Ratas-asiakkuudenhallintajärjestelmän suunnittelu

Juuso Raitala

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos/tko
Pro gradu -tutkielma
Ohjaaja: Pirkko Nykänen
28.10.2009

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos/tko
Juuso Raitala: Ratas-asiakkuudenhallintajärjestelmän suunnittelu
Pro gradu -tutkielma, 48 sivua
Lokakuu 2009

Asiakkuudenhallinta yhdistää asiakkaiden ja myyntiorganisaation eri kosketuspinnat, kuten sähköpostin, puhelinmyynnin ja myymälät, yhtenäiseksi näkymäksi asiakkaasta ja tarjoaa myyntiorganisaatiolle entistä tarkemman kuvan asiakkaista ja heidän tarpeistaan myyntityön edistämiseksi. Asiakkuudenhallintajärjestelmien käyttö parantaa asiakastyytyväisyyttä ja -uskollisuutta ja auttaa kohdentamaan myyntityötä asiakkaiden tarpeiden mukaan ja edistämään myyntiorganisaation ja asiakkaiden välistä kommunikaatiota.

Tämän tutkimuksen tavoitteena oli luoda suunnitelma yrityksen Y liiketoimintatarpeita vastaavan asiakkuudenhallintajärjestelmän, työnimeltään Ratas, toteuttamiseksi. Järjestelmä on tarkoitettu tarjottavaksi palveluna yrityksen Y asiakkaille. Tavoitteena oli määritellä sovellusalueen käsitteet eli yhteinen kieli järjestelmän sovellusalueen käsitteiden kuvaamiseksi. Näiden käsitteiden pohjalta pyrkimyksenä oli luoda käsitteiden välisiä suhteita kuvaavat mallit, joita ovat asiakasorganisaation myyntiprosessia kuvaava prosessimalli ja järjestelmän tietosisältöä ja tietueiden välisiä suhteita kuvaava tietomalli. Edellisten vaihden tuotosten perusteella pyrittiin luomaan menetelmät, joita käyttäen järjestelmän suunnittelua voidaan viedä eteenpäin. Viimeisessä vaiheessa, toteutukset, pyrkimyksenä oli laatia konkreettinen suunnitelma järjestelmän toteuttamiseksi, sisältäen tietojärjestelmä- ja teknologia-arkkitehtuuriset valinnat, kuten hyödynnettävät arkkitehtuuriset ratkaisut ja käytettävät sovellusalustat ja -teknologiat, sekä ohjelmiston rakenteen. Tämän vaiheittaisen etenemismallin, jossa kunkin vaiheen tuotokset perustuivat edellisille vaiheille, käytön tavoitteena oli tarjota tutkimukselle selkeä runko ja rakenne.

Tämän tutkimuksen puitteissa järjestelmän toteutuksesta luotiin onnistuneesti suunnitelma, jonka pohjalta varsinainen ohjelmointityö voidaan aloittaa. Välituotoksina syntyivät suunnitellut käsitteet, mallit, menetelmät ja toteutukset. Työtä evaluoitiin kunkin vaiheen aikana ja jälkeen iteratiivisesti yrityksen Y edustajien kanssa, joten tutkimuksen ja sen pohjalta myöhemmin laadittavan konkreettisen toteutuksen voidaan olettaa vastaavan yrityksen Y liiketoimintatarpeita.

Avainsanat: asiakkuudenhallinta, tuotannonohjaus, yritysarkkitehtuuri

KIITOKSET

Haluan kiittää yritystä Y mahdollisuudesta tehdä pro gradu -tutkielma mielenkiintoisesta aiheesta, työkavereitani ja pomoani rakentavasta yhteistyöstä työn viemiseksi eteenpäin, avopuolisoani Hanna-Kaisa Heikkilää tuesta ja ymmärryksestä työn vaikeina hetkinä ja pro gradu -työni ohjaajaa Pirkko Nykästä rohkaisevasta palautteesta ja arvokkaista kommenteista pitkin työprosessia. Teidän tukenne avulla käsissäni on nyt valmis pro gradu -tutkielma ja olen valmis siirtymään seuraaviin haasteisiin.

Tampereella, lokakuussa 2009
Juuso Raitala

SISÄLLYS

1 JOHDANTO

Asiakkuudenhallinta eli CRM (engl. customer relationship management) on ihmisten, prosessien ja teknologian välinen integroitu yhdistelmä yrityksen asiakkaiden ymmärtämiseksi, säilyttämiseksi ja asiakassuhteen kehittämiseksi [Chen & Popovich, 2003]. CRM-sovellukset yhdistävät asiakasrajapinnan toiminnot, kuten myynnin, markkinoinnin ja asiakaspalvelun, ja taustaprosessit, kuten taloushallinnon, logistiikan ja henkilöstön hallinnan, asiakkaan kosketuspintoihin (engl. "touch point") [Fickel, 1999; Chen & Popovich, 2003]. Kosketuspintoja voivat olla esimerkiksi Internet, sähköposti, myynnit, puhelinmyynti, mainonta ja myymälät [Chen & Popovich, 2003]. CRM integroi eri kosketuspinnat yhtenäiseksi näkymäksi asiakkaasta [Eckerson & Watson, 2000]. CRM on siis monimutkainen eri kosketuspinnoista asiakasdataa keräävä järjestelmä, jonka avulla kyetään määrittelemään avainasiakkaat, ennustamaan heidän tulevaa ostokäyttäytymistään, kohdentamaan markkinointia ja palkitsemaan kanta-asiakkaita [Chen & Popovich, 2003]. CRM mahdollistaa entistä paremman vastaavuden asiakkaan tarpeiden ja organisaation tarjoamien tuotteiden ja palveluiden välillä [Christopher *et al.*, 2002].

CRM on kehittynyt Sales Force Automation (SFA) -järjestelmistä [Peppers & Rogers, 1999]. SFA voidaan suomentaa esimerkiksi myynnin tai myyntityön automatisoinniksi. SFA-ohjelmistot automatisoivat rutiininomaisia tehtäviä kuten asiakaskontaktien seurantaa ja ennustamista, jolloin myyntihenkilöstö voi paremmin keskittyä myymiseen, ei niinkään hallinnollisiin tehtäviin [Chen & Popovich, 2003]. Baysan ja muut [2005] määrittelevät SFA-työkalun tarkemmin tietokoneistetuksi järjestelmäksi, jonka avulla myyntihenkilöstö ja -johto voivat pitää kirjaa myyntijohtolangoista eli liideistä (engl. sales lead), hallinnoida kontakteja, hallita asiakassuhteita, seurata myyntiprosesseja, aikatauluttaa tapaamisia, ennustaa myyntejä ja arvioida työntekijöiden suoriutuvuutta. Heidän mukaansa SFA-työkalut pyrkivät myös kasvattamaan myyntitiimin tehokkuutta. CRM:llä on kuitenkin juuret myös suhdemarkkinoinnissa (engl. relationship marketing), jonka tavoitteena on maksimoida asiakkaan yritykselle tuoma arvo koko asiakkuuselinkaaren aikana. Suhdemarkkinointi mahdollistaa markkinoinnin fokusoinnin useille eri markkinoille. [Christopher *et al.*, 2002].

Monet kaupallisesti saatavilla olevat SFA-työkalut eivät vastaa kaikkien yritysorganisaatioiden erityistarpeisiin liiallisen yleisluontoisuutensa vuoksi [Baysan *et al.*, 2005]. Tästä syystä CRM- ja SFA-markkinoilla on tilaa yritys Y:n tarpeisiin räätälöidylle CRM-ohjelmistoratkaisulle. Tämän tutkimuksen tavoitteena on suunnitella ja osin toteuttaa asiakkuudenhallintajärjestelmä. Järjestelmän työnimi on Ratas. Tässä tutkimuksessa laaditun suunnitelman pohjalta yritys Y luo järjestelmästä tuotteen, jota

se voi myydä eri asiakasorganisaatioiden käyttöön. Tuote myydään kullekin asiakasorganisaatiolle asiakaskohtaisesti konfiguroituna ja tarvittaessa räätälöitynä yritys Y:n ylläpitämänä palveluna (engl. Software as a Service eli SaaS).

Tämän tutkimuksen luvussa 2 esitellään tutkimuksen tieteellinen tausta eli käytettävät tutkimusmenetelmät ja asiakkuudenhallintajärjestelmän suunnittelun tutkimukselliset lähtökohdat. Luvussa 3 määritellään suunniteltavan järjestelmän toteutuksen perusperiaatteet ja ne tarpeet, joihin järjestelmä toteutettaessaan vastaa, ja lisäksi rajataan tutkimus. Neljännessä luvussa käydäään läpi järjestelmän suunnittelun prosessi Marchin ja Smithin [1995] tietojärjestelmätieteen tutkimuksen eri vaiheisiin – käsitteet, mallit, menetelmät ja toteutukset – perustuvan jaottelun mukaan. Viidennessä luvussa kootaan yhteenveto koko tutkimuksesta ja sen toteutuksesta sekä evaluoidaan tutkimus ja sen kulku.

2 SUUNNITTELUN TUTKIMUKSELLISET LÄHTÖKOHDAT

Tässä luvussa esitellään tutkimuksessa käytettävät tutkimusmenetelmät ja asiakkuudenhallintajärjestelmän suunnittelun tutkimukselliset lähtökohdat.

2.1 Käytettävät tutkimusmenetelmät

Tietojärjestelmätieteen tutkimus voidaan jakaa kahteen pääosaan, luonnontieteelliseen ja suunnittelutieteelliseen. Luonnontieteellinen tutkimus on luonteeltaan deskriptiivistä ja siinä yritetään ymmärtää informaatioteknologian luonnetta. Preskriptiivisen tutkimuksen tavoitteena on parantaa informaatioteknologian suorituskykyä. [March & Smith, 1995]. Suunnittelutieteellisen tutkimuksen tavoitteena on luoda artefakteja päämäärien saavuttamiseksi [Simon, 1996].

Hevnerin ja muiden [2004] mukaan suunnittelutieteellisen tutkimuksen tavoitteena on hyödyllisyys. Suunnittelutieteellinen paradigma pohjautuu insinööritaitoon ja innovaatioiden suunnitteluun [Simon, 1996]. Se on pohjimmiltaan ongelmanratkaisuparadigma [Hevner *et al.*, 2004], ja se pyrkii luomaan ideoita, käytäntöjä, teknisiä mahdollisuuksia ja lopputuotteita määritteleviä innovaatioita, joita voidaan hyödyntää analyysissa, suunnittelussa, toteutuksessa, ylläpidossa ja tietojärjestelmien käytössä [Denning, 1997; Tsichritzis, 1998]. Suunnittelutiede luo ja arvioi organisatoristen ongelmien ratkaisuun tarkoitettuja ja tunnistettuja businesstarpeita vastaavia tietoteknisiä artefakteja [Hevner *et al.*, 2004].

Marchin ja Smithin [1995] luoma tietojärjestelmätieteen tutkimuskehys jakautuu suunnittelutieteellisen ja luonnontieteellisen lähestymistavan välillä (Taulukko 1). Taulukon pystyakselilla on suunnittelutieteellisen tutkimuksen tuotokset: käsitteet, mallit, menetelmät ja toteutukset. Vaaka-akselilla on suunnittelu- ja luonnontieteelliset tutkimusaktiviteetit: artefaktien luominen tai rakentaminen, niiden arvioiminen, teorioiden luominen ja niiden oikeaksi osoittaminen (perustelu).

Suunnittelutieteellisiä tutkimusaktiviteetteja ovat *rakentaminen* (engl. build) ja *evaluointi* eli *arviointi* (engl. evaluate). Näistä edellinen tarkoittaa artefaktin luomista. Jälkimmäisessä aktiviteetissa arvioidaan, kuinka hyvin artefakti täyttää tarkoituksensa. Tätä voidaan mitata erilaisilla järjestelmän ominaisuuksia mittaavilla mittareilla. Metriikoiden puute ja epäonnistuminen artefaktin suorituskyvyn mittaamisessa vakiintuneita kriteerejä vasten johtavat heikkoihin tuloksiin. Luonnontieteellisiä tutkimusaktiviteetteja ovat *teoretisointi* eli *teorian luominen* (engl. theorize) ja teorian *perustelu* (engl. justify). Näistä edellisessä tutkitaan, miten ja miksi järjestelmä toimii toimin-

	Rakentaminen	**Arviointi**	**Teorisointi**	**Perustelu**
Käsitteet				
Mallit				
Menetelmät				
Toteutukset				

Taulukko 1: Informaatioteknologian tutkimuksellinen viitekehys [March & Smith, 1995].

taympäristössään. Jälkimmäinen aktiviteetti on esitetyn teorian testaamista ja perustelemista. [March & Smith, 1995]. Tämä tutkimus, asiakkuudenhallintajärjestelmän suunnittelu, on luonteeltaan suunnittelutieteellistä, eli suoritettavat aktiviteetit ovat rakentaminen ja evaluointi.

March ja Smith [1995] jakavat informaatioteknologisen järjestelmän kehittämisen suunnittelutieteellisen prosessin neljään vaiheeseen niiden tuotoksien perusteella:

1. Ensimmäisessä vaiheessa määritellään järjestelmän sovellusalueen kannalta olennaiset *käsitteet* (engl. constructs). Ne muodostavat sovellusalueen kielen ja jaetun tietämyksen. Käsitteet voivat joko erittäin formaaleja (esimerkiksi datamallinnuksen käsitteet kuten entiteetti ja attribuutti) tai epäformaaleja (esimerkiksi yhteistoiminnallisen työn käsitteet konsensus, osallistuminen ja tyytyväisyys). Käsitteellistämisellä luodaan kieli, jota voidaan käyttää tehtävien kuvaamiseen ja ajattelemiseen.

2. Toisessa vaiheessa määritellään *mallit* (engl. models) eli käsitteiden väliset suhteet. Tähän vaiheeseen kuuluu toiminnallisuuden, tehtävien ja artefaktien kuvaaminen. Mallit ovat eri tilanteiden ongelma- ja ratkaisukuvauksia. Mallin arvo syntyy sen hyödyllisyydestä, ei niinkään totuudesta, eikä mallin tarvitse olla täysin tarkka ja virheetön ollakseen hyödyllinen. Mallin kuitenkin on syytä vastata riittävissä määrin todellisuutta.

3. Kolmannessa vaiheessa, *menetelmät* (engl. methods), suunnitellaan miten aikaisempien vaiheiden tuotosten mukainen järjestelmä toteutetaan. Menetelmä on sarja tehtäviä jonkin tehtävän suorittamiseksi. Menetelmät perustuvat niiden taustalla oleviin käsitteisiin ja malleihin [Nolan, 1973]. Menetelmät koostuvat luonnostaan tehtävistä ja lopputuloksista ja voivat perustua tiettyihin malleihin siten, että malleja tai niiden osia käytetään menetelmät-vaiheen tiettyjen askelten

syötteenä. Usein malleja käytetään muuttamaan malli tai kuvaus toiseksi ongelmaa ratkaistaessa.

4. Neljännessä vaiheessa luodaan järjestelmästä *toteutus* tai toteutukset (engl. instantiations) tiettyyn ympäristöön laadittujen käsitteiden, mallien ja menetelmien perusteella. Toteutusten taustalla on aina jotkin käsitteet, mallit ja menetelmät, vaikkei niitä eksplisiittisesti olisi määriteltykään.

Eri vaiheiden nimet ovat siis käsitteet, mallit, menetelmät ja toteutukset. Hyödynnän järjestelmän suunnittelussa vaiheisiin perustuvaa jakoa ja tutkimukseni kohdistuu näiden neljän vaiheen tutkimustuotosten luontiin ja osin arviointiin. Marchin ja Smithin [1995] mukaan kunkin tutkimustuotoksen luonnin jälkeen vaiheen tuotoksiin voidaan soveltaa evaluointiaktiviteettia. Luonti- ja arviointiaktiviteetteja tyypillisesti iteroidaan lukuisia kertoja ennen lopullisen suunnitteluartefaktin luomista [Markus *et al.*, 2002]. Tutkimuksen edetessä luotavat artefaktit annetaan yritys Y:n edustajien arvioitavaksi ja artefakteja kehitetään edelleen heidän antamansa palautteen perusteella. Tämän tarkoituksena on päästä yhteisymmärrykseen kunkin vaiheen tuotoksista ja onnistumisesta mahdollisimman aikaisessa vaiheessa eli varmistua, että käytetyt käsitteet, hyödynnetyt mallit, laadittu suunnitelma järjestelmän toteutuksesta ja toteutettu järjestelmä ovat yritys Y:n vaatimusten mukaisia.

Taulukossa 2 on esitetty tutkimukseni kattavuusalue, joka sisältää rakentamisaktiviteetit ja osan arviointiaktiviteeteista (merkitty X:llä). Koska toteutusta ei luoda valmiiksi tämän tutkimuksen puitteissa, joudun rajaamaan sen arvioinnin pois.

	Rakentaminen	**Arviointi**	**Teorisointi**	**Perustelu**
Käsitteet	X	X		
Mallit	X	X		
Menetelmät	X	X		
Toteutukset	X			

Taulukko 2: Informaatioteknologian tutkimuksellisen viitekehyksen soveltaminen Ratas-järjestelmän suunnittelussa.

2.2 Yritysarkkitehtuurin viitekehys

Tietojärjestelmien roolia kuvaavaa, koko organisaation kontekstin huomioivaa arkkitehtuurimallia kutsutaan yritysarkkitehtuuriksi (engl. enterprise architecture, EA) [Sil-

tanen, 2004]. Se on yleissuunnitelma organisaation toiminnasta ja kuvaa organisaation tieto- ja tietojärjestelmätarpeita [Schekkerman, 2004; Siltanen, 2004]. Weill [2007, s. 2] kuvaa yritysarkkitehtuuria seuraavasti: "yritysarkkitehtuuri on liiketoimintaprosessien ja tietoteknisen infrastruktuurin organisointilogiikka suhteutettuna yrityksen toimintamallin integraatio- ja standardointivaatimuksiin". Koherentin ja kokonaisen yritysarkkitehtuurin luomiseksi on otettava huomioon kaikki sen kannalta olennaiset liiketoiminta-aspektit, kuten ihmiset, operatiivisen toiminnan rakenne, tietotarpeet, tietojärjestelmät sekä niiden taustalla olevat hallinnolliset periaatteet ja liiketoiminnan tavoitteet ja strategiat [Vasconcelos *et al.*, 2004; Siltanen, 2004]. Kyseessä on mille tahansa päämääräohjautuvalle organisaatiolle, ei siis vain kaupallisille, pätevä kokonaisarkkitehtuuriajattelu [Schekkerman, 2004; Siltanen, 2004]. Schekkerman [2004] jakaa yrityksen kokonaisarkkitehtuurin neljään osaan:

1. *Liiketoimintataso*, liiketoiminta-arkkitehtuuri (engl. business architecture tai enterprise architecture) määrittelee organisaation "operaativisen toiminnan eli prosessien kulun" [Siltanen, 2004, s. 45], tarkennettuna liiketoimintarakenteisiin, suhteisiin, tehtäviin ja aktiviteetteihin riittävällä tarkkuudella tarvittavan teknologisen tuen ja suorituskykymetriikkojen validoinnin tukemiseksi [Schekkerman, 2004].

2. *Tietotaso*, tietoarkkitehtuuri (engl. information architecture) kuvaa "toimintaan liittyvät tietotarpeet" [Siltanen, 2004, s. 45], eli tärkeimmät informaation luonteet ja -virrat liiketoiminta-alueella sovittaen ne kokonaisarkkitehtuuriin, ja tarjota tiedon liikkumisen ja tarvittavien tietoturvapalveluiden määrittely [Schekkerman, 2004].

3. *Tietojärjestelmätaso*, tietojärjestelmäarkkitehtuuri (engl. information system architecture) kuvaa järjestelmät, joilla tietoarkkitehtuurin sisältämiä tietoja prosessoidaan ja hallinnoidaan liiketoimintatason edellyttämillä tavoilla [Siltanen, 2004].

4. *Teknogiataso*, teknologia-arkkitehtuuri (eng. technology architecture) kuvaa teknologiset ratkaisut, joilla tietojärjestelmätason mukaiset järjestelmät toteutetaan [Siltanen, 2004].

Schekkermanin jaottelussa erotetaan operatiivinen toiminta ja sovellusten taustalla oleva teknologia toisistaan. Operatiivisen toiminnan prosessit ovat erillään konkreettisista sovelluksista (järjestelmistä) ja teknisistä ratkaisuista. [Siltanen, 2004]. Hyö-

dynnän tutkielmassani kokonaisarkkitehtuuriajattelun mukaista jaottelua. Aloitan organisaation liiketoiminta-arkkitehtuurista, jossa mallinan liiketoimintaprosessit. Tämän jälkeen etenen tietoarkkitehtuuriin, jossa mallinnan ja määrittelen tarvittavat tiedot ja niiden väliset suhteet. Tietojärjestelmäarkkitehtuurin määrittelen määrittelemällä ja kuvaamalla ne järjestelmät, joiden avulla tietoarkkitehtuurin mukaisia tietoja käsitellään liiketoimintatason vaatimusten toteuttamiseksi. Tämän jälkeen määrittelen käytettävät teknologiset ratkaisut. Luon siis kussakin vaiheessa vaiheen mukaisia evaluoitavissa olevia artefakteja. Myöhemmän vaiheen prosessit (ja niiden aikana luodut artefaktit) perustuvat aikaisempien vaiheiden tuotoksille.

2.3 Suunnittelun lähtökohdat

Tässä kohdassa esittelen Hevnerin ja muiden [2004] määrittelemää suunnittelutieteellisen tietojärjestelmätutkimuksen seitsemän lähtökohtaa tai ohjetta ja miten niitä sovellan tutkimuksen puitteissa. Esittelen kunkin lähtökohdan; arvioin niiden soveltuvuutta käsillä olevaan tutkimukseen; ja määrittelen, miten sovellan ohjeistusta tutkimuksessa ja mitä kriteerejä ohjeistuksen perusteella voidaan luoda, kun arvioidaan tutkimuksen kunkin tutkimustuotoksen (käsitteet, mallit, menetelmät ja toteutukset) onnistumista. Hevner ja muut [2004] painottavat, ettei näitä ohjeistuksia kannata totella mekaanisesti, vaan he pitävät niitä apuvälineenä tehokkaaseen suunnittelutieteelliseen tutkimukseen. Siksi hyödynnän ohjeistuksia ainoastaan soveltuvin osin. Hevner ja muut [2004] määrittelevät ohjeistukset tai lähtökohdat seuraavasti:

1. *Ohje 1 (luo artefakti)*: Suunnittelutieteellisen tietojärjestelmätutkimuksen on luotava toteuttamiskelpoisia ja toimintakykyisiä artefakteja käsitteiden, mallien, menetelmien tai toteutuksien muodossa. Artefaktien luominen osoittaa sekä suunnitteluprosessin että suunniteltavan lopputuotteen soveltuvuutta.

    - Tämän tutkimuksen tavoitteena on luoda käsitteitä, malleja, menetelmiä ja toteutuksia. Mikäli nämä lopputuotokset syntyvät, on tutkimus tämän ohjeistuksen osalta onnistunut.

2. *Ohje 2 (ongelman relevanssi)*: Suunnittelutieteellisen tietojärjestelmätutkimuksen tavoitteena on omaksua tietämystä ja ymmärrystä teknologiaperusteisten ratkaisujen toteuttamiseksi tärkeisiin ja ratkaisemattomiin liiketoiminnan ongelmiin.

    - Yritys Y:n liiketoiminta-ajattelun kannalta ongelma on relevantti ja räätälöitävälle asiakkuudenhallintajärjestelmälle on tarve.

3. *Ohje 3 (suunnittelun arviointi)*: Suunnitteluartefaktin hyödyllisyys, laatu ja tehokkuus on perusteellisesti todennettava hyvin toteutetuilla arviointimenetelmillä. Liiketoiminnallinen ympäristö luo vaatimukset, joihin artefaktin arviointi perustuu. Luodun tietoteknisen artefaktin arviointi vaatii tarkoituksenmukaisten metriikoiden määrittelemisen ja mahdollisesti tarkoituksenmukaisen datan keräämisen ja analysoinnin. Tietoteknisiä artefakteja voidaan arvioida niiden toiminnallisuuden, täydellisyyden, konsistenssin, virheettömyyden, suorituskyvyn, luotettavuuden, käytettävyyden, organisaatioon sopivuuden ja muiden relevanttien laatuattribuuttien perusteella. Koska suunnittelu on luontaisesti iteratiivista ja inkrementaalista toimintaa, tarjoaa arviointivaihe elintärkeää palautetta suunnitteluprosessin ja sen lopputuotteen laadun parantamiseksi. Suunnitteluartefakti on täydellinen ja tehokas, kun se täyttää ratkaistavan ongelman vaatimukset ja ottaa huomioon sen rajoitukset.

    - Yritys Y on arvioinut kussakin vaiheessa tuotetut tuotokset ja todennut ne kattaviksi ja hyödyllisiksi.

4. *Ohje 4 (tutkimuksen kontribuutiot)*: Tehokas suunnittelutieteellinen tutkimus tarjoaa selkeitä kontribuutioita suunnitteluartefaktien muodossa, lisäyksinä tietämysperustaan ja/tai arviointimenetelmiin ja -metriikoihin. On pyrittävä löytämään mielekäs vastaus kysymykseen, mitkä ovat tutkimuksen tarjoamat uudet ja mielenkiintoiset kontribuutiot. Useimmiten suunnittelutieteellisen tutkimuksen kontribuutio on artefakti. Artefaktien on edustettava teknologista ja liiketoiminnallista ympäristöä, jota tietojärjestelmä mallintaa. Tutkimuksen on osoitettava selkeitä kontribuutioita businessympäristölleen ja ratkaistava tärkeä aikaisemmin ratkaisematon ongelma.

    - Tämän tutkimuksen tärkeimmät kontribuutiot ovat tutkimuksen kussakin vaiheessa luotavat tuotokset sekä tutkimus itsessään mukaan lukien tutkimusprosessin kuvauksen. Tutkimuksen tulokset ovat tutkimuksen kontribuutioita, joiden luominen jossain määrin todentaa ohjeistuksen noudattamista. Tutkimuksen tuotosten lopullista arvoa on vaikea arvioida tämän tutkimuksen puitteissa, sillä tutkimuksen aikana ei synny valmista ja business- ja teknologisessa ympäristössään arvioitavaksi kelpaavaa tietojärjestelmää organisaation käyttöön, mutta vaiheittaiset tulokset rakentavat hyvän perustan toteutukselle.

5. *Ohje 5 (tutkimuksen perusteellisuus)*: Suunnittelutieteellinen tutkimus vaatii pe-

rusteellisia ja täsmällisiä menetelmiä sekä artefaktien luomiseen että arviointiin. Perusteellisuus ja täsmällisyys perustuu tietämysperustan, teoreettisen pohjan ja tutkimusmenetelmien, tehokkaaseen käyttöön. Onnistuneeseen lopputulokseen päästään valitsemalla asianmukaisia tekniikoita artefaktin luomiseksi ja arvioimiseksi.

- Tutkimuksen puitteissa pyritään tutustumaan kattavasti alan aiempaan tutkimukseen sekä asiakkuudenhallinnan käytäntöihin ja tarpeisiin yrityksessä Y ja perustamaan niihin artefaktien luomis- ja arviointiprosessit.

6. *Ohje 6 (tutkimus etsintäprosessina)*: Suunnittelu on pohjimmiltaan etsintäprosessi, jonka tavoitteena on keksiä tehokas ratkaisu ongelmaan. Tehokkaan artefaktin etsiminen vaatii käytettävissä olevien keinojen (toimenpiteet ja resurssit) hyödyntämistä haluttuun lopputulokseen (ratkaisun tavoitteet ja rajoitukset) pääsemiseksi ottaen huomioon ongelmaympäristön rajoitteet (ympäristön tekijät, joihin ei voida vaikuttaa).

   - Tässä tutkimuksessa tuotetaan prosessi- ja tietomallit, joiden avulla on etsitty ratkaisut räätälöitävän asiakkuudenhallintajärjestelmän toteutukselle.

7. *Ohje 7 (tutkimuksen kommunikointi)*: Suunnittelutieteellinen tutkimus on esiteltävä sekä teknologia- että johtamisorientoituneelle yleisölle. Teknologiaorientoitunut yleisö tarvitsee riittävän yksityiskohtaisen kuvauksen artefaktin toteuttamiseksi ja käytettäväksi asiaankuuluvassa organisatorisessa kontekstissa. Johtamisorientoitunut yleisö tarvitsee riittävät yksityskohtaisen kuvauksen päättääkseen pitäisikö organisatorisia resursseja sitoa artefaktin luomiseen ja käyttämiseen tietyssä organisatorisessa kontekstissa.

   - Tutkimus pyritään kuvaamaan siten, että se tarjoaa riittävät tiedot sekä teknologia- että johtamisorientoitunille osapuolille. Tutkimustulokset esitetään käsitteinä, malleina, menetelminä ja toteutuksina.

Tämä tutkimus, asiakkuudenhallintajärjestelmän suunnittelu yritykselle Y, on pyritty toteuttamaan edellä mainittuja ohjeistuksia ja lähtökohtia noudattaen.

3 SUUNNITELTAVAN JÄRJESTELMÄN PERUSPERIAATTEET JA MÄÄRITTELY

Tässä luvussa esittelen asiakkaan näkemyksen suunniteltavasta järjestelmästä, suunnittelun perusperiaatteet ja järjestelmän rajauksen eli laajuuden määrittelyn. Schekkermanin kokonaisarkkitehtuurijaottelussa tämän luvun sisältö edustaa liiketoimintaarkkitehtuuria.

3.1 Suunnittelun perusperiaatteet

Järjestelmä suunnitellaan ja toteutetaan seuraavia yrityksen Y johdon kanssa sovittuja perusperiaatteita noudattaen:

- *Yksinkertaisuus ja minimalistisuus*: Järjestelmä oletusarvoisesti sisältää ainoastaan asiakasorganisaatiolle tarpeellista toiminnallisuutta. Tämä tarkoittaa toiminnallisuuden rajaamista mahdollisuuksien mukaan kunkin asiakasorganisaation tarpeisiin. Yrityksen Y näkemyksen mukaan saatavilla olevat vapaasti muokattavat ja jatkokehitettävät avoimen lähdekoodin asiakkuudenhallintajärjestelmät ovat lähtökohtaisesti liian monimutkaisia ja sisältävät toiminnallisuutta, jonka poistaminen asiakasorganisaatiolle räätälöitävästä tuotteesta on liian työlästä. Kun yksinkertaisuus otetaan suunnitteluperiaatteeksi alusta lähtien, saadaan myös lopputuloksesta yksinkertainen.

- *Räätälöitävyys*: Järjestelmää tulee voida tarvittaessa helposti muokata eri asiakasorganisaatioiden tarpeisiin. Järjestelmän on siis tarkoitus olla geneerinen ratkaisu, josta voidaan räätälöidä asiakasorganisaatiolle sopiva järjestelmä. Tavoitteena on saada kilpailuetua verrattuna vaikeasti tai heikosti mukautettaviin kilpaileviin tuotteisiin. Järjestelmän on lisäksi kyettävä sopeutumaan yhteiseloon muiden järjestelmien – perinnejärjestelmät (engl. legacy systems) mukaanlukien – kanssa.

  - *Laajennettavuus*: Järjestelmää on voitava tarvittaessa laajentaa tukemaan myyntiprosessien lisäksi muita asiakasorganisaation työprosesseja.

- *Ylläpidettävyys*: Järjestelmä on pidettävä helposti ylläpidettävänä räätälöinnistä ja laajennettavuudesta huolimatta. Tähän sisältyy eri asiakasorganisaatioiden järjestelmäasennusten ylläpidon helppous. Käytännössä järjestelmä on syytä suunnitella siten, että eri asiakasorganisaatioiden Ratas-asennukset ovat versioita sa-

masta lähdekoodipuusta ja koodipuuhun tehtävät korjaukset ja parannukset voidaan välittää helposti kaikkiin asennuksiin.

Järjestelmän tärkein tehtävä on tukea asiakasorganisaation myyntiprosesseja. Myyntiprosessin kuvaus otetaan lähtökohdaksi järjestelmän suunnittelussa. Muita tuettavia prosesseja ovat asiakasorganisaation tuotanto-, lähettämö-, varastonhallinta- ynnä muut vastaavat prosessit, jotka tämän tutkimuksen puitteissa hahmotellaan. Näitä prosesseja ei kuitenkaan välttämättä tarkasti mallinneta ja suunnitella järjestelmän tässä kehitysvaiheessa, mutta ne otetaan huomioon järjestelmän laajennettavuudessa.

Järjestelmän tukema myyntiprosessi rajataan siten, että prosessi alkaa yhteydenotosta asiakkaalle ja päättyy laskun lähettämiseen asiakkaalle ja kirjanpitoon. Toistaiseksi järjestelmän ulkopuolelle rajataan esimerkiksi rajapinnat kirjanpitojärjestelmiin. Tuotanto-, varasto- ja lähettämöprosesseja ei tässä yhteydessä juurikaan mallineta, koska eri asiakasorganisaatioiden käyttöön suunniteltavien toteutusten välillä voi olla merkittäviä eroja.

3.2 Toiminnallisuuden kuvaus

Tässä kohdassa kuvataan järjestelmän toiminnallisuus pääpiirteissään sekä esitellään tuettavat prosessit.

3.2.1 Myyntiprosessin tukeminen

Koska myyntiprosessit voivat olla erilaisilla pk-yrityksillä hyvinkin samanlaisia, voidaan tätä hyödyntää myyntiprosessin mallintamisessa ja sitä kautta järjestelmän suunnittelemisessa ja toteuttamisessa. Järjestelmän tukevan myyntiprosessin rakenne voidaan siis mallintaa hyvinkin samanlaiseksi eri asiakasorganisaatioiden välillä ja täten järjestelmä voidaan suunnitella tukemaan tätä tiettyä myyntiprosessin mallia. Myyntiprosessi mallinnetaan tarkemmin tutkimuksen myöhemmässä vaiheessa.

3.2.2 Muiden prosessien tukeminen

Järjestelmän tukee tukea myyntiprosessin lisäksi myös muita prosesseja. Näitä edustavat suunnittelun tässä vaiheessa ainakin seuraavat nähtävissä olevat prosessit:

- Kontaktien ja asiakkaiden hallinta:
  - Järjestelmän tulee tukea mahdollisuutta luoda kontakteja (suppeita asiakastietoja ilman laskutustietoja), luoda kontakteista asiakkaita (asiakastietu-

eessa on kaikki laskutukseen tarvittavat tiedot) ja muokata asiakkaiden tietoja.

- Tilauksen koostaminen eri tuotteista; monimutkaisimmillaan tämä on suurinpiirtein seuraavaa:
  - osa tuotteista tehdään itse;
  - osa tuotteista voi olla omassa varastossa; ja
  - osa tilataan muualta.
- Varastotietojen päivitys ja ylläpito (varastonhallinta).
- Tuotteiden lähettäminen asiakkaille (lähettämöprosessit).

3.2.3 Sekalaisia ominaisuuksia

Järjestelmän on syytä tukea ainakin joitakin edellisiin ryhmiin kuulumattomia ominaisuuksia. Näitä on muun muassa tarjouksien ja tilauksien selaus ja hakeminen mahdollisesti asiakkaan perusteella, asiakkaiden selaus ja hakeminen ja tuotteiden selaus ja hakeminen. Myös muut selaus- ja hakumahdollisuudet voivat olla tarpeellisia.

Yksi tarpeellinen ominaisuus on myyjän henkilökohtainen kalenteri, joka pitää kirjaa ja muistuttaa tulevista ja/tai käsittelemättömistä tapahtumista. Niitä ovat muun muassa voimet myyntimahdollisuukset eli ne myyntimahdollisuudet, joita ei ole merkitty loppuunkäsitellyksi. Tapahtumilla on määräaika (deadline), ja järjestelmän on syytä ilmoittaa käyttäjälle selkeästi määräajan lähestymisestä ja siitä myöhästymisestä. Kalenteri voi esimerkiksi näyttää tapahtumat tietyn määrän päiviä päähän ja ilmoittaa näkyvästi lähestyvistä määräajoista ja niitä vastaavista tapahtumista. Huomautusaikaa on voitava muuttaa tapahtumakohtaisesti.

Muita toteutettavia ominaisuuksia ovat ilmoitustaulu ja tekstiviestien lähetysjärjestelmä. Ilmoitustaulu on tarkoitettu organisaation sisäiseen viestintään, ja sen tavoitteena on lisätä asiakasorganisaatiossa järjestelmän käytön yhteisöllisiä piirteitä. Ainakin aluksi ilmoitustaulu voidaan toteuttaa siten, että se on kaikille avoin: kuka tahansa voi kirjoittaa ilmoitustaululle ja lukea sieltä. Tekstiviestien lähetysjärjestelmä toteutetaan, koska yrityksellä Y on vahva osaaminen tekstiviestipalveluista. Tämä palvelu on siis *technology-driven*, eli se toteutetaan teknologialähtöisesti olemassaolevaan osaamiseen perustuen. Tällä ominaisuudella myyjä voi lähettää asiakkaille tekstiviestejä nähdessään sen tarpeelliseksi, esimerkiksi ilmoittaessaan asiakkaalle tilauksen edistymisen eri vaiheissa asiakkaan pyynnöstä.

Järjestelmään on tarve luoda resurssikalenteri työvoimaresurssien hallintaan ja aikataulukseen. Kalenteriin merkitään työntekijöiden tulevat työt, kuten keikkaluontoiset vierailut asiakkaiden luona, ja lomat. Esimerkkejä resurssikalenteria tarvitsevista organisaatioista ovat esimerkiksi kiinteistönylläpitoyritys, jolla on paljon toistuvaa työtä, ja uuninasennusyritys, jolla jokainen työviikko on resurssiallokoinniltaan hyvin erilainen. Tästä voidaan päätellä, että näiden organisaatioiden tarve resurssikalenterille on hyvinkin erilainen. Läheskään kaikki järjestelmää käyttävät organisaatiot eivät tule resurssikalenteria tarvitsemaan tai tulevat tarvitsemaan sitä muista organisaatioista poikkeavalla tavalla, joten resurssikalenteria ei ole syytä mallintaa kovinkaan tarkasti tämän tutkimuksen puitteissa.

Yritys Y:n mukaan on tilausta yksinkertaiselle tuntiseuranta- ja kirjausominaisuudelle. Yksinkertaisimmillaan tämän ominaisuuden tai palvelun avulla kukin työntekijä ilmoittaa, kuinka paljon on tehnyt tunteja kullekin asiakkaalle. Syntyy siis vastaavuus, jossa kukin työtunti on jonkun työntekijän tekemä jollekin asiakkaalle. Työdataa voi selata esimerkiksi kalenterilla, jossa näkyy eri työntekijöiden menneisyydessä ilmoittamia työtunteja. Tämä toiminnallisuus on tarkoitettu asiakasorganisaation palkanmaksun ja työlaskutuksen avuksi.

3.2.4 Tuettavien prosessien räätälöimistarpeet

Tietojärjestelmän tulee olla helposti räätälöitävissä kunkin asiakasorganisaation tarpeisiin. Toistaiseksi tiedossa oleviin räätälöimistarpeisiin kuuluvat ainakin myyntiprosessin ja tuotantoprosessin räätälöinti sekä asiakaskohtaiseksi toteutettavat resurssikalenteri ja tuntiseuranta- ja kirjaus. Myyntiprosessi pyritään pitämään mahdollisimman samanlaisena eri asiakasorganisaatioiden välillä. Täten sen räätälöintitarve on pieni. Tuotantoprosessien räätälöintitarpeet ovat mahdollisesti suuria. Näillä näkymin tuotantoprosessi joudutaan räätälöimään kullekin asiakasorganisaatiolle erikseen, koska eri asiakasorganisaatioiden tuotantoprosessit voivat erota toisistaan hyvinkin radikaalisti. Täten tuotantoprosessien mallintaminen tässä vaiheessa ei ole mielekästä, vaan se toteutetaan myöhemmin yhteistyössä todellisten ja konkreettisten asiakasorganisaatioiden kanssa. Järjestelmän suunnittelussa on valmistauduttava liki kaikkien ominaisuuksien räätälöintiin: esimerkiksi resurssikalenteri voi keskenään olla hyvin erilainen jo edellä mainitun esimerkin, kiinteistönylläpito- ja uuninasennusyrityksen, välillä.

3.2.5 Potentiaaliset laajennustarpeet

Tulevaisuudessa järjestelmä voi kohdata laajennustarpeita. Yksi potentiaalinen tarve on mahdollisuus integroida Ratas-järjestelmä muihin, jo olemassa oleviin järjestel-

miin, tai laajentaa Ratas kattamaan tuotannonohjausta. Laajennuskohteena olisi tässä tapauksessa tuotannon- ja varastonhallinta. Toinen laajennussuunta on projektiorganisaation tuotannonhallinta, joka sisältää tuntien kirjaamisen esimerkiksi eri virstanpylväiden tai projektin vaiheiden välillä. Tällä seurataan, pysytäänkö aikataulu- ja kustannusarvioissa ja -rajoissa. Tavoitteena on kaiken organisaation tekemän työn puristaminen yhteen tietovarastoon, jonka perusteella voidaan muodostaa asiakkaalle lasku ja organisaation sisäinen palkanmaksu. Tätä kutsutaan projektiorganisaation hallintajärjestelmäksi. Kolmas tarvittava laajennos järjestelmään on myyntiprosessin tukeminen niin, että järjestelmä tukee laskun välittämistä suoraan kirjanpitojärjestelmään. Tämä tarkoittaa laskutuksen integrointia kirjanpitojärjestelmään. Näitä ominaisuuksia ei tarvitse suunnitella vielä tässä vaiheessa, mutta nämä potentiaaliset kasvutarpeet on syytä ottaa huomioon järjestelmän suunnittelussa.

4 JÄRJESTELMÄN SUUNNITTELU

Tässä luvussa esitellään järjestelmän suunnittelun prosessi pääpiirteissään. Luku on jaoteltu Marchin ja Smithin [1995] informaatiotieteiden suunnittelutieteellisen tutkimuksen eri vaiheiden tutkimustuotosten mukaan. Niitä ovat käsitteet, mallit, menetelmät ja toteutukset. Tällä on vastaavuus myös Schekkermanin [2004] kokonaisarkkitehtuuriajattelussa: liiketoimintatasoa edustaa edellisessä luvussa määritellyt liiketoimintatarpeet, tietotasoa vastaavat määriteltävät käsitteet ja niiden väliset suhteet (mallit), tietojärjestelmätasoa menetelmät ja toteutukset sekä teknologiatasoa toteutusvaiheen teknologiaosuus.

4.1 Käsitteet

Tässä kohdassa määritellään järjestelmän sovellusalueen kannalta olennaiset käsitteet mahdollisimman tarkasti ja yksiselitteisesti. Järjestelmän suunnittelun kannalta ehkä tärkeimmät käsitteet ovat toimeksiantaja ja asiakasorganisaatio. Toimeksiantaja eli yritys Y on järjestelmän suunnitteleva ja toteuttava organisaatio, jonka tarpeisiin tämä tutkimus tehdään ja jonka myymäksi tuotteeksi tutkimuksen tuotokset edelleen kehitetään. Asiakasorganisaatio on yritys Y:n asiakas, joko potentiaalinen tai todellinen, jonka käyttöön ja tarpeita vastaamaan järjestelmä suunnitellaan.

4.1.1 Asiakkuudenhallinta

Tässä alakohdassa esitellään asiakkuudenhallinnan riskit ja mahdollisuudet. Asiakkuudenhallinta on määritelty johdannossa. Chenin ja Popopvichin [2003] mukaan asiakkuudenhallintajärjestelmien etuja ovat ainakin seuraavat:

- Asiakkuudenhallintajärjestelmien käyttö on parantanut asiakasuskollisuutta ja -tyytyväisyyttä, lisännyt kilpailukykyä, kasvattanut voittoja ja alentanut operatiivisia kustannuksia monissa organisaatioissa.

- Asiakkuudenhallintajärjestelmät tarjoavat valtavasti tietoa asiakkaiden tottumuksista ja mieltymyksistä täydellisenä ja integroituna näkymänä asiakkaasta.

- CRM-sovellukset auttavat organisaatioita arvioimaan asiakasuskollisuutta ja tuottavuutta eri kriteereillä, kuten toistuvien ostojen, käytetyn rahamäärän ja pitkäikäisyyden (engl. longevity) perusteella, mikä mahdollistaa organisaation entistä tehokkaamman kommunikoinnin asiakaspalvelun tason, asiakasuskollisuu-

den ja markkinoinnin kohdentamisen – eri asiakassegmenteille tai jopa yksittäisille asiakkaille – parantamiseksi.

- Internetissä yritykset voivat hyödyntää asiakassuhteista kerättyä dataa ostokokemuksen mukauttamiseen asiakkaan tarpiden mukaan, arvioida kunkin asiakkaan yritykselle tuomaa taloudellista hyötyä, entistä paremmin ennakoimaan tulevaa ostokäyttäytymistä, houkutella asiakkaita erikoistarjouksilla, rakentaa kumpaakin osapuolta hyödyttävää suhdetta ja parantaa mahdollisuuksia itsepalveluun.

Chenin ja Popovichin [2003, s. 673] mukaan CRM-sovellukset auttavat vastaamaan muun muassa kysymyksiin "Mitkä tuotteet tai palvelut ovat tärkeitä asiakkaillemme?"ja "Miten meidän tulisi kommunikoida asiakkaidemme kanssa?". Asiakkaat hyötyvät mielikuvasta, että he säästävät aikaa ja rahaa ja saavat parempaa informaatiota ja erityiskohtelua. Riippumatta asiakkaan käyttämästä kosketuspinnasta yritykseen saa asiakas samaa tehokasta ja yhdenmukaista palvelua.

Asiakkuudenhallinnan käyttöönotto ei ole vailla riskejä. Ohjelmistotoimittajat voivat houkutella organisaatioita lupauksilla kaikenkattavista sovelluksista, vaikka täysin kattavia ratkaisuja ei olekaan vielä olemassa. On otettava huomioon riskit kuten projektin epäonnistuminen, riittämätön sijoitetun pääoman tuotto, suunnittelemattomat muutokset projektin budjettiin, tyytymättömät asiakkaat, henkilöstön luottamuksen menetys ja johdon ajan ja resurssien väärä suuntaaminen. On syytä varmistua käyttäjien hyväksynnästä (engl. user acceptance) eli että järjestelmän suunnitellut käyttäjät alkavat käyttää järjestelmää suunnitellulla tavalla. Tästä on esimerkkinä suuren ICT-yrityksen esittelemä 10 000 dollaria käyttäjältä maksanut CRM-järjestelmä, jonka tuhannesta suunnitellusta käyttäjästä vain alle sata käyttivät järjestelmää. Pitkittyneet tai epäonnistuneet CRM-projektit ovat usein seurausta siitä, ettei ymmärretä, mitä CRM-hankkeet asiakasorgaanisaatiolta vaativat ja mitä ne tarjoavat. CRM vaatii koko organisaation laajuista eri toimintoja yhdistävää (engl. cross-functional) liiketoimintaprosessien uudelleensuunnittelua. Vaikka suuri osa CRM:stä on teknologiaa, onnistunut CRM-järjestelmän käyttöönotto vaatii CRM:n näkemisenä myös muuna kuin pelkkänä teknologisena ratkaisuna. [Chen & Popovich, 2003]. Apicellan ja muiden [1999] mukaan 65 prosenttia CRM-hankkeista epäonnistuu ja sitoutuminen tiettyyn CRM-ratkaisuun sisältää riskejä.

4.1.2 Tuotannonohjaus

Läheisesti CRM:ään liittyvä käsite on tuotannonohjaus (engl. enterprise resource planning eli ERP). Kirjallisuudessa on esitetty lukuisia eriäviä määritelmiä ERP-käsitteelle,

eikä tarkkaa yhteisymmärrystä käsitteen sisällöstä ole [Klaus *et al.*, 2000]. ERP tarkoittaa tietyn tyyppistä ohjelmistoa. Estevesin ja Pastorin [2001] mukaan ERP-järjestelmät ovat lukuisista moduuleista, kuten työntekijöistä, myynneistä, taloushallinnosta ja tuotannosta, koostuvia organisaation eri toimintojen tietoja businessprosessien kautta yhdistäviä järjestelmiä, jotka voivat olla räätälöityjä organisaation käyttöön. Klaus ja muut [2000] määrittelevät ERP:n kokonaisvaltaisiksi paketoiduiksi ohjelmistoratkaisuksi, jotka pyrkivät kattamaan koko organisaation businessprosessit -ja toiminnot. Pyrkimyksenä on kokonaisvaltainen näkemys organisaatiosta saman tieto- ja IT-arkkitehtuurin puitteissa. Kumarin ja van Hillegersbergin [2000] mukaan ERP-järjestelmät ovat ovat konfiguroitavia ohjelmistoja, jotka integroivat tietoja ja tietopohjaisia prosessoja organisaation eri toimintojen sisällä ja välillä. Hyödynnän näitä keskenään varsin samansisältöisiä käsitteiden määrittelyitä tämän tutkimuksen puitteissa.

Chenin ja Popovichin [2003] mukaan tuotannonohjausjärjestelmät eroavat CRM-järjestelmistä ainakin seuraavin tavoin:

- ERP tarjoaa perustan tiukasti integroiduille taustaoperaatioille; CRM edustaa linkkiä asiakasrajapinnan ja yrityksen taustaprosessien välillä asiakassuhteiden ja -tyytyväisyyden ylläpitämiseksi.

- ERP integroi kaikki liiketoiminnan toiminnalliset alueet alihankkijoiden ja asiakkaiden kanssa; CRM kehittää asiakasrajapinnan ja asiakkaan kosketuspisteiden toimntaa parempaan asiakastyytyväisyyteen ja tuottavuuteen pääsemiseksi.

- ERP yrittää ratkaista pirstaloituneiden tietojärjestelmien ongelman, CRM pirstaloituneen asiakasdatan ongelman.

Aiemmin esitellyistä yritys Y:n asiakasvaatimuksista voidaan huomata, että suunniteltavassa järjestelmässä on ERP-määritelmien mukaisia piirteitä erityisesti silloin, kun myyntiprosessien lisäksi tuotantoprosessien ja koko yrityksen tuottama informaatiosisältö integroidaan yhteen. Täten suunnitteltavaa Ratas-järjestelmää ei voi pitää puhtaana asiakkuudenhallintajärjestelmänä, muttei toisaalta tuotannonohjausjärjestelmänäkään.

4.1.3 Ohjelmisto palveluna

Kun tarjotaan sovellus asiakasorganisaatiolle palveluna (engl. Software as a Service eli SaaS), asiakas käyttää palveluntarjoan ylläpitämää ohjelmistoa tämän laitteistolla

etänä Internetin yli [Campbell-Kelly, 2009]. Läheisesti SaaS:n liittyvä käsite on ASP-palvelu (engl. application service provision). Siinä paketoidun ohjelmiston myyvä yritys vastaa sen käyttöönotosta, ylläpidosta ja hallinnosta ja toimittaa sen keskitetystä palvelinkeskuksesta verkon yli asiakkaalle [Bennett *et al.*, 2001].

4.1.4 Järjestelmän sovellusalueen käsitteet

Tässä alakohdassa esitellään järjestelmän sovellusalueen eli asiakkuudenhallinnan ja asiakkuudenhallintajärjestelmän kannalta olennaiset käsitteet. Järjestelmän kannalta olennaisia käsitteitä ovat ainakin seuraavat:

- *Tuote* on joko aineeton (kuten palvelu tai tieto) tai aineellinen (fyysisessä maailmassa olemassa oleva) hyödyke. Tuote on jotain, jota myydään asiakkaalle.

- *Yritys* on tuotteita asiakasorganisaatiolta (potentiaalisesti) ostava organisaatio. Yrityksiä ovat liikeyritykset, yhteisöt ja julkisoikeudelliset tahot (julkisen sektorin tahot). Järjestelmässä yritys on yritystietue, joka sisältää laskutuksen kannalta olennaiset tiedot. Yrityksellä voi olla useita osoitteita, joista jotkut voidaan asettaa oletusarvoisiksi käynti-, toimitus- ja laskutusosoitteiksi. Yrityksellä on kontaktihenkilöitä. Kauppaa hierotaan yrityksen kontaktihenkilön, ei itse yrityksen, kanssa. Tarjoukset ja tilaukset tehdään kuitenkin yritykselle niiden virallisen luonteen johdosta.

- *Yritysasiakas* on yritys, joka on tehnyt tilauksen asiakasorganisaatiolta tai potentiaalisesti on sellaisen tulevaisuudessa tekevä.

- *Yksityishenkilö* on tuotteita asiakasorganisaatiolta (potentiaalisesti) ostava luonnollinen henkilö.

- *Yksitysasiakas* on yksityishenkilö, joka on tilannut tai potentiaalisesti tilaava jotain asiakasorganisaatiolta.

- *Oikeussubjekti* on järjestelmässä yritys tai yksityishenkilö, eli siis yrityksen ja yksityishenkilön yläluokka.

- *Asiakas* on yritysasiakkaan ja yksityisasiakkaan yläluokka; käytetään puhuttaessa tarkemmin määrittelemättä, onko kyseessä yritys- vai yksityisasiakas tai kun konteksti on selvää, kummasta on kysymys.

- *Kontaktihenkilö* on palveluita ja/tai tuotteita asiakasorganisaatiolta potentiaalisesti ostavan yrityksen edustaja, ja kontaktihenkilö viittaa aina johonkin yritykseen. Järjestelmässä kontaktihenkilö on yrityksen edustajan nimi ja yhteystieto tai -tiedot kuten puhelinnumero ja/tai sähköpostiosoite.

- *Liidi* on yritys tai yksityishenkilö, joka on potentiaalisesti kiinnostunut ostamaan asiakasorganisaation tarjoamia tuotteita, tai tämän tahon nimi ja yhteystieto tai -tiedot kuten puhelinnumero ja/tai sähköpostiosoite. Liidi edustaa myyntiprosessin ensimmäistä vaihetta. Liidillä on lähde, eli mistä on saatu tieto tästä potentiaalisesta asiakkaasta. Lähde voi olla esimerkiksi myyjä (keksii ympäristöstään, esimerkiksi lehdistä tai messuilta), myyjän tuttava, ostettu yritysten yhteystietolista, peräisin asiakkaan lähettämästä yhteydenottopyynnöstä esimerkiksi webitse tai muusta kommunikaatiosta ("osoitetietojasi voidaan käyttää suoramarkkinointiin").

- *Myyntimahdollisuus* sisältää kaikki tiedot, joita tarvitaan tarjouksen tekoon. Käytännössä myyntimahdollisuus sisältää tiedot tarvitsevasta tahosta (kuka tarvitsee?) ja tarpeen luonteesta (mitä tarvitaan?). Myyntimahdollisuus edustaa yhtä kaupankäyntiprosessia, joka voidaan jättää kesken myöhemmin täytettäväksi, voidaan hävitä tai voidaan voittaa. Viimeisin tapahtuu, kun myyntimahdollisuudesta luodaan tarjous, jonka asiakas edelleen hyväksyy tilaukseksi. Myyntimahdollisuudella on tila (esimerkiksi avoin, suljettu) ja, mikäli kauppa on hävitty, syy kaupan häviämiseen.

- *Osoite* on yrityksen tai yksityishenkilön osoite. Osoite voi olla käynti-, toimitus- ja/tai laskutusosoite. Järjestelmässä osoite koostuu yrityksen tai yksityishenkilön nimestä, katuosoitteesta, postinumerosta ja niin edelleen.

- *Tarjous* on yritykselle tai yksityishenkilölle tehtävä esitys myytävistä tuotteista, niiden kustannuksista ja toimitusehdoista. Tarjous tiedot siitä, mitä tarjotaan (lista tarjoukseen liittyvistä tuotteista), kenelle tarjotaan (yritys tai yksityishenkilö ja yrityksen tapauksessa myös yrityksen kontaktihenkilö) ja millä ehdoilla (toimitusehdot, hinta).

- *Tilaus* on aktualisoitunut tarjous eli sopimus järjestelmää käyttävän organisaation ja asiakkaan välillä tarjouksen mukaisten tuotteiden toimituksesta asiakkaalle tarjouksen mukaisilla ehdoilla.

- *Myyntitapahtumat* edustavat myyntiprosessin vaiheita dokumentoituna järjestelmään. Näitä ovat puhelut, sähköpostiviestit ja muut yhteydenotot joko yksityishenkilöön tai yrityksen kontaktihenkilöön.

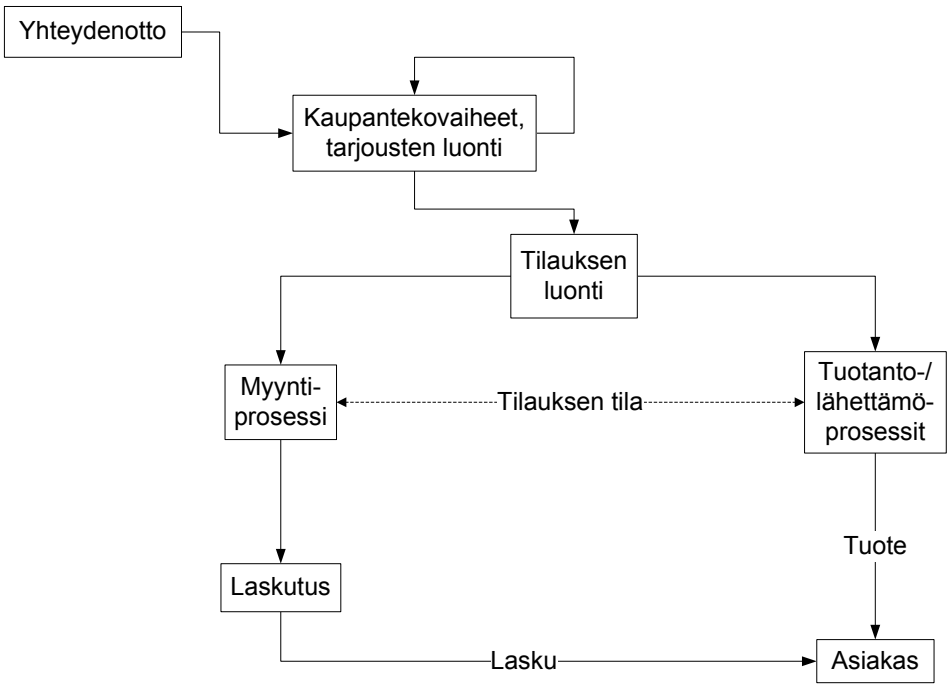
4.2 Mallit

Tässä kohdassa mallinnetaan myyntiprosessi, jonka ympärille järjestelmä luodaan, ja tuotetaan määriteltyihin käsitteisiin ja myyntiprosesikaavioihin perustuva tietomalli. Prosessimalli kuvataan prosessikaavioina (engl. process model). Tietomallinnukseen käytetään ER-kaaviota (engl. entity-relationship diagram) [Chen, 1977].

4.2.1 Myyntiprosessin kuvaus

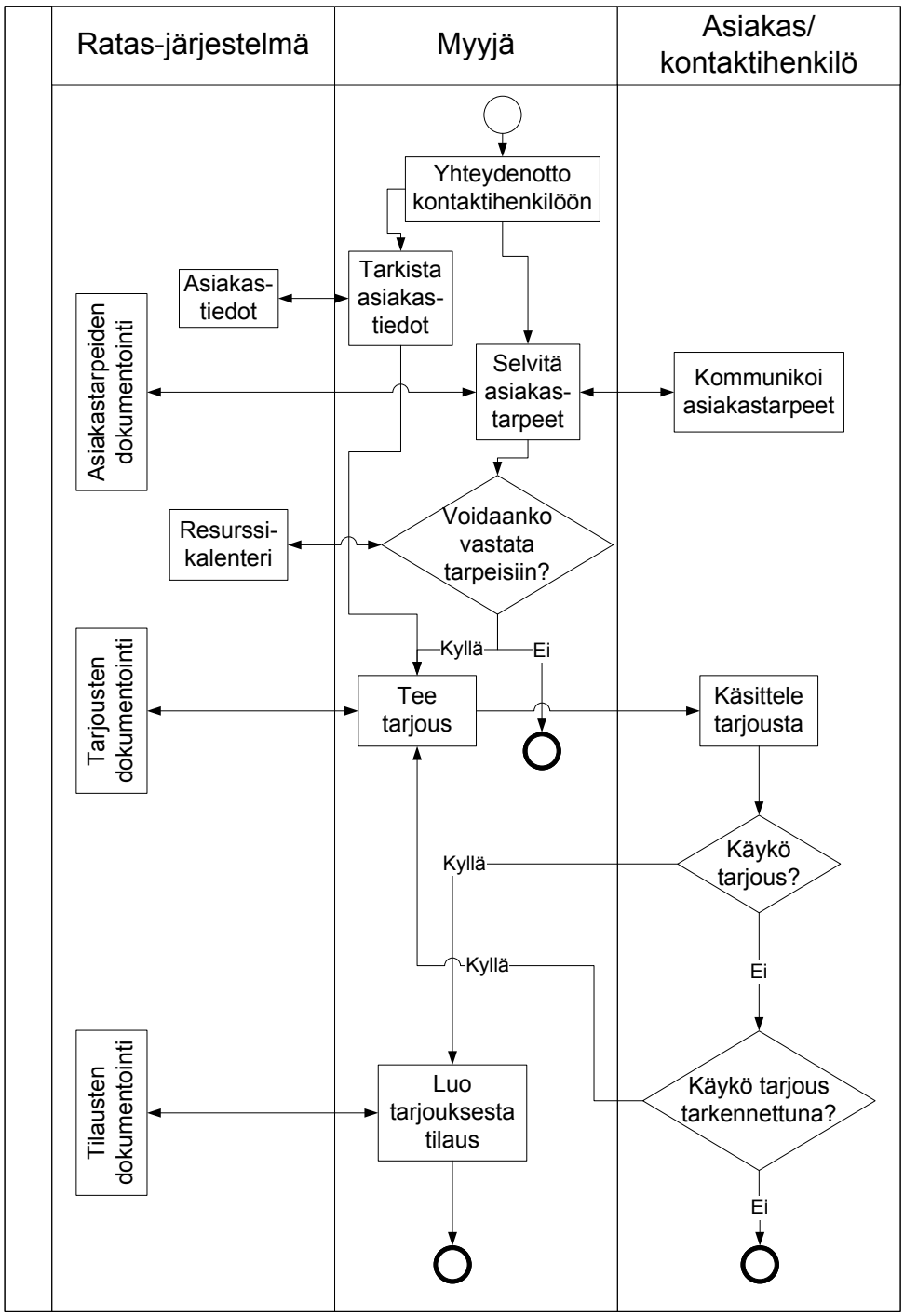
Kuvassa 1 on kuvattu järjestelmän rajaus myyntiprosessin ja siihen läheisesti liittyvien prosessien osalta. Myyntiprosessi alkaa yhteydenotosta. Kauppaa hierottaessa kommunikoidaan eri kanavilla, kuten puhelimitse tai sähköpostitse. Asiakkaalle luodaan tarjouksia. Kun asiakas hyväksyy tarjouksen, siitä syntyy tilaus. Tästä prosessi jakautuu kahtia, ikään kuin kahteen "reittiin": tuotanto- ja toimitusprosessiin sekä myyntiprosessiin. Tuotanto- ja toimitusprosessin tavoitteena on toimittaa asiakkaalle tuote. Myyntiprosessin rooli tässä vaiheessa on toimia asiakkaan kontaktipintana asiakasorganisaatioon. Asiakas voi ottaa yhteyttä myyjään tiedustellakseen tilauksensa tilaa, ja myyjä selvittää sen tuotanto- ja toimitusprosessilta. Toimituksen ja laskutuksen tarkempi mallintaminen jätetään myöhemmäksi eikä sitä tehdä tämän tutkimuksen puitteissa.

Myyntiprosessin edeltävänä vaatimuksena on asiakaskontaktin syntyminen. Tämä voi tapahtua miettimällä tai keksimällä, jolloin myyjä keksii potentiaalisen asiakkaan. Toisaalta joku ulkopuolinen voi kertoa potentiaalisesta asiakkaasta (liidi). On myös mahdollista, että asiakas ottaa yhteyttä myyjään. Joka tapauksessa myyntiprosessi alkaa asiakaskontaktista: joko myyjä ottaa yhteyttä asiakkaaseen tai asiakas myyjään. Myyntiprosessi on kuvattu kuvassa 2. Yhteydenoton jälkeen selvitetään asiakkaan tarpeet. Mikäli asiakkaan tarpeisiin ei kyetä vastaamaan (esimerkiksi asiakas haluaa ostaa tuotteen, jota emme myy), ohjataan asiakas toisaalle ja myyntiprosessi päättyy. Mikäli asiakkaan tarpeisiin voidaan vastata, laaditaan suunnitelma ja jatketaan. Myyjä voi myyntiprosessin edetessä luoda tarjouksen tai tarjouksia. Kunkin tarjouksen jälkeen asiakas pui tarjousta. Mikäli tarjous vaatii tarkennusta, myyjä palaa tarjouksen luontiin eli luo uuden tarjouksen, ja usein pohjana on vanha tarjous pohjana. Tarjouksia voi siis syntyä myyntiprosessin aikana monia niin rinnakkain ja peräkkäinkin. Mikäli asiakkaalle ei tarjous kelpaa tarkennuksienkaan jälkeen, myyntiprosessi loppuu. Jos



Kuva 1 Myyntiprosessin rajaus.

asiakkaalle ei kelpaa mikään tehty tarjous, on sille aina jokin syy. Näitä syitä ovat ainakin tarjouskilpailun häviäminen (asiakas sai muualta paremman tarjouksen), tuotteen maksaminen asiakkaalle liikaa (ei kannata asiakkaalle) tai jokin muu syy. Tämä tieto on syytä kirjata järjestelmään. Jos asiakas kuitenkin hyväksyy tarjouksen eli tekee tilauksen, kirjataan se järjestelmään ja luodaan tilauslasku tai -laskut. Myyjän on syytä kirjata myyntiprosessin jokaisessa vaiheessa jokainen yhteydenotto potentiaalisen asiakkaan kanssa ja siitä aiheutuvat muutokset muistiinpanoihin ja syntyviin tarjouksiin. Näin tieto säilyy koko organisaation käytössä myös tulevaisuudessa.



Kuva 2 Myyntiprosessin kuvaus.

4.2.2 Tietomalli

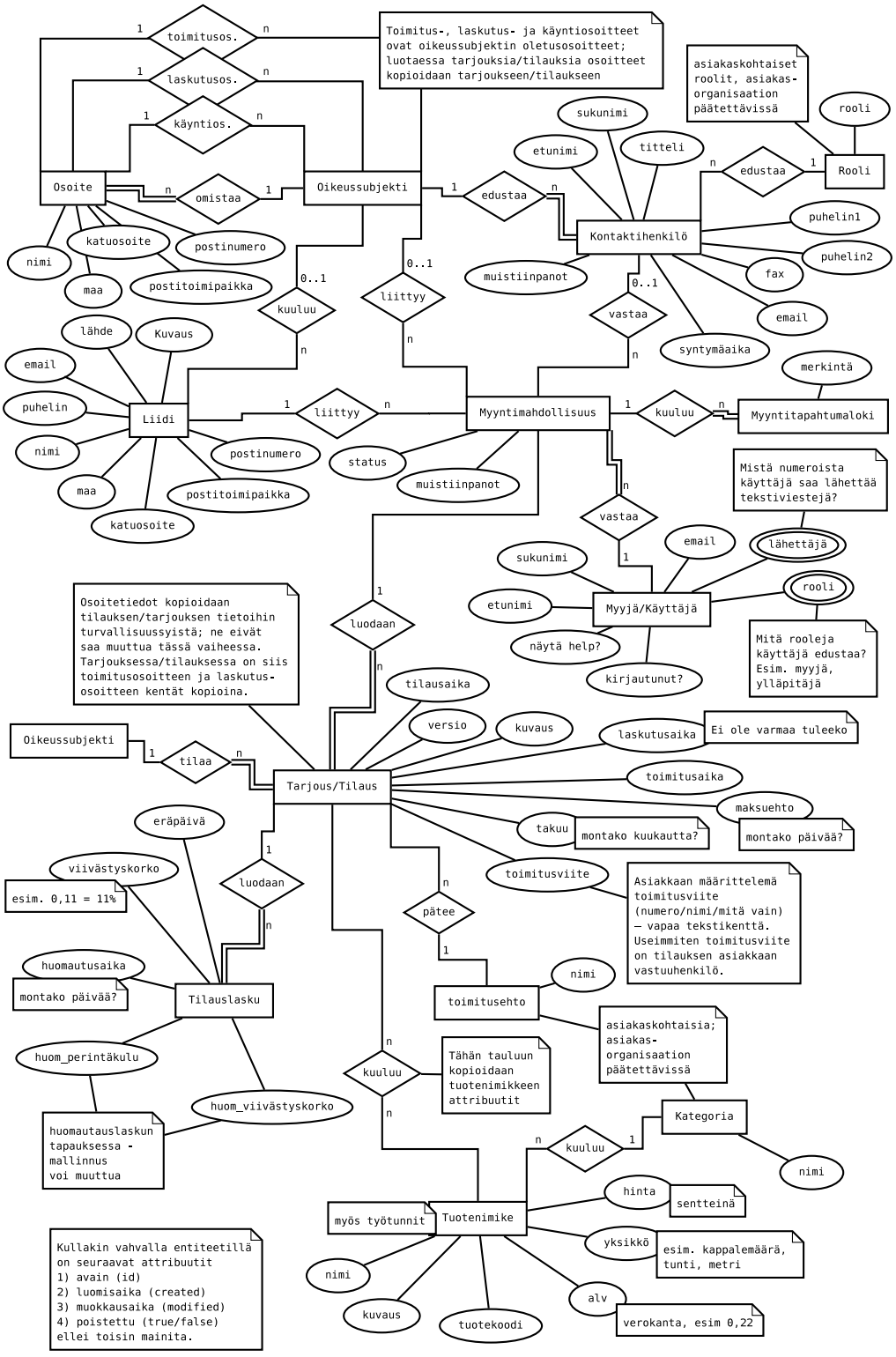
Rattaan sisältämä tieto mallinnetaan ER-kaaviolla [Chen, 1977]. Kullakin vahvalla entiteetillä on pääavain (engl. primary key, attribuutin nimenä id), mikäli kyseessä ei ole monesta moneen -suhteen liitostaulu. Muita attribuutteja vahvoilla entiteeteillä ovat luomisaika, muokkausaika ja boolean-tyyppinen poistettu-attribuutti. Tietokannasta ei siis tarvitse poistaa rivejä, vaan ne voidaan merkitä poistetuksi muokkaamalla poistettu-attribuutin arvoa. Tietomalli on esitetty kuvissa 3 ja 4.

Myyntiprosessin tietomalli rakentuu myyntimahdollisuuden ympärille. Myyntimahdollisuus edustaa kaupanhierontaa ja sisältää lopulta kaikki tiedot, joita tarjouksen tekemiseen tarvitaan. Käytännössä myyntimahdollisuus sisältää tiedot siitä sekä tarpeen luonteesta ja tarvitsijasta.

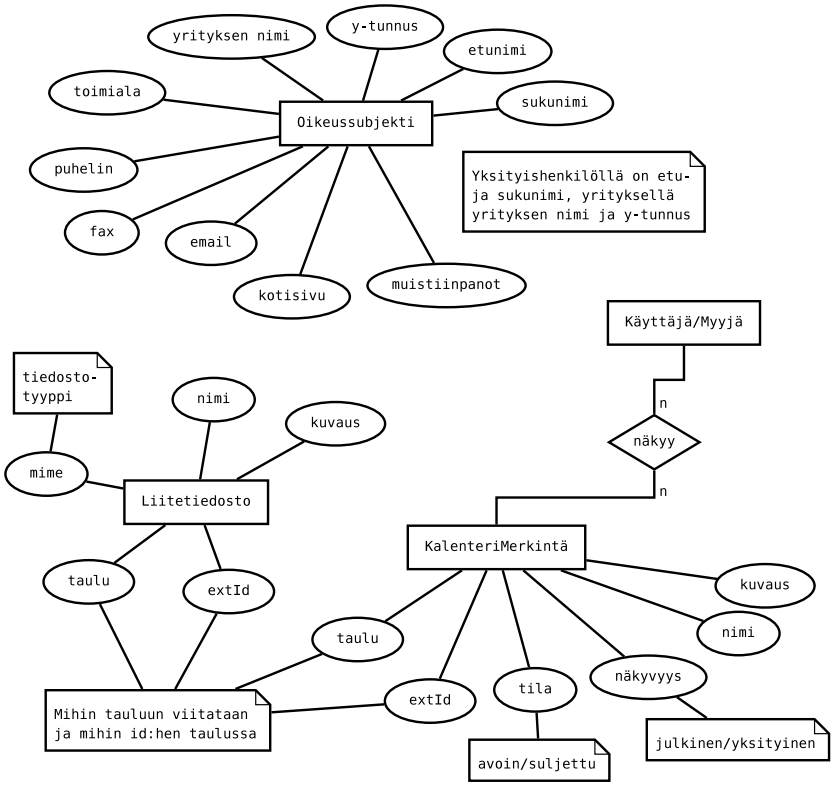
Oikeussubjekti on taho, jolla on tarve. Oikeussubjektilla voi olla useampia osoitteita ja sille voidaan määritellä oletusarvoiset toimitus-, laskutus ja käyntiosoitteet (yksi kutakin). Mikäli oikeussubjekti on yritys, sillä on y-tunnus, nimi ja toimiala; mikäli oikeussubjekti on yksityishenkilö, sillä on etu- ja sukunimi. Muita oikeussubjektin attribuutteja ovat puhelin- ja fax-numerot, sähköpostiosoite, kotisivu ja muistiinpanot. Oikeussubjektilla voi olla kontaktihenkilöitä (luonnollisia henkilöitä), joiden kanssa kauppaa käytännössä hierotaan. Kontaktihenkilö edustaa siis sitä henkilöä, johon yrityksessä otetaan yhteyttä kaupankäynnin yhteydessä. Yksi kontaktihenkilö vastaa oikeussubjektin puolesta myyntimahdollisuuden hoitamisesta eli on oikeussubjektin puolesta kaupankäynnin vastuuhenkilö kyseessä olevassa kaupanhieronnassa. Kontaktihenkilö voi edustaa jotain asiakasorganisaation määrittämää roolia. Mikäli oikeussubjekti on yksityishenkilö, ei kontaktihenkilön määrittely ole mielekästä.

Muistiinpanot ja myyntitapahtumaloki kuvaavat tietoa siitä, mitä tarvitaan. Lokiin voi tallentaa merkintöjä myyntiprosessista tiettynä ajanhetkenä, eli tallennetaan lokimerkintä ja aikaleima. Muistiinpanoihin tallennetaan myyntiprosessin merkintöjä, joihin ei liity yksiselitteistä aikaleimaa. Myyntimahdollisuudella on vastuullinen myyjä, joka vastaa myyntimahdollisuuden viemisestä eteenpäin ja siten myyntimahdollisuuden ja sen tietojen hallinnasta, tarjousten teosta ja välittämisestä oikeussubjektille, tilauksen käsittelystä ja niin edelleen. Myyntimahdollisuuden vastuumyyjä on aina myyntimahdollisuuden luoja. Myyntimahdollisuudella on status, jota myyjä voi muokata. Statuksia ovat avoin ja suljettu; muita voidaan lisätä järjestelmään tarvittaessa.

Myyntimahdollisuus luodaan liidistä, potentiaalisen asiakkaan yhteystiedosta. Täten myyntimahdollisuus liittyy aina johonkin liidiin. Mikäli myyntimahdollisuus luodaan ilman liidiä (esimerkiksi asiakkaan ottaessa yhteyttä), luodaan järjestelmässä automaattisesti liidi, jonka lähteeksi merkitään asiakasyhteydenotto tai vastaava. Liidillä



Kuva 3 Järjestelmän tietomalli ER-kaaviona.



Kuva 4 Tarkennuksia tietomallin oikeussubjekti-, liitetiedosto- ja kalenterimerkintäkäsitteisiin.

on siis aina jokin lähde, josta se on peräisin, ja järjestelmään voidaan lisätä uusia lähdevaihtoehtoja tarvittaessa.

Liidillä voi olla oletusoikeussubjekti, johon se liittyy. Kun tällöin liidistä luodaan myyntimahdollisuus, voidaan oletusoikeussubjekti liittää myyntimahdollisuuteen automaattisesti. Tämä mahdollistaa sen, että liidien tietoja voidaan hyödyntää myöhemmissä myyntimahdollisuuksissa. Liidillä voi olla myös oletusarvoiset osoitetiedot. Näitä voidaan käyttää oikeussubjektin oletusosoitteena luotaessa liidistä myyntimahdollisuus ja sitä kautta myyntimahdollisuuden oikeussubjekti, mikäli oikeussubjektia ei ole vielä olemassa. Osoitetiedot eivät ole viittauksia osoite-tauluun, jotta niitä ei tahattomasti muokata oikeussubjektin osoitteen muokkaamisen yhteydessä.

Kun samasta liidistä on mielekästä luoda monta myyntimahdollisuutta (esimerkiksi saman asiakkaan kanssa halutaan hieroa samanaikaisesti useaa kauppaa), voidaan myyntimahdollisuus monistaa. Tällöin myyntimahdollisuuden asiakastiedot (oikeussubjekti, kontaktihenkilö ja liidi) kopioituvat uuteen myyntimahdollisuuteen, mutta tarpeiden kuvaus (muistiinpanot ja myyntitapahtumaloki) jätetään kopioimatta. Myyn-

timahdollisuuden vastuumyyjä on myyntimahdollisuuden monistaja eli myyntimahdollisuuskopion luoja. Vastuumyyjää voi kuitenkin muuttaa jälkikäteen.

Myyjä on samalla järjestelmän käyttäjä, eli luonnollinen henkilö, jolla on tunnus järjestelmään. Käyttäjällä/myyjällä voi olla useita rooleja (esimerkiksi myyjä, ylläpitäjä) ja sille voidaan asettaa käyttjäkohtaisia asetuksia. Käyttäjä voi myös järjestelmän tekstiviestinlähetysosiolla lähettää viestejä tietokannassa määritellyistä numeroista.

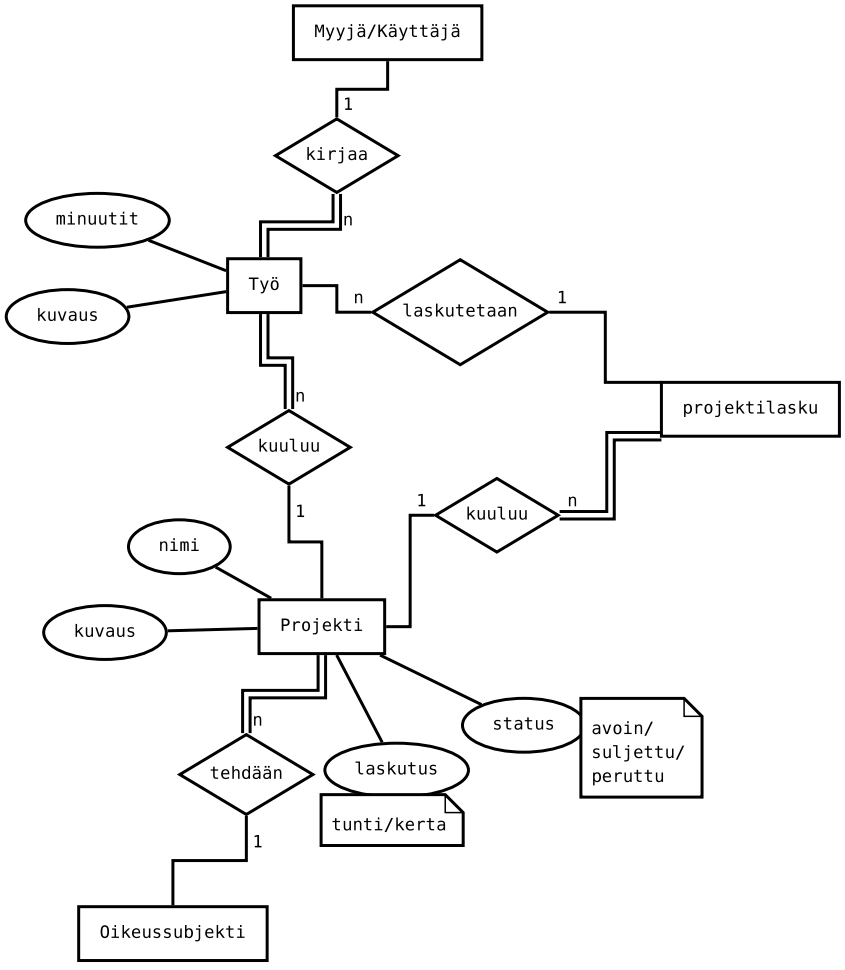
Myyntimahdollisuuden aikana myyjä voi luoda asiakkaalle tarjouksia. Kun asiakas hyväksyy tarjouksen, siitä tulee tilaus. Tietojen, kuten oikeussubjektin nimen, mahdollinen Y-tunnuksen, osoitetietojen yms. oikeellisuus on tärkeää tarjouksessa ja erityisesti tilauksessa. Täten oikeussubjektin tiedot kopioidaan tarjouksen/tilauksen tietoihin, koska ne eivät saa muuttua tarjouksen tekemisen jälkeen. Tarjouksella on versio, jota inkrementoidaan tarjousta muokatessa. Tämä on tarkoitettu tarvittaessa tarjouksen yksiselitteiseen tunnistamiseen. Tilauksella on tilausaika, joka jonka asettaminen tekee tarjouksesta tilauksen. Muita attribuutteja ovat kuvaus, laskutusaika, toimitusaika, maksuehto (montako päivää aikaa maksaa?), takuu (montako kuukautta voimassa?), toimitusehto ja toimitusviite. Näistä viimeisin kenttä on yritysasiakkaan määrittelemä vapaa tekstikenttä, ja useimmiten se on esimerkiksi asiakkaan tilauksesta vastaava henkilö.

Tuotenimikkeet ovat tuotteita tai työtunteja, joita voidaan liittää tarjoukseen/tilaukseen. Tuotenimike kuuluu johonkin asiakasorganisaation määrittelemään kategoriaan. Muita tuotenimikkeen attribuutteja tuotteen nimi, kuvaus, yksikkö (esimerkiksi kappalemäärä, metri tai tunti), hinta (sentteinä), arvonlisäverokanta (esimerkiksi 0,22 eli 22%) ja asiakasorganisaation määrittelemä tuotekoodi. Tuotenimikkeet liitetään monesta moneen -suhteella tarjoukseen/tilaukseen. Syntyvään yhteystauluun kopioidaan tuotenimikkeiden tiedot, koska ne eivät saa muuttua jälkikäteen. Poikkeuksena muista vastaavista yhteystaulusta on tällä taululla oma id, koska tuotenimikkeet voivat muuttua tai niitä voidaan järjestelmästä jopa poistaa.

Kun tarjous muuttuu tilaukseksi, siitä voidaan luoda tilauslaskuja. Tilauslaskulla on huomautusaika päivissä, tietty viivästyskorko (esimerkiksi 0,11 eli 11%) ja eräpäivä. Mikäli tilauslasku on huomautuslasku, eli aiempaa laskua ei ole maksettu ajoissa ja tehdään uusi lasku, määritellään lisäksi huomautuslaskun perintäkulu ja viivästyskorko.

Liitetiedostot ovat käyttäjien järjestelmään liittämiä tiedostoja. Liitetiedostoilla on nimi, kuvaus ja MIME-tiedostotyyppi (esimerkiksi "text/plain") [Borenstein & Freed, 1993]. Liitetiedostoja voi liittää järjestelmän eri osiin. Mielekkäitä liitoskohtia ovat ainakin myyntimahdollisuus, käyttäjä, asiakas ja kontakti. Liitos toteutetaan siten, että liitetiedostolla on attribuutteina entiteetti (taulu), johon tiedosto liitetään, ja id-kenttä

tietyssä taulussa. Kalenterimerkinnät ja niiden viittaukset järjestelmän eri entiteetteihin mallinnetaan samaan tyyliin. Tällöin mielekkäitä liitoskohtia ovat esimerkiksi myyntimahdollisuus, liidi, kontaktihenkilö ja oikeussubjekti. Sinänsä tämä tietomalli ei rajaa liitoskohtia, vaan niitä voidaan lisätä ja poistaa tarvittaessa. Kalenterimerkinnällä voi olla nimi, kuvaus, tila (avoin tai suljettu) ja näkyvyys (yksityinen tai julkinen). Lisäksi yhdistetään kalenterimerkintään ne käyttäjät, joille merkintä näkyy. Kalenterimerkintäentiteetti mahdollistaa myyjän/käyttäjän henkilökohtaisen kalenterin, joka muistuttaa tulevista tai käsittelemättömistä tapahtumista, ja myös näiden tapahtumien jaon muiden käyttäjien kanssa.



Kuva 5 Tuntikirjauksen tietomalli.

Tuntikirjauksen ja -seurannan mahdollistavat työ-, projekti- ja projektilaskuentiteetit. Näiden käsitteiden väliset suhteet on esitetty kuvassa 5. Tuntikirjauksen ja -seurannan tarkoitus on mahdollistaa pysyminen perillä siitä, mille asiakkaalle on teh-

ty tunteja milloinkin ja näiden tuntien laskutuksesta. Käyttäjä voi kirjata työaikaa eri asiakkaiden eri projekteille. Työllä on kuvaus ja pituus minuutteina. Lisäksi työ kuuluu aina johonkin projektiin. Projektia tehdään aina jollekin oikeussubjektille. Projektilla on nimi, kuvaus laskutustyyppi (tunti- tai kertalaskutus) ja status (avoin, suljettu tai peruttu). Projektilasku liittyy aina johonkin projektiin. Kun työtunti on laskutettu, luodaan yhteys työ- ja projektilaskuentieettien välille ja täten merkitään työtunti laskutetuksi. Työ-, projekti- ja projektilaskuentiteetit voivat olla asiakasorganisaatiolle ominaisia eivätkä kaikki asiakasorganisaatiot tarvitse tällaista toiminnallisuutta. Tuntikirjaus- ja seuranta on tässä mallinnettu esimerkinomaisesti yksinkertaisessa muodossaan, ja tätä mallinnusta voidaan tarvittaessa laajentaa.

4.3 Menetelmät

Suunnittelussa olen hyödyntänyt Schekkermanin [2004] neljään osaan jaoteltua arkkitehtuurikehystä, jonka tasot ovat liiketoiminta-, tieto-, tietojärjestelmä- ja teknologia-arkkitehtuuri. Suunnittelun pohjana on käytetty liiketoimintatarpeita ja niistä johdettuja tietotarpeita. Nämä on kuvattu tekstuaalisesti ja tarvittaessa graafisesti, hyödyntäen esimerkiksi prosessi- ja ER-kaavioita. Käytettyjä menetelmiä ovat siis käsite-, prosessi- ja tietomallinnus. Liiketoiminta- ja tietotarpeiden pohjalta suunnitellaan edellä mainitut tarpeet tyydyttävä tietojärjetestelmäarkkitehtuuri, jonka mukainen järjestelmä suunnitellaan tämän tutkimuksen puitteissa ja myöhemmin toteutetaan teknologia-arkkitehtuurivaiheessa valittujen teknologioita ja toteutustapoja käyttäen. Tietojärjestelmäarkkitehtuuri suunnitellaan tutustumalla erilaisiin arkkitehtuuriratkaisuihin. Käytettävät teknologiat ja toteutustavat valitaan perustuen yritys Y:n aiempiin kokemuksiin aiemmista vastaavanlaisista projekteista ja Ratas-järjestelmän erityisvaatimuksiin. Teknologia-arkkitehtuurin yhteydessä on suunniteltu myös relaatiotietokannan ja olioparadigman yhdistäminen. Järjestelmän rakenne suunnitellaan tietojärjestelmäarkkitehtuurin mukaiseksi ottaen huomioon Ratas-järjestelmän erityistarpeet ja käytettävät teknologiat. Järjestelmän rakennetta kuvataan esimerkiksi UML:n (Unified Modeling Language) [OMG, 2009] mukaisilla luokkakaaviolla sekä tekstuaalisesti.

4.4 Toteutukset

Tässä luvussa käsitellään toteutuksen suunnittelu, jota jäsennän aiemmin esitetyn organisaation kokonaisarkkitehtuuriajattelun [Schekkerman, 2004; Siltanen, 2004] avulla. Koska järjestelmää tukevat businessvaatimukset (liiketoiminta-arkkitehtuuri) ja tietotarpeet (tietoarkkitehtuuri) on jo määritelty, tässä kohdassa esitellään siis

- tietojärjestelmäarkkitehtuuri, eli tietojärjestelmien arkkitehtuuriratkaisut, joilla hallinnoidaan tietoarkkitehtuurin sisältämää tietoa businessvaatimusten mukaan; ja

- teknologia-arkkitehtuuri, eli järjestelmän toteutuksessa käytetyt teknologiset ratkaisut.

4.4.1 Tietojärjestelmäarkkitehtuuri

Tässä luvussa käsitellään erilaisia tietojärjestelmätason arkkitehtuuriratkaisuja jo aiemmin esitellyn ja tässä tutkimuksessa noudatettavan Schekkermanin [2004] kokonaisarkkitehtuuritehtuurijaottelun lisäksi ja kuvataan, mitä arkkitehtuurimalleja Ratas-järjestelmä edustaa tai sen suunnittelussa on hyödynnetty.

4.4.1.1 Asiakas-palvelin-arkkitehtuuri

Internet ja tietoverkot ovat perinteisesti suurelta osin perustuneet asiakas-palvelin-arkkitehtuuriin (engl. client-server architecture). Siinä asiakkaan ja palvelimen roolit vahvasti eriytetään. Asiakas pyytää palvelimelta jotain sen tarjoamaa palvelua, esimerkiksi www-sivua, ja palvelin vastaa pyyntöön. [Kurose & Ross, 2007].

4.4.1.2 Kolmitasoinen arkkitehtuuri

Kolmitasoinen arkkitehtuuri (engl. 3-tier architecture) on laajennus asiakas-palvelin-arkkitehtuuriin, ja asiakkaan rooli säilyy samana kuin asiakas-palvelin-arkkitehtuurissa. Tätä asiakkaan tasoa kutsutaan presentaatiotasoksi (engl. presentation tier). [Manuel & AlGhamdi, 2003]. Käyttöliittymä sijaitsee tällä tasolla [Aarsten *et al.*, 1996]. Sovelluksen businesslogiikka ja businessentiteettejä koskevat operaatiot sijaitsevat keskitasolla (engl. middle tier) [Manuel & AlGhamdi, 2003; Aarsten *et al.*, 1996]. Datataso (engl. data tier) hoitaa keskitason tarvitsemien tietojen varastonnoinnin [Manuel & AlGhamdi, 2003] yhdellä tai useammalla palvelimella [Aarsten *et al.*, 1996].

4.4.1.3 Palvelukeskeinen arkkitehtuuri

Palvelukeskeinen arkkitehtuuri (engl. service-oriented architecture eli SOA) on nouseva lähestymistapa, joka keskittyy löyhästi kytkettyyn (engl. loose coupling), standardeihin pohjautuvaan ja protokollariippumattomaan hajautettuun käsittelyyn. SOA:ssa ohjelmistoresurssit on paketoitu palveluiksi, jotka ovat tarkasti määriteltyjä standardin mukaista business-toiminnallisuutta tarjoavia moduuleita. [Papazoglou & Heuvel,

2007]. SOA on suunnittelufilosofiana riippumaton mistään tietystä teknologiasta [Papazoglou & Heuvel, 2007]; se on siis alusta- ja toimittajariippumaton. Lisäksi SOA:ssa on epäolennaista, pyörivätkö palvelut paikallisella vai etätietokoneella [Papazoglou & Heuvel, 2007]. Kaikki toiminnallisuudet SOA:ssa on määritelty palveluiksi [Arsanjani, 2002], joita tarjotaan joko loppukäyttäjien ohjelmistoille tai muille palveluille [Papazoglou & Heuvel, 2007]. Palvelu pitää yllä omaa tilatietoaan, ja palvelu on riippumaton toisten palveluiden tilasta ja kontekstista [Papazoglou & Heuvel, 2007]. Palvelun kuluttajan ei tarvitse välittää tai olla tietoisia siitä, miten palvelu on toteutettu, kunhan palvelu tarjoaa haluttua toiminnallisuutta riittävällä palvelun laadulla [Papazoglou & Heuvel, 2007].

Palvelut toteuttavat tietyn, yksiselitteisesti määritellyn rajapinnan eli tiedon siitä, millaisia viestejä palvelu vastaanottaa ja miten saatuihin viesteihin vastataan [Hashimi, 2003]. Palvelu toisaalta ilmentää rajapintaansa ja toisaalta toteutustaan; palvelulla on siis rajapinta ja toteutus. Rajapinta määrittelee palvelun identiteetin ja kutsumislogistiikan. Palvelun toteutus toteuttaa työn, joka palvelun on määrä tehdä. Toteutus ei näy palvelun ulkopuolelle ja on siten rajapinnan takana. [Papazoglou & Heuvel, 2007]. Palvelut on määritelty standardinmukaisella määrittelykielellä, niillä on julkaistu rajapinta ja ne kommunikoivat yhdessä tukeakseen liiketoiminnallista tehtävää tai prosessia [Fremantle *et al.*, 2002]. Palveluiden tarjoajat ja kuluttajat kommunikoivat viesteillä, jotka ovat tyypillisesti XML-skeeman mukaisia XML-dokumentteja. XML mahdollistaa rajapinnan yksiselitteisen määrittelyn. [Hashimi, 2003]. Toiminnot eli palvelut voidaan määritellä kuvauskielellä, kuten WDSL:llä (Web Services Description Language) [Papazoglou & Heuvel, 2007]. Palvelukeskeiseen arkkitehtuuriin kuuluu dynaaminen palveluiden löytäminen (engl. dynamic discovery) [Hashimi, 2003], joka mahdollistaa sen lisäksi palveluiden dynaamisen kutsumisen ja (uudelleen) yhdistelemisen [Papazoglou & Heuvel, 2007]. Palveluiden löytämisen mahdollistamiseksi palvelun tarjoaja rekisteröityy palvelunvälittäjälle, joka pitää kirjaa palvelua tarjoavista tarjoajista, ja palvelun tarvitsija voi kysyä välittäjältä eri kriteerein sopivaa palveluntarjoajaa [Hashimi, 2003].

4.4.1.4 Eri arkkitehtuuriratkaisujen soveltaminen Ratas-järjestelmässä

Tietojärjestelmäarkkitehtuurin suunnittelu on toteutettu Schekkermanin [2004] kokonaisarkkitehtuuriajattelun mukaisesti. Ratas edustaa kolmitasoista arkkitehtuuria, jossa loppukäyttäjä ja käyttöliittymä ovat presentaatiotasolla, businesslogiikka on keskitasolla ja tiedot varastoidaan tietotasolla. Ratas edustaa myös asiakas-palvelin-arkkitehtuuria, jossa asiakasohjelmana on loppukäyttäjän selain ja palvelimena yritys Y:n

ylläpitämä sovelluspalvelin. Toiminnot Ratas-järjestelmässä asiakkuudenhallinnan ja muut toiminnot voidaan nähdä palvelukeskeisen arkkitehtuurin mukaan järjestelmän tarjoamina palveluina.

4.4.2 Teknologia-arkkitehtuuri

Tässä luvussa käsitellään Ratas-järjestelmän toteutuksessa hyödynnettäviä paradigmoja ja niitä edustavia konkreettisia teknologisia ratkaisuja. Tavoitteena on valita ja perustella Ratas-järjestelmän toteutukseen hyödynnettävät paradigmat ja käytettävät teknologiat.

4.4.2.1 Ohjelmointiparadigma

Yritys Y:n projekteissa on perinteisesti käytetty olioparadigmaa. Olio-ohjelmointi voidaan selittää esimerkiksi seuraavasti: Oliot ovat kokoelma operaatioita, jotka jakavat tilatiedon; tilatieto ja toiminnallisuus on kapseloitu olion sisään. Olioilla on operaatioita tai kutsuja (metodeja), joihin se vastaa, ja olion ulkopuolelle näkyvät metodit muodostavat olion rajapinnan. [Wegner, 1990]. Tilatieto ja ohjelmakoodi on siis pakattu yhteen olioksi. Koodin uudelleenkäyttöä rohkaistaan olioparadigmalla paitsi tilatiedon ja toiminnallisuuden yhteenpakkaamisella, myös toiminnallisuutta perimällä. [Moser, 1991]. Luokat ovat mallineita, joista oliot luodaan, ja luokat voivat periä toiminnallisuutta yläluokiltaan [Wegner, 1990]. Olioiden välillä voi olla suhteita (engl. relationship), joilla ne liitetään toisiinsa [Moser, 1991]. Olioita, joihin isäntäoliolla on viittaus, kutsutaan isäntäolion attribuuteiksi [Wegner, 1990; Lodhi *et al.*, 2004]. Olio-ohjelmoinnissa olennainen käsite on myös monimuotoisuus eli polymorfismi [Koskimies, 2000]. Se tarkoittaa sitä, että "ohjelmassa esiintyvä nimetty kohde voi tarkoittaa useita erilaisia asioita" [Koskimies, 2000, s. 96].

Olio-ohjelmointi on käsitteenä toki paljon laajempi, mutta sen tätä yksityiskohtaisempi tarkastelu ei liene tässä yhteydessä tarpeen. Ratas-projekti toteutetaan käyttäen tätä paradigmaa, koska se on yritys Y:lle ennestään tuttu ja täten sen käytön riskit ovat helpoimmin hallittavia.

4.4.2.2 Tiedon varastoinnin paradigma

Perinteisesti yritys Y:n toteuttamissa projekteissa on käytetty tietovarastona relaatiotietokantaa, joten Ratas-projekti toteutetaan tukeutuen relaatioparadigmaan. Relaatiomallin on esitellyt Codd [1970]. Relaatiomallissa data jaetaan tauluihin, jotka liitetään yhteen niiden välisillä linkeillä [Codd, 1982] eli tarkemmin sanottuna tauluissa ole-

villa vierasavaimilla (engl. foreign key) [Lodhi *et al.*, 2004]. Tietokannan hallintajärjestelmässä (engl. database management system, DBMS) voi kyselyitä tehdä kielellä, joka tukee relaatioalgebraan perustuva relaatioprosessointia [Codd, 1982]. SQL (engl. Structure Query Language) on tällainen kieli [IBM, 2006]. Kuten olio-ohjelmoinnin tapauksessa, ei relaatiomalliakaan liene syytä tarkentaa tämän tutkimuksen yhteydessä.

4.4.2.3 Olio- ja relaatioparadigman epäyhteensopivuus

Relaatiotietokantaan perustuvan tavan varastoida tietoa eli relaatiomallin ja oliomallin välinen yhteiselo ei ole vailla ongelmia. Olio-ohjelmoinnissa luokka sisältää täydellisen tiedon tietystä abstraktiosta [Lodhi *et al.*, 2004] eli oliolla on sekä tilatieto että niihin liittyvä toiminnallisuus [Ambler, 1997]. Relaatiotietokannassa data tallennetaan riveiksi tauluihin. Oliomallissa oliot liitetään toisiin yksisuuntaisilla suhteilla [Ambler, 1997]; relaatiomallissa liitetään tauluja yhteen kaksisuuntaisilla vierasavaimilla [Lodhi *et al.*, 2004]. Tästä paradigmojen konfliktista muodostuvaa ongelmaa kutsutaan impedance mismatch -ongelmaksi [Ambler, 1997]. Sovelluskehittäjän täytyy luoda datasta kaksi mallia, oliomalli ja relaatiomalli, ja lisäksi määritellä näiden välinen kuvaus (engl. mapping) [Lodhi *et al.*, 2004]. Kuvaus voidaan toteuttaa kirjoittamalla ohjelmakoodiin SQL-kyselyitä, mutta siinä on puutteensa [Amber, 1997]. Paljon kehitysresursseja tuhlataan impedance mismatch -ongelman ratkaisuun [Lodhi *et al.*, 2004]. Jopa 20%–30% ohjelmoinnin kokonaismäärästä liittyy tämän ongelman ratkaisuun ja SQL-kyselyiden käsinkirjoittamiseen [Keller *et al.*, 1993]. Tämän kuvauksen ylläpito sovelluksissa on kallista [Lodhi *et al.*, 2004].

Lukuisia eri ratkaisuja kuvaukseen on mietitty, mutta ne eivät täysin eliminoi impedance mismatch -ongelmaa tai aiheuttavat suorituskykyongelmia. Ongelmaan on kuitenkin helpotuksia. On olemassa erilaisia tapoja tehdä olio- ja relaatiomallit toisiaan vastaaviksi. Olion attribuuteilla voi olla vastaavuus nollaan tai useampaan sarakkeeseen relaatiotietokannan taulussa. Attribuutit voivat olla toisia olioita, jolloin ne voidaan tallentaa relaatiotietokantaan joko saman taulun sarakkeisiin tai uusiin tauluihin, joihin viitataan relaatiotietokannassa vierasavaimella. [Lodhi *et al.*, 2004]. Olio voidaan tallentaa relaatiotietokantaan esimerkiksi seuraavilla kolmella tavalla:

1. käyttäen yhtä taulua koko luokkahierarkian tietojen tallentamiseen (suodatettu kuvaus, engl. filtered mapping);

2. käyttäen yhtä taulua konkreettista luokkaa kohden (horisontaalinen kuvaus, engl. horizontal mapping); tai

3. käyttäen yhtä taulua luokkaa kohden (vertikaalinen kuvaus, engl. vertical mapping) [Ambler, 1997; Lodhi *et al.*, 2004].

Ensimmäisessä vaihtoehdossa koko luokkahierarkian kaikki attribuutit on sijoitettu relaatiotietokannassa yhteen tauluun olion tyypin lisäksi [Lodhi *et al.*, 2004]. Tämän tavan haittapuolena on se, että mihin tahansa kohti luokkahierarkiaa lisätty attribuutti täytyy lisätä tauluun eli taulun kaikille riveille. Ad hoc -raportointi ja monimuotoisuuden tukeminen on kuitenkin helppoa. [Ambler, 1997]. Toisessa vaihtoehdossa kukin taulu sisältää omat attribuuttinsa ja abstraktien yläluokkiensa attribuutit [Lodhi *et al.*, 2004]. Tässä tavassa ad hoc -raportointi on edelleen helppoa. Heikkoutena on se, että attribuutin lisääminen yläluokalle johtaa tarpeeseen lisätä attribuutti kaikkien konkreettisten luokkien tauluihin. [Ambler, 1997]. Kolmannessa vaihtoehto vastaa näistä vaihtoehdoista parhaiten olioajattelua. Se tukee hyvin monimuotoisuutta ja luokkahierarkian muokkaaminen on helppoa, kun luokkaa muokattaessa täytyy muokata vain yhtä taulua. Heikkoutena tietokantaan syntyy useita tauluja, joiden tietojen muokkaaminen ja lukeminen on hidasta ja ad hoc -raportointi hankalaa. [Ambler 1997]. Eri vaihtoehtojen etuja ja haittoja on punnittu taulukossa 3.

Kriteeri	**Yksi taulu hierarkiaa kohden**	**Yksi taulu konkreettista luokkaa kohden**	**Yksi taulu luokkaa kohden**
Toteutuksen helppous	helppo	keskiverto	vaikea
Datan haun helppous	helppo	helppo	keskiverto/helppo
Uniikkien avainten asettamisen helppous	helppo	keskiverto	keskiverto
Kytkentäisyys (engl. coupling)	erittäin korkea	korkea	matala
Datan haun nopeus	nopea	nopea	keskiverto/nopea
Tuki monimuotoisuudelle	keskiverto	matala	korkea

Taulukko 3: Relaatio-olio-ristiriidan ratkaisuun kehitettyjä strategioita tiedon mallintamiseksi [Ambler, 1997].

Mikään yksittäinen esitetyistä malleista ei riitä, vaan näitä malleja tulee yhdistellä parhaaseen lopputulokseen pääsemiseksi [Ambler, 1997]. Näin on toimittu aiemmin tutkielman tietomallinnuksessa. Käytännön vaikutuksen näkee parhaiten syntyneistä ER-kaavioista. Impedance mismatch -ongelmaan on kehitetty myös monimutkaisempia ratkaisuja, jotka vaativat huomattavasti monimutkaisempaa taulurakennetta [Lodhi *et al.*, 2004]. Tällaiset ratkaisut ovat kuitenkin yritys Y:n yrityskulttuurin vastaisia, joten niiden käyttäminen on vaihtoehdoista poissuljettu eikä niitä tässä yhteydessä tarkemmin tarkastella. Lisäksi yritys Y:n ohjelmistokehittäjät hallitsevat SQL-kyselykielen kattavasti, joten sekä sovelluksessa olevien käsin tehtyjen että ad hoc -kyselyiden teko on SQL-kielellä tehokasta. Oliotietokannoista yritys Y:llä ei ole kokemusta, joten niitä ei huomioida vaihtoehtona järjestelmän kehityksessä. Yritys Y:n kokemus relaatiokantaan tietovarastona perustuvassa kehityksessä on linjassa tutkimusten kanssa: oliosovelluksen ja relaatiotietokannan välinen datamuunnos on työläs ja lisää sovelluskehityksen kustannuksia [Keller *et al.*, 1993; Lodhi *et al.*, 2004]. On olemassa valmiita työkaluja helpottamaan olio- ja relaatiokannan yhteistyötä. Näitä kutsutaan ORM-työkaluiksi (engl. object-relational mapping tools) [Cabibbo & Carosi, 2005]. Nykyiset ORM-työkalut hyödyntävät kolmea esiteltyä strategiaa olio- ja relaatiomallien yhteensovittamiseksi [ibid.]. Yritys Y ei ole ORM-työkaluja aiemmissa projekteissa hyödyntänyt. Tämän tutkimuksen puitteissa tutustaan yhteen ORM-ohjelmistoon, Doctrineen [Doctrine, 2009a], ja teknologia-alustan pohdinnan yhteydessä pohditaan, onko ORM-ohjelmiston käyttö mielekästä.

4.4.2.4 Käytetty toteutusalusta ja konkreettiset teknologiset ratkaisut

Yritys Y käyttää palvelinratkaisunaan Debian GNU/Linuxia [Debian, 2009]. Ratas suunnitellaan toimivaksi tällä alustalla. Myös testi- ja kehitysalusta on sama. Perinteisesti yritys Y:n vastaavat projektit on toteutettu PHP-ohjelmointikielellä [PHP, 2009] käyttäen MySQL-relaatiotietokantaa [MySQL, 2009]. Sivupohjat (engl. templatess) toteutetaan käyttäen yritys Y:n tarpeita vastaavaksi havaittua Smarty-sivupohjaohjelmistoa [Smarty, 2009]. Kehityksessä hyödynnetään yritys Y:n muita projekteja varten kehittelemää PHP-ohjelmistokehystä. Koska yritys Y:llä on vahvin osaaminen nimenomaan näistä teknologioista, käytetään niitä myös Ratas-projektin toteuttamiseen. Tämä myös minimoi järjestelmän toteutukseen liittyviä riskejä.

4.4.2.5 Olioperustaisen toteutuksen ja relaatiotietokannan yhteensovittaminen

Pohdittaessa hyödynnetäänkö projektissa ORM-työkalua vai ei, ja mikäli hyödynnetään, mitä ORM-työkalua, on asetettava lähtökohdaksi yritys Y:n mielestä olennaiset

kriteerit. Näitä ovat ainakin:

1. MySQL-tietokannan ja SQL-kyselykielen tuki.

2. kaikenlaisten suhteiden (1:n, 1:1, n:n) tuki;

3. ad hoc ja muiden kyselyiden käsin määrittelemisen helppous: työkalun luomien taulujen yksinkertaisuus, eli kuinka lähellä taulurakenne on ER-kaaviosta käsin luotua rakennetta.

4. ad hoc ja muiden käsin määriteltyjen kyselyiden tehokkuus eli kuinka tehokkaiksi ad hoc -kyselyt voidaan määritellä;

5. työkalun luomien kyselyiden tehokkuus sovelluksessa;

6. työkalun resurssinkulutus; ja

7. ratkaisun tulee olla riittävän yksinkertainen.

Osa edellä mainituista kriteereistä liittyy läheisesti toisiinsa eikä lista ole kattava. Lista ei myöskään määrittele tarkkoja metriikoita tai rajoja eri kriteereille. Listan tarkoitus on määritellä yleissuuntaiset periaatteet, jotka ORM-työkalun pitäisi täyttää, jotta se voitaisiin ottaa Ratas-projektin yhteydessä käyttöön. Tämän tutkimuksen, asiakkuudenhallintajärjestelmän suunnittelun, yhteydessä tutustuttiin Doctrineen [Doctrine, 2009a], joka on PHP-pohjainen ORM-työkalu, ja verrataan, miten hyvin se toimii annettujen kriteerien suhteen.

Dokumentaationsa [Doctrine, 2009a] ja testiemme mukaan Doctrine kykenee generoimaan tietomallinsa olemassa olevan MySQL-tietokannan perusteella ja luoda tarvittavat linkit taulujen välillä vierasavainrajoitteiden (engl. foreign key constrains) perusteella. Taustalla on siis tavallinen relaatiotietokanta, ja kannan rakenne voi olla samanlainen kuin käsin luodulla tietokannalla [Doctrine, 2009b]. Täten kriteerit 1, 3 ja 4 täyttyvät, sillä tietokannan rakenne ja sitä kautta kyselyt voidaan optimoida tarvittaessa käsin. Doctrine myös tukee kaikenlaisia suhteita (1:n, 1:1, n:n) (kriteeri 2) [Doctrine, 2009b]. Doctrinelle voi luoda kyselyitä omalla kyselykielellä [Doctrine, 2009b]. Tämä esimerkkitietomalli ja esimerkkikysely on määritelty Doctrinen dokumentaatiossa [Doctrine, 2009b]:

```
$query = Doctrine_Query::create()
    ->from('User u')
    ->leftJoin('u.Email e')
    ->leftJoin('u.Phonenumbers p')
    ->where('u.id = ?', 2);
```

Tämä kysely muuttuu Doctrinessa seuraavaan muotoon:

```
SELECT u.id AS u__id,
    u.username AS u__username,
    u.password AS u__password,
    e.id AS e__id,
    e.user_id AS e__user_id,
    e.address AS e__address,
    p.id AS p__id,
    p.user_id AS p__user_id,
    p.phonenumber AS p__phonenumber,
    p.primary_num AS p__primary_num
    FROM user u
    LEFT JOIN email e
        ON u.id = e.user_id
    LEFT JOIN phonenumber p
        ON u.id = p.user_id
    WHERE u.id = 2
```

Täten muodostettu kysely itsessään on riittävän tehokas (kriteeri 5). Testiemme mukaan Doctrine-työkalun lataaminen PHP-skriptiin ja ensimmäisen kyselyn tekeminen tietokantaan vie huomattavasti enemmän muistia kuin yritys Y:n sisäisen, huomattavasti yksinkertaisemman ORM-ratkaisun käyttäminen. Doctrinen ja yksinkertaisen tietomallin lataaminen PHP-skriptille, edellä mainitun SQL-kyselyn ja siitä syntyvän puumaisen tietorakenteen luominen vie noin kuusi megatavua muistia. ORM:n lataaminen ja saman kyselyn ja siitä syntyvän puurakenteen luominen yritys Y:n omalla ORM:lla vie noin 600 kilotavua muistia. Kyseessä on yhden rivin haku yksinkertaisesta taulurakenteesta (kolme taulua, kaksi LEFT JOIN -ehtoa taulujen välillä, yksi WHERE-ehto). Tämä on siis kunkin yksittäisen sivulatauksen viemä muistinkulutus. Se kertautuu kunkin sivulatauksen yhteydessä, koska ORM joudutaan lataamaan

uudestaan kutakin PHP-skriptin suoritusta kohden uudestaan. Täten kriteeri 6 ei täyty. Doctrinella haut voi kuitenkin tehdä helpommin sen omalla kyselykielellä, jolloin LEFT JOIN -hauissa ei tarvitse määritellä hakuehtoa, vaan se on kirjattu Doctrinen tietomalliin valmiiksi [Doctrine, 2009b]. Kriteerin 7 mittaamiseksi ei ole mitään metriikkaa, joten sitä ei testattu. Muistin kulutuksesta ja toimeksiantajan yrityskulttuurista johtuen Ratas-järjestelmän kehityksessä päädytään käyttämään yritys Y:n omaa ORM-työkalua ja täten työnantaja sitoutuu sen ylläpitämiseen ja kehittämiseen osana Ratasta ja muita projekteja.

4.4.2.6 Ohjelmiston rakenne

Ohjelmiston rakenne kuvataan tarpeellisilta osiltaan UML:n mukaisella luokkakaaviolla [OMG, 2009]. Koska ohjelma toteutetaan englanniksi, on taulukossa 4 määritelty vastaavuudet järjestelmän kehityksen kannalta olennaisimmille suomen- ja englanninkielisille termeille. Taulukko sisältää vastaavuudet suomenkielisten käsitteiden eli ER-kaavion entiteettien ja PHP-ohjelmistossa toteutettujen luokkien välillä. Kutakin taulukossa mainittua entiteettiä vastaa siis PHP-puolella luokka, ja vastaavasti kutakin tietokannan riviä vastaa yksi olio (poislukien viittaukset toisiin tauluihin). Entiteettien ja siten luokkien väliset suhteet on jo mallinnettu ER-kaaviolla, joten samaa tietomallia ei esitetä tässä uudelleen UML-luokkakaaviona.

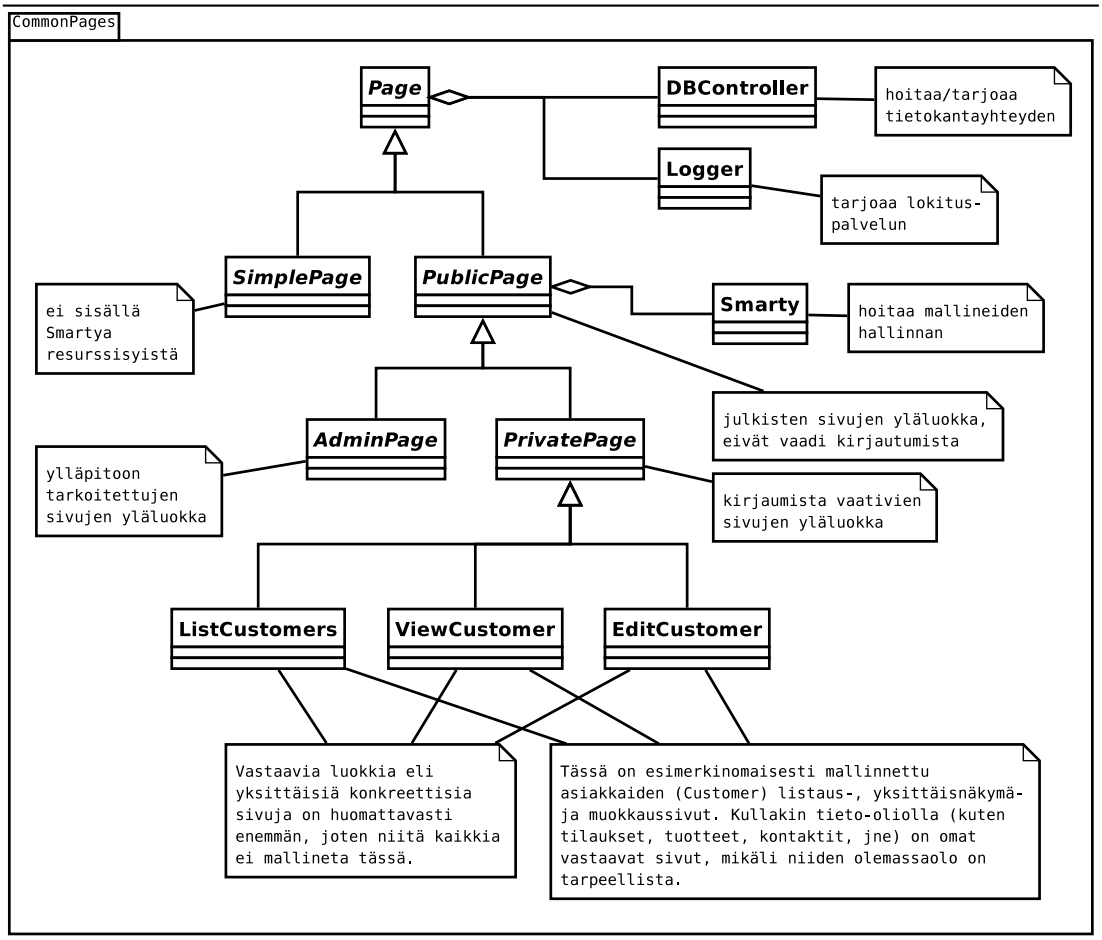
Kuvassa 6 on esitelty ohjelmiston kaikille asennuksille yhteinen sivujen luokkahierarkia. Se on sovitettu yritys Y:n muiden projektien käyttöön kehitetyn PHP-ohjelmistokehyksen mukaiseksi, mikä hieman monimutkaistaa luokkahierarkiaa. Sivu eli *Page* on loppukäyttäjälle näytettävä yksi www-sivu. Kaikki sivut perivät abstraktin Pagen. Pagella on viittaus tietokantarajapintaa ylläpitävään tietokantakontrolleriin (*DBController*) ja lokipalveluita tarjoavaan lokijärjestelmään (*Logger*). *SimplePage* on Pagen perivä yksinkertainen sivu; sitä käytetään, kun sivupohjaohjelmiston (Smarty) lataaminen on tarpeetonta. *PublicPage* on abstrakti Pagen perivä julkinen sivu, jonka nähdäkseen loppukäyttäjän ei tarvitse olla kirjautunut järjestelmään. Tällaisia sivuja ei Ratas-järjestelmässä kirjautumissivua lukuunottamatta ole. *AdminPage* on ylläpitoon tarkoitettujen sivujen abstrakti yläluokka. *PrivatePage* on kirjautumista vaativien sivujen, eli käytännössä ylivoimaisesti useimpien Ratas-järjestelmän sivujen, abstrakti yläluokka. Konkreettiset luokat ovat Ratas-järjestelmän tapauksessa PrivatePagen aliluokkia. Näitä konkreettisia sivuja ovat muun muassa eri olioiden listaus-, yksittäisnäkymä- ja muokkaussivut. Nämä kolme sivu toteutetaan oletusarvoisesti jokaista taulukossa 4 esitettyä dataoliota kohden, mikäli niiden toteuttaminen on tarpeen.

Tämä rakenne ei vielä mahdollista Ratas-asennusten asiakasorganisaatiokohtaista

Entiteetti (käsite suomeksi)	**Luokka (käsite englanniksi)**	**Lisämainintoja**
Myyntimahdollisuus	Prospect	Lyhennetty ohjelmointia ajatellen
Liidi	Lead	
Oikeussubjekti (asiakas)	Customer	Oikeussubjektit ovat asiakkaita, joko yksityis- tai yritysasiakkaita; lyhennetty
Osoite	Address	
Kontaktihenkilö	Contact	Lyhennetty
Myyjä (käyttäjä)	User	Myyjä on järjestelmässä käyttäjä; yhdenmukaistettu ohjelmointia ajatellen
Tarjous ja tilaus	Offer	Tilaus on tarjous, joka on asetettu tilaukseksi; tietosisällöltään ne ovat identtisiä ja ne varastoidaan samassa taulussa tietokannassa. Ne mallinnetaan myös PHP-puolella samaksi luokaksi. Sanaa *Order* käytetään lähdekoodissa viittaamaan nimenomaan tilaukseen.
Tuotenimike	Product	Lyhennetty
Tilauslasku	OrderBill	Luokan nimi on lyhennetty
Liitetiedosto	Attachment	
Kalenterimerkintä	CalEntry	Luokan nimi on lyhennetty sanoista Calendar Entry

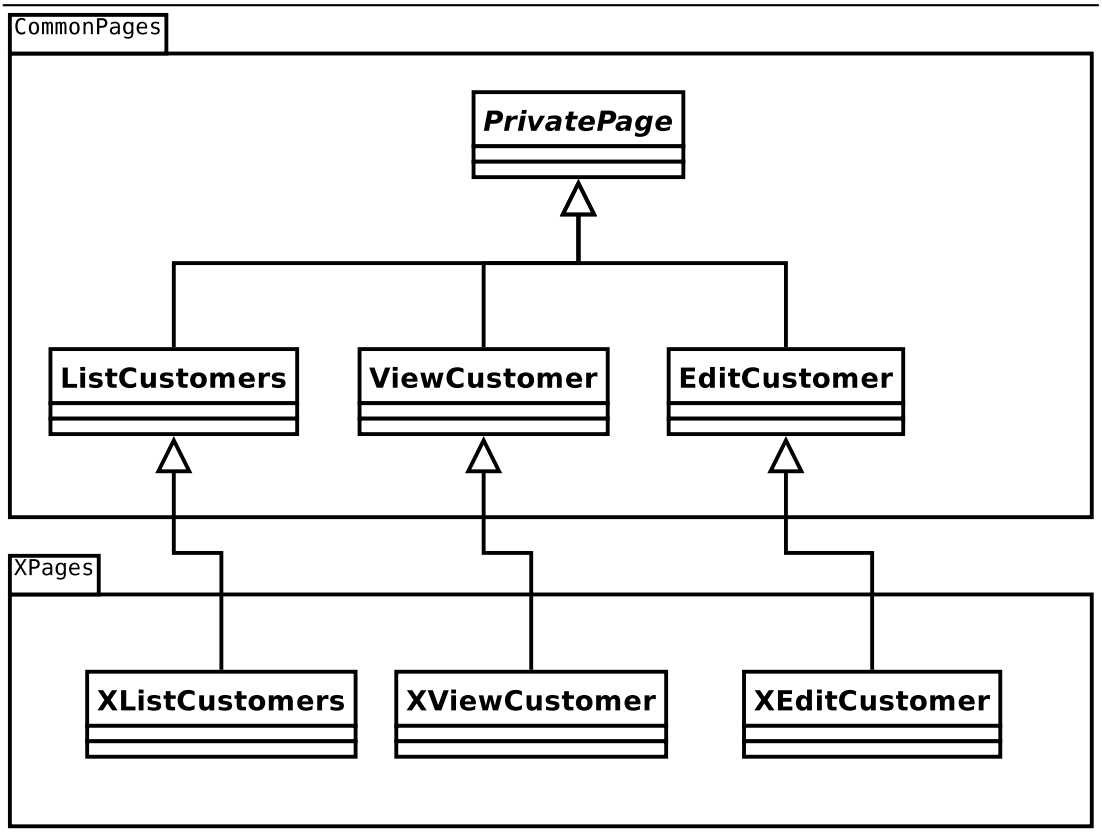
Taulukko 4: Suomen- ja englanninkielisten käsitteiden vastaavuus Ratas-järjestelmän suunnittelussa.

mukautusta, vaan sen mahdollistava rakenne on esitetty kuvassa 7. Siinä on on kuvattu osa *CommonPages*-paketista, joka on siis sama kuin kuvassa 6. Jokaisesta konkreettisesta CommonPages-paketin sivusta voi perimällä muokata asiakasorganisaatiokohtaisen sivun. Periminen joko laajentaa tai supistaa sivun toiminnallisuutta. Mikäli yksi (tai harva) asiakasorganisaatio haluaa laajennettua toiminnallisuutta, peritään kuten kuvassa 7 ja laajennetaan toiminnallisuutta luomalla yläluokkaan metodikutsu(t) metodiin tai metodeihin, jotka yläluokassa jätetään tyhjäksi (ei mitään tekeväksi). Näin meto-



Kuva 6 Kaikille Ratas-asennuksille yhteinen Page-luokkarakenne (CommonPages-paketti).

di(t) voidaan ylikirjoittaa eli toiminnallisuutta voidaan laajentaa alaluokassa. Vastaavasti tilanteessa, jossa yksi (tai harva) asiakasorganisaatio haluaa muita suppeamman toiminnallisuuden, peritään kuten kuvassa 7 ja luodaan yläluokkaan metodikutsu(t) metodiin tai metodeihin, joiden määrittely jätetään alaluokassa tyhjäksi, mutta jotka yläluokassa toteutetaan. Tällöin perimisellä supistetaan toiminnallisuutta.



Kuva 7 Page-hierarkian mukautettavuus eri Ratas-installaatioiden välillä.

5 YHTEENVETO

Tässä tutkimuksessa on suunniteltu Ratas-asiakkuudenhallintajärjestelmä. Järjestelmän suunnittelun teoreettisena pohjana on käytetty Marchin ja Smithin [1995] tietojärjestelmätieteen tutkimuskehystä ja Schekkermanin [2004] neljään eri tasoon (liiketoimintataso, tietotaso, tietojärjestelmätaso, teknologiataso) perustuvaa kokonaisarkkitehtuurimallia. Marchin ja Smithin tutkimuskehyksen rakentamisaktiviteetin eri vaiheet – käsitteiden, mallien, menetelmien ja toteutusten luonti – on käyty läpi tässä järjestyksessä, ja kuhunkin seuraavaan vaiheeseen on edetty edellisen jälkeen. Käsitteitä tässä tutkimuksessa ovat järjestelmän sovellusalueen käsitteet; malleja ovat käsitteiden pohjalta luodut prosessi- ja tietomallit; menetelmiä ovat suunnitelma siitä, miten aikaisempien vaiheiden mukainen järjestelmä toteutetaan; ja toteutuksia ovat toteutuksen suunnittelu eli Ratas-järjestelmän kehityksessä sovellettavan tietojärjestelmä- ja teknologia-arkkitehtuurin sekä ohjelmiston rakenteen määrittely. Kokonaisuudessaan suunnittelutieteelliseen vaihettaiseen lähestymistapaan perustuva jaottelu tarjosi tälle tutkimukselle mielekkään rakenteen ja kunkin vaiheen tarkka erottelu helpotti työn vaiheistamista ja aikatauluttamista.

Ratas-järjestelmä on suunniteltu käyttäen käsite-, prosessi- ja tietomallinnusta. Tässä asiakkuudenhallintajärjestelmän suunnittelussa käsitteiden määrittelyt perustuvat yrityksen Y näkemyksiin sovellusalueella käytetystä kielestä. Määritellyt käsitteet ja niiden suhteita kuvaavat prosessi- ja tietomallit ovat käyneet läpi iteratiiviset prosessin, jossa käsitteitä ja malleja on yhä tarkentuvasti määritelty ja arvioitu yrityksen Y liiketoimintatarpeita ja järjestelmän jatkosuunnittelua ajatellen yrityksen Y edustajien kanssa. Myös menetelmät on arvioitu yhteistyössä yrityksen Y edustajien kanssa. Yritys Y hyväksyi tutkimuksen pohjana käytetyt käsitteet, mallit ja menetelmät pohjaksi tutkimuksen jatkolle. Tarkasti ja yksityiskohtaisesti määritellyt käsitteet olivat tärkeitä kommunikoinnin onnistumiselle yrityksen Y kanssa – ikään kuin varmistukseksi siitä, että osapuolet puhuvat samaa kieltä ja samoista asioista – ja kommunikoinnin onnistuminen oli edellytys koko tutkimuksen onnistumiselle. Mallien luominen eli mallintaminen oli arvokasta niin ikään samasta syystä, mutta myös ajattelun selkiyttämiseksi: mallien luominen on paitsi kommunikaatiota, myös ajattelua ja ajattelun konkretisointia. Esiteltyjen käsitteiden ja mallien käytössä tutkimuksen puitteissa ei esiintynyt ongelmia ja ne osoittautuivat hyödyllisiksi kommunikaation välineiksi yrityksen Y edustajien kanssa.

Ratas-järjestelmän toteutus on suunniteltu, muttei toteutettu. Toteutuksen suunnittelu on perustettu yrityksen Y ohjelmointiparadigmaan ja käytössä oleviin teknologiavaihtoehtoihin. Suunnittelu on tässä tutkimuksessa viety niin pitkälle kuin se on

mielekästä ilman ohjelmoinnin aloittamista. Luoduilla käsitteillä, malleilla, menetelmillä ja toteutuksilla on luotu pohja ohjelmiston toteutukselle, eikä sille ole nähtävissä mitään esteitä. Olioparadigman ja relaatiotietokannan yhteensovittamisessa päädyttiin resurssi- ja suorituskykysyistä oman ORM-ohjelmiston jatkokehitykseen valmiin kolmannen osapuolen ORM-ohjelmiston käytön sijasta. Tämä voi aiheuttaa lisätyötä yritykselle Y, koska ulkopuolisen kirjaston hyödyntäminen oman tuotekehitysresurssien käyttämisen sijasta olisi todennäköisesti vaihtoehtoaan kustannustehokkaampaa. Muutoin tehdyt teknologiavalinnat heijastavat yrityksen Y nykyisiä vahvuuksia.

Tämän tutkimuksen arvioinnissa on hyödynnetty Hevnerin ja muiden [2004] suunnittelutieteellisen tutkimuksen seitsemää lähtökohtaa tai ohjetta. Arvioinnissa on pohdittu, onko tämä tutkimus, sen suunnittelu ja toteutus yleisesti suunnittelutieteellisen tutkimuksen lähestymistavan mukaista. Ohjeiden mukaan toimittiin ja toimimisessa onnistuttiin seuraavasti:

1. *Ohje 1 (luo artefakti)*: tutkimuksen aikana on luotu ne tuotokset, jotka tutkimuksen alussa määriteltiin tutkimuksen tavoitteiksi, eli käsitteet, mallit, menetelmät ja toteutukset.

2. *Ohje 2 (ongelman relevanssi)*: yritys Y:n liiketoiminta-ajattelun näkökulmasta tutkimuksen ongelman katsottiin olevan relevantti.

3. *Ohje 3 (suunnittelun arviointi)*: Toteutuksen voi olettaa vastaavan asetettuja tarpeita, koska eri vaiheiden tuotokset on todettu kattaviksi ja yrityksen Y tarpeita vastaaviksi. Yksi merkittävä vajavuus eri vaiheiden arvioinnissa on metriikoiden puute, koska mitään mielekkäitä metriikoita ei ole mietitty tai asetettu. Toteutuksen ominaisuuksia, kuten soveltuvuutta suunniteltuun käyttöönsä ja sovelluksen tehokkuutta, voidaan arvioida vasta Ratas-järjestelmän toteuttamisen jälkeen.

4. *Ohje 4 (tutkimuksen kontribuutiot)*: Tässä tutkimuksessa luotiin tutkimuksen alussa määritellyt tuotokset, ja tässä tutkimuksessa kuvattiin tutkimuksen kulku. Nämä ovat tutkimuksen tärkeimmät kontribuutiot.

5. *Ohje 5 (tutkimuksen perusteellisuus)*: Tämän tutkimuksen puitteissa tutustuttiin sovellusalueen aikaisempaan tutkimukseen ja käytäntöihin ja yrityksen Y erityistarpeisiin suunniteltavan asiakkuudenhallintajärjestelmän osalta.

6. *Ohje 6 (tutkimus etsintäprosessina)*: Tässä tutkimuksessa luotiin prosessi- ja tietomallit, joiden avulla etsittiin ratkaisut räätälöitävän asiakkuudenhallintajärjestelmän toteutukselle.

7. *Ohje 7 (tutkimuksen kommunikointi)*: Tämä tutkimus on esitetty käsitteinä, malleina, menetelminä ja toteutuksina, ja tutkimus on pyritty kuvaamaan siten, että se tarjoaa riittävät tiedot sekä teknologia- että johtamisorientoituneille osapuolille.

Tämän tutkimuksen puitteissa laadittiin suunnitelma Ratas-asiakkuudenhallintajärjestelmän toteuttamiseksi. Järjestelmän ominaisuuksia ja siten osin tämän tutkimuksen onnistumista voidaan arvioida vasta, kun järjestelmä on toteutettu. Tutkimuksen jatkoa on mielekästä miettiä Ratas-järjestelmän valmiin toteutuksen ja/tai käyttöönoton jälkeen.

VIITELUETTELO

[Aarsten *et al.*, 1996] Amund Aarsten, Davide Brugali, & Giuseppe Menga. Patterns for three-tier client/server applications. In *Proceedings of Pattern Languages of Programs*, volume 96, 1996.

[Ambler, 1997] Scott W. Ambler. Mapping objects to relational databases. `http://jeffsutherland.com/objwld98/mappingobjects.pdf`, 1997.

[Apicella *et al.*, 1999] Mario Apicella, Karen Mitchell, & Sommer Dugan. Customer relationship management: ramping up sales service. *InfoWorld*, 21(33):68–80, 1999.

[Arsanjani, 2002] Ali Arsanjani. Introduction to the special issue on developing and integrating enterprise components and services. *Commun. ACM*, 45(10):30–34, 2002.

[Baysan *et al.*, 2005] Can Baysan, Arielle Bertman, Raul Maynigo, Griff Norville, Nicholas Osborne, & Timothy Taylor. The design and development of a sales force automation tool using business process management software. In *Systems and Information Engineering Design Symposium*, 318–327, 2005.

[Bennett *et al.*, 2001] Keith Bennett, Malcolm Munro, Nicolas Gold, Paul Layzell, David Budgen, & Pearl Brereton. An architectural model for service-based software with ultra rapid evolution. In *Software Maintenance, IEEE International Conference on*, 292–300, Los Alamitos, CA, 2001. IEEE Computer Society.

[Borenstein & Freed, 1993] Nathaniel Borenstein & Ned Freed. *MIME (Multipurpose Internet Mail Extensions) part one: mechanisms for specifying and describing the format of Internet message bodies*, 1993.

[Cabibbo & Carosi, 2005] Luca Cabibbo & Antonio Carosi. Managing inheritance hierarchies in object/relational mapping tools. In *Advanced Information Systems Engineering, 17th International Conference on*, 135–150, 2005.

[Campbell-Kelly, 2009] Martin Campbell-Kelly. Historical reflections the rise, fall, and resurrection of software as a service. *Commun. ACM*, 52(5):28–30, 2009.

[Chen & Popovich, 2003] Injazz J. Chen & Karen Popovich. Understanding customer relationship management (CRM): people, process and technology. *Business Process Management Journal*, 9(5):672–688, 2003.

[Chen, 1977] Peter P.S. Chen. The entity-relationship model: a basis for the enterprise view of data. In *Proceedings of National Computer Conference*, 77–84. ACM New York, NY, 1977.

[Christopher *et al.*, 2002] M. Christopher, A. Payne, & D. Ballantyne. *Relationship Marketing: Creating Shareholder Value*. Butterworth-Heinemann, 2002.

[Codd, 1970] Edgar F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13:377–387, 1970.

[Codd, 1982] Edgar F. Codd. Relational database: a practical foundation for productivity. *Commun. ACM*, 25(2):109–117, 1982.

[Creighton, 2000] S. Creighton. Partnering for success to the e-business world. In *Proceedings of DCI Customer Relationship Management Conference, Boston, MA*, 2000.

[Debian, 2009] Debian. Debian GNU/Linux. `http://www.debian.org`, 2009. Checked: 7.10.2009.

[Denning, 1997] Peter J. Denning. A new social contract for research. *Commun. ACM*, 40(2):132–134, 1997.

[Doctrine, 2009a] Doctrine. Doctrine – PHP Object Relational Mapper. `http://www.doctrine-project.org`, 2009. Checked: 7.10.2009.

[Doctrine, 2009b] Doctrine. Guide to Doctrine for PHP. `http://www.doctrine-project.org/documentation/manual/1_1/en/pdf`, 2009. Checked: 7.10.2009.

[Eckerson & Watson, 2001] W. Eckerson & H. Watson. Harnessing customer information for strategic advantage: technical challenges and business solutions. *Industry Study*, 2001.

[Esteves & Pastor, 2001] José Esteves & Joan Pastor. Enterprise resource planning systems research: an annotated bibliography. *Commun. Assoc. Inf. Syst.*, 7(1):8, 2001.

[Fickel, 1999] Louise Fickel. Know your customer. *CIO Magazine*, 12(21):62–72, 1999.

[Fremantle *et al.*, 2002] Paul Fremantle, Sanjiva Weerawarana, & Rania Khalaf. Enterprise services. *Commun. ACM*, 45(10):77–82, 2002.

[Hackney, 2000] D. Hackney. Business intelligence technology and tools for CRM. In *Proceedings of DCI Customer Relationship Management Conference, Boston, MA*, 2000.

[Hashimi, 2003] Sayed Hashimi. Service-Oriented Architecture Explained. `http://ondotnet.com/pub/a/dotnet/2003/08/18/soa_explained.html`, 2003. Checked: 7.10.2009.

[Hevner *et al.*, 2004] Alan R. Hevner, Salvatore T. March, J. Park, & Sudha Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–106, 2004.

[IBM, 2006] IBM. DB2 Version 9 for Linux, UNIX, and Windows. `http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.admin.doc/doc/c0004100.htm`, 2006. Checked: 7.10.2009.

[Kassanoff, 2000] B. Kassanoff. Build loyalty into your e-business. In *Proceedings of DCI Customer Relationship Management Conference, Boston, MA*, 2000.

[Keller *et al.*, 1993] Arthur M. Keller, Richard Jensen, & Shailesh Agarwal. Persistence software: bridging object-oriented programming and relational databases. *ACM SIGMOD Record*, 22(2):523–528, 1993.

[Klaus *et al.*, 2000] Helmut Klaus, Michael Rosemann, & Guy G. Gable. What is ERP? *Inform. Syst. Front.*, 2(2):141–162, 2000.

[Koskimies, 2000] Kai Koskimies. *Oliokirja*, 2. painos. Talentum Oyj, 2000.

[Kumar & van Hillegersberg, 2000] Kuldeep Kumar & Jos van Hillegersberg. Enterprise resource planning: introduction. *Commun. ACM*, 43(4):22–26, 2000.

[Kurose & Ross, 2007] James F. Kurose & Keith W. Ross. *Computer Networking: A Top-Down Approach*, 4th ed. Addison Wesley, March 2007.

[Lodhi & Ghazali, 2004] Fakhar Lodhi & Muhammad A. Ghazali. Improving maintainability by unifying the object and relational models. In *Multitopic Conference, 2004. Proceedings of INMIC 2004. 8th International*, 635–640, Dec. 2004.

[Manuel & AlGhamdi, 2003] Paul D. Manuel & Jarallah AlGhamdi. A data-centric design for n-tier architecture. *Information Sciences*, 150(3-4):195–206, 2003.

[March & Smith, 1995] Salvatore T. March & Gerald F. Smith. Design and natural science research on information technology. *Decis. Support Syst.*, 15(4):251–266, 1995.

[Markus *et al.*, 2002] M. Lynne Markus, Ann Majchrzak, & Les Gasser. A design theory for systems that support emergent knowledge processes. *MIS Quarterly*, 179–212, 2002.

[Moser *et al.*, 1991] Kathleen A. Moser, Daniel J. Mazzola, Robert T. Keim, & Andrew S. Philippakis. Modeling the information systems architecture: an object-oriented approach. In *Proceedings of the 24th Annual Hawaii International Conference on System Sciences*, volume iv, 83–92, 1991.

[MySQL, 2009] MySQL. MySQL: The world's most popular open source database. `http://www.mysql.com`, 2009. Checked: 7.10.2009.

[Nolan, 1973] Richard L. Nolan. Managing the computer resource: a stage hypothesis. *Commun. ACM*, 16(7):399–405, 1973.

[OMG, 2009] OMG. Object Management Group – UML. `http://www.uml.org`, 2009. Checked: 7.10.2009.

[Papazoglou & Heuvel, 2007] Mike P. Papazoglou & Willem-Jan Heuvel. Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal*, 16(3):389–415, 2007.

[Patton, 2001] Susannah Patton. The truth about CRM. *CIO Magazine*, 14(9), 2001.

[Peppers & Rogers, 2002] D. Peppers & M. Rogers. *The one to one manager: real-world lessons in customer relationship management*. Currency, 2002.

[PHP, 2009] PHP. PHP: Hypertext Preprocessor. http://www.php.net, 2009. Checked: 7.10.2009.

[Schekkerman, 2004] Jaap Schekkerman. *How to Survive in the Jungle of Enterprise Architecture Framework: Creating or Choosing an Enterprise Architecture Framework*. Trafford Publishing, 2004.

[Schweigert, 2000] D. Schweigert. Balancing idealistic vs realistic processes. In *Proceedings of DCI Customer Relationship Management Conference, Boston, MA*, 2000.

[Siltanen, 2004] Juha Siltanen. Tietoarkkitehtuurin perustuva sovellusintegraatiometodi – tapaus Tieliikelaitos. Pro gradu -tutkielma, Tietojenkäsittelytieteiden laitos, Tampereen yliopisto, 2004.

[Simon, 1996] Herbert A. Simon. *The Sciences of the Artificial*, 3rd ed. MIT Press, Cambridge, Mass., 1996.

[Smarty, 2009] Smarty. Smarty: Template engine. http://www.smarty.net, 2009. Checked: 7.10.2009.

[Tsichritzis, 1997] Dennis Tsichritzis. The dynamics of innovation. In *Beyond Calculation: The Next Fifty Years of Computing*, 259–265. Copernicus, New York, NY, 1997.

[Vasconcelos *et al.*, 2004] André Vasconcelos, Carla M. Pereira, Pedro Sousa, & José Tribolet. Open issues on information system architecture research domain: the vision. In *Proceedings of the 6th International Conference on Enterprise Information System*, 273–282, 2004.

[Wegner, 1990] Peter Wegner. Concepts and paradigms of object-oriented programming. *SIGPLAN OOPS Mess.*, 1(1):7–87, 1990.

[Weill, 2007] Peter Weill. Innovating with Information Systems: What do the most agile firms in the world do? http://www.iese.edu/en/files/6_29338.pdf, 2007. Checked: 7.10.2009.