LAPPEENRANNAN TEKNILLINEN YLIOPISTO

Teknistaloudellinen tiedekunta

Tietotekniikan osasto

Testaushallintapalvelu SaaS-mallissa

Diplomityö

Tarkastajat: Professori Jari Porras

DI Risto Paavola

Helsingissä 23. toukokuuta 2010

Raine Brisk

TIIVISTELMÄ

LAPPEENRANNAN TEKNILLINEN YLIOPISTO
Teknistaloudellinen tiedekunta
Tietotekniikan osasto
Raine Brisk

Testaushallintapalvelu SaaS-mallissa
Diplomityö
2010
57 sivua, 4 kuvaa, 4 taulukkoa ja 0 liitettä
Tarkastajat: Professori Jari Porras
DI Risto Paavola

Avainsanat: SaaS, Software as a Service, Avoin lähdekoodi, Testaushallinta

Testaushallinta on ohjelmiston laadunvarmistusprosessin oleellinen osa, joka tarvitsee onnistuakseen työkalun. Testaushallintaohjelmiston tarjoaminen SaaS-palveluna luo mahdollisuuden tarjota tämän työkalun helposti ja kustannustehokkaasti, sekä valmiiksi määritellyin prosessein.

Tässä työssä tutkitaan testaushallintaohjelmiston SaaS-palveluna tarjoamisen mahdollisuuksia ja rajoitteita pienen projektin näkökulmasta. SaaS-Palvelumallia tutkitaan osiensa muodostamana kokonaisuutena ja selvitetään mallin soveltumista testaushallintapalvelun tuottamiseen. Lisäksi tutkitaan tapaustutkimuksena kyselyn ja haastattelun avulla käyttäjien kokemuksia SaaS-palveluna toteutetun testaushallintaohjelmiston käyttämisestä.

Tutkimuksen tulokset viittaavat siihen, ettei SaaS-malli luo erityisiä rajoitteita testaushallintaohjelmiston tarjoamiseen laadunvarmistusprojektin käyttöön ainakaan tutkitussa mittakaavassa. Palvelun toimintamallien suunnitteluun ja vastuiden jakoon on kiinnitettävä erityistä huomiota, jotta palvelu voidaan toimittaa loppukäyttäjien tarpeita mahdollisimman paljon huomioiden.

ABSTRACT

LAPPEENRANTA UNIVERSITY OF TECHNOLOGY
Lappeenranta University of Technology
Faculty of Technology Management
Department of Information Technology
BRISK, RAINE

Test management service in SaaS-model
Thesis for the Degree of Master of Science in Technology
2010
57 pages, 4 pictures, 4 tables and 0 appendices
Examiners: Professor Jari Porras
M. Sc. Risto Paavola

Keywords: SaaS, Software as a Service, Open Source Software, Test Management

Test Management is important part of software quality assurance processes, which needs a tool to be successful. Test management software offered as a SaaS creates the opportunity to offer this tool to easily and cost effectively as well as with pre-defined processes.

This paper examines the opportunities and constraints for providing the test management software as a SaaS from small project's perspective. The service model is examined as the sum of its components and explains the model compatibility to produce test management services. In addition, as a case study survey and an interview of the user experiences in the use of the test management software service is carried out.

The results of the study indicate that the SaaS model does not create any specific constraints on the provided test management software for the quality assurance project, at least in the studied scale. Service's process planning and allocation of responsibilities practices should be considered with high attention, so that the service can be provided to end-users while taking their needs into account as much as possible.

ALKUSANAT

Tähän diplomityöhön kiteytyy vuosien opiskelut ja kokemukset. Tätä työtä on tehty välillä huolellisella hartaudella taustoja tutkien ja perehtyen syvällisesti aiheeseen, sanojen hakiessa huolella paikkaansa tämän työn riveiltä. Välillä taas edistystä on tullut huimin pyrähdyksin sanojen löytäessä riveiltä paikkansa kuin itsekseen. Nyt tämä on kuitenkin tässä ja niin kuin piste päättää lauseen, päättää tämä diplomityö pitkähköksi venähtäneen vaiheen elämässäni antaen mahdollisuuden katsoa eteenpäin nähden siellä kenties jotain uutta.

Haluan tässä suuresti ja erityisesti kiittää isääni Jarmoa, joka jaksoi sitkeästi koko opiskeluni ajan tukea ja muistuttaa asioiden todellisesta tärkeysjärjestyksestä. Nyt tuo tärkein on hoidettu. Kiitän myös tyttöystävääni Lauraa tämän työn oikoluvusta ja opintojeni aikana osoittamastaan tuesta opintojeni saattamiseksi kohti loppua.

Haluan myös kiittää kaikkia opiskelutovereitani, sekä erityisesti Piimäkassi ry:n jäseniä sen asian korostamisesta, että opiskeluaika on myös elettävää elämää. Yhteinen aika Lappeenrannassa ei unohdu. Erityiset kiitokset Antti Soloselle, jonka merkittävä apu kriittisillä metreillä auttoi ratkaisevasti koko opiskeluprojektin loppuun saattamisessa.

Erityisen suuret kiitokset ansaitsee työn toisena ohjaajana toiminut Risto Paavola. Hän koki työni ohjaamisen omakseen työnantajan vaihtumisesta huolimatta. Kiitokset myös työn tarkastajalle professori Jari Portaalle kattavista ja tarkentavista kommenteista. Lopuksi kiitokset työnantajalleni Qentinel Oy:lle tämän työn mahdollistamisesta sekä aiheesta että saamastani tuesta.

Helsingissä, 23. toukokuuta 2010,

Raine Brisk

SISÄLLYSLUETTELO

LYHENNELUETTELO, TERMINOLOGIA JA MERKINTÄTAVAT

Lyhenne	*Selitys*
IP	Internet Protocol.
LTS	Long time support.
SAAS	Software as a service.
SUT	Software under test.
Test Case	Testitapaus, vaiheittainen kuvaus yksittäisestä testistä.

Kuvaluettelo

Taulukkoluettelo

1 JOHDANTO

Ohjelmistotestauksen muuntuessa nykypäivän Suomessa yhä enemmän tavallisesta testaustoimesta kohti kokonaiskattavaa projektin laadunhallintaa muodostuvat tarvittavat työkalut yhä enemmän ja enemmän tärkeimmiksi ja ratkaisevimmiksi. Nämä työkalut on myös pyrittävä tarjoamaan tehokkaasti ja luotettavasti sekä mieluiten vielä edullisesti.

Ohjelmistojen tarjoamiseen palveluna liittyy vaatimuksia, tavoitteita, etuja, riskejä sekä haittoja. Valmiin palvelumallin, Software as a Service:n (SaaS), avulla on mahdollista määrittää tarkasti mistä on kysymys ja mitä ratkaisuja on olemassa sekä miten niitä voidaan toteuttaa. Palvelun tuottamiseen kustannustehokkuutta voidaan hakea käyttämällä avoimen lähdekoodin ohjelmistoa. Tällöin on pohdittava avoimen ohjelmistokehityksen kaikkia puolia, jotka vaikuttavat lopputuotteen käyttöön vastuunalaisessa liiketoiminnassa ja ohjelmistoprojektissa.

On pyrittävä tuottamaan palvelu ohjelmistotestauksen hallintaan, joka on sekä saatavilla että hallittu ja jonka toimintaprosessit ovat määritelty. Testaushallinnan työkaluja on saatavilla useita: avoimen lähdekoodin tuotteista, täysin kaupalliseen käyttöön tarkoitetuihin. Muuttuvia tekijöitä näissä, kuten oikeastaan missä tahansa tuotteessa, ovat hinta ja laatu. Kaupallisissa tuotteissa hinta määräytyy myyjän määrittämänä tiettyjen ehtojen puitteissa. Avoimen lähdekoodin tuotteen käytön hinta muodostuu tapauskohtaisesti sen ylläpidosta, asennuksesta, muokkauksesta tai opiskelusta aiheutuvista välillisistä kuluista vaikka itse ohjelmisto ei mitään maksaisikaan.

Tässä diplomityössä tutkitaan avoimen lähdekoodin laadunvarmistustyökalun tuottamista ylläpidettynä palveluna pienen yrityksen näkökulmasta. Työssä selvitetään ja tutkitaan käyttöön ja ylläpitoon vaikuttavia asioita.

Työ toteutetaan tutkimalla ylläpidetyn testaushallintapalvelun rakentamista ja käyttöä ensin teorian pohjalta ja sitten tapaustutkimuksena menetelminä soveltaen kyselyä ja haastattelua.

1.1 Taustaa

Ohjelmistojen laadunvarmistuspalveluja tuottava yritys Qentinel Oy totesi tarvitsevansa palvelutuotteidensa oheen lisäpalvelun, jolla voitaisiin tarjota useimmiten asiakasprojekteissa tarvittavaa työkalua laadunvarmistushenkilöstön käyttöön. Kyseessä oleva työkalu on avoimenlähdekoodin testaushallintaohjelmisto. Käytännössä työkalu on saatettava laadunvarmistushenkilöstön käyttöön IT-palvelutoimittajan tarjoamana ostopalveluna, jolloin muodostuu monitoimijaympäristö ja tilanne, jossa kokonaisvastuuta on jakamassa kolme osapuolta. IT-palveluja tarjoava yritys, Qentinel ja palvelun hankkiva asiakasyritys.

Ongelmana monitoimijaympäristössä on kokonaisvastuu. Palvelu monitoimiympäristössä on jaettu osiin ja osille on yleensä sovittu palvelutasosopimukset. Tämä toimii hyvin niin kauan, kun ongelma on yksiselitteisesti kohdennettavissa palvelun tiettyyn osaan. Ikävä kyllä monissa ongelmatilanteissa asia ei ole niin yksinkertainen ja tällöin palvelun kuluttajasta voi tuntua siltä, ettei kukaan palvelun osatoimittajista ota vastuuta palvelusta kokonaisuudessaan.

Avoimenlähdekoodin ohjelmiston ollessa kyseessä on tutkittava myös avointa ohjelmistokehitystä suhteessa suljettuun ohjelmistokehitykseen. Käytännössä tämä tarkoittaa vertailuja taloudellisesta näkökulmasta, sekä tietoturvaan liittyvien seikkojen pohtimista. Tutkittava on myös käytettävyyttä sekä prosesseja, joilla palvelua on tarkoitus hallita ja toimittaa. Tämä on lähtökohtatilanne tässä työssä tutkittavan ongelman ratkaisussa.

Testaushallintapalvelu

Testaushallinta yleisesti on toimi, jolla testaustoimia hallitaan testauksen hallitsemiseksi aikataulun, kattavuuden, järjestelmällisyyden ja tulosten varmistamiseksi. Testaushallintaa käsitellään tässä työssä tarjottavan palvelun kannalta, ns. testaushallintapalveluna. On olemassa useita ohjelmistoja, jotka tarjoavat työkalun testaushallintapalvelun toteuttamiseen. Tarkasteluun otetaan avoimen lähdekoodin tarjoamat työkalut. Avoimen lähdekoodin ohjelmistojen tarkoituksena tässä yhteydessä on tarjota kustannustehokas vaihtoehto kaupallisten ohjelmistojen usein kalliidenkin lisenssien sijaan. Tämä tuo myös etua pieniin testausprojekteihin, joissa ei välttämättä ole aikataulullisia tai budjetillisia mahdollisuuksia soveltaa kaupallisia vaihtoehtoja.

SaaS

SaaS eli Software as a service – mallia tutkitaan tässä työssä osatekijöidensä kautta, johtuen sen perusluonteesta palveluna aivan kuten mikä tahansa aiemminkin maailmalla tarjottu palvelu. SaaS on kuitenkin nähtävä omana erityispiirteitä sisältävänä palvelumallinaan, koska ohjelmistoja voidaan nykyteknisin keinoin tarjota käytettäväksi suoraan tietoverkkojen yli ilman että asiakkaalle tarvitsee toimittaa mitään fyysistä objektia. Useimmiten kyseessä on Web-palvelu jolla voidaan suorittaa eri toimintoja. Tämä toki vaatii asiakkaalta peruslaitteiston kuten tietokoneen yhteensopivine selaimineen, mutta tätä voidaan nykypäivänä pitää kohtuullisena perusvaatimuksena lähes mihin tahansa kommunikointiin niin viranomaisten kuin markkinoiden suhteen.

Tässä työssä SaaS-mallia käsitellään avoimen lähdekoodin testaushallintaohjelmiston palveluna tarjoamisen kannalta. Erityispiirteinä ja rajauksina tässä työssä toimii Qentinel Oy:ssä suunnitteilla oleva SaaS-mallin sovellus, jossa varsinaisen testauspalvelun lisäksi tarjotaan työkalu asiakkaan testausprojektiin. Ohjelmisto ei siis ole Qentinelin palvelussa pääosassa, vaan tukipalveluna jota käytetään työkaluna varsinaisen palvelun toimittamisessa.

Avoin lähdekoodi

Avoimen lähdekoodin ohjelmistoilla tarkoitetaan ohjelmistoja, joiden lähdekoodi on saatettu kaikkien saataville ja muokattavaksi. Ajatukseen kuuluu myös, ettei avoimen lähdekoodin ohjelmistoa saa myydä eteenpäin, vaan se on annettava ilmaiseksi. Paketoimisesta ja toimittamisesta saa kuitenkin ottaa kulut kattavan maksun.

Tässä työssä avoimen lähdekoodin ohjelmistot ovat mukana nimenomaan kustannussyistä. Qentinelin tarkoituksena ei ole myydä ohjelmistoa eteenpäin, vaan tarjota asiakkaan käyttöön arvioitu ja toimivaksi todettu työkalu osana suurempaa palvelukokonaisuutta, kuitenkaan sulkematta pois mahdollisuutta tarjota palveluna pelkkää työkalun ylläpitoa. Tällöin kyseessä olisi SaaS-malli puhtaimmillaan.

1.2 Tavoitteet ja rajaus

Tämän työn tarkoituksena on hakea ratkaisua niihin aihealueisiin, jotka liittyvät avoimen lähdekoodin periaatteella toteutetun ohjelmiston tarjoamiseen palveluna. Työ tarkastelee asiaa avoimen lähdekoodin, monitoimijaympäristön, tietoturvan ja testaushallinnan ratkaisujen kannalta. Vastaavasti työ ei käsittele käytännön

testaushallintapalvelun pystyttämiseen liittyviä seikkoja, eikä myöskään vastaavanlaisen järjestelmän suorituskyvyn mitoitusta, sillä tässä tapauksessa tarve on pienen käyttäjäkunnan palvelu, jonka suorituskykyvaatimusten täyttämiseen riittää tavallinen pöytätietokone.

SaaS

Software as a Service -malliin perehdytään osiensa summan lähtökohdasta, eli palvelusta ja ohjelmistosta. Luodaan katsaus painottaen mallin niitä osia, jotka ovat oleellisimpia tässä tapauksessa, eli pienimuotoisen testaushallintaohjelmistopalvelun tarjoamiseen liittyen.

Avoimen lähdekoodin ohjelmisto ja tietoturva

Avointa lähdekoodia käsitellään perusajatuksen kautta painottaen potentiaalisia kustannusrakenteita ja riskejä. Tietoturvaa käsitelään yleisesti avoimen lähdekoodin tuotteiden kannalta. Yksityiskohtaiseen analyysiin ei ole nähtävissä tarvetta, sillä tarkoituksena on luoda pohjaa järjestelmälle, jonka on tarkoitus toimia palomuurein suljetussa ympäristössä. Oleellista on pohtia tietoturvaan liittyviä uhkia kaupallisen ja suljetun lähdekoodin välillä.

Testaus ja testaushallinta

Koska kyseessä on nimenomaan testaushallintapalvelu, ei tässä työssä voida jättää huomioitta testaukseen liittyviä toimia. Testausta käsitellään laadunvarmistustoimena, johon liittyy tiettyjä erityispiirteitä. Lisäksi esitellään muutamia avoimen lähdekoodin testaushallintaohjelmistoja.

1.2.1 Työn rakenne

Tämä diplomityö koostuu kahdesta toisiaan tukevasta osasta: työn aluksi, luvussa kaksi, luodaan katsaus käsiteltävään aiheeseen teoreettiselta kannalta kirjallisuuden näkökulmasta ja luvussa kolme kuvataan empiirinen tutkimus, jota

tarkastellaan teoriaa vasten. Luvussa neljä vedetään kokonaisuus yhteen ja pohditaan tuloksia.

Työ aloitetaan luomalla katsaus SaaS-malliin sen osiensa kautta luoden näin pohjaa aiheen jatkokäsittelylle. SaaS-mallia tutkitaan pienen yrityksen hyötynäkökulmasta avoimen ohjelmistolähdekoodin kautta. Avoimenlähdekoodin ohjelmistojen tietoturvaan ja kehitykseen luodaan katsaus, sekä perehdytään näiden yhdistämiseen taloudellisesti kannattavalla mallilla. Lisäksi käsitellään tarjottavan SaaS-palvelun erityspiirrettä testausta ja testaushallintaa.

Kolmannessa luvussa kerrotaan palvelun teknisen toteutuksen rakentamisesta ja siihen liittyvistä ongelmista. Tätä jatketaan palvelun käyttöön osallistuneiden kokemusten tiedustelemisella ja tarkentavan sekä kohdennetun haastattelun tulosten esittelyllä ja analysoinnilla. Neljännessä luvussa pohditaan tuloksia sekä vedetään työ yhteen hakemalla esitettyihin tutkimuskysymyksiin vastaukset.

Tutkimuskysymykset

Tämä diplomityö pyrkii löytämään ratkaisuvaihtoehtoja tilanteisiin joissa testaushallintapalveluohjelmistoa, SaaS-mallia, avoimen lähdekoodin ohjelmistoa ja monitoimijaympäristöä hyödynnetään yhdessä toimivan kokonaisuuden luomiseksi ja myymiseksi.

Työssä pyritään löytämään vastaus seuraavia aiheita käsitteleviin kysymyksiin.

- **SaaS-malli**
    - Mitä tarkoitetaan SaaS-mallilla ja mitä se mahdollistaa testaushallintapalvelun tarjoamisen, ylläpidon ja pienen yrityksen kannalta?

- **Avoin lähdekoodi**
  - Miten avoimenlähdekoodin kehitysmallit vaikuttavat ohjelmiston loppukäyttöön, ylläpitoon ja tietoturvaan testaushallintapalvelun näkökulmasta?
- **Testaushallintaohjelmisto**
  - Testaushallintaohjelmisto on Web-sovellus, mutta miten testaus- ja laadunvarmistustoimien erikoisluonne on mahdollisesti huomioitava?
- **Monitoimijaympäristö**
  - Mitä on huomioitava että vastuunjako olisi mahdollisimman selkeä ja toiminta sujuvaa minimoiden asiakasorganisaatiolle näkyvät palvelukatkokset sekä palvelun tarjoajan kustannukset?

2 TESTAUSHALLINTAPALVELU JA SAAS

Palvelun tuottaminen ohjelmistoa hyväksi käyttäen asettaa vaatimuksia sekä ohjelmistolle että palvelumallille. Ohjelmiston on oltava riittävän laadukas palvelun tuottamiseen kaupallisesti kannattavalla tavalla ja palvelumallin on määriteltävä toiminnan rajat. Testaushallintapalvelun tuottamiseen äärimmäiset rajat asettaa palvelun luonne osana testausta ja testauksen hallintaa. Testaus itsessään jakaantuu neljään osaan ohjelmistokehityksen vaiheiden mukaisesti. Testaushallinnan toimilla pyritään hallitsemaan testausta jokaisessa vaiheessa ja vaiheesta toiseen.

Testaushallinnan tarjoaminen ohjelmistopalveluna tarjoaa mahdollisuuden etukäteen määritellä asioita, joita muuten määriteltäisiin vasta testausprojektin alkaessa. Tällaisia asioita ovat esimerkiksi kommunikaatio sekä kirjaamiskäytännöt itse testauksen osalta ja tarvittava koulutus käytettäviin ohjelmistoihin sekä vastuut ohjelmistojen ylläpidosta. Asioita palvelumuotoon järjestäessä tulee siis kirjoitettua valmis prosessimalli testausprojektin eri vaiheisiin. Tästä syystä onkin tarpeen tutkia miten ohjelmisto tuotetaan SaaS-mallin mukaisesti palveluksi ja miten tämä heijastuu testaushallinnan toimiin.

2.1 SaaS

SaaS-mallista ei voitane puhua määrittelemättä mistä lähtökohtaisesti on kysymys. Yksinkertaistaen kysymyksessähän on melko suoraviivainen ajatus siitä miten ohjelmisto, eli tietokoneohjelma, tarjotaan palveluna. Näiden kahden asian ymmärtäminen siis on lähtökohta siihen mitä SaaS oikeastaan on. Seuraavaksi

pyrin esittelemään SaaS:n kirjallisuuden kautta osiensa summana, josta muodostuu pohja testaushallintapalvelun tuottamiseen.

2.1.1 Palvelu

Palvelun yksiselitteinen määrittäminen ei ole aivan yksinkertainen tehtävä. Palvelun vastakohdaksi on nähtävä fyysinen tuote. Tuotekaupassa asiakas ostaa kyseessä olevan tuotteen ja selkeästi maksaa saadakseen kyseisen tuotteen haltuunsa. Palvelua ei samalla tavoin voi määritellä. On pohdittava sitä mitä asiakas on oikeastaan ostamassa ostaessaan palvelun.

Yksi tapa nähdä asia on että asiakas ostaa arvoa [1]. Hankkimallaan palvelulla on asiakkaalle jotain arvoa, joka voidaan mahdollisesti hyödyntää asiakkaan omassa toiminnassa. Tämä on palvelun määrittäminen arvon vaihdoksi (engl. value exchange).

Palvelu voidaan nähdä myös vastakohdan kautta, mitä palvelu ei ole. Eli palvelu ei ole fyysinen tuote. Tuote on jotain konkreettista mitä voi kosketella, mahdollisesti haistaa, maistaa ja joka lattialle pudotettuna päästää äänen [2],[1]. Tältä kannalta nähden palvelu on jotain, mikä ei ole konkreettista eikä fyysisillä mittareilla havaittavaa.

Palvelun määritelmää voidaan lähestyä myös kielitieteen näkökulmasta. Siinä missä tuote on substantiivi, on palvelu verbi. Tuote on jokin asia, artikkeli, laite tai materiaali. Palvelu on jokin suorite, toimi, ponnistus tai pyrkimys. Tuotteen hankinnassa ostaja saa tuotteesta itselleen jonkin edun. Palvelun hankkiessaan ostaja altistaa itsensä kuluille [2].

2.1.2 Ohjelmisto

Ohjelmisto on joukko (ennalta määrättyjä) käskyjä, joiden määräysten mukaan

tietokone suorittaa tietyn määrätyn tehtävän. Tässä työssä ohjelmisto nähdään palvelun tarjoamisen välineenä. Varsinkin tietoverkkojen yli käytettävien ohjelmistojen kohdalla tilanne on se, että ohjelmalla käytetään toista ohjelmaa jotain erillistä tarkoitusta varten. Puhutaan yleisemmin asiakas-palvelin (client-server) ohjelmistoista. Yleisin esimerkki tästä on selainohjelma jolla käytetään Web-palvelua. Tässä työssä tarkasteltu tilanne koskee juuri tällaista asiakas-palvelin ohjelmistoa. Tarkastelun painon ollessa palvelimen puolella kokonaisuutta.

Ohjelmiston rinnalla vastaavanlainen käsite on myös tietojärjestelmä, joka on terminä vaikuttaa lähes ohjelmiston synonyymiltä, mutta käsittää kuitenkin laajemman kokonaisuuden. Tietojärjestelmä tulkitaan käsittävän useita ohjelmistoja sekä näiden käyttäjiä jotka ovat toistensa kanssa vuorovaikutuksessa. Kyseessä ovat siis eriävät käsitteet. Tässä työssä käsitellään periaatteellisella tasolla tietojärjestelmää. Näkökulma on kuitenkin tietojärjestelmän osiin joten käsittelyn paino on ohjelmistossa.

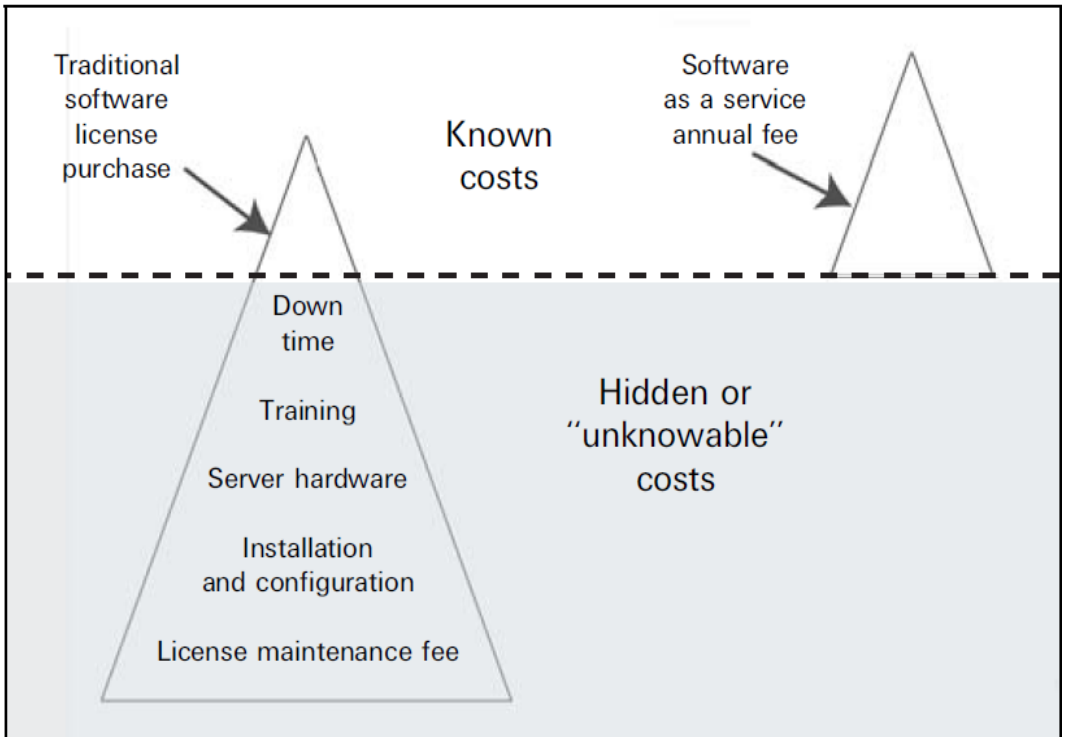
2.1.3 Hyödyt SaaS-mallista

SaaS:stä puhuttaessa tarkoitetaan palveluksi tarjottua ohjelmistoa. Tämä on terminä hyvin laaja ja käytännössä tarkoittaa ohjelman tarjoamista suoraan käyttöön, ilman asennuksia tai muuta fyysistä paikalla oloa. Ohjelma on siis vain palvelun ostajan käytettävissä, ei varsinaisesti ostajalla.

SaaS:n hyödyt yritykselle muodostuvat kustannusten ennakoitavuudesta. Tämä tulee ehkä selkeinten esille perinteisen ohjelmistokaupankäyntiin rinnastettuna. Perinteisesti ohjelmistot on kaupattu asiakkaalle lokaalisti käytettäväksi maksullisella lisenssillä.

Lisenssi on näin ollen muodostanut asiakkaalle selkeästi näkyvän kuluosan, joka on monesti ollut vain jäävuoren huippu tai kolmion kärki kokonaiskustannuksista. Näistä kustannuksista moni on jäänyt piiloon välillisiksi kuluiksi, kuten ylläpitoon, asennuksiin, päivityksiin ja niin edelleen. Lisäksi on vielä mahdollisesti tarvittu useitakin ulkopuolisija konsultteja ylläpitämään ohjelmiston infrastruktuuria ja jopa itse ohjelmistoa [3].

SaaS tarjoaa tähän kustannusmallin, jossa kaikki asiakkaalle kohdistuvat kulut ovat suoraan nähtävissä. Tosin myös SaaS-mallista palvelua käytettäessä on huomioitava, että uuden opettelu vie aina työaikaa ja täten tuottaa välillisiä kustannuksia myös SaaS-mallisen ohjelmiston käyttöönottovaiheeseen. Kuvassa 1 on havainnollistettu kulujen näkyvyyttä. Kertaeränä kyseessä saattaa olla suurempi kustannus kuin perinteisen lisenssien hankinnassa, mutta muita kuluja ei katsota olevan tai niiden oletetaan olevan huomattavasti vähäisempiä kuin perinteisessä ohjelmistohankinnassa.



Kuva 1: Piilokustannusten kolmio [3]

Tämä tuo varsinkin pienemmille, vähemmän pääomaintensiivisille yrityksille, paremman mahdollisuuden hankkia käyttöönsä suuria ohjelmistoja ilman suuren ohjelmiston hankinnan tuomia aikataulu- tuotanto- ja muita mahdollisia riskejä [3].

SaaS-palvelut ovat yleisesti hinnoiteltu käyttäjäperusteisesti, jos käyttäjiä on suhteellisen vähän, on myös hinnoiteltu tietoliikenne- tai levykapasiteettiperustaisesti. Tästä syystä SaaS-mallisen ohjelmiston kaupallisen hyödyntämisen tuomat tuotot ovat yleisesti pienemmät kuin perinteisen ohjelmistokaupan lisenssimaksuista ja ylläpidosta saatavat tuotot. SaaS:stä saatavat tulot ovat tosin toistuvampia sekä virraltaan tasaisempia ja täten paremmin ennakoitavissa. Kuten perinteisesti ohjelmistojen ylläpitomaksut.

SaaS-mallista palvelua tarjoavalle yritykselle muodostuu myös suoraa hyötyä itse mallista, sillä se mahdollistaa isojenkin, tai ainakin useampien, asiakkaiden palvelemisen käytännössä yhden kustannuksella. Näin varsinkin sellaisissa tapauksissa, joissa asiakkaiden tarpeet ovat hyvin lähellä toisiaan. Mahdollisuus on siis tehdä säästöä myös ohjelmiston konfiguraation kierrätyksellä. Mitä useampia toisiaan lähellä olevia asiakkaita on palveltavissa hyvin samankaltaisilla palvelujärjestelyillä, sitä suhteellisesti halvemmaksi yhden asiakkaan palveleminen tulee.

2.1.4 SaaS luonteenpiirteet

Tietyt piirteet tekevät SaaS-mallista juuri SaaS-mallin. Lähtökohtaisesti kysymyksessä on:

- Tietoverkon kautta käytettävä ja hallittava kaupallinen tai kaupallisesti tarjottava ohjelmisto.
- Toiminnallisuus on hallittu keskitetysti palveluntarjoajan puolella

mahdollistaen asiakkaiden pääsyn ohjelmistoon ilman lokaation asettamia rajoitteita.

- Palvelun toimitus on tyypillisesti lähempänä yksi-monelle (one-to-many) mallia kuin yksi-yhdelle (one-to-one) mallia. Arkkitehtuurit, hinnoittelu, kumppanuudet ja hallinta noudattavat yksi-monelle mallia.
- Palvelun ominaisuudet päivitetään keskitetysti poistaen asiakkaan tarpeen ladata päivityksiä tai lisäosia.
- Jatkuva integraatio suurempiin järjestelmien muodostamiin verkkoihin.

Palveluihin perustuvat ohjelmistoarkkitehtuurit ovat luonnostaan monimutkaisempia kuin perinteisemmät ohjelmistojen arkkitehtuurit. Näin jo pelkästään siksi, että palveluorientaatiomallit sisältävät monesti yhteyksiä useisiin tietokantoihin sekä oheispalveluihin.

2.1.5 SaaS implementaatiotasot

SaaS jaetaan yleensä neljään tasoon sen mukaan kuinka kypsä käytettävän SaaS mallin katsotaan olevan [13].

- 1. Taso
  - Palvelut kustomoituja ja asiakkaille luotu erilliset palveluinstanssit tarjolla olevasta palvelusta.
- 2. Taso
  - Konfiguroitavia palveluita jotka tarjoavat suuremman joustavuuden asiakkaalle konfiguroitavan metadatan kautta. Useat asiakkaat käyttävät nyt samaa ohjelmaa eri instansseina.
- 3. Taso
  - Konfiguroitavia usean asiakkaan ohjelmistoja. Yksi ohjelmistoinstanssi palvelee nyt useita asiakkaita. Käyttää tehokkaammin palvelimen resursseja, mutta skaalautuvuus aiheuttaa ongelmia.

- 4. Taso
    - Skaalautuva, konfiguroitava usean asiakkaan hoitava palvelu. Tällä tasolla ohjelmisto skaalautuu, ohjelmiston toiminnallisuudet on eriytetty omille palvelimilleen ja kuormatasatulla palvelinfarmilla pyörii identtisiä ohjelmistoinstansseja. Tällä tasolla palveluntarjoaja voi lisätä kapasiteettia palveluun joustavasti tarpeen mukaan ilman vastaavia ohjelmistomuutoksia.

Implementaatiotasot ovat melko karkea tapa esittää SaaS. Tasot kertovat tuotetun palvelun kypsyydestä ja jalostusasteesta, mutta monesti todellinen SaaS-mallina tuotettu palvelu ei osu suoraan mihinkään kuvatuista tasoista, vaan ilmentää osia useammasta tasosta. Kuten tässä diplomityössä tutkittavassa esimerkissä, jossa tuotettava palvelu omaa piirteitä sekä ensimmäiseltä että toiselta tasolta.

2.1.6 Virtualisointi

Virtualisointi on yksi tapa toteuttaa SaaS, sillä virtualisoinnilla tietokonekapasiteetti tehdään abstraktiksi. Virtualisoinnilla saadaan olemassa olevasta tietotekniikkaraudasta tehot tarvittaessa täysin hyödynnettyä käyttöjärjestelmän tai sovellusten käyttöön simuloimalla ympäristö, joka ei fyysisesti vastaa todellisuutta. Tämä helpottaa SaaS-toimittajan skaalaustoimintaa ja täten mahdollistaa pienemmät kustannukset myös asiakkaalle [4].

2.1.7 Konfiguroitavuus

Konfiguroitavuus määrittyy SaaS:n kannalta siten että useaa asiakasta palveleva ohjelmisto ei enää ole ohjelmakoodiltaan kustomoitavissa. Tällöin sovellusten konfigurointia on mukautettava. Asiakkaan on huomioitava ettei hänen ole mahdollista saada kaikkea haluamaansa oman mielensä mukaisesti. Konfiguroitavina asioita pidetään ainakin: käyttöliittymää, työvirtaa, dataa, ja järjestelmään pääsyä. Konfiguroitava data jakaantuu vielä kahteen, sovellusdataan ja järjestelmädataa. Lisäksi voi olla myös muita asioita, jotka ovat enemmän tapauskohtaisia riippuen implementoidun sovelluksen luonteesta [4].

Konfiguroinnin ollessa enemmissä määrin asiakkaan vastuulla on SaaS-implementaatiossa kiinnitettävä huomiota konfiguraatiokäyttöliittymien käytettävyyteen. Käyttäjäkokemus konfiguroinnin aikana vaikuttaa paljon SaaS-implementaation hyväksymiseen ja onnistumiseen [4].

2.2 Avoin lähdekoodi

Termillä avoimen lähdekoodin ohjelmisto viitataan ohjelmistoon joka on tekijänoikeuksiltaan avoin ja noudattaa jotakin avoimen lähdekoodin lisenssiä. Nämä lisenssit antavat oikeuden muokata ja kehittää ohjelmistoa sekä edelleen levittää sitä muutettuna tai muuttamattomana. Avoimen lähdekoodin ohjelmistoja kehitetään tietyn, useimmiten avoimen kehitysyhteisön toimesta.

2.2.1 Avoimenlähdekoodin hyödyt

Avoimen lähdekoodin ohjelmistojen käyttöönotolla ja käytöllä yritys säästää rahaa kaupallisten lisenssien hankkimiseen verrattuna ja pyrkii toimimaan kustannustehokkaasti. Sillä avoimen lähdekoodiin ohjelmiston yritys tai yksityinen henkilö saa ottaa ohjelmiston käyttöön ilman kehittäjien palkkiosta muodostuvaa kuluerää. Vastaavasti kuitenkin kaikki ylläpito-, muutos-, hallinta ja vastaavat ohjelmiston käytön tukeen liittyvät kulut ovat täysin käyttäjän vastuulla. Tämä saattaa luoda tilanteen jossa avoimen lähdekoodin hankinnan edullisuus häviää ylläpitoon liittyviin kustannuksiin. On siis tarkkaan mietittävä saatavilla olevan avoimen lähdekoodin ohjelmiston soveltuvuutta haluttuun tarkoitukseen.

Tässä diplomityössä tutkittavana olevassa tilanteessa ohjelmiston pääasiallinen tarkoitus on olla aputyökaluna varsinaisessa laadunhallintaprojektissa, sekä myyntiä edistävänä tekijänä, ikään kuin kaupantekijäisenä. Seikka, joka ei toki vähennä ylläpitotoimien tarvetta, mutta vaikuttaa ylläpidettävään valmiusasteeseen. Tietojärjestelmä voidaan siis asentaa ja ottaa käyttöön vain

tarvittaessa ja määräajaksi. Tähän avoin ohjelmistokehitys suo joustavuutta. Lisenssi tai muitakaan kuluja ei ole, ellei ole ohjelmiston käyttöäkään.

Haasteet

Avoimen lähdekoodin ohjelmisto on yleisesti vallitsevan käsityksen mukaan tietyiltä ominaisuuksiltaan kaupallisen kehityksen ohjelmistoon nähden enemmän haasteita sisältävä valinta. Moni näistä riskeistä on ymmärrettävissä ohjelmiston laatuun liittyviksi ja kehitysmenetelmästä syntyviksi. Ehkä parhaiten avoimen ohjelmistokehityksen tuotteiden tuomiin haasteisiin voidaan vaikuttaa ohjelmistotuotteen valinnalla.

Riskit

Avoimen lähdekoodin projekteista epäonnistuu suurin osa [5]. Tämä jo yksistään asettaa avoimen lähdekoodiin ohjelmiston valinnalle ikärajoitteen. Kun projekti on elänyt jo pitkään, on sen katoaminen tai muuttuminen passiiviseksi oleellisesti epätodennäköisempää. Kehitystyön päättyminen tai katoaminen on avoimen lähdekoodiin liittyvistä riskeistä vaikutuksiltaan suurin, joskaan ei vaikutuksiltaan välitön. Käytössä oleva ohjelmisto säilyy toimintakykyisenä ja rajoituksin käyttökelpoisena kehitystyön päättymisestä huolimatta.

Tässä vaiheessa yhdenlaisen riskin tuo esiin tekninen vanheneminen, joka johtaa testaushallintapalvelua tarjotessa puutteen markkinoille tarjottavassa tuotteessa. Toisenlainen riski tulee esiin, jos kehityksensä lopettaneesta ohjelmistosta löytyy jokin vakava virhe joka esimerkiksi aiheuttaa datan korruptiota tai mahdollisesti altistaa järjestelmän tietomurrolle. [5],[6]

Mahdollisuudet

Havainnot osoittavat että avoimen lähdekoodiin ohjelmiston käyttöön ovat etupäässä syynä positiivinen käyttäjäkokemus ja ohjelmiston laatu. Käyttäjäkokemukseen oleellisena tekijänä vaikuttaa itse ohjelmiston laadun lisäksi

yhteisön tarjoaman palvelun laadukkuus. [5] Ohjelmiston käyttäjä tarvitsee tukea ja yhteistyötä niin ohjelmiston kehityksen kuin sen käytönkin aikana. Myös päivitykset, korjaukset ja uudet julkaisut ohjelmistosta on tulkittava ohjelmiston tueksi ja siten selkeästi vaikuttava käyttäjien kokemukseen ohjelmistosta. Täten aktiivinen kehitystyöryhmä vaikuttaa positiivisesti käyttäjien kokemuksiin ohjelmistoprojektista. Vastaavasti huonosti hyödynnetty avoin ohjelmistokehitysyhteisö vaikuttaa heikentävästi käyttäjän kokemukseen.

Tosin on huomioitava, että avoimen kehityksen ohjelmistotuotteeseen vaikuttaa osaltaan moni tekijä, joten käyttäjien kokemukset eivät ole suoraan johdettavissa pelkästään yhdenkään osatekijän onnistumisesta tai epäonnistumisesta. Tällöin käyttäjä helpommin jatkaa toimintaansa avoimen lähdekoodin ohjelmiston parissa [5].

2.2.2 Avoimen lähdekoodin tietoturva

Tietoturva on laaja-alainen käsite ja avoimen lähdekoodin ohjelmistojen tietoturvallisuus on edelleen laajan keskustelun alaisena. Tätä keskustelua käydään monitahoisesti eri ohjelmistojen ympärillä. Etupäässä kuitenkin akselilla Windows – Linux. Keskustelun ydin voidaan kiteyttää seuraavaan vertailevaan analogiaan [7]:

- Kumpaan lukkoseppään luottaisit enemmän?
    - Tuttuun ammattitaitoiseen kaveriin, joka muiden katseilta piilossa omassa pajassaan rakentaa lukot, jotta pahat pojat (tai kukaan muukaan) eivät pääse perille hänen tekemiensä lukkojen salaisuuksista.
    - Vai ehkä vähemmän tuttuun joka antaa työnsä julkiseen tarkastukseen, jotta kaikki voivat arvioida lukkojen toimintaa sekä hyvässä että pahassa?

Keskustelun ydin on siis julkisessa tarkastelussa. Onko etu siinä, että kaikki hyvät

ja huonot ominaisuudet ovat mahdollisesti heti kaikkien arvioitavana vai onko etu siinä, että tekniset ominaisuudet on piilotettu salaisuuksien taakse? Tällöin oletetaan että salaisuus on osa suojausta, myöskin hyvässä ja pahassa.

Kuitenkaan ei ole olemassa minkäänlaista takuuta itsessään siitä, että laittamalla lähdekoodi julkisesti saataville saavutettaisiin jotain parannusta ohjelmiston tietoturvassa. Määrällinen argumentointi voi kyllä auttaa päätöksen tekemisessä, mutta ei tarjoa kunnollisia mittareita sen arviointiin, ovatko ohjelmistot alunperinkään tietoturvallisia [8].

Haavoittuvuudesta

Kaikki ohjelmistot ovat haavoittuvimmillaan aikavälillä, joka kuluu altistavan virheen löytymisestä ja dokumentoinnista virheen testatun korjauksen toimittamiseen. Ohjelmistojen jonkinlaisena tietoturvallisuuden mittana voidaan siis pitää tämän aikavälin pituutta [8].

Tietoturvaa voidaan täten mitata asettamalla keskenään vaihtoehtoisten ohjelmistojen virheen löytymisestä korjaamiseen kuluvat ajat vertailuun. Wittenin, Landwehrin ja Caloyannidesin artikkelin mukaan [8] vertaillun avoimen lähdekoodiin ohjelmiston (Red Hat Linux) ja suljetun ohjelmiston (Microsoft) välinen ero korjauksen toimittamiseen kuluneessa ajassa oli noin kaksikertainen avoimen lähdekoodiin yhteisön eduksi. Lisäksi artikkelissa todetaan, että pienellä aktiivisuuden lisäyksellä olisi avoin yhteisö saanut kiertoajan helposti puoliintumaan.

On kuitenkin huomioitava että [8]:

- Kaikki virheet eivät välttämättä ole keskenään yhtä vaarallisia.
- Paljon virheitä löytävä ja ne nopeasti korjaava ei välttämättä ole yhtään parempi kuin vain muutaman virheen löytävä ja pitkään niitä korjaava.

- Keskenään erilaisten ohjelmistojen vertailu vaikuttanee tuloksiin.
- Oletettujen korjausten vaikutus voi vaihdella.

Eli kokonaisuutena avoimen lähdekoodiin ohjelmistokehityksen korjausherkkyys on suurempi, mutta ei välttämättä tehokkaampi kuin kaupallisen, suljetun ohjelmistokehityksen. Avoimen ohjelmiston tietoturvan taso lepää vapaaehtoisten vastuulla, jos heillä sattuu olemaan kykyä ja resursseja tehtävään. Toiselta puolelta voidaan nähdä, etteivät suljettujen ohjelmistojen kehitysmallit kannusta tuottamaan turvallisempia järjestelmiä.

2.3 Ohjelmistotestaus

Mitä on ohjelmistotestaus? Useimmilla meistä on jonkinlainen käsitys termistä testaus. Testaus on kuitenkin monitahoinen ja laaja käsite, jonka määritteleminen yksioikoisesti ei ole helppoa tai edes mielekästä. Parhaiten testauksen määritelmän voi mieltää prosessina. Testaus on siis prosessi ohjelmiston tai sen osan analysointia erojen havainnoimiseksi olemassa olevien ja oletettujen tilojen väliltä [10]. Testaus ei ole muusta maailmasta eristäytynyt toimenpide. Testaus ei myöskään ole ohjelmistonkehitystoimia. Testaus on tukitoimi, merkityksetön ilman ohjelmistokehityksen prosesseja, eikä tuota itsessään mitään. Ilman ohjelmistokehitystä ei tuoteta mitään testaukseen [10]. Testaus on joka tapauksessa hyvin tärkeä osa minkä tahansa ohjelmistotuotteen elinkaarta alkuperäisestä ideasta, kehitykseen, tuotantoon ja käytöstä poistoon. Testauksella on paikkansa kaikkien näiden vaiheiden välissä ja aikana [10]. Ohjelmistotestaus voidaan siis nähdä kaikkialle ulottuvana empiirisenä tutkimuksena, jonka tarkoitus on tuottaa osapuolille tietoa testattavan ohjelmiston laadusta suhteessa sen aiottuun käyttöympäristöön.

2.3.1 Testaus

Testaus on oleellisesti ohjelmistotuotannon kustannuksiin vaikuttava tuotannon vaihe. Testaukseen liittyy työvaiheina karkeasti seuraavia asioita: testauksen

suunnittelu, testiympäristön luonti, testin suorittaminen ja tulosten tarkastelu sekä virheiden korjaaminen. Kaikenkattavaa testausta on mahdotonta tehdä ja näin ollen päätös siitä mitä testataan, miten paljon ja milloin on testattu tarpeeksi, on aina kompromissi aikataulun, resurssien, välineiden ja testitulosten luotettavuuden välillä.

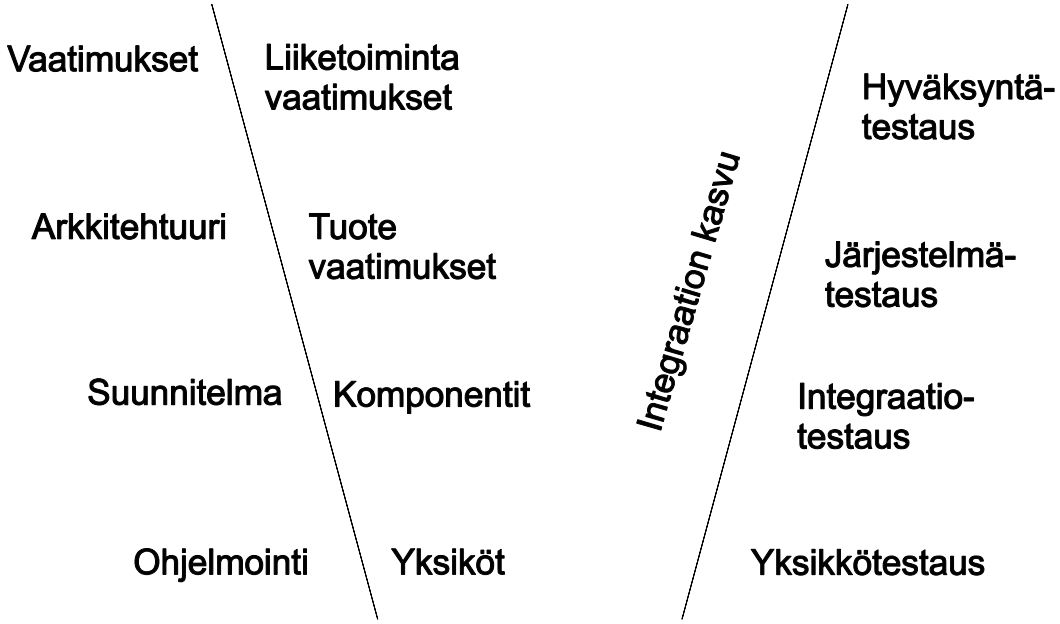
Tavallisesti työskentely testaus- tai laadunvarmistusorganisaatiossa osana ohjelmistokehitystä käsittää testauksen suunnittelun, valmistelun, itse testaustyön ja raportoinnin sekä mahdollisesti projektin päättämisen. Testauksen suunnittelu käsittää, tapauskohtaisesti vaihdellen, testausstrategian, aikataulujen ja testien ajon suunnittelun sekä suunnitelmat raportoinnista ja testauksen päättämisestä. Valmistelu vaiheessa perehdytään yksityiskohtaisemmin testattavaan ohjelmistoon, Software under test:iin (SUT). Mahdollisesti asennetaan hallinta- ja muita työkaluja sekä joissain tapauksissa itse SUT. Testauksen valmisteluun kuulu olennaisesti testitapausten, test case:en suunnittelu ja kirjaaminen. Suunnittelu- ja valmisteluvaiheet ovat monesti pidempi kestoisia ja hankalampia kuin varsinainen testausvaihe. Näin varsinkin automaatiotestauksen tapauksessa, jossa automaatiota voidaan rakentaa laajaa ohjelmistoa vasten pitkään, mutta automaatiotestien suoritus ja tulosten raportointi voi olla ohi yhdessä vuorokaudessa. Testauksen suoritusvaiheessa suunnitellut testit suoritetaan ja tulokset raportoidaan. Testauksen etenemistä usein valvotaan ja hallitaan tarkoitukseen tehdyillä ohjelmistotyökaluilla, joiden tarjoamista laadunvalmistusprojektien käyttöön tämä diplomityö käsittelee. Lopputulokset raportoidaan laadunvarmistusprojektin jälkeen intressiryhmille, useimmiten ulkoistetuissa laadunvarmistusprojekteissa tilaajayrityksen vastuullisille henkilöille.

Testaukseen liittyvien haasteiden takia suoritettava testaus tulisi aina määritellä mahdollisimman tarkkaan ja soveltaa ohjelmiston elinkaaren kaikkiin vaiheisiin. Monesti kuitenkin testaus toteutetaan menetelmiltään, työkaluiltaan ja elinkaaren

kannalta kattavuudeltaan puutteellisesti. Vaikkei täydellinen testaus olekaan mahdollista, on hyvin suunnitellulla testauksella kuitenkin mahdollista pudottaa ohjelmistotuotannon kokonaiskustannuksia.

2.3.2 Testauksen tasot

Testauksen tasoja katsotaan olevan virheiden paikallistaminen ja korjaus, yksikkötestaus, integrointitestaus, järjestelmätestaus sekä hyväksymistestaus [9]. Virheiden paikallistaminen ja korjaus sekä yksikkötestaus ovat testauksen alinta tasoa ja yleensä ne katsotaan ohjelmoijan tehtäväksi. Muilla tasoilla on hyödyllisempää käyttää varsinaista ja mieluiten ohjelmistokehityksestä riippumatonta ulkopuolista testaajaa testauksen tehostamiseksi sekä virheiden paremman havaittavuuden vuoksi.



Kuva 2: Ohjelmistotestauksen V-malli

Testauksen tasot asettuvat luonnollisena jatkumona ohjelmistokehityksen erivaiheisiin. Kuvassa 2 on esitetty ohjelmistotestauksen V-malli, joka kuvaa ohjelmistokehityksen tasoja ja niihin liittyviä vastaavia testauksentasoja. Kuvan V-malli lähtee ylhäältä vasemmalta liiketoimintavaatimuksista ja niistä määrittyvistä ohjelmiston vaatimuksista. Tätä vastaa testauksen tasoissa hyväksyntätestaus, jossa määritellään täyttääkö tuotettu ohjelmisto sille asetettuja

liiketoiminnan vaatimuksia ja missä määrin. Seuraavalla tasoilla tulevat parit: arkkitehtuuri – järjestelmätestaus, ohjelmistosuunnitelma – integraatiotestaus ja ohjelmointi – yksikkötestaus. V-malli siis sovittaa ohjelmistokehityksen tasot vastaaviin testauksen tasoihin. Testauksen taso myös määrittää laadunvarmistusprojektin luonnetta asettaen vaatimuksia käytettäville työkaluille, palveluille ja asiantuntijoille.

2.3.3 Testaushallinta

Testaushallinnalla tarkoitetaan niitä toimia, joita laadunvarmistushenkilöstö suorittaa hallitakseen meneillään olevaan testausprojektia. Testaushallinan pääkategorioita ovat testaussuunnittelu, testitapausten suunnittelu ja suorittaminen, virheiden kirjaaminen, mittaus sekä raportointi.

Virheiden hallintaa voidaan pitää myös osittain erillisenä varsinaisesta testaushallinnasta, niiden selkeästi oman elinkaarensa takia. Virheitä voi olla tarpeen kirjata ja hallita vielä testauksen ja varsinaisen testaushallinnan päätyttyä ohjelmiston ollessa jo tuotannossa tai testauksen siirtyessä seuraavaan vaiheeseen, mahdollisesti uuden testaussuunnitelman alle.

2.3.4 Testaushallinnan työkaluja

Testaushallintaohjelmisto on siis työkalu, joka vähintään pitää kirjaa testitapauksista ja niiden tilasta. Siis onko testitapaus suoritettu ja milloin sekä mihin tulokseen testiajo päättyi. Monesti testaushallintatyökaluista löytyy myös ominaisuuksia spesifikaation ylläpitoon, testitapausten niputtamiseen testikokonaisuuksiksi ja virheiden hallintaan.

Avoimen lähdekoodin testaushallintaohjelmistoja on saatavilla vaihtelevan laatuisina ja vaihtelevalla ominaisuusvalikoimalla. Suoritetun perusominaisuuksien katsauksen perusteella muutamasta avoimen lähdekoodin testaushallintaohjelmistosta muodostettiin alustava käsitys markkinoilla vallitsevista mahdollisuuksista.

Rth

Luvatuilta ominaisuuksiltaan kattava työkalu, mutta edelleen keskeneräisen oloinen. Ominaisuuksia ovat muun muassa vaatimustenhallinta, virheidenhallinta ja raportointityökalut. Käytettävyydessä on parantamisen varaa, sillä moni hyvinkin erilainen toiminto näyttää näytöllä hyvin samanlaiselta. Kokonaisuutena tuote on kuitenkin keskeneräinen ja kehittävä yhteisö kärsii kehittäjien puutteesta.

TestLink

Työkalu antaa vaikutelman huolellisesta työstä. Käyttäjien hallinta on toteutettu kustomoitavilla rooleilla, joita on kattavasti jo oletusasetuksissa. Virheidenhallintaan on tarjolla integraatio erilliseen virheidenhallintaohjelmistoon. Yhteisö on aktiivinen ja uusia versioita ilmestyy noin muutaman kuukauden välein. Kokonaisuutena tuote on hyvin toimiva ja suurimmat puutteetkin pääosin kosmeettisia tai käyttäjän toimilla kierrettävissä.

Testopia

Testopia on osa tunnettua Mozilla-projektia ja oikeastaan rakennettu laajentamaan Mozilla-projektin virheidenhallinta työkalua Bugzillaa testaushallinnan ominaisuuksilla. TestLinkin tavoin aktiivisen yhteisön tuote. Kattavasti ominaisuuksia ja käyttökelpoinen tuote.

2.3.5 Valinta

TestLink:n osalta oli Qentinelin henkilökunnalla jo käyttökokemuksia, jotka olivat valtaosin positiivisia. Suoritetun katsauksen ja Qentinelin henkilökunnan aiempien positiivisien kokemusten perusteella päätettiin ottaa TestLink tarjottavaan palveluun ohjelmistoksi.

TestLink valittiin myös aktiivisen kehitysyhteisönsä perusteella. Tämä osoittautui

hyväksi valinnaksi myös siksi, että mukana olleen Rth ohjelmiston kehitys näyttää tätä kirjoittaessa pysähtyneen jo lähes kokonaan. Edellisestä uuden version julkaisustakin on jo yli vuosi aikaa, normaalin julkaisutahdin ollessa noin nelisen kertaa vuodessa. Täten alleviivaten kehitysyhteisöön liittyviä riskejä työkalun valinnassa.

TestLink:n valinta suhteessa Testopiaan ei ollut yhtä ilmeinen kuin Rth:n suhteen. TestLink:n valinnan puolesta puhuivat kuitenkin helpommin suoritettava asennustyö, joka johtui pääasiassa Testopian vaatimien ohjelmakirjastojen määrästä ja laadusta. Täten eri laiteympäristöissä ongelmaksi tulisi helpommin ohjelmakirjastojen saatavuus ja näihin liittyvät versio-ongelmat. Testopiaa vastaan puhui myös valinnan aikainen henkilökunnan vähempi käyttökokemus, joten vertailun ja positiivisten käyttökokemusten perusteella valittiin käyttöön TestLink.

3 TAPAUSTUTKIMUS TESTAUSHALLINTAPALVELUN TOTEUTUKSESTA SAAS-MALLISSA

Tätä diplomityötä varten suoritettiin tapaustutkimus menetelminään kyselytutkimus sekä haastattelu. Tutkimuskysymyksiä lähestyttiin kvalitatiivisella tutkimuksella ja käytettiin metodeina tapaustutkimusta ja kyselyä. Tarkoituksena oli selvittää käytettävän avoimen lähdekoodin ohjelmiston soveltuvuutta suppean käyttäjäryhmän työvälineeksi käytetyn palvelumallin mukaisesti. Tapaustutkimus valittiin menetelmäksi koska kyseessä on yksittäinen järjestelmä, jonka soveltuvuutta tarkoitukseensa tarkasteltiin. Tämän tarkastelun tuloksia pyritään tässä työssä yleistämään laajemmin pienen käyttäjäryhmän testaushallinta SaaS-implementaatioihin.

Tutkimuksessa järjestelmän kokoonpanoon liittyvä tutkimus suoritettiin tapaustutkimuksena kokeillen vaihtoehtoista ratkaisua, ennen varsinaisen ja lopullisen ratkaisun toteuttamista. Tapaustutkimusta laajennettiin loppukäyttäjiin kohdistuneella kyselytutkimuksella, jolla selvitettiin käyttäjien kokemuksia. Käyttäjien kokemuksia selvitettiin käytettävästä ohjelmistosta, prosesseista ja järjestelmästä kokonaisuutena.

3.1 Tapaustutkimus tutkimusmenetelmänä

Tapaustutkimuksessa tutkitaan jotain tiettyä yksittäistä tapahtumaa, esiintymää tai ilmiötä. Tämän yksittäisen tapauksen ominaisuuksista pyritään johtamaan yleistettävissä olevia laadullisia johtopäätöksiä, jotka koskevat vastaavia tapauksia laajemminkin, tosin vain vastaavien reunaehtojen täyttyessä. Tapaustutkimuksen

tavoitteena ei ole löytää yleismaailmallisia, yleistettävissä olevia totuuksia tai edes tyypillisiä syy- ja seuraussuhteita. Sen sijaan tutkimuksen painotus on itse tapauksen tutkimisessa ja kuvaamisessa.

Sovellettavassa tapauksessa tapaustutkimuksen käytön puolesta puhuu myös todella rajattu tutkimusalue. Yksittäisen ja pienen käyttäjäryhmän ohjelmistopalveluinstanssin tutkimiseen ei ole saatavilla suuria aineistomääriä, jotta voitaisiin tehdä johtopäätöksiä aineiston kvantitatiivisista ominaisuuksista. On siis pitäydyttävä tutkimusaineiston kvalitatiivisessa puolessa ja pyrittävä tulkitsemaan niitä johdonmukaisesti tulosten saamiseksi.

Tapaustutkimus pyrkii vastaamaan kysymyksiin miten ja miksi. Tapaustutkimus on kelvollinen menetelmä tilanteessa jossa tutkijalla on jonkinlainen kontrolli tutkittavaan tapaukseen ja lisäksi tapaus on kohdistettavissa nykyhetken kontekstiin. Täten pyritään saamaan kokonaisvaltaisempi tulkinta tapauksen ominaisuuksista. Tapaustutkimuksessa on myös järkeenkäypää valita tutkimuksen kohteeksi ne tapaukset, jotka edustavat ääripäitä tutkittavasta aiheesta. [12]

Toisessa valitussa tutkimusmenetelmässä, kyselyssä tutkimuksen paino on kvantitatiivisissa asioissa ja tutkimusta pyritään tekemään mahdollisimman laajaa materiaalia analysoiden hakien vastauksia kysymyksiin kuten: kuka, missä, kuinka paljon ja kuinka monta asettaen tutkittavan kohteen usein jonkinlaiseen historialliseen yhteyteen. Kysely on soveltuva tutkimusmenetelmä, kun tarkoituksena on selvittää ihmisten kokemuksia, tuntemuksia ja uskomuksia. Lisäksi kysely on kattava menetelmä, kun kerätään informaatiota kuvaamaan, vertailemaan tai selittämään tietoa, asenteita ja käytöstä [11]. Tässä työssä kyselyä sovelletaan tapaustutkimuksen välineenä. Käyttäjiltä kysytään kysymyksiä, joilla tutkitaan palveluprosessien ja ohjelmisto valinnan onnistumista. Pienestä käyttäjämäärästä johtuen, myös kyselyn aineisto on pieni.

3.2 Tutkimuksen toteutus kyselytutkimuksena

Tapaustutkimus käyttäjien kokemasta avoimen lähdekoodin Testlink-testaushallintaohjelmiston soveltumisesta SaaS-implementaatioksi toteutettiin strukturoidun kyselytutkimuksen menetelmällä. Strukturoidussa kyselytutkimuksessa kysymyksiin on asetettu valmiit vastausvaihtoehdot kulloiseen tarkoitukseen soveltuvassa skaalassa vastaajan voidessa valita mieleisensä vastausvaihtoehdon skaalan negatiivisen ja positiivisen ääripään väliltä.

Tutkimuksen yksi kysymyksistä oli selvittää valitun Testlink-ohjelmiston soveltuvuutta sille asetettuun tehtävään. Tätä soveltuvuutta tutkittiin strukturoidulla käyttäjäkokemuskyselyllä. Soveltuvuutta haluttiin tutkia nimenomaan käyttäjien näkökulmasta, jolloin rakennetun järjestelmän suorituskyvyn mittaaminen ei edes olisi tuonut oikeita vastauksia. Kyselytutkimukseen päädyttiin myös siitä syystä, että vaikka ohjelmistoa ja sen palvelinalustaa voitaisiin mitata kuormituksen aikana, ei tämä mittaus olisi tuonut mitään arvoa soveltuvuuden tutkimiseen. Näin siksi, että pienelle käyttäjämäärälle tarkoitettu palvelu ei olisi merkittävästi kuormittunut pienen käyttäjäryhmän sitä käyttäessä. Lisäksi palvelu oli lopulta toteutettu ostopalveluna virtualisoituun ympäristöön, jonka käytettävissä oleviin resursseihin vaikuttaa myös muut samalla fyysisellä palvelinalustalla olevien virtualisoitujen palvelujen kuorma. Mittaamalla palvelimen teknisiä ominaisuuksia, olisivat mittaustulokset mitä todennäköisemmin tuottaneet tulokseksi yksittäisen tilannekuvan palvelimen kuormasta, joka välttämättä ei edes olisi toistettavissa samanlaisena samanlaisella tutkittavan palvelimen kuormalla, eikä näin ollen kertoisi mitään itse tutkimuskohteesta.

3.3 Toteutettu SaaS-implementaatio

Kuten aiemmin tässä työssä mainittu, toteutettava palveluntarjontamalli on hyvin

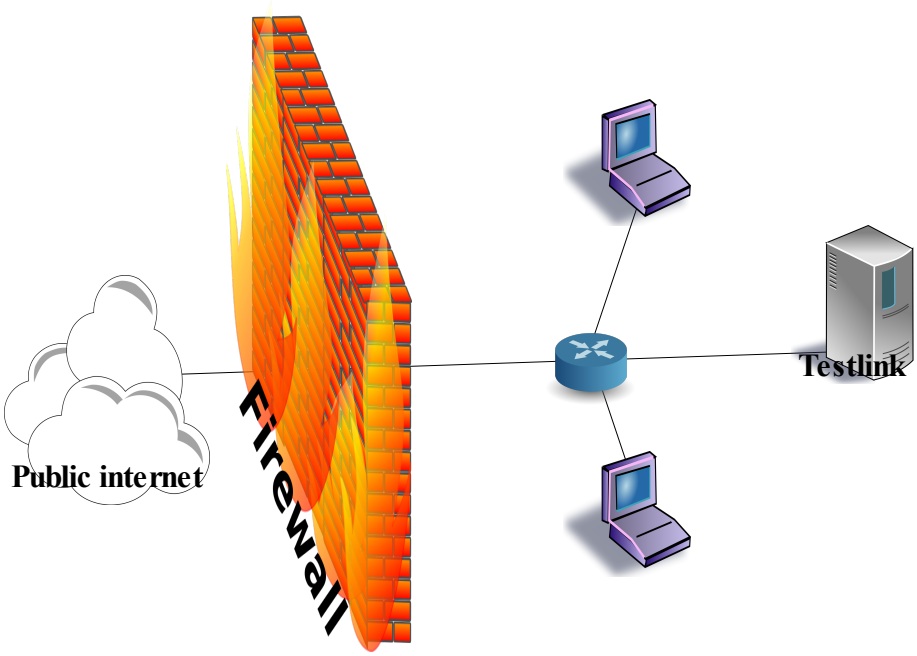
matalan tason SaaS-implementaatio. Kyseen ollessa varsinaisen ohjelmiston laadunvarmistukseen painottuvan palvelun myynnin tukemiseen tähtäävä ohjelmistollinen lisäpalvelu, voidaan ohjelmiston asiakaskohtainen kustomointitaso pitää alhaisena ja tarjota toimivaksi todettu peruspalvelu jokaiselle asiakkaalle erillisenä virtualisoituna ohjelmistoinstanssina. Tällä kuvauksella SaaS-tasoksi muodostuu korkeintaan taso 2. Useimmiten kuitenkin alhaisimman ykköstason implementaatio. Käytännössä tämä tarkoittaa, rajoitetta asiakkaan kannalta palvelun saatavuuteen. Palvelu on saatavilla vain erillisestä tilauksesta erilliseksi ajaksi. Ei jatkuvana palveluna, jonka laskutus muodostuu asiakkaan käytön mukaan.

3.3.1 Palvelinympäristön toteutus

Palvelu rakennettiin aluksi siten, että sitä käytettiin Qentinelin sisäisessä verkossa omalta virtualisointipalvelimelta. Tässä oli sekä rajoitteena että tavoitteena suljettu verkkoympäristö, jossa oli mahdollista todeta asennuksen onnistuminen ilman riskiä ulkopuolisen verkkohyökkäyksen kohteeksi joutumisesta tai asiakastietojen vuotamisesta julkiseen tietoverkkoon.

Ensimmäinen vaihe

Ensimmäisessä vaiheessa järjestelmä rakennettiin Qentinelin toimiston sisäverkkoon. Tätä varten virtualisoitiin Ubuntu 8.04.4 LTS (Hardy Heron), jolle asennettiin TestLink 1.7. Toimiston sisäverkon verkkotopologia oli valmiiksi rakennettu siten, ettei sisäverkon ulkopuolisista IP-osoitteista ollut pääsyä palvelimelle.



Kuva 3: Ensimmäisen vaiheen verkkotopologia

Asennustoimien ja välttämättömän konfiguroinnin jälkeen ohjelmisto otettiin asiakasprojektin käyttöön. Käytön alkuvaiheessa konfiguraatiota kustomoitiin vastaamaan paremmin juuri kyseisen asiakkaan projektin tarpeita ja erityispiirteitä. Tällä verkkojärjestelyllä (kuva 1) ja konfiguraatioilla järjestelmän toiminnallisuudesta ei saatu käyttäjiltä negatiivista palautetta, mutta hyvin pian projektin alun jälkeen tarve päästä käyttämään järjestelmää myös Qentinelin toimistoverkon ulkopuolelta kävi ilmeiseksi. Tarve koski lähinnä mahdollisuutta käyttää järjestelmää asiakkaan verkosta ja siten hallitusta ja rajoitetusta verkko-osoiteavaruudesta käsin. Tästä syystä tehtiin nopeasti päätös siirtää järjestelmä Qentinelin IT-palveluntarjoajan ylläpidettäväksi ja hallintaan.

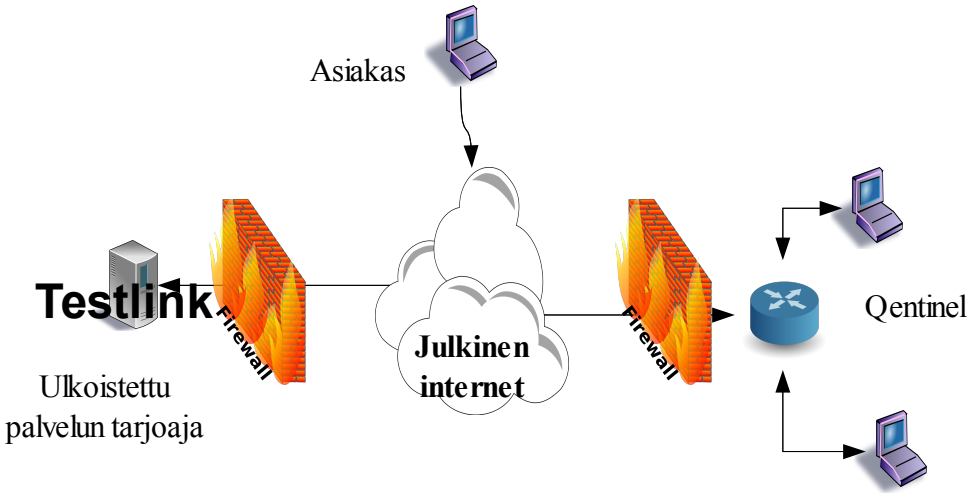
Tähän siirtoon liittyi käytetyn tietokannan korruptoitumisriski, siksi ohjelmiston käyttö oli koeluontoista. Tässä vaiheessa myös määriteltiin ohjelmistoon liittymättömiä toimintaprosesseja, joilla pyrittiin varmistamaan asiakkaan palvelutaso ja ohjelman käytettävyys mahdollisimman laajasti. Lisäksi näiden prosessien määritteleminen koettiin tässä vaiheessa erittäin tarpeelliseksi, sillä mukaan tuli kolmasosapuoli. Osapuolien määrä kasvoi ja täten myös virheiden ja

väärin ymmärrysten mahdollisuus katsottiin kasvaneen niin suureksi, että prosessien tarkka ja yksiselitteinen määritys katsottiin tarpeelliseksi. Näitä prosesseja olivat muun muassa käyttäjätunnusten ja salasanojen toimitukseen liittyvät prosessit.

Toinen vaihe

Testlink -ohjelmiston siirtäminen it-palveluntarjoajan virtualisoitavaksi tapahtui ulkoistussopimuksen puitteissa ja sisälsi käytännössä tilauksen palvelusta ja määrityksen käytettävästä ohjelmistosta. Tässä tapauksessa toimitettiin kuvaus Testlink 1.7 -ohjelmistosta sekä käytössä olleesta asennuksesta tietokantadumppi, eli kopio tietokannasta. Palveluntarjoaja toimitti tämän jälkeen virtualisoituun palveluun URL-osoitteen ja ”admin”-käyttäjätunnuksen, jolla luotiin loput käyttäjätunnukset ja määriteltiin kullekin käyttäjätasolle tarvittavat oikeudet. Virtualisoinnin jälkeen varmennettiin verkkoyhteyksien toimivuus määrityksen mukaisesti yrittämällä yhteyttä sekä ulkoverkosta että Qentinelin sisäverkosta sekä ulkoverkosta VPN -yhteyden yli. Yhteyksien tuli toimia ulkoverkkoa lukuun ottamatta. Yhteydet ja ohjelmisto saatiin toimimaan suunnitellulla tavalla.

Tässä ratkaisussa oli edelleen puutteena Qentinelin toimistoverkon ulkopuolelta tulevien yhteyksien estyminen. Tätä palvelua varten pyydettiin tieto asiakkaan verkko-osoitteista niiden IP-osoitteiden osalta, joista oli tarkoitus käyttää palvelua. Tiedon IP-osoitteista saavuttua avattiin näiden perusteella portti toimistoverkkoa ja julkista Internet-verkkoa erottavaan palomuuriin siten, että toimistoverkon lisäksi palveluun pääsi vain näistä verkoista.



Kuva 4: Toisen vaiheen verkkotopologia

Lopullinen implementaatio

Toisessa vaiheessa muotoutunut ratkaisu osoittautui kyseisen projektin tarpeet tyydyttäväksi joten se pidettiin sellaisenaan projektin käytössä. Tästä saatiin suoraa käytönaikaista palautetta. Aktiivisten käyttäjien ei tarvinnut enää matkustaa asiakkaan tiloista Qentinelin toimistolle käyttämään palvelua.

Tämän ratkaisun käyttökokemuksista tehtiin kyselytutkimus, jossa pyrittiin selvittämään käyttäjien mielipiteitä ja kokemuksia, sekä selvittämään selkeitä parannusehdotuksia palvelun muokkaamiseksi mahdollisimman helpoksi ja edulliseksi sekä käyttää että ylläpitää kaikkien näkökulmasta katsottuna. Erityistarkastelussa suurimman kustannustehokkuuden saavuttaminen Qentinelin toiminnassa.

3.4 Kyselytutkimus

Kyselytutkimuksella pyrittiin selvittämään käyttäjien käytönaikaisia kokemuksia itse ohjelmistosta, että käytetyistä palveluprosesseista. Kyselytutkimus toteutettiin lähettämällä laadunhallintaprojektiin osallistuneelle yhdeksälle henkilölle kyselylomake, jossa oli yhdeksän kysymystä. Jokaiseen kysymykseen oli vakioidusti asetettu kiinteitä vastausvaihtoehtoja viisi kappaletta. Vastauksia saatiin annettuun määräaikaan mennessä kolme kappaletta. Lisäksi sähköpostitse

välitettynä kyselylomakkeen saatetekstiin oli lisätty kehotus kertoa kokemuksestaan tarkemmin ja jättää vastaamatta mikäli ei mielestään ollut käyttänyt palvelua. Tarkentavassa haastattelussa selvisi myös, että palvelua jossain määrin käyttäneitä oli kokonaisuudessaan noin viisi. Vastausprosentiksi muodostui siis noin 60 prosenttia kyselylomakkeen saaneista ja palvelua käyttäneistä. Käyttäjille lähetetyssä kyselylomakkeessa oli seuraavat kysymykset:

1. Oletko käyttänyt Testlinkiä projektin aikana
2. Testlink oli käytettävyydeltään hyvä
3. Testlink oli saavutettavissa/käytettävissä
4. Testlink vasteaika oli mielestäsi riittävä
5. Testlink:ssä esiintyi ohjelmallisia toimintahäiriöitä
6. Testlink suoriutui toiminnallisesti sille annetuista tehtävistä
7. Ongelmatilanteessa tiesit keneen ottaa yhteyttä
8. Käyttäisitkö kyseistä tai vastaavaa järjestelmää jatkossa mieluiten
9. Piditkö ylläpidettyä testauksenhallintapalvelua (TestLink & Qentinelin ylläpito) kokonaisuutena käyttökelpoisena ratkaisuna projektin testauksenhallintaan?

Lomake oli muotoiltu kyselylomakkeeksi siten, että vastaajan oli mahdollisimman helppo ensiksi hahmottaa kyselyn laajuus ja toiseksi vastata kysymyksiin nopeasti. Vastausvaihtoehtoina oli toiminnan toistuvuutta ja toistettavuutta kuvaava sanallinen skaala: harvoin, silloin tällöin, yleensä, lähes poikkeuksetta, aina. Vastaukset eivät ehkä ole kielellisesti luontevia vastauksia kysymyksiin, mutta niiden, kuten kysymyksien muotoilussakin, on pyritty helppoon tulkintaan ja tulosten analysointiin.

Vapaamuotoisesti vastaten olisi kyselylomakkeen täyttäminen vaatinut enemmän aikaa ja täten vieläkin suurempi osa olisi mitä ilmeisimmin jättänyt vastaamatta.

Vapaamuotoisissa vastauksissa olisi myös suurempi työ analysoinnissa ja tulkinnoissa, joskin tietoa olisi voinut saada jossain määrin enemmän. Saatteessa olleeseen kehotukseen vapaamuotoisesta kokemusten kertomisesta ei kukaan tarttunut. Täten analysoitavaksi saatiin yhteensä 27 vastausta.

3.4.1 Vastaukset yleisesti

Vastaukset järjestettiin taulukoksi, josta oli nähtävissä vastausten jakaantuminen kysymysten suhteessa. Taulukossa 1 on esitetty vastausten jakaantuminen.

Kysymys	Harvoin	Silloin tällöin	Yleensä	Lähes poikkeuksetta	Aina
1			x		xx
2		xx	x		
3			x	x	x
4				x	xx
5	xxx				
6		x	x	x	
7	x		x		x
8	x	x		x	
9	x		x		x

Taulukko 1: Kysymysten jakaantuminen

Kysymys 1: ”*Oletko käyttänyt Testlinkiä projektin aikana*”

Vastanneista 2 vastasi ”aina”. Kyselyyn vastanneista useammat siis katsovat käyttävänsä Testlinkiä päivittäin päivittäisessä työssään. Tämä kysymys on kalibrointi kysymys, jolla tarkoituksena on hakea kokemuseroille selitystä paljon käyttävien ja vähän käyttävien välille.

Väite 2: ”*Testlink oli käytettävyydeltään hyvä*”

Vastaukset ovat yhtenevän suuntaisia. Tämä ohjaa ajattelemaan, ettei ohjelma ole

käyttäjien mieleen toiminnallisen logiikkansa suhteen. Toisaalta kysymyksen johdattelevuus väittämällä ohjelmaa hyväksi saattoi ohjata vastaajat protestoimaan väitettä ehkä turhankin rankasti.

Väite 3: ”*Testlink oli saavutettavissa/käytettävissä*”

Kysymys mittaa käyttäjien kokemaa palvelun katkonaisuutta. Tässä oli hieman hajontaa havaittavissa, mutta kaikki vastaukset ovat kuitenkin skaalan positiivisessa päässä tai neutraalissa välissä. Tähän saattaa vaikuttaa se, että osa käyttäjistä osallistui palvelun rakentamiseen koekäyttäjinä, jolloin palvelu oli ajoittain yllättäen alhaalla tai saavuttamattomissa johtuen rakennustoimista. Osa käyttäjistä taas tuli mukaan vasta palvelun toimiessa jo IT-palvelutarjoajan virtualisointipalvelimelta ja täten melko tukevasti käyttäjien tavoitettavissa.

Väite 4: ”*Testlink vasteaika oli mielestäsi riittävä*”

Tällä tiedusteltiin käyttäjän kokemaa viivettä palvelun toiminnassa. Vastausten keskittyminen skaalan positiiviseen päähän antaa ymmärtää, että palvelu oli mitoitettu oikein tai ainakaan kukaan vastannut käyttäjä ei joutunut käyttämään palvelua hitaiden verkkoyhteyksien takaa. Toisaalta Testlink ei ohjelmistona ole ainakaan silmin nähden havaittavasti erityisen verkkokaistaa vaativa ohjelmisto. Tämä osoittaa myös ettei käytetty virtualisointiympäristö tai muu asiaan vaikuttava laitteisto ollut käyttäjien kokeman perusteella ylikuormitettu.

Väitteet 5 ja 6

Taulukosta 1 on nähtävissä, että vain yhteen kysymykseen kaikki vastanneet vastasivat samoin. Tämä oli kysymys numero 5 ”*Testlinkissä esiintyi ohjelmallisia toimintahäiriöitä*”, jotka vastanneet katsoivat yksimielisesti vastauksen ”harvoin” arvoiseksi. Suora johtopäätös tästä olisi, että valittu avoimenlähdekoodin ohjelmisto on laadultaan hyvä, eikä sen mahdollinen

oikkuilu aiheuta käyttäjän työhön ainakaan sanottavia katkoksia. Tätä vastaan vaikuttaa hieman omituiselta kysymyksen 6 ”*Testlink suoriutui toiminnallisesti sille annetuista tehtävistä*” vastausten hajaantuminen ja painottuminen skaalan keskelle. Kysymyksen asettelu antaa jossain määrin olettaa, että kysymyksessä kysyttäisiin samaa asiaa. Varsinainen tarkoitus tällä kysymysparilla oli tehdä eroa käytettävyyden, siis intuitiivisen ohjelman toiminnan, ja selkeiden ohjelmointivirheistä johtuvan toimintavirheen aiheuttamien ongelmien välille.

Väitteet 7, 8 ja 9

Kolmen viimeisen väitteen osalta ei yleisessä vastausten arvioinnissa voida katsoa olevan mieltä vastausten hajaantuessa täysin koko skaalan mitalta. Erityisesti tätä seikkaa pohditaan seuraavassa osassa, jossa puretaan vastauksia vastaajien aiempien vastauksien perusteella ryhmiteltyinä.

3.4.2 Vastaukset suhteellisesti

Kyselyä voi tulkita myös suhteessa siihen miten tietyllä tavalla tiettyyn kysymykseen vastannut vastaaja vastaa joihinkin muihin kysymyksiin. Tällä pyritään selvittämään vastausten suhteellista riippuvuutta. Onko jollain näkemyksellä vaikutusta vastaukseen jotain toista kohtaa käsittelevään kysymykseen. Vastausten pienestä määrästä johtuen yleispäteviä suuntauksia tai johtopäätöksiä ei voitane johtaa. Sen sijaan on mahdollista havaita käyttäjien vastauksissa linjauksia esimerkiksi suhteessa siihen kuinka paljon he palvelua käyttivät. Suuremman aineiston ollessa saatavilla tässä analysoitaisiin tietyllä perusteella jaoteltuja käyttäjäryhmiä.

Kysymys	Harvoin	Silloin tällöin	Yleensä	Lähes poikkeuksetta	Aina
1					x
2			x		
3				x	
4					x
5	x				
6				x	
7					x
8				x	
9					x

Taulukko 2: Käyttäjä X

Taulukossa 2 on esitetty käyttäjän X vastaukset esitettyihin kysymyksiin. Huomataan että hän käytti palvelua paljon ja vastaukset painottuvat yleisesti katsoen skaalan positiiviseen päähän. Poikkeuksen muodostaa kysymys 5, jossa siis väitettiin ohjelmassa usein esiintyvän ohjelmallisia toimintahäiriöitä. Tästä käyttäjä X on kaikkien muiden vastanneiden kesken samaa mieltä.

Kysymys	Harvoin	Silloin tällöin	Yleensä	Lähes poikkeuksetta	Aina
1			y		
2		y			
3			y		
4				y	
5	y				
6		y			
7			y		
8	y				
9			y		

Taulukko 3: Käyttäjä Y

Taulukoita 3 ja 4 tarkastellen havaitaan, että käyttäjien Y ja Z kokemukset ovat saman suuntaisia huolimatta käytön määrässä olevassa erosta. Ainoastaan vastaukset kysymyksiin 3 ja 4 näyttävät korreloivan. Kysymys 4 täysin ja kysymys 3 saman suuntaisesti. Kysymykset käsittelevät saavutettavuutta ja vasteaikaa. Tämä siis koetaan riittäväksi niiden parissa, jotka käyttivät palvelua paljon. Mikä johtaa ajattelemaan, että vähemmän käyttänyt käyttäjä on osunut

käyttämään palvelua hetkellisten häiriöiden aikana.

Kysymys	Harvoin	Silloin tällöin	Yleensä	Lähes poikkeuksetta	Aina
1					z
2		z			
3					z
4					z
5	z				
6			z		
7	z				
8		z			
9	z				

Taulukko 4: Käyttäjä Z

Muilta osin käyttäjien Y ja Z kokemukset vastaavat toisiaan huolimatta käytön määrässä olevassa erossa. Suurin ja räikein ero tässä suhteessa muodostuu käyttäjien X ja Z välille. Siinä missä X katsoo kysymyksen 9 perusteella vastaavaa ratkaisua kelvollisena aina samaan tarkoitukseen jatkossa, katsoo Z ettei ratkaisua tule käyttää kuin harvoin. Mahdollista on myös, että käyttäjät arvioivat oman käyttönsä määrän eritavalla ja ristiin.

Käyttäjien Z ja Y vastausten painottuminen enemmän skaalan negatiiviseen päähän tukihenkilöstöä ja jatkokäyttöä käsittelevissä kysymyksissä 7-9 viittaa ennen kaikkea kehittymättömiin prosesseihin, jotka olivat varsinkin palvelun alkuvaiheessa kehittymättömiä ja keskeneräisiä. Prosessien paranemiseen käytön aikana viittaa tässä suhteessa käyttäjän X vastausten näiltä osin erilainen suuntaus.

3.4.3 Tarkentava haastattelu

Kyselytutkimusaineiston vähäisyyden, vastauksien hajonnan, sekä osittaisten ristiriitaisuuksien ilmentyminen loi tarpeen suorittaa tarkentava haastattelu. Haastattelulla pyrittiin selventämään ennen kaikkea ristiriitaisuuksia sekä hajonnan syytä. Tarkentavaan haastatteluun tavoitettiin puhelimitse kaksi kolmesta alkuperäiseen kyselyyn vastanneista.

Käyttäjä Y

Käyttäjän Y vastaukset tarkentuivat eniten. Osoittautui että vastausskaalaa oli tulkittu hieman muista poikkeavasti sekä kysymyksiä ymmärretty osittain väärin. Käyttäjä Y kertoi käyttäneensä palvelua mielestään eniten koko ryhmästä, koska hän oli käyttänyt palvelua aina tehdessään töitä projektin parissa. Puhelinhaastattelussa selvisi, että käyttäjän Y vastaukset kysymyksiin 1 ja 3 olivat erheellisesti asetettu. Puhelinhaastattelussa käyttäjä Y kertoi oikeiden vastausten olevan kyseisiin kysymyksiin olevan ”Aina”. Tämä asettaa hänen kokemuksensa palvelusta suoraan linjaan muiden käyttäjien kokemusten kanssa ja täten mahdollistaa selkeämpiä tulkintoja kyselytutkimuksen aineistosta. Myös käyttäjältä Y saadut tarkentavat vastaukset olivat linjassa sekä käyttäjän Z vastausten, että alkuperäisen kyselyn tulosten perusteella tehtyjen alustavien linjausten kanssa.

Käyttäjä Z

Käyttäjä Z oli tulkinnut kysymykset kyselyn laatijan suunnittelemalla tavalla, joten hänen osaltaan vastaukset pysyivät samoina. Käyttäjän Z antamat tarkentavat vastaukset hajaantuneisiin kysymykseen toivat selkeyttä tulkintoihin ja mahdollistivat selkeämpien johtopäätösten tekemisen käsillä olevan aineiston perusteella.

Käyttäjien vastauksien tarkennukset

Taulukoista 3 ja 4 on nähtävissä, että vastaukset hajoavat eniten taulukoiden vasemmassa alaneljänneksessä. Tarkentavista vastauksista oli saatavissa tieto vastausten hajontaan sekä perustelut tähän. Käyttäjät Y ja Z kokivat puutteiksi samoja asioita, mutta ilmaisivat niitä kyselyyn eri tavoin. Molemmat ilmaisivat

perusteluina kysymysten 5 ja 6 vastauksiinsa seuraavia seikkoja:

- Ohjelma toimi virheellisesti harvoin.
- Yksittäisenä toiminnallisena virheenä esiin nousi liitetiedostojen lisäämisen toimimattomuus, Mutta muuten virheellinen toiminta ohjelmassa oli satunnaista.
- Ominaisuuksia oli selkeästi vähemmän kuin tarpeita.
- Testlink koettiin parhaaksi avoimen lähdekoodin testaushallinnan työkaluista, joita oli käytetty.

Kysymysten 7-9 osalta tarkentavissa vastauksissa huomion arvoista on, että molemmat käyttäjät vastasivat linjassa mielipiteenään, että käyttäjätukea ei ollut saatavilla silloin, kun sitä olisi tarvittu eikä ylläpitokaan ollut tavoitettavissa aina tarvittaessa. Vastanneet käyttäjät kokivat hallintaprosessit keskeneräisiksi ja puutteellisiksi sekä kaipasivat näihin täydennystä. Käyttäjä Y toi kuitenkin esille mielipiteensä, että pieneen ja nopeasti läpivedettävään laadunvarmistusprojektiin palvelu soveltui varsin hyvin. Tällöinkin tosin olisi tarpeen määrittää joitakin tukitoimien prosesseja paremmin ja valmiimmiksi.

3.4.4 Kyselyn loppupäätelmä

Pienestä aineistomäärästä johtuen ei suuria yleispäteviä linjauksia tämän perusteella voi tehdä. Kysely ja haastattelu pyrkivät selvittämään kuitenkin pääosin käytettävyyteen liittyviä seikkoja ja on todettu, että jo viiden käyttäjän otoksella saadaan käytettävyydestä 80 prosenttia saatavissa olevasta tiedosta [14]. Tässä suhteessa kyselyn tuloksia on siis pidettävä varmasti vähintään suuntaa antavina. On kuitenkin nähtävissä kehittymättömien prosessien vaikutus käyttäjän kokemukseen palvelusta, vaikka palvelu ohjelmallisesti toimisikin hyvin.

Käyttäjien kokemukseen vaikuttaa myös tekninen toteutus. Tämän takia on

käytettävä erityistä huolta valittaessa ohjelmistoa, jota palvelussa tarjotaan. Tämä valinta on suhteutettava sekä tarpeeseen, että sen todellisiin loppukäyttäjiin.

Puhelinhaastatteluista saadun tiedon perusteella päätelmiä voidaan tarkentaa. Nähtävissä on kolme päätelmää, jotka voidaan esittää ohjeen muodossa:

1. Käytettävä testaushallintaohjelmisto on valittava siten, että muodostuvalle palvelulle on saatavilla ohjelmiston tunteva tukihenkilö.
2. Ohjelmisto on valittava siten, että se toimivien toiminnallisuuksiensa osalta vastaa kulloisenkin laadunvarmistusprojektin tarpeita.
3. Käytönaikaisen tuen sekä ylläpidon prosessit on määriteltävä yksiselitteisesti ja määrättävä näistä prosesseista vastuussa olevat henkilöt.

Suoraan SaaS-malliin liittyviä kysymyksiä ei kyselylomakkeeseen tarkoituksellisesti laitettu, muuten kuin yleisluontoisena kysymyksenä vastaavan palvelujärjestelyn käytöstä myöhemmin. Tämä on perusteltua koska loppukäyttäjä ei näe kovinkaan paljon taustalla suoritettuvia toimintoja ja näin ollen SaaS-malliin liittyvien kysymysten vastaukset olisivat olleet hyvin satunnaisia.

4 POHDINTA, JOHTOPÄÄTÖKSET JA YHTEENVETO

Testaus tai laadunvarmistus on monipuolinen ja laaja ohjelmistokehityksen toimi. Laadunvarmistuksen soisi ulottuvan kaikkialle ja kaikkiin ohjelmistokehitysprosessien vaiheisiin. Usein näin ei kuitenkaan ole, vaan riippumaton laadunvarmistus soitetaan paikalle sammuttamaan jo syttynyttä tulipaloa. Tilatun ohjelmiston kehitystyö saattaa olla myöhässä, sovitusta laadusta on erimielisyyksiä ja koko projekti pahasti ylittää budjetin. Tällaisessa erityistilanteessa laadunvarmistuksella on oltava käytössään luotettavat ja helppokäyttöiset työkalut ja ennen kaikkea niiden on oltava saatavilla ja käytettävissä mahdollisimman nopealla aikataululla. Onneksi tilanne ei kuitenkaan aina ole pahimmasta päästä, vaan laadunhallintaprojektia ehditään suunnitella ajallaan ja yhdessä muun ohjelmistoprojektisuunnittelun kanssa, parhaassa tapauksessa jopa aivan ohjelmistoprojektisuunnittelun alusta lähtien. Tämän toteuttamiseen tarvitaan selkeitä toimintamalleja, etukäteen mietittyjä prosesseja sekä tehtäviinsä harjaantunutta henkilökuntaa.

Tässä työssä tarkoituksena oli tutkia ohjelmistojen tarjoamista palveluna SaaS-mallin mukaisesti, siihen liittyviä ongelmia, etuja, riskejä sekä mahdollisuuksia. Ohjelmiston tarjoaminen palveluna on nähtävissä monipuolisena mahdollisuutena, joka ei ole tiukasti sidottuna tiettyyn tekniikkaan tai prosessiin. Tiettyjä asioita on kuitenkin hyvä ottaa huomioon ja soveltaa omaan tarkoitukseen parhaiten soveltuvalla tavalla. Huomioitavia asioita SaaS-mallin näkökulmasta ovat implementaatiotasot ja kypsyysaste. Käyttöön valittava ohjelmisto on puolestaan mietittävä tietoturvan, kehitystyön, muokattavuuden sekä ominaisuuksien perusteella. Eikä lopullisten loppukäyttäjien näkemyksiä pidä vähätellä.

4.1 Johtopäätökset

Testaushallintapalvelun pystyttäminen ja tarjoaminen laadunvarmistusprojektin käyttöön ei ole erityisen vaikea tehtävä. Se voi olla monimutkainen ja monipuolinen, mutta pohjimmiltaan kyseessä on suhteellisen yksinkertaiseen ratkaisuun perustuva palvelun toteutus, jota tehdessä on kiinnitettävä huomiota yhteen jos toiseenkin seikkaan.

Saatavilla olevan tiedon valossa on pantava merkille ja pohdittava useita seikkoja rakennettaessa SaaS-malliin testaushallintapalvelua. Näitä seikkoja pyrittiin selvittämään tämän työn alussa esitettyjen tutkimuskysymysten kautta. Kysymysten osalta päädyttiin seuraaviin johtopäätökseen:

- **Mitä tarkoitetaan SaaS-mallilla ja mitä se mahdollistaa testaushallintapalvelun tarjoamisen, ylläpidon ja pienen yrityksen kannalta?**

  SaaS-mallilla tarkoitetaan ennen kaikkea tapaa, jolla ohjelmisto tuotetaan palveluksi. SaaS terminä kattaa laajan alan ohjelmistopalveluita ja näitä nimitetäänkin SaaS:in eri tasoiksi. Tasot kuvaavat palvelun hienostuneisuutta ja kypsyysastetta. Korkeimman tason SaaS-implementaatiot ovat käyttötarpeen mukaan konfiguroituvia ohjelmistoja, kun alimmat tasot ovat käytännössä tiettyyn erikoistarkoitukseen luotuja Web-palveluja. SaaS-malli tarjoaa pienelle rajoitetulla budjetilla toimivalle yritykselle valmiin raamin ohjelmistopalvelun tarjoamiseen, jolloin pitkälle valmiiksi suunniteltujen käyttötapausten ja prosessien avulla voidaan haluttu palvelu tuottaa kustannustehokkaasti.

- **Miten avoimen lähdekoodin kehitysmallit vaikuttavat ohjelmiston loppukäyttöön, ylläpitoon ja tietoturvaan testaushallintapalvelun näkökulmasta?**

  Avoimen lähdekoodin ohjelmistoissa on perustavan laatuisena ongelmana vastuullisten puuttuminen kokonaan. Tämä ei ohjelmallinen seikka

vaikuttaa taustalla kaikkeen yritysmaailmassa käytettävissä avoimen lähdekoodin ohjelmistoissa. Näin on myös testaushallinnan ohjelmistojen kohdalla, joiden laadun arviointiin vaikuttaa olennaisesti käyttäjien varsinainen tehtävä laadunvalvojina. Avoin lähdekoodi on sekä riski että mahdollisuus. Riski siksi, ettei esimerkiksi tietoturvaan, jatkokehitykseen tai käyttötukeen ole saatavilla yksiselitteisiä vastauksia vaan pelkästään pohdintoja ja suosituksia. Näin ollen osallistuminen kehitykseen antaa avoimen lähdekoodiin ohjelmistoa hyödyntävälle yritykselle mahdollisuuden suhteellisen pienellä omalla panoksella vaikuttaa suurestikin omien loppukäyttäjiensä käyttökokemukseen. [5]

- **Testaushallintaohjelmisto on Web-sovellus, mutta miten testaus- ja laadunvarmistustoimien erikoisluonne on mahdollisesti huomioitava?**

  Kyselytutkimuksen ja sen tarkennushaastattelujen perusteella voidaan sanoa, että ohjelmiston valinta kohdennettuna oikeaan käyttötarkoitukseen on tehtävä oikein. Ohjelmassa saa tässä suhteessa olla toiminnallisia virheitäkin, kunhan etukäteen on pidetty huolta että tarvittavat ominaisuudet toimivat.

- **Mitä on huomioitava että vastuunjako olisi mahdollisimman selkeä ja toiminta sujuvaa minimoiden asiakasorganisaatiolle näkyvät palvelukatkokset sekä palvelun tarjoajan kustannukset?**

  Vastuunjaon suhteen tärkeintä ovat oikein ja selkeästi määritellyt ohjeet ja prosessit mielellään kaikkiin kuviteltavissa oleviin tilanteisiin. Kyselyiden perusteella on sanottava, että hyvätkin lähtökohdat omaava laadunvarmistusprojekti saadaan tuntumaan työntekijöistään epämieluisalta, jos he eivät voi luottaa käytössään olevien työkalujensa toimintavalmiuteen. Tämän toimintavalmiuden ylläpidossa selkeät ja tarkkaan määritellyt prosessit jakavat vastuuta ja helpottavat näin varsinaisen laadunvarmistustyön tekemistä.

4.2 Yhteenveto

Testaushallintapalvelu nykyaikaisessa laadunvarmistuspalveluliiketoiminnassa on palvelu, jossa yksi taho määrittelee, toinen tuottaa ja kolmas käyttää palvelua. Tämä jo yksissään vaati prosessien määrittelyä ja käytäntöjen selkeyttämistä. Kun testaushallintapalvelua lähdetään toteuttamaan avoimen lähdekoodin ohjelmiston pohjalta on huomioitava vielä lisäksi itse käytettävissä olevan ohjelmiston kehitysmallin riskit sekä edut.

Palveluntarjontamalli asettaa valmiit raamit toiminnan perustalle, mutta jättää paljon lopullisesta toteutuksesta palveluntuottajan itsensä vastattavaksi. SaaS-malli tuo mukaan tasot, jotka auttavat arvioimaan oman tuotetun palvelun vaatimuksia sekä etuja oikeassa suhteessa.

Qentinelin tapauksessa rakennettiin kahdessa vaiheessa SaaS-tasolle yksi tai kaksi yltävä testaushallintapalvelu. Palvelun rakentamisen vaiheita tutkimalla pyrittiin hahmottamaan paras toteutus, niin verkkoratkaisuiden kuin palveluvastuiden kannalta. Ensimmäisessä vaiheessa rakennettiin palvelu toimiston sisäverkkoon, jossa todettiin verkkoyhteyksistä riippuva saavutettavuusongelma. Tämä ongelma ratkaistiin siirtämällä palvelu ulkoistetun palvelutarjoajan haltuun ja määrittämällä saavutettavuus toimistoverkosta riippumattomalla tavalla.

Palvelun käyttöä tutkimalla selvitettiin käyttäjien kokemuksia ja mieltymyksiä palvelun käytöstä. Tutkimus suoritettiin kyselytutkimuksena, jota laajennettiin ja tarkennettiin kohdennetulla käyttäjien puhelinhaastatteluilla. Puhelinhaastattelujen myötä kyselytutkimuksen tulokset tarkentuivat ja ilmenneeseen vastausten hajontaan löytyi selitys kysymysten vaihtelevista tulkinnoista kokemusten ollessa kuitenkin kaikilla vastanneilla saman suuntaisia.

Yhteenvetona tutkimuksesta on voidaan sanoa, että paras tulos saavutetaan määrittelemällä vastuut, käytännöt ja prosessit mahdollisimman pitkälle etukäteen. Tällä tavoin luodaan käyttäjille helposti kokemus toimivasta palvelusta vaikka käytettävässä ohjelmistossa olisikin puutteita. Ohjelmistokin on toki valittava siten että täyttää tarkoituksensa ja toimii mahdollisimman moitteetta niiltä osin joita kulloisessakin laadunvarmistusprojektissa tarvitaan.

LÄHDELUETTELO

[1] Baida, Z., Akkermans, H., Gordijn, J., ”Service Classification versus Configuration”, Free University Amsterdam, 2005

[2] Rathmell, John M., ”What is meant by services?”, Journal of marketing, October, 1966

[3] Waters, B., ”Software as a service: A look at the customer benefits”, Journal of digital asset management, Vol. 1, 2005

[4] Nitu, ”Configurability in SaaS (Software as a Service) Applications”, ISSA/DRDO, Metcalfe House, Delhi, India, 2009

[5] Sang-Yong T.L., Hee-Woong K., Sumeet G., ”Measuring open source software success”, Omega-international journal of management science, April 2007

[6] Capra, E., Francalanci C., Merlo, F., ”An Empirical Study on the Relationship among Software Design Quality, Development Effort, and Governance in Open Source Projects”,IEEE Transactions On Software Engineering, Vol. 34, No. 6, November/December 2008

[7] Hoepman, J-H., Jacobs, B., ”Increased Security Through Open Source”, Communications of the ACM, Vol. 50, No. 1, January 2007

[8] Witten, B., Landwehr, C., Caloyannides, M., ”Does Open Source Improve

System Security?", DARPA, Mitretek Systems, IEEE SOFTWARE, September / October, 2001

[9] Toroi, Tanja, "Testing Component-Based Systems – Towards Conformance Testing and Better Interoperability", University of Kuopio, Doctoral dissertation, ISBN 978-951-781-994-7, sivut 25-33, 2009

[10] Hass, Anne, Mette, Jonassen, "Guide to Advanced Software Testing", ISBN-13: 978-1-59693-285-2, ARTECH HOUSE, INC., 2008

[11] Taipale, Ossi, "Observations on software testing practice", Lappeenrannan teknillinen yliopisto, Thesis for the degree of Doctor of Science (Technology), ISBN 972-952-214-428-7, sivut 33-34, 2007

[12] Eisenhardt, Kathleen M. "Building Theories from Case Study Research", Stanford University, Academy of Management Review, vol 14, No. 4, 532-550, 1989

[13] Chong, Frederick, Carraro, Gianpaolo, "Building Distributed Applications - Architecture Strategies for Catching the Long Tail", Microsoft Corporation, viitattu 20.5.2010, Saatavissa: http://msdn.microsoft.com/en-us/library/aa479069.aspx#docume_topic6, April 2006

[14] Virzi, R, A, "Refining the test phase of usability evaluation: how many subjects is enough?", Hum. Factors 34, No. 4, 457-468, August 1992