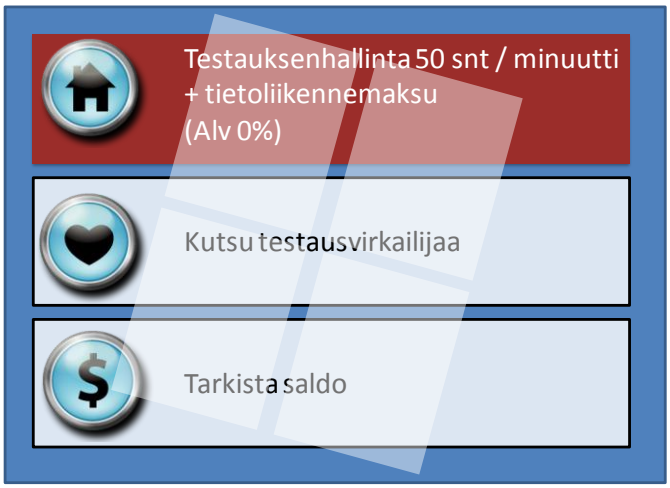
Matti Vuori

50 pointtia testauksenhallintajärjestelmän hankkimiseen palveluna

Järjestelmien hankinta palveluna on 2010-luvulla saanut vihdoin tuulta purjeisiin.Sen sovellukset lisääntyvät varmasti myös tuotekehityksessä ja ohjelmistotuotannossa. Tässä paperissa käsitellään näkökulmia kysymykseen kannattaisiko testauksenhallintaympäristö (ohjelmisto, tietokanta, ylläpito, käyttäjätuki, varmuuskopioinnit yms.) ostaa palveluna sen sijaan, että se hostattaisiin nykyisin yleisimpään tapaan itse organisaation IT-osastolla. Asiaan on organisaation eri osastoilla erilaisia suhteita, tässä tarkastelussa on pääosin tuotekehityksen / ohjelmistotuotannon näkökulma.

Raha ratkaisee

1. Kannattaako palveluratkaisu vai ei? Siihen ei ole kategorista vastausta, kuten ei siihenkään, että siirtyisikö joku yritys palveluratkaisuun vai ei. **Kaikki riippuu olosuhteista, casesta, vaatimuksesta ja kulloisestakin muodista.**
2. Raha ja sen säästäminen on usein niin tärkeää, että **jos palvelu-ratkaisu on halvempi, ”anything goes”**.
3. Harvoin kuitenkaan osataan laskea edes suuntaa-antavasti kaikkia tietojärjestelmän suoria ja epäsuoria kokonaiskustannuksia. Tietokoneiden ja lisenssien laskenta on helppoa.



Palveluissa mahdollisuuksia

4. Vanha sananlasku: ”Näyttäkää minulle testauspäällikkö, joka on tyytyväinen organisaationsa tietojärjestelmien palveluihin, niin minä näytän teille henkilön, joka olisi hyödyllisempi suorittavissa tehtävissä”. Tai sitten harvinaisen hyvän organisaation!
5. Sisäisissä palveluissa on näet parantamista – palvelu on usein kankeaa, byrokraattista ja hidasta – ja parannus onkin yksi asia, jota muutoksilla pitäisi hakea. Tämä ei koske suinkaan vain testauksenhallintaa, vaan kaikkia ohjelmistotuotannon ja projektitoiminnan järjestelmiä. Palvelumalli voi olla kipeästi kaivattu apu – tai sitten ei – paljon riippuu palveluntarjoajasta.
6. Yksi ongelma on se, että – kuten tulemme huomaamaan – jo riittävän hyvä palvelu on väistämättä kallista, erityisen hyvästä palvelusta puhumattakaan.
7. Eli jos palvelun hintalappu on kovin halpa, kannattaa selvittää huolella miksi näin on, ettei tule yllätyksiä.
8. **Tuotekehityksen näkökulmasta ”ostettukin” järjestelmä on jo hankittu palveluna omalta ICT-osastolta!** Käytön kannalta ei ole mitään eroa, käytetäänkö ohjelmia nettiselaimella omasta vai toisen toimijan verkosta. Ylläpito on vähänkin suuremmassa organisaatiossa samalla tavalla kasvotonta – korkeintaan nimiä palvelutiketeissä.
9. **Tietoturvallisuus** on kriittinen asia ja se ei saa heikentyä, vaan useimmiten sen on syytä parantua.
10. Järjestelmän pitää täyttää omat, omien sertifikaattien, päämiesten, asiakkaiden ja kumppanien vaatimukset.
11. Hostauksen palvelukyky voi olla parempi kuin omalla organisaatiolla, koska rahaa vastaan on helppo vaatia asioita. Sopimus on aina ”katkolla” toisin kuin omassa organisaatiossa, jossa ei ole painostuskeinoja – vaan päinvastoin, hostaaja voi esittää vaatimuksiaan palveltaville yksiköille!

12. On muistettava sopia tarkasti sellainen palvelun laatu, mitä tarvitaan. Ongelmanratkaisun vasteajat yms. Unohtamatta suorituskykyä: palvelusta menee maku, jos vasteajat ovat kovin pitkiä.

Palveluntajoaja eli hostausyritys

13. Kuten missä tahansa bisneskriittisessä asiassa kumppani ei saa olla kuka tahansa, vaan sen pitää olla luotettava, stabiili, ei nyrkkipaja. Ideana on varmistaa se, että kumppani ei kaadu alta ja että sillä on riittävästi hyviä ihmisiä takaamaan palvelun laatu ja joustavuus.
14. Tällaisenkin erityisjärjestelmän hostaajalla pitää olla **eturivin hostausyrityksen kaikki kyvykkyydet**. Se ei siis riitä, että yritys on esimerkiksi itse kehittänyt palvelussa käytettävän ohjelman.
15. **Tietoturvallisuuden hallintajärjestelmä** välttämätön. Hyvä standardi sellaiselle on ISO/IEC 27000, koska se kattaa järjestelmän toimintamallit ja ottaa kantaa suojaustapoihin.
16. Tietoriskien kannalta tuotekehityksen järjestelmillä on se erityisasema, että ne käsittelevät joskus ulkopuolisten kannalta seksikästä teknologiaa. Jos perinteisen mallin suurimmat uhat tulevat ”sisäpiiristä”, nyt avautuu yksi uusi sisäpiiri lisää, joka on lähellä kaikkien projektien tietoja.
17. Järjestelmistä riippumatta tilaturvallisuus on ensimmäinen tarkistettava asia. Käynti hostausyrityksessä ja konesalissa (niin… pääsetkö konesaliin…) tai vastaavassa kertoo hyvin paljon tulevan palvelun laadusta.
18. **Laadunhallintajärjestelmä** välttämätön. ISO 9001 on tälläkin alueella ihan hyvä.
19. **Palveluprosessit** ja niiden menettelyt kunnossa – esim. ISO 20 000 tai ITIL pohjana. Tuotekehityksen kumppanille sopisi toisaalta myös uusi CMMi for Services, koska se on niin läheinen ohjelmistokehityksen CMMi:n kanssa.
20. **Sertifiointien tai puolueettomien assesmointien puuttuessa hostaaja kannattaa itse auditoida – niinhän tehdään usein muillekin ohjelmistokehityksen alihankkijoille.**
21. Kokemusta hostaamisessa ei tietenkään voita mitään.
22. Hostaajalla pitää olla jatkuvuussuunnitelmat ongelmien varalta.
23. Hostaajalla pitää olla toiminnastaan riskianalyysi.

Järjestelmä

24. Tuotekehityksen asiakkaiden / päämiesten data saattaa olla pakollista erottaa fyysisesti eri palvelimiin (tai ainakin virtuaalisesti eri virtuaalipalvelimiin) sekä sovelluspalvelimessa että tietokantapalvelimessa. Tämän onnistuminen sujuvasti on selvitettävä.
25. Käyttäjänhallinta kannattaa olla pääosin itsepalveluna. Projektipäällikön pitää pystyä määrittämään oman projektinsa käyttäjät ilman ulkopuolista apua.
26. Miten järjestelmään saadaan organisaatiosta henkilöiden perustiedot ja samojen tunnusten käyttö?
27. Miten ohjelmisto toimii hajautettuna – jos kaikkien maiden tietoja ei haluta laittaa samaan tietokantaan / käyttää samalla sovelluspalvelimella?
28. Testauksenhallinta ei toimi yksin. Se pitää integroida useaan muuhun järjestelmään (vaatimustenhallinta, vikakanta, henkilöhakemisto jne…). Miten tämä kaikki onnistuu turvallisesti ja tehokkaasti?
29. Nykyajan palvelujen piirre on räätälöinti ja konfigurointi lisäosapaketeilla. Miten helposti tämä onnistuu? Testauksenhallinnassa tarvitaan vaikkapa työnkulkujen ja käsitteiden räätälöintiä ja lisäpalikoiden tarve vaihtelee yksiköittäin ja projekteittain.

Jatkuvuus

30. Pitää olla **jatkuvuussuunnitelma** isojen ongelmien varalta. Mitä, jos hostajan rakennus palaa tai tulva vie palvelimet tai kaapelit katkeavat?
31. **Varmuuskopiot ja varapalvelimet**. Toinen palvelin pitää saada hetkssä käyttöön ja varmuuskopioilta data sisään. Onneksi palvelimet ovat usein virtuaalisia, jolloin palvelin syntyy uudelle koneelle konkreettisesti käden käänteessä.
32. Millainen on ohjelmiston saatavuus, jos päätetään **vaihtaa palveluntarjoajaa**. 2000-luvun trendi on ollut välttää riippuvuuksia toimittajista. Kaikkia lukkoja pitää varoa.
33. Palveluntarjoajat tietysti haluavat lukita asiakkaansa niin tiukasti kuin mahdollista. Kun mm. avoin lähdekoodi yms. tarjoaa uudenlaista vapautta, pitää kehittää uusia keinoja.

34. **Datan saatavuus ja käytettävyys, jos palvelusta luovutaan**. Miten tietokannan sisältöä kyetään hyödyntämään? Saadaanko se konvertoitua toisen järjestelmän muotoon? Ohjelmien tietokannoille ei ole mitään standardia, paitsi relaatiomallin mukainen looginen tietomalli, mikä ei auta vielä pitkällekään. Tämä ongelma on sinänsä ihan sama ”ostetuillakin” ohjelmilla, mutta omille palvelimille asennettuna niiden tietokantaa pystyy paremmin tutkimaan ja suunnittelemaan konversioita.

Siirtymisprosessi – jos siirrytään

35. Tietenkin täytyy selvittää kaikki testauksenhallinnan vaatimukset ja tsekata, miten uudistus vaikuttaa niihin.
36. Samalla joudutaan mahdollisesti vaihtamaan ohjelmisto toiseksi, mikä tuottaa lisää ongelmia muutostilanteeseen.
37. Vaatimuksia ei ole pelkästään itsellä, vaan myös verkostolla, **alihankkijoilla, päämiehellä** jne…
38. Kannattaa myös miettiä, miten muut analogiset prosessit, joissa tarvitaan työnkulkuja, toimivat. Kenties niitä saisi yhdistettyä samaan työkaluun? Esimerkiksi ketterässä kulttuurissa suhtaudutaan monenlaiseen tekemiseen samanlaisella modulaarisella tavalla, mikä on hyvin miellyttävä asia.
39. On hyvä muistaa, että jos omaa testauksenhallintaa on pyöritetty jo pitkään, sen ominaisuudet ovat niin itsestäänselviä, että niitä ei enää huomaakaan! Vaaditaan tarkkuutta, että ne osataan tunnistaa ja ottaa huomioon palvelun vaatimuksissa.
40. Mutta joka tapauksessa palvelua pitää ensin **testata ja kokeilla** jossain projektissa ennen toiminnan siirtämistä sinne.
41. Varsinkin, jos kyseessä on ensimmäinen ohjelmistotuotannon järjestelmän ulkoistus, ulkoistukselle kannattaa tehdä formaali **riskianalyysi**. Tietoriskit pitää analysoida tiukasti.
42. Riskianalyysiä pitää joissakin tapauksissa kuunnella tarkemmin kuin talouspäällikön laskelmaa.
43. Ulkoistetun järjestelmän arkkitehtuuria ei ole helppo muuttaa, joten sen lähtökohdaksi kannattaa tunnistaa kaikki **lähivuosien muutokset** toiminnassa, organisaatiossa, asiakkaiden ja päämiesten toiminnassa ja organisaatiossa. Tietenkään näkyvyys tulevaisuuteen ei aina ole pitkä, mutta otetaan huomioon se, mitä kyetään.
44. Joka tapauksessa 2000-luvun järjestelmien pitää olla sellaisia, että ne sopeutuvat **organisaatiomuutoksiin ja yrityskauppoihin** helposti.
45. Mutta palvelumalli ei ole ainoa reitti! ”**Downgreidaus**” on toinen strategia: parannetaan järjestelmien toimintaa ottamalla ne pois IT:ltä ja antamalla projektien huolehtia järjestelmistään. Pienuudessa on joskus tehokkuus-, kustannus- ja järkevyysoptimi! Mutta ei toki aina.
46. Päätös SaaS:iin siirtymisestä pitää tietenkin jättää palvelua käyttävälle yksikölle.

Eräät sivuvaikutukset on hyväksyttävä

47. Omalla koneella olevan ohjelman tietokantaan on mahdollista päästä tekemään **bulkkimuutoksia** tai tekemään tietomassasta **erityisiä analyysejä**, joihin testauksenhallinta-ohjelma ei taivu. Mutta kun tietokanta on palveluntarjoajalla, se ei onnistukaan. Mutta palveluja saa toki rahalla, jos palveluntarjoaja on palvelumyönteinen. (Hyvällä onnella ohjelman skriptikieli tekee kaiken tarvittavan.)
48. Samoin avoimen lähdekoodin ohjelman **räätälöinti** on asia, jonka saa palvelumallissa samassa määrin unohtaa.
49. Ylipäätään, kaikki oppivalle organisaatiolle oleelliset adhoc-kokeilut voivat vähetyä, jos palvelu ei toimi riittävän sujuvasti.

Extra-pointit!

50. Mieti, mitä geneerisesti oleellista on jäänyt mainitsematta! Mieti, mitä sinun kontekstissasi oleellista on jäänyt mainitsematta!

Termihuomio: Edellä on puhuttu geneerisesti vain järjestelmän hankinnasta palveluna. Tavallinen ohjelmiston ajaminen palvelun-tarjoajan koneelta olisi ”jo perinteinen” Software as a Service (SaaS) -ratkaisu, mutta kun sitä laajennetaan kokonaisratkaisuihin sekä asiakaskohtaisessa laitteistossa, muussa infrastruktuurissa ja erilaisissa palveluissa, ollaankin jo ”pilvessä”.