Joona Niemi

Metatason ohjeistus logistisen laatikkojärjestelmän vaatimusmäärittelylle

Tekniikan ja innovaatiojohtami-
sen yksikkö

Tietojärjestelmätieteen pro
gradu -tutkielma

Vaasa 2019

VAASAN YLIOPISTO
Akateeminen yksikkö Tekniikan ja innovaatiojohtamisen yksikkö
Tekijä: Joona Niemi
Tutkielman nimi: Metatason ohjeistus logistisen laatikkojärjestelmän vaatimusmäärittelylle
Tutkinto: Kauppatieteiden maisteri
Oppiaine: Tietojärjestelmätiede
Työn ohjaaja: Petri Helo ja Tero Vartiainen
Valmistumisvuosi: 2020 **Sivumäärä:** 114

TIIVISTELMÄ:
Yritysten on pystyttävä reagoimaan reaaliaikaisiin muutoksiin, esimerkiksi elintarviketeollisuudessa korostuvaan kysynnän nopeisiin muutoksiin. Asiakkaat haluavat lyhyempien toimitusaikojen lisäksi myös joustavuutta ja mukautuvuutta eli ketteryyttä heidän tarpeisiinsa. Ketteryyden mahdollistaa läpinäkyvä ja reaaliaikainen tietovirta, joka helpottaa varautumista ongelmatapauksiin ja toiminnan ohjaamista. Kuitenkin erityisesti elintarviketeollisuudessa ongelmana ovat useat rajapinnat ja niissä tapahtuvien toimenpiteiden läpinäkymättömyys ja suhteellisen suuri työmäärä. Monilla aloilla toimitusaikavaatimukset ovat tiukentuneet. Erityisesti kehitys korostuu elintarviketeollisuudessa. Nopeat toimitusaikavaatimukset yhdistettynä kysynnän nopeisiin muutoksiin ja tuoreusvaatimuksiin aiheuttavat myös uusia vaatimuksia tietojärjestelmille. Toimitusketjuissa ja yritysverkostoissa tarvitaan eri osapuolten välille tiedonkulun kanava, jotta toiminnan tehostaminen ja kommunikointi on mahdollista.

Tutkimuksen tavoitteena oli suunnittelutieteen avulla kehittää artefakti, jonka avulla voidaan luoda mahdollisimman tehokas, tuottava ja ketterä logistinen laatikkojärjestelmä. Tutkimuksessa pyrittiin vastaamaan seuraavaan tutkimuskysymykseen: Millainen on metatason ohjeistus, jonka avulla pystytään luomaan vaatimusmäärittely, jonka pohjalta voidaan toteuttaa mahdollisimman tehokas, tuottava ja ketterä logistinen laatikkojärjestelmä? Tutkimus toteutettiin suunnittelutieteellisenä tutkimuksena, koska siinä pyritään rakentamaan artefakti, jonka avulla ratkaistaan kohdeympäristön ongelma. Tutkimus noudatti suunnittelutieteellisen tutkimuksen DSRM-prosessimallia. Aineistoa artefaktin rakentamista varten kerättiin tieteellisestä kirjallisuudesta ja käytännön aineistosta, jota alfa-yritykselle tehtävää vaatimusmäärittelyprojektia varten kerättiin.

Tutkimustuloksena oli artefakti, metatason ohjeistus, joka toimii laatikkojärjestelmän suunnittelijoiden ohjeistuksena ja tukena ja se auttaa heitä ymmärtämään, mitä laatikkojärjestelmän vaatimusmäärittelyssä tulisi ottaa huomioon ja miten se tulisi toteuttaa. Artefaktin relevanttiutta testattiin demonstroimalla se alfa-yrityksen uuden laatikkojärjestelmän vaatimusmäärittelyssä. Metatason ohjeistus toimi hyvänä ohjeena alfa-yrityksen vaatimusmäärittelyprojektissa, joten tutkimuksessa luotua metatason ohjeistusta voidaan pitää geneerisenä työkaluna, jota kaikki toimialan yritykset voivat hyödyntää toimeksiantajan lisäksi. Artefaktin tehokkuus, tuottavuus ja ketteryys perustuivat lisäarvon tuottamiseen, käytettävyyden huomioimiseen, tiiviiseen yhteistyöhön sidosryhmien kanssa ja selkeään dokumentaatioon. Artefaktin yleistettävyyttä, luotettavuutta ja relevanttiutta voitaisiin kuitenkin parantaa hyödyntämällä sitä myös muiden logististen laatikkojärjestelmien vaatimusmäärittelyprojekteissa. Kaikkein konkreettisin jatkotutkimusaihe olisi tutkia tämän tutkimuksen alfa-yrityksen laatikkojärjestelmää sen käyttöönoton jälkeen. Jatkotutkimusten avulla saataisiin selvyys siitä, tuliko artefaktin perusteella suunnitelluista logistisista laatikkojärjestelmistä oikeasti tuottavia, tehokkaita ja ketteriä.

AVAINSANAT: Ohjelmistotuotanto, vaatimusmäärittely, suunnittelutiede, logistiikka, tietojärjestelmät, artefakti

Sisällys

Kuviot

Taulukot

1 Johdanto

Logistiikka voidaan Haapasen (1993) mukaan kytkeä strategiseen johtamiseen. Logistiikka on tuotantoon, tavaran hankintaan ja jakeluun liittyvä strategisesti johdettu materiaalivirtojen, tietovirtojen ja pääomavirtojen integroitu prosessi, jonka päämääränä on yrityksen tuoton parantaminen oikeasuuntaisilla strategisilla valinnoilla, kehittämällä asiakkaille lisäarvoja ja hyötyjä, lisäämällä kierrätystä sekä parantamalla materiaalitoimintojen kustannustehokkuutta.

Logistiikka voidaan määritellä vielä käytännönläheisemmin. Logistiikka on materiaalivirtojen, tietovirtojen, pääomavirtojen, tuotannon, hankinnan, kierrätyksen ja jakelun, huolto- ja tukipalveluiden, kuljetus-, varastointi- ja muiden lisäarvopalveluiden sekä asiakassuhteiden ja -palvelun kokonaisvaltaista kehittämistä ja johtamista. (Karrus, 2001.)

Logistiikkaa voidaan pitää osana jotain suurempaa, jota nimitetään suomeksi toimitusketjuksi tai tarjontaketjuksi (engl. Supply Chain). The Council of Supply Chain Management Professionalsin (CSCMP) mukaan toimitusketju voidaan määritellä kahdella eri tavalla. Ensiksikin toimitusketju voi olla lopullista valmistetta käyttävään loppukäyttäjään päättyvä ja käsittelemättömistä raaka-aineista alkava useita yrityksiä toisiinsa yhdistävä toimitusketju tai logistisessa prosessissa tietojen ja materiaalien vaihdot ulottuen raaka-aineiden hankinnasta valmiiden tuotteiden toimitukseen loppukäyttäjille. Toiseksi toimitusketjua voidaan pitää kaikkien palveluntarjoajien, toimittajien ja asiakkaiden välisenä linkkinä. (Hokkanen, Karhunen & Luukkainen, 2011, s. 12.)

Toimitusketjun hallinta käsittää CSCMP:n määritelmän mukaan kaikki materiaalien jalostukseen ja hankintaan liittyvien toimien hallinnan ja suunnittelun sekä kaikki toiminnot, jotka liittyvät logistiikkahallintoon. Kanavayhteistyö kumppaneitten kanssa kuuluu siihen olennaisena osana. Kumppanit voivat olla välittäjiä, toimittajia, asiakkaita ja kolmannen osapuolen palveluntarjoajia. Se integroi erityisesti tarjonnan ja kysynnän hallinnan yrityksissä ja yritysten välillä. Toimitusketjun hallinta on integroiva toiminto, jonka ensisijaisena vastuualueena on pääasiallisten tehtäväkokonaisuuksien ja liiketoimintaprosessien

yhdistäminen yritysten ja niiden välillä suorituskykyiseksi ja yhtenäiseksi liiketoimintamalliksi. Kaikki edellä mainitut logistiikkahallinnon ja tuotannon toiminnot kuuluvat siihen. Lisäksi se ohjaa markkinointiin, myyntiin, tuotesuunnitteluun, informaatioteknologiaan ja rahoitukseen liittyviä prosesseja ja niiden välistä yhteistyötä. (Hokkanen, Karhunen & Luukkainen, 2011, s. 12-13.)

Toimitusketjuissa ja yritysverkostoissa tarvitaan eri osapuolten välille tiedonkulun kanava, jotta toiminnan tehostaminen ja kommunikointi on mahdollista (Helo, Tuominen, Sandvik & Tukeva, 2007, s. 15). Yritysten on mukauduttava reaaliaikaiseen aikakäsitykseen, koska reaaliaikaiseksi muuttunut maailma sitä vaatii. Yritysten on pystyttävä reagoimaan suunnitelmien, kuten tilausten lisäksi reaaliaikaisiin muutoksiin, esimerkiksi kysyntään. (Christopher, 1998.) Asiakkaat haluavat lyhyempien toimitusaikojen lisäksi myös joustavuutta ja mukautuvuutta heidän tarpeisiinsa. Tätä voidaan pitää ketteryytenä. Ketteryyden mahdollistaa läpinäkyvä ja reaaliaikainen tietovirta, joka helpottaa varautumista ongelmatapauksiin ja toiminnan ohjaamista. Toimitusketjun eri osapuolten välille on tärkeää saada oikeaan aikaan oleellinen tieto, oikeaan paikkaan, jotta toimitusketjun toimivuuden optimoiminen on mahdollista. (Helo ja muut, 2007, s. 15.)

Informaatiojärjestelmien myötä informaation merkitys on korostunut logistiikan ja tehokkaan toimitusketjun kehityksessä. Uusien logistiikan strategioiden toteuttaminen on mahdollista informaatioteknologian ansiosta. Esimerkiksi ERP-järjestelmät (engl. Enterprise Resource Planning) linkittävät eri yritysten toiminta-alueiden informaatiojärjestelmiä toisiinsa, myynnistä tuotantoon ja hankinnasta jakeluun. Tämän tyyppinen informaation yhdistäminen luo toimitusketjun hallinnasta hyviä kehitysnäkymiä kysyntäketjun hallintaan. Kun tieto kysynnästä siirtyy reaaliajassa tuotantoon suoraan kysynnän mukaisesti, ei yrityksen tuotanto toimi enää ainoastaan kysynnän ennusteiden mukaan. Logistiikan tietojärjestelmät pitäisi nähdä mahdollistajana, niin yhteistyön, kuin suunnittelun, tai hallinnan välineenä sen sijaan, että ne jaoteltaisiin ainoastaan välineiksi toimitusverkoston ja toimitusketjun eri osien integroimiseen. (Christopher, 1998.)

Monilla aloilla toimitusaikavaatimukset ovat tiukentuneet. Erityisesti kehitys korostuu elintarviketeollisuudessa. Toimitusaikaa mitataan usein jo tunneissa uusimmissa suurienkin määrien elintarviketeollisuuden toimitusketjuissa. Lähettämö- ja keräilytoiminnalle nopeat toimitusaikavaatimukset yhdistettynä kysynnän nopeisiin muutoksiin ja tuoreusvaatimuksiin aiheuttavat myös uusia vaatimuksia. Kun kyseessä ovat herkästi pilaantuvat elintarvikkeet ja elävät eläimet, käsiteltävä materiaali on hyvin herkkää varastoinnin ja kuljetuksen laadulle ja pituudelle. Verkostoa leimaavat elintarviketeollisuudessa useat rajapinnat ja niissä tapahtuvien toimenpiteiden läpinäkymättömyys ja suhteellisen suuri työmäärä. Kehityspotentiaali on ilmeinen ja sen realisoiminen tulee markkinoiden yhä globalisoituessa elintärkeäksi suomalaiselle elintarviketeollisuudelle. (Helo ja muut, 2007, s. 5-6.)

1.1 Tutkimuksen tavoite ja tutkimusmenetelmä

Tutkimuksen tavoitteena on suunnittelutieteen avulla kehittää artefakti, jonka avulla voidaan luoda mahdollisimman tehokas, tuottava ja ketterä logistinen laatikkojärjestelmä. Tässä tutkimuksessa laatikkojärjestelmällä tarkoitetaan tietojärjestelmää, jota käytetään toimitusketjun hallintaan ja laatikoiden kuljetusten ja sijainnin seurantaan.

Tutkimuksessa pyritään vastaamaan seuraavaan tutkimuskysymykseen: Millainen on metatason ohjeistus, jonka avulla pystytään luomaan vaatimusmäärittely, jonka pohjalta voidaan toteuttaa mahdollisimman tehokas, tuottava ja ketterä logistinen laatikkojärjestelmä?

Tutkimus toteutetaan suunnittelutieteellisenä tutkimuksena, koska siinä pyritään rakentamaan artefakti, jonka avulla ratkaistaan kohdeympäristön ongelma. Tutkimus noudattaa suunnittelutieteellisen tutkimuksen DSRM-prosessimallia (Design Science Reseach Methodology). Tutkimuksen aihe tuli toimeksiantona alfa-yritykseltä, joka omistaa ja hallinnoi logistista laatikkojärjestelmää. Järjestelmää käytetään elintarviketeollisuudessa alfa-yrityksen asiakkaiden toimesta. Laatikkojärjestelmän avulla seurataan ja

hallinnoidaan kuljetuslaatikoita. Alfa-yritys haluaa järjestelmästään tehokkaamman, tuottavamman ja ketterämmän. Aineistoa tutkimukseen kerätään tieteellisestä kirjallisuudesta ja alfa-yritykselle tehtävästä käytännön vaatimusmäärittelystä. Tutkimustuloksena syntyvän artefaktin relevanttiutta testataan demonstroimalla se alfa-yrityksen uuden laatikkojärjestelmän vaatimusmäärittelyprojektissa. Tutkimuksessa syntyvää artefaktia voidaan käyttää ohjeistuksena minkä tahansa logistisen laatikkojärjestelmän vaatimusmäärittelyssä.

1.2 Tutkimuksen kohde

Tutkimuksen kohteena on logististen tietojärjestelmien kehittäminen. Tässä tutkimuksessa keskitytään logistisiin laatikkojärjestelmiin. Tutkimus painottuu vahvasti vaatimusmäärittelyyn eli järjestelmien suunnitteluvaiheeseen. Tutkimus on erittäin ajankohtainen ja perusteltu, koska tietojärjestelmien rooli logistiikan tukena kasvaa ja kehittyy jatkuvasti. Lisäksi vastaavaa aikaisempaa metatason ohjeistusta logistisen laatikkojärjestelmän vaatimusmäärittelyyn ei ole olemassa.

Nykypäivän toimitusketjujen hallintaan on olemassa monenlaisia tietojärjestelmiä, joilla pyritään hankkimaan kilpailijoihin nähden strategista yliotetta ja maksimoimaan toiminnan tehokkuus (Helo ja muut, 2007, s. 16). Toimitusketjut oikein johdettuina muun muassa vähentävät läpimenoaikaa ja kustannuksia (Beesley, 1996). IT-järjestelmät ja teknologia ovat tulleet jäädäkseen prosessointiin ja toimitusketjujen informaatiovirtojen keruuseen, koska parempaa kontrollia vaaditaan pitkien toimitusketjujen hallinnan parantamiseksi (Prasad & Tata, 2000).

1.3 Tutkielman rakenne

Johdannon jälkeen tutkielmassa on kirjallisuuskatsaus, jossa esitellään kaikki tutkimuksen ymmärtämiseen vaadittava teoria. Kirjallisuuskatsauksessa perehdytään ensin

yleisellä tasolla lyhyesti ohjelmistotuotantoon. Kappaleessa tarkastellaan ohjelmistotuotantoa eri näkökulmista ja perehdytään ohjelmistotuotannon osa-alueisiin. Tämän jälkeen kirjallisuuskatsauksessa perehdytään vaatimustenkäsittelyyn. Vaatimustenkäsittely pitää sisällään vaatimusmäärittelyn ja vaatimustenhallinnan. Tämän jälkeen tutkielmassa esitellään tutkimuksessa käytettävä tutkimusmenetelmä, suunnittelutieteellinen tutkimus ja DSRM-prosessimalli. Tutkimusmenetelmän esittelyn jälkeen tutkielmassa on varsinainen tutkimusosio, jossa kuvataan artefaktin rakennusprosessi ja esitelleen tutkimustulokset. Viimeinen kappale on keskustelu, jossa tutkimustulokset analysoidaan.

2 Ohjelmistotuotanto

Ohjelmistotuotannolla tarkoitetaan tietokoneohjelmistojen rakentamisessa usein käytettyjä tekniikoita, menettelytapoja, työkaluja ja periaatteita. Käsite ohjelmisto sisältää sekä tietokoneohjelman että siihen kuuluvan dokumentaation. Termiä järjestelmä käytetään, kun kyseessä on laitteiston ja ohjelmiston muodostama kokonaisuus. Ohjelmistotuotanto koostuu periaatteista ja toimintatavoista, jotka ovat hyväksi havaittuja. Voidaan ajatella, että ohjelmistotuotannon tarkoituksena on kohtuulliset asiakkaan odotukset täyttävien tietokoneohjelmien tuottaminen siten, että aikataulu ja kehityskustannukset ovat riittävällä tarkkuudella ennustettavissa. Kaikki ohjelmistotyön vaiheet ovat osa ohjelmistotuotantoa. Nämä ovat määrittely, suunnittelu, toteutus, testaus ja laadunvarmistus. Myös tuotantoprosessin osa-alueet kuuluvat ohjelmistotuotannon piiriin. Näitä ovat muun muassa tuotteenhallinta, laatujärjestelmä, dokumentointi ja projektinhallinta. (Haikala & Mikkonen, 2011, s. 11-12.)

Ohjelmistotuotanto on tietojenkäsittelytieteen ala, joka on tekemisissä sellaisten ohjelmistojärjestelmien kanssa, joiden luomiseen tarvitaan yksi tai useampi ryhmä kehittäjiä järjestelmien monimutkaisuuden tai laajuuden vuoksi. Yleensä tällaisista järjestelmistä on olemassa useita versioita, ja ne ovat käytössä useita vuosia. Järjestelmät kokevat useita muutoksia niiden elinkaaren aikana. Muutokset voivat olla esimerkiksi virheiden korjaamista, olemassa olevien toimintojen parantamista, uusien ominaisuuksien lisäämistä, vanhojen ominaisuuksien poistamista tai järjestelmien muokkaamista siten, että ne toimivat uusissa käyttöympäristöissä. (Ghezzi, Jazayeri & Mandrioli, 2003, s. 1.)

Ghezzin ja muiden (2003, s. 1) mukaan ohjelmistotuotannon varhainen edelläkävijä David Parnas määritteli vuoden 1978 teoksessaan ohjelmistotuotannon ”moniversioisten ohjelmistojen rakentamiseksi usean henkilön toimesta.” Tämä määritelmä on ohjelmistotuotannon ytimessä ja se korostaa ohjelmoinnin ja ohjelmistotuotannon välistä eroa. Ohjelmoija koodaa valmiin ohjelmiston, kun taas ohjelmistosuunnittelija valmistelee ensin ohjelmiston komponentin, joka yhdistetään muiden ohjelmistosuunnittelijoiden valmistelemiin komponentteihin, jotka yhdessä muodostavat järjestelmän.

Ohjelmistosuunnittelijan valmistemaa komponenttia voidaan muokata toisten ohjelmistosuunnittelijoiden toimesta. Sitä voidaan käyttää myös muiden toimesta erilaisen version rakentamiseksi vielä kauan sen jälkeen, kun alkuperäinen suunnittelija on jättänyt projektin. Ohjelmointi on ensisijaisesti henkilökohtainen aktiviteetti, kun taas ohjelmistotuotanto on pohjimmiltaan ryhmäaktiviteetti.

Ohjelmistotuotanto on tekniikan tieteenala, joka kattaa kaikki ohjelmistotuotannon näkökulmat järjestelmän suunnitteluvaiheesta aina järjestelmän käyttöönoton jälkeiseen ylläpitovaiheeseen saakka. Ohjelmistotuotanto ei siis liity ainoastaan ohjelmistokehityksen teknisiin prosesseihin, vaan se sisältää myös ohjelmistoprojektien hallintaa ja sellaisten työkalujen, menetelmien ja teorioiden kehittämistä, jotka tukevat ohjelmistojen tuottamista. Ohjelmistotuotannossa perimmäisenä tarkoituksena on päästä haluttuun lopputulokseen vaaditun laadun, aikataulun ja budjetin puitteissa. (Sommerville, 2011, s. 7-8.)

2.1.1 Ohjelmistotuotanto projektina

Ohjelmistojen tuottaminen tyypillisesti organisoidaan projekteiksi, ja toimittajanäkökulma korostuu yleensä ohjelmistotuotannon koulutuksessa. Projektissa toteutetaan haluttu ohjelmisto asiakkaan vaatimusten perusteella. Projektit perustuvat asiakkaan liiketoiminnallisiin tavoitteisiin, eivätkä ne synny todellisuudessa tyhjästä. (Haikala & Mikkonen, 2011, s. 19.) Seuraavaksi tarkastellaan projekteja sekä **toimittajanäkökulmasta** että **asiakasnäkökulmasta**.

Asiakasnäkökulmasta ohjelmistoprojektilla on yleensä selkeät liiketoimintatavoitteet. Tästä syystä asiakas näkee usein projektin laajempana kokonaisuutena kuin ohjelmistotoimittaja. Asiakkaan näkökulmasta projektin tavoitteena voi olla yrityksen toiminnan kehittäminen liiketoimintaympäristön kannalta paremmaksi. Tällaisesta laajemmasta kokonaisuudesta voidaan käyttää nimitystä hanke, joka toteutetaan yksittäisinä projekteina. Usein asiakkaan on kuitenkin helpompi jakaa hankkeet useiksi osaprojekteiksi, jotta

voidaan välttää ylettömän pitkiä projekteja. Osaprojektit voivat olla esimerkiksi esitutkimusta, määrittelyä ja käyttöönottoa. (Haikala & Mikkonen, 2011, s. 19-20.)

Esitutkimusprojektin tekee usein asiakas itse. Siinä kartoitetaan vaihtoehdot, tehdään tarveanalyysi ja alustava määrittely, riskien ja kannattavuuden arviointi sekä päätös seuraavan vaiheen käynnistämisestä. Projektin lopputulos kuvataan määrittelyprojektissa. Siihen kuuluu muun muassa alustavan ohjelmistosuunnittelun tasolla lopputuloksen toiminnallinen määrittely, riskien arviointi ja päätös seuraavan vaiheen käynnistämisestä. Prosessin varsinainen tekninen toteutus tehdään toteutusprojektissa, johon sisältyy usein teknistä määrittelyä, yksityiskohtaista suunnittelua, ohjelmointia ja suuri osa testausta. (Haikala & Mikkonen, 2011, s. 20-21.)

Projekteissa hyödynnetään usein aliurakoitsijoita osaan työstä, vaikka tyypillisesti osa projekteista toteutetaan itse. Aliurakoitsijoita hyödynnetään, jotta oman yrityksen ei tarvitse hankkia työvoimaa ja osaamista, joita tarvitaan ainoastaan väliaikaisesti ohjelmiston toteutustyössä. Mahdollisten toimittajien kilpailutus, ylläpito ja käyttötuki voivat olla myös ohjelmistokehitykseen liittyviä tehtäviä osaprojektien lisäksi. Projekteilla on yhteistä niille määriteltävät tavoitteet ja selkeä sisältö. Projektit saavat käyttöönsä resursseja tavoitteiden saavuttamiseksi. Resurssit mahdollistavat projektin läpiviennin. Resursseja ovat esimerkiksi työhön osallistuva henkilöstö tai alihankintaan käytettävä raha. Projekteilla on oltava lisäksi aikataulu. Aikataulun laatiminen on sitä helpompaa mitä lyhempi ja pienempi projekti on kyseessä. (Haikala & Mikkonen, 2011, s. 21.)

Toimittajanäkökulmasta asiakkaalta tulevat vaatimukset ovat ohjelmistoprojektin lähtökohtana. Asiakkaan kannalta vaatimukset kuvaavat projektin sisältöä mahdollisimman tarkasti. Vaatimusten ollessa riittävän yksityiskohtaisia projekti on toisin sanoen kuvaus vaatimuksista toteutukseksi. Vaatimuksiin tulee kuitenkin aina tarkennuksia, lisäyksiä ja muutoksia koko projektin ajan, joten vaatimusten kuvaaminen näin tarkasti on käytännössä mahdotonta. Muutoksia tulee ajan myötä myös yrityksen omiin lähtökohtiin. Vaatimuksiin voi vaikuttaa eri tavoin useat toteutukseen liittyvät yksityiskohdat. Ennen

toteutusta näiden ottaminen huomioon on haastavaa, vaikka tunnettaisiinkin vaatimukset. Tämän takia usein tyydytäänkin riittävän tarkkoihin vaatimuksiin ja yritetään varautua projektin aikana tapahtuviin muutoksiin. Poikkeuksena voidaan pitää turvallisuuskriittisiä järjestelmiä, joiden muutosten aiheuttamat riskit halutaan minimoida. (Haikala & Mikkonen, 2011, s. 21-22.)

Projektissa muutosten tekemiseen ja uusien ominaisuuksien lisäämiseen vaikuttaa tekijän ja tilaajan välinen sopimus. Projekti voidaan suorittaa aikataulun mukaisesti, jossa toteutetaan kiinteään hintaan ennalta määritellyt ominaisuudet. Projekti voi olla myös sellainen, jossa ei noudateta kiinteää aikataulua. Tällaisessa projektissa tehdään töitä tuntiperustaisesti ja tekijä on valmis ottamaan välittömästi huomioon tilaajan muutospyynnöt. Pyrkimyksenä on yleensä päästä jonkinlaiseen sopuun välimuodosta, jossa laskutetaan tuntiperustaisesti vaatimusmuutoksista ja alkuperäiset asiakasvaatimukset toteutetaan kiinteällä hinnalla. Toinen nykyohjelmistotuotantoa kuvaava ominaisuus jatkuvien muutosten lisäksi on se, ettei uusia ohjelmistoja yleensä tuoteta tyhjästä. Ohjelmistot pohjautuvat johonkin sovelluskehykseen, ohjelmistoalustaan, aiempaan versioon tai johonkin toiseen järjestelmään, joka on saatavilla hieman erilaiseen käyttötarkoitukseen tai toiseen ympäristöön. (Haikala & Mikkonen, 2011, s. 23.)

Tuotekeskeisyyden ja asiakaskohtaisuuden välinen tasapainoilu on ohjelmistotyössä haastavaa. Asiakkaat tulevat sitä tyytyväisemmiksi, mitä tarkemmin heidän vaatimuksensa otetaan huomioon. Toisaalta sitä useampaa järjestelmää joudutaan ylläpitämään, mitä räätälöityneempiä erillisiä ohjelmistoja tehdään. Mikäli löydettäisiin usealle asiakkaalle yhteisiä vaatimuksia, olisi yhden ohjelmistotuotteen toteuttaminen ehkä mahdollista. Tällaisessa tilanteessa sopivasti konfiguroituna ohjelmistotuote täyttäisi kaikkien asiakkaiden vaatimukset ja ylläpito helpottuisi. On toki mahdollista, etteivät kaikki asiakkaat halua samoja muutoksia, vaan heillä voi olla omia vaatimuksia. (Haikala & Mikkonen, 2011, s. 23.)

2.1.2 Ohjelmistotuotannon osa-alueet

Ohjelmisto kehittyy ja muuttuu asteittain aina ideasta siihen saakka, kunnes se on saatu valmiiksi ja toimitettu asiakkaalle tai jopa vielä tämän jälkeenkin. Sanotaan, että ohjelmistoilla on ”elinkaari”, joka koostuu useista vaiheista. Jokaisessa vaiheessa saadaan aikaan ohjelmiston osa tai jotain, mikä liittyy kyseiseen ohjelmistoon, esimerkiksi testisuunnitelma tai käyttöohje. Perinteisessä ohjelmistotuotannon elinkaarimallissa, vesiputousmallissa, jokaisella vaiheella on selkeä aloitus- ja lopetuspiste, joissa saadaan aikaiseksi selkeät tuotokset seuraavaa vaihetta varten. Käytännössä ohjelmistotuotanto on harvoin näin yksinkertaista. (Ghezzi ja muut, 2003, s. 6.)

On olemassa useita erilaisia ohjelmistoprosesseja, mutta niiden kaikkien on sisällettävä vähintään neljä vaihetta, jotka ovat olennaisia ohjelmistotuotannolle. Näistä ensimmäinen on **vaatimusmäärittely**. Tässä vaiheessa määritellään ohjelmiston toiminnallisuudet ja toiminnan rajoitukset. Seuraava vaihe on **ohjelmiston suunnittelu ja toteutus**. Tässä vaiheessa ohjelmisto tuotetaan vaatimusten mukaisesti. Kolmas vaiheista on **ohjelmiston validointi**. Tässä vaiheessa varmistetaan, että ohjelmisto toimii käyttäjien haluamalla tavalla. Viimeinen vaihe on **ohjelmiston ylläpito**. Ohjelmiston on kehityttävä vastaamaan muuttuvia asiakkaiden tarpeita. (Sommerville, 2011, s. 28.)

Edellä mainitut vaiheet ovat kaikki osana jokaisessa ohjelmistoprosessissa jollain tasolla. Käytännössä nämä vaiheet ovat kuitenkin monimutkaisia aktiviteetteja ja ne sisältävät itsessään useita erilaisia vaiheita. Kaikki älylliset ja luovat prosessit ovat ihmisten tekemien päätösten ja ratkaisujen varassa. Ei ole olemassa yhtä optimaalista prosessia. Monet organisaatiot ovat kehittäneet jopa omia ohjelmistokehitysprosesseja. Prosessit ovat muovautuneet organisaatioissa olevien ihmisten kykyjen ja kehitettävän ohjelmiston ominaispiirteiden mukaan. Joillekin ohjelmistoille, kuten esimerkiksi kriittisille järjestelmille, vaaditaan jäsennelty kehitysprosessi. Liiketoimintajärjestelmiin, joissa vaatimukset muuttuvat nopeasti, vähemmän muodollinen, joustava prosessi on todennäköisesti tehokkaampi. (Sommerville, 2011, s. 28-29.)

Ohjelmistoprosessit voidaan luokitella joko suunnitelmavetoisiksi tai ketteriksi prosesseiksi. Suunnitelmavetoiset prosessit ovat prosesseja, joissa kaikki toiminnot suunnitellaan etukäteen ja edistymistä arvioidaan suunnitelman perusteella. Ketterissä prosesseissa suunnittelu on inkrementaalista. Inkrementaalisuudella tarkoitetaan sitä, että ohjelmistoa kehitetään asteittain. Jokainen versio lisää toiminnallisuuksia aikaisempaan versioon. Ketterät prosessit palvelevat myös paremmin muuttuvia asiakkaiden vaatimuksia. (Sommerville, 2011, s. 29-30.) Tässä tutkielmassa ei perehdytä tarkemmin jokaiseen ohjelmistotuotantomenetelmään. Seuraavaksi kuitenkin esitellään havainnollistavana esimerkkinä ohjelmistotuotannon perinteisin malli, vesiputousmalli. Tarkoituksena on luoda kokonaisvaltainen ja selkeä kuva ohjelmistotuotannon osa-alueista. Vesiputousmalli sisältää seuraavat vaiheet:

Vaatimusten analysointi ja määrittely. Vaatimusten analyysi on yleensä ensimmäinen vaihe laajamittaisessa ohjelmistokehitysprojektissa. Se suoritetaan esitutkimuksen jälkeen. Esitutkimuksessa määritellään ohjelmiston hyödyt ja tarkat kustannukset. Tämän vaiheen tarkoituksena on tunnistaa ja dokumentoida järjestelmän tarkat vaatimukset. Vaatimusten analysointi ja määrittely voidaan suorittaa asiakkaan, ohjelmistokehittäjän, markkinointiorganisaation tai minkä tahansa näistä muodostettavan yhdistelmän toimesta. Tapauksissa, joissa vaatimukset eivät ole selkeitä, vaaditaan paljon vuorovaikutusta asiakkaan ja ohjelmistokehittäjän välillä. Esimerkkinä tällaisesta tilanteesta on järjestelmä, jota ei ole vielä koskaan tehty. Tässä vaiheessa vaatimusten pitäisi olla loppukäyttäjien ymmärrettävissä. Eri ohjelmistotuotantomenetelmien mukaan tässä vaiheessa pitäisi tehdä myös käyttöohjeita ja suunnitelmia ohjelmistotestejä varten, jotka suoritetaan ennen ohjelmiston toimitusta. (Ghezzi ja muut, 2003, s. 6.)

Ohjelmiston suunnittelu. Kun ohjelmiston vaatimukset on dokumentoitu, ohjelmistokehittäjät suunnittelevat ohjelmiston vastaamaan vaatimuksia. Tämä vaihe jaetaan joskus kahteen osaan, arkkitehtuurisuunnitteluun ja yksityiskohtaiseen suunnitteluun. Arkkitehtuurisuunnittelussa tehdään järjestelmästä korkean tason kuva, josta selviää yleisellä tasolla siinä käytettävät komponentit ja niiden välinen vuorovaikutus. Yksityiskohtaisessa

suunnittelussa komponentit hajotetaan alempitasoisiin moduuleihin, joissa on tarkasti määritellyt rajapinnat. (Ghezzi ja muut, 2003, s. 6-7.) Moduuli on järjestelmän komponentti, joka tarjoaa muille moduuleille palveluja ja käyttää muiden moduuleiden palveluja. Moduuli ei ole itsenäinen, kuten osajärjestelmä, vaan se toteuttaa osajärjestelmän toiminnot muiden moduulien kanssa tiiviissä yhteystyössä. (Taina, 2006.) Kaikki suunnittelutasot on dokumentoitu määrittelydokumentteihin, jotka seuraavat suunnittelupäätöksiä (Ghezzi ja muut, 2003, s. 7).

Vaatimusten analysointivaiheen ja ohjelmiston suunnitteluvaiheen voi erottaa ”mitä-miten” kahtiajaolla. Yleisen periaatteen mukaan on pystyttävä erottelemaan selkeästi mikä on ongelma ja miten se ratkaistaan. Tässä tapauksessa ohjelmiston määrittelyvaiheessa pyritään määrittelemään mikä on ongelma ja suunnitteluvaiheessa pyritään ratkaisemaan se. Suunnitteluvaiheen tarkoituksena on suunnitella sellainen ohjelmisto, joka täyttää asetetut vaatimukset. (Ghezzi ja muut, 2003, s. 7.)

Koodaus ja moduulitestaus. Tässä vaiheessa tuotetaan varsinainen koodi, joka toimitetaan asiakkaalle toimivana ohjelmistona. Myös muissa ohjelmiston elinkaaren vaiheissa voidaan tuottaa koodia esimerkiksi prototyyppeihin, testeihin ja ajureihin, mutta nämä ovat ainoastaan kehittäjien käytössä. Koodausvaiheessa tuotetut yksittäiset moduulit testataan ennen kuin ne toimitetaan seuraavaan vaiheeseen. (Ghezzi ja muut, 2003, s. 7.) Sommerville (2011, s. 31) on nimennyt tämän vaiheen toteutukseksi ja yksikkötestaukseksi. Ohjelmiston suunnittelu realisoituu ohjelmina tai ohjelmayksikköjen joukkona. Yksikkötestauksessa varmistetaan, että jokainen yksikkö vastaa määrittelyä.

Integrointi ja järjestelmän testaus. Kaikki aikaisemmin tuotetut ja yksitellen testatut moduulit yhdistetään ja testataan yhtenäisenä järjestelmänä tässä vaiheessa. (Ghezzi ja muut, 2003, s. 7.)

Toimitus ja ylläpito. Kun järjestelmä läpäisee kaikki testit, se toimitetaan asiakkaalle. Tästä alkaa ylläpitovaihe. Kaikki alkuperäisen toimituksen jälkeiset järjestelmään

tehtävät muutokset luokitellaan yleensä tähän vaiheeseen. (Ghezzi ja muut, 2003, s. 7.) Ylläpito sisältää aikaisemmin paikantamattomien virheiden korjaamista, järjestelmäyksiköiden toteutuksen parantamista ja järjestelmän palveluiden parantamista, kun uusia vaatimuksia löydetään (Sommerville, 2011, s. 31).

3 Vaatimustenkäsittely

Vaatimus on ominaisuus, joka tuotteella tulee olla tai jotain, mitä tuotteella pystyy tekemään (Haikala & Mikkonen, 2011, s. 61). Järjestelmää koskevat vaatimukset ovat kuvauksia siitä, mitä järjestelmän pitäisi tehdä, palveluita, joita se tarjoaa ja sen toimintarajoituksia (Sommerville, 2011, s. 83). Vaatimukset voivat vaihdella korkean tason abstrakteista kuvauksista formaaleihin, matemaattisesti täsmällisiin, määrittelyihin. Syynä vaatimusten vaihtelevaan esitysmuotoon on sidosryhmien tarpeet eri tasoilla. Sidosryhmillä on myös vaihtelevat kyvyt luoda ja lukea esitysmuotoja, mikä johtaa vaatimusten monipuoliseen laatuun. (Laplante, 2014, s. 3-4.)

Vaatimukset voivat olla **asiakasvaatimuksia** tai **ohjelmistovaatimuksia**. Asiakasvaatimukset lähtevät suoraan asiakkaiden tarpeista, eikä niiden kuvaamiseen tarvita yleensä sovellusalueen sanaston lisäksi muuta sanastoa. Joskus käytetään myös termiä ominaisuus, jolloin tarkoitetaan yleensä asioita, jotka liittyvät ohjelman toiminnallisuuteen. Ominaisuudet ratkaisevat yhdessä asiakkaan ongelman. Asiakasvaatimukset toteutetaan ohjelmistovaatimuksilla. Ohjelmistovaatimukset ovat usein toimintoja, jotka pyrkivät määrittelemään, miten kyseinen asiakasvaatimus toteutettavassa ohjelmistossa tuotetaan käyttäjälle. (Haikala & Mikkonen, 2011, s. 62.)

Vastaavista vaatimuksista voidaan käyttää myös termejä käyttäjävaatimukset ja järjestelmävaatimukset. Käyttäjävaatimukset ovat luonnollisen kielen kuvauksia ja diagrammeja siitä, mitä palveluita järjestelmän odotetaan tarjoavan järjestelmän käyttäjille ja rajoitteita, joiden puitteissa järjestelmän on toimittava. Järjestelmävaatimukset ovat tarkempia kuvauksia järjestelmän toiminnoista, palveluista ja toimintarajoituksista. Järjestelmän vaatimusdokumentin pitäisi määrittää tarkalleen, mitä toteutetaan. Joskus tätä dokumenttia kutsutaan myös toiminnalliseksi määrittelyksi. Se voi olla osana järjestelmän ostajan ja ohjelmistokehittäjän välistä sopimusta. (Sommerville, 2011, s. 83.)

Vaatimukset voidaan luokitella myös **toiminnallisiin** ja **ei-toiminnallisiin** vaatimuksiin. Toiminnalliset vaatimukset esittävät, mitä palveluita järjestelmän pitäisi sisältää, miten

järjestelmän pitäisi reagoida tiettyihin syötteisiin (engl. inputs) ja miten järjestelmän pitäisi käyttäytyä tietyissä tilanteissa. Joissain tapauksissa toiminnalliset vaatimukset voivat myös esittää, mitä järjestelmän ei pitäisi tehdä. Ei-toiminnalliset vaatimukset ovat järjestelmän palveluiden ja toiminnallisuuksien rajoituksia. Ne sisältävät ajoitusrajoituksia, kehitysprosessin rajoituksia ja standardien asettamia rajoituksia. Ei-toiminnalliset rajoitukset pätevät usein järjestelmään kokonaisuudessaan yksittäisten ominaisuuksien tai palveluiden sijaan. (Sommerville, 2011, s. 84-85.) Ei-toiminnalliset vaatimukset kertovat, mitä ehtoja järjestelmän on täytettävä, jotta toiminnalliset vaatimukset voivat toteutua. Tästä syystä ei-toiminnalliset vaatimukset eivät suoraan liity palveluihin. (Digitaalinen Helsinki, 2019.)

Vaikein yksittäinen vaihe ohjelmistojärjestelmän rakentamisessa on päättää tarkalleen, mitä rakennetaan. Mikään muu osa käsitteellistä työtä ei ole yhtä vaikeaa määriteltäessä yksityiskohtaisia teknisiä vaatimuksia, mukaan lukien rajapinnat ihmisille, koneille ja muille ohjelmistojärjestelmille. Mikään muu työn osa ei pilaa työn tuloksia, jos se tehdään väärin. Mitään muuta osaa ei ole vaikeampaa korjata myöhemmin. (Brooks, 1986.) Monet asiantuntijat ovat todenneet saman, ja useat tutkimukset ovat osoittaneet, että yli 60-prosenttia epäonnistuneista ohjelmistoprojekteista ovat kaatuneet huonoon **vaatimustenkäsittelyyn** (engl. requirements engineering). Huolellinen vaatimustenkäsittely on ohjelmistoprojektin yksi perusedellytyksistä. Vaatimustenkäsittely on yhdistävä tekijä muiden ohjelmistotuotantoon liittyvien aktiviteettien välillä. (Haikala & Mikkonen, 2011, s. 61.)

Vaatimustenkäsittely on tekniikan haara, joka liittyy järjestelmien todellisiin tavoitteisiin, toimintoihin ja rajoituksiin. Vaatimustenkäsittely liittyy myös näiden tekijöiden välisiin suhteisiin järjestelmän käyttäytymisen tarkan määrittelyn ja järjestelmään ja siihen liittyvien järjestelmien ajan myötä tapahtuvan kehityksen vuoksi. (Laplante, 2014, s. 2-3.) Vaatimustenkäsittely voidaan jakaa kahteen eri osa-alueeseen, vaatimusmäärittelyyn ja vaatimustenhallintaan (Haikala & Mikkonen, 2011, s. 65).

On huomionarvoista, että vaatimustenkäsittelyn ja vaatimusmäärittelyn käsitteet ja suomennokset vaihtelevat hyvinkin paljon riippuen lähteestä. Esimerkiksi Sommerville (2011, s. 36) puhuu käsitteistä jopa samana asiana, ”software specification or requirements engineering.” Sommerville ei siis tavallaan erottele käsitteitä toisistaan, vaan pitää niitä vaihtoehtoisina termeinä. Sen sijaan Laplante (2014, s. 2-3) antaa määritelmän vaatimustenkäsittelylle (engl. requirements engineering), mutta Laplante ei enää erikseen määrittele vaatimusmäärittelyä. Haikala ja Mikkonen (2011, s. 61-65) nimittävät termiä ”requirements engineering” vaatimustenkäsittelyksi ja ”requirements definition” termiä vaatimusmäärittelyksi. Professori Jukka Paakin (2011) luentomateriaalin mukaan vaatimusmäärittely tulee termistä ”requirements engineering.” Eriävien määritelmien vuoksi helpointa on ajatella, että vaatimustenkäsittely on ainoastaan **yläkäsite**, joka pitää sisällään vaatimusmäärittelyn ja vaatimustenhallinnan. Seuraavaksi perehdytään näihin osa-alueisiin.

3.1 Vaatimusmäärittely

Vaatimusmäärittelyssä pyritään selvittämään mahdollisimman perusteellisesti järjestelmän asiakasvaatimukset. Vaatimusten tärkeimmät lähteet ovat asiakas ja ohjelmiston tulevat käyttäjät. Vaatimukset perustuvat liiketoiminnan tarpeisiin. (Haikala & Mikkonen, 2011, s. 65.) Tekijöille on usein melko epäselvä kuva siitä, millainen uuden palvelun tulisi olla, kun sitä aletaan suunnittelemaan. Vaatimusmäärittelyvaiheen tavoitteena on muuttaa tämä epäselvä mielikuva selkeäksi ja ristiriidattomaksi uuden palvelun kuvaukseksi. (Sinkkonen, Nuutila & Törmä, 2009, s. 49.) Vaatimusmäärittelyssä selvitetään, miten löydetyt vaatimukset saadaan kuvattua jatkokehitykseen soveltuvalla tavalla ja mitä järjestelmältä vaaditaan (Paakki, 2011).

Vaatimusmäärittely tarkoittaa eri asioita eri ihmisille. Periaatteessa ainoastaan muutamista asioista ollaan samaa mieltä. Järjestelmälle asetetaan toiminnallisia ja ei toiminnallisia vaatimuksia, jotka sen on täytettävä. Osa vaatimuksista on järjestelmää koskevia rajoitteita. Vaatimukset tulevat sidosryhmiltä ja kukin sidosryhmä on jollakin tapaa

tekemisissä järjestelmän kanssa. Lisäksi vaatimukset on kuvattava asianmukaisella tavalla järjestelmän ominaisuuksiksi. (Paakki, 2011.) Tässä tutkielmassa käytetään Haikalan ja Mikkosen (2011, s. 66-67) käyttämää vaatimusmäärittelyn vaiheistusta. Vaatimusmäärittelyprosessin vaiheet ovat kartoittaminen, analysointi, dokumentointi ja validointi.

Sommervillen mukaan esitutkimus on vaatimusmäärittelyprosessin ensimmäinen vaihe. Esitutkimuksessa tehdään arvio, voidaanko tunnistetut käyttäjän tarpeet tyydyttää nykyisiä ohjelmisto- ja laitetekniikoita käyttämällä. Tutkimuksessa otetaan huomioon, onko ehdotettu järjestelmä liiketoiminnan kannalta kustannustehokas ja voidaanko sitä kehittää nykyisten budjettirajoitteiden puitteissa. Esitutkimuksen pitäisi olla suhteellisen halpa ja nopea. Tutkimuksen tulos osoittaa, kannattaako projektia jatkaa yksityiskohtaisempaan analyysiin. (Sommerville 2011, s. 37.) Esitutkimus voidaan mieltää joko vaatimusmäärittelyprosessin ensimmäiseksi tai sitä edeltäväksi vaiheeksi. Joka tapauksessa esitutkimus on suoritettava ohjelmistoprojektin alussa. Seuraavaksi tutkielmassa esitellään Haikalan ja Mikkosen (2011, s. 66-67) käyttämät vaatimusmäärittelynprosessin vaiheet.

3.1.1 Kartoittaminen

Vaatimusten kartoittaminen tai vaatimusten löytäminen on prosessi, jossa kerätään informaatiota vaaditusta järjestelmästä ja olemassa olevista järjestelmistä ja erotellaan käyttäjä- ja järjestelmävaatimukset. Informaatiolähteitä vaatimusten löytämisvaiheen aikana ovat dokumentaatiot, järjestelmän sidosryhmät ja vastaavien järjestelmien määrittelyt. Kartoitusvaiheessa ollaan vuorovaikutuksessa sidosryhmien kanssa haastattelujen ja havainnoinnin avulla. Kartoituksessa voidaan käyttää myös skenaarioita ja prototyyppejä auttamaan sidosryhmiä ymmärtämään, millainen järjestelmä tulee olemaan. Järjestelmän sidosryhmien lisäksi vaatimuksia voi tulla myös samalta toimialueelta ja muilta järjestelmiltä, jotka ovat vuorovaikutuksessa määriteltävän järjestelmän kanssa. Kaikki nämä on otettava huomioon vaatimusten kartoitusprosessissa. (Sommerville, 2011, s. 103.)

Vaatimusten kartoittaminen ja löytäminen sisältävät asiakkaan tarpeiden ja halujen selvittämistä. Osa vaatimuksista on ilmeisiä, mutta monet vaatimukset on saatava asiakkaalta hyvin määriteltyjen lähestymistapojen avulla. Vaatimusten kartoitukseen kuuluu myös sidosryhmien löytäminen. (Laplante, 2014, s. 11-12.)

Tehtävän onnistumisen kannalta vaatimusten kokoaja on keskeisessä roolissa. Vaatimusten kokoojan rooli ei saa olla passiivinen erilaisten vaatimusehdotusten ja vaatimusdokumenttien vastaanottajana, koska tällöin saadaan kerättyä ainoastaan sidosryhmien tiedostamat tarpeet. Mahdolliset todelliset tarpeet saattavat jäädä huomioimatta. Esitetyt vaatimukset voivat myös johtaa ylimääräisiin kustannuksiin tai ne voivat olla epärealistisia. Lisäksi eri sidosryhmien vaatimukset voivat olla ristiriitaisia. Vaatimusten kokoojan on oltava pikemminkin vaatimusten hallitsija ja tarvittaessa hänen on kyettävä tarkistamaan ja tarkentamaan vaatimuksia sidosryhmien kanssa. Vaatimusten ehdottaminen kustannussäästöjen ja synergiaetujen aikaansaamiseksi on myös mahdollista. (Kosola, 2013, s. 30-31.)

Eri vaatimusten lähteitä, kuten sidosryhmiä, toimialuetta ja järjestelmiä voidaan tarkastella eri näkökulmista. Kukin näkökulma kuvaa järjestelmävaatimusten osajoukkoa. Eri näkökulmat ongelmaan näkevät ongelman eri tavoin. Näkökulmat eivät kuitenkaan ole täysin riippumattomia, vaan ne ovat usein päällekkäisiä, joten niillä on yhteisiä vaatimuksia. Näkökulmia voidaan käyttää sekä vaatimusten löytämisessä että vaatimusten dokumentoinnissa. (Sommerville, 2011, s. 103-104.) Jotta vaatimuksista saadaan mahdollisimman kattavat, pyritään vaatimukset kartoittamaan eri näkökulmista (Taina, 2005).

Vaatimusten kartoittamiseen löytyy paljon erilaisia tekniikoita. Nuseibeh ja Easterbrook (2000) ovat jakaneet kartoitustekniikat kuuteen eri ryhmään. Seuraavaksi tutkielmassa esitellään nämä ryhmät lyhyesti.

Perinteiset tekniikat sisältävät laajan valikoiman yleisluontoisia tiedonkeruutekniikoita. Perinteisiä tekniikoita ovat kyselylomakkeiden käyttö, tutkimukset, haastattelut ja

olemassa olevan dokumentaation, kuten organisaatiokaavioiden, prosessimallien tai standardien ja olemassa olevien järjestelmien käyttöohjeiden tai muiden ohjeiden analysointi. (Nuseibeh & Easterbrook, 2000.)

Ryhmätyöskentelytekniikat (engl. group elicitation techniques) pyrkivät edistämään sidosryhmien välistä yhteisymmärrystä samalla, kun joukkuedynamiikkaa hyödynnetään tarpeiden kokonaisvaltaisempaan ymmärtämiseen. Ryhmätyöskentelytekniikat sisältävät aivoriihiä, kohderyhmiä sekä RAD- ja JAD-työpajoja. (Nuseibeh & Easterbrook, 2000.)

Prototyyppejä käytetään kartoituksessa, kun vaatimuksiin liittyy paljon epävarmuutta tai kun sidosryhmiltä tarvitaan aikaista palautetta. Prototyyppien käyttö on helposti yhdistettävissä muihin tekniikoihin. Prototyyppiä voidaan käyttää esimerkiksi keskustelun herättämäiseksi ryhmätyöskentelytekniikoissa tai kyselyiden perustana tai ääneen ajattelu protokollana. (Nuseibeh & Easterbrook, 2000.)

Malliperustaiset tekniikat (engl. Model-driven techniques) tarjoavat tietyn mallin riippuen kerättävästä informaatiosta ja käyttävät tätä mallia kartoitusprosessin edistämiseksi. Malliperustaiset tekniikat sisältävät tavoite- ja skenaariopohjaisia menetelmiä. (Nuseibeh & Easterbrook, 2000.) Malliperustaiset tekniikat pyrkivät kuvaamaan järjestelmää mallien avustuksella. Vaikka malliperustaiset tekniikat suunniteltiin alun perin toteutusta, suunnittelua ja arkkitehtuuriin liittyvää mallinnusta varten, myös vaatimustenkäsittelyssä on hyödytty niistä. Malliperustaiset tekniikat voivat auttaa rakentamaan varhaisia järjestelmä- tai arkkitehtuurimalleja automaattisesti vaatimuksista, esimerkiksi johtamalla yksityiskohtaisempi UML-malli tavoite- tai skenaariomallista, mikä mahdollistaa vaatimusmallien ja arkkitehtuurimallien tiiviimmän integraation. (Chuan & Mussbacher, 2016.) Malliperustaisen ohjelmistotuotannon suosituin kieli on UML. Sen avulla voidaan määritellä, mitä järjestelmän täytyy tehdä. Alemmalla abstraktiotasolla määritellään myös, miten tämä käytös saavutetaan. (Nwokeji, Clark & Barn, 2013.)

Kognitiiviset tekniikat sisältävät sarjan tekniikoita, jotka on alun perin kehitetty tiedon hankkimiseen tietopohjaisille järjestelmille. Tällaisia tekniikoita ovat protokolla-analyysi, porrastaminen, korttien lajittelu ja repertory grid -tekniikka. (Nuseibeh & Easterbrook, 2000.) Ääneen ajattelu -menetelmässä eli protokolla-analyysissä tutkija pyytää koehenkilöitä kertomaan ääneen, miten he havaitsevat, tulkitsevat tai päättelevät asioita (Jyväskylän yliopiston Koppa, 2015). Porrastamisen on tekniikka, jonka tavoitteena on saada selville sidosryhmien tuntemukset ja tuntemusten rakenne tiedustelemalla (Nuseibeh & Easterbrook, 2000). Korttien lajittelussa osallistujia pyydetään lajittelemaan ryhmiin erillisille korteille kirjoitettuja aiheita. Korttien lajittelu on helposti käyttäjien ymmärtämä menetelmä ja se toimii erityisesti erilaisten valikkorakenteiden käyttäjälähtöisessä suunnittelussa. (Handlaamo, 2018.) Repertory grid -tekniikka on psykologiasta omaksuttu haastattelumenetelmä, jota käytetään tuotekehityksessä asiakkaiden piilevien tarpeiden paljastamiseksi (Baxter, Goffin & Szwejczewski, 2014).

Kontekstuaaliset tekniikat ilmaantuivat 1990-luvulla perinteisten- ja kognitiivisten tekniikoiden vaihtoehdoksi. Kontekstuaaliset tekniikat sisältävät etnografisten tekniikoiden käyttöä, esimerkiksi osallistujien havainnointia. Ne sisältävät myös etnometodologian ja keskusteluanalyysin, jotka molemmat soveltavat hienorakenteista analyysiä keskusteluja vuorovaikutusmallien tunnistamiseksi. (Nuseibeh & Easterbrook, 2000.)

Taulukko 1. Vaatimusten kartoitustekniikat (Tuunanen, 2005, s. 17).

Ryhmä	Esimerkkejä tekniikasta
1. Perinteiset tekniikat	Kyselyt ja tutkimukset, haastattelut, olemassa olevien dokumenttien analysointi
2. Ryhmätyöskentelytekniikat	Aivoriihet, kohderyhmät, RAD- ja JAD-työpajat
3. Prototyypit	Prototyyppien luominen käyttöliittymän varhaisista versioista
4. Malliperustaiset tekniikat	Tavoite- ja skenaariopohjaiset tekniikat

5. Kognitiiviset tekniikat	Protokolla-analyysi, porrastaminen, korttien lajittelu, repertory grid -tekniikka
6. Kontekstuaaliset tekniikat	Etnografiset tekniikat, kuten osallistujien havainnointi

Taulukossa 1 on Tuunasen (2005, s. 17) kokoama taulukko Nuseibehin ja Easterbrookin (2000) vaatimusten kartoitustekniikoista. Tässä taulukossa kartoitustekniikat on vaihdettu alkuperäisen artikkelin mukaiseen esitysjärjestykseen. Lisäksi taulukon sisältö on vapaasti suomennettu Tuunasen väitöskirjasta.

3.1.2 Analysointi

Vaatimusten analysointiin sisältyy tekniikoita, joita käytetään ongelmiin, jotka liittyvät asiakkailta kerättyihin käsittelemättömiin vaatimuksiin. Käsittelemättömät vaatimukset voivat sisältää muun muassa seuraavanlaisia ongelmia. Vaatimukset eivät aina ole järkeviä ja ne ovat usein ristiriidassa keskenään. Vaatimukset voivat olla myös epäjohdonmukaisia ja keskeneräisiä. Lisäksi ne voivat olla epämääräisiä ja riippuvaisia toisistaan. (Laplante, 2014, s. 12.)

Vaatimusten analysointi on usein informaalia. Asiakkaille ja sidosryhmille voidaan esimerkiksi esittää kartoitetut vaatimukset molemmille ymmärrettävässä muodossa, jotta saadaan varmuus, että vaatimusten käsittelijä on tulkinnut kaikki toiveet oikein. Formaalit menetelmät tarjoavat täsmällisempiä tekniikoita vaatimusten analysointia varten. (Laplante, 2014, s. 81.) Tässä kappaleessa formaaleihin menetelmiin ei perehdytä tarkemmin.

Vaatimusten analyysissä voidaan tarkentaa vaatimuksia sekä selvitellä niiden keskinäisiä suhteita ja prioriteettia (Haikala & Mikkonen, 2011, s. 66). Vaatimusten analysoinnissa tarkoituksena on arvioida, ovatko vaatimukset riittävän laadukkaita, jotta niiden pohjalta voidaan aloittaa ratkaisun suunnittelu ilman merkittäviä riskejä. Lisäksi analysoinnissa on

tunnistettava, mitkä ovat suorituskyvyn, aikataulun ja kustannusten kannalta keskeisimmät vaatimukset. Vaatimuksia voidaan arvioida yksilöinä, kaikista vaatimuksista muodostuvana kokonaismassana tai vaatimusten muodostamana verkkona. (Kosola, 2013, s. 48.)

Vaatimusten analyysissa tarkoituksena on hahmottaa ne vaatimukset, jotka on saavutettava korkealaatuisen järjestelmän luomiseksi. Ilman vaatimusten analyysiä on erittäin todennäköistä, että rakennetaan ohjelmisto, joka ratkaisee väärän ongelman. Tuloksena on hukattua aikaa, rahaa, henkilökohtaista turhautumista ja tyytymättömiä asiakkaita. Vaatimusten analyysissa vaatimuksia tarkennetaan ja analysoidaan niiden selkeyden, valmiuden ja johdonmukaisuuden arvioimiseksi. Vaatimusten analyysin tulokset on muutettava esitettävään muotoon. Vaatimukset voidaan esittää prototyyppiä, määrittelyä (engl. specification) tai symbolista mallia käyttämällä. Vaatimusten analyysin tuloksia arvioidaan vaatimusten selkeyden, valmiuden ja johdonmukaisuuden perusteella. (Pressman, 2000, s. 266-267.)

Vaatimusten analyysi sisältää vaatimusten tarkastamista, jolla varmistetaan, että kaikki sidosryhmät ymmärtävät vaatimukset. Lisäksi analyysissä tutkitaan vaatimukset virheiden, puutteiden ja muiden heikkouksien varalta. Analyysi sisältää korkean tason vaatimusten hajottamisen yksityiskohtiin, prototyyppien rakentamisen, toteutettavuuden arvioinnin ja prioriteettineuvottelut. Tavoitteena on kehittää riittävän laadukkaita ja yksityiskohtaisia vaatimuksia, jotta johtavat voivat tehdä realistisia projektiarvioita ja tekninen henkilöstö voi edetä suunnittelussa, rakentamisessa ja testauksessa. Usein on hyödyllistä esittää osa vaatimuksista usealla tavalla, esimerkiksi kirjallisessa ja graafisessa muodossa. Erilaiset näkökulmat paljastavat näkemyksiä ja ongelmia, joita yksi näkökulma ei pysty tarjoamaan. Useat näkökulmat auttavat sidosryhmiä pääsemään yhteiseen näkemykseen siitä, mitä he saavat, kun tuote toimitetaan. (Wiegers, 2003, s. 50.)

3.1.3 Dokumentointi

Vaatimusten dokumentoimiseen tulee kiinnittää erityistä huomiota, koska vaatimukset muodostavat keskeisen pohjan tietojärjestelmän elinkaaren seuraavien vaiheiden suorittamiselle (Pohjonen, 2002, s. 30). Vaatimusmäärittely nimetään ja jaotellaan alan kirjallisuudessa usealla eri tavalla. Synonyymeinä käytetään termejä vaatimusmäärittely, analyysi ja määrittely. Määrittelyyn liittyvät tehtävät ovat käytännössä projektin toteuttamiskelpoisuuden ja tarpeellisuuden selvittäminen, vaatimusten ja tavoitteiden asettaminen sekä ratkaisumallin laatiminen. Tärkein vaatimusmäärittelyprosessin tuloksena syntyvä dokumentti on toiminnallinen määrittely (engl. functional specification), johon dokumentoidaan vaatimukset ja vaatimukset täyttävän järjestelmän kuvaus. (Haikala & Märijärvi, 2004, s. 78.)

Dokumentointivaiheesta käytetään usein termiä ”requirements specification” (Wiegers, 2003, s. 52; Pressman, 2000, s. 287; Sommerville, 2011, s. 94). Täten dokumentointivaihetta voidaan nimittää myös **määrittelyvaiheeksi**. Pressmanin (2000, s. 254) mukaan termi määrittely (engl. specification) tarkoittaa eri asiaa eri ihmisille. Määrittely voi olla kirjallinen dokumentti, graafinen malli, formaali matemaattinen malli, kokoelma käyttöskenaarioita, prototyyppi tai mikä tahansa näiden yhdistelmä.

Järjestelmä- ja ohjelmistovaatimukset dokumentoidaan yleensä formaaliin dokumenttiin, jota käytetään vaatimusten viestinnässä asiakkaille, järjestelmä- ja ohjelmistosuunnittelijoille järjestelmän suunnitteluprosessin johtajille. Tälle dokumentille ei ole olemassa vakio nimeä. Eri organisaatioissa tätä dokumenttia voidaan nimittää vaatimusdokumentiksi (engl. requirements document), toiminnalliseksi määrittelyksi tai järjestelmän vaatimusmäärittelyksi (engl. the system requirements specification). (Kotonya & Sommerville, 1998, s. 15.)

Kotonyan ja Sommervillen (1998, s. 15) mukaan vaatimusdokumentissa kuvataan seuraavat asiat:

1. Palvelut ja toiminnot, jotka järjestelmän tulisi tarjota.
2. Rajoitukset, joiden puitteissa järjestelmän on toimittava.
3. Järjestelmän yleiset ominaisuudet.
4. Muihin järjestelmiin kohdistuvat integrointimääritykset.
5. Tiedot järjestelmän sovellusalueesta.
6. Järjestelmän kehittämisprosessin rajoitukset.

Järjestelmissä, jotka ovat pääasiassa ohjelmistojärjestelmiä, vaatimusdokumentti voi sisältää kuvauksen laitteistosta, jolla järjestelmää on tarkoitus käyttää. Dokumentissa pitäisi aina olla johdantokappale, joka antaa yleiskuvan järjestelmästä. Lisäksi johdannossa pitäisi kertoa liiketoiminnan tarpeista, joita järjestelmän on tarkoitus tukea. Johdannossa pitäisi olla myös sanasto, jossa määritellään dokumentissa käytettävät tekniset termit. Sanasto on erityisen tärkeä, koska eri taustoista tulevat eri sidosryhmät lukevat vaatimusdokumentteja ja käyttävät niitä eri tavoilla. Sanasto on välttämätön, jotta voidaan varmistaa yhteisymmärrys vaatimusdokumentissa käytettävistä termeistä. (Kotonya & Sommerville, 1998, s. 15.) Kuviossa 1 on havainnollistava esimerkki vaatimusdokumenttien käyttäjistä ja käyttötarkoituksista.

Kuvio 1. Vaatimusdokumenttien käyttäjät (Sommerville, 2011, s. 92).

Vaatimusdokumentti on perusta kaikelle myöhemmälle suunnittelulle, koodauksella, järjestelmän testaukselle ja käyttäjädokumentaatiolle. Sen tulisi kuvata järjestelmän käyttäytymistä tarpeen vaatimalla tavalla eri olosuhteissa. Sen ei pitäisi sisältää tunnettujen suunnittelu- ja toteutusrajoitusten lisäksi muuta järjestelmän suunnittelua, rakentamista, testausta tai projektihallinnan yksityiskohtia. (Wiegers, 2003, s. 166.)

Vaatimusdokumentti voidaan jäsentää usealla eri tavalla. Dokumentin rakenne riippuu kehitettävän järjestelmän tyypistä, vaatimusten yksityiskohtaisuudesta, organisointikäytännöstä ja vaatimusten suunnitteluprosessin budjetista ja aikataulusta. Kaikkien olennaisten tietojen sisällyttämisen varmistamiseksi organisaatiot voivat itse määritellä oman standardin vaatimusdokumenteille. Standardi osoittaa, mitä dokumenttiin on sisällytettävä. (Kotonya & Sommerville, 1998, s. 15.)

Taulukko 2. Vaatimusdokumentin rakenne (Sommerville, 2011, s. 93).

Kappale	Kuvaus
Alkusanat (engl. preface)	Määritellään dokumentin odotettu lukijakunta ja kuvataan järjestelmän versiohistoria. Versiohistoria sisältää perustelut uuden version luomisella ja yhteenvedon kussakin versiossa tehdyistä muutoksista.
Johdanto	Kuvataan järjestelmän tarve. Lisäksi kuvataan lyhyesti järjestelmän toiminnot ja selitetään, miten se toimii muiden järjestelmien kanssa. Kuvataan myös, miten järjestelmä sopii organisaation yleisiin liiketoiminnallisiin tai strategisiin tavoitteisiin.
Sanasto	Määritellään dokumentissa käytettävät tekniset termit. Dokumentissa ei saa tehdä oletuksia lukijan kokemuksesta tai asiantuntemuksesta.
Käyttäjävaatimusten määrittely	Kuvataan käyttäjälle tarjotut palvelut. Ei-toiminnalliset vaatimukset kuvataan myös tässä kappaleessa. Kuvaus voi käyttää luonnollista kieltä, kaavioita tai muita notaatioita, jotka ovat ymmärrettäviä asiakkaalle. Määritellään tuote- ja prosessistandardit, joita on seurattava.
Järjestelmäarkkitehtuuri	Esitetään korkean tason katsaus odotetusta järjestelmäarkkitehtuurista, joka osoittaa toimintojen jakautumisen järjestelmämoduulien välillä. Uudelleenkäytettäviä arkkitehtonisia komponentteja korostetaan.

Järjestelmävaatimusten määrittely	Kuvataan toiminnalliset ja ei-toiminnalliset vaatimukset yksityiskohtaisemmin. Tarvittaessa ei-toiminnallisiin vaatimuksiin voidaan lisätä tietoja. Voidaan määritellä myös rajapinnat muihin järjestelmiin.
Järjestelmämallit	Sisältää graafisia järjestelmämalleja, jotka kuvaavat järjestelmän komponenttien, järjestelmän ja sen ympäristön välisiä suhteita. Esimerkkejä mahdollista malleista ovat objektimallit, tietovirtamallit ja semanttiset tietomallit.
Järjestelmän kehittyminen (engl. system evolution)	Kuvataan järjestelmään liittyvät olennaiset oletukset ja ennusteet, muun muassa laitteiston kehityksestä johtuvat odotetut muutokset ja muuttuvat käyttäjätarpeet. Tämä osio on hyödyllinen järjestelmän suunnittelijoille, koska se auttaa heitä välttämään suunnittelupäätöksiä, jotka rajoittavat todennäköisiä tulevia järjestelmämuutoksia.
Liitteet	Tarjotaan yksityiskohtaista ja täsmällistä tietoa, joka liittyy kehitettävään sovellukseen, esimerkiksi laitteisto- ja tietokantakuvauksia. Laitteistovaatimukset määrittelevät järjestelmän minimaaliset ja optimaaliset konfiguraatiot. Tietokantavaatimukset määrittelevät järjestelmän käyttämän datan loogisen järjestelyn ja datan väliset suhteet.

Hakemisto (engl. index)	Dokumenttiin voidaan sisällyttää useita dokumentteja. Normaalin aakkosellisen hakemiston lisäksi voidaan sisällyttää hakemisto muun muassa kaavioille ja funktioille.

Taulukossa 2 on esimerkki Sommervillen vaatimusdokumentin rakenteesta. Sommervillen (2011, s. 92) mukaan vaatimusdokumentin rakenne pohjautuu IEEE:n (Institute of Electrical and Electronics Engineers) vuoden 1998 standardiin. Esimerkkirakenteessa standardia on laajennettu sisällyttämällä tietoja järjestelmän ennustetusta kehityksestä. Tiedot auttavat järjestelmän ylläpitäjiä ja antavat suunnittelijoille mahdollisuuden tukea tulevia järjestelmän ominaisuuksia.

Jokaisen ohjelmistokehitysorganisaation tulisi hyödyntää yhtä tai useampaa standardin mukaista vaatimusdokumenttipohjaa. Jos organisaatiossa tehdään erityyppisiä tai kokoisia projekteja, kuten uuden suuren järjestelmän kehittämistä ja vanhojen järjestelmien pieniä parannuksia, kannattaa valita oma vaatimusdokumenttipohja jokaisella suurelle projektiluokalle. Pohjaa voi muokata projektin tarpeiden ja luonteen mukaiseksi. Jos pohjassa on osa, jota projektissa ei voida soveltaa, jätetään otsikko paikalleen ja ilmaistaan lukijalle, ettei se ole soveltuva. Tämä estää lukijaa ihmettelemästä, onko jotain tärkeää jätetty vahingossa pois. Jos projektissa jatkuvasti jätetään pohjan osioita pois, on järkevää vaihtaa pohjaa. (Wiegers, 2003, s. 171.)

Käyttäjävaatimukset tulisi kirjoittaa luonnollisella kielellä. Luonnollisen kielen tukena voidaan käyttää yksinkertaisia taulukoita, lomakkeita ja intuitiivisia kaavioita. Järjestelmävaatimukset ovat laajennettuja versioita käyttäjävaatimuksista. Ohjelmistosuunnittelijat käyttävät järjestelmävaatimuksia järjestelmän suunnittelun lähtökohtana. Ne lisäävät yksityiskohtia ja selittävät, miten järjestelmän on tarjottava käyttäjävaatimukset. Ihannetapauksessa järjestelmävaatimusten tulisi ainoastaan kuvata järjestelmän ulkoinen käyttäytyminen ja toimintarajoitukset. Järjestelmävaatimuksiin ei pitäisi sisällyttää

järjestelmän suunnittelua tai toteutusta. Käytännössä kuitenkin kaikkien suunnittelutietojen poisjättäminen on mahdotonta, koska monimutkaisen järjestelmän täydellinen määrittely vaatii riittävää yksityiskohtaisuustasoa. (Sommerville, 2011, s. 94-95.)

Taulukko 3. Järjestelmävaatimusten dokumentointitapoja (Sommerville, 2011, s. 95).

Notaatio (tapa)	Kuvaus
Luonnollisen kielen lauseet	Vaatimukset kirjoitetaan numeroiduilla lauseilla luonnollisella kielellä. Jokaisessa virkkeessä tulisi ilmaista yksi vaatimus.
Jäsennelty luonnollinen kieli	Vaatimukset on kirjoitettu luonnollisella kielellä standardiin muotoon tai mallipohjaan. Jokainen kenttä tarjoaa tietoa vaatimuksesta tietystä näkökulmasta.
Suunnittelukuvauskielet (engl. design description languages)	Notaatio käyttää ohjelmointikielen tapaista kieltä. Notaatio käyttää kuitenkin ohjelmointikieliä abstraktimpia ominaisuuksia vaatimusten määrittelyyn määrittelemällä järjestelmän toimintamallin. Tätä lähestymistapaa käytetään nykyään harvoin, vaikka se voi olla hyödyllinen käyttöliittymän määrittelyissä.
Graafiset notaatiot	Graafisia malleja, joita tuetaan tekstimerkinnöillä, käytetään järjestelmän toiminnallisten vaatimusten määrittelyyn. Usein käytetään UML-käyttötapaus- ja sekvenssikaavioita.
Matemaattiset määrittelyt	Notaatiot perustuvat matemaattisiin käsitteisiin, kuten äärellisiin automaatteihin (engl. finite-state machines) ja sarjoihin (engl. sets). Vaikka määrittelyt voivat

	vähentää vaatimusdokumentin epäselvyyttä, useimmat asiakkaat eivät ymmärrä formaalia määrittelyä. He eivät kykene tarkistamaan, edustaako määrittely heidän tarpeitaan. Tästä syystä he saattavat olla haluttomia hyväksymään tällaisia määrittelyitä järjestelmäsopimukseen.

Järjestelmävaatimukset voidaan myös kirjoittaa luonnollisella kielellä, mutta lisäksi graafisiin ja matemaattisiin malleihin sekä lomakkeisiin perustuvia notaatioita voidaan käyttää. Graafiset mallit ovat hyödyllisimpiä, kun halutaan kuvata tilan muutosta tai toimintasarjoja. UML-sekvenssi- ja tilakaaviot kuvaavat tietyn viestin tai tapahtuman seurauksena tapahtuvia toimintasarjoja. Formaaleja matemaattisia määrittelyitä käytetään joskus kuvaamaan turvallisuus- tai turvakriittisten järjestelmien vaatimuksia. Niitä harvoin käytetään muissa tilanteissa. (Sommerville, 2011, s. 95.) Taulukossa 3 on yhteenveto mahdollisista notaatioista, joita voidaan käyttää järjestelmävaatimusten kirjoittamiseen.

Vaatimusdokumentin sisällön ja dokumentointitapojen lisäksi on perehdyttävä siihen, miten yksittäisiä vaatimuksia pitäisi dokumentoida. Kotonyan ja Sommervillen (1998, s. 19) mukaan luonnollista kieltä voidaan käyttää vaatimusten selkeään kuvaamiseen, mutta silti luonnollisen kielen vaatimuksia on usein vaikea ymmärtää. Vaatimukset voivat olla monitulkintaisia, vaikeaselkoisia ja ne ymmärretään usein väärin. Seuraavat asiat ovat yleisiä ongelmia. Vaatimuksia kirjoitetaan käyttämällä monimutkaisia ehdollisia lauseita, jotka ovat hämmentäviä. Terminologiaa käytetään huolimattomalla ja epäjohdonmukaisella tavalla. Lisäksi vaatimusten kirjoittajat olettavat, että lukijalla on yksityiskohtaista tietoa aihealueesta ja järjestelmästä, jonka vuoksi he jättävät olennaista tietoa pois vaatimusdokumentista.

Nämä ongelmat tekevät vaatimusten virheiden ja puutteiden tarkastamisen vaikeaksi. Vaatimusten erilaiset tulkinnat voivat luoda erimielisyyksiä asiakkaan ja

järjestelmäsuunnittelijan välisestä sopimuksesta. Vaikka luonnollisen kielen tukena voidaan käyttää erilaisia notaatioita, hyvin kirjoitetuille luonnollisen kielen vaatimuksille on aina tarvetta. Järjestelmämallit (engl. system models) ovat usein epäsoveltuvia oleellisen tiedon viestimiseen henkilöille, joiden ei tarvitse analysoida vaatimuksia yksityiskohtaisesti (Kotonya & Sommerville, 1998, s. 20.)

Kotonyan ja Sommervillen (1998, s. 20) mukaan riippumatta vaatimuskuvauksen yksityiskohtaisuudesta, on olemassa kolme välttämätöntä asiaa, jotka on otettava huomioon vaatimuksia dokumentoitaessa. (1) Vaatimuksia luetaan useammin kuin niitä kirjoitetaan. Panostaminen helppolukuisten ja ymmärrettävien vaatimusten kirjoittamiseen on lähes aina kustannustehokasta. (2) Vaatimusten lukijat tulevat eri taustoista. Vaatimusten käsittelijä ei saa olettaa, että lukijoilla olisi samanlainen tausta ja ymmärrys aiheesta kuin hänellä itsellään. (3) Selkeästi ja tiiviisti kirjoittaminen ei ole helppoa. Vaatimuskuvausten laatimiseen, tarkastamiseen ja parantamiseen on varattava riittävästi aikaa.

Taulukko 4. Yksittäisten vaatimusten ominaisuudet (ISO/IEC/IEEE 29148, 2011, s. 11).

Ominaisuus	Kuvaus
Tarpeellisuus	Vaatimus määrittää olennaisen kyvykkyyden, ominaispiirteen, rajoituksen ja/tai laatutekijän. Jos vaatimus poistetaan, järjestelmä jää puutteelliseksi. Puutteellisuus ei ole korvattavissa muilla ominaisuuksilla. Vaatimus on tällä hetkellä sovellettavissa eikä se ole vanhentunut. Vaatimukset, joissa on suunniteltu voimassaolopäivä, on yksilöyty selvästi.
Toteutuksen riippumattomuus	Vaatimus välttää tarpeettomia rajoituksia arkkitehtisuunnittelussa. Tavoitteena on toteutuksen riippumattomuus.

	Vaatimuksessa todetaan, mitä tarvitaan, ei sitä, kuinka vaatimus tyydytetään.
Yksiselitteisyys	Vaatimus esitetään siten, että se voidaan tulkita ainoastaan yhdellä tavalla. Vaatimus esitetään yksinkertaisesti ja se on helppo ymmärtää.
Johdonmukaisuus	Vaatimus ei ole ristiriidassa muiden vaatimusten kanssa.
Täydellisyys	Vaatimusta ei tarvitse tarkentaa enempää, koska se on sellaisenaan mitattavissa ja vastaa sidosryhmän asettamia tarpeita.
Ainutlaatuisuus	Vaatimus sisältää ainoastaan yhden itsenäisen vaatimuksen. Se ei ole päällekkäinen muiden vaatimusten kanssa.
Toteutettavuus	Vaatimus on teknisesti toteutettavissa eikä se vaadi merkittävää teknologista kehitystä. Vaatimus on toteutettavissa järjestelmälle asetettujen rajoitteiden puitteissa.
Jäljitettävyys	Vaatimus on jäljitettävissä eteen- ja taaksepäin. Vaatimuksen lähde ja toteutustapa voidaan jäljittää.
Todennettavuus	Vaatimus ilmenee järjestelmässä siten, että sen täyttyminen voidaan todentaa. Todennettavuus paranee, kun vaatimus on mitattavissa.

Taulukossa 4 esitetään ISO/IEC/IEEE 29148 standardin (2011, s. 11) mukaiset ominaisuudet, jotka jokaisen vaatimuksen tulisi täyttää. Haikalan ja Mikkosen (2011, s. 64) mukaan muita vaatimuksista dokumentoitavia asioita ovat muun muassa luontipäivämäärä,

vaatimuksen kirjaaja, vaatimuksen alkuperä, vaatimuksen tyyppi ja kuvaus, suhde muihin vaatimuksiin, muutosherkkyys, testattavuus ja työmääräarvio.

3.1.4 Validointi

Vaatimusten validointi on prosessi, jolla tarkistetaan, että vaatimukset todella määrittelevät järjestelmän asiakkaiden haluamalla tavalla. Vaatimusten validointi on tärkeää, koska virheet vaatimusdokumentissa voivat johtaa suuriin korjauskustannuksiin, jos nämä ongelmat havaitaan järjestelmän kehittämisen aikana tai sen jälkeen, kun järjestelmä on jo käytössä. Järjestelmämuutosta vaativan vaatimusongelman kustannukset ovat paljon suuremmat kuin suunnittelu- tai koodausvirheiden korjaamisesta aiheutuvat kustannukset. Syynä tälle on, että vaatimusten muutos tarkoittaa yleensä sitä, että myös järjestelmän muotoilua (engl. design) ja toteutustapaa on muutettava. Lisäksi järjestelmä on testattava uudelleen. (Sommerville, 2011, s. 110.)

Validoinnista käytetään myös nimitystä tosittaminen. Validointi on oikean asian tekemisen varmistamista. Vaatimusten validoinnissa arvioidaan ja päätetään, ovatko asetetut vaatimukset oikeita ja tarpeellisia. Vaatimukset laaditaan usein kehitysprosessin alussa vähäisillä tiedoilla siitä, millaiseen ratkaisuun asetut vaatimukset johtavat. Ratkaisun hahmottuessa on syytä tarkastella ensin, vastaako ratkaisu asetettuja vaatimuksia eli onko asia tehty oikein. Tämän jälkeen tarkastellaan, vastaavatko asetetut vaatimukset tarvetta eli onko tehty oikeaa asiaa. (Kosola, 2013, s. 54.)

Vaatimusten validoinnissa tarkastellaan vaatimusdokumenttia varmistaakseen, että kaikki järjestelmävaatimukset on ilmaistu yksiselitteisesti, epäjohdonmukaisuudet, puutteet ja virheet on havaittu ja korjattu. Lisäksi työn tulosten on oltava prosessille, projektille ja tuotteelle asetettujen standardien mukainen. (Pressman, 2000, s. 255.) Validointi varmistaa, että vaatimukset ovat oikeita, osoittavat halutut laatuominaisuudet ja tyydyttävät asiakkaiden tarpeet (Wiegers, 2003, s. 53).

Validointiprosessissa ovat mukana järjestelmän sidosryhmät, vaatimusten käsittelijät ja järjestelmän suunnittelijat, jotka analysoivat vaatimuksia ongelmien, puutteiden ja epäselvyyksien varalta (Kotonya & Sommerville, 1998, s. 87). Vaatimusten validointitekniikoita ovat muun muassa **vaatimusten tarkastelut** (engl. requirements reviews), **prototyypit** ja **testitapausten luominen** (engl. test-case generation). Vaatimusten tarkastelussa vaatimuksia analysoidaan systemaattisesti arviointiryhmän toimesta. Arviointiryhmä tarkistaa virheitä ja epäjohdonmukaisuuksia. Jos validoinnin lähestymistapana on prototyyppi, loppukäyttäjille ja asiakkaille esitetään järjestelmästä luotu malli. Mallin avulla voidaan kokeilla, vastaako järjestelmä heidän todellisia tarpeitaan. Testitapausten luomisen avulla vaatimuksista pystytään paljastamaan ongelmia. Jos testin suunnittelu on vaikeaa tai mahdotonta, se tarkoittaa yleensä, että vaatimuksia on vaikea toteuttaa. Testien luominen käyttäjävaatimuksista ennen koodin kirjoittamista on olennainen osa XP:tä (extreme programming). (Sommerville, 2011, s. 111.)

Vaatimusten validointi on analyysin kanssa päällekkäistä, koska molemmat liittyvät ongelmien löytämiseen vaatimuksista (Sommerville, 2011, s. 110). Näiden aktiviteettien välillä on kuitenkin merkittäviä eroja. Vaatimusten analysointi koskee käsittelemättömiä vaatimuksia, jotka on saatu järjestelmän sidosryhmiltä. Vaatimukset ovat yleensä epätäydellisiä ja ne ilmaistaan informaalisti ja jäsentämättä. Vaatimusten analysoinnin aikana tulisi keskittyä varmistamaan, että vaatimukset vastaavat sidosryhmän tarpeita. Analysoinnissa ei keskitytä vaatimuskuvausten yksityiskohtiin. Vaatimusten analysoinnissa tulisi pääasiassa vastata kysymykseen ”Onko meillä oikeat vaatimukset?” (Kotonya & Sommerville, 1998, s. 88.)

Vaatimusten validointi liittyy vaatimusdokumentin lopullisen luonnoksen tarkistamiseen. Lopullinen luonnos sisältää kaikki järjestelmävaatimukset. Lisäksi puutteellisuudet ja epäjohdonmukaisuudet on poistettu. Dokumentin ja vaatimusten tulisi noudattaa laatustandardeja. Sidosryhmien tarpeiden huomioimisen lisäksi validointiprosessissa pitäisi keskittyä enemmän tapaan, jolla vaatimuksia kuvataan. Vaatimusten validoinnissa tulisi

pääasiassa vastata kysymykseen ”Onko vaatimukset ymmärretty oikein?” (Kotonya & Sommerville, 1998, s. 88.)

Analyysin tulisi keskittyä sidosryhmien tarpeisiin ennen kuin yksityiskohtaista vaatimusdokumenttia laaditaan. Ihannetilanteessa vaatimusdokumentin tulisi sisältää ainoastaan vaatimuksia, jotka ovat hyväksyttäviä sidosryhmille. Väistämättä kuitenkin joitain sidosryhmiin liittyviä ongelmia löydetään vasta vaatimusten validoinnin yhteydessä. Nämä ongelmat on korjattava ennen kuin vaatimusdokumentti hyväksytään ja sitä käytetään järjestelmän kehittämisen perustana. Joissain tapauksissa ongelmat ovat ainoastaan dokumentointiongelmia, jotka ovat korjattavissa parantamalla vaatimuskuvausta. Joissain tapauksissa ongelmat kuitenkin johtuvat puutteista vaatimusten kartoittamisessa, analysoinnissa ja mallinnuksessa. Tällaisissa tilanteissa voidaan joutua palaamaan näihin aikaisempiin vaiheisiin. (Kotonya & Sommerville, 1998, s. 89.)

3.2 Vaatimustenhallinta

Vaatimustenhallinnan keskeisin prosessi on **vaatimusmuutosten hallinta**. Vaatimusten muutosprosessi vaiheistukseen kuuluu muun muassa tyypillisesti muutospyynnön tekeminen, analysointi, testaus ja hyväksyminen. Vaatimuksia käsitellään muutosprosessissa samaan tapaan kuin vaatimusmäärittelyssäkin. Muutospyynnöt hyväksytään projektin johtoryhmissä. (Haikala & Mikkonen, 2011, s. 67.)

Suurten ohjelmistojärjestelmien vaatimukset muuttuvat jatkuvasti. Yksi syy tähän on, että näitä järjestelmiä kehitetään yleensä vastaamaan sellaisiin ongelmiin, joita ei voida täysin ennakkoon määritellä. Ohjelmistoprosessin aikana sidosryhmien käsitys järjestelmästä muuttuu jatkuvasti, joten järjestelmävaatimusten on myös mukauduttava muutoksiin. Kun uusi järjestelmä on otettu käyttöön ja sitä käytetään säännöllisesti, uusia vaatimuksia ilmenee väistämättä. Käyttäjien ja järjestelmäasiakkaiden on vaikea ennustaa, mitä vaikutuksia uudella järjestelmällä on heidän liiketoimintaprosesseihinsa ja työnteon tapaan. Kun loppukäyttäjät saavat kokemusta järjestelmästä, he löytävät uusia

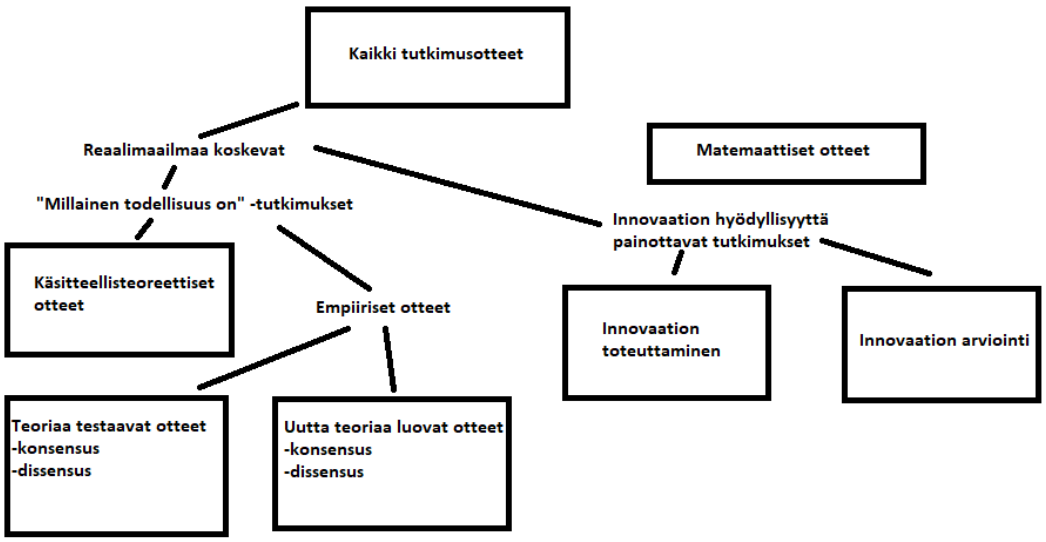
tarpeita ja prioriteetteja. Vaatimustenhallinta on prosessi, jonka tarkoituksena on ymmärtää ja hallita järjestelmävaatimusten muutoksia. (Sommerville, 2011, s. 112.)

Asiakasvaatimuksista johdettavat ohjelmistovaatimukset selviävät pääasiassa esitutkimus- ja määrittelyvaiheissa. On kuitenkin tavallista, että koko projektin ajan niihin tulee muutoksia. Muutoksia tulee viimeistään ylläpidon ja ohjelmistosta tehtävien uusien versioiden yhteydessä. Muutoksen syitä ovat muun muassa seuraavanlaiset asiat. Projektin alkuvaiheessa kaikkia asiakasvaatimuksia ei välttämättä ymmärretä oikein ja osa asiakasvaatimuksista voi jäädä huomaamatta. Projektin aikana ohjelmiston toimintaympäristössä voi tapahtua muutoksia, esimerkiksi muutokset toimintatavoissa, laitteistossa tai ohjelmistossa. On myös mahdollista, että jonkin asiakasvaatimuksen täyttämiseksi määritelty ominaisuus osoittautuu käyttökelvottomaksi. Joitain ominaisuuksia voidaan jättää aluksi toteuttamatta aikataulupaineiden vuoksi. Lisäksi markkinatilanteen muuttumiseen voidaan reagoida muuttamalla ja lisäämällä vaatimuksia. (Haikala & Märijärvi, 2004, s. 98-99.)

Projektissa on oltava sovitut menettelyt muutostenhallintaa varten. Projektissa on sovittava esimerkiksi, miten asiakasvaatimuksiin hyväksytään muutoksia. Muutoksissa ongelmana on, että yhteen vaihetuotteeseen tehty muutos vaikuttaa usein myös muihin vaihetuotteisiin. Projektissa on selvitettävä, miten ja minne. Muutostenhallinnan kannalta tärkeää on jäljitettävyys sekä eteenpäin että taaksepäin. Eteenpäin jäljitettävyyden avulla pystytään päättelemään, miten muutokset vaikuttavat vaihetuotteisiin vaatimusdokumentista aina ohjelmakoodiin saakka. Toisin sanoen voidaan siis päätellä, mitä muutoksia jonkin asiakasvaatimuksen muuttaminen tai poistaminen aiheuttaa toteutuksessa. Taaksepäin jäljitettävyyden avulla voidaan tutkia, mitkä asiakasvaatimukset jäävät täyttämättä, jos aikataulupaineiden vuoksi jokin osa ohjelmakoodista jätetään toteuttamatta. (Haikala & Märijärvi, 2004, s. 99.)

4 Suunnittelutieteellinen tutkimus

Järvinen ja Järvinen (2011, s. 9) jakavat kaikki tutkimusotteet kahteen luokkaan sen mukaisesti, tutkitaanko reaalimaailmaa vai symbolijärjestelmiä. Symbolijärjestelmillä ei ole reaalitodellisuudessa vastinetta ja siihen luokitellaan matemaattiset tutkimusotteet. Reaalimaailman tutkimusotteet jaetaan kahteen luokkaan riippuen siitä, painotetaanko tutkimuksessa innovaation eli ihmisen tavoitteleman tai rakentaman luomuksen hyödyllisyyttä vai sitä, millainen reaalitodellisuus on. Reaalimaailmaa koskevista tutkimuksista erotetaan käsitteellisteoreettisen tutkimuksen otteet empiirisen tutkimuksen otteista. Empiirisen tutkimuksen otteet jaetaan uutta teoriaa luoviin ja teoriaa testaaviin tutkimusotteisiin. Sekä teoriaa testaavien että uutta teoriaa luovien tutkimusotteiden taustalla on tärkeä ennakko-oletus, oletetaanko tutkimuskohteissa vallitsevan yksi- vai erimielisyys (konsensus, dissensus). Innovaation hyödyllisyyttä painottavat tutkimusotteet jaetaan innovaation toteuttamiseen ja arviointiin. Normaalisti tulosten hyvyyttä arvioidaan myöhemmin.



Kuvio 2. Tutkimusmetodien taksonomia (Järvinen & Järvinen, 2011, s. 10).

Tutkimussuoritteet voidaan jakaa neljään luokkaan. Nämä neljä luokkaa ovat käsitteistö (engl. constructs), malli, metodi ja toteutus. Käsitteistö muodostaa tutkimusaiheen

sanaston. Malli on joukko lauseita tai propositioita, jotka ilmaisevat käsitteiden väliset suhteet. Metodi on joukko tehtävän suorittamiseen käytettäviä askelia, esimerkiksi algoritmi tai ohjeisto. Toteutuksella tarkoitetaan artefaktin toteutusta ympäristössään. Artefaktin toteutukset operationalisoivat käsitteistöjä, metodeja ja malleja. Tutkimus voi olla suunnittelutieteellistä tai luonnontieteellistä. Suunnittelutiede koostuu kahdesta aktiviteetista, rakentamisesta ja arvioinnista. Luonnontiede jaetaan teorian luomiseen ja teorian testaamiseen. (March & Smith, 1995.) Marchin ja Smithin viitekehyksessä luonnontiede edustaa melkein kaikkia muita tieteitä kattavasti. Se kattaa myös käyttäytymis- ja yhteiskuntatieteet (Järvinen, 2006).

		Tutkimuksen toiminnot			
		Suunnittelutiede		Luonnontiede	
		Rakentaa	Arvioida	Luoda teoriaa	Testata teoriaa
Tutkimussuoritteet	Käsitteistö				
	Malli				
	Metodi				
	Toteutus				

Kuvio 3. Tutkimuksen viitekehys (March & Smith, 1995; Järvinen & Järvinen, 2011, s. 11).

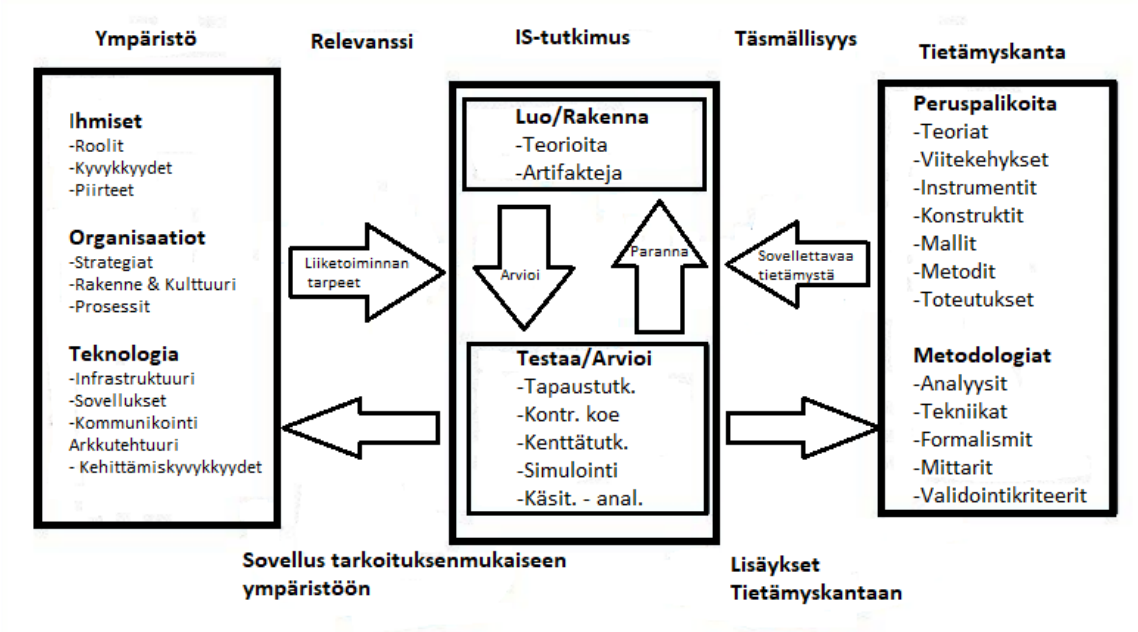
March ja Smith käyttävät termiä artefakti viitatessaan tekniseen innovaatioon. Artefaktista voidaan käyttää myös termiä innovaatio, jotta sen alaan voidaan sisällyttää myös sosiaaliset ja tiedolliset sekä teoreettiset innovaatiot ja näiden yhdistelmät. Hyväksymällä sosiaaliset, tekniset ja tiedolliset innovaatiot hyödynnetään fyysisten, inhimillisten ja tietoresurssien sekä niiden yhdistelmien innovaatioita. (Järvinen & Järvinen, 2011, s. 11.)

Suunnittelutieteessä luodaan artefakteja tavoitteiden saavuttamiseksi. Suunnittelutiede yrittää luoda asioita, jotka palvelevat inhimillisiä tarkoituksia. Suunnittelutiede on teknologiaorientoitunutta. Suunnittelutieteen tulokset ovat käsitteistöjä, malleja, metodeja ja toteutuksia. Suunnittelutiede koostuu kahdesta aktiviteetista, rakentamisesta ja arvioinnista. Rakentamisessa luodaan artefakti tiettyyn tarkoitukseen. Arvioinnissa määritetään, miten hyvin artefakti täyttää sille asetetut vaatimukset. Arviointikriteerejä ovat muun muassa artefaktin arvo ja hyödyllisyys. Suunnittelutiede ja luonnontiede eroavat

toisistaan, koska niillä on eri tutkimustavoitteet. Luonnontiede pyrkii selittämään ja ymmärtämään ilmiöitä, kun taas suunnittelutieteellisessä tutkimuksessa kehitetään keinoja inhimillisten tavoitteiden saavuttamiseksi. (March & Smith, 1995.)

Suunnittelutieteessä merkittäviä ongelmia syntyy siitä, että artefakti sijoitetaan aina sille tarkoitettuun ympäristöön. Ympäristön epätäydellinen ymmärtäminen voi johtaa epätarkoituksenmukaisesti suunniteltuihin artefakteihin ja epätoivottuihin sivuvaikutuksiin. Artefaktin arvioinnissa ongelmana on, että se perustuu ennakoituihin käyttötarkoituksiin. Arviointi vaatii mittavälineiden ja niillä mitattavien kriteerien kehittämistä. (March & Smith, 1995.)

Hevner, March, Park ja Ram (2004) ovat laatineet oman informaatiosysteemien tutkimuksen viitekehyksen täydentämään Marchin ja Smithin (1995) tutkimuksen viitekehystä. Hevnerin ja muiden viitekehys on esitetty kuviossa 4.



Kuvio 4. Informaatiosysteemien (IS) tutkimuksen viitekehys (Hevner ja muut, 2004; Järvinen & Järvinen, 2011, s. 12).

Ympäristö, joka on kuvion vasemmalla puolella, määrittelee ongelma-alueen (Järvinen & Järvinen, 2011, s. 12). Hevnerin ja muiden (2004) mukaan ympäristö koostuu ihmisistä, liiketoimintaorganisaatioista ja niiden suunnitelluista tai olemassa olevista teknologioista. Ympäristö sisältää tehtävät, tavoitteet, ongelmat ja mahdollisuudet, jotka määrittelevät liiketoiminnan tarpeet organisaatioissa toimivien ihmisten näkemysten mukaisesti. Liiketoiminnan tarpeiden näkemyksiin vaikuttavat ihmisten kyvykkyydet, roolit ja piirteet. Niitä arvioidaan suhteessa rakenteeseen, strategioihin, kulttuuriin ja nykyisiin liiketoimintaprosesseihin. Liiketoiminnan tarpeita asemoidaan sovelluksiin, nykyiseen teknologiseen infrastruktuuriin, kommunikointiarkkitehtuureihin ja kehittämismahdollisuuksiin. (Hevner ja muut, 2004; Järvinen & Järvinen, 2011, s. 12-13.)

Kuvion keskiosassa esitetään kaksi mahdollisuutta. Ensimmäinen mahdollisuus on suorittaa liiketoiminnan tarpeisiin liittyen käyttäytymistieteellinen teoriaa luova tai teoriaa testaava selittävä tai ennustettava tutkimus. Toinen mahdollisuus on suorittaa suunnittelutieteellinen tutkimus, jossa liiketoiminnan tunnistettuihin tarpeisiin rakennetaan ja arvioidaan artefakti. Käyttäytymistieteellisessä tutkimuksessa tavoitellaan totuutta, suunnittelutieteellisessä tutkimuksessa hyötyä. Molemmat tutkimustyypit tarvitsevat toisiaan. Käyttäytymistieteellinen tutkimus tuottaa totuudellista tietoa artefaktien toiminnasta ja rakenteesta. Lisäksi artefaktin rakentamisen tai arvioinnin yhteydessä käyttäytymistieteellisestä teoriasta voi löytyä heikkouksia. (Hevner ja muut, 2004; Järvinen & Järvinen, 2011, s. 13.)

Kuvion oikealla puolella on tietämyskanta, joka koostuu peruspalikoista ja metodologioista. Peruspalikat koostuvat teorioista, viitekehyksistä, instrumenteista, käsitteistöistä, malleista, metodeista ja toteutuksista. Peruspalikoita käytetään teorian luontivaiheessa ja artefaktin rakentamisvaiheessa. Metodologiat tarjoavat ohjeita, joita käytetään artefaktin arviointiin ja teorian testaamiseen. (Hevner ja muut, 2004; Järvinen & Järvinen, 2011, s. 13.)

Käyttäytymis- ja suunnittelutieteellisiä tutkimuksia sovellettaessa liiketoimintatarpeisiin asianmukaisessa ympäristössä ja niiden lisätessä tietämyskannan sisältöä palvelemaan käytäntöön soveltamista ja jatkotutkimusta, tutkimusten tuloksia arvioidaan. Jos tutkimus tuottaa selvästi lisää joko tietämyskannan perustietämykseen tai metodologioihin, poikkeaa suunnittelututkimus rutiinitutkimuksesta. (Hevner ja muut, 2004; Järvinen & Järvinen, 2011, s. 13.)

Suunnittelutieteellisessä tutkimuksessa tarkoituksena on pysyvän muutoksen aikaansaaminen systeemissä, alkutilasta haluttuun lopputilaan. Suunnittelutieteellinen tutkimus voi pyrkiä vastaamaan esimerkiksi seuraavanlaisiin kysymyksiin. Onko tietyn innovaation rakentaminen mahdollista ja kuinka hyödyllinen se on? Millainen tietyn innovaation pitäisi olla ja kuinka se tulisi rakentaa? Esimerkiksi miten käyttöliittymää voidaan parantaa, jotta käyttäjän virheet vähenisivät? (Järvinen & Järvinen, 2011, s. 103.) Suunnittelutieteessä tarkoituksena on uuden tietämyksen luominen suunnittelua ja toteutusta varten tai nykyisten systeemien suorituskyvyn parantaminen. Toisin sanoen tarkoituksena on joko konstruktio-ongelmien tai parantamisongelmien ratkaiseminen. Uuden innovaation hyödyllisuus on arvioitava jossain vaiheessa. (Van Aken, 2004; Järvinen & Järvinen, 2011, s. 103.)

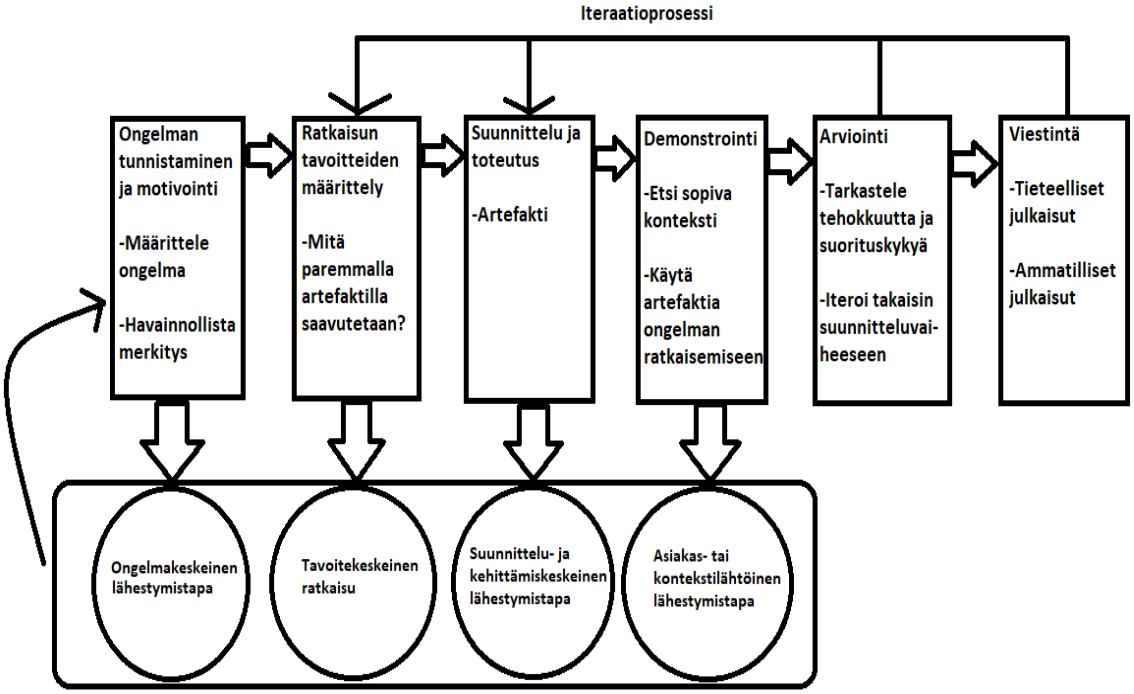
Uusi systeemi ei ole suunnittelutieteen tulos, vaan suunnittelutieteen tavoitteena on uuden suunnittelutietämyksen tuottaminen, siis tietämyksen, jota ammattilaiset voivat käyttää tulevien konstruointi- ja suunnitteluongelmien ratkaisemisessa. Suunnittelutietämys sisältää kolmea eri suunnittelua. Suunnittelu voi olla kohteen, toteutuksen ja prosessin suunnittelua. Kohteella viitataan artefaktiin tai interventioon, toteutuksella suunnitelman laatimiseen artefaktin tai intervention toteuttamiseksi ja prosessilla toista ilmaisua käyttäen metodin kehittämistä suunnitteluongelman ratkaisemiseksi tai ammattilaisen oman suunnitelman laatimista ongelman ratkaisemiseksi. Vaikka alan ammattilaisten ongelmat ovat aina määrättyjä ja ainutlaatuisia, suunnittelutietämys on yleistä. Suunnittelutiedettä voidaan käyttää tapausten joukkoon. Yleistietämystä on aina sovellettava tapauskohtaisesti. (Van Aken, 2004; Järvinen & Järvinen, 2011, s. 104.)

4.1 DSRM-prosessimalli

Metodologialla tarkoitetaan tietyllä tieteenalalla sovellettavia periaatteita, käytäntöjä ja menettelytapoja. Metodologia voi auttaa tietojärjestelmätieteen tutkijoita tuottamaan ja esittämään korkealaatuisia suunnittelutieteellisiä tutkimuksia, jotka voidaan hyväksyä arvokkaiksi, täsmällisiksi ja julkaistaviksi. Suunnittelutieteellisen tutkimuksen metodologia sisältää kolme elementtiä: käsitteelliset periaatteet suunnittelutieteellisen tutkimuksen määrittelemiseksi, käytännön säännöt ja prosessi tutkimuksen toteuttamiseksi ja esittämiseksi. (Peffers, Tuunanen, Rothenberger & Chatterjee, 2007, s. 49.)

Aikaisemmassa tutkimuksessa on esitelty periaatteita, jotka määrittelevät, mitä suunnittelutieteellinen tutkimus on ja mihin tavoitteisiin sen tulisi pyrkiä, sekä käytännön sääntöjä, jotka antavat opastusta tutkimuksen toteuttamiseen ja perustelemiseen. Siitä huolimatta periaatteet ja käytännön säännöt ovat vain kaksi metodologian kolmesta ominaisuudesta. Puuttuva osa on menettelytapa, joka tarjoaa yleisesti hyväksytyn prosessin tutkimuksen suorittamiseksi. (Peffers ja muut, 2007, s. 50.)

Peffers ja muut ovat laatineet suunnittelututkimuksen metodologian, DSRM-prosessimallin (DSRM – Design Science Reseach Methodology), miten suunnittelutieteellistä tutkimusta tulee tehdä. Metodologia on jaettu kuuteen vaiheeseen: 1) ongelman tunnistaminen ja motivointi, 2) ratkaisun tavoitteiden määrittely, 3) suunnittelu ja toteutus, 4) demonstrointi, 5) arviointi ja 6) viestintä. DSRM-prosessimalli on synteesi, joka on luotu seitsemän suunnittelututkimusta yleisesti käsittelevän paperin pohjalta. (Peffers ja muut, 2007; Järvinen & Järvinen, 2011, s. 111.) Yhdessä aikaisemman tutkimuksen kanssa DSRM-prosessimalli tarjoaa kokonaisen ja valmiin metodologian suunnittelutieteelliselle tutkimukselle (Peffers ja muut, 2007, s. 50). Seuraavissa kappaleissa esitellään DSRM-prosessimallin vaiheet.



Kuvio 5. DSRM-prosessimalli (Peffers ja muut, 2007, s. 54).

Aktiviteetti 1**: Ongelman tunnistaminen ja motivointi**. Erityinen tutkimusongelma on määriteltävä ja ratkaisun arvo on pystyttävä osoittamaan. Koska ongelman määrittelyä käytetään myöhemmin vaikuttavasti ratkaisua tarjoavan artefaktin toteuttamisessa, on ongelman pilkkominen pieniin osiin hyödyllistä. Ratkaisun arvon osoittaminen saa aikaan kaksi asiaa. Ensiksikin se motivoi tutkijaa ja tutkimuksen yleisöä etsimään ratkaisua ja hyväksymään tulokset. Toiseksi se auttaa ymmärtämään ongelman tutkijan näkökulmasta. Tätä aktiviteettia varten tarvittavat resurssit sisältävät tietämystä ongelman tilasta ja sen ratkaisun merkityksestä. (Peffers ja muut, 2007, s. 54-55.)

Aktiviteetti 2: **Ratkaisun tavoitteiden määrittely**. Ratkaisun tavoitteet on johdettava ongelman määrittelystä sekä siitä, mikä on mahdollista ja toteutettavissa. Tavoitteet voivat olla kvantitatiivisia tai kvalitatiivisia. Kvantitatiiviset tavoitteet kuvaavat, miten haluttu ratkaisu on parempi kuin nykyinen ratkaisu. Kvalitatiiviset tavoitteet kuvaavat, miten uusi artefakti auttaa tähän saakka tuntemattomien ongelmien ratkaisussa. Tätä aktiviteettia varten tarvitaan tietämystä ongelmien tilasta ja nykyisistä ratkaisuista, jos sellaisia on, sekä niiden tehokkuudesta. (Peffers ja muut, 2007, s. 55.)

Aktiviteetti 3: **Suunnittelu ja toteutus**. Tässä aktiviteetissa luodaan varsinainen artefakti. Suunnittelututkimuksen artefakti voi käsitteellisesti olla mikä tahansa suunniteltu objekti, johon on sisällytetty suunnittelussa tutkimuskontribuutio. Tämä aktiviteetti sisältää artefaktin halutun toiminnallisuuden ja sen arkkitehtuurin määrittämisen ja tämän jälkeen varsinaisen artefaktin luomisen. Resurssit, joita tarvitaan tavoitteiden määrittelystä suunnitteluun ja toteutukseen siirryttäessä sisältävät tietämystä teoriasta, jota voidaan käyttää ratkaisun aikaansaamiseksi. (Peffers ja muut, 2007, s. 55.)

Aktiviteetti 4: **Demonstrointi**. Artefaktin käyttöä on demonstroitava ratkaisemalla yksi tai useampi esimerkkiongelma. Demonstrointi voi olla artefaktin käyttöä simuloinnissa, kokeilussa, testissä, tapaustutkimuksessa tai muussa sopivassa toiminnassa. Demonstrointiin tarvittavat resurssit sisältävät tietämystä siitä, miten artefaktia käytetään ongelman ratkaisemiseksi. (Peffers ja muut, 2007, s. 55.)

Aktiviteetti 5: **Arviointi**. Tässä aktiviteetissa havainnoidaan ja mitataan, miten hyvin artefakti tukee ongelman ratkaisua. Tämä aktiviteetti sisältää toisessa aktiviteetissa määriteltyjen ratkaisun tavoitteiden vertailua demonstroinnissa artefaktin käytöstä havainnoituihin todellisiin tuloksiin. Vertailu vaatii relevanttien mittareiden ja analysointitekniikoiden käyttämistä. Arviointi voidaan suorittaa monin eri tavoin ongelmakentän luonteesta ja artefaktista riippuen. Arvioinnissa voidaan hyödyntää määriteltyjen toiminnallisuuksien vertailua, budjetoituja suureita, tyytyväisyysselvityksiä, asiakaspalautetta tai simulointeja. Se voi sisältää myös suorituskyvyn mittaamista esimerkiksi vasteaikana tai saatavuutena. Käsitteellisesti arvioinnissa voidaan hyödyntää mitä tahansa asianmukaista empiiristä näyttöä tai loogista todistetta. Arvioinnin jälkeen tutkijat voivat päättää, palataanko kolmanteen aktiviteettiin parantamaan artefaktin vaikuttavuutta, vai jatketaanko seuraavaan aktiviteettiin jättäen lisäparannukset seuraaville hankkeille. Tutkimuskentän luonne vai määrätä, onko tällainen iterointi mahdollista vai ei. (Peffers ja muut, 2007, s. 56.)

Aktiviteetti 6: **Viestintä**. Tässä aktiviteetissa viestitään ongelmasta ja sen tärkeydestä, artefaktista, sen hyödyllisyydestä ja uutuudesta, suunnittelun täsmällisyydestä ja sen vaikuttavuudesta tutkijoilla ja muille relevanteille yleisöille, kuten käytännön ammattilaisille. Tieteellisissä julkaisuissa tulee noudattaa DSRM-prosessin mukaista rakennetta. (Peffers ja muut, 2007, s. 56.)

4.2 Artefaktin arviointikriteereistä

Hevner ja muut (2004) ovat esitelleet seitsemän ohjetta suunnittelutieteellisen tutkimuksen tekemiseksi. Ohjeet esitellään taulukossa 5. Ohjeet auttavat arvioijia, tutkijoita, lehtien päätoimittajia ja lukijoita **arvioimaan** tutkimuksia (Järvinen & Järvinen, 2011, s. 115).

Taulukko 5. Suunnittelutieteellisen tutkimuksen ohjeet (Hevner ja muut, 2004, s. 83).

Ohje	Kuvaus
1. Suunnittele artefakti	Suunnittelutieteellisessä tutkimuksessa on tuotettava toteuttamiskelpoinen artefakti. Artefakti voi olla käsitteistö (engl. construct), malli, metodi tai toteutus.
2. Ongelman merkityksellisyys	Suunnittelutieteellisessä tutkimuksessa tavoitteena on kehittää teknologiapohjaisia ratkaisuja tärkeisiin ja merkityksellisiin liiketoimintaongelmiin.
3. Evaluointi	Artefaktin hyödyllisuus, laatu ja tehokkuus on täsmällisesti havainnollistettava hyvin toteutetuilla arviointimenetelmillä.
4. Tutkimuksen kontribuutiot	Tehokkaan suunnittelutieteellisen tutkimuksen tulee tuottaa selkeää ja todennettavissa olevaa kontribuutiota joko

	suunnitellun artefaktin, perustietämyksen ja/tai metodologioiden alueilla.
5. Tutkimuksen tarkkuus	Suunnittelutieteellinen tutkimus perustuu tarkkojen menetelmien soveltamiseen artefaktin rakentamisessa ja arvioinnissa.
6. Suunnittelu etsintäprosessina	Tehokkaan artefaktin toteutus vaatii saatavilla olevien keinojen hyödyntämistä tavoitteiden saavuttamiseksi. Samalla on kuitenkin noudatettava ongelmaympäristössä vallitsevia lakeja.
7. Tutkimuksesta viestiminen	Suunnittelutieteellisen tutkimuksen tulokset on tehokkaasti esitettävä sekä teknologia- että johtamiskeskeisille yleisöille.

Käytännössä hienon innovaation toteuttaminen on mahdollista. Kuitenkin tieteellisen meriitin saamiseksi innovaation rakentamisprosessi on kuvattava yksityiskohtaisesti ja valinnat ja päätökset on perusteltava. Lisäksi ratkaisun alkuperäisyys ja paremmuus muihin tunnettuihin nähden on osoitettava. (Järvinen & Järvinen, 2011, s. 115.)

5 Artefaktin toteutus

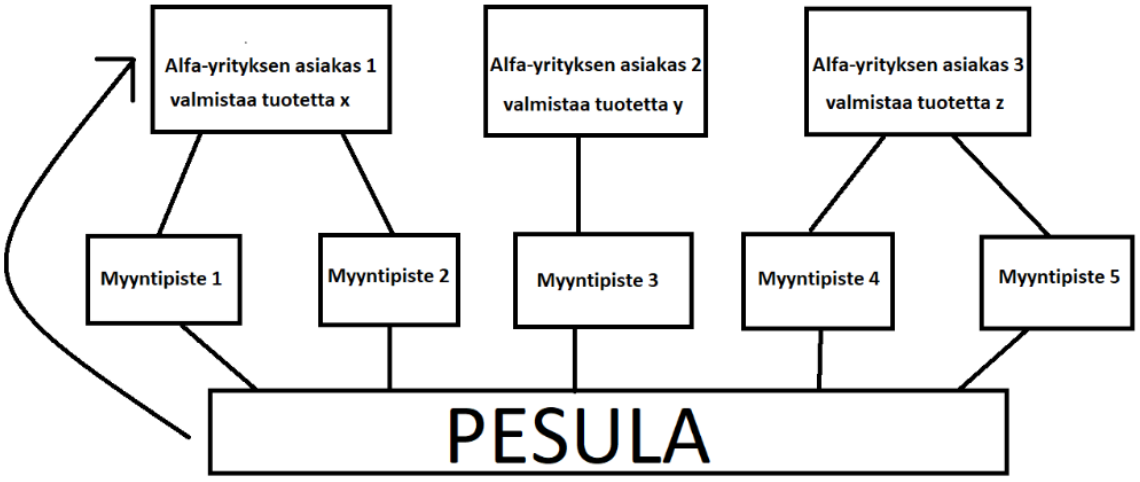
Tässä kappaleessa kuvataan tutkimuksen eli tässä tapauksessa kehitettävän artefaktin toteutusprosessi. Artefakti kehitetään seuraamalla suunnittelutieteellisen tutkimuksen DSRM-prosessimallia aktiviteettien 1-6 mukaisesti. Ennen artefaktin toteutusprosessin kuvaamista, kappaleessa esitellään artefaktin ympäristö ja tietopohja. Kappaleessa 5.6, artefaktin demonstrointi alfa-yrityksessä, artefaktia käytetään ohjeistuksena alfa-yrityksen oman laatikkojärjestelmän vaatimusmäärittelyyn.

5.1 Artefaktin ympäristö

Artefaktin ympäristö koostuu organisaatioista, jotka joko omistavat logistisen tietojärjestelmän tai suunnittelevat sellaista. Logististen tietojärjestelmien tavoitteena on toimitusketjun prosesseissa tarvittavan tiedon kerääminen, säilyttäminen ja hyödyntäminen (Ballou, 2004, s. 146). Logistiset tietojärjestelmät muodostuvat osajärjestelmistä, joita ovat tilaustenhallintajärjestelmät, varastonhallintajärjestelmät ja kuljetustenhallintajärjestelmät (Ballou, 2004, s. 148). Tässä tutkimuksessa artefaktin käytännön ympäristö painottuu kuljetustenhallintajärjestelmiin. Lisäksi toimeksiannon luonteen takia kuljetustenhallintajärjestelmät painottuvat tässä tutkimuksessa järjestelmiin, joissa kuljetusten kohteena ovat laatikot. Laatikoissa kuljetettavalla sisällöllä ei ole järjestelmän toiminnan kannalta merkitystä. Tässä tutkimuksessa laatikoiden kuljetustenhallintajärjestelmistä käytetään nimitystä logistinen laatikkojärjestelmä. Tutkimuksessa syntyvä metatason ohjeistus luodaan helpottamaan nimenomaan logististen laatikkojärjestelmien vaatimusmäärittelyn toteuttamista, jotta artefakti olisi uniikki. Luonnollisesti artefaktin ympäristöön kuuluu myös kaikki sidosryhmät, jotka käyttävät laatikkojärjestelmää tai ovat sen kanssa tekemisissä.

Alfa-yritys omistaa elintarviketeollisuudessa käytettävän laatikoiden seurantajärjestelmän. Alfa-yritys toimii ainoastaan palveluntarjoajana eikä se myöskään omista laatikoita. Alfa-yrityksen asiakkaat vuokraavat laatikot suoraan laatikoiden omistajalta. Varsinaiset

järjestelmän käyttäjät ovat alfa-yrityksen asiakkaita, jotka valmistavat erilaisia laatikoissa kuljetettavia elintarvikkeita. Alfa-yrityksen asiakkaat myyvät elintarvikkeitaan niille sopivissa paikoissa, yleensä päivittäistavarakaupoissa. Alfa-yrityksen tapauksessa laatikot toimivat itsessään myyntialustoina. Tyhjät laatikot toimitetaan päivittäistavarakaupoista pesulaan, josta puhtaat laatikot palaavat takaisin elintarvikkeiden valmistajien käyttöön.



Kuvio 6. Laatikoiden toimitusketju alfa-yrityksessä

Kaikki alfa-yrityksen asiakkaat ja laatikkopesulat käyttävät laatikoiden seurantajärjestelmää. Laatikoiden seurantajärjestelmää käytetään nimensä mukaisesti laatikoiden seuraamiseen. Se on luotu laatikoiden toimitusketjun hallintaan. Laatikoita seuraamalla saadaan varmuus siitä, missä vaiheessa toimitusketjua ja kenellä kukin laatikko on viimeksi ollut. Laatikoita seuraamalla pyritään puuttumaan poikkeamiin, kuten hitaaseen laatikonkiertoon ja hävikkiin. Laatikkopesulat käyttävät laatikoiden omistajan seurantajärjestelmää. Kaikki järjestelmät, joita käytetään samojen kuljetuslaatikoiden seuraamiseen, ovat yhteydessä laatikoiden omistajan järjestelmään.

Alfa-yrityksen laatikoiden seurantajärjestelmä toimii seuraavalla tavalla. Jokaisella laatikolla on yksilöity viivakoodi, joka luetaan aina asiakaskohtaisen tuotekeruun yhteydessä ja yhdistetään samalla kyseiseen asiakkaaseen. Laatikoiden luentaprosessin jälkeen vastuu kyseisistä laatikoista siirtyy toimitusketjun seuraavalle osapuolelle. Laatikot luetaan

viivakoodinlukijalla, joka on integroitu laatikoiden seurantajärjestelmään. Alfa-yrityksen seurantajärjestelmässä jokainen laatikko luetaan normaalisti kaksi tai kolme kertaa yhden kierron aikana. Ensimmäinen luenta tapahtuu elintarvikkeiden pakkauksen yhteydessä. Laatikot luetaan, kun laatikot on täytetty myytävillä elintarvikkeilla ja ne ovat valmiita kuljetusta varten. Tällöin vastuu laatikoista siirtyy elintarvikkeiden valmistajalta toimitusketjun seuraavalle osapuolelle. Toinen mahdollinen luenta tapahtuu terminaalissa, jonne laatikot kuljetetaan jakelua varten. Terminaalista vastuu siirtyy luennan jälkeen laatikkopesulalle. Kolmas luenta tapahtuu, kun puhtaat laatikot luovutetaan pesulasta takaisin elintarvikkeiden valmistajalle. On mahdollista, että yhden kierron aikana suoritetaan ainoastaan kaksi luentaa, jos elintarvikkeita ei kuljeteta jakeluterminaalin kautta, vaan ne menevät suoraan päivittäistavarakauppoihin. Kaupoissa ei tällä hetkellä lueta laatikoita.

5.2 Artefaktin tietopohja

Artefaktin tietopohja koostuu olemassa olevasta ohjelmistotuotannon, vaatimusmäärittelyn ja logistiikan tietojärjestelmien teoriasta. Lisäksi artefaktin tietopohjaan kuuluu kaikki se käytännön aineisto, jota alfa-yritykselle tehtävää vaatimusmäärittelyä varten kerätään. Tämä aineisto sisältää kyselyitä, haastatteluita ja järjestelmän käytön havainnointia.

Tutkimus toteutetaan suunnittelutieteellisenä tutkimuksena. Näin ollen artefaktin tietopohjaan kuuluu myös kappaleessa 4 esitelty suunnittelutieteellisen tutkimuksen teoria. Artefaktin kehitysprosessia ohjaa DSRM-prosessimalli, Hevnerin ja muiden (2004) informaatiosysteemien tutkimuksen viitekehys ja Hevnerin ja muiden (2004) suunnittelutieteellisen tutkimuksen seitsemän ohjetta.

5.3 Ongelman tunnistaminen ja motivointi

Tutkimuksen tavoitteena on selvittää, millainen on metatason ohjeistus, jonka avulla pystytään luomaan vaatimusmäärittely, jonka pohjalta voidaan toteuttaa mahdollisimman tehokas, tuottava ja ketterä logistinen laatikkojärjestelmä. Syy tällaisen ohjeistuksen tarpeellisuudelle perustuu käytännön logistisiin ja liiketoiminnallisiin ongelmiin.

Yritysten on mukauduttava reaaliaikaiseen aikakäsitykseen, koska reaaliaikaiseksi muuttunut maailma sitä vaatii. Yritysten on pystyttävä reagoimaan reaaliaikaisiin muutoksiin, esimerkiksi elintarviketeollisuudessa korostuvaan kysynnän nopeisiin muutoksiin. Asiakkaat haluavat lyhyempien toimitusaikojen lisäksi myös joustavuutta ja mukautuvuutta eli ketteryyttä heidän tarpeisiinsa. Ketteryyden mahdollistaa läpinäkyvä ja reaaliaikainen tietovirta, joka helpottaa varautumista ongelmatapauksiin ja toiminnan ohjaamista. Kuitenkin erityisesti elintarviketeollisuudessa ongelmana ovat useat rajapinnat ja niissä tapahtuvien toimenpiteiden läpinäkymättömyys ja suhteellisen suuri työmäärä. Monilla aloilla toimitusaikavaatimukset ovat tiukentuneet. Erityisesti kehitys korostuu elintarviketeollisuudessa. Nopeat toimitusaikavaatimukset yhdistettynä kysynnän nopeisiin muutoksiin ja tuoreusvaatimuksiin aiheuttavat myös uusia vaatimuksia tietojärjestelmille ja näin ollen uusia ongelmia.

Ongelmakenttää on hyvä tarkastella myös alfa-yrityksen näkökulmasta. Alfa-yrityksen nykyinen laatikoiden seurantajärjestelmä on hyvin pelkistetty ja vanha. Koska järjestelmä on vanha, se ei vastaa enää nykyisin käyttäjien vaatimuksia ja tarpeita. Nykyinen järjestelmä ei ole ketterä eikä se tuota lisäarvoa sen käyttäjille. Järjestelmän käyttäjät käyttävät järjestelmää ainoastaan laatikoiden lukemiseen. Ongelmana on, että järjestelmä on ilman sen kehittämistä sen pelkistettyjen ominaisuuksien vuoksi helposti kopioitavissa. Järjestelmä toisi lisäarvoa, jos se sisältäisi laatikoiden lukemisen lisäksi myös muita ominaisuuksia. Jos lisäominaisuudet lisäävät työn tehokkuutta ja ketteryyttä, järjestelmällä voisi saada enemmän tuottoa. Järjestelmän vanha ikä luo myös muita ongelmia. Järjestelmällä on laaja käyttäjäkunta, joka tarkoittaa sitä, että käyttäjien tietotekninen osaaminen vaihtelee hyvin paljon. Järjestelmän heikko käytettävyys ja käyttöönoton vaikeus

ovat nykyisessä järjestelmässä ongelmia. Järjestelmää ei saa otettua itse helposti käyttöön, vaan se vaatii manuaalisen asennuksen tietokoneelle asiantuntijan toimesta, mikä hidastaa käyttöönottoa. Järjestelmän käytettävyysongelmiin perehdytään tarkemmin kappaleessa 5.6. Lisäksi liitteissä on kuvakaappauksia alfa-yrityksen nykyisestä laatikkojärjestelmästä.

Kaikki edellä mainitut ongelmat ovat ratkaistavissa tehokkaan, tuottavan ja ketterän logistisen laatikkojärjestelmän avulla. Ongelmana ei ole tällaisen laatikkojärjestelmän toteuttaminen, vaan sen suunnittelu. Ei ole olemassa selkeää ohjeistusta, mitä tehokkaan, tuottavan ja ketterän logistisen laatikkojärjestelmän suunnittelussa tulee ottaa huomioon. Tästä syystä ohjeistus luodaan vaatimusmäärittelyä varten, sillä se on ohjelmistotuotannon tärkein vaihe. Vaatimusmäärittely on kuvaus ja suunnitelma uudesta palvelusta, jonka pohjalta järjestelmän toteutetaan. Metatason ohjeistus logistisen laatikkojärjestelmän vaatimusmäärittelylle toimii geneerisenä työkaluna, jota kaikki toimialan yritykset voivat hyödyntää toimeksiantajan lisäksi. Täten artefakti hyödyttää toimeksiantajan projektin lisäksi koko toimialaa.

5.4 Ratkaisun tavoitteiden määrittely

Artefaktin tavoitteena on antaa edellytykset tehokkaan, tuottavan ja ketterän logistisen laatikkojärjestelmän vaatimusmäärittelylle. Artefakti toimii laatikkojärjestelmän suunnittelijoiden ohjeistuksena ja tukena ja se auttaa heitä ymmärtämään, mitä laatikkojärjestelmän vaatimusmäärittelyssä tulisi ottaa huomioon ja miten se tulisi toteuttaa. Ohjeistusta voidaan käyttää sekä olemassa olevien laatikkojärjestelmien kehittämisessä että uusien laatikkojärjestelmien suunnittelussa. Ohjeistuksen on perustuttava tieteelliseen kirjallisuuteen ja artefaktin ympäristöstä tehtyihin havaintoihin. Lisäksi konkreettisena mitattavana tavoitteena on, että artefakti noudattaa DSRM-prosessimallin lisäksi Hevnerin ja muiden (2004) suunnittelutieteellisen tutkimuksen seitsemää ohjetta.

5.5 Suunnitellun artefaktin esittely

Tässä kappaleessa esitellään tutkimustuloksena syntynyt varsinainen artefakti, metatason ohjeistus logistisen laatikkojärjestelmän vaatimusmäärittelylle. Metatason ohjeiden lisäksi jokainen ohje sisältää kuvauksen, jossa kerrotaan, mitä ohjetta noudattamalla pyritään saavuttamaan. Kuvauksen tarkoituksena on lisätä ohjeen ymmärrettävyyttä ja perustella ohjeen täsmällisyyttä juurruttaen se kirjallisuuteen ja käytännön havaintoihin.

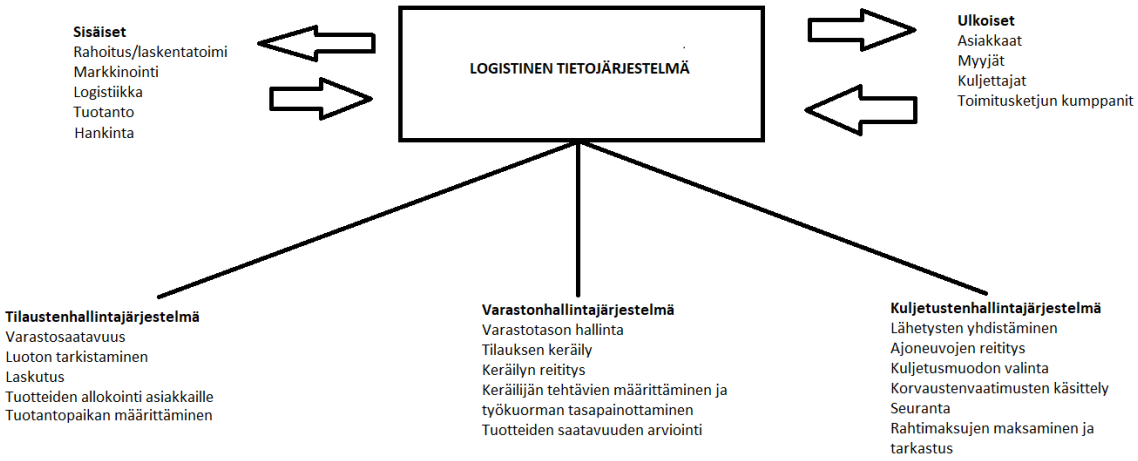
Ohje 1. Tunnista vaatimusten kartoittamista varten kaikki sidosryhmät, jotka ovat laatikkojärjestelmän kanssa tekemisissä.

Todennäköisesti työkokonaisuudesta kiinnostuneet sidosryhmät, kuten asiakas, ovat ilmaisseet vaatimuksensa esimerkiksi erillisessä vaatimusdokumentissa tai toimeksiannossa. On kuitenkin hyvin todennäköistä, että työhön liittyy sidosryhmiä, jotka eivät jostain syystä ole ilmaisseet näkemyksiään. Tällaisten sidosryhmien odotukset on kerättävä aktiivisella toiminnalla. Myös ainoastaan vähän kiinnostusta osoittavien sidosryhmien näkemyksiä on kerättävä, koska niillä saattaa olla suuri merkitys projektin onnistumisen kannalta. (Kosola, 2013, s. 29.)

Kuten jo aikaisemmin todettiin, logistiset tietojärjestelmät koostuvat tilausten-, varaston- ja kuljetustenhallintajärjestelmistä. Näistä jokainen sisältää informaatiota liiketoiminnallisiin tarkoituksiin ja päätöksenteon apuvälineitä, jotka auttavat tietyn toiminnan suunnittelussa. Informaation virtaaminen yrityksen eri toimintojen ja tietojärjestelmien välillä luo integroidun järjestelmän. (Ballou, 2004, s. 148.)

Logistisen tietojärjestelmän tulisi olla kattava ja riittävän kykenevä kommunikoimaan yrityksen toimialueiden lisäksi myös toimitusketjun jäsenten välillä. Valittujen myyntitietojen, lähetysten, tuotantoaikataulujen, varastosaatavuuden, tilauksen tilan ja vastaavien tietojen jakaminen myyjien ja ostajien välillä auttaa vähentämään epävarmuustekijöitä koko toimitusketjussa käyttäjien löytäessä tapoja hyötyä informaation saatavuudesta.

Kuitenkaan tietoja, jotka voivat vaarantaa yrityksen kilpailuaseman, ei haluta jakaa. (Ballou, 2004, s. 147.)



Kuvio 7. Yleiskatsaus logistiseen tietojärjestelmään (Ballou, 2004, s. 147).

Logistinen laatikkojärjestelmä ei sisällä aina välttämättä kaikkia logistisen tietojärjestelmän ominaisuuksia. Laatikkojärjestelmä voi olla esimerkiksi laatikoiden kuljetustenhallintajärjestelmä tai ainoastaan laatikoiden seurantajärjestelmä. Laatikkojärjestelmän ominaisuuksien määrä vaikuttaa siihen, miten paljon sidosryhmiä järjestelmään liittyy. Ominaisuuksien lisääntyessä myös järjestelmän käyttäjien tai sen suunnitteluun vaadittava henkilöstön määrä usein lisääntyy.

Laatikkojärjestelmän ollessa ainoastaan laatikoiden seurantajärjestelmä sidosryhmiä ovat vähintään kaikki toimitusketjun kumppanit. Esimerkiksi elintarviketeollisuudessa käytettävässä laatikkojärjestelmässä sidosryhmiä ovat vähintään elintarvikkeita valmistavat yritykset ja laatikkopesula. Jos laatikkojärjestelmä sisältää seurantajärjestelmän lisäksi tilaustenhallintajärjestelmän, varastonhallintajärjestelmän ja kuljetustenhallintajärjestelmän, sidosryhmiä ovat tällöin myös tilaustenkäsittelijät, varastotyöntekijät ja laatikoiden kuljettajat. Luonnollisesti muita ulkoisia laatikkojärjestelmän sidosryhmiä ovat kaikki yhteistyökumppanit, jotka ovat mukana järjestelmän suunnittelussa ja toteutuksessa.

Ohje 2. Valitse sopiva aineistonkeruumenetelmä vaatimusten kartoittamista varten.

Sidosryhmien tunnistamisen jälkeen niiden odotukset ja vaatimukset on selvitettävä. Kun vaatimuksia kootaan, on huomioitava, että sidosryhmien vaatimukset voivat kohdistua sekä tehtävän suorittamiseen että tuotteeseen. Eri sidosryhmien vaatimukset ovat tyypillisesti erilaisia. Käyttömukavuus ja helppous korostuvat usein loppukäyttäjien vaatimuksissa, kun taas kustannukset ja tehtävän suorittaminen korostuvat operaattoritahon vaatimuksissa. Kaikkien sidosryhmien vaatimukset tulee ottaa huomioon, mutta tämä ei tarkoita sitä, että kaikki vaatimukset otettaisiin mukaan sellaisenaan. (Kosola, 2013, s. 29-30.)

Vaatimusten kartoittamista varten on valittava sopiva aineistonkeruumenetelmä. Vaatimusten kartoitustekniikat esiteltiin luvussa 3.1.1. Laatikkojärjestelmän vaatimusten kartoitus kannattaa aloittaa yksinkertaisilla perinteisillä tekniikoilla, esimerkiksi kyselemällä ja haastattelemalla järjestelmän tulevia käyttäjiä. Kartoitustekniikoilla pyritään saamaan selville, mitä vaatimuksia käyttäjillä on uudelle laatikkojärjestelmälle.

Kysely voidaan toteuttaa kyselylomaketutkimuksena. Kyselylomaketta pidetään perinteisenä tapana kerätä tutkimusaineistoa. Kyselylomakkeen muoto vaihtelee kohderyhmän ja tarkoituksen mukaan. Selvimmät ulkoiset kyselylomakkeen erot liittyvät tutkijan läsnäoloon aineistonkeruutilanteessa ja aineistonkeruun suorittamiseen joko yksittäin tai yhtäaikaisesti suurelle joukolle. Kysymykset luovat tutkimuksen onnistumiselle perustan, joten niiden muotoilussa tulee olla huolellinen. Kysymysten muotoilu aiheuttaa tutkimustuloksiin eniten virheitä, jos vastaaja ei ymmärrä kysymyksiä tutkijan tarkoittamalla tavalla. Tällöin tulokset vääristyvät. Kysymykset eivät saa olla johdattelevia ja niiden tulee olla yksiselitteisiä. Kysymykset rakennetaan tutkimusongelmien ja tutkimuksen tavoitteiden mukaisesti. Aineiston kerääminen aloitetaan vasta tutkimusongelmien täsmentymisen jälkeen, koska tällöin tiedetään mitä kyselyn avulla pyritään löytämään. Samalla muistetaan kysyä kaikki oleellinen ja vältytään turhilta kysymyksiltä. (Aaltola & Valli, 2007, s. 102-103.)

Kyselylomaketutkimus sopii harvoin ainoaksi aineistonkeruumenetelmäksi, koska erityisesti internet- ja sähköpostikyselyissä vastausprosentit jäävät usein pieniksi. Lisäksi tutkijan kokemus ja aihepiirin ymmärtäminen vaikuttavat kysymyslomakkeen muotoiluun ja sisältöön, jotka taas suoraan vaikuttavat tutkimustuloksiin. Kyselyn ollessa esimerkiksi internet- tai sähköpostikysely osa vastaajista voi kokea kyselyn turhaksi ja työlääksi. Kyselyn vastaanottaja ei myöskään välttämättä koe velvollisuudekseen vastata kyselyyn, jos kukaan ei ole fyysisestä valvomassa, vastaako vastaanottaja kyselyyn vai ei. Mikäli vastaanottaja päättää vastata kyselyyn, on mahdollista, että kyselyyn vastataan huolimattomasti. Näin ollen laatikkojärjestelmän vaatimusten kartoittamista varten suositellaan kyselyn lisäksi käyttämään jotain toista kartoitustekniikkaa, esimerkiksi tulevien käyttäjien haastattelua.

Haastattelu on ilmeinen ja helppokäyttöinen vaatimusten kartoitustekniikka. Haastattelu voi olla strukturoimaton, strukturoitu tai puolistrukturoitu haastattelu. Strukturoimattomat haastattelut ovat luonteeltaan keskustelunomaisia ja rentoja. Strukturoimaton haastattelu voidaan järjestää milloin ja missä tahansa, kunhan sekä vaatimusten kokoaja ja asiakas ovat yhdessä. (Laplante, 2014, s. 58-59.) Strukturoimattomassa tai avoimessa haastattelussa tarkoituksena on keskustella haastattelijan etukäteen päättämistä teemoista, mutta strukturoimattomassa haastattelussa edetään keskustelunomaisesti ja luonnollisesti antaen haastateltavan kokemuksille, muistoille, mielipiteille ja perusteluille tilaa tarkkojen kysymysten sijasta. (Eskola & Suoranta, 2000, s. 86-88.) Strukturoimattomien haastatteluiden onnistuminen riippuu haastattelijan taitotasosta. Tämän takia strukturoidut tai puolistrukturoidut haastattelut ovat suositeltavia. (Laplante, 2014, s. 59.)

Strukturoidut haastattelut ovat luonteeltaan paljon formaalimpia ja niissä käytetään ennalta määriteltyjä kysymyksiä, jotka ovat tarkasti suunniteltuja. Mallipohjat ovat erittäin hyödyllisiä strukturoidun tyylin haastatteluissa. Haittapuolena strukturoiduissa haastatteluissa on, että osa asiakkaista saattaa jättää oleellista informaatiota antamatta haastattelun kontrolloidun rakenteen vuoksi. (Laplante, 2014, s. 59.)

Puolistrukturoitu haastattelu yhdistää strukturoidun ja strukturoimattoman haastattelun parhaat ominaisuudet. Toisin sanoen haastattelija valmistelee huolellisesti harkitun luettelon kysymyksistä, mutta haastattelun aikana ilmenevien strukturoimattomien kysymysten esittäminen on myös sallittua. (Laplante, 2014, s. 59.)

Vaikka strukturoidut haastattelut ovat suositeltavia, haastattelutavan valinta on opportunistinen päätös. Esimerkiksi asiakkaan yrityskulttuurin ollessa informaalia, rentoa ja luotettavaa, strukturoimattomat haastattelut saattavat sopia paremmin. Prosessikeskeisissä organisaatioissa strukturoidut ja puolistrukturoidut haastattelut ovat todennäköisesti parempia. Alla olevassa taulukossa on haastattelukysymysesimerkkejä, joita voidaan käyttää kaikissa haastattelutyypeissä. (Laplante, 2014, s. 59.)

Taulukko 6. Haastattelukysymysesimerkkejä vaatimusten kartoittamiseen (Laplante, 2014, s. 59).

Nimeä järjestelmän olennainen ominaisuus.
Miksi tämä ominaisuus on tärkeä?
Asteikolla yhdestä viiteen, miten luokittelisit tämän ominaisuuden tärkeyden?
Kuinka tärkeä tämä ominaisuus on suhteessa muihin ominaisuuksiin?
Mitkä muut ominaisuudet ovat riippuvaisia tästä ominaisuudesta?
Minkä muiden ominaisuuksien on oltava tästä ominaisuudesta riippumattomia?
Mitä muita havaintoja voit tehdä tästä ominaisuudesta?

Mikäli kyseessä on olemassa olevan laatikkojärjestelmän kehitysprojekti, voidaan perinteisten kartoitustekniikoiden lisäksi käyttää etnografista kartoitustekniikkaa, havainnointia. Järvisen ja Järvisen (2011, s. 95) mukaan etnografista metodia noudattava tutkija menee pidemmäksi aikaa tutkittavaan kohteeseen ja yrittää perehtyä toimintaan syvällisesti. Etnografista menetelmää soveltava tutkija tavoittelee tulevansa syntyperäiseksi (engl. native). Tuloksena vierailusta hän selittää ja kuvaa tutkittavan kohteen ilmiöitä ja toimintoja.

Havainnointi sopii erinomaisesti olemassa olevan laatikkojärjestelmän kehitysprojektiin. On hyvin mahdollista, että järjestelmän käyttäjät eivät vaatimusten kartoitusvaiheessa osaa antaa riittävästi palautetta tai kehitysehdotuksia. Tämä voi johtua siitä, että järjestelmän käyttäjät ovat tottuneet käyttämään sitä. Käyttäjät eivät välttämättä tästä syystä itse tunnista ja huomaa järjestelmän vikoja ja puutteita. Ulkopuolinen tutkija tarkastelee järjestelmää uudesta näkökulmasta ja voi löytää ongelmia ja kehitysehdotuksia, joita järjestelmän käyttäjä ei välttämättä koskaan huomaisi. Havainnointi toteutetaan käytännössä seuraamalla laatikkojärjestelmän käyttäjien työntekoa. Havainnoinnin tueksi tutkija voi havainnoinnin yhteydessä esittää käyttäjille tarkentavia kysymyksiä ja kehitysideoita.

Ohje 3. Selvitä, miten laatikkojärjestelmällä tuotetaan lisäarvoa asiakkaalle.

Asiakkaalle tuotettava arvo tarkoittaa hyödyn määrää, jonka asiakas saa palvelusta suhteessa näkemäänsä vaivaan (Kinnunen & Sinivuori, 2004, s. 21). Lisäarvon käsitteeseen pätee sama määritelmä. Lisäarvo voidaan ymmärtää myös arvona, jonka asiakas saa palvelussa oletetun arvon lisäksi. Tällöin tarkoitetaan jotain ylimääräistä, jopa odotukset ylittävää, palvelua suhteessa entiseen hintaan. Lisäarvo voi tarkoittaa esimerkiksi asiakkaan palvelusta saamaa hyötyä verrattuna eri palveluntarjoajien samoihin palveluihin. (Ala-Mutka & Talvela, 2004, s. 15.) Lisäarvo on kilpailutekijä, joka näkyy tuotteen hyödyn lisäyksenä tai tuotteen hinnan laskuna (Tuulaniemi, 2011, s. 37).

Vaatimusten kartoittamisen yhteydessä ja jälkeen on syytä pohtia, miten laatikkojärjestelmällä tuotetaan lisäarvoa asiakkaalle. Laatikkojärjestelmän ollessa esimerkiksi ainoastaan laatikoiden seurantajärjestelmä eli osa kuljetustenhallintajärjestelmää, se on helposti kopioitavissa pelkistettyjen ominaisuuksien vuoksi. Logistisen tietojärjestelmän ominaisuudet esiteltiin kuviossa 7. Laatikkojärjestelmän suunnittelussa tulee pohtia, mitä logistisen tietojärjestelmän ominaisuuksia voisi sisällyttää logistiseen laatikkojärjestelmään. Mitä enemmän työtä helpottavia, mielekkäitä, toimitusketjun läpinäkyvyyttä lisääviä ja työkuormaa vähentäviä ominaisuuksia laatikkojärjestelmässä on, sitä

enemmän asiakkaat todennäköisesti järjestelmää käyttävät ja tarvitsevat. Tällöin myös järjestelmän kopioimisesta tulee vaikeampaa. Lisäarvoa tuottavia ominaisuuksia kannattaa etsiä myös tutkimalla asiakkaiden työympäristöä. Liittyykö asiakkaiden eli järjestelmän käyttäjien työhön muita järjestelmiä tai toimintoja, jotka olisi mahdollista sisällyttää laatikkojärjestelmään?

Ohje 4. Ota käytettävyys kokonaisvaltaisesti huomioon.

ISO 9241-11 -standardin mukaan käytettävyys on mittari, joka kertoo, kuinka tehokas, käyttökelpoinen ja miellyttävä tuote on käyttää, kun sen omat käyttäjät käyttävät sitä tuotteen oikeassa käyttöympäristössä. Käyttökelpoisuuden määritelmän mukaan lopputulos on tarkalleen oikea, virheetön ja täydellinen. Tehokkuutta taas mitataan resursseina, rahana ja aikana. Tuotteen tehokkuus sisältää opittavuuden ja osittain myös helppokäyttöisyyden. (Sinkkonen ja muut, 2009, s. 20.)

Käytettävyys on laatuominaisuus, jonka perusteella arvioidaan, kuinka helppo käyttöliittymiä on käyttää (Nielsen, 2012). Käyttöliittymällä tarkoitetaan sitä ohjelman tai laitteen osaa, jonka avulla käyttäjä käyttää ohjelmaa tai laitetta (Tietotekniikan termitalkoot, 2000). Nielsenin (2012) mukaan käytettävyyttä voidaan mitata viiden eri laatukomponentin avulla, jotka ovat opittavuus, tehokkuus, muistettavuus, virheet ja tyytyväisyys.

Käytettävyys kannattaa huomioida kokonaisvaltaisesti jo logistisen laatikkojärjestelmän vaatimusmäärittelystä lähtien. Järjestelmän hyvällä käytettävyydellä saadaan aikaan useita positiivisia vaikutuksia. Sinkkosen ja muiden (2009, s. 28, 30) mukaan käyttäjäkeskeisesti tehty verkkopalvelu luo esimerkiksi kilpailuetua ja voittoa yritykselle. Käyttäjille sopiva ja helppokäyttöinen palvelu myy paremmin.

Sama pätee logistisiin laatikkojärjestelmiin. Järjestelmä, jonka käyttöönotto ja käyttäminen on helppoa, myy paremmin. Logistisilla laatikkojärjestelmillä on usein myös laaja käyttäjäkunta, joiden tietotekniset taidot vaihtelevat hyvin paljon. Laatikkojärjestelmän

tulisi olla sellainen, että jokainen työntekijä osaa käyttää järjestelmää käyttäjän taitotasosta riippumatta. Lisäksi kynnys järjestelmän ostamiselle on oltava matala. Toisin sanoen järjestelmän käyttöönottamisen tulee olla helppoa ja vaivatonta.

Laatikkojärjestelmän vaatimusmäärittelyssä on pyrittävä löytämään tasapaino järjestelmän lisäarvoa tuottavien ominaisuuksien ja käytettävyyden välille. Järjestelmään ei saa lisätä ominaisuuksia käytettävyyden kustannuksella. Tämä tarkoittaa sitä, että järjestelmään kannattaa lisätä ainoastaan helppokäyttöisiä ominaisuuksia. Suurin osa asiakkaista ei osaa käyttää monimutkaisia ja vaikeakäyttöisiä toimintoja. Toisin sanoen käyttämättömät ominaisuudet eivät tuota lisäarvoa asiakkaalle. Kun laatikkojärjestelmän vaatimuksia kartoitetaan, tulee järjestelmäsuunnittelijoiden jokaisen vaatimuksen kohdalla pohtia, miten kukin vaatimus toteutetaan käyttäjäystävällisesti ja onko se ylipäätään mahdollista.

Ohje 5. Analysoi kerätyt ja havaitut vaatimukset ja priorisoi ne.

Kun uuden laatikkojärjestelmän vaatimukset on sopivilla menetelmillä kerätty ja havaittu pohtimalla samalla lisäarvon tuottamista käytettävyys kokonaisvaltaisesti huomioiden, on vaatimukset analysoitava ja priorisoitava. Vaatimusten analysointiin perehdyttiin luvussa 3.1.2. Tiivistetysti vaatimusten analysoinnissa vaatimuksia voidaan tarkentaa sekä selvitellä niiden keskinäisiä suhteita ja prioriteettia (Haikala & Mikkonen, 2011, s. 66). Vaatimusten analysoinnissa tarkoituksena on arvioida, ovatko vaatimukset riittävän laadukkaita, jotta niiden pohjalta voidaan aloittaa ratkaisun suunnittelu ilman merkittäviä riskejä (Kosola, 2013, s. 48).

Kaikki vaatimusten kartoitusvaiheessa kerätyt ja havaitut vaatimukset eivät todennäköisesti ole toteuttamiskelpoisia sellaisenaan. Vaatimukset voivat olla esimerkiksi epärealistisia, epämääräisiä, ristiriidassa keskenään, riippuvaisia toisistaan, epäjohdonmukaisia ja keskeneräisiä. Vaatimukset on analysointivaiheessa tarkennettava ja muutettava selkeiksi kaikkien ymmärrettäviksi vaatimuksiksi. On myös hyvin mahdollista, että

järjestelmäsuunnittelijat eivät ole ymmärtäneet riittävän tarkkaan kaikkia sidosryhmien määrittelemiä vaatimuksia, jolloin epäselvyydet on hyvä selvittää yhdessä kyseisten sidosryhmien kanssa. Myös havaitut ja tunnistetut vaatimukset on hyvä esittää ja hyväksyttää laatikkojärjestelmän käyttäjillä ja muilla sidosryhmillä. Analysointivaiheessa on tutkittava, mitä asiakas- ja käyttäjävaatimukset vaativat järjestelmältä. Toisin sanoen on selvitettävä, mitkä ovat uuden laatikkojärjestelmän järjestelmävaatimukset.

Analysoinnissa on lisäksi tunnistettava, mitkä ovat suorituskyvyn, aikataulun ja kustannusten kannalta keskeisimmät vaatimukset (Kosola, 2013, s. 48). Tämä tarkoittaa sitä, että kaikkia vaatimuksia ei voi todennäköisesti sisällyttää uuteen järjestelmään, vaikka vaatimukset olisi analysoitu erittäin perusteellisesti ja muutettu ymmärrettävään muotoon, vaan ne on priorisoitava.

Projekteissa, joissa tuotetaan suuria monimutkaisia ohjelmistointensiivisiä (engl. software-intensive) järjestelmiä, on usein satoja tai jopa tuhansia yksittäisiä vaatimuksia. Ei ole myöskään epätavallista, että tällaisia järjestelmiä hankkivilla asiakasorganisaatioilla on päteviä syitä haluta, että kaikki vaatimukset toteutetaan. Tällaisissa hankkeissa ei voida kuitenkaan välttyä ongelmilta. (Firesmith, 2004.)

Kaikki vaatimukset eivät ole yhtä tärkeitä ja järjestelmän monet eri sidosryhmät eivät yleensä ole samaa mieltä siitä, mitkä vaatimukset ovat tärkeimpiä. Kaikilla hankkeilla on rajalliset resurssit budjetin, henkilöstön ja aikataulun suhteen. Kaikkien vaatimusten toteuttaminen on yleensä mahdotonta, ainakin järjestelmän ensimmäisen version julkaisun aikana. Sen vuoksi tyypillisesti ei-triviaaliset järjestelmät toteutetaan inkrementaalisesti eli vaiheittain. Suurten järjestelmien kehitykseen kuluu kuukausia tai usein jopa useita vuosia. Tänä aikana yritysten ympäristö ja tarpeet muuttuvat sekä uusia vaatimuksia ilmenee. Vaatimusten asianmukainen priorisointi tarjoaa projektille merkittäviä hyötyjä. (Firesmith, 2004.) Priorisoinnista saatavia hyötyjä on esitetty seuraavassa taulukossa.

Taulukko 7. Vaatimusten priorisoinnista saatavia hyötyjä (Firesmith, 2004).

Hyöty	Kuvaus
Aikataulun muokattavuus	Iteratiivista inkrementaalista kehityssykliä käytettäessä projektipäälliköllä ja asiakkaalla on mahdollisuus muokata projektiaikataulua, jotta rajoitettujen resurssien ja kiinteiden määräaikojen kanssa pärjätään.
Parempi asiakastyytyväisyys	Asiakkaan tärkeimpien vaatimusten toteuttaminen ensin parantaa asiakastyytyväisyyttä.
Matalampi peruutusriski	Projektin peruutusriski pienenee, koska jokaisella inkrementaalisella kehityssyklillä pystytään osoittamaan, että projekti etenee. Vaikka projekti peruutettaisiinkin ennen järjestelmän viimeistä versiota, tehty työ ei mene hukkaan, koska joitakin tärkeitä toimintoja on jo otettu käyttöön ja toimitettu asiakkaalle.
Kaikkien sidosryhmien vaatimusten huomioiminen	Kun vaatimukset priorisoidaan, tulee kaikkien sidosryhmien vaatimukset otettua huomioon.
Hyötyjen arviointi	Prioriteetit antavat johdolle ja suunnittelulle karkean arvion erilaisten vaatimusten eduista, mikä on hyödyllistä vaatimusten kustannus-hyötyanalyysejä suorittaessa. Nämä analyysit auttavat määrittämään, mihin vaatimuksiin rajalliset projektiresurssit kannattaa käyttää.

Investointien priorisointi	Vaatimusprioriteetit voivat auttaa määrittämään, kuinka rajoitetut projektiresurssit tulisi priorisoida. Esimerkiksi projekti voi suunnata suurimman osan rajallisista resursseistaan laadunvarmistukseen ja järjestelmän testaamiseen korkeimman prioriteetin vaatimusten mukaisesti.

Firesmithin (2004) mukaan vaatimukset voidaan priorisoida monien erilaisten, toisiinsa liittyvien ja jopa vastakkaisten näkökulmien mukaan. Eri sidosryhmät voivat arvostaa tiettyä näkökulmaa enemmän. Seuraavassa taulukossa on esitelty tavat, joiden perusteella vaatimukset voidaan priorisoida.

Taulukko 8. Vaatimusten priorisointitapoja (Firesmith, 2004).

Priorisointitapa	Kuvaus
Sidosryhmien omat henkilökohtaiset mieltymykset	Eri sidosryhmät pitävät tiettyjä vaatimuksia parempana kuin muut. Tällainen tilanne syntyy yleensä silloin, kun käytännön syitten takia kaikkia vaatimuksia ei voida toteuttaa.
Liiketoiminnallinen arvo	Osa vaatimuksista tarjoaa liiketoiminnalle enemmän arvoa kuin toiset.
Haittojen välttäminen	Vaatimukset voidaan priorisoida sen mukaan, mitä vahinkoa tapahtuu, jos vaatimusta ei toteuteta. Tätä tapaa käytetään esimerkiksi turvallisuusvaatimuksissa.
Riskit	Vaatimukset voidaan priorisoida toteutukseen liittyvän riskin perusteella.

	Priorisointi voidaan aloittaa joko suurimman tai pienimmän riskin vaatimuksista.
Kustannukset	Erilaisten vaatimusten toteutuksissa on erilaiset kehitys- tai elinkaarikustannukset. Tämän takia vaatimukset, jotka projektilla on varaa toteuttaa, toteutetaan usein ensin.
Vaikeusaste	Priorisointi voidaan toteuttaa vaatimuksen toteutuksen vaikeusasteen perusteella vaikeimmasta tai helpoimmasta vaatimuksesta alkaen.
Kiireellisyys	Vaatimukset voidaan priorisoida kiireellisyyden perusteella esimerkiksi silloin, kun kilpailijat tarjoavat vastaavanlaisia tuotteita. Tällöin tuote on saatava nopeasti markkinoille.
Vaatimusten vakaus	Kehitysprojektin aikana muuttumattomien vaatimusten toteuttaminen ensin on järkevää, jotta vältytään ylimääräiseltä työltä.
Vaatimusten välinen riippuvuus	Toisistaan riippuvien vaatimusten on oltava vähintään yhtä korkean prioriteetin vaatimuksia. Esimerkiksi, jos vaatimus A riippuu vaatimuksesta B, on vaatimus B toteutettava joko ennen vaatimusta A tai vaihtoehtoisesti samanaikaisesti, jos mahdollista.
Riippuvuudet toteutuksessa	Kun isoja järjestelmiä kehitetään, tietyt järjestelmäkomponentit ovat riippuvaisia toisista komponenteista, esimerkiksi

	perus- ja infrastruktuurikomponentteihin liittyvät vaatimukset on toteutettava ennen muita vaatimuksia.
Erilaiset vaatimukset	Erilaiset vaatimukset saattavat tarvita erilaisia lähestymistapoja priorisointiin. Esimerkiksi ei-toiminnallisia vaatimuksia voidaan priorisoida suoraan, kun taas toiminnallisia vaatimuksia voidaan priorisoida epäsuorasti käyttötapausten ja skenaarioiden avulla.
Lakisääteiset valtuudet	Vaatimuksille voidaan antaa korkeampi prioriteetti, jos niin velvoitetaan esimerkiksi lailla, asetuksella tai standardilla.
Käyttötiheys	Toiminnallisille vaatimuksille voidaan antaa korkeampi prioriteetti odotetun käyttötiheyden tai -määrän perusteella.
Uudelleenkäytettävyys	Vaatimuksen ollessa erittäin uudelleenkäytettävissä tuotteessa, voi olla viisasta antaa sille korkeampi prioriteetti, jotta minkään muun toiminnallisuuden ei tarvitse odottaa kyseisen vaatimuksen toteuttamista.

Ohje 6. Käytä vaatimusmäärittelyn tukena UML-kaavioita.

UML eli Unified Modelling Language on mallinnuskieli menetelmän sijaan. Useimmat menetelmät sisältävät sekä prosessin että mallinnuskielen. Kuvauskieliä eli notaatioita käytetään menetelmissä suunnitelmien kuvaamiseen. Mallinnuskieli on pääosin graafinen kuvauskieli. Puolestaan menetelmissä käytettävää ohjeistusta suunnitelman laatimiseen tarvittavista vaiheista kutsutaan prosessiksi. (Fowler & Scott, 2002, s. 2.)

Ohjelmistokehityksessä varsinaisena päämääränä on lopulta aina koodin tuottaminen. UML:n käyttöä harkittaessa on aina otettava huomioon, miten se auttaa koodin kirjoittamisessa ja mihin kieltä tarvitaan. Perussyy UML:n käyttöön liittyy viestintään. UML:n avulla voidaan esittää tiettyjä käsitteitä selkeämmin kuin muita vaihtoehtoja käyttämällä. Luonnolliset kielet ovat liian epätäsmällisiä monimutkaisten käsitteiden esittämiseen, kun taas ohjelmointikielet ovat täsmällisiä, mutta liian yksityiskohtaisia. Kehitystyössä yksi suurimmista haasteista on rakentaa juuri oikeanlainen järjestelmä, joka vastaa käyttäjien tarpeita eikä tule liian kalliiksi. Käyttäjien maailman ymmärtäminen ja hyvien viestintäyhteyksien muodostaminen ovat laadukkaan ohjelmistokehityksen avaimia. (Fowler & Scott, 2002, s. 7, 9-10.)

Käyttötapaukset ovat ilmeisin tekniikka tämän ongelman ratkaisemiseksi. Käyttötapaus kuvaa järjestelmää yhdestä näkökulmasta. Ulkoinen kuva järjestelmästä muodostuu kaikkien käyttötapausten summasta, ja tämä auttaa selittämään järjestelmän toimintaa. Käyttäjien tarpeiden kartoittamisessa keskeisenä tekijänä voidaan pitää hyvää käyttötapauskokoelmaa. Käyttötapaukset ovat hyviä välineitä projektin suunnitteluun, koska ne ohjaavat iteratiivista suunnittelua, joka puolestaan antaa säännöllistä palautetta käyttäjille siitä, mihin suuntaan ohjelmisto on kehittymässä. (Fowler & Scott, 2002, s. 10.) Käyttötapauskaavio on notaation osalta UML:n kaaviotekniikoista yksinkertaisin (Haikala & Mikkonen, 2011, s. 77).

Käyttötapaus on vaatimusten kartoitustekniikka. Yksinkertaisimmassa muodossaan käyttötapaus yksilöi vuorovaikutukseen osallistuvat toimijat ja nimeää vuorovaikutuksen tyypin. Käyttötapauksia täydennetään järjestelmän vuorovaikutusta kuvaavilla lisätiedoilla. Lisäinformaatio voi olla tekstimuotoinen kuvaus tai yksi tai useampi graafinen malli, kuten sekvenssi- tai tilakaavio. (Sommerville, 2011, s. 107.)

Haikalan ja Mikkosen (2011, s. 83) mukaan käyttötapausten monikäyttöisyys selittää osaltaan niiden yleistymistä:

1.Rajattaessa ja hahmotettaessa järjestelmää käyttötapauskaavio toimii apuna luomalla yhteisen näkemyksen järjestelmästä. Kaavio määrittää korkealla abstraktiotasolla järjestelmän toiminnallisuuden.

2. Usein yhdestä käyttötapauksesta löytyy monia ohjelmaan toteutettavia toimintoja.

3. Kun useissa eri käyttötapauksissa käytetään samoja toimintoja, uudelleenkäytettävyys lisääntyy.

4. Alustavat asiakasvaatimukset tarkentuvat käyttötapausten ansioista, sillä ne auttavat toimittajaa ja asiakasta ymmärtämään vaatimukset samalla tavalla.

5. Käyttötapauskaavion avulla voidaan tunnistaa järjestelmän sidosryhmiä.

Käyttötapauskaaviot auttavat myös siirtymistä toteutukseen ohjelmiston määrittelyvaiheesta, sillä käyttötapauskaavio auttaa toiminnallisuuden jakamisessa osajärjestelmiin. Käyttötapauskaavioita on mahdollista käyttää ohjelmistokehityksen organisoinnissa. Lisäksi käyttötapauskaavioita voidaan hyödyntää testitapausten suunnittelun pohjana. (Haikala & Mikkonen, 2011, s. 83.)

Käyttötapauskaavioiden lisäksi on olemassa myös paljon muita ohjelmistotuotannossa hyödynnettäviä UML-kaavioita. Niitä ovat muun muassa luokkakaaviot, tapahtumasekvenssikaaviot, tila- ja aktiviteettikaaviot ja tietovirtakaaviot (Haikala & Mikkonen, 2011, s. 8). Tämän ohjeen tarkoituksena ei ole esitellä jokaista erilaista UML-kaaviota, vaan auttaa järjestelmän suunnittelijoita ymmärtämään yleisellä tasolla, että eri UML-kaavioiden avulla on mahdollista saavuttaa useita erilaisia edellä mainitun esimerkin kaltaisia lisähyötyjä. Tämän vuoksi logistisen laatikkojärjestelmän vaatimusmäärittelyn tukena UML-kaavioiden hyödyntäminen on suositeltavaa. UML-mallinnuskielen avulla pystytään helpottamaan koodaajien työtä merkittävästi, sillä on mahdollista, että sovellusalue

on heille entuudestaan täysin outo ja vieras. Tällöin ihannetilanteessa he hyödyntävät esimerkiksi juuri käyttötapausten kuvauksia hyvinkin suoraviivaisesti.

Ohje 7. Dokumentoi vaatimukset selkeään ja ymmärrettävään muotoon.

Kun uuden järjestelmän kokonaiskuva alkaa edellä mainittujen ohjeiden avulla hahmottumaan, on vaatimukset dokumentoiva vaatimusdokumenttiin. Vaatimusten dokumentointiin perehdyttiin luvussa 3.1.3. Tämän ohjeen tarkoituksena on tiivistää kyseisen luvun sisältö ja poimia siitä tärkeimmät kohdat.

Vaatimusdokumenttia käytetään viestinnän välineenä. Vaatimusdokumenttia käytetään formaaliin muotoon dokumentoitujen järjestelmä- ja ohjelmistovaatimusten viestimiseen järjestelmä- ja ohjelmistosuunnittelijoille, asiakkaille ja järjestelmän suunnitteluprosessin johtajille. Vaatimusdokumentissa kuvataan järjestelmän tarjoamat toiminnot ja palvelut, järjestelmän toiminnan rajoitukset, yleiset järjestelmän ominaisuudet, muiden järjestelmien integraatiomääräykset, tiedot järjestelmän sovellusalueesta ja rajoitukset järjestelmän kehittämisprosessissa. (Kotonya & Sommerville, 1998, s. 15.)

Vaatimusdokumentin toteuttaminen on mahdollista monilla eri tavoilla. Dokumentin rakenteeseen vaikuttaa vaatimusten yksityiskohtaisuus, järjestelmän tyyppi, organisointikäytäntö ja vaatimusten suunnitteluprosessin aikataulu ja budjetti. (Kotonya & Sommerville, 1998, s. 15.) Taulukossa 2 on esitelty Sommervillen (2011, s. 93) vaatimusdokumentin esimerkkirakenne. Kaikkien ohjelmistokehitysorganisaatioiden tulisi hyödyntää standardin mukaisia vaatimusdokumenttipohjia ja jokaiselle suurelle projektiluokalle kannattaa valita oma vaatimusdokumenttipohja (Wiegers, 2003, s. 171). Myös oman vaatimusdokumenttistandardin määrittely on mahdollista kaikkien olennaisten tietojen säilyttämisen varmistamiseksi (Kotonya & Sommerville, 1998, s. 15). On hyvin todennäköistä, että suuren uuden logistisen laatikkojärjestelmän suunnitteluprojektiin ei sovellu sama vaatimusdokumenttipohja kuin sellaiseen projektiin, jossa laatikkojärjestelmään tehdään ainoastaan pieniä parannuksia.

Vaatimukset voidaan dokumentoida usealla eri tavalla. Vaatimusten dokumentointitavat on esitetty taulukossa 3. Logistisen laatikkojärjestelmän vaatimukset kannattaa dokumentoida ensisijaisesti kaikkien ymmärtämällä luonnollisella kielellä, jotta jokainen vaatimusdokumenttia lukeva henkilö saa riittävän kokonaiskuvan järjestelmästä. Luonnollisen kielen lisäksi vaatimusdokumenttiin sisällytetään graafisia notaatioita, joita ovat tässä tapauksessa kaikki vaatimusmäärittelyä tukevat UML-kaaviot. UML-kaavioiden tarkoituksena on helpottaa järjestelmän koodaajien työtä ja auttaa heitä hahmottamaan järjestelmän rakenne yksityiskohtaisemmin. UML-kaaviota suositeltiin käyttämään vaatimusmäärittelyn tukena ohjeessa 6.

Vaatimusdokumenttipohjan ja dokumentointitapojen lisäksi on kiinnitettävä huomiota siihen, miten yksittäisiä vaatimuksia on dokumentoitava. Vaatimukset voivat olla vaikeaselkoisia, monitulkintaisia ja ne ymmärretään usein väärin, vaikka ne olisi kirjoitettu luonnollisella kielellä (Kotonya & Sommerville, 1998, s. 19). ISO/IEC/IEEE 29148 standardin (2011, s. 11) mukaan jokaisen vaatimuksen tulisi olla tarpeellinen, toteutuksesta riippumaton, yksiselitteinen, johdonmukainen, täydellinen, ainutlaatuinen, toteutettava, jäljitettävä ja todennettava. Vaatimusten ominaisuuksia on kuvattu tarkemmin taulukossa 4.

Ohje 8. Hyväksytä vaatimusdokumentti sidosryhmillä.

Kun kaikki vaatimukset on dokumentoitu, vaatimusdokumentti on hyväksytettävä sidosryhmillä. Tästä prosessista voidaan käyttää nimitystä vaatimusten validointi. Vaatimusten validointiin perehdyttiin kappaleessa 3.1.4. Validointiprosessissa tarkastetaan, että vaatimukset määrittelevät järjestelmän asiakkaiden tarpeiden mukaisesti (Sommerville, 2011, s. 110). Validoinnissa varmistetaan, että vaatimukset ovat oikeita ja että ne osoittavat halutut laatuominaisuudet (Wiegers, 2003, s. 53). Vaatimusten validointi on päällekkäistä analyysin kanssa, koska molemmissa etsitään ongelmia vaatimuksista (Sommerville, 2011, s. 110). Prosessit kuitenkin eroavat tavoitteiltaan toisistaan. Tiivistetysti vaatimusten analysoinnissa pyritään vastaamaan kysymykseen ”Onko meillä oikeat

vaatimukset?”, kun taas vaatimusten validoinnissa pyritään vastaamaan kysymykseen ”Onko vaatimukset ymmärretty oikein?” Vaatimusten validointi on vaatimusdokumentin lopullisen luonnoksen tarkistamista. (Kotonya & Sommerville, 1998, s. 88.)

Ihannetilanteessa vaatimusmäärittely toteutetaan tiiviissä yhteistyössä sidosryhmien ja järjestelmän asiakkaiden kanssa. Tästä syystä on myös hyvien tapojen mukaista, että vaatimusdokumentin lopullinen versio hyväksytetään sidosryhmillä. On mahdollista, että vaatimusmäärittelyprojektin aikana asiakkaiden tarpeet ja tavoitteet muuttuvat, varsinkin jos kyseessä on suuren järjestelmän pitkäkestoinen suunnitteluprojekti. Järjestelmän fyysistä toteuttamista ei kannata aloittaa ennen kuin tärkeimmistä vaatimuksista on saavutettu lopullinen yhteisymmärrys sidosryhmien välille.

5.6 Artefaktin demonstrointi alfa-yrityksessä

Tässä kappaleessa suunniteltu artefakti demonstroidaan alfa-yrityksen uuden laatikkojärjestelmän kehitysprojektissa. Ohjeistuksen avulla alfa-yrityksen uudelle tietojärjestelmälle toteutetaan vaatimusmäärittely. Demonstroinnin avulla osoitetaan, miten hyvin artefakti soveltuu logistisen laatikkojärjestelmän kehitysprojektiin käytännössä.

Ohje 1. Tunnista vaatimusten kartoittamista varten kaikki sidosryhmät, jotka ovat laatikkojärjestelmän kanssa tekemisissä.

Alfa-yrityksen laatikkojärjestelmään liittyy kolme pääsidosryhmää, joiden mielipiteet on otettava huomioon vaatimusten kartoituksessa. Näitä ovat alfa-yrityksen asiakkaat, toimeksiantaja ja laatikoiden omistaja. Kaikkien sidosryhmien vaatimukset on kerättävä aktiivisella toiminnalla. Kehitysprojektiin ei liity muita sidosryhmiä, joiden mielipiteet tulisi ottaa huomioon. Luonnollisesti vaatimusmäärittelyn valmistuttua projektissa sidosryhmänä on yritys, joka vastaa järjestelmän fyysisestä toteutuksesta ja koodauksesta vaatimusmäärittelyn pohjalta. Seuraavissa kappaleissa kuvataan lyhyesti jokaista pääsidosryhmää.

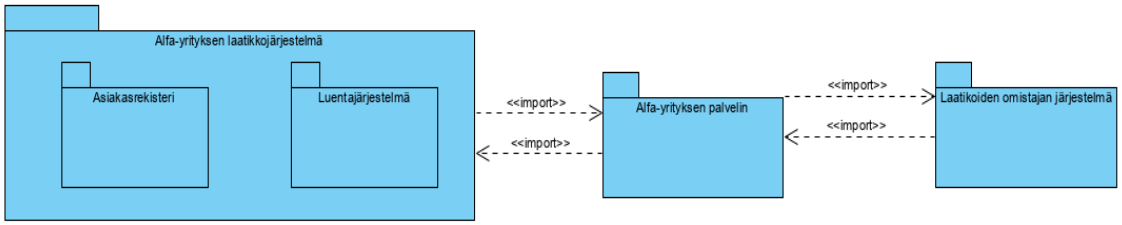
Alfa-yrityksellä on yli 50 asiakasta, joista kaikki valmistavat erilaisia elintarvikkeita. Riippuen valmistettavista tuotteista asiakkailla saattaa olla erilaisia vaatimuksia uudelle laatikkojärjestelmälle. Asiakkaiden koko ja tästä syystä myös tarvittavien laatikoiden määrä vaihtelee merkittävästi, mikä saattaa myös vaikuttaa asiakkaiden tarpeisiin.

Alfa-yritys on toimeksiantajana tässä tutkimuksessa. Toimeksiantajaa voidaan pitää tärkeimpänä yksittäisenä laatikkojärjestelmän kehitysprojektin sidosryhmänä, koska kaikki järjestelmään tulevat ja ehdotetut muutokset vaativat viime kädessä alfa-yrityksen hyväksynnän. Alfa-yrityksellä on jo myös useita ennakkovaatimuksia, jotka uuden laatikkojärjestelmän on täytettävä. Lisäksi kaikki kerätyt ja havaitut vaatimukset analysoidaan yhdessä alfa-yrityksen kanssa.

Laatikoiden omistaja on alfa-yrityksen laatikkojärjestelmän kehitysprojektissa rajoittavana tekijänä ja näin se on myös tärkeä sidosryhmä. Kuten aikaisemmin mainittiin, kaikkien laatikoiden omistajan laatikoiden seuraamiseen käytettävien tietojärjestelmien on oltava yhteydessä laatikoiden omistajan tietojärjestelmään. Syynä tälle on se, että laatikoiden omistaja haluaa seurata myös itse kaikkia kuljetuslaatikoita. Näin ollen alfa-yrityksen laatikkojärjestelmän on oltava yhteensopiva laatikoiden omistajan järjestelmien kanssa. Toisin sanoen uudet laatikkojärjestelmän ominaisuudet on hyväksytettävä ja suunniteltava yhdessä laatikoiden omistajan kanssa.

Ohje 2. Valitse sopiva aineistonkeruumenetelmä vaatimusten kartoittamista varten.

Vaatimusten kartoitus aloitettiin tutustumalla ensin yhdessä alfa-yrityksen kanssa nykyiseen laatikkojärjestelmään ja järjestelmäarkkitehtuuriin. Alfa-yritys oli jo ennakkoon määrittänyt vaatimuksia uudelle järjestelmäarkkitehtuurille. Nykyisen järjestelmän ja toivotun uuden järjestelmän ominaisuuksia ja toimintaa pyrittiin selventämään keskustelemalla yhdessä alfa-yrityksen kanssa. Ennakkovaatimusten ymmärtämisessä auttoi tarkentavien kysymysten esittäminen alfa-yritykselle haastattelemalla. Seuraavassa kuviossa kuvataan nykyistä järjestelmäarkkitehtuuria pakettikaaviona.



Kuvio 8. Alfa-yrityksen nykyinen järjestelmäarkkitehtuuri

Tällä hetkellä asiakkaiden käyttämä laatikkojärjestelmä pitää sisällään sekä laatikoiden luentaohjelman että asiakasrekisterin. Lisäksi laatikkojärjestelmällä pystyy tarkastelemaan laatikoiden lukemistietoja ja tulostamaan raportin laatikkotiedoista tietyltä aikaväliltä. Alfa-yrityksen palvelin vastaanottaa laatikoiden lukemistiedot, joita on myös mahdollista tarkastella. Laatikkotietojen tarkastelu on kuitenkin vaikeaa, koska siihen ei ole tällä hetkellä erillistä järjestelmää. Luentatiedot on haettava suoraan palvelimelta, mikä on vaikea ja työläs prosessi. Lisäksi luentatiedot tulevat vaikeasti ymmärrettävässä muodossa (käsittelemätöntä dataa excel-muodossa). Alfa-yrityksen palvelimelta laatikoiden lukemistiedot siirtyvät laatikoiden omistajan järjestelmään, josta seurataan kaikkia kuljetuslaatikoita.

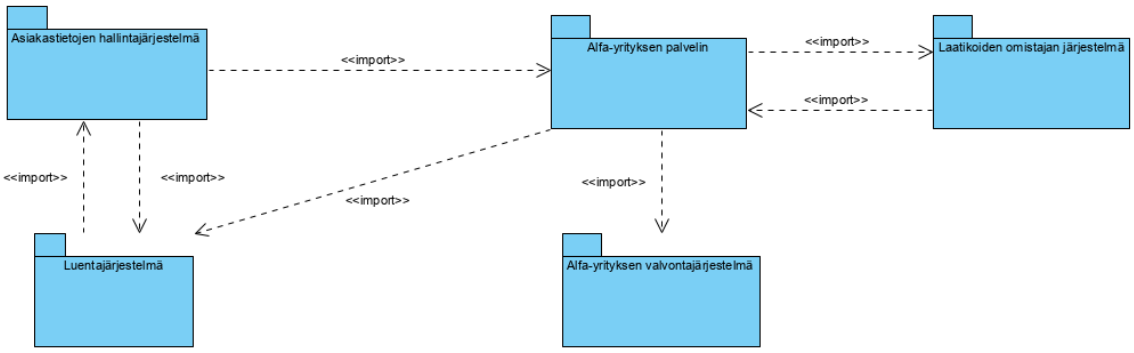
Alfa-yrityksellä oli ennakkovaatimuksena, että uuden järjestelmäkokonaisuuden jokainen osa toimisi itsenäisesti mobiiliverkossa. Tämä tarkoittaisi sitä, että järjestelmää ei varsinaisesti enää asennettaisi tietokoneelle asiantuntijan toimesta, vaan sitä pystyisi käyttämään suoraan netin kautta. Tällöin asiakas ei ole riippuvainen tietystä laitteesta ja sen toiminnasta, johon laatikkojärjestelmä on asennettu. Uudelle asiakkaalle lähetetään käyttöönoton yhteydessä kirjautumistunnukset, joiden avulla kyseinen asiakas pääsee käyttämään järjestelmiä.

Toisena ennakkovaatimuksena oli, että uudessa järjestelmäarkkitehtuurissa laatikoiden luentajärjestelmä ja asiakasrekisterin ylläpito erotetaan toisistaan kahdeksi erilliseksi ohjelmaksi. Tulevaisuudessa asiakkaalla olisi siis luentajärjestelmä, jonka avulla laatikoiden lukeminen tapahtuu ja erillinen järjestelmä, jonka kautta asiakas ylläpitää ja hallitsee

asiakastietoja. Asiakastietojen hallinnan pitäisi onnistua verkkoselaimella millä laitteella tahansa.

Luenta-asemana voi käyttää esimerkiksi mobiiliverkossa toimivaa tablettia, josta järjestelmän käyttäjä valitsee asiakkaan, jonne laatikoita luetaan. Asiakkaan valinta tehdään lukemalla kyseisen asiakkaan viivakoodi. Tabletilta järjestelmän käyttäjä näkee valitun asiakkaan tiedot ja kyseisen luentakerran laatikkotiedot liitteen 4 tyylisesti. On myös mahdollista, että laatikoiden luentaohjelmaa käytetään vaihtoehtoisesti puhelimella, joka yhdistetään suoraan fyysisesti viivakoodinlukijaan. Tällöin erillistä näyttöä ei tarvita. Tärkeintä on, että laatikoita luettaessa luentaohjelman käyttäjä saa varmuuden siitä, että luenta onnistuu. Tämä edellyttää, että viivakoodinlukija antaa äänimerkin, kun luenta onnistuu. Käyttäjän on myös saatava merkki siitä, jos luenta ei onnistu. Varsinainen luentajärjestelmä on Android-pohjainen ladattava sovellusapplikaatio.

Kolmantena alfa-yrityksen ennakkovaatimuksena oli, että uudessa järjestelmäarkkitehtuurissa olisi erillinen valvontajärjestelmä, jonka avulla alfa-yritys pääsee itse tarkastelemaan helpommin asiakkaidensa luentatietoja ja laskutushistoriaa. Seuraavassa kuviossa on alfa-yrityksen ennakkoon määrittelemä uusi järjestelmäarkkitehtuuri pakettikaaviona. Ohjeessa 6 perehdytään uuden laatikkojärjestelmän mallinnukseen tarkemmin. Mallinnuksen lisäksi järjestelmäkokonaisuuden toimintaa kuvataan kirjallisesti.



Kuvio 9. Alfa-yrityksen uusi järjestelmäarkkitehtuuri

Kun alfa-yrityksen ennakkovaatimuksiin ja nykyiseen ja uuteen järjestelmäarkkitehtuuriin oli tutustuttu, seuraavana aineistonkeruumenetelmänä vaatimusten kartoitusta varten oli kyselytutkimuksen toteuttaminen alfa-yrityksen asiakkaille. Kyselyn tarkoituksena oli alfa-yrityksen laatikkojärjestelmän nykytilanteen kartoittaminen ja kehitysehdotusten kerääminen järjestelmän käyttäjiltä. Kysely toteutettiin verkkokyselynä Google Forms -alustalla. Kysely lähetettiin kaikille alfa-yrityksen yli 50 asiakkaalle. Vastauksia kyselyyn tuli 14 kappaletta.

Suurin osa vastanneista kokee järjestelmän käytön tällä hetkellä helpoksi ja vaivattomaksi. Kyselyssä nousi kuitenkin esiin muutamia käytettävyyteen liittyviä ongelmia. Osalla vastanneista on ajoittaisia ongelmia laatikkotietojen lähettämisessä. Järjestelmä ei aina suostu lähettämään yhden luentakerran eriä eli tällöin kyseessä on yhteysongelma. Yksi asiakkaista koko turhauttavaksi, kun koko asiakasrekisterin joutuu tulostamaan, jos järjestelmään lisää uuden asiakkaan ja haluaa tulostaa ainoastaan kyseisen asiakkaan viivakoodin laatikoiden luentaa varten (liite 2). Lisäksi yhdellä vastaajalla oli ongelmia asiakkaiden lisäämisen kanssa asiakasrekisteriin, koska ohjelma tallentaa välillä tahattomasti tietoja.

Asiakkailla oli mielessä myös ominaisuuksia, joilla järjestelmän käyttöä voitaisiin helpottaa. Yksi asiakkaista kertoi, ettei asiakkaiden paikkaa asiakasrekisterissä pystyisi muokkaamaan, vaan uutta asiakasta lisättäessä tämä menee aina asiakasrekisterissä alimmaksi (liite 2). Lisäksi kehitysehdotuksia tuli myös laatikoiden ja asiakkaiden hakutoimintoihin. Tällä hetkellä yksittäisen laatikon luentatietojen löytäminen järjestelmästä on vaikeaa (liite 3). Asiakkaan mukaan olisi myös hyvä, jos esimerkiksi laatikkonumerolla pystyisi etsimään tietyn laatikon (liite 4). Lisäksi olisi hyvä, jos raportista näkisi selkeämmin jälkikäteen, mitkä laatikot on lähetetty kullekin asiakkaalle (liite 3). Myös asiakkaan haussa asiakasrekisteristä on yhden asiakkaan mukaan kehitettävää (liite 2). Toiveita tuli myös järjestelmän vakauden parantamisesta. Toive on otettava vakavasti, koska vastanneista yli 70% lukee laatikoita päivittäin. Uuden järjestelmän on kuitenkin tuettava

langattomia viivakoodinlukijoita nykyisen järjestelmän tapaan, koska vastanneista yli 90% käyttää tällä hetkellä langatonta lukijaa.

Alkuperäisenä ajatuksena oli, että kyselytutkimuksen jälkeen mentäisiin yhdessä alfa-yrityksen kanssa havainnoimaan asiakkaiden järjestelmän käyttöä useampaan eri yritykseen, koska oletuksena oli, että vastauksia kyselyyn ei tule riittävästi. Havainnoinnin tarkoituksena olisi seurata asiakkaiden järjestelmän käyttöä ja näin tutustua järjestelmään käytännön ympäristössä ja löytää parannusehdotuksia. Havainnoinnin etu perustuu siihen, että järjestelmän käyttäjä ei välttämättä itse osaa tunnistaa järjestelmään liittyviä ongelmia, koska hän on tottunut käyttämään järjestelmää sellaisenaan. Ulkopuolinen havainnoitsija saattaa löytää ongelmia, joihin järjestelmän käyttäjä ei välttämättä itse koskaan kiinnittäisi huomiota. Havainnointikäynnit päätettiin kuitenkin jättää tekemättä, koska kyselytutkimuksesta löydettiin jo useita käytettävyyteen liittyviä ongelmia. Havainnointikäynteihin kuluisi lisäksi paljon aikaa eivätkä ne välttämättä toisi esiin enempää uusia tällä hetkellä tiedostamattomia käytettävyysongelmia. Uudessa järjestelmässä käyttöliittymä uudistetaan kokonaan tietojärjestelmätieteen ammattilaisten toimesta, joten nykyisen käyttöliittymän tutkimiseen ei kannata käyttää enempää aikaa. Asiakkailla ei myöskään kyselytutkimuksen perusteella ollut mielessä mahdollisia uuteen järjestelmään tulevia lisäominaisuuksia, joten voidaan olettaa, ettei niitä tulisi havainnointikäyntien yhteydessäkään.

Vaatimusmäärittelyprojektin alkuvaiheessa suoritettiin kuitenkin yksi havainnointikäynti alfa-yrityksen asiakkaan x luona. Käynti suoritettiin, jotta nykyjärjestelmän toiminnan hahmottaminen olisi helpompaa. Järjestelmän käyttäjällä ei ollut varsinaisia parannusehdotuksia, mutta hän toivoi, että viivakoodinlukija olisi langaton. Kyseisessä yrityksessä langaton lukija helpottaisi laatikoiden lukemista ja se vähentäisi tarvetta siirrellä kuljetuslaatikoita huoneesta toiseen.

Huomio havainnointikäynnillä kiinnittyi laatikoiden luentaprosessiin. Yrityksellä oli järjestelmästä tulostettu paperilista, jossa oli kunkin jakelupaikan viivakoodi. Tämä

viivakoodi luettiin ensin, jolloin järjestelmään ilmestyi kyseisen toimituspaikan tiedot. Tämän jälkeen toimituspaikka yhdistettiin oikeaan laatikkoon lukemalla laatikon viivakoodi. Viivakoodinlukija antoi äänimerkin, kun luenta onnistui. Samaan aikaan järjestelmän näytölle ilmestyi asiakastietojen viereen kyseisen laatikon viivakoodin numerosarja tapahtuman todentamiseksi. Kun järjestelmän käyttäjä aloitti seuraavan laatikon luennan, tiedot edellisestä luennasta hävisivät näytöltä. Olisi hyvä, että kaikki yhden luentakerran tapahtumat näkyisivät näytöllä esimerkiksi allekkain. Näytettävissä tiedoissa olisi hyvä olla asiakkaan nimi ja laatikon/viivakoodin numerosarja. Tällöin voidaan varmistua siitä, että yksikään laatikko ei jää vahingossa lukematta. Yrityksessä laatikot luettiin seitsemän kappaleen erissä. Toki, kun kaikki luennat oli suoritettu ja tiedot lähetetty eteenpäin, pääsivulla näkyi, montako luentaa suoritettiin. Jos kuitenkin laatikoita luettaisiin enemmän kerralla, olisi silti mahdollista, että laatikoita jäisi lukematta.

Ohje 3. Selvitä, miten laatikkojärjestelmällä tuotetaan lisäarvoa asiakkaalle.

Luonnollisesti alfa-yrityksen laatikkojärjestelmän uudistamisen ja kehittämisen tarkoituksena on helpottaa asiakkaiden päivittäistä elämää. Alfa-yrityksen asiakkailta ei kuitenkaan tullut kyselytutkimuksen ja havainnointikäynnin yhteydessä varsinaisia ehdotuksia uusista lisäarvoa tuottavista ominaisuuksista. Alfa-yrityksellä oli valmiiksi ennakkotoiveissa kaksi uutta lisäarvoa tuottavaa ominaisuutta. Molemmat ominaisuudet sisällytetään uudessa järjestelmäarkkitehtuurissa asiakastietojen hallintajärjestelmään, jotta varsinainen luentajärjestelmä pysyy mahdollisimman selkeänä, yksinkertaisena ja helppokäyttöisenä.

Ensimmäinen lisäarvoa tuottava ominaisuus on hyvin yksinkertainen, kuljetuslaatikoiden tilaaminen suoraan asiakastietojen hallintajärjestelmästä. Ominaisuuden lisääminen vaatii hyväksynnän laatikkopesulalta ja laatikoiden omistajalta. Myös laatikkopesulan ja alfa-yrityksen uuden järjestelmän on oltava yhteensopivia keskenään. Laatikoiden tilaaminen onnistuu kirjautumalla yrityskohtaisilla tunnuksilla asiakastietojen hallintajärjestelmään. Järjestelmästä voi aina halutessaan tilata laatikoita tarvitsemansa määrän.

Ominaisuus sisältää myös saldonhallintaominaisuuden. Tämä tarkoittaa sitä, että asiakas voi asettaa haluamansa saldorajan, jonka alittuessa järjestelmä lähettää automaattisen laatikkotilauksen. Ominaisuus on yhteydessä laatikoiden luentajärjestelmä kanssa, jotta saldotiedot päivittyvät asiakastietojen hallintajärjestelmään. Aina, kun laatikko luetaan eteenpäin, vähenee laatikoiden varastosaldo automaattisesti. Laatikoiden tilaamisominaisuus on erittäin perusteltu, koska kyselytutkimukseen vastanneista asiakkaista kaikki tilaavat tällä hetkellä laatikoita puhelimitse tai sähköpostitse. Uusi ominaisuus vähentää asiakkaiden manuaalista työtä ja näin luo lisäarvoa asiakkaille.

Toinen lisäarvoa tuottava uusi ominaisuus on asiakastarrojen tulostaminen. Asiakastarralla tarkoitetaan kuljetuslaatikkoon liimattavaa tarraa, jossa tyypillisesti lukee esimerkiksi laatikon toimituspaikka, terminaalin putki ja pudotuspaikka. Tiedot ovat siis olennaisia laatikoiden kuljettajia varten. Havainnointikäynnin kohteena olleessa yrityksessä liimattiin erikseen tulostettu tai käsin kirjoitettu asiakastarran jokaiseen laatikkoon. Yrityksellä oli myös erillinen kansio, jossa oli kaikki toimitustiedot. Tämä toimintatapa vaikutti epäkäytännölliseltä. Uudessa laatikkojärjestelmässä asiakastarrojen tulostaminen on mahdollista laatikoiden lukemisen yhteydessä. Tällöin asiakkaiden ei tarvitse tehdä omia tarrapohjia tai kirjoittaa toimitustietoja käsin. Käytännössä jokaisen luennan yhteydessä järjestelmä tulostaa automaattisesti oikean asiakastarran. Laatikoiden toimitustiedot ovat saatavissa laatikoiden omistajan järjestelmästä. Tämäkin ominaisuus edellyttää hyväksynnän laatikoiden omistajalta. Vaihtoehtoisesti on mahdollista, että alfa-yrityksen asiakkaat itse lisäävät jokaisen asiakkaansa toimitustiedot järjestelmään. Tällöin toki asiakas joutuisi itse päivittämään tiedot, jos toimituspaikka muuttuu. Asiakastarrojen tulostusominaisuus vaatii toimiakseen yhteyden asiakkaan omaan tarrakoneeseen.

Asiakastarrojen tulostusominaisuutta hallitaan laatikoiden tilaamisominaisuuden tavoin asiakastietojen hallintajärjestelmästä. Jokainen asiakas voi itse päättää, haluaako pitää tätä ominaisuutta päällä vai ei. Alfa-yrityksellä on asiakkaita, joilla ei ole tarrakonetta. Tämä tarkoittaa sitä, että ominaisuuden käyttäminen ei ole välttämätöntä. Uusi

ominaisuus on perusteltu, sillä se vähentää asiakkaiden manuaalista työtä ja luo näin laatikoiden tilaamisominaisuuden tavoin lisäarvoa asiakkaille.

Ohje 4. Ota käytettävyys kokonaisvaltaisesti huomioon.

Käytettävyyden huomioiminen ja parantaminen olivat lähtökohtina tämän projektin toteuttamiselle. Alfa-yrityksen määrittelemän uuden järjestelmäarkkitehtuurin avulla tarkoituksena on parantaa kaikkien osapuolten käytettävyyttä. Uusi järjestelmäarkkitehtuuri selkeyttää ja helpottaa laatikkojärjestelmän käyttöönottoa ja käyttämistä. Lisäksi uuteen järjestelmäarkkitehtuuriin sisältyy uusi valvontajärjestelmä, jonka avulla alfa-yritys pystyy tarkastelemaan nykyistä helpommin asiakkaidensa laatikkotietoja. Jotta uusi järjestelmä saadaan toimimaan mobiiliverkossa ja mobiililaitteilla, se edellyttää asiakasrekisterin ja luentajärjestelmän eriyttämistä toisistaan. Varsinaisen luentajärjestelmän yksinkertaistaminen tähtää jokapäiväisen työskentelyn helpottamiseen.

Kyselytutkimuksen perusteella myös alfa-yrityksen asiakkaiden vaatimuksista lähes kaikki liittyivät käytettävyyden parantamiseen. Tärkeimpänä vaatimuksena nousi esiin toimintavarmuuden parantaminen. Tähän vaatimukseen pyritään vastaamaan juuri uuden järjestelmäarkkitehtuurin avulla. Kun jokainen laatikkojärjestelmän osa on itsenäisesti yhteydessä mobiiliverkkoon, tietojen pitäisi päivittyä reaaliajassa, jolloin laatikoiden luentavirheiden määrän pitäisi vähentyä merkittävästi. Asiakkailla oli vaatimuksia myös nykyiseen käyttöliittymään liittyen. Asiakkaat halusivat, että asiakasrekisterin ja laatikkotietojen hallintaa ja hakutoimintoja kehitettäisiin. Näihin ongelmiin pyritään vastaamaan uudistamalla käyttöliittymät kokonaan asiakkaiden vaatimusten mukaisesti.

Ohje 5. Analysoi kerätyt ja havaitut vaatimukset ja priorisoi ne.

Alfa-yrityksen ennakkovaatimukset ja kyselytutkimuksella kerätyt asiakasvaatimukset olivat valmiiksi ymmärrettävässä ja selkeässä muodossa. Tämän takia vaatimuksia ei ole tarpeellista analysoida enää tätä syvällisemmin. Kerätyt ja havaitut vaatimukset ovat

sellaisenaan riittävän laadukkaita uuden logistisen laatikkojärjestelmän suunnitteluun. Niitä ei ole myöskään tässä vaiheessa tarpeen käydä läpi sidosryhmien kanssa.

Priorisoinnin osalta alfa-yrityksen ennakkoon määrittelemiä vaatimuksia voidaan pitää kaikkein tärkeimpinä, koska aloite projektin käynnistämiselle tuli alfa-yritykseltä itseltään. Alfa-yrityksen vaatimusten toteuttaminen on myös kaikkein työläin ja aikaa vievin vaihe, koska alfa-yritys haluaa uudistaa koko järjestelmäarkkitehtuurin. Toiseksi tärkeimmät vaatimukset ovat asiakkailta kyselytutkimuksessa kerätyt vaatimukset, joista suurin osa liittyi järjestelmän käytettävyyteen. Koska jo pelkästään järjestelmäarkkitehtuurin ja käyttöliittymän uudistaminen parantavat järjestelmän käytettävyyttä, asiakkaiden vaatimukset ovat vasta toisessa prioriteettiluokassa. Kolmanneksi tärkeimpinä vaatimuksina voidaan pitää uusia lisäarvoa tuottavia ominaisuuksia, jotka uuteen järjestelmään lisätään. Vaatimuksia ei määrällisesti ole paljoa, joten niiden toteuttaminen samanaikaisesti pitäisi olla mahdollista. Priorisointitapana tässä projektissa voidaan pitää haittojen välttämistä ja sidosryhmien henkilökohtaisia mieltymyksiä, painottaen projektin käynnistäjän mielipiteitä. Prioriteettiluokat on kuvattu seuraavassa taulukossa karkealla tasolla.

Taulukko 9. Uuden laatikkojärjestelmän vaatimusten prioriteettiluokat karkealla tasolla

Vaatimus	Tavoite	Prioriteetti
Järjestelmäarkkitehtuurin uudistaminen	Järjestelmän käyttöönoton helpottaminen, käytettävyyden ja toimintavarmuuden parantaminen	1
Asiakasvaatimusten toteuttaminen	Käytettävyyden ja toimintavarmuuden parantaminen	2
Uusien ominaisuuksien lisääminen	Lisäarvon tuottaminen	3

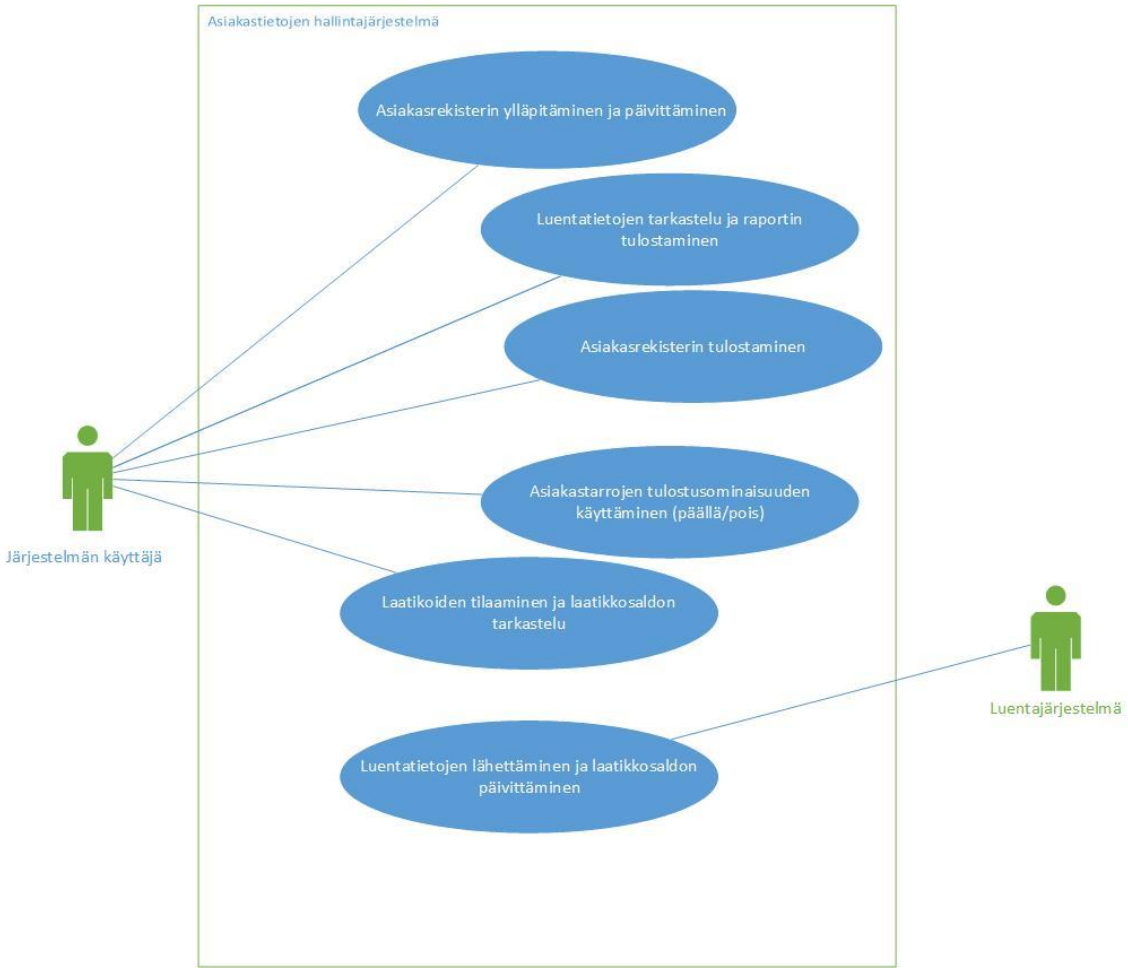
Ohje 6. Käytä vaatimusmäärittelyn tukena UML-kaavioita.

Alfa-yrityksen projektissa UML-kaavioita käytetään uuden järjestelmäkokonaisuuden mallintamiseen. Projektissa UML-kaavioiden tarkoituksena on helpottaa järjestelmän

fyysisten toteuttajien eli koodaajien työtä ja auttaa havainnollistamaan järjestelmäkokonaisuutta graafisesti. UML-mallinnus aloitetaan mallintamalla asiakastietojen hallintajärjestelmä ja luentajärjestelmä käyttötapauskaavioina.

Uudessa asiakastietojen hallintajärjestelmässä järjestelmän käyttäjällä on viisi käyttötapausta. Ensimmäinen käyttötapaus on asiakasrekisterin ylläpitäminen ja päivittäminen. Asiakasrekisterin ylläpito pitää sisällään asiakastietojen lisäämistä, poistamista ja muokkaamista. Toinen käyttötapaus on luentatietojen tarkastelu ja raportin tulostaminen. Järjestelmän avulla asiakas pystyy tarkastelemaan, mitä ja minne laatikoita on lähetetty mihinkin aikaan. Asiakas pystyy myös tulostamaan raportin laatikkotiedoista haluamaltaan aikaväliltä. Kolmas käyttötapaus on asiakasrekisterin tulostaminen. Asiakasrekisteri on tulostettava, jotta laatikoiden luentavaiheessa ennen luennan aloittamista luettavat laatikot saadaan yhdistettyä viivakoodin avulla oikeaan asiakkaaseen. Neljäs käyttötapaus on asiakastarrojen tulostusominaisuuden käyttäminen. Tämä käyttötapaus on hyvin yksikertainen, koska asiakastarrojen tulosominaisuus on joko päällä tai ei. Viidentenä käyttötapauksena on laatikoiden tilaaminen ja laatikkosaldon tarkastelu. Käyttäjä voi tilata laatikoita joko manuaalisesti itse aina halutessaan tai hän voi asettaa saldorajan, joka alittuessaan lähettää automaattisesti uuden laatikkotilauksen. Mikäli saldoraja on käytössä, ei laatikkosaldoa tarvitse välttämättä manuaalisesti seurata.

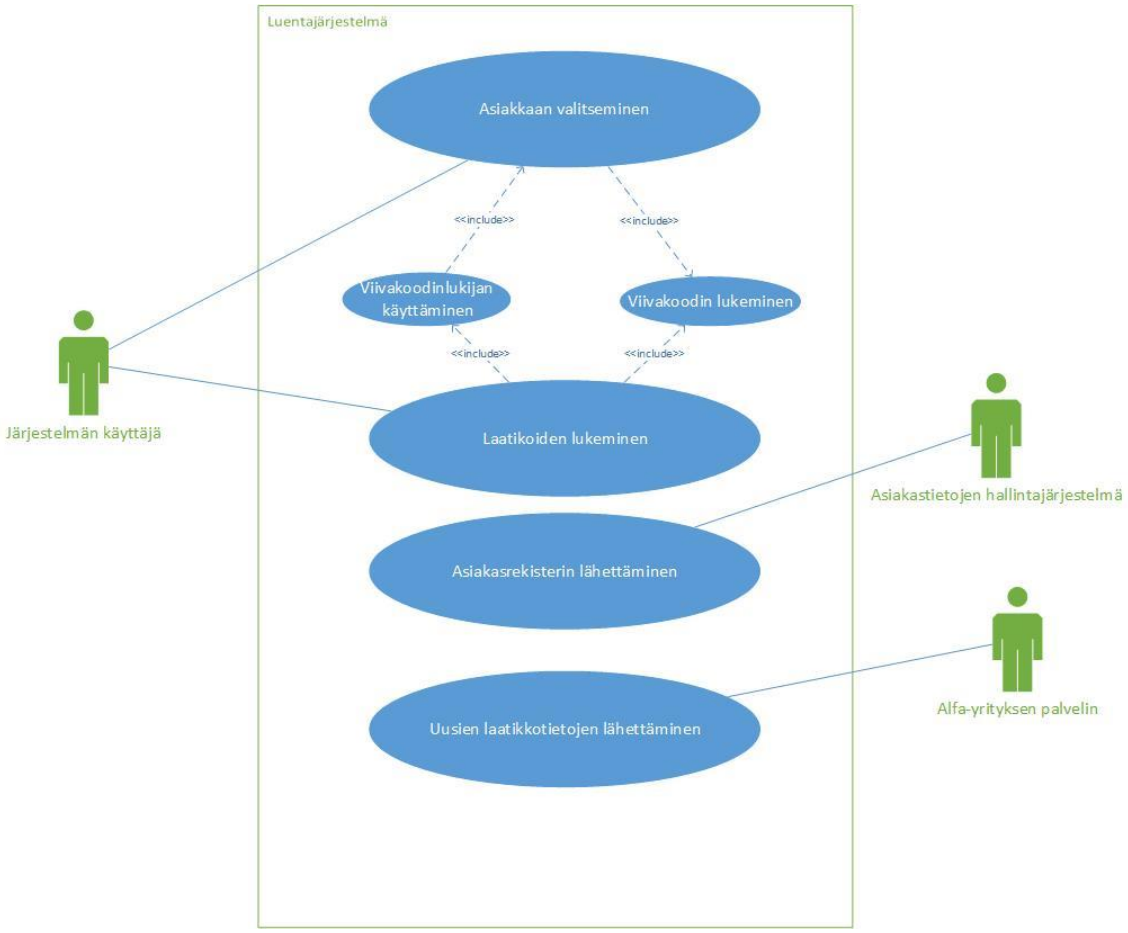
Uusi laatikoiden luentajärjestelmä toimii tiiviisti yhteistyössä asiakastietojen hallintajärjestelmän kanssa, joten myös luentajärjestelmällä on käyttötapaus asiakastietojen hallintajärjestelmässä. Käyttötapaus on luentatietojen lähettäminen ja laatikkosaldon päivittäminen. Aina, kun laatikoita luetaan, tiedot siirtyvät luentajärjestelmästä automaattisesti asiakastietojen hallintajärjestelmään tarkastelua varten. Samoin myös jokainen luentatapahtuma vähentää automaattisesti yrityksen laatikkosaldoa, josta tieto siirtyy samaan tapaan asiakastietojen hallintajärjestelmään. Asiakastietojen hallintajärjestelmän käyttötapauskaavio on esitetty seuraavassa kuviossa.



Kuvio 10. Asiakastietojen hallintajärjestelmän käyttötapauskaavio

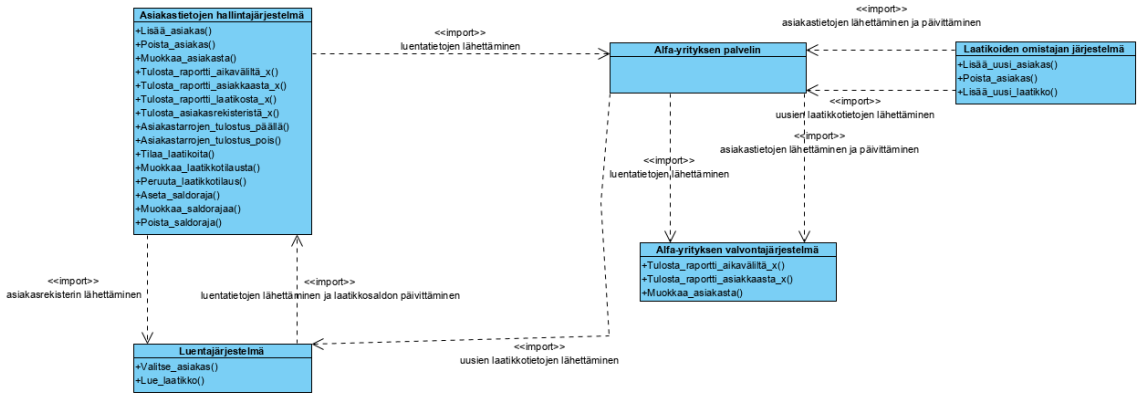
Uudessa luentajärjestelmässä järjestelmän käyttäjällä on ainoastaan kaksi käyttötapausta, sillä tarkoituksena on, että se on mahdollisimman yksinkertainen ja selkeä. Ensimmäinen käyttötapaus on sen asiakkaan valitseminen, jonne laatikoita luetaan. Asiakkaan valitseminen tapahtuu lukemalla kyseisen asiakkaan asiakastietojen hallintajärjestelmästä tulostettu viivakoodi. Asiakkaan valitsemisen jälkeen toisena käyttötapauksena on laatikoiden lukeminen. Laatikoita luetaan lukemalla jokaisen lähetettävän laatikon viivakoodi. Luentatiedot lähtevät jokaisen luennan jälkeen automaattisesti asiakastietojen hallintajärjestelmään. Viivakoodien lukeminen edellyttää viivakoodinlukijan käyttämistä. Lisäksi asiakastietojen hallintajärjestelmällä on käyttötapauksena luentajärjestelmässä asiakasrekisterin lähettäminen. Aina, kun asiakasrekisteriä päivitetään, tiedot lähetetään automaattisesti luentajärjestelmälle. Asiakastietojen hallintajärjestelmän ohella alfa-

yrityksen palvelimella on luentajärjestelmässä käyttötapauksena uusien laatikkotietojen lähettäminen. Kun laatikoiden omistaja lisää uusia laatikoita kiertoon, uudet laatikkotiedot lähetetään alfa-yrityksen palvelimen kautta luentajärjestelmälle, jotta luentajärjestelmä tunnistaa uudet laatikot luennan yhteydessä. Luentajärjestelmän käyttötapauskaavio on esitetty seuraavassa kuviossa.



Kuvio 11. Luentajärjestelmän käyttötapauskaavio

Asiakastietojen hallintajärjestelmän ja luentajärjestelmän käyttötapauskaavioiden lisäksi uutta järjestelmäarkkitehtuuria on hyvä tarkastella kokonaisuutena. Järjestelmäkokonaisuus on kuvattuna seuraavassa kuviossa luokkakaaviona. Luokkakaavio auttaa hahmottamaan, miten järjestelmäkokonaisuuden toiminnot rakentuvat.



Kuvio 12. Luokkakaavio järjestelmäarkkitehtuurista

Luokkakaaviossa jokainen järjestelmäkokonaisuuden osa kuvaa yhtä luokkaa. Kuten jo aikaisemmin todettiin, alfa-yrityksen asiakkaat käyttävät asiakastietojen hallintajärjestelmää omien asiakastietojensa hallintaan. Asiakastietojen hallintajärjestelmä lähettää päivitetyn asiakasrekisterin reaaliaikaisesti luentajärjestelmään aina silloin, kun asiakasrekisteriä muokataan.

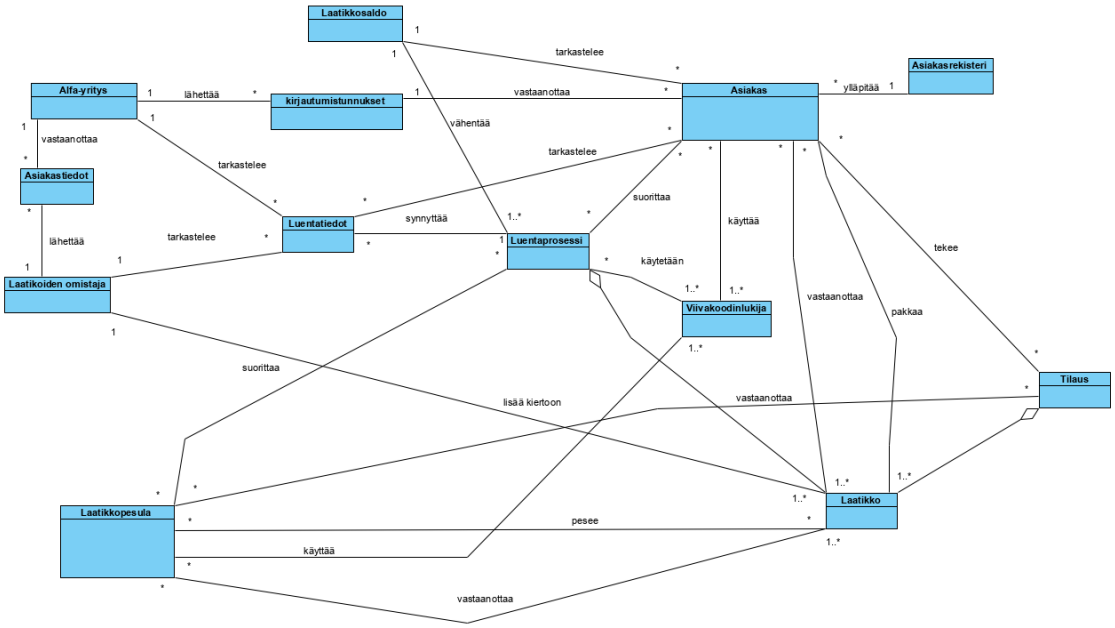
Varsinaista luentajärjestelmää alfa-yrityksen asiakkaat käyttävät ainoastaan kuljetuslaatikoiden lukemiseen. Laatikoiden luenta pitää sisällään asiakkaan valitsemisen ja tämän jälkeen lähetettävien laatikoiden lukemisen. Luentatiedot lähtevät jokaisen luennan jälkeen automaattisesti asiakastietojen hallintajärjestelmään vähentäen samalla yrityksen laatikkosaldoa.

Asiakastietojen hallintajärjestelmästä luentatiedot siirtyvät alfa-yrityksen omalle palvelimelle. Palvelimelta luentatiedot siirtyvät alfa-yrityksen valvontajärjestelmään ja laatikoiden omistajan järjestelmään. Valvontajärjestelmästä alfa-yritys pystyy tarkastelemaan omien asiakkaidensa luentatietoja ja laskutushistoriaa. Omien asiakkaidensa tiedot alfa-yritys saa alfa-yrityksen palvelimelta, jonne tiedot tulevat laatikoiden omistajan järjestelmästä. Kun alfa-yritys saa uuden asiakkaan, pyydetään laatikoiden omistajaa perustamaan uusi asiakas ensin laatikoiden omistajan järjestelmään. Tiedot asiakkaista siirtyvät ja päivittyvät palvelimen kautta reaaliaikaisesti alfa-yrityksen

valvontajärjestelmään. Valvontajärjestelmästä asiakastietoja voi halutessaan muokata. Laatikoiden omistajalla on aina tiedot kaikista asiakkaista, jotka käyttävät omistajan kuljetuslaatikoita, koska laatikoiden omistaja seuraa kaikkia kuljetuslaatikoita. Laatikoiden käyttäminen edellyttää aina sopimuksen sekä laatikoiden omistajan että alfa-yrityksen kanssa.

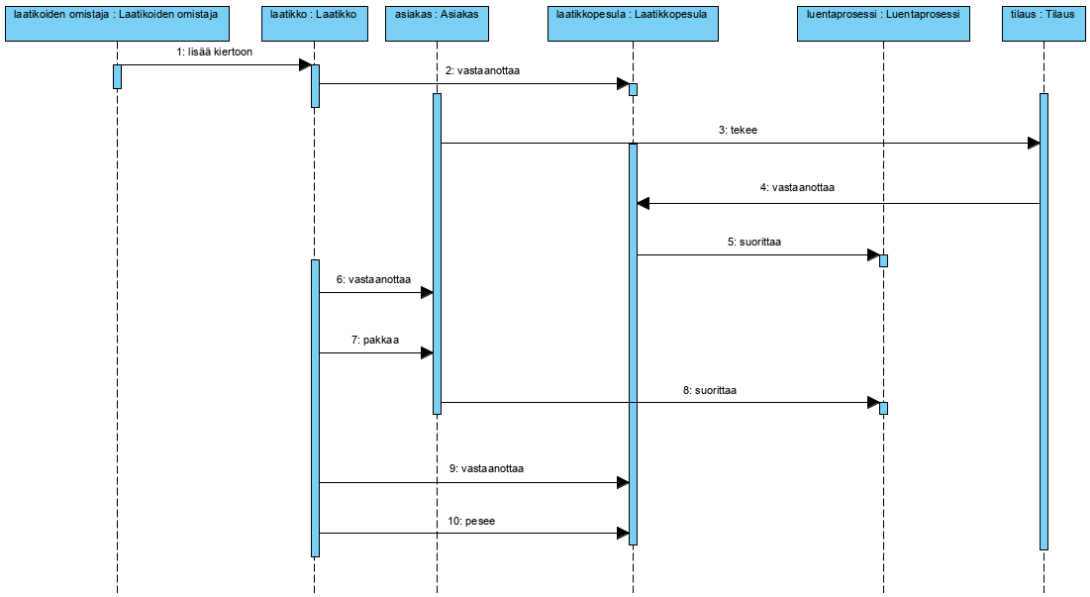
Vastaavalla tavalla uusien laatikoiden lisääminen kiertoon tapahtuu laatikoiden omistajan järjestelmästä. Laatikoiden omistaja lisää uuden laatikon tiedot järjestelmään, jotka lähtevät alfa-yrityksen palvelimen kautta suoraan luentajärjestelmään. Luentajärjestelmän on saatava uusien laatikoiden tiedot, jotta luentajärjestelmä tunnistaa uudet laatikot luennan yhteydessä.

Uutta laatikkojärjestelmää on hyvä tarkastella myös yleisemmällä tasolla. On tärkeää, että järjestelmän koodaajat ymmärtävät järjestelmäarkkitehtuurin ja järjestelmien yksittäisten ominaisuuksien lisäksi yleisellä tasolla uuden järjestelmäkokonaisuuden keskeisimpien käsitteiden väliset riippuvuudet. Käsitteiden väliset riippuvuudet on kuvattu seuraavassa kuviossa luokkakaaviona. Jokaisen käsitteen riippuvuus on selitetty liitteessä 5.



Kuvio 13. Luokkakaavio järjestelmäkokonaisuuden keskeisimpien käsitteiden välisistä riippuvuuksista

Lopuksi on vielä mallinnettava logistiikkaprosessin oleellinen osa, yksittäisen laatikon kierron vaiheet. Laatikon kierron vaiheet on esitetty seuraavassa kuviossa sekvenssikaaviona. Sekvenssikaaviota luetaan ylhäältä alas ja tapahtumat etenevät aikajärjestyksessä.



Kuvio 14. Laatikon kierron vaiheet

Uuden laatikon kierto alkaa siitä hetkestä, kun laatikoiden omistaja lisää tiedot uudesta laatikosta omaan järjestelmäänsä ja laatikkopesula vastaanottaa kyseisen laatikon. Tämän jälkeen asiakas tekee laatikkotilauksen, jonka laatikkopesula vastaanottaa. Tilauksen saavuttua laatikkopesula suorittaa luentaprosessin eli toisin sanoen lähettää laatikon asiakkaalle. Asiakas vastaanottaa ja pakkaa laatikon, jonka jälkeen myös asiakas suorittaa luentaprosessin. Tällöin vastuu laatikosta siirtyy lähtökohtaisesti takaisin laatikkopesulalle, koska tällä hetkellä kaupoissa ei lueta laatikoita. Lopuksi laatikkopesula vastaanottaa laatikot myymälöistä ja kaupoista, jonka jälkeen se pesee ne seuraavaa tilausta varten.

Ohje 7. Dokumentoi vaatimukset selkeään ja ymmärrettävään muotoon.

UML-mallinnuksen lisäksi vaatimukset on dokumentoiva selkeään ja ymmärrettävään muotoon kirjallisesti. Tässä kappaleessa dokumentoidaan asiakastietojen hallintajärjestelmän, luentajärjestelmän ja alfa-yrityksen valvontajärjestelmän toiminalliset vaatimukset omiin erillisiin taulukoihin. Tämän jälkeen dokumentoidaan koko järjestelmäkokonaisuuden ei-toiminnalliset vaatimukset. Toiminnalliset vaatimukset priorisoidaan ja esitellään taulukossa 9 esiteltyjen prioriteettiluokkien mukaisesti. Ensimmäisen prioriteettiluokan vaatimukset liittyvät uuteen järjestelmäarkkitehtuuriin ja järjestelmän nykyisiin välttämättömiin ominaisuuksiin. Toisen prioriteettiluokan vaatimukset ovat asiakkailta tulleita kehitysehdotuksia. Kolmannen prioriteettiluokan vaatimukset ovat uusia lisäarvoa tuottavia palveluita.

Taulukko 10. Asiakastietojen hallintajärjestelmän toiminnalliset vaatimukset

Vaatimus	Prioriteetti
1. Käyttäjien on pystyttävä lisäämään, poistamaan ja muokkaamaan asiakasrekisterin asiakkaita.	1
2. Käyttäjien on pystyttävä tarkastelemaan laatikoiden luentatietoja ja halutessaan tulostamaan raportti luentatiedoista haluamaltaan aikaväliltä.	1

3. Käyttäjien on pystyttävä tulostamaan asiakasrekisteri laatikoiden luentaa varten.	1
4. Järjestelmän on lähetettävä päivitetty asiakasrekisteri reaaliajassa sen muokkaamisen jälkeen luentajärjestelmälle.	1
5. Järjestelmän on lähetettävä luentatiedot reaaliajassa alfa-yrityksen palvelimelle.	1
6. Käyttäjän on pystyttävä tulostamaan raportti myös ainoastaan tietyn asiakkaan luentatiedoista tai tietystä laatikosta haluamaltaan aikaväliltä.	2
7. Asiakasrekisteristä on pystyttävä manuaalisesti valitsemaan asiakkaat, joiden viivakoodit käyttäjä haluaa tulostaa. Käyttäjän ei ole pakko tulostaa koko asiakasrekisteriä eli kaikkia viivakoodeja.	2
8. Asiakasrekisterin asiakkaiden järjestystä on pystyttävä muokkaamaan erilaisin kriteerein, esimerkiksi järjestämällä asiakkaat aakkosjärjestykseen. Tämä ominaisuus on ilmaistava asiakkaalle selkeästi erillisen painikkeen tai huomion avulla.	2
9. Asiakasrekisteristä on pystyttävä hakemaan asiakkaita esimerkiksi nimen, osoitteen, postinumeron ja puhelinnumeron avulla.	2
10. Järjestelmästä on pystyttävä etsimään tiettyjä laatikoita asiakkaan nimen, lähetetyn paikan ja laatikkonumeron avulla, jotta laatikkotietojen tarkastelu on helpompaa.	2
11. Järjestelmässä on oltava laatikoiden tilaamisominaisuus. Laatikoita voi tilata manuaalisesti tai käyttäjä voi vaihtoehtoisesti asettaa saldorajan, joka alittuessaan lähettää automaattisesti uuden laatikkotilauksen. Laatikkotilauksia on pystyttävä myös muokkaamaan ja perumaan. Lisäksi saldorajaa on pystyttävä muuttamaan tai tarvittaessa poistamaan.	3
12. Järjestelmässä on oltava asiakastarrojen tulostusominaisuus. Ominaisuus on joko päällä tai pois päältä. Mikäli asiakas haluaa käyttää ominaisuutta, on järjestelmä yhdistettävä asiakkaan omaan tarrakoneeseen. Käyttäjä määrittelee itse järjestelmän asiakasrekisteriin, mitkä tarratiedot kunkin asiakkaan kohdalla automaattisesti tulostetaan.	3

Taulukko 11. Luentajärjestelmän toiminnalliset vaatimukset

Vaatimus	Prioriteetti
13. Käyttäjän on pystyttävä valitsemaan laatikoiden luentaa varten oikea asiakas lukemalla asiakastietojen hallintajärjestelmästä tulostettu viivakoodi viivakoodinlukijalla. Käyttäjän on voitava varmistua siitä, että asiakkaan valitseminen onnistuu eli käyttäjän on nähtävä näyttöpäätteeltä valitun asiakkaan asiakastiedot (liite 4).	1
14. Käyttäjän on pystyttävä lukemaan laatikoita viivakoodinlukijalla ja jokaisen lukutapahtuman yhteydessä järjestelmän on lähetettävä luentatiedot reaaliaikaisesti asiakastietojen hallintajärjestelmään.	1
15. Käyttäjän on saatava luennan yhteydessä varmuus siitä, että luenta onnistuu. Viivakoodinlukijan on annettava äänimerkki ja käyttäjän on nähtävä käytettävältä näyttöpäätteeltä kyseisen luentakerran laatikkonumerot (liite 4).	1
16. Käyttäjän on pystyttävä poistamaan luentatapahtumia luennan yhteydessä tilanteessa, jossa käyttäjä lukee vahingossa väärän laatikon. Luentatietojen poistamisen tulee olla mahdollista vähintään siihen saakka, kunnes seuraava asiakas valitaan luentaa varten.	1
17. Luentajärjestelmässä on oltava laatikon hakutoiminto. Hakutoiminto on tarkoitettu yksittäisen luentakerran laatikoiden etsimistä ja poistamista varten.	2
18. Jokaisen luentatapahtuman on automaattisesti vähennettävä laatikkosaldoa, jotta asiakastietojen hallintajärjestelmän laatikkosaldo ja saldoraja näkyvät ja toimivat oikein.	3

Taulukko 12. Alfa-yrityksen valvontajärjestelmän toiminnalliset vaatimukset

Vaatimus	Prioriteetti
19. Järjestelmän on saatava kaikki tiedot alfa-yrityksen omista asiakkaista palvelimen kautta laatikoiden omistajan järjestelmästä.	1

20. Järjestelmän on saatava luentatiedot palvelimelta.	1
21. Järjestelmästä on pystyttävä tarkastelemaan laatikoiden luentatietoja, luentahistoriaa ja laskutushistoriaa. Laskutushistorialla tarkoitetaan sitä, miten paljon asiakkaat ovat laatikoiden käytöstä velkaa alfa-yritykselle.	1
22. Järjestelmästä on pystyttävä tulostaa haluamastaan asiakkaasta luentatiedot haluamaltaan aikaväliltä.	1

Taulukko 13. Järjestelmäkokonaisuuden ei-toiminnalliset vaatimukset

Vaatimus	Prioriteetti
23. Jokaisen järjestelmäkokonaisuuden osan on oltava itsenäisesti mobiiliverkossa, jotta tiedot voivat kulkea reaaliajassa järjestelmien välillä.	1
24. Verkkoyhteyden on oltava luotettava eivätkä luentatiedot saa kadota yhteyshäiriön aikana.	1
25. Luentajärjestelmän käyttäjän on saatava ilmoitus siitä, jos järjestelmä on virhetilassa. Järjestelmän on palauduttava virhetilasta nopeasti.	1
26. Usean käyttäjän on pystyttävä käyttämään laatikkojärjestelmää (asiakastietojen hallintajärjestelmä + luentajärjestelmä) yhtäaikaisesti. Jokaisen alfa-yrityksen asiakkaan on pystyttävä käyttämään asiakastietojen hallintajärjestelmää samanaikaisesti (tuki noin 100-200 käyttäjälle). Jokaisen asiakkaan on pystyttävä lukemaan laatikoita usealla laitteella samanaikaisesti (luentajärjestelmään tarvitaan tuki noin 500 käyttäjälle).	1
27. Järjestelmäarkkitehtuuri edellyttää, että jokainen asiakas saa yrityskohtaiset kirjautumistunnukset palvelun käyttöönoton yhteydessä, joilla pääsee käyttämään asiakastietojen hallintajärjestelmää ja luentajärjestelmää.	1
28. Laatikoiden luennan on onnistuttava viivakoodinlukijoilla.	1
29. Asiakastietojen hallintajärjestelmää ja alfa-yrityksen valvontajärjestelmää on pystyttävä käyttämään selaimella.	1

30. Luentajärjestelmän tulee olla Android-pohjainen sovellus, jonka voi ladata tablettiin tai puhelimeen. Puhelimen yhdistämisen fyysisesti viivakoodinlukijaan tulee olla mahdollista, jolloin asiakas ei välttämättä tarvitse erillistä näyttöpäätettä luentajärjestelmää varten.	1
31. Viivakoodinlukijan täytyy olla yhdistettävissä luentajärjestelmään langattomasti.	1
32. Käyttöliittymien on oltava helposti opittavissa. Käyttäjien tulee osata käyttää järjestelmiä ilman erillistä ohjeistusta. Käyttöliittymien sisäisten ohjeiden on riitettävä järjestelmien käyttöön. Toisin sanoen käyttöönoton on oltava helppoa.	1
33. Käyttöliittymien on oltava helposti muistettavissa. Käyttäjien on muistettava pitkänkin tauon jälkeen, miten järjestelmiä käytetään.	1
Käyttöliittymien on oltava niin yksinkertaisia ja selkeitä, etteivät käyttäjät tee virheitä.	1
34. Käyttöliittymien tulee mahdollistaa järjestelmien tehokas käyttö. Käyttäjän on päästävä nopeasti tavoitteisiinsa. Järjestelmien käyttämisessä ei saa ilmaantua viivettä eli käyttäjän on aina tiedettävä, miten hän pääsee nopeasti haluamaansa lopputulokseen.	1
35. Laatikoiden tilaamisominaisuus edellyttää yhteensopivuutta laatikoiden omistajan järjestelmän kanssa.	3
36. Asiakastarrojen tulostusominaisuus edellyttää yhteensopivuutta asiakkaiden tarrakoneiden kanssa.	3

Ohje 8. Hyväksytä vaatimusdokumentti sidosryhmillä.

Vaatimusdokumentti hyväksytetään vähintään alfa-yrityksellä ja laatikoiden omistajalla. Tässä vaiheessa ei ole varmuutta, onko vaatimusdokumentti tarpeellista hyväksyttää myös alfa-yrityksen asiakkailla. Asiakkaat ovat jo tähän mennessä ilmaisseet vaatimuksensa selkeästi ja ne on otettu tässä vaatimusmäärittelyssä huomioon. Kaikki ilmi tulleet asiakkaiden vaatimukset toteutetaan.

5.7 Artefaktin arviointi

Artefaktin tavoitteena oli antaa edellytykset tehokkaan, tuottavan ja ketterän logistisen laatikkojärjestelmän vaatimusmäärittelylle. Tutkimuksessa luotiin artefakti, metatason ohjeistus, jonka tarkoituksena oli toimia laatikkojärjestelmän suunnittelijoiden ohjeistuksena ja tukena ja auttaa heitä ymmärtämään, mitä laatikkojärjestelmän vaatimusmäärittelyssä tulisi ottaa huomioon ja miten se tulisi toteuttaa. Ohjeistusta sovellettiin alfa-yrityksen laatikkojärjestelmän kehitysprojektin vaatimusmäärittelyssä. Ohjeistus soveltui alfa-yrityksen projektiin hyvin, joten tutkimuksessa luotua metatason ohjeistusta voidaan tämän tutkimuksen perusteella pitää geneerisenä työkaluna, jota myös muut toimialan yritykset voivat hyödyntää. Artefaktin käyttöä tulisi kuitenkin soveltaa myös muiden logististen laatikkojärjestelmien vaatimusmäärittelyissä, jotta artefaktin yleistettävyyttä, luotettavuutta ja relevanttiutta voitaisiin parantaa.

Tutkimuksessa luotu artefakti perustuu pääosin aihepiirin tieteelliseen kirjallisuuteen. Ohjeistus on hyvin geneerinen ja se olisi sovellettavissa pienillä muutoksilla lähes minkä tahansa tietojärjestelmän vaatimusmäärittelyyn. Kuitenkin ohjeet 3 ja 4 muotoutuivat lopulliseen artefaktiin käytännön aineiston pohjalta ja ne myös erottavat ohjeistuksen perinteisestä vaatimusmäärittelyprosessista. Tutkimuksen aikana huomattiin, että laatikkojärjestelmien suunnittelussa on erityisen tärkeää ottaa huomioon lisäarvon tuottaminen (ohje 3) ja järjestelmän käytettävyys (ohje 4). Seuraavissa kappaleessa arvioidaan, miten ohjeistus antaa edellytykset tehokkaan, tuottavan ja ketterän logistisen laatikkojärjestelmän vaatimusmäärittelylle.

Logistiset laatikkojärjestelmät ovat usein ominaisuuksiltaan hyvin pelkistettyjä ja yksinkertaisia. Tästä syystä on tärkeää huomioida, miten järjestelmä saadaan erottumaan muista vastaavanlaisista järjestelmistä ja miten se luo kilpailuetua. Lisäarvoa tuottavilla ominaisuuksilla ja palveluilla voidaan luoda mukautuvuutta ja joustavuutta eli ketteryyttä asiakkaiden tarpeisiin, mikä luonnollisesti luo myös kilpailuetua. Asiakkaiden työhön saattaa liittyä muita päivittäisiä toimintoja ja aktiviteetteja, joiden yhdistäminen laatikkojärjestelmään luo ketteryyttä. Laatikkojärjestelmissä ketteryyden mahdollistaa

läpinäkyvä ja reaaliaikainen tietovirta. Alfa-yrityksen uuden laatikkojärjestelmän vaatimusmäärittelyssä ketteryys näkyy muun muassa uutena automaattisena laatikoiden tilaamisominaisuutena, laatikkosaldon seuraamismahdollisuutena ja luenta- ja asiakastietojen reaaliaikaisena päivittymisenä. Ketteryys itsessään parantaa myös työn tuottavuutta ja tehokkuutta, kun se vähentää ylimääräisen työn määrää.

Vastaavasti logistisilla laatikkojärjestelmillä on usein hyvin laaja käyttäjäkunta erilaisilla tietoteknisillä taidoilla. Tällaisissa tilanteissa käytettävyyden huomioiminen on erityisen tärkeää. Käyttöliittymien tulee olla helppokäyttöisiä, selkeitä ja yksinkertaisia, jotta jokainen yrityksen työntekijä pystyy käyttämään laatikkojärjestelmää tarvittaessa ongelmitta. Tämän lisäksi lisäämällä mukautuvuutta ja joustavuutta käyttöliittymiin, esimerkiksi lisäämällä käyttäjän mahdollisuuksia etsiä, tarkastella ja muokata asiakas- ja laatikkotietoja helpommin, parannetaan laatikkojärjestelmien ketteryyttä, tuottavuutta ja tehokkuutta.

Ohjeistus pyrkii myös kokonaisuutena antamaan edellytykset ketterän, tuottavan ja tehokkaan laatikkojärjestelmän vaatimusmäärittelylle. Ohjeistuksessa korostuu tiivis yhteistyö kaikkien järjestelmään liittyvien sidosryhmien ja järjestelmän tulevien käyttäjien kanssa. Sidosryhmien toiveet ja vaatimukset pyritään huomioimaan kaikissa vaatimusmäärittelyn vaiheissa. Kun sidosryhmien vaatimukset huomioidaan jokaisessa vaatimusmäärittelyn vaiheessa, ei laatikkojärjestelmän vaatimusmäärittelystä todennäköisesti jää puuttumaan tärkeitä tai olennaisia ominaisuuksia, joiden lisääminen myöhemmin tulisi kalliiksi. Ohjeistuksessa huomioidaan erityisesti myös järjestelmän toteuttajat eli koodaajat. Ohjeistus pyrkii luomaan koodareille valmiin ohjeen selkeän ja ymmärrettävän dokumentaation avulla. Dokumentaation tueksi järjestelmä ohjeistetaan mallintamaan UML-kaavioiden avulla, jotka kuvaavat graafisesti järjestelmän toimintaa. Kun järjestelmän koodaajat saavat selkeän vaatimusdokumentin, jonka pohjalta laatikkojärjestelmä toteutetaan, vältytään todennäköisesti hintavilta väärinymmärryksiltä. Kun vaatimusdokumentti on selkeä, järjestelmästä tulee suunnitelman ja ohjeiden mukainen.

Vaikka tutkimuksessa ei tuotettu varsinaista uutta tietoa tai teoriaa, tuottaa artefakti kuitenkin selkeää tutkimuskontribuutiota. Tutkimuksessa vaatimusmäärittelyn teoria ja laatikkojärjestelmien ominaispiirteet on yhdistetty ja niistä on luotu selkeä ja valmis uusi metatason ohjeistus laatikkojärjestelmän suunnittelijoille. Tuotettua ohjeistusta voidaan pitää lisäyksenä aiempaan tietopohjaan. Kappaleessa 6.2 tutkimusta arvioidaan vielä kokonaisvaltaisemmin suunnittelutieteellisenä tutkimuksena.

5.8 **Viestintä**

Suunniteltu artefakti esitellään alfa-yritykselle ja Vaasan yliopiston pro gradu -tutkielmaseminaarissa muille maisteriopiskelijoille. Lisäksi koko tutkielma julkaistaan Vaasan yliopiston tietokannassa.

6 Diskussio

Tässä kappaleessa tutkimusprosessista tehdään ensin yhteenveto. Tämän jälkeen esitellään tutkimuksen johtopäätökset, jossa arvioidaan tulosten merkitystä. Tämän jälkeen tutkimusta arvioidaan suunnittelutieteellisenä tutkimuksena. Lopuksi kappaleessa esitellään tutkimuksen rajoitukset ja jatkotutkimusaiheet.

6.1 Yhteenveto ja johtopäätökset

Tutkimuksen tavoitteena oli selvittää, millainen on metatason ohjeistus, jonka avulla pystytään luomaan vaatimusmäärittely, jonka pohjalta voidaan toteuttaa mahdollisimman tehokas, tuottava ja ketterä logistinen laatikkojärjestelmä. Tutkimus toteutettiin suunnittelutieteellisenä tutkimuksena ja se seurasi suunnittelutieteellisen tutkimuksen DSRM-prosessimallia. Artefaktin tietopohja koostui olemassa olevasta ohjelmistotuotannon, vaatimusmäärittelyn ja logistiikan tietojärjestelmien teoriasta. Lisäksi artefaktin tietopohjaan kuului kaikki se käytännön aineisto, jota alfa-yritykselle tehtävää uuden laatikkojärjestelmän vaatimusmäärittelyä varten kerättiin. Aineistoa kerättiin kyselyiden ja haastatteluiden avulla ja alfa-yrityksen järjestelmän käyttöä havainnoimalla.

Tutkimuksessa luotiin artefakti, metatason ohjeistus, joka toimii laatikkojärjestelmän suunnittelijoiden ohjeistuksena ja tukena ja se auttaa heitä ymmärtämään, mitä laatikkojärjestelmän vaatimusmäärittelyssä tulisi ottaa huomioon ja miten se tulisi toteuttaa. Artefaktin relevanttiutta testattiin demonstroimalla se alfa-yrityksen uuden laatikkojärjestelmän vaatimusmäärittelyssä. Metatason ohjeistus toimi hyvänä ohjeena alfa-yrityksen vaatimusmäärittelyprojektissa, joten tutkimuksessa luotua metatason ohjeistusta voidaan pitää geneerisenä työkaluna, jota kaikki toimialan yritykset voivat hyödyntää toimeksiantajan lisäksi. Täten artefakti hyödyttää toimeksiantajan projektin lisäksi koko toimialaa. Toisin sanoen tutkimuksessa luotiin suunnittelutietämystä, jota ammattilaiset voivat käyttää tulevien konstruointi- ja suunnitteluongelmien ratkaisemisessa. Artefaktin antamat edellytykset tehokkaan, tuottavan ja ketterän logistisen laatikkojärjestelmän

vaatimusmäärittelylle perustuvat lisäarvon tuottamiseen, käytettävyyden huomioimiseen, tiiviiseen yhteistyöhön sidosryhmien kanssa ja selkeään dokumentaatioon.

6.2 Arviointi suunnittelutieteellisenä tutkimuksena

Tässä kappaleessa artefaktin toteutusprosessia arvioidaan kokonaisuutena suunnittelutieteellisenä tutkimuksena. Toteutusprosessi arvioidaan tarkastelemalla sitä Hevnerin ja muiden (2004) suunnittelutieteellisen tutkimuksen seitsemän ohjeen kautta. Seuraavassa taulukossa perustellaan, miten toteutusprosessi noudatti kutakin ohjetta.

Taulukko 14. Artefaktin arviointi suunnittelutieteellisenä tutkimuksena

Ohje	Arviointi artefaktin toteutusprosessissa
1. Suunnittelutieteellisessä tutkimuksessa on tuotettava toteuttamiskelpoinen artefakti. Artefakti voi olla käsitteistö (engl. construct), malli, metodi tai toteutus.	Artefakti toteutettiin seuraamalla DSRM-prosessimallia. Tutkimustuloksena oli metatason ohjeistus, joka toimii laatikkojärjestelmän suunnittelijoiden tukena ja ohjeistuksena ja auttaa heitä ymmärtämään, mitä laatikkojärjestelmän suunnittelussa tulisi ottaa huomioon ja miten se tulisi toteuttaa. Valmista ohjeistusta voidaan pitää mallina tai metodina. Metatason ohjeistus on suunnittelutieteellisestä näkökulmasta metodi, koska se on joukko tehtävän suorittamiseen käytettäviä askelia.
2. Suunnittelutieteellisessä tutkimuksessa tavoitteena on kehittää teknologiapohjaisia ratkaisuja tärkeisiin ja merkityksellisiin liiketoimintaongelmiin.	Artefakti eli metatason ohjeistus luotiin merkityksellisen ongelman ratkaisemiseksi. Logistisille laatikkojärjestelmille, erityisesti elintarviketeollisuudessa, ongelmana ovat useat rajapinnat ja niissä

	tapahtuvien toimenpiteiden läpinäkymättömyys ja suhteellisen suuri työmäärä. Lisäksi nopeat toimitusaikavaatimukset yhdessä kysynnän nopeiden muutosten kanssa aiheuttavat tietojärjestelmille uusia vaatimuksia. Artefakti pyrkii ratkaisemaan ongelmat luomalla tehokkuutta, tuottavuutta ja ketteryyttä kannustamalla lisäarvon tuottamiseen, käytettävyyden huomioimiseen, tiiviiseen yhteystyöhön sidosryhmien kanssa ja vaatimusten selkeään dokumentaatioon.
3. Artefaktin hyödyllisuus, laatu ja tehokkuus on täsmällisesti havainnollistettava hyvin toteutetuilla arviointimenetelmillä.	Artefaktin hyödyllisyys, laatu ja tehokkuus havainnollistettiin käyttämällä sitä alfa-yrityksen uuden laatikkojärjestelmän vaatimusmäärittelyn ohjeistuksena ja runkona. Artefakti soveltui alfa-yrityksen kehitysprojektiin hyvin. Kappaleessa 5.7 artefaktia arvioitiin sen perusteella, miten ohjeistuksen avulla luotiin tehokkuutta, tuottavuutta ja ketteryyttä alfa-yrityksen uuden laatikkojärjestelmän vaatimusmäärittelyssä.
4. Tehokkaan suunnittelutieteellisen tutkimuksen tulee tuottaa selkeää ja todennettavissa olevaa kontribuutiota joko suunnitellun artefaktin, perustietämyksen ja/tai metodologioiden alueilla.	Tutkimuksessa vaatimusmäärittelyn teoria ja laatikkojärjestelmien ominaispiirteet yhdistettiin ja niistä luotiin selkeä ja valmis uusi metatason ohjeistus laatikkojärjestelmän suunnittelijoille. Tuotettua ohjeistusta voidaan pitää lisäyksenä aiempaan tietopohjaan.

5. Suunnittelutieteellinen tutkimus perustuu tarkkojen menetelmien soveltamiseen artefaktin rakentamisessa ja arvioinnissa.	Artefakti toteutettiin ja rakennettiin pääosin vaatimusmäärittelyn, ohjelmistotuotannon, logistiikan tietojärjestelmien ja suunnittelutieteellisen tutkimuksen teorian pohjalta. Artefaktin toteutusprosessi seurasi suunnittelutieteellisen tutkimuksen DSRM-prosessimallia.
6. Tehokkaan artefaktin toteutus vaatii saatavilla olevien keinojen hyödyntämistä tavoitteiden saavuttamiseksi. Samalla on kuitenkin noudatettava ongelmaympäristössä vallitsevia lakeja.	Teorian lisäksi artefaktin rakenteeseen ja sen lopulliseen sisältöön vaikutti käytännön aineisto, jota alfa-yritykselle tehtävää vaatimusmäärittelyä varten kerättiin. Ohjeistuksen demonstroinnin avulla varmistettiin, että artefakti soveltuu kohdeympäristöön.
7. Suunnittelutieteellisen tutkimuksen tulokset on tehokkaasti esitettävä sekä teknologia- että johtamiskeskeisille yleisöille.	Suunniteltu artefakti esiteltiin alfa-yritykselle ja Vaasan yliopiston pro gradu -tutkielmaseminaarissa muille maisteriopiskelijoille. Lisäksi koko tutkielma julkaistaan Vaasan yliopiston tietokannassa.

6.3 **Rajoitukset ja jatkotutkimusaiheet**

Artefaktin relevanttiutta testattiin käyttämällä sitä alfa-yrityksen vaatimusmäärittelyprojektin ohjeistuksena. Demonstroinnin perusteella artefaktia voidaan pitää geneerisenä ohjeistuksena, jota voidaan soveltaa myös muissa samantyyppisissä projekteissa. Kuitenkaan tutkimuksessa tehdyn demonstroinnin perusteella luotua artefaktia ei voida pitää absoluuttisena totuutena tai täydellisenä ohjeena. Lopulliseen artefaktiin vaikutti vahvasti alfa-yrityksen oma vaatimusmäärittelyprojekti, koska artefaktin käytännön ympäristö koostuu alfa-yrityksestä ja sen sidosryhmistä. Artefaktin suunnittelu aloitettiin alfa-

yrityksen toimeksiannon vuoksi. Lisäksi artefaktin lopullinen rakenne on tutkijan subjektiivinen näkemys. Toisen tutkijan toimesta tehty vastaavanlainen artefakti voisi näyttää hyvinkin erilaiselta. Tutkijalla ei myöskään ollut aikaisempaa kokemusta vastaavanlaisista tietojärjestelmäprojekteista, joten myös kokemattomuudella voi olla vaikutusta tutkimustuloksiin ja niiden luotettavuuteen.

Artefaktin yleistettävyyttä, luotettavuutta ja relevanttiutta voitaisiin parantaa hyödyntämällä sitä myös muiden logististen laatikkojärjestelmien vaatimusmäärittelyprojekteissa. Näin artefaktista saataisiin monipuolisempi ja luotettavampi kuva. Jatkotutkimusaiheina voisi olla myös artefaktin avustuksella luotujen järjestelmien tutkiminen niiden käyttöönoton jälkeen. Jatkotutkimusten avulla saataisiin selvyys siitä, tuliko artefaktin perusteella suunnitelluista logistisista laatikkojärjestelmistä oikeasti tuottavia, tehokkaita ja ketteriä. Kaikkein konkreettisin jatkotutkimusaihe olisi tutkia tämän tutkimuksen alfa-yrityksen laatikkojärjestelmää sen käyttöönoton jälkeen.

Lähteet

Aaltola, J. & Valli, R. (2007). *Ikkunoita tutkimusmetodeihin: 1, Metodin valinta ja aineiston keruu: virikkeitä aloittelevalle tutkijalle* (2. painos). Jyväskylä: PS-kustannus.

Ala-Mutka, J., Talvela, J. & Talvela, E. (2004). *Tee asiakassuhteista tuottavia: asiakaslähtöinen liiketoiminnan ohjaus.* Helsinki: Talentum.

Ballou, R. (2004). *Business logistics / supply chain management: planning, organizing, and controlling the supply chain* (5. painos). New Jersey: Prentice hall.

Baxter, D., Goffin, K. & Szwejczewski M. (2014). The Repertory Grid Technique as a Customer Insight Method. *Research-Technology Management*, 57(4), 35-42. https://doi.org/10.5437/08956308x5704229

Beesley, A. (1996). Time compression in the supply chain. *Industrial Management & Data Systems*, 96(2), 12-16. https://doi.org/ 10.1108/02635579610112606

Brooks, F. (1986). No Silver Bullet: Essence and Accident in Software Engineering. University of North Carolina at Chapel Hill.

Christopher, M. (1998). *Logistics and supply chain management: strategies for reducing cost and improving service* (2. painos). London: Financial Times, Prentice Hall

Chuan, H. & Mussbacher, G. (2016). Model-Driven Engineering and Elicitation Techniques: A Systematic Literature Review. *IEEE 24th International Requirements Engineering Conference Workshops*. https://doi.org/10.1109/rew.2016.041

Digitaalinen Helsinki (2019). Vaatimukset (perinteinen hanke). Noudettu 2019-10-14 osoitteesta https://digi.hel.fi/kehmet/menetelmalaari/vaatimukset/

Eskola, J. & Suoranta, J. (2000). *Johdatus laadulliseen tutkimukseen* (4. painos). Tampere: Vastapaino.

Firesmith, D. (2004). Prioritizing requirements. *Journal of Object Technology*, 3(8), 35-47. https://doi.org/ 10.5381/jot.2004.3.8.c4

Fowler, M. & Scott, K. (2002). *UML*. Jyväskylä: Docendo.

Ghezzi, C., Jazayezi, M. & Mandrioli, D. (2003). *Fundamentals of Software Engineering* (2. painos). New Jersey: Upper Saddle River.

Haapanen, M. (1993). *Yritysjohdon logistiikka*. Hämeenlinna: Karisto.

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

Haikala, I. & Märijärvi, J. (2004). *Ohjelmistotuotanto*. Helsinki: Talentum.

Helo, P., Tuominen, T., Sandvik, K. & Tukeva, P. (2007). *MLOG: informaatioteknologian käyttö toimitusketjun hallinnan kehittämiseksi.* Vaasa: Vaasan yliopisto.

Hevner, A., March, S., Park, J. & Ram, S. (2004). Design science in Information Systems research. *Mis Quarterly*, 28(1), 75-105. https://doi.org/10.2307/25148625

Handlaamo (2018). Top 5 käyttäjäkeskeisen suunnittelun menetelmät. Noudettu 2019-10-24 osoitteesta https://handlaamo.fi/top-5-kayttajakeskeisen-suunnittelun-menetelmat/

Hokkanen, S., Luukkainen M. & Karhunen, J. (2011). *Johdatus logistiseen ajatteluun* (6. painos). Kangasniemi: Sho Business Development.

International Organization for Standardization (2011). *Systems and software engineering — Life cycle processes — Requirements engineering.* (Standard No. 29148). Retrieved from https://doi.org/10.1109/IEEESTD.2011.6146379

Jyväskylän yliopiston Koppa (2019). Haastattelut. Noudettu 2019-10-24 osoitteesta https://koppa.jyu.fi/avoimet/hum/menetelmapolkuja/menetelmapolku/aineistonhankintamenetelmat/haastattelut

Järvinen, P. (2006). Onko innovaatioiden suunnittelu tiedettä? Noudettu 2019-11-27 osoitteesta http://www.sytyke.org/lehtiarkisto/kirj/st20062/ST062-25A.pdf

Järvinen, P. & Järvinen, A. (2011). *Tutkimustyön metodeista*. Tampere: Opinpajan kirja.

Karrus, K. (2001). *Logistiikka* (2. painos). Helsinki: WSOY.

Kinnunen, R. & Sinivuori, E. (2004). *Palvelujen suunnittelu*. Helsinki: WSOY.

Kosola, J. (2013). *Vaatimustenhallinnan opas*. Tampere: Juvenes Print.

Kotonya, G. & Sommerville, I. (1998). *Requirements engineering: processes and techniques.* Chichester: Wiley cop.

Laplante, P. (2014). *Requirements engineering for software and systems* (2. painos). Boca Raton: CRC/Taylor & Francis cop.

March, S. T. & Smith, G. F. (1995). Design and natural science research on information technology. *Decision Support Systems,* 15(4), 251-266. https://doi.org/10.1016/0167-9236(94)00041-2

Nielsen, J. (2012). Usability 101: Introduction to Usability. Noudettu 2020-2-1 osoitteesta https://www.nngroup.com/articles/usability-101-introduction-to-usability/

Nuseibeh, B. & Easterbrook S. (2000). Requirements Engineering: A Roadmap. *Proceedings of the Conference on The Future of Software Engineering*, 35-46. https://doi.org/10.1145/336512.336523

Nwokeji, J., Clark, T. & Barn (2013). Towards a comprehensive Meta-Model for KAOS. *3rd International Workshop on Model-Driven Requirements Engineering*. https://doi.org/10.1109/MoDRE.2013.6597261

Paakki, J. (2011). Ohjelmistojen vaatimusmäärittely. Noudettu 2019-10-16 osoitteesta https://www.cs.helsinki.fi/u/paakki/Vaatimus-11-Luentokalvot-1.pdf

Peffers, K., Tuunanen, T., Rothenberger, M. T. & Chatterjee, S. (2007). A Design Science Research Methodology for Information System Research. *Journal of Management Information Systems*, 24(3), 45-77. https://doi.org/10.2753/MIS0742-1222240302

Pohjonen, R. (2002). *Tietojärjestelmän kehittäminen* (1. painos). Jyväskylä: Docendo

Prasad, S. & Tata, J. (2000). Information investment in supply chain management. *Logistics Information Management,* 13(1), 33-38. https://doi.org/10.1108/09576050010306378

Pressman, R. (2000). *Software Engineering: A Practitioner's Approach* (5. painos). London: McGraw-Hill cop.

Sinkkonen, I., Nuutila, E. & Törmä S. (2009). *Helppokäyttöisen verkkopalvelun suunnittelu*. Helsinki: Tietosanoma Oy.

Sommerville, I. (2011). *Software engineering* (9. painos). Boston: Pearson cop.

Taina, J. (2005). Ohjelmistotuotanto, vaatimusanalyysi. Noudettu 2019-10-20 osoitteesta https://www.cs.helsinki.fi/u/taina/ohtu/k-2005/2luku4.pdf

Taina, J. (2006). Ohjelmistotuotanto. Noudettu 2019-10-07 osoitteesta https://www.cs.helsinki.fi/u/taina/ohtu/k-2006/pdf/Ohjelmistotuotanto-2006-2.pdf

Tietotekniikan termitalkoot (2000). Käyttöliittymä. Noudettu 2020-2-1 osoitteesta http://www.tsk.fi/tsk/termitalkoot/fi/node/266

Tuulaniemi, J. (2011). *Palvelumuotoilu*. Helsinki: Talentum.

Tuunanen, T. (2005). *Requirements for wide-audience end-users*. Väitöskirja, Aalto-yliopisto, Helsinki.

Van Aken, J. (2004). Management Research Based on the Paradigm of the Design Sciences: The Quest for Field-Tested and Grounded Technological Rules. *The Journal of Management Studies*, 219-246. https://doi.org/10.1111/j.1467-6486.2004.00430.x

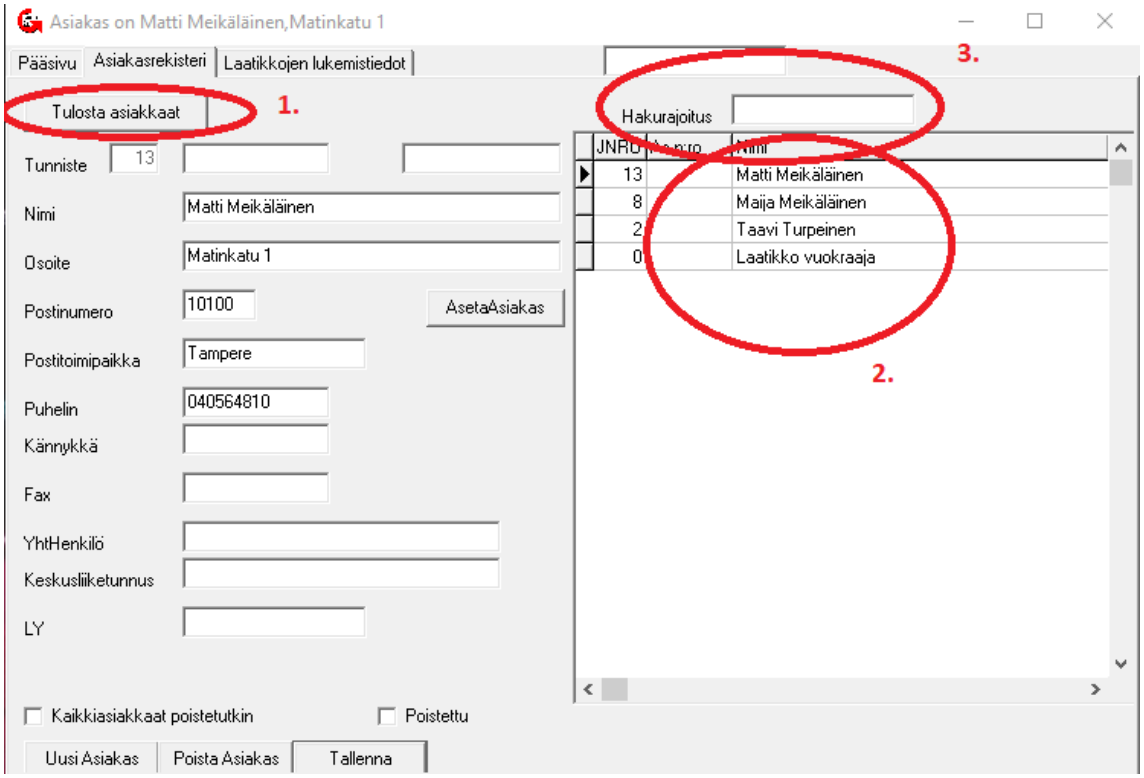
Wiegers, K. (2003). *Software requirements: practical techniques for gathering and managing requirements throughout the product development cycle* (2. painos). Redmond (Wash.): Microsoft Press.

Liitteet

Liite 1. Kyselytutkimuksen kysymykset

- Yrityksen nimi?
- Vastaajan nimi?
- Miten tilaatte Transbox-laatikoita?
- Kuinka monta laatikkoa tilaatte kerralla?
- Millaiseksi koette järjestelmän tällä hetkellä? Millaista järjestelmän käyttäminen on?
- Ketkä käyttävät järjestelmää?
- Mistä järjestelmän ominaisuuksista pidätte?
- Miten järjestelmän käyttöä voitaisiin helpottaa?
- Mitä lisäpalveluita haluaisitte järjestelmään? Kehitysideoita?

Liite 2. Nykyisen laatikkojärjestelmän asiakasrekisteri

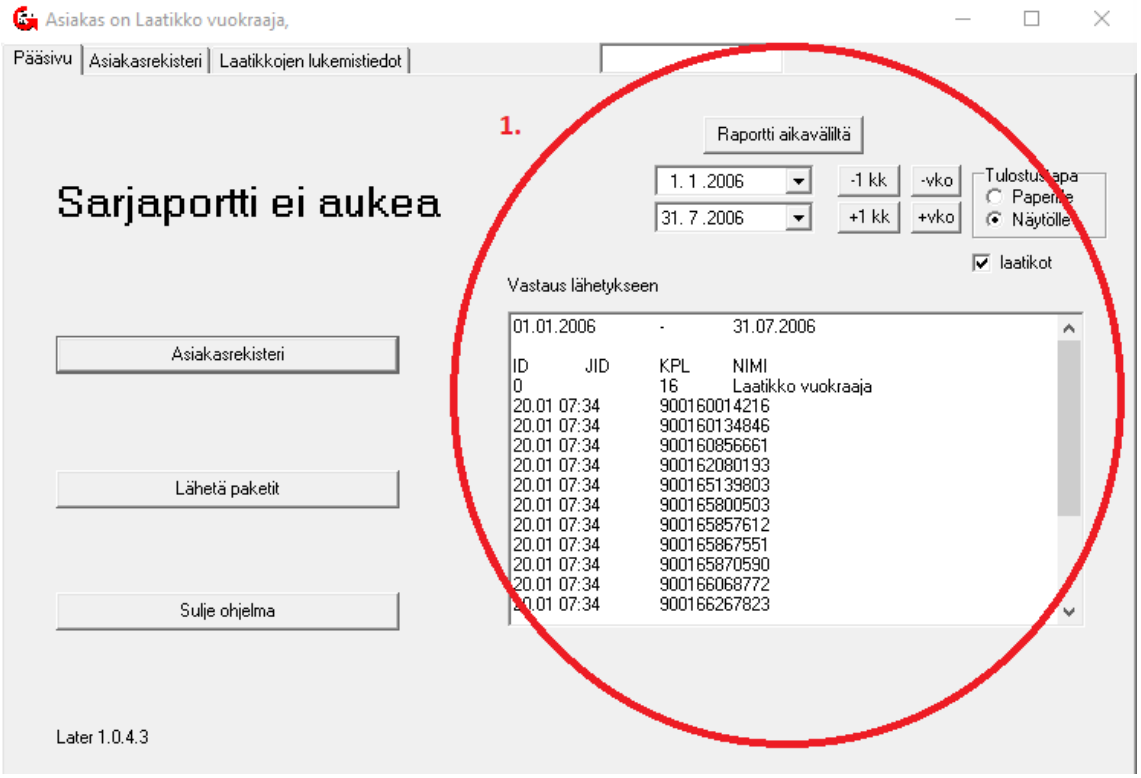


1. Käyttäjä joutuu aina tulostamaan koko asiakasrekisterin, vaikka hän haluaisi tulostaa ainoastaan yhden asiakkaan viivakoodin laatikoiden luentaa varten.

2. Asiakasrekisterin järjestyksen muokkaamisen käytettävyys on tällä hetkellä heikolla tasolla. Asiakasrekisterin järjestystä pystyy kyllä muokkaamaan klikkaamalla otsikkoriviä, jolloin asiakkaat järjestyvät kyseisen otsikon kriteerin mukaan pienimmästä suurimpaan tai aakkosjärjestykseen alkaen a:sta, mutta käyttäjälle tämä toiminto on epäselvä. Toimintoon ei ole mitään erillistä painiketta tai huomiota, jolloin saattaa käydä niin, että asiakas luulee, ettei asiakasrekisteriä pysty muokkaamaan. Näin kävi tässäkin tutkimuksessa. Asiakasrekisteriä ei voi kuitenkaan järjestää vastakkaiseen järjestykseen.

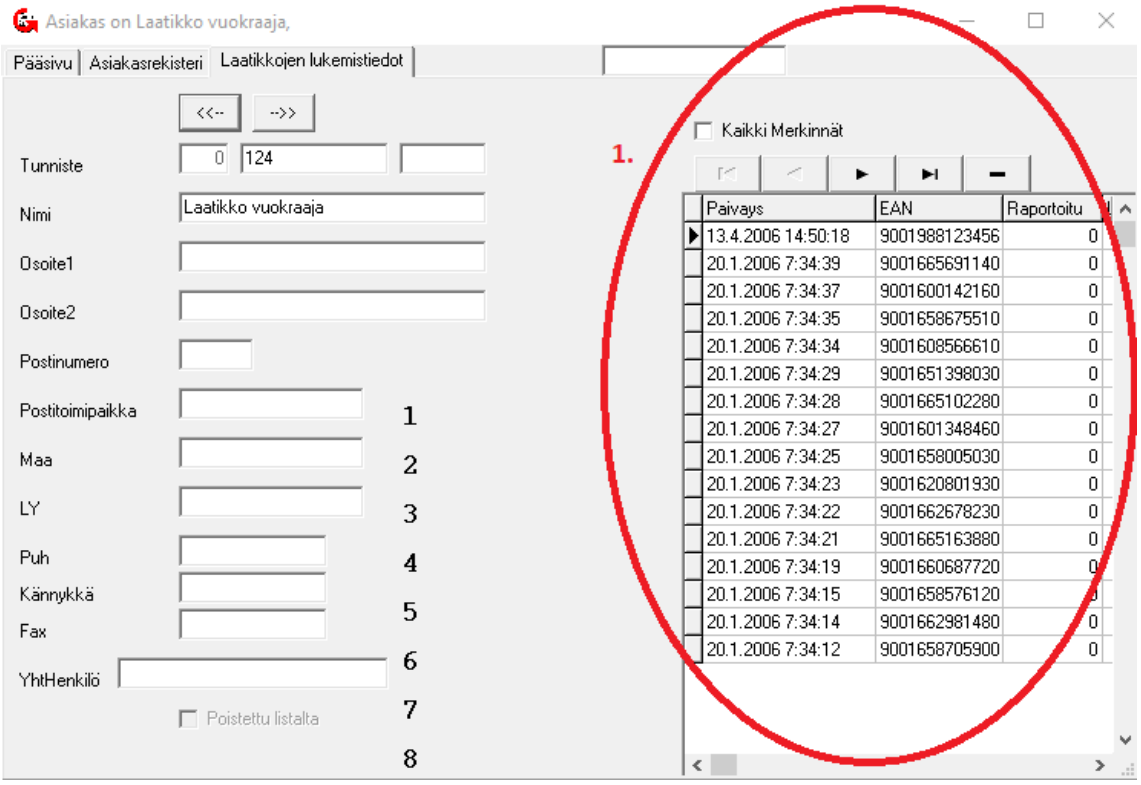
3. Asiakasrekisterissä on hakutoiminto, mutta se on hyvin pelkistetty. Nykyisellä hakutoiminnolla asiakkaita pystyy hakemaan ainoastaan asiakasnumeron, nimen tai osoitteen perusteella.

Liite 3. Nykyisen laatikkojärjestelmän pääsivu



1. Järjestelmästä pystyy tulostamaan raportin haluamaltaan aikaväliltä, joko paperille tai näytölle. Raportti ei ole kuitenkaan kovin käyttäjäystävällinen. Mikäli luettuja laatikoita kyseisellä aikavälillä on paljon, on tietyn laatikon löytäminen työlästä. Käyttäjä ei pysty erikseen tulostamaan raporttia esimerkiksi pelkästään tietyn asiakkaan luentatiedoista tai tietystä laatikosta. Tällä hetkellä ainoa kriteeri raportin tulostamiseen on aikaväli.

Liite 4. Nykyisen laatikkojärjestelmän luentanäkymä



1. Laatikoiden luentasivulla ei ole hakutoimintoa luettujen laatikoiden etsimistä varten. Jos käyttäjä lukee yhdellä luentakerralla paljon laatikoita, on tietyn laatikon luentatietojen löytäminen työlästä. Hakutoiminto luentasivulla saattaisi olla tarpeellinen esimerkiksi tilanteessa, jossa käyttäjä huomaa vasta myöhemmin lukeneensa väärän laatikon väärään paikkaan ja haluaa poistaa tämän. Hakutoiminnon avulla laatikon löytäisi helposti ja nopeasti laatikkonumerolla.

Liite 5. Kuvion 13 kirjallinen kuvaus

Alfa yritys. Vastaanottaa asiakastiedot laatikoiden omistajalta uuden asiakkaan luomisen yhteydessä ja myös silloin, kun asiakastietoihin tulee muutoksia. Lähettää kirjautumistunnukset jokaiselle asiakkaalle järjestelmän käyttöönoton yhteydessä. Tarkastelee asiakkaiden luentatietoja.

Asiakastiedot. Vastaanotetaan laatikoiden omistajalta ja lähetetään alfa-yritykselle käyttöönoton ja muutosten yhteydessä.

Laatikoiden omistaja. Lähettää asiakastiedot alfa-yritykselle uuden asiakkaan luomisen yhteydessä ja silloin, kun asiakastietoihin tulee muutoksia. Tarkastelee kaikkien laatikoiden käyttäjien luentatietoja. Lisää kiertoon uusia laatikoita.

Laatikkopesula. Vastaanottaa laatikoita pesua varten. Pesee laatikot käytön jälkeen. Vastaanottaa asiakkaiden laatikkotilauksia. Laatikkopesula lukee laatikoita lähettäessään ne takaisin laatikoita tarvitseville yrityksille. Luentaprosessi edellyttää pesulalta viivakoodinlukijan käyttöä.

Laatikkosaldo. Asiakkaat tarkastelevat laatikkosaldoaan. Luentaprosessi vähentää laatikkosaldoa automaattisesti.

Kirjautumistunnukset. Lähetetään alfa-yrityksen toimesta. Vastaanotetaan asiakkaiden toimesta.

Luentatiedot. Jokainen yksittäinen luentatapahtuma synnyttää aina luentatiedon. Laatikoiden omistaja, alfa-yritys ja asiakkaat tarkastelevat luentatietoja.

Luentaprosessi. Asiakkaat ja laatikkopesulat suorittavat luentoja laatikoiden lähettämisen yhteydessä. Kaikki luentatiedot syntyvät aina yksittäisessä luentaprosessissa.

Luentaprosessi vähentää automaattisesti aina kyseisen asiakkaan laatikkosaldoa. Luentaprosessi koostuu laatikoiden käsittelystä.

Asiakas. Jokainen asiakas tarkastelee omaa laatikkosaldoaan. Asiakas ylläpitää asiakasrekisteriä omista asiakkaistaan. Jokainen asiakas vastaanottaa yhdet kirjautumistunnukset järjestelmien käyttöä varten. Jokainen asiakas tarkastelee omia luentatietojaan. Jokainen asiakas suorittaa luentaprosesseja lähettäessään laatikot eteenpäin omille asiakkailleen. Jokainen asiakas käyttää vähintään yhtä viivakoodinlukijaa laatikoiden lukemiseen. Jokainen asiakas tilaa, vastaanottaa ja pakkaa laatikoita.

Viivakoodinlukija. Käytetään luentaprosesseihin asiakkaiden ja laatikkopesuloiden toimesta.

Asiakasrekisteri. Ylläpidetään jokaisen asiakkaan toimesta.

Laatikko. Laatikot ovat luentaprosessissa käsittelyn kohteena. Tilataan, pakataan ja vastaanotetaan asiakkaiden toimesta. Vastaanotetaan ja pestään laatikkopesuloiden toimesta. Laatikoiden lähettäminen tapahtuu luentaprosessin yhteydessä.

Tilaus. Tehdään asiakkaiden toimesta. Vastaanotetaan laatikkopesuloiden toimesta.