Markus Penttilä

Agenttipohjaisten suunnittelumenetelmien vertailu

Tietojärjestelmätieteen
pro gradu - tutkielma
31.8.2004

Jyväskylän yliopisto
Tietojenkäsittelytieteiden laitos
Jyväskylä

TIIVISTELMÄ

Penttilä, Markus Ensio
Agenttipohjaisten suunnittelumenetelmien vertailu / Markus Penttilä
Jyväskylä: Jyväskylän yliopisto, 2004.
140 s.
Pro gradu – tutkielma

Agenttipohjainen ohjelmistotuotanto edustaa yhtä uusimmista paradigmoista ohjelmistotuotannon alueella. Agentilla tarkoitetaan ohjelmiston osaa, joka kykenee itsenäiseen ja älykkääseen toimintaan käyttäjänsä puolesta saavuttaakseen sille suunnitellut tavoitteet. Agenttipohjaisten järjestelmien suunnitteluun on kehitetty varsin kirjava joukko menetelmiä. Tutkielman päätavoitteena on kuvata, analysoida ja vertailla agenttipohjaisia menetelmiä sen selvittämiseksi, mitä yhtäläisyyksiä ja eroja niiden käsitteissä, malleissa ja prosesseissa. Tutkielma on otteeltaan käsitteellisteoreettinen ja aihetta lähestytään agentteja, agenttipohjaisia menetelmiä ja menetelmien vertailua käsittelevän kirjallisuuden pohjalta.

Tutkielman keskeisinä tuloksina ovat 28 menetelmää kattava agenttipohjaisten menetelmien kartoitus, menetelmien vertailua varten muodostettu viitekehys sekä menetelmien vertailusta tehdyt johtopäätökset. Vertailuun on valittu neljä menetelmää: Gaia, Tropos, MaSE ja MESSAGE. Menetelmiä vertaillaan käsitteiden, mallien, prosessien ja menetelmien tarjoaman käytännön tuen osalta. Käsitteiden osalta menetelmät ovat melko samankaltaisia. Sen sijaan menetelmien mallien käsitteelliset sisällöt eroavat toisistaan varsin paljon. Myös notaatioiltaan menetelmät ovat hyvin erilaisia. Menetelmät kattavat ohjelmistotuotannon vaiheista selkeästi analyysin ja suunnittelun, ja lisäksi Tropos kattaa myös vaatimusmäärittelyn ja toteutuksen. Menetelmät eivät eksplisiittisesti tue projektin johtamista. Kaikki vertailun menetelmät ovat lähtökohtaisesti tarkoitettu uusien järjestelmien suunnitteluun. Tropos pohjautuu BDI-arkkitehtuurin käsitteille, mutta muut menetelmät eivät pohjaudu mihinkään tiettyyn arkkitehtuuriin.

AVAINSANAT: Ohjelmistoagentit, agenttipohjainen ohjelmistotuotanto, agenttipohjaiset suunnittelumenetelmät, menetelmien vertailu

SISÄLLYSLUETTELO

KUVIOT

TAULUKOT

1 JOHDANTO

Suurten ohjelmistojen kehittämistä pidetään yhtenä monimutkaisimmista ihmisen suorittamista tehtävistä. Ohjelmistojen kehittäminen on muuttumassa entistäkin monimutkaisemmaksi muiden muassa hajautettujen järjestelmien yleistymisen myötä. Ohjelmistotuotannon tehtävänä on tarjota välineitä ja tekniikoita tämän monimutkaisuuden hallitsemiseksi. Agenttipohjainen ohjelmistotuotanto (Agent-Oriented Software Engineering) edustaa yhtä uusimmista paradigmoista ohjelmistotuotannon alueella. Agenttipohjaisten sovellusten lisääntyminen antaa viitteitä lähestymistavan mahdollisista eduista. (Jennings 2001)

Alan nuoruudesta johtuen peruskäsitteistö ei ole vielä vakiintunut. Tässä tutkielmassa *agentilla* tarkoitetaan ohjelmiston osaa, joka kykenee itsenäiseen ja älykkääseen toimintaan saavuttaakseen sille suunnitellut tavoitteet (Genesereth. & Ketchpel 1994; Wooldridge & Jennings 1995; Nwana 1996). *Agenttipohjainen järjestelmä* on järjestelmä, jonka toiminta perustuu agentteihin (Jennings ym. 1998). *Moniagenttijärjestelmä* taas on järjestelmä, jossa itsenäiset agentit toimivat vuorovaikutuksessa toisten agenttien kanssa (Jennings 2001).

Vaikka agenttipohjainen ohjelmistotuotanto onkin uusi paradigma, on agentteja sovellettu eri sovellusalueilla jo varsin pitkään. Yhtenä ensimmäisistä alueista agentteja alettiin soveltaa teollisuudessa, muun muassa prosessinvalvonnassa ja teollisessa tuotannossa (Jennings & Wooldridge 1998). Toinen merkittävä sovellusalue on kaupalliset sovellukset. Kaupallisella puolella agentteja on sovellettu tiedonhallintaan (Jennings ym 1998), elektroniseen liiketoimintaan (He ym. 2003) ja liiketoimintaprosessien hallintaan (Jennings ym. 2000). Kolmas tärkeä sovellusalue on lääketieteelliset sovellukset. Tällä alueella agentteja on sovellettu muun muassa potilasvalvontaan ja terveydenhuoltoon (Jennings & Wooldridge 1998). Nykyisin myös viihdeteolliset sovellukset ovat saavuttaneet vakavasti otettavien sovellusten aseman. Agentteja on sovellettu tietokonepeleihin, virtuaalitodellisuutta mallintaviin sovelluksiin sekä elokuviin (Jennings & Wooldridge 1998; Luck ym. 2002).

Suhteellisen laajasta sovellusaluekirjosta huolimatta reaalimaailmaan sijoittuvia sovelluksia on kuitenkin toteutettu melko vähän. Agenttipohjainen ohjelmistotuotanto

on vasta kehityksensä alkuvaiheessa, ja sillä on vielä monia haasteita edessään ennen kuin lähestymistapa saavuttaa laajan hyväksynnän. Yksi suurimmista haasteista on agenttipohjaisuutta tukevat suunnittelumenetelmät. Agenttipohjaisten sovellusten suunnitteluun on kirjallisuudessa esitetty lukuisia menetelmiä (esim. Lind 2001; Omicini 2001; Wood & DeLoach 2001; Zambonelli ym. 2003), mutta suurin osa niistä on vielä melko kypsymättömiä, eikä mikään menetelmistä ole saavuttanut valta-asemaa. (Wooldridge & Ciancarini 2001)

Erilaisia menetelmien vertailuja on kirjallisuudessa esitetty satoja (esim. Iivari 1994; Snoeck ym. 1996; Henderson-Sellers ym. 2001; Boertien ym. 2001). Agenttipohjaisten menetelmien vertailua on kuitenkin tehty vasta vähän (Dam & Winikoff 2004; Sturm & Shehory 2004). Avisonin ja Fitzgeraldin (1995) mukaan menetelmien vertailua tehdään sekä akateemisista että käytännön syistä. Syyt ovat osittain päällekkäisiä, ja tavoitteena on, että akateemiset tutkimukset tukisivat myös menetelmien valintaa käytännön tilanteissa ja vastaavasti käytännön tarpeet vaikuttaisivat akateemisten tutkimusten kriteereihin.

Tämän tutkielman tavoitteena on ensiksikin selvittää, mistä agenteissa on kysymys ja millaisiin sovelluksiin niitä on käytetty. Toisena, ja tärkeimpänä, tavoitteena tutkielmassa pyritään kuvaamaan, analysoimaan ja vertailemaan agenttipohjaisia menetelmiä sen selvittämiseksi, mitä yhtäläisyyksiä ja eroja niiden käsitteissä, malleissa ja prosesseissa on. Tutkielmassa on kaksi keskeistä tutkimusongelmaa, joista toinen voidaan jakaa useammaksi osaongelmaksi:

1. Millaisia agenttipohjaisia menetelmiä kirjallisuudessa on esitetty?
2. Kuinka kattavia eri menetelmät ovat ja mitä yhtäläisyyksiä ja eroja niistä on löydettävissä?
    2.1. Mitkä ovat menetelmien keskeiset käsitteet ja käsiterakenteet?
    2.2. Mitkä ovat menetelmien keskeiset mallit?
    2.3. Mitä ohjelmistotuotannon vaiheita menetelmien prosessit kattavat?

Aluksi tutkielmalle muodostetaan käsitteellinen perusta. Tämän jälkeen vastataan ensimmäiseen ongelmaan tekemällä laaja, 28 menetelmää kattava, kartoitus

kirjallisuudessa esitetyistä agenttipohjaisista menetelmistä. Tämän jälkeen suoritetaan menetelmien valinta tarkempaan vertailuun. Valitut menetelmät esitellään aluksi tarkemmalla tasolla kunnollisen kokonaiskuvan saamiseksi. Tämän jälkeen muodostetaan viitekehys vertailua varten, minkä jälkeen suoritetaan menetelmien syvällinen vertailu, joka antaa vastaukset toiseen tutkimusongelmaan.

Agentit ovat alun perin lähtöisin tekoälyn tutkimuksesta ja näin ollen siltä saralta on löydettävissä paljon erittäin teoreettista lähdemateriaalia, joka käsittelee mm. logiikkaa ja erilaisia tekoälyn teorioita agentteihin liittyen. Näihin asioihin ei tässä tutkielmassa puututa tarkemmin, vaan pääpaino on agenttien ohjelmistotuotantoon ja tietojärjestelmiin liittyvissä asioissa. Tutkielmassa keskitytään ohjelmistoagentteihin.

Tutkielma on otteeltaan käsitteellisteoreettinen ja aihetta lähestytään olemassa olevan kirjallisuuden pohjalta. Näin ollen menetelmien vertailu on lähtökohdiltaan akateeminen (vrt. Avison & Fitzgerald 1995). Vertailuun on valittu neljä menetelmää, mikä kirjallisuudesta löydettyjen menetelmien lukumäärään nähden on melko pieni. Tähän on kaksi syytä. Ensinnäkin menetelmistä on oltava saatavilla riittävästi materiaalia, jotta niistä saadaan kunnollinen kokonaiskuva ja niitä voidaan kattavasti vertailla. Toiseksi menetelmien suuri määrä vertailussa aiheuttaisi sen, että siitä tulisi varsin pinnallinen. Pitämällä menetelmien määrä suhteellisen pienenä varmistetaan se, että vertailusta saadaan riittävän syvällinen palvelemaan tutkielman tarkoitusta. Vertailumenetelmänä käytetään piirreanalyysiä (ks. tarkemmin Sol 1983).

Tutkielman keskeisinä tuloksina ovat 28 menetelmää kattava agenttipohjaisten menetelmien kartoitus, viitekehys agenttipohjaisten menetelmien arviointiin ja vertailuun sekä menetelmien vertailusta saadut arviot ja johtopäätökset, joista selviävät menetelmien keskeiset yhtäläisyydet ja erot. Vertailtavat menetelmät ovat Gaia (Zambonelli ym. 2003), Tropos (Castro ym. 2002; Bresciani ym. 2004), MaSE (Wood & DeLoach 2001) ja MESSAGE (Evans ym. 2001). Huomiota kiinnitetään erityisesti menetelmien käsitteisiin, malleihin ja prosesseihin.

Tutkielman tuloksista odotetaan olevan sekä tutkimuksellista että käytännön hyötyä. Tutkimuksellisena hyötynä tutkielma tarjoaa viitekehyksen agenttipohjaisten menetelmien arviointiin ja vertailuun. Kehystä voidaan käyttää sekä olemassa olevien

agenttipohjaisten menetelmien että uusien kehitettävien menetelmien vertailuun. Käytännön hyötynä tutkielma tarjoaa katsauksen agentteihin ja agenttien sovellusalueisiin sekä esittelee agenttipohjaisten menetelmien nykytilan, mikä tukee menetelmän valintaa ja soveltamista käytännön tilanteessa.

Tutkielma etenee siten, että luvussa 2 luodaan tutkielman käsitteellinen perusta määrittelemällä agenttikäsite ja sen keskeiset ominaisuudet sekä luokittelemalla agentit ja agenttisovellukset. Luvussa 3 kuvataan agenttipohjaisia menetelmiä. Aluksi kartoitetaan, millaisia menetelmiä kirjallisuudessa on esitetty, minkä jälkeen esitellään neljä vertailuun valittua menetelmää. Valitut menetelmät ovat Gaia, Tropos, MaSE ja MESSAGE. Neljännessä luvussa muodostetaan menetelmien vertailun pohjana käytettävä viitekehys. Aluksi tarkastellaan menetelmien vertailuun tarkoitettuja yleisiä viitekehyksiä sekä agenttipohjaisten menetelmien vertailuun tarkoitettuja kehyksiä. Tämän jälkeen valitaan yksi kehyksistä, josta muokataan sopiva kehys tämän tutkielman tarpeisiin. Luvussa 5 suoritetaan valittujen menetelmien arviointi ja vertailu. Menetelmiä vertaillaan käsitteiden, mallien, prosessien ja eräiden käytännön tukeen liittyvien seikkojen suhteen. Kuudennessa luvussa esitellään vertailusta tehdyt johtopäätökset. Tutkielma päättyy yhteenvetoon luvussa 7.

2 AGENTIT JA AGENTTIPOHJAISET SOVELLUKSET

Tässä luvussa käsitellään agentteja ja määritellään tutkimuksen kannalta keskeisiä käsitteitä. Ensin tarkastellaan agentin käsitettä, jonka jälkeen vertaillaan agentteja olioihin ja komponentteihin. Ensimmäisessä kohdassa esitetään myös erityyppisten agenttien luokittelu. Toisessa kohdassa kartoitetaan sovelluksia ja sovellusalueita, joihin agentteja on käytetty. Luku päättyy yhteenvetoon.

2.1 Mitä ovat agentit?

Sana agentti esiintyy yleiskielessä monissa yhteyksissä. Agentti on henkilö, joka toimii tai käyttää valtaa. Agentti tarkoittaa myös henkilöä, joka on oikeutettu toimimaan jonkun puolesta tai sijaisena. (Merriam-Webster OnLine 2003)

Tekoäly- ja tietojenkäsittelytieteen alalla agentin käsitteestä ei vallitse yhteisymmärrystä, ja kirjallisuudessa siitä ilmeneekin useita hieman toisistaan poikkeavia käsityksiä. Nwana (1996) vertaa agentin käsitteen määrittelyn vaikeutta tekoälyn käsitteen määrittelyyn. Tekoälyn tutkijoiden keskuudessa ei ole päästy yhteisymmärrykseen tekoälyn käsitteestä. Erilaisten määritelmien runsaus voidaan Luckin ja d'Invernon (2001) mukaan nähdä sekä vahvuutena että heikkoutena. Vahvuutena on se, että agentin käsitettä voidaan soveltaa hyvin laajasti ja erilaisiin tarkoituksiin. Heikkoutena on juuri se, ettei ole päästy yhteisymmärrykseen siitä, mistä agentti koostuu. Luck ja d'Inverno (2001) yrittävät artikkelissaan päästä eroon agentin käsitteen moniselitteisyydestä ja määrittelevät käsitteellisen viitekehyksen agentille käyttäen formaalia Z-notaatiota (Spivey 1992). Yleensä kirjoittajat esittävät tapauskohtaisesti artikkeleissaan, mitä he tarkoittavat agentin käsitteellä. Franklin ja Graesser (1997) ovat koonneet yhteen useiden tutkijoiden määritelmiä agentista. Wooldridgen ja Jenningsin (1995) mukaan kiinnostus agentteja kohtaan pysyi melko alhaisena jopa 1980-luvun loppupuolelle saakka. Siitä lähtien kiinnostus on ollut kasvavaa tekoälyn tutkijoiden lisäksi myös tietojenkäsittelytieteiden valtavirran tutkijoiden keskuudessa.

2.1.1 Agentin määritelmä

Agentin käsite juontaa juurensa hajautetun tekoälyn (distributed artificial intelligence, DAI) tutkimuksen alueelle ja tarkemmin Carl Hewittin (1977) aktorimalliin. Mallissa Hewitt esittää käsitteen interaktiivisesta ja samanaikaisesti toimivasta oliosta, jonka hän nimeää aktoriksi (actor). Aktorilla on kapseloitu sisäinen tila, ja se pystyy vastaamaan muiden samankaltaisten olioiden lähettämiin viesteihin. (Nwana 1996)

Wooldridgen ja Jenningsin (1995) mukaan agentti on laitteisto- tai ohjelmistopohjainen tietokonejärjestelmä, jolla on seuraavat ominaisuudet:

- *Autonomia* (autonomy): Agentit toimivat itsenäisesti ilman ihmisen suoraa puuttumista asiaan. Agentit voivat myös kontrolloida toimintojaan ja sisäistä tilaansa.
- *Sosiaalisuus* (social ability): Agentit toimivat vuorovaikutuksessa muiden agenttien kanssa jonkinlaisen agenttikommunikointikielen (agent communication language) välityksellä.
- *Reaktiivisuus* (reactivity): Agentit tekevät havaintoja ympäristöstään ja mukauttavat toimintaansa ympäristön muutosten mukaan.
- *Proaktiivisuus* (pro-activeness): Sen lisäksi, että agentit vastaavat ympäristön muutoksiin, ne kykenevät myös tavoitesuuntautuneeseen toimintaan tekemällä itse aloitteita.

Nwana (1996) määrittelee agentin ohjelmisto- tai laitteistokomponentiksi, joka kykenee vaativaan toimintaan suorittaakseen tehtäviä käyttäjänsä puolesta. Lisäksi agenttien täytyy olla itsenäisiä, yhteistyökykyisiä sekä oppimiskykyisiä. Edellä olevat määritelmät kattavat ohjelmiston lisäksi myös mahdollisen laitteiston. Genesereth ja Ketchpel (1994) käsittelevät agentteja nimenomaan ohjelmistoagentteina (software agents). Heidän mukaansa agentti on ohjelmistokomponentti, joka kommunikoi toisten agenttien kanssa ilmaisuvoimaisen agenttikommunikointikielen välityksellä. Heidän mukaansa agentti ei ole ohjelmistoagentti, jos se ei käytä agenttikommunikointikieltä viestiäkseen toisten agenttien kanssa. Agenttikommunikointikielen perusajatuksena on se, että erilaiset ja eri järjestelmissä toteutetut agentit voisivat kommunikoida keskenään jollakin standardinmukaisella kielellä. Esimerkkejä agenttikommunikointikielistä ovat FIPA ACL (FIPA 2003) ja KQML (Genesereth & Ketchpel 1994).

Tässä tutkielmassa *agentilla* tarkoitetaan ohjelmiston osaa, joka kykenee itsenäiseen ja älykkääseen toimintaan käyttäjänsä puolesta saavuttaakseen sille suunnitellut tavoitteet. Määritelmässä on käytetty hyväksi Wooldridgen ja Jenningsin (1995), Nwanan (1996) sekä Geneserethin ja Ketchpelin (1994) määritelmiä. Lisäksi agentilla on edellä mainitut Wooldridgen ja Jenningsin (1995) määrittelemät ominaisuudet. Agentti toimii sijoitettuna johonkin ympäristöön, ja se voi myös toimia esimerkiksi sulautetussa järjestelmässä. Agenttia voidaan kutsua myös ohjelmistoagentiksi tai älykkääksi agentiksi.

Agenttipohjainen järjestelmä (agent-based system) on järjestelmä, jonka toiminta perustuu agentteihin (Jennings ym. 1998). *Moniagenttijärjestelmä* (multiagent system) on järjestelmä, jossa itsenäiset agentit toimivat vuorovaikutuksessa toisten agenttien kanssa (Jennings 2001). Moniagenttijärjestelmää voidaan myös kuvailla eräänlaiseksi agenttien organisaatioksi. Etenkin agenttipohjaisia menetelmiä käsittelevässä kirjallisuudessa puhutaan usein organisaatiosta. Tässä tutkielmassa *organisaatio* on joukko agentteja, jotka toimivat yhdessä saavuttaakseen yhteisen tavoitteen. Organisaatio koostuu rooleista, jotka ovat ominaisia organisaation agenteille. (Sturm & Shehory 2004)

2.1.2 Agenttien vertailua olioihin ja komponentteihin

Agenttilähestymistapaa on vertailtu muun muassa oliolähestymistapaan (Wooldridge 1997; Jennings & Wooldridge 2001; Jennings 2001) ja komponenttilähestymistapaan (Jennings & Wooldridge 2001). Agenttien ja olioiden välillä on paljon yhtäläisyyksiä. Jenningsin (2001) mukaan molemmissa lähestymistavoissa painotetaan entiteettien välisen vuorovaikutuksen tärkeyttä. Lisäksi molemmissa lähestymistavoissa yhtenä periaatteena on tiedonpiilotus (information hiding).

Monista yhtäläisyyksistä huolimatta agentti- ja oliolähestymistavoilla on myös joitakin keskeisiä eroja. Ensinnäkin oliot ovat luonteeltaan passiivisia. Oliolle täytyy lähettää viesti ennen kuin se ”herää eloon” ja tulee aktiiviseksi. Toiseksi olioilla ja agenteilla on eroja tavassa, jolla ne kapseloivat tietoja. Oliot kapseloivat tilansa ja käyttäytymisensä toteutuksen, mutta ne eivät kapseloi käyttäytymisen aktivointia. Tämä tarkoittaa sitä, että mikä tahansa toinen olio voi kutsua mitä tahansa toisen olion julkista metodia. Kun

metodia on kutsuttu, olio toimii metodin määrittelyn mukaisesti. Oliot ovat tässä suhteessa hyvin kuuliaisia toisilleen. Agentti taas voi itse päättää, suorittaako se siltä pyydettyä toimintoa. Se voi esimerkiksi todeta, että siltä pyydetty toiminto aiheuttaa enemmän haittaa kuin hyötyä ja näin ollen kieltäytyä toiminnasta tai ainakin antaa jonkinlaisen varoituksen mahdollisista haitoista. Kolmanneksi, oliolähestymistapa ei tarjoa riittävästi käsitteitä ja mekanismeja monimutkaisten hajautettujen järjestelmien mallintamiseen. Esimerkiksi oliolähestymistavan metodien kutsu on liian alkeellinen mekanismi kuvaamaan monimutkaisissa hajautetuissa järjestelmissä tapahtuvan vuorovaikutuksen luonnetta. Neljänneksi, oliolähestymistapa ei tarjoa riittävästi tukea monimutkaisten rakenteiden muodostamiseen. Monimutkaisissa järjestelmissä on erilaisia organisatorisia suhteita, joista periytymis- ja koostesuhteet ovat vain yksinkertaisimpia esimerkkejä. (Jennings & Wooldridge 2001)

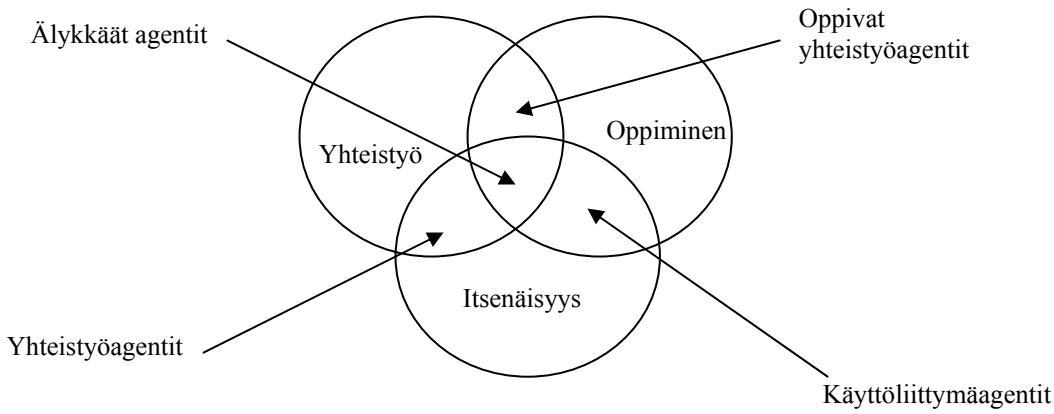
Komponenttipohjainen lähestymistapa liittyy läheisesti oliopohjaiseen lähestymistapaan. Komponenttipohjaisella ohjelmistojen kehittämisellä pyritään ohjelmistojen uudelleenkäyttöön. Yleisesti ottaen suurimmassa osassa ohjelmistoprojekteja ohjelmistokomponentit kehitetään alusta alkaen tyhjästä. Tämä on tehotonta, koska monet ratkaisut on kehitetty menestyksekkäästi jo moneen kertaan aiemmin. Ratkaisuna tähän ongelmaan on tarjottu komponenttipohjaista ohjelmistojen kehittämistä. Yksi tunnetuimmista ohjelmistokomponenttiarkkitehtuureista on ohjelmistoyhtiö Sun Microsystemsin kehittämä JavaBeans-teknologia (ks. tarkemmin Sun Microsystems 2003). Agentit ja komponentit ovat molemmat omavaraisia entiteettejä, joita ei tarvitse sijoittaa muiden komponenttien sekaan niiden tarjoamien palvelujen toteuttamiseksi. Komponentit eivät kuitenkaan ole itsenäisiä siinä mielessä kuin agentit ovat. Niiden käyttäytymisestä puuttuu myös reaktiivisuus, proaktiivisuus ja sosiaalisuus, jotka taas ovat olennainen osa agenttien käyttäytymistä. (Jennings & Wooldridge 2001)

Muiden lähestymistapojen puolustajat ja agenttilähestymistavan kritisoijat voivat sanoa, että agenttien käyttämät mekanismit voidaan toteuttaa myös muita tekniikoita ja lähestymistapoja käyttäen. Tämä voi hyvinkin pitää paikkansa, koska agenttipohjaiset ohjelmistot ovat vain ohjelmistoja muiden joukossa. Erilaisten lähestymistapojen arvo

onkin niiden ajattelutavassa ja tekniikoissa, joita ne tarjoavat ohjelmistojen kehittäjille. (Jennings & Wooldridge 2001)

2.1.3 Agenttien luokittelu

Agentteja voidaan luokitella useilla eri tavoilla esimerkiksi niiden käyttötapojen tai käyttäytymisen mukaan. Nwana (1996) toteaa artikkelissaan, että täydellinen, kaikki erilaiset ominaisuudet ja niiden yhdistelmät kattava luokittelu on käytännössä mahdoton muodostaa. Agenttien monipuolisuudesta johtuen erilaisia rooleja ja luokkia voidaan esittää jopa liikaakin. Esimerkiksi King (1995) esittää artikkelissaan agenteista rooliperusteisen luokittelun. Roolit ovat: hakuagentit, raportointiagentit, esittämisagentit, navigointiagentit, roolipeliagentit, hallinta-agentit, kohdealuespesifiset agentit, kehittämisagentit, analyysi- ja suunnitteluagentit, testausagentit, pakkausagentit sekä apuagentit. Tämä luokittelu tuntuu jo hieman liian laajalta ja sekavalta. Nwana (1996) luokittelee artikkelissaan agentit seitsemään luokkaan. Kuviossa 1 on esitetty agenttien ominaisuuksia ja agenttien osittainen luokittelu.



KUVIO 1. Agenttien osittainen luokittelu (Nwana 1996, 6).

Tarkastellaan seuraavaksi Nwanan (1996) tekemää luokittelua tarkemmin.

Yhteistyöagentit

Yhteistyöagentit (collaborative agents) korostavat tehtävien suorittamisessa autonomiaa sekä yhteistyötä muiden agenttien kanssa. Pystyäkseen yhteistyöhön agenttien on kyettävä neuvottelemaan toistensa kanssa saavuttaakseen yhteisymmärryksen.

Kommunikointi tapahtuu jonkin agenttikommunikointikielen välityksellä. Yhteistyöagentit kykenevät toimimaan järkevästi ja itsenäisesti moniagenttiympäristöissä (multiagent environments). Perusteluja yhteistyöagenttijärjestelmien tarpeellisuudelle voidaan esittää useita. Ne voivat esimerkiksi ratkaista ongelmia, jotka ovat liian suuria yhden agentin ratkaistavaksi tai niiden avulla voidaan ratkaista tilanteita, joissa tietoa tarvitaan useista hajautetuista lähteistä. (Nwana 1996)

Käyttöliittymäagentit

Käyttöliittymäagenttien (interface agents) toiminnassa korostuvat autonomia ja oppiminen. Käyttöliittymäagentti on eräänlainen henkilökohtainen avustaja, joka työskentelee yhteistyössä ohjelmiston käyttäjän kanssa samassa työympäristössä. Käyttöliittymäagentit ovat tavallaan eräänlaisia yhteistyöagentteja, mutta nyt yhteistyö tapahtuu agentin ja käyttäjän välillä, kun taas yhteistyöagenttien tapauksessa yhteistyö ilmenee agenttien keskinäisessä vuorovaikutuksessa. Yhteistyö käyttäjän kanssa ei välttämättä vaadi erillistä agenttikommunikointikieltä kun taas agenttien keskinäisessä vuorovaikutuksessa se on välttämätön kommunikoinnin väline. Käyttöliittymäagentit tukevat käyttäjää ja tarjoavat apua erilaisissa tilanteissa kuten esimerkiksi lomakkeen täytössä tai uuden sovelluksen opettelussa. Käyttöliittymäagenttien avulla voidaan vähentää käyttäjälle aiheutuvaa kuormitusta rutiininomaisissa työtehtävissä. (Nwana 1996) Esimerkkinä käyttöliittymäagentista Tveit (2001) esittää Microsoft Officen avustajan.

Liikkuvat agentit

Liikkuvat agentit (mobile agents) ovat nimensä mukaisesti agentteja, jotka kykenevät vaeltamaan verkoissa (esimerkiksi internetissä). Ne voivat kerätä tietoa käyttäjänsä puolesta ja sitten palata ”takaisin kotiin” suoritettuaan käyttäjänsä toivomat tehtävät. Ne voivat esimerkiksi tehdä lentovarauksia tai huolehtia tietoliikenneverkosta. Liikkuvuus yksinään ei tee liikkuvista agenteista agentteja. Ne ovat agentteja, koska ne toimivat itsenäisesti ja yhteistyössä muiden agenttien kanssa (tosin eri tavalla kuin yhteistyöagentit). Liikkuvien agenttien käyttö voi joissain tapauksissa vähentää tiedonsiirtokustannuksia. Otetaan esimerkiksi tilanne, jossa pitäisi usean kuvan joukosta valita yksi sopiva. Sen sijaan, että siirrettäisiin kaikki kuvat verkon läpi omalle koneelle

tarkastelua varten, lähetetäänkin liikkuva agentti tutkimaan kuvia ja valitsemaan niistä sille annettujen kriteerien perusteella sopiva. Tällaisessa tilanteessa säästyy sekä aikaa että tietoliikenneresursseja. Liikkuvat agentit mahdollistavat myös joustavan ja hajautetun järjestelmäarkkitehtuurin. Liikkuvien agenttien kohdalla on joitakin olennaisia haasteita, jotka saattavat rajoittaa niiden käyttöä. Näitä ovat esimerkiksi autentikointi, käyttäjän yksityisyyden säilyttäminen ja turvallisuus (esimerkiksi suojautuminen viruksilta). (Nwana 1996)

Informaatio/internet-agentit

Informaatio/internet-agenttien (information/internet agents) tehtävänä on auttaa käyttäjäänsä hallitsemaan sitä valtavaa informaatiotulvaa, jota verkosta on saatavilla. Informaatioagentit hallinnoivat, manipuloivat ja lajittelevat useista eri hajautetuista lähteistä saatavilla olevaa tietoa. (Nwana 1996) Ns. web-hämähäkit (web-spiders) ovat esimerkki internet-agenteista. Ne kokoavat dataa internetin hakukoneiden, kuten esimerkiksi Googlen, käyttämiä indeksejä varten (Tveit 2001).

Reaktiiviset agentit

Reaktiiviset agentit (reactive agents) ovat edellä kuvailtuja agenttiluokkia huomattavasti yksinkertaisempia. Niillä ei ole sisäisiä symbolisia malleja toimintaympäristöstään eikä monimutkaisia käyttäytymismalleja. Ne ovat vuorovaikutuksessa toisten agenttien kanssa aivan yksinkertaisimmilla tavoilla. Reaktiiviset agentit suorittavat spesifejä tehtäviä (esimerkiksi havaintojen tekeminen, laskenta jne.) ja niiden kommunikointi muiden agenttien kanssa on vähäistä. (Nwana 1996)

Hybridiagentit

Hybridiagentilla (hybrid agents) tarkoitetaan sellaista agenttia, jolla on kahden tai useamman eri agenttityypin ominaisuudet. Perusteluna hybridiagenttien olemassaololle on esitetty se, että joissain tapauksissa saavutetaan suurempia etuja hybridiagenteilla kuin ratkaisuilla, joissa käytetään useita eri agenttityyppejä erikseen. (Nwana 1996)

Älykkäät agentit

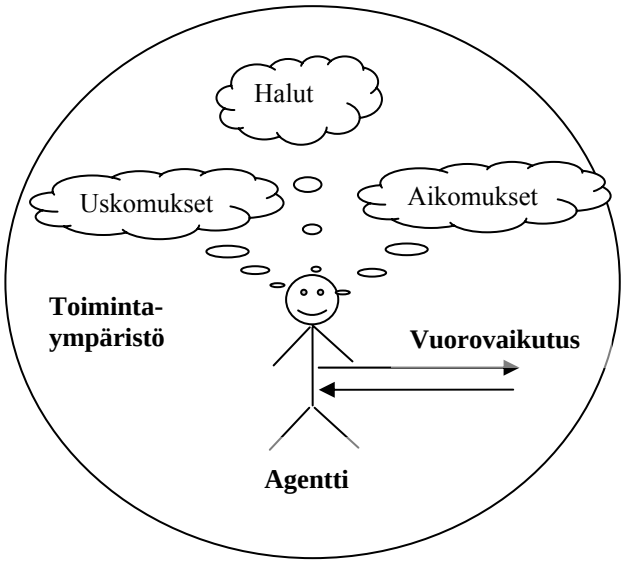
Älykkäitä agentteja (smart agents, intelligent agents) Nwana (1996) ei käsittele artikkelissaan kovin syvällisesti, koska hänen mukaansa todellisia älykkäitä agentteja ei

ole olemassa. Tässä tutkielmassa ei kuitenkaan oteta niin tiukkaa kantaa asiaan vaan käsitellään myös älykkäitä agentteja, koska nykykirjallisuudessa älykkäät agentit ovat saavuttaneet yleisen hyväksynnän. Älykkäällä agentilla on sellaisia älyllisiä ominaisuuksia kuin uskomukset, halut ja aikomukset. Wooldridgen (1997) mukaan parhaiten tunnetut viitekehykset älykkäille agenteille ovat Cohenin ja Levesquen (1990) aikomusteoria (theory of intention) sekä Raon ja Georgeffin (1995) BDI-malli (belief-desire-intention model). Lisäksi Nwanan (1996) mukaan älykkään agentin täytyy kyetä oppimaan toiminnastaan (ks. KUVIO 1).

Cohenin ja Levesquen malli ottaa pohjakseen kaksi käsitettä: uskomukset ja tavoitteet. Muut käsitteet, etenkin aikomuksen (intention) käsite, johdetaan näistä kahdesta peruskäsitteestä. BDI-mallissa taas aikomukset käsitetään peruskäsitteiksi uskomusten ja tavoitteiden lisäksi. (Wooldridge 1997)

Älykkäitä agentteja kutsutaan joskus myös BDI-agenteiksi. BDI on lyhenne sanoista uskomus (**B**elief), halu (**D**esire) ja aikomus (**I**ntention). Agentin uskomukset ovat sen tietoa toimintaympäristöstään. Uskomukset muuttuvat sitä mukaa, kun agentin toimintaympäristö muuttuu. Agentin halut liittyvät järjestelmälle asetettuihin tavoitteisiin. Tavoitteilla voi olla eri prioriteetit, ja agentit voivat priorisoida toimintaansa näiden tavoitteiden mukaisesti. BDI-mallissa agentin aikomukset kuvaavat, mitä toimintaa agentti on parhaillaan suorittamassa tai on suorittanut viimeksi (esimerkiksi viimeisin funktiokutsu). Aikomus-komponentti tuo agentin toimintaan harkintaa. (Rao & Georgeff 1995)

Kuviossa 2 havainnollistetaan hyvin yksinkertaistetusti älykkään agentin toimintaperiaate. Ilman BDI-mallin käsitteitä se sopii myös toisten agenttityyppien havainnollistamiseen. Kuviossa 2 ympyrä kuvaa agentin toimintaympäristöä. Toimintaympäristö voi olla esimerkiksi agenttipohjainen järjestelmä tai sulautettu järjestelmä. Tikku-ukolla kuvataan agenttia. Kuvion nuolet puolestaan kuvaavat agentin vuorovaikutusta ympäristönsä kanssa.



KUVIO 2. Älykäs agentti.

2.2 Agenttien sovellusalueita

Edellä luokiteltiin agentteja niiden ominaisuuksien mukaan. Agentteja voidaan tarkastella myös sovellusalueittain. Agentteja on sovellettu laajasti erilaisille alueille ulottuen teollisuuden prosessinvalvonnasta elektroniseen liiketoimintaan, lentoliikenteen valvontaan, tietoliikenneverkkojen valvontaan sekä elokuviin ja tietokonepeleihin. (Luck ym. 2002, 27) Seuraavaksi tarkastellaan joitakin sovellusalueita hieman lähemmin. Tarkoitus ei ole yksityiskohtaisesti kuvata sovelluksia vaan antaa yleiskatsaus sovellusalueisiin.

2.2.1 Teolliset sovellukset

Yhtenä ensimmäisistä alueista agentteja sovellettiin teollisuuteen. Prosessinvalvonta (process control) on luonnollinen sovellusalue agenteille, sillä prosessinohjailijat ovat itsessään autonomisia ja reaktiivisia järjestelmiä. Reaktiivisella järjestelmällä tarkoitetaan sellaista järjestelmää, joka on jatkuvasti vuorovaikutuksessa ympäristönsä kanssa. Parhaiten tunnettu agenttipohjainen sovellusalusta prosessinvalvontaan on ARCHON (Corera ym. 1996), jota on sovellettu useissa prosessinvalvontajärjestelmissä mukaan lukien sähkönsiirron hallinta ja hiukkaskiihdyttimen valvonta. ARCHON on myös yksi ensimmäisistä moniagenttijärjestelmistä, joita on testattu käytännössä.

Agentteja on sovellettu myös teollisessa tuotannossa (manufacturing). (Jennings & Wooldridge 1998)

Ljungberg ja Lucas (1992) esittelevät artikkelissaan OASIS-nimisen lentoliikenteen valvontajärjestelmän, joka pohjautuu agenttiteknologiaan. Järjestelmä kehitettiin helpottamaan lennonjohtajan työtaakkaa eräällä hyvin vilkkaasti liikennöidyllä lentokentällä. Järjestelmässä jokainen lentokone mallinnetaan omana agenttinaan. Lentokonetta mallintava agentti sisältää lentosuunnitelmaan liittyvää informaatiota, lentokoneen suorituskykymallin sekä tietoa koneen tilasta (kuten onko kone laskeutumassa vai nousemassa). Järjestelmän tarkoitus ei ole korvata ihmistä vaan toimia lennonjohtajan apuna. Callantine (2003) puolestaan on tutkimuksissaan pyrkinyt kehittämään agentteja, jotka voisivat toimia ihmisen sijaisena laajoissa lentoliikenteenvalvontasimulaatioissa.

Yksi luonnollinen kohdealue agenttilähestymistavalle on kuljetusten hallinta (transportation management), koska kohdealueen toiminta on luonteeltaan maantieteellisesti hajautettua. Esimerkiksi Burmeister ym. (1997) kuvailevat artikkelissaan sovellusta, jonka tarkoituksena on hallita yhteiskäyttöautojen käyttöä ja niillä tapahtuvia kuljetuksia. (Jennings ym. 1998)

Gustavsson (1999) esittelee artikkelissaan mielenkiintoisen kokeiluprojektin, jossa agenttiteknologiaa soveltamalla koerakennuksen energiankulutus pieneni 40 %, kun agentit säätelivät rakennuksen valaistusta ja lämpötilaa sen mukaan, miten ihmiset liikkuivat rakennuksessa. Ratkaisu perustuu sähköenergian ja agenttien liikkumiseen samassa verkossa.

2.2.2 Kaupalliset sovellukset

Kun teolliset sovellukset ovat hyvin monimutkaisia ja räätälöityjä sovelluksia, ovat kaupalliset sovellukset puolestaan tarkoitettu suuremmille massoille. Eräs tärkeimmistä kaupallisista sovellusalueista on tiedonhallinta (information management). Nykyisenä tietoverkkojen aikakautena verkoista (etenkin internetistä) tulvivan tiedon seasta on yhä vaikeampi löytää relevanttia informaatiota. (Jennings ym. 1998) Maes (1994) tarttui tähän ongelmaan jo 1990-luvun alkupuolella. Hän esittelee artikkelissaan agentteja,

jotka auttavat suodattamaan informaatiota. Informaation hallintaan tarkoitettuja agentteja ovat muun muassa sähköpostiagentit, uutisryhmistä informaatiota suodattavat agentit sekä kalenterinhallinta-agentit. Sähköpostiagentit voivat priorisoida, tuhota, arkistoida ja lähettää postin eteenpäin käyttäjänsä puolesta. Kalenteriagentit puolestaan voivat järjestää tapaamisia käyttäjänsä puolesta. Yhtenä näiden agenttien ominaisuutena on se, että ne oppivat käyttäjänsä tapoja ja voivat tarvittaessa automatisoida toimintaansa sen mukaan.

Elektroninen liiketoiminta (electronic commerce) on eräs merkittävimmistä sovellusalueista, johon agenttilähestymistapaa voidaan soveltaa. Toistaiseksi agenttipohjaiset sovellukset alueella ovat suhteellisen yksinkertaisia, mutta tulevaisuudessa etenkin yritysten välisen (B2B, business-to-business) elektronisen liiketoiminnan lisääntyminen kasvattaa sovellusten määrää ja monimutkaisuutta. Myös asiakaslähtöisen (B2C, business-to-customer) elektronisen liiketoiminnan merkitys kasvaa jatkuvasti, kun yhä useammilla ihmisillä on pääsy internetin kautta elektronisille kauppapaikoille. Agentit tarjoavat aivan uusia mahdollisuuksia kaupankäyntiin. Agentti voi esimerkiksi hoitaa automaattisesti koko kaupankäyntitapahtuman tavaran etsimisestä maksusuoritukseen asti. Tämä sisältää myös hinnasta neuvottelemisen ja mahdolliset hintavertailut kauppapaikkojen välillä. Myös sellainen tilanne on mahdollinen, että agentilla on käyttäjänsä profiili, jota hyödyntäen agentti tarkkailee markkinoita ja ilmoittaa käyttäjälle tätä mahdollisesti kiinnostavista tuotteista. (He ym. 2003)

Agenttilähestymistapa tarjoaa mahdollisuuksia myös liiketoimintaprosessien hallintaan (business process management). Organisaatioiden johtajat tekevät päätöksiä monista eri lähteistä saadun informaation perusteella (esim. eri osastot). Ideaalitilanteessa päätöksiä tehdään vasta silloin, kun on saatu kaikki relevantti informaatio kokoon päätöksentekoa varten. Yhtenäisen ja merkityksellisen informaation kokoaminen suuressa organisaatiossa on monimutkainen ja aikaa vievä tehtävä. Tästä syystä organisaatiot ovat kehittäneet erilaisia järjestelmiä liiketoimintaprosessiensa hallintaan. (Jennings & Wooldridge 1998) Jennings ym. (2000) kuvaavat artikkelissaan ADEPT-nimisen (Advanced Decision Environment for Process Tasks) agenttipohjaisen järjestelmän liiketoimintaprosessien hallintaan. Perusajatuksena on se, että liiketoimintaprosessi

nähdään kokoelmana itsenäisiä ongelmanratkaisijoita (agentteja), jotka neuvottelevat keskenään ja sopivat tehtävien koordinoinnista. Tämä lisää joustavuutta perinteisiin menetelmiin verrattuna. Järjestelmä on kehitetty yhteistyössä British Telecomin kanssa.

2.2.3 Lääketieteelliset sovellukset

Agentteja on sovellettu lääketieteen alueella jo 1980-luvun lopulla. Jennings ja Wooldridge (1998) kuvailevat Guardian-nimistä järjestelmää, joka on tarkoitettu potilasvalvontaan (patient monitoring). Järjestelmä kehitettiin kirurgisen tehohoidon yksikköön, jossa potilaiden hoito toimii siten, että joukko asiantuntijoita eri alueilta tekevät yhteistyötä potilaiden hoidossa. Järjestelmän tarkoituksena on toimia laajan spesialistijoukon apuna potilaan tilaa valvottaessa. Usein lääkäreillä ei ole mahdollisuutta valvoa potilastaan jatkuvasti vaan seuranta lankeaa sairaanhoitajien tehtäväksi. He eivät kuitenkaan välttämättä osaa tulkita potilaan tilasta tekemiään havaintoja samalla lailla kuin asiantuntija. Järjestelmässä agentit muodostavat kolmitasoisen hierarkian: havainto/toiminta-agentit, päättelyagentit sekä valvonta-agentit. Toinen lääketieteellinen sovellusala, johon agentteja on sovellettu, on terveydenhuolto. (Jennings & Wooldridge 1998)

1990-luvun lopussa ja 2000-luvun alussa lääketieteellisten sovellusten kehittäjien parissa on esiintynyt kasvavaa kiinnostusta agenttiteknologiaa kohtaan. Agenttilähestymistavan käyttöä on harkittu sovellettavaksi esimerkiksi lääketieteellisen tietämyksen hakemiseen internetistä, potilaiden valvontaan ja diagnosointiin tarkoitettuihin päätöstukijärjestelmiin, hajautettuun potilassuunnitteluun sairaalassa, sairaaloiden väliseen koordinointiin siirtoelimien hallinnassa, lääketieteellisen koulutuksen parantamiseen sekä henkilökohtaisten terveystietojen tarkasteluun mobiililaitteella. (Moreno & Carbay 2003)

2.2.4 Viihdeteolliset sovellukset

Viihdeteollisuutta ei vielä muutama vuosi sitten otettu kovin vakavasti tietojenkäsittelytiedeyhteisössä. Viihdesovellukset nähtiin toisarvoisena ns. vakaviin tietokonesovelluksiin verrattuna. Agentteja on sovellettu muun muassa tietokonepeleihin, interaktiiviseen teatteriin sekä virtuaalitodellisuutta mallintaviin

sovelluksiin. Edellä mainitun kaltaiset järjestelmät ovat usein jossain määrin automaattisia ja niissä on paljon animoituja hahmoja, jotka voidaan toteuttaa luonnollisesti käyttäen agenttilähestymistapaa. (Jennings & Wooldridge 1998)

Hyvä esimerkki tietokonepelistä, johon on sovellettu agenttiteknologiaa, on peli nimeltä Creatures, joka tarjoaa simuloidun ympäristön täynnä agentteja, joiden kanssa pelaaja voi toimia vuorovaikutuksessa reaaliaikaisesti. Agentteja on sovellettu myös elokuvateollisuuteen, jossa niitä on käytetty mallintamaan digitaalisia roolihahmoja, jotka käyttäytyvät aidon näyttelijän tavoin. Lähestymistapaa on sovellettu muun muassa takavuosien menestyselokuvassa Titanic sekä suuren suosion saaneen Taru sormusten herrasta -trilogian massiivisissa taistelukohtauksissa. Esimerkiksi elokuvakohtauksissa agenttien oppiminen lisää kohtausten vaikuttavuutta. (Luck ym. 2002, 28-29)

2.3 Yhteenveto

Tässä luvussa käsiteltiin agenttien perusteita ja tutustuttiin agenttien sovellusalueisiin. Ensin tarkasteltiin erilaisia agenttikäsitteitä ja määriteltiin tutkielman kannalta keskeisiä käsitteitä (agentti, agenttipohjainen järjestelmä, moniagenttijärjestelmä). Lisäksi luvussa esitettiin Nwanan (1996) esittämä agenttien luokittelu, jossa agentit luokiteltiin seitsemään luokkaan. Lopuksi luvussa tarkasteltiin sovellusalueita, joihin agenttilähestymistapaa on sovellettu. Merkittävimmät sovellusalueet ovat teolliset, kaupalliset, lääketieteelliset ja viihdeteolliset sovellukset. Tässä luvussa esitettyjen sovellusalueiden lisäksi on vielä muitakin alueita, joihin agentit tarjoavat mahdollisuuksia, mutta tässä tutkielmassa niihin ei ole mahdollista perehtyä tarkemmin.

Suhteellisen laajasta sovellusaluekirjosta huolimatta tosimaailmaan sijoittuvia sovelluksia on kuitenkin toteutettu vasta melko vähän. Agenttipohjainen ohjelmistotuotanto on vasta kehityksensä alkuvaiheessa, ja sillä on vielä monia ongelmia kohdattavanaan ennen kuin lähestymistapa saavuttaa laajan hyväksynnän informaatioteknologian valtavirrassa. Yksi näistä ongelmakohdista on asianmukaiset suunnittelumenetelmät. Agenttipohjaisten sovellusten suunnitteluun on esitetty lukuisia menetelmiä, mutta mikään niistä ei ole saavuttanut valta-asemaa. Lisäksi ei ole saavutettu yhteisymmärrystä siitä, mitä kaikkea menetelmien tulisi kattaa. (Wooldridge

& Ciancarini 2001) Seuraavassa luvussa kartoitetaan kirjallisuudessa esitettyjä agenttipohjaisia menetelmiä ja valitaan tarkempaan kuvaukseen neljä menetelmää.

3 AGENTTIPOHJAISIA MENETELMIÄ

Tässä luvussa esitellään kirjallisuudessa esitettyjä agenttipohjaisia menetelmiä. *Agenttipohjainen suunnittelumenetelmä* tarkoittaa menetelmää, joka on erityisesti tarkoitettu sovellettavaksi agenttipohjaisten järjestelmien ja moniagenttijärjestelmien suunnitteluun ja mallintamiseen. Iglesiasin ym. (1999) mukaan agenttipohjaisia menetelmiä ei ole kehitetty täysin tyhjästä, vaan tutkijat ovat pyrkineet laajentamaan olemassa olevia menetelmiä. Pääasiallisesti vaikutteita on saatu oliopohjaisista menetelmistä ja tietämystekniikan (knowledge engineering) menetelmistä (Iglesias ym. 1999).

Menetelmien tarkastelu jakautuu tässä luvussa kahteen osaan. Ensimmäisessä kohdassa kuvataan tiivistetysti kirjallisuudessa esitettyjä menetelmiä. Kuvauksella selvitetään lähinnä, mihin sovellusalueisiin menetelmiä on kehitetty. Tämän jälkeen seuraavissa kohdissa otetaan lähempään tarkasteluun neljä menetelmää, joita vertaillaan myöhemmissä luvuissa keskenään. Tarkempaan tarkasteluun valittujen menetelmien osalta selvitetään menetelmien peruskäsitteet, vaiheet ja perusmallit. Lopuksi menetelmistä tehdään tiivis yhteenveto.

3.1 Yleiskatsaus agenttipohjaisiin menetelmiin

Erilaisia agenttipohjaisia menetelmiä on kirjallisuudessa esitetty kymmeniä. Taulukkoon 1 on kerätty kirjallisuudesta löytyneitä menetelmiä. Menetelmät on löydetty erilaisten hakusanojen perusteella sekä tutkimalla menetelmäartikkeleiden lähdeluetteloita, joissa on viittauksia muihin menetelmiin. Taulukossa kerrotaan menetelmän nimi, tärkeimmät lähteet, joissa menetelmää kuvataan, sekä sovellusalue (tai käyttötarkoitus), johon menetelmä on tarkoitettu sovellettavaksi. Menetelmät on lajiteltu lähteiden mukaan aikajärjestykseen vanhimman lähteen perusteella. Näin taulukosta nähdään myös se, milloin menetelmän kehittäminen on alkanut ja onko menetelmästä sen jälkeen ilmestynyt uutta materiaalia.

TAULUKKO 1. Kirjallisuudessa esitettyjä agenttipohjaisia menetelmiä.

Menetelmän nimi	Lähteet	Sovellusalue
Menetelmä agenttipohjaisten järjestelmien kehittämiseen yrityksen toimintojen integroimiseksi	Kendall ym. (1995)	Yrityksen toimintojen integrointi (enterprise integration)
MASB	Moulin & Brassard (1996)	Ryhmätyöjärjestelmät
AAII	Kinny & Georgeff (1997)	Yleinen
CoMoMAS	Glaser (1997)	Tietämysjärjestelmät
DESIRE	Brazier ym. (1997)	Yleinen
MAS-CommonKADS	Iglesias ym. (1998)	Tietämysjärjestelmät
Agenttipohjainen menetelmä	Elanmari & Lalonde (1999)	Yleinen
Gaia	Wooldridge ym. (1999); Wooldridge ym. (2000); Zambonelli ym. (2003);	Yleinen
ADEPT	Jennings ym. (2000)	Liiketoimintaprosessien suunnittelu
AUML	Odell ym. (2000); Odell ym. (2001); AUML Web Site (2004)	Yleinen (ei varsinainen menetelmä vaan UML:n laajennos)
AMT	Jo (2001)	Yleinen
Arkkitehtuurikeskeinen lähestymistapa moniagenttijärjestelmien kehittämiseen	Park ym. (2001)	Elektroninen liiketoiminta
MaSE	Wood & DeLoach (2001); DeLoach ym. (2001)	Yleinen
MASSIVE	Lind (2001)	Yleinen
MESSAGE	Evans ym. (2001; Caire ym. (2002)	Yleinen
SODA	Omicini (2001)	Internet-pohjaiset järjestelmät
Tropos	Giunchiglia ym. (2001); Castro ym. (2002); Tropos (2003); Bresciani ym. (2004)	Yleinen
INGENIAS	Gómez-Sanz & Fuentes (2002)	Yleinen

(jatkuu)

TAULUKKO 1. Kirjallisuudessa esitettyjä agenttipohjaisia menetelmiä. (jatkuu)

Menetelmän nimi	Lähteet	Sovellusalue
Moniagenttijärjestelmien extreme-ohjelmointi	Knublauch (2002)	Internet-pohjaiset järjestelmät
PASSI	Cossentino & Potts (2002)	Yleinen
ROADMAP	Juan ym. (2002)	Monimutkaiset avoimet järjestelmät
Viitekehys agenttipohjaisten järjestelmien analyysiin, suunnitteluun ja mallintamiseen	Kavi ym. (2002)	Yleinen
ADELFE	Bernon ym. (2003)	Avoimet ja adaptiiviset järjestelmät
ODAC	Gervais (2003)	Avoimet ja monimutkaiset hajautetut järjestelmät
OPM/MAS	Sturm ym. (2003)	Yleinen
Prometheus	Padgham & Winikoff (2003)	Yleinen
RoMAS	Yan ym. (2003)	Yleinen
SABPO	Dikenelli & Erdur (2003)	Yleinen

Taulukkoon on kerätty kaiken kaikkiaan 28 menetelmän tiedot. Tiedoista nähdään, että ensimmäiset menetelmät kehitettiin 1990-luvun puolessa välissä, mutta vasta 2000-luvun alusta alkaen menetelmien kehittämiseen on alettu toden teolla kiinnittää huomiota. Taulukosta havaitaan, että ensimmäiset menetelmät kehitettiin lähes yksinomaan jotakin tiettyä tarkoitusta varten (esim. Kendall ym. 1995; Moulin & Brassard 1996; Glaser 1997). 2000-luvun alusta lähtien menetelmiä on selvästi enemmän alettu kehittää yleisesti sovellettaviksi. Suurin osa menetelmistä onkin tarkoitettu yleisesti sovellettaviksi. Tulkinta on tehty niin, että joko menetelmän lähteessä on mainittu, että kyseistä menetelmää on tarkoitus soveltaa yleisesti, tai sitten sovellusaluetta ei ole mainittu erikseen. Yleisten menetelmien lisäksi on olemassa joukko menetelmiä, jotka on suunniteltu jotain tiettyä sovellusaluetta silmällä pitäen. Tällaisia sovellusalueita ovat muun muassa tietämysjärjestelmät (Glaser 1997; Iglesias ym. 1998), liiketoimintaprosessien suunnittelu (Jennings ym. 2000), elektroninen liiketoiminta (Park ym. 2001) ja internet-pohjaiset järjestelmät (Omicini 2001; Knublauch 2002). Kaikille taulukon 1 menetelmille ei ole esitetty nimeä. Tällaisissa

tapauksissa on taulukkoon menetelmän nimeksi merkitty suomennos menetelmän lähteen nimestä.

Taulukossa 1 on pyritty mahdollisimman kattavasti esittelemään tämänhetkisessä kirjallisuudessa esitettyjä menetelmiä. Taulukko ei välttämättä ole täydellinen, sillä kaikkien mahdollisten menetelmien kartoittaminen on käytännössä mahdotonta. Edellä esitettyjen menetelmien lisäksi on kirjallisuudessa esitetty erilaisia arkkitehtuureja (Wooldridge & Jennings 1995; Sycara ym. 1996) ja sovelluskehyksiä (Kendall ym. 2000) agenteille. Wooldridge ja Jennings (1995) kuvaavat myös joitakin formaaleja menetelmiä agenttipohjaisten järjestelmien suunnitteluun, toteuttamiseen ja verifiointiin. Niihin ei tässä tutkielmassa kiinnitetä sen enempää huomiota, vaan tarkoitus on tutkia epäformaaleja menetelmiä. Erilaisiin agenttiarkkitehtuureihin viitataan tarpeen mukaan menetelmien esittelyn yhteydessä.

Taulukossa 1 esitettyjen menetelmien lisäksi on olemassa esityksiä, joiden tarkoituksena on laajentaa UML-kieltä agenttipohjaisten menetelmien mallintamiseen. Eniten huomiota näistä laajennoksista on saanut Agent UML eli AUML (Odell ym. 2000; Odell ym. 2001; AUML Web Site 2004), jota käytetään monissa menetelmissä osittaiseen mallintamiseen. AUML on saamansa huomion vuoksi merkitty taulukkoon 1, vaikka se ei varsinainen menetelmä olekaan (kuten ei myöskään itse UML). AUML kehitettiin, koska UML nähtiin joissain tapauksissa riittämättömäksi agenttien ja agenttipohjaisten järjestelmien mallintamiseen (Odell ym. 2000). Odellin ym. (2000) mukaan AUML pyrkii laajentamaan UML:ää, jotta sillä voitaisiin kuvata paremmin agenttien ja moniagenttijärjestelmien monimutkaista toimintaa. AUML:ää kehitetään jatkuvasti ja uusimmat kehitysehdotukset on saatavilla AUML:n internet-sivuilta (AUML Website 2004).

Seuraavissa kohdissa kuvataan tarkemmin neljää taulukossa 1 esiintyvää menetelmää. Tarkemmin kuvattavat menetelmät ovat Gaia (Zambonelli ym. 2003), Tropos (Castro ym. 2002; Bresciani ym. 2004), MaSE (Wood & DeLoach 2001) ja MESSAGE (Evans ym. 2001). Menetelmiä vertaillaan keskenään luvussa 5. Menetelmät on valittu niistä saatavilla olevan materiaalin perusteella. Ensinnäkin materiaalia on oltava riittävästi, jotta menetelmää voidaan esitellä riittävällä tarkkuustasolla. Tämä tarkoittaa sitä, että saatavilla on muutakin materiaalia kuin lyhyitä konferenssijulkaisuja. Valituista

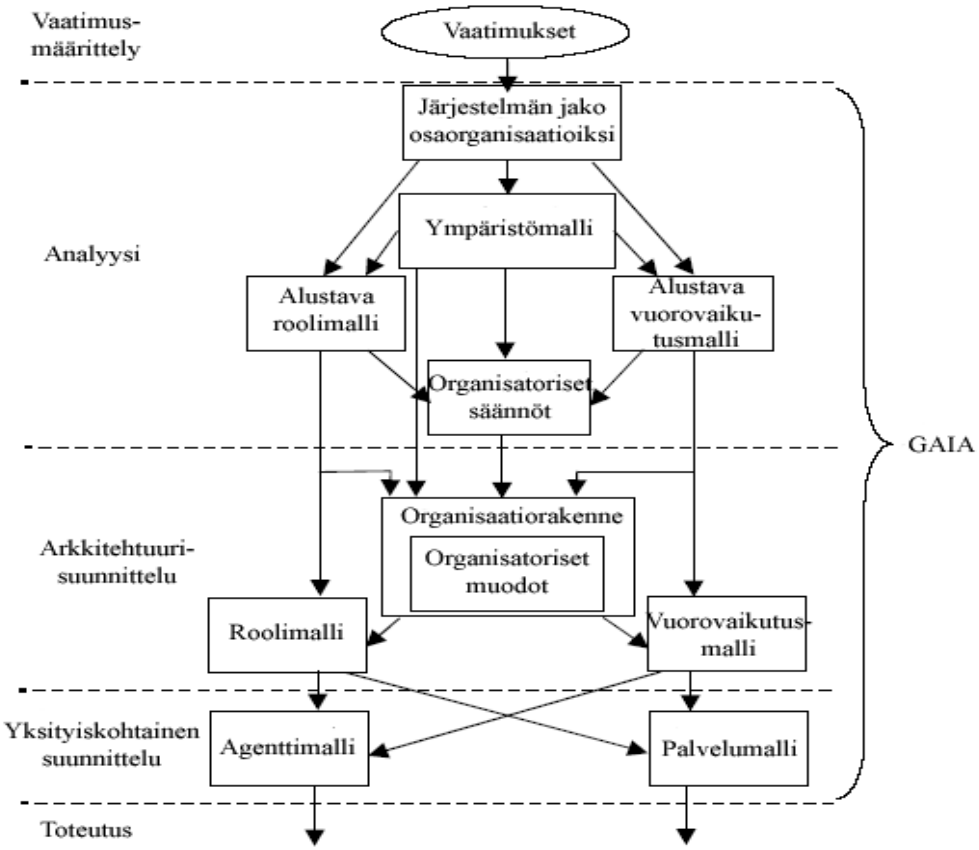
menetelmistä tulee olla esimerkiksi artikkeleita tieteellisissä aikakauslehdissä. Lisäksi materiaalin runsaus kertoo jotakin menetelmän kehitysasteesta ja kypsyydestä. Toisena valintaperusteena on käytetty menetelmien sovellettavuutta. Tällä tarkoitetaan sitä, että menetelmää voidaan soveltaa yleisesti laajalle alueelle eikä vain johonkin yksittäiseen tarkoitukseen.

3.2 Gaia

Gaia sai alkunsa 1990-luvun loppupuolella, kun Wooldridge ym. (1999) esittelivät AOM-menetelmän agenttipohjaisten järjestelmien analyysiin ja suunnitteluun. Wooldridge ym. (2000) täydensivät menetelmää ja samassa yhteydessä se sai nimekseen Gaia. Menetelmä oli vielä kuitenkin kohtalaisen rajoittunut soveltuen ainoastaan suhteellisten pienten agenttipohjaisten järjestelmien suunnitteluun, kuten Wooldridge ym. (2000) artikkelissaan toteavat. Juan ym. (2002) esittävät ROADMAP-menetelmän, joka pyrkii laajentamaan Gaiaa, jotta se soveltuisi monimutkaisten ja avointen järjestelmien suunnitteluun. Zambonelli ym. (2003) esittävät Gaia-menetelmän nykymuodossaan. Artikkelissa esitetään lukuisia parannuksia Wooldridgen ym. (2000) esittämään versioon. Gaian mallit ja niiden väliset suhteet on esitetty Zambonellin ym. (2003) mukaisesti kuviossa 3.

Gaia kattaa ohjelmistotuotannon vaiheista analyysin ja suunnittelun. Vaatimusmäärittelyyn ja toteutukseen liittyviin seikkoihin menetelmä ei ota kantaa. Gaia ottaa lähtökohdakseen organisatorisen näkökulman järjestelmien suunnitteluun ja ohjaa suunnittelijaa ajattelemaan agenttipohjaisten järjestelmien rakentamista organisaatiosuunnittelun prosessina. Tämä tarkoittaa sitä, että järjestelmän jokainen agentti toimii jossakin roolissa ja sillä on tarkoin määritellyt vastuut ja tavoitteet. Rooli määrittelee agentin tehtävän organisaatiossa. Gaian ensimmäinen vaihe on analyysivaihe, jonka tarkoituksena on tuottaa spesifikaatio, jonka pohjalta voidaan edetä suunnitteluvaiheeseen. Analyysivaiheessa muodostettavat mallit ovat ympäristömalli (environmental model), alustava roolimalli (preliminary role model), alustava vuorovaikutusmalli (preliminary interaction model) sekä organisatoriset säännöt (organizational rules). Organisatoriset säännöt muodostetaan jokaiselle osaorganisaatiolle, joista lopullinen järjestelmä muodostuu. Suunnitteluvaihe jakaantuu arkkitehtuurisuunnitteluun ja yksityiskohtaiseen suunnitteluun. Suunnitteluvaiheessa

tarkennetaan alustavat rooli- ja vuorovaikutusmallit lopullisiksi sekä määritellään järjestelmän organisaatiorakenne (organizational structure), agenttimalli (agent model), sekä palvelumalli (services model). (Zambonelli ym. 2003)



KUVIO 3. Gaian mallit ja niiden väliset suhteet (Zambonelli ym. 2003, 336).

Menetelmä lainaa joitakin käsitteitä ja notaatioita oliopohjaisista menetelmistä ja erityisesti Colemanin ym. (1994) esittämästä FUSION-menetelmästä. Gaia on yleiskäyttöinen, joten sitä voidaan soveltaa monille alueille. Menetelmä on tarkoitettu keskikokoisten ja suurten järjestelmien suunnitteluun. (Zambonelli ym. 2003) Seuraavaksi tarkastellaan lähemmin menetelmän analyysivaihetta ja suunnitteluvaihetta.

3.2.1 Analyysivaihe

Analyysivaiheen tavoitteena on muodostaa ymmärrys kehitettävästä järjestelmästä ja sen rakenteesta. Vaiheen ensimmäisenä askeleena mietitään, onko suunniteltavassa järjestelmässä useita (osa)organisaatioita, jotka toimivat itsenäisinä

moniagenttijärjestelminä vuorovaikutuksessa toistensa kanssa. Tämä on suhteellisen helppoa, jos nämä on tunnistettu jo vaatimusmäärittelyssä tai jos järjestelmä matkii reaalimaailman rakennetta. Seuraava askel on ympäristömallin muodostaminen. Tähän tarkoitukseen Gaia ei tarjoa mitään yleisiä malleja, koska eri järjestelmien toimintaympäristöt voivat olla luonteeltaan hyvin erilaisia. Yksinkertaisimmassa muodossaan ympäristömalli voi olla lista resursseja, kuten esimerkiksi muuttujia tai taulukoita, joita agentti hyödyntää toiminnassaan. Lisäksi ympäristöstä voidaan muodostaa yksinkertaisia graafisia esityksiä (Zambonelli ym. 2003)

Gaiassa organisaatio (järjestelmä) nähdään kokoelmana rooleja, joilla on tietyt suhteet toisiinsa ja jotka ovat vuorovaikutuksessa muiden roolien kanssa. Roolit voivat vastata esimerkiksi yksilöitä tai osastoja reaalimaailman organisaatiossa. Analyysivaiheessa määritellään alustavat roolit, joista muodostetaan alustava roolimalli, joka viimeistellään suunnitteluvaiheessa. Rooli määritellään neljän attribuutin avulla: vastuut (responsibilities), luvat (permissions), aktiviteetit (activities) ja protokollat (protocols). Vastuut määrittelevät toiminnallisuuden ja ovat siten ehkä roolin tärkeimpiä attribuutteja. Vastuut jaetaan kahteen kategoriaan: elinkaariominaisuudet (liveness properties) ja turvallisuusominaisuudet (safety properties). Elinkaariominaisuudet kuvaavat sitä asioiden tilaa, joka agenttien tulee toiminnallaan saada aikaan, kun tietyt ehdot ympäristöstä on annettu. Turvallisuusominaisuudet ovat muuttumattomia eli ne pysyvät samana kaikissa tapauksissa. Esimerkiksi turvallisuusominaisuus voisi olla seuraavanlainen: ”Varmista, että liukuhihnan tavaravirta pysyy aina välillä 10–30 tavaraa/min.” Vastuiden toteuttamista varten rooleilla on joukko lupia. Luvat ovat ”oikeuksia”, joita eri rooleilla on. Tietyn roolin luvat ilmaisevat näin ollen ne resurssit, jotka kyseisellä roolilla on käytettävissä vastuidensa toteuttamiseen. Resurssit ovat yleensä informaatioresursseja. Aktiviteetit ovat rooliin liittyviä toimenpiteitä, joita agentti voi suorittaa itsenäisesti ilman vuorovaikutusta muiden agenttien kanssa. Roolille määritellään myös useita protokollia. Protokollat määrittelevät tavan, jolla rooli on vuorovaikutuksessa toisten roolien kanssa. (Zambonelli ym. 2003)

Jokaisesta tunnistetusta roolista muodostetaan roolikaavio, jossa määritellään roolin attribuutit. Roolikaavio sisältää roolin nimen, kuvauksen roolista, protokollat, aktiviteetit, luvat ja vastuut. Esimerkki roolikaaviosta roolille ”TARKASTAJA” on

esitetty kuviossa 4. Roolin kuvaus kirjoitetaan lyhyesti normaalilla kirjakielellä. Protokollista ja aktiviteeteista mainitaan kaaviossa ainoastaan nimet. Protokollista muodostetaan myöhemmin omat määrittelynsä. Aktiviteetit tarkoittavat suurin piirtein samaa kuin metodit oliopohjaisessa mallintamisessa. Aktiviteetit erotetaan protokollista alleviivauksella.

Roolikaavio: TARKASTAJA

Kuvaus:
Tämän roolin tehtävänä on vastaanottaa artikkelit tarkastettavaksi joltakulta konferenssivirkailijalta, tarkastaa artikkeli sekä lähettää takaisin täytetty tarkastuslomake.

Protokollat ja aktiviteetit:
VastaanotaArtikkeli, TarkastaArtikkeli, LähetäTarkastusLomake

Luvat:

reads	*Artikkelit*	*//kaikki vastaanotetut artikkelit*
changes	*TarkastusLomakkeet*	*//yksi lomake jokaiselle artikkelille*

Vastuut
Elinkaari:
TARKASTAJA = (VastaanotaArtikkeli.TarkastaArtikkeli.LähetäTarkastusLomake)$^{max\_lukumäärä}$

Turvallisuus:
- artikkeleiden_lukumäärä = tarkastusten_lukumäärä

KUVIO 4. Esimerkki Gaian roolikaaviosta (Zambonelli ym. 2003, 347).

Gaia käyttää lupien määrittelyyn formaalia notaatiota, joka perustuu Colemanin ym. (1994) FUSION–menetelmän operaatiokaavioiden notaatioon. Kuvion 4 esimerkissä agentilla, jolla on rooli ”TARKASTAJA”, on lupa lukea arvoa ”*Artikkelit”*. Lisäksi sillä on lupa sekä lukea että muuttaa arvoa ”*TarkastusLomakkeet*”. Vastuiden ilmaisemiseen on myös oma tapansa. Elinkaariominaisuudet ilmaistaan erityisellä syntaksilla, joka määrittelee roolin elinkaaren (ks. TAULUKKO 2). Elinkaariominaisuuden yleinen muoto on ”RooliNimi” = *ilmaus*. (Zambonelli ym. 2003) Esimerkiksi kuviossa 4 roolin ”TARKASTAJA” elinkaariominaisuus ilmaistaan seuraavasti:

$$\text{TARKASTAJA} = (\text{VastaanotaArtikkeli.TarkastaArtikkeli.LähetäTarkastusLomake})^{max\_lukumäärä}$$

Taulukossa 2 kuvataan elinkaariominaisuuksien kuvaamisessa käytettävä syntaksi.

TAULUKKO 2. Operaattorit roolien elinkaariominaisuuksien ilmaisemiseen (Zambonelli ym. 2003, 345).

Operaattori	Tulkinta
x.y	ensin x, jonka jälkeen y
x | y	x tai y
x*	x ilmenee 0 kertaa tai useammin
x+	x ilmenee kerran tai useammin
x^{ω}	x ilmenee äärettömän usein
[x]	x on vapaaehtoinen
x || y	x ja y ilmenevät limittäin

Vastuiden turvallisuusominaisuuksille ei ole mitään erityistä syntaksia vaan ne listataan yksinkertaisesti allekkain. Jokainen listan kohta vastaa yhtä ominaisuutta (ks. KUVIO 4). Menetelmässä oletetaan, että turvallisuusvastuut pätevät järjestelmän jokaisessa tilassa. Roolikaavioita voi olla vaikeaa ja osittain mahdotontakin muodostaa täydellisiksi analyysivaiheessa. Alustavat roolikaaviot voivatkin jäädä tässä vaiheessa vajavaisiksi ja ne täydennetään myöhemmin suunnitteluvaiheessa. (Zambonelli ym. 2003)

Analyysivaiheessa muodostetaan myös alustava vuorovaikutusmalli, jossa kuvataan roolien välisiä riippuvuuksia ja vuorovaikutussuhteita. Malli koostuu joukosta protokollamäärityksiä sekä niiden välisistä suhteista. Kuviossa 5 esitetään ”TARKASTAJA”–roolin ”VastaanotaArtikkeli”–protokollan määrittely.

KUVIO 5. Gaian protokollamäärittely (Zambonelli ym. 2003, 348).

Jokaiselle roolien välisen vuorovaikutuksen tyypille tehdään oma protokollamäärittelynsä. Protokollamäärittely koostuu seuraavista attribuuteista: protokollan nimi, aloittajarooli (initiator), vastaanottajarooli (partner), syötteet, tulosteet sekä kuvaus protokollasta. Malli on alustava aivan kuten roolimallikin ja kuten kuviosta

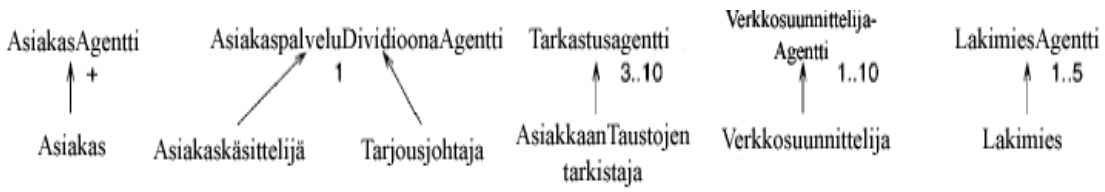
5 nähdään, vuorovaikutuksen aloittajaroolia ei esimerkin tapauksessa määritellä vielä tässä vaiheessa (merkattu kysymysmerkeillä), vaan se tarkennetaan suunnitteluvaiheessa. Määrittelyn syötteet tarkoittavat informaatiota, jota vuorovaikutuksen aloittajarooli käyttää vuorovaikutukseen. Tulosteet taas tarkoittavat informaatiota, jonka vuorovaikutuksen vastaanottajarooli tarjoaa vuorovaikutuksen seurauksena. Protokollamäärityksiä voidaan vapaasti tarkentaa esimerkiksi käyttämällä AUML:n protokollakaavioita (Odell ym. 2001). (Zambonelli ym. 2003)

Analyysivaiheen viimeisenä askeleena on organisatoristen sääntöjen muodostaminen. Organisatoriset säännöt kuvaavat roolien ja protokollien välisiä yleisiä suhteita, jotka eivät välttämättä tule huomioiduiksi alustavissa rooli- ja vuorovaikutuskaavioissa. Organisatoriset säännöt ovat koko järjestelmän (organisaation) vastuita. Ne voidaan rinnastaa yksittäisten roolien vastuisiin. Myös organisatorisille säännöille voidaan määritellä elinkaari- ja turvallisuusominaisuudet. Organisatoriset säännöt voidaan ilmaista käyttämällä samaa formalismia kuin aikaisemmin roolien vastuiden muodostamisessa. (Zambonelli ym. 2003)

3.2.2 Suunnitteluvaihe

Gaian suunnitteluvaihe koostuu kahdesta askeleesta, jotka ovat arkkitehtuurisuunnittelu ja yksityiskohtainen suunnittelu. Arkkitehtuurisuunnittelun vaiheessa muodostetaan järjestelmälle organisaatiorakenne ja tarkennetaan alustavat rooli- ja vuorovaikutusmallit lopullisiksi. Organisaatiorakenteen valitseminen on kriittinen vaihe moniagenttijärjestelmien kehittämisessä, koska se vaikuttaa kaikkiin seuraaviin vaiheisiin. Gaia ei anna tarkkoja ohjeita, kuinka valinta tulisi tehdä, mutta se antaa joitakin yleisiä suuntaviivoja valinnan tekemiseen. Valintaan vaikuttaa esimerkiksi asianmukaisen topologian valinta (ks. tarkemmin Zambonelli ym. 2003, 352). Gaiassa ei määritellä tarkkaa notaatiota organisatoristen sääntöjen kuvaamiseen. Säännöt voidaan kuvata esimerkiksi käyttämällä jotakin formaalia notaatiota tai graafisia esityksiä, kuten AUML-kaavioita. Kun organisatoriset säännöt on määritelty, täydennetään alustaviin rooli- ja vuorovaikutusmalleihin epätarkoiksi jääneet määrittelyt ja lisätään tarvittaessa uusia rooleja ja protokollia. (Zambonelli ym. 2003)

Suunnitteluvaiheen toisena askeleena on yksityiskohtainen suunnittelu, jonka aikana muodostetaan agenttimalli ja palvelumalli. Nämä mallit toimivat järjestelmän toteutuksen pohjana. Agenttimallissa määritellään roolien ja agenttien vastaavuudet. Agentit ovat agenttiluokkien ilmentymiä (vrt. olioluokat ja oliot). Agenttiluokat voidaan saada suoraan roolimallista, mutta suunnittelija voi myös yhdistää useita keskenään samankaltaisia rooleja samaan agenttiluokkaan edellyttäen, että tämä ei vaikuta järjestelmän tehokkuuteen ja että organisatorisia sääntöjä ei rikota. Myöskään agenttimallin määrittämiseen Gaia ei esitä mitään erityistä notaatiota, vaan antaa melko vapaat kädet. Agenttimalli muodostetaan esimerkiksi käyttämällä jotakin yksinkertaista kaaviota (KUVIO 6) tai taulukkoa, jossa määritellään agenttien ja roolien vastaavuudet. Agenttimallissa voidaan myös dokumentoida, kuinka monta ilmentymää kustakin luokasta järjestelmässä esiintyy. (Zambonelli ym. 2003)



KUVIO 6. Esimerkki Gaian agenttimallista (Wooldridge ym. 2000, 304)

Kuvion 6 esimerkissä roolit ovat alarivissä ja niitä vastaavat agenttiluokat ylärivissä. + -symboli tarkoittaa, että agentteja on yksi tai enemmän ja *-symboli tarkoittaa, että ilmentymiä on 0 tai enemmän. Merkintä *m..n* Tarkoittaa, että järjestelmässä esiintyy vähintään *m* ja enintään *n* kyseisen luokan ilmentymää. Kuviossa 6 on kaksi roolia (”Asiakaskäsittelijä” ja ”Tarjousjohtaja”) yhdistetty yhteen agenttiluokkaan (”AsiakaspalveluDivisioonaAgentti”).

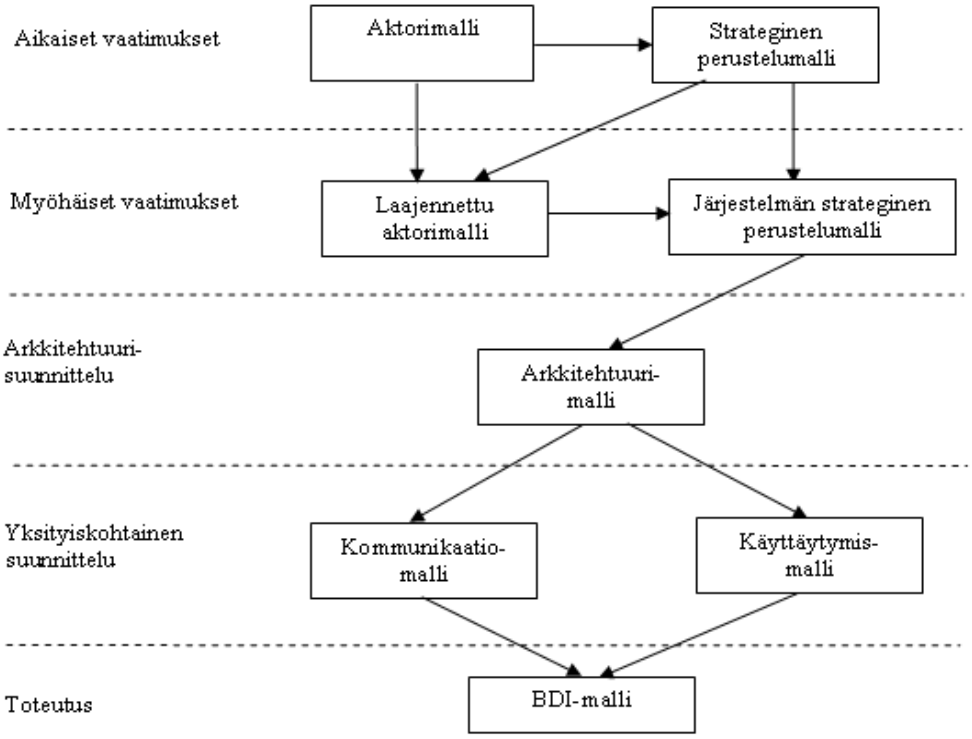
Agenttimallin muodostamisen jälkeen on vuorossa palvelumallin muodostaminen. Palvelumallin tarkoituksena on tunnistaa eri rooleihin tai agenttiluokkiin liittyvät palvelut ja määritellä näiden palveluiden ominaisuudet. Oliotermein ajateltuna palvelu vastaisi metodia. Palvelut eivät kuitenkaan ole samalla lailla saatavilla muille agenteille kuin olioiden metodit ovat saatavilla muille olioille. Jokaisella analyysivaiheessa tunnistetulla aktiviteetilla tulee olla vastine palvelumallissa, joskaan kaikilla palveluilla ei tarvitse olla vastinetta analyysivaiheen roolimallissa. Jokaiselle palvelulle

määritellään syötteet, tulosteet, esiehdot sekä jälkiehdot ja ne voidaan määritellä esimerkiksi taulukkomuodossa (ks. Wooldridge ym. 2000, 305). Palveluiden syötteet ja tulosteet saadaan analyysivaiheen protokollamäärittelyistä ja ympäristömallista. Esi- ja jälkiehdot ovat palveluiden rajoitteita. Ne saadaan roolimallien turvallisuusominaisuuksista sekä organisatorisista säännöistä. Palvelumalli ei kuvaa, kuinka palvelut tulisi toteuttaa, vaan tämä jätetään ohjelmoijan harkinnan varaan. (Zambonelli ym. 2003)

3.3 Tropos

Tropos on agenttipohjainen menetelmä, joka painottaa voimakkaasti vaatimusmäärittelyn merkitystä järjestelmän kehittämisen alkuvaiheissa (Castro ym. 2002). Se on näin lähtökohdiltaan hieman erilainen kuin muut tutkielmassa esiteltävät menetelmät. Tropos on tässä luvussa esiteltävistä menetelmistä kaikkein kattavin. Se kattaa ohjelmistotuotannon vaiheet vaatimusmäärittelystä toteutukseen. Troposin mukaan järjestelmän suunnittelu jakautuu kolmeen vaiheeseen, jotka ovat vaatimusmäärittely, suunnittelu ja toteutus. Vaatimusmäärittely jakautuu edelleen kahteen askeleeseen. Ensin määritellään aikaiset vaatimukset (early requirements), joissa keskitytään eri osapuolten vaatimuksiin. Tämän jälkeen tulee myöhäisten vaatimusten (late requirements) määrittely, jossa määritellään kehitettävän järjestelmän toiminnalliset ja ei-toiminnalliset vaatimukset. Myöhäisten vaatimusten määrittelyn voidaan katsoa vastaavan ainakin osittain muissa menetelmissä esiintyvää analyysivaihetta. Tähän asiaan palataan myöhemmin menetelmien vertailun yhteydessä. Myös menetelmän suunnitteluvaihe jakautuu kahteen askeleeseen. Ensimmäinen on arkkitehtuurisuunnittelu, jonka tuloksena saadaan malli järjestelmän rakenteesta. Toinen askel on yksityiskohtaisen suunnittelu, jonka aikana tarkennetaan arkkitehtuurisuunnittelun aikana tehtyjä suunnitelmia. Viimeisenä vaiheena on toteutusvaihe, jonka tarkoituksena on toteuttaa järjestelmä käyttämällä jotakin BDI-agenttiarkkitehtuuria. Troposin käyttämät käsitteet pohjautuvat aikaisessa määrittelyvaiheessa käytettäviin käsitteisiin ja ne on omaksuttu Yun (1995) väitöskirjassaan esittämästä i* - mallinnuskehyksestä. Käytettäviä käsitteitä ovat aktori (actor), tavoite (goal), yleistavoite (softgoal), tehtävä (task), ja resurssi (resource).

(Castro ym. 2002) Kuviossa 7 esitetään Troposin mallit ja niiden väliset suhteet Castroa ym. (2002) mukaillen.



KUVIO 7. Troposin mallit ja niiden väliset suhteet Castroa ym. (2002) mukaillen.

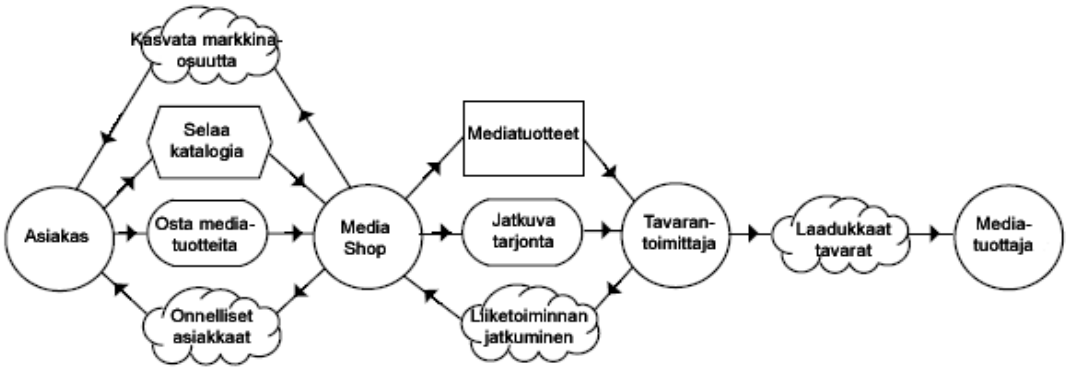
Troposille on myös esitetty ainakin yksi laajennus. Mouratidis ym. (2003) esittävät artikkelissaan Tropos-menetelmän laajennoksen (Secure Tropos), jossa kiinnitetään erityistä huomiota järjestelmän turvallisuusvaatimuksiin. Troposin kehittäjät ovat luoneet myös formaalin kielen (Formal Tropos), jolla voidaan täydentää menetelmän graafisia esityksiä (Castro ym. 2002). Tässä yhteydessä laajennoksia ei ole mahdollista käsitellä tämän enempää. Seuraavaksi kuvataan Troposia vaatimusmäärittelyvaiheen, suunnitteluvaiheen ja toteutusvaiheen osalta tarkemmin.

3.3.1 Vaatimusmäärittelyvaihe

Vaatimusmäärittely on tärkeä vaihe uutta järjestelmää kehitettäessä, koska silloin määritellään se, mitä järjestelmältä halutaan. Vaatimusmäärittelyn aikana tehtäviä virheitä on myöhäisemmissä vaiheissa vaikea korjata ja korjauskustannukset ovat suuret. Troposin vaatimusmäärittely jakaantuu kahteen askeleeseen: aikaisiin

vaatimuksiin (early requirements) ja myöhäisiin vaatimuksiin (late requirements). (Castro ym. 2002)

Castro ym. (2002) esittelevät artikkelissaan menetelmää havainnollisen esimerkin avulla. Media Shop on kauppa, joka myy erilaisia media-artikkeleita kuten kirjoja, aikakauslehtiä, CD-levyjä, videonauhoja jne. Tarkoituksena on kehittää järjestelmä, jonka avulla asiakas voi tilata tuotteita internetin välityksellä. Järjestelmälle annetaan nimeksi Medi@. Esimerkkiä käytetään myös tässä yhteydessä kuvausten havainnollistamiseen. Aikaisten vaatimusten analysointi keskittyy eri osapuolten vaatimuksiin. Tässä vaiheessa määritellään järjestelmän ei-toiminnalliset vaatimukset eikä oteta vielä kantaa järjestelmän toimintoihin. Ensin muodostetaan aktorimalli (KUVIO 8), jossa eri osapuolet esitetään aktoreina, joiden tavoitteiden saavuttaminen riippuu toisista aktoreista. Kehitettävää järjestelmää ei vielä tässä vaiheessa sijoiteta kuvioon. Castro ym. (2002) käyttävät aktorimallista nimitystä strateginen riippuvuusmalli (strategic dependency model), jota käytetään i*-mallissa. Tässä yhteydessä käytetään kuitenkin asioiden yksinkertaistamiseksi Giorginin ym. (2001) ja Brescianin ym. (2004) käyttämää termiä aktorimalli, koska se kuvaa paremmin mallin olemusta.

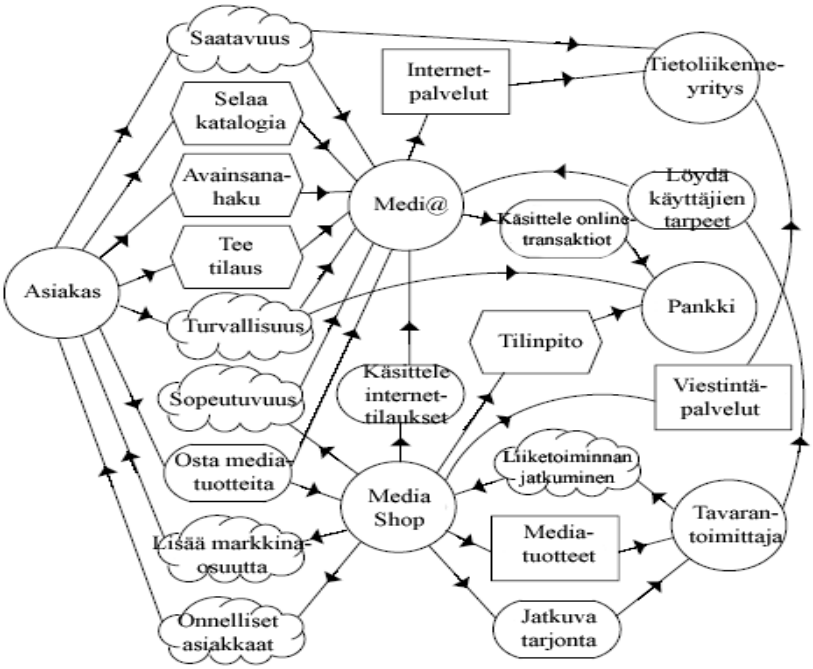


KUVIO 8. Esimerkki Troposin aktorimallista (Castro ym. 2002, 368).

Kuviossa ympyrä kuvaa aktoria, pyöristetty suorakulmio tavoitetta, pilvikuvio yleistavoitetta, suorakulmio resurssia ja kuusikulmio tehtävää. Bresciani ym. (2004) käyttävät tehtävän käsitteen sijasta suunnitelman käsitettä. Käsitteiden merkitys on sama. Nuolet ilmaisevat riippuvuussuhteita. Esimerkiksi asiakas on riippuvainen medioita myyvästä kaupasta saavuttaakseen tavoitteensa, joka on ostaa mediatuotteita.

Yleistavoitteet ovat samankaltaisia kuin ”normaalit” tavoitteet, mutta niitä ei voida määritellä täsmällisesti. Esimerkiksi onnellisuus on subjektiivinen käsite ja eri ihmisillä on erilaiset käsitykset onnellisuudesta. Kun kaikki osapuolet ja niiden tavoitteet on tunnistettu, muodostetaan strateginen perustelumalli (strategic rationale model, ks. Castro ym. 2002, 369, kuva 3), jonka avulla mallinnetaan se, kuinka aktorimallissa määritellyt tavoitteet saavutetaan muiden aktoreiden avulla. Malli muodostetaan käyttämällä keino-tavoiteanalyysiä (means-ends analysis). Malli on verkko, jonka solmuja ovat tavoitteet, tehtävät, resurssit ja yleiset tavoitteet. Lisäksi mallissa on kahdenlaisia linkkejä: keino-tavoitelinkkejä sekä tehtävien osituslinkkejä (task decomposition links), joiden avulla tehtäviä voidaan osittaa pienemmiksi kokonaisuuksiksi. (Castro ym. 2002)

Myöhäisten vaatimusten analyysin tuloksena saadaan vaatimusmäärittely, joka kuvaa kaikki kehitettävän järjestelmän toiminnalliset ja ei-toiminnalliset vaatimukset. Tässä vaiheessa laajennetaan aktorimallia siten, että kehitettävä järjestelmä lisätään kaavioon yhdeksi tai useammaksi aktoriksi. Kuviossa 9 esitetään laajennettu aktorimalli. ”Medi@”-aktori esittää kehitettävää järjestelmää.



KUVIO 9. Troposin laajennettu aktorimalli (Castro ym. 2002, 371)

Malli on muuten samanlainen kuin aiemmin muodostettu aktorimalli, mutta se on laajempi, koska järjestelmän myötä mukaan tulee uusia tavoitteita ja riippuvuuksia. Tässä vaiheessa muodostetaan myös samankaltainen strateginen perustelumalli kuin aikaisten vaatimusten määrittelyvaiheessa. Tällä kertaa keino-tavoiteanalyysi kohdistuu ainoastaan kehitettävään järjestelmään ulkoisten aktoreiden sijasta. (Castro ym. 2002)

3.3.2 Suunnitteluvaihe

Troposin suunnitteluvaihe koostuu kahdesta askeleesta, jotka ovat arkkitehtuurisuunnittelu ja yksityiskohtainen suunnittelu. Arkkitehtuurisuunnittelun tarkoituksena on muodostaa järjestelmän rakenteesta malli, joka kuvaa, kuinka järjestelmän eri komponentit toimivat yhdessä. Troposin yhteydessä on kehitetty organisatorisia arkkitehtuurityylejä, jotka sopivat yleisiä arkkitehtuureja paremmin dynaamisten ja hajautettujen sovellusten, kuten moniagenttijärjestelmien suunnitteluun. Nämä arkkitehtuurit perustuvat organisaation johtamisen tutkimuksesta tuleviin käsitteisiin. (Castro ym. 2002) Tämän tutkielman puitteissa eri arkkitehtuurityylejä ei ole mahdollista tarkastella lähemmin.

Arkkitehtuurisuunnittelun ensimmäisenä tehtävänä on valita vaihtoehtoisista arkkitehtuurityyleistä sopiva kehitettävää järjestelmää varten. Seuraavaksi muodostetaan valitun arkkitehtuurimallin mukainen laajennettu aktorikaavio, johon lisätään edellisissä kaavioissa olevien aktorien lisäksi uusia aktoreita ja ositetaan olemassa olevia tarpeen mukaan. Näin saadaan yksityiskohtaisempi kuvaus järjestelmän toiminnallisuudesta. Tämän jälkeen määritellään, kuinka agentit voivat täyttää aktoreille asetetut tavoitteet sosiaalisten mallien (social patterns) avulla. Tässä vaiheessa suunnittelija voi käyttää apunaan erilaisia agenttimalleja (agent patterns, Aridor & Lange 1998), joissa on ennalta määritelty mahdollisia suunnitteluratkaisuja (vrt. suunnittelumallit (design patterns) olioparadigmassa). Sosiaalisten mallien analysoinnin perusteella voidaan määritellä joukko pystyvyyksiä (capabilities) agenteille, jotka ovat mukana mallissa. Pystyvyydet ilmaisevat sen, miten agentti kykenee saavuttamaan sille asetetut tavoitteet. (Castro ym. 2002)

Yksityiskohtaisen suunnittelun vaiheessa mallinnetaan aktorien (agenttien) kommunikointi ja käyttäytyminen. Tähän tarkoitukseen on useita vaihtoehtoisia tapoja.

Kommunikaation mallintamiseen käytetään jotakin agenttikommunikointikieltä kuten KQML:ia (Genesereth & Ketchpel 1994) tai FIPA ACL:ia (FIPA 2003). Lisäksi tässä vaiheessa voidaan käyttää mallintamiseen AUML:ia tai Troposia varten laajennettua UML-kieltä. Kaavioiden tekemisessä hyödynnetään arkkitehtuurisuunnittelun aikana muodostettuja kaavioita. Esimerkiksi laajennetun aktorikaavion pohjalta voidaan muodostaa UML-luokkakaavio, jossa käytetään Troposia varten muodostettuja stereotyyppejä. Samalla lailla voidaan menetellä strategisten periaatemallien kanssa. (Castro ym. 2002) Esimerkkejä kaavioista löytyy muun muassa lähteistä Castro ym. (2002, 381-383) ja Giorgini ym. (2001)

Agenttien (tai aktorien) sisäisen prosessoinnin mallintamiseen voidaan käyttää suunnitelmakaavioita (plan diagrams). Ne ovat matalimman mallinnustason kaavioita ennen toteutusvaihetta. Jokainen tunnistettu suunnitelma esitetään yhtenä suunnitelmakaaviona. Suunnitelmakaaviot ovat tarkkoja malleja siitä, miten BDI-agentin tulisi toimia saavuttaakseen tavoitteensa. Suunnitelmakaavio koostuu kolmenlaisista solmuista: alkutila, lopputila ja sisäiset tilat. Lisäksi kaavioon mallinnetaan siirtymät tilojen välillä.(Castro ym. 2002)

3.3.3 Toteutusvaihe

Troposin mukaisesti suunniteltu järjestelmä voidaan toteuttaa esimerkiksi käyttäen JACK Intelligent Agents – ympäristöä (AOS 2003). Aluksi suunnitteluvaiheen mallien käsitteet muunnetaan BDI-mallin käsitteiksi. Tämän jälkeen BDI-mallin käsitteet puolestaan muunnetaan JACK-ympäristön käsitteiksi. Taulukossa 3 on esitetty käsitteiden vastaavuus.

TAULUKKO 3. Troposin, BDI-mallin ja JACK-ympäristön käsitteiden vastaavuus Castroa ym. (2002) ja AOS:ia (2003) mukaillen.

Tropos	BDI-malli	JACK
aktori	agentti	agentti
resurssit	uskomukset	uskomusjoukkorelaatiot
tavoitteet ja yleistavoitteet	halut	tapahtumat (erityisesti BDIGoalEvent)
tehtävät	aikomukset	suunnitelmat

JACK on laajennus Javaan. Se on agenttipohjainen kehittämisympäristö, jonka pohjana on aiemmin esitelty BDI-malli älykkäille agenteille. JACK-agentit ovat itsenäisiä ja

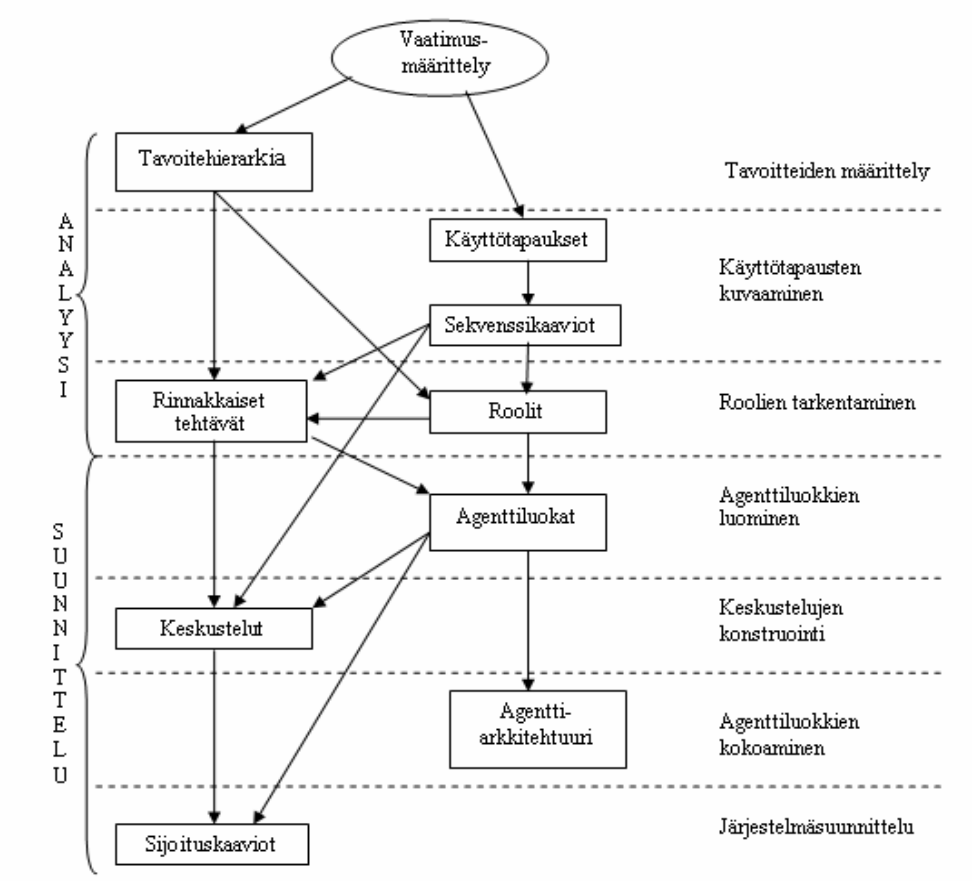
älykkäitä ohjelmistokomponentteja, joilla on eksplisiittiset tavoitteet tai tapahtumat (events), joista niiden pitää selvitä. Tapahtumat vastaavat BDI-mallin haluja (desires). Agentit ohjelmoidaan suunnitteluvaiheessa määriteltyjen suunnitelmakaavioiden avulla. Suunnitelmat vastaavat BDI-mallin aikomuksia (intentions). Jokainen suunnitelma kuvaa, kuinka tavoite saavutetaan eri olosuhteissa. Uskomukset (beliefs) puolestaan kuvaavat agentin senhetkistä tietoa toimintaympäristöstään. JACK tarjoaa viisi peruskäsitettä agenttien ohjelmointiin: agentit (agents), kyvyt (capabilities), uskomusjoukkorelaatiot (beliefset relations), tapahtumat (events) ja suunnitelmat (plans). Kyvyt ovat komponentteja, jotka edustavat agentin toiminnallisuutta. Uskomusjoukkorelaatiot varastoivat agentin uskomukset ja datan tietokannan tauluihin. Tapahtumat (events) tunnistavat olosuhteet ja viestit, joihin agentti voi vastata. Tapahtumat ovat kaikkien agentin toimintojen lähde. Jos ei ole tapahtumia, agentti ei tee mitään. Suunnitelmat puolestaan ovat ohjeita, joita agentti noudattaa saavuttaakseen tavoitteensa. Suunnitelma kuvaa sarjan toimintoja, jotka suoritetaan tapahtuman (event) ilmetessä. (AOS 2003)

3.4 MaSE

Multiagent Systems Engineering, eli MaSE, on yleiskäyttöinen menetelmä moniagenttijärjestelmien kehittämiseen (DeLoach ym. 2001). MaSE on tässä luvussa esiteltävistä menetelmistä ehkä kaikkein lähimpänä perinteisiä suunnittelumenetelmiä. Menetelmä ei ota kantaa siihen, ovatko agentit älykkäitä vai eivät, eikä se myöskään ohjaa järjestelmän kehittämistä minkään tietyn agenttiarkkitehtuurin pohjalta. MaSE kattaa ohjelmistotuotannon vaiheista analyysin ja suunnittelun. Vaatimusmäärittelyyn menetelmä ei ota kantaa lainkaan. (DeLoach ym. 2001) Kuviossa 10 on esitetty MaSEn mallit eri vaiheissa sekä mallien väliset suhteet.

Analyysivaihe koostuu kolmesta askeleesta, jotka ovat tavoitteiden määrittely (capturing goals), käyttötapausten kuvaaminen (applying use cases) sekä roolien tarkentaminen (refining roles). Suunnitteluvaihe koostuu neljästä askeleesta: agenttiluokkien luominen (creating agent classes), keskustelujen konstruointi (constructing conversations), agenttiluokkien kokoaminen (assembling agent classes) ja järjestelmäsuunnittelu (system design). Kuvion 10 suorakulmiot kuvaavat eri vaiheissa muodostettavia malleja ja nuolet kuvaavat mallien välisiä suhteita. Käytännössä

siirtyminen vaiheiden välillä ei tapahdu niin suoraviivaisesti kuin kuvio 10 antaa ymmärtää, vaan malleja on mahdollisuus tarkentaa iteratiivisesti. (DeLoach ym. 2001)



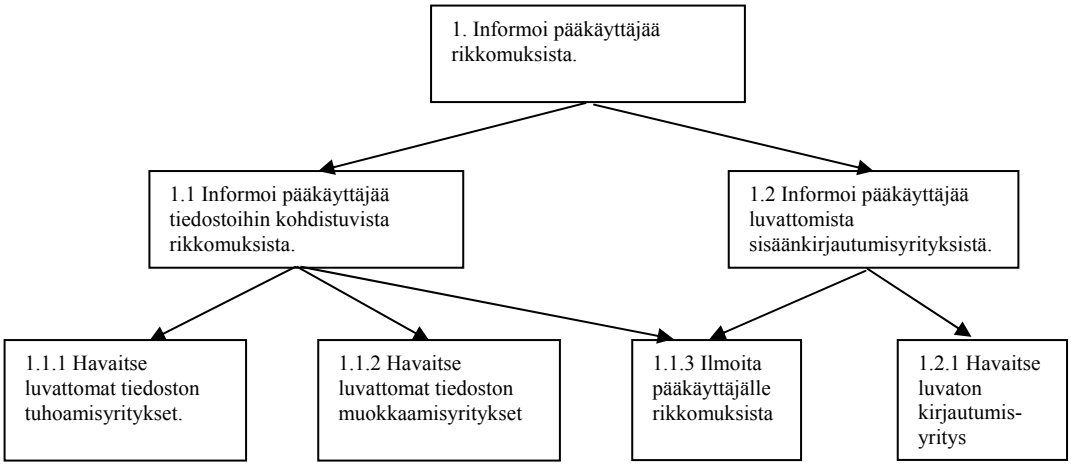
KUVIO 10. MaSEn mallit ja niiden väliset suhteet (Deloach ym. 2001, 233).

MaSE tarjoaa myös työkalutuen menetelmän vaiheiden mallintamiseen. Mallintamista varten on kehitetty agentTool-niminen työkalu (DeLoach & Wood, 2001), joka tukee menetelmän kaikkia seitsemää askelta. Seuraavassa perehdytään tarkemmin MaSE:n vaiheisiin havainnollisten esimerkkien avulla.

3.4.1 Analyysivaihe

MaSE:n analyysivaiheen tarkoituksena on saada selville roolit, joiden tehtävät kuvaavat, mitä järjestelmän tulee tehdä. Jokainen rooli on vastuussa jonkin tavoitteen saavuttamisesta. Analyysivaiheen ensimmäinen askel on tavoitteiden määrittely. MaSE:ssa tavoitteita tarkastellaan järjestelmän näkökulmasta. Järjestelmän kokonaistavoitteena on täyttää järjestelmän käyttäjän vaatimukset. Tavoitteet saadaan

aikaisemmin tuotetusta järjestelmän vaatimusmäärittelystä. Kun tavoitteet on tunnistettu, ne järjestetään hierarkiaksi (KUVIO 11). (DeLoach ym. 2001) Tässä yhteydessä hierarkia ei välttämättä edellytä puumaista rakennetta.



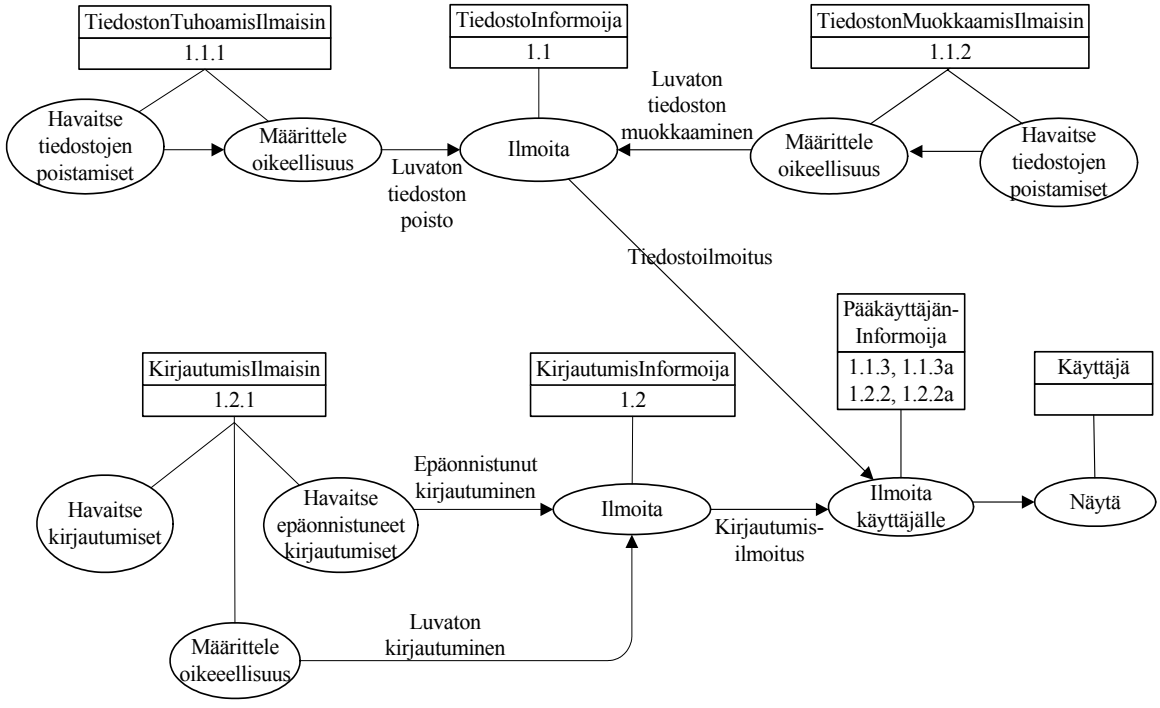
KUVIO 11. MaSE:n tavoitehierarkiakaavio (DeLoach ym. 2001, 236)

Kuvion 11 esimerkissä päätavoitteena on ilmoittaa järjestelmän pääkäyttäjälle järjestelmään kohdistuvista rikkomuksista. Tässä vaiheessa ei olla vielä kiinnostuneita siitä, miten tavoitteet saavutetaan. Tavoitteet voidaan edelleen jakaa osatavoitteisiin, kuten kuviosta 11 nähdään. Jokaisen osatavoitteen pitää tukea ylemmän tason tavoitteita. Tavoitteiden osittaminen jatkuu niin kauan, kunnes osittamisen tuloksena saadaan joku toiminnallinen vaatimus tavoitteen sijaan. (DeLoach ym. 2001)

Analyysivaiheen toisena askeleena on käyttötapausten kuvaaminen. Tavoitteena on kerätä käyttötapaukset vaatimusmäärittelystä (tai mistä tahansa muusta lähteestä) ja muodostaa niiden pohjalta sekvenssikaavioita, joiden perusteella voidaan tunnistaa alustavasti järjestelmän roolit ja viestintäpolut. Käyttötapaukset ovat nimensä mukaisesti kuvauksia siitä, millä tavalla järjestelmää voidaan käyttää. Käyttötapaukset kuvataan sanallisesti. Käyttötapauksia kuvattaessa tulisi kuvata sekä positiivisia että negatiivisia käyttötapauksia. Positiiviset käyttötapaukset kuvaavat, kuinka järjestelmä toimii normaalitapauksissa ja negatiiviset käyttötapaukset puolestaan kuvaavat, miten järjestelmä toimii virhetilanteissa. Käyttötapausten pohjalta muodostetaan sekvenssikaaviot. Sekvenssikaavio kuvaa tapahtumasarjan, joka ilmenee roolien välillä. Roolit tunnistetaan käyttötapauksista. Sekvenssikaavio on samankaltainen UML:n

sekvenssikaavion kanssa kuitenkin niin, että MaSE:n sekvenssikaavion laatikot kuvaavat järjestelmässä esiintyviä rooleja. Viivojen väliset nuolet puolestaan kuvaavat roolien välisiä tapahtumia (tai viestejä). Normaalisti yhdestä käyttötapauksesta muodostetaan yksi sekvenssikaavio. Sekvenssikaavioita voi olla enemmänkin, jos sama käyttötapaus voidaan suorittaa useammalla eri tavalla. (DeLoach ym. 2001)

Analyysivaiheen viimeisenä askeleena on roolien tarkentaminen. Tarkoituksena on muuntaa tavoitteet ja sekvenssikaaviot rooleiksi ja niihin liittyviksi tehtäviksi. Roolit muodostavat perustan suunnitteluvaiheen agenttiluokkien määrittelylle ja esittävät järjestelmän tavoitteita suunnitteluvaiheessa. Normaalisti tavoitteiden muuntaminen rooleiksi onnistuu suoraviivaisesti niin, että yksi tavoite vastaa yhtä roolia. Joissakin tapauksissa voi olla hyödyllisempää muodostaa yksi rooli vastaamaan useampaa tavoitetta. Esimerkiksi kuvion 11 tavoitehierarkiasta voidaan muodostaa roolit TiedostoInformoija (1.1), KirjautumisInformoija (1.2), TiedostonTuhoamisIlmaisin (1.1.1) jne. Käyttöliittymää varten on hyvä muodostaa oma roolinsa. Myös tietokannat, tiedostot ja vanhat järjestelmät (legacy systems) saattavat vaatia omat roolinsa. Roolimääritykset esitetään roolikaaviossa (KUVIO 12), jossa kuvataan myös roolien tehtävien välinen vuorovaikutus. (DeLoach ym. 2001)

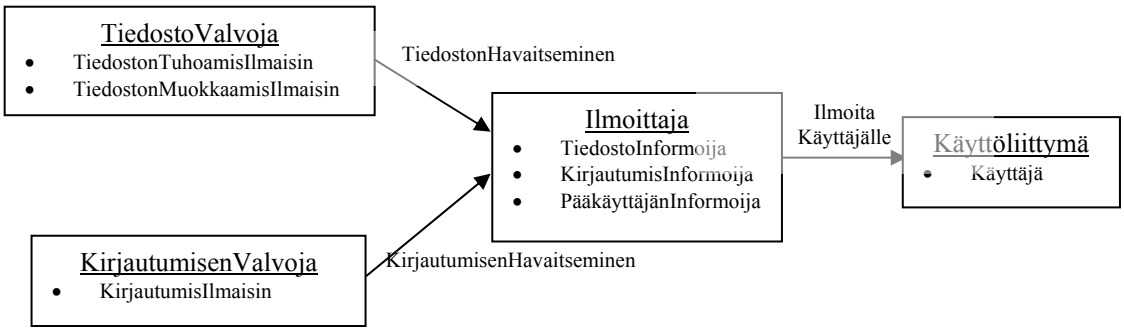


KUVIO 12. MaSE:n roolimalli (DeLoach ym. 2001, 242)

Kuviossa 12 roolit kuvataan suorakulmioina ja roolien tehtävät kuvataan ellipseinä. Viivat tehtävien välissä kuvaavat tehtävien välisiä viestintäprotokollia. Nuolet kuvaavat viestinnän aloittaja/vastaanottajasuhteita. Roolin tehtävät johdetaan yleensä tavoitteista, joista kyseinen rooli on vastuussa. Tehtäviä voidaan myös osittaa useammiksi tehtäviksi. Samalla roolilla voi olla useita samanaikaisesti suoritettavia tehtäviä, jotka määrittelevät roolin käyttäytymisen. Rinnakkaiset tehtävät määritellään graafisesti rinnakkaisten tehtävien kaaviossa, joka kuvataan tilakaaviona. Tehtäviä on kahdentyyppisiä: pysyviä ja hetkellisiä. (DeLoach ym. 2001)

3.4.2 Suunnitteluvaihe

Suunnitteluvaiheen ensimmäisenä askeleena on agenttiluokkien luominen. Agenttiluokat muodostetaan analyysivaiheessa määritellyistä rooleista. Agenttiluokka on malli järjestelmässä esiintyville agenteille. Agentti on agenttiluokan ilmentymä samalla tavalla kuin olio on olioluokan ilmentymä. Saman luokan eri ilmentymät voivat ajonaikana toimia eri rooleissa. Tässä vaiheessa määritellään myös keskustelut, joihin eri agenttiluokat ottavat osaa. Agenttiluokat ja keskustelut kuvataan agenttiluokkakaaviossa (KUVIO 13), joka on samankaltainen olioluokkakaavion kanssa. Näiden kaavioiden välillä on kuitenkin kaksi merkittävää eroa. Agenttiluokkia ei määritellä attribuuttien ja metodien avulla, vaan roolien avulla. Toinen ero on agenttiluokkien suhteiden semantiikka. Kaikki luokkien väliset suhteet ovat keskusteluja, jotka käydään kahden agenttiluokan välillä. (DeLoach ym. 2001)



KUVIO 13. MaSE:n agenttiluokkakaavio (DeLoach ym. 2001).

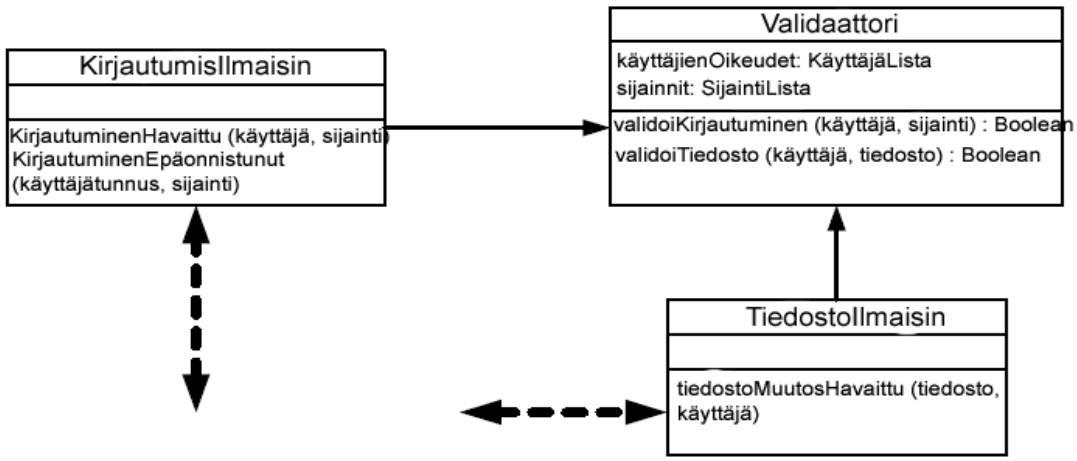
Kuviossa laatikot kuvaavat agenttiluokkia, joille on määritelty nimi (alleviivattuna), sekä roolit, joihin agentti on osallisena. Nuolet kuvaavat keskusteluja agenttien välillä.

Nuolien suunta osoittaa keskustelun aloittajasta ja päättyen vastaanottajaan. (DeLoach ym. 2001)

MaSE:n suunnitteluvaiheen seuraava askel on keskustelujen konstruointi. Tähän saakka on määritelty, että keskusteluja esiintyy, mutta niiden yksityiskohtiin ei ole puututtu. Keskustelu määrittelee kahden agentin välisen koordinointiprotokollan. Keskustelu muodostuu kahdesta viestintäluokkakaaviosta (conversation class diagram). Sekä keskustelun aloittajalle että vastaanottajalle muodostetaan omat kaavionsa. Viestintäluokkakaavio on tilakaavio (ks. tarkemmin DeLoach ym. 2001, 249), joka määrittelee keskustelun tilat kahden agenttiluokan välillä. Analyysivaiheessa määriteltyjen rinnakkaisten tehtävien yksityiskohdat ovat tärkeitä keskustelujen yksityiskohtia määriteltäessä. Keskustelujen suunnittelussa on otettava huomioon myös kestävyys- ja virheidensietonäkökohdat. (DeLoach ym. 2001)

Kolmantena askeleena suunnittelussa on agenttiluokkien kokoaminen. Tässä vaiheessa määritellään agenttien sisäiset asiat kahden askeleen avulla. Ensin määritellään agenttiarkkitehtuuri, jonka jälkeen määritellään komponentit, joista arkkitehtuuri muodostuu. Suunnittelija voi valita, haluaako hän suunnitella oman arkkitehtuurinsa vai valita jonkun valmiiksi määritellyistä arkkitehtuureista (esim. BDI). Suunnittelija voi myös kehittää arkkitehtuurin komponentit itse tai käyttää valmiita komponentteja. Komponentit koostuvat joukosta attribuutteja, metodeja ja monimutkaisilla komponenteilla voi olla myös aliarkkitehtuureja. Komponentit kuvataan agenttiarkkitehtuurikaaviossa (KUVIO 14). (DeLoach ym. 2001)

Kuviossa 14 on esitetty agenttiarkkitehtuuri ”TiedostoValvoja”-agentille. Esimerkkikuviossa laatikot kuvaavat arkkitehtuurikomponentteja. Ohuet nuolet kuvaavat komponenttien välistä näkyvyyttä ja paksut katkoviivanuolet kuvaavat yhteyksiä ulkoisiin resursseihin. Ulkoisia resursseja voivat olla esimerkiksi toiset agentit, sensorit tai tietokannat. Komponenttien sisäinen käyttäytyminen voidaan määritellä esimerkiksi tilakaavioiden avulla. Arkkitehtuurin ja komponenttien sisäisten määrittelyjen tulee olla yhteneväisiä aiemmin määriteltyjen keskustelujen kanssa. (DeLoach ym. 2001)



KUVIO 14. Esimerkki MaSE:n agenttiarkkitehtuurista (DeLoach ym. 2001).

MaSE:n suunnitteluvaiheen viimeisenä askeleena on järjestelmäsuunnittelu. Tässä vaiheessa muodostetaan sijoituskaaviot (deployment diagrams), joissa kuvataan järjestelmän agenttien lukumäärät, tyypit ja sijainnit (ks. DeLoach ym. 2001, 252). Tämä askel on ehkä MaSE:n yksinkertaisin, koska suurin osa työstä on tehty jo aikaisemmissa vaiheissa. Toteutuksen aikana tarvittavan informaation lisäksi sijoituskaaviot mahdollistavat järjestelmän hienosäätämisen. Suunnittelija voi esimerkiksi määritellä agenttien ja tietokoneiden konfiguraatioita maksimoidakseen prosessoritehon tai verkon kaistaleveyden. (DeLoach ym. 2001)

3.5 MESSAGE

MESSAGE (Methodology for Engineering Systems of Software Agents) on agenttipohjainen menetelmä, joka pohjautuu vahvasti UML:n käyttöön menetelmän eri vaiheiden mallintamisessa. Menetelmä kattaa järjestelmän kehittämisestä analyysin ja suunnittelun. MESSAGE laajentaa UML-kieltä tietämystason käsitteillä. (Evans ym. 2001)

MESSAGE:n peruskäsitteet ovat agentti, organisaatio, rooli, resurssi, tehtävä, vuorovaikutus, sekä tavoite. Organisaatio on agenttien ryhmä, joka työskentelee yhdessä saavuttaakseen yhteisen tavoitteen. Roolin käsite on hyvin samanlainen kuin Gaiassa. Agentti voi hoitaa useita rooleja, ja useat agentit voivat puolestaan hoitaa yhtä roolia. Resursseilla tarkoitetaan ei-itsenäisiä kohteita, kuten tietokantoja ja järjestelmän

ulkoisia ohjelmia, joita agentit hyödyntävät tavoitteidensa saavuttamisessa. Tehtävä on tietämystason toimintoyksikkö, jolle määritellään esi- ja jälkiehdot. Jos tehtävän esiehto on validi, voidaan olettaa, että myös jälkiehto pätee tehtävän suorittamisen jälkeen. Myös MESSAGE:n vuorovaikutuksen käsite on samankaltainen Gaian vastaavan käsitteen kanssa. Vuorovaikutuksessa on vähintään kaksi osallistujaa. Vuorovaikutusprotokolla puolestaan määrittelee vuorovaikutuksen tavan ja säännöt. (Evans ym. 2001)

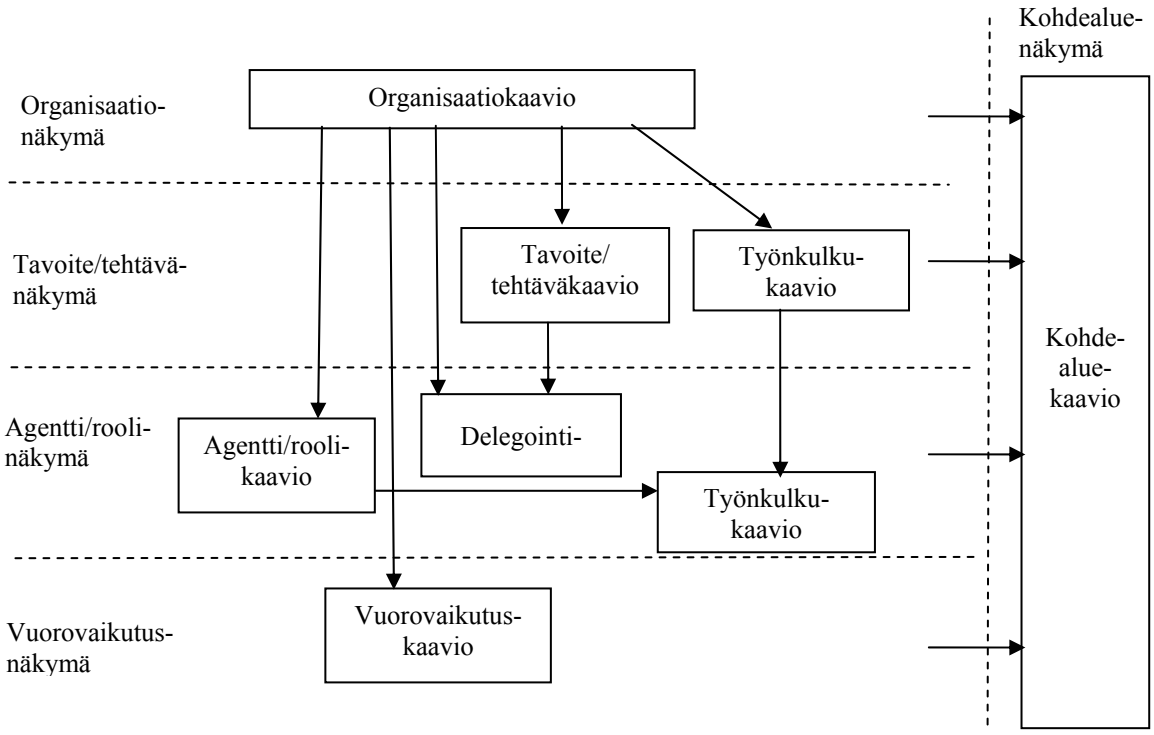
Seuraavaksi tarkastellaan MESSAGE:n analyysivaihetta ja suunnitteluvaihetta Evansin ym (2001) esimerkkien avulla.

3.5.1 Analyysivaihe

MESSAGE käyttää eri vaiheissa luotavista malleista nimitystä näkymä (view). Analyysivaiheen tuloksena saadaan malli, joka koostuu viidestä näkymästä, jotka ovat organisaationäkymä (organization view), tavoite/tehtävänäkymä (goal/task view), agentti/roolinäkymä (agent/role view), vuorovaikutusnäkymä (interaction view) sekä kohdealuenäkymä (domain view). Näkymät puolestaan muodostuvat erilaisista kaavioista. Menetelmän analyysiprosessi etenee tasoittain, eli ensin muodostetaan tason 0 malli, jonka jälkeen edetään tasolle 1 jne. 0-tason mallissa määritellään järjestelmä ja sen suhteet ympäristöönsä. Järjestelmän sisäisiin toimintoihin ei tällä tasolla puututa lainkaan. (Evans ym. 2001) Kuviossa 15 esitetään MESSAGE:n näkymät ja analyysivaiheen kaaviot Evansia ym. (2001) mukaillen. Kaavioiden väliset nuolet kuvaavat sitä, miten kaaviot vaikuttavat muihin kaavioihin. Muista näkymistä kohdealuekaavioon suuntautuvat nuolet merkitsevät sitä, että kohdealuekaaviota muokataan mallinnusprosessin aikana jatkuvasti tarpeen mukaan ja siihen vaikuttavat kaikki muut näkymät. Jokaista näkymää tarkennetaan sellaisella tarkkuustasolle kuin on tarpeellista.

Ensimmäiseksi muodostetaan organisaatio- ja tavoite/tehtävänäkymät, jotka toimivat syötteinä muodostettaessa agentti/rooli- ja kohdealuenäkymiä. Tasolla 1 määritellään järjestelmän käyttäytyminen ja tarpeen mukaan muodostetaan lisää tasoja. Malleja tarkennettaessa on varmistettava mallien johdonmukaisuus edellisiin tasoihin nähden. Mallien tarkentamiseen voidaan käyttää erilaisia lähestymistapoja.

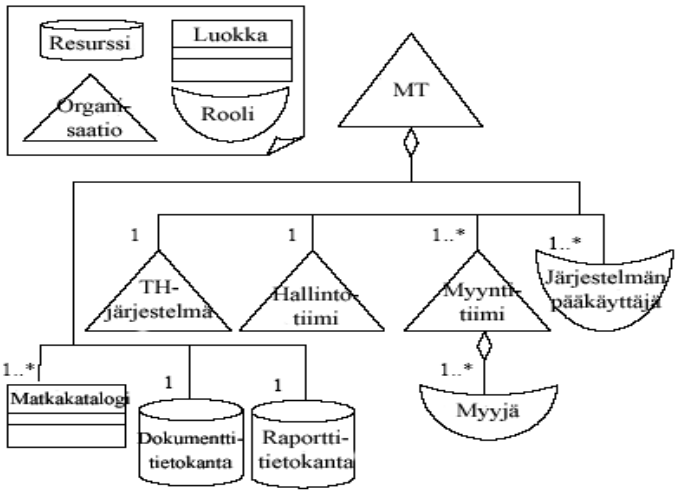
Organisaatiokeskeinen lähestymistapa keskittyy analysoimaan järjestelmän yleisiä ominaisuuksia kuten rakennetta, järjestelmän tarjoamia palveluja, globaaleja tehtäviä, päärooleja sekä resursseja. Agenttikeskeinen lähestymistapa puolestaan keskittyy tunnistamaan agentit, joita tarvitaan järjestelmän toiminnallisuuden tuottamiseen. Tavoitteiden ja tehtävien osittamiseen perustuva lähestymistapa perustuu järjestelmän toiminnalliseen osittamiseen. MESSAGE ei ohjeista suunnittelijaa käyttämään mitään tiettyä lähestymistapaa, vaan antaa vapauden valita sen, mikä sopii suunnittelijalle parhaiten. Analyysimallin muodostamiseen käytetään syötteinä vaatimusmäärittelydokumenttia sekä sitä tukevaa taustadokumentaatiota, kuten organisaatiokaavioita ja liiketoimintaprosessien määrittelyjä. (Evans ym. 2001)



KUVIO 15. MESSAGE:n analyysivaiheen näkymät ja kaaviot Evansia ym. (2001) mukaillen.

Seuraavassa esitellään MESSAGE:n eri näkymiä Evansin ym. (2001) kuvaaman esimerkkijärjestelmän avulla. Esimerkkijärjestelmän tarkoituksena on avustaa matkustajaa matkustukseen liittyvissä toimenpiteissä niin, että matkustaminen olisi mahdollisimman mukavaa ja vaivatonta. Järjestelmän tulee itsenäisesti analysoida käyttäjän matkavaatimukset ja tunnistaa sopivat palvelupaketit (lennot, hotellihuone

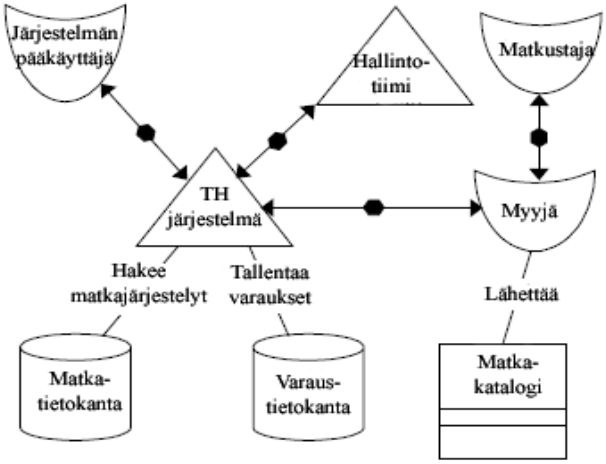
jne.) eri matkatoimistoilta (MT). Lisäksi järjestelmän tulee käyttäjän suostumuksella tehdä varaukset ja valvoa, että järjestelyt sujuvat suunnitelmien mukaan. Kuviossa 16 on esitetty järjestelmän 0-tason organisaatiokaavio, jossa kuvataan rakenteelliset suhteet.



KUVIO 16. MESSAGE:n 0-tason organisaatiokaavio (rakenne). (Evans ym. 2001, 24)

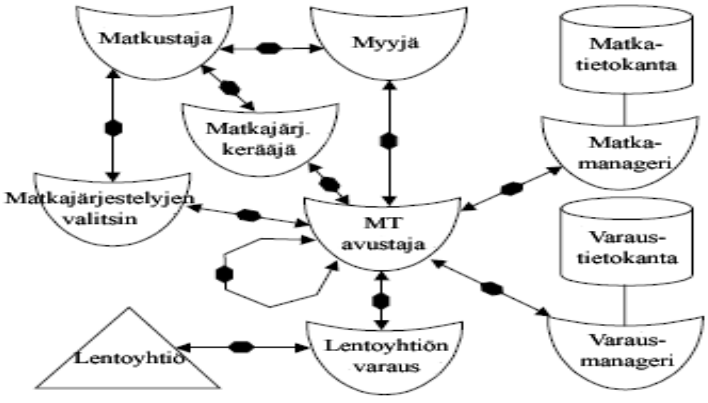
Kuvion 16 kaavio vastaa UML:n luokkakaaviota, jossa symbolit on muutettu vastaamaan MESSAGE:n käsitteitä. TH-järjestelmä tarkoittaa suunnittelun kohteena olevaa tietämyksen hallintajärjestelmää (knowledge management system). 0-tason kaaviossa järjestelmä nähdään organisaationa, jota analysoidaan tarkemmin tason 1 kaavioissa. Kuviossa 16 on esitetty rakenteelliset suhteet. Lisäksi voidaan muodostaa toinen kaavio, jossa esitetään entiteettien yhteyssuhteet (KUVIO 17).

Kuviossa 17 esitetään 0-tason organisaatiokaavio, joka kuvaa entiteettien välistä vuorovaikutusta. Vuorovaikutusta ilmaisee nuolikuvio. Kehitettävä järjestelmä on vuorovaikutuksessa kahden roolin ja kahden ulkoisen resurssin kanssa. Lisäksi järjestelmä on yhteydessä hallintotiimiin, joka laatii laskut ja lähettää ne asiakkaille. Organisaationäkymän jälkeen laaditaan tavoite/tehtävänäkymä. 0-tason tehtäväkaavio muistuttaa ajatukseltaan MaSE:n tavoitehierarkiakaaviota. Ensin muodostetaan järjestelmän päätavoite, joka ositetaan tarpeen mukaan alatavoitteiksi (ks. Evans ym. 2001, 25, kuva 8). Tehtäväkaavion lisäksi tai vaihtoehtoisena ratkaisuna voidaan käyttää myös työnkulkukaavioita, jotka kuvaavat, kuinka tietty palvelu toteutetaan järjestettyjen tehtävien avulla. (Evans ym. 2001)



KUVIO 17. MESSAGE:n 0-tason organisaatiokaavio (yhteyssuhteet). (Evans ym. 2001, 24)

0-tason kaavioiden muodostamisen jälkeen siirrytään tasolle 1. Tällä tasolla keskitytään järjestelmän itsensä ja sen toiminnallisuuden mallintamiseen. 0-tason organisaatiokaavioihin lisätään järjestelmän toiminnallisuutta kuvaavia rooleja. 0-tason kaavioissa järjestelmä esitetään organisaationa, joka on vuorovaikutuksessa muiden entiteettien kanssa. Tasolla 1 järjestelmä esitetään joukkona rooleja (tai agentteja), jotka toimivat käyttäjänsä puolesta saavuttaakseen järjestelmän päätavoitteen (matkan järjestäminen esimerkkitapauksessa). (Evans ym. 2001) Kuviossa 18 esitetään 1-tason organisaatiokaavio.



KUVIO 18. MESSAGE:n 1-tason organisaatiokaavio (yhteyssuhteet). (Evans ym. 2001, 26)

Tasolla 1 muodostetaan myös agentti/roolinäkymä. Näkymän esittämiseen voidaan käyttää erilaisia kaavioita, kuten delegointirakennekaaviot (delegation structure diagrams), työnkulkukaaviot (workflow diagrams) ja tekstipohjaiset agentti/roolikaaviot. Delegointirakennekaavio kuvaa, kuinka aiemmin määritellyt organisaation tavoitteet jaetaan organisaation agenttien/roolien kesken. Vaihtoehtoisesti voidaan käyttää työnkulkukaaviota, jossa kuvataan roolit, joille määritellään tietyt tehtävät järjestelmän kokonaistavoitteen saavuttamiseksi. Jokaiselle roolille laaditaan tekstimuotoinen roolikaavio (vrt. Gaian roolikaaviot), joka esitetään taulukkomuodossa. Analyysivaiheessa roolikaavion sisältämä informaatio on hyvin epäformaalia ja siksi käytetään mieluummin tekstimuotoista kuvausta kuin graafista notaatiota. Roolikaaviossa määritellään roolin nimi, tavoitteet, roolin kyvyt, roolin tietämys ja uskomukset sekä vaatimukset roolia esittävälle agentille. (Evans ym. 2001)

Vuorovaikutusnäkymä kuvaa nimensä mukaisesti agenttien/roolien välistä vuorovaikutusta. Näkymässä kuvataan se, mitkä roolit kommunikoivat keskenään. Vuorovaikutusnäkymä muodostuu joukosta vuorovaikutuskaavioita (ks. Evans ym. 2001, 28, kuva 12), joita tarkennetaan iteroiden sitä mukaa, kun roolien välillä ilmenee uusia kommunikointitarpeita. Vuorovaikutuksen yksityiskohdat voidaan kuvata joko tässä vaiheessa tai myöhemmin suunnitteluvaiheessa. Yksityiskohdat voidaan kuvata esimerkiksi käyttämällä AUML:n sekvenssikaavioita. (Evans ym. 2001)

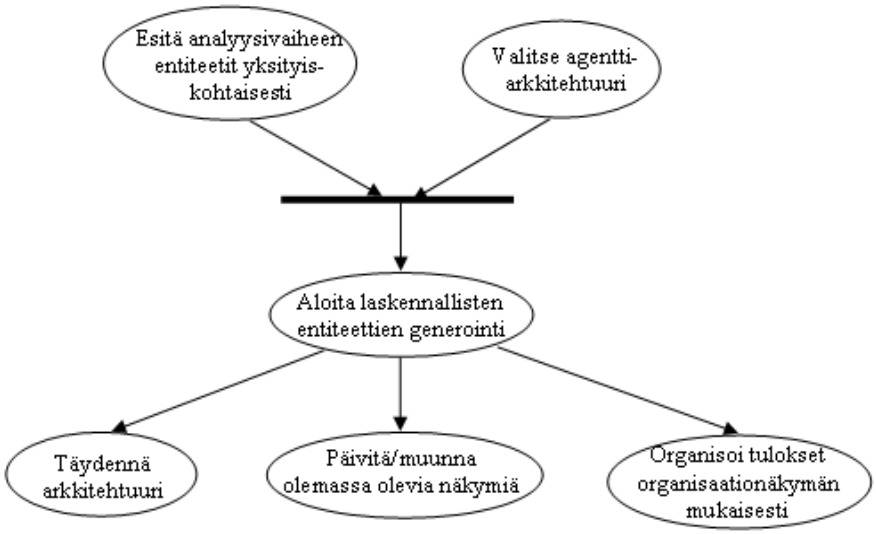
Analyysivaiheen viimeisenä näkymänä muodostetaan kohdealuenäkymä. Se voidaan esittää käyttämällä perinteisiä UML-luokkakaavioita, joissa luokat esittävät kohdealuespesifisiä käsitteitä ja nimetyt assosiaatiot kohdealuespesifisiä suhteita. Kohdealuenäkymää rakennetaan rinnakkain muiden näkymien kanssa ja siihen lisätään käsitteitä ja suhteita sitä mukaa, kun niitä esiintyy muissa näkymissä. (Evans ym. 2001)

3.5.2 Suunnitteluvaihe

MESSAGE esittelee kaksi vaihtoehtoista tapaa suunnitteluvaiheen läpiviemiseksi. Ensimmäinen lähestymistapa keskittyy moniagenttijärjestelmän organisaatioon ja agenttiarkkitehtuuriin. Agentti voidaan nähdä osajärjestelmänä, jolla on sisäinen arkkitehtuuri, joka määrittelee agentin komponenttien väliset suhteet. Komponentit muodostetaan transformoimalla ja tarkentamalla analyysivaiheen malleja.

Komponenttien (esim. agentin kontrolli) käyttäytyminen voi olla monimutkaista (esim. BDI-agentit) tai sitten hyvin yksinkertaista (reaktiiviset agentit). Suunnitteluprosessi etenee organisaatiomallin ohjaamana ja siinä määritellään agenttien vastuut, vuorovaikutus ja sosiaalinen tietämys. Toinen suunnittelulähestymistapa on agenttialustasuuntautunut. Oletuksena on, että jokainen agentti voidaan toteuttaa luokkana. Tätä lähestymistapaa käytetään etenkin sellaisissa tilanteissa, joihin on tarjolla valmis kehittämisympäristö. Lähestymistapa on pätevä silloin, kun agentit ovat suhteellisen yksinkertaisia. Ongelmana lähestymistavassa on se, miten agenttien vuorovaikutus järjestetään. (Evans ym. 2001, 29) Tässä yhteydessä ei ole mahdollista esitellä molempia lähestymistapoja, joten seuraavassa keskitytään ensiksi mainittuun lähestymistapaan.

Organisaatiokeskeiseen lähestymistapaan perustuva suunnitteluprosessi koostuu erilaisista transformointitehtävistä ja analyysivaiheen mallien tarkentamisesta. Kuviossa 19 esitetään MESSAGE:n suunnitteluvaiheen aktiviteetit.



KUVIO 19. MESSAGE:n suunnitteluvaiheen päätehtävät (Evans ym. 2001, 30).

Ensimmäisenä tehtävänä on analyysivaiheen entiteettien tarkentaminen yksityiskohtaisiksi. Tämä tarkoittaa luokkien, rajanpintojen, attribuuttien, metodien ja UML-kaavioiden määrittelyä käyttäen perinteisiä ohjelmistotuotannon tekniikoita. Esimerkiksi analyysivaiheen vuorovaikutusnäkymän tarkentamiseen käytetään UML:n

sekvenssikaavioita ja käyttötapausten toteutuksia. Tämä tapahtuu niin, että ensin luodaan jokaiselle vuorovaikutusnäkymän ilmentymälle käyttötapaus, jonka jälkeen käyttötapausta tarkennetaan tarvittavalle tasolle. Tämän lisäksi jokaisesta käyttötapauksesta tehdään sekvenssikaaviot, jotka tarkennetaan niin yksityiskohtaisiksi kuin mahdollista. Kaavioihin määritellään tehtävien ja resurssien suhteet sekä yksityiskohtaisesti määritellyt viestit. Ainoastaan rooleja ja agentteja ei määritellä tarkemmin. Ne käsitellään ”mustina laatikkoina”. (Evans ym. 2001)

Seuraavaksi valitaan sopiva agenttiarkkitehtuuri. Valintakriteerit perustuvat agenttien toiminnallisuuden vaatimuksiin, jotka on määritelty analyysivaiheessa. Kognitiivisia arkkitehtuureita tarvitaan silloin, kun agenttien välillä esiintyy monimutkaista kommunikointia tai kun agenttien täytyy kyetä oppimaan. Yksinkertaisemmissa tapauksissa voidaan käyttää reaktiivisia arkkitehtuureita. Kirjallisuudessa esitetään useita arkkitehtuureja (Wooldridge & Jennings 1995; Nwana 1996), joihin on syytä perehtyä huolellisesti ennen valinnan tekemistä. Jokaisella niistä on omat erikoisuutensa kuten sosiaaliset kyvyt, liikkuvuus ja älykkyys. (Evans ym. 2001)

Arkkitehtuurin valinnan jälkeen sitä on täydennettävä. Suunnittelijan tulee tuottaa sovellusagentteja muuntamalla komponentteja, joita on tarjolla alkuperäisessä arkkitehtuurissa. Tämän vaiheen eteneminen riippuu paljolti valitusta arkkitehtuurista. Arkkitehtuurin täydentämisen jälkeen voi olla tarpeen päivittää tai muuttaa analyysivaiheessa muodostettuja näkymiä. Kun malleja muutetaan, on huolehdittava, että analyysi- ja suunnitteluvaiheen yhdenmukaisuus säilyy. Lopuksi suunnittelun tulokset organisoidaan vastaamaan organisaationäkymää. Näin järjestelmästä saadaan rakenteellinen esitys. Agenttien ryhmittely ja agenttien väliset suhteet toimivat hyvänä lähtökohtana järjestelmän arkkitehtuurin määrittelylle. (Evans ym. 2001)

MESSAGE pyrkii olemaan teknologiasta riippumaton eikä se ota suunnittelun pohjaksi mitään tiettyä teknologiaa. Esimerkiksi käytettävän agenttialustan ei tulisi vaikuttaa suunnitteluun organisaatiokeskeistä lähestymistapaa sovellettaessa.

3.6 Yhteenveto

Tässä luvussa on kuvattu kirjallisuudessa esitettyjä agenttipohjaisia menetelmiä. Löytyneet menetelmät kerättiin taulukkoon, jonka jälkeen valittiin neljä menetelmää (Gaia, Tropos, MaSE ja MESSAGE) lähempään tarkasteluun. Valinta tehtiin menetelmistä löytyneen lähdemateriaalin perusteella. Kaikista neljästä tarkemmin kuvatusta menetelmästä on saatavilla muutakin materiaalia kuin konferenssijulkaisuja. Gaiasta, Troposista ja MaSE:sta löytyy artikkeleita tieteellisistä aikakauslehdistä ja MESSAGE:sta puolestaan on olemassa laajempi kuvaus menetelmän kehittämisprojektin tuloksena. Tämän lisäksi esitellyt menetelmät ovat suhteellisen ”valmiita”, eli ne kattavat useita vaiheita ohjelmistotuotantoprosessista.

Menetelmistä esitettiin peruskäsitteet, vaiheet ja perusmallit. Lisäksi menetelmien malleista esitettiin yksityiskohtaisia esimerkkejä ja kuvioita menetelmien ymmärtämisen helpottamiseksi. Kaikki esitellyt menetelmät kattavat ohjelmistotuotannon vaiheista analyysin ja suunnittelun. Lisäksi Tropos kattaa myös vaatimusmäärittelyn ja toteutuksen. Gaia ottaa järjestelmien kehittämiseen vahvasti organisaatiokeskeisen näkökulman. Järjestelmä ajatellaan organisaationa, joka koostuu joukosta rooleja, joille määritellään tietyt tehtävät, tavoitteet ja vastuut. Tropos puolestaan painottaa vahvasti vaatimusmäärittelyä. Keskeisenä käsitteenä on aktori. MaSE ja MESSAGE ovat lähempänä perinteisiä ohjelmistotuotannon menetelmiä ja etenkin MESSAGE pohjautuu vahvasti UML:n käyttöön mallintamisessa.

Menetelmien tarkastelu jatkuu seuraavasti. Luvussa 4 muodostetaan viitekehys agenttipohjaisten menetelmien arviointiin ja vertailuun. Luvussa 5 edellä kuvattuja menetelmiä arvioidaan ja vertaillaan. Menetelmien tarkastelu päättyy lukuun 6, jossa esitetään vertailun johtopäätökset.

4 MENETELMIEN VERTAILUN VIITEKEHYS

Tässä luvussa muodostetaan viitekehys edellisessä luvussa kuvattujen menetelmien vertailua varten. Ensimmäisessä kohdassa tarkastellaan menetelmien arviointia ja vertailua yleisesti. Toisessa kohdassa kuvataan lyhyesti yleisiä menetelmien vertailun viitekehyksiä ja tarkastellaan lähemmin kolmea viitekehystä. Kolmannessa kohdassa kuvataan agenttipohjaisten menetelmien vertailuissa käytettyjä viitekehyksiä. Neljännessä kohdassa esitetään yhteenveto esitellyistä viitekehyksistä ja valitaan sopiva viitekehys, josta kohdassa 4.5 muokataan viitekehys seuraavassa luvussa tehtävää menetelmien vertailua varten. Luku päättyy yhteenvetoon.

4.1 Menetelmien vertailusta yleisesti

Menetelmiä vertaillaan sekä akateemisista että käytännön syistä. Akateemisista syistä menetelmiä vertaillaan, jotta ymmärrettäisiin paremmin menetelmien luonnetta (piirteitä, tavoitteita, filosofioita jne.). Tämän ymmärtämyksen perusteella menetelmiä voidaan luokitella ja parantaa tulevaisuuden tietojärjestelmien kehitystä. Käytännön syistä menetelmiä vertaillaan, jotta organisaatiossa voitaisiin valita jokin menetelmä, tai mahdollisesti useita menetelmiä, järjestelmän kehittämiseen. Nämä syyt ovat osittain päällekkäisiä, ja tavoitteena on, että akateemiset tutkimukset tukisivat myös menetelmien valintaa käytännön tilanteissa ja vastavuoroisesti käytännön syyt vaikuttaisivat akateemisten tutkimusten kriteereihin. (Avison & Fitzgerald 1995, 434)

Menetelmiä voidaan arvioida ja vertailla useilla eri tavoilla. Yksi tavoista on piirreanalyysi (feature analysis), jota sovelletaan myös tässä tutkielmassa. Sol (1983) esittää artikkelissaan tapoja piirreanalyysin tekemiseen. Eräs tapa suorittaa piirreanalyysi on luoda ideaalinen menetelmä, jota vasten muita menetelmiä verrataan. Ongelmana tässä tavassa on, kuinka ideaali menetelmä määritellään. Piirreanalyysi voidaan suorittaa myös erottamalla joukko tärkeiksi katsottuja piirteitä tai ominaisuuksia induktiivisella tavalla useista menetelmistä ja muodostamalla niistä viitekehys vertailua varten. Kolmas tapa suorittaa piirreanalyysi on muodostaa etukäteen hypoteesi piirteiden joukosta ja johtaa mahdollinen viitekehys useista menetelmistä saadun empiirisen aineiston perusteella. Tämän lähestymistavan ongelmana on hypoteesin muodostaminen. Neljäntenä vaihtoehtona Sol (1983)

mainitsee metakielen määrittelyn. Metakieli toimisi kommunikoinnin välineenä ja viitekehyksenä, jolla eri menetelmiä voidaan kuvata. Lähestymistavan etuna on se, että menetelmän implisiittiset ja kontekstisidonnaiset piirteet ja prosessit voidaan ilmaista eksplisiittisesti. Lähestymistavan ongelmaksi voi muodostua metakielen rajallinen ilmaisuvoima. Se voi myös estää vertailijaa näkemästä joidenkin menetelmien erityispiirteitä. Viimeisenä vaihtoehtona piirreanalyysin tekemiseen Sol (1983) mainitsee kontingenssilähestymistavan. Lähestymistavassa yritetään liittää menetelmien piirteet kontingensseihin soveltamalla menetelmää spesifisiin ongelmatilanteisiin. (Sol 1983)

Piirreanalyysin ongelmaksi saattaa muodostua sen subjektiivisuus. Avison ja Fitzgerald (1995) mainitsevat myös muita tapoja vertailun suorittamiseen. Vaihtoehtoja piirreanalyysille ovat esimerkiksi käsitteiden ja kielten teoreettiset tutkimukset ja menetelmien soveltaminen reaalimaailman case-tapauksiin. Ensimmäisen lähestymistavan väitetään vähentävän piirreanalyysin subjektiivisuutta. Lähestymistavassa suoritetaan tutkimuksia hyvin määritellyillä ja rajatuilla kohdealueilla tarkoin määriteltyjen käsitteiden avulla. Toisen lähestymistavan mukaiset vertailut ovat vaikeita suorittaa ja kuluttavat runsaasti resursseja, mutta voivat osoittautua hyödyllisiksi. Tutkimusta tehtäessä on vaikeaa arvioida, mikä on ympäristötekijöiden, kuten esimerkiksi tutkijan pätevyyden, vaikutus tällaiseen tutkimukseen. Lisäksi case-tutkimusten ongelmana on se, että niistä on vaikea tehdä keskenään vertailtavia. (Avison & Fitzgerald 1995, 438-439)

Menetelmien vertailu on aina haastava tehtävä. Vertailun tuloksia on tarkasteltava kriittisesti, koska vertailuja tekevät ihmiset ja näin ollen ne ovat aina enemmän tai vähemmän subjektiivisia. Erityisesti kaupallisten menetelmien vertailuihin on suhtauduttava varauksella, etenkin silloin, jos vertailun suorittaja on jonkin vertailussa mukana olevan menetelmän kehittäjä. (Avison & Fitzgerald 1995, 443-445)

Menetelmien vertailuja ja arviointeja on kirjallisuudessa tehty satoja. Tässä yhteydessä mainitaan joitakin esimerkkejä aiemmin tehdyistä vertailuista ja viitekehyksistä. Mukaan on pyritty ottamaan sekä vanhempia että uudempia viitekehyksiä. Menetelmien vertailukehyksiä voidaan luokitella eri tavoin. Yleisiä kehyksiä ovat esittäneet muun muassa Floyd (1986), Jayaratna (1994), Avison ja Fiztgerald (1995), Tolvanen (1998) ja

Moody (2003). Kehyksiä voidaan luokitella myös tiettyjen näkökulmien tai teorioiden mukaan. Janson ja Woo (1995) muodostavat artikkelissaan viitekehyksen tietojärjestelmien kehittämismenetelmien ja -työkalujen vertailuun. He käyttävät viitekehyksensä pohjana puheaktiteoriaa. Coughlan ja McRedie (2002) puolestaan vertailevat menetelmiä järjestelmälle asetettavien vaatimusten näkökulmasta. Päivärinta (2002) vertailee tietojärjestelmien kehittämisen genrepohjaista lähestymistapaa 11 muuhun lähestymistapaan. Menetelmien vertailun viitekehyksiä on myös tietyille menetelmätyypeille. Oliomenetelmien vertailun viitekehyksiä esittävät muun muassa Iivari (1994) ja Snoeck ym. (1996). Henderson-Sellers ym. (2001) puolestaan esittävät viitekehyksen ja suorittavat kvalitatiivisen vertailun kahdelle merkittävälle oliopohjaiselle prosessille: RUP (Kruchten 1999) ja OPEN (Graham ym. 1997). Komponenttipohjaisia menetelmiä arvioivat artikkelissaan Boertien ym. (2001). Agenttipohjaisten menetelmien viitekehyksiä ovat esittäneet Tran ym. (2003), Dam ja Winikoff (2004) sekä Sturm ja Shehory (2004).

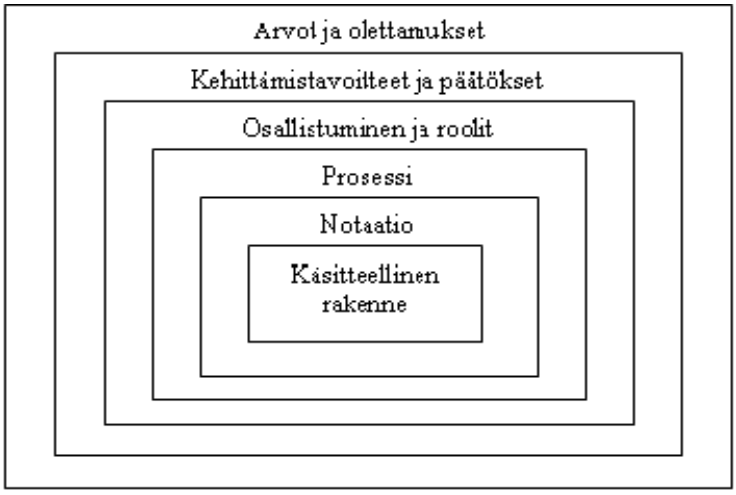
Seuraavassa kohdassa kuvataan tarkemmalla tasolla kolme yleistä menetelmien vertailun viitekehystä. Kuvattavat kehykset ovat Tolvasen (1998) menetelmätietämyksen malli, Avisonin ja Fitzgeraldin (1995) viitekehys ja Iivarin (1994) viitekehys. Tolvasen kehys esitellään, koska se on yksinkertaisin ja selväpiirteisin. Siitä saa myös hyvän yleiskuvan menetelmien sisällöllisestä rakenteesta. Avisonin ja Fitzgeraldin kehys puolestaan on hyvä esimerkki kattavasta menetelmien vertailun viitekehyksestä. Tässä tutkielmassa esitellyt agenttipohjaiset menetelmät perustuvat olioparadigmaan ja tästä syystä on hyvä tarkastella myös oliopohjaisten menetelmien vertailukehyksiä. Iivarin (1994) viitekehys on hyvä esimerkki oliopohjaisten menetelmien vertailusta. Yleisten kehysten esittelyn jälkeen tarkastellaan kolmea agenttipohjaisten menetelmien vertailuun tarkoitettua viitekehystä. Esiteltävät kehykset ovat Tranin ym. (2003), Damin ja Winikoffin (2004) sekä Sturmin ja Shehoryn (2004) viitekehykset. Kyseiset kehykset on valittu tarkasteluun siksi, että agenttipohjaisten menetelmien vertailua on tehty hyvin vähän ja ne ovat käytännössä ainoat kehykset, jotka tällä hetkellä ovat saatavilla. Viitekehysten esittelyn jälkeen niitä vertaillaan lyhyesti toisiinsa.

4.2 Yleisiä menetelmien vertailun viitekehyksiä

Kirjallisuudessa on esitetty lukemattomia viitekehyksiä menetelmien vertailuun. Tässä kohdassa kuvataan kolmea menetelmien vertailun viitekehystä. Aluksi esitellään Tolvasen (1998) menetelmätietämyksen malli, jonka jälkeen tarkastellaan Avisonin ja Fitzgeraldin (1995) viitekehystä. Lopuksi kuvataan Iivarin (1994) viitekehys.

4.2.1 Tolvasen menetelmätietämyksen malli

Kirjallisuudessa on esitetty monia lähestymistapoja menetelmien eri puolten analysointiin ja luokitteluun. Tolvasen (1998) mukaan luokitteluja on lähes yhtä paljon kuin menetelmiäkin. Tolvanen (1998) yhdistelee kirjallisuudessa esitettyjä näkemyksiä ja jäsentää väitöskirjassaan menetelmätietämyksen simpukkamallin (shell model) mukaisesti (KUVIO 20). Mallia voidaan käyttää myös menetelmien arvioinnin pohjana.



KUVIO 20. Menetelmätietämyksen tyypit. (Tolvanen 1998, 35)

Menetelmän *käsitteellinen rakenne* muodostaa perustan muille menetelmätietämyksen tyypeille. Kaikki tietojärjestelmien kehittämismenetelmät perustuvat joihinkin käsitteisiin. Luonnollisesti eri menetelmien käsitteellinen perusta saattaa vaihdella riippuen esimerkiksi kohdealueesta ja menetelmän formaalisuuden asteesta. Menetelmän käsitteet puolestaan voidaan esittää ainoastaan käyttämällä jotakin *notaatiota*. Ideaalitilanteessa jokaisella menetelmän käsitteellä on vain yksi esitystapa notaatiossa. (Tolvanen 1998)

Seuraava menetelmätietämyksen taso koskee menetelmän *prosesseja*. Menetelmät sisältävät kahdenlaisia prosesseja: mallinnukseen liittyviä prosesseja ja johtamiseen liittyviä prosesseja. Edelliset kuvaavat, kuinka menetelmän mukaan tuotetaan tuloksia kun taas jälkimmäiset ohjeistavat projektin suunnittelua, organisointia ja johtamista. Kaikista menetelmistä ei välttämättä löydy prosessia eksplisiittisesti ilmaistuna. Mallin neljäs taso on *osallistuminen ja roolit*. Tietojärjestelmien kehittäminen on aina useamman henkilön vuorovaikutukseen perustuvaa toimintaa, johon ihmiset osallistuvat eri rooleissa (esim. johtajat, ohjelmoijat, suunnittelijat ja käyttäjät). Useimmat menetelmät olettavat, että niitä käyttävät vain tietojärjestelmäammattilaiset ja heistäkin lähinnä analysoijat ja suunnittelijat. (Tolvanen 1998)

Menetelmillä ei ole tarkoitus ainoastaan kuvailla kehitettävää järjestelmää, vaan niiden tarkoituksena on myös parantaa nykyistä tilannetta viemällä läpi muutosprosessi. Tähän liittyvät menetelmätietämyksen *kehittämistavoitteet ja päätökset*. Tavoitteet ja päätökset liittyvät menetelmän implisiittisiin tai eksplisiittisiin perusteluihin siitä, kuinka ”hyvä” tietojärjestelmä tulisi kehittää. Kehittämistavoitteet ovat yleisiä ja liittyvät ratkaisutyyppeihin, joita pidetään tavoiteltavina. Päätökset taas ovat enemmän eksplisiittisiä ja liittyvät läheisesti menetelmän käyttöön. Menetelmät perustuvat aina joihinkin filosofisiin *arvoihin ja olettamuksiin*. Näitä voidaan kutsua myös ”näkymättömiksi” tai ”piilotetuiksi” olettamuksiksi eli paradigmoiksi. Tosiasiassa monet menetelmät eivät eksplisiittisesti määrittele tai tunnista perustana olevia olettamuksia. (Tolvanen 1998)

4.2.2 Avisonin ja Fitzgeraldin viitekehys

Avison ja Fitzgerald (1995) esittävät menetelmien vertailuun viitekehyksen, joka pyrkii akateemisen lähestymistavan lisäksi tarjoamaan käytännön tukea menetelmien vertailuun. Kehyksen ei ole tarkoitus olla kaiken kattava, vaan sitä voidaan muokata tilanteen mukaan esimerkiksi lisäämällä vertailtavia piirteitä (esimerkiksi menetelmän tuottaman dokumentaation ja spesifikaatioiden määrä). Viitekehys sisältää seitsemän peruselementtiä: filosofia, malli, tekniikat ja työkalut, laajuus, tuotokset, käytäntö ja tuote. Menetelmän *filosofiaan* on olennaista kiinnittää huomiota, koska se vaikuttaa menetelmän kaikkiin muihin elementteihin. Filosofia voi olla ilmaistu eksplisiittisesti, mutta useimmissa menetelmissä filosofia ilmenee implisiittisesti. Filosofia jakautuu

viitekehyksessä neljään osaelementtiin, jotka ovat paradigma, tavoitteet, kohdealue (domain) ja kohde (target), jolla viitataan menetelmän sovellettavuuteen. Viitekehyksen toinen elementti koskee *mallia*, jota menetelmä noudattaa. Tietojärjestelmien alueella käytettävien menetelmien mallit ovat lähes aina ikoni-, kuvio- ja kaaviopohjaisia. Malleilla pyritään esittämään kohteena oleva ongelma sopivalla abstraktiotasolla. (Avison & Fitzgerald 1995, 445-455)

Viitekehyksen kolmantena ja tärkeimpänä elementtinä on *tekniikat ja työkalut*, joita menetelmä käyttää. Neljäs vertailuelementti on menetelmän *laajuus*. Laajuudella tarkoitetaan tässä yhteydessä sitä, mitä järjestelmän kehityksen elinkaaren vaiheita menetelmä kattaa. Elinkaarien vaiheiden käyttäminen laajuuden mittarina on ongelmallista, koska ei ole olemassa sopimusta siitä, mitkä vaiheet kuuluvat järjestelmän kehittämisen elinkaareen. Viitekehyksen viidentenä elementtinä on *tuotokset*, joita menetelmää käyttämällä saadaan aikaiseksi. On tärkeää tietää, mitä tuotoksia menetelmää käyttämällä saadaan missäkin vaiheessa. Tärkeintä on selvittää, millainen on lopputuotos. Lopputuotos vaihtelee menetelmien välillä ja voi jossain menetelmässä olla esimerkiksi analyysispesifikaatio ja jossain toisessa menetelmässä toimiva järjestelmä. (Avison & Fitzgerald 1995, 455-456)

Viitekehyksen kuudentena elementtinä on *käytäntö*. Huomioitavia seikkoja ovat seuraavat: menetelmän tausta (akateeminen vai kaupallinen), käyttäjäkunta, menetelmän osanottajat (voivatko käyttäjät ryhtyä käyttämään menetelmää itse vai tarvitaanko mukaan ammattilaisia), sekä mitä taitotasoja tarvitaan. Käytäntöön tulisi sisältyä myös aiemmin kohdattujen ongelmien arviointi ja aiemmat havainnot onnistumisista ja epäonnistumisista. Nämä voidaan selvittää keräämällä kokemuksia menetelmän aiemmilta käyttäjiltä. Viitekehyksen viimeisenä elementtinä on menetelmän *tuote*. Tuotteella tarkoitetaan sitä, mitä asiakas saa rahojensa vastineeksi hankkiessaan menetelmän. Tämä voi sisältää esimerkiksi ohjelmiston, kirjallisen dokumentaation, koulutuksen, konsultaation jne. (Avison & Fitzgerald 1995, 456-457)

4.2.3 Iivarin viitekehys

Iivari (1994) esittää artikkelissaan viitekehyksen oliopohjaisten analyysimenetelmien arvioimiseksi. Viitekehys muodostuu kahdesta ulottuvuudesta. Ensimmäinen ulottuvuus

erottaa tietojärjestelmän mallintamisen kolme päätasoa: organisatorinen taso, käsitteellinen/infolooginen taso ja tekninen/datalooginen taso. Organisatorinen taso kuvaa järjestelmän organisatorisen kontekstin. Käsitteellinen/infolooginen taso sisältää mallin järjestelmän perustana olevasta kohdemaailmasta (universe of discourse), järjestelmän yleisen spesifikaation sekä käyttöliittymän kuvauksen. Tekninen/datalooginen taso määrittelee toteutusriippuvan ratkaisun järjestelmälle. Kehyksen toinen ulottuvuus erottaa toisistaan järjestelmän rakenteen (data), toiminnan (prosessi) ja käyttäytymisen (kontrolli). (Iivari 1994)

Taulukossa 4 esitetään Iivarin (1994) tietojärjestelmän viitemalli, joka muodostuu edellä mainituista ulottuvuuksista ja toimii näin ollen viitekehyksen pohjana. Taulukkoon on sijoitettu myös esimerkkejä käsitteistä, jotka kuuluvat tietylle tasolle ja tiettyyn abstraktioon. (Iivari 1994)

TAULUKKO 4. Tietojärjestelmän viitemalli (Iivari 1994, 88)

		Rakenneabstraktio	**Toiminta-abstraktio**	**Käyttäytymis-abstraktio**
Organisatorinen taso	Suunniteltu konteksti	Organisatoriset toimijat ja positiot Organisaatiokanavat	Organisatoriset toiminnot	Tapahtumat ja tilanteet
	Sovelluskonsepti	Käyttäjät	Tietojärjestelmä-palvelut	Tietojärjestelmän käyttö
Käsitteellinen /Infolooginen taso	Kohdemaailma	Oliotyypit Assosiaatiotyypit Attribuutit	Toiminto/ operaatiotyypit	Tapahtumat Tapahtumia käynnistävät ehdot
	Tietojärjestelmä-spesifikaatiot	Informaatiokanta	Tietojärjestelmä-toiminnot	Tapahtumat Tapahtumat käynnistävät ehdot
	Käyttöliittymä	Käyttöliittymän oliotyypit	Käyttöliittymän operaatiotyypit	Käyttöliittymä-toiminnot
Tekninen/Datalooginen taso		Abstraktit koneet ja datavälitys	Tietokannat/ tiedostot, Ohjelmisto-komponentit, I/O – tietotyypit ja algoritmit	Tapahtumat, Kontrollitilat, Kontrollisäännöt, Kontrollitilasiirty-mät ja data-prosessit

Iivari (1994) tekee viitekehyksen pohjaksi kymmenen olettamusta oliopohjaisuudesta ja tietojärjestelmistä. Tässä yhteydessä ei ole mahdollista käsitellä kaikkia olettamuksia,

mutta tarkastellaan lähemmin kolmea olettamusta, jotka havainnollistavat edellä esitettyä viitemallia. Iivari (1994) olettaa, että tietojärjestelmä on tietokonepohjainen järjestelmä, joka tarjoaa joukolle käyttäjiä informaatiota spesifisistä aiheista tietyssä organisatorisessa ympäristössä. Tämän määritelmän mukaan voidaan erottaa kolme tietojärjestelmien mallintamisen pääkomponenttia: isäntäorganisaatio (organisatorinen konteksti ja käyttäjät), kohdemaailma (aiheet) sekä teknologia (tietokoneet). Toiseksi oletetaan, että tietojärjestelmä kohdemaailmana sisältää kolme tasoa. Kohdemaailman malli määrittelee reaalimaailmasta johdetut abstraktiot. Informaatiota kuvaava malli määrittelee informaatioabstraktiot (dokumentit, raportit, kyselyt jne., ja niiden semanttisen sisällön). Käyttöliittymämalli puolestaan määrittelee näkymät, joiden kautta käyttäjät voivat nähdä informaatioabstraktiot. Kolmanneksi oletetaan, että tietojärjestelmä on luontaisesti monimutkainen järjestelmä, joka sisältää kolme näkökulmaa: rakenne, toiminta ja käyttäytyminen. (Iivari 1994)

Viitekehyksensä pohjalta Iivari (1994) vertailee keskenään kuutta oliopohjaista analyysimenetelmää. Hän analysoi menetelmien keskeisiä käsitteitä ja niiden merkityksiä sijoittamalla ne esiteltyyn viitekehykseen. Lisäksi Iivari tutkii, millaisia malleja menetelmiin sisältyy ja mikä on mallien käsitteellinen sisältö. Hän havainnollistaa mallien sisältöä täydentämällä kehystä malleja kuvaavilla suorakaiteilla, jotka sulkevat sisäänsä mallin käsitteet. Edelleen Iivari pyrkii arvioimaan menetelmien mukaisia prosesseja analysoimalla, millä askeleilla ja missä järjestyksessä mitäkin kaksiulotteisen viitekehyksen osa-aluetta tarkastellaan.

4.3 Agenttipohjaisten menetelmien vertailun viitekehyksiä

Kohdassa 4.2 esitellyt viitekehykset ovat esitetty melko yleisellä tasolla ja ne eivät sellaisenaan sovellu agenttipohjaisten menetelmien vertailuun, vaikkakin ne antavat hyvän pohjan vertailulle. Agenttipohjaisten menetelmien vertailua on tehty kirjallisuudessa vasta varsin vähän. Tämä on luonnollista ottaen huomioon sen, että suurin osa agenttipohjaisista menetelmistä on kehitetty muutaman viime vuoden aikana. Agenttipohjaisista menetelmistä on tehty muutama survey-tutkimus (Iglesias ym. 1999; Tveit 2001; Wooldridge ja Ciancarini 2001), joiden lisäksi on olemassa joitakin viitekehyksiä agenttipohjaisten menetelmien arviointia ja vertailua varten. Viitekehyksiä agenttipohjaisten menetelmien arviointiin ja vertailuun ovat kehittäneet Tran ym.

(2003), Dam ja Winikoff (2004) sekä Sturm ja Shehory (2004). Seuraavassa kuvataan kehyksiä edellä mainitussa järjestyksessä.

4.3.1 Tranin ym. viitekehys

Tran ym. (2003) muodostavat artikkelissaan piirreanalyysikehyksen agenttipohjaisten menetelmien arviointia ja vertailua varten. Itse menetelmien vertailua he eivät kuitenkaan suorita. Viitekehys muodostuu neljästä komponentista:

- *Prosesseihin liittyvät kriteerit* (process related criteria): arvioidaan menetelmän tukea agenttijärjestelmän kehittämisprosessille
- *Tekniikoihin liittyvät kriteerit* (technique related criteria): arvioidaan menetelmän tekniikoita
- *Malleihin liittyvät kriteerit* (model related criteria): tutkitaan menetelmän mallien pystyvyyttä (capabilities)
- *Menetelmän tarjoamaan tukeen liittyvät kriteerit* (supportive feature criteria): arvioidaan menetelmän korkean tason pystyvyyttä

Komponentit sisältävät yhteensä 51 menetelmien arviointikriteeriä. Prosesseihin liittyviä kriteereitä on määritelty 15 kappaletta. Kriteereillä arvioidaan esimerkiksi elinkaaren kattavuutta, menetelmän sovellusaluetta ja menetelmän prosessin ymmärtämistä. Menetelmän tekniikoihin liittyviä kriteereitä on määritelty viisi. Niillä arvioidaan mm. menetelmän tekniikoiden heikkouksia ja vahvuuksia, tekniikoiden ymmärtämisen helppoutta ja sitä, onko tekniikoista annettu esimerkkejä ja heuristiikoita. Malleihin liittyviä kriteereitä on puolestaan määritelty 23. Kriteerit kattavat mm. mallien käsitteet, ilmaisuvoiman, täydellisyyden, monimutkaisuuden ja abstraktiotason. Viitekehyksen neljäntenä komponenttina on kriteerit, joilla arvioidaan erilaisia menetelmien tarjoamaan tukeen liittyviä piirteitä. Näitä kriteereitä on määritelty kahdeksan. Kriteereillä tarkastellaan erimerkiksi sitä, antaako menetelmä tukea perinteisille olioille, liikkuville agenteille ja avoimille järjestelmille.

Edellä mainittujen kriteerien lisäksi Tran ym. (2003) määrittelevät listan askeleita ja käsitteitä, jotka menetelmien tulisi kattaa. He kutsuvat näitä käsitteitä ja askeleita standardikäsitteiksi ja -askeleiksi. Ne on määritelty tutkimalla olemassa olevia

agenttipohjaisten järjestelmien kehittämismenetelmiä. Käsitteet ja askeleet on annettu ainoastaan listana, eivätkä Tran ym. (2003) määrittele niitä tässä yhteydessä tarkemmin.

4.3.2 Damin ja Winikoffin viitekehys

Dam ja Winikoffin (2004) vertailukehys koostuu neljästä pääosasta: käsitteet, mallinnuskieli, prosessi ja käytäntö. *Käsitteet* on määritelty tutkimalla alan kirjallisuutta. Tutkimuksen perusteella on määritelty joukko yleisesti hyväksyttyjä käsitteitä, jotka menetelmien tulisi kattaa. Käsitteet kattavat mm. agenttien määritelmän, niiden ominaisuudet kuten autonomia, sopeutumiskyky, älylliset ominaisuudet (BDI), agenttien välisen viestinnän ja muita käsitteitä kuten roolit, tavoitteet, tapahtumat ja toiminnot. (Dam & Winikoff 2004)

Mallinnuskielen osalta Dam ja Winikoff (2004) jakavat arviointikriteerit kahteen ryhmään, käytettävyyskriteereihin ja teknisiin kriteereihin. Käytettävyyden osalta arvioidaan, mitä keinoja menetelmän mallinnuskieli tarjoaa järjestelmän kehittäjille ajatustensa ja ideoidensa ilmaisemiseen. Lähinnä tarkoituksena on arvioida, kuinka helppoa mallien ja notaation oppiminen ja käyttö on. Lisäksi notaation osalta arvioidaan, voidaanko sen avulla esittää kaikki määritellyt käsitteet ja onko se riittävän ilmaisuvoimainen. Teknisten kriteereiden perusteella arvioidaan mallinnuskielen yksiselitteisyyttä ja johdonmukaisuutta. Yksiselitteisyydellä tarkoitetaan sitä, että mallit ovat tulkittavissa yksiselitteisesti ilman erehtymisen vaaraa. Johdonmukaisuudella puolestaan tarkoitetaan sitä, että mallit ovat keskenään ristiriidattomia. (Dam & Winikoff 2004)

Prosessin osalta tärkeänä arviointikriteerinä on menetelmän prosessin askelten yksityiskohtaisuus. Prosessin askelista arvioidaan, onko askel mainittu, onko askeleen suorittamiseksi annettu ohjeet, annetaanko siitä esimerkkejä ja annetaanko askeleen suorittamiselle heuristiikkoja. Viimeisenä arviointikohteena on *käytäntö*. Käytäntöä arvioidaan sekä johtamisen että teknisestä näkökulmasta. Johtamisen osalta arvioidaan, millaista tukea menetelmä tarjoaa johtamiseen, kun menetelmää otetaan käyttöön. Arviointi kohdistuu menetelmän käyttöönoton kustannuksiin, menetelmän kypsyyteen sekä sen vaikutuksiin organisaation nykyisiin liiketoimintakäytäntöihin. Teknisillä

kriteereillä tarkastellaan, onko menetelmä tarkoitettu jollekin tietylle kohdealueelle. (Dam & Winikoff 2004)

Edellä mainittujen piirteiden lisäksi viitekehyksellä pyritään arvioimaan menetelmien markkinakelpoisuutta ja tukea ohjelmistotuotannolle. Markkinakelpoisuudesta tutkijat tosin toteavat, että koska kaikki vertailtavat menetelmät ovat edelleen kehitteillä, mikään niistä ei täytä markkinakelpoisuuden kriteereitä, joten niitä ei huomioida arvioinnissa lainkaan. Ohjelmistotuotannon tukeen liittyviä kriteereitä arvioidaan viitekehyksen jokaisen neljän pääosan yhteydessä. (Dam & Winikoff 2004)

Dam ja Winikoff (2004) ottavat vertailuun kolme menetelmää, jotka ovat MaSE, Prometheus ja Tropos. Viitekehykseen perustuen tutkijat kehittivät 60 kysymystä sisältävän kysymyslomakkeen, joka lähetettiin vertailussa mukana olevien menetelmien kehittäjille. Tällä pyrittiin lisäämään tutkimuksen objektiivisuutta. Lisäksi tutkijat antoivat ryhmälle opiskelijoita tehtäväksi kehittää sovellus käyttämällä vertailtavia menetelmiä. Lopuksi opiskelijat vastasivat samaan kyselyyn kuin menetelmän kehittäjät ja antoivat palautetta menetelmän käytöstä. (Dam & Winikoff 2004)

4.3.3 Sturmin ja Shehoryn viitekehys

Sturm ja Shehory (2004) esittävät artikkelissaan hieman samankaltaisen kehyksen menetelmien vertailuun kuin Dam ja Winikoff (2004). Viitekehys on jatkotyötä Shehoryn ja Sturmin (2001) aikaisemmin määrittelemälle viitekehykselle. Kehys on jäsentelyn ja arviointikriteerien osalta selkeämpi kuin Damin ja Winikoffin (2004) ja Tranin ym. (2003) kehykset.

Sturmin ja Shehoryn (2004) viitekehys koostuu neljästä elementistä: käsitteet ja ominaisuudet, notaatio ja mallinnustekniikat, prosessi sekä käytäntö. Ensimmäisenä elementtinä viitekehyksessä on *käsitteet ja ominaisuudet*. Käsitteet jaetaan yleisiin käsitteisiin ja erityisiin agenttipohjaisten järjestelmien ”perusrakennuspalikoihin” (basic building blocks), jotka menetelmien tulisi kattaa. Yleiset käsitteet (tai ominaisuudet) ovat autonomia, reaktiivisuus, proaktiivisuus ja sosiaalisuus. Perusrakennuspalikat ovat seuraavat: agentti, uskomus, halu, aikomus, viesti, sääntö, organisaatio, protokolla,

rooli, palvelu, yhteisö ja tehtävä. Ominaisuuksilla (properties) tarkoitetaan menetelmän jotakin erityistä kykyä tai ominaispiirrettä. (Sturm & Shehory 2004)

Toisena elementtinä vertailukehyksessä on *notaatio ja mallinnustekniikat*. Notaatiolla tarkoitetaan symbolijärjestelmää, jolla kuvataan suunnittelun kohteena olevaa järjestelmää. Mallinnustekniikoilla puolestaan tarkoitetaan joukkoa malleja, joilla järjestelmää kuvataan tarkoituksenmukaisilla abstraktiotasoilla ja eri näkökulmista. Arviointikriteerejä ovat seuraavat: helppokäyttöisyys, analysoitavuus, monimutkaisuuden hallinta, suoritettavuus, ilmaisuvoima, modulaarisuus ja täsmällisyys. (Sturm & Shehory 2004)

Viitekehyksen kolmantena elementtinä on menetelmän *prosessi*. Prosessilla tarkoitetaan niitä vaiheita ja askeleita, joita menetelmä tarjoaa järjestelmän kehittämiseen. Prosessi kuvaa, missä järjestyksessä ja millä tavalla haluttu järjestelmä saadaan rakennettua. Prosessia arvioidaan kehittämiskontekstin ja järjestelmän elinkaaren vaiheiden kattavuuden perusteella. Lisäksi prosessista pyritään selvittämään, antaako se tukea järjestelmän verifiointiin ja validointiin, sekä ovatko laatunäkökohdat ja projektinhallinta huomioitu. (Sturm & Shehory 2004)

Viitekehyksen viimeisenä elementtinä on *käytäntö* (pragmatics). Tässä vaiheessa tarkastellaan menetelmän käyttöönoton ja käytön kannalta tärkeitä seikkoja. Elementin tarkoituksena on antaa käytännön tukea menetelmän valinnalle jossakin projektissa tai organisaatiossa. Käytännön kannalta tärkeitä arviointikriteereitä ovat seuraavat: resurssit; menetelmän käyttöön tarvittava asiantuntemus; kieli-, paradigma- ja arkkitehtuuriyhteensopivuus; kohdealuesoveltuvuus; skaalautuvuus. (Sturm & Shehory 2004)

Edellä esitellyn viitekehyksen tueksi Sturm ja Shehory (2004) määrittelevät arviointiasteikon, jonka perusteella menetelmiä on tarkoitus arvioida. Asteikko on numeropohjainen ja arviointi suoritetaan välillä 1-7. Numero 1 tarkoittaa, että menetelmä ei ota kyseistä arviointikriteeriä huomioon millään lailla. Numero 7 taas tarkoittaa, että menetelmä huomioi kriteerin täysin. Lopuksi kirjoittajat antavat esimerkin viitekehyksen käytöstä ja arvioivat Gaia-menetelmän vanhempaa versiota (Wooldridge ym. 2000).

4.4 Yhteenveto esitellyistä viitekehyksistä

Taulukossa 5 esitetään tiivistetty yhteenveto edellä kuvatuista kehyksistä. Taulukon sarakkeet koostuvat esitellyistä viitekehyksistä. Taulukon rivit puolestaan ilmaisevat, miten hyvin viitekehys kattaa taulukon vasemmanpuolimmaisimmassa sarakkeessa esitetyt yleiset elementit. Elementit noudattelevat pääasiassa Tolvasen (1998) kehyksen rakennetta, mutta mukaan on otettu myös osia Avisonin ja Fitzgeraldin (1995) kehyksestä. Näin on varmistettu se, että kaikkien kehysten kaikki elementit tulevat huomioitua taulukossa. Taulukossa on nimetty jokaisen viitekehyksen osalta elementti, joka vastaa taulukon vasemmanpuolimmaisimmassa sarakkeessa esitettyjä elementtejä. Jos vastaavaa elementtiä ei esiinny kehyksessä, ilmoitetaan, sisältyykö elementti johonkin toiseen elementtiin. Jos kehys ei huomioi lainkaan kyseistä elementtiä, osoitetaan se viivalla.

Taulukosta 5 voidaan tehdä seuraavia mielenkiintoisia havaintoja. Esimerkiksi kaikki muut esitellyt kehykset, paitsi Avisonin ja Fitzgeraldin (1995) kehys, ottavat huomioon menetelmän käsitteet. Tämä on selkeä puute Avisonin ja Fitzgeraldin kehyksessä. Tämä saattaa johtua siitä, että kirjoittajat pyrkivät teoksessaan myös käytännönläheisyyteen. Menetelmän mallit ja notaatio sekä prosessi otetaan huomioon kaikissa esitellyissä kehyksissä. Osallistumisen ja roolit kattavat Tolvasen (1998), Avisonin ja Fitzgeraldin (1995) sekä Sturmin ja Shehoryn (2004) kehykset. Sen sijaan Tolvasen (1998) esittämät kehittämistavoitteet ja päätökset jäävät kaikkien muiden kehysten ulkopuolelle. Menetelmän filosofiaa käsitellään ainoastaan Tolvasen (1998) sekä Avisonin ja Fitzgeraldin (1995) kehyksissä. Nämä ovat luonteeltaan yleispäteviä, kun taas muut kehykset perustuvat tiettyyn paradigmaan. Iivarin (1994) kehys pohjautuu olioparadigmaan ja loput kehykset agenttiparadigmaan. Näin ollen näissä kehyksissä on filosofia huomioitu implisiittisesti. Tekniikoita ja työkaluja käsitellään jollakin tasolla Avisonin ja Fitzgeraldin (1995) sekä Tranin ym. (2003) kehyksissä. Käytännön seikkoja huomioidaan Avisonin ja Fitzgeraldin (1995), Damin ja Winikoffin (2004) sekä Sturmin ja Shehoryn (2004) kehyksissä. Menetelmää tuotteena arvioivat ainoastaan Avison ja Fitzgerald (1995). Tämä johtunee siitä, että kirjoittajat haluavat korostaa menetelmän tuotteen merkitystä, jos sen halutaan menestyvän kaupallisesti.

TAULUKKO 5. Yhteenveto menetelmien vertailun viitekehyksistä

Viitekehys/ Elementti	**Tolvanen (1998)**	**Avison & Fitzgerald (1995)**	**Iivari (1994)**	**Tran ym. (2003)**	**Dam & Winikoff (2004)**	**Sturm & Shehory (2004)**
Käsitteet	Käsitteellinen rakenne	-	Organisatorinen taso, Käsitteellinen/Info-looginen taso, Tekninen/datalooginen taso	Käsitteitä arvioidaan malleihin liittyvien kriteereiden yhteydessä.	Käsitteet	Käsitteet ja ominaisuudet
Mallit ja notaatio	Notaatio	Malli	Käsitteellinen/Info-looginen taso, Tekninen/dataloogi-nen taso	Malleihin liittyvät kriteerit	Mallinnuskieli	Notaatio ja mallinnustekniikat
Prosessi	Prosessi	Laajuus	Prosessi	Prosessiin liittyvät kriteerit	Prosessi	Prosessi
Osallistuminen ja roolit	Osallistuminen ja roolit	Tämä osuus sisältyy kehyksen käytäntö-elementtiin	-	-	-	Sisältyy kehyksen käytäntö-elementtiin
Kehittämistavoitteet ja päätökset	Kehittämistavoitteet ja päätökset	-	-	-	-	-
Filosofia	Arvot ja olettamukset	Filosofia	-	-	-	-
Tekniikat ja työkalut	-	Tekniikat ja työkalut		Tekniikoihin ja menetelmien tarjoamaan tukeen liittyvät kriteerit	-	-
Käytäntö	-	Käytäntö	-	-	Käytäntö	Käytäntö
Tuote	-	Tuote	-	-	-	-

Tutkielman tavoitteena on arvioida ja vertailla menetelmien käsitteitä, malleja, prosesseja ja jonkin verran myös käytäntöön liittyviä seikkoja. Taulukosta 5 nähdään, että näihin tavoitteisiin esitellyt viitekehykset antavat hyvin tukea. Kolme ensimmäistä elementtiä sisältyy Tolvasen (1998) viitekehykseen. Myös muissa kehyksissä on nämä elementit lukuun ottamatta Avisonin ja Fitzgeraldin (1995) kehystä, jossa ei huomioida menetelmien käsitteitä. Käytännön seikkojen vertailua tukevat Avisonin ja Fitzgeraldin (1995), Damin ja Winikoffin (2004) sekä Sturmin ja Shehoryn (2004) viitekehykset.

Seuraavassa kohdassa muodostetaan viitekehys aiemmin esiteltyjen agenttipohjaisten menetelmien arviointiin ja vertailuun. Viitekehyksen pohjaksi valitaan Sturmin ja Shehoryn (2004) viitekehys. Se on alunperin tehty agenttipohjaisten menetelmien vertailuun, joten se sopii hyvin myös tämän tutkielman tarkoitukseen. Vertailulla yleisiin viitekehyksiin tuli osoitetuksi, että se kattaa olennaiset osat tämän tutkielman tavoitteiksi asetetuista arviointikohteista. Lisäksi Sturmin ja Shehoryn (2004) kehys on esitykseltään selkeämpi kuin Damin ja Winikoffin (2004) ja Tranin ym. (2003) kehykset. Sturm ja Shehory (2004) esittävät kehyksessään jokaisen elementin ja kriteerin lisäksi selityksen kriteereille, kun taas Tran ym. esittävät vertailukriteerinsä hyvin tiivistetysti ja niitä vain vähän perustellen. Damin ja Winikoffin käyttämä kehys taas on ollut kyselytutkimuksen pohjana, kun taas Sturmin ja Shehoryn kehys on tarkoitettu sovellettavaksi sellaisenaan. Vertailtavien kriteereiden osalta kehykset ovat suurin piirtein samanlaiset. Joitakin pieniä eroja esiintyy, mutta tärkeimmät kriteerit tulevat kyllä huomioiduksi kaikissa kehyksissä.

4.5 Viitekehys agenttipohjaisten menetelmien arviointiin ja vertailuun

Tässä kohdassa muodostetaan viitekehys agenttipohjaisten menetelmien arviointiin ja vertailuun. Vertailumenetelmänä käytetään piirreanalyysiä (ks. Sol 1983), joka on hyvin toteutettavissa tutkielman laajuuden puitteissa. Kohdassa 4.1 todettiin, että menetelmiä vertaillaan sekä akateemisista että käytännön syistä. Tutkielman viitekehys painottuu menetelmien käsitteellisteoreettiseen vertailuun ja on näin ollen lähtökohdiltaan akateeminen. Mukaan pyritään ottamaan myös jonkin verran käytännön elementtejä, jotka tukisivat menetelmän valintaa käytännön tilanteessa.

Seuraavassa luvussa suoritettavan vertailun tavoitteena on antaa yhtenäinen kuva luvussa 3 esitellyistä menetelmistä. Tavoitteena on selvittää, mitä yhtäläisyyksiä ja eroja menetelmien käsitteissä, malleissa ja prosesseissa on. Edellisessä kohdassa valittiin viitekehyksen pohjaksi Sturmin ja Shehoryn (2004) kehys. Taulukossa 6 esitetään Sturmin ja Shehoryn (2004) viitekehys tiivistetyssä muodossa. Viitekehyksen elementit on nimetty tässä tutkielmassa käytettävien käsitteiden mukaisesti. Taulukon tummennetut rivit ovat kriteereitä, joita sovelletaan myös tämän tutkielman viitekehyksessä. Lihavoidulla tekstillä ilmaistut rivit ovat kokonaan uusia elementtejä, joita ei esiinny Sturmin ja Shehoryn (2004) kehyksessä. Uudet elementit ovat käsiterakenteet ja mallien käsitteellinen sisältö. Käsiterakenteiden esittämisellä pyritään luomaan yhtenäinen kuva menetelmien käsitteistä ja niiden suhteista toisiinsa. Lisäksi päästään vertailemaan, miten eri menetelmien käsiterakenteet eroavat toisistaan. Mallien käsitteellisen sisällön esittäminen taas tuo syvyyttä menetelmien mallien vertailuun.

TAULUKKO 6. Agenttipohjaisten menetelmien vertailun ja arvioinnin viitekehys.

Elementit	**Kriteerit**
Käsitteet	• Peruskäsitteet • Ominaisuudet • **Käsiterakenteet**
Mallit	• Helppokäyttöisyys • Analysoitavuus • **Käsitteellinen sisältö** • Notaatio • Monimutkaisuuden hallinta • Suoritettavuus • Ilmaisuvoima • Modulaarisuus • Täsmällisyys
Prosessi	• Kehittämiskonteksti • Elinkaaren kattavuus • Tuotokset • Johtamisen tuki
Käytäntö	• Resurssit • Tarvittava asiantuntemus • Kieli-, paradigma- ja arkkitehtuuriyhteensopivuus • Kohdealuesoveltuvuus • Skaalautuvuus

Kaikkia Sturmin ja Shehoryn (2004) viitekehyksessä mainittuja kriteereitä ei ole mielekästä ottaa viitekehykseen mukaan, koska kriteeristö on varsin laaja ja näin ollen vertailusta tulisi tutkielman laajuuden puitteissa pinnallinen. Mallien osalta kriteereistä

on jätetty pois helppokäyttöisyys, analysoitavuus, suoritettavuus ja täsmällisyys. Käytännön osalta on luovuttu resurssien, tarvittavan asiantuntemuksen ja skaalautuvuuden arvioinnista. Kaikki edellä mainitut kriteerit ovat sellaisia, joita voidaan arvioida ainoastaan tilannekohtaisesti tai menetelmää sovellettaessa johonkin esimerkkiin. Lisäksi prosessi-elementin kehittämiskonteksti on sisällöltään hyvin samankaltainen käytäntö-elementin kohdealuesoveltuvuuden kanssa, joten molempia kohtia arvioidaan yhdessä käytäntöön liittyvien seikkojen yhteydessä.

Seuraavaksi esitellään tämän tutkielman viitekehys elementtikohtaisesti. Aluksi esitetään käsitteiden vertailukriteerit, jonka jälkeen kuvataan mallien vertailukriteerit. Kolmanneksi esitetään viitekehyksen prosessi-elementti, ja lopuksi kuvataan lyhyesti käytännön asioiden arviointikriteerit.

4.5.1 Käsitteet

Ensimmäisenä vertailukohteena tarkastellaan menetelmien *käsitteitä*. Käsitteet jaetaan Sturmin ja Shehoryn (2004) mukaisesti ”perusrakennuspalikoihin” (tästä eteenpäin käytetään termiä ”peruskäsitteet”) ja agenttien yleisiin ominaisuuksiin. Taulukossa 7 esitellään määritelmineen 12 peruskäsitettä, jotka menetelmien tulisi kattaa. Jo aiemmin tutkielmassa todettiin, että kaikista agentteja koskevista käsitteistä ei ole päästy yhteisymmärrykseen. Taulukossa 7 määritellyt käsitteet eivät ole ehdottomia ja niitä voidaan lisätä kehykseen tarpeen mukaan. Käsitteillä on kuitenkin vahva perusta, sillä niitä käytetään toistuvasti alan lähdekirjallisuudessa. Käsitteiden määritelmät on suurimmaksi osaksi otettu Sturmin ja Shehoryn (2004) käsitelistasta, mutta joidenkin käsitteiden osalta käytetään tutkielmassa aiemmin tehtyjä määritelmiä, jotka sopivat paremmin käytettäväksi tässä yhteydessä. Agentin määritelmä on yhdistelty Wooldridgen ja Jenningsin (1995), Nwanan (1996) sekä Geneserethin ja Ketchpelin (1994) määritelmistä. BDI-mallin käsitteet (uskomus, halu ja aikomus) on omaksuttu Raon ja Georgeffin (1995) artikkelista.

TAULUKKO 7. Agenttijärjestelmien peruskäsitteet.

Peruskäsitteet	Määritelmä
Agentti	Agentilla tarkoitetaan ohjelmiston osaa, joka kykenee itsenäiseen ja älykkääseen toimintaan käyttäjänsä puolesta saavuttaakseen sille suunnitellut tavoitteet. (Wooldridge & Jennings 1995; Nwana 1996; Genesereth & Ketchpel 1994)
Uskomus	Agentin uskomukset ovat sen tietoa toimintaympäristöstään. (Rao & Georgeff 1995)
Halu	Agentin halut liittyvät järjestelmälle asetettuihin tavoitteisiin. (Rao & Georgeff 1995)
Aikomus	Agentin aikomukset kuvaavat, mitä toimintaa agentti on parhaillaan suorittamassa tai on suorittanut viimeksi. (Rao & Georgeff 1995)
Organisaatio	Organisaatio on joukko agentteja, jotka toimivat yhdessä saavuttaakseen yhteisen tavoitteen. Organisaatio koostuu rooleista, jotka ovat ominaisia organisaation agenteille. (Sturm & Shehory 2004)
Yhteisö	Kokoelma agentteja ja organisaatioita, jotka tekevät yhteistyötä edistääkseen yksilöllisiä tavoitteitaan. (Sturm & Shehory 2004)
Viesti	Keino välittää informaatiota agenttien välillä. (Sturm & Shehory 2004)
Sääntö	Ohje tai suositus, joka on luonteenomaista agenttien yhteisölle. Agentin täytyy noudattaa yhteisön sääntöjä, jos se haluaa olla osa sitä. (Sturm & Shehory 2004)
Protokolla	Määrittelee agenttien välisen viestinnän säännöt ja käytännöt. (Sturm & Shehory 2004)
Rooli	Abstrakti esitys agentin tehtävästä, palvelusta tai tunnistamisesta ryhmässä (Sturm & Shehory 2004)
Palvelu	Rajapinta, jonka agentti tarjoaa ulkomaailmalle. Palvelu on joukko tehtäviä, jotka yhdessä tarjoavat jonkin toiminnallisen operaation. Palvelu voi koostua toisista palveluista. (Sturm & Shehory 2004)
Tehtävä	Toiminto, joka voidaan antaa yksittäiselle agentille suoritettavaksi. Tehtävällä voi olla aikarajoitteita. (Sturm & Shehory 2004)

Agenttien ominaisuudet ovat autonomia, reaktiivisuus, proaktiivisuus ja sosiaalisuus. Kyseiset ominaisuudet on yleisesti hyväksytty alan lähdekirjallisuudessa ja siten niitä voidaan myös hyvin soveltaa menetelmien vertailussa. Lisäksi on olemassa joukko ominaisuuksia, jotka eivät ole laajasti hyväksyttyjä (Wooldridge & Jennings 1995). Vertailussa käytettävät ominaisuudet määritelmineen esitetään taulukossa 8. Taulukon määritelmät ovat Wooldridgen ja Jenningsin (1995) mukaisia. Ominaisuuksista tutkitaan, miten ne ilmenevät eri menetelmissä.

TAULUKKO 8. Agenttien ominaisuudet. (Wooldridge & Jennings 1995)

Ominaisuudet	Määritelmä
Autonomia	Agentit toimivat itsenäisesti ilman ihmisen suoraa puuttumista asiaan. Agentit voivat myös kontrolloida toimintojaan ja sisäistä tilaansa.
Reaktiivisuus	Agentit tekevät havaintoja ympäristöstään ja mukauttavat toimintaansa ympäristön muutosten mukaan.
Proaktiivisuus	Sen lisäksi, että agentit vastaavat ympäristön muutoksiin, ne kykenevät myös tavoitesuuntautuneeseen toimintaan tekemällä itse aloitteita.
Sosiaalisuus	Agentit toimivat vuorovaikutuksessa muiden agenttien kanssa jonkinlaisen agenttikommunikointikielen välityksellä.

Käsitteiden lisäksi tarkastellaan myös menetelmien käsiterakenteita. Vertailu suoritetaan muodostamalla peruskäsitteistä (ja muista tärkeistä käsitteistä) ja niiden välisistä yhteyksistä UML-kaaviot, joista ilmenevät käsitteiden yhteydet toisiinsa. Tällä tavoin päästään tutkimaan, millaiset yhteydet samoilla käsitteillä on eri menetelmissä.

4.5.2 Mallit

Toisena elementtinä vertailukehyksessä on *mallit*. Menetelmien mallit ovat pääpiirteissään esitelty luvussa 3. Elementti sisältää seuraavat kriteerit: käsitteellinen sisältö, notaatio, monimutkaisuuden hallinta, ilmaisuvoima sekä modulaarisuus. Seuraavassa kuvataan kriteerit tarkemmin.

Käsitteellinen sisältö. Tämän kohdan tarkoituksena on havainnollisten kaavioiden avulla selvittää, miten keskeiset käsitteet sijoittuvat menetelmien malleihin.

Notaatio. Arvioidaan ja vertaillaan menetelmien käyttämiä notaatioita. Menetelmittäin tutkitaan, mitä notaatio(i)ta menetelmän malleissa käytetään. Lisäksi pyritään arvioimaan notaatioiden ominaisuuksia, kuten yksinkertaisuutta ja sitä, miten moninaisesti eri menetelmissä notaatioita käytetään.

Monimutkaisuuden hallinta tarkoittaa menetelmän kykyä tukea järjestelmän kehittämisen eri vaiheita eri abstraktiotasoilla. Järjestelmän kehittämisen eri vaiheissa on tarpeen esittää asioita eri abstraktiotasoilla. Vaatimusten määrittelyssä ja analyysivaiheessa riittää asioiden esittäminen yleisellä tasolla, kun taas suunnitteluvaiheessa tarvitaan huomattavasti yksityiskohtaisempia esityksiä.

Ilmaisuvoima. Dam ja Winikoff (2004) toteavat olevan tärkeää, että menetelmän mallinnuskieli antaa mahdollisuuden mallintaa järjestelmää toiminnallisesta, rakenteellisesta ja käyttäytymisen näkökulmasta. Tässä yhteydessä arvioidaan, kuinka hyvin menetelmät kuvaavat seuraavia kohteita:

- järjestelmän rakenne
- järjestelmän tietovirrat
- järjestelmän ja agenttien sisäiset rinnakkaiset toiminnot
- järjestelmän resurssirajoitteet (esim. aika, prosessoriteho ja muistin määrä)
- järjestelmän fyysinen arkkitehtuuri
- agenttien liikkuvuus
- järjestelmän vuorovaikutus ulkoisten järjestelmien kanssa
- käyttöliittymämäärittelyt

Edellä esitetty lista on hieman supistettu esitys Sturmin ja Shehoryn (2004) esittämästä kriteerilistasta.

Modulaarisuus (inkrementaalisuus) tarkoittaa menetelmän kykyä määritellä järjestelmä iteratiivisella ja inkrementaalisella tavalla. Siten järjestelmän kehittämisen aikana ilmenevien uusien vaatimusten ei pitäisi vaikuttaa olemassa oleviin spesifikaatioihin. (Sturm & Shehory 2004)

4.5.3 Prosessi

Menetelmän *prosessilla* tarkoitetaan niitä vaiheita ja askeleita, joita menetelmä tarjoaa järjestelmän kehittämiseen. Näin ollen menetelmän prosessi kuvaa, missä järjestyksessä ja millä tavalla haluttu järjestelmä saadaan rakennettua. Menetelmän prosessia arvioidaan seuraavien kriteerien perusteella.

Elinkaaren kattavuus. Tällä kriteerillä pyritään selvittämään, mitä järjestelmän elinkaaren vaiheita menetelmä kattaa. Tässä tutkielmassa kiinnitetään huomiota seuraaviin järjestelmän elinkaaren vaiheisiin: vaatimusmäärittely, analyysi, suunnittelu, toteutus, testaus ja ylläpito. *Vaatimusmäärittelyvaiheessa* kerätään järjestelmälle halutut vaatimukset eri osapuolilta. Vaatimusten keruuseen on olemassa useita tapoja. Vaatimuksia voidaan kerätä esimerkiksi haastattelemalla käyttäjiä tai asiakasta. Tässä

vaiheessa ei vielä määritellä järjestelmän toiminnallisuutta. *Analyysivaiheessa* kuvataan järjestelmän ulkoisesti havaittavat ominaisuudet, kuten esimerkiksi järjestelmän toiminnallisuus ja suorituskyky. Tässä vaiheessa ollaan kiinnostuneita siitä, mitä järjestelmä tekee, ei siitä, miten haluttu toiminnallisuus toteutetaan. *Suunnitteluvaiheessa* määritellään tapa, jolla järjestelmä toteuttaa sille asetetut vaatimukset ja analyysivaiheessa määritellyn toiminnallisuuden. Analyysivaiheen mallit joko tarkennetaan tai transformoidaan suunnittelumalleiksi, joissa kuvataan järjestelmän fyysiset ja loogiset toiminnot. (Sturm & Shehory 2004) *Toteutusvaiheessa* järjestelmä ohjelmoidaan suunnittelumallien mukaisesti. Ohjelmointi suoritetaan joko käsin tai koodi generoidaan automaattisesti. Ohjelmisto voidaan rakentaa myös valmiista komponenteista. *Testausvaiheessa* järjestelmää testataan mahdollisten virheiden löytämiseksi ennen kuin järjestelmä luovutetaan asiakkaalle tai otetaan käyttöön. Lisäksi testauksella varmistetaan, että järjestelmä todella täyttää sille asetetut vaatimukset. Järjestelmän virheettömyys ei sinällään riitä, vaan on myös varmistuttava, että järjestelmä tekee oikeita asioita. *Ylläpitovaiheessa* järjestelmän virheitä korjataan niiden ilmaantuessa. Tässä vaiheessa järjestelmään myös lisätään uusia toiminnallisuuksia, jos uusia vaatimuksia ilmenee. Ylläpito katsotaan huomioiduksi vasta, jos menetelmässä mainitaan erityisiä ohjeita ylläpitoa varten. Sitä, että menetelmän käyttö saattaa yleisellä tasolla parantaa ohjelmiston ylläpidettävyyttä, ei tässä yhteydessä katsota ylläpidoksi elinkaaren vaiheiden kannalta. Näin tekevät myös Avison ja Fitzgerald (1995) omassa viitekehyksessään. Elinkaaren kattavuuden lisäksi pohditaan, tarkoittavatko eri vaiheet samaa asiaa eri menetelmissä. Toisin sanoen selvitetään, vastaavatko eri menetelmien vaiheet toisiaan, ja jos eivät vastaa, niin mitä eroja menetelmien välillä samoissa vaiheissa on.

Tuotokset. Prosessin tuotoksilla tarkoitetaan lähinnä dokumentaatiota, jota prosessin eri vaiheet tuottavat. Menetelmistä tutkitaan, millaisia tuotoksia niiden prosessit tuottavat. Lopuksi tarkastellaan sitä, tarjoaako prosessi tukea tuotosten verifiointiin ja validointiin ja otetaanko prosessissa laatunäkökohdat huomioon.

Johtamisen tuki. Menetelmien prosesseista pyritään selvittämään, tukevatko ne jollakin tavalla johtamista eli onko esimerkiksi projektin suunnitteluun, seurantaan ja johtamiseen annettu ohjeita.

4.5.4 Käytäntö

Viitekehyksen viimeisenä elementtinä on *käytäntö*. Seuraavaksi listataan asiat, joihin tässä yhteydessä kiinnitetään huomiota. Tarkasteltavien asioiden yhteydessä esitetään joukko kysymyksiä, joihin vastausta etsimällä saadaan suuntaviivoja siitä, onko menetelmä sopiva organisaatiolle ja mitä menetelmän käyttöönotto ja käyttö vaatii henkilöiltä, jotka menetelmää tulevat käyttämään. Tarkasteltavat kohdat ja kysymykset on esitetty Sturmin ja Shehoryn (2004) viitekehyksessä.

Kieli-, paradigma- ja arkkitehtuuriyhteensopivuuteen tulee kiinnittää huomiota sen varmistamiseksi, että menetelmä pitäytyy organisaation infrastruktuurissa ja tietämyksessä. Vastausta etsitään seuraaviin kysymyksiin: Perustuuko menetelmä jonkin tietyn arkkitehtuurin käsitteille? Perustuuko menetelmä johonkin tiettyyn ohjelmointikieleen? Menetelmä voi rajoittua esimerkiksi vain BDI-järjestelmien suunnitteluun tai olla suunnattu jollekin tietylle ohjelmointikielelle.

Kohdealuesoveltuvuus. Tämä seikka tulisi tarkistaa sen varmistamiseksi, että menetelmä todella sopii kehitettävän järjestelmän kohdealueelle. Seuraaviin kysymyksiin etsitään vastausta: Onko menetelmä tarkoitettu sovellettavaksi jollekin tietylle kohdealueelle kuten reaaliaikajärjestelmät tai Internet-pohjaiset järjestelmät? Lisäksi tutkitaan, soveltuuko menetelmä uusien järjestelmien kehittämiseen, olemassa olevien ohjelmistojen uudelleensuunnitteluun (reengineering) ja protoiluun.

4.6 Yhteenveto

Tämän luvun tavoitteena on ollut muodostaa viitekehys agenttipohjaisten menetelmien arviointiin ja vertailuun. Luvun aluksi käsiteltiin menetelmien arviointia ja vertailua yleisesti ja esiteltiin erilaisia vertailuviitekehyksiä. Toisessa kohdassa kuvattiin tarkemmalla tasolla Tolvasen (1998) menetelmätietämyksen malli, Avisonin ja Fitzgeraldin (1995) viitekehys sekä Iivarin (1994) viitekehys. Kolmannessa kohdassa kuvattiin agenttipohjaisten menetelmien vertailuun tarkoitettuja viitekehyksiä. Esitellyistä viitekehyksistä tehtiin myös lyhyt vertailu. Sturmin ja Shehoryn (2004) viitekehys valittiin pohjaksi seuraavassa luvussa tehtävää menetelmien vertailua varten.

Viitekehys muodostuu neljästä pääelementistä: käsitteet, mallit, prosessi ja käytäntö. Käsitteiden osalta tutkitaan 12 viitekehyksessä määriteltyä peruskäsitettä sekä yleisesti hyväksyttyjä agenttien ominaisuuksia. Lisäksi tutkitaan menetelmien käsiterakenteita. Menetelmien malleista tutkitaan käsitteiden sijoittumista malleihin, malleissa käytettäviä notaatioita, monimutkaisuuden hallintaa, ilmaisuvoimaa ja modulaarisuutta. Prosesseista tutkitaan elinkaaren kattavuutta, tuotoksia sekä prosessin tarjoamaa tukea johtamiselle. Lopuksi menetelmistä tutkitaan käytännön osalta kieli-, paradigma- ja arkkitehtuuriyhteensopivuutta sekä kohdealuesoveltuvuutta.

Viitekehys kattaa Tolvasen (1998) menetelmätietämyksen mallin kolme sisintä kerrosta. Loput kerrokset jäävät tämän tutkielman ulkopuolelle. Seuraavassa luvussa suoritetaan menetelmien vertailu edellä kuvatun viitekehyksen pohjalta.

5 MENETELMIEN VERTAILU

Tässä luvussa vertaillaan tutkielmassa aiemmin esiteltyjä menetelmiä edellisessä luvussa kuvatun viitekehyksen pohjalta. Aluksi tarkastellaan menetelmien käsitteitä. Menetelmistä selvitetään, kuinka ne kattavat viitekehyksessä esitetyt peruskäsitteet ja agenttien yleiset ominaisuudet. Toisessa kohdassa vertaillaan menetelmien käsiterakenteita. Kolmannessa kohdassa vertaillaan menetelmien malleja ja notaatioita, ja neljännessä kohdassa analysoidaan menetelmien prosesseja. Viidennessä kohdassa arvioidaan lyhyesti menetelmien käytäntöön liittyviä seikkoja. Luku päättyy yhteenvetoon. Johtopäätökset vertailusta esitetään seuraavassa luvussa.

5.1 Käsitteet

Tässä kohdassa vertaillaan, miten viitekehyksessä määritellyt käsitteet ja ominaisuudet ilmenevät eri menetelmissä. Lopuksi vertaillaan menetelmien käsiterakenteita muodostamalla niistä havainnolliset UML-kaaviot.

5.1.1 Peruskäsitteet

Taulukossa 9 esitetään tiivistettynä eri menetelmien vastineet viitekehyksessä esitetyille agenttijärjestelmän peruskäsitteille. Lihavoidulla tekstillä taulukossa ilmaistaan käsitteen nimitys kyseisessä menetelmässä, jonka jälkeen tekstimuotoisesti kuvataan käsitteen merkitys. Jos käsite on tutkielmassa jo määritelty, eikä menetelmän määritelmä eroa siitä mitenkään, mainitaan tämä lyhyesti kyseisen käsitteen yhteydessä. Jos käsitettä ei esiinny menetelmässä lainkaan, osoitetaan se viivalla (-).

TAULUKKO 9. Peruskäsitteiden merkitykset menetelmittäin

Menetelmä/ Käsite	Gaia	Tropos	MaSE	MESSAGE
Agentti	**Agentti/Agenttityyppi** Agentin käsite on tämän tutkielman määritelmän mukainen.	**Aktori/Agentti** Aktorilla tarkoitetaan entiteettiä, jolla on strategiset tavoitteet ja jokin tarkoitus järjestelmässä tai organisatorisessa ympäristössä. Yksityiskohtaisen suunnittelun vaiheessa siirrytään käyttämään agentin käsitettä, joka on tutkielman määritelmän mukainen.	**Agentti/ Agenttityyppi** Agentit ovat olioiden erikoistumia. Ne koordinoivat toimintojaan toistensa kanssa ja toimivat proaktiivisesti saavuttaakseen sekä yksityiset että koko järjestelmän laajuiset tavoitteet. Agentit voivat olla älykkäitä, mutta niiden ei välttämättä tarvitse olla.	**Agentti** Itsenäinen atominen kokonaisuus, joka kykenee suorittamaan jonkin hyödyllisen toiminnon. Määritelmä on hieman väljempi kuin tässä tutkielmassa käytetty agentin käsite. Agentin erikoistumia ovat ohjelmistoagentit ja ihmisagentit.
Uskomus	-	**Uskomus** Uskomuksella tarkoitetaan aktorin/agentin tietämystä maailmasta.	**Uskomus** Uskomuksia ei määritellä eksplisiittisesti, vaan ne toteutuvat analyysivaiheessa määriteltävien tavoitteiden ja tehtävien kautta.	**Tietämys** Agentin tietämys ympäristöstään varastoidaan agentin tietämyskantaan.
Halu	-	**Tavoite/Yleistavoite** Tavoite merkitsee aktorin/agentin strategista mielenkiinnon kohdetta.	**Tavoite** Tavoite merkitsee asioiden tilaa, joka agentin tulee toiminnallaan saada aikaan.	**Tavoite** Kuvaa tilaa, johon agentin tulee pyrkiä.
Aikomus	-	**Tehtävä/Suunnitelma** Tapa tehdä jotakin. Tehtävän/suunnitelman toteuttaminen on keino saavuttaa tavoite tai yleistavoite.	**Tehtävä** Agentin rooliin liittyvä toimenpide, jonka agentti voi suorittaa itsenäisesti saavuttaakseen tavoitteensa.	-
Organisaatio	**Organisaatio/ (moniagentti)järjestelmä** Koostuu joukosta agentteja, joista jokaisella on yksi tai useampi rooli.	**Järjestelmä** Agenttien joukko, joka toimii yhteistyössä.	**Järjestelmä/ Moniagenttijärjestelmä** Koostuu joukosta agentteja, joista jokaisella on vähintään yksi rooli.	**Organisaatio** Ryhmä agentteja, jotka työskentelevät yhdessä saavuttaakseen yhteisen tavoitteen.

(jatkuu)

TAULUKKO 9. Peruskäsitteiden merkitykset menetelmittäin (jatkuu)

Menetelmä/ Käsite	Gaia	Tropos	MaSE	MESSAGE
Yhteisö	**Organisaatio** Ks. määritelmä edellä	-	-	-
Sääntö	**Organisatorinen sääntö** Organisatoriset säännöt ovat rajoitteita, joita organisaation agenttien tulee noudattaa.	-	-	-
Viesti	**Viesti** Käsite on määritelmän mukainen.	**Viesti** Käsite on määritelmän mukainen.	**Viesti** Käsite on määritelmän mukainen.	**Viesti** Käsite on määritelmän mukainen.
Protokolla	**Protokolla** Käsite on määritelmän mukainen.	**Protokolla** Käsite on määritelmän mukainen.	**Protokolla** Protokollat määritellään mallintamalla ne rinnakkaisina tehtävinä.	**Vuorovaikutusprotokolla** Käsite on määritelmän mukainen.
Rooli	**Rooli** Määrittelee agentin tehtävän ja aseman organisaatiossa. Tähän asemaan liittyy myös käyttäytyminen.	**Rooli** Roolilla tarkoitetaan sosiaalisen aktorin käyttäytymisen abstraktia luonnehdintaa tietyssä kontekstissa. *Positiolla* tarkoitetaan roolien joukkoa, jota tyypillisesti hoitaa yksi agentti.	**Rooli** Rooli määrittelee agentin tehtävän järjestelmässä. Jokainen rooli on vastuussa jonkin tavoitteen saavuttamisessa.	**Rooli** Rooli kuvailee agentin ulkoiset piirteet tietyssä kontekstissa. Agentilla voi olla monta roolia ja usea agentti voi esittää samaa roolia.
Palvelu	**Palvelu** Rajapinta, jonka agentti tarjoaa ulkomaailmalle. Palvelun käynnistämiseen ei välttämättä tarvita ulkoista pyyntöä, vaan agentti voi itse käynnistää palvelun. Palvelun ei myöskään välttämättä tarvitse käynnistyä ulkoisesta pyynnöstä, vaan agentti voi itsenäisesti päättää, toteuttaako se pyynnön (autonomia).	**Rajapinta** Rajapinnat ulkoisille aktoreille voidaan antaa yksittäisten aktoreiden tehtäväksi. Yksityiskohtaisen suunnittelun vaiheessa määritellään rajapinnat toteuttavat agentit.	**Palvelu** Menetelmässä ei varsinaisesti määritellä palveluita erikseen, mutta agenttien rajapinnat ulkomaailmalle määritellään yksityiskohtaisesti tehtävien ja keskusteluiden avulla. Rajapintojen määrittelyyn voi olla tarpeellista muodostaa oma roolinsa.	**Palvelu** Agentin toiminnalliset kyvyt kuvataan agentin palveluina. Agentin palvelu on analoginen olion operaation käsitteen kanssa.

(jatkuu)

TAULUKKO 9. Peruskäsitteiden merkitykset menetelmittäin (jatkuu)

Menetelmä/ Käsite	Gaia	Tropos	MaSE	MESSAGE
Tehtävä	**Aktiviteetti** Agentin rooliin liittyvä toimenpide, jonka agentti voi suorittaa itsenäisesti ilman vuorovaikutusta muiden agenttien kanssa.	**Pystyvyys (capability)** Tarkoittaa agentin kykyä määritellä, valita ja toteuttaa suunnitelma tavoitteen täyttämiseksi	**Tehtävä** Agentin rooliin liittyvä toimenpide, jonka agentti voi suorittaa itsenäisesti saavuttaakseen tavoitteensa.	**Tehtävä** Käsite on määritelmän mukainen.

Taulukon 9 perusteella voidaan tehdä vertailu siitä, miten menetelmät kattavat viitekehyksessä mainitut peruskäsitteet. Pääsääntöisesti käsitteet huomioidaan menetelmissä hyvin. Agentin käsite on samankaltainen kaikissa menetelmissä. BDI-mallin käsitteiden (uskomus, halu, aikomus) osalta menetelmissä esiintyy jonkin verran eroja. Gaia ei ota näitä käsitteitä huomioon lainkaan. Lisäksi MESSAGE ei huomioi aikomuksen käsitettä. Organisaation käsitteen osalta Gaia ja MESSAGE käsittelevät moniagenttijärjestelmää agenttien organisaationa. Kyseisissä menetelmissä järjestelmä pyritään mieltämään kuten reaalimaailman organisaatio. Troposissa ja MaSE:ssa organisaation käsitettä ei käytetä sellaisenaan, vaan todetaan, että järjestelmä muodostuu joukosta yhteistyötä tekeviä agentteja. Järjestelmällä tarkoitetaan samaa asiaa kaikissa menetelmissä, mutta Gaiassa ja MESSAGE:ssa järjestelmää korostetaan agenttien organisaationa. Yhteisön käsitteeseen kantaa ottaa ainoastaan Gaia. Muissa menetelmissä käsitettä ei tunneta. Samoin hieman yllättäen säännön käsite, siten kun se viitekehyksessä on määritelty, on selkeästi määritelty ainoastaan Gaiassa, jossa säännöt toteutetaan organisatorisina sääntöinä. Viestin käsite on selkeästi kaikissa menetelmissä sama, samoin protokolla. Myös roolin käsite on jokseenkin sama kaikissa menetelmissä. Ainoastaan käsitteen merkitys vaihtelee hieman menetelmien välillä. Palvelun käsite esiintyy kaikissa menetelmissä jossain muodossa. Troposissa puhutaan rajapinnasta, mutta käsitteen merkitys on sama. MaSE:ssa tosin palveluita ei määritellä eksplisiittisesti, mutta ne toteutuvat kuitenkin tehtävien ja keskusteluiden kautta. Tehtävän käsite on jokseenkin sama kaikissa menetelmissä.

Yhteenvetona taulukosta 9 voidaan sanoa, että menetelmät kattavat viitekehyksessä määritellyt peruskäsitteet varsin hyvin. Käsitteet ovat myös melko samankaltaisia kaikissa menetelmissä eikä suuria eroja esiinny. Käsitteiden nimityksistä tosin löytyy jonkin verran eroavaisuuksia.

5.1.2 Ominaisuudet

Taulukossa 10 vertaillaan, miten menetelmät huomioivat viitekehyksessä mainitut agenttien ominaisuudet. Ominaisuudet ovat autonomia, reaktiivisuus, proaktiivisuus ja sosiaalisuus.

TAULUKKO 10. Agenttien ominaisuudet menetelmittäin

Menetelmä/ Ominaisuus	Gaia	Tropos	MaSE	MESSAGE
Autonomia	Autonomia ilmenee agentille roolikaaviossa määriteltävän toiminnallisuuden kautta. (ks. kuvio 4). Roolikaaviossa agentille määritellään tarkasti ne aktiviteetit ja protokollat, joista se on vastuussa toimiessaan kyseisessä roolissa. Aktiviteetit ovat toimenpiteitä, joita agentti voi suorittaa ilman vuorovaikutusta muiden agenttien kanssa.	Agentin autonomia ilmaistaan yksityiskohtaisen suunnittelun vaiheessa määriteltävillä pystyvyyskaavioilla (capability diagrams). Pystyvyyskaaviot voidaan määritellä käyttäen UML:n toimintokaavioita.	Agentin autonomia huomioidaan roolikaaviossa määriteltävän toiminnallisuuden kautta. Rooleille määritellään tehtävät, jotka roolia esittävä agentti suorittaa itsenäisesti.	Autonomia määräytyy agentin *tarkoituksen* (purpose) kautta. Agentin tarkoitus vaikuttaa esimerkiksi siihen, suostuuko agentti tarjoamaan siltä pyydettyä palvelua. Tarkoitus vaikuttaa myös tapaan, jolla agentti palvelun tarjoaa.
Reaktiivisuus	Myös reaktiivisuus huomioidaan roolikaavioissa. Agentille määriteltyjen vastuiden turvallisuusominaisuudet määrittelevät järjestelmän halutun tilan. Jos tässä tilassa tapahtuu odottamattomia muutoksia, agentin tulisi reagoida siihen ja pyrkiä palauttamaan tilanne halutuksi. Tämä saattaa aiheuttaa muutoksia myös vastuiden elinkaariominaisuuksiin.	Agentin reaktiivisuus toteutuu agentin uskomusten kautta. Agentin uskomukset ovat sen tietoa ympäristöstään. Kun ympäristössä tapahtuu muutoksia, myös agentin uskomukset ympäristöstään muuttuvat. Toteutusvaiheessa uskomukset tallennetaan relaatiotietokantaan.	Agentin reaktiivisuus toteutuu uskomusten kautta. Uskomukset puolestaan määrittyvät agenteille määriteltävien tavoitteiden ja tehtävien kautta. Tehtävät määritellään yksityiskohtaisesti tilakaavioissa. Kun tehtävän tilassa tapahtuu muutos, on agentin reagoitava siihen.	Reaktiivisuus toteutuu agentin tietämyksen kautta. Kun agentin toimintaympäristössä tapahtuu, muutoksia, jotka vaikuttavat agentin tavoitteiden saavuttamiseen, on sen mukautettava toimintaansa uuden tietämyksen mukaisesti.

(jatkuu)

TAULUKKO 10. Agenttien ominaisuudet menetelmittäin (jatkuu)

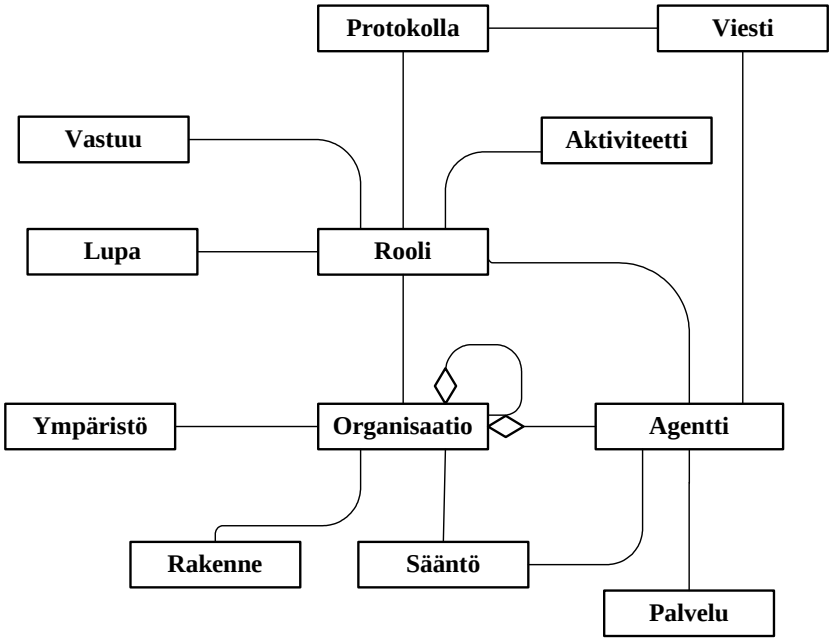
Menetelmä/ Ominaisuus	Gaia	Tropos	MaSE	MESSAGE
Proaktiivisuus	Proaktiivisuus huomioidaan roolikaaviossa määriteltyjen vastuiden elinkaariominaisuuksien kautta.	Proaktiivisuus ilmenee eri vaiheissa määriteltävien tavoitteiden kautta.	Proaktiivisuus huomioidaan analyysivaiheessa määriteltävien tavoitteiden kautta.	Agenttien proaktiivisuus huomioidaan analyysivaiheessa tehtävä/tavoite- ja roolikaavioissa.
Sosiaalisuus	Agentin sosiaalisuus ilmenee vuorovaikutusmallissa. Vuorovaikutusmalli koostuu roolien protokollamäärityksistä, joissa määritellään agenttien välisen kommunikaation säännöt ja tavat. Lisäksi organisatoriset rakenteet ja säännöt korostavat agenttien sosiaalisuutta.	Agentin sosiaalisuus ilmenee agenttien välisen vuorovaikutuksen mallintamisessa. Vuorovaikutuksen mallintamiseen voidaan käyttää esimerkiksi AUML:n sekvenssikaavioita.	Agentin sosiaalisuus huomioidaan suunnitteluvaiheessa, jossa luodaan agenttien väliset keskustelut. Keskustelut määritellään keskusteluluokkakaavioita käyttäen. Sosiaalisuus ilmenee myös käyttötapausten sekvenssikaavioissa.	Agentin sosiaalisuus huomioidaan menetelmän vuorovaikutusnäkymässä, jossa mallinnetaan agenttien välinen vuorovaikutus.

Helpoin tapa tutkia, miten menetelmät huomioivat agenttien ominaisuudet, on tarkastella menetelmien malleja. Menetelmät eivät eksplisiittisesti huomioi agenttien ominaisuuksia, mutta suurin osa ominaisuuksista tulee implisiittisesti huomioiduksi järjestelmän mallintamisen jossakin vaiheessa. Tätä näkökulmaa on käytetty taulukon 10 vertailussa. Taulukon 10 perusteella voidaan todeta, että kaikki vertailussa mukana olevat menetelmät huomioivat agenttien yleiset ominaisuudet hyvin. Tapa, jolla ominaisuudet huomioidaan, vaihtelee jonkin verran menetelmittäin. Gaiassa ja Troposissa autonomia ilmenee pitkälti samalla tavalla, eli roolikaavioissa määriteltävän toiminnallisuuden kautta. Troposissa autonomia puolestaan ilmaistaan vasta yksityiskohtaisen suunnittelun vaiheessa ja MESSAGE:ssa ominaisuus ei ilmene eksplisiittisesti missään malleissa, vaan abstraktisti agentin tarkoituksen kautta. Troposissa ja MaSE:ssa agenttien reaktiivisuus ilmenee agenttien uskomusten ja MESSAGE:ssa tietämyksen kautta. Gaia ei ota kantaa BDI-mallin käsitteisiin, joten siinä myös reaktiivisuus ilmenee roolikaavioissa. Agenttien proaktiivisuus ilmenee lähes kaikissa menetelmissä rooleille/agenteille määriteltävien tavoitteiden kautta. Tässäkin suhteessa Gaia on poikkeus, koska menetelmässä kyseinen ominaisuus ilmenee roolikaaviossa määriteltyjen vastuiden välityksellä. Agenttien sosiaalisuus ilmenee kaikissa menetelmissä jokseenkin samalla periaatteella, eli agenttien välisenä vuorovaikutuksena.

5.1.3 Käsiterakenteet

Seuraavaksi vertaillaan menetelmien käsiterakenteita käyttäen UML-notaatiota. Käsiterakenteissa esitetään peruskäsitteiden lisäksi myös menetelmien muut keskeiset käsitteet ja niiden väliset yhteydet. Peruskäsitteiden lisäksi muita keskeisiä käsitteitä ovat sellaiset käsitteet, jotka ovat merkittävässä asemassa menetelmässä. Tällä tarkoitetaan sitä, että ne esiintyvät menetelmän malleissa tai ovat muuten merkittävässä asemassa.

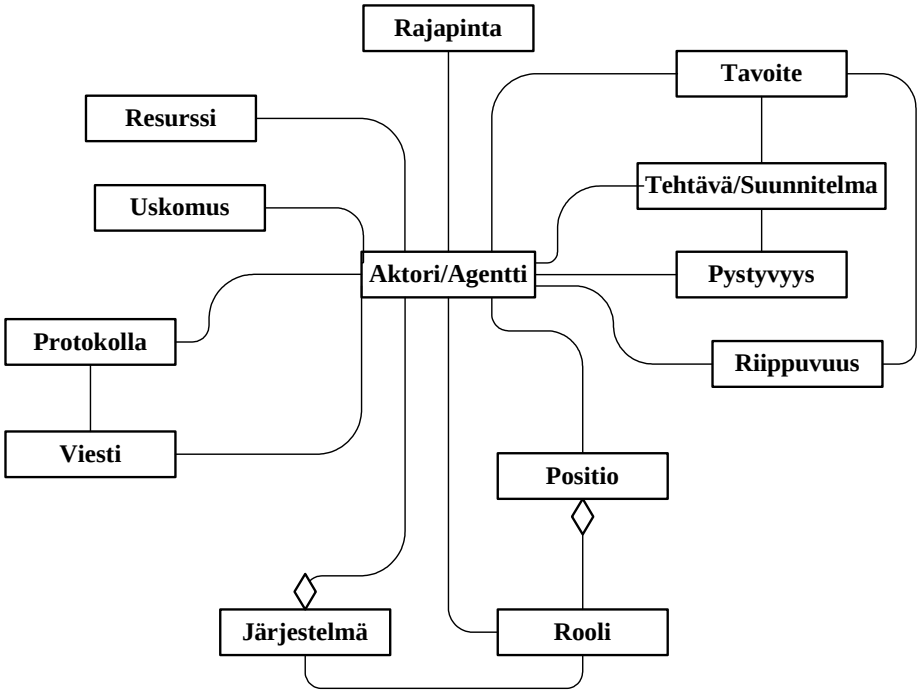
Kuviossa 21 esitetään käsiterakenne Gaian keskeisten käsitteiden osalta. Kuviosta käyvät ilmi myös käsitteiden väliset suhteet.



KUVIO 21. Gaian käsiterakenne.

Kuviosta nähdään, että roolin käsite menetelmässä on erittäin keskeinen. Roolin käsitteellä pyritään korostamaan Gaian tapaa käsitellä järjestelmän kehittämistä organisatorisen suunnittelun prosessina. Organisaatio koostuu joukosta agentteja, jotka toimivat määrätyssä roolissa organisaatiossa. Jokaisella roolilla puolestaan on vastuut ja luvat (tai oikeudet) sekä protokollat muiden roolien kanssa viestimiseen, aivan kuten reaalimaailman organisaatioissakin on. Usein rooleista ja agenteista puhuttaessa tarkoitetaan jotakuinkin samaa asiaa. Gaiassa niillä on kuitenkin selvä käsitteellinen ero. Kuten menetelmien esittelyn yhteydessä jo todettiin, agentti voi hoitaa useita rooleja, ja vastaavasti monet agentit voivat hoitaa samaa roolia. Tästä syystä vastuut, luvat, protokollat ja aktiviteetit määritellään rooleille eikä suoraan agenteille. Organisaatiolle määritellään organisatoriset säännöt, joita jokaisen organisaatioon kuuluvan agentin tulee noudattaa.

Kuviossa 22 esitetään Troposin käsiterakenne. Peruskäsitteiden lisäksi kuviossa on myös mallinnettu *resurssi*, joka ei määritelmältään sopinut peruskäsitteiden joukkoon. Resurssilla tarkoitetaan fyysistä resurssia tai informaatioresurssia, jota aktori tarvitsee tavoitteidensa saavuttamiseksi (Bresciani ym. 2004).

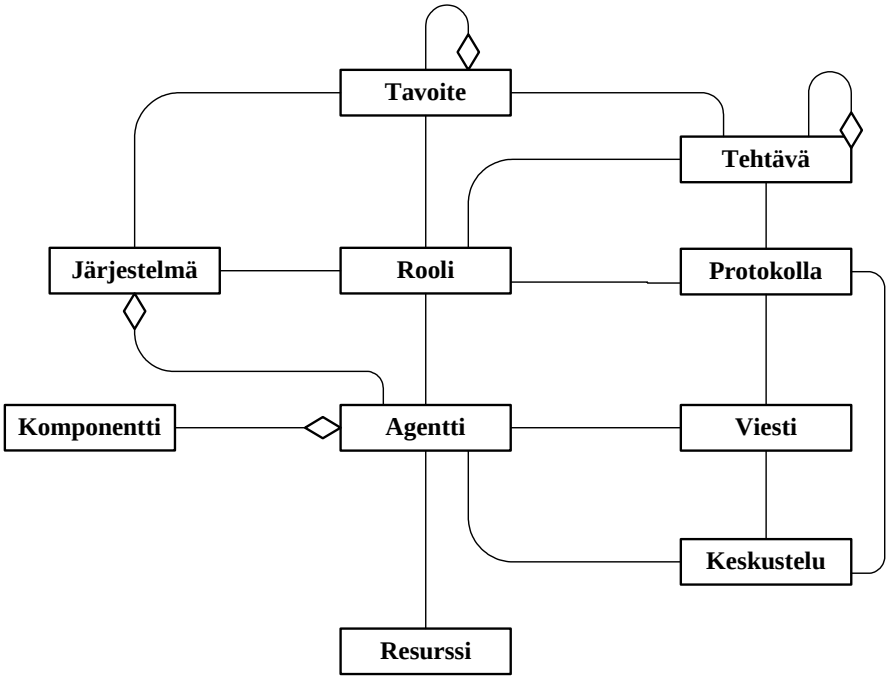


KUVIO 22. Troposin käsiterakenne.

Kuviosta nähdään, että keskeisessä asemassa on aktorin käsite. Roolin käsite on huomattavasti vähäisemmässä asemassa kuin muissa menetelmissä. Muuten sen merkitys on sama kuin muissa menetelmissä. Positiolla tarkoitetaan roolien joukkoa, jota tyypillisesti hoitaa yksi agentti. Järjestelmä koostuu joukosta aktoreita/agentteja. Agenteilla on tavoitteita ja tehtäviä/suunnitelmia, joilla tavoitteet voidaan saavuttaa. Tehtävän käsitteelle on Troposissa lähteestä riippuen esitetty kaksi erilaista nimitystä. Castro ym. (2002) käyttävät artikkelissaan tehtävän käsitettä, kun taas Bresciani ym. (2004) käyttävät suunnitelman käsitettä. Käsitteiden merkitys on kuitenkin sama. Agentille määritellään pystyvyyksiä, joiden avulla suunnitelmat voidaan toteuttaa. Agenttien väliseen viestintään käytetään jotakin valmiiksi määriteltyä protokollaa, kuten esimerkiksi FIPA ACL (FIPA 2003). Kuviossa 22 ei järjestelmää esitetä omana käsitteenään, koska se käsitetään Troposissa yhdeksi aktoriksi. Riippuvuus määritellään Troposissa eksplisiittisesti omana käsitteenään. Agenteilla on tavoitteet, joiden saavuttamiseksi ne ovat usein riippuvaisia toisista agenteista.

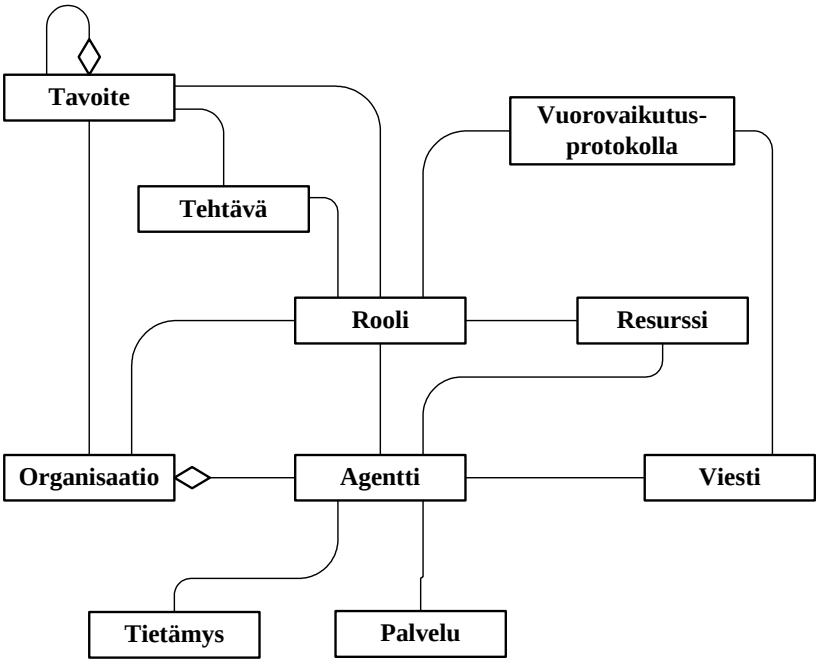
Kuviossa 23 esitetään MaSE:n käsiterakenne. MaSE:ssa järjestelmä koostuu joukosta agentteja, jotka esittävät yhtä tai useampaa roolia. Agentit puolestaan koostuvat sisäisistä komponenteista. Tavoitteita on kahdenlaisia: järjestelmätason tavoitteet ja

yksittäisten agenttien tavoitteet. Tavoitteet voidaan edelleen osittaa osatavoitteiksi. Myös tehtävät voidaan osittaa pienemmiksi kokonaisuuksiksi. Rooleille määritellään myös protokollat, joita rooleja esittävien agenttien on noudatettava keskinäisissä keskusteluissaan. Resurssilla tarkoitetaan ulkoisia resursseja, kuten ulkoiset järjestelmät ja tietokannat. Myös käyttäjät mielletään MaSE:ssa ulkoisiksi resursseiksi.



KUVIO 23. MaSE:n käsiterakenne.

Kuviossa 24 esitetään MESSAGE:n käsiterakenne. MESSAGE:n käsiterakenne muistuttaa selvästi Gaian ja MaSE:n käsiterakenteita. Järjestelmä nähdään organisaationa, joka koostuu joukosta agentteja. MESSAGE:ssa rooleille määritellään tavoitteet ja tehtävät, joista ne ovat vastuussa. Roolit ja agentit voivat olla yhteyksissä ulkoisiin resursseihin (ulkoiset järjestelmät, tietokannat jne.). Roolien välinen viestintä tapahtuu vuorovaikutusprotokollassa määriteltyjen sääntöjen mukaisesti. Agentin tietämys varastoidaan agentin tietämyskantaan.



KUVIO 24. MESSAGE:n käsiterakenne.

Menetelmien käsiterakenteita vertailtaessa huomataan, että Troposin käsiterakenne eroaa selvästi muiden menetelmien käsiterakenteista. Troposin käsiterakenteen erilaisuus johtuu siitä, että menetelmässä pyritään käyttämään samoja käsitteitä menetelmän koko prosessin ajan. Lisäksi menetelmä pohjautuu BDI-mallin käsitteille. Troposissa roolin käsite ei ole läheskään niin merkittävä kuin muissa menetelmissä. Muut vertailtavat menetelmät käyttävät roolin käsitettä eräänlaisena ”siltana” analyysi- ja suunnitteluvaiheiden välillä. Analyysivaiheessa on järkevää määritellä roolit, joita yksittäiset agentit esittävät. Siten voidaan suunnitteluvaiheessa lisätä järjestelmään uusia agentteja ilman, että sillä on välttämättä vaikutusta analyysivaiheen malleihin. Kaikki menetelmät määrittelevät jollain tavalla agenttien yhteydet ulkoisiin resursseihin (kuten tietokantoihin). Gaia käyttää tähän tarkoitukseen ympäristön käsitettä, kun taas muut menetelmät käyttävät resurssin käsitettä. Kaikissa menetelmissä määritellään myös agenttien välisen vuorovaikutuksen säännöt eli protokollat. Vaihtoehtona on joko määritellä säännöt itse tai sitten soveltaa jotakin valmiiksi määriteltyä protokollaa.

5.2 Mallit

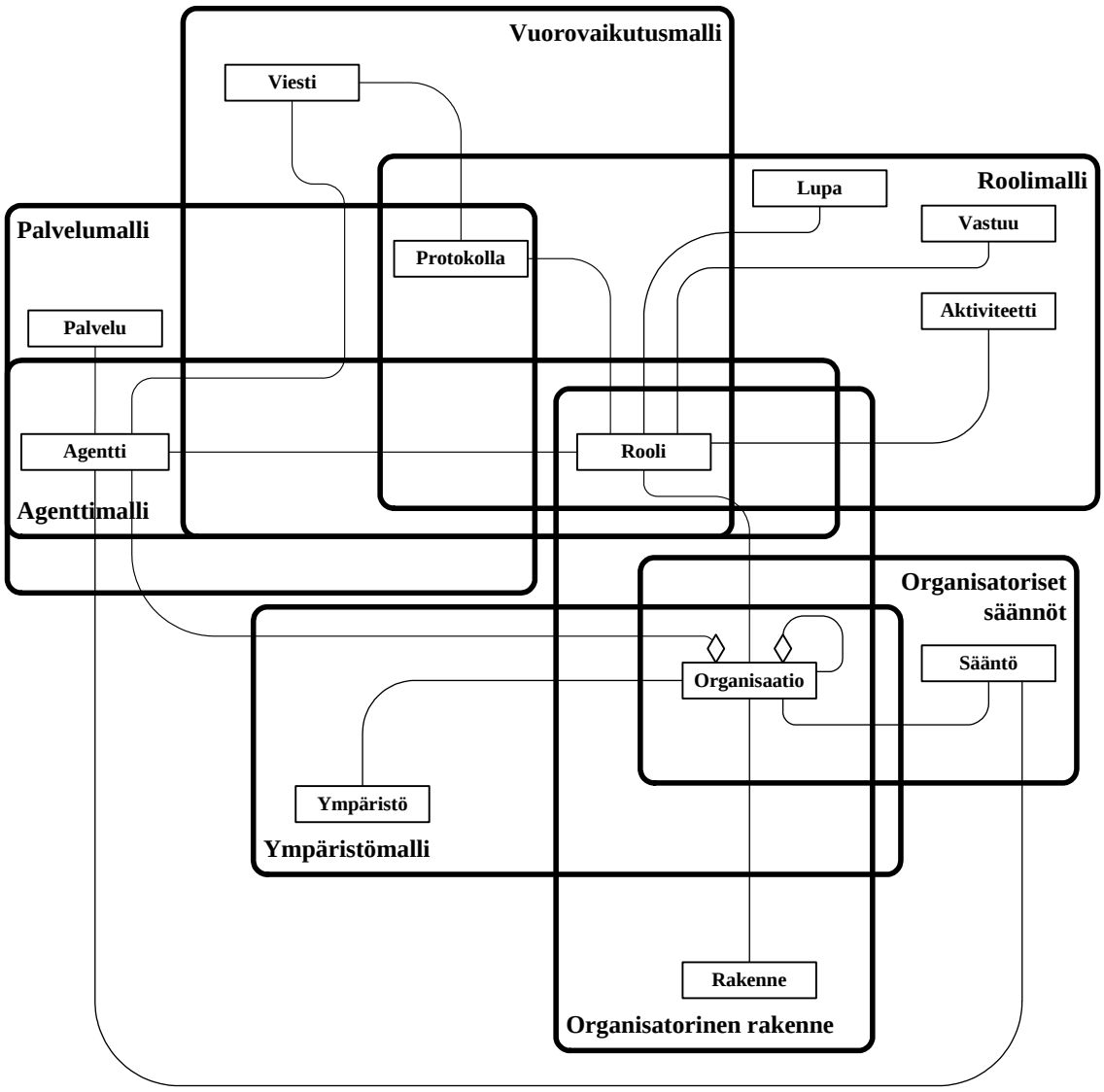
Tässä kohdassa vertaillaan menetelmien malleja. Aluksi selvitetään havainnollisten kuvioiden avulla kunkin menetelmän osalta mallien käsitteellinen sisältö, jonka jälkeen

arvioidaan mallien käyttämiä notaatioita. Lisäksi mallien osalta arvioidaan niiden monimutkaisuuden hallintaa, ilmaisuvoimaa ja modulaarisuutta.

5.2.1 Käsitteellinen sisältö

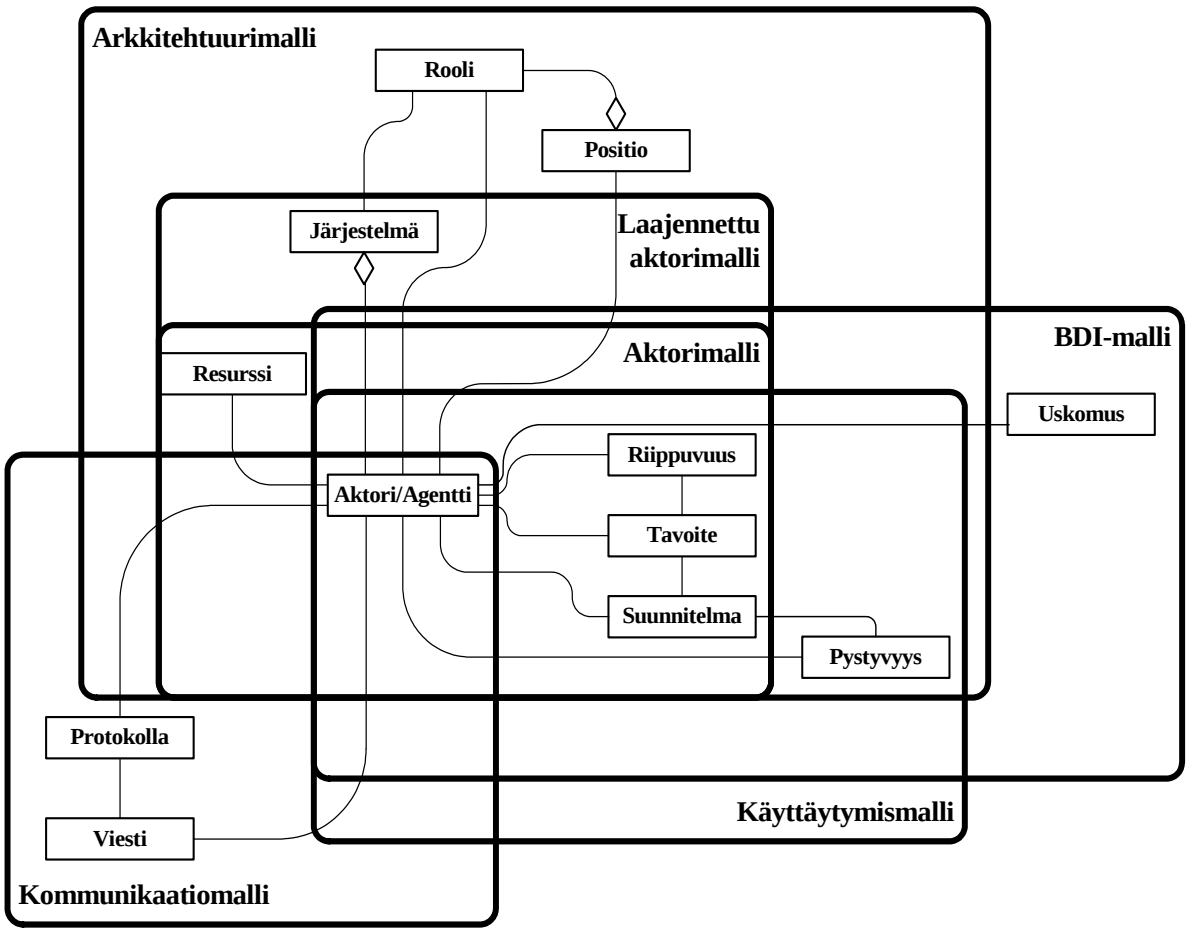
Seuraavaksi tarkastellaan edellisessä kohdassa esitettyjen käsitteiden sisältymistä menetelmien malleihin. Menetelmien käsitteellistä sisältöä havainnollistetaan kuvioiden avulla. Kuvioissa käsitteiden väliset suhteet ovat peräisin suoraan menetelmien käsiterakenteista. Malleja kuvataan paksunnetuin viivoin esitettävillä suorakaiteilla. Mallien päällekkäisyyksillä on pyritty tuomaan esiin sitä, miten käsitteet vaikuttavat eri malleissa. Malleja esittävien suorakaiteiden koolla ei sinänsä ole merkitystä, vaan jotkut suorakaiteet ovat piirrosteknisistä syistä esitetty suurempina tai pienempinä kuin toiset. Tässä yhteydessä ei kiinnitetä huomiota mallien välisiin edeltävyyssuhteisiin. Edeltävyyssuhteet ja mallien sijoittuminen menetelmän prosessin eri vaiheisiin on esitelty luvussa 3 menetelmien esittelyn yhteydessä. Tätä asiaa tarkastellaan myös alakohdassa 6.1.4.

Kuviossa 25 havainnollistetaan Gaian mallien käsitteellistä sisältöä. Kuviosta voidaan todeta, että roolimallin asema on hyvin keskeinen, aivan kuten roolin käsite on menetelmän käsiterakenteessa. Toinen keskeinen malli on vuorovaikutusmalli. Näillä kahdella mallilla on keskeinen vaikutus agentti- ja palvelumalleihin, joiden pohjalta järjestelmää aletaan toteuttaa. Kuviossa 25 ei ole alustavia vuorovaikutus- ja roolimalleja esitetty lainkaan, koska alustavissa malleissa esiintyvät samat käsitteet kuin lopullisissakin. Samojen käsitteiden esittäminen moneen kertaan tekisi kuviosta vain vaikeaselkoisemman. Roolimallissa protokollat esiintyvät roolien määrittelyn yhteydessä. Vuorovaikutusmallissa protokollamäärittelyt tehdään tarkemmalla tasolla.



KUVIO 25. Gaian mallit ja käsitteet.

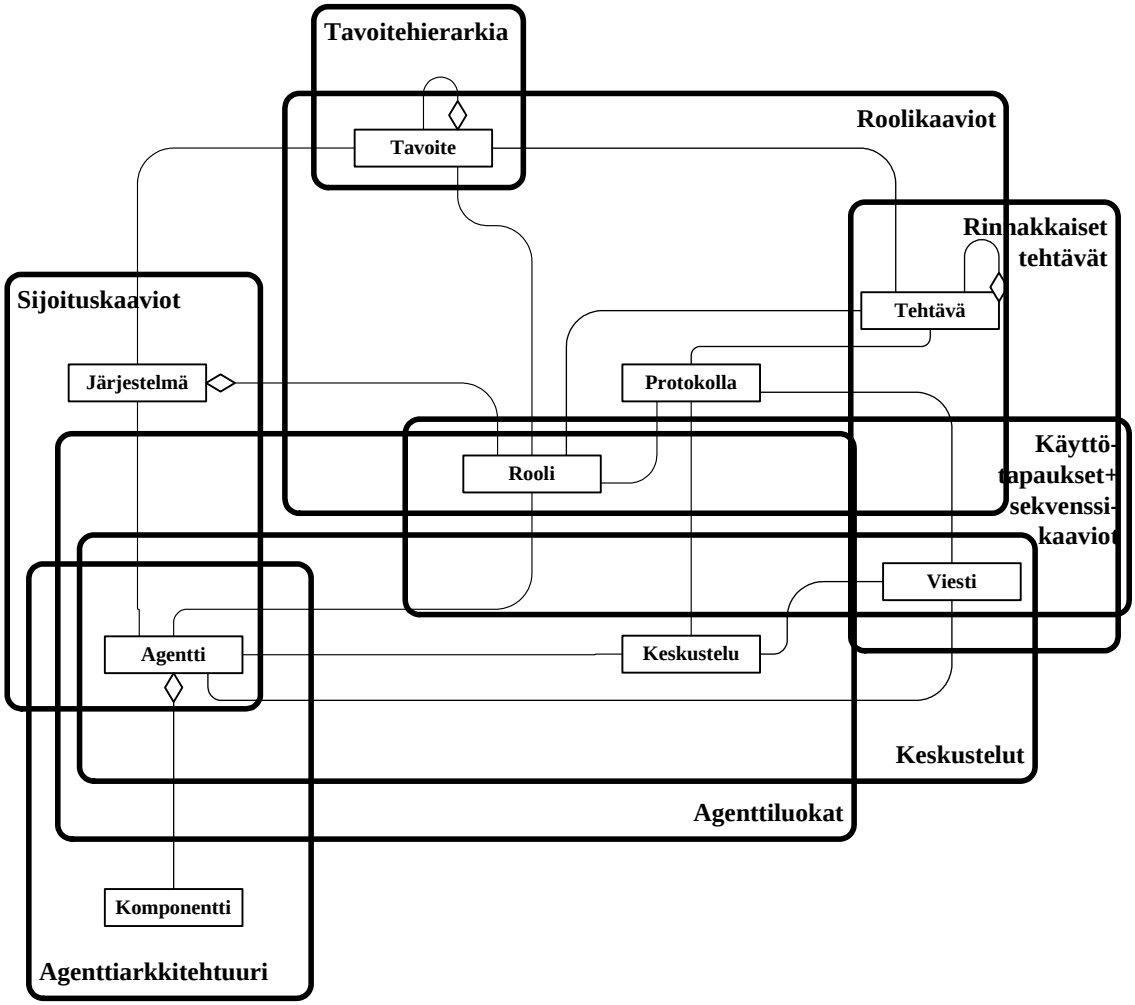
Kuviossa 26 havainnollistetaan Troposin mallien käsitteellistä sisältöä. Kuviosta nähdään, että aktorin käsite esiintyy kaikissa Troposin malleissa. Vaatimusmäärittely- ja arkkitehtuurisuunnitteluvaiheiden malleissa aktorilla tarkoitetaan yleensä sosiaalisia aktoreita, kuten esimerkiksi järjestelmän käyttöön liittyviä osapuolia. Yksityiskohtaisen suunnittelun vaiheessa aktorilla tarkoitetaan yleensä ohjelmistoagenttia. Kuviosta on jätetty pois strategiset perustelumallit, koska niissä esiintyvät käsitteet ovat täsmälleen samat kuin aktorimallissa ja laajennetussa aktorimallissa. Ainoastaan mallien käyttötarkoitukset eroavat toisistaan. Aktorimalleilla mallinnetaan kokonaisuuteen liittyviä asioita (vastaa kysymykseen ”mitä”), kun taas strategisilla perustelumalleilla mallinnetaan sitä, miten aktorit saavuttavat tavoitteensa.



KUVIO 26. Troposin mallit ja käsitteet.

Myös tässä yhteydessä on helppo todeta, että Troposissa roolin käsite on vähäisemmässä asemassa kuin muissa menetelmissä. Agenteille määritellään roolit vasta arkkitehtuurisuunnittelun yhteydessä, kun taas muissa menetelmissä roolit kuvataan jo analyysin alkuvaiheessa. Troposissa ei myöskään määritellä rooleille minkäänlaista roolimallia, kuten tehdään muissa menetelmissä. Troposissa keskeisimpänä käsitteenä on aktori, joka jossain muodossa esiintyy kaikissa Troposin malleissa.

Kuviossa 27 havainnollistetaan MaSE:n käsitteiden sijoittumista eri malleihin. Kuviosta nähdään, että myös MaSE:ssa roolimallin asema on hyvin keskeinen. Mallissa kuvataan roolien yhteydet tavoitteisiin, protokollat joiden mukaan roolia esittävät agentit viestivät sekä roolien tehtävät, joita yksittäiset agentit voivat suorittaa.

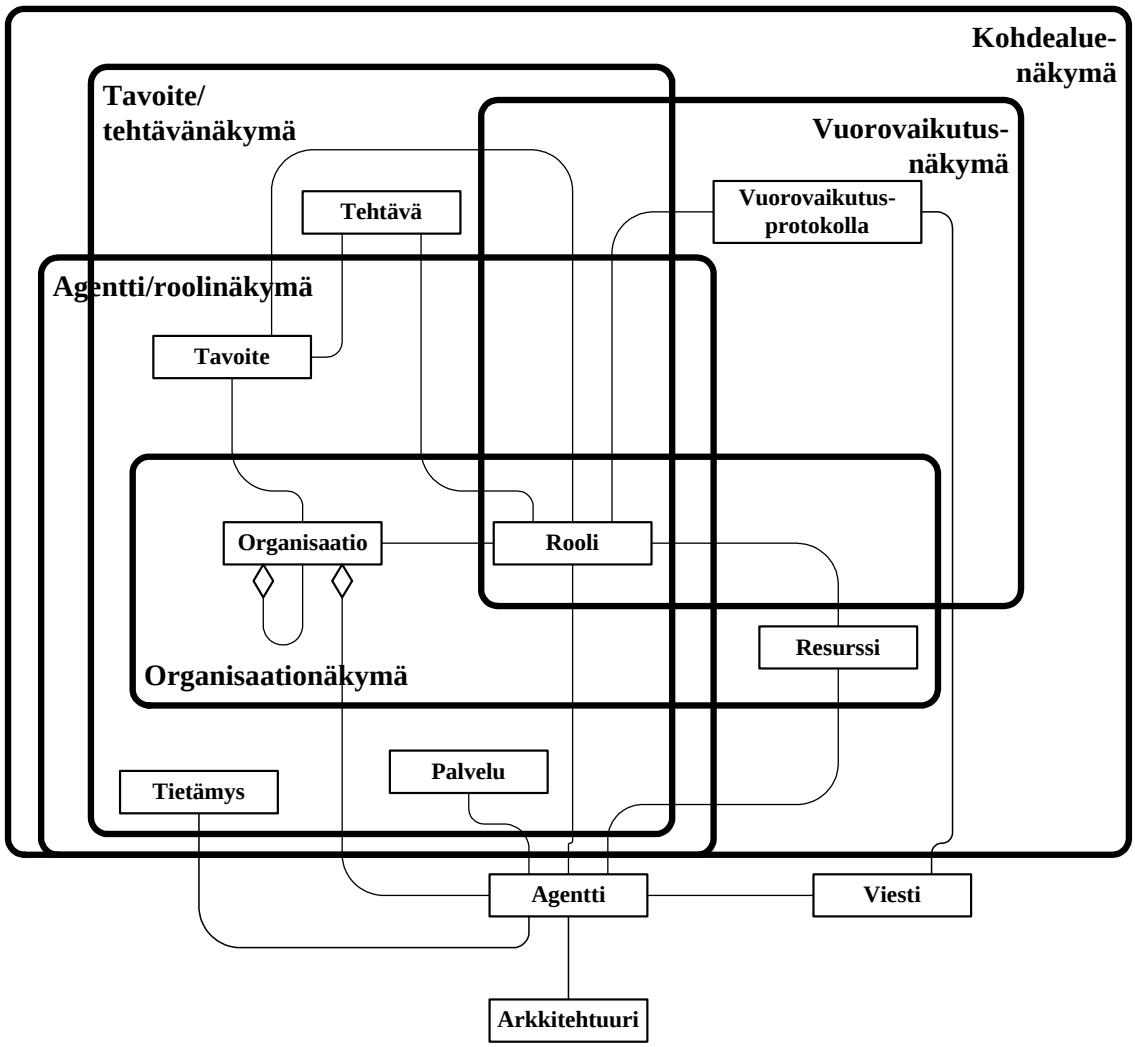


KUVIO 27. MaSE:n mallit ja käsitteet.

Kuviosta voidaan todeta, että MaSE:ssa analyysi- ja suunnitteluvaiheiden käsitteellinen ero on selvä. Ainoastaan viestin ja roolin käsitteet esiintyvät sekä analyysi- että suunnitteluvaiheen malleissa. Analyysivaiheen mallinnuskohteita ovat tavoitehierarkia, käyttötapaukset ja niihin liittyvät sekvenssikaaviot sekä rinnakkaiset tehtävät. Suunnitteluvaiheessa mallinnetaan agenttiluokat, agenttien väliset keskustelut, agenttiarkkitehtuuri sekä agenttien sijoittelu fyysisille alustoille.

Kuviossa 28 havainnollistetaan MESSAGE:n käsitteiden sisältymistä analyysivaiheen eri näkymiin. Kuviossa voitaisiin mallintaa myös näkymien sisältämät kaaviot, mutta tämä tekisi kuviosta turhan monimutkaisen. Kuviossa on esitetty ainoastaan menetelmän analyysivaiheen näkymät (tai mallit), koska menetelmä ei tarjoa selkeää esitystä suunnitteluvaiheen malleista. Menetelmä ohjeistaa ainoastaan tarkentamaan

analyysivaiheen näkymiä. Tästä Evans ym. (2001) antavat esityksessään joitakin esimerkkejä ja osittain tarkkojakin ohjeita, jotka ovat liian laajoja käsiteltäväksi tarkalla tasolla tutkielman laajuuden puitteissa. Lisäksi suunnitteluvaiheen mallinnus riippuu siitä, mikä lähestymistapa suunnitteluun valitaan. Evans ym. (2001) esittävät kaksi vaihtoehtoista lähestymistapaa suunnitteluun. Toinen tapa pohjautuu moniagenttijärjestelmän organisaatioon ja agenttiarkkitehtuuriin, kun taas toinen lähestymistapa on enemmän agenttialustasuuntautunut. Näistä ensimmäistä tapaa on kuvailtu tiivistetysti luvussa 3 menetelmän esittelyn yhteydessä.



KUVIO 28. MESSAGE:n analyysivaiheen mallit ja käsitteet.

Aluksi muodostetaan 0-tason organisaationäkymä, jossa mallinnetaan järjestelmän organisatorinen ympäristö. Seuraavaksi mallinnetaan tavoite/tehtävänäkymä, jossa

kuvataan järjestelmän tavoitteet. Agentti/roolinäkymässä määritellään, kuinka järjestelmälle asetetut tavoitteet jaetaan agenttien/roolien kesken. Vuorovaikutusnäkymässä puolestaan mallinnetaan roolien välisen vuorovaikutuksen säännöt eli vuorovaikutusprotokollat. Kohdealuenäkymää täydennetään koko mallinnusprosessin ajan sitä mukaa, kun uusia kohdealueelle tärkeitä käsitteitä ilmaantuu. Kuviossa 28 agentin, viestin ja arkkitehtuurin käsitteet jäävät analyysivaiheen mallinnuksen ulkopuolelle. Kyseiset käsitteet tulevat mukaan suunnitteluvaiheen mallinnuksessa, kun analyysivaiheen malleja tarkennetaan. Agenttiarkkitehtuuri valitaan tapauskohtaisesti. Suunnitteluvaiheen tärkeimmät mallinnuskohteet ovat juuri agenttiarkkitehtuurin ja agenttien välisen vuorovaikutuksen mallintaminen. Agenttien välisen vuorovaikutuksen yksityiskohdat saadaan esille tarkentamalla analyysivaiheen vuorovaikutusnäkymää.

Yhteenvetona voidaan sanoa, että menetelmien käsitteelliset sisällöt eroavat toisistaan huomattavasti enemmän kuin menetelmien käsiterakenteet. Myös mallien lukumäärä vaihtelee menetelmittäin. Ainoastaan Gaiassa on agenttien tarjoamille palveluille oma mallinsa. Muissa menetelmissä palveluita ei mallinneta eksplisiittisesti tai ne mallinnetaan muiden mallien yhteydessä. Roolin käsite näyttäisi Troposia lukuun ottamatta olevan hyvin keskeinen kaikissa menetelmissä. Gaiassa, MaSE:ssa ja MESSAGE:ssa käsite esiintyy useissa malleissa. Troposissa vastaavasti aktorin käsite on erittäin keskeinen. Se esiintyy menetelmän jokaisessa mallissa.

5.2.2 Notaatio

Tässä kohdassa arvioidaan ja vertaillaan menetelmien käyttämiä notaatioita. Jokaisen menetelmän osalta tutkitaan, mitä notaatiota tai notaatioita menetelmän malleissa käytetään. Lisäksi pyritään arvioimaan notaatioiden ominaisuuksia, kuten yksinkertaisuutta ja sitä, miten moninaisesti eri menetelmissä notaatioita käytetään.

Gaia ei käytä malleissaan mitään tarkasti määriteltyä notaatiota. Zambonelli ym. (2003) toteavat, että nykytilanteessa, jossa asianmukaisia notaatioita pyritään kehittämään vahvasti, tämä olisi ennenaikaista. Gaia kylläkin antaa joitain suosituksia notaatioiden käyttöön. Kuten jo aiemmin luvussa 3 menetelmän esittelyn yhteydessä todettiin, esimerkiksi ympäristömallin esittämiseen voidaan käyttää Colemanin ym. (1994)

FUSION-menetelmän operaatiokaavioiden notaatiota. Lisäksi vuorovaikutusmallin protokollamäärittelyjä voidaan tarkentaa AUML:n (Odell ym. 2001) protokollakaavioiden avulla. Myös organisatoristen sääntöjen kuvaamiseen annetaan hyvin väljät ohjeet. Zambonelli ym. (2003) toteavat, että organisatoristen sääntöjen esittämiseen voidaan käyttää jotakin formaalia notaatiota tai graafisia esityksiä, kuten AUML-kaavioita. Se, että menetelmä ei määrittele tarkkaa notaatiota, voidaan nähdä sekä positiivisena että negatiivisena asiana. Toisaalta suunnittelijalla on melko vapaat kädet muodostaa malleja niillä työkaluilla, joita hänellä on käytössään. Toisaalta taas menetelmän tarkoituksena tulisi olla tarjota selkeitä ohjeita mallien muodostamiseen ja tästä näkökulmasta katsoen määrätyn notaation puuttuminen saattaa aiheuttaa sekaannuksia joissakin tilanteissa. Zambonelli ym. (2003) toteavat, että AUML:n kehittyessä sen käyttö Gaian yhteydessä todennäköisesti lisääntyy.

Troposissa tukeudutaan vahvasti Yun (1995) i*-mallin käyttöön vaatimusmäärittely-analyysi- ja arkkitehtuurisuunnittelun vaiheissa. i*-mallin notaatio on kohtalaisen yksinkertainen ja helppo oppia. Sillä voidaan mallintaa aktorit, tavoitteet (myös yleistavoitteet), resurssit ja suunnitelmat sekä näiden väliset riippuvuudet. Samaa notaatiota johdonmukaisesti käyttäen voidaan edetä yksityiskohtaisen suunnittelun vaiheeseen saakka. Yksityiskohtaisen suunnittelun vaiheessa menetelmä tukeutuu vahvasti UML:n ja AUML:n (Odell ym. 2001) käyttöön. Agenttien vuorovaikutuksen mallintamiseen käytetään AUML:n sekvenssikaavioita. Agenttien pystyvyyksien mallintamiseen puolestaan UML:n toimintokaaviot sopivat hyvin. Agenttien mallintamiseen voidaan käyttää tavallisia UML:n luokkakaaviota.

MaSE:ssa menetelmän kehittäjät ovat määritelleet osittain oman notaationsa menetelmän käyttöä varten ja osittain mallinnuksessa tukeudutaan UML:n käyttöön. Analyysivaiheessa ensimmäisenä muodostettava tavoitehierarkia esitetään suunnattuna verkkona, lähtösolmuna toimii järjestelmän päätavoite ja muina solmuina osatavoitteet (DeLoach ym. 2001). Käyttötapausten kuvaaminen tapahtuu tekstimuotoisesti sekvenssikaavioita apuna käyttäen. Periaatteessa käyttötapausten havainnollistamiseen voidaan käyttää myös UML:n käyttötapauskaavioita, vaikka menetelmän kehittäjät eivät tätä mainitsekaan. Sekvenssikaavioilla kuvataan roolien väliset tapahtumat ja viestit. Roolimalli kuvataan käyttäen yksinkertaista, mutta hyvin informatiivista graafista

esitysmuotoa (ks. KUVIO 12). Analyysivaiheen lopuksi roolien tehtävät määritellään tarkemmalla tasolla UML:n notaation mukaisilla tilakaavioilla. Suunnitteluvaiheessa määritellään agenttiluokkakaavio, agenttien väliset keskustelut, agenttiarkkitehtuuri ja järjestelmän fyysinen arkkitehtuuri. Agenttiluokkakaavion notaatio on samankaltainen UML:n olioluokkakaavion notaation kanssa. Agenttien väliset keskustelut määritellään keskusteluluokkakaavioissa. jotka muodostuvat samanlaisista tilakoneista kuin rinnakkaisten tehtävien määrittelyt analyysivaiheessa. Agenttiarkkitehtuurin määrittelyssä agentin komponentit ilmaistaan UML:n luokkakaavion notaatiolla. Lopuksi järjestelmän fyysinen arkkitehtuuri ilmaistaan graafisesti yksinkertaisella sijoituskaaviolla, joissa kuvataan järjestelmän agenttien lukumäärät, tyypit ja sijainnit.

MESSAGE tukeutuu hyvin vahvasti UML:n käyttöön lähes kaikissa mallinnuksen vaiheissa. Notaatio on muuten UML:n mukainen, mutta menetelmän kehittäjät ovat laajentaneet UML:ää tukemaan paremmin agenttipohjaisten järjestelmien suunnittelua. UML:n lisäksi käytetään myös tekstimuotoisia kuvauksia. Näin menetellään esimerkiksi analyysivaiheessa roolikaavioita muodostettaessa. Suunnitteluvaiheessa käytetään hyvin pitkälle UML:n mukaista notaatiota.

Menetelmien käyttämät notaatiot poikkeavat toisistaan paljon. Kuten aiemmin tutkielmassa todettiin, Gaia ei käytä mitään määrättyä notaatiota malleissaan. Ainoastaan roolikaavioille, protokollamäärityksille ja agenttimallille annetaan selkeät esimerkit (Zambonelli ym. 2003). Muuten notaatioiden käyttöön annetaan suunnittelijalle melko vapaat kädet. Tosin tämä tulee todennäköisesti muuttumaan tulevaisuudessa. Tropos puolestaan käyttää johdonmukaisesti i*-mallin mukaista notaatiota yksityiskohtaisen suunnittelun vaiheeseen asti. Yksityiskohtaisen suunnittelun vaiheessa voidaan käyttää UML:ää ja AUML:ää. MaSE on notaatioiden käytön suhteen monimuotoisin. Lähes kaikissa menetelmän malleissa on käytössä erilainen notaatio. Tämä saattaa aiheuttaa sen, että menetelmän käyttö koetaan työlääksi oppia. Notaatio ei sinänsä ole kovin vaikea oppia ja jos UML:n tilakaaviot ovat tuttuja, on loput notaatiot melko helposti omaksuttavissa. MESSAGE tukeutuu lähes kokonaan UML:n käyttöön mallinnuksessaan. Menetelmän analyysivaihetta varten menetelmän kehittäjät ovat laajentaneet UML:ää (ks. Evans ym. 2001, 57-63). Suunnitteluvaiheessa

käytetään pääasiassa perinteistä UML-notaatiota. Jos UML on tuttu entuudestaan, menetelmän notaatio on helposti omaksuttavissa.

5.2.3 Monimutkaisuuden hallinta

Monimutkaisuuden hallinnalla tarkoitetaan menetelmän kykyä tukea järjestelmän kehittämisen eri vaiheita eri abstraktiotasoilla. Järjestelmän kehittämisen eri vaiheissa on tarpeellista esittää järjestelmään liittyviä kuvauksia eri abstraktiotasoilla. Esimerkiksi analyysivaiheessa asiat esitetään huomattavasti yleisemmällä tasolla kuin suunnitteluvaiheessa.

Gaiassa monimutkaisuuden hallinta on toteutettu jakamalla menetelmä kahteen vaiheeseen: analyysiin ja suunnitteluun. Suunnitteluvaihe jakautuu edelleen kahteen askeleeseen eli arkkitehtuurisuunnitteluun ja yksityiskohtaiseen suunnitteluun. Myös mallien yksityiskohtaisuus noudattelee tätä jakoa. Analyysivaiheessa muodostettavat mallit ovat ympäristömalli, alustavat rooli- ja vuorovaikutusmallit sekä organisatoriset säännöt. Nämä esitetään melko yleisellä tasolla. Esimerkiksi ympäristömalli voi olla yksinkertainen graafinen kuvaus järjestelmän toimintaympäristöstä. Tässä vaiheessa rooli- ja vuorovaikutusmallit ovat nimensä mukaisesti alustavia. Niissä olevat tiedot ovat tässä vaiheessa yleisiä ja osa tiedoista voi vielä puuttuakin. Alustavassa roolimallissa mallinnetaan ainoastaan sellaiset roolit, jotka voidaan tunnistaa analyysivaiheessa ilman sitoutumista mihinkään tiettyyn arkkitehtuuriratkaisuun. Samoin menetellään alustavan vuorovaikutusmallin kohdalla. Arkkitehtuurisuunnittelun malleissa siirrytään jo tarkemmalle tasolle. Aluksi määritellään järjestelmän organisaatiorakenne (eli järjestelmän arkkitehtuuri). Gaia ei ohjeista arkkitehtuurin muodostamista, vaan pikemminkin sen valintaa olemassa olevista ratkaisuista. Kun sopiva arkkitehtuuri on valittu, tulee se esittää jollakin tavalla. Tähän tarkoitukseen Zambonelli ym. (2003) esittävät ratkaisuksi formaalin notaation ja graafisen esityksen yhdistelmää. Kun organisaatiorakenne on muodostettu, voidaan täydentää rooli- ja vuorovaikutuskaaviot. Lopullisissa kaavioissa on kuvattu yksityiskohtaisesti kaikki roolien ominaisuudet, jotka tulevat toteutumaan lopullisessa järjestelmässä (Zambonelli ym. 2003). Lopuksi yksityiskohtaisen suunnittelun aikaisessa mallinnuksessa siirrytään kaikista yksityiskohtaisimmalle tasolle. Tässä vaiheessa muodostetaan agentti- ja palvelumallit. Mallit muodostetaan luvussa 3 esitetyllä tavalla. Agenttimallissa

määritellään yksityiskohtaisesti agenttien ja roolien vastaavuudet sekä kuinka monta kunkin agenttiluokan ilmentymää järjestelmässä esiintyy. Palvelumallissa puolestaan määritellään agenttien tarjoamat palvelut.

Myös Troposissa monimutkaisuuden hallinta on toteutettu hyvin. Menetelmä on ositettu kolmeen vaiheeseen, jotka ovat vaatimusmäärittely, suunnittelu, ja toteutus. Vaatimusmäärittelyvaihe sisältää kaksi askelta. Askeleet ovat aikaiset vaatimukset ja myöhäiset vaatimukset. Myös suunnitteluvaihe sisältää kaksi askelta, jotka ovat arkkitehtuurisuunnittelu ja yksityiskohtainen suunnittelu. Troposin mallien monimutkaisuus noudattelee tätä jakoa hyvin. Aikaisten vaatimusten määrittelyssä lähdetään liikkeelle erittäin yksinkertaisesta mallinnuksesta (ks. esim. KUVIO 7), jolloin määritellään aktorimalli. Myöhäisten vaatimusten määrittelyssä mallia tarkennetaan lisäämällä mukaan uusia tavoitteita ja aktoreita ja mukaan lisätään myös kehitettävä järjestelmä. Tarkentamista jatketaan edelleen arkkitehtuurisuunnittelun aikana, jolloin myös muodostetaan järjestelmälle arkkitehtuuri. Yksityiskohtaisen suunnittelun aikaisten mallien tarkkuudessa päästään tasolle, josta toteutus voidaan hyvin suorittaa.

MaSE jakautuu analyysi- ja suunnitteluvaiheisiin, jotka jakautuvat edelleen yhteensä seitsemään askeleeseen. Monimutkaisuuden hallinta on toteutettu siten, että sekä analyysi- että suunnitteluvaiheessa esiintyy kohtalaisen yksinkertaisten mallien lisäksi myös monimutkaisempia malleja. Analyysivaiheen yksinkertaisimmat mallit ovat tavoitehierarkia ja käyttötapausten määrittelyt. Roolimalli on jo hieman mutkikkaampi ja monimutkaisimmat mallit muodostuvat roolien rinnakkaisten tehtävien määrittelyssä käytettävistä tilakoneista. Myös suunnitteluvaiheen keskusteluluokkakaavioissa käytetään tilakoneita, mistä johtuen kaaviot ovat melko mutkikkaita ja yksityiskohtaisia. Agenttiluokkakaaviot ja agenttiarkkitehtuurien määrittelyt ovat taas yksinkertaisempia ja kaikkein yksinkertaisin malli, sijoittelukaavio, muodostetaan suunnitteluvaiheen lopuksi.

MESSAGE:ssa monimutkaisuuden hallinta toteutuu analyysivaiheessa huomattavasti paremmin kuin suunnitteluvaiheessa. Analyysivaiheessa mallit muodostavat eritasoisia näkymiä. Ensin muodostetaan 0-tason näkymät, jonka jälkeen tarkennetaan malleja tarvittaessa 1-tason näkymiksi. Näkymiä on viisi: organisaationäkymä,

tavoite/tehtävänäkymä, agentti/roolinäkymä, vuorovaikutusnäkymä ja kohdealuenäkymä. Suunnitteluvaiheessa malleilla ei enää ole selkeää luokittelua vaan analyysivaiheen malleja tarkennetaan käyttämällä pääasiassa UML:n luokkakaavioita.

Yhteenvetona voidaan sanoa, että pääpiirteittäin monimutkaisuuden hallinta toteutuu menetelmissä hyvin. Tosin MESSAGE:n osalta voidaan moittia suunnitteluvaiheen lievää sekavuutta, jossa monimutkaisuutta ei pystytä hallitsemaan niin hyvin kuin muissa menetelmissä.

5.2.4 Ilmaisuvoima

Seuraavaksi arvioidaan menetelmien ilmaisuvoimaa viitekehyksen kohdelistan mukaisesti. Taulukossa 11 on esitetty yhteenveto siitä, kuinka menetelmät kattavat viitekehyksessä mainitut kohteet.

Taulukosta 11 nousee esiin selvästi kaksi seikkaa. Missään vertailtavista menetelmistä ei mallinneta agenttien liikkuvuutta. Tämä on hieman yllättävää ottaen huomioon sen, että liikkuvista agenteista on viimeaikaisessa kirjallisuudessa kirjoitettu paljon ja näyttäisi siltä, että liikkuviin agentteihin perustuvien sovellusten lukumäärä tulee tulevaisuudessa lisääntymään. Toinen merkittävä seikka on käyttöliittymämäärittelyiden laiminlyönti. Tosin Tropos ja MaSE huomioivat käyttöliittymän mainitsemalla, että sitä varten voidaan muodostaa oma rooli (MaSE) tai aktori (Tropos). Järjestelmien käytettävyys on tärkeä seikka tietojärjestelmän rakentamisessa ja juuri käytettävyyden osalta käyttöliittymä on ratkaisevassa asemassa. Tämä onkin seikka, johon tulevaisuudessa tulisi kiinnittää erityistä huomiota. Toisaalta menetelmät on tarkoitettu sovellettaviksi monille kohdealueille ja käyttöliittymän suunnittelu riippuu pitkälti myös kohdealueesta. Aina käyttöliittymää ei edes tarvitse suunnitella (vrt. sulautetut järjestelmät). Siten käyttöliittymäsuunnittelun lisääminen menetelmään voisi rajoittaa menetelmän käytettävyyttä eri kohdealueilla. Edellä mainittujen seikkojen lisäksi menetelmistä löytyy jossain määrin pieniä puutteita. Gaiassa ei mallinneta järjestelmän tietovirtoja lainkaan graafisesti. Myös järjestelmän vuorovaikutus ulkoisten järjestelmien kanssa jää mallintamatta. Menetelmistä ainoastaan MaSE huomioi jollain tasolla järjestelmän fyysisen arkkitehtuurin sijoittelukaavioiden muodossa.

TAULUKKO 11. Menetelmien ilmaisuvoiman arviointi viitekehyksen kohdelistan mukaisesti

Menetelmä/ Kohde	**Gaia**	**Tropos**	**MaSE**	**MESSAGE**
Järjestelmän rakenne	Järjestelmän rakenne esitetään arkkitehtuurisuunnittelun vaiheessa organisatorista rakennetta muodostettaessa.	Järjestelmän rakenne esitetään arkkitehtuurisuunnittelun yhteydessä i*-mallin notaatiota käyttäen.	Ilmaistaan roolimallissa ja agenttiarkkitehtuurin määrittelyissä.	Ilmaistaan analyysivaiheessa organisaatiokaavioissa ja suunnitteluvaiheessa tarkemmalla tasolla järjestelmän arkkitehtuurin suunnittelun yhteydessä.
Järjestelmän tietovirrat	Tietovirtojen esittämiseen ei Gaiassa ole graafista esitysmuotoa.	Järjestelmän tietovirrat esitetään i*-mallin mukaisesti aktoreiden välisinä riippuvuuksina.	Mallinnetaan käyttötapausten sekvenssikaavioissa sekä roolimallissa roolien viestintäsuhteina.	Tietovirtoja mallinnetaan työnkulkukaavioilla.
Järjestelmän ja agenttien sisäiset rinnakkaiset toiminnot	Ei esitetä eksplisiittisesti.	Järjestelmän rinnakkaisia toimintoja ei esitetä menetelmässä eksplisiittisesti. Agenttien sisäiset rinnakkaiset toiminnot voidaan puolestaan esittää suunnitelmakaavioissa.	Järjestelmän rinnakkaiset toiminnot voidaan mallintaa suunnitteluvaiheessa viestintäluokkakaaviossa mallintamalla agenttien välinen viestintä. Agenttien sisäiset rinnakkaiset toiminnot mallinnetaan rinnakkaisten tehtävien kaaviossa analyysivaiheessa.	Ei esitetä menetelmässä eksplisiittisesti.
Järjestelmän resurssirajoitteet (esim. aika, prosessoriteho ja muistin määrä)	Esitetään osittain rooleille määriteltyjen lupien kautta. Lupamäärittelyissä määritellään muun muassa ne arvot, joita rooli saa muuttaa. Järjestelmän fyysisiä resurssirajoitteita ei määritellä eksplisiittisesti.	Ei esitetä eksplisiittisesti.	Resurssirajoitteet huomioidaan suunnitteluvaiheen lopuksi sijoittelukaaviota muodostettaessa.	Järjestelmän resurssirajoitteita ei mallinneta.

(jatkuu)

TAULUKKO 11. Menetelmien ilmaisuvoiman arviointi viitekehyksen kohdelistan mukaisesti (jatkuu)

Menetelmä/ Kohde	**Gaia**	**Tropos**	**MaSE**	**MESSAGE**
Järjestelmän fyysinen arkkitehtuuri	Ei mallinneta eksplisiittisesti.	Järjestelmän fyysistä arkkitehtuuria ei kuvata millään lailla.	Fyysinen arkkitehtuuri voidaan ilmaista sijoittelukaaviossa.	Järjestelmän fyysistä arkkitehtuuria ei menetelmässä mallinneta.
Agenttien liikkuvuus	Ei mallinneta.	Ei mallinneta.	Ei mallinneta.	Ei mallinneta.
Järjestelmän vuorovaikutus ulkoisten järjestelmien kanssa	Ei esitetä eksplisiittisesti.	Voidaan mallintaa i*-mallin mukaisilla aktorikaavioilla.	Ilmaistaan agenttiarkkitehtuurin määrittelyssä yhteyksinä ulkoisiin resursseihin.	Voidaan mallintaa organisaationäkymässä organisaatiokaavioita käyttäen.
Käyttöliittymämäärittelyt	Ei huomioida lainkaan..	Ei huomioida muuten kuin mainitsemalla, että käyttöliittymä voidaan mallintaa yhtenä aktorina.	Varsinaisia käyttöliittymämäärittelyjä ei esitetä. Menetelmän kehittäjät huomioivat käyttöliittymän kuitenkin mainitsemalla, että järjestelmän käyttöliittymää varten tulisi muodostaa oma roolinsa. Käyttäjä nähdään ulkoisena resurssina aivan kuten tietokannat, tiedostot ja ulkoiset järjestelmätkin. Myös näitä varten tulisi muodostaa omat rajapintaroolit.	Menetelmä ei anna tukea käyttöliittymien määrittelyyn.

Edellä mainittuja puutteita lukuun ottamatta mallien ilmaisuvoima toteutuu melko hyvin tässä tutkielmassa käytetyn viitekehyksen kriteerien mukaan. Järjestelmän rakenne esitetään selkeästi kaikissa menetelmissä. Myös järjestelmän tietovirrat esitetään hyvin kaikissa muissa menetelmissä paitsi Gaiassa. MaSE:ssa tulevat hyvin ilmi myös järjestelmän ja agenttien sisäiset rinnakkaiset toiminnot. Myös Tropos tarjoaa hyvän mallin agenttien sisäisten toimintojen mallintamiseen.

5.2.5 Modulaarisuus

Modulaarisuus tarkoittaa menetelmän kykyä määritellä järjestelmä iteratiivisella ja inkrementaalisella tavalla. Tämä tarkoittaa sitä, että järjestelmän kehittämisen aikana ilmenevien uusien vaatimusten ei pitäisi vaikuttaa olemassa oleviin spesifikaatioihin.

Gaia toteuttaa malliensa osalta modulaarisuusperiaatetta pääpiirteissään hyvin. Esimerkiksi roolimalliin voidaan lisätä tarpeen mukaan uusia rooleja ilman, että sillä on vaikutusta olemassa olevien roolien määrittelyihin. Samoin agenttimallin agenteille voidaan antaa rooleja ja usealle agentille voidaan antaa sama rooli ilman, että sillä on vaikutusta olemassa oleviin määrittelyihin. Tosin uusien roolien lisääminen saattaa aiheuttaa muutoksia vuorovaikutusmallissa. Lisäksi muutokset protokollissa saattavat aiheuttaa muutoksia roolimäärityksissä. Esimerkiksi jos jokin protokolla päätetään poistaa kokonaan, tällöin on myös muutettava niiden roolien määrityksiä, joissa kyseinen protokolla on ollut mukana.

Troposin malleissa modulaarisuus toteutuu kohtalaisen hyvin. Jos järjestelmänkehityksen aikana esitetään järjestelmälle uusia vaatimuksia, aiheuttaa tämä todennäköisesti uusien aktoreiden, tavoitteiden, suunnitelmien ja resurssien sekä näiden välisten riippuvuuksien lisäämistä aktorikaavioihin. Agenttien välisiin vuorovaikutussuhteisiin uudet vaatimukset luonnollisesti saattavat vaikuttaa, mutta agenttien sisäiset toiminnot on toteutettu modulaarisesti suunnitelmakaavioiden muodossa.

MaSE:n malleissa modulaarisuusperiaate toteutuu varsin hyvin. Roolimäärittelyt toimivat pohjana agenttiluokkien suunnittelulle. Roolit muodostavat sillan analyysivaiheen tavoitteiden ja suunnitteluvaiheen agenttiluokkien välille, eli sille, mitä

järjestelmä yrittää tavoittaa ja miten se yrittää tavoitteensa saavuttaa. Suunnittelija voi helposti muutella roolien organisointia ja allokointia agenteille, koska roolimäärittelyt ovat modulaarisia. (DeLoach ym. 2001) Modulaarisuus toteutuu myös agenttiarkkitehtuurin määrittelyssä.

MESSAGE:ssa modulaarisuusperiaate toteutuu hyvin eri näkymien välillä. Jos joissain näkymissä tapahtuu muutoksia, sillä ei ole automaattisesti kovin suurta vaikutusta muihin näkymiin. Varsinkin roolit määritellään hyvin modulaarisesti aivan kuten Gaiassa ja MaSE:ssa. Menetelmän vaiheiden välillä modulaarisuus ei toteudu niin hyvin kuin esimerkiksi Gaiassa tai MaSE:ssa.

5.3 Prosessi

Tässä kohdassa arvioidaan ja vertaillaan menetelmien mukaisia prosesseja viitekehyksessä annetun kriteerilistan mukaisesti. Prosessien osalta vertaillaan elinkaaren kattavuutta, tuotoksia ja tukea johtamiselle.

5.3.1 Elinkaaren kattavuus

Kuviossa 29 havainnollistetaan menetelmien kattavuutta viitekehyksessä määritellyn vaihejaon avulla. Kuviossa vinoviivoin esitetyt alueet tarkoittavat vaiheita, jotka menetelmät selkeästi kattavat ja joihin on annettu ohjeet. Harmaat alueet puolestaan tarkoittavat vaiheita, joihin menetelmä ei anna ohjeita, mutta menetelmän kehittäjät kuitenkin käsittelevät alueita tavalla tai toisella menetelmän yhteydessä. Valkoiset alueet ovat vaiheita, joita menetelmät eivät huomioi millään tasolla. Kuvion 29 ajatusmalli on samankaltainen Avisonin ja Fitzgeraldin (1995, 465) viitekehyksessä esitetyn mallin kanssa.

Vaihe / Menetelmä	Vaatimus-määrittely	Analyysi	Suunnittelu	Toteutus	Testaus	Ylläpito
Gaia						
Tropos						
MaSE						
MESSAGE						

KUVIO 29. Menetelmien prosessien kattavuudet ohjelmistotuotannon vaiheiden näkökulmasta

Gaia kattaa ohjelmistotuotannon vaiheista analyysin ja suunnittelun. Suunnitteluvaihe jakautuu arkkitehtuurisuunnitteluun ja yksityiskohtaiseen suunnitteluun. Zambonelli ym. (2003) toteavat, että vaatimusmäärittely oletetaan tehdyksi jo aikaisemmin eikä siihen oteta menetelmässä kantaa. Lisäksi he mainitsevat, että menetelmään voitaisiin helposti lisätä tavoitesuuntautunutta lähestymistapaa (Mylopoulos ym. 1999; Castro ym. 2002) noudattava vaatimusmäärittelyvaihe. Myös järjestelmän toteutukseen liittyvät seikat on rajattu menetelmän ulkopuolelle. Zambonellin ym. (2003) mukaan prosessin tuotoksena on yksityiskohtainen, mutta teknologianeutraali spesifikaatio, joka pitäisi pystyä toteuttamaan helposti käyttämällä tarkoituksenmukaista agenttiohjelmointikehystä tai jotakin modernia olio/komponenttipohjaista kehystä (sellaista, joka tukee hajautettuja ja rinnakkaisia olioita). Sen sijaan analyysi- ja suunnitteluvaiheet on määritelty ja rajattu tarkasti ja selkeästi. Analyysivaiheessa keskitytään järjestelmän mallintamiseen organisaationa ja sen jäsenten (roolien) mallintamiseen. Lisäksi mallinnetaan alustavasti roolien välinen vuorovaikutus. Analyysivaiheessa pyritään siis mallintaman järjestelmän toiminta kokonaisuutena eikä mennä kovin tarkasti roolien (ja sitä myötä agenttien) yksityiskohtien suunnitteluun. Arkkitehtuurisuunnittelun vaiheessa täydennetään alustavat kuvaukset ja määritellään organisaatiolle rakenne (eli järjestelmälle arkkitehtuuri). Tässä vaiheessa aletaan myös tutkia agenttien toiminnallisuutta. Yksityiskohtaisen suunnittelun vaiheessa mallinnetaan agenttien tarjoamat palvelut ja määritellään agenteille roolit, eli mikä agentti esittää mitäkin roolia.

Tropos on tässä tutkielmassa vertailtavista menetelmistä kattavin. Menetelmä kattaa ohjelmistotuotannon vaiheet vaatimusmäärittelystä toteutukseen. Menetelmän päähuomio keskittyy vaatimusmäärittelyvaiheeseen, joka on jaettu kahteen osaan: aikaisten ja myöhäisten vaatimusten määrittelyyn. Tässä tutkielmassa määritellyin termein ajateltuna aikaisten vaatimusten määrittely vastaisi vaatimusmäärittelyvaihetta ja myöhäisten vaatimusten määrittely analyysivaihetta. Menetelmän kehittäjät eivät käytä analyysivaiheen nimitystä menetelmässä lainkaan, mutta myöhäisten vaatimusten määrittelyvaiheessa suoritettavat toiminnot vastaavat suunnilleen muissa menetelmissä analyysivaiheessa suoritettavia toimintoja. Suunnitteluvaihe jakautuu arkkitehtuurisuunnitteluun ja yksityiskohtaiseen suunnitteluun. Muista vertailtavista menetelmistä poiketen Tropos kattaa myös järjestelmän toteutuksen. Testaukseen ja ylläpitoon ei oteta kantaa lainkaan. Troposissa agenttien rakenne ja toiminnallisuus määritellään vasta yksityiskohtaisen suunnittelun vaiheessa, eli varsin myöhään verrattuna esimerkiksi Gaiaan, jossa agenttien rakenteelliset piirteet määritellään osittain jo analyysivaiheessa. Arkkitehtuurisuunnittelun vaiheeseen asti agentteja käsitellään ”mustina laatikoina”.

MaSE kattaa ohjelmistotuotannon vaiheista analyysin ja suunnittelun. Vaatimusmäärittely ja toteutus rajataan selvästi pois menetelmän laajuudesta. Analyysivaihe on jaettu kolmeen askeleeseen, jotka ovat tavoitteiden määrittely, käyttötapausten kuvaaminen ja roolien tarkentaminen. Suunnitteluvaiheen askeleet ovat agenttiluokkien luominen, keskustelujen konstruointi, agenttiluokkien kokoaminen ja järjestelmäsuunnittelu. Agenttien mallintaminen lähtee liikkeelle agenttien toiminnallisuuden mallintamisesta. Roolien väliset suhteet ja keskustelut määritellään alustavasti jo analyysivaiheessa kun taas agenttien rakenne määritellään vasta myöhemmin suunnitteluvaiheessa. Raja analyysi- ja suunnitteluvaiheen välillä on selkeä. Analyysivaiheessa käytetään roolin käsitettä ja suunnitteluvaiheessa käytetään johdonmukaisesti agentin käsitettä. Vaiheiden ja askeleiden välillä voidaan liikkua iteratiivisesti.

Myös MESSAGE kattaa ohjelmistotuotannon vaiheista analyysin ja suunnittelun. Evans ym. (2001) toteavat, että vaatimusten määrittelyn tulisi sisältyä menetelmän analyysivaiheeseen. MESSAGE:ssa vaatimusmäärittely ei toteudu ainakaan tämän

tutkielman määritelmän mukaisesti. Analyysivaiheen mallintamisessa voidaan syötteinä käyttää esimerkiksi organisaatiokaavioita, yrityksen tavoitteiden kuvauksia ja liiketoimintaprosessien kuvauksia. Periaatteessa MESSAGE:n 0-tason kaavioita voidaan pitää osittaisena vaatimusmäärittelynä. Suunnitteluvaiheessa mallinnus on melko yksityiskohtaista ja MESSAGE ylettyykin lähemmäs toteutustasoa kuin Gaia ja MaSE. MESSAGE tutkii aluksi agenttien toiminnallisuutta ja rakenteellisuutta samanaikaisesti. Tämä tapahtuu osittain jo analyysivaiheessa agentti/roolinäkymässä. Samoin suunnitteluvaiheessa agenttien rakenteellisten ja toiminnallisten piirteiden tutkiminen tapahtuu jokseenkin samanaikaisesti. MESSAGE:n suunnitteluvaiheen puutteeksi voidaan todeta, että se ei ole niin hyvin organisoitu kuin muissa menetelmissä.

Kaikki menetelmät kattavat selkeästi ohjelmistotuotannon vaiheista analyysin ja suunnittelun. Lisäksi Tropos kattaa vaatimusmäärittelyn ja toteutuksen. Myös MESSAGE:n analyysivaiheen 0-tason mallinnuksessa on vaatimusmäärittelylle tyypillisiä piirteitä. Analyysi- ja suunnitteluvaiheiden rajat tuodaan menetelmissä selvästi esille siten, että analyysivaiheen mallinnus tapahtuu selvästi yleisemmällä tasolla kuin suunnitteluvaiheen mallinnus. Lisäksi analyysivaiheessa käytetään pääasiassa roolin (Troposissa aktorin) käsitettä, kun taas suunnitteluvaiheessa käytetään agentin käsitettä. Testaukseen ja ylläpitoon ei menetelmissä anneta ohjeistusta lainkaan. Menetelmien käyttö ja huolellinen dokumentointi parantavat todennäköisesti ylläpidettävyyttä, mutta sitä ei tässä yhteydessä katsota riittäväksi edellytykseksi, jotta voitaisiin sanoa menetelmän huomioivan myös ylläpidon. Minkään menetelmän osalta ei voida sanoa, että se kattaisi jonkin vaiheen selkeämmin kuin jokin toinen menetelmä. Kaikissa menetelmissä analyysi- ja suunnitteluvaiheet ovat painotuksiltaan samankaltaisia. Tropos kattaa lisäksi vaatimusmäärittelyn ja toteutuksen. Niiltä osin menetelmää ei voi verrata muihin, koska muissa menetelmissä ei kyseisiä vaiheita ole.

5.3.2 Tuotokset

Seuraavaksi vertaillaan menetelmien prosessien tuotoksia. Tuotoksilla tarkoitetaan dokumentaatiota, jota prosessin eri vaiheista saadaan. Tässä yhteydessä arvioidaan myös, tarjoaako prosessi tukea tuotosten verifiointiin ja validointiin ja huomioidaanko prosessissa laatunäkökohtia.

Gaian prosessin tuotoksena saadaan seuraava dokumentaatio: ympäristömalli, organisatoriset säännöt, vuorovaikutusmalli, roolimalli, järjestelmän organisaatiorakenne (arkkitehtuuri), agenttimalli ja palvelumalli. Tuotosten verifiointia ja validointia ei käsitellä menetelmässä millään lailla. Myöskään laatunäkökohtia ei huomioida.

Troposin prosessin tuotoksena saadaan seuraava dokumentaatio: aktorimalli, strateginen perustelumalli, laajennettu aktorimalli, järjestelmän strateginen perustelumalli, arkkitehtuurimalli, kommunikaatiomalli, käyttäytymismalli ja BDI-malli. Lopullisena tuotoksena dokumentaation lisäksi toteutuksen jälkeen on valmis järjestelmä. Tuotosten verifiointiin ja validointiin ei kiinnitetä erikseen huomiota. Sama pätee myös laatunäkökohtiin.

MaSE:n prosessin tuotoksena saadaan seuraava dokumentaatio: tavoitehierarkia, käyttötapausmäärittelyt sekvenssikaavioineen, roolimalli ja rinnakkaisten tehtävien määrittely, agenttiluokkien määrittely, keskusteluluokkakaaviot, agenttiarkkitehtuurimäärittely sekä sijoittelukaaviot. MaSE:n käytön tueksi on kehitetty agentTool-niminen työkalu (DeLoach 2001). Työkalulla voidaan mallintaa kaikki MaSE:n vaiheet. Lisäksi se tarjoaa tuen agenttien välisten keskustelujen verifiointiin. Tämä on hyvä ominaisuus, koska on tärkeää varmistaa, ettei agenttien välisissä keskusteluissa tule lukkiutumia (deadlocks). Tuotosten validointiin ja laatunäkökohtiin menetelmässä ei kiinnitetä huomiota.

MESSAGE:n analyysivaiheen tuotoksina saadaan organisaationäkymä, tavoite/tehtävänäkymä, agentti/roolinäkymä, vuorovaikutusnäkymä ja kohdealuenäkymä. Jokaisesta kaaviosta saattaa olla useita versioita (0-taso. 1-taso jne.). Suunnitteluvaiheen tuotoksena saadaan joukko dokumentteja, jotka syntyvä analyysivaiheen mallien tarkentamisen tai transformoinnin seurauksena. Myöskään MESSAGE:ssa ei tuotosten verifiointia tai validointia eksplisiittisesti huomioida. Sama pätee laatunäkökohtiin.

Yhteenvetona menetelmien tuotoksista voidaan sanoa, että kaikista menetelmistä saadaan melko paljon dokumentaatiota tukemaan järjestelmän kehittämistä. Ainoa menetelmä, jossa tuotosten osittainen verifiointi on mahdollista, on tällä hetkellä MaSE,

joka tarjoaa siihen työkalutuen. Tuotosten validointia tai johdonmukaisuustarkistuksia ei huomioida missään menetelmissä. Myöskään tuotosten laatunäkökohtia ei huomioida menetelmissä eksplisiittisesti. Toisaalta voidaan kysyä, onko tuotosten validointi ja laatunäkökohdat välttämättä huomioitava tietojärjestelmän kehittämismenetelmässä? Kyseisiä toimintoja vartenhan on olemassa erilaisia katselmointikäytäntöjä ja kypsyysmalleja, kuten esimerkiksi CMM (Capability Maturity Model, SEI Home Page 2004).

5.3.3 Johtamisen tuki

Mikään vertailun menetelmistä ei eksplisiittisesti huomioi johtamista. Esimerkiksi projektin suunnitteluun, seurantaan ja johtamiseen ei anneta minkäänlaisia ohjeita. Toisaalta voidaan ajatella niin, että käytettäisiin menetelmien vaiheistuksia ainakin osittain myös projektin vaiheistuksena. Tämä voisi onnistua kaikkien muiden menetelmien, paitsi MESSAGEN osalta. MESSAGE:n vaiheita ei ole samalla tavalla jaettu pienempiin askeleisiin kuin muissa menetelmissä, joten menetelmän vaiheiden käyttäminen projektin vaiheistuksessa voi muodostua ongelmalliseksi.

5.4 Käytäntö

Tässä kohdassa menetelmiä arvioidaan ja vertaillaan viitekehyksessä esitettyjen käytäntöön liittyvien kriteerien pohjalta. Kriteerit ovat kieli-, paradigma- ja arkkitehtuuriyhteensopivuus sekä kohdealuesoveltuvuus. Edellinen on tärkeä sen varmistamiseksi, että menetelmä pitäytyy organisaation infrastruktuurissa ja tietämyksessä. Jälkimmäinen on tärkeää varmistaa siksi, että menetelmä todella soveltuu kehitettävän järjestelmän kohdealueelle.

5.4.1 Kieli-, paradigma- ja arkkitehtuuriyhteensopivuus

Gaia ei pohjaudu minkään tietyn arkkitehtuurin käsitteille. Esimerkiksi BDI-mallin käsitteitä Gaia ei huomioi lainkaan. Myöskään minkään ohjelmointikielen osalta ei anneta suosituksia. Vaikka menetelmä ei otakaan kantaa BDI-mallin käsitteisiin, ei tämä kuitenkaan estä toteuttamasta BDI-arkkitehtuurin mukaisia älykkäitä järjestelmiä. Tropos pohjautuu BDI-mallin käsitteille. Järjestelmän kehityksen alkuvaiheissa käytetään mallista hieman poikkeavia käsitteitä (ks. TAULUKKO 9). Toteutusvaiheessa

käsitteet muunnetaan vastaamaan BDI-mallin käsitteitä. Toteutus suoritetaan käyttämällä Java-pohjaista JACK Intelligent Agents -kehitysympäristöä.

MaSE on tarkoitettu kaikenlaisten järjestelmien suunnitteluun. Se ei pohjaudu minkään tietyn arkkitehtuurin käsitteille, vaan se antaa vapauden valita tarkoituksenmukaisen arkkitehtuurin toteutusta varten. Myöskään MESSAGE ei pohjaudu minkään tietyn arkkitehtuurin käsitteille, vaan antaa suunnittelijalle vapaat kädet muodostaa oma arkkitehtuurinsa tai valita olemassa olevista agenttiarkkitehtuureista tilanteeseen sopivin.

5.4.2 Kohdealuesoveltuvuus

Kaikki vertailtavat menetelmät on lähtökohtaisesti tarkoitettu uusien järjestelmien suunnitteluun. Gaian osalta Zambonelli ym. (2003) mainitsevat, että menetelmän nykyversio soveltuu hyvin keskikokoisten ja suurten järjestelmien suunnitteluun. Gaia on tarkoitettu yleisesti sovellettavaksi eri kohdealueille. Menetelmä ei siis rajoitu millekään tietylle kohdealueelle, kuten esimerkiksi internet-pohjaisiin järjestelmiin. Kuten aiemmin jo mainittiin, menetelmä pyrkii olemaan mahdollisimman riippumaton mistään tietystä teknologiasta. Tällä pyritään siihen, että menetelmää voitaisiin käyttää mahdollisimman laajasti useilla kohdealueilla.

Myös Tropos sopii parhaiten uusien järjestelmien suunnitteluun. Se pohjautuu vahvasti vaatimusten määrittelyn käsitteille koko kehitysprosessin ajan. Tarkoituksena on pyrkiä ymmärtämään kysymysten ”mitä” ja ”miten” lisäksi myös, miksi tietty järjestelmä kehitetään (Bresciani ym. 2004). Tropos soveltuu BDI-malliin pohjautumisestaan huolimatta monenkaltaisten järjestelmien suunnitteluun. Joitakin rajoituksia kuitenkin on. Parhaiten Tropos soveltuu organisatoriseen ympäristöön sijoittuvien järjestelmien suunnitteluun, koska varsinkin vaatimusmäärittelyvaiheessa organisatoristen toimijoiden (sosiaalisten aktoreiden) merkitys on suuri. Näin ollen menetelmä soveltuu huonommin esimerkiksi sulautettujen järjestelmien suunnitteluun.

Myös MaSE on tarkoitettu yleiskäyttöiseksi ja sovellettavaksi monille sovellusalueille. MaSE on tarkoitettu ensisijaisesti uusien järjestelmien suunnitteluun. Menetelmässä ei mallinneta järjestelmän käyttäjiä sosiaalisina aktoreina, kuten Troposissa tehdään.

Käyttäjä nähdään yksinkertaisesti ulkoisena resurssina aivan kuten tietokannat ja ulkoiset järjestelmätkin. Tästä syystä menetelmä soveltuu hyvin myös sulautettujen järjestelmien suunnitteluun. Menetelmä soveltuu myös olemassa olevien ohjelmistojen uudelleensuunnitteluun.

Evans ym. (2001) mainitsevat, että MESSAGE kehitettiin vastaamaan erityisesti tietoliikennealan tarpeisiin. Tästä huolimatta menetelmä sopii yleisesti käytettäväksi ja sen käyttöönoton kynnystä saattaa madaltaa se, että menetelmä pohjautuu vahvasti UML:ään, joka on laajasti käytössä organisaatioissa. MESSAGE pohjautuu vahvasti oliopohjaisille käsitteille ja UML:n käyttöön, joten uusien järjestelmien kehittämisen lisäksi kynnys olemassa olevien oliopohjaisten järjestelmien uudelleensuunnitteluun ei ole kovin suuri.

Protoiluun menetelmät eivät sovellu kovin hyvin, koska MaSE:a lukuun ottamatta työkalutukea menetelmille ei ainakaan toistaiseksi ole tarjolla. Menetelmissä ei myöskään huomioida käyttöliittymän suunnittelua, mikä on olennainen osa protoilua ainakin sellaisissa järjestelmissä, joiden käyttöön tarvitaan käyttöliittymää. Sulautetuissa järjestelmissä tätä ongelmaa ei ole.

5.5 Yhteenveto

Tässä luvussa arvioitiin ja vertailtiin luvussa 3 esiteltyjä menetelmiä edellisessä luvussa muodostetun viitekehyksen avulla. Aluksi vertailtiin menetelmien käsitteitä ja ominaisuuksia, jonka jälkeen muodostettiin menetelmien käsiterakenteista UML-kaaviot ja vertailtiin niitä. Kolmannessa kohdassa tutkittiin menetelmien malleja. Aluksi tutkittiin menetelmien keskeisten käsitteiden sijoittumista eri malleihin graafisten esitysten avulla. Tämän jälkeen tutkittiin mallien käyttämiä notaatioita, mallien monimutkaisuuden hallintaa, ilmaisuvoimaa sekä modulaarisuutta. Neljännessä kohdassa tutkittiin menetelmien prosesseja. Prosesseista selvitettiin, mitä järjestelmänkehityksen elinkaaren vaiheita ne kattavat. Lisäksi tutkittiin mitä samannimiset vaiheet tarkoittavat eri menetelmissä ja mihin agenttien piirteisiin (rakenteellisuus, toiminnallisuus) eri vaiheissa keskitytään. Prosessien osalta tutkittiin vielä lyhyesti, tukevatko ne johtamista ja lopuksi tutkittiin prosessien tuotoksia. Viimeisessä kohdassa tutkittiin lyhyesti menetelmiin liittyviä käytännön seikkoja.

Käytännön osalta tarkasteltiin kieli-, paradigma- ja arkkitehtuuriyhteensopivuutta sekä menetelmien soveltuvuutta eri kohdealueille. Seuraavassa luvussa esitetään johtopäätökset menetelmien vertailusta sekä siihen käytetystä viitekehyksestä.

6 JOHTOPÄÄTÖKSET

Tässä luvussa esitetään johtopäätökset edellisessä luvussa suoritetusta menetelmien vertailusta sekä vertailun pohjana olleesta viitekehyksestä. Tutkielman päätavoitteena oli kuvata ja vertailla agenttipohjaisia menetelmiä sen selvittämiseksi, mitä yhtäläisyyksiä ja eroja niiden käsitteissä, malleissa ja prosesseissa on. Aluksi esitetään johtopäätökset vertailusta osa-alueittain, jonka jälkeen pohditaan vertailussa käytettyä viitekehystä.

6.1 Johtopäätökset menetelmien vertailusta

Menetelmien vertailun johtopäätökset esitetään elementtikohtaisesti vertailun mukaisessa järjestyksessä. Vertailtavat menetelmät olivat Gaia, Tropos, MaSE ja MESSAGE.

6.1.1 Käsitteet

Menetelmien käsitteiden vertailussa keskityttiin kolmeen seikkaan: peruskäsitteisiin, agenttien ominaisuuksiin ja menetelmien käsiterakenteisiin. Viitekehyksessä määriteltiin 12 peruskäsitettä, jotka menetelmien tulisi huomioida. Määritellyt peruskäsitteet olivat agentti, uskomus, halu, aikomus, organisaatio, yhteisö, sääntö, viesti, protokolla, rooli, palvelu ja tehtävä. Lisäksi määriteltiin agenttien yleiset ominaisuudet (autonomia, reaktiivisuus, proaktiivisuus ja sosiaalisuus) ja tutkittiin, miten ne ilmenevät eri menetelmissä. Lopuksi vertailtiin menetelmien käsiterakenteita. Käsitteistä ja niiden välisistä suhteista muodostettiin UML-notaation mukaiset luokkakaaviot, joista nähtiin menetelmien keskeiset käsitteet ja niiden väliset suhteet.

Peruskäsitteiden vertailun perustella voidaan sanoa, että menetelmät kattavat määritellyt käsitteet varsin hyvin. Myös käsitteiden merkitykset ovat melko samankaltaisia kaikissa menetelmissä eikä menetelmien välillä ole perustavaa laatua olevia eroja. Jonkin verran erilaisuutta esiintyy käsitteiden nimityksissä. Tärkeimmän, eli agentin, käsitteen osalta menetelmien välillä ei eroja juurikaan löytynyt. Agenttien ominaisuuksien osalta havaittiin, että kaikki vertailtavat menetelmät huomioivat määritellyt ominaisuudet hyvin. Tapa, jolla ominaisuudet huomioidaan, vaihtelee jonkin verran menetelmittäin. Gaia erottuu selkeästi muista menetelmistä ominaisuuksien huomioimisessa.

Menetelmässä kaikki ominaisuudet sosiaalisuutta lukuun ottamatta ilmenevät roolimääritysten kautta. Muissa menetelmissä samat ominaisuudet ilmenevät keskenään jokseenkin samalla tavalla. Sosiaalisuus ilmenee kaikissa menetelmissä agenttien välisen vuorovaikutuksen mallintamisessa.

Menetelmien käsiterakenteita vertailtaessa havaittiin, että Troposin käsiterakenne poikkeaa selvästi muiden menetelmien käsiterakenteista. Tämä johtuu pääasiassa siitä, että Tropos pyrkii käyttämään samoja käsitteitä koko kehittämisprosessin ajan. Lisäksi huomattiin, että Troposissa roolin käsite on huomattavasti vähäisemmässä asemassa kuin muissa menetelmissä. Gaia, MaSE ja MESSAGE käyttävät roolin käsitettä siltana analyysi- ja suunnitteluvaiheiden välillä, kun taas Troposissa roolin käsitteellä on huomattavasti vähäisempi asema.

6.1.2 Mallit

Vertailun laajin osuus keskittyi menetelmien malleihin. Malleista tutkittiin ja vertailtiin niiden käsitteellistä sisältöä, notaatiota, monimutkaisuuden hallintaa, ilmaisuvoimaa ja modulaarisuutta.

Käsitteellisten sisältöjen osalta menetelmien erot osoittautuivat selvästi suuremmiksi kuin käsiterakenteiden suhteen. Myös tässä yhteydessä havaittiin, että roolin käsite on keskeinen kaikissa menetelmissä Troposia lukuun ottamatta. Troposissa puolestaan aktorin käsite osoittautui erittäin keskeiseksi, sillä se esiintyy menetelmän kaikissa malleissa. Mallien käsitteellisten sisältöjen perusteella keskeisimmiksi malleiksi näyttäisivät nousevan mallit, joissa määritellään agenttien roolit, ja toisaalta mallit, joissa mallinnetaan agenttien välinen vuorovaikutus. Troposissa roolien mallintamisen sijaan keskeisimpään asemaan nousevat aktoreiden ja niiden välisten riippuvuuksien mallinnus.

Mallien notaatiot eroavat toisistaan paljon. Gaia ei käytä lainkaan määrättyä notaatiota. Zambonelli ym. (2003) mainitsevat notaation kiinnittämisen olevan nykytilanteessa ennenaikaista. Tässä tutkielmassa esitetyt notaatiot ovat esimerkkejä Zambonellin ym. (2003) artikkelista, joten on mahdollista, että mallit näyttävät tulevaisuudessa hyvinkin erilaisilta. Tämä on jopa todennäköistä, sillä Zambonellin ym. (2003) mukaan

esimerkiksi AUML:n käyttö Gaian yhteydessä tulee todennäköisesti jatkossa lisääntymään. Notaatioiden käytöltään kaikista selkein on Tropos. Menetelmässä käytetään johdonmukaisesti Yun (1995) i*-mallin notaatiota yksityiskohtaisen suunnittelun vaiheeseen asti. Notaatio on lisäksi melko yksinkertainen oppia, mikä helpottaa menetelmän omaksumista. Yksityiskohtaisen suunnittelun vaiheessa käytetään UML:n ja AUML:n mukaisia notaatioita. Notaatioiltaan monimuotoisin on MaSE. Lähes kaikissa MaSE:n malleissa on käytössä erilainen notaatio. Lisäksi notaatio on UML:n tilakaavioita ja sekvenssikaavioita lukuun ottamatta menetelmän kehittäjien itsensä määrittelemä, mikä saattaa vaikeuttaa menetelmän omaksumista. Vaikka notaatiot itsessään ovat melko yksinkertaisia, niiden suuri lukumäärä vaikeuttaa niiden muistamista. Vastakkainen esimerkki puolestaan on MESSAGE, jossa pitäydytään lähes yksinomaan UML:n mukaisen notaation käytössä. Tosin analyysivaiheessa käytetään menetelmän kehittäjien UML-notaation laajennusta, joka voi aluksi olla hankala oppia, mutta suunnitteluvaiheessa pitäydytään perinteisen UML:n mukaisessa notaatiossa.

Monimutkaisuuden hallinta toteutuu kaikissa menetelmissä pääpiirteissään hyvin. Menetelmien mallit noudattelevat melko hyvin menetelmien vaihejakoa ja mallien abstraktiotaso madaltuu mitä pidemmälle mallinnusprosessissa edetään. Tosin MESSAGE:n osalta voitiin analysoida ainoastaan analyysivaiheen mallinnusta, koska suunnitteluvaiheeseen ei esitetty selkeitä malleja, vaan Evans ym. (2001) ohjeistavat tarkentamaan tai transformoimaan analyysivaiheen malleja käyttäen perinteisiä ohjelmistotuotannon tekniikoita. MESSAGE:n suunnitteluvaiheen mallinnus vaikuttikin lievästi sekavalta, mistä johtuen monimutkaisuutta ei pystytä hallitsemaan niin hyvin kuin muissa menetelmissä.

Menetelmien ilmaisuvoimaa arvioitiin kahdeksan viitekehyksessä määritellyn kohteen avulla. Arvioitavat kohteet olivat järjestelmän rakenne, järjestelmän tietovirrat, järjestelmän ja agenttien sisäiset rinnakkaiset toiminnot, järjestelmän resurssirajoitteet, järjestelmän fyysinen arkkitehtuuri, agenttien liikkuvuus, järjestelmän vuorovaikutus ulkoisten järjestelmien kanssa sekä käyttöliittymämäärittelyt. Vertailusta nousi esille mielenkiintoisia seikkoja. Mikään menetelmistä ei mallinna agenttien liikkuvuutta. Tämä on seikka, johon tulevaisuudessa on kiinnitettävä huomiota, sillä liikkuvista agenteista on viimeaikaisessa kirjallisuudessa kirjoitettu paljon ja näyttäisi siltä, että

liikkuviin agentteihin perustuvien sovellusten lukumäärä tulee tulevaisuudessa lisääntymään muun muassa mobiiliteknologian kehittymisen myötä. Toinen huomiota herättävä seikka on käyttöliittymämäärittelyiden puutteellinen huomiointi. Gaiassa ja MESSAGE:ssa asiaan ei oteta lainkaan kantaa. Tropos ja MaSE huomioivat käyttöliittymän mainitsemalla, että käyttöliittymä mallinnetaan omana aktorina (Tropos) tai roolina (MaSE). Muiden kohteiden osalta menetelmien ilmaisuvoima toteutuu hyvin. Järjestelmän rakenne esitetään selkeästi kaikissa menetelmissä. Myös järjestelmän tietovirrat esitetään hyvin kaikissa muissa menetelmissä paitsi Gaiassa. MaSE ja Tropos tarjoavat hyvät mallit myös agenttien sisäisten rinnakkaisten toimintojen mallintamiseen. Ilmaisuvoimaltaan heikoimmaksi jää Gaia. Muut menetelmät puolestaan ovat mallien ilmaisuvoimaltaan melko tasavahvoja.

Lopuksi mallien osalta arvioitiin niiden modulaarisuutta. Gaia, Tropos ja MaSE noudattelevat modulaarisuusperiaatetta hyvin koko mallinnusprosessin ajan. MESSAGE:n osalta puolestaan havaittiin, että modulaarisuusperiaate toteutuu analyysivaiheen mallinnuksessa huomattavasti paremmin kuin suunnitteluvaiheessa.

6.1.3 Prosessi

Menetelmien mukaisia prosesseja arvioitiin ja vertailtiin kolmen kriteerin perusteella. Vertailun kohteena olivat prosessin elinkaaren kattavuus, prosessin tuotokset sekä prosessin tuki projektin johtamiselle.

Prosessien kattavuutta arvioitiin ohjelmistotuotannon vaiheiden näkökulmasta. Viitekehyksessä määriteltiin prosessille seuraavat vaiheet: vaatimusmäärittely, analyysi, suunnittelu, toteutus, testaus ja ylläpito. Kaikki menetelmät kattavat selkeästi ohjelmistotuotannon vaiheista analyysin ja suunnittelun. Lisäksi Tropos kattaa vaatimusmäärittelyn ja toteutuksen. Myös MESSAGE:n analyysivaiheen mallinnuksessa on joitakin vaatimusmäärittelylle tyypillisiä piirteitä, mutta ne katsottiin tässä yhteydessä analyysivaiheeseen kuuluviksi. Analyysi- ja suunnitteluvaiheiden rajat tuodaan menetelmissä selvästi esille siten, että analyysivaiheen mallinnus tapahtuu selvästi yleisemmällä tasolla kuin suunnitteluvaiheen mallinnus. Lisäksi menetelmien kehittäjät eksplisiittisesti ilmaisevat eri vaiheiden rajat. Testaukseen ja ylläpitoon ei menetelmissä anneta ohjeistusta lainkaan. Samojen vaiheiden välillä ei eri menetelmissä

ole merkittäviä eroja. Suoritetun vertailun perusteella ei voida sanoa, että jokin menetelmä kattaisi jonkin vaiheen paremmin kuin jokin toinen. Troposin osalta vertailukohtaa vaatimusmäärittely- ja toteutusvaiheille ei luonnollisestikaan ollut.

Menetelmien prosessien tuotoksia vertailtaessa havaittiin, että menetelmien käytöllä saadaan melko paljon dokumentaatiota tukemaan järjestelmän kehittämistä. Tuotosten verifiointi on tällä hetkellä mahdollista ainoastaan MaSE:ssa, joka tarjoaa myös työkalutuen menetelmän eri vaiheisiin. Tuotosten validointia tai yhdenmukaisuustarkistuksia ei huomioida missään menetelmässä. Sama pätee laatunäkökohtiin. Tässä yhteydessä voidaan kysyä, tarvitseeko validointia ja laatunäkökohtia eksplisiittisesti huomioida tietojärjestelmän kehittämismenetelmässä. Menetelmän asianmukainen käyttö kuitenkin jo itsessään mahdollistaa laadukkaiden tuotteiden valmistamisen. Tuotosten validointiin on olemassa erilaisia menetelmistä irrallisia katselmointikäytäntöjä. Laadunvarmistukseen puolestaan on olemassa erilaisia prosesseja ja kypsyysmalleja, kuten esimerkiksi CMM (SEI Home Page 2004). Toisaalta voidaan kuitenkin myös ajatella, että jokaisessa menetelmässä tulisi määritellä täsmällisesti kriteerit, joiden mukaan järjestelmän kehittämisen tulisi edetä. Näin ollen järjestelmän kehittämisen voisi katsoa olevan laadukasta ainoastaan silloin, kun noudatetaan tarkasti näitä kriteerejä.

Menetelmien tarjoamaa tukea johtamiselle arvioitiin projektin suunnittelun ja projektin johtamisen näkökulmasta. Mikään vertailun menetelmistä ei eksplisiittisesti huomioi johtamista. Toisaalta voidaan ajatella, että menetelmien prosessien vaihejakoa käytettäisiin ainakin osittaisena projektin vaiheistuksena. MESSAGE:n kohdalta tämäkin saattaisi muodostua ongelmalliseksi, koska sen vaiheita ei ole samalla tavalla jaettu pienempiin askeleisiin kuin muissa menetelmissä.

6.1.4 Prosessien ja mallien integroiva tarkastelu

Seuraavaksi tarkastellaan tiivistetysti menetelmien mallinnusprosessien ja mallien käsitteellisten sisältöjen suhdetta. Luvussa 3 menetelmien esittelyn yhteydessä kuvattiin se, mitä malleja missäkin järjestelmän kehityksen vaiheessa syntyy. Luvussa 5 menetelmien vertailun yhteydessä puolestaan esitettiin mallien käsitteelliset sisällöt (Kuviot 25-28). Näiden kuvioiden perusteella voidaan tehdä joitakin johtopäätöksiä

siitä, mitkä käsitteet ovat eri menetelmissä keskeisiä järjestelmän kehityksen eri vaiheissa ja mihin seikkoihin mallintamisen eri vaiheissa keskitytään.

Gaiassa käsitteellinen ero analyysi- ja suunnitteluvaiheiden välillä on selvä. Analyysivaiheessa keskitytään järjestelmän ympäristön ja organisatoristen sääntöjen muodostamiseen. Lisäksi mallinnetaan alustavasti roolit, joista organisaatio koostuu sekä niiden välinen vuorovaikutus. Suunnitteluvaiheessa järjestetään roolit organisaatioksi sekä täydennetään rooli- ja vuorovaikutusmallit. Lisäksi määritellään agenttimalli ja palvelumalli. Roolin käsite toimii selkeästi eräänlaisena siltana analyysi- ja suunnitteluvaiheiden välillä. Roolin käsite on selvästi hallitseva analyysivaiheessa ja agentin käsite tulee mukaan vasta suunnitteluvaiheessa. Gaiassa keskitytään aluksi järjestelmän kokonaisuuden mallintamiseen ja vasta sen jälkeen agenttien rakenteellisiin ja toiminnallisiin piirteisiin.

Troposissa puolestaan ei ole näin selkeää käsitteellistä jakoa eri vaiheiden välillä. Keskeisin käsite on aktori, joka käytännössä kulkee mukana koko mallinnusprosessin ajan. Muutkin vaatimusmäärittelyn aikaiset käsitteet esiintyvät myös suunnitteluvaiheessa. Tämä on tietoinen ratkaisu, kuten Castro ym. (2002) ja Bresciani ym. (2004) mainitsevat. Troposissa lähdetään liikkeelle asiakkaan vaatimuksista, josta edetään järjestelmän toimintaympäristön mallintamisen kautta järjestelmän rakenteen mallintamiseen. Vasta aivan suunnittelun loppuvaiheissa aletaan suunnitella yksittäisten agenttien rakennetta. Siihen asti aktoreita/agentteja käsitellään ”mustina laatikoina”. Tässä on selkeä ero esimerkiksi Gaiaan, jossa agenttien rakennetta suunnitellaan osittain jo roolimääritysten yhteydessä.

MaSE:ssa on Gaian tapaan hyvin selkeä ero analyysi- ja suunnitteluvaiheiden käsitteiden välillä. Myös MaSE:ssa roolin käsite toimii siltana analyysi- ja suunnitteluvaiheiden välillä. MaSE:ssa mallintaminen lähtee liikkeelle sekä järjestelmän että yksittäisten roolien tavoitteiden määrittelystä. Roolit ja niiden välinen vuorovaikutus ilmenee osittain jo käyttötapausten ja sekvenssikaavioiden määrittelyssä. Tarkemmin vuorovaikutusprotokollat määritellään roolien mallintamisen jälkeen. Agenttien rakenteelliset piirteet mallinnetaan selkeästi vasta suunnitteluvaiheessa.

Myös MESSAGE:ssa roolin käsitteen merkitys on samankaltainen kuin Gaiassa ja MaSE:ssa, eli toimia siltana analyysin ja suunnittelun välillä. MESSAGE:n käsitteistä suurin osa esiintyy jo analyysivaiheen mallinnuksessa (ks. KUVIO 28). Agentin käsite jää selkeästi suunnitteluvaiheeseen. Kuviossa 28 on esitetty ainoastaan MESSAGE:n analyysivaiheen mallinnus, joten aivan yhtä selkeää kuvaa menetelmän suunnitteluvaiheesta ei ole saatavilla kuin muissa menetelmissä. Tämä johtuu yksinkertaisesti siitä, että menetelmän kehittäjät eivät esitä selkeitä malleja suunnitteluvaiheeseen, vaan ohjeistavat käyttämään perinteisiä ohjelmistotuotannon tekniikoita analyysivaiheen näkymien tarkentamiseen.

6.1.5 Käytäntö

Lopuksi arvioitiin ja vertailtiin lyhyesti menetelmien tarjoamaa tukea käytännön asioihin. Arvioitavat kriteerit olivat kieli-, paradigma- ja arkkitehtuuriyhteensopivuus sekä kohdealuesoveltuvuus.

Vertailun menetelmistä ainoastaan Tropos pohjautuu selkeästi BDI-mallin käsitteille. Tästä huolimatta menetelmä soveltuu hyvin kaikenlaisten järjestelmien suunnitteluun. Gaia, MaSE ja MESSAGE eivät pohjaudu minkään tietyn arkkitehtuurin käsitteille, vaan antavat suunnittelijalle vapauden valita arkkitehtuurin tilanteen mukaan. Troposissa toteutusvaihe suoritetaan käyttämällä Java-pohjaista JACK Intelligent Agents -kehitysympäristöä.

Kaikki vertailun menetelmät on lähtökohtaisesti tarkoitettu uusien järjestelmien suunnitteluun. Menetelmät on tarkoitettu yleisesti sovellettaviksi eli ne eivät rajoitu millekään tietylle kohdealueelle, kuten internet-pohjaisiin järjestelmiin tai elektronisen liiketoiminnan järjestelmiin. Menetelmien soveltuvuus useille kohdealueille oli myös yksi vertailuun valittujen menetelmien valintakriteereistä. MESSAGE kehitettiin alun perin vastaamaan tietoliikennealan tarpeisiin, mutta mitään estettä sen soveltamiseen muillekin kohdealueille ei ainakaan vertailussa havaittu. Uusien järjestelmien suunnittelun lisäksi menetelmiä voidaan käyttää myös olemassa olevien ohjelmistojen uudelleensuunnitteluun. Etenkin MESSAGE soveltuu tähän hyvin, koska se pohjautuu vahvasti oliopohjaisille käsitteille ja UML:n käyttöön, mikä voi edesauttaa sen käyttöönottoa organisaatioissa. Olemassa olevien järjestelmien uudelleensuunnittelun

tukea parantaisi se, jos menetelmissä annettaisiin ohjeistusta siihen, miten ohjelmistojen rakenteet voitaisiin muuttaa agenttiperusteisiksi. Protoiluun menetelmät eivät sovellu kovin hyvin, koska MaSE:a lukuun ottamatta työkalutukea ei ainakaan tällä hetkellä ole muihin menetelmiin tarjolla.

6.2 Johtopäätökset menetelmien vertailun viitekehyksestä

Menetelmien vertailussa käytetty viitekehys pohjautuu pääosin Sturmin ja Shehoryn (2004) viitekehykseen. Sen tukena toimivat hyvin myös muut luvussa 4 esitellyt kehykset. Uusina elementteinä tutkielman viitekehyksessä olivat käsiterakenteet, joka esiintyy Tolvasen (1998) viitekehyksessä, sekä mallien käsitteellinen sisältö, joka on ajatukseltaan samankaltainen kuin Iivarin (1994) kehyksessä esitetty malli. Menetelmien prosessien arviointiin tarjosi hyvin tukea Avisonin ja Fitzgeraldin (1995) kehys.

Sturmin ja Shehoryn (2004) esittämä kehys on varsin laaja ja siitä karsittiin joitakin elementtejä pois, koska muuten vertailusta olisi tullut liian pinnallinen. Osa elementeistä oli karsittava pois myös siksi, että niiden käyttäminen vertailussa olisi vaatinut menetelmien soveltamista johonkin esimerkkiin ja tämä puolestaan olisi ollut liian työlästä tutkielman laajuuden puitteissa. Toisaalta viitekehykseen lisättiin kaksi uutta elementtiä: käsiterakenteet ja mallien käsitteellinen sisältö.

Sturmin ja Shehoryn (2004) kehyksessä esitetään prosessin osalta yhtenä arviointikohteena kehittämiskonteksti. Arviointikohde oli alun perin myös tämän tutkielman viitekehyksessä, mutta menetelmien vertailun aikana havaittiin, että kehittämiskontekstin osalta arvioitavat asiat menevät osin päällekkäin käytäntö-elementissä olevan kohdealuesoveltuvuuden kanssa. Näin ollen päädyttiin yhdistämään nämä kaksi arviointikohdetta (kohdealuesoveltuvuus käytäntö-elementissä). Mallien vertailun aikana havaittiin myös, että mallien modulaarisuus ja monimutkaisuuden hallinta sisälsivät jonkin verran samankaltaisuuksia, mutta nämä elementit päätettiin kuitenkin pitää erillään, koska päällekkäisyyksiä ei juuri esiintynyt.

Menetelmien vertailun aikana tehtyjen muokkausten jälkeen viitekehys toimi varsin hyvin tämän tutkielman vertailun pohjana. Väitettä tukee se, että menetelmien vertailu

täytti tutkielmalle asetetut tavoitteet. Viitekehyksestä löytyy myös kehittämiskohteita. Esimerkiksi taulukossa 7 esitettyä peruskäsitteistöä voitaisiin mahdollisesti laajentaa. Viitekehyksessä määritellyt tietojärjestelmien kehittämisen elinkaaren vaiheet eivät suinkaan ole ehdottomia. Vaiheet – vaatimusmäärittely, analyysi, suunnittelu, toteutus, testaus ja ylläpito – määriteltiin tätä tutkielmaa varten, mutta eri lähteissä esitetään erilaisia näkemyksiä siitä, mitkä vaiheet kuuluvat järjestelmän kehityksen elinkaareen. Vertailu voitaisiin suorittaa myös erilaista vaihejakoa käyttäen. Myös käytännön seikkojen arvioinnin kriteereissä olisi kehittämisen varaa, jos viitekehyksestä haluttaisiin käytännönläheisempi. Vertailuun voitaisiin ottaa mukaan esimerkiksi menetelmien käyttöönotosta aiheutuvien kustannusten arviointi.

7 YHTEENVETO

Agenttipohjainen ohjelmistotuotanto edustaa yhtä uusimmista paradigmoista ohjelmistotuotannon alueella. Vaikka kyseessä onkin uusi paradigma, on agentteja sovellettu eri sovellusalueille melko kauan. Sovellusalueita ovat olleet muiden muassa teolliset, kaupalliset, lääketieteelliset ja viihdeteollisuuden sovellukset. Varsin laajasta sovellusaluekirjosta huolimatta reaalimaailmaan sijoittuvia sovelluksia on kuitenkin toteutettu melko vähän. Ennen kuin agenttiparadigma voi saavuttaa laajan hyväksynnän, se tarvitsee asianmukaisia suunnittelumenetelmiä tukemaan järjestelmien kehittämistä. Kirjallisuudessa on esitetty varsin kirjava joukko menetelmiä agenttipohjaiseen järjestelmien kehittämiseen, mutta suurin osa niistä on vielä melko kypsymättömiä, eikä mikään menetelmistä ole saavuttanut laajaa suosiota.

Tutkielman päätavoitteena oli kuvata ja vertailla agenttipohjaisia menetelmiä sen selvittämiseksi, mitä yhtäläisyyksiä ja eroja niiden käsitteissä, malleissa ja prosesseissa on. Lisäksi tavoitteena oli selvittää, mistä agenteissa on kysymys ja millaisiin sovelluksiin niitä on käytetty. Erilaisia menetelmien vertailuja on esitetty kirjallisuudessa satoja. Agenttipohjaisia menetelmiä on kuitenkin vertailtu vasta vähän johtuen paradigman uutuudesta. Tämän tutkielman menetelmien vertailu oli lähtökohdiltaan akateeminen (vrt. Avison & Fitzgerald 1995). Vertailumenetelmänä käytettiin piirreanalyysiä, joka soveltui hyvin tutkielman tarpeisiin. Vertailtavat menetelmät olivat Gaia (Zambonelli ym. 2003), Tropos (Castro ym. 2002; Bresciani ym. 2004), MaSE (DeLoach ym. 2001) ja MESSAGE (Evans ym. 2001). Pääpaino menetelmien vertailussa oli niiden käsitteissä, malleissa ja prosesseissa. Lisäksi menetelmiä vertailtiin joidenkin käytännön tukeen liittyvien seikkojen osalta. Tällä pyrittiin tuomaan vertailuun hieman käytännönläheisyyttä muuten varsin akateemisen näkökulman tueksi. Aikaisemmin kirjallisuudessa esitetyt agenttipohjaisten menetelmien vertailut (esim. Dam & Winikoff 2004; Sturm & Shehory 2004) ovat olleet melko pinnallisia. Tässä tutkielmassa pyrittiin menetelmien syvällisempään vertailuun erityisesti menetelmien käsitteiden, mallien ja prosessien osalta.

Tutkielman keskeisinä tuloksina ovat 28 menetelmää kattava agenttipohjaisten menetelmien kartoitus, viitekehys agenttipohjaisten menetelmien arviointiin ja

vertailuun sekä menetelmien vertailusta tehdyt johtopäätökset. Menetelmien kartoituksia on tehty kirjallisuudessa aikaisemminkin (ks. esim. Wooldridge & Ciancarini 2001), mutta ne eivät ole kattavuudessaan yltäneet tämän tutkielman kartoituksen tasolle. Yleensä ne sisältävät kuvauksen vain muutamasta menetelmästä. Tämän tutkielman kartoituksesta selviää, että agenttipohjaisten menetelmien kehittäminen on alkanut 1990-luvun puolessa välissä. Suurin osa menetelmistä on kuitenkin kehitetty 2000-luvun alussa. Kartoituksesta saa hyvän yleiskuvan agenttipohjaisten menetelmien tämänhetkisestä määrästä ja sovellusalueista. Siten kartoituksen perusteella on mahdollista löytää tietylle sovellusalueelle (esim. tietämysjärjestelmät) kehitettyjä tai yleisesti käytettäväksi tarkoitettuja menetelmiä tarkasteltaviksi, arvioitaviksi tai edelleen kehitettäviksi. Lisäksi kartoituksen tuloksista voidaan päätellä, milloin eri menetelmien kehittäminen on alkanut, ja milloin niistä on viimeksi ilmestynyt uutta materiaalia. Kartoitus sisältää tämän hetken tuoreimmatkin lähteet.

Menetelmien vertailuun on olemassa yleisiä viitekehyksiä ja erityisiin tarkoituksiin, kuten agenttipohjaisten menetelmien vertailuun, tarkoitettuja kehyksiä. Tässä tutkielmassa vertailun viitekehyksen pohjaksi valittiin Sturmin ja Shehoryn (2004) viitekehys. Viitekehystä muokattiin jonkin verran vertailun aikana, jotta se palvelisi paremmin tutkielman tarkoitusta. Kehykseen otettiin mukaan menetelmien käsiterakenteiden ja mallien käsitteellisten sisältöjen vertailu, joita ei esiinny aikaisemmissa agenttipohjaisten menetelmien vertailukehyksissä.

Viitekehystä testattiin analysoimalla ja vertailemalla valittuja menetelmiä. Käsitteitä vertailtaessa havaittiin, että menetelmät huomioivat viitekehyksessä esitetyt peruskäsitteet ja ominaisuudet hyvin. Menetelmien käsiterakenteet poikkeavat toisistaan vain vähän lukuun ottamatta Troposia, jonka käsiterakenne poikkeaa muista menetelmistä enemmän. Troposissa esimerkiksi roolin käsite on huomattavasti vähäisemmässä asemassa kuin muissa menetelmissä. Sen sijaan menetelmien mallien käsitteelliset sisällöt poikkeavat toisistaan paljon. Myös notaatioiltaan menetelmät eroavat toisistaan varsin paljon. Notaatioiltaan monimuotoisin on MaSE, ja yhtenäisimpiä ovat Tropos ja MESSAGE. Menetelmien ilmaisuvoimasta löydettiin sekä vahvuuksia että heikkouksia. Vahvuuksia ovat etenkin järjestelmän rakenteen ja

tietovirtojen mallintaminen. Puutteiksi havaittiin mm. käyttöliittymämäärittelyiden puutteellinen huomiointi ja agenttien liikkuvuuden mallintaminen. Gaia jäi ilmaisuvoimaltaan heikoimmaksi muiden ollessa melko tasavahvoja. Menetelmien prosesseja vertailtaessa voitiin todeta, että kaikki menetelmät kattavat ohjelmistotuotannon vaiheista selkeästi analyysin ja suunnittelun, ja Tropos kattaa lisäksi vaatimusmäärittelyn ja toteutuksen. Sen sijaan mikään menetelmä ei huomioi testausta ja ylläpitoa. Samojen vaiheiden välillä eri menetelmissä ei havaittu suuria eroja. Tuotosten osalta todettiin, että kaikki menetelmät tuottavat melko paljon dokumentaatiota tukemaan järjestelmän kehittämistä. Tuotosten osittainen verifiointi on mahdollista ainoastaan MaSE:ssa. Tuotosten validointia ei menetelmissä huomioida. Viimeisenä kohteena vertailtiin menetelmiä muutamien käytännön seikkojen suhteen. Kaikki vertailun menetelmät on lähtökohtaisesti tarkoitettu uusien järjestelmien suunnitteluun. Tropos pohjautuu BDI-arkkitehtuurin käsitteille, mutta muut menetelmät eivät pohjaudu mihinkään tiettyyn ratkaisuun. Kaikki menetelmät on tarkoitettu yleisesti sovellettaviksi eri kohdealueille.

Menetelmien vertailujen tuloksiin tulee aina suhtautua kriittisesti. Sama pätee myös tämän tutkielman vertailuun. Vertailuja tekevät ihmiset ja näin ollen ne ovat aina jonkin verran subjektiivisia. Soveltamalla menetelmiä johonkin käytännön esimerkkiin tätä subjektiivisuutta voidaan vähentää, mutta laajempien esimerkkien laatiminen jätettiin tämän tutkielman ulkopuolelle. Tehty vertailu eroaa aiemmin tehdyistä agenttipohjaisten menetelmien vertailuista siinä, että se on syvällisempi kuin aikaisemmat vertailut. Esimerkiksi edellä mainittuja käsiterakenteita ja mallien käsitteellisiä sisältöjä ei ole aikaisemmissa tutkimuksissa juurikaan vertailtu. Tutkielmasta odotetaan olevan sekä tutkimuksellista että käytännön hyötyä. Tutkimuksellisena hyötynä tutkielma tarjoaa edellä mainitun menetelmien kartoituksen sekä viitekehyksen menetelmien vertailuun. Kehystä voidaan käyttää muiden olemassa olevien agenttipohjaisten menetelmien sekä uusien kehitettävien menetelmien vertailuun. Tällä tavalla voidaan askel askeleelta rakentaa kattavampaa käsitystä menetelmätuen tarjonnasta, luonteesta ja laadusta. Viitekehystä voivat hyödyntää myös menetelmien kehittäjät. Tutkielmassa on nostettu esiin joitakin seikkoja, joihin tulisi tulevaisuudessa kiinnittää enemmän huomiota (esim. käyttöliittymäsuunnittelu ja agenttien liikkuvuuden mallintaminen). Lisäksi suoritettu vertailu kertoo tutkijoille

vertailtujen menetelmien vahvuudet ja heikkoudet, ja auttaa siten heitä esimerkiksi kohdentamaan tutkimustaan jollekin tietylle ongelma-alueelle. Käytännön hyötynä tutkielma esittää katsauksen agenttipohjaisten menetelmien nykytilasta ja pyrkii näin tukemaan menetelmän valintaa ja soveltamista käytännön tilanteissa.

Agenttipohjaisten menetelmien vertailun osalta jatkotutkimusta on tehtävissä paljon. Seuraavana askeleena voisi olla vertailun laajentaminen koskemaan useampia menetelmiä. Tällä hetkellä muiden menetelmien vertailua hankaloittaa jonkin verran se, että vain harvasta menetelmästä on saatavilla riittävästi materiaalia, jonka perusteella vertailu voitaisiin suorittaa. Lyhyet konferenssijulkaisut soveltuvat huonosti menetelmien vertailuun. Toisena jatkotutkimusaiheena voisi olla syventää tämän tutkielman vertailua. Esimerkiksi menetelmien käsitteiden syvällisempi vertailu soveltuisi omaksi tutkielman aiheekseen. Menetelmien keskeisistä käsitteistä voitaisiin muodostaa metamallit ja tutkia niitä. Kolmantena jatkotutkimusaiheena voisi olla laajentaa tämän tutkielman vertailua koskemaan niitä Sturmin ja Shehoryn (2004) viitekehyksessä esitettyjä elementtejä, jotka jäivät tämän tutkielman ulkopuolelle. Tutkielman vertailu oli lähtökohdiltaan käsitteellisteoreettinen. Käytännönläheisempi case-tyyppinen vertailu soveltamalla menetelmiä johonkin konkreettiseen esimerkkiin olisi myös kelvollinen jatkotutkimusaihe. Kokonaan uuden agenttipohjaisen menetelmän kehittäminen taas olisi väitöskirjatasoisen tutkimuksen aihe.

LÄHDELUETTELO

AOS 2003. JACK Manual, release 4.1. Agent Oriented Software Pty. Ltd. Saatavilla pdf-muodossa <http://www.agent-software.com/shared/demosNdocs/JACKManual.pdf> [viitattu 13.1.2004].

Aridor Y. & Lange B. 1998. Agent Design Patterns: Elements of Agent Application Design. Teoksessa: Sycara K. & Wooldridge M. (toim.), Proceedings of the Second International Conference on Autonomous Agents (AGENTS'98), Minneapolis, USA, May 10-13. ACM Press, New York, USA, 108-115.

AUML Web Site 2004. [online]. Saatavilla www-osoitteessa <http://www.auml.org/> [viitattu 28.1.2004].

Avison D. & Fitzgerald G. 1995. Information Systems Development: Methodologies, Techniques and Tools (2nd ed.). McGraw-Hill Publishing Company, London, U.K.

Bernon C., Gleizes M.-P., Peyruqueou S., & Picard G. 2003. ADELFE: A Methodology for Adaptive Multi-Agent Systems Engineering. Teoksessa: Petta P., Tolksdorf R. & Zambonelli F. (toim.), Engineering Societies in the Agents World III: Third International Workshop, ESAW 2002, Madrid, Spain, September 16-17, 2002. Springer Verlag, Heidelberg, Germany, 156-169.

Boertien N., Steen M. & Jonkers H. 2001. Teoksessa: Krogstie J., Siau K. & Halpin T. (toim.), Proceedings of the Sixth CAiSE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'01), Interlaken, Switzerland, 4-5 June.

Brazier F., Dunin-Keplicz B., Jennings N. & Treur J. 1997. DESIRE: Modelling Multi-Agent Systems in a Compositional Formal Framework. International Journal of Cooperative Information Systems 6(1), 67-94.

Bresciani P., Perini A., Giorgini P., Giunchiglia F. & Mylopoulos J. 2004. Tropos: An Agent-Oriented Software Development Methodology. Journal of Autonomous Agents and Multiagent Systems 8(3), 203-236.

Burmeister B., Haddadi A. & Matylis G. 1997. Application of Multi-Agent Aystems in Traffic and Transportation. IEE Proceedings on Software Engineering 144(1), 51-60.

Caire G., Coulier W., Garijo F., Gomez J., Pavon J., Leal F., Chainho P., Kearney P., Stark J., Evans R. & Massonet P. 2002. Agent Oriented Analysis Using MESSAGE/UML. Teoksessa: Wooldridge M., Ciancarini P. & Weiss G. (toim.), Proceedings of the Second International Worshop on Agent-Oriented Software Engineering (AOSE 2001), Montreal, Canada, May 29, 2001. Springer Verlag, Heidelberg, Germany, 119-135.

Callantine T. 2003. Air Traffic Controller Agents. Teoksessa: Rosenschein J., Wooldridge M., Sandholm T. & Yokoo M. (toim.), Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '03), Melbourne, Australia. ACM Press, New York, USA, 952-953.

Castro J., Kolp M. & Mylopoulos J. 2002. Towards Requirements-Driven Information Systems Engineering: the Tropos Project. Information Systems 27(6), 365-389.

Cohen P. & Levesque H. 1990. Intention is Choice With Commitment. Artificial Intelligence 42(2-3), 213-261.

Coleman D., Arnold P., Bodoff S., Dollin D., Gilchrist H., Hayes F. & Jeremas P. 1994. Object-Oriented Development: The FUSION Method. Prentice-Hall International, Hemel Hampstead, U.K.

Corera J., Laresgoiti I. & Jennings N. 1996. Using Archon, Part 2: Electricity Transportation Management. IEEE Expert 11 (6), 71-79.

Cossentino M. & Potts C. 2002. A CASE Tool Supported Methodology for the Design of Multi-Agent Systems. Teoksessa: Arabnia H. (toim.), Proceedings of the 2002 International Conference on Software Engineering Research and Practice (SERP'02), June 24-27, Las Vegas, USA. CSREA Press, Las Vegas, USA.

Coughlan J. & Macredie R. 2002. Effective Communication in Requirements Elicitation: A Comparison of Methodologies. Requirements Engineering 7(2), 47-60.

Dam K. & Winikoff M. 2004. Comparing Agent-Oriented Methodologies. Teoksessa: Giorgini P., Henderson-Sellers B. & Winikoff M. (toim.), Proceedings of the Fifth International Bi-Confernece on Agent-Oriented Information Systems (AOIS 2003), 14 July 2003, Melbourne, Australia. Springer-Verlag, Heidelberg, Germany, 78-93.

DeLoach S., Wood M. & Sparkman C. 2001. Mutliagent Systems Engineering. International Journal of Software Engineering and Knowledge Engineering 11(3), 231-258.

DeLoach S. & Wood M. 2001. Developing Multiagent Systems with agentTool. Teoksessa: Castelfranchi C. & Lspérance Y. (toim.), Proceedings of the 7th International Workshop on Agent theories, Architectures, and Languages (ATAL 2000), July 7-9, 2000, Boston, USA. Springer Verlag, Heidelberg, Germany, 46-60.

Dikenelli O. & Erdur R. 2003. SABPO: A Standards Based and Pattern Oriented Multi-Agent Development Methodology. Teoksessa: Petta P., Tolksdorf R. & Zambonelli F. (toim.), Engineering Societies in the Agents World III: Third International Workshop, ESAW 2002, Madrid, Spain, September 16-17, 2002. Springer Verlag, Heidelberg, Germany, 213-226.

Elammari M. & Lalonde W. 1999. An Agent-Oriented Methodology: High-Level and Intermediate Models. Teoksessa: Wagner G. & Yu E. (toim.) Proceedings of the First International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS 1999), June 14-15, Heidelberg, Germany.

Evans R., Kearny P., Stark J., Caire G., Garijo F., Gomez J., Pavon J., Leal F., Chainho P. & Massonet P. 2001. MESSAGE: Methodology for Engineering Systems of Software Agents. The European Institute for Research and Strategic Studies in Telecommunications (EURESCOM), Heidelberg, Germany. Saatavilla pdf-

muodossa <http://www.eurescom.de/~pub-deliverables/P900-series/P907/TI1/p907ti1.pdf> [viitattu 22.1.2004]

Ferber J. & Gutknecht O. 1998. A Meta-Model for the Analysis and Design of Organizations in Multi-Agent Systems. Teoksessa: Demazeau Y., Durfee E., Georgeff M & Jennings N. (toim.), Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS98), July 4–7, Paris, France. IEEE Computer Society, 128-135.

Floyd C. 1986. A Comparative Evaluation of System Development Methods. Teoksessa: Olle T., Sol H. & Verrij-Stuart A. (toim.), Information Systems Design Methodologies: Improving the Practice, North-Holland, 19-54.

Franklin S. & Graesser A. 1997. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. Teoksessa: Müller J., Wooldridge M. & Jennings R. (toim.), Proceedings of the Third International Workshop on Agent Tehories, Architectures and Languages, Budabest, Hungary, August 12-13, 1996. Springer Verlag, Berlin, Germany, 21-36.

FIPA 2003. FIPA ACL Message Structure Specification. Foundation for Intelligent Physical Agents [online] Saatavilla www-muodossa <http://www.fipa.org/specs/fipa00061/SC00061G.html> [viitattu 27.11.2003].

Genesereth M. & Ketchpel S. 1994. Software Agents. Communications of the ACM 37(7), 48-53.

Gervais M.-P. 2003. ODAC: An Agent-Oriented Methodology Based on ODP. Journal of Autonomous Agents and Multi-Agent Systems 7(3), 199-228.

Giorgini P., Perini A., Mylopoulos J., Giunchiglia F. & Bresciani P. 2001. Agent-Oriented Software Development: A Case Study. Teoksessa: Sen S., Muller J., Andre E. & Frassen C. (toim.), Proceedings of the 13th International Conference on Software Engineering & Knowledge Engineering (SEKE'01), Buenos Aires, Argentina, June 13-15.

Giunchiglia F., Mylopoulos J. & Perini A. 2001. The Tropos Software Development Methodology: Processes, Models and Diagrams. Technical Report # 0111−20. Istituto Trentino di Cultura, Trento, Italy.

Glaser N. 1997. The CoMoMAS Methodology and Enironment for Multi-Agent System Development. Teoksessa: Zhang C. & Lukose D. (toim.), Multi-Agent Systems: Methodologies and Applications, Second Australian Workshop on Distributed Artificial Intelligence, Cairns, Queensland, Australia, August 27, 1996, Revised Papers, 1-16.

Gómez-Sanz J. & Fuentes R. 2002. Agent Oriented Software Engineering with IGENIAS. Teoksessa: de la Cruz J. & Pavon J. (toim.), Proceedings of the 4th Iberoamerican Workshop on Multi-Agent Systems, November 11-12, Malaga, Spain. University of Malaga, Spain.

Graham I., Henderson-Sellers B. & Younessi H. 1997. The OPEN Process Specification. Addison Wesley/Longman, Reading, London.

Gustavsson R. 1999. Agents with Power. Communications of the ACM 42(3), 41-47.

He M., Jennings N. & Leung H.-F. 2003. On Agent-Mediated Electronic Commerce. IEEE Transactions on Knowledge Engineering and Data Engineering 15(4), 985-1003.

Henderson-Sellers B., Collins G., Due R. & Graham I. 2001. A Qualitative Comparison of Two Processes for Object-Oriented Software Development. Information and Software Technology 43(12), 705-724.

Hewitt C. 1977. Viewing Control Structures as Patterns of Passing Messages. Artificial Intelligence 8(3), 323-364.

Iglesias C., Garijo M. & González J. 1999. A Survey of Agent-Oriented Methodologies. Teoksessa: Müller J., Singh M. & Rao A. (toim.), Proceedings of the 5th International Workshop on Agent Theories, Architectures, and Languages (ATAL '98), Paris, France, July 4-7, 1998. Springer Verlag, Heidelberg, Germany, 317-330.

Iglesias C., Garijo M., Gonzalez J. & Velasco J. 1998. Analysis and Design of Multiagent Systems using MAS-CommonKADS. Teoksessa: Singh M., Rao A. & Wooldridge M. (toim.), Proceedings of the 4th International Workshop on Agent theories, Architectures, and Languages (ATAL 97), Providence, Rhode Island, USA, July 24-26. Springer Verlag, Heidelberg Germany, 313-327.

Iivari J. 1994. Object-Oriented Information Systems Analysis: A Comparison of Six Object-Oriented Analysis Methods. Teoksessa: Verrijn-Stuart A. & Olle T. (toim.), Methods and Associated Tools for the Information Systems Life Cycle, IFIP Transactions A-55. North-Holland, 85-110.

Janson M. & Woo C. 1995. Comparing IS Development Tools and Methods: Using Speech Act Theory. Information & Management 28(1), 1-12.

Jayaratna N. 1994. Understanding and Evaluating Methodologies: NIMSAD – a Systemic Framework. McGraw-Hill Book Company Europe, London, U.K.

Jennings N. 2001. An Agent-Based Approach for Building Complex Software Systems. Communications of the ACM 44 (4), 35-41.

Jennings N., Norman T., Fratin P., O'Brien P. & Odgers P. 2000. Autonomous Agents for Business Process Management. International Journal of Applied Artificial Intelligence 14 (2) 145-189.

Jennings N., Sycara K. & Wooldridge M. 1998. A Roadmap of Agent Research and Development. Journal of Autonomous Agents and Multi-Agent Systems 1(1), 275-306.

Jennings N. & Wooldridge M. 1998. Applications of Intelligent Agents. Teoksessa: Jennings N. & Wooldridge M. (toim.), Agent Technologies: Foundations, Applications, and Markets. Springer Verlag, Heidelberg, Germany, 3-28.

Jennings N. & Wooldridge M. 2001. Agent-Oriented Software Engineering. Teoksessa: Bradshaw J. (toim.), Handbook of Agent Technology. AAAI/MIT Press. Saatavilla myös pdf muodossa <http://www.ecs.soton.ac.uk/~nrj/download-files/agt-handbook.pdf> [viitattu 20.11.2003]

Jo C.-H. 2001. A Seamless Approach to the Agent Development. Teoksessa: Lamont G. (toim.), Proceedings of the 2001 ACM symposium on Applied computing, Las Vegas, USA, March 11-14. ACM Press, New York, USA, 641-647.

Juan T., Pearce A. & Sterling L. 2002. ROADMAP: Extending the Gaia Methodology for Complex Open Systems. Teoksessa: Gini M., Ishida T., Castelfranchi C. & Johnson W. (toim.), Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '02), Bologna, Italy 15-19 July. ACM Press, New York, USA, 3-10.

Kavi K., Aborizka M. & Kung D. 2002. A Framework For Designing, Modeling and Analyzing Agent Based Software Systems. Teoksessa: Goscinski A., Li G.-J., Zhou W. & Chi X. (toim.), Proceedings of the 5th International Conference on Algorithms & Architectures for Parallel Processing, October 23-25, Beijing, China. IEEE Computer Society, 196-201.

Kendall E., Krishna P., Suresh C. & Pathak C. 2000. An Application Framework for Intelligent and Mobile Agents. ACM Computing Surveys 32(1es), Article No. 20.

Kendall E., Malkoun M. & Jiang C. 1995. A Methodology fo Developing Agent Based Systems for Enterprise Integration. Teoksessa: Bernus P. & Nemes L. (toim.), Proceedings of the Ifip Tc5 Working Conference on Models and Methodologies for Enterprise Integration, Queensland, Australia, November. Chapman & Hall.

King J. 1995. Intelligent Agents: Bringing Good Things to Life. AI Expert, February, 17-19.

Kinny D. & Georgeff M. 1997. Modelling and Design of Multi-Agent Systems. Teoksessa: Müller J., Wooldridge M. & Jennings R. (toim.), Proceedings of the Third International Workshop on Agent Tehories, Architectures and Languages (ATAL'97), Budabest, Hungary, August 12-13, 1996. Springer Verlag, Berlin, Germany, 1-20.

Knublauch H. 2002. Extreme Programming of Multi-Agent Systems. Teoksessa: Gini M., Ishida T., Castelfranchi C. & Johnson W. (toim.), Proceedings of the First

International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '02), Bologna, Italy 15-19 July. ACM Press, New York, USA, 704-711.

Kruchten P. 1999. The Rational Unified Process. An Introduction. Addison Wesley/Longman, Reading, London.

Lind J. 2001. Iterative Software Engineering for Multiagent Systems - The MASSIVE Method. Volume 1994 of Lecture Notes in Computer Science, Springer Verlag, Heidelberg, Germany.

Ljungberg M. & Lucas A. 1992. The OASIS Air Traffic Management System. Technical Note 28. Australian Artificial Intelligence Institute, Melbourne, Australia.

Luck M. & d'Inverno M. 2001. A Conceptual Framework for Agent Definition and Development. The Computer Journal 44(1), 1-20.

Luck M., McBurney B. & Preist C. 2002. Agent Technology: Enabling Next Generation Computing. A Roadmap for Agent Based Computing, Agent Link. Saatavilla pdf-muodossa <http://www.agentlink.org/roadmap/download.html> [viitattu 5.12.2003]

Maes P. 1994. Agents that Reduce Work and Information Overload. Communications of the ACM 37(7), 31-40.

McLeod G. & Roeleveld D. 2002. Method Evaluation in Practice: UML/RUP & Inspired Method. Teoksessa: Halpin T., Siau K. & Krogstie J. (toim.), Proceedings of seventh CaiSE/IFIP WG8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'02), Toronto, Canada, 27-28 May, 90-101.

Merriam-Webster 2003. Merriam-Webster Dictionary. Merriam-Webster Inc. [online] Saatavilla www-osoitteessa < http://www.m-w.com/ > [viitattu 5.12.2003].

Moody D. 2003. The Method Evaluation Model: a Theoretical Model for Validating Information Systems Design Methods. Teoksessa: Proceedings of the 11th

European Conference of Information Systems (ECIS 2003), Naples, Italy, 16-21 June.

Moreno A. & Garbay C. 2003. Software Agents in Health Care. Artificial Intelligence in Medicine 27(3), 229-232.

Moulin B. & Brassard M. 1996. A Scenario-Based Design Method and an Environment for the Development of Multiagent Systems. Teoksessa: Zhang C. & Lukose D. (toim.), Proceedings of the First Australian Workshop on Distributed Artificial Intelligence, November 13, 1995, Canberra Australia. Springer Verlag, Heidelberg, Germany, 216-232.

Mouratidis H., Giorgini P. & Manson G. 2003. Modelling Secure Multiagent Systems. Teoksessa: Rosenschein J., Wooldridge M., Sandholm T. & Yokoo M. (toim.), Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '03), July 14-18, Melbourne, Australia. ACM Press, New York, USA, 859-866.

Mylopoulos J., Chung L. & Yu E. 1999. From Object-Oriented to Goal-Oriented Requirements Analysis. Communications of the ACM 42(1), 31-37.

Nwana H. 1996. Software Agents: An Overview. The Knowledge Engineering Review 11(3), 205-244.

Odell J., Parunak H. & Bauer B. 2000. Extending UML for Agents. Teoksessa: Wagner G., Lesperance Y. & Yu E. (toim.), Proceedings of the Agent-Oriented Information Systems Workshop (AOIS 2000) at the 17th National Conference on Artificial Intelligence (AAAI 2000), 30 July, Austin, USA. iCue Publishing, Berlin, Germany, 3-17.

Odell J., Parunak H. & Bauer B. 2001. Representing Agent Interaction Protocols in UML. Teoksessa: Ciancarini P. & Wooldridge M. (toim.), Proceedings of the First International Workshop on Agent-Oriented Software Engineering (AOSE 2000), Limerick, Ireland, June 10, 2000. Springer Verlag, Heidelberg, Germany, 121-140.

Omicini A. 2001. SODA: Societies and Infrastructures in the Analysis and Design of Agent-Based systems. Teoksessa: Ciancarini P. & Wooldrige M. (toim.), Proceedings of the First International Workshop on Agent-Oriented Software Engineering (AOSE 2000), Limerick, Ireland, June 10, 2000. Springer Verlag, Heidelberg, Germany, 185-193.

Padgham L. & Winikoff M. 2003. Prometheus: A Methodology for Developing Intelligent Agents. Teoksessa: Giunchiglia F., Odell J. & Weiss G. (toim.), Proceedings of the Third International Workshop on Agent-Oriented Software Engineering (AOSE 2002) at AAMAS'02, Bologna, Italy, July 15, 2002. Springer Verlag, Heidelberg, Germany, 174-185.

Park S., Sugumaran V. & Lee S. 2001. An Architecture-Centric Approach for Multi-Agent System Development and Application. Teoksessa: Wu K-L. & Datta A. (toim.), Proceedings of the Third IEEE International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS '01). June 21-22, San Jose, USA. IEEE Computer Society.

Päivärinta T. 2002. Comparison of the Genre-Based ISD Approach to 11 Others. Teoksessa: Halpin T., Siau K. & Krogstie J. (toim.), Proceedings of Seventh CaiSE/IFIP WG8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'02), Toronto, Canada, 27-28 May, 109-120.

Rao A. & Georgeff M. 1995. BDI Agents: From Theory to Practice. Teoksessa: Lesser V. & Gasser L. (toim.), Proceedings of the First International Conference on Multiagent Systems (ICMAS-95), June 12-14, San Francisco, USA. MIT Press, Cambridge, USA, 312-319.

Shehory O. & Sturm A. 2001. Evaluation of Modeling Techniques for Agent-Based Systems. Teoksessa: André E., Sen S., Frasson C. & Müller J. (toim.), Proceedings of the Fifth International Conference on Autonomous Agents (AGENTS'01), February 11-13, Montreal, Canada. ACM Press, New York, USA, 624-631.

SEI Home Page 2004. Capability Maturity Model for Software (CMM). Carnegie Mellon Software Engineering Institute. [online] Saatavilla www-osoitteessa <http://www.sei.cmu.edu/cmm/> [viitattu 11.6.2004].

Snoeck M., Wijsen J. & Dedene G. 1996. Formal Specification Techniques in Objet-Oriented Analysis: a Comparative View. Teoksessa: Siau K. & Wand Y. (toim.), Proceedings of the Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'96), Crete, Creece.

Sol H. 1983. A Feature Analysis of Information Aystems Design Methodologies: Methodological Considerations. Teoksessa: Olle T., Sol H. & Tully C. (toim.), Information Systems Design Methodologies: a Feature Analysis. North-Holland, Amsterdam, 1-8.

Spivey J. 1992. The Z Notation: A Reference Manual, Second Edition. Prentice Hall, Hemel Hempstead, UK.

Sturm A., Dori D. & Shehory O. 2003. Single-Model Method for Specifying Multi-Agent Systems. Teoksessa: Rosenschein J., Wooldridge M., Sandholm T. & Yokoo M. (toim.), Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '03), July 14-18, Melbourne, Australia. ACM Press, New York, USA, 121-128.

Sturm A. & Shehory O. 2004. A Framework for Evaluating Agent-Oriented Methodologies. Teoksessa: Giorgini P., Henderson-Sellers B. & Winikoff M. (toim.), Proceedings of the Fifth International Bi-Confernece on Agent-Oriented Information Systems (AOIS 2003), 14 July 2003, Melbourne, Australia. Springer-Verlag, Heidelberg, Germany, 94-109.

Sun Microsystems 2003. JavaBeans. Sun Microsystems Inc. [online]. Saatavilla www-osoitteessa <http://java.sun.com/products/javabeans/> [viitattu 8.12.2003].

Sycara K., Pannu A., Williamson M., Zeng D. & Decker K. 1996. Distributed Intelligent Agents. IEEE Expert 11(6), 36-46.

Tolvanen J.-P. 1998. Incremental Method Engineering with Modeling Tools. University of Jyväskylä, Jyväskylä Studies in Computer Science, Economics and Statistics.

Tran Q.-N., Low G. & Williams M.-A. 2003. A Feature Analysis Framework for Evaluating Multi-agent System Development Methodologies. Teoksessa: Zhong N., Ras Z., Tsumoto S. & Suzuki E. (toim.), Proceedings of the 14th International Symposium of Foundations of Intelligent Systems (ISMIS 2003). October 28-31, Maebashi City, Japan. Springer Verlag, Heidelberg, Germany, 613-617.

Tropos 2003. [online]. Saatavilla www-osoitteessa <http://www.troposproject.org> [viitattu 13.1.2004].

Tveit A. 2001. A Survey of Agent-Oriented Software Engineering. NTNU Computer Science Graduate Student Conference, Norwegian University of Science and technology, Trondheim, Norway.

Wood M. & DeLoach S. 2001. An Overview of the Multiagent Systems Engineering Methodology. Teoksessa: Ciancarini P. & Wooldridge M. (toim.), Proceedings of the First International Workshop on Agent-Oriented Software Engineering (AOSE 2000), Limerick, Ireland, June 10, 2000. Springer Verlag, Heidelberg, Germany, 207-222.

Wooldridge M. 1997. Agent-Based Software Engineering. IEE Proceedings on Software Engineering 144(1), 26-37.

Wooldridge M. & Ciancarini P. 2001. Agent-Oriented Software Engineering: The State of the Art. Teoksessa: Ciancarini P. & Wooldridge M. (toim.), Proceedings of the First International Workshop on Agent-Oriented Software Engineering, Limerick, Ireland, June 10, 2000. Springer Verlag, Heidelberg, Germany, 1-28.

Wooldridge M. & Jennings N. 1995. Intelligent Agents: Theory and Practice. The Knowledge Engineering Review 2(10), 115-152.

Wooldridge M, Jennings N. & Kinny D. 1999. A Methodology for Agent-Oriented Analysis and Design. Teoksessa: Etzioni O., Müller J. & Bradshaw J. (toim.),

Proceedings of the third annual conference on Autonomous Agents (AGENTS'99), May 1-5, Seattle, USA. ACM Press, New York USA, 69-76.

Wooldridge M., Jennings N. & Kinny D. 2000. The Gaia Methodology for Agent-Oriented Analysis And Design. Journal of Autonomous Agents and Multi-Agent Systems 3(3), 285-312.

Yan Q., Shan L.-J., Mao X.-J. & Qi Z.-H. 2003. RoMAS: A Role-Based modeling Method for Multi-Agent System. Department of Computer Science and Technology, National University of Defense Technology, ChangSha, 410073, China. Saatavilla pdf-muodossa <http://www.auml.org/auml/supplements/RoMAS.pdf> [viitattu 19.12.2003].

Yu E. 1995. Modelling Strategic Relationships for Process reengineering. Ph.D. Thesis, University of Toronto, Department of Computer Science.

Zambonelli F., Jennings N. & Wooldridge M. 2003. Developing Multiagent Systems: The Gaia Methodology. ACM Transactions on Software Engineering and Methodology 12(3), 317-370.