Hyväksymispäivä Arvosana

Arvostelija

ODP-viitemalli yrityksen kokonaisarkkitehtuurin hallinnassa

Sanna Häkkinen

Helsinki 13.3.2012

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

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

Tekijä – Författare – Author

Sanna Häkkinen

Työn nimi – Arbetets titel – Title

Oppiaine – Läroämne – Subject

Tietojenkäsittelytiede

Työn laji – Arbetets art – Level	Aika – Datum – Month and year	Sivumäärä – Sidoantal – Number of pages
Pro gradu -tutkielma		

Tiivistelmä – Referat – Abstract

Avainsanat – Nyckelord – Keywords

UML4ODP, ODP, Kokonaisarkkitehtuuri, Enterprise architecture

Säilytyspaikka – Förvaringställe – Where deposited

Muita tietoja – Övriga uppgifter – Additional information

1 Johdanto

Kokonaisarkkitehtuurin kuvaaminen on tärkeä osa yrityksen toimintaa ja yrityksen toiminnan hallintaa. Kokonaisarkkitehtuurin avulla pyritään antamaan kokonaisvaltainen kuva yrityksen strategiasta, prosesseista, tavoitteista sekä toimijoista. Perimmäisenä tavoitteena on kuvata liiketoimintaprosessien ja IT:n välinen yhteys erilaisten perspektiivien/tavoitteiden avulla [1]. Kokonaisarkkitehtuurin mallinnuksen tärkeys nousee esille ympäristöissä, joissa prosessit ulottuvat yli organisaatiorajojen ja liiketoimintaprosessien hallinta vaatii yhteistoiminnallisuutta.

Tutkielmassa keskitytään verkottuneen organisaation kokonaisarkkitehtuurin mallinnukseen. Tutkielmaa varten toteutettiin osa verkottuneen organisaation kokonaisarkkitehtuurista. Kokonaisarkkitehtuurin mallintamisella pyritään verkottuneen organisaation toiminnan hallintaan, päätöksenteon ja prosessien tukemiseen. Mallinnuksen viitekehykseksi valittiin avoin hajautettu-viitemalli, RM-ODP tai lyhyemmin ODP (Reference Model for Open Distributed Processing).

ODP-viitekehyksen tavoitteena on tarjota välineet suurten, hajautettujen, kompleksisten järjestelmien suunnitteluun ja rakentamiseen. ODP-viitekehyksen tarjoama runko luo arkkitehtuurin, joka tukee hajautusta, yhteistoiminnallisuutta ja siirrettävyyttä [2]. ODP ei ole toteutusstandardi, mutta ODP-viitemalli on kuitenkin tarkoitettu suuriin yhteistyönä toteutettaviin projekteihin, joilla on iteratiivinen luonne, joten viitekehystä pystyy käyttämään jo olemassa olevien iteratiivisten toteutusstandardien rinnalla [2].

ODP-viitekehyksessä järjestelmäkuvaus on hajautettu viiteen näkökulmaan, joista kukin on tarkoitettu tukemaan tiettyä suunnitteluprosessin osaa. Jokainen näkökulma on täydellinen kuvaus järjestelmästä, mutta näkökulma vastaa vain oman alueensa kysymyksiin. Näkökulmat ovat: tavoitenäkökulma, informaationäkökulma, toiminnallinen näkökulma, toteutusnäkökulma ja teknologianäkökulma [2].

Tavoitenäkökulma pyrkii kuvaamaan järjestelmän tavoitteet, liiketoimintasäännöt ja -käytännöt. Tavoitenäkökulma korostaa liiketoimintayksiköitä ja sosiaalisia toimijoita, sekä näiden keskinäisiä riippuvuuksia. *Informaationäkökulma* tarkoituksena on mallintaa järjestelmässä käytetty tieto ja tietotyypit. Informaationäkökulma pyrkii tarjoamaan samanlaiset määritelmät järjestelmän sisältämästä tiedosta jokaisella järjestelmän suunnittelijalle. *Toiminnallinen näkökulma* kuvaa korkean tason toteutuksen prosesseille

ja sovelluksille, jotka tukevat tavoitenäkökulmassa määriteltyjä organisaation toimintoja. Toiminnallinen näkökulma käyttää hyväkseen informaationäkökulman määrittelemiä tietotyyppejä. *Toteutusnäkökulma* kuvaa toiminnallisessa näkökulmassa kuvattujen toimintojen toteutukseen vaadittavan infrastruktuurin. *Teknologianäkökulma* määrittelee käytettävän laitteiston. Jotta viisi näkökulmaa kuvaisi johdonmukaisesti saman järjestelmän, näkökulmat linkitetään toisiinsa *vastaavuuksien* avulla [2].

Tutkielmassa on käytetty ODP-viitemallin kuvaukseen kehitettyä UML standardia, UML4ODP [3]. UML4ODP määrittelee seitsemän UML-profiilia, jokaiselle näkökulmalle, vastaavuuksille ja yhden jolla kuvataan tuotetun järjestelmäkuvauksen yhteensopivuus [2].

Tutkielmaa varten toteutettiin osa verkottuneen organisaation kokonaisarkkitehtuurista, jonka pohjana toimii todellinen organisaatio, sen prosessit ja toimijat. Kokonaisarkkitehtuurin mallinnuksen aikana kiinnitettiin huomiota erityisesti ODP-viitemallin käytön vaikutuksiin. Tavoitteena oli tutkia tuoko ODP-viitemallin käyttö esille uutta informaatiota organisaation toimijoista, prosesseista tai rakenteesta, kun vertauskohtana toimi organisaation prosessien mallinnuksen kautta toteutettu kuvaus. Mielenkiinnon kohteena on myös UML4ODP:lle tuotetun kokonaisarkkitehtuurin laatu. Pystytäänkö UML4ODP:llä tuottamaan ODP-viitemallin mukainen arkkitehtuurikuvaus, ja kuinka laadukas arkkitehtuurikuvauksen UML4ODP:llä saadaan tuotettua? Tutkielmassa analysoidaan myös organisaation kypsymistä kokonaisarkkitehtuurin mallintamisen avulla.

Tutkielman luvussa 2 esitellään tarkemmin tutkimuksen pohjalla oleva organisaatio sekä organisaation asettamat vaatimukset toteutetulle arkkitehtuurille, luvussa 3 esitellään ODP-viitekehys ja ODP-viitemallin asettamat vaatimukset standardinmukaiselle arkkitehtuurikuvaukselle, sekä esitellään tarkemmin UML4ODP, luvussa 4 tarkastellaan laadukkaan arkkitehtuurikuvauksen piirteitä, ja luvussa 5 esitellään ja analysoidaan tutkielman tulokset.

2 Kokonaisarkkitehtuurin hallinta ja muutos

Laajan verkottuneen organisaation hallinta ilman kokonaisarkkitehtuuria on haastavaa. Ilman selvää määrittelyä organisaation strategiasta ja päämääristä, ei toimintakaan voi olla kovin määrätietoista. Kokonaisarkkitehtuurin mallintamisella pyritään tarjoamaan organisaatiolle välineet oman toimintansa ja omien prosessien hallintaan. Kokonaisark-

kitehtuurin ja organisaation kypsyyden arvioinnilla tähdätään organisaation toiminnan kehittymiseen ja kehittymisen tarjoamien mahdollisuuksien kartoitukseen.

2.1 Kokonaisarkkitehtuuri

Kokonaisarkkitehtuuri voidaan määritellä organisaation kokonaisvaltaiseksi kuvaukseksi, joka sisältää kaikki organisaation toimijat, toimijoiden väliset suhteet, organisaation prosessit, IT-resurssit ja infrastruktuurin [2]. Kokonaisarkkitehtuuri ei siis sisällä vain kuvausta organisaation nykytilasta vaan myös tavoitteen siitä minkälainen arkkitehtuurin tulisi olla ja mihin liiketoiminnalla tähdätään. Tavoitteena on, että kaikilla verkottuneen organisaation toimijoilla olisi sama käsitys organisaation kokonaisarkkitehtuurista.

Kokonaisarkkitehtuurin haasteet nousevat esiin toimintaympäristössä, jossa yritykset eivät ole enää itsenäisiä toimijoita vaan yhteistyötä tehdään yli organisaatiorajojen. Organisaatiot tarjoavat toisilleen palveluita ja toimivat verkottuneessa toimintaympäristössä, jonka toimintamallit ovat jatkuvassa muutoksessa. Tämä asettaa uudenlaisia vaatimuksia myös kokonaisarkkitehtuurin mallinnukseen. Yhteistoiminnallisuus ja liiketoimintapalveluiden hallinta nousee tärkeäksi osaksi verkottoutuneen liiketoimintaympäristön kokonaisarkkitehtuurin mallinnusta [4]. Hajautetun liiketoimintaympäristön arkkitehtuurikuvauksen mallintaminen on tämän tutkielman aihealueen ulkopuolella.

Tutkielmassa tullaan tarkastelemaan Metso Oyj:n konsernin taloushallinnon sisäisten organisaatioiden verkottunutta yhteistoimintaa sekä näiden organisaatioiden välisten prosessien ja resurssien mallintamista. Metso on perinteisesti toiminut holding yhtiönä mikä on muokannut yrityksen sisäistä toimintakulttuuria suuntaan, jossa jokaisella organisaatiolla on ollut omat toimintatavat. Uuden strategian myötä, joka tähtää vahvasti eri liiketoiminta-alueiden yhteistoimintaan, yhteistoiminta ja yhteiset toimintamallit ovat alkaneet korostua eri organisaatioiden toiminnassa.

Viimeaikaisista muutoksista taloushallinnan puolella suurin ja näkyvin on ollut siirtyminen yhteiseen taloudenhallintajärjestelmän raportointimalliin, jonka seurauksena koko yrityksen sisällä toimineet neljä erillistä raportointimallia yhdistettiin. Samalla myös järjestelmän ylläpito keskitettiin. Raportointimalli on tarkoitettu sekä ulkoisen- että sisäisen raportoinnin tarpeisiin. Käyttäjäkunta on siis todella laaja ulottuen Metson ylimmästä johdosta pienimpienkin raportointiyksiköiden käyttäjiin. Käyttäjiä on kokonaisuudessaan noin 500, ja lukuja järjestelmässä on 5,000,000. Raportoinnin tuottamaa

tietoa käytetään sekä segmenttien, että koko konsernia koskevien päätösten pohjana. Tästä syystä tarkka ja ajantasainen tieto on elintärkeää oikeiden päätösten tekemisessä. Koska sekä lukuja että käyttäjiä on paljon, järjestelmän käyttöä tukevien prosessien ja tarjottujen palveluiden tulisi olla tarkkaan suunniteltuja.

Metson konsernin taloushallinnon kokonaisarkkitehtuurin mallinnustarpeen taustalla on raportointimallien yhdistämisen mukanaan tuoma organisaatiomuutos. Uudessa organisaatiossa järjestelmän ja raportointimallin ylläpidosta vastaa Metso konsernin taloushallinnon alaisuuteen perustettu Reporting Center of Excellence (RCE) -tiimi yhdessä RCE Advisory Board:n kanssa. RCE jakautuu edelleen kahteen erilliseen tiimiin. Järjestelmän teknisestä ylläpidosta vastaa RCE:n tekninen tiimi ja sisällöllisistä asioista vastaa RCE Finance – tiimi. Yhteistyössä tiimit tarjoavat raportoinnin kehittämisen ja raportointimallin ylläpitopalveluita liiketoiminta-alueille sekä monille raportoijille ympäri maailmaa. Toimintaympäristössä, jossa palveluita tarjotaan näin suurelle käyttäjämäärälle, hyvin suunniteltujen prosessien ja palveluiden merkitys korostuu.

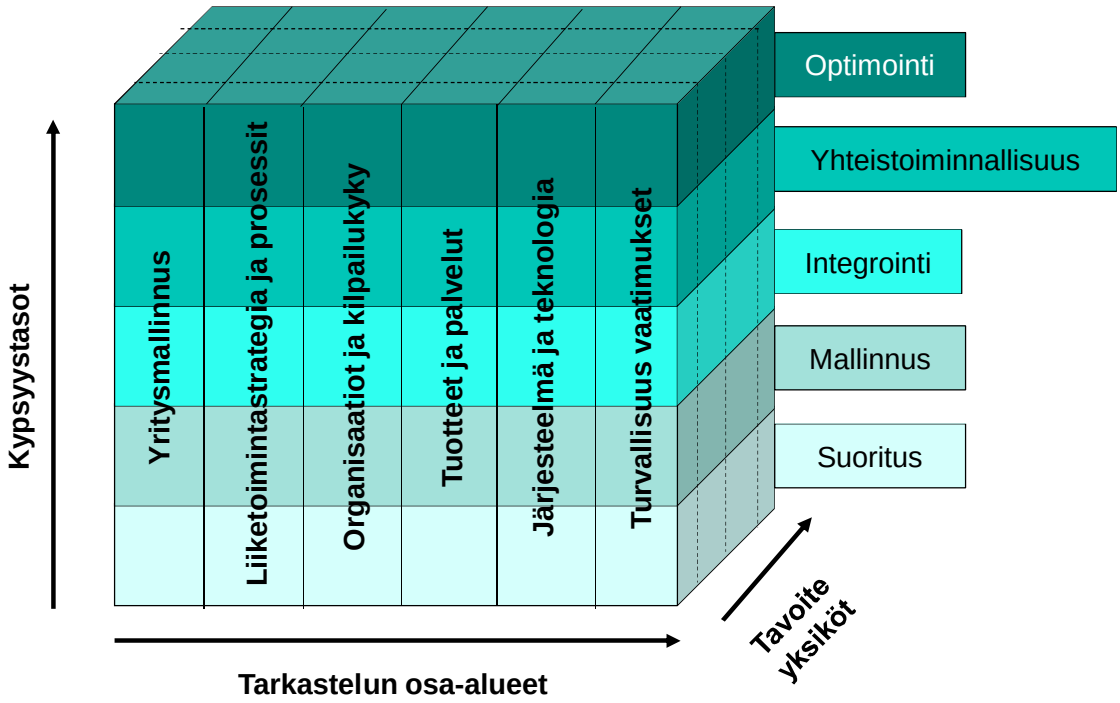
Tilanteessa, jossa organisaatio muuttuu ja uusia tiimejä perustetaan tiimien vastuualueet ja sisäiset prosessit jäävät varsin usein vain hahmotelmien tasolle, prosessit ja vastuut hahmotellaan vain yleisellä tasolla ilman tarkempaa analyysia prosessien tai organisaation tehokkuudesta. Koska organisaation prosesseja ja vastuita ei ole määritelty selvästi, toiminnat toteutetaan ad hoc -prosesseina ilman vastuuhenkilöitä. Metson tapauksessa tämä johti tilanteeseen, jossa dokumentaatiota järjestelmään tehdyistä muutoksista ei ollut saatavilla ja kuukausittain tapahtuvat tehtävät eivät toteutuneet aikataulun mukaan tai ne tehtiin virheellisesti. Tiimien aika kului virheiden etsimiseen ja korjaamiseen.

2.2 Kypsyys organisaatiossa

Kokonaisarkkitehtuurin mallinnus on myös osa yrityksen kypsyyden arviointia. Verkottuneen organisaation kypsyyttä voidaan arvioida monella viitekehyksellä riippuen organisaation toimintatavasta ja arvioinnin tavoitteista. Kypsyysmalleilla pyritään kuvaamaan se tila, jossa tarkasteltava kohde on ja tavoitteet joihin tulisi päästä. Riippuen käytetystä kypsyysmallista, tarkastelun yksiköt sekä kypsyystasot vaihtelevat. Perinteisesti kypsyysmallien alimpana tasona on kaoottinen tila, jossa toimintaa luonnehtii ad hoc -ratkaisut. Tässä tutkielmassa viitekehykseksi on valittu yrityksen yhteistoiminnallisuuden kypsyyden viitekehys (Enterprise Interoperability Maturity Model, EIMM), koska halutaan erityisesti tarkastella verkottuneen organisaation prosessien ja toiminnan

kypsymistä organisaation, prosessien ja tarjottujen palveluiden kehittymisen näkökulmasta.

EIMM jakaa tarkastelun kuuteen osa-alueeseen; yrityksen kuvaus, liiketoimintastrategia ja prosessit, organisaatio ja kilpailukyky, tuotteet ja palvelut, järjestelmä ja teknologia sekä turvallisuus vaatimukset. Näitä osa-alueita tarkastellaan kypsyystasojen avulla; suoritettu, mallinnettu, integroitu, yhteistoiminnallinen ja optimoitu [5]. Eri osa-alueiden ja kypsyystasojen matriisin avulla pystytään arvioimaan verkottuneen organisaation kypsyys, puutteet toiminnassa sekä tavoitteet joihin organisaation tulisi pyrkiä optimoidun yhteistoiminnallisuuden saavuttamiseksi. Kuva 2.1 on kuvaa kypsyystasojen, osa-alueiden ja tavoitteiden väliset suhteet. Vasemmalta oikealle on kuvattu organisaation eri osa-alueet, joita kypsyysmallissa arvioidaan. Alhaalta ylös on taas kuvattuna eri kypsyystasot, joilla osa-alueet voivat olla. Kuution syvyysakseli kuvaa määritellyt tavoitepisteet tai -yksiköt, jotta organisaatio kehittyminen olisi suunniteltua.



Kuva 2.1 EIMM-kypsyysmalli

Metson taloushallinnon verkottunut organisaatio on kiistämättä suoritus-tasolla, jossa toiminta on melko kaoottista ja perustuu pitkälti ad hoc -ratkaisuihin, toimijoiden välisiä suhteita ei ole suunniteltu ja suoritettavien toimintojen aikataulu venyy [5]. Tutkielman aikana toteutettavan kokonaisarkkitehtuurin avulla organisaation tulisi päästä mallinnustasolle, jossa määritelmän mukaan kokonaisarkkitehtuuri on toteutettu ja työkalut arkki-

tehtuurin ylläpitoon on määritelty. Mallinnus-tasolla oletetaan myös, että tietoisuus kokonaisarkkitehtuurin sisältämästä informaatiosta on tavoittanut organisaation toimijat, ja jokainen toimija osaa toimia määriteltyjen prosessien ja mallien mukaan [5]. Seuraavien kypsyystasojen toteuttaminen on tämän tutkielman ulkopuolella. Organisaation tavoitteena on kuitenkin kypsyä optimointi-tasolle, jossa organisaation on mahdollista reagoida liiketoimintakentän muuttuviin vaatimuksiin nopeasti kokonaisarkkitehtuurin avulla. Tutkielman tavoitteena on tarjota ideoita ja mahdollisuudet tämän projektin toteutukseen.

2.3 Kokonaisarkkitehtuurin mallinnus

Tutkielman aikana mallinnetaan osa Metso konsernin taloushallinnon kokonaisarkkitehtuurista. Tutkielman tarkastelun kohteena on raportointimallin ylläpitoprosessi (Change management process). Kokonaisarkkitehtuurin tavoitteena on selkiyttää erityisesti RCE:n roolia järjestelmän ylläpidossa, sekä muiden organisaatioiden rooleja sekä eri toimijoiden rooleja organisaatioiden sisällä. Isona osana kokonaisarkkitehtuurin mallinnusta on eri organisaatioiden välisten rajapintojen ja prosessien selkiyttäminen.

Kokonaisarkkitehtuurin kuvauksessa tullaan käyttämään ODP-viitekehystä, jonka avulla mallinnetaan organisaation prosessit, toimijat, toimijoiden väliset rajapinnat sekä teknologia, joilla kuvattuja prosesseja tuetaan. *Mallinnusprosessin aikana pyritään tutkimaan pystytäänkö ODP-viitekehyksen avulla tuomaan esille uutta tietoa organisaation toiminnasta ja toimijoista.*

UML4ODP:llä tuotettua arkkitehtuurikuvausta arvioidaan ODP:n omilla työkaluilla sekä kirjallisuudessa *esitellyillä laadukkaan arkkitehtuurin määritelmillä.* Kehityshanketta arvioidaan myös EIMM-kypsyysmallilla, jonka avulla tutkitaan arkkitehtuurikuvauksen hyötyjä organisaation toiminnalle, sekä arvioidaan organisaation toiminnan jatkokehityshankkeita. Tavoitteena on parantaa verkottuneen organisaation hallintaa omista prosesseistaan ja toiminnastaan, sekä tarjota organisaatiolle tietoisuus kypsyyden kehittymisen tarjoamista mahdollisuuksista.

3 ODP-viitemalli

ODP-viitemalli, RM-ODP (Reference Model for Open Distributed Processing) julkaistiin 1990-luvun puolivälissä lähes kymmenen vuoden työn tuloksena. RM-ODP julkais-

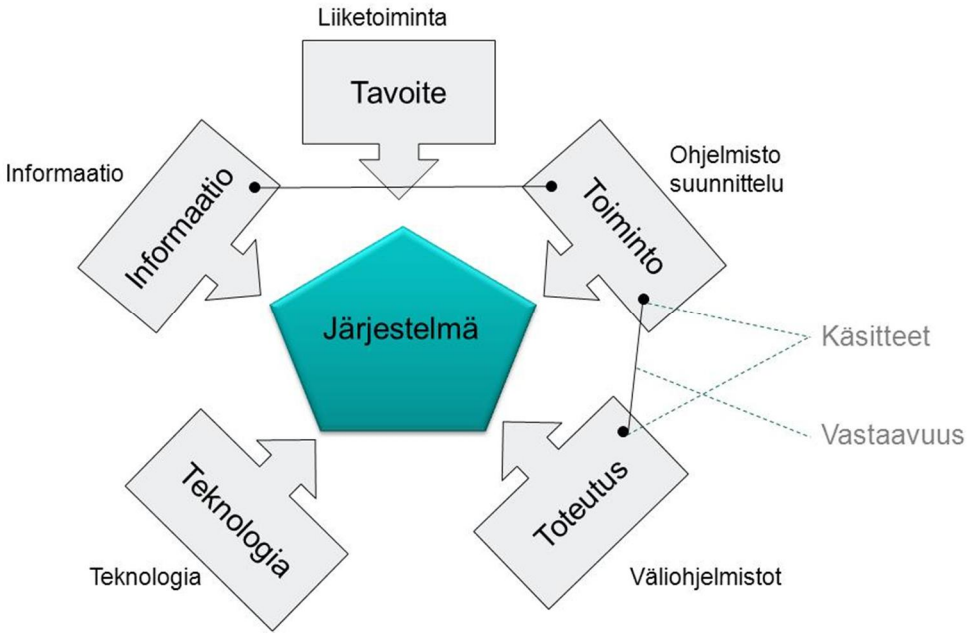
tiin neljässä osassa ISO/IEC IS 10746- 1-4 ([6] [7] [8] [9] [10]). Tutkielman lähteenä käytetyn kirjan kirjoittajat ovat olleet mukana kehittämässä ODP standardia. Tästä syystä tässä kohdassa lähteenä on käytettä vain kirjaa *"Building enterprise systems with ODP - An introduction to Open Distributed Processing"* [2].

3.1 Termit ja käsitteet

ODP on oliopohjainen viitekehys, joka on suunniteltu erityisesti hajautettujen järjestelmien mallintamiseen. Viitekehyksen olioparadigma mahdollistaa mallien korkean abstraktio tason, standardoidut suunnittelukäytäntöjen käytön ja toiminnan kapseloinnin. Korkean abstraktiotason avulla turvataan tuotettujen kuvausten uudelleen käytettävyys ja pitkäikäisyys. Kapseloinnin avulla avoimien ja hajautettujen järjestelmien toiminnan heterogeenisyys saadaan piilotettua, jolloin järjestelmän komponentit ovat toisistaan riippumattomia ja niitä voidaan vaihtaa tai päivittää ilman, että tällä on vaikutusta ympäristöön.

Olioparadigma tarjoaa peruskäsitteistön (toiminta, objekti, vuorovaikutus ja rajapinta), joka muodostaa pohjan ODP-viitekehyksen käsitteistölle. ODP-viitekehyksen peruskäsitteet on määritelty niin, että niitä pystytään käyttämään kaikissa näkökulmissa, joissa käsitteen käyttö voidaan spesifioida vastaavaan näkökulman tarpeita, kuitenkin niin, että se vastaa aina peruskäsitteen määrittelyä.

ODP-viitekehys perustuu viiteen eri näkökulmaan, jotka tarjoavat abstraktin kuvauksen järjestelmän kaikista osa-alueista. Kuva 3.1 kuvaa ODP:n viisi näkökulmaa sekä vastaavuuksien ja näkökulmien suhteen. Tavoitenäkökulman kuvaa järjestelmän liiketoiminta-aspektin, informaationäkökulma mallintaa järjestelmän sisältämän informaation, toimintonäkökulma mallintaa järjestelmän ohjelmistosuunnitelman, toteutusnäkökulma kuvaa järjestelmän toteutuksessa käytetyt väliohjelmistot ja teknologia näkökulma kuvaa järjestelmän käyttämän teknologian. Jotta kaikki näkökulmat muodostaisivat koherentin kuvauksen samasta järjestelmästä, näkökulmien välille määritellään ***vastaavuuksia (correspondencies)***, jotka linkittävät näkökulmat toisiinsa. Jokaiselle näkökulmalla on oma ***näkökulmakieli (viewpoint language)***, jota käytetään kuvaamaan järjestelmää juuri tietystä näkökulmasta.



Kuva 3.1 ODP:n viisi näkökulmaa

ODP:n pohjalta tuotettu järjestelmä kuvaus perustuu ***objekteihin***, jotka on määritetty niiden ***käyttäytymisen*** ja ***tilan*** mukaan. Olioparadigman käyttö mahdollistaa kuvausten ***abstraktoinnin*** sekä ***kapseloinnin***. Abstraktiot mahdollistavat järjestelmän kuvauksen tietystä näkökulmasta. Kapselointi mahdollistaa objektien tietosisällön käytön ainoastaan vuorovaikutuksessa objektin tukemilla rajapinnoilla. Kapseloinnin avulla piilotetaan objektin sisäinen toiminnallisuus.

ODP-viitemallissa objektien käyttäytyminen kuvataan joukolla ***toimintoja (actions)***. Toiminnot voivat tapahtua ***vuorovaikutuksessa*** muiden objektien kanssa ***(interactions)***, tai ne voivat olla objektin ***sisäisiä toimintoja (internal actions)***. Jokaisella toimintaan osallistuvalla objektilla on ***toimintorooli (action role)***, joka määrittelee mitä informaatiota objekti käsittelee toiminnossa ja onko objekti toiminnon käynnistäjä.

ODP-viitemallissa järjestelmä koostuu vuorovaikutuksessa olevista objekteista, jotka kommunikoivat ***rajapinnoilla (interfaces). Tapahtuma (event)*** on piste, jossa toiminto on tapahtunut. Tapahtuman hetkellä suoritetun toiminnon tieto tulee osaksi järjestelmän tilaa, ja voidaan jakaa edelleen muihin vuorovaikutus toimintoihin. ***Vuorovaikutus pisteiden (interaction point)***, jotka on sidottu aikaan ja järjestelmän tilaan, avulla voidaan määritellä erityisiä ***viittaus- (reference point)*** ja ***yhteensopivuuspisteitä (conformance point)***, joiden avulla voidaan suorittaa järjestelmän testausta.

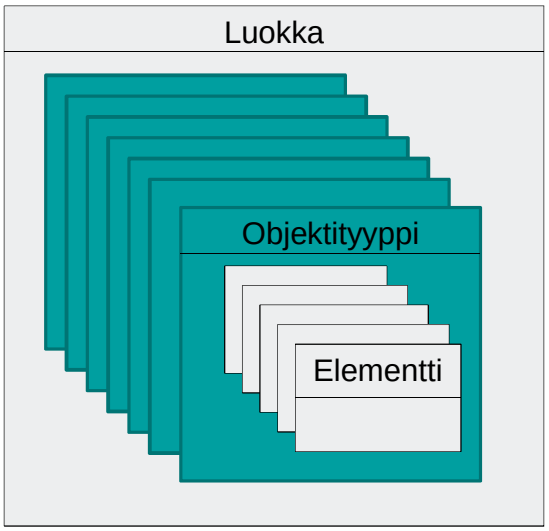
Palvelu (service) on käyttäytymistä, jonka laukaisee vuorovaikutus palvelin- ja asia-

kasobjektien välillä.

3.1.1 Objektityypit ja luokat

Objektin ***tyyppi (object type)*** määrittää objektia kuvaavat elementit, sekä identifioi ja kuvaa määritellyt elementit. Tyyppien avulla objektit voidaan jakaa kokoelmiin, joita ODP-kielessä kutsutaan ***luokiksi (object class)***. Tässä yhteydessä on huomattava, että UML:n luokka käsite eroaa ODP:n käsitteestä siinä, että se kuvailee mitkä objektit kuuluvat luokkaan, kun taas ODP:n luokka on varsinainen objektien kokoelma.

Objektin elementit luodaan erilaisten ***kaavainten (template)*** avulla [2]. Kaavain on elementin spesifikaatio, jonka avulla luodaan elementtien esiintymiä. Kuva 3.2 on kuvaa luokkien, objektien ja elementtien välisen suhde.



Kuva 3.2: Objektit, luokat ja kaavaimet

Rakenteinen objekti (composite object) on kahden tai useamman objektin muodostama kokonaisuus, jonka piirteet koostuvat yhdistettyjen objektien piirteistä sekä tavasta, jolla objektit ovat liitetty yhteen. ***Rakenteinen toiminto (action composite)*** voidaan määrittää prosessin kuvaukseksi [2].

3.1.2 Objektien ryhmittely ja liittäminen

Yksinkertaisimmillaan objektit, joiden tavoitteiden tai rakenteen välillä on ***yhteys (relationship)***, voidaan ryhmitellä ***ryhmiksi (group)***. ***Määrittelyaluetta (domain)*** käytetään kuvaamaan näitä ryhmittelyjä. Määrittelyalue kuvaa objektit, jotka liitetään niiden välisen yhteyden avulla ***hallintaobjektiin (control object)***. Hallintaobjekti ohjaa määrittelyalueen objektien toimintaa.

Objektit voidaan organisoida myös ***konfiguraatioiden (configuration)*** avulla. Konfiguraatio on samoilla rajapinnoilla toimivien objektien kokoelma. Objektikokoelmien toimintaa määrittää kokoelman ***sopimuksessa (contract)*** määritellyt säännöt ja tavoitteet. ***Federaatio (a federation)*** on määrittelyalueiden yhteisö.

Politiikkojen avulla määritellään järjestelmän käyttäytymisen osat, jotka voivat muuttua järjestelmän elinkaaren aikana, ja näiden käyttäytymisten rajoitukset. Politiikat määritellään ***sääntöjen (rules)*** avulla. Säännöt kuvataan ***velvollisuuksina (obligations), lupina (permissions), valtuutuksina (authorization) ja kieltoina (prohibition)***.

3.2 Tavoitenäkökulma

3.2.1 Yhteisöt ja roolit

Tavoitenäkökulma määrittelee suunniteltavan järjestelmän käyttöympäristön organisationaalisen, liiketoiminnallisen ja sosiaalisen kontekstin. Tavoitenäkökulman kuvauksessa käytetään liiketoimintakentälle spesifiä kieltä, jotta liiketoiminta-alueiden sidosryhmät ja suunnittelutiimi pystyvät jakamaan saman käsityksen järjestelmästä sekä järjestelmän tarkoituksesta. Tavoitenäkökulma vastaa kysymyksiin kuten ”Mikä on järjestelmän tarkoitus?”, ”Mitkä ovat järjestelmälle asetetut liiketoimintatavoitteet?” tai ”Mitkä ovat toimintaympäristön sidosryhmät ja miten ne ovat vuorovaikutuksessa toistensa kanssa?”.

Tavoitenäkökulmassa käytetään ***yhteisöjä (communities)*** kuvaamaan liiketoimintaprosesseja ohjaavia sääntöjä ja toimintoja. Yhteisö määrittelee kuinka tietty joukko osallistujia käyttäytyy saavuttaakseen tietyn tavoitteen. Yhteisö on tavoitenäkökulmakielessä *tavoiteobjektien kokoelman konfiguraatio*, joka on muodostettu annetun sopimuksen tavoitteiden saavuttamiseksi. Jotta yhteisön säännöt olisivat uudelleenkäytettäviä, ne kuvataan ***yhteisöroolien (community role)*** välisenä vuorovaikutuksena. Objekti, joka toteuttaa yhteisöroolin, hyväksyy tavoitteekseen saavuttaa asetetut liiketoiminnalliset päämäärät. Tavoitenäkökulmassa yhteisön käsitettä voidaan käyttää kuvaamaan yksittäisen organisaation rakennetta, erillisten organisaatioiden yhteistyötä yhteisösopimuksen määriteltyjen liiketoimintasääntöjen tai palvelusopimusten mukaan tai erillisten määrittelyalueiden federaatiota.

[Kuva: <<EV_RCECommunity>>]

Yhteisösopimuksessa (community contract) on määritelty yhteisön ***käyttäytyminen***

sekä kuvattu yhteisön tavoitteet. Yhteisön käyttäytyminen koostuu kokoelmasta prosesseja, jotka kuvaavat kuinka yhteisön tavoitteet saavutetaan. Prosessi saa parametrina yhteisön roolit. Yhteisön määrityksiin kuuluu usein myös ***politiikkoja/menettelytapoja (policy)***, jotka mahdollistavat yhteisön käyttäytymisen muokkaamisen vaatimusten vaihtuessa. Yhteisösopimus sisältää myös ***toiminnallisia objekteja (enterprise objects)***, jotka kuvaavat yhteisön sisäisen tilan ja resurssit.

[Kuva: <<EV_RCEContract>> Change Management]

Verkottuneet organisaatiot ovat liian kompleksisempia, jotta ne voitaisiin kuvata yhden yhteisökuvauksen avulla. Tavoitenäkökulmassa yhteisö itsessään on yksi tavoiteobjekti, ja voi näin ollen täyttää ylemmän tason yhteisöroolin.

[Kuva: <<EV_ChangeManagementCommunity>>]

3.2.2 Yhteisöjen käyttäytyminen

Tavoitenäkökulmassa yhteisön ***tapahtumiin (action)*** osallistuvat objektit jaotellaan kyseisen tapahtuman ***toimijoihin, resursseihin ja artefakteihin***. Toimija osallistuu aktiivisesti tapahtuman suoritukseen, resurssin allokointi on välttämätöntä tapahtuman suorituksen kannalta ja objekti, joka mainitaan tapahtuman suorituksessa, mutta ei osallistu aktiivisesti suoritukseen on artefakti. Artefaktien tilamuutokset voidaan kuvata tilakoneella.

[Kuva: <<EV_RFCStateMachine>>]

Liiketoimintapalvelu (business service) on kuvaus toiminnasta, jossa yksi tai useampi yhteisön rooli tarjoaa toiminnan tai kapasiteetin muille toimijoille, jotka pystyvät tarjotun toiminnan avulla toteuttamaan omat liiketoimintatarpeensa. Palvelut voivat olla yhteisön sisäisiä tai ne voivat olla julkisia, jolloin myös toimijat muista yhteisöistä voivat hyödyntää tarjotun palvelun.

Yhteisön toiminta koostuu siis kokoelmasta prosesseja (kuva).

[Kuva: <<EV_ChangeManagementProcess>>]

Prosessit (process) ovat määrättyjen ***askelten (step)*** muodostamia tapahtumasarjoja, joissa askeleet voivat olla peräkkäisiä tai rinnakkaisia. Prosessien suoritukseen osallistuu yksi tai useampi yhteisön rooli, jotka osallistuvat liiketoiminta askelten suoritukseen. Jokainen prosessi tähtää yhteisön osatavoitteen saavuttamiseen.

[Kuva: <<EV_ApprovalProcess>>] (ChangeManagementProcess:in osa prosessi)

Vuorovaikutus (interaction) on yhteisön toiminnan peruselementti. Vuorovaikutuskuvauksessa ilmenee mitä rooleja vuorovaikutus pitää sisällään, ja minkä vuorovaikutusroolin jokainen täyttää.

3.2.3 Vastuiden kuvaaminen yhteisön toiminnassa

Liiketoimintaympäristössä joku on aina vastuussa prosessien suorituksesta ja niiden lopputuloksesta. ODP:n tavoitenäkökulmassa vastuut määritellään erityisten tapahtumien avulla. ***Sitoutuminen (commitment)*** on tapahtuma, jossa tapahtuman suorittava objekti ottaa vastaan vastuun. Tällöin vastuiden nykyinen omistaja (***principal)*** siirtää vastuut ***agentille (agent).*** Objekti voi myös toteuttaa ***tiedoksiannon (declaration)***, joka pohjautuu saatavilla olevaan informaatioon. Objekti voi myös toteuttaa tapahtuman nimeltä ***kuvaus (prescription)***, jonka seurauksena syntyy uusi sääntö, joka kuva yhteisön tulevaisuuden toimintaa.

3.3 Informaationäkökulma

Informaationäkökulma kuvaa liiketoiminta-alueen käyttämän tiedon. Informaationäkökulman avulla tuotetaan jaetun informaation abstrakti malli, joka takaa yhtenäisen näkemyksen eri sidosryhmien välillä. Näkökulma rajaa tarkastelun ulkopuolelle kaikki toteutukseen tai toteutusalustaan liittyvät kysymykset. Informaationäkökulma on myös erillään rajapinnoista ja funktioista, jotka muokkaavat dataa. Tuotettu informaatiomalli vastaa kysymykseen kuten ”Mitkä ovat järjestelmän tietotyypit?”, ”Mikä on eri tietotyyppien välinen suhde?”, ”Kuinka datan tila muuttuu järjestelmän toimiessa?”, Mitkä ovat järjestelmän sallimat toiminnot ja kuinka nämä toiminnot vaikuttavat datan tilaan?”, ”Mitkä ovat dataa ja datan prosessointia koskevat rajoitukset?”.

3.3.1 Informaatio-objektit

Informaatiospesifikaatio määrittelee ***informaatio-objektit (information object)***, jotka kuvaavat järjestelmän käyttämän datan. Jokaisella informaatio-objektilla on identiteetti, tila ja määritelty käyttäminen, ja ne ovat vuorovaikutuksessa toisten informaatio-objektien kanssa. Informaatio-objektin kuvauksen voi rinnastaa ohjelmointikielten abstrakteihin luokkiin, joita ODP-viitemallissa vastaa objektien ***tyypit (information type)***. Jokaisella informaatio-objektilla on tyyppi. Samantyyppiset objektit jakavat samat piir-

teet ja käyttäytymisen.

[Kuva: <<IV_RCE_Information_Object_Types>>]

Informaatiotoiminnot (information actions) kuvaavat tiedon prosessointia järjestelmässä. Jokainen toiminto on liitetty ainakin yhteen informaatio-objektiin. Myös toiminnoilla on tyyppi, joka yhdistää samat ominaisuudet omaavat toiminnot.

[Kuva: <<IV_RCE_Information_Object_Actions>>]

3.3.2 Skeemat

Toiminnot aiheuttavat tilamuutoksia niihin yhdistettyihin informaatio-objekteihin. Nämä tilamuutokset kuvataan ***skeemojen (schema)*** avulla. Informaationäkökulmakieli määrittelee datan käyttäytymisen skeemojen avulla. ***Dynaaminen skeema (dynamic schemata)*** kuvaa informaatio-objektien käyttäytymisen. Dynaaminen skeema määrittelee kuinka data kehittyy järjestelmän toimiessa kuvaamalla yhden tai useamman informaatio-objektin mahdolliset tilamuutokset. Tilamuutos on tilamuutokseen liittyvien informaatio-objektien välistä vuorovaikutusta. Dynaaminen skeema kuvaa myös informaatio-objektien luomisen ja poistamisen.

[Kuva: <<IV_RFC_Dynamic_Schema>>]

Invariantti skeema (invariant schemata) asettaa informaatio-objektien ja niiden käyttäytymisen rajoitteet. Invariantti skeema voi kuvata informaatio-objektien tyypit, tyyppien väliset suhteet sekä tyyppejä ja niiden välisiä suhteita koskevat rajoitukset. Invariantissa skeemassa määritellyt rajoitteet koskevat kaikkia dynaamisessa skeemassa määriteltyjä käyttäytymisskeemoja.

[Kuva: <<IV_RFC_Invariant_Schema>>]

Staattinen skeema (static schemata) kuvaa informaatio-objektien konfiguraation tietyllä ajanhetkellä. Skeemat voivat liittyä järjestelmän johonkin tiettyyn alueeseen tai koko järjestelmään. Staattinen skeema kuvaa esimerkiksi RCE:n muutosprosessin informaatio-objektien tilan onnistuneen prosessin lopussa.

[Kuva: <<IV_RFC_Static_Schema>>]

3.4 Toimintonäkökulma

Toimintonäkökulma kuvaa järjestelmän toiminnallisuuden ja järjestelmäarkkitehtuurin

joka tukee toiminnallisuutta, mitä palveluita järjestelmä tarjoaa ja miten palvelut on toteutettu eri komponenttien ja yhteyksien avulla. Toimintonäkökulma ei ota kantaa järjestelmäkomponenttien hajautukseen tai teknologiaan, jolla komponentit on toteutettu. Näin järjestelmäarkkitehtuurikuvauksesta saadaan hajautuksesta ja alustasta riippumaton, jolloin kuvaus on uudelleenkäytettävissä.

Toimintonäkökulma on oliopohjainen, se jakaa järjestelmän loogisiin olioihin, jotka kommunikoivat rajapintojen avulla. Toimintonäkökulma kuvaus koostuu ***toiminto-objekteista (computational object), rajapintakuvauksesta,*** rajapinnoilla tapahtuvan objektien välisen ***vuorovaikutuksen*** kuvauksesta ja objektien ***ympäristökuvauksesta (environment contract)***.

Ympäristökuvauksessa määritellään toiminto-objektien ja niiden välisen vuorovaikutuksen ei-toiminnallisten ominaisuudet. Ei-toiminnallisia ominaisuuksia ovat esimerkiksi palvelun laadun – määritykset (Quality of Service, QoS) ja palvelutason sopimukset.

3.4.1 Toiminto-objektit ja niiden välinen vuorovaikutus

Toiminto-objektit kuvaavat järjestelmän toiminnallisuuden palveluina, joita objektit tarjoavat ja käyttävät. Jokaiselle toiminto-objektille on määritelty vähintään yksi rajapinta, joka kuvaa mitä palveluita toiminto-objekti tarjoaa ja mitä palveluita se käyttää. Jokainen palvelu on määritelty siihen liittyvän toiminnan sekä sen elementtien syntaksin avulla. Syntaktisuus kuvataan ***toimintojen (operation), vuon (stream)*** tai ***signaalin (signal) allekirjoituksissa (signatures)***.

ODP-viitekehys määrittelee kolme vuorovaikutuksen muotoa sekä näitä tukevat rajapinnat: ***toiminnalliset rajapinnat (operation interface), vuorajapinnat (stream interface)*** ja ***signaalirajapinnat (signal interface)***.

Toiminnallisissa rajapinnoissa asiakas kutsuu palvelimen operaatioita, käsite toteuttaa siis palvelin-asiakas – mallin. Toiminnallinen rajapinta koostuu nimetyistä operaatioista, joilla on määritelty parametrit, lopetukset ja tulokset. ODP:ssä on kahdenlaisia operaatioita: ***kysely (interrogation)*** ja ***ilmoitus (announcement).*** Operaatioiden tyypityksellä erotetaan yksisuuntainen ja kaksisuuntainen vuorovaikutus toisistaan. Kysely palauttaa tuloksen, kun taas ilmoitus käynnistää operaation. Kysely pysäyttää asiakkaan suorituksen kunnes operaatio saa paluuarvon. Ilmoitus ei välttämättä pysäytä asiakkaan suoritusta, asiakas ja palvelin voivat näin toimia synkronisesti.

Vuorajapinnat tarjoavat jatkuvia informaatiovirtoja tuottajan ja asiakkaan välillä. Useita informaatiovirtoja voidaan yhdistää saman rajapinnan alle. Vuorajapinnat onkin tarkoitettu multimedia- ja tietoliikennesovelluksille.

Signaalirajapinnat ovat rajapinnoista alkeellisimpia ja ne tarjoavat hyvin alhaisen tason kommunikointitoimintoja. Signaalien avulla voidaan siirtää alhaisen tason viestejä, kuten tapahtumia ja keskeytyksiä. Sekä toiminnalliset että vuorajapinnat perustuvat signaalirajapintaan.

[Kuva: <<CV_Esimerkki>>]

3.4.2 Toiminto-objektien sidokset

ODP:ssä ***sidoksella (binding)*** tarkoitetaan kahden tai useamman objektin välille muodostettua vuorovaikutussuhdetta, joka luodaan ***sidotoiminnon (binding action)*** avulla. Normaalisti toiminto-objekti aloittaa vuorovaikutuksen toisen objektin kanssa suorittamalla sidostoiminnon, joka mahdollistaa palveluiden ja tiedon jakamisen rajapintojen kautta objektien välillä. Sidos voidaan muodostaa myös kolmannen objektin toimesta ***(third-party binding)***, jolloin objekti voi muodostaa sidoksen useamman muun objektin kanssa.

Yksinkertainen sidonta (primitive binding) tapahtuu suoraan kahden toiminto-objektin välillä. Tällaisessa sidoksessa kahden objektin tarjoamat ja tarvitsemat palvelut yhdistetään jakamalla UML-rajapinta, joka määrittelee allekirjoituksen operaatiot.

Koosteinen sidonta (compound binding) tapahtuu erityisen sidosobjektin avulla. Sidosobjekti kapseloi toiminnallisuuden, joka tarvitaan kahden tai useamman toiminto-objektin yhdistämiseen. Sidos-objekti tarjoaa myös kontrollirajapinnan, joka mahdollistaa liitäntä mekanismien konfiguroinnin ja hallinnan.

3.5 Toteutusnäkökulma

Toteutusnäkökulman tärkeimpänä tavoitteena on tukea toiminto-objektien hajautuksen ***tuntumattomuuden (transparency)*** toteutusta. Toteutusnäkökulma identifioi ja määrittelee ne mekanismit, joilla hajautettu vuorovaikutus toimintonäkökulmassa määriteltyjen objektien välillä toteutetaan. Tuotettu kuvaus keskittyy määrittelemään kuinka hajautus on toteutettu, kuinka objektit on hajautettu järjestelmän ***solmujen (nodes)*** välillä, ja kuinka solmut ja ***kanavat (shannels)***, jotka yhdistävät solmut, tullaan toteuttamaan.

Kuvauksessa määritellään myös hajautuksen vaatiman tuntumattomuuden toteutukseen tarvittavat funktiot.

Toteutusnäkökulma keskittyy siis järjestelmän infrastruktuurin kuvaukseen. Jälleen toteutusnäkökulman tärkeimpänä tavoitteena on tarjota teknologiariippumaton arkkitehtuuri-viitemalli, jonka avulla pystytään suunnittelemaan uusia järjestelmiä tai vertailemaan jo olemassa olevia hajautettujen järjestelmien infrastruktuuriteknologioita. Näin toteutuksessa käytetyt teknologiat voivat kehittyä vaikuttamatta järjestelmäkuvaukseen.

Yhteensopivuustestaus (conformance testing) suoritetaan myös tämän näkökulman pohjalta. Toteutusnäkökulmassa määritellään teknologia riippumattomat testauksen perusvaatimukset.

3.5.1 Objektien hajautus

Toiminto-objektien vuorovaikutuksen tulee olla vähintään ***saantituntumatonta (access transparency)*** sekä ***paikkatuntumatonta (location transparency)***, myös muita tuntumattomuusvaatimuksia on voitu määritellä. Saantituntumattomuus peittää eroavaisuudet datan esityksessä ja palveluiden käynnistysmekanismit. Paikkatuntumattomuus peittää sovelluksen tarpeen tietää lokaatio informaatiota käynnistääkseen palvelun.

Jokaista toiminto-objektia vastaa toteutusnäkökulmassa ***toteutusobjekti (Basic Engineering Object, BEO)***. Tässä kohtaa ollaan lähinnä kiinnostuneita koneiden ja verkottuneen vuorovaikutuksen toteutuksesta. Ihmisten kuvaaminen on toteutusnäkymän ja ODP-viitemallin ulkopuolella. Toteutusobjektien joukko voidaan nähdä toteutussuunnitelman abstraktiona. Tämän tason abstraktiolla pystytään piilottamaan eroavaisuudet samanlaiset vuorovaikutusvaatimukset omaavien toiminto-objektien välillä ja kuvaamaan toiminto-objektit tarjotun hajautetun alustan käyttäjinä. Näin määriteltynä BEO:t ovat objekteja, jotka sijoitetaan tiettyyn solmuun, ja jotka aloittavat kommunikoinnin verkossa. Muut toteutusnäkymän objektit ovat apuobjekteja, jotka määritellään solmuarkkitehtuurissa ja joiden tehtävänä on tukea hajautuksen toteutusta.

[Kuva: <<NV_Esimerkki>>]

3.6 Teknologianäkökulma

Usein järjestelmäprojekteissa suunnittelijat joutuvat ottamaan suunnitelmissaan huomioon yrityksen asettamat infrastruktuurirajoitukset. Nämä vaatimukset asettavat usein

melko tiukatkin rajat järjestelmän toteutukselle, ja tätä kautta myös ohjelmistoarkkitehtuurille, käytettävälle ohjelmointikielelle ja järjestelmän toiminnallisuuden hajautukselle. Vaikka nämä vaatimukset ovat suuri osa järjestelmäkuvausta, ja vaikuttavat ratkaisevasti järjestelmän suunnitteluun ja toteutukseen, ei niiden kuvaamista ole otettu huomioon useissakaan spesifikaatioissa tai vaatimukset määritellään hyvin myöhäisessä suunnitteluvaiheessa, jolloin järjestelmäsuunnitelmaan ei pystytä tekemään enää suuria muutoksia.

Yhteensopivuus- ja standardinmukaisuustestauksen suunnitteluun tarvitaan tietoa järjestelmälle asetetuista vaatimuksista. ***Yhteensopivuustestaus (conformance)*** varmistaa, että järjestelmän toteutuksessa käytetyt komponentit, kieli ja protokollat vastaa järjestelmälle asetettuja pakollisia vaatimuksia. Yhteensopivuustestaus määritellään ODP:ssä tarkistuspisteiden avulla, joissa on mahdollista verrata toteutusta ja vaatimuksia keskenään. ***Standardinmukaisuustestauksella (compliance)*** varmistetaan, että toteutettu järjestelmäspesifikaatio on yhdenmukainen käytetyn viitekehyksen kanssa.

Järjestelmäspesifikaation tulisi sisältää myös suunnitelma teknologian valintaprosesseille tulevaisuudessa sekä suunnitelman järjestelmän kehityksestä. Näiden suunnitelmien tarve on erittäin suuri, koska teknologia muuttuu jatkuvasti, ja muutoksiin tulisi varautua jo suunnitteluvaiheessa.

Teknologianäkökulman on tarkoitus vastata näihin kysymyksiin ja tarjota välineet johdonmukaiselle järjestelmäsuunnittelulle. Teknologianäkökulma tarjoaa käsitteet ja konstruktiot laitteisto- ja ohjelmistokomponenttien määrittämiseksi. Tuotetun kuvauksen avulla pystytään testaamaan toteutuksen yhteensopivuus muiden näkökulmien määritysten kanssa ja määrittelemään järjestelmäosien valinta- ja hankintaprosessit sekä evoluutio järjestelmän elinkaaren ajaksi.

Teknologianäkökulma määrittelee neljä peruskäsitettä, joiden avulla kuvaus tuotetaan. ***Teknologiaobjektit (technology objects)***, jotka kuvaavat järjestelmän laitteisto- ja ohjelmistokomponentit, ***toteutustandardit (implementation standard)***, itse toteutus ja ***IXIT (Implmentation eXtra Information for Testing),*** joka määrittelee yhteensopivuustestauksen testauspisteet.

3.7 Vastaavuuksien hallinta

Eri näkökulmat kuvaavat saman järjestelmän eri sidosryhmien näkökulmasta. Vastaa-

vuudet liittävät nämä eri näkökulmaa yhteen määritellen eri näkökulmat limittyvät ja mitä entiteettejä voidaan näkökulmista löytää. Vastaavuudet yhdistävät aina kaksi näkökulmaa keskenään. Määritelty kuvaus on ***vastaavuusspesifikaatio (correspondence specification)***. Vastaavuusspesifikaatioita voi maksimissaan olla kymmenen, ja niitä tulee olla vähintään neljä, jotta kokonaiskuvaus olisi linkittynyt kaikkien näkökulmien välillä.

3.7.1 Vastaavuuksien tyypit

Vastaavuudet siis määrittelevät objektien välisiä vastaavuuksia eri näkökulmien välillä. Vaikka kaksi objektia kuvaisikin samaa entiteettiä, voi niiden ominaisuudet erota toisistaan, koska eri näkökulmat keskittyvät kuvaamaan eri asioita. Näkökulmien objektit kuvaavat saman entiteetin eri abstraktiot. Spesifikaatiossa tuleekin määritellä koskeeko määritelty vastaavuus objektien tyyppiä kokonaisuudessaan vai yksittäisiä attribuutteja, ja vastaavatko nämä attribuutit todella toisiaan. Eri näkökulmat voivat myös kuvata saman entiteetin ominaisuudet erilaisilla tarkkuustasoilla. Vastaavuudet voivat myös lisätä spesifikaation informaatiota määrittelemällä korrelaation eri näkökulmissa määriteltyjen attribuuttien välillä.

ODP-viitemalli määrittelee neljä eri vastaavuusluokkaa. ***Ohjelmallinen testauspiste (programmatic reference point)*** on järjestelmän rajapinta, jossa vuorovaikutusta voidaan tarkkailla. ***Ulkoisten vaikutusten testauspiste (perceptual reference point)*** on piste, jossa järjestelmä on näytön tai näppäimistön avulla vuorovaikutuksessa ympäristön kanssa. Näiden pisteiden testaus on konkreettista järjestelmän havainnointia sillä hetkellä kun vuorovaikutus tapahtuu. ***Kommunikoinnin testauspisteessä (interworking reference point)*** testataan fyysisiä kommunikoinnin kanavia kuten alijärjestelmien linkkejä tai verkkolinkkiä. S***iirrettävyyden testauspisteessä (interchange reference point)*** tarkkaillaan jonkin fyysisen (medium) tilaa ennen määriteltyä prosessia ja prosessin jälkeen.

3.7.2 Vastaavuuksien toteutuksen reunaehdot

ODP-viitemalli asettaa hyvin vähän vaatimuksia sille missä vastaavuuksia tulee määritellä, joitakin reunaehtoja kuitenkin on. Ensimmäinen vaatimus on, että informaationäkökulmassa määritellyt informaation tilamääritykset, jotka on kuvattu invariantti- ja dynaamisissa skeemoissa, tulee olla jäljitettävissä toimintonäkökulmassa vuorovaikutuksiin tai sisäisiin tapahtumiin vastaavissa objekteissa. Toinen vaatimus on, että toimin-

tonäkökulmassa määriteltyjen tuntumattomuuksien toteutukset määritellään toteutusnäkökulmassa. Näin varmistetaan, että vaaditut tuntumattomuudet pystytään toteuttamaan ilman odottamattomia sivuvaikutuksia.

3.8 UML4ODP

UML4ODP standardi määrittelee sekä UML-pohjaisen notaation ODP järjestelmäkuvausta varten että järjestelmäkuvausten strukturoinnin työkalut. UML4ODP määrittelee UML-profiilin jokaiselle näkökulmalle ja vastaavuuksien ja ODP-mallin yhteensopivuuksien kuvaamiseen [2].

4 Arvioinnin työkalut

4.1 ATAM

4.2 Standardinmukaisuus ja yhteentoimivuus

- Mitä ODP tuo lisää vrt. vanhemmat EA:t?
- Miten ODP:ssä arvioidaan standardinmukaisuutta?

4.3 EIMM

- Miten organisaation kypsyyttä arvioidaan?
- Miten kokonaisarkkitehtuuri liittyy kypsyysmalliin

5 Kokonaisarkkitehtuurin mallinnus ODP:n avulla

5.1 Projektin toteutus

- Case study (teoria): [1]
    - Miksi päätökset tehtiin?
    - Miten päätökset toteutettiin?
    - Mikä oli toteutuksen lopputulos?

5.2 Kokonaisarkkitehtuurin arviointi

- ATAM

5.3 Projektin vaikutusten arviointi

5.4 Organisaation kypsyys

- EIMM

5.5 Kehitysehdotukset

6 Yhteenveto

7 Lähdeviitteet

[1] Matthias Lange and Jan Mendling, "An experts' perspective on enterprise architecture goals, framework adoption and benefit assessment," in *15th International Enterprise Distributed Object Computing Conference Workshop*, 2011, pp. 304-313.

[2] Peter F. Linington, Zoran Milosevic, Akira Tanaka, and Antonio Vallecillo, *Building enterprise systems with ODP - An introduction to open distributed processing.*, 2011.

[3] ISO/IEC IS 19793, Use of UML for ODP systems specifications (IS), 2007, Also published as ITU-T Recommendation X.906.

[4] Lea Kutvonen, "Using the ODP reference model for Enterprise Architecture," in *11th EDOC Conference Workshop*, Annapolis, MD, 2007, pp. 231 - 238.

[5] ATHENA. (2010, Dec) ATHENA Integrated Project. [Online]. http://modelbased.net/aif/methodology/eimm.html

[6] ISO/IEC IS 10746-1, Informatio Techonology – Open Distributed Processing – Reference Model: Overview, 1998, Also published as ITU-T Recommendation X.901.

[7] ISO/IEC IS 10746-2, Informatio Techonology – Open Distributed Processing – Reference Model: Foundation, 2009, Also published as ITU-T Recommendation X.902.

[8] ISO/IEC IS 10746-3, Informatio Techonology – Open Distributed Processing – Reference Model: Architecture, 2009, Also published as ITU-T Recommendation X.903.

[9] ISO/IEC IS 10746-4, Informatio Techonology – Open Distributed Processing – Reference Model: Architectural Semantics, 1998, Also published as ITU-T Recommendation X.904.

[10] ISO/IEC IS 19793, Information technology - Open distributed processing - Use of UML for ODP system specifications, 2008, Also published as ITU-T Recommendation X.906.

[11] Robert K. Yin, *Case study and research: Design and methods 4th ed.*: SAGE Publications, Inc., 2009.

[12] Len Bass, Clements Paul, and Rick Kazman, *Software architecture in practice.* Boston: Addison-Wesley, 2003.

[13] Sabine Buckl, Florian Matthes, Sascha Roth, Christopher Schulz, and Christian M. Schweda, "A Conceptual Framework for Enterprise Architecture Design," in *Trends in Enterprise Architecture Research 5th International Workshop, TEAR 2010*, vol. 70, Delft, The Netherlands, 2010, pp. 44-56.

[14] ITU-T Rec X.906 | ISO/IEC 19793, "Use of UML for ODP systems specifications (IS)," 2007.