Testaus osana ohjelmistojen elinkaarta I

Luento 3

Antti-Pekka Tuovinen

Oppimistavoitteet

- Ohjelmistokehityksen V-malli
- Testauksen tasot
  - Komponenttitestaus
  - Integrointitestaus
  - Järjestelmätestaus
  - Hyväksyntätestaus

Ohjelmistokehityksen V-malli

- *Kurssikirjan kuva 3-1 General V-model s. 40*
- Nostaa testauksen muitten kehitysaktiviteettien rinnalle
  - Korostaa testauksen suunnitelmallisuutta
- Malli tunnistaa ohjelmiston eri rakennetasot, jotka vaativat omanlaisensa testauksen
  - Tavoitteet, tekniikat, osaaminen

V-mallin Veet

- *Validointi* – soveltuuko ohjelma ajateltuun käyttöönsä?
- *Verifiointi* – onko ohjelman osa toteutettu määrittelynsä mukaisesti?
- Validoinnin rooli korostuu V-mallin ylemmillä testaustasoilla

Testaustasot - Komponenttitestaus

- A.k.a *Yksikkötestaus*
- Ohjelman pienimpien toiminnallisten rakenneosien testausta
  - Luokka, moduuli, funktio, skripti, …

Komponenttitestauksen lähtökohdat

- Komponentit testataan yksitellen ja erillään muista komponenteista
  - Löydetyt virheet johtuvat testattavassa komponentissa olevista vioista
- Testataan komponentin sisäistä toimintaa ja käyttäytymistä
  - White-box testauksen rooli korostuu

Komponenttitestauksen ympäristö

- Testaus täytyy tehdä läheisessä yhteistyössä kehittäjien kanssa - usein kehittäjät tekevät yksikkötestauksen itse
- Tarvitaan *testiajuri* (test driver), joka
  - Alustaa testitapausten suorituksen
  - Kutsuu komponentin palveluja/toimintoja
  - Vastaanottaa kutsujen tuottamat tulokset ja vertaa niitä odotettuihin arvoihin
  - Kirjaa testien suorituksen ja tulokset (pass/fail)

Komponenttitestauksen ympäristö

- Yksikkötestauskehikot (test framework, test fixture)
  - xUnit arkkitehtuuri
  - JUnit Javalle (www.junit.org)
- Esimerkki JUnit:n käytöstä NetBeans kehitysympäristössä: https://wiki.helsinki.fi/display/ohma/JUnit

Komponenttitestauksen tavoitteet

- Toiminnallisuuden verifiointi
  - Varmistetaan oikea toiminta valituilla testitapauksilla eli syöte/tulos kombinaatioilla
- Vikasietoisuuden (robustness) testaus
  - Testataan toimintaa virheellisillä (ja määrittelemättömillä) syötteillä
  - Testataan toimintaa muissa poikkeustilanteissa
  - Engl. *negative tests*

Komponenttitestauksen tavoitteet

- Tehokkuustestit
  - Laskentaresurssien käyttö, suoritusnopeus
  - Vain kriittisille komponenteille (sulautetut ohjelmistot, tiukat aikavaatimukset)
- Ylläpidettävyyden testaaminen
  - Käytetään *staattista analyysiä*, ei ohjelman suorittamista
  - Koodin rakenne, modulaarisuus, kommentointi, koodauskonventioiden noudattaminen, ymmärrettävyys jne.

Testilähtöinen kehitys

- *Test Driven Development* (TDD)
- Suosittu iteratiivisessa ja ketterässä kehityksessä
- Koodattavan moduulin/luokan *yksikkötestit kirjoitetaan ensin*
  - Aluksi testi epäonnistuu, koska implementaatiota ei vielä ole
  - Koodausta jatketaan, ja kun kaikki testit lopulta menevät läpi, implementaatio on valmis
- Testit automatisoidaan ja ne ajetaan koodin muuttuessa

Testaustasot - Integrointitestaus

- Integrointi = komponenttien/yksiköiden koostaminen suuremmiksi rakenneyksiköiksi (alijärjestelmiksi)
- Integroitavat komponentit on jo testattu
- Tavoitteena on löytää virheitä yhdistettyjen komponenttien
  - Rajapinnoista (interface)
  - Vuorovaikutuksesta (interaction)

Testaustasot - Integrointitestaus

- Eri tiimien toteuttamien komponenttien vuorovaikutus/yhteistoiminta on altis puutteellisen määrittelyn, väärinymmärrysten ja heikon kommunikaation aiheuttamille vioille
- Eri tahtiin etenevä kehitystyö asettaa haasteensa komponenttien yhteistoiminnan toteutttamiselle
- Erityisen hankalia ovat tapaukset, joissa vieraan komponentit käyttävät sisäisiksi (ei-julkiksi) tarkoitettuja rajapintoja, jotka voivat muuttua odottamatta

Integrointitestauksen lähtökohdat

- Testitapausten pohjana käytetään arkkitehtuurisuunnittelua
  - Arkkitehtuuri / järjestelmäarkkitehtuuri
  - Käyttötapaukset
  - Työnkulkujen kuvaukset (workflow)
- Valmisohjelmistojen/komponenttien (COTS) vuorovaikutus itse kehitettyjen komponenttien kanssa kuuluu integraatiotestauksen piiriin

Integrointitestauksen ympäristö

- Pyritään käyttämään hyväksi komponenttitestauksen testiajureita
  - Komponentit tarjoavat palvelunsa rajapintojen kautta, joita komponenttitestauksenkin ajurit käyttävät
- Integroitujen komponenttien yhteistoiminnan seurantaa varten tarvitaan *monitoreita* (monitor)
  - Komponenttien välisen dataliikenteen lukeminen ja kirjaaminen lokiin

Integrointitestauksen tavoitteet

- Integroitavien komponenttien yhteistoiminnan virheiden ja ristiriitojen löytäminen
  - Käännösaikaiset (staattiset) rajapintamäärittelyjen ristiriidat löytyvät yleensä koostamisen (build) aikana
  - Suoritusaikaiset (protokolla-) viat löytyvät vain testaamalla

Integrointitestauksen tavoitteet

- Tyypillisiä kommunikaatioon liittyviä vikoja
  - Syntaktisesti vääränmuotoisen datan lähettäminen tai datan lähettämättä jättäminen aiheuttaa poikkeuksen vastaanottavassa komponentissa
  - Datan välitys toimii, mutta komponentit tulkitsevat datan merkityksen eri tavoin
  - Tiedonsiirrossa on ajoitusongelmia
    - Väärään aikaan, liian myöhään, liian tiheään …

Integrointitestauksen tavoitteet

- K: Voisiko komponenttitestauksen jättää pois, ja ajaa kaikki testit vasta integroinnin jälkeen?
- V: Ei ole järkevää. Perusteluita mietitään tarkemmin seuraavissa laskareissa.

Integrointistrategioista

- Komponentit valmistuvat usein eri tahtiin, jolloin voi olla vaikea etukäteen tietää, milloin integraatiotestausta päästään tekemään
- Testauspäällikön (test manager) on tehtävä integrointitestausta varten suunnitelma, joka ottaa huomioon ohjelmiston testausstrategian, arkkitehtuurin ja projektisuunnitelman
- Parhaassa tapauksessa testaustarpeet otetaan jo huomioon projektisuunnitelmassa ja komponenttien implementointijärjestyksessä

Yleisiä integrointistrategioita

- *Top-down*
- *Bottom-up*
- *Ad hoc*
- *Backbone / skeleton*

- …ja tietysti viimeiseen asti vältettävä, ei-inkrementaalinen *Big Bang*
  - Lähinnä seurausta integrointistrategian puuttumisesta!

Testaustasot Järjestelmätestaus

- Tuo mukaan *asiakkaan* ja *käyttäjän* näkökulman teknisten vaatimusten rinnalle
- Monien toimintojen suorittaminen ja järjestelmän piirteiden havainnointi vaativat kaikkien komponenttien yhteistoimintaa
  - Niitä voidaan testata vain järjestelmätasolla

Järjestelmätestauksen lähtökohdat

- Järjestelmä- ja ohjelmistovaatimukset, määrittelyt, käyttöoppaat, asennusohjeet, ylläpito-ohjeet jne.
- Testit suoritetaan mahdollisimman samanlaisessa laitteisto- ja ohjelmistoympäristössä kuin lopullinen käyttöympäristö
- Ohjelmiston eri konfiguraatiot ja suorituskyky eri tilanteissa on myös testattava

Järjestelmätestauksen lähtökohdat

- Datan laatu on entistä tärkeämpää nykyisissä ohjelmistoissa
  - Järjestelmän käyttämän datan eheys, täydellisyys ja ajantasaisuus on varmistettava

Järjestelmätestauksen lähtökohdat

- Järjestelmätestaus on syytä tehdä erillisessä testausympäristössä – ei asiakkaan tuotantoympäristössä
  - Vältetään testattavan ohjelmiston vioista tuotantoympäristöön aiheutuvat vahingot
  - Testausolosuhteita (test condition) voi olla vaikeaa hallita tuotantoympäristössä
  - Testien toistettavuus on parempi

Järjestelmätestauksen tavoitteet

- Vaatimusten väärästä, epätäydellisestä tai epäyhtenäisestä implementoinnista johtuvien virheiden löytäminen
- Puuttuvien tai unohdettujen vaatimusten huomaaminen ja tunnistaminen

Järjestelmätestauksen ongelmia

- Jos vaatimuksia ei ole kirjattu ylös, järjestelmätestien suunnittelijoiden vaikeana tehtävänä on haalia tarvittava informaatio hajallaan olevista lähteistä
- Tässä tilanteessa paljastuu yleensä myös erilaisia tulkintoja samoista vaatimuksista, jolloin testaajien täytyy ottaa aloite yhteisen näkemyksen muodostamiseksi (testitapausten tuottamiseksi)
- Näissä olosuhteissa voi olla parasta turvautua tutkivaan testaukseen (exploratory testing)

Testaustasot - Hyväksyntätestaus

- Edellä kuvatut testit tekee ohjelmiston toimittaja
- Ennen ohjelmiston ottamista käyttöön asiakkaankin on osallistuttava validointiin
  - Asiakkaan/käyttäjän näkökulma ja evaluointi
  - Erityisen tärkeää räätälöidyille ohjelmistoille
  - Vakiintuneen valmisohjelmiston (COTS) hankinnan yhteydessä tehtävät hyväksyntätestaus voi olla kevyempi

Hyväksyntätestauksen lähtökohdat

- Mikä tahansa järjestelmää käyttäjän/asiakkaan näkökulmasta kuvaava dokumentaatio
  - Käyttötapaukset, liiketoimintaprosessit, lait ja säännökset, tietohallinnon ja ylläpidon säännöt ja prosessit …

Hyväksyntätestauksen lähtökohdat

- Hyväksyntätestejä voidaan tehdä osana muittenkin tasojen testausta
  - Valmisohjelmistojen käyttöä voidaan testata jo integroinnin aikana
  - Käyttöliittymään liittyvien komponenttien käytettävyyttä voidaan testata komponenttitestauksen yhteydessä
  - Prototyyppejä voidaan käyttää uuden toiminnallisuuden testaamiseen jo ennen järjestelmätestausta
- Asiakkaan ja käyttäjän näkökulman saaminen mukaan projektin alkuvaiheessa on tavoiteltava asia
  - Ketterien menetelmien yksi kulmakivi

Hyväksyntätestauksen muodot

1. Sopimukseen kirjattujen hyväksymisehtojen täyttymisen testaus
2. Loppukäyttäjän hyväksyntätestaus
3. Tuotannon/operoinnin hyväksyntätestaus
4. Kenttätestaus

Hyväksymisehtojen testaus

- Asiakas varmistaa, että ohjelmistossa ei ole (merkittäviä) puutteita ja että ohjelmisto muuten on tilaussopimuksen mukainen
- Toimittajan ja asiakkaan välisessä sopimuksessa on usein kirjattu erikseen hyväksymisehdot (acceptance criteria)
- Toimittajan pitäisi olla testannut ehtojen täyttyminen jo omissa järjestelmätesteissään
- Yleensä riittää hyväksynnän kannalta riittävien testien toistaminen yhdessä asiakkaan kanssa

Hyväksymisehtojen testaus

- On tärkeää, että asiakas on *itse määritellyt* hyväksymistestauksen testitapaukset tai ainakin ne huolellisesti katselmoinut
- Toisin kuin järjestelmätestaus, hyväksyntätestaus tehdään nyt *asiakkaan (tuotanto-)ympäristössä*
- Myös ohjelmiston jakelu ja asennus testataan asiakkaan ympäristössä

Loppukäyttäjän hyväksyntätestaus

- Asiakas ja loppukäyttäjät voivat olla eri henkilöitä
- Eri käyttäjäryhmillä on erilaisia tarpeita ja odotuksia
  - Ohjelmistoa päivittäin työssään käyttävät
  - Satunnaiset käyttäjät
  - Ylläpidon ja tuotannon työntekijät, jotka vastaavat ohjelmiston saatavuudesta
  - Johto, jolla on omia seuranta- ja raportointitarpeitaan
- Asiakkaan tehtävänä on usein valita testitapaukset kullekin käyttäjäryhmälle

Loppukäyttäjän hyväksyntätestaus

- Tässä vaiheessa havaittujen merkittävien ongelmien tai virheiden korjaaminen voi olla hyvin kallista…
- Siksi on syytä esimerkiksi *prototyyppien ja käytettävyystestien* kautta varmistua jo projektin aikaisessa vaiheessa, että kehitettävä ohjelmisto tulee täyttämään käyttäjäryhmien tarpeet
  - Yleisimpien käyttäjän tehtävien pitää onnistua helposti ja intuitiivisesti; harvemmin tarvittavien toimintojen käyttämiseen voidaan edellyttää käyttöoppaiden lukemista

Tuotannon/operoinnin hyväksyntätestaus

- Validoidaan ohjelmiston ominaisuudet ylläpidon (system administration) näkökulmasta
  - Varmistukset
  - Toipuminen vakavista häiriöistä (disaster recovery)
  - Käyttäjien hallinta
  - Turvaominaisuudet
  - Jne.

Kenttätestaus

- Ohjelmistoa voidaan käyttää hyvin monenlaisissa ympäristöissä
- Kenttätesteillä (field test) tavoitteena on tunnistaa eri käyttöympäristöjen mukanaan tuomat vaikutukset, joita on vaikea ennakoida
  - Tärkeää erityisesti COTS ohjelmille
- Valituille käyttäjille toimitetaan esiversioita ohjelmistosta, jotka antavat siitä palautetta toimittajalle
  - Alpha-, Beta-testaus

Oppimistavoitteet

- Ohjelmistokehityksen V-malli
- Testauksen tasot
  - Komponenttitestaus
  - Integrointitestaus
  - Järjestelmätestaus
  - Hyväksyntätestaus