testitapaus.
  - Valitut testitapaukset edustavat koko ekvivalenssiluokkaa.
  - Onnistuneessa jaottelussa mikä tahansa luokan jäsen kelpaa testitapaukseksi.
  - Käytännössä suositaan luokan rajoilla olevia arvoja, sillä ne ovat parhaimpia havaitsemaan jaottelun puutteita ja virheitä.

Ekvivalenssiluokkaesimerkki

- Olkoon meillä metodi
  boolean alkuluku(int n).
- Tällöin ekvivalenssiluokat voidaan valita vaikka seuraavasti.
  - n < 0
  - n = 0
  - n > 0 ja alkuluku
  - n > 0 ja ei ole alkuluku
  - n on jotain muuta kuin integer

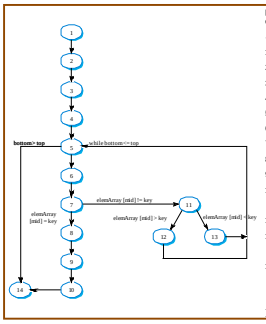
Rakenteellinen testaus

- Rakenteellisessa testauksessa testitapaukset valitaan ohjelmakoodin perusteella.
- Koodia tutkimalla löydetään sellaisia testitapauksia, joita on vaikea keksiä mustalaatikkotestauksella.
  - Esim. kaikkien koodissa käsiteltyjen poikkeustilanteiden testaus voidaan tehdä rakenteellisilla testausmenetelmillä.

Polkutestaus

- Polkutestaus on yleisin rakenteellisen testauksen menetelmä. Siinä lähtökohtana on koodista tehty siirtymäverkko (flow graph):
  - Jokaisesta lauseesta tulee solmu.
  - Jokaisesta siirtymästä lauseesta toiseen tulee suunnatun verkon särmä.
  - Verkossa on yksi lähtösolmu ja mahdollisesti monta maalisolmua.

Binäärihaun siirtymäverkko




```
public static void search(int key, int []
elemArray, Result r)
{
1.  int bottom = 0;
2.  int top = elemArray.length - 1; int mid;
3.  r.found = false;
4.  r.index = -1;
5.  while (bottom <= top) {
6.     mid = (top + bottom) / 2;
7.     if (elemArray[mid] == key) {
8.        r.index = mid;
9.        r.found = true;
10.       return;
       } else {
11.       if (elemArray[mid] < key)
12.         bottom = mid + 1;
          else
13.         top = mid - 1;
        }
    }
}
```

Kuvalla (C) I. Sommerville 2004

Siirtymäverkon polku

- Yksi siirtymäverkon polku on reitti lähtösolmusta johonkin maalisolmuun.
- Testauksessa tavoitteena on löytää sellainen testitapaus, joka kulkee halutun polun läpi.
- Aina tällaista testitapausta ei löydy, sillä metodissa saattaa olla kuollutta koodia.
- Täydellinen polkutestaus kävisi läpi kaikki mahdolliset polut.

Testattavat polut

- Yleensä kaikkien polkujen testaus ei ole mahdollista.
  - Esimerkiksi ehdot ja silmukat yhdessä kasvattavat nopeasti kaikkien mahdollisten polkujen määrää.
- Tämän johdosta valitaan osajoukko esimerkiksi seuraavilla yleisillä ehdoilla:
  - Kaikissa solmuissa on käytävä.
  - Kaikissa särmissä on käytävä.