SaaS, multi-tenant ja vahinkovakuutusohjelmiston kehitys

Pekka Vanninen

Helsinki 30.04.2013

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta – Fakultet –Faculty	Laitos – Institution – Department
Matemaattis-luonnontieteellinen tiedekunta	Tietojenkäsittelytieteen laitos

Tekijä – Författare – Author

Pekka Vanninen

Työn nimi – Arbetets titel – Title

SaaS, multi-tenant ja vahinkovakuutusohjelmiston kehitys

Oppiaine – Läroämne – Subject

Tietojenkäsittelytiede

Työn laji – Arbetets art – Level	Aika – Datum – Month and year	Sivumäärä – Sidoantal – Number of pages
	20.3.2013	

Tiivistelmä – Referat – Abstract

Tämä kirjoitus kuvaa yritysohjelmistojen toimittamisen ja käytön muutosta ohjelmistopalvelun tarjoamiseksi verkon välityksellä. Kirjoituksessa esitellään multi-tenant asiakkuusmalli ohjelmistoille ja tarkastellaan mahdollisia tapoja olemassa olevan ohjelmiston muuttamiseksi SaaS-mallin mukaiseksi. Tarkastelun kohteena on erityisesti yrityskäyttöön toteutetun asiakas-palvelin arkkitehtuurin mukaiseen Windows-työasemaohjelmistoon tarvittavat muutokset ja niiden hyötyjen arviointi asiakkaan arkkitehtuuripäätöksen tueksi. Muutoksen toteutus tietokantaohjelmistotoimittajan ja ohjelmistotyökalutoimittajan työvälineillä on tarkastelun ja arvioinnin kohteena.

Avainsanat – Nyckelord – Keywords

ASP, SaaS, Multi-tenant, Cloud Computing

Säilytyspaikka – Förvaringställe – Where deposited

Muita tietoja – Övriga uppgifter – Additional information

1 Johdanto

Yritysohjelmistojen käyttö on siirtynyt enenevässä määrin internetissä tapahtuvaksi. Internet ja yhteysnopeuksien kasvaminen ovat toimineet muutoksen ajurina, kun ohjelmistojen käyttökokemus on saatu riittävän hyväksi verkkoyhteyden yli tapahtuvana. Siirtymä on tapahtunut myös yksityiskäytössä, kun kuluttajat käyttävät verkossa olevia ohjelmistoja päivittäisten asioiden hoitamiseen ja yritykset ja yhteisöt tarjoavat tarkoitukseen sopivia ohjelmistoja käyttöön selaimella käytettäväksi. Tämä kehitys on johtanut myös pienille yrityksille ja yhteisöille suunnattujen web-sovellusten tuloon markkinoille. Tyypillisesti sovellukset ovat ilmaisia tai tapahtuma- tai kuukausimaksupohjaisesti hinnoiteltuja ja toteuttavat jonkin yksinkertaisen toiminnallisuuden intuitiivisella tavalla.

Perinteistä asiakas-palvelin-arkkitehtuuria noudattavia yritysohjelmistoja on myös siirretty toimittajien omiin tai valitsemiin palvelukeskuksiin. Ohjelmistojen päivittäinen käyttö tapahtuu internet yhteyden avulla. Valtaosa nykyisistä yritysten operatiivisista ohjelmistoista on siirtynyt pois yritysten omilta palvelimilta ohjelmistotoimittajan tai kolmannen osapuolen konesaleihin. Muutoksen syinä ovat yritysten ydinliiketoimintaan keskittyminen, internetin mahdollisuudet ohjelmistopalvelujen siirtoon sekä sovellusten toimitukseen liittyvien kustannusten vähentäminen, kun sovelluksen käyttäjäasiakkaan ei tarvitse hankkia palvelinkapasiteettia ja osaamista ohjelmistojen ja järjestelmien ylläpitoa varten. Syynä ovat myös ohjelmistojen kypsyminen ja sitä kautta ohjelmistojen lisensiointimallit, jotka ovat siirtäneet ohjelmistojen omistuksen ohjelmistotoimittajalle. Käyttäjäasiakkaan itse tekemä ohjelmiston sovitustyö, joka oli yleistä vielä 1990-luvun alussa, on jäänyt pois ja asiakkaiden organisaatioita on karsittu ydinliiketoiminnan tehostamiseksi. Kun ohjelmistotuotteet tai tuotteiden integraation kautta syntynyt kokonaisuus ovat kasvaneet kattamaan koko yritystoiminnan, niin yksittäiselle asiakkaalle tehtävä sovitustyö on muuttunut entistä enemmän konfiguroinniksi ja parametroinniksi ja asiakkaalle erikseen tehtävän ohjelmointityön osuus toimitusprojektin työstä ja kustannuksista on vähentynyt.

Näissä ohjelmistoissa käyttöliittymä on usein toteutettu toimimaan Windows-työasemassa. Tällaisen sovelluksen suoraviivainen siirto internet-yhteyden yli käytettäväksi on tehty terminaalipalvelinratkaisuilla, jotka siirtävät käyttöliittymän kuvan ja käyttöliittymätoiminnot tietoliikenneyhteyden yli sovelluspalvelimelle, joka on yhtey-

dessä ohjelmiston käyttämään tietokantaan. Näin on voitu toteuttaa ympäristöjä, joissa monta sovelluspalvelinta palvelee käyttäjiä ja käyttää yhtä tietokantapalvelinta ohjelmiston käytön yhteydessä (farmi sovelluspalvelimista). Tällaista toimintamallia on kutsuttu ASP-malliksi (Application Service Provider) ja malli on mahdollistanut uusia hinnoittelumalleja, joissa ohjelmiston hankintakustannus jaetaan ohjelmiston käytön ajalle kiinteisiin kuukausieriin. Tästä terminologiasta ollaan siirtymässä eteenpäin, kun sovellusten tarjoaminen on siirtymässä internetin yli tapahtuvaksi normitapauksissa. ASP – SaaS ero on määritelty seuraavaksi: SaaS ohjelmisto on vakio-ohjelmisto, jossa on mahdollisuus muunteluun konfiguroinnilla, kun taas ASP on asiakaskohtainen toteutus, jossa on tehty koodimuutoksia asiakkaan haluaman toiminnallisuuden saavuttamiseksi.

Ohjelmistojen tarjoamista asiakkaan käyttöön internet-yhteyden yli kutsutaan yleisesti SaaS –malliksi (Software as a Service). Malli mahdollistaa ohjelmiston käyttöönoton ilman asiakkaan mittavia investointeja tekniseen infrastruktuuriin (kantapalvelin, sovelluspalvelin, tarvittavat muut teknisen ympäristön laitteet ohjelmiston käyttöönottoa varten). Ohjelmiston toimittajalle se antaa mahdollisuuden hyödyntää konesalista löytyvää palvelinkapasiteettia ja ohjelmistoon tehtyä kehitystyötä uusien asiakkaiden ohjelmiston käyttöönoton yhteydessä. Ohjelmiston kohderyhmästä ja ohjelmiston toimintojen ja kohderyhmän ohjelmiston tuella toteutettavien toimintaprosessien kompleksisuudesta riippuu, missä määrin ohjelmiston toimittaja voi hyötyä olemassa olevasta ohjelmistosta. Suuri merkitys on myös ohjelmiston tarjoamilla muuntelutekijöillä ja ohjelmiston sovitettavuudella asiakkaan prosessien toteuttamiseksi. Jos sovitettavuus ei ole riittävää, ohjelmiston käyttö muuttuu liiketoimintaprosessin tukemisesta pelkästään tietojen tallennuskohteeksi ja ohjelmiston tuottama hyöty käyttäjälle jää saavuttamatta.

Multi-tenancy on saman ohjelmiston käyttöä usean asiakkaan toimesta. Pyrkimyksenä on, että asiakkaat voivat käyttää ohjelmistoa niin, että asiakkaan omat tiedot pysyvät vain asiakkaan tiedossa ja yhden asiakkaan ohjelmiston käyttö ei vaikuta muiden asiakkaiden ohjelmiston käyttöön. Multi-tenancy-toteutuksia on usean tasoisia, ja toteutuksen rakenne määrittää hyödyn ohjelmistotoimittajalle. Mitä vähemmän ohjelmiston ylläpito vaatii henkilöresursseja ja palvelinkapasiteettia asiakkaiden liiketoimintaprosessien tukemiseksi, sitä parempi ratkaisu on toimittajan kannalta. Kilpailutilanteessa toimittajan kustannustehokkuus ohjelmistopalvelun tarjoamisessa näkyy suoraan asiakkaalle edullisempana hinnoitteluna.

Pilvi käsitteenä kuvaa rajaamatonta joukkoa resursseja, jota on saatavissa internetin

välityksellä. Pilvi tarjoaa tallennus- ja prosessointikapasiteettia jatkuvaan ja tilapäiseen käyttöön. Pilven palvelujen hyödyntämisen yhteydessä on kiinnitettävä huomio käytön turvallisuuteen ja tietojen suojaamiseen sekä palvelutasovaatimuksiin ja –sopimuksiin. SaaS ohjelmistot tarjotaan pilvipalveluna. Pilvi voidaan luokitella yksityiseksi, suojatuksi tai julkiseksi. Näin voidaan ohjelmistotoimittajan tai palvelinkapasiteettitoimittajan konesali luokitella yksityiseksi tai suojatuksi pilveksi. Julkisten pilvien käyttö ja niiden käyttöyhteyksien rajaaminen ja käyttäjätunnusten ja käyttöoikeuksien hallinta rooli- ja henkilökohtaisesti asiakkaittain on otettava tarkasteluun.

Tässä työssä erityisenä tarkastelun kohteena on asiakas-palvelin-arkkitehtuuria noudattava vahinkovakuutusohjelmisto. Ohjelmistoa on kehitetty yli kahden vuosikymmenen aikana käytettäväksi vahinkovakuutuskäsittelyssä. Ohjelmistosta on rinnakkaisversioita, jotka ovat käytössä muilla vakuutusyhtiöillä. Näiden ohjelmistoversioiden kehitys on erikseen tapahtuvaa ja keskitettyä versionhallintajärjestelmää ei ole käytössä. Ohjelmistoa on toimitettu ASP-mallilla niin, että jokaisella asiakkaalla on oma sovellus ja tietokanta palveluntarjoajan kontrolloimassa konesalissa. Ohjelmiston käyttäjäyritys on viime vuonna fuusioitunut toisen vakuutusyhtiön kanssa ja yritysjärjestelyn tuloksena syntyneessä yhtiössä ohjelmisto on käytössä olleiden ohjelmistojen arvioinnin yhteydessä valittu myös syntyneen yrityksen vahinkovakuutusohjelmistoksi. Vakuutusyhtiössä on myös muita ohjelmistoja, joiden kehitykseen ja ylläpitoon on perustettu erillinen ohjelmistokehityspalveluja tarjoava yritys. Samalla kun vakuutusyhtiön organisaatiossa on käynnissä muutoksia fuusion vuoksi, on myös ohjelmiston kehityspuolella tarpeita, jotka tulee toteuttaa. Tässä yhteydessä tarkastellaan myös valittujen ohjelmistojen kokonaisarkkitehtuuria ja huomioida multi-tenant-ajattelu ohjelmistojen tulevan kehityksen osana. Kehityksen yhteydessä tehtävät arkkitehtuurivalinnat vaikuttavat merkittävästi tuleviin kehitysmahdollisuuksiin ja kustannuksiin sekä asiakkaan mahdollisuuteen hyödyntää ohjelmistoa prosessien tukena.

Erityispiirteenä mahdolliseen multi-tenancy-käsittelyyn asiakkaalla on paikallisyhdistysrakenne, jossa peruskäyttäjä kuuluu paikallisyhdistykseen (asiakas), mutta käyttäjällä tulisi olla pääsy myös niin sanottuun palvelukantaan, jossa on kaikkien asiakkaiden vakuutustiedot. Käyttäjällä tulisi siis olla mahdollisuus nähdä yleistä tietoa, mutta esimerkiksi vakuutustarjoukset pitäisi olla piilossa, jotta paikallisyhdistykset eivät saisi naapurin tarjouksia omiensa pohjaksi.

Tämän työn luvussa 2 käydään ohjelmistojen kehityksen historiaa läpi. Ohjelmistojen

käyttöikä on pidentynyt ja ohjelmistojen uudelleenkirjoittaminen on haastavaa ohjelmistoon kertyneen hiljaisen tiedon vuoksi ja arkkitehtuurien ikääntyessä. Luvussa 3 esitellään pilvipalveluihin liittyvää termistöä ja päästään SaaS määritelmän kautta lukuun 4 ja SaaS vaatimuksiin (multi-tenancy ja isolaatio ja varioitavuus) eri tasoilla. Luvun 5 sisältönä on Progress ohjelmistokehitysvälineet ja erityisesti tietokantaratkaisun sisäänrakennetun multi-tenancy käsittelyn esittely, sen tarjoamat mahdollisuudet ja tuki sovellusten SaaS-toteutuksiin. Luvussa käsitellään myös kehitysohjelmistotoimittajan tarjoamat välineet muutoksen toteuttamiseksi sekä muutoksen toteuttaminen vaihe vaiheelta. Luvussa 6 esitellään yrityksen käyttämän vahinkovakuutusohjelmiston rakenne ja toiminnallisuus, ja pyritään arvioimaan ohjelmiston multi-tenancy-kannan hyödyntämismahdollisuudet ja multi-tenancyn mahdolliset ongelmat toteutuksessa. Luku 7 on yhteenveto alueesta ja johtopäätökset, kannattaako ohjelmiston muutosprojektiin lähteä ja mitä pitää ottaa huomioon projektissa.

2 Yritysohjelmistojen kehitys nykymalliin

Luvussa tarkastellaan, miten ohjelmistojen toteutus ja toimitus on muuttunut ja miten tämä historia vaikuttaa nykyisiin ohjelmistototeutuksiin.

2.1 Yritysohjelmistot asiakkaalla

Yritysohjelmistojen toimitukset asiakkaan käyttöön on aiemmin tehty asiakkaan tiloihin ja asiakkaan palvelimelle. Tähän syynä on ollut erityisesti tietoliikenteen aiheuttama pullonkaula ohjelmistojen käyttämiseen, kun lähiverkko- ja internet-yhteydet ovat olleet hitaita tai niitä ei ole ollut lainkaan. Toisaalta ohjelmistot olivat tyypillisesti asiakkaan vaatimusten mukaan toteutettuja asiakkaan omistamia ohjelmistoja. Varhaisimmat ohjelmistot olivat merkkipohjaisia keskuskoneilla tai terminaaleilla ajettavia ohjelmistoja, joissa tiedon prosessointi tapahtui palvelinkoneella ja käyttöliittymä välitettiin terminaalille tai terminaalia emuloivalle työasemaohjelmalle. Kun tietoliikenneyhteydet kehittyivät, ohjelmistot monimutkaistuivat ja käyttäjämäärät kasvoivat, muodostui palvelimen suorituskyky ongelmaksi ja haluttiin siirtää osa prosessoinnista työasemalle, jossa oli käytettävissä olevaa prosessointikapasiteettia. Tämä johti asiakas-palvelin-arkkitehtuuriin, jossa ohjelmisto asennettiin työasemalle ja palvelimelta haettiin data, jota voitiin työasemalla prosessoida. Ohjelmistojen ominaisuuksien kehittyessä ja käyttäjien yhteyksien lisääntyessä tapahtui taas muutos toimintaympäristössä, kun osa käyt-

täjistä oli pitkän tai hitaan tietoliikenneyhteyden päässä. Tällöin prosessointia haluttiin taas siirtää takaisin sovelluspalvelimelle. Tässä yhteydessä syntyi joukko työkaluja, joilla voidaan pääte-emulointityyppisesti siirtää käyttöliittymä ja siihen liittyvät toiminnallisuudet verkon yli (Windows Terminal Server, Citrix, isojen ohjelmistotoimittajien omat ratkaisut). Käytännössä tämä oli paluuta vanhaan terminaaliajatteluun, lisäyksenä kuitenkin työasemalle paikallisten verkon kautta saatavien palvelujen tuki (kirjoittimet ja muu vastaava paikallinen toiminnallisuus). Kun tähän lisättiin yhteyden muodostaminen julkisen internet yhteyden yli, niin mahdollistettiin ohjelmiston etäkäyttö monessa toimipisteessä ilman erillistä tietoliikenneyhteyttä toimipisteiden välillä. Tämän mallin hyödyntäminen niin, että siirretään sovelluspalvelimet kokonaan asiakkaalta pois ohjelmistotoimittajan tai laiteympäristötoimittajan konesaliin toi toimittajille mahdolliseksi ASP-mallin (Application Service Provider). Näin ohjelmiston käyttöönottoon ei enää välttämättä tarvittu investointeja omaan laiteympäristöön, vaan ohjelmistotoimittaja varasi tarvittavat resurssit palvelinkeskuksesta ja kustannukset voitiin jakaa ohjelmiston käytön ajalle ja työasemien resurssivaatimukset olivat pieniä.

2.2 Yritysohjelmistojen siirto web-ympäristöön ajettavaksi

Kun internet tarjosi riittävän nopean ja suorituskykyisen yhteyden liiketoimintasovellusten käyttöön, oli luonnollinen ajatus rakentaa sovelluksen toiminnallisuuden päälle uusi web-selaimessa toimiva käyttöliittymä, jota voidaan käyttää HTTP-yhteyden yli. Tähän tarkoitukseen kehitettiin myös sovelluskehyksiä ja ohjelmointityökaluja. Tyypillisesti sovelluksen tietokantayhteyttä vaativat osiot irrotettiin käyttöliittymätoiminnoista niin, että ohjelmiston osa, joka siirrettiin asiakkaan työasemalle suoritettavaksi, ei tarvinnut jatkuvaa tietokantayhteyttä. Tietokantakäsittely ja tiedon prosessointi hoidettiin sovelluspalvelimilla, jotka huolehtivat istuntokohtaisesti sovelluksen tilasta ja vastasivat työasemalta tulleisiin pyyntöihin halutulla toiminnallisuudella. Usein laajojen ohjelmistojen kohdalla web-käyttöliittymätoteutuksia rajattiin vain ohjelmiston käytetyimpiin tai kriittisimpiin osiin, jolloin sovelluksen jatkokehitystä jouduttiin tekemään useampaa eri käyttöliittymäratkaisua varten. Työmäärään vaikutti kasvattavasti myös asiakas-palvelin mallin mukaisesti toteutettujen ohjelmistojen rakenne, jossa käyttöliittymäkerrosta ei ollut eriytetty sovelluslogiikkakerroksesta. Tämä vaati joko ohjelmiston rakenteen uudelleensuunnittelua ja toteutusta tai toiminnallisuuden monistamista eri käyttöliittymätoteutuksiin, vaikeuttaen ohjelmiston jatkokehitystä ja ylläpitoa. Vaihtoehtoiset lähes-

tymistavat olivat selaimessa ajettavat HTML-sovellukset ja ohuet asiakasohjelmistot, jotka voitiin ladata webin yli käytettäväksi työasemalla.

2.3 Web ohjelmistojen kehitys

Kun internetin käyttö yleistyi, alkoi syntyä erilaisia sivustoja, jotka tarjosivat yksityishenkilöille palveluja. Tyypillisiä palveluja ovat esim. uutissivut, yritysten kotisivut, webkaupat tai keskustelupalstat. Nämä oli usein suunnattu yksityishenkilöille. Jotta yksityishenkilöt voisivat säilyttää omia asetuksiaan ja palata omien tietojensa käsittelyyn, käyttäjätunnistus oli keino saada tallennettua tarvittavat tiedot. Käyttäjätunnistuksen kautta ohjelmisto osasi päätellä, mitä käyttäjälle voidaan tai halutaan näyttää ja miten tietoja pitäisi säilyttää. Internetiin syntyi myös yrityksille suunniteltuja palveluja, joiden avulla yritykset voivat seurata yhteistyökumppanien ja palveluntarjoajien tilannetta ja tehdä esimerkiksi tilauksia toisilta yrityksiltä. Yritystenvälisten ohjelmistojen kehitystä jarrutti kuitenkin se, että suuremmilla yrityksillä oli usein jo ohjelmistot, joihin tapahtumia kirjattiin sisäisesti ja web-sovellusten käyttö myös ulkoisiin järjestelmiin olisi ollut tuplatyötä. Näin web-sovellukset jäivät pienten yritysten käytettäväksi ja niiden toiminnallisuus jäi yksinkertaiseksi ja perusprosesseja toteuttavaksi. Web sovelluksia yrityksille kuitenkin syntyi ja osa niistä on saavuttanut merkittävän suosion myös suurten yritysten käytössä. (SalesForce.com ym.).

2.4 Uudet ohjelmistohankkeet

Uusien ohjelmistojen kehittämiseen nykyiset sovelluskehykset ja ohjelmistokehitysvälineet antavat mahdollisuuden SaaS-mallin hyödyntämiseen. Uuden ohjelmiston kehityksessä on mahdollista ottaa arkkitehtuurisuunnittelussa huomioon toteutus SaaS-ympäristöön. Tällöin SaaSin tunnistetut ominaispiirteet ovat toteutettavissa suoraan uuteen ohjelmistoon. Kaikkia asiakkaiden ohjelmiston käytön eriyttämiseen ja muunneltavuuteen liittyviä ongelmia ei ole kuitenkaan ratkaistu, vaan toteutuksen yhteydessä on huomioitava mahdollisesti asiakkaittain vaihtelevat palvelutasovaatimukset ja -sopimukset huolellisesti, jotta eriytyvät vaatimukset eivät aiheuta liikaa ylläpitotyötä ja -kustannuksia

2.5 Yritysohjelmistojen tilanne

Nykyään yritysten käytössä on perinteisiä asiakas-palvelin-mallin mukaisia Windows-

työasemalla ajettavia ohjelmia, jotka on sovelluspalvelinratkaisuilla siirretty SaaS-ympäristöön ja web-arkkitehtuuriin perustuvia ohjelmia, jotka sellaisenaan soveltuvat ainakin pääosin SaaS-ympäristössä käytettäväksi. Näiden erona on toiminnallisuuden taso ja ohjelmistojen laajuus.

Windows-työasemaohjelmat ovat laajoja toiminnallisuuksia kattavia kriittisiä prosesseja valvovia kokonaisuuksia, joiden käyttökokemus perustuu Windows-käyttöliittymään. Näihin kokonaisuuksiin on usein tehty asiakaskohtaisia sovituksia ohjelmiston toteutukseen. Ohjelmistojen käyttöiät ovat pitenemässä ja niihin tehtyjen investointien takaisinmaksuaika on vaikeasti laskettavissa. Ohjelmistojen vaihto vaatii usein suurta projektia, kun vanhan ohjelmiston tiedot pitää konvertoida uuteen ja varmistua, että uudella ohjelmistolla saadaan päivittäiset kriittiset toiminnot hoidettua. Asiakkailla on pyrkimys saada näiden ohjelmistojen ylläpito- ja käyttökustannuksia pienennettyä samalla kun käytettävyyttä ja ominaisuuksia joudutaan muuttamaan. Ohjelmistotoimittajan tavoitteena on usein selvitä tarvittavista kehityshankkeista mahdollisimman kevyesti.

Vanhemmat kaupalliset Web-sovellukset ovat rajatumman toiminnallisen kokonaisuuden toteuttavia ohjelmistoja, joissa ei välttämättä ole niin paljon asiakaskohtaista muuntelua ohjelmiston toteutuksen tasolla. Muuntelu on toteutettu käyttöliittymän muokkauksella ja vakioiduilla muuntelupisteillä ohjelmistossa. Ohjelmistot on usein integroitu yrityksen kriittisiin järjestelmiin. Uudemmat Web-ohjelmistot ovat laajempia ja ne on usein jo tarkoitettu käytettäväksi SaaS-ympäristössä.

Monet yritysten sisäisessä käytössä olevat ohjelmistot on avattu osittain ulkopuoliseen käyttöön Web Services -rajapintojen avulla. Näin käytössä oleva sovellus on saatu myös rajoitettuun yhteistyökumppanien käyttöön ja yritysten välinen integraatio on mahdollistunut.

Eri tavoin kehittyneet ohjelmistot ovat kehittyessään muodostuneet tietämyksen tallennuspaikoiksi, kun asiakkaiden prosessit ovat muovanneet ohjelmistojen toiminnallisuutta. Halu tämän tiedon hyödyntämiseen ohjelmistojen myynnissä ja jatkokehityksessä on muodostanut keskeisen osan tuotepohjaisten ohjelmistojen kehittymisessä. Kun ohjelmistojen jakelu on muuttunut keskitettyyn palvelumalliin, on kustannusten välttäminen sekä ohjelmiston kehittämisessä että ylläpidossa muuttunut kriittiseksi tekijäksi uusien asiakkuuksien hankkimisessa.

3 Ohjelmistojen tarjoaminen palveluna

Software as a Service (SaaS) tarkoittaa ohjelmiston tarjoamista asiakkaan käyttöön palveluna. Asiakkaan tekniseen ympäristöön ei tarvitse tehdä mitään asennuksia, asiakkaalle annetaan vain osoite, minkä kautta palveluun pääsee kirjautumaan. Palvelu voidaan ottaa käyttöön jopa ilman erillistä sopimusta, perustamalla käyttäjätili ja alkamalla käyttää ohjelmistoa (esimerkiksi Zoho, Oiko.fi). Tavoitteena on, että asiakas voi valita haluamansa sovelluksen käyttöön helposti, muokata sen toiminnallisuutta haluamakseen ja käyttää sitä tarpeen mukaan. Sovelluksen käytön hinnoittelu voi perustua esimerkiksi kuukausiveloitusmalliin käyttäjämääräisesti, käyttökertoihin perustuvaan tai johonkin muuhun hinnoittelumalliin. Palvelun tulisi olla käytettävissä ja skaalautua käyttötarpeen mukaisesti.

SaaS-mallin perustana on pilvipalvelu-ajattelu. Asiakas ei tarvitse omaa palvelinkapasiteettia, vaan kaikki tiedon prosessointiin ja käyttöön liittyvä on verkossa. NIST on tehnyt määritelmän pilvipalveluista (Cloud Computing). Pilvi sanana kuvaa rajaamatonta joukkoa resursseja, jota on saatavissa internetin välityksellä. Pilvi tarjoaa tallennus- ja prosessointikapasiteettia jatkuvaan ja tilapäiseen käyttöön. Pilven palvelujen hyödyntämisen yhteydessä on kiinnitettävä huomio käytön turvallisuuteen ja tietojen suojaamiseen sekä palvelutasovaatimuksiin ja -sopimuksiin.

3.1 Pilvipalvelun oleelliset piirteet

NIST listaa pilvipalvelun piirteet seuraavasti [NIST11]:

On-demand self-service. A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.

Broad network access. Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).

Resource pooling. The provider’s computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over

the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.

Rapid elasticity. Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.

Measured service. Cloud systems automatically control and optimize resource use by leveraging a metering capability1 at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

3.2 Pilviluokittelu

Pilvipalvelut ovat olleet useamman vuoden ajankohtaisia, kun suuret palveluntarjoajat ja ICT-yritykset ovat rakentaneet omia palvelinkeskuksiaan omiin ja asiakkaidensa tarpeisiin. Peruskäsitteenä pilvi tarkoittaa tiedonkäsittelykapasiteettia ja tiedonkäsittelyvälineistöä, joka on jossakin muualla kuin oman yrityksen tiloissa. NIST määritelmä pilvipalvelulle (Cloud Computing) [NIST11] Jakelumallit (Deployment models) are:

3.2.1 Private

Yksityinen pilvi, yksityisessä käytössä oleva palvelinkapasiteetti, joka on suojassa ulkomaailmalta. Voi olla oman tietoverkon ulkopuolella, mutta rajattu vain yrityksen käyttöön.

3.2.2 Community

Yhteisö, joka muodostuu useasta yrityksestä voi jakaa pilven, tarkoituksena mahdollistaa yhteisten intressien hoitaminen samassa pilvessä.

3.2.3 Public

Kaikille käytössä oleva palvelinkapasiteetti, joka on palveluntarjoajan tiloissa (Google, Amazon, MS Azure).

3.2.4 Hybrid

Esimerkiksi kokonaisuus, jossa erityyppiset pilvet on kytketty yhteen ja voidaan hyödyntää, kun tarvitaan esimerkiksi kuorman tasausta sovellusten välillä. Erityyppisiä palveluita, mitä pilveen on toteutettu esimerkkeinä. Pilvien välinen yhteistyö mahdollistaa rajoittamattoman skaalautumisen ja kuorman tasauksen.

3.3 Pilvipalvelumallit

NIST-määritelmän [NIST11] mukaan nämäkin: Esimerkkejä PaaS-palvelusta ja IaaS palvelusta tulisi löytää, sen jälkeen keskittyä SaaSiin. Näistä voinee mainita Amazon, Google, … Microsoft ? Näillä vähän eri lähestymistapa, käytännössä Microsoft tarjoaa alustaa, jonka kautta ohjelmistotoimittajat voivat tarjota omaa ohjelmistoaan asiakkaille SaaS palveluna. Google tarjoaa omille tuotteilleen alustan ja Amazon tarjoaa palvelinkapasiteettia.

Platform as a Service

Infrastructure as a Service

Software as a Service

3.4 Software as a Service

Software as a Service määritelmä: Suomalaisten paperissa useampia määritelmiä, kriittinen Web-browser, ei Multi-Tenant ajatusta, joka muualla on aivan oleellinen [MJ10]. Muissa lähteissä korostetaan Multi-Tenant-käsittelyä, varioitavuutta [tähän niitä lähteitä]. Standardoitu ohjelmisto, jota tarjotaan sellaisenaan, erona ASP-ratkaisuun, jossa jokainen asiakastoteutus voi olla asiakaskohtaisesti sovitettu. Tähän kohtaan pitäisi saada kasattua SaaS-vaatimukset listausta lähteistä ja kantaa ottaen, mitä pitäisi olla.

Tässä muistilistaa aiheesta: SaaS-palvelun toteuttamiseksi turvallisuus, käytettävyys, skaalautuvuus, varioitavuus (prosessit, käyttöliittymät, integrointimahdollisuudet ja -rajapinnat) on oltava. Usean version ajo rinnakkain ja päivitykset omavalintaisesti oli esitetty lähteessä [], jotta asiakkaalla on riittävä itsemääräämisoikeus ohjelmistostaan. Kaksi versiota on jo paljon, ja pakotusta uudempaan versioon tarvitaan, jotta käytössä olevien versioiden ylläpito ei hajoa käsiin.

4 SaaS toteutuksen periaatteet

SaaS toteutuksen kannalta tärkeää on resurssien tehokas käyttö ja keskeisessä roolissa on yritysohjelmiston kannalta yrityskohtaisten tietojen tallentaminen ja käyttö. Tässä osassa käydään läpi SaaS toteutuksen oleelliset piirteet.

4.1 Multi-tenancy

Tenant on asiakas ja multi-tenant tarkoittaa monta asiakasta samaa ohjelmistoa käyttämässä. Asiakas on tyypillisesti organisaatio, jossa voi olla useampia käyttäjiä, eli käyttäjähierarkia on kaksitasoinen. Käyttäjän tunnistuksen yhteydessä on tarpeen tunnistaa myös, minkä asiakkaan käyttäjä on kyseessä. Tämä voidaan tehdä usealla tavalla. Tapa riippuu tyypillisesti ohjelmiston arkkitehtuuriratkaisusta, jolla multi-tenancy on toteutettu. Multi-tenancyn toteutus voidaan tehdä usealla tavalla. Toteutustapa vaikuttaa suoraan SaaS-vaatimusten toteutumiseen.

- Palvelin / asiakas. Ei suoraan liittymistä palvelun käyttäjäksi ilman erillisiä asennuksia. Ei resurssien tehokasta käyttöä. Kaiken monistaminen.
- Virtuaalikone / asiakas. Ei suoraan liittymistä palvelun käyttäjäksi ilman erillisiä asennuksia ja konfigurointeja. Ei resurssien tehokasta käyttöä, käyttöjärjestelmien monistaminen.
- Sovellusinstanssi / asiakas. Asiakkaat ovat samalla (virtuaali-)koneella, jokaisella oma instanssi ohjelmistosta käytössä. Ei suoraan liittymistä, ohjelmiston monistaminen. Voidaan varioida asiakaskohtaisesti ohjelmistoa koodaamalla.
- Yhteinen sovellus asiakkaille: tiedontallennusvaihtoehtoja
    - omat kannat
    - yhteinen kanta, omat schemat
    - yhteinen kanta, yhteinen schema
        - fyysisesti tallennus samalle alueelle (tietueet ja indeksit)
        - fyysisesti tallennus eri alueille (tietueet ja indeksit)

Lähteet luokittelevat multi-tenancyn eri tasot eri tavalla ja asettavat ristiriitaisia vaatimuksia toteutuksille. [Tähän lähteitä] Selvää on, että yhteinen kanta ja schema asettaa

erityisvaatimuksia ohjelmiston muunneltavuudelle. Tällöin ohjelmiston ohjausparametroinnin avulla tulee pystyä toteuttamaan eri asiakkaiden tarvitsemia prosesseja ja tarvittaessa lisätä tai rajoittaa ohjelmiston käsittelemiä tietoja. Tähän on erilaisia varautumiskeinoja esitelty lähteissä, lähtien kaikkiin tauluihin lisättävistä sarakkeista, jotka voidaan ottaa tarvittavaan käyttöön, tiedon tiivistämisestä ja tallennusmenetelmistä. Eri menetelmien vaikutukset ohjelmiston toteutuskerroksiin vaikuttavat vanhojen ohjelmistojen muuntamiseen SaaS-malliin. Vaikutukset vaihtelevat ohjelmiston toteutukseen eri kerroksissa [lähteissä otetaan kantaa, miten vanhoja muunnetaan ja mikä on paras tietokantamalli]. Lähteissä todetaan myös, että tietokantatoimittajilla ei ole tuotteissaan tarvittavaa välineistöä multi-tenancyn toteuttamiseen. Tässä työssä esitetään erään kaupallisen tietokantaohjelmiston (Progress Software) multi-tenancy ratkaisu, jossa tietokannan käsittelyyn on toteutettu multi-tenancy-ominaisuuksia. Huomioitava on, että multi-tenancy ei ole pelkästään SaaS-toteutuksia varten kehitetty ominaisuus tietokantaan, ominaisuuden avulla voidaan myös perinteisen toimitusmallin ohjelmistoihin toteuttaa esimerkiksi konserniyrityskäsittelyn piirteitä.

4.2 Muunneltavuus

Käyttöliittymän muunneltavuus, konfiguroitavia piirteitä käyttöliittymässä.

Toiminnallisuuden muunneltavuus, toteutettava sovellukseen muuntelupisteinä (ohjelmistokehys), ja konfiguroinnilla.

Tietokannan muunneltavuus (tai käytettävien tietojen muunneltavuus), mikä on mahdollinen ratkaisu ja eri ratkaisumallien ongelmat.

4.3 Tietoturvallisuus

SaaS-palvelun käyttö vaatii luottamusta siihen, että yhden asiakkaan tiedot pysyvät suojassa muilta käyttäjiltä kaikissa tilanteissa. Käyttäjä ja asiakas hierarkia, asiakastiedon käsittely istuntokohtaisesti niin, että koko ajan on tiedossa oikea asiakkuus ja niin, että asiakkuustietoa ei voi manipuloida.

4.4 Suorituskyky

Käyttäjän ei tulisi häiriintyä muiden käyttäjien toimista, vaan ohjelmiston tulisi pitää vasteajat kunnossa riippumatta siitä, kuinka monta käyttäjää ohjelmistolla on ja mitä

ohjelmiston käyttäjät tekevät. Tässä yhteydessä voisi esitellä viitteitä, joissa esitellään ratkaisuja tähän [liite löytyy listasta].

4.5 Skaalautuvuus

Skaalautuvuus tarkoittaa, että käyttäjien lisääntyessä voidaan joustavasti ottaa käyttöön tarvittavia.

4.6 Administrointi

Jokaisella asiakkaalla tulisi olla mahdollista hallinnoida omia tietojaan ja konfiguraatiotaan ohjelmistosta. Joissain lähteissä mainitaan myös asiakkaan vaikutusmahdollisuus siihen, mitä ohjelmistoversiota käytetään. Tämä vaatii mahdollisesti useamman ohjelmistoversion rinnakkaista suoritusmahdollisuutta.

Lisäksi ohjelmiston ylläpitoon tarvitaan super-tenant, eli käyttäjä, joka voi käsitellä tarvittaessa kaikkien asiakkaiden tietoja ja tehdä tarvittavia huolto- ja ylläpitotoimia ohjelmistolle. Näitä voivat olla esimerkiksi asiakkuuden siirto toiselle palvelimelle kuormituksen tasaamiseksi tai tietokannan konfigurointimuutokset, samoin kuin palvelimen hallinnolliset toimet.

4.7 Vanhan ohjelmiston migraatio uuteen malliin

Mahdolliset ratkaisutavat, millaisia vaihtoehtoja kirjallisuudessa tarjotaan.

5 Progress

Proress Software tuoteperheen esittely. Ohjelmistokehitysvälineet, tietokantaratkaisu ja tietokantaan sisäänrakennettujen multi-tenant ominaisuuksien kuvaus. Vertailua muihin tietokantatoimittajiin.

6 Vahinkovakuutusohjelmiston esittely ja multi-tenancy-toteutuksen rakenne

Kuvaus vahinkovakuutusohjelmistosta asiakkaan sallimalla tasolla. Tarvittavat muutokset, jotta käytettävä tietokanta voidaan muuntaa multi-tenant-käsittelyyn. SaaS-toteutuksen asettamat vaatimukset ja rajoitukset. Projektin sisällön arviointi ja projektin hallintaan liittyviä asioita.

7 Yhteenveto

Kannattaako vai eikö kannata. Onko vanhoilla asiakas-palvelin-arkkitehtuurin tietokantaohjelmistoilla tulevaisuutta SaaS-palveluna. Löytyykö SaaS hyödyistä riittävän suurta tekijää, joka puoltaa multi-tenancy-käsittelyä. Onko multi-tenancy kantaratkaisuna muuten hyödyllinen? Näihin ja moniin muihin kysymyksiin pyritään vastaamaan.

Lähteet

JA07 Jacobs, D., Aulbach, S., Ruminations on Multi-Tenant Databases. Datenbanksysteme in Büro, Technik und Wissenschaft (German Database Conference) - BTW 2007. Pages 514-521.

BA10 Bezemer, C. Zaidman, A., Multi-tenant SaaS applications: maintenance dream or nightmare?. IWPSE-EVOL '10 Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE) 2010. Pages 88-92.

Per10 Perera, S. et al, Multi-tenant SOA Middleware for Cloud Computing. Cloud Computing (CLOUD), 2010. Pages 458-465.

GSM08 Grund, M.,Schapranow, M., Krueger, J., Schaffner, J., Bog, A., Shared Table Access Pattern Analysis for Multi-Tenant Applications. Advanced Management of Information for Globalized Enterprises, 2008. Pages 1-5.

AS11 Aulbach, S., Seibold, M., Jacobs, D., Kemper, A., Extensibility and Data Sharing in evolving multi-tenant databases. Data Engineering (ICDE), 2011. Pages 99-110.

Bez10 Bezemer, C. et al, Enabling multi-tenancy: An industrial experience report. Software Maintenance (ICSM), 2010. Pages 1-8.

LSZ09 Lin, H., Sun, K., Zhao, S., Han, Y., Feedback-Control-Based Performance Regulation for Multi-Tenant Applications. Parallel and Distributed Systems (ICPADS), 2009. Pages 134-141.

WZ10 Wang, H., Zheng, Z., Software Architecture Driven Configurability of Multi-tenant SaaS Application. Proceedings of the 2010 international conference on Web information systems and mining. Pages 418-424.

KNL08 Kwok, T., Nguyen, T., Lam, L., A Software as a Service with Multi-tenancy Support for an Electronic Contract Management Application. Services Computing, 2008. SCC '08. IEEE International Conference. Pages 179-186.

AJ09 Aulbach, S., Jacobs, D., Kemper, A., Seibold, M., A Comparison of flexible schemas for software as a service. Proceedings of the 2009 ACM SIGMOD International Conference on Management of data. Pages 881-888.

JLZ09 Jiang, D., Li, G., Zhou, Y., Supporting Database Applications as a Service. Data Engineering, 2009. IDCE '09. IEEE 25th International Conference, pages 832-843.

SR11 Sengupta, B., Roychoudhury, A. Engineerint Multi-Tenant Software-as-a-Service Systems. Proceedings of the 3rd International Workshop on Principles of Engineering Service-Oriented Systems, 2011. Pages 15-21.

Pal10 PalsonKennedy, R., Assessing the risks and opportunities of Cloud Computing — Defining identity management systems and maturity models. Trendz in Information Sciences & Computing (TISC), 2010. Pages 138-142.

ZSTC10 Zhang, X., Shen, B., Tang, X., Chen, W., From isolated tenancy hosted application to multi-tenancy: Toward a systematic migration method for web application. Software Engineering and Service Sciences (ICSESS), 2010. Pages 209-212.

Kaz09 Kazan, H. Jr., Cloud Software Service: Concepts, Technology, Economics. Service Science Winter 2009 vol. 1 no. 4. Pages 256-269.

Kaz08 Kazan, H. Jr., Cloud Computing, I-Service, And IT Service Provisioning. Journal Of Service Science, Vol 1, No 2, 2008. Pages 57-64.

MC11 Momm, C. ja Krebs, R., A Qualitative Discussion of Different Approaches for Implementing Multi-Tenant SaaS Offerings. In Proceedings of Software Engineering 2011. Pages 131-150.

LSL09 Li, H., Shi, Y., Li, Q., A Multi-granularity Customization Relationship Model for SaaS. Web Information Systems and Mining, 2009. Pages 611-615.

SD09 Shwartz, L., Diao, Y., Grabarnik, G., Multi-tenant solution for IT service management: A quantitative study of benefits. Integrated Network Management, 2009. Pages 721-731.

LH10 Lizhen, C., Haiyang, W., Lin, J. Pu, H., Customization modeling based on metagraph for multi-tenant applications. Pervasive Computing and Applications (ICPCA), 2010. Pages 255-260.

Ka10 Kang, S. et al. A General Maturity Model and Reference Architecture for SaaS Service. Proceedings of the 15th international conference on Database

Systems for Advanced Applications - Volume Part II, 2010. Pages 337-346.

MK09 Müller, J., Krüger, J., Enderlein S., Helmich M., Zeier A., Customizing Enterprise Software as a Service Applications: Back-End Extension in a Multi-tenancy Environment. Enterprise Information Systems Lecture Notes in Business Information Processing Volume 24, 2009. Pages 66-77.

NIST11 The NIST Definition of Cloud Computing. Special Publication 800-145. Recommendations of the National Institute of Standards and Technology, 2011.

Jatko seuraa

ja niin edelleen