TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

ROOPE PARVIAINEN

TUOTERUNKOON PERUSTUVAN TIETOVARASTORATKAISUN KEHITTÄMINEN

Diplomityö

Tarkastaja: professori Samuli Pekkola
Tarkastaja ja aihe hyväksytty
Talouden ja rakentamisen tiedekuntaneuvoston kokouksessa 5. marraskuuta 2014

TIIVISTELMÄ

ROOPE PARVIAINEN: Tuoterunkoon perustuvan tietovaraston kehittäminen
Tampereen teknillinen yliopisto
Diplomityö, 95 sivua, 7 liitesivua
Maaliskuu 2015
Tietojohtamisen diplomi-insinöörin tutkinto-ohjelma
Pääaine: Tiedonhallinta
Tarkastaja: professori Samuli Pekkola

Avainsanat: ohjelmistotuoterunko, tuotteistaminen, tietovarastointi

Tuoterunko tuo ohjelmistokehitykseen omat erityispiirteensä. Tietovarastojen toteutus on monesti täysin räätälöity toteutus asiakkaan vaatimuksista, kun taas ohjelmistotuoterunko pyrkii vastaamaan useiden samanaikaisten asiakkaiden tarpeisiin. Tutkimuksessa ongelmana oli tuoterunkoon perustuvan tietovarastoratkaisun kehittäminen. Tavoitteena oli luoda ymmärrys siitä, mitä vaiheita ja mitä huomioonotettavia seikkoja kehitysprosessissa on. Toisena tavoitteena oli selvittää, miten ratkaisusta saadaan laajennettava, räätälöitävä ja uudelleenkäytettävä.

Tutkimusotteena oli sulautettu yhden tapauksen tapaustutkimus. Tutkimusongelmaa lähestyttiin tarkastelemalla tavoitteen kannalta kahta keskeistä aihealuetta, jotka olivat tuoterunkoon perustuva ohjelmistokehitys ja tietovarastointi. Tämä tehtiin kahdessa vaiheessa siten, että ensin teorian pohjalta muodostettiin keskeisin käsitteistö molemmista aihealueista kirjallisuuskatsauksen avulla. Käsitteistön avulla muodostettiin tarkistuslista, joka toimi pohjana tutkimuksen jälkimmäiselle osiolle. Seuraavaksi suoritettiin empiirinen osuus, jossa etsittiin kohdeorganisaatiosta vastauksia edellisiin aihealueisiin toteuttamalla teemahaastatteluita. Lopuksi näiden kahden osion keskeisimpiä löydöksiä analysoimalla etsittiin yhtäläisyyksiä ja tehtiin tulkintoja tutkimusongelmaan vastaamiseksi. Tulokseksi saatiin prosessimalli ja lista prosessin vaiheissa huomioonotettavista seikoista.

Tuoterunkoon perustuvan tietovarastonratkaisun kehittämisen katsottiin koostuvan kolmesta erillisestä vaiheesta: tietotarpeiden määritys, tiedon mallinnus sekä toteutus ja testaus. Viimeisen vaiheen katsottiin jakautuvan kahdeksi alaprosessikseen. Tärkeimpinä poimintoina kehittämisestä nähtiin olevan riittävä tuntemus sovellusalasta, jolle ratkaisua kehitetään. Tämä luo edellytykset sille, että tietovaraston tietomalli saadaan muodostettua sellaiseksi, että se palvelee mahdollisimman monen yrityksen tietotarpeita. Tietovaraston tietomalli tulee mallintaa dimensionaalisessa muodossa ja nähdä riittävästi vaivaa mahdollisimman kattavien dimensioiden muodostamiseksi. Mallinnettavan datan on oltava dimensioissa ja faktoissa matalimmalla tarkkuustasolla. Lähdedatan käsittelemiseen ja tietovarastoon viemiseen toteutettava ETL-prosessin suunnittelussa ja toteutuksessa tulee soveltaa erilaisia suunnittelumalleja ja -ratkaisuja. Sekä tietomallin että ETL-prosessin toteutuksien tulee olla teknologiariippumattomia.

ABSTRACT

ROOPE PARVIAINEN: Developing a data warehouse based on software product line
Tampere University of Technology
Master of Science Thesis, 95 pages, 7 Appendix pages
March 2015
Master's Degree Programme in Information and Knowledge Management
Major: Business Information Management
Examiner: Professor Samuli Pekkola

Keywords: software product line, domain engineering, data warehousing

Software product line brings its own special characteristics to software development. Developing a data warehouse is primarily a custom implementation whereas the goal of software product line is to offer a comprehensive solution that satisfies requirements of multiple concurrent customers. The goal of this study was to find out how to develop a data warehouse based on software product line. The first objective was to explain what phases there are and what needs to be taken into account when developing data warehouse based on software product line. Second objective was to solve how the solution can be designed extensible, customizable and reusable.

The research approach for this study was an embedded case study with single case. Answering the research problem was made by examining two separate themes: software product line development and data warehousing. The study was carried out in two phases. First the concepts and checklist of aforementioned themes were formed based on literature review. Next an empirical study was carried out in the form of semi-structured interview. The findings of the separate materials were then analyzed in parallel and as a result interpretations were made to answer the research problems. The result was established in the form of process model with list of issues that need to be taken into account.

Development of data warehouse based on software product line was seen to be constituted from three stages: requirement specification, conceptual modelling, realization and testing. The key highlight from the reflection in the first place was comprehensive knowledge about the field of operation to which data warehouse solution is implemented. It is a prerequisite to formulate the data warehouse conceptual model that satisfies the requirements of multiple concurrent customers. Furthermore, the model needs to be modelled dimensionally and shaping the dimensions has to be made thoroughly in bottom-up fashion. Data in the dimensions and facts need to be stored in the lowest level of granularity. ETL-process that handles and transfers the source data to data warehouse also needs to be designed by using different design patterns and architectural styles to provide extensibility and customizability. Both data warehouse model and ETL-process implementations need to be technology-independent to be reusable.

ALKUSANAT

Tähän diplomityöhön kulminoituu viimeisen reilun viiden vuoden uurastus Tampereen teknillisessä yliopistossa. Työn tekeminen on ollut henkisesti haastava mutta samalla opettava kokemus, jonka varrelle on mahtunut tunteita epäuskosta onnistumisen riemuun. Nyt on aika lausua kiitos sen ansaitseville ihmisille.

Ensinnäkin haluan kiittää arvokkaista kommenteista työni ohjaajaa professori Samuli Pekkolaa, joka suunnittelun alkuvaiheissa ohjeisti olennaisimpien asioiden huomioimisessa ja ammattitaidollaan sekä ripeillä vastauksillaan ohjasi työtä oikeaan suuntaan sen toteutuksen aikana. Kiitokset osoitan Leanware Oy:ssä työskenteleville työni ohjaajalle Mika Valkoselle ja Anssi Tikalle mielenkiintoisen aiheen antamisesta ja kakista kannustavista kommenteista kirjoitusprosessin aikana.

Erityiskiitokset kuuluvat puolisolleni Tiinalle, joka on tukenut ja rohkaissut minua opintojeni eri vaiheissa sekä auttanut diplomityössä oikolukemalla kirjallista tuotostani. Kiitän myös omia sekä Tiinan vanhempia henkisestä ja taloudellisesta tuesta varsinkin opiskeluiden alkuvaiheessa. Ilman teitä tämä koulutus olisi tuskin ollut mahdollista.

Opiskelukavereistani haluan kiittää erityisesti Samua ja Petteriä, joiden kanssa olemme kyenneet saumattomaan yhteistyöhön useiden harjoitustöiden ja vapaa-ajanaktiviteettien parissa.

Viimeisen kiitoksen kohdistan koripallojaosto Pompulle, jonka aktiiviset jäsenet ovat olleet olennaisesti TTY:llä viettämänäni aikana vaikuttamassa positiivisesti niin opiskeluuni kuin sen ulkopuoliseen elämään. Yhdessä on nähty ja koettu kaikenlaista, joka ei tule koskaan unohtumaan.

Tampereella, 17.2.2015

Roope Parviainen

TERMIT JA NIIDEN MÄÄRITELMÄT

Tuoterunkoartefakti	Alustan- tai tuotekehityksen jonkin aliprosessin seurauksena syntynyt tuotos. Näihin kuuluvat esimerkiksi vaatimukset, arkkitehtuuri, ohjelmistokomponentit ja testit. (Pohl et al. 2005)
Tuoteportfolio	Joukko tietyn tyyppisiä tuotteita, joita yritys tarjoaa. (Pohl et al. 2005)
Kääntäjä	Ohjelma, joka muuttaa korkean tason ohjelmointikielen lähdekoodin abosoluuttiseksi, kone- tai assemblykoodiksi. (Daintith & Wright 2008)
Linkkaus	Operaatio, jossa apuohjelma yhdistää käännetyt ohjelmamoduulit yhdeksi kokonaisuudeksi selvittäen sisäiset viitteet moduulien välillä. (Daintith & Wright 2008)
Jäljitettävyys	Ohjelmistoartifaktien ominaisuus, jonka avulla voidaan määrittää eri siihen liittyvä historiatieto, jolloin saavutetaan tuoterunkoon perustuvassa kehityksessä systemaattinen ja johdonmukainen uudelleenkäyttö sekä tuoteperheen evoluutio. (Pohl et al. 2005)
Konfiguraatio	Konfiguraatiolla tarkoitetaan komponenttien ja niiden vaihtoehtojen järjestämistä ja asetusta siten, että saadaan osittain tai kokonaan toteutettua ohjelmistotuote (Deelstra et al. 2005)
Referenssiarkkitehtuuri	Referenssiarkkitehtuuri on tietyn kaltainen arkkitehtuuri, joka helpottaa ymmärtämään ja selittämään, mitä jossakin ratkaisussa esiintyy. Se ei kuvaa mitään konkreettista järjestelmää, joten se toimii ohjeena muodostettaessa jotain uutta, kuten tietovarastoa. (Ariyachandra & Watson 2010, s. 201; Koskimies & Mikkonen 2005, s. 33)

Ohjelmistoarkkitehtuuri	Järjestelmän komponenttien muodostama rakenne, jossa kuvataan komponenttien välisistä suhteista ja ympäristöstä sekä periaatteet, jotka ohjaavat sen suunnittelua ja toteutusta (Maier et al. 2001). Ohjelmistoarkkitehtuurin rooli tuoteperheissä on kuvata tuotteiden yhteneväisyyksiä ja eroavaisuuksia sekä tarjota yhteinen rakenne. (Svahnberg & Bosch 2000).
Ohjelmistokomponentti	Arkkitehtuurin itsenäinen ohjelmistoyksikkö, joka toteuttaa tietyn osa-alueen toiminnallisuuden ja tarjoaa palvelujaan rajapintojensa kautta. Komponentit toteutetaan usein olioviitekehyksiä käyttämällä. (Svahnberg & Bosch 2000; Koskimies & Mikkonen 2005)
Datan atomisuus	Termillä tarkoitetaan mahdollisuutta jakaa jokin tietty resurssi pienempiin osiin. (Daintith & Wright 2008)
ETL	Poiminta, transformaatio ja lataus (engl. Extract-Transform-Load) on prosessi, jossa tietovarastoinnissa lähdejärjestelmien dataa prosessoidaan ja siirretään tietovarastoon. (Ponniah 2010)
Tietomalli	Tietomallilla tarkoitetaan abstraktia kuvausta jostakin todellisen maailman kohteesta, josta tallennetaan tietoa tietokantaan. Tietomalleja ovat esimerkiksi hierarkkinen tai relationaalinen. (Daintith & Wright 2008)
Normalisointi	Normalisoinnilla tarkoitetaan menetelmiä, joilla taataan, että tietomalli täyttää vaatimukset tarkkuuden, yhdenmukaisuuden, yksinkertaisuuden, ei-toisteisuuden ja stabiiliuden suhteen. (Imhoff et al. 2003)
Granulariteetti	Granulariteetilla tarkoitetaan datan tarkkuustasoa tietovarastossa. Mitä alhaisempi se on sitä enemmän se vastaa operatiivisten tietojärjestelmien sisältämää transaktiodataa. (Imhoff et al. 2003)
Vieras-/pääavainviittaus	Relaatiotietokannoissa vieras- ja pääavainviittauksella tarkoitetaan jonkin alkion viittaamista toiseen. Tämä saavutetaan upottamalla viitattavan alkion pääavain viittaavan alkion vierasavaimeksi. (Silvers 2008, s. 62)

Datakomero	Datakomero (engl. data mart) on käyttötarkoitukseltaan tietovaraston kaltainen datan tallennuspaikka, mutta se on kooltaan huomattavasti pienempi ja sen sisältämä data liittyy rajoitettuun määrään aihealueita, kuten myynti tai markinointi. (Watson 2002)
OLAP	Tietovarastoihin kohdistuvaa anaalyyttista prosessointia (engl. OnLine Analytical Processing), kuten monimutkaisia kyselyoperaatioita. Porautuminen (engl. drill-down) on esimerkki tällaisesta operaatiosta. (Chaudhuri & Dayal 1997)
PAC-malli	Ohjelmiston suunnittelumalli, joka muodostaa hierarkkisen rakenteen, jossa on yhdessä toimivia agentteja. Kukin agenteista sisältää esitys-, abstraktio- ja hallintakomponentin, joista kaksi ensimmäistä piilottaa agentin toiminnallisuuden ja kolmas hoitaa agenttien välisen kommunikaation. (Buschmann et al. 2001)
Mikroydin-malli	Mikroydin-malli on käyttöjärjestelmätieteestä alun perin liikkeelle lähtenyt sovellusten suunnittelumalli, jossa ohjelmiston ydin sisältää olennaisimmat palvelut ohjelmiston toiminnalle ja tarjoaa kytkentämahdollisuuden uusille toiminnallisuuksille. Ydintä laajennetaan sisäisillä ja ulkoisilla palveluilla. (Buschmann et al. 2001)
Kerrosarkkitehtuuri	Järjestelmän kokonaisrakenteen määrittelevä arkkitehtuurityyli tai suunnittelumalli, jossa rakenne koostuu tietyn abstrahointiperiaatteen mukaisista kerroksista. Ylemmän tason komponentit toteutetaan alemman tason komponentteja hyödyntäen. (Koskimies & Mikkonen 2005, s. 126)

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

ETL	Engl. Extract, Transform, Load. suom. poiminta, tranformaatio, lataus
OLAP	Engl. OnLine Analytical Processing. suom. datan analyyttista prosessointia
PAC	Engl. Presentation, Abstraction, Control. suom. esitys, abstraktio, hallinta. Sovellusarkkitehtuurin suunnittelumalli
UML	Engl. Unified Modelling Language. kuvaustekniikka
UX	Engl. User eXperience. suom. käyttäjäkokemus
WMS	Engl. Warehouse Management System, suom. varastonhallintajärjestelmä

1. JOHDANTO

Tuoteperheet ja tuoterunkoon perustuva ohjelmistokehitys on yksi viimeisten vuosikymmenien merkittävimmistä kehityssuunnista ohjelmistotekniikan alalla. (Boehm 2006) Tuoterunkoon perustuvaan ohjelmistokehitykseen siirtymisessä on sitä suorittavalle yritykselle useita eri hyötyjä, joista merkittävimpiä ovat kustannusten aleneminen pitkällä aikavälillä, ohjelmiston laadun ja luotettavuuden paraneminen sekä markkinoille viemisen nopeutuminen. Hyödyt perustuvan laajan mittakaavan uudelleenkäyttöön, joka vaatii yrityksiltä kuitenkin mittavia etukäteispanostuksia. (Linden et al. 2007, s. 3)

Tietovarastot tarjoavat tehokkaan pääsyn organisaation eri lähdejärjestelmien tuottamaan historiadataan, jonka perusteella voidaan tehdä päätöksiä yrityksen toimintaan ja sen ympäristöön liittyen. (Inmon et al. 2010) Tietovarastot mahdollistavat erilaisten trendien seuraamisen ja helpottavan ennusteiden tekemisessä. (Chaudhuri & Dayal 1997) Tietovaraston kehittämisen voidaan tulkita olevan asiakkaan tietotarpeista ja vaatimuksista riippuvainen toteutus, joka vaihtelee toteutusten välillä (ks. esimerkiksi Kimball & Ross 2002; Inmon 2005). Tämä tutkimus tarkastelee mahdollisuutta rakentaa tietovarastoja soveltaen tuoterunkoon perustuvan ohjelmistokehityksen paradigmaa ja saavuttaa kehityksessä tavoiteltavat hyödyt. Tutkimuksessa tilaajana toimii teollisuuden ja logistiikan ohjelmistotalo Leanware Oy, jonka osa kehittämistä ohjelmistoista noudattaa ohjelmistotuoterunkoajattelua.

1.1 Tutkimuksen tausta

Tutkimus tehdään logistiikan ja teollisuuden alan ohjelmistotalo Leanware Oy:n toimeksiannosta. Leanware kehittää asiakkailleen ohjelmistoratkaisuja logistiikan ja valmistavan teollisuuden yrityksille pyrkimyksenään tehostaa yrityksen toimintaa.

Leanware on tehnyt merkittäviä panostuksia viimeisen kahden vuoden aikana siirtyäkseen perinteisestä yksittäisten ja toisistaan riippumattomien ohjelmistojen kehityksestä tuoterunkoon perustuvaan ohjelmistokehitykseen. Yritykseen on tehty muutamia opinnäytetöitä liittyen juuri siirtymisestä tuoterunkoa hyödyntävään ohjelmistokehitykseen. Logistics-liiketoimintayksikössä ensimmäisenä tuoterunkoon perustuvana tuotteena on varastonhallintajärjestelmiä (engl. Warehouse Management System, WMS), josta on otettu käyttöön lukuisia installaatioita asiakasyrityksissä alati kiihtyvällä tahdilla. Nyt tuoterungon piiriin ollaan tuomassa uusi tuotetyyppi ja tämä diplomityö toimii osaselvityksenä tässä prosessissa.

Yrityksessä on entuudestaan kokemusta tietovarastojen toimittamisesta asiakasyrityksille korkeakouluympäristössä sekä logistiikan alan yrityksissä. Uusi liiketoimintamalli vaatii kuitenkin uudenlaista lähestymistapaa ratkaisujen kehittämiseen, sillä myös tietovarastojen kehityksessä halutaan noudattaa mahdollisimman pitkälle tuoterunkoajattelua. Tietovarastointiprojektit ovat pääasiassa yksilöllisiä toteutuksia yrityksen tietotarpeiden tyydyttämiseen, joten tavoitteena on selvittää, onko tietovaraston perustuminen yhteiseen tuoterunkoon mahdollista.

1.2 Tutkimuksen tavoite ja tutkimuskysymykset

Yrityksen nykytilanteeseen heijastaen on muodostettu tutkimusongelma, jota tukemaan on valittu alaongelmia.

Päätutkimusongelma:

- Miten tuotteistetaan yhtenäiseen tuoterunkoon perustuva tietovarastoratkaisu?

Alaongelmat:

- Mitä vaiheita kuuluu ohjelmiston tuotteistamiseen?
- Millainen on tietovaraston kehittämisprosessi?
- Mitä asioita kehityksessä tulee huomioida, että ratkaisusta tulee laajennettava, räätälöitävä ja uudelleenkäytettävä?

Tutkimuksen tavoitteena on luoda lista seikoista, jotka tulee ottaa huomioon tietovarastoratkaisun suunnittelussa ja toteutuksessa. Lisäksi luodaan kuvaus siitä, mitä vaiheita on kehitettäessä tietovarastoratkaisua, joka perustuu ohjelmistotuoterunkoon ja jonka sisältämä data ja ratkaisun lopullinen laajuus on ennen kehitystä vain osittain tiedossa. Tutkimuksen tavoite on siis pyrkiä laajentamaan yrityksen näkökulmaa siitä, miten toimia tämänkaltaisessa prosessissa.

1.3 Tutkimuksen näkökulma ja rajaukset

Tutkimuksen näkökulma on tuoterunkoon perustuvan tietovarastoratkaisun kehittämisprosessissa, sen vaiheissa ja vaiheiden sisällössä. Tutkimusongelman piiriin kuuluu myös teknisiä ratkaisuja. Tämä aiheuttaa sen, että tutkimuksen käsittelyyn sisällytetään jonkin verran prosessia tukevaa teknistä aihesisältöä, joka esitellään melko yleisellä tasolla.

Tuotteistaminen voidaan sekoittaa tuoterunko- tai alusta-ajatteluun perustuvan ohjelmistokehitykseen siirtymisenä, mutta tämän tutkimuksen kannalta se on väärä näkökulma. Tuoteperheiden kirjallisuudessa esiintyy runsaasti kirjoituksia siirtymisprosessista ja yrityksen maturiteetista kyseisessä paradigmassa. Kohdeyritys on hyvin pitkällä jo tässä

prosessissa, joten siirtymistä ohjelmistotuotteiden liiketoimintaan tai nykytilan arviointia ei huomioida tutkimuksessa. Tuotteistamisella tässä tutkimuksessa tarkastellaan siis sitä ohjelmistokehityksen prosessia, jolla tuoterunkoon perustuva ohjelmistotuote saadaan aikaan.

Tuoterunkoon perustuva kehitys ei toimi tyhjiössä vaan siihen vaikuttaa voimakkaasti yrityksen liiketoimintaympäristö. Tuoteperheen liiketoimintanäkökulma rajataan kuitenkin tutkimuksen ulkopuolelle, joten ulkopuolelle jäävät asiakkaan ja markkinoiden vaikutus tuotteistamiseen. Lisäksi kysymys siitä, missä vaiheessa tulisi tuotteistaa tai millainen ajoitus markkinoille tuloon olisi sopiva, ovat tutkimuksen fokuksen ulkopuolella. Tutkimuksessa ei myöskään arvioida, mille markkinoille tuotteistus suunnataan tai minkälaisia myynnillisiä keinoja tullaan käyttämään. Yleisessä tuotteistamisen teoriassa esiintyvät ideointi- tai innovointivaiheet, markkina-analyysin teko, tuoteportfolion arviointi ja tuotteen kaupallistamisaspektit rajataan myös ulkopuolelle. Edellä mainitut vaiheet on yrityksessä jo otettu huomioon ja päätökset niiden suhteen on tehty, joten niitä ei ole tarpeen käydä läpi tässä tutkimuksessa.

Tuoterunkoon perustuva kehitys muodostuu kahdesta erillisestä prosessista: alustankehityksestä ja tuotekehityksestä. Pääkysymyksen kannalta keskeinen on alustankehitys, joka mielletään nimenomaan tuotteistamiseksi. Prosessit perustuvat toisiinsa voimakkaasti, mutta tutkimuksen kannalta vähemmän olennainen vaihe tuotekehitys jätetään rajauksen ulkopuolelle.

Tietovaraston teoriaa käsitellään pääasiassa kehittämisprosessin kannalta, joten tekniset yksityiskohdat eivät kuulu tämän tutkimuksen aihepiireihin. Myös tietovaraston toteutusprojektin johtamisen käsittely rajataan pois. Leanware Oy:ssä on vakuuttavaa osaamista tietojärjestelmä- ja tietovarastoprojekteista, joten ei ole tarpeellista keskittyä niihin tämän tutkimuksen puitteissa. Toteuttamisprojekti rajataan lisäksi koskemaan vain toteutuksen valmistelua ja suunnittelua koskevaksi, joten käyttöönotto sekä ylläpito- ja kasvuvaihe jäävät tutkimuksen ulkopuolelle.

Tietovarastoinnin kanssa olennaista aihealuetta liiketoimintatiedon hallintaa ja siihen liittyvää datan analysointia ja sen sovelluksia ei myöskään käsitellä tässä tutkimuksessa, vaikka ne esiintyvätkin usein yhdessä tietovarastoinnin kirjallisuuden kanssa.

1.4 Tutkimuksen metodologiset valinnat

Metodologisten valintojen esittelyllä kuvataan tutkimuksen suorittamisen taustalla olevat oletukset, jotka ohjaavat tutkimuksen tutkimusotteen ja edelleen tiedonkeruumenetelmien valintaa. Valinnan tekemistä tukemaan on kuvailtu tutkimusmetodologista kenttää hieman laajemmin, jotta hahmotettaisiin tämän tutkimuksen asema paremmin tieteen alalla.

1.4.1 Tieteenkäsitys

Muodostettaessa käsitystä tieteestä keskeisin kysymys on, mihin tieteellisen tiedon hankkimisen perustuu ja miten tiedon tieteellisyys perustellaan. Tällöin puhutaan tieteenkäsityksestä, jonka merkittävimpiä valtakäsityksiä ovat hermeneutiikka ja positivismi. (Olkkonen 1994, s. 26; 28) Saunders et al. (2006) luettelevat näiden lisäksi realismin ja pragmatismin.

Hermeneuttisen tai interpretivistisen tieteenkäsityksen mukaan positivismin tekemät lainmukaiset yleistykset ilmiöistä karsivat kompleksiset todellisen maailman tulkinnat. Hermeneuttisessa tieteenkäsityksessä tutkija on osana tutkimusta, sillä tutkimus on tutkijan omiin arvoihin ja käsityksiin sitoutunutta. Aineistonkeruulle luonteenomaista on, että kerätään pieniä otoksia, joita tutkitaan syvällisesti. (Saunders et al. 2006, ss. 118-119) Kuten mainittu, hermeneutiikka tarkastelee tutkimusaineistoa tutkijan ymmärryksen pohjalta, joten on varsin luonnollista, että kaksi eri lukijaa ymmärtää tutkimuksen antaman informaation ja merkityksen toisistaan poikkeavalla tavalla. Siksi tutkijan on tärkeää ilmaista aineistonsa tiedeyhteisössä ymmärrettävissä yhteneväisellä tavalla. (Olkkonen 1994, s. 36)

Positivistisen tieteenkäsityksen yksi keskeisimmistä ajatuksista on, että tutkimuksen on oltava tutkijasta riippumaton ja toistettavissa siten, että kuka tahansa pääsee samoihin tuloksiin käyttämällä samaa materiaalia ja menetelmää. Pohjana on olettamus tiedon perustumisesta todettuihin tosiasioihin, kuten ulkoisiin havaintoihin ja ”koviin faktoihin”. (Olkkonen 1994, s. 35)

Liiketaloustieteen alalle sijoittuvassa tutkimuksessa nähdään harvoin puhtaasti joko hermeneuttisia tai positivistisia tutkimusotteita. Ratkaistavien ongelmien luonne tai saatavilla oleva aineisto pakottaa toisinaan tinkimään positivistista tai hermeneuttisista tutkimusperiaatteista. (Olkkonen 1994, s. 53)

1.4.2 Tutkimustyyppi

Tutkimuksen tyypin yleisimmin käytetty lajittelu on määrällinen eli kvantitatiivinen ja laadullinen eli kvalitatiivinen tutkimus. (Hirsjärvi et al. 2000, ss. 124 -125) Termit kvalitatiivinen ja kvantitatiivinen ovat saaneet kritiikkiä muun muassa niiden luomasta mielikuvasta vastakohtaisina strategioina toisilleen. Mielekkäämpää on asettaa tämäntyyppiset dikotomiat eräänlaiselle jatkumolle ja korostaa tutkimustyyppien eroja tutkimuskäytänteissä ja periaatteellisissa kysymyksissä (esimerkiksi epistemologioissa). (Hirsjärvi et al. 2001, ss. 124 - 125) Seuraavaksi havainnollistetaan näitä eroja

Kvantitatiiviselle tutkimukselle ominaista on luotettava ja ”kova” lähdeaineiston luonne. Aineistosta pyritään varmistamaan olemassa olevan teorian pitävyyttä. (Hirsjärvi et al. 2001, s. 124) Kvantitatiivinen tutkimus, jonka juuret ovat luonnontieteissä, korostaa yleispätevän syyn ja seurauksen lakeja. Todellisuuden katsotaan muodostuvan objektii-

visesti todettavista tosiasioista. Tutkimuksen keskiössä ovat hypoteesit sekä johtopäätökset aikaisemmista tutkimuksista ja teorioista. (Hirsjärvi et al. 2001, s. 129) Hypoteeseja testataan ja osoitetaan oikeiksi tai ne hylätään, mikä johtaa lisäteorioiden syntymiseen tai lisätutkimukseen. (Saunders et al. 2006, s. 113).

Kvalitatiivinen tutkimus on puolestaan luonteeltaan strukturoimaton ja lähdeaineiston luonne on useasti rikasta ja syvällistä. Lähtökohtana laadullisessa tutkimuksessa on todellisen elämän kuvaaminen. Siinä todellisuutta ei voi kvantitatiivisen lähestymistavan tavoin pirstoa osiin vaan tarvitaan ihmisen tulkintaa kokonaisuuden muodostamiseksi. (Hirsjärvi et al. 2001, s. 152; Saunders et al. 2006, s. 115) Vaikka kvalitatiivinen ja kvantitatiivinen tutkimusstrategia nähdään toistensa vastakohtina, toimivat ne usein toisiaan täydentävinä suuntauksina. (Hirsjärvi et al. 2001, s. 125) Huomion arvoinen seikka on, että kvalitatiivisella tutkimuksella on omia erityispiirteitä ja traditioita riippuen tieteenalasta, jolle tutkimusta tehdään. (Hirsjärvi et al. 2001, s. 153) Tässä tutkimuksen tekemistä on siis tarkasteltava omassa kontekstissaan, joka on tässä tapauksessa liiketaloustiede. Tämän tutkimuksena tutkimusstrateginen valinta sijoittuu kvalitatiiviseen suuntaan johtuen tutkimusongelmasta ja tavoitteesta kartoittaa tiettyä ilmiötä.

1.5 Tutkimuksen rakenne

Tämän tutkimuksen voidaan katsoa rakentuvan kolmesta eri osa-alueesta. Ensimmäisenä osiona on kirjallisuuskatsaus, joka rakentaa teoreettisen viitekehyksen tutkimuksen empiiriselle osuudelle. Teoriaosuus jakaantuu edelleen kahteen osioon, joista ensimmäisessä käsitellään ohjelmistotuoteperheisiin ja tuoterunkoon perustuvan kehityksen teoriaa. Tämän jälkeen on vuorossa tietovarastoinnin teoria. Molempien aihepiirien teoriassa lähtökohtana on niistä aikaisemmin julkaistut tutkimukset ja muu kirjallisuus. Tavoitteena on muodostaa kokonaiskuva tutkimuksessa käsiteltäville aihepiireille.

Teoriaosuuden jälkeen vuorossa on empiirinen osuus, joka toimii rinnakkaisena ja täydentävänä aineistomuotona tässä tapaustutkimuksessa. Empiria jakautuu teorian tapaan kahteen aihepiiriin: ohjelmistotuoterunkoon perustuvaan kehitykseen ja tietovarastointiin.

Kolmannessa osiossa esitetään tutkimuksen metodologisia valintoja ja niiden taustalla olevia olettamuksia, jotta lukija voi arvioida tutkimuksen luotettavuutta. Myös kohdeorganisaatiota kuvataan omassa alaluvussaan. Tämän jälkeen kootaan tutkimuksen tärkeimmät tulokset, empiirisen tutkimuksen pohjalta. Seuraavaksi tuloksia ja teoriaa analysoimalla tehdään pohdintoja. Yhteenvedossa tutkimuksen pääkohdat nivotaan yhteen ja pyritään tulkitsemaan tulosten merkitystä. Lopuksi tarkastellaan tutkimuksen validiteettia ja tehdään ehdotelmia mahdollista jatkotutkimusta varten.

2. OHJELMISTOTUOTEPERHEET

Tässä luvussa perehdytään ohjelmistotuoterunkoon perustuvan kehityksen teoriaan. Ensin johdatellaan aiheeseen kertomalla yleisesti ohjelmistotuoteperheistä, minkä jälkeen siirrytään käsittelemään siihen liittyvää kehitysprosessia. Tämän jälkeen käydään läpi kehitysparadigmassa olennaista muunneltavuuden hallintaa ja lopuksi kehitysprosessiin liittyvän tuotteenhallintaan liittyviä aihepiirejä.

2.1 Tuoteperhe ja tuoterunko

Tässä alaluvussa kuvataan yleisesti ohjelmistotuoteperheisiin ja -tuoterunkoon liittyvät periaatteet ja havainnollistetaan keskeisimmät erot perinteiseen ohjelmistokehitykseen.

2.1.1 Yleinen kuvaus

Ohjelmistotuoteperheet ja tuoterunkoon perustuva kehitys ovat yksi merkittävimmistä kehityssuunnista ohjelmistotekniikan alalla viimeisten vuosikymmenien aikana. (Boehm 2006) Tuoteperheen määritelmä Koskimiehen ja Mikkosen (2005, s. 160) mukaan kuuluu: se on kokoelma rakenteeltaan ja toiminnaltaan samankaltaisia ohjelmistotuotteita, jotka tyypillisesti kattaa yrityksen omalle toiminta-alueelleen tekemät ohjelmistotuotteet. Kehitys tapahtuu hyödyntämällä tuoterunkoa, joka koostuu tuoteperhearkkitehtuurista, joukosta uudelleenkäytettäviä komponentteja ja ohjelmistotuotteita, joita voidaan järjestellä halutulla tavalla. (Svahnberg & Bosch 2002, s. 147; Northrop et al. 2002) Tuoterunkoon perustuvalla kehityksellä (engl. Software product line engineering) tarkoitetaan paradigmaa, jossa kehitetään sovelluksia hyödyntämällä ohjelmistoalustoja ja massakustomointia. Ohjelmistoalustalla viitataan rakenteeseen, jonka mahdollistaa kokonaisten tuotteiden kehityksen uudelleenkäytettävien komponenttien avulla. (Pohl et al. 2005, s. 14) Tämä rakenne on tuoterunko.

Tarkemmin määriteltynä tuoterunko on uudelleenkäytettävä infrastruktuuri, joka puolestaan on joukko järjestelmiä ja rajapintoja. Järjestelmät ja rajapinnat muodostavat yhteisen rakenteen. Alustasta tai tuoterungosta voidaan kehittää ja tuottaa johdannaisia tuotteita tehokkaasti. (Meyer & Lehnerd 1997) Tuoterunkoon kuuluu esimerkiksi ohjelmistoarkkitehtuuri, uudelleenkäytettävät ohjelmistokomponentit, vaatimusmäärittelyt, testitapaukset, dokumentaatio ja muut määrittelyt, budjetit ja prosessikuvaukset (Northrop 2002, s. 32), joita kutsutaan artefakteiksi. Bosch (2002, ss. 264-266) tyypittelee artefaktit seuraavasti: tuoterungon arkkitehtuurikuvaukset, yhteiset komponentit ja tuotteet, jotka on toteutettu käyttämällä yhteisiä artefakteja. Artefaktilla tarkoitetaan osittaista järjestelmän osittaista ratkaisua. (Geyer & Becker 2002, s. 2)

Kirjallisuudessa käsitteet tuoteperhe (engl. product family) ja tuotelinja (engl. product line) esiintyvät monesti virheellisesti toistensa synonyymeina (ks. esimerkiksi Linden 2002, Chastek et al. 2004), mutta kysymys on hiukan eri asioista. (Linden 2002; Chastek et al. 2004) Nimitysten eroa ja epäloogista käyttöä selittää paradigmaa tutkivien pohjoisamerikkalaisen ja eurooppalaisen tiedeyhteisön rinnakkaiset tutkimukset, jotka saatiin yhdistettyä vasta vuonna 1996. (Linden 2002) Termien tuotelinja ja tuoteperhe välinen yhteys voidaan nähdä siis esimerkiksi Geyerin ja Beckerin (2002) mukaan tulkitsemalla niiden tarkastelevan samaa asiaa eri näkökulmasta. Tuoteperheellä he tarkoittavat joukkoa järjestelmiä, jotka on kehitetty yhteisistä artefakteista eli järjestelmiä, jotka pohjautuvat samaan uudelleenkäytettävään infrastruktuuriin. Termi tuotelinja puolestaan kuvaa joukkoa järjestelmiä, joilla on yhteistä toiminnallisuutta. (Geyer & Becker 2002, s. 2) Koskimies ja Mikkonen (2005, s. 160) täydentävät, että tuotelinjaa voidaan nimittää tuoterunkoarkkitehtuuriksi. Tuotelinjaan kuuluvalla tuotteella tarkoitetaan tuoteperheen arkkitehtuuriin ja sen komponentteihin pohjautuvia tuoteinstansseja, joihin saattaa liittyä myös tuotekohtaisia laajennoksia. (Svahnberg & Bosch 2000; Gurp et al. 2001, s. 46) Tässä tutkimuksessa käytetään tuotelinjasta eli tuoterunkoarkkitehtuurista nimitystä tuoterunko. Tuoterungolla tämän kohdeorganisaation kontekstissa tarkoitetaan kaikille yrityksen varastonhallintajärjestelmäinstallaatioille yhteistä arkkitehtuuria. Lisäksi termillä tuoteperhe viitataan samaan tuoterunkoon perustuviin ohjelmistoihin, joilla kaikilla ei välttämättä ole yhteistä toiminnallisuutta, kuten varastonhallintajärjestelmä ja tietovarastoratkaisu, mutta pohjautuvat samaan infrastruktuuriin.

Viimeisen kymmenen vuoden aikana kasvava trendi on ollut, että tuoterunkoon pohjautuvia tuotetyyppejä on kaksi tai useampia. Tuoterunko voi siis koostua useammasta tuotelinjasta. Tämänlaisesta tilanteesta käytetään esimerkiksi nimitystä monituotelinjat (engl. multi product line). (Holl et al. 2012) Tässä tutkimuksessa toimeksiannon tehnyt yritys pyrkii kyseiseen tilanteeseen ja tavoittelee sitä kautta saavutettavia hyötyjä.

Jotta tuoteperheiden luomisessa ja tuotteiden rakentamisessa onnistuttaisiin, tiettyjä periaatteita tulee noudattaa. Yksi näistä periaatteista on muunneltavuuden hallinta (engl. variability management). Sen mukaan yksittäiset järjestelmät nähdään variaatioina yhteisestä teemasta. Varianssi tai muunneltavuus tulee tehdä eksplisiittiseksi ja sitä tulee hallita systemaattisesti. (Linden et al. 2007, s. 7) Toinen periaate on nähdä tuoteperheet strategisena liiketoiminnan muotona, jossa kehitys nähdään hyödyttävän koko markkinoita eikä pelkästään yksittäistä asiakasta. Tuoteperheiden taustalla olevan tuoterungon, on tekniseltä puoleltaan tuettava tätä ajatusta siten, että se hyödyntää yksittäisten tuotteiden välisiä yhteneväisyyksiä ja ymmärtää eroavaisuudet. (Linden 2007, s. 8)

Tuoteperheiden kehityksen onnistumisen yksi merkittävimmistä vaatimuksista on myös riittävä tuntemus sovellusalueesta, jolle tuoteperhettä ollaan tekemässä. Vain henkilöt, jotka tuntevat markkinat ja sen asiakkaat, kykenevät määritellä tuoteperheessä esiintyvät yhteiset piirteet ja vaihtelevuudet luotettavasti, jotta sopiva alusta saadaan kehitettyä.

Määrittelyssä epäonnistuminen johtaa lisäkustannuksiin myöhemmissä kehityksen vaiheissa. (Pohl et al. 2005, ss. 17-18)

2.1.2 Erot perinteiseen ohjelmistokehitykseen

Merkittävin ero tuoterunkoon perustuvassa kehityksessä verrattuna perinteiseen yksittäisten järjestelmien kehitykseen on periaatteellinen muutos ajattelutavassa järjestelmää ja projektia kohtaan. Tuoteperheiden tapauksessa ei keskitytä vain yksittäisiin projekteihin, jotka seuraavat toisiaan ja lähtevät liikkeelle aina alusta alkaen. Näkemys on strategisempi ja kohdistuu kokonaista toimialaa kohtaan. (Linden et al. 2005, s. 6) Bosch (2000, s. 5) lisää, että perinteisissä ohjelmistoprojekteissa päätökset priorisoidaan usein sen perusteella, että toteutus on saatava sovittuun deadlineen mennessä valmiiksi asiakkaalle. Tuoterunkoon perustuvassa kehityksessä tuote pyritään luomaan siten, että sen avulla voidaan tyydyttää mahdollisimman kattavasti yhden liiketoiminta-alueen tarpeet ja se noudattaa riittäviä laatuvaatimuksia.

Käsitteenä ohjelmistojen uudelleenkäyttö on yhtä vanha kuin ohjelmistotekniikan ala, mutta verrattuna yksittäisten järjestelmien ohjelmistokehitykseen siitä voidaan mainita kahdenlaisia eroavaisuuksia. Ensinnäkin perinteisessä lähestymistavassa koodin uudelleenkäyttö on ollut luonteeltaan opportunistista, tilapäistä ja mittakaavaltaan pienimuotoista. (Bosch 2000, s. 8) Tuoteperheissä uudelleenkäyttö nähdään strategisena seikkana ohjelmistokehityksessä ja sen suunnitteluun panostetaan. Toiseksi perinteisessä ohjelmistokehityksessä uudelleenkäyttö on kohdistunut yksinomaan ohjelmakoodiin, mikä on tuoteperheissä vain osa uudelleenkäyttöä. Tuoterunkoon perustuvassa kehityksessä uudelleenkäytön kohteena ovat kaikki kehitysartefaktit, joita tuoterunkoon kuuluu ja jotka ovat relevantteja tuoterungon kehityksen eri elinkaaren vaiheissa. (Linden et al. 2007, ss. 5-6; Northrop 2002, s. 33)

Edellä mainituista seikoista johtuen tuoteperheiden kehitys vaikuttaa organisaation rakenteeseen ja prosessiin, jolla ohjelmistoja tuotetaan. Tuoterunkoon perustuva ohjelmistokehitys jakautuukin kahteen erilliseen prosessiin: alustankehitykseen ja tuotekehitykseen (engl. domain engineering & application engineering) (Pohl et al. 2005, s. 24; Koskimies & Mikkonen 2005, s. 164) Karkea jako vastuiden suhteen on, että alustankehitysprosessissa kehitetään uudelleenkäytettävä tuoterunko, jossa on otettu huomioon mahdolliset yhteneväisyydet ja eroavaisuudet tuotteiden välillä yleisten vaatimusten pohjalta. Tuotekehitysprosessi puolestaan vastaa tuotteiden johtamisesta tuoterungosta. Siinä hyödynnetään tuoterunkoon toteutettua muunneltavuutta ja varmistetaan muunneltavuusvaatimuksien oikea sitomisaika tuotekohtaisten vaatimusten perusteella. (Pohl et al. 2005, s. 20; Northrop et al. 2007) Vaikka molemmat prosessit ovat tuoterunkoon perustuvan kehityksen kannalta olennaisia, käsitellään tässä tutkimuksessa tarkemmin vain alustankehitysprosessi.

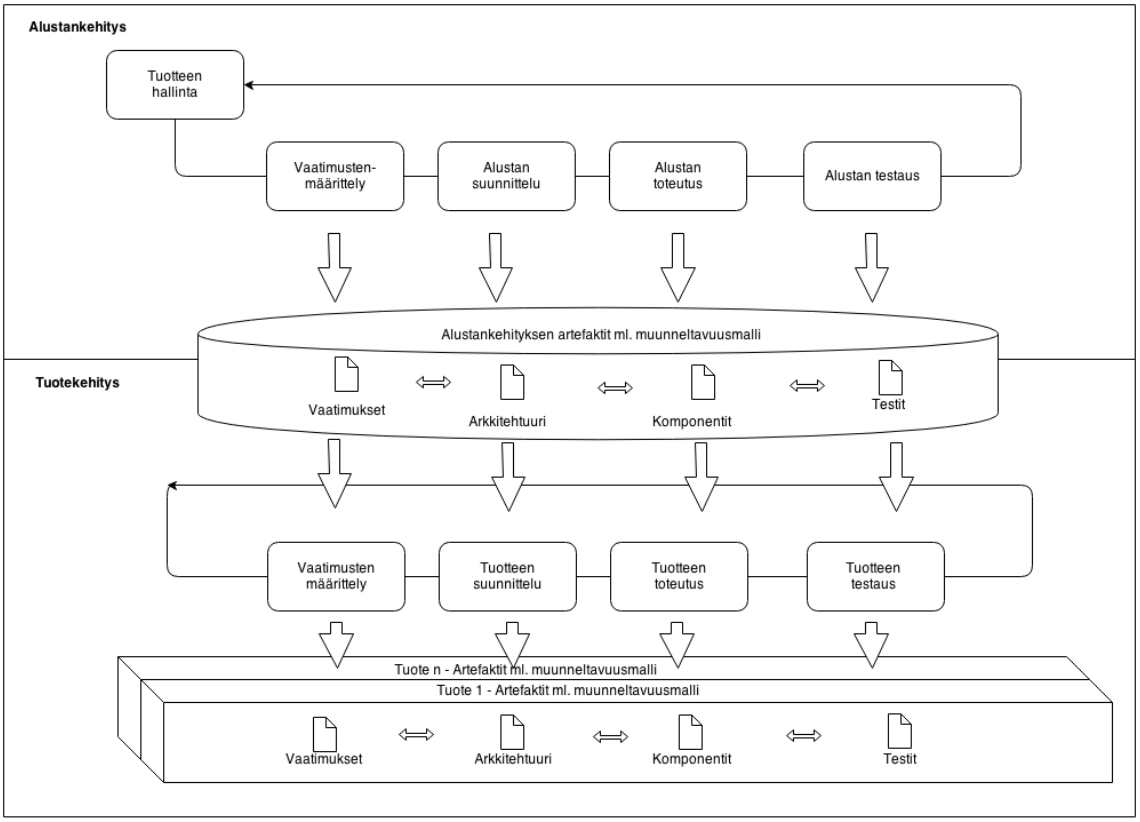
Yhtenä merkittävimpänä erona perinteiseen ohjelmistokehitykseen on muunneltavuuden hallinta, joka kuuluu osaksi alustankehitysprosessia. Koska tuoteperheitä kehitettäessä pyritään uudelleenkäytettävään tuoterunkoon, joka tukee mahdollisimman laajaa skaalaa tuotteita, tulee sitä kehittää kokonaisuuden kannalta. Tällä tarkoitetaan, on huomioitava muunneltavuus yksittäisten tuoteperheen tuotteiden välillä. Muunneltavuutta on kolmea tyyppiä: yhteiset, muunneltavat ja tuotespesifit ominaisuudet. (Linden et al. 2007, s. 8) Muunneltavuusvaatimukset määräävät, mitkä ominaisuudet tuotteessa voivat vaihdella, missä rajoissa vaihtelu tapahtuu ja missä vaiheessa muunnelma kiinnitetään tietyllä tavalla. (Koskimies & Mikkonen 2005, s. 164) Yhteiset ja muunneltava ominaisuudet kuuluvat alustankehitysprosessin vastuualueelle, kun taas tuotespesifit ominaisuudet ja niiden kehitys kuuluvat pelkästään tuotekehitysprosessin vastuulle. Muunneltavuuden hallinta on olennainen osa tuoterungon elinkaaren kaikkia vaiheita aina projektin rajauksen määrittelystä, käyttöönottoon ja tuoteperheen evoluutioon. (Linden et al. 2007, s. 9)

Bosch (2000, s. 5) mainitsee edellisten lisäksi, että perinteisessä ohjelmistokehityksessä ei välttämättä oteta huomioon ohjelmiston kehittymistä. Koska keskittyminen on yksittäisen järjestelmän toimittamisessa, ei suunnitella välttämättä sitä, että järjestelmää täytyy ylläpitää ja kehittää hyvin pitkän ajan. Ohjelmistotuoteperheissä tämä aspekti on välttämätön, sillä ohjelmistotuoterungon elinkaari on pidempi kuin tavallisen ohjelmiston. (Bosch 2000, s. 5)

2.2 Ohjelmistokehitys tuoterunkoa käytettäessä

Ohjelmistotuoterunkoon perustuvassa kehityksessä on tunnistettavissa kaksi erillistä prosessia: alustankehitys ja tuotekehitys (Pohl et al. 2005). Northrop et al. (2002) käyttävät prosesseista termejä ydinartefaktien kehitys (engl. core asset development) ja tuotekehitys (engl. product development) sekä mainitsevat näiden lisäksi omaksi prosessikseen johtamisen, joka takaa resursseja ohjelmistokehityksen aktiviteeteille ja koordinoi sekä valvoo toimintaa niin teknisellä kuin organisaation eri tasoilla.

Alustankehitys- ja tuotekehitysprosessien kunkin prosessin alavaiheen seurauksena syntyy välituotteita, joita kutsutaan artefakteiksi. Kuva 1 havainnollistaa kehitysprosessikokonaisuutta, osaprosessien alavaiheita, vaiheiden tuloksena syntyviä artefakteja sekä vaiheiden ja artefaktien kytkeytymistä muihin prosessin osiin. Tutkimuksessa käytettäessä termiä tuoterunko viitataan kuvassa 1 alustan- ja tuotekehityksen välissä olevaan sylinterin muotoiseen kuvioon.



Kuva 1. *Ohjelmistokehitys tuoterunkoa käytettäessä (mukailtu lähteestä Pohl et al. 2005)*

Keskeistä prosessin kannalta on huomata niin sanotun muunneltavuusmallin (engl. variability model), jossa kuvataan tuoterungon yhteiset ja vaihtelevat ominaisuudet, olevan yksi keskeisin kehitysartefakti ohjelmistotuoteperheessä. (Pohl et al. 2005, s. 58) Koskimies ja Mikkonen (2005, s. 165) eivät käytä termiä muunneltavuusmalli, mutta luettelevat dokumentaatioon kuuluvaksi käsitemallin, muunneltavuusvaatimukset sekä yhteiset vaatimukset. Käsitemallia käytetään yhteisen kommunikaation välineeksi ja sen avulla varmistetaan eri osapuolten käyttämän käsitteistön yhdenmukaisuus. Yhteiset ja muunneltavuusvaatimukset puolestaan kuvaavat tuoterunkoon toteutettavaa muunneltavuutta. (Koskimies & Mikkonen 2005, s. 165) Muunneltavuusmallin ensimmäisen määrittelyn jälkeen se kulkeutuu alustankehityksen kaikkien vaiheiden lävitse, jolloin sitä tarkennetaan aina yhä matalammalle abstraktiotasolle aina teknisiin suunnittelupäätöksiin asti. (Pohl et al. 2005, s. 58)

Prosessista voidaan vielä mainita yleisellä tasolla vielä, että kukin vaihe sekä alustan- että tuotekehityksessä on voimakkaasti kytköksissä prosessia edeltävään ja seuraavaan vaiheeseen. Vaiheet tarjoavat palautetta taaksepäin ja seuraavana oleva vaihe vaatii erilaisia syötteitä edeltävältä vaiheelta. Alustankehitys on aina kytköksissä tuotekehityksessä vastaavaan vaiheeseen, jolle se tarjoaa lähtökohdan esimerkiksi arkkitehtuurisuunnitelman muodossa. Tuotekehityksessä ilmenneitä parannusehdotuksia saattaa kulkeutua myös alustankehityksen suuntaan, jolloin ominaisuuksia voidaan ottaa osaksi

tuoterunkoa. (Pohl et al. 2005) Tästä johtuen kuvan 1 yksisuuntaiset nuolet ovat hieman harhaanjohtavia, sillä todellisuudessa tieto ja artefaktit kulkevat molempiin suuntiin.

2.2.1 Alustankehitys

Alustankehityksen ensimmäisenä ja prosessin kokonaisuuden kannalta tärkeimpänä vaiheena on **tuotteenhallinta**. Tuotteenhallinta voidaan tuoteperheiden tapauksessa määritellä alustankehityksen aliprosessiksi, jossa kontrolloidaan tuoterungon ja siitä johdettujen ohjelmistojen kehitystä, tuotantoa ja markkinointia. (Pohl et al. 2005, s. 167) Northrop et al. (2007) sijoittavat tämän aktiviteetin erilliseksi mutta vahvasti kytköksissä alustankehityksen kanssa olevaksi kokonaisuudeksi. Toisaalta tämä jaottelu rajaa alustankehityksen ulkopuolelle projektinhallinnalliset ja ulkopuolisten rajapintojen tai sidosryhmien vaikutukset. Helfereich et al. (2006, s. 66) sekä Weerd et al. (2006) toteavat, että tuotteenhallinta ohjelmistotuoteperheissä on saanut liian vähän huomiota ja aihepiirin kirjallisuus on hajautunutta. Monesti tuotteenhallinta on organisoitu yrityksissä markkinointifunktion yhteyteen, mikä ohjelmistotuoteperheiden tapauksessa on riittämätön lähtökohta. Lähestymistapa ei ota huomioon sitä, miten ohjelmistotuotteet tuotetaan ja miten tekniset mahdollisuudet saadaan parhaiten hyödynnettyä. Tuotteenhallinnan tulisi huomioida samanaikaisesti sekä markkinoiden että tekniset syötteet yhdenvertaisesti. Niinpä tuotteenhallinnan olennaisin tehtävä on toimia eräänlaisena välittäjänä edellä mainittujen osa-alueiden välillä. (Helfereich et al. 2006, ss. 66-67).

Joka tapauksessa tuotteenhallintafunktion suorittaman markkinoiden ja organisaation tarkastelun perusteella tuotteenhallinnassa määritellään tuoterungon tuoteportfolio ja siihen liittyvä tiekartta (engl. roadmap), jossa esitetään keskeisimmät yhteiset ja muuttuvat ominaisuudet suunnitelluille tuotteille sekä aikataulun niiden toimittamiseksi asiakkaalle tai markkinoille. Tiekartan muodostamista kutsutaan myös rajauksen määrittelemiseksi (Pohl et al. 2005, s. 164; 167) Tuoteportfoliossa yrityksen kehittämien uusien tuoteperheen tuotteiden kehityshankkeita ja olemassa olevien tuotteiden kehitystä arvioidaan ja priorisoidaan. (Pohl et al. 2005, s. 169) Portfolion hallinta tarkoittaa käytännössä päätöksentekoa olemassa olevien ja uusien tuotteiden suhteen markkinatrendien seurannan, tuotekehitysstrategian tai kumppanuuksien ja ulkopuolisten sopimuksien tarkastelun perusteella. Päätöksenteon ja suunnittelun seurauksena saadaan syötteitä, joiden perusteella muodostetaan tuoterungon rajaus. (Weerd et al. 2006).

Tuoterungon konkreettisella rajauksella (engl. scope) tarkoitetaan kuvausta siitä, mitä tuotteita tuoterunkoon sisältyy tai joita tuoterunkoon on mahdollista sisällyttää. Yksinkertaisimmillaan rajaus sisältää listan tuotenimistä, mutta tyypillisesti siinä luetellaan yhteiset ja eroavat ominaisuudet. (Northrop et al. 2007) Tasoja, jolla rajausta voidaan tehdä, on tunnistettavissa edelliseen viitaten kolme: tuoteportfolio-, sovellusalue- ja komponenttitaso. Tuoteportfoliotasolla tehtävä rajauspäätös määrittelee, mitkä tuotteet voidaan sisällyttää tuoterunkoon sekä mitä vaatimuksia tuotteilla täytetään. Sovellusaluetasolla tarkoitetaan yhtenäisiä toiminnallisuusryhmiä ja siihen liittyvässä rajauk-

sessa päätetään, mitkä ominaisuudet kuuluvat sovellusalueeseen ja jotka tulisi olla osana tuoterunkoa. Kyse on toisin sanoen sovellusalueen rajojen määrittelystä. Komponenttitasolla tunnistetaan ne komponentit eli toiminnalliset toteutukset, jotka kehitetään uudelleenkäyttöä varten. (Schmid 2000, s. 514) Rajaukseen pitää kiinnittää erityistä huomiota, sillä mikäli rajaus on liian väljä, tuoteperheen jäsenten ominaisuudet vaihtelevat liian paljon eikä tuoterunko pysty tyydyttämään tuotekehityksen tarpeita riittävän tehokkaasti. Toisaalta liian tiukka rajaus heikentää tuoterungon kasvupotentiaalia. (Northrop et al. 2007; Schmid 2000, s. 514) Tuotteenhallinnassa pitää huomata, että tuotteen rajaus muuttuu ajan myötä esimerkiksi markkinoiden olosuhteiden tai yrityksen suunnitelmien myötä. Rajauksen kehittyminen on lähtökohta tuoterungon kehittämisessä tai kasvattamisessa. (Northrop et al. 2007)

Olennaista rajauksen määrittelyssä on sovellusalueen syvällinen tuntemus ja ymmärtäminen. (Northrop et al. 2007) Monilla nuorilla ohjelmistotuoterunkoon perustuvaa kehitystä harjoittavilla yrityksillä portfolio käsittää vain yhden tuotteen, kun taas suuremmilla yrityksillä portfolioon kuuluu useita tuotteita. Niin myös rajauksen määrittämisen vaativuus vaihtelee. (Weerd et al. 2006) Haasteellista rajauksen määrittelyssä on, että siinä tulee ymmärtää monia erilaisia aloja, niiden tyypillisimpiä ongelmia ja ratkaisuja näihin ongelmiin. Niinpä tuotteenhallinnassa mukana olevilla henkilöillä tulee olla taito kommunikoida ja esittää tämä tavoilla, joita kaikki alustankehityksessä olevat sidosryhmät ymmärtävät. (Northrop et al. 2007)

Seuraava olennainen osa alustankehitysprosessia on sovellusalueen **vaatimustenhallinta** (engl. domain requirements engineering). Se käsittää toimenpiteet, joiden tuloksena saadaan aikaan ja dokumentoidaan tuotteen tuoterungon tuoteperheen yhteiset ja vaihtelevat ominaisuudet, toisin sanoen muunneltavuusvaatimukset. Tuotteenhallinnan aikaan saama rajaus toimii tässä lähtökohtana. (Pohl et al. 2005, s. 25) Vaatimusten hallinnan lähtökohta on ulkoisilta sidosryhmiltä kerätyt ohjelmiston toiminnallisuuteen liittyvät vaatimukset, jotka muunnetaan itse tuoterungon vaatimuksiksi poistamalla päällekkäisyydet, yhdistämällä samaa kokonaisuutta käsittelevät vaatimukset ja kirjoittamalla ne puhtaaksi yhtenäiseksi dokumentaatioksi. (Weerd et al. 2006) Vaatimuksien lähteitä ovat esimerkiksi olemassa olevat tuotteet, eri sidosryhmät, kilpailijan tuotteet ja raportoidut virheet. (Pohl et al. 2005, s. 201) Dokumentaatio tuoteperheen vaatimuksista muodostaa tärkeän tuoterunkoartefaktin. Vaatimusten lisäksi siinä kuvataan tuoteperheen tukemat tuotteet, niiden ominaisuudet ja rajoitteet. (Northrop et al. 2007) Vaatimukset kehittyvät yhdessä rajauksen kanssa rinnakkain. (Clements 2002, s. 180)

Vaatimustenhallinnan keskeisimpiä aktiviteetteja ovat tuoterunkoa luotaessa yhteisten piirteiden analyysi ja muunneltavuusanalyysi sekä muunneltavuuden mallinnus. (Pohl et al. 2005, s. 199) Yhteisten piirteiden analyysissa lähdetään liikkeelle tunnistamalla kaikkien tuoteperheeseen mahdollisesti kuuluvien tuotteiden vaatimukset. Päällekkäiset vaatimukset ovat hyviä kandidaatteja yhteisiksi vaatimuksiksi. (Pohl et al. 2005, ss. 202 - 206) Yhteisten vaatimusten analysoinnin ja määrittelyn jälkeen ne dokumentoidaan,

jolloin saadaan käsitemalli, jonka perusteella määritellään tuoteperheen sovellusalueen keskeisin käsitteistö. Näin varmistetaan, että kaikki osapuolet ymmärtävät sovellusalueen samalla tavalla. (Koskimies & Mikkonen 2005, s. 165) Muunneltavuusanalyysi puolestaan pyrkii määrittelemään variaatiopisteet eli kohdat, jossa muunneltavuutta esiintyy, ja vaihtoehdot eli variantit näissä pisteissä. Muunneltavuutta pyritään tunnistamaan rakenteessa, toiminnallisuudessa, käyttäytymisessä tai laatuaspekteissa. Jokainen muunneltavuusvaatimus ei välttämättä kelpaa variaatiopisteeksi, sillä se saattaa vaatia suunniteltavalta arkkitehtuurilta liian monimutkaista ratkaisua. Tällöin päädytään täysin tuotekohtaiseen ratkaisuun. Variaatioanalyysissä tulee tuntea kaikki tuoteperheeseen mahdollisesti kuuluvat tuotteet ja niiden vaatimukset, jotta lopputulos onnistunut. Variaatioanalyysin perusteella syntyvä kuvaus on monesti graafinen esitys tuoterungon variaatiopisteistä, niihin lukeutuvista varianteista ja mahdollisesti niiden sidonta-ajasta. (Pohl et al. 2005, ss. 202 - 206) Vaatimusten hallinnan piiriin kuuluu myös kontrolloida analyysin perusteella syntyneiden vaatimusten välistä vuorovaikutusta ja riippuvuuksia. Tämä tapahtuu monesti erilaisin kuvauskäytännöin, kuten piirremallin avulla. (Maßen & Lichter 2004, s. 169)

Seuraava vaihe alustankehitysprosessissa on **alustan suunnittelu** (engl. domain design). Alustan suunnittelussa päätetään, miten sovellukset rakennetaan teknisessä mielessä. (Pohl et al. 2005, s. 218) Alustan suunnittelun keskeisin syöte ovat vaatimusmäärittelyvaiheesta syntyvät alustan vaatimukset, jotka käsittävät yhteiset sekä muunneltavat vaatimukset, sekä muunneltavuus- tai sovellusalueen käsitemalli. (Pohl et al. 2005, s. 26; Koskimies & Mikkonen 2005, s. 165) Alustan suunnittelun tuloksena syntyy referenssiarkkitehtuuri, jossa on otettu huomioon vaatimusten muunneltavuus edellä mainittujen artefaktien perusteella. (Pohl et al. 2005, s. 218) Toisin sanoen suunnitellaan yhteinen arkkitehtuuri saman sovellusalueen kaikille ohjelmistotuotteille. Arkkitehtuurin muodostamiseen ja kuvaamiseen voidaan käyttää useita erilaisia tieteellisen tutkimuksen tuloksena syntyneitä lähestymistapoja. Kirjoittajat mainitsevat esimerkkinä QA-DA:n (engl.Quality-driven Architecture Design) ja PuLSE:n (engl. Product Line Software Engineering), jotka määrittelevät vaiheet ja toimenpiteet muun muassa arkkitehtuurin suunnittelutyön toteuttamiseen. (Souza Filho et al. 2008, s. 51)

Tärkeimpänä piirteenä tuoterungon arkkitehtuurissa on mahdollisuus valita ja konfiguroida siihen pohjautuvia ohjelmistoartefakteja. (Pohl et al. 2005, s. 218) Tässä tapauksessa arkkitehtuurista puhuttaessa tarkoitetaan sovellusarkkitehtuuria tai sen kuvausta. Arkkitehtuurityöhön soveltavat menetelmät tuoterunkoon perustuvassa kehityksessä eivät eroa yksittäisten järjestelmien kehittämisestä, mutta erona siinä on kuitenkin useamman tuotteen vaatimuksien huomioon ottaminen samalla kertaa (Linden et al. 2007, s. 40). Alustan toteutusvaiheessa arkkitehtuuriin määritellään joukko ohjelmistokomponentteja ja muita artefakteja. Northrop et al. (2007) mainitsee että, arkkitehtuuri on merkittävin tekijä, joka ohjaa kuinka tuotekehitys hyödyntämällä tuoterunkoa etenee.

Arkkitehtuurityöhön voidaan mainita kolme eri korkean tason periaatetta muunneltavuuden sisällyttämiseksi arkkitehtuuriin: sopeuttaminen, korvaaminen ja laajentaminen. Sopeuttamisessa (engl. adaptation) komponentilla on vain yksi toteutus, joka kuitenkin tarjoaa rajapintoja sen käyttäytymisen sopeuttamiseen. Korvaustekniikassa (engl. replacement) samalle komponentille on olemassa useita toteutuksia, joista yksi valitaan tai kehitetään oma tuotespesifi toteutus. Laajentamisessa (engl. extension) komponentin rajapinnat mahdollistavat uusien komponenttien lisäämistä siihen. (Linden et al. 2007, s. 40)

Seuraavana vaiheena kehityksessä on **alustan toteutus** (engl. domain realisation). Sen päämääränä on uudelleenkäytettävien ohjelmistokomponenttien yksityiskohtainen suunnittelu ja toteutus tuketuen edellisessä vaiheessa toteutettuun referenssiarkkitehtuuriin ja komponenttien karkeaan suunnitteluun. (Pohl et al. 2005, s. 242) Monesti toteutusperiaatteena ohjelmistotuoteperheissä käytetäänkin komponenttipohjaista ohjelmistokehitystä (engl. component-based software engineering), jossa järjestelmä on rakennettu modulaarisesti. Lopullinen järjestelmä voidaan rakentaa kytkemällä yhteen standardoituja ja itsenäisiä komponentteja. (Park et al. 2007, s. 182) Tuoterunkoon perustuvan ohjelmistokehityksen vaatimuksena on nopea ja tehokas lähdekoodin mukautuminen eri ympäristöihin aivan binääritasolle asti, mihin komponenttipohjainen ohjelmistokehitys kykenee vastaamaan. Lähestymistapa mahdollistaa kaikille tuotteille yhteisen ohjelmistoviitekehyksen toteutuksen ja jälkikäteen toteutettavien komponenttien liittämisen tähän rakenteeseen. Voidaan todeta, että korkean tason periaatteet, kuten muunneltavuuden hallinta, tuoterunkoon perustuvassa kehityksessä mahdollistavat uudelleenkäytön suuressa mittakaavassa, kun taas komponenttipohjainen ohjelmistokehitys täydentää sitä mahdollistamalla uudelleenkäytön pienessä mittakaavassa ohjelmakooditasolla. (Atkinson et al. 2000, s. 2) Ensiarvoisen tärkeää alustan komponenttien toteutuksessa on niiden rajapintojen suunnittelu ja toteutus, sillä varianttien olemassaolo rajapinnoissa saattaa aiheuttaa yhteensopimattomuusongelmia. (Pohl et al. 2005, s. 245)

Toteutusvaiheen kannalta olennaisin päämäärä on itsenäisesti toimivan järjestelmä muodostaminen. Tähän sisältyy luonnollisesti komponenttien suunnittelun ja toteutuksen lisäksi koodin kääntäminen, linkkaus ja konfigurointi. Toteutuksen kohteena ovat ohjelmistokomponenttien lisäksi myös muut tuoterungon artefaktit, kuten tietokantataulut. (Pohl et al. 2005, s. 244) Toteutusvaiheessa syntyvän toteutusympäristön tarkoituksena on tarjota tuotteen kehittäjille selkeä toteutusvälineistö, jonka perustella yksittäisiä tuotteita voidaan rakentaa. Tuoterunkoarkkitehtuuri ei riitä yksinään tällaiseksi ympäristöksi. (Koskimies & Mikkonen 2005, s. 166)

Viimeiseksi vaiheeksi Pohl et al. (2007) lukevat alustankehityksessä **alustan testauksen**. Siinä ohjelmistokomponentteja testataan vertailemalla toteutusta määrittelyn, toisin sanoen vaatimusten, arkkitehtuurin ja rajapintasuunnitelmien, kanssa. Edellä esitellyt tuoterunkoartefaktit ovatkin välttämättömiä syötteitä testauksessa. (Pohl et al. 2005, s. 27) Myös testaus itsessään synnyttää tuoterunkoon artefakteja, joita ovat testitapaukset,

testausdokumentaatio, testidata ja testausohjelmistot. Niitä hyödynnetään myös testattaessa useita tuoterunkoon pohjautuvia tuotteita. Lisäksi testauksessa testataan muita tuoterungon artefakteja silmälläpitäen niiden uudelleenkäytettävyys ja laatuvaatimukset. Northrop et al. (2007) huomauttavat, että tuoterungon tapauksessa riittävän testin kattavuuden saavuttaminen on työläämpää, sillä komponenttien on pystyttävä käsittelemään laajempi kirjo syötteitä ja käsittämään enemmän tiloja. Toisaalta alustan testauksen mielenkiinnon kohteena on pelkkien komponenttien testaus, jolloin puhutaan yksikkötestauksesta. Haastavaa tästä tekee kuitenkin sen, ettei tuoterunko ole välttämättä sellaisenaan ajettava järjestelmä. Testausta järjestelmätasolla täydentää kuitenkin tuotekehityksen yhteydessä tehty testaus. (Pohl et al. 2005, s. 266)

2.3 Muunneltavuuden hallinta

Muunneltavuuden hallinta on tuoterunkoon perustuvassa kehityksessä merkittävä huomioonotettava osa-alue. Tässä alaluvussa keskitytään kuvaamaan muunneltavuuden hallintaan liittyvät käsitteet, muunneltavuuden hallinnan olennaisin sisältö ja sen kuvaaminen.

2.3.1 Muunneltavuus, variaatiopisteet ja variantit

Muunneltavuuden hallinta on lähtökohta ohjelmistotuoteperheiden toteuttamisessa ja yksi keskeisimmistä aihepiireistä tämän tutkimuksen kannalta. Se koskettaa kaikkia tuoterungon ja tuotteiden toteuttamisen vaiheita. (Koskimies & Mikkonen 2005, s. 169). Tuoterunkoon perustuvassa ohjelmistokehityksessä pyritään kehittämään tuoterunko, joka tukee mahdollisimman laajaa skaalaa asiakastarpeita tietyllä markkinasegmentillä. Yhden ohjelmiston toiminnallisuuden ymmärtämisen sijaan tulee ymmärtää tuoterungosta johdettujen tuoteinstanssien muunneltavuus eli miten tietyt ominaisuudet poikkeavat toisistaan. (Linden et al. 2007, s. 8) Muunneltavuudella tarkoitetaankin kykyä laajentaa, muuttaa, kustomoida tai konfiguroida ohjelmistoa tai ohjelmistoartefaktia (Gurp et al. 2001, s. 49; Svahnberg et al. 2005, s. 706) Muunneltavuutta pitää määritellä, esittää, hyödyntää, toteuttaa sekä kehittää eli toisin sanoen hallita. (Linden et al. 2007, s. 8)

Se, miten tuoterungon sisältämä muunneltavuus esitetään elinkaaren aikana, muuttaa muotoaan abstraktiotasonsa ja esitystapansa suhteen. Tuoterunkoon perustuvan ohjelmistokehityksen alkuvaiheessa tunnistetuista vaatimuksista muodostetaan ohjelmiston vaihtelevia ominaisuuksia, ominaisuuksista muunneltavuuden toteuttava arkkitehtuurisuunnitelma, jotka edelleen jalostuvat teknisemmiksi ratkaisuiksi konkreettisessa toteutuksessa. (Gurp et al. 2001, s. 49)

Toiminnallisuutta, piirteitä tai laatuvaatimuksia ohjelmistotuoterungossa voidaan tunnistaa kolmea eri tyyppiä: yhteinen toiminnallisuus, muunneltavuus ja tuotespesifit ominaisuudet. (Linden et al. 2007, s. 8) Yhteisellä toiminnallisuudella tarkoitetaan omi-

naisuuksia, jotka ovat kaikissa tuoteperheen tuotteissa täsmälleen samassa muodossa. (Koskimies & Mikkonen 2005, s. 169; Linden et al. 2007, s. 8 Geyer & Becker 2002, s. 2) Muunneltavuudella tarkoitetaan toiminnallisuutta, jotka voivat vaihdella järjestelmästä toiseen. Tuotespesifit ominaisuudet ovat täysin tuotekohtaisia toteutuksia tietystä toiminnallisuudesta, eivätkä kuulu osaksi tuoterunkoa. (Gurp et al. 2001, s. 49).

Halmans et al. (2008) luonnehtivat muunneltavuutta jakamalla sen tekniseen ja toiminnalliseen muunneltavuuteen. Tekninen muunneltavuus liittyy esimerkiksi järjestelmän infrastruktuuriin, konkretisointiin ja toteutukseen, jolloin etsitään vastauksia kysymyksiin ”miten” jokin ominaisuus vaihtelee. Toiminnallisella muunneltavuudella tarkoitetaan toiminnallisia eli järjestelmän vaatimuksista johdettuihin piirteisiin ja laatuun liittyviin ominaisuuksiin. Jälkimmäisessä tapauksessa vastataan kysymykseen ”mikä” vaihtelee. (Halmans et al. 2008) Kuvaukset voidaan tulkita olevan olennaisia tuoterungon elinkaaren vaiheissa, sillä niiden abstraktiotaso on eri. Toiminnallinen muunneltavuus sijoittuu alustankehityksen alkuvaiheille kun taas tekninen muunneltavuus myöhemmäksi prosessiin.

Keskeinen käsite muunneltavuuden hallinnassa on variaatiopiste, jolla tarkoitetaan tietyn ominaisuuden muunneltavuusvaatimuksen ja sitä tukevan suunnitteluratkaisun muodostamaa kokonaisuutta. (Koskimies & Mikkonen 2005, s. 172) Variaatiopiste kuvaa kohdan, jossa eroja toteutettujen järjestelmien välillä esiintyy. (Linden et al. 2007, s. 10; Gurp et al. 2001, s. 49) Se siis konkretisoi järjestelmälle asetetut vaatimukset muunneltavuuden suhteen (Bosch et al. 2002, s. 14) Variaatiopisteeseen voidaankin yhdistää siihen liittyvää lisätietoa, kuten miksi se on valittu variaatiopisteeksi (Pohl et al. 2005, ss. 61-62).

Toinen olennainen käsite on variantti, jolla tarkoitetaan variaatiopisteessä vaihtoehtoisen ratkaisun instanssia eli muunneltavuuden ilmentymää. (Pohl et al. 2005, s. 62) Variantit ovat olemassa olevia mahdollisuuksia tyydyttää variaatiopisteen tarpeet (Linden et al. 2007, s. 11). Varianttiin liittyy tieto siitä, mihin variaatiopisteeseen tai -pisteisiin se liittyy ja onko sillä riippuvuuksia muihin variantteihin. Variantin riippuvuus voi olla pakollisia tai vapaaehtoisia, jolloin tietyn variantin käyttäminen vaatii siitä riippuvan variantin käytön. Variantteihin liittyy myös rajoittavia riippuvuuksia, jolloin variantti estää tiettyjen varianttien mukaan ottamisen järjestelmään. (Buhne et al. 2005)

Variaatiopisteen ominaisuuksia voidaan havainnollistaa sen suhteen, kuinka paljon variantteja variaatiopisteeseen liittyy ja kuinka valinta tehdään. Variaatiopisteeseen voidaan valita varianttien joukosta yksi pakollinen variantti. Variaatiopiste voi olla myös vapaaehtoinen, jolloin variantti voidaan jättää valitsematta. Kolmanneksi on variaatiopisteitä, joihin voidaan valita yksi tai useampia variantteja. (Gurp et al. 2001, s. 51)

2.3.2 Muunneltavuuden hallinta ohjelmistotuoteperheissä

Kruegerin (2002) mukaan muunneltavuuden hallinnan piiriin ohjelmistotuoteperheissä kuuluu sekä ajan ja tilan suhteen vaihtelevat tuoterungon artefaktit. Niistä on tunnistettavissa kolmea eri tasoa, joita ovat tiedostot, komponentit ja tuotteet. (Kruger 2002, s. 41) Svahnberg ja Bosch (2000) jakavat tasot puolestaan hienojakoisemmin koodi-, alikomponentti-, komponentti-, tuote- ja tuotelinjatasoon. Näihin artefakteihin kohdistuvia muunneltavuuden hallinnan toimenpiteitä ovat konfiguraationhallinta, komponenttien muodostus sekä ohjelmiston massakustomointi. (Krueger 2002, ss. 38-39)

Konfiguraationhallinnalla tarkoitetaan toimenpiteitä ohjelmistotuotteen koko elinkaaren ajan siitä, että se on muodostettu asianmukaisesti eli esimerkiksi, että tietyt toimenpiteet on suoritettu sen rakentamisessa. (Daintith & Wright 2008) Kruegerin (2002, s. 44) mukaan toimenpiteet käsittävät ohjelmiston versionhallinnan, kehityshaarojen hallinnan, ohjelmiston lähtötasonhallinta (engl. baseline management) sekä haarautuneiden lähtötasojen hallinta. Pohl et al. (2005, s. 255) mainitsee, että haasteita tällä alueella ilmenee, kun komponentit kehittyvät riippumatta toisistaan. Ylläpidon kannalta on siis tärkeää tietää, mitä versioita komponenteista on käytössä missäkin tuoteinstanssissa. Siksi konfiguraationhallinnan merkitys tuoterunkoon perustuvassa kehityksessä on suurempi verrattuna yksittäisen ohjelmiston kehitykseen. (Krueger 2002, s. 44)

Massakustomointiin liittyviä toimenpiteitä tarvitaan, kun halutaan tukea useita tuotteita samanaikaisesti tietyllä sovellusalueella. Massakustomoinnin käsitteen alle luettaviksi toimenpiteiksi voidaan mainita variaatiopisteiden hallinnan, kustomoinninhallinnan sekä kustomoitavien komponenttien hallinnan. (Krueger 2002, s. 44) Suuressa osassa ohjelmistotuoteperheiden muunneltavuudenhallintaa käsittelevää kirjallisuutta nuo toimenpiteet lasketaan keskeisimmäksi muunneltavuuden hallinnan sisällöksi. Tästedes Kruegerin (2002) termistä massakustomointi käytetään siis nimitystä muunneltavuuden hallinta.

Muunneltavuuden hallinta voidaan nähdä prosessina, jossa Gurp et al. (2001, s. 50) määrittelevät toimenpiteiksi muunneltavuuden tunnistamisen, rajoittamisen, toteuttamisen ja varianttien hallinnan. Riippuen alustankehityksen vaiheesta kehitysartefaktien muunneltavuus määritellään hiukan eri abstraktiotasoilla. Korkeimmalla tasolla muunneltavuuteen vaikuttavat ennen kaikkea sidosryhmien tarpeet mutta myös kansalliset lait ja standardit, joten ne kuvataan hyvin yksinkertaisesti. Siirryttäessä vaatimusmäärittelyyn siirrytään myös abstraktiotasolla alaspäin tarkempaan kuvaukseen. Samalla kehitysartefaktien variaatiopisteiden ja varianttien määrä kasvaa, koska ylätason vaatimukset heijastuvat alemmalle tasolle mahdollisesti lukuisiin eri kohtiin. (Pohl et al. 2005, s. 72; Gurp et al. 2001, s. 49)

Muunneltavuuden hallinnan toimenpiteistä muunneltavuuden tunnistamisessa pyritään määrittelemään, missä kohdissa järjestelmää muunneltavuutta tarvitaan. Muunneltavuu-

den tunnistaminen tarkoittaa käytännössä vaihtelevien piirteiden (engl. feature) listaamista. Piirteet ja muunneltavuus ovat yhdistävä tekijä järjestelmän vaatimusten ja teknisten suunnittelupäätösten välillä. Tuotteen piirteet erottavat sen muista tuoteperheen tuotteista, jolloin tuoteperheen tulee tukea muunneltavuutta näille piirteille. Muunneltavuuden tunnistamisessa voidaan apuna käyttää piirremallia (engl. feature model). (Gurp et al. 2001, s. 51; Svahnberg et al. 2005, ss. 708-709)

Kun variaatiopiste on tunnistettu, pitää sitä rajoittaa, mikä tapahtuu prosessissa seuraavana. Ideana on kyetä tarjoamaan riittävästi joustavuutta nykyisten ja tulevaisuuden tarpeiden toteuttamiselle mahdollisimman kustannustehokkaasti. Tässä vaiheessa tulee ottaa kantaa asioihin, kuten missä vaiheessa variantin kiinnitys tapahtuu. (Gurp et al. 2001, s. 51) Tätä ajankohtaa kutsutaan kiinnitysajaksi (engl. binding time). Se kuuluu tuotekehityksen vastuulle ja voi tapahtua menetelmästä riippuen käännösvaiheessa, linkkausaikana, sovellusta käynnistettäessä tai ajon aikana. (Koskimies & Mikkonen 2005, s. 165; Linden et al. 2007, ss. 11-12) Lisäksi pitää ottaa kantaan milloin ja miten variantteja lisätään järjestelmään vai voidaanko niitä lisätä ollenkaan tai onko variantin valinta vapaaehtoinen, pakollinen vai voidaanko valita useampi variantti. Lopuksi päättää kunkin variaatiopisteen kuvaustapa. (Gurp et al. 2001, s. 51; Oliveira Jr. et al. 2005, s. 6)

Tunnistamisen jälkeen muunneltavuus täytyy toteuttaa. Siinä valitaan muunneltavuuden rajoittamiseen perustuen sopiva toteutustekniikka. (Gurp et al. 2001, s. 51) Oliveira Jr. et al. (2005, s. 9) mainitsevat toteutustekniikoiksi esimerkiksi periyttämisen, laajentamisen ja parametroinnin. Jacobsen et al. (1997) listaavat edellisten lisäksi konfiguroinnin ja johdettujen komponenttien generoinnin korkean tason ohjelmointikieltä käyttäen. Toteutustekniikka valitaan sen perusteella, mikä on päätetty kullekin variaatiopisteelle kiinnitysajankohdaksi. (Oliveira Jr. et al. 2005, s. 9) Kun tuoteinstanssit on tuotekehitysvaiheessa luotu, eivät ne enää kuulu muunneltavuuden hallinnan piiriin. Näin vältetään tarve tehdä muutoksia sekä tuoterunkoon että yksittäisiin tuotteisiin ja vähennetään lukuisten yksittäisten irrallaan elävien kehityshaarojen tuomaa monimutkaisuutta. (Krueger 2002, ss. 41-42)

2.3.3 Muunneltavuuden mallinnus

Muunneltavuuden mallinnus on yksi merkittävimmistä tutkimuksen kohteista muunneltavuuden hallinnassa. Tämän todistaa akateemisten julkaisujen määrä suhteessa muihin tuoterunkoon perustuvan ohjelmistokehityksen aihealueisiin. (Chen et al. 2009) Muunneltavuuden mallinnus on tärkeää monimutkaisten järjestelmien tapauksessa ja siitä voisi kirjoittaa kokonaisen tutkimuksen verran. Tässä tutkimuksessa pyritään esittelemään vain yleisiä piirteitä ja vaatimuksia.

Muunneltavuuden mallinnukseen on käytetty aihealuetta käsittelevissä tutkimuksissa useita eri lähestymistapoja. Niiden keskinäiset erot voi tiivistää kahteen muuttujaan:

miten termi ominaisuus tai piirre käsitetään ja kuuluuko kuvaus osaksi tuoterunkoa. (Linden et al. 2007, s. 9) Pääasiallisia kuvaamismuotoja ovat integroitu dokumentaatio ja ortogonaalinen dokumentaatio. Integroidussa dokumentaatiossa muunneltavuusmalli on osa tuoterunkoa, sillä muunneltavuus on mallinnettu laajentamalla muita mallinnus- ja dokumentaatiokäytäntöjä. Ortogonaalisessa dokumentointiperiaatteessa muunneltavuutta kuvaavat mallit on erotettu muista kehitysartefakteista sekä järjestelmän suunnittelunäkökohdista ja sitä kuvaamaan on kehitetty oma kuvauskielensä. (Metzger & Pohl 2014)

Yksi tunnetuin integroiduista lähestymistavoista on piirrevetoinen, jossa muunneltavuuden mallinnuskeinona käytetään piirremallia. Yksi merkittävin luomus siitä on FODA (Feature-Oriented Domain Analysis), josta on kehitetty useita eri laajennoksia. (Chen et al. 2009, s. 83) Piirremalli pyrkii tarjoamaan korkean tason katsauksen tuoteperheen tärkeimmistä yhteisistä ja vaihtelevista piirteistä ja hyödyntää kuvaustekniikkanaan UML:ää. Siinä muunneltavuutta ja sen tarpeellisuutta kuvataan käyttämällä kuvausnotaation laajennoksia, kuten stereotyyppejä, lisätietomääreitä (engl. tagged value) ja rajoitteita. (Myllymäki 2002, s. 5)

Vaikka kuvaustekniikoita sekä integroidussa että ortogonaalisessa periaatteessa on monia, voidaan mallinnukselle määritellä yleisiä vaatimuksia. Näitä ovat mahdollisuus kuvata yhteiset ja vaihtelevat ominaisuudet yksiselitteisesti. Mallinnusnotaatiolla tulee myös pystyä ilmaisemaan erilaiset muunneltavuuden tyypit eli millä tavalla jokin vaihtelee. Kolmanneksi mallinnuksella tulee kyetä esittämään muunneltavuuksien välisiä riippuvuuksia. Koska toimintaympäristö muuttuu, tulee myös malleja pystyä kehittämään. Mallin tarkoituksena on toimia ensisijaisesti kommunikaation välineenä, joten on myös ensiarvoisen tärkeää, että mallit ovat yksinkertaisia ja helppoja ymmärtää. (Maßen & Lichter 2002) Muunneltavuuden mallinnukseen käytetään graafisen notaation ohella myös sanallista muotoa tai graafisen ja sanallisen muodon yhdistelmää. (Maßen & Lichter 2002)

Erilaisilla malleilla on yksittäin käytettyinä omat rajoitteensa ja erillisinä ne muodostavat sekavan kokonaisuuden, josta ei voida määrittää, missä kehityksen vaiheessa ne ovat relevantteja. (Pohl et al. 2005, ss. 74-75) Muita yleisiä haasteita muunneltavuuden mallinnuksessa on tunnistettu esimerkiksi puute yhtenäiselle mallinnukselle tuoterungon elinkaaren eri vaiheissa, mallien kyvyttömyys kuvata muunneltavuuden riippuvuuksia ja riippuvuuksien keskinäistä vuorovaikutusta sekä järjestelmää uudelleenkäyttöä varten. Muita haasteita ovat muodollisuuden ja tarkkuuden puute sekä monimutkaisuus, jotka heikentävät kykyä kommunikoida ulkopuolisille sidosryhmille. (Chen et al. 2009, s. 86) Maßen ja Lichter (2004, s. 169) toteavat edellisten lisäksi, että mallit muodostuvat helposti hyvin monimutkaisiksi ja vaikeasti ylläpidettäviksi, joten niiden kuvaamiseen tarvitaan konkreettisia toimintaohjeita ja työkaluja.

Kuten dokumentaatiosta yleensä, muunneltavuuden mallinnuksesta on useita hyötyjä. Konkreettinen malli parantaa kommunikaatiota eri sidosryhmien, kuten asiakkaiden, suuntaan. Dokumentaatio tarjoaa välineen jäljittää järjestelmään toteutetun muunneltavuuden alkuperän. Lisäksi dokumentaatio pakottaa kehittäjät perustelemaan variaatiopisteiden ja varianttien suhteen tehdyt valinnat, mikä helpottaa päätöksentekoa asiakkaan päässä päätettäessä esimerkiksi jonkin tietyn variantin valintaa. Ohjelmistokehittäjillä tuotekehityksessä dokumentaatio helpottaa kiinnitysajankohdan ja toteutustavan valintaa. (Pohl et al. 2005, ss. 73-74)

2.4 Tuotteenhallinnan kysymyksiä ohjelmistotuoterungoissa

Tuotteenhallinnan katsottiin olevan olennainen osa tuoterunkoon perustuvaa ohjelmistokehitystä. Tässä alaluvussa käsitellään tuotteenhallintaan liittyviä erityiskysymyksiä, kuten monituotelinjoja ja tuoterungon evoluutiota.

2.4.1 Useiden tuotelinjojen tuoteperheet

Tyypillisen tuotelinjan rajauksen tapauksessa on kyse tuoteinstanssien johtamisesta itsenäisestä ja autonomisesta tuotelinjasta. Nykypäivän ohjelmistosovellukset ovat mittakaavaltaan suuria. Samalla tavalla kun yritykset ovat omaksuneet tuoteperheparadigman tavallisissa ohjelmistoissa, on suuren mittakaavan ohjelmistoihin pyritty tuomaan samaa ajattelumallia. (Holl et al. 2012, s. 828)

Holl et al. (2012) määrittelevät suuret ja erittäin suuret järjestelmät käyttämällä termiä monituotelinja (engl. multi product line). Ne ovat joukko itsenäisiä, mutta toisistaan riippuvia tuotelinjoja, jotka yhdessä muodostavat mittakaavaltaan suuria ohjelmistoja. Monituotelinjan erilliset tuotelinjat voivat esiintyä itsenäisesti, mutta tyypillisesti hyödyntävät jaettuja resursseja täyttääkseen järjestelmälle asetetut kokonaisvaatimukset. Monituotelinjoja on luonnehdittu kirjallisuudessa myös nimityksillä ohjelmistoekosysteemit, kokoonpantu tuotelinja ja tuotepopulaatiot. (Holl et al. 2012, s. 829) Savolainen et al. (2012, s. 220) toteavat, että kun tuoterunkoa ”venytetään” liikaa, kärsii sen kehityksen tehokkuus ja tuloksena joudutaan muodostamaan erillisiä tuotelinjoja, jotka edelleen hyödyntävät samaa uudelleenkäytettävää arkkitehtuuria.

Monituotelinjat vaativat infrastruktuurin, joka tarjoaa perusrakenteen ja puitteet tuotteiden kehittämiselle ja käytölle. Tällaista infrastruktuuria voidaan kuvailla ja sen tunnusomaisia piirteitä havainnollistaa määrittelemällä infrastruktuurin kyvykkyydet, jotka määrittelevät sen pääominaisuudet. (Holl et al. 2012, s. 829)

Holl et al. (2012) ovat tunnistaneet tekemänsä kirjallisuuskatsauksen perusteella vaatimuksia monituotelinjaa tukevalle infrastruktuurille ja käytännöille sen perusteella, mitä vaikeuksia monituotelinjat aiheuttavat ja mitä ratkaisuja näihin on pyritty kehittämään. Ensinnäkin monituotelinjan tulisi tukea hajautetusti tehtävää tuotteiden johtamista, jois-

sa esiintyy muunneltavuuksien välisiä riippuvuuksia tuotelinjojen kesken. (Holl et al. 2012, s. 837) Monesti tuotelinjakohtaiset muunneltavuusmallit perustuvat poikkeaviin mallinusnotaatioihin, sillä tuotelinjojen kuvattavat komponentit ovat heterogeenisiä. Näin ollen niitä on vaikeita integroida yhdeksi yhtenäiseksi malliksi. (Rabiser et al. 2010, s. 2) Monituotelinjoja varten tulisi kuitenkin olla mallinnuskieli, joka tukee mittakaavaltaan suurien järjestelmän kuvaamista, ja joka olisi myös mahdollista se monituotelinjan parissa työskentelevien ihmisten kesken. (Holl et al. 2012, s. 837)

Tuotelinjoilla tulisi lisäksi olla selkeät rajapinnat erilaisten tietojen välittämiseen muille tietoa tarvitseville tuotelinjoille. Tätä tietoa on esimerkiksi muunneltavuuden suhteen tehdyt päätökset, joilla on vaikutusta muihin järjestelmiin. Tähän liittyen tuotteiden johtamiseen tulisi olla ohjeistus, jossa käsiteltäisiin huomioon otettavat riippuvuudet tuotelinjojen välillä. Valintojen johdonmukaisuus tuotelinjojen kesken tulisi niin ikään kyetä tarkistamaan jollakin mekanismilla. Myös riippuvuudet ja niiden tyypit tulee määritellä epäjohdonmukaisuuksien välttämiseksi. (Holl et al. 2012, s. 837) Edellisten kohtien perusteella voidaan päätellä, ettei kyseessä ole tuoterunkoon perustuvassa kehityksessä periaatteellisesti mitään mullistavaa. Tuoterungon muunneltavuuden ja muiden päätösten suhteen joudutaan vain kiinnittämään korostetusti huomiota, jotta kokonaisuus säilyy koherenttina, ylläpidettävänä ja helposti ymmärrettävänä.

2.4.2 Tuoterungon evoluutio

Toimintaympäristö muuttuu ja tuotelinjan tulisi pystyä mukautumaan muuttuviin olosuhteisiin ja uusiin vaatimuksiin, joita tulee niin organisaation ulkoa kuin sisältäkin. Ulkopuolisia tekijöitä ovat esimerkiksi kiristyvä kilpailu, ostajien vaatimukset tai käytettyjen teknologioiden kehitys. Organisaation sisäpuolelta tulevia muutosta ajavia voimia ovat muun muassa alustankehittäjien näkemykset teknologioiden hyödyllisyydestä tulevaisuudessa tai tuotekehityksen muutospyynnöt yhteisen tuoterungon toiminnallisuuteen. (McGregor 2003, ss. 8-9; Linden et al. 2007, s. 302) Seikkoja, kuten tuoterunkoarkkitehtuurin tilaa, tuoterunkoon liittyvin prosessien tehokkuutta, organisaatiorakenteen ja liiketoiminnan tavoitteiden välinen suhdetta tai tuotelinjaan kohdistuvat tulevaisuuden haasteita ja mahdollisuuksia, tulisi pohtia jatkuvasti. Yksi tapa pitää nämä säännöllisesti yrityksen asialistalla, on roadmappaus eli tiekartan suunnittelu. Suunnitelmassa tulisi aluksi määritellä lähitulevaisuuden aikana mahdollisesti realisoituvat muutokset ja toimenpiteet niille. Suunnitelmassa tulisi myös ottaa kantaa keskipitkällä aikavälin ratkaisuihin ja pitkän aikavälin epävarmoihin odotuksiin. (Linden et al. 2007, s. 302) McGregor (2003) on samoilla linjoilla ja toteaakin tuotelinjan evoluution edellytykseksi suunnittelun. Siinä muutoksen tavoite ja suunta tulee olla määritelty. Suunnitelman laatiminen edellyttää ymmärrystä tavoitteesta ja kunkin muuttuvan artefaktin nykyisestä konfiguraatiosta. Suunnitelmassa on näin ollen määriteltävä miten kukin artefakti kehittyy sen nykyisestä tilasta tavoiteltuun tilaan. (McGregor 2003, s. 9;15;16) Linden et al. (2007, s. 320) kuitenkin muistuttavat, että mitä pidemmälle suunnitelma sijoittuu, sitä vähemmän uskottava se on.

Ulko- ja sisäpuolelta tulevat vaatimukset kohdistavat muutospainetta tuotelinjaa kohtaan ja se, miten muutos tai kehittyminen tapahtuu, määrittyy pitkälti sen perusteella, kuinka paljon vaihtelua organisaatio sallii tuoterungon infrastruktuurissa. (Schmid & Verlage 2002, ss. 54-55) McGregorin (2003) mukaan tuotelinjan evoluutio eli kehitys aiheuttaa organisaatiolle monia haasteita erityisesti siksi, koska monia eri tuotteita kehitetään samasta joukosta kehitysartefakteja. Monimutkaiset suhteet artefaktien välillä korostavat kehityksen aiheuttamia vaikutuksia tuotelinjaan. Muutokset voivat olla odotettuja ja ohjattuja tai ne voivat jäädä tunnistamattomaksi siihen asti kunnes useammat muutokset kasautuvat ja aiheuttavat tarpeen suuremmalle muutokselle. (McGregor 2003, s. 1) Tämä korostuu etenkin kun puhutaan monituotelinjoista, joiden eri tuotetyyppien välillä on riippuvuuksia. (Holl et al. 2012)

Kun evoluution suunnitteluun on otettu kantaa, tarvitaan konkreettisia toimintaohjeita tai tekniikoita, joilla mahdollistetaan ja hallitaan esimerkiksi tuoterungon arkkitehtuurin evoluutiota. Tuoteperheiden kontekstissa evoluution mahdollistuu pitkälti muunneltavuuden hallinnan kautta. Svahnberg (2003, s. 155) mainitseekin, että edellytys evoluution onnistumiselle on sen tarkka noudattaminen koko tuoterungon kehityksessä elinkaaren aikana. Hänen mukaan evoluutio ja muunneltavuus kulkevat käsi kädessä, sillä evoluutiossa on pohjimmiltaan kyse varianttien laajentamisesta tai muutoksista, kun taas muunneltavuuden hallinnassa hallittaan tuotteen variantteja. Tämä ei kuitenkaan päde aivan kaikissa evoluution tapauksissa, mutta on riittävä lähtökohta. (Svahnberg 2003, s. 155)

Evoluutio voidaan jakaa tuoteperheiden tapauksessa kolmeen kategoriaan. Niitä ovat vaatimusten, tuoteperhearkkitehtuurin ja ohjelmistokomponenttien evoluutio. Kategoriat ovat kytköksissä toisiinsa, sillä esimerkiksi muutokset vaatimuksissa voivat laukaista tiettyjä muutoksia komponenteissa tai arkkitehtuurissa. Seuraavaksi on tarkennettu kategorioiden piiriin lukeutuvia evoluution tyyppejä. (Svanhberg 2003, s. 219)

Tuoteperheen vaatimuksien evoluutiossa muutosvaatimukset voivat liittyä tuotteiden laatuaspekteihin. Ne ovat saattaneet jäädä tuotteen elinkaaren alkuvaiheen toissijaisiksi. Lisäksi tuoterungon vaatimukset liittyvät infrastruktuuriin, johon lukeutuu kolmannen osapuolen komponentit, käyttöjärjestelmä tai laitteistoa. Infrastruktuurin muutokset vaikuttavat itse tuoterunkoon ja siitä johdettaviin tuotteisiin. Lisäksi käyttäjät saattavat ehdottaa tuotteeseen uutta toiminnallisuutta tai vaatia laajempaa tukea tuotteellensa. (Svanhberg & Bosch 1999, s. 4; Svahnberg 2003) Rajauksen nimissä ei ole aina välttämättä viisasta lisätä tällaista toiminnallisuutta tuoterunkoon. Tietyissä tapauksissa tuotelinjan tuote johdetaan normaalisti ja tehdään tarvittavat räätälöinnit. Äärimmäisessä tapauksessa on tarve täysin uuden tyyppiselle tuotteelle, mikä johtaa kokonaan uuden tuotelinjan ja myös sitä tukevan liiketoimintayksikön perustamiseen. (Svanhberg & Bosch 1999, s. 4; Svahnberg 2003)

Uusilla vaatimuksilla, jotka tulisi sisällyttää kaikkiin tuotteisiin, on vaikutus tuotelinjan arkkitehtuuriin. Alettaessa kehittämään uutta joukkoa tuotteita täytyy päättää, voidaanko kehitys pitää samassa kehityshaarassa, jolloin arkkitehtuurista johdetaan tuote, vai onko välttämätöntä jakaa tuotelinjan arkkitehtuuri kahteen siten, että alkuperäistä arkkitehtuuria käytetään pohjana toiselle. Kehitettäessä uutta tuotetta voidaan arkkitehtuuriin kuuluvia komponentteja myös jakaa, vaihtaa tai lisätä kokonaan uusia riippuen siitä, mihin muutoksella pyritään. Komponenttien välisiä yhteyksiä on usein välttämätön päivittää poistamalla tai lisäämällä uusia riippuvuuksia. (Svahnberg & Bosch 1999, s. 4)

Arkkitehtuuriin tehdyt muutokset heijastuvat edelleen komponentteihin, joista arkkitehtuuri koostuu. Komponenttien sisäistä toteutusta voidaan joutua muuttamaan tai luomaan sille täysin uusi toteutus. Myös sen toiminnallisuutta voidaan muuttua lisäämällä tai poistamalla komponentin toteutusta siten, että myös tähän suhteessa oleviin komponentteihin aiheutuu muutoksia. Viimeisenä vaihtoehtona on lisätä ulkopuolisia komponentteja toteuttamaan uusi toiminnallisuus. (Svahnberg & Bosch 1999, s. 5)

Edellisten perusteella voidaan päätellä, että tuoterunkoarkkitehtuurin joustavuudella on evoluution kannalta merkittävä vaikutus. Breivold et al. (2010) ovat kirjallisuuskatsauksessaan tarkastelleet ohjelmistojen mahdollisuutta evoluutioon arkkitehtuurin tasolla. Heidän löydöstensä perusteella siihen vaikuttaa esimerkiksi arkkitehtuurin mallinnus, sillä toisiinsa vaikuttavien ohjelmistoartefaktien kuvaus vähentää epäjohdonmukaisuutta ja evoluution mahdollisuuden heikentymistä ajan myötä. Tarkemmin käsiteltynä mallinnettavia asioita ovat jäljitettävyys vaatimusten, ominaisuuksien, arkkitehtuurin elementtien ja toteutusten välillä. Toisia merkittäviä havaintoja ovat esimerkiksi erilaisten taktiikoiden käyttö ei-toiminnallisten vaatimusten mahdollistamiseksi arkkitehtuurissa, arkkitehtuuriin liittyvän tietämyksen formalisointi ja mallinnus tai laatuvetoisen sovellusten uudelleenkehitysmallin käyttäminen. Arkkitehtuurin suunnittelussa tulisi myös käyttää tekniikoita, jotka asettavat laadulliset aspektit suunnittelun keskiöön. (Breivold et al. 2010, ss. 17-18) Mainitut kehitysehdotukset ovat laaja-alaisia ja niiden tarkempi esittely ei ole tutkimuksen puitteissa tarkoituksenmukaista. On kuitenkin tärkeämpää yleisellä tasolla tunnistaa mallinnuksen merkitys myös evoluution kannalta.

Breivold et al. (2010) kirjoittavat, että ohjelmiston tarjoaman evoluution mahdollisuutta käsitellessä on tarkasteltava teknisten kysymyksien ohella myös organisatorisia tekijöitä. Heidän mukaan on tarkasteltava kuinka paljon joustavuutta kannattaa mahdollistaa joustavuuden mukanaan tuomien taloudellisten riskien takia. Arkkitehtuuria on evaluoitava säännöllisin väliajoin tarkastellen muun muassa sen laadullisia tekijöitä, mikä on puolestaan ennaltaehkäisevää toimintaa arkkitehtuurityössä. (Breivold et al. 2010, s. 16) Vaikka suuria panostuksia arkkitehtuurin luomisen alkuvaiheissa tehdäänkin, eivät toimet saa jäädä huomiotta sen elinkaaren myöhäisemmissä vaiheissa. Voidaan päätellä, että yleisen ohjelmistoarkkitehtuurikirjallisuuden havainnot eivät poikkea ohjelmistotuoteperheiden tapauksessa, sillä samansuuntaisia poimintoja löytyy molemmista. Esi-

merkiksi aikaisemmin tässä tutkimuksessa mainitut muunneltavuuden mallinnus riippuvuuksien suhteen sekä tuoterungon ylläpidon tärkeys ovat tämän tyyppisiä poimintoja.

3. TIETOVARASTOINTI

Tässä kappaleessa on tarkoitus käsitellä tietovarastoinnin perusteita käymällä läpi sen perusperiaatteita. Ensin käsitellään aihepiiriä yleisesti, minkä jälkeen seuraa tietovarastoarkkitehtuurin teoria. Seuraavaksi siirrytään tietovarastojen pohjana olevien tietomallien pariin. Lopuksi tutkitaan itse kehitysprosessia ja tietovarastointiin liittyviä haasteita.

3.1 Tietovarastointi yleisesti

Nykyaikana yksikään yritys ei voi harjoittaa liiketoimintaa ilman tietojärjestelmiä. Tietojärjestelmien sisältämä operatiivinen data on elintärkeätä yrityksen menestymiselle. Informaatio, jota tarvitaan strategisten päätösten tekemiseen, vaatii kuitenkin laaja-alaisesti tietoa koko organisaatiosta. Tietotarpeet saattavat liittyä yrityksen operatiiviseen toimintaan, kriittisiin suorituskykyindikaattoreihin, ajan myötä tapahtuvaan kehitykseen erilaisissa liiketoiminnalle tärkeissä asioissa tai yrityksen vertailuun suhteessa muihin toimijoihin alalla. Operatiivisten järjestelmien data on usein liian hajautunutta, monimuotoista ja tietotarpeiden kannalta väärässä formaatissa, joten sellaisenaan se ei tue strategista päätöksentekoa. Tähän tarpeeseen vastaavat tietovarastot. (Ponniah 2010, ss. 4-5) Yksinkertaistettuna tietovarastojen tavoitteena on siis tarjota kokonaisvaltainen kuva yrityksestä ja sen toiminnasta integroimalla dataa yhteen paikkaan useasta eri lähteestä. (Inmon et al. 2010, s. 7)

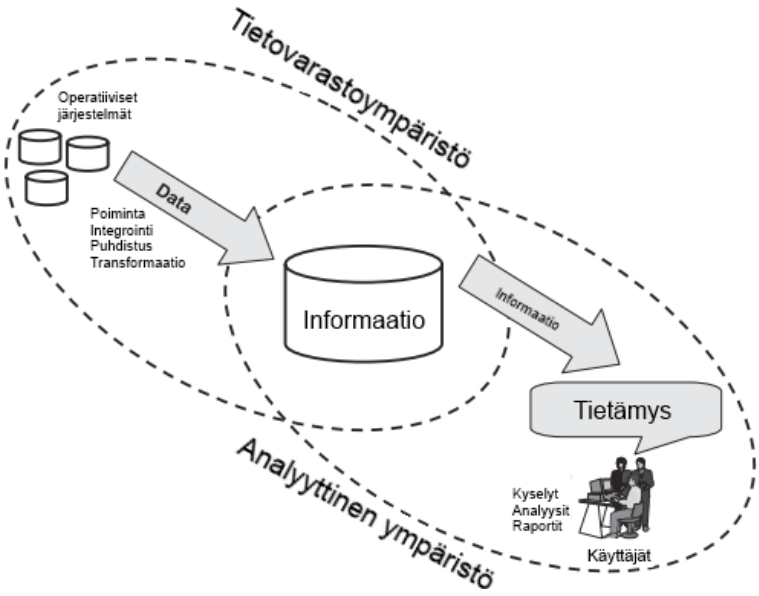
Tietovarastolla tarkoitetaan järjestelmää, johon erilaisista lähdejärjestelmistä haetaan ja yhdistellään dataa tiettyjen ajanjaksojen välein ja varastoidaan se joko dimensionaaliseen tai normalisoituun muotoon. (Rainardi 2008, s. 1) Termillä tietovarastointi (engl. data warehousing) tarkoitetaan puolestaan koko prosessia ja sen vaiheisiin liittyviä toimenpiteitä, jotka vaaditaan datan saamiseksi lähdejärjestelmistä tiedon sovellusten ja tarvitsijoiden saataville. (Watson 2002, s. 2) Liiketoiminnan näkökulmasta tietovarasto määritelläänkin päätöksentekoa tukevaksi järjestelmäksi. Käsiteltävä data voi käsittää kaikkea yksinkertaisista analyyseista monimutkaisiin useiden lähteiden dataa yhdisteleviin analyyseihin. (Westerman 2001, s. 2) Tietovarastot tarjoavat integroidun ja kokonaisvaltaisen näkymän koko yrityksestä, nykyisestä toiminnasta ja kauempaa menneisyydestä ilman, että kuormitetaan yrityksen operatiivisia järjestelmiä. (Ponniah 2010, s. 15)

Tietovarastot erotetaan usein fyysisesti yrityksen operatiivisista järjestelmistä ja ylläpidetään erillään, sillä niiden käytöllä on toisistaan poikkeavia tavoitteita. Operatiivisissa järjestelmissä tietokantoihin kohdistuvat operaatiot ovat strukturoituja ja toistuvat useasti. Lisäksi ne ovat lyhyitä, atomisia ja toisistaan erillisiä sekä vaativat yksityiskohtais-

ta ja ajantasaista dataa luettavaksi ja päivitettäväksi. Operatiivisissa tietokannoissa yhdenmukaisuus ja virhetilanteista toipuminen sekä samanaikaisuudenhallinta ovat kriittisiä. Tietovarastojen tavoite on puolestaan tukea päätöksentekoa. Dataan kohdistettavat kyselyt ovat luonteeltaan tilapäisiä ja suorittavat monimutkaisia kyselyitä, jotka käsittelevät miljoonia rivejä summaten arvoja ja yhdistellen tauluja. Näin ollen suoritusteho ja kyselyiden vasteajat ovat tietovarastojen tapauksessa tärkeitä. (Chaudhuri & Dayal 1997, ss. 517-518) Tietovaraston ja operatiivisen tietojärjestelmän käytön välillä on myös se ero, että tietovarastoon kohdistetaan useimmiten lukuoperaatiota, kun taas operatiivisissa järjestelmissä tehdään luvun lisäksi kirjoitusta ja päivityksiä. (Ponniah 2010, s. 13)

Tietovarastoja luonnehditaan usein sen keskeisimpinä ominaisuuksien kautta, joita ovat aiheorientoituneisuus, integroituneisuus, vakaus, aikavarianttius. (Imnon et al. 2010, s. 7) Näiden termien yhteydessä puhutaan usein tietovarastojen filosofiasta. Termit eivät sellaisenaan anna selkeätä kuvaa merkityksestä niiden takana, joten niitä tulee hieman avata. Aiheorientoituneisuudella tarkoitetaan sitä, että kaikki data järjestetään sen perusteella mihin asiaan sen liittyy. Esimerkiksi valmistukseen, myyntiin tai mainostukseen liittyvä data talletetaan samaan paikkaan huolimatta siitä, mistä se on peräisin. Datan integrointi merkitsee sitä, että eri lähteiden välillä esiintyvät eroavaisuudet datassa karsitaan, jotta datan muoto, käyttötarkoitus ja abstraktiotaso täsmäävät. Vakaudella kuvataan taas sitä, ettei dataa poisteta tietovarastosta, kun sen käyttö lähdejärjestelmissä päättyy. Historiadata on siis aina saatavilla. Tietovaraston datan aikavariantti luonne puolestaan merkitsee sitä, että kaikki data liittyy siihen kontekstiin, jossa se oli tallennettaessa. (Silvers 2008, ss. 2-4) Yksi keskeinen ominaisuus datan tallennuksessa tietovarastoon on sen granulariteetti, jolla tarkoitetaan sen tarkkuustasoa. Kun tallennus tehdään alhaisimmalla tarkkuustasolla, mahdollistuu analyysin etenemisen korkean abstraktion summatasosta aina alhaisimpaan tarkkuustasoon, jolloin puhutaan datan olevan atomista. Päätös granulariteetista vaikuttaa aina tietovaraston kokoon siten, että mitä tarkemmin data on tallennettu, sitä suurempi tietovaraston koko tulee olemaan. (Ponniah 2010, s. 28)

Kimball et al. (2011, s. 11) toteavat, että tietovarastot rakentavat perustan liiketoimintatiedon hallinnalle (engl. business intelligence), joten tietovarastoja ei rakenneta ilman näitä datalle lisäarvoa antavia sovelluksia. Ponniah (2010) määritteleekin liiketoimintatiedon hallinnan koostuvan kahdesta loogisesta kokonaisuudesta tai ympäristöstä: tietovarastoinnista ja tiedon analysoinnista. Kuva 2 havainnollistaa tilannetta. Tämän tutkimuksen fokus on pelkästään tietovarastoinnissa.



Kuva 2. *Tietovarastoinnin kytkeytyminen liiketoimintatiedon hallinnan käsitteeseen (mukailtu lähteestä Ponniah 2010)*

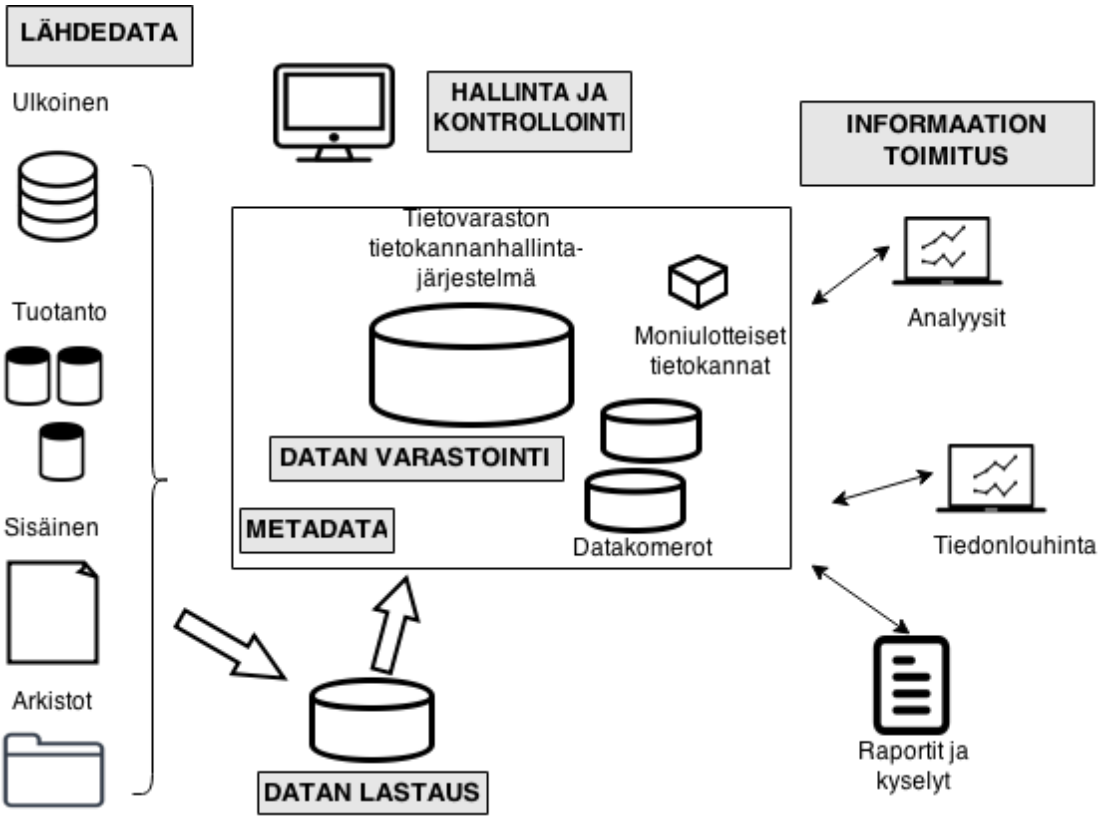
Tietovarastoja ei saa nähdä yksittäisenä ohjelmistotuotteena, joka voidaan ostaa yritykseen. Se tulee mieltää kokonaisvaltaisena ja käyttäjäkeskeisenä ympäristönä, jossa käyttäjät ovat suoraan kosketuksissa datan kanssa. (Ponniah 2010, ss. 15-16)

3.2 Tietovarastoarkkitehtuuri

Tässä alaluvussa käsitellään tietovarastojen keskeisimpiä rakennuspalikoita. Ensin esitellään komponentit, joita tietovarastoissa esiintyy. Sen jälkeen havainnollistetaan erilaisia tietovarastoarkkitehtuureja eli sitä, miten eri tavoilla komponentit voidaan järjestää tietovarastoissa. Kolmantena tarkastellaan arkkitehtuurin valintaan vaikuttavia seikkoja ja lopuksi käsitellään arkkitehtuurin valintaan johtavia perusteluita ja lopuksi havainnollistetaan tietovarastojen rakentamisessa relevantteja arkkitehtuurien tasoja.

3.2.1 Tietovarastojen keskeisimmät komponentit

Tietovarastoinnissa on tunnistettavissa kolme pääkokonaisuutta, jotka muodostavat tietovarastoarkkitehtuurin: datan hakeminen, datan varastointi ja tiedon toimittaminen. Kokonaisuuden alle kuuluu erilaisia komponentteja, jotka tarjoavat tietynlaista palvelua toisilleen. Keskeisimpiä komponentteja tietovarastoarkkitehtuurissa ovat datan lähteet, lastaus, varastointi, metadata ja informaation toimitus sekä hallinta ja kontrollointi. (Ponniah 2010, s. 144) Kuvassa 3 on esitelty nämä itsenäiset kokonaisuudet. Edellä mainituista kategorioista tiedon toimittamisen alle lukeutuvat komponentit kuuluvat tutkimuksen rajauksen ulkopuolelle.



Kuva 3. *Tietovarastojen komponentit (mukailtu lähteestä Ponniah 2010)*

Datan lähteitä ovat *tuotantodata*, joka saadaan organisaation erilaisista operatiivisista tietojärjestelmistä. Data on monesti erilaisilla alustoilla, tietokannoissa ja käyttöjärjestelmissä. Lisäksi on *sisäistä dataa*, jolla tarkoitetaan yksittäisten työntekijöiden hallussa olevaa dataa laskentataulukoiden, erilaisten dokumenttien, asiakasprofiilien ja jopa henkilökohtaisten tietokantojen muodossa. Kolmantena tyyppinä on *arkistoitu data*, joka on operatiivisista järjestelmistä erotettua vanhaa dataa. Se voi olla erillisissä tietokannoissa, tiedostoina kovalevyllä tai jopa magneettinauhoilla. Neljäs datan lähde on *ulkopuolinen data*, joka vaihtelee yrityksestä ja johdon tietotarpeista riippuen. Se voi olla tilastoja teollisuudenalalta, kilpailijoiden markkinaosuuksia tai standardiarvoja taloudellisten mittarien suhteen. (Ponniah 2010, s. 36)

Lastauskomponentti pitää sisällään kolme erillistä operaatiota: datan poiminta, transformaatio ja lataus (engl. extract, transform, load). Voidaan myös käyttää termiä ETL-prosessi. Lastauskomponentti koostuu työkaluista näiden operaatioiden suorittamiseen tarjoamalla alueen, jolla lastaus on mahdollista. Koska tietovaraston data tulee useista eri järjestelmistä, on lastausalue välttämättömyys tietovaraston rakentamisessa. Operaatioista ensimmäisenä on *poiminta*, jolla tarkoitetaan datan hakemista lähdejärjestelmistä. Operaatio saattaa olla monimutkaista erilaisten teknologioiden kirjon johdosta. *Transformaatiolla* tarkoitetaan datan muuttamista yhteneväiseen muotoon. Operaatio pitää sisällään toimenpiteitä kuten tietojen puhdistusta kirjoitusvirheistä, oletusarvojen täydennystä arvon puuttuessa tai duplikaattiarvojen poistamista, tiedon standardointia sekä

summien laskemista. *Lataus* koostuu puolestaan kahdesta vaiheesta. Näitä ovat alkulataus. Latauksen tuloksiin tehdään tarkastuksia, joiden perusteella suoritetaan tiettyjä korjauksia. Korjaavien toimenpiteiden jälkeen suoritetaan korjaava lataus. (Ponniah 2010, ss. 37-38) Yleensä komponentteihin luetaan myös operatiivinen datavarasto (engl. operational data store, ODS), jonka tarkoitus on toimia tiheästi päivitettävänä integroituna operatiivisen datan tallennuspaikkana. Se on kopio operatiivisesta datasta ja toimii operatiivisten raporttien toimittajana taktisessa päätöksenteossa. Tietovarastoista tämä komponentti eroaa siinä, että historiadata säilytetään siellä vain tietyn ajan. (Kimball & Ross 2002; Imhoff et al. 2003, s. 14)

Datan varastointikomponentti on keskeisin palanen koko tietovarastoarkkitehtuurissa. Kun datalle on tehty tarvittavat toimet, se ladataan itse tietovarastoon. Tänä päivänä suuri osa tietovaraston toteutuksista pohjautuu relaatiotietokantateknologiaan tehokkuuden, käytön helppouden ja joustavuuden takia. (Ponniah 2010, s. 152) Riippuen arkkitehtuurityypistä datan varastointikomponentti voidaan mieltää hiukan eri tavoilla. Se voi olla yksittäinen tietokanta tai koostua useista rinnakkaisista tietokannoista, joissa on hyödynnetty relaatiotietokantojen lisäksi muita teknologioita. Kokonaisuus yhdessä muodostaa tietovaraston varastointikomponentin, kuten tulemme seuraavassa alaluvussa näkemään.

Metadata tietovarastossa toimii tiedon katalogina, eikä se poikkea juuri operatiivisten tietokantojen metadatasta. Metadata on dataa tietovarastosta ja siihen kuuluu tieto datan loogisista rakenteista, tiedostoista, niiden osoitteista eli indekseistä, tauluista, sarakkeista ja näkymistä. (Ponniah 2010, ss. 41-42; Gardner 1998, s. 59). Tietovarastojen tapauksessa metadatalla kuvataan lisäksi kyselyitä, raportteja, liiketoimintasääntöjä tai transformaatioon tehtyjä algoritmejä. (Gardner 1998, s. 59) Metadata voidaankin luokitella edelleen kolmeen: operatiiviseen, ETL- ja loppukäyttäjän metadataan. Operatiivinen metadata kuvaa datan lähteille tehdyt operaatiot, kuten tietueiden pilkkomiset ja yhdistelemiset, eri merkistökoodausten käytön ja kenttien pituudet. ETL-metadata puolestaan sisältää tiedon siitä, miten usein lähdejärjestelmiä luetaan, menetelmät lukemiseen ja säännöt operaation tekemiseen. Metadata sisältää myös tiedon kaikista niistä muunnoksista, joita datalle tehdään lastausalueella. Loppukäyttäjän metadata helpottaa tietovaraston käyttöä kääntämällä tietovarastossa käytetyt käsitteet liiketoiminnan käyttämän terminologian muotoon. (Ponniah 2010, ss. 41-42) Imhoff et al. (2003, s. 15) lisäävät tähän hallinnollisen metadatan, jolla kuvataan tietovaraston toimintaan liittyvät tiedot, kuten audit-tiedot, suorituskyky- ja datan laadun metriikat sekä muut statistiikat.

Tietovarastoihin ja sen tietolähteisiin liittyvä metadatan tärkeys monesti aliarvioidaan tietovarastoja kehitettäessä. Syitä tähän on osittain epävarmuus metadatan tallennuspaikasta ja -muodosta ja puute menetelmistä, joilla metadataa saadaan jaettua eri teknologiatoimittajien järjestelmien kesken. (Watson 2000, s. 10) Vetterli et al. (2001) mainitsevatkin, että lähdejärjestelmien ja tietovarastointiprosessin kompleksisuuden hallitsemiseksi on hyödynnettävä johdonmukaista metadatan hallintaa. Tämä tarkoittaa sitä,

että metadata on tallennettu keskitetysti, jotta eri käyttäjät ja ohjelmistokomponentit pääsevät siihen käsiksi. (Vetterli et al. 2000, ss. 68-69) Sen ja Sinha (2005, s. 81) kehottavat metadatan hallintaan siksi, että verrattuna operatiivisten järjestelmien metadataan, tietovaraston tarvitsema metadatan määrä on huomattavasti suurempi.

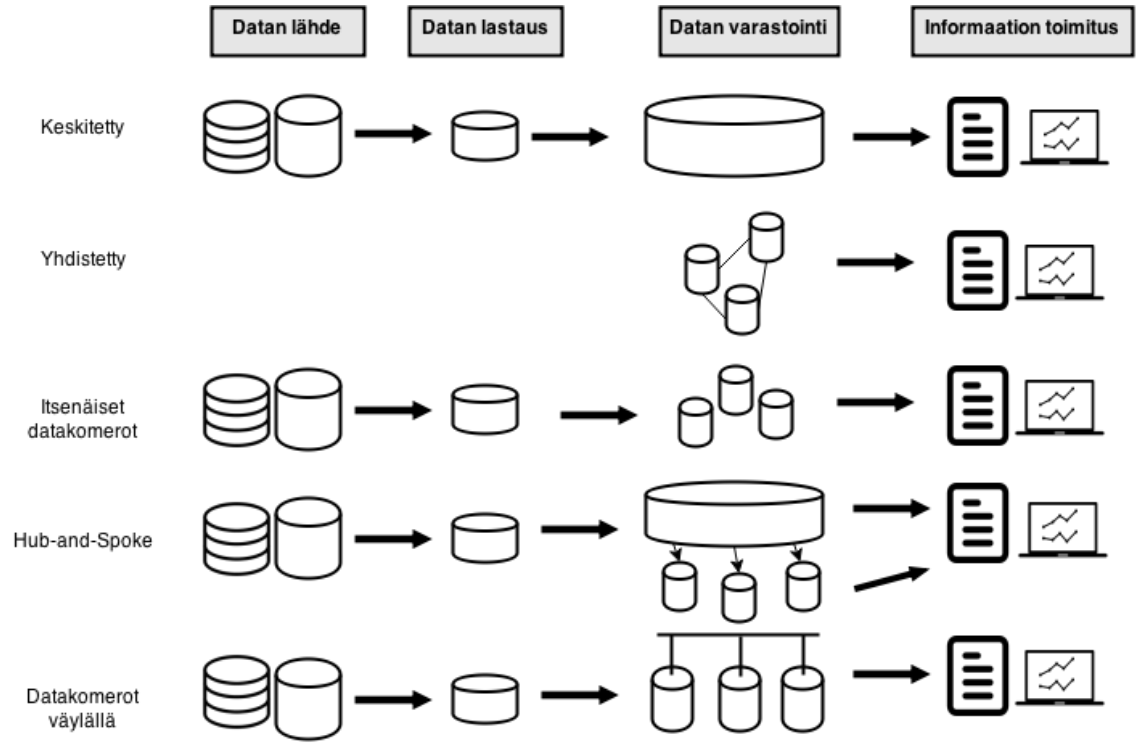
3.2.2 Erilaiset tietovarastoarkkitehtuurit

Tietovarastoarkkitehtuurilla tarkoitetaan pohjakarttaa, joka mahdollistaa tietovarastojen kehityksessä kommunikoinnin, suunnittelun, ylläpidon, oppimisen ja uudelleenkäytön. Se käsittää sisälleen alueita, kuten tiedon mallinnuksen, teknisen suunnittelun ja laitteisto- sekä ohjelmistoinfrastruktuurin suunnittelun. (Sen & Sinha 2005, s. 80) Tietovarastoarkkitehtuurin valinta on yksi tietovarastojen rakentamisen tärkeimmistä päätöksistä. (Silvers 2008, s. 55) Tässä kontekstissa puhuttaessa arkkitehtuurista tarkoitetaan referenssiarkkitehtuuria eli kuvausta, joka ohjaa konkreettisen toteutuksen tekemisessä.

Kirjallisuudessa esiintyy viittä erilaista tapaa toteuttaa tietovaraston arkkitehtuuri. Erilaisia arkkitehtuurityylejä ovat

- Keskitetty tietovarastoarkkitehtuuri
- Itsenäisten datakomeroiden arkkitehtuuri
- Hub-and-Spoke -arkkitehtuuri
- Yhdistetty arkkitehtuuri
- Väylällä integroitu datakomeroiden arkkitehtuuri (Ponniah 2010; Sen & Sinha 2005)

Ariyachandra ja Watson (2010) jaottelevat arkkitehtuurityypit hyvin samalla tavalla, mutta he eivät tee eroa organisaationlaajuisen ja hub-and-spoke -arkkitehtuurin välillä, sillä niissä molemmissa tavoitteena on rakentaa lopulta organisaationlaajuinen tietovarasto. Kuva 4 havainnollistaa eri arkkitehtuurien välisiä periaatteellisia eroja. Keskeisin eroavaisuus niissä on se, miten tiedon varastointi järjestetään. Seuraavaksi esitellään arkkitehtuuriratkaisuja hieman tarkemmin.



Kuva 4. *Eri tietovarastojen arkkitehtuurityypit (mukailtu lähteestä Ponniah 2010)*

Ensimmäinen arkkitehtuurityyppi on **organisaationlaajuinen tietovarastoarkkitehtuurissa** (engl. enterprise data warehouse architecture), joka rakennetaan ottamalla huomioon koko organisaatiotason vaatimukset tiedolle. Tämä pohjalta muodostetaan kokonaisvaltainen infrastruktuuri. Tietovarastoon talletettava data alhaisimmalla granulariteettitasolla sekä usein kolmannessa normaalimuodossa. Toisinaan saatetaan tallettaa summadataa erillisiin tietorakenteisiin. Organisaationlaajuisessa tietovarastossa ei ole erillisiä datakomeroita. (Ponniah 2010, s. 32)

Seuraavana on **itsenäisten datakomeroiden arkkitehtuuri** (engl. independent data mart architecture). Itsenäiset datakomerot ovat toisistaan riippumattomia ja tarjoavat palveluita vain niille suunniteltuun tarkoitukseen, kuten esimerkiksi yksittäisen organisaatioyksikön raportointitarpeisiin. Suunnittelusta puuttuu kokonaisvaltaisuus, joten operaatiot datan saamiseksi kuhunkin datakomeroon saatetaan tehdä toisistaan erillään. (Ponniah 2010, s. 32;156) Samaan tapaan kuin yrityksissä esiintyy vanhoja tietojärjestelmiä, esiintyy myös vanhoja datakomeroita. Eristyneinä tietolähteinään ne eivät kuvaa ”yhtenäistä totuutta” organisaatiosta. (Ariyachandra & Watson 2010, s. 201) Tuloksena tästä on suuri todennäköisyys, että datan määritelmät ja standardit ovat datakomeroiden välillä epäjohdonmukaisia. Tämä heikentää datan analysointia datakomeroiden kesken. (Ponniah 2010, s. 32) Vaikka itsenäiset datakomerot esitellään usein omana arkkitehtuurityyppinään Ariyachandra & Watson (2010, s. 201) toteavat, ettei ole tarkoituksenmukaista rakentaa tällaista ratkaisua, sillä arkkitehtuuri on usein seurausta pyrkimyksestä yhdistää ajan saatossa syntyneitä itsenäisiä tietovarastoratkaisuja.

Hub-and-spoke -arkkitehtuuri on koko organisaationlaajuinen tietovarasto. Atominen data on tallennettu keskustietovarastoon kolmannessa normaalimuodossa. Erona organisaationlaajuiseen tietovarastoon on keskustietovaraston ympärille rakennetut datakomerot, jotka rakennetaan vaihtelevia analyysitarpeita varten. Näitä ovat esimerkiksi liiketoiminnan eri organisaatioyksiköiden tietotarpeet, erikoistetut kyselyt tai tiedonlouhinta. Näin ollen myös datakomeroiden rakenne voi olla sen tarkoituksesta riippuen normalisoitu, denormalisoitu tai summadataa sisältävä. (Ponniah 2010, s. 33) Hub-and-spoke-arkkitehtuuria muodostettaessa huomio on kiinnitetty infrastruktuurin skaalautuvuuden takaamiseksi. Näkökulma ja metodologia arkkitehtuurin muodostamisessa ovat alusta asti organisaatiossa kokonaisuutena eli top-down -periaatteen muikainen. (Ariyachandra & Watson 2010, s. 202).

Väylällä integroidun tietovarastoarkkitehtuurin (engl. data mart bus architecture) synnyn taustalla on prosessi, jossa tietyn liiketoimintaprosessin vaatimusten pohjalta lähdetään kehittämään prosessiin liittyviä tietotarpeita tyydyttävää datakomeroa. (Ariyachandra & Watson 2010, s. 201) Datakomeron dimensiot muodostetaan siten, että tulevaisuudessa rakennettavien datakomeroiden faktataulut käyttävät näitä dimensioita. Dimensiot on niin sanottuja mukautettuja dimensioita (engl. conformed dimensions). Dimensiot muodostavat tavallaan väylän datakomeroiden välille. (Kimball et al. 2011; Ponniah 2010, s. 34; Ariyachandra & Watson 2010, s. 201) Mukautetut dimensiot ovat joko identtisiä keskenään tai alijoukkoja kaikkein laajimmasta ja atomisimmasta dimensiosta. Niillä on keskenään yhdenmukaiset pääavaimet, sarakkeiden nimet, määrittelyt ja arvot. Teknisessä mielessä mukautetut dimensiot voivat olla sama fyysinen taulu kaikkien faktataulujen kesken tai synkronoitu duplikaatti, joka on oma kullakin sitä käyttävällä faktataululla. Myös tietovarastojen faktojen tulee olla mukautettuja, mikä tarkoittaa mitattavien suureiden yhdenmukaista määrittelyä faktataulujen kesken. (Kimball & Ross 2002)

Väylällä integroitujen datakomeroiden arkkitehtuurissa datakomeroista muodostuu looginen ja integroitu kokonaisuus, joka tarjoaa organisaationlaajuisen näkymän koko yrityksen toimintaan. Datakomeroissa tieto on järjestetty usein käyttämällä tähti- tai lumihiutalemallia. Faktojen sisältämä data on pääasiassa atomista, mutta suorituskyvyn parantamiseksi sitä voidaan tallentaa summamuotoiseksi. (Ponniah 2010, s. 35) Kimball et al. (2011) täydentävät, että mukautetuilla dimensioilla saavutetaan väylärakenteen lisäksi se, että ne vähentävät kehityksestä muodostuvia kustannuksia, koska samoja kehitysvaiheita dimensioiden suhteen ei tarvitse enää toistaa. Väylällä integroitu datakomeroiden arkkitehtuuri ja aikaisemmin mainittu hub-and-spoke -arkkitehtuuri ovat käytetyimmät arkkitehtuurivalinnat, sillä ne tarjoavat parhaimman integraation, skaalautuvuuden ja korkea suoritustehon. (Spruijt 2014)

Yhdistetty tietovarastoarkkitehtuuri (engl. federated architecture) koostuu useista erillisistä tietovarastoista tai muista päätöksenteon järjestelmistä, jotka on yhdistetty tiedonhakukerroksella. (Rainardi 2008, s. 39) Data integroidaan joko fyysisesti ja loogi-

sesti käyttämällä erilaisia menetelmiä, kuten jaettuja avaimia, globaalia metadataa tai hajautettuja kyselyitä. Pyrkimyksenä on muodostaa organisaationlaajuinen näkymä tiedosta. (Ariyachandra & Watson 2010, s. 202; Ponniah 2010, s. 159) Yksinkertaisin tapa integroida data lähteiden kesken olisi tehdä se fyysisellä tasolla, mikä tarkoittaa datan kopioimista lähteistä yhteiseen tietovarastoon. Tämä ei usein ole mahdollista eri yksityisyyteen tai teknisiin aspekteihin liittyvien rajoitteiden takia. Näin ollen integrointi tehdään usein loogisella tasolla, mikä on juuri tiedonhakukerroksen rakentamista. (Berger & Schrefl 2008) Integroituun ratkaisuun voidaan päivittää data kuitenkin vain kaikkein abstrakteimman datan mukaan, joka heikentää sen kykyä monipuoliseen analyysiin. (Rainardi 2008, s. 39) Tämä arkkitehtuurityyppi on periaatteessa sama, kuin itsenäiset datakomerot, mutta ne kaksi erottaa datakomeroiden välinen integraatio. (Ponniah 2010, s. 159)

3.2.3 Arkkitehtuurin valinta

Organisaation ollessa tietoinen arkkitehtuurin valintaan vaikuttavista tekijöistä, voi se ymmärtää tai jopa kontrolloida päätöstä valinnasta. (Ariyachandra & Watson 2010) Tutkimuksen kontekstissa, jossa tietovarastoa rakentava yritys toimittaa sitä toiselle toimijalle, on myös hyvä tunnistaa päätökseen vaikuttava tekijät, jotta voidaan tunnistaa näitä piirteitä asiakaskunnasta ja muodostaa ratkaisu, joka palvelee asiakaskunnan tarpeita ja ottaa huomioon rajoitteet.

Ariyachandra ja Watson (2010) tarkastelevat tutkimuksessaan arkkitehtuurin valintaan vaikuttavia seikkoja. He havaitsivat, että tietovaraston strategisuus, IT-henkilöstön osaaminen ja resurssien rajoitteet ovat kolme merkittävintä tekijää arkkitehtuurin valinnassa. Spruijt (2014, s. 56) puolestaan tunnisti tutkimuksessaan valintaan vaikuttaviksi tekijöiksi arkkitehtuurin ominaisuuksia, kuten ylläpidettävyys, kyky toistuviin päivityksiin, hyvä suorituskyky, kyky käsitellä suurta määrää dataa, integroitavuus ja skaalautuvuus. Ariyachandra ja Watson (2006) määrittelevät arkkitehtuurin valinnan onnistuneisuutta puolestaan tietovaraston tarjoaman informaation tarkkuudella, täydellisyydellä ja yhdenmukaisuudella sekä järjestelmän joustavuudella, skaalautuvuudella ja integroitavuudella. He katsoivat vaikuttaviksi tekijöiksi myös yksityisten käyttäjien suorittamien datan hakujen nopeuden ja helppouden sekä tietovarastoratkaisun kyvyn täyttää sille asetetut liiketoimintavaatimukset. (Ariyachandra & Watson 2006, s. 6)

Valittaessa organisaationlaajuisen ja itsenäisten datakomeroiden arkkitehtuurin tai väylällä integroitujen datakomeroiden välillä strateginen näkökulma on dominoiva tekijä, sillä kun hanke nähdään strategiseksi, allokoidaan sille resursseja ja valinta on todennäköisesti organisaationlaajuinen arkkitehtuuri. (Ariyachandra & Watson 2010, s. 208) Kun vaaditaan nopeaa kehitysaikaa, alhaisia kustannuksia sekä alhaista monimutkaisuutta, valinta on usein itsenäiset datakomerot. (Spruijt 2014, s. 56) Ariyachandra ja Watson (2006) toteavat kyselynsä tulosta perusteella, että vaikka itsenäisten datakome-

roiden arkkitehtuuria on yrityksissä käytössä, on se informaation ja järjestelmän laadun näkökulmasta mitattuna huono ratkaisu.

Väylällä yhdistetyt datakomerot tähtäävät ennen pitkää organisaationlaajuisen tietovaraston rakentamiseen, eivätkä vaadi alkuun organisaationlaajuista näkökulmaa ja strategista merkitystä tai suuria alkuinvestointeja. (Ariyachandra & Watson 2010, s. 209) Se on eräänlainen kompromissi organisaationlaajuisen ja itsenäisten datakomeroiden arkkitehtuurin välillä, kun huomioidaan tietovarastojen mahdollisuus tarjota yksi versio yrityksen tilasta. Väylällä yhdistetyt datakomerot on hyvä tapa integroida useita datan lähteitä vähitellen yhdeksi integroiduksi kokonaisuudeksi. (Spruijt 2014, s. 56)

Yhdistettyyn tietovarastoarkkitehtuuriin päädytään usein esimerkiksi yritysostojen, yhdistymisten ja uudelleenorganisointien seurauksena (Ariyachandra & Watson 2010, s. 202) tai kun ei ole taloudellisesti järkevää hylätä olemassa oleviin järjestelmiin tehtyjä panostuksia ja aloittaa uudestaan puhtaalta pöydältä (Ponniah 2010, s. 33). Sinänsä tämäntyyppistä arkkitehtuuria ei ole tarkoituksenmukaista rakentaa, mikäli alkutilanteessa yrityksellä ei ole tietovarastoa. Toisaalta Spruijt'n (2014, s. 56) tutkimuksen perusteella yhdistetty arkkitehtuuri on monesti vain väliaikainen ratkaisu tietovarastointiongelman ratkaisuun.

Organisaationlaajuinen, väylällä yhdistetty datakomeroiden ja hub-and-spoke- arkkitehtuuri ovat Ariyachandran ja Watsonin (2006) sekä Spruijt'n (2014) mukaan käytetyimpiä, sillä ne ovat muokkautuneet ajan mittaan periaatteiltaan samankaltaisiksi ja koska niitä on mahdollista kehittää aloittaen pienestä mittakaavasta. Molemmissa fokus on pitkällä tähtäimellä organisaationlaajuinen ratkaisu.

3.2.4 Arkkitehtuurien tasoista

Käsiteltäessä järjestelmien arkkitehtuureita käytetään usein geneeristä termiä arkkitehtuuri kuvaamaan tietoteknisen järjestelmän rakennetta ja sen komponentteja. Tällä ilmaisulla viitataan usein sovellusarkkitehtuuriin, joka on kuitenkin riittämätön kuvaamaan ohjelmistojen ympärilleen muodostamaa teknistä kokonaisuutta. (Vasconcelos et al. 2007, s. 91).

Monipuolisemmin tähän antaa tukea tietojärjestelmäarkkitehtuuri (engl. information system architecture), joka samaan tapaan määritellään kuvauksena komponenttien rakenteesta, niiden välisestä suhteesta ja periaatteista. (Garlan et al. 1995) Se on tarkkuudeltaan kuitenkin hiukan korkeamman tason kuvaus ja sillä pyritään auttamaan enemmän järjestelmän liiketoiminnallisten tavoitteiden kuvaamisessa. (Maes et al. 2000) Ohjelmisto- ja tietojärjestelmäarkkitehtuuri ovat osa suurempaa kokonaisuutta, jossa kolmantena tekijänä voidaan tunnistaa kokonaisarkkitehtuuri. (engl. enterprise architecture). (Vasconcelos et al. 2007, s. 91) Vaikka ohjelmistoarkkitehtuurissa voidaan kuvata järjestelmää eri näkökulmista vaihtelevilla näkymillä, eivät ne ota kantaa esimerkiksi tiedon rakenteisiin tietojärjestelmissä.

Tietojärjestelmäarkkitehtuuri koostuu kolmesta eri tasosta tai toimialueesta, joita ovat informaatio tai tieto-, sovellus- ja teknologinen arkkitehtuuri. Informaatioarkkitehtuuri kuvaa pääasialliset datatyypit, jotka tukevat liiketoiminnan suorittamista. Sovellusarkkitehtuuri puolestaan määrittelee sovellukset, joita tarvitaan datan hallintaan ja liiketoiminnan tukemiseen. Teknologinen arkkitehtuuri havainnollistaa sovellusten toteutukseen käytettyjä teknologioita ja sitä infrastruktuuria, jonka varaan sovellusten toteutusympäristö rakentuu. (Spewak et al. 1993)

Kuten tutkimuksessa tullaan myöhemmin huomamaan, suunnittelussa on otettava kantaa kaikilla edellä mainittujen arkkitehtuurien tasoilla, joten on olennaista tunnistaa kunkin osa-alueelle kuuluvat kuvaukset. Toisaalta edelliset määritelmät kuvauksista ovat melko rajoittavia. Esimerkiksi tietovaraston tapauksessa on tarve kuvata muutakin kuin pelkät tietotyypit ja käytettävät arvot. Näin ollen tarkastelutasoa on laajennettava merkittävästi. Arkkitehtuurin tasot antavat kuitenkin kuvan kokonaisuudesta, jonka tasoihin tietovaraston kehityksessä tulee ottaa kantaa. Tieto- tai informaatioarkkitehtuurin sisältöä ja vaatimuksia käsitellään seuraavassa alaluvussa. Ponniah (2010, s. 142) kuvaakin tietovarastoarkkitehtuuri kokonaisvaltaiseksi pohjakartaksi, joka määrittelee standardit, mittarit, yleisen rakenteen ja toteutusta tukevat teknologiat.

3.3 Tiedon mallinnus

Tiedon mallinnus on olennainen osa tietovarastointia. Tässä kappaleessa käydään läpi erilaisia tietomalleja tietovarastoiden toteuttamisessa sekä tietomallien tasoja.

3.3.1 Erilaiset tietomallit tietovarastoissa

Tietovarastojen kirjallisuudessa on kaksi vallalla olevaa käsitystä siitä, miten tietoa tulisi mallintaa tietovarastoissa. Edellä esitellyt arkkitehtuurityypit ovat myötävaikuttamassa siihen, mitä mallinnustapaa tietovaraston rakennukseen sovelletaan.

Mallinnustyyppejä ovat dimensionaalinen[1] ja kolmannessa normaalimuodossa oleva data (Silvers 2013, s. 5). Normalisoidussa muodossa olevan tietovaraston suunnittelu noudattaa rakentamisessaan samoja periaatteita, joita käytetään tavallisten operatiivisten järjestelmien tietokantojen muodostuksessa. Siksi se on ollut tietovarastojen alkuaikoina varsin käytetty mallinnustapa. Normalisoinnilla pyritään datan redundanssin poistoon. (Sen & Sinha 2005, s. 80; Silvers 2008) Dimensionaalisessa tietomallissa normalisoinnin poistetaan ja haetaan suorituskykyyn liittyviä hyötyjä. Dimensionaalisessa tietomallissa kokonaisuus koostuu fakta- ja dimensiotauluista. (Sen & Sinha 2005, s. 80) Faktat ovat päätöksenteossa kiinnostuksen kohteena olevia analysoitavia suureita, kuten myytyjen tuotteiden määriä tai tuotteiden hintoja. (Golfarelli 2010, s. 1; Chaudhuri & Dayal

[1] Dimensionaalinen tietomalli on toisessa normaalimuodossa (2NF), mutta siitä käytetään yleisimmin nimitäystä dimensionaalinen. (Kimball et al. 2011) Tässä tutkimuksessa kolmannessa normaalimuodossa olevasta datasta käytetään yksinkertaisuuden vuoksi normalisoitu tietomalli.

1997, s. 520) Dimensiot puolestaan ovat faktojen kontekstiin liittyvää tietoa. Faktaan liittyvät dimensiot näin ollen yksikäsitteisesti määrittelevät käsiteltävän suureen ja täydentävät faktataulua kontekstiin liittyvällä tiedolla. Dimensionaalisuuteen perustuvia tietomalleja ovat esimerkiksi tähtimalli (engl. star schema) ja lumihiutalemalli (engl. snowflake schema) (Chaudhuri & Dayal 1997, s. 520; Sen & Sinha 2005, s. 80).

Analyysiin suorittavien kyselyiden kannalta tarkasteltuna normalisoitu malli vaatii useampia liitoksia kuin dimensionaalinen malli. (Silvers 2008, s. 91) Monimutkaista analyysia suorittavat kyselyt vaativat monia liitoksia ja tulosten yhteen laskemista, joita dimensionaalisessa mallissa ei tarvita samalla tavalla kuin normalisoidussa datassa. Liitosoperaatiot ovat raskaita ja vaativat hyvää suorituskykyä järjestelmältä. Normalisoinnin poistaminen laskee suorituskykyvaatimusta tällaisissa tapauksissa, joten se on yleisimmin käytetty tiedon mallinnustapa tietovarastoissa. (Sen & Sinha 2005, s. 80) Inmonin (2005) mukaan puolestaan normaalimuotoinen tietomalli on optimaalinen ratkaisu tietovarastojen toteutukseen, koska se on joustava, sopii atomisen datan käsittelyyn eikä ole optimoitu tiettyjen prosessointivaatimusten mukaan. Tietovarastojen tulisi palvella mahdollisimman laajaa asiakaskuntaa ja tietotarpeita, mikä on päällimmäisenä pyrkimyksenä organisaationlaajuisissa tietovarastoissa. Tähän vastaa Inmonin (2005) mukaan paremmin normaalimuotoinen data. Dimensionaalinen malli tyydyttää tietotarpeita kapeammalta alueelta ja on näin ollen optimaalisempi ratkaisu datakomeroiden toteutukseen. (Inmon 2005)

Inmon (2005) toteaa, että laajan käyttäjäkirjon lisäksi tietovaraston tulisi tukea vielä tuntemattomia tietotarpeita ja käyttäjiä. Hänen mukaan datan atomisuus takaa, että siitä voidaan tehdä ääretön määrä yhdistelmiä. Kimball et al. (2011) toteavat puolestaan, että dimensionaaliset mallit, joita hub-and-spoke -arkkitehtuurin datakomeroissa hyödynnetään, mukautuvat sulavasti odottamattomiin muutoksiin datassa. Mukautuminen tapahtuu lisäämällä sarakkeita, jolloin muodostettuja kyselyitä tai raportteja ei ole välttämättä tarpeen muokata. (Kimball et al. 2011, s. 237)

3.3.2 Tietomallien tasot

Tiedon mallinnuksessa tuotettavat mallit voidaan jakaa kolmeen tasoon. Näitä ovat käsitemalli, looginen malli ja fyysinen malli, joissa tarkkuustaso kehittyy siirryttäessä käsitemallista kohti fyysistä mallia. (Inmon 2005; Rizzi et al. 2006, s. 3; Silvers 2008, s. 55) Mallinnustekniikassa normalisoidun ja dimensionaalisen tapauksen välillä ei ole merkittäviä eroja. Tekijä, joka mallinnuksessa kuitenkin erottaa menetelmät on tiedon normaalisuus, joka poistaa dimensionaalisesta malinnuksesta tarpeen ylimääräiselle prosessivaiheelle. (Rizzi et al. 2006, s. 3) Toinen seikka tietovarastoihin liittyvien mallien erottavana tekijänä on se, että niiden tulee tukea arvojen yhteen laskemista eli aggregointia. (Chaudhuri & Dayal 1997, s. 520) Lisäksi tietovarastojen mallinnukseen sisällytetään kuvaus ETL-prosessista (Malinowski & Zimányi 2008).

Käsitemalli on tietovarastojen vaatimusmäärittelyn pohjalta toteutettu kuvaus tulevasta tietovaraston rakenteesta. Se toimii perustana niin tietovarastoon suoritettaville analyysitehtäville kuin työvälineenä tietovaraston tulevaisuuden evoluution toteuttamisessa. (Malinowski & Zimányi 2008, s. 271) Halpin (2008) lisää tähän kommunikoinnin mahdollistaminen eri osapuolten, kuten mallintajien ja sovellusalueen asiantuntijoiden, välillä. Käsitemallissa kuvataan korkealla tasolla ominaisuudet ja niiden välillä vallitsevat suhteet. Rajausta, joka määrittelee, mitkä käsitteet kuuluvat malliin ja mitkä eivät, kutsutaan integroinnin rajaukseksi (engl. scope of integration). (Inmon 2005) Käsitteellinen malli pyrkii toteutustavasta ja siihen liittyvistä ongelmista riippumattomaan kuvaukseen. Malliin katsotaan joskus kuuluvaksi tietoarkkitehtuurin lisäksi myös kuvauksen ETL-prosessista. Kuvaus tehdään useimmiten laajentamalla luokkakaaviota tai muita UML-kuvaustekniikoita tai kehittämällä muita tarpeeseen sopivia ratkaisuja. (Rizzi et al. 2006, s. 4) Myös ETL-prosessin kulkua ja mappauksia voidaan samaan tapaan mallintaa UML:n erilaisilla laajennoksilla. Siitä ei kuitenkaan ole kehitetty yleisesti hyväksyttyä ja laajalti käytettyä notaatiota. (Vassilidis et al. 2002)

Looginen malli toteutetaan käsitemallin pohjalta. Käsitemalli muutetaan siis muotoon, josta se voidaan toteuttaa ja optimoida halutun tyyppisessä tietokannassa. (Rizzi et al. 2006, s. 5) Loogisessa mallissa kuvataan pää- ja vierasavainkandidaatit, attribuutit sekä riippuvuussuhteet mallin eri osien välillä (Inmon et al. 2008, s. 159). Tietokantatoteutuksen käytetyimpiä tyyppejä ovat relaatio- ja moniulotteiset tietokannat sekä joskus oliotietokannat. Relaatiotietokannoissa tieto tallennetaan joko normalisoituun tai dimensionaaliseen muotoon, kun taas moniulotteisissa tietokannoissa tietorakenteita ovat esimerkiksi kuutiot. (Rizzi et al. 2006, s. 5; Halpin 2008)

Fyysinen malli muodostetaan edelleen keskitason loogisen mallin pohjalta laajentamalla sitä konkreettisilla pää- ja vierasavaimilla. Fyysinen malli muistuttaa siis sitä tietokantarakennetta, joka tietovarastoon on tarkoitus rakentaa. Siihen pitää kuitenkin tehdä optimointia, ennen kuin se voidaan toteuttaa itse tietovarastoon. (Inmon 2005) Fyysisen suunnittelun päätavoitteena onkin optimaalisten indeksien löytäminen loogisen mallin ja arvioidun työkuorman perusteella. Optimaalisimmat indeksit ovat niitä, joiden kanssa tietovarastoon kohdistettavat kyselyt ovat tehokkaimmat. (Rizzi & Golfarelli 1998, s. 9) Tärkeitä päätöksiä tässä on myös tietokantaan syötettävän datan granulariteetti ja mahdollinen partitiointi. (Inmon 2005)

3.4 Tietovaraston kehitys

Tässä kappaleessa käsitellään tietovaraston kehitystä prosessinäkökulmasta. Aluksi aihepiiriä käsitellään yleisesti, minkä jälkeen siirrytään prosessin, sen vaiheiden ja niiden sisällön käsittelyyn.

3.4.1 Yleistä tietovaraston kehityksestä

Tietovarastojen kehittäminen on monimutkainen ja kallis prosessi. Perinteiseen ohjelmistojen kehitysprojektiin verrattuna siinä on useita samoja näkökulmia ja se vaatii samalla tavalla kehitykseen liittyvien aktiviteettien määrittelyä. Perinteisiä vaiheita tietojärjestelmän kehityksessä ovat muun muassa vaatimusten kerääminen, suunnittelu ja toteutus. Vaikka ohjelmistokehityksestä on kirjoitettu useita julkaisuja, harvat niistä ovat keskittyneet yksinomaan tietovarastojen kehittämiselle. Näistä useimmat ovat käytännön harjoittajien kirjoittamia ja perustuvat heidän kokemuksiinsa tietovarastojen rakentamisessa. Tieteellinen yhteisö on tarjonnut joitakin lähestymistapoja kehitykseen, mutta menetelmät ovat liian monimutkaisia käytettäväksi todellisessa liiketoimintaympäristössä. (Malinowski & Zimányi 2008, s. 251) Tässä tutkimuksessa ei pyritä vertailemaan kaikkia mahdollisia akateemisessa kirjallisuudessa esiintyviä viitekehyksiä ja etsimään niistä parasta vaihtoehtoa vaan esittelemään yleisimpiä ominaisuuksia käytetyimmistä menetelmistä.

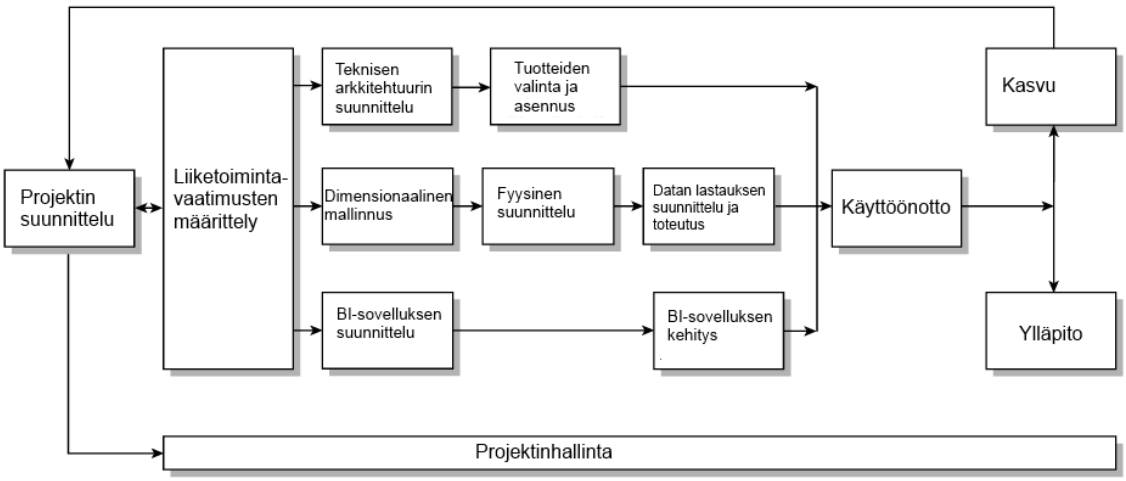
Ponniah (2010) mainitsee, että kehitystä voidaan lähestyä kahdesta eri näkökulmasta, joita ovat niin sanotut bottom-up - ja top-down -näkökulmat. Top-down-lähestymistavassa fokuksena on yritys kokonaisuutena. Vaatimuksen tietovarastolle kerätään eri puolilta organisaatiota, minkä perusteella pyritään muodostamaan koko käyttäjäkunnan tietotarpeita palveleva kokonaisuus. (Malinowski & Zimányi 2008, s. 253) Esimerkiksi hub-and-spoke- arkkitehtuurin mukaisen tietovarastoratkaisun toteuttaminen noudattaa tätä lähestymistapaa. (Inmon 2005; Sen & Sinha 2005; Ariyachandra & Watson 2010) Inmon (2005) kuitenkin tähdentää, ettei tarkoituksena ole kerralla rakentaa koko suunniteltua organisaationlaajuista arkkitehtuuria vaan sitä tulee lähestyä pieniä askeleita käyttäen.

Bottom-up -lähestymistapa noudattaa periaatetta, jossa tarkastelun kohteeksi otetaan pienempi kokonaisuus yrityksestä. Sinä toteutetaan ensin datakomeroita tietyille mielenkiinnon kohteille. Datakomeroiden sisältämä data saattaa olla sekä alhaisella tarkkuustasolla että summamuodossa riippuen analyysin tarpeista. Datakomerot yhdistetään dimensioiden avulla. (Ponniah 2010, s. 31) Väylällä integroitujen datakomeroiden arkkitehtuuri rakennetaan juuri tämän periaatteen mukaisesti. (Kimball & Ross 2002; Sen & Sinha 2005; Ariyachandra & Watson 2010)

Organisaationlaajuisen tietovaraston suunnittelu ja toteutus top-down -lähestymistapaa käyttäen on kustannuksiltaan ja kestoltaan ylivoimainen ponnistelu monille yrityksille, joten monet päätyvätkin bottom-up -periaatteen omaksumiseen. (Malinowski & Zimányi 2008, s. 253) Inmon (2005) toteaa tämän olevan merkittävin syy esimerkiksi itsenäisten datakomeroiden suosioon, jolloin ei kuitenkaan välttämättä saavuteta yhtenäistä kuvaa organisaation tilasta. (Inmon 2005) Toinen tapa välttää suuret alkuinvestoinnit, mutta samalla saada organisaationlaajuisen tietovaraston kautta saavutetut edut on noudattaa Ponniah'n (2010) esittelemää praktista lähestymistapaa. Siinä yhdistellään

piirteitä molemmista edellä mainituista periaatteista. Siinä suunnittelu pohjautuu top-down -periaatteen tavoin kokonaiskuvaan yrityksestä ja eri organisaation osista ja tasoilta tunnistettuihin vaatimuksiin. Vaatimusten pohjalta määritellään ympäröivä arkkitehtuuri tietovarastolle, minkä jälkeen määritellään kunkin datakomeron sisältö. Tämän jälkeen ryhdytään kehittämään niitä yksitellen huolellisesti määritellyn suunnitelman kautta. (Ponniah 2010, ss. 31-32) Väylällä integroitujen datakomeroiden arkkitehtuuri on esimerkkitapaus tämäntyyppisestä lähestymistavasta.

Itse tietovaraston toteutusprojekti voidaan silti jakaa perinteisen järjestelmien kehityksen elinkaarimallin mukaan karkeasti kuuteen eri vaiheeseen. Niitä ovat projektin suunnittelu, vaatimusten määrittely, suunnittelu, toteutus, käyttöönotto sekä ylläpito ja jatkokehitys. Merkittävä seikka, joka tietovaraston kehityksen erottaa perinteisestä ohjelmistokehityksestä, on suunnittelun ja rakentamisen jakautuminen kolmeen erilliseen prosessiin. (Ponniah 2010, s. 83) Kuva 5 havainnollistaa tilannetta.



Kuva 5. *Tietovaraston kehityksen elinkaarimalli (mukailtu lähteestä Kimball et al. 2011)*

Elinkaarimalli kuvaa eri tehtävien ja komponenttien välisen asianmukaisen yhdistämisen, joka on tärkeää tietovarastointiprojektin onnistumisen kannalta. Ei riitä, että käytössä on parhaat teknologiat, vaan nämä korkean tason tehtävät on kyettävä tekemään oikeassa järjestyksessä ja oikeaan aikaan. (Kimball et al. 2011, ss. 3-4) Suunnittelun ja toteutuksen kolme eri ”rataa” voidaan suorittaa rinnakkain, mutta tietyt tehtävät tulee olla tehtynä rinnakkaisella radalla, ennen kuin voidaan siirtyä toteuttamaan jotakin tiettyä vaihetta.

Ratojen sisältöä kuvaa niin ikään Ponniah’n (2010) tietovarastoarkkitehtuurin kolmen osa-alueen jako datan hakemiseen, varastointiin ja informaation toimittamiseen. Kimball et al:n (2011) käyttävät radoista nimityksiä datarata, teknologiarata ja liiketoimintatiedon hallinnan sovellusten rata. Malliin he lisäävät itse projektia tai ohjelmaa koko sen

elinkaaren ajan monitoroivan ja kontrolloivan johtamisprosessin, joka näkyy kuvassa 5 alimmaisena. (Kimball et al. 2011, s. 4)

Teknologiaradalla suoritettavat toimenpiteet tähtäävät teknologisen arkkitehtuurin suunnitteluun ja sen toteuttavien tuotteiden valintaan. Dataradalla käsitteellisen mallin pohjalta muodostetaan fyysinen malli ja toteutetaan datan tuonti lähdejärjestelmistä tietovarastoon. Liiketoimintatiedon hallinnan sovellusten radan toimenpiteet keskittyvät päätöksentekoa tukevien sovellusten valintaan. (Kimball et al. 2011, ss. 5-7) Mainittakoon, että tämän tutkimuksen kannalta olennaisia ovat kaksi ensimmäisenä mainittua. Näin ollen liiketoimintatiedon hallintaa kuvaileva prosessi jätetään pois käsittelystä. Elinkaaren vaiheista rajauksen puitteissa jätetään pois käyttöönoton jälkeiset vaiheet.

3.4.2 Projektin suunnittelu

Ensimmäisenä elinkaarimallissa on projektin suunnittelu. Projektin suunnitteluun liittyy Kimball et al:n (2011) elinkaarimallin mukaan projektin määrittely, suunnittelu ja hallinta. Projektina ja sen hallitsemisen näkökulmasta ei tietovarastoprojekti eroa tavallisesta tietojärjestelmäkehityksen toimenpiteistä merkittävästi. Kuitenkin läpinäkyvyyttä vaativat monialaiset projektitiimit asettavat lisävaatimuksensa sen seurannalle, dokumentaatiolle, rajauksen hallinnalle sekä kommunikoinnille. (Kimball et al. 2011, s. 58)

Määrittelyn ensimmäisessä vaiheessa tulisi arvioida omat valmiudet tietovaraston kehittämiseen. Kirjoittajat määrittelevät kolme kriteeriä, jotka on vähintään oltava kunnossa, ennen projektin toteuttamista. Ensinnäkin tietovarastohankkeella on oltava vahva tuki ja kannattaja eli sponsori liiketoiminnan näkökulmasta. Toisekseen tietovarastot ovat onnistuneita, jos ne hyväksytään sopivina työkaluina liiketoiminnan päätöksenteossa. Kolmas arvioitava tekijä ennen hankkeen aloittamista on datan käyttökelpoisuus. Jos päätöksentekoon käytettävät data on liian epäpuhdasta, liian monimutkaista tai sitä ei ole edes kerätty, on edessä mittavia toimenpiteitä. (Kimball et al. 2011, ss. 16-17)

Projektin määrittelyssä seuraava vaihe on alustavan rajauksen luominen ja hyväksyttäminen sekä liiketoimintamallin rakentaminen. (Kimball et al. 2011, s. 22) Riippuen onko kehityksen lähestymistapa top-down vai bottom-up rajauksen koko vaihtelee. Rajauksen valmistuttua se dokumentoidaan ja dokumentista on käytävä ilmi seuraavat seikat: olennaiset liiketoimintavaatimukset, odotettu data ja kohdekäyttäjät, mukaan kuuluvat sidosryhmät, onnistumiskriteerit sekä olettamukset ja riskit. (Kimball et al. 2011, ss. 22-27)

Seuraava kokonaisuus on projektin suunnittelu. Ensimmäisenä tulisi määritellä sopivat roolit projektin toteuttamiseen. Roolit voidaan jakaa liiketoiminnan tukijoihin, valmentajiin, kehitystiimiin, erikoistiimiin sekä sekalaisiin henkilöihin, kuten konsultteihin. Roolien määrittelyssä tulee huomata, että yksi henkilö voi edustaa useampaa kuin yhtä roolia samanaikaisesti. (Kimball et al. 2011, ss. 33-40) Tietovarastoja kehitettäessä kannattaa siis muistaa, että osaamista tarvitaan laajalta alueelta, jolloin monialaista osaa-

mista sisältävät tiimit ovat olennaisia. Ponniah (2010, s. 80) mainitseekin, että yksi huomioonotettava tekijä tietovarastoprojekteissa on verrattuna operatiivisten järjestelmien kehityksen vaatimus laajalle osaamiselle, jolloin myös tarvittaessa organisaation ulkopuolista apua saatetaan tarvita.

Projektin suunnittelun tuloksena syntyy lisäksi konkreettinen projektisuunnitelma sekä kommunikointisuunnitelma. Projektisuunnitelman tulee määritellä suoritettavat tehtävät riittävän tarkalla tarkkuustasolla. Kommunikaatiosuunnitelman tarkoitus on mahdollistaa käytäntö, jossa projektipäällikkö kuulee eri sidosryhmien tarpeita ennaltaehkäisevästi. Sen tarkoituksena on myös hallita odotuksia, joka on yksi tietovarastoprojektin edellytyksistä onnistumiselle. (Kimball et al. 2011, ss. 40-44) Projektin suunnittelun ja liiketoimintavaatimusten kerääminen voi olla osittain rinnakkaisia, koska vaatimusten keräämisen aikana voidaan joutua palaamaan tietovaraston rajaukseen tai projektisuunnitelmaan. (Kimball & Ross 2002)

3.4.3 Vaatimusten kerääminen ja määrittely

Kimball et al:n (2011) elinkaarimallin seuraava vaihe on kuvan 5 mukaisesti liiketoimintavaatimusten määrittely. Tietovarastolle määritellyillä vaatimuksilla on vaikutus läpi tietovaraston suunnittelun ja toteutuksen. Vaatimusten määrittely vaikuttaa aina oikean rajauksen muodostamisesta, oikean datan mallinnukseen, oikeiden työkalujen valintaan, transformaatiosääntöjen, analyysien rakentamisen ja käyttöönoton tuen tarjoamiseen. (Kimball et al. 2011, s. 63)

Olemassa olevat menetelmät vaatimusten keräämiseen voidaan karkeasti jaotella kolmeen kategoriaan: datavetoiset (engl. data-driven), tavoitevetoiset (engl. goal-driven) ja käyttäjävetoiset menetelmät. (List et al. 2002, s. 204 - 205; Golfarelli 2010, s. 4) Datavetoisen vaatimusmäärittelyn perusajatus on, että tietovaraston kehityksen lähtökohta on yrityksessä oleva data, ei niinkään vaatimukset tai tietotarpeet. Käyttäjät ymmärtävät tietotarpeet vasta, käyttäjät pääsevät suorittamaan hakuja tietovarastoon. (List et al. 2002, s. 204 - 205) Tietovaraston tietomalli muodostetaan näin ollen analysoimalla lähdejärjestelmien sisältämää dataa. (Malinowski & Zimányi 2008, s. 255) Tämä menetelmä on käyttökelpoinen, kun yksityiskohtaista tietoa datan lähteistä on saatavilla, lähdedata on normalisoidussa muodossa ja datan lähteiden tietokantaskeemat eivät ole monimutkaisia. (Giorgini et al. 2005)

Tavoite- tai liiketoimintavetoisessa vaatimusmäärittelyssä taustalla on ajatus, että järjestelmien käyttäjät eivät kykene selkeästi muodostamaan tarkkoja vaatimuksia tietovarastolle. Tietorakenteiden muodostaminen lähtee liikkeelle liiketoimintavaatimusten ja -prosessien analysoinnista. Liiketoimintaprosessien analysoinnissa täytyy määritellä liiketoiminnan aktiviteetit, joista syntyy tietty tuotos. (Malinowski & Zimányi 2008, s. 255)

Käyttäjävetoisessa vaatimusmäärittelyssä käyttäjillä on keskeinen rooli vaatimusten määrittelyssä. Heidät valitaan organisaation eri tasoilta. (Malinowski & Zimányi 2008, s. 255) Käyttäjien avustuksella määritellään liiketoiminnan tavoitteet tärkeimpien kysymysten kautta, minkä jälkeen ne priorisoidaan ja aikaansaannoksen pohjalta muodostetaan rakenne tietovaraston datalle. (List et al. 2002, s. 205)

3.4.4 Suunnittelu ja toteutus

Yleisesti suunnittelusta sanoa, että vaikka prosessimallissa tietovaraston suunnittelu etenee vaihe toisensa jälkeen, todellisuudessa edeltäviin vaiheisiin saatetaan joutua palaamaan. Kuvassa 5 esitellyssä elinkaarimallissa vain suunnittelun ja vaatimusten määrittelyn välinen nuoli on kaksisuuntainen. Jälkimmäisiin vaiheisiin palaamista tapahtuu ennen kaikkea iteratiivisuutta hyödyntävissä viitekehyksissä, joissa tietovarastosta kehitetään iteraatioittain toiminnallisuudeltaan kohennettu versio. (Malinowski & Zimányi 2008, s. 256) Kuten kuvasta 5 voidaan havaita, suunnittelu ja toteutus jakautuvat rinnakkaisille radoille. Näistä käsitellään seuraavaksi kahta ylimmäistä: teknologia- ja datarataa.

TEKNOLOGIARATA

Teknologiaradalla kulkevan prosessin tarkoitus on lyhyesti ilmaistuna suunnitella tietovaraston tekninen arkkitehtuuri ja valita tuotteet, joilla arkkitehtuuri saadaan toteutettua. (Kimball et al. 2011, s. 179) Arkkitehtuurin suunnittelun piiriin kuuluvat osa-alueet ovat aikaisemmin tässä tutkimuksessa esitellyt tietovarastoarkkitehtuurin lastaukseen, varastointiin, metadataan ja tiedon toimittamiseen tarkoitetut komponentit. (Ponniah 2010, ss. 146-147)

Ensimmäisenä vaiheena prosessissa on kuvan 6 mukaisesti teknisen arkkitehtuurin suunnittelu. Kimball et al. (2011, s. 183) ovat hahmotelleet kahdeksanvaiheisen prosessin teknisen arkkitehtuurin luomiseksi. Sen vaiheita ovat arkkitehtuuritiimin kokoaminen, arkkitehtuuriin liittyvien vaatimusten kerääminen, arkkitehtuurivaatimusten dokumentointi, korkean tason arkkitehtuurimallin kehittäminen, alijärjestelmien määrittely ja suunnittelu, arkkitehtuurin toteutuksen vaiheiden määrittely, itse arkkitehtuurin dokumentointi sekä arkkitehtuurin katselmointi ja viimeistely. (Kimball et al. 2011, s. 183)

Meister et al. (2004) ehdottavat suunnittelumallien (engl. design pattern) soveltamista arkkitehtuurin suunnittelemisessa varsinkin, kun on kyseessä tuoterunkoon perustuvan sovelluksen rakentaminen. Suunnittelumallin käyttämiseen he mainitsevat hyödyksi esimerkiksi arkkitehtuurin ajan myötä tapahtuvan evoluution tukemisen. *Kirjoittajien tutkimuksessaan kehittämä tuoterunkoon perustuva datan analyysisovellus tukee muun muassa sovelluksen laajentumista esimerkiksi erikoistettuilla tilastollisilla sovellutuksilla, erilaisia lähestymistapoja datan prosessointiin vaihtelevissa teknisissä ympäristöissä sekä skaalautuvuutta käyttäjien määrän kasvun suhteen.* Tuen näille ominaisuuksille kirjoittajat ovat saavuttaneet hyödyntämällä niin sanottua Presentation-Abstraction-

Control (PAC)-arkkitehtuuriratkaisua ja mikroydinrakennetta (engl. microkernel pattern), joilla on saatu muodostettua tuotelinjan perusarkkitehtuuri. (Meister et al. 2004) Tämän jälkeen vaatimusten pohjalta voidaan ryhtyä muodostamaan korkean tason arkkitehtuurimalleja, jotka jaotellaan kuuluvaksi esimerkiksi datan lastaukseen, käsittelyyn, metadataan ja infrastruktuuriin liittyviksi. Seuraavaksi näiden luokitteluiden malleja jalostetaan yksityiskohtaisemmaksi, jolloin myös tietoturvaan ja fyysiseen infrastruktuuriin liittyviin kysymyksiin tulee etsiä vastauksia. Lopuksi tehdään arkkitehtuurin toteutussuunnitelma ja se dokumentoidaan yhdessä arkkitehtuurin kanssa sekä esitellään muulle projektitiimille, sponsoreille ja johtajille. (Kimball et al. 2011, ss. 183-188) Tietovarastoarkkitehtuurin komponenttien pohjalta arkkitehtuuritiimi määrittelee funktiot, proseduurit ja datasäilöt, jotka ovat olennaisia kullekin arkkitehtuurissa määritellylle komponentille. Suunnitelma toimii pohjapiirroksena tietovaraston suunnittelulle ja toteutukselle. (Ponniah 2010, s. 149)

Teknologiaradan toinen vaihe on tuotteiden valinta ja asennus. Valinnassa arkkitehtuurisuunnitelma toimii listana, jonka mukaan valitaan vaadittua toiminnallisuutta tukevat tuotteet. Tuotekategoriat, joihin valinta kohdistuu, ovat laitteisto, tietokannanhallintajärjestelmä, ETL-työkalu ja BI-työkalut. Tuotteiden valintaa ohjaa pääasiassa liiketoiminta- ja toiminnalliset vaatimukset. (Kimball et al. 2011, s. 191) Tuotteet ovat siis tapoja toteuttaa määriteltyä teknistä arkkitehtuuria. (Ponniah 2010, s. 150) Westermanin (2001, ss. 111-112) ja Silversin (2008, s. 54) mukaan yksi tärkeimmistä päätöksistä teknologioiden suhteen on tietokannanhallintajärjestelmän valinta. Siinä tulisi vaihtoehtoja vertailla kolmen kriteerin mukaan. Niitä ovat yhteensopivuus esimerkiksi lähdejärjestelmien kanssa, ylläpidettävyys ja mahdollisuus lineaariseen kasvuun. (Westerman 2001, ss. 111-113) Inmon (2005) luettelee samansuuntaisesti tietovaraston tietokannanhallintajärjestelmän yleisiksi vaatimuksiksi kyvyn hallita suuria määriä dataa ja erilaisia medioita, kyvyn monipuoliselle indeksoinnille ja datan helpolle monitoroinnille ja tarjota monipuoliset rajapinnat sekä lähdejärjestelmille että raportointisovelluksille. (Inmon 2005)

Tietovarastoa rakentava organisaatio voi myös päättää tehdä ETL-prosessin toteutuksen itse. (Vassilidis et al. 2002, s. 14) Muut toimenpiteet, joita tässä vaiheessa suoritetaan, pitävät sisällään esimerkiksi markkinatutkimuksen tekemisen, tuotteiden tarkemman evaluoinnin, prototypoinnin ja neuvottelut tuotteiden toimittajien kanssa. (Kimball & Ross 2002)

DATARATA

Kimball et al:n (2011, s. 233) elinkaarimallissa suunnittelun ja toteutuksen vaiheiden keskimmäinen prosessi on niin sanottu datarata. Sen vaiheita ovat dimensionaalinen mallinnus, fyysinen mallinnus sekä datan lastauksen suunnittelu ja toteutus. Dimensionaalinen malli muodostetaan vaatimusmäärittelyn tuloksena tunnistettujen liiketoimintavaatimuksien perusteella. Ennen kuin mallia voidaan ryhtyä muodostamaan, pitää lähdedatan ominaisuuksia tutkia. Tutkittavia ominaisuuksia ovat granulariteetti, yhden-

mukaisuus ajan suhteen, arvojen validiteetti ja attribuuttien saatavuus. (Kimball & Ross 2002) Koska mallin luoneet kirjoittajat perustavat kehityksensä väylällä integroitujen datakomeroiden arkkitehtuurien aikaansaamiseen, käyttävät he vaiheelle yläotsikkoa **dimensionaalinen mallinnus**. Inmon et al:n (2008) kannattaman yrityksenlaajuisen tietovaraston tapauksessa mallinnus ei ole dimensionaalista vaan malli muodostetaan kolmannessa normaalimuodossa. Vaikka näkemyseroja lähestymistapojen välillä on, voidaan ylätason elinkaarimalli esittää tietovarastoarkkitehtuurien välille yhtenäiseksi. Joka tapauksessa dataradalla merkittäviä päätöksiä ennen suunnitteluun siirtymistä on juuri tietomallin ja tietovarastoarkkitehtuurin valinta (Silvers 2008, s. 55).

Kimball et al:n (2011) elinkaarimallin dimensionaalisen mallinnuksen vaiheeseen voidaan monen muun kirjoittajan mukaan jakaa käsitteelliseen ja loogiseen mallinnukseen. (ks. esim. Malinowski & Zimányi 2008; Silvers 2008). Eri mallien ominaisuuksia käsiteltiin kappaleessa 3.3.2.

Malinowskin ja Zimányin (2008) mukaan tietovaraston suunnittelu poikkeaa tässä vaiheessa riippuen siitä, minkälaista lähestymistapaa suunnitteluun käytetään. He määrittelevät toisistaan hiukan poikkeavat suunnitteluprosessit analyysivetoiselle, johon kuuluvat käyttäjä- ja tavoitevetoinen, ja datavetoiselle lähestymistavalle. Analyysivetoisessa ensimmäinen askel on luoda hahmotelma käsitteellisestä mallista, liiketoiminta- tai käyttäjävaatimusten pohjalta. Datakomeroiden tapauksessa tämä tarkoittaa selkeästi erotettavia dimensioita, faktoja, mittareita ja hierarkioita. (Malinowski & Zimányi 2008, s. 257) Tässä vaiheessa tehtäviä päätöksiä ovat esimerkiksi mallinnettavan prosessin valinta, datan granulariteetin valinta, dimensioiden tunnistus ja mukautus, tutkittavien suureiden eli faktojen valinta sekä niiden määrittäminen, kuinka pitkältä menneisyydestä dataa halutaan kerätä. (Ponniah 2010, s. 226) Mukautettuja dimensioita luodessa työkaluna voidaan käyttää esimerkiksi väylämatriisia, jossa riveillä luetellaan tarvittavia dimensioita ja sarakkeina tietovaraston vaatimuksia. (Kimball & Ross 2002) Kun malli on muodostettu, tarkistetaan datan saatavuus peilaamalla käsitemallin kaikkia elementtejä dataan lähdejärjestelmissä. Näin saadaan myös dokumentoitua mappaus lähdejärjestelmien ja tietovaraston datan välillä. Määrittelyssä tulee kuvata myös tarvittavat transformaatiot lähdedatassa. Viimeisenä vaiheena on lopullisen tietomallin muodostaminen ja mappausten parantelu. Näiden kaikkien vaiheiden aikana teknistä ja liiketoiminnan metadataa kehitetään jatkuvasti. Metadataan kuuluvat ETL:ään liittyvät tiedot, tiedot lähteiden ja tietovaraston skeemasta, käytetyistä yksiköistä, valuutoista tai lyhenteistä. (Malinowski & Zimányi 2008, ss. 272-273) Metadatan ylläpito on erityisen tärkeää kompleksisuuden poistamiseksi ja ylläpidettävyyden parantamiseksi. (Vetterli et al. 2000, s. 68)

Datavetoisessa lähestymistavassa suunnittelu koostuu niin ikään kolmesta vaiheesta: tietokantamallin muodostamisesta, mallin käyttökelpoisuuden määrittelystä käyttäjilleen sekä mallin viimeistelystä ja mappausten parantelusta. Mallin muodostaminen tässä tapauksessa on suoraviivainen, sillä mallin elementit on tunnistettu edellisessä vaihees-

sa. Koska malli toimii kommunikoinnin välineenä, tulee skeeman elementit nimetä tavalla, joka on kohderyhmälle luonteva. (Malinowski & Zimányi 2008, ss. 281-282) Kimball & Ross (2002) ehdottavatkin standardin nimeämiskäytännön käyttöönottoa kehityksen alkuvaiheessa. Mallin käyttökelpoisuuden hyväksyttämisessä käydään tietovaraston käyttäjien kanssa läpi yksityiskohtaisesti, minkälaisia analyyseja mallin avulla voidaan tehdä. Malli ei välttämättä täytä tässä vaiheessa kaikkia analyysitarpeita, joitain elementtejä voidaan joutua muuttamaan tai malli sisältää mahdollisesti liikaa elementtejä. Viimeisessä vaiheessa otetaan huomioon parannusehdotukset, joiden perusteella mallia parannellaan. Tämän lisäksi myös mappaukset datan lähteen ja mallin välillä määritellään abstraktilla tasolla. (Malinowski & Zimányi 2008, s. ss. 281-282)

Malinowski ja Zimányi (2008) esittelevät myös kolmannen lähestymistavan mallinnukseen, jossa suoritetaan molempien lähestymistapojen mallinnus ja lopuksi mallit yhdistetään. Kirjoittajat varoittavat kuitenkin tehtävän kompleksisuudesta tapauksessa jossa mallit eroavat merkittävästi toisistaan. (Malinowski & Zimányi 2008, ss. 284-285)

Käsitteellinen suunnittelu ja sitä edeltävä vaatimusten määrittely ovat tietovaraston onnistumisen kannalta kriittisiä vaiheita. Ne määrittelevät vastaako toteutettu malli todellista maailmaa eli käyttäjien tietotarpeita. (Malinowski & Zimányi 2008, s. 256) Kun mallinnustehtävät on suoritettu, mallit toimivat kommunikaation välineenä suuremmalle yleisölle, jotta yhteisymmärrys toteutettavasta järjestelmästä saadaan muodostettua. Teknisten ihmisten kanssa peilataan mallia dataan liittyviin ongelmiin ja liiketoimintahenkilöiden kanssa selvitetään, saadaanko mallin avulla tyydytettyä analyyseihin tarkoitetut tietotarpeet. (Kimball & Ross 2002; Malinowski & Zimányi 2008, s. 272)

Seuraavaksi vuorossa on **looginen suunnittelu**, jossa on kaksi tärkeää huomioonotettavaa seikkaa: käsitteellisen mallin muuttaminen loogiseen esitysmuotoon ja ETL-prosessien määrittely edellisen vaiheen mappausten pohjalta. (Malinowski & Zimányi 2008, ss. 289-290) Tässä vaiheessa tulee jo tietää tietokantateknologia, jonka mukaan käsitemalli muutetaan loogiseen muotoon. Yleisesti käytetään relaatiotietokantoja, mutta mahdollisuus esimerkiksi oliotietokantojen käyttöön on myös olemassa. (Halpin 2008)

Kimball et al:n (2011) elinkaarimallissa dataradan seuraava vaihe on **fyysinen suunnittelu**, jossa edellä aikaansaatuja malleja jalostetaan edelleen muotoon, joka on riippuvainen fyysisestä tietokannanhallintajärjestelmästä, jolle tietovarastoratkaisua ollaan toteuttamassa. Fyysisen mallin yksityiskohdat ovat pitkälti riippuvaisia, loogisesta tietomallista, käytettävästä tietokannanhallintajärjestelmästä, datan määrästä, käyttötavasta ja tietokantaa käyttävistä sovelluksista. (Kimball et al. 2011, s. 327) Fyysisessä suunnittelussa päätetään taulujen sarakkeiden nimistä, tietotyypeistä, pää- ja vierasavainten määrittelystä sekä tyhjistä eli niin sanoituista null-arvoista. Toimenpiteisiin kuuluu tietokannan optimointi indeksien ja summataulujen muodostuksesta. (Kimball & Ross 2002) Operaation suoritus riippuu luonnollisesti siitä fyysisestä tietokannanhallintajärjestel-

mästä, joka tietovaraston toteuttamiseen on valittu. Käytetyimpiä esimerkkejä ovat Microsoftin SQL Server - ja IBM:n DB2 -teknologiat. (Halpin 2008)

Dataradan kolmas ja viimeinen vaihe on **datan lastauksen suunnittelu ja toteutus**, mikä tarkoittaa ETL-prosessin toteuttamista. Aivan kuten teknologiaradan arkkitehtuurisuunnittelussa edetään abstraktista yksityiskohtaiseen, samaa periaatetta noudatetaan ETL-suunnitelman kanssa. (Kimball & Ross 2002) ETL:n toteutus voidaan jakaa kahteen vaiheeseen: datan lähteiden rakenteen ja sisällön analysointi sekä lähteiden mappaus tietovaraston tietomalliin. Vaiheen merkittävimpinä tuotoksina syntyy alkuperäisten tietolähteiden tarkasteluiden perusteella tehtyjen mallien tarkempi uudelleenmäärittely käsitteellisen mallin muodossa. (Vassilidis et al. 2002, s. 14)

Monet kirjoittajat toteavat, että ETL-prosessin toteuttaminen on tietovarastointiprojektissa kaikkein haasteellisin ja eniten aikaa kuluttava työvaihe, sillä noin 50-70 % työmäärästä kuluu sen toteuttamiseen. Työvaiheen haastavuus johtuu monesti lähdedatan keskinäisistä eroavaisuuksista. (Ponniah 2010, ss. 283-284) Kimball et al. (2011) suosivatkin kaupallisten työkalujen hankintaa oman ETL-toteutuksen sijaan. Lisäksi he mainitsevat, että ETL-prosessin toteuttaminen luo pohjan koko tietovarastolle ja sen onnistuminen määrittelee tietovarastohankkeen kohtalon. (Kimball et al. 2011, s. 370) Vassilidis et al. (2002) korostavat, että kun toteutus suoritetaan organisaation sisällä, on sillä oltava käytössään riittävät lähtökohdat ja osaaminen suunnittelulle, toteutukselle ja mallinnukselle. Rizzi et al. (2006) kuitenkin toteavat, että ETL:n mallinnus on vähemmän standardoitua verrattuna tietomalleihin. Kirjoittajat näkevät sen kuitenkin tärkeäksi, sillä sen mallinnuksella on olennainen vaikutus tietovaraston suunnittelun luotettavuuteen ja suunnitteluprosessin kestoon. (Rizzi et al. 2006, s. 4) Giorgini et al. (2005) toteavat ETL:ään käytettyyn työmäärään liittyen, että siihen vaikuttaa merkittävästi, miten tietovaraston vaatimusmäärittelyä on lähestytty. Datavetoinen lähestymistapa yksinkertaistaa ETL:n suunnittelua, sillä siinä käyttäjien tietotarpeet ovat toissijainen huolenaihe kehittäjille. (Giorgini et al. 2005, s. 47)

3.5 Tietovarastoinnin haasteet

Tietovaraston kehityksen haasteita ja järjestelmän onnistumiseen vaadittavia menestystekijöitä on tutkittu monien kirjoittajien toimesta. Haasteita tai menestystekijöitä on tunnistettu sekä projektin vaiheittain tai organisaationaalisten ja teknisten tekijöiden perusteella. (ks. Wixom & Watson 2001; Kimball & Ross 2002; Shin 2003; Yeoh & Koronisos 2010) Tarkastelut pohjautuvat usein DeLonen ja McLeanin (1992) tietojärjestelmien onnistumisen malliin, jossa tutkittavia ulottuvuuksia ovat sen sisältämän datan laatu, järjestelmän laatu, järjestelmän käytön, käyttäjien tyytyväisyys sekä yksittäiset ja organisationaaliset vaikutukset. Seuraavaksi tarkastellaan tietovarastojen kehitykseen liittyviä haasteita ja menestystekijöitä

3.5.1 Organisatorisia haasteita

Aivan ensimmäisenä Kimball et al. (2011, ss. 19-21) kehottavat huomioimaan muutamia kompastuskiviä ennen projektin toteuttamisen aloittamista. Ensinnäkin liiketoimintasponsorin, jolla on käsitys tietovaraston mahdollisuuksista ja on sitoutunut tavoitteeseen, puuttuminen vaikeuttaa tietovaraston toteutuksen suunnittelua ja toteutusta. Tietovarastoprojektille tulee saada resursseja ja sen prioriteetti täytyy säilyttää, vaikka hanke ajautuisi vaikeuksiin Toisaalta sponsori, joka on liian aggressiivinen ja vaativa, vaarantaa tietovarastoprojektein onnistumisen liiallisilla vaatimuksillaan. Myös useiden päällekkäisten sponsorien olemassaolo hankaloittaa järjestelmän vaatimusten täyttymistä. (Kimball et al. 2011, ss. 19-21) Wixom ja Watson (2001, s. 36) ovat sponsorien määrän suhteen eri linjalla, sillä heidän tutkimuksensa osoittaa, ettei yksittäisellä sponsorilla ole välttämättä riittävästi vaikutusvaltaa ja ymmärrystä osaamisalueensa ulkopuolelta. Tästä syystä tukijoita tulisi olla useampia. Haasteita aiheuttaa myös se, että tietovaraston toteutus poikkeaa operatiivisten järjestelmien kehityksestä projektin näkökulmasta. Tietovaraston toteutuksella on tapana olla laajuudeltaan tai rajaukseltaan huomattavasti suurempi varsinkin lähdejärjestelmien määrän kasvaessa. Myös organisaation ulkopuolisen osaamisen tarve monesti lisääntyy, sillä hallittavien teknologioiden kirjo on suuri. Koska suunnittelu ja toteutus nähdään monesti rinnakkaisina vaiheina, aiheutuu siitä projektin hallinnalle omat haasteensa. (Ponniah 2010, s. 80)

Ponniah (2010) luettelee edelleen varoitusmerkkejä, joita projektin aikana tulee seurata ja hallita sen onnistumisen takaamiseksi. Ensinnäkin jos vaatimustenmäärittely venyy odotettua pidemmäksi, tulee karsia tarpeettoman informaation keräämistä ja asettaa selkeä tavoitepäivä vaiheen valmistumiselle. Lisäksi jos tietovarastoa rakentavan yrityksen tarvitsee toteuttaa suuri määrä arkkitehtuuria tukevista ohjelmista, tulee pohtia kolmannen osapuolen tarjoamien sovellusten käyttöä. (Ponniah 2010, s. 92)

3.5.2 Teknisiä haasteita

Kimball ja Ross (2002) esittävät dimensionaaliseen mallinnukseen liittyviä yleisiä virheitä, jotka vaarantavat tietovaraston onnistumisen. Näistä tärkeimpänä he näkevät mukautettujen dimensioiden ja faktojen muodostamisessa epäonnistumisen. Kirjoittajien mukaan mukautetut dimensiot toimivat pohjana koko tietovarastolle ja auttavat sen tavoitteiden saavuttamisessa. Toiseksi kirjoittajat ohjeistavat mallintamaan datan dimensionaalisessa muodossa alhaisimmalla granulariteetin tasolla ja muodostamaan erillisiä summatauluja tarpeen vaatiessa. Alhainen granulariteetti takaa sen, että tilapäiset eli ad hoc -kyselyt ja porautuminen ovat mahdollisia. Seuraavaksi kirjoittajat neuvovat välttämään tietovaraston faktataulujen suunnittelua siten, että se pohjautuu tiettyyn raporttiin tai tietotarpeeseen. Lähestymistapa pitäisi sen sijaan olla prosessin mittauksessa. Viidentenä virheenä on tunnistettu operatiivisten tai niin sanottujen älykkäiden avainten käyttöä faktojen ja dimensioiden pää- ja vierasavaimissa. Lisäksi suorituskykyongelmien ratkaisemisessa ei tulisi yksinomaan käyttää laitteiston suorituskyvyn parantamista,

vaan optimointia tulisi tehdä summataulujen, indeksien ja esimerkiksi rinnakkaistamisen muodossa. (Kimball & Ross 2002)

Dimensioiden suunnitteluun liittyviksi virheiksi kirjoittajat mainitsevat hierarkioiden jakaminen useaksi dimensioksi, dimensioiden attribuuttien muutosten seurannan laiminlyönti, dimensioita kuvaavien attribuuttien käyttämättä jättäminen tai niiden sijoittamien faktatauluun. (Kimball & Ross 2002)

Wixom ja Watson (2001) tutkivat tietovaraston toteutuksen onnistumiseen vaikuttavia tekijöitä. He havaitsivat, että tietovarastoissa järjestelmän laatuun vaikuttaa mallinnettavien aihealueiden valinta ja tallennettava data sekä valittu tietovarastoarkkitehtuuri. Tietovarastoa rakennettaessa ei välttämättä omata osaamista tai visiota tulevaisuudesta, mikä voi juuri johtaa ratkaisun joustamattomuuteen ja ongelmiin integroinnissa. (Wixom & Watson 2001, ss. 35-36) Kimball ja Ross (2002) sekä Ponniah (2010) mainitsevat niin ikään datan laadun merkittäväksi huomioonotettavaksi seikaksi. Samalla tavalla kuin oikean datan kerääminen oikealla tarkkuustasolla vaikuttaa, myös puutteet datan laadussa heikentävät tietovaraston onnistumista. Se onkin yksi DeLonen ja McLeanin (1992) mallin yksi onnistumisen ulottuvuuksista. Kirjoittajat eivät kuitenkaan tarkemmin erittele, minkä ulottuvuuksien suhteen datan laadukkuudessa pitää erityisesti kiinnittää huomiota. Datan laatua voidaan arvioida nimittäin useiden ulottuvuuksien suhteen. Pipino et al. (2002, s. 112) listaavat datan laadulle 16 erilaista ulottuvuutta, joista esimerkkejä ovat uskottavuus, saatavuus, yhdenmukainen esitystapa, ajantasaisuus ja ymmärrettävyys.

Arkkitehtuurin valintaan liittyen Kimball et al. (2013, ss. 19-21) mainitsevat, että organisaatiossa olemassa olevat arkkitehtuurin komponentit, kuten eristäytyneet datakomerot tai tietovarastot, joihin sama data on tallennettuna ilman kokonaisvaltaista suunnitelmaa, aiheuttavat haasteita projektin toteuttamisessa. (Kimball et al. 2013, s. 19-21) Ponniah (2010, s. 80) mainitsee arkkitehtuurisuunnittelun haasteellisuuden yhdeksi haasteelliseksi tekijäksi, mikä erottaa tietovaraston kehityksen jossain määrin operatiivisten järjestelmien kehityksestä. Myös metadatan hallinta on merkittävä huomioonotettava aihe, jonka laiminlyöminen lisää monimutkaisuutta toteutuksessa ja ylläpidettävyydessä. (Ponniah 2010, s. 80)

Watsonin (2002, s. 6) mukaan ETL-prosessin liittyvät ongelmat ovat niin ikään ajaneet tietovarastointiprojekteja epäonnistumaan. ETL-prosessiin liittyviä ongelmia hän mainitsee lähdejärjestelmien kanssa ilmenevät ongelmat, kuten muutokset datan arvoissa ja lähdejärjestelmien rakenteessa, epäjohdonmukaisuus eri lähteiden välillä, vanhentuneiden tietokantateknologioiden olemassa olo sekä vaihtelevat käyttöjärjestelmät ja alustat. Lisäksi ongelmia aiheuttaa tiettyä tehtäväänsä suorittamaan valittujen teknologioiden puutteellisuus ja tietovarastointiosaamisen puuttuminen. Näiden haasteiden määrä ja vaativuus riippuvat kuitenkin lähdejärjestelmien määrästä, jonka kasvaessa vaadittava työmääräkin lisääntyy. (Watson 2002, s. 6; Ponniah 2010, s. 283)

Teknisten haasteiden yhteydessä mainittakoon kuitenkin, että Yeoh ja Koronios (2010) havaintojen mukaan organisationaaliset ja prosessiin liittyvät tekijät ovat tärkeämmässä roolissa tietovaraston toteuttamisessa, sillä tekniset seikat ovat monesti helposti ratkaistavissa. Niitä kannattaa pitää mielessä jo suunnitteluvaiheessa, vaikka monet niistä realisoituvatkin vasta käyttöönottovaiheessa.

4. TUTKIMUSMETODOLOGIA

Tässä kappaleessa selvitetään tutkimuksen ympäristö esittelemällä ensin kohdeorganisaatio. Tämän jälkeen siirrytään tutkimuksen toteuttamisen taustalla oleviin oletuksiin käymällä läpi tutkimusotetta, tutkimus- ja tiedonkeruumenetelmiä sekä aineiston analyysia. Lopuksi selostetaan tutkimuksen suorittamisen vaiheita ja niiden aikana syntyneitä havaintoja.

4.1 Kohdeorganisaatio

Tämän tutkimuksena kohteena on tamperelainen logistiikan ja teollisuuden ohjelmistoyritys, joka työllistää noin 70 henkilöä. Yrityksen ydinajatuksena on toimittaa asiakkailleen ohjelmistoratkaisuja, jotka tehostavat niiden toimintaa ja vähentävät tuottamatonta työtä. Leanware Oy:n toteuttamat ohjelmistoratkaisut palvelevat esimerkiksi satamien, tehtaiden ja logistiikkakeskusten tarpeita. Tämä tutkimus on syntynyt ohjelmistoja logistiikkakeskuksille ja varastoille toteuttavan Logistics-yksikön tietotarpeiden seurauksena. Siksi tämä tutkimus on rajattu käsittelemään kyseisen yksikön toimintaa ja prosesseja.

Logistics-yksikön pääasiallinen asiakkaille toteutettava järjestelmä on varastonhallintajärjestelmä, joka pohjautuu ohjelmistotuoterunkoon. Nykyinen tuoterunko kattaa yleisimmät varastonhallintajärjestelmän toiminnallisuudet, joten useissa tapauksissa pelkällä konfiguroinnilla saadaan räätälöityä asiakkaan toimintaympäristöön soveltuva järjestelmä. Tämän hetkinen tuoterunko koostuu useista kehitystä helpottavista artefakteista. Näitä ovat uudelleenkäytettävä järjestelmän käsitemalli, ohjelmistoarkkitehtuuri ja sen eri ohjelmistokomponentit, tietokantadokumentaatio, erilaiset käännösskriptit ja asennustiedostot sekä ohjelmistokomponenttien dokumentaatio.

Tuoterungon kehittämisestä ja ylläpitämisestä vastaa niin kutsuttu Core-tiimi, jolla on myös tietovaraston tuotteistamisessa merkittävä rooli. Core-tiimi koostuu tuoterungon omistajasta, kahdesta arkkitehdistä, UX-suunnittelijasta ja kolmesta sovelluskehittäjästä. Pääarkkitehti yhdessä toisen sovellusarkkitehdin kanssa ovat vastuussa tuoterungon suhteen tehtävistä kehityspäätöksistä ja omaavat parhaan kokonaisnäkemyksen tuoterungosta. Muut sovelluskehittäjät ovat vastuussa pienemmän mittakaavan suunnittelupäätöksistä ja niiden toteutuksista.

Teknistä suunnittelua tukee tuoterungon omistaja, jolla on keskeinen rooli liiketoimintavaatimusten kokonaisuuden ymmärtämisestä ja yleisimmistä sisälogistiikan prosesseista. Keskitettyä hallintaa ja kontrollia tuoterungon pidemmän aikavälin suuntavii-

voista harjoittaa niin sanottu heimoneuvosto, jonka jäseniä ovat tuoterungon omistaja, asiakasprojektien tuotteiden omistajat, avainasiakasvastaavat ja ohjelmistoarkkitehdit.

4.2 Tutkimusote

Tieteellisen tutkimuksen kirjallisuudessa esiintyy useita luokitteluita siitä, miten tutkimuksen tutkimusstrategioita eli tutkimusotteita tai tutkimusmetodeita tulisi luokitella (ks. esim. Hirsjärvi et al. 2002.) Myös nimitykset tutkimusotteiden ja menetelmien välillä vaihtelevat, mikä aiheuttaa entisestään sekaannusta laajaan tutkimuskäsitteistön kenttään. Erään luokittelun muodostavat Hirsjärvi et al. (2001, s. 124), joiden mukaan tieteessä on tunnistettavissa kolme pääasiallista tutkimusotetta: kokeellinen, survey- sekä tapaustutkimus. Tämä on varsin karkean tason luokittelu, joten hienojakoisemman jaottelun tarjoavat Neilimo ja Näsi (1980), jotka ovat luoneet tyypittelyn neljästä tutkimusotteesta: käsiteanalyyttinen, nomoteettinen, päätöksentekometodologinen ja toiminta-analyyttinen tutkimusote. Lisäyksenä tähän on vielä konstruktiivinen tutkimusote. (Kasanen et al. 1991) Näiden taustalla on varsin erilaisia oletuksia tieteen tekemisen yleislinjojen suhteen, jot et al. en niitä on syytä hiukan käsitellä.

Tutkimusotteen valintaan vaikuttavia tekijöitä ovat tutkimuksen tavoite, tutkijan tiedon taso tutkimuksen lähtötilanteessa, saatavilla olevan aineiston tyyppi sekä tutkimuksen tavoitteet tulosten suhteen. Lisäksi tutkimuksen tulokselle asetetut kriteerit, kuten niiden totuus ja hyöty sekä tarvittava evidenssi, vaikuttavat valintaan. (Olkkonen 1994, s. 59;82). Liiketaloustieteen tutkimusotteita ei voida sijoittaa tieteenkäsityksen taustaperiaatteiden mukaisesti luokkiin, sillä edellä esitettyjen tutkimusotteiden väliset rajat eivät ole selkeät. Tutkimusotteiden ominaisuuksia esittelemällä voidaan muodostaa käsitys tutkimusotteiden käyttötilanteita.

Konstruktiivisella tutkimusotteella tarkoitetaan innovatiivisia konstruktioita tuottavaa metodologiaa, jossa pyritään tuottamaan tosielämän ongelmaan ratkaisu eli konstruktio (Lukka 2010) Toisena mainittu tutkimusote on päätöksentekometodologinen tutkimusote. Otteessa pyritään kehittämään malleja yhdistelemällä aikaisemmin muodostettuja logiikoita. Tulokset antavat jonkin suosituksen päätöksenteon tueksi. (Olkkonen 1994, ss. 70 - 71)

Toiminta-analyyttinen tutkimusote pyrkii hermeneuttisen tieteenkäsityksen mukaisesti ymmärtämään tutkimukselle asetettua ongelmaa, kun kysymyksessä ovat vaikeasti strukturoitavissa olevat ongelmat ja nopeasti muuttuvat tilanteet. Myöskään alussa lähdeaineiston saatavuudesta ei voida olla täysin varmoja. Tutkijan ja tutkittavan kohteen välinen suhde on keskeistä ja tulokset muodostuvat tutkijan tulkinnan perusteella. Käytettävä aineisto on usein empiiristä ja määrältään vähäistä. Tämän lisäksi saatetaan käyttää sisäistä, tarkastelua ohjaavaa tai selittävää ”kovaa” faktaa. Tutkimuksen rakennetta on etukäteen vaikea strukturoida, sillä tutkimusotteen menetelmiin liittyy niukasti metodologisia ohjeita. Tulokset toiminta-analyyttisessa tutkimusotteessa ovat hyvin moni-

muotoisia. Niihin kuitenkin liittyy ongelma yleistettävyyden suhteen, sillä tarkasteltavien tapausten joukko on usein suppea. Monesti tulokset ovat uusia teorioita tai hypoteeseja, käsitejärjestelmiä tai jopa normatiivisia ohjeita, joiden verifiointi jää monesti ratkaisematta kyseisessä tutkimusotteessa. Tulokset kuitenkin saatetaan testata käytännössä, mikä osoittaa niiden kelpoisuuden ja niitä voidaan alkaa omaksua laajemmalti. (Olkkonen 1994, ss. 72-74)

Käsiteanalyyttisessa tutkimusotteessa tarkoitus on luoda käsitejärjestelmiä esimerkiksi ilmiöiden kuvaamiseen ja tunnistamiseen, tyypittelyyn tai tiedon järjestämiseen. Käsitteistö ei sellaisenaan ole juuri merkityksellinen, vaan se palvelee yleensä jotakin tarkoitusta. Aineistona käsiteanalyyttisessa tutkimuksessa ovat yleensä empiirinen tieto kohdeilmiöstä, muut käsitemallit tai kohdeilmiötä koskevat teoriat. Aineistoa analysoidaan ja siitä tehdään synteesiä sekä vertailua, jolloin tuloksena saadaan käsitteistö, joka on perusteltu koettelemalla sitä edellä mainituilla tavoilla. (Olkkonen 1994, ss. 65-66)

Edellä mainituista tutkimusotteista olennaisia tämän tutkimuksen empiriaosuuden kannalta on toiminta-analyyttinen, jonka suhde empiriaan on suora ja käytännönläheinen. Lisäksi tapaustutkimusmetodin soveltaminen on tutkimusotteessa tärkeässä roolissa. (Lukka 2001) Tutkimuksen kirjallisuuskatsaus sijoittuu käsiteanalyyttisen tutkimusotteen piiriin.

4.3 Tutkimusmenetelmä

Olkkosen (1994, ss. 64-65) mukaan tutkimusmenetelmillä tarkoitetaan niiden tieteellisten tiedonhankinta- ja tiedonkeruumenettelyiden joukkoa, joilla tutkimusstrategiaa toteutetaan. Menetelmien joukko on varsin moninainen eikä niiden käyttö välttämättä rajoitu tiettyihin tutkimusotteisiin. Tyypillisiä vastapareja on kuitenkin havaittavissa. Tämän tutkimuksen tutkimusmenetelmäksi valikoitui tapaustutkimus (engl. case study), jonka taustateoriaa seuraavaksi käsitellään.

Tapaustutkimus on empiirinen selvitys, jossa tutkitaan ajankohtaista ilmiötä syvällisesti ja sen todellisessa kontekstissa. Tapaustutkimusta sovelletaan varsinkin, kun rajat ilmiön ja kontekstin välillä eivät ole ilmeiset. (Yin 2003, s. 13) Toisen määritelmän mukaan tapaustutkimus on syvällinen todellisen elämän projektin, instituution, säännöstelyn tai järjestelmän kompleksisuuden ja ainutkertaisuuden tutkimista useista eri näkökulmista. Se on riippumatonta tutkimusmenetelmästä ja on todistusaineistovetoista. Päätarkoitus tapaustutkimuksella on tuottaa ymmärrystä tietystä aihepiiristä ja luoda tietämystä käytännön sovellutuksia varten. (Simons 2009, s. 21) Tarkasteltaessa ilmiötä sen todellisessa asiayhteydessä löydökset luovat käsityksiä siitä, miten ilmiö todellisuudessa käyttäytyy kyseisessä tilanteessa. (Farquhar 2012) Tapaustutkimuksen määritelmällä on Yin:in (2003, s. 14) mukaan toinenkin puoli. Se tukeutuu useisiin aineiston lähteisiin, joiden on lopulta konvergoiduttava keskenään. Siinä hyödynnetään aiempia teoreettisia väittämiä,

jotka ohjaavat aineiston hankintaa ja analyysia. Menetelmänä tapaustutkimus muodostaa kaikenkattavan ohjeistuksen tutkimuksen suorittamisesta. (Yin 2003, s. 14)

Tapaustutkimusta käytetään tutkimusmenetelmänä usein, kun kysytään tutkimuskysymyksinä miten ja miksi. Tapaustutkimus voi olla ekpsloratiivinen eli etsinnällinen, selittävä tai deskriptiivinen eli kuvaileva. (Yin 2003, s. 12) Tapaustutkimusta ei voida sijoittaa joko kvalitatiivisen tai kvantitatiivisen tutkimuksen tekemisen piiriin, vaikkakin siinä on paljon samoja epistemologisia perusteluita ja käytännön menetelmiä kuin yleisesti kvalitatiivisissa tutkimusmenetelmissä. (Simons 2009, s. 14) Yin:in (2003, s. 15) mukaan tapaustutkimus ei siis rajoita kvantitatiivisen ja kvalitatiivisen aineiston käyttöä tutkimuksessa, joten aineisto voi koostua eri tyyppisten aineistojen sekoituksesta.

Yin (2003, ss. 39-40) jakaa tapaustutkimuksen neljään perustyyppiin sen mukaan, käytetäänkö tutkimuksessa yhtä vai useampia tapauksia ja onko saman tapauksen yhteydessä useampia analyysin kohteita, jolloin puhutaan holistisesta (engl. holistic) ja sulautetusta (engl. embedded) lähestymistavasta. Holistisen ja sulautetun tapaustutkimuksen välisen eron määrittelee analyysin kohteiden (engl. unit of analysis) lukumäärä. Kun saman tapauksen puitteissa analyysin kohteita on useita, on kysymys sulautetusta tapauksesta. (Yin 2003, ss. 39-40). Tässä tutkimuksessa analyysin kohteena on yksittäinen tapaus, jossa analyysin kohteita on useampi. Tämä on Yin:in (2003, s. 40) mukaan perusteltua esimerkiksi, kun tapaus edustaa kriittistä tapausta testattaessa taustalla olevaa teoriaa. Toisaalta tapaus saattaa olla jossain määrin uniikki tai äärimmäinen, jolloin mahdollisuutta useampien tapausten tarkasteluun ei ole. Tapaus voi lisäksi olla edustavuudeltaan tai tyypillisyydeltään sopiva. Yhden tapauksen valinnassa onkin mahdollista, että jälkikäteen havaitaan tapauksen olevan jotain muuta kuin sen alun perin tulkittiin olevan. (Yin 2003, ss. 41-42)

Kuten mainittiin, tapaustutkimuksessa voidaan hyödyntää hyvinkin erilaisia aineistoja. Yin (2003, s. 85) mainitsee yleisimmin käytetyiksi lähdeaineistoiksi erilaiset dokumentit, arkistoidut asiakirjat, haastattelut, suoran havainnoinnin, osallistuvan havainnoinnin ja fyysiset artefaktit. Näiden aineostolähteiden hyöty saadaan maksimoitua noudattamalla kolmea tiedonkeruun periaatetta, joita havainnollistetaan seuraavaksi. Ensinnäkin tapaustutkimuksessa tulee käyttää useita erityyppisiä lähteitä, mikä on tapaustutkimuksen yksi vahvuus verrattuna esimerkiksi survey-tutkimuksiin tai historiikkeihin. (Yin 2003, s. 97) Tällä periaatteella viitataan triangulaatioon, joka tarkoittaa yksinkertaistettuna moninäkökulmaisuutta eli eri menetelmien ja lähestymistapojen yhdistelyä. (Tuomi & Sarjajärvi 2002, ss. 141-142) Toiseksi Yin (2003) mainitsee hyviksi tiedonkeruun periaatteiksi myös tutkimustietokannan ylläpitämisen ja kolmanneksi aineistojen välisten yhteyksien havainnollistamisen. Näiden avulla tutkija organisoi keräämäänsä materiaalia ja ulkopuolinen tarkastelija kykenee seuraamaan tutkijan ajatusketjua, mikä kasvattaa tutkimuksen luotettavuutta. (Yin 2003, s. 101;105) Tässä tutkimuksessa käytetään erityyppisiä aineistoja, kuten dokumentteja ja asiakirjoja sekä haastatteluita.

Tapaustutkimus on saanut kritiikkiä tutkimusmenetelmänä, joka ei täytä tieteellisen tutkimuksen kriteereitä. Tutkimuksen laatua voidaan kuitenkin mitata erilaisilla testeillä, jotka mittaavat konstruktiivista validiteettia, sisäistä validiteettia, ulkoista validiteettia sekä luotettavuutta. Konstruktiivinen validiteetti on yleisimmin tapaustutkimuksen suorittamisessa kritisoitu aspekti, sillä monet tutkijat epäonnistuvat aineiston keräämisessä liiallisen subjektiivisuuden takia tai sen vuoksi, etteivät he kykene luomaan riittävän käyttökelpoisia perusteita ilmiöiden tulkinnalle. Sisäinen validiteetti puolestaan viittaa tapahtumien välisen kausaliteetin todistamiseen liittyviin uhkiin. Ulkoisella validiteetilla tarkoitetaan tutkimuksen yleistettävyyttä muihin läheisiin tapausiin. Luotettavuus tarkoittaa, että tutkimus voidaan toistaa saaden samoja tuloksia. Monien näiden testien tulosten parantamiseen mainitaan esimerkiksi edellisessä kappaleessa luetellut hyvän tiedonkeruun periaatteet. (Yin 2003, ss. 35 - 39)

Teorian kehittämistä tapaustutkimuksissa toteutetaan niin sanotun analyyttisen yleistämisen (engl. analytical generalization) avulla. Tällä tarkoitetaan aikaisemmin kehitetyn teorian käyttämistä mallina tai pohjana empiirisesti kerätyn aineiston vertailulle ja tulkinnalle. Tälle vastakkainen yleistämisen muoto on nimeltään tilastollinen yleistäminen. Se nojaa puolestaan tietyn otoksen piiristä kerätyn empiirisen datan tulkintaan ja päätelmiin. Näiden kahden yleistämisen välinen ero on, että tapauksia tilastollisessa yleistämisessä ei tulisi ajatella otoksina, kuten tilastollisissa tutkimusmenetelmissä. (Yin 2003, ss. 32-33)

Tapaustutkimus voidaan nähdä joko tutkimusmenetelmänä tai tutkimusotteena. Yin (2003) luokittelee sen tutkimusotteeksi, joka antaa raamit käytettävien tutkimusmenetelmien valintaan. Tässä tutkimuksessa tapaustutkimus sijoittuu enemmän tutkimusotteen puolelle juuri edellä mainitun seikan johdosta.

4.4 Tiedonkeruumenetelmät ja aineiston analyysi

Tutkimusote ja tutkimusmenetelmä ohjaavat tiedonkeruumenetelmiä, jotka ohjaavat analyysin tekemistä. Seuraavaksi esitellään tutkimuksessa empiirisen aineiston keräämiseen käytettävää tiedonkeruumenetelmää.

4.4.1 Tutkimushaastattelu

Tutkimushaastatteluiden kirjo on laaja ja jossain määrin sekava. Erot eri tutkimushaastattelutyyppien välillä on kuitenkin määritettävissä pitkälti siitä, kuinka strukturoituja ne ovat. Nimityksiä samaan lopputulokseen pyrkivistä haastatteluista on useita, mutta erään nimeämiskäytännön mukaan niitä ovat strukturoitu eli lomakehaastattelu, teemahaastattelu ja strukturoimaton eli avoin haastattelu. Nämä ovat haastattelumuotojen perustyypit, joista käytetään eri tieteenaloilla ja tutkimusotteissa muunnelmia. Mainituista haastattelutyypeistä lomakehaastattelu on strukturoiduin ja nimensä mukaan strukturoi-

maton haastattelu sijoittuu tällä asteikolla toiseen ääripäähän. (Hirsjärvi & Hurme 2011, ss. 43-44; Hirsjärvi et al. 2007, ss. 203-204)

Strukturoitu eli lomakehaastattelu pohjautuu nimensä mukaisesti kysymyslomakkeeseen. Kysymysten keskinäinen järjestys ja muoto on ennalta määrätty. (Hirsjärvi et al. 2007, s. 203) Haastatteluprosessin kannalta vaikein osuus on usein kysymysten muotoilu ja hahmottaminen, minkä jälkeen itse haastattelu on varsin nopea ja helppo toteuttaa. (Hirsjärvi & Hurme 2011, s. 46) Avoimessa haastattelussa ei ole puolestaan selkeää runkoa tai kysymyksiä, vaan haastattelija pyrkii selvittämään haastateltavan ajatuksia, mielipiteitä tai tunteita sen mukaan, miten ne aidosti esiintyvät haastattelussa. (Hirsjärvi et al. 2007, s. 204) Haastateltavien valinta ei kuitenkaan ole yhtä vapaata, sillä tietojen antamiseen valitaan usein asiaan erikoistuneita henkilöitä, joiden lukumäärä on rajallinen. (Hirsjärvi & Hurme 2011, s. 46)

Teemahaastattelu mielletään edellä mainittujen lomakehaastattelun ja avoimen haastattelun välimuodoksi. Teemahaastattelusta voidaan käyttää nimitystä puolistrukturoitu tai puolistandardoitu haastattelu. (Hirsjärvi & Hurme 2011, s. 47) Teemahaastattelussa käytettävät teemat ovat tiedossa, mutta käytettävien kysymysten tarkka muoto ja esittelyjärjestys puuttuvat. (Hirsjärvi et al. 2007, s. 203) Teemahaastattelua voidaan soveltaa sekä yksittäisille henkilöille että kokonaisille ryhmille. Sen tavoitteena on testata, arvioida tai tuottaa hypoteeseja tietystä ilmiöstä, jossa haastateltavat ovat olleet osallisina. (Merton & Kendall 1946) Näin ollen se on sopiva haastattelumuoto tämän tutkimuksen eräänä tiedonkeruumuotona. Lisäksi tässä tutkimuksessa sovelletaan ryhmähaastattelua, jossa samanaikaisesti haastateltavia on kolme.

Teemahaastattelun kysymykset kohdennetaan nimensä mukaisesti ennalta määritettyihin teemoihin, joista haastattelussa keskustellaan. Oletuksena teemahaastattelun suorittamisessa on, että kaikkia yksilön kokemuksia, ajatuksia, uskomuksia ja tunteita voidaan tutkia kyseisellä menetelmällä. (Hirsjärvi & Hurme 2011, s. 48) Teemahaastattelu vastaa hyvin kvalitatiivisen tutkimuksen lähtökohtia, mutta sitä ei voida sitoa yksinomaan kvantitatiivisen tai kvalitatiivisen tutkimuksen piiriin, sillä kerätystä aineistosta voidaan laskea esimerkiksi frekvenssejä tai aineistoa voidaan muuntaa tilastollisen analyysin edellyttämään muotoon. (Hirsjärvi et al. 2007, s. 203) Teemahaastattelussa ei tarvitse ottaa kantaa siihen, kuinka tarkka tai syvällinen aiheen käsittely haastattelussa on. (Hirsjärvi & Hurme 2011, s. 48)

4.4.2 Aineiston analyysi

Haastatteluiden suorittamista seuraa aineiston analyysi. Koska tapaustutkimusta mielletään tämän opinnäytetyön puitteissa enemmän tutkimusotteena eikä niinkään menetelmänä, ei sille pystytä antamaan tarkkoja ohjeita, joita seurata empiirisen aineiston analyysissä. Yin (2003) ehdottaa kuitenkin, että myös tapaustutkimuksella tulisi olla yleinen analyyttinen strategia, joka priorisoi analysoitavat asiat ja määrittelee miksi niitä

analysoidaan. Kirjoittaja jakaa analyysistrategiat kolmeen kategoriaan, joista ensimmäinen on teoreettisiin ehdotelmiin nojautuminen. Ehdotelmien avulla helpotetaan määrittelyä siitä, mihin dataan keskitytään ja mikä jätetään huomiotta sekä auttaa organisoimaan koko tapaustutkimusta ja määrittämään tutkittavat vaihtoehtoiset selitykset. Toinen analyysistrategia onkin vaihtoehtoisten selitysten tutkiminen. Strategia on kytköksissä edellisen kanssa, mutta toimii myös, kun teoreettisia ehdotelmia ei ole tapaustutkimukselle olemassa. Datankeruu keskittyy näin ollen keräämään todisteita mahdollisista vaihtoehtoisista tekijöistä, joilla on vaikutusta tapaustutkimukseen. Kolmantena strategiana mainitaan selityksen muodostaminen, joka tarkoittaa deskriptiivisen viitekehyksen luomista tapauksesta. Sen avulla tapaustutkimuksen kulkua voidaan organisoida sekä myös paljastaa kausaalisuhteita, joita ryhdytään analysoimaan. (Yin 2003, ss. 109-114)

Tuomi ja Sarajärvi (2009) jaottelevat puolestaan kvalitatiivisessa tutkimuksessa käytettävät analyysimenetelmät kolmeen pääluokkaan: aineistolähtöinen, teoriaohjaava ja teorialähtöinen analyysi. Aineistolähtöisessä analyysissä analyysin suorittaminen etenee täysin kerätyn aineiston ehdoilla ja aineistosta pyritään luomaan teoreettinen kokonaisuus. Teoriaohjaavassa analyysissa aineistosta etsitään teoriasidonnaisia kytkentöjä tai teoria toimii apuna analyysin etenemisessä, jolloin päättely on abduktiivista. Viimeisenä mainittu teorialähtöinen analyysi pohjautuu aikaisempaan teoriaan, malliin tai auktoriteettiin, jolloin tutkittava ilmiö määritellään jo aikaisemmin tunnetun tiedon perusteella. (Tuomi & Sarajärvi 2009, ss. 95-97) Tässä tutkimuksessa ensimmäiseksi on luotu teoreettiset lähtökohdat suorittamalla tutkittavasta aiheesta kirjallisuuskatsaus, jonka perusteella on muodostettu käsitemalli ja eräänlainen tarkistuslista tutkimusongelman ratkaisemiseen teorian näkökulmasta. Tämä tarkistuslista toimii empiirisen osion pohjana, joten suoritettavan analyysin voidaan sanoa Sarajärven ja Tuomen (2009) jaottelun mukaan olevan teoriaohjaava tai Yin:in (2003) jaottelun mukaan teoreettisiin ehdotelmiin nojautuminen.

Tieteellisessä yhteisössä on kritisoitu, että tapaustutkimuksen tuottamien teorioiden validiteetin suhteen on puutteita (ks. esimerkiksi Flyvbjerg 2006; Yin 2003). Yin (2003) ehdottaakin tämän validiteettiongelman hoitamiseksi tarkemman analyysitekniikan soveltamista. Vaikka haastattelun analyysitavan yksi merkittävin määrittelijä on tutkimustyyppi, on siitä huolimatta tekniikoista tunnistettavissa rajallinen määrä konkreettisia tapoja. Näitä ovat esimerkiksi laskeminen, asteikointi, teemoittelu, alaryhmien analysointi ja metaforien käyttö. (Hirsjärvi & Hurme 2011, s. 153) Yinin (2003) nimeämiä tekniikoita puolestaan ovat kaavojen yhdistely, selityksen rakentaminen, aikasarja-analyysi, loogiset mallit sekä tapausten väliset synteesit. Kirjoittaja mainitsee, että analyysistrategian valinta ohjaa tekniikan toteuttamista, mutta ei sinänsä tarjoa tarkasti määriteltyjä askeleita analyysin toteuttamiselle. (Yin 2003) Edellä mainituista tekniikoista olennaisimpia tämän tapaustutkimuksen kannalta ovat kaavojen yhdistely, selityksen rakentaminen ja loogiset mallit. Aikasarja-analyysi ei ole datan määrän kannalta

validi tekniikka. Tapausten välinen syntetisointi ei ole mahdollisesti yksinkertaisesti sen takia, että käsiteltävänä on vain yksi tapaus.

Analyysivaihe alkaa monesti jo aineiston keruun yhteydessä, mikä on luonteenomaista kvalitatiiviselle tutkimusmenetelmälle. Haastattelija tekee havaintoja haastattelun kuluessa ilmiöistä niiden tapahtumatiheyden tai toistuvuuden kannalta. Laadullisen tutkimuksen analyysissa alkuperäinen aineisto säilytetään analyysissä usein juuri siinä sanallisessa muodossa, jossa se on kerätty. Lisäksi tutkija käyttää joko induktiivista tai abduktiivista päättelyä. (Hirsjärvi & Hurme 2011, s. 136) Induktiivinen päättely tarkoittaa, että päättely lähtee liikkeelle aineistosta, joka käsitteellistetään ja käsitteiden perusteella muodostetaan teoria. Abduktiivisessa päättelyssä teoria muodostetaan havaintoihin liittyvän johtoajatuksen, esimerkiksi hypoteesin, pohjalta. (Anttila 1998) Kvalitatiivisen aineiston analyysissa on myös tavallista, että siihen käytettävät tekniikat ovat monipuolisia ja standardoimattomia. (Hirsjärvi & Hurme 2011, s. 136).

4.5 Tutkimuksen suorittaminen

Tutkimuksen voidaan katsoa jakautuvan kahteen selkeästi erilliseen, mutta ajallisesti osittain limittäiseen vaiheeseen. Ensimmäisessä vaiheessa kirjallisuuden perusteella luotiin tutkittavaan aiheeseen kirjallisuuskatsaus ja käsitemalli, joka toimi empiirisen osan suorittamisen pohjana. Näin ollen empiirisen osan voidaan sanoa olevan teoriaohjaava. Teorian koostaminen oli pitkä ja haastava prosessi, sillä ajallisesti se vei koko tutkimusprosessista yksinään reilusti yli puolet työmäärästä. Teoriakehys saatiin kuitenkin luotua ennen alkuperäisen suunnitelman mukaista päivää, jolloin voitiin aloittaa tutkimuksen empiirinen osuus. Koska käsiteltävä teoria tuli kerätä hyvin laajalta alueelta tutkimuskysymyksiin vastaamiseksi, oli ajoittain haastavaa pysyä rajauksen sisällä ja pitää asiasisällöt määriteltyjen rajausten ja diplomityön laajuuden sisällä. Monia aihepiirejä olisi voinut raportoida laajemmin, mutta vaatimus tiiviistä esityksestä esti tämän. Kirjallisuuden perusteella muodostettiin tarkistuslista, jossa esitettiin kirjallisuudesta löytyneitä ratkaisuja tutkimusongelmiin. Analyysi suoritettiin heijastamalla teoriaa tutkimuskysymyksiä vasten.

Toinen vaihe koostui empiirisen aineiston keräämisestä eli haastatteluiden suorittamisesta. Haastattelumuotona käytettiin teemahaastattelua eli puolistrukturoitua haastattelua, jossa käsiteltiin kolmea eri teemaa. Teemahaastattelussa haastateltavilta kyseltiin tuoterunkoon perustuvan ohjelmiston tai tietovaraston kehittämisprosessista, ohjelmiston tai tietovaraston laajennettavuudesta, räätälöitävyydestä ja uudelleenkäytettävyydestä. Haastattelun lopuksi haastateltaville esiteltiin teorian pohjalta koostettu tarkistuslista, jossa esiteltiin löydettyjä tuloksia tutkimusongelmiin. Haastattelumuoto oli ryhmähaastattelu, jossa ryhmän koko oli kolme henkilöä. Haastateltavat valittiin kohdeorganisaatiosta tutkimuksen suorittajan käsityksen mukaan heidän kokemuksesta ja osaamisesta tutkittavan ongelman suhteen. Haastattelujoukko muodostui näin ollen eri organisaatioyksiköissä työskentelevistä henkilöistä, sillä esimerkiksi merkittävää tietovarasto-

osaamista ei ollut saatavissa tuoterungon parissa työskennelleiden ihmisten keskuudesta ja päinvastoin. Haastatteluryhmiä havainnollistaa alla oleva taulukko 1.

Haastatteluryhmien osaamisalueet voidaan eritellä siten, että ensimmäisessä ryhmässä oli alustankehityksessä suuressa roolissa olevia henkilöitä, kuten pääarkkitehti, arkkitehti ja vanhempi ohjelmistokehittäjä. Toisessa ryhmässä yhdistyi sekä tietovarasto- että alustankehitysosaaminen ja kolmannessa ryhmässä oli kokonaisvaltaista osaamista ja näkemystä tietovaraston kehitykseen. Haastattelut suoritettiin vajaan kahden viikon sisällä muutaman päivän välillä toisistaan, mikä mahdollisti jo pienimuotoisen analyysin ja tarkentavien kysymysten miettimisen istuntojen välillä. Ennen haastattelun aloittamista tutkimuksen tavoite pyrittiin havainnollistamaan haastateltaville ja keskusteltiin lyhyesti aihepiirien keskeisimmistä käsitteistä.

Taulukko 1. *Lista haastatteluryhmistä, haastateltavista ja heidän taustasta*

Ryhmä-numero	**Osallistujat**	**Taustatiedot**
1	Jani, pääohjelmistoarkkitehti	Omaa kokonaisnäkemyksen tuoterungon suunnittelusta, sen tarjoamista mahdollisuuksista ja rajoitteista. On vastuussa teknisestä kokonaisuudesta.
	Miika, ohjelmistoarkkitehti	On yhdessä pääarkkitehdin kanssa vastuussa teknisistä suunnitteluratkaisuista.
	Antti-Ville, vanhempi ohjelmistokehittäjä	On ollut alusta asti kehittämässä varastonhallintajärjestelmän tuoterunkoa jäsenenä Core-tiimissä.
2	Matti, tuoterungon omistaja	Hallitsee liiketoiminnallisen näkökulman tuoterunkoon ja tuntee eri asiakkaiden vaatimukset varastonhallintajärjestelmälle
	Tomi, sovelluskehittäjä	On ollut alusta asti kehittämässä varastonhallintajärjestelmän tuoterunkoa jäsenenä Core-tiimissä.
	Turo, sovelluskehittäjä	On ollut alusta asti kehittämässä varastonhallintajärjestelmän tuoterunkoa jäsenenä Core-tiimissä.

3	Sami, liiketoimintatiedon hallinnan tekninen vastaava	Tuntee sekä kokonaisuuden että teknisiä yksityiskohtia toteutetuista tietovarastointiratkaisuista.
	Simo, tietovarasto- ja BI-asiantuntija	Tuntemus tietovarastojen ja raportointiratkaisujen toteutuksista korkeakouluille ja asiakkuuksienhallintajärjestelmiin.
	Henri, tuotteen omistaja	Useamman tuotteen omistajana omaa laaja-alaisen näkemyksen asiakasvaatimuksista, niiden teknisistä totutuksista sekä tietovarastojen toteuttamisesta varastonhallintaympäristöön.

Teemahaastattelu koostui kolmesta teemasta, joista kaikista keskusteltiin jokaisen ryhmän kanssa, tosin hieman eri näkökulmista haastateltavien ydinosaamisen mukaan. Teemahaastattelun runko on esitetty liitteessä 2. Ennen toisen teeman käsittelyä käytiin myös läpi termien laajennettavuus, räätälöitävyys ja uudelleenkäytettävyys tarkemmat merkitykset. Laajennettavuudella tarkoitettiin mahdollisuutta laajentaa ratkaisua jälkikäteen. Räätälöitävyydellä tarkoitettiin mahdollisuutta tuoterunkoon tehtäville muutoksille ilman, että siitä aiheutuu ongelmaa tuoterunkoa käyttävissä asiakasinstallaatioissa. Uudelleenkäytettävyydellä tarkoitettiin mahdollisuutta kopioida ratkaisu asiakkaalta toiselle tehokkaasti ja vaivattomasti. Kolmannessa teemassa haastateltavia pyydettiin ottamaan kantaa tarkistuslistan sisältöön lisäämällä tai poistamalla kohtia siihen.

Haastatteluiden aikana tehtiin muistiinpanoja haastateltavien tärkeimmistä huomioista teemoihin liittyen ja käydyt keskustelut nauhoitettiin myöhempää tarkastelua varten. Kukin haastatteluaineisto käytiin läpi samalla muotoillen ja tarkentaen muistiinpanoja.

Seuraavaksi oli vuorossa analyysivaihe, joka aloitettiin haastatteluaineiston analysoinnilla. Jo haastatteluiden aikana syntyi haastattelijoiden ajatusten ja teorian välille kytköksiä, jotka vahvistuivat varsinaista analyysityötä suoritettaessa. Aineiston analyysin voidaan sanoa olleen teoriaohjaavaa, sillä haastatteluaineistosta etsittiin kytkentöjä ja tukea teoriaosassa muodostettuun viitekehykseen. Viitekehystä on havainnollistettu tarkistuslistan muodossa. Tarkistuslista on kuvattu tarkemmin tämän tutkimuksen liitteessä 1. Aiheiden merkittävyyttä tutkimuksen kannalta arvioitiin osittain niiden esiintymistiheyden perusteella ja haastateltavien näkemyksen mukaan niiden olennaisuudesta tutkimuskysymyksen kannalta. Sinänsä ei voida sanoa, että olisi käytetty mitään tiettyä analyysimenetelmää, mutta ylätasolla prosessia voidaan luonnehtia sisällönanalyysiksi.

5. TULOKSET

Tässä kappaleessa esitellään tutkimuksen empiirisen osan perusteella syntyneet tulokset pääpiirteissään. Käsittelyjärjestys seuraa teemahaastattelurunkoa, joka on esitetty liitteessä 2. Ensin käydään läpi tuoterungon kehitysprosessin vaiheisiin liittyvät tulokset. Sen jälkeen siirrytään vastaavan prosessin esittelyyn tietovarastoinnin tapauksessa. Kolmantena keskustellaan ohjelmiston tai tietovaraston laajennettavuuteen, räätälöitävyyteen ja uudelleenkäytettävyyteen liittyvät tulokset ja lopuksi haastateltavien mielipiteitä teorian ja käytännön vastaavuuksista edellä mainituissa aihepiireissä.

5.1 Tuoterunkoon perustuvan ohjelmiston kehittäminen ja prosessissa huomioitavat asiat

Haastateltavat kuvailivat tuotteistusprosessin koostuvan vaiheista, joita ovat esitutkimusvaihe, järjestelmän suunnittelu määritys, toteutus ja testaus. Vaikka vaiheet tunnistettiin erilliseksi, monien katsottiin kulkevan limittäin toisiinsa nähden.

5.1.1 Esitutkimusvaihe

Ensimmäisenä vaiheena tuotteistamisessa tunnistettiin siis esitutkimusvaihe. Ensimmäisen haastatteluryhmän osalliset käyttivät myös nimityksiä tarvekartoitus tai vaatimusten määrittely. He näkivät tämän vaiheen keskeisimpänä toimintana asiakkaan vaatimusten tunnistamisen ja käsitekartan muodostaminen keskeisimpien käsitteiden ympärille asiakkaan vaatimusten perusteella. Asiakasvaatimusta kerätään suoraan asiakkaalta, olemassa olevista järjestelmistä ja liiketoimintatuntemuksen ja muun kokemuksen kautta. Olennaista vaatimusten keräämisen kannalta on se, että niissä huomioidaan useampia asiakkaita, sillä ratkaisun pitää palvella mahdollisimman laajaa asiakaskuntaa. Ryhmän arkkitehdit huomauttivat, ettei välttämättä riitä, että vaatimukset tukevat nykyhetken tarpeita, vaan järjestelmän on kyettävä vastaamaan asiakkaan vaatimuksia myös tulevaisuudessa.

Toisen haastatteluryhmässä osallisena ollut tuoterungon omistaja täydensi edellistä toteamalla, että esitutkimusvaiheessa on tehtävä toteutettavasta järjestelmästä itsenäinen määrittely, joka perustuu tietynlaiseen synteesiin tunnistetuista vaatimuksista sitä kohtaan. Asiakastarpeet on otettava huomioon, mutta yhden asiakkaan ei voi antaa liikaa viedä suunnitelmaa tiettyyn suuntaan. Tämä tarkoittaa kompromissien tekoa ja priorisointia vaatimusten suhteen. Tuoterungon omistaja lausui: ”*Ei pidä määritellä yhden asiakkaan kanssa liikaa, vaan muodostettu ratkaisu tarjotaan asiakkaalta toiselle sellaisenaan”.*

Esitutkimusvaihetta suoritettaessa on elintärkeää huomioida ennen kaikkea siihen vaadittavat resurssit. Vaiheen suorittaminen vaatii jatkuvaa kompromissien tekoa käytettävien panostusten ja mahdollisten hyötyjen suhteen. Toisaalta on varottava liiallista analysointia sekä liian laajan ja monimutkaisen järjestelmän rakentamista, mutta toisaalta liian suppea ratkaisu kostautuu tehokkuuden näkökulmasta muun muassa tuoteinstansseja myöhemmin luotaessa. Mikäli asiakkaiden tarpeita ei ole hahmotettu riittävällä tasolla, ilmaantuu muutostarpeita jälkikäteen, jolloin korjaukset tuoterunkoon ovat monesti teknisesti haastavia ja kuluttavat resursseja.

5.1.2 Suunnittelu ja määrittely

Seuraavaksi vaiheeksi haastatteluissa tunnistettiin järjestelmän suunnittelu ja määrittely. Vaihe koostuu pääasiassa arkkitehtuurikuvauksen luomisesta ja sen tarkemmasta teknisestä määrittelystä. Vaiheen sisällön katsottiin etenevän iteratiivisesti siten, että ensin toteutettiin tietty valittuun suunnittelufilosofiaan perustuva perusarkkitehtuuri, jota kehitettään eteenpäin pienin pyrähdyksin. ”*Suunnittelussa kokonaisuuden hahmottaminen ja näkökulman laajentaminen on vaikeaa, mutta onnistumisen kannalta se on välttämätöntä*”, totesivat ensimmäisen haastatteluryhmän arkkitehdit. Sinänsä sekä ensimmäisessä että toisessa haastatteluryhmässä suunnittelun mainittiin olevan lähes tavallisen ohjelmistosuunnittelun kaltainen.

Suunnitteluvaiheessa muiksi olennaisiksi seikoiksi tunnistettiin esimerkiksi se, että tehtävillä arkkitehtuurin suunnitteluratkaisuilla ja muodostetulla tietomallilla on kauaskantoiset ja laaja-alaiset vaikutukset, joten niihin tulee kiinnittää erityistä huomiota. Tulevaisuudessa tarvittavien muutosten vaikutuksia tulee pystyä rajaamaan mahdollisimman pienelle alueelle tietynlaisilla arkkitehtuuri- ja suunnitteluratkaisuilla.

5.1.3 Toteutus

Tuotteistuksen seuraavaksi vaiheeksi nimettiin toteutus, jossa arkkitehtuurisuunnitelman ja suunnittelufilosofian mukaan muodostettiin konkreettinen ohjelmistoarkkitehtuuri. Toteutus prosessina on suoraviivainen arkkitehtuurin konkretisointi.

Tärkeimpiä huomioon otettavia asioita toteutuksessa mainittiin esimerkiksi koodin korkeammat laatuvaatimukset. Toinen ensimmäisen haastatteluryhmän arkkitehdeista tuumasi: ”*Jälkikäteen tehtävä refaktorointia on pelottavaa, joten sitä pitää pysytä välttämään viimeiseen asti. Ikinä ei voida olla täysin varmoja, että refaktoroinnin jälkeinen logiikka toimii täysin samalla tavalla osana suurempaa kokonaisuutta*”. Ryhmässä osallisena ollut vanhempi ohjelmistosuunnittelija lisäsi tähän: ”*Korjaukset ja muutokset poikivat helposti suuren lisätyön, koska muutokset vaikuttavat laajemmalle alueelle ja kokonaisuus tulee ottaa aina huomioon*”.

Ensimmäisessä haastatteluryhmässä pääarkkitehti totesi tuoterunkoon perustuvan ohjelmistokehityksen tapauksessa toteutuksessa korostuvan kaikenlainen systemaattinen

toiminta ja automatisoitujen prosessien luominen, sillä monet operaatiot ovat ympäristöstä riippumatta samankaltaisia ja vaativat samanlaisia toimenpiteitä. Myös muutosten hallinnan mahdollisuus eli toteutettujen muutosten hallittu vieminen eri asiakasinstallaatioihin nähtiin tärkeänä piirteenä tuotteistamisessa. Edellä mainittujen asioiden mahdollistamisen todettiin kuuluvan tuotteistamisesta vastaavien henkilöiden vastuualueelle.

5.1.4 Testaus

Viimeisenä varsinaisena vaiheena mainittiin testaus. Kaikissa haastatteluryhmissä oltiin sitä mieltä, että testausta suoritetaan myös toteutuksen yhteydessä, mutta se kuitenkin nähdään omana vaiheenaan. Testauksessa tuoterunkoa pyritään testaamaan tavallisen ohjelmistotestauksen tapaan, mikä kohdeyrityksessä tapahtuu voimakkaan tietokantasidonnaisuuden takia lähinnä yksikkötestauksena. Silti testauksen merkitys tuoterunkoon perustuvassa kehityksessä on suuri, sillä toimivuus tulee varmistaa kaikissa tuoterunkoa hyödyntävissä ympäristöissä. Tuoteinstanssien määrän kasvaessa merkitys korostuu entisestään. Ensimmäisen haastatteluryhmän vanhempi ohjelmistosuunnittelija totesi: ”*Testauksessa pitää kiinnittää erityistä huomiota ympäristöihin, jotka tiedetään merkittävästi muista poikkeaviksi ja joissa on syytä epäillä toteutuksen puutteellisuutta”*.

Testauksessa tulee kiinnittää huomiota siihen, että toteutuksia, joita ei yksinkertaisesti voida testata ilman asiakasympäristöä, toteuttajan tulee kommunikoida mahdollisista huomioista toiminnallisuutta käyttöönotettaessa tuoteinstanssissa. Katselmointikäytännöt ovat toimineet osittain paikkaamassa tätä puutosta.

5.1.5 Keskitetty hallinta

Oman loogisena kokonaisuutena haastatteluissa kävi ilmi keskitetty hallinta, joka nähtiin jatkuvana toimintana tuoterungon kaikissa elinkaaren. ”*Keskitetty hallinta tarkoittaa tuoterungon kehityksen vastuun jakamista useammalle kuin yhdelle henkilölle, koska kukaan ei voi olla yksin vastuussa näin suuresta kokonaisuudesta*”, tuumattiin ensimmäisessä haastatteluryhmässä. Kohdeorganisaatiossa tätä ryhmää kutsutaan nimellä ”Heimoneuvosto” ja sen jäseniä ovat tuotteen arkkitehdit, tuoteinstansseista vastuussa olevat tuotteiden omistajat, tuoterungon omistaja ja avainasiakasvastaavat. Heimoneuvoston tarkoituksena on koordinoida tuoterunkoon perustuvien tuotteiden projekteja. Kokoontumisissa jaetaan tietoa tuoterunkoon toteutettavista ominaisuuksista ja tiedotetaan muuttuvista asioista. Kokoontuminen toimii myös foorumina asiakasprojekteissa esiintyneistä muutostarpeista tiedottamiseen.

5.2 Tietovaraston kehittäminen ja siinä huomioitavat asiat

Kolmannen ryhmän haastattelussa, jossa keskusteltiin tietovaraston kehittämisestä ja siihen liittyvästä prosessista, vaiheiksi nähtiin tietotarpeiden määrittely tai kartoittami-

nen, tietomallin suunnittelu sekä toteutus ja testaus. Myös toisessa haastatteluryhmässä käytiin tietovaraston kehitysprosessia läpi, mutta sitä tarkasteltiin tuotteistusnäkökulman kautta, joten pieniä nyansseja näiden välillä oli havaittavissa. Mainittakoon prosessista yleisesti, että varsinkin sen alkuvaiheet nähtiin osittain limittäisiksi. ”*Kokemus on osoittanut, että suunnitellessa havaitaan periaatteellisia seikkoja, joihin täytyy palata ja löytää vastaus ennen luotettavan toteutuksen rakentamista*”, tuumasi kohdeorganisaation tietovarastoinnista ja liiketoimintatiedon hallinnan teknisestä puolesta vastaava henkilö. Niinpä edellisiin vaiheisiin saatetaan joutua palaamaan.

5.2.1 Tietotarpeiden määrittely

Ensimmäisenä vaiheena kehitysprosessissa on tietotarpeiden määrittely, jossa nimensä mukaisesti yrityksen toimintaa tarkastelemalla pyritään ymmärtämään sovellusaluetta, jolle tietovarastoa rakennetaan. Haastateltavat mainitsivat, että käytössä tähän on kaksi perustapaa, joista ensimmäisessä rakennetaan pohja suunnittelulle hyödyntämällä käyttäjien tietotarpeita. Toisena vaihtoehtona nähtiin organisaation prosessien tarkastelu ja siihen liittyvien tietotarpeiden määrittely. Ensimmäisessä vaihtoehdossa katsottiin huonona puolena olevan, että kun mallin muodostamisen jälkeen tietotarpeet muuttuvat, alkuperäinen malli voi muuttua radikaalisti siten, että historiadatan saatavuus muutosta edeltävältä ajalta menetetään. Mallinnettaessa prosessia voidaan teoriassa saada mitä tahansa vastauksia myös tietotarpeiden muuttuessa.

Tärkeää tietotarpeiden määrittelyssä mainittiin olevan ensisijaisesti tuntemus organisaation toiminnasta, toimialasta ja siellä esiintyvistä käsitteistä, sillä niiden hahmottaminen on tulevan mallin rakentamisessa kaiken lähtökohta.

5.2.2 Tietomallin suunnittelu

Seuraava vaihe on tietomallin suunnittelu tai tietomallinnus. Tässä vaiheessa tietovarastolle tunnistetut tietotarpeet muunnetaan käsitteellisen tietomallin muotoon. Tietomallin muotoon otetaan tässä vaiheessa kantaa. Tämä tarkoittaa siis päätöstä joko normalisoituun tai dimensionaaliseen tietomalliin perustuvan tietovarastoarkkitehtuurin rakentamista. Arkkitehtuureista kolmannen haastatteluryhmän osanottajilla oli käytännössä kokemusta väylä-, hub-and-spoke -arkkitehtuureista ja eräästä näiden välimuodosta. He kuitenkin tähdensivät, että tietoarkkitehtuuri on useissa projekteissa tullut annettuna asiakasorganisaatiolta. Asiakas on siis alun perin määrittänyt, mitä mallinnustapaa tietomallissa tullaan käyttämään. ”*Tietovarastoinnin problematiikka on pyörinyt voimakkaasti tietomallin suunnittelun ympärillä*”, huomautti vanhempi tietovarastoasiantuntija.

Toisen haastatteluryhmän osalliset olivat konsensuksessa siitä, että tietomallin suunnittelussa tulee määritellä tietomallin lisäksi tietolähteiden ja tietovaraston tietomallin väliset mappaukset sekä se, miten data saadaan siirron toteutuksessa yksikäsitteisesti ja standardissa muodossa tietovarastoon. Teknistä toteutusta ei vielä silti käsitellä. ”*Tässä vaiheessa mappaukset ja tietovaraston tietomalli on sanallisessa muodossa, joten ETL-*

prosessin tarkempi määrittely selkiytyy vasta suunnittelussa”, totesivat ryhmän ohjelmistokehittäjät.

Huomioon otettaviksi seikoiksi mallinnuksessa nähtiin ensimmäisessä haastatteluryhmässä mallinnustavan valitseminen. Normalisoitu malli vaatii todella paljon esityötä ja valmistautumista itse mallinnusvaiheessa. Kun kokonaisuus on mallinnettu, työvaiheisiin kuuluu vielä tietotarpeisiin vastaavien analyysien tekeminen. Dimensionaalisen mallin tapauksessa saadaan nopeasti käyttäjälle käyttökelpoista tietoa. Mallinnuksen fokuksen pitämisen prosessissa takaa skaalautuvuuden säilymisen.

Erityisen tärkeää on ottaa huomioon datan tarkkuustaso sitä tietovarastoon tallennettaessa. Data tulee tallentaa kaikkein matalimmalla tarkkuustasolla, jotta porautuminen aivan lähdedatan tasolle on mahdollista. Näin yllättävätkin tietotarpeet saadaan tyydytettyä ilman muutoksia malliin. Tässä vastaan tulevat rajoitteet suorituskykyyn ja tallennustilaan liittyen, sillä datamäärä tulee olemaan valtava. Dimensioiden lisäämisen suhteen ottaa kantaa miten toimintaan, kun dimensioiden sisältö lähteissä päivittyy. Monesti se hoidetaan päivittämällä arvot ja pitämällä pääavain samana, sillä muuten menetetään kytkös aikaisempaan dataan.

Tietovarastointia enemmän suorittaneet ensimmäisen ryhmän osallistujat mainitsivat, että yrityksen toimialan tuntemus on suuressa roolissa, sillä siten tietotarpeet ymmärretään paremmin ja niitä osataan jopa ennakoida. Tässä vaiheessa pitää muistaa, että suunnitteluvaiheessa tehtävät suunnittelupäätökset ovat merkittäviä, sillä päätöksen korjaaminen johtaa usein historiatiedon menettämiseen. ”*Ideaalitilanteessa tietomalli on niin yleisellä tasolla, että esimerkiksi vanhan lähdejärjestelmän korvaaminen uudella ei aiheuttaisi muutoksia tietomalliin*”, kertoi vanhempi tietovarastoasiantuntija. Tällaisessa tapauksessa lähdejärjestelmien datan voi sijoittaa samaan faktatauluun ja data eri järjestelmien ajoilta on keskenään vertailukelpoisia.

Toisessa haastatteluryhmässä todettiin, että mallinnusvaiheessa on kyettävä muodostamaan tietomalli, joka sopii samalla toimialalla toimivasta yrityksestä riippumatta tilanteeseen. Se on tärkeää etenkin tuoterunkoon perustuvan tietovarastoratkaisun tapauksessa. Tämä vaatii yleisesti osaamista järjestelmien sisällöstä ja niiden taustalla olevasta logiikasta eli siitä, mitä tarkoitusta tai prosessia varten järjestelmä on alun perin kehitetty.

5.2.3 Toteutus ja testaus

Seuraavana vaiheena prosessissa katsottiin olevan toteutus ja testaus. Vaiheen nähtiin etenevän siten, että ensin toteutetaan edellisessä vaiheessa muodostettu tietomalli. Tämä tarkoittaa edellisen vaiheen käsitemalli muuntamista tietovaraston toteutusteknologiaksi valitun tietokannanhallintajärjestelmän ymmärtämään muotoon. Operaatio on usein melko suoraviivainen prosessi, kun päätös dimensionaalisuuden tai normalisoinnin suhteen on tehty. ”*Tietomalli pyritään testaamaan yhdessä käyttäjien kanssa esimerkiksi*

pyytämällä käyttäjää muodostamaan haluamaan raportteja omien tietotarpeidensa perusteella", mainitsi tietovarastoasiantuntija. Kolmannen haastatteluryhmän osallistujat kertoivat kuitenkin, että tietomallin testaus tapahtuu suurelta osin vasta tietovaraston käyttöönoton jälkeen, mikä on ongelmallista onnistumisen kannalta.

Seuraavana siirryttiin ETL-prosessin toteutuksen pariin, mikä tarkoittaa sen toteutusta joko itse ohjelmoimalla tai siihen tarkoitettua kolmannen osapuolen sovellusta käyttämällä. Testauksesta luodaan testaussuunnitelman ja suoritetaan lopuksi testaus testidatalla. Tietovarastoasiantuntija ja tietovarastointivastaava määrittelivät prosessissa olevan kyse lähdejärjestelmän tietojen muokkaamisesta ja siirtämisestä tietovaraston tietomalliin. Toisen haastatteluryhmän ohjelmistokehittäjät täydensivät, että ETL-prosessin itse toteuttaminen pohjautuu lähdejärjestelmien ja tietomallin välisiin mappauksiin ja prosessiin liittyvät suunnitteluratkaisut tehdään tässä vaiheessa.

Ennen toteutusta on pitänyt ottaa kantaa siihen, mitä teknologioita tietokannanhallintajärjestelmässä ja ETL-prosessissa käytetään. Kolmannen haastatteluryhmän haasteltavilla ei ollut valintaprosessista kokemusta, sillä asiakkaalle toteutusta tehdessä nämä päätökset on otettava asiakasyrityksiltä annettuina.

Toteutus- ja testausvaiheessa huomioonotettavat seikat voidaan jakaa tietomalliin ja ETL-prosessiin kuuluviksi. Tietomallia toteutettaessa tulee ottaa kantaa analyysien suorituskyvyn ja käyttäjämäärien vaikutukseen. Esimerkiksi kolmannen haastatteluryhmän osanottajat olivat hiljattain olleet toteuttamassa tietovarastoa, jolla samanaikaisia käyttäjiä on mahdollista olla useita satoja, mikä ratkaistiin muodostamalla erillinen datakomeron, jossa oleva data on valmiiksi analyysin vaatimaan muotoon laskettu. Kyseessä oli muun muassa keskiarvoja, mediaanin ja summien laskemista.

ETL:n kannalta huomioitavia asioita nostettiin esimerkiksi sen testattavuuden suunnittelu. Tällä hetkellä toisen ja kolmannen haastatteluryhmän osallistujat näkivät sen erityisen haastavaksi, vaikka ETL-työkalujen avulla toimenpiteet ovatkin logiikaltaan helposti seurattavissa. Testaus on monesti tietovarastosta tehtävien satunnaisotosten varassa. Toiseksi huomioitavaksi seikaksi mainittiin prosessin suorituskyky, joka nousee esiin datamäärien ollessa suuria tai aikavaatimusten ollessa tiukkoja. Kolmannessa ryhmässä osallisena ollut tuotteen omistaja huomautti, että datan ensimmäisen latauksen yhteydessä saattaa tulla vastaan vakavia ongelmia aikataulurajoitteiden suhteen ja tietojen hakua operatiivisesta kannasta saatetaan joutua jaksottamaan. Tietovarastovastaava mainitsi, että tiettyjä latauksia voidaan rinnakkaistaa, mutta sekin tulee suunnitella taulujen välisiin riippuvuuksiin nojaten. Tietovaraston tietomallin tai lähdejärjestelmien skeeman muuttuessa tulee ottaa kantaa, toimintaan, kun historiadatasta ei ole saatavissa dataa, jonka uusi tietovaraston tietomalli edellyttää tallennettavaksi. Samalla tavalla dimensiotaulun tietojen päivittyessä ei lähdetä luomaan uutta riviä, sillä siten kytkökset historiaan menetetään.

Viimeisenä huomioonotettavana seikkana kolmas haastatteluryhmä mainitsi ETL-prosessin virhesietoisuuden. Tällä haastateltavat tarkoittivat virheistä toipumisten, virheilmoitusten tekemisen ja suorituksen keskeyttämisen suunnittelua. Monesti prosessi ajetaan hiljaiseen aikaan, minkä jälkeen prosessin epäonnistumisesta on tiedotettava jotenkin.

5.3 Ohjelmiston laajennettavuus, räätälöitävyys ja uudelleenkäytettävyys

Kolmannen alatutkimusongelman kohteena olevat laajennettavuuteen, räätälöitävyyteen ja uudelleenkäytettävyyteen liittyvät huomiot esiintyivät jossain määrin sekä ohjelmiston tuotteistamisen ja tietovaraston kehittämisen vaiheiden yhteydessä, mutta ne esitellään tässä erillään hieman seikkaperäisemmin. Tarkastellut ominaisuudet esitellään siten, miten ne esiintyvät omissa konteksteissaan eli tuotteistamisessa ja tietovarastoinnissa.

5.3.1 Laajennettavuus, räätälöitävyys ja uudelleenkäytettävyys tuoterunkoon perustuvassa kehityksessä

Tuoterunkoon perustuvassa ohjelmistokehityksessä laajennettavuus ja räätälöitävyys kulkevat ensimmäisen ja toisen haastatteluryhmän mukaan hyvin pitkälti käsi kädessä ja niiden hallintaan teknisessä mielessä voidaan lähestyä monesta eri näkökulmasta. Laajennettavuus ja räätälöitävyys voidaan saavuttaa ohjelmistoarkkitehtuurin suunnittelumalleilla. Esimerkkinä niistä on ohjelmiston modulaarisuus, jota on hyödynnetty myös olemassa olevan varastonhallintajärjestelmän tuoterungossa. ”Suunniteltu ohjelmisto on jaettu loogisiin moduuleihin, joiden välillä on yksi tai kaksisuuntaisia riippuvuuksia. Kutsukohdat muihin moduuleihin on rajattu, jolloin muutoksia tehdessä muutettavien kohtien määrä saadaan minimoitua”, mainitsivat ohjelmistoarkkitehdit. Samaan he sanoivat pyrkivän kerrosrakenteeseen perustuva ohjelmistoarkkitehtuuri, johon kohdeyrityksen tämänhetkinen ohjelmistotuoterunko nojautuu. Siinä ohjelmisto rakentuu eri operaatioita hoitamaan suunnitelluista kerroksista, joiden roolit ja vastuut tietyn toiminnon suorittamiseksi on tarkkaan määritelty. Haastateltavien mukaan tällä suunnitteluratkaisulla saadaan rajattua muutosten vaikutuksia ja tiettyjen moduulien puuttuminen ei vaikuta kokonaisrakenteeseen.

Edellisten lisäksi laajennettavuuden ja räätälöitävyyden mahdollistamiseksi mainittiin plugin-suunitteluratkaisut, jotka nousivat esille sekä ensimmäisessä että toisessa haastattelussa. Plugin-ratkaisussa tiettyihin kohtiin ohjelmistoa toteutetaan rakenteita, joihin voidaan tarpeen mukaan kiinnittää vaihtoehtoisia ohjelmistokomponentteja tai komponenttiryhmiä. Arkkitehdit tähdensivät, että tietyissä kohdissa tavallinen konfigurointi tai parametrointi ei riitä, vaan toteutus se vaatii monimutkaisempaa mekanismia. Haastateltavat korostivat, että vaikka teoriassa tämänkaltaisia kohtia voidaan rakentaa lähes mihin tahansa kohtaan ohjelmistossa, on jo esiselvitysvaiheessa tärkeää tunnistaa paljon

vaihtelua sisältävä kohdat. Kohdeyrityksen toteuttaman varastonhallintajärjestelmän tuoterungossa tällaisia pisteitä sanottiin olevan useita kymmeniä. Edellisen lisäksi tulee arvioida, kuinka paljon vaihtoehtoisia toteutuksia kohdassa tulee mahdollisesti olemaan.

Ohjelmiston räätälöinnin asiakaskohtaiseksi mahdollistaa pääarkkitehdin mukaan niin sanottu palapelirakenne, jossa tiettyä rajattua operaatiota suorittamaan toteutetaan itsenäinen ohjelmistokomponentti. ”*Komponenttien ei tarvitse tietää muiden rakenteiden olemassaolosta tai prosessien suoritusjärjestyksestä*”, kertoi vanhempi ohjelmistosuunnittelija. Tämänlaiset palapelin palat voidaan kytkeä toisiinsa halutussa järjestyksessä tai vaihdella operaatioita suorittavia komponentteja prosessin suorituksen räätälöimiseksi. Teknisesti rakennelmat toteutetaan määrittelemällä ne workflow-konfiguraatiotiedostossa.

Ensimmäinen ja toinen haastattelu paljasti, että laajennettavuuden ja räätälöitävyyden kannalta ohjelmistokomponenttien suunnittelussa tärkeää rajapintojen suunnittelu. ”*Riittävän laajat rajapinnat tukevat monipuolisempia toteutuksia*”, mainitsivat arkkitehdit. He olivat monesti törmänneet liian suppeisiin rajapintoihin, joiden avulla saavutetaan vain näennäistä räätälöitävyyttä.

Kaikkien haastatteluiden yhteydessä korostui se, etteivät tekniset seikat ja hyvä arkkitehtuurisuunnittelu yksin riitä takaamaan ohjelmiston laajennettavuutta ja räätälöitävyyttä. Erittäin merkittävässä asemassa on tuntemus sovellusalueesta tai toimialasta, jolle tuoterunkoon perustuvaa ohjelmistoa ollaan toteuttamassa. Käsitteiden ja toimintalogiikan huonoa tuntemusta ja niiden myötä määritettyjä yleisiä tuoterungon vaatimuksia ei saa väheksyä tuotteistamisen alkuvaiheessa. ”*Hyvää arkkitehtuuria ei voida synnyttää pelkällä teknisellä osaamisella, vaan teknisen osaamisen ja sovellusalueen tuntemuksen yhdistelmällä”,* mainittiin ensimmäisessä haastattelussa.

Ohjelmiston uudelleenkäytettävyys nähtiin ensimmäisessä ja toisessa haastattelussa tuoterungon sisäänrakennetuksi ominaisuudeksi. Edellä mainittujen räätälöintimekanismien ohella yhdeksi tärkeäksi seikaksi mainittiin esimerkiksi peruskäsitteisiin liittyvien olettamusten kovakoodauksen välttämisen. ”*Perinteisillä versionhallintatyökaluja ei pidä unohtaa, koska niillä on olennainen rooli uudelleenkäytettävyyden mahdollistamisessa*”, mainitsi vanhempi ohjelmistokehittäjä.

5.3.2 Laajennettavuus, räätälöitävyys ja uudelleenkäytettävyys tietovarastoinnissa

Keskeisessä asemassa tietovarastojen laajennettavuuden kannalta on sen tietomallin suunnittelu. Haastateltavien mukaan suunnitellessa tietovarastoa, jossa tulevien lähdejärjestelmien tarkkaa toteutusta tai muotoa voida tietää etukäteen, pitää olla tuntemus siitä funktionaalisesta kokonaisuudesta jota ryhdytään mallintamaan. Esimerkkinä mainittiin tuntikirjausjärjestelmä. Tekijän pitää tuntikirjausjärjestelmän tapauksessa tietää, mitkä ovat yleiset vaatimukset tuntikirjausjärjestelmille ja mitä tarkoitusta varten ne on

rakennettu. Haastatteluun osallistuneet mainitsivat kuitenkin, että tällaisissa tapauksissa kannattaa miettiä, kuinka suuri työmäärä on vaivan arvoista, sillä lisättävien järjestelmien toteutus saatetaan joutua tekemään kuitenkin räätälöintinä.

Toisessa ja kolmannessa haastattelussa molemmissa todettiin, että tietomallissa laajennettavuutta mahdollistavia seikkoja ovat tietomalliin tallennettavan datan tarkkuustaso, siinä käytettävän käsitteistön geneerisyys ja järjestelmäriippumattomuus. Tuoterungon omistaja tähdensi, että niillä saavutaan samalla uudelleenkäytettävyyttä, koska geneerinen käsitteistö voi sopia toimialalla eri toimijoiden käyttöön. Tietovarastoasiantuntija määritteli tarkkuustason tarkoittavan datan tallennusta dimensioissa ja faktoissa alhaisimmalle mahdolliselle tarkkuustasolle, jottei alun perin luotu ratkaisu muodostu rajoittavaksi muita tietolähteitä lisättäessä. Käsitteistön geneerisyys ja järjestelmäriippumattomuus sitä, että se on muodostettu prosessin yleisten käsitteiden perusteella eikä ole esimerkiksi käytetty lähdejärjestelmäsidonnaista termistöä eikä vaikkapa tietokantataulujen nimiä. Lisäksi dimensioiden suunnittelu mahdollisimman yleiskäyttöisiksi tukee tätä ajatusta.

Kolmannen haastatteluryhmän kanssa keskusteltiin myös ETL-prosessin laajennettavuudesta. Kaupalliset työkalut mahdollistavat melko helpon muutosten suorittamisen ETL-prosessiin, sillä käyttöliittymät niissä ovat useimmiten graafisia. Niiden heikkoutena haastateltavat näkivät, että ne sitovat tiettyihin teknologioihin, mikä heikentää uudelleenkäytön mahdollisuuksia. Uudelleenkäytettävyyttä nähtiin parantavaksi kolmannen osapuolen tietokantariippumattomien ETL-työkalujen, kuten cloverETL:n tai Pentaho:n, käyttö. Yrityksen tietovarasto- ja BI-vastaava tiesi, että niissä suorituskyky ei välttämättä ole samalla tasolla kuin kaupallisissa työkaluissa.

Toisessa haastatteluryhmässä oltiin edellisen vastaisesti vahvasti sitä mieltä, että parhaiten laajennettavuus ETL-prosessissa saavutetaan toteuttamalla se itse. Ratkaisussa tulee soveltaa sopivia suunnitteluratkaisuja, kuten plugin-rakennetta ja kerrosarkkitehtuuria. Haastateltavat totesivat, että ETL:n eri osa-alueet tulee nähdä omina plugin-toteutuksinaan, jotka voidaan helposti korvata toisenlaisella toteutuksella. Haastatteluissa tulkittiin mahdolliseen uuden tietolähteen lisääminen ja siihen liittyvä ETL-prosessi omiksi loogisiksi kokonaisuuksikseen, joilla ei ole sidonnaisuutta muihin prosesseihin. Laajennettavuus myös ETL:n tapauksessa palaa lopulta tietomalliin ja sen suunnittelun onnistumiseen.

Kolmannen ryhmän tuotteen omistaja mainitsi, että kun räätälöintejä tehdään lähdejärjestelmän tietorakenteisiin, aiheutuu tästä toimenpiteitä usein ETL-prosessiin ja tietovaraston tietomalliin. ”*Tietovaraston elinkaaren aikana tietomalli tullaan rikkomaan moneen kertaan, mihin pitää osata varautua*”, hän painotti. Riippuvuussuhde tietovaraston ja lähdejärjestelmien kehityksen välillä muodostuu voimakkaammaksi ja muutostyöt vaativat aina molempien ympäristöjen huomioimisen. Näin ollen kehityksestä tulee jäykempää.

Uudelleenkäytettävyyden suhteen kolmannelta haastatteluryhmällä oli joitain ajatuksia. He mainitsivat, että teoriassa tietovaraston tietomalli voisi olla uudelleenkäytettävä siten, että se sopii kahden tai useamman eri yrityksen toiminnan kuvaamiseen. Tällaisen yleisen tietomallin luomisen vaatimuksia keskusteltiin jo laajennettavuuden tapauksessa. Kertauksena mainittakoon, että mallintajan pitää tuntea toimiala ja yleisimmät prosessit, joista tietovaraston data tullaan keräämään. Näin ollen tietomallista saadaan riittävän geneerinen. Vanhempi tietovarastoasiantuntija kaavaili, että myös ETL-prosessista voidaan tehdä tiettyyn pisteeseen uudelleenkäytettävä esimerkiksi rakentamalla lähteiden ja kohteiden väliin niin sanottuja vakiomuotoisia varjotauluja. Niiden populoinnista vastaava ETL-prosessi toteutettaisi kuitenkin asiakaskohtaisesti. Tämä tehtäisi luonnollisesti siksi, että harvalla yrityksellä on täysin identtisiä lähdejärjestelmiä. Varjotauluista tehtävä ETL-prosessi voitaisiin kuitenkin vakioida. Haastateltavat mainitsivat, että tällaisessa tilanteessa pitää puntaroida toteutuksen hyötyjä, sillä tietty osuus joudutaan kuitenkin tekemään asiakaskohtaisena räätälöintinä.

Toinen näkökulma uudelleenkäytettävyyteen saatiin ETL-prosessin itse toteuttaminen käyttämällä esimerkiksi alustariippumatonta .NET C# -teknologiaa, mihin oli vahva konsensus toisen haastatteluryhmän osanottajilla. Heidän mukaan tästä saavutetaan se hyöty, että asiakkaalta ei vaadita mitään tiettyä tietokantateknologiaa vaan olemassa olevat lisenssit ovat hyödynnettävissä. Näin ollen toteutus voidaan toteuttaa todella laajalle kirjolle asiakkaita. Käytännössä kuitenkin kohdeorganisaation nykyisillä asiakkailla haastateltavat mainitsivat olevan käytössä pääasiassa Oraclen ja Microsoft SQL Serverin tietokantateknologioita, joten kirjo ei tällä hetkellä ole kovin laaja. Teknologiariippumattomuutta vähentää myös tietovaraston raportointikerroksen eriyttäminen, mikä mahdollistaa vaihtelevien raportointi- ja analyysisovellusten käytön.

Toisen haastatteluryhmän ohjelmistokehittäjät mainitsivat, että uudelleenkäytettävyyden takaa myös parametrien käyttö. Ne antavat mahdollisuuden muutella tietokannanhallintajärjestelmän eri ominaisuuksia. Haastateltavat nostivat esiin esimerkiksi lisättävien rivien määrän tietokantatransaktiota kohden. Näin ollen suorituskyvyltään eritasoiset ympäristötkin voidaan huomioida etukäteen.

5.4 Tarkistuslistan vastaavuus käytännön kanssa

Tutkimushaastattelun yhtenä teemana tai vaiheena oli teoriaosan perusteella muodostetun tarkistuslistan esittely ja kommenttien kerääminen. Tarkistuslista itsessään jakautui kolmeen osaan: ohjelmiston tuotteistus, tietovaraston kehitys sekä laajennettavuuden, räätälöitävyyden ja uudelleenkäytettävyyden huomioiminen yleisesti ohjelmiston tai tietovaraston tapauksessa. Tarkistuslista on esitetty tässä tutkimuksessa liitteessä 1, eikä sen sisältöä käydä tässä yksityiskohtaisesti läpi.

5.4.1 Teorian ja käytännön suhde vastaavuus tuoterunkoon perustuvassa kehityksessä

Tarkistuslistan läpikäynti tuotteistamisprosessin ja sen aikana huomioitavien seikkojen osalta ensimmäisessä ja toisessa haastatteluissa näkivät käytännön toiminnan kanssa melko samankaltaisiksi. Prosessin vaiheet olivat suurelta osalta sisällöltään samanlaiset. Pieniä eroja tulkinnoissa suoritettavien toimenpiteiden järjestyksestä kuitenkin esiintyi.

Alustan suunnitteluvaiheeseen ensimmäisen ryhmän arkkitehdit halusivat lisätä huomioonotettavaksi asiaksi arkkitehtuuriin toteutettavien rajapintojen tärkeyden sekä arvioinnin siitä, miten tehtävät muutokset vaikuttavat tuotannossa asiakaskohtaisiin installaatioihin. He arvioivat monituotelinjojen tapauksessa kehityksen monimutkaistuvan entisestään ja vaativan tulevaisuudessa enemmän huomiota. Alustan testauksesta haastateltavat mainitsivat huomioitavaksi sen, että monissa tapauksissa tuoteperhearkkitehtuurin toimivuutta pitää testata asiakaskohtaisten installaatioiden kautta, sillä itsenäisenä ja suuresti tietokannasta riippuvaisena toteutuksena se ei ole alustankehittäjien suoritettavissa. Haastateltavat korostivat myös automaattisten testaustyökalujen merkitystä tässä vaiheessa.

Haastateltavat ensimmäisessä ja toisessa ryhmässä eivät olleet mieltäneet teoriassa esiintyvän tuoterungon tai tuotelinjan käsitteen sisältävän niin laajasti ohjelmistoartefakteja kuin teoriassa oli esitetty. Esimerkiksi varastonhallintajärjestelmän tuoterunko miellettiin yksinomaan ohjelmistoarkkitehtuurin konkreettiseksi toteutukseksi, jota kohdeyrityksessä kutsutaan nimellä MotusLite. Heidän puheistaan oli kuitenkin tulkittavissa, että tuotteen asennusta ja testaus helpottavat apuohjelmat sekä dokumentoitu käsitemalli ja tietokantadokumentaatio ovat olennainen osa MotusLite:ä ja näin ollen kuuluisivat tuoterunkoon.

5.4.2 Teorian ja käytännön suhde tietovarastoinnissa

Tietovarastojen kehitysprosessin suhteen nähtiin käytäntö ja teorian ehdottama malli melko yhteneväisinä kolmannessa ryhmähaastattelussa. Todellisuudessa vaiheita ei ajatella mallin ehdottaman kuvan tarkkuustasolla, mutta tarkemmin tarkasteltuna mallista löytyy kaikki ne elementit, jotka tulevat toteutettua käytännössä.

Tarkistuslistaa läpikäytäessä tietovarastovastaava ja -asiantuntija olivat yhtä mieltä operatiivisten järjestelmien eroista tietovarastoinnin kanssa, mutta halusivat lisätä, että tietovarastointiprojekteissa asiakkaan päässä osallistuvien henkilöiden kirjo on huomattavasti laajempi. Esimerkkinä he nostivat varastonhallintajärjestelmän toteuttaminen, jossa pääasiassa suunnittelussa ja toteutuksessa on mukana henkilöitä, joiden osaaminen liittyy vahvasti varaston toimintaan ja sen prosesseihin. Tietovarastoja toteutettaessa ollaan tekemisissä parhaimmassa tapauksessa konsernin eri toimintojen edustajien kanssa, joiden osaamisalueet vaihtelevat.

Tietovarastoarkkitehtuurin valintaan löytyi kolmannen ryhmän haastateltavilta niin ikään mietteitä. Muodostetussa listassa esiintynyttä resurssien rajoitteiden vaikutusta tietovarastoarkkitehtuurin valintaan nähtiin perustelluksi. Sen lisäksi yhdeksi merkittäväksi tekijäksi nähtiin kohdeorganisaation liiketoiminta-alue ja sen erityispiirteet, jonka haastateltavat halusivat lisätä listaan. Vastakkainasettelua esimerkiksi finanssisektorin ja varastonhallinnan välillä käytettiin kuvaamaan tätä seikkaa. Arkkitehtuurityypin valintaan nähtiin vaikuttavaksi tietovarastoa kehittävän organisaation käyttämät kehitysmenetelmät. Ketteriä kehitysmenetelmiä käytettäessä datakomeroihin vahvasti sidonnaiset arkkitehtuurit, kuten väyläarkkitehtuuri, sopivat luontevimmin valinnaksi.

Viimeisenä nostona tietovarastovastaava mainitsi arkkitehtuurin valinnassa kehityksessä mukana olevien liiketoimintapuolen henkilöiden osaaminen tarkistuslistassa mainitun IT-henkilöstön osaamisen ohella. Hän tiesi kokemuksen perusteella, että esimerkiksi organisaationlaajuinen tai Hub-and-Spoke -arkkitehtuuri ovat vaikeasti hahmotettavissa, mikä saattaa johtaa siihen, että asiakas ehdottaa arkkitehtuurityypiksi jotain muuta, kuin sovellusalueeseen sopisi parhaiten.

Prosessien vaiheiden huomioon otettavista seikoista oltiin kolmannessa ja toisessa haastattelussa pitkälti samaa mieltä. Osaa seikoista ei ollut osattu ottaa huomioon, mutta niiden huomioimatta jättämisestä ei nähty olevan vakavia seurauksia. Vanhempi tietovarastoasiantuntija otti kantaa kuitenkin vaatimusten määrittelyssä esiintyneeseen jakoon data-, tavoite- ja käyttäjävetoisesta vaatimusten keräämisestä siten, etteivät vaihtoehdot olisi toisensa poissulkevia vaan pikemminkin täydentäviä menetelmiä. Hän näki esimerkiksi mahdolliseksi, että lähdetään liikkeelle datavetoisesti, mutta jossain vaiheessa käyttäjät otetaan mukaan ja kerätään vaatimuksia tavoitevetoisen lähestymistavan mukaisesti.

Suunnittelun ja kehityksen sisällöissä ja huomioon otettavissa seikoissa oli myös olennaisia yhteneväisyyksiä. Toisen ja kolmannen ryhmän haastateltavien oli kuitenkin vaikea ottaa kantaa teknologiaradan vaiheisiin, sillä useimmiten projekteja toteutettaessa teknologiset valinnat, kuten tietokannanhallintajärjestelmä ja käytettävä ETL-työkalu, tulevat asiakkaalta annettuna eikä niihin ole mitään vaikutusvaltaa. Keskeisimpänä vaikuttavana tekijänä tilanteessa on olemassa olevat lisenssit eli taloudellinen näkökulma. Dataradalla puolestaan metadatan mallinnukseen ja ylläpitoon kiinnitettiin huomiota. Sen nähtiin teoriassa toimivan, mutta harvassa tietovarastoprojektissa sitä käytännössä toteutetaan. Ajatuksen tasolla lähteestä ylläpidettävä metadata on hyödyllisenä tapauksessa, jossa yksi lähdejärjestelmä vaihtuu kokonaan.

Kahdessa jälkimmäisessä haastattelussa keskusteltaessa tietovarastoratkaisun laajennettavuudesta aikaisemmin tietovaraston kehitysprosessin yhteydessä esiintyneet asiat toistuivat, mikä linjassa tarkistuslistaan tehtyjen poimintojen kanssa. Haastateltavat halusivat lisätä räätälöitävyyden mahdollistamiseen automaattisen testauksen rakentamisella.

Näin lähdejärjestelmiin ja tietovarastoon tehtävien muutosten vaikutuksia esimerkiksi raporteille voitaisiin tehokkaasti testata.

6. POHDINTA

Tässä kappaleessa tarkastellaan tutkimuksen teoriaosuudessa käytetyn kirjallisuuden ja empiriaosuuden tulosten yhteyttä tunnistamalla keskeisimmät seikat ja eroavaisuudet. Ensin käsitellään erikseen tuoterunkoon perustuvan kehityksen ja tietovarastojen prosesseja ja huomioitavia seikkoja. Seuraavaksi pohditaan tuoterungon ja tietovarastojen laajennettavuutta, räätälöitävyyttä ja uudelleenkäytettävyyttä. Lopuksi alatutkimusongelmien teemat yhdistämällä muodostetaan prosessimalli tietovaraston tuotteistamiselle, mikä on varsinainen vastaus päätutkimusongelmaan.

Kokonaisuudessaan haastatteluiden voidaan sanoa onnistuneen, sillä saadut kommentit olivat hyvin linjassa käsitellyn teorian kanssa. Aihepiirejä käsiteltiin hiukan eri näkökulmista ja käyttämällä eri terminologiaa kuin teoriassa, mutta perusajatus oli käytännössä sama. Haastattelujoukon koko suunniteltiin alun alkaen lopullista suuremmaksi, mutta tuoterunkoon perustuvan kehityksen osaajia haastatellessa hyvin pian samat vastaukset alkoivat toistua, jolloin voidaan katsoa saturaatiotason tulleen saavutetuksi. Näin ollen päätettiin olla järjestämättä neljättä haastattelutilaisuutta, jossa oli alun perin tarkoituksena käsitellä pääosin tuoterunkoon perustuvaa ohjelmistokehitystä. Tietovarastointiosaamista omaavien henkilöiden suhteen olisi toivonut enemmän osallistumista, mutta realiteetit relevanttia osaamista omaavien henkilöiden määrän tulivat vastaan.

6.1 Kehitysprosessi ja siinä huomioitavat seikat

Tässä kappaleessa käsitellään erikseen sekä tuoterunkoon perustuvan ohjelmiston sekä tietovaraston kehitysprosessia ja prosessin vaiheiden aikana huomioitavia seikkoja.

6.1.1 Ohjelmiston tuotteistaminen ja huomioitavat asiat

Ohjelmiston tuotteistaminen nähtiin sekä kirjallisuudessa että käytännössä varsin yhteneväiseksi prosessiksi (ks. esimerkiksi Pohl et al. 2005; Linden et al. 2007; Koskimies & Mikkonen 2005). Eroavaisuudet olivat pieniä ja liittyivät lähinnä käytettyyn termistöön ja siihen, miten minkä vaiheen vastuulle kukin toimenpide kuuluu. Vaiheiden rajojen ei haastatteluissa nähty olevan niin ehdottomia, kuin ne on kirjallisuudessa esitetty. Kokonaisuudessaan sisältö oli sama. Tuotteistamisen eli tuoterunkoon perustuvan ohjelmistokehityksen katsottiin tapahtuvan kuuluvan täysin alustankehitysprosessin vastuulle.

Esimerkiksi Pohl et al. (2005) mainitsivat vaiheiksi tai toimenpiteiksi prosessissa seuraavat: vaatimustenmäärittely, alustan suunnittelu, alustan toteutus sekä testaus. He

edelleen totesivat prosessin kunkin vaiheen tuloksena syntyvän tuoterunkoartefakteja, jotka toimivat pohjana seuraavan vaiheen toimille. Prosessiin katsotaan kuuluvaksi edellisten lisäksi tuotteen keskitettyä hallintaa suorittava kokonaisuutensa, joka on enemmänkin jatkuvaa toimintaa kuin yksittäinen prosessivaihe.

Prosessin vaiheista alustan vaatimustenmäärittelyssä tunnistetaan eri asiakkaiden tarpeiden perusteella yleiset vaatimukset tuotteistettavalle järjestelmälle ja tunnistetaan ne vaatimukset, jotka vaihtelevat asiakaskohtaisesti. Tunnistamisen tuloksena syntyy Pohl et al:n (2005) mukaan muunneltavuusmalli, joka on tämän vaiheen yksi tärkeimmistä tuotoksista. Metzgerin ja Pohlin (2014) mukaan muunneltavuusmalli voidaan mieltää joko osaksi tuoterunkoa tai itsenäiseksi kokonaisuudekseen. Haastatteluissa huomautettiin, että vaihe vaatii huomattavat resurssit toteutuksessa onnistuakseen. Muita tärkeitä huomioitavia seikkoja on yleisten asiakasvaatimusten löytäminen siten, etteivät yksittäisen asiakkaan tarpeet ohjaa liikaa tulevan tuoterungon määrittelyä. Vaatimuksia on myös kyettävä ennakoimaan jossain määrin lähitulevaisuudesta. Linden et al. (2007) mainitsivat lähitulevaisuuden lisäksi tässä vaiheessa huomioitaviksi seikoiksi myös pitkällä aikavälillä mahdollisesti realisoituvat tuotteet ja asiakasvaatimukset, mitä ei haastatteluissa nähty juuri tarpeelliseksi.

Seuraavana prosessissa on alustan suunnittelu, jonka keskeisimpiä toimenpiteitä on referenssiarkkitehtuurin suunnittelu sekä sen toteutus (Pohl et al. 2005; Koskimies & Mikkonen 2005). Lisäksi muunneltavuusmallin perusteella variaatiopisteiksi tunnistettuihin kohtiin suunnitellaan komponentit karkealla tasolla ja toteutetaan niille rajapinnat. Suunnittelun tuloksena toteutetaan uudelleenkäytettävät komponentit eri ympäristöistä tunnistettuihin muunneltavuustarpeisiin. Arkkitehtuurisuunnittelua toteutettaessa tulee muistaa, että tehdyillä arkkitehtuuriratkaisuilla on kauaskantoiset ja laaja-alaiset vaikutukset, sillä arkkitehtuuri tulee olemaan käytössä lukuisilla eri asiakasorganisaatioilla.

Seuraavana vaiheena on alustan toteutus, jossa alustan suunnittelun perusteella syntynyt arkkitehtuurikuvaus konkretisoidaan toimivaksi järjestelmäksi. (Pohl et al. 2005). Sekä kirjallisuudessa (esimerkiksi Atkinson et al. 2000) että haastatteluissa mainittiin, että toteutus tehdään useimmiten tukeutumalla komponenttipohjaiseen ohjelmistokehitykseen. Haastatteluissa nousi esiin myös että, toteutuksen aikana samalla toteutetaan kaikkea muutosten hallintaa, asennusta ja muuta ylläpitoa automatisoivat työkalut. Haastateltavat korostivat, että huomioitavan arvoista vaiheen aikana on se, että toteutettavassa ohjelmakoodissa korostuu sen laatuvaatimukset, sillä pienetkin muutokset uudelleenkäytettävään tuoterunkoon poikivat helposti suuria työmääriä. Siksi koodin refaktorointi nähtiin merkittäväksi riskiksi.

Viimeisenä vaiheena itse prosessissa nähtiin testaus, jossa arkkitehtuurikokonaisuus pyritään testaamaan painottaen sitä siten, että kaikkein monimutkaisimmat tai epävarmuutta sisältävät muunneltavuustapaukset saavat suuremman painoarvon. Tiettyjä omi-

naisuuksia ei voida testata tässä ympäristössä, joten osa testausvastuusta siirtyy tuotekehitysprosessille. Northrop et al. (2007) mainitsivat myös, että tuoterungon testaaminen on perinteistä ohjelmistoa työläämpi operaatio, sillä toteutus on monimutkaisempi ja soveltuu useampaan käyttötilanteeseen. Haastateltavat mainitsivat edellä mainittuja puutetta paikkaamaan esimerkiksi katselmointikäytännöt.

Erillisenä vaiheena alustankehityksessä varsinaisten toteutusvaiheiden ohella tuotteenhallinta, joka oli Pohl et al. (2005) sekä Koskimiehen ja Mikkosen (2005) käyttämä nimitys. Kohdeorganisaatio oli omaksunut termin keskitetty hallinta. Siinä koordinoidaan tuoterungon ominaisuuksien kehitystä ja eri tuoteinstanssien omistajien esiin nostamia asiakasvaatimuksia. Schmid:in (2000) mukaan tämä on käytännössä tuotteen rajauksen ylläpitoa eli ottamista kantaa ominaisuuksiin, joita tuoterunko tukee. Tuotteenhallinnan vastuulle kuuluu myös kaupallisiin näkökulmien, kuten markkinoinnin ja tuotannon, pohdiskelua.. Rajauksen määrittelyssä tulee huomioida sen laajuus. Määritettäessä rajausta on muistettava, että liian tiukka rajaus heikentää tuoterungon laajennettavuutta ja kasvupotentiaalia. Liian väljä rajaus aiheuttaa puolestaan tehottomuutta kehityksessä.

Tuotteenhallinta oli suurin piirtein samansisältöistä sekä kirjallisuudessa että haastatteluissa. Haastatteluissa kuitenkin vaihe ei ollut yhtä strategisen tason toimintaa kuin kirjallisuudessa esitettiin. Lisäksi esimerkiksi Weerd et al. (2006) esittivät tuotteenhallinnan monet tehtävät markkinointifunktion yhteydessä tehtäväksi toiminnaksi, mikä taas haastatteluissa ei noussut ollenkaan esille.

Tuoterunkoon perustuvan kehityksen yksi olennaisimmista aktiviteeteista alustankehityksessä on muunneltavuuden hallinta ja mallinnus (Pohl et al. 2005; Gurp et al. 2001; Chen et al. 2009). Haastatteluissa tätä osa-aluetta ei tunnistettu omaksi osa-alueekseen, mutta saatujen vastausten perusteella kirjallisuudessa esiintyneitä toimenpiteitä suoritetaan jollain tasolla. Muunneltavuuden hallinnan tuloksena syntynyt kuvaus toiminnallisista ja teknisistä ominaisuuksista helpottavat järjestelmän ylläpitoa ja kommunikointia sekä tuotekehitysprosessille että muille sidosryhmille. Muunneltavuusmalli voidaan nähdä yhtenä osana tuoterunkoa. Muunneltavuuden mallinnuksessa tulee huomioida yhtenäisen kuvaustavan käyttö kuvausten kesken. Tässä apuna voidaan käyttää sekä graafista että tekstimuotoista notaatiota.

Muunneltavuuden hallinta edesauttaa tuoterunkoon kohdistuvien muutosten toteutumista. Tuoterunko kehittyy toimintaympäristönsä muukaan joskus radikaalistikin. Vaatimukset ja toteutusteknologiat muuttuvat, joten kehityksessä on kyettävä huomioimaan näiden muutosten etenemismekanismit ja kohdat, joihin muutokset vaikuttavat. Jatkuva muutosten seuranta ja konkreettisen suunnitelman eteneminen auttavat hallitsemaan evoluution toteutumista (Svahnberg 2003).

Monituotelinjat tai -tuoterungot, joissa kehitetään samaan uudelleenkäytettävään arkkitehtuuriin perustuvia ja toisiinsa riippuvuussuhteessa olevia ohjelmistoja, aiheuttavat

kehityksessä lisähuomioita vaativia kysymyksiä, mikä mainittiin sekä haastatteluissa ja esimerkiksi Holl et al:n (2012) tekstissä. Tärkeimpiä lisähuomioita ovat riippuvuussuhteiden tunnistaminen ja niihin liittyvien muutosten hallinta. Muita ovat esimerkiksi suuren mittakaavan järjestelmien mallinnukseen soveltuva notaatio ja selkeät rajapinnat tuotelinjojen välillä.

6.1.2 Tietovaraston kehittäminen ja huomioitava seikat

Tietovaraston kehitys poikkeaa tutkimuksen kohteena olevan yrityksen toimintatavoista tuotteistamisteorian tapaan hyvin vähän. Tutkimukseen empiiriseen osaan osallistuneet totesivatkin, että selvästi asioita tehdään oikein kohdeorganisaatiossa. Tietovaraston kehitys jaetaankin tutkimuksen rajauksen mukaisten toimenpiteiden osalta projektin suunnitteluun, vaatimusten määrittelyyn sekä suunnitteluun ja toteutukseen (Ponniah 2010).

Projektin suunnitteluvaiheessa tehdään projektisuunnitelma tietovaraston toteutusta varten. Kimball et al. (2013) toteavat, ettei tietovarastointiprojekti suuresti eroa tavallisesta ohjelmistoprojektista, mutta monialaisuus aiheuttaa lisävaatimuksia projektin seurannalle, rajauksen hallinnalle ja sidosryhmille kommunikoinnille. Tutkimushaastattelun perusteella voidaan tehdä sama johtopäätös ja huomata, että projektin toteutuksessa osallisena olevien henkilöiden osaamistaustat ovat hyvinkin heterogeenisiä. Kimball et al. (2013) jatkaa, että projektin suunnitteluvaiheessa tulisi määritellä tarkasti projektin toteutuksessa mukana olevien henkilöiden roolit ja arvioida yrityksen valmiudet tietovarastoprojektin läpiviennille. Teorian mukaan valmiuksista teknisestä näkökulmasta katsottuna mainittakoon lähdedatan laatu (Kimball & Ross 2002; Ponniah 2010), mikä ei kuitenkaan haastatteluissa noussut juuri lainkaan esille. Tietovarastonnin vaatima osaaminen on operatiivisten järjestelmien toteutusta monitahoisempi, joten tarve ulkopuoliselle osaamiselle saattaa tulla kysymykseen. Kannattaa myös tiedostaa, että myös tulevan tietovaraston käyttäjäkunta on kirjavampi kuin tavallisesti.

Vaatimusten määrittely on prosessissa seuraavana. Siinä rakennetaan pohja tulevalle suunnittelulle tunnistamalla tietovaraston liiketoimintavaatimukset. Vaatimusten keräämistä voidaan lähestyä kolmella eri tavalla: olennaisimmat liiketoimintaprosessit tunnistamalla, käyttäjien tietotarpeet keräämällä tai lähdejärjestelmien dataa analysoimalla (List et al. 2002; Golfarelli 2010), joista haastatteluissa mainittiin vain kaksi ensimmäistä. Haastateltavat kuitenkin lisäsivät, että lähestymistavat eivät ole toisiaan poissulkevia vaan niillä voidaan tarvittaessa täydentää toisiaan. Sekä teoriassa että haastatteluissa korostettiin sitä, että liiketoimintavaatimuksia määritettäessä ensisijaista on toimialan ja yrityksen toiminnan tuntemus ja hahmottaminen sekä niissä keskeisimpien käsitteiden hallinta (ks. esimerkiksi Kimball & Ross 2002). Vaatimusten määrittelyssä on tärkeä tiedostaa erilaiset lähestymistavat liiketoimintavaatimusten keräämiseen, sillä niiden määrittely luovat pohjan koko prosessissa suoritettaville toimenpiteille ja määrittelevät tietovarastoinnin onnistumisen.

Suunnittelun ja toteutuksen voidaan tulkita tietovarastojen kehityksessä jakautuvan kahdelle eri radalle (Kimball et al. 2013). Radoilla otetaan kantaa suunnittelun eri aspekteihin, joita ovat teknologinen arkkitehtuuri ja tietoarkkitehtuuri. Teknologioita valittaessa on suunniteltava tietovarastolle tekninen arkkitehtuuri ja valita sitä toteuttamaan soveltuvat sovellukset. Kimball & Ross (2002) määrittelivät ratojen toimenpiteet täysin tietovarastoa rakentavan organisaation hallitsemiksi, mutta haastateltavat totesivat päinvastaisesti olevan monimutkaisempi. Esimerkiksi tietovarastoa toteutettaessa teknologioihin kantaa ottavan radan toiminnassa edetään asiakkaan päätöksen mukaisesti. Se on silti olennainen osa suunnittelua.

Kimball:n ja Ross:in (2002) mukaan tietomallin suunnittelussa luodaan liiketoimintavaatimuksia vastaava käsitemalli, joka toteutetaan tietokannanhallintajärjestelmän kanssa yhteensopivaan muotoon. Haastatteluissa oltiin hyvin samoilla linjoilla, mutta kirjallisuudesta poiketen mainittiin, että myös tiedon mallinnuksesta saatetaan joutua palaamaan edeltävään vaatimustenmäärittelyvaiheeseen. Käsitemallin luonnin jälkeen suunnitellaan ja toteutetaan datan siirtoa, puhdistusta ja muita tarvittavia operaatioita suorittava ETL-prosessi. Sekä kirjallisuudessa että haastatteluissa oltiin yhtä mieltä, että toteutus voidaan suorittaa tähän suunniteltuilla kaupallisilla tai avoimen lähdekoodin työkaluilla sekä itse toteuttamalla. Kimball ja Ross (2002) suosittelivat valmiiden työkalujen hyödyntämistä mahdollisuuksien mukaan, kun taas haastateltavilla ei ollut selkeää preferenssiä asiaan. He mainitsivat valinnan olevan tapauskohtainen. Dataradalle voidaan katsoa lukeutuvaksi myös testaus, jota suoritetaan tiedon mallinnuksen ja ETL-prosessin toteutuksen lomassa sekä niiden valmistuttua.

Suunnittelutyössä huomioitavia seikkoja ovat käsitteellisen mallin kriittisyys toteutuksen kannalta, mikä nousi esiin tutkimuksen molemmissa osioissa. Tietovarastoarkkitehtuurin rakentaminen riippuu toimialasta ja yrityksestä sekä sen tietotarpeista. On tunnistettava vaihtoehtoiset lähestymistavat ja huomattava, että yksi ratkaisu ei välttämättä sovi kaikkiin ympäristöihin. Kimball et al. (2013) mainitsivat, että jos yrityksessä on olemassa olevaa tietovarastoarkkitehtuuria, hankaloittaa se suunnittelun etenemistä. Dimensionaalisuuteen pohjautuvissa ratkaisuissa tietomallin dimensioiden suunnittelu tulee tehdä alusta alkaen yrityksen kokonaisuus huomioiden eli bottom-up -periaatetta noudattaen. Tällöin luodaan niin kutsutut mukautetut dimensiot. Kirjallisuudessa esiintyi myös top-down -lähestysmistapaan pohjautuva suunnittelu esimerkiksi Inmonin (2005) teoksessa. Haastatteluissa sen mainittiin kuitenkin vaativan mittavia panostuksia toteutuksen alkuvaiheessa ja nähtiin siksi ongelmalliseksi. Yhtä mieltä haastatteluissa ja teoriassa oltiin data tallennuksesta. Ponniah (2010) esittää, että data tulee tallentaa faktoihin ja dimensioihin matalimmalla tarkkuustasolla, jotta mahdollistetaan dataan porautuminen ja ad hoc -kyselyt. Muuta suunnittelussa huomioitavaa oli, ettei tietomallia pitäisi muodostaa tietyn raportin tietotarpeiden pohjalta, sillä siitä on vaarana tulla liian yksityiskohtainen yleisiä tietotarpeita silmälläpitäen.

Monet kirjoittajat, kuten Kimball ja Ross (2002) sekä ETL-prosessia toteutettaessa tulee ottaa huomioon sen toteutuksen aikaa vievyys. Toteutuksessa tulee muistaa ylläpidettävyyden parantaminen prosessin mallinnuksella ja metadatan määrittelyllä tietolähteistä, prosessin kulusta ja tietovaraston rakenteesta. Kaikelle tälle tulisi olla yhdenmukainen mallinnusnotaatio.

6.2 Laajennettavuuden, räätälöitävyyden ja uudelleenkäytettävyyden huomioiminen

Alatutkimusongelmaksikin muodostettujen ominaisuuksien toteutumisen voidaan katsoa tapahtuvan kahdella eri tasolla: ohjelmistoarkkitehtuurin suunnittelussa sekä tietomallissa. Koska laajennettavuus, räätälöitävyys ja uudelleenkäytettävyys ovat sisäänrakennettuna tuoterunkoon perustuvassa kehityksessä, on niiden toteutumiseen huomioitavia seikkoja lueteltu jo aikaisemmissa alaluvuissa. Selvyyden vuoksi ne luetellaan vielä tässä osiossa erillään.

6.2.1 Laajennettavuus, räätälöitävyys ja uudelleenkäytettävyys tuoterunkoon perustuvassa kehityksessä

Empirian perusteella muodostetun tulkinnan mukaan laajennettavuus ja räätälöitävyys kulkevat tuoterunkoon perustuvan ohjelmiston kehityksessä rinnakkain. Mahdollisuus näiden ominaisuuksien toteutumiselle saadaan aikaan tuotteistamisen alkuvaiheessa, kun luodaan käsitemallia toteutettavasti ohjelmistosta. Käsitemallin tulee olla riittävän kattava ja geneerinen siten, että koko asiakassegmentin vaatimukset voidaan tyydyttää. Tämä vaatii vankkaa osaamista toimialalta, jolle ohjelmisto ollaan rakentamassa

Pohl et al. (2005) ja Meister et al. (2004) toteavat, että laajennettavuus ja räätälöitävyys voidaan mahdollistaa myös ennakoivalla arkkitehtuurisuunnittelulla sekä muunneltavuuden hallinnan toteuttamisella. Sama havainto tehtiin empiriassa, jossa lisättiin se, että jo arkkitehtuurin suunnittelun ja mallinnuksen suorittaminen takaa laajennettavuuden ja räätälöitävyyden tiettyyn pisteeseen. Lisäksi Breivold et al. (2010) mainitsevat, että ohjelmiston laatuattribuutit huomioivat suunnittelutekniikat edesauttavat näiden tavoitteiden saavuttamisessa. Meister et al. (2004) lisäsivät, että arkkitehtuurityylien ja suunnittelumallien, kuten mikroydinrakenteen tai PAC-mallin, soveltaminen. Empiriaosuudessa malleihin lisättiin kerrosarkkitehtuuriin perustuva sekä modulaarinen arkkitehtuuri. Näistä jälkimmäisenä mainitun katsottiin edesauttavan laajennettavuutta ja räätälöitävyyttä rajoittamalla muutosten vaikutukset vain tiettyihin osaan ohjelmistoa. Haastateltavat nostivat esiin myös plugin-suunnitteluratkaisun käytön, mikä mahdollistaa räätälöityjen komponenttien kytkemisen ohjelmistoon. Yhdessä konfiguroitavan palapelirakenteen avulla asiakkaan prosesseja voidaan räätälöidä asiakaskohtaisilla tai uudelleenkäytettävillä komponenteilla. Rajapintojen suunnittelu näissä tapauksissa on myös kriittistä, sillä liian suppeat rajapinnat heikentävät komponenttien monipuolisuutta.

Breivold et al. (2010) toteavat edelleen, että arkkitehtuurityössä laajennettavuuden mahdollistumista voidaan parantaa jäljitettävyyden mallinnuksella. Siinä ominaisuuksien ja asiakasvaatimusten suhde kuvataan eksplisiittisesti. Myös muunneltavuuden hallinta ja mallinnus helpottavat järjestelmän laajennettavuutta (Svahnberg 2003).

Kun muunneltavuutta vaativat kohdat on määritelty ja muunneltavuutta tukeva arkkitehtuuri ja ohjelmistokomponentit on luotu, uudelleenkäytettävyys on tuoterungoissa sisäänrakennettuna ominaisuutena. Kirjallisuudessa esimerkiksi Pohl et al. (2005) sekä Koskimies ja Mikkonen (2005) kutsuivat näitä kohtia variaatiopisteiksi ja niihin valittavia vaihtoehtoisia ratkaisuja varianteiksi. Lisänä teoriaosuuteen empiriassa nousi esiin versionhallintatyökalujen merkitys uudelleenkäytettävyyden mahdollistajana. Ne ovat olennainen osa käytännön toteuttamista. Myös parametrisoinnin ja konfiguroinnin nähtiin esimerkiksi Oliveira Jr. et al:n (2005) sekä Jacobsen et al:n (1997) mukaan mahdollistavan uudelleenkäytettävyyttä. Empirian perusteella voidaan todeta, että ne edesauttavat lisäksi laajennettavuutta ja uudelleenkäytettävyyttä.

6.2.2 Laajennettavuus, räätälöitävyys ja uudelleenkäytettävyys tietovarastoinnissa

Tietovarastoinnissa laajennettavuus, räätälöitävyys ja uudelleenkäytettävyys toteutuvat hiukan eri tavoin tietovaraston tietomallissa ja ETL-prosessissa, jossa uudelleenkäytettävyys on mahdollista empirian tulkinnan perusteella vain tiettyyn pisteeseen asti. Sekä kirjallisuudesta tehtyjen löydösten (esimerkiksi Kimball et al. 2011) että haastatteluiden perusteella tietovarastoinnissa korostuu sovellusalan tuntemus, kun muodostetaan käsitemallia. Tietomalli muodostaa lähtökohdan koko tietovarastoinnille.

Tietomallin laajennettavuus toteutuu jo dimensionaalisen mallinnuksen käytön myötä, mutta dimensioiden suunnittelulla voidaan parantaa laajennettavuutta olennaisesti. Niin sanottujen mukautettujen dimensioiden muodostaminen mahdollistaa sen, että samoja dimensioita voidaan käyttää eri faktatauluissa, joista kaikkia ei välttämättä ole alkuvaiheessa tunnistettu. Mukautetut dimensiot muodostavat eri datakomeroiden välille väylärakenteen (Kimball & Ross 2002). Inmon (2005) mainitsi, että datan granulariteetti tulee olla mahdollisimman atomisella tasolla. Dimensionaalisen tietomallin tapauksessa tämä pätee sekä dimensioiden että faktojen mallinnuksessa. Siten tauluja voidaan tulevaisuudessa laajentaa eikä alkuvaiheessa tallennettu data rajoita myöhemmin lisättyjä tietolähteitä. Hyvin pitkälti samat havainnot esiintyivät empiriaosuudessa.

Tietovaraston toteuttava tietokannanhallintajärjestelmän ominaisuudet vaikuttavat ratkaisun laajennettavuuteen. Käytettävän tietokantateknologian yhteensopivuus lähdejärjestelmien ja erilaisten raportointisovellusten on oltava laaja (Kimball et al. 2011). Tietokannanhallintajärjestelmän on tuettava lineaarista kasvua ja erityyppisiä indeksointeja, jotka ovat tyypillisiä tietovarastoissa (Inmon 2005; Westerman 2000). Empiriaosuudessa näin tekniset seikat eivät nousseet esille. Empiriassa todettiin kuitenkin, että esimer-

kiksi Microsoftin ja Oraclen tietokannanhallintajärjestelmien, joita kohdeorganisaatiossa käytetään, kanssa edellä mainittujen ominaisuuksien kanssa ei olla havaittu ongelmaa.

Empiriaosuudesta nousi esiin eräs tärkeä seikka liittyen lähdejärjestelmien tietorakenteiden muutoksiin. Nimittäin lähdejärjestelmien tietorakenteiden muutokset heijastuvat suoraan ETL-prosessiin ja tietovaraston tietomalliin. Kehitettäessä lähdejärjestelmiä ja tietovarastoa rinnakkain, muutosten riippuvuussuhteet ja kehityksen tuleminen työläämmäksi on huomioitava resursseja allokoidessa. Räätälöitävyyttä ja samalla ylläpidettävyyttä voidaan kuitenkin helpottaa dokumentoinnin keinoin. Metadatan hallinta ja mappausten dokumentointi auttaa hahmottamaan tietolähteiden ja tietovaraston tietomallin välisiä suhteita (Vetterli et al. 2000). Myös automaattinen testaus tietolähteiden rakenteen muuttuessa palvelee tätä tarkoitusta.

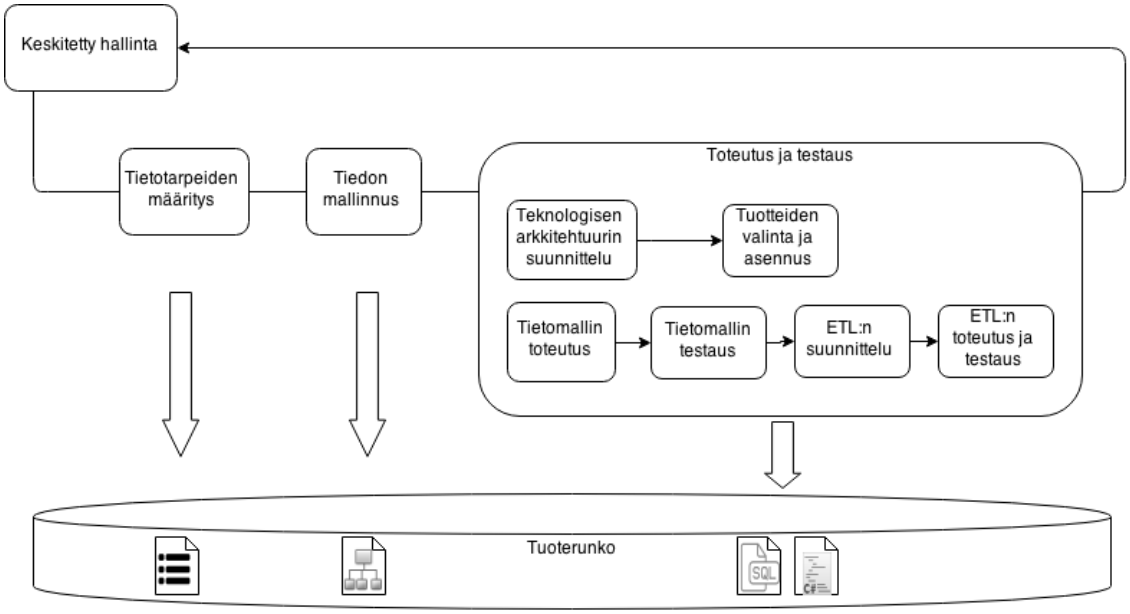
Toinen empiriassa esiin noussut seikka, jota ei teoriassa juurikaan esiintynyt, oli tietovaraston eri komponenttien uudelleenkäytettävyys. Tietomallin uudelleenkäytettävyys nähtiin mahdolliseksi, jos tietomalli on riittävän geneerinen ja kuvaa toimialan yleisimpiä prosesseja. Tässä tapauksessa tietovaraston liiketoimintavaatimusten kerääminen on tavoitevetoista, joka kyllä esiintyi esimerkiksi Malinowskin ja Zimányin (2008) teoksessa. Empiriaosuuden haastatteluissa mainittiin, että pelkät tietotarpeet tai lähdedata edellä etenevä liiketoimintavaatimusten kerääminen ja mallinnus johtavat asiakaskohtaiseen ratkaisuun, mikä heikentää merkittävästi uudelleenkäytettävyyttä. Muodostettavan käsitemallin on oltava järjestelmäriippumaton eli käytettävä käsitteistö ei saa pohjautua tietyn teknologian tai lähdejärjestelmien rakenteisiin. Tässä palataan jälleen osaamiseen mallinnettavasti sovellusalasta, jota ei suoranaisesti korostettu kirjallisuudessa. Esimerkiksi Kimball et al:n (2011) lähestymistapa tietotarpeiden keräämiseen on kuitenkin tulkittavissa pyrkivän sen yrityksen toiminnan tuntemiseen.

Empirian perusteella uudelleenkäytettävyyden voidaan katsoa olevan tietovaraston ETL-prosessissa mahdollista vain osittain. Tosin tämäkin vaatii sen, että tietovaraston käsitemallin pitää olla vakio kaikissa tapauksissa. Koska yritysten lähdejärjestelmät ovat harvoin identtiset, on datan poiminta lähteistä aina yksilöity prosessi (Ponniah 2010; Kimball et al. 2010). Empiriassa esiintyi ehdotus ratkaisusta, jossa tietolähteiden ja tietovaraston väliin rakennetaan eräänlaisia varjotauluja. Niiden tietomalli on vakio ja sen sisältämä operatiivinen data ladataan tietovarastoon uudelleenkäytettävällä ETL-prosessilla. Varjotaulut kuuluvat osaksi tuoterunkoa, mutta datan lataus varsinaisista lähdejärjestelmistä varjotauluihin on poikkeuksetta toteutettava yrityskohtaisesti.

6.3 Tuoterunkoon perustuvan tietovaraston kehittäminen

Tässä alaluvussa esitellään tuoterunkoon perustuvan tietovarastoratkaisun kehitysprosessi ja huomioon otettavat seikat, kun ratkaisu pyritään saamaan laajennettavaksi, räätälöitäväksi ja uudelleenkäytettäväksi. Tietovaraston tuotteistamisen voidaan kirjalli-

suuskatsauksen ja empirian perusteella tulkita koostuvan neljästä erillisestä, mutta toisiinsa sidoksissa olevasta vaiheesta. Kokonaisuutta kuvaa alla oleva kuva 6.



Kuva 6. *Tietovaraston tuotteistamisprosessi*

Tietovaraston laajennettavuutta, räätälöitävyyttä ja uudelleenkäytettävyyttä edesauttavat seikat toteutuvat eri vaiheissa kehityksen elinkaaren aikana. Nämä seikat muiden tärkeiden huomioiden ohella on esitetty taulukossa 2, jossa ne on listattu kunkin vaiheen alle kuuluvan toimenpiteiden yhteyteen. Laajennettavuus (L), räätälöitävyys (R) ja uudelleenkäytettävyys (U) on merkitty taulukkoon käyttäen kirjanlyhennelmää.

Taulukko 2. *Tuotteistamisen vaiheet, niiden toimenpiteet ja huomioon otettavat seikat*

Prosessin vaihe	**Toimenpiteet**	**Huomioitavat seikat**
Keskitetty hallinta	Tiekartan määrittely	• Rajauksen määrittely (L)
	Muunneltavuuden hallinta	• Muunneltavuuden mallinnus (L, R, U)
Tietotarpeiden määritys	Vaatimusten tunnistaminen	• Sovellusalueen tai toimialan ja sen käsitteiden tuntemus (L, U) • Prosessin tarkastelu (L, U) • Tulevaisuuteen suuntautuva näkemys
	Synteesin tekeminen	• Tarpeiden tärkeyden tunnistaminen • Resurssien riittävyys

Tiedon mallinnus	Käsitemallin suunnittelu	• Mallinnustavan valinta • Dimensioiden suunnittelu (L) • Mallin yleiskäyttöisyys, yleisyys (U)
Tietovaraston suunnittelu ja toteutus	Teknologisen ympäristön suunnittelu	• Yleisimmin käytössä olevien teknologioiden valinta (U)
	Tietomallin toteutus ja testaus	• Tietokannanhallintajärjestelmän toteutusteknologia • Kyselyiden suorituskyky • Datan tarkkuustason valinta (L)
	ETL:n arkkitehtuurin toteutus ja testaus	• Teknologiariippumaton ETL-prosessin toteutus (L, R, U) • Suunnittelumallit ja arkkitehtuuriratkaisut (L, R) • Lähdejärjestelmien riippuvuussuhteet ETL-posessiin ja tietomalliin (R) • Konfiguroitavuus (L, R) • Komponenttien rajapintojen suunnittelu (L, U) • Testattavuus, prosessin suorituskyky, virhesietoisuus • Historiadatan saatavuus muutostilanteissa • Dimensioiden päivitys

Tuotteistamisprosessiin kuvan 6 mukaan kuuluu tietotarpeiden määritys, tiedon mallinnus sekä suunnittelu ja toteutus. Suunnittelu ja toteutus jakautuvat tietovaraston kahdeksi rinnakkaiseksi kehitysradakseen, joilla otetaan kantaa eri näkökulmiin rakennettavasta tietovarastoratkaisusta. Tuotteistamisprosessia tukee keskitetyn hallinnan vaihe, jossa koordinoidaan tuoterungon kehitystä sen ominaisuuksien osalta. Lisäksi kaikissa kehityksen vaiheissa harjoitetaan muunneltavuuden hallintaa eli hallinnoidaan tuoterungon mahdollistamaa laajennettavuutta, kustomoitavuutta ja konfiguroitavuutta.

Tuotteistamisprosessin vaiheiden seurauksena syntyy tuotoksia, joita kutsutaan tuoterungon artefakteiksi. Se, missä muodossa tai tarkkuustasolla artefaktit sisällytetään tuoterunkoon, riippuu toteuttavasta organisaatiosta. Tuoterunkoa havainnollistaa kuvassa 6 sen alaosiossa oleva kartion muotoinen kuvio.

Tietotarpeiden määrityksessä tietovaraston liiketoimintavaatimuksia kerätään tarkastelemalla organisaation olennaisimpia prosesseja. Kun tietomalli muodostetaan prosessin

tarkastelun perusteella, tietovaraston käytön aikana syntyneitä tietotarpeita varten tietomalliin ei tarvita muutoksia. Samalla vaatimukset ovat suurella todennäköisyydellä samat toisen saman alan toimijan tapauksessa. Vaiheen tuloksena on lista olennaisimmista liiketoimintavaatimuksista tietovarastolle.

Seuraava vaihe tietovaraston tuotteistamisessa on tiedon mallinnus. Siinä luodaan liiketoimintavaatimuksiin pohjautuva käsitemalli dimensionaalisessa muodossa. Normalisoiduissa tietomalleissa on vaatimuksena heti koko tietomallin kehittäminen. Tapauksessamme tämä ei ole mahdollista, joten ainoa vaihtoehto mallinnustyön tekemiseen on dimensionaalinen mallinnus. Oikea dimensioiden suunnittelu takaa laajennettavuuden muutostarpeiden ilmentyessä. Mallin dimensiot on määritettävä tarkkaan tarkastellen yritystä kokonaisuutena. Samojen dimensioiden on sovelluttava kuvaamaan mitä tahansa mitattavan suureen kontekstiin liittyvää tietoa. Mallin käsitteet on oltava yleiskäyttöisiä ja ympäristöstä riippumattomia.

Viimeinen vaihe tuotteistamisessa on tietovaraston suunnittelu ja toteutus. Vaihe jakautuu kahteen osaan. Ylemmällä radalla tehdään päätöksiä teknologioihin, kuten tietokannanhallintajärjestelmään, ETL-työkaluihin tai palvelinympäristöön. Tätä kutsutaan nimellä teknologiarata.Sen vaiheita ovat teknologisen arkkitehtuurin suunnittelu ja tuotteiden valinta ja asennus. Ensimmäisessä vaiheessa kerätään teknologiselle arkkitehtuurille vaatimukset, dokumentoidaan ne, suunnitellaan arkkitehtuurin toteutusta ja katselmoidaan aikaansaatuja tuotoksia. Seuraavaksi valitaan arkkitehtuuria toteuttamaan määritellyt tuotteet. Arkkitehtuuri toimii listana, johon valintapäätökset perustuvat. Tuotekategoriat ovat palvelinlaitteisto, tietokannanhallintajärjestelmä ja ETL-toteutusteknologia. Tuotteet asennetaan ja ympäristö toimii lähtökohtana varsinaiselle tietovarastolle.

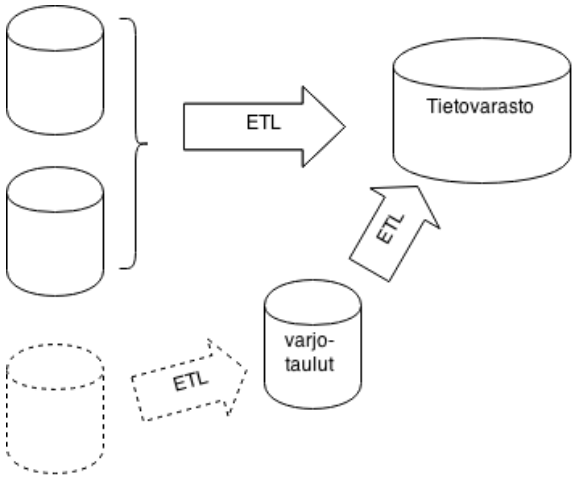
Teknologioita valittaessa on muistettava, että ympäristön on oltava toteutettavissa useaan eri ympäristöön, jolloin tietyn teknologian asettamat rajoitukset on huomioitava valinnassa. ETL-prosessin tapauksessa tulee valita esimerkiksi tietokannanhallintajärjestelmästä riippumaton avoimen lähdekoodin ratkaisu tai toteuttaa työkalu itse. Erityisesti tietokannanhallintajärjestelmän valinnassa vaadittavia kriteereitä ovat laaja yhteensopivuus eri lähteiden sekä raportointi- ja analyysisovellusten kanssa, mahdollisuus lineaariseen kasvuun, monipuolisille indekseille ja suorituskyvyn monitoroinnille.

Alemmalla radalla luodaan liiketoimintavaatimuksia vastaava käsitemalli dimensionaalisessa muodossa eli muodostetaan tietokannan fyysinen rakenne. Dimensiot on suunniteltava siten, että ne ottavat yrityksen kokonaisuuden huomioon ja dimensiot muodostavat eräänlaisen väylän faktataulujen välille. Data dimensioissa ja faktatauluissa suunnitellaan alhaisimmalle tarkkuustasolle, jotta alkuvaiheessa luodut dimensiot ja faktat eivät muodostu rajoittaviksi tekijöiksi tietomallia laajennettaessa. Tällöin myös ennalta tuntemattomiin tietotarpeisiin vastaavia kyselyitä on mahdollista muodostaa. Käsitemallin tulee olla lisäksi käsitteistöltään yleiskäyttöinen eikä se saa pohjautua tietyn järjes-

telmän rakenteeseen tai nimeämiskäytäntöihin. Seuraavaksi käsitemallia jalostetaan pää- ja vierasavaimilla ja muunnetaan valitun tietokannanhallintajärjestelmän kanssa yhteensopivaan muotoon. Seuraavaksi määritellään lähdedatan ja tietovaraston tietomallin väliset mappaukset. Lopuksi testataan yhdessä käyttäjien kanssa tietomallin oikeellisuus.

Kun käsitemalli on muodostettu, ryhdytään suunnittelemaan ETL-prosessin toteutusta. Suunnittelu pohjautuu edellisessä vaiheessa määritettyihin mappauksiin lähdedatan ja tietovaraston välillä. Toteuttamalla prosessi itse ei sitouduta mihinkään teknologiaan ja toteutus on uudelleenkäytettävä. Toteutus tarkoittaa käytännössä prosessin ohjelmoimista esimerkiksi käyttämällä jotakin ohjelmistoviitekehystä. Toteutettaessa tulee hyödyntää arkkitehtuurityylien ja suunnittelumallien, kuten kerrosarkkitehtuuria ja plugin-rakennetta, jotta prosessia voidaan räätälöidä tarvittaessa ja laajennetaa uusilla lähteillä. Uusien lähteiden mukaan tuominen tarkoittaa käytännössä uuden ETL-prosessin toteutusta, jonka liittäminen mahdollistetaan konfiguroitavuudella. Lopuksi ETL-prosessi testataan.

Kun kyseessä on tietovarasto, jossa kaikkea lähdedataa ei vielä tarkasti tiedetä, mutta tunnetaan yleiset vaatimukset erillisistä järjestelmistä tarvittavalle datalle, voidaan muodostaa niin sanottuja varjotauluja. Varjotaulut sijoittuvat oletettujen lähteiden ja tietovaraston välille. ETL-prosessi näistä varjotauluista voidaan näin muodostaa vakioksi, mutta siirto lähteistä varjotauluihin täytyy aina tehdä asiakaskohtaisena räätälöintinä. Rakenne tehdään konfiguroitavaksi niin, että se voidaan tarpeen tullessa kytkeä päälle jälkikäteen. Kuva 7 havainnollistaa tilannetta. Siinä katkoviivalla on havainnollistettu tuotteistamisvaiheessa tuntemattomia osia. Yhtenäisellä viivalla merkityt kuviot voidaan muodostaa etukäteen ja ovat alusta alkaen osa tuoterunkoa.



Kuva 7. *Periaate varjotaulujen muodostamisessa*

Toteutettaessa edellisen kaltaista rakennetta kannattaa kuitenkin punnita etukäteisen toteutuksen hyötyjä, sillä tietyt osat prosessista joudutaan joka tapauksessa tekemään räätälöitynä.

Toteutus- ja testausvaiheen artefakteja ovat muun muassa ETL-prosessin arkkitehtuuri, tietokannanhallintajärjestelmäsidonnaiset tietomallin toteutukset, laajennettavuutta helpottavat aputyökalut. Näitä artefakteja 6 on havainnollistettu kartion muotoisella kuviolla.

Olennaisena osana kaikkia vaiheita on muunneltavuuden hallinta. Siinä tuoterungon variaatiopisteet tunnistetaan, määritellään ja toteutetaan. Hallinnan tuloksena muodostetaan muunneltavuusmalli, jossa edellä mainittujen toimenpiteiden tulokset dokumentoidaan graafisesti sekä sanallisesti. Mallinnus toimii kommunikaatiovälineenä sekä asiakkaan että asiakaskohtaisia installaatioita toteuttavien kehittäjien suuntaan.

7. YHTEENVETO

Tässä kappaleessa tehdään ensin keskeisimpiä tuloksia ja johtopäätöksiä suoritetusta tutkimuksesta sekä arvioidaan sen luotettavuutta. Lopuksi esitellään tutkimuksen suorittamisen aikana syntyneitä jatkotutkimusehdotuksia.

7.1 Johtopäätökset

Tässä diplomityössä tutkittiin tuoterunkoon perustuvan tietovarastoratkaisun kehittämistä. Tätä varten muodostettiin päätutkimuskysymys:

- Miten tuotteistetaan yhtenäiseen tuoterunkoon perustuva tietovarastoratkaisu?

Päätutkimusongelmaan haettiin vastausta etsimällä vastauksia kahden päätutkimuskysymykseen liittyvän teeman avulla. Teemoja olivat ohjelmistotuoterunkoon perustuva kehitys eli tuotteistaminen ja tietovaraston kehitys sekä niihin liittyvät olennaisimmat huomioitavat seikat. Näitä varten muodostettiin seuraavat alatutkimuskysymykset:

- Mitä vaiheita kuuluu ohjelmiston tuotteistamiseen?
- Millainen on tietovaraston kehittämisprosessi?

Viimeinen alatutkimuskysymys muotoutui seuraavanlaiseksi:

- Mitä asioita kehityksessä tulee huomioida, että ratkaisusta tulee laajennettava, räätälöitävä ja uudelleenkäytettävä?

Tässä molempiin teemoihin liittyen etsittiin vastauksia siihen, miten ratkaisusta saadaan yleinen siten, että se on toteutettavissa asiakkaalta toiselle tehokkaasti ja niin, että rakenteeseen voidaan tehdä jälkikäteen muutoksia.

Tutkimus eteni siten, että kaikkiin alakysymyksiin etsittiin ensin vastauksia aihepiirien kirjallisuudesta, jonka perusteella muodostettiin tarkistuslista. Tarkistuslista toimi pohjana empiirisen osuuden suorittamisessa ja se esiteltiin yhtenä teemana haastatteluissa.

Tutkimuksen toisessa osiossa suoritetussa empiirisessä osuudessa etsittiin täydentäviä vastauksia ja hyväksyntää tutkimuksen alatutkimusongelmiin ja saatuja tuloksia analysoitiin yhdessä teorian keskeisimpien havaintojen kanssa. Kirjallisuuden havaintojen ja haastatteluiden keskeisimpien löydösten perusteella vastattiin ensin alatutkimuskysymyksiin alaluvuissa 6.1 ja 6.2. Viimeisenä alatutkimusongelmien tuloksia yhdistelemällä ja tulkitsemalla vastattiin päätutkimuskysymykseen. Vastaukseksi muodostettiin pro-

sessimalli, sen vaiheet ja keskeisimmät huomioonotettavat tekijät, jotka on esitetty alaluvussa 6.3. Tekijät liittyivät yleisesti prosessin suorittamiseen sekä laajennettavuuden, räätälöitävyyden ja uudelleenkäytettävyyden toteutumiseen.

Tuoterunkoon perustuvan ohjelmiston tuotteistaminen nähtiin koostuvan vaatimusten määrittelystä, alustan suunnittelusta, toteutuksesta ja testauksesta. Prosessiin omaksi kokonaisuudekseen katsottiin kuuluvan tuotteenhallinta, jossa otetaan muun muassa kantaa tuoterungon rajaukseen ja tulevaisuuden kehityssuuntiin. Merkittäväksi toiminnaksi prosessin vaiheiden ohella havaittiin olevan muunneltavuuden hallinta ja mallinnus, joissa tuoterungon tarjoamaa varianssia hallitaan systemaattisesti ja mallinnetaan asianmukaisella tavalla muun muassa kommunikaation ja ylläpidettävyyden helpottamiseksi.

Toisen alaongelman kohteena olleen tietovaraston kehittämisprosessin katsottiin koostuvan seuraavista vaiheista: projektin suunnittelu, vaatimusten määrittely sekä suunnittelu ja toteutus. Näistä suunnittelun ja toteutuksen todettiin jakautuvan kahdeksi rinnakkaiseksi prosessikseen, joista toinen käsittelee teknologista ja toinen tietoarkkitehtuuria. Keskeisimmiksi huomioonotettaviksi seikoiksi kehitysprosessissa nähtiin muun muassa käsitemallin luominen, datan tarkkuustason valinta ja dimensioiden suunnittelu.

Viimeiseen alaongelmaan löydettiin vastauksia sekä tietovaraston kehittämisen että tuoterungon kehittämisen aihealueista. Huomioitavia seikkoja löydettiin useita, mutta keskeisimmät niistä olivat tuntemus toimialasta, jolle ratkaisua kehitetään, kattavan ja geneerisen tietomalli luominen sekä suunnittelumallien soveltaminen ohjelmistoarkkitehtuurissa.

Itse pääkysymykseen vastattiin alakysymysten teemoja yhdistämällä ja tekemällä tulkintoja. Tämän perusteella luotiin prosessimalli ja lista huomioitavista seikoista kussakin prosessin vaiheessa. Malli syntyi varsin korkean tason kuvaukseksi ja sisältää jonkin verran liikkumavaraa tuotteistamisen toteuttamiseksi. Mallissa tuoterunkoon perustuvan tietovarastoratkaisun kehittäminen pääteltiin koostuvan tietotarpeiden määrityksestä, tiedon mallinnuksesta sekä toteutus- ja testausvaiheesta. Toteutus- ja testausvaihe jakaantui kahdeksi erilliseksi aliprosessikseen. Omana kokonaisuutenaan nähtiin keskitetty hallinta ja prosessin vaiheiden läpi kulkeva muunneltavuuden hallinta ja mallinnus. Merkittävimpiä huomioonotettavia seikkoja koko prosessin kannalta tulkittiin olevan sovellusalueen osaaminen, tiedon mallinnus, teknologiariippumattomuus ja ohjelmistoarkkitehtuurinsuunnitteluratkaisut ja konfiguroinnin mahdollistaminen. Pääkysymykseen saadut vastaukset on esitelty kattavammin tutkimuksen pohdintaosiossa.

7.2 Tutkimuksen arviointi

Laadullisessa tutkimuksessa yleisesti ja erityisesti tapaustutkimuksen tapauksessa on kritisoitu tutkimuksen tuottaman tiedon luotettavuutta. (Tuomi & Sarajärvi 2009; Yin

2003) Tutkimuksen luotettavuuden määrittämiseen on käytetty useimmiten validiteettia ja reliabiliteettia, vaikkakin niiden käyttöön liittyy paljon ristiriitaisia kommentteja. (ks. esim. Tuomi & Sarajärvi 2009)

Saunders et al. (2011, s. 156) kuvaavat reliabilitettia ominaisuudeksi, jonka mukaan tutkimuksen datan keräystekniikat tai analysointimenetelmät tuottavat johdonmukaisia löydöksiä. Tuomi ja Sarajärvi (2009, s. 136) puhuvat puolestaan tutkimustulosten toistettavuudesta. Tässä tutkimuksessa reliabiliteetin toteutumiseen tiettyä varmuutta tuo havainto, että kirjallisuudesta ja empiriasta nostetut ehdotukset tutkimuskysymyksiin poikivat hyvin samantyylisiä tuloksia. Toisaalta kuten kvalitatiivisessa tutkimuksessa yleensä todetaan, itse tutkijalla on subjektiivisuutensa takia vaikutus tulosten muodostamisessa. Näin ollen täyttä varmuutta reliabiliteetin toteutumisesta ei voi tämänkään tutkimuksen tapauksessa antaa.

Validiteetti koskee taas sitä, ovatko tutkimuksen tulokset todellakin sitä mitä niiden väitetään olevan. Eli voidaanko havaintojen ja tulosten välille muodostaa kausaalinen suhde. (Saunders et al. 2011, s. 157) Tuomi ja Sarajärvi (2009, s. 136) lähestyvät validiteettikysymystä hiukan toisella tavalla määrittelemällä sen kuvaavan sitä, onko tutkimuksessa tutkittu sitä, mitä siinä on luvattu. Tutkimuskysymyksiä verratessa pohdintaosiossa tehtyihin päätelmien kanssa on todettavissa, että tulokset vastaavat hyvin asetettujen tutkimuskysymyksiä. Sitä, voidaanko havaintojen ja tulosten välille kausaliteettia, tulee tarkastella systemaattisemmin ulkopuolisesta näkökulmasta.

Tutkimuksen luotettavuutta voidaan pyrkiä parantamaan esimerkiksi käyttämällä triangulaatiota, jolla tarkoitetaan moninäkökulmaisuutta hyödyntämällä useita eri lähteitä ja menetelmiä. (Tuomi & Sarajärvi 2009; Hirsjärvi et al. 2009) Yin (2003) mainitsee myös tutkimuksen luotettavuutta parantavaksi toimenpiteeksi tutkimustietokannan ylläpitämisen. Tässä tutkimuksessa on ollut tarkoituksena tarjota monista tutkittavista asioista vaihtoehtoisia näkökulmia sekä käyttää tutkimuskysymyksiin vastaamiseen kahta vaihtoehtoista aineistonkeruumenetelmää: dokumentoitua tietoa sekä haastatteluita. Haastatteluiden edetessä alkoi myös valjeta, että teoria ja haastateltavien kuvaama käytäntö ovat melko hyvin linjassa keskenään aivan pienimpiä yksityiskohtia lukuun ottamatta.

Luotettavuutta on pyritty parantamaan myös siten, että empirian tulokset on raportoitu erillään teoriasta ja tutkijan pohdinnasta. Haastattelut on taltioitu kokonaisuudessaan sekä teoriaosuudessa käytetyt kirjallisuuslähteet on merkitty mahdollisimman tarkasti, joten esimerkiksi ulkopuolisen tarkastelijan on mahdollista palata takaisin luotujen ajatusketjujen alkulähteelle.

Yin (2003) mainitsee myös, että konkreettisten analyysitekniikoiden käyttö kasvattaa tutkimuksen luotettavuutta erityisesti validiteetin näkökulmasta. Koska haastatteluaineisto ei ollut kovin strukturoitua käytetyn haastattelumuodon johdosta, ei mitään ana-

lyysitekniikkaa noudatettu orjallisesti vaan aineiston tulkinta perustui enemmän intuitioon.

7.3 Jatkotutkimusehdotukset

Koska tutkimuksen aihealue oli varsin laaja, heräsi tutkimuksen suorittamisen aikana tämän tutkimuksen kanssa samansuuntaisia ja toimeksiannon tehneelle organisaatiolle mahdollisesti hyödyllisiä tutkimusaiheita. Tutkimuksen rajauksen säilyttämiseksi tämän tyyppiset asiat tuli sivuuttaa ja todeta kuuluvaksi tutkimuksen ulkopuolelle. Aiheita pohditaan tässä lyhyesti.

Ensinnäkin muunneltavuuden mallinnus on tuoterunkoon perustuvassa kehityksessä melko keskeisessä roolissa. Kohdeorganisaatiossa muunneltavuuden mallinnukseen ei ole kiinnitetty systemaattisesti huomiota, sillä muunneltavuutta sisältävät kohdat varastonhallintajärjestelmän tuoterungossa havaitsee vain toteutusta tarkastelemalla. Graafinen esitys muunneltavuudesta voisi parantaa ylläpidettävyyttä ja auttaisi hahmottamaan kompleksista kokonaisuutta ja vaihtoehtoisia installaatioita. Tarve korostuu entisestään variaatiopisteiden lisääntyessä.

Loogisena jatkumona tälle tutkimukselle toimisi tuoterunkoon perustuvan ohjelmistokehityksen toinen alaprosessi eli tuotekehitys ja sen problematiikka. Tehokkaan tuotekehityksen toteutuminen ja prosessissa huomioitavat seikat erityisesti tietovarastoinnin tapauksessa olisivat oivia jatkotutkimuskohteita.

Tietovarastojen kehittämisen puolella esiin nousi myös kysymyksiä ETL-prosessin ja tietovaraston testattavuutta parantavista tekniikoista. Tällä hetkellä testattavuus tietovarastototeutuksissa on satunnaisotosten ja vahvan luottamuksen varassa. Selvitystä tarvittaisiin mahdollisista lähestymistavoista automatisoida ja varmistaa ETL-prosessin toimivuus. Jatkotutkimusehdotuksena tietovarastoinnissa voidaan mainita metadatan hallinnan harjoittaminen ja sen tuomat ylläpidettävyyden hyödyt, joita kirjallisuudessa painotettiin voimakkaasti.

LÄHTEET

Anttila, P. (1998). Tutkimisen taito ja tiedon hankinta. Taito-, taide- ja muotoilualojen tutkimuksen työvälineet. Akatiimi.

Ariyachandra, T., & Watson, H. (2010). Key organizational factors in data warehouse architecture selection. Decision Support Systems, Vol. 49(2), 200-212.

Ariyachandra, T., & Watson, H. J. (2006). Which Data Warehouse Architecture Is Most Successful?. Business Intelligence Journal, Vol. 11(1), 4.

Atkinson, C., Bayer, J., & Muthig, D. (2000). Component-based product line development: the KobrA approach. In Software Product Lines s. 289-309.

Beck, K., & Fowler, M. (2001). Planning extreme programming. 139 s.

Berger, S., & Schrefl, M. (2008). From federated databases to a federated data warehouse system. Hawaii International Conference on System Sciences, Proceedings of the 41st Annual. s. 394-394.

Boehm, B. (2006). A view of 20th and 21st century software engineering. Proceedings of the 28th international conference on Software engineering. s. 12-29.

Bosch, J. (2000). Design and use of software architectures: adopting and evolving a product-line approach. Pearson Education. 354 s.

Bosch, J. (2002). Maturity and evolution in software product lines: Approaches, artefacts and organization. Software Product Lines, s. 257-271.

Bosch, J., Florijn, G., Greefhorst, D., Kuusela, J., Obbink, J. H., & Pohl, K. (2002). Variability issues in software product lines. In Software Product-Family Engineering, s. 13-21.

Breivold, H. P., Crnkovic, I., & Larsson, M. (2012). A systematic review of software architecture evolution research. Information and Software Technology, Vol. 54(1), s. 16-40.

Buhne, S., Lauenroth, K., & Pohl, K. (2005). Modelling requirements variability across product lines. Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on, s. 41-50.

Chastek, G., Donohoe, P., & McGregor, J. D. (2004). A study of product production in software product lines (No. CMU/SEI-2004-TN-012). CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST.

Chaudhuri, S., & Dayal, U. (1997). An overview of data warehousing and OLAP technology. ACM Sigmod record, Vol. 26(1), s. 65-74.

Chen, L., Ali Babar, M., & Ali, N. (2009). Variability management in software product lines: a systematic review. Proceedings of the 13th International Software Product Line Conference, s. 81-90.

Daintith, J., & Wright, E. (2008). A dictionary of computing. Oxford University Press, Inc.

Deelstra, S., Sinnema, M., & Bosch, J. (2005). Product derivation in software product families: a case study. Journal of Systems and Software, Vol. 74(2), s.173-194.

DeLone, W. H., & McLean, E. R. (1992). Information systems success: the quest for the dependent variable. Information systems research, Vol. 3(1), s. 60-95.

Farquhar, J. D. (2012). Case study research for busines. SAGE. 134 s.

Flyvbjerg, B. (2006). Five misunderstandings about case-study research. Qualitative inquiry, Vol. 12(2), s. 219-245.

Gardner, S. R. (1998). Building the data warehouse. Communications of the ACM, Vol. 41(9), s. 52-60.

Garlan, D., Allen, R., & Ockerbloom, J. (1995). Architectural mismatch or why it's hard to build systems out of existing parts. Software Engineering, 1995. ICSE 1995. 17th International Conference on, s. 179-179.

Geyer, L., & Becker, M. (2002). On the influence of variabilities on the application-engineering process of a product family. Software Product Lines, s. 1-14.

Giorgini, P., Rizzi, S., & Garzetti, M. (2005). Goal-oriented requirement analysis for data warehouse design. Proceedings of the 8th ACM international workshop on Data warehousing and OLAP, s. 47-56.

Golfarelli, M. (2010). From User Requirements to Conceptual Design in Data Warehouse Design. Data Warehousing Design and Advanced Engineering Applications: Methods for Complex Construction, s. 1-16.

Golfarelli, M., & Rizzi, S. (1998). A methodological framework for data warehouse design. Proceedings of the 1st ACM international workshop on Data warehousing and OLAP, s. 3-9.

van Gurp, J., Bosch, J., & Svahnberg, M. (2001). On the notion of variability in software product lines. Software Architecture, 2001. Proceedings. Working IEEE/IFIP Conference on, s. 45-54.

Halmans, G., Pohl, K.., & Sikora, E. (2008). Documenting application-specific adaptations in software product line engineering. Advanced Information Systems Engineering, s. 109-123.

Halpin, Terry. (2008). Information modeling and relational databases, second edition. Saatavissa: http://common.books24x7.com/toc.aspx?bookid=32332 (viitattu 1.12.2014).

Helferich, A., Schmid, K., & Herzwurm, G. (2006). Product management for software product lines: an unsolved problem?. Communications of the ACM, Vol. 49(12), s. 66-67.

Hirsjärvi, S., & Hurme, H. (2011). Tutkimushaastattelu. Gaudeamus Helsinki University Pres. 213 s.

Hirsjärvi, S., Remes, P., & Sajavaara, P. (2007). Tutki ja kirjoita. Tammi. 447 s.

Holl, G., Grünbacher, P., & Rabiser, R. (2012). A systematic review and an expert survey on capabilities supporting multi product lines. Information and Software Technology, Vol. 54(8), s. 828-852.

Imhoff, C., Galemmo, N., & Geiger, J. G. (2003). Mastering data warehouse design: relational and dimensional techniques. John Wiley & Sons.

Inmon, W. H. (2005). Building the Data Warehouse, Fourth Edition. John Wiley & Sons, Books24x7. Saatavissa: http://common.books24x7.com/toc.aspx?bookid=12458 (viitattu 21.10.2014)

Inmon, W. H., Strauss, D., & Neushloss, G. (2010). DW 2.0: The Architecture for the Next Generation of Data Warehousing: The Architecture for the Next Generation of Data Warehousing. Morgan Kaufmann. 371 s.

Jacobson, I., Griss, M., & Jonsson, P. (1997). Software reuse: architecture process and organization for business success. New York: ACM Press. 286 s.

Kimball, R. & Ross, M. (2002). The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, Second Edition. John Wiley & Sons, Books24x7. Saatavissa: http://common.books24x7.com/toc.aspx?bookid=6694 (viitattu 21.10.2014)

Kimball, R., Ross, M., & Thornthwaite, W. (2011). Data Warehouse Lifecycle Toolkit (2nd Edition). Hoboken, NJ, USA: John Wiley & Sons. 674 s.

Koskimies, K., & Mikkonen, T. (2005). Ohjelmistoarkkitehtuurit. Talentum. 250 s.

Krueger, C. W. (2002). Variation management for software production lines. Software Product Lines, s. 37-48.

van der Linden, F. (2002). Software product families in Europe: the Esaps & Cafe projects. IEEE software, Vol. 19(4), s. 41-49.

van der Linden, F. J., Schmid, K., & Rommes, E. (2007). Software product lines in action. Springer-Verlag Berlin Heidelberg. 333 s.

List, B., Bruckner, R. M., Machaczek, K., & Schiefer, J. (2002). A comparison of data warehouse development methodologies case study of the process warehouse. Database and Expert Systems Applications, s. 203-215.

Lukka, Kari. (2001). Konstruktiivinen tutkimusote [WWW]. Saatavissa: www.metodix.com (viitattu 11.10.2014).

Maes, R. E., Rijsenbrij, D., Truijens, O., & Goedvolk, H. (2000). Redefining business-IT alignment through a unified framework.

Maier, M. W., Emery, D., & Hilliard, R. (2001). Software architecture: Introducing IEEE standard 1471. Computer, Vol. 34(4), s. 107-109.

Malinowski, E., & Zimányi, E. (2008). Advanced data warehouse design: from conventional to spatial and temporal applications. Springer Science & Business Media. 435 s.

Meister, J., Reussner, R., & Rohde, M. (2004). Applying patterns to develop a product line architecture for statistical analysis software. Software Architecture, 2004. WICSA 2004. Proceedings. Fourth Working IEEE/IFIP Conference on, s. 291-294.

von der Maßen, T., & Lichter, H. (2004). Requiline: A requirements engineering tool for software product lines. Software Product-Family Engineering. s. 168-180.

Merton, R. K. & Kendall, P. L. (1946). The focused interview. American Journal of Sociology Vol. Vol. 51 (6), s. 514–557

Metzger, A., & Pohl, K. (2014). Software product line engineering and variability management: achievements and challenges. Proceedings of the on Future of Software Engineering, s. 70-84.

Myllymäki, T. 2002. Variability management in software product lines. Tampere, Tampere University of Technology. 46 s.

Neilimo, K., & Näsi, J. (1980). Nomoteettinen tutkimusote ja suomalainen yrityksen taloustiede: tutkimus positivismin soveltamisesta. Tampereen yliopisto. 82 s.

Northrop, L., Clements, P., Bachmann, F., Bergey, J., Chastek, G., Cohen, S., Donohoe, P., Jones, L., Krut, R. & Little, R. (2007). A framework for software product line practice, version 5.0. Software Engineering Institute. Saatavissa: http://www.sei.cmu.edu/productlines/index.html (viitattu 21.10.2014).

de Oliveira Junior, E. A., Gimenes, I., Huzita, E. H. M., & Maldonado, J. C. (2005). A variability management process for software product lines. Proceedings of the 2005 conference of the Centre for Advanced Studies on Collaborative research, s. 225-241.

Olkkonen, T. (1994). Johdatus teollisuustalouden tutkimustyöhön. Teknillinen korkeakoulu. 2. painos. 143 s.

Park, C. M., Hong, S., Son, K. H., & Kwon, J. (2007). A component model supporting decomposition and composition of consumer electronics software product lines. Software Product Line Conference, 2007. SPLC 2007. 11th International s. 181-192.

Pipino, L. L., Lee, Y. W., & Wang, R. Y. (2002). Data quality assessment. Communications of the ACM, 45(4), s. 211-218.

Pohl, K., Böckle, G., & van der Linden, F. (2005). Software product line engineering. Springer. 467 s.

Ponniah, P. (2010). Data Warehousing Fundamentals for IT Professionals, Second Edition. John Wiley & Sons, Inc. 504 s.

Rabiser, R., Grünbacher, P., & Dhungana, D. (2010). Requirements for product derivation support: Results from a systematic literature review and an expert survey. Information and Software Technology, Vol. 52(3), s. 324-346.

Rainardi, V. (2008). Building a data warehouse: with examples in SQL Server. John Wiley & Sons.

Rizzi, S., Abelló, A., Lechtenbörger, J., & Trujillo, J. (2006). Research in data warehouse modeling and design: dead or alive?. Proceedings of the 9th ACM international workshop on Data warehousing and OLAP, s. 3-10.

Saunders, M. N., Saunders, M., Lewis, P., & Thornhill, A. (2011). Research methods for business students, 5/e. Pearson Education. 504 s.

Savolainen, J., Mannion, M., & Kuusela, J. (2012). Developing platforms for multiple software product lines. Proceedings of the 16th International Software Product Line Conference-Volume 1, s. 220-228.

Schmid, K & Verlage, M. (2002) The economic impact of product line adoption and evolution. IEEE Software, Vol. 19(6). s. 50–57

Schmid, K. (2000). Scoping software product lines. Software Product Lines, s. 513-532. Springer US.

Sen, A., & Sinha, A. P. (2005). A comparison of data warehousing methodologies. Communications of the ACM, Vol. 48(3), s. 79-84.

Shin, B. (2003). An exploratory investigation of system success factors in data warehousing. Journal of the Association for Information Systems, Vol. 4(1), s. 141-158.

Silvers, F. (2008). Building and maintaining a data warehouse. CRC Press. 304 s.

Simons, H. (2009). Case Study Research in Practice. SAGE. 181 s.

de Souza Filho, E. D., de Oliveira Cavalcanti, R., Neiva, D. F., Oliveira, T. H., Lisboa, L. B., de Almeida, E. S., & de Lemos Meira, S. R. (2008). Evaluating domain design approaches using systematic review. Software Architectures. s. 50-65.

Spewak, S. H., & Hill, S. C. (1993). Enterprise architecture planning: developing a blueprint for data, applications and technology. QED Information Sciences, Inc. 359 s.

Spruijt, R. (2014). Selecting a BI Architecture that fits Organisation's Requirements.

Svahnberg, M. (2003). Supporting Software Architecture Evolution. Blekinge Institute of Technology Dissertation Series, 278 s.

Svahnberg, M., & Bosch, J. (1999). Characterizing evolution in product line architectures. Proceedings of the 3rd annual IASTED International Conference on Software Engineering and Applications, s. 92-97.

Svahnberg, M., & Bosch, J. (2000). Issues concerning variability in software product lines. Software Architectures for Product Families, s. 146-157.

Svahnberg, M., Van Gurp, J., & Bosch, J. (2005). A taxonomy of variability realization techniques. Software: Practice and Experience, Vol. 35(8), s. 705-754.

Tuomi, J. & Sarajärvi, A. (2002) Laadullinen tutkimus ja sisällönanalyysi. Gummerus Kirjapaino Oy. 158 s.

Vasconcelos, A., Sousa, P., & Tribolet, J. (2007). Information system architecture metrics: an enterprise engineering evaluation approach. The Electronic Journal Information Systems Evaluation, 10(1), s. 91-122.

Watson, H. J. (2002). Recent developments in data warehousing. Communications of the Association for Information Systems, Vol. 8(1). s. 1-26.

van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., & Bijlsma, L. (2006). Towards a reference framework for software product management. Requirements Engineering, 14th IEEE International Conference. s. 319-322.

Westerman, P. (2001). Data warehousing: using the Wal-Mart model. Morgan Kaufmann. 297 s.

Vetterli, T., Vaduva, A., & Staudt, M. (2000). Metadata standards for data warehousing: open information model vs. common warehouse metadata. ACM Sigmod Record, Vol. 29(3), s. 68-75.

Wixom, B. H., & Watson, H. J. (2001). An empirical investigation of the factors affecting data warehousing succes. MIS quarterly, s. 17-41.

von der Maßen, T., & Lichter, H. (2002). Modeling variability by UML use case diagrams. Proceedings of the International Workshop on Requirements Engineering for product lines, s. 19-25.

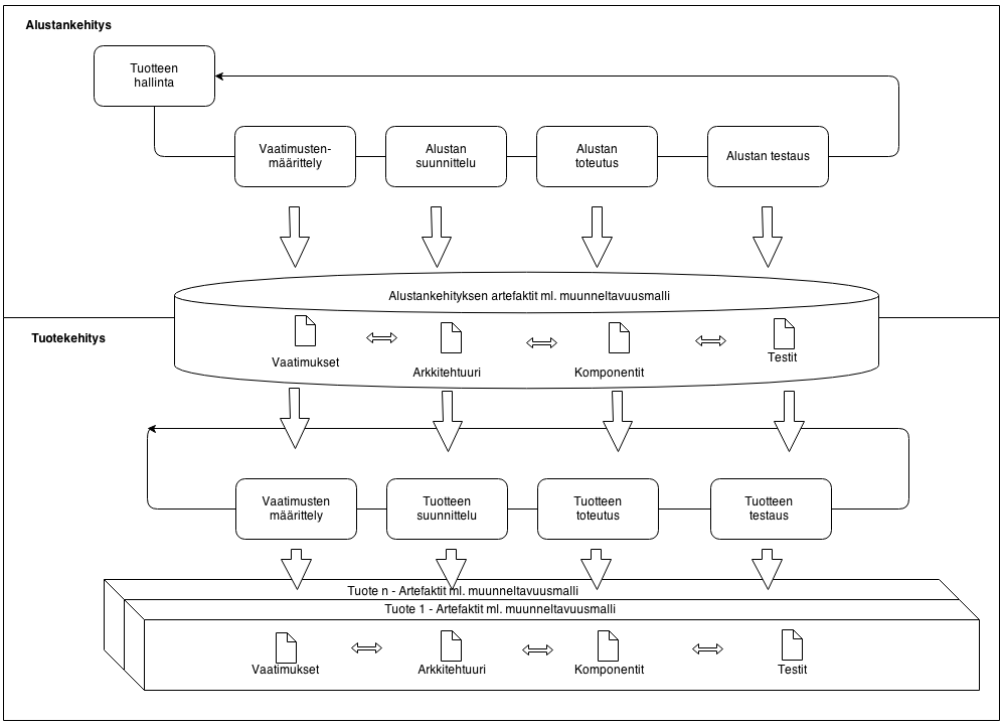
Yeoh, W., & Koronios, A. (2010). Critical success factors for business intelligence systems. Journal of computer information systems, Vol. 50(3), s. 23-32.

Yin, R. K. (2003). Case study research: Design and methods. SAGE publications. 219 s.

LIITE 1: TARKISTUSLISTA

1. TARKISTUSLISTA

1.1 Tuotteistus tuoterunkoon perustuvassa ohjelmistossa



Kuva 8. *Tuoterunkoon perustuvan kehityksen prosessimalli (mukailtu lähteestä Pohl et al. 2005)*

1.1.1 Prosessi ja vaiheiden sisältö

1. Tuotteenhallinta
   - Tuoteportfolion hallinta
   - Tiekartan määrittely / rajaus (engl. scoping)
2. Vaatimustenhallinta
   - Vaatimusten määritys
   - Tuoteperheen yhteisten ja vaihtelevien ominaisuuksien dokumentointi
     - yhteisten piirteiden analyysi
     - muunneltavuusanalyysi
       → Muunneltavuusmalli

3. Alustan suunnittelu
   - Arkkitehtuurisuunnittelu
   - Arkkitehtuurin toteutus ja uudelleenkäytettävien komponenttien karkea suunnittelu
     → Referenssiarkkitehtuuri

4. Alustan toteutus
   - Uudelleenkäytettävien komponenttien tarkempi suunnittelu ja toteutus
5. Alustan testaus

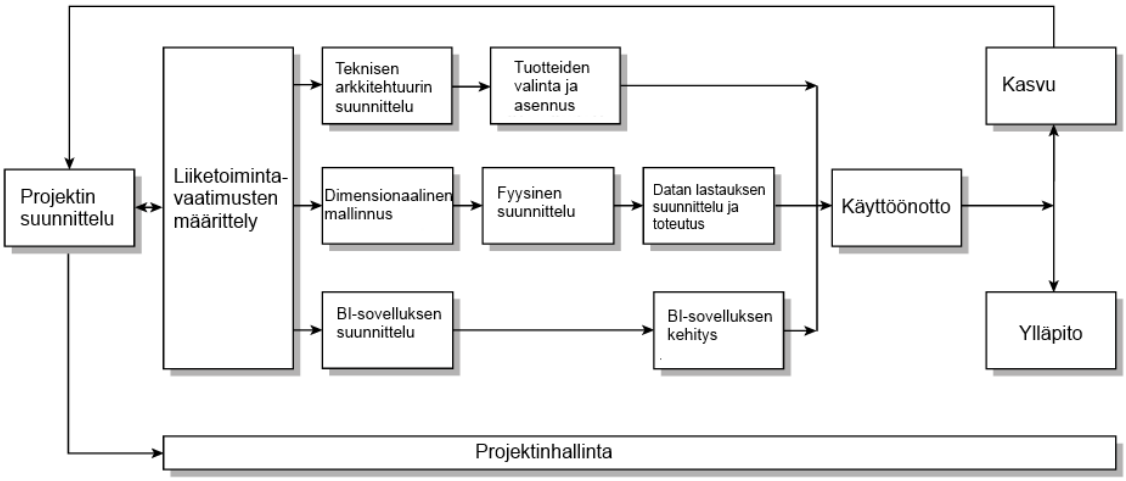
1.1.2 Merkittävät huomioonotettavat asiat

1. Tuotteenhallinta
   - rajauksen (sovellusaluetasolla) määrittäminen (Schimd 2000)
     - liian väljä: tehokkuus kärsii liiallisesta vaihtelusta
     - liian tiukka: kasvupotentiaali kärsii
2. Vaatimuksenhallinta
   - muunneltavuuden mallinnus; variaatiopisteet ja variantit (Pohl et al. 2005; Svahnberg 2003)
3. Alustan suunnittelu
4. Alustan toteutus
   - Rajapintojen suunnittelu ja toteutus komponenteissa (Pohl et al. 2005, s. 52)
5. Alustan testaus

Muut osa-alueet:

1. Muunneltavuuden hallinta
   - Edellytys tuoterungon evoluutiolle, koska kulkevat käsi kädessä (Svahnberg 2003)
2. Monituotelinjojen kyvykkyydet
   - mallinnusnotaatio, joka tukee suuren mittakaavan järjestelmiä
   - rajapinnat muunneltavuuspäätösten esittämiseen tuotelinjoittain
   - ohjeistus tuotteiden johtamiseen
   - riippuvuuksien määrittely variaatiopisteissä (Holl et al. 2012)

1.2 Tietovaraston kehitys



Kuva 9. *Tietovaraston kehityksen elinkaarimalli (mukailtu lähteestä Kimball et al. 2011)*

1.2.1 Prosessi ja vaiheiden sisältö

1. Projektin suunnittelu
   - valmiuksien arviointi
   - projektin rajauksen luominen
   - roolien määrittely
     - dokumentoidaan projektisuunnitelma ja tehtävät, korkean tason liiketoimintavaatimukset, kerättävä data tunnistaminen, odotetut käyttäjät
2. Liiketoimintavaatimusten määrittely
   - Liiketoimintavaatimusten kerääminen
3. Tietovaraston suunnittelu ja toteutus
   - Teknologiarata: teknisen arkkitehtuurin suunnittelu, tuotteiden valinta ja asennus
   - Datarata: dimensionaalinen/käsitteellinen ja looginen mallinnus, fyysinen mallinnus, datan lastauksen suunnittelu ja toteutus
4. Käyttöönotto
5. Ylläpito ja kasvu

1.2.2 Vaiheiden huomioon otettavat seikat

- Lähestymistavan valinta: Bottom-up, top-down tai practical approach (Ponniah 2010; Malinowski & Zimányi 2008)

- Erot perinteisten operatiivisten järjestelmien kehitykseen
    - tarve integroinnille, puhdistamiselle ja tallentamiselle
    - datan määrän nopea kasvu
    - monenlaiset tietokantateknologiat
    - toteutuksen rajaus suurempi
    - projektinhallinnan haasteet rinnakkaisten vaiheiden takia (Ponniah 2010)

- Arkkitehtuurin valinta, vaikuttavat tekijät
    - tietovaraston strateginen merkitys liiketoiminnalle
    - IT-henkilöstön koettu kyvykkyys
    - resurssien rajoitteet (Ariyachandra & Watson 2010, s. 210)
    - Arkkitehtuurin ylläpidettävyys, kyky päivityksille, hyvä suorituskyky, kyky käsitellä suuri määrä dataa, integroitavuus, skaalautuvuus (Spruijt 2014)

Prosessin vaiheet ja huomioonotettavat seikat:

1. Projektin suunnittelu
    - roolien määrittely
    - valmiuksien arviointi
        - datan laatu
        - liiketoimintasponsori
        - hyödyt liiketoiminnalle (Kimball et al. 2011; Kimball & Ross 2002)
2. Vaatimusten määrittely
    - Tarkemmat liiketoimintavaatimukset
        - data-vetoinen
        - tavoitevetoinen
        - käyttäjävetoinen

      lähestymistapa
3. Suunnittelu ja kehitys
    - Datarata
        - käsitteellinen ja looginen malli on tietovaraston onnistumisen kannalta kriittisiä vaiheita (Malinowski & Zimányi 2008)
            1. mukautettujen dimensioiden ja faktojen suunnittelu
            2. data alhaisimman granulariteetin mukaan

3. suunnittelu tietyn raportin mukaan
4. suorituskykyongelmien ratkaisu optimoimalla indeksejä, summataulujen rakentamisella, rinnakkaistamisella (Kimball & Ross 2002)

- Metadatan mallinnus ja ylläpito:
  - yhtenäinen mallinnusnotaatio (Vetterli et al. 2001)
  - ETL-prosessin mallinnus, koska se on aikaa vievin työvaihe (Ponniah 2010; Vassilidis et al. 2002; Rizzi et al. 2006)

- Teknologiarata
  - teknisen arkkitehtuurin luominen
    - tietokannanhallintajärjestelmän valinta, vaatimuksena
      - yhteensopivuus lähteiden ja raportointi- ja analyysisovellusten kanssa
      - ylläpidettävyys
      - lineaarisen kasvun mahdollisuus (Westerman 2000)
      - kyky monipuoliselle indeksoinnille ja suorituskyvyn monitoroinnille (Inmon 2005)

1.3 Tuoterunkoon perustuvan tietovarastoratkaisun laajennettavuus, räätälöitävyys ja uudelleenkäytettävyys

Laajennettavuus:

Ohjelmistoarkkitehtuurissa:

- Suunnittelumallin soveltaminen järjestelmään: mikroydinrakenne, PAC-malli (Meister et al. 2004)
- Konfiguroinnin mahdollistaminen referenssiarkkitehtuurissa ja komponenteissa (Pohl et al. 2005)
- Arkkitehtuurin ja jäljitettävyyden mallinnus, laatuattribuutit huomioonottavien suunnitteluteknikoiden käyttö (Breivold et al. 2010)

Tietomallissa*:*

- dimensioiden suunnittelu, mukautettujen dimensioiden rakennus (Kimball et al. 2011)

- tuntemus sovellusalasta (Pohl et al. 2005)
- Data alhaisimman granulariteetin tasolla (Inmon 2005, Kimball & Ross 2002)
- Sopivan tietokannanhallintajärjestelmän käyttö
    - yhteensopivuus: lähteiden ja raportointisovellusten kanssa
    - mahdollisuus lineaariseen kasvuun
    - tuki erilaisille indeksoinneille ja monitoroinnille (Inmon 2005, Westerman 2000)

Räätälöitävyys:

- dokumentointi: muunneltavuusmalli (Pohl et al. 2005), metadatan hallinta, mappaukset (Kimball et al. 2011)
- Dimensionaalisen mallinnuksen käyttö, mukautetut dimensiot (Kimball et al. 2011)

Uudelleenkäytettävyys:

- Käsitteellinen ja looginen suunnittelu (Malinowski & Zimányi 2008)
- Tuoterungon / referenssiarkkitehtuurin luominen (Pohl et al. 2005)

LIITE 2: TEEMAHAASTATTELUN RUNKO JA APUKYSYMYKSET

- Ohjelmiston tai tietovaraston tuotteistamisen vaiheet
    - Mitä toimia vaiheet sisältää?
    - Mitä huomioon otettavia seikkoja on? kompastuskiviä? kokemuksia epäonnistumisista?
    - Mitä toimenpiteitä on huomioon otettaville seikoille?
- Laajennettavuus, räätälöitävyys ja uudelleenkäytettävyys ohjelmistossa tai tietovarastossa
    - Miten tuoterungon toteutuksessa otetaan huomioon ko. ominaisuudet?
- Tarkistuslistan läpikäynti
    - Ovatko vaiheet olennaisia?
    - Puuttuuko tai onko turhia toimenpiteitä?