Testaus osana ohjelmistojen elinkaarta II

Luento 4

Antti-Pekka Tuovinen

Oppimistavoitteet

- Ohjelmistoversioiden testaus
- Testityyppejä

Ohjelmistoversioiden testaus

Ohjelmistojen kehitys ei pysähdy

- V-mallin ja testaustasojen esittely keskittyi uuden ohjelmiston kehittämisessä tehtävään testaukseen
- Ohjelmiston elinaika kuitenkin vasta alkaa ensimmäisestä julkaistusta versiosta
  - Vikoja korjataan
  - Asiakas haluaa muutoksia
  - Toimintaympäristö muuttuu
  - Tehdään uusia (räätälöityjä) versioita uusille asiakkaille
  - Julkaistaan uusia versioita tuotesuunnitelman mukaisesti

Ohjelmistoversioihin liittyvä testaus

- Tarkastellaan asiaa kolmesta näkökulmasta
  - Ylläpito
  - Ohjelmiston tuoteversioiden kehitys
  - Inkrementaalinen ohjelmistokehitys

Ohjelmistojen ylläpito

- Ohjelmistoon ei käytön aikana tule vikoja fyysisen kulumisen tai rasituksen takia
- Kaikki viat ovat kehitysaikaisten erehdysten syytä
- Ohjelmistojen ylläpito (maintenance) on siis erilaista kuin esimerkiksi kiinteistöjen tai tuotantolaitteiden ylläpito

Tyypillisiä ylläpitotoimiin johtavia tilanteita

- Ohjelmistosta löytyy vikoja käytössä
- Ohjelmistoa ajetaan olosuhteissa, joihin ei ole osattu varautua
- Asiakkaalla on muutostoiveita
- Harvoin esiintyvien erikoistilanteiden vaatimat toiminnot ovat jääneet toteuttamatta
- Järjestelmässä ilmenee epävakautta satunnaisesti tai pitkien käyttöaikojen jälkeen

Ohjelmistojen ylläpito

- Ohjelmistojen ylläpitoa on kahta perustyyppiä
  - *Korjaavassa* ylläpidossa (corrective maintenance) poistetaan virheitä
  - *Muokkaavassa* ylläpidossa (adaptive maintenance) ohjelmistoa muokataan muuttuneisiin olosuhteisiin
- Ohjelmistoon tehdään muutoksia, jotka kootaan päivityspakettiin tai kokonaiseen uuteen ohjelmistoversioon ja toimitetaan asiakkaalle

Ylläpitopäivitysten testaus

- Tehdyt muutokset on luonnollisesti testattava ennen päivityksen julkaisua
- Testausstrategia on suoraviivainen
  - Muutetut ohjelman osat testataan (uudelleen), tarvittaessa uusilla testitapauksilla
  - Muulle ohjelmistolle tehdään *regressiotestaus* (kts. kalvo 32 →)

Päivitysjulkaisut

- Pika- tai hätäpäivitys (emergency / hot fix)
  - Tehdään vain välttämättomimmät testit nopean julkaisun mahdollistamiseksi
  - Kattavammat testit julkaisun jälkeen
- Suunnitellut korjauspäivitykset (update)
  - Julkaistaan yleensä säännöllisin väliajoin
  - Uusi toiminallisuus on testattu normaalisti ja muu ohjelmisto on regressiotestattu

Ohjelmistojen ylläpitoon liittyvä testaus

- Muita testausta vaativia muutoksia
  - Ohjelmiston migraatio alustalta toiselle
  - Ohjelmiston poisto käytöstä

Ohjelmistojen tuoteversioiden testaus

- Korjaavan ylläpidon ohella ohjelmistoihin tehdään *etukäteen suunniteltuja muutoksia* ja *laajennuksia* pidemmällä tähtäimellä
  - Proaktiivisuus vs. reaktiivisuus
- Ohjelmistolla on *tuotesuunnitelma* (product roadmap), johon sisältyy useita julkaisuja
  - Uusia piirteitä
  - Tuen uusille alustoille

Ohjelmistojen tuoteversioiden testaus

- Käytännössä uuden julkaisun kehittäminen käynnistää uuden kehitysprojektin
  - Kehitys on *syklistä* ja *iteratiivista*
- Uusien versioiden julkaisut kannattaa synkronoida ylläpitoon liittyvien korjauspäivitysten julkaisutahtiin
  - Esim. tuotepäivitys 12kk välein ja korjauspäivitys 6kk julkaisusta

Ohjelmistojen tuoteversioiden testaus

- Uudet ohjelmistoversiot pitäisi testata kaikki testaustasot läpikäyden, kuten ensimmäinenkin versio
  - Muutetut ja lisätyt osat testataan
  - Uudet osat vaativat uudet testitapaukset, mahdollisesti myös muutetut osat
  - Muulle ohjelmistolle tehdään regressiotestaus

Inkrementaalinen ohjelmistokehitys

- Inkrementaalisen kehityksen idea on tuottaa ohjelmisto sarjana pienempiä kehitysaskeleita sen sijaan, että ohjelmisto rakennetaan kerralla valmiiksi
- Jokainen kehitysaskel eli inkrementti lisää uutta toiminnallisuutta järjestelmään kasvattaen ohjelmistoa siivu siivulta
- Ohjelmistolla on varhaisesta vaiheesta lähtien jonkinlainen perusrunko tai luuranko (skeleton), johon inkrementtien tuotokset integroidaan

Inkrementaalinen ohjelmistokehitys

by Man vyi
wikimedia.org

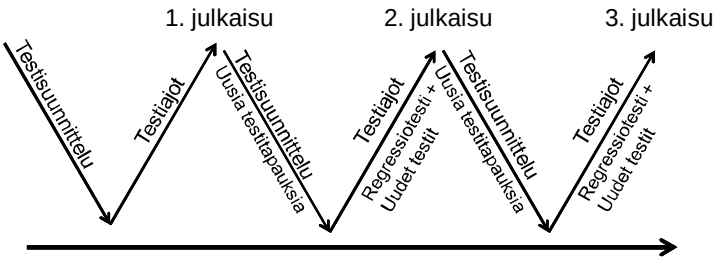
Inkrementaalinen ohjelmistokehitys

- Inkrementaalinen kehitysprosessi tekee mahdolliseksi asiakkaan osallistumisen jo varhaisessa vaiheessa toimintojen testaukseen
- Hyväksyntätestausta ja käytettävyystestausta voidaan tehdä koko kehitysprojektin ajan

Inkrementaalinen ohjelmistokehitys

- Muukin testaus on mukautettava jatkuvaan muutokseen
  - Jatkuva integrointitestaus (frequent integration, continuous integration)
  - Jatkuva regressiotestaus
  - Testitapausten uudelleenkäytettävyys inkrementtien välillä

Inkrementtien testaus



Testityyppejä

Testityyppejä

- On olemassa perustestityyppejä, joita voidaan käyttää useilla eri testaustasoilla
- Ohjelmiston *testattava ominaisuus on* testityypin määrittävä tekijä – ei kehitysprojektin vaihe

Toiminnallinen testaus (functional testing)

- Toiminnallinen testaus verifioi ohjelmiston tuottamat tulokset ja käyttäytymisen ohjelmiston syötteillä (input-output behavior)
  - Ohjelmistoa tarkastellaan ”mustana laatikkona” (black box) eli tarkastellaan vain sen ulkoisesti havaittavaa käyttäytymistä, ei sen sisäistä tilaa
  - Testien lähtökohtana ovat ohjelmiston toiminnalliset vaatimukset, jotka määritelevät *mitä* ohjelmisto *tekee*
- Tämä testaustyyppi sopii parhaiten järjestelmä- ja hyväksyntätestaukseen

Kirjattuihin vaatimuksiin perustuva t:nen testaus

- Jos toimintoja koskevat vaatimukset on kirjattu ylös, testitapaukset suunnitellaan niiden perusteella
  - Jokaista vaatimusta kohden tehdään vähintään yksi testitapaus
  - Usein vaatimukseen sisältyy (eksplisiittisesti tai implisiittisesti) useita mahdollisia tapoja tehdä vaadittu asia (ja poikkeustilanteita), joten testitapauksia tarvitaan yleensä useita

Kirjattuihin vaatimuksiin perustuva t:nen testaus

- Toiminnallisista vaatimuksista johdetaan usein käyttötapauksia (use case), jotka toimivat käyttöliittymä- ja myös ohjelmistosuunnittelun lähtökohtina
  - Käyttötapaus kuvaa täydellisen vuorovaikutusjakson (skenaarion) ohjelmiston ja ulkoisen *aktorin* välillä siten, että jokin aktorin tavoite (syy ohjelmiston käytölle) tulee täytetyksi skenaarion aikana
- Käyttötapaukset ovat hyvä pohja testitapausten suunnittelulle

Liiketoimintaprosesseihin perustuva t:nen testaus

- Yksittäisiä käyttötapauksia laajempia kuvauksia ovat *liiketoimintaprosessien* (business process) ja *-tapahtumien* kuvaukset
- Liiketoimintatapahtuma kokoaa yhteen useita käyttötapauksia loogiseksi toimintojen ketjuksi, joka täyttää jonkin asiakkaan liiketoiminnan tavoitteen
- Tapahtuman testaus vaatii oman *testiskenaarionsa*

Liiketoimintaprosesseihin perustuva t:nen testaus

- Esim ”auton myynti” liiketoimintatapahtuma koostuu käyttötapauksista
  1. ”autotarjonnan selaus”,
  2. ”auton ja lisävarusteiden valinta”,
  3. [”rahoitustarjouksen pyyntö”],
  4. [”vakuutustarjouksen pyyntö”],
  5. ”hankintasopimuksen teko ja tallennus”
  6. ”auton ja lisävarusteiden tilauksen toimitus maahantuojalle”

Ei-toiminnallisten (non-functional) ominaisuuksien testaus

- *Ei-toiminnallisilla ominaisuuksilla* tarkoitetaan järjestelmän toimintaa tai sitä kokonaisuutena määrittäviä laatupiirteitä
  - *Miten hyvin* järjestelmä *tekee* sen minkä se tekee
- ISO 9126 laatustandardin ei-toiminnalliset piirteet
  - *Reliability*, *Usability*, *Efficiency*
- Laatupiirteet *Maintainability* ja *Portability* liittyvät lähinnä ohjelmiston rakenteeseen
  - Näilläkin on epäsuorasti vaikutusta asiakkaan tyytyväisyyteen!

Konkreettisia testauskohteita

- Käytös kuormituksen (load test) alla
- Suorituskyky tärkeissä käyttötapauksissa ja kuorman alla
- Suurten datamäärien käsittely (volume test)
- Ylikuormitustilanteet (stress test)
- Turvauhkien käsittely ja torjunta (security)
- Vakaus ja luotettavuus jatkuvassa käytössä (stability)
- Käytös virhetilanteissa (robustness)
- Yhteensopivuus, datamuunnokset (compatibility)
- Eri laitteisto- ja ohjelmistokonfiguraatiot
- Käytettävyys
- Dokumentointi
- Ylläpidettävyys

Ei-toiminnallisen testauksen haasteita

- Ei-toiminnalliset vaatimukset ovat usein niin epämääräisesti muotoiltu, että konkreettisia testitapauksia ei voi määritellä
  - ”Ohjelmiston pitää olla helppokäyttöinen”
  - ”Järjestelmän pitää olla nopea”
- Myös ei-toiminnallisten vaatimusten pitää olla ilmaistu testattavassa muodossa!

Ei-toiminnalliset testitapaukset??

- Valitaan toiminnallisista testeistä skenaario, joka edustaa poikkileikkausta koko järjestelmän toiminnasta (esim. käyttöliittymästä tietokantapalvelimeen asti)
- Testattavan ei-toiminnallisen ominaisuuden tulee olla *havainnoitavissa* ja *mitattavissa* skenaarion suorituksen aikana
- Jos mittauksen tulos on sallituissa rajoissa tavoitearvoon verrattuna, testin tulos on ”pass” ja muuten ”fail”

Ohjelmiston rakenteen testaus

- Testitapausten suunnittelussa käytetään hyväksi tietoa ohjelmiston arkkitehtuurista ja sisäisistä tiloista (esim. mallien avulla)
- Tarkoitus on suunnitella ja ajaa riittävästi testitapauksia kaikkien rakenteiden testaamiseksi

Ohjelmiston muutosten testaus ja regressiotestaus

- Uudelleentestaus (retesting) muutosten jälkeen
  - Toistetaan aikaisemmat testit sen osoittamiseksi, että korjatut viat ovat todella hävinneet
- Regressiotestaus muutosten jälkeen
  - Tarkoituksena on osoittaa, että muutokset eivät ole aiheuttaneet *uusia vikoja* (tai vanhojen vikojen uusiutumista) ohjelmistoon

Ohjelmiston muutosten testaus ja regressiotestaus

- Regressiotestaukseen valitut testitapaukset suoritetaan toistuvasti, joten ne on dokumentoitava hyvin
  - Testit kannattaa mahdollisuuksien mukaan automatisoida

Regressiotestauksen laajuus

- Tärkeää on päättää regressiotestauksen laajuus
  - Suoritetaan uudestaan kaikki testit, jotka löysivät jonkin nyt korjatuista vioista
  - Testataan kokonaan uudelleen kaikki ne ohjelman osta joita on nyt muutettu
  - Testataan kaikki vastikään integroidut ohjelmiston osat
  - Testataan uudelleen koko ohjelmisto

Regressiotestauksen laajuus

- Ohjelmistomuutoksilla voi olla arvaamattomia sivuvaikutuksia myös niissä ohjelman osissa, joihin ei ole koskettu
  - Osien välillä voi olla suuri määrä suoria ja epäsuoria riippuvuuksia – niitä on helppo saada aikaan mutta työlästä poistaa
  - Syynä usein huonosti suunniteltu tai muuten sopimaton arkkitehtuuri ja piittaamattomuus riippuvuuksia koskevista säännöistä
- Periaatteessa vain koko ohjelmiston täydellinen uudelleentestaus on riittävän laaja regression havaitsemiseen

Regressiotestauksen laajuus

- Käytännössä koko ohjelmiston uudelleentestaus on kuitenkin liian kallista ja hidasta
- On siis valittava huomattavasti suppeampi joukko testitapauksia, joitten suoritus tuottaa riittävän varmuuden regression poissaolosta
  - On yritettävä päätellä, mihin ohjelmiston osiin sivuvaikutuksia voisi tulla
  - On muutenkin pyrittävä rajoittamaan ajettavien testitapausten määrä minimiin

Regressiotestauksen laajuus

- Regressiotestitapausten valintakriteereitä
  - Vain testisuunnitelman korkeimman prioriteetin testit ajetaan
  - Toiminnallisten testien tietyt variantit voidaan jättää pois
  - Rajoitetaan testaus vain tiettyihin tuotekonfiguraatioihin (vain yksi kieliversio ja yksi käyttöjärjestelmä)
  - Testataan vain tietyt alijärjestelmät tai vain tietyt testitasot

Oppimistavoitteet

- Ohjelmistoversioiden testaus
- Testityyppejä