Ohjelmistotestauksen perusteita II

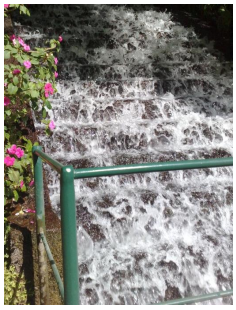
Luento 2

Antti-Pekka Tuovinen

Luennon oppimistavoitteet

- Testausprosessin perustoiminnot
- Testauksen psykologiaa
- Testauksen seitsemän periaatetta

Ohjelmistokehitysprosessit



Testauksen yleinen kulku

Kurssikirja, kuva 2-4 *Fundamental test process*

Testauksen perustoiminnot

- Testauksen suunnittelu ja ohjaus
- Testaustarpeiden analysointi ja testien suunnittelu
- Testien implementointi ja suoritus
- Testauksen lopetusehtojen täyttymisen toteaminen ja raportointi
- Testauksen päättäminen

Testauksen suunnittelu ja ohjaus

- Testaus on työmäärältään ja tarkoitukseltaan niin merkittävä osa ohjelmistoprojektia, että se vaatii oman suunnitelmansa
  - Suunnitelmaa on myös varauduttava muuttamaan projektin edetessä
- On mietittävä
  - Testauksen tavoitteet
  - Tarvittavat resurssit – henkilöstö, aika, laitteet jne.
  - Koulutustarpeet
  - Testauksen organisointi ja hallinto

Testauksen suunnittelu ja ohjaus

- Testauksen ohjaus (test control) tarkoittaa testausaktiviteettien seurantaa suhteessa testaussuunnitelmaan
  - Poikkeamien raportointi
  - Korjaavat toimet
  - Suunnitelman päivitys
- Testauksen hallintatehtäviin (test management) kuuluvat testausprosessin, testi-infran ja testiohjelmistojen hallinto ja ylläpito

Testausstrategia

- Testauksen suunnittelun päätehtävä on *testausstrategian* laatiminen
- Kuten aikaisemmin nähtiin, täydellinen (exhaustive) testaus ei ole mahdollista
- Testauskohteet ja testit on siis *priorisoitava* riskianalyysin perusteella
  - Kriittisiksi luokitellut ohjelmiston osat on testattava perusteellisimmin
- Tavoitteena on testauksen optimaalinen kohdentaminen ”oikeisiin paikkoihin”
- Esimerkki – VSR –järjestelmän testausstrategia (kurssikirja s. 20)

Testauksen suunnittelu ja ohjaus

- Testauksen suunnittelun kuuluu myös sen päättäminen, milloin testaus voidaan lopettaa (exit criteria)
- Lopetuskriteerien tarkka muoto riippuu käytetyistä testausmenetelmistä - esimerkiksi
  - Tietyn lausekattavuuden (90%) saavuttaminen
  - Tietty osuus (70%) kaikista järjestelmän transaktioista testattu vähintään kerran
- Lopetuskriteerien määrääminen kuvaa samalla testauksen *intensiteetin* (test intensity) jakautumisen eri osa-alueille (alijärjestelmät/toiminnot)

Testauksen suunnittelu ja ohjaus

- Ajan hallinta – kun tulee taas kiire…
  - Testien priorisointi ohjaa testaamaan kriittisimmät osa ensin, joten ne tulevat ainakin testatuiksi
- Testausympäristöjen hallinta
  - Testaustyökalujen valinta ja hankinta
  - Testikehikkojen ja -alustojen (test harness, test bed) kehittäminen otettava mukaan kehitystiimien työlistoille jo aikaisessa vaiheessa, jotta ohjelmistoa voidaan testata osissa kehitysprosessin aikana

Testaustarpeiden analysointi ja testien suunnittelu

- Testien suunnittelun lähtökohtia (test basis)
  - Vaatimukset
  - Ohjelmiston arkkitehtuuri
  - Riskianalyysit (virheiden/vikojen ”hinta”)
- Ensimmäinen askel on arvioida, onko riittävä pohja olemassa testien suunnittelulle
  - Esimerkiksi - onko tietty vaatimus kuvattu niin, että sen perusteella tiedetään, mikä on ohjelmiston odotettu käyttäytyminen/tuottama tulos ao. tilanteessa?

Testaustarpeiden analysointi ja testien suunnittelu

- Vaatimusten lisäksi myös testattavan ohjelmiston eli *testin kohteen* (test object) ominaisuuksilla on vaikutusta testien suunnitteluun
- Ohjelmiston testattavuus (testability) tarkoittaa
  - Miten hyvin ohjelmisto on jaettavissa erikseen testattavissa oleviin osiin
  - Miten hyvin testien tarvitsemat ohjelmalliset rajapinnat (toiminnot ja data) ovat olemassa ja käytettävissä
- Testattavuuden tarpeet onkin syytä ottaa huomioon ohjelmiston kehityksessä alusta lähtien

Testien suunnittelu

- Jäljitettävyys (traceability) vaatimusten/määrittelyjen ja konkreettisten testien välillä on tärkeää
  - Tiedetään mitkä testitapaukset (test case) liittyvät minkäkin vaaditun ominaisuuden testaamiseen

Testien suunnittelu

- Valittu testausstrategia määrittelee, mitä testaustekniikoita käytetään
  - Turvallisuuskriittisen ohjelmiston testaaminen täytyy tehdä erittäin systemaattisesti, huolellisesti ja kattavasti
- Myös testattavan ohjelman kompleksisuus vaikuttaa testaustekniikoiden valintaan
- Käytettävät testaustekniikat vaikuttavat luonnollisesti testien ja testitapausten suunnitteluun

Testaustermien määritelmiä

- Testiobjekti, testin kohde (test object)
  - Ohjelma tai ohjelman osa, jota varten testi suunnitellaan
- Testiajo (test run, test suite)
  - Yhden tai useamman testitapauksen suoritus koottuna yhteen ajoon
- Testiskenaario (test scenario)
  - Ketjuttaa yhteen useita testitapauksia siten, että edellisen tapauksen tuottamat tulokset toimivat seuraavan tapauksen lähtötilanteena

Testaustermien määritelmiä

- Testitapaus (test case)
  - Määrittelee *lähtötilanteen* eli *alkuehdot* testitapauksen suoritukselle (test conditions)
  - Määrittelee *syötteet* ja *odotetut tulokset* / testiobjektin *odotetun käyttäytymisen*
  - Hyvällä testitapauksella on *suuri todennäköisyys paljastaa aiemmin tuntemattomia vikoja*

Testaustermien määritelmiä

- Testioraakkeli (test oracle)
  - Testitapauksen odotetun/oikean tuloksen kertova (”ennustava”) lähde
  - Esimerkiksi ohjelman toimintaa kuvaava käyttötapaus (use-case) tai sen vaatimusmäärittelyssä annettu laskentakaava
  - Testitapauksen suunnittelija tarvitsee oraakkelia johtaakseen testitapauksen odotetun tuloksen, johon testiobjektin testin suorituksen aikana tuottamaa tulosta verrataan

Testitapausten suunnittelusta

- Suunnittelu etenee yleensä yleisemmistä tapauksista (logical test case) yksityiskohtaisempiin, konkreettisiin tapauksiin (concrete test case)
  - Ensiksi määritetään kullekin testitapaukselle vain arvoalueet, joista syötteet valitaan (looginen testitapaus, ei voida sellaisenaan suorittaa)
  - Myöhemmin kirjoitetaan konkreettiset suoritettavat testitapaukset, joihin valitaan tietyt syötearvot

Testitapausten suunnittelusta

- Testitapauksia voidaan määritellä projektin monessa eri vaiheessa
  - Käyttötapauksiin pohjautuvia toiminnallisia testitapauksia voidaan alkaa määritellä heti kun käyttötapaukset ovat selvillä
  - Ohjelman sisäiseen rakenteeseen perustuvat white-box testitapaukset voidaan määritellä vasta kun testattava koodi on olemassa

Testitapausten suunnittelusta

- Testitapauksia on kahta perustyyppiä
  - Testiobjektin toimintaa ja käyttäytymistä odotetuissa käyttötilanteissa ja *odotetuilla syötteillä* testaavat tapaukset
  - Testiobjektin toimintaa ja käyttäytymistä *odottamattomilla ja virheellisillä syötteillä* testaavat tapaukset
- Spesifioitujen virhetilanteiden ja poikkeusten käsittelyn testitapaukset kuuluvat ensimmäiseen kategoriaan
  - Ne ovat osa järjestelmän määrittelyä – ne kuvaavat *odotettua käyttäytymistä tietyissä erikoistilanteissa*

Testien implementointi ja suoritus

- Akitiviteetit
  - Luodaan konkreettiset testitapaukset
  - Testausympäristö valmistellaan käyttöön
  - Testit suoritetaan, ja
  - Tulokset kirjataan ylös

Testien suorituksen valmistelu

- Testitapauksiin voi olla syytä liittää selkeät ohjeet, kuinka ne suoritetaan
  - Jos kehittäjä itse ajaa testit, hän todennäköisesti jo tietää mitä tehdä
- Testitapausten priorisointi on tärkeää, jotta varmistetaan tärkeimpien testien suorittaminen viivästysten sattuessa (ajan puutteen tai teknisten ongelmien vuoksi)
- Yksittäiset testitapaukset on syytä koota testiajoihin (test suite) testauksen tehostamiseksi

Testien suorituksen valmistelu

- Usein testauksessa tarvitaan testikehikkoja (test harness), -ajureita tai simulaattoreita testien suorittamista varten
  - Varsinkin kun testataan osatoimintoja/alijärjestelmiä ohjelmiston kehitysversioissa
- Testikehikkojenkin toiminta on testattava ☺

Testien suoritus

- Testiobjekti asennetaan testausympäristöön ja varmistutaan, että kaikki tarvittavat osat ovat mukana ja ne latautuvat/käynnistyvät
- On suositeltavaa aloittaa testin suoritus aivan perustoimintoja testaavista tapauksista (smoke test)
  - Jos savu nousee, testausta ei ole mielekästä jatkaa, ennen kuin pahimmat virheet on korjattu
  - Jos yksinkertaiset perustapaukset menevät läpi, voidaan suorittaa loputkin testistä

Testauksen kirjanpito

- Jokaisen testitapauksen suorituksen tulos täytyy kirjata testilogiin
- Testilogin perusteella pitää voida nähdä, että testausstrategian mukaiset testit on tehty
  - Kuka on testannut minkäkin osan, milloin, kuinka intensiivisesti ja millaisin tuloksin
- Testilogin avulla voidaan perustella asiakkaille ja johdolle, että ohjelmisto on testattu vaaditulla kattavuudella

Virheen löytyessä

- Testien toistettavuus on tärkeää, joten kaikki testien suoritukseen liittyvät oleelliset parametrit on kirjattava ylös (ohjelmistoversio, suoritusympäristön kokoonpano jne.)
- Testitapauksen epäonnistuessa (saatu tulos ei vastaa odotettua → ”fail”) on ensiksi tutkittava, onko kyseessä todella virhe testiobjektissa vai testin suorituksessa
- Jos kyseessä todella on virhe, löydös on dokumentoitava (incident handling) korjaustoimenpiteitä varten

Virheen korjauksen jälkeen

- Jos löydetty virhe todettiin korjausta vaativaksi, siihen johtanut vika aikanaan korjataan
- Korjauksen jälkeen on suoritettava uudelleen testit, joiden tuloksena virhe löytyi, ja varmistettava, että virhe on poistunut
- Riippuen vian laadusta ja laajuudesta, on joskus myös varmistettava, että vian/vikojen korjaus ei ole aiheuttanut uusia virheitä (regression test)
  - Voidaan joutua suorittamaan uudelleen isokin joukko testejä

Virheen korjauksen jälkeen

- Käytännössä jokaisen yksittäisen virheen korjauksen jälkeen ei tehdä heti uudelleentestausta, vaan testaus tehdään testiobjektin seuraavan kehitysversion julkaisulle
- Testiobjektin kehitysversiot sisältävät yleensä monien virheiden korjauksia
  - Korjauksilla voi olla ennalta arvaamattomia yhteisvaikutuksia, jotka johtavat kokonaan uusiin virheisiin tai aikaisemmin paljastumatta jääneiden vikojen paljastumiseen
- Uutta koodia varten voidaan joutua määrittelemään uusia testitapauksia

Testauksen päättäminen

- Osana testauksen suunnittelua päätettiin testauksen päättävistä lopetusehdoista (exit criteria)
- Yksinkertaistaen: kun kaikki testit on suoritettu ja lopetusehdot täyttyvät (testien tulosten suhteen) testaus voidaan lopettaa
- Jos tavoitteet eivät kuitenkaan täysin täyty, on syytä arvioida pitääkö testausta jatkaa, vai voidaanko kriteerejä löysentää
  - Saadaanko resurssien lisäkäytöstä sittenkään riittävää hyötyä?

Testauksen päättäminen

- Muita käytettyjä kriteereitä
  - Failure rate (kurssikirja kuva 2-5) eli uusien virheiden löytymisen tahti putoaa tietyn kynnysarvon alapuolelle
    - Virheiden vakavuus otettava kuitenkin huomioon
  - Ajan ja loppuminen ja kustannusten nousu!
    - Valitettava mutta yleinen peruste lopettaa testaus

Lopetustoimet

- ”Lessons learned” eli Mitä tästä opittiin?
  - Poikkeamat suunnitelmista ja niiden syyt
  - Koko testausprosessin tulosten evaluointi
  - Parannusehdotusten kerääminen
- Testitapausten, -datan ja työkalujen ym. (testware) arkistointi tai luovutus ylläpitoa varten

Testauksen psykologiaa

- Testaajan tehtävä on löytää virheitä
  - Missio: Keksi mahdollisimman monta tapaa rikkoa tämä softa
- Testausta pidetään helposti destruktiivisena toimintana, vaikka se vaatii luovuutta ja älyä
- Erityisesti tutkiva testaus (exploratory testing) korostaa testaajan omaa panosta ja asiantuntemusta testauksen onnistumisessa

Testaava kehittäjä?

- Useimmat kehittäjät haluavat mieluummin näyttää, että heidän koodinsa toimii kuin että se ei toimi
- Tehokkaan (= virheitä löytävän) testauksen kannalta kehittäjä ei ehkä pysty ottamaan riittävää etäisyyttä omaan koodiinsa
  - Testitapauksista tulee helposti liian optimistisia ja todella ilkeät tapaukset jäävät testaamatta
  - Perustavanlaatuiset vaatimusten väärinymmärrykset jäävät löytymättä!

Erillisen testaustiimin käyttö

- Ohjelmistokehittäjistä riippumattoman testaustiimin käyttö yleensä parantaa testauksen laatua ja kattavuutta
  - Ei ennakkoasenteita testiobjektia kohtaan
  - Erityisosaamista ja kokemusta testauksesta
- Vaatii kuitenkin perehtymistä testattavaan ohjelmaan

Testauksen löytämien virheiden raportointi

- Virheiden raportointi vaatii huolellisuutta ja asiallisuutta
  - Vältettävä vastakkainasettelun syntymistä kommunikoinnissa
  - Virheraporttien täytyy sisältää riittävät tiedot virheen toistamiseksi kehittäjien ympäristössä
- Täytyy sopia etukäteen siitä mikä katsotaan virheeksi ja mikä ei
  - ”It’s a feature”
  - ”Works as specified”
- Tarvitaan molemminpuolista kunnioitusta ja ymmärrystä toisten työn luonteesta

Testauksen seitsemän periaatetta

1. Testauksella voi vain osoittaa, että vikoja on, mutta sillä ei voi osoittaa, että niitä ei ole
2. Täydellinen (exhaustive) testaaminen ei ole mahdollista
3. Testausaktiviteetit tulisi aloittaa mahdollisimman aikaisin projektissa
4. Viat kerääntyvät usein yhteen

Testauksen seitsemän periaatetta

5. Samanlaisina toistetut testit menettävät tehoaan ajan mittaan
6. Testaus on kontekstiriippuvaa
7. Virheettömyys ei välttämättä tarkoita sitä, että testattu järjestelmä on käyttäjilleen hyödyllinen

Luennon oppimistavoitteet

- Testausprosessin perustoiminnot
- Testauksen psykologiaa
- Testauksen seitsemän periaatetta