28.11.2010

Matti Vuori

Näkemyksiä uuden teknologian ja SOA-pohjaisten järjestelmien testauksesta

Lueskelin erästä 600-sivuista SOA-järjestelmien suunnittelusta ja toteutuksesta kertovaa kirjaa. Koska siinäkään ei käsitelty testausta hyvin kursorisesti ja nettihakukaan ei nostanut esille kovin tolkullista aineistoa asiasta, tulin siihen tulokseen, että asiaa ei ole ajateltu riittävästi. Siksi ajattelin pistää paperille omina näkemyksiäni, täydennettynä yleisemmillä uuden teknologian testaukseen liittyvillä näkemyksillä. Toivottavasti niistä on hyötyä muillekin. Teksti ei pyri olemaan täydellinen, vaan tarkoituksenmukaisen kevyesti teemoja luotaava.

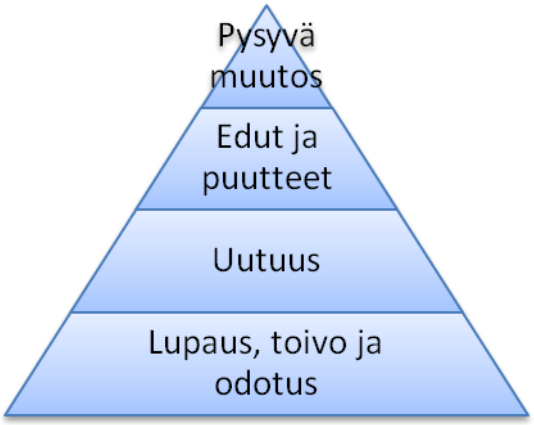
Paperin idea

Tarkoituksena on käsitellä SOA-pohjaisten järjestelmien testausta vertailemalla sitä ”tavallisten” järjestelmien testaukseen, eli sterotyyppisiin kohteisiin, joista alalla on paljon kokemusta ja joihin testausosaaminen kohdistuu. Käsittelemme siis SOA:n mukanaan tuomia muutoksia ajatteluun ja toimintaan, mutta myös sitä, mikä pysyy samana!

Ennen kuin menemme SOA-asioihin pari sanaa uudesta teknologiasta

Ennen varsinaiseen asiaan menemistä on hyvä miettiä, millaisiin asiakokonaisuuksiin pitää kiinnittää huomiota, kun luomme suhdetta uuteen teknologiaan ja sen testaamiseen ja muuhun laadunvarmistukseen. Niitä kokonaisuuksia voi olla esimerkiksi tällaisia:

- **Mikä on muuttunut?** Kun kokemuspiirimme koostuu vanhoista asioista, on hyvä pyrkiä näkemään keskeisiä, konseptitason muutoksia ja niiden vaikutuksia. Onko muutos kenties osin näennäistä – joskus asiat vain saavat uuden nimen.
- **Mikä siis on hypeä**, jonka läpi on nähtävä? Yleensäkin kaikkiin uusiin ilmiöihin liitetään monia ihanteita, joilla ei ole mitään suoraa yhteyttä kyseiseen ilmiöön. Ainoa yhteys on akuuttius *nyt*.
- **Mitä on pysynyt samanlaisena?** Ihmiselle on tärkeää nähdä asioissa pysyvyyttä. Sen päälle voidaan rakentaa muuksia. Tämä asia on siis monimerkityksinen: rakenne, toiminta, psykologia…
- Mitä uusia asioita teknologiassa on, jotka **edellyttävät testausta?** Joskus voi esimerkiksi



järjestelmän viestiliikenne tai asynkronisuus tuoda aivan uudenlaisia piirteitä testaukseen.

- Yleisemmin tämä on kysymys **teknologian riskeistä**. Niistä ei puhuta koskaan yhtä paljoa kuin teknologian eduista. Testaajan on kuitenkin ymmärrettävä ne.
- Tarjoaako teknologia **uusia mahdollisuuksia** testaukseen, esimerkiksi uudenlaisia rajapintoja? Avaako se uudenlaisia ovia, esimerkiksi soveltuu hyvin sellaiseen testiautomaatioon, joka aiemmin eri ole ollut realistista?
- Muuttuuko testattavien **asioiden painotus** jollain tavalla ja jostain syystä? Nouseeko esimerkiksi tietoturvatestaus erilaiseen rooliin kuin muilla teknologioilla?
- Millaisia **kulttuuriin, suhtautumistapoihin ja asenteisiin** liittyviä vaikutuksia teknologialla on? Niille pitää olla tarkkana, ettemme ole esimerkiksi hypen vietävinä!

- Mitä kaikkea **oppimista** meillä on? Millä tavalla pitää uudistaa ajattelua, tietopohjaa, menetelmiä ja käytäntöjä?
- Testaus ei ole umpio. Mitä muutoksia syntyy tai pitää luoda **muuhun laadunvarmistukseen** tai kehittämisen prosessimalleihin?
- Jne… tässä paperissa ei käsitellä ihan kaikkia näitä asioita, mutta lukijan on hyvä pohtia niitä, varsinkin, jos hänellä on käsillä konkreettinen SOA-järjestelmä.

Mikä ihmeen SOA?

SOA = Service Oriented Architecture, palvelukeskeinen arkkitehtuuri. Ks. Wikipedia-artikkelit:

- http://en.wikipedia.org/wiki/Service-oriented_architecture
- http://fi.wikipedia.org/wiki/Palvelukeskeinen_arkkitehtuuri

SOA:ssa on erittäin hienoa, että se tuo tervettä kokonaisvaltaisuutta organisaation järjestelmien palettiin, lisää harmonista selkeyttä, avaa rajapintoja, parantaa siten integroitavuutta ja poistaa päällekäisyyksiä.

Onko SOA sitten uutta teknologiaa? On – vaikka sitä on kehitelty jo vuosikausia, se on vierasta, siitä pitää puhua erikseen ja se ei ole muokannut ajattelumallejamme. Teknologian tuleminen vanhaksi ja tutuksi vie aikaa yllättävän kauan.

Tärkeimmät muutokset

Aikaperspektiivi: SOA-järjestelmiä tehdään tulevaisuutta varten. Testaustakaan ei silloin tehdä vain lähiajan tarpeisiin, vaan koko tiedossa olevaa elinkaarta varten.

Rakenne: Järjestelmä on järjestelmien järjestelmä. Järjestelmätestaukseen tulee järjestelmäintegrointitestauksen piirteitä. Samalla jokaisen yksittäisen palvelun pitää olla itsenäinen testauksen kohde, eikä vain kokonaisuuden "komponentti".

Riskit: Hajautus palveluihin ja metatieto palveluista muuttaa riskien, erityisesti tietoriskien, maailmaa.

Käyttäjäkokemus: Käyttäjät eivät voisi vähempää välittää, onko järjestelmä SOA-pohjainen tai mikä tahansa. Siksi käyttäjänäkökulmasta tehty testaus selvittää samoja asioita kuin ennenkin käyttäen samanlaisia lähestymistapoja.

Käyttäytyminen: Mahdollinen asynkronisuus voi tuoda aikakäyttäytymiseen muutoksia verrattuna monoliittiseen synkroniseen järjestelmään.

Järjestelmän muuttaminen: Koska järjestelmän elementit ovat rajapintojen takana, yksittäisen palvelun muuttaminen on turvallista ja regressioriski on pienempi kuin monoliittisilla järjestelmillä. Tämä helpottaa regressiotestauksen suunnittelua ja tehostaa sen suoritusta. Näyttää siltä, että SOA myös ehkäisee arkkitehtuurin rapautumista vuosien ja vuosikymmenten varrella, mikä auttaa pitämään testaukselle otollisen tilanteen sellaisena jatkossakin!

Samaa kuin ennenkin

Tällaiset testaukset ovat luonteeltaan hyvin samanlaisia kuin ennnekin:

- Liiketoimintaprosessien end-to-end -testaus.
- Eräajotehtävien testaus.
- Käytettävyystestaus ja käyttökokemuksen testaus.
- Liiketoimintaprosessien suorituskykytestaus.
- Yksikkötestaus palvelun toteuttavissa luokissa.

Yksittäisten palveluiden testauksesta

Keskeinen ajatus on palvelun yleiskäyttöisyys – ei palvella enää vain yhtä järjestelmää tai liiketoimintaprosessia.

- SOA:ssa ei ole järkeä, jos palvelun elinkaari on lyhyt. Siksi katse tulevaisuuteen – joka on tuntematon palvelun tulevien käyttäjien osalta.

Lähtökohta on se, että yksittäistä palvelua voi kutsua "kuka tahansa" (kunhan siihen on annettu lupa).

- Laajempi testaus kuin vain yhden liiketoimintaprosessin tarpeisiin.
- Vaaditaan korkeaa robustiutta.
- Negatiivinen testaus tärkeää.
- Fuzzing on myös hyvä idea (datan ja sanomien satunnaistaminen ja rikkominen).
- Koska palvelut on kuvattu systemaattisella tavalla, on mallipohjaisella testauksella hyvä potentiaali.
- Kontekstin vaihto – eri kutsuja – testattava. Tällaisiin järjestelmiin jää helposti muistiin kutsujaan liittyvää tietoa, koska ne kehitetään yhdelle tiedossa olevalle liiketoimintaprosessille.

Palvelut elävät kokonaisarkkitehtuurin "ekosys-teemissä". Niillä on oltava "käyttäytymissäännöt" ja niiden noudattamisen varmistus on testauksenkin tehtävä:

- Suorituskyky
- Muistin tarve
- Tietoliikenteen volyymi
- Rajapintakäytännöt
- Tietoturvakäytännöt
- Dokumentointitavat
- Jne...

Palvelun ydintä on toisaalta palvelukohtainen palvelusopimus, jonka sisällöt on testattava – onnistuuko testaus sallituilla tavoilla, onko muut tavat estetty.

- Ketkä saavat käyttää.
- Millaisilla sanomilla.
- Käytön ehdot.

Adapterit, joilla varsinkin vanhoja järjestelmiä sovitetaan mukaan. Mitä tapahtuu, jos palvelu ei toimikaan? Poikeusten hallinta on kriittisintä silloin, jos tarvitaan sovitusta erityyppisten komponenttien välillä. Historiallisesti häiriöitä ei silloin aina hallita riittävän hyvin. Entä tietoturva? Pisteet, joissa varmistetaan, että palvelua käyttävä on oikealla asialla, voivat siirtyä ja eheys on vaarassa. Nämä ovat testauksessa keskeisiä huomioonotettavia asioita.

Suuri luottamuksen ristiriita!

Palveluja pitäisi kehittää tulevaisuuden tarpeisiin, mutta jos niitä kehitetään yksittäisten projektien yhteydessä, voi olla varma, että vaikka yleiskäyttöisyydestä maksettaisiin, se ei ole lainkaan varmalla pohjalla. Siksipä kaikkiin palveluihin pitää suhtautua epäillen järjestelmää laajennettaessa. Järjestelmätestaus ottaa asian toki ”luonnostaan” pitkälti huomioon.

Osaaminen

Koska SOA:ssa on uusia piirteitä, pitää tarkastella myös kehittävän tiimin osaamista. Jos tiimille ovat SOA-ajattelu, teknologia ja alustat uusia, pitää varautua kyseisen tiimin tuotosten testaamiseen erityisellä ”avoimuudella”.

Päivänselvää on, että testaajia pitää perehdyttää uuteen arkkitehtuurityyliin ja sen vaikutuksiin.

Yhden liiketoimintaprosessin toteuttava kokonaisuus

- Jos asynkroninen, on poikkeusten käsittely tärkeää – miten ne hallitaan jokaisessa vaiheessa ja prosessi keskeytetään tai perutaan hallitusti.
- Viiveet voivat vaihdella enemmän kuin monoliittisissä järjestelmissä. Siksi on testattava timeoutien toiminta (ja järkevyys).
- Testausta pitää edeltää tarkastus, että kaikki palvelut ymmärtävät sanomat samalla tavalla – tarkoitus, rakenne, muuttujat, tietotyypit jne…

Kokonaisarkkitehtuuri

Kokonaisarkkitehtuurin – tai yritysarkkitehtuurin – valinta on tietenkin erittäin keskeinen päätös tulevaisuuden kannalta. Arkkitehtuurin arviointi on aina tärkeä askel, vaikka käytettäisiinkin jonkin toimittajan perusratkaisuja, puhumattakaan siitä, jos kokonaisuus kootaan eri toimittajien komponenteista (vaikkapa avoimen lähdekoodin SOA-ohjelmia sovittaen).

Tietoturvallisuuden analysointi ja testaus on tärkeää kokonaisuuden tasolla:

- Palveluhakemiston suojaus – esimerkiksi palvelun vaihtomahdollisuus.
- Sisäiset hyökkäykset entistä houkuttelevampia.

Palveluihin liittyvän metadatan oikeellisuus on osin tarkastettava eikä testaava asia.

Testattavuudesta

Testattavuuskatselmus on hyvin tärkeä osa järjestelmäprojekteja.

Koska testattavaa on paljon, pitää varmistaa, että arkkitehtuuri pysyy eheänä ja kaikkia palveluita voidaan testata samoilla tavoilla.

Palveluhakemisto on hieno keksintö. Koska sen on pakko olla ajantasalla, se tarjoaa ajantasaista tietoa myös testaajille!

On tunnettua, että esimerkiksi XML-kuvauksia voidaan tehdä monella tavalla ja tietokoneohjelmille monet tavat voivatkin olla samanarvoisia. Kuitenkin, tiedostoja ja sanomia pitää usein kyetä lukemaan ihmisissilmilläkin tai käsittelemään rajoittuneilla välineillä, joten ”konepellin allekin” on hyvä luoda sääntöjä.

Tunnettuihin tai standardoituihin tekniikoihin perustuvat rajapinnat – ja jokaisen palvelun julkistaminen niiden kautta – on erinomainen asia testattavuudelle. Tarve järjestelmän osa-alueita varten rakennettaville testausrajapinnoille voi vähentyä. Palveluhakemiston kautta on simulaattorien yms. liittäminen koodiin koskematta periaatteessa (!) yksinkertaista – käytäntö riippuu aina kustakin SOA-alustasta.

Jokainen palvelu hyötyy omasta testausohjelmasta, jolla sille voi lähettää sanomia ja tarkistaa tuloksen.

SOA:ssa on sellaisia piirteitä, jotka suosivat avoimen lähdekoodin järjestelmiä ja niiden avoimuuden tuomaa autitoitavuutta ja testattavuutta: suurempi robustiusvaatimus, yleiskäyttöisyys ja tulevaisuusnäkökulma.

Kulttuurin ja käytäntöjen luominen

Ensimmäinen SOA-pilotti on kriittinen paikka testauskäytäntöjen luomiselle. Ideana on luoda stabiili malli kaikkien tulevien palvelujen toteutukselle! Testattavuuden pitää olla silloin äärimmäisen vahvasti esillä.

SOA edustaa tietynlaista tasapäistävää mekanisaatiota, mikä on testaukselle aina vaarallista. Tämä asia hallitaan tiedostamalla se.

SOA:ssa on muutamia uusia asioita, jotka edellyttävät tiukkaa yhteistyötä testauksen ja kehittäjien välillä.

Ylipäätään SOA:ssa on – nimensä mukaisesti – kyse arkkitehtuurista ja SOA-kulttuuri on siksi arkkitehtien kulttuuria. Muiden – kuten vaikka laatuihmisten ja testaajien –- voi tämän vuoksi olla vaikea saada siihen otetta. Silloin kannattaa etsiä teknologisen sälän joukosta jotain tuttua ja pysyvää:

- SOA:n tärkein tavoite on kuvata liiketoimintaprosesseja. Vaikka niiden tietotekninen toteutus olisikin vaikeasti hahmotettava ketju järjestelmän syövereissä, ne kulminoituvat aina joihinkin pisteisiin, joissa jotain on saatu aikaan tai alkaa jotain uudenlaista. Putket tällaisten pisteiden välillä ovat kaikkein tärkein asia tunnistaa ja testata.
- Kaikenkaikkiaan liiketoiminta ja ihmisten toiminta ovat asiat, joita vartean tekniikkaa rakennetaan. Ne ovat myös näkökulma, josta pitää aloittaa laadunvarmistuksen miettiminen.
- Kun asiat muuttuvat, voi olla hyvä tehdä oman ajattelun inventaariota ja vaikkapa tarkistaa, miten laatumallistandardi ISO 9126:n sisällöt suhtautuvat uuteen tilanteeseen – nouseeko siellä esille jotain uutta: asioita ja niiden mittareita, jotka olisi nyt mahdollista ja hyvä ottaa mukaan testaukseen.

Yhteenveto: parhaat hyvät uutiset

Hyvä uutinen on se, että SOA-järjestelmiä voidaan testata **ihan kuin mitä tahansa järjestelmiä**. Joskus voi ajatella: arkkitehtuuri kuin arkkitehtuuri! Yhtä hyvä uutinen on se, että SOA:n tekniset piirteet ovat hyödyllisiä testattavuuden ja testaamisen kannalta ja vain lisäävät **testauksen mahdollisuuksia**, mutta eivät pakota mihinkään uuteen ja vaikeaan. Vaikeaa on oikeastaan sopivan lähestymistavan löytäminen tulevaisuuteen ja palvelujen yleiskäyttöisyyteen. Miten paljon siihen kannattaa panostaa ja miten sen testaamiseen kannattaa kulloinkin uhrata aikaa ja rahaa? Vastaus tähän riippuu paljon suhdanteesta ja toimialasta.