Helena Kasvi

TUOTEHALLINNAN JA LAAJUUDENHALLINNAN KÄYTÄNNÖT PIENESSÄ OHJELMISTOYRITYKSESSÄ: TAPAUSTUTKIMUS

JYVÄSKYLÄN YLIOPISTO
TIETOJENKÄSITTELYTIETEIDEN LAITOS
2017

TIIVISTELMÄ

Kasvi, Helena
Tuotehallinnan ja laajuudenhallinnan käytännöt pienessä ohjelmistoyrityksessä: tapaustutkimus
Jyväskylä: Jyväskylän yliopisto, 2017, 86 s.
Tietojärjestelmätiede, pro gradu -tutkielma
Ohjaajat: Käkölä, Timo, Forselius, Pekka ja Luoma, Eetu

Yhä useammin ohjelmistotuotetta tarjotaan avoimille markkinoille pikemminkin kuin yhdelle tietylle asiakkaalle. On myös olemassa runsaasti pieniä yrityksiä, jotka tekevät tätä ns. markkinalähtöistä ohjelmistokehitystä. Pienetkin yritykset haluavat hoitaa liiketoimintaansa ja johtaa prosessejaan menestyksekkäästi, mutta niiden kohtaamiin ongelmiin eivät suurille yrityksille tarkoitetut prosessienhallintamenetelmät sovi kovinkaan hyvin. Tämä tapaustutkimus pyrkii löytämään kirjallisuudessa esitettyjä ratkaisuja pienen ohjelmistotalon ongelmiin tuotteidensa ylläpidossa ja kehittämisessä. Yhtenä ratkaisuna esitellään sopiva tuotehallinnan viitekehys. Ohjelmistokehitysprojektien suurin ongelma, hallitsemattomuus, on ongelma myös tuotekehitysprojekteissa. Tutkimuksessa esitellään ohjelmistojen laajuuden mittaukseen kehitetty FiSMA 1.1 toimintopistelaskenta ja sitä hyödyntävä projektien ohjausmenetelmä northernSCOPE™-konsepti. Lopuksi tutkitaan, ovatko tuotehallinnan ja laajuudenhallinnan käytännöt otettavissa käyttöön pienessä, tuotteita valmistavassa ohjelmistotalossa.

Asiasanat: pieni yritys, ohjelmistotuote, markkinalähtöinen ohjelmistokehitys, tuotehallinta, toimintopistelaskenta

ABSTRACT

Kasvi, Helena
Product Management and Software Scope Management Practices in a Small Software Company: case research
Jyväskylä: University of Jyväskylä, 2017, 86 p.
Information Systems, Master's Thesis
Supervisors: Käkölä, Timo, Forselius, Pekka and Luoma, Eetu

More and more often a software product is offered to the open market instead of one specific customer. There are a great number of small enterprises doing so-called market-driven development in this manner. Small enterprises also want to manage their business and processes successfully, but process management methods designed for large companies are not suitable for them. This case research tries to find solutions presented in literature for problems arising in a small company's product management and development. The presented solution is the use of a suitable product management framework. The biggest problem in software projects, one of uncontrollability, is also a problem in software product development projects. This research proposes a Functional Size Measurement Method FiSMA 1.1 and northernSCOPE™ concept developed for project costing and software scope management. Finally these two methods will be evaluated alongside the product management framework to identify the potential usability for a small company in solving product development problems.

Keywords: small enterprise (SME), software product, market-driven software development, product management, function points

KUVIOT

TAULUKOT

SISÄLLYS

1 JOHDANTO

Euroopan Unionin Komissio (European Commission) on julkaissut vuonna 2005 määritelmän, jossa se jaottelee yritykset seuraavasti:

- Keskikokoisessa yrityksessä on alle 250 työntekijää ja sen vuosittainen liikevaihto on alle 50 milj. euroa
- Pienessä yrityksessä on alle 50 työntekijää ja sen vuosittainen liikevaihto on alle 10 milj. euroa
- Mikroyrityksessä on alle 10 työntekijää ja sen vuosittainen liikevaihto on alle 2 milj. euroa

Yrityksistä ja teollisuudesta vastaava EU-komissaari Günter Verheugen on sanonut (European Comission, 2005):

> Micro, small and medium-sized enterprises (SMEs) are the engine of the European economy. They are an essential source of jobs, create entrepreneurial spirit and innovation in the EU and are thus crucial for fostering competitiveness and employment.

Laporte, Alexandre ja O'Connor mainitsevat tutkimuksessaan (2008), että 85 %:ssa eurooppalaisista IT-sektorin yrityksistä on 1-10 työntekijää. Alan kirjallisuudessa yritykset jaotellaan *pieniin tai keskikokoisiin* (engl. Small to Medium Enterprises, SME) tai *hyvin pieniin yrityksiin* (engl. Very Small Enterprise, VSE). Jaottelu voidaan tehdä myös yksittäisen ohjelmistoprojektin työntekijämäärään tai projektin kestoon perustuen, jolloin Enterprise korvataan sanalla Entity (Laporte ym., 2008).

Pienet ohjelmistoyritykset ovat hyvin tärkeitä kansantaloudellisesti, mutta jopa puolet pienistä yrityksistä ei ole enää toiminnassa viiden vuoden kuluttua perustamisestaan. Suurimpana syynä tähän pidetään riittämätöntä ja osaamatonta hallinnointia (Berry, 2002). Systemaattisille ja tarkoitukseen sopiville toiminnoille pienten ohjelmistotuoteyritysten liiketoiminnan päätöksenteon tukemiseen on siis todella tarvetta (Vähäniitty, 2003).

Pieniä yrityksiä pidetään usein IT-alan kasvun ”moottoreina”; ne ovat hyvin dynaamisia, innovatiivisia ja tehokkaita ja niiden kasvupotentiaali on merkittävä (Kamsties, Hörmann & Shlich, 1998). Pienillä ohjelmistoyrityksillä on kuitenkin omat ongelmansa, tarpeensa ja toiveensa. Nämä kiteytetään hyvin Saastamoisen ja Tukiaisen artikkelin (2004) johdannossa: Ohjelmistoprosessien parantaminen (engl. Software Process Improvement, SPI) nostaa todistetusti tuotteiden ja palvelujen laatua. Pienten yritysten prosessien parantamistarpeet ovat samat kuin suuremmillakin yrityksillä eli myös ne pyrkivät toteuttamaan tuloksellisia ohjelmistoprojekteja, parantamaan ohjelmistojen laatua ja kasvattamaan asiakastyytyväisyyttä. Prosessien parannustyöhön ei ole kuitenkaan osoittaa aikaa, rahaa ja resursseja. Ulkopuolisen tuen tarve on ilmeinen, sillä yrityksiltä puuttuu kokemusta ja osaamista parannussuunnitelmien ja –toimenpiteiden läpiviemiseen. (Saastamoinen & Tukiainen, 2004.)

Ohjelmistotuotteiden valmistus on maailmanlaajuisesti ja taloudellisesti merkittävää toimintaa ja vuonna 2001 sen laskettiin kattavan jopa 9% kaikista ICT-alan markkinoista. OECD on arvioinut, että ohjelmistotuotteiden valmistus on yksi nopeimmin kasvavia aloja ja se tuottaa runsaasti lisäarvoa, luo työpaikkoja ja aikaansaa tutkimus- ja kehitysinvestointeja. Yhä useammin organisaatiot päättävät ostaa valmiin ohjelmistotuotteen kuin ryhtyvät tekemään tai teettämään omaa räätälöityä sovellustaan. (Xu & Brinkkemper, 2007.) Sawyerin (2000) mukaan suurin ero räätälöidyn ohjelmiston ja tuotteen kehittämisen välillä liittyy vaatimustenhallintaan, sidosryhmien ominaisuuksiin ja aikataulujen asettamiin rajoitteisiin. Tuotekehittämisen muita erityisominaisuuksia ovat myös julkaisujen suunnittelu, vaatimusten priorisointi sekä jatkuva uusien vaatimusten tulva (Carmel & Becker, 1995).

Projektinhallinnalla (engl. project management) tarkoitetaan yksittäisten projektien läpiviemistä järjestelmällisellä tavalla niin, että projektien ohjaamiseen käytetään prosessimalleja (Rautiainen, Lassenius, Vähäniitty, Vanhanen & Pyhäjärvi, 2002). Wikipediasta (2013) löytyy hakusanalla ”projektinhallinta” seuraava määritelmä:

> Projektinhallinta tarkoittaa resurssien (kuten työvoiman) organisointia ja hallintaa sellaisella tavalla, että projekti voidaan päättää suunnitellun sisältöisenä ja laatuisena, aikataulun sekä budjetin mukaisesti.

Ohjelmistoprojektit ovat saaneet mediassa laajaa negatiivista julkisuutta epäonnistumistensa takia. Projektien aikataulut venyvät ja kustannukset paisuvat hallitsemattomasti puhumattakaan siitä, että asiakkaan tilaamaa toiminnallisuutta ei ole pystytty toimittamaan. Ratkaisuksi ongelmaan on esitetty *ohjelmistoprojektien estimointia ja mittaamista* (engl. software project estimation and measurement, PEM) ja toiminnalliseen laajuuteen perustuva estimointi onkin tunnustettu yhdeksi ohjelmistoprojektien hallinnan parhaista käytännöistä (Forselius & Käkölä, 2009). Finnish Software Measurement Assosiation (FiSMA ry) on kehittänyt laajuudenhallintakonseptin, joka on saanut nimekseen northernSCOPE™. Konseptia on käytetty vuosien ajan lähinnä asiakaskohtaisten ohjelmistoprojektien hallintaan.

Tämän tutkimuksen aiheena on tietyn pienen ohjelmistotuotteita valmistavan yrityksen tuotehallinta, sen ongelmat ja hallintaprosessien kehittäminen. Projektinhallinnan parantamismenetelmänä tarkastellaan northernScope™ –konseptia. Tutkimus pohjautuu tuotehallintaa käsittelevään kirjallisuuteen ja tutkimusaineistoihin. Teoriaosuuden lähteitä haettiin pääasiassa Google Scholarin avulla ja hakusanoina käytettiin mm. termejä 'SME', 'software product management', 'software process estimation and measurement, PEM', 'scope management' ja 'functional size measurement'. Myös löydettyjen lähteiden lähteistä löytyi monta kiinnostavaa artikkelia. Lähteitä ei pyritty arvioimaan kriittisesti, vaan niistä etsittiin lähinnä taustatietoa tutkittavaan aiheeseen. Tutkimusmenetelmäksi valittiin kvalitatiivinen tapaustutkimus.

Tutkielman pääasiallinen tutkimusongelma on: "Miltä osin tuotehallinnan ja laajuudenhallinnan käytänteet ovat omaksuttavissa pienessä ohjelmistoyrityksessä?" ja lisäkysymyksenä haluttiin tutkia "Ovatko tuotehallinnan ja laajuudenhallinnan konseptit yhdistettävissä miltään osin?". Tutkielmassa käsitellään pienen ohjelmistoyrityksen erityispiirteitä ja ongelmia ja esitellään kirjallisuudesta löytynyt tuotehallinnan malli, jota syvennetään tuotepäällikön tehtävien ja vastuiden avulla. Tutkielmassa kohderyhmäksi rajataan pääasiassa hyvin pieni tuotekehitystä tekevä yritys. Toisessa luvussa esitellään tutkielman keskeiset käsitteet eli pieni ohjelmistoyritys, ohjelmistotuote ja tuotepäällikön tehtäväkenttä. Kolmannessa luvussa esitellään tuotehallinnan viitekehitys ja tutkitaan sen sisältämiä prosesseja yksityiskohtaisemmin pienen yrityksen näkökulmasta. Neljännessä luvussa esitellään ohjelmistojen toiminnallisuuteen perustuva laajuudenhallinta ja sitä varten kehitetty northernSCOPE™-konsepti. Viidennessä luvussa esitellään tutkimusmenetelmä ja –analyysi. Kuudennessa luvussa laaditaan arvioitava malli, joka yhdistää tuotehallinnan viitekehyksen ja northernSCOPE™-konseptin. Mallin käyttöönoton arviointia varten tehtiin tutkittavassa yrityksessä pienimuotoinen haastattelu, jossa pyrittiin selvittämään pienen tuotekehitystä tekevän yrityksen valmiuksia ja edellytyksiä selvitä muutoshankkeesta, jossa laadittu malli otetaan käyttöön. Seitsemännessä luvussa vastataan tutkimuskysymykseen ja arvioidaan tutkimuksen luotettavuutta. Viimeisessä luvussa vedetään yhteen ongelmat ja niihin löydetyt ratkaisut sekä esitetään jatkotutkimusaihe.

2 PIENI OHJELMISTOYRITYS, OHJELMISTOTUOTE JA TUOTEPÄÄLLIKÖN TEHTÄVÄT

Tässä luvussa käsitellään pienen ohjelmistoyrityksen tarvetta prosessiensa parantamiseen, parantamisessa kohdattavia ongelmia ja mahdollisia ratkaisuja niihin. Lisäksi esitellään lyhyesti ohjelmistotuote eri variaatioineen ja kuvataan tuotepäällikön tehtäväkenttää.

2.1 Pieni ohjelmistoyritys ja sen prosessit

Pienet, itsenäiset ohjelmistoyritykset ovat erittäin tärkeitä tekijöitä kotimaidensa talouskasvun kannalta. Säilyäkseen ja kasvaakseen pienet yritykset tarvitsevat tehokkaat työkalut ja prosessit ohjelmistonkehitykseen ja tuotteidensa hallintaan. Yleinen käsitys on, että hyvät käytännöt ovat kalliita, aikaa vieviä ja tarkoitettu suurille yrityksille ja siten vaikeita ottaa käyttöön pienissä yrityksissä. (Richarson & Gresse von Wagenheim, 2007.) Mutta myös pienet yritykset tarvitsevat hyviä käytäntöjä. Tore Dybå (2003, 148) on löytänyt Brownin ja Duguidin (1999) esseestä merkittävän havainnon:

> Käytäntö ilman prosessia muuttuu hallitsemattomaksi; prosessi ilman käytäntöä johtaa innovaatioihin tarvittavan luovuuden häviämiseen.

Dybå (2003) on tutkinut isojen ja pienten yritysten eroja prosessien ja parhaiden käytänteiden soveltamisessa. Hän on tullut tutkimuksessaan tulokseen, että suurin ero pienten ja suurten ohjelmistoyritysten välillä on tavassa, kuinka ne reagoivat epävakaissa ja muuttuvissa olosuhteissa. Pienissä yrityksissä vastaus muuttuneeseen tilanteeseen on tutkimuksen määrän nostaminen. Turbulenteissa olosuhteissa pienet ohjelmistoyritykset käyttävät tutkivia strategioita ja samalla pyrkivät hyödyntämään aikaisempia kokemuksia, kun taas isommat yritykset jatkavat valitsemallaan ja hyväksi katsomallaan linjalla. Pienet yritykset ovat mukautuvaisia, kun taas suuret yritykset ovat byrokraattisia, suosivat eri-

koistuneita ja jyrkkärajaisia tehtävänkuvia sekä painottavat vakautta, järjestystä ja kontrollia. Tästä johtuen suurilla yrityksillä on usein vaikeuksia mukautua muuttuviin olosuhteisiin; niiden on tarkoitus saavuttaa ennalta asetetut tavoitteet eikä suinkaan olla innovatiivisia. Suuret yritykset painottavat omien ”parhaiden käytäntöjensä” eli muodollisten menettelyjen, prosessimallien, ohjeistojen, sääntöjen, tarkistuslistojen jne. käyttämistä ohjelmistoprosessien hallintaan ja parantamiseen. Pienet yritykset taas haluavat hyödyntää henkilöresurssiensa monipuolisuutta ja luovuutta ja painottavat ohjelmistoprojekteissaan uusien mahdollisuuksien tutkimista. (Dybå, 2003.)

Habra, Alexandre, Desharnais, Laporte ja Renault luonnehtivat tutkimuksessaan (2008) pieniä yrityksiä siten, että niillä ei ole osoittaa resursseja prosessien parantamiseen ja erityisesti laadunparantamistehtävät jäävät tekemättä. Pienillä yrityksillä on määritelmänsä mukaisesti vähän työntekijöitä ja heidän työpanoksensa tarvitaan tuotannon tehtävien hoitamiseen tiukkojen aikataulujen paineessa. Yleiset laatumallit, kuten CMMI (SEI, 2002) ja ISO 15504 (ISO, 2004), eivät ole sellaisenaan soveltuvia pienille yrityksille, sillä niiden käyttöönotto on aivan liian monimutkaisia ja kallista. Tutkimuksessaan Habra ym. (2008, 764) luettelevat joukon pienille ohjelmistoyrityksille ja niiden toiminnalle tyypillisiä ominaisuuksia:

- Käytetty ohjelmistokehityksen elinkaarimalli on hyvin yksinkertainen. Edes pääkohtia, kuten suunnittelua, käyttöönottoa ja ylläpitoa ei ole tarkoin määritelty. Toteutus ja testaaminen ovat päävaiheet ja vaikka testaamista pidetään erittäin tärkeänä, siitäkin tingitään resurssien puuttuessa tai sovitun aikarajan lähestyessä.
- Saman organisaation sisällä prosessien kypsyystasot vaihtelevat eli korkealaatuiset ja heikot käytännöt sekoittuvat. Laadultaan parhaita ovat lakiin ja sopimuksiin perustuvat asiakas-toimittajasuhteiden käytännöt, jotka kuitenkin usein koetaan taakaksi eikä suinkaan eduksi organisaatiolle.
- Joillakin pienillä yrityksillä on käytössään erittäin hyvät projektinhallintamenetelmät, joillakin taas ei. Mutta saman organisaation sisälläkin käytännöt voivat olla hyvin eritasoisia riippuen projektista, asiakkaasta, projektipäälliköstä ja kehitystiimistä.
- Koulutukseen ja henkilöstöhallintaan käytetään vähän resursseja johtuen budjettirajoitteista. Koulutusta järjestetään usein viimeisenä keinona selvitä tilanteessa, jossa tiimin jäsenten itseopiskelu ei ole ollut riittävän tehokasta.
- Pienet yksiköt ovat projektisuuntautuneita eikä niiden prosesseja ohjata pitkän tähtäimen suunnitelmilla. Tästä johtuen oppimisen ja tiedonhallinnan käytäntöjä ei juuri ole.
- Tavallisesti minkäänlaista riskienhallintaa ei ole. Suuria riskejä voidaan ottaa hallitsemattomasti, mutta tämä myös osaltaan selittää sen, miksi pienet yksiköt voivat menestyä ilmiömäisesti.

Christian Hofer tutki vuonna 2002 itävaltalaisia pieniä ja hyvin pieniä ohjelmistoyrityksiä, jotka suhtautuivat tulevaisuuteensa ja kasvupotentiaaliinsa erittäin optimistisesti. Tutkimuksessa selvisi, että yrityksillä on kommunikaatioongelmia sidosryhmiensä kanssa ja että nämä ongelmat ovat pääsyynä projektien myöhästymiseen tai epäonnistumiseen. Projektien aikana muuttuvat vaatimukset ja vaihtuvat tavoitteet ovat toinen suuri ongelmalähde. Ohjelmistoprojektit kärsivät myös henkilöstö-, budjetti- ja aikatauluongelmista. Yritykset hyötyvät tehokkaasta ja säännöllisestä henkilöstön kouluttamisesta, sillä tällöin henkilöstön motivaatio käyttää uusimpia teknologioita ja menetelmiä kasvaa. Yritys saa näin etua suhteessa kilpailijoihinsa, koska ohjelmistokehitys sujuu nopeammin ja tuotteiden pääsy markkinoille nopeutuu. (Hofer, 2002.)

Ita Richardson (2002, 111-112) on listannut ohjelmistoprosessin parannusmallin tekijöitä ja piirteitä, jotka pienen yrityksen kehityssuunnitelmassa tulisi olla mukana ja ottaa huomioon:

- **Yhdistä yrityksen liiketoimintatavoitteisiin**. Pienen yrityksen rahoitus on rajallista ja usein omistaja/yrittäjä on mukana suurella henkilökohtaisella panoksella. Tämän takia minkä tahansa projektin, myös ohjelmistoprosessien kehitysprojektin, on oltava linjassa yrityksen liiketoimintatavoitteiden kanssa.
- **Keskity kaikkein tärkeimpiin ohjelmistoprosesseihin**. Kun uusia ja muokattuja käytänteitä otetaan käyttöön, on tärkeää, että niiden aiheuttamat muutokset vaikuttavat eniten yrityksen tärkeimpiin prosesseihin. Nämä prosessien parannukset tuovat eniten liiketoimintahyötyä.
- **Etsi tuottoisin sijoitus**. Yksikään yritys, suuri tai pieni, ei halua haaskata rahojaan tuottamattomiin projekteihin. Pienessä yrityksessä ei ole osastoja, jotka voisivat paikata muiden aiheuttamat tappiot, joten projektiin sijoitettujen resurssien tuottama arvo on erityisen tärkeää.
- **Ehdota parannuksia, jotka vaikuttavat mahdollisimman nopeasti**. Pienessä yrityksessä ohjelmistoprosessien parantaminen annetaan usein ohjelmistosuunnittelijoiden tehtäväksi eikä käytettävissä ole pelkästään laadunhallintaan keskittyvää henkilöä. Jos parannuksen hyödyt eivät ole heti nähtävissä, käy luultavasti niin, että koko projekti painuu unholaan.
- **Hae nopea tuotto investoinnille**. On epärealistista odottaa, että omistaja sijoittaa rahojaan projekteihin, joista on odotettavissa tuottoa vasta pitkän ajan kuluttua.
- **Keskity prosesseihin**. Monet ohjelmistoprosessimallit perustuvat prosessien kypsyyden arviointiin eivätkä itse prosesseihin tehtäviin muutoksiin. Pienelle yritykselle on tärkeämpää keskittyä jokapäiväiseen selviämiseen ja siinä ohessa miettiä, millaiset parannukset ja muutokset prosesseihin lopulta johtaisivat tuotteiden laadun paranemiseen.
- **Ole joustava ja helppokäyttöinen**. Käytettävän prosessinparannusmallin on oltava tarpeeksi yksinkertainen, että sen käyttöönotto sujuu

ilman laajamittaista koulutusta. Koulutukseen varatut resurssit halutaan luultavasti suunnata perustoimintoihin eikä laatumalleihin.

- **Osoita vaikutus yrityksen sisällä**. Kun nykyisiä ohjelmistokehityksen käytäntöjä muutetaan, pitäisi sillä olla huomattavaa vaikutusta myös muihin yrityksen prosesseihin. Vaikutuksen osoittaminen vain saattaa olla vaikeaa.

Joensuun yliopistossa toteutettiin vuonna 2002 pienille ohjelmistoyrityksille suunnattu prosessien arviointi- ja parannusohjelma, tSoft (Saastamoinen & Tukiainen, 2004). Aivan aluksi osallistuvia yrityksiä pyydettiin nimeämään omien prosessiensa joukosta kolme tärkeintä, jotka ne kokivat vahvuuksina, heikkouksina, mahdollisuuksina ja riskeinä. Saadut vastaukset koottiin perinteiseen SWOT-nelikenttään (taulukko 1):

TAULUKKO 1 Pienten yritysten prosessien SWOT-analyysi (Saastamoinen ym., 2004, 73)

Vahvuudet	**Heikkoudet**
- henkilöstön osaaminen (kokemus ja koulutus) - projektinhallinta (suunnittelu ja läpivienti) - tekninen osaaminen	- testaaminen - tuotehallinta (dokumentointi ja muutostenhallinta) - projektinhallinta (resurssit, välitavoitteet, asiakashallinta)
Parannusmahdollisuudet	**Riskit**
- vaatimustenhallinta - asiakasyhteistyö - testaaminen	- henkilöstön vaihtuvuus - kustannus- ja aikatauluylitykset - tyytymättömät asiakkaat johtuen ohjelmistovirheistä ja kustannus- ja aikatauluylityksistä

Yrityksen henkilöstön pätevyys ja tekninen osaaminen mainittiin kriittisinä vahvuuksina samoin kuin projektin toteuttamiskäytännöt. Heikkouksista nousivat esille testaaminen, tuotehallinta ja projektin resurssien ja asiakkaidenhallinta. Yritykset olivat eniten kiinnostuneita parantamaan testausta, vaatimustenhallintaa ja yhteistyötä asiakkaiden kanssa. Suurimpina uhkina yrityksen toiminnalle mainittiin henkilöstön vaihtuvuus, projektien aikataulujen ja kustannusten ylitys sekä näistä ja ohjelmistovirheistä johtuva asiakastyytymättömyys. (Saastamoinen & Tukiainen, 2004.)

2.2 Ohjelmistotuotteet

Yhä useammin ohjelmistotuotetta tarjotaan avoimille markkinoille pikemminkin kuin yhdelle tietylle asiakkaalle. Tällaista ohjelmistokehitystä kutsutaan

markkinalähtöiseksi ja se viittaa tilanteeseen, jossa geneerisen tuotteen kehityskustannukset jaetaan usean markkinoilta löytyvän ostajan kesken. Markkinalähtöinen kehitys eroaa siis merkittävästi asiakaslähtöisestä, räätälöidystä ohjelmistokehityksestä, jossa yksi ainoa asiakas maksaa kaikki tuotantokulut ja lopputuote vastaa täysin yhden asiakkaan tarpeita ja toiveita. Markkinalähtöinen ohjelmistokehitys aiheuttaa erityisiä haasteita etenkin vaatimusmäärittelyn prosesseihin ja hallintaan. (Regnell & Brinkkemper, 2005.)

Ohjelmistotuotteella on tiettyjä ominaisuuksia, jotka erottavat sen asiakaskohtaisesta sovelluksesta. Software Engineering Institute (SEI) määrittelee (Brownsword, Oberndorf & Sledge, 2000, 49) ohjelmistotuotteen seuraavasti:

- sovellus myydään, vuokrataan tai lisensoidaan kohderyhmälle
- myyjä myy sovellusta ja pyrkii hyötymään siitä kaupallisesti
- myyjä tukee ja kehittää sovellusta
- myyjä omistaa sovelluksen oikeudet
- myyjä jakaa ohjelmasta identtisiä kopioita ilman, että lähdekoodiin tehdään muutoksia.

Ohjelmistotuote kehittyy tyypillisesti peräkkäisissä *julkaisuissa* (engl. release), joista jokainen sisältää uusia ja parannettuja toiminnallisuuksia ja joiden tarkoituksena on pitää asiakkaat tyytyväisinä ja kilpailijat takana. (Rautiainen ym., 2002).

Ohjelmistotuotteista puhuttaessa ensimmäisenä tulevat mieleen ”sellofaanisofta” (engl. shrink-wrapped software), commercial off-the-shelf (COTS)-ohjelmisto, paketoitu ohjelmisto tai kaupallinen ohjelmisto. Muut käsitteet, kuten avoimen lähdekoodin ohjelmisto (engl. Open Source Software) tai sovellusten tarjoaminen palveluna (Application Software Provision, ASP tai Software as a Service, SaaS), on myös otettava huomioon. On tavallista, että kirjallisuudessa näiden eri käsitteiden rajat ovat hämärtyneet. (Xu & Brinkkemper, 2007.) ”Selloofaanisofta” on paketoitu muoviin käärittyyn laatikkoon ja sitä myydään kaupassa. Myös yleiset käyttöjärjestelmät kuuluvat tähän joukkoon. COTS-ohjelmisto taas:

- on kehitetty kokonaiselle markkinasegmentille yksittäisen asiakkaan sijaan
- tuote myydään, vuokrataan tai lisensoidaan yleisölle
- myyjä pyrkii hyötymään tuotteesta taloudellisesti
- myyjä omistaa tuotteen oikeudet sekä tukee ja kehittää tuotetta
- tuotteesta on useita identtisiä kopioita, eikä sen lähdekoodi ole muokattavissa.

Paketoidulla ohjelmistolla tarkoitetaan nykyään laajoja yritysohjelmistoja (engl. Enterprise Resource Planning, ERP). Nämä ohjelmistot eivät useinkaan ole sellaisenaan otettavissa käyttöön, vaan ne vaativat jonkin verran muutoksia ja parametrointia. Laajan ERP-ohjelmiston käyttöönotto saattaa vaatia jopa kuukau-

sien työn, kun se sovitetaan yksittäisen organisaation liiketoiminnan tarpeisiin. Kaupallinen ohjelmisto tai sen käyttöön oikeuttava lisenssi ostetaan jälleenmyyjän kautta eikä ohjelmiston kopiointi ole mahdollista ilman oikeudenomistajan lupaa. Avoimen lähdekoodin ohjelmisto on nimensä mukaisesti kenen tahansa kiinnostuneen tarkasteltavissa ja muokattavissa; päinvastoin kuin kaupallisissa ohjelmistoissa, joissa lähdekoodi on tarkoin varjeltu liikesalaisuus. Myös avoimen lähdekoodin ohjelmistojen käyttöön sisältyy usein jonkinlainen lisensointi, joka kattaa oikeudet esimerkiksi muokkaukseen, jakamiseen tai kaupalliseen käyttöön. Ohjelmisto ei ole välttämättä ilmainen, mutta yleensä jakaminen edelleen sallitaan. ASP-ohjelmisto pyörii palveluntarjoajan palvelimella ja sen käytöstä useimmiten veloitetaan. (Xu & Brinkkemper, 2007.)

Xu ja Brinkkemper (2007, 3) esittävät artikkelissaan tiiviin määritelmän ohjelmistotuotteille: ”*Ohjelmistotuote on paketoitu kokoelma ohjelmistokomponentteja ja/tai ohjelmistoihin perustuvia palveluita, mukaan lukien lisämateriaalit, joka julkaistaan ja myydään tietylle markkinasegmentille*”. Tässä määrittelyssä ’paketoidut ohjelmistokomponentit’ viittaavat kaikkiin edellä esiteltyihin ohjelmistoihin, niiden koodiin, suoritettaviin osiin ja web-sivuihin. ’Ohjelmistoihin perustuvat palvelut’ kattavat esim. ASP- ja SaaS-palvelut. ‘Lisämateriaaleja’ ovat ohjelmistojen dokumentaatiot, käyttöoppaat, koulutusmateriaalit, esitteet, jne. ‘Julkaisu ja myynti’ taas määrittelee tuotteen kaupallisen arvon.

Ohjelmistotuotteissa voidaan Regnellin ja Brinkkemperin (2005, 289) mukaan tunnistaa erilaisia räätälöinnin asteita:

- Ohjelmiston sanotaan olevan yleinen, jos sitä on tarkoitus käyttää sellaisenaan, ”out-of-box”, ehkä joitakin pieniä käyttäjän itsensä tekemiä asetuksia lukuun ottamatta.
- Ohjelmisto on asiakaskohtaisesti sovitettu, jos se parametroidaan tietylle asiakkaalle esimerkiksi lisäämällä uusia moduuleja avoimen ohjelmointirajapinnan avulla.
- Toisessa ääripäässä on ohjelmistotuote, joka kehitetään täysin yhden asiakkaan toiveiden mukaan.

Tästäkin jaottelusta voidaan huomata, ettei markkina- ja asiakaslähtöisen ohjelmiston kehittämisen erottaminen toisistaan ole välttämättä aivan selkeää. Saattaa myös olla, että kehitysorganisaatio myy yleistä tuotetta avoimilla markkinoilla, mutta tarjoaa samanaikaisesti tuotteensa räätälöintiä konsultointipalveluna. Samoin uusi ohjelmiston osa saatetaan kehittää ensin jollekin tietylle asiakkaalle osittain hänen kustantamanaan ja sitten myöhemmin liittää tuotteeseen muidenkin asiakkaiden saataville. Näin ohjelmistoyrityksen on mahdollista saada investoinnista enemmän tuottoa. (Regnell & Brinkkemper, 2005.)

2.3 Tuotepäällikön tehtävät

Ohjelmistotuotteisiin erikoistuneissa yrityksissä tuotepäällikön tehtävä on laajentunut vuosien kuluessa ja sille kertyy strategista arvoa ja painetta, mutta samalla tehtävän toteuttaminen on muuttunut entistä hankalammaksi. Tuotepäällikkö on vastuussa vaatimushallinnasta, julkaisujen suunnittelusta ja tuotteen määrittelystä ympäristössä, jossa monet ulkoiset ja sisäiset sidosryhmät ovat vaikuttamassa. Tuotepäällikkökonsepti on ollut valmistavassa teollisuudessa käytössä jo vuosikymmeniä ja monet sieltä tutut käytännöt voidaan siirtää suoraan myös ohjelmistoteollisuuteen. (Cusumano, 2004.)

Tuotepäällikön tehtävä asettuu jonnekin tuotekehityksen, myynnin ja markkinoinnin välimaastoon. Hän joutuu pohtimaan laitteiden, ohjelmistojen ja palveluiden yhdistelmiä. Tuotepäällikkö vastaa tuotteidensa menestyksestä, joten hän on tavallaan vastuussa myös yrityksen menestyksestä. Tuotepäällikön tehtäväkenttä on hyvin laaja. Kirjassaan Jari Parantainen (2013) luettelee joitakin tyypillisiä tuotepäällikön tehtäviä: osallistu myyntikäynneille, kartoita kilpailutilanne, selvitä tulevia asiakastarpeita, toimita ennuste budjettia varten, oikolue käyttöohjeet, ratkaise tekninen ongelma, testaa tuotetta, myy asiakastarpeet tuotekehittäjille, kouluta myyjät ja asiakastuki, jne. Tuotepäällikkö on harvoin kenenkään esimies, joten hän joutuu etsimään apua ja ratkaisuja moninaisiin tehtäviinsä suostuttelun ja perustelujen avulla. (Parantainen, 2013.) Tuotepäällikön vastuut ja velvollisuudet tuotteen toiminnallisuuden suhteen ovat laajat, mutta koska hän ei ole kuitenkaan kehitystiimin esimies, päätösten tekeminen vaatii useiden osapuolten yhteistyötä (van de Weerd ym., 2006).

Pragmatic Marketingin (2010) tekemän maailmanlaajuisen kyselyn mukaan tuotepäällikön vastuulle katsotaan kuuluvan seuraavat asiat: tuotteen roadmap (91%), vaatimukset (86%), markkinaongelmat (77%), käyttötapaukset (74%) ja kilpailuympäristö (73%). Maglyas, Nikula ja Smolander (2012) tutkivat haastatteluin tuotepäällikön tehtäväkenttää ja saivat selville, että tuotepäällikön roolia kuvataan usein termillä ”tuotteen mini-toimitusjohtaja” ja tehtäviksi katsotaan kaikki toimet, joita tuotteen menestyminen vaatii. Tuotepäällikkö on myös asiakkaan ääni, joka edustaa asiakasta yrityksen sisällä. Saatujen tulosten perusteella tutkijat luettelevat tuotepäällikön kuusi ydintehtävää: tuoteanalyysi, roadmapping, strateginen hallinta, visio, tuotteen elinkaaren hallinta sekä sisäinen ja ulkoinen yhteistyö. (Maglyas ym., 2012.)

Tuotepäälliköt keräävät vaatimuksia useista eri lähteistä ja valikoivat niistä sopivat seuraavaa tuotejulkaisua varten. Jos tekeillä on kokonaan uusi tuote tai merkittävä toiminnallinen laajennus nykyiseen tuotteeseen, on vaatimusten etsiminen ja selville saaminen erityisen tärkeää. Kun vaatimukset annetaan syötteenä kehitysprosessille, saadaan tuloksena uusi tuote, jonka lopullista paketointia varten tuotepäällikön on valmistettava lisämateriaalit, kuten manuaalit sekä koulutus- ja markkinointimateriaali. (Regnell & Brinkkemper, 2005.) Parantaisen (2013, 17-18) mukaan tuotepäällikön kannattaa kuitenkin keskittyä pääasiassa viiteen eri tehtäväkokonaisuuteen:

- **Aikataulun saneeraus**. Tuotepäällikön kannattaa lopettaa tulipalojen sammuttelu ja karsia työtehtäviään reilusti. Näin aikaa jää esimerkiksi osaamisen kasvattamiselle ja markkinatrendien seuraamiselle.
- **Tuotestrategian laatiminen**. Tuotepäällikön laatima tuotestrategia saattaa olla jopa parempi ja käyttökelpoisempi kuin yritysjohdon laatima yleinen liiketoimintastrategia.
- **Tiedustelu**. Tuotepäällikön on yritettävä selvittää, millaisia tuotteita tai palveluita asiakkaat haluavat ja tarvitsevat tulevina vuosina. Asiakkaat osaavat harvoin antaa toiveistaan täsmällistä tietoa, vaan tuotepäällikön on erilaisista vihjeistä ja merkeistä pääteltävä tuotteen tulevat linjaukset.
- **Tuotteistaminen**. Tuotepäällikön on paketoitava tuote ja siihen liittyvät palvelut helposti omaksuttavaan muotoon. Tuote ei kuitenkaan ole koskaan valmis, vaan siihen on jatkuvasti tehtävä muutoksi.
- **Sisäinen markkinointi**. Tuotepäällikön työ on lähes pelkkää viestintää. Hänen on myytävä ajatuksensa sekä sisäisille että ulkoisille sidosryhmille, joiden huomiosta kilpailevat myös lukuisat muut tahot.

Tuotepäällikkö on joka tapauksessa yrityksen ylimmän johdon alainen ilman suoraa käsky- ja toimeenpanovaltaa. Hänen rooliaan voidaan myös kuvata termeillä asiantuntija, strategianlaatija, vetäjä tai ongelmanratkaisija. (Maglyas ym., 2012.)

2.4 Yhteenveto käsitteistä

Tässä luvussa esiteltiin tutkielman keskeiset käsitteet; pieni ohjelmistoyritys, ohjelmistotuote ja tuotepäällikön tehtävät ja vastuut.

Pienet yritykset ovat kasvuhakuisia ja innovatiivisia ja haluaisivat hoitaa liiketoimintaansa ja tuotekehitystään hyvin hallittujen prosessien avulla. Tarjolla olevat prosessimallit on kuitenkin kehitetty suurten organisaatioiden tarpeisiin ja niiden käyttöönotto olisi rajoitetuin resurssein toimiville pienyrityksille kohtuuton rasite. Pienen yksikön on jo tarpeeksi haastavaa selvitä normaaliarjessaan ohjelmistoprojektiensa läpiviennistä saati, että vähäisen henkilöstön tehtäviin lisättäisiin vielä prosessien ja menetelmien tutkimista ja kehittämistä. Yrityksillä ei ole kuitenkaan mitään menetelmien käyttämistä vastaan, jos vain niille löytyisi riittävän kevyt ja helposti implementoiva prosessimalli tuotteiden elinkaaren hallintaan.

Ohjelmistotuote on yleisille markkinoille kehitetty ohjelmisto, joka voidaan ottaa sellaisenaan tai pienin muutoksin usean asiakkaan käyttöön. Tuotteen julkaisija kehittää sitä edelleen ja tarjoaa asiakkailleen käyttötukea. Kehittäminen tapahtuu sidosryhmien esittämien vaatimusten perusteella ja uudet versiot toimitetaan asiakkaille peräkkäisten julkaisujen muodossa.

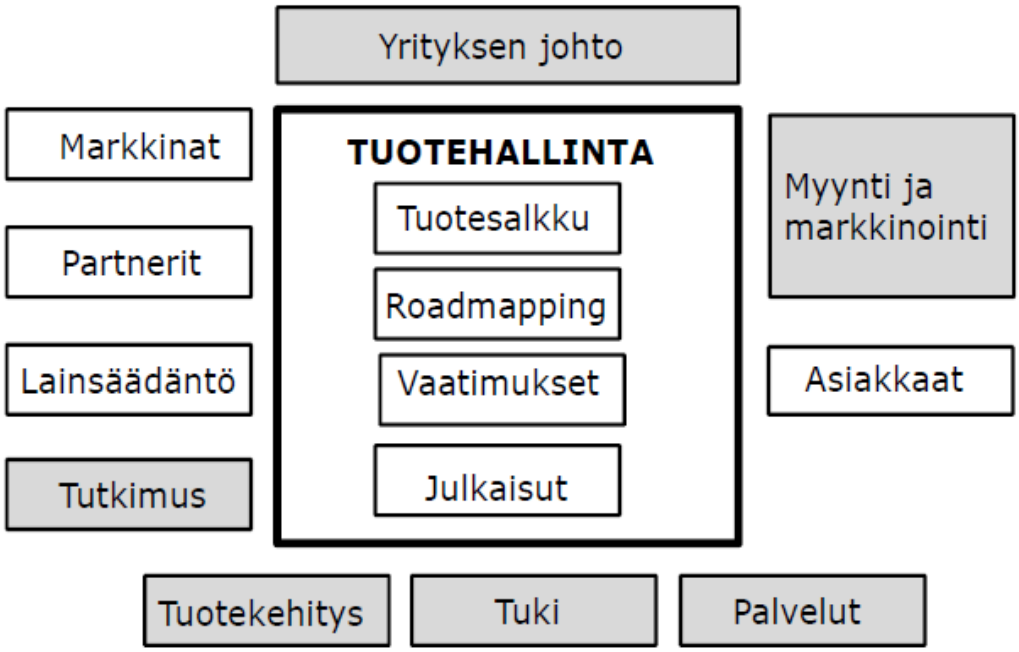
Tuotepäällikön tehtävät ja vastuut ovat hyvin laajat ja moninaiset. Tuotepäällikkö vastaa tuotteensa määrittelystä, vaatimustenhallinnasta, tuotteistamisesta ja markkinoinnista. Hän toimii usean eri sidosryhmän ristipaineessa; asiakkaat, yrityksen johto, myynti, tukitoiminnot, ohjelmistokehitys ja monet ulkoiset sidosryhmät odottavat saavansa tuotepäällikön palveluita. Tuotteen, ja samalla usein myös yrityksen, menestyminen on kiinni tuotepäällikön onnistumisesta tehtävässään.

3 OHJELMISTOTUOTTEIDEN HALLINTA

Christof Ebert (2007, 852) määrittelee artikkelissaan tuotteiden hallinnan tiedonalaksi tai rooliksi, joka hallinnoi tuotetta (tai ratkaisua tai palvelua) sen alkuhetkestä aina markkinoille ja asiakastoimitukseen asti ja jonka tehtävänä on tuottaa liiketoiminnalle paras mahdollinen arvo. Ohjelmistotuotteet eroavat kuitenkin muista tuotteista siinä, että niistä tehtävien uusien kopioiden valmistaminen ja jakelu eivät käytännössä aiheuta yritykselle ylimääräisiä kustannuksia (Cusumano, 2004). Valmiita ohjelmistotuotteita voidaan myös muuttaa helposti ja jo myytyjä ja käytössä olevia tuotteita voidaan päivittää uusilla julkaisuilla ja päivityspaketeilla. Muutosten tekemisen helppoudella on myös kääntöpuolensa: vaatimusten hallinta ja muutosten jäljittäminen ovat monimutkaisia tehtäviä ja muutosten tekemisen helppous saattaa johtaa julkaisutiheyden nousuun. (van de Weerd ym., 2006.)

Viitekehysten on todettu hyödyttävän monien eri alojen tutkimusta ja käytänteitä. Hollantilaiset tutkijat van de Weerd, Brinkkemper, Nieuwenhuis, Versendaal ja Bijlsma (2006) ovat huomanneet ohjelmistotuotteiden valmistamiseen sopivan kattavan viitekehyksen puuttumisen. He esittelevätkin artikkelissaan kehyksen, jossa tunnistetaan ja määritellään kaikki tuotehallinnan avainprosessit eli ohjelmistosalkun hallinta, tuotteen roadmapping, julkaisujen suunnittelu ja vaatimusten hallinta unohtamatta myöskään sidosryhmiä ja niiden suhteita tuotehallintaan. Viitekehystä tarvitaan koko tuotehallinnan alueen ymmärtämiseen ja sen eri osista tehdyn tutkimuksen yhdistämiseen ja soveltamiseen. Lisäksi viitekehystä voidaan käyttää lähtökohtana avointen tutkimuskysymysten identifiointiin ja avainkäsitteiden määrittelyyn, tuotepäälliköiden koulutukseen ja osaamisen lisäämiseen sekä työkalujen kehittämiseen. (van de Weerd ym., 2006.)

Esiteltävä viitekehys on alkuperäisenä liitteessä 1 ja mukaelma siitä kuviossa 1. Viitekehys soveltunee muutamin muutoksin myös pienille ohjelmistoyrityksille tuotteiden hallinnan prosessirungoksi ja sen ominaisuuksia tutkitaan seuraavaksi.



KUVIO 1 Tuotehallinnan viitekehys (mukaillen van de Weerd ym., 2006)

Tehokkaassa tuotehallinnassa on perimmältään kysymys hyvin organisoiduista ja prosessoiduista käsitteistä, jotka liittyvät vaatimuksiin, tuotteisiin ja julkaisuihin. Näiden käsitteiden hierarkkinen järjestäminen määrää prosessialueiden rakenteen. (van de Weerd ym., 2006.)

Ensimmäisenä hierarkiassa on ns. *tuotesalkku* (engl. product portfolio) eli kokoelma, joka sisältää yrityksen kaikki tuotteet. Pienillä tai uusilla yrityksillä saattaa olla salkussa vain yksi tuote, mutta isommilla yrityksillä tuotteita on tyypillisesti useampia. Jokaisesta tuotteesta on olemassa *julkaisujen* (engl. release) historia, nykytilanne ja tulevaisuus. Markkinoille julkaistujen versioiden välissä saattaa olla useita sisäisiä versioita. Julkaisujen numerointi noudattaa yrityksen sisäisiä käytäntöjä, mutta yleensä jokin suurehko tekninen arkkitehtuuri- tai muu muutos aiheuttaa julkaisun päänumeron nostamisen. Jokainen julkaisu koostuu joukosta *vaatimuksia* (engl. requirements), jotka ovat lisäyksiä tai muutoksia tuotteen teknisiin tai toiminnallisiin ominaisuuksiin. (van de Weerd ym., 2006, 2.) Eri tasoilla sijaitsevien käsitteiden työstäminen eroaa toisistaan ja tästä voidaan johtaa tuotehallinnan neljä prosessialuetta (van de Weerd ym., 2006, 2):

- *tuotesalkun hallinnan* (engl. portfolio management) tehtävänä on hoitaa salkussa olevia tuotteita
- *tuotteiden roadmapping* (engl. product roadmapping) suunnittelee jokaisen tuotteen erilliset julkaisut
- *vaatimusten hallinta* (engl. requirements management) käsittelee jokaisen erillisen vaatimuksen

- *julkaisujen suunnittelussa* (engl. release planning) hoidetaan jokaiseen julkaisuun valitut vaatimuskokoelmat.

Tuotepäälliköt joutuvat käsittelemään ja arvioimaan suuren määrän vaatimuksia, joiden alkuperänä on joko sisäisiä tai ulkoisia sidosryhmiä. Ainakin seuraavat sisäiset sidosryhmät ovat tunnistettavissa (van de Weerd ym., 2006, 2):

- *Johto* (engl. company board) on vastuussa yrityksen strategian, vision ja mission määrittelystä ja viestinnästä.
- *Tutkimus- ja innovaatio-osasto* (engl. research & innovation) etsii uusia tapoja sisällyttää parannuksia tai uusia ominaisuuksia tuotteisiin.
- *Palveluosaston* (engl. services) konsultit ovat vastuussa ohjelmistotuotteen käyttöönotosta asiakasorganisaatiossa.
- *Kehitysosasto* (engl. development) on päävastuussa julkaisusuunnitelman toimeenpanosta.
- *Tukipalvelu* (engl. support) vastaa asiakkaiden kysymyksiin (helpdesk) ja selvittää mahdolliset pienet viat.
- *Myynti ja markkinointi* (engl. sales & marketing) on ensimmäinen kontakti potentiaaliseen asiakkaaseen.

Seuraavat ulkoiset sidosryhmät voidaan tunnistaa (van de Weerd ym., 2006, 2):

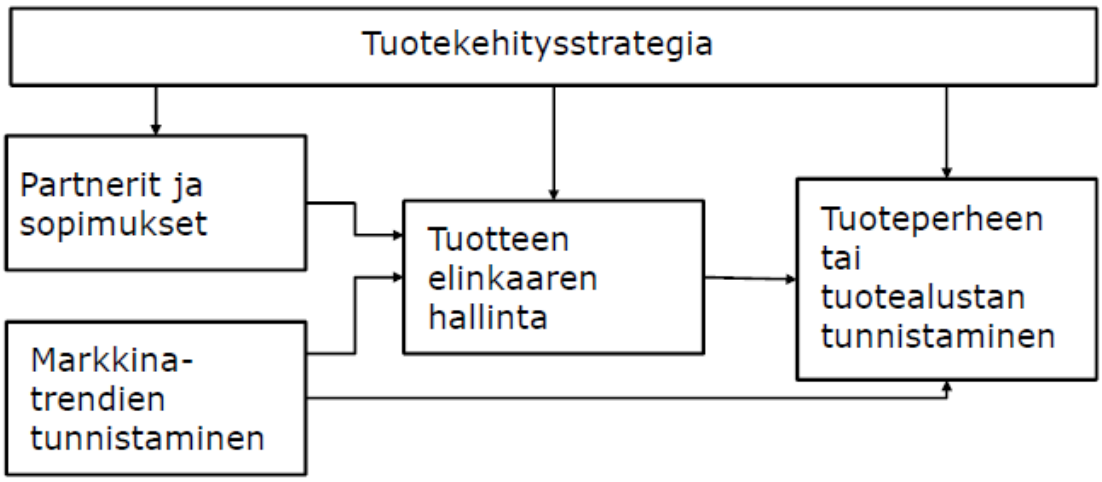
- *Markkinat* (engl. market) on abstrakti sidosryhmä, joka käsittää potentiaaliset asiakkaat, kilpailijat ja analyytikot.
- Useimmilla yrityksillä on useita erilaisia *kumppaneita* (engl. partners), kuten esim. käyttöönotto-, kehitys-, ja jakelukumppaneita.
- *Asiakkaat* (engl. customers) esittävät usein pyyntöjä uusista ominaisuuksista, jotka voidaan ohjata myyntiin ja markkinointiin, tukipalveluihin tai suoraan tuotepäällikölle.
- *Lainsäädäntö* (eng. legislation) tuottaa uusia vaatimuksia, joihin on reagoitava hyvinkin nopeasti.

Mainitut sidosryhmät ovat yleisellä tasolla ja niiden nimeäminen ja jaottelu voi vaihdella ohjelmistoyrityksestä toiseen. Pienessä yrityksessä sisäiset sidosryhmät voivat myös lomittua ja koostua samoista henkilöistä.

Seuraavissa kappaleissa tarkastellaan kutakin tuotehallinnan prosessialuetta yksityiskohtaisemmin.

3.1 Tuotesalkun hallinta

Tuotekehityksen toimintojen hallinta erityisen tuotesalkun avulla on elintärkeää tuotteita valmistavan yrityksen pitkän aikavälin menestymisen kannalta (Vähäniitty & Rautiainen, 2005). Tuotesalkun hallinta on esitetty kuviossa 2:



KUVIO 2 Tuotesalkun hallinta (mukaillen van de Weerd ym., 2006)

Tuotesalkun hallinnassa tehdään päätöksiä olemassa olevista tuotteista, esitetään valmistettavaksi uusia tuotteita perustuen markkinatrendien tutkimiseen ja tuotestrategiaan, päätetään ohjelmistojen elinkaaresta ja hankitaan sekä ylläpidetään kumppaneita ja sopimuksia. Tuotesalkun hallinta koostuu neljästä pääprosessista: *partnerit ja sopimukset* (engl. partnering & contracting), *markkinatrendien tunnistaminen* (engl. market trend identification), *tuotteen elinkaaren hallinta* (engl. product lifecycle management) ja *tuoteperheen tunnistaminen* (engl. product line identification). Nämä prosessit saavat syötteensä yrityksen johdolta, markkinoilta ja kumppaneilta. (van de Weerd ym., 2006.) Jos yrityksen tuotekehitys ei perustu tuoteperheisiin, voidaan sen paikalle nostaa *tuotealustan* (engl. product platform) tunnistaminen. Tuotealusta sisältää kaikille tuotteille yhteiset, uudelleenkäytettävät *ydinkomponentit* (engl. core assets).

Vähäniitty ja Rautiainen (2005) ovat tutkineet pienten yritysten tuotesalkun hallintaa ja esittävät artikkelissaan niille soveltuvan perusmenetelmän, joka yhdistää strategiset linjaukset, salkun tasapainottamisen ja jatka/lopeta/pidä (go/kill/hold) –tyyppisen päätöksenteon nykyaikaiseen, nopeaan ohjelmistokehitykseen. Tutkimustulostensa perusteella he ovat valmiita esittämään, että menetelmän käyttö nostaa tietoisuutta projektien ja muiden kehityshankkeiden tilasta ja vaiheesta sekä niiden resursointitarpeista. Menetelmä auttaa myös tekemään perusteltuja päätöksiä ja tarpeellisia kompromisseja tuotteiden suhteen. Vähäniityn (2003) mukaan menestyminen ohjelmistotuoteliiketoiminnassa vaatii ”ison kuvan” (engl. big picture) näkemistä ja tuotteiden valmistuminen oikeaan aikaan oikeilla ominaisuuksilla varustettuna on erittäin tärkeää. Ohjelmistotuoteperheiden hallinnassa käytetään tuotesalkkua melko kapeasti; esimerkiksi työkaluna, jonka avulla voidaan määritellä ominaisuuksia ja piirteitä, jotka tuoteperheessä tulisi pystyä huomioimaan. Tuotesalkkua voidaan kuitenkin käyttää myös dynaamiseen ja liiketoimintalähtöiseen resurssiallokointiin tai kehityspanosten priorisointiin. (Vähäniitty, 2003.)

Vähäniitty ja Rautiainen tutkivat (2005) kolmea pientä, ohjelmistotuotteita valmistavaa yritystä, joilla ei ollut käytössään tuotesalkun hallintaa, joiden tuotteille ei ollut laadittu pitkän aikavälin suunnitelmia eikä kehitystoimenpiteitä ollut priorisoitu. Ei siis ollut yllättävää, että kaikilla näillä yrityksillä oli huonoja kokemuksia tehtävistä, jotka olisi voinut hoitaa tuotesalkun hallinnan avulla ja kaikki olivat sitä mieltä, että heidän prosessinsa kaipasivat kohentamista tällä alueella. Tärkeät kehityspäätökset tehtiin usein avainhenkilöiden mielipiteisiin pohjautuen ilman keskustelua tai perusteluita. Joskus päätöksiä ei tehty tarkoituksella, vaan jopa tahattomasti tai jättämällä asioita hoitamatta. Lisäksi avainhenkilöstö ei yleensä ymmärtänyt tekevänsä tärkeitä päätöksiä hoitaessaan normaaleja kehitystoimia. Tulorahoituksella toimiva ohjelmistokehitys saattaa tarvita lisätuottoja lisenssimyynnin ohella. Niinpä kaikissa näissä yrityksissä ohjelmistokehittäjien huomio jakaantui uusien tuotejulkaisujen, asiakaskohtaisen räätälöinnin, asiakastoimitusten ja muiden palveluiden, kuten konsultoinnin kesken. Kaikissa yrityksissä myös tehtiin asiakaskohtaista kehitystä ajatuksella, että valmistunut osio liitetään myöhemmin varsinaisen tuotteen tulevaan julkaisuun. Eri toimintoja ei johdettu välttämättä edes projekteina, eikä niitä tunnistettu tuotesalkkuun kuuluviksi, jolloin resurssisuunnittelu oli hankalaa tai se jätettiin turhana kokonaan tekemättä. Tärkeimpänä pontimena kehitystoimiin ryhtymiseen oli halu ymmärtää kokonaiskuva eli saada aikaan yhteinen käsitys siitä, mitä projekteja ja hankkeita oli käynnissä, kuinka ne oli resursoitu ja mikä niiden keskinäinen tärkeysjärjestys oli. Yrityksissä kärsittiin resurssien päällekkäisvarauksista eikä uudelleenkohdistaminen tai -priorisointi ollut mahdollista kokonaiskuvan puuttumisen takia. Sen sijaan kehittäjät joutuivat luottamaan omaan harkintaansa valitessaan tai ohittaessaan tehtäviä, mikä aiheutti myöhemmin projektin kuluessa yllätyksiä, jos jokin tärkeä tehtävä oli jäänyt aiemmin huomioimatta. Yrityksissä ei ollut myöskään ymmärrystä siitä, mitä yhden projektin tasolla tehty päätös aiheutti muille projekteille levitessään koko salkun alueelle, ja josta taas seurasi kokonainen uusien päätösten vyöry. Toinen alue, jossa kaivattiin parannusta, oli liiketoiminnan (eli ylimmän johdon ja myynnin) ja tuotekehityksen välisen keskustelun käyminen. Tuotekehityksen suunnasta ja pitkän tähtäimen suunnitelmista ei keskusteltu liiketoiminnan näkökulmasta. Toisaalta tätä ei pidetty vakavana ongelmana, jos suuri osa yrityksen tuotoista tuli palvelutuotannosta. Haittapuolena pidettiin sitä, että keskustelut aiheuttivat impulsiivisuutta uusien projektien aloittamiseen ja poukkoilevuutta tärkeysjärjestykseen, jolloin vanhat projektit eivät koskaan valmistuneet uusien yliajamina. (Vähäniitty & Rautiainen, 2005.)

Tekemässään kirjallisuuskatsauksessa Vähäniitty ja Rautiainen (2005) ovat vertailleet ison ja pienen tuotekehitystä tekevän yrityksen tuotesalkkujen hallintaan liittyvän tutkimuksen välisiä eroja. Ison yrityksen tuotesalkun hallinnassa on kysymys päätöksenteosta monien kehitysprojektien ja niihin liittyvien tuotelinjojen ja useiden liiketoimintayksiköiden suhteen/välillä. Pienessä yrityksessä sen sijaan tuotesalkun hallinta painottuu muutamien tuotteiden tulevien julkaisujen sisältöön ja erilaisten kehitykseen liittyvien (tai jopa liittymättömien) aktiviteettien resursointiin. Näiden aktiviteettien hoitamiseen tarvitaan pienessä

organisaatiossa tuotekehityksen henkilöstöä, sillä heillä on monia vastuita ja rooleja ja resurssien suunnittelu ja allokointi on ailahtelevaa. Pienen yrityksen resurssien vähäisyys on absoluuttista eli sen ei ole mahdollista selviytyä lähestyvästä deadlinesta työmäärää ja resursseja lisäämällä, kuten suuressa organisaatiossa. Toisaalta pienen yrityksen ylemmät toimihenkilöt ovat koko ajan mukana käytännön työssä, joten heillä on jatkuvasti hyvä käsitys kokonaiskuvasta. Tämä ei kuitenkaan poista erityisten liiketoiminnallisten päätöksentekopisteiden tarvetta, sillä muutoin toimihenkilöt tekevät epähuomiossa isojakin tuotesalkkuun liittyviä päätöksiä. Päätöksen teon vaiheistus pitää kuitenkin sovittaa organisaation kokoon ja tarpeisiin. Yleisenä huomiona Vähäniitty ja Rautiainen (2005) toteavat, että tuotesalkkuja koskeva tutkimus on keskittynyt isojen organisaatioiden ongelmiin ja saatuja tuloksia ei voi sellaisenaan hyödyntää pienissä yrityksissä.

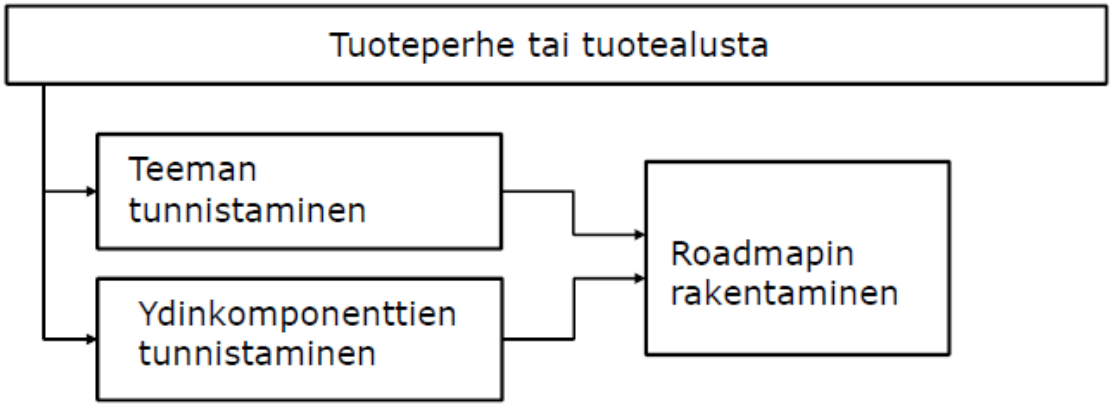
Cooperin, Edgettin ja Kleinschmidtin (2002, 358) mukaan menestyksekäs tuotesalkun hallinta tarkoittaa tasapainon saavuttamista neljän jopa ristiriitaisen tavoitteen välillä:

- Salkun liiketoiminnallisen arvon maksimointi
- Salkun liittäminen strategiaan
- Salkun tasapainottaminen
- Meneillään olevien hankkeiden määrän pitäminen toteutusmahdollisuuksien rajoissa.

Käytännössä salkun hallinta tapahtuu yhdistämällä projektien arvioinnin, valinnan ja priorisoinnin tekniikat menossa olevien projektien vaiheittaiseen katselmointiin. Katselmointiprosessi järjestää tuotekehitysprojektit peräkkäiseen joukkoon vaiheita, joilla on omat teemat, alkaen ideoinnista ja päättyen tuotejulkistukseen ja ylläpitoon. Kunkin vaiheen lopussa on liiketoiminnan ja priorisoinnin päätöksentekopiste eli katselmointi. Kehitystyötä ohjataan vaiheen aikana ja samalla kerätään tarvittavaa tietoa seuraavan katselmoinnin läpäisyyn. Cooper ym. (2002) esittävät kaksi perusvaihtoehtoa tuotesalkun hallintaprosessien toteuttamiseksi käytännössä. Ensimmäinen painottaa päätöksentekoa syvällisten, jokaisen menossa olevan projektin katselmointien kautta, jolloin salkun hallinnointi tapahtuu alhaalta ylöspäin. Toinen vaihtoehto toimii ylhäältä alaspäin ja siinä päätökset tehdään tutkimalla tuotesalkkua kokonaisuutena. Ensimmäinen vaihtoehto sopii paremmin isoille yrityksille, joilla on tarvittavat resurssit ja suhteellisen pysyvät salkut ja joiden on mahdollista tehdä yksittäisten projektien suhteen varmoja jatka/lopeta-päätöksiä. Toinen taas soveltuu paremmin pienille, nopeatempoisella ja joustavalla markkina-alueella toimiville yrityksille, jotka pystyvät ajoittaisilla koko salkun katselmoinneilla allokoimaan resurssinsa dynaamisemmin. (Cooper ym., 2002.)

3.2 Roadmapping

Van de Weerdin ym. (2006) viitekehyksessä roadmapping-prosessi (kuvio 3) saa syötteen tuotesalkun puolelta ohjelmistotuoteperheiden tai tuotealustan hallinnasta. Tätä lähtötietoa käytetään *tunnistamaan teema* (engl. theme identification) ja *ydinkomponentit* (engl. core asset identification), joita taas käytetään myöhemmin vaatimusten järjestämisessä. Näin kerätystä ja kuvatusta informaatiosta *rakennetaan tuotteiden roadmap* (engl. roadmap construction). (van de Weerd ym., 2006.)



KUVIO 3 Tuotteen roadmapping (mukaillen van de Weerd ym., 2006)

Lehtola, Kauppinen ja Kujala toteavat tutkimuksessaan (2005), että tuotestrategiasta kumpuavat kehitystarpeet ovat niin korkealla abstraktilla tasolla, että kuilu yksittäisiin tuotevaatimuksiin on aivan liian leveä. Roadmapping on tekniikka, jonka avulla kuilua voidaan kaventaa linkittämällä liiketoiminnan näkemys vaatimusten hallintaan ja näin tehdä tuotekehityksessä parempia liiketoimintalähtöisiä ratkaisuja. (Lehtola ym., 2005.) Asiakkaat haluavat olla varmoja, että tuotteella, johon he luottavat liiketoiminnassaan, on tulevaisuus, joka on linjassa heidän omien suunnitelmiensa kanssa. Erityisesti tilanteessa, jossa ohjelmistotoimittajan vaihtamisen kustannukset ja seuraukset ovat suuret ja laajat, asiakas haluaa osallistua tuotteen tulevaisuuteen liittyvään päätöksentekoon (Regnell ym., 2005). Roadmappingia voidaan siis käyttää luomaan sidosryhmien välille yhteinen käsitys tulevista tuotekehitysaskelista. Liiketoimintatavoitteet ja markkinointiponnistelut on helpompi linkittää korkean tason ominaisuuksiin kuin yksittäisiin ja melko pieniin tuotevaatimuksiin. Näin myynti ja markkinointi voivat ajoittaa omat toimintonsa alkamaan yhtäaikaisesti kun tuotekehitystä vasta tehdään. (Lehtola ym., 2005.)

Vähäniityn, Lasseniuksen ja Rautiaisen (2002, 2) mukaan tuotteen roadmapping on prosessi, joka koostuu tieteellisten ja teknisten resurssien ja elementtien sekä niiden välisten rakenteiden suunnittelusta ja kuvaamisesta tiettynä ajanjaksona. Tämä toiminta on monimutkaista, koska siinä on otettava huomioon esimerkiksi riippuvuudet toisiin tuotteisiin (jopa partnereiden tuot-

teisiin), teknologiamuutokset ja hajautettu ohjelmistokehitys. Roadmappingin juuret ovat valmistavassa teollisuudessa, missä sitä käytettiin liiketoiminnan pitkän aikavälin suunnitteluun ja teknologian ennustamiseen. Tyypillinen roadmapping-prosessi kokoaa yhteen organisaation eri sidosryhmät suunnittelemaan ja päättämään tuotteen tulevaisuudesta ja jonka päätösten tuloksena syntyy tuotteen roadmap. Roadmapin pitäisi kuvata tehokkaasti ja visuaalisesti tuotteen ennustettu kehityspolku 2-3 vuoden aikajaksolle. (Lehtola ym., 2005.) Kappel (2001) painottaa kuitenkin, että roadmapilla on kaksijakoinen luonne: se on sekä ennuste että suunnitelma. Ennusteella hän tarkoittaa, että roadmap kuvaa sen, mitä todennäköisesti tulee tapahtumaan ja suunnitelmalla sitä, että roadmap kertoo myös toimintatavat, joilla ennusteeseen pyritään.

Vähäniitty, Lassenius ja Rautiainen (2002) esittävät artikkelissaan visuaalisen tavan kuvata pienen yrityksen roadmapia. Esimerkki erään tuotteen roadmapista on liitteessä 2. He yhdistävät samaan kaavioon tuotteen julkaisujen ja kehittämisen aikataulut, yksittäisten julkaisujen sisällön, muutokset perusteknologiaan, palvelut, joissa tarvitaan kehittäjien apua sekä suunnitelman resurssitarpeesta. Kaikki nämä aktiviteetit muodostavat kaavioon viisi eri kerrosta ja toiminnot piirretään kerroksiin erivahvuisilla ja -värisillä palkeilla aikajanalle. Toimintojen väliset riippuvuudet kuvataan nuolilla ja palkin paksuus kuvaa toiminnon resurssitarvetta. Kirjoittajien mielestä myös palveluiden sijoittaminen roadmapiin on tärkeää, koska pienessä yrityksessä kehittäjien työpanosta tarvitaan esimerkiksi asiakaskohtaisten muutosten tekemiseen. Projektien johtaminen ja resurssien hallinta on helpompaa, kun kokonaiskuva on näkyvissä. (Vähäniitty ym., 2002.)

Tutkimuksessaan Lehtola ym. (2005) suosittavat kahden erillisen roadmapin laatimista: julkaisujen roadmap ja tuotteen roadmap. Nämä roadmapit ovat melko samanlaisia, mutta julkaisujen roadmap on paljon yksityiskohtaisempi ja tarkemmalla tasolla sisältäen mm. julkaisuun kuuluvat vaatimukset. Tuotteen roadmap säilytetään ominaisuustasolla ja sen avulla on helpompi nähdä kokonaiskuva tulevasta kehityksestä. Tutkijat tekivät muutamia tärkeitä havaintoja roadmappingin käytöstä suomalaisessa tuotekehitysyrityksessä (Lehtola ym., 2005, 4):

- **Roadmapping vahvisti liiketoimintapäätösten ja vaatimustenhallinnan välistä yhteyttä**. Julkaisujen roadmap toimi hyvänä lähtötilanteena tuotekehitysprojekteille, sillä julkaisun tavoite pystyttiin kääntämään vaatimustenhallinnan avulla yksittäisten projektien tasolle.
- **Toimijoiden mielestä roadmapit vanhentuivat hetkessä.** Roadmap oli valmistuessaan pätevä, mutta myöhemmin uudet asiakastarpeet ja resursointiongelmat aiheuttivat sen, että muutoksia ei enää dokumentoitu roadmapin tasolle.
- **Toimijoilta puuttui tapa yhdistää tuotekehitysresurssit roadmapeihin.** Kehitysresurssien allokointi yhtä aikaa ja etukäteen roadmapin suunnitelmiin koettiin haastavaksi. Kehittäjillä on erilaisia taitoja, joi-

den saaminen tietyn projektin käyttöön etukäteissuunnitelman perusteella aiheutti ristiriitoja muiden projektien kanssa.

- **Toimijat laativat roadmapit vaadittua lyhyemmälle ajanjaksolle.** Tuotepäälliköiden mielestä ei ollut kannattavaa laatia tuotteen roadmapia koko kolmelle vuodelle, koska kohderyhmät ja käyttäjähyödyt eivät muutu joka vuosi. Myös julkaisujen roadmapit laadittiin ulottumaan vain yhden tai kahden tulevan julkaisun päähän, koska toimijoiden mielestä heillä ei ollut tarpeeksi tietoa pystyäkseen päättämään usean vuoden julkaisusta etukäteen.

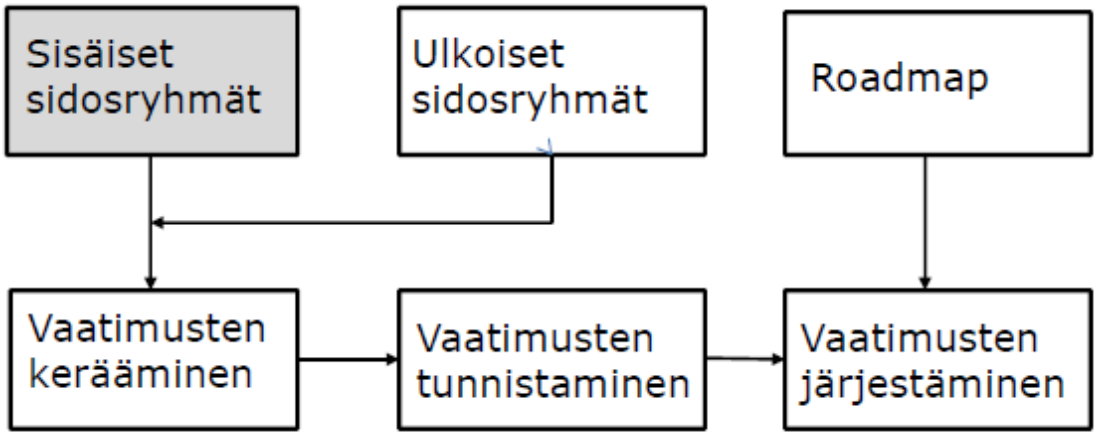
Hyvän ja toimivan roadmapin laatiminen ei siis ole helppoa. Liian yksityiskohtainen ja tekninen roadmap häivyttää näkyvistä liiketoiminnan näkökulman, kun taas karkean tason yleinen roadmap ei ole käyttökelpoinen tulevassa kehitystyössä. Roadmapping-prosessin aloittaminen ja käynnissä pitäminen ei ole yksinkertaista eikä sitä missään nimessä pitäisi antaa yhden henkilön (tuotepäällikön) tai osaston tehtäväksi eikä käyttää budjetoinnin perusteena. Nämä Lehtolan ym. (2005) havainnot tukevat myös Phaalin, Farrukin ja Probertin (2003) näkemystä, että suurimmat roadmappingin ongelmat ovat organisatorisia; vankan roadmapping-prosessin kehittäminen, sen käynnistäminen ja pitäminen käynnissä.

Roadmappingin hyöty saavutetaankin käyttämällä sitä työkaluna, jolla laajennetaan näkemystä lyhyen aikavälin suunnittelusta pidemmälle ulottuvaan ajatteluun. Roadmapping-prosessi myös kokoaa yhteen erilaiset toiminnalliset tiimit, jolloin niiden yhteistyö ja sitoutuminen suunnitelmiin paranevat. Myös yhteinen sanasto ja jokaisen erityisosaaminen saadaan käyttöön unohtamatta sitä tosiasiaa, että kokonaiskäsitys ongelmista, tiedosta, mahdollisuuksista ja uusista ideoista saadaan jaettua kaikille osallistujille. (Phaal ym., 2003.) Yhteinen matka (roadmapping-prosessi) on siis tärkeämpi kuin saavutettu päämäärä (roadmap).

3.3 Vaatimusten hallinta

Vaatimusten hallinta käsittää toiminnot, joilla kerätään, tunnistetaan, tarkistetaan ja järjestetään eri sidosryhmiltä tulevat ohjelmistoon liittyvät vaatimukset. Vaatimusten hallinta on avainasemassa ohjelmistotuotteita valmistavassa yrityksessä (Carlshamre & Regnell, 2000). Erityisesti vaatimusten analysointi vaatii paljon aikaa johtuen siitä, että uusia vaatimuksia tulee jatkuvasti lisää. Van de Weerdin ym. (2006) viitekehyksessä (kuvio 4) vaatimusten hallinta alkaa kaikkien *vaatimusten kokoamisella* (engl. requirements gathering) yrityksen sisältä ja ulkoisilta sidosryhmiltä. Nämä *tunnistetaan* (engl. requirements identification) tai muunnetaan tuotevaatimuksiksi eli vaatimuksiksi, jotka tullaan toteuttamaan tuotteeseen. Muuntaminen tapahtuu poistamalla duplikaatit, yhdistämällä samaa toiminnallisuutta kuvaavat vaatimukset ja kirjoittamalla ne uudestaan.

Tämän jälkeen *vaatimukset järjestetään* (engl. requirements organizing) tuotteittain ja ydinosittain.



KUVIO 4 Vaatimusten hallinta (mukaillen van de Weerd ym., 2006)

Sawyer (2000) tiivistää, että suurimmat erot markkina- ja asiakaslähtöisen tuotekehityksen vaatimustenhallinnassa ovat sidosryhmien ominaisuuksissa ja aikataulujen asettamissa rajoitteissa. Tärkein sidosryhmä on itse toteuttava organisaatio. Yrityksessä toimivat kehittäjät *keksivät* vaatimukset, koska ei ole olemassa erillistä asiakasta, joka osaisi muotoilla omat vaatimuksensa. Samanaikaisesti paine saada ohjelmistotuotteita markkinoille on kova. Kuitenkin myös laaja markkina-alue ja asiakkaat yrityksen ulkopuolella sekä useat sidosryhmät yrityksen sisällä on otettava huomioon tuotekehityksessä eli tuotteen tulevia kehitysaskeleita ei voi suunnitella vain muutaman asiakkaan ja/tai sidosryhmän kanssa. Joka tapauksessa päätökset on tehtävä strategisesti yrityksen sisällä ja organisaatio kantaa kaikki niihin liittyvät taloudelliset riskit. (Sawyer, 2000.) Lisäksi tuotekehityksen ulkopuoliset sidosryhmät tarvitsevat tietoa tulevasta kehityksestä pystyäkseen suunnittelemaan esimerkiksi omia myynti- ja markkinointitoimenpiteitään (Lehtola ym., 2005).

Ruotsalaiset tutkijat Karlsson, Dahlstedt, Regnell, Natt och Dag ja Persson (2007) tekivät haastattelututkimuksen 8 pienen ja keskisuuren markkinalähtöistä kehitystä tekevän yrityksen parissa. Tutkimuskysymyksenä oli ”Mitä haasteita yritykset kohtaavat vaatimusten hallinnan alueella?”. Seuraavaksi katsotaan heidän löytämiänsä vastauksia selityksineen (Karlsson ym., 2007, 593-597):

- **Yksinkertaiset tekniikat perustarpeisiin.** Pienillä yrityksillä on vaikeuksia löytää prosesseja ja tekniikoita, jotka sopivat niiden tarpeisiin. Yritykset eivät halua ottaa käyttöön laajoja markkinoilla olevia vaatimushallintaa tukevia ohjelmistoja niiden monimutkaisuuden ja koulutustarpeen vuoksi. Työkalut ovat myös liian haastavia asiakkaiden kanssa tapahtuvaan kommunikointiin. Lisäksi testaus- ja vaatimushallinnan työkalujen integrointi koetaan hankalaksi.

- **Kommunikaatiokuilu markkinointihenkilökunnan ja kehittäjien välillä.** Yhteistyö organisaation eri osien ja erityisesti markkinoinnin ja kehittäjien välillä ontuu. Markkinoinnin mielestä vaatimusten määrittely tarkoittaa ideoiden heittelyä tulevista toiminnoista, kun taas kehittäjät haluaisivat selviä ja yksityiskohtaisia vaatimuksia, joiden perusteella työn voi suunnitella. Ratkaisuna ongelmaan jotkut yritykset käyttävät ”välimiestä” tai tulkkia (tuotepäällikkö), joka muokkaa ideat järjestelmänhallinnan kielelle. Myyntihenkilökunta saattaa myös luvata asiakkaalle jonkin uuden ominaisuuden tuotteeseen, ennen kuin he selvittävät kehittäjiltä, onko sitä edes mahdollista toteuttaa.
- **Ymmärrettävien vaatimusten laatiminen**. Useimmat yritykset käyttävät luonnollista kieltä vaatimusten kirjoittamiseen, sillä se on paras tapa kommunikoida asiakkaiden kanssa. Yrityksen sisällä tarvitaan kuitenkin eri tavoin laadittuja vaatimusmäärittelyjä riippuen henkilön työtehtävästä. Myös termistöjen ja käsitteiden sekavuus aiheuttaa ongelmia. Kaikille osapuolille selkeät ja ymmärrettävät vaatimukset saadaan aikaan vain yhteisten keskustelujen avulla.
- **Uusien vaatimusten tulvan hallitseminen**. Sidosryhmiltä tulevien uusien potentiaalisten ideoiden kerääminen koetaan tärkeäksi ja sidosryhmät myös odottavat saavansa palautetta esittämistään ehdotuksista. Kaikilla tutkimuksen yrityksillä on käytössään jonkinlainen varasto, jonne ideat kerätään ja josta sitten valikoidaan toteutuskelpoiset seuraavaan julkaisuun. Valikoinnin tuloksena saattaa kuitenkin olla, että jokin kiinnostava idea hukkuu massaan. Jos asiakkaat saavat itse kirjoittaa ideoita varastoon, on jonkun työntekijän (tuotepäällikkö) selvennettävä niitä ja poistettava ristiriitaisuudet ja päällekkäisyydet. Tämä ja palautteen antaminen asiakkaille on erittäin työllistävää.
- **Vaatimusten epävakaus**. Vaatimukset muuttuvat kesken toteutuksen monista syistä; asiakas muuttaa mielensä, markkinatilanne muuttuu tai törmätään teknisiin ongelmiin. Ratkaisuna tilanteeseen käytetään esimerkiksi ketterää kehitystä tai tuote esitellään sidosryhmille kommentointia varten puolivalmiina, jolloin muutosten teko helpottuu. Myös varsinaisten vaatimusmäärittelyjen muuttamiseen kaivataan yksinkertaista työkalua.
- **Vaatimusten keskinäiset riippuvuudet**. Toisistaan riippuvat vaatimukset pyritään hoitamaan niputtamalla, ts. samaan osaan tuotetta tai saman ohjelmoijan tehtäväksi tulevat erilliset vaatimukset yhdistetään. Hankala ongelma ovat myös ei-toiminnalliset laatuvaatimukset, jotka vaikuttavat joka puolelle tuotetta tai pikemminkin hankalaa on löytää tapa, kuinka ne vaikuttavat. Päällekkäiset vaatimukset aiheuttavat myös päänvaivaa, sillä samasta ominaisuudesta voi olla useita hieman erilaisia ja jopa ristiriitaisia tai toisensa poissulkevia vaatimuksia.
- **Sisäiset vs. ulkoiset vaatimukset.** Yrityksissä tekniset ja innovatiiviset vaatimukset tulevat usein yrityksen sisältä ja ei-toiminnalliset vaatimukset ulkoisilta sidosryhmiltä. On tärkeää löytää tasapaino asiak-

kaiden tarpeiden ja uusien innovaatioiden välille. Jos yritys käyttää jälleenmyyjiä, ei vaatimuksia ole helppoa saada suoraan asiakkailta. Tärkeän asiakkaan on myös helpompi saada omat vaatimuksensa läpi. Toisaalta yhden asiakkaan esittämän vaatimuksen muuntaminen tarpeeksi yleiseksi niin, että sen toteuttaminen hyödyttäisi laajempaa asiakaskuntaa, ei ole aina yksinkertaista.

- **Vaatimusten määrittelyn ja hallinnan toteuttaminen ja parantaminen yrityksen sisällä.** Uusien prosessien käyttöönotto vaatii henkilökunnalta sitoutumista ja muutosvastarintaa esiintyy aina. Vastuuhenkilöiden on järjestettävä koulutusta ja huolehdittava siitä, että uudet sovitut käytännöt otetaan käyttöön. Aikaisemmista projekteista on myös hyvä ottaa oppia.
- **Resurssien ohjaaminen vaatimusmäärittelytyöhön.** Yrityksissä on vaikea osoittaa resursseja vaatimusten määrittelyyn. Osaksi tämä johtuu siitä, että johto ei ymmärrä, kuinka aikaa vaativaa työ on ja osaksi siksi, että projektit ja määrittelytyö aloitetaan yhtä aikaa, jolloin avainresurssit menevät projektille. Kunnollisten vaatimusmäärittelyjen aikaansaamiseksi työ pitäisi aloittaa ja lopettaa ennen varsinaisen ohjelmointiprojektin alkamista.
- **Organisaation vakaus.** Pienen yrityksen tietopääoma on sen työntekijöissä. Jos avainhenkilö lähtee, menetetään paljon arvokasta tietotaitoa ja osaamista. Projektit saattavat jopa vaarantua tällaisessa tapauksessa, koska projektien läpiviemiseen tarvitaan kaikkien panosta ja osaamista kehitysprosessin eri vaiheissa.
- **Oikean prosessin valitseminen.** Yritykset kamppailevat oikeanlaisen vaatimusmäärittelyprosessin aloittamisen, kehittämisen ja säätämisen kanssa. Prosessin pitäisi olla kattava, mutta ei liian yksityiskohtainen, jotta sitä noudatettaisiin. Prosessin kautta kuitenkin kommunikoidaan eri sidosryhmien kanssa ja siksi rakenteen täydellinen puuttuminen hankaloittaa yrityksen toimintaa. Prosessin pitäisi kuitenkin olla tarpeeksi mukautuva eri tilanteita varten ja sen täytyy myös olla yhteensopiva muiden prosessien kanssa.
- **Julkaisujen suunnittelu perustuu epävarmoihin ennusteisiin**. Julkaisua varten vaatimukset priorisoidaan ja niiden työmäärät arvioidaan. Vaikka yritykset pitivät työmäärien arviointia erittäin tärkeänä, yksikään ei käyttänyt arviointiin mitään formaalia menetelmää. Työmäärät arvioitiin esim. sidosryhmien välisissä keskusteluissa, joissa pyrittiin pääsemään yksimielisyyteen, mutta kehittäjät kuitenkin antoivat lopulliset arviot. Työmäärien arviointia kuitenkin helpotti se, että kehittäjät saivat keskusteluissa arvokasta tietoa tehtävästä ominaisuudesta. Arviointia ei pysty tekemään, jos ei tunne koko järjestelmää ja uuden ominaisuuden vaikutuksia siihen.

Monet näistä löydöksistä soveltuvat sekä asiakas- että markkinalähtöisen sovelluskehityksen vaatimusmäärittelyn ja vaatimusten hallinnan haasteiksi. Muu-

tamat niistä ovat kuitenkin tyypillisesti enemmän markkinalähtöisen kehityksen ominaisuuksia eli ne johtuvat laajasta asiakaskunnasta ja käyttäjien määrästä. Tämä erityispiirre aiheuttaa jatkuvan vaatimusten tulvan ja vaikeuttaa kaikille osapuolille ymmärrettävien vaatimusten kirjoittamista. Kehittäjien ja asiakkaiden välillä ei ole myöskään suoraa yhteyttä, vaan väliin tarvitaan jokin muu sidosryhmä (tuotepäällikkö) dokumentoimaan ja valikoimaan vaatimuksia. Asiakkailta tulevat ideat ovat vaatimusten pääasiallinen lähde, mutta yrityksen sisällä kehitetyt innovatiiviset ideat ovat myös tärkeitä. Ne eivät vaan saa ohittaa asiakkaiden tarpeita. Julkaisujen suunnittelu on erittäin tärkeää markkinoilla vallitsevan kilpailun takia. Jotta käyttäjille voidaan tarjota oikeita tuotteita oikeaan aikaan, on vaatimusten priorisoinnin ja työmäärien arvioinnin oltava kunnossa. (Karlsson ym., 2007.)

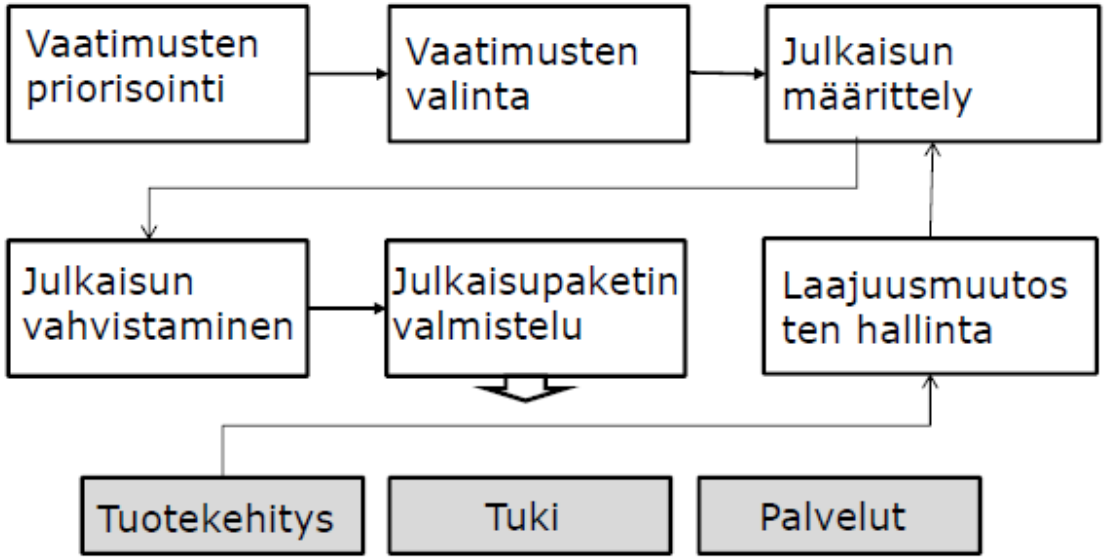
Vaatimusten hallinta on tuotepäällikön ydintehtäväaluetta. Hänen toimenpiteitään tarvitaan jokaisessa vaiheessa, kun vaatimuksia vastaanotetaan, käsitellään ja valitaan julkaistavaksi. Tuotepäällikkö ei kuitenkaan voi yksin hoitaa kaikkia tehtäviä, vaan hänen on otettava työhön mukaan myös muut sidosryhmät. Myös roadmapin asettamat suuntaviivat on pidettävä mielessä koko ajan.

3.4 Julkaisujen suunnittelu

Julkaisujen suunnittelu on prosessi, jonka kautta ohjelmisto valmistellaan käyttäjien saataville ja käyttöön (van der Hoek, Hall, Heimbigner & Wolf, 1997, 4). Suunnitteluprosessissa vaatimusten joukko kootaan seuraavaa julkaisua varten. Van de Weerdin ym. (2006) viitekehyksessä (kuvio 5) julkaisun suunnittelu alkaa *vaatimusten priorisoinnilla* (engl. requirement prioritization). Sen jälkeen *vaatimuksista valitaan* (engl. requirement selection) ne, jotka toteutetaan seuraavaan julkaisuun. Kun tuotevaatimukset on valittu, kirjoitetaan *julkaisun määrittely* (engl. release definition), jonka eri sidosryhmät vahvistavat. Määrittely lähetetään yrityksen johdon *vahvistettavaksi* (engl. release validation), *julkaisun valmistelupaketti* (engl. launch preparation) rakennetaan ja lähetetään sidosryhmille. Tuotekehityksen prosessista saattaa tulla tarpeita julkaisun laajuuden tarkistamiseen ja se hoidetaan erillisessä *laajuusmuutosten hallinnan* (engl. scope change management) prosessissa. (van de Weerd ym., 2006.)

Julkaisun suunnittelussa vaatimusten asettaminen tärkeysjärjestykseen vaatii monien eri näkökohtien huomioon ottamista. Yrityksillä on arviointia varten käytössään erilaisia lähestymistapoja, kuten kustannus/saatu arvo, käyttäjälle tuotettu arvo, kehittämiskustannukset, strateginen arvo tai asema markkinoilla. Myös muut näkökohdat, kuten vaikutus arkkitehtuuriin, käyttöönoton riskit, keskinäiset riippuvuudet ja aikarajat, on otettava huomioon. (Karlsson ym., 2007.) Priorisointiin voidaan käyttää erilaisia luokitteluja tai jopa sitä varten kehitettyjä työkaluja. Myös ad hoc -menettely on käytössä eli ”kovaäänisin voittaa”. (Karlsson ym., 2007.) Julkaisujen suunnittelu on hankala tehtävä, sillä vaatimuksilla on paljon riippuvuuksia. Siksi ei voida vain valita vaatimuksia,

joilla on korkein prioriteetti. Riippuvuuksia voidaan kuvata esimerkiksi seuraavasti: *ajallinen*, (X on tehtävä ennen Y:tä), *looginen* (X ilman Y:tä) tai *arvoon perustuva* (X:n kustannus kasvaa, jos tehdään myös Y). (Carlshamre & Regnell, 2000.)



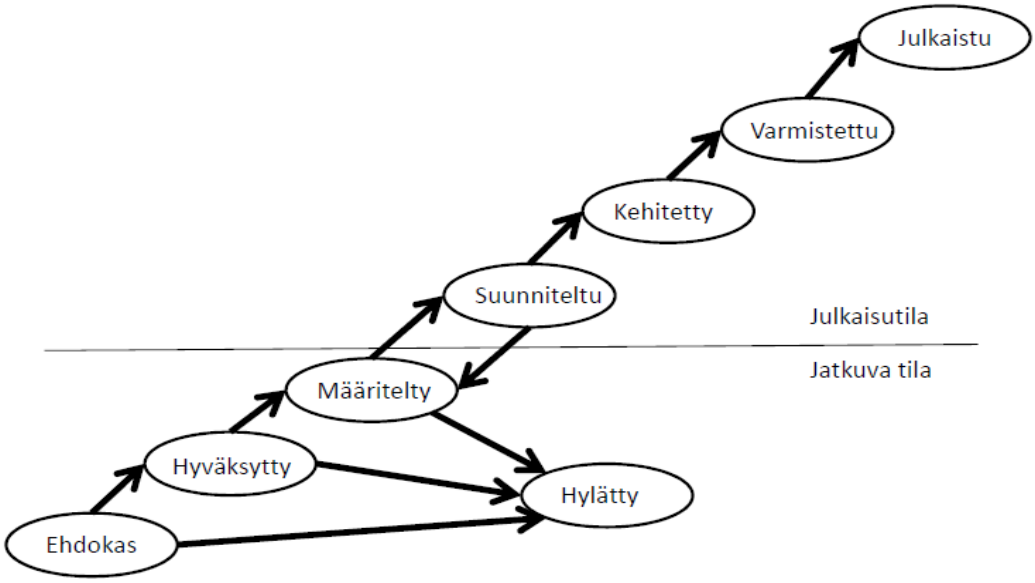
KUVIO 5 Julkaisujen suunnittelu (mukaillen van de Weerd ym., 2006)

Kuinka sitten tehdään tuottoisaa julkaisujen suunnittelua? Suunnitteluprosessin tärkein tulos on strateginen päätös siitä, mitä toimitetaan ja milloin, eli kuinka muodostetaan lista riittävän yksityiskohtaisia vaatimuksia, jotka julkaistaan markkinoille tarkoin harkittuna ajankohtana. Päätöksenteossa pitää ottaa huomioon kehitysorganisaation jäsenten osaaminen, ohjelmistoarkkitehtuuri, nykyinen asiakaskanta ja yhdistää nämä kaikki kaikenkattavaan liiketoimintastrategiaan. On siis huomioitava kehitysprosessin tyyppi, jakelukanavat, hinta- ja lisensointipolitiikka, markkinoiden tyyppi, arvo asiakkaille, tuotteen monimutkaisuus, kilpailun luonne, asiakkaiden käyttäytyminen, tuotteen joustavuus ja mukautuvuus, käyttöliittymän monimutkaisuus, myyntiennusteet, myyntikanavat jne. On selvää, että organisaation kehitysprosessin kypsyys kehittäjien osaamisen kera sekä markkinoiden kypsyys ja asiakkaiden tietämys tavoista, joilla he voivat käyttää teknologiaa omaksi hyödykseen, ovat markkinalähtöisen julkaisusuunnitteluprosessin tärkeimmät tekijät. (Regnell & Brinkkemper, 2005.)

Asiakaslähtöisen kehityksen vaatimukset kirjoitetaan yleensä yhteen dokumenttiin, *vaatimusmäärittelyyn* (engl. requirements specification). Markkinalähtöisessä kehityksessä tämä ei ole käytännöllistä, koska vaatimuksia tulee koko ajan lisää ja ne myös muuttuvat jatkuvasti. On parempi tallettaa vaatimukset varastoon, joka kehittyy dynaamisesti eri tasoilla; nykyiset ja potentiaaliset asiakasprofiilit, nykyinen ja aikaisemmat julkaisusisällöt, vaatimusehdokkaiden ja jo kehitykseen valittujen vaatimusten tilan seuranta. Vaatimusten määrää voidaan rajoittaa seulonnalla jo ennen kuin ne pääsevät varsinaiseen vaati-

musmäärittelyprosessiin. Vaatimus voidaan hylätä tekemällä pikainen arvio sen arvosta/kustannuksesta ja vaikka näin jokin hyvä vaatimus saattaa jäädä valitsematta, on karsiminen parempi vaihtoehto kuin prosessin kuormittaminen ja mahdollisesti julkaisun aikataulun siirtyminen. (Regnell & Brinkkemper, 2005.)

Vaatimuksen syntyessä on hyvin epävarmaa, toteutetaanko sitä koskaan tuotteen julkaisussa. Käytettävissä olevat resurssit ja aikataulut rajoittavat minkä tahansa toiveen päätymistä tuotteeseen. Samoin tuotteen visio ja laajuus määräävät, sopiiko vaatimus niihin tai onko se liian asiakaskohtainen. (Regnell & Brinkkemper, 2005.) Regnellin ja Brinkkemperin (2005) mukaan varastossa olevien vaatimusten seuraaminen koko kehitysprosessin läpi vaatii tilamallin (kuvio 6) käyttöä.



KUVIO 6 Vaatimusten tilamalli (Regnell & Brinkkemper, 2005, 297)

Vaatimuksia saapuu koko ajan, mutta tuotteen kehitys tehdään julkaisuissa, jotka tapahtuvat tiettyinä aikoina. Sen takia mallissa eritellään kaksi moodia tai tilaa, jatkuva tila ja julkaisutila. Jatkuvassa tilassa tuotepäällikkö kirjaa kaikilta eri sidosryhmiltä tulevat vaatimukset varastoon. Kun julkaisun tekeminen aloitetaan roadmapin mukaan, vaatimushallinnan toiminnot siirtyvät julkaisutilaan. Julkaisun sisältö eli scope kiinnitetään, jotta julkaisun kehitysprojekti pysyy hallinnassa. Muutokset sisältöön hoidetaan sen jälkeen erillisen muutoskäsittelyn kautta. Vaatimushallinnan edistymisen seuraamista varten eritellään seuraavat tilat (Regnell & Brinkkemper, 2005, 298-299):

- **Ehdokas.** Jokainen uusi vaatimus saa aluksi tilakseen 'Ehdokas'. On suositeltavaa, että vaatimuksen sanamuoto säilytetään mahdollisimman tarkoin sellaisena kuin esittäjä sen kirjoitti. Näin esittäjän sitoutuneisuus vaatimukseen säilyy.

- **Hyväksytty.** Säännöllisin väliajoin ehdokas-tilassa olevia vaatimuksia katselmoidaan ja tutkitaan, voitaisiinko ne sisällyttää seuraavaan julkaisuun. Vaatimuksen tilaksi muutetaan tällöin 'Hyväksytty'. Tämä tuotepäällikölle kuuluva tehtävä on vaikea ja vastuullinen. Siinä on otettava huomioon ensinnäkin roadmapissa tuotteelle esitetty pitkän aikavälin visio. Sitten tarvitaan vankkaa tuotteen toiminnallisuuden ja tekniikan ymmärrystä, jotta vaatimuksen tarkoitus ja vaikutukset voidaan päätellä. Lopuksi tuotepäällikön täytyy vielä luovia poliittisten ja strategisten kysymysten aallokossa ja ottaa huomioon mahdolliset uudet sopimukset, tärkeät asiakkaat ja vaativat myyjät.
- **Määritelty.** Alkuperäinen vaatimus ei useinkaan ole riittävä suunnittelun ja ohjelmoinnin tarpeisiin ja on syytä laatia yksityiskohtaisempi määrittely siihen liitettäväksi. Määrittely voi olla pelkkä selventävä teksti, mutta myös luokkakaavioita tai käyttötapauksia voidaan tehdä. Tämän jälkeen vaatimus voidaan siirtää tilaan 'Määritelty'.
- **Hylätty.** Jos vaatimus päätetään hylätä, se asetetaan tilaan 'Hylätty'. Vaatimuksen esittäjälle lähetetään tästä tieto ja perustelut. Hylättyjä vaatimuksia ei poisteta varastosta, jotta niitä voidaan tutkia vielä myöhemminkin.
- **Suunniteltu.** Tuleva julkaisupäivä ja käytettävissä olevat henkilöstöresurssit määräävät sen, kuinka monta henkilötyöpäivää on käytettävissä kehitykseen, testaukseen ja tuotteen viimeistelyyn. Tuotejulkaisun suunnittelussa yritetään priorisoinnin ja työmäärien arvioinnin avulla mahduttaa julkaisuun maksimimäärä vaatimuksia, jotka sitten asetetaan tilaan 'Suunniteltu'. Nämä vaatimukset toimivat jatkossa syötteenä suunnittelu- ja ohjelmointiprosesseille. Koska arviot pyrkivät aina olemaan liian optimistisia, osa vähemmän tärkeistä vaatimuksista saatetaan joutua poistamaan julkaisusta ajanpuutteen takia.
- **Kehitetty.** Kehittäminen sisältää teknisen suunnittelun, ohjelmoinnin, yksikkötestauksen ja liitännäismateriaalien, kuten käyttöohjeiden, esitteiden, markkinointikampanjan, ja koulutusmateriaalin, valmistamisen. Kun kaikki nämä on toiminnot on saatu valmiiksi, voidaan vaatimus asettaa tilaan 'Kehitetty'. On huomattava, että vaatimus voidaan milloin tahansa kehittämisen aikana poistaa julkaisusta. Tällöin on huolehdittava siitä, että ohjelmakoodi palautetaan tilaan, jossa se oli ennen vaatimuksen kehittämisen aloittamista. Julkaisusta poistaminen tehdään tavallisesti ajanpuutteen tai muuttuneen tärkeysjärjestyksen takia.
- **Varmistettu.** Tarvitaan useita testejä varmistamaan, että kehitetty vaatimus on riittävällä laadullisella tasolla julkaistavaksi. Tyypillisiä testejä ovat: toiminnallinen yksikkötestaus jonkin kehitysryhmän ulkopuolisen testaajan suorittamana; integraatiotestaus, joka keskittyy ohjelmamoduulien välisiin riippuvuuksiin; järjestelmätestaus koko ohjelmistojärjestelmässä; hyväksymistestaus koko tuotteelle; ja lopulta asennustiedostojen testaus.

- **Julkaistu.** Kun kaikki toimenpiteet julkaisua varten on tehty, vaatimus asetetaan tilaan 'Julkaistu' ja vaatimuksen esittäjälle lähetetään tästä tieto. Vaatimus säilytetään edelleen varastossa mahdollisia jatkotutkimuksia varten.

Useimmat kaupalliset vaatimusten hallintaan tarkoitetut työkalut sallivat omien tilojen lisäyksen ja määrittelyn, jolloin ne voidaan sovittaa yrityksen omien käytäntöjen mukaisiksi. Kehitystoimintojen sisällä tapahtuvien siirtymien, kuten suunnittelu- ja testausdokumenttien linkityksen, hallinnoimiseksi tarvitaan usein manuaalisia toimintoja. Vaatimusten kunnolliseen hallinnointiin tarvitaan aina jonkinlainen työkalu; pienissä tuotekehitysprojekteissa yksinkertainen taulukko voi olla riittävä, mutta laajemmissa hankkeissa tarvitaan välttämättä jo vaatimusten runsauden takia kunnollinen varasto. Massiiviset vaatimusmäärittelydokumentit ovat myös ongelmallisia, sillä dokumentin rakenne haittaa rinnakkaisten vaatimusten yhtäaikaista laatimista. Myös vaatimusten jäljittäminen suunnitelmiin, koodiin ja testiraportteihin asti vaatii riittävän kehittyneen työkalun käyttöä, ettei jäljittämisessä tarvitse mennä virheille alttiiseen ja työlääseen manuaaliseen työhön. (Regnell & Brinkkemper, 2005.)

3.5 Yhteenveto tuotehallinnan prosesseista

Tuotehallinnan viitekehyksessä on otettava huomioon varsinaisten prosessien lisäksi myös lukuisat sidosryhmät, joilla on sanansa sanottavana tuotteen kehittämisen suhteen. Jo sidosryhmien moninaisuus aiheuttaa sen, että esitettyjen vaatimusten määrä on mittava ja niiden käsittelyyn on kiinnitettävä erityistä huomiota.

Ylimmällä tasolla viitekehyksen prosessikaaviossa on tuotesalkku, jonka avulla pyritään saamaan kokonaisnäkemys yrityksen tuotteista ja niiden tulevasta kehityksestä. Pienessä yrityksessä tuotesalkun tasolla voidaan tehdä jopa resurssien allokointia ja kehityspanosten priorisointia. Tavallisesti kuitenkin salkun avulla tuotteet liitetään yrityksen liiketoimintasuunnitelmaan ja sen avulla tehdään tärkeitä päätöksiä yrityksen tulevaisuudesta.

Seuraavalla tasolla viitekehyksessä on roadmapping, jonka avulla liiketoimintastrategia ja yksittäiset tuotevaatimukset pyritään saamaan lähemmäksi toisiaan. Sen avulla muodostetaan eri sidosryhmille yhteinen käsitys tuotteen tulevaisuudesta ja siihen tarvittavista kehitysaskelista. Roadmapping-prosessin tuloksena syntynyt roadmap ei useinkaan ole kovin käyttökelpoinen tai ainakin sen ylläpitäminen on vaivalloista. Prosessin läpikäyminen eri sidosryhmien kanssa on kuitenkin palkitsevaa, sillä sen tuloksena saavutetaan yhteinen ymmärrys tuotteen kehitysaskelista.

Kolmannella tasolla viitekehyksessä on vaatimusten hallinta. Eri sidosryhmiltä kerätyt vaatimukset tallennetaan erityiseen varastoon, jossa tuotepäällikkö jatkuvasti seuraa niiden tilaa, poistaa duplikaatteja ja muuntaa niitä tuo-

tekehityksen kielelle tai jopa hylkää tuotteeseen sopimattomat vaatimukset. Sidosryhmät on myös pidettävä ajan tasalla vaatimusten tilasta.

Alimmalla tasolla sijaitseva julkaisujen suunnittelu käsittelee myös vaatimuksia ja menee siksi päällekkäin vaatimushallinnan kanssa. Julkaisujen suunnittelussa vaatimukset asetetaan tärkeysjärjestykseen ja niistä valitaan tulevaan julkaisuun sopiva joukko. Nämä tuotevaatimukset siirtyvät sitten kehitysprosessiin ohjelmoitaviksi ja testattavaksi. Vaatimusten hallinta ja julkaisujen suunnittelu tarvitsee aina tuekseen jonkinlaisen työkalun.

Yritysten mielestä julkaisujen työmäärien arviointi etukäteen olisi erittäin tärkeää, mutta ne eivät useinkaan käytä siihen mitään formaalia menetelmää. Vaatimusten ja niiden muutosten hallinta on hankalaa ja työlästä. Julkaisun sisältö eli scope kiinnitetään etukäteen, mutta laajuuteen saattaa tulla muutoksia työn edistyessä. Vaatimuksia tulee lisää, ne muuttuvat tai osa niistä hylätään vielä kehittämisvaiheessa, joten yrityksellä pitäisi olla käytettävissään julkaisun laajuudenhallintaan soveltuva menetelmä. Seuraavissa luvuissa esitellään räätälöidyn ohjelmistoprojektin laajuudenhallintaan kehitetty northernSCOPE™-konsepti ja tutkitaan, soveltuisiko se myös tuotekehitysprojektien hallintaan.

4 OHJELMISTOPROJEKTIN LAAJUUDENHALLINTA JA NORTHERN SCOPE™ -KONSEPTI

Ohjelmistoprojektit ovat saaneet mediassa laajaa negatiivista julkisuutta epäonnistumistensa takia. Standish Groupin (2014) raportti "Big Bang Boom" osoitti, että projektien onnistumisaste laski merkittävästi tarkastelujakson 2003-2012 aikana; vain 6 prosenttia projekteista onnistui eli ne valmistuivat ajoissa, pysyivät budjettinsa rajoissa ja niissä toimitettiin halutut ominaisuudet ja toiminnallisuus. 52 prosenttia projekteista oli "haastavia", tarkoittaen, että ne myöhästyivät, ylittivät budjettinsa ja/tai vaadittuja ominaisuuksia ja toiminnallisuutta puuttui. 42 prosenttia projekteista epäonnistui eli ne keskeytettiin ennen valmistumista tai toimitettiin, mutta tuotosta ei koskaan otettu käyttöön. Suurimpia ongelmia ohjelmistoprojekteissa ovat aikataulun pettäminen, kustannusarvion ylittyminen sekä projektista tilaajalla ja toimittajalla ollut erilainen näkemys (Forselius, 2013). Tuoreessa kirjassaan Onnistunut tietojärjestelmän hankinta Pekka Forselius (2013, 14) summaa ongelmat näin:

> On aivan liian tuttua, että projektit paisuvat hallitsemattomasti, aikataulu pettävät ja kehittäjien työpaine kasvaa kohtuuttomaksi. Lopputulos on heikkolaatuinen ja käyttäjiä huonosti palveleva. Ongelmien merkittävänä osasyynä ovat hyvän ammatillisen käytännön vastaiset työtavat. Vaatimuksien määrittely, suunnittelu ja ohjaus jäävät usein vaillinaisiksi, muutosten ja riskien hallinta unohtuu ja osapuolten vastuut on määritelty epämääräisesti. Osapuolten välinen kommunikaatio ei toimi.

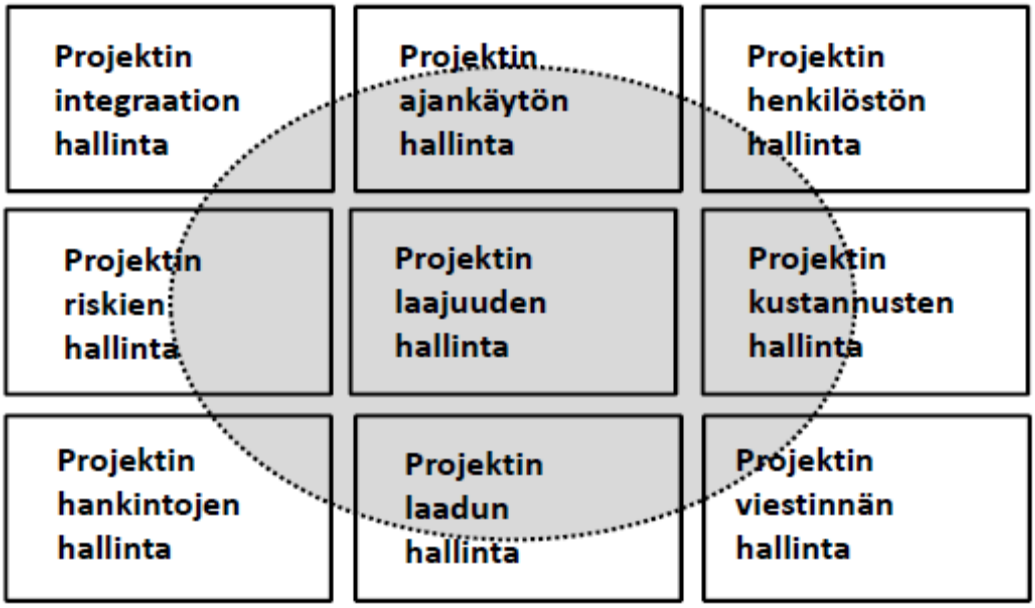
Laajuudenhallinta (engl. scope management) on kokoelma prosesseja, joita käyttämällä ohjelmistoprojektin vetäjällä on mahdollisuus mitata projektin kokoa, virtaviivaistaa vaatimusten esittämistä ja hallintaa sekä kontrolloida muutostenhallintaa päämääränään projektin pitäminen aikataulussa ja budjetissa (Dekkers & Forselius, 2007). *Ohjelmistoprojektien estimointi ja mittaaminen* (engl. software project estimation and measurement, PEM) tähtää projektin koon, tuottavuuden, työmäärän ja/tai hinnan sekä aikataulun ennustamiseen usein ja missä tahansa projektin vaiheessa. Toiminnalliseen laajuuteen perustuva estimointi eli toimintopistelaskenta on tunnustettu yhdeksi ohjelmistoprojektien hallinnan parhaista käytännöistä. (Forselius & Käkölä, 2009.) Finnish Software

Measurement Assosiation (FiSMA ry) on kehittänyt toiminnallisen laajuden hallintaan perustuvan konseptin, joka on saanut nimekseen northernSCOPE™. Konseptin avulla projektien aikataulut, kustannukset ja sovitut toiminnallisuudet on tutkitusti (Käkölä & Forselius, 2015) saatu pysymään kurissa ja tiedot projektien toteumista on kerätty laajaan kokemustietokantaan jatkokäyttöä varten. Seuraavaksi esitelläänkin ohjelmistoprojektien laajuudenhallintaa yleisesti, siihen olennaisesti liittyvää toimintopistelaskentaa ja lopuksi kerrotaan, kuinka northernSCOPE™-konseptin avulla saadaan ohjelmistoprojektit pysymään aikataulussaan sekä tilaajan että toimittajan yhteistyöllä.

4.1 Ohjelmistoprojektin laajuudenhallinta

Perinteisessä ICT-projektien hallinnoinnissa projektipäällikön edellytetään suunnittelevan, arvioivan, ennustavan, aikatauluttavan, resursoivan ja ohjaavan projekteja, jotka koostuvat hyvinkin erilaisista osista. Projektit eivät ole puhtaita ohjelmistokehitysprojekteja, vaan ”hybridejä”, sisältäen mm. koulutusta, asennusta, dokumentointia jne. Kun projektipäällikkö yrittää hallita kaikkia näitä yhtenä isona tehtäväjoukkona, ei ole tavatonta, että laajuutta on hankala hallita ja arviot ja aikataulut alkavat huomaamatta lipsua, kunnes on liian myöhäistä tehdä mitään. Aikataulun ja kustannusten kiinniottaminen myöhemmin projektin aikana ei käytännössä ole mahdollista. (Dekkers & Forselius, 2007.)

Laajuudenhallinta esitellään oppaassa Project Management Body of Knowledge (PMBOK®) (2004) yhdeksi yhdeksästä projektijohtamisen tietämysalueista. Dekkers ja Forselius (2010) nostavat artikkelissaan laajuuden hallinnan keskeiseen asemaan muihin projektijohtamisen tietämysalueisiin nähden (Kuvio 7).

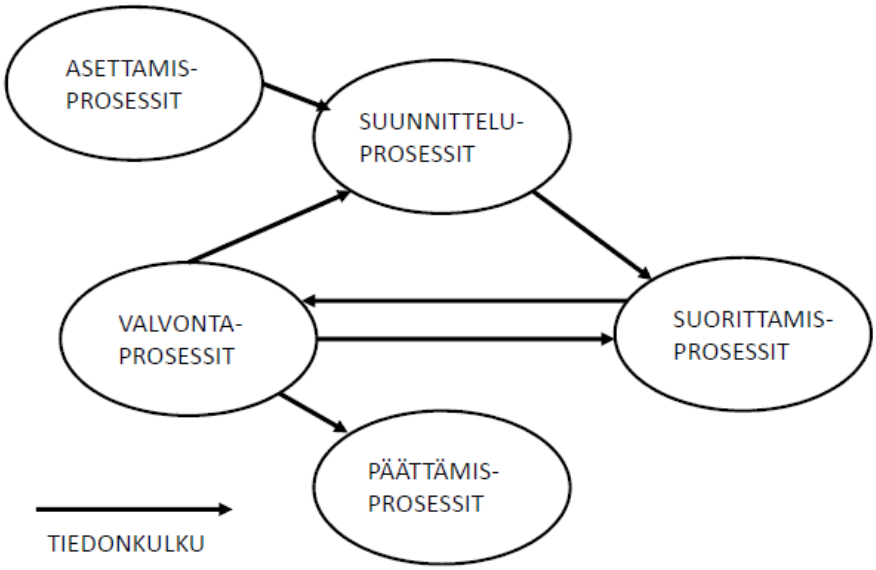


KUVIO 7 PMBOK®in projektijohtamisen tietämysalueet (Dekkers & Forselius, 2010, 16)

Tietämysalueiden suomenkieliset määritelmät on esitetty kirjassa Hankehallinnan työkalupakki (Forselius, Dekkers, Karvinen & Kosonen, 2009, 18) seuraavasti:

- **Projektin integraation hallinta** sisältää projektin johtamisen osa-alueiden välisen koordinaation edellyttämät prosessit.
- **Projektin laajuuden hallinta** muodostuu niistä prosesseista, jotka vaaditaan sen varmistamiseen, että projektissa ovat mukana kaikki ne työt, mutta *vain* kaikki ne työt, jotka projektin menestyksellinen loppuun suorittaminen edellyttää.
- **Projektin ajankäytön hallinta** muodostuu niistä prosesseista, jotka vaaditaan varmistamaan, että projekti viedään loppuun suunnitellussa ajassa.
- **Projektin kustannusten hallinta** muodostuu niistä prosesseista, jotka vaaditaan sen varmistamiseen, että projekti pystytään viemään loppuun sille hyväksytyn budjetin mukaisena.
- **Projektin laadun hallinta** muodostuu niistä prosesseista, jotka vaaditaan sen varmistamiseksi, että projekti täysin tyydyttää ne tarpeet, joiden takia se asetettiin.
- **Projektin henkilöstön hallinta** muodostuu niistä prosesseista, jotka vaaditaan sen varmistamiseen, että osallistuvia henkilöstövoimavaroja käytetään mahdollisimman tehokkaasti.
- **Projektin viestinnän hallinta** muodostuu niistä prosesseista, jotka vaaditaan sen varmistamiseen, että projekti-informaatio tuotetaan, kerätään, levitetään, varastoidaan ja hävitetään oikea-aikaisesti ja tarkoituksenmukaisesti.
- **Projektin riskien hallinta** on järjestelmällinen riskien tunnistamisprosessien, analysointiprosessien sekä niihin varautumis- ja reagointiprosessien muodostama kokonaisuus, jonka tarkoituksena on varmistaa asetettujen tavoitteiden saavuttaminen optimoimalla riskien ja niiden hallintatoimenpiteiden välinen tasapaino.
- **Projektin hankintojen hallinta** on tavoitteeseen pääsyn edellyttämien tuotteiden ja palveluiden ostamisprosessien muodostama kokonaisuus.

Projektijohtaminen edellyttää tietämisen lisäksi toimenpiteitä (Forselius ym. 2009). PMBOK®-kirjassa tarvittavat prosessit jaetaan viiteen eri ryhmään (kuvio 8). Prosessiryhmät kuvaavat projektin etenemistä asettamisesta suunnittelun, suorittamisen ja valvonnan kautta päättämiseen. Tieto kulkee aina edellisestä prosessiryhmästä seuraavaan. Joskus on valvonnan yhteydessä tehtyjen havaintojen perusteella syytä palata uudelleen suunnitteluun ja jatkaa sitten suorittamista ja valvontaa, kunnes päättämisen edellytykset saavutetaan. (Forselius ym. 2009.)



KUVIO 8 Projektijohtamisen prosessiryhmät (PMBOK® ja Forselius ym., 2009, 19)

Laajuudenhallinta on kuvion 7 keskiössä, vaikkakin PMBOK® esittää tietämysalueet alun perin toisessa järjestyksessä. Järjestys sinänsä ei ole olennainen, mutta on hyvä korostaa laajuudenhallinnan keskeistä roolia ohjelmistokehityksessä. Laajuudenhallinnalla on vahvat yhteydet muihin tietämysalueisiin, erityisesti kuitenkin ajan, kustannusten, laadun ja riskien hallintaan. Laajuutta ei voi muuttaa ilman, että siitä aiheutuu seuraamuksia projektin aikatauluun, budjettiin, laatuun ja riskitasoon. Tämä pätee myös toisinpäin; jos aikataulua tai budjettia täytyy tiukentaa, saatetaan tarvitaan muutoksia myös laajuuteen ja laatuvaatimuksiin tai projektin riskit kasvavat. (Dekkers & Forselius, 2007.)

Ei voida liikaa korostaa sitä, että täydelliset, täsmälliset ja dokumentoidut vaatimukset ovat laajuudenhallinnan ytimessä. Vaatimuksia ovat toiminnalliset käyttäjän vaatimukset (mitä ohjelmisto tekee), ei-toiminnalliset vaatimukset (kuinka ohjelmisto vastaa laatu- ja suorituskykyvaatimuksiin) ja tekniset vaatimukset (fyysiset toteutusvaatimukset kuten tiimin taidot, työkalut, laitteisto-alusta jne.). (Dekkers & Forselius, 2007.) Suurin osa ohjelmistojen piilevistä virheistä voidaan johtaa vaatimusten tasolle. Laajuudenhallinta ei takaa täydellisiä vaatimuksia, mutta ainakin laajuuden määritteleminen rajaa sen, mitä vaatimuksiin sisältyy ja mitä ei. (Dekkers & Forselius, 2010.) Projektin laajuudenhallinta sisältää tarpeelliset prosessit sen varmistamiseksi, että projekti sisältää vain ja ainoastaan sen työn, joka vaaditaan projektin päättämiseen onnistuneesti. Tähän työhön kuuluvat seuraavat prosessit (PMBOK®, 2004, 103):

- Kerää vaatimukset – määrittele ja kirjaa sidosryhmien tarpeet
- Määrittele laajuus - laadi yksityiskohtainen kuvaus projektista
- Jaa projekti osiin – pilko projekti helpommin hallittaviin palasiin
- Varmenna laajuus – hyväksy projektin tuotokset

- Tarkista laajuus – seuraa projektin tilaa ja hallinnoi muutoksia

Mainituissa prosesseissa on useita toimintoja, jotka eivät ole mukana perinteisissä ICT-projekteissa: Näitä toimintoja ovat (Dekkers & Forselius, 2007,143-144):

- ICT-hankkeen vaatimusten analysointi ja jakaminen itsenäisesti hallinnoitaviin projekteihin (tai aliprojekteihin).
- Toiminnallisen koon mittaaminen, jolla arvioidaan ohjelmiston toiminnallinen laajuus.
- Tilanneanalyysi, jolla selvitetään laadulliset ja tekniset vaatimukset.
- Ohjelmakoodin uudelleenkäytön vaikutuksen arviointi.
- Projektin lähtötilanteen laajuuden arviointi perustuen karkean tason vaatimuksiin.
- Lähtötason asettaminen projektin mittareille lopullisten vaatimusten perusteella. Mittareita ovat toiminnallinen koko (projekteissa, joissa se on laskettavissa), kustannukset ja työmäärä. Asettamisessa käytetään jotakin kokemusperäisistä tietokannoista, esim. Experience® tai ISBSG (International Software Benchmarking Standards Group).
- Projektin työmäärän, keston ja kustannusten estimointi perustuen aikaisempiin projektitoteumiin tai kokemustietokantoihin.
- Projektin aikana syntyvien muutostarpeiden estimointi, hyväksyntä, jäljittäminen ja liittäminen olemassa olevaan projektidokumentaatioon.

Laajuudenhallinta on paras antaa itsenäisen ja asiantuntevan S*cope Managerin* (käytetty myös nimitystä mitoitusvastaava (Forselius, 2013)) tehtäväksi. Hänet on koulutettu ICT-projektien hallintaan, asiakassuhteiden ylläpitoon, ohjelmistojen estimointiin, vaatimusten esittämiseen ja dokumentointiin, toiminnallisen koon laskentaan, ohjelmistojen mittaamiseen, muutosten hallintaan ja projektin pilkkomiseen osaprojekteiksi. Scope Manager voi siis toimia myös osaavana projektipäällikön apulaisena. Lisäksi riippumattoman Scope Managerin käyttö ohjaamassa ja suorittamassa monia laajuudenhallinnan prosesseja helpottaa raskaasti työllistettyä projektitiimiä. Scope Manager palvelee riippumattomana asiakkaan edunvalvojana ja projektitiimiin kuuluvana tarkastajana vastuullaan seurata ja arvioida projektin edistymistä. Kun projektin laajuus on koko ajan tarkkailun alla, asiakas ja toimittaja voivat paremmin määritellä, rakentaa ja hankkia laadukkaita ohjelmistoja. (Forselius, 2013.) ECQA (European Certification & Qualification Association) on myöntänyt Certifed Scope Manager – sertifikaatteja vuodesta 2007 lähtien. Suomessa northernSCOPE™- konseptin mukaista Certifed Scope Manager -koulutusta tarjoaa TIVIA ry (Tieto- ja viestintätekniikan ammattilaiset ry).

4.2 Toiminnallisen laajuuden mittaamiseen perustuva ohjelmistoprojektin työmäärän ja keston arviointi

Finnish Software Measurement Assosiation (FiSMA) –yhdistyksen työryhmä on kehittänyt kaikentyyppisille ohjelmistoille soveltuvan yleisen ja parametrisoidun toiminnallisen laajuuden mittausmenetelmän nimeltään FiSMA 1.1. Menetelmä on standardoitu numerolla SFS-ISO/IEC 29881. Standardin johdannossa sanotaan:

> Toiminnallinen laajuus on keskeinen mittari, kun vertaillaan ohjelmistokehityksen työkokonaisuuksia ja vaihtoehtoisia kehittämistapoja. Toiminnallista laajuutta on hyödynnetty etenkin työmäärien arvioinnissa ja tuottavuuden analysoinnissa, mutta se on osoittautunut käyttökelpoiseksi myös projektien suunnittelussa, seurannassa, valvonnassa ja sopimusten solmimisessa. Toiminnallisen laajuuden mittaaminen (FSM) on parhaimmillaan, kun kaikki käyttäjän haluamat mittaushetkellä tunnistetut toiminnallisuudet ja toiminnot on nimetty ja lueteltu. Tällöin laajuuden ja muutosten hallinta on tehokasta, luotettavaa ja verrattain helposti ymmärrettävää jopa loppukäyttäjän näkökulmasta.

Ohjelmiston laajuus on yksi tärkeimmistä ohjelmistoprojektin hintaan vaikuttavista tekijöistä. Onhan selvää, että suuren ohjelmiston kehittäminen vaatii enemmän työtä ja rahaa kuin pienen ohjelmiston. Kuitenkin alalla käytetään laajuuden mittaamiseen tarkoitettuja toimintopistelaskentamenetelmiä harvoin, vaikka ne ovat ainoita ISO-standardoituja menetelmiä, joilla ohjelmiston laajuus voidaan kiistattomasti ja toistettavasti todentaa. Käyttämättömyyteen on monia syitä; asiakkaat ja toimittajaorganisaatioiden johtajat eivät tiedä näiden menetelmien olemassaolosta, pitävät niitä hankalina ja aikaa vievinä käyttää tai luulevat niiden tuottavan eri tuloksia riippuen mittaajasta. (Forselius & Savioja, 2013.) Ohjelmistojen kehittäminen on riskialtis, monimutkainen ja kallis prosessi. Monimutkaisuus johtaa siihen, että kehittämiseen tarvittavaa työmäärää ja kustannuksia on hankala arvioida ennakolta. Ohjelmistoprojektien *arviointi* (engl. estimation) ei ole kuitenkaan salatiedettä. Käytettävissä on hyviä arviointitekniikoita, projektien toteutumatietojen varastoja ja luotettavia yhtälöitä arvioiden tekemiseen. (Hill, 2011.)

Ohjelmistoprojekti sisältää kolmenlaisia vaatimuksia; toiminnallisia, laadullisia ja teknisiä (Hill, 2011, 3):

- Toiminnalliset vaatimukset kertovat, MITÄ ohjelmisto tekee. Nämä ovat prosesseja, joita kehitettävän ohjelmiston on tarkoitus suorittaa tai tukea.
- Laadulliset eli ei-toiminnalliset vaatimukset kertovat, MITEN ohjelmiston täytyy toimia. Näitä ovat mm. luotettavuus, turvallisuus, tehokkuus, käytettävyys, ylläpidettävyys, siirrettävyys jne.
- Tekniset vaatimukset kertovat, KUINKA ohjelmisto on rakennettava. Näihin sisältyvät mm. työkalut, menetelmät, arkkitehtuuri, tiimin kokoonpano, laitteistot, tietokannat jne.

Toiminnallisen laajuuden mittaamiseksi on olemassa useita standardeja. Mittayksikkönä käytetään *toimintopistettä* (engl. function point, fp), joka perustuu järjestelmän käsittelemien tietojen sekä järjestelmän toimintojen määrään. Vanhin toimintopistemenetelmä kehitettiin jo 1970-luvun lopussa IBM:llä auttamaan kehittäjiä työmäärien arvioinnissa. Sama menetelmä on edelleen käytössä ja se tunnetaan nimellä IFBUG (International Function Point User Group). Menetelmästä on kehitetty vuosien kuluessa useita variaatioita, mutta mikään niistä ei ole koskaan saavuttanut IT-alan varauksetonta suosiota. Syynä lienee se, että nämä menetelmät ovat prosessilähtöisiä ja tarkastelevat toiminnallisuutta puhtaasti ohjelmistokehittäjän näkökulmasta. Suomessa kehitetty FiSMA 1.1-menetelmä on toimintolähtöinen ja siinä mitataan ohjelmiston käyttäjälle tuottamia palveluja ja käytetään tietojärjestelmän omistajan ja käyttäjän ymmärtämiä käsitteitä ja termejä. (Forselius, 2013.)

FiSMA 1.1 -menetelmässä käyttäjien ja kehitetyn tai kehitettävän ohjelmiston välinen yhteys rakentuu käyttäjien tarpeiden perusteella määriteltyjen toiminnallisten vaatimusten ja niistä johdettujen ohjelmiston toimintojen kautta. Mittaamisen edellytyksenä on siis, että kaikki mitattavan ohjelmiston käyttäjälle tuottamat eri palvelut tunnistetaan, nimetään ja luetteloidaan.. FiSMA 1.1-menetelmällä voidaan saada luotettavia arvioita ohjelmiston toiminnallisesta laajuudesta jo hyvin aikaisessa kehittämisvaiheessa. (Forselius, 2013.)

Toiminnallisen laajuuden mittaamiseen riittävä vaatimusten kuvaustaso on esimerkiksi seuraava (Forselius, 2013, 52):

- Käyttötilannekuvaukset (käyttöliittymä, toiminnallisuus) ulkoisine liittymineen
- Käsitemalli (luokkakaavio)
- Liiketoimintaprosessien kuvaukset
- Tekninen arkkitehtuuri (järjestelmän rakenne)

FiSMA 1.1 -menetelmässä näiden tietojen avulla laaditaan luettelo tulevan ohjelmiston erilaisista toiminnoista. Menetelmä jakaa ohjelmiston toiminnot seitsemään toimintoluokkaan ja luokat edelleen 28 toimintotyyppiin (liite 3). Ohjelmistohankkeen alkuvaiheessa toimintoluettelon laadinnan pohjaksi riittää melkeinpä toimintoluokkien taso ja hankkeen edetessä ja vaatimusmäärittelyn valmistuessa luetteloa voidaan tarkentaa (Forselius, 2013). Toimintoluetteloon listataan seuraavanlaiset erityyppiset toiminnot (Forselius, 2013, 41):

- Hakunäytöt, joilla näytetään järjestelmän tietoja
- Päivitysnäytöt, joilla päivitetään järjestelmän tietoja
- Tulosteet eli raportit ja lomakkeet
- Vastaanotettavat sanomat ja tiedostot
- Lähetettävät sanomat ja tiedostot
- Algoritmiset palvelut koskien tietoturvaa, laskentaa jne.
- Käsitteet eli tietokannan luokat

Valmisteluvaiheessa toiminnallisen laajuuden arvioimiseksi riittää, että luetteloon listattujen toimintojen kokona käytetään FiSMA 1.1:n tarjoamia oletustoimintopisteitä (keskimääräinen koko). Esimerkiksi kolmetoimisen (lisäys/muutos/poisto) päivitysnäytön oletuskoko on 16,8 toimintopistettä. Kun myöhemmin vaatimusten tarkentuessa näytöstä tiedetään yksityiskohtaisempia tietoja, voidaan sen koko toimintopisteinä S laskea kaavasta

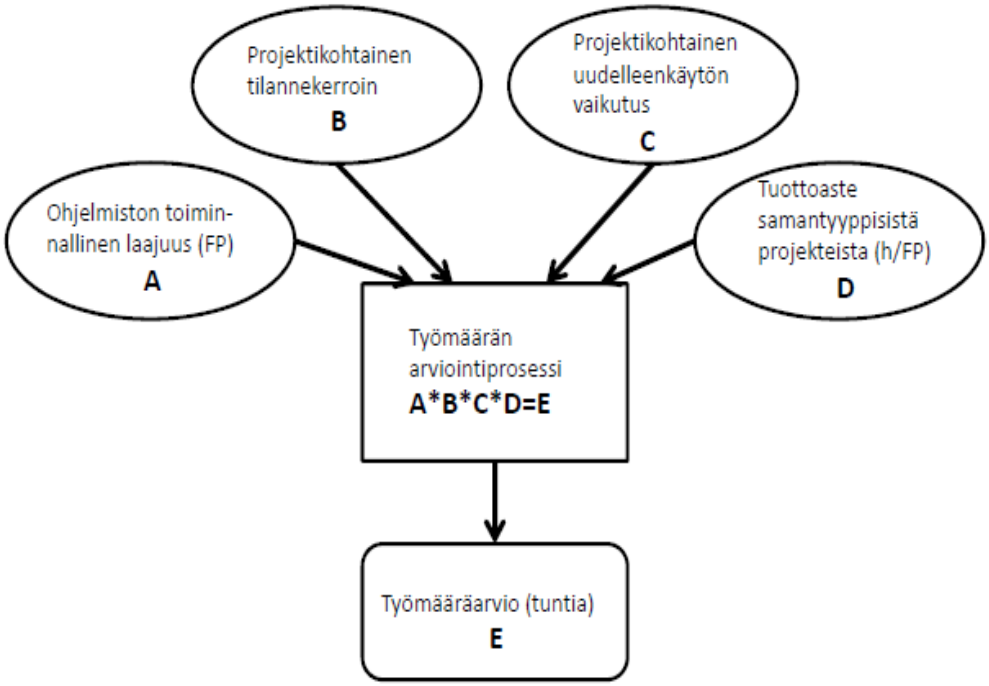
$$S = m(0,2 + n/5 + w/1,5 + r/2),$$

jossa m = päivitystoimintojen lukumäärä, 0,2 on perustamisvakio, n = dataelementtien lukumäärä, w = kirjoitusviittausten lukumäärä ja r = lukuviittausten lukumäärä. Tällöin esimerkiksi kolmetoimisen päivitysnäytön, jossa on 9 syöttökenttää, yksi valintalista ja kolme painiketta, kooksi saadaan 3(0,2 + 15/5 + 1/1,5 + 2/2) = 14,4 toimintopistettä. Kaikille 28 toimintotyypille on oma toimintopisteiden laskentakaavansa, jotka löytyvät standardista SFS-ISO/IEC 29881 tai esimerkkeineen FiSMAn sivuilta. (FiSMA 1.1, 2008.)

Toimintopistelaskennan systemaattisen ja laajemman käytön esteenä pidetään usein laskentaan kuluvaa aikaa. Menetelmän soveltaminen koetaan hitaana ja siten liian kalliina käyttää. Yksi maailman tunnetuimmista toimintopistelaskennan asiantuntijoista, Capers Jones, halusi osoittaa, ettei ajankäyttö ole välttämättä suurin este toimintopistelaskennan käytölle. Hän laski omalla menetelmällään 40 sovelluksen toimintopisteet 75 minuutissa eli keskimääräinen laskentanopeus oli vähemmän kuin 2 minuuttia sovellusta kohden. (Forselius & Savioja, 2013 ja Jones, 2013.) Jonesin innoittamana FiSMAn piirissä tehtiin kesällä 2013 nopeuskoe, johon kutsuttiin mukaan vapaaehtoisia Scope Managereita ja joille annettiin tehtäväksi arvioida viisi erilaista ohjelmistoa. Pohjatiedoksi annettiin laskettavien ohjelmistojen erityyppisten toimintojen lukumäärät ja tehtävänantona oli laskea ohjelmiston koko FiSMA 1.1 -toimintopisteinä. Jatkotehtävänä oli annettujen lisätietojen avulla myös arvioida ohjelmiston tekemiseen tarvittava työmäärä. Arviointiin käytetty aika mitattiin ja nopeuden lisäksi tutkittiin myös arvion tarkkuutta verrattuna järjestäjien laskemiin benchmarkarvoihin. Keskimääräiseksi arviointinopeudeksi saatiin 2,37 minuuttia ohjelmistoa kohden, mikä vastaa hämmästyttävän hyvin Jonesin saamaa tulosta. Ottaen huomioon sen, että laajojen ohjelmistojen tekemiseen menee tuhansia ja tuhansia tunteja, ei ole perusteltua säästää arviointikustannuksissa, jotka tehdyn kokeen mukaan eivät muodosta kovinkaan merkittävää osaa kokonaiskustannuksista. (Forselius & Savioja, 2013.)

Kun ohjelmiston koko on arvioitu toimintopisteinä, on mahdollista laskea sen toteuttamiseen kuluva aika eli *työmäärä (*engl. effort) perustuen johonkin kokemustietokantaan. FiSMA on kerännyt ylläpitämäänsä Experience®-tietokantaan vuosien ajan Suomessa toteutuneiden ohjelmistoprojektien mittaustietoja. Maailmalla laajimmin käytetty kokemustietokanta on ISBSG. Yritykset voivat myös kerätä omaa kokemustietokantaansa toteutuneista projekteista ja niiden avulla saadaankin kaikkein tarkin arvio. (Forselius, 2013.) Ohjelmistoprojektin työmäärän arviointiprosessi on esitetty kuviossa 9.

Ohjelmiston tekemiseen tarvittava työmäärän arvioinnissa on otettava huomioon myös projektikohtainen tilannekerroin ja projektikohtainen uudelleenkäytön vaikutus.



KUVIO 9 Ohjelmistoprojektin työmäärän arviointiprosessi (Forselius, 2013, 51)

Tilannekertoimen laskemiseksi FiSMA on kehittänyt ND21-tilanneanalyysimenetelmän, jossa 21 tuottavuustekijän avulla arvioidaan projektin olosuhteiden vaikutus työmäärään. Tuottavuustekijät jakaantuvat neljään ryhmään: projekti-, prosessi- tuote- ja henkilöstötekijöihin. Projektitekijöissä arvioidaan mm. asiakkaan sitoutuneisuutta, sidosryhmien määrää ja aikataulupaineita. Prosessitekijöitä ovat mm. standardien, metodien ja työkalujen vaikutus sekä kehitysprosessin kypsyys. Tuotetekijöissä arvioidaan toiminnallisten ja laatuvaatimusten tasoa ja henkilöstötekijöissä mm. tiimin ryhmätyötaitoja, työkaluosaamista ja projektijohdon kokeneisuutta. Kukin tekijä arvioidaan asteikolla --, -, +/-, + ja ++, joista –- ja – vaikuttavat työmäärää nostavasti ja + ja ++ sitä vähentävästi. ND21- menetelmän tarkat arviointikriteerit löytyvät FiSMAn sivuilta www.fisma.fi ja tilannekertoimen laskentakaavio liitteestä 4. (FiSMA, 2001.)

Uudelleenkäytön on sanottu olevan tehokkain tapa parantaa ohjelmistotuotteollisuuden tuottavuutta. Useat organisaatiot ovat ilmoittaneet tavoitteekseen nostaa uudelleenkäytön astetta. Uudelleenkäytön mittaamiseksi ei ole kuitenkaan ollut olemassa yleisessä käytössä olevia hyviä menetelmiä ja siksi FiSMAn työryhmä on kehittänyt tarkoitukseen soveltuvan menetelmän FiSMA RRM (Reuse Measurement Method) versio 2002:n. Menetelmä perustuu toiminnalli-

sen laajuuden mittaamiseen ja yksinkertaiseen malliin ohjelmistoprojektin suoritteista. Kolmas RRM:n peruskäsite on sisäinen tai ulkoinen uudelleenkäyttö, joita molempia voi esiintyä samassa projektissa. Jos uudelleenkäyttö ei ylitä projektin rajoja, sitä kutsutaan sisäiseksi. Se on projektin kannalta yhtä käyttökelpoista kuin ulkoinenkin uudelleenkäyttö, mutta jos organisaatio haluaa nostaa uudelleenkäyttöastettaan, on sen syytä keskittyä enemmän ulkoiseen. Tarkoituksena on siis käyttää toisaalla kehitettyjä uudelleenkäytettäviä komponentteja ennen projektin mittaamista. Ohjelmistoprojektin suoritteiden mallissa ovat ne konkreettiset suoritteet, jotka ovat suoraan kytköksissä jokaiseen toiminnalliseen vaatimukseen. Näitä suoritteita ovat ohjelmakoodi, testitapaukset sekä ohjelmisto- ja käyttäjädokumentaatio. Jokainen suorite edustaa tiettyä prosenttiosuutta työmäärästä, joka tarvitaan toiminnon käyttöönottoon. Näiden nk. painokertoimien summan täytyy olla 100 ja jokainen organisaatio kalibroi kertoimensa perustuen yleiseen ja mitattuun tietoon kyseisen suoritteen erityisominaisuuksista. Uudelleenkäytön vaikutus ohjelmistoprojektin toteutuksen työmäärään voi olla joko lisäävä tai vähentävä. Jos uuden ohjelmiston toimintojen ja suoritteiden halutaan olevan uudelleenkäytettäviä, tarvittava työmäärä on suurempi. Jos taas on käytettävissä hyviä uudelleenkäytettäviä komponentteja, työmäärä pienenee. FiSMA RMM:n tuloksena saadaan uudelleenkäyttökerroin R. Sen arvo on tasan 1, jos uudelleenkäytöllä ei ole vaikutusta tai jos vastakkaiset vaikutukset kumoavat toisensa. Jos uudelleenkäytettävistä komponenteista on enemmän hyötyä kuin uudelleenkäytettävyysvaatimusten aiheuttamasta kuormasta, R:n arvo on 0:n ja 1:n välissä. Jos taas uudelleenkäytettävyysvaatimusten vaikutus on suurempi, kertoimen arvo sijoittuu 1:n ja 2:n väliin. Ennen uudelleenkäytön vaikutuksen mittaamista on rakennettavan ohjelmiston toiminnallinen laajuus mitattava ja täydellinen lista erityyppisistä toiminnoista on oltava olemassa. Jokaisen toiminnon kompleksisuus ja koko täytyy siis olla tiedossa. Jos parempaa arviota ei saatavissa, suoritetyyppien painokertoimina voidaan käyttää näitä: ohjelmistokoodi 40%, testitapaukset 30%, ohjelmistodokumentaatio 20% ja käyttäjädokumentaatio 10%. Jos minkä tahansa toiminnon jollekin suoritteelle on olemassa uudelleenkäytettävä komponentti, vaikutus työmäärään saadaan kertomalla suoritteen painokerroin toiminnon koolla. Etumerkki on miinus, koska tässä tapauksessa vaikutus kokonaistyömäärään on vähentävä. Jos jonkin toiminnon suorite täytyy tehdä uudelleenkäytettäväksi, vaikutus työmäärään saadaan samalla laskutoimituksella, mutta etumerkki on plus, koska työmäärä kasvaa. Lopullisen uudelleenkäytön vaikutuksen mittaaminen tapahtuu seuraavasti:

1. Käydään läpi kaikki laajuusmittauksessa löydetyt toiminnot ja poimitaan niistä ne, joilla tunnistetaan olevan vaikutusta uudelleenkäyttöön.
2. Lasketaan jokaisen tällaisen toiminnon uudelleenkäytön vaikutus r_i edellä kuvatulla tavalla.
3. Lasketaan uudelleenkäytön kertoimen R arvo kaavasta $R = (S + ?r_i)/S$, jossa S on ohjelmiston toiminnallinen laajuus ja ? joko plus tai miinus. (FiSMA, 2002.)

Projektin kokonaistyömäärän arvioimiseksi on vielä selvitettävä tuottoaste (tuntia/toimintopiste) samantyyppisistä projekteista. Käytettävissä olevasta kokemustietokannasta etsitään mahdollisimman samantyyppisiä aikaisemmin valmistuneita projekteja. Esimerkiksi Experince®-kokemuskantaa käytettäessä valinnassa on mahdollista käyttää seuraavia luokittelukriteerejä (4SUM Partners Ltd, 2014):

- Kehittämistyön tyyppi (ohjelmistoprojektin tai ylläpitotyön tyyppi)
- Liiketoiminta-alue (asiakkaan liiketoiminta-alue tai teollisuusala)
- Alustatyyppi (kehitettävän ohjelmiston alusta)
- Ensisijainen kehitystyökalu (esim. käytettävä ohjelmointikieli)

Kuviossa 9 esitetyn ohjelmistoprojektin työmäärän arviointi voidaan nyt suorittaa kertomalla keskenään projektin toimintopisteet, tilannekerroin, uudelleenkäytön kerroin ja kokemustietokannasta saatu tuottoaste:

> *työmäärä (h) = toimintopisteet (fp) * tilannekerroin * uudelleenkäytön kerroin(R) * tuottoaste (h/fp).*

Tyypillisesti tilannekerroin vaihtelee välillä 0,5-2,5 ja uudelleenkäytön kerroin välillä 0,7-1,5 (Forselius, 2013). Jos arvioidaan esimerkiksi projektia, jonka laskennallinen koko toimintopisteinä on 500 fp ja kokemusperäinen tuottoaste 5 h/fp, arvioiduksi työmääräksi saadaan suoraan 2500h ilman kerrointen vaikutusta. Kertoimet huomioonottamalla työmääräarvioksi saadaan pienimmillään 875h ja suurimmillaan 9375h. Työpäiviksi muutettuna työmääräarvio vaihtelee näin välillä 117 -1250 tpv. Tilannekertoimen ja uudelleenkäytön kertoimen vaikutusta ei siis ole syytä väheksyä. Lisäksi arvioinnin alkuvaiheessa on aina otettava huomioon projektin *laajuuden kasvaminen* (engl. scope creep), jonka vaikutus on tyypillisesti 20-30 prosenttia (Hill, 2011).

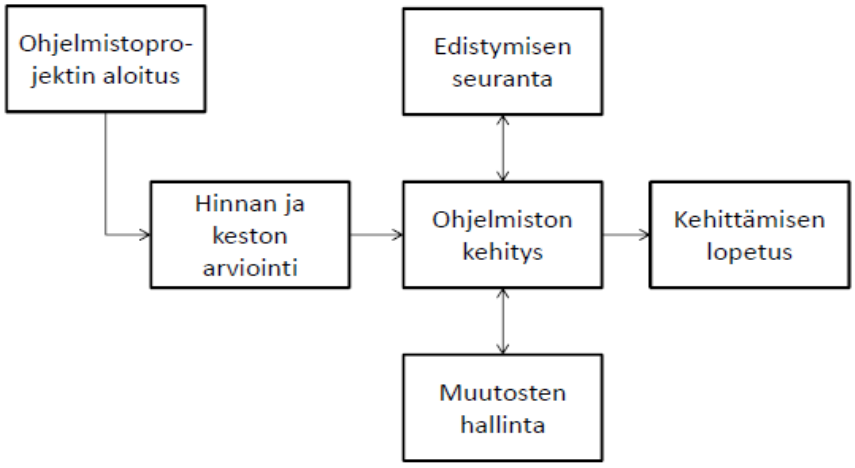
Työmäärien arviointia helpottamaan on olemassa erilaisia työkaluja. NortherScope™-konseptia tukeva työkalu Experience® Service on hankittavissa osoitteesta http://www.4sumpartners.com. Tällä työkalulla voidaan tehdä ensin karkean määrittelyn perusteella projektin alustavan työmäärän laskenta, myöhemmin toimintojen tasolle tarkentuva laskenta, projektin valmiusasteen seuranta ja lopuksi toteumatiedon tallennus kokemustietokantaan. Experience® Service tukee FiSMA 1.1 –menetelmää toimintopisteiden laskennassa, FiSMA RMM -menetelmää uudelleenkäytön arvioinnissa ja FiSMA ND21 –menetelmää tilanneanalyysiä tehtäessä. (4SUM Partners Ltd., 2014.)

4.3 northernSCOPE™-konsepti

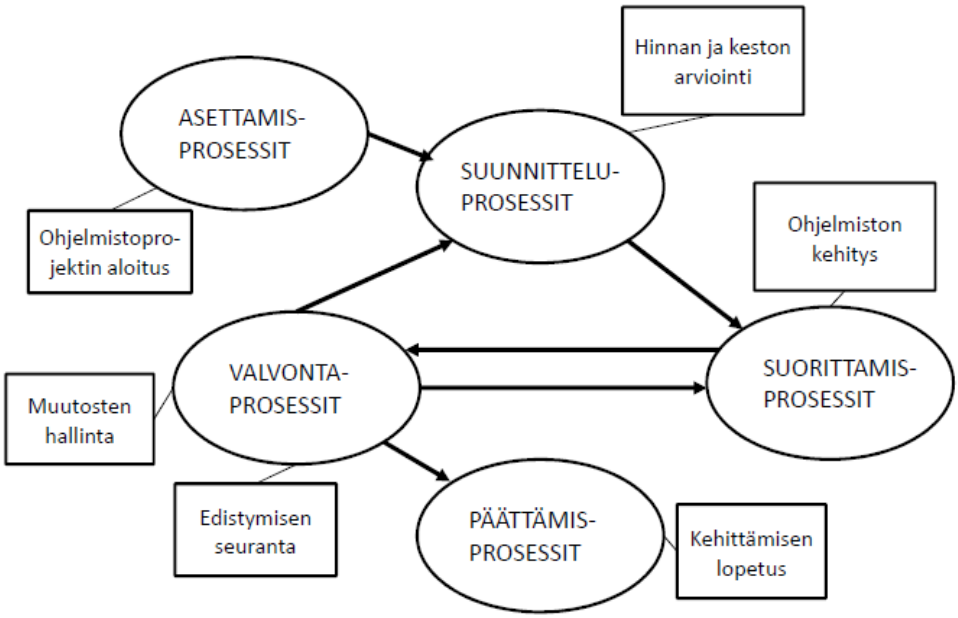
Tässä luvussa esitellään Finnish Software Measurement Assosiationin (FiSMA ry) kehittämä ohjelmistoprojektien hallinnan parhaisiin käytäntöihin perustuva laajuudenhallintakonsepti, joka on saanut nimekseen northernSCOPE™. Kon-

septin avulla projektien aikataulut, kustannukset ja sovitut toiminnallisuudet on tutkitusti (Käkölä & Forselius, 2015) saatu pysymään kurissa ja tiedot projektien toteumista on kerätty laajaan kokemustietokantaan jatkokäyttöä varten.

Samalla tavoin kuin PMBOK®:ssa myös northernScope™-konseptissa voidaan erottaa viisi eri laajuudenhallinnan prosessia (Kuvio 10). Ohjelmistokehitysprojektin ydinprosessi on luonnollisesti ohjelmiston kehitys. Se ei ole itsessään hallinnointiprosessi vaan kohde, jonka prosessia on välttämättä hallinnoitava. Viisi laajuudenhallinnan prosessia ovat ohjelmistoprojektin aloitus, hinnan ja keston arviointi, muutosten hallinta, edistymisen seuranta ja kehittämisen lopetus. Kuviossa 11 esitetään näiden prosessien liityntä PMBOK®:in prosessiryhmiin.



KUVIO 10 Laajuudenhallintalähtöisen projektin arviointi- ja mittausprosessit (Forselius & Käkölä, 2009)



KUVIO 11 PMBOKin prosessiryhmien ja laajuudenhallinnan prosessien vastaavuus (Forselius & Käkölä, 2009)

Varsinainen NorthernSCOPE™-konsepti koostuu 12 askeleesta tai portaasta, jotka on koottu taulukkoon 2.

TAULUKKO 2 northernSCOPE™-konseptin 12 askelta (Dekkers & Forselius, 2007, 17)

Askel	Toiminto
1	Asiakas laatii karkean tason vaatimukset ja Scope Manager (SM) kiinnitetään
2	Projekti jaetaan pienempiin aliprojekteihin
3	SM laskee alustavat toimintopisteet jokaiselle aliprojektille ja laatii arvion koko projektin koosta
4	Asiakas ja SM määrittävät projektin laatuvaatimukset
5	Asiakas laatii projektista tarjouspyynnön
6	Toimittaja valitaan toimintopistekohtaisen yksikköhinnan perusteella
7	Vaatimukset kiinnitetään ja dokumentoidaan
8	SM laskee lähtötilanteen toimintopisteet (baseline)
9	SM laskee haluttujen muutosten vaikutukset kustannuksiin
10	SM valvoo projektin edistymistä
11	Projekti päättyy ja asiakas maksaa toteutuneiden toimintopisteiden mukaan
12	SM kerää ja tallettaa kokemusdatan

Seuraavaksi tarkastellaan kutakin askelta yksityiskohtaisemmin:

Askel 1: Asiakas laatii karkean tason vaatimukset ja Scope Manager (SM) kiinnitetään

NorthernSCOPE™-konseptin käyttö alkaa, kun tietojärjestelmähankintaa suunnitteleva asiakas tunnistaa tarpeen Scope Managerin käytölle ja kiinnittää sellaisen projektiin. Ensimmäisenä tehtävänä on kutsua koolle tapaaminen projektin ohjausryhmän kanssa. Tällöin voidaan sopia ja selventää Scope Managerin, asiakkaan ja toimittajan roolit ja vastuut projektissa. Scope Manager katselmoi asiakkaan valmiina olevien vaatimusmäärittelyjen selkeyden ja valmiustason ja ottaa huomioon asiakkaan tavoitteet ja toiveet vaatimusten suhteen. (Dekkers & Forselius, 2010.)

Askel 2: Projekti jaetaan pienempiin aliprojekteihin

Jo varhaisessa vaiheessa laaja ohjelmistohanke on syytä jakaa pienempiin ja paremmin hallittaviin aliprojekteihin. Jakamisen tekevät asiakas ja Scope Manager yhteistyössä perustuen karkean tason vaatimusmäärittelyyn ja ennalta määrättyihin kriteereihin. (Dekkers & Forselius, 2010.) Hankehallinnan työkalupakissa (Forselius, Dekkers, Karvinen, & Kosonen, 2009, 22) esitetään kriteerit, joiden perusteella tietojärjestelmähanke olisi syytä jakaa aliprojekteihin. Kriteerejä ovat mm.

- Jos hankkeeseen liittyy organisaation muuta kehittämistä ohjelmistoprojektin lisäksi, on nämä erityyppiset työt syytä erottaa eri projekteihin.

- Jos hanke on erittäin laaja ja useista lisäävistä ja iteroivista kierroksista koostuva, kukin kierros pitää erottaa omaksi projektikseen.
- Jos hankkeen sisältö koostuu useista selkeästi erityyppisistä osista, on ne syytä erottaa omiksi projekteikseen.
- Jos hankkeeseen tulee pitkä tauko, kannattaa taukoa edeltävä ja seuraava osa jakaa omaksi projektikseen.
- Jos hankkeessa tunnistetut samanlaiset projektityypit eroavat toisistaan esim. laatuvaatimusten, riskien, projektihenkilöstön osaamisen tai kehitys- ja käyttöympäristöjen suhteen, on nämä paras erottaa omiksi projekteikseen.

Niin kuin monessa muussakin lähestymistavassa, myös tässä on hyviä ja huonoja puolia. Jakamisen tuloksena saattaa olla lukuisa määrä pieniä aliprojekteja, mutta niiden hallinnointi ja mitattavuus on niin paljon parempi, että tuloksesta hyötyvät kaikki osapuolet. (Forselius ym., 2009.) Askeleella 2 on iso merkitys, kun projekti pyritään toimittamaan ajoissa ja budjetin rajoissa. Tarkoituksena on jakaa työ pieniin ja helpommin hallittaviin paketteihin, joista keskusteleminen ja joiden ymmärtäminen molemmin puolin on yksinkertaisempaa. Taulukossa 3 on luettelo erilaisista ohjelmistoprojektityypeistä: (Forselius ym., 2009, 23)

TAULUKKO 3 ICT-projektityypit

Tyyppi
Asiakaskohtainen ohjelmistoprojekti
Ohjelmistotuoteprojekti
Ohjelmistoversioprojekti
ICT-palveluprojekti
Valmisohjelmiston konfigurointiprojekti
Tietokonversioprojekti
Integrointiprojekti

Kunkin projektin esitutkimus, määrittely, toteutus, testaus ja käyttöönotto ovat projektin vaiheita eivätkä osaprojekteja (Dekkers & Forselius, 2010).

Askel 3: Scope Manager laskee alustavat toimintopisteet jokaiselle aliprojektille ja laatii arvion koko projektin koosta

Toiminnallinen koko on tärkein parametri ohjelmistoprojektin työmäärän arvioinnissa (Forselius ym., 2009). Scope Manager voi laskea vaiheessa 2 määriteltyjen osaprojektien toiminnallisen koon asiakkaan tekemän karkean tason vaatimusmäärittelyn perusteella. On huomattava, että toiminnallisen laajuuden mittaamisen lisäksi on otettava yhtälailla huomioon myös laatu- ja tekniset vaatimukset. Laadittavan ohjelmiston toiminnallinen laajuus saadaan laskemalla yhteen sen aliprojektien laajuudet. Tällä hetkellä toiminnallisen laajuuden mittaamista varten on olemassa viisi erilaista ISO/IEC 14143-1 -standardin kanssa yhteensopivaa menetelmää. northernSCOPE™-konseptissa käytetään eniten

FiSMA 1.1 (SFS-ISO/IEC 29881, 2013) –menetelmää, mutta kaikki muutkin menetelmät ovat samalla tavoin käytettävissä. Samaa menetelmää on kuitenkin käytettävä kaikkiin aliprojekteihin. Toimintopistelaskentaa ei voi soveltaa kaikkiin projektityyppeihin. Näitä ovat esim. valmisohjelmiston konfigurointiprojektit ja ylläpitoprojektit. Tällöin on käytettävä jotain muuta arviointitapaa, kuten tuntimäärää tai kokemukseen perustuva arviointia. (Dekkers & Forselius, 2007.) Jos suunnitellun ohjelmiston toiminnalliset vaatimukset on saatu kerättyä ja dokumentoitua kattavasti, pystyy Scope Manager antamaan asiakkaalle hyvän arvion tulevan ohjelmiston laajuudesta. Jos vaatimukset ovat liian epämääräisiä, täytyy asiakkaan ainakin miettiä ne liiketoimintaprosessit, joita ohjelmiston on tarkoitus tukea. Jos asiakas ei pysty kertomaan tarpeitaan tarkasti, ei myöskään toimittaja pysty tuottamaan tarvittavaa sovellusta. Toimittajan kanssa saatetaan sopia myös vaatimusten keräämisestä, mutta silloinkin asiakkaan on tehtävä lopullinen päätös siitä, mitä halutaan ja paljonko siitä ollaan valmiita maksamaan. (Dekkers & Forselius, 2010.)

Askel 4: Asiakas ja Scope Manager määrittävät projektin laatuvaatimukset
NorthernSCOPE™-konsepti poikkeaa muista vastaavista menetelmistä siinä, että se ottaa huomioon myös jokaisen aliprojektin laadulliset vaatimukset pohjautuen ISO/IEC 9126 –laatumalliin. On selvää, että ohjelmiston ei-toiminnalliset vaatimukset voivat vaikuttaa dramaattisesti ohjelmistokehityksen työmäärään ja kustannuksiin. Usein nämä vaatimukset huomioidaan projektissa kuitenkin liian myöhään, jotta niihin voitaisiin vastata tehokkaasti. (Dekkers & Forselius, 2010.)

Askel 5: Asiakas laatii projektista tarjouspyynnön
Tässä vaiheessa asiakas on valmis Scope Managerin avustuksella laatimaan tarjouspyynnön ja arviointikriteerit lähetettäväksi joukolle ohjelmistotoimittajia. Luettelo toiminnallisista ja laadullisista vaatimuksista jaettuina aliprojekteittain liitetään tarjouspyyntöön mukaan. Näin toimittaja voi laskea ja antaa tarjouksessaan arvion yksikköhinnasta. (Dekkers & Forselius, 2010.)

Askel 6: Toimittaja valitaan toimintopistekohtaisen yksikköhinnan perusteella
Scope Manager auttaa asiakasta arvioimaan tulleet tarjoukset ja pystyy myös vertaamaan tarjottuja yksikköhintoja alalla aiemmin toteutuneisiin yksikköhintoihin. Asiakas valitsee yhden tai useamman toimittajan projektin tarpeisiin. Toiminnallisten osien hinta ilmoitetaan euroina/toimintopiste ja muissa osissa voidaan soveltaa esim. tuntihinnoittelua. Scope Manager pystyy laskemaan kokonaistyömäärän kokemuskantoihin tallennettujen samantapaisten projektien tuottoasteen avulla (tuntia/toimintopiste). (Dekkers & Forselius, 2010.)

Askel 7: Vaatimukset kiinnitetään ja dokumentoidaan
Tämä on ensimmäinen vaihe, johon Scope Manager ei aktiivisesti osallistu. Asiakas ja toimittaja työskentelevät yhdessä ja laativat, tarkentavat ja konkretisoivat aliprojektien vaatimukset. (Dekkers & Forselius, 2010.)

Askel 8: Scope Manager laskee lähtötilanteen toimintopisteet
Käyttäen edellisessä vaiheessa syntyneitä vaatimusmäärittelydokumentteja Scope Manager laskee jokaiselle vaatimukselle lähtötilanteen toimintopisteet. Samalla valmistuu perusta (baseline), johon kaikkia muutoksia ja projektin edistymistä verrataan. (Dekkers & Forselius, 2010.)

Askel 9: Scope Manager laskee haluttujen muutosten vaikutukset kustannuksiin
Kun projektiin ehdotetaan muutoksia, joko asiakkaan tai toimittajan taholta, Scope Manager kerää ja rekisteröi tiedot muutoksista ja laskee vaikutuksen toiminnallisuuteen toimintopisteinä. Muutoksesta laaditaan hinta- ja työmääräarviot perustuen alkuperäiseen toimittajan antamaan yksikköhintaan. Asiakas joko hyväksyy tai hylkää muutosehdotuksen ja hyväksytty muutos tehdään muodollisen muutoshallinnan ohjaamana. (Dekkers & Forselius, 2010.)

Askel 10: Scope Manager valvoo projektin edistymistä
Projektin kuluessa Scope Manager saa selvitykset osaprojektien etenemisestä toimittajalta. Hän rekisteröi valmiusasteet lähtötilanteeseen nähden ja toimittaa edistymisraportin tilaajalle. Tavallisesti tämä tapahtuu kuukausittain tai ketteriä menetelmiä käytettäessä sprinteittäin, mutta vaiheessa 7 on voitu yhdessä sopia muustakin raportointijaksosta. (Dekkers & Forselius, 2010.)

Askel 11: Projekti päättyy ja asiakas maksaa toteutuneiden toimintopisteiden mukaan
Tämä on toinen niistä northernSCOPE™-konseptin vaiheista, joihin Scope Manager ei osallistu aktiivisesti. Asiakas maksaa kunkin osaprojektin toimittajalle toimitettujen toimintopisteiden ja yksikköhinnan mukaisen laskun. Projekti on nyt valmis ja projektipäällikkö sulkee sen. (Dekkers & Forselius, 2010.)

Askel 12: Scope Manager kerää ja tallettaa kokemusdatan
Viimeisessä vaiheessa Scope Manager viimeistelee projektin ja tallettaa sen aikana kerätyn tiedon. Lopulliset työmäärät sekä projektin muuttujat ja attribuutit talletetaan kokemustietokantaan myöhempää käyttöä varten. (Dekkers & Forselius, 2010.)

4.4 Yhteenveto ohjelmistoprojektien laajuudenhallinnasta

Laajuudenhallinnan on todettu monissa tutkimuksissa ja myös käytännössä auttavan projektijohtoa laatimaan paikkansapitäviä ohjelmistoprojektien aikataulu- ja kustannusarvioita sekä vaatimusmäärittelyjä niin, ettei projekti paisu hallitsemattomasti. Ohjelmistoprojektin laajuuden laskeminen toimintopisteinä

mahdollisimman varhaisessa vaiheessa antaa koko projektin onnistumisen kannalta tärkeän baselinen, johon jatkossa vaatimusten tarkentuessa voidaan tukeutua. Ennen varsinaisen ohjelmoinnin aloittamista projektille on voitu laskea kokemukseen perustuva arvio tulevasta työmäärästä niin, että myös laadulliset vaatimukset on huomioitu ja mahdolliset riskit on kartoitettu. Projektin edistyessä muutostarpeiden vaikutus voidaan heti laskea ja ottaa huomioon aikataulussa ja kustannuksissa. Puolueettoman Scope Managerin käyttö hyödyttää sekä projektin tilaajaa että toimittajaa, sillä hänen laskelmansa projektin edistymisestä perustuvat ammattitaitoiseen toimintopisteisiin perustuvan mittariston käyttöön.

NorthernSCOPE™-konsepti on lähtökohtaisesti tarkoitettu yksittäisten, asiakkaan tilaamien ohjelmistoprojektien laajuudenhallintaan ja kokemukset tällä ohjelmistoteollisuuden sektorilla ovat olleet erittäin hyvät. Ohjelmistotuotteiden valmistuksessa konseptia on kuitenkin käytetty hyväksi liian vähän. Myös ohjelmistotuotteiden valmistajat tarvitsevat laadukkaita prosesseja pystyäkseen kilpailemaan omilla tuotteillaan markkinoiden paineessa. Luvussa 3 esitelty ohjelmistotuotteiden viitekehys tarjoaa kokoelman hyviä prosesseja ja parhaita käytäntöjä tuotekehityksen tarpeisiin, mutta projektin laajuudenhallinnan käsittely siitä puuttuu. Seuraavissa luvuissa tutkitaankin pienen jyväskyläläisen ohjelmistotuotteita valmistavan yrityksen käytössä olevia menetelmiä ja pyritään selvittämään, ovatko tuotehallinnan ja laajuudenhallinnan käytänteet sovitettavissa yrityksen tuotekehityksen prosesseihin.

5 TUTKIMUKSEN TOTEUTUS

Tutkimuksen empiirisen osan tavoitteena on selvittää, onko pienen, tuotteita valmistavan ohjelmistoyrityksen mahdollista omaksua ja ottaa käyttöön tuotehallinnan käytänteet ja liittää niihin mukaan myös laajuudenhallinnan konsepti. Tässä luvussa esitellään tutkittava yritys, kerrotaan tutkimusstrategiasta ja tiedonkeruumenetelmästä ja lopuksi analysoidaan saatuja tuloksia.

5.1 Tavoite ja tutkimusstrategia

Luvuissa 3 ja 4 esiteltiin kirjallisuudesta löytynyt tuotehallinnan viitekehys ja FiSMAn laatima laajuudenhallinnan konsepti. Tutkimuskysymykseksi oli asetettu "Miltä osin tuotehallinnan ja laajuudenhallinnan käytännöt ovat omaksuttavissa pienessä ohjelmistoyrityksessä?" ja lisäkysymyksenä haluttiin tutkia "Ovatko tuotehallinnan ja laajuudenhallinnan konseptit yhdistettävissä miltään osin?". Tutkimuksen empiirisessä osassa käydään kaikki tuotannonhallinnan prosessialueet läpi ja selvitetään, kuinka niitä on pystytty toteuttamaan tutkittavassa yrityksessä. Lisäksi esitetään malli, jossa laajuudenhallinta on yhdistetty tuotehallintaan ja pyritään tutkimaan mallin soveltuvuutta pienen ohjelmistoyrityksen käyttöön.

Tutkimuskohteena on pieni jyväskyläläinen ohjelmistoyritys (jatkossa Yritys), joka valmistaa ohjelmistotuotteita PK-sektorin rakennusliikkeille. Yrityksessä ei ole tietoisesti noudatettu tuotehallinnan prosesseja, mutta useat niistä ovat kuitenkin olleet käytännön työn sanelemina jo usean vuoden ajan käytössä. Yrityksessä on myös huomattu, että ylläpitoversioiden ja varsinkin uusien ohjelmien julkaisujen suunnittelussa tarvittaisiin luotettavaa menetelmää työmääräarvioiden tekemiseen. Kehitysprojektien aikana ilmenevä scope creep on ollut yhä paheneva ongelma, joten yritys haluaa löytää projektiensa laajuudenhallintaan sopivan ja tehokkaan menetelmän. Suunnitelmallisuus ja resurssien kohdentaminen oikeisiin asioihin ilman kalliita kokeiluja olisi myös saatava raiteilleen.

Tutkimusmenetelmäksi valikoitui kvalitatiivinen tapaustutkimus. Hirsjärven, Remeksen ja Sajavaaran (2009, 160-164) mukaan kvalitatiivisen tutkimuksen lähtökohtana on todellisen elämän kuvaaminen ja kohdetta pyritään tutkimaan mahdollisimman kokonaisvaltaisesti. Tutkimuksen pyrkimyksenä on löytää ja paljastaa tosiasioita pikemminkin kuin todentaa jo olemassa olevia väittämiä. Kvalitatiivisessa tutkimuksessa aineisto kootaan todellisissa tilanteissa, tapauksia käsitellään ainutlaatuisina ja tutkija määrää, mikä on tärkeää. Tutkimuksen kohdejoukko valitaan tarkoituksenmukaisesti ja tutkimussuunnitelma muotoutuu tutkimuksen edetessä, aineistoa analysoidaan monipuolisesti ja yksityiskohtaisesti. Tapaustutkimuksen olemusta ja menetelmiä selvitettiin Jyväskylän yliopiston Koppa-sivuston avulla (Lähdesmäki, Hurme, Koskimaa, Mikkola & Himberg, 2016). Sivustolla kerrotaan, että tapaustutkimus on tutkimusstrategia, jossa tarkoituksena on tutkia syvällisesti vain yhtä tai muutamaa kohdetta tai ilmiökokonaisuutta. Tapaus ymmärretään usein jollain tavoin rajautuneeksi omaksi kokonaisuudekseen ja sen tutkimuksessa pyritään tuottamaan yksityiskohtaista tai intensiivistä tietoa. Myös Järvinen & Järvinen (2011) kertovat, että tutkimus sopii tilanteisiin, joissa tutkitaan tämän päivän ilmiöitä ja tavoitellaan yksityiskohtaista tietoa ja tarkempaa ymmärrystä tutkimuskohteesta.

Tähän tutkimukseen kvalitatiivinen tapaustutkimus soveltuu hyvin, koska tutkimuskohde on tarkoin rajattu eli yksittäisen yrityksen tuotekehitysprosessi, ja tarkoituksena on tutkia ja ymmärtää sen tilaa juuri tänään. Wikipediassa (2016) laadullista tutkimusta luonnehditaan näin:

> Teoria on mukana laadullisessa tutkimuksessa kahdella tavalla: teoria keinona, joka auttaa tutkimuksen tekemisessä ja teoria päämääränä, jolloin tutkimuksella pyritään kehittämään teoriaa edelleen. Ensimmäisessä merkityksessä eli keinona laadullinen tutkimus tarvitsee sekä taustateoriaa, jota vasten aineistoa arvioidaan, että tulkintateoriaa, joka auttaa muodostamaan kysymykset ja sen, mitä aineistosta etsitään. Teoria voi olla laadullisessa tutkimuksessa myös päämääränä. Tämä tulee esiin silloin, kun tehdään induktiivista päättelyä aineiston pohjalta eli edetään yksittäisistä havainnoista yleiseen. Tällöin pyrkimyksenä on luoda uutta teoreettista tietoa.

Tutkimuksen teoriaosuudessa perehdyttiin tuotannonhallinnan ja laajuudenhallinnan yleisiin prosessimalleihin, joita sitten pyrittiin soveltamaan tämän nimenomaisen yrityksen prosesseihin. Molempien prosessimallien yhdistämisen mahdollisuus oli myös tutkimuksen tavoitteena. Tässä toteutuu edellä mainittu laadullisessa tutkimuksessa käytössä oleva teorian kahtalainen mukanaolo: taustateoriaa käytetään tutkimuksen tekemiseen ja teorian avulla laaditaan myös uusi, ehkä yleisempäänkin käyttöön soveltuva malli.

5.2 Tiedonkeruu

Tutkimuksen oletuksena oli, että prosessimallien käyttö tehostaa tuotekehitystä ja siksi tällaisia malleja ryhdyttiin etsimään kirjallisuudesta. Teoriaosuuden läh-

teitä haettiin pääasiassa Google Scholarin avulla, jolloin hakusanoina käytettiin mm. termejä 'SME', 'software product management', 'software process estimation and measurement, PEM', 'scope management' ja 'functional size measurement'. Myös löydettyjen lähteiden lähteistä löytyi monta käyttökelpoista artikkelia. Laajuudenhallinnan teoriaosuudessa käytettiin myös tunnustettujen alan tutkijoiden julkaisemia kirjoja. Lähteitä ei pyritty arvioimaan kriittisesti, vaan niitä etsittiin tarkoitushakuisesti antamaan taustatietoa tutkittavaan aiheeseen. Näin tehty tiedonkeruu auttoi myös osaltaan tutkimustuloksen syntymiseen, sillä löydettyä aineistoa pystyttiin käyttämään uuden mallin luomisessa.

Kun tutkittavasta ilmiöstä kootaan tietoa sitä seuraamalla ja tekemällä havaintoja, on käytettävä tutkimusmenetelmä nimeltään havainnointi. Havainnointia voidaan tehdä ulkopuolisesta tai sisäpuolisesta näkökulmasta suhteessa tutkimuskohteeseen. Sisäpuolisen näkökulmasta tehty havainnointi voi muodostua osallistuvaksi havainnoinniksi, jossa tutkija toimii osana havainnoitavaa tilannetta ja yhteisöä. (Lähdesmäki ym., 2016.)

Havainnointi valikoitui tiedonkeruumenetelmäksi, koska tutkijalla oli erinomainen mahdollisuus sisäpuolisen näkökulmasta tehtyyn havainnointiin. Hän on työskennellyt vuosikausia Yrityksen tuotekehitystiimissä eri tehtävissä ja toimii nyt System Managerina. Taustansa vuoksi hän tuntee Yrityksen tuotekehitysprosessin läpikotaisin ja voi arvioida teorian ja mallien soveltuvuutta Yrityksen käytäntöihin. Tarkoituksena ei ollut kuitenkaan esittää pelkästään tutkijan omaa näkemystä, joten tutkimustuloksen syntyneen mallin soveltuvuutta testattiin haastattelemalla neljää Yrityksessä töissä olevaa henkilöä. Näin pyrittiin löytämään mahdollisia mallin soveltamisen esteitä. Haastatteluun valittujen pieni määrä sopii myös siihen, mitä Wikipediassa (2016) sanotaan laadullisen tutkimuksen otannasta: "Laadullisessa tutkimuksessa käytetään yleensä harkinnanvaraista otantaa. Tutkittavia yksiköitä ei valita kovin suurta määrää ja niitä tutkitaan perusteellisesti, jolloin tärkeää on aineiston laatu."

5.3 Analyysi

Analyysimenetelmänä käytettiin laadullista analyysia. Laadullisen eli kvalitatiivisen analyysin tavoitteena on jäsentää tutkimuskohteen laatua, ominaisuuksia ja merkityksiä kokonaisvaltaisesti. Laadullisissa menetelmissä yhteisenä piirteenä korostuu muun muassa kohteen esiintymisympäristöön ja taustaan, kohteen tarkoitukseen ja merkityksiin liittyvät näkökulmat (Lähdesmäki ym., 2016.) Tässä tutkimuksessa tarkoituksena oli tulkita Yrityksen tuotekehityksen toimintoja siihen soveltuvan viitekehyksen läpi ja tehdä havaintoja viitekehyksen toimivuudesta. Tähän lisättiin laajuudenhallinnan näkökulma ja tutkittiin, antaisiko se Yrityksen toimintoihin tarvittavaa lisäarvoa projektien hallinnan alueella.

Löydettyä van de Weerdin ym. (2006) tuotehallinnan viitekehystä (Liite 1) yksinkertaistettiin hieman aluksi (Kuvio1). Sen jälkeen Yrityksen tuotekehityk-

sen toiminnoista pyrittiin löytämään ne käytänteet, jotka sopivat malliin ja toisaalta ne, jotka eivät. Jokainen viitekehyksen prosessialue ja sidosryhmä käytiin läpi ja analysoitiin, löytyykö Yrityksen nykyisistä toiminnoista niille vastaavuutta. Seuraavassa vaiheessa laadittiin malli, jossa laajuudenhallinnan konsepti NorthernSCOPE™ liitettiin tuotehallinnan viitekehykseen ja jokaista konseptin 12 askelta analysoitiin kriittisesti. Tarkoituksena oli löytää ne kohdat, joissa mallin soveltaminen on Yrityksessä mahdollista ja ne kohdat, joissa tätä mahdollisuutta ei ole. Jälkimmäisten kohdalla esitetään myös muutosehdotuksia Yrityksen tuotekehityksen toimintaan, jos se katsottiin tarpeelliseksi.

Luodun mallin toimivuutta arvioitiin haastattelujen avulla. Haastatteluun valittiin neljä henkilöä, joita toimenkuvansa perusteella voidaan pitää prosessialueiden ”omistajina”. Haastattelut tehtiin keväällä 2016 niin, että kerralla haastatteluun osallistui kaksi henkilöä. Ensin haastateltaville esiteltiin luotu uusi malli ja sitten keskustellen pyrittiin löytämään siitä hyviä ja huonoja piirteitä. Keskustelun tukena käytettiin Willmanin (1996) laatimaa kehystä, jossa esitetään edellytykset minkä tahansa organisatorisen muutoshankkeen onnistumiselle. Keskusteluja ei nauhoitettu, mutta niistä tehtiin muistiinpanoja, jotka heti keskustelun jälkeen kirjoitettiin puhtaaksi.

6 TUOTEHALLINNAN VIITEKEHYKSEN JA NORTHERNSCOPE™-KONSEPTIN YHDISTÄMINEN

Tämä luku on tutkimuksen empiirinen osuus. Yrityksen tuotehallintaa tarkastellaan viitekehyksen avulla prosessialue kerrallaan, esitetään laajuudenhallinnan ja tuotehallinnan yhdistetty malli ja lopuksi haastattelujen avulla selvitetyt haasteet mallin käyttöönottoon Yrityksessä.

6.1 Tuotesalkun hallinta

Yrityksen tuote koostuu useista erillisistä palasista, joita rakennusliike voi ottaa käyttöön tarpeidensa mukaan. Kyseessä ei ole tuoteperhe sanan varsinaisessa merkityksessä, vaan paremminkin kokoelma erillisiä, toisiinsa integroituja tuotteita, jotka muodostavat kokonaisjärjestelmän toiminnanohjausta varten. Mikään tuotteista ei toimi täysin yksin, sillä mukana on aina oltava muutamia ydintuotteita. Kokoelman tuotteet on kaikki integroitu toisiinsa niin, ettei samaa tietoa tarvitse syöttää ja käsitellä useampaan kertaan ja kertaalleen syötetty tieto näkyy reaaliaikaisesti eri puolilla ohjelmistoa. Tuotekokoelma kattaa lähes kaikki rakennusliikkeen tarvitsemat prosessit rakennusprojektin käynnistämiseen, hallintaan ja seurantaan lähtien tarjouslaskennasta ja päätyen työmaan tuotannonhallinnan ja yrityksen taloushallinnan kautta kirjanpitoon ja viranomaisilmoituksiin. Tuotekokoelma on elinaikanaan kirjoitettu monta kertaa uudelleen alustojen vaihtuessa. Nykyinen versio on koodattu Microsoftin VB6-kielellä ja tietokantana on Microsoftin SQL Server. Parhaillaan on meneillään jälleen uuden ohjelmistoalustan vaihto. Osa ohjelmistosta on jo koodattu Microsoftin VB.NET-kielellä ja seuraavien kahden vuoden kuluessa on tarkoitus siirtää loputkin ohjelmiston osat uudelle alustalle. Myös web- ja mobiilipohjaisia käyttöliittymiä lisätään tuotteeseen jatkuvasti. Ohjelmisto tarjotaan asiakkaille paria poikkeusta lukuun ottamatta ainoastaan SaaS-palveluna.

Van de Weerdin ym. (2006) viitekehyksessä mainituista ulkoisista sidosryhmistä voidaan Yrityksen tapauksessa tunnistaa seuraavat; markkinat eli

suomalaiset PK-sektorin rakennusliikkeet; kumppanit eli esimerkiksi SaaS-palvelinalustan tarjoaja; asiakkaat eli nykyiset n. 450 asiakasyritystä, ja lainsäädäntö, joka varsinkin viime vuosina on myllertänyt rakennusalaa harmaiden markkinoiden torjuntatoimenpiteillä. Yrityksen tuotestrategiaan kuuluu, että kolmannen osapuolen ohjelmistoja integroidaan mahdollisuuksien mukaan omaan ohjelmistoon. Niinpä vuosien kuluessa on toteutettu lukuisia erilaisia integraatiorajapintoja; mm. kirjanpito-, konevuokraus-, tuntikirjaus-, määrälaskenta- ja aikataulutusohjelmiin. Näiden lisäksi on tehty myös rajapinnat verkkolaskuoperaattoreiden, työnantajaliittojen, verottajan ja muiden yhteistyötahojen järjestelmiin. Tulevaan tuotestrategiaan kuuluu myös, ettei kaikkia ohjelmiston osia valmisteta enää itse, vaan pyritään löytämään valmiita ohjelmistoja, jotka sopimuksin ja integraatioin liitetään kiinteäksi osaksi nykyistä tuotevalikoimaa. Tästä aiheutuu väistämättä se, että kumppaneiden määrä kasvaa ja niiden hallinnointi tulee olemaan Yrityksessä tärkeä haltuun otettava toiminto. Viitekehyksessä mainittuja sisäisiä sidosryhmiä ovat service desk, kouluttajat, myynti ja markkinointi sekä tietenkin Yrityksen ja sen taustalla olevan konsernin johto. Sidosryhmiä on siis lukuisia, ja kaikkien niiden toiveet ja tarpeet on huomioitava tuotesalkun hallinnassa.

Rakennusalan IT-järjestelmät ovat perinteisesti olleet käytössä toimistoissa ja niillä on hoidettu taloushallintoa, tarjouslaskentaa, hankintaa, kustannusseurantaa, sopimuksia ja jälkilaskentaa. Yhä enenevässä määrin tietokoneet ja mobiililaitteet ovat tulleet nyt myös varsinaisille työmaille. Niillä hoidetaan työmaan reaaliaikaista seurantaa ja ohjausta, sekä omien ja alihankkijoiden työntekijöiden kulunseurantaa, perehdytystä ja laillisuusvalvontaa. Harmaat markkinat ja pimeä työvoima ovat olleet pitkään rakennusalan vitsaus, mutta lainsäädännöllisin toimin niistä pyritään nyt pääsemään eroon. Verottajan ja Aluehallintoviraston vaatima seuranta ja raportointi eivät enää onnistu ilman sitä varten kehitettyjä järjestelmiä ja sähköisiä menetelmiä. Yritys on tunnistanut tämän markkinatrendin ajoissa ja tarjoaa siihen liittyviä palveluja ja tuotteita integroituna muuhun kokonaisjärjestelmään.

Tuotevalikoimassa on elinkaareltaan hyvin eri vaiheissa olevia palasia. Vanhimmat osat ovat vuodelta 2000 ja uusimmat viime vuodelta. Käynnissä on tuotekehitysstrategiaan kuuluva uudistus, jossa vanhimmat osat joko kirjoitetaan uudelleen .NET-alustalle tai korvataan kumppaneilta ostettavilla valmiilla ohjelmistoilla. Se, kumpaan vaihtoehtoon päädytään, riippuu oleellisesti resurssien eli tuotekehittäjien riittävyydestä. Vaihtoehtojen puntaroinnissa tarvittavien laskelmien tuottamisessa laajuudenhallinta voi olla hyödyksi. Nykyisen ohjelmiston toimintopisteet laskemalla saadaan vähimmäisarvio työmäärälle, jonka uuden ohjelmiston kehittäminen vaatisi. Onhan uudessa tuotteessa oltava ainakin samat toiminnot kuin nykyisessä ja luultavasti myös lukuisia uusia.

.NET-alustalle jo siirrettyjen ohjelmisto-osien kehityksen aikana on samalla tehty oma aliohjelmakirjasto, framework, joka sisältää lukuisan määrän erilaisia, uudelleenkäytettäviä toimintoja. Kirjaston käytöllä saavutetaan etua, kun eri puolilla ohjelmaa toistuvat rakenteet ja komponentit on määritelty valmiiksi. Tällä varmistetaan myös se, että kaikki toiminnot näyttävät ja tuntuvat käyttä-

jän näkökulmasta samanlaisilta. Näin esimerkiksi taulukot eli gridit toimivat aina täsmälleen samoin tai hiiren kakkospainikkeen alta löytyvät helposti kaikki mahdolliset, samanlaiset lisätoiminnot. Samoin uuden ohjelmiston ulkonäkö on kaikkialla yhteneväinen. Kun uuden ohjelmiston laajuutta hahmotetaan toimintopisteinä, voidaan tämä framework laskea yhtenä tuotteena erikseen ja siihen tehtävien muutosten ja lisäysten työmäärän laskennassa on käytettävä suurehkoa uudelleenkäytön kerrointa. Kun frameworkissa olevia komponentteja, esim. taulukoita ja painikkeita, käytetään ohjelmistossa, tarvitsee niistä siis laskea vain lukumäärä, koska kaikki toiminnallisuus on otettu huomioon jo frameworkin laajuuden laskennassa.

Koko ohjelmiston tietovarastona on Microsoftin SQL Server –tietokanta. Kaikki ohjelmiston käyttämät taulut ovat samassa tietokannassa, jolloin kukin tieto syötetään vain kertaalleen ja tieto liikkuu eri ohjelmiston osien välillä saumattomasti. Laajuudenhallinnan näkökulmasta tietokanta voidaan myös käsitellä omana tuotteenaan ja laskea sen laajuus toimintopisteinä. Kun tietokantaan tehdään muutoksia ja lisäyksiä, vaikutus kohdistuu enimmäkseen vain tietokannan toimintopisteisiin ja ainoastaan vähäisessä määrin muuhun ohjelmistoon.

Yrityksessä on tällä hetkellä 5-7 tuotepäällikköä laskentatavasta riippuen. Joillakin tuotepäälliköistä on myös muita tehtäviä tuotteesta vastaamisen ohella. Jokaisella tuotepäälliköllä on useita tuotteita hallittavanaan ja varsinkin ydintuotteiden kohdalla rajanveto siitä, kenen tuotepäällikön vastuulle tuote kuuluu, on hankalaa. Myös historiallisista syistä johtuen tuotepäällikön salkkuun on saattanut jäädä osa jostakin toiselle tuotepäällikölle kuuluvasta tuotteesta. Tuotesalkun vastuiden uudelleenorganisointi on yksi Yrityksen tulevista tehtävistä.

6.2 Roadmapping

Yrityksessä on parhaillaan menossa roadmapping-projekti, jonka tuloksena syntyy vuoteen 2020 ulottuva tuotteiden roadmap. Projektissa pyritään miettimään, miltä rakentamisen maailma näyttää vuonna 2020 ja vastaamaan omilla tuotteilla silloin vallitsevaan kysyntään. Koska Yritys on kiistaton markkinajohtaja PK-sektorin rakennusliikkeille tarjottavien ohjelmistojen saralla, se tuntee myös vastuunsa kysynnän ohjaajana ja tarjottavien palvelujen sekä ratkaisujen kehittäjänä. Roadmappingissa tehtävillä päätöksillä ei ole vaikutusta vain Yrityksen omaan toimintaan vaan myös koko PK-sektorin rakentamistoimialan IT-kehitykseen Suomessa. Siksi roadmapping-prosessissa on ollut mukana rakentamistoimialaa syvällisesti tuntevia asiantuntijoita konsultteina. Omasta henkilöstöstä roadmapping-prosessiin osallistuvat kaikki tuotepäälliköt, mutta myös muiden tiimien kuin tuotekehityksen jäseniä tarvitaan kokonaiskuvan muodostamiseen.

Yrityksen kokonaisjärjestelmä on kehittynyt yli kolmenkymmenen vuoden kuluessa ja siinä on rönsyjä, joiden olemassaolo ei ole enää perusteltavissa. Joistakin geneerisistä tuotteista, kuten maksuliikenteestä ja pääkirjanpidosta, on

jo luovuttu aiemmin ja nyt on tarkoitus päättää siitä, mitkä tuotteet ovat kokonaisjärjestelmään oleellisesti kuuluvia ydinosia. Karsitut osat joko poistetaan, yhdistetään uudistettaviin tuotteisiin tai ne korvataan valmiilla ohjelmistoilla, jotka integroidaan kokonaisjärjestelmään. Yritys haluaa kohdistaa vähäiset tuotekehitysresurssinsa niihin osioihin, joissa se tuntee olevansa vahva osaaja. Roadmapin luomisen teemana tulee olemaan rakentamiseen suoraan liittyvien osien valitseminen ja niiden kehittämistyön aikatauluttaminen. Keskittymällä pelkästään rakentamisosioihin Yritys pyrkii myös laajentamaan asiakassegmenttiään suurten rakennusliikkeiden suuntaan.

Sama tuotekehitystiimi, joka tulee kehittämään uudet tuotteet, joutuu myös huolehtimaan olemassa olevien tuotteiden ylläpidosta. Ylläpitoversioita toimitetaan kaksi vuodessa, keväällä ja syksyllä. Versioprojektit pyritään pitämään suhteellisen pieninä, sillä piakkoin kokonaan uusittaviin ohjelmiin ei ole enää mielekästä eikä kannattavaa tehdä isoja muutoksia. Versioprojekteja varten varataan roadmapiin kiinteät ajanjaksot, joiden aikana uuden tuotteen kehittäminen on pysähdyksissä. Kaikki tuotekehitysresurssit ovat siis tällöin lyhyen aikaa vain versioprojektin käytössä.

Yhtenä resursoinnin mahdollisuutena pidetään myös tuotekehitystyön ostamista alihankkijoilta. Tästä on Yrityksessä jo jonkin verran kokemusta, sekä hyvää että huonoa. Näin säästetään kyllä varsinaisten kehittäjien aikaa, mutta tuotepäälliköiltä vaaditaan tällä tavoin toimittaessa suurempaa työpanosta. Ulkopuolisen kehittäjän ohjaaminen on työläämpää ja vaatimusten kunnolliseen määrittelemiseen on löydyttävä enemmän aikaa. Vaatimusten kirjoittaminen ulkopuoliselle kehittäjälle on tehtävä paljon yksityiskohtaisemmin ja tarkemmin kuin samassa työtilassa työskentelevälle omalle kehittäjälle. Niinpä näiden alihankintaprojektien vetämiseen on hankittava lisää osaamista.

Tuotteiden roadmapia hahmoteltaessa on siis päätettävä, mitkä tuotteista tehdään uudestaan, minkä kehittäminen lopetetaan, mitkä ostetaan valmiina tuotteina ja minkä kehitys ulkoistetaan. Scope Managerin tehtävänä tulisi olla nykyisen kokonaisjärjestelmän kaikkien tuotteiden koon laskeminen toimintopisteinä. Tästä laskennasta saadaan lähtötaso tuotteiden uudelleenkehityksen työmäärälle. Kun tuotepäälliköiltä valmistuu uusia vaatimusmäärittelyitä, Scope Manager pystyisi laskemaan tuotteiden laajuudet ja arvioimaan työmäärät tarkemmin. Roadmapissa arvioitu työmäärä näkyy palkin pituutena nykyisillä resursseilla mitattuna. Palkki siis lyhenee, jos resursseja saadaan lisää; tai palkkeja voidaan sijoittaa päällekkäin, jos kehitystä ulkoistetaan. Tuotteet on sijoitettava aikajanalle sen mukaan, missä järjestyksessä ne voidaan toteuttaa. Ajankohtien järjestyksessä on otettava huomioon se, että jonkin tuotteen kehittäminen ei ole mahdollista ennen kuin jokin toinen tuote on valmiina. Myös nykyisten ohjelmistojen päivitysajankohdat on merkittävä aikajanalle. Lyhyen aikavälin roadmapista muodostuu näin samalla suunnitelma julkaisuista, joka sisältää sekä versiopäivitykset että uudet tuotteet. Scope managerin tehtävänä olisi jatkossa seurata tuotteiden valmistumisen tilannetta ja tehdä muutosten ilmetessä vastaavat muutokset roadmapiin.

6.3 Vaatimusten hallinta

Vaatimusten hallinta on avainasemassa ohjelmistotuotteita valmistavassa yrityksessä (Carlshamre & Regnell, 2000). Vaatimuksia tulvii kaikilta sidosryhmiltä ja niiden kerääminen ja pitäminen järjestyksessä on tuotepäällikön tärkein tehtävä. Yrityksessä ei ole erikseen tuotekehitystä ja ylläpitoa tekeviä tiimejä, joten tuotepäällikkö joutuu huolehtimaan molempien tehtäväkenttien vaatimusmäärittelystä.

Yrityksessä on käytössä kaksiportainen järjestelmä, jolla nykyisen tuotannossa olevan tuotekokoelman eli ylläpidon vaatimuksia hallitaan. Service Desk vastaanottaa asiakkailta tulevat kehitysehdotukset ja kirjaa ne CRM-järjestelmään tuotekehitysideoiksi. Myös oman talon sisältä tulevat ideat ja ehdotukset kirjataan samaan järjestelmään. Tuotepäällikön tehtävänä on seurata omien tuotteidensa tilannetta CRM-järjestelmässä ja joko hyväksyä tai hylätä tulleet tuotekehitysideat. Hylkäämisen syyt voivat olla moninaiset: esitetty idea on liian spesifi eli se ei hyödyttäisi laajaa asiakaskuntaa; idea on esitetty tuotteeseen, johon ei enää sen elinkaaren tässä vaiheessa tehdä muutoksia; idea on esitetty ja kirjattu jo aiemmin; idea on hyvä, mutta tullaan toteuttamaan vasta tuotteen uudistetussa versiossa jne. Hylättyyn ideaan kirjoitetaan kommentit hylkäyksen syistä ja Service Deskin tehtävänä on saattaa nämä syyt ehdotuksen tehneen asiakkaan tietoon. Jos idea on kehityskelpoinen, tuotepäällikkö siirtää idean tuotekehityksen versionhallintajärjestelmään, Microsoftin Team Foundation Serveriin (TFS), tehtäväksi. Alkuperäiseen tuotekehitysideaan kirjataan tehtävän numero seurantaa varten ja tehtävälle kirjataan samoin idean tunnus. Näin molemmista järjestelmistä on mahdollista jäljittää vaatimuksen tilanne. Versiojulkaisua suunniteltaessa TFS:iin kirjatuista tehtävistä valitaan sitten ne, jotka versioon toteutetaan. Yrityksessä ei tällä hetkellä nähdä tarvetta sille, että ylläpitoversion tehtäville laskettaisiin laajuus toimintopisteinä ja useimmissa tapauksissa se olisikin mahdotonta. Ylläpitotehtävät ovat usein hyvin pieniä tai ne eivät vaikuta varsinaiseen toiminnallisuuteen, jolloin toimintopistelaskenta ei ole mielekästä.

Tuotepäällikkö määrittelee myös uusien tuotteiden ominaisuudet. Vaatimusmäärittelyt on tehty perinteisesti luonnollisella kielellä jollakin tekstinkäsittelyohjelmalla. Määrittelyihin on lisätty tarvittaessa kuvaruutumalleja ja tietoluetteloita. Viime vuonna Yrityksessä otettiin käyttöön vaatimusmäärittelyjen tekemiseen tarkoitettu erikoistyökalu CaseComplete®. Sen avulla voidaan määritellä käyttötapaukset, vaatimukset, määritelmät, luokat, testitapaukset ja liiketoimintasäännöt ja tulostaa sitten erilaisia kaavioita ja yhtenäinen dokumentti lopullisesta vaatimusmäärittelystä. Työkalu on vielä koekäytössä, mutta sen avulla pyritään saamaan järjestelmällisyyttä ja yhtenäisyyttä kaikkiin tuleviin tuotekehitysprojektien määrittelyihin. Missään vaiheessa ei pyritä siihen, että uuden tuotteen vaatimusmäärittely olisi heti alusta pitäen täydellinen. Se on vain tuotepäällikön näkemys tulevan tuotteen ominaisuuksista ja määrittely tarkentuu sitä mukaa, kun tuotetta ryhdytään tekemään. Arkkitehdin ja ohjel-

mistosuunnittelijan ammattitaidon hyödyntäminen on suunnitteluvaiheessa ensiarvoisen tärkeää ja määrittelyjä täytyy pystyä muuttamaan vielä toteutusvaiheessakin. Työkalun avulla pyritään siihen, että vaatimusmäärittelyyn projektin kestäessä tehdyt muutokset kirjataan myös määrittelydokumenttiin, jolloin testaajilla tulisi olemaan ajan tasalla oleva määrittely helpottamassa testausta. CaseComplete®:n tuottama dokumentti on riittävä siihen, että Scope Manager pystyisi laskemaan ensimmäisen arvion tulevan tuotteen laajuudesta. Projektien laajuuslaskennassa on tarkoitus käyttää 4SUM Partnersin julkaisemaa työkalua Experience® Service.

6.4 Julkaisujen suunnittelu

Roadmapissa suunniteltuina aikoina Yritys julkaisee joko ylläpitoversion tai lanseeraa täysin uuden tuotteen olemassa olevan tilalle. Ylläpitoversion julkaisun suunnittelu alkaa siitä, että julkaisupäivämäärä lyödään lukkoon ja siitä lasketaan taaksepäin tarvittavat aikaikkunat suunnittelulle, toteutukselle, testaukselle ja dokumentoinnille. Suunnitteluvaiheessa tuotepäällikkö valitsee TFS:stä aiemmin hyväksymistään tuotekehitysideoista joukon vaatimuksia, jotka tullaan julkaisemaan seuraavassa versiossa. Hän määrittelee vaatimukset tarkemmin ja lisää myös uusia, esim. lainsäädännöstä tulevia, vaatimuksia tehtäviksi. Jokainen tehtävä priorisoidaan asteikolla korkea/normaali/alhainen. Tuotepäälliköt pitävät yhdessä Service Deskin ja koulutuspalvelujen edustajan kanssa kokouksen, jossa jokainen esittelee valitsemansa tehtävät ja niistä valitaan keskustelujen jälkeen joukko, jotka tullaan toteuttamaan seuraavaan versioon. Suunnitelma katselmoidaan Yrityksen johtoryhmässä ja sillä on mahdollisuus ehdottaa vielä muutoksia suunnitelmaan. Toteutus tehdään priorisoidussa järjestyksessä ja tehtäviä tehdään niin monta kuin annetussa aikaikkunassa on mahdollista. Toteutuksen aikana on ollut tyypillistä, että julkaisun laajuus kasvaa, sillä aina jostain ilmestyy sellaisia tehtäviä, jotka on ”pakko” sisällyttää julkaisuun. Uusien tehtävien mukaan ottamisesta päättää viime kädessä tuotekehitysjohtaja, mutta toteutuksen aikaikkunaa ei laajenneta kevyesti. Julkaisun päivämäärä ei jousta yhtään, sillä se on jo ehditty tiedottaa asiakaskunnalle. Toteutuksen kestäessä testaajat ryhtyvät testaamaan valmistuvia tehtäviä ja tuotepäälliköt tekevät tarvittavat muutokset käyttöohjeisiin sekä valmistavat esittelydokumentit tuotteensa uusista ominaisuuksista. Kun kaikki versioon tulevat tehtävät on tehty ja alfa-testattu tuotekehitystiimissä, Yrityksen kouluttajat ja asiakastukihenkilöt aloittavat beta-testauksen. Löydetyt virheet ja puutteet raportoidaan tuotekehittäjille, jotka korjaavat ne. Loppuun asti testausta ja korjatusta versiosta kootaan asennuspaketti, joka asennetaan SaaS-palvelimille sovittuna ajanhetkenä.

Uuden tai uudistettavan tuotteen julkaisun suunnittelu keskittyy eniten myynnin, markkinoinnin ja koulutuspalvelun tiimeille. Tuotepäällikön tehtävänä on siksi ensin ”myydä” uusi tuote näille tiimeille. Parhaiten tämä onnistunee esittelemällä uuden tuotteen ominaisuuksia sopivin väliajoin sitä mukaa

kun sen osia valmistuu. Näin myyjät voivat kertoa uudesta tuotteesta omille asiakkailleen jo etukäteen, markkinointi voi ryhtyä valmistelemaan tiedottamista ja materiaaleja hyvissä ajoin ja koulutuspalvelu voi ryhtyä suunnittelemaan asiakkaiden kurssittamista. Kaiken oheistoiminnan pitäisi olla valmista siinä vaiheessa kun uusi tuote lopulta julkaistaan. Tuotejulkaisuja koordinoimaan Yrityksessä on perustettu ”tuoteraati”, jonka tehtävänä on katsoa, että roadmap ja julkaistavat tuotteet pysyvät aikataulussa sovitulla tavalla.

6.5 northernSCOPE™-askeleet yrityksen tuotehallinnassa

Seuraavaksi tutkitaan, onko northernSCOPE™-konseptin 12 askeleella vastineensa tuotehallinnan prosesseissa. Osa askelista on sellaisenaan samoja kuin asiakaskohtaisissa projekteissa, mutta osaa niistä täytyy soveltaa. Taulukkoon 4 on koottu kaikki konseptin 12 askelta sekä niiden ehdotetut soveltuvuudet tuotekehitysprojektiin.

TAULUKKO 4 northernSCOPE™-konseptin 12 askelta tuotekehitysprojektiin sovellettuna

Askel	northernSCOPE™-konsepti	Tuotekehitysprojekti
1	Asiakas laatii karkean tason vaatimukset ja Scope Manager (SM) kiinnitetään	Tuotepäällikkö laatii karkean tason vaatimukset tai käytetään olemassa olevan tuotteen toiminnallisuutta pohjana vaatimuksille.
2	Projekti jaetaan pienempiin aliprojekteihin	Tuotekokoelma jakaantuu roadmapissa määriteltyihin osaprojekteihin, joiden on valmistuttava tietyssä järjestyksessä. Yhteinen framework ja tietokanta käsitellään omina tuotteinaan.
3	SM laskee alustavat toimintopisteet jokaiselle aliprojektille ja laatii arvion koko projektin koosta	SM laskee käytettävissä olevien tietojen pohjalta alustavat toimintopisteet jokaiselle osaprojektille.
4	Asiakas ja SM määrittävät projektin laatuvaatimukset	Tuotepäällikkö ja SM määrittävät projektin laatuvaatimukset. Tuotekehitystiimi arvioidaan ND21-menetelmällä.
5	Asiakas laatii projektista tarjouspyynnön	SM voi laskea projektin kustannus- ja työmääräarvion kokemusperäisen yksikköhinnan ja tuottoasteen perusteella
6	Toimittaja valitaan toimintopistekohtaisen yksikköhinnan perusteella	Yrityksen johto tekee päätöksen, aloitetaanko tuotekehitysprojekti

7	Vaatimukset kiinnitetään ja dokumentoidaan	Tuotepäällikkö kiinnittää ja dokumentoi vaatimukset
8	SM laskee lähtötilanteen toimintopisteet (baseline)	SM laskee lähtötilanteen toimintopisteet (baseline)
9	SM laskee haluttujen muutosten vaikutukset kustannuksiin	SM laskee haluttujen muutosten vaikutukset kustannuksiin ja aikatauluun
10	SM valvoo projektin edistymistä	SM valvoo projektin edistymistä
11	Projekti päättyy ja asiakas maksaa toteutuneiden toimintopisteiden mukaan	Projekti päättyy ja tuotepäällikön johdolla tuote siirtyy tuotantoon ja ylläpitoon
12	SM kerää ja tallettaa kokemusdatan	SM kerää ja tallettaa kokemusdatan

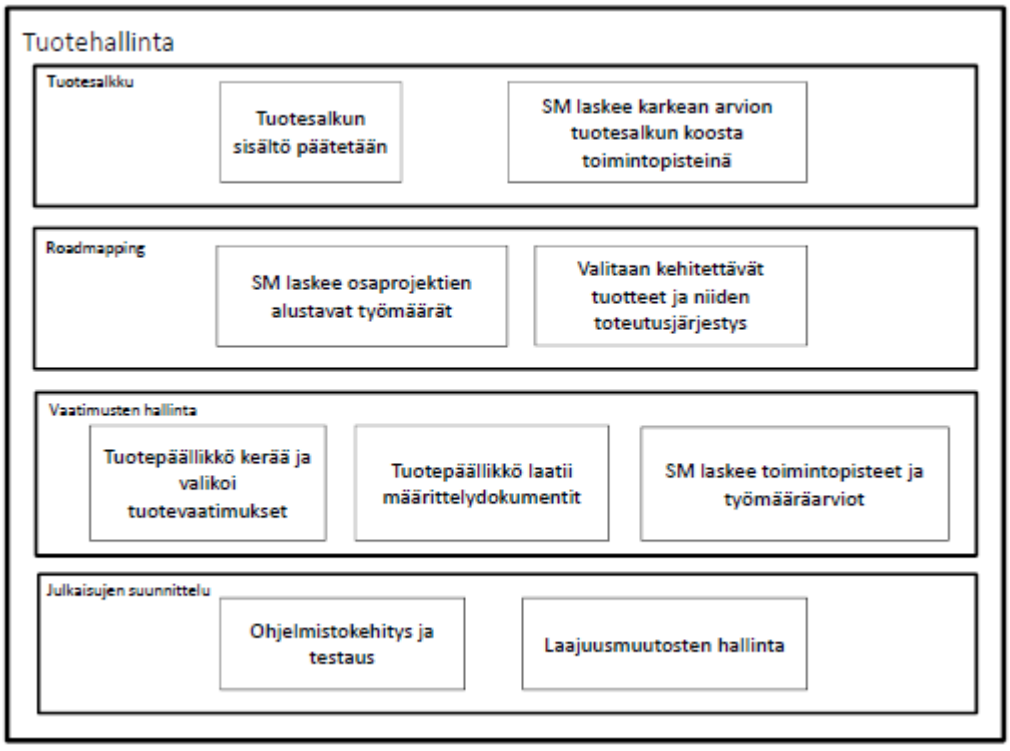
Lienee luonnollista ja itsestään selvää, että konseptin askeleet 7-12 ovat lähes identtiset sekä asiakaskohtaisessa että tuotekehitysprojektissa. Nämä askeleethan tapahtuvat varsinaisen kehittämisen aikana ja erityyppisten ohjelmistojen kehittämisprojektit eivät sanottavasti eroa toisistaan sen jälkeen, kun työhön on päästy käsiksi. Vaatimukset täytyy kiinnittää ja muutoksia on hallittava koko projektin keston ajan. Scope Manager valvoo ehkä enemmän ”tilaajan” eli Yrityksen etua, mutta yhteistyö ”asiakkaan” eli tuotepäällikön kanssa täytyy olla hyvin tiivistä. Kummankin toimijan päämääränä on kuitenkin saada valmiiksi mahdollisimman laadukas tuote resurssien antamissa rajoissa ja kustannus- ja työmääräarviossa pysyen. Kun projektin tilanne on koko ajan tiedossa, voidaan mahdolliset ulkopuolelta tulevat ennakoimattomat keskeytykset ja ongelmat resursoinnissa hoitaa tehokkaammin. Tai ainakin niiden vaikutus voidaan paremmin osoittaa Yrityksen johdolle. Projekti ei näin veny jostain tuntemattomasta syystä, vaan jokaiselle viivästykselle on osoitettavissa sen alkuperäinen aiheuttaja.

Tuotekehitysprojektissa kuusi ensimmäistä askelta eivät käsittele pelkästään yhtä tilattavaa hanketta vaan niissä otetaan kantaa koko Yrityksen tuotesalkun tilaan. Kun salkussa olevien tuotteiden karkeat koot ovat selvillä, on uusien tuotteiden kehittämisen aloittaminen roadmapin mukaisessa järjestyksessä hallitumpaa. Työmääräarviot ovat laskettavissa kokemusperäisen tuottoasteen avulla ja Yrityksen johto pystyy sisäisen tuntihinnan perusteella arvioimaan projektin kustannukset ja tulevaisuudessa odotettavissa olevat tuotot suhteellisen tarkasti. Näin johdon on helpompi laatia projekteille budjetit ja antaa niille aloitusluvat. Jos resursseja ei ole riittävästi tai tuotto-odotus ei täyty, voi johto etsiä jonkin muun tavan toteuttaa kehitysprojekti tai etsiä jo markkinoilla olevan valmiin tuotteen omansa korvaajaksi. Vaihtoehtoisen toteutustavan työmäärän arvioinnissakin projektista lasketut toimintopisteet ovat tärkeänä apuna, joten niiden laskentaan käytetty aika ei valu hukkaan.

NorthernSCOPE™-konseptia käyttöönotettaessa Yrityksen lienee parasta nimittää ensin Scope Manager. Hänen tehtävänsä olisi aluksi laskea nykyisten tuotteiden toiminnallisuus ja määrittää niiden avulla karkeat työmääräarviot.

Jatkossa hänen täytyisi pisteyttää tuoteroadmapin mukaiset määrittelyt heti niiden valmistuttua. Scope Managerin työlle on varattava riittävästi aikaa ja tuotepäälliköt olisi ainakin jossain määrin perehdytettävä toimintopistelaskennan periaatteisiin. Scope Manager on velvoitettava esittämään ajan tasalla olevat laskelmat projektikatselmuksissa ja hänen on myös säännöllisesti informoitava kaikkien meneillään olevien projektien tilanteet ja valmiusasteet yritysjohdolle. Scope Manager toimisi siis yhdistävänä tekijänä projektin osapuolien välillä ja voisi esittää tarkkoja faktoja projektin edistymisestä ja jäljellä olevasta työmäärästä. Tietyssä vaiheessa mukaan otetaan sitten tuotteen julkaisusta ja markkinoinnista vastaavat tiimit. Myös asiakkaiden informointi tulevista tuotevaihdoista ja uusista tuotteista voidaan aloittaa varmemmalta pohjalta, kun aikataulut ovat paremmin selvillä.

Kuviossa 12 esitetään Yritykselle soveltuva tapa yhdistää tuotehallinnan prosessialueet laajuudenhallinnan kanssa:



KUVIO 12 Tuotehallinnan prosessialueet yhdistettynä laajuudenhallintaan

Kuten huomataan, Scope Manager voi jokaisella tasolla auttaa arvioimaan tulevien ja menossa olevien kehitysprojektien laajuutta ja sitä kautta työmäärien arviointia. Alussa arviot ovat karkeita, mutta niidenkin avulla johto voi tehdä päätöksiä projektien käynnistämisestä ja toteutustavoista. Roadmapin muodostamiseen saadaan lujempia rakennuspalikoita, kun osaprojektien työmäärät ja resurssitarpeet ovat edes alustavasti selvillä. Tuotepäällikkö hallinnoi vaatimuksia, mutta heti kun määrittelydokumentit alkavat valmistua, Scope Manager voi niiden perusteella laskea toimintopisteiden baselinen ja työmäärän kehitysprojektille. Julkaisujen suunnittelu ei tässä pienessä Yrityksessä ole pelkäs-

tään suunnittelua, vaan siinä on huomioitava myös varsinainen julkaisun valmistaminen eli ohjelmistokehitys ja laajuusmuutosten hallinta. Tuotepäällikön on vedettävä tuotekehitysprojektia tiiviissä yhteistyössä kehittäjien, testaajien ja Scope Managerin kanssa, sillä Yrityksessä käytössä oleva iteratiivinen ohjelmistokehitysmenetelmä vaatii kaikkien osapuolten näkemystä ja osaamista.

Uusien prosessien käyttöönotto ja omaksuminen eivät ole koskaan itsestään selvyyksiä organisaation toiminnassa. Käytössä olevien ydinprosessien kypsyystason täytyy olla tarpeeksi korkealla, jotta niiden muodostamalle perustalle voidaan rakentaa uusia toimintoja. Parhaalla tahdollakaan prosessien parantaminen ei välttämättä aina onnistu. Epäonnistumisten syiden kirjo voi vaihdella laidasta laitaan; tuen ja ymmärryksen puutteesta aina jatkuviin muutoshankkeisiin väsymiseen. Organisaation kaikkien tasojen onkin oltava motivoituneita ja uusien prosessien käyttöönotto on hoidettava samanlaisin toimenpitein kuin minkä tahansa muun hankkeen toteuttaminen. Projektille on laadittava budjetti ja aikataulu, sille on allokoitava resursseja ja se on vietävä läpi kontrolloidusti ja tehokkaasti. (Dekkers & Forselius, 2007.) Seuraavassa luvussa pyritäänkin selvittämään, olisiko laaditun mallin käyttöönotto Yrityksessä mahdollista ja mitä ongelmia siinä vaiheessa olisi odotettavissa.

6.6 Mallin käyttöönoton haasteet

Dekkersin ja Forseliuksen (2007) artikkelissa esitellään Willmanin (1996) laatimat edellytykset minkä tahansa organisatorisen muutoshankkeen onnistumiselle ja myös lopputulos, jos jokin edellytyksistä puuttuu (taulukko 5). Taulukossa on plus-merkki, jos edellytys on olemassa ja tyhjä, jos edellytys puuttuu. Äärimmäisenä oikealla on tulos, joka hankkeesta on edellytysten perusteella odotettavissa. Esimerkiksi, jos johtajuus ja visio puuttuvat, hanke saattaa kaatua henkilöstön vastustukseen.

TAULUKKO 5 Muutoshankkeen onnistumisen tekijät (P.Willman, 1996)

Muutos-paine	Johtajuus ja visio	Kyvykäs henkilöstö	Toimivat ensiaskeleet	Tehokkaat kannustimet	Tulos
+	+	+	+	+	**Onnistunut toteutus**
	+	+	+	+	**Välinpitämättömyys**
+	+	+	+		**Kuihtuminen**
+	+	+		+	**Turhautuminen**
+	+		+	+	**Sitoutumattomuus**
+		+	+	+	**Vastustaminen**

Tämän opinnäytetyön yhtenä tarkoituksena oli myös selvittää, olisiko laajuudenhallinnan ja tuotehallinnan käytänteiden omaksuminen pienessä tuotekehi-

tystä tekevässä yrityksessä mahdollista ja nähtäisiinkö mallista saatavat hyödyt niin suuriksi, että prosessien muutoshankkeeseen kannattaa ryhtyä. Selvitystä varten Yrityksessä haastateltiin neljää henkilöä ja heidät valittiin siten, että kullekin tuotehallinnan prosessialueelle määriteltiin ”omistaja”. Tuotesalkun omistaja on Yrityksen toimitusjohtaja ja roadmappingin tuotekehitysjohtaja. Vaatimusten hallinnan omistaa tuotepäällikkö ja julkaisujen suunnittelun omistaja on projektipäällikkö. Haastatteluissa ei ollut käytössä etukäteen laadittua kysymyssarjaa, vaan asiaa käsiteltiin keskustellen. Jokaisella haastateltavalla oli jo aikaisemmin hankittua perustietoa laajuudenhallinnasta ja toimintopistelaskennasta, joten nähtiin riittäväksi keskustelun aluksi vain kerrata lyhyesti käsitteiden pääpiirteet. Sen jälkeen esiteltiin konseptin yhdistäminen tuotehallinnan prosesseihin tässä työssä laadittujen taulukoiden ja kuvioiden avulla. Keskustelussa haastateltavilta kysyttiin, pystyvätkö he näkemään laaditun mallin (taulukko 4 ja kuvio 12) käyttöönotosta saatavan hyötyä omassa työssään ja yleisesti tuotekehityksessä. Haastateltavia pyydettiin arvioimaan, ovatko edellytykset hankkeen onnistumiselle Yrityksessä heidän mielestään jo olemassa tai laitettavissa kuntoon. Heitä pyydettiin myös valitsemaan taulukosta 5 mielestään pahimmat esteet hankkeen onnistumisen tiellä ja myös perustelemaan mielipiteensä. Jokaisen henkilökohtaista motivaatiota muutokseen tiedusteltiin myös ja esimiesasemassa olevilta (toimitusjohtaja ja tuotekehityspäällikkö) kysyttiin tahtoa edesauttaa omien alaistensa osallistumista muutoshankkeeseen.

Jokainen haastateltava suhtautui myönteisesti ajatukseen, että tuotekehitysprojekteja ryhdyttäisiin hoitamaan esitetyn mallin mukaisesti. Mallin käyttöönotto olisi perusteltua jo siksikin, että Yrityksessä ei ole tällä hetkellä käytössä mitään formaalia projektinhallintamenetelmää. Muutosvastarintaa ei siis pitäisi ilmetä ja poisoppiminen vanhasta menetelmästä ei aiheuta huolta. Yrityksen valmiudet mallin käyttöönottoon kaikki näkivät hyvinä, mutta suurimmat pohdinnat aiheutti kysymys hankkeelle annettavissa olevasta ajasta. Käyttöönotto pitäisi sijoittaa hetkeen, jolloin ei ole menossa kriittisessä vaiheessa olevia muita projekteja ja sille pitäisi varata budjetissa aikaa ja varoja. Tuotepäällikkö ja projektipäällikkö etenkin pelkäsivät sitä, että he joutuisivat osallistumaan muutosprojektiin tilanteessa, jossa varsinaiset tuotekehitysprojektit vaativat kaiken heidän liikenevän aikansa. Esimiehet kuitenkin ymmärsivät, että muutosprojektia on alusta alkaen syytä käsitellä samoilla kriteereillä kuin varsinaisia tuotekehitysprojektejakin ja sille on allokoitava riittävästi tilaa valittujen henkilöresurssien kalenteriin. Tuotepäällikkö esitti myös toiveen, että toimintopistelaskenta otettaisiin käyttöön ensimmäisen kerran jossakin pienemmän kokoluokan projektissa, jonka hallittavuus olisi parempi ja näkyviä tuloksia saataisiin nopeammin. Missään tapauksessa ei haluttu, että Yrityksessä edes yritettäisiin siirtyä kertarysäyksellä uuteen projektien hallintamalliin, vaan vähittäinen, portaittainen siirtyminen nähtiin parempana lähestymistapana.

Toimitusjohtajan mielestä tärkein edellytys muutoshankkeen onnistumiselle on toimivien ja tehokkaiden ensiaskelten ottaminen. Näin projekti saadaan kunnolla vauhtiin ja kaikille osallistujille tulee heti selväksi, että johto on sitoutunut hankkeeseen. Tuotekehitysjohtaja taas näki, että projektin kestäessä ja

mahdollisiin vastoinkäymisiin törmättäessä vaaditaan johtajuutta. Hankkeen vision on oltava vahva ja se on jo alusta alkaen viestittävä osallistujille selkeästi. Näin saavutetaan se, että myös osallistujat sitoutuvat hankkeeseen ja ovat motivoituneita kohtaamaan muutoksen. Osallistujien omat voimavarat uuden oppimiseen saadaan käyttöön ja heidät saadaan myös ponnistelemaan hankkeen eteen. Ensimmäinen vastoinkäyminen tai alussa ilmenevä osaamattomuus ei heti näy lannistumisena ja luovuttamisena, vaan haasteet otettaisiin innostuneina vastaan. Tuotepäällikkö ja projektipäällikkö valitsivat myös tärkeimmiksi edellytyksiksi johtajuuden ja vision sekä toimivat ensiaskeleet. Heille oli tärkeää, että johto seisoo hankkeen takana ja antaa heille konkreettisesti resursseja projektimallin käynnistämiseen. Tarkoin valittu ensimmäinen projekti takaisi varmemmin onnistumisen ja mallin hyödyntäminen saataisiin paremmin juurrutettua yrityksen normaaleihin käytäntöihin.

Suurin hyöty mallin käytöstä saataisiin projektien ennustettavuuden ja seurattavuuden huomattavana parantumisena. Eniten arvostettiin sitä, että työmääräarvioiden laatiminen ei olisi enää arvauksen varassa, vaan mallia käytettäessä arviot paranisivat projektien seurantatietojen lisääntyessä. Erityisesti toimitusjohtaja ja tuotekehitysjohtaja pystyivät näkemään projektien ennustamisen hyödyt esim. seuraavan vuoden budjettia laadittaessa. Ovathan projekteille lasketut karkeatkin toimintopisteet muunnettavissa suoraviivaisesti kustannuksiksi ja henkilötyöpäiviksi. Tuotepäällikkö ja projektipäällikkö taas arvostivat enemmän projektien keston aikana saatavissa olevaa seurantatietoa. Kerran kuussa pidettävissä projektikokouksissa tiedettäisiin tarkalleen, missä vaiheessa tuotteen valmistuminen on ja kuinka paljon työtä on vielä jäljellä. Tavoiteltavana hyötynä nähtiin myös vaatimusmäärittelyyn projektin kestäessä tulevien muutosten aiheuttaman työmäärän lisäyksen parempi seurattavuus. Mielenkiintoisena, mutta ei kuitenkaan kaikkein tärkeimpänä, nähtiin myös ajatus siitä, että mallin avulla voitaisiin laatia ja ottaa käyttöön projektien onnistumista kuvaavat mittarit ja tavoitteet ja sen jälkeen myös niiden saavuttamisesta maksettavat kannustinpalkkiot.

Projektipäällikkö olisi valmis itsekin laskemaan toimintopisteitä ja käyttämään Experience® Service –ohjelmaa, mutta yleisesti pidettiin parempana sitä, että Yritykseen nimitettäisiin Scope Manager, jonka vastuulla olisi kaikkien meneillään olevien tuotekehitysprojektien laskenta ja seuranta. Näin toimien kaikkien projektien toimintopisteet olisivat varmasti yhteismitallisia, vaikkakaan eri laskijoiden arviot tuskin poikkeaisivat kovin paljon toisistaan.

7 POHDINTA

7.1 Vastaus tutkimuskysymykseen

Tutkielman pääasiallinen tutkimusongelma oli: ”Miltä osin tuotehallinnan ja laajuudenhallinnan käytänteet ovat omaksuttavissa pienessä ohjelmistoyrityksessä?” ja lisäkysymyksenä haluttiin tutkia ”Ovatko tuotehallinnan ja laajuudenhallinnan konseptit yhdistettävissä miltään osin?”.

Yrityksen tuotehallinnassa on oikeastaan ollut jo käytössä tuotehallinnan eri prosessialueet, mutta jäsentymättöminä. Tuotehallinnan toimintoja eli tuotesalkkua, roadmappingia, vaatimusten hallintaa ja julkaisujen suunnittelua on tehty osana Yrityksen jokapäiväisiä toimintoja, mutta niiden hallinnasta on osin puuttunut järjestelmällisyys. Tilanne on kuitenkin muuttumassa parempaan suuntaan, sillä tuotesalkun hallinnan ja roadmappingin tarpeellisuus on Yrityksessä jo huomattu ja toimenpiteitä niiden parempaan haltuun ottoon on tehty. Vaatimusten määrittely ja julkaisujen suunnittelu on ollut paremmalla tolalla, mutta niiden hallinnan terävöittäminen on tarpeen. Uudet käyttöön otetut työkalut tulevat auttamaan tässä prosessissa. Laajuudenhallinta ei ole ollut Yrityksessä lainkaan käytössä, mutta se soveltuisi mainiosti antamaan tukea kaikille tuotehallinnan tasoille. Työmäärien tarkemmasta arvioinnista olisi suurta hyötyä, sillä niiden avulla tuotteiden laajuus ja muutokset olisivat kontrollissa kehitysprojektin eri vaiheissa.

Tuotesalkun kokoaminen, sen määrittely ja jatkuva ylläpito on Yritykselle erittäin tärkeää. Monen nykyisen ohjelmiston osan elinkaari on päättymässä ja nyt hyvä hetki tarkastella ohjelmiston kokonaisuutta ja karsia vuosien varrella kertyneet rönsyt pois. Yrityksen täytyy miettiä, mitä ohjelmistoja sen kannattaa valmistaa itse ja mitkä osat voidaan korvata joko markkinoilta jo löytyvillä valmisohjelmistoilla tai teettämällä alihankintana. Yrityksen vahvuus on vuosien aikana kertynyt rakennusalan ohjelmisto-osaaminen, jonka lisäksi myös henkilökunnan osaaminen on erittäin tarkasti kohdentunut rakennusalalle. Omien tuotekehityspanosten valjastaminen juuri rakennusalalle soveltuvien ohjelmistojen kehittämiseen tuottaa varmasti parhaan tuloksen. Markkinoilta

löytyy valmiita, myös rakennusalalle konfiguroitavissa olevia, taloushallinnon ohjelmistoja, jotka voidaan liittää älykkäiden rajapintojen avulla omien tuotteiden saumattomiksi osiksi. Soveliaiden kumppanien löytäminen vaatii tietenkin tarkkaa harkintaa, mutta sellaisten löytyessä on käsillä tilanne, josta molemmat osapuolet hyötyvät. Myös ohjelmiston osien teettäminen ulkopuolisilla alihankkijoilla on tutkittava perusteellisesti ja tähän suuntaan yrityksessä onkin jo otettu varovaisia askelia. Mobiilialustoille rakennettavien käyttöliittymien ja applikaatioiden ostaminen ulkopuolelta on tuntunut sopivalta tavalta aloittaa yhteistyö sellaisten alihankkijoiden kanssa, joilla on jo valmiiksi hankittua osaamista tältä alueelta. Sidosryhmien määrän lisääntyessä on kumppaniverkoston ylläpitämiseen panostettava. Erilaisten sopimusten laatiminen ja niiden noudattamisen seuraaminen on työ, jossa toimintopistelaskennasta tulisi olemaan hyötyä. Juuri tämäntapaista ohjelmistojen hankkimista helpottamaan northernSCOPE™-konsepti on alun perin laadittu. Omassa tuotekehityksessä ohjelmiston osien laajuuden laskeminen edes karkealla tasolla ennen projektien aloittamista antaisi varmuutta siihen, että nyt ei olla käynnistämässä työtä, johon omat resurssit ja aikataulu eivät riitä.

Kun kehitettävät tuotteet ja niiden toteutustapa on valittu, voidaan laatia pitkän aikavälin roadmap ja sijoittaa kehitysprojektit aikajanalle. Roadmapissa voi olla menossa useita projekteja yhtä aikaa; omaa tuotekehitystä, alihankintaa ja valmiiden ohjelmistojen integrointia. Oikeastaan vain tuotepäälliköiden kyky tehdä määrittelyjä kaikille näille kehityspoluille asettaa rajat yhtäaikaisten hankkeiden eteenpäinviemiselle. Aikataulujen asettamisessa ollaan usein liian optimistisia, mutta toimintopistelaskennan kautta saatu kokemusperäinen tieto antaisi tähän työkaluja. Saman tuotekehitystiimin on huolehdittava myös olemassa olevien ohjelmistojen ylläpidosta ja autettava Service Deskiä asiakasongelmien ratkaisemisessa. Näille toimille on siis varattava roadmapissa tilaa ja aikaa.

Tuotepäällikön tärkein tehtävä on hallita sidosryhmiltä tulevia vaatimuksia. Nykyisiin tuotteisiin ehdotetut muutosvaatimukset ovat nykyisessä murrosvaiheessa hyvin tarkan punninnan alla. Yrityksen resurssit halutaan kohdentaa lähes täysimääräisesti uusien tuotteiden valmistukseen ja ylläpitotehtäviä otetaan vastaan mahdollisimman vähän. Tasapaino uustuotannon ja ylläpidon välillä on kuitenkin löydettävä, sillä nykyiset tuotteet on pidettävä toimintakuntoisina laajan asiakaskunnan tarpeita ajatellen. Uusien tuotteiden määrittelyt on tehtävä siten, että niiden avulla on mahdollista jakaa tehtävät ohjelmistokehitykselle ja että niistä on myöhemmin apua testauksessa. Samoista määrittelyistä tulisi olla mahdollista laskea myös toimintopisteet projektin etenemisen seurantaa ja loppulaskentaa varten. Kun uutta tuotetta ryhdytään valmistamaan, tuotepäällikön tehtävänä on vaatimusmäärittelyihin kirjoitettujen käyttötapausten perusteella jakaa käsillä oleva ohjelman osa ohjelmoijille sopiviksi tehtäviksi. Arkkitehdit suunnittelevat tietokannan taulut ja liittymärajapinnat muihin ohjelmiston osiin. Tuotteen vähitellen valmistuessa testaajat voivat ryhtyä työhön ja tuotepäällikkö voi aloittaa julkaisun suunnittelun. Määrittelyihin

työn kuluessa tulevat muutokset tulee huomioida laajuuslaskennassa ja valmistumisaikataulussa.

Tehtyjen haastattelujen perusteella Yrityksessä oltaisiin valmiita kokeilemaan laajuudenhallinnan yhdistämistä tuotehallinnan osa-alueisiin antamaan lisätietoa tulevista tai meneillään olevista kehitysprojekteista. Yrityksen johdon ja esimiesten tuki on välttämätöntä, että prosessien muutostyö saadaan käyntiin. Työntekijöille on annettava aikaa tehdä kokeiluja ja löytää näin sopivin tapa edetä. Projektien seurattavuus ja ennustettavuus tuo varmasti opetteluun käytetyn ajan myöhemmin moninkertaisesti takaisin.

NorthernSCOPE™-konsepti ei sellaisenaan sovellu pienen yrityksen tuotekehitysprojektien hallintaan, mutta konseptin periaatteet ja menetelmät ovat myös sen projekteille käyttökelpoisia. Scope Managerin nimittäminen ja laajuudenhallinnan valjastaminen tuotteiden arviointiin jo varhaisessa vaiheessa vahvistavat työmääräarvioita. Varsinkin, jos arvioita ei ole aiemmin ole tehty lainkaan tai ne ovat perustuneet lähes puhtaaseen arvaukseen. Projektin baselinen kiinnittäminen ja muutostenhallinta projektin aikana antaisivat tarvittavaa ryhtiä projektin seurantaan. Projektin jälkeinen kokemusdatan kokoaminen ja tallentaminen auttaisivat yritystä seuraavien projektien suunnittelussa ja samalla voitaisiin kriittisesti tarkastella päättyneen projektin onnistumista. Saatujen mittaustulosten avulla on myös mahdollista rakentaa yritykselle sopiva mittaristo esimerkiksi tulospalkkioiden laskemiseksi. Yrityksen johto ja henkilöstö tunnustavat uuden projektimallin hyödyllisyyden, mutta sen käyttöönottoon on annettava tarpeeksi resursseja ja johdon on tuettava hanketta varauksetta. Scope Managerin tarpeellisuutta on syytä korostaa, sillä hän pystyisi kokoamaan ja esittämään tarpeellisen projektin laajuustiedon sekä ennen projektia, sen kestäessä että projektin valmistuttua. Pienessä yrityksessä Scope Manager pystyy hoitamaan myös muita asiantuntijatehtäviä, sillä laajuuksien laskeminen ei vaadi häneltä kokoaikaista työpanosta.

7.2 Luotettavuuden arviointi

Laadullisessa tutkimuksessa on olennaista arvioida tutkimuksen uskottavuutta ja luotettavuutta. Laadullisen tutkimuksen tulokset eivät esimerkiksi saa olla sattumanvaraisia ja tutkimuksessa käytetyillä menetelmillä on voitava tutkia sitä, mitä tutkimuksessa on tarkoitus tutkia. (Lähdesmäki ym., 2016.) Tässä tutkimuksessa tuloksena on yhdelle yritykselle laadittu tuotekehitysmalli, joka on johdettu kahdesta erilaisesta konseptista yhdistämällä. Suurin piirtein samaan tulokseen päätyisi luultavasti toinenkin tutkija samoista lähtökohdista. Havainnointi tutkimusmenetelmänä ja tietty yritys tutkimuskohteena aiheuttaa tapaustutkimuksessa sen, etteivät tulokset kahdessa eri tapauksessa voi olla täsmälleen samanlaiset. Tutkimuksen toistettavuus on siis hieman kyseenalaista.

Luotettavuutta voidaan arvioida laadullisessa tutkimuksessa monin tavoin. Esimerkiksi tutkimuksessa käytettyjen käsitteiden on sovittava tutkimusongelman ja aineiston sisältöihin. Samoin eräs laadullisen tutkimuksen luotet-

tavuuteen liittyvä näkökulma on yleistettävyys tai siirrettävyys: ovatko tutkimuksen tulokset yleistettävissä tai siirrettävissä myös muihin kohteisiin tai tilanteisiin. (Lähdesmäki ym., 2016.) Tässä tutkimuksessa käytetyt käsitteet on pyritty selvittämään kirjallisuuden avulla ja tärkeimmät käsitteet on aina kursivoitu. Tutkimus itsessään ei ole – tapaustutkimuksen luonteesta johtuen - sellaisenaan yleistettävissä, mutta tulos saattaa olla hyödynnettävissä samantapaisessa tilanteessa. Samanlainen tutkimus voitaisiin silti toistaa jossain toisessa tuotekehitystä tekevässä ohjelmistotalossa. Tutkimukseen valitut haastateltavat edustivat työtehtäviensä perusteella yksi kutakin prosessimallin osa-aluetta, eikä heitä ollut syytä valita enempää, sillä pienessä yrityksessä he edustivat kattavasti tuotekehitysprosessiin osallistuvia. Tehtyjen haastattelujen keskusteleva tapa aiheuttaa sen, ettei haastatteluja ole mahdollista toistaa uudestaan. Haastattelija on myös saattanut tahtomattaan johdatella haastateltavia vastaamaan tietyllä tavalla osallistuessaan keskusteluun. Tämä ei ole kuitenkaan ollut tarkoitus, mutta jo se, että tutkija ja haasteltavat ovat olleet vuosikausia samassa työpaikassa ja käsitelleet samoja tuotekehityksen ongelmia työssään, aiheuttaa sen, että suurempaa kritiikkiä ei haastateltavien taholta ilmennyt.

8 YHTEENVETO

Tutkimuksessa tarkasteltiin pienen, tuotekehitystä tekevän ohjelmistoyrityksen tarpeita ja ongelmia omien tuotteidensa hallinnan alueella. Pienen yrityksen tarpeet ja toiveet tuotehallinnan suhteen ovat paljolti samanlaiset kuin isommankin yrityksen. Molemmat pyrkivät saamaan oikeat tuotteet oikeille markkinoille oikeaan aikaan ja samalla ne pyrkivät toteuttamaan omaa liiketoimintastrategiaansa ja saavuttamaan taloudellista hyötyä. Isolla yrityksellä on käytettävissään laajat resurssit ja enemmän pääomia ja mahdollisuuksia pyrkimystensä tueksi. Pienen yrityksen on kamppailtava arkipäiväisten ongelmien kanssa ja käytettävä vähät resurssinsa tuottavaan työhön eikä siltä liikene aikaa ja rahaa prosessien parantamiseen. Isoille yrityksille tarkoitetut prosessimallit kypsyystasoineen eivät sovellu pienille yrityksille, vaan ne tarvitsevat paljon kevyemmät ja nopeasti haltuun otettavat mallit oman toimintansa tueksi.

Ohjelmistotuotteiden valmistus eroaa asiakaskohtaisten, räätälöityjen ohjelmistojen valmistamisesta monessa suhteessa. Tuotteella ei ole yksittäistä asiakasta, joka osaisi kertoa haluamansa vaatimukset yksityiskohtaisesti. Sen sijaan tuotteen ympärillä vaikuttaa lukuisa sidosryhmien joukko, jonka toiveet ja vaatimukset on otettava huomioon. Tuotteesta vastaava tuotepäällikkö joutuu luovimaan monenlaisten paineiden ristiaallokossa; yrityksen johto, asiakkaat, myynti ja tuotekehitys ovat kukin tahoillaan esittämässä ajatuksiaan ja pyytämässä häneltä palveluksia. Tuotetta kehitetään ennalta sovituissa julkaisuissa, joiden aikataulu on otettava huomioon kaikilla tasoilla. Jos aikataulussa pysyminen näyttää epävarmalta, on julkaisuun valittuja vaatimuksia vähennettävä. Pienessä yrityksessä ei ole mahdollista siirtää vähäisiä resursseja muista projekteista julkaisuprojektiin, joten se tinkii aikataulupaineiden alla jopa ohjelmiston laadusta testausta vähentämällä. Pitemmän päälle tämä ei ole kestävä ratkaisu, sillä virheettömästi toimiva ohjelma ja siitä seuraava asiakastyytyväisyys on kuitenkin kaiken liiketoiminnan perusta.

Viitekehysten on todettu auttavan monien ongelmien selville saamisessa ja parantamisessa. Työssä esiteltiin hollantilaisten tutkijoiden (van de Weerd ym. 2006) laatima tuotehallinnan viitekehys, jota sitten purettiin auki taso tasolta. Viitekehyksen prosessit soveltuvat sellaisenaan tai vähäisin muutoksin

myös pienelle tuotekehitysyritykselle. Viitekehyksen soveltamisesta tulee käytännössä pienelle yritykselle koko liiketoimintaa, eikä vain tuotekehitystä, ohjaava malli. Pienen yrityksen kaikki toiminnot nivoutuvat yhteen niin tiiviisti, että eri prosessitasojen erottaminen toisistaan on jopa mahdotonta. Tuotesalkun hallinnassa tehdyt päätökset vaikuttavat suoraan alimman tason resurssiallokointiin ja myös toisinpäin; tuotekehityksessä tehdyt yhtä tuotetta koskevat nopeat päätökset vaikuttavat usein muihin salkussa oleviin tuotteisiin. Prosessimallin käyttäminen on joka tapauksessa hyödyllistä, sillä sen avulla saadaan ryhtiä ja ennustettavuutta tuotehallintaan. Myös lukuisat sidosryhmät tulevat huomioon otetuiksi ja saavat äänensä kuuluviin.

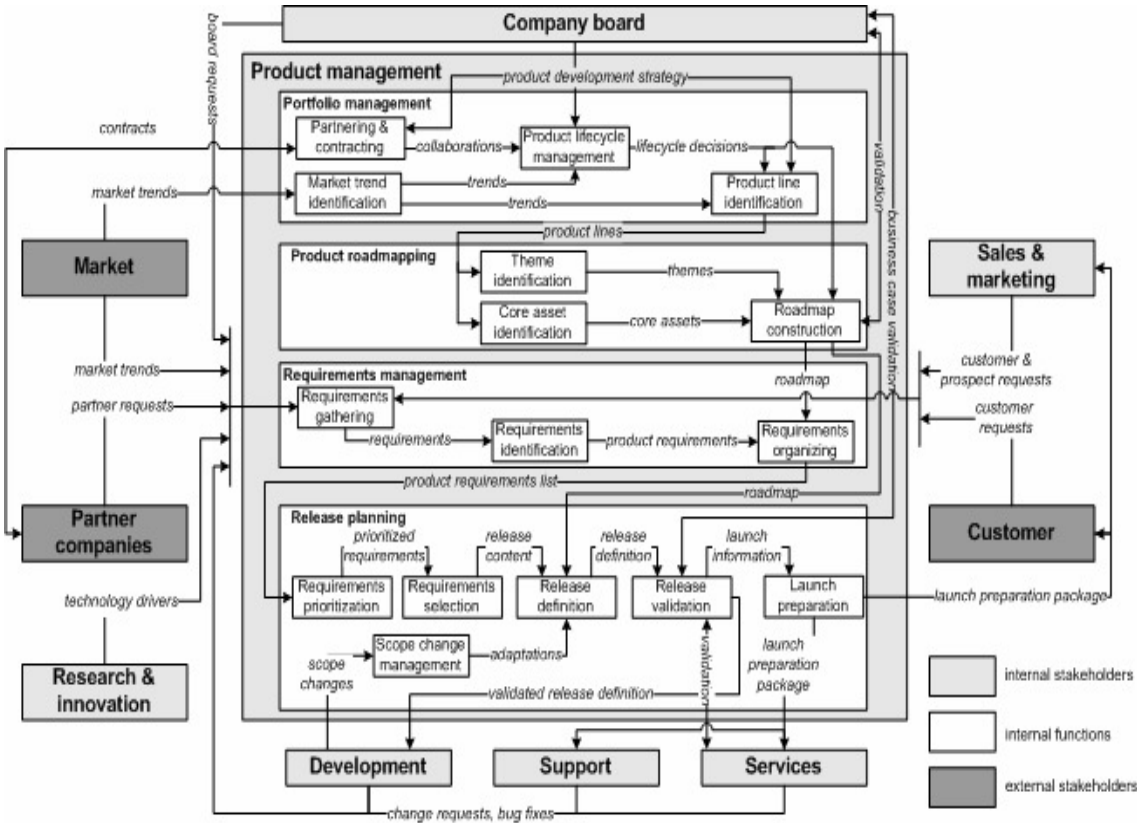
Vaatimusten siirtyessä tuotekehityksen prosessiin pitäisi niiden kehittämiseen tarvittavat työmäärät pystyä arvioimaan melko tarkasti. Yrityksissä, suurissa tai pienissä, käytetään harvoin mitään formaalia menetelmää työmäärien arviointiin. Karkeita työmääräarvioita pitäisi pystyä tekemään jo aikaisessa vaiheessa osana vaatimusten hallintaa ja tässä ohjelmistoprojektien laajuudenhallinta voi tulla avuksi. Toimintopisteisiin perustuva laajuudenhallinta on ohjelmistojen koon arviointiin ja estimointiin kehitetty menetelmä, jolla on mahdollista määrittää rakennettavan ohjelmiston koko ja työmäärä etukäteen. Pelkkä toimintopisteiden laskeminen ei kuitenkaan riitä ohjaamaan kokonaista ohjelmistoprojektia. Lisäksi tarvitaan tarkkaa seurantaa ja varsinkin muutostenhallinta on hoidettava huolellisesti. Räätälöityjen ohjelmistokehitysprojektien hallintaa varten on Suomessa kehitetty northernScope™-konsepti, jonka avulla ohjelmistoprojektit on saatu pysymään aikataulussa sekä tilaajan että asiakkaan tyydytykseksi. Konseptiin olennaisena osana kuuluvan puolueettoman Scope Managerin tehtävänä on varmistaa, että projektin alkuvaiheessa yhdessä sovitut toiminnallisuudet sisältyvät valmiiseen tuotteeseen ja että asiakas vaatii vain sitä, mitä on tilannut. Kummankin osapuolen etuna on, ettei missään vaiheessa synny epäselvyyttä siitä, mitä alun perin oli tilattu tai mitä muutoksia projektin kuluessa on sovittu tehtäväksi.

Työssä tutkittiin, onko saman konseptin soveltaminen mahdollista myös pienen ohjelmistotuotteita valmistavan yrityksen toiminnassa ja voidaanko konseptin askelia sovittaa tuotehallinnan prosesseihin. Konseptin ensimmäiset kuusi askelta, joissa ohjelmistokehityshanketta valmistellaan, voidaan tuotteiden kohdalla kohdistaa tuotesalkun ja roadmapin hallintaan. Kun tuotesalkun sisältö on karkeasti mitattu toimintopisteinä ja muutettu työmääräarvioiksi, voidaan roadmapia rakennettaessa tehdä perustaltaan konkreettisempia päätöksiä. Yritysjohdolla on tällöin paljon paremmat valmiudet tehdä kauaskantoisia suunnitelmia ja sijoittaa myös muut tarvittavat projektit tuotekehitysprojektien lomaan. Yritys tarvitsee Scope Managerin laskemaan ja ylläpitämään tuotesalkun tuotteiden alustavia toimintopisteitä ja kun kehitysprojekti alkaa, määrittämään tarkemman toiminnallisuuden yhdessä tuotepäällikön kanssa. Konseptin viimeiset kuusi askelta liittyvätkin varsinaisen tuotekehitysprojektin hallintaan ja sen edistyessä Scope Manager laskee työn valmiusasteen, hallinnoi muutosvaatimuksia sekä pitää projektiryhmän ja johdon ajan tasalla tilanteesta.

Pienenkin yrityksen on mahdollista, ja myös ehdottoman tarpeellista, hoitaa tuotekehityksensä prosessit hallitusti ja suunnitelmallisesti. Isoille yrityksille suunnitellut prosessit voidaan soveltaa pienen yrityksen käyttöön ottaen huomioon käytettävissä olevat resurssit ja tarpeet. Liian raskaat prosessit kuluttavat turhaan vähiä resursseja ja jäävät lopulta noudattamatta, kun ne koetaan mieluummin työnteon esteiksi kuin avuksi. Tutkimuksessa esitelty tuotehallinnan viitekehys soveltuu myös pienelle yritykselle, mutta sen sisältämiä prosesseja voidaan karsia tai yhdistää yrityksen tarpeiden mukaan. Kuviossa 1 esitetty viitekehyksen rakenne sinänsä on erittäin sopiva sidosryhmineen ja tuotehallinnan tasoineen, sillä nämä ovat pienelläkin yrityksellä aivan samat kuin suurella. Pienen yrityksen kohdalla hallinnan tasoja ei vain ole helppoa – tai ehkä mielekästäkään – erottaa toisistaan. Pääasiassa tämä johtuu sitä, että yrityksen päätöksentekijät ja toimijat ovat samoja henkilöitä kaikilla tasoilla ja jollakin tasolla tehty päätös heijastuu saman tien muille tasoille. Tämä ei suinkaan tarkoita sitä, että asioita tehdään suunnittelemattomasti. Pieni yritys vain pystyy olemaan ketterä ja joustava myös päätöksenteossaan. Kun tuotesalkun sisältö on arvioitu ja selvitetty, voidaan roadmapping-työssä ottaa kaikki sidosryhmät huomioon mietittäessä tuotesalkun tulevaisuutta. Vaatimusten hallinta ja julkaisujen suunnittelu taas lankeaa pienessä yrityksessä melkein yksinomaan tuotepäällikön harteille. Hänen on huolehdittava vaatimusten keräämisestä, priorisoinnista ja valinnasta. Määrittelytyö on hänen vastuullaan, samoin kuin projektin ohjaus ja valmiin tuotteen ensivaiheen testaus. Tuotepäälliköllä on toki apunaan arkkitehti ja ohjelmoijat ja toivottavasti myös Scope Manager, joiden kanssa tehtävän yhteistyön on oltava saumatonta. Prosessit, vaikka kevyetkin, ohjaavat tätä yhteistyötä ja projektien valmistumisen seuranta antaa selkeän tavan raportoida edistymisestä johdolle ja sidosryhmille. Ei voida siis liikaa korostaa yhteisen tekemisen merkitystä tuotekehitysprojektien menestyksekkäässä läpiviennissä. Tutkimuksen kohteena olleessa yrityksessä kaikki osapuolet olivatkin valmiita ja innokkaita yrittämään uuden prosessimallin käyttöönottoa ja näkivät kaikkien tuotehallinnan tasojen mahdollisuudet hyödyntää toimintopistelaskentaa tuotekehitysprojekteissa. Tärkeimpinä onnistumisen edellytyksinä nähtiin johdon tuki ja tarvittavien resurssien varaaminen muutoshankkeen käyttöön.

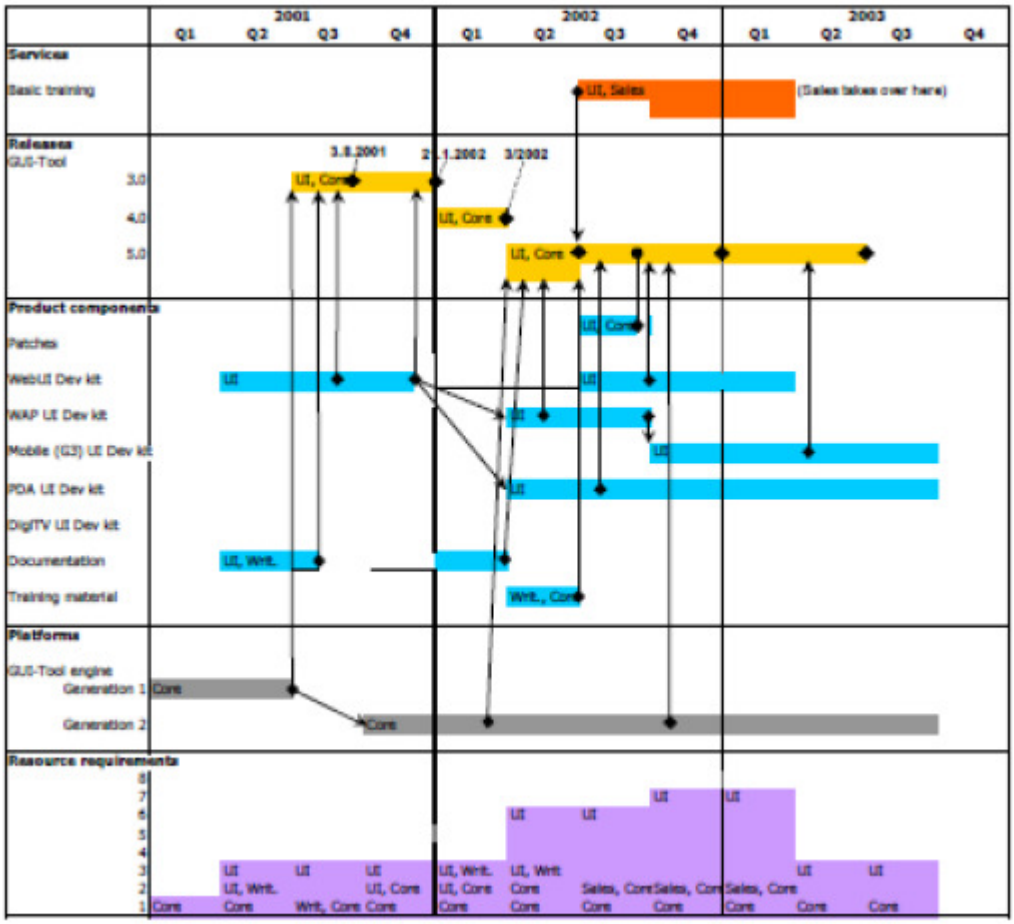
Tämän tutkimuksen aikana ei tutkittavassa yrityksessä ole vielä otettu laajuudenhallintaa jokapäiväiseen käyttöön, eikä siis ole saatavilla tuloksia sen tuomasta hyödystä. Jatkotutkimusta olisi syytä tehdä toimintopistelaskennan käytöstä ja saavutettavista eduista tuotekehitysprojekteja toteutettaessa. On luultavaa, että tuotekehitysprojektien työmäärien ennustaminen, seuranta ja muutostoiveiden hallittu käsittely parantavat projektien tuloksia ja syntyvien tuotteiden laatua.

LIITE 1 TUOTEHALLINNAN VIITEKEHYS



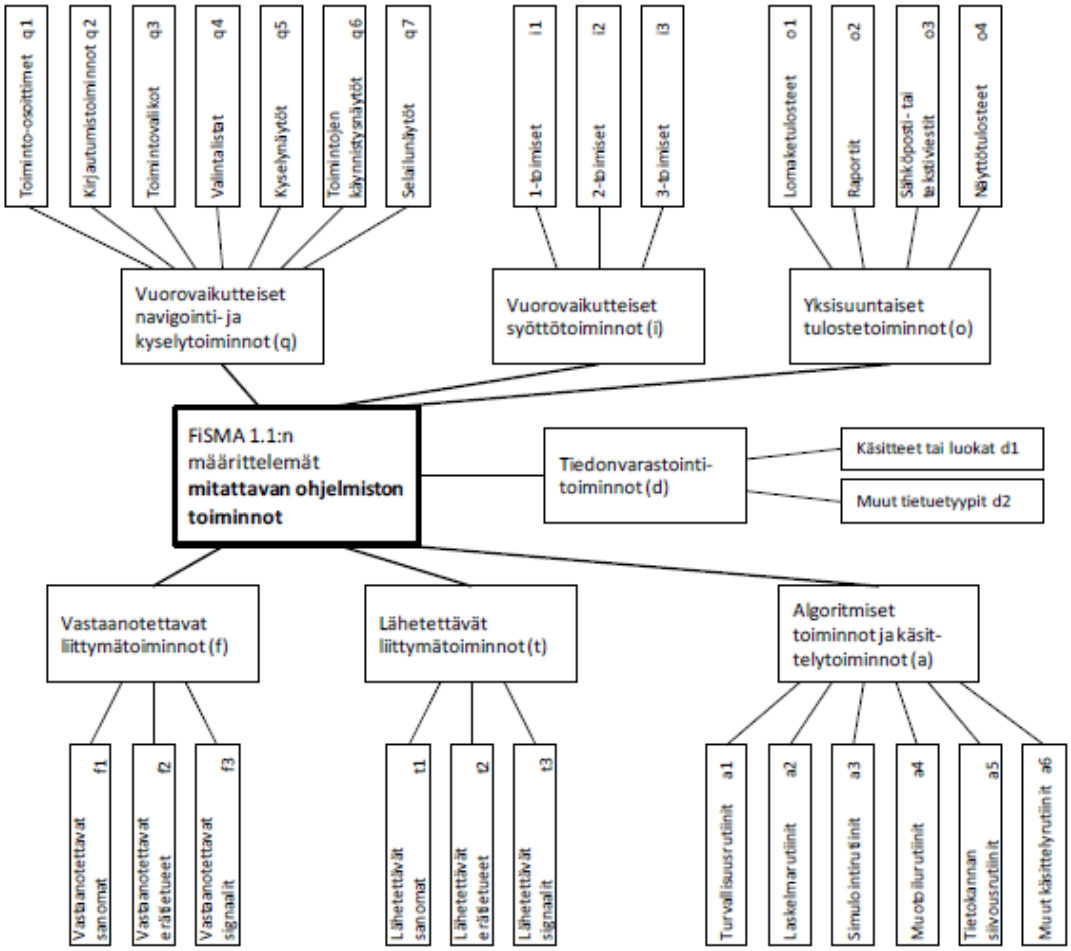
Van de Weerd ym. (2006)

LIITE 2 ESIMERKKI ERÄÄN TUOTTEEN ROADMAPISTA



Vähäniitty, Lassenius ja Rautiainen (2002)

LIITE 3 FISMA 1.1, TIETOJÄRJESTELMÄN TOIMINNALLISTEN VAATIMUSTEN TOIMINTOLUOKAT JA -TYYPIT



SFS-ISO/IEC (2013)

LIITE 4 FISMA 1.1, ND21-TUOTTAVUUSTEKIJÖIDEN LASKENTAKAAVIO

PROJECT SITUATION ANALYSIS Date: _____

Experience ND21 method

Project id and name: _____ _____

Author(s): _____

PRODUCTIVITY FACTORS

	--	-	+/-	+	++	Notes:
PROJECT FACTORS						
Involvement of the customer						
Performance of the environment						
Availability of IT staff						
Number of stakeholders						
Pressure on schedule						
PROCESS FACTORS						
Impact of standards						
Impact of methods						
Impact of tools						
Level of change management						
Maturity of development process						
PRODUCT FACTORS						
Functionality requirements						
Reliability requirements						
Usability requirements						
Efficiency requirements						
Maintainability requirements						
Portability requirements						
PEOPLE FACTORS						
Analysis skills of staff						
Application knowledge of staff						
Tool skills of staff						
Experience of project management						
Team skills of the project team						
Numbers of responses by column						Total 21
	a	b	c	d	e	

Counting formula for the situation coefficient:

$T = (1,10)^a * (1,05)^b * (1,0)^c * (0,95)^d * (0,90)^e =$ _____

LÄHTEET

4SUM Partners Ltd. (2014). http://www.4sumpartners.com/, katsottu 14.12.2014.

Berry, M. (2002). Strategic planning in small high-tech companies, *Long Range Planning*, 31(3), 455-466.

Brown, J. S., & Duguid, P. (1999). Balancing act: how to capture knowledge without killing it, *Harvard business review*, 78(3), 73-80.

Brownsword, L., Oberndorf, T. & Sledge, C.A. (2000). Developing new processes for COTS-based systems, *Software*, 17(4), 48-55.

Bundschuh, M. & Dekkers, C. (2008). *The IT Measurement Compendium: Estimating and Benchmarking Success with Functional Size Measurement*. Berliini: Springer.

Carlshamre, P. & Regnell, B. (2000). Requirements Lifecycle Management and Release Planning in Market-Driven Requirements Engineering Processes, teoksessa *Proceedings of the 11th International Workshop on Database and Expert Systems Applications*, 961-965, Los Alamitos: IEEE.

Carmel, E. & Becker, S. (1995). A process model for packaged software development, *IEEE Transactions on Engineering Management*, 42(1), 50-61.

Cooper, R. G., Edgett, S. J & Kleinschmidt, E. J. (2002). Portfolio management: fundamental to new product success, teoksessa P. Belliveau, A. Griffin & S. Somermeyer (toim.), *The PDMA Toolbook for New Product Development*, 331-364, New York: John Wiley & Sons, Inc.

Cusumano, M.A. (2004). *The Business of Software*, New York: Free Press.

Dekkers, C. & Forselius, P. (2007). Increase ICT Project Success with Concrete Scope Management, teoksessa *Proceedings of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications*, 385-392, Los Alamitos: IEEE.

Dekkers, C. & Forselius, P. (2010). Scope Management: 12 Steps for ICT Program Recovery, *CrossTalk: the Journal of Defense Software Engineering*, 23(1), 16-21.

Dybå, T. (2003). Factors of software process improvement success in small and large organizations: an empirical study in the scandinavian context, teoksessa *Proceedings of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering*, 148-157, New York: ACM.

Ebert, C. (2007). The impacts of software product management, *Journal of Systems and Software*, 80(6), 850-861.

European Commission. (2005). The New SME Definition: User Guide and Model Declaration, haettu 17.11.2013 osoitteesta http://ec.europa.eu/enterprise/policies/sme/files/sme_definition/sme_user_guide_en.pdf

FiSMA. (2001). Experience ND21 Situation Analysis Method, haettu 3.11. 2013 osoitteesta http://www.fisma.fi/wpcontent/uploads/2006/09/fisma_situation_analysis_method_nd21.pdf.

FiSMA. (2002). FiSMA Reuse Measurement Method, FiSMA RMM version 2002, haettu 3.11.2013 osoitteesta http://www.fisma.fi/wpcontent/uploads/2006/09/fisma_reuse_analysis_method_10.pdf.

FiSMA 1.1. (2008). A Functional Size Measurement Method with continuous scale: Basic principles and practical examples, haettu 3.11.2013 osoitteesta http://www.fisma.fi/wp-content/uploads/2008/02/fisma-11-introduction-with-examples.pdf

Forselius, P. (2013). *Onnistunut tietojärjestelmän hankinta*. Helsinki: Talentum.

Forselius, P., Dekkers, C., Karvinen, M. & Kosonen, M. (2009). *Hankehallinnan työkalupakki*. Helsinki: Talentum Media.

Forselius, P. & Käkölä, T. (2009). An Information Systems Design Product Theory for Software Project Estimation and Measurement Systems, teoksessa *Proceedings of the 42nd Hawaii International Conference on System Sciences*, 1-10, Los Alamitos: IEEE

Forselius, P. & Savioja, E. (2013). Super Fast Size and Effort Estimation, haettu 19.11.2014 osoitteesta http://www.4sumpartners.com/wp-content/uploads/2014/11/SuperFastEstimationForseliusSavioja2013MetriKon.pdf

Habra, N., Alexandre, S., Desharnais, J., Laporte, C. Y. & Renault, A. (2008). Initiating software process improvement in very small enterprises:

Experience with a light assessment tool, *Information and Software Technology*, 50(7-8), 763-771.

Hill, P. (2011). *Practical Software Project Estimation: A Toolkit for Estimating Software Development and Duration*, New York: McGraw & Hill.

Hofer, C. (2002). Software development in Austria: results of an empirical study among small and very small enterprises, teoksessa *Proceedings of the 28th Euromicro Conference*, 361-366, IEEE

ISO (2004). *ISO/IEC 15504, Information technology – Process assessment.*

Jones, Capers (2013). Function Points as a Universal Software Metric, haettu 24.11.2014 osoitteesta http://namcookanalytics.com/wp-content/uploads/2013/07/Function-Points-as-a-Universal-Software-Metric2013.pdf

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

Kamsties, E., Hörmann, K. & Shlich, M. (1998). Requirements engineering in small and medium enterprises, *Requirements Engineering*, 3(2).

Kappel, T. (2001). Perspectives on Roadmaps: How Organizations Talk about the Future, *The Journal of Product Innovation Management*, 18(5), 39-50.

Karlsson, L., Dahlstedt, Å., Regnell, B., Natt och Dag, J. & Persson, A. (2007). Requirements engineering challenges in market-driven software development – An interview study with practitioners, *Information and Software Technology*, 49(6), 588-604.

Käkölä, T., & Forselius, P. (2015). Validating the Design Theory for Managing Project Scope during Software Sourcing and Delivery, teoksessa *Proceedings of Thirty Sixth International Conference on Information Systems, ICIS 2015*, Fort Worth: AISel.

Laporte, C.Y., Alexandre, S. & O'Connor, R. (2008). A Software Engineering Lifecycle Standard for Very Small Enterprises, teoksessa *R.V.O'Connor et al (toim), Proceedings of EuroSPI,* 16, 129-141, Heidelberg: Springer-Verlag.

Lehtola, L., Kauppinen, M. & Kujala, S., (2005) Linking the business view to requirements engineering: long-term product planning by roadmapping,, teoksessa *Proceedings of the 13th IEEE International Conference on Requirements Engineering*, 439-443, Los Alamitos: IEEE

Lähdesmäki, T., Hurme, P., Koskimaa, R., Mikkola, L. & Himberg, T., Menetelmäpolkuja humanisteille. Jyväskylän yliopisto, humanistinen tiedekunta. Katsottu 5.12.2016 osoitteesta http://www.jyu.fi/mehu

Maglyas, A., Nikula, U. & Smolander, K. (2012). What do practitioners mean when they talk about product management?, teoksessa *Requirements Engineering Conference (RE)*, 261-266, Los Alamitos: IEEE International

Parantainen, J. (2013). *Tuotepäällikön pelastuspakkaus*, Helsinki: Talentum Media Oy.

Phaal, R., Farruk, C., & Probert, D. (2003), Technology Roadmapping – A Planning Framework for Evolution and Revolution, *Technological Forecasting & Social Change*, 71(1-2), 5-26.

Pragmatic Marketing, (2010). The Annual Product Management and Marketing Survey, haettu 29.4.2014 osoitteesta http://www.pragmaticmarketing.com/publications/survey/2010

Project Management Institute, (2004). *A guide to Project Management Body of Knowledge (PMBOK® Guide)*, Newtown Square: PMI Books

Rautiainen, K., Lassenius, C., Vähäniitty, J., Vanhanen, J. & Pyhäjärvi, M. (2002). A Tentative Framework for Managing Software Product Development in Small Companies, teoksessa *Proceedings of 35th Hawaii International Conference on System Sciences*, 3409-3417, Los Alamitos: IEEE

Regnell, B. & Brinkkemper, S. (2005). Market-Driven Requirements Engineering for Software Products, teoksessa A. Aurum. & C. Wohlin (toim.), *Engineering and Managing Software Requirements*, 287-308, Berliini: Springer-Verlag

Richardson, I. (2002). SPI Models: What Characteristics Are Required for Small Software Development Companies?, teoksessa J. Kontio & R. Conradi (toim.), *Software Quality – ECSQ 2002, Lecture Notes in Computer Science,* vol. 2349, 100-113, Berliini: Springer-Verlag.

Saastamoinen, I. & Tukiainen, M. (2004). Software process improvement in small and medium sized enterprises in eastern Fnland: a state-of-the practice study, teoksessa T. Dingsoyr (toim.), *EuroSPI 2004*, *Lecture Notes in Computer Science*, 3281, 69–78, Berliini: Springer-Verlag.

Sawyer, P. (2000). Packaged Software: Challenges for RE, teoksessa *Proceedings of the sixth Int. workshop on Requirements Engineering: Foundation of Software Quality (REFSQ'00)*, 137-142, julkaisematon.

SEI, Software Engineering Institute, (2002). *CMMI, Capability Maturity Model Integration*, Version 1.1, Carnegie Mellon University.

SFS-ISO/IEC 29881, (2013). Tietotekniikka, järjestelmä- ja ohjelmistotekniikka. FiSMA 1.1 – toiminnallisen laajuuden mittausmenetelmä.

The Standish Group International, (2014). Big Bang Boom, haettu 28.10.2014 osoitteesta http://www.standishgroup.com/sample_research_files/BigBangBoom.pdf

van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J. & Bijlsma, L. (2006). Towards a reference framework for software product management, teoksessa *Proceedings in 14th IEEE International Conference on Requirements Engineering*, 319-322, Los Alamitos: IEEE Computer society.

van der Hoek, A., Hall, R. S., Heimbigner, D., & Wolf, A. L. (1997). *Software release management*, 22(6), 159-175.

Wikipedia, (2013). Haettu 5.11.2013 osoitteesta http://fi.wikipedia.org/wiki/Projektinhallinta

Wikipedia, (2016). Haettu 5.12.2016 osoitteesta https://fi.wikipedia.org/wiki/Laadullinen_tutkimus

Willman, P. (1996). Organizational Change in the 1990's, Lecture, University of Limerick, Irlanti.

Vähäniitty, J. (2003). Key Decisions in Strategic New Product Development for Small Software Product Businesses, teoksessa *Proceedings of the 29th EUROMICRO CONFERENCE, Software Process and Product Improvement track*, 375-383, Los Alamitos: IEEE.

Vähäniitty, J., Lassenius, C., & Rautiainen, K. (2002). An Approach to Product Roadmapping in Small Software Product Businesses, teoksessa J. Kontio & R. Conradi (toim.) *Quality Connection - 7th European Conference on Software Quality - Conference Notes*, 12-13, Berliini: Springer.

Vähäniitty, J. & Rautiainen, K. (2005). Towards an Approach for Development Portfolio Management in Small Product-Oriented Software Companies, teoksessa *Proceedings of the 38th Hawaii International Conference on System Sciences (HICSS-38)*, 314c, Los Alamitos: IEEE.

Xu, L. & Brinkkemper, S. (2007). Concepts of product software: Paving the road for urgently needed research, *European Journal of Information Systems*, 16(5), 531- 541.

Yin, R. (1994). *Case Study Research : Design and Methods*. California: SAGE Publications.