Vaatimusmäärittelymenetelmät komponenttituotannon tukena – havaintoja soveltamisesta

Irmeli Minkkinen
Pro gradu -tutkielma
Kuopion yliopisto
Tietojenkäsittelytieteen laitos
Elokuu 2004

TIIVISTELMÄ

KUOPION YLIOPISTO, Informaatioteknologian ja kauppatieteiden tiedekunta
Tietojenkäsittelytieteen koulutusohjelma
Ohjelmistotekniikka

MINKKINEN, T. IRMELI: Vaatimusmäärittelymenetelmät komponenttituotannon tukena – havaintoja soveltamisesta

Pro gradu -tutkielma, 113 s., 2 liitettä (5 s.)

Ohjaajat: FT Anne Eerola
FM Marika Toivanen
Elokuu 2004

Avainsanat: Vaatimusmäärittely, vaatimusmäärittelymenetelmä, toiminnan teoria, tapahtumalähtöinen, liiketoimintalähtöinen, komponentti

Vaatimusmäärittelyprosessin tehtävänä on saavuttaa riittävä ymmärrys kohdealueesta ja tuottaa selkeät vaatimukset ohjelmistosuunnittelun perustaksi. Vaatimusmäärittely tehdään erilaisia menetelmiä käyttäen. Menetelmätuntemus on tarpeen, sillä menetelmistä on kulloinkin valittava tilanteeseen ja kohdealueeseen parhaiten sopiva. Komponenttipohjaiset ohjelmistot yleistyvät. Kuitenkaan vaatimusmäärittelymenetelmät eivät tue suoraan komponenttituotantoa. Komponenttien ominaisuuksien ymmärtäminen on tärkeää, jotta vaatimusmäärittelymenetelmiä osataan kehittää komponenttituotantoa palvelevaan suuntaan. Hyvin tehty vaatimusmäärittely on onnistuneen ohjelmiston perusta.

Tutkielmassa perehdytään komponentteihin ja niiden ominaisuuksiin sekä vaatimusmäärittelyprosessin vaiheisiin ja ongelmiin. Kokeellisessa osassa esitellään ja tarkastellaan lähemmin toiminnan teoriasta lähtevää, tapahtumalähtöistä ja liiketoimintalähtöistä vaatimusmäärittelymenetelmää. Menetelmien käyttökelpoisuutta tutkitaan kotihoitoon kohdistuvien esimerkkien avulla. Esimerkit havainnollistavat menetelmien käyttämistä ja nostavat esille menetelmien ansioita ja puutteita. Soveltamisesta saatujen kokemusten perusteella menetelmiä arvioidaan, esitetään ratkaisuja havaittuihin ongelmiin sekä osoitetaan kehityskohteita ja jatkotutkimusaiheita.

Tutkielma liittyy valtakunnalliseen Tekes-rahoitteiseen PlugIT-hankkeeseen. Hankkeessa tutkittiin ja tuotettiin terveydenhuollon palvelutoimintaa tukevia avoimia rajapintoja ja kehitettiin ohjelmistotuotannon menetelmiä. Hankkeeseen osallistuivat Kuopion yliopisto, Savonia-ammattikorkeakoulu, eri sairaanhoitopiirit sekä joukko terveydenhuollon ohjelmistoja tuottavia yrityksiä.

Esipuhe

Tämä tutkielma on tehty Kuopion yliopiston tietojenkäsittelytieteen laitokselle kevään ja kesän 2004 aikana PlugIT-Teho -tutkimushankkeessa. Ohjaajina toimivat FT Anne Eerola ja FM Marika Toivanen, joiden kanssa käydyt keskustelut auttoivat muovaamaan tutkielmasta kokonaisuuden. Lämmin kiitos teille, Anne ja Marika! Tutkielman valmistumista edisti myös hyvä ja kannustava työilmapiiri. Kiitos työtovereille, erityisesti Kotihoidon tiedon tarpeet -projektin jäsenille.

Kuopiossa, 31.8.2004

_____

Irmeli Minkkinen

SISÄLLYS

1 Johdanto

Tiedon ja tiedonkäsittelyn määrä ja merkitys on nyky-yhteiskunnassa lisääntynyt räjähdysmäisesti. Kaikilla elämän aloilla tarvitaan tietojärjestelmiä. *Tietojärjestelmällä* tarkoitetaan ihmisistä, tietojenkäsittelylaitteista, tiedonsiirtolaitteista ja ohjelmistoista koostuvaa järjestelmää, jonka tarkoitus on tietojen käsittelyn avulla tehostaa tai helpottaa jotakin toimintaa tai tehdä toiminta mahdolliseksi [Tie01]. Tietojärjestelmien käyttö on arkipäivää yhä useammille ihmisille, ja tietojärjestelmiä käytetään apuna yhteiskunnan tärkeissä peruspalveluissa, kuten sähkönjakelussa, liikenteenohjauksessa tai pankkipalveluissa. Tietojärjestelmien suunnittelussa ja tuotannossa onkin entistä tärkeämpää kiinnittää huomiota ohjelmistojen luotettavuuteen, käytettävyyteen, toimivuuteen ja joustavuuteen.

Komponenttien käytöllä pyritään ohjelmistotuotannon tehokkuuteen, ohjelman osien uudelleenkäytettävyyteen ja ohjelmiston rakenteen selkeyteen. Komponenttituotannon ongelmina ovat kuitenkin muun muassa komponenttien ”löytämisen” vaikeus sekä huono jäljitettävyys liiketoiminnan tavoitteisiin. Komponenttien keskinäisen yhteistoiminnallisuuden ja komponenttisysteemien ja perinnejärjestelmien välisen yhteistoiminnan varmistaminen on vaikeaa. Näihin asioihin on kiinnitettävä huomiota jo vaatimusmäärittelyvaiheen aikana. Vaatimusmäärittely on ohjelmistotuotannon tärkeä, mutta käytännön työssä liian vähälle huomiolle jäänyt osa-alue. Suurimmat kustannussäästöt tehdään jo vaatimusmäärittelyvaiheessa: virheen korjaus ohjelmistotuotantoprosessin alussa tulee halvemmaksi kuin koodausvaiheessa ja paljon halvemmaksi kuin jo markkinoille lasketusta tuotteesta. Käytettävät vaatimusmäärittely- ja suunnittelumenetelmät ovat kuitenkin peräisin ajalta ennen komponentteja, eivätkä siten ota riittävästi huomioon komponenttiajattelua. Ei ole itsestään selvää, miten komponentit voidaan huomioida vaatimusmäärittelyn eri vaiheissa.

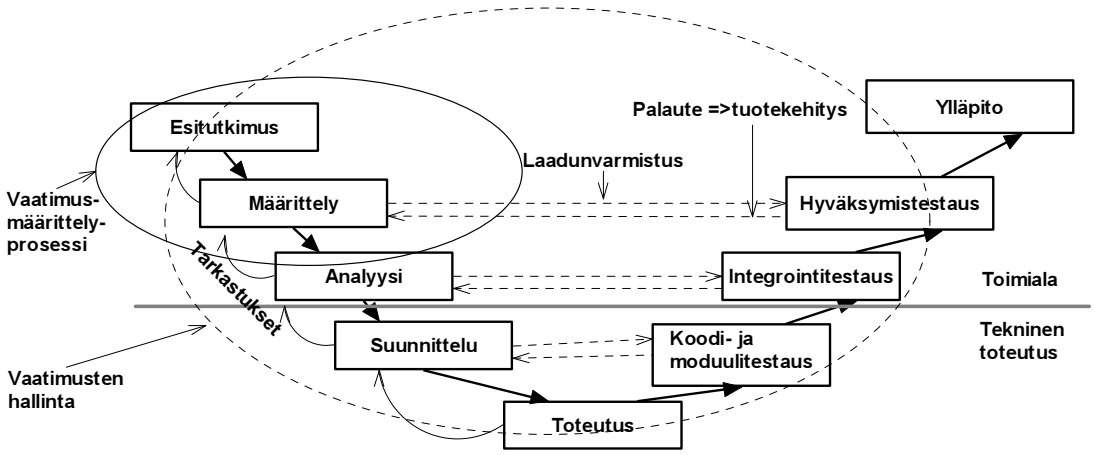
Tässä tutkimuksessa esitellään aluksi vaatimusmäärittelyyn liittyviä käsitteitä ja vaatimusmäärittelyprosessin vaiheita. Sen jälkeen esitellään komponentteihin liittyviä käsitteitä ja komponentteihin liittyviä vaatimuksia. Vaatimusmäärittely voidaan tehdä erilaisia menetelmiä käyttäen. Tutkimuksessa esitellään lyhyesti toiminnan teoriasta lähtevä, tapahtumalähtöinen ja liiketoimintalähtöinen menetelmä kukin omassa luvus-

saan. Menetelmiä tutkitaan esimerkkitapausten avulla sekä pohditaan esille nousseita menetelmien kehityskohteita, hyviä ja huonoja puolia. Esimerkkien kohdealueena on kotihoito, jonka piirteitä esitellään omassa luvussaan. Lopuksi arvioidaan menetelmiä teoriasisällön ja soveltamisesta saatujen kokemusten valossa sekä esitetään parannusehdotuksia ja jatkotutkimuksen aiheita. Työn tarkoituksena on esittää erilaisia malleja siitä, miten vaatimuksia voidaan kerätä ja analysoida. Lisäksi pohditaan, millaisista ajatusmalleista on apua, kun yritetään löytää polku asiakasvaatimuksista kohti komponentteja.

Tutkielma liittyy läheisesti PlugIT-hankkeen Teho-osaprojektiin, ja erityisesti sen kotihoidon tiedon tarpeita tutkivaan Kotihoito-projektiin. PlugIT on valtakunnallinen Tekesin rahoittama tutkimus- ja kehittämishanke. PlugIT-hankkeen tavoitteena on tukea terveydenhuollon palvelutoimintaa ohjelmistotuotannon palveluketjun kautta, paremmin integroituvien ohjelmistokokonaisuuksien avulla. PlugIT tuottaa avoimia ohjelmistorajapintojen määrityksiä sekä niihin liittyviä menetelmiä ja osaamista terveydenhuollon ohjelmistoyrityksille ja niiden asiakkaille [www01]. PlugIT-Teho toimii Kuopion yliopiston Tietojenkäsittelytieteen laitoksella. PlugIT-Tehon erikoisalueena on yhteistyöyritysten toimintatapojen ja yhteisten pelisääntöjen ja ratkaisujen kehittäminen ohjelmistojen vaatimusmäärittelyyn, arkkitehtuuri- ja komponenttikeskeiseen ohjelmistotuotantoon sekä ohjelmistojen testaukseen ja laadunvarmistukseen. Tavoitteena on yritysten tuotteiden parempi hallittavuus, integroituvuus sekä kokonaisuuksien yhteistoiminnallisuus ja integraatiotestaus. [www02]. PlugIT-hankkeeseen osallistuu Kuopion yliopisto, sairaanhoitopiirit sekä joukko terveydenhuollon ohjelmistoja tuottavia yrityksiä.

2 Vaatimusmäärittely

Ohjelmiston *elinkaarella* tarkoitetaan aikaa, joka kuluu ohjelmiston suunnittelun aloittamisesta siihen hetkeen, kun ohjelmisto poistetaan käytöstä. Ohjelmiston kehitystyö tai koko elinkaari voidaan jakaa vaiheisiin, jolloin puhutaan *vaihejakomalleista* [HaM02 s. 36]. Vaihejakomalleja on useita ja jokaisesta on olemassa eri muunnelmia. Ohjelmistotuotantoprosessin vaiheita ovat vaatimusmäärittely, analyysi, arkkitehtuurisuunnittelu, suunnittelu, toteutus, testaus sekä käyttöönotto ja ylläpito. Eri vaihejakomalleissa näitä vaiheita käydään läpi hieman eri tavoin. Vesiputousmallissa vaiheet käydään läpi peräkkäin [Roy70]. V-malli ottaa jokaisessa vaiheessa tarkastuksen ja testauksen tarkemmin huomioon (kuva 1). Spiraalimallissa korostuu riskienhallinta [Boe88]. Protoilumallissa toistetaan vesiputousmallin vaiheita nopeasti ja tuotetaan prototyyppejä, joita kehittämällä tuotetaan vaatimusten mukainen järjestelmä. Inkrementaalinen kehittäminen tuottaa ohjelmistosta toinen toistaan kehittyneempiä versioita inkrementtien eli lisäysten avulla [Som01].

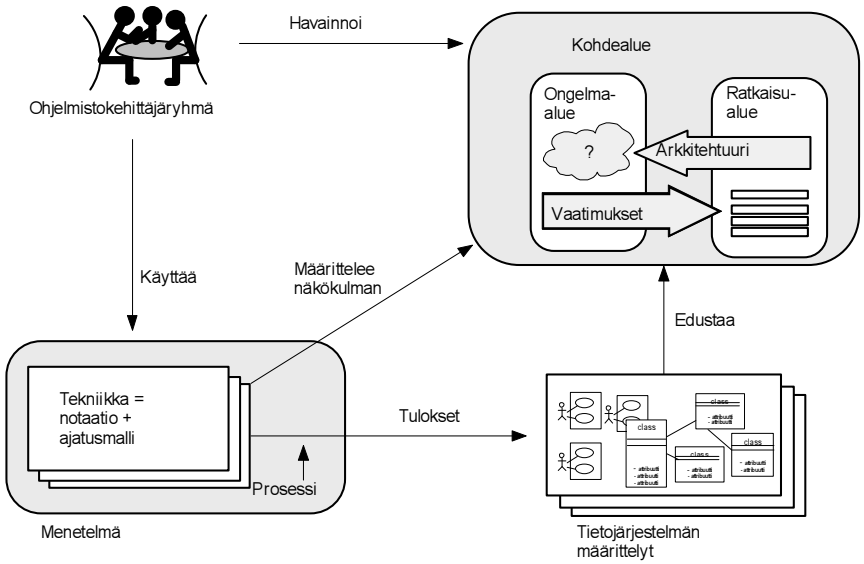


Kuva 1: Esimerkki ohjelmistotuotantoprosessista, V-malli

2.1 Vaatimusmäärittelyn lähtökohdat ja menetelmät

Ohjelmistotuotantoprosessin *kohdealue* (domain) on se reaalimaailman osa, johon ohjelmisto suunnitellaan. Kohdealueelta voidaan erottaa *ongelma-alue*, joka asettaa vaatimuksia suunniteltavalle ohjelmistolle, sekä *ratkaisualue,* joka vastaa ongelma-

alueen vaatimuksiin (kuva 2). Ratkaisualue kuvataan tietojärjestelmän määrittelydokumenteissa. Kohdealuetta voidaan lähestyä eri näkökulmista ja eri tavoin. Tietoa kohdealueesta voidaan jäsentää ja kuvata eri menetelmin. Lähestymistapoja (approach) ja menetelmiä (method) on useita. Tässä työssä *lähestymistavalla* tarkoitetaan sitä näkökulmaa tai lähtökohtaa, josta kohdealuetta tarkastellaan. *Näkökulma* (viewpoint) voi olla esimerkiksi asiakkaan, tietyn käyttäjäryhmän tai ohjelmistotuottajan näkökulma kohdealueeseen. *Lähtökohdalla* (starting point) tarkoitetaan sitä asiaa, käsitettä tai kokonaisuutta, mistä kohdealueen tarkastelu aloitetaan sekä sitä teoriaa tai mallia, mihin kohdealueen tarkastelu perustuu. Lähestymistapa on väljempi käsite kuin menetelmä: *menetelmä* on vaiheittain etenevä prosessi, jossa on määritelty askelten suoritusjärjestys ja käytettävä notaatio.



Kuva 2: Ohjelmistotuotantoprosessin käsitteitä [Smo03]

Kohdealueen mallintamisen ja vaatimusmäärittelyn lähtökohtana voi olla esimerkiksi

- sidosryhmäanalyysi, jonka avulla voidaan tunnistaa ne henkilöt ja ryhmät, joilta vaatimuksia saadaan kerättyä,
- liiketoiminnan tavoitteet ja ydinprosessit (liiketoimintalähtöinen, business driven, ks. luku 7) [Ars02, LeA02],

- liiketoiminnan mallintaminen komponenttiajattelun mukaisesti käyttäen perusosina liiketoimintaprosesseja, -entiteettejä, -tapahtumia ja -sääntöjä [HeS00],
- Asiakkaan työ ja sen yhteydet muuhun maailmaan (tapahtumalähtöinen, event driven, ks. luku 6) [RoR99] ja
- kohdealueen työtoiminta ja toimintaprosessit (toiminnan teoriasta lähtevä, Activity Analysis and Development, ActAD, ks. luku 5) [KoS00].

Tässä tutkielmassa esitellään liiketoimintalähtöinen, tapahtumalähtöinen ja toiminnan teoriasta lähtevä vaatimusmäärittelymenetelmä ja sovelletaan niitä esimerkkitapauksiin. Esimerkkien kohdealueena on kotihoito. Nämä menetelmät kohdistuvat vaatimusmäärittely- ja ohjelmistotuotantoprosessin eri vaiheisiin. Activity Analysis and Development (ActAD) mallintaa kohdealueen työtoimintaa toiminnanteorian avulla ja sijoittuu vaatimusten keräämis- ja analysointivaiheeseen ja on esiteltävistä menetelmistä ainoa, jota tässä tutkimuksessa on käytetty vaatimusten keräämisvaiheessa. Tapahtumalähtöinen menetelmä keskittyy kohdealueen mallintamiseen, eikä esimerkissä ole otettu kantaa siihen, miten vaatimukset kerätään. Myös esiteltävä liiketoiminnan tavoitteista ja prosesseista lähtevä menetelmä alkaa kohdealueen mallintamisesta, mutta siinä edetään pidemmälle komponenttisuunnitteluun.

Lisäksi tietoa voidaan kerätä erilaisilla käyttäjälähtöisillä menetelmillä. Näitä ovat muun muassa käyttäjäkeskeinen suunnittelu (user centered design) ja osallistuva suunnittelu (participatory design). Käyttäjäkeskeistä suunnittelua käytetään varsinkin vuorovaikutteisten ohjelmistojen, erityisesti internet-sivustojen suunnittelussa [www03]. Käyttäjäkeskeisiin menetelmiin kuuluvat muun muassa havainnointi ja etnografiset menetelmät, joissa suunnittelija havainnoi toimialan työntekijää tämän omassa työympäristössä, tai osallistuu itsekin työhön [Haw01]. Osallistuvassa suunnittelussa toimialan työntekijät ovat aktiivisesti mukana tietojärjestelmän suunnittelussa Osallistuva suunnittelu on osa käyttäjäkeskeistä suunnittelua, mutta sitä voidaan käyttää osana muissakin menetelmissä.

Kohdealueen mallintamisen tavoitteena on ymmärtää kohdealueen toiminta ja työ. Tämän jälkeen on taitoa selvittää, miten suunniteltava systeemi helpottaa tuota työtä. Mikään edellä luetelluista mallintamismenetelmistä ei kuitenkaan suoraan esitä koko

polkua kohdealueen kartoittamisesta komponenteiksi. Siksi on tarpeellista tutkia, miten eri lähtökohdista alkaen voidaan päätyä toimivaan komponenttisovellukseen ja millaisista ajatusmalleista on apua komponenttiprojektin vaatimusmäärittelyn eri vaiheissa.

2.2 Vaatimukset

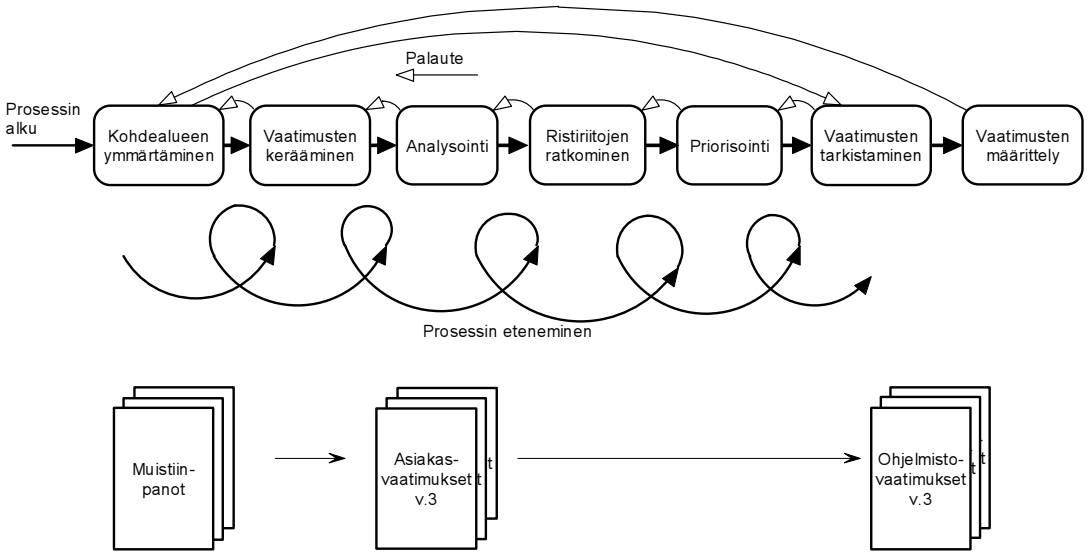
Vaatimusmäärittely (requirements engineering) on yksi ohjelmistotuotantoprosessin osa-alue ja se sijoittuu ohjelmistotuotantoprosessin alkupuolelle (kuva 1). *Vaatimusmäärittelyssä* kuvataan ja analysoidaan sekä kohdealue, että suunniteltava systeemi. Vaatimusmäärittelyn tarkoituksena on tuottaa selkeä kuvaus tarvittavasta järjestelmästä. Vaatimukset voidaan jakaa toiminnallisiin ja ei-toiminnallisiin vaatimuksiin. *Toiminnalliset vaatimukset* kuvaavat sitä, mitä toimintoja tuotettavan järjestelmän tulee toteuttaa. *Ei-toiminnalliset vaatimukset* liittyvät järjestelmän laadullisiin ominaisuuksiin, kuten käytettävyyteen, suorituskykyyn ja ylläpidettävyyteen.

On otettava huomioon myös *toimialan vaatimukset*, jotka kuvaavat kohdealueelle tyypillisiä vaatimuksia, kuten esimerkiksi desimaalien pyöristyssäännöt tai terveydenhuollon alalla se, ettei vanhaa diagnoosia saa poistaa. *Systeemivaatimukset* ovat suunniteltavalle systeemille asetettuja vaatimuksia, jotka voivat johtua esimerkiksi asiakkaan perinnejärjestelmästä, jota halutaan vielä hyödyntää. *Rajoitteet* eivät ole varsinaisia vaatimuksia, mutta ne vaikuttavat olennaisesti suunniteltavaan tuotteeseen. Rajoitteisiin kirjataan esimerkiksi vaatimus valmiiden komponenttien uudelleenkäytöstä. *Projektivaatimukset* ovat tuotantoprojektin asettamia vaatimuksia, kuten esimerkiksi projektin kustannusarvio ja käytettävissä oleva aika [RoR99]. Kun ohjelmistoyritys haluaa käyttää ohjelmiston rakentamisessa komponentteja, on otettava huomioon myös *komponenttiteknologian ja -markkinoiden aiheuttamat vaatimukset.*

2.3 Vaatimusmäärittelyprosessi

Vaatimusmäärittelyprosessin aikana selvitetään asiakkaan toiveet ja tarpeet (asiakasvaatimukset) ja niitä tarkentamalla määritellään tuotettavan ohjelmiston toiminnot ja ominaisuudet (ohjelmistovaatimukset), jotka toteuttavat asiakkaan tarpeet. Vaatimus-

määrittelyprosessi ei etene suoraviivaisesti askel askeleelta yhteen suuntaan, vaan tiedon lisääntyessä palataan tarvittaessa edelliseen askeleeseen. Prosessi on syklinen ja iteratiivinen (kuva 3). Prosessin aikana vaatimusten keräämisvaiheen muistiinpanot ”jalostetaan” ohjelmistovaatimuksiksi.



Kuva 3: Vaatimusmäärittely on iteratiivinen prosessi. Mukailtu [Som01 s.125]

Vaatimusmäärittelyprosessin vaiheita ovat toteutettavuustutkimus, vaatimusten kerääminen (synonyymi: vaatimusten kartoitus) (requirements elicitation), vaatimusten analysointi (requirements analysis), määrittely (specification), vaatimusten validointi (validation) ja vaatimusten hallinta [Som01 s.145]. Brayn mukaan käyttäjärajapinnan (human machine interface, HMI) suunnittelu on edellisten lisäksi yksi vaatimusmäärittelyn osatehtävistä [Bra02 s.24].

Vaatimusten kerääminen, analysointi ja dokumentointi kietoutuvat yhteen. Niitä tehdään jossakin määrin lomittain sekä iteratiivisesti. Eli kerätään tietoa, analysoidaan, huomataan lisää tiedon tarpeita, täsmennetään, tarkennetaan ja priorisoidaan vaatimuksia (kuva 3). Vaatimukset tulee dokumentoida huolellisesti ja riittävän yksiselitteisesti. Tuotetut dokumentit toimivat syötteenä ohjelmistotuotantoprosessin seuraavalle vaiheelle, eli suunnittelulle. *Vaatimusten validoinnilla* varmistetaan se, että vaatimukset ovat oikeellisia, ristiriidattomia ja realistisia. Vaatimustenhallinta (requirements management) erotetaan omaksi prosessikseen, joka jatkuu läpi ohjelmistotuo-

tantoprosessin (kuva 1). *Vaatimusten hallinnan* tehtävä on varmistaa, että lopputuote vastaa asiakkaan vaatimuksia [HaM02 s.93 -94]. Vaatimustenhallintaa tarvitaan, koska vaatimukset muuttuvat ohjelmistotuotantoprosessin aikana, ja on tärkeää tietää mikä on muuttunut, miten ja miksi.

2.3.1 Vaatimusten kerääminen

Vaatimusmäärittelyn alkuvaiheella (elicitation) on monta eri nimitystä suomeksi. Puhutaan vaatimusten keräämisestä, kartoittamisesta, etsimisestä tai jopa kalastelusta. Vaatimusten 'kerääminen' on huono sana, sillä ohjelmistovaatimukset eivät ole valmiina esillä keräämistä odottelemassa, vaan niitä on etsittävä ja kalasteltava, ikään kuin houkuteltava esiin erilaisin keinoin [Bra02 s.41]. Sanan elicitation suomennokseen 'saada selville, saada ongituksi tietoonsa jotakin' liittyy ajatus siitä, että tiedon saamiseksi tulee jollakin tavalla ponnistella.

Kun aloitetaan vaatimusten keräämisvaihetta, on päätettävä, *mitä tietoa kerätään*, *mistä lähteistä* tietoa saadaan, *millä menetelmillä ja keinoilla* tieto voidaan saada. Vaatimusten kartoittamista voi edeltää esitutkimusvaihe, jolloin perehdytään ongelma-aluetta koskevaan kirjallisuuteen, muihin julkaisuihin tai ongelma-alueelta aiemmin tehtyihin tutkimuksiin.

Kerättävä tieto voidaan jaotella esimerkiksi seuraavasti: kohdealueen (toimialan) kuvaamiseen tarvittavat tiedot, lista niistä ongelmista, jotka halutaan ratkaista sekä asiakkaan antamat rajoitukset [Bra02 s.42]. Kun mietitään, mitä asioita kohdealueesta on saatava selville, voidaan tiedontarvetta jäsennellä esimerkiksi liiketoiminnan prosessien, yrityksen organisaatiorakenteen tai kohdealueen toiminnan mukaan. Alkuvaiheessa ei yleensä tiedetä, mitä tietoa tarvitaan, ellei kohdealue ole entuudestaan tuttu. Tiedon tarpeet tarkentuvat prosessin edetessä.

Myös tarvittavan tiedon laatu on erilainen eri tilanteissa. Esimerkiksi, kun suunnitellaan olemassa olevan järjestelmän uudistusta, joudutaan perehtymään vanhaan järjestelmään ja päättämään, missä määrin vanha systeemi on käyttökelpoinen, mitkä osat uusitaan ja miten, mitä uutta toiminnallisuutta tarvitaan ja miten uudet osat integroidaan vanhaan. Suunniteltaessa järjestelmää alueelle, jolla ei ole aikaisempaa järjes-

telmää, voidaan keskittyä kohdealueen tutkimiseen ilman vanhan järjestelmän asettamia rajoitteita.

Tietolähteet voivat olla ihmisiä: asiakas, loppukäyttäjä, ongelma-alueen asiantuntija, tai ei-inhimillisiä: olemassa olevat järjestelmät, kilpailijoiden tuotteet ja niiden dokumentaatiot sekä olemassa olevat standardit ja lainsäädäntö. Kun tietolähteenä on ihmisiä, tietoa voidaan kerätä esimerkiksi erilaisten haastattelujen, kyselyiden, työpajojen, aivoriihien, havainnoinnin tai osallistuvan suunnittelun avulla. Tietolähteiden valinta tehdään tilanteen ja tietolähteen tärkeyden mukaan.

Vaatimusmäärittelyn ”ymmärtämisongelma” on se, ettei ohjelmistosuunnittelija välttämättä ymmärrä asiakkaan työtä ja työssä käyttämää työslangia oikein. Toisaalta asiakas ei välttämättä tiedä riittävästi tietotekniikan mahdollisuuksista, eikä siten osaa ilmaista sitä, mitä todella tarvitaan ongelman ratkaisemiseksi tietotekniikan keinoin. asiakkaalla, mutta myös ICT-ammattilaisilla (information and communication technology, ICT), on työstään niin sanottua *hiljaista tietoa* (tacit knowledge), joka on olennaista tai luonteenomaista toimialalle, mutta jota on vaikea ilmaista. Asiakas tai ohjelmistosuunnittelija saattaa pitää joitakin tietoja niin itsestään selvinä, ettei tule ajatelleeksi, että nekin pitää kertoa.

Vaatimusten kartoittamisvaiheen tuloksena on kartoitusmuistiinpanot (elicitation notes) ja luonnos asiakasvaatimuksista. Näitä analysoidaan ja tarkennetaan seuraavassa, eli vaatimusten analysointivaiheessa.

2.3.2 Vaatimusten analysointi

Analysointivaiheessa tarkoituksena on saavuttaa ymmärrys kohdealueesta ja ongelmasta, joka vaatii ratkaisua. Edellisessä vaiheessa kerätty tieto on vielä hajanaista (eirakenteista), ja se voi sisältää tietojärjestelmän suunnittelun kannalta tarpeetontakin tietoa. Kunkin asiakasvaatimuksen kohdalla etsitään perimmäinen syy, miksi kyseinen vaatimus on asetettu ja miten tärkeää vaatimus on toteuttaa. Keskenään ristiriitaiset vaatimukset sovitetaan yhteen [HaM02 s.96].

Vaatimus on *ristiriidassa* jonkin toisen vaatimuksen kanssa, jos yhden vaatimuksen toteuttaminen estää tai rajoittaa toisen vaatimuksen täyttämisen tai tuottaa virheellisen tuloksen. Esimerkiksi terveydenhuollon järjestelmällä tulee olla korkeatasoinen tietoturva ja toisaalta järjestelmän on oltava helppokäyttöinen. Tällöin on päätettävä, kumpi vaatimus on tärkeämpi toteuttaa. Ristiriidat vaatimuksissa tulee dokumentoida, jotta ne muistetaan käsitellä. Kun ristiriidat ratkaistaan, on myös tehdyt ratkaisut perusteluineen dokumentoitava.

Analysoitaessa tietojärjestelmän suunnittelun kannalta olennainen tieto erotellaan epäolennaisesta. Kohdealueen elementit kuvataan, ryhmitellään ja elementtien väliset suhteet esitetään. Kuvattavia asioita ovat kohdealueen luonteenomaiset ominaisuudet, kohdealueen organisaation tai toiminnan rakenne, työssä tarvittavat tietokokonaisuudet, kohdealueeseen vaikuttavat tapahtumat ja miten tuotettavan systeemin tulisi vaikuttaa kohdealueeseen [Bra02 s.53].

Analysoinnissa voidaan käyttää eri menetelmiä, esimerkiksi rakenteinen analyysi (Structured Analysis, SA), oliosuuntautuneet (Object Oriented) menetelmät, esimerkiksi OOA [CoY90], OMT [RuB91], OMT++ [Jaa97], SOOA [Put94] ja OOSE [JaC92], kohdealuesuuntautunut analyysi (Problem Domain Oriented Analysis, PDOA), komponenttisuuntautunut liiketoiminnan mallinnus (Component-Based Business Modelling, CBBM [HeS00]) ja toiminnan teoriaan pohjautuva analysointi (Activity Analysis and Developnment, ActAD). ActAD:ia lukuun ottamatta menetelmät ovat pelkästään ohjelmistotuotantoon tarkoitettuja. *ActAD* on alunperin työtoiminnan analysointiin ja kehittämiseen suuntautunut menetelmä, jota voidaan soveltaa työtoiminnan ja tietojärjestelmän yhtäaikaiseen kehittämiseen. Sitä esitellään tarkemmin kappaleessa 5.2. Kappaleessa 5.3 on esimerkki ActADin soveltamisesta.

Vanhin edellä luetelluista menetelmistä on 1970-luvun lopulla Tom DeMarcon esittämä *rakenteinen analyysi*, jota on edelleen kehittänyt esimerkiksi Edvard Yourdon [You89]. Rakenteisessa analyysissa systeemin toiminnot (system functions) kuvataan *tietovirtakaavioilla* (dataflow diagram), jotka koostuvat prosesseista, tietovarastoista ja tietovirroista. Käsitteet kuvataan *ER-kaavioilla* (Entity-Relationship Diagram). Systeemin ajasta riippuva toiminta kuvataan *tilasiirtymäkaavioilla* (State-Transition Diagram). Ohjelman rakenne kuvataan *rakennekartalla* (Structure Chart). *Tietohakemis-*

tossa kuvataan systeemin tiedot tarkasti käyttäen tarkoin määriteltyä syntaksia. Environmental Model sisältää kuvauksen systeemin käyttötarkoituksesta. Systeemin rajat ja suhteet ympäristöön kuvataan *kontekstikaavion* ja *tapahtumalistan* avulla. Näiden avulla systeemistä luodaan *toiminnallinen malli* [You89].

Oliosuuntautuneet menetelmät yleistyivät oliokielien myötä 1980- ja 1990-luvuilla. Oliosuuntautunut analyysi etsii kohdealueelta oliot ja luokat ja määrittelee niille tarvittavat attribuutit ja metodit. Apuna tässä käytetään usein UML:n (Unified Modelling Language) erilaisia kaavioita: käyttötapauskaavio (use case diagram [JaC92]), aktiviteettikaavio, luokkakaavio, tilakaavio ja niin edelleen [RuJ99, Fow97].

Kohdesuuntautuneen analyysin (PDOA) esitti Michael Jackson 1990-luvulla. PDOA:ssa erotellaan kohdealueen kuvaus ja suunniteltavan systeemin toimintaan liittyvät vaatimukset selkeästi omiksi dokumenteikseen. Analysointivaiheessa kohdealueen relevantit elementit tunnistetaan ja kuvataan Tunnistettavia elementtejä ovat entiteetit, tapahtumat, arvot, tilat, totuudet ja roolit [Jac99]. Analysointivaiheessa listataan myös ratkaisua kaipaavat ongelmat. Määrittelyvaiheessa kuvataan, miten suunniteltavan systeemin tulee toimia, jotta edellisessä dokumentissa luetellut ongelmat saadaan ratkaistua. Kohdealueen kuvaamisessa käytetään apuna *ongelmakehyksiä* (problem frames). Kohdealueelta tunnistetaan erityyppisiä ongelma-alueita (problem domains):

- työkaluohjelma (workpiece), joka käsittelee reaalisia kohteita, esimerkiksi sana-prosessori,
- ohjaus (control system), joka säätelee kohdealueen toimintaa, esimerkiksi hissinohjaussysteemi,
- tiedonkäsittely (information system), joka käsittelee kohdealueeseen kohdistuvia tietotarpeita, esimerkiksi opintorekisteri,
- kääntäjä (transformation system), joka huolehtii tietovirran muodon muuntamisesta, esimerkiksi systeemi, joka muuntaa transaktioita pankkikuiteiksi ja
- yhdistäjä (connection system), joka huolehtii kahden erillään olevan osa-alueen yhdistämisestä, esimerkiksi videoneuvottelu.

Kun ongelma-alueiden tyypit on tunnistettu, käsitellään kutakin aluetta juuri sille hyväksi havaitulla menetelmällä [Bra02 s. 92 -127]. Tämä on hyödyllistä, sillä eri ongelma-alueiden komponenttisysteemit tarvitsevat eri tyyppisiä komponentteja toteutukseen. Esimerkiksi hissin ohjausjärjestelmän kokoonpanossa tarvitaan erilaisia pe-

rusosia kuin opintorekisterissä. Suunnittelutyöhön kuluu vähemmän aikaa, kun kunkin ongelma-alueen järjestelmiin tarvittavat perusosat on valmiiksi määritelty.

Komponenttisuuntautuneessa liiketoiminnan mallintamisessa (CBBM) lähtökohtana ovat liiketoiminnan prosessit. Kohdealue liitetään suunniteltavan ohjelmiston toiminnallisuuteen käyttämällä liiketoiminnan mallintamisen elementteinä liiketoiminnan prosesseja, entiteettejä, tapahtumia ja sääntöjä. Näistä voidaan edetä edelleen prosessi-, entiteetti- ja varuskomponenttien löytämiseen ja määrittelyyn [HeS00 s.431].

Reaalimaailmassa toiminnat muuttuvat nopeammin kuin käsitteet. Kun ohjelmiston perustana ovat käsitteet, saadaan vakaa (stabiili) järjestelmä. Rakenteisessa analyysissä perustana ovat toiminnat, jolloin järjestelmään joudutaan todennäköisesti tekemään usein muutoksia, ja järjestelmästä tulee epävakaa. Rakenteisen analyysin huonona puolena on myös se, että kohdealueen ja olemassa olevien systeemien kuvaamiseen ei kiinnitetä huomiota, vaan aletaan mallintaa suoraan suunniteltavaa systeemiä. Kun vaatimukset kuvataan teknisin termein, analysoinnin ja suunnittelun raja on häilyvä. Tällöin systeemin tekninen ja sisäinen suunnittelu aloitetaan epäkypsistä vaatimuksista liian varhain. Myös kunnollinen toiminnallinen määrittely puuttuu [Bra02].

Sekä SA- että OO-menetelmät mallintavat suunniteltavaa systeemiä, eivätkä niinkään kohdealuetta tai ratkaistavaa ongelmaa. Kohdealueesta on oltava jo tässä vaiheessa selvä kuva ja asiakasvaatimusten tulee olla kerättynä. PDOA ja ActAD ovat uudempia menetelmiä, jotka suuntautuvat kohdealueen elementtien kuvaamiseen.

Analyysivaiheen tuloksena on suunniteltavan systeemin toiminnallinen määrittely, jossa kuvataan ohjelmistovaatimukset. Dokumentoinnista kerrotaan seuraavassa luvussa.

2.3.3 Vaatimusten dokumentointi

Jotta tuotettujen vaatimusmäärittelydokumenttien pohjalta voidaan suunnitella toimiva, asiakkaan vaatimukset oikein täyttävä ja asiakkaan työtä parhaiten tukeva sovellus, on dokumenttien laadintaan kiinnitettävä huomiota. Vaatimusmäärittelyvaiheen eri dokumenteilla on eri tarkoitus ja eri lukijakunta. Siksi niiden on oltava tyyliltään

erilaisia. Esimerkiksi asiakas ja ohjelmistoammattilainen tarvitsevat eri sisältöiset dokumentit.

Vaatimusmäärittely on iteratiivinen, kierros kierrokselta tarkentuva prosessi, jonka aikana vaatimusten keräämisvaiheen muistiinpanot jalostuvat ja täydentyvät ensin asiakasvaatimuksiksi ja edelleen ohjelmistovaatimuksiksi. *Asiakasvaatimuksissa* kuvataan kohdealueen nyky- ja tavoitetila sekä suunniteltavan systeemin toiminta. *Ohjelmistovaatimuksissa* kuvataan suunniteltavan systeemin toiminnalliset ja ei-toiminnalliset vaatimukset tarkalla tasolla sekä projektin asettamat ja muut mahdolliset rajoitteet.

Asiakasvaatimukset kuvataan usein käyttäen tavallista kieltä. Tämä on ongelmallista, koska puhuttu kieli ei ole riittävän yksiselitteistä, jotta sen perusteella voidaan suunnitella oikein toimiva ohjelmisto. Asiakasvaatimukset voidaan kuvata myös käyttötapausten (use case) avulla. *Käyttötapauksissa* kuvataan järjestelmän toiminnallisuus joukkona järjestelmän käyttäjän suorittamia tapahtumaketjuja [JaC92]. Käyttötapausten lisäksi myös kohdealue on kuvattava riittävän hyvin, muuten käyttötapauskuvaukset jäävät irrallisiksi ja epävakaiksi [www04].

Asiakasvaatimukset on muunnettava ohjelmistovaatimuksiksi, jotka kuvataan formaalimmin kuin asiakasvaatimukset, käyttäen esimerkiksi UML-notaatiota apuna. Vaatimukset *priorisoidaan*, eli päätetään, mitkä vaatimuksista ovat ensiarvoisen tärkeitä toteuttaa ja mitkä voidaan toteuttaa joko myöhemmin tai seuraavassa ohjelmaversiossa.

Jokainen vaatimus yksilöidään omalla tunnisteella. Kaikki yksittäiset vaatimukset kuvataan sanallisesti mahdollisimman yksiselitteisesti. Jokaiselle vaatimukselle kirjataan prioriteetti (esimerkiksi kriittinen, tärkeä tai toivottava) sekä vaatimuksen kohde, eli mihin järjestelmän osaan vaatimus kohdistuu. Lisäksi voidaan tarvittaessa kirjata

- vaatimuksen asettaja ja lähde, josta vaatimus on saatu,
- perustelut sille, miksi vaatimus on kirjattu,
- täyttymiskriteerit, jotka on toteutettava, jotta vaatimus katsotaan täyttyneeksi,
- tukimateriaali, esimerkiksi standardit,
- vaatimuksen historia, eli milloin vaatimus on kirjattu, onko sitä muutettu, ja jos on, miten ja miksi sekä

- vaatimuksen tyyppi: onko kyseessä asiakas-, ohjelmisto- vai sovellusalueen vaatimus.

Vaatimusmäärittelyvaiheessa tuotettuja dokumentteja käytetään vaatimusten *validointiin*. Vaatimukset käydään läpi asiakkaan kanssa, jotta varmistutaan, että on ymmärretty ongelma oikein ja ollaan suunnittelemassa ratkaisua oikeaan ongelmaan. Tällöin on hyvin tärkeää, että asiakkaalle ja ohjelmistoammattilaiselle tulee yhteinen käsitys suunniteltavasta systeemistä. Asiakkaan ei voida olettaa ymmärtävän ohjelmistotekniikan ammattikieltä ja teknisiä termejä. Tätä varten tarvitaan yleisellä tasolla kirjoitettua dokumenttia ja kohdealueen kuvausta. Kohdealueesta kuvataan nykytila ja tavoitetila.

Validoinnin lisäksi vaatimusmäärittelydokumentteja tarvitaan *suunnitteluvaiheen ja testauksen pohjana*. Vaatimusten pohjalta suunnitellaan ja toteutetaan vaatimukset täyttävä ohjelmisto. On myös testattava, että vaatimukset toteutuvat ja ohjelmisto toimii vaatimuksissa määritellyllä tavalla. Tällöin on tärkeää, että vaatimukset on kuvattu riittävän yksiselitteisesti ja formaalisti. Tätä tarkoitusta palvelevat systeemin ohjelmistovaatimukset.

Vaatimusmäärittely on *sopimus* asiakkaan ja ohjelmistotuottajan välillä siitä, millainen toteutettavan ohjelmiston tulee olla. Systeemin valmistuttua vaatimusmäärittelydokumentista voidaan tarkistaa, että toteutetusta systeemistä löytyy kaikki vaaditut toiminnot ja ominaisuudet (*verifiointi*).

Vaatimusmäärittelyprosessin yhtenä ongelmana on dokumentoinnin työläys: aina ei ole aikaa tai resursseja kirjoittaa kaikkea tarvittavaa huolella. Huolimattomuus vaatimusmäärittelyvaiheessa kuitenkin kostautuu myöhemmin ohjelmistotuotantoprosessin seuraavissa vaiheissa. Dokumentoinnin apuna voidaan käyttää erilaisia malleja, menetelmiä ja työvälineitä. Esimerkiksi Volere Requirenments Process antaa dokumentointimallin niin kokonaisuuden kuin yksittäisen vaatimuksenkin kannalta [RoR99 s.137 - 164]. Volere-mallia on esitelty tarkemmin kappaleessa 4.3.

2.3.4 Vaatimusmuutostenhallinta ja vaatimusten jäljitettävyys

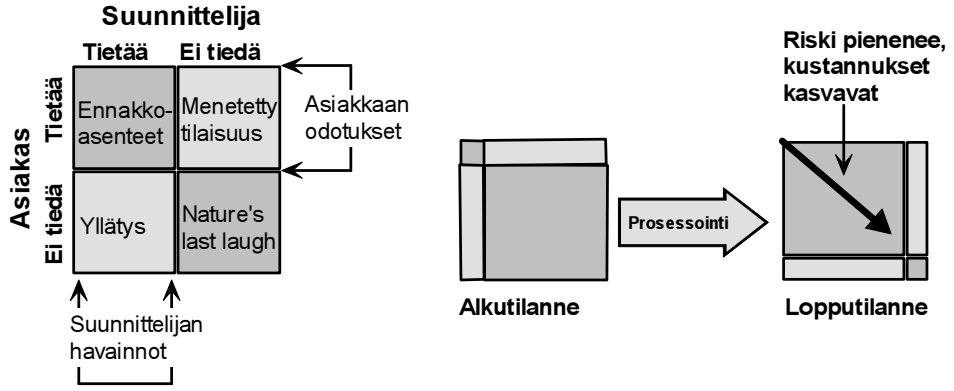
Asiakasvaatimusten kartoittamisen ja analysoinnin lisäksi vaatimustenhallinnan osa-alueita ovat vaatimusten jäljitettävyys ja vaatimusmuutosten hallinta. Vaatimusten *jäljitettävyydellä* tarkoitetaan sitä, että pystytään kustakin tietojärjestelmän ominaisuudesta osoittamaan, minkä asiakasvaatimuksen se toteuttaa. *Eteenpäin jäljitettävyydellä* tarkoitetaan sitä, että pystytään asiakasvaatimuksesta päättelemään, mitkä ohjelmistovaatimuksissa kuvatuista toiminnoista toteuttavat kyseessä olevan asiakasvaatimuksen ja edelleen, mitkä osat ohjelmakoodista toteuttavat kyseessä olevan kohdan teknisessä määrittelyssä. *Taaksepäin jäljitettävyydellä* tarkoitetaan sitä, että ohjelmakoodi voidaan vaiheittain jäljittää asiakasvaatimuksiin. Jäljitettävyys on yksi ohjelmiston hyvän laadun kriteeri [Myö02]. Kun vaatimuksia muutetaan, jäljitettävyyden avulla nähdään, mihin kaikkiin muihin vaatimuksiin tehty muutos vaikuttaa ja asiakkaat, joilla piirre on käytössä[HaM02 s.93 - 101]. Muuttunut vaatimus voi olla esimerkiksi ristiriidassa jonkin muun vaatimuksen kanssa.

Vaatimusmuutostenhallinnalla on keskeinen osa vaatimustenhallinnassa. On selkeästi sovittava muutoskäytännöt ja -menettelyt, esimerkiksi muutosten hyväksymiskriteerit, sekä muutosten dokumentointi. Varsinkin isoissa ohjelmistokehitysprojekteisssa on tyypillistä, että vaatimukset muuttuvat prosessin aikana. Syinä muutoksiin voi olla esimerkiksi se, että jokin vaatimus on ymmärretty väärin tai jäänyt alkuvaiheessa huomaamatta. Myös asiakkaan liiketoiminnassa voi tapahtua muutoksia, jotka vaikuttavat tietojärjestelmälle asetettaviin odotuksiin ja tarpeisiin [HaM02].

2.4 Vaatimusmäärittelyn ongelmat

Vaatimusmäärittely on tärkeä vaihe sekä asiakkaan että ohjelmistotuottajan kannalta. Hyvä vaatimusmäärittely on edellytys sille, että saadaan tuotettua oikein toimiva ohjelmistotuote, joka parhaalla mahdollisella tavalla tukee asiakkaan työtä ja toimintaa. Vaatimusmäärittely ei saa olla pelkkä suunniteltavan systeemin kuvaus, vaan siihen kuuluu olennaisena osana kohdealueen todellisen maailman kuvaus [Bra02 s.32]. Teoria ja käytäntö eivät kohtaa vaatimusmäärittelyn saralla. Vaikka hyviä vaatimusmäärittelymenetelmiä on olemassa, kaikki ohjelmistokehittäjät eivät niitä kuitenkaan käytä [Sub99].

Olennaista on, että ohjelmistosuunnittelijalla ja asiakkaalla on yhteinen käsitys suunniteltavasta tuotteesta ja suunnitteluun vaikuttavista asioista. *Suunnitteluikkuna* (kuva 4) kuvaa asiakkaan ja suunnittelijan tietoisuutta suunnittelussa merkityksellisistä asioista ja erilaisen tietoisuuden aiheuttamista suunnitteluriskeistä. Aina vaatimusmäärittelyprosessin alussa sekä asiakkaalla että suunnittelijalla on vain vähän tietoa tarvittavasta tuotteesta ja vain osa tästä tiedosta on yhteistä. Prosessin edetessä yhteinen tietoisuus lisääntyy ja suunnitteluriskit pienenevät (kuva 4) [GaL99]. Vaatimukset voidaan analysoida, spesifioida, validoida ja verifioida vain, jos sekä ohjelmiston kehittäjät että asiakas ymmärtävät vaatimukset samoin, eli vaatimukset on esitetty riittävän yksiselitteisesti [Sub99].



Kuva 4: Suunnitteluikkuna (Design Window) [GaL99]

Kaikkia vaatimuksia ei voida kerralla havaita tai ymmärtää, vaan ne tulevat esille vasta kohdealueen analysointivaiheessa tai sen jälkeen. Lisäksi vaatimukset muuttuvat nopeasti. Tämän vuoksi tarvitaan vaatimusten hallintaa ja versiointia: vaatimukset kehittyvät iteratiivisesti prosessin edetessä.

Yhteenvetona vaatimusmäärittelyssä on seuraavanlaisia ongelmia:

- *Kommunikointiongelma*: Ohjelmistosuunnitteluun osallistuu väärä ihminen, joka ei osaa tai huomaa kertoa olennaisia asioita. asiakkaalla voi olla tiedostettuja, tiedostamattomia ja sellaisia tarpeita, joita hän ei osaa edes kuvitella (undreamed). Näistä tarpeista kahden viimeisen esille saaminen on kaikkein haas-

tavinta [RoR99 s.82]. Vaatimuksia ymmärretään väärin ja eri lähteistä saadaan ristiriitaista tietoa.

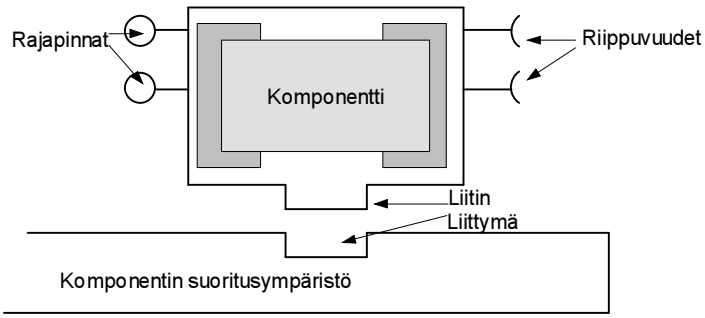
- *Analysointiongelma*: Ohjelmistosuunnittelijalla on puutteelliset ajatusmallit siitä, mitä tietoa tarvitaan suunnittelua varten, ja erityisesti komponenttisuunnitteluun.
- *Dokumentointiongelma*: Vaatimusmäärittelyssä laaditaan epätarkat tai epätarkoituksenmukaiset dokumentit.
- *Vaatimusmuutostenhallintaprosessin ongelma*: Muutoksia ei kirjata riittävän tarkalla tasolla. Muutoksista on kirjattava mihin muutos kohdistuu, mitä on muuttunut, miksi ja milloin sekä mihin muihin asioihin muutos vaikuttaa.

Vaatimusmäärittelyn kaikkiin vaiheisiin ja osa-alueisiin on siis kiinnitettävä huomiota, jotta vaatimusmäärittely onnistuisi. Vaatimusmäärittelyn onnistuminen on avainasemassa koko ohjelmistotuotantoprosessin onnistumisen kannalta.

Seuraavassa luvussa esitellään komponentteihin liittyviä peruskäsitteitä, komponenttituotantoon kohdistuvia odotuksia ja komponenttituotannon ongelmia sekä pohditaan, miten nämä vaikuttavat vaatimusmäärittelyyn.

3 Komponentit ja arkkitehtuurit

Komponentista on olemassa monenlaisia määritelmiä. Yhteistä niille kaikille kuitenkin on se, että *komponentti* määritellään itsenäiseksi *ohjelmayksiköksi,* joka tarjoaa palveluja *rajapinnan (interface)* kautta [Kos00]. Komponentti tarvitsee toimiakseen jonkin järjestelmän, *suoritusympäristön* (component execution environment, CEE), sekä *liittimen* (plug), jonka avulla se liittyy järjestelmän *liittymään* (socket). Vaikka komponentti onkin itsenäinen osa, se on tarkoitettu toimimaan yhdessä toisten komponenttien kanssa. Komponentti voi toimiakseen tarvita palveluja toisilta komponenteilta (dependencies) (kuva 5).



Kuva 5: Komponentti ja suoritusympäristö [HeS00]

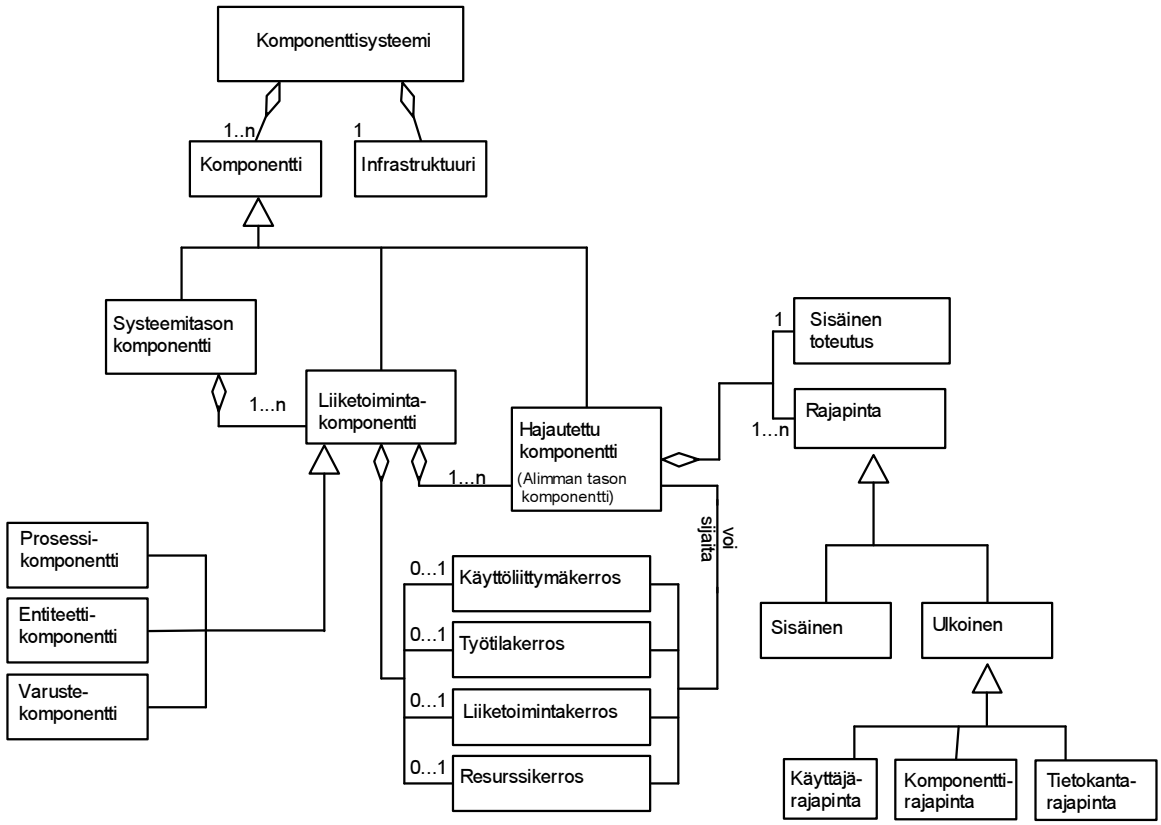
Komponenttipohjainen suunnittelu (Component-Based Development, CBD) määritellään ohjelmistosuunnittelun lähestymistavaksi, jossa ohjelmistotuotantoprosessin kaikissa vaiheissa, tuettavaa teknistä infrastruktuuria valittaessa sekä ohjelmistoprojektin hallinnassa lähtökohtana ovat komponentit [HeS00]. CBD on poiminut hyvät puolet perinteisistä menetelmistä, kuten oliotekniikoista (object-oriented ja distributed objects). Oliotekniikan perusajatuksista voidaan suoraan siirtää komponenttiajattelun pohjaksi seuraavat periaatteet:

- Komponentti koostuu tiedoista ja toiminnoista näiden tietojen käsittelemiseksi.
- Komponentin sisäinen rakenne on käyttäjältä piilotettu ja komponentin toimintoja kutsutaan sen rajapintojen kautta (kapselointi). [ChD01].

Vakiintuneet käytännöt (best practice) sekä menestyneet tekniikat ja periaatteet luovat hyvän teoriapohjan ohjelmiston komponenttipohjaiselle suunnittelulle. CBD:n avulla voidaan hallita laajoja monimutkaisia kokonaisuuksia (myös liiketoiminnan mallin-

taminen). CBD muuttaa tietojärjestelmien luonnetta monoliittisesta modulaariseksi [HeS00].

Komponentit voidaan ryhmitellä usean eri kriteerin perusteella, esimerkiksi rakeisuuden, muodon, rakentamis- tai hankkimistavan perusteella. *Rakeisuudella* tarkoitetaan komponentin kokoa ja sitä, millaisista osista komponentti muodostuu: mitä karkeampi rakeisuus, sitä suuremmista osista komponentti koostuu. Herzum & Sims jakaa komponentit rakeisuuden perusteella kolmeen pääluokkaan: hajautettu komponentti, liiketoimintakomponentti ja systeemitason komponentti [HeS00]. Näihin liittyviä käsitteitä on esitetty kuvassa 6.



Kuva 6: Komponenttikäsitteitä

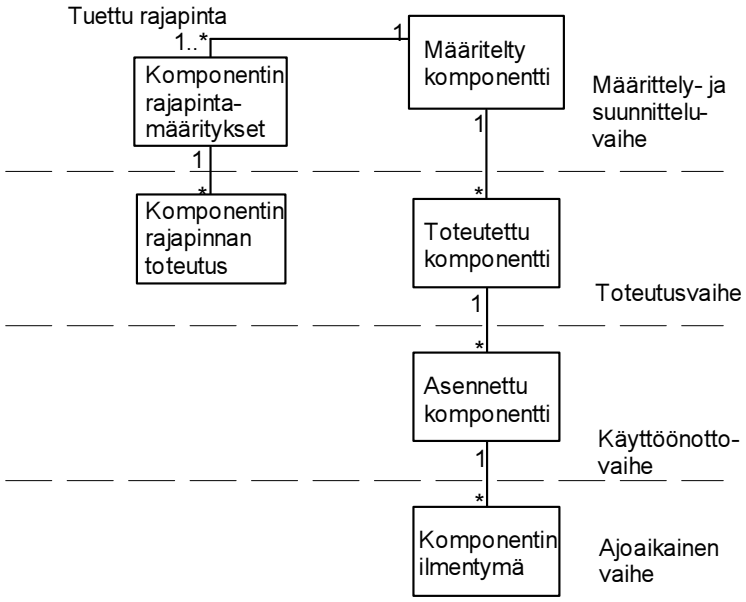
Hajautettu komponentti (Distributed Component, DC) on alimman rakeisuustason komponentti. Se ei sisällä enää toisia komponentteja. Hajautetut komponentit voivat sijaita liiketoimintakomponentin eri kerroksissa (kuva 6). Hajautetulla komponentilla on *sisäinen toteutus*, joka piilotetaan käyttäjältä sekä *rajapinta*, jonka avulla sen tarjoamat palvelut näytetään ulospäin. Rajapinta voi olla komponentin *sisäinen* tai *ulkoinen*. Ulkoisista rajapinnoista eritellään käyttäjään päin näytettävä *käyttäjärajapinta*,

toisiin komponentteihin päin toimiva *komponenttirajapinta* sekä tietokantaa käsittelevä *tietokantarajapinta*. Hajautettu komponentti on yleensä toteutettu komponenttiteknologialla, kuten J2EE, Corba tai .net [HeS00]. Kuitenkaan hajautetun teknologian käyttö toteutuksessa ei ole välttämätöntä, joten nimitys hajautettu komponentti on jossain mielessä harhaan johtava ja olisi hyvä käyttää esimerkiksi nimitystä *alimman tason komponentti* (low-level component).

Liiketoiminta- tai myös *toimialakomponentti* (Business Component, BC) on keskirakeinen komponentti, joka sisältää yhden liiketoimintaprosessin tai käsitteen tarvitsemat toiminnot ja käsitteet. Liiketoimintakomponentit voidaan jakaa edelleen kolmeen ryhmään käytön perusteella: prosessi- (process BC), entiteetti- (entity BC) ja varustekomponenteiksi (utility BC) (kuva 6) [HeS00]. Entiteettikomponentista voidaan käyttää myös nimitystä käsitekomponentti ja varustekomponentista avustava komponentti. Liiketoimintakomponentit toteutetaan nelikerrosarkkitehtuurilla, eli niissä on käyttöliittymä-, työtila-, liiketoiminta- ja resurssikerrokset (kuva 6). Liiketoimintakomponentin rajapinnat muodostuvat sen sisältämien hajautettujen komponenttien rajapintojen joukoista ja sisäinen toteutus hajautetuista komponenteista ja niiden välisistä suhteista. Tässä työssä tarkastellaan pääasiassa liiketoimintakomponentteja ja kun puhutaan pelkästään komponentista, tarkoitetaan liiketoimintakomponenttia.

Systeemitason (System-level) [HeS00] tai myös *yritystason komponentti* [Ars02] (Enterprise Component, EC) on suurimman rakeisuuden omaava komponentti, joka koostuu liiketoimintakomponenteista ja hajautetuista komponenteista. Yritystason komponentti voi olla itsenäinen sovellus [HeS00].

Cheesman ja Daniels luokittelevat komponentteja komponenttimuodon mukaan. Muodoltaan komponentti voi olla määritelty, toteutettu, asennettu tai ajoaikainen komponentti. Kuvassa 7 esitetään komponenttimuotojen välisiä suhteita. Komponentin *määrittelyissä* kuvataan komponentin toiminta. Komponentin sisäinen toteutus on kapseloitu ja näkyvyys ulospäin kuvataan rajapintojen avulla. *Rajapintamäärittelyssä* kuvataan komponentin tarjoamat operaatiot ja niiden tarvitsemat parametrit. Komponentin määrittelyjen perusteella tiedetään, mitä vaatimuksia komponentti toteuttaa. Yhdelle komponentille voidaan määritellä yksi tai useita eri rajapintoja. Yksi rajapintamäärittely voidaan toteuttaa usealla eri tavalla.[ChD01]



Kuva 7: Komponentin eri muodot ja niiden suhteet, mukailtu [ChD01]

Komponentin *toteutus* erotetaan määrittelystä, eli yhdellä tavalla määritelty komponentti voidaan toteuttaa usealla eri tavalla. Toteutus piilotetaan käyttäjältä, jolloin komponentti voidaan ajatella mustana laatikkona. Jos komponentti ei toteuta kaikkia vaatimuksia, voidaan siihen tehdä lisäyksiä, jotta vaatimukset saadaan toteutetuiksi.

Asennettu komponentti toimii jonkin kokoonpanon osana toimintaympäristössään. Komponentin on oltava jonkin standardin tai sopimuksen mukainen, ts. se on pystyttävä liittämään johonkin ympäristöön. Tämä standardointi kertoo komponentin liittimen muodon. Yksi toteutettu komponentti voidaan asentaa yhden tai usean kokoonpanon osaksi.

Yhdestä asennetusta komponentista voi olla yhtä aikaa useampia ajoaikaisia ilmentymiä (instances) toimimassa. Komponentti sisältää toiminnallisuutensa lisäksi myös toimintaan liittyvän tiedon hallinnan. Vaikka kaksi komponenttia tarjoaisivatkin täsmälleen samat palvelut, voidaan vanhan komponentin sisältämät tiedot menettää korvattaessa komponentti toisella, jos pysyvyys hoidetaan tallentamalla tiedot komponentin sisäiseen tietokantaan. Vaikka komponentti olisi asennettu, se ei tee mitään, ennen kuin siitä käynnistyksen kautta tulee ajoaikainen ilmentymä, ja vain komponentin ilmentymällä on oma identiteetti.

Komponentti voidaan tehdä itse tai hankkia valmiina markkinoilta, jolloin puhutaan *valmiskomponenteista* (Commercial-Off-The-Shelf, COTS). Komponentit voidaan räätälöidä asiakkaan tarpeen mukaisiksi käyttäen pohjana jo ennestään tehtyjä komponentteja tai parametrointia. Valmiskomponentit ovat kaupalliseen tarkoitukseen tuotettuja liiketoimintakomponentteja. Niiden etuna on se, että on huomattavasti halvempaa ostaa valmiita komponentteja kuin räätälöidä kaikki itse. Myös markkinoilletuloaika pienenee, mikäli COTSit ovat helposti löydettävissä ja täyttävät vaatimukset.

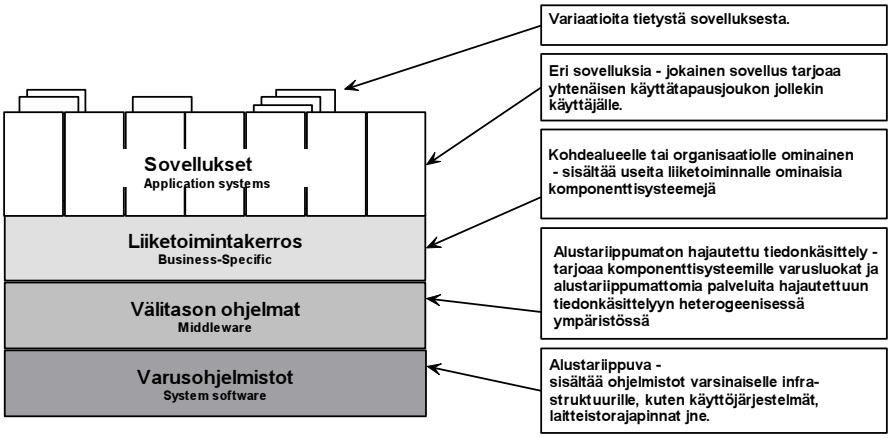
3.1 Arkkitehtuurit, suunnittelumallit ja komponentit

Arkkitehtuurit, suunnittelumallit ja komponentit ovat käsitteitä, jotka liittyvät läheisesti toisiinsa. Ohjelmiston varhaisessa suunnitteluvaiheessa näitä käsitteitä voidaan käyttää ohjaamaan ajattelua. Suunnittelumallien avulla voidaan kuvata niin arkkitehtuuria kuin komponenttejakin. *Arkkitehtuuri* kuvaa komponentit ja niiden väliset suhteet. Tietoisuus arkkitehtuurien, suunnittelumallien ja komponenttien luonteenomaisista piirteistä on hyödyksi jo vaatimusten keräämis- ja analysointivaiheessa. Seuraavissa kappaleissa esitellään näitä käsitteitä lähemmin vaatimusmäärittelyn näkökulmasta.

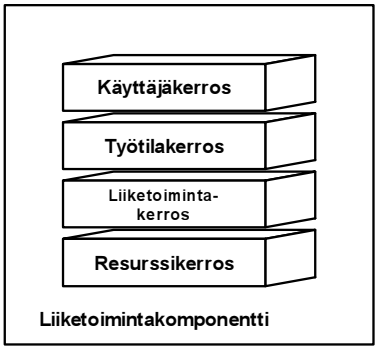
3.1.1 Arkkitehtuurit

Ohjelmistoarkkitehtuurilla (software architecture) kuvataan korkealla abstraktiotasolla sovelluksen rakenne, eli sovelluksen keskeiset osat ja niiden väliset staattiset ja dynaamiset suhteet sekä periaatteet (principles), joita noudatetaan [Kos00 s. 197]. Arkkitehtuuri voidaan ajatella joukoksi suunnitteluratkaisuja, jotka dokumentoivat suunnittelutietämystä ja vähentävät tarpeetonta uudelleen keksimistä [DSW99]. Yleisiä arkkitehtuurimalleja on standardoitu ja nimetty, joten niitä on helppo käyttää uudelleen suunnittelussa. Koska arkkitehtuurikuvaukset ovat korkealla abstraktiotasolla, voidaan niiden avulla keskustella jo hyvin aikaisessa vaiheessa suunniteltavasta systeemistä ja sen toteuttamisesta. Arkkitehtuurityylejä ovat esimerkiksi kerrosarkkitehtuuri, putki-suodatin-arkkitehtuuri, palvelu- tai tietovarastopohjainen arkkitehtuuri [Bos00]. Kerrosarkkitehtuuri voi olla kolme- tai nelikerrosarkkitehtuuri. Kerroksellisuutta voidaan ajatella olevan sekä komponenttisysteemissä (kuva 8), että yksittäisen komponentin sisällä (kuva 9). Palvelupyynnöt kulkevat ylhäältä alas. Eri kerrosten vä-

lillä tulee olla mahdollisimman vähän liikennettä ja kerroksia ei ohiteta. Jokainen kerros on tietoinen vain itseään alemmasta kerroksesta. Tästä johtuen komponenttisysteemin kerrokset ovat vaihdettavissa ja kokonaisuus on hallittavissa.



Kuva 8: Kerroksellinen systeemi, jossa sekä sovellus- että komponenttisysteemeitä [JaG97 s. 172]



Kuva 9: Liiketoimintakomponentin kerrokset [HeS99 s. 336]

Kun käytetään *tietokeskeistä* arkkitehtuuria, on järjestelmän ytimenä tietokanta, ja sovellusten toiminta määritellään tiedon käsittelynä. Tästä seuraa tiedon eheysongelmia komponenttijärjestelmissä. *Palveluarkkitehtuuri* tarkoittaa sitä, että jo suunnitteluvaiheessa järjestelmä jäsennetään joukoksi palveluja, joita komponentit tarjoavat toisilleen. Palvelut abstrahoidaan siten, että ne näytetään ulospäin rajapinnan avulla, ja toteutus jää komponentin sisään palvelun toteuttajan vastuulle. Kunkin komponentin sisällä on palveluiden tarvitsemaa pysyvää tietoa varten pieni looginen tietokanta, joka

sisältää viitteitä. Tämä tarkoittaa käytännössä sitä, että infrastruktuuri ohjaa tiedon johonkin tietokantaan ja valvoo tiedon eheyttä, tallennusta ja hakuja. [Käh00]

Arkkitehtuuria kuvataan monesta näkökulmasta. Eri näkökulmia tarvitaan, koska suunniteltavasta sovelluksesta keskustellaan eri asianosaisten (stakeholders) kanssa. Philippe Kruchtenin 4+1 mallissa näkökulmia on viisi [Kos00 s. 198 - 199] :

- *Käyttötapausnäkökulma* kuvaa järjestelmän käyttäytymisen ulkoapäin katsottuna.
- *Looginen näkökulma* kuvaa järjestelmään kuuluvat staattiset ohjelmayksiköt ja niiden välisen vuorovaikutuksen.
- *Prosessinäkökulma* kuvaa järjestelmän rinnakkaisten prosessien ja säikeiden organisoinnin ja vuorovaikutuksen.
- *Toteutusnäkökulma* kuvaa järjestelmän jakamisen fyysisiin osiin, esimerkiksi tiedostoihin, jotka kootaan tietyllä tavalla julkistuksiksi.
- *Sijoittelunäkökulma* kuvaa järjestelmän laitteistokokoonpanon, laitteiden tarvitsemat yhteydet sekä ohjelmistojen ja prosessien sijoittelun eri laitteisiin.

Herzum & Sims ovat määritelleet neljä ohjelmistotuotantoa ja kehitystyötä palvelevaa näkökulmaa seuraavasti [HeS00]:

- *Tekninen arkkitehtuuri* käsittää tekniset ominaisuudet, jotka tarvitaan komponenttipohjaisen sovelluksen kehittämisessä ja suorittamisessa, kuten suoritusympäristön, työkalut ja käyttöliittymäkehykset.
- *Sovellusarkkitehtuuri* sisältää päätökset käytetyistä arkkitehtuureista ja suunnittelumalleista, ohjeistuksesta sekä standardeista, joita tarvitaan perustoimintojen, kuten virheenkäsittelyn tai tietokannan käsittelyn, toteuttamiseen edellä lueteltujen teknisten asioiden lisäksi.
- *Projektinhallinta-arkkitehtuuri* kuvaa ohjelmistosuunnittelussa käytettävät yleiset suunnitteluperiaatteet, eli ne asiat, jotka mahdollistavat suuren tiimin kustannustehokkaan yhteistyön komponenttipohjaisen ohjelmiston tuottamiseksi.
- *Toiminnallinen arkkitehtuuri* käsittää systeemin toiminnallisuuden, joka toteuttaa toiminnalliset vaatimukset.

Herzumin jaottelu poikkeaa hieman Kruchtenin näkökulmista, mutta esimerkiksi Herzumin käyttötapausnäkökulma kuvaa suunnilleen samat asiat kuin Kruchtenin toiminnallinen arkkitehtuuri. Herzumin arkkitehtuurikuvausten tarkoituksena on toimia nimenomaan ohjelmistokehityksen välineenä ja apuna, kun taas Kruchtenin näkökulmat palvelevat ohjelmistotuotannon muitakin vaiheita, kuten esimerkiksi markkinointia.

Arkkitehtuurikuvausten avulla on mahdollista jo suunnitteluvaiheessa ymmärtää ja jäsentää tulevan systeemin toimintaa, tarpeita ja rajoituksia. Systeemin laatuvaatimukset ohjaavat arkkitehtuurisuunnittelua. Laatuvaatimuksia ovat esimerkiksi ylläpidettävyys, suorituskyky ja käytettävyys. Ohjelmistolle suunnitellaan toiminnallinen arkkitehtuuri, tarkastetaan laatuvaatimusten toteutuminen ja tehdään tarvittavat muutokset. Näin voidaan varmistaa ohjelmiston laatu jo ohjelmistotuotannon aikaisessa vaiheessa [Bos00].

Standardoidut arkkitehtuurityylit toimivat kommunikaatiovälineenä sekä ohjelmistotuottajien ja erilaisten sidosryhmien välillä että ohjelmistotuotannon sisällä. Ohjelmistotuotannon sisällä arkkitehtuureja voidaan kuvata arkkitehtuurinkuvauskielillä (Architecture Description Language, ADL).

3.1.2 Suunnittelumallit

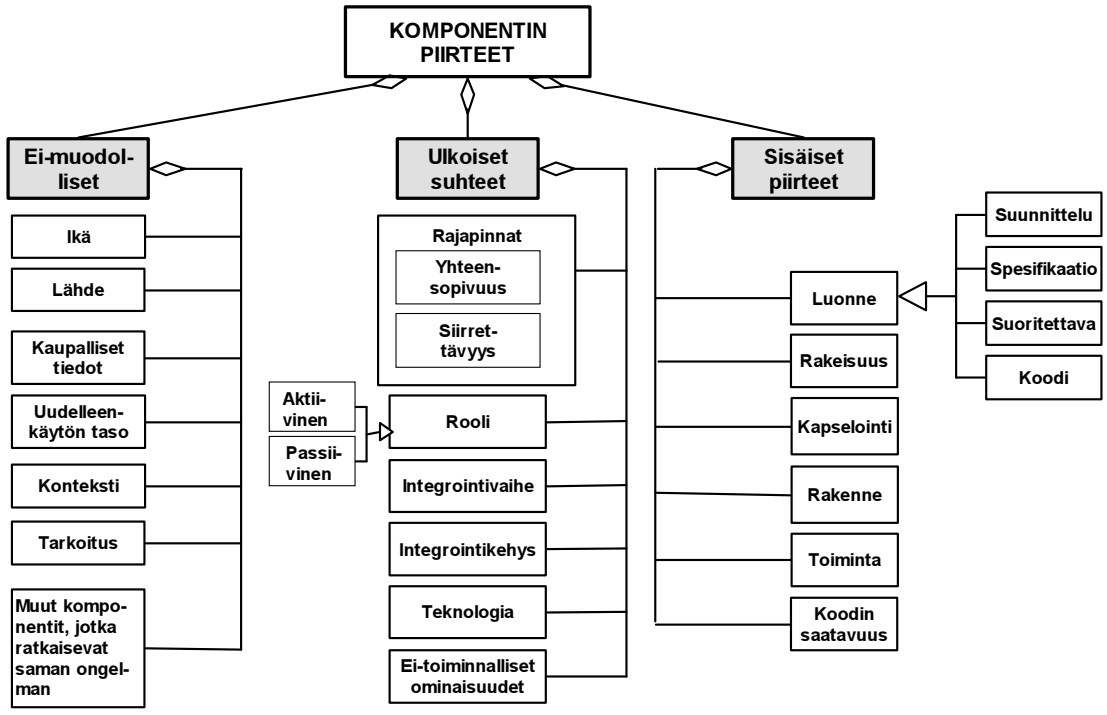
Suunnittelumallilla (design pattern) kuvataan toimivaksi todettua tapaa ratkaista jokin ohjelmistotuotannossa toistuvasti esiintyvä ongelma. Suunnittelumalliajattelu on alun perin kotoisin rakennusten arkkitehtuurisuunnittelusta. Ajatuksena on, että kun on keksitty hyvä ratkaisu johonkin ongelmaan, kuvataan ongelma ja sen ratkaisumalli niin hyvin, että samaa ratkaisua osataan käyttää uudelleen. Kokeneiden ohjelmistotuottajien tietotaitoa voidaan näin siirtää ja jakaa edelleen. Hyvin kirjoitetussa suunnittelumallin dokumentissa kuvataan huolellisesti seuraavat elementit: mallin nimi, tarkoitus, ongelma, konteksti, vaikuttavat voimat, ratkaisu, seuraukset ja toteutus [GaH94, Imm00].

Suunnittelumalli on hyvin ilmaisuvoimainen kuvaustapa. Sen avulla voidaan kuvata suunnitteluratkaisuja sekä yleisen että tarkan teknisen tason ongelmiin. Suunnittelumallit edistävät ohjelmistojen ja niiden osien modulaarisuutta, uudelleenkäytettävyyt-

tä, joustavuutta ja ymmärrettävyyttä. Näitä samoja tavoitteita asetetaan komponenttituotannolle. On siis hyvin luontevaa käyttää suunnittelumalleja apuna komponenttituotannon eri vaiheissa.

3.1.3 Komponentin erityispiirteistä

Ohjelmistokomponentteja on monenlaisia ja niillä on erilaisia luonteenomaisia piirteitä. Komponentin eri piirteet voidaan jakaa kolmeen kategoriaan: ei-muodolliset piirteet, komponentin ulkoiset suhteet ja komponentin sisäiset piirteet (kuva 10). Tämä kategorisointi ja piirteiden ymmärtäminen helpottaa komponenttien toiminnan ja olemuksen ymmärtämistä sekä komponenttien luokittelua. Seuraavassa on kerrottu kuhunkin kategoriaan kuuluvista piirteistä. Kategorisointia voidaan käyttää apuna komponenttituotannon eri vaiheissa.[YaA99]



Kuva 10: Komponentin piirteet luokiteltuna

Komponentin ei-muodolliset piirteet on esitetty kuvassa 10 vasemmalla. *Ei-muodollinen kuvaus* sisältää ihmisten kannalta olennaisia tietoja komponentista. Komponentin *ikä* kertoo komponentin kypsyydestä ja toimintavarmuudesta: mitä use-

ammin komponenttia käytetään, sen kypsemmäksi komponentti tulee, ja käytön riski pienenee. Komponenttia voi luonnehtia sen *hankintalähteen* perusteella, esimerkiksi Commercial-Off-the-Shelf (COTS) tai Military-Off-the-Shelf (MOTS). *Kaupalliset ominaisuudet,* kuten hinta, vaikuttavat ostopäätökseen. *Uudelleenkäytön taso* kertoo, missä komponentin elinkaaren vaiheessa komponenttia voi käyttää uudelleen, esimerkiksi koodinpala on käytössä kehitystyön aikana ja Dynamic Link Library (DLL) on käytössä ajoaikana. *Konteksti* kertoo komponentin kohdealueen ja sovellukset, jossa sitä voidaan käyttää. Komponentin *tarkoitus* kuvataan ja samoin *ongelma*, joka ratkaisemiseksi komponentti on tehty. Komponentin ymmärtämistä helpottaa, jos luetellaan ja kuvataan myös muita tunnettuja komponentteja, jotka ratkaisevat saman ongelman.

Komponentin ulkoiset suhteet, jotka on esitetty kuvassa 10 keskellä, kuvaavat komponentin yhteistoiminnan muiden sovellusalueen artifaktien ja suoritusympäristön kanssa. Komponentin yhteistoiminnallisuus muiden komponenttien ja sovellusten kanssa kuvataan *sovellusrajapintojen* (Application Interfaces) avulla. *Alustarajapintojen* (Platform Interfaces) avulla kuvataan komponentin siirrettävyys, eli missä suoritusympäristössä komponentti toimii, ja miten se voidaan siirtää toisiin ympäristöihin.

Myös komponentin *rooli* kuvaa komponentin suhteita ulospäin. Komponentti voi olla rooliltaan aktiivinen tai passiivinen. *Aktiivinen* tarkoittaa sitä, että komponentti voi vaikuttaa (affect) toisiin artifakteihin, esimerkiksi käyttöliittymään. *Passiivinen* komponentti ei itse kutsu toisia artifakteja, esimerkiksi tietokanta.

Lisäksi komponentin ulkoisiin piirteisiin kuuluu *integrointivaihe*. Komponentti voidaan integroida sovellukseen eri vaiheissa, esimerkiksi kehitysvaiheessa tai ajoaikana. Komponentit voivat toimia integroituna sovelluksena joko suoraan, tai jonkin *integrointikehyksen* (Integration Framework) avulla. *Toteutustekniikka* ja *ei-toiminnalliset* vaatimukset voidaan myös ajatella ulkoisiin piirteisiin. Komponentti voi olla riippuvainen jostakin toteutustekniikasta. Spesifikaatiotasolla määritelty komponentti on yleensä tekniikkariippumaton. Komponentin ei-toiminnallisia ominaisuuksia ovat esimerkiksi tietoturvaan liittyvät ominaisuudet, ajoaikaiset prosessointivaatimukset ja komponentin luotettavuus.

Kuvassa 10 oikealla on esitetty komponentin sisäiset piirteet. Komponentin luonne on sisäinen piirre, joka kertoo, missä kehitysprosessin vaiheessa komponenttia voidaan käyttää. Suunnittelumallit, kuten esimerkiksi *Tarkkailija* (Observer) tai *Julkisivu* (Facade) [GaH94] voidaan ajatella komponenteiksi, joita käytetään suunnitteluvaiheessa kuvaamaan tulevan sovelluksen rakennetta. Määrittelyvaiheessa tuotettu spesifioitu komponentti on yleensä tekniikkariippumaton, mutta sen toiminnallisuus ja käyttäytyminen on määritelty. Suoritettava komponentti voi olla esimerkiksi DLL tai joku muu suoritettava ohjelmiston osa, yleensä kaupallinen musta laatikko, jonka koodi on piilotettu. Komponentti voidaan ajatella myös koodinpalana, jota käytetään implementointivaiheessa.

Sisäisiin piirteisiin kuuluu myös komponentin *rakeisuus*, joka kuvaa komponentin kokoa ja sitä miten laajaan liiketoiminnan kokonaisuuteen se liittyy (yritystaso, liiketoiminnan prosessi tai osaprosessi jne). *Kapseloinnilla* piilotetaan käyttäjältä komponentin suunnittelu- ja toteutusratkaisut. Komponentin *rakennepiirteet* sisältävät sisäiset rakenneosat ja niiden väliset suhteet yleisellä tasolla, esimerkiksi luokkakaaviona. Komponentin *toiminnalliset piirteet* voidaan kuvata komponentin vasteina erilaisiin herätteisiin. Komponenttia luonnehtii myös *koodin saatavuus*. Koodi voi olla käyttäjältä piilotettu, käyttäjälle näkyvä mutta ei muokattava tai käyttäjälle näkyvä ja muokattavissa oleva.

3.2 Komponenttituotannolle asetettuja vaatimuksia

Muilla tuotannon aloilla (ns. kypsä teollisuus, Mature Industry) on luonteenomaista, että käytetään paljon komponenttiajattelua. Tuotteita koostetaan valmiista komponenteista ja osista, jotka voidaan ostaa markkinoilta esimerkiksi alihankkijoilta tai jotka löydetään omasta varastosta. Ylläpito on edullisempaa, kun voidaan keskittyä pieneen osaan koko tuotteen sijasta. Koko toimitusketjun kustannustehokkuuteen kiinnitetään huomiota. Lisäksi komponenttien käytöstä tulee olla hyötyä myös tuotteen käyttäjälle. [HeS00]

Seuraavassa tarkastellaan lähemmin sitä, mitä hyötyä komponenttien käytöstä on ohjelmistotuotannossa sekä sitä, millainen komponentin pitää olla ja miten tulee toimia, jotta hyödyt saavutetaan.

3.2.1 Miksi komponentteja?

Komponentti ajatellaan mustana laatikkona, jonka toteutus ja sisäinen rakenne piilotetaan käyttäjältä. Komponentin tarjoamat palvelut kuvataan korkean tason rajapintojen avulla. Tällöin ohjelmiston osien uudelleenkäytettävyys helpottuu, kun integroijan ei tarvitse selvittää koodia lukemalla komponentin toimintaa. Uudelleenkäyttö säästää aikaa ja rahaa [Käh00]. Uudelleenkäyttö parantaa myös komponentin toimintavarmuutta [YaA99].

Kehittynyt infrastruktuuri peittää alleen järjestelmien erilaisuudet ja tekee automaattisesti tarvittavat konversiot (muunnokset) esimerkiksi protokollissa tai tiedon esitystavassa. Komponentti-infrastruktuuriin voidaan upottaa järjestelmänhallinnan tarvitsemat toiminnot ja infrastruktuurin sisään voidaan rakentaa erilaisia tietoturvan tasoja. Komponentti-infrastruktuurissa on valmiiksi rakennettua toiminnallisuutta ja valmis kehikko uusille, mikä vähentää ohjelmistosuunnittelutyötä. Kehittyneen infrastruktuurin avulla laajojen järjestelmien toteutuksen hallinta on helpompaa ja saadaan parempi tuki heterogeenisille järjestelmille. [Käh00]

Cheesman ja Daniels pitävät tärkeimpänä syynä komponenttiajatteluun muutoksenhallinnan vaatimusta. Tämä tarkoittaa sitä, että arkkitehtuurissa ja suunnittelussa kiinnitetään päähuomio komponenttien välisiin riippuvuuksiin ja riippuvuuksien hallintaan, toisin sanoen kokonaisuuteen, ei osiin. Perusteluna esitetään se, että nykymaailmassa muutos on hallitseva tekijä ja tämä edellyttää sitä, että komponentteja on voitava helposti korvata toisilla, jolloin päähuomio on arkkitehtuurissa eli koko systeemin hallinnassa. Vaatimukset muuttuvat nopeasti, jolloin yksittäisiä komponentteja muutellaan ja kehitetään kokonaisuuden osana kokonaisuuden muuttumatta. [ChD01]

3.2.2 Komponentteihin kohdistuvia vaatimuksia

Komponenttien uudelleenkäyttö edellyttää, että on olemassa joku paikka, mistä komponentteja saadaan uudelleen käytettäväksi. Valmiit komponentit voidaan hankkia joko vapailta markkinoilta (COTS) tai ohjelmistoyrityksen omasta komponenttivarastosta. Jotta komponenttimarkkinat voisivat toimia, on oltava olemassa sekä ostajia että myyjiä, eli tarpeita ja tavaraa. Jonkun on siis suunniteltava komponentit komponent-

timarkkinoita silmälläpitäen. Komponenttien tuottajien tulee olla tietoisia niistä komponenttisovelluksista, joihin komponentteja tarvitaan nyt ja tulevaisuudessa. Tarvitaan tietoa myös kohdealueista, jotta voidaan suunnitella toimialakohtaisia komponentteja (domain-specific components). Myytävistä komponenteista tulee saada riittävästi tietoa, jotta niitä voidaan käyttää. On siis oltava komponenttiluetteloita, joissa kuvataan komponentit ja niiden tarkoitus. Komponenttien luokittelu helpottaa oikean komponentin löytämistä. Myös komponenttikehys on oltava määritelty, eli se ympäristö, jonka komponentti tarvitsee toimiakseen.

Komponenttien on kyettävä toimimaan yhdessä, eli tarvitaan standardeja ja protokollia, jotka ovat komponentin tarvitsijan saatavilla. Osittainen ylläpito vaati, että on voitava kohdistaa muutoksen tarve tiettyyn komponenttiin, jotta komponentti voidaan vaihtaa toimittajasta riippumatta. Komponenttien tulee olla helposti räätälöitäviä ja niistä on pystyttävä muodostamaan helposti monenlaisia koosteita kulloiseenkin tarpeeseen. Komponenteilla tulee olla hyvin määritellyt selkeät rajarajapinnat ja selvästi määritelty toimintaympäristö, jolloin saavutetaan hyvä modulaarisuus ja korkea integroitumistaso. Standardeja tarvitaan siis sekä toiminnallisten että teknisten ominaisuuksien määrittelyyn. [HeS00]

Komponentin tulee toimia oikein ja luotettavasti, mutta se ei pelkästään riitä. Komponentin on oltava myös hyvin määritelty. Mikäli komponentti ei ole hyvin määritelty, sen uudelleenkäyttö estyy, koska sitä ei osata löytää käytettäväksi. Kunnollinen komponenttivarasto on edellytys komponenttien tehokkaalle uudelleenkäytölle. Komponenteista tulee tallentaa varastoon koodin lisäksi myös tekniset kuvaustiedot, käyttörajoitukset ja versionhallintatiedot [Käh00]. Lisäksi komponentista tarvitaan yleisen tason ei-muodollinen kuvaus, joka sisältää ihmisen kannalta olennaisia tietoja komponentista [YaA99]. Näiden tietojen avulla voidaan arvioida komponentin soveltuvuutta uuteen käyttökohteeseen. Hyvä komponentti toimii oikein ja luotettavasti hyvin määritellyssä ympäristössä hyvin määritellyllä tavalla.

3.2.3 Ohjelmistotuotannon työtapoihin kohdistuvia vaatimuksia

Liiketoiminnan muutoksiin ja toisaalta tekniikan kehitykseen pitää pystyä vastaamaan nopeasti. Komponenttimarkkinoilla on toimittava nopeasti, komponentin kehitys ei

saa kestää liian kauan. Kun uudelleen käytettävän komponentin kehitystyö on yleensä hitaampaa ja työläämpää kuin kertakäyttöisen ohjelmiston osan, komponenttityössä on järkevää käyttää suunnittelumalleja ja aiemmin hyväksi havaittuja ratkaisuja apuna, jolloin kehitystyö nopeutuu.

Komponenttituotannossa eri projektien koordinointi ei aina toimi ohjelmistoyrityksen sisällä. Rinnakkaisissa tai eri projekteissa toteutetut komponentit ovat toiminnaltaan päällekkäisiä tai limittäisiä, jolloin niistä on vaikea koota yhteen toimivia kokonaisuuksia. Siksi komponenttituotannossa tarvitaan standardointia ja tuotannon järjestelyä kokonaisuudeksi.

Kannatta myös miettiä, miten komponenttisuunnittelu saadaan tuntumaan ohjelmistoammattilaisistakin houkuttelevammalta. Esimerkiksi kannustuspalkkiot voisivat perustua myös uudelleenkäyttöön, ei pelkästään tuotettujen koodirivien tai sovellusten määrään.

3.3 Komponenttien vaatimusmäärittelystä

Komponenttisysteemin suunnittelussa on kohdealueen ja suunniteltavan systeemin toiminnan lisäksi otettava huomioon komponenttiarkkitehtuurin ja komponenttien toiminnan mukanaan tuomat vaatimukset. *Uudelleenkäytettävyyden vaatimus* tarkoittaa sitä, että komponentit on suunniteltava siten, että niitä voidaan mahdollisimman pienin muutoksin käyttää uudelleen. Lisäksi komponentin on oltava *muunneltava, testattava ja skaalautuva* sekä riittävän *yleinen.* Skaalautuvuudella tarkoitetaan komponentin kyvykkyyttä toimia oikein muuttuvissa olosuhteissa, esimerkiksi käyttäjämäärien kasvaessa [www05].

Komponenttiajattelun tulee ohjata jo vaatimusten keräämisvaihetta: tietoja on ryhmiteltävä ja vaatimuksia osoitettava suoraan oikeille liiketoimintakomponenteille. Liiketoiminatakomponentit on tällöin ensin osattava löytää ja määritellä. Komponentit ja komponenttisysteemin osat abstraktilla tasolla on siis selvitettävä jo ennen projektin alkua. On tiedettävä, mistä osista komponentit ja komponenttisysteemit muodostuvat: on oltava komponenttimalleja. Eri toimialueilla on erilaiset vaatimukset ja toiminnot, tarvitaan siis kohdealuekohtaisia (domain-specific) malleja. Kun vaatimukset ja niistä

johdetut käyttötapaukset jaetaan heti alussa suunnitelluille liiketoimintakomponenttiehdokkaille, saadaan systeemin monimutkaisuus hallintaan [HeS00]. Komponenttien eri luonteenpiirteiden ymmärtäminen helpottaa myös vaatimusmäärittelyvaihetta [YaA99]. Jos kohdealuekohtaisia käyttötapauksia ja niihin liittyviä toteutuksia on hyvin määriteltyinä varastossa, niiden uudelleenkäyttö on helppoa ja nopeaa.

Vaatimustenkeräämisvaiheessa tulee ryhmitellä tarvittavia palveluja ja toimintoja miettimällä niiden samankaltaisuuksia ja eroavaisuuksia. Vaatimusten keräämisen jälkeen voidaan suorittaa eräänlainen komponenttianalyysi. Varsinainen *komponenttianalyysi* liittyy hahmontunnistukseen. Siinä suuresta tietomassasta tunnistetaan samankaltaisia osia, joita tutkitaan ja ryhmitellään ja joista tehdään johtopäätöksiä. Ohjelmistosuunnittelussa vaatimusten komponenttianalyysi johtaa eri tyyppisten komponenttien tunnistamiseen ja löytämiseen. Kun tämä analyysi erotetaan omaksi toiminnakseen ohjelmistotuotantoprosessissa, voidaan apuna käyttää komponenttiasiantuntijoita, joilla on tarvittava tietämys eri tyyppisistä komponenteista, komponenttisysteemeistä sekä markkinoilla olevista valmiskomponenteista.

Komponenttisuunnittelua helpottaa myös se, että analysointivaiheessa asiakasvaatimukset voidaan yhdistää arkkitehtuurillisiin elementteihin. Tämä edistää kokonaisuuden hallintaa, asiakasvaatimusten jäljitettävyyttä ja ei-toiminnallisten vaatimusten mallintamista. Grünbacher ja kumppanit esittävät tähän jakamiseen CBSP-mallin (Component-Bus-System-Properities), jonka mukaan asiakasvaatimuksesta erotellaan sen komponenteille, yhdistimille ja systeemille kohdistuvat vaatimukset. Esimerkiksi asiakasvaatimuksesta *Systeemissä on oltava Web-selainrajapinta* voidaan johtaa komponenttitason vaatimus *Web-selaimen tuleen sisältyä yhtenä komponenttina systeemiin* ja yhdistimeen kohdistuva vaatimus *Systeemissä on oltava yhdistin, joka mahdollistaa yhteistoiminnan kolmannen osapuolen komponenttien kanssa.* [GrE01]

Ohjelmistoyrityksen sisällä voidaan myös käyttää metamallinnuksen keinoja, jolloin uudelleenkäytön kohteena on koodin lisäksi myös suunnittelumallit ja dokumentaatiot sekä ohjelmisto-osaaminen. Tässä voidaan käyttää apuna järjestelmänkehitystyökaluja, kuten metaEdit+ tai MetaCASE, jotka ovat metaCASE- välineitä (Computer Aided Software Engineering, CASE) [ZhL01].

Komponenttituotannossa on tarpeen olla historiallinen, tulevaisuuteen suuntautunut näkökulma. Jo vaatimuksia kerättäessä voidaan miettiä liiketoiminnan tulevaa kehitystä. Tällöin pystytään kiinnittämään huomiota siihen, millä tavoin komponenttien tulee olla muunneltavia ja millaisia muutospisteitä niihin voidaan rakentaa. *Muutospisteellä* tarkoitetaan sitä kohtaa komponentissa, jota muuttamalla voidaan komponentin ominaisuuksia muunnella [JaG97]. Samoin tulee kiinnittää huomiota sekä sisällölliseen että tekniseen skaalautuvuuteen. Komponentin tulee skaalautua erilaisten käyttäjien tarpeisiin, erilaisiin yhteysnopeuksiin ja -tapoihin sekä uusiin tekniikoihin [www05].

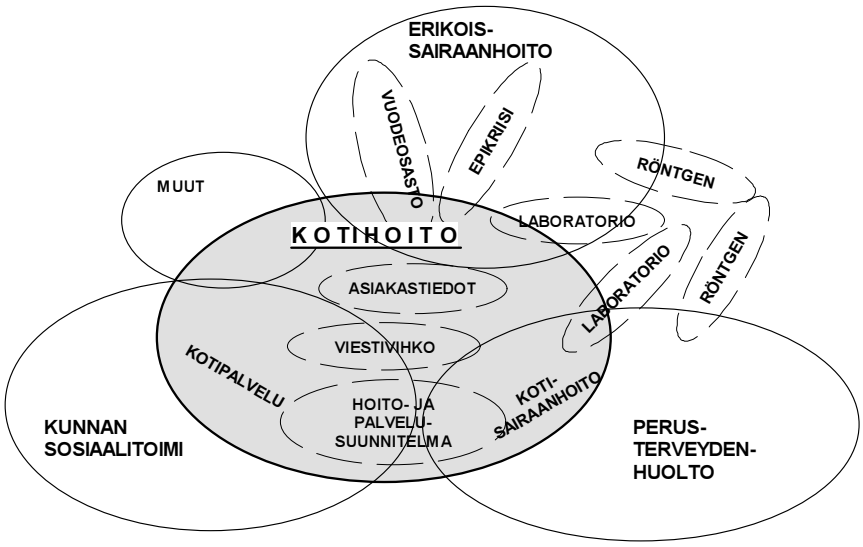
Komponenttien tulee olla testattavia kaikilla rakeisuustasoilla. Testauksessa lähdetään liikkeelle alimman tason komponenteista ja edetään karkeampijakoisiin komponentteihin [ToM03]. Komponenttien testaamisessa voidaan käyttää apuna UML:n käyttötapauksista johdettuja testitapauksia [Jän03]. Käyttötapaukset saadaan suoraan vaatimusmäärittelystä.

Komponentin suunnittelijalla on oltava käsitys sekä yksittäisten komponenttien toiminnasta ja käyttäytymisestä että komponenttien välisestä vuorovaikutuksesta ja yhteistoiminnasta, sillä usean komponentin järjestelmä saattaa käyttäytyä toisin kuin yksittäisten komponenttien toiminnan perusteella voisi olettaa. On otettava huomioon tarkoituksenmukainen kapselointi, hyvin määritellyt rajapinnat, sopiva toiminnallisuuden rakeisuus, tasapaino uudelleenkäytettävyyden ja erikoistumisen välillä sekä miten hyvin tavoiteltu toiminnallisuus toteutuu (completeness of functional coverage). [Par03]

4 Kotihoito

Tässä tutkimuksessa esiteltyjä menetelmiä havainnollistetaan kotihoidon kohdealueella luvuissa 5 – 7. Kotihoito on yksi tutkittavista osa-alueista laajassa PlugIT-hankkeessa, joka kokonaisuutena tutkii terveydenhuollon tietojärjestelmien integrointia. Kirjoittaja on toiminut harjoittelijana PlugITin Kotihoito-projektissa, jossa kartoitettiin kotihoidon tiedon tarpeita. Menetelmänä käytettiin ActAD-viitekehystä, jonka soveltuvuutta vaatimusten keräämiseen tutkittiin ja samalla kehitettiin menetelmää. Tutkimusalue oli Kuopion kotihoidon Leväsen alue.

Kotihoidolla tarkoitetaan asiakkaan kotona annettavia hoito- ja hoivapalveluja, jotka ovat osa terveys- ja hyvinvointipalveluja (kuva 11). Kotihoidon pääasiallinen kohderyhmä ovat vanhukset. Muita kohderyhmiä ovat esimerkiksi laitos- tai sairaalahoidosta kotiutuvat henkilöt sekä lapsiperheet. [ToL04a]



Kuva 11: Kotihoito terveydenhoidon ja hyvinvoinnin kentällä

Useimmat ihmiset haluavat asua omassa kodissaan mahdollisimman pitkään. Kun ihminen ikääntyy, toimintakyky laskee ja itsenäinen kotona selviytyminen alkaa tuottaa vaikeuksia. Monet vanhukset elävät yksin tai puolisonsa kanssa, joka voi olla yhtä huonossa kunnossa. Lapset asuvat kaukana, tai heillä ei ole muuten mahdollisuutta huolehtia vanhemmistaan päivittäin. Apua tarvitaan sekä päivittäisiin toimintoihin että terveyden ja sairauksien hoitoon. Päivittäin saatetaan tarvita apua esimerkiksi pukeu-

tumisessa, aterioinnissa tai hygienian hoidossa. Kun liikkuminen iän myötä vaikeutuu, on helpompi järjestää osa sairaanhoitoon liittyvistä toimista asiakkaan kotona, kuin asiakkaan mennä terveyskeskukseen (esimerkiksi haavahoito tai pistokset). Kotihoidolla mahdollistetaan paljon hoitoa ja apua tarvitsevien kotona asuminen [Sin95].

4.1 Kotihoidon osa-alueet ja prosessit

Kotihoidon tärkeimmät selkeästi erotettavat osa-alueet ovat kunnan perusterveydenhuoltoon kuuluva *kotisairaanhoito*, jonka tehtävänä on antaa asiakkaalle tarvittava sairaanhoidollinen palvelu, ja sosiaalitoimen piiriin kuuluva *kotipalvelu*, jonka tehtävänä on auttaa asiakasta selviämään päivittäisistä toimista (kuva 11). Muita kotihoidon osa-alueita ovat *neuvonta* ja erilaiset *tukipalvelut*, esimerkiksi turvapuhelinpalvelu, ateriapalvelu, kuljetuspalvelu. Lisäksi on erilaisia yksityisen sektorin *hoivapalveluyrityksiä*, jotka tarjoavat monipuolisia kotihoidon palveluja. [ToL04a, ToH03]

Kotihoito kokonaisuudessaan on kohdealueena hyvin laaja ja monimutkainen, koska kotihoitoon osallistuu monen eri ammattiryhmän sekä monen eri organisaation edustajia. Lisäksi työ tapahtuu asiakkaan kotona, jossa organisaatioiden tietovälineet eivät ole käytettävissä [ToH04]. Myös eri tyyppisten asiakkaiden kotihoitoprosessit ovat keskenään erilaisia. Soveltavat esimerkit tutkituista vaatimusmäärittelymenetelmistä on kohdistettu vanhuksille suunnattuun kotihoitoon. Lisäksi esimerkeissä on yksinkertaistettu kotihoidon monimutkaisia prosesseja ja keskitytty menetelmien soveltamisen kannalta kiinnostaviin asioihin. Tarkempaa ja yksityiskohtaisempaa tietoa kotihoidosta löytyy esimerkiksi lähteistä [Sin95, PäH02 ja VoV02].

Kotihoidon prosessi asiakkaan näkökulmasta alkaa yhteydenotolla kotihoitoon, joko kotisairaanhoitajaan tai kotipalvelunohjaajaan. Ensimmäisessä yhteydenotossa tehdään alustava tarvekartoitus, eli selvitetään, minkä tyyppistä apua tarvitaan ja kuinka usein. Esitietojen pohjalta tehdään asiakkaan kotiin suunnittelu- ja arviointikäynti, jolloin laaditan *Hoito- ja palvelusuunnitelma*. Suunnitelmaan kirjataan asiakkaan hoitotavoite, suunnitellut asiakaskäynnit ja niillä tehtävät toimenpiteet. Samalla käynnistetään usein tarvittavat tukipalvelut, kuten esimerkiksi ateria- tai turvapuhelinpalvelu. Suunnittelukäynnille osallistuu yleensä *kotipalvelun ohjaaja* tai kotisairaanhoitaja sekä kotipalvelusta asiakkaalle nimetty *omahoitaja*. Kotihoitoprosessi jatkuu suunnitel-

tujen hoitotoimien toteuttamisella: kotipalvelu käy sovittuina aikoina avustamassa asiakasta päivittäisissä toimissa ja kotisairaanhoitaja huolehtii terveydenhoitoon liittyvistä toimista. Tarvittaessa kotisairaanhoito ja kotipalvelu tekevät yhteiskäyntejä asiakkaan luo. Asiakkaan tilannetta seurataan kotikäynneillä ja tarvittaessa tehdään muutoksia Hoito- ja palvelusuunnitelmaan. [ToL04a]

4.2 Kotihoitoon liittyvistä tietojärjestelmistä

Kotihoidossa tarvittavaa, asiakasta koskevaa tietoa voi olla tallennettuna eri organisaatioiden tietojärjestelmiin, kuten esimerkiksi perusterveydenhuollon, erikoissairaanhoidon, yksityisen lääkärin vastaanoton, hoivapalveluyrityksen, Kansaneläkelaitoksen (KELA) ja kunnan sosiaalitoimen järjestelmiin. Esimerkiksi erikoissairaanhoidossa on useita eri järjestelmiä, joihin tallennetaan asiakasta koskevaa tietoa, esimerkiksi laboratorio-, röntgen-, vuodeosasto- ja ajanvarausjärjestelmät. Eri organisaatioiden järjestelmät eivät nykyisellään kommunikoi keskenään, eivätkä välttämättä edes yhden organisaation sisällä eri järjestelmät keskenään. asiakasta koskevaa tietoa on tallennettuna myös erilaisille paperisille tietovälineille, kuten Hoito- ja palvelusuunnitelma tai asiakkaan kotona oleva viestivihko. Lisäksi kotihoidossa on paljon hiljaista tietoa eri ammattilaisilla, joilla saattaa olla vuosikymmenten kokemus kotihoidosta ja jotka tuntevat asiakkaansa ja heidän historiansa.

Saman asiakkaan luona käy useita eri kotihoidon toteuttajia eri organisaatioista ja usein eri aikoihin. Asiakkaan hoidossa on tavoitteena saumaton palveluketju, joka perustuu moniammatilliseen yhteistyöhön ja edellyttää saumatonta tiedonkulkua. asiakkaan ja hänen hoitoonsa osallistuvien on tarpeen saada tietoa, jonka perusteella pystytään nopeasti muodostamaan kokonaiskuva palveluketjun keskeisistä tavoitteista, toimijoista, tapahtumista ja etenemisestä. Koska ketju muodostuu eri organisaatioiden antamasta palvelusta, on tiedonkulun ajantasainen toteuttaminen erityisen vaikeaa [VoV02]. Kotihoidon tietojärjestelmän tulee turvata, että oikea tieto on oikeaan aikaan oikeassa paikassa, esimerkiksi tietoa hoitotyön toteuttamiseen tarvitsevan toimijan saatavana [ToH03].

Asiakkaaseen liittyvät hoidon toteuttamisen kannalta tärkeät tiedot kulkevat eri toimijoiden välillä vaihtelevien käytäntöjen mukaan [ToH03]. Asiakkaan kotona säilytettävä Hoito- ja palvelusuunnitelma toimii tärkeänä yhteistyön välineenä kotihoidon eri toimijoiden välillä. ”Hoito- ja palvelusuunnitelmassa on kirjattuna kokonaiskuva asiakkaan tarvitsemasta ja hänen tarpeisiinsa juuri käsillä olevassa tilanteessa suunnitellusta opastuksesta, neuvoista, tuesta, avusta, yhteistyöverkostoista, palveluista, tutkimuksista, hoivasta ja hoidosta.” Osa tiedoista kuuluu salassa pidettäviin tietoihin, mutta asiakas voi halutessaan antaa sosiaali- ja terveydenhuollossa suostumuksensa siihen, että häntä koskevia tietoja voidaan luovuttaa niille tahoille, jotka tietoja tarvitsevat hoidon toteuttamiseksi. [PäH02]

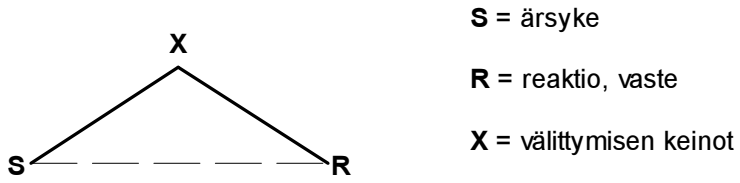
Kotihoidossa ei ole olemassa kaikkien toimijoiden yhteisessä käytössä olevaa ohjelmistoa, joka toimisi yhteistyövälineenä asiakkaan hoidossa. Kotisairaanhoito voi käyttää perusterveydenhuollon ohjelmiston tietoja, mikäli sellainen on. Useimmin yhteydenpitoväline eri organisaatioiden toimijoiden välillä on puhelin, mutta käytössä on myös erilaisia lomakkeita ja viestivihkoja. Kotihoidon tarvitsemia tietoja on tallennettuna myös esimerkiksi erikoissairaanhoidon järjestelmiin, joista niitä on haettava kotihoidon tarpeisiin. Osa kotihoidossa tuotetusta tiedosta on tarpeellista muille terveydenhoidon toimijoille. Nykyisellään järjestelmät eivät kommunikoi, vaan tiedon kulussa on katkoksia. [ToL04a]

5 Toiminnan teoriasta lähtevä vaatimusmäärittely

Menetelmät vaatimusten keräämiseksi ja kohdealueen kartoittamiseksi ovat usein vailla teoriapohjaa. Kohdealueelta etsitään suoraan asiakasvaatimuksia ja ohjelmistopalasia. Ohjelmiston on tuettava kohdealueella tehtävää työtä. Tällöin tuon työn ymmärtäminen on olennaisen tärkeää. Toiminnan teoria tarjoaa vankan, ihmisen toimintaa koskevan teoriapohjan ohjelmistosuunnittelulle, jonka lähtökohtana on kohdealueen työn ymmärtäminen. [KoM04]

5.1 Taustaa

Toiminnan teorian juuret ulottuvat 1800-luvulle. Sen filosofisena taustana on se, että ihmisyksilön ja lajin tajunta kehittyy käytännön toiminnassa ja yhteisön osana. Lev S. Vygotsky esitti 1920-luvulla, että ihmisen vaste ärsykkeisiin ei ole suora, vaan välittynyt (kuva 12). Ihminen voi kontrolloida toimintaansa ja käyttää apuna esimerkiksi merkkejä tai muita apuvälineitä [Vyg78].

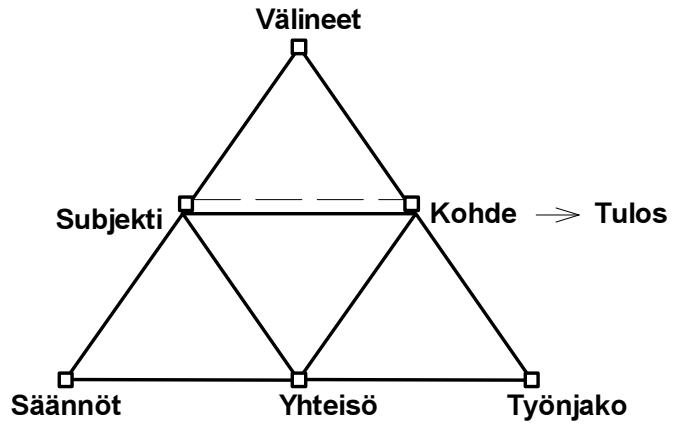


Kuva 12: Välittynyt toiminta

Vygotskyn malli koski yksilön toimintaa. 1970-luvulla Aleksei N. Leontiev lisäsi teoriaan yhteisön näkökulman. Hänen mukaansa yksilön teot voidaan ymmärtää vain sen toiminnan kautta, jonka osia ne ovat. Yksilö suorittaa tiedostamattomia *operaatioita*, joista koostuu yksittäinen tavoitteen ohjaama *teko*. Teko voi olla yksilön tai ryhmän suorittama. Teoista muodostuu *kollektiivinen toiminta*, jolla on kohde ja motiivi [Leo78].

Yrjö Engeström esitti 1980-luvulla kollektiivisen toiminnan rakennemallin, joka laajentaa yksilöllisen teon mallia yhteisöllä ja yhteisön välittyneillä suhteilla muihin toi-

minnan osiin (kuva 13). Useat kollektiiviset toiminnat muodostavat yhdessä *toimintojen verkon* [Eng87].



Kuva 13: Toiminnan rakenne systeemisenä kokonaisuutena

5.2 Toiminnan teoria ja ohjelmistosuunnittelu

Toiminnan teoriaa on käytetty apuna tietojärjestelmien suunnittelussa 1990-luvun alusta. Kuutti esitti ajatuksen, että tietojärjestelmien suunnittelussa analysoidaan kohdealueen tietojärjestelmän sijasta kohdealueen toimintajärjestelmää [Kuu91]. Bødger herätti keskustelun siitä, että toiminnan teorian mukaan tietokone ja sovellukset voidaan ajatella toiminnan välineiksi ja keinoiksi [Bød91].

1990- ja 2000-luvuilla toiminnan teoriaa on sovellettu erityisesti tutkittaessa ihmisen ja tietokoneen vuorovaikutusta (Human Computer Interaction, HCI) ja tietokoneavusteista yhteistyötä (Computer-Supported Cooperative Work, CSCW) sekä kehittävässä työntutkimuksessa (Developmental Work Research, DRW ja Activity Analysis and Development, ActAD) [KoM04].

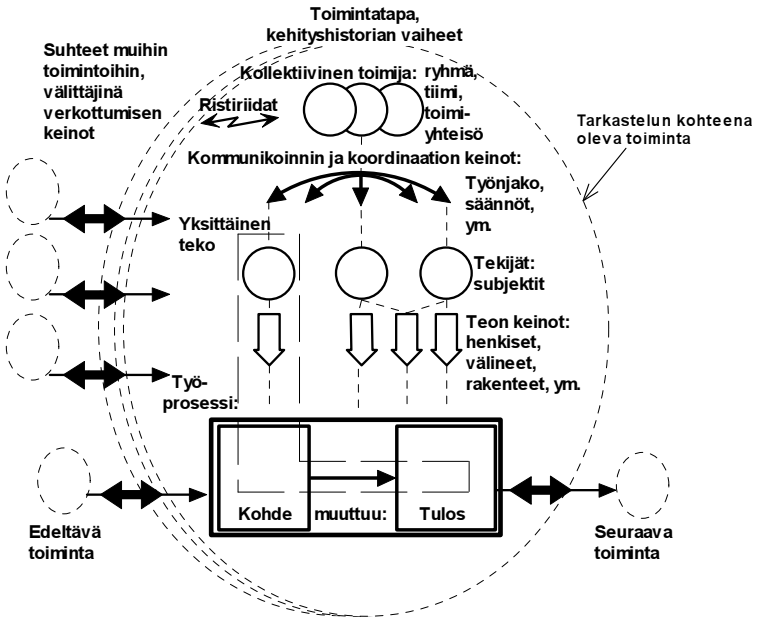
5.3 Työtoiminnan tutkiminen ja kehittäminen (ActAD)

Engeströmin mallia toiminnan rakenteesta ovat edelleen kehittäneet Mikko Korpela ja Anja Mursu [Kor94, Mur02]. Tuloksena on viitekehys *työtoiminnan tutkimiseen ja kehittämiseen* (Activity Analysis and Development, ActAD). ActAD on muunnelma Engeströmin aloittamasta kehittävästä työntutkimuksesta (Developmental Work Re-

searc, DRW). Ajatuksena on tietojärjestelmän ja työtoiminnan yhtäaikainen kehittäminen, jolloin yhdistetään tietotekniikan ja työtoiminnan ammattilaisten tietotaito työtoiminnan paremman sujuvuuden aikaansaamiseksi. Osallistuva suunnittelu on olennainen osa ActADia.

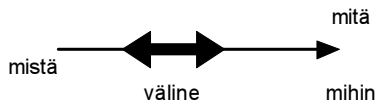
Työtoiminta tai *toimintajärjestelmä* ymmärretään kokonaisuutena, jossa joukko ihmisiä työskentelee jonkin yhteisen *kohteen* parissa *järjestäytyneellä* tavalla. *Tavoitteena* on tuottaa jokin yhteinen lopputulos. Kaikkien ihmisten toiminta ei välttämättä tapahdu samassa paikassa ja yhtä aikaa, eivätkä kaikki toimijat ehkä ole edes tietoisia toisistaan [KoM03]. Työtoiminnan elementit on esitetty kuvassa 14.

Työtoiminnan kehittäminen lähtee aina liikkeelle epätasapainosta tai ristiriidasta toiminnassa. Kun yksi toiminnan elementeistä muuttuu, aiheuttaa se muutoksentarvetta myös muissa. Usein tietotekniikan käyttöönotto on tällainen muutos [KoM03]. Työtoiminnan ja tietotekniikan yhtäaikaisella kehittämisellä pyritään saamaan aikaan tietojärjestelmiä, jotka todella tukevat kyseessä olevaa työtoimintaa. Joskus ratkaisu tietojärjestelmän ongelmaan on uusi tietokonesovellus, joskus työtavan muutos tai työn uudelleen organisointi [ToH04].



Kuva 14: Työtoiminnan elementit, ActAD-kehys [KoM03]

Toiminta (activity) on kollektiivinen, systeeminen kokonaisuus – pienempien tekojen (actions) joukko – jolla on kohde ja tavoite (kuva 14). Toiminta kokonaisuutena on kuvattu soikiona. Toiminnan ulkopuolella on toisia toimintoja, joilla on yhteyksiä tarkastelun kohteena olevaan toimintaan. Yhdessä nämä muodostavat *toimintojen verkon*. Yhteys voi olla toimintojen keskinäistä kommunikointia. Yhteys voi olla myös toisesta toiminnasta toiseen siirtyvää materiaalia, eli toisen toiminnan tulosta tarvitaan toisessa. Yhteyksissä toisiin toimintoihin tarvitaan *kommunikoinnin ja verkottumisen välineitä*. Suhteet kuvataan moniosaisilla nuolilla. Nuolen avulla voidaan kuvata, mistä tieto tulee, millä välineellä, mitä tietoa välittyy ja mihin (kuva 15).



Kuva 15: ActAD-mallin moniosainen nuoli

Toiminnalla on ajallinen perspektiivi suhteessa toisiin toimintoihin: toiminnalla on edeltäviä ja seuraavia toimintoja. Toiminnalla on myös oma historiallinen perspektiivi: ajan myötä toimintakokonaisuus voi muuttua esimerkiksi uusien välineiden tai sääntöjen vaikutuksesta. Tätä kuvataan päällekkäisillä soikioilla, joista tämänhetkinen toimintatapa tai tavoitetila on päällimmäisenä näkyvillä.

Toiminta on monen ihmisen erillisistä teoista koostuva yhteiseen päämäärään tähtäävä kokonaisuus. Eri toimijat eivät aina toimi samanaikaisesti, eivätkä välttämättä ole edes tietoisia toistensa toimista tai rooleista. Toiminnan sisällä kollektiivinen toimija on kuvattu ympyräjoukolla soikion yläosassa (kuva 14). *Kollektiivisella toimijalla* tarkoitetaan sitä ihmisjoukkoa, joka tarvitaan toiminnan tavoitteen saavuttamiseksi. Kollektiivinen toimija voi olla tarkoin määritelty tiimi, työyhteisö tai myös löyhemmin sidoksissa oleva ryhmä ihmisiä.

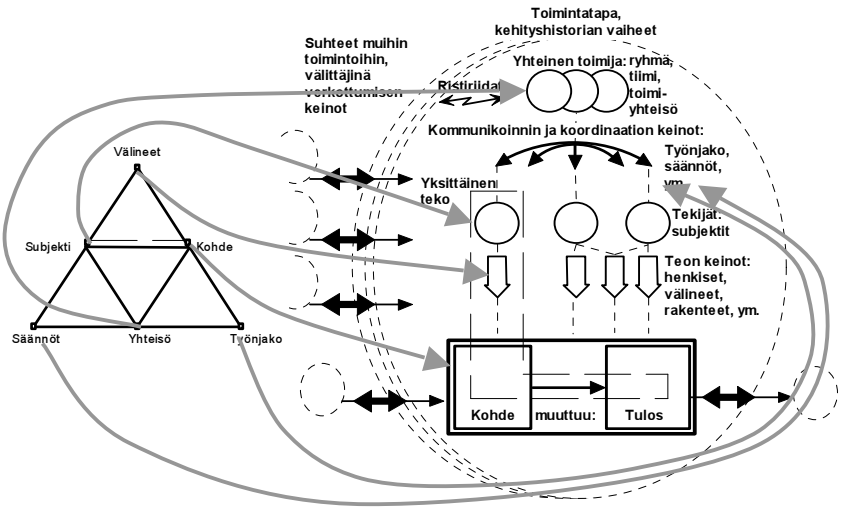
Eri toimijoiden teot eivät ole vain satunnainen joukko toisistaan riippumattomia toimia, vaan ne on koordinoitava. On kyseessä toimintakokonaisuus. Jotta yhteinen tavoite saavutettaisiin, on toimijoiden kommunikoitava keskenään. *Kommunikaation ja koordinaation keinoja* ovat esimerkiksi kokous, puhelin, tietokonesovellus, työvuoro-

lista ja säännöt. Kommunikaation ja koordinoinnin välineet on kuvattu kollektiivisen toimijan alapuolelle kaarevilla nuolilla (kuva 14).

Toiminnasta voidaan havaita yksittäisten toimijoiden suorittamia *tekoja*. Yksittäiset *toimijat* on kuvattu ympyröillä soikion keskivaiheilla (kuva 14). Yksittäiset toimijat tarvitsevat tekojen suorittamiseen *työvälineitä ja -keinoja*, jotka voivat olla materiaalisia, esimerkiksi koneita ja laitteita, tai ei-materiaalisia, kuten ammattitaitoa ja työkokemusta. Nämä on kuvattu nuolilla, jotka osoittavat tekijöistä työprosessiin kuvassa 14.

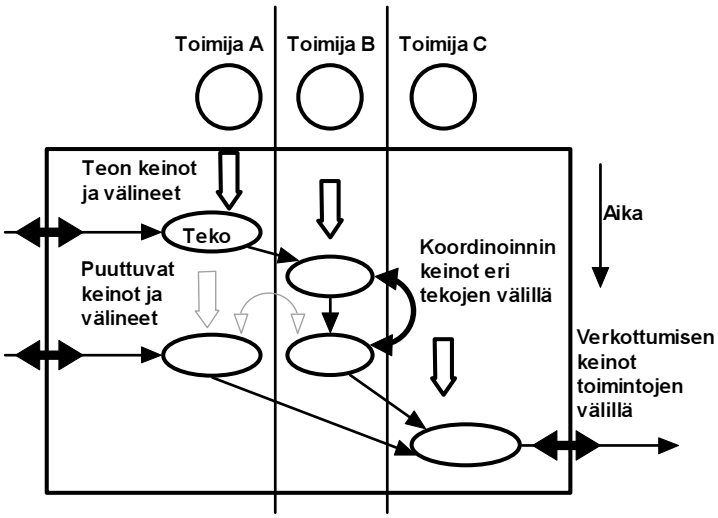
Toiminnalla on aina *kohde* ja *tavoite*. Toiminnan *prosessin* tuloksena kohteessa tapahtuu muutos tavoitteen suuntaan. Toiminnan kohde, tulos ja työprosessi kuvataan soikion alalaidassa suorakaiteilla (kuva 14).

ActAD-soikion ja Engeströmin kolmiomallin yhteys on esitetty kuvassa 16. Sekä kolmio- että soikiomalli sisältävät pääosin samat peruselementit, mutta soikiomallissa korostetaan elementtien erilaisuutta kuvaamalla ne eri symboleilla. Soikiossa erotellaan myös yksilö ja kollektiivinen toimija sekä näiden työvälineet ja keinot. Näin toiminnan luonne monien toimijoiden yhteistyönä tulee selvemmin esille. Mallissa tulevat esille myös toiminnan suhteet toisiin toimintoihin. Kuten yksilöidenkin väliset suhteet, myös toimintojen väliset suhteet ovat välittyneitä ja tapahtuvat verkottumisen keinoin. Lisäksi soikiomalli antaa mahdollisuuden käyttää symboleina havainnollisia kuvia eri elementeistä, kuten puhelin, tietokone, lääke tai ihminen. Tämä helpottaa kuvaustavan ymmärtämistä, jolloin soikiomallia voidaan käyttää ajatusten herättäjänä ja keskustelun apuna eri alojen ammattilaisten välillä [KoS00].



Kuva 16: Yhteys kolmio- ja soikiomallin välillä [KoM03]

Kun toiminnan kokonaiskuva on saavutettu, voidaan analysointia jatkaa tarkemmalle tasolle prosessikuvauksilla. Työprosessin kuvauksessa on mahdollista käyttää UML:n aktiviteettikaaviota muistuttavaa kuvaustapaa (kuva 17), josta saadaan muodostettua myös käyttötapauksia. Kun joissakin kehityskohteissa tietotekniikasta katsotaan olevan hyötyä ja jokin teko halutaan tehdä ohjelmiston avulla, käyttötapauskaavioista on helppo edetä kohti ohjelmistovaatimuksia.

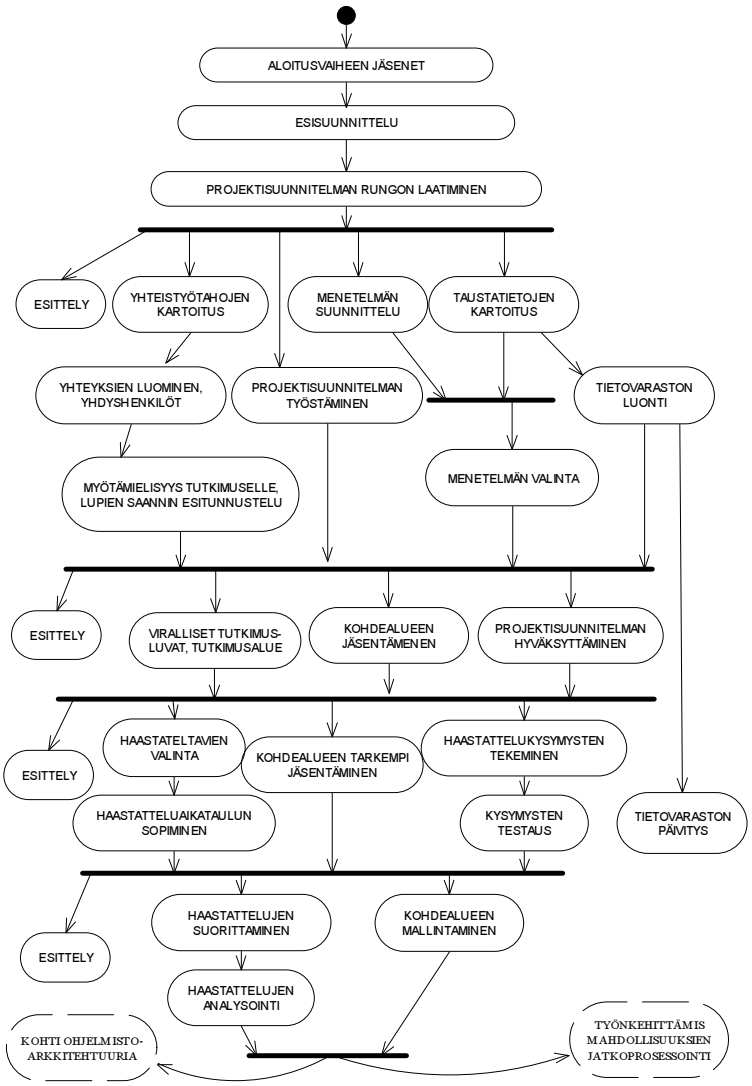


Kuva 17: Työprosessikuvaus [KoM04]

5.4 ActAD-mallin soveltamisen käytännön vaiheita

Kun aletaan kartoittaa kohdealuetta, on päätettävä, mitä tietoa tarvitaan, mistä tietoa saadaan ja millä menetelmillä. Kotihoito-projektin alkupään työvaiheita olivat taustatietojen etsiminen kotihoidosta, yhteistyötahojen kartoitus, yhteyksien luominen, haastateltavien valinta ja haastatteluaikataulujen sopiminen sekä haastattelukysymysten tekeminen ja testaaminen. Näitä on kuvattu aktiviteettikaavion avulla kuvassa 18.

Projektin käytännön työvaiheisiin kuului myös *projektinhallinta*: projektisuunnitelman laatiminen, hyväksyttäminen ja ylläpito. Projektin aikana kerättiin *tietovarasto*, johon koottiin tietoa kotihoitoa sivuavista muista projekteista, julkaisuista ja tietojärjestelmistä. Kun projektin kannalta kiinnostavaa tietoa löytyi, tietolähde lisättiin tietovarastoon, josta se oli helposti kaikkien tietoa tarvitsevien käytössä. Tietovarasto toteutettiin Internet-sivulle luotuna linkkilistana, jossa oli linkin lisäksi lyhyt kuvaus kohteesta. Tietovaraston ylläpito on tärkeää, sillä on hyödyllistä olla selvillä siitä, mitä muut ovat samasta aiheesta tehneet. Kotihoito-projekti oli *tutkimusprojekti*, joten siihen kuului myös tarvittavien tutkimuslupien hankinta, menetelmän kehittäminen ja tulosten esittely.



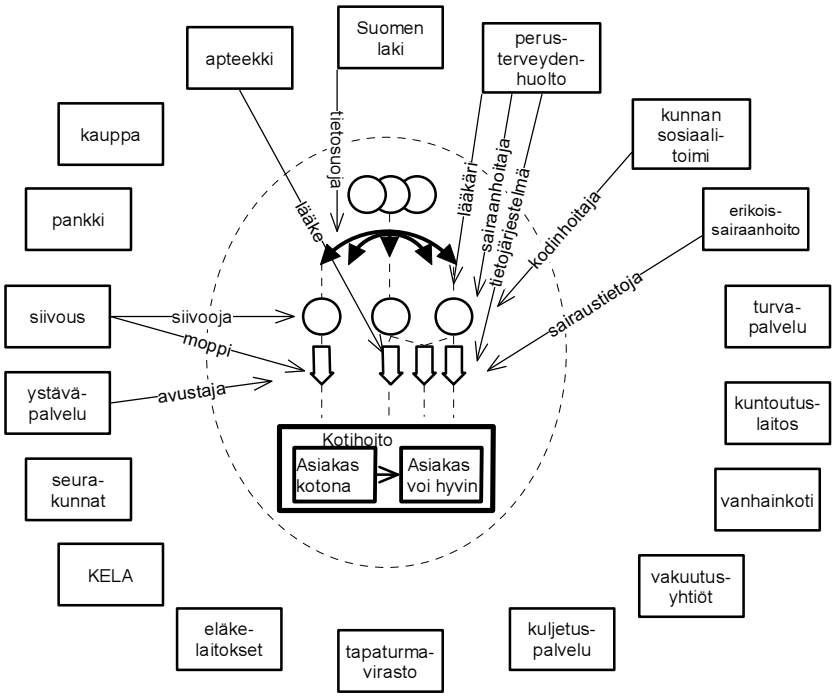
Kuva 18: Kotihoito-projektin työvaiheita aktiviteettikaaviona

5.4.1 Esitutkimus

Jotta ohjelmistotuotantoprosessiin kuuluva vaatimusmäärittely voidaan tehdä, tulee toimialasta olla ensin selkeä kuva. On tunnettava toimijoiden työ, eli mitä tehdään ja kuka tekee sekä työssä tarvittavat tiedot ja tietokokonaisuudet. Kirjallisuuskatsaus ja tietovaraston kerääminen antavat alkutietoja kohdealueesta. Alkutietoja jäsennetään ActADin toiminnan rakennemallin avulla. *Jäsentämisellä* tarkoitetaan toiminnan rakenneosien tunnistamista ja nimeämistä kohdealueelta.

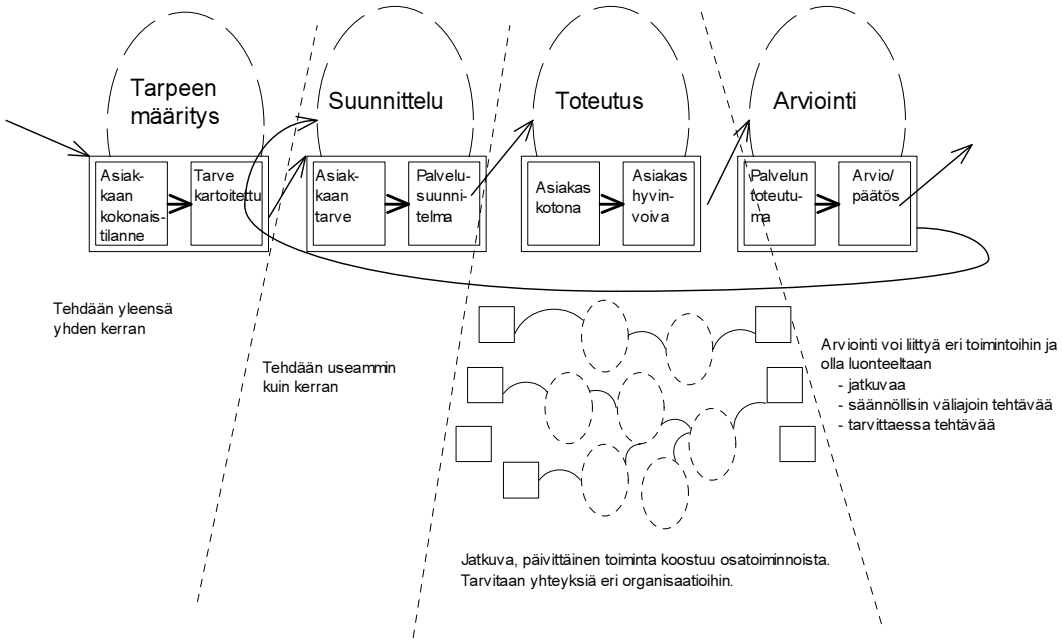
Aluksi tunnistetaan ja määritellään kohdealueen keskeinen *toiminta*, tässä tapauksessa kotona tapahtuva hoiva. Toiminnalla on aina *kohde* ja *tavoite*. Kotihoidon kohteena

on asiakas ja tavoitteena asiakkaan kotona asumisen mahdollisuus. *Toimijoina* on joukko kotihoitoon osallistuvia henkilöitä, esimerkiksi kotisairaanhoitaja ja kodinhoitaja. Kotihoitoa ympäröi suuri joukko erilaisia *organisaatioita*, joista kotihoitoon voi tulla toimijoita, tietoa tai välineitä (kuva 19). Mallin avulla kohdealueelta löydetään erilaisia toimijoita ja organisaatioita, joista he tulevat. Tämä helpottaa löytämään ja valitsemaan konkreettisia haastateltavia, joilta voidaan saada lisää tietoa.



Kuva 19: Osa kotihoitoa ympäröivistä organisaatioista, joista kotihoitoon voi tulla toimijoita tai välineitä (Kotihoito-projekti, PlugIT)

Kotihoito prosessina on liian suuri käsiteltäväksi yhtenä toimintakokonaisuutena ja siksi kotihoito jaetaan pienempiin palasiin. Prosessi yleensäkin voidaan jakaa peräkkäisiin vaiheisiin ja toiminta osatoiminnoiksi. Kotihoitoprosessi jaetaan neljään peräkkäiseen vaiheeseen: *tarvekartoitus, suunnittelu, toteutus* ja *arviointi*. Suunnitteluvaihe erotetaan omaksi osatoiminnakseen, koska se sisältää paljon tiedon hankkimista, käsittelyä ja tallentamista. Kotihoidon suunnitelma on perusta toteutusvaiheen toiminnoille. Toteutusvaihe on monien osatoimintojen summa. Kotihoidon toteuttamisen osatoiminnoiksi tunnistetaan esimerkiksi *kotisairaanhoito, kotipalvelu, turvapalvelu* ja *siivouspalvelu*. Osatoiminnot tarvitsevat sekä keskinäisiä yhteyksiä, että yhteyksiä ympäröiviin organisaatioihin, jotta asiakas saisi parhaan mahdollisen tarvettaan vastaavan palvelun. Kotihoidon vaiheita ja osatoimintoja on esitetty kuvassa 20.



Kuva 20: Korkean abstraktiotason kuva kotihoidosta kokonaisuutena, mukailtu [ToE03]

Kohdealueelta löydetyistä osatoiminnoista valitaan tarkastelun kohteeksi keskeisimmät. Näistä toiminnoista valitaan potentiaaliset haastateltavat. Tässä esimerkissä tarkastellaan lähemmin kotisairaanhoitoa. Esitietojen perusteella tiedetään, että kotisairaanhoitoon osallistuvia ammattilaisia ovat ainakin sairaanhoitaja, lähihoitaja ja lääkäri, ja että organisaatio, josta he tulevat on kaupungin terveystoimi. Heidän työtehtävänsä ovat keskenään erilaisia, joten on perusteltua valita haastateltavaksi yksi kunkin ammatin edustaja. Heidän kanssaan sovitaan haastattelusta henkilökohtaisesti. Kohdealueesta saadaan esille monipuolinen kokonaiskuva, kun kukin toimija kuvaa kohdealuetta omasta näkökulmastaan. Toisaalta saadaan esille myös eri syvyystasolla olevia näkökulmia: kunkin osatoiminnan yhteinen näkökulma ja toisaalta erilaisten toimijoiden näkökulmat kunkin osatoiminnan sisällä.

Kaikista osatoiminnoista valitaan keskeisimmät toimijat ja hankitaan haastateltavat. Tarvitaan myös kotihoidon hallinnon ja johdon sekä kohdealueella toimivan tietotekniikan ammattilaisen näkökulmaa. Myös kotihoidon asiakasta voidaan haastatella, vaikka asiakkaan haastattelu ei tässä tapauksessa ole avainasemassa. Kotihoidon asiakkaina on hyvin monenlaisia ihmisiä, joilla jokaisella on yksilöllinen palvelujen yh-

distelmä. Kotihoidon kohteena asiakkaalla ei ehkä ole sellaista kotihoidon tietojärjestelmän suunnitteluun tarvittavaa tietoa, jota kotihoidon toimijoilta ei voi saada.

5.4.2 Tiedon kerääminen ja kuvaaminen

Tietoa kohdealueelta saadaan haastattelemalla erilaisia toimijoita. Haastattelu tehdään teemahaastatteluna, jonka teemat saadaan toiminnan teoriasta. *Teemahaastattelu* on puolistrukturoitu haastattelumenetelmä, jonka avulla saadaan aikaan melko vapaamuotoinen ja luonteva haastattelutilanne, joka etenee kuitenkin valittujen aiheiden ympärillä. Teemahaastattelun käyttäminen edellyttää jonkin asteista kohdealueen tuntemusta, jotta haastatteluteemat ja -kysymykset voidaan suunnitella [HiH88, s. 8]. Haastatteluteemat ovat: työ toimintana, kollektiivinen toimija, tieto työn välineenä ja yhteistyön ja kommunikoinnin välineet. Teemoista laaditaan kysymyksiä, ja teoriatason abstraktit käsitteet korvataan kohdealueen ammattilaisille tutuilla termeillä. Kysymykset voivat olla kaikille toimijoille samat.

Työ toimintana -teemasta saadaan esimerkiksi seuraavia kysymyksiä (Kotihoitoprojekti, PlugIT):

- Ketkä ovat kotihoidon asiakkaita?
- Mitä palveluja kotihoidossa tuotetaan?
- Mistä ja miten saadaan tarvittavat resurssit?
- Miten asiakkaat tulevat kotihoidon palvelujen piiriin? Onko eroja siinä, miten eri asiakasryhmät tulevat kotihoidon asiakkaiksi?
- Mitä tietoa saadaan ensimmäisessä yhteydenotossa? Onko tämä tieto riittävää palvelun käynnistämiseksi? Mitä muuta tietoa tarvitaan ja mistä sitä saadaan?
- Määritelläänkö asiakkaan kotihoidolle tavoite? Miten ja miksi se määritellään? Ketkä määrittelevät tavoitteen ja missä yhteydessä? Mitä tietoja tarvitaan tavoitteen määrittelyssä ja miten tarvittavat tiedot saadaan? Kenelle määritelty tavoite ilmaistaan ja miten?

Näiden kysymysten avulla löydetään toiminnan kohde, tavoite, osatoimintoja ja prosesseja sekä toimintaa edeltäviä ja seuraavia toimintoja.

Tieto työn välineenä -teema selvittää, mitä tietoja kotihoidon toteuttamiseksi tarvitaan, mistä tarvittavat tiedot saadaan ja missä muodossa. Näin löydetään toiminnassa tarvittavia tietokokonaisuuksia, joita voidaan myöhemmin täsmentää.

Kollektiivinen toimija -teeman avulla selvitetään, ketkä kaikki toimivat kotihoidossa, mistä organisaatioista he tulevat ja millaisia yhteistyötahoja kotihoidolla on. Näin voidaan löytää toimintojen verkko ja yhteyksiä ympäröiviin organisaatioihin. Yhteistyöverkon avulla voidaan hahmottaa, mitkä eri osatoiminnat ovat paljon yhteistyössä, mitkä vähemmän.

Yhteistyön ja kommunikaation välineet -teema sisältää kysymyksiä työnjaosta, säännöistä, toimintatavoista sekä konkreettisista välineistä, joita toiminnassa tarvittavan tiedon välityksessä käytetään. Näiden kysymysten avulla löydetään niin tiedonkulun välineet ja rajoitteet, kuin myös ongelmakohdat, joissa tieto ei kulje niin kuin pitäisi.

Toiminnan teoria ohjaa siis kysymysten laadintaa. Kysymyksiä laaditaan *toiminnan rakenneosien* (tulos, kohde ja prosessi, välineet, tekijät sekä sosiaaliset suhteet ja välineet) sekä *toimintaverkon muodostumisen ja rakenteen* etsimiseen [Mur02]. Kysymykset ryhmitellään siten, että haastattelu voi edetä loogisesti ja sujuvasti. Haastattelija voi lisätä kysymyksen yhteyteen haastattelutilannetta varten täsmennyksiä siitä, mitä kysymyksellä tarkoitetaan. Liiallista haastateltavan johdattelua on kuitenkin vältettävä.

Haastattelutulosten kannalta on edullista, jos haastattelut voidaan tehdä haastateltavan omassa työympäristössä. Tällöin haastateltava ei jännitä haastattelutilannetta, eikä hänen tarvitse nähdä ylimääräistä vaivaa tullakseen haastatteluun. Haastattelijoita on hyvä olla useampi. Yksi haastattelija pääsääntöisesti kyselee ja muut tekevät muistiinpanoja. Haastattelutilanne on onnistuessaan haastattelijan ja haastateltavan välillä käytävä hyvin informatiivinen keskustelu, jossa on vapautunut tunnelma. Haastateltava ei aina vastaa pelkästään esitettyihin kysymyksiin, vaan hän voi kertoa lisäksi täsmentäviä esimerkkejä tai vastata myöhemmin tuleviin kysymyksiin.

Muistiinpanoihin kirjataan mahdollisimman tarkkaan haastateltavan vastaukset, mutta ei pyritä kuitenkaan sanatarkkaan kirjaukseen. Kirjaajalla pitää olla etukäteen käsitys siitä, mitä kysymyksillä pyritään selvittämään pystyäkseen kirjaamaan olennaisen. Haastattelu kannattaa myös nauhoittaa siltä varalta, että joku asia jää kuulematta, tai kaikkea ei ehditä kirjoittaa haastattelutilanteessa ylös.

Haastattelijat kirjoittavat haastattelun puhtaaksi ryhmätyönä mahdollisimman pian haastattelun jälkeen. Tällöin varmistetaan se, etteivät asiat unohdu. Keskustelu ryhmän kanssa auttaa muistamaan ja tulkitsemaan saadut vastaukset oikein.

5.4.3 Kerättyjen tietojen analysointi

Kun kaikki haastattelut on tehty, analysoidaan tulokset. Analyysissä tieto pilkotaan osiin ja järjestellään uudelleen. Analysoinnin alkuvaiheessa on hyödyllistä piirtää kaavioita ja kuvia kerätyn aineiston perusteella. Mallinnettavia asioita ovat esimerkiksi toiminnassa havaitut erilaiset tietosisällöt, tietojen sijainti ja siirtyminen sekä erilaiset prosessit. Käyttökelpoisia kaavioita ovat esimerkiksi aktiviteettikaaviot, joiden avulla voidaan kuvata toimintaprosesseja, tai taulukot, joihin voidaan kerätä erityyppisiä tietoja. Kuvissa on havainnollista käyttää erilaisia symboleita. Esimerkkinä on liitteessä 1 kuvio tietojen sijainnista ja kulusta kotisairaanhoidossa (kuva 1), kotipalvelun tiimitilassa sijaitsevien kansioiden ja muiden tietolähteiden sisältöjä (kuva 2), aktiviteettikaavio asiakkaan hoidosta (kuva 3) sekä puhelinkartta, joka kuvaa kotisairaanhoitajan työssään tarvitsemia puhelinyhteyksiä (kuva 4). Kaaviot ja kuvat auttavat hahmottamaan kohdealuetta ja sen toimintoja.

Myös toimintatarinoita voi käyttää lisäämään ymmärrystä kohdealueesta. *Toimintatarina* on kuvitteellinen, kohdealueen toimintaa kuvaava tarina. Tarinan laatiminen tuo esille kohtia, joista täytyy hankkia lisää tietoa. Esimerkki kotihoidon toimintatarinasta on liitteessä 2. Toimintatarina tuo kohdealueen toiminnan konkreettisemmaksi ja siten helpommin ymmärrettäväksi. Toimintatarinan avulla voidaan miettiä erilaisia tiedonkulkuja ja vastuuhenkilöitä. Samalla voi löytää solmukohtia, joissa tieto ei kulje niin kuin pitäisi. Tieto voi tulla myöhässä tai hävitä kokonaan, olla puutteellista, vanhentunutta, ristiriitaista tai virheellistä. Myöhemmin toimintatarinaa voidaan käyttää myös keskustelun herättäjänä, kun tarkastetaan analysoinnin tuloksia kohdealueen toimijoiden kanssa.

Analysointivaiheessa ActAD-malli ohjaa kerätyn toimintoja koskevan tiedon jäsentämistä kahdella tasolla. Ensinnäkin etsitään ja mallinnetaan kunkin *toiminnan sisäinen rakenne*. Toiseksi tunnistetaan eri *toiminnoista muodostuva verkko yhteyksineen*. Lisäksi analysoidaan toiminnoissa tarvittavat *tiedot*.

Yksittäisen toiminnan sisäisen rakenteen selvittämiseksi haastatteluista kerätyt tiedot jäsennellään toiminnan teorian mukaisiin palasiin. Saman osatoiminnan kaikki haastattelut kootaan yhteen, esimerkiksi kotisairaanhoidon kohdalla sairaanhoitajan, lähihoitajan ja lääkärin haastattelut. Tiedot ryhmitellään seuraavien toiminnan rakennemallista saatavien otsikoiden alle:

- *Edeltävät tai vaikuttavat toiminnat.* Tässä kuvataan ne toiminnat, jotka suoraan edeltävät käsiteltävää toimintaa, tai vaikuttavat siihen. Esimerkiksi toimintaa kotisairaanhoito edeltää tilaus, esitietojen selvitys ja suunnittelu. Kotisairaanhoitoon vaikuttavia toimintoja ovat lainsäädäntö, työnjako sekä sosiaali- ja terveyslautakunnan päätökset.

- *Kollektiivinen toimija* on toiminnan sisällä vaikuttavista toimijoista muodostuva ihmisryhmä, joka tarvitaan toiminnan toteuttamiseen. Joskus voidaan löytää useita kollektiivisia toimijoita. Esimerkiksi kotisairaanhoidossa kollektiivisena toimijana on sairaanhoitaja-lääkäri -pari sekä alueelliseen työnjakoon perustuva kotisairaanhoidon tiimi.

- *Yhteistyö.* Toiminnan yhteistyön, kommunikaation ja koordinoinnin välineet ja keinot voidaan jakaa toiminnan sisäisiin ja eri toimintojen välisiin. Kotisairaanhoidon sisäisiä yhteistyön välineitä ovat esimerkiksi terveydenhuollon tietojärjestelmä, sähköposti, palaverit, kaupungin intranet sekä erilaiset viestilaput. Yhteistyövälineinä kotipalveluun päin toimivat esimerkiksi puhelin, keskustelut, asiakkaan luona olevat Hoito- ja palvelusuunnitelma ja viestivihko sekä yhteiskäynnit asiakkaan luona. Kotisairaanhoitaja toimii yleensä välittäjänä kotipalvelun työntekijän ja lääkärin välisessä viestinnässä. Toimintaa ohjaa sovitut yhteistyö käytännöt, säännöt sekä työnjako.

- *Toimijat*. Kotisairaanhoidon toimijoita ovat lääkäri, terveydenhoitaja, sairaanhoitaja, perushoitaja, lähihoitaja, osastonhoitaja sekä asiakas itse.

- *Välineet.* Varsinaisessa kotisairaanhoitotyössä tarvittavat työvälineet voivat olla materiaalisia tai ei-materiaalisia. Kotisairaanhoidossa työvälineinä ovat esimerkiksi terveydenhuollon tietojärjestelmä, paperiset sairauskertomukset, lääkärin sanelut, epikriisit, reseptit ja muut lääkitystiedot, puhelin, asiakkaan havainnointi, sairaanhoitolaitteet ja -välineet, koulutus, työkokemus, erikoisosaaminen sekä sähköiset ja kirjalliset tietolähteet.

- *Kohde.* Kotihoidon kohteena on asiakas ja asiakkaan hyvinvointi. Kotisairaanhoidossa kohde tarkentuu asiakkaan *sairaanhoidollisiin tarpeisiin*, kuten haavahoito, pistokset, erilaisten laboratorionäytteiden otto sekä hoitotarvikejakelu. asiakkaan terveydentila, sen tarkkailu ja arviointi sekä jatkohoidon suunnittelu ovat myös kohteena, samoin asiakkaan ohjaus ja neuvonta esimerkiksi ruokavalion suhteen, tiedottaminen sekä etujen ja tarpeiden kartoitus esimerkiksi hoitotukiasioissa.

- *Tavoite.* Kotisairaanhoidossa on tavoitteena, että asiakkaan akuutit sairaudet tai vammat paranevat ja krooniset sairaudet pysyisivät hallinnassa. Tämä on osa asiakkaan toimintakyvyn hyvänä pysymisen tavoitetta, joka taas on edellytys sille, että asiakas voi asua kotonaan mahdollisimman pitkään. Viime kädessä tavoitteena on asiakkaan kokonaisvaltainen hyvinvointi.

- *Tuotos.* Kotisairaanhoidosta muissa toiminnoissa tarvittavia tuotoksia ovat kotikäynnin tietojen kirjaaminen terveydenhuollon järjestelmään sekä asiakkaan kotona olevaan viestivihkoon.

- *Seuraavat toiminnat.* Seuraavia toimintoja voivat olla esimerkiksi asiakkaan tilan seuranta, käyntisuunnitelman päivittäminen tarvittaessa, tilastointi ja laskutus.

Yksittäisten toimintojen sisäisen rakenteen lisäksi mallinnetaan toimintojen verkko. Tätä varten tunnistetaan haastattelutiedoista yhteyksiä toimijoiden ja toimintojen välillä. Selvitetään, ketkä kommunikoivat keskenään, miksi ja milloin. Toiminnassa tarvittavien tietojen sisältö selvitetään. Myös tiedon siirtyminen toimijoiden välisessä viestinnässä tutkitaan ja tunnistetaan, mitä välineitä viestinnässä käytetään. Toiminnalla on yhteyksiä myös ympäröiviin organisaatioihin. On tärkeää tunnistaa kriittiset pisteet, joissa tiedon kululla on hyvin tärkeä merkitys, ja kehityskohteet, joissa tieto ei kulje niin hyvin kuin pitäisi tai voisi.

Kotisairaanhoidolla on yhteyksiä moniin kotihoidon osatoimintoihin, eniten kotipalvelun työntekijöihin. Kuopiossa pääasiallisin viestintäväline kotipalveluun päin on matkapuhelin ja useimmiten kyseessä on asiakkaan vointiin liittyvät tiedot, joita vaihdetaan. Myös Hoito- ja palvelusuunnitelma on tärkeä viestinnän väline, sillä siihen on kirjattu asiakkaan kotihoitoon vaikuttavat tiedot mahdollisimman kattavasti. Kotisairaanhoidolla on myös paljon yhteyksiä ympäröiviin sairaanhoidon organisaatioihin. Kaupungin perusterveydenhuollon järjestelmä on tietolähteenä tärkeä, sillä kotisairaanhoito kuuluu samaan organisaatioon, ja voi siten tutkia järjestelmään tallennettuja asiakkaan terveys- ja sairaustietoja. Erikoissairaanhoidon puolelle viestintävälineenä ovat epikriisit, hoitajan ja lääkärinlähetteet, sekä kiireellisissä asioissa, esimerkiksi kotiutustilanteessa usein puhelin.

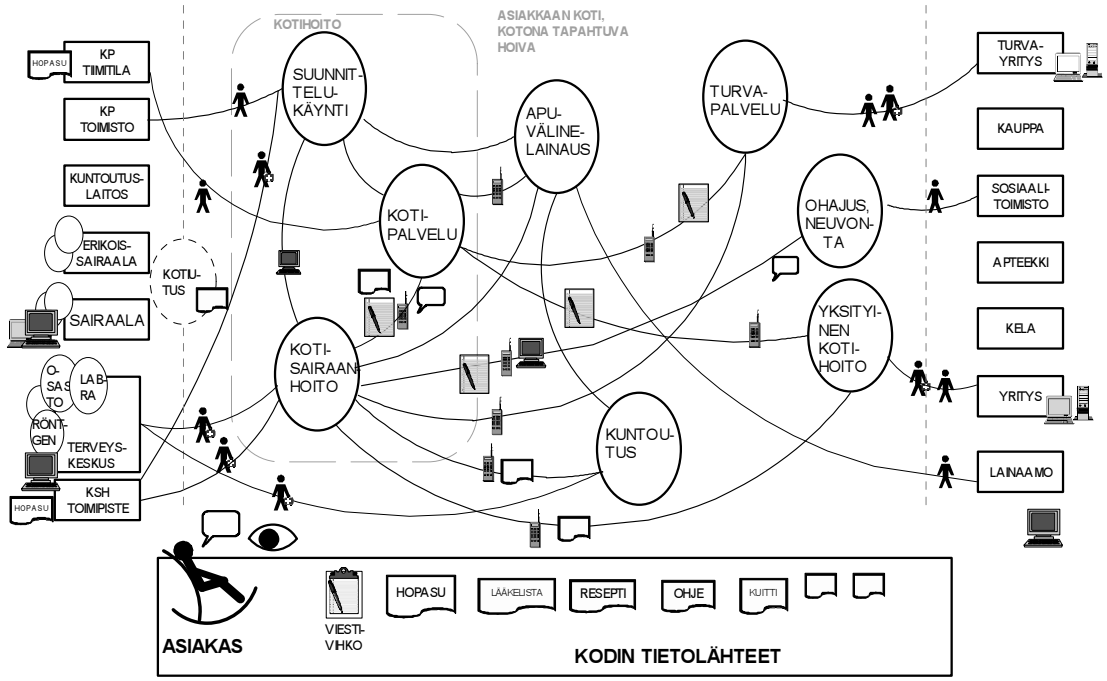
Toimintoihin liittyvien tietojen analysoimiseksi haastatteluaineistosta etsitään keskeisiä pienemmistä tietoyksiköistä koostuvia *tietokokonaisuuksia*, jotka kuvataan. Kotihoidossa keskeisiksi tietokokonaisuuksiksi nousivat haastatteluista hoito- ja palvelusuunnitelma, hoito- ja palvelusuunnitelmakansio, lääkelista, kotiutustiedot sekä asiakkaan kotona oleva viestivihko [ToL04a].

5.4.4 Tulosten tarkistuttaminen toimialan asiantuntijoilla

Haastatteluissa kerätty tieto kootaan yhteen dokumenttiin, jossa kuvataan kohdealueen toiminnat ja tietokokonaisuudet, toimintojen väliset yhteydet, tietojen sijainti ja

saatavuus sekä havaitut kehityskohteet. Toimintojen dokumentoinnissa käytetään hyväksi edellä kuvattua ActAD-mallin mukaista toiminnan rakennetta [ToL04a].

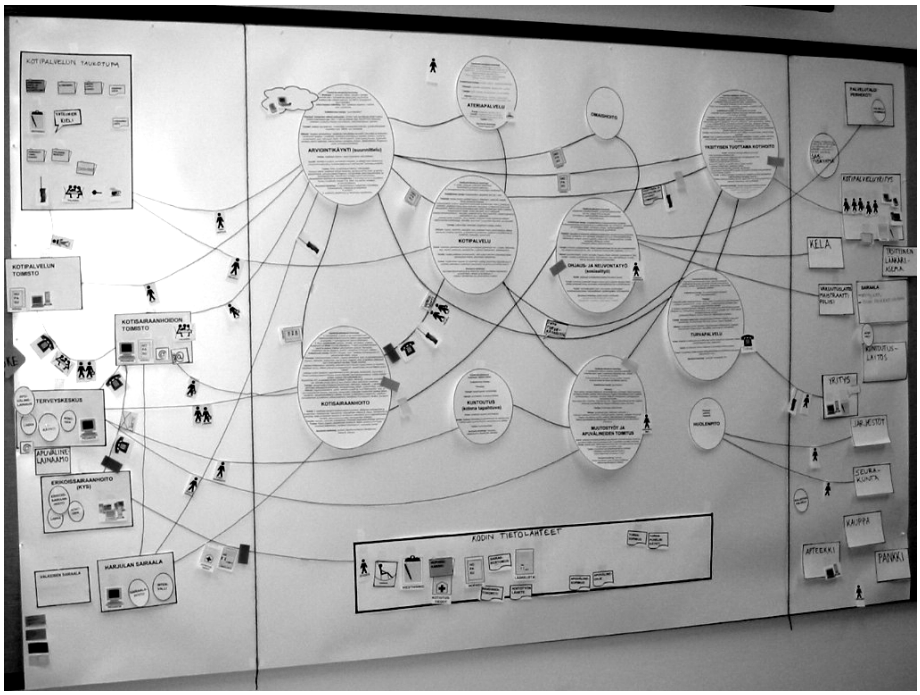
Jotta voidaan varmistaa, että haastatteluilla kerätyt tiedot on ymmärretty oikein ja että niistä ei puutu mitään olennaista, tiedot tarkistetaan yhdessä kohdealueen toimijoiden kanssa. Tarkistus tehdään työpajassa, johon kutsutaan osa haastatelluista, mielellään yksi jokaisesta osatoiminnasta. Kutsutuille toimijoille lähetetään pelkistetty kuva toimijoiden verkosta, jotta he voivat tutustua siihen etukäteen. Keskustelun pohjana työpajassa käytetään seinätekniikalla esitettyä toimintojen verkkoa (kuva 21). *Seinätekniikassa* huoneen seinälle asetettuun isoon pohjapaperiin kootaan irrallisista symboleista halutun mallin mukaisia ryhmittymiä. Symbolit kiinnitetään esimerkiksi teipillä tai tarralla, jolloin niitä voidaan helposti siirtää ja ryhmitellä tarvittaessa uudelleen [Saa87].



Kuva 21: Seinätekniikalla esitetty toimintojen verkko

Seinätauluelementit piirretään ja tulostetaan valmiiksi ennen työpajaa. Toiminnat kuvataan soikioilla, joiden sisään kirjoitetaan toiminnan rakenne. Yhteydet toimintojen välillä merkitään toimintoja yhdistävillä langoilla. Tietovälineiden symboleina käytetään havainnollisia kuvia, kuten puhelin ja tietokone. Kodin rajat merkitään, ja kodin rajojen sisäpuolelle sijoitetaan toiminnat sekä kotona olevat tietolähteet. Kodin ulko-

puolelle sijoitetaan suorakaiteina kuvatut organisaatiot tai rakennukset, joista toimijat tulevat, esimerkiksi kotipalvelun tiimitila, sekä ne ympäröivät yhteisöt, joihin kotihoidon toimijoilla on yhteyksiä, kuten erikoissairaala. Seinätaululle kootaan valmiiksi mahdollisimman paljon haastatteluista saatua tietoa. Kuvassa 22 on Kotihoito-projektin seinätaulu työpajan alkuvaiheessa.



Kuva 22: Seinätaulu, Kotihoito-projektin työpaja 11.12.2003, kuva: Päivi Röppänen

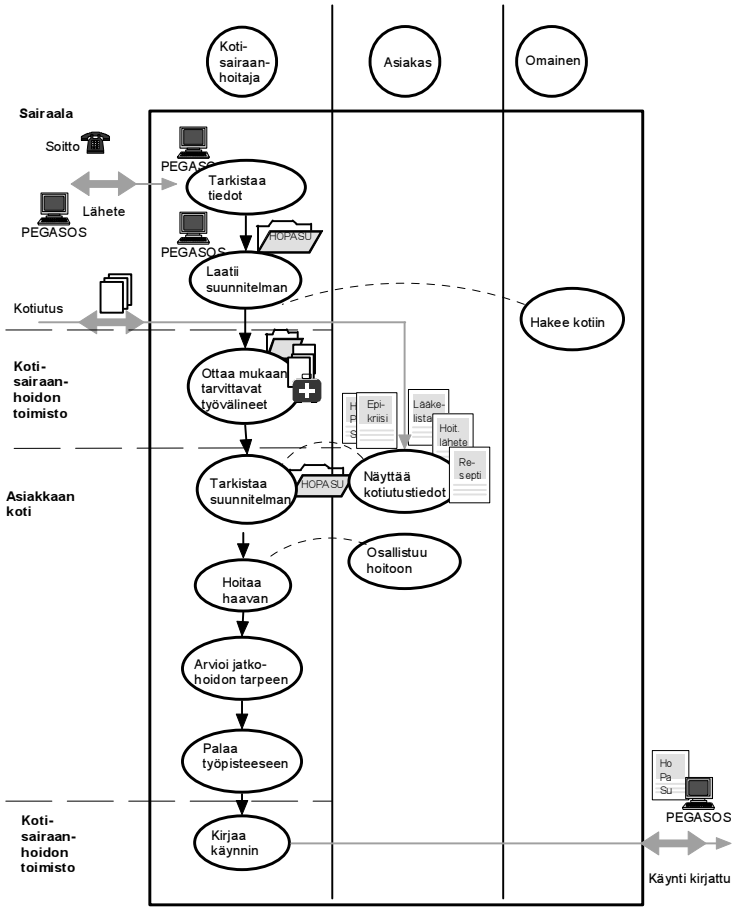
Työpajassa esitellään käytetty kuvaustekniikka ja symbolit ensin yleisellä tasolla. Sen jälkeen kukin toiminta esitellään tarkemmin, jolloin syntyy keskustelua toimijoiden kesken. Samalla seinätauluun korjataan virheet ja lisätään puuttuvat osat. Seinätaulu on muistiinpanoväline, ja se tulee säilyttää muistin tukena. Työpajan keskusteluista kirjataan ylös olennaiset asiat, erityisesti uudet asiat ja ne, jotka tarvitsee muuttaa. Käytännössä tämä onnistuu parhaiten, jos toimintojen esittelyt suunnitellaan ja jaetaan etukäteen tutkimusryhmän jäsenille ja muistiinpanoja tekemässä on useampi kuin yksi henkilö. Keskustelut voidaan myös nauhoittaa tai videoida.

Työpajan jälkeen tehdään tarvittavat muutokset toimintojen ja tietojen kuvaukseen. Korjattu dokumentti lähetetään työpajaan osallistuneille, jotka tarkistavat muutokset. Tässä vaiheessa on saatu selvitettyä kohdealueen yksittäiset toiminnat ja toimintojen verkko sekä tarvittavat tietoyksiköt ja niiden liikkuminen toimintojen ja organisaati-

oiden välillä. Näitä tuloksia voidaan käyttää hyväksi sekä työtoiminnan kehittämisessä organisaatio- tai yhteisötasolla että tietojärjestelmän kehittämisessä.

5.4.5 Analyysin tarkentaminen

Tietojärjestelmän kehittämistä varten edetään toimintojen ja tietojen kuvauksessa tarkemmalle tasolle. Keskeisistä toiminnoista kuvataan työprosessit, jotka koostuvat teoista käyttäen apuna aktiviteettikaavioita muistuttavaa kuvaustapaa. Kaavion lisäksi laaditaan sanallinen kuvaus prosessista. Kaaviossa eri toimijat kuvataan rinnakkaisilla ”uimaradoilla”. Kuvassa 23 on esitetty esimerkiksi Kotisairaanhoidon prosessi ”haavan hoito”. Kaaviossa voidaan kuvata prosessia edeltäviä toimintoja, kuten esimerkissä soitto sairaalasta ja kotiutus. Erona UML:n aktiviteettikaavioihin on se, että tässä kuvataan myös prosessin edetessä tarvittavat tai tuotettavat tiedot ja käytettävät tietovälineet. Jos prosessi tapahtuu fyysisesti eri paikoissa, voidaan myös tapahtumapaikat kirjata kaavioon, kuten kuvan 23 esimerkissä kotisairaanhoidon toimisto ja asiakkaan koti. Tästä voidaan edetä ohjelmistovaatimusten suuntaan laatimalla tarvittavat käyttötapaukset.



Kuva 23: Kotisairaanhoito, haavanhoitoprosessi [ToL04b]

Työtoiminnassa tieto on tärkeä työ- ja yhteistyöväline. Tietojen analysointia jatketaan tarkemmalle tasolle kuvaamalla tietoarkkitehtuuri. Tietokokonaisuudet jaetaan pienempiin tietoyksiköihin. Lisäksi on tärkeää selvittää, *mitä tietoa* työssä tarvitaan, *ketkä* tietoa tarvitsevat, *missä* tieto syntyy ja on varastoituna, *miten* tieto saadaan sinne missä sitä tarvitaan ja mitkä seikat *rajoittavat* tiedon kulkua. Keskeisistä tietokokonaisuuksista voidaan kuvata tarpeen mukaan elinkaari, toisin sanoen milloin ja missä tietoa syntyy, milloin sitä tarvitaan ja mihin tieto on varastoitu [ToL04b].

Tietoarkkitehtuurin hahmottamisessa on hyödyllistä miettiä, miten kohdealueen tietoja voisi luokitella erilaisin perustein. Apuna tiedon luokittelussa voidaan käyttää esimerkiksi taulukoita. Tilanteesta riippuu, käytetäänkö yhdellä perusteella luokittelua vai jotakin luokittelujen yhdistelmää.

Tietojen luokittelun yhtenä perusteena voi olla se, missä tieto syntyy. Tällöin tiedot luokitellaan esimerkiksi seuraavasti: *toiminnan sisällä* syntyvät ja tarvittavat tiedot ja *toimintojen välillä kulkevat* tiedot ja *kotihoidon ulkopuolelta haettavat* tiedot. Näin luokittelemalla voidaan tarkastella tiedon saatavuutta, johon taas vaikuttavat tietosuoja, käytettävissä olevat välineet, keinot ja käytännöt sekä tiedon tallennusmuoto ja säilytyspaikka. Esimerkiksi kotisairaanhoitajan tarvitsema lääkelista voidaan laatia erikoissairaalassa, jossa se tallennetaan sairaalan omaan järjestelmään, josta ei kotihoidon työntekijä sitä voi suoraan nähdä, koska yhteyttä kotihoidon ja erikoissairaanhoidon ohjelmistojen välillä ei ole. Tällöin tarvitaan paperinen versio lääkelistasta asiakkaan kotiin.

Toinen luokittelu voidaan tehdä tarvittavan tiedon pysyvyyden ja käyttöajan perusteella: *pysyväluonteiset, pitkäaikaiset* ja *akuutit* tiedot. Tämä vaikuttaa tiedon tallennusmuotoon, esimerkiksi asiakkaan perustiedot, kuten nimi ja sosiaaliturvatunnus ovat pysyväluonteisia tietoja, kun taas ensiapuohjeet asiakkaan saadessa äkillisen sairauskohtauksen ovat akuutteja tietoja. Pitkäaikaiset ja pysyväluonteiset tiedot tallennetaan johonkin yhteiseen tietovarastoon, josta ne ovat asianmukaisten toimijoiden saatavana, kun taas akuutteja tietoja joudutaan hakemaan esimerkiksi puhelimella.

Myös tiedon kohteen perusteella voidaan luokitella. *Asiakkaaseen liittyvä* tieto sisältää asiakkaan terveys-, tulo- ja tilannetiedot. *Yhteiskuntaan liittyvä* tieto sisältää tietoa saatavana olevista palveluista, niiden saantiin liittyvistä ehdoista ja hakumenettelyistä. *Työn organisointiin liittyvä* tieto, kuten työhön liittyvä aluejako ja työvuorolistat, ovat tarpeen työn ohjauksessa ja delegoinnissa. Kotihoitotoiminnassa eri toimijat tarvitsevat erilaista tietoa. Jotkut toimijat tarvitsevat enimmäkseen vain asiakkaaseen liittyviä tietoa, joidenkin on oltava tietoisia myös yhteiskuntaan liittyvistä tiedoista, jotta osaavat opastaa asiakasta. Tämä luokittelu on tarpeen silloin, kun suunnitellaan eri toimijoille erilaisia käyttöoikeuksia eri tietoihin, tai toimijaryhmän mukaan yksilöityä käyttöliittymää.

5.5 ActAD-mallin anti vaatimusmäärittelylle

ActAD on monipuolinen apuväline työtoiminnan ja tietotekniikan yhtäaikaiseen kehittämiseen. Kohdealueen kartoittamisvaiheen alussa toiminnan teoria ohjaa tietolähteiden löytämisessä ja antaa muistilistan selvitettävistä asioista.

Kuvausmenetelmä mahdollistaa kokonaisuuden ja sen osien tarkastelun eri näkökulmista, esimerkiksi yhden toimijan, yhden asiakkaan, palveluketjun, yhden hallinnonalan näkökulma ja niin edelleen. Näkökulman lisäksi voidaan tarkastelun syvyyttä muunnella. Ensinnäkin kutakin toimintoa voidaan tarkastella lähempää, jolloin saadaan esille toiminnon sisäinen rakenne ja tiedot. Voidaan tunnistaa toiminnan malleja, tapoja ja toistuvuuksia. Toisaalta voidaan toiminnot kuvata yleisellä tasolla ja katsoa kokonaisuutta kauempaa, jolloin toimintojen väliset yhteydet tulevat esille selvemmin. Näin tarkastelussa säilyy koko ajan jäljitettävyys yksittäisten toimintojen ja kokonaisuuden välillä. Kokonaiskuvasta voidaan myös havaita tiedonkulun kriittisiä pisteitä ja ongelmakohtia, joissa esimerkiksi ei ole tietotekniikkaa käytettävissä vielä lainkaan. Eri näkökulmista ja eri etäisyydeltä tarkastelemalla saadaan esille erilaisia asioita.

Tunnistettuja tietoja voidaan tarpeen mukaan koostaa tietokokonaisuuksiksi tai purkaa pienemmiksi osiksi. Joitakin paikallisesti saatuja tuloksia voidaan yleistää koskemaan esimerkiksi alueellisesti suurempaa ryhmää, tai vastaavasti voidaan erikoistaa yleistä tietoa spesifimmäksi.

ActAD-kehys on selkeä ja helposti omaksuttava toiminnan kuvaustapa ja antaa mahdollisuuden käyttää selkeitä ja kuvaavia symboleja mallintamisessa. Kotihoitoprojektissa saatujen kokemusten mukaan myös toimialan toimijoiden on helppo omaksua ja ymmärtää kuvaustapa. Tämä mahdollistaa yhteisen ymmärryksen syntymisen ohjelmistosuunnittelun ja toimialan ammattilaisten välille. Tällöin esimerkiksi yhteisissä työpajoissa toimijoiden kanssa saadaan aikaan keskustelua ja tarkennetaan haastattelemalla saatuja tuloksia. Näin voidaan löytää kehityskohteita ja miettiä ratkaisuja ongelmiin. Joissakin tapauksissa ratkaisu on ohjelmisto tai tietotekninen laite, joskus työtoiminnan uudelleenorganisointi, esimerkiksi yksittäisen toimijan toimipisteen siirtäminen lähemmäs jotakin toista. Tietojärjestelmän suunnitteluun kerättyjä tietoja voidaan käyttää työtoiminnan kehittämisessä. Kun työtoimintaa ja tietotekniikkaa kehitetään yhtä aikaa ja samoja lähtötietoja käyttäen, varmistetaan, että tietotekniikka tukee työtoimintaa parhaalla mahdollisella tavalla. Samojen tietojen käyttö kahteen tarkoitukseen tuo myös kustannussäästöjä.

Kun ongelmakenttä pilkotaan selkeisiin osakokonaisuuksiin ohjelmistosuunnittelua varten, voidaan suunnitella ja toteuttaa ohjelmisto osissa. Ohjelmapalaset voidaan tehdä yksi kerrallaan tai monta rinnakkain ja silti varmistua siitä, että osat liittyvät keskenään yhteen toimivaksi kokonaisuudeksi. Toiminnan teoria mahdollistaa osien keskinäisten suhteiden ja kokonaiskuvan selkeän hahmottamisen. Kokonaiskuva toimii ’karttana’ pienemmille osille. Suhde kokonaisuuteen sitoo eri toiminnot paikoilleen. Priorisoinnilla varmistetaan se, että tärkeimmät ja kriittisimmät osa-alueet ratkaistaan ensin.

ActAD sopii erinomaisesti vaatimusmäärittelyn varhaisimpaan vaiheeseen, kohdealueen kartoittamiseen ja analysointiin. Menetelmällä voidaan tehokkaasti tutkia ennen kartoittamatonta kohdealuetta, jolla toimii monia toimijoita eri organisaatioista. Kun kohdealueen kartoittamisessa on käytetty apuna toiminnan teoriaa, voidaan edetä teorian mukaiseen analysointiin, mutta kerätyt tiedot voidaan haluttaessa analysoida muillakin menetelmillä. Toiminnan teoria ei siis sido, vaan sitä voidaan käyttää ja soveltaa juuri niissä vaiheissa, kuin on tarpeen.

Menetelmän puutteena on se, ettei sen avulla saada suoraan ohjelmistovaatimuksia. On siis tärkeää jatkaa menetelmän kehittämistä siten, että polku kohdealueen kartoittamisesta ohjelmistovaatimuksiin voidaan mallintaa selvästi. Aiheesta on jatkotutkimus meneillään [ToE03].

6 Tapahtumalähtöinen vaatimusmäärittely

Tapahtumalähtöisestä vaatimusmäärittelystä puhutaan, kun vaatimusmäärittelyn lähtökohtana ovat asiakkaan liiketoimintatapahtumat. Asiakkaalla tarkoitetaan tässä sitä *kohdealueen yritystä tai organisaatioita*, jonka työn tueksi ollaan suunnittelemassa ohjelmistoa. *Liiketoimintatapahtuma* on asiakkaan työn ulkopuolinen tapahtuma, jolla on asiakkaan liiketoiminnan ja toiminta-ajatuksen kannalta arvoa, ja joka aiheuttaa työssä toimenpiteitä.

Volere-vaatimusmäärittelyprosessin (Volere Requirements Process, myöhemmin Volere tai Robertsonin menetelmä) ovat kehitelleet Susan ja James Robertson. Voleresta on julkaistu vuonna 1999 kirja ”Mastering the Requirements Process”, johon tämä luku perustuu. Robertsonin menetelmä korostaa kohdealueen työn ymmärtämistä ohjelmistosuunnittelussa. Vaatimusmäärittelyn perustana ovat käyttötapaukset, jotka on johdettu asiakkaan liiketoimintatapahtumista.

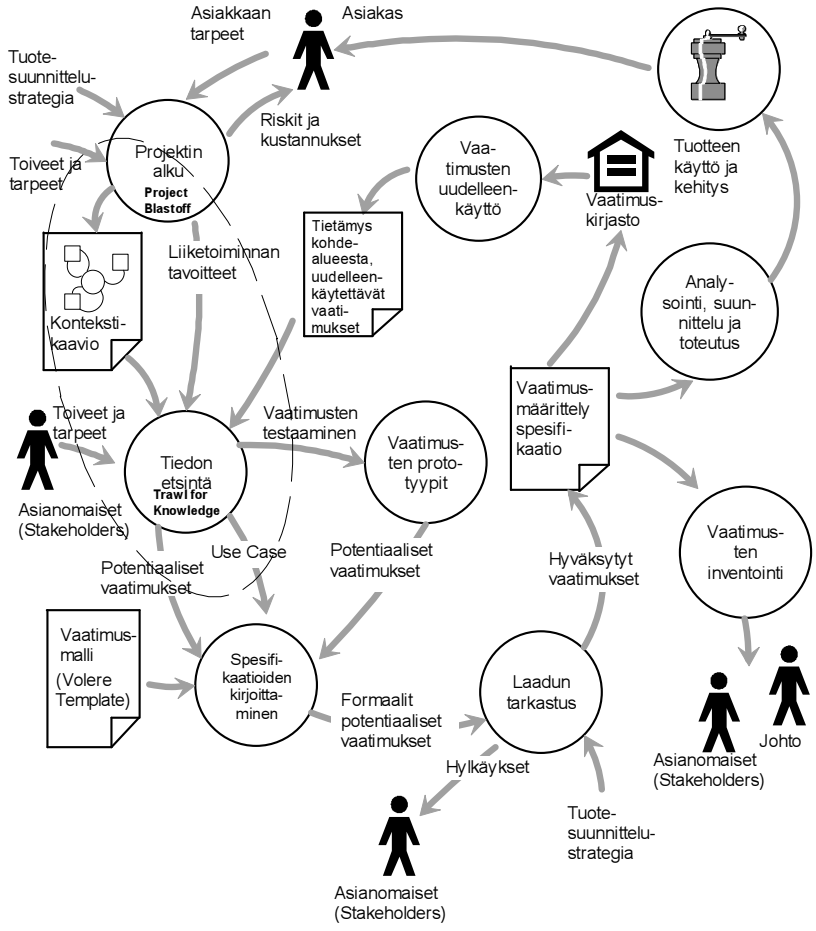
6.1 Robertsonin menetelmä

Volere-prosessi, siihen liittyvät dokumentit ja henkilöt on kuvattu kaaviona kuvassa 24. Kuvaan on katkoviivalla merkitty tässä tutkielmassa lähemmin tarkasteltava osa. Projektin aloittamisvaiheessa (Project Blastoff) luodaan kohdealueesta karkea kuva kontekstikaavion (Context Diagram) avulla. Kontekstikaaviossa kuvataan asiakkaan työ ja sen yhteydet ulkomaailmaan.

Tiedon etsintävaiheessa (Trawl for Knowledge[1]) tarkennetaan kuvaa asiakkaan työstä sekä suunnitellaan tulevan tuotteen rajat ja vastuut työssä. Yhteyksistä ulkomaailman kanssa löydetään liiketoimintatapahtumat (Business Events). Tavoitteellista toimintaa mallinnetaan prosessiketjukuvauksilla. Suunnitellaan systeemin paras vaste kuhunkin liiketoimintatapahtumaan. Mietitään, mitkä prosessiketjun osat on järkevää automatisoida, eli päätetään ohjelmistotuotteen rajat (Product Scope). Liiketoimintatapahtumista johdetaan käyttötapaukset (Event-Driven Use Cases), joista päästään varsinai-

[1] Trawl for something voidaan suomentaa ’troolata, kalastaa laahusnuotalla’. Tällä analogialla pyritään kuvaamaan tiedon etsinnän luonnetta.

siin vaatimuksiin. Vaatimukset kirjataan Volere-mallin mukaisesti, tarkastetaan laadun varmistamiseksi ja kirjataan sopimukseen. Hyväksytyt vaatimukset kirjataan varsinaiseen vaatimusmäärittelyyn. Tällä määrittelyllä on kolme eri käyttötarkoitusta. Sen perusteella analysoidaan, suunnitellaan ja toteutetaan tuote, jota projektissa oltiin alunperinkin tekemässä. Vaatimukset tallennetaan vaatimuskirjastoon, josta niitä voidaan tulevaisuudessa käyttää uudelleen. Vaatimusten inventoinnissa asiakkaan työn asianosaiset ja johto arvioivat vaatimusmäärittelyn avulla suunniteltavan systeemin hyvyyttä vaatimusten osalta ja vaatimusten ristiriidattomuutta sekä riskejä ja kustannuksia.



Kuva 24: Volere-vaatimusmäärittelyprosessi [RoR99 s.11]

6.1.1 Menetelmän vaiheet

Voleressa projektin alussa (Project Blastoff) määritellään projektin ja siinä tuotettavan tuotteen tarkoitus, eli mitattavissa oleva hyöty asiakkaan liiketoiminnalle (Purpose,

Advantage, Measurement, PAM). Projektin aloitusvaiheeseen kuuluu myös projektissa käytettävästä nimeämiskäytännöstä päättäminen ja kustannusarvion teko.
Asiakkaan työn määrittämisellä on keskeinen merkitys Voleressa, ja kohdealueen tutkiminen aloitetaan tämän kuvauksen laatimisesta. *Työllä* (Work) tarkoitetaan asiakkaan liiketoimintaa ja kaikkea, mitä siihen kuuluu. Työ voi olla paitsi kaupallista, myös esimerkiksi terveydenhoitoa, koulutusta tai muuta sellaista. Työn sisälle kuuluvat ihmiset, työvälineet ja ohjelmistot eli kaikki, mitä asiakas tarvitsee tuottaakseen tuotettaan tai palveluaan. Lisäksi määritellään työhön liittyvät asianosaiset (stakeholders). *Asianosaisilla* tarkoitetaan kaikkia niitä, jotka ovat kosketuksissa suunniteltavaan ohjelmistoon, tai joiden elämään ohjelmisto vaikuttaa. Erityisesti määritellään, kuka on ostaja (client), maksaja (customer) ja kuka loppukäyttäjä (user). Suomenkielessä kaikista kolmesta edellä luetellusta henkilöstä käytetään sanaa asiakas. Tämä on harhaanjohtavaa, sillä useinkin ostajalla, maksajalla ja käyttäjillä on ohjelmiston suhteen eri hyötytavoitteet ja merkitykset.

Työ on yhteyksissä ulkomaailmaan. Nämä yhteydet kuvataan prosessin aloitusvaiheessa korkean tason kontekstikaaviolla. Aluksi merkitään kontekstikaavioon työ ja sen ympärille kaikki systeemit, joihin työ on yhteyksissä. Ulkopuolinen systeemi voi olla aktiivinen, autonominen tai yhteistyötä tekevä. *Aktiivinen systeemi* odottaa vastetta välittömästi ja on interaktiivisessa suhteessa työhön. Usein nämä ovat ihmisiä, esimerkiksi tietojärjestelmän käyttäjiä. *Autonominen systeemi* toimii itsenäisesti, mutta viestii kuitenkin työn kanssa. Vuorovaikutus ei ole interaktiivista, eli vastausta ei odoteta välittömästi. *Yhteistyötä tekevä* systeemi tekee luotettavasti sen mitä siltä odotetaankin ja on yleensä automatisoitu. Se voi olla joku ohjelmisto, joka suorittaa työn tarvitseman palvelun. Esimerkiksi terveyskeskuksen vastaanoton ajanvarausohjelmistolle varausta suorittava asiakas on aktiivinen naapurisysteemi, tilastointi on autonominen systeemi ja terveyskeskuksen laboratorion ohjelmisto on yhteistyötä tekevä systeemi.

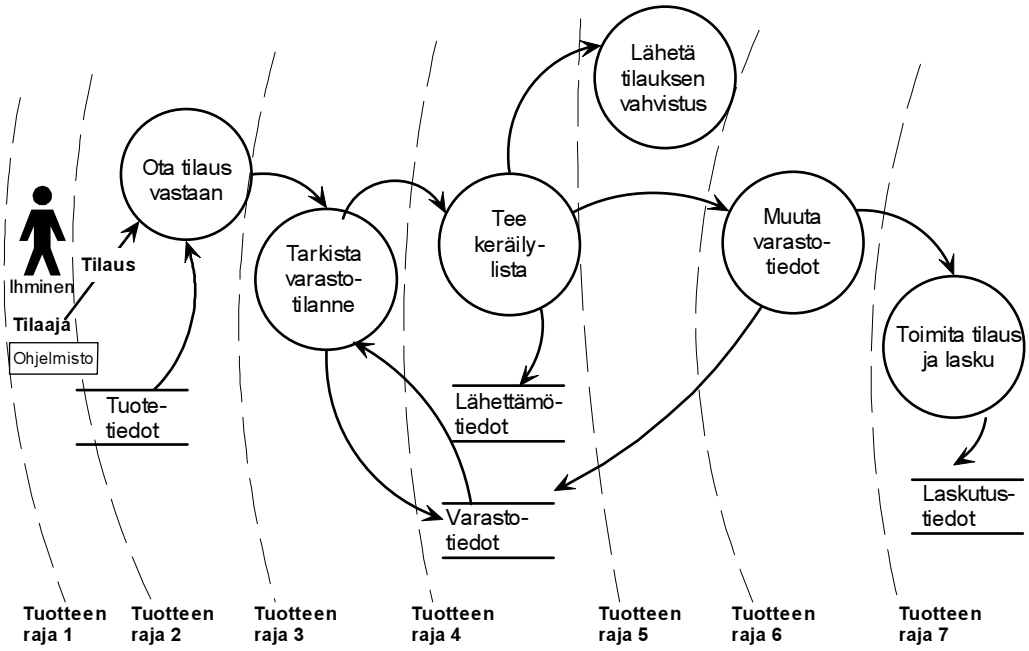
Yleensä asiakkaan työ on liian monimutkainen ja suuri kokonaisuus, jotta sen voisi mallintaa yhteen kuvaan tarkalla tasolla. Tiedon keräämistä ja analysointia varten on järkevää jakaa työ pienempiin, helpommin ymmärrettäviin osiin. Näitä osia voivat olla esimerkiksi työhön kuuluvat osatehtävät, joilla on selkeä tarkoitus ja rajat.

Kontekstikaavion yhteyksistä etsitään liiketoimintatapahtumat. *Liiketoimintatapahtuma* on työn ulkopuolella tapahtuva asia, joka vaikuttaa työhön ja jolla on työn kannalta merkitystä. Kun työhön tulee viestejä (herätteitä) ulkopuolisilta systeemeiltä, työssä käynnistyy prosessiketju, jonka tarkoitus on tuottaa herätteeseen paras mahdollinen vaste. Esimerkiksi nettikaupassa asiakkaan suorittama tilaus on liiketoimintatapahtuma, jonka vasteena voisi olla se, että asiakas saa vuorokauden kuluessa tilauksenvahvistuksen ja viikon kuluttua tilauksesta tilaamansa tuotteen ja laskun siitä. Tilaus aiheuttaa myös varastokirjanpidossa muutoksia. Jollei viestejä ulkopuolelta tule, työn oletetaan olevan passiivinen.

Löydetyt liiketoimintatapahtumat kirjataan tapahtumalistaan (Event List). Jokaisesta tapahtumasta kirjataan sen aiheuttamat tietovirrat ja se, onko tietovirta sisään työhön (input) vai ulos työstä (output).

Kun työ saa naapurisysteemiltä herätteen, työssä käynnistyy prosessiketju, joka käsittelee herätteen, varastoi tarvittavan tiedon sekä muodostaa herätteeseen tarvittavan vasteen (kuva 25). Prosessiketjussa voi olla sekä ihmisten suorittamia toimia, että tietokoneen toimintaa. Kukin prosessiketju on eristyksissä muista työhön kuuluvista prosesseista. Prosessiketju on yhteyksissä muihin prosesseihin vain varastoidun tiedon välityksellä. Robertsonin mukaan tietovarastoja ovat esimerkiksi tietokannat tai paperiset tietovälineet.

Kiinnostava kysymys on, voidaanko tietovaraston käsitettä laajentaa siten, että ihmisten omaama tieto (tacit knowledge, ks. kappale 2.3.1, s.11) otetaan huomioon tietovarastona. Jos näin ajateltaisiin, on prosessiketjuja muodostettaessa määriteltävä ihmisten tarvitsema tieto ja varmistettava esimerkiksi koulutuksella, että kaikilla prosessiin osallistuvilla, myös uusilla työntekijöillä ja sijaisilla, on tarvittavat tiedot. Tämä on mielenkiintoinen jatkotutkimuksen aihe, sillä hiljainen tieto on merkityksellistä työn sujuvuuden kannalta.



Kuva 25: Esimerkki prosessiketjusta

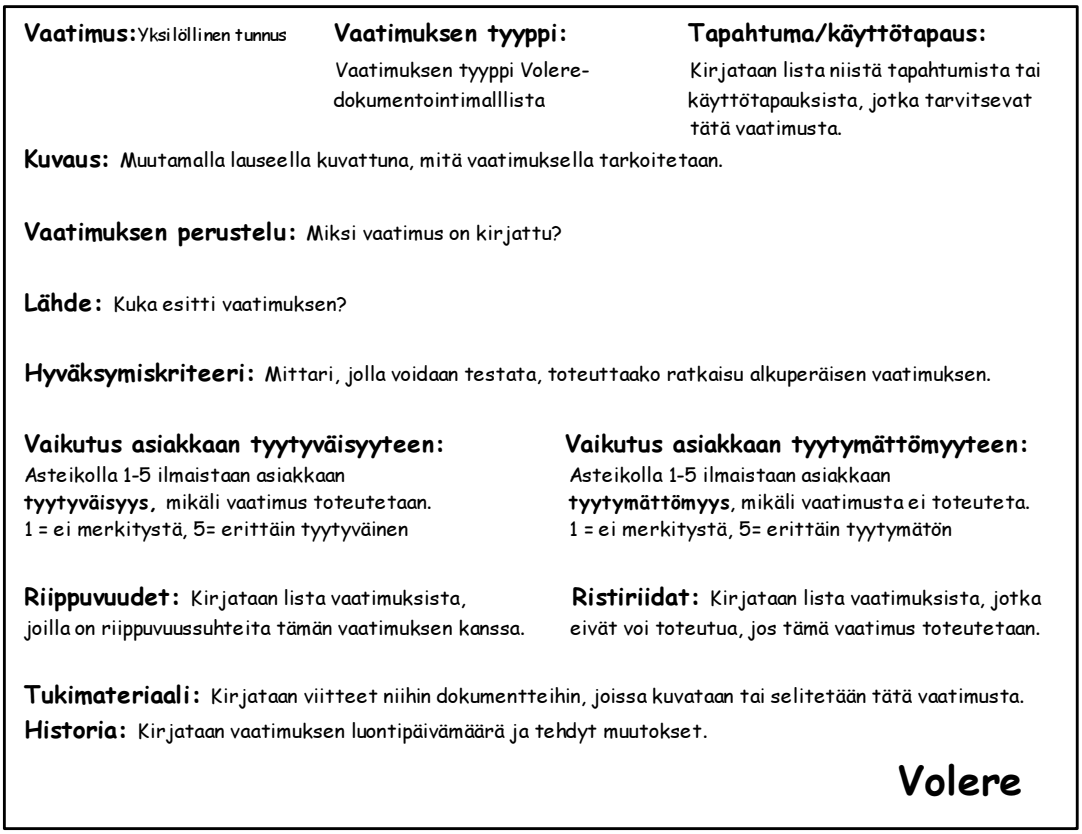
Määriteltävän tuotteen rajaamiseksi on päätettävä, mitkä prosessit automatisoidaan ja mistä ohjelmisto alkaa. Tietotekniikka ei ole aina paras mahdollinen ratkaisu. Kuvassa 25 on esitetty katkoviivoilla erilaisia vaihtoehtoisia tuotteen rajoja. Jos tuotteen rajana on raja numero seitsemän, vain tilauksen toimitus ja laskutus automatisoidaan, muut prosessin vaiheet toteutetaan manuaalisesti. Mikäli noudatetaan rajaa numero kuusi, varastotietojen päivitys ja laskutus tapahtuisi automaattisesti ja niin edelleen. Jos automatisoinnin rajana on raja numero kaksi, kaikki prosessiketjun prosessit on automatisoitu. Esimerkiksi kyseessä voisi olla internetin kautta tapahtuva ostos, jossa tilaajana on ihminen. Jos valitaan tuotteen rajaksi raja numero yksi, eli tilaajakin rajataan tuotteen sisälle, asiakkaan ei tarvitse erikseen tilata tuotetta, vaan tilaus tapahtuu automaattisesti. Näin voisi olla esimerkiksi terveyskeskuksen tarvikevaraston tapauksessa, jossa laboratorio-ohjelmisto on tilaajana. Kun laboratoriohoitaja suorittaa terveyskeskuksen asiakkaalle testin, käytetyt materiaalit poistetaan laboratorion varaston kirjanpidosta automaattisesti ja varaston vähettyä tilausrajaan tilataan materiaalia automaattisesti. Terveydenhuollon piirissä toinen esimerkki asiakkaan rajaamisesta tuotteen sisälle voisi olla ihmisen kutsuminen määräaikaistarkastukseen automaattisesti tietyn iän täytyttyä.

Kun tuotteen rajat on päätetty, ja tiedetään, mitkä prosessit automatisoidaan, laaditaan korkean tason käyttötapauskaavio, jossa näkyvät kaikki liiketoimintatapahtumista johdetut käyttötapaukset. Yhdestä liiketoimintatapahtumasta tulee yksi tai useampi käyttötapaus. Tässä mallissa käyttötapauksen toimijoina (actors in use case) voivat ihmisen lisäksi olla muukin työhön yhteyksissä olevat systeemit, eli esimerkiksi toinen tietokonesovellus tai tietokanta. Näiden käyttötapausten avulla ja niitä tarkentamalla saadaan aikaan suunniteltavalle systeemille potentiaaliset vaatimukset. Vaatimusten keskinäiset riippuvuudet ja ristiriidat on otettava huomioon, kun priorisoidaan ja valitaan toteutettavia vaatimuksia. asiakkaan mielipide vaatimuksen toteutumisen tärkeydestä ohjaa vaatimusten priorisointia ja valintaa. Vaatimusten on vielä läpäistävä *laadun tarkastus* (Quality Gateway), ennen kuin ne lopullisesti hyväksytään ja niitä voidaan käyttää suunnitteluvaiheen perustana.

6.1.2 Tiedon keräämisestä ja dokumentoinnista

Tiedon etsintävaiheessa tarkennetaan kuvaa asiakkaan työstä. Tietolähteinä toimivat yleensä asiakas, loppukäyttäjä ja asiakkaan asiakas. *Asiakkaan asiakas* (ihminen tai organisaatio) on potentiaalinen naapurisysteemi ja siten tärkeä tietolähde. Tietoa voidaan kerätä erilaisin menetelmin, kuten dokumentteja tutkimalla, haastatteluilla, osallistuvalla havainnoinnilla tai työpajoissa. Nykytilanteen kuvaaminen auttaa ohjelmistosuunnittelijaa ymmärtämään asiakkaan työtä. Nykytilan kuvaus on tärkeä senkin vuoksi, että voidaan päättää, mitä osia entisestä systeemistä säilytetään, mitä muutetaan ja mitä lisätään. Kuvaustekniikkana käytetään kontekstikaaviota ja siitä johdettua tapahtumalistaa, miellekarttaa sekä prosessiketjukuvausta. Työstä pyritään löytämään rakenteisuutta ja erilaisia abstrakteja malleja, joita hyödynnetään suunnittelussa. Jos usealla työvaiheella tai osatehtävällä on samankaltaisia osia, voidaan näitä käyttää abstraktina kehyksenä ja näin nopeuttaa vaatimusten löytämistä ja suunnittelutyötä.

Liiketoimintatapahtumista johdetuilla käyttötapauksilla on keskeinen osa tiedon etsinnän apuvälineenä. Niiden pohjalta järjestetään työpajoja, joissa käyttäjä sekä vaatimusten analysoija ja ohjelmistosuunnittelija käyvät läpi kaikki käyttötapaukset. Toimialan asiantuntijan tietämys tavallaan käännetään ohjelmistosuunnittelussa tarvittavaksi tiedoksi. Samalla pohditaan, mikä on paras vaste kuhunkin tapahtumaan, eli voidaanko asiakkaan yrityksen toimintatapoja jotenkin parantaa.



Kuva 26: Volere-kortti, mukailtu [www06]

Vaatimusmäärittelyn dokumentoinnissa käytetään apuna Volere-spesifiontimallia (The Volere Specification Template [RoR99 s.137 - 164]). Malli sisältää vaatimusmäärittelydokumentin rungon, josta poimitaan omaan projektiin sopivat osat, sekä vaatimuksen mallipohjan (Volere-kortti, Volere Shell, kuva 26) yksittäisen vaatimuksen kirjaamiseksi.

Vaatimusmäärittelydokumentin sisältö jaetaan neljään kategoriaan: tuoterajoitteet (Product Constraints), toiminnalliset vaatimukset, ei-toiminnalliset vaatimukset (ks. kappale 2.2 s. 8)ja projektiin liittyvät asiat (Project Issues). Toiminnallisiin vaatimuksiin kuuluu tuotteen rajaaminen ja tulevan tuotteen varsinainen toiminnallisuus. Toiminnalliset ja ei-toiminnalliset vaatimukset kirjataan Volere-korteille. Volere-mallia käytetään apuna jo vaatimusten keräämisvaiheessa muistilistana selvitettävistä asioista.

Tuoterajoitteet eivät ole varsinaisia vaatimuksia, mutta ne vaikuttavat olennaisesti suunniteltavaan tuotteeseen. Tuoterajoitteita aiheuttavat muun muassa asiakkaan olemassa olevat sovellukset ja projektin käytettävissä oleva aika ja raha. Product Const-

raints -otsikon alle kirjataan myös tuotteen tarkoitus, projektin asiakas, maksaja ja tuotteen käyttäjä, käytettävät nimeämiskäytännöt, oletukset ja muut suunniteltavan tuotteen kannalta olennaiset asiat. Kaikkien edellä lueteltujen asioiden kirjaaminen auttaa varsinaisten vaatimusten ymmärtämistä.

Projektiin liittyvistä asioista kirjataan perusasioiden lisäksi esimerkiksi avoimiksi jätettävät kysymykset, esille tulleet uudet ongelmat sekä uudelleenkäytettävät valmiit ratkaisut. Kun suunniteltavan systeemin vaatimukset on määritelty, voidaan tässä vaiheessa jo arvioida toteutusprojektin riskit ja kustannukset, jotka kirjataan Project Issues -otsikon alle. [RoR99]

6.2 Robertsonin menetelmän soveltaminen kotihoitoon

Volere-menetelmää on tässä tutkimuksessa sovellettu sen alkuvaiheen osalta kotihoidon kohdealueeseen. Soveltamisessa ei ole järjestetty erikseen tapaamisia kotihoidon asianosaisten kanssa, vaan esimerkeissä esille tuleva tieto kotihoidosta on peräisin havainnoista ja kokemuksista PlugITin Kotihoito-projektista (Luku 4). Vaatimusten dokumentointi on rajattu esimerkin ulkopuolelle ja keskitytty vaatimusmäärittelyn alkuvaiheeseen: kohdealueen mallintamiseen ja vaatimusten etsimiseen.

6.2.1 Työn rajaaminen (Work)

Tämän kappaleen esimerkeissä ohjelmistosuunnittelun asiakkaalla tarkoitetaan kotihoito-organisaatioita kokonaisuudessaan ja siihen viitataan nimellä *Kotihoito*. Nimellä *Asiakas* viitataan Kotihoidon asiakkaaseen.

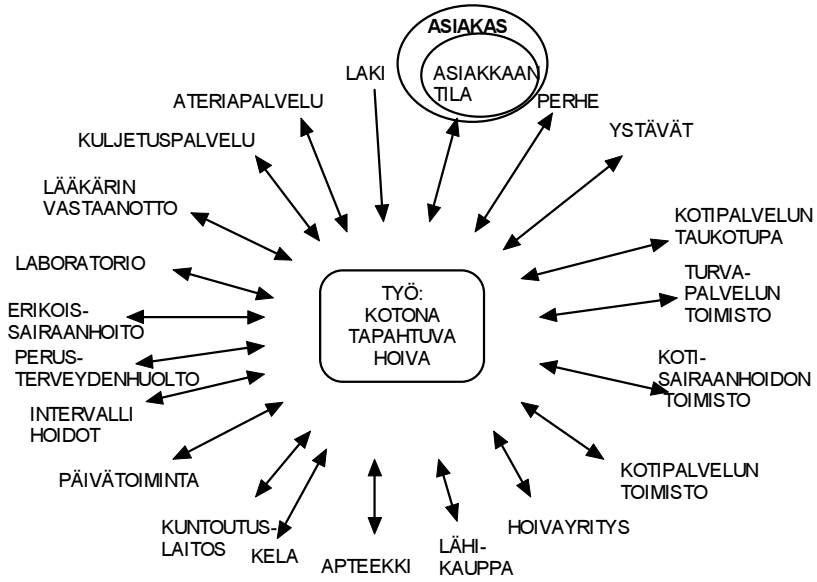
Kotihoito tuottaa Asiakkaan kotona tapahtuvia hoito- ja hoivapalveluja. Palveluja toteuttavat Asiakkaan kotona useat eri toimijat eri organisaatioista, esimerkiksi kotipalvelu, kotisairaanhoito, ateriapalvelu ja apuvälinelainaamon työntekijät. Yhteyksiä ulkopuoliseen maailmaan on paljon, ja nämä yhteydet vaihtelevat Asiakas- ja tilannekohtaisesti hyvinkin paljon.

Aluksi määritellään suunniteltavan tuotteen *tarkoitus* (purpose), *hyödyt* (advantage) Kotihoidon (client) toiminnalle sekä miten hyödyt *mitataan* (measurement). Tämä

määrittely on projektin aloituksen olennainen osa, ja määrittelyyn osallistuvat kaikki keskeisimmät asianosaiset (stakeholders). Todellisessa maailmassa on ehkä ongelmana se, että projektin alussa ei vielä ole tarpeeksi tietoa kohdealueesta, jotta kaikki asianosaiset voidaan löytää. Tässä esimerkissä määritellään tuotteen tarkoitus, Asiakkaan tuotteesta saama hyöty ja saadun hyödyn mittaaminen seuraavasti:

> **Tuotteen tarkoitus on tukea Asiakkaan kotona tapahtuvaa hoivatyötä, jota toteuttavat monesta eri organisaatiosta tulevat toimijat saumattomana palveluketjuna.**
>
> **Kotihoidon tuotteesta saama hyöty on, että Kotihoidon työntekijöiden aikaa kuluu vähemmän tiedon etsimiseen ja kirjaamiseen ja aikaa jää enemmän varsinaiseen hoitotyöhön.**
>
> **Tätä hyötyä mitataan Kotihoidon Asiakkaiden lukumäärällä työntekijää ja työpäivää kohti. Kotihoidon parantunutta laatua mitataan asiakaskyselyllä. (**Hyödyn mittaamista ei tässä esimerkissä toteuteta.)

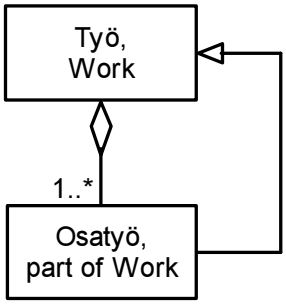
Työksi määritellään tässä esimerkissä *Kotona tapahtuva hoiva*, jolloin ympäröiviä systeemejä ovat esimerkiksi eri toimijoiden tietojärjestelmät toimipisteissä ja taukotuvilla Asiakkaan kodin ulkopuolella, sekä iso joukko kotihoidon Asiakkaan hoivaamiseen liittyviä tai vaikuttavia henkilöitä, palveluita ja organisaatioita. Näitä on esitetty kuvassa 27. Asiakkaan tilassa tapahtuva muutos on selvästi tapahtuma, johon työn on vastattava jollakin tavalla. Mikäli halutaan pitää itse työ passiivisena, ja ajatella, että prosessiketjun voi käynnistää vain työn ulkopuolella tapahtuva liiketoimintatapahtuma, on myös Asiakkaan tila sijoitettava työn ulkopuolelle. Tämä puolestaan johtaa ajatukseen, että Asiakkaan tilasta on oltava jonkinlainen ulkoistettu tietovarasto (Asiakas) työn ulkopuolella. Tässä vaiheessa ei vielä tiedetä tarkkaan, missä työn rajat ovat tai mitkä ovat automatisoitavia prosesseja.



Kuva 27: Korkean tason kontekstikaavio, työ: kotona tapahtuva hoiva

Asiakas itse on hoitotyöhön vaikuttava keskeisin tietolähde. Toisaalta Asiakkaasta on paljon tietoa tallennettuna kodin ulkopuolella eri organisaatioiden järjestelmiin, josta sitä tulee saada Kotihoidon työntekijöiden ja Asiakkaan itsensäkin käyttöön. Asiakkaan tiedot ovat nykyään hajallaan eri paikoissa ja eri muotoisena (paperilla, eri organisaatioiden tietojärjestelmissä, hiljaisena tietona ihmisten päässä). Asiakkaan tietojen hajautuneisuus vaikeuttaa Kotihoidon työtä.

Koska kotona tapahtuva hoiva sisältää hyvin monenlaista ja monien eri toimijoiden tekemää työtä, kokonaisuus on liian laaja tutkittavaksi yhtenäisenä. Tiedon keräämisessä ja analysoinnissa ei voida varmistua siitä, että laaja alue tulee kartoitettua riittävällä tarkkuudella. Siksi työ on paloiteltava pienempiin osiin *osatöiksi (part of Work)* (kuva 28). Osatyöt käsitellään kuten työ. Robertson ei esitä osatöihin jakoa, mutta PlugIT Kotihoito-projektissa saatujen kokemusten perusteella tämän kaltainen jako on välttämätöntä ja perusteltua, kun tarkastellaan laajaa tai etukäteen määrittelemätöntä kohdealuetta, kuten kotihoito. Kotihoidossa yksi mahdollinen jakoperuste on toimijaryhmien mukainen jako

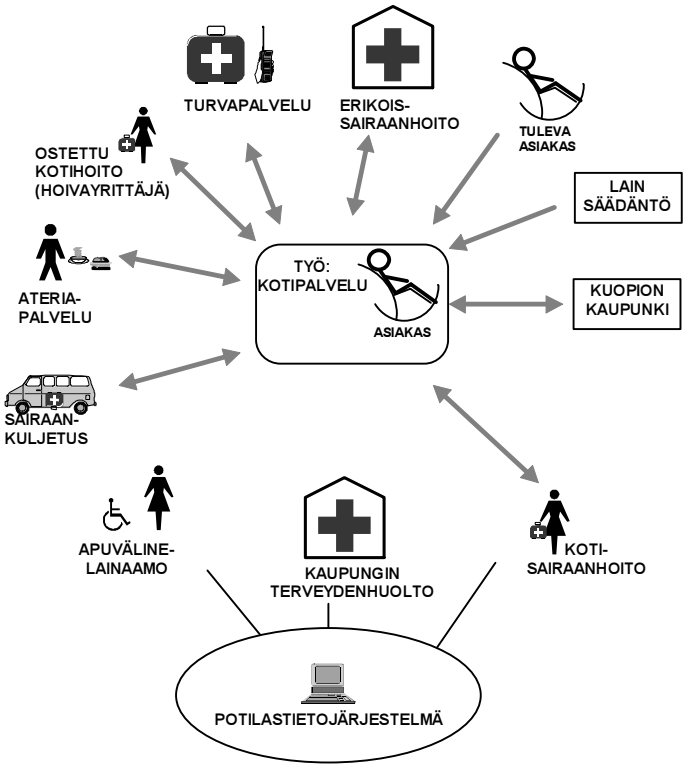


Kuva 28: Työn ja osatyön käsitteet

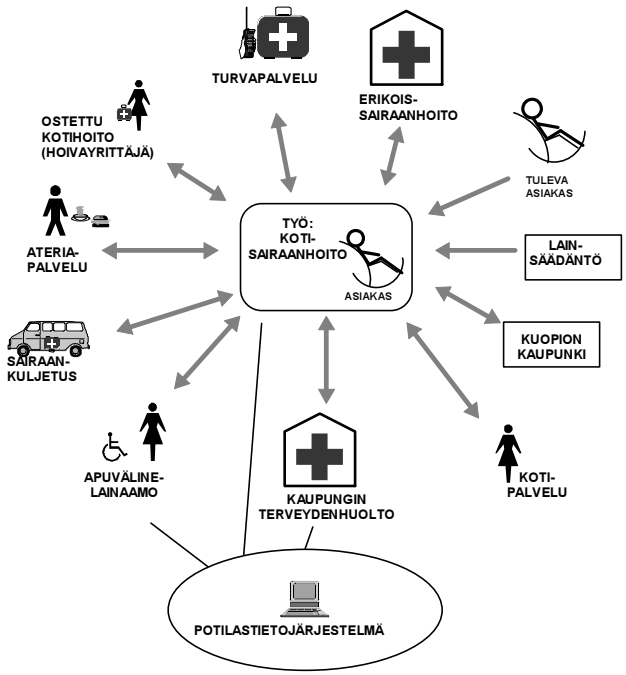
Kotihoito sisältää useita osatöitä, joista keskeisimmät ovat kotipalvelu ja kotisairaanhoito. Ympäröivät systeemit ovat molemmille osatöille melkein samat (kuvat 29 ja 30). Kaikista ympäröivistä systeemeistä ei tule herätteitä molempiin osatöihin. Kotipalvelu ja kotisairaanhoito eroavat toisistaan paitsi työnkuvaltaan, myös siltä osin, että kotisairaanhoito voi käyttää perusterveydenhuollon ohjelmistoa, joka sisältää kotihoito-osion. Kotipalvelun työntekijöillä ei ole käytössään tietokoneita, eikä käyttöoikeuksia perusterveydenhuollon ohjelmistoon. Yhteydenpito tapahtuu keskustelun avulla esimerkiksi puhelimitse tai erilaisilla paperisilla tietovälineillä kuten lähete. Robertsonin menetelmän puutteena on, ettei käytettyjä tietovälineitä eritellä.

Kotihoitoa tukevan systeemin suunnittelussa tulisi ottaa huomioon eri käyttäjäryhmät ja heidän erilaiset tiedon tarpeensa. Herätteen aiheuttama prosessointi sisältäisi siis sekä Asiakkaan kotona tapahtuvat toimet että toimistolla tai tiimitilassa tehtävät toimet, esimerkiksi haavahoito Asiakkaan kotona ja kotikäynnin kirjaus terveydenhuollon järjestelmään.

Asiakkaan sijoittaminen kontekstikaavioon on pulmallista. Tuleva Asiakas on työn ulkopuolella, ja aloittaa kotihoidon tilaamalla (kuva 29). Toisaalta Asiakas osallistuu kotihoitoon itse aktiivisesti toimintakykynsä mukaan ja on kotihoidon kohteena, jolloin voidaan ajatella Asiakkaan olevan työn sisällä (vrt. ActAD). Jos Asiakkaan kotihoito lakkaa, hänen tietonsa jäävät Kotihoidon tietovarastoon, ja ovat sieltä käytettävissä, mikäli Asiakkuus on tarpeen aloittaa uudelleen.



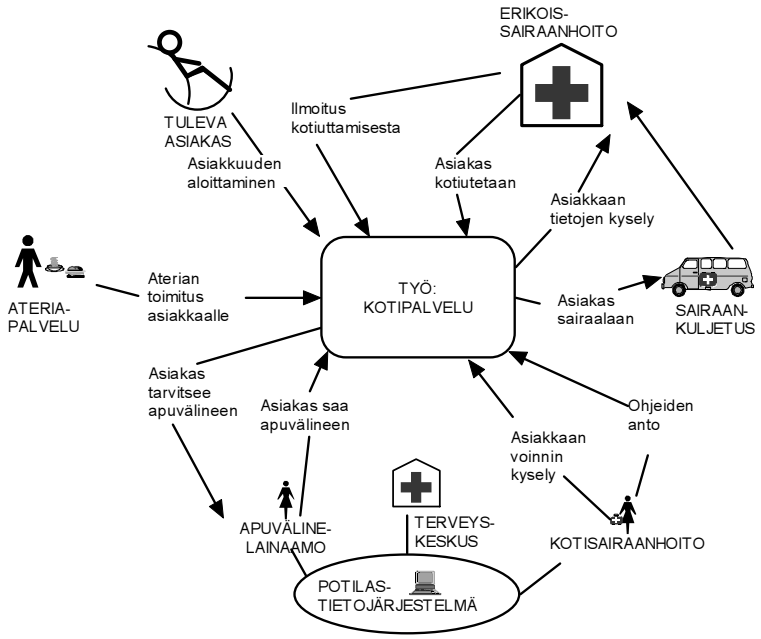
Kuva 29: Kontekstikaavio, työ: kotipalvelu



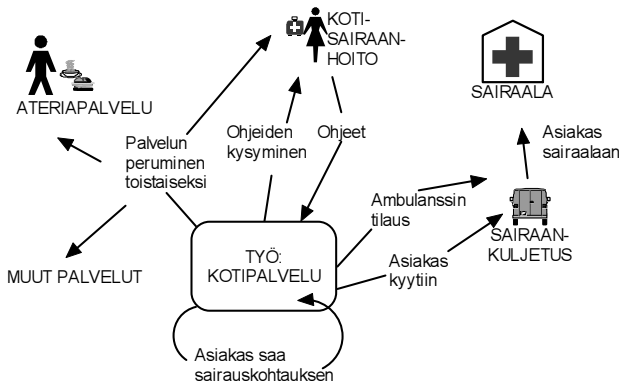
Kuva 30: Kontekstikaavio, työ: kotisairaanhoito

6.2.2 Yhteyskaavioiden luonti

Yhteyskaaviosta nähdään ne (liike)toimintatapahtumat, jotka aiheuttavat työssä prosessointia. Kotihoidon luonteesta johtuu, että tapahtumia on paljon ja monilta eri tahoilta. Näitä kaikkia ei voi kuvata samassa kaaviossa selkeästi. Tästä johtuen esimerkissä on keskitytty muutamaan menetelmän kannalta kiinnostavaan tapahtumaan sekä kotipalvelun että kotisairaanhoidon kohdalta. Jotkut tapahtumat aiheuttavat monta eri toimintaa, ja tuntuu selkeämmältä kuvata ensin yleistasoinen yhteyskaavio (kuva 31), jota voi tarkentaa yksittäisillä tarkemman tason yhteyskaavioilla (kuva 32). Robertson ei esitä tätä tarkentamista Volere-mallissa.



Kuva 31 Yhteyskaavio (yksinkertaistettu yleiskuva)



Kuva 32: Yhteyskaavio (tarkennettu tilanteeseen: Asiakas saa sairauskohtauksen)

Koska kotipalvelun ja kotisairaanhoidon työnkuvat eroavat toisistaan, voi sama impulssi naapurisysteemiltä aiheuttaa erilaisen prosessiketjun näissä töissä. Esimerkiksi, kun sairaalasta ilmoitetaan Asiakkaan kotiuttamisesta, kotipalvelu vastaa järjestelemällä jonkun työntekijän ottamaan vastaan Asiakasta tämän kotiin, mutta kotisairaanhoito saattaa käydä esimerkiksi kotiutusta seuraavana päivänä. Samankaltaista näissä prosessiketjuissa kuitenkin on se, että molempiin kirjataan Asiakkaan kotiintuloaika. Syntyy rinnakkaiset prosessiketjut, jotka vaikuttavat toisiinsa. Robertson ei tarkastele rinnakkaisia prosessiketjuja. Rinnakkaisuutta on kuitenkin hyödyllistä tarkastella, koska sitä kautta löytyy potentiaalisia kehityskohteita tiedonkulussa ja palveluketjussa. Kannattaa varmistaa, ettei samaa työtä tehdä useampaan kertaan ja että työ on mahdollisimman sujuvaa.

Jotkut herätteet aiheuttavat prosesseja työssä, mutta eivät anna lähettävälle järjestelmälle mitään varsinaista vastausta, esimerkiksi kotiutustilanteessa ei nykyisin sairaalaan mene tietoa siitä, tuliko Asiakas perille vai ei. Tämä voi olla yksi niistä asioista, jotka olisi muutettava tulevaan järjestelmään.

Kotihoidossa tapahtumia voi tulla myös työn sisältä. Asiakas hoidon kohteena on työn sisällä, ja esimerkiksi hänen terveydentilassaan tapahtuva muutos, vaikkapa sairauskohtaus, aiheuttaa herätteen (kuva 32) ja siten kotihoidon prosessiketjun käynnistymisen. Toisaalta myös kotihoidon työntekijät ovat työn sisällä ja esimerkiksi työvuorolistan muuttaminen työntekijän sairastumisen vuoksi saa herätteensä työn sisältä. Robertsonin menetelmässä heräte tulee aina työn ulkopuolelta, mikä rajoittaa ajattelua. Kun etsitään liiketoimintatapahtumia vain yhteyskaaviosta, eli työn yhteyksistä ulkomaailmaan, jäävät työn sisältä päin lähtevät herätteet huomioimatta. Lisätutkimusta tarvitaan siitä, miten työ voidaan muuttaa aktiiviseksi.

6.2.3 Tapahtumalistojen luonti ja prosessiketjut

Tapahtumalistoihin kirjataan (liike)toimintatapahtumat sekä niiden ja työn väliset tietovirrat. Tietovirta voi olla työhön sisään (in) tai työstä ulos (out). Kukin tietovirta liittyy yhteen ja vain yhteen yhteyskaaviossa esitettyyn tapahtumaan [RoR99, s.60]. Samaan tapahtumaan voi liittyä useita tietovirtoja. Taulukossa 1 on esitetty osa kotihoidon tapahtumista.

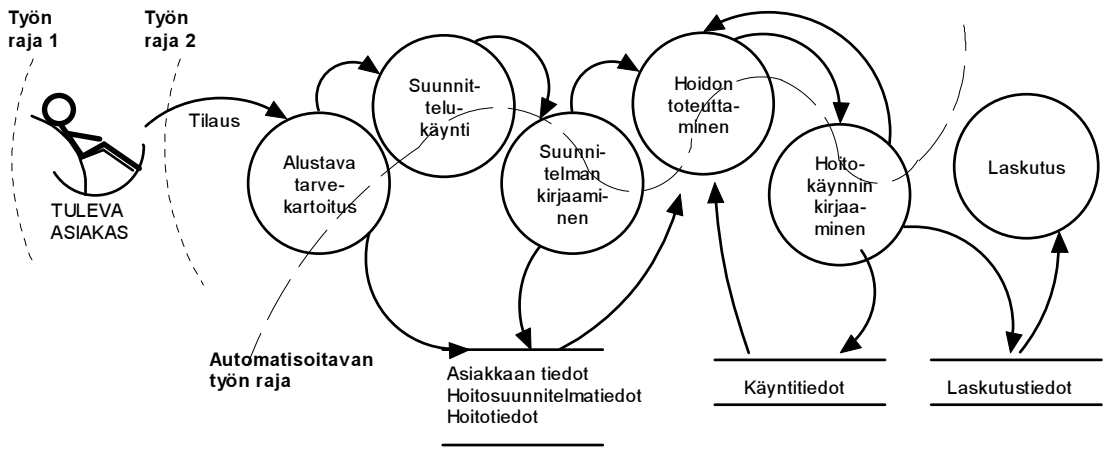
Taulukko 1: Tapahtumalista, WORK: Kotipalvelu

Tapahtuma	**Input & Output**
Asiakkaan kotiuttamisesta ilmoitetaan	Kotiutusaika (in)
Asiakas kotiutetaan	Kotiutustiedot (in)
Asiakas tarvitsee apuvälineen	Välineen nimi, Ajanjakso (out)
Asiakas saa apuvälineen	Aika, Käyttöohjeet (in)
Asiakkaan ateriointi	Ateria (in)
Asiakas saa sairauskohtauksen (kotipalvelun käyntiaikana)	Asiakkaan tila (in |out) Sairasauton tilaus (out) Ohjeiden kysely (out) Ohjeiden vastaanotto (in) Muiden palvelujen peruminen (out)
Turvapalvelun käynti Asiakkaan luona (ei kotipalvelun käyntiaikana)	Käyntitiedot (in) Onko Asiakas viety sairaalaan (in) Mahdolliset jatkohoito-ohjeet (in)
Kotisairaanhoitajan käynti	Kysely Asiakkaan tilanteesta (in) Mahdolliset hoito- ja tarkkailuohjeet (in)
Asiakkuuden aloittaminen	Asiakkaan tiedot (in) Hoitosuunnitelmatiedot (out)

Tapahtumalistan jokaista tapahtumaa pitäisi Volere-mallin mukaan käsitellä työpajassa, jossa on mukana kunkin tapahtuman asianosaiset. Työpajoissa tapahtumiin mietitään paras mahdollinen vaste ja määritellään prosessiketju, jonka heräte aiheuttaa työssä. Tätä esimerkkiä laadittaessa ei ollut mahdollista järjestää työpajaa, joten esimerkit laadittiin ActADin soveltamisessa toteutettujen työpajojen perusteella.

Esimerkki 1:

Kun Asiakas haluaa aloittaa kotihoitopalvelun, hän soittaa puhelimella tilauksen kotipalvelunohjaajalle, joka kyselee Asiakkaalta esimerkiksi Asiakkaan terveydentilaan ja liikuntakykyyn liittyviä asioita ja arvioi alustavasti, millaisia kotihoidon palveluja Asiakas tarvitsee. Kotipalvelunohjaaja tekee myöhemmin yhdessä Asiakkaan asuinalueen omahoitajan kanssa suunnittelukäynnin, jonka tuloksena syntyy Hoito- ja palvelusuunnitelma. Hoitokäynnit toteutetaan suunnitelman mukaisesti ja käynnit kirjataan Asiakkaan kotona olevaan viestivihkoon. Kotipalvelunohjaaja kirjaa käynnit myös laskutusta varten (kuva 33).



Kuva 33: Prosessiketju kotihoidon tilaus, WORK kotipalvelu

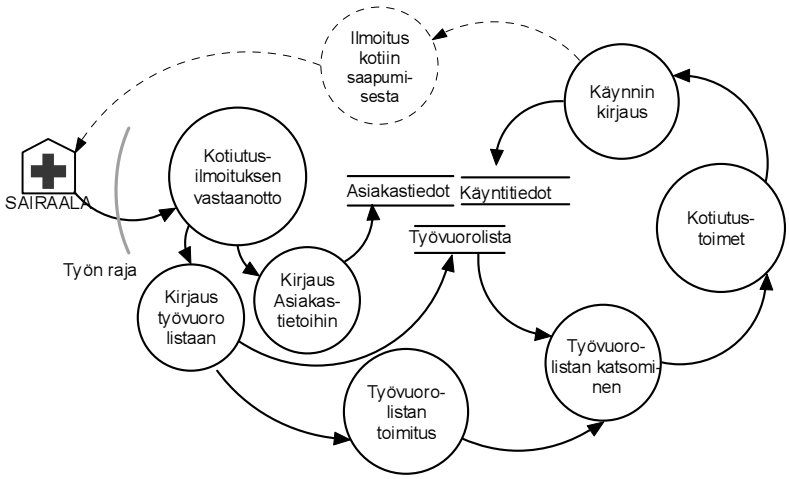
Jos työn rajaksi otetaan kuvassa 33 merkitty työn raja 2, heräte on Asiakkaan tilaus. Jos taas valitaan työn raja 1, se tarkoittaa, että kotipalvelu ottaa Asiakkaaseen yhteyttä esimerkiksi Asiakkaan ylittäessä jonkin määritellyn ikärajan. Heräte on tässä tapauksessa Asiakkaan iän lisääntyminen ja tieto tästä voisi tulla vaikkapa Väestörekisteristä. Kotipalvelu voi ottaa yhteyttä Asiakkaaseen myös sellaisissa tapauksissa, joissa esimerkiksi huolestunut naapuri ilmoittaa kotihoidon tarpeesta. Tällöin heräte on naapurin yhteydenotto. Vaste kaikkiin herätteisiin on hyvin suunniteltu ja toteutettu kotihoito.

Kuvassa 33 esitetyt prosessit ovat korkealla abstraktiotasolla. Työn automatisoinnin rajaa ei ole helppo määritellä. Selkeimmin automatisoitavissa on laskutus-prosessi. Mikäli käynnin kirjaaminen sisältää vain käyntiaikatiedot, olisi se hyvin yksinkertainen automatisoida. Jos käyntitietoihin kirjataan tietoja Asiakkaan tilan muutoksesta, automatisoinnissa on otettava huomioon kaikki muut Kotihoidon asianosaiset, joille tiedon pitää mennä, ja joilla on erilaiset käyttöoikeudet ja -mahdollisuudet kirjauksen katsomiseen. Esimerkiksi kotisairaanhoidon ja Asiakkaan omaisen tulee olla tietoisia Asiakkaan terveyden tilan muutoksista. Robertsonin mallissa kiinnitetään huomiota tiedon kirjaamiseen tietovarastoon, mutta ei tarkastella erilaisia käyttäjäryhmiä tai sitä, kenellä tiedon pitäisi olla käytettävissä.

Esimerkki 2:

Prosessiketjuja mietittäessä löydetään kehittämiskohteita myös työtavoissa ja käytännöissä, mikä havaitaan tässä esimerkissä.

Kuvassa 34 on esitetty prosessiketju Asiakkaan kotiutus. Sairaala ilmoittaa puhelimitse kotipalvelun ohjaajalle, että kotipalvelun Asiakkaana oleva, sairaalassa intervallijaksolla ollut henkilö kotiutetaan. Kotipalvelunohjaaja merkitsee kotiutusajan kyseisen Asiakkaan omahoitajan paperiseen työvuorolistaan, joka toimitetaan kotipalvelun tiimitilaan. Kotiutuspäivänä omahoitaja ottaa Asiakkaan vastaan kotiin ja avustaa kotiutumisessa. Omahoitaja kirjaa käynnin Asiakkaan kotona olevaan viestivihkoon. Tässä heräte on sairaalan ilmoitus ja vaste on Asiakkaan turvallinen kotiutus. Nykyiseen käytäntöön voisi lisätä ilmoituksen sairaalaan kotiutuksen tapahduttua (kuvassa 34 esitetty katkoviivalla).



Kuva 34: Prosessiketju Asiakkaan kotiutus

Kotipalvelun käytössä ei tällä hetkellä ole ohjelmistoa lainkaan, joten suunniteltavaan kotihoitoa tukevaan ohjelmistoon löytyy tästä prosessikuvauksesta työvuorolistojen hallinta sekä käyntitietojen hallinta.

Lisäksi tässä automatisoitavana prosessina voi olla se, että sairaalan järjestelmän tekee ilmoituksen kotihoidon järjestelmään, ja kotiutusaika merkitään suoraan Asiakkaan omahoitajan työvuorolistaan ja asiakastietoihin. Vastaanottokotikäynnin kirjaaminen kotihoidon järjestelmään lähettää kuittauksen sairaalan järjestelmään. Tämä puolestaan edellyttää sitä, että sairaalan järjestelmä kommunikoi kotihoidon järjestel-

män kanssa. Lisäksi ongelmana on se, että kotiuttavia tahoja on monta sekä yksityiseltä että julkiselta sektorilta. Tällöin on varmistettava, että kaikki kotiutusilmoitukset voidaan hoitaa automaattisesti, toisin sanoen tarvitaan integrointia useaan ohjelmistoon.

Esimerkki 3:
Sama heräte voi tilanteesta riippuen saada aikaan erilaisia prosesseja. Asiakkaan kunnosta riippuen kotiutustilanteessa saatetaan tarvita sekä kotipalvelun että kotisairaanhoitajan läsnäoloa. Joskus Asiakas tarvitsee myös jonkin apuvälineen, esimerkiksi kyynärsauvat, tai Asiakkaan kotona joudutaan tekemään liikkumista helpottavia muutostöitä. Tällöin yksi heräte, eli sairaalan kotiuttamisilmoitus, saa aikaan useita rinnakkaisia prosesseja, jotka ovat suoraan yhteyksissä toisiinsa. Tarvitaan eri ammattilaisten yhteistyötä. Robertsonin menetelmä ei tarkastele rinnakkaisia prosesseja, eikä näiden välisiä yhteyksiä.

Polvileikkauspotilaan kotiutuksessa on seuraavan kaltaisia prosesseja: Sairaala ilmoittaa kotipalveluun kotiutuksesta. Kotipalvelusta otetaan yhteyttä kotisairaanhoitoon ja apuvälinelainaamoon. Kotipalvelun työntekijä noutaa sauvat apuvälinelainaamosta, jossa lainaus merkitään lainaamon tietojärjestelmään. Samalla mukaan tulee rakennusmies, jonka tehtävänä on poistaa Asiakkaan asunnosta kynnykset. Kotisairaanhoitaja ja kotipalvelun työntekijä avustavat Asiakasta kotiintulossa. Rakennusmies poistaa kynnykset. Sairaanhoitaja tarkistaa haavasiteen ja jakaa lääkkeet annostelulaitteeseen. Kodinhoitaja huolehtii, että Asiakkaalla on tarvittavat tavarat käden ulottuvilla ja valmistelee ilta-aterian valmiiksi jääkaappiin. Samalla keskustellaan jatkohoidosta ja huomataan, että WC:n seinälle kannattaa asentaa ylös nousemista helpottava kahva. Rakennusmiehellä on niitä pakettiautossaan mukana ja kahva asennetaan. Kodinhoitaja kirjaa käynnin Asiakkaan viestivihkoon, sairaanhoitaja kirjaa käynnin toimipisteessään omaan järjestelmäänsä ja rakennusmies ilmoittaa apuvälinelainaamoon asennetusta kahvasta.

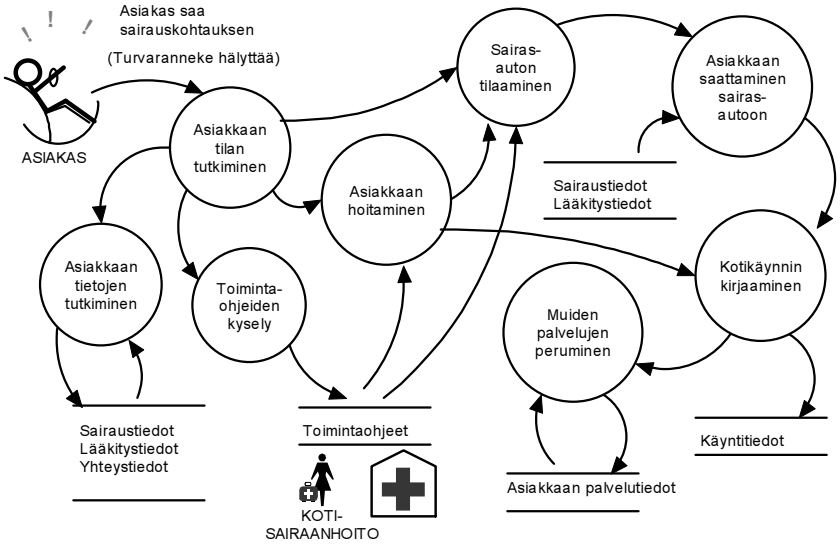
Jos systeemiä ollaan suunnittelemassa tehtäväkokonaisuuksien määrääminä palasina (kotipalvelu, kotisairaanhoito, apuvälinelainaus jne.), on jokaisen palasen yhteinen korkean tason vaste Asiakkaan turvallinen kotiutus ja tietojen kirjaaminen. Tarkalla tasolla kukin vaste sisältää erilaisia prosesseja, jotka ovat keskenään yhteyksissä muu-

tenkin kuin tietovaraston kautta. Tämänkaltaisen, tilanteen mukaan muuttuvan prosessiryppään kuvaaminen Robertsonin menetelmällä ei ole mahdollista.

Esimerkki 4:

Työn sisällä voi olla tapahtumia, jotka toimivat herätteenä ja käynnistävät prosessoinnin. Työ voi olla siis myös aktiivinen. Joskus paras mahdollinen vaste ei muodostu prosessiketjusta, vaan ryhmästä prosesseja, jotka voidaan suorittaa useassa eri järjestyksessä tilanteesta ja toimijoista riippuen.

Jos Asiakas saa sairauskohtauksen kotipalvelun työntekijän työkäynnin aikana, paras mahdollinen vaste riippuu kohtauksen laadusta ja vakavuudesta (tarvitaanko ambulanssia, elvytystä tai muuta hoitoa), työntekijän kokemuksesta ja koulutuksesta (tarvitseeko kysyä lisäohjeita esimerkiksi sairaanhoitajalta tai sairaalasta) sekä Asiakkaan luona olevien henkilöiden lukumäärästä (samalla kun toinen elvyttää, toinen voi tilata ambulanssin). Kuvassa 35 on esitetty sairauskohtauksen herättämiä prosesseja. Koska tässä notaatiossa ei voi esittää if-rakenteita, jokaisesta eri vaihtoehdosta tulisi piirtää oma kaavionsa. Saman herätteen aiheuttama vaste voi siis olla erilainen eri tilanteissa. Vaste voi koostua erilaisista prosesseista, jotka eivät muodosta ketjua, vaan verkon.



Kuva 35: Prosessiverkko Asiakas saa sairauskohtauksen

Tästä prosessikuvauksesta löytyy suunniteltavaan systeemiin asiakastietojen (sisältäen sairaus-, lääkintä-, yhteys- ja palvelutiedot) hallinta. Sairauskohtauksen sattuessa tar-

vitaan aina ihminen toimimaan. Sähköisessä muodossa oleva tietovarasto, josta työntekijä voi hakea toimintaohjeita olisi avuksi ja parantaisi kotihoidon joustavuutta ja laatua. Tällöin ei aina välttämättä tarvitsisi soittaa kenellekään ohjeiden saamiseksi. Jos Asiakkaalla on turvaranneke, hän voi sen avulla hälyttää itse apua. Turvarannekkeessa voi myös olla elintoimintoja seuraava laite, joka tunnistaa hälyttävät muutokset ja tilaa esimerkiksi sairasauton tai ennalta sovitun avustajan Asiakkaan luo.

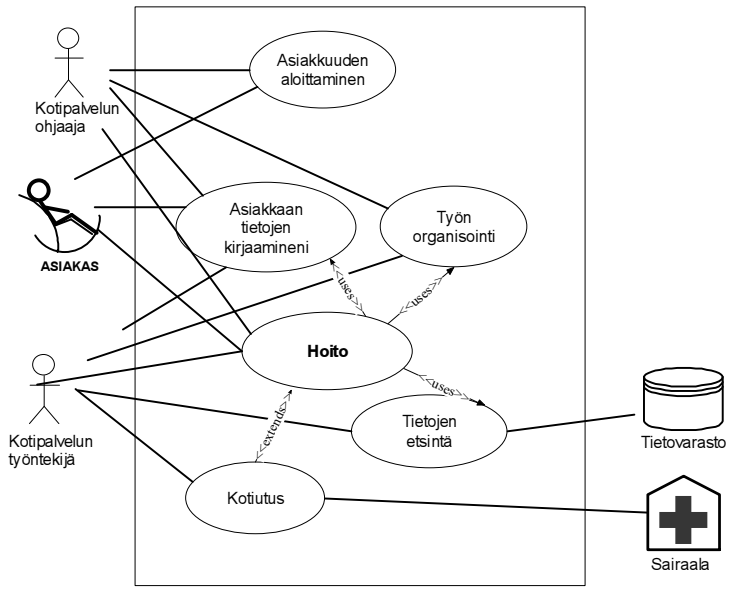
Työtoimintaa kehitetään prosesseja kehittämällä. Tässä kehityskohteena on esimerkiksi vastuujako eri vaiheissa ja tiedon siirtäminen kaikille, jotka tietoa tarvitsevat. On päätettävä esimerkiksi siitä, missä tapauksessa turvarannekehälytykseen liitetään automaattinen sairasauton tilaus ja onko kotipalvelun työntekijän vastuulla perua sairasauto, mikäli hälytys tapahtuu hänen kotikäyntinsä aikana ja Asiakas toipuu annetun ensiavun avulla.

6.2.4 Tuotteen rajaaminen ja tapahtumalähtöiset käyttötapaukset

Kun kaikista tapahtumalistalla olevista tapahtumista on mietitty niiden käynnistämät prosessiverkot, on päätettävä, mitkä prosesseista automatisoidaan. Toisin sanoen määritellään suunniteltavan tuotteen rajat. Prosessien automatisointia on pohdittu osin jo edellä esimerkeissä 1, 3 ja 4.

Kotihoidossa monet prosesseista ovat Asiakkaaseen ja tämän hyvinvointiin kohdistuvia käytännön töitä, esimerkiksi haavan puhdistus, aterian tarjoilu tai pukeutumisapu. Näitä ei luonnollisestikaan voida automatisoida. Usein tarvitaan ihmisen päätös ja tilannearvio ennen seuraavan prosessin käynnistämistä, esimerkiksi Asiakkaan sairauskohtauksen tapauksessa on ennen sairasauton tilaamista arvioitava Asiakkaan kunto ja tilanne. Selkeimmin automatisoitavissa ovat työn organisointiin ja hallintoon liittyvät prosessit, kuten työlistojen teko ja päivitys sekä laskutus. Kotihoidon Asiakkuuden aloittaminen ja Asiakastietojen hallinta ovat osittain automatisoitavia prosesseja. Automatisoinnissa on otettava huomioon eettiset kysymykset.
Yhteyksiä ulkopuolisiin organisaatioihin on paljon. Tämä aiheuttaa sen, että Kotihoito tulee integroida moneen eri ohjelmistoon, joista keskeisin on esimerkin mukaan erikoissairaanhoidon järjestelmä.

Kuvassa 36 on esitetty esimerkkejä tapahtumalähtöisistä käyttötapauksista. Näitä tarkentamalla saadaan kuhunkin käyttötapaukseen liittyvät toiminnalliset vaatimukset. Keskeisenä käyttötapauksena on Hoito, joka käyttää muita käyttötapauksia. Käyttötapaus Kotiutus laajentaa käyttötapausta Hoito. Hoitoa ei voida automatisoida, ja Kotiutuksestakin voidaan automatisoida vain osa. Kuva ei ole kovin informatiivinen, sillä siitä ei nähdä, mitä osia on hyödyllistä automatisoida. Myöskään kokonaiskuva Kotihoidosta monine osatöineen ei tule esille. Robertsonin menetelmästä puuttuu keino tarkastella osatöistä muodostunutta, useiden toimijoiden suorittamaa verkottunutta työtä.



Kuva 36: Esimerkkejä tapahtumalähtöisistä käyttötapauksista

6.3 Kokemuksia Robertsonin menetelmän soveltamisesta

Robertsonin menetelmä korostaa ohjelmistoasiakkaan työn ymmärtämisen merkitystä menestyksekkäässä ohjelmistosuunnittelussa. Työn mallinnuksessa käytetään lähtökohtana liiketoimintatapahtumia, joilla on työlle taloudellista merkitystä. Tapahtumista edetään asiakkaan, loppukäyttäjän ja muiden toimialan asiantuntijoiden avulla käyttötapauksiin, joiden avulla vaatimukset ilmaistaan, kuten yleensä muissakin ohjelmistosuunnittelun menetelmissä.

Tällöin käyttötapaukset eivät ole irrallisia, vaan liittyvät jäljitettävästi asiakkaan liiketoimintaan. Menetelmässä korostetaan myös erilaisten asianosaisten merkitystä tieto-

lähteenä käyttämällä esimerkiksi osallistuvaa havainnointia ja työpajoja tiedon etsinnässä. Kun asianosaisten kanssa mietitään työn parasta mahdollista vastetta herätteisiin, pyritään myös parantamaan prosesseja. Tavoitteena on siis myös työn kehittäminen, ei pelkästään nykyisten prosessien automatisointi. Tässä esimerkissä ei ole ollut mahdollista järjestää tapaamisia kotihoidon asianosaisten kanssa erikseen, vaan on edetty Kotihoito-projektissa saatujen kokemusten perusteella. Tämä tietysti vaikuttaa menetelmän arviointiin. Menetelmässä ei ole mitenkään mietitty valmiiksi komponenttiajattelun näkökulmaa. Tässä pohdintakappaleessa siihen kuitenkin otetaan vähän kantaa.

Työn rajaaminen oli esimerkissä aika ongelmallista. Todellisessa reaalimaailman tilanteessa rajanvetoon vaikuttaa voimakkaasti tulevan systeemin maksaja. Rajanvetoon vaikuttaa myös se, mitä näkökulmaa halutaan korostaa: paras mahdollinen palvelu Asiakkaalle vai tehokkain työajan käyttö. Jos suunnitellaan koko kotihoitoa palvelevaa systeemiä ja paloitellaan työ toimijaryhmien mukaan (kotipalvelu, kotisairaanhoito, turvapalvelu jne.) osasysteemeihin, tulee jokaisesta osasysteemistäkin melko laaja. Lisäksi on huomioitava näiden palojen integrointi lopullisessa kokonaissysteemissä ja edelleen kokonaissysteemin yhteydet ympäröiviin systeemeihin (esimerkiksi erikoissairaanhoito).

Jokaisessa osasysteemissä on paljon samoja tai samankaltaisia osia, esimerkiksi kotikäynnin kirjaaminen tai Asiakkaan tietojen katselu, mutta erilaisin käyttöoikeuksin. Esimerkiksi laki voi rajoittaa käyttöoikeutta: Asiakkaan sairaustietoja ei voi ilman Asiakkaan lupaa tutkia muut kuin terveydenhuollon palvelukseen kuuluvat työntekijät. Kotihoito kokonaisuutena on niin laaja, ettei kotihoidon systeemiä voi suunnitella yhtenä palana. Osiin jako tuottaa sirpaleista tietoa, jolloin kokonaiskuva hämärtyy. Volere-menetelmä ei tarjoa valmista mallia suurten kokonaisuuksien mallintamiseen ja hallintaan. Siksi menetelmän laajentaminen käsitteellä osatyö (part of Work) on tarpeellista.

Yhteyksiä ympäröiviin systeemeihin on hyvin paljon. Niistä on vaikea johtaa kotihoidon tapauksessa (liike)toimintatapahtumia, joilla kuitenkin on keskeisin merkitys tulevan systeemin suunnittelussa. Liiketoimintatapahtuma käsitteenä ja nimikkeenä ei

oikein hyvin sovellu kotihoitoon, sillä kotihoidossa on kyseessä inhimillisiin arvoihin perustuva hoivatyö, jonka arvoa ei aina voi mitata rahallisesti.

Yhteyskaaviosta löytyy kuitenkin työhön vaikuttavia tapahtumia, jotka esitetään tapahtumalistalla. Tapahtumalistalla tulee esittää omana tapahtumanaan esimerkiksi kotiutus jokaisesta eri laitoksesta erikseen. Käytännössä tämä on hyvin hankalaa toteuttaa, eikä edes tarkoituksenmukaista, sillä kotiuttamisessa tarvittavat tiedot ovat jokseenkin samat, mutta kotiutustietoja lähettävät järjestelmät eivät ole yhdenmukaiset, eli tiedon muoto vaihtelee. Tämä johtaa ajatukseen, että kotihoidon järjestelmää voisi ympäröidä ”kääre-komponentti”, joka tunnistaa ympäröivien järjestelmien viestit ja kääntää ne kotihoidon tarvitsemaan muotoon.

Kun pohditaan sitä, miten työ antaa parhaan mahdollisen vasteen ulkopuolelta tuleviin herätteisiin, ja vain niihin, ei näin löydetty systeemi luultavasti palvele kovin hyvin kotihoitoa. Suuri osa työhön vaikuttavista tapahtumista tapahtuu Asiakkaalle muuttaen hänen tilannettaan ja hoidon tarvettaan. Tällöin on ajateltava, että työ voi olla aktiivinen, eli liiketoimintatapahtumia voi tulla työn sisältäkin päin. Näitä tapahtumia ei välttämättä löydetä yhteyskaavion avulla. Joskus Asiakkaan tilanne muuttuu nopeasti (esimerkiksi sairauskohtaus), joskus hitaasti (esimerkiksi dementoituminen). Kotihoidon tulee pystyä vastaamaan molempiin ja vastaamaan myös silloin, kun Asiakas ei itse huomaa tilanteen muutosta.

Esimerkiksi kun Asiakas saa sairauskohtauksen, työn paras mahdollinen vaste on Asiakkaan asianmukainen hoito, joka tilanteesta riippuen voi olla erilainen. Lisäksi vasteeseen kuuluu tietojen kirjaaminen kotihoidon järjestelmään, josta ne ovat kaikkien muiden Asiakkaan kotihoitoon osallistuvien käytössä, muiden palveluiden peruminen ja tarvittavien tietojen luovuttaminen Asiakkaan hoitoa jatkavalle laitokselle, esimerkiksi sairaalalle. Työn sisäinen tapahtuma saa aikaan prosesseja, jotka tuottavat yhteyksiä ulkomaailmaan. Prosessit muodostavat rinnakkaisia toisiinsa yhteyksissä olevia prosessiketjuja ja viime kädessä prosessien verkon. Prosessiketjun valmistuminen edellyttää joissakin tapauksissa naapurisysteemin vastausta. Ulkopuoliset systeemit ovat keskenään yhteyksissä muutenkin kuin työn kautta, ja nämä yhteydet ja niissä kulkeva tieto saattavat vaikuttaa tehtävään työhön. Näiden monimutkaisten yhteyksi-

en ja vuorovaikutusten löytämiseen Robertsonin menetelmä ei tunnu olevan kovin tehokas.

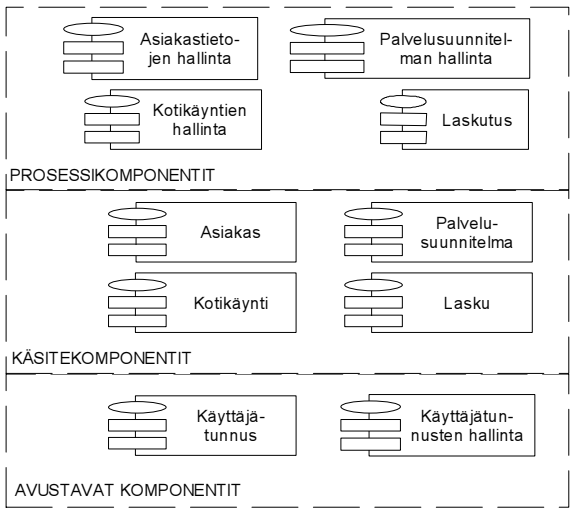
Kohdealueen perusongelmien ja luonteen määrittelyn ja tyypittämisen tulee ohjata tietojärjestelmän rakenteen, tietojen ja toiminnallisuuden suunnittelua. Kotihoidolle tyypillistä on hoitotyössä tarvittavan tiedon siirtäminen ihmiseltä toiselle, oikeaan aikaan, oikeaan paikkaan, yli organisaatiorajojen. Esimerkiksi, kun kotisairaanhoitaja käy hoitamassa Asiakkaan leikkaushaavan ja kirjaa sen omaan järjestelmäänsä, ei kotipalvelun välttämättä tarvitse reagoida siihen millään tavalla. Korkeintaan on hyvä, jos kaikilla toimijoilla on jonkinlainen yleiskuva Asiakkaan tilasta, niin että mahdolliset ongelmat voidaan havaita ennen kuin Asiakas niistä kärsii. Mutta jos kotisairaanhoitaja käynnillään muuttaa Asiakkaan lääkitysohjeita, on tiedon tästä oltava kotipalvelun työntekijän saatavilla, kun hän avustaa Asiakasta lääkkeenotossa. Voleressa on keskeistä tiedon kirjaaminen, kotihoidossa taas konkreettinen työ. Kotihoidossa tarvitaan paljon päätöksiä, joissa ihmisen on suoritettava arviointia päätöksen perusteeksi. Tarvitaan myös eri organisaatioista olevien työntekijöiden yhteistyötä ja kommunikointia.

Kotihoidon luonteen vuoksi on prosessien automatisointia harkittava hyvin tarkkaan esimerkiksi tietoturvan ja eettisten kysymysten kannalta. Kun mietitään, mitkä prosessit ovat automatisoitavissa, löydetään tietotekniikan potentiaalisia käyttöpaikkoja. Automatisointia voi olla lähinnä hallinnollisella puolella, esimerkiksi raporttien laadinnassa, laskutuksessa, tai vaikkapa laboratoriotulosten kirjaamisessa. Päätöksiä ja tilannearvioita tarvitsevien kriittisten tapahtumien hoidossa ei prosesseja voida automatisoida. Ei voida esimerkiksi automaattisesti lisätä lääkitystä ja hoitokäyntejä sairauskohtauksen jälkeen, vaan muutospäätösten tekemiseen tarvitaan aina ihminen. Päätösten tukena voi kuitenkin olla järjestelmä, jonka avulla tarpeelliset tiedot välittyvät oikeaan paikkaan. Esimerkiksi, kun Asiakas saa sairauskohtauksen, kotihoidon työntekijällä tarvitsee selkeät toimintaohjeet. Lisäksi Asiakkaan hoito- ja lääkitystiedot ovat heti sairasauton ja vastaanottavan sairaalan käytössä, kun hälytys sairauskohtauksen vuoksi on annettu. Volere-mallin tapahtumalistan avulla näitä tietovirtoja löydetään ja niiden tietosisältöjä täsmennetään esimerkiksi työpajoissa.

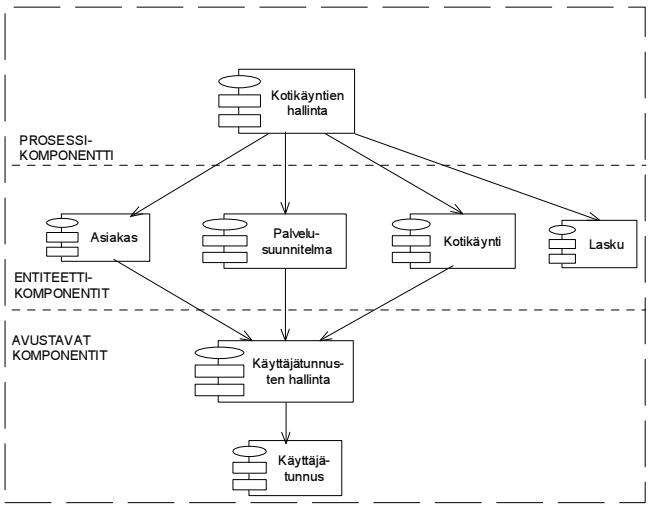
Kun analysointivaiheessa etsitään samankaltaisuuksia eri tehtävien välillä, löydetään komponenteiksi sopivia osia. Prosessiketjujen ja -ryppäiden mallintamisvaiheessa löydetään prosessikomponentteja ja niiden tarvitsemia käsitekomponentteja. Esimerkin 1 prosessiketjusta löytyvät prosessikomponentit

- *Asiakastietojen hallinta,* jonka avulla käsitellään asiakastietoja
- *Kotikäyntien hallinta*, joka hoitaa käyntitietojen kirjaamiseen ja katseluun liittyvät palvelut
- *Palvelusuunnitelmien hallinta*
- *Laskutus*

Lisäksi löytyvät käsitekomponentit *Asiakas, Kotikäynti, Palvelusuunnitelma, Lasku* sekä joukko esimerkiksi tietoturvaan liittyviä varustekomponentteja, kuten esimerkiksi käyttäjätunnukset ja niiden hallinta. Komponentit on esitetty Herzumin esittämän jaon mukaisesti kuvassa 37 ja kuvassa 38 on esitetty esimerkki komponenttien riippuvuusgraafista (depency graph). Riippuvuusgraafi kuvaa komponenttien välisiä riippuvuuksia.



Kuva 37: Kotihoidon liiketoimintakomponentteja



Kuva 38: Esimerkki komponenttien riippuvuusgraafista

Kokonaiskuvaksi muodostui, että kotihoito kohteena on liian laaja ja tuntematon, jotta Robertsonin menetelmää voitaisiin hyödyntää täysin tehokkaasti. Monimutkaiset työntekijäverkostot ja organisaatioiden moninaisuus tekevät kotihoidon mallintamisen tällä menetelmällä työlääksi. Jos eri organisaatioita tarkastellaan omina paloinaan, ei saada selkeää kuvaa näiden palojen yhteistoiminnasta, vaikka palojen prosessit löydettäisiin ja niiden avulla saataisiin mallinnettua osasysteemeitä. Tällöin jää huomaamatta esimerkiksi organisaatioiden väliseen työnjakoon tai -järjestelyyn liittyviä kysymyksiä, joiden ratkaiseminen saattaa parantaa kotihoidon prosesseja. Suunniteltava systeemi ei ehkä palvelisi kovin hyvin kotihoidon tavoitteena olevaa saumatonta palveluketjua, eikä siten Asiakkaan kokonaishyvinvointia.

7 Liiketoimintalähtöinen vaatimusmäärittely

Tässä luvussa tarkastellaan liiketoiminnan tavoitteita ja prosesseja ohjelmistotuotannon vaatimusmäärittelyn lähtökohtana. Kohdealueen liiketoiminnan suunnittelijat (business modellers) (ei-IT-ammattilaiset) pyrkivät kohdealueen mallintamisella esittämään liiketoiminnan rakenteen, tavoitteet ja visiot sekä löytämään liiketoiminnan kehityskohteita. Ohjelmistoarkkitehtuurin suunnittelijat (functional architects) taas ottavat kohdealuetta mallintaessaan ohjelmistotuotannon ja komponenttiajattelun koko ajan huomioon. Liiketoiminnan suunnittelijoiden malli otetaan yhtenä tietolähteenä huomioon ohjelmistosuunnittelussa, mutta se ei ole suunniteltavan systeemin ainut kohdealueen kuvaus, vaan lisäksi tarvitaan ohjelmistoarkkitehdin kuvaus kohdealueesta. Herzumin esittelemä Component-Based Business Modelling (CBBM) käyttää kohdealueen mallintamisessa perusosina liiketoiminnan prosesseja, käsitteitä, tapahtumia ja sääntöjä. [HeS00, s. 428-430]. Ongelmana on, että liiketoiminnan mallintamisessa ohjelmistoarkkitehti ja liiketoiminnan suunnittelija puhuvat samoilla sanoilla eri asioista. Toinen ongelma on, ettei liiketoiminnan suunnittelija ymmärrä riittävästi ohjelmistorakennetta eikä ohjelmistosuunnittelija ymmärrä riittävästi liiketoiminnan rakennetta. Puuttuu yhteinen ymmärrys kohdealueesta (ks. kappale 2.4, s.19 suunnitteluikkuna [GaL99]).

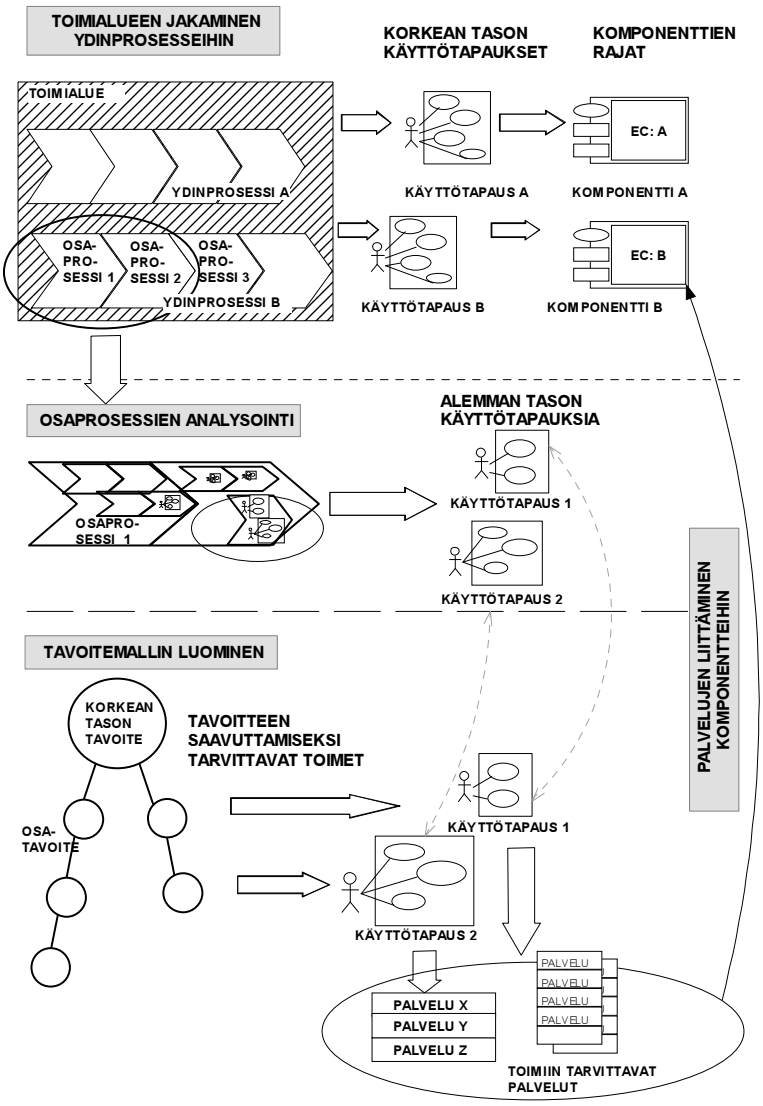
Käytännössä liiketoiminnan korkean tason visiot ja tavoitteet katoavat usein siinä vaiheessa, kun tekninen arkkitehtuuri luodaan [LeA02]. Ali Arsanjani ja Keith Levi ovat esitelleet *liiketoiminnan tavoitteista lähtevän vaatimusmäärittely- ja komponenttien suunnittelumenetelmän* [LeA02, Ars02]. Arsanjanin menetelmällä pyritään liiketoiminnan ja ohjelmistosuunnittelun välillä olevan kuilun poistamiseen. Suunnittelu aloitetaan liiketoiminnan tavoitteista ja prosesseista, ja edetään kohti komponentteja ja palveluja, jotka tarvitaan tavoitteiden toteuttamiseen. Tavoitteena on jäljitettävyys jokaisesta palvelusta takaisin liiketoiminnan tavoitteeseen, jolloin jokaisella toteutetulla palvelulla on liiketoiminnalle (taloudellista) merkitystä, eikä turhia palveluja toteuteta.

7.1 Arsanjanin menetelmän kuvaus

Kohdealueen liiketoiminnan arkkitehtuuri tulee mallintaa ennen ohjelmiston toiminnallisen arkkitehtuurin mallintamista. Liiketoiminnan malli ohjaa ohjelmiston rakenteen suunnittelua. Kohdealueen liiketoiminnan mallintamiseen tarvitaan kaksi analyysiä. Ensimmäisessä analyysissä kohdealue jaetaan pienempiin osiin aloittaen liiketoiminnan ydinprosesseista. *Ydinprosessilla* tarkoitetaan koko yrityksen läpi kulkevaa prosessia. Ohjelmistosuunnittelijat määrittelevät ydinprosessien avulla yritystason komponenttien (Enterprise Component, EC) rajat. Toinen analyysi aloitetaan tarkastelemalla liiketoiminnan tavoitteita. Analyysin avulla löydetään tavoitteiden toteuttamiseen tarvittavia palveluita, jotka ovat jäljitettävissä liiketoiminnan tavoitteisiin. Nämä kaksi analyysiä yhdistämällä liitetään palvelut oikeisiin komponentteihin.

Liiketoiminnan asiantuntijat ja mallintajat tekevät kohdealueen liiketoimintaa koskevat analyysit. Mallinnusvaiheessa tarvitaan tiivistä yhteydenpitoa ohjelmistotuotannon asiantuntijoiden ja liiketoiminnan asianosaisten kesken. Liiketoiminnan, joka on tässä tapauksessa ohjelmistosuunnittelun kohdealue, mallintamisessa käytetään komponenttipohjaisen ohjelmistosuunnitteluun hyvin sopivaa mallinnustapaa. Ohjelmistoarkkitehtuuri suunnitellaan vastaamaan liiketoiminnan arkkitehtuuria. Menetelmän vaiheet on esitetty kaaviona kuvassa 39.

Komponenttien kuvaamisessa ja sisäisen rakenteen suunnittelussa käytetään apuna suunnittelumalleja, jolloin komponenttien uudelleenkäytettävyys ja ymmärrettävyys paranevat. Suunnittelumallien avulla saadaan keskenään samoilla periaatteilla toimivia komponentteja, jotka toimivat hyvin yhdessä. Suunnittelu- ja rakennustyötä voidaan jakaa useille henkilöille. Kun suunnittelumalli on omaksuttu, neuvonnan osuus työajasta vähenee – ei tarvitse keksiä pyörää moneen kertaan. Suunnittelumalli toimii myös muistilistana siitä, että kaikki osakomponentit tulevat huomioiduksi.



Kuva 39: Menetelmän vaiheet kaaviona

7.1.1 Kohdealueen mallintaminen ja ohjelmistoarkkitehtuuri

Toimialue jaetaan ydinprosesseihin. Arsanjani ja Levi esittävät, että ydinprosessi voidaan ajatella myös korkean tason käyttötapauksena [LeA02]. Käyttötapausanalogia yhdistää liiketoiminnan ja ohjelmistoarkkitehtuurin mallintajat. Esimerkiksi nettikaupan ydinprosesseita voisivat olla TuotteenHallinta, AsiakasHallinta, TilauksenHallinta, VarastonHallinta ja TaloudenHallinta. Ydinprosessien rajat muodostavat rajat yritystason komponenteille (Enterprise Component, EC).

Kukin ydinprosessi jaetaan pienemmiksi osaprosesseiksi, usein yrityksen osastorajojen mukaan. Käyttötapaus-ajattelua jatketaan kuvaamalla osaprosessit pienemmiksi

käyttötapauksiksi, jotka tarkentavat alkuperäistä [LeA02]. Osaprosesseita tarkastelemalla löydetään liiketoimintakomponentit (Business Component, BC) [HeS00] ja osasysteemeitä (subsystems).

Ohjelmiston tulee toteuttaa ja tukea liiketoiminnan tavoitteita. Ohjelmiston sisältämien toimintojen tulee olla jäljitettävissä tavoitteisiin. Liiketoiminnan korkean tason tavoitteet voidaan Levin ja Arsanjanin mukaan yhdistää ohjelmiston ei-toiminnallisiin vaatimuksiin. Niiden toteuttamiseen tarvitaan tietty joukko toiminnallisia ominaisuuksia, jotka taas kuvataan käyttötapauksina ja toteuttaa ohjelmiston toimintoina. Tavoitemalli luodaan Goal-Service -graafin avulla. Tavoitteen saavuttamiseksi määritellään osatavoitteita, jotka saavutetaan tiettyjen konkreettisten toimien avulla. Toimet kuvataan käyttötapauksina, joiden toteuttamiseen tarvitaan tietyt palvelut (services). *Goal-Service -graafi* on puumainen rakenne, jonka juurena on korkean tason tavoite ja joka etenee osatavoitteiden ja toimien kautta lehdissä oleviin palveluihin. Tällöin kukin palvelu on jäljitettävissä jonkin liiketoiminnan tavoitteen toteuttamiseen.

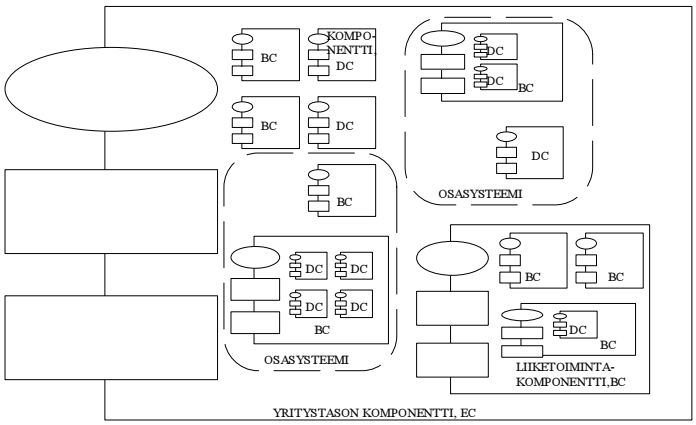
Esimerkiksi nettikaupan liiketoiminnan tavoite *lisätä myyntiä helpottamalla asiakkaiden ostotapahtumaa* voidaan yhdistää ohjelmiston ei-toiminnallisiin vaatimukseen *Asiakkaan on voitava löytää ja tilata helposti haluamansa tuote.* Tämän toteuttamiseen tarvitaan toiminnalliset ominaisuudet *tuoteluettelon selaaminen ja tuotehaku* sekä *tilauksen teko.* Kun nämä kuvataan käyttötapauksina, tarkentuvat tarvittavat palvelut, esimerkiksi käyttöliittymässä tarvittavat ominaisuudet ja toiminnot.

Prosessi- ja tavoiteanalyysien jälkeen verrataan Goal-Service -graafin avulla löydettyjä käyttötapauksia niihin, jotka löytyivät osaprosesseista. Kun löydetään molemmista analyyseistä sama käyttötapaus, se sijoitetaan prosessien perusteella määriteltyyn komponenttiin. Kaikki tavoitteiden toteuttamiseksi tarvittavat palvelut sijoitetaan. Jos osaprosessista löydetään käyttötapaus, jota ei Goal-Service -graafissa ole, ei käyttötapaus tue liiketoiminnan tavoitteita. Näin ollen se ei tuo lisäarvoa liiketoiminnalle, eikä sen toteuttaminen ohjelmistoon ole perusteltua. Kun liiketoiminnan rakenne on mallinnettu käyttäen näitä kahta analyysiä, saadaan liiketoiminnan tavoitteet kytkettyä ohjelmiston toteutukseen.

7.1.2 Komponenttien kuvaaminen ja suunnittelu

Yritystason komponentit määritellään rajapintojen (interfaces), sopimusten (contracts) ja toimintakuvausten (manners) avulla. *Rajapintamäärittelyissä* kuvataan komponentin tarjoamat palvelut ja riippuvuudet. *Palveluiden* määrittely kertoo, miten komponentti tukee liiketoiminnan tavoitteita ja prosesseja. *Sopimuksissa* määritellään ja kuvataan esi- ja jälkiehtojen avulla, mitä komponentti vaatii edellä mainittujen palveluiden tuottamiseen. Toimintokuvauksissa kuvataan sanallisesti komponentin kontekstista riippuva käyttäytyminen eli, miten komponentti toimii annetussa kontekstissa ja mitä sääntöjä toiminnot milloinkin noudattavat [LeA02]. Rajapinnat, sopimukset ja toimintakuvaukset näkyvät ulospäin. Komponentin sisäinen toteutus piilotetaan käyttäjältä.

Liiketoiminnan ydinprosessien mukaan muodostetuista yritystason komponenteista tulee isoja, hierarkkisesti monitasoisia kokonaisuuksia. Osaprosessien perusteella on löydetty liiketoimintakomponentteja sekä alemman tason komponentteja (DC, ks. luku 3 s. 23), jotka saattavat vielä ryhmittyä osasysteemeiksi yritystason komponentin sisällä (kuva 40).



Kuva 40: Yritystason komponentti on hierarkkinen.

Jotta monimutkainen kokonaisuus saadaan hallittua, käytetään yritystason komponentin sisäisen rakenteen suunniteluun suunnittelumallia Enterprise Component Compound Pattern [Ars02]. Sen mukaan suunnittelussa käytetään apuna viittä muuta suunnittelumallia: Julkisivu (Facade), Välittäjä (Mediator), Kooste (Composite), Sovitin (Adaptor) ja Rule Object [GaH98, Imm00 ja Ars01]. Lisäksi tarvitaan liiketoimintakomponentteja, jotka tuottavat analyysissä löydetyt palvelut.

Julkisivu tarjoaa yhden yhtenäisen rajapinnan liiketoiminnan prosesseihin linkittyneille alisysteemeille ja komponenteille yritystason komponentissa. Välittäjä huolehtii yritystason komponentin sisällä siitä, että palvelupyynnöt menevät oikeille liiketoimintakomponenteille ja alisysteemeille. Kooste-suunnittelumallin mukaisesti jokaiselle rakenteessa olevalle komponentille suunnitellaan yliluokka, joka toteuttaa yhteiset operaatiot. Kooste-mallin avulla käsitellään osakomponentit ja niistä koostuva komponentti hierarkkisesti. Sovitin huolehtii rajapintojen yhteensopivuudesta [Imm00]. Rule Object pitää yllä kulloinkin voimassa olevia liiketoiminnan sääntöjä [Ars02]. Käyttämällä näitä suunnittelumalleja saadaan yritystason komponentin sisäiset, liiketoimintakomponenttien väliset ja ulkoiset yhteydet hallittua.

Komponentin ulkopuolelle jätetään profiloija, joka huolehtii kulloinkin käyttöön tulevan komponentin sisällöstä ennalta määriteltyjen käyttäjä- tai käyttäjäryhmäprofiilien avulla. *Profiloijan* tehtävä on ajon aikana tai käynnistettäessä valita komponenteista kullekin käyttäjälle kuuluvat näkymät ja toiminnot. Kun käyttäjien profilointi tehdään komponentin ulkopuolelta, profiileja voidaan helposti lisätä tai muuttaa koskematta varsinaisiin yritystason komponentteihin. Näin mahdollistetaan erilaisten käyttäjäryhmien dynaaminen, turvallinen ja joustava järjestelmän käyttö. Profiloija voi olla muodoltaan komponentti tai esimerkiksi yksinkertainen XML-tiedosto.

7.2 Esimerkki Arsanjanin menetelmän soveltamisesta

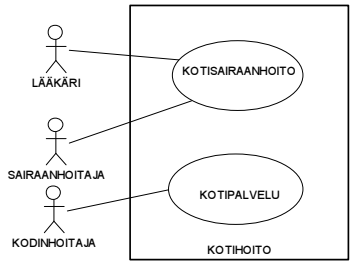
Kotihoitoon osallistuu useita toimijoita. Tärkeimmät selkeästi erotettavat osa-alueet ovat julkisen sektorin tarjoamat *kotisairaanhoito*, jonka tehtävänä on antaa asiakkaalle tarvittava sairaanhoidollinen palvelu, ja *kotipalvelu*, jonka tehtävänä on auttaa asiakasta selviämään päivittäisistä toimista. Muita kotihoidon toimijoita ovat esimerkiksi turvapuhelinpalvelu, ateriapalvelu, kuljetuspalvelu ja yksityiset kotipalvelun tuottajat.

Kotihoito eroaa liiketoiminnasta siinä, että kotihoidossa ei ole kyse yhden yrityksen sisällä tapahtuvasta prosessoinnista, vaan toimijoita, yrityksiä ja organisaatioita on monta. Kaikkien toiminta kohdistuu kuitenkin asiakkaaseen, joka asuu yhdessä kunnassa ja siksi asuinkunta voidaan ajatella ’yritykseksi’ ja eri organisaatiot ’osastoiksi’. Koko kotihoito prosessina elää asiakkaan hoitotarpeen ja palveluiden tarjoajien käytettävissä olevien resurssien mukaan. Esimerkiksi, kun asiakkaan kunto huonontuu tai

kun saadaan käyttöön uusi turvalaite tai apuväline, tehdään muutoksia asiakkaan kotihoitoprosessiin.

7.2.1 Kotihoidon prosessit

Kun aletaan suunnitella tietojärjestelmää kotihoidon tarpeisiin soveltamalla Arsanjanin menetelmää, voidaan ajatella (tässä esimerkissä lyhyyden vuoksi) kotihoidon keskeisimmiksi ydinprosesseiksi kotisairaanhoito ja kotipalvelu, jotka voidaan artikkelissa esitetyn ajatustavan mukaan mieltää korkean tason käyttötapauksina [LeA02] (kuva 41). Tällöin luonteviksi yritystason komponenteiksi (EC) tulisivat Kotisairaanhoito ja Kotipalvelu.

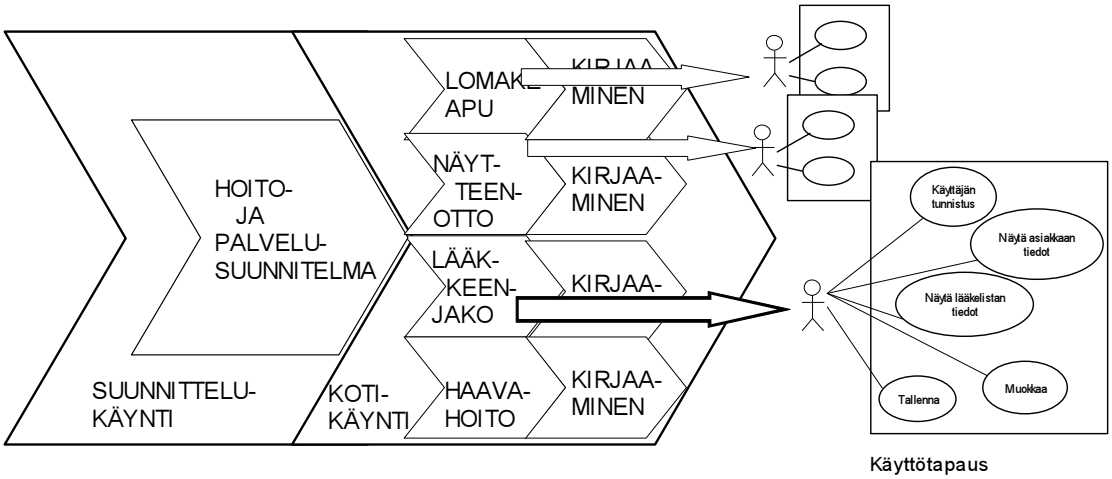


Kuva 41: Kotihoidon ydinprosessit

Kotisairaanhoidon osaprosesseita ovat hallinnolliset tehtävät, kuten suunnittelu ja kotikäynnit, jotka koostuvat edelleen esimerkiksi haavahoidosta, näytteiden otosta ja tulosten tulkinnasta sekä lääkelistan tarkistuksesta ja lääkkeiden jaosta. Kotipalvelussa osaprosesseina ovat esimerkiksi aterian tarjoilu, pukeutumisapu, lääkkeiden antaminen. Kentällä käytännön työssä myös kotisairaanhoitajat auttavat asiakasta päivittäisissä toimissa ja erilaisten lomakkeiden ja hakemusten täytössä. Kotipalvelun työntekijöiden osallistumista sairaanhoidollisiin toimenpiteisiin rajoittaa koulutustaso. Lakiin perustuva tietosuoja rajoittaa eri käyttäjäryhmien pääsyä katselemaan tai muokkaamaan asiakkaan tietoja, esimerkiksi sairauskertomuksia, laboratoriotuloksia tai lääkitystietoja. asiakkaan luvalla joitakin käyttöoikeuksia voidaan kuitenkin muuttaa. asiakkaan tietoja voi olla tallennettuna eri sairaaloiden tai terveyskeskuksen eri järjestelmiin sekä paperilla kotona.

Tässä esimerkissä on lyhyyden vuoksi rajusti yksinkertaistettu kotihoidon prosesseja, eikä oteta kantaa terveydenhuollon järjestelmien moninaisuuteen tai niiden keskinäiseen yhteensopivuuteen, vaan oletetaan, että kaikki tiedot löytyvät keskitetysti yhdellä yhteydellä asiakkaan kotoa toimijan mobiililaitteella. Kuvassa 42 on esitetty kotisairaanhoidon osaprosesseja. Kotisairaanhoito koostuu osaprosesseista suunnittelukäynti ja kotikäynti. Suunnittelukäynnin sisälle kuuluu osaprosessi palvelusuunnitelman laatiminen ja kotikäynnin sisälle osaprosessit haavahoito, näytteenotto, lääkkeenjako ja lomakeapu. Kotikäynnin kaikkiin osaprosesseihin kuuluu tietojen kirjaaminen käynnin jälkeen.

Liiketoimintakomponentteja (BC) voisivat siis olla Suunnittelu, jonka sisällä olisi pienempi BC Palvelusuunnitelma, sekä Kotikäynti, jonka sisällä olisivat edelleen pienemmät BC:t Haavahoito, Näytteet ja Lääkelista.



Kuva 42: Kotisairaanhoitoprosessi, osaprosesseja ja niistä johdettuja käyttötapauksia

7.2.2 Kotihoidon tavoitteet

Vanhusten kotihoidon korkean tason tavoite (visio) esimerkiksi Kuopiossa Kuopion sosiaali- ja terveyspalvelujen vanhustenhuollossa v 1999 – 2010 on

> **Kotona asuva vanhus elää omaehtoista ja virikkeellistä elämää omaisten ja muiden läheistensä avulla, yhdistysten, järjestöjen ja yksityisten yritysten palveluja käyttäen ja tarvittaessa tukeutumalla kotihoidon palveluihin.** [www07]

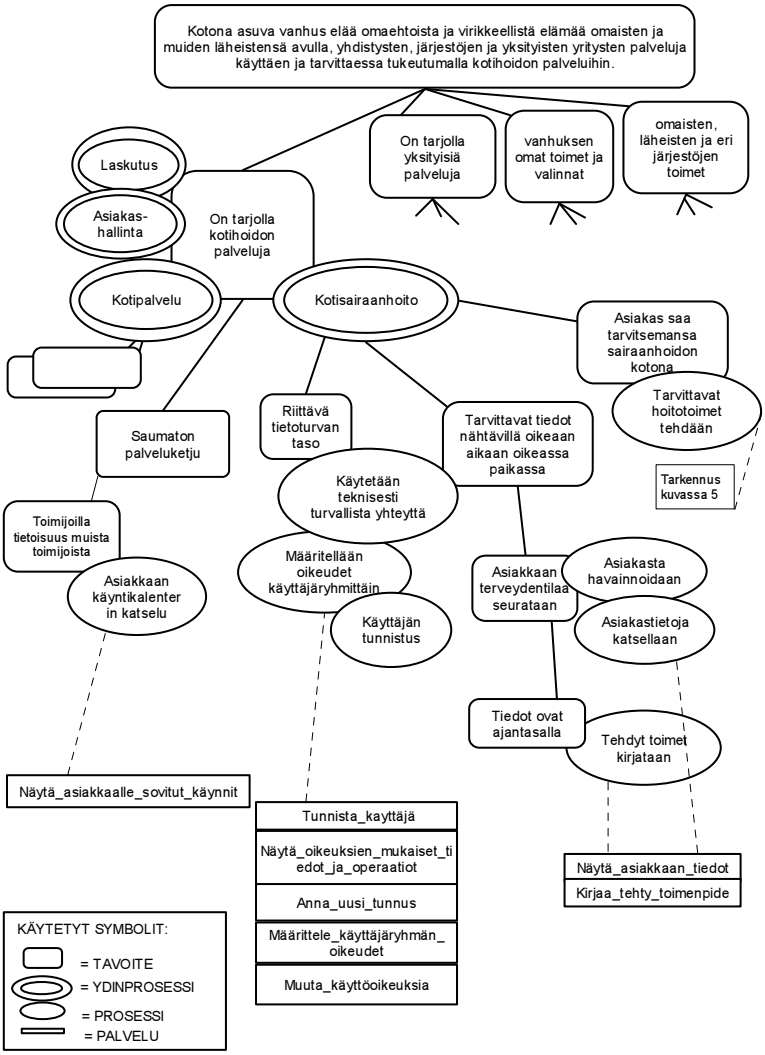
Kotihoidon tavoitteena on saumaton palveluketju ja tehokas yhteistoiminta. Tämä edellyttää, että eri toimijoilla on tietoisuus toisista saman asiakkaan luona käyvistä toimijoista. Kotihoidon toimijoiden haastatteluissa löytyneitä kotihoidolle asetettuja tavoitteita ovat asiakkaan selviäminen kotona mahdollisimman pitkään ja laitoshoidon lykkääminen sekä mahdollisimman hyvän ja kattavan palvelun antaminen. Nämäkin ovat korkean tason tavoitteita ja niitä pitää tarkentaa. Tarkennetut tavoitteet kuvaavat konkreettisemmin, mitä on tehtävä, jotta asiakas selviytyy kotona asumisesta: Asiakkaan terveydentilaa ja kuntoa seurataan riittävän hyvin. Asiakas saa tarvitsemansa avun päivittäisiin toimiin kotona. Apua annetaan esimerkiksi pukeutumisessa, aterioinnissa, hygieniassa ja lääkkeiden otossa. Asiakas saa tarvitsemansa sairaanhoitoon liittyvät palvelut kotona, mikäli ne eivät edellytä sairaalassaoloa. Näitä palveluja ovat esimerkiksi pistoshoidot, näytteiden otto, haavahoito, lääkityksen tarkistaminen. Terveys- ja lääkitystiedot pidetään ajan tasalla ja asiaankuuluvat toimijat voivat nähdä ja päivittää niitä aina siellä, missä tarvitaan. Huolehditaan riittävästä tietoturvan tasosta. Eri toimijoilla on erilaiset oikeudet nähdä, kirjata ja muokata asiakkaan terveydentilaan liittyviä tietoja (esimerkiksi koetulokset, lääkelista, sairauskertomus). Asiakkaan luvalla näitä oikeuksia voidaan muuttaa.

Kuvissa 43 ja 44 on esitetty osia Goal-Service -graafista (GSG), joka muodostuu edellä mainituista tavoitteista ja niiden toteuttamiseksi tarvittavista palveluista. Kuvassa 43 puurakenteen juurena on korkean tason tavoite, joka tarkentuu osatavoitteiksi. Lisäksi on kuvattu tavoitteiden toteuttamiseen tarvittavia prosesseja ja prosessien tarvitsemia palveluja. Tavoitteet on kuvattu pyöreäkulmaisilla suorakaiteilla, ydinprosessit kaksinkertaisilla soikioilla, muut prosessit yksinkertaisilla soikioilla ja palvelut kapeilla suorakaiteilla. Kuvassa 44 on esitetty tarkennus kuvaan 43 sekä esimerkki käyttötapauksesta, joka on johdettu graafista.

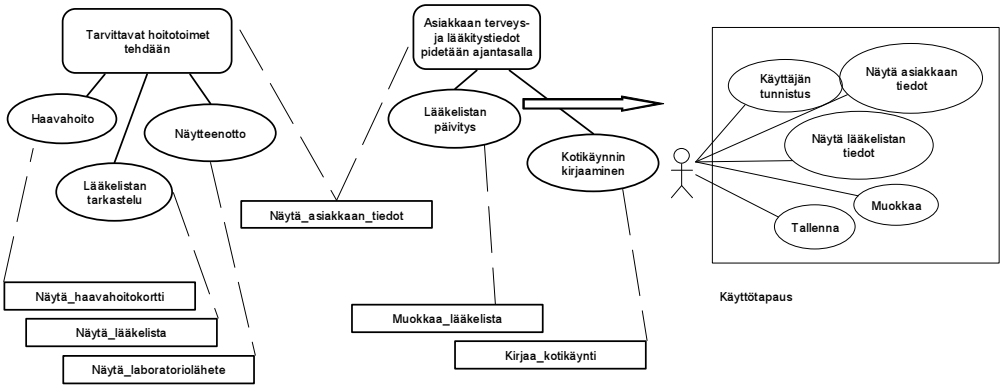
Seuraavassa on esitetty samat tavoitteet ja niiden saavuttamiseksi tietojärjestelmältä vaadittavia palveluita hierarkkisena puurakenteena. Esimerkin tarkoituksena on luoda mielikuva siitä, millaisia tavoitteet, toiminnot ja palvelut voivat olla, eikä se pyri olemaan kattava kuvaus kotihoidon tarvitsemasta järjestelmästä. **Tavoitteet** on kirjoitettu **vahvennettuna**, niiden saavuttamiseksi tarvittavat KONKREETTISET TOIMINNOT on kirjoitettu ISOILLA KIRJAIMILLA ja *tietojärjestelmän palvelut* on kirjoitettu *viistotettuna*.

1. **Saumaton palveluketju ja tehokas yhteistoiminta**
    a. **Eri toimijoilla on tietoisuus toisista saman asiakkaan luona käyvistä toimijoista** => ASIAKKAAN KÄYNTIKALENTERIN KATSELU
        i. *Näytä_Asiakkaalle_sovitut_käynnit*
        jne…

2. **Asiakkaan selviäminen kotona mahdollisimman pitkään ja laitoshoidon lykkääminen antamalla mahdollisimman hyvä ja kattava palvelu**
    a. **Asiakkaan terveydentilaa ja kuntoa seurataan riittävän hyvin** => ASIAKKAAN TUTKIMINEN JA HAVAINNOINTI, ASIAKKAAN TERVEYSTIETOJEN KATSELU JÄRJESTELMÄSTÄ
        i. *Näytä_asiakkaan_tiedot*
    b. **Asiakas saa tarvitsemansa avun päivittäisiin toimiin kotona** => TEE KOTIKÄYNTI
        i. pukeutumisessa => PUE
        ii. aterioinnissa => SYÖTÄ
        iii. hygieniassa => AUTA VESSAAN TAI PESULLE
        iv. lääkkeiden otossa => ANNA LÄÄKE ASIAKKAALLE
            1. *Näytä_lääkelista*
    c. **Asiakas saa tarvitsemansa sairaanhoitoon liittyvät palvelut kotona, mikäli ne eivät edellytä sairaalassaoloa**
        i. pistoshoidot => ANNA PISTOS
        ii. näytteiden otto => OTA NÄYTE
            1. *Näytä_laboratoriolähete*
        iii. haavahoito => HOIDA HAAVA
            1. *Näytä_haavahoitokortti*
        iv. lääkityksen tarkistaminen => TUTKI LÄÄKELISTA, TUTKI ASIAKAS, MUUTA LÄÄKELISTAA TARVITTAESSA
            1. Näytä_lääkelista
            2. *Muokkaa_lääkelista*
    d. **Asiaankuuluvat toimijat voivat nähdä ja päivittää tietoja aina siellä missä niitä tarvitaan**
        i. **Terveystiedot pidetään ajan tasalla** => KÄYTETÄÄN TEKNISESTI TURVALLISTA YHTEYTTÄ, KATSELE ASIAKKAAN TIETOJA, KIRJAA TARVITTAVAT TIEDOT, KÄYTTÄJÄN TUNNISTUS, OIKEUKSIEN MUKAISTEN TIETOJEN JA OPERAATIOIDEN NÄYTTÄMINEN
            1. Näytä_asiakkaan_tiedot
            2. Kirjaa_tehdyt_toimenpiteet
            3. Kirjaa_kotikäynti
            4. Tunnista_käyttäjä
            5. *Näytä oikeuksien__mukaiset_tiedot_ja_operaatiot*
    e. **Eri toimijoilla on erilaiset oikeudet nähdä, kirjata ja muokata asiakkaan terveydentilaan liittyviä tietoja. asiakkaan luvalla oikeuksia voidaan muuttaa** (esim. koetulokset, lääkelista, sairauskertomus) => ANNA TOIMIJALLE TUNNUS, MÄÄRITTELE TOIMIJARYHMÄN OIKEUDET, MUUTA TIETYN TOIMIJAN OIKEUKSIA SUHTEESSA TIETTYYN ASIAKKAASEEN,
        i. Anna_uusi_tunnus
        ii. Määrittele_käyttöoikeudet
        iii. Muuta_käyttöoikeuksia



Kuva 43: Tavoitemalli (Goal-Service -graafi), jossa sekä karkeajakoisia että hienojakoisia palveluita



Kuva 44: Tavoitemalli, jossa palvelut ovat hienojakoisia ja esimerkki graafiin liittyvästä käyttötapauksesta

Tässä esimerkissä on keskitytty vain Kotisairaanhoito-komponenttiin. Muita komponentteja ovat esimerkiksi Kotipalvelu, Asiakashallinta ja Laskutus. Esimerkissä

löydetyt palvelut sijoittuvat siis Kotisairaanhoito-komponenttiin lukuun ottamatta palveluja *Anna_uusi_tunnus, Määrittele_käyttöoikeudet* ja *Muuta_käyttöoikeuksia,* jotka kuuluvat Profiloija-komponenttiin

Tässä vaiheessa kiinnitetään myös huomiota löydettyjen palvelujen rakeisuuteen: palvelut voivat koostua toisista pienemmistä palveluista ja eri konteksteissa samalla palvelulla voi olla erilainen sisältö. Esimerkiksi korkean rakeisuuden palvelu *Näytä_oikeuksien_mukaiset_tiedot_ja_operaatiot* sisältää pienirakeiset palvelut *Näytä_sairauskertomukset, Näytä_lääkelista, Näytä_laboratoriotulokset, Kirjaa_kotikäynti,* kun käyttäjäksi on tunnistettu **kotisairaanhoitaja**. Jos käyttäjäksi on tunnistettu **lääkäri**, edellisten palveluiden lisäksi on myös esimerkiksi palvelu *Muokkaa_lääkelistaa*. Rajapinnassa näytetään komponentista ulospäin vain suurirakeisia palveluja. Tällöin rajapinnat ovat selkeitä.

7.2.3 Yritystason komponentin määrittelyt

Määrittely koostuu kolmesta osiosta: palvelut (rajapinnat), sopimukset ja toiminta. Palveluiden määrittely kertoo, miten komponentti tukee liiketoiminnan tavoitteita ja prosesseja. Sopimuksissa määritellään esi- ja jälkiehtojen avulla, mitä komponentti vaatii edellä mainittujen palveluiden tuottamiseen. Toimintokuvauksissa kuvataan sanallisesti komponentin kontekstista riippuva käyttäytyminen eli, miten komponentti toimii annetussa kontekstissa ja mitä sääntöjä toiminnot milloinkin noudattavat.

Seuraavassa on esitetty esimerkinomaisesti osa edellä mainitusta määrittelystä Kotisairaanhoito-komponentin osalta. Kohdassa toiminta on kuvattu myös profiloijan toiminta käytön aloittamisen yhteydessä. Profiloijan tehtävänä on valita kullekin käyttäjälle tämän käyttöoikeuksien mukaiset näkymät tai toiminnot. *Rooli* on käyttäjän attribuutti ja tarkoittaa käyttäjän ammatin tai aseman mukaan käyttäjälle annettavaa nimikettä. Käyttäjän rooli voi olla esimerkiksi lääkäri, kotisairaanhoitaja tai kotipalvelunohjaaja. Käyttäjällä voi olla useampi kuin yksi rooli yhtä aikaa, esimerkiksi kotisairaanhoitaja ja kotipalvelunohjaaja. *Roolityyppissä* määritellään eri rooleille ohjelmistoon liittyvät käyttöoikeudet. Esimerkiksi Kotisairaanhoitaja-roolityypin käyttöoikeuksiin kuuluu asiakkaan hoitotietojen katselu ja kotikäyntitietojen katselu ja kirjaaminen.

Rajapinta: Oikeuksien_mukaiset_tiedot (käyttäjän_tunnus, roolityyppi)
Esiehdot: Käyttäjä on avannut yhteyden.
Jälkiehdot: Järjestelmään tallentuu muutetut ja lisätyt tiedot oikein.
Toiminta: Käyttökerran alussa yhteyden muodostamisen jälkeen järjestelmä kysyy käyttäjätunnusta ja salasanaa. Käyttäjä syöttää henkilökohtaisen käyttäjätunnuksensa ja salasanansa niille varattuihin kenttiin. Profiloija tunnistaa käyttäjän, hakee tämän roolin mukaiset oikeudet ja palauttaa käyttäjän tunnisteen ja roolityypin. Kotihoito-komponentti ottaa ne syötteenä vastaan ja avaa oikeuksien mukaisen näkymän.
Vaihtoehtoinen toiminta:
Tunnus tai salasana väärin, jolloin järjestelmä kysyy korkeintaan kaksi kertaa uudestaan. Mikäli tiedot vieläkin väärin, yhteys katkeaa automaattisesti.

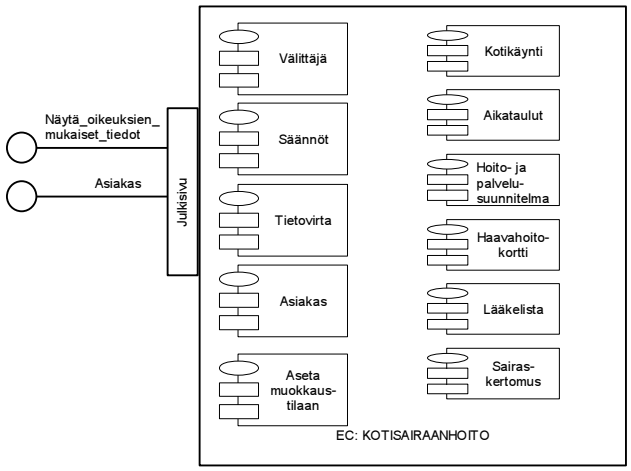
Jos kyseessä on esimerkiksi roolityyppi lääkäri, Kotihoito-komponentista otetaan käyttöön kaikki osakomponentit. Jos käyttäjä on kotipalvelunohjaaja, käyttöön otetaan osakomponentit Hoito- ja palvelusuunnitelma, Kotikäynti ja Aseta Muokkaustilaan. Sairaanhoitajan näkymässä on mahdollisuus hakea asiakkaan sairauskertomus, laboratoriotiedot, kotikäyntikertomukset ja -aikataulut ja lääketiedot näkyville sekä kotikäynnin kirjausoikeus.

Esimerkiksi lääkelistan katselu ja muuttaminen tapahtuu seuraavasti:

> Käyttäjä hakee asiakkaan nimen avulla asiakkaan tiedot ja valitsee lääkelistan. Käyttäjä näkee lääkelistasta asiakkaan nykyiset lääkitystiedot (lääkkeen nimi, annostus, kuka määrännyt, milloin määrätty). Käyttäjä valitsee Muokkaatoiminnon, jolloin vanha lääkelista kopioituu uudeksi kappaleeksi muokattavaan tilaan, jolloin käyttäjä voi lisätä uuden lääkkeen tai muuttaa entisten lääkkeiden annostusta. Muuttuneiden tietojen kohdalle tulee automaattisesti käyttäjän nimi ja muutospäivämäärä. Vanha lääkelista jää 'arkistoon', josta sitä voi selata.

7.2.4 Yritystason komponentin sisäisen rakenteen suunnittelu

Kotisairaanhoito-komponentti suunnitellaan käyttäen suunnittelumallia Enterprise Component compound pattern [Ars02]. Kuvassa 45 on esitetty karkea hahmotelma Kotisairaanhoito-komponentista. Kuvan jälkeen on esitelty pääpiirteittäin kuvan osakomponenttien tehtävät.



Kuva 45: Kotisairaanhoito-komponentti

Rajapinta Näytä_oikeuksien_mukaiset_tiedot:
Saa syötteenä profiloijalta käyttäjän oikeudet, ja palauttaa oikeuksien mukaisen näkymän järjestelmään.

Rajapinta Asiakas:
Tätä kautta haetaan asiakkaan tietoja muista järjestelmistä.

Julkisivu: Julkisivu huolehtii yhtenäisestä rajapinnasta

Välittäjä: Poimii oikeat komponentit, jotka tarvitaan kunkin käyttäjän näkymään.

Säännöt: Sisältää toimintaa ohjaavat säännöt, esimerkiksi tieto siitä, että asiakas on antanut kotipalvelun toimijoille luvan tarkastella sairauskertomusta.

Tietovirta: Huolehtii tietovirran oikeasta muodosta, joka on erilainen esimerkiksi, kun otetaan yhteyttä mobiililaitteella tai tavallisella päätelaitteella.

Kotikäynti: Sisältää kotihoidon Kotikäynti-näkymän ja tarvittavat toiminnot. Hakee asiakaskohtaisia tietoja eri järjestelmistä.

Asiakas: Sisältää asiakaskohtaisen näkymän ja toiminnot asiakkaan kotihoitopalveluista. Hakee perustiedot (nimi, syntymäaika, yhteystiedot) esimerkiksi terveyskeskuksen järjestelmän asiakastiedoista.

Aseta muokkaustilaan:
Asettaa käyttäjän oikeuksien mukaiset näkymät muokkaustilaan (aktivoi sallitut toiminnot), jolloin käyttäjä voi lisätä tai muokata tietoja. Esimerkiksi lääkärin näkymässä Lääkelista on muokkaustilassa, mutta sairaanhoitaja voi vain katsella sitä.

Lääkelista: Sisältää Lääkelista-näkymän, jolla on näkyvissä asiakkaan sen hetkinen lääkitys sekä lääkityshistoria, ja Lääkelistaan liittyvät toiminnot.

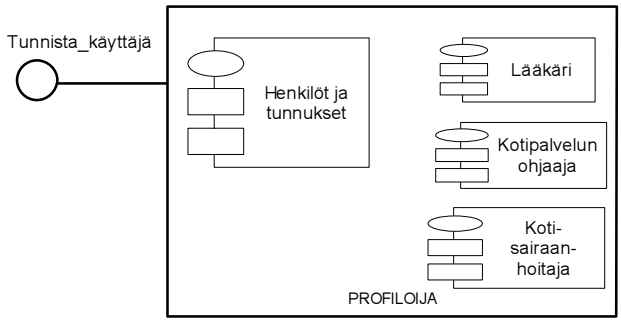
Hoito- ja palvelusuunnitelma:
Sisältää näkymän Hoito- ja palvelusuunnitelma, joka sisältää tiedot asiakkaan suunnitellusta hoidon tavoitteesta, hoidosta, tarvittavista apuvälineistä, käyntikerroista sekä mahdollisen seurannan, sekä Hoito- ja palvelusuunnitelmaan liittyvät toiminnot.

Haavahoitokortti:
Sisältää näkymän Haavahoito-ohje ja -seuranta sekä siihen liittyvät toiminnot.

Sairauskertomus:
Hakee asiakkaan nimen perusteella asiakkaan sairauskertomukset sairaalan ja terveyskeskuksen järjestelmistä.

Erilaisiin näkymiin liittyviä toimintoja ovat esimerkiksi tietojen lisäys ja muokkaus, erilaisten tilastojen tai kalenterien laatiminen ja tietojen yhdistely tarvittaviksi kokonaisuuksiksi.

Kotihoidon toimijat tulevat monista eri organisaatioista ja heillä on erilaiset tehtävät ja erilaiset käyttöoikeudet järjestelmään. Kuitenkin osa tiedoista on samoja. Profiloijan tehtävä on ajon aikana tai käynnistettäessä valita komponenteista kullekin käyttäjälle kuuluvat näkymät ja toiminnot. Kun käyttäjien profilointi tehdään komponentin ulkopuolelta, profiileja voidaan helposti lisätä tai muuttaa koskematta varsinaisiin yritystason komponentteihin. Profilointi on esitetty kuvassa 46 Profiloija-komponentin avulla.



Kuva 46: Profiloija-komponentti

Profiloijan eri osien tehtävät ovat:

Henkilöt ja tunnukset:
Huolehtii käyttäjien käyttäjätunnuksista, rooleista ja salasanoista.
Lääkäri Sisältää tiedot roolityypin Lääkäri käyttöoikeuksista.
Kotipalvelunohjaaja:
Sisältää tiedot roolityypin Kotipalvelunohjaaja käyttöoikeuksista.
Kotisairaanhoitaja:
Sisältää tiedot roolityypin Sairaanhoitaja käyttöoikeuksista.

7.3 Arsanjanin menetelmän arviointi

Arsanjanin menetelmässä on hyvää tavoitteiden ja palveluiden yhdistäminen. Kun toteutettavat palvelut ovat jäljitettävissä liiketoiminnan tavoitteisiin, varmistetaan, että sovellus luo yrityksen liiketoiminnalle lisäarvoa. Kotihoito on hyvin selkeästi tavoitteellista toimintaa. Tutkimalla tavoitteita löydetään tarvittavia palveluita ja ohjelmis-

tolta vaadittavia toimintoja. Kotihoidon ongelma-alue moninaisuudessaan poikkeaa kuitenkin kovasti liiketoiminnasta. Goal-Service -graafin laatiminen oli vaikeaa, koska erilaisia toimijoita on paljon, ja heillä on osin yhteiset tavoitteet ja tiedot, osittain eri. Graafista tuntuu tulevan helposti verkko. Menetelmän käyttäminen suoraan on hankalaa, ja oma soveltaminen on välttämätöntä. Tällöin joutuu miettimään, häviääkö tai muuttuuko alkuperäinen ajatus ja menetelmällä saavutettavat hyödyt.

Menetelmällä saavutettu ohjelmiston dynaamisuus ja skaalautuvuus (ks. kappale 3.3, s. 39) on hyvä. Ohjelmisto mukautuu käyttäjän oikeuksien ja käytössä olevan laitteen käyttöliittymän mukaan ja käyttäjän oikeuksien muuttaminen käy helposti. Tämä saadaan aikaan käyttämällä kuvattua suunnittelumallia komponentin sisäisen rakenteen suunnittelussa ja ulkoistamalla profilointi. Kotihoidossa ei toistaiseksi ole käytettävissä yhtenäistä tietojärjestelmää, joka olisi kaikkien toimijoiden käytössä. Toimijat tulevat monista eri organisaatioista ja heillä on erilaiset oikeudet. Eri organisaatioilla on valmiina omia sovelluksiaan, joihin on tallennettuna tietoja asiakkaasta. Tietojen tulee olla saatavana oikeaan aikaan oikeassa paikassa. Tähän tarvitaan erilaisia laitteita ja käyttöliittymiä. Artikkeleissa kuvatun kaltainen komponenttirakenne on hyvä lähtökohta suunniteltaessa tietojärjestelmää kotihoidon kentälle.

Komponenttisuunnittelussa suunnittelumalli Enterprise Component compound pattern tarjoaa hyvän mallin siitä, mitä osia yritystason komponenttiin tarvitaan (ks. kappale 3.3 s. 36). Kun käytetään suunnittelumallia, voidaan ohjelmistokehitystyötä jakaa eri suunnittelijoille, jotka voivat samanaikaisesti rinnakkain rakentaa järjestelmän eri osia. Tällöin markkinoilletuloaika pienenee, mutta silti voidaan varmistaa, että eri ihmisten tekemät osat toimivat keskenään yhteen.

Tämän menetelmän suunnittelumallit eivät tarjoa apua liiketoimintakomponenttien tunnistamiselle, tai niiden sisäisen rakenteen suunnittelulle. Liiketoimintakomponenttien tunnistamisessa voidaan käyttää apuna esimerkiksi Herzumin jakoa prosessi-, entiteetti- ja varustekomponenteiksi. Tässä esimerkissä liiketoimintakomponentit löytyivät intuitiivisesti.

Huonona puolena voisi mainita, että esitetyt suunnittelumallit ratkaisevat pääasiassa teknisiä ongelmia. Esimerkiksi käytettävyyden suunnittelu jää käyttöliittymien suunnittelijoille ja toteuttajille. Myöskään työ- ja liiketoimintaa ei pyritä millään lailla kehittämään. Oletetaan, että liiketoiminnan prosessit ovat kunnossa ja komponenttisovellus rakennetaan niihin perustuen.

8 Pohdinta

Ohjelmistosuunnittelu tuottaa sovelluksia hyvin erilaisiin kohteisiin. Eri lähtökohdista alkaen kohdealueesta löytyy erilaisia ohjelmistosuunnittelussa tarvittavia tietoja. Siksi tarvitaan monenlaisia menetelmiä. Menetelmien tunteminen on tarpeellista, jotta osataan poimia oikea menetelmä oikeaan ongelmaan tai poimia eri menetelmistä parhaat puolet ja yhdistellä niitä luovasti parhaan lopputuloksen saavuttamiseksi.

Komponentteja käytetään nykyään yleisesti ohjelmistosuunnittelussa, mutta yleiset vaatimusmäärittelymenetelmät eivät ole erityisesti komponenttisuuntautuneita. Komponenttisysteemejä on hyvin monenlaisia ja niissä on kussakin erilaisia tyypillisiä piirteitä. Muilla komponentteja käyttävillä aloilla on selvät skeemat lopullisesta koostetuotteesta. Tiedetään esimerkiksi, että juna koostuu veturista ja erilaisista vaunuista. Erilaiset mallit eri tyyppisistä komponenttisovelluksista ovat hyödyksi ohjelmistotuotannonkin alalla. Tällöin tiedetään jo ennen suunnittelua, minkä tyyppisiä komponentteja tarvitaan mihinkin sovellukseen ja komponenttien määrittely on helpompaa. Tarvitaan siis kohdealuekohtaisia skeemoja. Problem Domain Oriented Analysis (PDOA) on menossa tähän suuntaan, samoin metamallinnusta käyttävät menetelmät. Nämä ovat kuitenkin uudehkoja ja kehitteillä olevia menetelmiä, ja niistä ei vielä ole kovin paljon tutkittua tietoa.

Suunnittelumalli-idea tukee komponenttituotannon kanssa samoja tavoitteita: uudelleenkäytettävyyttä, joustavuutta ja ymmärrettävyyttä. Suunnittelumalleja käytetään apuna komponenttien ja arkkitehtuurien suunnittelussa. Suunnittelumallien käyttöä voidaan laajentaa koskemaan yrityksen komponenttituotantoprosessia siten, että yrityksen käyttämät toimintatavat ja mallit esitetään suunnittelumallin avulla, jolloin niitä voi käyttää ohjenuorana prosessissa.

Tutkimuksessa tarkasteltiin eri lähtökohdista alkavia vaatimusmäärittelymenetelmiä. Lähemmin tarkasteltiin ActADia (toiminnan teoriasta lähtevä), Robertsonin menetelmää eli Volerea (tapahtumalähtöinen) ja Arsanjanin menetelmää (liiketoimintalähtöinen). Tutkitut kolme menetelmää sijoittuvat vaatimusmäärittely- ja ohjelmistotuotantoprosessin eri vaiheisiin. ActAD ja Volere alkavat varhaisesta kohdealueen kartoit-

tamisesta ja mallintamisesta. Volere-prosessissa edetään vaatimusten tarkkaan kuvaamiseen ja vaatimusten varastointiin uudelleenkäyttöä varten. Arsanjanin menetelmä alkaa ohjelmistotuotantoprosessin myöhemmästä vaiheesta kuin kaksi muuta esitettyä menetelmää, mutta etenee pidemmälle komponenttisuunnitteluun.

Jokaisessa menetelmässä on jotakin ansiokasta. ActADin paras puoli on tehokas ja monipuolinen tiedon kerääminen ja kohdealueen työtoiminnan mallintaminen toimijaverkkona. Lisäksi ActADiin kuuluu olennaisena osana myös työtoiminnan ja tietotekniikan yhtäaikainen kehittäminen. Voleressa puolestaan on hyvä vaatimusten dokumentointimalli. Arsanjanin menetelmässä on hyvä suunnittelumalli komponenttiarkkitehtuurille ja komponenteille. Mielenkiintoista olisi tutkia, voidaanko menetelmiä yhdistää poimimalla kustakin menetelmästä paras anti ja saada aikaan uusi kattavampi menetelmä.

Volere ja ActAD ovat tutkituista menetelmistä eniten samankaltaisia: molemmissa korostetaan asiakkaan työn ymmärtämisen tärkeyttä ohjelmistosuunnittelussa. Kohdealueen työntekijöillä ja asiantuntijoilla sekä ohjelmistosuunnittelijoilla on oltava yhteinen käsitys kohdealueesta kokonaisuutena, ratkaistavasta ongelmasta ja ongelman ratkaisusta. Voleressa kohdealue mallinnetaan kontekksti- ja yhteyskaavioiden sekä prosessiketjujen avulla, jolloin kokonaiskuva saattaa jäädä hämäräksi. ActADissa puolestaan on mallintamisen pohjana teoria työtoiminnan rakenteesta, joka sitoo osaalueet kokonaisuuteen. ActAD-malli on selkeä ja Volerea ilmaisuvoimaisempi kuvaustapa, joka on helposti opittavissa. Tällöin se toimii hyvin myös eri asianosaisten kommunikointia helpottavana välineenä.

Kun kohdealue on laaja ja monimutkainen, on tarpeen jakaa kokonaisuus osiin, jotta voidaan analysoida kohdealueen tietoja riittävän pieninä ja helposti hallittavina paloina. Volere-menetelmän puutteena on se, ettei siinä ole keinoa työn osiin jakamiseen tai osien toisiinsa sitomiseen. Tässä tutkielmassa Volere-menetelmää laajennettiin osatyön (part of Work) käsitteellä. Lisäksi voisi tutkia, miten ilmaistaan useiden osatöiden välisiä yhteyksiä ja liittymistä kokonaisuuteen. ActADissa puolestaan on valmiina keinot mallintaa monimutkainen ja monen toimijan yhteistyönä tapahtuva työtoiminta kolmella eri tasolla: yksittäisen toimijan teko, yksittäinen toiminta usean toimijan ryhmätyönä sekä toiminnoista muodostuva toimintojen verkko. Yksittäiset

teot ja toiminnat voidaan yhdistää saumattomasti kokonaisuuteen. Taustalla oleva toiminnan teoria antaa ajattelumallin ja nimeää työtoiminnan rakenneosat, joita voidaan hyödyntää kohdealueen kartoittamisessa, kohdealueen tieto- ja toimintarakenteiden hahmottamisessa sekä kuvaamisessa.

Tutkittujen menetelmien suhde komponentteihin on hyvin erilainen. ActADissa ei edetä komponenttisuunnitteluun saakka. Olisi kiinnostavaa tutkia, onko mahdollista käyttää ActAD-mallia komponenttien etsimisen ja suunnittelun apuna. Volere-menetelmäkään ei ole suoraan sidoksissa komponentteihin, mutta prosessikuvauksista voidaan mahdollisesti edetä prosessikomponentteihin. Arsanjanin menetelmä antaa käyttökelpoisen abstraktin mallin siitä, mitä osia (osakomponentteja) komponentti tarvitsee toimiakseen laajan komponenttisysteemin osana.

Vaatimusmäärittelyn ja vaatimusmäärittelymenetelmien muokkaaminen paremmin komponenttituotantoa huomioivaksi sisältää useita haasteita. Eräs lisätutkimusta kaipaavista asioista on, miten tietoja tulee ryhmitellä vaatimusten keräämis- ja analysointivaiheessa, jotta ryhmittely tukisi komponenttituotantoa. Dokumentointitavoista tulisi tutkia, voidaanko vaatimusmäärittelydokumenteista johtaa suoraan komponenttikuvauksia, jolloin työläs dokumentointivaihe helpottuisi ja tuotetut dokumentit yhdenmukaistuisivat. Tämä johtaisi komponenttimarkkinoiden vilkastumiseen ja sitä kautta ohjelmistotuotannon tuottavuuden paranemiseen.

Lähteet

[Ars01] Arsanjani, Ali: Rule Object 2001: A Pattern Language for Adaptive and Scalable Business Rule Construction*, Proceedings of The 8th Conference on Pattern Language of Programs*, 11 – 15 September, Monticello, Illinois 2001.

[Ars02] Arsanjani, Ali: "Towards a Pattern Language for Web Services Architecture", *Proceedings of the 9th Conference on Pattern Languages on Programs 2002,* Monticello, Illinois, 2002.

[Boe88] Boehm, B. W.: A Spiral Model of Software Development and Enhancement, *IEEE Computer*, Vol. 21, No.5 (May 1988) pp. 61-72.

[Bos00] Bosch, Jan: *Design and use of software architectures, Adopting and evolving a product-line approach,* Addison-Wesley, 2000.

[Bra02] Bray, Ian K.: *An Introduction to Requirements Engineering,* Addison-Wessley, 2002.

[Bød91] Bødker, S.: Activity theory as a challenge to systems design, *Information Systems Research: Contemporary Approaches and Emergent Traditions,* H-E. Nissen, H. K. Klein and R. Hirscheim (eds.), Elsevier, Amsterdam, 1991, pp. 551-564.

[ChD01] Cheesman, John and Daniels, John: *UML Components A Simple Process for Specifying Component-Based Software*, Addison Wesley, Boston, 2001.

[CoY90] Coad, P. and Yourdon, E.: *Object-oriented Analysis,* Prentice Hall, Englewood Cliffs, New Jersey 1990.

[DSW99] D'Souza, Francis and Wills, Alan Cameron: *Objects, Components and Frameworks with UML, The Catalysis Approach,* Addison-Wesley, 1999.

[Eng87] Engeström, Yrjö: *Learning by Expanding,* Orienta-Konsultit Oy, Helsinki 1987.

[Fow97] Fowler, Martin: *UML distilled: Applying theObject Modelling Language,* Reading MA: Addison Wesley Longman 1997.

[GaL99] Gause, Donald G. and Lawrence, Brian: User-Driven Design Incorporating users into the requirements and design phase *Testing & Quality* Jan/Feb 1999 Vol 1, Issue.

[GaH94] Gamma, E., Helm R., Johnson R., Vlissides J.: *Design Patterns – Elements of Reusable Object-Oriented Software,* Addison-Wesley 1994.

[GrE01] Grünbacher, Paul, Egyed, Alexander, Medvidovic, Nenad: Reconciling Software Requirements and Architectures: The CBSP Approach, *Proceedings of the Fifth International Symposium on Requirements Engineering,* Toronto, Canada, August 27. - 31. 2001.

[HaM02] Haikala, Ilkka ja Märijärvi, Jukka: *Ohjelmistotuotanto,* Satku, Pieksämäki 2002.

[Haw01] Hawryszkiewycz, Igor: *Introduction to systems analysis & design,* Prentice Hall 2001.

[HeS00] Herzum, Peter and Simms, Oliver: *Business Component Factory,* Wiley, New York, 2000.

[HiH88] Hirsjärvi, Sirkka ja Hurme, Helena: *Teemahaastattelu,* Yliopistopaino, Helsinki 1988.

[Imm00] Immonen, Mirja: *Suunnittelumallit,* Erikoistyö, Kuopion yliopisto, 2000. Saatavilla www-muodossa: <URL:http://www.cs.uku.fi/research/Teho/>, viitattu 10.8.2004.

[Jaa97] Jaaksi, Ari: *Object-Oriented Development of Interactive Systems,* Tampereen teknillinen korkeakoulu Julkaisuja 201, Tampere 1997.

[JaG97] Jacobson, Ivar; Griss, Martin; Jonson, Patrik: *Software Reuse,* ACM Press, 1997.

[JaC92] Jacobson, I., Christerson, M., Jonson, P. and Övergaard, G.: *Object-oriented software engineering: A Use case driven approach,* Addison-Wesley, 1992.

[Jac99] Jacson, Michael: *Keynote Talk.* ITG/SEV Symposium, Zürich, 29 September 1999.

[Jän03] Jäntti, Marko: *Testitapausten suunnittelu UML-mallinnuksen avulla,* Pro Gradu -tutkielma, Kuopion yliopisto, 2003. Saatavilla www-muodossa: <URL:http://www.cs.uku.fi/research/Teho/>, viitattu 10.8.2004.

[KoM03] Korpela, Mikko and Mursu, Anja: *Means for cooperative work and activity networks: An analytical framework,* Workshop at ECSCW'03, 8th

European Conference of Computer-Supported Cooperative Work Helsinki, Finland, 14th September 2003.

[KoM04] Korpela M, Mursu A, Soriyan A, Eerola A, Häkkinen H and Toivanen M.: *IS research and development by activity analysis and development: Dead horse or the next wave?* In: Relevant Theory and Informed Practice: Looking Forward from a 20-Year Perspective on IS Research, Manchester, 15-17 July 2004, Amsterdam: Kluwer Academic, 2004.

[Kor94] Korpela, Mikko: *Nigerian Practice in Computer Systems Development: A Multidisciplinary Theoretical Framework, Applied to Health Informatics,* Academic Dissertation, Helsinki University of Technology, 1994.

[Kos00] Koskimies, Kai: *Oliokirja,* Gummerus Kirjapaino Oy, Jyväskylä 2000.

[KoS00] Korpela, M., Soriyan, H. A. and Olufokunbi K. C.: Activity Analysis as a Method for Information Systems Development: General Introduction and Experiments from Nigeria and Finland, *Scandinavian Journal of Information Systems*, 2000, 12: 191-210.

[Kuu91] Kuutti, K.: Activity Theory and its applications to information systems research and development, *Information Systems Research: Contemporary Approaches and Emergent Traditions,* H-E. Nissen, H. K. Klein and R. Hirscheim (eds.), Elsevier, Amsterdam, 1991, pp. 529-549.

[Käh00] Kähkipuro, Pekka: Komponenttiarkkitehtuurien vaikutus komponenttituotantoon, *Sytyke ry Systeemityö* 1/2000.

[LeA02] Levi, Keith and Arsanjani, Ali: A Goal-driven Approach to Enterprise Component Identification and Specification Mapping a Business Architecture to a Component-based Software Architecture, *Communications of the ACM* Volume 45, Issue 10 (October 2002).

[Leo78] Leontiev, A. N.: *Activity, Consciousness and Personality,* Prentice Hall, Englewood Cliffs NJ 1978.

[Mur02] Mursu, Anja: *Information Systems Development in Developing Countries: Risk Management and Sustainability Analysis in Nigerian Software Companies*, Academic Dissertation, Department of Computer Science and Information Systems, University of Jyväskylä, 2002.

[Myö02] Myöhänen, Hannu: *Jäljitettävyys ohjelmistotuotannon tukena,* Pro gradu -tutkielma, Kuopion yliopisto 2002. Saatavilla www-muodossa: <URL:http://www.cs.uku.fi/research/Teho/>, viitattu 10.8.2004.

[Par03] Parsons, Rebecca: Components and the World of Chaos. *IEEE Software,* pp. 83-85, May/June 2003.

[Put94] Putkonen, Anne: *A Methodology for Supporting Analysis, Design and Maintenance of Object-oriented Systems*. Academic Dissertation, Kuopion yliopiston julkaisuja C. Luonnontieteet ja ympäristötieteet 19, Kuopio 1994.

[PäH02] Päivärinta, Eeva, Haverinen, Riitta: *Ikäihmisten hoito- ja palvelusuunnitelma opas työntekijöille ja palveluista vastaaville,* STAKES oppaita 52, Gummerus Kirjapaino Oy, Jyväskylä 2002.

[RoR99] Robertson, Suzanne and Robertson, James: *Mastering the Requirements Process,* Addison-Wesley, Harlow 1999.

[Roy70] Royce, W. W.: *Managing the development of large software systems.* IEEE Westcon, Los Angeles 1970.

[RuJ99] Rumbaugh, J., Jacobson, I. and Booch, G.: *The Unified Modelling Language Reference Manual,* Addison-Wesley 1999.

[RuB91] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W.: *Object-oriented Modelling and Design,* Prentice-Hall, 1991.

[Saa87] Saaren-Seppälä, Kari: *Seinätekniikka, Seinäkuvien käyttö suunnittelussa ja ryhmätyössä*, Vihdin Kirjapaino Oy, 1987.

[Sin95] Sinkkonen, Sirkka: *Kotihoidon sisältö ja laatu Kuopiossa 1994,* Kuopion yliopiston julkaisuja E yhteiskuntatieteet, Kuopio 1995.

[Smo03] Smolander, Kari: *On the role of architecture in systems development,* Academic Dissertation, Lappeenrannan teknillinen yliopisto, Digipaino 2003.

[Som01] Sommerville, Ian: *Software Engineering,* Addison-Wesley 2001.

[Sub99] Subramanian, Usha V: An Event, Activity and Process Based Methodology for Requirements Elicitation and its Application to an Educational Information System, *Proceedings of the 6th Asia Pacific Software Engineering Conference*, Takamatsu, Japan 1999.

[Tie01] Tietotekniikan liitto ry:n sanastotoimikunta: *Tietotekniikan liiton ATK-sanakirja,* Satku, Pieksämäki 2001.

[ToH03] Toivanen, Marika, Häkkinen, Heidi, Laitinen, Pertti ja Röppänen, Päivi: Toimintalähtöisten integraatiotarpeiden vaatimusmäärittely kotihoidon kontekstissa, SoTeTiTe2003, *Sosiaali- ja terveydenhuollon tietotekniikan ja tiedonhallinnan tutkimuspäivät*, Saranto, Kaija; Häyrinen, Kristiina (toim.). Osaavien keskusten verkoston julkaisuja, 2003.

[ToE03] Toivanen, Marika, Eerola, Anne and Korpela, Mikko: From information systems requirements to software components - home care case, Sanna Laukkanen, Sami Sarpola, eds. *Electronic Proceedings of the 26th Information Systems Research Seminar in Scandinavia* (Proceedings of the 26th IRIS), Haikko Manor, Finland, August 9-12, 2003.

[ToH04] Toivanen, Marika, Häkkinen, Heidi, Eerola, Anne, Korpela, Mikko and Mursu, Anja: *Gathering, Structuring and Describing Information Needs in Home Care: A Method for Requirements Exploration in a "Gray Area",* In: MEDINFO 2004: Building High Performance Health Care Organizations, San Francisco, 7-11 September 2004, IMIA, 2004.

[ToL04a] Toivanen, Marika, Laitinen, Pertti, Häkkinen, Heidi, Minkkinen, Irmeli ja Röppänen Päivi: *Kotihoidon toiminnat ja tiedot. Kotihoidon tiedon tarpeet. Toimintalähtöinen vaatimusmäärittely*, PlugIT-hanke-julkaisu, 2004.

[ToL04b] Toivanen, Marika, Laitinen, Pertti, Häkkinen, Heidi, Minkkinen, Irmeli, Röppänen, Päivi ja Tuomainen, Tuula: *Kotihoidon prosessit ja tietoarkkitehtuuri, Kotihoidon tiedon tarpeet. Toimintalähtöinen vaatimusmäärittely*, PlugIT-hankejulkaisu, 2004.

[ToM03] Toroi, T., Mykkänen, J., Jäntti, M. ja Eerola, A.: Komponenttisysteemien testaus, SoTeTiTe2002, *Sosiaali- ja terveydenhuollon tietotekniikan ja tiedonhallinnan tutkimuspäivät*, Nykänen, Pirkko (toim.). Osaavien keskusten verkoston julkaisuja, 2002.

[VoV02] Voutilainen, Päivi, Vaarama, Marja, Backman, Kaisa, Paasivaara Leena, Eloniemi-Sulkava, Ulla ja Finne-Soveri Harriet U.: *Ikäihmisten hyvä palvelu, Opas laatuun,* STAKES OPPAITA 49, Gummerus Kirjapaino Oy, Jyväskylä 2002.

[Vyg78] Vygotsky, L.S.: *Mind in society: The Development of Higher Psychological Processes.* Compiled from several sources and edited by Cole,

M., John-Steiner, V., Scribner, S. and Souberman E. Harvard University Press, London, England 1978.

[ZhL01] Zhang, Zheying and Lyytinen, Kalle: A Framework for Component Reuse in a Metamodeling-Based Software Development, *Requirements Engineering* 2001 Vol 6 p.116-131.

[YaA99] Yacoub, Sherif, Ammar, Hany, Mili, Ali: Characterizing a Software Component, *Proceedings of International Workshop on Component-Based Software Engineering,* Los Angeles, USA, May 17. – 18. 1999

[You89] Yourdon, Edward: *Modern Structured Analysis,* Prentice-Hall, New Jersey, 1989.

[www01] *PlugIT: terveydenhuollon sovellusintegraatio.* <URL:http://www.uku.fi/tike/plugit/>, viitattu 4.6.2004.

[www02] *PlugIT-Teho*. <URL: http://www.cs.uku.fi/research/Teho/>, viitattu 30.6.2004

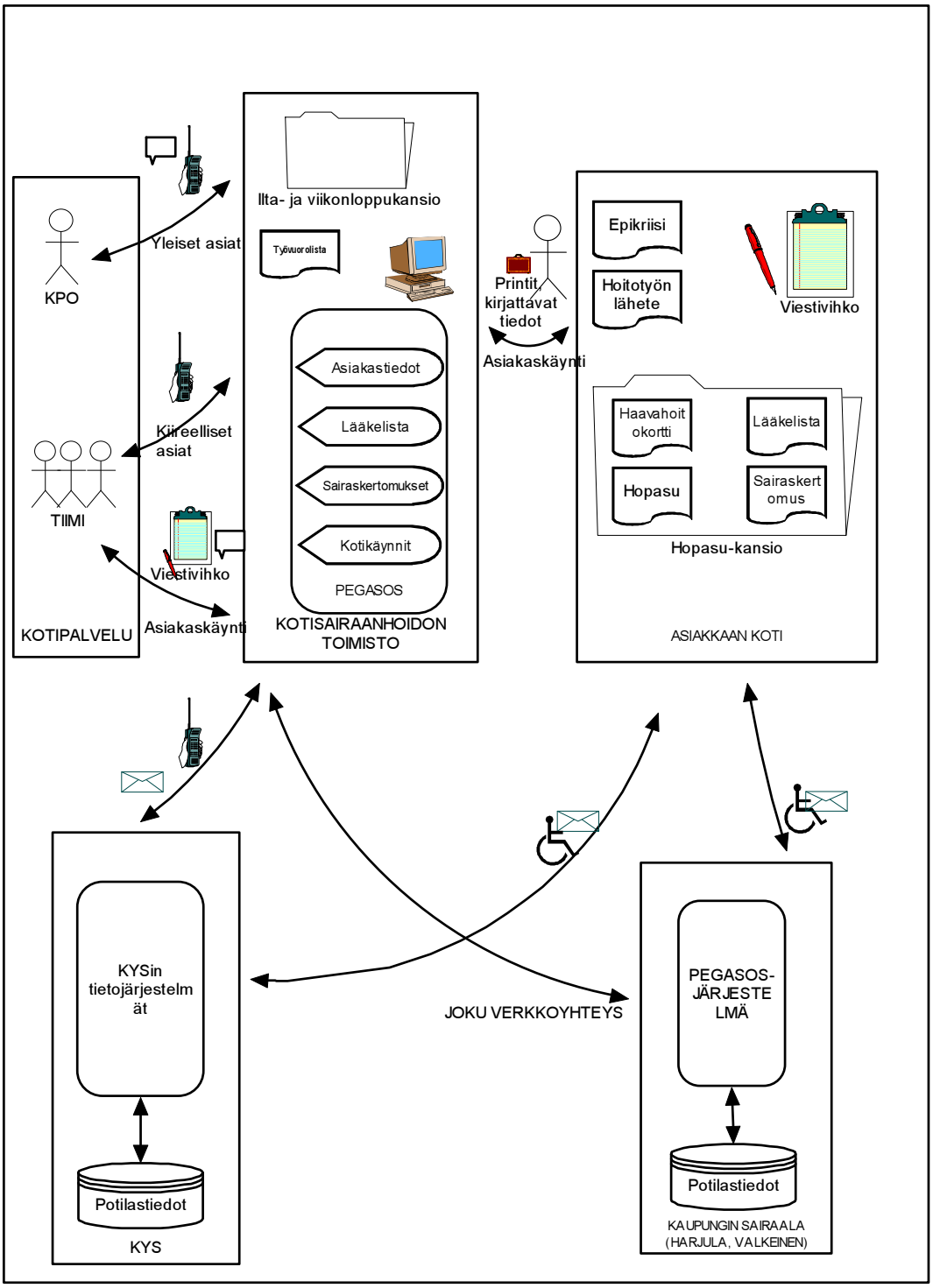
[www03] Alexander, Dey: *Empowering users through user-centred web design* <URL: http://www.its.monash.edu.au/web/slideshows/ucd/spusc.html>, viitattu 21.06.2004

[www04] Ferg, Stephen: *What's Wrong with the Use Cases?*. <URL:http://www.ferg.org/papers/ferg--whats_wrong_with_use_cases.html>, viitattu 28.4.2004.

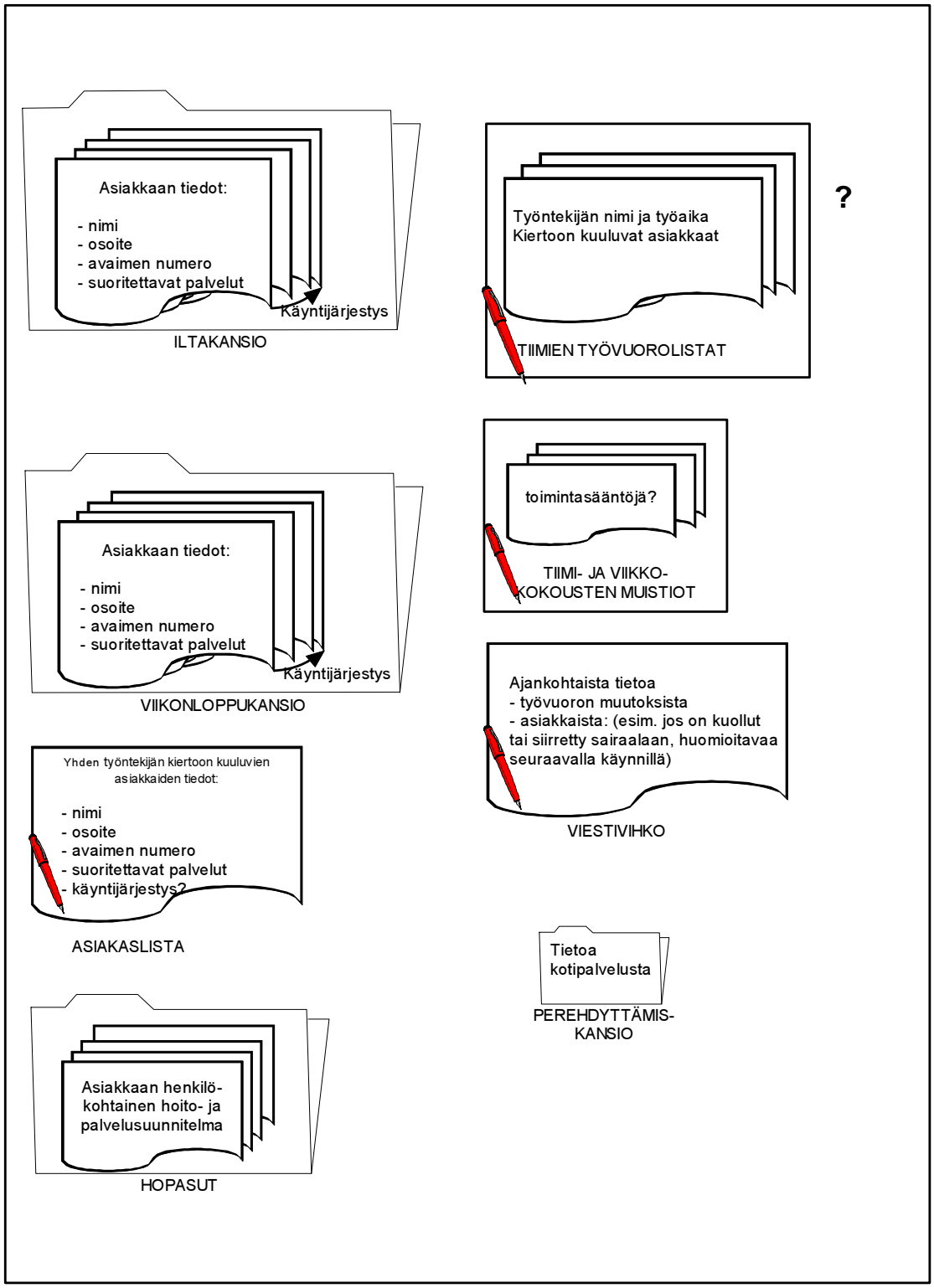
[www05] Auer, Antti: *TieVie verkkojakso: Sisällöntuotanto*, <URL:http://virtuaaliyliopisto.jyu.fi/oppimateriaali/tievie/index.html>

[www06] Robertson, S. and Robertson J.: *Volere Requirements Specification Template.* Edition 9. Copyright © 1995 – 2003 Atlantic Systems Guild. <URL:http://www.volere.co.uk>, viitattu 14.06.2004.

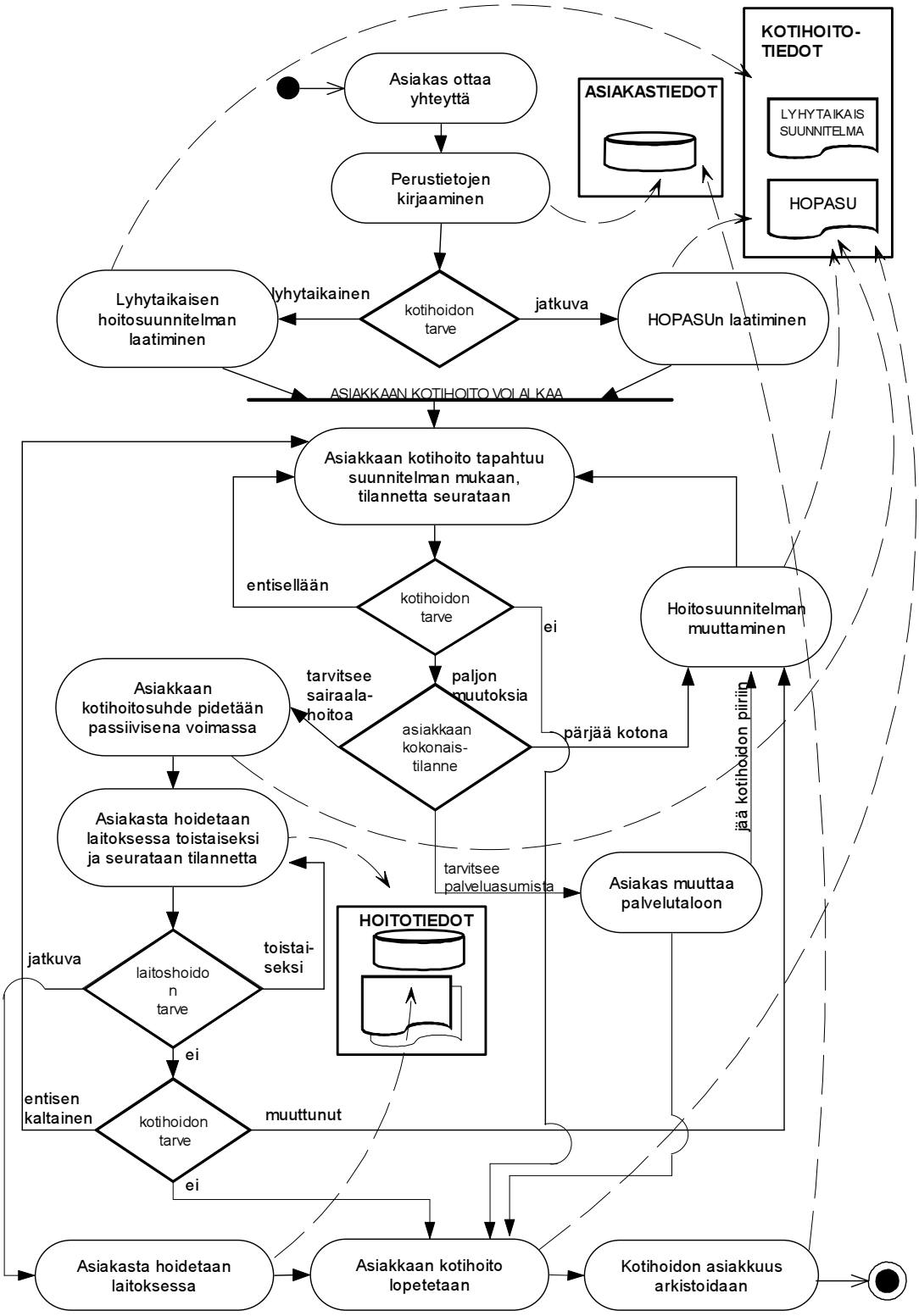
[www07] *Kuopion sosiaali- ja terveyspalvelut, vanhustenhuollon strategia v 1999 – 2010.* <URL: http://www.kuopio.fi/soste/i_stra.htm>, viitattu 20.8.2003.



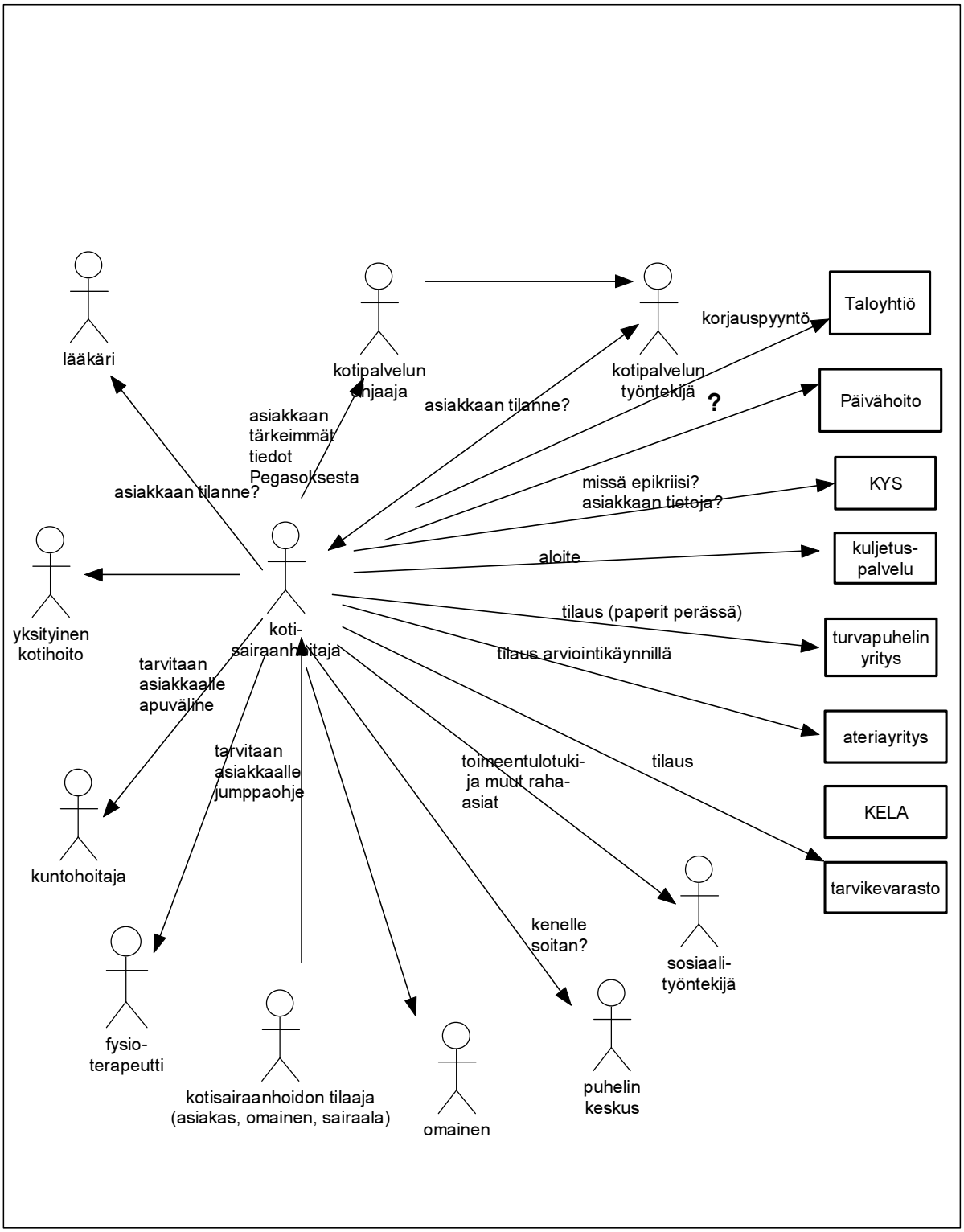
Kuva 1: Tiedon muoto ja sijainti kotisairaanhoidossa



Kuva 2: Tiimitilan kansiot



Kuva 3: Aktiviteettikaavio, asiakkaan hoito



Kuva 4: Puhelinkartta, kotisairaanhoitaja

LIITE 2: Esimerkki toimintatarinasta

HILJAN TARINA: ASIAKAS TARVITSEE PITKÄAIKAISTA KOTIHOITOA

Hilja, 85 v. asuu yksin kotona Kuopion Leväsellä, ja hänen toimintakykynsä on heikentynyt: ateriointi, pukeminen ja peseytyminen omin voimin on hankalaa. Lääkkeet unohtuvat joskus ottamatta. Hiljan naapurissa käy kotihoidon väkeä ja Hilja päättää selvittää, olisiko hänenkin mahdollista saada apua. Hilja kysyy naapurissa käyvältä hoitajalta, kenelle pitää soittaa. Hoitaja antaa kotipalvelunohjaaja Ailin puhelinnumeron. *Hilja soittaa Ailille.*

Puhelun aikana Aili kyselee Hiljan *perustiedot: nimi, syntymäaika, osoite, lähiomainen. Tulotiedot* löytyvät suoraan KELA:lta. Koska Hiljalla on vain peruseläke ja pieni leskeneläke, tulorajat mahdollistavat kunnallisen kotihoidon saamisen. Aili kirjaa uuden asiakkaan perustiedot tietojärjestelmään. Alustavasti kysellään myös millaista apua ja kuinka usein Hilja tarvitsisi ja millaiset olot on kotona. *Aili arvioi, että Hilja tarvitsee luultavasti apua joka päivä jatkuvasti*. Sovitaan ajankohta, jolloin Aili tulee käymään Hiljan kotona ja tekemään tarkemman Hoito- ja palvelusuunnitelman (HOPASU). Mukaan tulee myös Hiljan asuinalueella toimiva lähihoitaja Sari, josta tulee Hiljan omahoitaja.

Ensimmäisellä kotikäynnillä Aili ja Sari tutustuvat Hiljaan ja hänen poikaansa Mikkoon 60 v., joka on Hiljan lähin omainen ja asuu Siilinjärvellä. Hiljalla on reuma ja pahoja kulumavikoja selässä sekä lonkissa. Näistä johtuen Hiljan on vaikea pukea vaatteita päälleen tai peseytyä yksin. Kädet vapisevat joskus niin pahasti, ettei ruuanlaitto tai ruokailukaan suju hyvin omin voimin. Sisällä hän pystyy liikkumaan jonkin verran omin avuin keppiin nojaten, mutta ulkoiluun ja asiointiin tarvitaan apua. Hiljalla on lääkkeet reumaan, alhaisen verenpaineen hoitoon ja joitakin särkylääkkeitä. Joskus Hilja on huomannut, että on vaikea muistaa, tuliko kaikki lääkkeet otettua oikeaan aikaan. Joskus niitä näyttää myös hävinneen. Myös joitakin Hiljan kodin tavaroita tuntuu hävinneen. Mikko epäilee, että ”äiti ei enää oikein muista kaikkea”.

Aili *kirjoittaa HOPASU- lomakkeelle Hiljan perustiedot, Mikon yhteystiedot, Hiljan sairaudet ja lääkitystiedot.* Aili ja Sari arvioivat, että kotipalvelukäyntejä tarvittaisiin päivittäin kolme: aamu-, päivä- ja iltakäynnit. Aamukäynnillä hoitaja auttaa Hiljan aamutoimissa: pukemisessa ja tarjoilee aamupalan sekä antaa aamulääkkeet dosetista. Päiväkäynnillä hoitaja tarjoilee Ateriapalvelusta toimitetun aterian ja laittaa välipalan valmiiksi. Tarvittaessa soitetaan yhdessä Hiljan kanssa ostoslista kaupan kotiintoimituspalveluun. Mikko lupaa tehdä sopimuksen Ateriapalvelun kanssa. Iltakäynnillä hoitaja avustaa iltatoimissa ja antaa iltalääkkeet. Kotisairaanhoitajan käyntejä arvioidaan tarvittavan kerran viikossa. Käynnillään ksh jakaa viikon lääkityksen dosettiin ja mittaa verenpaineen. Ksh arvioi myös Hiljan muistin toimintaa. *Sovitut käynnit ja tehtävät toimet kirjataan HOPASU-lomakkeelle.* Tavoitteeksi kirjataan Hiljan itsenäinen asuminen kotona kotihoidon avustuksella. Sovitaan, että HOPASU tarkistetaan viimeistään puolen vuoden kuluttua.

Seuraavana päivänä Sari tulee ensimmäiselle hoitokäynnille ja kotihoito käynnistyy. *Kotihoidon suunnitelmaa noudatetaan* ja Hilja tuntuu pärjäävän hyvin kotona saamansa avun turvin. *Puolivuotistarkastuksessa* ei huomata mitään muutosten tarvetta ja suunnitelmaa jatketaan entisellä mallilla.