ODP-viitemalli

Avoimien hajautettujen järjestelmien tutkimusseminaari
07.03.2000
Eero Raunio
puh. 09 – 638 959

SISÄLLYS

1. JOHDANTO 2

2. ODP:N OLIOMALLI 3

3. NÄKÖKULMAT 4

3.1 TAVOITENÄKÖKULMA 5
3.2 INFORMAATIONÄKÖKULMA 7
Invariantti skeema 7
Staattinen skeema 7
Dynaaminen skeema 7
3.3 TOIMINNALLINEN NÄKÖKULMA 7
3.4 TOTEUTUSNÄKÖKULMA 9
Solmut, kapselit ja klusterit 9
Kanavat 10
3.5 TEKNOLOGIANÄKÖKULMA 11
3.6 NÄKÖKULMIEN RIIPPUVUUDET TOISISTAAN 11

4. FUNKTIOT 12

4.1 HALLINTAFUNKTIOT 12
Solmun hallintafunktio 12
Kapselin hallintafunktiot 12
Klusterin hallintafunktio 12
Olion hallintafunktio 12
4.2 KOORDINOINTIFUNKTIOT 12
Tapahtuman tiedonanto -funktio 13
Tarkistuspiste ja palautus –funktio 13
Aktivointi ja passivointi –funktio 13
Ryhmäfunktio 13
Toisinnusfunktio 13
Siirtymisfunktio 13
Sidospisteen viitteen seuraamis –funktio 13
Transaktiofunktio 14
ACID - transaktio-funktio 14
4.3 TIETOVARASTO-FUNKTIOT 14
Tallennusfunktio 14
Informaation organisointi -funktio 14
Uudelleensijoitusfunktio 14
Tyyppienhallintapalvelu 15
Meklauspalvelu 15
4.4 TURVALLISUUSFUNKTIOT 15

4. TUNTUMATTOMUUDET 15

SAANTITUNTUMATTOMUUS 16
VIKATUNTUMATTOMUUS 16
PAIKKATUNTUMATTOMUUS 16
SIIRTYMISTUNTUMATTOMUUS 16
KESKEYTYSTUNTUMATTOMUUS 17
UUDELLEENSIJOITUMISTUNTUMATTOMUUS 17
TOISINNUSTUNTUMATTOMUUS 17
TRANSAKTIOTUNTUMATTOMUUS 17

5. YHTEENVETO 17

LÄHTEET 19

1. Johdanto

ODP:n tarkoituksena on mahdollistaa standardien tuottaminen, jotka hyödyntävät heterogeenistä laitealustaa usean organisaation alueella. ODP-viitemalli (RM-ODP, The Reference Model for Open Distributed Processing) on standardi, joka koostuu neljästä osasta: Osa 1 – Yleiskatsaus, Osa 2 - Perusteet, Osa 3 –Arkkitehtuuri ja Osa 4 - Arkkitehtuurin semantiikka. Osat 2 ja 3, ja niiden selitykset osassa 1, muodostavat metastandardin, eli viitemallin. Metastandardi ei itse määrittele standardeja avoimiin ja hajautettuihin järjestelmiin, vaan luo kehysrakenteen, joka määrittää käsitteet ja toiminnat. Osa 4 antaa muodollisen tulkinnan.

Avoimien ja hajautettujen järjestelmät ovat erittäin monimutkaisia ja niiden suunnittelussa on otettava asioita huomioon laajalta alueela. ODP jakaa määrittelyn osiin viiden (5) näkökulman avulla, joista kukin on tarkoitettu tukemaan tiettyä suunnitteluprosessin osaa. Jokainen näkökulma on täydellinen kuvaus järjestelmästä, mutta näkökulmä vastaa vain kysymyksiin, jotka kuuluvat sen alueeseen.
Näkökulmat ovat: tavoite-, informaatio-, toiminnallinen-, toteutus- ja teknologinennäkökulma. Tavoitenäkökulma käsittelee järjestelmän tehtäviä yrityksessä liiketoiminnan kannalta. Informaationäkökulma käsittelee organisaation tietoelementtejä ja niiden liikettä. Toiminnallinennäkökulma käsittelee järjestelmän käyttäytymistä. Toteutusnäkökulma käsittelee järjestelmän toteuttamisen vaatimaa infrastruktuuria. Teknologianäkökulma käsittelee järjestelmän vaatimuksia konkreettisten laitteistojen ja ohjelmistojen kannalta.

ODP on oliopohjainen määrittelytapa. ODP:n oliomalli on tehty avoimia ja hajautettuja järjestelmiä varten. ODP:ssä olioita käytetään rajapintojen kautta. Kukin rajapinta kuvaa olion tiettyä roolia, olio voi toteuttaa useita rajapintoja. Tyyppi on ominaisuus, jonka tietyn olioiden joukon tulee omata. Luokka on joukko olioita, jotka ovat tiettyä tyyppiä.

ODP määrittelee joukon funktioita, joita tarvitaan ODP sovellusalustassa. Tuntumattomuuden avulla voidaan peittää hajautettuihin järjestelmiin liittyviä ongelmia, näin ohjelmoijan ei tarvi välittää näistä ongelmista. Hajautuksen tuntumattomuudet ODP:ssa ovat: saantituntumattomuus, vikatuntumattomuus, paikkatuntumattomuus, siirtymistuntumattomuus, keskeytystuntumattomuus,

uudelleensijoitustuntumattomuus, toisinnustuntumattomuus ja transaktiotuntumattomuus. Tuntumattomuudet toteutetaan funktioiden avulla

2. ODP:n oliomalli

ODP:ssä järjestelmä koostuu vuorovaikutuksessa olevista olioista. Olioiden abstraktiotasoa ei ole rajoitettu. Oliot kapsuloivat toteutuksensa yksityiskohdat ja tarjoavat abstraktin kuvan käyttäytymisestään. Olioiden metodeja käytetään rajapinnan kautta. Oliolla voi olla monta rajapintaa, joista kukin on yksi abstraktio oliosta. Syynä monirajapintaisuuteen on funktionaalinen hajautus ja erottaminen. Rajapinnassa erotellaan asiakas ja palvelin puoli, oliolla on joko palvelun kuluttajan tai tarjoajan rooli.

Avoimet ja hajautetut ovat hyvin heterogeenisiä, ne sisältävät erilaisia toteutuksia, mekanismejä ja tekniikoita. Abstraktoinnin avulla tämä heterogeenisyys saadaan piilotettua. Näin oliot ovat toisistaan riippumattomia. Oliota voidaan vaihtaa tai päivittää ilman, että ympäristöä tarvii muuttaa.

Kapsulointi tarkoittaa sitä, että olio sisältämän tiedon käsittely onnistuu vain määritellyn rajapinnan kautta. Olion tila tarkoittaa sen sisältämien tietojen arvoja. Olion tila voi muuttua vain vuorovaikutuksen tai sisäisen tapahtuman tuloksena. Vuorovaikutuksella ei siis voi olla huomaamattomia vaikutuksia. Olio voidaan myös toteuttaa monella eri tavalla, monessa eri järjestelmässä, kunhan kukin toteutus sisältää vaaditun rajapinnan.

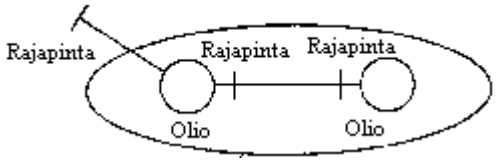
Olion käyttäytyminen on ODP:ssä määritelty joukoksi potentiaalisia toimintoja joihin olio voi osallistua. Oliomalli ei rajoita käyttäytymisen muotoa eikä luonnetta. Käyttäytyminen kuvaa kaikki potentiaaliset tilan vaihdot. Käyttäytyminen määritellään joukkona rajapintoja. Kaksi oliota on käyttäytymiseltään yhteensopivia, jos olio voi korvata toisen, ilman että se vaikuttaa ympäristöön lainkaan.

Olio kuvataan kaavaimen avulla. Kaavain kuvaa täysin olion ominaisuudet valitulla abstraktiotasolla, esim. tilaparametrit ja metodit. Kun kaavain sisältää tarpeeksi informaatiota, sen avulla olio voidaan luoda. Tätä kutsutaan instantioinniksi. Olioita voidaan luoda järjestelmään myös esittelyn avulla [6]. Esittelyssä riittää tieto olion

tyypistä. Tyyppi on vaatimus, eli se on oliojoukon ominaisuus, esim. punainen. Olio on tiettyä tyyppiä jos vaatimus pätee sille. Saman tyypin olioiden ei tarvi olla samanlaisia, riittää että niillä on tyypin ominaisuudet. Luokka tarkoittaa niitä olioita, joilla on tyypin ominaisuudet. Kaavain tyyppi on vaatimus, joka pätee kaikille kaavaimen avulla instatioiduille olioille.

ODP:n oliomalli määrittelee myös alityypin ja –luokan käsitteet. Luokka A on luokan B aliluokka vain jos A on B:n osajoukko. Tyyppi C on tyypin D alityyppi vain jos A:n vaatimukset implikoivat B:n vaatimukset. Jos tyyppiin C liittyy luokka A ja tyyppiin D luokka B, tyyppi C on D:n alityyppi jos A on B:n aliluokka.

ODP-viitekehys määrittelee myös rakenteisen olion käsitteen. Rakenteisuuden avulla kuvataan rakenteisen olion ja sen komponenttien hierarkiaa. Rakenteinen olio on tilojen ja käyttäytymisien yhdistelmä. Rakenteisen olion komponenttien välinen vuorovaikutus ei näy olion ulkopuolelle. Rakenteista oliota käytetään sen rakenteisten rajapintojen kautta.



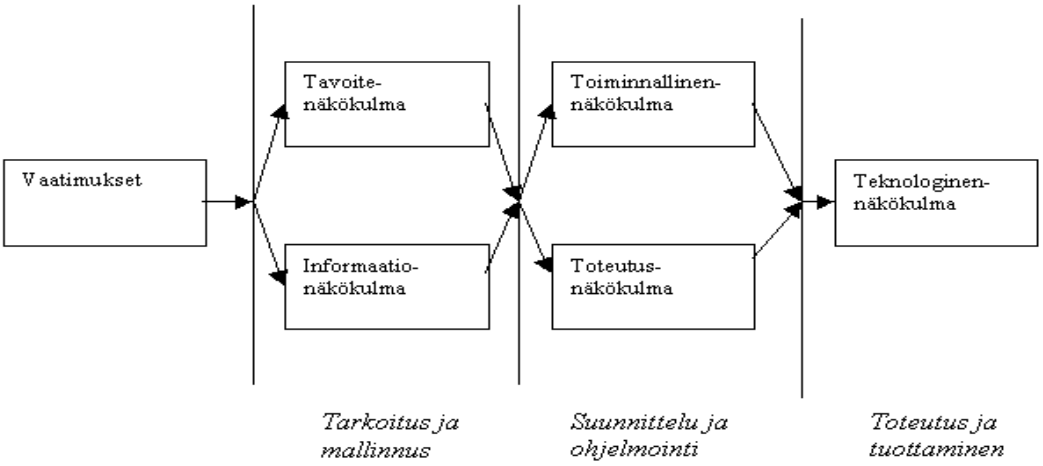
Kuva 1 Yksinkertainen rakenteinen olio [1]

3. Näkökulmat

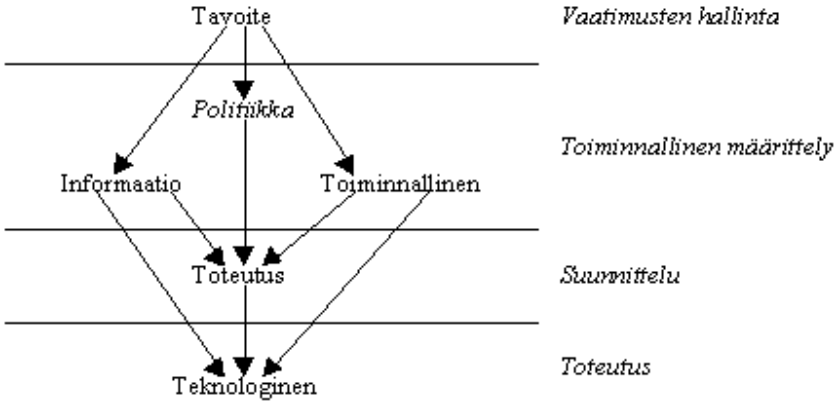
Näkökulmien tarkoitus on luoda kehys, jonka päälle määritellään ODP:n komponentti standardeja. Näkökulmat ovat myös hyödyllinen työkalu järjestelmien suunnittelussa. Hajautettujen ja avoimien järjestelmien määrittelyyn liittyy suuria määriä informaatiota hyvin laajalta alueelta. Näkökulmien avulla tätä informaatiota voidaan jakaa helpommin käsiteltäviin osiin.

Näkökulmia on viisi, ne ovat: tavoite-, informaatio-, toiminnallinen-, toteutus- ja teknologinen näkökulma. Kukin näkökulma on abstraktio, joka tuottaa määrittelyn koko systeemistä, kohdeyleisön kannalta. Kuitenkin päällekkäisyyden määrän tulee olla mahdollisimman pieni.

Kuhunkin näkökulmaan kuuluu oma näkökulma kieli, joka ilmaisee konseptit ja säännöt, jotka ovat oleellisia sille alueelle järjestelmää, jonka näkökulma kuvailee. Käytännössä näkökulmakieli lähinnä määrittelee näkökulman mallintamisessa käytettävän terminologian.



Kuva 2 ODP näkökulmat ja ohjelmistotuotanto [8]



Kuva 3 ODP näkökulmat [5]

3.1 Tavoitenäkökulma

Tavoitenäkökulman tarkoituksena on määritellä järjestelmän tehtävät, vastuualueet ja politiikat. Tavoitenäkökulma käsittelee järjestelmän liiketaloudellista roolia ja aktiviteetteja. Kohde aluetta ei ole määritelty tarkasti, se voi olla yritys, yliopisto tai teknisempi kohde, kuten tiedosto- tai käyttöjärjestelmä. Tavoitenäkökulma keskittyy järjestelmän tarkoitukseen, rajaamiseen ja menettelytapoihin. Tuloksena on malli järjestelmästä ja ympäristöstä, johon järjestelmä sijoittuu sekä näiden välillä vaadittavasta kommunikaatiosta.

Näkökulma sisältämät asiat kuvataan olioiden, yhteisöjen ja roolien avulla. Oliot voivat olla joko aktiivisia, esim. pankin johtaja, asiakas, kassa, tai passiivisia, esim. raha ja tili. Olioiden määrittely syntyy roolien avulla. Roolien käyttäytyminen määritellään politiikkojen avulla. Esimerkiksi pankkiautomaatilla roolina on asiakkaiden palvelu, joka tapahtuu määriteltyjen politiikkojen mukaan. Politiikkoja on sekä sisäisiä, että ulkoisia, kuten lait. Aktiviteetteja ovat esim. rahan nosto ja talletus [5].

Politiikkoja on kolmea tyyppiä: luvat, kiellot ja velvoitteet. Luvat määrittelevät mitä voidaan tehdä, esim. rahaa voi tallettaa pankkitilille. Kiellot määrittelevät mitä ei voi tehdä, esim. tililtä ei saa päivässä nostaa rahaa yli nostorajan. Velvoitteet määrittelevät mitä pitää tehdä, esim. asiakkaan tulee todistaa henkilöllisyytensä nostaessaan rahaa tililtään.

Yhteisö tarkoittaa olioiden ryhmittämistä jonkin tavoitteen saavuttamiseksi, esim. pankin konttori sisältää johtaja-, tili- ja kassa-oliot, tavoitteena pankkipalvelujen tarjoaminen tietyllä alueella.

Tavoitenäkökulma tunnistaa myös federaatiot. Federaatio on oliojoukkojen muodostama yhteisö, federaatiosta on kyse esim. eri pankkien pankkiautomaattien ollessa käytettävissä kunkin yksittäisen pankin asiakkaalle. Federaatiosta määritellään mitkä roolit, politiikat, aktiviteet ja yhteisöt kustakin osallistuvasta järjestelmästä ovat voimassa ja miten ristiriidat ratkaistaan.

Tavoitenäkökulman määrittelemät asiat asettavat rajoituksia ja vaatimuksia, joita muiden näkökulmien tulee noudattaa.

Vastaa kysymyksiin [6]

- “Mikä on järjestelmän tarkoitus?”
- “Mitä palveluja se tarjoaa?”
- “Kuka tarvitsee näitä palveluja?”

3.2 Informaationäkökulma

Informaationäkökulma määrittelee tiedon, jota järjestelmä tarvii, ja sen prosessoinnin semantiikan. Tätä varten määritellään tieto elementit organisaatiossa, niiden virta, varastointi ja prosessi tiedon manipulointiin. Vain informaation semantiikka kuvataan, ei tietorakenteita. Tämä tehdään kolmen skeeman avulla, jotka ovat: invariantti skeema, staattinen skeema ja dynaaminen skeema.

Invariantti skeema
Määrittelee ehdot joiden on aina oltava tosia tietylle informaatiolla, eli olion tilalle, esim. tilin saldon pitää olla aina suurempi kuin nolla.

Staattinen skeema
Määrittelee olion tilan ehtoja, jotka ovat tosia tietyllä hetkellä, siis tiettyjen tapahtumien jälkeen, esim. päivän nostot ovat keskiyöllä 0 mk. Staattisen skeeman tulee noudattaa invariantin skeeman ehtoja.

Dynaaminen skeema
Määrittelee olion tilan sallitut muutokset ympäristön ja olioiden muutosten tapahduttua, esim. x mk:n nosto tililtä vähentää saldoa x mk:aa ja lisää päivän nostojen määrää x mk:aa.

Dynaaminen skeema määrittelee tilan vaihdon välttämättömät ehdot ja loppuehdot, joiden tulee olla voimassa operaation jälkeen. Dynaaminen skeema on aina invariantin skeeman rajoittama, eli tulosten tulee noudattaa invariantissa skeemassa annettuja ehtoja.

Vastaa kysymyksiin [6]

- “Mitä tietoa tarvitaan systeemin palvelujen tukemiseksi?”
- “Mistä tämä tieto tulee ja mihin se menee?”
- “Onko tarpeen tallentaa tieto johonkin?”

3.3 Toiminnallinen näkökulma

Toiminnallinen näkökulma määrittelee järjestelmän käyttäytymisen, abstraktion siitä kuinka asiat tehdään oikein. Näkökulma on olio-opohjainen, se jakaa järjestelmän loogisiin olioihin, jotka kommunikoivat rajapintojen välityksellä. Määriteltävät oliot

voivat infrastruktuuriolioita, esim. meklari, tai sovellusolioita, esim. pankkikonttori. Nämä määriteltävät oliot luovat samalla myös potentiaalisen jaon hajautusta varten. Jokaisella oliolla on yksi tai useampi rajapinta, jotka kaikki kuvataan. Rajapinnan käyttäminen vaatii ensin sen sitomista kohdeolioon (esim. asiakas).

ODP-viitekehys määrittelee kolme vuorovaikutuksen muotoa: toiminnalliset rajapinnat, vuorajapinnat sekä signaalirajapinnat.

Toiminallisissa rajapinnoissa asiakas kutsuu palvelimen operaatioita, käsite toteuttaa palvelin-asiakas -mallin. Toiminnallinen rajapinta koostuu nimetyistä operaatioista, joille on määritelty parametrit, lopetukset ja tulokset. ODP:ssä on kahdenlaisia operaatioita: kysely ja ilmoitus. Kysely palauttaa tuloksen, kun taas ilmoitus käynnistää operaation, eikä palauta tulosta. Kysely pysäyttää asiakkaan suorituksen kunnes operaatio saa paluuarvon. Ilmoitus ei välttämättä pysäytä asiakkaan suoritusta, asiakas ja palvelin voivat toimia sykronisesti.

Vuorajapinnat tarjoavat jatkuvia informaatiovirtoja tuottajan ja asiakkaan välillä, esim. audio- ja video-virtoja tai tiedoston siirto. Useita informaatiovirtoja voidaan yhdistää saman rajapinnan alle, esim. audio- ja video. Vuorajapinnat ovat tarkoitettu multimedia- ja tietoliikennesovelluksille.

Signaalirajapinnat tarjoavat hyvin alhaisen tason kommunikointi toimintoja. Signaalien avulla voidaan siirtää alhaisen tason viestejä, kuten tapahtumia ja keskeytyksiä. Sekä toiminnalliset että vuorajapinnat, jotka ovat siis ylemmän tason rajapintoja, perustuvat signaalirajapintaan.

Toiminnallinen rajapinta määrittelee myös toiminnot, jotka ovat mahdollisia toiminnallisille olioille. Nämä ovat: [8]

- olion luominen ja tuhoaminen
- rajapinnan luominen ja tuhoaminen
- meklaus rajapinnan saamiseksi
- sitominen rajapintaan
- olion tilan luku ja kirjoitus
- operaation käynnistäminen toiminallisesta rajapinnasta

- virran tuottaminen tai kuluttaminen vuorajapinnassa
- signaalin anto tai vastaus signaalin signaalirajapinnassa.

Vastaa kysymyksiin [6]

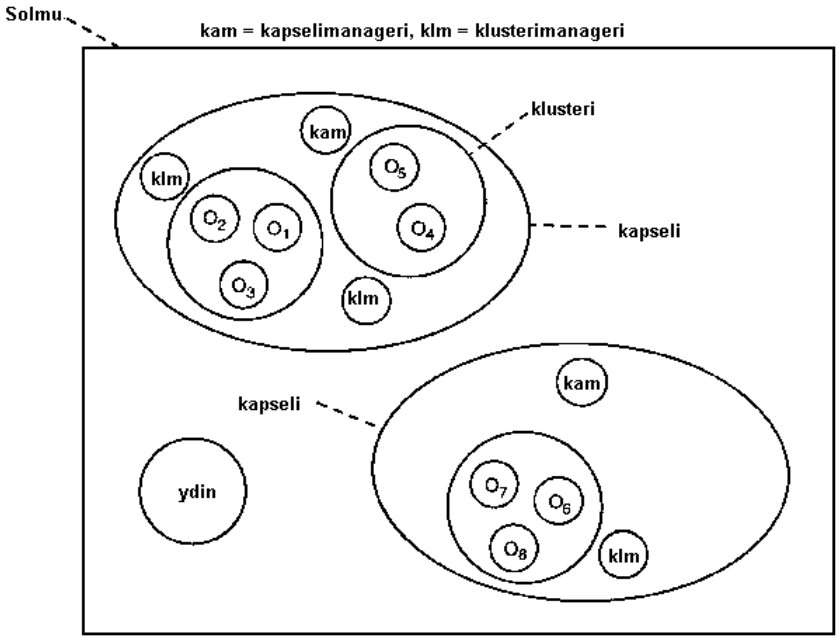
- “Mitä operaatioita on saatavilla?”
- “Kuka suorittaa operaatiot?”

3.4 Toteutusnäkökulma

Toteutusnäkökulma määrittää järjestelmän vaatimat palvelut, jotka infrastruktuurin tulee tarjota. Järjestelmän vaatimat palvelut muotoutuvat muiden näkökulmien määrittelyistä. Palvelut määritellään kommunikoinnin ja järjetelmän resurssien osalta. Kommunikointi mallinnetaan kanavien avulla, jotka koostuvat tyngistä, sidosobjekteista ja protokollaolioista. Järjestelmän resursseja mallinnetaan solmujen, kapselien ja klusterien avulla. Näkökulmassa käytetyt oliot voidaan jakaa kahteen eri kategoriaan: perus toteutusoliot, jotka vastaavat toiminnallisen näkökulman olioita, ja infrastruktuuriolioihin, esim. protokollaolio. Infrastruktuurioliot eivät esiinny toiminnallisessa näkökulmassa.

Solmut, kapselit ja klusterit

Solmu on jokin yksittäinen yksikkö tilassa. Solmu on usein fyysinen tietokone, mutta se voi myös olla jokin kommunikaatio laite. Solmu sisältää ytimen sekä joukon kapseleita. Ydin tarjoaa solmun hallinnointi funktiot, esim. käyttöjärjestelmä. Kapseli omistaa tallennustilaa ja osan solmun prosessointi resursseista. Kapseli vastaakin erässä mielessä perinteistä prosessia. Kapseli sisältää joukon olioita, jotka yhdessä suorittavat jonkin toiminnon. Olioiden kokoaminen samaan kapseliin vähentää niiden välisen kommunikaation kustannuksia. Kapseli voi olla esim. yksittäinen ohjelma. Kapselin sisällä oliot kootaan ryhmiin, joita kutsutaan klustereiksi. Tämä tehdään olioiden manipuloinnin kustannusten alentamiseksi.



Kuva 4 Klusterit, kapselit ja solmut [1]

Kanavat

Kanava vastaa toiminnallisennäkökulman sitomista tai sitomisolioita. Kanavat tarjoavat perus toteusolioiden välisen kommunikointi mekanismin ja kontrolloivat perus toteutusolioiden vaatimia tuntumattomuus funktioita. Kanava luo olioiden välisen sidonnan. Tuntumattomuudet toteutetaan tynkien ja sidosobjektien avulla.

Sidonnat samassa solmussa tai kapselissa olevien olioiden välillä ovat ns. lokaaleja sidontoja, ne toteutetaan systeemi riippuvaisilla mekanismeilla. Eri kapsuleissa tai solmuissa olevien olioiden välisessä kommunikoinnissa tarvitaan sen sijaan hajautettuja sitomisia. Hajautetut sitomiset ovat todennäköisesti kalliimpia, mutta ne mahdollistavat toisaalta hajautuksen edut, esim. kuorman jakamisen muodossa.

Perus toteutusoliot kutsuvat suoraan niitä tukevia tyngiä, jotka vastaavat siirrettävän tiedon sisällöstä. Tyngät pakkaavat ja purkavat siirrettävät parametrit. Sidosoliot vastaavat toteutusolioiden välisen kanavan eheyden säilymisestä. Tämä vaatii konfiguraatio muutosten sekä olio- ja kommunikaatiovirheiden käsittelyn. Protokollaoliot vastaavat itse kommunikaatiosta. Ne tarjoavat palvelemilleen sidosoliolle luotettavan ja riittävän laadukkaan kommunikaatioväylän.

Vastaa kysymyksiin [6]

- “Minkä palveluiden avulla toiminnalliset oliot toimivat?”

3.5 Teknologianäkökulma

Teknologia näkökulma määrittää konkreettiset ohjelmistot ja laitteistot joiden avulla järjestelmä voidaan toteuttaa, sekä testauksen tarvitsemat tiedot. Tämän tavoitteen täyttymiseksi näkökulma identifio mahdolliset laitteistot ja ohjelmistot sekä tutkii niiden hankinta- ja asennus-mahdollisuuksia. Tuloksena on luettelo mahdollisia teknologioista, jotka täyttävät järjestelmän vaatimukset. Laitteistomäärittely määrittelee esim. käytettävien prosessorien määrän, tyypin ja tehon. Ohjelmistomäärittely määrittelee esim. käyttöjärjestelmät ja kehitysympäristöt .[5]

Näkökulma voi myös määritellä kuinka järjestelmä toteutetaan jollakin tietyllä teknologialla. Näkökulmalla on suuri rooli yhteensopivuuden testaamisessa [2]. Yhteensopivuus, joka on avoimille ja hajautetuille järjestelmille välttämätöntä, tarkoittaa määrittelyn ja toteutuksen suhdetta. ODP:ssä yhteensopivuutta testataan referenssi kohtien avulla. Toimintaa, joka ei ole näkyvää refenssikohdassa, ei testata. [6]

Vastaa kysymyksiin [6]

- Kuinka infrastruktuurin palvelut toteutetaan?

3.6 Näkökulmien riippuvuudet toisistaan

Näkökulmien välinen vastaavuus on tehty selkeäksi, jotta rajapintojen tunnistaminen ja tuntumattomuuksien tarjoaminen olisi mahdollista automatisoida. Suurin osa vastaavuuksista on toiminnallisen- ja toteutusnäkökulman välillä.

Muiden näkökulmien tulee täyttää tavoitenäkökulman määrittelemät politiikat. Osa tavoitenäkökulmassa tunnistetuista rooleista, toiminnallisuuksista ja käyttäytymisistä vastaavat suoraan informaationäkökulman skeemamäärittelyjä sekä toiminnallisen näkökulman olioita. Esim. politiikoista tulee rajoituksia skeemoihin, rooleista ja aktiviteeteista tulee dynaamisen skeeman lauseita. Informaationäkökulman määrittelemien rajoitusten tulee päteä toiminnallisessa näkökulmassa. Kaikki toiminnallisen näkökulman oliot näkyvät toteutusnäkökulmassa perus toteutusolioina. Tavoitteena on, että teknologinennäkökulma sisältää mahdollisimman vähän riippuvuuksia muista näkökulmista. Syynä tähän on teknologian nopea kehittyminen.

Teknologisen näkökulman erillisyydellä mahdollistetaan tehokkaampien ja halvempien teknologioiden yksinkertaisempi käyttöönotto. Näin vältetään riskiä riippuvaisuudesta perinnejärjestelmiin [5].

4. Funktiot

ODP määrittelee joukon funktioita joita tarvitaan ODP-viitekehyksen luoman kehysrakenteen tukemiseen. Funktiot jaetaan neljään ryhmään: hallinta-, koordinointi-, tietovarasto- ja turvallisuusfunktiot.

4.1 Hallintafunktiot

Hallintafunktiot ovat: solmun, kapselin, klusterin ja olion hallintafunktiot.

Solmun hallintafunktio

Solmun hallintafunktion avulla luodaan ja hallitaan säikeitä, tarjotaan kellojen ja ajastimien saanti, mahdollistetaan kanavien luonti sekä instantioidaan ja tuhotaan kapseleita [1]. Ydin tarjoaa funktion palvelut [8].

Kapselin hallintafunktiot

Kapselin hallintafunktion huolehtii kapselin klustereiden instantioinnista, tarkistuspisteistä ja palauttamisesta sekä passivoinnista ja tuhoamisesta. Kapselimanageri tarjoaa funktion palvelut.

Klusterin hallintafunktio

Klusterin hallintafunktio tarjoaa palvelut klusterien aktivointii, passivointiin, tarkistuspisteiden asettamiseen ja siirtymiseen. Klusterimanageri tarjoaa funktion palvelut.

Olion hallintafunktio

Olion hallintafunktio huolehtii klusterissa olevien olioiden tarkistuspisteiden asettamisesta ja tuhoaminen. Perus toteustuoliot toteuttavat palvelun.

4.2 Koordinointifunktiot

Koordinointifunktiot ovat:

- tapahtuman tiedonanto -funktio
- tarkistuspiste ja palautus -funktio
- passivointi ja aktivointi -funktio
- ryhmäfunktio

- toisinnusfunktio
- siirtymisfunktio
- sidospisteen viitteen seuraamis –funktio
- transaktiofunktio
- ACID - transaktio–funktio

Tapahtuman tiedonanto -funktio
Funktio tallentaa ja tekee saataviksi tapahtuma historioita. Asiakasoliot rekisteröityvät tiedonanto-funktiolle, jonka jälkeen ne saavat ilmoituksen uuista tapahtuma historioista.

Tarkistuspiste ja palautus –funktio
Funktio päättelee koska klustereille pitää asettaa tarkistuspiste ja käynnistää palautuksen tarkistuspisteestä kun klusteri vikaantuu.

Aktivointi ja passivointi –funktio
Koordinoi klusterin aktivointia ja passivointia.

Ryhmäfunktio
Tarjoaa mekanismit monen osapuolen välisten sitomisten koordinointiin. Funktio kontrolloi mitkä ryhmän jäsenet osallistuvat tiettyyn vuorovaikutukseen, varmistaa yhtenäisen kuva vuorovaikutuksista, varmistaa vuorovaikutusten järjestyksen sekä lisää ja poistaa jäseniä ryhmästä.

Toisinnusfunktio
Toisinnufunktio varmistaa, että ryhmä näkyy muille olioille yhtenä oliona. Toisinnusfunktio on ryhmäfunktion erikoistapaus. Toisinnusfunktiota käyttää siirtymisfunktio.

Siirtymisfunktio
Siirtymisfunktio koordinoi klusterin siirtymistä kapsulista toiseen. Siirtyminen voidaan toteuttaa toisintamisen tai passivoinnin ja aktivoinnin avulla.

Sidospisteen viitteen seuraamis –funktio
Funktio valvoo toteutusrajanpintojen viitteiden siirtämistä eri klustereissa sijaitsevien toteutusolioiden välillä. Valvonnan tarkoituksena on päätellä, koska toteutusrajapintaan liittyvää infrastruktuuria ei tarvita enää. Näin luodaan tuki roskien keruulle järjestelmässä [5].

Transaktiofunktio

Transaktiofunktio koordinoi operaatioita, tavoitteena näkyvyys, palautettavuus ja jatkuvuus. Näkyvyys viittaa siihen kuinka suuria osa operaation välivaiheista on näkyviä toisille operaatioille. Palautettavuus tarkoittaa epäonnistuneen transaktion aiheuttamien tilan vaihtojen perumisia. Jatkuvuus tarkoittaa operaation epäonnistumisen mahdollisuuksia vaikuttaa valmistuneisin operaatioihin.

ACID - transaktio-funktio

ACID – transaktio-funktio on transaktiofunktion erikoistapaus. ODP määrittelee hyvin yleisen transaktiofunktion, standardin tarkoituksena ei ole sitoa toteutusta. Käytännössä ACID tulee olemaan useissa järjestelmissä ainoa tuettu transaktiomekanismi.

4.3 Tietovarasto-funktiot

Tietovarasto-funktio ovat:

- tallennusfunktio
- informaation organisointi –funktio
- uudelleensijoitusfunktio
- tyyppienhallintapalvelu
- meklauspalvelu

Tallennusfunktio

Funktio tallettaa tiedon tietovarastoon. Tietovarastoon voi olla esim. tiedosto, tietokanta, oliokanta [5].

Informaation organisointi -funktio

Funktio hallinnoi tietovarastoa, joka sisältää informaatioskeeman määrittelemän informaation. Lisäksi funktio mahdollistaa informaatioskeeman muutokset sekä kyselyt tietovarastoon ja datan muutokset ja päivitykset.

Uudelleensijoitusfunktio

Funktio hallinnoi tietovarastoa, joka sisältää tiedon rajapintojen sijainnista. Asiakkaat käynnistävät palvelun sen nimen perusteella. Funktio tekee automaattisesti nimen muunnoksen fyysiseksi verkko-osoitteeksi. Siirtymistuntumattomuus tarvitsee funktion hallinnoimaa informaatiota [8].

Tyyppienhallintapalvelu
Palvelu hallinnoi tietovarastoa, joka sisältää tyyppimäärittelyjä ja tyyppien suhteita. Palvelulla on oma rajapinta jokaiselle tallentamalleen tyyppimäärittelylle. Palvelun ylläpitämiä tietoja tarvitaan tyypin tarkastuksessa meklauksessa ja rajapintojen sitomisessa. Tietovarasto tarjoaa funktiot tyyppimäärittelyjen kyselyihin, suhteiden kyselyyn sekä suhteiden tutkimiseen ja johtamiseen tyyppien välille.

Meklauspalvelu
Funktio ottaa vastaa palvelutarjouksia ja välittää niitä vastauksena kyselyihin. Olio julkistaa rajapintansa lähettämällä palvelutarjouksen meklarille. Palveluntarjoaja voi myös poistaa palvelutarjouksensa. Olio, joka tarvii jotain palvelua tekee palvelukyselyn meklarille. Palvelukyselyssä määritellään vaatimukset, jotka sisältävät palvelun tyypin ja siihen liittyvät rajoitteet. Meklari etsii tarjoituista palveluista sellaisen, joka täyttää vaaditut ehdot. Palvelu vastaa tietyssä mielessä puhelinluettelon keltaisia sivuja.

Meklauspalvelu on keskeinen osa ODP-viitemallia [1].

4.4 Turvallisuusfunktiot

Funktio	Tarkoitus
Saantikontrolli	Estää luvattoman rajapintojen käytön.
Turvallisuusauditointi	Valvoo ja kerää turvallisuusinformaatiota.
Autentikointi	Varmistaa olion identiteetin.
Yhtenäisyys	Estää luvattoman datan luomisen, muutoksen ja tuhoamisen.
Luottamuksellisuus	Estää luvattoman tiedon levittämisen.
Ei kiistämistä	Estää olion vuorovaikutukseen osallistumisen kieltämisen.
Avaimen hallinnointi	Hallinnoi kryptausavaimia.

Taulukko 1 Turvallisuusfunktiot [1]

4. Tuntumattomuudet

Tuntumattomuuden avulla kätketään järjestelmän monimutkaisuutta. Käyttäjät ja ohjelmoijat saavat yhtenäisen kuvan järjestelmästä. Kumpienkaan ei tarvi välittää hajautuksen luonteesta ja sen toteutustavasta. Eli ohjelma käyttäytyy samalla tavalla kuin jos se ei olisi hajautettu. Joskus ohjelmoijan tulee toisaalta tietää hajautuksesta, esim. reaaliaikaisia järjestelmiä tehtäessä. ODP:ssa tuntumattomuus on valinnaista, eli ohjelmoija voi itse valita läpinäkyvyyden tason.

Hajautuksen tuntumattomuudet ovat: saanti-, vika-, paikka-, siirtymis-, keskeytys-, uudelleensijoitus-, toisinnus- ja transaktiotuntumattomuus.

Saantituntumattomuus

Saantituntumattomuus on kriittinen useimmissa hajautetuissa järjestelmissä, se on yleensä oletusarvoisesti käytössä. Saantituntumattomuus peittää yhteydessä olevien olioiden saantimetodien erilaisuudet. Tämä tapahtuu peittämällä tiedon esittämistavan ja käynnistys mekanismien erot.

Saantituntumattomuus toteutetaan kanavien avulla. Kanavaan kuuluva tynkä muuntaa vuorovaikutuksen, esim. operaatiokutsu, siirrettäviksi viesteiksi. Tynkä tekee siirron tarvitsemat tiedon konversiot.

Vikatuntumattomuus

Vikatuntumattomuus suojaa oliota sekä itsensä, että muiden olioiden virhetilanteilta. Näin ohjelmoija toimii ihanteellisesssa maailmassa.

Vikantuntumattomuus voidaan toteuttaa kahden eri funktion avulla. Yksi tapa on toisinnusfunktio. Toinen mahdollisuus on tarkistuspiste ja palautus -funktio.

Paikkatuntumattomuus

Piilottaa olion fyysisen sijainnin tunnistettaessa ja sidottaessa rajapintoja. Nimeämiskema on paikkatuntumaton, nimeäminen tapahtuu loogisten nimien avulla. Näin oliot voivat käyttää rajapintoja käyttämättä paikkatietoja.

Saanti- ja paikkatuntumattomuus mahdollistavat yhdessä olioiden vuorovaikutuksen ilman tietoa sijainnista ja saanti tavasta.

Siirtymistuntumattomuus

Peittää oliolta järjestelmän sille tekemät paikan muutokset. Syynä olion siirtämiseen voi olla esim. kuormituksen jako tai viiveen vähentäminen.

Tuntumattomuuden toteuttaa siirtymistuntumattomuusfunktio. Ennen siirtymistä oliolle asetetaan tarkistuspiste ja olio tuhotaan. Siirtymisen jälkeen uudelleensijoitustuntumattomuus löytää olion [5].

Keskeytystuntumattomuus

Peittää olion itsensä sekä muiden olioiden passivoinnin ja uudelleen aktivoinnin, jossa olion tilan tulee säilyä. Toiminta on analoginen käyttöjärjestelmän virtuaalimuistiin. Ominaisuus mahdollistaa resurssien jaon kun järjestelmä ei pysty tarjoamaan prosessointi-, tallennus- ja kommunikointi-palveluita jatkuvasti. Tuntumattomuus riippuu passivointi ja aktivointi –funktiosta.

Uudelleensijoitumistuntumattomuus

Peittää vuorovaikutuksessa mukana olevan olion uudelleensijoittumisen muilta olioilta, jotka ovat siihen yhteydessä rajapinnan sidonnan kautta. Kun olio sijoitetaan uudelleen kanava täytyy asentaa tämän uuden sijainnin mukaan.
Uudelleensijoitusfunktio toteuttaa tuntumattomuuden.

Toisinnustuntumattomuus

Peittää ryhmän molemminpuolisesti, toiminnallisesti yhteensopivien olioiden käytön rajapinnan toteutuksessa. Järjestelmä replikoi olioita eri paikoissa ja tarjoaa näin vikatuntumattomuuden, lisää saatavuutta sekä parempaa suorityskykyä nopeamman tiedon saannin muodossa. Toisinnusfunktio toteuttaa tuntumattomuuden.

Transaktiotuntumattomuus

Peittää mekanismit joiden avulla oliokoukon toimintoja koordinoidaan yhtenäisyyden saavuttamiseksi. Sisältää skeduloinnin, monitoroinnin ja toipumisen.

5. Yhteenveto

Avoin hajautettu tietojenkäsittely (ODP) tarkoittaa järjestelmiä, jotka tukevat hajautusta huolimatta järjestelmän osien heterogeenisyydestä sekä organisaatioiden rajoista. ODP-viitekehyksen standardisoinnin tavoitteena on kehittää runko järjestelmien ja niiden infrastruktuurin määrittelyyn. Kyseessä ei ole toteutusstandardi. ODP-viitekehyksen tarjoama runko luo arkkitehtuurin, joka tukee

hajautusta, keskinäistä yhteistyötä, keskinäistoimivuutta ja siirrettävyyttä. ODP viitekehys sisältää näkökulmat, funktiot ja tuntumattomuudet. Näkökulmien avulla määritellään funktiot, jotka toteuttavat tuntumattomuudet.

ODP-viitemallin määrittelemät näkökulmat jakavat ohjelmistojen määrittelyyn viiteen osaan. Näin helpotetaan järjestelmien monimutkaisuuden hallintaa. Näkökulmat määrittelevät kehysrakenteen jota käytetään myös muiden ODP standardien teossa.

ODP-viitemallin funktiot toteuttavat tuntumattomuudet sekä hajautetun järjestelmän vaatimia peruspalveluita. Tuntumattomuuksien avulla voidaan järjestelmän hajautus peittää sen käyttäjiltä.

Lähteet

[1] Gordon Blair & Jean-Bernard Stefani. Open Distributed Processing and Multimedia. Addison-Wesley, 1997.

[2] ISO/IEC JTC1. Informatio Techonology – Open Systems Interconnection, Data Management and Open Distributed Processing. Reference Model for Open Distributed Processing. Part 1: Overview, 1996. IS10746-1

[3] ISO/IEC JTC1. Informatio Techonology – Open Systems Interconnection, Data Management and Open Distributed Processing. Reference Model for Open Distributed Processing. Part 2: Foundations, 1996. IS10746-1

[4] ISO/IEC JTC1. Informatio Techonology – Open Systems Interconnection, Data Management and Open Distributed Processing. Reference Model for Open Distributed Processing. Part 3: Architecture, 1996. IS10746-1

[5] Ian Joyner. Open Distributed Processing: Unplugged! Saatavana: http://homepages.tig.com.au/~ijoyne/ODPUnplugged.html

[6] Lea Kutvonen. Architectures for Distributed System: Open Distributed Processing Reference Model. HeCSE Workshop on Emerging Techonologies in Distributed Systems pöytäkirja. Lammi, Tammikuu 1998. Helsinki University of Technology, Research Reports Sarja A Numero 50.

[7] Peter Linington. An ODP Approach to the Development of Large Middleware Systems. Toisen kansainvälisen IFIP työkonferenssin pöytäkirja aiheesta: Distributed applications and interoperable systems, Helsinki Kesäkuu 1999. Kluwer Academic Publishers.

[8] Kerry Raymond. RM-ODP: Introduction. ICODP´95. Saatavana: "http://www.dstc.edu.au/research news/odp/ref model/".