Noora Korpelainen

TIETOJÄRJESTELMIEN MALLINTAMINEN – TARPEET JA HAASTEET

JYVÄSKYLÄN YLIOPISTO
INFORMAATIOTEKNOLOGIAN TIEDEKUNTA
2021

TIIVISTELMÄ

Korpelainen, Noora Karoliina
Tietojärjestelmien mallintaminen – tarpeet ja haasteet
Jyväskylä: Jyväskylän yliopisto, 2021, 158 s.
Tietojärjestelmätiede, pro gradu -tutkielma
Ohjaaja: Halttunen, Veikko

Kuten muillakin suunnittelualoilla, myös tietojärjestelmäkehityksessä mallintamisella on käsitetty olevan merkittävä rooli. Alan kirjallisuudessa mallintaminen ja mallien käyttö esitetään hyötyjen valossa ja standardoitu UML (Unified Modeling Language) ohjelmistokehittäjien yhteisenä kielenä. Empiiriset tutkimukset kuitenkin osoittavat, että mallintamisen ja UML:n hyödyntäminen voi olla jopa erittäin vähäistä. Tällä pro gradu -tutkielmalla pyrittiin selventämään tietojärjestelmien mallintamisen nykytilaa. Tutkielman empiirinen osuus toteutettiin laadullisin menetelmin, haastattelemalla 13 ammattilaista kymmenestä eri ohjelmisto- ja IT-palveluyrityksestä. Mallintamisen roolia nykypäivän ketterässä tietojärjestelmäkehityksessä tutkittiin selvittämällä Suomessa toimivien ammattilaisten näkemyksiä ja kokemuksia mallintamisen ja mallien käytön hyödyistä, mallintamiseen liittyvistä haasteista sekä käytössä olevista mallintamismenetelmistä ja -työkaluista. Haastatteluaineiston analyysi suoritettiin teoriaohjaavasti. Teemahaastattelu ja alustava analyysi pohjautuivat kirjallisuudesta muodostettuihin teemoihin, kun taas yksityiskohtaisempi analyysi toteutettiin aineistolähtöisesti. Tutkimuksessa havaittiin, että mallintamisella on tietojärjestelmäkehityksessä selkeä viestinnällinen rooli, joka korostuu kehitystyön alkuvaiheessa ja myöhemmin ylläpidossa. UML on käytetyin mallintamismenetelmä ja sitä hyödynnetään vapaamuotoisella tavalla lähinnä piirtotyökaluja käyttäen. Ammattilaisten koulutus ja kokemus sekä organisaation toimintatavat ja asiakasvaatimukset vaikuttavat käytäntöihin ja mallintamisen koettuun hyödyllisyyteen. Haasteissa esiintyivät etenkin puutteelliset resurssit, kuten ajan, osaamisen ja organisaation tuen puute. Ammattilaisten näkemysten ja kokemusten perusteella todetaan, että koodikeskeiset asenteet voivat vaikuttaa haasteiden muodostumiseen. Haasteet voivat johtaa vapaamuotoiseen mallintamiseen tai mallintamisen sivuuttamiseen kokonaan, jolloin kommunikointiongelmien kautta voidaan havaita laadun ja tuottavuuden laskua. Haasteiden ylittäminen vaatii todennäköisesti useita toimia, kuten koulutusta, ohjeistusten laatimista sekä työkalukehitystä ja -käyttöönottoa.

Asiasanat: tietojärjestelmien kehittäminen, mallintaminen, mallintamismenetelmät, mallinnuskielet, kaaviotekniikat, UML, ketterä ohjelmistokehitys

ABSTRACT

Korpelainen, Noora Karoliina
Information systems modeling – needs and challenges
Jyväskylä: University of Jyväskylä, 2021, 158 pp.
Information Systems, Master's Thesis
Supervisor: Halttunen, Veikko

Modeling has always been considered essential to all engineering fields information systems (IS) development included. IS literature states multiple benefits received from modeling and portrays UML (Unified Modeling Language) as lingua franca of software developers. However, empirical research has revealed that modeling and UML use could be considered even low. This master's thesis aims to get a clearer view on the state of practice of IS modeling. The empirical part of this thesis has been conducted as a qualitative study containing theme interviews of 13 practitioners from ten different software and IT service organizations operating in Finland. The opinions and experiences of these professionals were used to determine the role of modeling in agile IS development, discovering the needs, benefits, and challenges associated with modeling, as well as the used modeling methods and tools. The analysis of the interview data was abductive in which interview themes and preliminary analysis were derived from literature, and subsequent, detailed analysis emerged inductively from the interview material. The results show that in IS development modeling plays a clear communicative role which is emphasized in the early stages of development work and later in maintenance. UML is the most used modeling method and is used in an informal manner utilizing mainly drawing tools. The educational and experiential modeling backgrounds of the practitioners, organizational procedures, and customer requirements are factors that affect the use and perceived usefulness of modeling. Lack of resources such as time, competence, and organizational support emerges as the most prominent challenge. The opinions and experiences of IS professionals suggest that the challenges may stem from code-centric attitudes. These challenges can cause professionals to resort to informal modeling or to disregard modeling altogether which in turn may lead to communicative problems resulting in productivity and quality issues. Overcoming the challenges is likely to require several actions, such as training, guidelines, and tool development and deployment.

Keywords: information systems development, modeling, modeling methods, modeling languages, diagramming techniques, UML, agile software development

KUVIOT

TAULUKOT

SISÄLLYS

1 JOHDANTO

Tietojärjestelmien kehittäminen (engl. information systems development) on pitkälti yhteistyötä. Erilaisten sidosryhmien – kehitystiimin jäsenten, yhteistyöyritysten ja asiakkaiden – on ymmärrettävä toisiaan ja toimittava yhteen, jotta monimutkainen järjestelmä tai sen osa saadaan rakennettua tarkoituksenmukaisesti. Kehitystyö on perinteisesti sisältänyt mallintamista (engl. modeling), aktiviteettia, jonka tuloksena syntyy kuvaus toteutettavasta tietojärjestelmästä (ks. Clarke, Burton-Jones & Weber, 2016). Tämä kuvaus, malli (engl. model), esittää monimutkaisen kokonaisuuden pelkistetyssä, ymmärrettävämmässä muodossa ja siten edistää yhteistyötä (Mylopoulos, 1992; Selic, 2003).

Mallintaminen on oikeastaan itsestäänselvyys kaikilla suunnittelun aloilla. Ilman malleja, kuten pohjapiirroksia, prototyyppejä ja 3D-malleja, emme edes harkitsisi talon rakentamista tai auton valmistamista. Saman lähtökohdan voisi kuvitella koskevan myös monimutkaisten ja abstraktien tietojärjestelmien kehittämistä, varsinkin kun mallintamista pidetään olennaisena osana tietojärjestelmäkehitystä (ks. Bubenko, 2007; Clarke ym., 2016; Wand & Weber, 2002). Tästä huolimatta tietojärjestelmien mallintamisen käytännön rooli on ollut epäselvä ja vaihteleva (Forward, Badreddin & Lethbridge, 2010; Selic, 2003; Störrle, 2017) ja ammattilaisten kerrotaan jopa välttelevän mallien käyttöä (ks. Selic, 2012).

Tietojärjestelmiä on mallinnettu jollain tapaa 1950-luvulta lähtien (Bubenko, 2007). Nykypäivänä mallintaminen tietojärjestelmien kehittämisessä voidaan jakaa kahteen koulukuntaan: perinteiseen mallintamiseen ja mallipohjaiseen kehittämiseen (engl. model-driven development, model-driven engineering). Jos tietojärjestelmien kehittämistä tarkastellaan kolmen eri vaiheen eli analyysin (engl. analysis), suunnittelun (engl. design) ja toteutuksen (engl. implementation) kautta, koskee perinteinen mallintaminen niistä kahta ensimmäistä (Liddle, 2011; Wand, Monarchi, Parsons & Woo, 1995). Analyysissä luodaan korkean abstraktiotason yleismalli, jonka avulla suunnitellaan yksityiskohtainen, toiminnallinen järjestelmän malli, joka toteutuksessa toimii koodaustyön pohjana (Liddle, 2011; Selic, 2003; Wand ym., 1995).

Mallipohjaisessa kehittämisessä mallit eivät ainoastaan tue suunnittelua ja toteutusta, vaan ovat järjestelmän toteutuksen keskeinen väline (Brambilla, Ca-

bot & Wimmer, 2017). Mallipohjaisuuteen liittyy nimittäin ohjelmiston automaattinen toteuttaminen generoimalla koodi suoraan mallista (France & Rumpe, 2007; Selic, 2003). Mallipohjaisella suuntauksella on oma vankka kannattajajoukkonsa, mutta toistaiseksi perinteinen mallintaminen on alalla yleisempää (Elaasar, Noyrit, Badreddin & Gérard, 2019; Mussbacher ym., 2014). Tässä tutkielmassa keskitytään perinteiseen tietojärjestelmäkehitykseen, jossa analyysin ja suunnittelun aikana luotuja malleja voidaan hyödyntää tietojärjestelmän koko elinkaaren ajan erilaisissa aktiviteeteissa, kuten testauksessa (engl. testing) ja ylläpidossa (engl. maintenance) (ks. Pressman, 2005, s. 143, 210, 262).

Alalle on tyypillistä jatkuvasti uusien kehitystyössä hyödynnettävien menetelmien (engl. method, methodology) kehittäminen ja käyttöönotto (Fettke, 2009). Erilaisia menetelmiä on kehitetty niin paljon, että jo 1990-luvulla tutkijat kutsuivat tilannetta termeillä YAMA (Yet Another Modeling Approach) (Oei, van Hemmen, Falkenberg & Brinkkemper, 1992) ja NAMA (Not Another Modeling Approach) (Siau, Wand, & Benbasat, 1996). Menetelmien laajaa kirjoa pyrittiin yhtenäistämään, minkä seurauksena standardoitiin yleiskäyttöinen UML-mallinnuskieli (Unified Modeling Language, ks. Booch, Jacobson & Rumbaugh, 1996; OMG, 2017), joka on sittemmin yhdistetty erottamattomasti ohjelmistokehitykseen (Elaasar, Noyrit, Badreddin & Gérard, 2018; Lange, Chaudron & Muskens, 2006). Kiinnostus uusien menetelmien kehittämiseen tai käyttöönottoon ei niiden valtavasta määrästä huolimatta ole hiipunut (Siau & Rossi, 2011) ja menetelmät kehittyvät teknologioiden mukana ja tarpeiden muuttuessa (Platt & Thompson, 2019; Sinz, 2019).

Mallintaminen on aktiivisesti tutkittu ala tietojärjestelmätieteissä (Burton-Jones, Wand & Weber, 2009; Recker, 2012), mutta käytännön mallintamistyön tutkiminen on jäänyt vähäisemmälle huomiolle (Davies, Green, Rosemann, Indulska & Gallo, 2006; Budgen, Burn, Brereton, Kitchenham & Pretorius, 2011; Petre, 2013). Erilaisten menetelmien kehittämiseen on panostettu, mutta minkäänlaista ajan tasalla olevaa tilastoa ei niiden käytöstä ole. Ei ole myöskään tietoa siitä, missä laajuudessa mallintamista suoritetaan. Tämän lisäksi menetelmiä on kehitetty lähinnä intuition ja maalaisjärjen pohjalta, eikä perustuen mihinkään teoriaan tai empiiriseen todistusaineistoon (Siau, 2004; Siau & Rossi, 2011). Tutkimusaukko on alalla noteerattu ja viime vuosina tietojärjestelmäkehityksessä toimivien ammattilaisten suorittamaa mallintamista on alettu tutkimaan. Tutkimusten tulokset ovat kuitenkin osoittautuneet keskenään ristiriitaisiksi (ks. esim. Petre, 2013; Störrle, 2017).

Ristiriitaisuutta ja käytäntöjen monimuotoisuutta voi osaltaan selittää mallintamiseen liittyvät asenteet, jotka ovat vaihdelleet huomattavastikin vuosien varrella (Forward ym., 2010; Selic, 2003). 2000-luvun alusta lähtien suosiotaan kasvattaneella ketterällä ohjelmistokehityksellä (engl. agile software development, ks. Beck ym., 2001) on ollut merkittävä vaikutus tietojärjestelmien kehittämistyöhön ja oletettavasti myös mallintamiseen liittyviin asenteisiin. Juuri kun hajanaisia näkemyksiä oli yhtenäistetty UML:n muodossa ja näin toivottu selkeyttä käytännön mallintamistyöhön, toteutuskeskeinen ketteryys alkoi val-

lata alaa. Toteutuskeskeisyys ei välttämättä sovi yhteen mallintamislähtökohdan kanssa, jossa korostetaan analyysiä ja suunnittelua (Liddle, 2011).

Ketteryys on nähtävissä koodikeskeisessä kehitystyössä, jossa ei mahdollisesti mallinneta ollenkaan tai se tehdään ylimalkaisesti (Seidewitz, 2003). Inkrementaalisen ohjelmistotuotannon näkökulmasta mallintaminen voidaankin kokea toissijaiseksi jatkuvien muutoksien vuoksi. Mutta vaikka mallintamiseen liittyy haasteita, tietojärjestelmien rakenteiden ja toimintojen ymmärtämisen ja kommunikoinnin tarve ei kuitenkaan ole poistunut. Useat ongelmat tietojärjestelmäprojekteissa voidaankin jäljittää juuri analyysin vaatimusmäärittelyyn (engl. requirements engineering) (Laplante, 2018; Pressman & Maxim, 2015; Siau, 2002), jossa mallintamisen osuus on perinteisesti ollut keskeinen (Gemino & Wand, 2004) ja johon ketteryydellä on ollut merkittävä vaikutus.

Käytännön mallintamistyöhön liittyvän epäselvyyden vuoksi on mielenkiintoista tutkia mallien käytön merkitystä ja erilaisten menetelmien roolia nykypäivän tietojärjestelmäkehityksessä. Onko ohjelmistokehityksen standardi, UML, jonka sanotaan olevan laajalle levinnyt ja yleisessä käytössä (ks. esim. Rumpe, 2016), todella pääsääntöisesti ohjelmistoammattilaisten suosiossa vai kenties jokin muu menetelmä? Onko mahdollista, että ketterän kehityksen myötä mallintamista ei juurikaan toteuteta? Vai ovatko mallintamistarpeet jopa lisääntyneet, koska teknologian kehitys ja muuttuvat liiketoimintavaatimukset tekevät tietoteknisistä ympäristöistä jatkuvasti monimutkaisempia ja alttiimpia tietoturvaongelmille (ks. Bork, Buchmann, Karagiannis, Lee & Miron, 2019; Messe ym., 2020)? On tärkeää sekä alan tutkimukselle että käytännölle selvittää, kokevatko ammattilaiset mallintamisen hyödylliseksi nykypäivän kehitystyössä. Mallintamiseen liittyvä tutkimus on edelleen vilkasta (Härer & Fill, 2020) ja ponnistukset tulisi suunnata aiheisiin, jotka ovat käytännön työlle merkittäviä.

Tämän tutkielman tavoitteena on selvittää mallintamisen nykytilaa tietojärjestelmäkehityksessä. Menetelmien laajan kirjon ja mallintamiseen aiemmin liitetyn oletetun yleisyyden ja hyödyllisyyden, mutta toisaalta riittämättömän ja ristiriitaisen käytännön tutkimuksen motivoimana esitetään seuraavat tutkimuskysymykset:

1. Mikä on mallintamisen rooli nykypäivän tietojärjestelmäkehityksessä ja mitkä ovat käytössä olevat mallintamismenetelmät ja -työkalut?
2. Mitä ovat mallintamiseen ja mallien käyttöön liittyvät tarpeet ja haasteet?

Tutkielma sisältää kirjallisuuskatsauksen lisäksi laadullisen empiirisen tutkimuksen, jota varten on haastateltu Suomessa toimivissa ohjelmisto- ja IT-palveluyrityksissä työskenteleviä ammattilaisia. Tutkimuksen avulla kartoitetaan, miksi ja millaisissa tietojärjestelmäkehityksen aktiviteeteissa mallintamista hyödynnetään ja mitä menetelmiä sekä työtä tukevia työkaluja (engl. tools) ammattilaiset käyttävät. Tämän lisäksi selvitetään ammattilaisten näkemyksiä ja kokemuksia menetelmiin ja työkaluihin sekä niillä tuotettuihin malleihin liittyvistä haasteista. Tutkimuksella tuotetaan arvokasta tietoa menetelmien, työkalujen ja tietojärjestelmien kehittäjille.

Kirjallisuuskatsauksen tiedonhaussa on käytetty pääasiassa Google Scholarin hakukonetta ja tieteellisten julkaisujen lähdeluetteloita. Tietoa on haettu monipuolisesti usean vuosikymmenen ajalta malleista, mallintamisesta, menetelmistä ja menetelmä- sekä tietojärjestelmäkehityksestä. UML:n sisältämien kaaviotyyppien (engl. diagram type) lisäksi kirjallisuuskatsauksessa käsitellään esimerkinomaisesti myös muita mallinnustapoja. Näistä mainittakoon etenkin Chenin (1976) ER-malli (engl. entity relationship model), joka tietokantamallinnuksen (engl. database modeling) peruspilarina on vaikuttanut merkittävästi tietojärjestelmien mallintamiseen (Gogolla, 2011; Storey, Trujillo & Liddle, 2015).

Tutkielman rakenne on seuraava. Johdannon jälkeen toisessa pääluvussa käsitellään mallia ja mallintamista tietojärjestelmäkehityksessä. Erilaisten mallien ja niiden tarjoamien hyötyjen lisäksi tarkastellaan mallintamismenetelmiä ja niiden kehittämistä. Kolmannessa pääluvussa käsitellään mallintamista käytännön esimerkkien kautta. Ohjelmistokehityksen mallinnusstandardin UML:n lisäksi tarkastellaan käytännön mallintamistyötä käsitteleviä aiempia empiirisiä tutkimuksia. Kirjallisuuskatsauksen jälkeen neljännessä pääluvussa esitellään haastattelututkimuksessa käytetyt tutkimusmenetelmät ja viidennessä tutkimuksen tulokset. Tutkielma päättyy pohdintaan ja johtopäätöksiin, joiden yhteydessä esitetään jatkotutkimusaiheet ja tavat, joilla tuloksia voidaan hyödyntää käytännön tietojärjestelmäkehityksessä.

2 MALLI JA MALLINTAMINEN

Ihmiset ovat hyödyntäneet malleja aikojen alusta lähtien; mallien käyttö on mahdollistanut ongelmanratkaisun tehostamisen ja toiminnan kehittämisen. Mallien luominen perustuu kykyymme käsitteellistää (engl. conceptualize) asioita, muodostaa käsitteitä (engl. concept) ja käsityksiä (engl. conceptions) (Leppänen, 2005). Käsitteiden avulla määrittelemme ja analysoimme ympäröivästä maailmasta tekemiämme havaintoja ja muodostamiamme käsityksiä. Käytämme käsitteitä jokapäiväisessä viestinnässä havainnoinnin, päättelyn ja ymmärryksen seurauksena. (Thalheim, 2018.) Käsitteille voi ymmärtää kaksi tarkoitusta: käsitteitä käytetään asioiden luokitteluun ja käsite sisältää kaiken tiedon asiasta, joka on yhdistettävissä käsitteen nimeen (Thalheim, 2018; White, 1994). Käsitys on käsitettä monimutkaisempi asia, se on ymmärrys ja selitys (White, 1994); tulkinta havainnosta (Falkenberg ym., 1998, s. 47); pelkistys tarkasteltavasta asiasta tietyn näkökulman mukaan (Bjeković, Proper & Sottet, 2014).

Tietojärjestelmät ovat abstrakteja järjestelmiä, joten niiden ymmärtämiseen tarvitaan malleja (Falkenberg ym., 1998, s. 12). Käsitteiden ja käsitysten avulla muodostetut ja usein graafisilla symboleilla esitetyt *käsitteelliset mallit* (engl. conceptual model) (Pohl, 2010, s. 360; Thalheim, 2018; Wand & Weber, 2002) ovat tietojärjestelmäkehityksen perusta (Krogstie, Opdahl & Brinkkemper 2007; Wand ym, 1995). Näiden mallien tuottamista jollain *mallinnuskielellä* (engl. modeling language) kutsutaan *käsitteelliseksi mallintamiseksi* (engl. conceptual modeling) (Mylopoulos, 1992). Tässä tutkielmassa käsitteellisestä mallintamisesta käytetään jatkossa yleistä termiä mallintaminen.

Tässä pääluvussa käsitellään malleja ja mallintamista tietojärjestelmäkehityksessä. Ensimmäisessä alaluvussa esitellään mallin määritelmiä, toisessa alaluvussa tietojärjestelmäkehityksen erilaisia malleja. Kolmannessa alaluvussa käsitellään mallintamista tietojärjestelmäkehityksen analyysissä ja suunnittelussa. Neljännessä alaluvussa läpikäydään mallin ja mallintamisen tarjoamia hyötyjä tietojärjestelmäkehityksessä, jonka jälkeen viimeisessä alaluvussa tarkastellaan mallintamisessa käytettäviä menetelmiä ja niiden kehittämistä.

2.1 Mikä on malli?

Mallille ei ole yhteisesti sovittua määritelmää, vaan se vaihtelee tieteenalan mukaan (Thalheim, 2018) ja myös tieteenalojen sisällä määritelmät ovat moninaisia (Kühne, 2005; Partridge, Gonzalez-Perez & Henderson-Sellers, 2013). Käsitteelliseen malliin liittyy kuitenkin käsitteitä, jotka ovat sille olennaisia ja jotka myös toistuvat määritelmissä. Avison ja Fitzgerald (2006, s. 109) määrittelevät mallin *abstraktioksi* (engl. abstraction), esitykseksi reaalimaailman osasta. Samaan tapaan Kühne (2006) määrittelee mallin abstraktioksi järjestelmästä, mutta tarkentaa sen mahdollistavan ennusteiden ja päätelmien tekemisen. Pohl ja Rupp (2015) sanovat mallin olevan abstrakti esitys (engl. abstract representation) todellisuudesta tai luotavasta todellisuudesta; Siau (2004) jostain reaalimaailman osasta. Wandin ym. (1995) mukaan malli on abstrakti kuvaus organisaatioympäristöstä.

Abstraktiolla tarkoitetaan tarkastelun kannalta olennaisten asioiden esiintuomista ja epäolennaisten asioiden häivyttämistä. *Abstrahointi* (engl. abstraction) on todellisuuden pelkistämistä; sen avulla voidaan hallita monimutkaista kokonaisuutta. (Avison & Fitzgerald, 2006, s. 109; Leppänen, 2005, s. 102.) Malli on siis tarkoituksellisesti abstrahoitu (Falkenberg ym., 1998, s. 51), pelkistys todellisuudesta (Booch, Rumbaugh & Jacobson, 2005) ja siten kuvaamaansa kohdetta korkeammalla abstraktiotasolla. Kohdetta voidaan kuvata myös usealla eri abstraktiotasolla, yleisestä korkean tason mallista yksityiskohtaisempaan (Pressman, 2005, s. 265). Pelkistyksestä huolimatta mallin tulisi sisältää kaikki kuvaamansa kohteen olennaiset piirteet (Avison & Fitzgerald, 2006, s. 109).

Vaikka mallin määritelmät eroavat toisistaan merkittävästikin, lähes kaikkien alojen tutkijat ovat kuitenkin yhtä mieltä siitä, että mallin avulla tuotetaan tietoa olennaisista asioista (Leppänen, 2005, s. 279). Mylopoulos (1992) esittää, että malli kuvaa oleellisia osia jostain maailmasta ja siellä tapahtuvista aktiviteeteista. Seidewitz (2003) määrittelee mallin joukoksi toteamuksia (engl. statement) jostain tarkasteltavasta järjestelmästä (engl. system under study), Lindland, Sindre ja Solvberg (1994) jostain kohdealueesta (engl. domain).

Tutkittavaa asiaa, todellisuutta tai järjestelmää kutsutaan tietojärjestelmätieteen kirjallisuudessa usein *kohdealueeksi* (engl. domain, universe of discourse). Kohdealue on mikä tahansa osa tai näkökulma tarkasteltavasta, olemassa olevasta tai suunnitellusta ”maailmasta” (Falkenberg ym., 1998, s. 46; Pohl, 2010, s. 360). Wand ym. (1995) esittävät, että analyysivaiheen mallin tulee kuvastaa tietoa kohdealueesta, ei niinkään toteutettavasta tietojärjestelmästä. Määritelmässä voi olla tärkeässä roolissa myös kohdealueen tarkastelija tai heistä muodostuva joukko. Shanks, Tansley ja Weber (2003) määrittelevät mallin reaalimaailman kohteen kuvaukseksi, jonka IT-ammattilaiset ovat luoneet jonkun ihmisen tai jonkun ryhmän käsityksen ja havaintojen perusteella.

Malli voidaan määrittää sen tarkoituksen ja käytön perusteella. Tarkoitus voi olla tarkkaan esitetty, kuten Mylopouloksen (1992) mukaan: malli toimii yhteisymmärryksen ja viestinnän välineenä. Moody (2005) sanoo mallia suun-

nitteluartefaktiksi, jolla aktiivisesti rakennetaan maailmaa, ei ainoastaan kuvailla sitä. Monilla suunnittelun aloilla mallin tarkoitus on olla spesifikaatio rakennettavasta ratkaisusta ja näin on usein myös tietojärjestelmäkehityksessä (Seidewitz, 2003). Tarkoitus voi myöskin olla avoin, jolloin mallin määritelmässä mainitaan, että se on luotu jotain tarkoitusta varten. Steinmüllerin (1993) määritelmässä malli on tietoa jostain asiasta, jonkun luoma jollekulle ja jotain tarkoitusta ja käyttöä varten. Pohl (2010, s. 360) määrittelee mallin abstraktiksi esitykseksi kohdealueesta tiettyä tarkoitusta, käyttöä varten.

Mallin voidaan käsittää tarkoittavan samaa asiaa kuin ihmisen ajattelussa muodostunut käsitys tarkasteltavasta asiasta, mutta tietojärjestelmäkehityksessä on hyvä tehdä ero ihmisen ajattelussa sijaitsevan mielikuvamallin ja jollain tavalla esitetyn mallin välillä (Bjeković ym., 2014). Ihmisen ajattelussa sijaitseva käsitys ei ole suoraan välitettävissä muille ilman, että se esitetään jollain fyysisellä tavalla (Bjeković ym, 2014; Leppänen, 2005). Jotta mallia voidaan hyödyntää yhteistyössä, on sen määritelmälle siis olennaista esitystapa. Mallia, joka esitetään jollain kielellä, kutsutaan *mallin esitykseksi* (engl. model denotation, Falkenberg ym., 1998, s. 55; Leppänen, 2005, s. 280). Kielen avulla tuotettua mallia kutsutaan kirjallisuudessa myös skriptiksi (engl. script) (ks. esim. Gemino & Wand, 2004; Recker, zur Muehlen, Siau, Erickson & Indulska, 2009; Weber, 2003; Wand & Weber, 2002), joka on mallinnusprosessin tuotos (Gemino & Wand, 2004; Wand & Weber, 2002).

Stachowiakia (1973) mukaillen Pohl ja Rupp (2015) esittävät mallille kolme keskeistä ominaisuutta, jotka kiteyttävät hyvin kirjallisuudessa esitetyt mallin määritelmät. Pohlin ja Ruppin (2015) mukaan malli (1) edustaa kuvaamaansa kohdetta, (2) on pelkistetty kuvaus kohteesta ja (3) on pragmaattinen, eli sillä on jokin tarkoitus. Tässä tutkielmassa tarkastellaan tietojärjestelmien kehityksessä käytettäviä ja jollain kielellä luotavia malleja eli mallin esityksiä. Tarkastelun ulkopuolelle jätetään muunlaiset mallit, kuten mielikuvamallit ja fyysisillä osilla rakennetut pienoismallit. Kun tutkielmassa ei ole tarpeen erottaa mallin esitystä mallista, käytetään yleistä termiä malli.

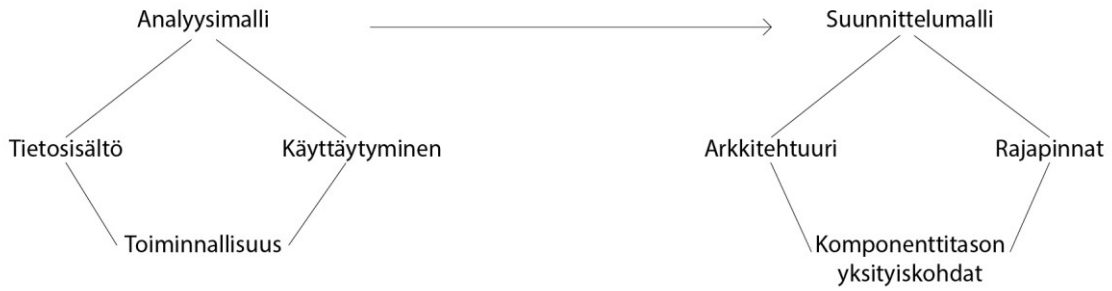
2.2 Mallit tietojärjestelmäkehityksessä

Kirjallisuudessa malleja luokitellaan monella eri tavalla (Leppänen, 2005). Kühnen (2006) mukaan tietojärjestelmäkehityksessä mallit jaetaan usein kuvaileviin (engl. descriptive) ja ohjeellisiin (engl. prescriptive) malleihin. Kuvailevia malleja käytetään jonkin tietämyksen, esimerkiksi vaatimusmäärittelyn tai kohdealueen analyysin esittämiseen (Kühne, 2005). Ohjeellisia malleja käytetään tietojärjestelmän suunnittelussa ja toteutuksessa (Kühne, 2006), ja ne voivat siten sisältää kaavoja, sääntöjä ja käskyjä (Leppänen, 2005). Malli voi olla samanaikaisesti sekä kuvaileva että ohjeellinen; mallia kehittävän näkökulmasta se voi olla kuvaileva ja samalla järjestelmää suunnittelevan näkökulmasta ohjeellinen (Pohl & Rupp, 2015).

Ohjeellinen malli voi olla myös prosessimalli (Pressman, 2005, s. 58—59), abstrakti esitys tietojärjestelmäprosessista (Sommerville, 2016, s. 46). Se määrittää yksityiskohtaiset toimintaohjeet ja tehtävät tietojärjestelmän toteuttamiselle (Pressman, 2005, s. 58—59; Sommerville, 2016, s. 44). Ensimmäinen julkaistu ohjelmistokehityksen prosessimalli on ns. vesiputousmalli (engl. the waterfall model, ks. Royce, 1970) (Sommerville, 2016, s. 47), joka edustaa perinteistä, suunnitteluvetoista kehittämistä. Tunnetuin ketterän ohjelmistokehityksen prosessimalli lienee Scrum (ks. Schwaber & Beedle, 2002). Sommerville (2016, s. 44) huomauttaa, että vaikka prosessimallit poikkeavat toisistaan, tulisi niistä jokaisen sisältää neljä keskeistä aktiviteettia. Nämä aktiviteetit ovat (1) vaatimusmäärittely järjestelmän toiminnallisuuksista ja rajoitteista, (2) toteuttaminen määrittelyn mukaisesti, (3) validointi asiakasvaatimuksiin ja (4) ylläpito, jossa ohjelmistoa kehitetään vastaamaan muuttuvia asiakastarpeita (Sommerville, 2016, s. 44).

Pressman (2005, s. 139; 2010, s. 106) jaottelee tietojärjestelmäkehityksessä käytettävät mallit *analyysimalleiksi* (engl. analysis model) ja *suunnittelumalleiksi* (engl. design model). Analyysimalli, jota kutsutaan myös vaatimusmalliksi (engl. requirements model), edustaa asiakkaiden vaatimuksia. Se esittää ohjelmiston kolmessa eri kohdealueessa tai näkökulmassa: tieto (engl. information domain, data perspective), toiminnallisuus (engl. functional domain, perspective) ja käyttäytyminen (engl. behavioral domain, perspective). (Pohl, 2010, s. 214—215; Pressman, 2010, s. 109; Pressman & Maxim, 2015, s. 114.)

Analyysimalli voidaan muuntaa suunnittelumalliksi. Suunnittelumalli edustaa järjestelmän ominaisuuksia, jotka auttavat ohjelmistoammattilaisia rakentamaan järjestelmän tehokkaasti. Näitä ominaisuuksia ovat arkkitehtuuri (engl. architecture), rajapinnat (engl. interface) ja komponenttitason (engl. component-level) yksityiskohdat. (Pressman, 2010, s. 109.) Luvussa 2.3 käsitellään tietojärjestelmäkehityksen analyysi- ja suunnittelumallien (kuvio 1) luomista.



KUVIO 1 Analyysi- ja suunnittelumallit

Analyysimalli on oikeastaan kokoelma malleja (Pressman, 2005, s. 207), joten se voidaan jakaa edelleen kolmeen mallityyppiin: *tietomalleihin* (engl. data model), *toiminnallisiin malleihin* (engl. functional model) ja *käyttäytymismalleihin* (engl. behavioral model) (Pohl, 2010, s. 223). Tietomallia voidaan kutsua myös rakenteelliseksi malliksi (engl. structural model) (Guizzardi, 2005), käsitekaavioksi

(engl. conceptual schema) (ks. esim. Mylopoulos, 1992; Olivé, 2007) tai staattiseksi malliksi (engl. static model), ja toiminnallista sekä käyttäytymismallia dynaamiseksi malliksi (engl. dynamic model) (ks. esim. Leppänen, 2005, s. 283).

Mallit dokumentoivat vaatimusmäärittelyssä käytettävää kolmea näkökulmaa, joista jokainen edustaa tiettyä näkymää ohjelmistoratkaisusta käsitteellisellä tasolla (Pohl, 2010, s. 214—215). Tietomalli dokumentoi järjestelmän staattiset aspektit, kuten kohdetyypit (engl. entity) ja niiden väliset suhteet (engl. relationship) sekä järjestelmän kannalta olennaiset ominaisuudet (engl. attribute) (Pohl, 2010, s. 223). Toiminnallinen malli dokumentoi järjestelmän vaatimukset toimintojen ja tietovaraston välisinä tietovirtoina (Pohl, 2010, s. 238). Käyttäytymismalli dokumentoi järjestelmän dynaamisen käyttäytymisen tyypillisesti tiloina (engl. state), tapahtumina (engl. event) ja tapahtumien laukaisemina tilan siirtyminä (engl. state transition) (Pohl, 2010, s. 249—250). Luvussa 2.3.1 esitellään mallien kuvaamiseen käytettäviä kaaviotyyppejä.

Liddle (2011) esittää, että on hyödyllistä jakaa mallit tekstimuotoisiin, graafisiin ja matemaattisiin malleihin. Samaan tapaan Leppänen (2005) luokittelee mallit niiden luomiseen käytettävien kielten perusteella epäformaaleiksi, puoliformaaleiksi ja formaaleiksi. *Epäformaalia mallia* (engl. informal model) kutsutaan myös vapaaksi malliksi (Wijers, 1991, s. 15), koska sen rakennetta eivät rajaa mitkään säännöt. Tyypillisiä epäformaaleja malleja ovat sanalliset kuvaukset ja vapaat piirrokset. *Puoliformaalin mallin* (engl. semi-formal model) rakennetta määrittää käytetyn kielen sisältämät säännöt. Kaaviot, taulukot ja matriisit ovat esimerkkejä puoliformaaleista malleista. *Formaali malli* (engl. formal model) esitetään tarkasti määritetyllä kielellä, kuten ohjelmointikielellä, tai loogisilla tai matemaattisilla merkeillä. (Leppänen, 2005, s. 282—283.) Tässä tutkielmassa painopiste on analyysin ja suunnittelun aikana tuotetuissa puoliformaaleissa malleissa, mallinnuskielellä luoduissa kaavioissa. Luvussa 2.5.1 käsitellään tarkemmin mallinnuskieliä.

2.3 Mallintaminen

Alan kirjallisuudessa mallintaminen on esitetty olennaisena osana tietojärjestelmien kehitystyötä (ks. Bubenko, 2007; Clarke ym., 2016; Wand & Weber, 2002). Olivén (2007) mukaan mallintaminen on toimintaa, jolla saadaan esiin ja kuvataan tietyn tietojärjestelmän kannalta tarpeellista yleistä tietämystä. Wand ym. (1995) kutsuvat mallintamista prosessiksi, jossa määritetään ja kuvataan tietämystä. He painottavat, että mallintamisessa ei ole ”suoraa pääsyä” todellisuuteen, vaan havainnot todellisuudesta tai tietämys kohdealueesta toimivat mallintamisen pohjana. Mylopouloksen (1992) määritelmä on alan kirjallisuudessa yleisesti tunnustettu ja käytetty (Verdonck ym., 2019). Hän määrittelee mallintamisen aktiviteetiksi, jossa formaalilla tavalla kuvataan fyysisen ja sosiaalisen maailman osia ymmärryksen ja viestinnän edesauttamiseksi. Mallintaminen tukee psykologisesti perusteltuja rakenteita ja päätelmiä ja on tarkoitettu ensisijaisesti ihmisten, ei koneiden käyttöön. (Mylopoulos, 1992.)

Tunnettuja oppikirjoja ohjelmistotuotannosta (engl. software engineering) ja mallintamisesta julkaisseet tahot lähestyvät mallintamista käytännönläheisemmin. Pressman (2005, s. 56) sanoo mallintamisen olevan aktiviteetti, jolla luodaan malleja, joiden avulla voidaan paremmin ymmärtää ohjelmistovaatimuksia ja niiden toteuttamiseksi laadittavaa suunnitelmaa. Sommerville (2016, s. 139) määrittelee mallintamisen prosessiksi, jossa luodaan järjestelmästä abstrakteja malleja. Malleista jokainen esittää erilaista näkökulmaa kyseisestä järjestelmästä (Sommerville, 2016, s. 139). UML:n kehittäjät Booch, Rumbaugh ja Jacobson (2005) sanovat mallintamisen olevan keskeinen osa aktiviteetteja, jotka johtavat hyvän ohjelmiston käyttöönottoon.

Määritelmästä riippumatta mallintamisen perimmäinen tarkoitus on tukea tietojärjestelmän toteuttamista. Myös tietojärjestelmistä on useita määritelmiä (Olivé, 2007), joita tarkastellaan perinteisesti järjestelmän tarjoamien toimintojen sekä sen rakenteen ja käyttäytymisen kautta (Dori, 2011; Hirschheim, Klein & Lyytinen, 1995, s. 11: Olivé, 2007). Tietojärjestelmän rakenne ja käyttäytyminen mahdollistavat sen toiminnot, joten mallintamisessa molemmat näkökohdat tulee ottaa huomioon (Dori, 2011). Jotta rakennettava järjestelmä vastaa tarkoitustaan, on mallinnettava sekä staattisia näkökohtia (esim. tietokohteet ja niiden ominaisuudet) että dynaamisia näkökohtia (esim. tapahtumat ja prosessit) (Fettke, 2009; Wand & Weber, 2002).

Mallintamisen ensisijaisiksi tavoitteiksi voidaan esittää tietojärjestelmän toiminnan kannalta olennaisen tietämyksen selvittäminen ja määrittäminen (ks. Olivé, 2007, s. 33) ja siihen liittyvän viestinnän edesauttaminen eri sidosryhmien välillä (ks. Parsons & Cole, 2005). Viestintä eri ammattilaisten kesken ei yleensä vaadi kovinkaan yksityiskohtaisen mallin luomista, vaan yhteisymmärrys kokonaiskuvasta on riittävä. Mallintamiseen liittyy kuitenkin myös tarkempia tehtäviä, jotka vaativat paljon yksityiskohtaisempien käsitteiden käyttöä. (Frank, 2002.) Mallintamista suositellaankin yleensä tehtäväksi usealla eri abstraktiotasolla (ks. esim. Frank, 2002; Pressman, 2005, s. 139). Ensin järjestelmä kuvataan asiakkaan näkökulmasta ja myöhemmin teknisemmällä ja yksityiskohtaisemmalla tasolla (Pressman, 2005, s. 139).

Mallintaminen koskee tietojärjestelmäkehityksen analyysiä ja suunnittelua (Pressman, 2005, s. 56; Wand ym., 1995). Analyysissä määritetään, millaista järjestelmää kehitetään, ja suunnittelussa määritetään tavat, joilla tämä järjestelmä toteutetaan (Pohl, 2010, s. 26). Analyysin aikana mallinnetaan kohdealuetta ja se on siten vielä toteutuksesta riippumatonta, toisin kuin suunnitteluvaiheen mallintaminen, joka koskee toteutettavaa järjestelmää (Parsons & Wand, 1997). Mallintamisen eri vaiheissa tarvitaan usein erilaisia tapoja esittää malleja (van Vliet, 2007. s. 249), kuten UML:n käyttötapauskaavioita (engl. use case diagram) analyysivaiheessa ja komponenttikaavioita (engl. component diagram) suunnitteluvaiheessa. Seuraavissa alaluvuissa käsitellään mallintamista analyysissä ja suunnittelussa. Vaiheiden näennäisestä peräkkäisyydestä ja erillisistä tuotoksista huolimatta ne voivat käytännössä olla toisiinsa limittyneitä ja osa iteratiivista prosessia.

2.3.1 Mallintaminen analyysissä

Tietojärjestelmän kehittämistä voidaan ajatella ongelmanratkaisuprosessina, jossa ensin määritetään ongelma ja sen jälkeen suunnitteluprosessi, jonka avulla ratkaisu saavutetaan (Boman, Bubenko, Johannesson & Wangler, 1997, s. 1). Analyysivaiheen mallintaminen on ensimmäinen askel tässä ongelmanratkaisussa (Pressman, 2005, s. 140) eli uuden järjestelmän tai sen osan luomisessa. Tavoitteena on järjestelmää koskevien vaatimusten eli toiminnallisten ja ei-toiminnallisten ominaisuuksien määrittäminen (Ramnath & Dathan, 2011, s. 134) ja niiden kuvaaminen mallien avulla. Tämä vaihe on tärkeä, koska sen aikana tehdyt virheet johtavat ongelmiin suunnittelussa ja toteutuksessa (Sommerville, 2016, s. 54).

Analyysi sisältää erilaisia vaatimuksiin liittyviä tehtäviä eli vaatimusten hankintaa, tarkentamista, neuvottelua, määrittelyä, validointia ja hallintaa (Pressman, 2005, s. 56; Pressman & Maxim, 2015, s. 133). Osa näistä tehtävistä suoritetaan samanaikaisesti ja kaikki mukautetaan projektin tarpeisiin (Pressman & Maxim, 2015, s. 133). Näitä tehtäviä kutsutaan yleensä *vaatimusmäärittelyksi*. Perinteisessä suunnitteluvetoisessa kehitystyössä vaatimusmäärittelyn tehtävät ovat peräkkäisiä ja niistä muodostetut dokumentit lyödään lukkoon ennen suunnitteluvaihetta. Ketterässä, iteratiivisessa kehityksessä tehtävät ja niiden järjestys ovat paljon epäformaalimpia. (Curcio, Navarro, Malucelli & Reinehr, 2018).

Analyysi aloitetaan kuvaamalla ongelma käyttäjien näkökulmasta ilman toteutuksen yksityiskohtia, yleensä tekstimuotoisilla skenaarioilla (engl. usage scenario), joita kutsutaan myös käyttötapauksiksi (engl. use case, ks. Jacobson, Christerson, Jonsson & Overgaard, 1992) (Pressman, 2005, s. 141; Pressman & Maxim, 2015, s. 146). Käyttötapauksilla kuvataan toiminnallisuutta käyttäjien ja järjestelmän välillä (Rumbaugh, Jacobson & Booch, 1998, s. 26; Cockburn, 2001). Tekstimuotoiset käyttäjätarinat (engl. user story) ovat yleistyneet vaatimusten kuvaustapana ketterässä kehityksessä (Schön, Thomaschewski & Escalona, 2017). Analyysissä tekstimuotoisia vaatimuksia voidaan selventää korkean abstraktiotasotason graafisilla malleilla, mikä on käsitetty yleiseksi toimintatavaksi perinteisessä suunnitteluvetoisessa kehitystyössä (Curcio ym., 2018).

Kaikkia järjestelmiä määrittää rakenne, toiminnot ja käyttäytyminen (Dori, 2011), ja analyysivaiheessa mallinnetaankin perinteisesti järjestelmän prosessoimaa tietosisältöä, järjestelmän tarjoamia toimintoja ja järjestelmän osoittamaa käyttäytymistä (Pohl, 2010, s. 215; Pressman, 2005, s. 140; Ramnath & Dathan, 2011, s. 134). Analyysin lopputuloksena syntyvä kuvaus koostuu siis jo luvussa 2.2. esitetyistä tieto-, toiminnallisista- ja käyttäytymismalleista, mutta voi projektista riippuen sisältää myös ainoastaan listan käyttötapauksista (Pressman & Maxim, 2015, s. 136).

Tiedon mallintaminen (engl. data modeling)**.** Järjestelmän tietosisällön voidaan sanoa olevan sen rakentamisen perusta (Avison & Fitzgerald, 2006, s. 111) ja sen määrittäminen on siten äärimmäisen tärkeä osuus mallintamisessa (Olivé, 2007, s. 41). Tietosisältö voi olla tietoa tuottava tai käyttävä ulkoinen kohde, asia

(esim. raportti), tapahtuma (esim. puhelu), rooli (esim. myyjä), organisaatioyksikkö (esim. myynti), paikka (esim. toimisto) tai rakenne (esim. tiedosto) (Pressman, 2005, s. 213). Tiedon mallintamisen kohteena on siis reaalimaailman osa, mikä tarkoittaa sitä, että mallintaminen on toteutuksesta riippumatonta. Tiedon mallintamisen prosessi ja sen tuloksena syntynyt tietomalli ovat käyttökelpoisia monenlaisissa käyttökohteissa (esim. tietokanta, tiedosto) ja silloinkin, kun tietojärjestelmää päivitetään tai vaihdetaan uuteen. (Avison & Fitzgerald, 2006, s. 111.) ER-malli (ks. Chen, 1976) ja siitä johdettu UML:n (ks. Booch ym., 1999) luokkakaavio (engl. class diagram) ovat tunnettuja tapoja tiedon mallintamisessa (Pohl, 2010, s. 224; van Vliet, 2007, s. 250).

Toiminnallisuuden mallintaminen (engl. functional modeling). Toiminnallinen näkökulma tyypillisesti määrittää järjestelmän tarjoamat prosessit, datan käsittelyn jokaisessa prosessissa ja prosessien syöte-tuloste-suhteet eli tietovirrat (Pohl, 2010, s. 215). Monitasoista tietovirtakaaviota (engl. data flow diagram) (ks. esim. DeMarco, 1979) pidetään erityisen hyödyllisenä analyysivaiheen viestinnässä (Avison & Fitzgerald, 2006, s. 111) ja se onkin perinteinen ja yleisesti käytetty tapa järjestelmän toimintojen kuvaamiseen ja dokumentoimiseen (Pohl, 2010, s. 215; Pressman, 2005, s. 226). Tietovirtakaavio ei lukeudu UML:ään, mutta vastaavia kaaviotyyppejä UML:ssä ovat aktiviteettikaavio (engl. activity diagram) (Meng, Chu & Zhan, 2010; Sommerville, 2016, s. 155) ja sekvenssikaavio (engl. sequence diagram) (Sommerville, 2016, s. 155; van Vliet, 2007, s. 250).

Käyttäytymisen mallintamisen (engl. behavioral modeling) tavoitteena on kuvata tietojärjestelmän kokonaisvaltainen käyttäytyminen (Pohl, 2010, s. 215), jota ohjaa vuorovaikutus ulkoisen ympäristön kanssa (Pressman, 2005, s. 140). Mallintamisessa määritellään järjestelmän ulkoiset ärsykkeet ja järjestelmän reaktiot sekä näiden ärsykkeiden ja reaktioiden välinen suhde (Pohl, 2010, s. 215). Lisäksi kuvataan järjestelmän tilat ja sallitut siirtymät tilojen välillä (Pohl, 2010, s. 215) sekä siirtymienjälkeiset toiminnot (Pressman, 2005, s. 254). Tilakaavio (engl. statechart, ks. Harel, 1987) ja siitä johdettu UML:n tilakaavio (engl. state machine diagram) sekä sekvenssikaavio ovat tunnettuja tapoja tietojärjestelmän käyttäytymisen kuvaamiseen (Pohl, 2010, s. 215, 250; Pressman, 2005, s. 254).

2.3.2 Mallintaminen suunnittelussa

Suunnitteluvaihe on tietojärjestelmäkehityksen tekninen ydin ja mallintamistyön viimeinen vaihe ennen koodaamista ja testaamista (Pressman, 2005, s. 259). Analyysin aikana luodut mallit sisältävät tietoa, joita tarvitaan suunnittelumallien luomiseen. Tavoitteena on muuntaa mallit muotoon, joka mahdollistaa toteutuksen. (Pressman, 2005, s. 57.) Suunnittelu (*miten tehdään*) on siis toimivan ratkaisun luomista analyysissä määritettyyn ongelmaan (*mitä tehdään*). Ongelmanmäärittely ja ratkaisun suunnittelu ei kuitenkaan ole yksisuuntainen prosessi, vaan suunnittelussa esiin tulleet asiat voivat johtaa takaisin vaatimusten määrittelyyn (Pohl, 2010, s. 27). Vaiheet voivat myös olla toisiinsa lomittuneita, kuten ketterälle kehitykselle on ominaista (Pressman & Maxim, 2015, s. 158; Sommerville, 2016, s. 79).

Ramnath ja Dathan (2011, s. 134) esittävät, että suunnittelu aloitetaan yksityiskohtaisella erittelyllä siitä, miten järjestelmän tulee toisintaa analyysimallissa kuvattua käyttäytymistä. Erittelyssä kaikki järjestelmän osat ja niiden tehtävät määritellään tarkasti (Ramnath & Dathan, 2011, s. 134) ja spesifikaatio koostetaan mallintamalla rakennettavan ratkaisun arkkitehtuuri, rajapinnat ja komponentit sekä niiden sisältämät tarkat tietorakenteet (Pressman, 2005, s. 57). Kehitettyä ratkaisua eli suunnittelumallia tai -spesifikaatiota voidaan vielä arvioida ja parannella ennen koodausta ja testaamista. Suunnitteluvaiheessa rakennetaankin pohja tietojärjestelmän laadulle. (Pressman, 2005, s. 258; van Vliet, 2007.) Sommerville (2016, s. 56) huomauttaa, että ketterässä ohjelmistokehityksessä edellä kuvaillun tapaista tarkkaa suunnitteludokumenttia ei yleensä tehdä eikä yksityiskohtaisia suunnittelun malleja erikseen luoda, vaan suunnittelu nivoutuu toteutukseen. Tällöin suunnittelupäätökset jätetään koodaajalle, joka usein käyttää työnsä pohjana järjestelmästä luotuja epäformaaleja malleja (Sommerville, 2016, s. 197).

Arkkitehtuurin eli järjestelmän rakenteen suunnittelu on kriittinen vaihe monimutkaisen tietojärjestelmän kehittämisessä, toimien siltana vaatimusmäärittelyn ja toteutuksen välillä (Garlan, 2000). Arkkitehtuuri sisältää järjestelmän komponentit, niiden ominaisuudet ja suhteet muiden komponenttien välillä (Bass, Clements & Kazman, 2003). Se toimii myös pohjana muiden samankaltaisten järjestelmien ja uudelleenkäytettävien komponenttien suunnittelussa (Bass ym., 2003; Garlan, 2000; van Vliet, 2007. s. 13). Arkkitehtuurin kuvaukset on käsitetty tärkeäksi osaksi tietojärjestelmän dokumentaatiota (ks. van Vliet, 2007, s. 13). Ne esittävät rakenteen ymmärrettävässä muodossa mahdollistaen viestinnän eri sidosryhmien kesken, vertailun vaatimusmäärityksiin, vaihtoehtoisten ratkaisujen läpikäynnin aikaisessa vaiheessa suunnittelutyötä ja sujuvan ylläpidon (Bass ym., 2003; Garlan, 2000; Pressman, 2005, s. 288).

Arkkitehtuurin kuvaaminen useasta eri näkökulmasta voi helpottaa suunnittelupäätösten tekemistä (Hofmeister, Nord & Soni, 1999). Tähän tarvitaan useita kaavioita, koska yksittäisellä mallilla voidaan kuvata ainoastaan yhtä näkökulmaa järjestelmästä (Sommerville, 2016, s. 173). Kruchtenin (1995) ”4+1 näkymää” on yleisesti hyväksytty tapa eri näkökohtien kuvaamiseen ja se voidaan toteuttaa UML:n erilaisilla kaaviotyypeillä. ”4+1 näkymää” erottaa arkkitehtuurin staattisen ja dynaamisen näkökulman kuvaukset jakaen ne eri kategorioihin: loogiseen, prosessi-, fyysiseen ja kehitysnäkymään. Viides näkymä käsittää neljän muun näkymän esittämisen ja validoinnin käyttötapausten avulla. (Riva & Rodrigues, 2002.) Looginen näkymä (esim. luokkakaavio) kuvaa järjestelmän toiminnalliset ominaisuudet, prosessinäkymä (esim. aktiviteettikaavio) keskittyy ajonaikaiseen käyttäytymiseen, fyysinen näkymä (esim. sijoittelukaavio, engl. deployment diagram) esittää järjestelmän suhteet laitteistoihin ja kehitysnäkymä (esim. komponenttikaavio) kuvaa järjestelmän staattisen rakenteen kehitysympäristössään (Kruchten, 1995; Muchandi, 2007).

Kirjallisuudessa painotettu arkkitehtuurin suunnittelun tärkeys on kyseenalaistettu ketterän ohjelmistokehityksen myötä. Etukäteissuunnittelu, arviointi ja dokumentointi ovat aikaa vieviä tehtäviä ja sen vuoksi niiden voidaan

katsoa olevan yhteensopimattomia ketteryyden kanssa, jossa korostetaan asiakkaalle tuotettavaa arvoa lyhyellä aikavälillä (Abrahamsson, Babar & Kruchten, 2010). Ketterässä ajattelutavassa arkkitehtuurin voidaan ajatella muodostuvan ilman etukäteissuunnittelua kehitystyön edetessä (Abrahamsson ym., 2010; Babar, 2009) ja sen kuvaamisessa keskitytään usein viestintään (Malavolta, Lago, Muccini, Pelliccione & Tang, 2012) yksityiskohtaisen suunnittelun ja dokumentoinnin sijaan.

Rajapinnat. Suunnitteluvaiheen rajapintaelementit kuvaavat tietovirtaa järjestelmään ja siitä ulos sekä kommunikointia arkkitehtuurissa määritettyjen komponenttien kesken. Rajapintojen suunnittelussa on kolme tärkeää osa-aluetta: käyttöliittymä (engl. user interface), ulkoiset rajapinnat muihin järjestelmiin, laitteisiin, verkkoihin tai muihin tietoa tuottaviin tai kuluttaviin elementteihin, ja sisäiset rajapinnat komponenttien välillä. (Pressman, 2010, s. 235.) Esimerkiksi käyttötapauksilla ja käyttäytymismalleilla tuotetaan rajapintojen suunnittelussa tarvittavaa tietoa (Pressman, 2005, s. 261). Käyttöliittymää voidaan esitellä asiakkaille paperiprototyyppien ja mock-up-kuvien avulla (Pohl, 2010, s. 459). Tässä tutkielmassa keskitytään mallinnuskielellä luotuihin kaavioihin, joten käyttöliittymien esittelyssä käytettävät kuvat rajataan käsittelyn ulkopuolelle.

Komponenttitason yksityiskohdat. Komponentti on tietojärjestelmässä vaihdettavissa oleva itsenäinen yksikkö, jolla on selkeästi määritetyt rajapinnat (OMG, 2017, s. 208). Komponentit määritetään korkealla tasolla jo arkkitehtuurivaiheessa (Pressman, 2005, s. 324). Suunnitteluvaiheen edetessä analyysi- ja arkkitehtuurimallit muunnetaan suunnittelumalliksi, joka on tarpeeksi yksityiskohtainen järjestelmän rakentamista eli koodausta ja testausta varten (Pressman, 2005, s. 339). Tietosisällön, arkkitehtuurin ja rajapintojen kuvaukset toimivat komponenttitason suunnittelun pohjana. Niistä muodostetut yksityiskohtaiset mallit täsmentävät jokaisen komponentin sisäiset tietorakenteet, rajapinnat ja prosessointilogiikan. (Pressman, 2005, s. 324—325.) Komponenttien uudelleenkäyttö on tärkeä aspekti komponenttitason mallintamisessa (OMG, 2017, s. 208), ja esimerkiksi web-pohjaiset järjestelmät rakennetaan useimmiten jo olemassa olevia komponentteja hyödyntäen (Sommerville, 2016, s. 28).

2.4 Mallin ja mallintamisen hyödyt

Tietojärjestelmäkehityksessä erilaisten tekniikoiden ja työkalujen käyttöönotolla tavoitellaan tuottavuuden kasvua ja parempaa laatua (Glass, 2002). Siten myös mallintaminen ja mallien käyttäminen perustuvat niiden oletettuun hyödyllisyyteen. Rollandin ja Cauvetin (1992) mukaan malli on erittäin hyödyllinen läpi tietojärjestelmän elinkaaren ja on siksi yksi tietojärjestelmäkehityksen keskeisimmistä välineistä. Frank (1999) esittää mallin olevan jopa edellytys onnistuneelle tietojärjestelmäsuunnittelulle. Mutta miten hyödyt, kuten tuottavuuden kasvu ja laadukas kehitystyön lopputulos saavutetaan mallien avulla? Mahdollisia hyötyjä on tullut esille jo tämän kirjallisuuskatsauksen aiemmissa luvuissa

mallin ja mallintamisen esittelyjen yhteydessä. Luvussa 2.1 kerrottiin, että abstraktio on olennainen käsite mallia määriteltäessä. Se on olennainen myös mallin käyttöä tarkasteltaessa, sillä juuri abstraktion kautta malli auttaa ymmärtämään käsiteltävää ongelmaa ja sen mahdollisia ratkaisuja (Booch ym., 2005; Selic, 2003).

Mallilla voidaan sanoa olevan tietojärjestelmäkehityksessä monta tärkeää tehtävää (ks. Avison & Fitzgerald, 2006, s. 209; Gemino & Wand, 2004; Kung & Soelvberg, 1986; Pressman & Maxim, 2015, s. 17; Wand & Weber, 2002):

- malli toimii kommunikaatiovälineenä
- malli edesauttaa ymmärrystä kohdealueesta ja toteutettavasta ratkaisusta
- malli toimii suunnittelun ja toteutuksen pohjana
- malli toimii dokumentaationa

Mallintamisen hyödyllisyys ilmenee näiden tehtävien kautta, ulottuen kaikkiin tietojärjestelmäprojektin vaiheisiin. Eri vaiheissa malleilla on eri merkitys. Analyysissä päätavoitteena on viestintä, ymmärryksen saavuttaminen ja jakaminen eri sidosryhmien kesken sekä vaatimusten määrittely. Suunnittelussa mallien avulla rakennetaan toiminnallinen ja laadukas järjestelmä. Toteutusvaiheessa mallit ohjaavat koodaajien työtä. Dokumentoidut mallit helpottavat testausta ja ylläpitoa sekä uusien työntekijöiden perehdyttämistä projektiin. (Chaudron, Heijstek & Nugroho, 2012; Dagenais, Ossher, Bellamy, Robillard & De Vries, 2010, Gemino & Wand, 2004; Kung & Soelvberg, 1986; Wand & Weber, 2002). Seuraavaksi käsitellään mallin tehtävien kautta ilmeneviä hyötyjä tarkemmin.

Mallin toimiminen kommunikaatiovälineenä eri sidosryhmien kesken on yksi yleisimmistä kirjallisuudessa mainituista mallin tarjoamista eduista. Asiakkaat eli toteutettavan järjestelmän ostajat ja käyttäjät ymmärtävät paremmin graafista mallia kuin ohjelmakoodia (Liddle, 2011). Rakennettavasta ratkaisusta voidaan keskustella mallin avulla (Frank, 1999) ja määrittää sen pohjalta käyttäjien vaatimukset (Wand & Weber, 2002). Mallin avulla käyttäjät voivat vahvistaa, vastaako ohjelmistoammattilaisten tulkinta todellisuudesta heidän näkemystään (Parsons & Cole, 2005; Shanks ym., 2003). Tätä vaihetta analyysissä kutsutaan mallin validoinniksi (Olivé, 2007, s. 28; Pressman, 2005, s. 179) ja se voidaan suorittaa onnistuneesti vain, jos vaatimukset ovat tarkasti kuvattuja (Olivé, 2007, s. 28). Mallin avulla helpotettu viestintä johtaa yhteisymmärrykseen, realistisempiin odotuksiin ja parempaan lopputulokseen (Liddle, 2011). Mallin riittävyys perustuukin sen kykyyn edistää yhteisymmärrystä ihmiskäyttäjien keskuudessa (Mylopoulos, 1992; Storey ym., 2015).

Malli edesauttaa ymmärrystä kohdealueesta ja kehitettävästä järjestelmästä. Jotta vaatimustenmukainen järjestelmä voidaan suunnitella ja toteuttaa, tarvitaan yleistä tietoa kohdealueesta (Olivé, 2005), kuten monimutkaisesta liiketoiminta-alueesta (Burton-Jones & Meso, 2008). Ymmärrys kohdealueesta, vaatimuksista ja ratkaisusta voidaan saavuttaa mallien avulla (Burton-Jones & Meso, 2008; Olivé, 2005; Pressman & Maxim, 2015, s. 17). Malleja luodaan ja käytetään, koska monimutkaisten kokonaisuuksien hahmottaminen helpottuu niiden avulla (Booch ym., 2005; Giaglis, 2001). Mallit tavallaan voimistavat ihmisen älykkyyt-

tä, sillä oikein valitun mallin avulla mallintaja voi työskennellä korkeammalla abstraktiotasolla (Booch ym., 2005). Tällöin järjestelmän yleiskuva näkyy selkeämmin ja yksityiskohdat vähemmän, eikä monimutkaisuutta tarvitse ottaa tarkastelussa samanaikaisesti huomioon (Liddle, 2011). Mallit voivat siis helpottaa päätöksentekoa (Giaglis, 2001; Gordijn, Akkermans & van Vliet, 2000) ja ohjata työpanoksen olennaisiin osiin tarkastelun alla olevasta kohteesta (Giaglis, 2001).

Vaatimusmäärittelyvaiheessa luotu malli tarjoaa, ainakin teoriassa, vakaan *pohjan tietojärjestelmän suunnittelulle ja toteuttamiselle* (Frank, 1999). Mallien avulla lopputulos voidaan visualisoida ennen sen toteuttamista (Liddle, 2011), mikä mahdollistaa erilaisten vaihtoehtojen läpikäymisen ja näin soveltuvimman ratkaisun löytämisen ilman tietojärjestelmän testaamista käytännössä (Pressman, 2005, s. 288). Ohjeellisella mallilla (ks. Kühne, 2006) voi olla tärkeä rooli myös projektin hallinnassa. Suunnitteluvaiheen malli käsittää tarkan spesifikaation työstä, joka tulee tehdä ja sen avulla voidaan arvioida, aikatauluttaa ja muutoin suunnitella järjestelmän toteutusvaihe (Liddle, 2011).

Malli dokumentoi tulevien käyttäjien ja kehitystyötä tekevien yhteisymmärryksen kohdealueesta ja toiminnoista, joita tietojärjestelmällä tulee olla (Olivé, 2005). Kun järjestelmään kohdistuvat vaatimukset ajan kuluessa muuttuvat, dokumentaatio toimii pohjana, jonka avulla keskustellaan ja suunnitellaan järjestelmälle tehtävät tarvittavat muutokset (Boman ym., 1997, s. 9). Ylläpidossa korkean abstraktiotason mallit edesauttavat ymmärrystä kohdealueesta (Burton-Jones & Meso, 2008; Kung & Soelvberg, 1986) ja järjestelmään perehtymistä koodirivien ja koodikommentoinnin lukemisen ohella (Hunt & Thomas, 2002). Myös projektiin tulevien uusien työntekijöiden on dokumentoitujen mallien avulla mahdollista muodostaa yleiskuva järjestelmästä (Dagenais ym., 2010).

Edellä mainittujen tehtävien kautta voidaan esittää, että *mallien avulla saavutetaan kustannussäästöjä.* Empiiriset tutkimukset ovat osoittaneet, että yli puolet tietojärjestelmäsuunnittelun aikana tehdyistä virheistä johtuvat viallisesta vaatimusmäärittelystä, joka on myös yleisin syy projektien epäonnistumisille (Moody, 2005). Uraauurtavassa julkaisussaan ohjelmistokehityksen taloudellisuudesta Boehm (1981) esittää, että vikojen korjaamisen kustannukset kasvavat eksponentiaalisesti projektin edetessä. Tämä pitää edelleen paikkaansa, on taloudellisinta havaita ja korjata virheet kehitystyön alussa eli vaatimusmäärittelyvaiheessa (Chaudron ym., 2012; Moody, 2005; Sommerville, 2016, s. 129). Laadukkaiden mallien käyttö vaikuttaa siten äärimmäisen oleelliselta, sillä juuri niiden avulla voidaan edesauttaa vaatimusmäärittelyä ja havaita sekä korjata virheet aikaisessa vaiheessa tietojärjestelmäprojektia (Liddle, 2011; Maes & Poels, 2007; Wand & Weber, 2002).

Alan kirjallisuudessa esitetään, että hyödyntämällä malleja kehitystyön aikana voidaan vaikuttaa merkittävästi niiden pohjalta tuotetun tietojärjestelmän laatuun (ks. esim. Endres & Rombach, 2003; Selic, 2012) ja vähentää toteutusvaiheen jälkeistä testauksen määrää (ks. esim. Chaudron ym., 2012). Myös mallien uudelleenkäyttö olisi hyödyllisempää aikaisemmassa vaiheessa ja korkeammalla abstraktiotasolla kuin myöhemmin kooditasolla (ks. esim. Siau & Rossi, 2001). Kun näiden lisäksi huomioidaan aiemmissa kappaleissa esitetyt

hyödyt, todetaan, että *malleilla voidaan vaikuttaa merkittävästi työn tuottavuuteen ja lopputuloksen laatuun* (ks. myös Chaudron ym., 2012; Selic, 2012).

2.5 Mallintamismenetelmät

Mallien luomiseen käytetään mallinnuskieltä (Clarke ym., 2016; Wand ym., 1995). Mallinnuskieliä, kuten UML:ää, kutsutaan usein myös *mallintamismenetelmiksi* (engl. modeling method) (ks. esim. Fettke, 2009; Recker ym., 2009 ja Siau & Rossi, 2011). Joissakin teoksissa (ks. esim. Fowler & Scott, 2000) painotetaan, että tämä on harhaanjohtavaa, koska menetelmien pitäisi ainakin periaatteessa sisältää kielen lisäksi myös prosessivaiheet. Menetelmä-termillä viitataankin yleensä kieltä laajempaan kokonaisuuteen tarkkoine toimintaohjeineen (ks. esim. Davies ym., 2006; vrt. prosessimalli, Pressman, 2005), kuten Soft Systems Methodology -(SSM, ks. Checkland, 1989) ja Rapid Application Development (RAD, ks. Martin, 1991) -menetelmiin. Mallinnuskieliä kutsutaan tällöin menetelmän sijaan usein *mallinnustekniikoiksi* (engl. modeling technique) (ks. esim. Davies ym., 2006) tai *kaaviotekniikoiksi* (engl. diagramming technique) (ks. esim. Siau & Loo, 2006).

Kuten mallille ja mallintamiselle, myös menetelmille löytyy siis kirjallisuudesta erilaisia määritelmiä. Alalla merkittäväksi esitetyn (ks. esim. Bucher, Klesse, Kurpjuweit & Winter, 2007) Brinkkemperin (1996, s. 275—276) julkaisun mukaan menetelmä on lähestymistapa järjestelmäkehitysprojektin suorittamiseksi. Se perustuu tiettyyn ajattelutapaan ja sisältää ohjeita, sääntöjä ja järjestelmällisiä kehitysaktiviteetteja, joiden suorittamisessa käytetään tekniikoita, kuten tiedon mallintamista ER-kaavioilla (Brinkkemper, 1996, s. 275—276). Siau ja Rossi (2011, s. 251) määrittelevät menetelmän samaan tapaan, mutta Brinkkemperistä (1996) poiketen he kuitenkin kutsuvat mallinnuskieliä myös menetelmiksi. Alan tutkimuksissa termejä menetelmä, kieli ja tekniikka käytetäänkin usein vastavuoroisesti (ks. esim. Fettke, 2009; Gemino & Wand, 2004).

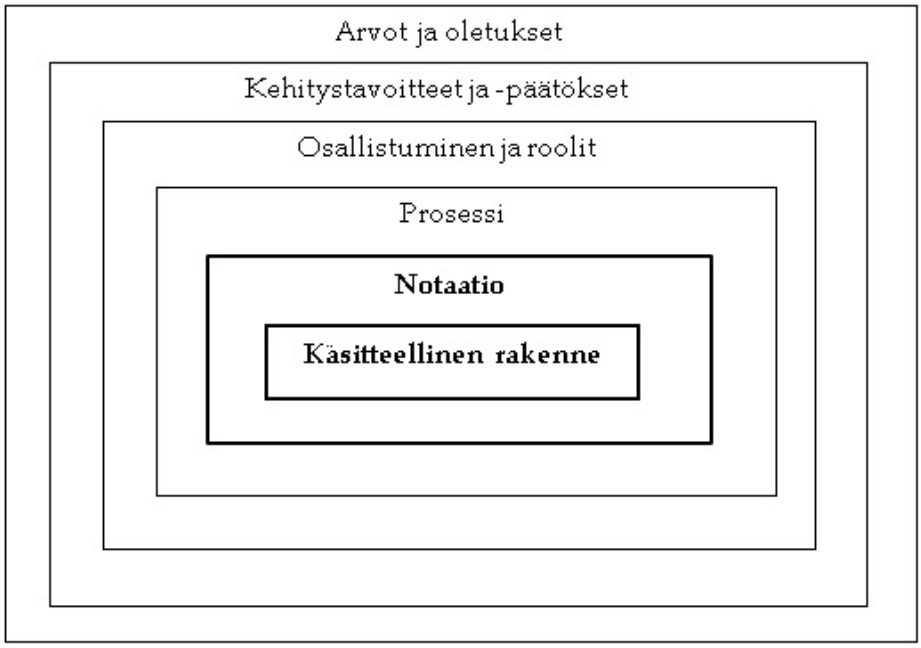
Jotkut tutkijat (ks. esim. Lyytinen, 1987) yhdistävät menetelmän määritelmään myös työkalut, kuten mallintamisessa käytettävät ohjelmistot. Tällaisia työkaluja ovat esimerkiksi Microsoftin piirtotyökalu Visio[1] tai Sparx Systemsin koko järjestelmän suunnittelun ja toteuttamisen mahdollistava Enterprise Architect[2]. Siau ja Rossi (2011) huomauttavat, että menetelmiä tukevat työkalut muodostavat oman tutkimusalansa, kuten tietokoneavusteisen ohjelmistosuunnittelun (engl. Computer Aided Software Engineering, CASE) tutkimuksen, ja ne tulisi siksi erottaa menetelmien määritelmistä.

Hirschheimin, Kleinin ja Lyytisen (1995, s. 11) mukaan tekniikat, mallit, menetelmät ja työkalut käsitetään yleensä osiksi, jotka muodostavat tietojärjestelmien kehittämisen metodologian (engl. information systems development methodology; ks. esim. Avison & Fitzgerald, 2006, s. 24). Tämän näkemyksen

[1] https://www.microsoft.com/fi-fi/microsoft-365/visio/flowchart-software
[2] https://sparxsystems.com/

mukaan aiemmin mainitut ja menetelmiksi kutsutut RAD ja SSM ovat metodologioita. Brinkkemper (1996) ja Tolvanen (1998) kuitenkin huomauttavat, että termi metodologia tarkoittaa alun perin menetelmien tutkimusta ja tietojärjestelmäkehityksen teorian rakentamista, eikä sitä tulisi sekoittaa menetelmätermiin. Tästä huolimatta termejä menetelmä ja metodologia käytetään kirjallisuudessa vastavuoroisesti.

Menetelmien ja niiden osien määrittelyjä ja luokitteluja löytynee lähes yhtä paljon kuin menetelmiäkin. Tässä tutkielmassa hyödynnetään Tolvasen (1998, s. 35—42) kokoamaa menetelmätietouden (engl. method knowledge) ”simpukkamallia” (kuvio 2). Tolvanen (1998) noteeraa kirjallisuudesta löytyvät moninaiset määritelmät ja kokoaa keskeisen menetelmätietouden simpukkamallin eri kerroksiin.



KUVIO 2 Menetelmätietous (Tolvasen, 1998, s. 35 mukaan)

Tolvasen (1998) kokoamassa menetelmätietoudessa on seuraavat kerrokset:

- *Käsitteellinen rakenne* määrittelee menetelmän avainkäsitteet ja niiden väliset suhteet ja rajoitteet. Käsitteet ovat usein kohdealuekohtaisia. Rakenne vaihtelee myös formaalisuusasteen mukaan. Tyypillisesti vaatimusmäärittelyssä ja liiketoiminnan mallintamisessa käytetyt menetelmät ovat rakenteeltaan löyhempiä kuin suunnittelussa käytetyt menetelmät. Tarkkaan määritelty rakenne johtaa yhtenäisiin kuvauksiin ja prosessiin, mutta rajoittaa menetelmän mukauttamista eri käyttötilanteisiin.
- *Notaatio* (engl. notation), käsitteiden esitystapa. Samoja käsitteitä voidaan esittää useilla tavoilla. Menetelmät käyttävät usein eri formaalisuusasteiden notaatioita eri vaiheissa tietojärjestelmäkehitystä, yleensä aloittaen epäformaaleista, vapaamuotoisista kuvauksista ja päättyen formaaliin, määrämuotoiseen esitystapaan (Pohl, 1993; Pohl & Ulfat-Bunyadi, 2013).

Jotkut notaatiot ovat sidoksissa tiettyyn menetelmään, jotkut taas ovat menetelmistä riippumattomia (van Vliet, 2007, s. 250). Seuraavassa luvussa 2.5.1 käsitellään tarkemmin notaatiota.

- *Prosessi*. Menetelmän ohjeet tietojärjestelmän kehitysprojektin läpiviemiseksi. Sisältää yleensä mallintamisen, suunnittelun, organisoinnin ja johtamisen prosesseja.
- *Osallistuminen ja roolit*. Tietojärjestelmän kehittämistyöhön liittyvät organisatoriset rakenteet, roolit ja vastuut.
- *Kehitystavoitteet ja -päätökset*. Kehitystavoitteissa kuvataan menetelmän suosimat kehitysratkaisut ja päätöksissä konkreettiset ohjeet tavoitteiden saavuttamiseksi.
- *Arvot ja oletukset*. Menetelmän taustalla oleva maailmankuva ja siihen liittyvät, usein implisiittiset, oletukset.

”Täydellinen” menetelmä sisältää kaikki kerrokset simpukkamallista. Jokainen menetelmä kuitenkin pohjautuu käsitteelliseen rakenteeseen, jota käytetään mallinnustekniikoissa mallien luomiseen jonkin notaation mukaisesti. Suuri osa menetelmistä keskittyy ainoastaan simpukkamallin kahteen sisimpään kerrokseen, jotka muodostavat mallintamisessa käytettävän kielen. (Tolvanen, 1998, s. 36.) Fowlerin ja Scottin (2000) mukaan menetelmien prosessivaiheet ovat usein melko ylimalkaisia, eikä niitä käytännössä useinkaan noudateta. Tolvasen (1998) tapaan Fowler ja Scott (2000) painottavat mallinnuskielen olevan menetelmän tärkein osa.

Tässä tutkielmassa käsitellään mallin esitystä ja sen merkitystä käytännön mallintamistyössä, joten menetelmien osalta keskitytään ainoastaan Tolvasen (1998) simpukkamallin kahteen sisimpään kerrokseen ja niistä muodostuviin kieliin. Tarkastelun ulkopuolelle jätetään muut menetelmätietouden kerrokset. Mallinnuskielistä käytetään jatkossa myös termejä mallinnustekniikka ja kaaviotekniikka.

2.5.1 Mallinnuskielet

Luonnollisten kielten tavoin myös mallinnuskielet noudattavat kielioppisääntöjä, jotka määrittävät miten käsitteet voidaan yhdistää merkityksellisiksi lausekkeiksi mallinnettavasta kohdealueesta (Gemino & Wand, 2004; Siau & Rossi, 2011; Wand ym., 1995). Mallinnuskieli sisältää konstruktioita (engl. modeling construct), jotka määrittävät kielen sanaston (Siau & Rossi, 2011) ja jotka esitetään usein graafisilla symboleilla (Gemino & Wand, 2004). Konstruktiot ovat käsitteitä, ideoita ja kuvia, joiden tarkoituksena on organisoida ja esittää tietämystä kohteesta (Siau & Rossi, 2011). Tyypillisiä käsitteitä ovat kohteet, suhteet, toiminnot, prosessit ja oliot (Wand ym., 1995). Esimerkiksi ER-kaavion pääkäsiteitä ovat kohteet, ominaisuudet ja suhteet; UML:n käyttötapauskaavion toimijat (engl. actor), käyttötapaukset ja suhteet (Siau & Rossi, 2011). Käsitteiden yhdistämistapa määrittää rajat sille, mitä kielellä voi mallintaa (Hirschheim ym., 1995).

Mallinnuskielet voidaan niiden sisältämien konstruktioiden mukaan jakaa yleiskäyttöisiin (engl. GPML, general-purpose modeling language) ja sovellusaluekohtaisiin kieliin (engl. DSL, domain-specific language; DSML, domain-specific modeling language). Yleiskäyttöisen kielen, kuten UML:n, konstruktiot ovat geneerisiä, eivätkä rajaa sen käyttöä tietylle kohdealueelle, vaan tukevat mahdollisimman laajamittaista käyttöä eri tarkoituksiin. (Silva, 2015.) Sovellusaluekohtaiset kielet sisältävät tiettyä kohdealuetta, kuten avioniikkaa, finanssialaa, videopelejä tai lääketieteen sulautettuja järjestelmiä, varten suunniteltuja konstruktioita (Silva, 2015; Pressman, 2005, s. 211). Niiden esitetään yleiskäyttöisiin kieliin verrattuna tarjoavan sovellusaluekohtaisten ohjelmointikielten tavoin helppokäyttöisyyttä ja tuottavuushyötyjä korkeamman abstraktiotason ansiosta. Sovellusaluekohtaiset kielet mahdollistavat myös koodigeneroinnin, joka on keskeistä mallipohjaisessa kehittämisessä. (Silva, 2015; Kelly & Tolvanen, 2008; Mernik, Heering & Sloane, 2005; Tolvanen & Kelly, 2016.)

Mallinnuskieli määritetään tarkemmin sen *syntaksin* (engl. syntax) ja *semantiikan* (engl. semantics) mukaan. Syntaksi sisältää symbolit eli mallinnuksen konstruktiot ja säännöt näiden symbolien yhdistämiselle. Kielen semantiikka määrittää jokaisen symbolin merkityksen ja niiden sallitut yhdistelmät. (Pohl, 2010, s. 370.) Semantiikka voidaan jakaa staattiseen ja dynaamiseen semantiikkaan. Staattinen semantiikka määrittelee kuinka konstruktio tulisi yhdistää toiseen konstruktioon ollakseen tarkoituksenmukainen ja dynaaminen semantiikka määrittää ”hyvin muodostetun” konstruktion merkityksen. (Leppänen, 2005, s. 97.)

Semantiikan tavoin myös syntaksi koostuu kahdesta osasta, abstraktista ja konkreettisesta syntaksista (Leppänen, 2005, s. 96; Pohl, 2010, s. 370). Abstrakti syntaksi määrittelee kielen käsitteelliset komponentit ja säännöt niiden yhdistämiselle, mutta ei määrää kuvaustapaa niille (ter Hofstede & Proper, 1998, s. 520; Pohl, 2010, s. 370). Konkreettinen syntaksi, jota kutsutaan myös visuaaliseksi syntaksiksi (ks. esim. Moody, 2009), määrittää kielen sanaston symbolit ja säännöt niiden ja abstraktin syntaksin käsitteiden yhdistämiselle (Leppänen, 2005, s. 96; Pohl, 2010; s. 371). Nämä konkreettiset kuvaukset muodostavat mallinnuskielen notaation eli kuvaustavan. Abstraktin ja konkreettisen syntaksin erottaminen mahdollistaa erilaisten notaatioiden kohdistamisen kielen abstraktille syntaksille. (Pohl, 2010, s. 371.) Esimerkiksi ER-kaaviolle on Chenin (1976) notaation lisäksi muitakin notaatioita, kuten SSADM-notaatio (ks. Weaver, Lambrou & Walkley, 1998), jossa hyödynnetään Everestin (1976) esittelemää Crow’s Foot -notaatiota.

Kielet voidaan jakaa niiden syntaksin ja semantiikan formaalisuusasteen mukaan epäformaaleihin, puoliformaaleihin ja formaaleihin kieliin (Pohl, 1993; Pohl & Rupp, 2015). Jokaisella formaalisuusasteella on omat etunsa. Epäformaalit kielet, kuten luonnolliset kielet, ovat ilmaisuvoimaisia ja käyttäjäystävällisiä. (Pohl, 1993.) Puoliformaalien kielten syntaksi on tarkkaan määritelty (Krogstie, Lindland & Sindre, 1995), mutta niiden väljä semantiikka mahdollistaa joustavan käytön (Latella, Majzik & Massink, 1999). UML ja Business Process Model and Notation (BPMN, ks. OMG, 2014) ovat tunnettuja puoliformaaleja kieliä.

Formaalien kielten tarkkaan määritelty syntaksi ja semantiikka estävät kuvausten monitulkintaisuuden ja näin ollen vähentävät väärinymmärrysten mahdollisuutta (Pohl, 1993; Pohl & Ulfat-Bunyadi, 2013; Pressman, 2005, s. 803). Formaalit kielet, kuten Vienna Development Method (ks. esim. Bjrner & Jones, 1978 tai Fitzgerald, Larsen & Verhoef, 2007) ja Z (ks. esim. Spivey & Abrial, 1992) perustuvat matemaattiseen logiikkaan ja ovat käännettävissä suoraan ohjelmakoodiksi (van Vliet, 2007, s. 249). Tässä tutkielmassa keskitytään puoliformaaleihin mallinnuskieliin.

2.5.2 Menetelmäkehitys

Mallintamisen alalla on tyypillistä kehittää uusia tekniikoita ja menetelmiä, tai uusia versioita jo olemassa olevista menetelmistä (ter Hofstede & van der Weide, 1992; Fettke, 2009). Itse asiassa koko tietojärjestelmäkehitys on ollut paljolti riippuvainen erilaisista menetelmistä, tekniikoista, prosesseista ja työkaluista. *Menetelmäkehitys* (engl. method engineering) on tietojärjestelmien kehittämisessä käytettävien menetelmien, tekniikoiden ja työkalujen suunnittelua, rakentamista ja soveltamista (Brinkkemper, 1996; Siau, 1999). Menetelmäkehitystä ajaa ajatus siitä, että yksikään menetelmä ei sovellu kaikkiin kehittämistilanteisiin, ja uusien menetelmien kehittämisen lisäksi siihen liittyy olemassa olevien menetelmien räätälöintiä organisaatioiden ja/tai projektien tarpeisiin (Leppänen, 2005, s. 442).

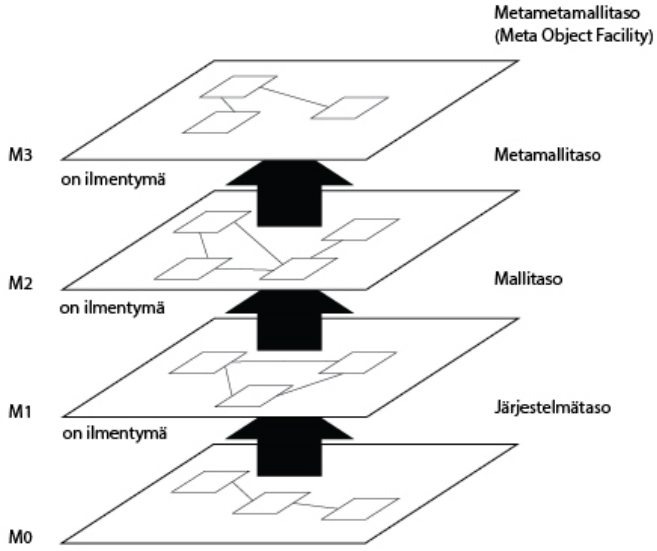
Menetelmäkehitys on aina ollut aktiivinen tutkimusala tietojärjestelmätieteissä, eikä ihme, sillä tutkijat painottavat menetelmän merkitystä kehitystyössä. Siau (2002; 2004) sanoo, että mallintamismenetelmät ovat avainasemassa tietojärjestelmien kehittämisprojektien onnistumisessa. Wyssusek (2006) ja Sommerville (2016, s. 18) ovat samoilla linjoilla kertoessaan, että ongelmat tietojärjestelmäkehityksessä voidaan usein kohdistaa puutteellisten tai riittämättömien menetelmien käyttöön. Alalla kehitettyjen menetelmien määrä on valtava ja johtanut jo 1990-luvulla ”menetelmäviidakon” syntyyn (ks. ter Hofstede & van der Weide, 1992), mutta teknologioiden kehittyessä myös mallintamismenetelmät jatkavat edelleen kehityskulkuaan (Sinz, 2019).

Metamallintaminen (engl. meta modeling) on tieteenala, joka tutkii mallien malleja eli *metamalleja* (engl. meta model) ja niiden tuottamiseen käytettäviä kieliä ja toimintatapoja (Leppänen, 2005, s. 57). Menetelmäkehityksen ja metamallintamisen raja on häilyvä, mutta edellinen on suunnittelupainotteisempi, tähdäten jatkuvasti parempien menetelmien kehittämiseen huomioiden tarvittaessa myös muut menetelmätietouden kerrokset (ks. Tolvanen, 1998), ja jälkimmäinen keskittyy enemmän mallintamiseen, ja kieliin ja niiden vertailuun (Siau & Rossi, 2001).

Metamalli on spesifikaatio mallinnuskielen abstraktista syntaksista (Guizzardi, 2007; Leppänen, 2005, s. 57). Metamallin abstrakti syntaksi määrittää mallinnuskielen käytölle sallitut konstruktiot ja nk. hyvinmuodostuneisuussäännöt (engl. well-formedness rules) konstruktioiden yhdistämiselle, jotta mallit ovat kieliopillisesti paikkansapitäviä (Guizzardi, 2007). Metamalli määrittää siis ta-

van, jolla reaalimaailman kohdetta tarkastellaan ja kuvataan (Falkenberg ym., 1998, s. 153). Metamalli esitetään kielellä, jota kutsutaan metamallinnuskieleksi tai metakieleksi (Leppänen, 2005, s. 57; Pohl, 2010, s. 372). Metakielen abstrakti syntaksi esitetään metametamallissa (Leppänen, 2005, s. 57) ja se tuotetaan metametakielellä (Pohl, 2010, s. 372).

Metatasoja voidaan teoriassa jatkaa loputtomiin, mutta käytännössä metamallihierarkia suljetaan jollain tasolla (Pohl, 2010, s. 372). Neljä tasoa on yleensä riittävä (OMG, 2016, s. 6). Object Management Groupin (OMG) metamalli, johon myös UML perustuu, on tunnettu esimerkki perinteisestä nelitasoisesta metamallihierarkiasta (engl. four-layer metamodel hierarchy) (kuvio 3), ja sen pohjalta voidaan kehittää myös uusia mallinnuskieliä ja -standardeja (Atkinson & Kühne, 2003). Nelitasoinen metamallihierarkia suljetaan rekursion avulla tasolle M3, jossa metamalli määräytyy metakielellä Meta Object Facility (MOF). Rekursiolla sulkeminen tarkoittaa, että metakieli tasolla M3 on itsemääräytyvä eli se on tarpeeksi ilmaisuvoimainen määrittääkseen omat sääntönsä (ks. Falkenberg ym., 1998, s. 58).



KUVIO 3 OMG:n nelitasoinen metamallihierarkia (Pohlin, 2010, s. 373 mukaan)

Nelitasoisessa metamallissa jokainen taso ylintä lukuun ottamatta on ilmentymä (engl. instance of) ylemmästä tasosta (Atkinson & Kühne, 2003). Alimman tason nimi ”M0” viittaa jo siihen, että taso ei ole mallitaso, vaan vastaa konkreettisia kohteita ja on siksi nimetty kuvassa järjestelmätasoksi. Ylempi taso M1, on malli tästä järjestelmästä, esimerkiksi UML:n luokkakaavio, jonka luokkien ilmentymistä järjestelmätaso muodostuu. Taso M2 eli metamalli määrittää kielen, jolla M1-tason malli kuvaillaan. Se on esimerkiksi luokkakaavion metamalli, jonka ilmentymiä alemman tason mallit eli luokkakaaviot ovat. Tasolla M3 on metametamalli, jonka ilmentymiä ovat metamallitason kaaviot ja joka kuvaa kielen, jolla metamalli määritetään. (Koskimies ym., 2004; Pohl, 2010, s. 372.) Seuraavassa pääluvussa käsitellään nelitasoiseen metamallihierarkiaan perustuvaa UML-mallinnuskieltä.

3 MALLINTAMINEN KÄYTÄNNÖSSÄ

Kirjallisuudessa esitettyjä näkemyksiä mallintamisen merkitsevyydestä (ks. esim. Booch ym., 2005; Frank, 1999) ei välttämättä huomioida käytännön mallintamistyössä (Seidewitz, 2003). Alalla yleistynyt ketterä ohjelmistokehitys (ks. Beck ym., 2001) korostaa muutosvalmiutta suunnitelmien noudattamisen sijaan ja toimivan ohjelmiston tuottamista dokumentaation sijaan (Beck ym., 2001), minkä voi ymmärtää jopa vastakkaisena suuntauksena mallintamiselle. Ketteryys on mullistanut ohjelmistotuotannon ja tehnyt siitä asiakkaiden muuttuviin vaatimuksiin sopeutuvaista. Projektien ongelmat voidaan kuitenkin usein jäljittää analyysin vaatimusmäärittelyyn (Laplante, 2018; Pressman & Maxim, 2015; Siau, 2002), jossa mallintamisen rooli on perinteisesti ollut keskeinen (Gemino & Wand, 2004), ja johon ketterällä kehityksellä on ollut suuri vaikutus.

Tässä pääluvussa tarkastellaan mallintamisen käytänteitä tietojärjestelmien kehittämisessä. Onko ketteryys ajanut mallintamisen alas vai pidetäänkö projektien onnistumista riippuvaisena mallintamisesta? Mikä mahtaa olla vuosikymmenien saatossa kehitettyjen erilaisten mallintamismenetelmien rooli käytännön mallintamistyössä? Ensimmäisessä alaluvussa esitellään ohjelmistokehityksen tunnetuin mallinnuskieli, alan standardi UML (Unified Modeling Language; ks. Booch ym., 1996; OMG, 2017), jonka sanotaan olevan laajassa käytössä (ks. esim. Rumpe, 2016). Toisessa alaluvussa käsitellään empiirisiä tutkimuksia käytännön mallintamistyöstä 2000-luvulla.

3.1 UML (Unified Modeling Language)

UML on yleiskäyttöinen ja graafinen mallinnuskieli, jonka avulla voidaan määrittää, visualisoida, toteuttaa ja dokumentoida tietojärjestelmien kehittämisessä käytettäviä malleja (OMG, 2017; Siegel, 2005). Kuten useimmat kielet ja menetelmät, myöskään UML ei perustu mihinkään teoriaan (Siau & Rossi, 2011), vaan se on koottu ohjelmistokehityksen parhaista käytännöistä (OMG, 2017). UML on menetelmäriippumaton (Siegel, 2005), mutta sille on erikseen kehitetty

myös toimintaohjeet käyttöä varten, the Unified Software Development Process (ks. Jacobson, Booch & Rumbaugh, 1999) tai lyhyemmin Unified Process (UP), ja myös ketterä versio Agile Unified Process (AUP, ks. Ambler, 2006).

UML on oliosuuntautunut (engl. object oriented) kieli ja sisältää vahvan notaation oliosuuntautuneiden järjestelmien mallintamista ja suunnittelua varten (Pressman, 2005, s. 63). UML:ää kehittävä yrityskonsortio The Object Management Group (OMG) kuitenkin painottaa, että UML:llä voi mallintaa mitä tahansa sovellusta mille tahansa laitteistolle, käyttöjärjestelmälle, ohjelmointikielelle tai verkolle, ja myös liiketoimintaa tai muita kohteita tietojärjestelmien ulkopuolelta (Siegel, 2005). UML myös tukee alusta- ja sovellusaluekeskeisiä laajennoksia (Lange ym., 2006; OMG, 2017, s. 252) ja mallipohjaista toteuttamista (Mellor, Mellor & Balcer, 2002).

UML:ää kutsutaan kieleksi, mutta todellisuudessa se on kokoelma mallinnuskieliä, joista jokainen on tarkoitettu tietyn ohjelmistoihin liittyvän näkökulman kuvaamiseen (Koskimies ym., 2004). UML:ään kuuluvia mallinnuskieliä kutsutaan kaaviotyypeiksi ja usean näkymän lisäksi ne mahdollistavat mallintamisen usealla abstraktiotasolla (Lange ym., 2006). UML:n spesifikaatio on ollut jatkuvassa kehityksessä ensimmäisen UML 1.0 -version jälkeen. Vuonna 2005 standardoitu versio UML 2.0 oli huomattava uudistus, ja sen myötä kielen kaavioiden määrä kasvoi yhdeksästä kolmeentoista (OMG, 2005, s. 660). Version UML 2.0 jälkeen on julkaistu vielä useita versioita, jotka sisältävät pienimuotoisempia muutoksia. Viimeisin versio UML 2.5.1 on vuodelta 2017 ja se sisältää neljätoista eri kaaviotyyppiä (OMG, 2017).

3.1.1 UML:n historiaa

Oliosuuntautuneet menetelmät saivat alkunsa 1970- ja 1980-lukujen taitteessa. Oliokielet ja alati monimutkaistuvat järjestelmät innostivat menetelmäkehittäjiä etsimään vaihtoehtoisia lähestymistapoja analyysiin ja suunnitteluun. (Booch ym., 1999.) Tätä ennen dataperustainen rakenteinen mallintaminen (engl. structured modeling) oli tärkein mallinnusparadigma. Rakenteisen lähestymistavan keskeisin ominaisuus on tietovirta (Siau, 2004) ja käytetyimpiä kaaviotyyppejä ER-kaavio, tietovirtakaavio ja rakennekaavio (engl. structure diagram, ks. esim. Jackson, 1983). Oliosuuntautuneessa lähestymistavassa keskiössä ovat oliot, jotka ovat järjestelmässä vuorovaikutuksessa toistensa kanssa luontaisen käyttäytymisensä kautta. Oliot mahdollistavat reaalimaailman ymmärryksen ja käytännöllisen pohjan luomisen toteutusta varten. (Rumbaugh, Blaha, Premerlani, Eddy & Lorensen, 1991.)

1990-luvun alkuun mennessä olioparadigma oli saavuttanut suuren suosion. Käyttöön otettiin lukuisia oliosuuntautuneita analyysi- ja suunnittelumenetelmiä, joiden ylivertaisuudesta näiden kannattajat kiistelivät. (Pressman, 2005, s. 63.) ”Menetelmäsodan” päättämiseksi OMG alkoi etsiä oliomallinnusstandardia (Booch ym., 1999). Tähän tarpeeseen vastasivat the Rational Corporationin menetelmäkehittäjät Grady Booch, Ivar Jacobson ja James Rumbaugh. He yhdistivät 1990-luvun alussa parhaat ominaisuudet omista oliosuuntautuneista

menetelmistään (Booch-menetelmä, ks. Booch, 1994; OOSE-menetelmä, ks. Jacobson ym., 1992; OMT-menetelmä, ks. Rumbaugh ym., 1991) ja lisäsivät niihin muiden asiantuntijoiden (ks. esim. Harel, 1987; Wirfs-Brock, Wilkerson & Wiener, 1990) ehdottamia ominaisuuksia. Lopputuloksena syntyi UML. (Pressman, 2005, s. 63.)

OMG jatkoi UML:n kehittämistä ja standardoi sen vuonna 1997 (Liddle, 2011; Pressman, 2005, s. 63). Samaan aikaan the Rational Corporation ja muut yritykset kehittivät automatisoituja työkaluja tukemaan UML:n käyttöä (Pressman, 2005, s. 63). UML sai ISO-standardin vuonna 2005 (ks. ISO, 2005; OMG, 2019) ja on vakiinnuttanut asemansa alalla (Elaasar ym., 2018). Nykyään UML:n käyttöä tukee laaja kirjo erilaisia työkaluja kaupallisista opetusvälineisiin ja avoimen lähdekoodin ohjelmistoihin (Elaasar ym., 2018). Kaupallisista työkaluista tunnettu esimerkki on Sparx Systemsin Enterprise Architect. Opetus- ja avoimen lähdekoodin työkaluja ovat mm. Umple[3], UMLet[4] ja PlantUML[5]. Myös yleisillä piirtotyökaluilla, kuten Microsoftin Visiolla, voidaan tuottaa UML-kaavioita.[6] UML:ää ja sitä tukevia työkaluja on kuitenkin kritisoitu liiasta monimutkaisuudesta (ks. esim. Dori, 2002; Elaasar ym., 2018; Kobryn, 2002; Recker ym., 2009), semantiikan epäjohdonmukaisuudesta ja ympäripyöreydestä (ks. esim. Dori, 2002; Siau & Loo, 2006), ja toisaalta myös joustavuuden puutteesta tietyissä sovellusalueissa (ks. esim. Duddy, 2002).

Boochin, Jacobsonin ja Rumbaugh'n julkaisujen lisäksi UML:stä on kirjoitettu runsaasti muitakin kirjoja (ks. esim. Bennett, McRobb & Farmer, 2006; Fowler, 2003; Lano, 2009; Pilone & Pitman, 2005) ja kielen opetus kuuluu ympäri maailman korkeakoulujen opetusohjelmiin. UML:n käyttöä on pohdittu useasta näkökulmasta ja tunnetuin niistä on mahdollisesti Fowlerin (2003) näkemys. Fowlerin (2003) mukaan UML:ää käytetään kolmella eri tavalla, joista yleisin on luonnostelu (engl. UML as sketch), jossa kehittäjät käyttävät UML:ää viestiessään jotain näkökulmaa järjestelmästä. Kaksi muuta tapaa, UML:n käyttö toimintasuunnitelmana (engl. UML as blueprint) tai ohjelmointikielenä (engl. UML as programming language), ovat Fowlerin (2003) mukaan harvinaisempia. Ne myös edellyttävät luonnostelusta poiketen UML:n käyttöä kokonaisuudessaan. Näitä käyttötapoja on käsitelty kirjallisuudessa myös myöhemmin (ks. esim. Chaudron ym., 2012; Störrle, 2017).

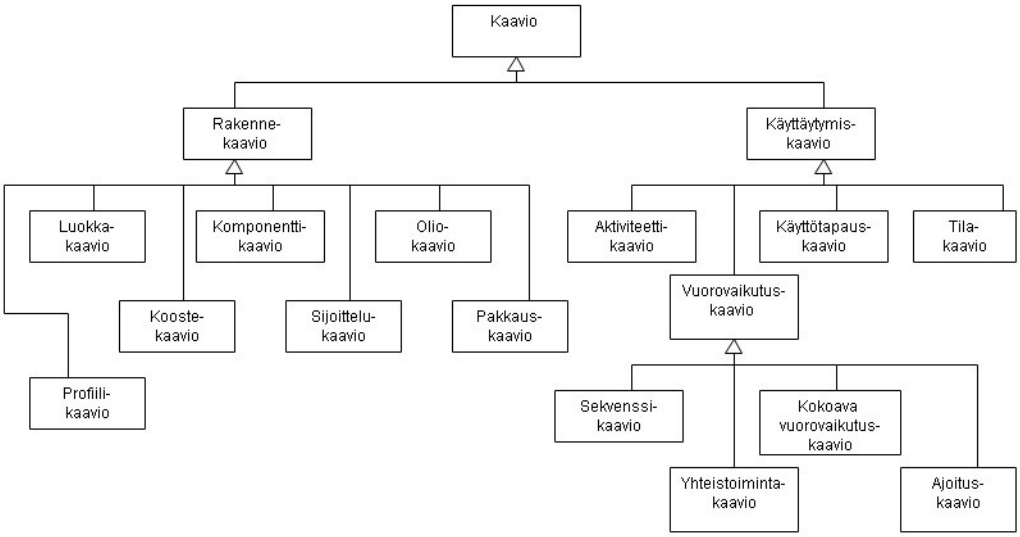
3.1.2 UML:n 14 kaaviotyyppiä

Järjestelmän malli esitetään UML:ssä erilaisten kaavioiden avulla. UML sisältää neljätoista kaaviotyyppiä eli osakieltä, joista seitsemän edustaa järjestelmän staattista rakennetta, kolme yleistä käyttäytymistä ja neljä erilaisia vuorovaikutuksen aspekteja (Siegel, 2005; OMG, 2017, s. 685):

[3] https://cruise.umple.org/umple/
[4] https://www.umlet.com/
[5] https://plantuml.com/
[6] Kuratoitu lista UML-työkaluista: https://modeling-languages.com/uml-tools/

- *Rakennekaaviot* eli luokkakaavio, oliokaavio (engl. object diagram), komponenttikaavio, koostekaavio (engl. composite structure diagram), pakkauskaavio (engl. package diagram) sijoittelukaavio ja profiilikaavio (engl. profile diagram);
- *Käyttäytymiskaaviot* eli käyttötapauskaavio, aktiviteettikaavio ja tilakaavio;
- *Vuorovaikutuskaaviot* eli sekvenssikaavio, yhteistoimintakaavio (engl. communication diagram), ajoituskaavio (engl. timing diagram) ja kokoava vuorovaikutuskaavio (engl. interaction overview diagram) (OMG, 2017, s. 685).

Tietojärjestelmää mallinnettaessa on huomioitava sekä järjestelmän rakenne että sen toiminnan aikainen käyttäytyminen. Nämä näkökulmat täydentävät toisiaan ja ovat toisistaan riippuvaisia. (Dori, 2011.) Näkökulmat huomioon ottaen UML:n kaaviotyypit on jaettu kahteen pääryhmään, rakennetta ja käyttäytymistä kuvaaviin (OMG, 2017, s. 685). Vuorovaikutuskaaviot kuvaavat myös järjestelmän käyttäytymistä, mutta ne erotetaan käyttäytymiskaavioista omaksi ryhmäkseen (Siegel, 2005). Kuviossa 4 esitetään UML 2.5.1-spesifikaation (OMG, 2017, s. 685) mukainen kaaviotyyppien jaottelu.



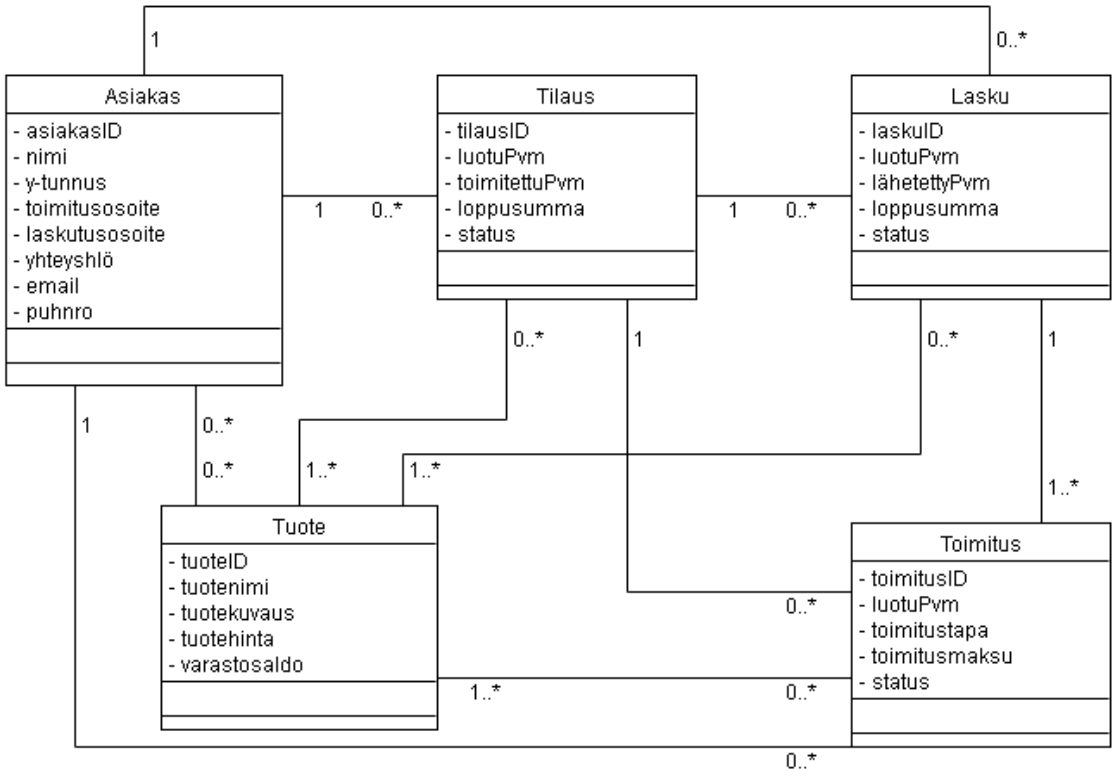
KUVIO 4 UML:n kaaviotyypit (OMG, 2017, s. 685 mukaan)

Rakennekaaviot kuvaavat järjestelmän olioiden staattista, ajasta riippumatonta rakennetta ja järjestelmän käsitteitä, jotka voivat olla abstrakteja, reaalimaailmaan tai toteutukseen liittyviä. Ne eivät sisällä dynaamisen käyttäytymisen yksityiskohtia, jotka kuvataan käyttäytymiskaavioissa. Käyttäytymiskaaviot kuvaavat järjestelmän olioiden dynaamista käyttäytymistä, sisältäen niiden yhteistoiminnan, tilahistorian, metodit ja aktiviteetit. Dynaaminen käyttäytyminen voidaan määritellä joukoksi järjestelmän muutoksia ajan mittaan. (OMG, 2017, s. 685.)

UML:n kaaviot koostuvat mallinnuselementeistä (engl. model element), jotka on jaettu kolmeen pääryhmään: luokittelija (engl. classifier), tapahtuma ja

käyttäytyminen. (OMG, 2017, s. 12.) Elementit vastaavat yleisiä olio-ohjelmoinnin käsitteitä, kuten luokkia, operaatioita ja viestejä (Siegel, 2005). Jokaista elementtiä vastaa graafinen symboli, esimerkiksi luokkia kuvataan suorakolmioilla ja suhteita erilaisilla suhdeviivoilla (Lano, 2009, s. 1). UML ei määrittele symboleille täysin tarkkaa ulkoasua, joten eri työkaluilla voidaan saada aikaiseksi hieman erinäköisiä kaavioita (OMG, 2017, s. 17).

Luokkakaavio on keskeisin väline järjestelmän staattisen rakenteen mallintamisessa ja muodostaa pohjan muille kuvaustavoille (Rumpe, 2016). Sillä voidaan dokumentoida järjestelmää usealla eri abstraktiotasolla tietojärjestelmäkehityksen kaikissa vaiheissa (Pohl, 2010, s. 231). Tyypillisesti sitä käytetään tiedon mallintamisessa, arkkitehtuurisuunnittelussa ja yksityiskohtaisessa suunnittelussa (Koskimies ym., 2004). Luokkakaaviolla kuvataan järjestelmään kuuluvia luokkia ja niiden välisiä suhteita: assosiaatioita (engl. association), yleistyksiä (engl. generalization) ja erilaisia riippuvuuksia (engl. dependency). Luokka on kuvaus sovellusalueen tai ratkaisun käsitteestä, ja se toimii kaavion keskipisteenä, johon muut elementit kiinnitetään. (Booch ym., 1999.) Kuviossa 5 esitetään kuvitteellisen ja yksinkertaistetun tilaus- ja toimitusjärjestelmän tietomalli luokkakaaviona. Kaavio on tehty UMLetinolla[7], UMLet-työkalun verkkoversiolla.

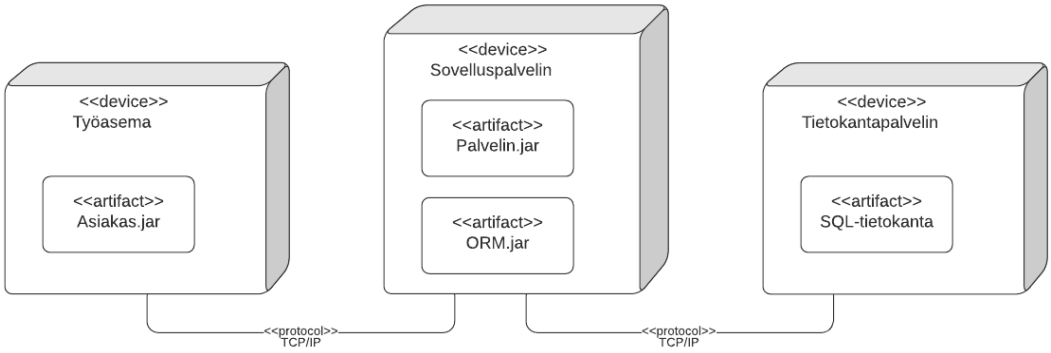


KUVIO 5 Esimerkki luokkakaaviosta

[7] http://www.umletino.com/

Rakennekaavioihin lukeutuvalla *oliokaaviolla* kuvataan olioita ja niiden välisiä suhteita tiettynä ajankohtana järjestelmän elinkaaressa (Rumpe, 2016, s. 104). Oliokaavio on siten luokkakaavion ajonaikainen ilmentymä ja täydentää sitä esimerkinomaisesti, eikä sillä voida kuvata koko järjestelmää (Kästner, Gogolla & Selic, 2018; Rumpe, 2016, s. 104). *Koostekaavio* on luokkakaavio, joka sisältää olioita tai komponentteja. *Komponenttikaaviolla* kuvataan komponentit ja niiden väliset suhteet. *Pakkauskaavio* kuvaa järjestelmän pakkausten (yleensä alijärjestelmien) väliset riippuvuussuhteet. *Sijoittelukaavio* kuvaa järjestelmän laitteisto-osat eli solmut (engl. node), ohjelmisto-osien eli artefaktien sijoittumisen laitteistoille ja kommunikointiväylät osien välillä. Uusi kaaviotyyppi *profiilikaavio* mahdollistaa laajennosmekanismien, kuten stereotyyppien (engl. stereotype) avulla alusta- ja sovellusaluekeskeiset laajennokset. (Koskimies ym., 2004; OMG, 2017.)

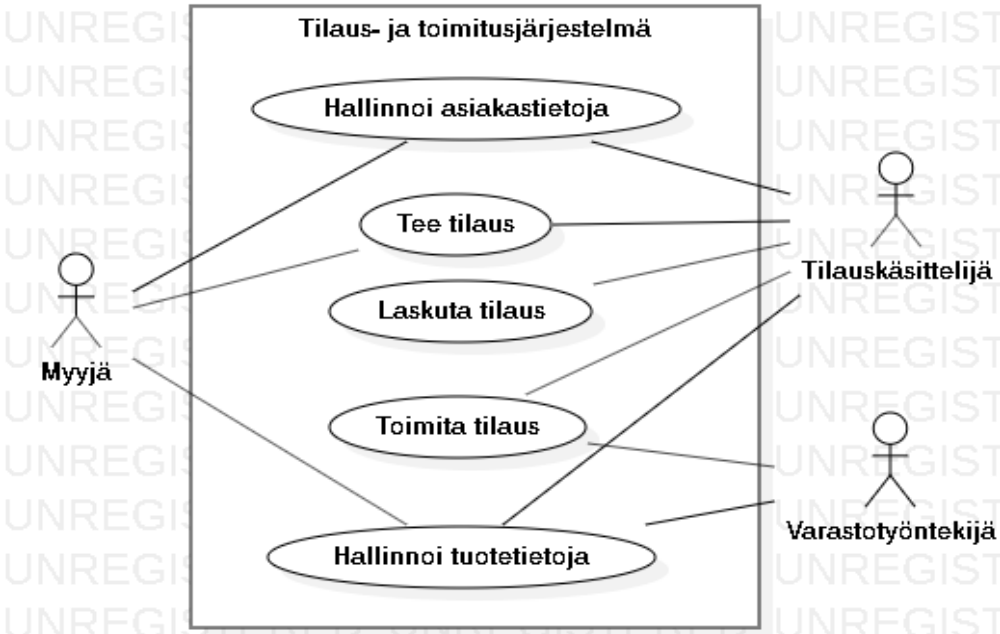
Kuviossa 6 on yksinkertainen esimerkki sijoittelukaaviosta. Kaavio on tehty LucidChartin[8] rajatulla ilmaisversiolla.



KUVIO 6 Esimerkki sijoittelukaaviosta

Käyttäytymiskaavioita voidaan luokkakaavioiden tapaan soveltaa tietojärjestelmäkehityksen eri vaiheissa. *Käyttötapauskaavion* avulla tunnistetaan, miten järjestelmää käytetään ja mitkä ulkoiset toimijat, kuten ihmiskäyttäjät, osallistuvat tähän käyttöön. (Lano, 2009, s. 10). Käyttötapauskaaviolla voidaan kuvata kaikki käyttötapaukset ja niiden suhteet, eli tarjota järjestelmän toiminnallisuuksista yleiskuva (Pressman & Maxim, 2015, s. 875). Kaavion notaatiota pidetään yksinkertaisena (ks. esim. Gemino & Parker, 2009; Haikala & Mikkonen, 2011, s. 77; Lano, 2009, s. 10; Lilly, 2000) ja sen esitetäänkin lisäävän ymmärrystä järjestelmästä ja edesauttavan viestintää eri sidosryhmien kesken (ks. Gemino & Parker, 2009; Haikala & Mikkonen, 2011, s. 81). Kuviossa 7 esitetään korkean tason käyttötapauskaavio, joka on tehty MKLabsin StarUML-työkalun[9] ilmaisella kokeiluversiolla.

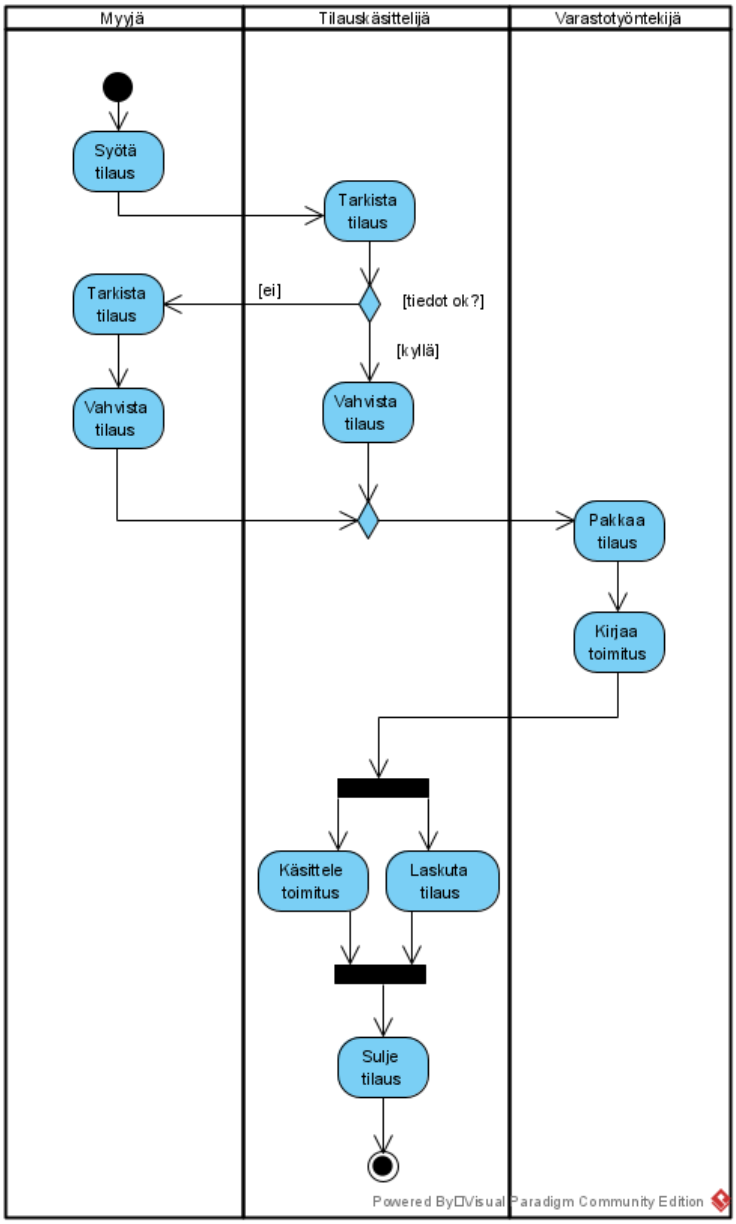
[8] https://www.lucidchart.com/pages/
[9] https://staruml.io/



KUVIO 7 Esimerkki käyttötapauskaaviosta

Aktiviteettikaaviolla voidaan tarkentaa käyttötapauksia ja keskeisiä operaatioita sekä kuvata järjestelmässä tapahtuvaa tiedon prosessointia (Koskimies ym., 2004; Lano, 2009, s. 22). Kaaviotyyppi mahdollistaa useasta aktiviteetista koostuvan käyttäytymisen kuvaamisen ja sitä käytetäänkin usein liiketoimintaprosessien kuvaamiseen (André, Choppy & Reggio, 2014; Lano, 2009, s. 22). Kuviossa 8 esitetään aktiviteettikaavio tilaus- ja toimitusjärjestelmään liittyvistä käyttötapauksista. Aktiviteettien suorittajat on osoitettu jakamalla kaavio vyöhykkeisiin tai ”uimaratoihin” (engl. partition, swimlane, ks. OMG, 2017 s. 408—410), joiden avulla kuvataan tehtävien siirtyminen eri toimijoiden välillä. Kaavio on tehty OMG:n Visual Paradigm -työkalun[10] ilmaisversiolla.

[10] https://www.visual-paradigm.com/



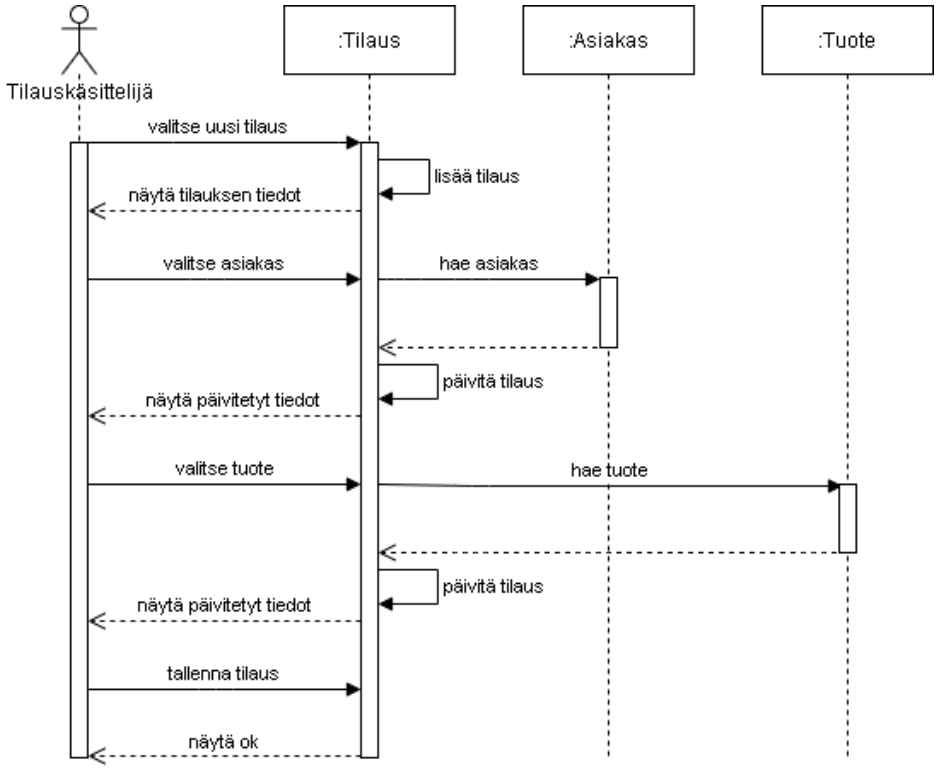
KUVIO 8 Esimerkki aktiviteettikaaviosta

Tilakaaviolla kuvataan olion käyttäytymistä siirtyminä tilasta toiseen. Tilakaavio esittää mahdollisten tilojen lisäksi tapahtumat, jotka aiheuttavat siirtymisen tilojen välillä. (Koskimies ym., 2004; Lano, 2009, s. 11.)

Vuorovaikutuskaavioista *sekvenssikaavio* kuvaa viestienvaihtoa tiettyjen osapuolien, kuten esimerkiksi olioiden välillä (Koskimies ym., 2004). Sekvenssikaavio on oliokaavion tapaan esimerkinomainen, ja sen kuvaama vuorovaikutus tai toiminto voi siten esiintyä usein, useita kertoja rinnakkain tai ei välttämättä kertaakaan järjestelmän prosessissa (Rumpe, 2016, s. 192). *Yhteistoimintakaaviolla* voidaan esittää olioiden vuorovaikutusta sekvenssikaavion tapaan. *Kokoava vuorovaikutuskaavio* ja *ajoituskaavio* ovat UML:ään lisätyistä kaaviotyypeistä uusimpien joukossa. Kokoava vuorovaikutuskaavio on aktiviteettikaavion muunnelma ja sillä voidaan kuvata esimerkiksi logiikka, jolla sekvenssikaa-

viot seuraavat toisiaan. Ajoituskaaviolla kuvataan yhden tai usean olion käyttäytymisen reaaliajan suhde. (Koskimies ym., 2004.)

Kuviossa 9 esitetään analyysivaiheen korkean tason sekvenssikaavio, joka on tehty avoimen lähdekoodin piirtotyökalun Draw.io:n[11] verkkoversiolla.



KUVIO 9 Esimerkki sekvenssikaaviosta

Sekvenssi- ja luokkakaavio näyttävät edustavan hyvin erilaisia näkymiä järjestelmästä, mutta niiden välillä on selkeä yhteys (Haikala & Mikkonen, 2011, s. 102). Jos noudatetaan Fowlerin (2003) esittämistä käyttötavoista muuta kuin luonnostelua, on sekvenssikaavioista löydyttävä luokkakaavioiden sisältämät luokat ja metodit. Jos UML:ää käytetään ainoastaan kuvioiden 5 ja 9 kaltaisella korkealla tasolla, on käyttö analyysiin liittyvää ja tähtää viestinnän edesauttamiseen ja yhteisymmärryksen saavuttamiseen yksityiskohtaisen teknisen suunnittelun sijaan. Seuraavissa alaluvuissa kerrotaan empiirisistä tutkimuksista, joissa raportoidaan UML:n ja muiden mallinnustekniikoiden käyttöä.

[11] https://app.diagrams.net/
https://www.diagrams.net/
https://drawio-app.com/
https://www.draw.io/

3.2 Käytännön kokemuksia

Kirjallisuudessa toteamukset käytännön mallintamisesta vaihtelevat sen yleisyydestä (ks. esim. Chaudron ym., 2012) alhaiseen käyttöasteeseen (ks. esim. Badreddin, Lethbridge & Elassar, 2013). Selvemmän kokonaiskuvan saamiseksi tässä luvussa tarkastellaan mallintamiseen ja UML:n käyttöön keskittyviä empiirisiä tutkimuksia 2000-luvun alusta lähtien. Tätä aiemmat tutkimukset ovat todennäköisesti vanhentuneita ottaen huomioon, että UML ja ketteryys ovat vaikuttaneet merkittävästi tietojärjestelmäkehitykseen 2000-luvun alkupuolelta lähtien. Empiirisiä tutkimuksia on etsitty pääasiassa Google Scholarin hakukoneella seuraavilla hakutermeillä ja niiden erilaisilla yhdistelmillä: *conceptual model(l)ing, software model(l)ing, information systems model(l)ing, UML, unified modeling language, model(l)ing language, model(l)ing method, model(l)ing technique, diagramming technique, diagram, use, practitioner, in practice, practices* ja *state of practice*, sekä käsiteltyjen tutkimusten lähdeviittauksista.

Tarkasteltaviksi on valikoitu tutkimuksia, joissa käsitellään UML:n käyttöä, erilaisten mallinnustekniikoiden ja -työkalujen käyttöä, mallintamiseen liittyviä hyötyjä ja haasteita, ja joissa tietoa näistä käytänteistä on kerätty mallintamistyötä tekeviltä ja malleja hyödyntäviltä ammattilaisilta. Mallintamisen käytäntöjen tutkimukset keskittyvät yleensä joko UML:n käyttöön (ks. esim. Dobing & Parsons, 2006; Petre, 2013), alan mallintamiskäytäntöjen kartoittamiseen (ks. esim. Davies ym., 2006; Störrle, 2017) tai mallipohjaisen kehityksen tutkimiseen (ks. esim. Hutchinson, Rouncefield & Whittle, 2011a). Lähes kaikkien tutkimusten aineisto on hankittu verkkokyselyn avulla, merkittävänä poikkeuksena mainittakoon Petren (2013) haastattelututkimus UML:n käytöstä. Luvussa 3.2.1 valitut tutkimukset esitellään lyhyesti julkaisujärjestyksessä, jonka jälkeen niitä tarkastellaan tämän tutkielman kannalta olennaisten teemojen kautta.

Tarkastelun ulkopuolelle on jätetty tutkimukset, jotka keskittyvät kaaviotekniikoiden yleisen käytön sijaan niiden sisältämien konstruktioiden käyttöön (ks. esim. Mendling, Reijers & Recker, 2010; Reggio, Leotta, Ricca & Clerissi, 2015), UML:n sijaan muihin mallinnuskieliin (ks. esim. Recker, 2008; Recker, Indulska, Rosemann & Green, 2010) tai kehitystyössä luotavien mallien sijaan valmiiden suunnittelumallien (engl. design pattern) käyttöön (ks. esim. Kassab, Mazzara, Lee & Succi, 2018). Tarkastelematta jätetään myös tutkimukset, jotka keskittyvät ammattilaisten sijaan opiskelijoiden mallintamiskokemuksiin (ks. esim. Wrycza & Marcinkowski, 2007), laajemman otannan sijaan yhden ainoan yrityksen tai ohjelmistoprojektin puitteissa tapahtuvaan mallintamiseen (ks. esim. Anda, Hansen, Gullesen & Thorsen, 2006; Cherubini, Venolia, DeLine & Ko, 2007; Vetro, Böhm & Torchiano, 2015), tai tietojärjestelmäkehityksen sijaan koko organisaation kattavaan kokonaisarkkitehtuuriin (engl. enterprise architecture) (ks. esim. Bricknall, Darrell, Nilsson & Pessi, 2006; Roth, Hauder, Farwick, Breu & Matthes, 2013; Yu, Strohmaier & Deng, 2006) tai tiettyyn sovellusalueeseen (ks. esim. Liebel, Marko, Tichy, Leitner & Hansson, 2018).

3.2.1 Käytännön tutkimusten esittely

Tutkimukset UML:n käytöstä (taulukossa 1: UML). Grossman, Aronson ja McCarthy (2005) toteavat, että kirjallisuudessa on keskitytty lähinnä UML:n puutteisiin, eikä sen käytöstä ole empiiristä tutkimustietoa. Heidän tutkimuksessaan ammattilaisilla oli kirjavia mielipiteitä UML:stä ja vastoin kirjallisuuden asettamia ennakko-odotuksia suurin osa heistä piti sitä sopivana käyttötarkoituksiinsa. (Grossman ym., 2005.) Myös Dobing ja Parsons (2006; tarkempi raportti tutkimuksesta Dobing & Parsons, 2008) noteeraavat UML:ään liittyvän vähäisen todistusaineiston ja tutkivat analyysivaiheessa käytettäviä UML:n kaaviotyyppejä, joita tulosten mukaan hyödynnettiin erittäin vaihtelevasti.

Langen ym. (2006) tutkimus keskittyi ohjelmistoarkkitehtien UML:n käytön lisäksi mallien laatuun. Malleja analysoidessaan tutkijat huomasivat useita tapoja, joilla ammattilaiset rikkoivat UML:n sääntöjä, yhtenäisten toimintatapojen puuttuessa. Epätäydellisistä malleista seurasi kommunikointi- ja laatuongelmia, väärän tuotteen toimittamista ja lisääntynyttä testaamista. (Lange ym. 2006). Laadun lisäksi Nugroho ja Chaudron (2008) tutkivat UML:n käytön vaikutusta tuottavuuteen. Positiivinen vaikutus kohdistui laadussa etenkin ymmärrettävyyteen ja modulaarisuuteen, ja tuottavuudessa analyysi-, suunnitteluja toteuttamisvaiheisiin (Nugroho & Chaudron, 2008).

Scanniello, Gravino ja Tortora (2010) tutkivat italialaisten IT-yritysten, Fitsilis, Gerogiannis ja Anthopoulos (2013) kreikkalaisten ohjelmistoammattilaisten ja Farias ym. (2018) brasilialaisten ohjelmistoammattilaisten UML:n käyttöä**.** Scanniello ym. (2010) ja Fitsilis ym. (2013) raportoivat UML:n olevan yleisesti käytössä, kun taas Farias ym. (2018) raportoivat vähäistä käyttöä UML:n tunnettuudesta huolimatta.

Petre (2013) huomauttaa, että tieto UML:n käyttökokemuksista on tärkeää tieteenalalle ja notaatioiden sekä työkalujen kehittämiselle. Hän haastatteli 50 ohjelmistoammattilaista 50 yrityksestä ympäri maailmaa selvittäessään, miten nämä käyttivät UML:ää, jos käyttivät. Petre (2013) määritteli UML:n käytölle viisi kaavaa: ei UML:ää (haastateltavista 35/50), käyttö jälkikäteen (engl. retrofit, käytti muuta menetelmää, mutta dokumentoi jälkikäteen UML:llä joko johtoa tai asiakasta varten) (1/50), automaattinen koodigenerointi (3/50), valikoiva käyttö (11/50), ja kokonaisvaltainen, organisaationlaajuinen sitoutuminen UML:ään (0/50). Petre (2014) suoritti kymmenen lisähaastattelua, jotka vahvistivat aiemman tutkimuksen tuloksia.

Reggio, Leotta ja Ricca (2014) tutkivat, missä määrin ammattilaiset ja akateemikot tunsivat ja käyttivät UML:n kaaviotyyppejä. He vertasivat tuloksia aiempaan tutkimukseensa UML:n eri kaaviotyyppien esiintyvyydestä oppikirjoissa, kursseissa, tutoriaaleissa ja työkaluissa (ks. Reggio, Leotta, Ricca & Clerissi, 2013). Reggion ym. (2014) mukaan UML:n eri kaaviotyyppien tunnettuuden ja käytön yleisyyden välillä oli vahva korrelaatio.

Fernández-Sáezin, Caivanon, Generon ja Chaudronin (2015) tutkimuksen tavoitteena oli selvittää UML-kaavioiden käyttöä ja hyötyjä ohjelmistojen ylläpitoprojekteissa. Tulosten mukaan ammattilaiset, jotka käyttivät UML:ää, koki-

vat sen nopeuttavan ylläpitotyötä. Käyttö oli yleisempää suurissa tiimeissä ja isoa järjestelmää ylläpidettäessä. Suurin osa dokumentaatiosta sisälsi käsin tehtyjä kaavioita, kun selkeästi pienempi osa oli työkalun avulla koodipohjasta generoituja takaisinmallinnettuja (engl. reverse-engineering, RE) kaavioita. (Fernández-Sáez ym., 2015.)

Tutkimukset mallintamisesta ja mallien käytöstä (taulukossa 1: M). Daviesin ym. (2006) tutkimus käsitteli analyysivaiheen mallintamisen nykytilaa australialaisten ammattilaisten keskuudessa. Fettken (2009) mukaan Daviesin ym. (2009) tutkimuksen tuloksia ei voinut siirtää koskemaan saksalaisia ammattilaisia, jotka hänen mukaansa olivat verrattain perehtyneitä mallinnustekniikoiden käyttäjiä. Fettken (2009) tutkimuksen tulokset osoittivat, että saksalaiset ammattilaiset arvostivat ja hyödynsivät mallintamista australialaisia enemmän, mutta käyttökohteet olivat samanlaisia.

Forward ym. (2010) halusivat esiintuoda syitä sille, miksi mallintaminen ei ollut kauttaaltaan käytössä, vaikka sen hyödyt ovat yleisesti tiedossa. He vertailivat koodikeskeisiä ja mallikeskeisiä toimintatapoja ja pyrkivät selvittämään, kuinka mallinnustyökaluja voisi kehittää. Lähtökohtana tutkimukselle he esittivät aiempien tutkimusten perusteella työkalujen, kielten ja prosessien heikkoudet, koulutuksen puutteen ja mallintamisen mieltämisen kalliiksi hyötyihin nähden. Tulosten perusteella vapaamuotoinen mallintaminen oli yleistä, kun taas formaali mallintaminen ei ollut laajalle levinnyttä. Suurin osa ammattilaisista kuitenkin ymmärsi mallintamisen arvon etenkin viestinnässä, vaikka toteuttivatkin sitä rajoitetusti. (Forward ym., 2010.)

Malavolta ym. (2012) selvittivät arkkitehtien tarpeita arkkitehtuurin mallintamisessa käytettäviin kieliin liittyen. Tulosten perusteella kielten tulisi tukea kommunikointia erilaisten sidosryhmien keskuudessa, mutta mahdollistaa myös erilaisten automatisoitujen aktiviteettien, kuten testauksen ja mallien tarkastusten suorittamisen. Malavolta ym. (2012) toteavat, että arkkitehdit valitsevat mieluummin yksinkertaisen ja intuitiivisen kielen formaalin sijaan. Tutkimuksen mukaan arkkitehdit hyödynsivätkin eniten UML:ää ja vapaamuotoisia notaatioita. Myös formaalin AADL:n (ks. Feiler, Lewis & Vestal, 2003) ja puoliformaalin ArchiMaten (ks. Lankhorst, Proper & Jonkers, 2010) käyttöä raportoitiin enemmän kuin muita kieliä, mutta ero UML:ään oli kuitenkin suuri. (Malavolta ym., 2012)

Gorschek, Tempero ja Angelis (2014) esittävät, että tutkimuksissa on keskitytty lähinnä siihen, miten ammattilaiset käyttävät malleja, eikä riittävästi käytön yleisyyteen. Muista empiirisistä tutkimuksista poiketen he keskittyivät mallien hyödyntämiseen kehitystyön aikana, eivätkä varsinaiseen mallintamistyöhön ja sen yleisyyteen. Tutkimuksen mukaan mallien tai varsinkaan UML:n käyttö ei ollut yleistä eivätkä tulokset siten vastanneet olettamuksia, joihin suuri osa alan tutkimusta perustuu. Gorschek ym. (2014) pitävät epätodennäköisenä, että mallipohjainen kehittäminen voisi yleistyä alalla. He selittävät yleistymisen edellyttävän, että suurin osa ammattilaisista hyödyntäisi edes perinteistä mallintamista, mikä heidän tutkimuksensa perusteella ei ollut vallitseva tila alalla. (Gorschek ym., 2014.)

Störrle (2017) esittää, että mallintamisen rooli on alalla kiistanalainen. Tutkimuksessa havaittiin, että mallintaminen oli alalla yleistä, mutta sen formaalius ja käyttötarkoitukset vaihtelivat suuresti projektista ja tekijästä riippuen. Vapaamuotoinen mallintaminen oli tutkimuksen mukaan puoliformaalia tai formaalia yleisempää. (Störrle, 2017). Myös Ozkaya (2018) tutki eri formaalisuusasteiden mallintamista, kontekstinaan ohjelmistoarkkitehtuurit. Tulokset osoittivat, että UML:n lisäksi ammattilaiset käyttivät täysin vapaamuotoisia ad hoc -mallintamistapoja enemmän kuin formaaleja ja arkkitehtuurin kuvauskieliä (engl. architecture description language, ADL) (Ozkaya, 2018).

Badreddin, Khandoker, Forward, Masmali ja Lethbridge (2018) kartoittivat mallintamiseen liittyviä trendejä ja suorittivat tutkimuksensa kahdessa osassa kymmenen vuoden välein. Ensimmäinen osa on raportoitu omana tutkimuksenaan (ks. Forward ym. 2010). Badreddinin ym. (2018) mukaan vapaamuotoisen mallintamisen lisäksi sovellusaluekeskeinen ja formaali mallintaminen ovat lisääntyneet alalla, kuten myös mallintaminen ainoastaan erikseen pyydettäessä.

Tutkimukset mallipohjaisesta tietojärjestelmien kehittämisestä (taulukossa 1: MD). Hutchinson, Whittle, Rouncefield ja Kristoffersen (2011b) yhdistivät määrällisen kyselyn, laadullisen haastattelun ja tapaustutkimuksen menetelmiä tutkiessaan mallipohjaisen tietojärjestelmäkehityksen nykytilaa. Aineistosta on tehty useita julkaisuja (ks. Hutchinson ym., 2011a**;** Hutchinson ym., 2011b; Hutchinson, Whittle &, Rouncefield, 2014**;** Whittle, Hutchinson & Rouncefield, 2013). Tulosten perusteella mallipohjaisuus oli oletettua yleisempää (Whittle ym, 2013) ja siinä hyödynnettiin odotettua enemmän sovellusaluekohtaisia kieliä (Hutchinson ym., 2014).

Italialaisessa tutkimushankkeessa (ks. Tomassetti, Torchiano, Tiso, Ricca & Reggio, 2012; Torchiano, Tomassetti, Ricca, Tiso & Reggio, 2011; Torchiano, Tomassetti, Ricca, Tiso & Reggio, 2013) kartoitettiin ammattilaisten näkemyksiä ja kokemuksia mallintamisesta perinteisessä ja mallipohjaisessa kehitystyössä. Torchiano ym. (2013) toteavat, että mallintaminen ja mallipohjaiset tekniikat ovat olennainen osa italialaista ohjelmistokehitystä. Mallipohjainen kehittäminen oli kuitenkin selkeästi harvinaisempaa, mikä voi johtua siitä, että ammattilaiset eivät olleet saaneet siihen liittyen riittävästi koulutusta (Tomassetti ym., 2012).

Mohagheghi, Gilani, Stefanescu ja Fernandez (2013) suorittivat neljä isoa organisaatiota kattavan tapaustutkimuksen mallipohjaisen kehittämisen nykytilasta. Tutkimuksen perusteella mallipohjaisuutta pidettiin hyödyllisenä tietyissä tilanteissa monimutkaisten järjestelmien kehittämisessä, mutta menetelmä ja työkalut eivät olleet helppokäyttöisiä. Haasteet kuitenkin lieventyivät käytön jatkuessa. (Mohagheghi ym., 2013.)

TAULUKKO 1 Yhteenveto käytännön kokemuksista

Tutkimus	**Tiedonkeruumenetelmä**	**Käytetyimmät mallinnuskielet/ kaaviot**	**Syitä käytölle / hyödyt**	**Syitä olla käyttämättä / haasteet**
Grossman ym. (2005) / UML	Kysely, 131 ammattilaista, etenkin Yhdysvalloista	Käyttötapaus-, luokka- ja sekvenssikaavio	Vaatimusten määrittely ja kommunikointi, ohjaa koodaustyötä, takaisinmallinnus, riittävä ymmärrettävyys, tarkkuus, johdonmukaisuus ja joustavuus	Monitulkintaisuus, ei varmuutta hyödyistä
Davies ym. (2006) / M	Kysely, 312 australialaista analyytikkoa	ER, erilaiset virta/vuokaaviot, UML, rakenteiset kaaviot	Koettu hyödyllisyys, kommunikaatioväline, helppokäyttöisyys, yhteensopivuus	Koettu hyödyttömyys, monimutkaisuus, yhteensopimattomuus
Dobing & Parsons (2006; 2008) / UML	Kysely, 182 analyytikkoa (käyttää UML:ää) + 102 analyytikkoa (ei käytä UML:ää)	Luokka-, käyttötapaus- ja sekvenssikaavio	Koettu hyödyllisyys, yhteisymmärryksen saavuttaminen, ohjaa koodaustyötä, dokumentointi, vaatimusmäärittely	Kaavioita ei ymmärretä, koettu hyödyttömyys, riittämätön laatu kustannuksiin nähden
Lange ym. (2006) / UML	Kysely + mallien analysointi 14 tapaustutkimuksen pohjalta, 80 arkkitehtiä			Mallien keskeneräisyys, toimintatapojen puute, työkalut eivät tue
Nugroho & Chaudron (2008) / UML	Kysely, 80 ammattilaista, valtaosa Alankomaista		Vaikuttaa positiivisesti laatuun ja tuottavuuteen	Mallien keskeneräisyys, epätarkkuus ja epäjohdonmukaisuus, työkalut ja toimintatavat eivät tue
Fettke (2009) / M	Kysely, 304 saksalaista ammattilaista	ER, UML, työvuokaavio, EPC, tilakaavio	Kielen ilmaisuvoima, usean näkymän kuvaaminen, mallin soveltuvuus	Työkalut eivät tue, menetelmien runsas määrä
Forward ym. (2010) / M	Kysely, 113 ammattilaista, valtaosa Kanadasta ja Yhdysvalloista	UML, SQL, rakenteiset kaaviot, ER	Kommunikaatioväline, alustava suunnittelu, dokumentointi	Mallien vanhentuminen, mallien yhteensopimattomuus eri työkalujen kesken, työkalut eivät tue
Scanniello ym. (2010) / UML	Kysely, 22 vastaajaa 22 italialaisesta IT-organisaatiosta	Käyttötapaus-, luokka- ja tilakaavio	Analyysi ja suunnittelu, ylläpito	
Hutchinson ym. (2011a); Hutchinson ym. (2011b); Hutchinson ym. (2014); Whittle ym. (2013) / MD	Kysely, haastattelut, neljän organisaation tapaustutkimus, 250 vastaajaa, 22 haastateltavaa 17 yrityksestä, (valtaosa Euroopasta)	UML, DSL, BPMN, SysML, MATLAB/Simulink	Tuottavuuden kasvu, ylläpidon helpottuminen, kokonaisvaltaiset hyödyt	Työkalut eivät tue, lisäkoulutuksen tarve, edellyttää investointeja ja organisaation sitoutumista
Malavolta ym. (2012) / M	Kysely (23/48) + haastattelut (25/48), 48 ammattilaista 40 organisaatiosta, 15 maasta	UML, vapaat notaatiot, AADL, ArchiMate	Kommunikaatioväline, tukee vaatimusmäärittelyä, suunnittelua ja yhteistyötä	Työkalut eivät tue

(jatkuu)

Taulukko 1 (jatkuu)

Tomassetti ym. (2012); Torchiano ym. (2011); Torchiano ym. (2013) / M+MD	Kysely, 155 italialaista ammattilaista + 3 tarkentavaa haastattelua	UML, DSL	Korkean tason suunnittelu, dokumentointi, ylläpidon helpottuminen, korkeampi laatu	Liian vaivalloista, ei tarpeeksi hyödyllistä, työkalut eivät tue, kompetenssin puute
Fitsilis ym. (2013) / UML	Kysely, 91 kreikkalaista ammattilaista	Luokka-, käyttötapaus- ja aktiviteettikaavio	Nopeuttaa työskentelyä, johtaa parempaan laatuun, vähentää kustannuksia	
Mohagheghi ym. (2013) / MD	Neljän organisaation tapaustutkimus		Viestinnän helpottuminen sidosryhmien kesken, johdonmukaisuus ja yhteneväisyys tuotosten välillä, mahdollinen tuottavuuden kasvu	Työkalut eivät tue, menetelmän ja työkalujen käytön vaikeus
Petre (2013; 2014) / UML	Haastattelut, 50 ammattilaista, etenkin Yhdysvalloista ja Yhdistyneestä kuningaskunnasta + 10 lisähaastattelua	Luokka-, sekvenssi- ja aktiviteettikaavio	Tukee ajattelutyötä, kommunikaatioväline, vaatimusmäärittely	Monimutkaisuus, semantiikan monitulkintaisuus, edellyttää organisatorisia muutoksia
Gorschek ym. (2014) / M	Kysely, 3785 ammattilaista, etenkin Pohjois-Amerikasta ja Euroopasta	Vapaat notaatiot, UML/UML:n tapainen (vain pieni osa vastaajista)	Viestintä ja yhteistyö, alkuvaiheen luonnostelu, korkean tason / ison kuvan visualisointi	
Reggio ym. (2014) / UML	Kysely, 275 ammattilaista/akateemikkoa, etenkin Euroopasta ja Italiasta	Sekvenssi-, luokka- ja käyttötapauskaavio		
Fernández-Sáez ym. (2015) / UML	Kysely, 268 ammattilaista 12 maasta, valtaosa Italiasta	Luokka-, käyttötapaus- ja sekvenssikaavio	Ylläpidossa edesauttaa ymmärrystä ja vikojen löytymistä, vähentää kustannuksia	Perehdytään mieluummin koodista katsomalla
Störrle (2017) / M	Kysely, 96 ammattilaista, etenkin Saksasta, Skandinaviasta ja Intiasta	UML, DSL, ECORE, ER, BPMN	Keskustelu kollegojen kanssa, idean tai käsitteen visualisointi, ajattelun tuki, johtaa parempaan laatuun	Työkalut eivät tue
Badreddin ym. (2018) / M	Kysely, 228 ammattilaista, valtaosa Kanadasta ja Yhdysvalloista	ER, UML, rakenteiset kaaviot, SQL, DSL	Kommunikaatioväline, käyttäytymisen ymmärtäminen, vaatimustenmukaisen järjestelmän luominen	Työkalut eivät tue, mallintamisen kapea-alainen hyödyntäminen
Farias ym. (2018) / UML	Kysely, 222 brasilialaista ammattilaista			Työkalut eivät tue, mallien päivittäminen
Ozkaya (2018) / M	Kysely, 115 ammattilaista	UML, DSL, vapaat notaatiot, formaalit kielet, ADL	Helppokäyttöisyys, yleiskäyttöisyys, visuaalisuus (UML, DSL/vapaat)	Monitulkintaisuus (UML/DSL/vapaat), jyrkkä oppimiskäyrä (formaalit), työkalut eivät tue (ADL)

3.2.2 Mallinnuskielet, -tekniikat ja -työkalut käytännössä

Kaikissa mallinnuskielten ja -tekniikoiden käyttöä erittelevissä tutkimuksissa UML on kuulunut käytetyimpien joukkoon (ks. Badreddin ym., 2018; Davies ym., 2006; Fettke, 2009; Fernández-Sáez ym., 2015; Forward ym., 2010; Gorschek ym., 2014; Hutchinson ym., 2011b; Ozkaya, 2018; Störrle, 2017; Tomassetti ym., 2012; Torchiano ym., 2011; Torchiano ym., 2013; Whittle ym., 2013) ja suuressa osassa myös ER-kaavio (Badreddin ym., 2018; Davies ym., 2006; Fettke, 2009; Forward ym., 2010; Störrle, 2017). Näiden lisäksi on raportoitu etenkin erilaisten rakenteisten kaavioiden (ks. Badreddin ym., 2018; Davies ym., 2006; Fettke, 2009; Forward ym., 2010) ja sovellusaluekohtaisten kielten käyttöä (ks. Badreddin ym., 2018; Hutchinson ym., 2011b; Ozkaya, 2018; Störrle, 2017; Tomassetti ym., 2012; Torchiano ym., 2011; Torchiano ym., 2013; Whittle ym., 2013).

Rakenteiset tekniikat olivat suosittuja etenkin Daviesin ym. (2006) tutkimuksen ajankohtana, kun taas myöhemmin suoritetut Fitsilisin ym. (2013), Forwardin ym. (2010), Hutchinsonin ym. (2011b), Ozkayan (2018) ja Störrlen (2017) tutkimukset osoittivat UML:n olevan selkeästi käytetyin mallinnuskieli. Toisaalta Farias ym. (2018), Gorschek ym. (2014) ja Petre (2013) raportoivat UML:n käytön olevan vähäistä. Grossmanin ym. (2005), Daviesin ym. (2006), Fettken (2009) ja Scanniellon ym. (2010) tutkimuksissa raportoitiin myös prosessiohjeita sisältävien mallintamismenetelmien (RAD, Unified Process) käyttöä.

Daviesin ym. (2006) tutkimuksessa yli puolet vastaajista ei käyttänyt tai tuntenut UML:ää. Fettken (2009) tutkimuksessa UML oli ER-kaavion ohella käytetyin mallinnustekniikka, mikä vahvisti tutkijan näkemystä alueellisista ja kulttuurisista eroista. Samaan tapaan Störrle (2017) nosti esille tulostensa erot verrattuna alalla tunnettuun Petren (2013) tutkimukseen. Petren (2013) tutkimukseen osallistujista suurin osa oli Yhdysvalloista ja Yhdistyneestä kuningaskunnasta, kun taas Störrlen (2017) kyselyyn vastaajat olivat suurimmaksi osaksi Saksasta, Skandinaviasta ja Intiasta. Kolmen viimeksi mainitun lisäksi UML:n käyttö oli yleistä Italiassa (Scanniello ym., 2010; Torchiano ym., 2011) ja Kreikassa (Fitsilis ym., 2013), vähäistä Brasiliassa (Farias ym., 2018). Forwardin ym. (2010) mukaan pohjoisamerikkalaiset käyttivät vähemmän mallinnustyökaluja kuin muualla toimivat ammattilaiset.

Jokaisessa UML:n eri kaaviotyyppien käyttöä vertailevassa tutkimuksessa luokkakaavio oli kolmen käytetyimmän kaaviotyypin joukossa, ja loput kolmen kärjessä vaihtelivat aktiviteetti-, käyttötapaus- ja sekvenssikaavioiden välillä (ks. Dobing & Parsons, 2006; Fernández-Sáez ym., 2015; Fitsilis ym., 2013; Grossman ym., 2005; Hutchinson ym., 2011b; Ozkaya, 2018; Petre, 2013; Reggio ym., 2014). Poikkeuksena Scanniellon ym. (2010) kartoittava kysely, jossa tilakaavio oli käytetyimpien joukossa. Reggio ym. (2014) lisäävät, että käytetyimmät ja tunnetuimmat UML:n kaaviotyypit ovat lukeutuneet UML:ään jo sen ensimmäisestä versiosta lähtien ja ne myös vastaavat esiintyvyyttä oppimateriaaleissa ja työkaluissa.

Vähiten käytetyimpiä kaaviotyyppejä olivat Grossmanin ym. (2005) mukaan sijoittelukaavio, Dobingin ja Parsonin (2006), Fernández-Sáezin ym. (2015)

ja Hutchinsonin ym. (2014) mukaan yhteistoimintakaavio sekä Ozkayan (2018) mukaan oliokaavio. Fernández-Sáez ym. (2015) raportoivat myös kooste-, kokoava vuorovaikutus- ja ajoituskaavioiden käyttöä vähäiseksi. Petren (2013) haastateltavat eivät pitäneet olio- ja käyttötapauskaavioita hyödyllisinä, mikä tarkoittanee myös niiden vähäistä käyttöä. Fitsilis ym. (2013) perustelevat pakkaus-, ajoitus- ja tilakaavioiden vähäistä käyttöä sillä, että heidän tutkimuksessaan projektit olivat valtaosin pieniä ja mainitut kaaviotyypit olivat yleisimmin käytössä suurissa ja monimutkaisissa kehitysprojekteissa. Reggion ym. (2014) mukaan profiili- ja ajoituskaaviot tunnettiin huonosti ja niitä myös käytettiin harvoin. Nämä kaaviotyypit ovat selkeästi spesifimpään käyttöön tarkoitettuja ja liitetty UML:ään vasta myöhemmissä versioissa.

Reggion ym. (2014) mukaan UML oli akateemikkojen keskuudessa kokonaisuudessaan hieman paremmin tunnettu ja enemmän käytetty kuin ammattilaisten keskuudessa. Ammattilaisten UML:n käytön ja tuntemisen välillä oli selkeä yhteys: mitä kauemmin UML oli ollut käytössä, sitä paremmin se tunnettiin harvinaisempia kaaviotyyppejä myöten (Reggio ym., 2014). Samaan tapaan Dobingin ja Parsonsin (2006) tulokset viittasivat siihen, että eri kaaviotyyppien käyttö lisääntyi kokemuksen karttuessa ja Grossmanin ym. (2005) siihen, että tyytyväisyys UML:ään lisääntyi jatkuvassa käytössä.

Vain muutamassa tutkimuksessa eriteltiin ammattilaisten käytössä olevia mallinnustyökaluja. Daviesin ym. (2006) ja Fettken (2009) mukaan työkalumarkkinoiden heterogeenisyydestä huolimatta niitä hallitsi Microsoft Visio. Myös Rational Rosen (huom. nykyään saatavilla IBM Rational Software Architect[12]) käyttöä raportoitiin (Davies ym, 2006; Fettke, 2009). Grossman ym. (2005) raportoivat UML:ää tukevista työkaluista selvästi käytetyimpien olevan Rational Rose ja Sparx Systemsin Enterprise Architect. Fernández-Sáezin ym. (2015) tutkimuksessa mainittiin aiempien lisäksi myös avoimen lähdekoodin työkaluja. Badreddin ym. (2018) raportoivat vapaamuotoisten tapojen, kuten valkotaulun käytön, olevan yleisempää kuin mallintaminen piirtotyökaluilla tai varsinaisilla mallinnustyökaluilla. Whittle ym. (2013) mainitsevat, että heidän tutkimuksessaan listattiin yli 100 jatkuvasti käytössä olevaa työkalua.

Badreddinin ym. (2018) mukaan tiedon mallintaminen ER-kaaviota hyödyntämällä oli lisääntynyt. Tämän lisäksi formaali mallintaminen ja siten myös tarkkaan määriteltyjen mallinnuskielten, kuten sovellusaluekohtaisten kielten, käyttö oli lisääntynyt (Badreddin ym., 2018). Whittle ym. (2013) lisäävät, että mallipohjaisessa kehittämisessä sovellusaluekohtaisia kieliä hyödynnettiin tiettyihin järjestelmän osiin, ei järjestelmän kehittämiseen kokonaisuudessaan. Menestyneissä käytännöissä mallipohjaisuutta hyödynnettiin tarpeen mukaan ja yhdistettiin joustavasti muihin kehitysmenetelmiin (Whittle ym., 2013). Badreddin ym. (2018) raportoivat myös vapaamuotoisen mallintamisen lisääntyneen ja se olikin Ozkayan (2018) ja Störrlen (2017) tutkimuksissa selkeästi yleisin mallintamisen muoto. Läpi kaikkien tutkimusten UML:ää raportoitiin käytettävän lähinnä vapaamuotoisella tavalla (ks. Fitsilis ym., 2013; Forward

[12] https://www.ibm.com/fi-en/marketplace/rational-software-architect-designer

ym., 2010; Grossman ym., 2005; Lange ym., 2006; Nugroho & Chaudron, 2008; Petre, 2013).

3.2.3 Mallien käyttökohteet ja mallintamisen hyödyt käytännössä

Daviesin ym. (2006) ja Fettken (2009) tutkimusten mukaan mallintamista käytettiin etenkin tietokantojen suunnittelussa ja hallinnassa, liiketoimintaprosessien kehittämisessä ja dokumentoinnissa sekä ohjelmistokehityksessä. Davies ym. (2006) lisäävät, että pienet yritykset ja kokemattomammat mallintajat hyödynsivät mallintamista teknisissä suunnittelun ongelmissa, kun taas suuret yritykset ja kokeneemmat mallintajat keskittyivät dokumentointiin ja liiketoimintaprosessien kehittämiseen. Störrlen (2017) tutkimuksessa ohjelmistokehitys oli selkeästi yleisin mallintamisen käyttökohde, kun tietokannat saivat vain vähän kannatusta.

Kun kehitystyöhön liittyviä aktiviteetteja ja mallintamisen hyötyjä tarkastellaan tarkemmin, malleja käytettiin eniten viestintään kollegoiden kesken (Dobing & Parsons, 2006; Gorschek ym., 2014; Störrle, 2017) ja eri sidosryhmien kanssa (Davies ym., 2006; Forward ym., 2010; Grossman ym., 2005; Mohagheghi ym., 2013; Petre, 2013). Muita yleisiä käyttökohteita olivat idean tai käsitteen visualisointi (Petre, 2013; Störrle, 2017), ajattelun tuki (Petre, 2013; Störrle, 2017), yhteisymmärryksen saavuttaminen (Gorschek ym., 2014; Hutchinson ym., 2014), ongelmanratkaisu (Gorschek ym., 2014; Petre, 2013), koodaustyön ohjaaminen (Dobing & Parsons, 2006; Grossman ym., 2005) ja arkkitehtuurin määrittäminen (Ozkaya, 2018). Dobing ja Parsons (2006; 2008) ja Petre (2013) erittelivät myös UML:n eri kaaviotyyppien käyttökohteita. Luokkakaaviota käytettiin rakenteen, käsitteiden, arkkitehtuurin sekä rajapintojen kuvaamiseen ja dokumentointiin, ja sekvenssi- ja aktiviteettikaavioita vaatimusten ja käyttäytymisen selvittämiseen sidosryhmien kanssa (Dobing & Parsons, 2006; 2008; Petre, 2013).

Badreddin ym. (2018) kertovat, että mallintamista käytettiin lähinnä vaatimusmäärittelyssä ja muutoin sen hyödyntäminen oli vähäistä. Tämä vastasi myös muiden tutkimusten havaintoja, lähes kaikissa ammattilaisten kerrottiin hyödyntävän UML:ää ja muita tekniikoita sekä työkaluja alkuvaiheessa kehitystyötä (ks. Fitsilis ym., 2013; Forward ym., 2010; Gorschek ym., 2014; Grossman ym., 2005; Mohagheghi ym., 2013; Nugroho & Chaudron, 2008; Petre, 2013). Störrle (2017) pitää yllättävänä, että ammattilaiset eivät juurikaan hyödyntäneet malleja järjestelmään perehtyessään ja epäilee syyksi myös Langen ym. (2006) esittämiä syitä: malleista voi olla vaikea löytää oleellista tietoa niiden epätarkkuuden ja paikkansapitämättömyyden vuoksi. Poikkeuksena voidaan pitää italialaisia tutkimuksia (Scanniello ym., 2010; Torchiano ym., 2013), joiden mukaan mallintamista hyödynnettiin suunnittelun ja dokumentaation ohella myös ylläpidossa.

UML:n käyttö myötäilee ymmärrettävästi kaikkea mallintamista koskevaa kaavaa: se oli yleisintä alkuvaiheessa kehitystyötä (Fitsilis ym., 2013; Grossman ym., 2005; Petre, 2013) ja vähäistä testaus- ja ylläpitovaiheissa (Nugroho & Chaudron, 2008). Nugroho ja Chaudron (2008) epäilevät, että testauksessa hyö-

dynnettiin lähinnä tekstimuotoisia vaatimuksia ja ylläpidossa UML-mallit sivuutettiin, koska ne eivät olleet ajantasaisia eivätkä siten vastanneet koodia. Ylläpidon osalta tämä vastaa Langen ym. (2006) ja Störrlen (2017) näkemyksiä. Poikkeuksena Scanniellon ym. (2010) tutkimus, jonka mukaan UML-mallien käyttö oli yleistä ylläpidossa ja osin myös Fernández-Sáezin ym. (2015) tutkimus, jossa lähes puolet vastaajista hyödynsi UML:ää. On huomioitavaa, että molemmat tutkimukset selvittivät UML:n roolia nimenomaan ylläpidossa.

UML:ää käytettiin etenkin viestintää edesauttamaan (Dobing & Parsons, 2006; Grossman ym., 2005; Petre, 2013). Nugroho & Chaudron (2008) korostivat käytön kohdistuvan monimutkaisiin ja kriittisiin järjestelmäosiin. Ammattilaisten mielipiteet UML:stä vaihtelivat arvostuksesta (ks. Fitsilis ym., 2013) kritiikkiin (ks. Petre, 2013) ja myös saman tutkimuksen puitteissa olivat hyvin kaksijakoisia (ks. Grossman ym., 2005; Hutchinson ym., 2011b). UML:n käytön hyötyinä esitettiin mm. ymmärrettävyys (Grossman ym., 2005; Nugroho & Chaudron, 2008), laadun ja tuottavuuden kasvu (Fitsilis ym., 2013; Nugroho & Chaudron, 2008) ja työskentelyn nopeutuminen (Fitsilis ym., 2013).

Mallipohjaisessa kehittämisessä raportoitiin ylläpidettävyyden (Hutchinson ym., 2011b; Torchiano ym., 2013) lisäksi työn tuottavuuden kasvua (Hutchinson ym., 2011b; Mohagheghi ym., 2013; Torchiano ym., 2013), jonka voidaan ajatella johtuvan etenkin koodigeneroinnista. Whittle ym. (2013) kuitenkin esittävät, että koodigenerointi ei ollut avainasemassa mallipohjaisen kehittämisen käyttöönotossa. Hyödyt olivat kokonaisvaltaisempia, koska mallipohjainen kehitystyö ohjasi tekijöitä paremman ohjelmistoarkkitehtuurin ja siten myös onnistuneemman ratkaisun rakentamiseen (Whittle ym., 2013). Myös perinteisessä tietojärjestelmäkehityksessä mallintamisen hyötyinä raportoitiin laadukkaan ohjelmiston tuottamista (ks. Fitsilis ym., 2013; Nugroho & Chaudron, 2008; Störrle, 2017; Torchiano ym., 2013).

3.2.4 Mallintamiseen liittyvät haasteet ja kehitysehdotukset käytännössä

Merkittävin haaste Forwardin ym. (2010) mukaan oli mallien vanhentuminen kehityksen kuluessa niin, että ne eivät enää vastanneet koodia. Myös Farias ym. (2018) esittävät mallien päivittämisen ja koodin kanssa yhtenäisenä pitämisen haasteena. Torchianon ym. (2011) mukaan yleisimpinä syinä olla mallintamatta mainittiin sen vaivalloisuus ja hyödyttömyys. Vain pieni osa ammattilaisista mainitsi ajanpuutteen syyksi olla mallintamatta. Vaadittavan osaamisen puute oli merkittävä syy mallintamisen sivuuttamiselle etenkin niissä yrityksissä, joissa mallintamista ei suoritettu koskaan. (Torchiano ym., 2011.) Badreddin ym. (2018) esittävät, että mallintamisen kapea-alainen hyödyntäminen ainoastaan vaatimusmäärittelyssä oli pysyvä haaste alalla. Gorschekin ym. (2014) mukaan malleja hyödynsivät enemmän akateemisen koulutuksen käyneet, mutta hekään eivät erityisen paljoa ja mallien käyttö väheni ammattilaisen kokemuksen lisääntyessä taustasta riippumatta.

Työkaluihin liittyvät puutteet ja ongelmat ovat olleet merkittävä haaste mallintamiseen liittyen läpi tutkimusten. Davies ym. (2006) kertovat, että työka-

lujen koettu hyödyttömyys oli yksi syy olla mallintamatta ja että ammattilaiset raportoivat työkaluihin liittyviä ongelmia. Forward ym. (2010) avaavat näitä ongelmia tarkemmin. Ammattilaisten mielestä työkalut olivat mm. liian raskaita, monimutkaisia tai kalliita. Mallien luominen ja editointi oli hidasta, eikä koodigenerointi ollut halutulla tasolla tai kaikkia toivottuja toimintoja saatavilla. (Forward ym., 2010.) Badreddin ym. (2018) lisäävät, että tyytymättömyys työkaluihin oli lisääntynyt merkittävästi vuosien varrella ja yhä useampi ammattilainen oli sitä mieltä, että työkalut eivät tukeneet kommunikointia kehittäjien välillä. Tyytymättömyys työkaluihin oli nähtävissä mallien vähäisessä käytössä ja uudelleenkäytössä sekä oletettavasti myös vapaamuotoisessa mallintamisessa (Badreddin ym., 2018).

Forward ym. (2010) raportoivat työkaluilla tehtävän koodigeneroinnin mallien pohjalta olevan vähäistä, mikä heidän mielestään osoitti, että generoidun koodin laatu ei ollut halutulla tasolla. Tutkijat esittävät, että mallintaminen voisi olla yleisempää, jos malleja voisi tuottaa tekstimuotoisina, samaan tapaan kuin koodikeskeiset ohjelmistokehittäjät työskentelevät. Mallien luomisen tulisi olla helppoa ja nopeaa, mutta graafiset työkalut eivät todennäköisesti tarjonneet intuitiivisia ja tehokkaita käyttöliittymiä. (Forward ym., 2010.) Mohagheghi ym. (2013) kertovat, että mallipohjaista kehitystyötä toivotulla tavalla tukevia työkaluja ei ollut vielä saatavilla, joten organisaatiot kehittävät niitä itse omaan käyttöönsä. Tomassettin ym. (2012) mukaan osa ammattilaisista koki työkalujen puutteet esteeksi mallipohjaisuuden käyttöönotolle.

Kuten kaikkeen mallintamiseen, luonnollisesti myös UML:n käytön haasteisiin yhdistettiin työkaluihin liittyvät ongelmat (Nugroho & Chaudron, 2008; Petre, 2013). Muina merkittävinä haasteina esitettiin kielen semantiikan väljyys (Grossman ym., 2005; Ozkaya, 2018; Petre, 2013) ja monimutkaisuus (Dobing & Parsons, 2006; Hutchinson ym., 2011b; Petre, 2013) sekä riittämätön laatu kustannuksiin nähden (Dobing & Parsons, 2006). Myös mallien keskeneräisyyttä, tiedon hajautumista eri näkymien kesken, mallien osien epäsuhtaista tarkkuutta ja kehitystyön epäjohdonmukaisuutta raportoitiin (Lange ym., 2006; Nugroho & Chaudron, 2008). UML:n käyttöön yhdistettiin filosofisia ja ideologisia piirteitä, jotka vaatisivat muutosten tekemistä yrityskulttuuriin, tuomatta kuitenkaan mitään etuja. Kritiikki kohdistui myös kontekstin puutteeseen. Osa ammattilaisista oli sitä mieltä, että UML käsittelee lähinnä ohjelmistoarkkitehtuuria, ei koko järjestelmää. (Petre, 2013.)

UML:n käyttöä voitaisiin tutkijoiden mukaan edesauttaa kattavammalla koulutuksella (ks. Dobing ja Parsons, 2006), toimintaohjeiden standardoinnilla (ks. Dobing ja Parsons, 2006; Lange ym., 2006) ja työkaluja kehittämällä (ks. Lange ym., 2006; Farias ym., 2018). Lange ym. (2006) keskittyivät tutkimuksessaan mallien laatuun, joten heidän kehitysideansa koskivat suunnitteluvirheitä ja epäjohdonmukaisuuksia havaitsevia ominaisuuksia, kun taas Farias ym. (2018) ehdottivat myös kehitystiimin reaaliaikaista yhteistyötä tukevia ominaisuuksia, kuten järjestelmän kokonaiskuvan, ns. ”ison kuvan” generointia ja kollaboratiivista mallintamista.

Merkittävinä haasteina mallipohjaiselle kehittämiselle Hutchinson ym. (2011b) raportoivat työkalujen lisäksi sovellusaluekohtaisiin kieliin liittyvät ongelmat sekä organisatoriset ja koulutukseen liittyvät tekijät. Mallintaminen ja abstrakti ajattelu eivät välttämättä onnistu kaikilta, mutta siitä, onko tämä enemmän kiinni sisäänrakennetuista kyvyistä, kokemuksesta vai mieltymyksistä, oltiin montaa mieltä (Hutchinson ym., 2011b). Suurin osa ammattilaisista oli kuitenkin sitä mieltä, että mallipohjainen kehittäminen edellyttää ylimääräistä koulutusta ja työkalutukea (Hutchinson ym., 2011b) ja samaan lopputulokseen päätyivät tutkimuksessaan myös Tomassetti ym. (2012).

Hutchinson ym. (2011a) raportoivat mallipohjaisen tietojärjestelmäkehityksen menestystekijöiksi progressiivisen ja iteratiivisen lähestymistavan, organisaation sitoutumisen ja keskittymisen liiketoimintaan, jossa mallipohjaisuus käsitetään ratkaisuna kaupallisiin ja organisatorisiin haasteisiin. Hutchinson ym. (2014) toteavatkin, että harkittaessa mallipohjaisen kehittämisen käyttöönottoa tulisi keskittyä muutosjohtamiseen pelkkien teknisten seikkojen sijaan.

4 TUTKIMUSMENETELMÄT

Tutkielman yhteydessä toteutetulla empiirisellä tutkimuksella selvitettiin Suomessa toimivien ammattilaisten näkemyksiä ja kokemuksia tietojärjestelmien mallintamisesta. Tässä pääluvussa kerrotaan käytetyt tutkimusmenetelmät ja perustelut niiden valinnoille. Ensimmäisessä alaluvussa motivoidaan empiirinen tutkimus, toisessa esitellään tutkimusote ja valittu tiedonkeruumenetelmä. Kolmannessa alaluvussa käydään läpi haastateltavien hankinta ja neljännessä haastattelujen toteutus. Viidennessä alaluvussa käsitellään aineiston analyysiä ja viimeisessä pohditaan tutkimuksen reliaabeliutta ja validiutta.

4.1 Empiirisen tutkimuksen motivointi

Mallintamisella on pitkät perinteet tietojärjestelmäkehityksessä ja aihetta on tutkittu jo vuosikymmenien ajan. Tutkimus on kuitenkin suurelta osin keskittynyt uusien mallintamismenetelmien esittelyyn (Siau & Rossi, 2011) ja olemassa olevien menetelmien ja työkalujen arviointiin (Davies ym., 2006). Tämän lisäksi nämä tutkimukset ovat perustuneet lähinnä oletuksiin empiirisen todistusaineiston sijaan (Siau, 2004; Siau & Rossi, 2011). Budgen ym. (2011) huomauttavat, että kun empiirisiä tutkimuksia on toteutettu, niihin on osallistettu paljolti opiskelijoita ammattilaisten sijaan. Viime vuosina mallintamistutkimukseen liittyvä epäsuhta on noteerattu entistä laajemmin ja empiirisiä tutkimuksia ammattilaisten keskuudessa on alettu toteuttamaan. Nämä tutkimukset ovat kuitenkin raportoineet osin ristiriitaisia tuloksia ja alueellisia sekä kulttuurisia eroavaisuuksia (ks. esim. Petre, 2013; Störrle, 2017). Farias ym. (2018) ja Störrle (2017) painottavat, että mahdolliset alueelliset erot tulee ottaa paremmin huomioon alan tutkimuksissa.

Kirjallisuudessa näkemykset tietojärjestelmien mallintamisesta vaihtelevat sen keskeisestä merkityksestä (ks. Clarke ym., 2016) väitteisiin sen aliarvostetusta asemasta alalla (ks. Lukyanenko, 2018). Perinteisesti mallintamisen on katsottu olevan erityisen tärkeä osa tietojärjestelmän analyysin vaatimusmäärit-

telyä (Gemino & Wand, 2004), mutta vaatimusten kirjaamiseen raportoidaan käytettävän kuitenkin lähinnä tekstiä luonnollisia kieliä hyödyntäen (Kassab, Neill & Laplante, 2014; Neill & Laplante, 2003). Tekstimuotoiset käyttäjätarinat ovat ketterän ohjelmistokehityksen myötä yleistyneet (Schön ym., 2017) mutta niitä on kritisoitu monitulkintaisuudesta ja kokonaiskuvan puutteesta (Robeer, Lucassen, van der Werf, Dalpiaz & Brinkkemper, 2016). Myös ketteryyteen yhdistettävä arkkitehtuurin suunnittelun laiminlyöminen on saanut paljon kritiikkiä osakseen (Babar, 2009). Siitä, mikä mahtaa olla mallintamisen osuus näissä analyysiin ja suunnitteluun liittyvissä käytännöissä, on ristiriitaista tietoa (ks. esim. Chaudron ym., 2012; Badreddin ym., 2013), eikä UML:n asemasta alalla ole muodostunut yhtenäistä kuvaa (Chaudron, 2017).

Empiirisen tutkimuksen toteuttaminen tämän tutkielman yhteydessä perustellaan mallintamisen oletetulla hyödyllisyydellä, mutta ristiriitaisilla käytännön todisteilla ja epätietoisuudella UML:n ja ketterän kehityksen vaikutuksesta mallintamiskäytäntöihin. Suomessa ammattilaisten näkemyksiä ja kokemuksia mallintamisesta ei ole vastaavalla tutkimuksella kartoitettu, eikä käytössä olevista menetelmistä tai työkaluista ole saatavilla ajantasaista tilastoa. Iivari (1995) tutki tekijöitä, jotka vaikuttavat näkemyksiin CASE-työkalujen tehokkuudesta ja totesi työkalujen käyttöönottotason (ja kuten Davies ym. (2006) esittävät, oletettavasti myös mallintamisen) suomalaisissa organisaatioissa matalaksi. Tutkimus on kuitenkin vanhentunut, etenkin kun otetaan huomioon teknologian kehityksen ja muuttuvien liiketoimintavaatimusten (ks. Bork ym., 2019; Messe ym., 2020) lisäksi ketteryys, joka on merkittävästi vaikuttanut tietojärjestelmäkehitykseen 2000-luvun alkupuolelta lähtien (Ghobadi & Mathiassen, 2016).

Luvussa 3.2 esiteltiin melko tuoreitakin tutkimuksia ammattilaisten näkemyksistä ja mallintamiskäytännöistä ympäri maailman. Näiden tutkimusten tulosten perusteella ei kuitenkaan voida niiden ristiriitaisuuden ja alueellisten erojen sekä erilaisten tutkimusasetelmien vuoksi tehdä päätelmiä Suomessa toimivien mallintajien näkemyksistä ja kokemuksista. Tällä empiirisellä tutkimuksella pyritään selvittämään, miksi ammattilaiset mallintavat, mitä menetelmiä ja työkaluja he käyttävät, ja mitä näkemyksiä heillä on mallintamiseen liittyvistä hyödyistä ja haasteista. Petren (2013; 2014) tavoin todetaan, että näitä näkemyksiä ja kokemuksia kartoittamalla tuotetaan tietojärjestelmä-, menetelmä- ja työkalukehittäjille tietoa, jonka esiintuomista pitävät arvokkaana myös ammattilaiset itse. Näin vältetään mallintamisen tutkimusta pitkään vaivannut empiiristen todisteiden puute ja huomioidaan alan jatkuva kehitys, joka edellyttää tuoreita raportteja käytännöistä.

4.2 Tutkimusote ja tiedonkeruumenetelmä

Tämän tutkimuksen tavoitteena on selvittää mallintamisen nykytilaa ilmiön syvemmän ymmärryksen kautta, joten tutkimusotteeksi valikoitui laadullinen tutkimus. Mallintamisen käytäntöjä ja niihin liittyviä tarpeita ja haasteita on

aiemmin kartoitettu lähinnä määrällisillä kyselytutkimuksilla (ks. esim. Davies ym., 2006**)**. Ilmiön tarkastelu laadullisin menetelmin on jäänyt vähemmälle huomiolle. Tällaisista merkittävistä tutkimuksista löytyy melko harvinaisena esimerkkinä luvussa 3.2 esitelty Petren (2013) haastattelututkimus. Laadullisen tutkimusotteen valinta tähän tutkielmaan on siis erityisen perusteltua, etenkin kun ammattilaiset itse näkevät laadullisen käytännön tutkimuksen merkityksellisenä tieteenalalle (ks. Petre, 2014). Tavoitteena ei ole määrälliseen tutkimukseen liitettävä tulosten yleistettävyys tai olemassa olevien hypoteesien testaaminen. Tarkoituksena on ymmärtää syvällisemmin, miksi ammattilaiset mallintavat tai miksi eivät mallinna, sekä tarjota mahdollisesti täysin uusia ja ajankohtaisia havaintoja, jollaisia ei ole tuotu esiin aiemmissa tutkimuksissa. (ks. Ghauri & Grønhaug, 2005, s. 112; Hirsjärvi, Remes, Sajavaara & Sinivuori., 2009, s. 161)

Haastattelu on keskeinen tiedonkeruumenetelmä laadullisessa tutkimuksessa (Hirsjärvi & Hurme, 2000). Koska tämän empiirisen tutkimuksen keskiössä on mallintamiskäytäntöjen ja ammattilaisten kokemusten ja näkemysten selvittäminen, oli näiden ammattilaisten haastatteleminen luonnollinen valinta tiedonkeruumenetelmäksi. Kyse on siis eri rooleissa toimivista asiantuntijoista, joiden haastattelemisessa on Barriballin ja Whilen (1994) mukaan hyödyllistä soveltaa puolistrukturoitua haastattelumenetelmää. Menetelmä mahdollistaa olennaisten jatkokysymysten esittämisen ja syvällisten näkemysten tavoittamisen monimutkaisista ilmiöistä (Barriball & While, 1994; Ghauri & Grønhaug, 2005, p. 112). Puolistrukturoidusta haastattelumuodosta on useita määritelmiä (ks. Hirsjärvi & Hurme, 2000, s. 47) ja niihin tutustuttua päädyttiin Hirsjärven ja Hurmen (2000) esittämään teemahaastatteluun. Tässä haastattelumuodossa yksi haastattelun näkökulma, haastattelun aihepiirit eli teema-alueet, on jokaisessa haastattelussa sama.

Haastattelurungon rakentaminen aloitettiin kattavan kirjallisuuskatsauksen kirjoittamisen jälkeen. Etenkin käytännön tutkimuksiin (esim. Badreddin ym., 2018; Petre, 2013) tutustuttiin huolella ja niistä johdettiin mahdollisia haastattelukysymyksiä. Tällä pyrittiin varmistamaan olennaisten aiheiden käsittely sekä sujuva reflektointi myöhemmin. Muodostetuista kysymyksistä ja kiinnostavista aiheista hahmoteltiin esiin nousevat teemat, jotka pysyivät muuttumattomina koko haastatteluprosessin ajan. Esitettävät kysymykset saivat vaihdella sen mukaan, mitä haastateltavat kertoivat ja mitkä olivat heidän mielipiteensä sekä kokemuksensa käsiteltävistä teemoista. Jotkin kysymykset oli tarkoitus esittää jokaisessa haastattelussa, kuten tutkimuksen kannalta kriittisimmät kysymykset eli käytössä olevat kaaviotekniikat ja syyt mallien käytölle. Haastattelujen teemat olivat seuraavanlaiset:

1. Haastateltavan taustatiedot
    - syntymäaika, koulutustausta, työtehtävä, kokemusvuodet
2. Prosessit ja projektit
    - millaisissa projekteissa työskentelee
    - millaista ohjelmistoa/tietojärjestelmää kehittää

   - millaista dokumentaatiota tuotetaan
   - miten mallintamispäätökset tehdään
3. Näkemykset mallintamisesta
   - miten tärkeänä pitää mallin merkitystä kehitystyössä
   - miten vastaa organisaation tai alan näkemystä
   - mitä mieltä UML:stä ja mallipohjaisesta kehityksestä
   - miksi mallinnetaan, miksi ei
4. Käytössä olevat tunnetut kaaviotekniikat ja työkalut
   - mitä kaaviotekniikoita käyttää, miten, miksi
   - mitä työkaluja käyttää, miten, miksi
5. Hyödyt ja haasteet
   - mallintamiseen ja mallien käyttöön liittyvät hyödyt ja haasteet
   - kaaviotekniikoiden ja työkalujen käyttöön liittyvät hyödyt ja haasteet
6. Vaihtoehtoiset tavat
   - mitä muita tapoja on käytössä, miten, miksi

Haastatteluteemoja ja -kysymyksiä muodostettaessa pyrittiin miettimään ja välttämään mahdollisia ongelmakohtia. Kirjallisuuskatsausta tehtäessä huomattiin, että malleihin ja mallintamiseen liittyvät termit ovat moninaisia ja tutkijat tulkitsevat niitä monin eri tavoin. Empiirisissä tutkimuksissa on havaittu, että myös ammattilaisilla on monenlaisia tulkintoja malleista (ks. esim. Forward ym., 2010). Väärinkäsityksiltä pyrittiin välttymään esittämällä tutkimuksen tavoitteet mahdollisimman selkeästi sekä valitsemaan haastattelua varten yksiselitteisiä ja käytännöllisiä termejä. Tämän vuoksi teemoissa ja kysymyksissä käytettiin esimerkiksi puoliformaaleista mallinnuskielistä termiä kaaviotekniikka. Termi kaavio viittaa selkeästi graafiseen kuvaan ja malliin, eikä esimerkiksi tekstimuotoiseen tai matemaattiseen mallintamiseen.

4.3 Haastateltavien hankinta

Tutkimuksen tavoitteena oli saada kattava kuva nykypäivän tietojärjestelmäkehityksen mallintamiskäytännöistä, joten haastateltaviksi tavoiteltiin eri työtehtävissä toimivia mallintajia useasta eri yrityksestä. Empiirinen tutkimus rajattiin tietojärjestelmiä ja IT-palveluita tuottaviin organisaatioihin, joten sen ulkopuolelle jätettiin ammattilaiset, jotka kehittivät tietojärjestelmiä yrityksen omaan käyttöön tai jotka toimivat ainoastaan kokonaisarkkitehtuuritasolla. Haastateltaviksi ei myöskään tavoiteltu ammattilaisia, jotka toimittivat turvallisuuskriittisiä (engl. safety-critical) tietojärjestelmiä. Tutkimuksessa tarkasteltiin mallintamisen käytäntöjä ja hyödyllisyyttä ympäristöissä, joissa se oli vapaaehtoista, ja turvallisuuskriittisten järjestelmien kehittämisessä edellytetään tiettyjen säädösten ja standardien noudattamista (Nair, De La Vara, Sabetzadeh & Briand, 2014).

Laadullisessa tutkimuksessa ei pyritä tilastolliseen yleistettävyyteen vaan ilmiön syvempään ymmärrykseen ja uusien näkökulmien löytämiseen, joten haastateltavien valinnassa käytettiin satunnaisen otannan sijaan harkinnanvaraista otantaa (ks. Hirsjärvi & Hurme, 2000, s. 59; Tuomi & Sarajärvi, 2018). Haastateltavien saavutettavuus ja soveltuvuus huomioiden tutkimuskohteeksi valittiin Suomessa toimivien merkittävien ohjelmisto- ja IT-palveluyritysten mallintajat. Aiempien käytännön tutkimusten perusteella mallintamismenetelmien käyttö on yleisempää suuremmissa organisaatioissa ja projekteissa (ks. esim. Davies ym., 2006; Fettke ym., 2009; Torchiano ym., 2011), joten haastateltaviksi päätettiin olla tavoittelematta pienempien kuin keskisuurten yritysten mallintajia. Rajauksella pyrittiin takaamaan se, että haastateltavat työskentelisivät sen suuruusluokan projekteissa, joissa mallintaminen koetaan perustelluksi ja näin saadaan tutkimuksen kannalta olennaista tietoa.

Tutkielmassa keskitytään perinteiseen tietojärjestelmäkehitykseen, jossa mallintamistyötä suoritetaan analyysin ja suunnittelun aikana. Malleja voidaan kuitenkin hyödyntää muissakin tehtävissä, ja käytön hyödyt ja haasteet voivat vaikuttaa moniin kehitystyön ja projektien osa-alueisiin. Mahdollisimman kattavan kokonaiskuvan saavuttamiseksi haastateltaviksi tavoiteltiinkin eri työtehtävissä olevia IT-ammattilaisia. Harkinnanvaraisen otannan vuoksi varsinkin alussa haastateltavien hankinta painottui ohjelmistoarkkitehteihin, joiden oletetaan hyödyntävän mallintamista muiden suunnittelualojen arkkitehtien tapaan (ks. esim. Pressman & Maxim, 2015, s. 869). Hankinnan edetessä tavoiteltiin myös esimerkiksi tuotepäälliköitä, jotka asiakasvaatimusten asiantuntijoina (ks. Ebert, 2014) voivat oletettavasti suorittaa mallintamista analyysissä. Monipuolisten näkemysten saavuttamiseksi haastateltaviksi pyydettiin myös konsultteja, projektipäälliköitä, ohjelmistokehittäjiä ja johtajia. Edellytyksenä haastateltaville oli työtehtävästä riippumatta omakohtainen kokemus mallintamistyöstä.

Haastateltavien tavoittelu aloitettiin tutkijan ja tutkielman ohjaajan kontakteista. Näiden jo tiedossa olevien potentiaalisten haastateltavien lisäksi käytettiin lumipallomenetelmää (engl. snowballing technique, snowball sampling), jossa tutkimuskohde auttaa uusien kohteiden hankinnassa esimerkiksi ehdottamalla mielestään sopivaa henkilöä, johon tutkijan tulisi ottaa yhteyttä (Biernacki & Waldorf, 1981). Menetelmä on todettu hyödylliseksi tutkittavasta ilmiöstä tarvittavan kokemuksen ja tietämyksen omaavien ammattilaisten tavoittamisessa (Miles & Huberman, 1994, s. 28), joten sen avulla uskottiin saavutettavan soveltuva ja motivoitunut haastateltavien joukko. Lumipalloilulla on myös rajoitteensa, sillä sen avulla saavutetaan haastateltavia vain tietyn verkoston sisällä (Biernacki & Waldorf, 1981).

Kattavan haastatteluotoksen saavuttamiseksi omien kontaktien ja lumipallomenetelmän lisäksi sopivia organisaatioita haettiin Ite wikin verkkosivustolta ja potentiaalisia haastateltavia LinkedInin verkkoyhteisöpalvelusta. Kaikkiin haastatteluehdokkaisiin tutkija otti yhteyttä puhelimitse tai sähköpostitse. Aineisto luvattiin käsitellä siten, että haastateltavia tai yrityksiä ei voida raportista tunnistaa. Tutkimuksesta kerrottiin avoimesti sen tavoitteet ja haastatteluteemat sekä haluttaessa myös esimerkkejä haastattelussa esitettävistä kysymyksis-

tä. Tutkimuksen lähtökohtana mainittiin UML:n sisältämät kaaviotekniikat niiden standardiaseman vuoksi, mutta tavoitteena painotettiin olevan todellisten mallintamiskäytäntöjen kartoittaminen, eikä haastateltavilta vaadittu UML:n käyttöä. Tästä huolimatta osa ehdokkaista kieltäytyi haastattelusta UML:n rajallisen tuntemuksen tai vähäisen käytön vuoksi.

Vaikka tutkimuksen kohderyhmäksi määritettiin keskisuurten ja suurten organisaatioiden perinteisen tietojärjestelmäkehityksen mallintajat, haastatteluun valikoitui myös yksi poikkeus. Lumipallomenetelmän ansiosta yksi mallipohjaisen tietojärjestelmäkehityksen ammattilainen osoitti mielenkiintonsa tutkimusta kohtaan. Mallipohjaisuuden tarkempi tutkiminen ei kuulunut alkuperäiseen tutkimussuunnitelmaan, mutta tilaisuutta ei haluttu jättää väliin. Tutkimuksen tavoitteen ollessa todellisten mallintamiskäytäntöjen kartoittaminen, myös mallipohjaisuus tuli ottaa huomioon. Haastateltavan kanssa keskusteltiin mahdollisesta tunnistettavuudesta, koska organisaation toiminta oli hyvin spesifiä muihin verrattuna. Haastateltava ei kuitenkaan nähnyt asiaa ongelmana, vaan suostui haastatteluun ja sen raportointiin.

4.4 Haastattelujen toteutus

Tutkielmaa varten haastateltiin 13:a tietojärjestelmäkehityksessä mallintamistyötä tekevää ammattilaista kymmenestä eri organisaatiosta. Haastattelut suoritettiin marraskuun 2019 ja helmikuun 2020 välisenä aikana. Osallistujia kehotettiin varaamaan haastattelua varten vähintään tunti aikaa rauhallisessa ympäristössä. Tällä ohjeistuksella pyrittiin välttämään yleiset haastatteluihin liittyvät häiriötekijät, jotka johtuvat ajan loppumisesta kesken ja levottomasta haastatteluympäristöstä (ks. Easton, McComish & Greenberg, 2000; Myers & Newman 2007). Kaikki haastattelut saatiinkin käytyä kiireettä läpi ilman taustahälyä tai häiritseviä keskeytyksiä.

Kolmestatoista haastattelusta yksitoista suoritettiin kasvokkain ja kaksi Microsoft Teamsin välityksellä. Vaikka haastatteluja kasvotusten pidetään yleisesti varmimpana tapana laadullisen haastatteluaineiston saavuttamiseksi (Schober, 2018), todettiin myös Teamsin kautta suoritetut haastattelut onnistuneiksi. Teamsin avulla saatiin arvokasta aineistoa, joka muutoin ei välttämättä olisi ollut mahdollista aika- ja kustannusrajoitusten vuoksi. Haastateltavat pystyivät myös valitsemaan heille varmasti sopivimman ajankohdan. Teamsin välityksellä tehdyt haastattelut olivatkin pisimpien joukossa.

Haastattelujen kestot vaihtelivat 43—90 minuutin välillä sen mukaan, miten paljon haastateltavilla oli kerrottavaa ja kuinka polveilevaa keskustelu oli. Keskimääräinen haastattelun kesto oli 66 minuuttia. Välillä haastateltavat kertoivat työstään tai alan toimintatavoista vuosien ja vuosikymmentenkin takaa. Vaikka näitä osuuksia ei voinut ottaa huomioon nykypäivän käytäntöjen raportoinnissa, toimivat ne kuitenkin havainnollisina taustatietoina haastateltavien näkemyksille. Haastattelun tukena käytettiin tarvittaessa UML:n kaaviojaotte-

lua ja kuvia eri kaavioista. Tällä pyrittiin välttämään erilaisten termien käytöstä johtuvia väärinkäsityksiä.

Kaikki haastattelut äänitettiin kahdella eri laitteella. Toimenpiteen avulla vältettiin yksi ilmeisimmistä haastattelun sudenkuopista, äänityksen epäonnistuminen laitevian vuoksi (Easton ym., 2000). Jokaisen haastattelun päätteeksi haastateltavalle tarjottiin mahdollisuutta lukea litteroitu haastattelu ja esittää siihen kommentteja sekä pyydettiin lupa esittää jälkikäteen tarkentavia kysymyksiä mahdollisesti epäselviksi jääneistä osuuksista. Kaikki eivät nähneet tarpeellisena litteroinnin lukemista, mutta osa haastateltavista toivoi, että heille lähetettäisiin lopulliseen raporttiin tulevat suorat lainaukset tarkistettaviksi. Litteroidut haastattelut tai sitaatit lähetettiin sähköpostitse halukkaille. Yhden haastateltavan sitaatteihin tehtiin pieniä muutoksia anonymiteetin suojaamiseksi. Kolmelle haastateltavalle esitettiin tarkentavia kysymyksiä sähköpostitse ja näin varmistettiin raportissa esitettävien asioiden paikkansapitävyyttä.

Haastattelujen lukumäärää ei lyöty lukkoon etukäteen. Tutkimuskohteita vasta tavoitellessa ei voitu tietää, miten kattavaan otokseen päädyttäisiin, joten tilannetta oli tarkasteltava haastattelujen edetessä. Jo muutaman haastattelun jälkeen huomattiin tiettyjen teemojen toistuminen. Saturaation eli kyllääntymispisteen tavoittelu voi olla laadullisessa tutkimuksessa ongelmallista jokaisen tapauksen oletetun ainutlaatuisuuden vuoksi (Hirsjärvi & Hurme, 2000, s. 60), eikä haastattelujen lukumäärää tässäkään tutkimuksessa pohjattu siihen. Teemojen toistuminen otettiin kuitenkin huomioon jatkoa suunnitellessa, koska aineiston kokoa ei voinut kasvattaa hallitsemattomiin mittoihin. Kvale (1996, s. 102) huomauttaa, että jos haastateltavia on liikaa, on syvällisten tulkintojen teko vaikeaa. Haastattelut päätettiin lopettaa 13:nnen jälkeen, koska siihen mennessä oli haastateltu useassa eri organisaatiossa ja roolissa toimivia mallintajia, joilla oli kokemusta erilaisista projekteista ja tietojärjestelmistä. Lisäksi oli raportoitu erilaisia näkemyksiä, mutta havaittu myös toistuvia teemoja.

4.5 Aineiston analyysi

Tämän tutkielman tavoitteena ei ollut ainoastaan kartoittaa nykypäivän tietojärjestelmien mallintamisen käytäntöjä ja verrata niitä aiempiin tutkimuksiin, vaan myös syvemmin ymmärtää mahdollisesti yllättäviäkin syitä käytäntöjen takana. Analyysin pohjaksi ei siten valittu aiemmista tutkimuksista johdettua viitekehystä, vaan aineistosta pyrittiin etsimään mallintamiseen liittyviä tarpeita, hyötyjä ja haasteita ilman etukäteen muodostettuja odotuksia. Täysin aineistolähtöiseksi analyysiä ei voi kuitenkaan kutsua, koska mallintaminen on alalla hyvin tunnettu aihealue ja aiemmat tutkimukset vaikuttivat vahvasti jo tiedonkeruuvaiheeseen. Tässä tutkimuksessa voi siten puhtaan induktiivisuuden sijaan puhua paremminkin abduktiivisuudesta eli teoriaohjaavuudesta, jolloin analyysiin vaikuttaa myös jo tiedossa olevat lähtökohdat (Hirsjärvi & Hurme, 2000, s. 136; Tuomi & Sarajärvi, 2018).

Haastatteluaineiston analyysi aloitettiin jo ennen kaikkien haastattelujen suorittamista loppuun. Näin analyysi vaikutti myös tiedonkeruuseen, mikä on Milesin ja Hubermanin (1994) mukaan tyypillistä laadulliselle tutkimukselle. Haastatteluaineiston litterointi suoritettiin mahdollisimman pian jokaisen haastattelun jälkeen. Haastattelut litteroitiin sanatarkasti, jotta mitään tärkeää näkökulmaa ei jäisi huomaamatta. Tallennetta kuunneltiin ja tekstiä luettiin useaan otteeseen, millä pyrittiin kokonaisuuden hahmottamiseen ja yhteyksien näkemiseen ilman hätäisiin johtopäätöksiin turvautumista (ks. Alhojailan, 2012).

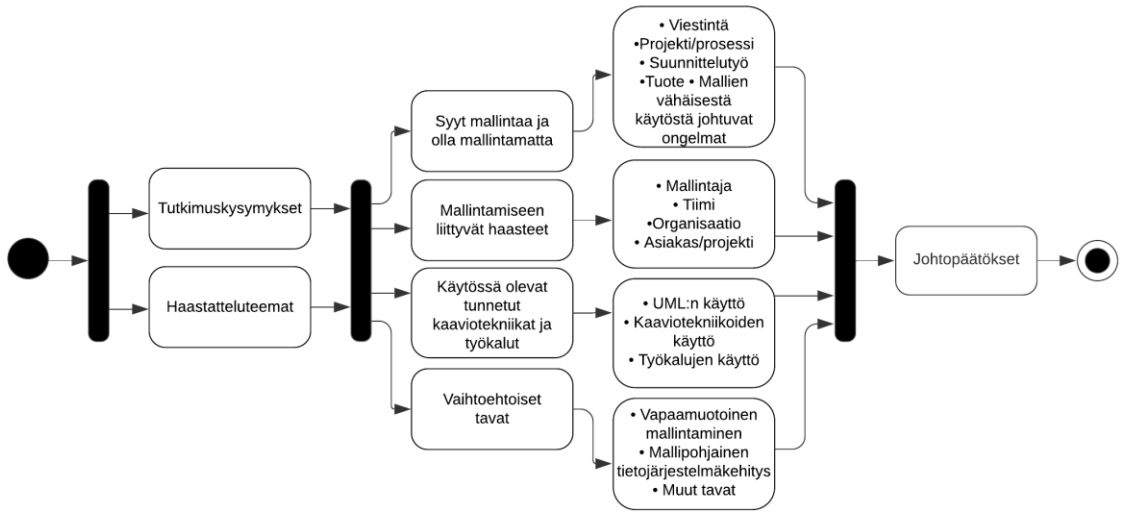
Analyysi aloitettiin tekstinkäsittelyohjelmalla, tekemällä merkintöjä litteroituun tekstiin, korostamalla esiin nousevia mielenkiintoisia ja odottamattomiakin näkökulmia. Seuraavissa haastatteluissa näihin näkökulmiin voitiin paneutua enemmän tai vastaavasti häivyttää vähemmän olennaisia aihealueita. Haastattelumetodi vahvistui prosessin aikana ja auttoi saavuttamaan entistä merkittävämpää tutkimustietoa. Haastatteluteemat säilyivät kuitenkin muuttumattomina koko empiirisen tutkimuksen ajan ja niistä kaikki käsiteltiin jokaisen haastattelun aikana, vaikka kysymykset ja painopisteet vaihtelivatkin haastattelusta toiseen.

Teemoittelu havaittiin luontevaksi tavaksi saada aineiston analyysi liikkeelle. Aineisto pilkottiin ja järjestettiin uudelleen haastatteluteemojen mukaan. Näin pyrittiin helpottamaan ja selkeyttämään seuraavaksi suoritettavaa tarkempaa analyysiä. (ks. Tuomi & Sarajärvi, 2018.) Hirsjärvi ja Hurme (2000, s. 173) painottavat, että pelkissä haastattelun teemoissa ei tule pitäytyä, vaan analyysin aikana esiin nousevat muut teemat ovat usein näitä lähtöteemoja kiinnostavampia. Tässä vaiheessa analyysiä havaittiinkin jo runsaasti uusia teemoja, jotka esiintyivät useammassa haastattelussa. Koska aineistoa ei haluttu pilkkoa liikaa mahdollisen kontekstin menettämisen ja sivuraiteille joutumisen vuoksi, alkoi aineisto kertautua määrällisesti välillä hallitsemattomalta vaikuttaviin mittoihin. Aineistoa läpikäytiinkin sekä alkuperäismuodossaan että teemoihin jäsenneltynä vielä useaan otteeseen analyysin aikana.

Aineiston koodaamisen tukena käytettiin tekstinkäsittelyohjelman lisäksi taulukkolaskentaohjelmaa. Koodeja muodostettiin systemaattisesti ja koko haastatteluaineisto kattaen, jotta epähuomiossa mikään olennainen osa-alue ei jäisi tarkastelematta. Koodit jaettiin ensin haastatteluteemojen ja tämän jälkeen myös esiin nousevien uusien teemojen mukaisesti omiin tiedostoihin, joiden alle eri aliteemoja koodattiin. Koodeja läpikäytiin useaan kertaan niitä muokaten, yhdistellen tai jakaen ja merkittäviä säännönmukaisuuksia sekä eroavaisuuksia etsien, välillä myös verraten niitä haastattelujen litterointeihin.

Kuviossa 10 esitetään aineiston analyysiprosessin eteneminen koodauksesta johtopäätöksiin. Samalla, kun aineisto jaettiin ja koodattiin haastatteluteemojen alle, pohdittiin tutkimuskysymyksiin vastaamista. Näin koodatusta aineistosta muodostettiin neljä pääteemaa, joiden avulla voidaan vastata tutkimuskysymyksiin. Teoriaohjaavaa analyysiä mukaillen nämä pääteemat (*syyt mallintaa ja olla mallintamatta*, *mallintamiseen liittyvät haasteet*, *käytössä olevat tunnetut kaaviotekniikat ja työkalut*, *vaihtoehtoiset tavat*) ovat sidoksissa teoriasta muodostettuihin haastatteluteemoihin, kun taas niiden alakategoriat on muo-

dostettu aineistolähtöisesti (ks. Tuomi & Sarajärvi, 2018). Haastatteluaineistosta nousi esiin jokaiseen pääteemaan liittyen kolmesta viiteen alakategoriaa, joihin tutkimuksen kannalta olennaiset ja mielenkiintoiset tulokset luokiteltiin.



KUVIO 10 Aineiston analyysiprosessin eteneminen

Pääteemojen ja alakategorioiden muodostamisen sekä luokittelun jälkeen analyysiä jatkettiin sisällön erittelyllä eli kvantifioimalla aineistoa (ks. Tuomi & Sarajärvi, 2018). Alakategorioihin aineistolähtöisesti muodostettujen luokkien esiintymistä aineistossa laskettiin sen mukaan, miten moni haastateltava mainitsi saman asian. Näin oli mahdollista esittää tiettyjen luokkien suhteellista merkitsevyyttä tämän aineiston sisällä ja johtopäätösten osalta, siitäkin huolimatta, että myös yksittäinen näkemys voi olla tutkimuksen kannalta arvokas. Numeerisesta esitystavasta huolimatta tarkoituksena ei siis ole määrälliseen tutkimukseen liittyvä tilastollisen merkitsevyyden esittäminen.

Luokittelun ja sisällön erittelyn lisäksi aineistosta esiin tulevia asioita tarkasteltiin suhteessa toisiinsa, mikä on Hirsjärven ja Hurmeen (2000, s. 174) mukaan analyysin mahdollisesti olennaisin osa. Analyysissä hyödynnettiin tyypittelyä, jossa aineistoa pyrittiin ryhmittelemään tiettyjen yhteisten tekijöiden, kuten mallintamismenetelmiin liittyvien haastateltavien näkemysten ja käytäntöjen perusteella (ks. Miles & Huberman, 1994). Luvussa 5.4.1 esitetään tutkimuksen kannalta keskeistä tyypittelyä, kun haastatteluissa esiintyvien yhteyksien tarkastelun kautta määritetään UML:n käytölle kaksi erilaista tapaa.

4.6 Reliabiliteetti ja validiteetti

Laadullisessa tutkimuksessa reliaabeliutta ja validiutta tulkitaan monin eri tavoin (Hirsjärvi ym., 2009, s. 232). Reliabiliteetti eli luotettavuus ja validiteetti eli pätevyys ovat määrällisestä tutkimuksesta lähtöisin olevia käsitteitä ja ne yhdis-

tetään yleensä asioiden mittaamiseen (Hirsjärvi & Hurme, 2000, s. 185—186). Käsitteet eivät siis ole laadullisen tutkimuksen kanssa täysin yhteensopivia eikä niitä tulisi soveltaa sellaisenaan (Hirsjärvi ym., 2009, s. 232). Etenkin validiteetti on laadullisen tutkimuksen yhteydessä epäselvä käsite (Wolcott, 1995) ja haastattelut voidaan käsittää niin ainutlaatuisiksi, että perinteiset, määrälliseen tutkimukseen tarkoitetut arviointitavat eivät yksinkertaisesti päde (Holstein & Gubrium, 1995, s. 9). Tutkimuksen luotettavuutta ja pätevyyttä tulee kuitenkin arvioida jollain tapaa, jotta täytetään tieteellisen tutkimuksen kriteerit (Hirsjärvi ym. 2009, s. 232; Long & Johnson, 2000).

Tutkimusmenetelmäkirjallisuudessa esitetään ratkaisuja, joiden kautta reliabiliteettia ja validiteettia voidaan soveltaa laadulliseen tutkimukseen ja näin nostaa sen laadukkuutta ja tieteellistä arvoa. Laadullisen tutkimuksen reliabiliteettia voidaan arvioida sen kautta, ovatko tulokset johdonmukaisia ja luotettavia (Merriam, 1995; Noble & Smith, 2015). Tutkija voi parantaa luotettavuutta kuvaamalla tarkasti tutkimusprosessinsa kulun (Hirsjärvi ym., 2009, s. 232), jolloin hänen tekemänsä valinnat ovat lukijalle läpinäkyviä ja selkeitä (Merriam, 1995; Noble & Smith, 2015). Validiteettia voidaan arvioida sen perusteella, miten hyvin tutkijan esittämät kuvaukset ja niistä johdetut tulkinnat ovat keskenään yhteensopivia ja siten uskottavia (Hirsjärvi & Hurme, 2000, s. 189). Sekä luotettavuus että uskottavuus ovat siten parhaiten arvioitavissa, kun tutkija esittää tutkimuksensa kulun avoimesti ja perustelee menettelynsä vakuuttavasti.

Tässä tutkimuksessa on pyritty mahdollisimman tarkkaan tutkimusprosessin kuvaamiseen. Tarkoituksena on tehdä lukijalle selväksi, miten tutkija on päätynyt valintoihinsa ja toisaalta, miksi ja mitä asioita hän on jättänyt käsittelemättä tai millä perusteilla sulkenut tiettyjä vaihtoehtoja pois (ks. Kvale, 1996). Kuten aiemmissa alaluvuissa kerrottiin, tutkija on tietoinen haastattelututkimukseen liittyvistä sudenkuopista, joita on pyrkinyt välttämään niihin valmistautumalla. Myös analyysi on pyritty suorittamaan systemaattisesti. On kuitenkin huomioitava, että haastattelujen lopputulos on riippuvainen haastattelijan ja haastateltavan vuorovaikutuksesta (Hirsjärvi & Hurme, 2000, s. 189) ja sama aineisto voidaan tulkita monin eri tavoin (Tuomi & Sarajärvi, 2018), joten varsinkin kokematon tutkija voi vaikuttaa tutkimuksen kulkuun myös ei-toivotulla tavalla. Totuudenmukaisella kerronnalla on pyritty avustamaan lukijaa luotettavuuden ja uskottavuuden arvioinnissa.

Tutkimuksen tiedonkeruun osalta nojataan Petren (2013) tavoin siihen, että asiantuntijahaastatteluiden avulla saavutetaan luotettavaa tietoa. Laadullisessa tutkimuksessa luotettavuus on kuitenkin tutkittavien sijasta enemmän tutkijan toimintaan yhdistettävää (Hirsjärvi & Hurme, 2000, s. 189), joten tutkijan tulee avata käyttämiään keinoja tulosten tuottamisessa. Mahdollisia väärinkäsityksiä on pyritty ehkäisemään haastatteluissa käytettävän terminologian käytännöllisyydellä, esittämällä haastatteluissa tarvittaessa oppikirjakuvia kaavioista ja tarjoamalla haastateltaville mahdollisuutta kommentoida tekstiä sekä selvittämällä epäselväksi jääneitä osuuksia haastateltavilta jälkikäteen. Näillä toimilla on pyritty lisäämään tutkimuksen uskottavuutta (ks. Long & Johnson, 2000).

Lopulliseen raporttiin on tulosten yhteyteen lisätty haastatteluista sitaatteja tarpeen mukaan. Näillä suorilla lainauksilla tuodaan haastateltavien ääni kuuluviin ja vahvistetaan tutkijan argumentointia (Hirsjärvi & Hurme, 2000, s. 194) ja tällä tavoin annetaan lukijalle mahdollisuus arvioida tutkijan päättelyprosessia. Suorien lainausten esittämisen yhteydessä on pyritty miettimään myös tasapainoa avoimen raportoinnin ja tarpeettoman yksilöinnin välillä. Kun suorien lainausten yhdistäminen tiettyyn haastateltavaan ei ole katsottu yhtä tarpeelliseksi kuin anonymiteetin säilyttäminen, on lainauksista jätetty pois tieto siitä, mistä haastatteluista ne ovat peräisin.

Kaikkien raportissa esitettävien tulosten kohdalla on käytetty samaa harkintaa kuin haastattelusitaattien esittämisessä. Raportissa ei esitetä sellaisia tuloksia tai yksittäisiä tietoja, jotka eivät ole tutkimuksen kannalta välttämättömiä ja jotka voisivat muihin tietoihin yhdistettäessä kyseenalaistaa haastateltavan anonyyminä pysymisen. Tällaiseksi tiedoksi arvioitiin mm. haastateltavien syntymävuodet, joiden ei koettu tuovan lisäarvoa kokemusvuosien raportoinnin lisäksi. Seuraavassa pääluvussa esitetään haastattelututkimuksen tulokset.

5 TUTKIMUSTULOKSET

Tässä pääluvussa esitetään tutkielman yhteydessä toteutetun haastattelututkimuksen tulokset. Ensimmäisessä alaluvussa esitellään haastateltavat, jonka jälkeen siirrytään varsinaisiin tulosteemoihin, joiden käsittelyssä hyödynnetään haastateltavien taustatietoja. Toisessa alaluvussa käydään läpi haastatteluissa esille nousseet syyt mallintaa ja olla mallintamatta sekä ongelmat, joita mallien vähäisestä käytöstä seuraa. Kolmannessa alaluvussa käsitellään mallintamiseen liittyviä haasteita. Neljännessä alaluvussa esitetään tulokset tunnettujen kaaviotekniikoiden ja työkalujen käytöstä ja viidennessä alaluvussa näiden käytölle vaihtoehtoiset tavat.

5.1 Haastateltavien esittely

Haastateltavat olivat tietojärjestelmäkehityksen eri työtehtävissä toimivia, mallintamistyötä ainakin jollain tasolla tekeviä ammattilaisia. He toimivat erilaisten toimialojen, projektien ja tietojärjestelmien parissa. Osa kehitti tuotteistettuja ratkaisuja tietyille toimialoille tai sovellusalueille, osa räätälöintejä vaihteleville asiakaskunnille. Vaikka haastateltaviksi ei tavoiteltu turvallisuuskriittisten tietojärjestelmien kehittäjiä, on joukossa ammattilaisia, jotka tarjosivat tällaisille toimialoille toimintaa tukevia palveluja ja ohjelmistoja, kuten energia-alan sähköistä asiointia. Organisaatiot, joissa haastateltavat työskentelivät, ovat Suomessa useilla paikkakunnilla toimivia ohjelmisto- ja IT-palveluyrityksiä. Suurin osa organisaatioista toimii myös Suomen ulkopuolella. Organisaatiota H (ks. taulukko 2) lukuun ottamatta organisaatiot ovat keskisuuria tai suuryrityksiä.

Työtehtävien ja projektien lisäksi haastateltavien koulutustaustat olivat vaihtelevia. Osalla mallintaminen oli ollut olennainen osa korkeakouluopintoja, osalla sitä ei ollut sisältynyt opintoihin lainkaan ja osa oli siirtynyt työelämään valmistumatta viimeisimmästä opinahjostaan. Haastateltavien työkokemus alalta vaihteli välillä 5—30 vuotta. Vaikka tutkimukseen osallistuneiden määrä on laadulliselle tutkimukselle tyypillisesti suppea määrälliseen tutkimukseen

verrattuna, tarjosivat haastateltavien kirjavat taustatiedot verrattain monipuolisen näkemyksen suomalaiseen tietojärjestelmäkehitykseen. Taulukossa 2 on tiivistettynä haastateltavien taustatiedot sekä jokaisen haastattelusta kuvaava kommentti mallintamisen ja mallien käytön merkityksestä kehitystyössä.

TAULUKKO 2 Haastateltavien taustatiedot

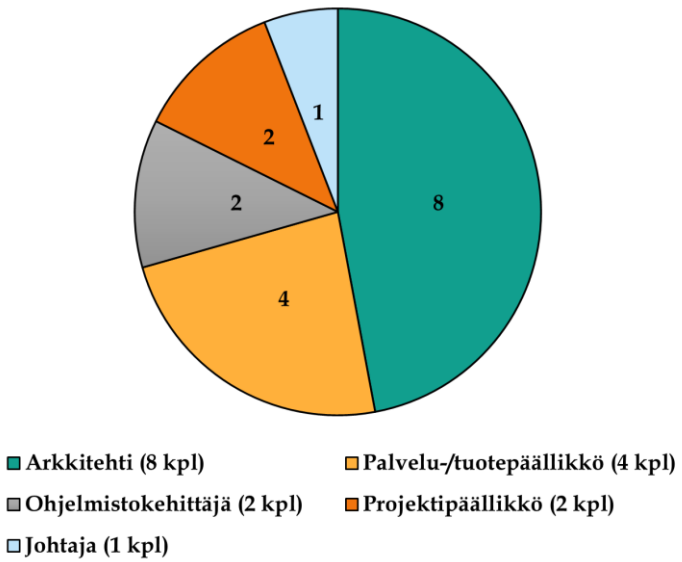
Haastattelu / organisaatio	Työtehtävä / projektirooli	Koulutus / mallintamisen opiskelu	Vuodet nykyisessä työssä / alalla	Nykyinen työ, projektit, ohjelmistot	Kommentti mallintamisen ja mallien käytön merkityksestä
H1 / org A	Ohjelmistoarkkitehti / ohjelmistokehittäjä	Ylempi korkeakoulututkinto / opinnoissa ei juurikaan; suorittanut aikoinaan organisaation tarjoamat UML-kurssit	10 / 20	Arkkitehtuurikonsultaatio; vaihtelevat asiakasprojektit; haastatteluhetkellä toimii projektissa koodaajana	"ei tässä hommassa mitä minä teen tai mitä yleensäkään ei välttämättä tällä ihan tuota, perusohjelmistokehityksen kannata. Ei ainakaan lisää niitten käyttöä."
H2 / org A	Ohjelmistoarkkitehti	Ylempi korkeakoulututkinto / opinnoissa formaalit mallintamismenetelmät; jatkokouluttautunut myös muihin tekniikoihin	9 / 25	Vaihtelevat projektit; haastatteluhetkellä kehittää asiakasprojektissa ryhmäkommunikaatiojärjestelmää	"valitettavan harva vaan sillei niinku ymmärtää sitä mallintamisen merkitystä"
H3 / org B	Ratkaisujohtaja	Korkeakoulutausta, ei valmistunut / opinnoissa jotain kursseja; työn ohessa arkkitehtuurikoulutus	1 / 13	Vastaa tarjonnasta; sisäisten prosessien kehittäminen; tuote- ja alustapohjaiset asiakasprojektit; verkkokauppa	"se on tosi tärkeetä sitten jos taas ajattelee sen projektin onnistumisen näkökulmasta. "
H4 / org C	Projekti- ja palvelupäällikkö	Ylempi korkeakoulututkinto / tietojärjestelmien mallintaminen oppiaineena, osana koulutusohjelmaa	10 / 30	Integraatio- ja API-hallinnan asiakkuuskohtainen toimitusvastuu	"käytännössä keneltä tahansa kysyy, niin asia on tärkee näin, että on tiedettävä mitä tehdään eli mehän ei saada tehtyä työtämme, jos ei tiedetä mitä pitää tehdä. -- mallithan on kuitenkin niin kun sen tiedon jakamista varten."
H5 / org D	Ohjelmistoarkkitehti	Alempi korkeakoulututkinto /opinnoissa ei; myöhemmin suorittanut tietojärjestelmäarkkitehdin valmennusohjelman	9 / 18	Käyttöliittymä- ja backend-puolen kehitys ja tietokantakehitys; hallinnolliset sovellukset	"se ois aika tärkee, tavallaan se auttaa sit niinku ehkä ymmärtämään sitä kokonaisuutta paremmin. Et meillä ehkä tehdään vähän, vähänlaisesti sitä mallinnusta, et sitä sais olla pikkusen enemmän"
H6 /org B	Tuotepäällikkö / projektipäällikkö / tiiminvetäjä / määrittelevä konsultti	Toisen asteen tutkinto / opinnoissa ei; itseopiskeluna työn ohessa	8 / 23	Energia-alan projektit ja tietojärjestelmät; käyttöönotot	"visuaalinen puoli siinä tota, mikä on sillei yksselitteinen, ni se on paljon kuvaavampaa kuin taas raamatullinen tekstiä ja tota, että siks itse suosin niinku mallintamista."
H7 / org E	Tuotepäällikkö / sovellusarkkitehti	Alempi korkeakoulututkinto /opinnoissa oli, esim. kolmannen vuoden projektissa	2 / 12	Energia-alan projektit ja tietojärjestelmät; integraatiot	"mallintamisen perusperiaate on se, että se, se toimii kommunikaatiotyövälineenä muille -- ne on niin kun helppo ja nopea tapa niin kun viestiä"
H8 / org F	Ohjelmistokehittäjä / tiiminvetäjä / teknin en tuotteenomistaja	Alempi korkeakoulututkinto / opinnoissa ainakin yksi kurssi	1 / 5	Asiakasprojektit; hyötytyökalut	"mää koen mallinnuksen viestinnälliseksi välineeks"
H9 / org G	Ratkaisuarkkitehti	Korkeakoulutausta, ei valmistunut / opinnoissa jotain kursseja; itseopiskeluna työn ohessa; organisaatiossa ollut kokonaisarkkitehtuurikoulutusta	5 / 20	Konsultointi; laajat julkisen sektorin projektit; verkkopalvelut	"mää koen sen erittäin tärkeäks, että varsinkin mitä isompi projekti on kyseessä, niin sitä tärkeämpää on se, että on jonkunlainen pohja, mihin se tekeminen perustuu."
H10 / org H	Pääarkkitehti / tiiminvetäjä	Ylempi korkeakoulututkinto / opinnoissa mm. SA/SD, OMT++, sovellusaluemallintamista; kouluttautunut myös muihin menetelmiin	5 / 20	Mallipohjainen automatisoitu sovelluskehitys; sisäinen teknologiakehitys; asiakasprojektit; liiketoiminnan järjestelmät	"mä pidän sitä tärkeämpänä kuin mitä useimmat."
H11 / org I	Tuotepäällikkö / tuotteenomistaja	Ylempi korkeakoulututkinto / opinnoissa jotain pakollisia kursseja	4 / 13	Finanssialan projektit ja tietojärjestelmät	"Niitten [liiketoimintaprosessien] kuvaaminen sanallisessa muodossa ois kyllä aivan, aivan mahdotonta, että kyllä se kaavio on siinä mielessä ihan tärkeä."
H12 / org I	Sovellusarkkitehti	Korkeakoulutausta, ei valmistunut / opinnoissa jotain kursseja	5 / 16	Finanssialan projektit ja tietojärjestelmät	"Aika vähän niitä varmaan käytetään niinku täällä meidänkin organisaatiossa. Mä vähän luulen, että aika ketterää se tekeminen nykyään on -- vaatimusmäärittelyssähän se on tosi tärkee edelleen"
H13 / org J	Ohjelmistoarkkitehti	Ylempi korkeakoulututkinto / opinnoissa UML:n perusteet ja muillakin kursseilla harjoituksia UML-lähtöisesti	5 / 10	Vaihtelevat projektit ja ratkaisut eri toimialoille; konsultointityötä; paljon integraatio-, rajapinta-, identiteetinhallinta-, pääsynhallinta- ja IoT-projekteja	"Mallihan on lähtökohtasesti mun mielestä vaan kommunikointiväline, että, että tota siis kommunikoinnista on loppupeleissä kyse. Että kaikki tietää, mitä järjestelmä tekee. Siihen se on se malli niinku, osuu."

Haastateltavista H3, H6, H8 ja H11 keskittyivät tietojärjestelmien käyttäytymispuolen mallintamiseen. Kaikki arkkitehdit mallinsivat odotetusti myös rakenteita, mutta H1 ja H13 suosivat hyvin vapaamuotoisia tapoja mallintaessaan. H1

kertoi mallintavansa vähän, H13 paljon. Haastateltavista H2, H4 ja H10 olivat saaneet mallintamista ja menetelmällisyyttä korostavan koulutuksen. Haastateltavan H9 tavoin he painottivat mallintamisen merkitystä ja hyödynsivät mallintamista työssään muita haastateltavia enemmän, varsinkin jos tarkastellaan formaalimpia tapoja vapaamuotoisen mallintamisen sijaan. Nämä haastateltavat olivat myös kokeneimpien ammattilaisten joukossa, mutta kokemusvuodet eivät tämän haastatteluaineiston perusteella tarkoita automaattisesti mallintamisen laajaa hyödyntämistä.

H1 oli yksi haastatteluaineiston kokeneimmista, mutta ei kertomansa mukaan mallintanut kuin ”pakollisten kuvien” verran eikä myöskään käyttänyt juurikaan UML:ää. Hänen mielestään mallintamista ei useinkaan tarvita tavallisessa ohjelmistokehityksessä. Muista vähintään 20 vuoden työkokemuksen kartuttaneista arkkitehdeistä hän erosi siten, että toimi osassa projekteja arkkitehtiroolin sijaan koodaajana ja kertomansa mukaan hänen koulutukseensa ”ei juurikaan” sisältynyt mallintamista.

Kuviossa 11 esitetään haastateltavien jakautuminen eri työtehtävien ja projektiroolien mukaan. Osalla haastateltavista oli useampaan kuin yhteen kategoriaan lukeutuva työtehtävä tai rooli, joten niiden perusteella yhteenlaskettu lukumäärä on suurempi kuin haastateltavien määrä. Kuviossa ei ole esitetty sellaisia haastattelijoiden esittämiä rooleja, jotka eivät ole heidän ensisijaisia työtehtäviään tai ammattinimikkeitään eivätkä oletettavasti vaikuta merkittävästi mallintamiseen. Kuviossa ei ole huomioitu myöskään esimerkiksi haastateltavan H6 konsulttiroolia, vaikka se liittyy olennaisesti mallintamiseen. Sen ei kuitenkaan katsota eroavan hänen ensisijaisesta työtehtävästään tuotepäällikkönä, koska molemmat edustavat analyysiin liittyvää mallintamista.



KUVIO 11 Haastateltavien työtehtävät ja projektiroolit

Haastateltavista suurin osa (8 kpl) oli arkkitehtejä. Tämä on tutkimuksen kannalta otollista, koska järjestelmän rakenteen ja toiminnot suunnitteleva arkkitehti on mallintamisen kannalta kenties keskeisin rooli tietojärjestelmäkehityk-

sessä (ks. esim. Pressman & Maxim, 2015, s. 869). Arkkitehti voi analyysin ja suunnittelun lisäksi vastata mallien käytöstä myös muissa tietojärjestelmäkehityksen aktiviteeteissa, kuten ylläpidossa. Seuraavaksi eniten (4 kpl) haastateltavien joukossa oli palvelu- ja tuotepäälliköitä. Tarjoamaansa palveluun tai tuotteeseen liittyvät vaatimukset tunteva tuote- tai palvelupäällikkö edustaa asiakkaan näkemystä (ks. esim. Ebert, 2014) ja siten mallintamista etenkin analyysissä.

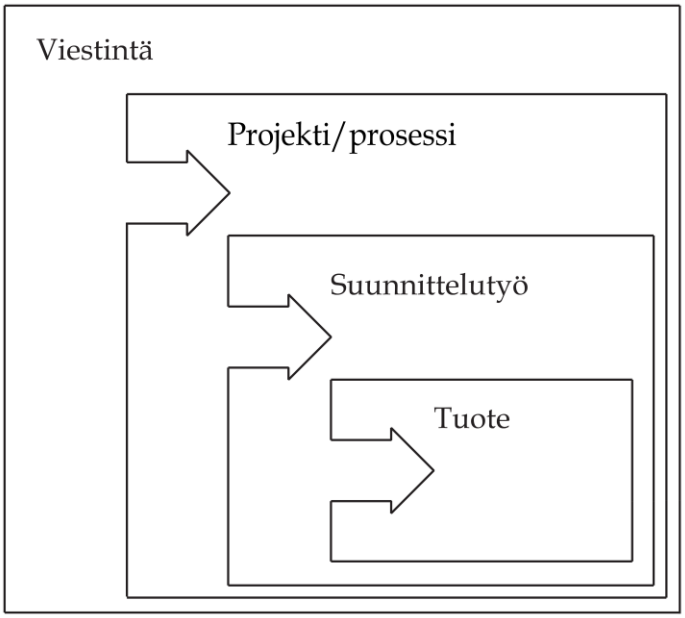
Haastateltaviin lukeutui arkkitehtien ja palvelu- ja tuotepäälliköiden lisäksi myös ohjelmistokehittäjiä (2 kpl) ja projektipäälliköitä (2 kpl). Haastateltavista yksi oli asiakkaille sopivia ratkaisuja etsivä johtaja (H3). Hänellä oli nykyisen tehtävänsä sekä arkkitehtitaustansa vuoksi hyvin kokonaisvaltainen näkemys mallintamisen vaikutuksesta kehitystyöhön sekä projektien onnistumiseen. Jatkossa haastateltavista käytetään tunnisteena haastattelunumeron (H1—H13) lisäksi mallintamisen kannalta olennaisia ja ensisijaisia työtehtäviä ja projektirooleja. Kuten seuraavista tulosluvuista selviää, haastateltavat tarjosivat monipuolisia näkemyksiä ja kokemuksia mallintamisesta sekä mallien käytöstä tietojärjestelmäkehityksen eri aktiviteeteissa ja projektin hallinnassa.

5.2 Syyt mallintaa ja olla mallintamatta

Haastatteluista kävi ilmi, että mallintaminen ei ole automaattinen osa nykypäivän tietojärjestelmäkehitystä pitkistä perinteistä huolimatta. Mallintamis- ja dokumentointimenettelyt olivat vaihtelevia haastateltavien organisaatioissa. Osalla organisaatioista ei ollut määriteltyjä toimintatapoja, vaan tavat vaihtelivat hankkeesta ja projektista toiseen, asiakkaan kanssa sovitun mukaisesti. Etenkin ohjelmistoarkkitehdit H2 ja H5 painottivat toimintatapojen puutetta. Osalla organisaatioista tai niiden liiketoimintayksiköistä oli dokumentaatiosuositusten lisäksi toimintatapoja mallintamiselle, määriteltyjä prosesseja, joiden mukaisesti kuvauksia tuotettiin. Näitä pyrittiin noudattamaan, ellei projektissa tai asiakkaan kanssa sovittu muuta. Haastateltavat H3, H4, H6, H7 ja H9 kertoivat heidän organisaatiossaan olevan tällaisia ohjeistuksia tai suosituksia. Suurimmaksi osaksi mallintamispäätökset olivat kuitenkin yksilön käsissä. Kehitystyön tai tuotetoimitusprojektin aikana mallinnettiin, jos se katsottiin tarpeelliseksi ja siihen oli vaadittavat resurssit.

Haastateltavat toivat esille näkemyksiään ja kokemuksiaan mallintamisen hyödyistä, mutta toisaalta myös tilanteista, joissa mallintaminen tai mallien käyttö ei heidän mielestään ollut tarpeellista tai perusteltua. Esiin tulleet syyt mallintaa ja olla mallintamatta ovat suurelta osin päällekkäisiä ja riippuvaisia toisistaan. Syyt ja hyödyt esitetään yhdistelemättä niitä liikaa, jotta haastateltavien ääni tulisi selkeämmin esiin ja jotta lukija voisi paremmin arvioida tutkijan päättelyprosessia. Tuloksista hahmotettiin yläkategoriat, joihin esiin tulleet syyt ja hyödyt voitiin luokitella. Nämä kategoriat, sisäkkäiset näkökulmat, ovat: *viestintä-*, *projekti/prosessi-*, *suunnittelutyö-* ja *tuotenäkökulma*. Kuvion 12 mukaisesti eri näkökulmiin luokitellut syyt ja hyödyt vaikuttavat toisiinsa, alkaen viestin-

nästä ja päättyen tuotteeseen eli kehitystyön lopputulokseen. Viestintä mahdollistaa mallintamisen hyödyt muissa näkökulmissa.



KUVIO 12 Syyt mallintaa ja olla mallintamatta: sisäkkäiset näkökulmat

Tietojärjestelmäkehitys on yhteistyötä, jossa ketterien tiimien toiminta perustuu paljolti kommunikoinnille. Mallintamisessa käytetään *kieltä*, joten mallien käyttö pohjautuu nimenomaan tämän viestinnän edesauttamiseen. Mallien kautta tapahtuva viestinnän helpottuminen auttaa tietojärjestelmäkehitykseen liittyvien prosessien ja niitä sisältävien projektien läpiviemistä. Tietojärjestelmäkehityksen prosessit (tai aktiviteetit tai vaiheet) voivat vaihdella projektista ja määritelmistä riippuen. Prosesseja ovat esimerkiksi analyysi, vaatimusmäärittely, suunnittelu, koodaustyö, testaus ja ylläpito, ja niitä tukevat dokumentointiin sekä laadun- ja projektin hallintaan liittyvät toiminnot. (ks. esim. Pressman & Maxim, 2015, s. 17—18; Sommerville, 2016, s. 44.)

Prosesseista on tuloksissa omaksi osiokseen nostettu suunnittelutyö, johon luokitellut syyt ja hyödyt edustavat etenkin arkkitehtien näkökulmaa. Sillä ei tarkoiteta kirjallisuuskatsauksessa esitettyä suunnitteluvaihetta, vaan yleisesti kaikkea suunnittelutyötä, johon mallintamisella voidaan vaikuttaa, kuten esimerkiksi tietomallin suunnittelua tai arkkitehtuurisuunnittelua. Sisin taso on tuotenäkökulma, joka sisältää projektin ja prosessien sekä suunnittelutyön tuloksena syntyvään ratkaisuun (tietojärjestelmä tai sen osa tai ominaisuus, räätälöity tai tuotteistettu ratkaisu) liittyvät syyt mallintaa ja siihen vaikuttavat hyödyt. Seuraavaksi tuloksia mallintamisen syistä ja hyödyistä tarkastellaan näkökulma kerrallaan, aloittaen viestinnästä ja päättyen tuotenäkökulmaan.

5.2.1 Viestintä

Jokaisessa kolmessatoista haastattelussa pääsyyksi mallintaa ja hyödyntää malleja nousi viestinnän edesauttaminen eri sidosryhmien kesken. Osa haastateltavista sanoitti mallin merkityksen työssään erittäin suoraan:

> …sehän on niinku se peruskommunikaatiotapa millä sää kommunikoit sitä, mitä ollaan tekemässä tai mitä muitten pitäis sun mielestä tehdä. (ohjelmistoarkkitehti H2)

> …ei halua samaa asiaa selittää moneen kertaan, jolloinka sitten mallinnus viestintävälineenä on ihan hyvä. Hyvä työkalu. (pääarkkitehti H10)

Mallit olivat monen haastateltavan mielestä tärkeä osa viestintää etenkin asiakkaiden ja kehitystiimin kanssa toimittaessa.

> Helpompi selittää kuvasta, kun tota näyttää koodia, ei oikein voi näyttää koodia asiakkaille, kun ei ne välttämättä ymmärrä siitä mitään. (ohjelmistoarkkitehti/-kehittäjä H1)

> …kun tehään tiimityötä, niin viestinnänhän täytyy pelata jatkuvasti ja sun täytyy pystyä asiakkaan kanssa asia kommunikoimaan. (ohjelmistokehittäjä H8)

Tulos ei ole yllättävä, koska läpi kirjallisuuden mallin on sanottu edistävän kommunikointia ja myös aiemmissa empiirisissä tutkimuksissa viestinnälliset hyödyt on todettu tärkeimmäksi syyksi käyttää malleja. Haastateltavat mainitsivat kuitenkin myös tilanteita, joissa tämä kaikkien mielestä kommunikointia edistävä tekijä voitiin tiimityössä sivuuttaa. Viisi haastateltavaa oli sitä mieltä, että mallien käytöstä ei ollut hyötyä silloin, kun tiimissä kaikki eivät niitä ymmärtäneet. Tätä aihetta käsitellään luvussa 5.3 Mallintamiseen liittyvät haasteet. Mallintamista ei myöskään koettu tarpeelliseksi silloin, kun tiimi oli pieni ja tuttu, eikä osaaminen ollut siiloutunutta.

> …aika paljon tehään sekasin noita hommia ni tuota, ei me nyt noita kaavioita piirretä. -- niin pitkään yhessä tehty töitä tossa ni, kyllä se aika äkkiä tulee selville, jos joku jotain ei ymmärrä, ja koko ajan keskustelemalla, kun kuitenkin siinä samassa toimistossa ollaan ja jutellaan keskenään ja kokeillaan ja katotaan, että okei, näin se toimii. Ei siihen niinku niitä mallinnuksia kyllä tarvii. (ohjelmistoarkkitehti/-kehittäjä H1)

> …meillä ei ole tämmöistä fordilaista liukuhihnaa, vaan että kaikki osaa kaikkea. Jolloin, mm, tiimin jäsenet keskustelee asioista toistensa kanssa, mutta heidän ei tartte välittää tämmöistä niinku tietopakettia, että ”nyt mulla on tämmöinen ongelma ja tässä on nyt malli, luepas se, jotta tiedät mitä mä seuraavaks kysyn.” (projekti- ja palvelupäällikkö H4)

Tiiviissä ja pitkäikäisessä yhteistyössä haastateltavat luottivat siihen, että asiat selviävät ja haluttu lopputulos saavutetaan keskustelemalla. Sovellusarkkitehti H12 oli työskennellyt vuosien ajan samassa projektissa ja tiimissä, jonka vähäistä mallien käyttöä hän perusteli näin:

> Samat ihmiset on pysyny tuossa vuosia ja ne tietää kaiken.

Konsulttina toimivalla ratkaisuarkkitehdillä H9 oli kokemusta monen eri yrityksen tietojärjestelmäkehityksen kulusta. Hän näki myös varjopuolen mallintamisen sivuuttamiselle hektisessä projektityössä:

> …käytännössähän siinä tulee, tulee semmosta, aika paljon tarvetta sitten keskustella asioista, mutta yleensä myös se keskustelu jää tekemättä…

Kaikki haastateltavat olivat sitä mieltä, että vaikka sisäinen viestintä onnistuisikin ilman malleja, muiden sidosryhmien kanssa toimittaessa ne olivat välttämättömiä. Syytkin tähän olivat hyvin samansuuntaisia läpi haastattelujen: mallit olivat haastateltavien mielestä tekstimuotoiseen tai suulliseen kuvaukseen verrattuina yksikäsitteisempiä, ytimekkäämpiä ja havainnollistavampia. Haastattelujen edetessä selvisi, että hekin, jotka kertoivat selviävänsä tiiminsä kesken viestinnästä ilman malleja, käyttivät kuitenkin silloin tällöin paperille tai valkotaululle piirrettyjä vapaamuotoisia malleja. Vapaamuotoisten mallien merkitystä ja käyttöä käsitellään luvussa 5.5 Vaihtoehtoiset tavat. Taulukossa 3 esitetään viestintänäkökulman syyt mallintaa ja olla mallintamatta. Esiintyvyys kertoo, kuinka monessa haastattelussa syy/hyöty esiintyi.

TAULUKKO 3 Syyt ja hyödyt: viestintä

Syyt käytölle / hyödyt	**Esiintyvyys**	**Syyt olla käyttämättä**	**Esiintyvyys**
Edesauttaa viestintää eri sidosryhmien kesken	13	Tiimissä kaikki eivät ymmärrä / kaikkia ei hyödytä	5
Edesauttaa asiakas-kommunikaatiota	13	Tiimi on pieni ja/tai tuttu	3
		Tiimissä kaikki osaavat/ tekevät kaikkea	2
Edesauttaa viestintää tiimissä	10		

5.2.2 Projekti/prosessi

Suurin osa haastateltavista painotti mallien olevan hyödyllisimpiä kehitystyön alkuvaiheessa, jossa järjestelmän tai sen osan vaatimukset määritellään ja kommunikoidaan tiimille. Ratkaisuarkkitehti H9 kiteytti syyn mallintaa näin:

> …todella hyödyllisiä ne on siinä alussa, jos yritetään kommunikoida kehittäjätiimille, että mistä on kyse, niin sillon, sillon ne olis hyvä olla ainakin korkealla tasolla jonkunlaiset raamit sille, sille tehtävälle palvelulle tai ohjelmistokomponentille. (ratkaisuarkkitehti H9)

Epämääräisellä ”alkuvaiheella” haastateltavat tarkoittivat useimmiten analyysiin lukeutuvaa määrittelyä ja alustavaa suunnittelua eli *mitä tehdään* ja tiedon jakamista virallisen dokumentaation laatimisen sijaan. Tavat, joilla tiimiä ohjeistettiin, vaihtelivat suuresti, riippuen projektin tavoitteista ja tekijän mallintamispieteetistä. Osa niistä haastateltavista, jotka kertoivat mallien olevan tärkeitä

suunnitteluvaiheessa, tarkoittivat myös sillä analyysiin lukeutuvaa korkean tason suunnittelua. Perustuen haastateltavien kuvailemien tekniikoiden käyttöön (esim. komponenttikaavio), alkuvaihe saattoi sisältää myös mallintamista suunnittelussa eli *miten tehdään*. Koska ketterässä kehityksessä toteutus usein nivoutuu suunnitteluun ja määrittelijä tai suunnittelija voi olla myös toteuttaja, myös koodaustyön ohjeistaminen mallien avulla voidaan laskea osaksi tätä alkuvaihetta.

Alkuvaiheen lisäksi haastatteluissa nousi esiin mallien merkitys viestinnässä myös kehitystyön myöhemmissä vaiheissa, dokumentaation muodossa. Suurin osa haastateltavista oli sitä mieltä, että dokumentoidut mallit edesauttoivat järjestelmään perehtymistä. Konsulttitöitä tekevät arkkitehdit painottivat visuaalisen mallin tärkeyttä, kun oli tutustuttava uuteen järjestelmään. Yhdeksän haastateltavaa, työtehtävästä riippumatta, oli sitä mieltä, että mallien avulla tietojärjestelmiä oli helpompi ylläpitää.

> …yleensä se on erillinen ylläpitotiimi, joka lähtee ylläpitämään näitä järjestelmiä. -- Nopeiten sitä yleensä pääsee sisälle, jos näkee jonkunlaisen diagrammin siitä rakenteesta ja sitten toiminnoista. (ratkaisuarkkitehti H9)

> …jokaiselle se on aina ollut järjestelmän kokonaisuus helpompi selittää ylläpitoon, kun sulla on se kaavio, josta voit sormella vaan osotella… -- Niin hirvittävän hyvä, jos niitä on, liian vähän niitä yleensä käytetään… (pääarkkitehti H10)

> …mitä lähempänä ollaan tällaisia yleisiä mallinnuskäytäntöjä, määrityskäytäntöjä, niin sen todennäköisempää, että se ylläpito on helpompaa. (projekti- ja palvelupäällikkö H4)

> Ilman malleja ei pärjää. -- Ja jos sitä yleisen tason kuvausta ei oo ja hyppää tuntematttomaan koodiläjään, niin sieltä on tosi vaikee saada kiinni sitä vanhaa prosessia. Sen takia ne mallit on ehottoman tärkeitä myös jatkossa ja että niitä ylläpidettäs. (ohjelmistoarkkitehti H13)

Ohjelmistoarkkitehdin H13 kommentti oli hyvin osuva. Vaikka hän ja osa muista haastateltavista painotti takaisinmallinnettujen kaavioiden merkitystä dokumentaatiossa niiden paikkansapitävyyden vuoksi, myös piirrettyjä korkean tason kuvauksia kaivattiin. Tämä todennäköisesti johtuu siitä, että takaisinmallinnus koskee yleensä luokkatason yksityiskohtaisia kuvauksia, jotka ovat jo hyvin lähellä koodia, eivätkä ne siksi yksinään ole riittäviä. Korkeammalla abstraktiotasolla olevat kaaviot ovat helpompia sisäistää nopeasti ja siksi tärkeitä, kun järjestelmään perehdytään. Tästä huolimatta osa suosi perehtymisessä usein koodin lukemista mallin sijaan.

> Ite oon tottunu siihen, että mää katon suoraan koodista, että miten se menee, mitä oon tehny sen parikymmentä vuotta, se on itelle se luontevin tapa. (ohjelmistoarkkitehti/-kehittäjä H1)

Ohjelmistoarkkitehti/-kehittäjä H1 lisäsi, että perehtymisen apuna olisi hyvä olla joku henkilö, joka tuntee järjestelmän ja osaa kertoa sen toiminnasta. For-

wardin ym. (2010) ja Badreddinin ym. (2018) tutkimuksissa suullinen tieto olikin yleisin tapa perehtyä järjestelmään.

Tuotepäällikön H11 mielestä tiedon välittäminen dokumentoitujen mallien kautta oli sen verran yksiselitteisempää tekstiin verrattuna, että jopa loppukäyttäjät hyötyivät malleista tutustuessaan järjestelmään. Tämä näkökohta ei noussut esiin muissa haastatteluissa eikä se ollut yleinen aiemmissakaan tutkimuksissa (ks. esim. Störrle, 2017). Ei ollut kuitenkaan yllättävää, että sen toi esille tuotepäällikkönä toimiva haastateltava, koska tuotepäällikön rooliin liittyy olennaisesti asiakastarpeiden syvä ymmärrys (Ebert, 2014).

Projekti- ja palvelupäällikkö H4 ja sovellusarkkitehti H12 kertoivat, että dokumentoidut mallit tai niiden puute vaikuttivat tehtäväkiertoon ja uusien työntekijöiden perehdyttämiseen:

> …siks me tehdään aina se tekninen dokumentaatio lopussa, jotta sitten seuraava sukupolvi, joka voi olla joku toinen kuin tekijä tai vaikka se tekijä ois edelleen tiimissä, niin hän ei oo sitten ikuisesti naitettuna siihen tehtävään… (projekti- ja palvelupäällikkö H4)

> …jos sä niin kun opetat jotain uutta ihmistä mukaan siihen, niin sit ne kaavioitten kautta on tosi helppo kertoa, et mikä tää homma on. -- …[koska emme tee juurikaan dokumentaatiota] ne kaverit, ketkä on siinä mukana projektissa, ne ylläpitää. Että eihän siihen voi sit uutta ihmistä noin vaan ottaa, kun se on se perehdytystyö aika iso ollu, että. (sovellusarkkitehti H12)

Testausprosessin osalta mallien hyödyntäminen jäi epäselväksi. Ainoastaan pari haastateltavaa mainitsi, että malleja todennäköisesti hyödynnettiin testauksessa edes jotenkin, perehdyttäessä järjestelmään korkealla abstraktiotasolla. Ohjelmistoarkkitehti H5 kertoi, että koska malleja ei heidän kehitystyössään juurikaan käytetty, ei niitä hyödynnetty myöskään testauksessa. Ratkaisuarkkitehti H9 selitti, että mallien käyttö ei ollut testauksessa yleistä, koska testaajat eivät niitä ymmärtäneet. Hän oli kuitenkin ohjelmistoarkkitehdin H5 tavoin sitä mieltä, että myös tämä ammattiryhmä voisi malleista hyötyä, mutta se vaatisi kouluttautumista.

Esimies- ja projektipäällikkötaustaisilla ja pitkän uran tehneillä, kuten haastateltavilla H2, H4 ja H9, oli kokonaisvaltaisin näkemys mallintamisen merkityksestä ja hyödyistä. He kertoivat mallintamisella olevan tärkeä rooli projektien kokonaisonnistumisessa ja tietojärjestelmän koko elinkaaren aikana, ei ainoastaan alkuvaiheen viestinnässä. Osa painotti mallin merkitystä toimivassa työnohjauksessa ja -johtamisessa, jossa hyvä suunnittelu malleineen takasi, että projekteissa työvauhti pysyy yllä, eivätkä työmäärät ylity. Ratkaisujohtaja H3 mainitsi, että ainoastaan tekstipohjaisten määrittelyjen perusteella projektin johdon oli erittäin hankalaa selvittää, olivatko kaikki määritykset päätyneet tehtäviksi työnohjausjärjestelmään ja myös toteutettu.

Kokeneet ja mallintamista korostavat haastateltavat esittivät, että mallien käytöllä voidaan tukea projektien kustannuksissa pysymistä. He kertoivat mallintamisen hyötyinä olevan väärinkäsityksiltä välttyminen ja siten yhteisymmärryksen saavuttaminen, virheiden huomaaminen aikaisessa vaiheessa kehi-

tystyötä sekä näiden ansiosta työmääräarviossa pysyminen. Muissakin haastatteluissa näitä mallien tarjoamia etuja nostettiin esille, mutta niitä ei esitetty laajemmassa mittakaavassa kustannusten tai projektin läpiviemisen näkökulmasta.

> …edelleen kuitenkin se, se halvin kohta muuttaa asioita on silloin, kun niitä suunnitellaan. -- …niissä hankkeissa, missä ehkä jollain tavalla on laiminlyöty sitä suunnitteluvaihetta, niin niissä kyl useammin ajaudutaan niin kun siihen, että työmäärät ylittyy, se on mun niin kun kokemus ja sitä kautta periaatteessa kaikille pitäs olla perusteltavissa, et se on hyödyllistä ja se pitäs tehä se suunnittelu hyvin. Ja se johtaa parempiin tuloksiin sekä niin kun laadun näkökulmasta että niitten kustannusten näkökulmasta. (ratkaisujohtaja H3)

> …se asia on, voi olla ymmärrettävissä vähän väärin tai että koska teksti on kumminkin välillä vähän monisyistä ja tota siis silleen, että se toinen puhuu appelsiineista ja toinen omenoista. -- Toki ne väärinkäsitykset varmasti ennemmin tai myöhemmin sitten niin kun sieltä jäisi kiinni anyway, mutta tää vähentää sitä turhaa työtä ihan huomattavasti. (tuotepäällikkö H6)

Mallintamisen vaikutuksista kustannuksiin ja tehtävän työn määrään oli myös toisenlaisia näkemyksiä. Arkkitehtien H1, H9 ja H12 haastatteluissa tuli esille, että toisinaan mallintaminen koettiin turhana työnä.

> Meillä menee niin päin yleensä, että tuota, koska me mielellään tehään se ensin, katotaan miten siitä tulee järkevä ja tehokas ja toimii hyvin, niin tota, ei käytetä turhaan siihen mallintamiseen aikaa. (ohjelmistoarkkitehti/-kehittäjä H1)

> Joskus on niin kun miettiny, että vois tehä [mallintamista], mut sitten kun se vie sitä aikaa, kun vähän miettii, et onks tässä nyt niinku järkee, se maksaa kuiteski sille asiakkaalle. Sitä kun miettii, niin eihän se saa tästä mitään ja sit me pärjätään ilmankin, ni ei oo sit tehty. (sovellusarkkitehti H12)

Kolmessa haastattelussa kävi ilmi, että yhteistyötä ja päätöksentekoa voitiin jouduttaa vakuuttamalla vastapuoli mallien avulla. Haastateltavat kertoivat, että mallit olivat usein laadun ja ammattitaidon tae.

> …siellä toisessa päässä voidaan niinku vakuuttua siitä toimittajan niin kun kyvystä miettiä, määritellä, tuottaa niinku laadukasta dokumentaatiota, se voi olla myös enne, että se itse tehty ratkasukin on niinku toivottavasti laadukasta, koska se on pystytty kuvaamaan. (tuotepäällikkö H6)

> …joka kerta kun mä saan kaavion joltakulta, niin mä oon sitä mieltä, että okei, tää tyyppi tietää mitä se tekee. (tuotepäällikkö/sovellusarkkitehti H7)

Moni haastateltavista kertoi, että malleja käytettiin etenkin monimutkaisissa ja keskeisissä tapauksissa. Tämän lisäksi mallintaminen koettiin tärkeäksi isoissa hankkeissa ja projekteissa, joihin osallistui paljon tekijöitä, tiimejä ja yrityksiä. Osa tarkensi, että uuden, räätälöidyn ratkaisun tuottamisessa mallien käyttö oli oleellisempaa kuin tuotepohjaisessa toimittamisessa.

…asiakkaan täsmätarpeisiin tuleva järjestelmä, jossa sitten se mallintaminen on paljon tärkeemmässä roolissa, koska siellä ei oo semmosta valmista pohjaa. (ratkaisujohtaja H3)

…[kaavioiden] tarve on sitten semmosissa isommissa, isommissa tota kehityshankkeissa ennemminkin kuin, meillä saattaa olla semmosia pieniä muutos, ihan pieniä muutaman työpäivän kokoisia, niin ehkä niihin ei sitten, ei oo tarvetta. -- …niissä on sanallista kuvausta ja käyttäjätarina. (tuotepäällikkö H11)

Taulukossa 4 esitetään haastatteluissa esiintyneet projekti/prosessinäkökulmaan liittyvät syyt käyttää malleja tai olla käyttämättä niitä.

TAULUKKO 4 Syyt ja hyödyt: projekti/prosessi

Syyt käytölle/hyödyt	**Esiintyvyys**	**Syyt olla käyttämättä**	**Esiintyvyys**
Alkuvaiheessa, tiedettävä mitä tehdään	11	Pienissä projekteissa/ominaisuuksissa ei tarvetta	6
- edesauttaa analyysiä/määrittelyä	8		
- edesauttaa suunnittelua	8	Tekstimuotoiset käyttötapaukset/määrittelyt/ prosessikuvaukset usein riittäviä	4
- ohjaa koodaustyötä	7		
Dokumentaatiossa	11	Tekeminen ketterää, vältetään kaikkea ylimääräistä	3
- edesauttaa ylläpitoa	9		
- edesauttaa perehtymistä	8	Työkalulla saa tarvittaessa generoitua kaaviot	3
- edesauttaa perehdyttämistä	2	Ei ole roolin puolesta tarvetta enemmälle	2
- ohjeistuksena testauksessa	2	Perehtyy koodia lukemalla	2
Isoissa projekteissa/hankkeissa/ominaisuuksissa	8	Toimitetaan valmistuote / projekti selkeä käyttöönotto	2
Yhteisymmärryksen saavuttamisessa / vältytään väärinkäsityksiltä	8		
Monimutkaisissa/keskeisissä tapauksissa	7	Oma kapean sektorin tuote tuttu, ei tarvitse malleja perehtymiseen	1
Uutta/räätälöityä tehtäessä	5	Perehtymisessä hyödynnetään suullista tietoa	1
Edesauttaa päätöksentekoa	4	Projektissa tietojärjestelmää ei ole suunniteltu alusta asti	1
Tarpeellinen läpi projektin/elinkaaren	4		
Varmistutaan, että kaikki tarvittava tulee tehtyä	4		
Vähentää turhaa työtä	4		
Nähdään pullonkaulat/puutteet / huomataan virheet ajoissa	3		
Pysytään kustannusarviossa / halvin kohta muuttaa	3		
Työvauhti pysyy yllä, nopeuttaa kehitystä	3		
Vakuuttaa ammattitaidosta	3		
Projektin kokonaisonnistumisessa	2		
Voidaan validoida nykytilanne suunniteltuun	1		

5.2.3 Suunnittelutyö

Etenkin arkkitehdit esittivät mallintamisen edesauttavan ymmärrystä, kokonaisuuksien hahmottamista ja ongelmanratkaisua. Nämä suunnittelutyötä helpottavat tekijät olivatkin ymmärrettävästi viestinnän lisäksi arkkitehdeille pragmaattisimmat syyt mallintaa.

Kyllä se mallintaminen vaan ja se, ni on kuitenkin osa sitä järjestelmän analyysiä ja ymmärtämistä, että mitä sää oot tekemässä. (ohjelmistoarkkitehti H2)

…kokonaisuuksien hahmottamisessa paras työkalu on yleensä vähän piirtää asioita auki. (pääarkkitehti H10)

> …ne niin kun ratkee siinä piirtäessä, sä huomaat, että no hei, näähän pitää jotenkin liittyä toisiinsa. Et ei se niinku sanallisesti onnistu, onnistu, et kuvan kautta se jäsentyy parhaiten. (sovellusarkkitehti H12)

Mallintamisen vähäisyyttä tai jopa sivuuttamista perusteltiin taitojen kehittymisellä kokemuksen karttuessa. Tällöin mallintamista ei nähty välttämättömänä apukeinona suunnittelussa. Myös ohjelmointikielten abstraktiotason nousu ja valmiiden komponenttien käyttö vähensivät haastateltavien mielestä mallintamistarpeita.

> …nyt ne on jotenkin niin paljon helpompia nää asiat. Tai sit onko omat taidot kehittyny tässä, mieltää mielessäkin jo paremmin... -- …se oli sitä siinä alussa, että sä piirrät kaikki, kaikki mitä vastaan tulee… (sovellusarkkitehti H12)

> …huippuammattilaisilla asiat siis pysyy yleensä päässä, ne niin kun visualisoi ne päässään. Ja jos, jos niitten ei tarvii kommunikoida kellekään, niin tota eipä niitten kannata niitä kauheasti auki piirrelläkään. Eräs toi entinen luennoitsija tota noin niin yliopistolla sanoi, että niin eihän näitä menetelmiä niin kun tehdä niin kun, et jos kaikki olis huippuammattilaisia, niin eihän tämmösiä menetelmiä tarvittas ja kyl se niin kun siinä on tavallaan oikeassa. (pääarkkitehti H10)

Abstrahoinnin kautta mallit edistivät itsenäistä suunnittelutyötä, mutta niiden viestinnällinen voima edesauttoi toki myös kollaboratiivista suunnittelua. Piirrettyjen tai takaisinmallinnettujen kaavioiden avulla haastateltavat pystyivät keskustelemaan tiimin kanssa ja yhdessä rakentamaan mahdollisimman hyvää ratkaisua kulloiseenkin ongelmaan.

Kolme haastateltavaa mainitsi uudelleenkäytön hyödyt malleihin liittyen. Jos malli oli aiemmassa käytössä toimivaksi todettu, sitä saatettiin käyttää pohjana uutta ratkaisua suunnitellessa. Systemaattisesti toteutettuna uudelleenkäyttö olisi merkittävä projekti/prosessi- tai tuotenäkökulman etu. Tämän tutkimuksen haastatteluissa esiintynyt uudelleenkäyttö tulkittiin kuitenkin lähinnä tekijän omaa suunnittelutyötä helpottavaksi, tarpeen mukaan, ei niinkään järjestelmälliseksi osaksi prosessia tai tuotenäkökulmaa.

Mahdollisuus tarkastella järjestelmää useasta eri näkökulmasta nousi esille yhtenä syynä mallintaa ainoastaan arkkitehtien H2, H9 ja H13 haastatteluissa. Muut haastateltavat pitivät tätä mahdollisesti itsestäänselvyytenä, mutta asiaan on myös toisenlainen selitys. Mallintamistaitojen ja motivaation puute, asiakasvaatimukset sekä kiire voivat johtaa siihen, että kaikki tarvittavat näkökulmat pyritään sisällyttämään yhteen ja ainoaan kaavioon.

> …ihmiset piirtää niin kun yhteen, yhteen malliin tai diagrammiin sen koko maailman, mikä on täysin väärä, väärä tapa tehdä sitä asiaa. Mutta ihmisten ajatusten ja toimintatapojen muuttaminen on niin hidasta, että on parempi vaan jotenkin sietää sitä. (ratkaisuarkkitehti H9)

Mallintamiseen liittyviin haasteisiin palataan seuraavassa luvussa 5.3 ja vapaamuotoisiin malleihin luvussa 5.5 Vaihtoehtoiset tavat. Suunnittelunäkökulmaan liittyvät syyt mallintaa ja niistä johtuvat hyödyt on esitelty taulukossa 5.

TAULUKKO 5 Syyt ja hyödyt: suunnittelutyö

Syyt käytölle / hyödyt	**Esiintyvyys**	**Syyt olla käyttämättä**	**Esiintyvyys**
Edesauttaa ymmärrystä	5	Hahmottaa asiat päässään	3
Edesauttaa kokonaisuuksien hahmottamista	4	On kokemusta, taidot kehittyneet	2
		Ei tehdä enää alusta kaikkea, yhdistellään valmiita komponentteja	2
Edesauttaa ongelmanratkaisua	4		
Edesauttaa kollaboratiivista suunnittelua	3	Ohjelmointikielten abstraktiotaso noussut	1
Tarkastelu useasta eri näkökulmasta	3		
Uudelleenkäytön hyödyt	3		

5.2.4 Tuote

Ohjelmistojen laatua pohdittiin useassa haastattelussa. Osa haastateltavista oli sitä mieltä, että laatu saavutettiin hyvillä tekijöillä, laadunvarmistuksella tai etenkin testauksella. Osa taas oli vahvasti sitä mieltä, että mallintamisella oli roolinsa laadukkaan lopputuloksen rakentamisessa.

> Ehkä se [laadukkaan ohjelmiston tekeminen] on sit kuitenkin siinä, että se on mallinnettu oikein, käytännössä. (ratkaisujohtaja H3)

> …se tekninen kokonaisuus pysyy eheänä ja se on suunniteltua, et se ei oo semmosta rönsyilevää. Eli siinä mielessä niin kun siinä hyötyy organisaatio, siinä hyötyy se tekninen ratkasu, laatu pysyy luultavasti vähän parempana, kun se homma tehään niin kun se on mietitty. (tuotepäällikkö H6)

Huomioitavaa on, että tuotteen laadussa yksi määrittävä tekijä on ylläpidettävyys (ks. esim. Nugroho & Chaudron, 2008). Kun haastateltavat kertoivat mallin olevan tärkeä ylläpidolle, he korostivat ylläpitoprosessin tai -projektin tehostamista, eivät niinkään ohjelmiston laatua. Mallintamisen hyödyt ylläpidossa on siten luokiteltu projekti/prosessinäkökulmaan tuotenäkökulman sijaan. Mutta kuten kuvion 12 sisäkkäiset näkökulmat esittävät, ylläpidon helpottuminen vaikuttaa myös lopputuloksen laatuun. Laatu ja ylläpidettävyys liittyvät myös pitkään elinkaareen, jonka mainitsi mallintamisen hyötynä kaksi haastateltavaa.

Tietoturvaa pidettäneen yleisesti merkittävänä ei-toiminnallisena vaatimuksena, mutta sen varmistamista ei erikseen mainittu kuin kahdessa haastattelussa syynä mallintaa. Voidaan toki jo lähtökohtaisesti olettaa, että turvallisuusominaisuudet ovat erottamaton osa laadukasta tietojärjestelmää ja sen arkkitehtuuria. Aihepiiriä myös sivuttiin haastatteluissa, joissa kerrottiin monimutkaisten rajapintojen ja eri komponenttien välisien kommunikaatioiden mallintamisesta. Näistä kerrotaan tarkemmin seuraavissa luvuissa 5.3 Käytössä olevat tunnetut kaaviotekniikat ja työkalut ja 5.4 Vaihtoehtoiset tavat.

Tietojärjestelmän laadun, tietoturvan ja elinkaaren pituuden lisäksi suorituskyvyn varmistaminen mainittiin parin arkkitehdin haastattelussa syynä käyttää malleja.

> …tietokantojen pitää olla tosi niin kun nopeeta, hyvin, hyvin suorituskykyisiä ja sitten myös niin kun integraatioitten totta kai pitää olla suorituskykyisiä, että näitä sitten niin kun ratkaistaan sekä niin kun sekä erilaisilla kaavioilla, et niin kun tai sillai niin kun periaatteessa lähetään justiin siitä, että piirretään se idea sillai kaaviolla, esitellään se muille, muille niin kun kehitystiimin jäsenille ja sitten niin kun, sitten lähetään tota siitä iteroimaan hiljalleen. (tuotepäällikkö/sovellusarkkitehti H7)

Tuotenäkökulmaan liittyvät syyt olla käyttämättä malleja liittyvät kehitettävän ratkaisun yksinkertaisuuteen, korkeaan abstraktiotasoon ja kehitystyökalujen tarjoamaan tukeen kehitystyössä. Taulukossa 6 esitetään tuotenäkökulmaan liitetyt syyt mallintaa ja niistä seuraavat hyödyt sekä niiden esiintyvyys haastatteluissa.

TAULUKKO 6 Syyt ja hyödyt: tuote

Syyt käytölle / hyödyt	Esiintyvyys	Syyt olla käyttämättä	Esiintyvyys
Johtaa parempaan laatuun	4	Integraatioarkkitehtuuri yksinkertainen / yksinkertaiset rajapinnat	2
Elinkaaresta pitkäkestoisempi	2		
Suorituskyvyn varmistaminen	2	Sovellusalueelle / ohjelmistokehyksellä korkean abstraktiotason ratkaisut	2
Tietoturvan varmistaminen	2		
		Kehitystyökalujen avulla tuotteen rakenteen muuttaminen	1

5.2.5 Mallien vähäisestä käytöstä johtuvat ongelmat

Tämän luvun 5.2 aiemmista alaluvuista sekä tulevista tulosluvuista voi erottaa ongelmia, joita haastateltavien mukaan seuraa, kun kehitystyön aikana ei hyödynnetä malleja riittävästi eikä tuoteta tarpeeksi laadukasta dokumentaatiota kaavioineen. Lähes kaikki haastateltavat harmittelivat alalla liian yleistä dokumentaation vähäisyyttä. Haastatteluista kuitenkin ilmeni, että juuri mallien puutetta voi toisinaan olla vaikea yhdistää ongelmiin.

> Sanotaanko, että semmosessa projektissa missä ei oo, ei oo dokumentaatiota, niin on, on tullut ongelmia. -- Se, että ratkaseeko, ratkaseeko se kaavion puuttuminen, ni siitä en oikeestaan osaa sanoa. -- Sanotaanko, että ne, ne tosiaan niinku auttaa ongelmien ratkasussa, mutta niin, niin. Ja varmasti, varmasti sillai niin kun on ollu tilanteita missä se semmonen niin kun jonkun tietyn kaavion puuttuminen on voinut olla sillei niinku keskeinen osa siitä, että minkä takia joku ongelma, ongelma on niin kun ylipäänsä ilmennyt. (tuotepäällikkö/sovellusarkkitehti H7)

Syy-seuraussuhteen havaitsemisen vaikeudesta huolimatta haastateltavat kertoivat useista ongelmista, joita heidän mielestään mallien vähäisestä käytöstä seuraa. Yleisimmiksi ongelmiksi nousivat ylläpidon ja järjestelmään perehtymisen vaikeutuminen, väärinkäsitysten syntyminen sekä tiedon puuttuminen. Eniten ongelmia esittivät konsultti- tai päällikköasemassa olevat ja pitkän uran

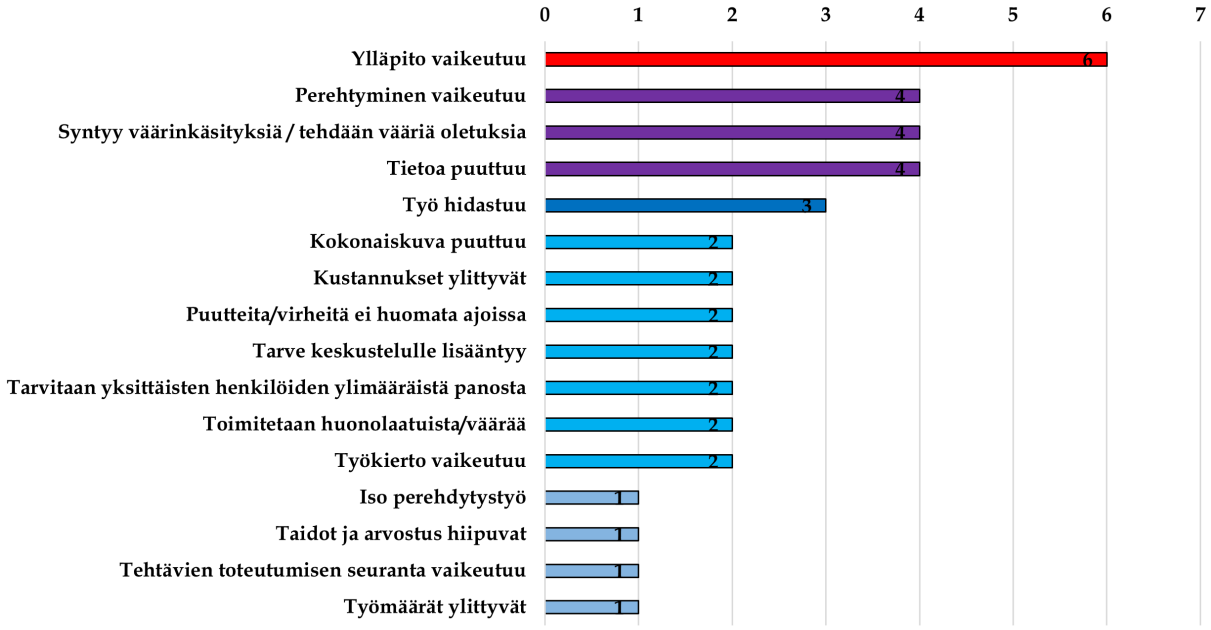
tehneet haastateltavat, luultavasti koska heillä oli laaja-alaisin kokemus aihepiiristä. Eräs haastateltava oli sitä mieltä, että nykypäivän tietojärjestelmäkehityksessä mallintamista ei arvostettu eikä hyödynnetty tarpeeksi. Hän kiteytti siitä seuraavat ongelmat näin:

> No, vaikka tuo nauhotus nyt on päällä, niin sutta ja paskaahan sitä tulee ihan paljon, että siitähän sä sen näät, että. Aika harvoin saat lukee sellaisen uutisen, että sen tuolta Tivistä sun muista, että tietojärjestelmän rakentaminen osui nappiin ja säästettiin sata miljoonaa.

Toinen haastateltava selitti hieman tarkemmin mallien puutteesta ilmeneviä ongelmia:

> …saatetaan huomata vasta jossakin integraatiovaiheessa tai jopa tuotantoon mennessä, että jotkut palvelut ei toimikaan yhteen. Että ollaan tehty oletuksia, jotka ei sitten pädekään. -- Tai sitten jotakin muita tämmösiä epäyhteensopivuuksia, mitä ei oo otettu huomioon, eikä myöskään oo saatettu huomata jotain tota performanssiongelmia, että siinäkin on tehty jotain oletuksia, mutta se ois voitu jossakin mallissa ehkä nähdä, että siinä tietyssä kohdassa on pullonkaula ja siihen pitäs kiinnittää huomiota. -- …tai sitten jopa unohdetaan joku integraatio kokonaan, että huomataan vasta sitten tuotannossa, että täältähän puuttuu tämä integraatio kokonaan.

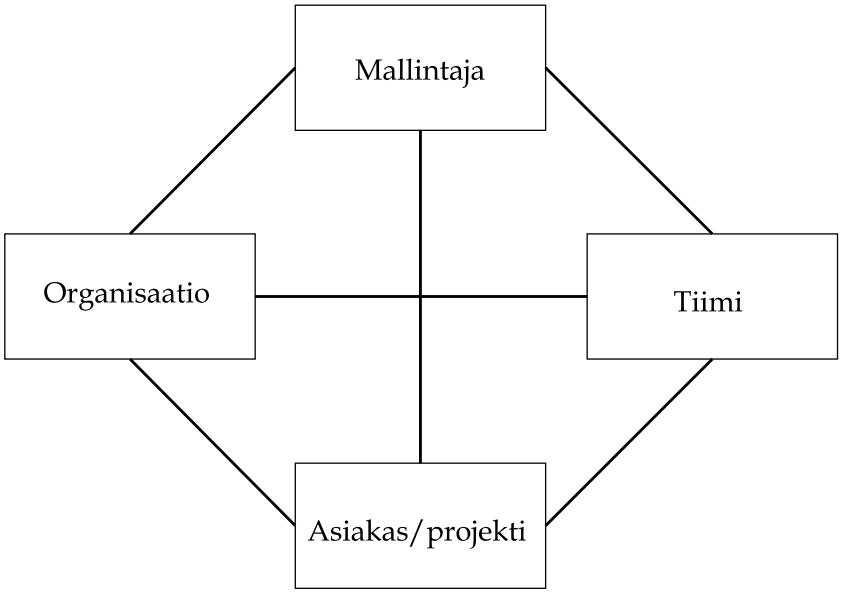
Kuviossa 13 esitetään yhteenveto ongelmista, joita haastateltavat kertoivat seuraavan mallien sivuuttamisesta tai vähäisyydestä. Luvut ilmaisevat, kuinka monessa haastattelussa kukin ongelma mainittiin.



KUVIO 13 Ongelmat, kun ei käytetä malleja

5.3 Mallintamiseen liittyvät haasteet

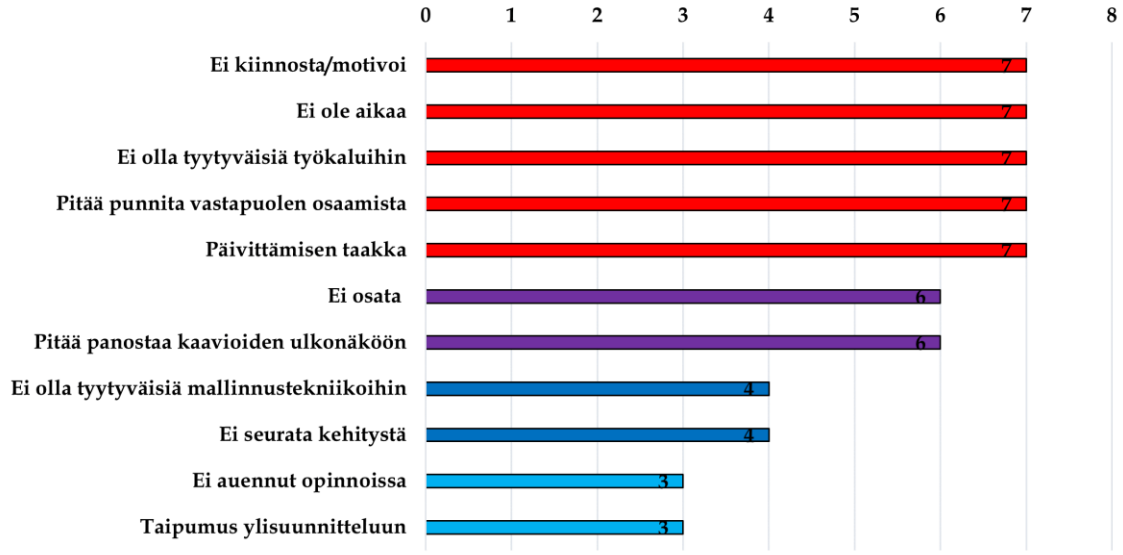
Mallintamisen merkitystä korostetaan alan kirjallisuudessa, mutta sen käytön yleisyydestä ja hyödyllisyydestä ollaan montaa mieltä. Forwardia ym. (2010) lainaten: jos mallintaminen on niin hienoa, miksi nämä ihmiset eivät tee sitä enemmän? Haastatteluissa nousikin esiin suuri määrä haasteita, jotka rajoittavat mallintamista usealla eri tasolla. Haasteet on luokiteltu neljään eri kategoriaan (kuvio 14): *mallintajaan ja tämän itsenäiseen työhön liittyviin haasteisiin, tiimin haasteisiin, organisatorisiin haasteisiin ja asiakkaasta/projektista kumpuaviin haasteisiin*. Haasteet ja siten myös nämä kategoriat ovat toisiinsa limittyneitä ja osin päällekkäisiä. Seuraavaksi haasteita tarkastellaan kategoria kerrallaan, aloittaen mallintajasta ja päättyen asiakkaaseen/projektiin.



KUVIO 14 Mallintamiseen liittyvien haasteiden kategoriat

5.3.1 Mallintaja

Moni haastateltava myönsi, että mallintamista olisi hyvä tehdä enemmän, mutta esitetyt haasteet ilmeisesti osoittautuivat usein tarpeita suuremmiksi. Merkittävä osa haasteista oli ymmärrettävästi suoraan liitettävissä mallintajaan ja tämän suorittamaan työhön. Kuviossa 15 esitetään mallintajaan liittyvät haasteet ja niiden esiintyvyys haastatteluissa.



KUVIO 15 Haasteet: mallintaja

Merkittäväksi teemaksi haastatteluissa nousi itse mallintamistyön luonne, sen mielekkyys ja kyky motivoida tekijäänsä. Seitsemässä haastattelussa ilmeni, että mallintamista ei pidetä kovinkaan kiinnostavana osana työtä. Osa haastateltavista sanoi suoraan, että mallintaminen ja mallien käyttö ei motivoinut heitä tai kollegoja.

> Mä en hirveesti tykkää noista kuvista muutenkaan, että mieluummin tehään ja katotaan sitte kuvia sen jälkeen… -- …kollegojen kanssa mitä keskustelee, ni ne haluis vielä vähemmän piirtää ku minä, ku keskittys vaan tekemään oikeesti. Jonkun niitäkin on piirreltävä aina välillä.
>
> Jos haluaa joku tehä, niin antaa tehä vaan. Että ei kukaan ei estä. Mutta ei oo kyllä kukaan halunnu tehä.
>
> Kukaan ei haluu piirtää kuvia.
>
> …ei se mieluisin tapa niinku tehdä, täyttää työpäiviä dokumentointiin, että on se sit teksti tai kuva tai mikä tahansa, että.
>
> …jos ei sitä oo pakko tehä, niin eihän sitä tehdä… -- On se vähän semmonen pakonomainen paha aina se dokumentointi.

Kiinnostuksen puute tuli esille haastatteluissa myös siten, että mallintamiseen liittyvää kehitystä ei seurattu. Alalla erilaisten tekniikoiden ja työkalujen kehityksen seuraamista voidaan pitää lähes itsestäänselvyytenä, mutta tämä ei jostain syystä ulottunut mallintamiseen. Kun haastateltavilta tiedusteltiin, että seurasivatko he UML:n kehitystä, oli vastaus lähes yksinomaan kieltävä. Haastateltavat kyllä esittivät toivottuja ominaisuuksia mallinnustyökaluille, mutta eivät kuitenkaan aktiivisesti etsineet uusia ja parempia työkaluja. Aineistosta ei

myöskään selvinnyt, miten hyvin haastateltavat olivat tietoisia jo käytössä olevien työkalujen tarjoamista mahdollisuuksista.

Mallintamisen kiinnostavuuden puute ei rajoittunut tiettyyn työtehtävään; haaste esiintyi arkkitehtien, tuotepäälliköiden ja ohjelmistokehittäjien haastatteluissa. Etenkin arkkitehtien osuutta voidaan pitää yllättävänä, koska mallintaminen on perinteisesti käsitetty olennaiseksi osaksi heidän työtään. Osa haastateltavista ei kuitenkaan esittänyt kiinnostuksen puutetta henkilökohtaisena haasteena, vaan mielestään alaa yleisesti vaivaavana ongelmana.

> …kun ihmiset helpommin ja helpommin pääsee tälle alalle ohjelmoimaan, näitä tietokoneita ja tietoa on enemmän saatavilla, niin tulee enemmän sitten alalle ihmisiä, joilla ei oo, ei oo koulutusta tai haluakaan oppia tämmösiä tylsempiä asioita niin kun mallinnusta ja dokumentaatiota ja kokonaisuuden hallintaa. (ratkaisuarkkitehti H9)

> …[ajatellaan], että kyllä mä nyt pärjään ilman näitäkin, mä saan tän jutun tehtyä. Ja se, et jääks tästä jotain esimerkiks dokumentaatiota jälkipolville, niin sehän ei oo kenenkään mielessä niin kun kovin korkeelle, että ne on sitten tulee, mitä sattuu tulemaan. Yleensä ei tuu mitään. (pääarkkitehti H10)

Kun mallintaminen yhdistettiin vahvasti dokumentointiin ”pakollisena pahana”, ei ollut yllättävää, että suurimmassa osassa haastatteluita mallien nopeasta vanhenemisesta johtuva päivittämisen taakka nousi esille haasteena. Haaste oli odotetusti etenkin arkkitehtien mainitsema, koska heidän harteilleen yleensä jäi sekä mallintamispäätökset että tuotosten ajantasaisina pitäminen. Haaste sekä rajoitti haastateltavien halua mallintaa että hyödyntää muiden tekemiä malleja.

> Kukaan ei jaksa ylläpitää niitä semmosia dokumentaatioita. Vähän näitten kaavioittenkin kanssa. Et sit ne vanhenee. Kun se yhenkin kerran käy niin, että sä huomaat, että tää ei oo luotettava tää kaavio, niin sit sä et enää välitä siitä ollenkaan. Et se on, se on yks, yks semmonen pointti, miks ei ehkä tehä näitä niin paljon. (sovellusarkkitehti H12)

Jatkuvaa ajantasaistamisen taakkaa minimoidakseen moni haastateltava kertoi suosivansa mallintamista korkealla abstraktiotasolla. Ainoastaan ratkaisujohtaja H3, ohjelmistoarkkitehti H5 ja tuotepäällikkö H6 mainitsivat, että kaavioita tehtiin ”tarkallakin tasolla”. Aineistosta ei kuitenkaan selviä, mikä tämä tarkka taso on tai koskiko se ainoastaan dokumentointia jälkikäteen vai myös kehitystyötä ohjaavaa suunnittelua. Moni muu haastateltava puhui enintään komponenttitason mallintamista tai alijärjestelmätason mallintamista. Komponenttitason mallintamisella haastateltavat eivät kuitenkaan tarkoittaneet kirjallisuuskatsauksessa esitettyä komponenttien sisäisten rakenteiden ja toimintojen mallintamista, vaan arkkitehtuuriin liittyvää komponenttien ja niiden rajapintojen kuvaamista ylemmällä tasolla.

> …jos menee liian detail-tasolle ja alkaa miettimään, että tämä luokka ja tämä toinen luokka ja tämä olio ja tämä olio juttelee keskenään, niin sitten se on jo melkeen hirveen lähellä sitä, että mitä ohjelmakoodissa tapahtuu ja se riski on se, että niin kun se, sitä se on taas se ylläpidon tai se ajantasaistamisen taakka. (ohjelmistoarkkitehti H13)

> …jos jotain piirrellään, niin piirretään sit semmonen ylemmän tason kuvia ja oikeestaan sellasia laatikkokuvia… -- …ei sit ihan semmosta luokkatason, objektitason kuvaa tuu käytännössä juurikaan piirreltyä. -- Ei niitä oikeen liian yksityiskohtasia voi tehä, koska tota, pitäis olla semmonen yleispätevä, että sitten se koodi voi muuttua sitten, niin kaavioita viitti koko aikaa olla päivittämässä. (ohjelmistoarkkitehti/-kehittäjä H1)

Kaikissa tiimeissä mallien ajantasaistamista ei kuitenkaan ollut jätetty täysin mallintajien harteille. Ohjelmistokehittäjä H8 kertoi, miten heidän tiimissään asia oli ratkaistu:

> Kyllä me ollaan päivitetty vuorotellen, että ollaan laitettu ihan taskeiksi itsellemme, että se, joka ensimmäisenä ehtii. Vähän niin kun kaikki muutkin työt.

Merkittäväksi haasteeksi haastatteluissa nousi mallintajan tasapainoilu mallien erilaisten esitystapojen välillä, puoliformaalin ja vapaamuotoisen mallintamisen välillä. Seitsemän haastateltavaa kertoi, että mallintaessa piti ottaa huomioon vastapuolen eli asiakkaan, kehitystiimin tai kumppaniyrityksen oletettu tekninen osaamistaso ja tarvittaessa tehdä samasta kaaviosta eri versioita eri vastaanottajille. Käytännössä tämä tasapainoilu johti usein vapaamuotoiseen mallintamiseen. Tätä haastetta käsitellään tarkemmin luvussa 5.5 Vaihtoehtoiset tavat.

Haastatteluista ilmeni, että moni piti mallintamista taitona, joka ei ole alalla itsestäänselvyys tai helposti saavutettavissa. Osaamisen puute nousi esiin haasteena kuudessa haastattelussa. Haaste esitettiin mallintamista rajoittavana tekijänä yleisesti ottaen alalla, ei niinkään omaan osaamiseen liittyvänä.

> …en ees tiiä voiko sitä opetella oikeestaan mitenkään, että se on varmaan vähän, tulee sitten, jotkut osaa piirtää ja toiset ei. (tuotepäällikkö H11)

> …kaikille se pensseli ei vaan sovi ihan yhtä hyvin käteen. (ohjelmistoarkkitehti H13)

Ohjelmistoarkkitehti H13 vertasi mallintamista hyvin osuvasti maalaamiseen. Samanlainen näkemys tuli vastaan myös muissa haastatteluissa, ja verrataan mallintamista alan kirjallisuudessakin usein taiteen tekemiseen tieteen sijasta (ks. esim. Mendling ym., 2010; Moody, 2005). Ratkaisuarkkitehti H9 kertoikin, että mallintamisesta saattoi kasautua suorituspaineita, joiden vuoksi se mahdollisesti sivuutettiin kokonaan.

> …mitä oon nähnyt, niin aika monella se syy on varmasti se, että ei osata piirtää niitä, niin se nähdään niin kun pelottavana asiana sitten. Ja ei haluta niin kun nolata itseään sitten, että piirretään huonoja malleja tai jotain vastaavaa. Että se ohjelman kirjoittaminen on tutumpaa ja turvallisempaa, niin halutaan pysyä siellä sitten.

Ohjelmistoarkkitehti H2 koki haasteen erityisen merkittäväksi. Hänen mukaansa nykypäivän tietojärjestelmäkehitystä vaivasi osaajien puute. Hän esitti, että alalla tekijät eivät ole enää kovin valikoituneita.

> …jos mä meen tonne ja kysyn keskiverto webbikoodarilta, että miten sä mallinnat, ni eihän, se kattoo mua kuin hoomoilasta, että ”mistä sä puhut, onks se näitä React-komponentteja?”

Nykypäivän tietojärjestelmäkehityksessä liiankin yleinen mallintamistaitojen ja kiinnostuksen puute näkyi ohjelmistoarkkitehdin H2 mukaan myös vastavalmistuneiden tasossa. Hänen mukaansa monikaan ei osannut edes tulkita UML-kaavioita. Tuotepäällikkö H6 taas oli havainnut eron ammattikorkeakoulu- ja yliopistotaustaisten välillä ja oli sitä mieltä, että jälkimmäisillä oli ”pikkasen parempi kyky mallintaa”. Kaksi haastateltavaa esitti, että vastavalmistuneet yleisesti ottaen osasivat mallintaa ja myös kokivat sen tärkeänä, mutta alalla vallitsevat haasteet johtivat yleensä mallintamisen sivuuttamiseen.

> …koulussa näistä vielä puhutaan, yleensä, että olis ees joitakin asioita ihan järkevä mallintaa. Mutta tota noin niin, musta tuntuu, että sitten kun ne on aikansa alalla olleet … -- …eikä kukaan muukaan tee, niin sit se vähän niin kun jää. (pääarkkitehti H10)

> …ihmiset kun ne ei mallinna, niin sitten sitä ei nähdäkään tarpeellisena ja sit se alkaa unohtua se, se taito ja kyky lukea niitä malleja myöskin. Ja myöskin se arvostus, että mä melkein nään, että niin kun mitä nuorempi on niin kun alalla, niin sitä paremmin se ehkä osaa ja ymmärtää malleja. (ratkaisuarkkitehti H9)

Mallintamisopintojen ongelmallisuutta ja tiedon soveltamisen vaikeutta käytäntöön tukee kuitenkin kolmen haastateltavan omat kokemukset opinnoistaan. Arkkitehtien H7 ja H12 sekä ohjelmistokehittäjän H8 mielestä mallintamisen tarkoitusta ei ollut helppoa ymmärtää opiskeluolosuhteissa ja mallien merkitys olikin avautunut vasta käytännön työn myötä. Ohjelmistokehittäjä H8 lisäsi, että erilaisten UML-kaavioiden käyttötarkoitus jäi hyvin epäselväksi opinnoissa.

Ohjelmistokehittäjä H8 oli sitä mieltä, että tarkastelun alla oleva asia pitäisi pystyä esittämään yhden kuvan avulla. Ratkaisuarkkitehdin H9 mielestä tällainen ajattelu johti usein siihen, että mallintamisen hyviä periaatteita ei noudatettu, vaan yhteen kaavioon mahdutettiin kaikki halutut näkökulmat. Hän kuitenkin lisäsi, että muiden puutteellisten taitojen tuloksena syntyneitä malleja oli siedettävä, jotta kommunikaatio onnistui. Projekti- ja palvelupäällikkö H4 taas koki, että huonosti tai huolimattomasti tehtynä mallintaminen ei edistä tiedon saamista tai yhteisymmärrystä. Ohjelmistoarkkitehti H13 oli samaa mieltä:

> …pahimmillaan se on tota viis laatikkoo, pari viivaa ja tota siit ei kukaan tiiä mitään ja sit se on jpg-muodossa jossain verkkolevyllä ja se on viis vuotta vanha ja siihen ei oo kukaan koskenu ja kaks järjestelmää niistä on jo poistunu. Et se, se ei sitten niinku palvele oikeen mitään…

Kiinnostuksen ja osaamisen puute huolestutti osaa haastateltavista. He uskoivat näiden tekijöiden olevan merkittävä haaste mallintamisen tulevaisuudelle.

> …se tulee kokonaisuudessaan luultavasti vähenemään jopa se mallinnus ja mallien ymmärtäminen sitten, ellei jotain, jotain tapahdu. (ratkaisuarkkitehti H9)

> …vähän huonommalle [näyttää perinteisen mallintamisen tulevaisuus]. -- …mallintamisen tulevaisuus on varmaan sitä, että mallinnetaan jatkossakin ad hoc, ehkä. (pääarkkitehti H10)

> En mää usko, et kenelläkään on niinku enää niinku pitkäjänteisyyttä [mallintamiseen]. (ohjelmistoarkkitehti H2)

Pari haastateltavaa oli kuitenkin sitä mieltä, että mallintamista tarvitaan jatkossa aiempaa enemmän:

> Jos jotakin, niin [mallintaminen] on vielä tärkeempää tulevaisuudessa. -- …koodarin ammatti on kuoleva ammatti, et tota et niin kun keinoäly tulee tekemään sen. (tuotepäällikkö/sovellusarkkitehti H7)

> Entistä enemmän on suunniteltavaa ja integroitavaa ja kommunikoitavaa. Koodaaminen menee koko aika helpommaks, koodin kirjottaminen ja muuta, tota. Entistä enemmän tarvitaan nuoli ja laatikko -ihmisiä. (ohjelmistokehittäjä H8)

Yleinen haaste mallintamiseen liittyen oli ajanpuute. Seitsemän arkkitehtiä kertoi, että kiire sanelee hyvin pitkälle sen, miten perusteellista mallintamista on järkevää suorittaa.

> Jos on projekti, ni ei niihin tota, arkkitehtuurikuviin, niin se on aika minimaalinen aika mitä käytetään. (ohjelmistoarkkitehti/-kehittäjä H1)

> …[mallintaminen] ei oo niin kauheen merkityksellinen nykypäivän työnteossa ja nykypäivän niin kun organisaatioissa, niin sitten se mallintamiseen käytetty aikakin on aika rajoittunutta. (ohjelmistoarkkitehti H2)

Viisi arkkitehtiä kertoi, kuinka kaavioiden ulkonäköön panostaminen vei runsaasti aikaa. Visuaalisen miellyttävyyden tavoittelu koettiin tärkeäksi, mutta myös haastavaksi osaksi työtä.

> …mä en haluu välittää siitä visuaalisuudesta, mutta mun on pakko välittää siitä visuaalisuudesta, koska jos mä en välitä siitä, ni se ei, se ei niinku mee se viesti, se viesti menee vielä huonommin läpi kuin ilman sitä. (ohjelmistoarkkitehti H2)

> …siinä menee tosi paljon aikaa, kun sää saat niitä näin siistin näköiseksi. Koska siinä on tärkeetä, et se näyttää vakuuttavalta se kaavio, eikä semmoselta viivasotkulta. (sovellusarkkitehti H12)

> …se omalla nimellä kulkee siellä asiakaskommunikaatiossa ja muussa, niin se täytyy olla ehottoman niinku siis niinku tavallaan tyylikäs, että viivat on suorat. Se, se vaan niinku tuntuu ainakin itelle tärkeelle. (ohjelmistoarkkitehti H13)

Useassa haastattelussa ilmeni, että mallinnustyökaluihin ei oltu tyytyväisiä. Työkaluihin liittyviin haasteisiin pureudutaan tarkemmin luvussa 5.4.3 Työkalujen käyttö. Lisäksi neljä haastateltavaa esitti, että käytettyihin mallinnustekniikoihin ei oltu täysin tyytyväisiä. Tätä asiaa käsitellään tarkemmin luvussa 5.4.2 Kaaviotekniikoiden käyttö. Ajoittaisen taipumuksen ylisuunnitteluun (ks. myös ”analysis paralysis”, esim. Glass, 2002) toteuttamisen sijaan mainitsi kolme haastateltavaa.

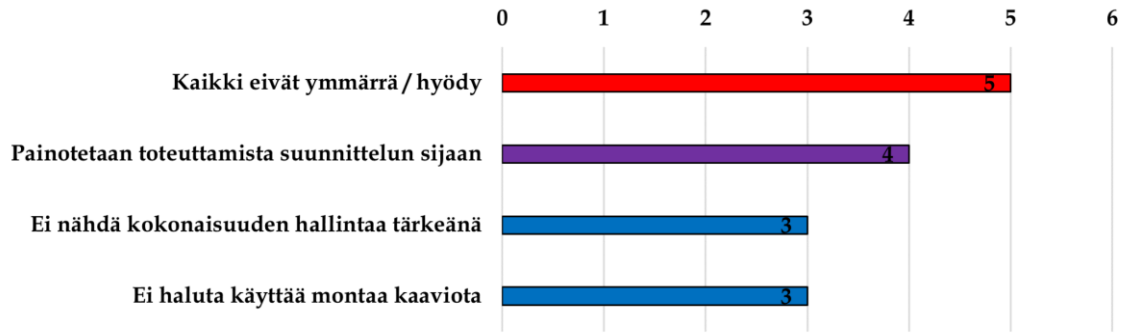
5.3.2 Tiimi

Tiimityöskentelyyn liittyvistä haasteista (kuvio 16) nousi haastatteluissa useimmiten esiin se, että kaikki eivät ymmärtäneet tai osanneet lukea malleja. Haaste liittyy olennaisesti jo aiemmin esitettyyn haasteeseen mallintamistaitojen puutteesta.

> …jos mä heitän UML-kaavion, niin välttämättä kaikki ei osaa sitä tulkita. (ohjelmistoarkkitehti H2)

> …jengi ei oo ollu ehkä kouluttautunu tällasiin asioihin, ne ei oo tuntenut niin kun tavallaan UML:ää, niin ihan turha niille on mallintaa. (pääarkkitehti H10)

> …[UML] on kommunikointikieli kehittäjien välillä, siinä on niinku tietynlainen, että miten sitä luetaan ja miten sitä piirretään ja sitten jos vaan yks osaa sen, niin ei se sitten siellä tiimissä välttämättä palvele niin kun tarkotustaan. (ohjelmistoarkkitehti H13)



KUVIO 16 Haasteet: tiimi

Ratkaisuarkkitehti H9 kertoi haasteen olevan nähtävissä esimerkiksi testaajien vähäisessä mallien hyödyntämisessä. UML oli kuitenkin hänen mielestään vaihtoehtoihin verrattuna hyvin ymmärretty mallinnuskieli alalla. Tämän vuoksi hän piti UML:n käyttöä hyödyllisenä, kuten myöskin sen mahdollista jatkoke-

hittämistä. Ohjelmistokehittäjän H8 mielestä tiimityöskentelyssä oli otettava huomioon, että kaikki eivät hyödy malleista oli käytetty kieli mikä tahansa:

> Osa hahmottaa paremmin vaikka listaa, sanoja ja osa ei välttämättä halua mitään konkreettista, ne haluaa, että niille on sanottu jotain ja sitten ne alkaa hahmottaan asiaa päässään, tehdessään ja osa haluaa tietää kokonaiskuvan, ennen kuin rupee tekemään ja osaa ei kiinnosta se kokonaiskuva lainkaan.

Tiimityöskentelyyn liittyvinä haasteina esitettiin, että suunnittelua tai kokonaisuuden hallintaa ei pidetä tärkeänä, vaan painopiste on toteutuksessa. Haastattelujen perusteella tiimeissä ajatellaan, että ilmankin etukäteissuunnittelua lopulta päädytään hyvään lopputulokseen.

> …tällä alalla yleisesti edelleen mun mutu on semmonen, et kyllähän siellä on paljon sitä, että, et enempi niinku lähetään vaan tekemään -tyyppisesti ja ajatellaan, että se niin kun, homma siitä, siitä kyllä menee kuosiin… (ratkaisujohtaja H3)

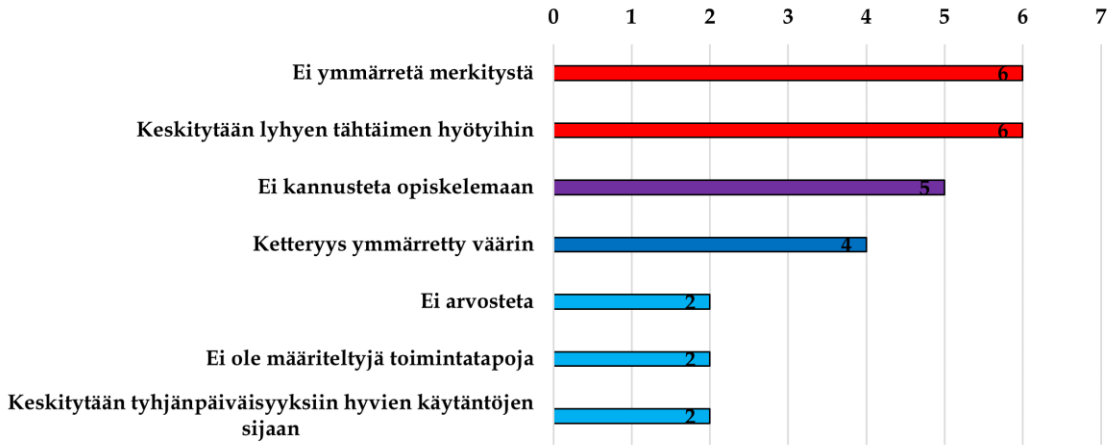
> …ei nähdä sitä kokonaisuuden hallintaa tärkeänä, vaan uskotaan, että kunhan vaan tehdään asioita, niin se jotenkin loksahtaa paikoilleen. (ratkaisuarkkitehti H9)

> ..tää Scrum on se, että kun niitä ei mietitä laajemmin niitä asioita, niin se on ehkä, mistä mä en ite tykkää. Koska mä näin sen ajan, kun tehtiin vesiputousmallilla, et se mua vähän harmittanu, että niin kun mennään vähän laput silmillä. Ainakin nuoremmat kehittäjät, niin mä aina sanon, että kattokaa vähän kauempaa, että mitä te ootte tekemässä. -- ne ei niinku ymmärrä, että mitä sillä niinku saavutetaan sillä jutulla laajemmin, mitä se asiakas saa. (sovellusarkkitehti H12)

Kolmessa haastattelussa esitettiin, että kehitystiimissä ei useinkaan haluta tutustua moneen eri kaavioon, vaan monesti suositaan yhden kaavion käyttöä. Tämä johtaa usein varsin vapaamuotoisten kuvaustapojen käyttöön, jota käsitellään enemmän luvussa 5.5 Vaihtoehtoiset tavat.

5.3.3 Organisaatio

Haastatteluista ilmeni, että organisaatiossa ei ymmärretä mallintamisen tärkeyttä eikä sen vuoksi kannusteta mallintamiseen tai siihen kouluttautumiseen. Organisatorisiksi luokitelluilla haasteilla (kuvio 17) on merkittävä vaikutus aiemmin esitettyihin mallintamistyöhön ja tiimityöskentelyyn liittyviin haasteisiin. Kun organisaatio ei tue mallintamista eikä anna siihen tarvittavaa aikaa, työntekijät eivät voi kohdistaa sen hyödyntämiseen ja kehityksen seurantaan omia resurssejaan. Myöskään kiinnostus mallintamistyötä kohtaan ei pysy yllä tällaisissa olosuhteissa.



KUVIO 17 Haasteet: organisaatio

Ohjelmistoarkkitehti H2 näki tilanteen kaikista lohduttomimpana. Hänen mukaansa isoissa organisaatioissa mallintamista ei ymmärretä, ei arvosteta eikä myöskään haluta hyödyntää. Tämä oli hänen mielestään todettavissa esimerkiksi organisaation tarjoamien mallintamiskoulutusten vähäisyytenä.

> Ei mallintamista saa lähtee opiskelemaan. Paljon tärkeempää on mennä suorittamaan joku Kubernetes- tai Docker-kurssi kuin se mallintamiskurssi.

Myös ohjelmistoarkkitehti H13 oli sitä mieltä, että koulutuksissa keskityttiin uusiin teknologioihin mallintamisen sijaan. Kun muilta haastateltavilta kysyttiin organisaation järjestämistä mallintamiskoulutuksista, ei heillekään ollut sellaisia tarjottu tai niistä oli jo aikaa. Osa kuitenkin lisäsi, että koulutusta varmasti saisi, jos itse olisi sen suhteen aktiivinen. Ohjelmistokehittäjä H8 mietti, että organisaatiossa ei todennäköisesti ollut edes ajateltu, että muutkin kuin arkkitehdit voisivat hyötyä mallintamiskursseista.

Ohjelmistoarkkitehti H5 kertoi mallintamisen olevan organisaatiossaan vähäistä ja epäili syyksi sitä, että asiaa ei ollut osattu edes ajatella hyötyjen näkökulmasta. Ratkaisujohtaja H3 ja pääarkkitehti H10 totesivat ohjelmistoarkkitehdin H2 tavoin, että yrityksissä ei yleensä ajateltu pitkän tähtäimen hyötyjä, joita huolellisesta dokumentaatiosta malleineen seuraa.

> …organisaatio suhtautuu tommosiin aika niinku sillei leväperäisesti, et se on sitte ehkä enemmän sen yksilön vastuulla, että miten se niinku suhtautuu siihen tilanteeseen. -- Yleensä se dokumentaatio on valitettavan paljon niinku PowerPointtia ja sitten sitä, että kyllä se koodi sen dokumentoi sen riittävän hyvin, että. (ohjelmistoarkkitehti H2)

Alalla vallitsevalla yleisellä asenneilmapiirillä voi olla merkittävä vaikutus organisaatiossa ilmenevään mallintamisen arvostuksen puutteeseen. Negatiivisia asenteita mahdollisesti ruokkii myös jaottelu koodikeskeisiin ja mallintamista suosiviin ammattilaisiin (ks. esim. Badreddin ym., 2018). Jaottelu näkyi haastateltavienkin puheissa, kun mallintamista välillä kutsuttiin ”piirtelyksi”, vasta-

painona ”oikealle tekemiselle”. Haastateltavien esiin tuomat negatiiviset asenteet vaihtelivat oman ja muiden käyttäytymisen välillä.

> Edellisessä työpaikassa oli ihan osasto, missä ne teki ihan vaan kaavioo. -- Mut kyl myö vähän niitä mollattiin koko ajan, että ei tuo oo oikeeta työtä tommonen.

> …mun koulutus oli ehkä siihen aikaan, että tota, se oli vielä niin kun niitä viimesiä aikoja, kun vielä uskottiin tällasiin, niin kun ehkä tietynlaisiin mallinnusmenetelmiin. -- …sen jälkeen tuli kauhee niinku vastaisku, että mitään nyt ei saa mallintaa, että, että tota UML:ää kun joku mainitsi, niin aina tuli sata vitsiä siellä, hehehe UML:ää.. -- ...olemme edelleen hieman siinä tilanteessa, että se on sitte unohettu monet hyvät asiat, mitä on tehty.

Haastattelujen perusteella organisaatioissa ja projekteissa kaivattiin usein yksittäisten henkilöiden merkittävää panosta, jotta asiat etenivät. Ohjelmistoarkkitehti H5 oli sitä mieltä, että heidän organisaatiostaan puuttui mallintamisnäkökulmaa ajava osaaja, joka toisi mallintamiseen ja dokumentointiin kuuluvia käytänteitä muille tutuiksi. Ohjelmistokehittäjä H8 oli todennäköisesti omassa työyhteisössään tällainen henkilö, koska oli projektiin liittymisestään lähtien huolehtinut siitä, että kaavioita tuotettiin riittävästi. Ratkaisujohtaja H3 ja ratkaisuarkkitehti H9 esittivät, että mallintamisen vähäisyyttä ja toimintaohjeiden puutteita korvaamaan tarvittiin yleensä yksittäisten henkilöiden ylimääräistä työpanosta (ks. 5.2.2 Mallien vähäisestä käytöstä johtuvat ongelmat).

> …usein siinä tulee sit vastaan semmonen niinku mitä mää sanon hero effectiksi, että on niinku semmosia tiettyjä, tämmösiä henkilöityviä sankareita, jotka jossain hankkeessa, jotka on esimerkiksi ollut istumassa jossain workshopeissa ja niin edespäin, ja sit lähetään kovaa tekemään juttuja ja saahaan aluks tosi nopeesti paljon aikaseks. (ratkaisujohtaja H3)

> Se saatetaan sitten käytännössä kuroa sillei umpeen, että siellä on henkilöitä, jotka tekee sen toisella tavalla sen kokonaisuuden hallinnan sitten. Joko juoksemassa, juoksemalla eri tiimeissä ja keskustelemassa tai jotenkin muuten vastaavasti. (ratkaisuarkkitehti H9)

Neljä arkkitehtiä mainitsi haasteena organisaatioiden halun saada koko ajan nopeammin ja nopeammin toimivaa ohjelmistoa ulos (keskitytään lyhyen tähtäimen hyötyihin). Vaatimukset koettiin välillä hyvin kohtuuttomina, kun tavoitteena kuitenkin oli laadukkaan ohjelmiston toimittaminen.

> Siihen se nykyisin menee, et ei piirretä enää niin tarkkoja kuvia, pitäs vaan saadaan sitä koodia ulos mahollisimman paljon. Se on ihan yleinen joka puolella, kaikki yritykset yrittää mahollisimman kevyesti, mahollisimman tota tehokkaasti ja poistaa kaikki overheadit. Ihan pahimmillaan menny siihen, että testaustakaan ei juuri enää ole. Mikä on tosi huono juttu. Ne nähään noista miten huonoja nuo ohjelmistot nykyisin on. (ohjelmistoarkkitehti/-kehittäjä H1)

Kiirettä ja mallintamisen aliarvostamista ruokki haastateltavien mielestä ketterän kehityksen yleistyminen ja sen ymmärtäminen väärin.

> …on nähty, että kun tehdään ketterästi asioita, niin sillon ei tarvis mallintaa asioita, mikä ei ollu kyllä Scrumin ja agiilin, ketterän kehityksen mukaista alun perinkään, mutta sitä on käytetty sitten semmosena tekosyynä, minkä takia ei tarvii mallintaa ja dokumentoida asioita. (ratkaisuarkkitehti H9)

> Me hyvin harvoin kehitämme mitään räjäyttävän todella uutta. Useimmitenhan IT:ssä tehdään asioita, tehdään nyt vaikka ne ostolaskut, niitä on kuule tehty kymmeniä vuosia ennenkin. -- ketteryys vetäs näistäkin kehitysjutuista, joissa se tiedon jakamisen takia riittävä mallintaminen on aina ollut tarpeen, niin vähän mentiin ehkä siinä yli. (projekti- ja palvelupäällikkö H4)

> Mää uskon ainoostaan niinku arkkitehtuurilähtöiseen ohjelmistokehitykseen. Se on ainoo, minkä mää oon huomannu, että on 25 vuodessa tuottanu ees jollain lailla järkevää, koherenttia softaa. -- No, otetaan vaikka tää agile mikä on mun suosikkiinhokki nykyään on se, että mitään ei tartte dokumentoida ja mitään ei tartte niinku tehä muuta kuin vaan lätkitään koodia pinoon, niin. Mut eihän se niin mee. (ohjelmistoarkkitehti H2)

Ohjelmistoarkkitehti H2 viittasi kirjallisuudessakin esitettyyn ongelmaan arkkitehtuurin laiminlyönnistä ketterän kehityksen yhteydessä (ks. Abrahamsson ym., 2010; Babar, 2009). Ohjelmistoarkkitehdin H2 mielestä alaa vaivasi tällä hetkellä kaiken kaikkiaan huonot käytännöt. Myös ohjelmistoarkkitehti/-kehittäjä H1 koki organisaatioiden toimintaan vaikuttavat trendit ongelmallisina (keskitytään tyhjänpäiväisyyksiin hyvien käytäntöjen sijaan).

> Ohjelmistotekniikka on vähän tämmönen niinku ööh, vähän niinku ois kiva olla vähän niinku tiedettä, mut ku ei oikein olla tiedettä, ni keksitään sitte jotain tyhjänpäiväistä. (ohjelmistoarkkitehti H2)

> …aina ku otetaan joku uus malli käyttöön, niin ensin sitä käy joku konsultti hehkuttamassa ja sitten mennään ihan ettei tutkita kunnolla, että onko sitä järkevää käyttää. (ohjelmistoarkkitehti/-kehittäjä H1)

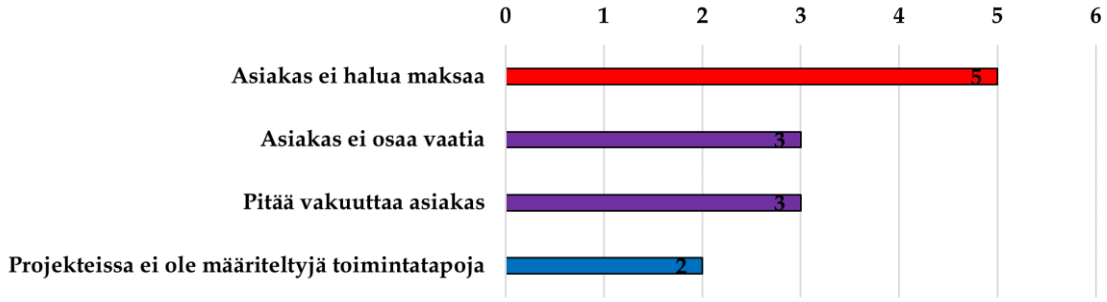
5.3.4 Asiakas/projekti

Haastatteluissa nousi esiin, että koska organisaatioissa ei tavoitella pitkän tähtäimen hyötyjä, asiakas ei osaa vaatia tietojärjestelmän laadukkuutta tukevia ominaisuuksia, kuten riittävää dokumentaatiota, eikä myöskään hyväksy mallintamista kustannuksissa. Asiakasrajapinnasta ja projekteista johtuvat haasteet (kuvio 18) heijastuvat organisatorisiin haasteisiin mallintamisen asettamisella taka-alalle, koska toimittajaorganisaatioilla ei välttämättä ole perusteita painottaa osa-alueita, joita kohtaan asiakkaalla ei ole kiinnostusta.

> …maturiteetti kasvaa koko ajan sen ympärillä, että mitä tää softan tekeminen oikeesti on ja, ja niinku mitkä asiat siellä on tärkeitä mitä pitäs painottaa, mistä maksetaan.

-- Edelleen niinku tosi paljon se, se ajatusmaailma pyörii siinä, että siitä koodaamisesta niinku maksetaan ja että se on se, joka tuottaa sitä tulosta. (ratkaisujohtaja H3)

…konsulttihommissa näkee sitä, että on aivan älyttömän niinku eritasoisia järjestelmiä dokumentaatioltaan ja laadultaan. Siihen on sit vaikuttanut monet tekijät, mutta jos asiakas vaatisi lähtökohtasesti dokumentaatiota, testausta, tietoturvaa, täntyyppisiä niinku alusta asti, niin tulis älyttömän paljon pitkäkestosempii siitä niinku sovelluksen elinkaaresta laadullisesti. (ohjelmistoarkkitehti H13)



KUVIO 18 Haasteet: asiakas/projekti

Projekti- ja palvelupäällikkö H4 esitti, että vaikka hänen omat asiakkaansa olivat valveutuneita, on toimittajaorganisaatioiden yleinen haaste saada asiakkaat ymmärtämään, että mallintaminen ja dokumentaatio ovat olennainen osa kokonaisuutta. Ohjelmistoarkkitehti H2 allekirjoitti väitteen:

…sun pitää pystyä perusteleen se, että jos mää käytän nytten, asiakas maksaa musta vaikka niinku 120 euroo tunti ja mä käytän kymmenen tuntia mallintamiseen ja se maksaa niille 1200 euroo plus jotain muita kuluja, ni kannattaako niitten maksaa siitä. -- …koko sen elinkaaren ajan sillä niinku laadukkaalla malleilla, laadukkaalla dokumentaatiolla on aina hyötyä. Mutta eihän ne nää sitä siinä vaiheessa, kun ne on lyömässä sen niinku, sanotaan nyt vaikka viis miljoona johonkin ohjelmistoprojektiin, ni eihän ne ny, ei ne niinku, vaikka se maksas sen kymmenen tonnia, elikkä se ei maksa käytännössä yhtään mitään, ni eihän ne oo sitä valmis niinku, ei ne niinku silleen niinku arvosta sitä kuin sitä ehkä pitäisi arvostaa.

Organisatorisissa haasteissa mainittu toimintatapojen puuttuminen koskee myös hankkeissa ja projekteissa ilmeneviä haasteita. Osa haastateltavista oli sitä mieltä, että ohjeistusten ja toimintatapojen määrittely lisäisi mallintamisaktiviteettia ja hyödyttäisi etenkin uraansa aloittelevia ohjelmistokehittäjiä.

…jos ne edellytettäs osaks, osaks tota, vaikka olis joku ehto, että ennen kuin projektissa päästään johonkin koodausvaiheeseen, niin pitää olla kuvattu joku asia. Tai sitten tai että asiakas vaikka vaatis, että osana, osana toimitettavaa järjestelmää pitää olla tietynlaiset asiat kuvattuina kaavioilla. (tuotepäällikkö H11)

Oisin sitä mieltä, että [dokumentaatiossa] aina sais olla joku peruskuvaus, vaikka lyhyt, kertoo mitä se tekee, muutama hyvä kuva siitä… Yleensä jos kirjotellaan jotain, niin laitetaan jonnekin Jiran sivulle, mitä mistä kukaan ei enää löydä niitä enää sitte parin vuoden päästä. (ohjelmistoarkkitehti/-kehittäjä H1)

> …ohjeistukset ja guidelinet, mitenkä esimerkiksi mallinnetaan, niin semmoset, semmosethan tukee [laadukkaan ohjelmiston tekemistä], että sitä ei jokainen itse koeta keksiä, että miten asioita pitäs dokumentoida tai, tai mallintaa. (ratkaisuarkkitehti H9)

> Varmaan vois olla hyötyä siitä, että olis enemmän niinku mallinnuskäytänteitä ja muita etenkin niin kun alkuvaiheessa uraa… (ohjelmistokehittäjä H8)

5.4 Käytössä olevat tunnetut kaaviotekniikat ja työkalut

Kuten kirjallisuuskatsauksen luvussa 3.1 kerrottiin, on UML kokoelma erilaisia alalla tunnettuja kaaviotekniikoita eli kaaviotyyppejä. Tässä luvussa esitellään haastateltavien mainitsemat käytössä olevat kaaviotekniikat ja työkalut. Ensin käsitellään UML:n käyttöä kokonaisuudessaan, jonka jälkeen siirrytään yksittäisiin tekniikoihin. Viimeiseksi käsitellään mallintamiseen käytettävien työkalujen käyttöä.

5.4.1 UML:n käyttö

Haastateltavat olivat yleisesti ottaen sitä mieltä, että UML tunnetaan alalla hyvin. Tietojärjestelmäkehittäjien välille muodostettu yhteinen kieli, jonka kaikki voivat ymmärtää samalla tavalla, nähtiin lähinnä positiivisessa valossa. Luvussa 5.3 tuli kuitenkin selväksi, että tätä ns. yhteistä kieltä osattiin hyvin vaihtelevasti. Haastateltavat kuvailivat UML:ää mm. seuraavasti: ”de facto”, ”hyvin määritelty”, ”hyvä työkalu”, ”looginen ja selkeä”, ”terminä kaikkein tutuin”, ”yksiselitteinen” ja ”yleisesti käytössä”. Myös täysin vastakkaisia mielipiteitä esitettiin: ”alalla suurin osa ei käytä”, ”ei oo koulun jälkeen oikein sillai näkyny”, ”liian vaikeekin monelta osin”, ”saa katseen lasittumaan”, joskus jopa saman haastattelun aikana.

Kaikki haastateltavat käyttivät työssään vähintään yhtä UML:n kaaviotyyppiä. Yksikään haastateltavista ei kuitenkaan kertonut hyödyntävänsä UML:ää kokonaisuudessaan tai rakentavansa järjestelmän ”täydellistä” mallia sen avulla. Ohjelmistoarkkitehti H2, joka hyödynsi UML:ää muihin haastateltaviin verrattuna laajamittaisesti, kuvaili käyttöään näin:

> Mä sanon sitä sekasotkumalliksi, elikkä mä en niinku millään tasolla pitäydy normaaliin UML-kaaviotyyppeihin.

Haastattelujen perusteella UML:n käyttö on vapaamuotoista, eikä sen yksityiskohtaisia sääntöjä noudateta tai niistä välttämättä olla edes tietoisia. Ohjelmistoarkkitehti H13 esitti, että UML:n laajamittainen ja tarkka käyttö todennäköisesti johtaisi lopputuloksen parempaan laatuun. Muiden haastateltavien tavoin hän ei kuitenkaan nähnyt tätä realistisena vaihtoehtona. Ketterässä kehityksessä vaatimusten jatkuvasti päivittyessä se olisi liian aikaa vievää ja kallista, eivätkä

kaikki tiimissä välttämättä edes osaisi UML:ää tarvittavalla tasolla. Osa piti UML:ää tarpeettoman monimutkaisena ja sen kokonaisvaltaisemman käytön edellyttämän usean kaavion tuottamista suhteettoman jäykkänä.

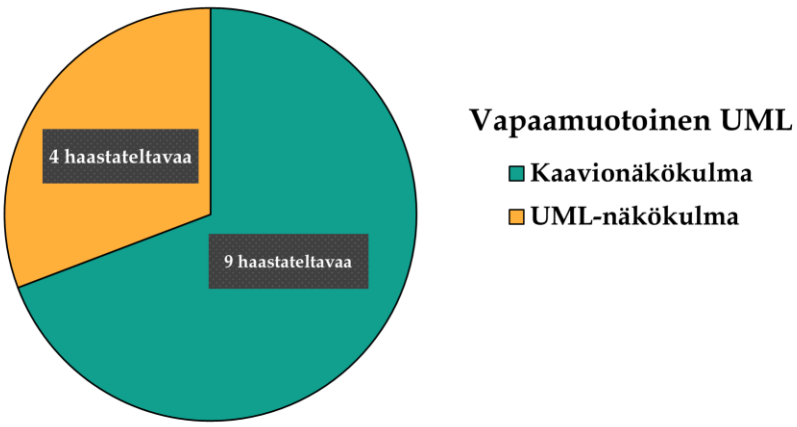
> ..se on vähän hukkatyötä. (ratkaisuarkkitehti H9)

> …varsinkin jostain UML kakkosesta, ni ei niissä oo mitään järkee, että ei kukaan jaksa nysvätä niitä, niitä kaavioita. Saati opetella, en mää ees viittiny opetella niin pitkälle. Mä oon opetellu joskus UML ykkösestä jotain perusteita ja niillä mennään, hyvin on riittäny. Se on enemmän ku useimmat tekee nykyään. (pääarkkitehti H10)

> …et sä saisit kaiken mitä sä haluat niinku kommunikoida, niin sä saattasit joutuu piirtämään viis kaaviota. Ei kukaan jaksa kattoo niitä viittä kaaviota. Haluu kattoo yhtä kuvaa. Se täytyy tulla selväks sillä. (ohjelmistokehittäjä H8)

Kaikkien haastateltavien osalta voidaan puhua UML:n vapaamuotoisesta käytöstä, joka vertautuu Fowlerin (2003) määritelmässä luonnosteluun ja Petren (2013) määritelmässä valikoivaan käyttöön. Haastateltavien UML:n käytön ja tuntemuksen välillä oli kuitenkin havaittavissa sen verran merkittäviä eroja, että kaikkien käytön niputtaminen yhteen kategoriaan ei ole mielekästä. Aineistosta nousikin esiin kaksi erilaista tapaa käyttää UML:ää vapaamuotoisesti: *kaavionäkökulmaan* ja *UML-näkökulmaan perustuva käyttö* (kuvio 19).

Kaavionäkökulmaan perustuvassa tavassa käytetään UML:n yksittäisiä kaaviotyyppejä valikoivasti ja satunnaisesti, tarpeen mukaan. Käyttöön ei sisälly systemaattisuutta eikä vakiintuneita toimintatapoja kuin enintään yksilötasolla, usein yhden ja saman kaaviotyypin hyödyntämiseen. Käyttö perustuu hyväksi havaittuun tapaan täyttää jokin tarve, yleensä viestinnällinen. UML-näkökulmaan perustuva tapa on järjestelmällisempi ja pohjaa koko kehitystyön UML:n käytölle ja sen tarjoamalle kokonaisuudelle (rakenne ja käyttäytyminen). Tapa sisältää useamman kaaviotyypin hyödyntämistä, mahdollisesti organisaation tai sen osan yleisten toimintatapojen ja ohjeistusten kautta.



KUVIO 19 UML:n käyttö

Suurin osa haastateltavista lukeutui kaavionäkökulmaan perustuvan tavan käyttäjiksi. Yhdeksän haastateltavaa tuotti itse yleensä vain yhdenlaisia UML-kaavioita. Kehitystyön aikana ei välttämättä hyödynnetty enempää myöskään muiden tuottamia kaavioita. Osan UML:n tuntemus oli hyvin pintapuolista. UML oli terminä ja mallintamiseen liittyvänä tuttu, mutta siitä ei välttämättä tiedetty tai muistettu juurikaan luokkakaaviota enempää. Osalle haastateltavista selvisikin vasta haastattelussa tai siihen valmistautuessa, että he käyttivät jotain UML:ään lukeutuvaa kaaviotyyppiä. Tietty kaaviotyyppi päätyi käyttöön yleensä siten, että se oli kokemuksen kautta todettu hyödylliseksi tai opintojen ajalta jäänyt mieleen.

> …myö käytettiin tosiaan koulussa [UML:ää] -- Et se tulee niin kun lähinnä tota sen, sillai että mä oon käyttänyt [sekvenssikaaviota] aikasemminkin, niin se on jollain tavalla tuttu. (tuotepäällikkö/sovellusarkkitehti H7)

> …kyllä niistä on niinku tuttuja asioita sieltä opiskeluajoilta, että miten niitä asioita siellä kuvataan. Et onks se nyt aktiviteettikaavioita tai joku semmonen, missä, missä tuota justiin ehkä tuntuu tutulle.. -- Sillä tavalla muistan kyllä, muistan kyllä UML:ää sieltä opiskeluajoilta. (tuotepäällikkö H11)

UML:n tuntemus saattoi olla hyväkin, mutta projektien luonne rajasi sen käyttöä. Ratkaisujohtaja H3 kertoi verkkokauppahankkeissa asiakkaille toimitettavista kolmannen osapuolen valmisratkaisuista, joiden yhteydessä ei ollut yleensä tarpeellista käyttää UML:ää laajamittaisesti.

Kolmestatoista haastateltavasta neljän käyttö ei perustunut satunnaisten kaaviotyyppien hyödyllisyyteen, vaan UML:n tarjoamaan yhtenäistettyyn (”unified”) näkemykseen. Kaikki neljä haastateltavaa piti mallintamista oleellisena osana kehitystyötä, mutta syyt järjestelmällisemmälle UML:n käytölle kuitenkin erosivat hieman toisistaan. Ohjelmistoarkkitehti H2 kritisoi UML:n tämänhetkistä työkalutukea kehnoksi ja osin myös kalliiksi. Tästä huolimatta UML oli hänen kokemuksensa mukaan soveltuvin olemassa oleva kuvaustapa, joten pohjasi kehitystyönsä sen käyttöön. Ratkaisuarkkitehti H9 oli samoilla linjoilla:

> …totta kai niin kun arkkitehdin roolissa, niin sehän [UML] on parasta ikinä. -- Ei maailman paras, mutta paras mitä meillä nytten on. -- …itse käytän sitä, niin pyrin, pyrin justiin tuohon, että niin kun tuossa [UML:n kaaviojaottelussa] näkyy, et se jakaantuu heti tuohon rakenteelliseen ja toiminnalliseen puoleen, että vähintään löytys jokaisesta palvelusta se ainakin yksi rakenteellinen kuva ja sitten yksi edes, mielellään useampikin tuommonen tota toiminnallinen kuvaus.

Projekti- ja palvelupäällikkö H4 ja tuotepäällikkö H6 esittivät myös ulkoisia tekijöitä merkittävinä syinä käyttää UML:ää. Heidän organisaationsa tai yhteistyökumppaneidensa toimintatapa joko kannusti tai velvoitti käyttämään UML:ää. Käyttöä perusteltiin myös UML:n standardiasemalla alalla.

> …UML on käytännössä sellanen de facto -standardi, jota voi olettaa, että kaikki IT-tekniset ihmiset pystyvät lukemaan ja pystyvät lukeen samalla tavalla. -- …mä toimin asiakkuudessa, joka itse käyttää tämmöistä kuin QPR:n softaa, niin siitähän löytyy kaikki UML:n tasot. Ni he itse kuvaavat omat prosessinsa, he kuvaavat järjestelmänsä… -- He tekee niin kun sen, me tehään siihen sitten teknisen toteutuksen kuvauksen, liitetään mukaan, teen niitä sekvenssikaavioita, teknistä suunnittelua ja sitten deployment-diagrammeja. (projekti- ja palvelupäällikkö H4)

> …UML:ää käytetään tota aika laajalti… -- …meilläkin nää isommat asiakkaat, sen niinku edellyttää et asiat pitää olla tietyllä tavalla kuvattu ja mallinnettu. (tuotepäällikkö H6)

UML-näkökulmaan pohjautuvien käyttäjien joukkoon kuuluivat yhtä lukuun ottamatta kaikki ne haastateltavat, jotka olivat opiskelleet mallintamista suhteellisen paljon verrattuina muihin haastateltaviin. Poikkeuksena oli pääarkkitehti H10, jonka työ perustui mallipohjaiseen, automatisoituun sovelluskehitykseen ja UML:n sijaan yrityksen omaan sovellusaluekohtaiseen kieleen. Toinen yhdistävä tekijä oli organisaation tuki. Ohjelmistoarkkitehtiä H2 lukuun ottamatta kaikki järjestelmällisesti UML:ää käyttävät kertoivat, että organisaatiossa oli ohjeistuksia ja suosituksia mallintamiseen liittyen. Haastatteluaineiston perusteella laaja-alainen mallintamisen opiskelu ja/tai organisaation tuki voivat johtaa myös sen laaja-alaiseen käyttöön.

5.4.2 Kaaviotekniikoiden käyttö

Haastatteluissa nousi esille selkeitä eroja käytettyjen kaaviotyyppien välillä, joten on hyödyllistä tarkastella niiden käyttöä myös erikseen. Kuviossa 20 esitetään kaikki haastatteluissa mainitut tunnetut kaaviotekniikat ja jaottelut sen mukaan, käyttivätkö haastateltavat niitä luonnostelussa[13], varsinaisessa suunnittelutyössä[14] vai takaisinmallinnettuina[15]. Muu käyttö tarkoittaa, että haastateltava kertoi organisaatiossaan/yksikössään/tiimissään jonkun muun käyttävän kyseistä tekniikkaa tai hyödyntävänsä sitä itse jollain muulla tavalla. Kuviossa 20 on vasemmalta lukien ensin UML:n rakennekaaviot, tämän jälkeen käyttäytymiskaaviot ja lopuksi UML:ään kuulumattomat kaaviotekniikat.

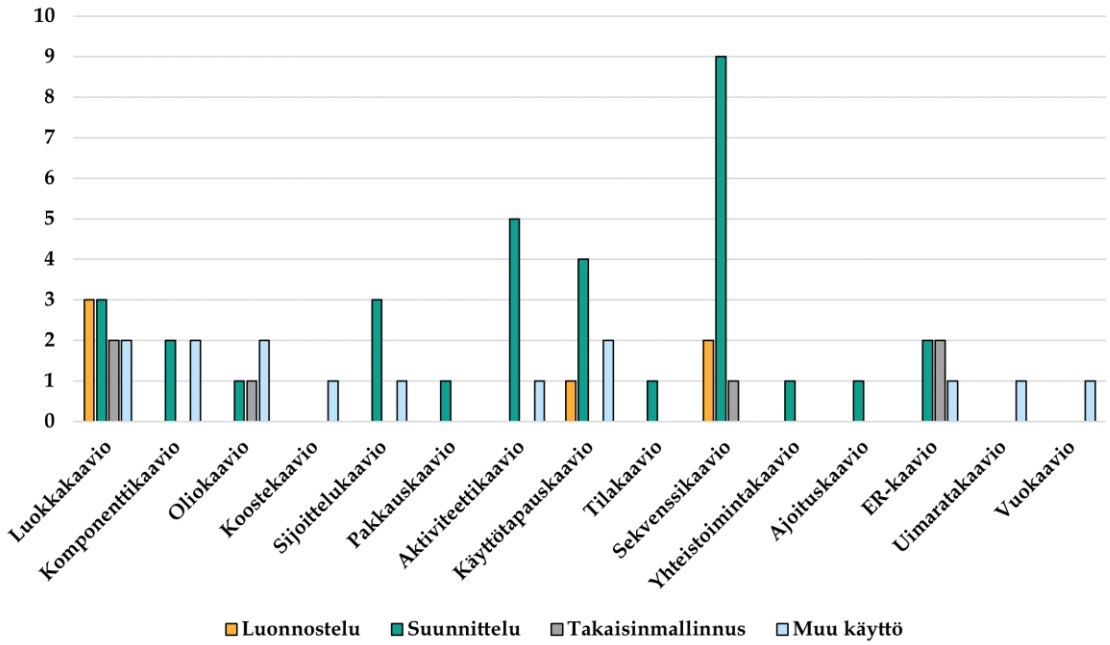
Sekvenssi-, aktiviteetti- ja käyttötapauskaaviot olivat käytetyimmät kaaviotyypit haastateltavien keskuudessa, etenkin varsinaisen suunnittelutyön osalta. Käyttö jakaantui eri työtehtävissä toimivien välille. Odotetusti osa arkkitehdeistä käytti muita ammattiryhmiä enemmän erilaisia kaaviotekniikoita ja osa muiden ammattiryhmien edustajista keskittyi ainoastaan käyttäytymispuolen tekniikoihin, mutta näidenkin suhteen oli hajontaa. Tämä johtui siitä, että osa arkkitehdeistä suosi vapaamuotoista mallintamista tunnettujen kaaviotekniikoiden sijaan ja siitä, että osalla haastateltavista oli useita ammattinimikkeitä.

---

[13] Luonnostelu esimerkiksi paperilla tai valkotaululla, ei dokumentointia

[14] Suunnittelu ja dokumentointi työkalulla

[15] Kaavion generointi koodipohjasta työkalun avulla



KUVIO 20 Kaaviotekniikoiden käyttö

Sekvenssikaavio oli selkeästi suosituin kaaviotekniikka haastateltavien keskuudessa. Jopa yksitoista haastateltavaa hyödynsi sekvenssikaaviota jollain tapaa työssään. Haastateltavista ainoastaan ohjelmistoarkkitehti H5 ja tuotepäällikkö H11 eivät maininneet sekvenssikaaviota ollenkaan. Arkkitehdit H10 ja H12 käyttivät kaaviota luonnostelussa ongelmanratkaisun ja viestinnän tukena, H12 myös takaisinmallinnettuna. Loput yhdeksän haastateltavaa tuotti ja dokumentoi sekvenssikaavioita työssään. Osalle kaavio oli keskeisin tai ainut aktiivisessa käytössä oleva kaaviotekniikka. Suunnittelutyö alkoi usein sekvenssikaavion piirtämisellä, sen avulla hahmoteltiin järjestelmän toiminnallisuus ja kommunikoitiin se muille. Sekvenssikaavio toimi merkittävänä tiedon välittäjänä ja ohjeistuksena.

Haastateltavien mukaan sekvenssikaavio oli erityisen hyödyllinen, kun kuvattiin rajapintojen välistä vuorovaikutusta. Kaaviotyyppiä käytettiin siis etenkin integraatiokeskeisissä projekteissa, joissa järjestelmiä ja siten myös rajapintoja oli määrällisesti paljon. Moni haastateltava kertoi, että kaaviotyyppiä hyödynnettiin lähinnä isommissa hankkeissa tai keskeisissä ja monimutkaisissa käyttötapauksissa, kuten järjestelmien välisissä tunnistautumisissa.

> …varsinkin jos on kyseessä joku tämmönen autentikaatio, koska, koska niin kun tää sanonta menee, security is hard, niin niissä tapauksissa tarvitaan yleensä jonkinlainen sekvenssi, sekvenssikaavio siitä, et mitenkä, mitenkä tota jutut juttelee keskenään. (tuotepäällikkö/sovellusarkkitehti H7)

> …varsinkin online- ja mobiiliapplikaatio, siinähän käyttäjä suoraan kommunikoi taustajärjestelmien kautta, joten siinä on tämmösiä sekvenssejä eli mistä pitää alkaa tunnistautumiset ja autorisoinnit ja muut eli siinä integraatio koostuu monesta tämmösestä pienestä osa-alueesta. Näissä mielellään käytetään sekvenssikaavioita. (projekti- ja palvelupäällikkö H4)

Jos rajapinnat olivat yksinkertaisia, haastateltavien mielestä tapahtumajärjestysten selventäminen kaavioiden avulla ei ollut tarpeellista, vaan sekvenssin kuvaaminen tekstimuodossa oli riittävää. Ratkaisujohtajan H3 mukaan sekvenssikaavioiden käyttö on vuosien varrella vähentynyt, koska integraatioarkkitehtuuri on muuttunut yksinkertaisemmaksi.

Haastateltavat olivat varsin tyytyväisiä sekvenssikaavioon ja sen käyttöön. Ohjelmistokehittäjät H1 ja H8 käyttivät tunnetuista kaaviotekniikoista ainoastaan sekvenssikaaviota jatkuvasti työssään. Molemmat haastateltavat kuvailivat kaaviotyyppiä helppokäyttöiseksi ja selkeäksi.

> …noilla on helppo käytännössä tehä nää ylätason määrittely, niin nähään suoraan, että miten se rajapinta toimii. (ohjelmistoarkkitehti/-kehittäjä H1)

> Mä olen itse tykännyt niistä. Mun mielestä se selventää asioita, että tota komponenttien välillä se vahva kommunikaatio tulee selväks niissä. -- Sekvenssi on aika suoraviivainen, se on hyvä kaaviomalli. (ohjelmistokehittäjä H8)

Ohjelmistokehittäjän H8 tapaan ohjelmistoarkkitehti H13 painotti sekvenssikaavion käyttöä komponenttitasolla:

> Että jos ja kun niitä teen, niin mä haluun pitää ehkä niin kun sen enemmän sellasella komponenttitasolla, että miten ne komponentit vuorovaikuttaa keskenään. Just sen takia, että mä tiedän, että sillon sillä kuvalla tai mallilla, niin sillä on sillon pidempi elinikä. Se, se on niinku ajantasaisempi aina.

Tunnettuja kaaviotekniikoita monipuolisemmin hyödyntävä ratkaisuarkkitehti H9 sanoi näin:

> …toiminnallisesta puolesta, niin tuo sekvenssidiagrammi on mun mielestä hyvin joustava ja jos sitä osataan käyttää, niin sieltä löytyy, siitä löytyy lähes kaikki mitä, mitä tarvii sinänsä kuvata.

Ainoastaan tuotepäällikkö/sovellusarkkitehti H7 mainitsi sekvenssikaaviosta jotain negatiivista:

> …siinä vaiheessa, kun osapuolia on hirveesti, niin kyl se alkaa ärsyttään hyvin nopeesti… -- …pieniä tämmösiä häiritseviä asioita, jotka niin kun, jotka selkeesti niin kun turhauttaa joka kerta kun niitä tekee…

Tuotepäällikkö/sovellusarkkitehti H7 selitti, että kun osapuolia oli paljon, joutui sekvenssikaaviossa piirtämään kutsuja osapuolten lävitse tai tekemään muunlaisia ratkaisuja, jotka tekivät kaavioista ajoittain vaikeaselkoisia. Hän kuitenkin lisäsi, että:

> …mä en tiiä, et onko tämä niin kun, olisko tässä standardissa joku, joku keino siihen…

Kuusi haastateltavaa kolmestatoista kertoi hyödyntävänsä **aktiviteettikaavioita**. Integraatioarkkitehtuurin mallintamiseen keskittyvä projekti- ja palvelupäällikkö H4 sai niitä asiakkaaltaan lähtötietoina prosessin etenemisestä eri toimijoiden ja järjestelmien välillä (muu käyttö). Ohjelmistoarkkitehti H2 käytti kaaviotyyppiä samaan tapaan kuin sekvenssikaaviota, suunnittelemansa järjestelmän sisältävien tapahtumien kuvaamiseen. Ratkaisuarkkitehti H9 suosi aktiviteettikaaviota sekvenssikaavion sijaan silloin, kun toiminnassa oli paljon päätöksiä. Ratkaisujohtaja H3 sekä tuotepäälliköt H6 ja H11 keskittyivät työssään toiminnallisen puolen mallintamiseen, jossa olennaisena osana oli liiketoimintaprosessien kuvaaminen ja niissä aktiviteettikaavion käyttö.

> …tuossakin järjestelmässä, mitä me ollaan tehty, niin siinä on, siinä taitaa olla useempi, no useempia kymmeniä tollasia liiketoimintaprosesseja, mitä siinä on sitten kuvattu, niin niillä tuota [aktiviteettikaavioilla], niin tuota, vaikea niitä ois ilman minkään sortin kaavioita kyllä ylläpitää, että. (tuotepäällikkö H11)

Aktiviteettikaavion soveltuvuus eri sidosryhmien välisessä kommunikaatiossa on tullut esille Dobingin ja Parsonsin (2006) ja Petren (2013) tutkimuksissa. Aktiviteettikaavio olikin haastateltavien keskuudessa yleisimmin hyödynnetty kaaviotyyppi asiakaskommunikaatiossa. Ratkaisujohtaja H3 kertoi kaaviotyypin olevan ”yleinen juttu, jonka vähän niinku kaikki ymmärtää”. Tuotepäällikkö H11 lisäsi, että vastuiden siirtyminen eri rooleissa olevien välillä on helppo selvittää aktiviteettikaaviosta, mutta selkeys edellytti kuitenkin tasapainoilua yksityiskohtaisen tiedon välittämisen ja abstrahoinnista saadun hyödyn välillä.

Sovellusarkkitehti H12 mainitsi muiden tekemät **uimaratakaaviot**, jotka hänen mielestään eivät vastanneet täysin aktiviteettikaavioita, ja ohjelmistoarkkitehti H5 **vuokaavioiden** tapaiset vapaamuotoiset kuvaustavat, joita mahdollisesti muut hänen organisaatiossaan käyttivät. Haastateltavat eivät yleisesti ottaen tienneet paljoakaan muissa kuin omissa tiimeissään tai projekteissaan suoritettavasta mallintamisesta, ohjelmistokehittäjän H8 sanoin: ”kaikki tekee omaa projektia”.

Ratkaisujohtaja H3 ja tuotepäällikkö H6 hyödynsivät **käyttötapauskaavioita** sovittaessaan yrityksensä tarjoamaa valmistuotetta asiakkaan tarpeisiin. Kaavioiden avulla tehtiin määrittelyjä, tarkennettiin toiminnallisuuksia ja suunniteltiin asiakaskohtaisia laajennoksia. Tuotepäällikkö/sovellusarkkitehti H7 käytti kaaviotyyppiä harvoin, koska yleensä suunnitteli yksittäisen käyttötapauksen ominaisuuksia. Käyttöä isommissa projekteissa hän perusteli organisaationsa tuotteeseen liittyvällä ohjeistuksella, jonka mukaisesti kaikki käyttötapaustilanteet oli hyvä nähdä yhdessä näkymässä. Ohjelmistoarkkitehti H2 kuvaili käyttötapauskaavion käyttöään hieman tarkemmin:

> Use case -diagrammia mä käytän monesti siihen, että kun mä lähen tekemään niinku tämmöstä kontekstuaalista mallia, elikkä mitkä ne on niinku ne, tavallaan ne neighbourin systeemit ja mitä aktoreita siinä niinku pyörii siinä sen järjestelmän ympärillä.

Käyttötapauskaavioita hyödynnettiin myös muilla tavoin. Projekti- ja palvelupäällikkö H4 sai kaavioita lähtötietoina asiakkaaltaan. Pääarkkitehdin H10 mallintaminen keskittyi yrityksen omaan mallinnuskieleen ja -teknologiaan, mutta käyttötapauskaaviota hän käytti luonnostelussa luokka- ja sekvenssikaavioiden tapaan.

> Siis ihan niin kun tällei viestinnällisestä, viestinnällisessä käytössä, että niissä on niin kun ilmasuvoimaa järkevällä tasolla, että sä saat riittävällä, riittävän pienellä vaivalla aika paljon niin kun tavallaan ilmaisuvoimaa yhteen kaavioon. Ja ne on sellasia yleiskäyttöisiä.

Ratkaisujohtaja H3 kertoi, että käyttötapauskaaviota voi hyödyntää monella eri abstraktiotasolla ja kuvaili kaaviota pääarkkitehdin H10 tavoin ilmaisuvoimaiseksi. Ohjelmistoarkkitehti H2 oli täysin vastakkaista mieltä:

> …jos puhutaan puhtaasti use case -mallintamisesta, ni siinä mun mielestä tuo UML ei oo niinku missään määrin hyvä työkalu. -- …siinä ei ole tarpeeksi ilmaisuvoimaa, että ne use caset pystyttäs niinku hyvin mallintamaan. -- …siihen sä et pysty niinku sanomaan niitä use casejä, että mä haluun tiettyjä pre conditioneja, mä haluan tiettyjä post conditioneja, mä haluun tietyt stepit sinne use casen sisällä, mitä se tarkottaa ja niistä diagrammeista tulee hirveitä sekasotkua.

Ohjelmistoarkkitehti H2 mainitsi mielestään käyttötapauskaaviota paremman vaihtoehdon:

> Siinä mun mielestä vanha kunnon Alistair Cockburnin use case -template[16] on paljon toimivampi, elikkä siihen pystyy niinku tarkemmin kuvaamaan sen sitten.

Ratkaisujohtajan H3 kokemukset vahvistavat, että käyttötapauskaavio jakaa mielipiteitä. Vaikka hän itse piti kaavioiden tekoa vaivattomana, hänen työyhteisössään moni muu oli vastakkaista mieltä. Ohjelmistoarkkitehti H13 ei tehnyt käyttötapauskaavioita eikä ottanut kantaa tekniikan soveltuvuuteen käyttötarkoitukseensa. Hän kuitenkin kertoi, että ei järjestelmään tutustuessaan pitänyt niitä varteenotettavina tiedon lähteinä, koska ne eivät yleensä olleet ajantasaisia.

Sovellusarkkitehti H12 työskenteli projektissa, joka oli kestänyt jo vuosien ajan ja oli haastatteluhetkellä ylläpitovaiheessa. Hänellä ei ollut nykyisessä projektissa tarvetta käyttötapauskaaviolle, mutta piti sen käyttöä tärkeänä uutta tietojärjestelmää suunnitellessa:

> No siinä tulee just mietittyä ne kaikki asiat läpi, et jos sä vaan niinku mietit päässäs, et okei, et käyttäjät tekee tämmösen toiminnon, etkä sä mieti sitä tarkkaan läpi, niin sit joku unohtuu sieltä. Ja sit jos se on jo tehty, isoja asioita siihen arkkitehtuuriin ja sit huomataan joku puuttuva ja se koitetaan jälkikäteen laittaa sinne mukaan, niin eipä ookaan enää helppoa. Ja sitten jos se budjetti ei saa ylittyä, niin sit tehdään semmosia hankalia ratkaisuja, mitkä ei oo kovin hyviä.

[16] ks. Cockburn (2001)

Ratkaisujohtajan H3 kokemuksen mukaan käyttötapauskaaviota ei kuitenkaan alalla useinkaan käytetä. Hänen mielestään vapaamuotoinen prosessien mallintaminen ja tekstimuotoiset määrittelyt olivat yleisemmin käytössä.

Luokkakaavio on UML:n keskeisin kaaviotyyppi (Rumpe, 2016) ja empiirisissä tutkimuksissa se onkin odotetusti ollut myös kielen käytetyin kaaviotyyppi. Haastateltavista kuitenkin ainoastaan kolme arkkitehtiä kertoi käyttävänsä luokkakaavioita varsinaiseen suunnitteluun. Ratkaisuarkkitehti H9 käytti luokkakaaviota tietomallien kuvaamiseen; tuotepäällikkö/sovellusarkkitehti H7 ainoastaan isoa tuoteominaisuutta suunnitellessaan. Ohjelmistoarkkitehti H2 kertoi luokkakaavion käytöstä ja sen hyödyistä osana loogista näkymää (ks. Kruchten, 1995) näin:

> …yleensä kaikista eniten on projektin sisällä kommunikaatiossa ollu hyötyä, ni on se niinku puhas tämmönen looginen näkymä siihen järjestelmään, koska sillon puhutaan tavallaan niistä bisneskonsepteista ja tavallaan niinku siihen domainin, domainin ötököistä. Ja ne on yleensä niitä mitä ne niinku ymmärtää, mitä niinku periaatteessa kaikki ymmärtää ketkä siinä projektissa on.

Muut haastateltavat esittivät paljon syitä sille, miksi luokkakaavioita ei käytetty. Tuotteistettuihin järjestelmiin muutoksia tehtiin usein kehitystyökalujen avulla, luokkakaavioita hyödyntämättä. Pienissä, moniosaajien tiimeissä luokkatason ohjeistuksia ei tarvittu ja kokeneet tekijät hahmottivat asiat ilman kaavioita tai koodia lukemalla. Arkkitehdit H1, H10 ja H13 kertoivat kuitenkin käyttävänsä luokkakaavioita tarvittaessa luonnosteluun, esimerkiksi palaverissa. Arkkitehdit H1 ja H13 hyödynsivät myös takaisinmallinnettuja luokkakaavioita tutustuessaan uuteen järjestelmään tai keskustellessaan järjestelmästä muiden kanssa. H13 hyödynsi näin myös **oliokaavioita**, joita ohjelmistoarkkitehti H2 käytti apuna käsitteiden mallintamisessa ja joita projekti- ja palvelupäällikkö H4 sai asiakkaaltaan lähtötietoina.

Ohjelmistoarkkitehti H2 kertoi käyttävänsä **sijoittelukaaviota** arkkitehtuurin mallintamisessa ja suunnittelussa; ohjelmistoarkkitehti H5 dokumentoidessaan tämänhetkistä tilannetta heidän kehittämänsä järjestelmän ympäristöstä. Arkkitehdin H5 käyttötarkoitus vastaa siten Kühnen (2006) esittämää kuvailevaa mallia ja arkkitehdin H2 ohjeellista mallia. Arkkitehdin H5 käyttö muistuttaa myös Petren (2013) määrittelemää käyttö jälkikäteen -tapaa (engl. retrofit), mutta Petren (2013) tapa pohjautui mallintajan ulkopuolelta (asiakas, esimies) tulevaan vaatimukseen. Arkkitehti H5 käytti sijoittelukaaviota, koska oli tietojärjestelmäarkkitehtikoulutuksen yhteydessä huomannut käytön hyödyllisyyden.

Projekti- ja palvelupäällikön H4 integraatio- ja API-hallinnan tiimi kuvasi sijoittelukaavioilla palveluiden sijainnin. Hän korosti kaavion merkitystä ylläpidossa:

> …näitä viedään sekä on-premiseen että järjestelmiä on pilvessä, se integraatiokin saattaa olla pieni palanen, istua esimerkiksi siellä pilven kulmalla.. ja sithän niitä voi olla Amazonia tai Azurea, Googlea, IBM:ää, palasia sitä sun tätä. Eli se deployment on hyvin tärkeä ylläpidon ja palvelun kannalta, että minne on viety mitäkin ja mistä

> se komponentti vastaa. – Ja jos meillä ei oo niitä deployment-diagrammeja ja siellä on jotain 500 integraatiota, niin siinä on aikamoinen laskeminen, et missäs mun kaikki lampaat on…

Kun projekti- ja palvelupäällikkö H4 sanoi sijoittelukaavion olevan hyödyllinen on-premise -ja pilvipalveluratkaisujen sijoittelun kuvaamisessa, oli ratkaisuarkkitehti H9 eri mieltä.

> …se tuntuu niin kun tukevan enemmän tämmösiä vanhanaikaisia niin kun staattisia konesaleja ja muita. -- …se johtaa monesti sitten siihen, että käytetään, käytetään vaikka jonkun pilvipalvelun omia, omia tuota kuvia tai muita vastaavia, joita itse, itse kutsun kyllä ihan markkinointikuviksi, että ne ei oikein diagrammeina toimi. Koska ne on enemmän semmosta vapaamuotoista piirtelyä sitten. Että, että siinä tuo UML on vähän jäänyt jälkeen, että maailma on mennyt tonne pilvimaailmaan, niin sitä on välillä vähän hankala kuvata sitten.

Käyttötapauskaavion ja sijoittelukaavion tavoin **komponenttikaavio** jakoi mielipiteitä. Haastateltavista arkkitehdit H2 ja H12 tekivät komponenttikaavioita. Tämän lisäksi tuotepäälliköt H6 ja H11 kertoivat, että vaikka eivät itse rakennekaavioita piirtäneet, heidän organisaatiossaan tai tiimissään arkkitehdit käyttivät tätä tekniikkaa luokkakaavion lisäksi. Ohjelmistoarkkitehti H2 käytti komponenttikaaviota arkkitehtuurin mallintamisessa, jossa hyödynsi joustavuutta UML:n eri kaaviotyyppien rajojen välillä (ks. OMG, 2017, s. 685).

> …[käytän] semmosta vähän sekasotkua komponentti- ja deployment-diagrammeista.

Sovellusarkkitehti H12 oli esittänyt ratkaisua asiakkaalle komponenttikaavion avulla. Hän kertoi kaaviotyypin soveltuvan hyvin tähän käyttötarkoitukseen korkean abstraktiotason vuoksi. Ratkaisuarkkitehti H9 oli eri mieltä. Hänen mielestään komponenttikaaviota ei yleensä ymmärtänyt sen lukija eikä myöskään tekijä.

> ..jos käyttäis esimerkiks komponenttidiagrammia, niin muut ihmiset ei välttämättä ymmärrä sitä… -- Ihmiset ei yleensä ymmärrä, ymmärrä sitä interfacejä ja muita, miten se piirretään, näitä tikkareita, vaan sinne yleensä pistetään sitten muitten diagrammien notaatioita sekasin ja sillon se ei enää oikein, oikein natsaa siihen tarkoitukseen. Ja sit siitä tulee semmonen sekasotku.

Hän itse käytti **pakkauskaaviota**, jota piti parempana vaihtoehtona järjestelmän rakenteen kuvaamiseen ja sen kommunikointiin muille.

> …se joustaa, joustaa aika hyvin niin kun korkeammallekin tasolle, että en oo, en oo itse ainakaan saanut suostuteltua itseäni käyttämään jotain muuta. -- …siinä on tiettyjä hyötyjä, että käyttää, käyttää semmosia diagrammeja, mistä on muillekin hyötyä, koska sitä muut ymmärtää, koska se on se tarkoitus näissä malleissa. (ratkaisuarkkitehti H9)

Muista UML-kaavioista haastatteluissa mainittiin **tilakaavio** ja **yhteistoimintakaavio**, joita tuotepäällikkö H6 kertoi käyttävänsä ainakin joskus. Tiiminsä kehityspuolella hän sanoi nähneensä luokka-, komponentti-, sijoittelu- ja oliokaavion lisäksi myös **koostekaavion** käyttöä. Ratkaisuarkkitehti H9 kertoi käyttävänsä **ajoituskaaviota** sulautetuissa järjestelmissä, joissa jopa sekunnin tuhannesosien eroilla on merkitystä.

UML:n kaavioiden lisäksi tietojärjestelmien mallintamiseen merkittävästi vaikuttanut **ER-kaavio** (Gogolla, 2011; Storey ym., 2015) nousi odotetusti esille haastatteluissa. Kaksi haastateltavaa kertoi käyttävänsä ER-kaavioita suunnitteluun ja kaksi hyödyntävänsä takaisinmallinnettuja ER-kaavioita. Pääarkkitehti H10 kertoi ER-kaavion olevan nykypäivänäkin hyvä kaaviotekniikka ja yrityksensä sovellusaluekohtaisen kielen tietomallinnuksen pohjautuvan pitkälti siihen (muu käyttö). Hän lisäsi, että asiakasvaatimukset dokumentaation suhteen koskivat usein tietomalleja.

> …se tietohan on sitä heidän omaisuuttaan. Softat tulee ja menee, mut tieto pysyy. Niin kyl se tietorakenne, tietorakenteet on se olennainen, sitä ne yleensä pyytää.

Tietokantojen mallintamiseen keskittynyt ohjelmistoarkkitehti H5 käytti pääasiallisena mallinnustapanaan ER-kaaviota. Hän kertoi käyttävänsä sitä kehitystyön alkuvaiheessa, suunnitellessaan tuoteominaisuuden tietomallia. Tuotepäällikkö/sovellusarkkitehti H7 tarkensi käyttävänsä tietomallin suunnittelussa Crow's Foot -notaatiota (ks. Everest, 1976).

> …mä yleensä käytän tota Crow's Footia eli sehän ei oo UML:ää, mutta tota mää ite tykkään siitä enemmän… -- …mää oon alottanut sillä, koulussa käytettiin sitä, niin sitten, sitten tota se, se on jotenkin tota iskostautunut, että se tulee sieltä sitä kautta.

Taulukossa 7 esitellään kaikki kaaviotekniikat, joita käsiteltiin haastattelujen aikana tarkemmin ja joista haastateltavilla oli mielipiteitä. Taulukossa mainitaan kaavioiden käyttötarkoitus ja hyödyt sekä käyttöön liittyvät haasteet.

TAULUKKO 7 Tunnettujen kaaviotekniikoiden käytön hyödyt ja haasteet

Kaaviotekniikka	Käyttötarkoitukset	Hyödyt	Haasteet
Sekvenssikaavio	Integraatioprojektit; toiminnallisuuden kuvaaminen; rajapintojen määrittely, toteutus ja dokumentointi; järjestelmien ja komponenttien välinen vuorovaikutus; järjestelmän sisäisten tapahtumien kuvaaminen; monimutkaiset ja keskeiset käyttötapaukset, kuten tunnistautuminen	Selkeä; suoraviivainen; joustava; monipuolinen; helppo tehdä ylätason määrittely	Sekava, jos osapuolia on paljon
Aktiviteettikaavio	(Liiketoiminta)prosessien mallintaminen; järjestelmän sisäisten tapahtumien kuvaaminen; toiminnallisuuden kuvaaminen, kun paljon päätöksiä; asiakaskommunikaatio	Selkeä; myös maallikot ymmärtävät	Sekava, jos sisältää paljon tekstiä
Käyttötapauskaavio	Käyttötapausten määrittely; toiminnallisuuksien tarkentaminen; isoissa projekteissa, kun useita toiminnallisuuksia; kontekstuaalinen mallintaminen	Ilmaisuvoimainen; helppo piirtää ja päivittää; kaaviosta näkee helposti kaikki käyttötapaukset, tulee mietittyä kaikki tilanteet läpi ennen toteuttamista	Ei tarpeeksi ilmaisuvoimaa, kaaviosta tulee sekava, Alistair Cockburnin template on hyödyllisempi; ei yleensä ajantasainen
Luokkakaavio	Tietomallin kuvaaminen; ison tuoteominaisuuden suunnittelu; käsitteiden mallintaminen; looginen näkymä; takaisinmallinnettuna järjestelmään perehtyminen	Kuvaa järjestelmän käsitteet, joita kaikki sidosryhmät ymmärtävät; edesauttaa järjestelmään perehtymistä	
Sijoittelukaavio	Arkkitehtuurin mallintaminen; järjestelmän ympäristön kuvaaminen; palveluiden sijainnit	Tärkeä ylläpidon kannalta	Ei sovellu pilvi-infrastruktuurin kuvaamiseen
Komponenttikaavio	Arkkitehtuurin mallintaminen; useiden järjestelmien yhteensovittaminen	Absraktioltaan sopivan korkealla tasolla	Notaatiota ei yleensä ymmärretä (rajapinnat)
Oliokaavio	Käsitteiden mallintaminen; tiedon mallintaminen; takaisinmallinnettuna järjestelmään perehtyminen	Auttaa tunnistamaan käsitteiden välisiä riippuvuuksia; edesauttaa järjestelmään perehtymistä	
Pakkauskaavio	Järjestelmän rakenteen kuvaaminen	Ymmärrettävä; joustaa korkeammalle abstraktiotasolle	
ER-kaavio	Tiedon mallintaminen; tietokantojen mallintaminen; tietomallin suunnittelu	Edelleen hyvä kaaviotekniikka	

5.4.3 Työkalujen käyttö

Monen muun mallintamista käsittelevän empiirisen tutkimuksen tavoin (ks. esim. Badreddin ym., 2018; Petre, 2013) myös tässä haastatteluaineistossa nousi

esiin ammattilaisten tyytymättömyys mallintamiseen käytettäviä työkaluja kohtaan. Ohjelmistoarkkitehdillä H2 oli paljon kokemusta erilaisista työkaluista ja hän oli tyytymätön kaikkiin tarjolla oleviin vaihtoehtoihin. Työkalut eivät hänen mielestään tukeneet mallintamistyötä eivätkä kommunikaatiota. Jo luvussa 5.3.1 ohjelmistoarkkitehti H2 kertoi, kuinka joutui kommunikaatiota edistääkseen käyttämään kaavioiden ulkonäön viimeistelyyn erittäin paljon aikaa ja vaivaa.

> …kun sää teet mallintamista, niin sunhan pitäs sillon keskittyä siihen mitä sä teet ja se työkalu ei saa niinku viedä sitä sun huomioo… -- …mikään noista työkaluista ei oikein niinku vastaa siihen tarpeeseen. Mulla menee liikaa aikaa siihen pikselikohdan hieromiseen.

Ohjelmistoarkkitehdin H2 tavoin useita eri vaihtoehtoja läpikäynyt ohjelmistoarkkitehti/-kehittäjä H1 totesi näin:

> No en mää tiiä onko niissä oikeen mitään positiivista, mä en oo löytäny semmosta hyvää työkalua…

Aiemmista tutkimuksista poiketen haastateltavat eivät esittäneet mallinnustyökalujen monimutkaisuutta haasteena tai mallintamistyötä rajoittavana tekijänä. Päin vastoin, kaksi haastateltavaa jopa painotti, että mallinnustyökalut olivat yleisesti ottaen melko yksinkertaisia, eikä oppimiskäyrä muodostunut merkittäväksi esteeksi niiden käytölle. Työkalujen monimutkaisuus ei noussut esille haastatteluissa mahdollisesti siksi, että työkaluilla haastateltavat viittasivat yleensä piirtotyökaluihin. Osa haastateltavista ei välttämättä ollut edes kokeillut varsinaista, koko järjestelmän suunnittelun ja toteuttamisen mahdollistavaa mallinnustyökalua. Tällaista raskaampaa työkalua käytti haastatteluajankohtana ainoastaan projekti- ja palvelupäällikkö H4. Hänen integraatiotiiminsä käytti **QPR**:n ohjelmistoa[17], koska se oli käytössä heidän palvelemassaan asiakkuudessa.

Suurin osa haastateltavista käytti erilaisia piirtotyökaluja kaavioiden tekoon. Microsoft **Visio**n käytöstä kertoi kymmenen haastateltavaa, joista kuusi täsmensi käyttävänsä sitä itse haastatteluhetkellä. Loput kertoivat Vision aiemmasta käytöstä tai sen käytön yleisyydestä organisaatioissaan. Ainoastaan projekti- ja palvelupäällikön H4, ohjelmistokehittäjän H8 ja ratkaisuarkkitehdin H9 haastatteluissa Visiota ei mainittu ollenkaan. Maksullisesta lisenssistä huolimatta Visio oli monelle helposti saavutettavissa oleva työkalu.

> Luultavasti Microsoft on siinä tehny tavallaan niinku jotain itselleen oikein, että samasta paikasta, mistä on saanu ladattua kehitystyökalut, niin on sitten voinut ladata ton, tota ton työkalun, millä sitten piirretään noita malleja. (ohjelmistoarkkitehti H13)

Visiota kuvailtiin monipuoliseksi. Tuotepäällikkö/sovellusarkkitehti H7 sanoikin Vision tarjoavan vaihtoehdoista laajimman paketin. Tuotepäällikön H6 ja

[17] https://www.qpr.com/fi

ohjelmistoarkkitehdin H13 tavoin hän käytti Visiota pääasiallisena mallinnustyökalunaan. Pääarkkitehti H10 käytti työkalua asiakasdokumentaatiossa tarpeen mukaan, muutoin käytössä oli organisaation oma koodia generoiva mallinnustyökalu. Kuten myös ohjelmistoarkkitehti H13, hän suosi Visiosta löytyviä valmiita kaaviopohjia ja muotoja, joita pystyi hyödyntämään moneen eri tarkoitukseen. Pääarkkitehdillä H10 oli kuitenkin enimmäkseen negatiivista sanottavaa Visiosta:

> Se oli ennen UML-mallinnustyökalu eli tavallaan sä pystyit ihan oikeesti tehä, siellä oli ikään kuin UML-malli taustalla ja sä pystyit tekemään useita kaavioita ja sit kun sä muutit sitä tavallaan mallii, niin se näky kaavioissa ja näin edespäin. Ja se niinku toimi, et näin edespäin. Mutta sittenhän ne muutti sen sellaseks piirtelytyökaluks ja eiiii, siis se on vaan yksinkertaisesti käytännössä niin kömpelö, että tota noin niin, pitäs varmaan ettii joku toinen. Sattuu tuleen Officen mukana, niin oon käyttänyt sitä. Ihan hirvee.

Pääarkkitehdin H10 tavoin ohjelmistoarkkitehti/-kehittäjä H1 piti Visiota kankeana, eikä ollut tyytyväinen työkalun tuottamaan jälkeen. Kaavioiden piirtäminen halutunlaiseksi oli hänen mielestään liian vaivalloista.

> …jotenkin se on vähän semmonen, ehkä kömpelö. -- …ei ne kuvat ei oo oikeen semmosia kun mää haluaisin tehä. -- Ne näyttää semmosilta vähän, niinku ois lyijykynällä piirretty tommosta. Saahan siinä sitä väriä ja muutaki tehtyy, mut taas työlästä ruveta tekemään mitään vähän semmosia.

Kukaan haastateltavista ei suoraan maininnut Vision olevan monimutkainen työkalu. Tuotepäällikkö H6 kuitenkin totesi sen käytön vaativan opettelua. Myös ohjelmistoarkkitehti/-kehittäjä epäili, että Vision käyttöön liittyvät ongelmat mahdollisesti ratkeaisivat, jos työkalua käyttäisi enemmän.

Sovellusarkkitehti H12 oli aiemmin käyttänyt Visiota, mutta lisenssi oli hänen mielestään liian kallis vähäiseen mallintamiseen nähden. Haastatteluhetkellä hän käytti netistä ilmaiseksi ladattavaa piirtotyökalua **Gliffy**ä[18], jota myös ohjelmistoarkkitehti/-kehittäjä H1 suosi Vision sijaan. Myös viisi muuta haastateltavaa käytti mallintamiseen pääasiallisesti ilmaista, kevyttä piirtotyökalua, kuten **Draw.io**:ta tai **PlantUML**:ää. Käyttäjistä moni oli sitä mieltä, että työkalut sopivat tarkoitukseensa paremmin kuin raskas mallinnustyökalu tai maksullinen, kattavammat ominaisuudet tarjoava piirtotyökalu.

> …se on niin tavallaan, ehkä vähäistä ollut se meidän mallinnus vielä tähän asti ainakin, niin ei oo sillä tavalla ehkä koettukaan, että ois tarpeellista olla semmonen oikee työkalu sit siihen. (ohjelmistoarkkitehti H5)

> …kun näitä malleja ei, ei hirveesti linkitetä toisiinsa, niin siinä ei oo hirveesti hyötyä käyttää jotain raskaampaa työkalua. -- Parempihan se ois, olisi totta kai, jos niitä saatas linkitettyä toisiinsa ja, ja sillei luotua semmonen kokonaiskuva siitä järjestelmästä,

[18] https://www.gliffy.com/

> mutta se on jostain syystä nähty sitten liian raskaana tapana toimia. Ja sille ei oo, ei olla nähty arvoa tai ei ymmärretty sen arvoa. (ratkaisuarkkitehti H9)

Kokonaiskuvan muodostamisen sijaan työkaluissa arvostettiin etenkin niiden soveltuvuutta ketterään kehitykseen ja wikityylisen dokumentaation tuottamiseen. Kevyiden työkalujen edut olivat pitkälti siinä, että ne saatiin yhdistettyä saumattomaksi osaksi dokumentaatio- ja työnohjausjärjestelmiä, kuten Atlassianin organisaatiowikiohjelmisto Confluencea[19] ja tehtävienhallintaohjelmisto Jiraa[20]. Tämän lisäksi työkalut mahdollistivat joustavan yhteistyön, jota ei esimerkiksi Visiolla saavutettu.

> Se Gliffy nyt on siitä hyvä, että se on siellä wikisivussa upotettuna, niin kuka vaan voi editoida, että se on tavallaan hyvä. Kun Visio oli silleen, että ku vaan harvoilla on se lisenssi, niin ne pystyy sitä muokkaan. (sovellusarkkitehti H12)

Arkkitehdit H5 ja H9 kertoivat Draw.io:n olevan monipuolinen verrattuna muihin vastaaviin työkaluihin.

> On tietysti monia muitakin, mitkä integroituu tämmösiin niin kun wikeihin, mutta Draw.io on ollu hyvä siitä, et sillä pystyy piirtämään aika monia UML-malleja ja myös sitten ArchiMate-notaatiota ja muitakin notaatioita. (ratkaisuarkkitehti H9)

Ratkaisuarkkitehti H9 muistutti kuitenkin, että ilmaiset työkalut eivät tarjonneet kaikkia haluttuja toimintoja:

> …eihän Draw.io täydellinen mitenkään ole, että kyllähän siihen kaipais totta kai tuota linkityksiä ja muuta tämmöistä keskitettyä, jaettua repositorya malleille ja niin edespäin, ja versionhallintaa ja tämmöstä.

Moni haastateltava kertoi, että heidän organisaatioissaan suosittiin ilmaisversioita tai Microsoft Officen mukana tulevaa Visiota. Kaikki eivät kuitenkaan osanneet sanoa, miksi tietyt työkalut olivat käytössä. Ratkaisuarkkitehti H9 mainitsi, että työkalujen käyttö oli yleensä projekti- tai hankekohtaista. Tätä vahvisti tuotepäällikön H11 kommentti:

> En tiedä millä perusteilla se [OmniGraffle-työkalu] on aikanaan valittu. (naurahdus) Siinä kohtaa, kun minä sain ne käsiini, niin ne on tehty sillä ja minä oon sitten päivittäny sillä samalla. (tuotepäällikkö H11)

Kuten luvussa 5.3.1 kerrottiin, haastateltavat eivät yleisesti ottaen aktiivisesti etsineet uusia työkaluja:

> No se, mitä mulla on valmiiks ollu koneella, käytän niitä. -- …ku oot sitä ehkä jotain editoria käyttäny ja sit sää käytät sitä aina. Jos ei oo pakko vaihtaa toiseen, niin ei niitä vaihda. (ohjelmistoarkkitehti/-kehittäjä H1)

---

[19] https://www.atlassian.com/software/confluence
[20] https://www.atlassian.com/software/jira

> No siis korporaatiomaailmassa fakta on se, että sä saat käyttää vaan sitä mihin sulla on varaa ostaa lisenssi. Mihin firma suostuu ostamaan lisenssin. Ja sen takia mää käytän näitä työkaluja. (ohjelmistoarkkitehti H2)

> Kai me ollaan jollain tavalla niinku organisaationa kasvettu niihin työkaluihin mitä meillä tällä hetkellä on käytössä, että. -- Niin, mistähän nekin on lopulta tullu (naurahdus). Joku ne on valinnu joskus (naurahdus). (ratkaisujohtaja H3)

> …on tullu vanhaksi ja mukavuudenhaluiseksi, en mää jaksa taas kaikkii uusia lähtee ees miettimäänkään, et mää pärjään tolla nyt ainakin tällä hetkellä varsin hyvin, et. (tuotepäällikkö H6)

Ohjelmistokehittäjän H8 tärkeimmät valintakriteerit työkalulle olivat ketteryys ja sekvenssikaavion luominen tekstinotaatiolla piirtämisen sijaan. Hän kertoi **Lucidchart**in olevan yleisesti käytössä heidän projekteissaan, mutta itse suosi haastatteluhetkellä avoimen lähdekoodin PlantUML-työkalua:

> …sä voit kirjottaa sen kaavion, sun ei tartte piirtää niitä nuolia hiirellä. Naputat menemään. (naurahdus) Sellasella niinku tekstinotaatiolla. Se on nopeeta.

Myös ratkaisuarkkitehti H9 ja ohjelmistoarkkitehti H2 kertoivat tekstinotaatiota tukevien työkalujen eduista, vaikka eivät sellaisia haastatteluhetkellä käyttäneetkään.

> Mun pitää niinku saada keskityttyä siihen mallintamiseen ja se kuvan syntyminen on niinku ihan sivuseikka. -- …mä haluan vaan merkata sinne ne, et mulla on tämmösiä asioita ja nää on ne riippuvuudet, anna mulle nyt hienoin mahdollinen kuva tästä ku tilanteesta. Se ois optimaalista. (ohjelmistoarkkitehti H2)

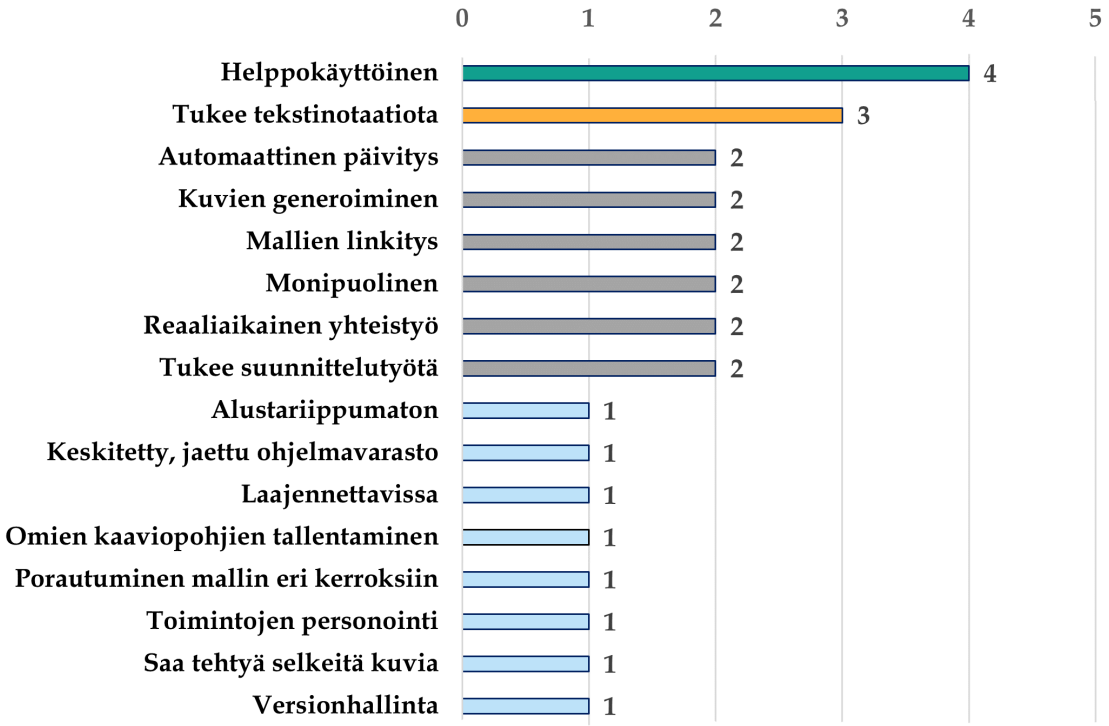
> …kun se on tekstipohjainen, sen saa pistettyä vaikka Gitin versionhallintaan, et sillä tavalla saadaan niin kun se versionhallinta siihen mukaan. -- Joku tykkää mieluummin kirjottaa, koska aiemmin aina niin nää jotkut työkalut oli graafisesti vähän heikkoja, että, että se mallin päivittäminen oli raskasta, jos sinne piti väliin saada joku uusi, uusi viiva, niin sillon se koko kaavio piti melkein käytännössä piirtää uudestaan. Mutta jos käyttää esimerkiks PlantUML:ää, niin se, silloin se teksti, kun päivittää sitä tekstiä, niin se työkalu itse sitten päivittää sen diagrammin ja reitittää ne kaikki nuolet sen mukaan, että se näyttää jotakuinkin järkevältä. (ratkaisuarkkitehti H9)

Mallien päivittämiseen liittyvät haasteet nousivat usein haastatteluissa esiin, eivätkä kaikki piirtotyökalut tukeneet tehtävää halutulla tavalla. Sovellusarkkitehti H12 kuvaili Gliffyn käytettävyyttä näin:

> …viivat ei mene sillai niinku hyvin, automaattisesti ja. Ei se kyllä paljosta ois kiinni, että se ois hyvä. Tai niin kun riittävä mulle, että ehkä se semmonen käytettävyys vois olla parempi.

The Omni Groupin piirtotyökalua **OmniGraffle**a[21] käyttävä tuotepäällikkö H11 sen sijaan oli tyytyväinen sillä tuotettuun jälkeen ja kaavioiden päivittämiseen. OmniGrafflen heikkoudet liittyivät sen rajatumpaan käyttöön. Tuotepäällikkö H11 kertoi, että maksullisen lisenssin lisäksi työkalu oli yhteensopiva ainoastaan Mac-käyttöjärjestelmän kanssa.

Kuviossa 21 esitetään haastateltavien mainitsemat toivotut ominaisuudet työkaluihin liittyen sekä niiden esiintyvyys haastatteluissa.



KUVIO 21 Työkalujen toivotut ominaisuudet

Vaikka haastatteluissa käsiteltiin suurimmaksi osaksi piirtotyökaluja (taulukko 8, jossa esiintyvyys ilmaisee, kuinka monella haastateltavalla työkalu oli haastatteluhetkellä käytössä), myös toisenlaisista työkaluista oli osalla haastateltavista kokemusta. Ohjelmistoarkkitehdillä H2 oli ollut pitkän aikaa käytössä **StarUML**, jonka sanoi avoimen lähdekoodin vaihtoehdoista olevan lähinnä varsinaista mallinnustyökalua. Ohjelmistoarkkitehti H5 käytti tietokantasuunnittelussa **Quest Toad Data Modeler**ia[22], joka mahdollisti tietokantojen kokonaisvaltaisen hallinnan ja generoinnin. Tuotepäällikkö/sovellusarkkitehti H7 käytti Microsoftin **SQL Server Management Studion Database Diagram** -työkalua[23] tietokannan takaisinmallintamiseen.

[21] https://www.omnigroup.com/omnigraffle
[22] https://www.quest.com/products/toad-data-modeler/
[23] https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver15

TAULUKKO 8 Käytössä olevat piirtotyökalut

Piirtotyökalu	Esiintyvyys	Syyt käyttää / hyödyt	Haasteet
Visio	6	Helposti saavutettavissa; markkinoiden laajin paketti; paljon kaaviopohjia ja muotoja	Kömpelö, käyttö vaivalloista; kallis lisenssi; ei tue yhteistyötä; vaatii opettelua; jälki näyttää lyijykynällä piirretyltä
Draw.io	3	Helppokäyttöinen; ilmainen; joustava; kevyt; monipuolinen; integroituu muihin järjestelmiin; tukee yhteistyötä	Ei keskitettyä, jaettua ohjelmavarastoa; ei mallien linkitystä; ei versionhallintaa
Gliffy	2	Helppokäyttöinen; ilmainen; kevyt; integroituu muihin järjestelmiin; tukee yhteistyötä	Kaavioiden päivittäminen vaivalloista
OmniGraffle	1	Helppokäyttöinen; kaavioiden päivittäminen vaivatonta	Maksullinen lisenssi; saatavilla ainoastaan Macille
PlantUML	1	Helppokäyttöinen; ilmainen; kevyt; tukee tekstinotaatiota; kaavioiden päivittäminen vaivatonta; mahdollistaa versionhallinnan	

Osalla haastateltavista tai organisaatioista oli aiemmin ollut käytössä raskaampia, varsinaisiksi mallinnustyökaluiksi luokiteltavia ohjelmistoja, jotka oli todettu liian jäykiksi ja siten tarpeettomiksi etenkin vain satunnaisesti kaavioita hyödyntävään kehitystyöhön. Ohjelmistoarkkitehti H13 ei ollut edes harkinnut tällaisen työkalun käyttöä:

> Mä luulen, että mun luovuus ei sit enää, niin, se ei sitten enää siinä toimis. Onneks saa piirtää vapaasti.

Arkkitehdit H2 ja H12 mainitsivat aiemmissa empiirisissä tutkimuksissa (ks. esim. Fettke, 2009) esillä olleen mallinnustyökalun **Rational Rose**n (nykyään IBM Rational Software) olevan liian kallis vaihtoehto. Haastatteluissa nousi esiin useammin **Sparx Systemsin Enterprise Architect**, josta oli kokemusta muutamalla haastateltavalla. Yhdelläkään se ei ollut haastatteluhetkellä käytössä. Haastateltavien kommenttien mukaan työkalu ei täysin vastannut nykypäivän tietojärjestelmäkehityksen vaatimuksia vanhanaikaisuuden ja jäykkyyden vuoksi. Lisäksi sen lisenssi oli maksullinen. Ratkaisuarkkitehti H9 kuitenkin kertoi Sparx Systemsin tai QPR:n ohjelmistojen käytön olevan vaatimuksena silloin, kun työskennellään kokonaisarkkitehtuuritasolla ja on tarve keskitetylle ohjelmavarastolle ja mallien linkitykselle toisiinsa. Vaikka kokonaisarkkitehtuuri ei tullut esille projekti- ja palvelupäällikön H4 haastattelussa, todennäköisesti heidän asiakkaansa toteutti sitä QPR:n Enterprise Architect -työkalulla.

Jopa seitsemän haastateltavaa kertoi, että kaavioita tehtiin heidän organisaatiossaan tai projekteissaan myös Microsoftin esitysgrafiikkaohjelmisto **PowerPoint**illa. Heistä kolme käytti PowerPointia itsekin, lähinnä asiakasdokumentaatioon. Aihetta käsitellään enemmän seuraavassa luvussa 5.5 Vaihtoehtoiset tavat.

5.5 Vaihtoehtoiset tavat

UML:n ja tunnettujen kaaviotekniikoiden sekä mallinnus- ja piirtotyökalujen käytön lisäksi tässä tutkimuksessa havaittiin myös vaihtoehtoisia tapoja mallintaa. Tässä luvussa esitellään kaikki haastatteluissa esiin tulleet tavat, niiden käytön syyt ja hyödyt sekä niihin liittyvät haasteet.

5.5.1 Vapaamuotoinen mallintaminen

Haastattelujen perusteella vapaamuotoinen mallintaminen on yleistä nykypäivän tietojärjestelmäkehityksessä, etenkin asiakaskommunikaatiossa. Haastateltavien asiakkaat olivat usein liiketoimintaihmisiä, joilla ei ollut puoliformaalien kuvaustapojen, kuten UML-kaavioiden ymmärtämiseen vaadittavaa osaamista. Vapaamuotoisilla kuvilla pyrittiin edesauttamaan yhteisymmärrystä ja asiakkaan päätöksentekoa.

> …jos mä sanon, että teksti saa katseen lasittumaan, niin kyllä se UML-kaaviokin siten, jos ihminen kattoo sitä myyntimateriaalia niin kun sekunnin tai kaksi, niin se on ahistavaa sitten yrittää keskittyä johonkin tommoseen, jos niinku oikeesti ei oo varma, että ees siis haluaako tän ratkaisun itelleen vai ei. Et se, se on pakko pitää maholisimman yksinkertaisena. (tuotepäällikkö H6)

> …saadaan se joku pelko siitä pois siitä keskustelusta, että tulee, että asiakas ei koe, että tulee nolanneeksi, nolanneensa itsensä, jos ei ymmärrä jotain. Niin sillon semmoset hyvin, hyvin yksinkertaiset, vapaamuotoiset piirrokset on parempia. (ratkaisuarkkitehti H9)

Vapaamuotoisilla malleilla ei ole määriteltyjä säännönmukaisuuksia, joten ne ovat oletettavasti monitulkintaisia ja näin alttiita väärinymmärryksille. Haastateltavat eivät pitäneet tätä ongelmana silloin, kun kyse oli asiakkaan kanssa käydystä keskustelusta näiden mallien avulla. Asiakkaan piti yleensä vain jollain tasolla ymmärtää esimerkiksi projektin vaatima työmäärä ja näin osata tehdä rahoituspäätös.

Haastateltavat kertoivat, että mallintaessa oli otettava huomioon, millaiselle ihmiselle malleja esitetään. Jos vastassa oli henkilö, jolla oletettavasti ei ollut teknistä taustaa, haastateltavat eivät noudattaneet UML:ää, vaan tekivät vapaamuotoisia malleja. Joissakin tapauksissa haastateltavat muuntelivat UML-kaavioita niin, että uskoivat niiden olevan ymmärrettäviä myös vastapuolen silmissä. Tämä tukee Chaudronin ym. (2012) väitettä siitä, että ohjelmistoammattilaiset tasapainoilevat malliensa viestinnällisen arvon ja UML:n noudattamisen välillä. Haastateltavat kuvailivat tasapainottelua näin:

> …tiedon välittäminen on tärkeetä ja se tulee siitä, kuinka paljon siellä on asiakkaalla sitä niin kun liiketoimintayksiköiden henkilöitä, joiden tarvii sitä ymmärtää, kuinka paljon siellä on taas sitten oikeesti IT-henkilöitä, jotka on teknisiä, niin se määrittelee sen, että minkälainen on esitystapa asiaan. (projekti- ja palvelupäällikkö H4)

> …en mä oo niin välittäny [notaation formaaliudesta]. Taas toisaalta sit, jos siinä vastapäässä on joku, joka tietää nää tosi tarkasti, niin sitten pitäs kyllä olla oikee notaatio, että on se uskottavuus siinä niin. (sovellusarkkitehti H12)

> …asiakkaillekin näytetään sitten ainakin jotain, jonkinnäköistä versiota [sijoittelukaaviosta]… -- Et vähän niinku ehkä siistitty sitä tarkkuustasoa ehkä pois sitten, otettu pois semmoset ei-asiakkaalle merkitsevät asiat, ehkä niistä asioista sitten. (ohjelmistoarkkitehti H5)

> …esimerkiks tietomallissa, ni hyvä, on jo erittäin kyseenalaista, että haluatko käyttää perintää vai et, koska se, jolle sää viestit sitä, niin on yleensä ihminen, joka ei ole alan asiantuntija ja mitä enemmäns siellä on jotain hämärän näköisii notaatioita, kuten, se on erittäin hämärää, niin tuota sitä vaikeemmaksi se viestiminen käy. Puhumattakaan sitten assosiaatioista, joissa alat miettimään sitä, että alat sä tekemään jotain koostesuhteita sinne, niin tai tälläsii, niin en mää ikinä käytä niitä, varsinkaan asiakkaalle viestimisessä. (pääarkkitehti H10)

Suurin osa haastateltavista piti vapaamuotoisten kaavioiden käyttöä riittävänä kuvaustapana jopa teknisessä suunnittelussa. Myös projekti- ja palvelupäällikkö H4 ja ratkaisuarkkitehti H9, jotka yleisesti ottaen suosivat standardoituja mallinnuskieliä kertoivat, että välillä oli hyödyllistä käyttää asiakaskommunikaatiota varten tehtyä mallia myös tiimin kesken:

> Mutta yleensä riittää tekniselleki eli ei tekninen tarvitse, että ”tohon sää oot piirtänyt ton mallin, saanks mä tän UML:nä”, kyllähän me se ymmärretään. (projekti- ja palvelupäällikkö H4)

> …riippuen tietysti niin kun ajasta, että onko, onko aikaa piirtää UML-kaavioita, niin saatetaan tiimillekin esittää ihan sama asia. Että siinä on yleensä se kustannustehokkuus järkevä, järkevä pitää mielessä. (ratkaisuarkkitehti H9)

Arkkitehdeille H1 ja H13 vapaamuotoisten kuvien tuottaminen ja käyttäminen oli yleisempää kuin tunnettujen, puoliformaalien kaaviotekniikoiden käyttö. Ohjelmistoarkkitehti H13 kertoi roolinsa vuoksi mallintavansa paljon, mutta luottavansa enemmän omiin, intuitiivisiin taitoihinsa kuin säännönmukaisuuksien noudattamiseen tai niihin ohjaavan mallinnustyökalun käyttöön. Hän oli saanut kaavioidensa selkeydestä positiivista palautetta, mikä oli rohkaissut häntä tekemään aiempaa enemmän kaavioita projekteissa. Tämä tukee luvussa 5.3.1 esitettyä ratkaisuarkkitehdin H9 näkemystä siitä, että mallintaminen voidaan sivuuttaa, jos ei uskota omiin taitoihin mallintajana. Ohjelmistoarkkitehti H13 perusteli vapaiden tekniikoiden käyttöä sillä, että se mahdollisti sujuvan ja mielekkään työskentelyn.

> …se pitää saada se tietynlainen ajatus päästä ulos ja kommentoitua, niin sen takia mä tykkään niinku enemmän vapaista menetelmistä.

Suurimmat erot vapaamuotoisiin malleihin suhtautumisessa koskivat näiden kuvien käyttöä kehittäjien välisessä työssä ja virallisessa dokumentaatiossa. Ohjelmistoarkkitehti H2 ei hyväksynyt tätä mielestään alalla valitettavan yleistä toimintatapaa, koska se johti väärinymmärryksiin ja lopulta epätyydyttävään lopputulokseen. Ratkaisuarkkitehti H9 myönsi vapaamuotoisten kuvien käytön olevan joskus tarpeen myös kehitystiimin kesken, mutta piti väärinkäsityksiä ja viestintäongelmia alalla yleisinä ja siten tunnettujen mallinnuskielien käyttöä perusteltuna. Osa haastateltavista oli sitä mieltä, että vaikka vapaamuotoinen mallintaminen voi johtaa väärinymmärryksiin, ei se niinkään johdu käytetystä kuvaustavasta, vaan mallintajan taidoista.

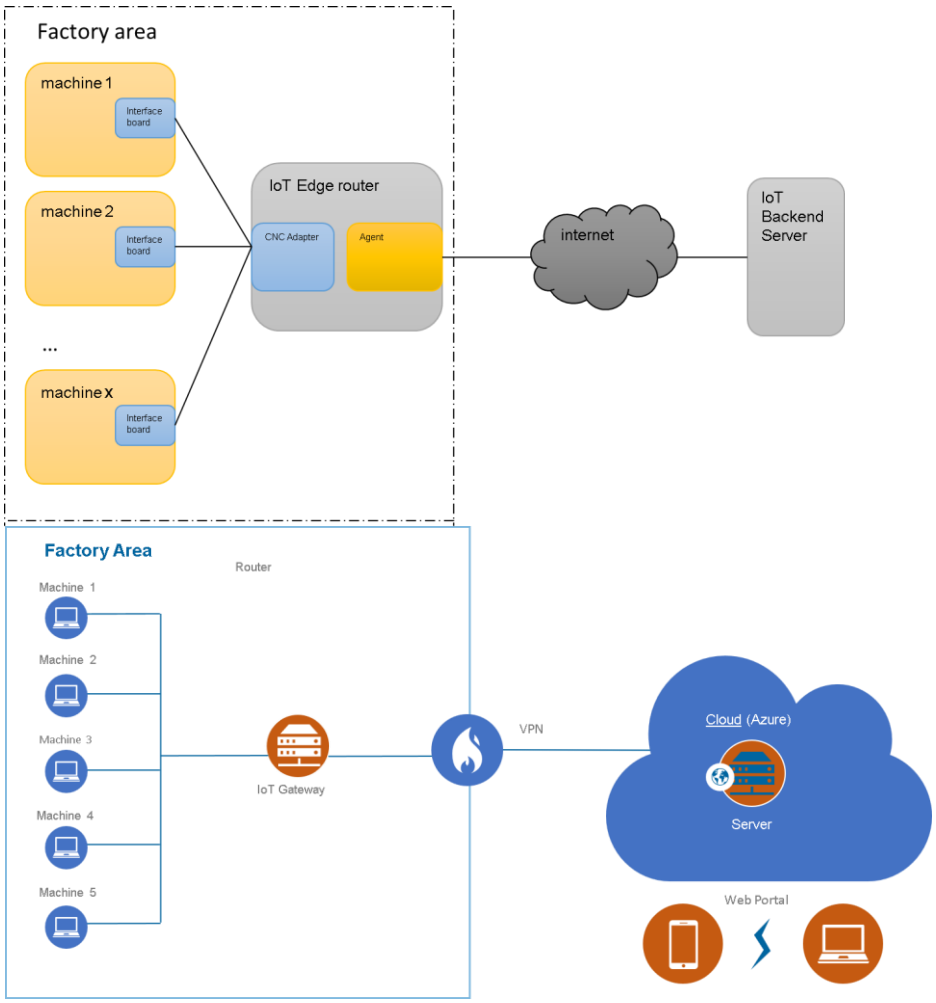
> …väärinymmärryksen mahdollisuus ei tuu suinkaan siitä notaatiosta mitä käyttää vaan, vaan sillai niin kun enemmän siitä, että sä teet niinku mallintajana, et ota jotain tiettyä askelta niin kun huomioon, mikä pitäisi siinä, siinä prosessissa ottaa. (tuotepäällikkö/sovellusarkkitehti H7)

> …[vapaamuotoisessa mallintamisessa] se riski niinku kasvaa. Kasvaa tehdä sitten niin kun, tehdä niinku loogisia virheitä tai niinku insinöörivirheitä, et tekee siellä, piirtää väärän viivan, mikä ei oo mahollinen ja sitten taas se toinen puoli, se taidepuoli on mun mielestä se, että miten sä saat sen kommunikoitua sille toiselle katsojalle, muulle kuin itselles, niin ne niinku tavallaan yhdistyy siinä. Sen takia se on myös semmonen niinku luova prosessi, et siitä saa sitten vapaamuotoisesta kuvasta semmosen, et se kykenee täyttään ne kaks tarkotusta, myös sen faktan ja sit sen laadukkuuden siinä, että se on pätevä ja selkeä ja loogisesti jäsennelty, yhtä lailla kuin se ohjelmakoodi. (ohjelmistoarkkitehti H13)

Ohjelmistoarkkitehti/-kehittäjä H1 kuvaili tekemiään vapaamuotoisia piirroksia ”ylemmän tason laatikkokuviksi”. Hän sanoi näiden kuvien olevan tärkeimpiä kuvia kehitystyön alussa ja ylläpidossa. Piirroksilla kuvattiin koko järjestelmän arkkitehtuuri ja ympäristö, ja niistä tuli hahmottaa selkeästi rajapinnat ja palveluiden sijainnit.

> …notaatiolla ei käytännössä väliä siinä tuota, siinä nähään vaan, että onko joku pilvessä, onko joku jossain konesalissa, onko se, missä päin yleensäkin sijaitsee, jos tarvii niihin koskee. Tai sit onko se vaan tämmönen yhen serverin tuote missä se pyörii. Lähinnä semmonen laajempi ympäristökuva. Se on yleensäkin se, minkä minä haluan ensimmäisenä, jos mä tiian, et rupeen tekemään tai ruvetaan tekemään jotain isompaa järjestelmää. (ohjelmistoarkkitehti / -kehittäjä H1)

Kuviossa 22 esitetään ohjelmistoarkkitehdin/-kehittäjän H1 PowerPointilla piirtämiä yksinkertaisia esimerkkikuvia vapaamuotoisesta mallintamisesta. Ylemmässä esimerkkikuvassa on käytetty ”laatikkoja ja viivoja” -tyyliä, alemmassa on hyödynnetty pilvipalveluun liittyviä symboleja.



KUVIO 22 Esimerkkejä vapaamuotoisesta mallintamisesta

Myös ratkaisujohtaja H3 kertoi tällaisten järjestelmäarkkitehtuurikuvien olevan vapaamuotoisia, vaikka muutoin projekteissa suosittiin UML:n käyttöä. Ohjelmistoarkkitehti H5 lisäsi aiempaan kommenttiinsa, että sijoittelukaavion version tilalla saatettiin toisinaan käyttää täysin yksinkertaistettua, vapaamuotoista ympäristökuvaa. Pääarkkitehti H10 kertoi, että yksityiskohtaisten UML-kaavioiden sijaan vapaamuotoiset ”järjestelmän kartat” olivat tyypillisiä malleja dokumentaatiossa. Käytännöt olivat hänen mielestään yllättävän vaihtelevia.

> …puhutaan arkkitehtuurista, puhutaan jostain järjestelmän infrastruktuurista tai verkkoinfrastruktuurista tai sitten niin kun loogisesta järjestelmien välisistä suhteista ja näin edespäin ja niihin ei oo oikeen löytyny mun mielestä sellasta, niihin ei oikeen ees ole kunnollista yhtenäistä dokumentaatiota tai notaatiota dokumentaatioon ja sillon ollaan kyllä siinä tilanteessa, että aika lailla keksitään päästä mitä millonkin kuvataan -- Hirveen vaihtelevaa, täytyy sanoo, että on paljon vaihtelevampaa kuin mitä esimerkiksi niinku vois kuvitella ottaen huomioon, että olemassa on myös standardejakin.

Myös projekti- ja palvelupäällikkö H4 mainitsi integraatioiden yhteydessä ”järjestelmän kartan”, jota QPR:ssä kuvattiin integration architecture -nimisellä kaaviolla. Kaaviossa esitettiin yleiskuvaus kaikista integraatiotapauksista yhte-

nä staattisena kuvana. Tällainen kaaviotyyppi ei lukeutunut UML:ään. Samaan tapaan tuotepäällikkö/sovellusarkkitehti H7 käytti Vision tarjoamaa verkkoarkkitehtuurikaaviopohjaa ja ohjelmistokehittäjä H8 kertoi pilvipalveluiden omien notaatioiden käytöstä.

Vapaamuotoisten kuvien tekoon käytettiin samoja työkaluja kuin puoliformaaleihin kaavioihin, mutta käytössä oli myös muita välineitä. Paperi ja valkotaulu olivat ymmärrettävästi suosittuja kuvausalustoja tiimipalavereissa ja itsenäisessä suunnittelussa. Etäpalavereissa saatettiin käyttää piirtoalustana myös ohjelmistoja, kuten Microsoft Paintia. Moni haastateltava kertoi, että PowerPoint soveltui hyvin asiakaskommunikaatioon.

> …se on pikkusen tyylikkäämpi, kun niinku mitä Visiolla saa. Niin sitten ja sitten, kun asiakkaalla on yleensä vähemmän teknisiä henkilöitä, niin mä oon ajatellut, että ne arvostaa ehkä sitä niin kun sillai semmosta pientä päälleliimattua kauneutta enemmän kuin sitä, että se on pikkusen rumempi ja sitten niin kun tarkempi. (tuotepäällikkö sovellusarkkitehti H7)

> PowerPointtiin piirrellään kaikenlaisia laatikko ja nuoli -leikkejä… -- …ne on kyllä todella ad hoc, niillä ei ole niin kun mitään sovittua syntaksia. (naurahdus) Se on mitä sattuu. -- …pikkusen tuunaamalla sitä erinäköseks ja laittamalla väriä siihen, niin sit se alkaa jo aukeemaan paremmin. (pääarkkitehti H10)

Suurin osa haastateltavista kertoi, että tiesivät PowerPointia käytettävän kaavioiden tekoon ainakin jollakin tasolla heidän projekteissaan tai organisaatioissaan. Ohjelmistoarkkitehti/-kehittäjä H1 suosi dokumentoidessaan Gliffyn ohella PowerPointia, jolla oli hänen mielestään vaivatonta tehdä vapaamuotoisia kuvia:

> PowerPointit. Sinne vaan laatikkoo laatikon perään. Viivoja väliin.

Ohjelmistoarkkitehti H2 oli jyrkästi tällaista ”PowerPoint-mallintamista” vastaan:

> …organisaatiothan käyttää mielellään semmosta PowerPoint-mallintamista, mikä on niinku se, että PowerPointtiin piirretään neljä laatikkoa ja. Kaikki järjestelmät pystyy mallintamaan niin sanottuna put-get -arkkitehtuurilla, missä on kaks laatikkoa ja niitten välillä on put- ja get-operaatioita, niin se on se organisaatioiden lähtötaso mallintamiseen. – No, eihän se kerro mitään, mitä ne laatikot on. Siis sen takiahan nää isot, tota organisaatiot, niitä UML:iä ja muita määrittelee, että siinä määritellään oikeesti, että jos mulla on tuossa ton näköinen laatikko, niin mitä se tarkoittaa, mikä sen laatikon semantiikka on ja sit jos siitä lähtee ton näköinen nuoli, niin mitä se nuoli tarkoittaa. Ja sehän on kaikissa, jos sää rakennat jotain, niin kyllähän sun pitää ymmärtää, et mikä se on siinä. Mitä ollaan tekemässä. Jos vaikka taloa rakennat, ja siinä menee jonkinnäköinen viiva, niin onko se sähköjohto vai viemäriputki, niin siinä on aika iso ero.

”PowerPoint-mallintamista” vastaavaa termiä käyttää myös Kruchten (2009) kuvaillessaan arkkitehtuurin mallintamisen käytäntöjä ajalta ennen julkaisu-

aan ”4+1 näkymästä” (Kruchten, 1995). Kruchten (1995; 2009) noteeraa ohjelmistoarkkitehdin H2 tavoin ”laatikkoihin ja nuoliin” liittyvän semantiikan puutteen ongelmallisuuden. Hän kuitenkin toteaa, että tällainen ”PowerPoint-tason dokumentointi” on edelleen hyödyllistä tiettyjen sidosryhmien kanssa kommunikoitaessa (Kruchten, 2009).

Monella haastateltavalla oli ohjelmistoarkkitehtiä H2 sallivampia näkemyksiä vapaamuotoisesta mallintamisesta.

> …kunhan se asia tulee ymmärrettäväksi ja se on kuvattu riittävällä tarkkuudella, niin tota se kuvan esittämistapa siinä mielessä tota voi olla joustava. (tuotepäällikkö H6)

> …mä luulen, että tämmöset niinku standardoidut kielet, niin niistä on otettu oppia, ne on tullu niinku osaksi sellasta kulttuuriperimää ja sitten vähän niin kun justeeraten, kaikki mallintelee vähän mitä mallintelee, mutta tuntuu, että näyttää kaikki niin kun toisiamme ymmärtävän. (pääarkkitehti H10)

Haastatteluista kävi ilmi, että työkalulla oli suuri merkitys sillä tehtävän kaavion sisältämään notaatioon tai kuvaustapaan. Osa haastateltavista ei etukäteen päättänyt käytettävää notaatiota tai välttämättä edes tiennyt, tekikö UML-kaaviota vai täysin vapaamuotoista kuvaa. Piirtotyökalusta usein valittiin sopivia eri kaaviopohjia tai muotoja tarpeen mukaan ja niistä muodostettiin haluttu kuva.

> No, [käytetty notaatio] on se, mitä se tarjoo se (naurua) työkalu. -- Jos se nyt tarjoo UML-notaatiota, niin sitä voidaan käyttää… (ohjelmistoarkkitehti/-kehittäjä H1)

> En mä niinku välttele [UML:n kaaviopohjien käyttöä], et monesti mä otan niinku vaan kivannäköisiä muotoja valmiiksi ja sit lähen siitä tekemään ja siitä tulee niinku semmonen mun näköinen kuva ennemminkin. Ja sen tilanteen mukaan. (ohjelmistoarkkitehti H13)

> …esimerkiks AWS:n ympäristöihin kun tehdään, niin siellähän on käytetty semmosta tietyntyyppistä notaatiota myös, erilaisiin komponentteihin vaikka ja muihin, niin tota, niin. En, en osaa sanoa, että vastaisko tää notaatio sitten UML:ää lainkaan vai. (ohjelmistokehittäjä H8)

Osa haastateltavista kertoi, että näki alalla käytettävän myös muunlaisia kuvaustapoja kuin UML:ää. He eivät useinkaan osanneet sanoa, olivatko nämä tavat täysin vapaamuotoisia vai noudattivatko ne kenties jotain tunnettua kaaviotekniikkaa. Ohjelmistokehittäjä H8 mietti, että harvemmin tulee täysin omia notaatioita keksittyä ja totesi ainakin omien vapaamuotoisten tekniikoiden yleensä pohjautuvan UML:ään.

Ohjelmistoarkkitehti H2 näki ongelmallisena sen, että organisaatiot suhtautuivat leväperäisesti vapaamuotoisten mallien käyttöön, määriteltyjä toimintatapoja kun ei ollut. Semantiikan puutetta ei korvattu millään, vaan oli yleensä tekijän ja lukijan vastuulla, miten näitä ympäripyöreitä kaavioita tulkittiin ja väärinymmärryksiltä vältyttiin.

> …se vähän vaihtelee. Sun pitää vaan tietää, että mistä se kaveri puhuu. (ohjelmistoarkkitehti H2)

Arkkitehdit H10 ja H13 olivat sitä mieltä, että vapaamuotoiset tavat vaativat tuekseen lisäselvitystä.

> Vaikka sä keksit itse notaation, niin sit pitää kirjottaa lukuohje siihen, että mitä tämä tarkoittaa. -- …niissä on aina se huono puoli, että ne pitää aina selittää. Hirveen pitkä lukuohje. (pääarkkitehti H10)

> Väärinymmärryksen vaara, no sitä tietysti voi olla. Se on ihan varteenotettava riski, mut sitten tota kommunikointi on ihan hirveen tärkeetä kanssa, että se niinku se kuva just kirjotetaan tai puhutaan auki… (ohjelmistoarkkitehti H13)

Haastatteluissa esiintyneet syyt käyttää vapaamuotoista mallintamista sekä tapaan liittyvät haasteet löytyvät taulukosta 9.

TAULUKKO 9 Vapaamuotoisen mallintamisen käytön syyt/hyödyt ja käyttöön liittyvät haasteet

Syyt käyttää / hyödyt	**Esiintyvyys**	**Haasteet**	**Esiintyvyys**
Asiakaskommunikaatio	12	Täytyy miettiä vastapuolen osaamista	7
- yhteisymmärryksen saavuttaminen	7	Monitulkintaisuus, väärinymmärrykset	3
- päätöksenteon edesauttaminen	3	Vaatii taitoa mallintajalta	3
Tekninen suunnittelu, tiimin kesken	11	Vaatii lukuohjeen tai keskustelua	2
- kustannustehokkuus	2	Yhteisymmärrystä ei varmisteta	1
Dokumentaatio	7		
Asiakkaat/kehittäjät eivät halua montaa eri kuvaa	3		
Ei ole standardia kuvaustapaa / tarvittavaa kuvaustapaa ei lukeudu UML:ään	2		
Saa tehdä haluamallaan tavalla ja nopeasti	2		

5.5.2 Mallipohjainen tietojärjestelmäkehitys

Pääarkkitehti H10 ei luonnostelun ja dokumentaation lisäksi lukeutunut muiden haastateltavien tavoin perinteisen mallintamisen hyödyntäjäksi. Pääarkkitehdin H10 organisaatio H kehitti mallipohjaisen tietojärjestelmäkehityksen keinoin asiakaskohtaisia liiketoimintajärjestelmiä. Hän kutsui tapaa automatisoiduksi sovelluskehitykseksi. Organisaatio oli kehittänyt oman, korkean abstraktiotason sovellusaluekohtaisen kielen ja koodia generoivan työkalun. Pääarkkitehti H10 oli jo opiskeluaikoina kiinnostunut sovellusaluekohtaisesta mallintamisesta ja nykyään piti sitä ylivertaisena tapana kehittää ohjelmistoja. Hänen kokemuksensa mukaan mallipohjaisella ohjelmistokehityksellä oli useita etuja perinteiseen kehitykseen verrattuna, kuten tehokkuus- ja uudelleenkäytön hyödyt ja koodaajille mielenkiintoisemman työn tarjoaminen rutiinitöiden automatisoinnin vuoksi. Asiakkaiden näkökulmasta hän sanoi näin:

> …projekti/prosessitasolla niin löytyy tällasia asioita kuten asiakkaille, kun me pystytään nyt demonstroimaan niin kun järjestelmää sitä mukaa, kun me sitä määritellään,

> niin ne on kokeneet sen sitouttavana ja innostavana ja erittäin miellyttävänä niin kun tapana määritellä järjestelmää.

Pääarkkitehdin H10 organisaation oma mallintamismenetelmä pureutui liiketoiminnan taustajärjestelmiin, mutta näiden ohessa he toimittivat myös perinteisin menetelmin tuotettuja osajärjestelmiä, kuten verkkopalveluja. Myös automaatiolla tuotettuihin järjestelmiin vaadittiin lisäksi räätälikoodin kirjoittamista. Toiminta vastasi Whittlen ym. (2013) näkemystä, jonka mukaan onnistuneessa mallipohjaisessa kehittämisessä organisaatiot yhdistävät sitä joustavasti muihin menetelmiin. Haastattelu ei kuitenkaan tue Whittlen ym. (2013) tutkimustulosta siitä, että koodigenerointi ei olisi mallipohjaisuudessa avainasemassa (ks. myös Mohagheghi ym., 2013). Pääarkkitehti H10 piti juuri koodigeneroinnin tarjoamia lyhyen tähtäimen hyötyjä niin merkittävinä, että mallipohjaisuus oli hänen mielestään perinteistä ohjelmistokehitystä kannattavampaa.

> …ei siitä pelkästä mallista nyt niin paljon hyötyä ole. Mut sit kun sä yhdistät siihen sen, että sä saat siitä mallista tavaraa ulos, niin sit se alkaa kiinnostaa.

Näkemys vastaa mallipohjaisen kehittämisen puolestapuhujan Bran Selicin (2003) toteamusta siitä, että mallintamisen hyödyt tulisi nähdä nimenomaan koodigeneroinnin kautta pelkän dokumentaation sijaan. Whittlen ym. (2013) ja Mohagheghin ym. (2013) tutkimusten organisaatiot poikkeavat pääarkkitehdin H10 organisaatiosta H siten, että ne tuottivat ohjelmistoa lähinnä omiin tarpeisiinsa, eikä niiden toiminta perustunut mallipohjaisuuteen. Nämä tekijät voivat selittää näkemyseroja tehokkuuteen ja tuottavuuteen liittyen.

Myös UML:ää voidaan käyttää sovellusaluekohtaiseen automatisoituun tietojärjestelmäkehitykseen. Tällöin stereotyyppejä lisäämällä luodaan sovellusaluekohtainen UML-profiili. Whittlen ym. (2013) tapaan pääarkkitehti H10 ei pitänyt tapaa optimaalisena.

> …[UML] on niin yleiskieli, niin yleinen. -- …mä en oikeen oo, en oo ees ite keksiny miten sitä niin kun järkevästi pystys niin kun mallintaa tai sit se ois ihan järkyttävää niin kun stereotypioiden, stereotyyppien niin kun viljelyä joka paikkaan. Mut sit alkaa herätä kysymys, että no, jos mulla nyt näitä stereotyyppejä on joka paikassa, niin miksen mää nyt vaan tekis omaa kieltä, jossa nää käsitteet on niin kun sitä, mitä mää haluan niitten olevan.

Pääarkkitehti H10 ei nähnyt UML:n viimeisimpien versioiden mukaista kehitystä mielekkäänä:

> …ainakaan monimutkaisemmaks sitä ei kannata viedä tai sit se on kuollut kieli. Et tota noin niin, ehkä siitä kannattas yrittää kehittää jotakin, joku semmonen niinku profiili, joka olis niin kun käytännönläheinen. Hyväksyttäs se tosiasia, että ei, ei näitä piirtele niin kun tällä tasolla kukaan.

UML:n käyttö mallipohjaisessa kehittämisessä ei noussut esille muissa haastatteluissa, vaikka mallipohjaisuutta muuten pohdittiinkin. Haastateltavat näkivät mallipohjaisessa kehittämisessä enemmän rajoitteita kuin hyötyjä. Haastateltavat mielsivät jo pelkän mallipohjaisen kehittämisen idean hyvin pitkälti toimimattomaksi ja vanhanaikaiseksi. Tuotepäällikön H11 kommentti oli hyvin kuvaava:

> Eikös se oo vähän semmosta vanhanaikasta, vähän vanhemman mallin tekemistä. Mulla on semmonen mielipide siitä ehkä. Vesi-, vesiputoukseen liittyvää. -- …saman kommentin oon kyllä kuullut muiltakin, että se saattaa olla vähän vanhemman liiton juttuja.

Koodigeneroinnin nykytilasta kysyttäessä moni haastateltava vastasi, että sitä hyödynnettiin heidän organisaatioissaan vain vähän tai ei ollenkaan, tai että ei ollut asiasta aivan varma. Monella haastateltavalla oli kuitenkin aiempaa kokemusta koodigeneroinnista, johon liittyvät epäonnistumiset vaikuttivat heidän mielipiteisiinsä mallipohjaisesta kehittämisestä. Koodigeneroinnin käyttö nähtiin myös täysin vastakkaisena toimintana tehokkaalle ja taidokkaalle koodaamiselle, joten mallipohjaiseen kehittämiseen voidaan epäillä liittyvän negatiivisia asenteita, ainakin koodikeskeisimpien ammattilaisten joukossa.

> …sitä käytiin ehkä sillon alussa, kun ei ollut vielä paljon kokemusta, mutta sen jälkeen niin tota en mää käytä sitä, nopeempi tehä käsin ite se koodi, ku tietää mitä tekee. (ohjelmistoarkkitehti/-kehittäjä H1)

> Joskus sitä [koodigenerointia] on kokeiltu 2000-luvun alussa, mutta aina se jotenkin mätti. Ei se toiminut. (sovellusarkkitehti H12)

Ratkaisuarkkitehti H9 oli sitä mieltä, että vaikka ajatus on mielenkiintoinen, ei mallipohjainen kehittäminen yksinkertaisesti toimi. Hän selitti, että onnistuminen vaatisi hyvin tarkkaa mallintamista ja toisaalta myös käsin koodaamista rinnalla, joten kustannustehokkuus ei olisi halutulla tasolla. Merkittävänä esteenä hän näki parin muun haastateltavan lisäksi myös koodigenerointia tekevien työkalujen riittämättömyyden. Osa haastateltavista ei kuitenkaan tuntunut olevan aiheesta kiinnostuneita, vaikka teknologia sen mahdollistaisikin, mikä vastaa Whittlen ym. (2013) havaintoa koodiguruista ja eri teknologioita testailevista, mallipohjaisuutta vastustavista ammattilaisista.

> Ei mua varmaan mikään, mikään saa kiinnostumaan. Mä oon valinnu tämän polun, niin mä meen tällä. (sovellusarkkitehti H12)

> …mää oon sen verran vanhan liiton koodari, ni en mää käytä semmosia… (ohjelmistoarkkitehti/-kehittäjä H1)

Haastateltavista moni lisäsi, että kiinnostuksen ja soveltuvan työkalutuen puutteesta huolimatta suhtautuu avoimin mielin koodigeneroinnin ja mallipohjaisuuden tarjoamiin mahdollisuuksiin. Ohjelmistokehittäjä H8 ja ohjelmistoarkki-

tehti H13 totesivat, että mallipohjaisuus todennäköisesti sopii moneen tarkoitukseen, mutta pitivät perinteistä tapaa helpompana ja sillä tuotetun koodin elinkaarta pidempänä. Ainoastaan tuotepäällikkö/sovellusarkkitehti H7 osoitti pääarkkitehdin H10 lisäksi selkeästi kiinnostusta mallipohjaista kehittämistä kohtaan ja näki vaihtoehdot luovina rajoittavien sijaan, yhdistäen ne ketteryyteen. Hän oli kiinnostunut etenkin ratkaisujen vaivattomasta demoilusta. Kuitenkaan edes hän ei uskonut, että teknologia tällä hetkellä taipuisi tehtävään halutulla tavalla.

Pääarkkitehti H10 ymmärsi mallipohjaisuuteen kohdistetun kritiikin, koska toiminta vaati onnistuakseen paljon osaamista ja investointeja sekä sitoutumista valittuun linjaan.

> …siis kyl sen saa toimimaan, mutta se on, se ei oo ihan halpaa ja et se vaatii ihan oikeesti investointeja ensinnäkin ihan puhtaasti teknologiaan, se vaatii sitoutumista eli käytännössä suomeksi tarkoittaa, että rahaa sen jatkuvaan kehittämiseen, eteenpäin viemiseen, se vaatii sitä, että sulla on ikään kuin kokemusta tarpeeks siitä niinku ongelma-alueesta, että sä voit ylipäänsä päättämään itsesi kanssa sen, että mitä kannattaa mallintaa ja mitä ei kannata mallintaa.

Taulukossa 10 on esitetty haastatteluissa mainitut hyödyt, haasteet ja ongelmat mallipohjaiseen kehittämiseen liittyen.

TAULUKKO 10 Mallipohjaiseen kehittämiseen liittyvät hyödyt ja haasteet

Hyödyt	**Esiintyvyys**	**Haasteet**	**Esiintyvyys**
Ratkaisujen protoilu aikaisessa vaiheessa	2	Ei kiinnosta	3
Koodaajalle mielekkäämpi työ	1	Ei toimi	3
Tehokkuushyödyt (koodi, dokumentointi)	1	Koodaaminen kannattavampaa (helppous, pidempi elinkaari)	3
Uudelleenkäytön hyödyt	1		
Yksinkertaisuus, korkea abstraktiotaso	1	Työkalut eivät tue	3
		Vaatii osaamista	3
		Vanhanaikaista	3
		Ei kustannustehokasta	2
		Vaati sitoutumista	2
		Liian rajoittavaa	1
		Vaatii investointeja	1
		Vaatii notaation tarkkaa noudattamista	1

5.5.3 Muut tavat

Haastatteluaineistossa esiintyi tunnettujen kaaviotekniikoiden, työkalujen, vapaamuotoisten kuvaustapojen ja mallipohjaisen sovelluskehityksen lisäksi kolme muuta mallinnus- ja dokumentointitapaa. Tavat ja niiden esiintyvyys haastatteluissa sekä tarkentavat huomiot löytyvät taulukosta 11.

TAULUKKO 11 Haastatteluissa mainitut muut tavat

Muut tavat	**Esiintyvyys**	**Huomiot**
Swagger/OpenAPI	5	Rajapintojen generointi ja dokumentointi
ArchiMate	1	Kokonais- ja ratkaisuarkkitehtuurin mallintaminen
C4-arkkitehtuurimalli	1	Haastatteluhetkellä ei vielä käytössä

Suuri osa haastateltavista piti tärkeinä tarkkoja rajapintakuvauksia, koska niiden avulla rakennettiin yhteyksiä järjestelmien, laitteiden ja sovellusten välille. Viisi haastateltavaa kertoi, että ohjelmointi- eli API-rajapintojen (engl. application programming interface) dokumentoinnissa oli käytössä rajapintojen kuvauskieli **Swagger/OpenAPI**[24], jonka avulla generoitiin API-dokumentti. Etenkin integraatioprojekteihin keskittyneet haastateltavat painottivat Swagger/OpenAPI:n merkitystä työssään.

Tuotepäällikkö/sovellusarkkitehti H7 kertoi generoituvan rajapintakuvauksen edustavan suosimaansa ketterää dokumentointitapaa. Ohjelmistoarkkitehti H13 painotti haastattelunsa aikana useaan otteeseen mallin ajantasaisuuden ja automaattisen generoitavuuden tärkeyttä. Swaggerin/OpenAPI:n avulla REST-rajapinnasta generoituva kuvaus oli hänen mielestään tällä hetkellä ohjelmistokehityksen tärkein dokumentti, koska se kertoi absoluuttisesti kahden järjestelmän välisen vuorovaikutuksen.

> …se on niinku nykykehityksessä ja webbikehityksessä, integraatiokehityksessä erittäin käytetty tällänen niin kun teknologia tällä hetkellä. Ja, ja onneksi se generoituu automaattisesti ja se on niin kun ehkä järjestelmän tänhetkisesti niin kun tärkein dokumentaatio tai ainakin sinne ulospäin, kolmannelle ulkopuolelle, kuka sitä järjestelmää käyttää, niin tällanen rajapintakuvaus. (ohjelmistoarkkitehti H13)

Ratkaisuarkkitehti H9 käytti kokonais- ja ratkaisuarkkitehtuurin mallintamisessa **ArchiMate**a, joka soveltui hänen mielestään UML:ää paremmin IT:n ja liiketoiminnan väliseen yhteistyöhön.

> …se helpottaa juuri sitä niin kun siinä rajapinnassa toimimista, missä ollaan niin kun tekniset ja bisnesihmiset toimitaan, niin se on myös tärkeä sitten. UML ei ihan taivu siihen sitten, että UML:ssä on vaikeampi kuvata jotenkin tämmösiä prosesseja ja bisnesvaatimuksia ja niitä stakeholdereita, että [ArchiMaten käytön] toivoisin myös yleistyvän.

Ohjelmistoarkkitehti H13 kertoi, että oli kollegoidensa kanssa kiinnostunut **C4-arkkitehtuurimalli**sta[25]. Käytössä malli ei vielä haastatteluajankohtana ollut, mutta suunnitteilla. Ohjelmistoarkkitehti H13 mainitsi C4-mallin etuina ohjelmiston tarkastelun eri näkymissä, eri abstraktiotasoilla ja tekstimuotoisen käsittelyn piirtämisen sijaan. Hän kertoi, että mallin alinta tasoa, joka voidaan toteuttaa UML:n luokkakaaviolla, he eivät olleet suunnitelleet käytettävän ollenkaan. Alin taso olisi jo niin lähellä koodia, että sen ajantasaisena pitäminen olisi turhaa työtä.

[24] https://swagger.io/docs/specification/about/
[25] https://c4model.com/

6 POHDINTA JA JOHTOPÄÄTÖKSET

Tässä pääluvussa vedetään johtopäätökset empiirisen tutkimuksen tuloksista ja verrataan niitä alan kirjallisuudessa esitettyihin näkemyksiin sekä aiempiin käytännön tutkimuksiin. Ensimmäisessä alaluvussa käsitellään mallintamisen roolia analyysissä ja suunnittelussa sekä muissa tietojärjestelmäkehityksen aktiviteeteissa. Toisessa alaluvussa pohditaan käytössä olevien menetelmien ja työkalujen merkitystä sekä niihin kohdistuvia tarpeita. Kolmannessa alaluvussa käsitellään tutkimuksessa esiin tulleita asenteita ja haasteita sekä niiden merkitystä tietojärjestelmäkehitykseen liittyvien ongelmien muodostumisessa. Neljännessä alaluvussa käsitellään tutkimukseen liittyviä rajoitteita, jonka jälkeen pohditaan tulosten merkitystä tutkimukselle sekä esitellään jatkotutkimusaiheita. Luvun lopuksi esitetään, miten tutkimustuloksia voidaan hyödyntää käytännön tietojärjestelmäkehityksessä.

6.1 Mallintamisen rooli tietojärjestelmäkehityksen aktiviteeteissa

Tietojärjestelmien mallintaminen, kuten koko tietojenkäsittelytiede, perustuu kieliin. Kehittyneiden kielten avulla abstraktiotason nosto on mahdollista ja sen seurauksena viestinnän edesauttaminen monimutkaisessa tietojärjestelmien kehittämistyössä. Mallin toimiminen kommunikaatiovälineenä esitetäänkin alan kirjallisuudessa (ks. esim. Avison & Fitzgerald, 2006, Liddle, 2011) sen perustehtäväksi ja myös empiirisissä tutkimuksissa se on havaittu pääsyyksi mallintaa (ks. esim. Davies ym., 2006; Störrle, 2017). Tämän tutkielman empiirinen osuus vahvistaa näitä näkemyksiä. Viestinnän edesauttaminen analyysissä ja suunnittelussa luotujen mallien avulla ulottuu tietojärjestelmäkehityksen eri aktiviteetteihin aina laadukkaan lopputuloksen tuottamiseen asti ja ylläpitoon. Viestintään pohjautuvien muiden hyötyjen ymmärtäminen on tämän tutkimuksen perusteella paljolti kiinni ammattilaisen työtehtävästä ja kokemuksista.

Tämän haastattelututkimuksen tulokset osoittavat, että nykypäivän tietojärjestelmäkehityksessä ammattilaisilla on tarve tehdä mallintamista korkealla abstraktiotasolla. Tätä johtopäätöstä tukevat mallin viestinnällisen roolin painottaminen, kuvailut eri kaaviotekniikoiden käytöstä, sovellusaluekohtainen mallintaminen, kiinnostus C4-mallia kohtaan ilman sen yksityiskohtaisinta tasoa, valmiiden komponenttien ja järjestelmien yhdistely sekä mallintamiseen liittyvät haasteet, kuten ajanpuute ja mallien päivittämisen taakka. Samaan tarpeeseen viitannee Badreddin ym. (2018) raportti tiedon mallintamisen ja sovellusaluekohtaisen mallintamisen lisääntymisestä. Tietomalli on pitkäikäinen, koska tieto pysyy, vaikka toteutus muuttuu (Avison & Fitzgerald, 2006), ja sovellusaluekielten hyödyt perustuvat abstraktiotason nostoon (Kelly & Tolvanen, 2008).

Mallintaminen korkealla abstraktiotasolla liittyy etenkin vaatimusmäärittelyyn ja alustavaan suunnitteluun. Empiiriset tutkimukset (ks. esim. Forward ym., 2010; Gorschek ym., 2014; Petre, 2013) osoittavatkin mallintamista hyödynnettävän lähinnä alkuvaiheessa kehitystyötä ja sen jälkeen mallit voidaan jopa sivuuttaa kokonaan (Badreddin ym., 2018; Petre, 2013). Kun mietitään mallin perustehtävää eli viestintää, on kehitystyön tai projektin saattaminen käyntiin hyvin riippuvainen siitä. Tämän haastattelututkimuksen perusteella malleilla varmistetaan yhteisymmärryksen saavuttamista ja sen jakamista eri sidosryhmien kesken. Ammattilaiset kokevat, että tätä tiedon jakamista tarvitaan eniten alkuvaiheessa kehitystyötä. Tämä ei kuitenkaan tarkoita, että mallintamista käytettäisiin jokaisen projektin tiedonvälityksessä. Ketterässä kehityksessä tulee punnita hyöty suhteessa käytettäviin resursseihin. Mitään turhaa ei tuoteta.

Tämän tutkimuksen perusteella voidaan todeta, että mallintamisen rooli vaihtelee projektista toiseen. Jos toteutettavana on pieni muutostyö tai ominaisuus ja etenkin, jos tiimi on pieni ja koostuu moniosaajista, myös kehitystyön alku- ja toteutusvaiheessa käyttötapauslistaus tai ketterässä ohjelmistokehityksessä yleistyneet käyttäjätarinat (Schön ym., 2017) voivat olla riittäviä. Tutkimuksessa havaittiin Babarin (2009) ja Sommervillen (2016) tavoin, että ketterissä tiimeissä arkkitehtuuri ja sen dokumentointi poikkeavat suunnitteluvetoisesta kehittämisestä (ks. esim. Bass ym., 2003; Garlan, 2000; Pressman, 2005; van Vliet, 2007). Arkkitehtuuri voi rakentua kehitystyön edetessä ja nivoutua toteutukseen. Arkkitehtuuridokumentaatio tehdään korkealla tasolla ja ad hoc -tyylisesti wikeihin tallennettuina, usein lähinnä käyttötapausten tai käyttäjätarinoiden muodossa. (ks. myös Babar, 2009.) Tutkimuksessa löydettiin kuitenkin myös vahvistusta tekstimuotoisille vaatimuksille esitetylle kritiikille monitulkintaisuudesta ja kokonaiskuvan puutteesta (ks. Robeer ym., 2016) sekä uusintatöiden tarpeesta ja riittämättömästä tuesta ylläpidossa (ks. Pressman & Maxim, 2015, s. 148).

Tutkimuksessa ilmeni, että arkkitehtuurin mallintamiseen käytössä oleva aika on erittäin rajallista, mikä usein johtaa hyvin vapaamuotoisten kuvaustapojen käyttöön. Mallintamisen rooli suunnittelussa painottuu siten ongelmanratkaisuun jopa luonnosten muodossa ja kollaboratiiviseen suunnitteluun mo-

niosaajien kesken tarkkojen ohjeellisten mallien tuottamisen sijaan. Myös Malavoltan ym. (2012) ja Ozkayan (2018) tutkimuksissa todettiin arkkitehtuurin vapaamuotoisen mallintamisen olevan yleistä. Malavoltan ym. (2012) ja Ozkayan (2018) tutkimuksissa raportoitua arkkitehtuurin kuvauskielien (AADL, ADL) käyttöä ei tämän tutkimuksen aineistossa esiintynyt, kuten ei myöskään työkalujen automatisoituihin ominaisuuksiin liittyviä tarpeita (ks. Malavolta ym., 2012), todennäköisesti niiden jäykkyyden ja siten ketterään kehitykseen soveltumattomuuden vuoksi.

Mallintamiseen korkealla abstraktiotasolla liittyy olennaisesti tarve tehdä mahdollisimman pitkäikäisiä malleja. Tämän haastattelututkimuksen perusteella liian yksityiskohtaiset, varsinkin koodin abstraktiotasoa lähellä olevat luokkakuvaukset koetaan hyödyttömiksi niiden mukana tulevan ajantasaistamisen taakan vuoksi, joka Fariaksen ym. (2018) ja Forwardin ym. (2010) tavoin todetaan merkittäväksi haasteeksi. Tällaisten mallien paikkansapitävyyteen ei voida enää kehitystyön alkuvaiheen jälkeen luottaa. Ajantasaisia, koodin kanssa yhteneväisiä malleja tarvitaan etenkin ylläpidossa, jossa mallin rooli koettiin mahdollisesti vielä tärkeämmäksi kuin alkuvaiheen viestinnässä. Tältä osin haastattelututkimuksen tulokset näyttävät poikkeavan useista aiemmin toteutetuista empiiristä tutkimuksista (ks. esim. Badreddin ym., 2018; Lange ym., 2006; Nugroho & Chaudron, 2008; Störrle, 2017).

Tarkasteltaessa mallin roolia ylläpidossa on erotettava haastateltavien kokema merkityksellisyys käytännön todellisuudesta. Ammattilaiset kokevat mallit erittäin hyödyllisiksi järjestelmään perehtymisessä ja siten ylläpidossa, mutta aineisto puoltaa yhtä paljon sitä, että malleja ei kuitenkaan tässä tarkoituksessa hyödynnetä tarpeeksi. Tämä myös tukee Langen ym. (2006), Nugrohon ja Chaudronin ym. (2008) ja Störrlen (2017) näkemystä siitä, että mallit sivuutetaan ylläpidossa niiden paikkansapitämättömyyden vuoksi. Mallien hyödyllisyys onkin tämän tutkimuksen perusteella noteerattu enemmänkin ilmenneiden ongelmien kuin niiden yleisen käytön kautta. Ammattilaiset ovat huomanneet järjestelmään perehtymisen vaikeaksi ja aikaa vieväksi ilman laadukasta dokumentaatiota, joka sisältää ajantasaisia, sopivan abstraktiotason malleja. Tämä on merkittävää, koska mahdollisimman hyvällä ylläpidettävyydellä voidaan edesauttaa lähtökohtaisesti kallista ylläpitotyötä (ks. Sommerville, 2016, s. 256) ja järjestelmän laatua (ks. Nugroho & Chaudron, 2008).

Mallintamisen esitetään vaikuttavan työn tuottavuuteen ja järjestelmän laatuun monella tavalla; väärinkäsityksiltä ja virheiltä välttymisestä tehokkaaseen projektin hallintaan ja vähentyneestä testauksesta jouhevaan ylläpitoon (ks. esim. Chaudron ym., 2012; Hunt & Thomas, 2002; Liddle, 2011). Tässä tutkimuksessa löydettiin vahvistusta näille väitteille ja usein näkemykset peilaavat ammattilaisten työtehtäviä ja -kokemusta. Myös vastakkaisia näkemyksiä havaittiin. Mallintaminen voidaan kokea haasteiden, kuten kehnon työkalutuen, vuoksi työn tehoa laskevana. Tämän lisäksi muut tiimiläiset eivät välttämättä edes ymmärrä malleja tarvittavalla tasolla, jolloin käsiteltävät asiat täytyy joka tapauksessa keskustella tai kirjoittaa auki. Kaikki ammattilaiset eivät yhdistä mallien käyttöä laatuun. Osa on sitä mieltä, että enemmän merkitystä on kehit-

täjien ammattitaidolla ja kokemuksella (jotka tosin voivat hyvien käytäntöjen noudattamisella viitata mallintamiseen) sekä testauksella. Mallien vähäistä käyttöä ongelmallisempana moni pitää nykypäivän kehitystyössä yleistä puutteellista testaamista.

Tässä haastattelututkimuksessa yksikään ammattilainen ei ilmaissut suoraan kokevansa, että mallien käyttäminen vähentäisi testaamistarvetta. Epäsuorasti tähän tosin viitattiin virheiden aikaisella havaitsemisella ja ylimääräisen työn välttämisellä. Kaiken kaikkiaan testauksen osalta mallien rooli jäi epäselväksi, todennäköisesti koska tutkimuksen haastateltaviin ei sisältynyt yhtään päätoimista testaajaa. Nugrohon ja Chaudronin (2008) tapaan todetaan, että testausprosessissa todennäköisesti hyödynnetään lähinnä tekstimuotoista dokumentaatiota. Tämä voi johtua joko siitä, että testaajat eivät malleja ymmärrä tai siitä, että dokumentaatioon ei sisälly malleja tai niiden paikkansapitävyyteen ja siten hyödyllisyyteen ei voida luottaa.

Grossmanin ym. (2005) ja Fernández-Sáezin ym. (2015) tutkimustulosten mukaan mallintamista hyödyntävät ovat usein korkeasti koulutettuja ja kokeneita, kun taas Daviesin ym. (2005) ja Gorschekin ym. (2014) mukaan kokeneimmat ammattilaiset hyödyntävät mallintamista vähiten. Tässä tutkimuksessa löydettiin vahvistusta molemmille tuloksille. Osa ammattilaisista kokee, että kokemuksen myötä mallintamistarve vähenee. Tämä johtunee siitä, että mallintaminen käsitetään oman ajattelun ja ongelmanratkaisun tukena viestinnän sijaan. Kun mallintamista ajatellaan kommunikaatiovälineenä, ei lisääntyneellä kokemuksella tulisi olla vaikutusta. Mallintamisen hyödyntämiseen viestinnässä vaikuttaakin kokemuksen lisäksi muut tekijät, kuten tiimin koko ja osaaminen, ja projektin sekä kehitettävän ratkaisun laajuus ja monimutkaisuus.

Mallintamisen väheneminen kokemuksen lisääntyessä voi selittyä myös urakehityksellä teknisestä suunnittelusta asiakaskommunikaatioon, kuten tässä tutkimuksessa havaittiin. Tähän viittaavat myös Daviesin ym. (2005) tulokset, joissa kokeneimmat ammattilaiset käyttävät vähiten mallinnustyökaluja ja keskittyvät liiketoimintaprosessien mallintamiseen. Tämä voi siis tarkoittaa, että mallintamisen määrä ei vähene, mutta käytössä on tekniseen suunnitteluun verrattuna suppeampi määrä tekniikoita ja vapaamuotoisempia työkaluja. Tulevissa tutkimuksissa tulee siis selventää, mitä mallintamisella tarkoitetaan (onko mallintamista myös paperille tehty luonnos vai ainoastaan työkalulla tuotetut dokumentit) sekä verrata kokemusvuosien ja mallintamisen hyödyntämistä tarkkoihin työtehtäviin ja mallintamisen käyttökohteisiin.

6.2 Erilaisten mallintamismenetelmien ja -työkalujen merkitys kehitystyössä

Empiirissä tutkimuksissa, joissa on kartoitettu eri mallintamismenetelmien käyttöä, UML on ollut aina käytetyimpien joukossa ja suurimmaksi osaksi yleisin menetelmä, mikä vastaa sen standardiasemaa alalla. Kun tarkastellaan lu-

vussa 3.2 esitettyjä tutkimuksia, on niissä havaittavissa UML:n käytön yleistyminen vuosien varrella (vrt. esim. Davies ym., 2006 ja Störrle, 2017) ja prosessiohjeita sisältävän menetelmällisyyden korvaaminen vapaamuotoisilla toimintatavoilla (vrt. esim. Davies ym., 2006 ja Ozkaya, 2018). UML:n käytön on raportoitu olevan vapaamuotoista Grossmanin ym. (2005) tutkimuksesta Petreen (2013). Tuoreimmissa tutkimuksissa ei käsitellä UML:n käytön formaalisuusastetta, mutta esimerkiksi Badreddinin ym. (2018) kyselyn avoimet vastaukset viittaavat vapaamuotoiseen käyttöön, kuten myös Fariaksen ym. (2018) päätelmät. Tämän empiirisen tutkimuksen aineisto vahvistaa aiempien tutkimuksen tuloksia: UML on suosituin mallintamismenetelmä ja sen käyttö on vapaamuotoista. Täysin säännönmukaista UML:n käyttöä ei raportoitu ollenkaan.

Fowler (2003) esittää, että UML:n käyttöä voidaan kolmen tavan (luonnostelu, toimintasuunnitelma, ohjelmointikieli) lisäksi tarkastella kaavio- ja metamallinäkökulmasta. Kaavionäkökulmassa UML:ssä keskeistä on nimenomaisesti kaaviot. Metamallinäkökulmassa kaaviot ovat toissijaisia, ne ovat esityksiä UML:n ytimestä eli metamallista. (Fowler, 2003.) Näillä näkemyseroilla on yhteys tässä tutkimuksessa esitettyyn jakoon UML:n vapaamuotoisesta käytöstä. Kaavionäkökulma käsittää UML:n sen erilaisten kaaviotyyppien kautta, joita voidaan hyödyntää eri tilanteissa. UML-näkökulma nojaa metamallinäkökulman tapaisesti UML:ään kokonaisvaltaisesti (rakenne ja käyttäytyminen), vaikka sitä ei hyödynnettäisi kaikilta osin. UML-näkökulmassa mallintaminen ei ole ”piirtelyä” eikä ainoastaan keskustelun apuna, vaan olennainen osa kehitystyötä. Käyttötapaan aineiston perusteella näyttää vaikuttavan ammattilaisen koulutustausta, organisaation toimintatavat sekä asiakkaan vaatimukset.

Tässä tutkimuksessa UML-näkökulmaa ei nimetty Fowlerin (2003) mukaisesti metamallinäkökulmaksi, vaikka näkökulmat muistuttavatkin toisiaan. Eriävään nimeämiskäytäntöön on kaksi syytä. Fowlerin (2003) metamallinäkökulma (kuten myös käyttötavat toimintasuunnitelma ja ohjelmointikieli) viittaa UML:n kokonaisvaltaiseen, säännönmukaiseen käyttöön, mitä ei tämän tutkimuksen aineistossa tullut esille. Näkökulma myös edellyttäisi, että UML:n käyttäjät olisivat tietoisia sen metamallista ja hyödyntäisivät siihen perustuvia mallinnustyökaluja. Näin ei kuitenkaan ole, sillä suurin osa UML-näkökulmaa hyödyntävistä ammattilaisista ei osoita kielen tuntemusta metamallitasolla eikä käytä varsinaisia mallinnustyökaluja. Käytössä ovat lähes yksinomaan piirtotyökalut, jotka eivät ohjaa UML:n oikeaoppiseen, sen metamalliin perustuvaan käyttöön.

Ero kaavio- ja metamallinäkökulmien välillä voi olla merkittävä myös aiempia empiirisiä tutkimuksia tarkasteltaessa. Esimerkiksi Petre (2013) vaikuttaa keskittyvän lähinnä metamalli- tai UML-näkökulmaan pohjautuvan (tai kuten Störrle (2017) toteaa, hyvin määrämuotoisen) UML:n käytön kartoittamiseen tai ainakaan hän ei selvennä raportissaan, miten hän tai haastateltavat ymmärtävät UML:n käytön. Suurin osa Petren (2013) tutkimukseen osallistuneista ei raportoinut UML:n käyttöä. On mahdollista, että ainakin osa heistä oletti UML:n käytön edellyttävän sen kokonaisvaltaista käyttöä. Sama voi koskea muitakin tutkimuksia, joissa UML:n käyttöä ei ole määritelty. Yhdelle am-

mattilaiselle sekvenssikaavion käyttö silloin tällöin voi olla UML:n käyttöä (kaavionäkökulma), toiselle ei (metamallinäkökulma).

Aiempiin tutkimuksiin on voinut osallistua samanlaisia ammattilaisia kuin tähän haastattelututkimukseen: osa ei ennen haastattelua ollut varma, että käyttikö UML:ään sisältyviä kaaviotekniikoita. Esimerkiksi Petren (2013) kuvailu tutkimuksensa menetelmästä ei sulje tätä mahdollisuutta pois. Lisäksi hänen raportissaan esitetyt sitaatit niiden ammattilaisten osalta, jotka eivät UML:ää käyttäneet vihjaavat UML:n käytön käsittämiseen hyvin määrämuotoisella tavalla, metamallinäkökulmasta tai vähintäänkin UML-näkökulmasta. Samaan tapaan Gorschekin ym. (2014) tutkimuksessa suurin osa malleja käyttävistä vastaajista kertoi, ettei käytä formaalia UML:ää, vaan sen tapaisia kaavioita. On siis tulkinnanvaraista, lasketaanko käyttö UML:n käytöksi. Tämän tutkimuksen aineiston perusteella voidaan sanoa, että Suomessa toimivat ammattilaiset käsittävät UML:n käytön enemmän kaavionäkökulman mukaisesti. Toisaalta ne ammattilaiset, jotka kieltäytyivät tutkimukseen osallistumisesta, voivat hyvinkin käsittää UML:n metamalli- tai UML-näkökulmasta.

UML:n kaaviotekniikat ovat olleet olemassa ja käytössä jo kauan ennen niiden yhdistämistä UML:ksi. On siis kohtuullista miettiä, voiko yksittäisen kaaviotekniikan käyttöä laskea laisinkaan UML:n käytöksi, etenkin jos käyttäjä ei edes tiedä sen lukeutuvan UML:ään. Bran Selic painottaa vastauksessaan Petren (2013) tutkimukselle (ks. Petre, 2014), että UML on tarkoitettu käytettäväksi valikoivasti, tarpeen mukaan. Myös Fowler (2003) esittää vapaamuotoisen käytön (luonnostelu) olevan yleisin UML:n käyttötapa. Kun lisäksi huomioidaan, että empiiriset tutkimukset mallintamiskäytännöistä tukevat näitä näkemyksiä, voidaan todeta, että on tarkoituksenmukaista laskea myös vapaamuotoinen, yksittäistenkin kaaviotyyppien käyttö UML:n käytöksi. Ilman UML:n notaatioiden yhdistämistä, ohjelmistostandardiksi julistamista ja sisältymistä koulutusohjelmiin, näitä yksittäisiä kaaviotekniikoita tuskin tunnettaisiin tai käytettäisiin näinkään laajasti kuin tällä hetkellä tehdään (ks. myös Reggio ym., 2014).

Tarkasteltaessa empiirisiä tutkimuksia UML:n käytöstä huomataan, että kaikissa luokkakaavio on ollut kolmen käytetyimmän kaaviotyypin joukossa, ja loput kolmen kärjessä ovat vaihdelleet aktiviteetti-, käyttötapaus- ja sekvenssikaavioiden välillä. Tulokset ovat samansuuntaisia myös tämän tutkielman yhteydessä toteutetun haastattelututkimuksen kanssa, luokkakaavion rooli tosin ei ole niin korostunut kuin muissa. Aineisto sisälsi paljon ainoastaan käyttäytymistä mallintavia ammattilaisia. Tämän lisäksi aineisto kuvastaa mallintamisen keskittymistä yhä enemmän integraatioihin järjestelmien laajentuessa ja monimutkaistuessa, ja valmiiden komponenttien käyttöä, jolloin luokkatason kaavioita ei tuoteta. Luokkakaaviota hyödynnetään kuitenkin ER-kaavion tavoin tiedon mallintamisessa, korkealla tasolla.

Scanniellon ym. (2010) tutkimusta voidaan pitää UML-tutkimusten osalta poikkeuksena tilakaavion käytön yleisyyden vuoksi. Scanniello ym. (2010) eivät avaa mahdollisia syitä käytölle ja sekä raportti että otos ovat melko suppeita. Tuloksia tarkasteltaessa kuitenkin huomataan, että Scanniellon ym. (2010) yrityksistä vain vajaa neljännes käytti ketteriä kehitysmenetelmiä, vesiputousmal-

lin ollessa yhtä yleinen. Selkeästi yleisin kehitysmenetelmä oli Unified Process, mikä voi viitata vastaajien projektien olevan monimutkaisia, prosessivaiheita vaativia ja hyödyntävän UML:ää paljon kokonaisvaltaisemmin kuin muissa tutkimuksissa. Tämä tukee myös Fitsilisin ym. (2013) näkemystä siitä, että tilakaavioita käytetään suurissa ja monimutkaisissa projekteissa.

Scanniellon ym. (2010) tutkimukseen liittyy projektien merkitsevyyden lisäksi myös toinen mahdollinen selittävä tekijä tilakaavion yleisyydelle. Scanniellon ym. (2010) tutkimus suunnattiin ohjelmistotalojen lisäksi yrityksille, joissa kehitetty ohjelmisto oli osa heidän tarjoamaansa tuotetta tai palvelua, mikä viittaa sulautettuja järjestelmiä toimittaviin yrityksiin. Tilakaavioita hyödynnetään yleisesti sulautettujen järjestelmien kehitystyössä (Pilone & Pitman, 2005, s. 7). Tämän haastattelututkimuksen aineistossa esiintyi myös sulautettujen järjestelmien kehittämistä sisältäviä projekteja, mutta niiden osalta tilakaavion käyttöä ei kuitenkaan tutkimushetkellä raportoitu. Tilakaavion todettiin kuitenkin olevan käytössä, joskin harvoin, energia-alalle suunnattujen tietojärjestelmien yhteydessä. Todennäköisesti laajemmassa otannassa tilakaavion käytöstä löytyisi lisää todistusaineistoa useammissa yhteyksissä.

Suomalaisessa tietojärjestelmäkehityksessä on tämän tutkimuksen perusteella käytössä jopa harvinaisemmat ja UML:ään viimeisimmissä versioissa lisätyt kaaviotyypit kuten ajoitus- ja pakkauskaaviot. Ero on merkittävä esimerkiksi Petren (2013) tutkimukseen, jossa 50 haastateltavaa käytti yhteensä ainoastaan viittä kaaviotyyppiä UML:stä. Kaksi kaaviotyyppiä, joiden käyttöä ei tässäkään tutkimuksessa raportoitu, ovat profiilikaavio ja kokoava vuorovaikutuskaavio. Syyt käyttämättömyydelle on melko suoraviivaisia päätellä. Profiilikaavio mahdollistaa sovellusaluekohtaiset laajennokset, eikä haastatteluaineiston perusteella UML:ää hyödynnetä tällä tavalla. Sovellusaluekohtaiset, paljon UML:ää yksinkertaisemmat mallinnuskielet koetaan hyödyllisemmiksi tässä tarkoituksessa. Kokoava vuorovaikutuskaavio koostuu nimensä mukaisesti useasta eri kaaviosta, joten sen käyttö voidaan kokea tarpeettomaksi. Todennäköisesti osa ammattilaisista ei ole edes tietoisia näiden kahden UML-kaavion käyttötarkoituksista tai mahdollisesti niiden olemassaolostakaan.

Ehkä hieman yllättäen käyttötapauskaavio nousee tämän tutkimuksen ristiriitaisimmaksi kaaviotekniikaksi. Käyttötapauskaaviota pidetään melko yksinkertaisena koko järjestelmän toiminnallisena mallina (ks. esim. Koskimies ym., 2004) ja sen notaatiota ymmärrettävänä ja siten viestintää edesauttavana eri sidosryhmien kesken (ks. esim. Gemino & Parker, 2009; Haikala & Mikkonen, 2011). Näille väitteille löytyikin tästä haastattelututkimuksesta vahvistusta, ammattilaisista osa pitää käyttötapauskaaviota ilmaisuvoimaisena ja hyödyllisenä osana UML:ää. Tutkimuksessa nousi esiin myös vastakkaisia näkemyksiä. Yhden haastateltavan mielestä käyttötapauskaavio ei ollut tarpeeksi ilmaisuvoimainen. Toisen mielestä kaaviotyyppi ei ollut tarpeeksi hyödyllinen, koska sen paikkansapitävyyteen ei voinut luottaa. Kolmas kertoi, että kaavion käyttö koettiin ajoittain vaivalloiseksi eikä sitä alalla juurikaan hyödynnetty.

Mistä näin kaksijakoiset mielipiteet johtuvat? Käyttötapauskaavioon näyttää kiteytyvän UML:ään liittyvä ristiriitaisuus ja polarisoituneet näkemykset,

joita aiemmissa empiirisissä tutkimuksissa on tullut esille (ks. esim. Grossman ym., 2005; Hutchinson ym., 2011b). Alan kirjallisuudessa on myös nostettu esille nimenomaan käyttötapauskaavion kiistanalaisuus (ks. Siau & Loo, 2006). Siau ja Loo (2006) selittävät, että varsinkaan kokemattomat mallintajat eivät ymmärrä notaation yksityiskohtia, kuten mahdollisesti harhaanjohtavaa laajennosta (engl. extend). Haikala ja Mikkonen (2011) noteeraavat tämän saman ongelman ja ohjeistavatkin mallintajia panostamaan viestinnän edesauttamiseen notaation yksityiskohtaisen käytön sijaan. Myös Lilly (2000) huomauttaa, että käyttötapauskaaviot tulisi pitää mahdollisimman yksinkertaisina.

Kirjallisuudessa esitetyistä ongelmista ja ohjeistuksista sekä tutkimuksessa havaitusta korkean abstraktiotason suosimisesta huolimatta käyttötapauskaavioita tuotetaan mitä ilmeisimmin usealla eri abstraktiotasolla ja niihin liitetään myös yksityiskohtaisia tietoja sekä suuria määriä käyttötapauksia. Nämä toimintatavat myös mahdollisesti hankaloittavat käyttötapauskaavioiden ymmärtämistä ja selittävät niiden paikkansapitämättömyyttä myöhemmissä vaiheissa kehitystyötä. On helppo myös päätellä, miksi kaaviotyyppiä ei välttämättä mielellään käytetä, kun ketterässä kehityksessä on pitäydyttävä ainoastaan välttämättömimmissä tuotoksissa. Notaation yksityiskohdat voivat oletetusta yksinkertaisuudestaan huolimatta hämmentää, eikä käyttötapauskaavio yleiskuvana välttämättä tarjoa lisäarvoa käyttötapausten ja muiden kaavioiden lisäksi, kuten myös Dobing ja Parsons (2008) toteavat. Siten sen käyttö ei ole hyödyllinen kehitystiimin jäsenten kesken, kun keskitytään yksittäisten toiminnallisuuksien toteuttamiseen (ks. Dobing & Parsons, 2008).

Käyttötapauskaavion lisäksi fyysistä arkkitehtuuria kuvaavat komponentti- ja sijoittelukaaviot jakoivat haastateltavien mielipiteitä. Merkittävää on, että suurin osa tutkimukseen osallistuneista arkkitehdeistä ei käyttänyt kumpaakaan tekniikkaa, vaan hyödynsivät lähinnä vapaamuotoisempia tapoja. Kun verrataan luvun 3.1.2 sijoittelukaaviota ja luvun 5.5.1 vapaamuotoisia arkkitehtuurikuvia, on selvää, että jälkimmäiset ovat ilmaisuvoimaisempia ja välittömästi helpommin ymmärrettävissä. UML:ssä ei ole näin ilmaisuvoimaisia symboleja, vaan kaikki osat tietokoneista pilvipalvelun palvelimiin esitetään samantapaisilla laatikon muotoisilla solmuilla (ks. OMG, 2017, s. 657—659). Jos arkkitehti käyttää työkalunaan esimerkiksi Visiota ja tekee valinnan sen ehdottamista erilaisista muodoista ja kaaviopohjista, ovat intuitiiviset vaihtoehdot todennäköisesti UML:n notaatiota houkuttelevampia ja myös soveltuvampia useampaan käyttökohteeseen.

Tämän tutkimuksen perusteella verkkoarkkitehtuuriin, -infrastruktuuriin tai ”järjestelmän karttaan” ei ole standardia kuvaustapaa eikä myöskään UML sisällä tähän tarkoitukseen erillistä kaaviotyyppiä. Petren (2013) tutkimuksessa esitettiin, että UML ei sovellu koko järjestelmän kuvaamiseen. Sijoittelukaaviota voidaan kuitenkin hyödyntää laajennosmekanismeja käyttäen, mutta kuten tässä tutkimuksessa tuli esille, tapa voi osoittautua liian työlääksi hyötyihin nähden. Ammattilaisten mielestä ”järjestelmän kartta” on toteutustavasta riippumatta hyödyllinen dokumentti, etenkin ylläpidossa (ks. myös Hunt & Thomas, 2002) ja lähes jokainen mainitsikin haastattelussaan jonkinlaisen version siitä.

Kuviossa 23 esitetään tässä tutkimuksessa esiin tulleet ja tärkeiksi havaitut mallinnus- ja dokumentointitekniikat sekä niiden keskeisimmät käyttötarkoitukset.

KUVIO 23 Käytössä olevat keskeiset mallinnus- ja dokumentointitekniikat sekä niiden käyttökohteet

Usean empiirisen tutkimuksen yhteydessä on nostettu esiin mallintamiseen mahdollisesti liittyvät alueelliset ja kulttuuriset erot. Grossman ym. (2005) huomauttavat eurooppalaisten hyödyntäneen olioparadigmaa yhdysvaltalaisia kauemmin, mikä voi osaltaan selittää eroja UML:ään liittyen ja myös Störrlen (2017) toteamusta eurooppalaisten ja etenkin saksalaisten mallintamiskeskeisyydestä yhdysvaltalaisiin verrattuna. Samaan tapaan Fettke (2009) esittää saksalaisten olevan erilaisten mallinnustekniikoiden käytössä kokeneempia kuin australialaiset (ks. Davies ym., 2006). Kun tarkastelussa huomioidaan myös italialaiset tutkimukset (ks. Scanniello ym., 2010; Torchiano ym., 2011) ja Forwardin ym. (2010) sekä Badreddinin ym. (2018) tulokset yhdysvaltalaisten vähäisemmästä työkalujen hyödyntämisestä, vaikuttavat eurooppalaiset suhtautuvan verrattain myönteisesti mallintamiseen ja työkalujen sekä formaalimpien tapojen käyttöön.

Tämän haastattelututkimuksen perusteella Suomessa toimivat ammattilaiset arvostavat vapaamuotoisen viestinnän lisäksi laadukasta dokumentaatiota malleineen, ovat harjaantuneita erilaisissa työkaluissa ja harvinaisemmissakin tekniikoissa sekä mallipohjaisuudessa, ja kaipaavat jopa hieman tarkempia menettelytapoja ketteryyden sijaan. Tulokset viittaavat yhtäläisyyksiin muiden eurooppalaisten, kuten saksalaisten ja italialaisten ammattilaisten kanssa. Haastatteluaineisto viittaa toisaalta myös siihen, että tutkimukseen osallistuneet ovat mallintamisessa verrattain perehtyneitä ja siitä kiinnostuneita, joten laajempi

otanta voi osoittaa mallintamisen hyödyntämisen ja arvostamisen paljon vähäisemmäksi.

Työkaluilla on suuri vaikutus mallintamistyöhön ja sen tuloksena syntyvään malliin. Tämän tutkimuksen perusteella ammattilaiset käyttävät lähinnä piirtotyökaluja. Mallintaja ei välttämättä etukäteen päätä, mitä notaatiota käyttää, vaan valitsee työkalun tarjoamia muotoja tai kaaviopohjia tarpeen mukaan ja muokkaa niistä haluamansa näköisiä kuvia, todennäköisesti erilaisia notaatioita yhdistelemällä. Piirtotyökalut eivät ohjaa mallintajaa varsinaisten mallinnustyökalujen tapaan oikeaoppiseen, säännönmukaiseen mallintamiseen, joten monitulkintaisen lopputuloksen lisäksi on mahdollista tehdä suunnitteluvirheitä. Piirtotyökalut, kuten Visio, tai muut ohjelmistot, kuten PowerPoint, tarjoavat kuitenkin saavutettavuuden ja helppokäyttöisyyden lisäksi ilmaisuvoimaisia, mallintajaa ja oletettavasti myös vastapuolta visuaalisesti miellyttäviä kuvausvaihtoehtoja. Tällainen vapaamuotoinen mallintaminen voi entisestään lisätä mallintamiseen liittyvää mystiikkaa, sen yhdistämistä taiteeseen tieteen sijaan (ks. esim. Moody, 2005), kun toimintaan ei liity yleisesti sovittuja standardeja eikä mallien laatua voida arvioida niiden mukaisesti.

Vapaamuotoinen mallintaminen on terminä monitulkintainen. Kirjallisuudessa se käsitetään usein täysin vapaiksi piirroksiksi tai tekstiksi (ks. Leppänen, 2005; Wijers, 1991). Tunnetussa Fowlerin (2003) määritelmässä vapaamuotoista UML:ää eli luonnostelua ei käytetä varsinaisessa suunnittelussa ja oheistuksena, vaan sitä vastaava tapa (toimintasuunnitelma) vaatisi jo täysin säännönmukaista UML:n noudattamista. Ozkaya (2018) taas määrittää UML:n puoliformaalin mallinnuskielen sijaan vapaamuotoiseksi. Selvää on ainoastaan se, että vapaamuotoisuuden(kin) suhteen mallintamisen määritelmät ovat kirjavia. Tämän tutkimuksen mukaan vapaamuotoista mallintamista hyödynnetään valkotaulupiirroksina, verkkoinfrastruktuurikuvina ja sekalaisina kaavioina, jotka mahdollisesti muistuttavat esimerkiksi UML:ää tai sitten eivät. Kuvia voidaan hyödyntää niin keskusteluissa, suunnittelussa kuin myös virallisessa dokumentaatiossa.

Tämän tutkimuksen perusteella alalla suoritettava mallintaminen voidaan Störrlen (2017) tapaan jakaa vapaamuotoiseen, puoliformaaliin ja määrämuotoiseen mallintamiseen, mikä vastaa myös Fowlerin (2003) määritelmää UML:n käytöstä alalla. Karkeasti jaoteltuna vapaamuotoinen mallintaminen on viestintää varten, puoliformaali varsinaista suunnittelua ja dokumentointia varten ja määrämuotoinen yksityiskohtaisen mallintamisen ansiosta koodigenerointia varten (ks. Fowler, 2003; Störrle, 2017). Tämän tutkimuksen haastatteluaineisto kuitenkin osoittaa, että vapaamuotoista mallintamista käytetään puoliformaalin mallintamisen tapaan myös varsinaiseen suunnitteluun ja viralliseen dokumentaatioon, puoliformaalia UML:ää toteutetaan vapaamuotoisesti, ja yksityiskohtaisten mallien sijaan korkean abstraktiotason sovellusaluekielellä generoidaan koodia.

6.3 Mallintamiseen liittyvien asenteiden ja haasteiden vaikutus

Grossmanin ym. (2005), Petren (2013) ja Ozkayan (2018) tutkimuksissa havaittua UML:n monitulkintaisuutta ei tässä haastattelututkimuksessa esitetty merkittävänä haasteena. Tämä johtuu todennäköisesti siitä, että UML on joka tapauksessa yleisesti hyödynnettyä vapaamuotoista mallintamista formaalimpaa ja siitä, että kieltä ei mahdollisesti tunneta tarpeeksi hyvin. Tässä haastatteluaineistossa UML:ää kritisoivatkin lähinnä kieleen perehtyneet ammattilaiset, esittäen sen soveltuvan heikosti pilvi-infrastruktuurin ja liiketoimintavaatimusten kuvaamiseen (ks. myös Petre, 2013) sekä mallipohjaiseen hyödyntämiseen (ks. myös Whittle ym., 2013). UML:n käytön sivuuttamiseen johtaakin usein muut syyt kuin kielen puutteet, joko mallintamista ei koeta tarpeelliseksi tai siihen ei ole riittävästi resursseja.

Tämän tutkimuksen perusteella ammattilaiset yhdistävät mallien vähäisen käytön laatu- ja tuottavuusongelmiin, mutta yhtä lailla tulokset viittaavat siihen, että tätä yhteyttä on vaikea osoittaa. Perinteisen mallintamisen kannalta onkin ongelmallista, että sen hyödyt ja siten myös sen puutteesta johtuvat ongelmat ovat suurimmaksi osaksi havaittavissa vasta pitkällä aikavälillä. Tutkimus osoittaa myös Grossmanin ym. (2005) tavoin, että kehitystyötä tekevät eivät useinkaan ole tietoisia kokonaiskuvasta. Tällaisessa tilanteessa on ymmärrettävää, että mallintamisen hyödyt ja siihen liittyvät tarpeet ovat välittömän alkuvaiheen viestinnän lisäksi vaikeita sisäistää. Mallipohjaisesta kehittämisestä poiketen perinteinen mallintaminen ei tarjoa välittömiä hyötyjä, joten se on useiden haasteiden vuoksi helppo sivuuttaa, etenkin jos ei ole vastuussa projektin onnistumisesta eikä asiakas vaadi mallien tuottamista.

Forward ym. (2010) ja Badreddin ym. (2018) ovat kartoittaneet koodi- ja mallikeskeisen työskentelyn eroja ohjelmistokehityksessä. He toteavat, että osa ammattilaisista on koodikeskeisiä ja asenteellisia mallintamista kohtaan. Samaan tapaan Hutchinson ym. (2011) esittävät, että koodikeskeiset ammattilaiset voivat osoittautua haasteellisiksi mallipohjaisuuden onnistuneessa käyttöönotossa. Myös tässä tutkimuksessa havaittiin koodikeskeisiä asenteita, jotka voivat vaikuttaa mallintamiseen liittyviin haasteisiin. Koodikeskeiset ammattilaiset eivät ole kiinnostuneita mallikeskeisistä ratkaisutavoista, selkeimpänä esimerkkinä mallipohjaisuus ja koodigenerointi mallien pohjalta. Koodikeskeisissä asenteissa mallintaminen on ”piirtelyä”, josta on hyötyä lähinnä kokemattomille tekijöille ja huippuammattilaisten tulisi selviytyä työstään ilman. Tämä on havaittavissa myös tiimiin liittyvissä haasteissa, joissa painotetaan tekemistä suunnittelun sijaan ja nimenomaan selviydytään ilman malleja. Toimittaja- ja asiakasorganisaatiot keskittyvät lopputulokseen eli koodiin, jolloin mallintamiseen ei panosteta eikä sitä hyväksytä kustannuksissa.

Mallintamisen on esitetty vaikuttavan positiivisesti työn tuottavuuteen ja lopputuloksen laatuun (ks. esim. Chaudron ym., 2012; Endres & Rombach, 2003; Selic, 2012). Tämän tutkimuksen perusteella todetaan, että mallien vähäinen käyttö voi puolestaan vaikuttaa negatiivisesti tuottavuuteen ja laatuun. Kuvios-

sa 24 esitetään tulosten perusteella, miten koodikeskeisten asenteiden kautta haasteet voivat johtaa joko monitulkintaiseen vapaamuotoiseen mallintamiseen tai mallien käytön sivuuttamiseen kokonaan, mikä lopulta voi johtaa tuottavuuden ja laatutason laskuun.

KUVIO 24 Koodikeskeiset asenteet voivat vaikuttaa haasteiden ja ongelmien syntyyn, mikä voi johtaa tuottavuuden ja laatutason laskuun

Tuloksissa kerrottiin ammattilaisten näkemyksiä ja kokemuksia mallien vähäisestä käytöstä johtuvista ongelmista, joita ovat mm. ylläpidon ja perehtymisen vaikeutuminen, väärinkäsitykset ja työn hidastuminen. Kuviossa 24 nämä ongelmat on kiteytetty kommunikointiongelmiksi. Toteutuessaan nämä ongelmat johtavat väistämättä tuottavuuden ja laatutason laskuun. Vaikka kuviossa asenteet johtavat haasteisiin ja lopulta ongelmiin, on selvää, että myös haasteet ja niistä johtuva mallien vähäinen käyttö voivat puolestaan lisätä tai pitää yllä koodikeskeisiä asenteita.

Tässä empiirisessä tutkimuksessa havaittiin koodikeskeisyyden vastapainoksi asenteita, joita voidaan kutsua mallikeskeisiksi. Asenteissa mallintaminen yhdistetään osaamiseen ja ammattitaitoon sekä kykyyn hahmottaa kokonaiskuva ja ymmärtää toimien pitkän aikavälin vaikutukset. Koodikeskeisten asenteiden tapaan myös mallikeskeisyydessä mallintaminen voidaan yhdistää taiteeseen (ks. myös Mendling ym., 2010; Moody, 2005), mutta ”piirtelyn” sijaan se nähdään vaikeasti saavutettavissa olevana taitona, josta ollaan ylpeitä. Mallikeskeisyydessä huippuammattilainen voikin siis olla se, joka tekee visuaalisesti hienoja kaavioita. Onnistuneen viestinnän ja yhteistyön kannalta olisi kuitenkin hyödyllistä häivyttää tätä yhteyttä taiteeseen ja tehdä mallintamisesta helpommin saavutettavaa sekä koodi- että mallikeskeisille ammattilaisille.

Aiemmat empiiriset tutkimukset esittävät ratkaisuiksi koulutusta niin mallintajille kuin asiakkaillekin (Dobing & Parsons, 2006; Hutchinson ym., 2011b; Tomassetti ym., 2012), toimintaohjeiden määrittelyä (Dobing & Parsons, 2006; Lange ym., 2006), muutosjohtamisen tapaan asiaa eteenpäin vieviä ”taistelijoita” (engl. champion) (Hutchinson ym., 2011b) ja etenkin työkalukehitystä (Farias ym., 2018; Hutchinson ym., 2011b; Lange ym., 2006; Tomassetti ym., 2012). Tämän tutkimuksen perusteella voidaan sanoa, että kaikkia näitä toimenpiteitä mahdollisesti tarvitaan, jos mallintamiseen liittyviä haasteita ja esteitä halutaan poistaa. Toki realiteetit tulee ottaa huomioon, esimerkiksi Dobingin ja Parsonsin (2006) ehdotusta UML-koulutuksesta asiakasorganisaatioille tuskin voi kannattaa nykypäivänä, jos edes toimittajapuolella ei panosteta tähän. Tästä huolimatta asiakkaiden tulisi lisätä tietämystään tietojärjestelmiin liittyvistä laatutekijöistä ja hyvistä käytännöistä sekä keskittyä elinkaariajatteluun pelk-

kien lyhyen tähtäimen hyötyjen tavoittelun sijaan, jos tavoitteena on muutakin kuin vain mahdollisimman nopea ja edullinen toimitus.

Tomassettin ym. (2012) tutkimuksessa asiantuntijat näkivät mallipohjaisuuden yleistymisen esteenä alalla vallitsevan suosion puutteen, jonka vuoksi ammattilaisille ei kerry tarvittavaa kokemusta. Tämän tutkimuksen perusteella sama noidankehä voi koskea myös perinteistä mallintamista. Osa ammattilaisista on sitä mieltä, että opittuja asioita ei haasteiden vuoksi pääse soveltamaan käytäntöön, mikä johtaa kiinnostuksen ja taitojen hiipumisen myötä käytön sivuuttamiseen. Herää myös kysymys: miksi UML:ää opiskellaan, jos se ei vastaa käytäntöä? (ks. myös Petre, 2014). Tämän tutkimuksen perusteella ammattilaiset käyttävät hyvin pitkälle niitä tekniikoita, jotka opinnoista ovat heille jääneet mieleen, mahdollisesti koska osaamisen päivittämiseen ei ole aikaa tai kiinnostusta tai yksinkertaisesti koska opitut tekniikat toimivat. Mallintamisen ja UML:n säilyttäminen koulutusohjelmissa vaikuttaa siis perustellulta, vaikka se ei vastaisikaan koodikeskeistä todellisuutta. Mitä enemmän nojataan ketteriin toimintatapoihin määriteltyjen menetelmien ja prosessien sijaan, sitä enemmän kaivataan ammattilaisilta kykyä punnita erilaisten hyväksi havaittujen insinööritieteen ja -taiteen tekniikoiden soveltuvuutta muuttuviin tilanteisiin. Myös mallintamisen sivuuttaminen edellyttää perusteluja, joiden tulisi perustua ammattitaitoon.

Mallintamisen monimutkaisuudesta kertoo se, että sitä ei ole helppo oppia ja toisaalta se, että kaikki eivät hyödy korkeammalla abstraktiotasolla toimimisesta. Hutchinson ym. (2011a; 2011b) huomauttavatkin, että korkeammasta abstraktiotasosta voi toisinaan olla jopa haittaa, koska abstrakti ajattelu ei sovi kaikille ja ihmiset usein suosivat konkreettisten esimerkkien käyttöä abstraktien käsitteiden sijaan. Kästner ym. (2018) huomioivat saman asian todetessaan, että UML:n käyttö ja siten myös mallintaminen perustuvat paljolti luokkiin ja luokkakaavioon. Jos abstrakti ajattelu ei luonnistu, voisi ratkaisua hakea käsitteellisen tason sijaan ilmentymien kautta. Kästner ym. (2018) esittävätkin ”oliot ennen luokkia” -ajattelun, jossa oliokaaviolla voidaan luoda nopeasti kuvaavia esimerkkejä järjestelmästä ja myöhemmin muuntaa ne luokkakaavioiksi. Tässä tutkimuksessa esiin tullut oliokaavion hyödyllisyys kohdealueeseen ja järjestelmään perehtymisessä tukee sitä, että oliokaavion käyttö voi edistää mallintamisen oppimista ja käyttöä, niin koulutusohjelmissa kuin käytännön työssäkin.

Torchiano ym. (2013) toteavat osaamisen ja työtä tukevien työkalujen puutteen tekijöiksi, jotka voivat estää mallintamisen käytön kokonaan ja joihin tulisi vastata korkeakoulujen opetusohjelmilla ja aktiivisella työkalukehityksellä. Torchiano ym. (2013) uskovat, että työkaluissa piilee valtava markkinapotentiaali. Osaamisen ja työkalutuen puutteet liittyvät kuitenkin myös muihin haasteisiin, kuten tässä tutkimuksessa tuli esille. Panostus pelkkään työkalukehitykseen ei ole riittävää, jos ammattilaisilla ei ole aikaa tai kiinnostusta tutustua uusiin työkaluihin. Sopivia työkaluja voi itse asiassa jo olla markkinoilla, mutta ammattilaiset eivät niitä aktiivisesti etsi. Myöskään kattava koulutus ei takaa mallintamisen hyödyntämistä, jos käytännön työssä sille ei ole sijaa. Halua pa-

nostaa mallintamiseen kaivataan siis organisaatioissa myös päättäjiltä, myyjiltä ja ostajilta. Näin mallintajilta ja heidän tiimiltään voidaan odottaa kiinnostusta mallintamista ja sitä tukevia työkaluja kohtaan. Organisaatiotasolla on oltava selvää, että mallintaminen (projektin ja kehitystyön tarpeisiin suhteutettuna) on osa onnistuneen lopputuloksen aikaansaamista.

Ketterän kehityksen yleistyttyä tähän tutkimukseen osallistuneiden kaltaisia, menetelmällisyyttä korostavan koulutuksen saaneita ammattilaisia ei pian enää alalla ole. Ala tarvitseekin panostusta mallintamiseen muulla tavoin kuin vain perinteisiin nojaamalla, jotta osaaminen ei vähitellen häviä kokonaan. Toiveikkaana voidaan pitää sitä, että tässä tutkimuksessa hieman nuorempaa sukupolvea edustavat ammattilaiset suhtautuvat myönteisemmin mallipohjaiseen kehittämiseen (todennäköisesti, koska heillä ei ole huonoja kokemuksia koodigeneroinnista) ja uusien tekniikoiden ja työkalujen käyttöönottoon. He myös uskovat, että mallintamista tullaan painottamaan tulevaisuudessa enemmän, kun koodaustyö väistämättä automatisoituu. Tästä huolimatta todetaan Gorschekin ym. (2014) ja Störrlen (2017) tavoin, että perinteiseen mallintamiseen liittyvä osaaminen ja laajamittainen hyödyntäminen ovat todennäköisesti perusedellytyksiä mallipohjaisen kehitystyön yleistymiselle. Tällä hetkellä käytännön todellisuus ei vaikuta rohkaisevan vaadittavan osaamisen kehittämiseen.

6.4 Tutkimuksen rajoitteet

Kuten kaikessa tutkimuksessa, myös tässä havaittiin rajoitteita. Tutkimuksen otanta on laadulliselle tutkimukselle ominaisesti suppea, joten tuloksia ei voida yleistää koskemaan kaikkia Suomessa toimivia tietojärjestelmäkehityksen ammattilaisia, mihin ei toisaalta olla edes pyritty. Otanta suoritettiin suureksi osaksi lumipallomenetelmällä, joten haastateltavien näkemykset ja kokemukset voivat mahdollisesti olla tyypillisiä ainoastaan sen avulla saavutetun verkoston sisällä. Lumipallo-otannalla saavutettiin etenkin hyvin kokeneita ammattilaisia, joten heidän mielipiteensä voivat erota suurestikin alan kokemattomampien kollegoiden näkemyksistä. On myös mahdollista, että tutkimuksen aihe on houkutellut haastateltaviksi ammattilaisia, jotka ovat erityisen kiinnostuneita mallintamisesta ja kokevat sen hyödyllisemmäksi kuin valtaosa alalla työskentelevistä.

Mallintamisen koetun hyödyllisyyden ja merkitsevyyden lisäksi otanta asettaa kyseenalaiseksi tulosten perusteella todetun UML:n käytön yleisyyden (kaikki haastateltavat käyttivät UML:ää). Haastateltaviksi valikoitui todennäköisesti UML:ää suhteellisen hyvin tuntevia ja usein käyttäviä ammattilaisia siitäkin huolimatta, että aineiston sisällä oli näiden suhteen huomattavaa hajontaa. UML:n käyttö ja tunnettuus voi siten olla alalla tutkimustuloksia selkeästi heikommalla tasolla ottaen huomioon, että tähän löytyi viitteitä myös aineistosta ja että haastattelusta kieltäytyi ammattilaisia UML:n vähäiseen käyttöön vedoten.

Mahdollisia haastattelutilanteiden ongelmia ja tulosten tulkintaan liittyvää epämääräisyyttä pyrittiin estämään hyvällä etukäteisvalmistelulla. Osa haastateltavista ei kuitenkaan kokenut tarpeelliseksi tarkastella haastattelun aikana esimerkkejä kaavioista, joten täydestä yhteisymmärryksestä termien ja käytössä olevien tekniikoiden suhteen ei ole varmuutta. Tutkija ei myöskään saanut luvusta 5.5.1 löytyvän kuvion 22 lisäksi näytteitä haastateltavien tekemistä kaavioista, joten tulokset perustuvat haastateltavien kertomuksiin sekä tutkijan tulkintoihin niistä. Haastateltavien kertomuksiin ovat puolestaan voineet vaikuttaa tutkijan ja haastateltavan välisen vuorovaikutuksen lisäksi se, miten todennäköisinä he pitivät anonymiteettinsä säilymistä.

Tutkimuksen tulokset perustuvat suurelta osin haastatteluaineistosta esiin nouseviin teemoihin ja tutkijan subjektiivisiin tulkintoihin niistä. Kaikkia teemoja tai esitettyjä tuloksia ei esiintynyt tai käyty läpi jokaisessa haastattelussa, haastatteluteemoja lukuun ottamatta. Toinen haastattelukierros olisi siten voinut tuottaa hieman erilaisia tuloksia, kuten myös toisen tutkijan tekemä analyysi. Yhtenä rajoitteena voidaan nähdä myös tutkijan kokemuksen puute käytännön mallintamiseen ja tietojärjestelmäkehitykseen liittyen, mikä toisaalta on myös mahdollistanut aiheen tarkastelun ilman ennakkoasenteita.

Tutkimuksessa todettiin, että mallintamiseen käytettävillä työkaluilla on merkittävä rooli niillä tuotetun kuvan muodossa ja laadussa. Työkaluihin liittyy myös yksi mahdollinen tutkimuksen rajoite. Työkaluilla on paljon ominaisuuksia, joilla todennäköisesti voidaan korvata mallinnuskieliin liittyviä puutteita ja toisaalta mahdollisesti myös hankaloittaa mallintamistyötä. Jos tutkijalla olisi ollut laaja ja syvällinen tuntemus markkinoilla olevien työkalujen räätäliominaisuuksista tai edes tarkasteltavana haastateltavien eri työkaluilla tuottamia kaavioita, olisi työkalujen merkitystä voitu pohtia syvällisemmin.

6.5 Tulosten hyödyntäminen tutkimuksessa

Kuten luvussa 4.1 todettiin, mallintamisen tutkimusta on pitkään vaivannut käytännön todisteiden puute etenkin mallintamista työkseen suorittavien ammattilaisten näkemyksistä ja kokemuksista. Tällä haastattelututkimuksella kerrytettiin arvokasta empiiristä todistusaineistoa mallintamisen käytännöistä, yleisyydestä ja hyödyllisyydestä nykypäivän tietojärjestelmäkehityksessä. Tutkimusote oli laadullinen, joten sen avulla pyrittiin tuomaan esiin ammattilaisten näkemyksiä ja kokemuksia sekä ymmärtämään niitä syvällisemmin. Näin voidaan tarjota tietoa ohjelmisto-, menetelmä- ja työkalukehittäjille, ohjata tutkimuspanosta mielenkiintoisiin ja hyödyllisiin osa-alueisiin, sekä muodostaa hypoteeseja määrällistä kyselytutkimusta varten (ks. Ghauri & Grønhaug, 2005, s. 111).

Tällä haastattelututkimuksella kartutettiin alueellista tietoa mallintamisen käytännöistä tarjoamalla Suomessa toimivien ammattilaisten näkemyksiä ja kokemuksia. Laadullisen tutkimusotteen ja siihen liittyvän otannan asettamien rajoitteiden vuoksi tuloksia ei voida yleistää koskemaan kaikkia Suomessa toi-

mivia ammattilaisia. Tulosten vahvistaminen (tai kumoaminen) ja yleistäminen vaativat laajempaa otantaa. Ilmeisimpänä jatkotutkimusaiheena esitetäänkin määrällisen kyselytutkimuksen toteuttaminen Suomessa toimivien ammattilaisten mallintamiskäytännöistä. Yleistettävillä tuloksilla voidaan tarjota entistäkin olennaisempaa tietoa toimittajaorganisaatioille sekä menetelmä- ja työkalukehittäjille.

Kuten tässä haastattelututkimuksessa todettiin, mallintamisen hyödyntämiseen vaikuttavat ammattilaisen koulutus ja työtehtävä, kehitettävä tai toimitettava ratkaisu ja siihen liittyvät asiakasvaatimukset, ja projektin luonteen ja koon lisäksi kehitystiimin koko sekä osaaminen. Nämä tekijät tulee huomioida kyselytutkimusta toteutettaessa sekä sen tuloksia analysoitaessa. Alan terminologia ja kaaviotekniikoiden hyödyntäminen ovat kirjavia, joten erityistä huomiota tulee kiinnittää kysymyksenasetteluun ja mahdollisten havainnollistavien kuvien valintaan. Haastattelututkimuksessa havaittiin myös, että erilaisten mallinnustekniikoiden käytön raportointiin voi vaikuttaa tekniikoihin liittyvä tuntemus ja oletukset. Tulevissa tutkimuksissa tuleekin määritellä tarkasti, mitä esimerkiksi UML:n käytöllä tutkimuskontekstissa tarkoitetaan, käsitetäänkö se kaavio-, UML- vai metamallinäkökulmasta.

Mallintaminen todettiin tässä tutkimuksessa hyödylliseksi etenkin analyysiin, alustavaan suunnitteluun ja ylläpitoon liittyvissä tehtävissä. Muiden tietojärjestelmäkehityksen vaiheiden, prosessien tai aktiviteettien osalta mallintamisen rooli ei ollut näin selkeä. Tutkimuksessa nousi esille korkean abstraktiotason mallit ja niissä hyvin vapaamuotoiset kuvaustavat. Tarkalla tasolla tehtävää mallintamista esiintyi kyllä tuotteistettujen ratkaisujen yhteydessä, mutta kyse on mahdollisesti ainoastaan dokumentoinnista. Jatkotutkimuksella voidaan selventää, millaisissa projekteissa ja sovellusalueissa suoritetaan nykypäivänäkin yksityiskohtaista mallintamista myös suunnittelussa ja mitä menetelmiä siinä käytetään. Myös mallien merkitystä ja niihin liittyvää osaamista koodaajien ja testaajien työssä tulee tutkia lisää.

Tässä tutkimuksessa havaittiin, että mallintamis- ja dokumentaatiomenettelyt sekä suhtautuminen niihin ovat hyvin vaihtelevia ohjelmisto- ja IT-palveluyrityksissä. Osassa haastatteluja tuli ilmi, että organisaatiossa tai sen osassa on määritelty toimintatapoja mallintamiseen liittyen, osassa taas, että toimintatapoja ei ollut tai ainakaan niistä ei oltu tietoisia. Jatkotutkimusaiheena esitetäänkin organisaatioiden mallintamiseen ja dokumentointiin liittyvien toimintatapojen ja suositusten merkityksen selvittämistä ja vaikuttavuuden arviointia. Noudatetaanko näitä suosituksia, kuinka usein niitä tarkistetaan ja onko niillä havaittu olevan toivottuja vaikutuksia, onko niillä kenties saatu työskentelyyn liittyviä haasteita häivytettyä?

Tutkimuksessa esiin tullut mallintamisen merkitys ylläpidossa on osin ristiriidassa aiempien tutkimusten kanssa, joissa esitetään muun muassa, että kaavioita ei yleisesti hyödynnetä järjestelmään perehtymisessä (ks. Badreddin ym., 2018; Störrle, 2017). Myös tämän tutkimuksen perusteella voidaan sanoa, että dokumentaatio ei yleensä ole toivotulla tasolla. Tuleva tutkimus voisikin pureutua asiaan tarkemmin ja selvittää menettelyjä ylläpidossa. Johtuuko mahdolli-

nen mallien sivuuttaminen niihin liittyvistä haasteista (malleja ei ymmärretä, niitä ei ole pidetty ajantasaisina tai ne ovat liian matalalla abstraktiotasolla) vai siitä, että muita perehtymistapoja pidetään hyödyllisempänä? Myös tässä tutkimuksessa yleiseksi käytännöksi esitetyn ”järjestelmän kartan” erilaisia esitysmuotoja ja niiden hyödyllisyyttä tulisi tutkia lisää.

UML ei tämän tutkimuksen perusteella sovellu kovin hyvin mallipohjaiseen kehittämiseen eikä sen käyttöä yhdistetä koodigenerointiin. Pois lukien automatisoitua sovelluskehitystä toteuttava organisaatio H, koodigeneroinnin hyödyntäminen nykypäivän ohjelmisto- ja IT-palveluyrityksissä jäi tämän tutkimuksen perusteella kuitenkin epäselväksi. Yleisesti ottaen haastateltavat eivät olleet kovin tietoisia organisaatioissaan suoritettavasta koodigeneroinnista. Tätä osa-aluetta voisi tutkia lisää, kuten myös koodia mallien pohjalta tuottavien työkalujen nykytilaa.

Monen muun tutkimuksen tavoin (ks. esim. Farias ym., 2018; Ozkaya, 2018) myös tässä tutkimuksessa raportoitiin ammattilaisten tyytymättömyyttä mallintamisessa käytettäviä työkaluja kohtaan. Kuten mihin tahansa tietojärjestelmäkehityksessä käytettävään työkaluun, myös mallinnustyökalujen kehitykseen tulee panostaa. Tässä tutkimuksessa esitettiin ammattilaisten toivomia ominaisuuksia työkaluille, mutta havaittiin myös, että ammattilaisilla on harvoin aikaa tai kiinnostusta aktiivisesti etsiä näitä ominaisuuksia sisältäviä työkaluja. Jatkotutkimuksella näihin tarpeisiin voidaan pureutua vielä yksityiskohtaisemmin sekä kartoittaa toivottuja ominaisuuksia sisältäviä työkaluja tai niiden kehityshankkeita.

6.6 Tulosten hyödyntäminen käytännössä

Tietojärjestelmäkehitys on suhteellisen nuori ala, joka jatkuvasti etsii parhaita käytäntöjään. Aiemmin vallalla olleen suunnitteluvetoisen kehitystyön menetelmällisyydestä ja raskaista prosesseista on luovuttu, mutta ketteryyden aikakausi on kuitenkin tuonut mukanaan uusia haasteita. Tämän tutkimuksen perusteella nämä haasteet voivat väärinkäsitysten ja uusintatöiden kautta johtaa laadun ja tuottavuuden laskuun. Ketteryys itse ei välttämättä ole ongelma, vaan suunnittelun ja dokumentoinnin sivuuttaminen myös silloin, kun ne olisivat tarpeen. Voidaankin todeta, että teknologian kehityksestä ja ketteryyden yleistymisestä huolimatta tietojärjestelmien rakenteiden ja toimintojen kuvaamisen tarve ei ole poistunut.

Käytännön kannalta on merkittävää, että tämän tutkimuksen mukaan ammattilaiset eivät useinkaan koe pystyvänsä toteuttamaan työtään riittävän laadukkaasti. Luvussa 5.3.1 kerrottiin monen ammattilaisen kokevan, että mallintamista olisi hyvä tehdä enemmän, mutta usein haasteet ylittävät nämä tarpeet. Mallintamiseen ei ole tarvittavaa aikaa, osaamista tai kiinnostusta, mikä osaltaan johtunee alan koodikeskeisyydestä. Tutkimuksessa esitettiin nykyisten toimintatapojen olevan jopa niin huonoja, että ne usein väistämättä johtavat kustannusten ylitykseen ja bugisiin tai vaatimuksia täyttämättömiin lopputu-

loksiin. Sekä toimittaja- että ostajaorganisaatioiden tulisi siis pohtia, kuinka pelkkien lyhyen tähtäimen hyötyjen tavoittelemisesta edetään elinkaariajatteluun, ei ainoastaan puheissa, vaan myös teoissa. Jatkossa mallintamista voidaan koodaustyön automatisoinnin vuoksi tarvita jopa enemmän, mutta sen nykyinen aliarvostaminen voi estää vaadittavien taitojen kehittymisen.

Kuten luvussa 6.1 todettiin, tämän tutkimuksen perusteella mallintamisella on selkeä viestinnällinen rooli, jonka merkityksellisyys kuitenkin vaihtelee projektista, tietojärjestelmästä ja kehitystiimistä toiseen. Mallintamisen mahdolliset hyödyt voivat olla vaikeita todentaa, etenkin kun kehitystyö tuntuu sujuvan hyvin ilmankin. Tämän tutkimuksen perusteella mallintamisen hyödyllisyys noteerataankin usein vasta ongelmien ilmaannuttua. Mallintaminen voidaan kokea turhaksi, jos ei ole tarvetta ajatella kokonaisuutta tai kehitettävän ratkaisun elinkaarta pidemmälle kuin oman työtehtävän osalta tai jos mallien puutteesta johtuvia ongelmia ei ole vielä itse kokenut. Perinteisen mallintamisen hyödyt realisoituvatkin vasta pidemmällä aikavälillä, kuten laadukkaammassa lopputuloksessa ja sujuvammassa ylläpitotyössä. Selvää on, että mallintamisesta saadaan eniten irti tilanteessa, jossa haasteita on vähiten eli motivoituneilla tekijöillä on tarvittavaa osaamista, organisaatio tukee toimintaa esimerkiksi panostamalla ohjeistuksiin ja työkaluihin, ja asiakas on laatutietoinen.

Tässä tutkimuksessa havaittiin aiempien tutkimusten (esim. Grossman ym., 2005; Hutchinson ym., 2011b) tavoin, että suhtautuminen mallintamiseen ja etenkin UML:ään on hyvin kaksijakoista. UML on terminä kaikille tuttu ja myös käytetyin kuvaustapa vapaamuotoisten ”laatikkojen ja viivojen” ohella. UML:ään liittyvä osaaminen on kuitenkin vaihtelevaa, kuten myös mielipiteet sen käytön ja jatkokehittämisen hyödyllisyydestä. Tutkimuksen perusteella voidaan sanoa, että nykypäivänä UML:n säännönmukainen noudattaminen ei ole hyödyllistä. UML:ää voidaan kuitenkin käyttää ohjeistuksena, josta etsitään kulloiseenkin viestintätarpeeseen sopiva kuvaustapa. Vaikka kaikki UML:n kaaviotyypit, säännöt tai työkalut eivät sopisi suurimpaan osaan kehitystyötä, ei se tarkoita, etteikö UML tarjoaisi ilmaisuvoimaisia ja moniin käyttökohteisiin soveltuvia tekniikoita. Todennäköisesti myös vapaamuotoiset notaatiot noudattelevat pitkälti UML:n tekniikoiden logiikkaa, koska UML sisältyy moniin koulutusohjelmiin ja tämän tutkimuksen perusteella ammattilaiset käyttävät niitä tekniikoita, joihin ovat koulutusta saaneet.

UML:ään alusta asti sisältyneiden kaaviotekniikoiden, kuten sekvenssi-, aktiviteetti-, käyttötapaus- ja luokkakaavioiden käyttö on tämän tutkimuksen mukaan yleisintä (ks. myös Reggio ym., 2014), ja ne myös todettiin hyödyllisimmiksi. Niiden käytössä näkevät arvoa myös ne ammattilaiset, jotka muutoin eivät UML:n tarjoamasta kokonaisuudesta perusta. Nämä tekniikat ovat olleet käytössä jo ennen UML:ää ja niiden tapaa kuvata rakenteita ja käyttäytymistä tarvitaan edelleen. Tunnettuuden ja hyödyllisyyden vuoksi myös asiakasvaatimukset koskevat todennäköisimmin juuri näitä tekniikoita. Tästä voi päätellä, että tietojärjestelmäkehityksessä työskentelevien kannattaa panostaa ainakin yleisimpien UML-kaavioiden koulutukseen ja käyttöön yhteistyön sujuvoittamiseksi.

Sekvenssikaavio on tämän tutkimuksen perusteella yleisin ja mahdollisesti siksi myös hyödyllisin UML:n sisältämä kaaviotekniikka. Sekvenssikaaviota voidaan käyttää monessa yhteydessä ja sen selkeys mahdollistaa sen ymmärtämisen useassa eri ammattiryhmässä. Kaavion suosio perustuu mahdollisesti ainakin osin sen esimerkinomaisuuteen. Tästä voidaan päätellä, että luvussa 6.2 esitelty Kästnerin ym. (2018) ehdotus oliokaavion käytöstä silloin, kun abstrakti ajattelu on haasteellista, voi olla käytännön työssä hyödyllinen. Oliokaavio on sekvenssikaavion tapaan esimerkinomainen ja siten todennäköisesti monelle käsitteellisen tason luokkakaaviota helpommin sisäistettävissä. Oliokaavion käyttöä voidaan suositella tiedon mallintamisessa, helpottamaan käsitteiden määrittämistä ja niiden välisten yhteyksien tunnistamista. Käytön hyödyt voivat ilmetä myös ylläpidossa, kun järjestelmään perehtymisessä koodi tai monimutkaiset luokkakaaviot eivät riitä.

Tämän tutkimuksen perusteella mallintamisen viestinnällinen rooli korostuu kehitystyön alkuvaiheen lisäksi ylläpidossa. Tämä on merkittävää, koska ylläpitotyö on kallista ja yhä useampi projekti keskittyy jatkokehittämiseen ja olemassa olevien järjestelmien yhdistämiseen. Vaikka malleja ei koettaisi tarpeellisiksi kehitystyön aikana omassa tai tiimin työskentelyssä, niitä tulisi tehdä jo pelkästään dokumentaation ja vääjäämättömän ylläpitotyön vuoksi, järjestelmään perehtymistä helpottamaan ja laatua varmistamaan. On kuitenkin lisättävä, että tämän tutkimuksen perusteella mallintamisella edesautetaan koko kehitystyön tai projektin onnistumista. Jopa kokeneimmat ammattilaiset pitävät mallintamista tärkeänä apuvälineenä omalle työskentelylleen ja kokevat sen edesauttavan ymmärrystä ja ongelmanratkaisua. Mallintamisen hyödyt eivät siten rajoitu dokumentaatioon.

Ylläpitotyö on ihmislähtöinen prosessi, joten siinä kaivataan korkean abstraktiotason kaavioita perehtymistä helpottamaan. Työkalukehitykseen ja -käyttöönottoon panostamalla koodipohjasta generoitavien kaavioiden muotoon ja laatuun voidaan vaikuttaa, mutta tällä hetkellä niillä ei kuitenkaan tämän tutkimuksen mukaan voida täysin korvata käsin tehtyjä malleja. On siis tärkeää, että tällaisia malleja luodaan, dokumentoidaan ja ylläpidetään myös jatkossa. Näin voidaan nopeuttaa järjestelmän rakenteisiin ja toimintoihin perehtymistä sekä vähentää perehdyttämiseen mahdollisesti sidottuja henkilöresursseja (ks. suullinen tieto, Forward ym., 2010; Badreddin ym., 2018). Kaikkien tietojärjestelmäkehitykseen osallistuvien tulisikin ymmärtää kokonaiskuva ja katsoa lyhyen tähtäimen hyötyjä pidemmälle. Tämän tutkimuksen perusteella onnistuminen voi vaatia organisaatiolta lisää tukea, jolla vaikutetaan työntekijöiden motivaatioon ja osaamiseen.

Luvussa 6.3 esitettiin, että kiire, mallien vanheneminen koodin muuttuessa ja tästä johtuva ajantasaisena pitämisen taakka ovat merkittäviä haasteita (ks. myös Farias ym., 2018; Forward ym., 2010). Aikaa ei voi taikoa lisää, mutta sen puutteeseen liittyviin haasteisiin voidaan vastata tämän tutkimuksen mukaan esimerkiksi ohjaamalla tietyn abstraktiotason mallintamiseen ja työkaluilla generoitavien eli ajantasaisten kaavioiden lisäämiseen dokumentaatioon, mallien tallentamiseen niin, että ne löytyvät vaivatta sekä mallien päivittämiseen sisäl-

tyvän työn jakamiseen tiimin kesken. Tarkoituksena ei ole tietyn toimintatavan pakottaminen joka tilanteeseen, mikä ei ketteryyteen sovi (ks. myös Whittle ym. 2013), vaan hyväksi havaittujen ratkaisujen tarjoaminen, joista jokainen voi saada ideoita omaan työskentelyynsä. Etenkään koko ajan parempien tekniikoiden ja työkalujen etsimiseen tai käyttöönottoon ei ole aikaa ja tämän vuoksi ohjeistuksilla ja suosituksilla voidaan edistää kehitystyötä. Kustannustehokkuus edellyttää, että ratkaisu on helposti saatavilla, eikä hyviä tapoja tarvitse jokaisen keksiä uudelleen, etenkään ongelmien kautta.

Tämän tutkimuksen mukaan malleja tarvitaan ja käytetään etenkin liiketoimintaprosessien, tietokantojen ja -mallien sekä rajapintojen kuvaamiseen ja dokumentoimiseen. Ylläpidon kannalta erityisen tärkeäksi malliksi osoittautui luvussa 5.5.1 käsitelty verkkoarkkitehtuurin kuvaus, ”järjestelmän kartta”. Tutkimuksessa ei kuitenkaan päästy täyteen yksimielisyyteen siitä, mikä olisi paras tapa tällaisen kartan kuvaamiseen. Erilaisia notaatioita mahdollisesti yhdistellään yhteen kuvaan tai käytetään työkalujen tarjoamia räätäliominaisuuksia, lopputulosten ollessa kirjavia. Tässä tutkimuksessa painotettiin korkeaa abstraktiotasoa palveluiden sijaintien ja niiden välisien vuorovaikutuksen kuvaamiseen. Karttoja täydentämään voidaan dokumentoida rakennetta ja toimintoja erikseen sekä generoida työkaluilla koodipohjasta tarkempia kuvauksia.

Dokumentaatiota tuotettaessa on hyvä muistaa, että taitavastikin muotoiltu kaavio voi olla hyödytön, jos sitä ei päivitetä kehitystyön edetessä. On myös mietittävä millä tavoin dokumentaation luotettavuutta voidaan parantaa. Dokumentaation vähäisyyden lisäksi sen huono laatu on yleinen ongelma alalla, kuten luvuissa 5.2.5, 5.3 ja 5.5.1 tuotiin esille. On siis mahdollista, että dokumentoidut mallit sivuutetaan automaattisesti niiden oletetun paikkansapitämättömyyden vuoksi. Laadittavien toimintatapojen tulisi siten sisältää ohjeistus dokumentaatiomerkinnöille, joilla varmistetaan lukijalle, että dokumentaatio ja sen sisältämät kaaviot ovat ajantasaisia.

Työkalujen puutteet ovat pysyvä haaste mallintamiseen liittyen (ks. Badreddin ym., 2018). Myös tässä tutkimuksessa nousi esiin haasteita, joiden perusteella voidaan ratkaisuksi ehdottaa Forwardin ym. (2010) tapaan tekstinotaatiota tukevien työkalujen käyttöä ja jatkokehittämistä. Tällaiset työkalut voisivat sopia jopa koodikeskeisimmille ammattilaisille ja myös häivyttää mallintamiseen liittyvää taideaspektia, koska kaavoita ei ”piirrellä”. Luvussa 5.4.3 kävi ilmi, että tekstinotaatiota tukevat työkalut, kuten PlantUML, myös ovat muihin verrattuina helppokäyttöisempiä eli nopeuttavat kaavioiden luomista ja ylläpitoa sekä mahdollistavat versionhallinnan. Tämän tutkimuksen perusteella ei kuitenkaan voida sanoa, miten monipuolisia tällaiset työkalut tällä hetkellä ovat. Voi olla, että ne soveltuvat hyvin rajattuun käyttöön ja niillä tuotettujen kaavioiden ulkonäkö vaatii vielä paljon kehitystyötä.

Hajautetun ohjelmistokehityksen ja yleistyvän etätyön vuoksi ammattilaisilla voi olla yhä enemmän tarvetta palavereissa valkotaulun piirtopintana korvaaville ohjelmistoille. Fariaksen ym. (2018) tavoin esitetäänkin kollaboratiivista suunnittelua ja järjestelmän kokonaiskuvan generointia tukevien työkalujen kehittämistä ja käyttöä, joilla tuetaan yhteisen vision ymmärtämistä ja laaduk-

kaan lopputuloksen tuottamista. Yhdessä ja nopeasti hyvän ratkaisun rakentaminen on olennaista, koska nykypäivän kehitystyössä ei todennäköisesti ole aikaa läpikäydä ja visualisoida useita eri vaihtoehtoja (Babar, 2009), mikä perinteisesti on käsitetty yhdeksi mallintamisen merkittäväksi hyödyksi (ks. Pressman, 2005). Tutkimuksessa nousi esille myös tarve työkaluille, jotka tuottavat yhä laadukkaampia takaisinmallinnettuja kaavioita, joilla helpotetaan järjestelmään perehtymistä ja minimoidaan tarvetta päivittää malleja käsin.

Organisaatioissa tai projekteissa määriteltyjä ohjeistuksia ja toimintatapoja pidetään tämän tutkimuksen mukaan hyödyllisinä apuvälineinä. Niiden laatiminen ja noudattaminen on toki luonnollisempaa ympäristöissä, joissa projekteissa ja kehitetyissä järjestelmissä on toisteisuutta, kuten tuotteistettujen ratkaisujen tai sovellusaluekohtaisen kehityksen yhteydessä. Kaikissa organisaatioissa, niiden osissa ja tiimeissä on kuitenkin tämän tutkimuksen perusteella vakiintuneita ja hyväksi todettuja toimintatapoja. Tällaisista käytännöistä muodostetuista ohjeistuksista voivat hyötyä etenkin kokemattomammat ohjelmistokehittäjät. Pohdittavaksi organisaatioissa jää, ovatko nämä vakiintuneet toimintatavat hyviä vai kaipaisivatko ne päivitystä esimerkiksi koulutuksen tarjoamien uusien ideoiden muodossa, ja löytyykö organisaatiosta ammattilaisia, jotka toisivat mallintamiseen liittyvää tietoutta esille.

Tutkimuksessa havaittiin, että ammattilaiset käyttävät niitä mallinnustekniikoita, joihin ovat koulutuksensa ja kokemuksensa kautta tottuneet. Koulutus ja mallien käyttö myös näyttävät lisäävän mallintamisaktiviteettia ja sen hyödyllisyyden ymmärtämistä. Sen sijaan epäilys omista taidoista mallintajana voi toimia esteenä mallintamisen toteuttamiselle. Jos mallintamiseen halutaan organisaatiotasolla vaikuttaa, on siihen kannustaminen koulutuksen ja ohjeistusten kautta siten hyvin perusteltua. Koulutus- ja ohjeistustarpeita määritettäessä kannattaa huomioida, että mallintamisen perusteet voivat osalta puuttua kokonaan ja että viestinnän onnistumiseksi osaamista voivat tarvita muutkin kuin arkkitehdit tai määrittelevät konsultit.

Tämän tutkimuksen perusteella todetaan, että mallintamiseen liittyvät haasteet ovat moninaisia, vaikuttavat toisiinsa ja ovat ainakin osin lähtöisin koodikeskeisistä asenteista. Koulutuksen, ohjeistusten ja suositusten tarjoaminen ei siten ole mikään taikaratkaisu, jolla haasteet voidaan välittömästi poistaa. Toiminnalla voidaan kuitenkin osoittaa, että organisaatiossa halutaan panostaa parhaisiin käytäntöihin ja laatuun, eikä ainoastaan lyhyen tähtäimen hyötyihin. Tällä tavoin on mahdollista vaikuttaa kehitystyöhön liittyviin asenteisiin ja ammattilaisten itseohjautuvuuteen sekä kenties jopa tuomaan esiin "taistelijoita", jotka ajavat mallintamiseen liittyvää sisäistä kehitystä eteenpäin. Mallintamisen asettamisella etualalle voidaan entisestään edistää viestintää ja siten yhteistyötä tietojärjestelmäkehityksessä.

LÄHTEET

Abrahamsson, P., Babar, M. A. & Kruchten, P. (2010). Agility and architecture: Can they coexist?. *IEEE Software*, *27*(2), 16–22.

Alhojailan, M. I. (2012). Thematic analysis: A critical review of its process and evaluation. *West East Journal of Social Sciences*, *1*(1), 39–47.

Ambler, S. (2006). The Agile Unified Process (AUP). Haettu osoitteesta http://www.ambysoft.com/unifiedprocess/agileUP.html

Anda, B., Hansen, K., Gullesen, I. & Thorsen, H. K. (2006). Experiences from introducing UML-based development in a large safety-critical project. *Empirical Software Engineering*, *11*(4), 555–581.

André, É., Choppy, C. & Reggio, G. (2014). Activity diagrams patterns for modeling business processes. Teoksessa *Software Engineering Research, Management and Applications* (197–213). Heidelberg: Springer International Publishing.

Atkinson, C. & Kühne, T. (2003). Model-driven development: a metamodeling foundation. *IEEE software*, *20*(5), 36–41.

Avison, D. & Fitzgerald, G. (2006). *Information Systems Development Methodologies, Techniques and Tools*. (4. uud. painos). Mateu Cromo: McGraw-Hill.

Babar, M. A. (2009). An exploratory study of architectural practices and challenges in using agile software development approaches. Teoksessa *Proceedings of the Joint 8th Working IEEE/IFIP Conference on Software Architecture & 3rd European Conference on Software Architecture (WICSA/ECSA)* (81–90). Cambridge: IEEE Computer Society.

Badreddin, O., Khandoker, R., Forward, A., Masmali, O. & Lethbridge, T. C. (2018). A Decade of Software Design and Modeling: A Survey to Uncover Trends of the Practice. Teoksessa *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems* (245–255). New York: ACM.

Badreddin, O., Lethbridge, T. C. & Elassar, M. (2013). Modeling practices in open source software. Teoksessa E. Petrinja, G. Succi, N. El Ioini & A. Sillitti (toim.), *Open Source Software: Quality Verification, (OSS 2013), IFIP Advances in Information and Communication Technology* (127–139). Berlin: Springer-Verlag.

Barriball, L. K. & While, A. (1994). Collecting Data using a semi-structured interview: a discussion paper. *Journal of advanced nursing, 19*(2), 328–335.

Bass, L., Clements, P. & Kazman, R. (2003). *Software architecture in practice*. Reading, MA: Addison-Wesley.

Beck, K., Beedle, M., Bennekum van. A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. (2001). Manifesto for agile software development. Haettu osoitteesta http://AgileManifesto.org

Bennett, S., McRobb, S. & Farmer, R. (2006). *Object-oriented systems analysis and design using UML* (3. uud. painos). London: McGraw-Hill.

Biernacki, P. & Waldorf, D. (1981). Snowball sampling: Problems and techniques of chain referral sampling. *Sociological methods & research*, *10*(2), 141–163.

Bjeković, M., Proper, H. A. & Sottet, J. S. (2014). Embracing pragmatics. Teoksessa *International Conference on Conceptual Modeling* (431–444). Cham: Springer International Publishing.

Bjrner, D. & Jones, C. B. (1978). *The Vienna Development Method: The Meta-Language*. Berlin: Springer-Verlag.

Boehm, B. W. (1981). *Software engineering economics* . Englewood Cliffs, NJ: Prentice-Hall.

Boman, M., Bubenko Jr, J. A., Johannesson, P. & Wangler, B. (1997). *Conceptual modelling*. London: Prentice-Hall.

Booch, G. (1994.) *Object-Oriented Analysis and Design with Applications*. (2. uud. painos). Redwood City, CA: Benjamin Cummings.

Booch, G., Jacobson I. & J. Rumbaugh, J. (1996). *Unified Modeling Language for Object-Oriented Development*. Santa Clara, CA: Rational Software Corporation.

Booch, G., Rumbaugh, J. & Jacobson, I. (1999). *The Unified Modeling Language User Guide*. Reading, MA: Addison-Wesley.

Booch, G., Rumbaugh, J. & Jacobson, I. (2005). *Unified Modeling Language User Guide*. (2. uud. painos). Reading, MA: Addison-Wesley.

Bork, D., Buchmann, R., Karagiannis, D., Lee, M. & Miron, E. T. (2019). An open platform for modeling method conceptualization: the OMiLAB digital

ecosystem. *Communications of the Association for Information Systems* 44, 673–697.

Bricknall, R., Darrell, G., Nilsson, H. & Pessi, K. (2006). Enterprise Architecture: Critical Factors affecting modelling and management. Teoksessa J. Ljungberg & M. Andersson (toim.), *Proceedings of the 14th European Conference on Information Systems* (ECIS '06) (1–13), Göteborg, Sweden, June 12-14, 2006.

Brinkkemper, S. (1996). Method engineering: engineering of information systems development methods and tools. *Information and software technology*, *38*(4), 275–280.

Brambilla, M., Cabot, J. & Wimmer, M. (2017). Model-driven software engineering in practice. *Synthesis lectures on software engineering*, *3*(1), 1–207.

Bubenko, J. A. (2007). From information algebra to enterprise modelling and ontologies—a historical perspective on modelling for information systems. Teoksessa *Conceptual Modelling in Information Systems Eng*ineering (1–18). Berlin: Springer-Verlag.

Budgen, D., Burn, A. J., Brereton, O. P., Kitchenham, B. A. & Pretorius, R. (2011). Empirical evidence about the UML: a systematic literature eview. *Software: Practice and Experience*, *41*(4), 363–392.

Bucher, T., Klesse, M., Kurpjuweit, S. & Winter, R. (2007). Situational method engineering. Teoksessa J. Ralyté, S. Brinkkemper & B. Henderson-Sellers (toim.), *Proceedings of the IFIP WG 8.1 Working Conference, Situational Method Engineering: Fundamentals and Experiences* (33–48). Boston, MA: Springer Publishing Company.

Burton-Jones, A. & Meso, P. (2008). The effects of decomposition quality and multiple forms of information on novices' understanding of a domain from a conceptual model. *Journal of the Association for Information Systems*, *9*(12), 1.

Burton-Jones, A., Wand, Y. & Weber, R. (2009). Guidelines for empirical evaluations of conceptual modeling grammars. *Journal of the Association for Information Systems*, *10*(6), 1.

Chaudron, M. R. (2017). Empirical studies into UML in practice: Pitfalls and prospects. Teoksessa *Proceedings of the 9th International Workshop on Modelling in Software Engineering (MISE '17)*, Buenos Aires, Argentina, May 20-28, 2017.

Chaudron, M. R., Heijstek, W. & Nugroho, A. (2012). How effective is UML modeling?. *Software & Systems Modeling*, *11*(4), 571–580.

Checkland, P. B. (1989). Soft systems methodology. *Human systems management*, *8*(4), 273–289.

Chen, P. P. S. (1976). The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems (TODS), 1*(1), 9–36.

Cherubini, M., Venolia, G., DeLine, R. & Ko, A. J. (2007). Let's go to the whiteboard: how and why software developers use drawings. Teoksessa *Proceedings of the SIGCHI conference on Human factors in computing systems* (557–566).

Clarke, R., Burton-Jones, A. & Weber, R. (2016). On the ontological quality and logical quality of conceptual-modeling grammars: The need for a dual perspective. *Information Systems Research, 27*(2), 365–382.

Cockburn, A. (2001). *Writing effective use cases*. Reading, MA: Addison-Wesley.

Curcio, K., Navarro, T., Malucelli, A. & Reinehr, S. (2018). Requirements engineering: A systematic mapping study in agile software development. *Journal of Systems and Software*, *139*, 32–50.

Dagenais, B., Ossher, H., Bellamy, R. K., Robillard, M. P. & De Vries, J. P. (2010). Moving into a new software project landscape. Teoksessa *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1* (275–284). New York: ACM.

Davies, I., Green, P., Rosemann, M., Indulska, M. & Gallo, S. (2006). How do practitioners use conceptual modeling in practice?. *Data & Knowledge Engineering*, *58*(3), 358–380.

DeMarco, T. (1979). Structure analysis and system specification. Teoksessa M. Broy & E. Denert (toim.), *Pioneers and Their Contributions to Software Engineering* (255–288). Berlin: Springer-Verlag.

Dobing, B. & Parsons, J. (2006). How UML is used. *Communications of the ACM*, *49*(5), 109–113.

Dobing, B. & Parsons, J. (2008). Dimensions of UML Diagram Use. *Journal of Database Management*, *19*(1), 1–18.

Dori, D. (2002). Why significant UML change is unlikely. *Communications of the ACM*, *45*(11), 82–85.

Dori, D. (2011). Object-process methodology for structure-behavior co-design. Teoksessa D. W. Embey & B. Thalheim (toim.), *Handbook of Conceptual Modeling: Theory, Practice, and Research Challenges* (209–258). Berlin: Springer-Verlag.

Duddy, K. (2002). UML2 must enable a family of languages. *Communications of the ACM*, *45*(11), 73–75.

Ebert, C. (2014). Software product management. *IEEE Software*, *31*(3), 21–24.

Easton, K. L., McComish, J. F. & Greenberg, R. (2000). Avoiding common pitfalls in qualitative data collection and transcription. *Qualitative health research*, *10*(5), 703–707.

Elaasar, M., Noyrit, F., Badreddin, O. & Gérard, S. (2018). Reducing UML Modeling Tool Complexity with Architectural Contexts and Viewpoints. Teoksessa *Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2018)* (129–138). Setúbal: SciTePress.

Elaasar, M., Noyrit, F., Badreddin, O. & Gérard, S. (2019). Adaptation and Implementation of the ISO42010 Standard to Software Design and Modeling Tools. Teoksessa *Springer Model-Driven Engineering and Software Development* (236–258). Cham: Springer International Publishing.

Endres, A. & Rombach, H. D. (2003). *A handbook of software and systems engineering: Empirical observations, laws, and theories*. Harlow: Pearson Education.

Everest, G. C. (1976). Basic data structure models explained with a common example. Teoksessa *Proceedings of the Fifth Texas Conference on Computing Systems* (39–46). Long Beach, CA: IEEE Computer Society Publications Office.

Falkenberg, E., Hesse, W., Lindgreeen, P., Nilsson, B., Oei, J., Rolland, C.; Stamper, R., Van Assche, F., Verrijn-Stuart, A. & Voss, K. (1998). F*RISCO: A framework of information system concepts: The FRISCO report (WEB edition)*. Laxenburg: International Federation for Information Processing (IFIP). Haettu osoitteesta https://research.utwente.nl/files/5157230/frisco-full.pdf

Farias, K., Gonçales, L., Bischoff, V., da Silva, B. C., Guimarães, E. T. & Nogle, J. (2018). On the UML use in the Brazilian industry: A state of the practice survey (S). Teoksessa *Proceedings of the 30th International Conference on Software Engineering and Knowledge Engineering, (SEKE 2018)* (372–371), Redwood City, California, July 1-3, 2018.

Feiler, P. H., Lewis, B. & Vestal, S. (2003). The SAE Avionics Architecture Description Language (AADL) Standard: A Basis for Model-Based Architecture-Driven Embedded Systems Engineering. Teoksessa *RTAS 2003 Workshop on Model-Driven Embedded Systems*, Washington, D.C., May 27-30, 2003.

Fettke, P. (2009). How conceptual modeling is used. *Communications of the Association for Information Systems*, *25*(1), 43.

Fernández-Sáez, A. M., Caivano, D., Genero, M. & Chaudron, M. R. (2015). On the use of UML documentation in software maintenance: Results from a survey in industry. Teoksessa *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)* (292–301). Ottawa, Ontario, September 30–October 2, 2015.

Fitsilis, P., Gerogiannis, V. C. & Anthopoulos, L. (2013). Role of unified modelling language in software development in Greece–results from an exploratory study. *IET software*, *8*(4), 143–153.

Fitzgerald, J. S., Larsen, P. G. & Verhoef, M. (2007). Vienna development method. *Wiley Encyclopedia of Computer Science and Engineering*, 1–11.

Forward, A., Badreddin, O. & Lethbridge, T. C. (2010). Perceptions of software modeling: a survey of software practitioners. Teoksessa *The Fifth Workshop From Code Centric to Model Centric: Evaluating the Effectiveness of MDD (C2M:EEMDD)* (12–24). Paris, France, June 15-18, 2010.

Fowler, M. (2003.) *UML Distilled Third Edition: A Brief Guide to the Standard Object Modelling Language*. Reading, MA: Addison-Wesley.

Fowler, M. & Scott, K. (2000.) *UML Distilled Second Edition: A Brief Guide to the Standard Object Modelling Language*. Reading, MA: Addison-Wesley.

France, R. & Rumpe, B. (2007). Model-driven development of complex software: A research roadmap. Teoksessa *Future of Software Engineering (FOSE)* (37–54). Los Alamitos, CA: IEEE Computer Society.

Frank, U. (1999). Conceptual modelling as the core of the information systems discipline - perspectives and epistemological challenges. Teoksessa *Proceedings of the Fifth Americas Conference on Information Systems (AMCIS 1999)* (695–697), Milwaukee, Wisconsin, August 13-15, 1999.

Frank, U. (2002). Multi-perspective enterprise modeling (memo) conceptual framework and modeling languages. Teoksessa *Proceedings of the 35th Hawaii International Conference on System Sciences (HICSS-35)* (1258–1267). Los Alamitos, CA: IEEE Computer Society.

Garlan, D. (2000). Software architecture: a roadmap. Teoksessa A. Finkelstein (toim.), *Proceedings of the Conference on the Future of Software Engineering* (91–101). New York: ACM.

Gemino, A. & Parker, D. (2009). Use case diagrams in support of use case modeling: Deriving understanding from the picture. *Journal of Database Management (JDM)*, *20*(1), 1–24.

Gemino, A. & Wand, Y. (2004). A framework for empirical evaluation of conceptual modeling techniques. *Requirements Engineering*, *9*(4), 248–260.

Ghauri, P. N. & Grønhaug, K. (2005). *Research methods in business studies: A practical guide*. Essex: Pearson Education.

Ghobadi, S. & Mathiassen, L. (2016). Perceived barriers to effective knowledge sharing in agile software teams. *Information Systems Journal, 26*(2), 95–125.

Giaglis, G. M. (2001). A taxonomy of business process modeling and information systems modeling techniques. *International Journal of Flexible Manufacturing Systems*, *13*(2), 209–228.

Glass, R. L. (2002). *Facts and Fallacies of Software Engineering*. Boston, MA: Addison-Wesley.

Gogolla, M. (2011). UML and OCL in Conceptual Modeling. Teoksessa D. W. Embey & B. Thalheim (toim.), *Handbook of Conceptual Modeling: Theory, Practice, and Research Challenges* (85–122). Berlin: Springer-Verlag.

Gordijn, J., Akkermans, H. & van Vliet, H. (2000). Business modelling is not process modelling. Teoksessa S.W Liddle, H.C. Mayr & B. Thalheim (toim.), *Conceptual Modeling for E-Business and the Web, ER 2000, Lecture Notes in Computer Science, vol 1921* (40–51). Berlin: Springer-Verlag.

Gorschek, T., Tempero, E. & Angelis, L. (2014). On the use of software design models in software development practice: An empirical investigation. *Journal of Systems and Software*, *95*, 176–193.

Grossman, M., Aronson, J. E. & McCarthy, R. V. (2005). Does UML make the grade? Insights from the software development community. *Information and Software Technology*, *47*(6), 383–397.

Guizzardi, G. (2005). *Ontological foundations for structural conceptual models* (Väitöskirja). Vol. 015, Telematica Instituut Fundamental Research Series. Enschede, The Netherlands: Telematica Instituut.

Guizzardi, G. (2007). On ontology, ontologies, conceptualizations, modeling languages, and (meta) models. *Frontiers in artificial intelligence and applications*, *155*, 18.

Haikala, I. & Mikkonen, T. (2011). *Ohjelmistotuotannon käytännöt.* (12. uud. painos). Helsinki: Talentum.

Harel, D. (1987). Statecharts: A visual formalism for complex systems. *Science of computer programming*, *8*(3), 231–274.

Hirschheim, R., Klein, H. K. & Lyytinen, K. (1995). *Information systems development and data modeling: conceptual and philosophical foundations*. Cambridge: Cambridge University Press.

Hirsjärvi, S. & Hurme, H. (2000). *Tutkimushaastattelu: Teemahaastattelun teoria ja käytäntö*. Helsinki: Yliopistopaino.

Hirsjärvi, S., Remes, P., Sajavaara, P. & Sinivuori, E. (2009). *Tutki ja kirjoita* . (15. uud. painos). Helsinki: Tammi.

Hofmeister, C., Nord, R. L. & Soni, D. (1999). Describing software architecture with UML. Teoksessa P. Donohoe (toim.), *Software architecture: Proceedings of the 1st Working IFIP Conference on Software Architecture (WICSA1)* (145–159). Dordrecht: Kluwer Academic Publishers.

ter Hofstede, A. H. & Proper, H. A. (1998). How to formalize it?: Formalization principles for information system development methods. *Information and Software Technology*, *40*(10), 519–540.

ter Hofstede, A. H. & van der Weide, T. P. (1992). Formalization of techniques: chopping down the methodology jungle. *Information and Software Technology*, *34*(1), 57–65.

Holstein, J. A. & Gubrium, J. F. (1995). *The active interview* (Vol. 37). Thousand Oaks: Sage Publications.

Hunt, A. & Thomas, D. (2002). Software archaeology. *IEEE Software*, *19*(2), 20–22.

Hutchinson, J., Rouncefield, M. & Whittle, J. (2011a). Model-driven engineering practices in industry. Teoksessa *Proceedings of the 33rd International Conference on Software Engineering , ICSE'11* (633–642). New York: ACM.

Hutchinson, J., Whittle, J., Rouncefield, M. & Kristoffersen, S. (2011b). Empirical assessment of MDE in industry. Teoksessa *Proceedings of the 33rd International Conference on Software Engineering , ICSE'11* (471–480). New York: ACM.

Hutchinson, J., Whittle, J. & Rouncefield, M. (2014). Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure. *Science of Computer Programming*, *89*, 144–161.

Härer, F. & Fill, H. G. (2020). Past Trends and Future Prospects in Conceptual Modeling-A Bibliometric Analysis. Teoksessa *International Conference on Conceptual Modeling* (34–47). Cham: Springer International Publishing.

Iivari, J. (1995). Factors affecting perceptions of CASE effectiveness. *European Journal of Information Systems*, *4*(3), 143–158.

International Organization for Standardization. (2005). *Information technology -- Open Distributed Processing -- Unified Modeling Language (UML)* (ISO/IEC Standard No. 19501:2005). Haettu osoitteesta https://www.iso.org/standard/32620.html

Jacobson, I., Christerson, M., Jonsson, P. & Overgaard, G. (1992.) *Object-Oriented Software Engineering: A use case driven approach*. Reading, MA: Addison-Wesley.

Jacobson, I., Booch, G. & Rumbaugh, J. (1999.) *The Unified Software Development Process*. Reading, MA: Addison-Wesley.

Jackson, M. (1983). *System Development*. Englewood Cliffs: Prentice-Hall.

Kassab, M., Mazzara, M., Lee, J. & Succi, G. (2018). Software architectural patterns in practice: an empirical study. *Innovations in Systems and Software Engineering*, *14*(4), 263–271.

Kassab, M., Neill, C. & Laplante, P. (2014). State of practice in requirements engineering: contemporary data. *Innovations in Systems and Software Engineering*, *10*(4), 235–241.

Kelly, S. & Tolvanen, J. P. (2008). *Domain-specific modeling: enabling full code generation*. Hoboken, NJ: John Wiley & Sons.

Kobryn, C. (2002). Will UML 2.0 be agile or awkward?. *Communications of the ACM*, *45*(1), 107–110.

Koskimies, K., Koskinen, J., Maunumaa, M., Peltonen, J., Selonen, P., Siikarla, M. & Systä, T. (2004). UML työvälineenä ja tutkimuskohteena. *Tietojenkäsittelytiede*, *21*, 19–51.

Krogstie, J., Lindland, O. I. & Sindre, G. (1995). Towards a deeper understanding of quality in requirements engineering. Teoksessa J. Iivari, K. Lyytinen & M. Rossi (toim.), *Advanced Information Systems Engineering, CAiSE 1995, Lecture Notes in Computer Science, vol 932* (82–95). Berlin: Springer-Verlag.

Krogstie, J., Opdahl, A. L. & Brinkkemper, S. (2007). *Conceptual Modeling in Information Systems Engineering*. Berlin: Springer-Verlag.

Kruchten, P. B. (1995). The 4+ 1 view model of architecture. *IEEE software*, *12*(6), 42–50.

Kruchten, P. (2009). Documentation of software architecture from a knowledge management perspective–design representation. Teoksessa M. A. Babar, T. Dingsøyr, P. Lago & H. van Vliet (toim.), *Software Architecture Knowledge Management* (39–57). Berlin: Springer-Verlag.

Kühne, T. (2005). What is a Model?. Teoksessa J. Bezivin & R. Heckel (toim.), *Language Engineering for Model-Driven Software Development: Dagstuhl Seminar 04101*, Schloss Dagstuhl: Internationales Begegnungs-und Forschungszentrum für Informatik (IBFI).

Kühne, T. (2006). Matters of (meta-) modeling. *Software & Systems Modeling*, *5*(4), 369–385.

Kung, C. H. & Soelvberg, A. (1986). Activity modeling and behavior modeling. Teoksessa T. W. Olle, H. G. Sol & A. A. Verrijn-Stuart (toim.), *Information System Design Methodologies: Improving the Practice* (145–171). Amsterdam: North-Holland Publishing Co.

Kvale, S. (1996). *Interviews: An introduction to qualitative research interviewing*. London: Sage.

Kästner, A., Gogolla, M. & Selic, B. (2018). From (imperfect) object diagrams to (imperfect) class diagrams: New ideas and vision paper. Teoksessa *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems* (13–22). New York: ACM.

Lange, C. F., Chaudron, M. R. & Muskens, J. (2006). In practice: UML software architecture and design description. *IEEE software*, *23*(2), 40–46.

Lankhorst, M. M., Proper, H. A. & Jonkers, H. (2010). The anatomy of the archimate language. *International Journal of Information System Modeling and Design (IJISMD)*, *1*(1), 1–32.

Lano, K. (2009). *UML 2 semantics and applications*. Hoboken, NJ: John Wiley & Sons.

Laplante, P. A. (2018). *Requirements engineering for software and systems.* (3. uud. painos). Boca Raton, FL: Taylor & Francis, CRC Press.

Latella, D., Majzik, I. & Massink, M. (1999). Towards a formal operational semantics of UML statechart diagrams. Teoksessa *Proceedings of FMOODS'99, IFIP TC6/WG6.1, 3rd International Conference on Formal Methods for Open Object-Based Distributed Systems* (331–347). Dordrecht: Kluwer Academic Publishers.

Leppänen, M. (2005). *An ontological framework and a methodical skeleton for method engineering: A contextual approach* (No. 52) (Väitöskirja). University of Jyväskylä.

Liddle, S. W. (2011). Model-driven software development. Teoksessa D. W. Embey & B. Thalheim (toim.), *Handbook of Conceptual Modeling*: Theory, Practice, and Research Challenges (17–54). Berlin: Springer-Verlag.

Liebel, G., Marko, N., Tichy, M., Leitner, A. & Hansson, J. (2018). Model-based engineering in the embedded systems domain: an industrial survey on the state-of-practice. *Software & Systems Modeling*, *17*(1), 91–113.

Lilly, S. (2000). How to avoid use-case pitfalls. *Software Development*, *8, 10*, 40–45.

Lindland, O. I., Sindre, G. & Solvberg, A. (1994). Understanding quality in conceptual modeling. *IEEE software*, *11*(2), 42–49.

Long, T. & Johnson, M. (2000). Rigour, reliability and validity in qualitative research. *Clinical effectiveness in nursing*, *4*(1), 30–37.

Lukyanenko, R. (2018). Rethinking the role of conceptual modeling in the introductory IS curriculum. Teoksessa *Proceedings of the 39th International Conference on Information Systems (ICIS 2018)* (1–9), San Francisco, CA, December 13-16, 2018.

Lyytinen, K. (1987). A taxonomic perspective of information systems development: theoretical constructs and recommendations. Teoksessa R. J. Boland & R. A. Hirscheim (toim.), *Critical issues in information systems research* (3–41). Chichester: John Wiley & Sons.

Maes, A. & Poels, G. (2007). Evaluating quality of conceptual modelling scripts based on user perceptions. *Data & Knowledge Engineering*, *63*(3), 701–724.

Malavolta, I., Lago, P., Muccini, H., Pelliccione, P. & Tang, A. (2012). What industry needs from architectural languages: A survey. *IEEE Transactions on Software Engineering*, *39*(6), 869–891.

Martin, J. (1991). *Rapid application development*. New York: Macmillan Publishing.

Mellor, S. J., Mellor, S. & Balcer, M. J. (2002). *Executable UML: a foundation for model-driven architecture*. Boston, MA: Addison-Wesley.

Mendling, J., Reijers, H. A. & Recker, J. (2010). Activity labeling in process modeling: Empirical insights and recommendations. *Information Systems*, *35*(4), 467–482.

Meng, F., Chu, D. & Zhan, D. (2010). Transformation from Data Flow Diagram to UML2.0 activity diagram. Teoksessa *2010 IEEE International Conference on Progress in Informatics and Computing (PIC 2010)* (Vol. 2, 1010–1014), Shanghai, China, December 10-12, 2010.

Mernik, M., Heering, J. & Sloane, A. M. (2005). When and how to develop domain-specific languages. *ACM computing surveys (CSUR)*, *37*(4), 316–344.

Merriam, S. (1995). What Can You Tell From An N of l?: Issues of validity and reliability in qualitative research. *PAACE Journal of lifelong learning*, *4*, 50–60.

Messe, N., Belloir, N., Chiprianov, V., El-Hachem, J., Fleurquin, R. & Sadou, S. (2020). An asset-based assistance for secure by design. Teoksessa *2020 27th Asia-Pacific Software Engineering Conference (APSEC)* (178–187), Singapore, Singapore, December 1-4, 2020.

Miles, M. B. & Huberman, A. M. (1994). *Qualitative data analysis: An expanded sourcebook* (2. uud. painos). Thousand Oaks: Sage Publications.

Mohagheghi, P., Gilani, W., Stefanescu, A. & Fernandez, M. A. (2013). An empirical study of the state of the practice and acceptance of model-driven engineering in four industrial cases. *Empirical Software Engineering*, *18*(1), 89–116.

Moody, D. L. (2005). Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. *Data & Knowledge Engineering*, *55*(3), 243–276.

Moody, D. (2009). The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. *IEEE Transactions on software engineering*, *35*(6), 756–779.

Muchandi, V. (2007). *Applying 4+ 1 view architecture with uml 2* [White paper]. FCG Software Services, Sparx Systems. Haettu osoitteesta https://sparxsystems.com/downloads/whitepapers/FCGSS_US_WP_Applying_4+1_w_UML2.pdf

Mussbacher, G., Amyot, D., Breu, R., Bruel, J. M., Cheng, B. H., Collet, P., Combemale, B., France, R. B., Heldal R., Hill, J., Kienzle, J., Schöttle, M., Steimann, F., Stikkolorum, D. & Whittle, J. (2014). The relevance of model-driven engineering thirty years from now. Teoksessa *International Conference on Model Driven Engineering Languages and Systems* (183–200). Cham: Springer International Publishing.

Myers, M. D. & Newman, M. (2007). The qualitative interview in IS research: Examining the craft. *Information and organization*, *17*(1), 2–26.

Mylopoulos, J. (1992). Conceptual modelling and Telos. Teoksessa P. Loucopoulos & R. Zicari (toim.), *Conceptual Modelling, Databases, and CASE: an Integrated View of Information System Development* (49–68). New York: John Wiley & Sons.

Nair, S., De La Vara, J. L., Sabetzadeh, M. & Briand, L. (2014). An extended systematic literature review on provision of evidence for safety certification. *Information and Software Technology*, *56*(7), 689–717.

Neill, C. J. & Laplante, P. A. (2003). Requirements engineering: the state of the practice. *IEEE software*, *20*(6), 40–45.

Noble, H. & Smith, J. (2015). Issues of validity and reliability in qualitative research. *Evidence-based nursing*, *18*(2), 34–35.

Nugroho, A. & Chaudron, M. R. (2008). A survey into the rigor of UML use and its perceived impact on quality and productivity. Teoksessa *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '08)* (90–99). New York: ACM.

Oei, J. L. H., van Hemmen, L. J. G. T., Falkenberg, E. D. & Brinkkemper, S. (1992). *The Meta Model Hierarchy: A Framework for Information Systems Concepts and Techniques* (Technical Report No. 92–17). Katholieke Universiteit Nijmegen, Department of Informatics, Faculty of Mathematics and Informatics.

Olivé, A. (2005). Conceptual schema-centric development: a grand challenge for information systems research. Teoksessa O. Pastor & J. Falcão e Cunha (toim.), *Proceedings of the 17th International Conference on Advanced Information Systems Engineering (CAiSE 2005)*, (1–15). Berlin: Springer-Verlag.

Olivé, A. (2007). *Conceptual modeling of information systems*. Berlin: Springer-Verlag.

OMG. (2005). Unified Modeling Language: Superstructure version 2.0. Haettu osoitteesta https://www.omg.org/spec/UML/2.0

OMG. (2014). Business Process Model and Notation (BPMN™) Version 2.0.2. Haettu osoitteesta https://www.omg.org/spec/BPMN/

OMG. (2016). OMG Meta Object Facility (MOF) Core Specification Version 2.5.1. Haettu osoitteesta https://www.omg.org/spec/MOF

OMG. (2017). OMG® Unified Modeling Language® (OMG UML®) Version 2.5.1. Haettu osoitteesta http://www.omg.org/spec/UML/2.5.1

OMG. (2019). OMG specifications go through ISO fast-track. Haettu 5.5.2019 osoitteesta https://www.omg.org/iso/index.htm

Ozkaya, M. (2018). Do the informal & formal software modeling notations satisfy practitioners for software architecture modeling?. *Information and Software Technology*, *95*, 15–33.

Parsons, J. & Cole, L. (2005). What do the pictures mean? Guidelines for experimental evaluation of representation fidelity in diagrammatical conceptual modeling techniques. *Data & Knowledge Engineering*, *55*(3), 327–342.

Parsons, J. & Wand, Y. (1997). Using objects for systems analysis. *Communications of the ACM*, *40*(12), 104–110.

Partridge, C., Gonzalez-Perez, C. & Henderson-Sellers, B. (2013). Are conceptual models concept models?. Teoksessa W. Ng, V. C. Storey & J. Trujillo (toim.), *Proceedings of the 32th International Conference on Conceptual Modeling, ER 2013,* (96–105). Berlin: Springer-Verlag.

Petre, M. (2013). UML in practice. Teoksessa *Proceedings of the 2013 International Conference on Software Engineering (ICSE'13),* (722–731). Piscataway, NJ: IEEE Press.

Petre, M. (2014). "No shit" or "Oh, shit!": responses to observations on the use of UML in professional practice. *Software & Systems Modeling*, *13*(4), 1225–1235.

Pilone, D. & Pitman, N. (2005). *UML 2.0 in a Nutshell*. Sebastopol, CA: O'Reilly Media.

Platt, R. & Thompson, N. (2019). The Past, Present, and Future of UML. Teoksessa *Advanced Methodologies and Technologies in Network Architecture, Mobile Computing, and Data Analytics* (1452–1460). Hershey, PA: IGI Global.

Pohl, K. (1993). The three dimensions of requirements engineering. Teoksessa C. Rolland, F. Bodart & C. Cauvet (toim.), *Proceedings of the 5th International Conference on Advanced Information Systems Engineering (CAiSE'93)* (275–292). Berlin: Springer-Verlag.

Pohl, K. (2010). *Requirements engineering: Fundamentals, principles, and techniques*. Berlin: Springer-Verlag.

Pohl, K. & Rupp, C. (2015). *Requirements engineering fundamentals: a study guide for the certified professional for requirements engineering exam - foundation level - IREB compliant*. (2. uud. painos). Santa Barbara, CA: Rocky Nook.

Pohl, K. & Ulfat-Bunyadi, N. (2013). The three dimensions of requirements engineering: 20 years later. Teoksessa J. Bubenko, J. Krogstie, O. Pastor, B. Pernici, C. Rolland & A. Sølvberg (toim.), *Seminal Contributions to Information Systems Engineering: 25 Years of CAiSE* (81–87). Berlin: Springer-Verlag.

Pressman, R. S. (2005). *Software engineering: a practitioner's approach*. (6. uud. painos). New York: McGraw-Hill.

Pressman, R. S. (2010). *Software Engineering: A Practitioner's Approach*. (7. uud. painos). New York: McGraw-Hill.

Pressman, R. S. & Maxim, B. R. (2015). *Software Engineering: A Practitioner's Approach*. (8. uud. painos). New York: McGraw-Hill.

Ramnath, S. & Dathan, B. (2011). *Object-Oriented Analysis and Design*. London: Springer London.

Recker, J. C. (2008). BPMN modeling–who, where, how and why. *BPTrends*, *5*(3), 1–8.

Recker, J. (2012). "Modeling with tools is easier, believe me" — The effects of tool functionality on modeling grammar usage beliefs. *Information Systems*, *37*(3), 213–226.

Recker, J., Indulska, M., Rosemann, M. & Green, P. (2010). The ontological deficiencies of process modeling in practice. *European Journal of Information Systems*, *19*(5), 501–525.

Recker, J. C., zur Muehlen, M., Siau, K., Erickson, J. & Indulska, M. (2009). Measuring method complexity: UML versus BPMN. Teoksessa *Proceedings of the 15th Americas Conference on Information Systems (AMCIS 2009)* (1–9), San Francisco, California, August 6-9, 2009.

Reggio, G., Leotta, M. & Ricca, F. (2014). Who knows/uses what of the UML: a personal opinion survey. Teoksessa J. Dingel, W. Schulte, I. Ramos, S. Abrahão & E. Insfran (toim.), *Proceedings of the 17th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2014)* (149–165). Cham: Springer International Publishing.

Reggio, G., Leotta, M., Ricca, F. & Clerissi, D. (2013). What are the used UML diagrams? A Preliminary Survey. Teoksessa M. Chaudron, M. Genero, S. Abrahão & L. Pareto (toim.), *Proceedings of the 3rd International Workshop on Experiences and Empirical Studies in Software Modeling (EESSMod 2013 co-located with MoDELS 2013)* (3–12). CEUR Workshop Proceedings. Haettu osoitteesta http://ceur-ws.org/Vol-1078/paper1.pdf

Reggio, G., Leotta, M., Ricca, F. & Clerissi, D. (2015). What are the used UML diagram constructs? A document and tool analysis study covering activity and use case diagrams. Teoksessa S. Hammoudi, L. Pires, J. Filipe & R. das Neves (toim.), *2nd International Conference on Model-Driven Engineering and Software Development, MODELSWARD 2014* (66–83). Cham: Springer International Publishing.

Riva, C. & Rodriguez, J. V. (2002). Combining static and dynamic views for architecture reconstruction. Teoksessa *Proceedings of the Sixth European*

Conference on Software Maintenance and Reengineering (47–55). Budapest: IEEE Computer Society.

Robeer, M., Lucassen, G., van der Werf, J. M. E., Dalpiaz, F. & Brinkkemper, S. (2016). Automated extraction of conceptual models from user stories via NLP. Teoksessa *Proceedings of the 24th IEEE International Requirements Engineering Conference (RE'16)* (196–205), Beijing, China, September 12-16, 2016.

Rolland, C. & Cauvet, C. (1992). Trends and perspectives in conceptual modelling. Teoksessa P. Loucopulous & R. Zicari (toim.), *Conceptual Modelling, Databases and CASE: An Integrated View on Information Systems Development* (27–48). New York: John Wiley & Sons.

Roth, S., Hauder, M., Farwick, M., Breu, R. & Matthes, F. (2013). Enterprise Architecture Documentation: Current Practices and Future Directions. Teoksessa R. Alt & B. Franczyk (toim.), *Proceedings of the 11th International Conference on Wirtschaftsinformatik (WI2013)* (911–925). Leipzig: Universität Leipzig.

Royce, W. W. (1970). Managing the Development of Large Software Systems-Concepts and Techniques. Teoksessa *the Proceedings of IEEE WESCON* (1–9). Los Alamitos, CA: IEEE Computer Society.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. & Lorensen, W. E. (1991). *Object-oriented modeling and design* (Vol. 199, No. 1). Englewood Cliffs, NJ: Prentice-hall.

Rumbaugh, J., Jacobson, I. & Booch, G. (1998). *The unified modeling language reference manual*. Reading, MA: Addison-Wesley.

Rumpe, B. (2016). *Modeling with UML: Language, Concepts, Methods*. Cham: Springer International Publishing.

Scanniello, G., Gravino, C. & Tortora, G. (2010). Investigating the role of UML in the software modeling and maintenance-a preliminary industrial survey. Teoksessa *Proceedings of the 12th International Conference on Enterprise Information Systems (ICEIS'2010)* (141–148). Setúbal: SciTePress.

Schober, M. F. (2018). The future of face-to-face interviewing. *Quality Assurance in Education, 26*(2), 290–302.

Schwaber, K. & Beedle, M. (2002). *Agile software development with Scrum* (Vol. 1). Upper Saddle River, NJ: Prentice Hall.

Schön, E. M., Thomaschewski, J. & Escalona, M. J. (2017). Agile Requirements Engineering: A systematic literature review. *Computer Standards & Interfaces*, *49*, 79–91.

Seidewitz, E. (2003). What models mean. *IEEE software*, *20*(5), 26–32.

Selic, B. (2003). The pragmatics of model-driven development. *IEEE software, 20*(5), 19–25.

Selic, B. (2012). What will it take? A view on adoption of model-based methods in practice. *Software & Systems Modeling*, *11*(4), 513–526.

Shanks, G., Tansley, E. & Weber, R. (2003). Using ontology to validate conceptual models. *Communications of the ACM*, *46*(10), 85–89.

Siau, K. (1999). Information modeling and method engineering: A psychological perspective. *Journal of Database Management, 10*(4), 44–50.

Siau, K. (2002). The psychology of information modeling. Teoksessa K. Siau (toim.), *Advanced Topics in Database Research, Vol. 1* (106–118). Hershey, PA: Idea Group Publishing.

Siau, K. (2004). Informational and computational equivalence in comparing information modeling methods. *Journal of Database Management (JDM)*, *15*(1), 73–86.

Siau, K. & Loo, P. P. (2006). Identifying difficulties in learning UML. *Information Systems Management*, *23*(3), 43–51

Siau, K. & Rossi, M. (2001). Information modeling in the Internet age - challenges, issues and research directions. . Teoksessa K. Siau & M. Rossi (toim.), *Information Modeling in the New Millennium* (1–8). Hershey, PA: Idea Group Publishing.

Siau, K. & Rossi, M. (2011). Evaluation techniques for systems analysis and design modelling methods–a review and comparative analysis. *Information Systems Journal, 21*(3), 249–268.

Siau, K., Wand, Y. & Benbasat, I. (1996). Evaluating information modeling methods - A cognitive perspective. Teoksessa K. Siau & Y. Wand (toim.), *Proceedings of the Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD '96) (*1–13), Crete, Greece, May 20-21, 1996.

Siegel, J. (2005). Introduction to OMG's Unified Modeling Language™ (UML®). Haettu 12.4.2019 osoitteesta https://www.uml.org/what-is-uml.htm

Silva, A. R. (2015). Model-driven engineering: A survey supported by the unified conceptual model. *Computer Languages, Systems & Structures*, *43*, 139–155.

Sinz, E. J. (2019). On the Evolution of Methods for Conceptual Information Systems Modeling. Teoksessa K. Bergener, M. Rackers & A. Stein (toim.),

The Art of Structuring: Bridging the Gap Between Information Systems Research and Practice (137–144). Cham: Springer International Publishing.

Sommerville, I. (2016). *Software engineering* (10. uud. painos). Boston: Pearson.

Spivey, J. M. & Abrial, J. R. (1992). *The Z notation* . Hemel Hempstead: Prentice Hall.

Stachowiak, H. (1973). *Allgemeine Modelltheorie*. Wien: Springer-Verlag.

Steinmüller, W. (1993). *Informationstechnologie und Gesellschaft: Einführung in die Angewandte Informatik*. Darmstadt: Wissenschaftliche Buchgesellschaft.

Storey, V. C., Trujillo, J. C. & Liddle, S. W. (2015). Research on conceptual modeling: Themes, topics, and introduction to the special issue. *Data & Knowledge Engineering*, (98), 1–7.

Störrle, H. (2017). How are Conceptual Models used in Industrial Software Development? A Descriptive Survey. Teoksessa E. Mendes, K. Petersen & S. Counsell (toim.), *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering (EASE'17)* (160–169). New York: ACM.

Thalheim, B. (2018). Conceptual model notions–a matter of controversy: Conceptual modelling and its lacunas. *Enterprise Modelling and Information Systems Architectures International Journal on Conceptual Modeling, 13*, 9–27.

Tolvanen, J. P. (1998). *Incremental Method Engineering with Modeling Tools: Theoretical Principles and Empirical Evidence* (Väitöskirja). Jyväskylä Studies in Computer Science, Economics and Statistics 47. Jyväskylän yliopisto.

Tolvanen, J. P. & Kelly, S. (2016). Model-driven development challenges and solutions: Experiences with domain-specific modelling in industry. Teoksessa *Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2016)* (711–719). Setúbal: SciTePress.

Tomassetti, F., Torchiano, M., Tiso, A., Ricca, F. & Reggio, G. (2012). Maturity of software modelling and model driven engineering: A survey in the Italian industry. Teoksessa *Proceedings of the 16th International Conference on Evaluation & Assessment in Software Engineering (EASE 2012)* (91–100), Ciudad Real, Spain, May 14-15, 2012.

Torchiano, M., Tomassetti, F., Ricca, F., Tiso, A. & Reggio, G. (2011). Preliminary findings from a survey on the MD state of the practice. In *Proceedings of the 5th International Symposium on Empirical Software Engineering and Measurement (ESEM 2011)* (372–375). Washington, DC: IEEE Computer Society.

Torchiano, M., Tomassetti, F., Ricca, F., Tiso, A. & Reggio, G. (2013). Relevance, benefits, and problems of software modelling and model driven techniques—A survey in the Italian industry. *Journal of Systems and Software*, *86*(8), 2110–2126.

Tuomi, J. & Sarajärvi, A. (2018). *Laadullinen tutkimus ja sisällönanalyysi* (Uudistettu laitos). Helsinki: Kustannusosakeyhtiö Tammi.

van Vliet, H. (2007). *Software engineering: principles and practice*. Wiley. Haettu osoitteesta http://15.222.11.163/wp-content/uploads/2020/01/Software-Engineering-Principles-and-Practice.pdf

Verdonck, M., Gailly, F., Pergl, R., Guizzardi, G., Martins, B. & Pastor, O. (2019). Comparing traditional conceptual modeling with ontology-driven conceptual modeling: an empirical study. *Information Systems*, *81*, 92–103.

Vetro, A., Bohm, W. & Torchiano, M. (2015). On the benefits and barriers when adopting software modelling and model driven techniques-an external, differentiated replication. Teoksessa *2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2015)* (1–4), Beijing, China, October 22-23, 2015.

Wand, Y., Monarchi, D. E., Parsons, J. & Woo, C. C. (1995). Theoretical foundations for conceptual modelling in information systems development. *Decision Support Systems, 15*(4), 285–304.

Wand, Y. & Weber, R. (2002). Research commentary: information systems and conceptual modeling—a research agenda. *Information systems research*, *13*(4), 363–376.

Weaver, P. L., Lambrou, N. & Walkley, M. (1998). *Practical SSADM Version 4+: a complete tutorial guide*. (2. uud. painos). Harlow: Pearson Education.

Weber, R. (2003). Conceptual modelling and ontology: Possibilities and pitfalls. *Journal of Database Management (JDM)*, *14*(3), 1–20.

White, R. T. (1994). Conceptual and conceptional change. *Learning and instruction*, *4*(1), 117–121.

Whittle, J., Hutchinson, J. & Rouncefield, M. (2013). The state of practice in model-driven engineering. *IEEE software*, *31*(3), 79–85.

Wijers, G. M. (1991). *Modelling support in information systems development* (Väitöskirja). Delft University of Technology, Amsterdam.

Wirfs-Brock, R., Wilkerson, B. & Wiener, L. (1990). *Designing object-oriented software.* Englewood Cliffs, NJ: Prentice-Hall.

Wolcott, H. F. (1995). *The art of fieldwork*. Walnut Creek, CA: AltaMira Press.

Wrycza, S. & Marcinkowski, B. (2007). A light version of UML 2: Survey and outcomes. Teoksessa *Proceedings of the 2007 Computer Science and IT Education Conference* (739–749). University of Technology Mauritius Press.

Yu, E., Strohmaier, M. & Deng, X. (2006). Exploring intentional modeling and analysis for enterprise architecture. Teoksessa *Proceedings of the EDOC 2006 Workshop on Trends in Enterprise Architecture Research (TEAR 2006)* (32–32). Hong Kong: IEEE Computer Society.