

Talonrakentamisen projektinohjausprosessi ja sen tehostaminen

Nico Liljestrand

Perustieteiden korkeakoulu

Diplomityö, joka on jätetty opinnäytteenä tarkastettavaksi
diplomi-insinöörin tutkintoa varten Espoossa 27.05.2019.

Työn valvoja

Risto Sarvas, Professor Aalto
University

Työn ohjaaja

Ari Pennanen, Adjunct professor
Tampere Technical University,
partner Haahtela Group



Aalto-yliopisto
Perustieteiden
korkeakoulu

Copyright © 2019 Nico Liljestrand

Tekijä Nico Liljestrand

Työn nimi Talonrakentamisen projektinohjausprosessi ja sen tehostaminen

Koulutusohjelma Informaatioverkostot

Pääaine Information Networks

Pääaineen koodi SCI3047

Työn valvoja Risto Sarvas, Professor Aalto University

Työn ohjaaja Ari Pennanen, Adjunct professor Tampere Technical University,
partner Haahtela Group

Päivämäärä 27.05.2019

Sivumäärä 65

Kieli Suomi

Tiivistelmä

Talonrakentamisen projektinohjausprosessi on Haahtela-yhtiöiden pääliiketoiminnan, rakennusten rakennuttamisen, tärkein prosessi. Talonrakentamisen projektinohjausprosessi koostuu kolmesta vaiheesta: ohjelmointivaiheesta, suunnitteluvaiheesta sekä rakentamisesta työmaalla. Tutkimuksessa keskitytään ohjelmointi- ja suunnitteluvaiheiden prosessien tutkimiseen siitä näkökulmasta, miten eri vaiheiden prosessit toimivat tällä hetkellä ja miten prosesseja voisi muokata toimimaan vieläkin paremmin. Tutkimuksen pohjana käytetään LEAN-ajattelun peruseriaatteita, jossa nykyisten prosessien selvittämisen jälkeen yritetään tunnistaa prosesseista hukkaa ja lopuksi esitellään ratkaisuehdotuksia hukkien poistamiseksi prosesseista. Ohjelmointi- ja suunnitteluvaiheiden prosesseissa on mukana useita eri toimijoita (asiakas, suunnittelijat, projektinjohto) ja vaiheiden prosessit ovat kompleksisia. Tutkimuksessa tutkitaan näiden vaiheiden prosesseja kompleksisten prosessien ja systeemien ymmärtämisen näkökulmasta. Ohjelmointi- ja suunnitteluvaiheiden kompleksisuus on luonteeltaan induktiivista kompleksisuutta, joka tarkoittaa sitä, ettei kysymyksiin ole olemassa yksikäsitteistä ratkaisua, vaan ongelma voidaan ratkaista monella eri tavalla. Tällöin on tärkeää, että ongelmaan löydetään mahdollisimman hyvin ongelman ratkaiseva ratkaisu. Ohjelmointi- ja suunnitteluvaiheiden prosessit ovat iteratiivisia ja molemmissa tehdään jonkin verran manuaalista työtä. Yksi olennaisimpia tutkimuskohteita onkin, miten tuota manuaalista työtä olisi mahdollista vähentää automatisoinnilla ja millaisia muutoksia prosessiin automatisoinnin käyttöön ottaminen vaatii. Tätä varten ohjelmointi- ja suunnitteluvaiheiden prosesseja verrataan yleisten tietomallivaatimusten määrittelemiін tiedon laatu- ja tarkkuustasoihin prosessien eri vaiheissa ja pyritään löytämään kohtia, joissa Haahtela-yhtiöiden projektinohjausprosessi ja yleiset tietomallivaatimukset käsittelevät samoja tietoja eri tavoin. Tämä on tärkeää, sillä tietomallien lukeminen automaattisesti Haahtelan TAKU[®] BIM-järjestelmään nopeuttaisi varsinkin suunnitteluvaiheen iteraatioita ja vähentäisi manuaalisen työn aiheuttamia ihmisen tekemiä virheitä esimerkiksi tiedon kopioinnissa. IFC tietomallistandardi voisi toimia yhtenä mahdollisena tietoformaattina projektinohjausprosessin toimijoiden välisessä tiedon jakamisessa.

Avainsanat TAKU[®], BIM, LEAN, Project management, Projektinohjaus, Suunnittelunohjaus

Author Nico Liljestrand

Title Talonrakentamisen projektinohjausprosessi ja sen tehostaminen

Degree programme Informaatioverkostot

Major Information Networks

Code of major SCI3047

Supervisor and advisor Risto Sarvas, Professor Aalto University

Date 27.05.2019

Number of pages 65

Language Finnish

Abstract

Steering building construction is the main component of project management in Haahtela Group's core business: managing building construction. The steering process consists of three phases: programming, design and construction on site. This research focuses on programming and design phases and the main target is to find out how the processes are currently working and after that how the processes could be improved. LEAN thinking is utilized to reveal waste in the processes and what causes it. By removing the cause of the waste, it is possible to achieve more efficient processes, which is one of the main reasons behind this research. Programming and design phases have multiple stakeholders (customer, designers, project management) and the processes are complex. The complexity in programming and design phases' processes is inductive complexity. Inductive problems have no clear right answers and many solutions would be good enough to solve the problem, but some of the solutions are better than others. In addition, the processes are iterative, which means that even small improvements will have cumulative effect over time. The processes are compared to common BIM requirements that are the standard for creating BIMs in Finland. Common BIM requirements define certain standard for how specific the information model has to be in each phase and their approach is compared to how the same information is handled in Haahtela TAKU[®] BIM system. Utilizing IFC standard for information flows from designers to TAKU[®] could be done automatically. This could result in faster iterations and it would reduce possibility for human errors in the process. Utilizing IFC would also help to control the information flow from designers to project management and vice versa during the design phase of the process.

Keywords TAKU[®], BIM, LEAN, Project management, Projektinohjaus, Suunnittelunohjaus

Sisältö

Tiivistelmä	3
Tiivistelmä (englanniksi)	4
Sisältö	5
Käsitteet ja lyhenteet	7
1 Johdanto	9
2 Tutkimusaineisto ja -menetelmät	11
2.1 Tutkimusaineisto	11
2.2 Tutkimusmenetelmät	12
2.3 Tutkimuskysymykset ja -hypoteesi	12
3 Teoreettinen tausta	14
3.1 Talonrakentamisen projektinohjausprosessi	14
3.1.1 Ohjelmointivaihe	15
3.1.2 Suunnitteluvaihe	16
3.1.3 Rakentaminen työmaalla - Rakennusvaihe	19
3.2 Kompleksisuus	20
3.2.1 Deduktiivinen kompleksisuus	20
3.2.2 Induktiivinen kompleksisuus	21
3.2.3 Kompleksiset järjestelmät	22
3.3 LEAN ja hukka	24
3.4 Kompleksisten prosessien ohjaus	27
3.4.1 Axiomatic design	28
3.4.2 CK-teoria	30
3.5 Yleiset tietomallivaatimukset	31
3.5.1 Vaatimusmalli	33
3.5.2 Ehdotussuunnittelu	34
3.5.3 Yleissuunnittelu	34
3.5.4 Toteutussuunnittelu	34
3.6 Tietovirrat	35
3.7 Tietoformaatit tietovirtojen hallinnassa	37
3.7.1 Ohjelmointivaihe ja COBie	37
3.7.2 Suunnitteluvaihe ja standardoidut tietomalliformaatit	38
4 Tutkimus	40
4.1 Haahtelan ohjausmalli ja kompleksisuus	40
4.1.1 Tausta	41
4.1.2 Prosessit	44
4.1.3 Kehitys	44
4.2 Tietovirrat ja niiden hallinta Haahtelan prosesseissa	45

	6
4.2.1 Ohjelmointivaihe	45
4.2.2 Suunnitteluvaihe	48
5 Tulokset	52
5.1 Tietovirrat ja tiedon oikea-aikaisuus	52
5.2 Haahtelan ohjausteorian haasteet ja kehitysideat	52
5.2.1 Haahtelan ohjausmalli ja hukka	53
5.2.2 Hukan hallitseminen	54
6 Yhteenveto	61
Viitteet	64

Käsitteet ja lyhenteet

Käsitteet

Haahtela-malli	Haahtela-yhtiöiden kehittämä kolmesta vaiheesta koostuva talonrakentamisen projektinohjausprosessi
Hukka	LEAN-ajattelussa turhat tai kompleksisuutta aiheuttavat toiminnot ovat hukkaa
Kompleksisuus	Kompleksisuus mittaa epätietoisuutta jonkin järjestelmän osan toiminnasta tai niiden vaikutuksista toisiinsa
Käyttöaste	Tarkoittaa sitä osuutta ajasta, jonka jonkin tila on sen tarkoituksen mukaisessa käytössä
LEAN	Ajattelutapa, jonka tavoitteena on tehostaa tuotantoa ja prosesseja
Odotettu kustannus	Asiakkaan tilaluettelon arvioitu kustannus (rakennusprojektin arvioitu hinta ohjelmointivaiheessa)
Ohjausväli	Suunnitelmia arvioidaan laadun ja hinnan perusteella. Ohjausvälillä tarkoitetaan tiettyä hinta-haarukkaa, jolta valitaan laadukkaimmat suunnitelmat jatkokehitettäväksi
Ohjelmointivaihe	Haahtela-mallin ensimmäinen vaihe, jonka aikana asiakkaan tarpeet kuvataan toiminnoiksi ja edelleen tilaluetteloksi
Rakennus	Tiloista koostuva toimintojen kokonaisuus. Rakennuksessa tiloilla on tietty geometria ja rakennuksella on sijainti
Rakennusvaihe	Haahtela-mallin kolmas vaihe, jossa rakennus rakennetaan suunnitteluvaiheen suunnitelmien pohjalta
Sallittu kustannus	Asiakkaan määrittämä maksimikustannus rakennushankkeelle
Suunnitelma	Suunnitteluvaiheessa arkkitehdin tekemä suunnitelma rakennuksen muodosta ja materiaaleista
Suunnitteluvaihe	Haahtela-mallin toinen vaihe, jossa tilaluettelon pohjalta luodaan rakennuksen suunnitelma, rakennuksen tulevassa ympäristössä
Systemi	Erilaisten toimintojen kokonaisuus, jolla voidaan suorittaa jokin tietty toiminto
Tavoitehintaa	Suunnitelman suurin mahdollinen kokonaishinta
Tila	Rakennuskomponenttien rajaama alue rakennuksessa, jossa voidaan tehdä tilalle tyyppillisiä toimintoja
Tilaluettelo	Ohjelmointivaiheen päätteeksi saatava luettelo rakennuksen sisältämistä tiloista ja toiminnoista

Lyhenteet

BIM	Building Information Modelling - talonrakentamisessa käytettävä konsepti, jossa rakentamisen keskiössä on rakennuksen tietomalli
COBie	Construction Operations Building Information Exchange, Ohjelmointivaiheessa käytettävissä oleva tiedon tallennusstandardi
IFC	Industrial Foundation Classes eli rakennuksen tietomallin tallennusformaattistandardi
JIT	Just in Time. Ennen LEAN termiä käytössä ollut termi.
LVI-järjestelmä	Lämpö-, vesi-, ja ilmastointijärjestelmä
TAKU [®]	Talonrakentamisen kustannustieto - Haahtela-yhtiöiden BIM ohjelmisto talonrakentamisen kustannusten arviointiin
YTV	Yleiset tietomallivaatimukset

1 Johdanto

Haahtela-yhtiöissä on tutkittu talonrakentamisen ohjausta jo 1980-luvulta lähtien. Tutkimuksen tavoitteena on rakennushankkeiden talouden hallinnan (toiminnallisuus ja kustannukset) laadun parantaminen. Hankkeiden hallintaa varten Haahtela-kehitys Oy on kehittänyt talouden hallinnan teorian sekä tuottanut tietomalliohjelmiston TAKU[®]. TAKU[®] toimii perustana kaikelle Haahtela-yhtiöissä tapahtuvalle projekti-ohjaukselle talonrakentamisen toimialalla. Talonrakentamisen kokonaisprosessia tarkastellaan kolmessa vaiheessa: ohjelmointivaiheessa, suunnitteluvaiheessa sekä rakentamisessa työmaalla.

Ohjelmointivaiheella tarkoitetaan asiakkaan tahtotilan määrittystä ennen rakennuksen suunnittelua. Yksinkertaistettuna tahtotilassa määritellään toiminnalliset tarpeet, hankkeen budjetti sekä toteutusaikataulu. Haahtela-yhtiöiden projektien hallinnan lähtötietona on asiakkaan liiketoiminta ja toiminnot. Näillä toiminnoilla on siis aina jokin asiakkaan liiketoiminnan kannalta tärkeä rooli; tilaa ei ole syytä rakentaa, mikäli siellä ei toteuteta jotakin tarkoituksenmukaista toimintoa. Rakennuttajana Haahtela-yhtiöt pyrkivät siis ensin selvittämään nämä asiakkaan liiketoiminnan kannalta tärkeät tarpeet ja tämän jälkeen kuvaamaan ne erilaisina toimintoina ja myöhemmin toiminnot tiloina. Tässä vaiheessa selviää rakennusprojektin kustannusten kannalta oleellisia tunnuslukuja kuten: sallittu kustannus, odotettu kustannus sekä tavoitehinta. Sallittu kustannus tarkoittaa asiakkaan määrittelemää maksimihintaa projektille ja odotettu kustannus puolestaan laskettua arvioita asiakkaan tarvitsemien toimintojen ja tilojen toimintojen hankkimiseksi. Tavoitehinta määräytyy sallitun kustannuksen sekä odotetun kustannuksen tuloksena, mikäli siitä päästään sopuun asiakkaan ja projektinjohdon välillä.

Tilat eivät kuitenkaan yksinään määrittele valmista rakennusta ja tilojen lisäksi vaaditaan myös rakennuksen sijainti, sekä geometriatiedot, jotta rakennuksen muoto ja tilojen suhteet toisiinsa tulevat esille. On tärkeää, että yhdessä toimivien tilojen sijainti toisiinsa nähden on käytön kannalta perusteltua. Esimerkiksi urheilutiloissa ei ole mielekää sijoittaa suihkuja ja pukuhuoneita kauaksi toisistaan. Tällaisia päätöksiä ei kuitenkaan tee rakennuttaja, vaan arkkitehti yhdessä tilaajan kanssa. Rakennuttajan tehtävänä on valvoa, että arkkitehdit pysyvät tavoitehinnan rajoissa ja arvioida sitä, kuinka hyvin mikäkin suunnitelma täyttää asiakkaan tarpeet.

Ohjelmointi- ja suunnitteluvaiheet ovat molemmat iteratiivisia ja projekteissa on mukana paljon toimijoita sekä muuttujia. Molemmat vaiheet ovat siksi kompleksisia. Tätä varten esitellään kompleksisuuden teoriaa sekä miten kompleksisuutta voidaan pyrkiä vähentämään ja hallitsemaan. Tarkoituksena on selvittää millaisia prosessit ovat tällä hetkellä, tutkia niitä LEAN-ajattelun mukaisesti ja pyrkiä löytämään prosesseista LEAN-ajattelun mukaista hukkaa. Lopuksi prosesseista pyritään löytämään kehitettävät kohteet löydettyihin hukkiin esitetään korjausehdotuksia.

Rakentaminen työmaalla voidaan aloittaa suunnitteluvaiheen jälkeen. Tutkimus keskittyy ohjelmointi- ja suunnitteluvaiheiden prosesseihin ja työmaavaihetta sivutaan mikäli se on kokonaisuuden kannalta tarpeellista. Itse työmaavaiheen prosesseihin ei oteta kantaa, sillä ohjelmointi- ja suunnitteluvaiheet ovat luonteeltaan erilaisia ohjattavia prosesseja. Työmaavaiheen tärkein tavoite on rakentaa rakennus ennalta

sovittuun sijaintiin suunnitteluvaiheen suunnitelmien mukaisesti.

2 Tutkimusaineisto ja -menetelmät

2.1 Tutkimusaineisto

Talonrakentamisen tutkimuksen tuloksena Haahtela-kehitys Oy on luonut ohjausteorian, jota hyödynnetään talonrakentamisen ohjelmointivaiheessa, suunnittelun ohjauksessa ja lopulta rakennustyömaalla rakentamisen ohjauksessa. Ohjausteorian perusteena on käytetty aiemmin julkaistuja teorioita kompleksisten prosessien ohjauksesta, joita mm. Pennanen(2004) [1] käsittelee väitöskirjassaan: “Workplace Planning”.

Aluksi esitellään Haahtela-kehitys Oy:n ohjausteoria. Tämän jälkeen esitellään teoriaa kompleksisuudesta ja sen vähentämisestä monimutkaisissa prosesseissa, jotka kytkeytyvät Haahtelan ohjausteoriaan. Ideana on ymmärtää projektin-ohjauksen periaatteet sekä tutkia prosessin tunnettuja ongelmakohtia ja etsiä niihin kehitysehdotuksia. Lisäksi prosessia tutkimalla yritetään löytää uusia mahdollisia kehityskohteita ja niihin mahdollisia ratkaisuja.

Teoriaosuus pitää sisällään LEAN-osuuden, jossa pyritään selvittämään yleisellä tasolla, miten prosesseja voidaan pyrkiä tehostamaan. On tärkeää ymmärtää, millä tavoilla prosessista on mahdollista löytää LEAN-ajattelun mukaista hukkaa ja kuinka hukkaa voidaan vähentää. Tällä tavoin prosessia on mahdollista tehostaa. LEAN-ajattelun myötä prosessissa täytyy keskittyä tietovirtoihin ja niiden hallintaan. Tietovirrat ovat Haahtelan ohjausmallissa sellaista informaatiota, jota liikkuu toimijalta toiselle tietynä aikana. Tietovirtojen kannalta olennaista on, että oikea tieto on saatavilla oikeaan aikaan, oikeassa paikassa. Lisäksi tiedon tulee olla tietyllä tarkkuustasolla, joka tarkentuu projektin edetessä. Tällä hetkellä Haahtelan ohjausmallissa eri osapuolet sopivat, milloin mikäkin tieto on valmista jaettavaksi.

Tietomallit ovat keskeinen osa nykypäivän rakennusprojekteja. Suunnitteluvaiheen ja rakentamisvaiheen tietomalleista käytetään yleisesti nimitystä Building Information Model eli BIM. Tietomallin tekeminen aloitetaan rakennusprojektin suunnitteluvaiheessa ja tietomallin luomista varten on yritetty luoda yleisiä tietomallivaatimuksia eli YTV (BuildingSMART, 2012)[2]. Tietomallivaatimukset määrittävät vähimmäisvaatimukset tietomallin tarkkuudelle ja tiedon määrälle suunnitteluvaiheen kussakin iteraativaiheessa. Yleiset tietomallivaatimukset selitetään teoriaosuudessa pintapuolisesti.

Myös itse tiedonsäilömisformaatti on tämän tutkimuksen kannalta tärkeää. Sen vuoksi tutkimus esittelee muutamia erilaisia tiedon tallennusstandardeja, joista osa on avoimessa käytössä ja osa yritysten sisäiseen käyttöön patentoituja. Lisäksi erilaisia formaatteja on tarjolla eri vaiheissa, kuten COBie ohjelmointivaiheessa ja IFC suunnitteluvaiheessa.

Projektinohjausprosessin tehostaminen on tärkeää kustannusten hallitsemiseksi ja rakentamisen korkean laadun ylläpitämiseksi. Talojen rakentaminen on suuria pääomia vaativa prosessi, joten pienilläkin tehokkuuden parannuksilla voidaan mahdollisesti saavuttaa suuria parannuksia. Iteratiivisissa prosesseissa nopeammat iteraatiot tuottavat säästöjä aikatauluissa tai mahdollistavat useamman iteraation samassa ajassa. Myös poistettavan hukan positiiviset vaikutukset kumuloituvat iteraatioiden

määrän kasvaessa.

2.2 Tutkimusmenetelmät

Teoriaosuus selittää Haahtelan ohjausteorian mukaisen projektinohjausprosessin talonrakentamisen alalla. Talonrakentamisen prosessin ymmärtämiseksi teoriaosuudessa käsitellään myös kompleksisuutta yleisellä tasolla, kompleksisten systeemien ymmärtämistä sekä kompleksisten prosessien ohjausta. Teoriaosuus on kirjallisuuskatsaus siitä, miten projektinohjausta tällä hetkellä tehdään Haahtela-yhtiöissä ja yleisemmin rakennusalalla.

Teoriaosuuden lisäksi tutkimuksessa on käytännön osuus, jossa selvitetään Haahtelan ohjausmallin prosessien kulku ja etsitään prosessista LEAN-ajattelun mukaista hukkaa. LEAN-ajattelua on rakennusalalla yleisesti käytetty prosessien tehostamistyökaluna, joten LEAN-ajattelun käyttäminen tutkimuksen pohjana on perusteltua. Lisäksi Käytännön osuudessa haastatellaan Haahtela-yhtiöissä talonrakentamisen projektinohjauksen ammattilaisia ja pyritään selvittämään, mitä parannettavaa prosesseissa voisi olla. Haastattelut perustuvat kirjallisuus osion perusteella löydettyihin tekijöihin. Haastattelut eivät olleet niinkään ennalta määriteltyä keskustelua, vaan enemmänkin työpajahenkisiä kokouksia, joiden tarkoituksena oli selvittää ohjelmointi- ja suunnitteluvaiheiden prosessit, sekä niihin liittyvät olennaisimmat tietovirratt. Kokouksia järjestettiin yhtensä kolme kappaletta ja niihin osallistuivat seuraavat asiantuntijat Haahtelan organisaatiosta:

- Ari Pennanen, Adjunct professor Tampere Technical University, partner Haahtela-yhtiöt
- Erkki Teittinen, Projektinjohto ja kehitys, tiiminvetäjä, Haahtela-yhtiöt
- Markus Mikkola, projektinjohto ja kehitys, Haahtela-yhtiöt

Haastattelut ovat tärkeässä roolissa Haahtelan prosessien ymmärtämiseksi, sillä prosesseja ei ole aiemmin kuvattu vastaavalla tavalla. Haastatteluja järjestettiin yhteensä kolme kappaletta ja niiden lopputuloksena saatiin myöhemmin tässä tutkimuksessa esiteltävät kaaviot tietovirroista ohjelmointi- ja suunnitteluvaiheissa.

2.3 Tutkimuskysymykset ja -hypoteesi

Tutkimuksen tavoitteena on ymmärtää miten talonrakentaminen Haahtelan ohjausteorian mukaisesti toimii tällä hetkellä ja olisiko prosesseja mahdollista kehittää entistäkin tehokkaammiksi. Tästä tavoitteesta voidaan johtaa tutkimuksen neljä tutkimuskysymystä:

1. Miten rakennushankkeen ohjausta voidaan tehostaa?
2. Mitä tietoa tarvitaan suunnittelunohjausprosessin eri vaiheissa?
3. Miten voidaan varmistaa tiedon laatu, oikeellisuus ja oikea-aikaisuus kussakin vaiheessa?

4. Millaisia formaatteja on käytössä talonrakentamiseen liittyvän tiedon tallentamiseen ja siirtoon?

Tutkimushypoteesi: Haahtelan ohjausmallin prosesseissa tehdään jonkin verran manuaalista työtä ja tietoa liikkuu paljon eri toimijoiden välillä. Tuomalla esiin LEAN-ajattelun mukaista prosessien tehostamista, voidaan parantaa rakennushankkeen ohjelmointi- ja suunnitteluvaiheiden tiedon kulkua ja oikea-aikaisuutta sekä vähentää ihmisen aiheuttamia virheitä.

3 Teoreettinen tausta

Talonrakentamisen ohjausta on tutkittu melko laajasti Suomessa ja ulkomailla viimeisten vuosikymmenien ajan. Haahtela-rakennuttaminen Oy käyttää niin kutsuttua Haahtela-mallia talonrakentamisen projektinohjauksessa. Haahtela-malli on asiakaslähtöinen ja ideana on ensin selvittää asiakkaan toimintojen kannalta olennaiset tarpeet ja tämän jälkeen ohjata asiakasta tekemään tarpeidensa täyttymisen kannalta parempia päätöksiä. Myöhemmin prosessin aikana pyritään ohjaamaan arkkitehti- ja sekä muita suunnittelijoita luomaan kustannustehokkaita ratkaisuja asiakkaan tarpeiden täyttämiseksi. Haahtelan ohjausteorian tarkoituksena on rakentaa realistinen kuva odotetuista kustannuksista ja rakennuksen ominaisuuksista. Tällainen suunnittelunohjaus on välttämätöntä rakennusprojektien onnistumiselle.

3.1 Talonrakentamisen projektinohjausprosessi

Ballard, Pennanen ja Haahtela(2011) esittelevät artikkelissaan: “Target costing and designing to targets in construction”[3] kolmivaiheisen talonrakentamisen projektinohjausprosessin, jonka pohjalta Haahtela-yhtiöt tekevät omaa projektinohjaustaan. Talojen rakentaminen on monimutkainen prosessi, joka vaatii asiakkaiden ohjaamista tarpeen määrittelyssä (toiminallinen kuvaus, kustannus, jolla toiminnallisuus halutaan saatavan sekä aikataulu), tarkkaa suunnittelua ja suunnittelijoiden ohjaamista tekemään parempia päätöksiä (Ballard et al. 2011)[3]. Talonrakentamisen ratkaisutavat ongelmat ovat monimutkaisia ja ongelmien ratkaisut eivät ole yksikäsitteisiä. Niukkanen(1980) kuvailee tutkimuksessaan: “Quality and Cost Factors in Architectural Design”, että ratkaisu ei ole ainut oikea vastaus, vaan on olemassa monia eri ratkaisuja, jotka kelpaavat ongelman ratkaisuksi[4]. Tällöin jotkin ratkaisut voivat kuitenkin olla toisia laadukkaampia tai parempia, vaikka kaikki ratkaisut sinänsä toimitusivatkin itse ongelmaan. Ratkaisu ei siis ole luonteeltaan oikea tai väärä, vaan pikemminkin hyvä tai huono (Ballard et al. 2011)[3]. Mikäli projektin ohjauksessa ei onnistuta, se saattaa kasvattaa projektin kuluja mikä puolestaan johtaa budjetin ylittymiseen. Rakennuksen rakennusvaihe voidaan jakaa kolmeen osaan: (Ballard et al. 2011)[3]

1. Ohjelmointivaihe
2. Suunnitteluvaihe
3. Rakentaminen työmaalla

Näistä osioista tämän tutkimuksen kannalta mielenkiintoisimmat ovat ohjelmointi- ja suunnitteluvaiheet, sillä näissä vaiheissa monimutkaisuus on suurinta. Ohjelmointi- ja suunnitteluvaiheet myös tuottavat uutta tietoa ja tiedon laatua on tärkeää ohjata ja haastaa (Ballard et al. 2011)[3]. Rakentaminen työmaalla ei niinkään enää tuota uutta tietoa projektin kannalta, vaan pikemminkin toteutetaan aiemmin tuotettuja suunnitelmia sekä valvotaan suunnitelmien toteutumista (Ballard et al. 2011)[3]. Rittel ja Webber(1972) ovat luoneet artikkelissaan: “Dilemmas in general theory

of planning” käsitteen vaikea ongelma (engl. wicked problem)[5]. Ohjelmointi- ja suunnitteluvaiheet ovat vaikeita ongelmia ratkaista ja ratkaisu ei ole yksikäsitteinen. Esimerkiksi Niukkanen(1980)[4] mukaan suunnitelman laatua saattaa olla haastavaa mitata ja vertailla.

Koska ohjelmointi ja suunnittelu ovat Rittelien ja Webberin (Rittel et al. 1972)[5] määrittelemiä vaikeita ongelmia, niihin ei voi löytää parasta optimiratkaisua. Periaatteessa ongelmiin voitaisiin aina löytää parempi ratkaisu, eikä ongelmanratkaisuun silloin olisi lopetussääntöä. Haahtelan ohjausmalli perustuu iteraatioihin ja nopeaan palautteeseen hankkeen kaikissa vaiheissa. Nopea palaute auttaa osapuolia arvioimaan päätösten tarkoituksenmukaisuutta ja auttaa myös sitoutumaan yhteisiin tavoitteisiin silloin kun riittävän hyvä ratkaisu on löytynyt. Yhteinen sitoutuminen siis luo hankkeen päätöksenteon lopetussäännön (Pennanen, 2004)[1]. Esimerkiksi ohjelmointivaiheessa tilaaja päättää lisätä auditorion hankesuunnitelmaan. Tilaajalle annetaan välittömästi palautteena, että auditorio kasvattaa budjettia 3 000 000 euroa ja sen käyttöasteeksi nykyisillä toiminnoilla tulisi 5 % arkipäivisin. Palautteen jälkeen tilaaja voisi todeta auditorioon investoinnin tehottomaksi ja päättää, että tarvittaessa auditorio vuokrataan muista rakennuksista. Vastaavanlaista palautejärjestelmää käytetään suunnitteluvaiheessa antamaan palautetta suunnittelijoille koskien heidän päätöksiään.

Palautteeseen perustuva ohjaus on useissa teorioissa esitetty ja sinällään yksinkertainen. Talonrakentamisessa se on kuitenkin ollut erittäin vaikeaa, koska palauteinformaatiota ei ole ollut. Perinteisesti esimerkiksi auditorion kustannus on määritetty mittaamalla suunnitelmista auditorion komponentit ja hinnoitteleamalla ne. Tällöin tilaajan päätökseen haluta auditorio tulisi kustannuspalaute yli puolen vuoden päästä, joka luonnollisesti ei olisi enää arvokas nopean ohjauksen näkökulmasta.

3.1.1 Ohjelmointivaihe

Ohjelmointivaiheen tarkoituksena on ensin selvittää asiakkaan tarpeet tulevalle rakennukselle. Rakennuksella on aina käyttötarkoitus, esimerkiksi asuinrakennus, toimistorakennus, liiketila tai muu vastaava. Nykyään rakennuksista löytyy kuitenkin enemmän ja enemmän useita eri käyttötarkoituksia. Tämä tarkoittaa sitä, että nykyään rakennetaan kokonaisuuksia, jotka sisältävät tietyn määrän liiketiloja, toimistotiloja ja asuntoja. Asiakas haluaa siis toteuttaa tiettyjä toimintoja omista tiloissaan, joten tilojen täytyy tarjota asiakkaalle mahdollisuus toteuttaa näitä liiketoiminnan kannalta tärkeitä toimintoja. (Ballard et al. 2011)[3]

Asiakkaan tarpeet eivät aina ole linjassa asiakkaan maksukyvyyn kanssa. Asiakas saattaa kuvitella saavansa haluamansa tilat eri hintaan, kuin todellisuudessa on mahdollista. Asiakas on valmis maksamaan haluamistaan toiminnoista tietyn hinnan, mutta ei enempää. Tätä kutsutaan sallituksi kustannukseksi engl. allowable cost (Niukkanen, 1980)[4] (Ballard et al. 2013)[6].

Haahtelan ohjausmallissa asiakkaan tavoitteet kuvataan toiminnallisilla kuvauksilla eli toimintoina. Toiminto voisi olla esimerkiksi ruokailutoiminto tai kokoustoiminto. Haahtelan ohjausmallia tukeva tietomalli TAKU[®] muuntaa nämä halutut toiminnot tiloiksi. Tila tarkoittaa rakennuskomponenttien rajaamaa aluetta, jossa voidaan

tehdä tilalle ominaisia toimintoja, kuten järjestää kokouksia, toimia oleskelutilana, toimia keittiönä tai toimia saniteettitilana.

Asiakkaalla saattaa esimerkiksi olla tarve pitää kokouksia. Tällöin Haahtela-mallissa selvitetään asiakkaalta tietoja, esimerkiksi kuinka monelle henkilölle kokouksiloja pitäisi löytyä ja minkälaisia välineitä tilaan tarvitaan. Osalle kokouksilan ominaisuuksista on TAKU[®]-tietomallissa valmiit oletusarvot, kuten millainen ilmanvaihtojärjestelmä, valaistus ja äänieristys tilaan tarvitaan, kunhan henkilömäärä ja todellinen käyttötarkoitus on selvillä. Kaikki tilat arvioidaan erikseen, jolloin lopulta muodostuu kokonaisuus rakennuksen tilojen ominaisuuksista. Tätä kutsutaan Haahtelan TAKU[®]-tietomallissa tilaluetteloksi. Tilaluettelon muodostamisen jälkeen TAKU[®]-tietomalli mallintaa tilojen ominaisuuksien perusteella rakennuksen ominaisuuksia tuottavat järjestelmät (luonnonvalo tuotetaan ikkunoilla, valaistus valaisimilla ja tilojen äänieristys väliseinäratkaisuilla) ja järjestelmien perusteella TAKU[®] määrittää investointi- ja ylläpitokustannukset. Ballard ja Pennanen(2013)[6] määrittelevät artikkelissaan: “Conceptual estimating and target costing” termin odotettu kustannus engl. expected cost. Odotettu kustannus pystytään ohjelmointivaiheessa määrittämään tällä hetkellä noin 95% tarkkuudella ja n. 6% virhemarginaalilla (Ballard et al. 2013)[6]. TAKU[®]-malli siis muodostaa ohjauksessa tarvittavan välittömän kustannuspalautteen.

Mikäli asiakkaan sallittu kustannus ja rakennuttajan odotettu kustannus ovat samat, niistä muodostetaan rakennusprojektin tavoitehintaa engl. target cost (Ballard et al. 2013)[6]. Tällöin asiakas sitoutuu tilaamaan ko. rakennuksen ja puolestaan rakennuttaja sitoutuu rakennuttamaan rakennukset yhdessä sovittuun hintaan. Mikäli asiakkaan sallittu kustannus on pienempi kuin rakennuttajan odotettu hinta, projektia ei voida aloittaa. Tällöin kyseeseen tulee joko asiakkaan tarpeiden uudelleen määrittäminen tai tietyistä toiminnoista luopuminen. Apuna tällaisissa tilanteissa Haahtela-mallissa käytetään esimerkiksi Axiomatic Design-menetelmää ja C-K-teoriaa. Nämä teoriat myös auttavat asiakkaan tarpeiden määrittämisessä tiloiksi ja toiminnoiksi. Lisäksi saadaan selville rakennuskomponenttien tavoitehinnat, vaikka materiaalit eivät vielä olekaan tiedossa. (Ballard et al. 2011)[3] (Ballard et al. 2013)[6]

3.1.2 Suunnitteluvaihe

Suunnittelun alkuvaiheessa on tiedossa rakennuksen tavoitehintaa ja toiminnot (tilaluettelo), mutta rakennuksen muodosta tai ulkonäöstä lopullisessa sijainnissa ei ole olemassa vielä tietoa (Ballard et al. 2011)[3]. Suunnitteluvaiheen tarkoituksena on juuri tämän tiedon aikaansaaminen. Tarkemmalla tasolla suunnitteluvaiheessa saadaan tieto: (Ballard et al. 2011)[3]

1. Rakennuksen muodosta sen tulevassa ympäristössä
2. Asiakkaan aktiviteettien sijainneista ja yhteyksistä toisiinsa
3. Rakennuskomponenteista ja materiaaleista

Suunnitteluvaihe on siis ohjelmointivaiheen tavoin Rittelin ja Webberin (Rittel et al. 1972)[5] määrittämä vaikea ongelma: Kaksi eri suunnittelijaa antaa samaan ongelmaan erilaisen suunnitteluratkaisun, joita molempia voidaan pitää oikeana.

Haahtelan ohjausmallissa TAKU[®]-tietomalli tuottaa rakennuksen kullekin järjestelmälle järjestelmäkohtaisen tavoitehinnan siten, että järjestelmien tavoitehinnan summa on sama kuin koko hankkeen tavoitehintana. Järjestelmät ovat kokonaisuuksia, jotka tuottavat rakennuksen halutut ominaisuudet. Sisäilmaston ominaisuudet tuotetaan lämmitysjärjestelmällä ja jäähdytysjärjestelmällä, luonnonvalo ikkunoilla ja häiriötön sähkönsyöttö generaattorilla tai akustolla. TAKU[®]-tietomalli siis tuottaa sisällöltään täydellisen kuvauksen rakennuksen järjestelmistä ennen suunnittelua. Tätä voidaan pitää yhtenä mahdollisena suunnitelmana muiden joukossa, puolustavana mestarina.

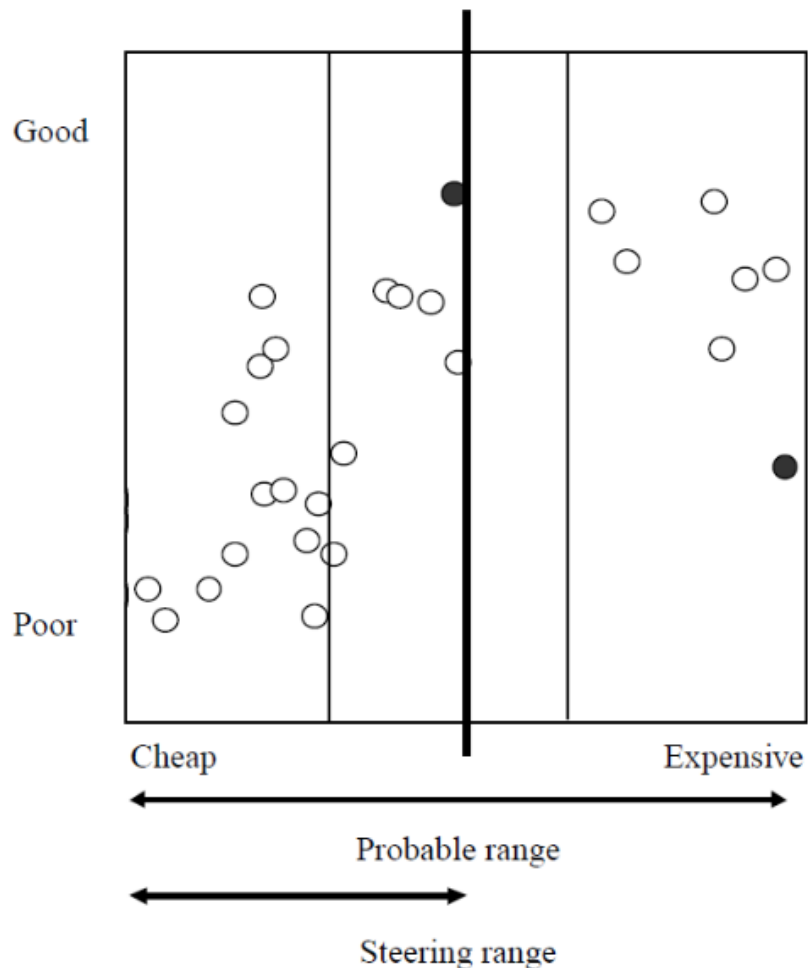
Ballard et al. (2011)[3] mukaan suunnitteluun liittyy suuri vaihtelu, sillä suunnittelijat todennäköisesti tarjoavat toisenlaista ratkaisua. Suunnittelu on kuitenkin luonteeltaan kumulatiivista, se lähtee 0 %:n tietosisällöstä (Ballard et al. 2011)[3]. Kuukauden suunnittelun jälkeen on hahmotettu ulkoseinät, vesikatto ja laatat. Vuoden kuluttua suunnitellaan jäähdytysjärjestelmä. Ohjauksen kannalta tämä on ongelmallista, koska loppukustannukset selviäisivät vasta kun suunnittelu on kokonaisuudessaan valmis (Ballard et al. 2011)[3].

Haahtelan ohjausmalli antaa mahdollisuuden tarkastella kustannuksia sitä mukaa, kun suunnittelu tuottaa tietoa suunnittelijoiden BIM-malleihin. Jokaista suunnittelutavaa järjestelmää kohden on jo ennalta luotu tavoite. Jos ulkoseinä osoittautuu kalliimmaksi kuin sitä vastaava tavoite (puolustava mestari), siitä annetaan palaute suunnittelijoille. Ballard et al. (2011)[3] mukaan tilanne voidaan korjata kahdella tavalla: suunnittelijat voivat muuttaa suunnitelmaa tai osoittaa, kuinka erotus voidaan kompensoida myöhemmällä suunnittelulla.

Ballard et al. (2011)[3] painottaa, että suunnitteluvaiheen suunnittelunohjaus vaatii tavoitteen, rajoitteita sekä nopeaa palautetta. Tämä puolestaan johtaa siihen, että suunnitteluvaihe on iteratiivinen, jossa rakennuttaja ja arkkitehti yhteistyössä sopivat esimerkiksi tilojen sijaintien parannuksista tai materiaalikustannusten karsimisesta. Suunnitteluvaiheessa saadaan ohjelmointivaihetta tarkempaa tietoa tulevista kustannuksista ja tässä vaiheessa on myös tärkeää, ettei suunnitelman odotettu kustannus ylitä asiakkaan sallittua kustannusta (Ballard et al. 2011)[3]. Suunnitelmaa parannellaan, kunnes sen hinta on sopivalla vaihteluvälillä asiakkaan sallittuun kustannukseen verrattuna (Ballard et al. 2011)[3].

Niukkanen(1980)[4] esittää, ettei suunnitelman kalliimpi hinta ei takaa suunnitelman laatua ja toisaalta halvemmat suunnitelmaratkaisut eivät välttämättä toteuta kaikkia asiakkaan tarpeita. Liian kalliita suunnitelmia ei kannata valita ohjattavaksi, sillä ne ovat vaikeammin tasapainotettavissa asiakkaan sallittuun kustannukseen nähden. Tämän vuoksi jatkokehittävänä valitaan sopivalta ohjausväliltä suunnitelmat ja niiden laatu pyritään määrittämään vertailukelpoisesti. Kuvassa 1 on esitetty erilaisia suunnitteluvaiheen ratkaisuja niiden laadusta kustannusten funktiona. Ohjausväliseksi on valittu tässä tapauksessa kaavion vasen puoli, jossa kaikki ratkaisut alittavat asiakkaan odotetun kustannuksen. Kuvasta nähdään myös, että suurempi hinta näyttäisi ohjausvälillä jossain määrin korreloivan positiivisesti laadun kanssa,

mutta vastaavasti hinnan ollessa korkeampi, suunnitelmien laatu ei enää välttämättä parane: jotkin kalleimmista ratkaisuista ovat laadultaan huonompia, kuin jotkin huomattavasti halvemmat suunnitelmat (Niukkanen, 1980)[4].



Kuva 1: Niukkanen 1980: Suunnitteluvaiheen ohjausväli. [4]

Suunnitteluvaiheesta päästään etenemään yhteisen sopimuksen synnyttyä rakennusvaiheeseen. Projektinjohto lupaa asiakkaalle, että suunnitelma kattaa asiakkaan tarpeet täyttävät toiminnot suunnitelman hintaan. Asiakas puolestaan sitoutuu maksamaan rakennuksesta sovitun hinnan. (Ballard et al. 2013)[6]

3.1.3 Rakentaminen työmaalla - Rakennusvaihe

Suunnitteluvaiheesta valitaan paras sallitun kustannuksen alittava ratkaisu. Tässä vaiheessa rakennuksen sijainti, muoto, tavoitehintaa ja rakennuskomponentit ovat tiedossa ja rakennus täytyy vielä rakentaa (Ballard et al. 2011)[3]. Tärkeää on toteuttaa rakennusprojekti sovitun mukaisesti eli kyse on asiakkaalle annettujen lupauksen täyttämistä. Suunnitteluvaiheessa tehtyihin ratkaisuihin ei yleensä tässä vaiheessa tehdä enää muutoksia, ilman pakottavaa tarvetta (Ballard et al. 2011)[3]. Päätöksenteossa hyödynnetään ohjelmointi- ja suunnitteluvaiheiden tuottamaa tietoa. Johtamisen filosofia muuttuu ohjauksesta enemmänkin kontrolloinnin puolelle: täytyy huolehtia siitä, että oikeat resurssit kuten työntekijät, materiaalit ja asiantuntijat ovat paikalla ja valmiina kun niitä tarvitaan.

Projektin toteutuneet kokonaiskustannukset ovat tiedossa vasta siinä vaiheessa, kun rakennus on valmis käyttöönottoon. Sitä aiemmin kustannukset perustuvat arvioihin. Talojen rakennuttaminen on siis kompleksinen, useasta vaiheesta ja prosesseista koostuva prosessi, jossa asiakkaan tarpeista luodaan lopulta asiakkaan toiminnan kannalta tärkeitä tiloista koostuvia rakennuksia.

3.2 Kompleksisuus

Systeemi tarkoittaa joukkoa erilaisia osia ja toimintoja, jotka yhdessä toimiessaan ovat toimivia prosesseja tai kokonaisuuksia (Serman, 2000)[7] (Serman, 1994)[8]. Nämä systeemin osat tai toiminnot voidaan ajatella omina muuttujinaan systeemissä (Serman, 2000)[7]. Mitä enemmän muuttujia systeemissä on sitä vaikeampaa systeemin toiminnan ymmärtämisestä tulee (Serman, 2000)[7]. Kompleksisuudella tarkoitetaan sellaista epävarmuutta systeemissä tai sen osassa joka on vaikeaa tai mahdotonta ymmärtää ja hallita täysin. Nam P. Suh:n (1990)[9] määritelmä kompleksisuudelle artikkelissaan: “The principles of design” on: “Measure of uncertainty in understanding what we want to know or in achieving a functional requirement”. Pennanen (2004)[1] määrittelee väitöskirjassaan kompleksisuudelle kaksi erilaista tyyppiä: deduktiivinen ja induktiivinen kompleksisuus. Suh puolestaan on jakanut kompleksisuuden neljään eri luokkaan: Ajasta riippumaton aito kompleksisuus (Time-independent real complexity), Ajasta riippumaton kuvitteellinen kompleksisuus (Time-independet imaginary complexity), Ajasta riippuva kombinatorinen kompleksisuus (Time-dependent combinatorial complexity) ja Ajasta riippuva jaksottainen kompleksisuus (Time-dependent periodic complexity) (Suh, 1990)[9]. Serman(2000)[7] puolestaan jakaa artikkelissaan: “Business dynamics: systems thinking and modelling for a complex world” systeemien kompleksisuuden kahteen eri tyyppiin: kombinatoriseen ja dynaamiseen kompleksisuuteen. Hänen mukaansa kombinatorinen kompleksisuus johtuu systeemissä olevien muuttujien määrästä ja hänen määritelmänsä on hyvin samankaltainen Suh:n määrittelemän kombinatorisen kompleksisuuden kanssa. Serman määrittelee dynaamisen kompleksisuuden johtuvan suuresta kanssakäymisen määrästä systeemissä. Muuttujat vaikuttavat toisiinsa ja systeemin toimivuuden kannalta uutta tietoa syntyy prosessin aikana. Sermanin dynaaminen kompleksisuus on tyyppiltään hyvin saman kaltaista kuin Pennanen (Pennanen, 2004)[1] määrittelemä induktiivinen kompleksisuus. Koska Sermanin määrittelemät kompleksisuuden tyytit sisältyvät jo hieman tarkemmin Suh:n ja Pennanen kompleksisuuden määritelmiin, käytetään teoriapohjana Suh:n ja Pennanen kompleksisuuden määritelmiä.

3.2.1 Deduktiivinen kompleksisuus

Sellaisia ongelmia, joille on olemassa selvä ratkaisu tai joille on mahdollista löytää oikea ratkaisu kutsutaan deduktiivisiksi ongelmiksi (Pennanen, 2004)[1] (Rittel et al. 1972)[5] (Nicolis, 1998)[10]. Pennanen(2004)[1], Rittel ja Webber(1972)[5] sekä Nicolis(1998)[10] artikkelissaan: “Chaos and information processing” käsittelevät kaikki deduktiivista kompleksisuutta samankaltaisesti. Tällaisissa tapauksissa ratkaisu voidaan päätellä olemassa olevan tiedon pohjalta esimerkiksi lineaarisen regression avulla eli keräämällä tietoa, analysoimalla sitä ja ratkaisemalla ongelman (Pennanen, 2004)[1]. Deduktiivinen ongelmanratkaisu ei tuota prosessin aikana sellaista uutta informaatiota, jota tarvitaan ongelman ratkaisemiseksi. Esimerkiksi matemaattiset ongelmat ovat deduktiivisia ongelmia (Pennanen, 2004)[1].

Kaikkiin deduktiivisiin ongelmiin ei kuitenkaan ole olemassa selviä tai helppoja oikeita ratkaisuja. Joskus ratkaisu voi olla jopa mahdotonta saavuttaa. Pennanen esittää kolme tilannetta, jossa ratkaisu deduktiiviseen ongelmaan saattaa olla vaikea

löytää (Pennanen, 2004)[1]:

1. **Ongelmaa ei osata kuvata riittävän hyvin**, jotta ratkaisu olisi mahdollista löytää. Ongelma voi olla esimerkiksi tiedon puuttumisessa tai tarkkuudessa, jolloin ongelman ratkaisut eivät välttämättä korjaa itse ongelmaa. Tällaisissa tapauksissa saatetaan kuitenkin löytää uutta informaatiota ongelman luoteesta ja näin saada ongelma kuvattua paremmin ja ratkaistua tämän uuden tiedon pohjalta
2. **Ongelma on systeemissä, joka on yksinkertaisesti liian monimutkainen**. Tietoa systeemin osista ja niiden vaikutuksista toisiinsa saattaa olla vaikea saada. Pennanen (Pennanen, 2004)[1] käyttää esimerkkinä sään ennustamista: tietoa on olemassa vain sään nykytilasta ja kaikki ennustukset perustuvat arvioihin siitä, miten säätila mahdollisesti muuttuu seuraavien päivien aikana. Koska muuttujia on paljon ja tietoa tulevaisuudesta ei ole tai se on epävarmaa, ei ole järkevää ennustaa säätä esimerkiksi kuukauden päähän.
3. **Systeemi on kaottinen**. Systeemin muuttujat vaikuttavat toisiinsa ja vaikutuksia on vaikea ennustaa. Pienet muutokset jossakin systeemin muuttujassa voivat aiheuttaa suuria ongelmia muualla systeemissä.

3.2.2 Induktiivinen kompleksisuus

Sellaisia ongelmia, joille ei ole olemassa selvää vastausta kutsutaan induktiivisiksi ongelmiksi (Pennanen, 2004)[1]. Induktiivisessa ongelmanratkaisussa pyritään löytämään toimiva ratkaisu ongelmaan käyttämällä hyväksi niitä tietoja, joita ongelman ratkaisijalla on käytössään alkutilanteessa (Pennanen, 2004)[1]. Tällöin ratkaisu perustuu olemassa olevaan tietoon ja lopputulosta voidaan yrittää ennustaa näiden tietojen perusteella. Ratkaisuja ongelmaan voi kuitenkin olla useita, sillä sama lopputulos voidaan saavuttaa usealla eri tavalla. Oikeita ratkaisuja voi siis olla useita ja aivan kaikkea informaatiota ei edes tarvitse olla heti alussa saatavilla, mikäli se voidaan tuottaa prosessin edetessä. Induktiivinen ongelmanratkaisuprosessi siis tuottaa uutta tietoa, jota tarvitaan ongelman ratkaisemiseksi (Pennanen, 2004)[1]. Induktiivisille ongelmille on tyypillistä, ettei ole olemassa oikeaa tai väärää ratkaisua vaan ratkaisujen laatu määritellään pikemminkin sen mukaan onko ratkaisu hyvä tai huono (Pennanen, 2004)[1]. Induktiiviset ongelmat ovat luonteeltaan vaikeita ratkaista siksi, ettei voida tietää milloin ratkaisu on riittävän hyvä. Lisäksi ratkaisun oikeellisuutta tai hyvyttä saattaa olla vaikeaa tai mahdotonta testata käytännössä.

Rittel ja Webber (Rittel et al. 1972)[5] käyttää vaikeista induktiivisista systeemien ongelmista nimitystä vaikea ongelma engl. wicked problem. Ongelma on heidän mukaansa vaikea, mikäli se toteuttaa seuraavat kymmenen ehtoa:

1. **Ongelma on vaikea määritellä** ja sitä ratkaistaessa alkuperäiset oletukset ongelman syistä osoittautuvat vääriksi. Kun ratkaisu on jo melkein valmis, saatetaan koko ongelma joutua määrittelemään uudestaan. Alkuperäistä ongelmaa voidaan täydentää saadulla informaatiolla.

2. **On vaikeaa tietää, milloin ongelma on ratkaistu**, sillä saattaa olla hankalaa määrittellä, milloin saatu ratkaisu ratkaisee ongelman riittävän hyvin. On siis vaikeaa tietää milloin ongelman ratkaiseminen voidaan lopettaa ja todeta ratkaisun olevan riittävän hyvä.
3. **Ratkaisut ongelmaan eivät ole “oikein” tai “väärin” vaan pikemmin “hyviä” tai “huonoja”**. Useampi ratkaisu saattaa korjata itse ongelman, mutta jokin ratkaisusta saattaa olla esimerkiksi parempi laadultaan, helpommin jatkokehittettävä tai kustannustehokkaampi. Ratkaisujen vertailu saattaa olla hankalaa.
4. **Ratkaisua on vaikeaa tai mahdotonta testata** ennen ratkaisun valmistumista.
5. **Ratkaisu on luonteeltaan sellainen, että se ratkaisee ongelman kerralla ja pysyvästi**. Mahdollisuutta iteroida valmiita ratkaisuja ei ole.
6. **Ratkaisut ongelmaan eivät ole yksikäsitteisiä** eli samaan ongelmaan voi olla useita eri ratkaisuja.
7. **Jokainen vaikea ongelma on uniikki**. Ei ole olemassa vastaavia ennakkotapauksia, joista voisi suoraan johtaa ratkaisuja ongelmiin.
8. **Jokaisen vaikean ongelman voidaan katsoa olevan oire jostain toisesta ongelmasta**.
9. **Ongelman sisältämät ristiriidat voidaan esittää monella eri tavalla**. Tällöin valittu esitystapa vaikuttaa ongelman ratkaisun luonteeseen.
10. **Ongelman ratkaisijalla ei ole oikeutta olla väärässä**.

Induktiivisen ongelman lopputuloksia ei voida tietää etukäteen. Lopputulosta voidaan kuitenkin jälkikäteen tutkia ja pyrkiä ymmärtämään ongelman luonne paremmin. Tästä ei kuitenkaan ole apua enää kyseisen ongelman ratkaisemiseksi, mutta tietoa voidaan yrittää hyödyntää tulevissa projekteissa. (Pennanen, 2004)[1]

3.2.3 Kompleksiset järjestelmät

Kompleksisissa järjestelmissä ja prosesseissa on yleensä paljon muuttujia ja muuttujat vaikuttavat toisiinsa eri tavoin. Muuttujien vaikutuksia toisiinsa ei siitä johtuen ole helppoa ja aina edes mahdollista testata siten, että vaikutuksia voisi ennustaa ja soveltaa käytännössä. (Pennanen, 2004)[1] (Koskela et al., 2005)[11] Yleisellä tasolla tällaisia kompleksisten järjestelmien muuttujia voivat olla esimerkiksi: tuotteen laatu, tuotantokustannukset, kilpailijoiden vastaavien tuotteiden ominaisuudet, työpaikat, työntekijöiden motivaatio ja siihen vaikuttavat tekijät, asiakkaat ja heidän tarpeensa sekä arvomaailmansa (Pennanen, 2004)[1].

Pennanen (2004)[1] mukaan kompleksiset järjestelmät eivät ole vain osiensa summa: Järjestelmän osien yksittäiset ominaisuudet eivät välttämättä ole tärkeitä, vaan

se miten ne vaikuttavat systeemiin ja sen muihin osiin. Vaikka systeemin yksittäisten osien ominaisuudet pystytään kuvaamaan, niiden vaikutukset systeemin muihin osiin eivät välttämättä ole kokonaan tai osittain tiedossa. Myös valitut ratkaisut saattavat vaikuttaa systeemin muihin osiin odottamattomilla tavoilla, jolloin vaikutukset näkyvät myös loppuratkaisussa. (Pennanen, 2004)[1]

Suh:n (2005)[12] mukaan systeemeissä on olemassa neljää eri tyyppistä kompleksisuutta ja kompleksisuuden tyypit on määritelty hänen kirjassaan: “Complexity - theory and applications”:

1. **Ajasta riippumaton aito kompleksisuus** (Time-independent real complexity): Mittaa epävarmuutta siitä, että jokin tehtävä voidaan suorittaa.
2. **Ajasta riippumaton kuvitteellinen kompleksisuus** (Time-independent imaginary complexity): Mittaa epävarmuutta käytettävän systeemin tai järjestelmän toimivuudesta tai käyttäytymisestä.
3. **Ajasta riippuva kombinatorinen kompleksisuus** (Time-dependent combinatorial complexity): Mittaa monimutkaisten järjestelmien osien vaikutuksia toisiinsa ajan funktiona. Esimerkiksi aikataulun myöhästyminen jossain osassa järjestelmää saattaa aiheuttaa ennustamattomia seurauksia myös muualla järjestelmässä.
4. **Ajasta riippuva jaksottainen kompleksisuus** (Time-dependent periodic complexity): Mittaa jaksottaista ajasta riippuvaa kompleksisuutta. Vaikka järjestelmän jokin osa ei pysy aikataulussa, aikataulu korjaantuu tietyn jakson aikana. Esimerkiksi kun työ on tauolla öisin.

Aitoa kompleksisuutta on Suh:n (2005)[12] mukaan mahdollista vähentää ja kuvitteellinen kompleksisuus on mahdollista eliminoida kokonaan. Lisäksi kombinatorista kompleksisuutta voidaan pyrkiä muuntamaan todelliseksi kompleksisuudeksi, jolloin myös sitä voidaan välillisesti vähentää. Näiden kategorioiden lisäksi Koskela et al. (2005)[11] jaottelevat kompleksisuuden edelleen välttämättömään (necessary complexity) ja tarpeettomaan (unnecessary complexity) kompleksisuuteen. Välttämätön kompleksisuus tarkoittaa sellaista kompleksisuutta, jota ei voida poistaa järjestelmästä. Järjestelmän tai toiminnallisten tarpeiden ominaisuudet tai toimitatavat siis ovat luonteeltaan jonkin verran kompleksisia ja tälle ei voida mitään. Suh:n määrittelemä aito kompleksisuus on tällaista välttämätöntä kompleksisuutta. Tarpeeton kompleksisuus on luonteeltaan sellaista, että se turhaan monimutkaistaa järjestelmää tai sen prosesseja. Suh:n määritelmän mukaan kuvitteellinen-, kombinatorinen ja jaksottainen kompleksisuus ovat tarpeetonta kompleksisuutta (Suh, 2005)[12]. Tarpeetonta kompleksisuutta voidaan yrittää vähentää tai se voidaan joissakin tapauksissa poistaa kokonaan järjestelmästä (Suh, 2005)[12].

3.3 LEAN ja hukka

Ballard ja Howell (1995)[13]: “Toward construction JIT”, Koskela (2004)[14]: “Making-do - the eight category of waste”, sekä Ballard, Koskela ja Tommelein (2002)[15]: “The foundations of lean construction” tutkimukset tutkivat LEAN-ajattelua talonrakentamisen näkökulmasta. Ohno(1988)[16] käsittelee puolestaan yleisemmin LEAN-ajattelua artikkelissaan: “Toyota production system”. LEAN on Japanissa Toyotan autotehtaalla alkunsa saanut ajattelutapa, jossa pyritään tehostamaan tuotantoa vähentämällä tuotannon kustannuksia ja tuotantoaikaa (Ohno, 1988)[16] (Ballard et al., 2002)[15] (Ballard et al., 1995)[13]. Samalla pyritään parantamaan tuotettavan tuotteen laatua sekä vähentämään tuotannon hukkaa (Ohno, 1988)[16]. LEAN-ajattelun alkuperäinen tarkoitus oli siis tuottaa autoja entistä kustannustehokkaammin tinkimättä laadusta (Koskela, 2004)[14]. Vaikka LEAN-ajattelu oli alunperin tuotannon parantamiseen tähtäävä ajatusmalli, LEAN-ajattelu on nykyään levinnyt laajasti myös muiden kuin tuotantoprosessien tehostamiseen (Ohno, 1988)[16]. LEAN-ajattelua kutsuttiin kirjallisuudessa myös termillä JIT (Ballard et al., 1995)[13]. JIT tulee sanoista Just In Time ja sen tarkoitus on tuottaa samoja hyötyjä kuin LEAN.

Hukalla tarkoitetaan ylimääräisiä, tuottamattomia toimintoja, jotka hidastavat prosessia tai tuottavat tarpeettomia kustannuksia (Ohno, 1988)[16]. Hukka on seurausta prosesseissa tapahtuvista vioista ja virheistä, jotka kompleksisuus aiheuttaa (Ohno, 1988)[16]. Hukkaa ovat siis käytännössä sellaiset tuotannon vaiheet, jotka kuluttavat liikaa aikaa tai resursseja. On olennaista tunnistaa, mikä todellisuudessa aiheuttaa hukan, sekä minkälaisia seurauksia hukka aiheuttaa prosessille. Tärkeintä on eliminoida hukan varsinainen syy, jolloin myös hukan pitäisi poistua tietyllä aikavälillä (Ohno, 1988)[16]. Mikäli keskitytään vain hukan vähentämiseen, todellista ongelmaa ei ole ratkaistu ja hukka palaa takaisin prosessiin (Ohno, 1988)[16]. Erilaista hukkaa ovat ainakin (Ohno, 1988)[16]:

1. **Ylituotannon hukka**
2. **Viallisten tuotteiden/osien korjaamisen hukka**
3. **Materiaalien liikkumisen hukka**
4. **Materiaalin tai tiedon käsittelyyn liittyvä hukka**
5. **Varastoinnin hukka**
6. **Odottamisen hukka**
7. **Liikkeen hukka**

Koskela(2000)[17] kertoo artikkelissaan: “An exploration towards a production theory and its application to construction”, ettei kaikkia eri tyyppisiä hukkia ei välttämättä löydy kaikista prosesseista ja hukka täytyykin pyrkiä tunnistamaan prosessista tutkimalla itse prosessia. Koskela esittää kolme yleistä syytä hukan syntymiselle (Koskela, 2000)[17]:

1. **Tuotantosysteemin rakenne.** Organisaation hierarkiset rakenteet haittaavat tiedonkulkua ja näin aiheuttavat ongelmia tuotannossa. Tämä näkyvä helposti esimerkiksi tuotantoaikojen pitkittymisenä tarpeettomasti. Useimmiten toiminnot riippuvat useista eri henkilöistä ja tuotannon osiot on pilkottu pienempiin osakokonaisuuksiin. Tällöin tietoa liikkuu paljon paikasta toiseen, mikä johtaa mm. odottamiseen, tarkkailuun ja tavaran liikkumiseen paikasta toiseen.
2. **Tuotannon kontrollointi.** Mikäli tuotantoa kontrolloidaan tiukasti, se hidastuu ja päin vastoin. Täytyy löytää sopiva kontrollin taso riippuen tuotettavasta asiasta.
3. **Vaikeasti ennustettavat ongelmat tuotannossa.** Joskus koneita hajoaa ja tuotanto keskeytyy, myös ihmiset tekevät virheitä. Tämän tyyppiseen hukkaan saattaa olla vaikeaa varautua etukäteen.

LEAN pyrkii vähentämään turhaa työtä tai hukkaa erilaisissa prosesseissa. Sen avulla voidaan pyrkiä parantamaan asiakastytyväisyyttä tai laatua, pienentämään toiminnan kustannuksia tai tehostamaan tuotantoprosesseja (Ohno, 1988)[16]. Perusajatuksena on, että oikea määrä oikean laatuista oikeita asioita saadaan oikeaan aikaan ja paikkaan. Hukka pyritään tunnistamaan nopeasti, jotta se voidaan eliminoida prosessista. Tällä tavalla saattaa olla mahdollista pienentää prosessin kustannuksia sekä parantaa laatua. (Ohno, 1988)[16] (Kuong, 2000)[18]

LEAN-ajattelu on tärkeää talonrakentamisessa, sillä aikataulut ovat tarkkoja, rakennuskomponenttien laatu on erittäin tärkeää ja aikataulussa pysymättömyys kasvattaa hankkeen kokonaiskustannuksia lisäämällä prosessin aikasidonnaista kompleksisuutta. LEAN on projektinhallinnassa yleisesti käytössä oleva ajattelutapa. LEAN pyrkii tehostamaan prosesseja, jonka myötä prosessi tuottaa enemmän arvoa projektissa. LEAN-ajattelu voi olla asiakaskeskeistä, joten se sopii hyvin myös talojen rakentamisen suunnittelun työkaluksi. (Ballard et al., 2002)[15]

Kuong (2000)[18] sanoo artikkelissaan: "Process performance measurement system: a tool to support process-based organizations", että prosesseissa hukkaa voidaan pyrkiä vähentämään tarkkailemalla prosessia ja keräämällä tietoa prosessista. Hukan vähentäminen voidaan esimerkiksi aloittaa muodostamalla tarkasteltavasta prosessista hypoteesi: mitä prosessissa oletetaan tapahtuvan ja miten. Tämän jälkeen prosessi aloitetaan, jolloin hypoteesia testataan käytännössä (Kuong, 2000)[18]. Tämän jälkeen voidaan vertailla hypoteesia toteutuneeseen tilanteeseen ja tehdä sen perusteella muutoksia toimintatapoihin. LEAN-ajattelu on siis iteratiivista ja hukkaa voidaan poistaa pienempinä paloina (Kuong, 2000)[18].

Koskela (2000)[17] tarjoaa viisi yleistä tapaa hukan vähentämiseksi:

1. **Minimoi läpimenoaikoja.** Nopeampi tuotanto vähentää tuotannosta aiheutuvaa hukkaa, mutta täytyy silti välttää ylituotantoa.
2. **Vähennä prosessin muuttujia.** Mitä enemmän liikkuvia osia, sitä enemmän hukkaa voi potentiaalisesti olla olemassa.
3. **Yksinkertaista prosessia.** Mitä monimutkaisempi prosessi on, sitä enemmän hukkaa se potentiaalisesti sisältää.

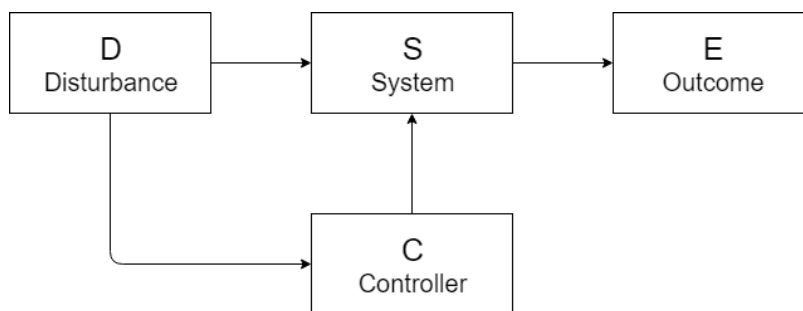
4. **Lisää joustavuutta.** Esimerkiksi väljemmät aikataulut tms.
5. **Lisää läpinäkyvyyttä.** Tieto kulkee paremmin, mikäli se on helpommin saatavissa. Läpinäkyvyys helpottaa informaation keräämistä prosessista tai systeemistä

LEAN-ajattelun mukaan prosessia tutkimalla, prosessista on mahdollista löytää hukkaa. Hukan syitä tutkimalla ja korjaamalla, prosesseja on mahdollista tehostaa. Monimutkaisia prosesseja voidaan tutkia sen perusteella, miten niitä ohjataan. Seuraavassa luvussa käsitellään kompleksisten prosessien ohjausta.

3.4 Kompleksisten prosessien ohjaus

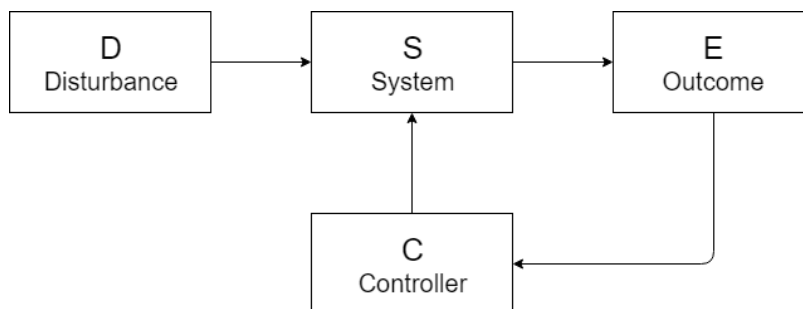
Kompleksisia systeemejä voidaan yrittää hallita ymmärtämällä miten systeemi toimii ja miten systeemin muuttujat vaikuttavat toisiin systeemin osiin (Principia Cybernetica, 1992)[19]. Kybernetiikka (engl. cybernetics) tutkii kompleksisuutta, sekä sen hallintaa ja ymmärtämistä systeemeissä (Principia Cybernetica, 1992)[19] (Pennanen, 2004)[20]. Tällä hetkellä yksi merkittävimmistä kybernetiikan tutkijoista on Principia Cybernetica, joka selittää kybernetiikan perusperiaatteet tutkimuksessa: “The nature of cybernetic systems” vuodelta 1992[19]. Pennanen tarjoaa kaksi erilaista tapaa pyrkiä hallitsemaan kompleksisuutta systeemeissä (Pennanen, 2004)[20]:

1. Täydellinen kontrolli(Perfect control)



Kuva 2: Täydellinen kontrolli (Pennanen, 2004)[20]

2. Suljettu kierto(Closed loop control)



Kuva 3: Suljetun kierron kontrolli (Pennanen, 2004)[20]

Kuvissa 2 ja 3 on neljä eri toimijaa tai toimintoa. S kuvaa järjestelmää, D häiriötä järjestelmässä, C kontrolloijaa ja E lopputulosta. Täydellisen kontrollin tapauksessa kuvassa 2 häiriötilanteen(D) sattuessa sillä on vaikutusta systeemiin(S) sekä kontrolloijaan(C). Tällöin voidaan kuitenkin saavuttaa haluttu lopputulos(E), mikäli kontrolloija(C) osaa reagoida häiriöön tai virheeseen(D) oikealla tavalla. Tällöin kontrolloija(C) vaikuttaa itse systeemiin(S) siten että lopputulos(E) pysyy häiriöstä(D) huolimatta haluttuna. Tämän tyyppinen systeemin kontrollointi vaatii kuitenkin tiedon välittymisen riittävän ajoissa kontrolloijalle(C), jotta muutokset systeemeissä(S)

ehtivät vaikuttaa tulokseen(E). Kompleksisissa systeemeissä ei kuitenkaan usein voida saavuttaa täydellistä kontrollia, jolloin kontrollointi voi muuttaa muotoaan suljetun kierron kontrollointiin. (Pennanen, 2004)[20]

Suljetun kierron kontrollissa, kuva 3, häiriö(D) vaikuttaa systeemiin(S) ja lopputulokseen(E) ennen kuin kontrolloija(C) pystyy puuttumaan tilanteeseen. Tällöin saadusta lopputuloksesta pyritään päättämään häiriön(D) seuraukset, joiden perusteella voidaan tehdä muutoksia systeemiin(S) ja tämän jälkeen tarkastella uudelleen lopputulosta(E). Tällainen kompleksisten järjestelmien ohjaaminen vaatii siis iteratiivisia toimintatapoja, joissa tietoa kerätään ja analysoidaan sekä tehdään niiden perusteella muutoksia systeemiin. Kompleksisten systeemien ymmärtämisen ja hallittamisen ympärille on myöhemmin rakennettu käytännönläheisiä kompleksisuuden hallinnan työkaluja kuten Axiomatic design ja C-K teoria.

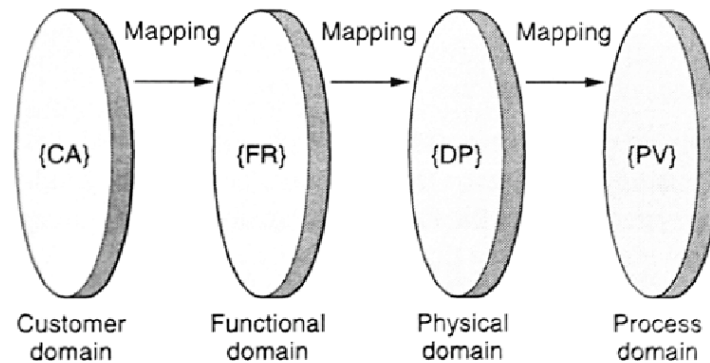
3.4.1 Axiomatic design

Axiomatic design on Nam P Suh:n kehittämä teoria kompleksisten ongelmien ratkaisemiseksi, joka on julkaistu vuonna 1990 artikkelissa “the principles of design” (Suh, 1990)[9]. Axiomatic design tapahtuu neljässä eri domainissa:

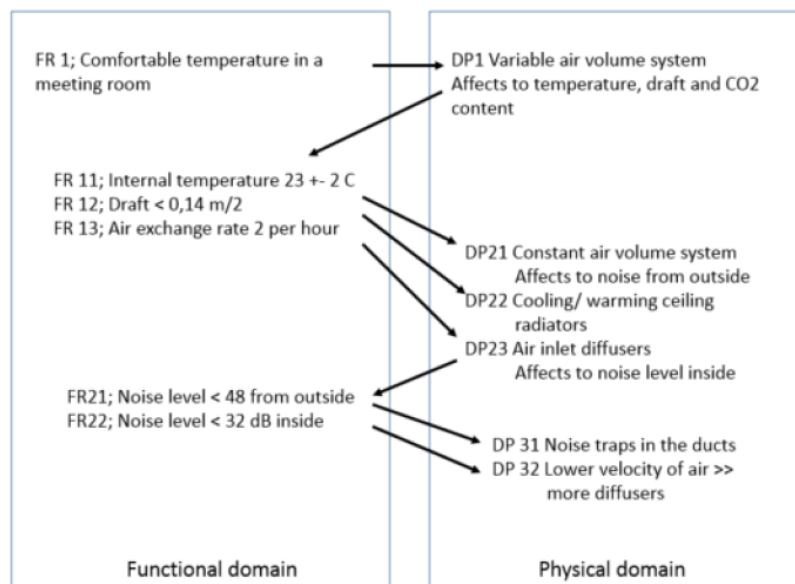
1. **Asiakkaan domain** (Customer domain): Määrittelee hyödyt, joita asiakas haluaa
2. **Toiminnallinen domain** (Functional domain): Määrittelee toiminnalliset vaatimukset suunnitteluratkaisuille asiakkaan tarpeiden mukaan
3. **Fyysinen domain** (Physical domain): Määrittelee suunnitteluparametrit tai tuotteet, jotka voivat tyydyttää asiakkaan tarpeet
4. **Prosessi domain**(Process domain): Liittyy tuotannon prosessien ohjaukseen

Eri domainit on kuvattu kuvassa 4. Asiakkaan domain koostuu asiakkaan tarpeista tai asiakkaan toimintojen tarpeesta, joten asiakas domain sisältää asioita, jotka ovat asiakkaalle tärkeitä. Toiminnallinen domain kytkee asiakkaan tarpeet erilaisiin toimintoihin ja pyrkii selventämään edelleen, minkälaisia toimintoja asiakkaan tarpeen tyydyttäminen vaatii. Toiminnallisessa domainissa määritellään toiminnallisuuksien vaatimukset, jotka ovat siis johdettuja suoraan asiakkaan tarpeista. Fyysisessä domainissa toiminnolle lisätään kytkös reaali maailmaan konkreettisten palveluiden tai tuotteiden avulla, joilla määriteltyjä toimintoja voidaan suorittaa. Lopulta prosessi domainissa luodaan fyysisten toiminnallisuuksien kokonaisuus yhtenäiseksi prosessiksi. Lopputuloksena pitäisi olla asiakkaan tarpeiden kannalta toimiva ja valmis prosessi tai järjestelmä. Yksinkertaistettuna kuvassa liikutaan vasemmalta oikealle. Vasemmalla on asiakkaan tarpeiden mukaiset määrittelyt ja oikealle päin siirryttäessä ratkaisu asiakkaan tarpeiden tyydyttämiseen on lähempänä. Prosessi domainissa saadaan siis jokin toiminto tai palvelu, joka tyydyttää asiakkaan alkuperäiset asiakas domainin tarpeet (Suh, 1990)[9]

Talonrakentamisessa ja tarkemmin projektin ohjauksessa Suh:n teoriaa on pyritty siirtämään käytäntöön mm. Ballard, Haahtela ja Pennanen (Ballard et al. 2013)[6]



Kuva 4: Suh domains

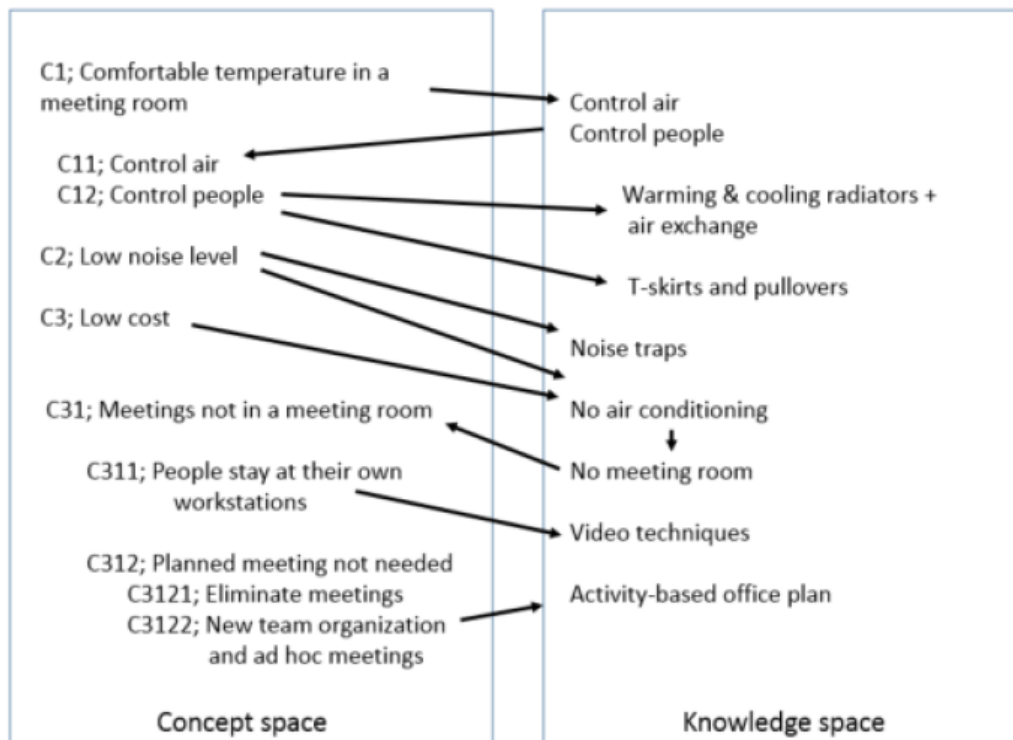


Kuva 5: Axiomatic reasoning

toimesta. He yksinkertaistavat Suh:n neljän domainin käyttämällä vain toiminnallista ja fyysistä domainia. Kuvassa 5 pyritään selvittämään, kuinka tietoa liikkuu domainista toiseen ja miten tietoa jalostetaan käyttämällä yksinkertaista esimerkkiä kokoushuoneen toiminnallisuuden ja fyysisten vaatimusten selvittämiseksi. Fyysisen domainin ratkaisut ovat aina vastauksia toiminnallisen domainin vaatimuksiin ja puolestaan fyysisen domainin ratkaisut saattavat vaatia toiminnallisten tarpeiden täsmentämistä eli uutta tietoa toiminnallisissa domainissa. Suunnittelu on joko yksisuuntaista tai edestakaista tietojen täsmentymistä domainien välillä. Axiomatic design voi löytää ratkaisuja jo tunnettuihin asiakkaan ongelmiin, mutta se ei haasta asiakkaan tarpeiden oikeellisuutta eli tässä tapauksessa tarvetta kokoushuoneelle.

3.4.2 CK-teoria

C-K teoria on Hatchuel et al. (2004)[21] artikkelissa: “C-K Theory in practice: Lessons from industrial applications” julkaisema teoria kompleksisten ongelmien ratkaisemiseksi. C-K teoria pitää sisällään kaksi domainia: konseptuaalinen domain sekä tieto domain (Hatchuel et al., 2004)[21]. Konseptuaalinen domain sisältää asiakkaan tarpeen ja tieto domain sisältää toiminnon, joka vaaditaan asiakkaan tarpeen täyttämiseksi. Myös C-K teoriassa liikutaan domaineissa edes takaisin ja molemmissa tuotetaan uutta tietoa, joka on hyödyllistä toisessa domainissa. Merkittävin eroavaisuus axiomatic design menetelmään nähden on se, että metodi haastaa asiakkaan alkuperäisen tarpeen oikeellisuuden. Kuvassa 6 on esitelty yksinkertaistettu C-K teorian tilanne talonrakentamisen alalla. Ongelma on sama, kuin axiomatic design menetelmässä eli asiakkaalla on tarve pitää kokouksia tiloissaan. Siinä missä axiomatic design pystyy löytämään ratkaisun kokousten järjestämiselle kokoushuoneessa, C-K teorian avulla voidaan havaita, että jotkin asiakkaan toiveet ovat ristiriidassa keskenään kuten ilmastoinnin tarve, hiljaisuuden tarve ja kustannustehokkuus. Näin asiakkaan alkuperäinen tarve kokoushuoneelle voidaan muodostaa uudelleen ja löytää kokousten järjestämiseen toinen ratkaisu: asiakas ei tarvitse kokoushuonetta vaan pikemminkin digitaalisen järjestelmän, jolla kokouksia voidaan järjestää videokonferenssien avulla.



Kuva 6: C-K teoria

3.5 Yleiset tietomallivaatimukset

Rakennuksen tietomalli (engl. Building Information Model, BIM) kuvaa rakennuksen tietoja rakennusprosessin aikana ja sen jälkeen digitaalisessa muodossa (BuildingSMART, 2012)[2]. Tietomalli sisältää tiedot rakennuksen perusosista sekä geometriasta kolmiulotteisesti (BuildingSMART, 2012)[2]. Tietomalleja käytetään rakennusten suunnittelussa, etenemisen seurannassa rakennusprosessin aikana, sekä rakennuksen valmistumisen jälkeen ylläpitotehtävissä, kuten remonteissa. Tietomalli päivittyy rakennusprosessin aikana sitä mukaa kun uutta tietoa syntyy tehtyjen päätösten tuloksena. Tietomallin sisältämä tieto siis kasvaa ja rakennusprosessin päättyessä tietomallissa on kaikki tiedot valmiista rakennuksesta. Azhar, Hein ja Sketo(2007) (Azhar et al., 2007)[22] artikkelissaan: “Building Information Modeling (BIM): Benefits, Risks and Challenges” sekä Hartmann, van Meerveld, Vossebeld ja Adriaanse (2011)[22] artikkelissaan: “Aligning building information model tools and construction management methods” pitävät tärkeänä sitä, että tietomallista voidaan tarkistaa rakennuksen tiedot esimerkiksi peruskorjauksia tai rakennuksen purkua varten. Tällöin on erittäin tärkeää tietää, miten rakennus on rakennettu(Hartmann et al. 2011)[23]. Tietomalli on siis merkittävässä roolissa nykypäivän talonrakentamisessa. Tietomalli on kokonaisuus, joka muodostetaan useiden eri toimijoiden tuottamasta tiedosta (BuildingSMART, 2012)[2].

Informaation digitalisoiminen helpottaa tiedon säilymistä ja sen käsittelyä, sekä mahdollistaa tiedon yhteiskäytön, eli moni henkilö voi tarkastella tietoa samanaikaisesti, vaikka he eivät fyysisesti sijaitse samassa paikassa. Digitaalinen malli auttaa projektin aikana laadunvarmistuksessa sekä mahdollisten muutosten hallinnassa prosessin aikana. Suunnitelmien muuttaminen on helpompaa ja muutokset saadaan helpommin kommunikoitua tarvittaville projektin sidosryhmille. Mallin avulla valittuja materiaaleja voidaan luonnostella ja testata. Testaus voidaan osittain automatisoida, niin että tietomallista luetaan dataa ja tarkistetaan ennalta määrättyjen ehtojen mukaisesti. (Azhar et al., 2007)[22] (Hartmann et al. 2011)[23]

Tietomallin tarkoitus on olla helposti luettavissa erilaisiin 3D malleihin. Tällöin tietoja voi tarkastella visuaalisesti, jolloin rakennuksen kokonaiskuva on selkeämpi rakennusprosessin aikana. Digitaalisesta mallista voidaan luoda erilaisia simulaatioita, esimerkiksi eri järjestelmien toimivuudesta, rakennuksen energiatehokkuudesta tai materiaalien kustannuksista. Malliin voidaan myös lisätä tarkentavia tietoja rakennusprosessin aikana, kuten aika- ja kustannustietoja, jolloin projektin toteutuneet kustannukset ja aikataulu ovat paremmin seurattavissa. (BuildingSMART, 2012)[2]

Yleiset tietomallivaatimukset toimivat suunnittelijoiden standardina koko suunnitteluprosessin ajan. Talonrakentamisen alalla kasvanut tietomallien käyttö on luonut tarpeen yhteisten standardien ja toimintatapojen määrittämiseen ja sopimiseen. Mallintamisessa on siis yhä tärkeämpää sopia etukäteen mitä mallinnetaan ja millä työkaluilla sekä miten mallinnus toteutetaan. (BuildingSMART, 2012)[2]

Yleiset tietomallivaatimukset ovat COBIM kehittämishankkeen tulos (BuildingSMART, 2012)[2]. COBIM on senaatti-kiinteistöjen mallintamisohjeiden laajentamis- ja päivittämishanke, jonka lopputuloksena ovat muodostuneet kansalliset mallintamisohjeet (BuildingSMART, 2012)[2]. Yleiset tietomallivaatimukset (BuildingSMART,

2012)[2] määrittävät tärkeimmiksi tavoitteikseen: “suunnittelun ja rakentamisen laadun, tehokkuuden, turvallisuuden ja kestävä kehityksen mukaisen hanke ja elinkaari-prosessin tukemisen”. Tietomallille on määritetty omat vaatimusmäärittelyt kuhunkin eri vaiheeseen (BuildingSMART, 2012)[2]. Ensimmäinen versio tietomallivaatimuksista julkistettiin vuonna 2007 ja sitä päivitettiin vuosien 2011 ja 2012 aikana, joka on edelleenkin uusin vaatimusmäärittelyn versio (BuildingSMART, 2012)[2]. Yleiset tietomallivaatimukset käsittävät yhteensä 14 osaa, joista tässä tutkimuksessa käytetään lähinnä yleistä osaa (BuildingSMART, 2012)[2].

Yleisten tietomallivaatimusten mukaan julkisten hankkeiden suunnittelussa tulisi käyttää vähintään IFC 2x3 sertifioitua mallinnusohjelmaa. Hankekohtaisesti voidaan kuitenkin määrittellä vielä tarkemmat rajoitukset sille, mitä työkaluja projektissa käytetään. IFC 2x3 sertifikaattia voidaan siis pitää tietomallinnuksen minimivaatimuksena. Uudemmat standardit ovat tapauskohtaisesti järkeviä käyttää, mikäli suunnitelmassa on joitakin erityisvaatimuksia standardin suhteen. Sisäisessä työskentelyssä ja dokumenttien tuottamisessa ei ole kovin paljoa merkitystä, mitä ohjelmistoja ja formaatteja käytetään. Projektin toimijat sopivat keskenään käytettävistä ohjelmista ja formaateista: tärkeintä ei ole käyttää juuri tiettyjä työkaluja vaan se, että toimintatavat ovat selkeät kaikille osapuolille ja että valittu tapa on riittävän hyvä käytettäväksi. Projektin alussa sovitaan yleisten tietomallivaatimusten mukaan nämä asiat: (BuildingSMART, 2012)[2]

1. **Käytettävät ohjelmistot ja formaatit**
2. **Mallien ja muun tiedon jakaminen ja luovuttaminen asiakkaalle**
3. **Mittayksiköt ja koordinaatisto**
4. **Mallien tarkkuus**
5. **Mallinnuksessa käytettävät työkalut**
6. **Rakennusten mallien rakenne**
7. **Tietomallien laadunvalvonta**

Projektin aikana tiedon lisääntyessä ja tarkkuuden kasvaessa yleiset tietomallivaatimukset sisältävät eri kohdissa eri vaatimukset tiedon laadulle ja tarkkuudelle. Projektin aikana sovitusta asioista pyritään pitämään kiinni, jotta voidaan välttyä sekaannuksilta ja tarpeettomasti luoda projektiin lisää kompleksisuutta. Tietomallin luomiseen ja ylläpitämiseen vaaditaan siis tärkeän rakennukseen liittyvän tiedon lisäksi, myös tietynlaista ajattelutapaa, jossa noudatetaan sovittuja käytäntöjä tiedon jakamiseen ja ajan tasalla pitämiseen. Yleisten tietomallivaatimusten mukaan ohjelmointi- ja suunnitteluvaiheet on jaettu pienempiin osavaiheisiin ja vaiheista edetään järjestyksessä aina seuraavaan. Järjestys on vaatimusmallin luominen, ehdotussuunnitteluvaihe, yleissuunnitteluvaihe sekä toteutussuunnitteluvaihe. Tiedon tarkkuus ja määrä kasvaa vaiheissa eteenpäin siirryttäessä ja tiedon tarkkuuden tulee olla yleisten tietomallivaatimusten määrittämällä minimitasolla, jotta voidaan edetä seuraavaan vaiheeseen. (BuildingSMART, 2012)[2]

3.5.1 Vaatimusmalli

Yleiset tietomallivaatimukset määrittelevät vaatimusmallin: “Vaatimusmallin minimaalivaatimus on taulukkomuodossa oleva tilaohjelma, jota voidaan käyttää ohjelman ja suunnitelmaratkaisujen vertailussa” (BuildingSMART, 2012)[24]. Haahtelan ohjausmallissa tämä vastaa tilaluetteloa, joka saadaan ohjelmointivaiheen aikana tarve- ja vaatimustietojensa avulla. “Tilaohjelman tulee sisältää tila- tai tilaryhmäkohtaiset pinta-ala- ja mahdolliset erityisvaatimukset. Taulukkomaista tilaohjelmaa voidaan täydentää käyttäjän ja/tai tilaajan tiloille asettamalla vaatimuksilla.” (BuildingSMART, 2012)[24]. Vaatimusmallissa tulee, yleisten tietomallivaatimusten mukaan, voida simuloida rakennuksen järjestelmiä tai järjestelmien osia, kuten kokonaisenergiankulutusta, jäähdytystarvetta jne. Tilaohjelmaa ja tilojen vaatimuksia tulee pystyä ylläpitämään sähköisessä muodossa, jotta suunnitelmaa voidaan myöhemmissä vaiheissa verrata vaatimukseen. Tilaohjelmassa esitettäviä vaatimuksia ovat esimerkiksi (BuildingSMART, 2012)[24]:

1. Tilan nettotarve ja tarvittaessa mittoihin ja muotoihin liittyviä vaatimuksia
2. Tilan käyttö ja käyttäjät
3. Keskeiset yhteydet ja vaikutukset muihin tiloihin
4. Sisäilman olosuhteet
5. Ääneneritys
6. Valaistus
7. Kuormitus
8. Kestävyys
9. Turvallisuus ja laatutaso
10. LVI-järjestelmät
11. Sähköjärjestelmät
12. Kalusteet
13. Varusteet
14. Laitteet
15. Sisäpuoliset pintarakenteet

3.5.2 Ehdotussuunnittelu

Ehdotussuunnitteluvaiheessa pyritään löytämään paras tarjolla oleva perusratkaisu asiakkaan tarpeisiin. Suunnitelmia siis vertaillaan karkealla tasolla keskenään ja valitaan niistä parhaalta vaikuttaa jatkokehitykseen. Tämä vaihe vastaa Haahtelan ohjausmallissa suunnitteluvaiheen osaa, jossa pyritään löytämään kelvolliset suunnitelmat useiden ehdotusten joukosta. Kyseeseen siis tulee suunnitelmien vertailu laadun ja hinnan perusteella, kuitenkin niin, että suunnitelma täyttää asiakkaan tarpeet. Kaikilla suunnittelijoilla tulee olla pääsy muiden suunnittelijoiden tuottamiin suunnitelmiin. Lisäksi tiedon tulee olla ajantasaista. Tämä varmistetaan hallinnomalla tietomalleja projektipankissa, johon kaikilla suunnittelijoilla on pääsy. Lisäksi suunnittelijat sopivat riittävän tiheästä tietomallien tallennusvälistä projektipankkiin, jotta suunnitelmat ovat ajan tasalla. Yleiset tietomallivaatimukset esittävät, että ehdotussuunnitteluvaiheessa mallien sopiva tallennusväli olisi suunnittelukokousten väli. (BuildingSMART, 2012)[24]

3.5.3 Yleissuunnittelu

Luonnosvaiheen ideana on jatkokehittää ehdotussuunnitteluvaiheessa valittua suunnitelmaa (BuildingSMART, 2012)[24]. Asiakkaan vaatimukset päivitetään ehdotussuunnitteluvaiheessa tehtyjen päätösten mukaiseksi (BuildingSMART, 2012)[24]. Projektijohdon tehtävänä yleissuunnitteluvaiheessa on suunnittelun ohjaaminen ja suunnitelman hyväksyminen toteutussuunnitteluvaihetta varten. Tietomalli mahdollistaa tässä vaiheessa erilaisia analyyseja sekä visualisointeja 3D malleiksi, joita voidaan hyödyntää päätöksenteon tukena. (BuildingSMART, 2012)[24]

Yleissuunnittelun vaatimuksena on, että kaikki suunnitelmat mallinnetaan kerroksittain. Tämä helpottaa erilaisten analyysien tekemistä, sillä myös ne tehdään yleisten tietomallivaatimusten mukaisesti kerroksittain. Projektikohtaisesti tästä tyylistä voidaan kuitenkin poiketa, mikäli se on suunnitteluratkaisujen kannalta perusteltua. Tärkeintä on sopia projektin alussa, miten tietomallinnus toteutetaan ja tämän jälkeen pysyä valitussa tavassa. (BuildingSMART, 2012)[24]

3.5.4 Toteutussuunnittelu

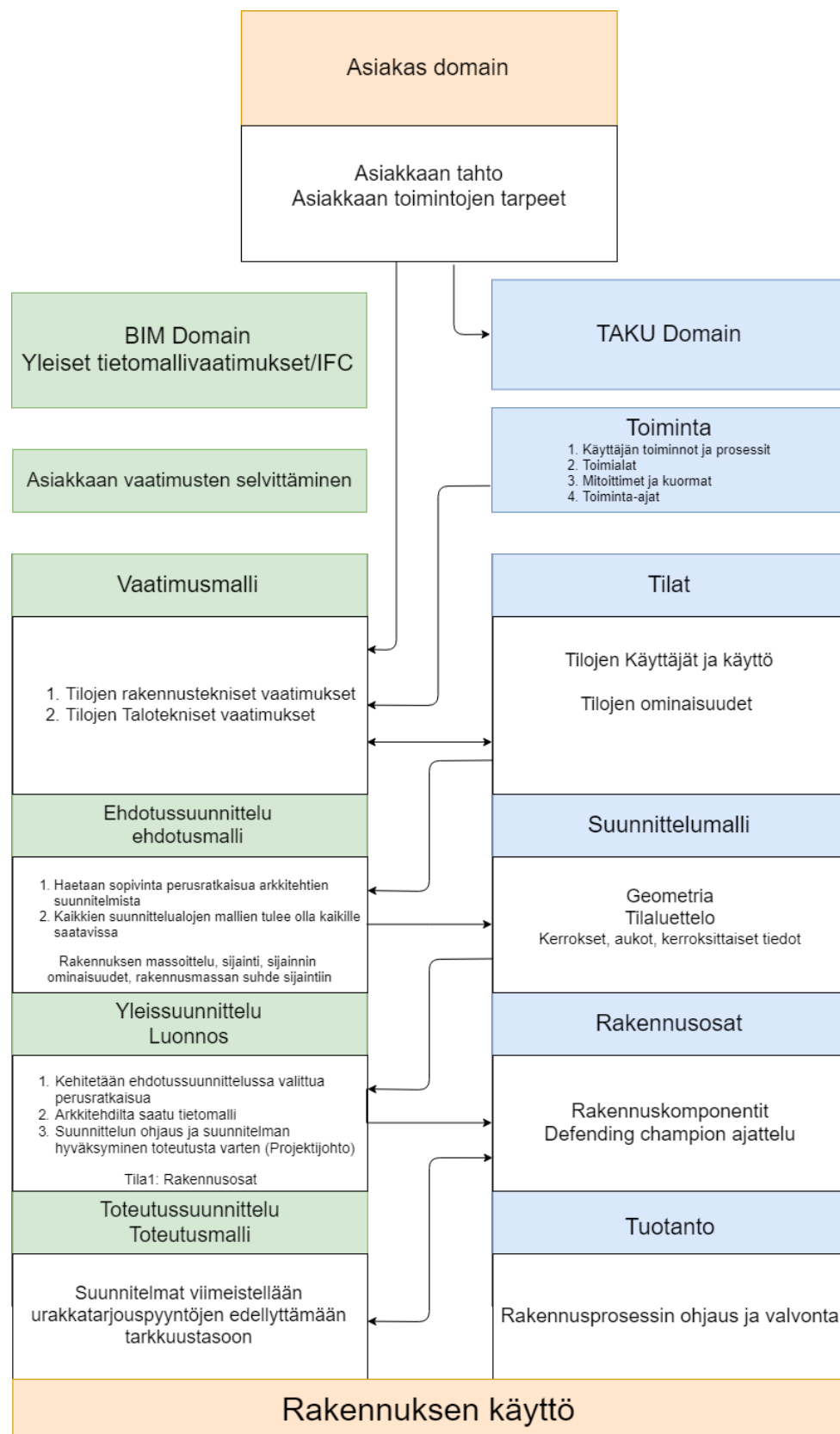
Toteutussuunnitteluvaihe on samankaltainen kuin yleissuunnitteluvaihe ja suurin ero on tiedon tarkkuustasossa, joka kasvaa olennaisesti toteutussuunnitteluvaiheessa. Suunnitelmat viimeistellään siten, että niiden tarkkuustaso on sellainen, että niiden perusteella voidaan aloittaa urakkatarjouspyyntöjen tekeminen. Tässäkin vaiheessa tietomallien tulee olla kaikkien suunnittelijoiden saatavilla yhteisessä projektipankissa. Mallien tietosisältö ja tarkkuustaso on määritelty suunnittelualakohtaisten tietomallinnusohjeiden osissa 3-5. Käytännössä tiedon tarkkuus pitää olla sellaisella tasolla, että siitä on hyötyä projektin muille toimijoille. (BuildingSMART, 2012)[24]

3.6 Tietovirrat

Tsoukas (2009)[25] kertoo artikkelissaan: “Dialogical Approach to the Creation of New Knowledge in Organizations”, että uutta tietoa syntyy organisaatiossa ja sen vaikutuspiirissä ihmisten välisessä keskustelussa sekä tehtävissä ja toiminnoissa, joita he suorittavat työskennellessään. Mikä tahansa dialogi ei kuitenkaan ole hyödyllistä uuden tiedon kannalta, sillä tiedon arvo määräytyy myös tiedon saatavuuden ja ymmärrettävyyden mukaan. Mikäli tietoa ei voida hyödyntää oikeaan aikaan, tiedon hyöty laskee. Tiedosta on siis eniten hyötyä kun se on saatavilla oikea-aikaisesti. Tsoukas (2009)[25] pitää tärkeänä, että uuden tiedon synnyttämiseksi keskustelun apuna käytetään jotakin prosessia. Haahtelan ohjausmallissa tähän tarkoitukseen hyödynnetään mm. aiemmin esiteltyjä iteratiivisia palautekierroksia, Axiomatic design ja C-K teoria prosesseja asiakkaan tarpeiden kartoittamisessa. Tällä tavalla uutta tietoa voidaan luoda hallitusti ja sen arvoa on mahdollista mitata. Arvon mittaaminen on tärkeää, sillä ohjaaminen on mahdollista vain, mikäli ohjausprosesista voidaan kerätä tietoa, jonka pohjalta prosessia voidaan muuttaa (Ballard et al. 2013)[6].

Ohjelmointi- ja suunnitteluvaiheissa eri toimijat tuottavat omaan tahtiinsa lisää tietoa projektista, jota he jakavat keskenään. Esimerkiksi projektinjohto kommunikoi asiakkaan kanssa tarpeista ohjelmointivaiheessa, tämän jälkeen jalostaa saamastaan tiedosta uutta tietoa, joka taas kerrotaan edelleen asiakkaalle. Tällaisen tiedon liikkuttelua eri toimijoiden välillä kutsutaan Haahtelan ohjausmallissa tietovirraksi. Tietovirroissa olennaista on sekä tiedon laatu, että tieto siitä, milloin mikäkin tieto on valmista jaettavaksi eteenpäin. Kuvassa 7 on kuvattu tiedon liikkumista eri toimijoiden välillä. BIM-domain kuvaa tietomallia, jonka rakentaminen aloitetaan vaatimusmallin luomisella ja tietomallia täydennetään projektin edetessä sitä mukaa kun tietovirroista saadun informaation määrä kasvaa. TAKU[®]-domain kuvaa BIM-domainin kanssa samankaltaista tietoa, mutta eri näkökulmasta: mitä tarkoittaa, että asiakas haluaa kokoushuoneen ja mitä sellainen tyypillisesti maksaa? TAKU[®]-domainin tieto perustuu tilaajan tavoitteista mallinnettuihin rakennuksen toiminnallisiin systeemeihin. TAKU[®] siis simuloi suunnittelijoiden päätöksiä luoden virtuaalirakennuksen ennen suunnittelijoiden BIMiin muodostamaa mallia. Laskelmat perustuvat aina nykyhetken tietoon materiaalien kustannuksista ja ne joudutaan aina laskemaan uudelleen projektikohtaisesti. Kaaviossa pyritään liikkumaan ylhäältä alaspäin, kuitenkin niin, että tietoa liikkuu jatkuvasti edestakaisin BIM-domainin ja TAKU[®]-domainin välillä. Tiedon määrä ja tarkkuus kasvavat alaspäin siirryttäessä ja saman domainin sisällä on olemassa aikaisemman vaiheen informaatio.

Prosessi alkaa tarvearvioinnista, jonka tarkoituksena on määrittää asiakkaan tarpeet ja selvittää minkälaisia tilavaatimuksia asiakkaan tarpeista voidaan generoida. Tässä vaiheessa tätä tietoa käytetään projektin odotettujen kustannusten määrittämiseen. Kun asiakkaan tarpeet on saatu määritettyä, tarpeiden pohjalta muodostetaan tilaluettelo, jota puolestaan voidaan hyödyntää rakennuksen toiminnallisten järjestelmien mallintamiseen ja sitä kautta myös kustannusten ennakkointiin. Tämän jälkeen suunnitteluvaiheessa siirrytään ehdotussuunnitteluvaiheeseen, jossa aloitetaan BIM-tietomallin luominen, jonka jälkeen etsitään arkkitehtien ratkaisujen



Kuva 7: Tietovirrat eri toimijoiden välillä rakennuksen elinkaaren alkuvaiheessa

joukosta hyviä ratkaisuja asiakkaan tarpeiden täyttämiseksi. Ehdotussuunnitelmassa esitettyjä toiminnallisia järjestelmiä verrataan TAKU[®]-mallinnuksessa saatuihin toiminnallisiin järjestelmiin. Jos ehdotettu järjestelmä on selkeästi kalliimpi kuin TAKU[®]-mallin tuottama, projektinjohto tietää tavoitteellisen kustannuksen ylittyvän tältä osin. Kustannusylityksestä tulee antaa välitön palaute suunnittelijoille. Yhteistyössä suunnittelijoiden ja projektinjohton kanssa päätetään, hyväksytäänkö ylitys ja kompensoidaan jossakin toisessa järjestelmässä vai muutetaanko suunnitteluehdotusta. Tätä tietoa voidaan siis käyttää arvioitaessa suunnitelman kokonaiskustannusta ja tarpeita suunnitelman muutoksille. Suunnitteluvaiheessa tietoa liikkuu erilaisina tietovirtoina projektinjohton sekä suunnittelijoiden välillä ja lopulta niiden pohjalta saadaan valmis suunnitelma, jossa rakennus on valmiina rakennuskomponentteja myöten. Tämän jälkeen rakentaminen voidaan aloittaa rakennusvaiheessa.

Tieto suunnitelman osion valmiudesta on olennaista, kun halutaan arvioida eri suunnitelmia ja vertailla niitä keskenään. Yleiset tietomallivaatimukset voivat toimia keinona tietää, milloin tieto on valmiina siirrettäväksi BIM domainista TAKU[®]-domainiin. Tietoa ei voi siirtää liian aikaisin, koska suunnittelu on luonteeltaan kumulatiivista. TAKU[®]-malli sisältää esimerkiksi kaikki tarvittavat väliseinät, mutta BIM vain osan (valmistuskeittiö on kuvattu vyöhykkeenä maantasokerrokseen, mutta kuivavarastoja ja tavaran vastaanottoa ei vielä ole erotettu omiksi tiloikseen keittiövyöhykkeessä). Näiden kahden domainin vertailu olisi siis harhaan johtavaa. Toisaalta kokonaisprosessin nopeuttamiseksi tieto täytyy saada mahdollisimman nopeasti jaettua sitä tarvitseville tahoille sen valmistuttua. Tällä hetkellä suunnittelijat ja rakennuttaja kommunikoivat ja sopivat milloin mitäkin tietoa voidaan siirtää.

3.7 Tietoformaatit tietovirtojen hallinnassa

Talonrakentamisessa olennaisen säilömiseen ja tiedon siirtämiseen toimijalta toiselle on kehitetty erilaisia tiedonsiirtoformaatteja. On olemassa patentoituja ja avoimia tiedostoformaatteja. Patentoidut formaatit eivät todennäköisesti sovellu Haahtelayhtiöiden käyttöön, sillä niitä ei ole oikeutta käyttää ilman lupaa. Patentoidut formaatit eivät ole tämän vuoksi mielekäs tutkimuskohde. Osa formateista on käytössä, vain patentin haltijan sisäisessä käytössä. Esimerkkejä patentoiduista formateista ovat mm. RVT (Autodesk Revit files), NWD (Autodesk naviswork files) sekä DWG (Autodesk format for AutoCAD).

Avoimia formateja on muutamia vaihtoehtoja, joista yleisimmin käytetyt ovat IFC ja COBie. IFC on COBie:ta hieman laajempi ja COBie onkin tavallaan osa IFC:tä. Näitä formateja voidaan käyttää rinnakkain: COBie:ssa tieto on ihmiselle helpommin ymmärrettävää ja toisaalta IFC sisältää mallintamiseen tarvittavan tiedon. COBie on käytännössä karsittua IFC:tä ja COBie:n pohjalle on mahdollista rakentaa IFC-formaatissa oleva tietomalli.

3.7.1 Ohjelmointivaihe ja COBie

TAKU[®] tuottaa suuren määrän asiakkaalta kerättyä tietoa (tilat, tilaryhmät ja yli sata tilan ominaisuutta joka tulisi tuottaa tilaajalle tilaajan hyväksymään hintaan).

Tällä hetkellä suunnittelijat joutuvat lukemaan suunnittelun lähtötietona olevat parametrit joko paperitulosteina tai tieto tulee lukea käyttäen TAKU[®]-applikaattia.

Tilaluetteloita ei laadita aina TAKU[®]-applikaatilla. Tilajaat voivat käyttää tiedon kokoamiseen esimerkiksi taulukko-ohjelmistoja. Taulukko-ohjelmistot eivät kuitenkaan mallinna tietoa budjetiksi tai rakennuksen virtuaalimalliksi. Tällöin tilaohjelman siirto TAKU[®]-ohjelmistoon tehdään manuaalisesti. Ajan hukan vuoksi tulisi tieto siirtää TAKU[®]:un digitaalisesti.

COBie on IFC:tä yksinkertaisempi ja sen tarkoitus on helpottaa eri sidosryhmien ymmärrystä rakennettavasta kohteesta. COBie voisi siis olla hyödyllinen ohjelmointivaiheessa asiakkaan kanssa kommunikoinnissa. “The format is intended to be easy to manage by any organisation, irrespective of size and IT capability. Its simplicity means that all tiers of the supply chain should be able to contribute to the data set, even if just by entering it directly into the spreadsheet.”(IFC-dokumentaatio)[26]. COBie:n yksi tarkoitus on siis pyrkiä vähentämään rakennusprojektin tarpeeton kompleksisuutta, teknologioiden vaihdoksista aiheutuvia ongelmia, järjestelmien yhteensopivuusongelmia sekä patentoitujen ohjelmistojen tuomia lisähaasteita. COBie:ta voisi siis käyttää formaattina ohjelmointivaiheen tiedon jakamiseen.

Tällä hetkellä ohjelmointivaiheen vaikeimmat ongelmat ovat tiedossa ja ne liittyvät asiakkaan kanssa kommunikointiin, asiakkaan tarpeiden määrittämiseen toiminnoiksi ja edelleen tiloiksi sekä kootun tiedon välittämiseen suunnittelijoille.

3.7.2 Suunnitteluvaihe ja standardoidut tietomalliformaatit

Rakennusten tietomalleissa IFC-formaatti (Industry Foundation Classes) on muodostunut yleiseksi standardiksi: ISO/PAS 16739. IFC:tä kehittää IAI-järjestö joka tunnetaan paremmin nimellä BuildingSMART. (IFC-dokumentaatio)[26] IFC-formaatti voidaan jakaa edelleen kahteen erilaiseen formaattiin: ifcXML ja ifcZip. IfcXML perustuu XML merkintään, jossa standardista IFC:stä konvertoidaan XML:ää ja ifcZip puolestaan on Zip tiedosto, joka muodostetaan standardin IFC:n tai ifcXML:n pohjalta. IfcXML ja ifcZip siis sisältävät saman informaation, mutta tiedosto on eri formaatissa. IFC formaatin kehitys on aloitettu vuonna 1994 ja sitä on alettu käyttää laajemmin vuonna 2000, IFC4 julkaistiin vuonna 2013 ja se on edelleenkin uusin versio IFC:stä. IFC-tiedostoformaatti voi olla muotoa .ifc, .ifcXML tai .ifcZIP.[26]

IFC-formaatin avulla tietoa voidaan siirtää järjestelmästä toiseen käytettävää ohjelmistosta riippumatta. Tämä on erittäin hyödyllistä rakennusprojektien eri vaiheissa, esimerkiksi suunnittelu- ja tuotantovaiheissa. Moni markkinoilla oleva järjestelmä ei tosin sisällä työkalua tiedon siirtämiseen järjestelmästä ulos päivitettyyn IFC-formaattiin, mikä aiheuttaa mm. ongelmia tiedon ajantasaisuudessa eri järjestelmissä. Samat muutokset täytyy tämän vuoksi tehdä moneen järjestelmään. Dokumentaatio IFC-formaatista löytyy BuildingSmartin sivuilta[26]. Dokumentaatio sisältää tiedot IFC-formaatin käyttöä varten. IFC-formaatista tulisi luoda yhteys Hahtela-nimikkeistöön, joka TAKU[®]:ssa on käytössä ja tarkasti miettiä, mitkä tiedot tarkoittavat mitään, jotta rakennuksen järjestelmiä ja siten myös kustannuksia voitaisi laskea suoraan rakennuksen IFC-mallista.

Yleiset tietomallivaatimukset tarjoavat 4 eri tapaa tuottaa IFC-formaatin tiedos-

toja (BuildingSMART, 2012)[27]:

1. Pääjärjestelmät mallinnetaan omina malleina **kerroksittain**
2. Pääjärjestelmät yhdistetään yhdeksi **kerroskohtaiseksi malliksi**
3. Pääjärjestelmistä tehdään omat, **erilliset koko kiinteistön kattavat mallit**
4. Pääjärjestelmät yhdistetään yhdeksi, **koko kiinteistön kattavaksi malliksi**

Valitusta tavasta riippumatta, IFC-tiedostojen tekovaiheessa kaikkien objektien tulee noudattaa IFC-hierarkiaa. Tämä tarkoittaa sitä, että kukin suunnitelman objekti sisältää tiedon siitä, mihin rakennukseen, kerrokseen ja osajärjestelmään se kuuluu. (BuildingSMART, 2012)[27]

Tällä hetkellä IFC-mallien ja Haahtela-nimikkeistöjen välillä on joitakin ongelmia, kuten tietojen linkitys eri tietoihin IFC-formaattissa, tilojen pinta-alan tai tilavuuden laskeminen tai seinien, lattioiden sekä kattojen materiaalien linkitykset. Suunnittelijat pystyvät kuitenkin jo nyt tuottamaan IFC-formaattia, mutta TAKU[®]:un ei ole vielä tällä hetkellä mahdollista siirtää tietoa automaattisesti IFC-formaatin avulla. Suunnitteluvaiheessa standardoidun tietomallin käyttäminen mahdollistaisi tiedon liikuttamisen IFC-formaatista suoraan TAKU[®]:un. Autocadilla pystyy tuottamaan IFC-formaatissa olevia tietomalleja. Tällöin tietoa voitaisi automaattisesti siirtää suunnittelijan luoman CAD-tietomallin pohjalta TAKU[®]:n, jossa hintatietoja voisi laskea entistä nopeammin. Iteratiivisessa prosessissa automaatiolla voidaan teoriassa säästää aikaa, eli tehostaa prosessia.

4 Tutkimus

Tutkimusvaiheessa selvitetään ensin tarkasti Haahtelassa talonrakentamisen projektinohjauksessa käytettävä kokonaisprosessi, Haahtelan ohjausmalli ja tämän jälkeen etsiä siitä LEAN-ajattelun mukaista hukkaa. Pohjana käytetään teoriaosuudessa esiteltyjä konsepteja ja tietoja sen jälkeen täydennettiin ja tarkennettiin haastattelujen pohjalta. Löytämällä hukkien syyt, hukat on mahdollista eliminoida prosessista. Haahtelan ohjausmallia tutkitaan kompleksisten prosessien hallinnan näkökulmasta, sillä löytämällä prosessien kompleksisia osia, on mahdollista löytää kompleksisuuden aiheuttamaa hukkaa. Kompleksisuuden hallintaan pyritään etsimään ratkaisuja jo olemassa olevien ratkaisujen lisäksi, hyödyntämällä tietomallinnusta sekä hallinnoimalla tietomallien ja TAKU[®]:n välisiä tietovirtoja. Haahtelan ohjausmallin prosessien selvittämisessä haastateltiin asiantuntijoita Haahtelan organisaatiossa:

- Ari Pennanen, Adjunct professor Tampere Technical University, partner Haahtela-yhtiöt
- Erkki Teittinen, Projektinjohto ja kehitys, tiiminvetäjä, Haahtela-yhtiöt
- Markus Mikkola, projektinjohto ja kehitys, Haahtela-yhtiöt

Haastattelut eivät olleet vain keskustelua, vaan keskustelun tukena luotiin kaavioita siitä, miten Haahtelan prosessit toimivat ja miten ne vertautuvat yleisten tietomallivaatimusten eri vaiheisiin. Lopulta keskityttiin myös määrittelemään, millaisia tietovirtoja BIM-domainin ja TAKU[®]-domainin välillä on. Haastattelujen pohjalta luotiin kaaviot tietovirroista eli kuvat: 7, 8 ja 9. Lisäksi haastatteluissa varmistettiin LEAN-osiossa löydettyjen hukkien olemassaolo sekä pohdittiin löydettyihin hukkiin mahdollisia ratkaisuja.

4.1 Haahtelan ohjausmalli ja kompleksisuus

Haahtelan ohjausmalli on Haahtela-kehitys Oy:n kehittämä malli talojen rakennuttamiseksi. Malli koostuu kolmesta vaiheesta:

1. Ohjelmointivaihe
2. Suunnitteluvaihe
3. Rakentaminen työmaalla

Mallia on kehitetty useiden vuosikymmenten ajan ja se toimii Haahtela-yhtiöiden pääliiketoiminnan tärkeänä perusosana. Haahtelan ohjausmalli on asiakaslähtöinen tapa toteuttaa tilakokonaisuuksia, joissa asiakkaat pystyvät suorittamaan omia liiketoimintansa kannalta olennaisia tehtäviä. Projektinohjaus lähtee liikkeelle asiakkaan tarpeiden määrittämisestä, jonka jälkeen edetään suunnittelijoiden ohjausvaiheeseen. Suunnittelun jälkeen rakennus rakennetaan ja luovutetaan asiakkaan käyttöön.

4.1.1 Tausta

Suh:n (Suh, 2005)[12] määrittelemät kompleksisuuden tasot on otettu huomioon Haahtelan käyttämässä projektinohjausprosessissa. Suh:n kuvaamaa ajasta riippumatonta aitoa kompleksisuutta esiintyy rakennusprojektin kaikissa vaiheissa. Tämä johtuu pääosin siitä, että rakennusprojektissa on luonnostaan monia eri toimijoita, jotka aiheuttavat lisää muuttujia projekteihin. Esimerkiksi ohjelmointivaiheessa asiakkaan tarpeiden määrittäminen ja linkittäminen erilaisiin toimintoihin on aitoa kompleksisuutta, jota pyritään hallitsemaan tietyillä prosesseilla ja toimintatavoilla. Ari Pennanen sanoo: “Ohjelmointivaiheessa on useita päätöksentekijöitä, kuten strateginen johto, operatiiviset johtajat, käyttäjät, joskus naapuritkin, jotka tarkastelevat tarpeita omasta näkökulmastaan.”

Kuvitteellista kompleksisuutta esiintyy rakennusprojekteissa jonkin verran, mutta sitä pyritään Haahtelan ohjausmallissa aktiivisesti vähentämään. Tällaista turhaa kuvitteellista kompleksisuutta näkyy esimerkiksi suunnitteluvaiheessa, jossa tehdään paljon manuaalista työtä. Manuaalisen työn aiheuttamaa kompleksisuutta voidaan pyrkiä vähentämään esimerkiksi erilaisilla tiedonsiirtoon ja säilömiseen tarkoitettuilla sähköisillä työkaluilla tai automatisoimalla manuaalisesti tehtäviä prosesseja.

Kombinatorista kompleksisuutta esiintyy tyypillisesti työmaavaiheessa. Tällöin koko rakennustyömaan aikataulu riippuu usean pienemmän toimijan aikataulujen pitävyydestä. Projekteissa tuleekin välillä viivästyksiä, kun jokin alihankkija ei pysty toimittamaan jotakin tuotetta sovitussa aikataulussa. Tällöin projekti viivästyy, sillä muut projektin toimijat ovat riippuvaisia myöhässä olevista projektin osista. Vaikutukset kokonaisaikatauluun vaihtelevat myöhästymisen kestoista ja myöhästyneiden komponenttien tärkeydestä riippuen.

Vaikka jokin projektin osa myöhästyy, se ei välttämättä vaikuta kokonaisprojektin keston, sillä myöhästymiset on usein mahdollista kuroa kiinni tietyllä aikavälillä: esimerkiksi teettämällä ylitöitä kun muu työ on tauolla (Suh, 2005)[12]. Rakennusprojekteissa on luonnollisia taukoja mm. öisin ja viikonloppuisin, jolloin aikatauluja voidaan kuroa kiinni tarvittaessa. Rakennustyömaalla esiintyy siis Suh:n kuvaamaa jaksottaista kompleksisuutta. Sekä kombinatorista että jaksottaista kompleksisuutta voidaan vähentää työmaalla kontrolloimalla ja hallinnoimalla työmaata. Aikataulujen hallintaan on Haahtelassa otettu käyttöön Takt- prosessi, jossa osapuolet pyritään sitouttamaan yhteiseen tuotantoon aikataulupakettien avulla.

Aitoa kompleksisuutta ja kuvitteellista kompleksisuutta esiintyy ohjelmointi- ja suunnitteluvaiheissa. Aito kompleksisuus on siis välttämätöntä kompleksisuutta. Ari Pennanen käyttää esimerkkinä asiakkaan kompleksisuutta: “Jos asiakas on kompleksinen, kompleksisuuden eliminointi tarkoittaisi asiakkaan poistamista.” On siis selvää, ettei kaikkea kompleksisuutta edes voida yrittää poistaa projekteista. Kuvitteellinen kompleksisuus puolestaan on turhaa kompleksisuutta. Aitoa kompleksisuutta pyritään vähentämään kasvattamalla ymmärrystä projekteista ja kuvitteellista kompleksisuutta pyritään vähentämään tehostamalla prosesseja ja muuntamalla kuvitteellista kompleksisuutta aidoksi kompleksisuudeksi, jota voidaan pyrkiä hallinnoimaan.

Haahtela-mallin kompleksisuus on luonteeltaan induktiivista kompleksisuutta (Pennanen, 2004)[1] (Rittel et al. 1972)[5]. Taulukossa 1 on vedetty yhteen ohjelmointi-

ja suunnitteluvaiheiden ongelmia ja niitä verrataan Rittelin ja Webberin (Rittel et al. 1972)[5] määrittelemään vaikeaan ongelmaan.

Ohjelmointi- ja suunnitteluvaiheiden ongelmat ovat siis Rittelin ja Webberin (Rittel et al. 1972)[5] määrittelemiä vaikeita ongelmia. Molempia vaiheita voidaan kuitenkin hallita ja ohjata, sillä kompleksisuutta voidaan pyrkiä tietoisesti vähentämään. Seuraavassa luvussa käsitellään ohjausteorian käytössä olevia prosesseja ohjelmointi- ja suunnitteluvaiheiden kompleksisuuden hallinnassa.

Wicked problem	Ohjelmointivaihe	Suunnitteluvaihe
Ongelman määrittely on vaikeaa	Asiakkaan osapuolet eivät ole yksimielisiä tarpeista.	Samaan asiakkaan tarvekokonaisuuteen on lukemattomia ratkaisuja.
On vaikea tietää, milloin ongelma on ratkaistu	Tiloihin uhratut investoinnit kilpailevat samoista varoista kuin muut liiketoimintaan uhratut investoinnit.	On vaikeaa tietää missä vaiheessa suunnitelma toteuttaa asiakkaan tarpeet riittävän hyvin.
Ratkaisut eivät ole "oikein" tai "väärin", pikemmin "hyviä" tai "huonoja".	Asiakkaan tarpeet voidaan tyydyttää monella eri tavalla.	Sama toiminto voidaan toteuttaa usealla eri tavalla.
Ratkaisua on vaikea testata.	Tietoa ei ole vielä riittävästi esimerkiksi rakennuksen mallintamiseksi.	Voidaan vertailla malleja, mutta valmista rakennusta ei vielä ole olemassa.
Ei ole mahdollista iteroida valmiita ratkaisuja.	Rakennuksen tietomallia ei ole vielä aloitettu rakentamaan, joten iterointi ei ole mahdollista.	Iterointia tehdään tietomallinnuksen pohjalta.
Ratkaisu ei ole yksikäsitteinen.	Asiakkaan tarpeet voidaan tyydyttää monella eri tavalla.	Asiakkaan tarpeet tyydyttävät tilat voidaan tehdä monella tavalla.
Ongelma on uniikki.	Asiakkaiden toiminnalliset tarpeet ovat asiakaskohtaisia.	Suunnittelijoiden suunnittelutapa on yksilöllistä.
Ongelma on oire muista ongelmista.	Muuttujia on paljon ja projektit ovat pitkäkestoisia. Kommunikointi asiakkaan ja projektinjohdon välillä aiheuttaa viiveitä.	Parhaan suunnitelman löytäminen suunnitelmien joukosta ei ole yksinkertaista.
Ongelman ristiriidat voidaan esittää usealla eri tavalla.	Ongelmat ovat erilaisia projektinjohdon ja asiakkaan näkökulmista.	Projektin toimijoilla saatavaa olla toisistaan poikkeavat tavoitteet ja omat intressit.
Ongelman ratkaisijalla ei ole oikeutta olla väärässä.	Asiakas luottaa projektinjohdon arvioihin hankkeen kustannuksista sekä asiakkaan tarpeiden täyttymisestä.	Suunnitelman kustannukset ovat projektinjohdon vastuulla. Toteutuneet kustannukset ovat selvillä vasta kun rakennus on rakennettu valmiiksi ja luovutettu asiakkaalle.

Taulukko 1: Ohjelmointi- ja suunnitteluvaiheet verrattuna Rittel et al. (1972)[5] määrittämään vaikeaan ongelmaan

4.1.2 Prosessit

Haahtelan ohjausmallin tärkeimmät prosessit ovat ohjaukseen liittyvät ohjelmointi- ja suunnitteluvaiheet, sekä suunnitelman täytäntöönpano rakennustyömaalla. Tutkimuksessa keskitytään Haahtelan ohjausmallin ohjelmointi- ja suunnitteluvaiheisiin, sillä niissä on eniten ohjausta vaativia kompleksisia elementtejä. Lisäksi työmaavaihe on jo kohtalaisen kattavasti tutkittu.

Ohjelmointivaiheen tärkeimmät tavoitteet ovat asiakkaan kanssa yhteisen kielen löytäminen, asiakkaan tarpeiden kartoittaminen ja jalostaminen, sekä asiakkaan toimintojen määrittely tarpeiden pohjalta. Ohjelmointivaiheessa selvitetään myös projektin toteuttamisen kannalta tärkeät tunnusluvut kuten: asiakkaan sallittu kustannus eli allowable cost, odotettu kustannus eli expected cost sekä tavoitehintaa eli target cost. Asiakkaan tarpeiden määrittämisessä hyödynnetään Axiomatic design ja C-K teorian käytännön sovelluksia. Käyttämällä prosessia, voidaan paremmin varmistaa, että kaikki tuotettu tieto kerätään talteen.

Suunnitteluvaiheen tärkeimmät tavoitteet ovat saada tiedot rakennuksen muodosta sen tulevassa ympäristössä, asiakkaan aktiviteettien sijainneista ja yhteyksistä toisiinsa sekä rakennuskomponentit ja materiaalit. Jotta tämä on mahdollista, Haahtelan ohjausmallissa tarjotaan suunnittelijoille tilaluettelo, tilojen ominaisuudet ja tavoitehintaa, johon suunnittelijan tulee pyrkiä. Suunnittelijat suunnittelevat saamiensa tietojen mukaan omat suunnitelmansa, joita arvioidaan TAKU[®]:n avulla. Haahtela-yhtiöt arvioi suunnitelmien laatua ja vertailee sitä hinnan funktiona. Projektin tavoitehintaa määrittää sopivan ohjausvälin ylärajan: suunnitelmat, jotka ovat kustannuksiltaan tavoitehintaa korkeampia ei kannata valita jatkokehittäväksi, sillä asiakkaalla ei ole riittävää maksukykyä eikä -halukkuutta.

Ohjausväliltä valitaan sopivimmat suunnitelmat jatkokehittäväksi. Parhaan suunnitelman löytämiseksi käytetään aiemmin esiteltyä ”puolustava mestari” ajattelua. Haastattelussa Ari Pennanen sanoo: ”TAKU[®]:n tuottama toiminnallisen järjestelmän suunnitelma ja kustannus on oikea, kunnes suunnittelijan tuottama tieto haastaa sen. Jos suunnittelijan tuottama järjestelmä on kalliimpi kuin TAKU[®]-mallin järjestelmä, suunnittelijan tulee osoittaa kompensatio muista järjestelmistä tai suunnitella halvempi järjestelmä ennen kuin suunnittelijan järjestelmä korvaa TAKU[®]:n puolustavan mestarin.”

4.1.3 Kehitys

Systemin monimutkaisuus kasvaa systeemin muuttujien määrän lisääntyessä. Yksi LEAN-ajattelun hukkaa aiheuttavista tekijöistä on muuttujien suuri määrä systeemisissä (Ohno, 1988)[16]. Muuttujien määrän lisäksi eräs merkittävä tekijä kompleksisuuden kasvattajana on viiveet prosessissa sekä tiedon käsittelyyn ja ymmärtämiseen kuluva aika (Ohno, 1988)[16]. Kompleksisuuden voidaan siis sanoa kasvattavan hukkaa prosessissa. LEAN-ajattelussa yksi hukan vähentämisen keino on prosessien automatisointi niin pitkälle, kuin se on mahdollista. Tällä tavalla prosessit toimivat nopeammin ja lisäksi ihmislähtöisten virheiden määrä laskee systeemisissä.

Iteratiivisessa prosessissa Haahtelan ohjausmallissa voidaan automatisoida tietomallien lukemista Haahtelan TAKU[®]-tietomalliin. Luomalla rajapinnan TAKU[®]-

ohjelmiston ja IFC-formaatin välille on mahdollista lukea arkkitehtien IFC-formaatissa tallentamia suunnitelmia suunnitteluvaiheessa vertailtaessa suunnitelmia keskenään. Lisäksi nopea tiedon siirtäminen nopeuttaa valittujen suunnitelmien parantelua. Manuaalisen työn vähentäminen suunnitteluvaiheessa pitäisi lyhentää suunnitteluvaiheen kestoja ja täten nopeuttaa koko rakennusprojektia. Toisaalta, mikäli kestoja ei haluta lyhentää, saattaisi tehokkaampi tiedonsiirtäminen mahdollistaa useamman iteraation samassa ajassa kuin aiemmin.

4.2 Tietovirrat ja niiden hallinta Haahtelan prosesseissa

Ohjelmointi- ja suunnitteluvaiheissa on paljon tietovirtoja, joiden hallinta on välttämätöntä. Molemmissa vaiheissa Haahtelan toimintatapaa verrataan yleisten tietomallivaatimusten ja tietomallintamisen näkökulmaan. Tämä on tärkeää, sillä Haahtela TAKU[®]:n sisältämän tiedon sekä tietomallin sisältämän tiedon tulisi olla ajantasaisia ja järjestelmien käytön tulee tukea toisiaan. Tietovirtoja tarkastellessa tärkeää on kiinnittää huomiota seuraaviin tekijöihin:

1. Tiedon täytyy olla saatavilla oikeaan aikaan
2. Tiedon tarkkuus täytyy olla kutakin vaihetta edellyttävällä tasolla
3. Tietoa täytyy olla helppoa jakaa ja vastaanottaa

Projektissa mukana olevat tahot sopivat keskenään käytettävät työkalut, tietomallit, tiedon tarkkuustasot, aikataulut ja muut olennaiset työskentelytavat. Projektijohto voi arvioida sovittavien asioiden kokonaisuutta tietovirtojen hallinnan näkökulmasta ja yrittää kehittää yleistä prosessia siitä, miten tietovirtoja voisi tehokkaasti hallinnoida projektin eri vaiheissa.

4.2.1 Ohjelmointivaihe

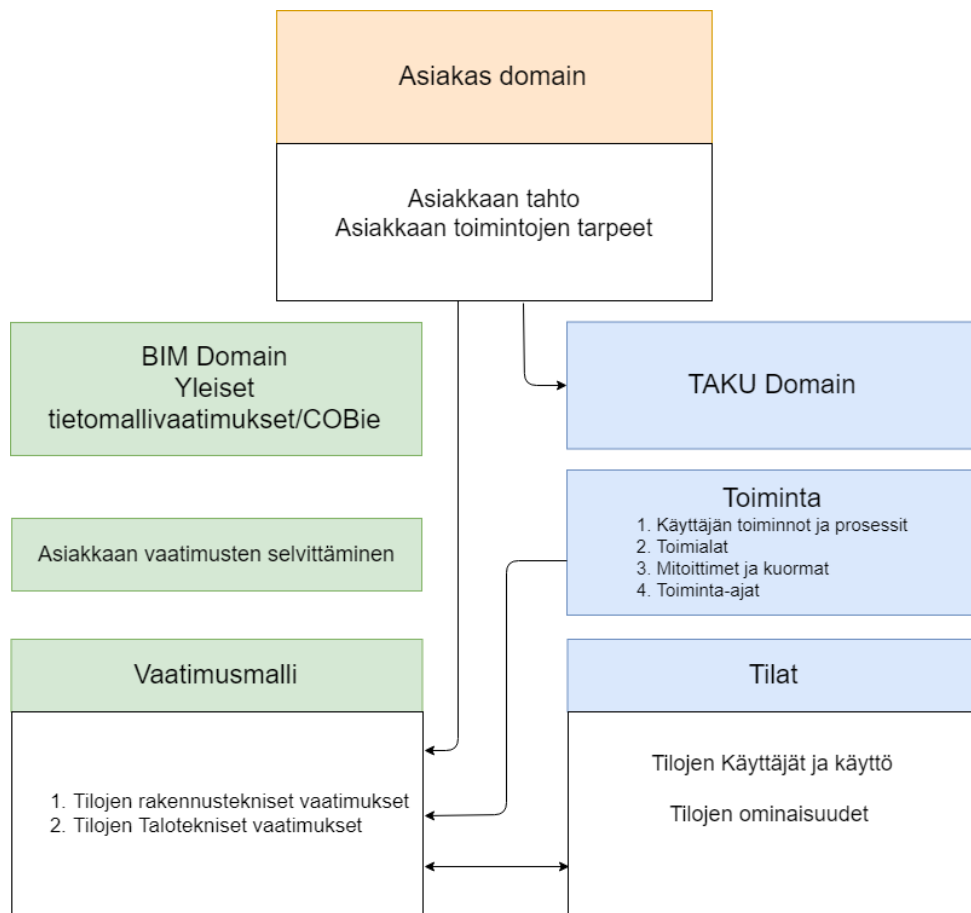
Ohjelmointivaiheen olennaisin tavoite on kartoittaa asiakkaan tarpeet ja niitä vastaavat toiminnot sekä tämän jälkeen muuntaa toiminnot sopiviksi tiloiksi. Ohjelmointivaiheen tuloksena saadaan tilaluettelo sekä hankkeen tavoitehinta. Tärkeimmät hallittavat tietovirrat ohjelmointivaiheessa ovat asiakkaan ja projektijohdon välinen kommunikaatio sekä TAKU[®]-domainin ja BIM domainin välinen tiedonkulku. Tärkeimmät ohjelmointivaiheen tietovirrat on kuvattu kuvassa 8. Asiakkaan ja projektijohdon välillä tietovirrat ovat luoteeltaan sellaisia, että tarve jollekin tilalle syntyy asiakkaan tahdon pohjalta ja projektijohto jalostaa asiakkaalta saamaansa tietoa, joka kerrotaan takaisin asiakkaalle. Prosessi on siis iteratiivinen. Kuvassa 5 on esimerkki siitä, miten projektijohto kommunikoi asiakkaan kanssa kokoushuoneen tarpeesta ja siitä miten asiakkaan tahto vaikuttaa tulevan tilan ominaisuuksiin. Tietovirtojen hallinnassa on ohjelmointivaiheessa haasteita monesta eri syystä kuten:

1. **Asiakkaan ja projektijohdon välinen kommunikointi.** On tärkeää varmistaa, että asiakas ymmärtää projektijohtoa ja toisaalta, että projektijohto ymmärtää asiakkaan tarpeen oikein.

2. **Asiakas ei välttämättä tiedä mitä jonkin tarpeen toteuttamien vaatii käytännössä.** Projektinjohdon tehtävänä on varmistaa, että asiakkaan tarve on mahdollista toteuttaa. Haastatteluihin käytiin läpi seuraava esimerkki: Asiakas haluaa sairaalaan leikkaussalin ja antaa samalla ehdon, että leikkaussaliin pitää aina tulla sähköä, vaikka muu kaupunki olisi ilman sähköä. Tämä tarkoittaa käytännössä sitä, että sairaalassa on oltava riittävä määrä omia tilojaan pelkästään generaattoreille, joilla sairaalan leikkaussalien sähkönsaanti voidaan taata tietyllä aikavälillä. Tämä taas edellyttää varoja investointivaiheessa.
3. **Asiakas ei välttämättä tiedä mitä haluaa.** Asiakas saattaa luulla tarvitsevansa jonkin tietyn ratkaisun ja ei ole ajatellut, että ongelmaan voisi olla myös muita ratkaisuja. Haastatteluihin käsitelty esimerkki: Asiakkaan aikaisemmassa sairaalassa on kolme leikkaussalia, joten asiakas luulee tarvitsevansa kolme leikkaussalia myös uudessa sairaalassa. Sama määrä leikkauksia voidaan mahdollisesti toteuttaa myös kahdessa leikkaussalissa, mikäli leikkaussalien käyttöastetta parannetaan. Kolmannesta salista luopuminen avaisi mahdollisuuden investoida johonkin muuhun arvokkaaseen. Toinen esimerkki on kuvassa 6, jossa asiakas haluaa itselleen kokoushuoneen, vaikka etäpalaverien mahdollistaminen ratkaisi ongelman yhtä hyvin.

Asiakkaan toimintojen määrittämisen jälkeen voidaan luoda tilaluettelo, joka Haahtelan TAKU[®]-mallissa sisältää tiedon tilojen käyttäjistä sekä tilojen ominaisuuksista. Haahtelan ohjausprosessissa asiakkaiden tarpeiden määrittämisen apuna iteraatioissa käytetään C-K teorian ja axiomatic design periaatteiden mukaisia tapoja. BIM-domainissa vaatimusmalli sisältää tilojen rakennus- ja talotekniset tiedot. Yleisten tietomallivaatimusten mukaan vaatimusmalli on taulukko muodossa oleva tilaluettelo. BIM-domainissa tämä tarkoittaa esimerkiksi COBie:n hyödyntämistä. Vaikka BIM-domainin vaatimusmalli ja TAKU[®]-domainin tilaluettelo sisältävät käytännössä saman tiedon, eli tiedon rakennukseen tulevista tiloista ja vaatimuksista, tieto on jäsennelty eri tavalla. Siinä missä BIM-domainissa tilat kuvataan rakennus- ja taloteknisten vaatimusten mukaisesti TAKU[®]-domainin tilaluettelossa tilojen käyttäjät, käyttö ja tilojen ominaisuudet sekä suhteet toisiinsa ovat keskiössä. Mikäli tietoa halutaan tässä vaiheessa liikuttaa BIM-domainista TAKU[®]-domainiin tai päin vastoin, täytyy ensin selvittää perinpohjaisesti miten tiedot ovat kuvattuna esimerkiksi COBie:ssä ja mikä on mitäänkin COBie:n tietoa vastaava tieto Haahtelan nimikkeistössä. Jotkin tiedot ovat selvästi mahdollista siirtää BIM-domainista TAKU[®]-domainiin, sillä ne sisältävät samankaltaista tietoa. Esimerkiksi BIM-domainin sisältämästä tiedosta sisäilman olosuhteet, ääneneritys ja valaistus ovat tiedossa myös TAKU[®]-domainissa kun asiakkaalla on tarve kokoushuoneelle.

Haastattelussa Ari Pennanen kertoo: “Tiedonsiirrolle on selkeä tarve molempiin suuntiin, TAKU[®]:sta BIMiin ja päinvastoin. Jos hankkeen ohjelmointi on laadittu TAKU[®]:lla, tuloksena on tilaluettelo, tilojen ominaisuudet sekä tavoitehinta. Tilojen ominaisuuksia on yli sata (korkeus, tarve vesipisteeseen, häiriötön sähkönsyöttö, lämpötilan hallinta. . .). Rakennusalalla on yleistä, että tekstitiedossa muodostettu luettelo ominaisuuksista johtaa tiedon hukkamiseen ja sitä kautta laadun menetykseen. Digitaalisesti siirretty tilatieto TAKU[®]:sta suunnittelijoille varmistaisi, ettei



Kuva 8: Tietovirrat ohjelmointivaiheen aikana

tietoa katoa.”. Myös muut asiantuntijat vahvistivat tarpeen tietomallien lukemisesta digitaalisesti ja automaattisesti TAKU[®]:uun.

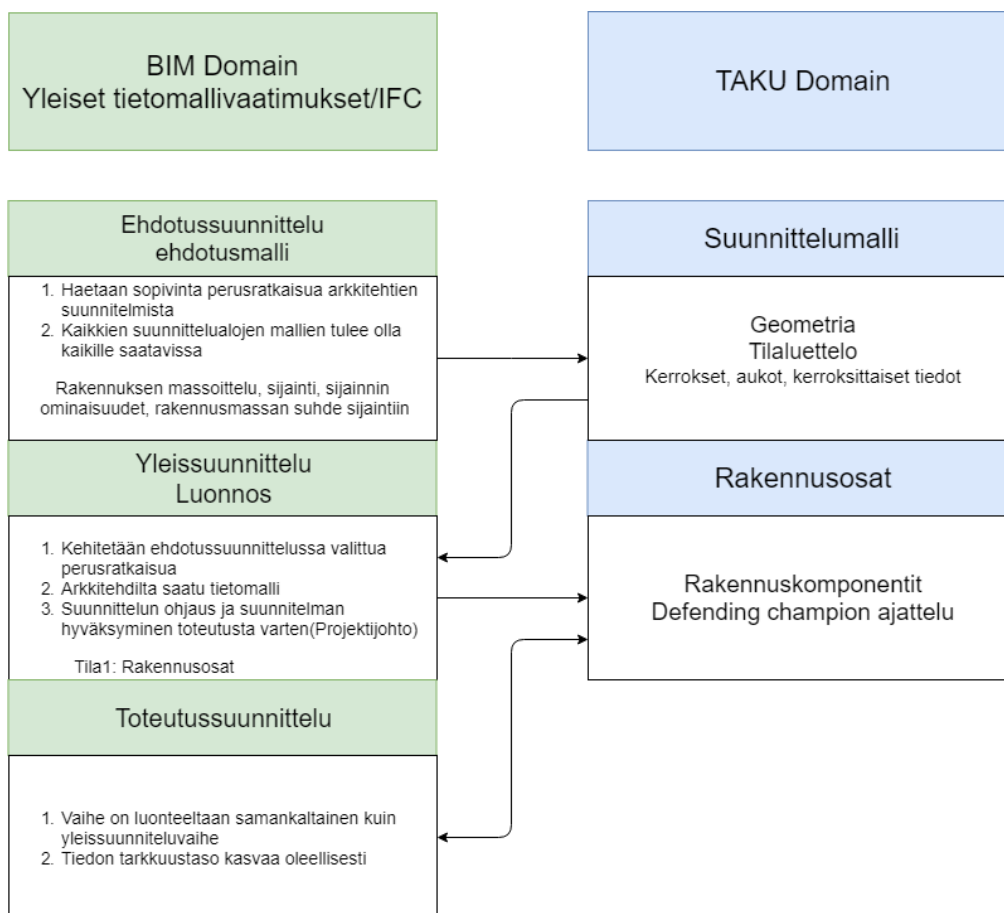
Toisaalta usein ohjelmointia ei tehdä TAKU[®]:lla, vaan tilaluettelo ja tilojen ominaisuudet tuotetaan muutoin. Jos tämä tieto pystyttäisiin siirtämään TAKU[®]-maliin, pystyisi TAKU[®] tuottamaan ehdotuksen hankkeen tavoitehinnaksi.

4.2.2 Suunnitteluvaihe

Suunnitteluvaiheen alussa on olemassa ohjelmointivaiheessa saatu tilaluettelo. Kuvassa 9 on kuvattu tietovirrat suunnitteluvaiheen aikana BIM-domainissa sekä TAKU[®]-domainissa. Molemmissa domaineissa ideana on kuvata tietomalliin sellaiset rakennusmarkkinoilla olevat toiminnalliset järjestelmät, joilla asiakkaan toiminnalliset tarpeet voidaan tuottaa. BIM-domainissa siirrytään ensin ehdotussuunnitteluvaiheeseen, jossa luodaan niin kutsuttu ehdotusmalli. COBie vaihtuu IFC:hen ja tietomallin rakentuminen alkaa. Ehdotusmallissa suunnittelija kuvaa asiakkaan toimintojen sijainnit rakennuksen kolmiulotteisessa mallissa, josta muodostuu rakennuksen kokonaismassa (kerroslukumäärä, kerrosten koot, rakennusten piirit, kerrosten korkeudet) annettuun tonttiin ja ympäristöön. Ehdotussuunnittelu on siis luonteeltaan suunnittelua tilaajalle. TAKU[®]-domainissa vastaavaa tietoa käsitellään suunnittelumallikomponentissa, jossa algoritmit määrittävät tilojen ominaisuuksien perusteella kerroslukumäärän, kerrosten koot, kerrosten piirit ja kerroskorkeudet. Tilojen täsmällinen sijainti ei TAKU[®]-mallissa määriy.

Yleissuunnittelu- ja toteutussuunnitteluvaiheissa suunnitellaan sellaiset järjestelmät, joilla asiakkaan toiminnalliset vaatimukset voidaan tuottaa (lämmön hallinta, äänen eristys...) ja suunnitellaan rakennuksen täsmällinen mittamaailma. Suunnittelu on luonteeltaan suunnittelua urakoitsijoille. Haahtelan asiantuntijoiden mukaan BIM-domainissa tämä vaihe valmistuu järjestelmä kerrallaan. Paaluja ei voi suunnitella ennen kuin anturat on suunniteltu, ilmastointikoneiden sähköistystä ei voi suunnitella ennen kuin ilmastointikojeet ja niiden teho on suunniteltu. TAKU[®]-domainissa puolestaan siirrytään suunnittelun simulaation kautta kaikkien järjestelmien mallintamiseen valmiiksi. TAKU[®]:n kumulatiivisuus on algoritmista ja sen vuoksi kestoja ei lasketa kuukausissa vaan sekunneissa. Yleissuunnitteluvaiheessa on tarkoituksena kehittää valittua suunnitteluratkaisua yhteistyössä arkkitehdin ja muiden suunnittelijoiden kanssa. Projektinjohto ja suunnittelijat sopivat muutoksista suunnitelmaan ja samaan aikaan luotuja tietomalleja täydennetään sovittujen asioiden pohjalta. Tietomalleja ylläpidetään prosessin aikana säännöllisesti ja tietomallin sisältämä tieto kasvaa projektin edetessä. Yleisten tietomallivaatimusten mukaisesti kaikki tietomallit tulee säilyttää yhteisessä projektipankissa, jossa ne ovat helposti kaikkien suunnittelijoiden saatavilla. Yleiset tietomallivaatimukset suosittelevat, että kaikki suunnittelijat mallintavat rakennukset kerroksittain, vaikka mallinnusohjelmat tukevat myös muita mallinnustapoja.

Rakennuskomponenttien määrittämisessä hyödynnetään “puolustava mestari” ajattelua, eli suunnittelun ohjaus on dialogia TAKU[®]-mallin tuottamien komponenttien ja suunnittelijoiden tuottamien komponenttien sekä näiden kustannusten välillä. Suunnitelmaa parannellaan iteratiivisesti siihen saakka, kunnes se toteuttaa



Kuva 9: Tietovirrat suunnitteluvaiheen aikana

asiakkaan toiveet ja on sopivassa hintahaarukassa. Koska aina hyväksytty suunnitelma korvaa TAKU[®]:n tiedon, lopulta molemmat mallit ovat identtisiä, ne tuottavat asiakkaan määrittelemän tarpeen ja vastaavat asiakkaan määrittämää tavoitehinta. Suunnitelman kustannuksia analysoidaan Haahtela TAKU[®]-ohjelmistolla. Tällä hetkellä asiantuntijat Haahtelassa mittaavat komponenttien määriä tietomallissa ja syöttävät tietoa TAKU[®]:un manuaalisesti.

Yleissuunnittelun tuloksena saadaan karkea arvio siitä, millaisista rakennuskomponenteista tuleva rakennus koostuu. Toteutussuunnitteluvaihe on myös luonteeltaan iteratiivinen ja yhteistyö projektinjohdon ja suunnittelijoiden välillä jatkuu. Tiedon tarkkuus kasvaa olennaisesti toteutussuunnittelun aikana ja toteutussuunnitteluvaiheen jälkeen tietomallissa on kaikki rakennuksen rakentamiseen tarvittava tieto. Suunnitteluvaiheeseen liittyy BIM-domainin ja TAKU[®]-domainin välillä joitakin eroavaisuuksia, mutta pääpiirteittäin molemmissa domaineissa pyritään samankaltaiseen päämäärään samankaltaisilla keinoilla: suunnitella ja rakennuttaa rakennus, joka koostuu asiakkaan tarpeiden mukaisista tiloista ja komponenteista. Suurimmat haasteet BIM-domainin ja TAKU[®]-domainin välisten tietovirtojen hallinnassa haastattelujen perusteella ovat:

1. Yleiset tietomallivaatimukset painottavat käyttämään tietomalleissa tapaa, jossa kaikki tietomallit on suunniteltu kerroksittain. TAKU[®]-domainissa parempi lähestymiskulma olisi, että pääjärjestelmistä tehdään omat, erilliset koko kiinteistön kattavat mallit.
2. TAKU[®]-malli on sikäli dynaaminen, että lähtötiedoissa tehty muutos laukaisee uuden simulaation kokonaan. Jos tilaohjelmaan lisätään auditorio ja sauna-kompleksi, kaikki järjestelmät mitoitetaan uudelleen. Tämä sinällään vastaa suunnittelun normaalia logiikkaa, mutta voi hämmentää rakennuksen kustannuslaskijoita, ja se tulee ottaa huomioon, kun TAKU[®]:n rajapintoja avataan tulevaisuudessa IFC-tiedostojen käsittelyä varten.
3. On haastavaa tietää, milloin kenenkin suunnittelijan tietomalli on sellaisessa vaiheessa, että sitä on järkevää käyttää pohjana muiden suunnitelmien tekemiseksi. Esimerkki: väliseiniä ei kannata Haahtelan asiantuntijoiden mukaan viedä TAKU[®]:un kovin aikaisessa vaiheessa, sillä tieto saattaa muuttua projektin aikana ja se on epätarkkaa melko suuren osan suunnitteluvaiheesta. TAKU[®]-domainissa saattaa kuitenkin olla tiedossa suunnitelman väliseinien kokonaisuus ja hinta. Tässä kohtaa tietovirrassa BIM-domainin ja TAKU[®]-domainin välillä voisi siis olla mahdollista lukea BIM-domainin tietoja suoraan TAKU[®]:un ja päätellä, kuinka valmista tieto väliseinän määrästä on ja aikaisemmassa vaiheessa tietää, että kustannukset väliseinien suhteen ovat liian korkeat.

TAKU[®]:n ja IFC:n välille tulisi luoda yhteys. Tämä tarkoittaisi sitä, että erilaisilla parametreilla varustetut IFC-komponentit tulee kohdentaa Haahtelan nimikkeistön komponentteihin. Rajapinnan luominen on pitkä prosessi, sillä IFC-kirjasto on kohtalaisen laaja. Tämän lisäksi talonrakentamisessa käytettävät komponentit

vaihtelevat eri projekteissa, joten rajapintaa täytyy päivittää eri projekteihin sopivaksi. Ylläpitotyön määrää on tässä vaiheessa vaikeaa arvioida, mutta todennäköisesti ylläpitotyön määrä vähenee ajan myötä, kunhan suurin osa yleisimmin käytettävistä komponenteista on saatu lisättyä rajapintaan.

5 Tulokset

Tässä tutkimuksessa tutkittiin BIM-ajattelun sekä Haahtelassa käytettävän talonrakentamisen projektinohjausprosessin yhtenäisyyksiä ja sitä, miten Haahtelassa käytettävä projektinohjausprosessi vertautuu yleisiin tietomallivaatimuksiin. Tämä tieto on tärkeää, sillä se selvittää millaisia muutoksia tietomallien ja TAKU[®]:n välille tarvitaan, mikäli integraatiota tietomallien ja TAKU[®]:n välistä tiedonsiirtoa halutaan automatisoida.

5.1 Tietovirrat ja tiedon oikea-aikaisuus

Ohjelmointi- ja suunnitteluvaiheissa on molemmissa omia haasteitaan jo olemassa olevien tietovirtojen kanssa ja lisäksi BIM-domainin ja TAKU[®]-domainin väliset mahdolliset tietovirrat vaativat tietojen yhdistelyä, automatisointia ja hallintaa, jotta BIM-domainin tietoja voidaan tehokkaasti hyödyntää Haahtela-yhtiöiden käytössä.

Ohjelmointivaiheessa on tietovirtoja asiakkaan ja projektinjohdon välillä. Aluksi tietovirta käsittää enemmän asiakkaalta projektinjohdon suuntaan tulevaa informaatiota kun asiakas kertoo projektinjohdolle tarpeistaan. Projektinjohto kysyy asiakkaalta lisäkysymyksiä ja pyrkii näiden tarpeiden pohjalta luomaan sopivia toimintoja ja toiminnoille tiloja. Ohjelmointivaiheen loppuvaiheessa tietovirran kulku siirtyy enemmän projektinjohdolta asiakkaalle päin ja asiakas hyväksyy tai hylkää projektinjohdon ehdotuksia. Projektinjohdon tehtävänä on tukea asiakasta ja ohjata heitä tekemään omien tarpeidensa kannalta järkeviä päätöksiä tulevan rakennuksen suhteen.

Ohjelmointivaiheessa asiakkaan tarpeet TAKU[®]-domainissa saadaan jo tällä hetkellä kohtalaisen hyvin kuvattua toiminnoiksi ja edelleen tilaluetteloksi. Tilaluettelon ja tilojen ominaisuuksien siirtäminen BIM-malleihin vähentäisi tiedon hukkaa. Toisaalta TAKU[®]-järjestelmän ulkopuolella tehdyn tilaluettelon ja tilojen ominaisuuksien siirtäminen TAKU[®]-järjestelmään mahdollistaisi hankkeen tavoitehinnan määrittämisen TAKU[®]:n avulla.

Takun ominaisuusnimikkeistö on määrämuotoinen, siihen voi syöttää arvoja vain tietyllä tavalla (tilan lämpötila ei saa nousta yli 24 asteen). Muiden järjestelmien kuvaukset voivat olla vapaita (tilan lämpötila ei saa olla liian kuuma). Tiedonsiirron ratkaisuksi voidaan tutkia COBie:ta mikäli erot tiedon nimikkeistössä voidaan ratkaista.

Suunnitteluvaiheessa tietomallien automaattinen lukeminen TAKU[®]:un parantaisi iteraatioiden tehokkuutta sekä vähentäisi inhimillisten virheiden määrää prosessin aikana. Tietovirta BIM-domainin ja TAKU[®]-domainin välillä on jo tällä hetkellä olemassa, mutta tietovirtaa hallinnoidaan manuaalisesti kopioimalla tietoa tietomallista TAKU[®]:un.

5.2 Haahtelan ohjausteorian haasteet ja kehitysideat

Ohjelmointi- ja suunnitteluvaiheiden prosessit ovat pitkäkestoisia ja iteraatiota tehdään projektien aikana useita. Tämän kaltaisessa toistuvassa prosessissa myös pro-

Hukka	Ohjelmointivaihe	Suunnitteluvaihe
Ylituotanto		
Viallisten tuotteiden tai osien korjaaminen	Asiakkaan ohjaus epäonnistuu. Asiakas huomaa myöhemmin suunnitteluvaiheessa tarvitsevansa tiloja joihin asiakkaalla ei ole varaa	Mikäli suunnittelussa on käytetty tietoa, mikä ei ole riittävän valmista, suunnitelmia joudutaan korjaamaan
Materiaalien liikkuminen	Informaation liikkuminen tietovirroissa asiakkaan ja projektijohdon välillä. Asiakas ei välttämättä osaa määrittellä ongelmansa ja toisaalta asiakasta saatetaan ymmärtää väärin	Informaation liikkuminen tietovirroissa BIM-domainin ja TAKU [®] -domainin välillä. Tietoja siirretään tietomallista TAKU [®] :un manuaalisesti
Varastointi		Tietomallien ja muun tuotetun tiedon varastointi projektin sidosryhmien väliseen tietopankkiin sekä tiedon pitäminen ajan tasaisena
Odottaminen	Asiakas ei välttämättä osaa vastata kysymyksiin nopeasti, jolloin asiakasta tulee auttaa tekemään parempia päätöksiä omien tarpeidensa mukaisesti	Jotkin tiedot vaativat aiemmassa vaiheessa tuotettua tietoa, jotta voidaan jatkaa
Liike	Asiakkaan antama toiminnallinen tieto (tilat, ominaisuudet) eivät välity suunnitteluun. Asiakas saa puutteellisen tuotteen	Vaikea tietää milloin mikäkin tieto on valmista ja sitä voidaan hyödyntää muun tiedon luonnissa

Taulukko 2: LEAN-ajattelun mukainen hukkaa aiheuttava tekijä verrattuna ohjelmointi- ja suunnitteluvaiheisiin

sessissa oleva hukka moninkertaistuu iteraatioiden määrän kasvaessa. Poistamalla pieniä hukkia prosessista voidaan siis saavuttaa suuria tehokkuuden parannuksia.

5.2.1 Haahtelan ohjausmalli ja hukka

Ohjelmointi- ja suunnitteluvaiheiden prosesseista löytyy teorian ja haastatteluiden perusteella jonkin verran mahdollista hukkaa. Perinteisiä erilaisia hukkia on kuutta erilaista. Hukat ohjelmointi- ja suunnitteluvaiheissa on esitetty taulukossa 2.

Taulukosta 2 voidaan huomata, että ohjelmointivaiheen hukat liittyvät lähinnä asiakkaan ja projektijohdon välisen tietovirran hallintaan. Projektinjohdon tärkein tehtävä hukan eliminoimiseksi on siis huolehtia, että asiakkaan tarpeet ja ongelmat osataan selvittää ja kuvata nämä tarpeet sopiviksi toiminnoiksi. Tämän lisäksi on tärkeää huolehtia siitä, että asiakas ymmärtää heille tarjotun ratkaisun ja ei esimerkiksi odota jotain aivan muuta kuin on tarkoitus. Työskentelykäytäntöjen sopiminen etukäteen sekä ennalta sovitut kommunikointikanavat auttavat asiakkaan ja projektinjohdon välisen tietovirran hallinnassa. Yleisissä tietomallivaatimuksissa esitetään, että ohjelmointivaiheen alussa projektinjohto ja asiakas sopivat keskenään käytettävistä ohjelmistoista ja formaateista, tiedon jakamisesta sekä tietomallien jakamisesta ja luovuttamisesta asiakkaalle. Asiakkaan ja projektinjohdon välisessä kommunikoinnissa ei Haahtelan ohjausmallissa liene kovinkaan paljon parannettavaa, sillä ohjelmointivaiheen tavoitteet saadaan käytössä olevassa mallissa todella hyvin täytettyä ja tilaluettelo muodostettua. Sen sijaan TAKU[®]:n tuottaman tiedon siirtyminen suunnittelijoille vähentäisi hukkaa selvästi.

Suunnitteluvaiheen prosesseissa toimijoiden määrä lisääntyy, sillä rakennusprojekteissa on useita suunnittelijoita. Suunnittelunimikkeitä ovat esimerkiksi arkkitehti, rakennesuunnittelija, ilmanvaihtosuunnittelija, putkisuunnittelija, sähkösuunnittelija, akustinen suunnittelija, maisemasuunnittelija, paloturvallisuussuunnittelija, turvajärjestelmäsuunnittelija, geosuunnittelija jne. Kaikki toimijat eivät kuitenkaan ole mukana koko projektin ajan. Alun ehdotussuunnitteluvaiheessa, suunnitteluvastuu on tavanomaisesti arkkitehdillä. Ehdotussuunnitelmien hyväksymisen jälkeen hankkeeseen sitoutuu uusia suunnittelijoita ja suunnittelun edetessä joidenkin suunnittelijoiden työmäärä vähenee. Taulukossa 2 esitetyt suunnitteluvaiheen hukat liittyvät TAKU[®]:n ja tietomallinnuksen välisten tietovirtojen hallintaan, sekä saatavilla olevan tiedon ajantasaisuuteen ja päivityksistä kommunikointiin. Tämän lisäksi on tärkeää myös arvioida mitä tietoja jonkin uuden tiedon tuottamien vaatii ja suunnitella suunnitelmien tuotanto siten, että tällaiset toisistaan riippuvat tiedot tuotetaan järkevässä järjestyksessä.

Tietomallinnuksen ja Haahtela TAKU[®]:n välisten tietovirtojen hallinnointi edellyttää tietojen lukemista tietomallista TAKU[®]:un. Tällä hetkellä suunnitelmien kustannusarvioita tehdään siten, että tietomallista mitataan kaikkien rakennuskomponenttien määrät ja syötetään manuaalisesti TAKU[®]:un. Prosessi on siis tällä hetkellä kohtalaisen hidas ja samalla se mahdollistaa inhimillisten virheiden tekemisen tietojen syöttämisessä. Laskelmat joudutaan siis tarkistamaan useaan kertaan, jotta oikeellisuus voidaan varmistaa.

Ylituotantoon liittyvää hukkaa ei havaittu ohjelmointi- ja suunnitteluvaiheiden prosesseissa.

5.2.2 Hukan hallitseminen

Ohjelmointivaiheen hukat ovat jo tällä hetkellä tiedossa ja ne myös osataan torjua Haahtelan ohjausprosessissa hyvin. Ohjelmointivaiheessa hyödynnettävät Axiomatic design ja C-K teorioiden kaltaiset prosessit asiakkaan tarpeiden selvittämisessä ovat riittävän hyviä keinoja selvittää asiakkaan todelliset ongelmat ja tarpeet. Tä-

män jälkeen tarpeiden mukaiset toiminnot voidaan muuntaa tilaluetteloksi. Prosessi on tällä hetkellä toimiva ja syytä suurille muutoksille ei ole. Mikäli tietomallien automaattista lukemista TAKU[®]:un halutaan edistää, ohjelmointivaiheessa on kuitenkin syytä kiinnittää huomiota siihen, että tiedon tarkkuustasot määritellään sellaisiksi, että suunnitteluvaiheen tuottamat tietomalleja voidaan helposti vertailla ohjelmointivaiheen tilaluetteloon. Tiedon tarkkuustaso vaihtelee projektien välillä ja tarkkuustaso täytyy määritellä projektikohtaisesti. Suunnitteluvaiheen havaittujen hukkien hallitseminen ja poistaminen ei ole aivan yksinkertaista. Tässä vaiheessa projektia hallittavan tiedon sekä projektin toimijoiden määrä kasvaa olennaisesti ohjelmointivaiheeseen verrattuna eli muuttujien määrä projektissa kasvaa. Prosessin iteratiivinen luonne aiheuttaa tarpeen prosessin hallinnalle. Yleiset tietomallivaatimukset suosittelvat käytettäväksi sellaista prosessia, jossa suunnittelijat tallentavat suunnittelemansa suunnitelmat säännöllisin välein tietopankkiin. Tärkeää on kiinnittää huomiota siihen, että kaikki toimijat tietävät, milloin heidän suunnitelmiansa pitää olla tietopankissa muiden saatavilla ja missä tarkkuustasossa tiedon on oltava, jotta sitä voidaan hyödyntää jonkin toisen tiedon luomisessa.

Yksi kätevä tapa hallinnoida tietomalleja voisi olla jokin ohjelmistokehityksessä käytössä oleva versionhallintatyökalu, kuten GIT tai SVN. Tämä mahdollistaa kaikkien suunnittelijoiden helpon pääsyn kaikkiin tietomaaleihin, antaa työkalut tehtyjen muutosten tarkasteluun sekä on nopea ja helppo tapa hallita ja ylläpitää tietomalleja. Versiohallinnasta näkee helposti mm. kuka suunnittelijoista on tuottanut mitään suunnitelmia ja millaisia muutokset ovat. Versionhallinnan käyttäminen siis kasvattaisi tiedon läpinäkyvyyttä, mikä puolestaan on yksi keino vähentää prosessin hukkaa. Mikäli jossain suunnitelmassa havaitaan virheitä, versionhallinnan avulla suunnitelmasta voitaisi helposti käyttää aiempaa versiota, jossa virhettä ei ole. Versiohallinnan käyttöönottoaminen ei kuitenkaan ole yksiselitteistä, sillä se vaatii esimerkiksi komentojen ja versionhallinnan perustoimintojen opettelemista.

Scrum on ICT-alalla, varsinkin ketterässä ohjelmistokehityksessä laajasti käytössä oleva toimintatapa, joka pyrkii kehityksen suunnitteluun, tarkasteluun ja hallintaan. Scrumin peruserätyyteenä on käyttää lyhyitä tasamittaisia kehitysjaksoja ja iteratiivisesti tuottaa haluttu lopputuote. Scrum siis pakottaa pilkkomaan projektin riittävän pieniin osiin, jotta osat voidaan tuottaa erikseen lyhyessä ajassa, sprintissä. Sprinttejä toteutetaan niin pitkään, että tuote on valmis. Talonrakentamisen suunnitteluvaiheen prosessi ja yleisten tietomallivaatimusten yhdistäminen muistuttaa monin paikoin scrumia. Jonkin asteista scrumin hyödyntämistä voisi pyrkiä tuomaan siis myös talonrakentamisen suunnittelunohjaukseen.

Yleisten tietomallivaatimusten mukaan tietomallit tulisi tallentaa säännöllisesti projektipankkiin, johon kaikilla tiedoa tarvitsevilla osapuolilla on pääsy. Ajantasaisuuden varmistamiseksi yleiset tietomallivaatimukset esittävät, että on järkevää käyttää ennalta sovittua aikataulua. Suunnitteluvaiheen alussa iteraatiot ovat hieinan pidempiä kuin lopussa. Yleiset tietomallivaatimukset esittävät, että ehdotus- ja yleissuunnitteluvaiheissa iteraation pituus olisi suunnittelukokousten välinen aika ja puolestaan toteutusvaiheessa sopiva iteraatio olisi yhden viikon mittainen. Iteraatioiden kesto siis vaihtelee sen mukaan, miten nopeasti uutta tietoa voidaan tuottaa. Alussa tietoa tuotetaan hitaammin ja lopussa nopeammin. Tärkeää on

pohtia mitkä suunnittelun vaiheet olisivat järkevimpiä suorittaa ensin, jotta muut suunnitelmat voidaan aloittaa oikeaan aikaan. Tällöin on tärkeää sopia ennalta, milloin mitäkin tieto on valmiina ja mikä tiedon tarkkuustason tulee olla. Projektinjohto on vastuussa suunnitteluvaiheen iteraatioiden sisällöstä ja aikatauluista. Eräs havaittu formaattiongelma yleisten tietomallivaatimusten ja TAKU[®]:n välillä on tapa, miten rakennukset yleensä mallinnetaan. TAKU[®]:n kannalta tällä hetkellä paras tapa luoda tietomallit olisi luoda kaikista pääjärjestelmistä omat erilliset koko kiinteistön kattavat mallit ja toisaalta yleisten tietomallivaatimusten suositus olisi puolestaan suunnitella pääjärjestelmät omina malleina kerroksittain. Ratkaisu tähän ongelmaan on joko yrittää luoda tapa tuoda tiedot tietomallista TAKU[®]:n kerroksittaisista malleista tai suunnitella pääjärjestelmät TAKU[®]:n edellyttämällä tavalla.

Toinen merkittävä ongelma on IFC:n ja TAKU[®]:n välille luotavan rajapinnan luominen ja tämän jälkeen sen ylläpitäminen. Rajapinnan luominen itsessään on aluksi melko suuri työ. Toisaalta tällainen kohdentaminen joudutaan jo nyt väkisin tekemään projektikohtaisesti, vaikka se tehdäänkin tällä hetkellä manuaalisesti. Pohjana voisi käyttää esimerkiksi toteutettujen projektien IFC-malleja ja saada näin tiedot tietyistä peruskomponenteista rajapintaan. Tällä tavalla siis selviää, minkälaista tietoa IFC:stä tyyppillisesti siirretään TAKU[®]:un. Rajapinnan ylläpitäminen on välttämätöntä kahdesta syystä:

1. Rakennuksissa käytettävät rakennusosat ja materiaalit vaihtelevat projektista riippuen. Esimerkiksi väliseinä voidaan toteuttaa kymmenillä eri tavoilla.
2. Automatisoinnin tuottamat hyödyt katoavat, mikäli tiedonsiirto TAKU[®]:n ei toimi tai rajapinnan tiedot ovat vanhentuneita.

Ensimmäistä kohtaa ei ole mahdollista välttää prosessissa. Rajapintaa täytyisi melko varmasti muokata projektikohtaisesti tai vähintään siihen täytyisi lisätä projektin vaatimia uusia komponentteja. Toinen kohta puolestaan voi olla vältettävissä, esimerkiksi mikäli IFC:stä kyetään muodostamaan TAKU[®]:lle sopivia JSON paketteja. Tällöin tietoa voisi hakea TAKU[®]:sta ja tehdä kustannusarviolaskelmia samaan tapaan kuin mallinnusta tehdään tällä hetkellä manuaalisesti. Tietomallit ovat kooltaan suuria, mikä johtaa siihen, että lukeminen TAKU[®]:un saattaa kestää jonkin aikaa. Tämä saattaa tuoda myös tietynlaisia haasteita rajapinnan vikasietoisuuteen. Rajapintaa luotaessa täytyy ottaa siis huomioon skaalautuvuus käytettäessä isompia tietomalleja.

Suunnitteluvaiheen prosessissa voitaisi saavuttaa merkittäviä tehokkuusparannuksia, mikäli tietomallista voisi lukea tietoja TAKU[®]:n ilman, että tietoja täytyy manuaalisesti lukea tietomallista, syöttää TAKU[®]:un ja laskea kustannusravio. Manuaalisen työn poistuessa inhimillisten virheiden mahdollisuus pienenee olennaisesti. Automaattisessa tiedonsiirrossa on tiettyjä ennalta mainittuja haasteita, mutta pitkällä aikavälillä sen tuottamat hyödyt vaikuttaisivat kannattavilta. Tehokkuuden parantamisen lisäksi automaattinen tiedon siirto voi mahdollistaa mallin valmiusasteen arvioinnin. Esimerkiksi väliseinän määrän voi laskea mallista ja verrata mallissa olevan väliseinän kustannusta arvioon väliseinän kokonaismäärän kustannukseen.

Ongelma	Ratkaisu
Informaation liikkuminen tietovirroissa asiakkaan ja projektijohdon välillä	Tämä prosessin osa on jo hyvin optimoitu ja asiakkaan tarpeet osataan tehokkaasti muuntaa tilaluetteloksi
Asiakas ei välttämättä osaa vastata kysymyksiin nopeasti, jolloin asiakasta tulee auttaa tekemään parempia päätöksiä omien tarpeidensa mukaisesti	Projektinjohto fasilitoi asiakkaan päätöksentekoa ja kertoo vaihtoehtoista ja niiden ominaisuuksista
Asiakkaan ohjaus epäonnistuu. Asiakas huomaa myöhemmin suunnitteluvaiheessa tarvitsevänsä tiloja joihin asiakkaalla ei ole varaa.	Asiakkaan ohjaaminen tekemään toimintojensa kannalta parempia päätöksiä. Asiakkaan odotusten pitäminen realistisina.
Asiakkaan antama toiminnallinen tieto (tilat, ominaisuudet) eivät välity suunnitteluun. Asiakas saa puutteellisen tuotteen	Projektinjohto on vastuussa siitä, että rakennuksen tilat vastaavat sitä, mitä asiakkaan kanssa on sovittu ohjelmointivaiheen päätteeksi

Taulukko 3: Ohjelmointivaiheen havaitut ongelmakohdat ja niiden ratkaisut

Tästä voisi olla mahdollista päätellä, milloin mikäkin tietovirta voisi olla valmista luettavaksi TAKU[®]:un.

Taulukossa 3 on koostettu teorian ja haastattelujen avulla havaitut ongelmakohdat ohjelmointivaiheessa ja taulukossa 4 on koostettu puolestaan suunnitteluvaiheen ongelmat. Taulukoista löytyy myös ratkaisuehdotus kuhunkin löydettyyn ongelmaan. Ohjelmointivaiheen ongelmat ovat jo olleet hyvin tiedossa, joten merkittävimmät prosessien parannukset voidaan saada suunnitteluvaiheen prosesseissa.

Vaikka yhteyden luominen IFC-tietoformaatin ja TAKU[®]:n välille vaatii aluksi paljon työtä, iteratiivisen prosessin kautta saatava kumulatiivinen hyöty tehokkuuden parantumisesta vaikuttaisi pitkällä tähtäimellä kannattavalta. Alun työn jälkeen rajapinta IFC:n ja TAKU[®]:n välillä vaatii myös jonkin verran ylläpitotyötä. Ylläpityön määrää on tällä hetkellä vaikeaa arvioida, mutta verrattuna nykyään tehtävään manuaaliseen tietojen kopiointiin, vaikutukset ovat melko varmasti pienemmät.

Tämän tutkimuksen tutkimuskysymykset olivat:

1. Miten rakennushankkeen ohjausta voidaan tehostaa?
2. Mitä tietoa tarvitaan suunnittelunohjausprosessin eri vaiheissa?
3. Miten voidaan varmistaa tiedon laatu, oikeellisuus ja oikea-aikaisuus kussakin vaiheessa?
4. Millaisia formaatteja on käytössä talonrakentamiseen liittyvän tiedon tallentamiseen ja siirtoon?

Ensimmäistä tutkimuskysymystä lähestyttiin kompleksisten prosessien ohjaamisen teorian näkökulmasta siten, että ohjelmointi- ja suunnitteluvaiheiden prosesseista

Ongelma	Ratkaisu
BIM- suunnitelmista luetaan tietoa tietämättä, että se ei ole valmista	Tietovirran oikea-aikaisuuden hallinta tai yleisten tietomallivaatimusten mukainen tietomallien päivittämissykli. Versionhallinnalla voidaan kasvattaa läpinäkyvyyttä
Informaation liikkuminen tietovirroissa BIM-domainin ja TAKU [®] -domainin välillä. Tietoja siirretään tietomallista TAKU [®] :un manuaalisesti	IFC:n lukeminen automaattisesti TAKU [®] :un nopeuttaa tiedonsiirtoa sekä vähentää inhimillisten virheiden määrää tiedon kopioinnissa
Tietomallien ja muun tuotetun tiedon varastointi projektin sidosryhmien väliseen tietopankkiin	Tietomallien säännöllinen päivittämisestä sopiminen tai tietomallien siirtäminen versionhallintaan
Jotkin tiedot vaativat aiemmassa vaiheessa tuotettua tietoa, jotta voidaan jatkaa	Iteraatioiden suunnitteleminen etukäteen. Tietomallien säännöllinen päivittäminen ja etukäteen sopiminen. Tärkeässä roolissa on tietovirran hallinta BIM-domainin ja TAKU [®] -domainin välillä
Vaikea tietää milloin mikäkin tieto on valmista ja sitä voidaan hyödyntää muun tiedon luonnissa	BIM-domainin ja TAKU [®] -domainin välisen tietovirran hallinta ja automaattinen tiedon siirtäminen ja siitä tiedon valmiusasteen arvioiminen

Taulukko 4: Suunnitteluvaiheen havaitut ongelmakohdat ja niiden ratkaisut

pyrittiin etsimään LEAN-ajattelun mukaista hukkaa. Systemien kompleksisuus aiheuttaa hukkaa, joten tutkimalla prosessien ongelmallisimpia kohtia, hukkaa voidaan poistaa. Tutkimuksessa havaittiin suunnitteluvaiheen prosessin sisältävän kohtalaisen paljon manuaalista työtä, jota voisi olla mahdollista vähentää ottamalla käyttöön automaattinen tiedonsiirto BIM-domainin ja TAKU[®]-domainin välille. Tiedonsiirron tehostaminen liittyy olennaisesti myös BIM-domainin ja TAKU[®]-domainin välisten tietovirtojen hallintaan. Tietovirtojen hallitsemiseksi voidaan käyttää apuna yleisten tietomallivaatimusten asettamia minimivaatimuksia kussakin suunnitteluvaiheen vaiheessa.

Toiseen tutkimuskysymykseen ei löytynyt suoraa vastausta, sillä tarvittava tieto riippuu aina kustakin vaiheesta ja tiedot ovat projektista riippuen käytettävissä eri aikaan: kaikki suunnittelijat eivät suunnittele komponentteja samassa järjestyksessä. Tutkimus lähestyy ongelmaa tutkimalla tietovirtaa BIM-domainin ja TAKU[®]-domainin välillä. Tietovirtaa voidaan hallinnoida sopimalla etukäteen tuotettavien suunnitelmien järjestys sekä tarkkuustaso. Sopimisen lisäksi tiedonsaannin läpinäkyvyyttä voidaan lisätä tiedonkulun helpottamiseksi. Myös erilaisten tiedon jakamista tukevien toimintamallien implementoiminen auttaa tietovirran hallinnassa.

Kolmas kysymys koostuu kahdesta osasta: miten tiedon laatu ja oikeellisuus voidaan varmistaa ja miten varmistetaan, että tieto on käytettävissä oikeaan aikaan. Tätäkin kysymystä tutkimus lähestyy tietovirtojen ja niiden hallitsemisen kautta. Tällä hetkellä suunnitteluvaiheen tietovirrassa BIM-domainin ja TAKU[®]-domainin välillä työ tehdään manuaalisesti. Tiedon oikea-aikaisuudesta huolehditaan kommunikoimalla suunnittelijoiden kanssa ja sopimalla siitä, milloin mikäkin tieto on valmista luettavaksi BIM-domainista TAKU[®]-domainiin. Yleisten tietomallivaatimusten mukaisesti projektin osapuolet sopivat projektin alussa kussakin vaiheessa käytettävät ohjelmistot, formaatit, tiedon tarkkuustasot sekä kommunikointikanavat. Tuomalla yleisten tietomallivaatimusten vaatimat asiat mukaan suunnittelunohjausprosessiin, voidaan entistä paremmin varmistaa tiedon laatu ja oikea-aikaisuus suunnitteluvaiheen edetessä. Tämän lisäksi prosessia voisi pyrkiä selkiyttämään tuomalla mukaan ohjelmistokehityksestä tuttuja scrum prosessin piirteitä sekä lisätä läpinäkyvyyttä projektipankin versionhallinnalla.

Neljäs tutkimuskysymys liittyy BIM-domainin ja TAKU[®]-domainin välisen tietovirran automatisointiin. Tutkimuksessa käsiteltiin tällä hetkellä laajassa käytössä olevia tietomalliformaatteja. Formaateja on käytännössä kahta eri tyyppiä: patentoidut ja käytölle avoimet formaatit. Patentoidut formaatit eivät sovellu hyvin Haahtelan käyttöön, sillä se vaatisi käyttöoikeutta. Kyseeseen tulee siis käyttää avointa formaattia. IFC eli Industrial Foundation Classes on IAI-järjestön kehittämä ja ylläpitämä tietomallinnuksen standardi ISO/PAS 16739. IFC:n hyvänä puolena on myös se, että mm. suunnittelijoiden käyttämä AutoCAD ohjelmisto pystyy jo nyt tuottamaan IFC-formaattia. IFC:n ja TAKU[®]:n välille täytyy kuitenkin tätä varten rakentaa yhteys joka tuo omat haasteensa rajapinnan luomisessa.

Parannettavia kohteita löydettiin jonkin verran prosesseista, kuten manuaalisen työn vähentäminen sekä tietovirtojen hallinnoimiseen liittyviä ongelmia. Tutkimus on kuitenkin melko pintapuolinen ja se ei sisällä konkreettisia esimerkkejä siitä, miten IFC-formaatin tietoja voidaan siirtää TAKU[®]:un. Löydetyt ongelmat ovat

kuitenkin haastattelujen perusteella Haahtela-yhtiöiden asiantuntijoiden mukaan todellisia ongelmakohtia ja periaatetasolla näihin ongelmakohtiin voidaan puuttua. Tämän tutkimuksen jälkeen olisi syytä toteuttaa yksinkertainen tiedonsiirto IFC-formaatista TAKU[®]:un esimerkiksi käyttämällä hyvin yksinkertaista IFC-mallia. Tämän jälkeen rajapinnan rakentamisen voisi aloittaa esimerkiksi hyödyntämällä aiemmin toteutettujen projektien tietomalleja.

Tutkimus keskittyy voimakkaasti Haahtelassa käytössä olevaan projektinohjausmalliin ja olisi mielenkiintoista vertailla eri yrityksillä käytössä olevia menetelmiä varsinkin suunnitteluvaiheen prosesseissa. Rakennusalan yritysten prosessit eivät kuitenkaan ole yleisesti tiedossa ja tämän tutkiminen vaatisi laajemman aineiston keräämistä rakennusprosessin ohjelmointi- ja suunnitteluvaiheista. Yleiset tietomallivaatimukset kuitenkin toiminevat toimintatapoja yhdistävänä tekijänä, mikäli kaikki pyrkivät yleisten tietomallivaatimusten osoittamiin vaatimuksiin. Tutkimuksen tavoitteena ei kuitenkaan ollut kartoittaa rakennusalan toimintamalleja yleisellä tasolla, vaan etsiä tehokkuutta parantavia toimia Haahtelan omista prosesseista.

6 Yhteenveto

Haahtela-yhtiöissä tutkitaan aktiivisesti käytössä olevaa talonrakentamisen projektinohjausprosessia. Tutkimusta lähestytään kompleksisten systeemien tai järjestelmien teorian ja niiden ymmärtämisen sekä hallitsemisen näkökulmasta. Kirjallisuusosion alussa esitellään lyhyesti, miten talonrakentamisen projektinohjausprosessi etenee kolmessa eri vaiheessa ja miten rakennus saadaan rakennetuksi asiakkaalta tulevan tilauksen pohjalta valmiiksi. Projektinohjausta tapahtuu talonrakentamisen ohjelmointi- ja suunnitteluvaiheissa, joten nämä kaksi vaihetta ovat eniten tarkastelussa. Kirjallisuusosiossa esitellään kaksi erilaista kompleksisuuden tyyppiä: deduktiivinen ja induktiivinen kompleksisuus. On tärkeää tietää, millaista kompleksisuutta järjestelmässä tai systeemissä esiintyy, mikäli kompleksisuutta halutaan hallita. Haahtelan projektinohjausmallissa esiintyy paljon induktiivista kompleksisuutta, joka aiheuttaa jonkin verran hukkaa ohjelmointi- ja suunnitteluvaiheissa. Ongelmat ovat luonteeltaan Rittel et al. (1972)[5] määrittelemiä vaikeita induktiivisia ongelmia. Projektinohjausprosessin vaiheet sisältävät paljon muuttujia ja uutta tietoa luodaan prosessien aikana, jota käytetään edelleen uuden tiedon luonnissa. Tämä vaikeuttaa mm. suunnittelua pitkälle tulevaisuuteen, sillä ei ole varmaa, milloin jokin tieto on varmasti olemassa.

Mitä monimutkaisempi systeemi on, sitä enemmän systeemissä on mahdollisia hukkaa aiheuttavia tekijöitä. Projektinohjausprosessi jaotellaan kolmeen vaiheeseen, jotka ovat ohjelmointivaihe, suunnitteluvaihe sekä rakentaminen työmaalla ja vaiheet toteutetaan järjestyksessä siten, että seuraava vaihe alkaa, kun edellinen päättyy. Vaiheet eivät siis ole käynnissä samaan aikaan ja edellisen vaiheen tuloksia hyödynnetään seuraavan vaiheen lähtötietoina. Haahtela-yhtiöiden projektinohjausprosessin ohjelmointi- ja suunnitteluvaiheista etsitään LEAN-ajattelun mukaisesti hukkaa ja pyritään tunnistamaan prosesseista hukkaa aiheuttavia kohtia. Kompleksisuutta voidaan pyrkiä vähentämään systeemistä, jolloin systeemin ennustettavuus parantuu ja mahdollisia paikkoja hukan syntymiseksi on vähemmän. Haahtela-yhtiöiden projektinohjausprosessissa ohjelmointivaiheen kompleksisuutta on pyritty hallitsemaan käyttämällä mm. axiomatic design ja C-K teorian kaltaisia menetelmiä. Suunnitteluvaiheessa puolestaan tiedon määrä sekä tarkkuus kasvavat ja toimijoiden määrä projektissa lisääntyy. Tämä johtaa tarpeeseen hallita tietoa järjestelmällisesti ja huolehtia siitä, että oikea tieto on tarjolla oikealla henkilöllä oikeaan aikaan. Tällaisesta tiedon hallitsemisesta käytetään termiä tietovirta. Haahtelan ohjausmallin prosesseissa on paljon erilaisia tietovirtoja, eli kommunikaatiota ja informaation jakamista asiakkaan ja projektinjohdon sekä suunnittelijoiden ja projektinjohdon välillä. Tietovirtojen tärkeimmät ominaisuudet ovat tiedon oikea formaatti ja tiedon oikea-aikaisuus. On siis tärkeää, että projektin osapuolet sopivat projektin alussa käytettävät formaatit, kommunikaatiokanavat, ohjelmistot, tiedon tarkkuustasot sekä aikataulut. Sopimisen perustana voidaan käyttää esimerkiksi yleisten tietomallivaatimusten kuhunkin vaiheeseen määrittelemiä minimivaatimuksia.

Tutkimusvaiheessa Haahtelan tällä hetkellä käyttämät prosessit projektinohjausprosessin ohjelmointi- ja suunnitteluvaiheissa selvitettiin tutkimalla Haahtela-yhtiöiden aikaisempaa tutkimusta aiheesta sekä keskustelemalla asiantuntijoiden

kanssa Haahtelan organisaatiossa. Tämän jälkeen selvitettiin, miten alalla yleisesti toimitaan kussakin vaiheessa. Yleisten tietomallivaatimusten sekä Haahtelassa käytettyjen projektinohjausprosessin vaiheista löytyy paljon samaa, mutta havaittiin myös muutamia ongelmia etenkin suunnitteluvaiheen prosesseissa. Mikäli tietomalliajattelua ja Haahtela TAKU[®]:n tietoja halutaan käyttää rinnakkain siten, että molemmat ajattelutavat tukevat toisiaan, on otettava huomioon tällä hetkellä olemassa olevat erot tavassa muodostaa rakennuksen tietomalli. Tärkeimmät suunnitteluvaiheessa havaitut ongelmat Haahtelan projektinohjausprosessin ja tietomallienyhdistämisessä ovat:

1. BIM-tietomallin tiedot on ensisijaisesti luotu järjestelmien mallintamista varten ja TAKU[®]:n tietoja puolestaan käytetään käyttäjän tahtotilan kuvaukseen sekä kustannusten määrittämiseen. Yleiset tietomallivaatimukset suosittelivat tietomalleissa käytettävän IFC:n organisoimista siten, että suunnitelmat pääjärjestelmistä luodaan omat kerroksittaiset mallit. TAKU[®] puolestaan käsittelee tietoa kokonaisuuksina ja parempi tapa olisi luoda pääjärjestelmistä koko kiinteistön kattavat mallit.
2. Tiedon vieminen TAKU[®]:un ja sen pohjalta laskelmien tekeminen vie tällä hetkellä paljon aikaa, joten on tärkeää, ettei jonkin tilakokonaisuuden tai järjestelmän tietoja käytetä liian aikaisin. Laskelmat kannattaa laskea, vasta siinä vaiheessa, kun suunnitelma on riittävän valmis.

Tietovirtojen hallitsemiseksi suunnitteluvaiheessa löydettiin mahdollinen keino tehokkuuden parantamiseksi tiedonsiirrossa BIM-domainin ja TAKU[®]-domainin välillä: mikäli tietoa voidaan siirtää automaattisesti IFC-formaatista TAKU[®]:uun, manuaalisen työn määrää voidaan vähentää prosessissa. Prosessin ollessa iteratiivinen, pienillä parannuksilla on kumulatiivisen vaikutuksensa vuoksi suuri merkitys prosessin kokonaistehokkuuden parantamisessa. Tietovirtojen hallitsemiseksi esitetään prosessia, jossa suunnitelmia tehdään pienempinä kokonaisuuksina ennalta määrättyllä aikavälillä. Tällöin voidaan varmistaa oikean tiedon oikea-aikainen päätyminen niille henkilöille, jotka tietoa tarvitsevat, sillä tarvittava tieto voidaan suunnitella tuotettavaksi oikeaan aikaan. Yleiset tietomallivaatimukset esittävät, että suunnitteluvaiheen alussa iteraation aikaväli on pidempi kuin suunnittelun lopussa: alussa riittää suunnittelukokouksen väli ja myöhemmin iteraatiot kestäisivät noin viikon.

Tiedon automaattinen lukeminen IFC-formaatista ja sen syöttäminen TAKU[®]:un ei ole aivan yksinkertaista, sillä se vaatii IFC:n tiedon tulkintaa Haahtela nimikkeistöön nähden. Rajapinnan luomisen jälkeen edessä on rajapinnan ylläpitämistyötä, jonka lisäksi rajapintaa pitäisi todennäköisesti päivittää projektikohtaisesti. Ajan mittaan ylläpitotyön määrä todennäköisesti kuitenkin vähenee, kun rajapinnasta löytyy jo suurin osa yleisimmin käytettävistä rakennuskomponenteista.

Ohjelmointivaiheessa asiakkaan tarpeiden määrittämisessä onnistutaan jo tällä hetkellä todella hyvin ja asiakkaan tarpeiden määrittämisessä toiminnoiksi ja edelleen tiloiksi ei ole mahdollista saavuttaa merkittäviä tehokkuusparannuksia. Sen sijaan asiakkaan tarpeiden siirtoa TAKU[®] ohjelmistosta suunnittelijoiden ohjelmistoihin

(esimerkiksi BIM) tulisi tehostaa. TAKU[®] käyttää lähtötietona tiloja ja tilojen laadullisia ominaisuuksia ja mallintaa lopputietona rakennuksen virtuaalisen mallin sekä investointi- ja ylläpitokustannukset, joilla rakennus tulee toteuttaa. Kustannus ja rakennuksen toiminnallisuus on siis synkronoitu yhdenmukaiseksi. Tämän tiedon siirto suunnittelijoiden käyttämiin ohjelmistoihin ei ole digitaalista ja vaarana on asiakkaan vaatimuksia sisältävän tiedon katoaminen suunnittelun aikana. Tämä taas johtaisi rakennustuotteen laadun menettämiseen tai mahdollisesti tavoitekustannuksen ylitykseen. Jo tällä hetkellä ohjelmointivaiheen tilaluettelo on yleisten tietomallivaatimusten mukaisesti taulukkomuodossa oleva tilaohjelma. Tiedon digitaaliseen siirtoon voidaan kokeilla COBie:n käyttöä. COBie:n käyttäminen saattaisi helpottaa suunnittelijan työtä, mikäli COBie:n käyttö yleistyy alalla, sillä COBie:sta on helppoa muodostaa siihen perustuvaa IFC:tä. Mikäli COBie ei ole suunnittelijalle ennalta tuttu, sen käyttöönottaminen tuskin on kannattavaa, sillä nykyiset menetelmät toimivat jo nyt riittävän hyvin asiakkaan tarpeiden määrittämiseksi.

Viitteet

- [1] Pennanen Ari: "Workplace Planning", 2004, s.23-24
- [2] BuildingSMART, COBIM-hanke: Yleiset tietomallivaatimukset, Osa 1: Yleinen osuus, s. 5-8, 2012
- [3] Ballard Glenn, Pennanen Ari, Haahtela Yrjänä, "Target costing and designing to targets in construction". Journal of financial management of property and construction 2011. Emerald group publishing limited.
- [4] Niukkanen, I. (1980), "Quality and Cost Factors in Architectural Design" (Rakennussuunnittelun sisällön ohjaustekijät), Department of Architecture, Helsinki University of Technology, Helsinki
- [5] Rittel H. & M. Webber: "Dilemmas in general theory of planning", 1972, Working paper no. 194. University of California at Berkeley.
- [6] Ballard Glenn, Pennanen Ari: "Conceptual estimating and target costing", IGLC Brasil 2013
- [7] Sterman JD: "Business dynamics: systems thinking and modelling for a complex world", 2000
- [8] Sterman JD: "Learning in and about complex systems", 1994
- [9] Suh N.P., "The principles of design", New York, Oxford University Press, 1990.
- [10] Nicolis, John S.: "Chaos and information processing", 1998, s.15
- [11] Lauri Koskela, Ari Pennanen: "Necessary and unnecessary complexity in construction" 2005
- [12] Nam p. Suh "Complexity - theory and applications", 2005
- [13] Ballard G, Howell G: "Toward construction JIT", 1995, Berkeley University
- [14] Koskela L: "Making-do - the eight category of waste", 2004
- [15] Ballard G, Koskela L, Tommelein Iris: "The foundations of lean construction", 2002
- [16] Ohno T: "Toyota production system", 1988
- [17] Koskela L: "An exploration towards a production theory and its application to construction" VTT Publications 408, Espoo, Finland, 2000
- [18] Peter Kueng: "Process performance measurement system: a tool to support process-based organizations", 2000
- [19] Principia Cybernetica: "The nature of cybernetic systems", 1992

- [20] Ari Pennanen: “Workplace Planning”, 2004, s.48-53
- [21] Hatchuel A, Le Masson P, Weil B, “C-K Theory in practice: Lessons from industrial applications”. International design conference, 2004, Dubrovnik
- [22] Salman Azhar, Michael Hein, Blake Sketo: “Building Information Modeling (BIM): Benefits, Risks and Challenges”, 2007
- [23] Timo Hartmann, Hendrik van Meerveld, Niels Vossebeld, Arjen Adriaanse: “Aligning building information model tools and construction management methods” 2011
- [24] BuildingSMART, COBIM-hanke: Yleiset tietomallivaatimukset, Osa 1: Yleinen osuus, s. 12-18, 2012
- [25] Tsoukas, H. 2009. “Dialogical Approach to the Creation of New Knowledge in Organizations”, Organization Science, Vol. 20, No. 6, November-December 2009, pp. 941-957.
- [26] BuildingSMARTin ylläpitämä IFC dokumentaatio <http://www.buildingsmart-tech.org/ifc/IFC4/final/html/>, Vierailtu 27.5.2019, Viitattu 1.2.2019
- [27] BuildingSMART, COBIM-hanke: Yleiset tietomallivaatimukset, Osa 4: Talotekninen suunnittelu, s.5-11, 2012