HAAGA-HELIA
ammattikorkeakoulu

Vaatimusten priorisointi kansainvälisessä ohjelmistotuoteyrityksessä

Miika-Markus Nurminen

Opinnäytetyö
Liiketalouden ylempi ammattikorkea-koulututkinto
Tietojärjestelmäosaamisen koulutus-ohjelma
2011

HAAGA-HELIA
ammattikorkeakoulu

Tiivistelmä

pp.kk.2011

Tietojärjestelmäosaamisen koulutusohjelma

Tekijät Miika-Markus Nurminen	**Ryhmä** YTI06K
Opinnäytetyön nimi Vaatimusten priorisointi kansainvälisessä ohjelmistotuoteyrityksessä	**Sivu- ja liitesivumäärä** 93 + 3
Ohjaajat Työn ohjaaja, Terttu Honkasaari, Johtaja, Tietotekniikan koulutusyksikkö, Haaga-Helia Työn valvoja, Tony Virtanen, Tuotejohtaja, QPR Software Oyj.	

Tämä tutkimus käsittelee ohjelmistotuotteille asetettujen vaatimusten priorisointia kansainvälisen ohjelmistotuoteyrityksen näkökulmasta.

Ohjelmistotuotteille asetetut vaatimukset ovat yhä monimutkaisempia ja vaativampia. Ohjelmistotuoteyritysten on valittava ohjelmistotuotteisiin sellaiset ominaisuudet, joiden avulla tuotetaan mahdollisimman suuri lisäarvo asiakkaille ja loppukäyttäjille. Näiden seurauksena on tehtävä valintoja toteuttavien ominaisuuksien suhteen, jotta valitut ominaisuudet pystytään toteuttamaan käytettävien resurssien puitteissa. Onnistuneella vaatimusten priorisoinnilla ohjelmistotuoteyritykset pystyvät huomioimaan markkinoiden vaatimukset tehokkaasti ja saavuttavat kilpailuetua eli ylivoimaista osaamista kilpailijoihin nähden.

Tutkimuksen tehtävänä on kehittää vaatimusten priorisointimalli, jota voidaan käyttää apuna päätettäessä, mitkä vaatimukset otetaan mukaan ohjelmiston seuraavaan julkaisuun ja mitkä vaatimukset voidaan jättää toteutettavaksi myöhemmin.

Tässä tutkimuksessa perehdytään vaatimusten hallintaan ohjelmistotuotannossa, minkä avulla pyritään selvittämään voidaanko ohjelmistotuotteille asetetut vaatimukset priorisoida systemaattisemmin ja läpinäkyvämmin kohdeyrityksessä. Vaatimusten priorisointia tutkitaan kohdeyrityksen käyttämien toimintatapojen, kirjallisuudessa esitettyjen käytäntöjen ja erilaisten priorisointimenetelmien näkökulmasta.

Tutkimus on rajattu niin, että siinä tutkitaan ainoastaan ohjelmistotuotteen vaatimusmäärittelyyn sisältyvää vaatimusten priorisointia ja vaatimusten priorisoinnin apuna käytettäviä menetelmiä. Tutkimuksen ulkopuolelle on jätetty kaikki muut ohjelmistotuotannon osa-alueet sulautetut ohjelmistotuotteet mukaan lukien.

Tutkimuksen kehittämistehtävä on toteutettu hyödyntämällä iteratiivista ja inkrementaalia lähestymistapaa toteuttamalla teoreettisen tiedon hankinta, käytännöllisen tiedon hankinta ja mallinnus ennalta määritellyissä sykleissä.

Tutkimuksen konstruktiivinen uudistus on ohjelmistotuotteen priorisointimalli, joka mahdollistaa ohjelmistotuotteelle asetettujen vaatimusten systemaattisen ja läpinäkyvän priorisoinnin kohdeyrityksessä.

Asiasanat
Ohjelmiston elinkaari, vaatimustenhallinta, vaatimusten priorisointi, liiketoimintavaatimus, käyttäjävaatimus, ohjelmistovaatimus, järjestelmävaatimus, priorisointimenetelmä

HAAGA-HELIA
ammattikorkeakoulu

Abstract

Date of presentation

Degree programme

Authors Miika-Markus Nurminen	**Group** YTI06K
The title of thesis Requirements prioritization in the international software company	**Number of pages and appendices** 93 + 3
Supervisors Instructor, Terttu Honkasaari, Director, IT Education Unit, Haaga-Helia Supervisor, Tony Virtanen, Vice President, Products & Technology, QPR Software Plc.	

This thesis wields requirements prioritization of requirements of software products in the international software company.

Requirements of software products are ever complex and demanding. In order to provide as much value for the customers and the end users as possible software companies need to include only particular functionalities to the software products. Because of this companies need to make decisions regarding functionalities they would like to implement so that selected functionalities can be implemented using available resources. Using a successful requirements prioritization practices software product companies can take account of requirements of the markets efficiently and gain competitive advantage comparing to its competitors.

The assignment of the thesis is to develop a prioritization model of the requirements that can be used when deciding which requirements will be included to next software release and which requirements will be postponed to the later software releases.

This thesis concerns requirement prioritization in the software engineering field that will be used as frame in order to figure out if the requirements of the software products can be prioritized more systematically and more transparently in the target organization. Requirements prioritization is investigated using the operation model of the target organization, literature review and various prioritization methods.

This thesis is delimited so that it studies only the prioritization of the requirements of the software product and the prioritization methods that can be used for requirements prioritization. All the other software engineering fields are excluded from this thesis including embedded systems.

The development project of the thesis is implemented using an incremental and an iterative project management method that contains predefined cycles for the theoretical and practical data management.

The constructive innovation of the thesis is the prioritization model of the requirements of the software product that enables the prioritization of the requirements systematically and transparently in the target organization.

Key words
Software life cycle, requirements management, requirement prioritization, business requirement, user requirement, software requirement, system requirement, prioritization method

Sisällys

1 Johdanto

”The indispensable first step to getting what you want is this: Decide what you want (Cohn 2006, 79).”

Ohjelmistotuotteen kehityksessä on harvoin, jos koskaan, aikaa toteuttaa kaikki sille esitetyt vaatimukset. Eri sidosryhmien asettamien vaatimusten hallinta on vaativa prosessi, joten pelkkä ohjelmistotuotteelle esitettyjen vaatimusten kokoaminen ei ole riittävä toimenpide. Ohjelmistotuotteeseen sisällytettävät vaatimukset on pystyttävä erittelemään niin, että niiden välillä voidaan tehdä selkeitä valintoja tuotekehityksen aikana.

Laura Lehtola toteaa tutkimuksessaan, että tuotekehitys on ohjelmistotuoteyritykselle investointi, jonka tarkoituksena on tuottaa mahdollisimman paljon lisäarvoa asiakkaille ja loppukäyttäjille. Näin ollen lisäarvon tuottaminen asiakkaille ja loppukäyttäjille on välttämätöntä ohjelmistotuotteen myynnin sekä yrityksen liiketoiminnan näkökulmasta. Yrityksen on pystyttävä valitsemaan tiettyyn ohjelmistotuotejulkaisuun sellaiset ominaisuudet, joiden avulla voidaan tuottaa mahdollisimman suuri lisäarvo asiakkaille ja loppukäyttäjille. (Lehtola 2006, 2.)

Asiakkaiden vaatimukset ohjelmistotuotteille ovat yhä monimutkaisempia ja vaativampia. Näiden seurauksena on pystyttävä tekemään valintoja toteuttavien ominaisuuksien suhteen, jotta ne pystytään toteuttamaan käytettävien resurssien puitteissa. Toteuttavien ominaisuuksien on tarjottava kilpailuetua, huomioitava tarjottavien ohjelmistoratkaisujen välttämättömät toiminnallisuudet ja toisaalta huolehdittava olemassa olevien asiakkaiden vaatimista toiminnallisuuksien laajennuksista.

Kansainvälisen ohjelmistotuoteyrityksen näkökulmasta vaatimusten priorisointi eroaa yksittäisistä asiakasprojekteista erityisesti siinä, että vaatimuksia esittävät samanaikaisesti useat sidosryhmät. Käytännössä tämä voi tarkoittaa tuhansia vaatimuksia loppuasiakkailta ja yhteistyökumppaneilta maailman laajuisesti ja oman liiketoiminnan esittämiä vaatimuksia, jotka pohjautuvat mm. kilpailijaseurantaan, markkina-analyyseihin, innovaatioihin sekä ohjelmiston laatuvaatimuksiin, kuten käytettävyyteen, suorituskykyyn ja luotettavuuteen.

Tämän tutkimuksen konstruktiivinen uudistus on tekemäni vaatimusten priorisointimalli ohjelmistotuotteelle, jonka tavoitteena on auttaa ja helpottaa kansainvälistä ohjelmistotuoteyritystä priorisoimaan ohjelmistotuotteisiin toteutettavia vaatimuksia systemaattisesti ja läpinäkyvästi. Keskityn tässä tutkimuksessa ohjelmistotuotteen priorisointimallin kehitysvaiheisiin ana-

lysoimalla kohdeyrityksen nykytilannetta ja perehtymällä kirjallisuudessa esiteltyihin suosituksiin ja menetelmiin.

1.1 Tulostavoitteet

Tämän tutkimuksen tehtävänä on kehittää vaatimusten priorisointimalli, jota voidaan käyttää apuna päätettäessä mitkä vaatimukset otetaan mukaan ohjelmiston seuraavaan julkaisuun ja mitkä vaatimukset voidaan jättää toteutettavaksi myöhemmin.

Itse tutkimukselle asetetut tulostavoitteet olen jakanut kolmeen osaan:

1. Aiemmista tutkimuksista on pystytty laatimaan perusta ohjelmistotuotteisiin toteutettavien vaatimusten priorisoinnin tutkimiselle eli johtamaan tutkimusongelmat
2. Uusi tutkimus on tuottanut vastaukset ja ratkaisut tutkimusongelmiin
3. Tutkimuksessa on arvioitu saatujen tuloksien merkitystä, luotettavuutta ja käytettävyyttä eli miten ongelmat onnistuttiin ratkaisemaan tutkimuksessa

1.2 Johtoajatus, tutkimusongelmat ja aihepiiri

Hallitakseen tehokkaasti ohjelmistotuotteelle esitettyjä vaatimuksia ohjelmistotuoteyrityksen on pystyttävä valitsemaan tietyn mallin avulla mitkä vaatimukset otetaan mukaan ohjelmistotuotteen seuraavaan julkaistavaan versioon ja mitkä vaatimukset voidaan jättää toteutettavaksi myöhemmin. Em. johtoajatuksesta olen kiteyttänyt tutkimuksen pääongelman seuraavasti: Voidaanko ohjelmistotuotteelle esitetyt vaatimukset priorisoida systemaattisemmin ja läpinäkyvämmin kohdeyrityksessä? Pääongelman olen jakanut edelleen kolmeen tarkentavaan osaongelmaan:

1. Mitä vaatimusten priorisointi tarkoittaa tällä hetkellä kohdeyrityksessä?
2. Millaisia suosituksia ja ratkaisuja kirjallisuudessa on tarjolla ohjelmistotuotteiden vaatimusten priorisointikäytännöille?
3. Millä menetelmillä toteutettavia vaatimuksia voidaan priorisoida kohdeyrityksessä?

Perehdyn tässä tutkimuksessa vaatimusten hallintaan ohjelmistotuotannossa vastatakseni em. tutkimusongelmiin, joka on näin ollen tutkimuksen aihepiiri.

1.3 Rajaus

Olen rajannut tutkimukseni niin, että tutkin pelkästään ohjelmistotuotteen vaatimusmäärittelyyn sisältyvää vaatimusten priorisointia ja vaatimusten priorisoinnin apuna käytettäviä mene-

telmiä. Tutkimuksen ulkopuolelle jätän kaikki muut ohjelmistotuotannon osa-alueet, kuten ohjelmiston suunnittelun, ohjelmiston toteutuksen ja yksikkötestauksen, integroinnin ja järjestelmätestauksen, sekä käytön ja ylläpidon mukaan lukien ohjelmistotuotteisiin korjattavien virheiden priorisoinnin. Sulautetut ohjelmistotuotteet eivät myöskään kuulu tämän tutkimuksen piiriin.

Keskityn tutkimuksessani parantamaan kohdeyrityksessä työskentelevien henkilöiden priorisointiprosessia. Yrityksen suorat asiakkaat, yhteistyökumppanit ja yhteistyökumppaneiden asiakkaat rajaan tämän tutkimuksen ulkopuolelle.

1.4 Tutkimusmenetelmät

Suunnitelmallinen ja tavoitteellinen tutkimus on prosessi, johon kuuluvat aiheeseen perehtyminen, suunnitelman laadinta, varsinaisen tutkimuksen toteutus ja tutkimusraportin laadinta. Kehittämiskohdetta voi lähestyä monin eri tavoin, joten ennen tutkimusmenetelmien lähempää tarkastelua on valittava myös tutkimusstrategia. Tässä alaluvussa esittelen tutkimuksessa käyttämäni tutkimusprosessin, tutkimusstrategian ja tutkimusmetodit.

1.4.1 Tutkimusprosessi

Tutkimusprosessin avulla voidaan jäsentää tutkimusta ja prosessin avulla on myös helpompi lähestyä tutkittavaa ongelmaa.

Tämän tutkimuksen tutkimusprosessi on laadittu soveltamalla tutkimussuunnitelmaa painottavaa tutkimuksen kulkua, Jenkinsin mallia ja konstruktiivisen tutkimuksen prosessia kuvion 1 mukaisesti (Hirsjärvi, Remes & Sajavaara 2009, 65; Järvinen & Järvinen 2004, 3; Ojasalo, Moilanen, & Ritalahti 2009, 67).

1. Perehtymisvaihe
    - Idea
    - Kirjallisuuskartoitus
    - Tutkimusaihe
2. Suunnitteluvaihe
    - Tutkimusongelmat
    - Tutkimusstrategia
        - Tutkimuksen menetelmällisten ratkaisujen kokonaisuus
    - Tutkimusmetodit
        - Aineistonkeruu- ja analysointimenetelmät
    - Tutkimusprosessi
3. Toteutusvaihe
    - Tietojen keruu
        - Syvällisen teoreettisen tiedon hankinta tutkimuksen kohteesta
        - Käytännöllisen tiedon hankinta tutkimuksen kohteesta
    - Ratkaisun laatiminen ja toteuttaminen
    - Tietojen analysointi
        - Ratkaisun toimivuuden testaus ja konstruktion oikeellisuuden osoittaminen
        - Ratkaisussa käytettyjen teoriakytkentöjen näyttäminen ja ratkaisun uutuusarvon osoittaminen
        - Ratkaisun soveltamisalueen laajuuden tarkastelu
4. Raportointivaihe
    - Tulosten julkistaminen

Kuvio 1. Tutkimusprosessin vaiheet.

Tutkimusprosessi koostui neljästä päävaiheesta; perehtymis-, suunnittelu-, toteutus- ja raportointivaihe.

Perehtymisvaiheessa tutkimusta tarkennettiin vastamaalla kysymykseen mitä tutkitaan. Kirjallisuuskartoituksen avulla muodostettiin alustava teoriatausta ja selkiinnytettiin tutkimuksen lähtökohdat, jotta tutkimusongelma pystyttiin rajaamaan tehokkaasti. Tehokas rajaus edesauttoi muodostamaan tutkimuksen suuntaviivat ja näkökulmat varsinaisille tutkimusongelmille. Näin ollen perehtymisvaiheessa muodostettiin hypoteesi tutkimuksen aihepiiristä eli oletus siitä mitä tutkimuksessa tullaan havaitsemaan.

Suunnitteluvaiheessa haettiin vastauksia kysymykseen miten tutkimus suoritetaan. Tutkimusongelma nimettiin täsmällisesti ja muotoiltiin ymmärrettävästi. Tutkimusstrategiaa valittaessa pohdittiin millaisella lähestymistavalla kehittämistyötä voidaan viedä parhaiten eteenpäin ja mitkä lähestymistavat ovat ylipäätänsä mahdollisia. Tutkimusmetodeja valittaessa pyrittiin hahmottamaan eri menetelmiä, joilla saadaan kehittämistyön tueksi erilaista tietoa ja erilaisia ideoita eri näkökulmista. Tutkimusmetodien valinnassa selvitettiin mitä metodeja on käytettävissä, mitkä metodit soveltuvat ongelman ratkaisemiseksi ja millaisia rajoituksia tutkimuksen käytännön toteutus asettaa käytettäville metodeille. Tutkimusprosessia määriteltäessä hahmotettiin kokonaiskuva tutkimukseen liittyvistä vaiheista, joiden avulla tutkimus toteutettiin systemaattisesti huomioimalla kunkin vaiheen sisältämät asiat ennen seuraavaan vaiheeseen siirtymistä.

Toteutusvaiheessa toteutettiin tutkimuksen varsinainen kehittämistehtävä valitun tutkimusstrategian ja tutkimusmetodien avulla. Toteutusvaiheen aikana syvennettiin myös tutkimuksen teoriataustaa ja hankittiin käytännön tietoa valittujen tutkimusmetodien avulla. Kehittämistehtävä toteutettiin idean pohjalta, jota iteroitiin, kunnes saavutettiin haluttu lopputulos. Lopputulosta hyödynnettiin kohdeorganisaatiossa käytännössä, jotta sen toimivuutta pystyttiin analysoimaan.

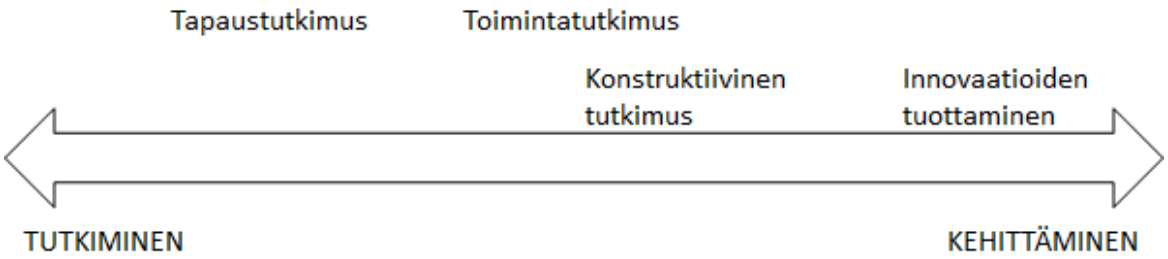
Raportointivaiheessa julkistettiin tutkimustyön tulokset ja itse tutkimuksen loppuraportti.

1.4.2 Tutkimusstrategia

Tutkimusstrategialla eli tutkimuksen valitulla lähestymistavalla tarkoitetaan tutkimuksen menetelmällisten ratkaisujen kokonaisuutta. Tutkimusmetodit eli tutkimustyössä käytettävät menetelmät erotetaan tutkimusstrategiasta suppeampana käsitteenä. Sekä tutkimusstrategian että yksittäisten tutkimusmetodien valinta riippuu tutkimustehtävästä ja tutkimuksen ongelmista. Näin ollen on selvää, että tutkimuksesta tulee erilainen riippuen siitä millaista tutkimusstrategiaa tutkimuksessa sovelletaan. (Hirsjärvi ym. 2009, 123–132.)

Tutkimuksessa toteutettava kehittämistehtävä määrittää, mikä lähestymistapa sopii parhaiten kehittämistyöhön. Todennäköisin lähestymistapa on tapaustutkimus, jos kehittämistehtävän tavoitteena on tuottaa yritykselle tiettyjä kehittämisehdotuksia. Sopivin lähestymistapa on konstruktiivinen tutkimus, jos tehtävänä on käytännön ongelman ratkaisu luomalla uusi konstruktio eli jokin konkreettinen tuotos, kuten malli tai menetelmä. (Ojasalo ym. 2009, 37.)

Kuviossa 2 havainnollistetaan eräitä lähestymistapoja tutkiminen-kehittäminen-jatkumolla, joka kuvaa kuinka paljon tavoitteet painottuvat eri lähestymistavoissa (Ojasalo ym. 2009, 36–37).

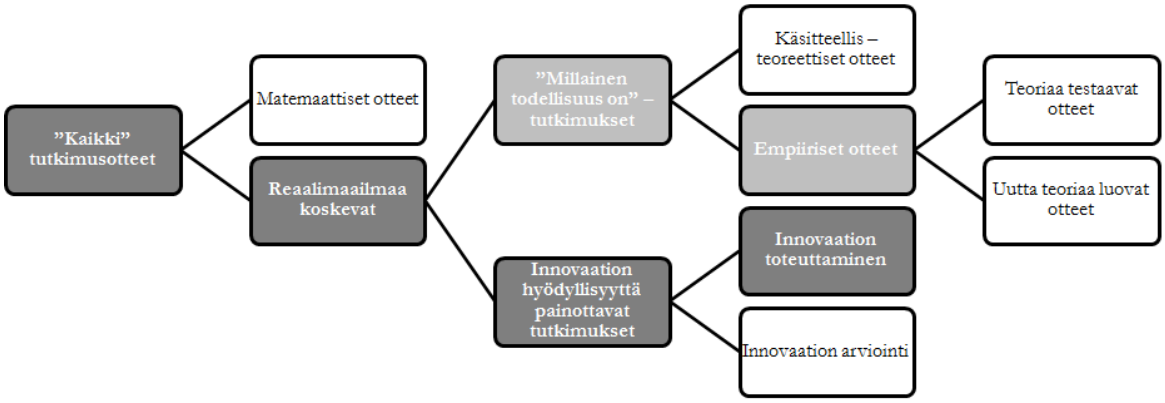


Kuvio 2. Lähestymistavat tutkiminen-kehittäminen-jatkumolla (Ojasalo ym. 2009, 37).

1.4.2.1 Konstruktiivinen tutkimus

Tässä tutkimuksessa tutkittiin ohjelmistotuoteyrityksen mahdollisuuksia priorisoida vaatimuksia ja ohjelmistoyrityksen toimintaympäristöä. Lisäksi tutkimuksessa oli olennaista tuntea kehitettävän mallin käyttäjien tarpeet hyvin. Tutkimuksessa pyrittiin käytännönläheiseen ongelmanratkaisuun luomalla uusi malli, jonka kehittämiseksi tarvittiin sekä olemassa olevaa teoreettista tietoa että käytännön ongelmista kerättävää tietoa. Toimintaympäristöä käsiteltiin ainoastaan siitä näkökulmasta, että pystyttiin ymmärtämään haasteet ja ongelmat, jotka auttoivat uudenlaisen toimintatavan eli mallin rakentamisessa.

Järvisen & Järvisen tutkimusmetodien taksonomian mukaisesti kyseessä oli reaalimaailmaa käsittelevä tutkimus. Tutkimuksessa rakennettavan mallin kehittäminen pohjautui näin ollen innovaation hyödyllisyyttä painottavaan tutkimukseen eli konstruktion toteuttamiseen kuvion 3 mukaisesti. (Järvinen & Järvinen 2004, 10.)



Kuvio 3. Järvisen ja Järvisen tutkimusmetodien taksonomia (Järvinen & Järvinen 2004, 10).

Konstruktiivisessa tutkimuksessa on kyse uudenkaltaisen todellisuuden rakentamisesta tutkimustiedon pohjalta, jossa pyritään käytännönläheiseen ongelmanratkaisuun luomalla uusi rakenne. Uuden rakenteen luomiseksi tarvitaan sekä olemassa olevaa teoreettista tietoa että käytännöstä kerättävää tietoa. Uusi rakenne voi myös parantaa aikaisempaa toimintaprosessia ja se on näin ollen aiempaa parempi ratkaisu todelliseen ongelmaan. Konstruktiivisessa tutkimuksessa kohdeorganisaatio saa puolueettoman ja teoreettiseen tietämykseen perustuvan ratkaisun ongelmaan, jossa korostuu tutkimuksen hyödyntäjän ja toteuttajan välinen kommunikaatio. (Ojasalo ym. 2009, 65–66.)

Järvinen & Järvinen analysoivat suunnittelutieteellistä tutkimusta eli konstruktiivista tutkimusta Marchin ja Smithin mukaan, jolloin tutkimusten tulokset ovat neljäntyyppisiä: käsitteistöjä, malleja, metodeja ja realisointeja. Käsitteistö muodostaa tutkimusaiheen sanaston. Malli on joukko lauseita, jotka ilmaisevat käsitteiden väliset suhteet. Metodi on joukko ohjeita, joita käytetään suorittamaan tehtävä. Realisointi on itse artefaktin toteutus ympäristössään. Artefaktin toteutukset operationalisoivat Marchin ja Smithin mukaan käsitteistöjä, malleja ja metodeja. (Järvinen & Järvinen 2004, 107.)

Edellä mainittuihin konstruktiivisen tutkimuksen teorioihin viitaten pystyttiin toteamaan, että tässä tutkimuksessa esitetylle kehityshankkeelle tunnistettiin konstruoinnin lähtökohta ja ajatus toivotusta tavoitetilasta. Tavoitetilan kuvaus oli malli tilanteesta, jolla ohjelmistotuotteen vaatimuksia pystyttäisiin priorisoimaan, kun idea oli realisoitu eli toteutettu metodilla, jonka avulla muutos lähtötilasta tavoitetilaan uskottiin saavutettavan.

1.4.3 Tutkimusmetodit

Tutkimus- ja kehittämistyössä oli mahdollista käyttää monenlaisia menetelmiä. Tämän tutkimuksen eteneminen määräytyi olennaisilta osin sen mukaan kuinka paljon aikaisempaa teoriaa ja tutkimustietoa tutkimusaiheesta löytyi.

Yinin mukaan tietoja ja näyttöä voi kerätä ainakin kuudesta eri lähteestä joita ovat: dokumentit, arkistot, haastattelut, vapaa havainnointi, osallistuva havainnointi ja fyysiset luomukset. Näiden lisäksi Yin painottaa kolmea eri periaatetta varsinaisessa tietojenkeruussa: useiden tietolähteiden käyttäminen, tietokannan laatiminen tietolähteistä mukaan lukien raakatiedot sekä tutkimusraportit ja perusteluketjujen ylläpitäminen. (Järvinen & Järvinen 2004, 81.)

Tämän tutkimuksen tietoperusta laadittiin pääasiallisesti erittelemällä sisältöä kirjallisuudesta, dokumenteista ja aikaisemmista tutkimuksista hyödyntämällä useita eri tietolähteitä. Kehittä-

mistehtävän toteutuksessa käytettiin pääasiallisesti laadullisia menetelmiä, kuten haastattelua ja osallistuvaa havainnointia sillä tutkimuksen kohteena olevia menetelmiä ei tunnettu entuudestaan kovin hyvin ja niitä haluttiin ymmärtää paremmin. Toisaalta tarkoituksena oli hankkia rajatusta kohteesta paljon tietoa ja ymmärtää tutkittavaa ilmiötä paremmin. Haastattelujen tehtävänä oli asioiden selventäminen ja syventäminen. Haastattelumenetelmänä käytettiin avointa haastattelua, jotta haastateltavilla oli mahdollisuus tuoda esille itseään koskevia asioita mahdollisimman vapaasti. Toisaalta voitiin olettaa, että tutkimuksen aihe tuottaa monitahoisia vastauksia. Haastateltaviksi valittiin henkilöt, jotka osallistuivat vaatimusten priorisointiin kohdeyrityksessä jollain tavalla. Havainnointien avulla pyrittiin täydentämään haastatteluja sekä varmistumaan siitä että haastateltavat toimivat oikeasti niin, kuin olivat haastatteluissa kertoneet. Lisäksi tutkimuksen tekijä oli hyvin lähellä tutkittavia ja osallistui myös itse aktiivisesti toimintaan, jottei tutkijaa koettu täysin ulkopuolisena. Havainnoinnin haluttiin olevan myös vapaasti tilanteessa muotoutuvaa.

Tutkimuksen kohde oli osa isompaa prosessia, joten työmenetelmänä käytettiin myös prosessianalyysiä, jonka avulla pyrittiin selvittämään missä vaiheissa, milloin ja ketkä osallistuivat vaatimusten priorisointiin.

1.5 Tutkimukseen liittyvät keskeiset käsitteet

Tässä luvussa esitellään tutkimukseen liittyvät keskeiset käsitteet, joita käytettiin laadittaessa tämän tutkimuksen tietoperusta erittelemällä sisältöä kirjallisuudesta, dokumenteista ja aikaisemmista tutkimuksista.

Asiakasvaatimus	Asiakkaan ja loppukäyttäjän tarpeen täyttävä toiminnallisuus.
Ei-toiminnallinen vaatimus	Esim. ohjelmistotuotteen suorituskykyyn ja laatuun liittyvät vaatimukset.
Järjestelmävaatimus	Ohjelmistotuotteen kokonaisarkkitehtuuriin liittyvä ohjelmisto- tai laitteistovaatimus.
Käyttäjävaatimus	Tehtäviä tai ominaisuuksia, joita käyttäjän pitää pystyä suorittamaan ohjelmistotuotteella.

Liiketoimintavaatimus	Korkean tason vaatimuksia, jotka kuvaavat tavoitteita, joita halutaan saavuttaa.
Ohjelmiston elinkaari	Ajanjakso ohjelmiston kehittämisen aloittamisesta ohjelmiston poistamiseen käytöstä.
Ohjelmistovaatimus	Asiakasvaatimuksen toiminnallinen määrittely ohjelmistossa. Toiminnallisuuksia ja toimintoja joita ohjelmistokehittäjät pystyvät toteuttamaan ohjelmistoon.
Ominaisuus	Toiminnallisen määrittelyn kokonaisuus, joka toteutetaan ohjelmistoon yksittäisinä toimintoina.
Priorisointimenetelmä	Menettelytapa, jonka avulla vaatimukset saadaan järjestettyä tärkeysjärjestykseen tietystä näkökulmasta.
Vaatimustenhallinta	Ohjelmistoprojektiin sisältyvät toimenpiteet, joilla varmistetaan asiakasvaatimukset täyttävä järjestelmä.
Vaatimusten priorisointi	Vaatimusten hallintaan sisältyvä toimenpide, jonka avulla valitaan ohjelmistoon toteutettavat ominaisuudet.

2 Vaatimusten priorisointi

Tässä kappaleessa käsitellään vaatimusten priorisointia ja vaatimusten eri vaiheita ohjelmiston elinkaaren näkökulmasta.

2.1 Priorisointi

Prioriteetilla tarkoitetaan ominaisuutta tai tilaa, joka on tärkeämpi tai etusijalla suoritettaessa vertailua saman aihepiiri sisällä (Merriam-Webster 2010 a). Priorisoinnilla tarkoitetaan esim. hankkeiden tai päämäärien listausta ja arviointia prioriteetin mukaisessa järjestyksessä (Merriam-Webster 2010 b).

Asiakkaiden todellisten tarpeiden selvittäminen ja täydellinen ymmärtäminen on silti aina vaikeaa. Lisäksi on tärkeää ymmärtää, että käytännössä asiakastarpeiden analysointi ja tarkentaminen jatkuvat yleensä koko vaatimusmäärittelyvaiheen ajan eli kyseessä ei ole kertaluontoinen harjoitus.

2.2 Vaatimus

Wiegers toteaa, että vaatimus voi olla mitä tahansa, mikä ohjaa suunnittelun valintoja. Lisäksi hän käsittelee vaatimuksia ominaisuuksina, jotka tuotteessa on oltava, jotta tuotteella voidaan luoda arvoa tuotteen eri sidosryhmille. (Wiegers 2003, 7-8.)

Robertsonit ovat samoilla linjoilla Wiegersin kanssa sillä heidän mielestään vaatimuksilla tarkoitetaan asioita, jotka pitäisi saada selville, ennen kuin tuotteen rakentaminen aloitetaan. He painottavat, että vaatimusten selvittäminen vasta rakentamis- tai tuotantovaiheessa on niin kallista ja tehotonta, että olettavat jokaisen järkevän ihmisen tiedostavan tämän tosiasian. (Robertson & Robertson 1999, 1.)

Kotonyan ja Sommervillen mukaan vaatimuksilla tarkoitetaan järjestelmäpalveluiden tai rajoitusten selostuksia puhuttaessa ohjelmiston vaatimusmäärittelystä. Näin ollen vaatimukset ovat kuvauksia siitä, miten järjestelmän tulisi toimia, kuvauksia sovelluksen piirissä käsiteltävästä informaatiosta, kuvauksia järjestelmän käyttöön liittyvistä rajoituksista ja määrittelyjä järjestelmän ominaisuuksista tai yksittäisistä attribuuteista. Kotonya ja Sommerville toteavat, että käytännössä vaatimukset sisältävät tietoa siitä, mitä järjestelmän pitäisi tehdä sekä siitä miten järjestelmä suorittaa vaaditut toimenpiteet. Tämä koetaan yleensä haasteelliseksi, koska vaatimuk-

set sisältävät sekoituksen ongelmasta, ominaisuuksista, suunnitelmasta, kuvauksen järjestelmän toiminnasta ja mahdollisista rajoituksista. (Kotonya & Sommerville 2003, 6-7.)

Davisin näkemyksen mukaan vaatimus on halutun järjestelmän ulkoisesti havaittavissa oleva ominaispiirre. Validin vaatimuksen pitää läpäistä kaksi testiä: 1) vaatimuksen toteutuminen pitää pystyä havaitsemaan järjestelmän ulkopuolisesta näkökulmasta ja 2) vaatimuksen pitää toteuttaa potentiaalisen asiakkaan tai sidosryhmän tarve. Lisäksi Davis korostaa, että ainoastaan asiakas tai loppukäyttäjä pystyy määrittämään, onko vaatimus järkevä vai ei. (Davis 2005, 3-6.)

Bray esittää, että vaatimukset ja ongelmat ovat lähestulkoon yksi ja sama asia. Hänen mukaansa vaatimukset ovat vaikutuksia, joita asiakas toivoo saatavan aikaiseksi ongelmien vaikutusalueella. Hänen mukaansa tämän lähestymistavan haasteena on tehdä ero itse vaatimuksen ja ongelman vaikutusalueen tosiasioiden välillä. Tämä johtuu siitä, että yksittäiset tosiasiat ainoastaan kuvaavat ongelman vaikutusalueen ominaispiirteitä. (Bray 2002, 14–15.)

Haikala ja Märijärvi toteavat, että vaatimusten käytölle ei ole olemassa mitään vakiintunutta käytäntöä sillä vaatimus on yleisnimitys, joka viittaa tietyn ohjelmistotyön vaiheen syötteeseen. Myös he esittävät, että mm. asiakasvaatimus on puhtaimmillaan asiakkaan ongelma, kuten Braykin toteaa määritelmässään. Käytännössä tämä tarkoittaa sitä, että asiakas haluaa tuotteeseen ominaisuuden, jonka avulla ongelmat voidaan poistaa. Ominaisuus voidaan toteuttaa toiminnoilla, jotka voivat edelleen liittyä useaan ominaisuuteen. (Haikala & Märijärvi 2004, 39.)

Edellä esitettyjen määritelmien pohjalta on suhteellisen helppo todeta, että vaatimukselle on vaikea löytää yleispätevä määritelmä. Em. vaatimusten määritelmistä voidaan mielestäni kuitenkin huomata, että kommunikointia eri sidosryhmien välillä on painotettava, jotta saadaan tietoon eri sidosryhmien näkökulmat. Mielestäni on myös tärkeää huomioida, että ohjelmistotuotannossa on käytössä paljon erilaisia vaatimuserittelyjä kuten käyttäjä-, ohjelmisto-, toiminnallinen -, järjestelmä-, tekninen -, liiketoiminta- tai tuotevaatimus. Em. jaottelun käyttöä eri sidosryhmien parissa on mietittävä huolella, jotta se ei aiheuta sekaannuksia ja turhautumista. Käytännössä mielestäni on kuitenkin tärkeintä, että kaikki vaatimukset dokumentoidaan systemaattisesti, jotta varsinainen ominaisuus voidaan toteuttaa mahdollisimman tehokkaasti.

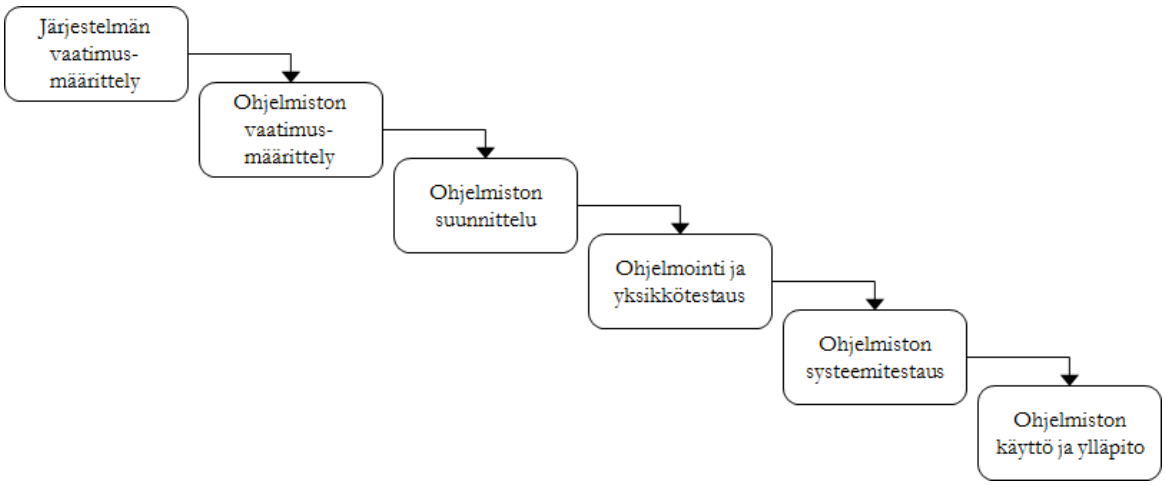
2.2.1 Vaatimuksien suhde vaihejakomalleihin

Wohlin ym. toteavat, että ohjelmistotuotteiden kehittäminen on usein monimutkainen tehtävä. Kehitysprojektit saattavat olla pitkäkestoisia ja niissä on mukana paljon ihmisiä erityisesti sil-

loin, kun kehitettävät ohjelmistotuotteet ovat monimutkaisia. Tämän takia myös ohjelmistotuotantoprosessit saattavat muodostua monimutkaisiksi. Yrityksille on kuitenkin tärkeää pyrkiä parantamaan liiketoimintaansa, mikäli ne haluavat säilyttää kilpailukykynsä. Yritykset pyrkivät jatkuvasti parantamaan ohjelmistotuotantoprosessejaan parantaakseen tuotteitaan, pienentääkseen kustannuksiaan, jne. Wohlin ym. painottavatkin ohjelmistotuotantoprosessien merkitystä systemaattisina ja kurinalaisina työskentelytapoina. (Wohlin ym. 2000, 2.)

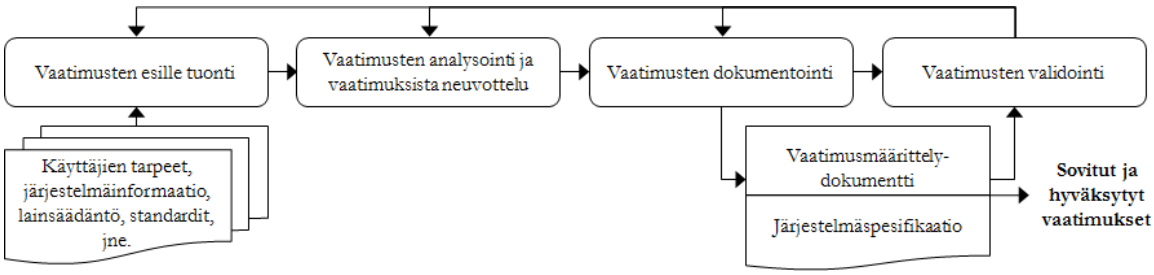
Haikala ja Märijärvi käsittelevät vaatimuksia ohjelmiston elinkaaren näkökulmasta, jolla tarkoitetaan aikaa, joka kuluu ohjelmiston kehittämisen aloittamisesta sen poistamiseen käytöstä. Kyseinen malli tunnetaan vaihejakomallina, jossa ohjelmiston kehitystyö jaetaan eri vaiheisiin. Mallista voidaan erottaa yleensä ainakin määrittely-, suunnittelu- ja toteutusvaihe. Ennen määrittelyvaihetta suoritetaan usein esitutkimus tai tarvekartoitus, jonka tehtävänä on asettaa yleiset järjestelmätason vaatimukset. Tällaiset vaatimukset ovat asiakasvaatimuksia, koska ne määrittelevät ainoastaan asiakkaan tarpeet mutta eivät ota kantaa siihen, millainen itse järjestelmän pitää olla. Tästä näkökulmasta esitutkimusta voidaan pitää elinkaaren tärkeimpänä vaiheena, koska sen avulla eliminoidaan väärien asiakasvaatimusten sisällyttäminen järjestelmään. Asiakasvaatimuksista keskustellaan ja niitä analysoidaan määrittelyvaiheessa. Niiden pohjalta määritellään sekä järjestelmävaatimukset että ohjelmistovaatimukset, jotka yhdessä muodostavat ohjelmiston toiminnallisen määrittelyn. Toiminnallinen määrittelydokumentti sisältää ohjelmiston toiminnot, ei-toiminnalliset vaatimukset ja rajoitukset. Määrittelyvaiheen avulla asiakasvaatimukset muunnetaan ohjelmistovaatimuksiksi. (Haikala & Märijärvi 2004, 36-39.)

Kotonya ja Sommerville kuvaavat ohjelmistokehitysprosessin kuvion 4 mukaisesti. Siinä ohjelmistokehitysprosessi noudattaa 1970-luvulla kehitettyä vesiputousmallia, jonka avulla saa hyvän yleiskuvan ohjelmistokehitysprosessin vaihejaosta. Kotonya ja Sommerville toteavatkin, ettei yksittäinen malli anna täydellistä ymmärrystä prosessista vaan siihen tarvitaan useita erillisiä malleja, joiden avulla voidaan tuottaa prosessi-informaatiota eri näkökulmista. Vaatimusmäärittelyn näkökulmasta järjestelmän ja ohjelmiston vaatimusmäärittelylle sekä ohjelmistosuunnittelulle on vaikea tehdä selkeää eroa käytännössä. (Kotonya & Sommerville 2003, 34.)



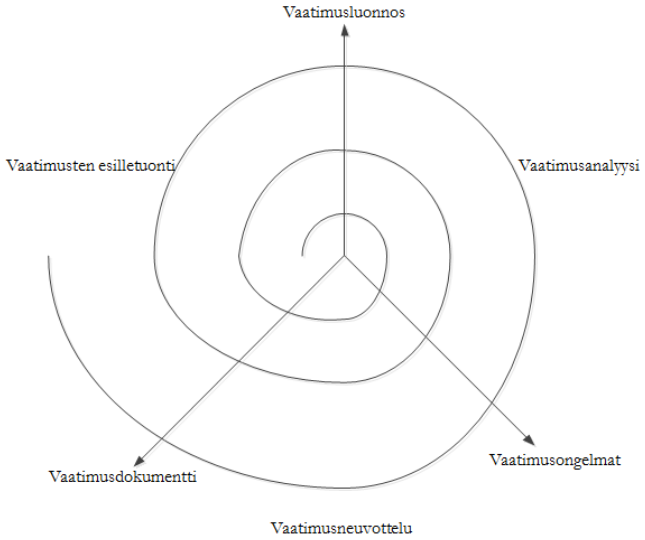
Kuvio 4. Ohjelmistokehitysprosessi (Kotonya & Sommerville 2003, 34).

Kuviossa 5 on kuvattu Kotonyan ja Sommervillen näkemys vaatimusten määrittelyprosessista, josta käy ilmi tärkeimmät aktiviteetit ja niiden keskinäinen järjestys. Vaatimusten määrittely-prosessissa ei ole selkeitä rajoja aktiviteettien välillä. Käytännössä tämä tarkoittaa sitä, että aktiviteettien suoritusjärjestys voi olla limittäinen eli prosessi sisältää sisäisiä iteraatioita ja aktiviteettien välisiä syötteitä sekä tuotoksia.



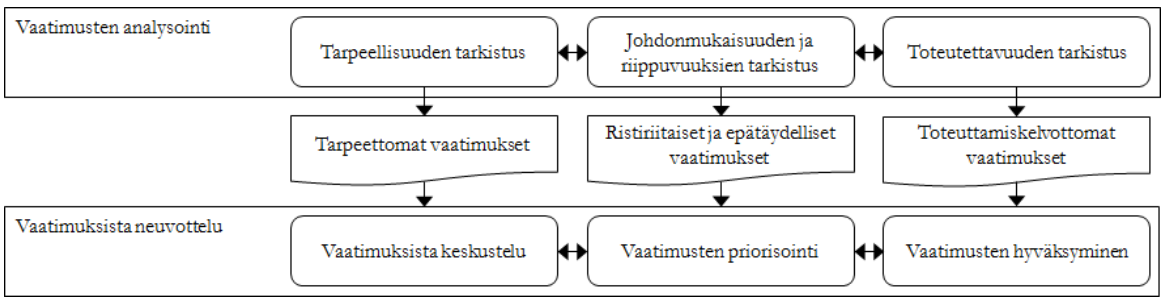
Kuvio 5. Vaatimusten määrittelyprosessi (Kotonya & Sommerville 2003, 32).

Kotonyan ja Sommervillen mukaan vaatimusten esilletuontivaihe, vaatimusten analysointivaihe ja vaatimusten neuvotteluvaihe pitää pystyä aktivoimaan uudestaan, milloin tahansa. Tämän takia em. vaiheita on heidän mukaansa hyvä käsitellä kuviossa 6 kuvatun spiraalin avulla. (Kotonya & Sommerville 2003, 58.)



Kuvio 6. Vaatimusten esilletuonti-, analysointi- ja neuvotteluvaiheen spiraali (Kotonya & Sommerville 2003, 58).

Kuviossa 7 kuvatun vaatimusten analysointi- ja neuvotteluprosessin tehtävänä on ratkaista vaatimuksiin liittyvät ristiriitaisuudet sekä päällekkäisyydet ja hyväksyä toteutettavat vaatimukset. Tarpeellisuuksien, johdonmukaisuuksien, riippuvuuksien ja toteutettavuuksien tarkistuksilla identifioidaan vaatimukset, joista keskustellaan tarkemmin vaatimusten neuvotteluprosessissa. Neuvotteluprosessin avulla varmistetaan, että kaikki sidosryhmät pystyvät kertomaan oman näkökulmansa, kiistanalaiset vaatimukset saadaan priorisoitua ja lopuksi ratkaisut vaatimuksiin liittyvistä ongelmista identifioidaan ja hyväksytään toteutettavat vaatimukset. Kotonya ja Sommerville toteavat, että vaatimusten analysointi ja vaatimuksista keskustelu sisältävät paljon konfliktitilanteita ja toteutettavat vaatimukset valitaan yleensä erilaisten kompromissien tuloksena. (Kotonya & Sommerville 2003, 59-61.)



Kuvio 7. Vaatimusten analysointi- ja neuvotteluprosessi (Kotonya & Sommerville 2003, 60).

Hamlet ja Maybee painottavat, että ohjelmistokehitysprosessin tärkein vaihe on vaatimusten analysointi, sillä analysointivaiheessa selvitetään mitä ohjelmistotuotteen täytyy tehdä (Hamlet & Maybee 2001, 119).

Sommervillen mukaan vaatimusten analysointiprosessi on vaikea prosessi useasta eri syystä, joista hän nostaa esille seuraavat haasteet: 1) sidosryhmät eivät yleensä tiedä mitä he haluavat ohjelmistotuotteilta, lukuun ottamatta yksinkertaisimpia käyttötapoja. Vaikka heillä on selkeä idea mitä ohjelmistotuotteen pitäisi pystyä tekemään, he kokevat vaikeaksi yksilöidä ja eritellä vaatimukset, 2) sidosryhmät kytkevät vaatimuksensa yleensä omaan toimintaansa ja kuvaavat vaatimuksensa oman toimintansa kontekstissa. Analysoitaessa vaatimuksia ne on pystyttävä tarvittaessa yleistämään monelle toiminta-alueelle, 3) eri sidosryhmillä on erilaisia vaatimuksia ja ne ilmaistaan eri tavoilla. Ohjelmistotoimittajien on pystyttävä tunnistamaan eri vaatimusten yhteneväisyydet ja ristiriitaisuudet, 4) liiketoimintajohdon henkilökohtaiset mieltymykset saattavat olla epäselviä todellisille loppukäyttäjille, sillä ohjelmistotoimittajat saattavat joutua toteuttamaan henkilökohtaisia vaatimuksia isojen kauppojen toivossa ja 5) liiketoimintaympäristö on dynaaminen ja se muuttuu analysointiprosessin aikana. Ohjelmistotoimittajien pitää pystyä reagoimaan uusien sidosryhmien vaatimuksiin, jotka liittyvät jo analysoituihin vaatimuksiin. (Sommerville 1996, 80.)

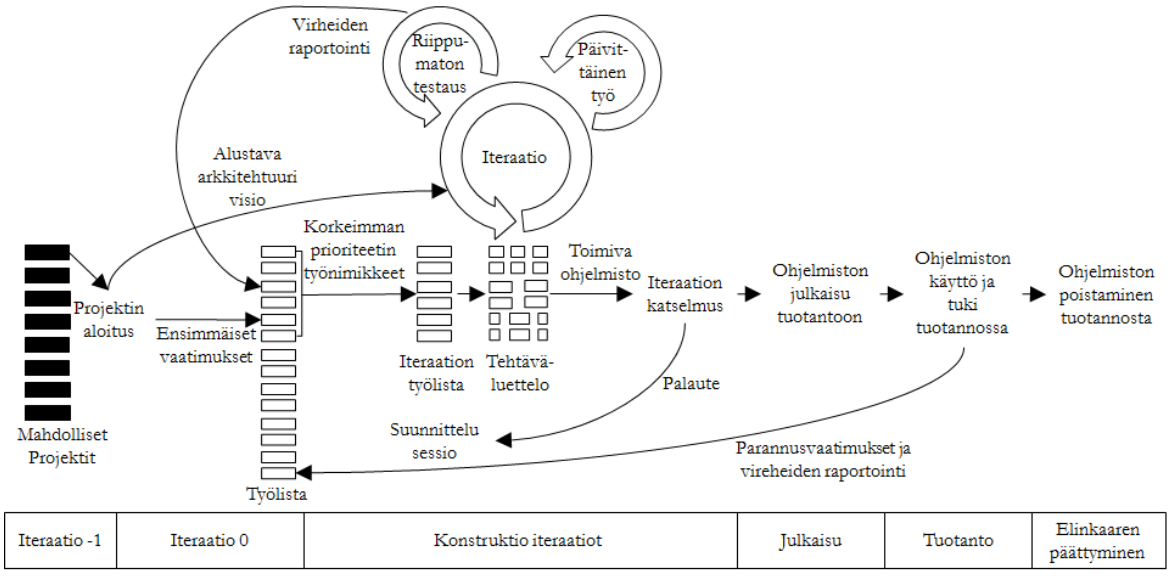
Walraet nostaa esille ohjelmistotuotannon sosiaalisen merkittävyyden, koska lopputulokset konkretisoituvat todellisessa maailmassa ja loppukäyttäjät käyttävät niitä organisaatioissa. Lopputulokset ovat näin ollen riippuvaisia ihmisten syötteistä tai ne ohjaavat ihmisten käyttäytymistä. (Walraet 1991, 30.)

Mielestäni Kotonyan ja Sommervillen näkemyksen mukaisesti on helppoa hahmottaa vaatimusten priorisoinnin suhde ohjelmistotuotantoon. Sen avulla ei ole kuitenkaan mahdollista suorittaa varsinaista priorisointia, sillä he eivät kuvaa, miten prioriteetit määritellään vaatimuksille eikä sitä miten vaatimuksille asetettuja prioriteetteja tulisi hyödyntää vaatimusten määrittelyssä. Mielestäni on hyvä, että he painottavat spiraalin hyödyntämistä vaatimusten esille tuonnissa, analysoinnissa ja neuvoteltaessa vaatimuksista kuvion 6 mukaisesti. Tämä mahdollistaa mielestäni sen, että myös uudet vaatimukset tulevat tarvittaessa huomioiduiksi siitä huolimatta, että valittuja vaatimuksia ollaan jo toteuttamassa. Käytännössä tämä on mielestäni hyvä esimerkki perinteisten vaihejakomallien haasteista erityisesti siinä tapauksessa, että koko ohjelmiston vaatimusmäärittely ja suunnittelu yritetään tehdä täydellisesti ennen ohjelmiston toteutusta. Em. spiraalien kytkeminen perinteisiin vaihejakomalleihin on ollut mielestäni lähtökohta ketterien menetelmien iteratiiviselle ajattelulle ja inkrementtien hyödyntämiselle.

2.2.2 Vaatimusten suhde ketteriin menetelmiin

Kuviossa 8 on esitelty ketterän kehityksen elinkaari Amblerin mukaan. Siinä elinkaari jakautuu kuuteen eri vaiheeseen, jotka ovat: iteraatio -1, iteraatio 0, konstruktioiteraatiot, julkaisu, tuo-

tanto ja elinkaaren päättyminen. Iteraatio -1 -vaiheessa potentiaalisia projekteja hallitaan koko projektisalkun näkökulmasta määrittelemällä liiketoimintamahdollisuudet, identifioimalla toteuttamiskelpoiset projektit ja arvioimalla soveltuvuus. Iteraatio 0-vaiheessa projekti käynnistetään keräämällä tarvittava tuki ja rahoitus, määrittelemällä alustava laajuus järjestelmälle sidosryhmien kanssa, kokoamalla tarvittava kehitystiimi, mallintamalla alustava tekninen arkkitehtuuri, asentamalla tarvittava kehitysympäristö ja arvioimalla projektin tarvitsemat resurssit. Varsinaiset konstruktioiteraatiot toteutetaan inkrementaalisti tiiviissä yhteistyössä sidosryhmien kanssa analysoimalla ja suunnittelemalla iteraatioita, varmistamalla laatu mm. koodaus konventioiden avulla, toimittamalla toimiva ohjelmisto jokaisen iteraation päätteeksi ja suorittamalla testausta koko konstruointivaiheen aikana. Halutut toiminnallisuudet implementoidaan prioriteettijärjestyksessä. Ketterän kehityksen periaatteiden mukaisesti vaatimuksiin voidaan kohdistaa vapaasti muutoksia kehitystiimien ulkopuolella niin, että ne vastaavat tarpeisiin. Sidosryhmille annetaan täysi kontrolli laajuuden, budjetin ja aikataulun suhteen. Julkaisuvaiheessa ohjelmisto siirretään tuotantoon suorittamalla tarvittavat systeemi- ja hyväksymistestit, muokkaamalla ohjelmistoa paremmaksi esim. korjaamalla löydetyt virheet, viimeistelemällä käyttäjädokumentaatio, kouluttamalla tarvittavat henkilöt ja julkaisemalla ohjelmisto mm. identifioimalla kohderyhmän erityisvaatimukset. Tuotantovaiheen tavoitteena on pitää ohjelmisto hyödyllisenä ja tuottavana julkaisun jälkeen tukemalla käyttäjiä. Ohjelmiston elinkaari päätetään hallitusti, kun toimittaja lopettaa käytössä olevan julkaisun tuen, ohjelmiston ei enää tarvitse tukea käyttäjäorganisaation liiketoimintamallia, ohjelmisto on tarpeeton, ohjelmisto on vanhentunut tai ohjelmisto halutaan korvata kokonaisuudessaan. (Ambler 2005.)



Kuvio 8. Ketterän kehityksen elinkaari (Ambler 2005).

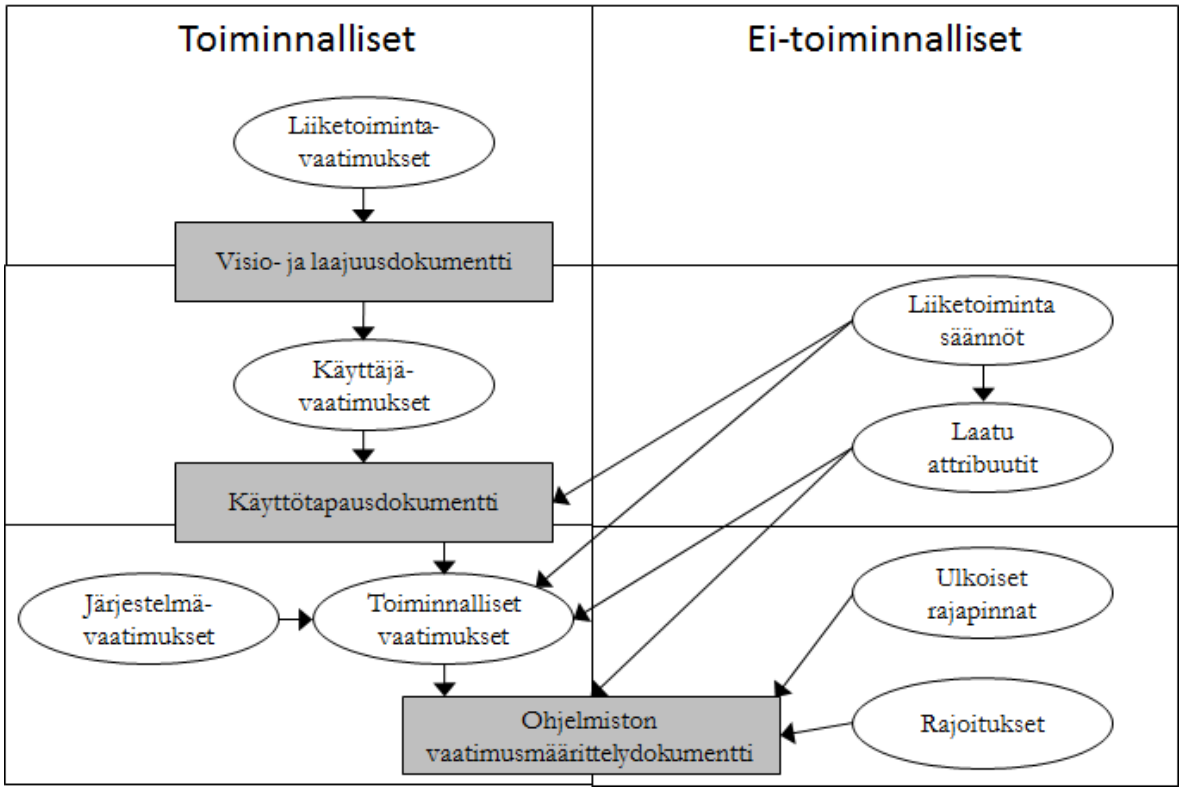
Steinberg ja Palmer esittävät, että ketterissä menetelmissä, kuten XP myös ohjelmistokehittäjä itse voi suorittaa vaatimusten priorisoinnin toimimalla sekä kehittäjän että asiakkaan roolissa. Aluksi hän arvioi kehittäjän roolissa vaatimusten työmäärät. Tämän jälkeen hän päättää asiakkaan roolissa tärkeimmät käyttäjätarinat loppukäyttäjän näkökulmasta. Tämän jälkeen hän analysoi kehittäjän roolissa kuinka monta käyttäjätarinaa hän ehtii toteuttamaan ennalta sovitussa iteraatiossa. Lopuksi ohjelmistokehittäjä valitsee vielä tärkeimmät iteraatioon puitteissa toteutettavat vaatimukset asiakkaan roolissa. (Steinberg & Palmer 2004, 29–30.)

Mielestäni kuviosta 8 voidaan helposti päätellä, että ketterissä menetelmissä priorisointi kohdistetaan ensin projekteille, jotka sisältävät laajempia vaatimuskokonaisuuksia. Tämän jälkeen vaatimuksia priorisoidaan koko ohjelmistotuotteen työlistalla, josta valikoituvat mukaan tietyt vaatimukset varsinaiseen konstruktioiteraatioon. Tämän lisäksi työlistaan kerätään parannusvaatimuksia, jotka ovat korjattaviksi tarkoitettuja virheitä tai olemassa olevien ominaisuuksien laajennoksia. Ketterän kehityksen elinkaari ei määrittele miten työlistalla olevia vaatimuksia pitäisi priorisoida. Mielestäni ketterän kehityksen elinkaari yrittää kuitenkin osoittaa, että työlistalla olevien työnimikkeiden tulisi olla priorisoituna koko ajan, jotta seuraava iteraatio voidaan käynnistää mahdollisimman tehokkaasti. Muussa tapauksessa kehitystiimien tehtäväluettelot saattavat olla tyhjiä pitkän aikaa ja se ei ole tarkoituksenmukaista.

Henkilökohtaisesti en pidä mahdollisena, että ohjelmistokehittäjä pystyy toimimaan ohjelmistokehittäjän, asiakkaan ja loppukäyttäjän roolissa sillä hän ei yleensä käytä eikä myy kehittämäänsä ohjelmistotuotetta.

2.2.3 Ohjelmistovaatimusten tasot

Wiegers jaottelee toiminnalliset ohjelmistovaatimukset kolmeen eri tasoon, jotka ovat liiketoiminta -, käyttäjä - ja toiminnalliset vaatimukset. Jaottelu on esitetty kuviossa 9. Liiketoimintavaatimukset ovat korkean tason vaatimuksia ja ne kuvaavat mm. tavoitteita, jotka halutaan saavuttaa. Käyttäjävaatimukset kuvaavat tehtäviä, jotka käyttäjän pitää pystyä suorittamaan tuotteella. Järjestelmävaatimukset kuvaavat eroteltavissa olevat loogiset kokonaisuudet, joita kutsutaan moduuleiksi. Toiminnalliset vaatimukset määrittävät toiminnallisuuden, jonka ohjelmistokehittäjät pystyvät toteuttamaan ohjelmistoon. Lisäksi Wiegers erittelee ns. ei-toiminnalliset vaatimukset, jotka sisältävät mm. yrityksen menettelytapoja, suorituskyky– ja laatutavoitteita. Em. vaatimukset on esitetty kuviossa ovaaleina ja suorakulmiot näyttävät tarkoituksenmukaisen dokumentin, mihin vaatimukset tallennetaan. (Wiegers 2003, 9.)



Kuvio 9. Vaatimusten väliset suhteen (Wiegers 2003, 8-11).

Mielestäni Wiegersin tapa kuvata vaatimusten väliset suhteet auttaa hahmottamaan eri tasoja, joissa vaatimuksia voidaan priorisoida. Lisäksi kuvaus on täysin linjassa Haikalan ja Märijärven kuvaaman vaihejakomallin kanssa, joka on esitelty luvussa 2.3. Toisaalta mielestäni olisi erittäin tärkeää, että myös ei-toiminnalliset laatuvaatimukset pystyttäisiin esittämään liiketoiminnalle toiminnallisina vaatimuksina. Tämän avulla olisi tehokkaampaa osoittaa liiketoiminnalle, etteivät esim. tietoturva ja suorituskyky synny ohjelmistoon automaattisesti. Lisäksi tämä mahdollistaisi sen, että liiketoiminta joutuisi itse tekemään päätöksiä mm. asetetuille laatuvaatimuksille, mikä toisi lisää läpinäkyvyyttä tuotekehitykselle.

Carlshamre ja Regnell käsittelevät artikkelissaan itsenäisesti kehitettyjä malleja, joiden avulla vaatimusten määrittely hallitaan markkinalähtöisesti. Markkinalähtöisen vaatimusmäärittelyn tavoitteena on hallita erityisesti jatkuvasti muuttuvia vaatimuksia. Tällöin painopiste on vaatimusten priorisoinnissa ja usein julkaistavissa inkrementaaleissa tuotejulkaisuissa. Markkinalähtöisessä vaatimusten määrittelyssä vaatimuksia edistetään elinkaarimallin avulla, jonka jokainen taso määrittelee tilan mille jokaisen vaatimuksen tulee nousta. Esimerkiksi Ericssonin käyttämä RDEM-elinkaarimalli määrittelee vaatimuksen edistymisen sen perusteella, mitä vaatimuksesta tiedetään. (Carlshamre & Regnell 2000, 961–962.)

2.2.4 Vaatimusten priorisointi

McConnell painottaa, että organisaatiot ovat päässeet eroon ohjelmistotuotteiden kehittämisongelmista, jos asiakassuuntautuneisuus on ollut ykkösprioriteetti. Hänen mukaansa ainoastaan asiakkaiden kokemalla kehittämisnopeudella on merkitystä sillä jos he eivät pidä tuotteesta he eivät maksa tuotteesta, jos he eivät maksa tuotteesta millään muulla ei ole enää mitään merkitystä. Kaikki muutkin ovat tyytyväisiä, jos asiakas on tyytyväinen. (McConnell 2002, 234.)

Haikala ja Märijärvi toteavat, että vaatimustenhallinnan perimmäinen tavoite on päätyä asiakasvaatimusten avulla asiakasvaatimukset täyttävään ohjelmistoon siitä huolimatta, mikä on käytössä oleva tapa vaiheistaa ohjelmistotyötä. He toteavat, että olennaisinta vaatimustenhallinnassa on varmistua siitä, että lopputuote vastaa asiakkaiden vaatimuksia ja, että lopputuotteessa on oltava ainoastaan kaikki halutut ominaisuudet. Lisäksi he painottavat, että vaatimusten priorisoinnin tarkoituksena on auttaa päätettäessä, mitkä ominaisuudet otetaan mukaan ohjelmiston seuraavaan versioon ja mitkä ominaisuudet toteutetaan myöhemmin. (Haikala & Märijärvi 2004, 91-96.)

Wiegersin mukaan asiakkaiden korkeat odotukset ja tiiviit aikataulut aiheuttavat sen, että julkaistavien tuotteiden pitää sisältää hyödyllisin toiminnallisuus niin aikaisessa vaiheessa, kuin mahdollista. Wiegers toteaa, että priorisoinnin avulla voidaan käsitellä vaatimuksia, jotka kilpailevat keskenään rajallisista resursseista. Hänen mukaansa suhteellisen prioriteetin laatiminen jokaiselle ohjelmiston kyvykkyydelle mahdollistaa suurimman arvon tuoton alhaisimmilla kustannuksilla. Lisäksi hän toteaa, että priorisointi on ratkaisevan tärkeää aikataulutetussa ja inkrementaalissa kehityksessä, joissa on kiinteät aikataulut. (Wiegers 2003, 248.)

Pressman toteaa, että vaatimusten priorisointi on tärkeää, koska yksittäiset asiakkaat väittävät yleensä juuri heidän vaatimuksien olevan välttämättömiä jostain tietystä näkökulmasta. Pressman painottaa eri sidosryhmien välistä vuorovaikutusta keskusteltaessa vaatimusten priorisoinnin aiheuttamista konflikteista. Hänen mukaansa keskustelussa esille tulevat mahdolliset riskit, kustannukset ja työmääräarviot saattavat muuttaa alkuperäisiä prioriteetteja. Pressmanin mukaan epäonnistumisen riski on erittäin suuri, jos eri sidosryhmät eivät ole samaa mieltä vaatimuksista. (Pressman 2000, 253-254.)

Sommerville ja Sawyer ovat samoilla linjoilla Pressmanin kanssa. Heidänkin mukaansa vaatimusten priorisointiin liittyvä keskustelu auttaa selvittämään erimielisyyksiä erityisesti silloin, jos vaatimusten prioriteetit on yhdistetty vaatimusten riskianalyysiin. (Sommerville & Sawyer

1997, 128.) He toteavat, että riskianalyysin suorittaminen vaatimuksille paljastaa usein puutteelliset vaatimukset, jolloin sidosryhmiä voidaan pyytää täydentämään vaatimuksia (Sommerville & Sawyer 1997, 137).

Ketterissä ja iteratiivisissa ohjelmistokehitysmenetelmissä, kuten Scrum vaatimukset priorisoidaan niin, että ne ovat valmiina tärkeysjärjestyksessä tuotteen keskeneräisten töiden työlistalla. Sellaiset vaatimukset, jotka tuottavat eniten arvoa ovat listalla korkeimmalla prioriteetilla. Koko ohjelmistotuotteen työlista on lähtökohta, kun uusi projekti käynnistyy ja silloin vaatimukset priorisoidaan vielä uudestaan. Muutokset tuotteen työlistalla heijastavat yleensä muutoksia liiketoiminnan esittämiin vaatimuksiin ja siihen, kuinka nopeasti vaatimukset saadaan kehitettyä toiminnallisuuksiksi. (Schwaber 2004, 8.)

Cohnin mukaan ketterät kehitystiimit keskittyvät liiketoiminnan prioriteetteihin. Hän toteaa, että se on saavutettavissa kahdella eri tavalla: 1) tuotteen omistaja priorisoi ominaisuudet julkaisuun optimoimalla ohjelmistotuoteyrityksen investoinnit projektiin. Hänen mukaansa tämä on saavutettavissa siten, että julkaisusuunnitelma luodaan tiimin kyvykkyyksien mukaan ja siten, että yksittäisten vaatimusten sisältämät tekniset riippuvuudet minimoidaan ja 2) kehitystiimit keskittyvät käyttäjien arvostamiin ominaisuuskokonaisuuksiin sen sijaan, että yrittäisivät saada valmiiksi yksittäisiä työtehtäviä. (Cohn 2006, 25-26.)

Ketterän kehityksen näkökulmasta Cohn painottaa myös, että julkaisua ei voida edes suunnitella, jos asiakas ei ole priorisoinut käyttäjäkertomuksia. Cohn jakaa kertomukset teknisiin kertomuksiin ja käyttäjäkertomuksiin. Lisäksi hän toteaa, mikäli teknisten ja käyttäjäkertomusten välillä on ristiriita käyttäjäkertomukset saavat aina korkeamman prioriteetin. (Cohn 2008, 98-99.)

Mielestäni priorisointiin käytettävien resurssien pitää olla linjassa priorisoinnilla saavutettaviin hyötyihin. Saavutettavia hyötyjä on vaikea mitata alkuvaiheessa ja tämän takia käytössä olevan priorisointimallin pitää olla niin joustava, että se voidaan sopeuttaa ja mukauttaa vastaamaan yrityksen tarpeita ketterästi. Jokin tietyn mallin mukainen vaiheistus vaatimusten priorisointiin pitää olla olemassa, jotta vaatimusten priorisointi saadaan kytkettyä mukaan yrityksen strategiaprosessiin. Tämän avulla priorisoinnista saadaan läpinäkyvä ja systemaattinen käytäntö yritykseen, mikä mahdollistaa lisäarvon ja paremman palvelun tuottamisen asiakkaille oikea-aikaisesti. Systemaattisen priorisointimallin avulla on helpompaa osallistaa tarkoituksenmukaisia sidosryhmiä vaatimusten priorisointiin. Mitä paremmin yritys pystyy osallistamaan sidosryhmiä, sitä tehokkaampia päätöksiä saadaan aikaiseksi.

2.2.5 Vaatimusten priorisointiin liittyvät haasteet

McConnell esittää, että ohjelmistotuotteiden muuttuvuus on yksi olennainen ongelma. Mitä suositumpi ohjelmistotuote on, sitä enemmän ihmiset käyttävät ohjelmistotuotetta ja löytävät sille uusia käyttötarkoituksia alkuperäisen käyttötarkoituksen ulkopuolelta. Ohjelmistotuotteesta tulee monimutkaisempi, kun sen ominaisuudet lisääntyvät ja sen pitää olla sopusoinnussa sille asetettujen rajoitusten kanssa. Mitä enemmän ohjelmistotuotetta mukautetaan eri käyttötarkoituksiin, sitä olennaisemmiksi muunnelmat muodostuvat. (McConnell 2004, 39.)

Charetten mukaan ohjelmistotuotteisiin liittyvät riskit voidaan jakaa kahteen luokkaan: 1) ohjelmistotuotteisiin sisältyvät riskit ja 2) tuotekehitysprosesseihin liittyvät riskit. Tuotekehitysprosesseihin liittyvistä riskeistä hän mainitsee mm. ohjelmistotuotteen lisääntyneen kysynnän ja monimutkaiset sekä kohtuuttomat vaatimukset. (Charette 1989, 73- 74.)

Lehtola luokittelee vaatimusten priorisointiin liittyvät käytännön ongelmat kolmeen eri luokkaan: 1) informaation yhdistämiseen liittyvät haasteet, 2) ihmisiin tai resursseihin liittyvät haasteet ja 3) kommunikaatioon tai kuvauksiin liittyvät haasteet. Informaation yhdistämiseen liittyvistä haasteista Lehtola toteaa priorisoinnin olevan monen eri näkökulman yhdistelmä mutta erittelee päänäkökulmiksi asiakas-, liiketoiminta- ja toteutusnäkökulmat. Ihmisiin ja resursseihin liittyvistä haasteista Lehtola nostaa esille eri sidosryhmien osallistamisen ja sidosryhmien tietojen yhdistämisen, vaatimusten priorisoinnin monitulkintaisena konseptina ja resurssien puutteen vaatimusten yksityiskohtaisessa ja systemaattisessa analysoinnissa. Kommunikaatioon ja kuvauksiin liittyvistä haasteista Lehtola painottaa tuotekehittäjien puutteellista tietämystä asiakkaiden todellisista käyttötarpeista, linkkien puuttumista tuotesuunnitelman sekä priorisoinnin eri tasojen väliltä ja tuotekehityksen tulevien kehitysaskeleiden kommunikointia asiakkaille. (Lehtola 2006, 33–37.)

Mielestäni vaatimusten priorisointi on vaikeaa sen takia, että siihen liittyy niin monta huomioitavaa asiaa, kuten eri näkökulmat, sidosryhmät, menetelmät, markkinat, asiakkaat, loppukäyttäjät ja yrityksen liiketoimintamalli. Lisäksi vaatimusten priorisoinnin pitäisi olla riittävän joustavaa, jolloin pitää huomioida myös ohjelmistotuotteen julkaisusykli, jotta vaatimukset priorisoidaan riittävän usein. Mielestäni on myös tärkeää ymmärtää, että vaatimusten priorisointi on osa tuotekehitysprosessia, jolloin priorisointiin liittyviä haasteita ja ongelmia voidaan käsitellä myös tuotekehitysprosessiin liittyvinä riskeinä.

2.2.6 Vaatimusten priorisointiin vaikuttavat tekijät

Cohn toteaa, että priorisoitaessa vaatimuksia tulisi huomioida ainakin uusien ominaisuuksien avulla saavutettava taloudellinen arvo eli kuinka paljon rahaa ohjelmistoyritys voi ansaista tai säästää, mikäli uudet ominaisuudet toteutetaan. Luonnollisesti myös kustannukset ovat ratkaiseva tekijä priorisoitaessa vaatimuksia. Näiden lisäksi Cohn suosittelee, että mahdollisten uusien teknologioiden käytöstä aiheutuva työkuorma ja uusien ominaisuuksien aiheuttamat riskit on huomioitava. (Cohn 2006, 79–85.)

Karlsson ja Ryan painottavat vaatimusten suhteellista arvoa ja toteutuskustannuksia kehittämässään Cost-Value lähestymistavassa. Heidän mukaansa ohjelmistotuotteen vaatimusten priorisoinnissa käytettävän prosessin pitää olla sekä yksinkertainen ja nopea että tuottaa tarkkoja sekä luotettavia tuloksia. Priorisointiprosessia ei todennäköisesti tulla käyttämään kaupallisten ohjelmistotuotteiden tuotekehityksessä, jos se ei täytä edellä mainittuja edellytyksiä. (Karlsson & Ryan 1997, 68.)

Wiegersin mukaan asiakkaat priorisoivat lähtökohtaisesti siitä näkökulmasta, kuinka arvokas kukin vaatimus on asiakkaalle itselleen. Vaatimus ei enää olekaan niin välttämätön asiakkaalle sen jälkeen, kun tuotekehittäjä osoittaa kustannukset, tekniset riskit ja muut mahdolliset kompromissit. Lisäksi tuotekehittäjät pystyvät määrittämään, että tietyt alemman prioriteetin toiminnallisuudet pitäisi toteuttaa kehityssyklin alussa, sillä niillä on voimakas vaikutussuhde tuotteen tekniseen arkkitehtuuriin. Wiegers toteaakin, että priorisointi on tasapainoilua uusien toiminnallisuuksien antamien liiketoimintahyötyjen ja toiminnallisuuksista aiheutuvien kustannuksien sekä seuraamuksien välillä, joita uudet toiminnallisuudet aiheuttavat ohjelmistotuotteen arkkitehtuuriperustan jatkokehitykselle. (Wiegers 2003, 20-24.)

Sommervillen ja Sawyerin mukaan vaatimukset pitäisi priorisoida ideaalitilanteessa jo vaatimusten hankintavaiheessa, jolloin eri sidosryhmien tulisi vain yksinkertaisesti päättää kuinka tärkeitä yksittäiset vaatimukset ovat. Heidän mielestään priorisointi tulisi pystyä tekemään luokittelemalla yksittäiset vaatimukset välttämättömiin, hyödyllisiin ja toivottaviin vaatimuksiin. Toisaalta he kuitenkin toteavat, että liiketoiminnan edustajat voivat priorisoida kaikki vaatimukset välttämättömiksi, jos he eivät tiedä esim. vaatimusten toteuttamisesta aiheutuvia kustannuksia ja työmääriä. (Sommerville & Sawyer 1997, 128–129.)

Mielestäni vaatimusten priorisointi on aina kompromissi eri sidosryhmien välillä, sillä priorisointi on aina jossain määrin subjektiivista. Vaatimusten prioriteetteja on mahdollista painottaa

eri näkökulmista mutta tällöinkin ajaudutaan yleensä tilanteeseen jossa painokertoimet on saatu aikaiseksi jonkinlaisen konsensuksen eli kompromissin tuloksena.

2.3 Aikaisempi tutkimus

Laura Lehtola tuo Teknilliselle Korkeakoululle tekemässään lisensiaatintutkimuksessa esille tärkeän lähtökohdan oman tutkimukseni näkökulmasta: kirjallisuudessa on tarjolla useita eri lähestymistapoja vaatimusten priorisointiin vaihdellen ylätason priorisointiprosessikuvauksista yksityiskohtaisiin priorisointialgoritmeihin. Lisäksi eri lähestymistavat käyttävät eri mitta-asteikkoja, keskittyvät eri näkökulmiin ja ovat eri kehitystasoilla. Vaatimusten priorisointia on tutkittu jo pitkään mutta käytettävissä ei ole vieläkään yksinkertaisia, tehokkaita ja teollisesti todistettuja menettelytapoja vaatimusten priorisointiin. (Lehtola 2006, 3.)

Berander, Khan ja Lehtola toteavat, että vaatimusten priorisointiin on käytössä suuri määrä erilaisia lähestymistapoja. Vaatimusten priorisoinnista on tehty useita empiirisiä tutkimuksia, mutta näyttö sopivimmasta menetelmästä tietyssä kontekstissa puuttuu, koska eri tutkimukset ovat päätyneet erilaisiin lopputuloksiin. Syyt erilaisiin lopputuloksiin selittyvät todennäköisesti sillä, että tutkimukset on suoritettu eri konteksteissa, eri muuttujilla ja erilaisilla arvojoukoilla, koska yhdenmukaista tutkimuskehystä ei ole ollut käytettävissä. Näin ollen eri tutkimuksia on vaikea vertailla keskenään, jotta pystyisi päättämään, milloin tietty priorisointimenetelmä olisi sopiva tietyssä kontekstissa. (Berander & Khan & Lehtola 2006, 39.)

Tutkimusaineistoa etsiessäni huomasin myös, että vaatimusten priorisointia käsittelevät tutkimukset keskittyvät pääasiassa vertailemaan olemassa olevia tekniikoita ja menetelmiä ainoastaan muutaman muuttujan suhteen kuten helppokäyttöisyyden, virheettömyyden ja skaalattavuuden näkökulmasta. Tämä johtuu todennäköisesti siitä, että rajaamalla tutkittavien muuttujien lukumäärä mahdollisimman pieneksi tutkimuksen laajuus pysyy hallittavissa ja toisaalta eri priorisointitekniikoiden ja -menetelmien välille on löydettävissä eroja helpommin. Toisaalta oli ilahduttavaa havaita, että tutkimuksissa yritettiin myös soveltaa useampaa eri tekniikkaa tai menetelmää riippuen mm. siitä millainen määrä priorisoitavia vaatimuksia oli käsiteltävänä analysoitaessa vaatimuksia. Olemassa olevien priorisointimenetelmien tai priorisointitekniikoiden omaksuminen ja niiden mukauttaminen edelleen kohdeyrityksen vaatimusten analysointiin oli myös tämän tutkimuksen lähtökohta.

3 Menetelmien hyödyntäminen vaatimusten priorisoinnissa

Kirjallisuudesta on eriteltävissä useita eri menetelmiä vaatimusten priorisointiin. Tässä kappaleessa käsitellään priorisointimenetelmien eroja ja niiden valintaan vaikuttavia tekijöitä.

Lehtolan mukaan priorisointimenetelmät voidaan jakaa karkealla tasolla kolmeen eri kategoriaan niiden käyttämien menettelytapojen perusteella: 1) ryhmittelymenetelmät, 2) näkökulmia yhdistävät menetelmät ja 3) äänestys- ja panostusmenetelmät. Ryhmittelymenetelmissä vaatimukset ryhmitellään yleensä tärkeyden tai kiireellisyyden perusteella. Näkökulmia yhdistävissä menetelmissä priorisointi pohjautuu eri näkökulmista arvioitujen tai laskettujen arvojen yhdistelmään. Panostusmenetelmät perustuvat ideaan, jonka mukaan jokaiseen tuotejulkaisuun on käytössä tietty määrä investoitavia resursseja, jotka määrittävät vaatimusten prioriteetit. Äänestysmenetelmää käytetään yleisesti, jos useiden eri sidosryhmien näkökulmat eroavat toisistaan olennaisesti, kuten tilanteessa jossa monta tuotepäällikköä ovat vastuussa eri asiakassegmenteistä. (Lehtola 2006, 13–15.)

3.1 Työnkulku

Liiketoiminnan edustajien kanssa järjestettävät priorisointisessiot sisältävät yleensä kolme vaihetta: 1) valmisteluvaiheessa vaatimukset listataan käytettävän menetelmän periaatteiden mukaisesti ja sovitaan toteutusvaiheessa käytettävät arviointikriteerit, 2) toteutusvaiheessa vaatimukset priorisoidaan edellisessä vaiheessa annettujen tietojen perusteella ja 3) esittelyvaiheessa esitellään toteutusvaiheen tulokset, jotka saadaan selville käytettävän menetelmän määrittelemien sääntöjen, kuten laskentasääntöjen avulla. (Karlsson & Wohlin & Regnell 1998, 940.)

3.2 Näkökulmat

Aurumin & Wohlinin mukaan vaatimuksia on mahdollista priorisoida monesta eri näkökulmasta. Näkökulma on yksi projektin tai vaatimuksen ominaisuus, jota voidaan käyttää vaatimusten priorisoinnissa. He erittelevät mm. tärkeys-, rangaistus-, kustannus-, aika-, riski- ja vaihtelevuusnäkökulmat. He toteavat, että priorisointi on suhteellisen helppoa yhdestä näkökulmasta mutta useamman näkökulman yhdistäminen tekee priorisoinnista monimutkaisempaa. (Aurum & Wohlin, 72.)

Tärkeyttä priorisoitaessa sidosryhmien pitäisi priorisoida ne vaatimukset, jotka ovat tärkeimpiä tuotteessa mutta se ei ole yksinkertaista sillä eri sidosryhmät lähestyvät tärkeyttä omasta näkökulmasta. Rangaistusseuraamuksia voidaan tarkastella esim. silloin, jos jotain vaatimusta ei ole

pystytty toteuttamaan tai tuotteesta puuttuu itsestään selviä ominaisuuksia, jotka tekevät siitä sopimattoman markkinoille. Kustannuksia arvioidaan usein henkilötyötunteina, koska työtunnit aiheuttavat yleensä suurimmat kustannukset ohjelmistokehityksessä. Aika kytketään usein yhteen kustannusten kanssa mutta lisäksi tarvittavaa aikaa voidaan tarkastella esim. lisäkoulutustarpeiden näkökulmasta jne. Vaatimusten toteuttamiseen liittyviä riskejä arvioitaessa pitäisi ottaa kantaa sekä riskien vaikutuksiin että niiden todennäköisyyteen, sillä se mahdollistaa riskitason laskemisen koko projektille. Vaatimusten vaihtelevuutta arvioitaessa pitäisi kiinnittää huomioita mm. mahdollisiin markkinoiden muutoksiin ja liiketoimintavaatimusten muutoksiin jo suunnitteluvaiheessa. (Aurum & Wohlin 2005, 72–74.)

3.3 Mitta-asteikot

Käytettävä mitta-asteikko kuvaa sitä asteikkoa, jonka mukaisesti vaatimukset voidaan priorisoida. Käytettävällä mitta-asteikoilla on suuri merkitys, kun vaatimukset sijoitetaan paremmuusjärjestykseen. Mitä vahvempi asteikko on käytössä, sitä hyödyllisempi arviointi vaatimuksille voidaan suorittaa. Mitta-asteikot heikoimmasta vahvimpaan ovat: nominaali- eli luokitusasteikko, ordinaali- eli järjestysasteikko, intervalli- eli välimatka-asteikko ja suhdeasteikko. (Karlsson ym. 1998, 943.)

Luokitteluasteikko on yksinkertaisin tapa osoittaa järjestys, koska siinä hyödynnetään jotain ominaisuutta, joka jakaa mitattavat yksiköt eri luokkiin. Järjestysasteikon avulla mitattavat yksiköt laitetaan jonkin ominaisuuden mukaiseen järjestykseen. Välimatka-asteikolla kuvataan luokkien välisiä välimatkoja, mutta se ei ota kantaa luokkien välisiin suhteisiin. Suhdeasteikko on kaikista voimakkain, koska sillä voidaan ilmaista mitattavien samanarvoisuus, järjestys, välimatka ja suhde. Suhdeasteikossa muuttujan arvojen suhde pysyy samana, vaikka mittayksikkö muuttuu esim. senttimetreistä metreiksi. (Stevens 1946, 678–680.)

3.4 Vaatimusten priorisointimenetelmiä

Tässä alaluvussa on esitelty priorisointimenetelmiä, joiden avulla vaatimukset voidaan priorisoida edellä esiteltyjen mitta-asteikkojen mukaisesti.

3.4.1 Prioriteettiluokittelu

IEEE:n määrittelemässä standardissa ohjelmistovaatimukset priorisoidaan tarpeellisuuden mukaisiin luokkiin. Luokkia on määritelty kolme ja ne kuvattu kuviossa 10. (IEEE Std 830 Recommended Practice for Software Requirements Specifications 1998, 7.)

Luokka	Kuvaus
Välttämätön	Ohjelmisto ei ole hyväksyttävissä, elleivät välttämättömät vaatimukset ole toteutettu sovitulla tavalla.
Ehdollinen	Parantaisivat ohjelmistoa mutta ohjelmisto on hyväksyttävissä, vaikka ehdollisia vaatimuksia ei ole toteutettu.
Valinnainen	Ryhmä toimintoja, jotka voivat olla hyödyllisiä tai ei. Antaa ohjelmiston toimittajalle mahdollisuuden ehdottaa toteutettavaksi toimintoja, jotka ylittävät jollain tavoin asetetut vaatimukset.

Kuvio 10. Vaatimusten tarpeellisuus (IEEE Std 830 1998, 7).

Aurumin & Wohlinin mukaan luokittelu tai ryhmittely on yleisimmin käytetty priorisointimenetelmä. Luokkien lukumäärä voi vaihdella, mutta kolmen eri luokan käyttö on yleisintä. Luokittelun tuloksena vaatimukset saadaan järjestysasteikon mukaiseen järjestykseen luokittain. Yksittäisiä vaatimuksia ei priorisoida luokkien sisällä. (Aurum & Wohlin 2005, 76–77.)

Wiegers toteaa, että luokittelu kolmen eri prioriteettiluokan avulla on subjektiivista ja epätarkkaa. Hänen mukaansa luokkien nimeämisellä ei ole juurikaan merkitystä, vaan ne koetaan aina korkean, keskitason tai matalan prioriteetin luokiksi. Tämän takia on tärkeää, että sidosryhmät sopivat etukäteen mitä luokittelussa käytettävän järjestysasteikon tasot tarkoittavat. (Wiegers 2003, 250–251.)

Karlsson ym. painottavat, että ohjelmistokehitysprojektit sisältävät yleensä kokonaisuuksia, joiden merkitys eroaa olennaisesti toisistaan. Ajan ja resurssien säästämiseksi on tällöin mahdollista käyttää esim. karkeaa luokittelua. Karkean luokittelun jälkeen tarkemman tason priorisoinnin voi tehdä jollain toisella menetelmällä. Luokittelun ja ryhmittelyn tärkein hyöty on se, että kahden eri luokan sisältämiä vaatimuksia ei tarvitse enää vertailla keskenään. (Karlsson ym. 1998, 942.)

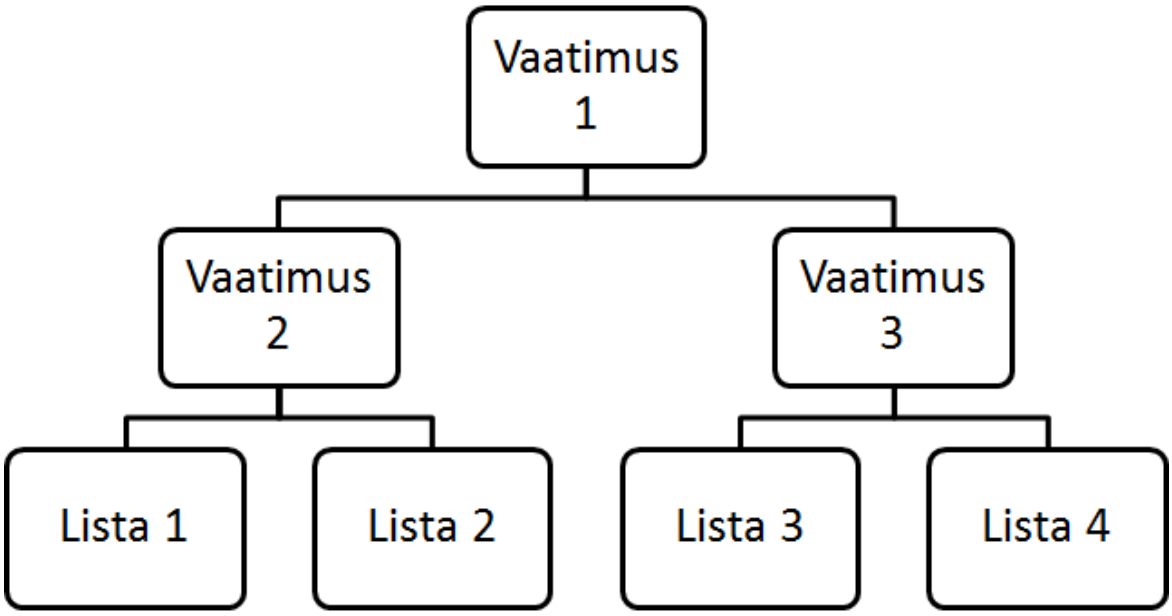
Mielestäni prioriteettiluokittelu on yksinkertainen ja sitä on helppo käyttää, koska se perustuu loogiseen ajatteluun. Pelkkä luokkien nimeäminen ei kuitenkaan yleensä riitä hyvään lopputulokseen, vaan luokkien käytölle pitää määritellä tarkemmat suodattimet, joita voi käyttää apuna. Mielestäni on olennaista, että prioriteettiluokittelu priorisoi vaatimukset järjestysasteikon mukaiseen järjestykseen eli se ei ota millään tavalla kantaa yksittäisten vaatimusten välisiin riippuvuuksiin. Tämä saattaa aiheuttaa mm. sen, että ainoastaan tärkeimpiä vaatimuksia toteutetaan eikä vaatimuskokonaisuuksia käsitellä millään tavoin. Toisaalta prioriteettiluokittelussa on suh-

teellisen todennäköistä ajautua tilanteeseen, jossa suurin osa vaatimuksista on priorisoitu korkeimmalle prioriteetille juuri sen takia, että vaatimuksien prioriteetteja tarkastellaan aina yhden vaatimuksen näkökulmasta.

Prioriteettiluokittelussa jokainen vaatimus arvioidaan yhden kerran eli 20 vaatimusta saadaan priorisoitua luokkien mukaiseen järjestykseen 20 arvioinnin tuloksena.

3.4.2 Binary Search Tree (BST)

Puolitushaku (Binary Search) on yleisesti käytetty algoritmi tiedon haussa ja lajittelussa. Puolitushakua voidaan hyödyntää myös ohjelmistovaatimusten priorisoinnissa. Kuviossa 11 on kuvattu kolmen vaatimuksen lista, joista jokainen sisältää alilistoja, jotka sisältävät lisää vaatimuksia. Puolitushaun rakenteen mukaisesti oikealla olevat vaatimukset ovat tärkeämpiä, kuin vasemmalla olevat vaatimukset. Näin olleen Vaatimus 1:n prioriteetti on pienempi, kuin Vaatimus 3:n, mutta korkeampi, kuin Vaatimus 2:n. (Bebensee & Weerd & Brinkkemper 2010, 70.)



Kuvio 11. Puolitushaun systemaattinen rakenne (Bebensee ym. 2010, 70).

Vaatimukset priorisoidaan BST:n avulla seuraavasti:

1. Kaikki vaatimukset kerätään yhteen pinoon.
2. Valitaan yksi vaatimus juurinoodiksi.
3. Valitaan toinen vaatimus ja verrataan sitä juurinoodin vaatimukseen.

4. Vaatimus rinnastetaan juurinoodin vasemmalle puolelle, jos toinen vaatimus on vähemmän tärkeä, kuin juurinoodissa oleva vaatimus. Vaatimus rinnastetaan juurinoodin oikealle puolelle, jos toinen vaatimus on tärkeämpi, kuin juurinoodissa oleva vaatimus. Uusi vaatimus asetetaan uudeksi lapsinoodiksi vasemmalle tai oikealle puolelle riippuen sen prioriteetista, jos noodissa ei ole tarkoituksenmukaista lapsinoodia. Vaiheet kolme ja neljä toistetaan, kunnes kaikki vaatimukset on vertailtu ja sijoitettu puurakenteeseen.
5. Puurakennetta voidaan hyödyntää niin, että tärkeimmät vaatimukset ovat listojen alussa ja vähemmän tärkeät listojen lopussa. (Racheva & Daneva & Buglione 2008, 52–53.)

Bellin ja Guptan mukaan BST on yleisesti käytössä oleva tietorakenne tietueiden tallettamisessa keskusmuistiin ja noutamisessa keskusmuistista, koska se takaa logaritmisen käsittelyn eri operaatioille, niin kauan kuin BST-puurakenne on tasapainoinen. BST-puun tasapainotusta on tutkittu pitkään ja koko puu on mahdollista tasapainottaa erilaisilla tekniikoilla yhdellä kerralla. (Bell & Gupta 1993, 369.)

Jokaisen vaatimuksen lisääminen BST:hen vaatii keskimäärin *O(n log n)* vertailua, missä n on vaatimusten lukumäärä. Vertailun tuloksena vaatimukset saadaan priorisoitua järjestysasteikolle. BST:ssä juurinoden keskimääräinen etäisyys puussa olevasta lehdestä on *O(log n)*, joten uuden vaatimuksen lisääminen vaatii keskimäärin *O(log n)* vertailua. (Karlsson ym. 1998, 942.)

BST:n avulla 20 vaatimusta saataisiin priorisoitua noin 87 vertailulla (*O(20 log 20)*). Yhden uuden vaatimuksen lisääminen em. puuhun vaatisi ainoastaan neljä vertailua (*O(log 21)*)

Mielestäni on tärkeää huomioida, että myös BST:ssä vaatimukset priorisoidaan järjestysasteikon mukaiselle mitta-asteikolle. Vaatimuksia kyllä vertaillaan keskenään, mutta jokaista vaatimusta ei tarvitse verrata keskenään, jotta vaatimukset saadaan sijoitettua puuhun puolitushaun systemaattisen rakenteen mukaiseen järjestykseen. BST:n logiikka on mielestäni erittäin helppo ymmärtää ja siitä on varmasti apua, kun pitää käsitellä suuria määriä vaatimuksia. Lisäksi mielestäni on olennaista huomata, että BST on erittäin tehokas ainoataan silloin, kun puurakenne pidetään tasapainossa. Tämä mahdollistaa uuden vaatimuksen priorisoinnin jo olemassa olevien vaatimusten suhteen tehokkaasti.

3.4.3 Wiegersin menetelmä

Wiegersin menetelmässä vaatimukset priorisoidaan jakamalla vaatimusten liiketoiminnallinen arvo vaatimusten toteuttamisesta aiheutuvien kustannusten ja riskien summalla. Liiketoiminnan edustajat arvioivat vaatimusten liiketoiminnallisen arvon sekä asiakkaille syntyvänä lisäar-

vona että haittavaikutuksina, jotka aiheutuvat, mikäli vaatimuksia ei toteuteta. Tuotekehittäjät arvioivat vaatimusten toteuttamisesta aiheutuvat kustannukset ja riskit. Kaikki arvioinnit suoritetaan välimatka-asteikolla yhden vaatimuksen suhteen. (Wiegers 2003, 252–256.)

Wiegers toteaa, että menetelmää tulisi hyödyntää ainoastaan valinnaisten vaatimusten kanssa, jotka eivät ole lähtökohtaisesti kaikista korkeimmalla prioriteetilla. Liiketoiminnan näkökulmasta kriittisimmät vaatimukset pitää identifioida ensin toteutettaviksi, jonka jälkeen menetelmän avulla voidaan priorisoida ns. valinnaiset vaatimukset, joista voidaan neuvotella. (Wiegers 2003, 253.)

Wiegersin menetelmällä vaatimukset voidaan priorisoida taulukossa 1 kuvatun matriisin mukaisesti kahdeksan eri vaiheen avulla. Menetelmän pelkistetty vaiheistus on seuraava:

1. Kaikki samalla abstraktiotasolla olevat vaatimukset listataan matriisiin.
2. Liiketoiminta arvioi vaatimusten suhteelliset hyödyt asiakkaalle tai liiketoiminnalle asteikolla yhdestä yhdeksään, jos vaatimukset toteutetaan. Yksi osoittaa pienintä hyötyä ja yhdeksän suurinta mahdollista hyötyä.
3. Liiketoiminta arvioi vaatimusten suhteelliset haitat asiakkaalle tai liiketoiminnalle asteikolla yhdestä yhdeksään, jos vaatimuksia ei toteuteta. Yksi osoittaa pienintä haittaa ja yhdeksän suurinta mahdollista haittaa.
4. Suhteellisten hyötyjen ja suhteellisten haittojen painoarvoja muokataan tarvittaessa.
5. Tuotekehitys arvioi vaatimusten suhteelliset toteutuskustannukset asteikolla yhdestä yhdeksään. Yksi osoittaa pienintä kustannusta ja yhdeksän suurinta mahdollista kustannusta.
6. Tuotekehitys arvioi vaatimusten suhteelliset toteutusriskit asteikolla yhdestä yhdeksään. Yksi osoittaa pienintä riskiä ja yhdeksän suurinta mahdollista riskiä.
7. Suhteellisten kustannusten ja suhteellisten riskien painoarvoja muokataan tarvittaessa.
8. Varsinainen prioriteetti lasketaan kaavalla:
   *Arvo % / (Kustannus % * Kustannuksen painoarvo + Riski % * Riskin painoarvo).*
   Matriisi järjestetään prioriteetin mukaan alenevassa järjestyksessä, jolloin ylimpänä olevalla vaatimuksella on suurin toteutusprioriteetti. (Wiegers 1999.)

Taulukko 1. Esimerkki Wiegersin priorisointimatriisista.

Suhteellinen painoarvo:	1	1			1		1		
	Suhteellinen hyöty	Suhteellinen haitta	Yhteensä	Arvo %	Suhteellinen kustannus	Kustannus %	Suhteellinen riski	Riski %	Prioriteetti
Vaatimus 3	7	6	13	30,2	2	12,5	2	12,5	1,21
Vaatimus 2	2	1	3	7,0	1	6,3	1	6,3	0,56
Vaatimus 1	9	8	17	39,5	8	50,0	8	50,0	0,40
Vaatimus 4	5	5	10	23,3	5	31,3	5	31,3	0,37
Yhteensä	23	20	43	100,0	16	100,0	16	100,0	2,53

Wiegers itse painottaa, että menetelmän tarkkuus perustuu tiimin kykyyn arvioida yksittäisten vaatimusten hyötyjä, kustannuksia ja riskejä. Näin ollen hän suosittelee, että laskennan tuottamaa prioriteettijärjestystä käytetään suosituksena, jota voidaan hyödyntää liiketoiminnan ja tuotekehityksen välisessä kommunikaatiossa. Prioriteettiarvojen pienet erot eri vaatimusten välillä eivät ole merkityksellisiä, sillä lähes samanarvoisia vaatimuksia tulisi tarkastella kokonaisuuksina. Lisäksi hän toteaa, ettei hänen semikvantitatiivinen menetelmä ole matemaattisesti tarkka menetelmä. (Wiegers 2003, 257.)

Lehtola ja Kauppinen toteavat empiirisessä tutkimuksessaan, että lähtökohtaisesti Wiegersin menetelmän käyttäjät olivat innoissaan menetelmästä, koska sen avulla vaatimukset pystyttiin priorisoimaan monesta eri näkökulmasta kontrolloidusti. Toisaalta he nostavat esille useita menetelmässä havaitsemiaan haasteita kuten: 1) käyttäjille oli epäselvää, mihin arkielämän tietoon arvioitavien muuttujien pitäisi pohjautua, 2) käyttäjille oli vaikeaa hahmottaa, mikä oli arvoasteikon arvojen todellinen ero, 3) käyttäjät kehittivät omia kaavoja menetelmän rinnalle, jotta pystyivät suorittamaan arvioinnin ja 4) käyttäjät muuttivat arvioita, jos priorisointijärjestys oli heidän mielestään väärä. (Lehtola & Kauppinen 2004, 167–168.)

Wiegersin menetelmässä jokaiselle yksittäiselle vaatimukselle suoritetaan neljä arviointia, jotta vaatimuksen hyödyille, haitoille, riskeille ja kustannuksille saadaan vertailuluku. Näin ollen 20 vaatimusta saadaan priorisoitua 80 arvioinnin tuloksena.

Mielestäni on hyvä, että Wiegers itse jaottelee vaatimukset lähtökohtaisesti liiketoimintakriittisiin vaatimuksiin, jotka pitää aina toteuttaa ja valinnaisiin vaatimuksiin, joista voidaan neuvotella. Toisaalta itse menetelmä ei korosta em. jaottelua, vaikka sillä on olennainen merkitys lopputuloksen kannalta. Lisäksi menetelmällä arvioitavat vaatimukset pitäisi olla samalla abstraktiotasolla keskenään, mutta yleensä liiketoiminta on kiinnostunut vain kriittisimmistä vaatimuksista, jotka siis pitäisi jättää tämän menetelmän ulkopuolelle. Jäljelle jäävät näin ollen neuvoteltavissa olevat pienemmät vaatimukset, jotka eivät yleensä ole niin merkityksellisiä liiketoiminnalle, eli liiketoiminnan edustajia saattaa olla hieman vaikeaa saada osallistumaan menetelmän käyttöön, jos heidän osoittamansa kriittiset vaatimukset ovat jo toteutuksessa. Mielestäni itse menetelmä on hieman ristiriitainen ja hieman monimutkainen. Toisaalta uskon, että eri sidosryhmien edustajien olisi aina hyvä miettiä ohjelmistotuotteille esitettyjä vaatimuksia Wiegersin menetelmän mukaisilla muuttujilla. Menetelmässä on ainoastaan neljä muuttujaa, jos painoarvoja ei oteta huomioon. Muuttujien systemaattinen hyödyntäminen liiketoiminnan ja tuotekehityksen välisessä kommunikaatiossa olisi varmasti helppoa ja tehokasta, koska muuttujat ovat loogisia ja niitä on vähän.

3.4.4 Analytical Hierarchy Process (AHP)

Analyyttinen hierarkiaprosessi (AHP) on päätöksenteko menetelmä. AHP:n avulla kaikki mahdolliset vaatimusparit vertaillaan keskenään. Vaatimusparien vertailussa arvioidaan vaatimusten keskinäistä tärkeyttä ja tärkeyden laajuutta. AHP on vaativa menetelmä, jos vaatimuksia on paljon, koska vertailtavia vaatimuspareja on paljon. Toisaalta AHP on erittäin luotettava menetelmä, koska se sisältää päällekkäisyystarkistuksia, jotka indikoivat arviointiin liittyviä epäjohdonmukaisuuksia. Lisäksi AHP:n avulla vaatimukset saadaan priorisoitua suhdeasteikon mukaisesti. (Karlsson ym. 1998, 943.)

AHP sisältää neljä vaihetta, joiden avulla vaatimukset saadaan priorisoitua. Menetelmän pelkistetty vaiheistus on seuraava:

1. Vaatimukset asetetaan taulukon 2 mukaiseen matriisin riveiksi ja sarakkeiksi.

Taulukko 2. Vaatimusten vertailu AHP:llä (Karlsson & Ryan 1997, 69).

	Vaatimus 1	Vaatimus 2	Vaatimus 3	Vaatimus 4
Vaatimus 1	=1	=1/3	=2	=4
Vaatimus 2	=3	=1	=5	=3
Vaatimus 3	=1/2	=1/5	=1	=1/3
Vaatimus 4	=1/4	=1/3	=3	=1

2. Suoritetaan vaatimusten pareittainen vertailu taulukossa 3 esitetyn asteikon mukaisesti arvioimalla vaakarivillä olevan vaatimuksen arvoa pystyrivillä olevaan vaatimukseen. Leikkauskohtaan merkitään murtoluku, jos vaakarivillä olevan vaatimuksen arvo on vähäisempi, kuin pystyrivillä olevan vaatimuksen. Vaatimus on itsensä suhteen samanarvoinen eli arvo on yksi. Arvioituja vaatimuspareja ei arvioida kahteen kertaan, vaan leikkauskohdissa käytetään käänteislukua. Näin ollen esim. vaakarivillä oleva Vaatimus 1 arvioidaan ensiksi Vaatimus 2 kanssa, jonka jälkeen vaakarivillä oleva Vaatimus 2 arvoksi merkitään jo suoritetun arvioinnin käänteisarvo. Edellä mainittujen sääntöjen perusteella vertailuja täytyy suorittaa $n*(n-1)/2$ kertaa, jotta kaikki vertailut on suoritettu.

Taulukko 3. Pareittaisen vertailun mitta-asteikko (Karlsson & Ryan 1997, 70).

Suhteellinen intensiteetti	Määritelmä	Selitys
1	Samanarvoiset	Kaksi vaatimusta ovat samanarvoisia
3	Hieman arvokkaampi	Kokemus puoltaa hieman enemmän toista vaatimusta
5	Olennaisesti tai vahvasti arvokkaampi	Kokemus puoltaa vahvasti toista vaatimusta
7	Erittäin vahvasti arvokkaapi	Toista vaatimusta puolletaan vahvasti ja sen paremmuus toiseen verrattuna on todistettu käytännössä
9	Äärimmäisesti arvokkaampi	Näyttö toisen vaatimuksen puolesta on korkein mahdollinen aineiston perusteella
2, 4, 6, 8	Arviointien välissä olevat arvot	Voidaan käyttää kompromissitilanteissa

3. Lasketaan ominaisarvot, jotka esittävät taulukon 4 mukaisia vaatimusten vertailuarvoja. Vaatimusten vertailuarvot saadaan laskemalla sarakkeiden arvot yhteen ja tämän jälkeen jokaisessa solussa oleva arvo jaetaan oman sarakkeen yhteenlasketulla arvolla. Lisäksi summataan vaakarivien arvot. Tämän jälkeen normalisoidaan rivien summat jakamalla jokaisen rivin summa vaatimusten lukumäärällä:

$$\frac{1}{4} * \begin{pmatrix} 1.05 \\ 1.98 \\ 0.34 \\ 0.62 \end{pmatrix} = \begin{pmatrix} 0.26 \\ 0.50 \\ 0.09 \\ 0.16 \end{pmatrix}$$

Taulukko 4. Vaatimusten vertailuarvot.

	Vaatimus 1	Vaatimus 2	Vaatimus 3	Vaatimus 4	Summa	Arvo
Vaatimus 1	0,21	0,18	0,18	0,48	1,05	26 %
Vaatimus 2	0,63	0,54	0,45	0,36	1,98	50 %
Vaatimus 3	0,11	0,11	0,09	0,04	0,34	9 %
Vaatimus 4	0,05	0,18	0,27	0,12	0,62	16 %
	100 %	100 %	100 %	100 %	4	100 %

4. Asetetaan jokaiselle vaatimukselle suhteellinen arvo ominaisarvon perusteella:
   - Vaatimus 1 käsittää 26 % vaatimusten kokonaisarvosta.
   - Vaatimus 2 käsittää 50 % vaatimusten kokonaisarvosta.
   - Vaatimus 3 käsittää 9 % vaatimusten kokonaisarvosta.
   - Vaatimus 4 käsittää 16 % vaatimusten kokonaisarvosta. (Karlsson & Ryan 1997, 69–70.)

Vaatimusten ominaisarvot ovat täysin eheät, jos kaikille vaatimuksille pystytään määrittelemään suhteelliset arvot täsmällisesti. Arvioinnin tuloksen tarkkuus laskee, jos suhteelliset arvot sisäl-

tävät epäjohdonmukaisuuksia esim. silloin, jos Vaatimus 1 on erittäin vahvasti arvokkaampi, kuin Vaatimus 2, Vaatimus 2 on hieman arvokkaamp,i kuin Vaatimus 3 ja Vaatimus 3 on hieman arvokkaampi, kuin Vaatimus 1. Pareittaisen vertailun sisältämän redundanssin takia AHP ei ole niin herkkä päätöksien sisältämille virheille. Lisäksi AHP mahdollistaa päätöksiin liittyvien virheiden mittaamisen vertailumatriisin konsistenssiluvun avulla, jota voidaan käyttää konsistenssisuhteen laskemisessa. (Karlsson & Ryan 1997, 69.)

Analyyttisen hierarkiaprosessin avulla 20 vaatimusta saataisiin priorisoitua 190 vertailun avulla (*n*(n-1)/2*).

Analyyttinen hierarkiaprosessi on mielestäni erittäin tarkka priorisointimenetelmä, mutta sen käyttäminen vaatimusten priorisoinnissa edellä esitetyllä tavalla on hankalaa monimutkaisten sääntöjen takia. Lisäksi sen käyttäminen vaatii erittäin suuren määrän vertailuja, jos vertailtavia vaatimuksia on paljon. Toisaalta vaatimusparien suhteellinen priorisointi helpottaa mielestäni priorisointia, koska ensin valitaan arvokkaampi vaatimus, jonka jälkeen arvioidaan suhteellinen arvo. AHP soveltuu mielestäni vaatimusten priorisointiin erittäin hyvin silloin, kun vaatimusten tai vaatimuskokonaisuuksien lukumäärä on pieni ja sidosryhmiä on paljon. Tällöin esim. jokainen sidosryhmä, kuten liiketoimintayksikkö, voi suorittaa priorisoinnin itsenäisesti ja vaatimukset saadaan priorisoitua suhdeasteikon mukaisesti tärkeysjärjestykseen laskemalla tulokset yhteen.

Priorisointimenetelmien tarkempi vertailu on esitetty tämän tutkimuksen liitteissä. Liitteessä 1 on esitelty priorisointimenetelmien yleiskuvaus. Liitteessä 2 esitellään priorisointimenetelmien helppous ja liitteessä 3 esitellään priorisointimenetelmien käytettävyys.

4 Olettamukset

Tässä kappaleessa käsitellään olennaisimmat olettamukset, jotka liittyvät kappaleissa kaksi ja kolme esitettyihin teoriakartoituksiin.

Vaatimusten priorisoinnin tavoitteena on saada aikaiseksi asiakkaiden vaatimukset täyttävä ohjelmisto riippumatta siitä, miten ohjelmistokehitys on vaiheistettu. Olennaisinta on varmistua siitä, että ohjelmistotuote vastaa asiakkaiden vaatimuksia ja että seuraavassa ohjelmistojulkaisussa on ainoastaan kaikki halutut ominaisuudet, joilla voidaan tuottaa arvoa ohjelmistotuotteen eri sidosryhmille.

Ohjelmistotuotannon tärkein vaihe on vaatimusten analysointi, koska siinä selvitetään, mitä ohjelmistotuotteen täytyy tehdä. Vaatimusten priorisointiprosessin avulla varmistetaan, että kaikki sidosryhmät pystyvät kertomaan oman näkökulmansa, vaatimukset saadaan priorisoitua ja hyväksytään toteutettavat vaatimukset.

Vaatimusten priorisointi kansainvälisessä ohjelmistotuoteyrityksessä on haastavaa, sillä loppukäyttäjät ovat yleensä löytäneet ohjelmistotuotteille uusia käyttötarkoituksia alkuperäisten käyttötarkoitusten ulkopuolelta. Näin ollen vaatimusten priorisointi on tasapainoilua uusien toiminnallisuuksien antamien liiketoimintahyötyjen ja toiminnallisuuksista aiheutuvien kustannuksien sekä seuraamuksien välillä.

Ohjelmistotuotteen vaatimusten priorisoinnissa käytettävän prosessin pitää olla sekä yksinkertainen ja nopea että tuottaa riittävän tarkkoja sekä tarkoituksenmukaisia tuloksia. Priorisointiprosessissa voidaan hyödyntää olemassa olevia priorisointimenetelmiä. Priorisointimenetelmät voidaan jakaa eri kategorioihin niiden käyttämien menettelytapojen perusteella. Priorisointimenetelmän käyttämällä mitta-asteikoilla on suuri merkitys, kun vaatimukset sijoitetaan paremmuusjärjestykseen.

5 Konstruktiivisen tutkimuksen tausta ja lähtötila

Ohjelmistotuotannon vaatimusten hallinnan näkökulmasta kohdeyritys halusi selvittää voidaanko ohjelmistotuotteille esitetyt vaatimukset priorisoida systemaattisemmin ja läpinäkyvämmin suhteessa sen hetkisiin käytäntöihin. Vaatimuksia haluttiin priorisoida systemaattisen menetelmän tai mallin avulla, jota voitaisiin hyödyntää tarvittaessa jokaisen toteutettavan vaatimuksen kohdalla.

5.1 Kohdeyritys

Tämän tutkimuksen kohdeorganisaatio oli suomalainen vuonna 1991 perustettu ohjelmistotalo. Tutkimukseen sisältyvän kehittämishankkeen tavoitteena oli auttaa ja helpottaa kansainvälistä ohjelmistotuoteyritystä priorisoimaan ohjelmistotuotteisiin toteutettavia vaatimuksia jäsentämällä ja systematisoimalla vaatimusten priorisointia.

5.1.1 Tuotekehitys

Kohdeyritys on tarjonnut asiakkailleen valmisohjelmiston, jonka avulla on yhdistetty prosessien mallinnus ja analysointi, prosessien automatisointi, suorituskyvyn johtaminen ja mittaaminen ja liiketoiminnan analysointi yhdeksi prosessijohtamisen, laatujohtamisen ja riskienhallinnan kokonaisratkaisuksi.

Kohdeyrityksen ohjelmistoratkaisujen avulla organisaatiot sekä julkisella – että yksityisellä sektorilla ovat tehostaneet liiketoimintaprosessejaan, tunnistaneet ja hallinneet liiketoimintariskejään, parantaneet suorituskykyään sekä toteuttaneet strategiaansa.

Kohdeyrityksen tuotekehityksessä työskenteli tutkimuksen aikana 20 henkilöä, mikä vastasi noin 35 % koko henkilöstöstä. Tuotekehityksen keskeinen osaaminen oli tuolloin keskitetty omaan organisaatioon. Yhtiön tuotekehitysmenot olivat noin 20 % yhtiön liikevaihdosta. Konsernin liikevaihto oli 6,6 milj. € vuonna 2009 jakautuen ohjelmistolisensseihin, ylläpitopalveluluihin ja asiantuntijapalveluihin.

5.1.2 Strategiset linjaukset

Kohdeyritys on kasvattanut toimittamiensa ohjelmistojen käyttäjämääriä niin uusien, kuin nykyistenkin asiakkaiden organisaatioissa. Tämän on uskottu lisäävän myös yhtiön asiantuntijapalveluiden myyntiä.

Kohdeyritys on pyrkinyt nopeuttamaan ohjelmistojensa myyntiprosessia ja ratkaisujensa käyttöönottoa kehittämällä niiden ympärille palvelukonsepteja sekä tarjoamalla kohderyhmäkohtaisiin tarpeisiin perustuvia malliratkaisuja, jotka ovat olleet monistettavissa.

Kansainvälisillä markkinoilla kohdeyritys on pyrkinyt monipuolistamaan ja vahvistamaan jälleenmyyntikanavaansa. Lähialueilla on panostettu erityisesti Venäjän myyntiin.

Kohdeyrityksen tuotekehitys on ollut kohdistettu tukemaan valittua palvelutarjontaa. Tässä on hyödynnetty olemassa olevia kohdeyrityksen tuotteita ja niiden vahvaa Microsoft-yhteensopivuutta.

5.1.3 Vaatimusten priorisoinnin haasteet

Asiakkaiden esittämät vaatimukset kohdeyrityksen ohjelmistoratkaisulle ovat olleet yhä monimutkaisempia ja vaativampia. Näiden seurauksena on pitänyt pystyä tekemään valintoja toteuttavien ominaisuuksien suhteen, jotta ne voitu toteuttaa käytettävien resurssien puitteissa. Kilpailu kohdeyrityksen ratkaisualueilla on ollut kovaa, joten toteuttavien ominaisuuksien on pitänyt tarjota kilpailuetua, huomioida tarjottavien ohjelmistoratkaisujen välttämättömät toiminnallisuudet ja huolehtia olemassa olevien asiakkaiden vaatimista toiminnallisuuksien laajennuksista.

Kansainvälisen ohjelmistotuoteyrityksen näkökulmasta vaatimusten priorisointi on eronnut normaalista asiakasprojektista erityisesti siinä, että vaatimuksia ovat esittäneet samanaikaisesti useat sidosryhmät. Kohdeyrityksen kohdalla tämä on tarkoittanut n. 900 loppuasiakasta ja 80 aktiivista partneria maailmalla. Lisäksi kohdeyrityksen oma liiketoiminta on esittänyt vaatimuksia, jotka ovat pohjautuneet mm. kilpailijaseurantaan, markkina-analyyseihin, innovaatioihin sekä ohjelmiston laatuvaatimuksiin kuten käytettävyyteen, suorituskykyyn ja luotettavuuteen.

5.2 Kohdeyrityksen ohjelmistokehitysprosessin nykytilan analyysi

Yrityksen käytössä olleet ohjelmistokehitysprosessit arvioitiin vertailemalla niitä teorioiden mukaisiin ohjelmistokehitysprosesseihin. Arvioinnin laajuudeksi päätettiin ohjelmistokehitysprosessin alkuvaiheiden toimenpiteet, joissa käsiteltiin vaatimuksia ennen niiden toteuttamista. Arviointiryhmässä oli tutkijan lisäksi mukana yrityksen tuotekehitysjohtaja tarkentamassa prosessien kuvauksia ja dokumentaatiota aina tarvittaessa. Arvioinnin tavoitteena oli tunnistaa, missä vaiheissa, milloin ja ketkä osallistuivat vaatimusten priorisointiin.

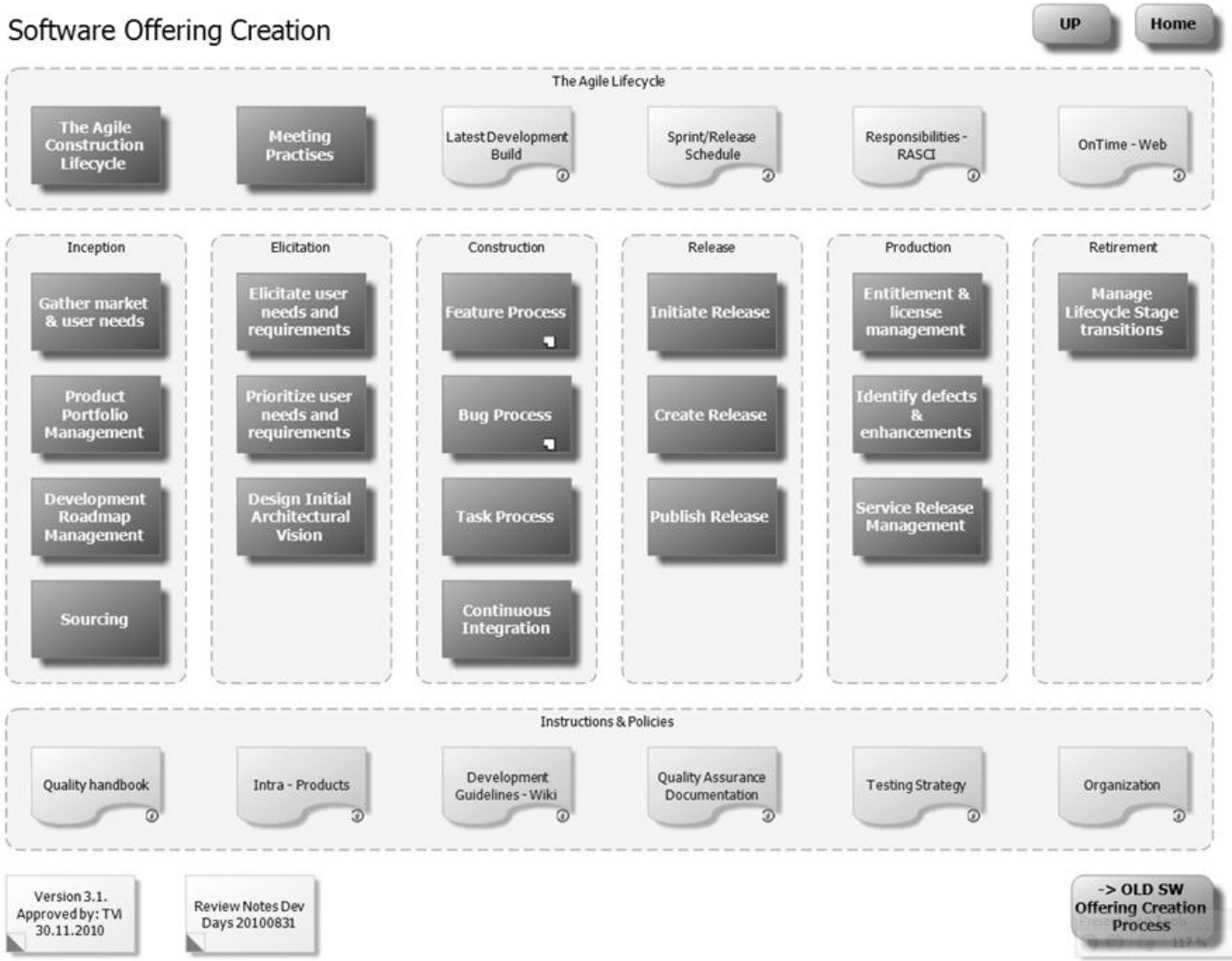
Arvioinnin suunnittelussa hyödynnettiin Laamasen määrittelemiä parhaita käytäntöjä prosessien arvioinnin suunnittelussa, joiden mukaisesti arvioinnin kohteena olivat prosessikokonaisuudet ja tutustuminen prosessien dokumentaatioon. Lisäksi noudatettiin Laamasen painotusta arviointeja suorittavien henkilöiden riittävästä kohdeprosessien asiantuntemuksesta. (Laamanen 2004, 112–113.)

5.2.1 Tuotekehityksen prosessikartta ja työohjeet

Kohdeyrityksen ohjelmistokehityksen käyttämät prosessit ja työohjeet oli mallinnettu keskitetysti koko yrityksen käyttöön ja niitä käytettiin apuna analysoitaessa vaatimusten priorisoinnin nykytilaa kohdeyrityksessä.

Ohjelmistotuotteiden vaatimukset priorisoidaan yleensä ohjelmistokehitysprosessin alussa, kuten tämän tutkimuksen alussa esitetty teoria ohjelmistokehitysprosessista ja sen sisältämästä vaatimusmäärittelystä, vaatimusten analysoinnista sekä - neuvottelusta osoitti. Kohdeyritykselle suoritetussa prosessianalyysissä todettiin, että käytössä ollut ohjelmistokehitysprosessi oli linjassa tutkittujen teorioiden kanssa. Em. toteamus tuki olettamusta, jonka tavoitteena oli saada sovitettua tämän tutkimuksen tuloksena syntyvä priorisointimalli yhteen olemassa olevien prosessien kanssa. Kohdeyrityksen tuotekehitysprosessien parantamisen painopistealue oli ollut konstruktiovaihe sekä julkaisuvaihe viimevuosina. Ohjelmistokehitysprosessin aikaisempien prosessivaiheiden parantamiseen ja kehittämiseen ei ollut panostettu yhtä paljon, mutta ne oli tunnistettu osaksi ohjelmistokehitysprosessia.

Kohdeyrityksen käyttämät tuotekehitysprosessit ja työohjeet on esitetty kokonaisuudessaan kuviossa 12.

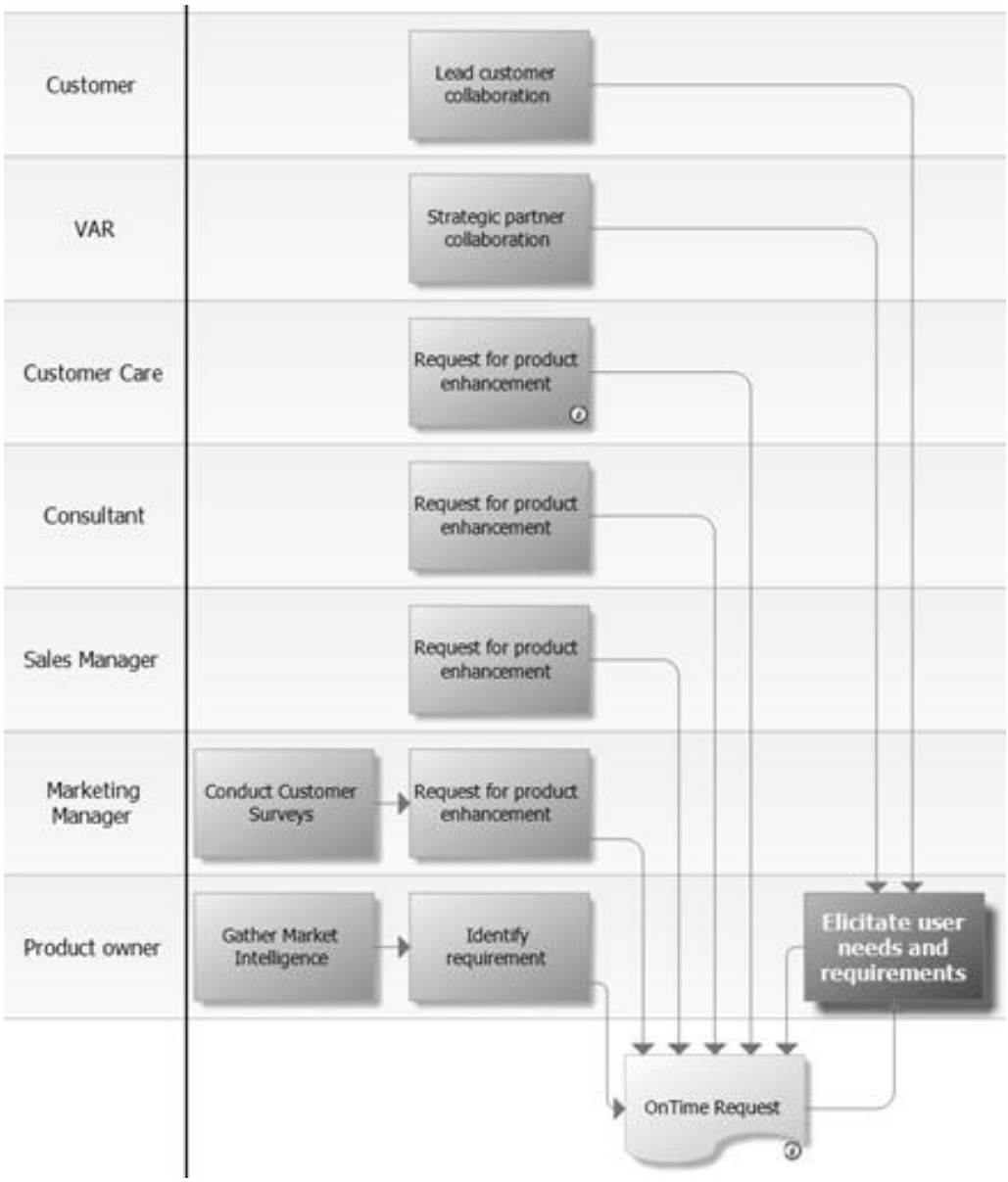


Kuvio 12. Kohdeyrityksen tuotekehityksen prosessikartta ja työohjeet.

5.2.2 Vaatimusten alku ja voimaantulo

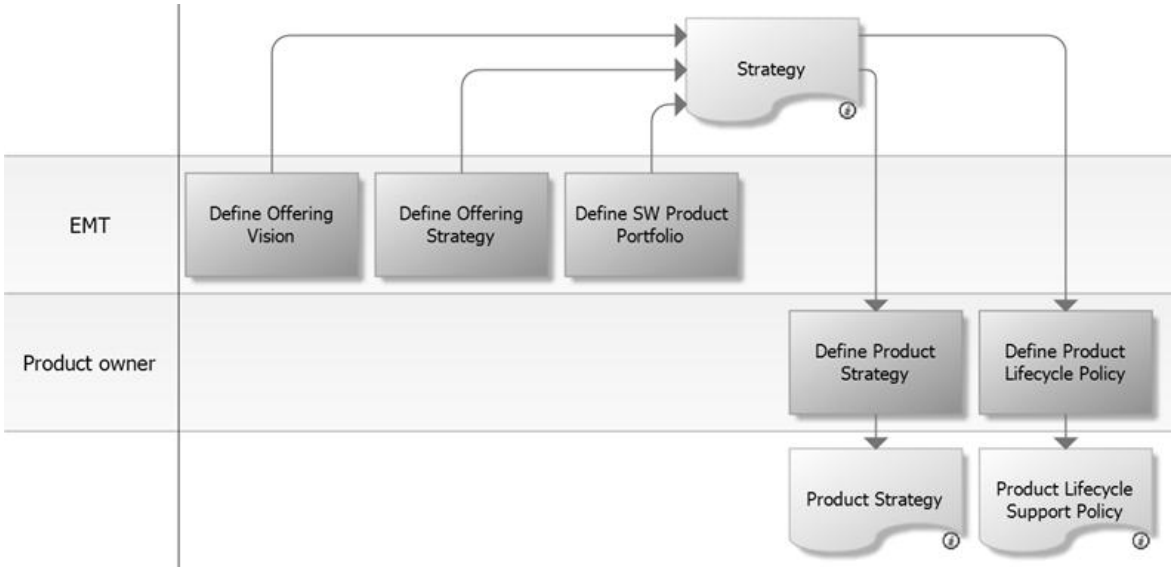
Vaatimusten alkua ja voimaantuloa hallittiin neljän prosessin avulla. Prosessien avulla kerättiin ja koottiin vaatimukset yhteen monesta eri näkökulmasta niin, että niistä saatiin muodostettua tuotesuunnitelma (product road map).

Markkinoiden ja käyttäjien esittämät vaatimukset koottiin keskitetysti vaatimusten hallintajärjestelmään. Kuviossa 13 on kuvattu, miten asiakkailta ja yhteistyökumppaneilta suoraan tulleet vaatimukset tarkennettiin erillisessä prosessissa, sillä yrityksen ulkopuolisilla sidosryhmillä ei ollut käyttöoikeuksia vaatimusten hallintajärjestelmään. Erillisen prosessin avulla varmistettiin, että yrityksen ulkopuolelta tulleet vaatimukset oli ymmärretty oikein ja, että ne sisälsivät kaiken tarvittavan tiedon.



Kuvio 13. Vaatimusten kerääminen.

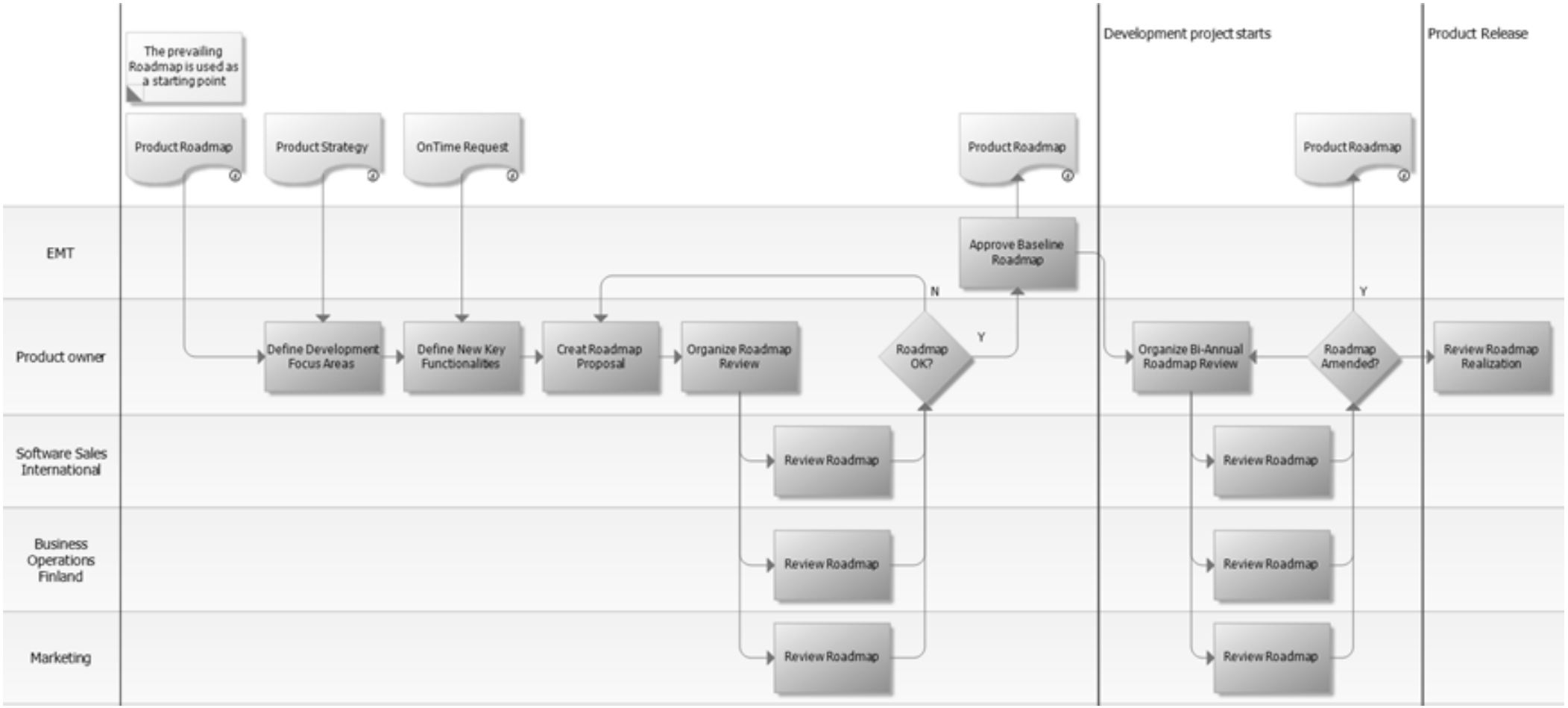
Tuotestrategia ja tuotteiden elinkaari määriteltiin tuotesalkun hallintaprosessissa kuvion 14 mukaisesti johtamalla ne koko yrityksen strategiasta. Kokonaisvastuu yrityksen tarjoamasta ja ohjelmistotuotesalkusta oli yrityksen johtoryhmällä. Tarjoama sisälsi yrityksen asiakkaille ja yhteiskumppaneille tarjottavat ohjelmistotuotteet, ratkaisut ja asiantuntijapalvelut.



Kuvio 14. Tuotesalkun hallinta.

Kohdeyrityksen tuotesuunnitelma laadittiin erillisen prosessin avulla, joka on kuvattu kuviossa 15. Tuotesuunnitelman laatimiseen käytetyn prosessin aktiviteetit ottivat erityisesti kantaa eri sidosryhmien vastuisiin, kuten katselmointeihin. Prosessi ei ottanut kantaa itse vaatimusten priorisointiin. Ohjelmistotuotteiden omistajien vastuu korostui tuotesuunnitelman laatimisessa ja ylläpitämisessä. Tuotesuunnitelma oli aktiivisessa käytössä kohdeyrityksessä. Tuotesuunnitelmaa tarkistettiin ja parannettiin tarvittaessa puolivuosittaisissa katselmoinneissa. Katselmointeihin osallistuivat johtoryhmän ja tuotteiden omistajien lisäksi myyntiyksiköiden sekä markkinoinnin edustajat. Systemaattiset katselmoinnit mahdollistivat uusien painopistealueiden lisäämisen tuotesuunnitelmalle mutta se oli haasteellista koska painopistealueita eri priorisoitu.

Lisäksi oli määritelty prosessi toimittajien hallintaan, joka mahdollisti jonkin tietyn vaatimuksen toteuttamisen ulkoisen palvelutoimittajan toimesta hallitusti ennalta määritellyn prosessin mukaisesti.

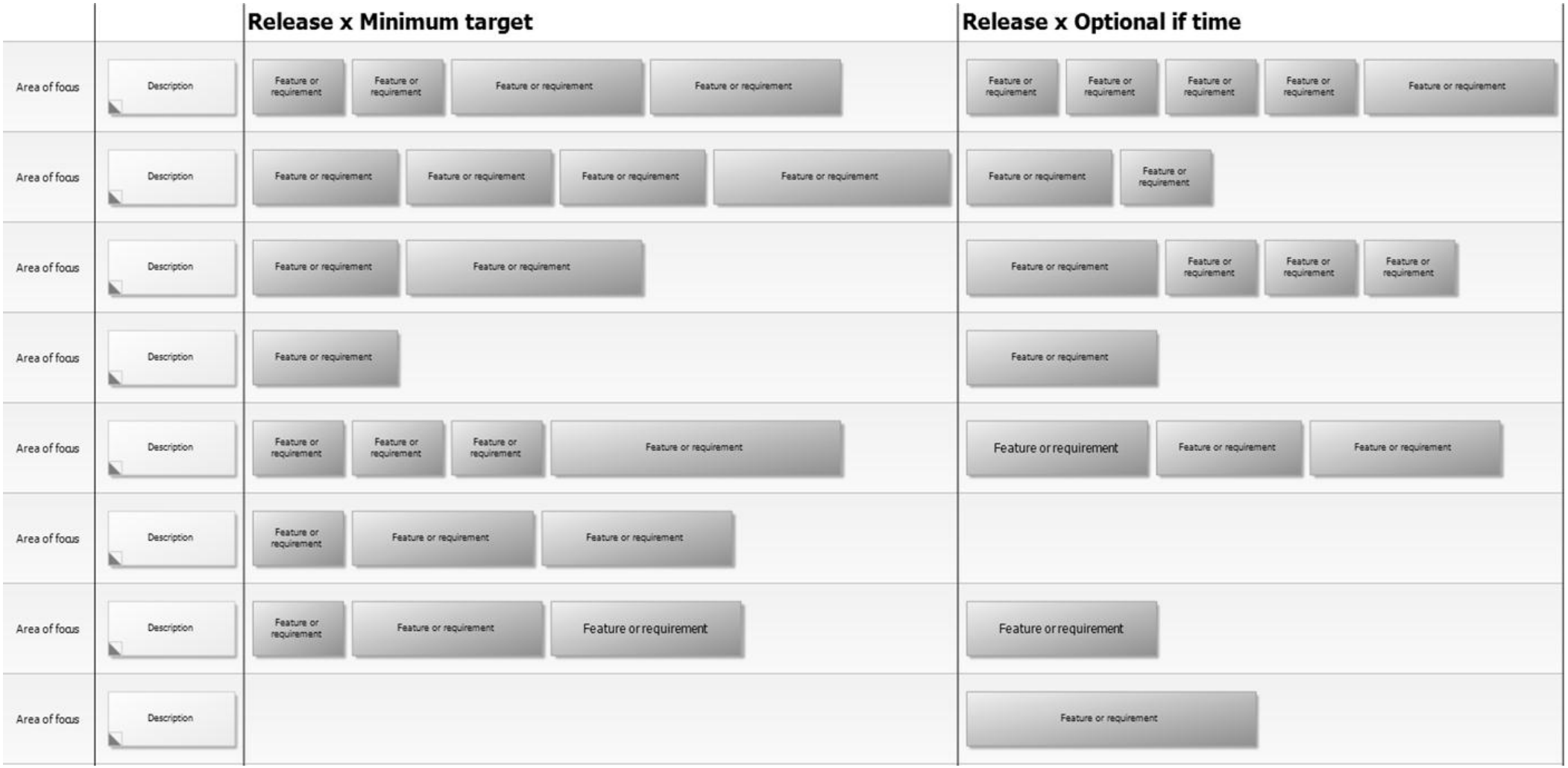


Kuvio 15. Tuotesuunnitelman hallintaprosessi.

5.2.2.1 Tuotesuunnitelma

Tuotesuunnitelman avulla hallittiin liiketoiminnan esittämiä vaatimuksia kuvion 16 mukaisesti. Tuotesuunnitelmaa hyödynnettiin erityisesti kommunikoinnissa liiketoiminnan ja tuotehallinnon välillä. Tuotesuunnitelma laadittiin seuraavalla julkaisulle. Tuotesuunnitelma oli jaettu horisontaalisti yrityksen strategian mukaisiin painopistealueisiin. Painopistealueille oli sijoitettu vaatimuksia, joista osa oli laajempia ja osa pienempiä. Vaatimuksia kuvaavan suorakaiteen leveys osoitti työmääräarvion suhteessa muihin vaatimuksiin. Osa tuotesuunnitelmalla olevista vaatimuksista sisälsi useamman pienemmän vaatimuksen, mutta niitä ei voinut nähdä tuotesuunnitelmalta. Tuotesuunnitelma oli jaettu kahteen osaan huomioimalla käytössä olevat resurssit ja julkaisuaikataulu: 1) vaatimukset, jotka piti toteuttaa ja 2) vaatimukset, jotka toteutettiin, mikäli käytettävissä oli ylimääräistä aikaa. Kaikki tuotesuunnitelmalla olevat ohjelmistovaatimukset olivat aina täysin uusia ohjelmistovaatimuksia eli siinä ei ollut vaatimuksia, jotka liittyivät esim. jo olemassa olevien ominaisuuksien jatkokehitykseen. Vaatimusten toteutusjärjestys perustui tuotehallinnan parhaaseen tietotaitoon ja tarveharkintaan.

Tuotesuunnitelman merkitys korostui aloitettaessa suunnittelemaan uutta tuotejulkaisua. Tuotesuunnitelma oli graafinen ja sen avulla pystyttiin kommunikoimaan tehokkaasti eri sidosryhmien välillä. Tuotesuunnitelman haasteena oli eri abstraktiotasoilla olevat vaatimukset. Tämä aiheutti mm. sen, että liiketoiminta saattoi ehdottaa pienehkön toiminnallisuuden korvaamista huomattavasti suuremmalla toiminnallisuuskokonaisuudella. Lisäksi tuotesuunnitelma kattoi ainoastaan yhden tuotejulkaisun tuotesuunnitelman, joten sen perspektiivi oli lyhytnäköinen.

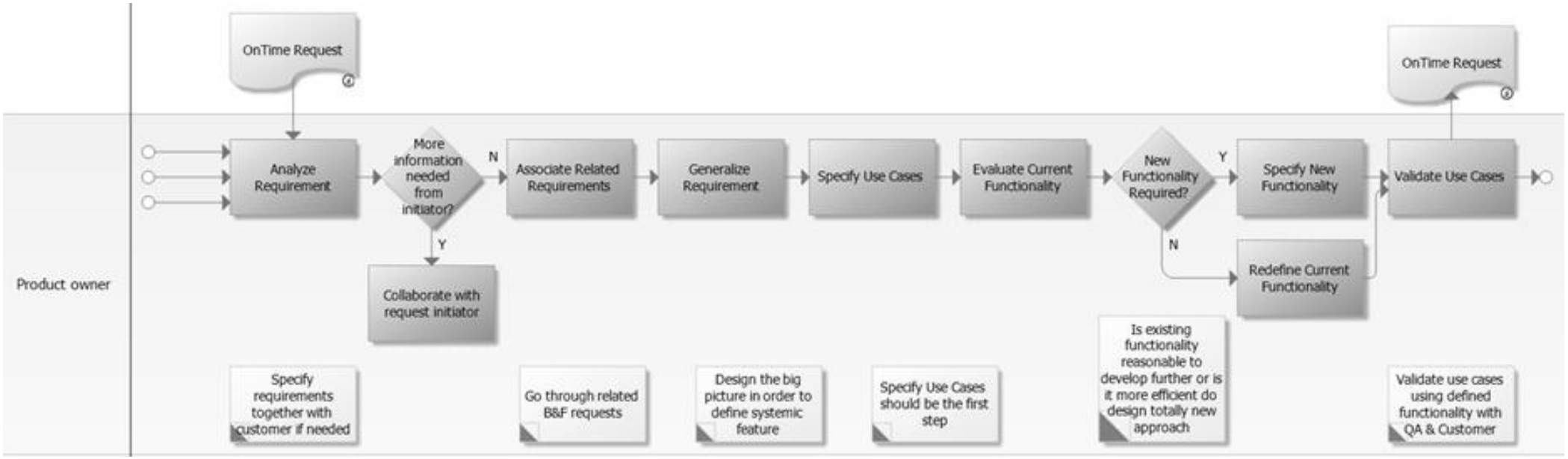


Kuvio 16. Tuotejulkaisun tuotesuunnitelma.

5.2.3 Vaatimusten esilletuonti ja selvitys

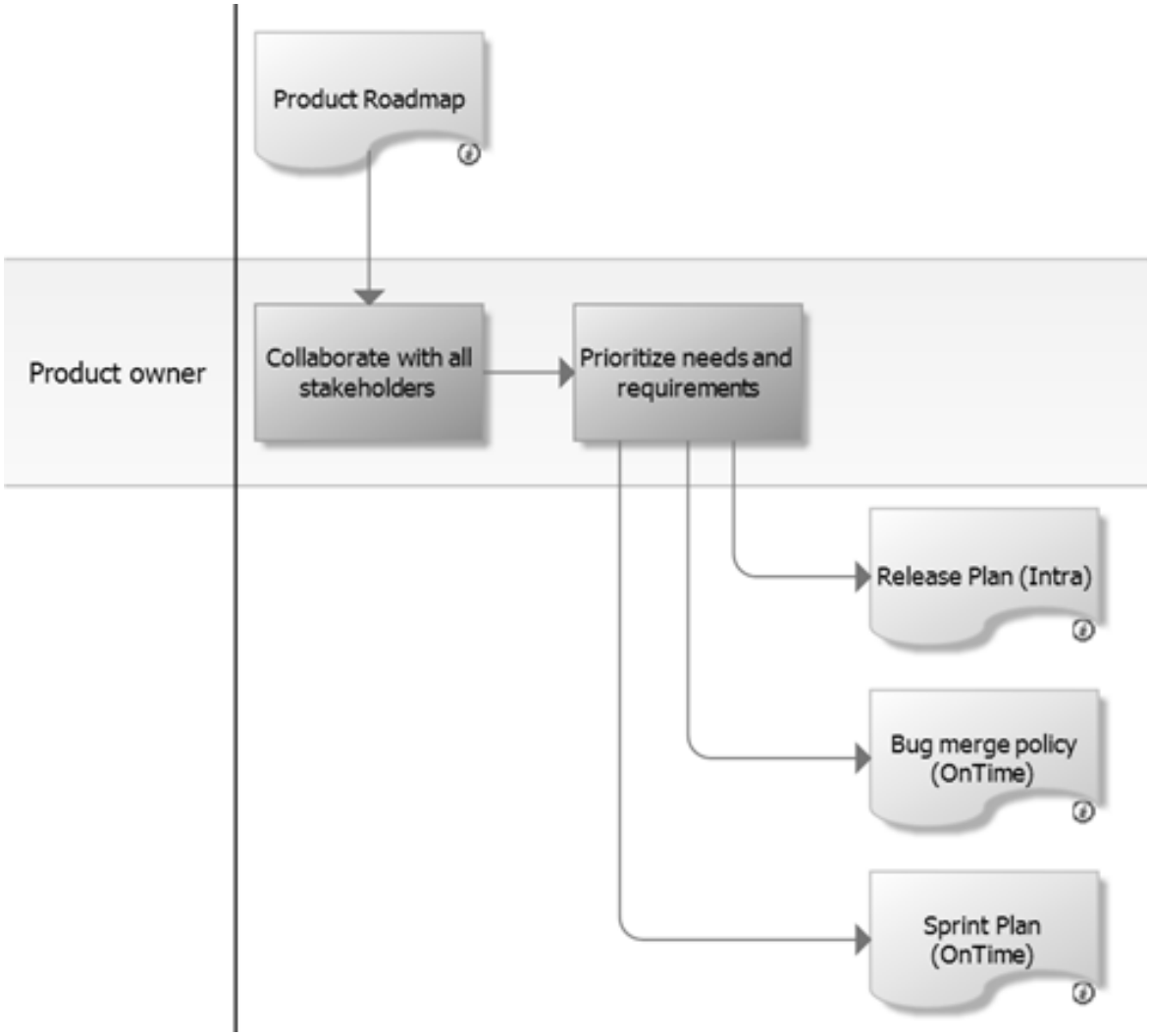
Vaatimusten jatkoselvityksen hallintaan oli tunnistettu kolme prosessia, joiden avulla pyrittiin varmistamaan, että kaikki tarvittavat tiedot olivat kaikkien tiedossa ennen vaatimusten toteuttamista.

Vaatimusten selvitys hallittiin omassa prosessissa, jossa tuotteiden omistajien vastuu korostui. Prosessissa oli takaisinkytkentä kuviossa 17 esiteltyyn vaatimusten keräämiseen, jossa painotettiin yrityksen ulkopuolisilta sidosryhmiltä tulleiden vaatimusten hallintaa. Prosessin avulla varmistettiin, että kaikki ymmärsivät vaatimusten tavoitteet ja taustat ja että kaikki mahdolliset liitännäiset sekä riippuvat vaatimukset oli huomioitu ennen vaatimusten toteuttamista.



Kuvio 17. Vaatimusten selvitys.

Vaatimusten priorisointiin oli tunnistettu oma prosessi. Prosessi on kuvattu kuviossa 18. Prosessi ei määritellyt tarkemmin miten vaatimukset pitäisi priorisoida. Prosessi korosti tuotteiden omistajien vastuuta ja priorisoitujen vaatimusten julkaisua yrityksen sisäiseen käyttöön.



Kuvio 18. Vaatimusten priorisointi.

Lisäksi vaatimusten selvitysvaiheessa suunniteltiin alustavat arkkitehtuurilliset muutokset teknisestä näkökulmasta, mikäli vaatimuksien toteuttaminen vaati sellaisia. Prosessin avulla varmistettiin, että vaatimusten mahdolliset riskit ja kustannukset pystyttiin esittämään liiketoiminnalle.

5.2.4 Prosessianalyysin tulokset

Yrityksen ohjelmistokehitysprosessi ja tarkoituksenmukaiset työohjeet olivat hyvin kuvattu. Prosessien kuvauksista ilmeni prosessien soveltamisala ja niissä oli huomioitu asiakkaiden tarpeet kontekstiin sidottuna. Prosessien syötteet, tuotteet ja palvelut, kuten toimintaohjeet olivat hyvin kuvattu ja niiden avulla loogisten prosessiketjujen seuraaminen oli tehokasta. Varsinai-

siin prosessikaavioihin oli kuvattu myös eri roolien vastuut, jotka selvensivät yrityksen päätöksentekokäytäntöjä.

Tuotehallintatiimi ylläpiti tuotesuunnitelmaa uusille tuotesukupolville ja tuoteversioille huomioimalla erityisesti liiketoiminnan vaatimukset. Tutkimuksen aikana oli käytössä tuotesuunnitelma kesällä 2011 julkaistavalle 8.2 ohjelmistoversiolle, joka sisälsi noin 10 liiketoiminnan strategiaan kytkettyä vaatimusta, joihin oli jo kiinnitetty alustavat resurssit ja aikataulut. Kaikki tuotesuunnitelmassa mukana olleet liiketoimintavaatimukset saivat automaattisesti niin korkean prioriteetin, että ne tultaisiin toteuttamaan, mikäli käytettävä teknologia sen mahdollistaisi. Vertailun vuoksi todettiin, että 8.0 ohjelmistoversioon toteutettiin 262 yksittäistä ominaisuutta ja 8.1 ohjelmistoversioon 216 yksittäistä ominaisuutta. Yksittäisten toteutettujen ominaisuuksien suuri lukumäärä selittyi osin sillä, että liiketoimintavaatimukset oli pilkottu pienempiin kokonaisuuksiin. Tämä ei kuitenkaan selittänyt kokonaan yksittäisten toteutettujen vaatimusten suurta lukumäärää, eikä kaikista vaatimuksista ollut minkäänlaista linkkiä tuotesuunnitelmaan. Toisaalta voitiin myös olettaa, että julkaistavaan 8.2 ohjelmistoversioon sisällytettäisiin vielä useita pienempiä vaatimuksia kaikkien olemassa olevien vaatimusten joukosta, joita tuli lisää joka kuukausi noin 20 kpl.

Ns. tarveharkintaan perustuvat vaatimukset jakautuivat kolmeen eri kategoriaan: välttämättömät, differentioivat ja pienet vaatimukset. Em. kolmijako tuli esille aikaisemmin esitetyssä kuviossa 15, jonka mukaisesti tuotteiden omistajat saivat syötteitä ohjelmistotuotekehityksen painopistealueille kolmesta eri lähteestä, jotka olivat a) tuotesuunnitelma, b) tuotestrategia ja c) yksittäiset vaatimukset. Välttämättömät vaatimukset tarkoittivat sellaisia vaatimuksia, jotka oli johdettu suoraan yrityksen strategiasta ja ne olivat yrityksen johtoryhmän päättämiä. Differentioivilla vaatimuksilla erottauduttiin kilpailijoista, mutta niitä ei johdettu välttämättä yrityksen strategiasta. Pienehköjä vaatimuksia toteutettiin yleensä suurien ja tärkeiden asiakkuuksien toiveesta sekä vastaamaan asiantuntijapalveluiden käyttötarpeisiin. Em. luokittelu ei kuitenkaan itsessään ratkaissut vaatimusten priorisointiin liittyviä haasteita, sillä vaatimuksen esittäjän näkökulmasta juuri hänen esittämänsä vaatimus olisi pitänyt priorisoida korkeimmalle prioriteetille syystä tai toisesta.

Tutkimuksen perehtymisvaiheessa suoritetun kirjallisuuskartoituksen perusteella todettiin, ettei ole olemassa yksiselitteitä priorisointimenetelmää, jota voitaisiin hyödyntää sellaisenaan kohdeyrityksessä. Tämä väittämä pohjautui siihen tosiasiaan, että vaatimusten priorisointiin osallistuvien henkilöiden piti pystyä hyödyntämään vaatimusten priorisointia tarkoituksenmukaisella tavalla omassa kontekstissaan. Oli tärkeää, että liiketoiminta pystyisi priorisoimaan strategiasta

lähtöisin olevat 10 - 20 laajempaa vaatimusta ja toisaalta tuotehallinnan piti pystyä priorisoimaan satoja pienempiä vaatimuksia tehokkaasti ja riittävän läpinäkyvästi päivittäin.

Prosessianalyysin perusteella vaatimusten priorisoinnin voitiin katsoa perustuvan projektikohtaiseen tarveharkintaan, koska mitään systemaattista menetelmää ei ollut käytössä vaatimusten priorisoinnille.

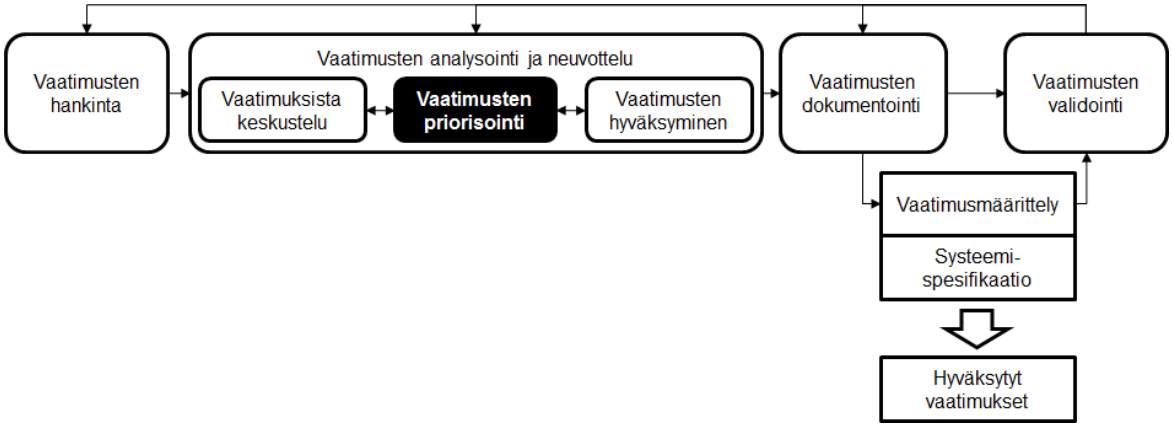
5.3 Avainhenkilöiden avoimet haastattelut

Kohdeyrityksen vaatimusten priorisoinnin nyky- ja tavoitetila eri roolien näkökulmasta selvitettiin kahdeksalle henkilölle suoritetulla avoimella haastattelulla. Haastatteluihin valittiin henkilöt, jotka osallistuivat jollain tavalla uuteen ohjelmistoversioon mukaan otettavien ominaisuuksien valintaan kohdeyrityksen sisällä. Tässä tutkimuksessa haluttiin keskittyä parantamaan lähtökohtaisesti kohdeyrityksessä työskentelevien henkilöiden priorisointiprosessia. Yrityksen suorat asiakkaat, yhteistyökumppanit ja yhteistyökumppaneiden asiakkaat rajattiin tutkimuksen ulkopuolelle. Toisaalta todettiin, että liiketoiminnan asettamat vaatimukset edustivat yleensä aina eri asiakasryhmiä ja markkinoita. Lisäksi loppukäyttäjänäkökulma huomioitiin ottamalla haastattelun mukaan palveluliiketoiminnan päällikkö. Haastattelut toteutettiin yksilöhaastatteluina. Haastateltavien toimenkuvat jakaantuivat seuraavasti: tuotepäällikkö, tuotteen omistaja, teknologiajohtaja, tuotekehitysjohtaja, Suomen liiketoimintayksikön johtaja, kansainvälisen liiketoimintayksikön johtaja, liiketoiminnan kehitysjohtaja ja palveluliiketoiminnan päällikkö.

Haastateltaville toimitettiin haastattelujen pohjaksi taulukon 5 mukainen runko kysymyksistä ja kuvion 19 mukainen kaavio vaatimusten priorisoinnin suhteesta määrittelyprosessiin. Kysymykset ja kaavio johdattelivat haastateltavat aiheeseen ja niitä käytettiin haastattelun tukena aina tarvittaessa. Lähtökohtaisesti haastattelut etenivät kuitenkin haastateltavien henkilöiden käsitysten, ajatusten ja mielipiteiden mukaisessa järjestyksessä sen mukaan, kun ne tulivat vastaan keskusteluissa. Haastattelujen runko laadittiin tutkimuksen pääongelman ja osaongelmien mukaisesti. Näin ollen haastatteluissa yritettiin selvittää vaatimusten priorisointiin osallistuvien henkilöiden näkemyksiä ja tuntemuksia kohdeyrityksessä jo käytössä olleesta priorisointikäytännöstä, kuten sen hyödyistä ja ongelmista. Tämän lisäksi haastateltavia pyydettiin arvioimaan tavoitetilaa vaatimusten priorisoinnin suhteen eli miten siitä saisi riittävän tarkan, systemaattisen ja läpinäkyvän eri sidosryhmille. Haastattelujen kesto vaihteli tunnista kahteen tuntiin.

Taulukko 5. Avoimen haastattelun runko.

Nykytila	Tavoitetila
• Miten vaatimusten priorisointi hoidetaan tällä hetkellä?	• Missä vaiheessa/vaiheissa vaatimusten priorisointi tulisi tehdä?
• Mitä hyviä puolia on nykyisessä priorisointikäytännössä?	• Mikä on loppukäyttäjien ja asiakkaiden rooli vaatimusten priorisoinnissa, jos mikään?
• Mitä huonoja puolia on nykyisessä priorisointikäytännössä?	• Millä tarkkuustasolla vaatimukset pitäisi erottaa toisistaan?
	• Pitäisikö priorisointi tehdä kaikille vaatimuksille systemaattisesti?
	• Miten ja kenelle vaatimusten prioriteetit tulisi kommunikoida?



Kuvio 19. Vaatimusten priorisointi määrittelyprosessissa (Kotonya & Sommerville 2003, 32–60).

5.3.1 Tuotepäällikkö

Tuotepäällikön mukaan vaatimusten priorisointi ei näkynyt omassa työssä juuri lainkaan, eikä yksittäisten vaatimusten prioriteettijärjestystä voinut tarkistaa mistään. Toisaalta tuotepäällikkö koki tuotesuunnitelman konkreettisena tuotoksena, jonka mukaan vaatimukset toteutettiin tai ei toteutettu. Tuotesuunnitelma antoi raamit ohjelmistojulkaisuille ja siellä oli myös vaatimuksia, jotka piti tehdä, jos oli ylimääräistä aikaa. Hän painotti, että tuotesuunnitelman lisäksi päivittäisessä työssä joutui työskentelemään loppukäyttäjiltä saatujen ominaisuuslaajennoksien ja tuotehallinnan lisäämien pienten vaatimusten kanssa, joita ei linkitetty tuotesuunnitelmaan. (Hilska, T. 8.6.2010.)

Tuotepäällikkö totesi, että tuotesuunnitelman aikahorisontin pitäisi olla jatkuvasti noin kaksi vuotta tulevaisuudessa ja varsinainen priorisointiprosessi pitäisi kytkeä tuotesuunnitelmaan esim. puolen vuoden sykleissä. Tuotepäällikkö kertoi, että priorisointi olisi hyvä tehdä sisäisesti yrityksessä ja tuotehallintatiimi voisi olla tarvittaessa yhteydessä asiakkaisiin sekä loppukäyttäjiin. Lisäksi hän painotti, että kaikki vaatimukset pitäisi saada systemaattisesti jonoon, jotta olisi mahdollista allokoida ne tehokkaasti toteutettaviksi. Tuotepäällikkö kaipasi myös enemmän läpinäkyvyyttä vaatimusten hallintaan priorisoinnin avulla, joka antaisi asianosaisille paremmat mahdollisuudet suunnitella tulevaisuutta ohjelmistojulkaisujen näkökulmasta. (Hilska, T. 8.6.2010.)

5.3.2 Tuotteen omistaja

Tuotteen omistajalle vaatimusten priorisointi tarkoitti liiketoiminnalle järjestettyjä työpajasessioita, joiden tuloksena koostettiin tuotesuunnitelma. Työpajoihin otettiin mukaan kaikki liiketoiminnalta tulleet vaatimukset. Työpajoissa oli käytössä ns. demokraattinen äänestysprosessi, jonka mukaisesti eniten ääniä saaneet vaatimukset valittiin tuotesuunnitelmalle. Tuotteen omistajalla ei ollut tietoa, mistä ja miten liiketoiminta oli kerännyt vaatimukset, mutta hän oletti, että ne olivat viimeisimpiä asiakkailta tulleita vaatimuksia. Tuotteen omistaja luokitteli vaatimukset tuotesuunnitelmassa oleviin isoihin kokonaisuuksiin, pienempiin loppukäyttäjävaatimuksiin ja teknisluontoisiin vaatimuksiin. Tuotteen omistajan mielestä liiketoiminnan laaja osallistuminen työpajoihin oli hyvä asia. Hän koki vaatimuksista äänestämisen suoraviivaiseksi ja läpinäkyväksi. Lisäksi hänen mielestään oli hyvä, että tuotehallinta ja tuotekehitys huolehtivat ohjelmistotuotteiden tekniseen arkkitehtuuriin vaikuttavista muutoksista. Tuotteen omistaja painotti, että todellinen prioriteettijärjestys puuttui, eikä sitä näin ollen voinut hyödyntää. (Siipola, T. 9.6.2010.)

Tuotteen omistaja kertoi, että priorisointia pitäisi pystyä tekemään liiketoiminnan tarpeiden mukaisesti eli pitäisi ymmärtää liiketoiminnan tarpeiden muutokset oikea-aikaisesti. Hänen mukaansa liiketoiminnan kanssa suoritettavia katselmointeja pitäisi järjestää tulevaisuudessa useammin esim. kaksi kertaa kuukaudessa. Tuotteen omistaja oletti, että markkinoiden asettamat vaatimukset oli huomioitu yrityksen johtoryhmässä, mutta hän kaipasi jonkinlaista takaisinkytkentää myös asiakkaiden ja loppukäyttäjien suuntaan. Tuotteen omistaja hahmotteli myös eritasoisten vaatimusten systemaattista linkittämistä, jotta vaatimuksia pystyisi jäljittämään tehokkaammin. Tuotteen omistajan mielestä jokaisen vaatimuksen systemaattinen priorisointi voisi olla mahdollista hyödyntämällä tarkoituksenmukaisia menetelmiä eri vaatimustasoilla. (Siipola, T. 9.6.2010.)

5.3.3 Teknologiajohtaja

Teknologiajohtajan mielestä vaatimusten priorisointi suoritettiin huutoäänestyksellä, vaikka tuotesuunnitelma saatiin lopulta rakennettua jonkinlaisella konsensusajattelulla. Tuotesuunnitelman lisäksi mitään muuta priorisointia ei juurikaan tehty. Lisäksi tuotehallintatiimi käsitteli erikseen teknisiä reunaehtoja, jotka sanelivat, mitä ylipäätänsä on mahdollista toteuttaa. Lisäksi teknologiajohtaja eritteli muut vaatimukset, jotka olivat yleensä asiakkailta tai tuotehallinnalta tulleita pieniä parannusehdotuksia. Teknologiajohtajan mielestä liiketoiminnalla oli hyvä näkemys siitä mitä kannattaa toteuttaa. Toisaalta hän painotti, että tuotesuunnitelman aikaperspektiivi on liian lyhyt, koska se kattaa vain yhden tuotejulkaisun. Hänen mielestään ohjelmistotuotteiden kehittämistä tehtiin liian lyhyessä syklissä, joka aiheutti mm. sen, että teknologisten vaatimusten analysointiin tarvittava aika oli usein liian lyhyt. (Junkkonen, K. 10.6.2010.)

Teknologiajohtajan oli sitä mieltä, että ohjelmistotuotteiden julkaisusykli pitäisi saada lyhyemmäksi. Lyhyempi julkaisusykli korostaisi hänen mielestään priorisoinnin merkitystä erittäin paljon, koska tärkeimmät vaatimukset priorisoitaisiin toteutettaviksi useamman kerran suhteessa käytössä olleeseen toimintatapaan. Lisäksi hän painotti, että myös teknologiavaatimukset pitäisi saada läpinäkyväksi liiketoiminnalle, joka mahdollistaa läpinäkyvämmän keskustelun kehittäjäresursseista. Hän totesi, että myös asiakkaiden ja loppukäyttäjien asettamat vaatimukset pitäisi pystyä kytkemään tuotesuunnitelmalle. Teknologiajohtaja hahmotti mielessään, että priorisointia pitäisi pystyä tekemään mahdollisimman kevyesti kontekstisidonnaisesti, jottei se muodostu liian byrokraattiseksi. Toisaalta hän toivoi, että vaatimusten priorisointi toteutettaisiin systemaattisesti esim. kolmen kuukauden sykleissä. Lisäksi hän painotti, että yksittäisten vaatimusten prioriteettien kommunikointi liiketoiminnalle pitäisi saada entistä läpinäkyvämmäksi. (Junkkonen, K. 10.6.2010.)

5.3.4 Tuotekehitysjohtaja

Tuotekehitysjohtajan mielestä oli lähtökohtaisesti järkevää tutkia tuotekehityksen prosessien ja menetelmien parantamista ohjelmistotuotteiden vaatimusten priorisoinnin näkökulmasta. Hän painotti, että yleensä tulevaisuuden ennustaminen perustuu näkemykseen, joten vaatimusten priorisointikaan ei saisi olla pelkkä mekaaninen harjoitus. Tuotekehitysjohtajalle vaatimusten priorisointi tarkoitti laajaa prosessia, jossa oli mukana useita sidosryhmiä. Vaatimusten karkea valinta alkoi yrityksen strategiaprosessin yhteydessä. Strategiatyö jatkui tuotesuunnitelmaprosessilla, jonka avulla muodostettiin tuotesuunnitelma seuraavalle tuotejulkaisulle. Käytännössä tuotesuunnitelma valmisteltiin liiketoiminnalle järjestetyissä työpajoissa. Tuotesuunnitelmalle otettiin mukaan isoja vaatimuskokonaisuuksia, joista osa oli johdettu yrityksen strategiasta ja

osa koettiin tärkeiksi muista syistä. Tuotehallinta pilkkoi suuret osakokonaisuudet pienemmiksi vaatimuksiksi, jotka toteutettiin parhaan tietotaidon mukaisessa järjestyksessä. Näiden lisäksi valittiin toteutettavaksi asiakkailta saatuja pienehköjä vaatimuksia ja korjattiin ohjelmistotuotteissa havaittuja virheitä testausstrategiassa määritellyn luokittelun mukaisesti. Vastuu tuotesuunnitelman laatimisesta ja toteuttamisesta oli tuotehallinnalla. Tuotesuunnitelma laadittiin hyöty- ja resurssinäkökulmasta. Tuotekehitysjohtajan mukaan nykykäytännössä oli hyvää se, että se otti huomioon eri näkökulmia ja markkinoita. Lisäksi se oli suhteellisen joustava, jotta sitä voitiin tarkentaa julkaisusyklin aikana. Haasteina tuotekehitysjohtaja koki tuotevision puutteellisuuden pitkällä aikavälillä ja konsensukseen perustuvan priorisoinnin, joka johti suureen määrään pieniä vaatimuksia. Lisäksi hän koki, että vaatimuksien hyödyt arvioitiin subjektiivisesti minusta tuntuu periaatteella. Tämä aiheutti hänen mukaansa sen, että arvioinnin perusteena oli aina jokin jo koettu kipupiste eikä se, mitä tultaisiin kokemaan. Hänen mielestään priorisointia pitäisi pystyä tekemään proaktiivisesti katsomalla tulevaisuuteen eikä reaktiivisesti katsomalla taaksepäin. (Virtanen, T. 15.6.2010.)

Tuotekehitysjohtaja totesi, että vaatimusten priorisointi pitäisi suorittaa systemaattisesti tietyssä kontekstissa, tietyin aikavälein esim. seuraavalla tavalla: 1) tuotevisiotasoiset vaatimukset kerran vuodessa, 2) strategiatasoiset vaatimukset kaksi kertaa vuodessa strategiaprosessin yhteydessä, 3) tuotesuunnitelman vaatimukset julkaisusyklin alussa ja 4) yksittäisiä vaatimuksia pitäisi priorisoida koko ajan. Lisäksi hänen mielestään pitäisi pystyä suorittamaan tarkistuksia ja katselmointeja em. syklien sisällä. Tuotekehitysjohtaja totesi, että asiakkaat voisivat esim. äänestää pienistä vaatimuksista. Hän arvioi, että loppujen lopuksi vaatimukset olivat aina joko reaktiivisia asiakasvaatimuksia tai tuotesuunnitelmalla olevia proaktiivisia vaatimuksia. Toisaalta hän totesi, että kaikki asiakkaiden vaatimukset ovat todennäköisesti olleet jossain vaiheessa myös tuotesuunnitelmalla, joten asiakasvaatimuksille pitäisi allokoida kiinteä pooli tuotesuunnitelmaan niin, että kaikki vaatimukset olisivat jäljitettävissä tuotesuunnitelmasta. Lisäksi hän arvioi, että esim. asiakkuuksien rahallisen arvon, vaatimuksen riskitason ja kustannusten ymmärtäminen auttaisivat priorisoinnissa. Hän painotti, että formaalit priorisointipäätökset olisi tärkeää saada näkyviin yrityksessä sisäisesti. Ulkoisen kommunikoinnin hän koki erittäin haastavaksi, koska asiakkaat eivät yleensä ymmärtäneet, jos jokin tietty vaatimus on jätetty pois jostain tietystä näkökulmasta. (Virtanen, T. 15.6.2010.)

5.3.5 Suomen liiketoimintayksikön johtaja

Suomen liiketoimintayksikön johtaja koki, että vaatimusten priorisointi suoritettiin laatimalla tuotesuunnitelma, johon kerättiin viimehetken kriittiset vaatimukset. Hänen mielestä vaatimukset tulivat henkilöiltä, joita kuunneltiin eniten. Hän totesi, että vaatimusten priorisointi on

lyhytnäköistä johtuen yrityksen liiketoimintamallista. Käytännössä tämä tarkoitti hänelle pitkiä julkaisusyklejä, mikä mahdollisti pienien vaatimusten sisällyttämisen ohjelmistojulkaisuihin. Hän koki julkaisusuunnitelman hyvänä viestikapulana eri sidosryhmien välillä. Toisaalta hän ei ollut varma, edistivätkö kaikki mukaan otetut vaatimukset myyntiä. Lisäksi hän totesi, että vaatimuksia oli toteutettu usean asiakkaan ja yhteiskumppanin näkökulmasta, minkä vuoksi kaikille oli yritetty toteuttaa kaikkea aika ajoin. Käytännössä hän ei jaksanut seurata kaikkia yksittäisiä vaatimuksia, koska niitä oli niin paljon. (Erkheikki, M. 18.6.2010.)

Suomen liiketoimintayksikön johtaja uskoi, että julkaisusyklin pitäisi olla lyhyempi, jotta saadaan mahdollisimman nopeasti lisäarvoa asiakkaalle. Hän koki, että palvelutuotteiden avulla vaatimusten priorisointi voidaan toteuttaa paremmin, koska silloin yrityksessä tiedetään sisäisesti paremmin, mitä vaatimuksia pitää toteuttaa. Hän painotti, että prioriteettien katselmointeja pitäisi pystyä tekemään riittävän usein, jotta saadaan tehtyä päätöksiä tehokkaasti. Suomen liiketoimintayksikön johtajalle asiakasnäkökulma oli tärkein, koska muuten hän koki, että vaatimukset kehittyvät ainoastaan osittain valmiiksi ominaisuuksiksi tuotteeseen. Hän hahmotteli, että vaatimukset pitäisi saada kerättyä yhteen palveluliiketoiminnan näkökulmasta, koska se toisi mukanaan aidon asiakasnäkökulman. Hänelle oli tärkeää, että pienissä vaatimuksissa olisi esim. toteutuksen työmääräarvio ja riskiarvio, jolloin liiketoiminnan olisi helpompi ostaa vaatimuksia toteutettaviksi. Hän oli sitä mieltä, että vaatimukset pitäisi priorisoida systemaattisesti ja tarkoituksenmukaisella tavalla, jotta yrityksen sisäiset sidosryhmät näkisivät mahdollisimman ajoissa, mitä vaatimuksia ollaan toteuttamassa ohjelmistojulkaisuihin. (Erkheikki, M. 18.6.2010.)

5.3.6 Kansainvälisen liiketoimintayksikön johtaja

Kansainvälisen liiketoimintayksikön johtajalle priorisointi tarkoitti työpajoja, joissa liiketoiminnan esittämät vaatimukset kerättiin yhteen muutaman iteraation avulla. Em. työpajojen pohjalta tuotehallinta laati alustavan tuotesuunnitelman, jota tarkennettiin tarvittaessa. Hän koki, että vaatimuksia käsitellään systemaattisesti ja liiketoimintalähtöisesti. Toisaalta hän oli sitä mieltä, että yksittäisten vaatimusten linkittäminen strategiaan ei ollut läpinäkyvää. Lisäksi hänelle oli epäselvää, missä järjestyksessä vaatimukset toteutetaan tuotekehityksessä. Hän painotti, että tiettyjen avainasiakkaiden ja avainkumppaneiden vaatimukset eivät välttämättä edustaneet koko markkinoita, vaan ainoastaan asiakasta itseään. Hänen mielestään liian pitkä julkaisusykli aiheutti sen, että tuotejulkaisut sisälsivät liian paljon pieniä vaatimuksia, joiden kommunikointi oli puutteellista. (Ainasoja, A. 16.6.2010.)

Kansainvälisen liiketoimintayksikön johtajan mielestä priorisoinnin pitäisi olla sidottu yrityksen strategiatyöhön niin, että laajempia vaatimuskokonaisuuksia pystyisi käymään läpi heti uuden strategiakierroksen alussa. Hän painotti, että pitäisi pystyä identifioimaan asiakkaat, jotka edustatavat markkinoita ja ne vaatimuskokonaisuudet joilla erottaudutaan kilpailijoista. Hänen mielestään nykyiset työpajat olivat huutoäänestyksiä, mikä johti politikointiin valittavien vaatimusten suhteen. Hän koki, että priorisoinnissa pitäisi olla käytössä jokin systemaattinen menetelmä, jonka avulla pystyisi aidosti vertailemaan toteutettavia vaatimuksia. Lisäksi hän painotti sisäisen kommunikaation tärkeyttä. (Ainasoja, A. 16.6.2010.)

5.3.7 Liiketoiminnan kehitysjohtaja

Liiketoiminnan kehitysjohtaja oli sitä mieltä, että priorisoinnissa oli käytössä selkeä prosessi pitkälle julkaisusyklille. Hän totesi, että tuotesuunnitelma oli avainasemassa, koska yksittäisiä vaatimuksia ei voinut priorisoida. Toisaalta hän oli sitä mieltä, että kaikki vaatimukset pitäisi olla linkitettynä tuotesuunnitelmaan jollain tavalla, sillä tuotesuunnitelma mahdollistaa riittävän avoimen kommunikaation tuotehallintaan. Hän painotti, että oli tärkeää pystyä tekemään budjetääristä estimointia isoilla vaatimuskokonaisuuksilla. Hän painotti tuotehallinnan roolia tuotesuunnitelman laatimisessa. Tuotehallinnan pitäisi pystyä ymmärtämään yrityksen strategiaa, markkinoita ja asiakaskuntaa, sillä pelkkä fasilitaattorin rooli ei riittänyt. Hänen mielestään yrityksessä ei voitu huomioida aikakriittisiä vaatimuksia tarpeeksi hyvin, koska keskustelu käytiin yleensä vaatimusten tärkeydestä ja toisaalta julkaisusyklit olivat liian pitkiä. (Lehto, T. 15.6.2010.)

Liiketoiminnan kehitysjohtajan mielestä kaikki vaatimukset pitäisi kytkeä heti laajemman tason vaatimuskokonaisuuksiin. Kaikki vaatimuskokonaisuudet pitäisi olla nähtävissä tuotesuunnitelmalla, jotta yksittäisten vaatimusten priorisoinnin voisi suorittaa tarvittaessa yhden vaatimuskokonaisuuden sisällä. Lisäksi hän koki, että kaikki tuotesuunnitelmalla olevat vaatimuskokonaisuudet pitäisi pystyä priorisoimaan keskenään. Hän totesi, että vaatimusten priorisointi on haasteellista asiakkaille, koska siihen kuluu aikaa eikä ole takuuta siitä, että vaatimukset toteutetaan. Hän hahmotteli, että asiakkaille pitäisi luoda kauppapaikka, josta asiakkaat pystyisivät ostamaan haluamansa ohjelmistotuotteen ominaisuudet. Hän uskoi vahvasti, että systemaattisen priorisointimenetelmän avulla pieni yritys voisi saavuttaa kilpailuetua erityisesti silloin, kun julkaisusyklin pituus olisi optimaalinen. Vaatimusten prioriteettien kommunikointi yrityksen sisällä oli hänelle tärkeää päätöksenteon näkökulmasta. (Lehto, T. 15.6.2010.)

5.3.8 Palveluliiketoiminnan päällikkö

Palveluliiketoiminnan päällikölle priorisointi tarkoitti tuotesuunnitelman isoja liiketoiminnan vaatimuksia. Toisaalta hänelle oli epäselvää, mistä tietyt tuotesuunnitelmalla olevat vaatimukset olivat lähtöisin. Varsinainen priorisointi oli hänelle musta-aukko, koska prosessi ei ollut hänelle tarpeeksi läpinäkyvä. Hän uskoi, että isot vaatimuskokonaisuudet vievät tuotekokonaisuutta eteenpäin ja hänen mielestään oli hyvä, että tuotehallinta valmistelee tuotesuunnitelman yhdessä liiketoiminnan kanssa. Hän totesi, että pienemmät vaatimukset eivät päässeet tuotesuunnitelmalle. Hän oli turhautunut nykykäytäntöön, eikä viitsinyt kirjoittaa uusia vaatimuksia enää juuri lainkaan. Hän koki, ettei todellista priorisointikäytäntöä ollut olemassa ja oli erittäin turhautunut huutoäänestykseen. (Heinonen, L. 17.6.2010.)

Palveluliiketoiminnan päällikölle oli tärkeää, että yksittäiset käyttäjävaatimukset priorisoitaisiin riittävän usein ja systemaattisesti tulevaisuudessa, koska tällä hetkellä ne hukkuivat jonnekin. Hän oli varma, että yrityksen omien konsulttien kirjoittamat asiakasvaatimukset olivat laadukkaita, koska konsulteilla oli hyvä kokonaisnäkemys yrityksen ohjelmistotuotteiden käytöstä. Hänen mielestään tuotesuunnitelmalle pitäisi allokoida kiinteä pooli jatkokehitystä vaativille ominaisuuksille ja hän halusi tietää, mille tasolle jokin tietty ominaisuus oli viety ohjelmistotuotteissa käyttötapausten suhteen. Hän ehdotti, että poolissa voisi olla tarkemman tason ryhmiä kuten käytettävyys, suorituskyky, jne., jotta olisi helpompaa siirtää painopistettä johonkin haluttuun suuntaan. Hän oli varma, että systemaattinen priorisointimenetelmä toisi myös läpinäkyvyyttä ja selkeämpiä eroja vaatimusten välille nykykäytäntöön verrattuna. Yrityksen sisällä tapahtuvaa kommunikointia vaatimusten prioriteettien suhteen hän piti erittäin olennaisena, koska tällä hetkellä oli epäselvää oliko esim. jokin tietty käyttäjävaatimus tulossa mukaan seuraavaan julkaisuun vai ei. Hän uskoi, ettei asiakkaille kannata kommunikoida mitään muuta, kuin sellaiset vaatimukset, jotka toteutetaan varmuudella, koska muutoin asiakkaat turhautuisivat. (Heinonen, L. 17.6.2010.)

5.3.9 Avointen haastattelujen johtopäätökset

Kohdeyrityksen ohjelmistotuotteiden vaatimusten priorisointi konkretisoitui eri henkilöille tuotesuunnitelmana, joka laadittiin yhdessä liiketoiminnan kanssa järjestettävissä työpajoissa. Priorisointi tarkoitti käytännössä äänestämistä ja panostamista liiketoiminnan esille nostamien vaatimuskokonaisuuksien välillä. Tuotesuunnitelmalla käsiteltiin seuraavaan tuotejulkaisuun sisällytettäviä uusia vaatimuskokonaisuuksia.

Olemassa olevassa toimintatavassa koettiin hyväksi itse tuotesuunnitelma ja että käytössä oli formaali prosessi, jonka avulla tuotesuunnitelma muodostettiin. Tuotesuunnitelman laatiminen yrityksen eri sidosryhmien kanssa oli parantanut kommunikaatiota ja sen avulla seuraavan ohjelmistojulkaisun vaatimuskokonaisuuksia pystyttiin linkittämään strategiaan.

Ohjelmistotuotteiden vaatimusten priorisoinnissa ei ollut käytössä varsinaista formaalia priorisointimenetelmää. Yksittäiset vaatimukset ja vaatimuskokonaisuudet allokoitiin toteutettaviksi tuotehallinnan parhaan tietotaidon mukaisesti, koska varsinaista prioriteettijärjestystä ei määritelty. Läpinäkyvyys yksittäisten vaatimusten suhteen koettiin huonoksi, koska niistä ei nähnyt olivatko ne mukana seuraavassa tuotejulkaisussa. Käsitemallia eri vaatimustasojen välillä ei ollut käytössä ja vaatimusten jäljittäminen koettiin vaikeaksi. Tuotesuunnitelman laatimisessa suoritettava valinta aiheutti politikointia toteutettavien vaatimusten suhteen ja se koettiin huutoäänestyksenä. Tuotesuunnitelman aikaperspektiivi koettiin liian lyhyeksi, koska se laadittiin ainoastaan seuraavalle tuotejulkaisulle. Lisäksi olemassa olevien tuoteominaisuuksien jatkokehitys koettiin puutteelliseksi, koska tuotesuunnitelmalla käsiteltiin ainoastaan uusia ohjelmistotuotevaatimuksia.

Vaatimuksia oli tarve priorisoida eri konteksteissa ja eri ajankohdissa. Yksittäisen vaatimukset haluttiin linkittää suurempiin vaatimuskokonaisuuksiin ja päinvastoin. Tuotesuunnitelma haluttiin säilyttää keskitettynä kommunikaatiovälineenä. Vaatimusten priorisointi haluttiin niin joustavaksi, että se voitaisiin sovittaa kulloinkin käytössä olevaan tuotteiden julkaisusykliin.

Vaatimusten prioriteetteja ei haluttu kommunikoida asiakkaille ja loppukäyttäjille, jos niiden toteuttamisesta ei ollut täyttä varmuutta. Toisaalta myös asiakkaiden ja loppukäyttäjien vaatimukset haluttiin läpinäkyvämmiksi, jos ne oli aikomus toteuttaa. Lisäksi yrityksen sisäistä kommunikaatiota vaatimusten prioriteettien suhteen haluttiin kasvattaa entisestään, koska se vaikutti olennaisesti mm. monen sisäisen sidosryhmän ajankäytön suunnitteluun.

Kaikki haastateltavat olivat sitä mieltä, että vaatimuksia pitäisi priorisoida systemaattisesti tietyn mallin mukaisesti, jotta vaatimusten priorisoinnista tulisi läpinäkyvä käytäntö tarkoituksen mukaisille sidosryhmille.

5.3.9.1 Priorisointikäytäntöjen parantaminen

Avoimia haastatteluja peilattiin luvussa neljä esitettyihin olettamuksiin, joiden avulla johdettiin parannusehdotukset kohdeyrityksen priorisointikäytännöille.

Olennaisinta oli varmistua siitä, että ohjelmistotuote vastasi liiketoiminnan esittämiin vaatimuksiin, joten vaatimusten priorisoinnin tavoitteena oli saada aikaiseksi liiketoiminnan vaatimukset täyttävä ohjelmistojulkaisu.

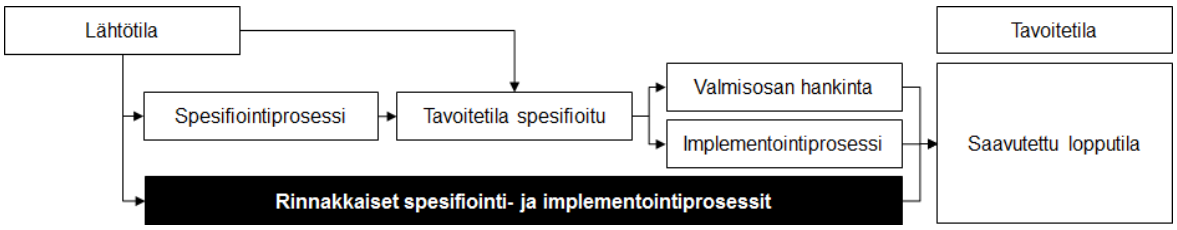
Vaatimusten systemaattisen ja läpinäkyvän priorisointiprosessin avulla varmistettiin, että kaikki kohdeyrityksen sisäiset sidosryhmät pystyivät kertomaan oman näkökulmansa, vaatimukset saatiin priorisoitua ja hyväksyttyä toteutettavat vaatimukset. Lisäksi priorisointiprosessin piti olla riittävän yksinkertainen ja nopea eri priorisointitasoilla.

6 Ohjelmistotuotteen vaatimusten priorisointimalli

Tässä kappaleessa käsitellään tutkimuksen kehittämishankkeen suunnittelu, toteutuksen tulokset ja toiminnan arviointi.

6.1 Suunnitelma

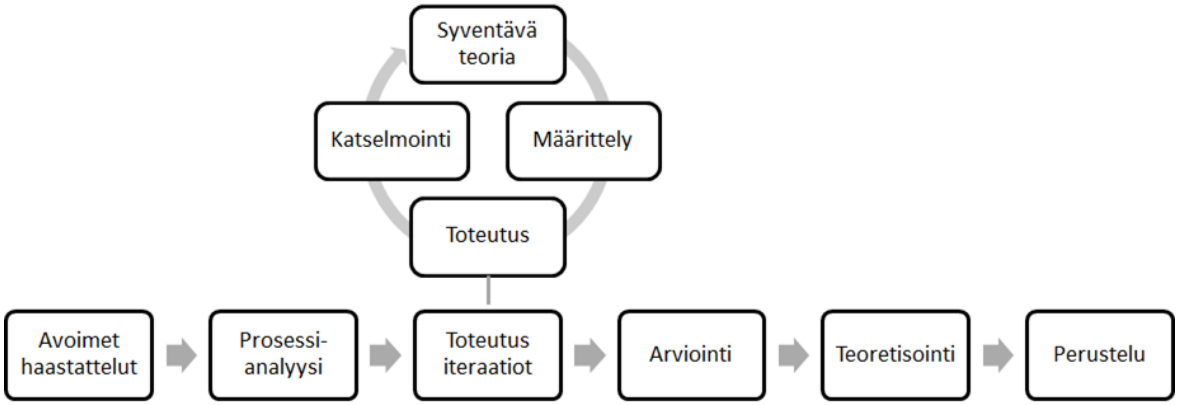
Suunnitteluvaiheessa hyödynnettiin Järvisen & Järvisen esittämä tapaa toteuttaa konstruktio, jonka mukaan prosessi eteni evolutionäärisesti eli tavoitetilan määrittely ja toteutus hoidettiin rinnakkain. Evolutionäärinen lähestymistavan tavoitteena oli iteratiivinen ja inkrementaali toteutus prototyyppien avulla. Prototyyppejä vertailtiin tavoitetilaan ja aina tarvittaessa toteutettiin uusi prototyyppi, jonka avulla pyrittiin pääsemään lähemmäksi tavoitetilaa. (Järvinen & Järvinen 2004, 108.)



Kuvio 20. Vaihtoehtoisia tapoja toteuttaa ratkaisu (Järvinen & Järvinen 2004, 108).

Tämän tutkimuksen tavoitetila oli vaatimusten priorisointimalli, jota voidaan hyödyntää systemaattisesti ja läpinäkyvästi kohdeyrityksessä, kun arvioidaan mitkä vaatimukset otetaan mukaan ohjelmiston seuraavaan julkaisuun ja mitkä vaatimukset voidaan jättää toteutettavaksi myöhemmin.

Tutkimuksen kehittämistehtävä suunniteltiin toteutettavaksi kuuden toteutusiteraation avulla aiempana kuvatun evolutionäärisen lähestymistavan mukaisesti. Iteraatiot suunniteltiin kahden viikon pituisiksi ja niiden aikana syvennettiin teoriaa, määriteltiin toteutus tarkoituksen mukaisella tarkkuudella ja suoritettiin katselmointi projektin ohjausryhmän kanssa. Toteutusiteraatioiden suhde tutkimusprosessin toteutusvaiheeseen on kuvattu kuviossa 21.



Kuvio 21. Tutkimusprosessin toteutusvaihe.

Tutkimuksen toteutus suunniteltiin soveltamalla Marchin ja Smithin esittelemää konstruktiivista viitekehystä, joka on esitetty taulukossa 6. Kyseisen viitekehyksen käyttöä puolsivat tässä tutkimuksessa hankittu teoreettinen - ja käytännöllinen tieto tutkimuksen kohteesta. Lisäksi todettiin, että oli tärkeää pystyä luomaan yksiselitteiset käsitteet priorisointimallille, jotta se voitiin kommunikoida eri käyttäjäryhmille tehokkaasti.

Kyseinen viitekehys on suunniteltu informaatioteknologian näkökulmasta huomioimalla keinotekoisten ilmiöiden mallintamisen. Viitekehyksessä tutkimuksen tulokset erotellaan riveille ja ne pohjautuvat suunnittelutieteen artefakteihin:

- Konstruktiot ovat rakenneosia, joista muut rakenteet koostuvat.
- Malli koostuu rakenneosista ja niiden välisistä yhteyksistä.
- Metodi on vaiheistus, jonka avulla tietty tehtävä voidaan suorittaa.
- Toteutus on tietyn ilmentymän toteutus käyttöympäristössä.

Tutkimuksen toiminnot on kuvattu viitekehyksen sarakkeina ja ne pohjautuvat luonnontieteen aktiviteetteihin:

- Artefakteja rakennetaan suorittamaan jokin tietty tehtävä.
- Artefakteja pitää arvioida, jotta nähdään miten hyvin ne toimivat.
- Teoretisointi selittää artefaktin ominaispiirteitä ja käyttäytymistä tietyssä ympäristössä.
- Perusteluiden avulla esitetään todisteita teorian toimivuudesta. (March & Smith 1995, 255-260.)

Tässä tutkimuksessa keskityttiin uuden priorisointimallin rakentamiseen, joten konstruktion arviointi, teoretisointi ja perustelut toteutettiin tarkoituksen mukaisella tasolla. Tässä tutkimuksessa suoritetut toimenpiteet ja tulokset on esitetty taulukossa 6 ja ne käydään läpi yksityiskohtaisemmin jäljempänä. Tutkimuksen toiminnan teoretisointi ja perustelu suoritettiin tarkoituk-

senmukaisella tarkkuudella, koska tutkimuksessa keskityttiin uuden priorisointimallin toteuttamiseen eikä jo toteutetun priorisointimallin arviointiin.

Taulukko 6. Konstruktiivinen tutkimuskehys (March & Smith 1995, 255).

		Tutkimuksen toiminta			
		Rakentaminen	**Arviointi**	**Teoretisointi**	**Perustelu**
Tutkimuksen tulokset	**Konstruktio**	Voidaanko käsitteet konseptoida?	Miten käsitteet toimivat?	Miksi käsitteet toimivat?	Miksi käsitteet pitäisi hyväksyä?
	Malli	Voidaanko käsitemalli rakentaa?	Miten käsitteet on yhdistetty?	Miksi malli toimii?	Miksi malli pitäisi hyväksyä?
	Metodi	Voidaanko tehtävät vaiheistaa?	Miten tehtäviä voi hyödyntää?	Miksi tehtävät on suoritettavissa?	Miksi tehtävät pitäisi hyväksyä?
	Toteutus	Voidaanko toteuttaa käyttöympäristöön?	Miten toteutus toimii käytännössä?	Miksi voidaan hyödyntää käytännössä?	Miksi toteutus pitäisi hyväksyä?

6.2 Tulokset

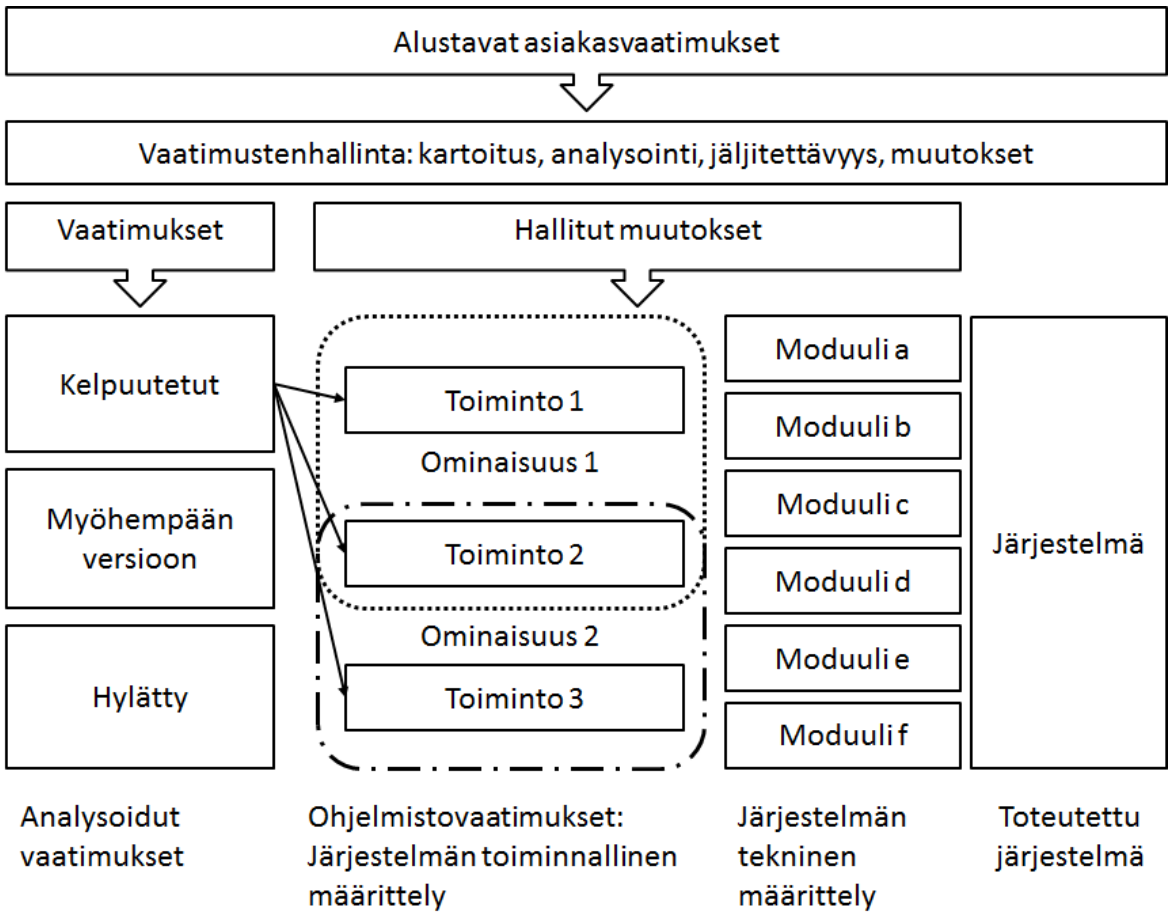
Tässä alaluvussa esitellään priorisointimallin rakentaminen hyödyntämällä edellä esitettyä konstruktiivista tutkimuskehystä.

6.2.1 Konstruktio

Avoimissa haastatteluissa ilmeni, että kohdeyrityksen käyttämät vaatimusten priorisointikäsitteet vaihtelivat. Haastatteluissa tulivat ilmi mm. seuraavat käsitteet: roadmap-vaatimukset, customer requirement-vaatimukset, pienet ominaisuudet, strategiset vaatimukset, requestit, ominaisuudet, vaatimukset, muut parannukset, small customer wishes, tekniset vaatimukset, roadmap-ominaisuudet, asiakkaiden vaatimukset, tuotehallinnan vaatimukset, liiketoiminnan isot vaatimukset, isot kokonaisuudet, pienet vaatimukset, pienet käyttötapaukset, viime hetken vaatimukset, roadmapin kriittiset vaatimukset, isomman tason vaatimukset ja backbone-

toiminnallisuudet. Toisaalta oli havaittavissa, että vaatimuksien alkuperää ei pystytty aina jäljittämään eikä pienempiä vaatimuksia pystytty aina kiinnittämään tuotesuunnitelmalla oleviin suurempiin kokonaisuuksiin.
Edellä kuvatun Marchin ja Smithin mallin mukaisesti aluksi hahmotettiin konsepti käytettävistä käsitteistä. Käsitteiden määritteleminen oli olennainen osa tutkimusta, koska ne määrittivät yhtenäisen termistön yksittäisille priorisointitehtäville ja toimenpiteille. Lisäksi käsitteiden määritteleminen auttoi vaatimusten jäljitettävyydessä eli vaatimusten aukottomassa linkittämisessä tuotesuunnitelmasta ohjelmistotuotteeseen.

Lähtökohtana käytettiin luvussa 2.2.3 kuvattua Wiegersin tapaa kuvata tietyt käsitteet eri käsitetasoille. Wiegersin kuvausta verrattiin kuviossa 22 esitettyyn Haikalan ja Märijärven tapaan kuvata tietyt käsitteet. Haikala ja Märijärvi erottelevat omassa kuvauksessaan asiakasvaatimukset, ohjelmistovaatimukset, ominaisuudet ja toiminnot. Kohdeyrityksen käyttämässä vaatimusten hallintajärjestelmässä toteutettavia ominaisuuksia ja niiden sisältämiä yksittäisiä toimintoja käsiteltiin fyysisesti samassa vaatimusmäärittelydokumentissa toiminnallisten ja teknisten määrittelyjen kanssa eikä tähän haluttu muutoksia, koska se olisi lisännyt vaatimusdokumenttien hallintaan liittyvää byrokratiaa. Käsitteet pyrittiin jäsentämään niin lähelle yrityksen jo käyttämää termistöä, kuin mahdollista, joten käsitteistö laadittiin sekä Wiegersin että Haikalan ja Märijärven kuvauksia hyödyntämällä.



Kuvio 22. Vaatimusten hallinta (Haikala & Märijärvi 2004, 93).

Vaatimusten priorisointi haluttiin kytkeä tiiviimmin mukaan kohdeyrityksen strategiaprosessiin. Lisäksi vaatimusten kommunikoinnissa haluttiin hyödyntää hyväksi havaittua tuotesuunnitelmaa. Haastattelujen perusteella todettiin, että vaatimuksia käsiteltiin käytännössä kolmesta eri näkökulmasta: 1) liiketoiminnan käsittelemät painopistealueet, 2) tuotehallinnan käsittelemät käyttäjävaatimukset ja 3) tuotehallinnan käsittelemät ohjelmistovaatimukset. Liiketoiminnan painopistealueet olivat korkeamman tason vaatimuskokonaisuuksia, jotka olivat johdettu suoraan strategiasta tai niihin haluttiin panostaa jostain toisesta näkökulmasta. Käyttäjävaatimukset olivat yleisen tason ominaisuuksia, joita käyttäjän piti pystyä suorittamaan ohjelmistotuotteella. Ohjelmistovaatimukset olivat toiminnallisuuksia, jotka pystyttiin toteuttamaan ohjelmistotuotteeseen. Lisäksi tuotesuunnitelmalle haluttiin sisällyttää esim. käytettävyyteen ja laatuun liittyviä ei-toiminnallisia vaatimuksia sekä ohjelmistotuotteiden tekniseen arkkitehtuuriin liittyvät järjestelmävaatimukset.

Asiakasvaatimusta ei haluttu käyttää käsitteenä sen takia, että kohdeyrityksessä se tarkoitti loppukäyttäjän haluamaa yksittäistä toimintoa, vaikka kirjallisuudessa se tarkoitti yleensä korkean tason asiakasongelmaa. Toisaalta liiketoiminnan vaatimus oli jo jossain määrin käytössä korkeimman tason vaatimuksina eikä sitä haluttu korvata asiakasvaatimuksella. Wiegersin esittämä

toiminnallinen vaatimus korvattiin Haikalan ja Märijärven käyttämällä ohjelmistovaatimuksella sen takia, ettei se aiheuttanut sekaannusta ei-toiminnallisten vaatimusten kanssa, koska jaottelu toiminnallisiin ja ei-toiminnallisiin vaatimuksiin tehtiin Wiegersin mukaan korkeammalla abstraktiotasolla. Em. valinnat käsitteiden suhteen tehtiin sen takia, että haluttiin välttää mahdolliset sekaannukset aikaisemmin käytössä olleiden käsitteiden ja termien kanssa.

Edellä esitettyjen perustelujen avulla laadittiin käsitteistö priorisointimallin pohjaksi. Käsitteet on kuvattu taulukossa 7. Käsitteet rajattiin niin, että mukaan otettiin vain sellaiset käsitteet, jotka liittyivät kohdeyrityksen priorisointityöhön.

Taulukko 7. Käsitteiden kuvaus.

Strategiaprosessi	Prosessi, jonka avulla luodaan ja toteutetaan strategia analysoimalla kohdeyrityksen ympäristön mahdollisuuksia sekä uhkia ja sisäisiä vahvuuksia sekä heikkouksia.
Tuotesuunnitelma	Tuoteominaisuuksien julkistussuunnitelma.
Liiketoimintavaatimus	Korkean tason vaatimuksia, jotka kuvaavat tavoitteita, joita halutaan saavuttaa.
Käyttäjävaatimus	Tehtäviä tai ominaisuuksia, joita käyttäjän pitää pystyä suorittamaan ohjelmistotuotteella.
Ohjelmistovaatimus	Toiminnallisuuksia ja toimintoja joita ohjelmistokehittäjät pystyvät toteuttamaan ohjelmistoon.
Ei-toiminnallinen vaatimus	Esim. ohjelmistotuotteen suorituskykyyn ja laatuun liittyvät vaatimukset.
Järjestelmävaatimus	Ohjelmistotuotteen kokonaisarkkitehtuuriin liittyvä ohjelmisto- tai laitteistovaatimus.

6.2.2 Malli

Käsitemallin rakentamisessa lähdettiin liikkeelle strategiaprosessin ja tuotesuunnitelman välisestä riippuvuudesta, koska se oli selkeä ja yksinkertainen. Tiettyyn strategiaprosessiin liittyi tietty tuotesuunnitelma ja tiettyyn tuotesuunnitelmaan liittyi tietty strategiaprosessi. Käytännössä tämä tarkoitti sitä, että uuteen strategiaprosessiin linkitettiin aina uusi tuotesuunnitelma.

Tuotesuunnitelmalle linkitettiin yksi tai useampi korkean tason liiketoimintavaatimus. Liiketoimintavaatimus oli mukana tuotannossa olevalla tuotesuunnitelmalla ja mahdollisesti myös uuteen strategiakierrokseen kytkettävällä tuotesuunnitelmalla.

Korkean tason liiketoimintavaatimukset sisälsivät yhden tai useamman käyttäjävaatimuksen. Käyttäjävaatimus liitettiin yhteen tuotesuunnitelmalla olevaan liiketoimintavaatimukseen. Käytännössä oli kuitenkin mahdollista, että yksi käyttäjävaatimus liittyi useampaan liiketoimintavaatimukseen, mutta selvyyden vuoksi käyttäjävaatimuksen liitettiin tarkoituksenmukaisimpaan liiketoimintavaatimukseen. Lisäksi todettiin, että matalammalla abstraktiotasolla olevat järjestelmävaatimukset ja ei-toiminnalliset vaatimukset piti pystyä liittämään korkean tason liiketoimintavaatimuksiin siitä huolimatta, että niihin ei liittynyt tuotesuunnitelmalla olevaa ohjelmistovaatimusta. Käytännössä tämä tarkoitti esimerkiksi sitä, että liiketoiminnan näkökulmasta seuraavassa tuotejulkaisussa oli tärkeää panostaa esim. jo toteutettujen ominaisuuksien käytettävyyteen, jotka kaipasivat parannuksia loppukäyttäjien näkökulmasta.

Käyttäjävaatimukseen liittyi yksi tai useampi toteutettava ohjelmistovaatimus. Yksi ohjelmistovaatimus liittyi yhteen käyttäjävaatimukseen. Käyttäjävaatimuksien laajuus vaihteli laajasta ominaisuudesta pieneen yksittäiseen toimintoon. Tarvittaessa laajat käyttäjävaatimukset pilkottiin pienempiin ohjelmistovaatimuksiin esim. siitä syystä, että ne pystyttiin toteuttamaan rinnakkain. Käytännössä oli tietenkin mahdollista, että ohjelmistovaatimus liittyi moneen käyttäjävaatimukseen, mutta priorisointikontekstissa tärkeintä oli pystyä linkittämään se tarkoituksenmukaisimpaan käyttäjävaatimukseen kommunikaation ja jäljitettävyyden näkökulmasta. Toisaalta kohdeyrityksen käyttämässä vaatimusten hallintajärjestelmässä toisiinsa liittyvät ohjelmistovaatimukset linkitettiin toisiinsa toteutuksen näkökulmasta, koska järjestelmä- ja ohjelmistovaatimukset sekä ei-toiminnallisiin vaatimuksiin liittyvät toimenpiteet kirjattiin erillisille ohjelmiston vaatimusmäärittelydokumenteille.

Ohjelmistovaatimuksiin saattoi liittyä jokin tietty tuotesuunnitelmalla oleva järjestelmävaatimus tai ei-toiminnallinen vaatimus. Toisaalta oli mahdollista, että tuotesuunnitelmalla oli ainoastaan jo toteutettuihin ominaisuuksiin liittyviä järjestelmä- ja ei-toiminnallisia vaatimuksia. Ohjelmiston vaatimusmäärittelydokumenttia ei sisällytetty käsitemalliin, koska vaatimukset pystyttiin priorisoimaan jo edellisellä abstraktiotasolla.

6.2.2.1 Käsitemalli

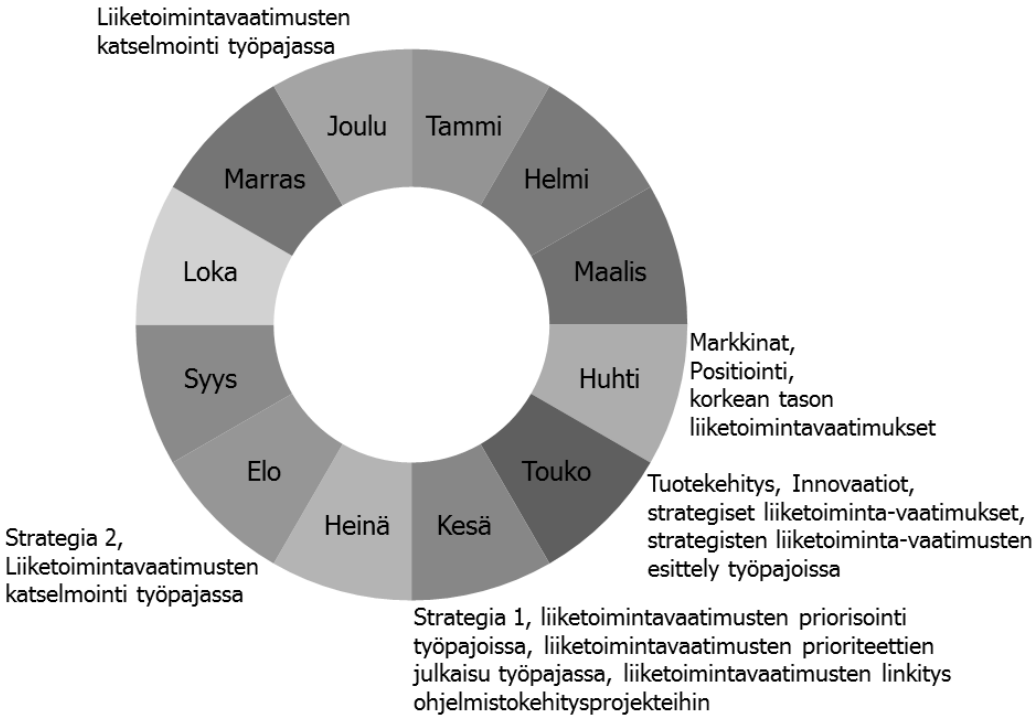
Vaatimusten priorisointiin liittyvä käsitemalli pidettiin mahdollisimman yksikertaisena, jotta se oli kommunikoitavissa tehokkaasti eri sidosryhmille. Kuviossa 23 esitetty käsitemalli laadittiin luokkadiagrammin avulla. Kerrannaisuudet määriteltiin sen takia, että pystyttiin hahmottamaan vaatimusten jäljitettävyys strategiaprosessista toteutettavaan ohjelmistovaatimukseen ja päinvastoin.

Kuvio 23. Käsitemalli.

6.2.3 Metodi

Vaatimusten priorisoinnin vaiheistus laadittiin kohdeyrityksen strategiaprosessin näkökulmasta kytkemällä vaiheistus aluksi kuviossa 24 esitettyyn strategiaprosessin vuosikelloon. Kuviossa on esitetty ainoastaan tässä tutkimuksessa kehitettyyn priorisointimalliin liittyvät asiat. Strategiaprosessin mukaisesti ensin tarkasteltiin kohdeyrityksen markkinoita ja yrityksen sijoittautumista markkinoille huhtikuussa. Samassa yhteydessä luotiin myös uusi karkean tason tuotesuunnitelma, jonne luonnosteltiin ensimmäisiä korkean tason liiketoimintavaatimuksia. Tuotesuunnitelma laadittiin tässä vaiheessa noin kolmen vuoden perspektiivillä erityisesti tuotevision näkökulmasta ja siitä vastasi kohdeyrityksen johtoryhmä. Tuotevision pohjana käytettiin tuotannossa olevaa tuotesuunnitelmaa, jotta nähtiin, missä vaiheessa tuotesuunnitelma oli tällä

hetkellä. Toisaalta haluttiin nähdä mahdolliset muutokset edellisenä vuotena suunniteltuun tuotevisioon. Näin ollen tuotevisio päivitettiin ajan tasalle tässä vaiheessa. Tuotehallinta oli vastuussa tuotesuunnitelman laatimisesta ja sen ylläpitämisestä.



Kuvio 24. Strategiaprosessin vuosikello.

Kohdeyrityksen johtoryhmä jatkoi strategiaprosessin mukaista työtä tutkimus- ja kehitystyön sekä erilaisten innovaatioiden parissa toukokuussa. Tässä yhteydessä tunnistettiin strategiset liiketoimintavaatimukset ja ne kirjattiin tuotesuunnitelmaan. Strategisten liiketoimintavaatimusten laajuus oli lähtökohtaisesti yksi strategiakierros eli yksi vuosi. Kesäkuussa suoritettiin liiketoimintavaatimusten priorisointi ja niiden linkitys ohjelmistokehitysprojekteihin. Elo- ja joulukuussa suoritettiin liiketoimintavaatimusten katselmointi ja tehtiin mahdolliset tarkennukset ja korjaavat toimenpiteet. Strategiaprosessin vaiheiden kytkeminen eri vaatimustasoille on kuvattu yksityiskohtaisemmin jäljempänä.

6.2.3.1 Priorisointimallin prosessi

Eri priorisointimenetelmät valittiin tässä tutkimuksessa kehitettyyn priorisointimalliin painottamalla menetelmän käytettävyyttä, helppoutta ja itse priorisointitapahtuman tavoitetta. Toisaalta priorisointimalli haluttiin toteuttaa modulaarisesti, jotta siinä hyödynnettäviä menetelmiä pystyttäisiin vaihtamaan tarvittaessa tietyssä priorisointikontekstissa. Lisäksi kohdeyrityksessä uskottiin, että käytännön priorisointityö mukauttaisi ja sopeuttaisi priorisointimallin kohdeyri-

tyksen toimintakulttuuriin, joten priorisointimallin piti olla joustava. Priorisointimenetelmien kuvausten ja yleisten ominaisuuksien vertailu on liitteessä 1.

Liiketoimintavaatimusten kohdalla oli olennaista pystyä suorittamaan aito vertailu eri liiketoimintavaatimusten välillä. Liiketoimintavaatimuksia oli lukumäärällisesti suhteellisen pieni määrä verrattuna muihin vaatimuksiin. Analyyttinen hierarkiaprosessi valittiin liiketoimintavaatimusten priorisointiin, koska vertailtavat liiketoimintavaatimukset edustivat korkeimman abstraktiotason vaatimuksia ja näin ollen ne kuuluivat samaan osakokonaisuuteen. Korkeimman tason vaatimuksia oli yleensä noin 10 – 15, joten niiden lukumäärä oli riittävän suppea, kun huomioitiin AHP:n vaatima vertailujen lukumäärä. Lisäksi haluttiin, että korkeimman tason liiketoimintavaatimuksista keskusteltiin huomioimalla niiden suhteet toisiinsa. Tavoitteena oli järjestää liiketoimintavaatimukset niiden arvon mukaiseen tärkeysjärjestykseen.

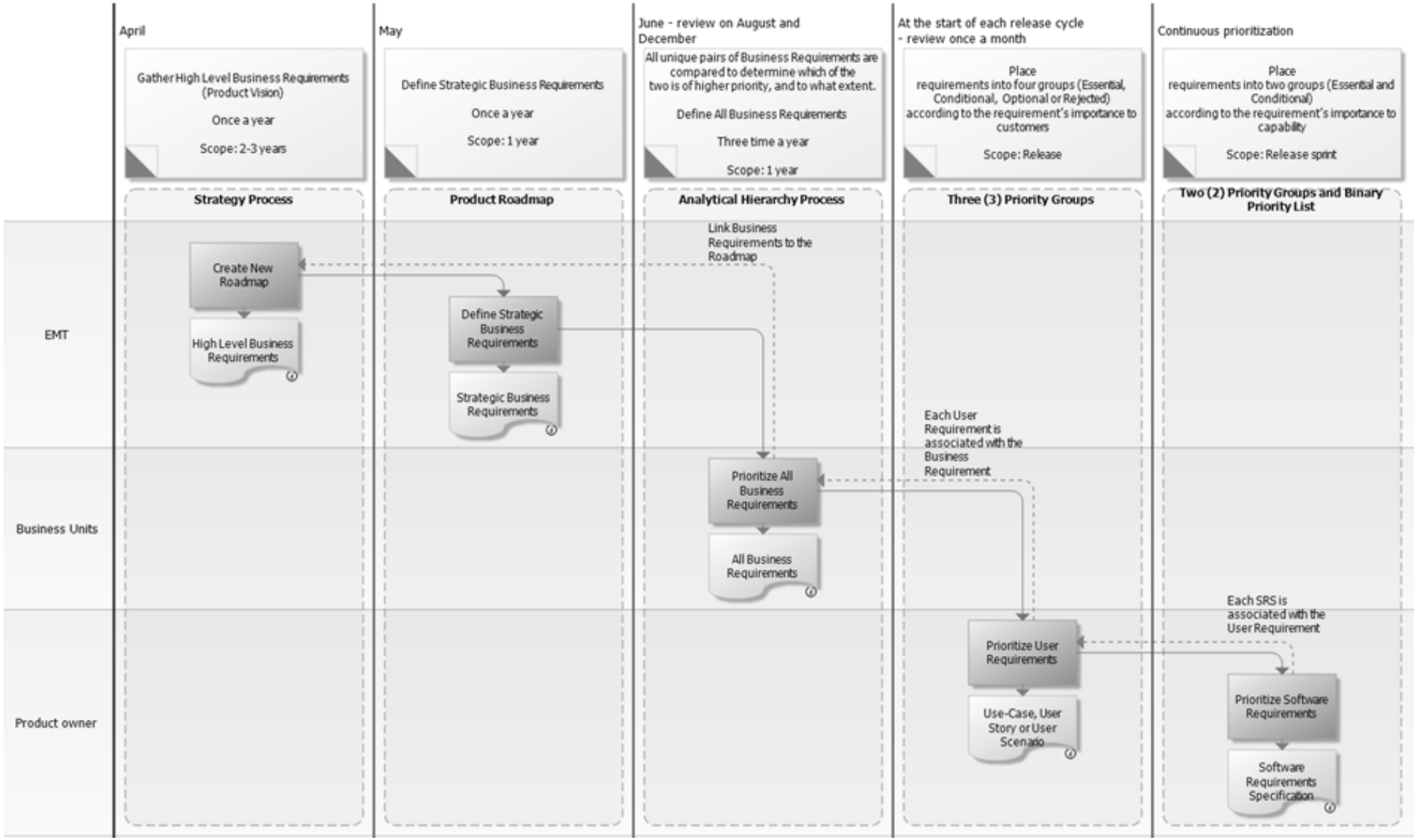
Ryhmittely valittiin käyttäjävaatimusten priorisointiin, koska haluttiin erotella mahdollisimman nopeasti ja tehokkaasti välttämättömät käyttäjävaatimukset, jotka piti saada toteutettua, ennen kuin toteutettu ominaisuus voitiin julkaista uudessa ohjelmistojulkaisussa. Uusia käyttäjävaatimuksia kirjattiin kohdeyrityksen käytössä olleeseen vaatimustenhallintajärjestelmään päivittäin, joten niiden käsittely piti olla mahdollisimman tehokasta. Lisäksi käyttäjävaatimusten laajuus vaihteli pienestä yksittäisestä toiminnosta laajoihin ja monimutkaisiin toiminnallisuuksiin. Toisaalta tuotehallinta käsitteli kaikki käyttäjävaatimukset, joten lähtökohtaisesti voitiin todeta, että käyttäjävaatimuksia käsittelevillä henkilöillä oli hyvä kokonaiskäsitys käyttäjävaatimuksista. Tiettyyn liiketoimintavaatimukseen liittyvien välttämättömien käyttäjävaatimusten keskinäisellä prioriteetilla ei ollut merkitystä, koska kaikki välttämättömät käyttäjävaatimukset piti toteuttaa. Käyttäjävaatimukset linkitettiin liiketoimintavaatimuksiin, jotta ne voitaan jäljittää ylhäältä alas ja päinvastoin. Tämä mahdollisti eri abstraktiotasolla olevien vaatimusten läpinäkyvyyden. Toisaalta se auttoi varmistamaan, että liiketoimintavaatimusten prioriteetit olivat halutut, koska liiketoiminta näki käyttäjävaatimukset, jotka olivat liiketoimintavaatimusten taustalla.

Osa käyttäjävaatimuksista voitiin luokitella puhtaiksi ohjelmistovaatimukseksi sellaisenaan, jos kyseessä oli yksittäinen pieni toiminto, jonka toiminnallisuus oli kuvattu riittävällä tarkkuudella. Osa käyttäjävaatimuksista oli niin isoja kokonaisuuksia, että ne piti pilkkoa pienemmiksi ohjelmistovaatimusosakokonaisuuksiksi. Ohjelmistovaatimukset olivat käytännössä toiminnallisuuksia ja toimintoja, joita ohjelmistokehittäjät pystyivät toteuttamaan ohjelmistoon. Näin ollen oli tärkeää, että myös ohjelmistovaatimuksista pystyttiin luokittelemaan välttämättömät toiminnallisuudet, mitkä piti toteuttaa. Toisaalta ohjelmistovaatimusten lukumäärä saattoi olla suhteellisen korkea eli käytännössä puhuttiin sadoista toteutettavista ohjelmistovaatimuksista,

mikäli julkaisusykli oli noin 12 kuukautta. Näin ollen ohjelmistovaatimusten priorisoinnissa päätettiin käyttää prioriteettiluokkien ja tasapainotetun puolitushaun yhdistelmää.

Yksityiskohtainen priorisointimenetelmien helppouden vertailu on esitetty liitteessä 2 ja priorisointimenetelmien käytettävyyden vertailu on esitetty liitteessä 3. Vertailuun otettiin mukaan viisi eri menetelmää, jotka hyödynsivät järjestys-, välimatka- ja suhdeasteikkoa priorisoinnissa. Lisäksi vertailtavat menetelmät edustivat ryhmitteleviä, näkökulmia painottavia ja panostukseen pohjautuvia menetelmiä.

Priorisointimallin prosessi eli vaiheistus on esitetty kuviossa 25. Prosessin kolme ensimmäistä vaihetta kytkettiin aiemmin esiteltyyn strategiaprosessin vuosikelloon, jolloin luotiin uusi tuotesuunnitelma seuraavalle strategiakierrokselle, määriteltiin strategiset liiketoimintavaatimukset ja tarkennettiin muut liiketoimintavaatimukset. Käyttäjä- ja ohjelmistovaatimusten priorisointi toteutettiin suhteessa julkaisusykliin.



Kuvio 25. Priorisointimallin prosessi.

6.2.3.2 Liiketoimintavaatimukset

Varsinainen liiketoimintavaatimusten priorisointi suoritettiin ensimmäisen kerran kesäkuussa käyttämällä analyyttistä hierarkiaprosessia. Pohjana käytettiin johtoryhmän aikaisemmin määrittelemiä strategisia liiketoimintavaatimuksia. Tässä vaiheessa mukaan otettiin myös eri liiketoimintayksiköiden liiketoimintavaatimukset. Kaikki liiketoimintavaatimukset priorisoitiin lähtökohtaisesti vuodeksi eteenpäin tuotesuunnitelmalle, mutta tuotesuunnitelma katselmoitiin liiketoiminnan kanssa kaksi kertaa syklin aikana. Tässä vaiheessa tuotehallinta oli linkittänyt liiketoimintavaatimuksiin konkreettiset käyttäjävaatimukset, joka mahdollisti käytännönläheisen arvioinnin. Liiketoimintayksikkökohtaisiin priorisointisessioihin osallistui useita edustajia liiketoimintayksiköstä. Tuotehallinta keräsi tulokset yhteen ja ne esiteltiin työpajassa, johon osallistui liiketoimintayksiköiden vastuuhenkilöt. Liiketoimintavaatimuksille suoritettiin katselmointi kesän jälkeen elokuussa. Analyyttistä hierarkiaprosessia ei suoritettu uudestaan elokuussa, koska sille ei nähty tarvetta. Kolmas katselmointi liiketoimintavaatimuksille suoritettiin vielä joulukuussa. Joulukuussa oli vielä mahdollista tarkistaa liiketoimintavaatimusten prioriteetit, ennen kuin tuotannossa oleva tuotesuunnitelma korvattiin uudella tuotesuunnitelmalla. Liiketoimintavaatimukset priorisoitiin tuotehallinnan järjestämissä liiketoimintayksikkökohtaisissa työpajoissa.

6.2.3.3 Käyttäjävaatimukset

Tuotehallinta aloitti käyttäjävaatimusten spesifioinnin käytännössä heti huhtikuussa, kun ensimmäiset strategiset liiketoimintavaatimukset oli tunnistettu. Käytännössä tämä tarkoitti sitä, että strategisiin liiketoimintavaatimuksiin linkitettiin välttämättömiä käyttäjävaatimuksia, jotka oli pakko toteuttaa, mikäli strateginen liiketoimintavaatimus haluttiin tyydyttää. Lisäksi tuotehallinta määritteli ja kokosi ehdollisia ja valinnaisia käyttäjävaatimuksia linkittämällä ne liiketoimintavaatimuksiin tai hylkäsi käyttäjävaatimuksia huhti- ja toukokuun aikana. Tuotehallinnan tavoitteena oli hahmottaa ja kuvata hieman suurempia käyttäjävaatimuskokonaisuuksia priorisoinnin tueksi. Apuna käytettiin markkina- ja kilpailija-analyysejä sekä asiakkaiden ja loppukäyttäjien esittämiä käyttäjävaatimuksia, jotka oli kirjattu kohdeyrityksen vaatimustenhallintajärjestelmään. Tuotehallinta kuvasi tarkemmat käyttäjätarinat ja käyttötapaukset tarkemmalla tasolla, kun käyttäjävaatimukset oli ryhmitelty välttämättömiksi.

Käyttäjävaatimukset piti olla priorisoituna jokaisen julkaisusyklin alussa. Metodia suunniteltaessa huomioitiin myös se, että metodin pitää tukea sekä lyhyitä että pitkiä julkaisusyklejä. Käytännössä tämä tarkoitti sitä, että käyttäjävaatimuksia priorisoitiin jatkuvasti sen jälkeen, kun tiedettiin liiketoimintavaatimukset. Lisäksi käyttäjävaatimukset priorisoitiin eksplisiittisesti ker-

ran kuukaudessa, koska kohdeyrityksen julkaisusyklin voitiin olettaa lyhenevän tulevaisuudessa mm. haastatteluissa esille tulleiden kommenttien perusteella.

6.2.3.4 Ohjelmistovaatimukset

Ohjelmistovaatimukset olivat toiminnallisuuksia ja toimintoja, joita ohjelmistokehittäjät toteuttivat ohjelmistoon. Lähtökohtaisesti ohjelmistovaatimukset priorisoitiin välttämättömiin ja ehdollisiin, koska oli tärkeää saada kaikki välttämättömät ohjelmistovaatimukset toteutukseen tuotekehittäjille mahdollisimman pian. Lisäksi oli tärkeää tietää, milloin tietty käyttäjävaatimus oli toteutettu välttämättömien ohjelmistovaatimusten osalta, jotta voitiin siirtyä toteuttamaan toista käyttäjävaatimusta.

Ehdollisia käyttäjävaatimuksia toteutettiin ainoastaan, jos jäi aikaa. Ehdolliset ohjelmistovaatimukset haluttiin saada priorisoitua suurin piirtein oikeaan järjestykseen, koska niitä oli niin paljon. Näin ollen ne priorisoitiin puolitushaun (BST) avulla. Tuotehallinta priorisoi käyttäjävaatimuksiin liittyvät ehdolliset ohjelmistovaatimukset.

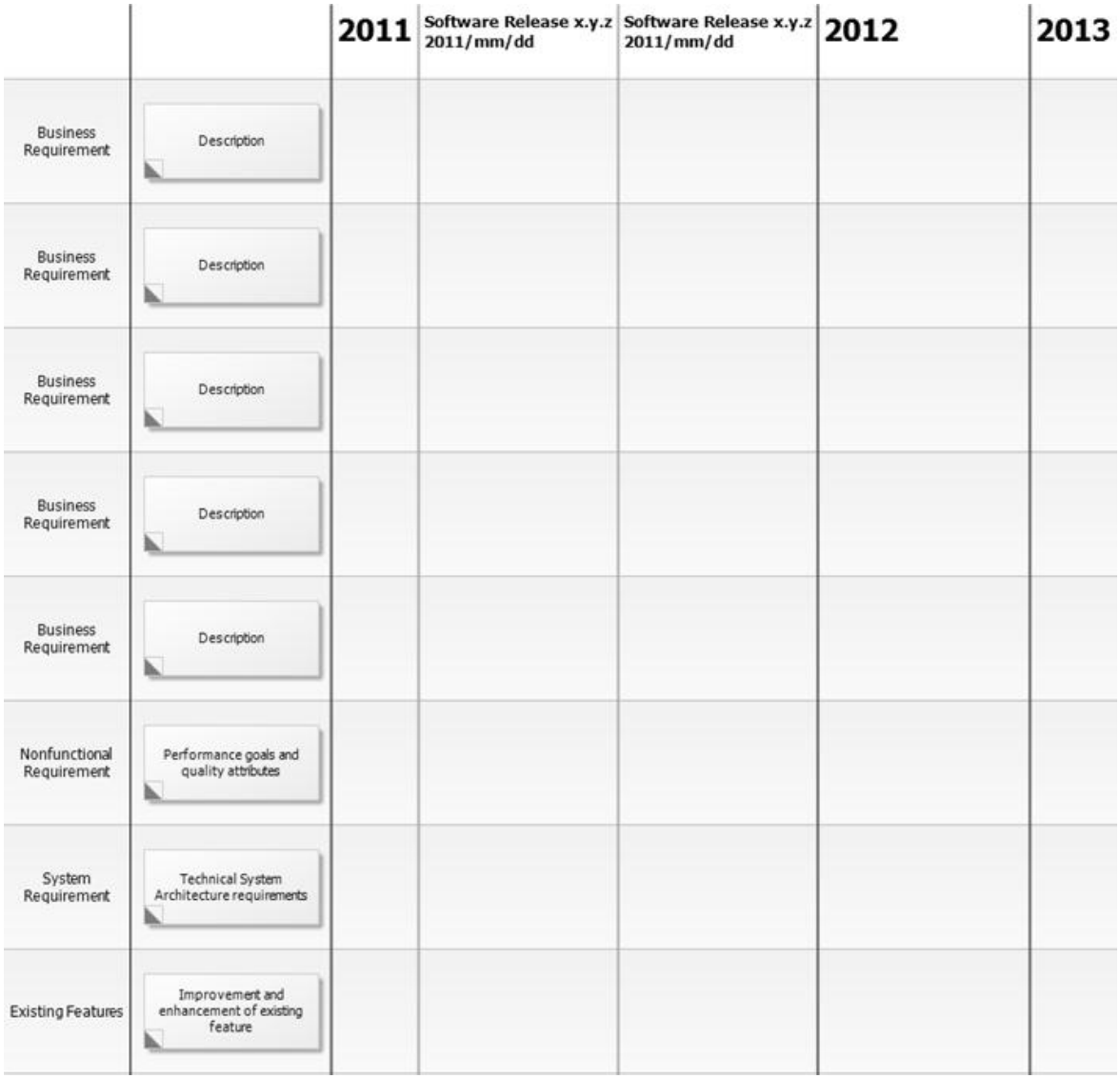
Tuotekehitys toteutti tekniset spesifikaatiot ohjelmistovaatimuksille ja arvioi mahdolliset toteutukseen liittyvät riskit sekä työmääräarvion. Mikäli ohjelmistovaatimusten toteuttamiseen sisältyi paljon riskejä ja kustannuksia ne pystyttiin esittämään liiketoiminnalle erikseen järjestettävässä katselmoinnissa tai viimeistään joulukuussa järjestettävässä strategiaprosessin mukaisessa katselmoinnissa. Liiketoiminta ohjeistettiin miettimään miten tärkeää oli saada ohjelmistovaatimus mukaan seuraavaan ohjelmistojulkaisuun ja toisaalta miten paljon aiheutui haittaa, jos sitä ei otettu mukaan seuraavan ohjelmistojulkaisuun.

6.2.4 Toteutus

Tässä alaluvussa kuvataan priorisointimallin käytännöntoteutus kohdeyrityksessä.

6.2.4.1 Tuotesuunnitelma

Alustava tuotesuunnitelma luotiin huhtikuussa alussa kuviossa 26 esitetyllä tavalla.



Kuvio 26. Tuotesuunnitelma.

Tuotesuunnitelmalle hahmoteltiin aluksi korkean tason strategisia liiketoimintavaatimuksia, jotka eriteltiin horisontaaleiksi altaiksi. Tuotesuunnitelmalle kuvattiin myös ohjelmistotuotteiden alustava julkaisuaikataulu ja pidemmän aikavälin tuotevisio. Lähtökohtaisesti tuotevisio pyrittiin laatimaan kolmen vuoden päähän nykyhetkestä. Lisäksi tuotesuunnitelmalle lisättiin kolme kiinteää allasta, jotka olivat: 1) ei-toiminnalliset vaatimukset, 2) järjestelmävaatimukset ja 3) olemassa olevat ominaisuudet. Ei- toiminnalliset vaatimukset tarkoittivat ohjelmistotuotteiden laatuun ja suorituskykykyyn liittyviä vaatimuksia, kuten käytettävyys- ja vasteaikavaatimukset. Järjestelmävaatimukset tarkoittavat ohjelmistotuotteiden tekniseen arkkitehtuurin liittyviä vaatimuksia, kuten tekniset alustavaatimukset. Olemassa olevat ominaisuudet tarkoittivat jo toteutettujen vaatimusten laajennoksia ja parannuksia.

Kiinteiden altaiden käyttö mahdollisti paremman läpinäkyvyyden mm. sellaisille liiketoimintavaatimuksille, jotka oli esitelty aikaisemmissa tuotejulkaisuissa esim. kilpailuedun näkökulmas-

ta. Tällöin kaikkia käyttäjävaatimuksia ei yleensä ehditty toteuttaa seuraavaan tuotejulkaisuun eikä kaikki käyttäjävaatimukset olleet tässä vaiheessa edes tuotehallinnan tiedossa. Näin ollen olemassa olevia ominaisuuksia piti pystyä kehittämään vastaamaan loppukäyttäjien tarpeisiin. Lisäksi oli tärkeää, että liiketoiminta pystyi painottamaan ei-toiminnallisia vaatimuksia esim. silloin, kun loppukäyttäjät ilmaisivat tyytymättömyytensä käytettävyyden suhteen. Käytännössä liiketoiminnalla oli nyt mahdollisuus todeta, että seuraavan tuotejulkaisun ainoa liiketoimintavaatimus on parempi laatu ja suorituskyky. Toisaalta tuotehallinta pystyi nostamaan esille ohjelmistotuotteiden tekniseen arkkitehtuuriin liittyvät kehitystarpeet niin, että ne näkyivät myös liiketoiminnalle.

6.2.4.2 Liiketoimintavaatimusten priorisointi

Tuotehallinta organisoi eri liiketoimintayksiköille priorisointityöpajat toukokuun puolestavälistä eteenpäin. Työpajat olivat liiketoimintayksikkökohtaisia ja niissä esiteltiin strategiset liiketoimintavaatimukset osallistujille. Osallistujia pyydettiin valmistautumaan työpajoihin selvittämällä mm. loppuasiakkaiden, loppukäyttäjien, partnereiden ja markkinoiden liiketoimintavaatimuksia. Työpajoissa tavoitellut liiketoimintavaatimukset sisälsivät yleensä useita tarkemman tason käyttäjävaatimuksia, mutta siitä ei ollut haittaa, koska tuotehallinta joutui joka tapauksessa tunnistamaan tarkemman tason käyttäjävaatimuksia, jotka linkitettiin liiketoimintavaatimuksiin.

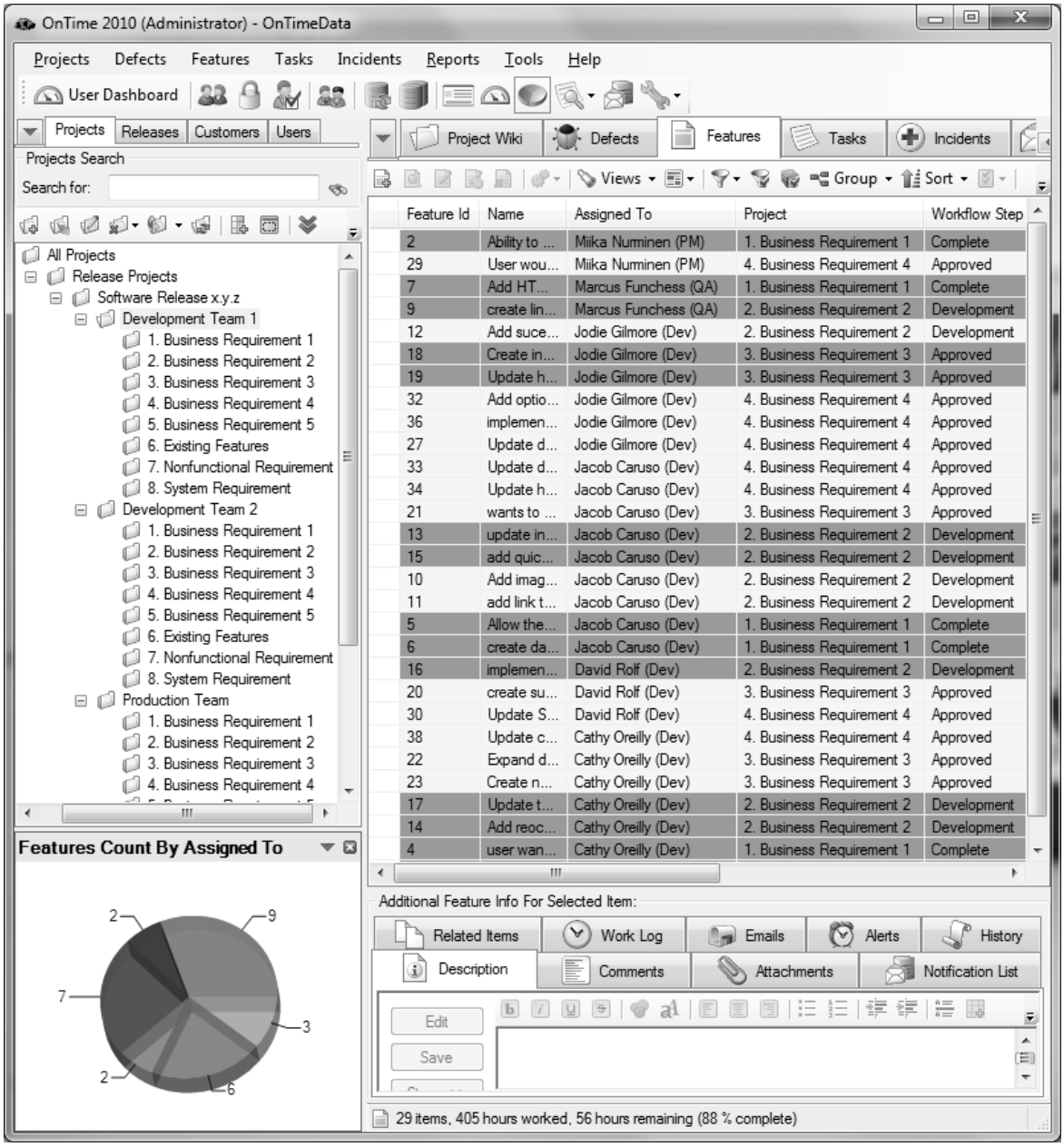
Tuotehallinta keräsi kaikki työpajoissa esille tulleet liiketoimintavaatimukset yhteen ja kokosi ne yhteen strategisten liiketoimintavaatimuksien kanssa kuviossa 27 esitettyyn MS Excel matriisiin. Näin ollen matriisissa yhdistettiin strategiset liiketoimintavaatimukset ja kaikki muut liiketoimintavaatimukset. Lisäksi tuotehallinta linkitti liiketoimintavaatimuksiin tunnistamiaan välttämättömiä käyttäjävaatimuksia. Käyttäjävaatimusten priorisointi käytännössä esitellään jäljempänä.

	A	B	C	D	E
1			**Definition**	**Intensity of importance**	**Explanation**
2			Equal importance	1	Two factors contribute equally to the objective.
3			Somewhat more important	3	Experience and judgement slightly favour one over the other.
4			Much more important	5	Experience and judgement strongly favour one over the other.
5			Very much more important	7	Experience and judgement very strongly favour one over the other. Its importance is demonstrated in practice.
6			Absolutely more important	9	The evidence favouring one over the other is of the highest possible validity.
7					
8	**Criteria**				
9	**A**	**B**	**More Important**	**Intensity**	
10	Business requirement 1	Business requirement 2	a	5	
11	Business requirement 1	Business requirement 3	a	5	
12	Business requirement 1	Business requirement 4	a	9	
13	Business requirement 1	Business requirement 5			
14	Business requirement 1	Business requirement 6			
15	Business requirement 1	Business requirement 7			
16	Business requirement 1	Business requirement 8			
17	Business requirement 2	Business requirement 3			
18	Business requirement 2	Business requirement 4			
19	Business requirement 2	Business requirement 5			
20	Business requirement 2	Business requirement 6			
21	Business requirement 2	Business requirement 7			
22	Business requirement 2	Business requirement 8			
23	Business requirement 3	Business requirement 4			
24	Business requirement 3	Business requirement 5			
25	Business requirement 3	Business requirement 6			
26	Business requirement 3	Business requirement 7			
27	Business requirement 3	Business requirement 8			
28	Business requirement 4	Business requirement 5			
29	Business requirement 4	Business requirement 6			
30	Business requirement 4	Business requirement 7			
31	Business requirement 4	Business requirement 8			
32	Business requirement 5	Business requirement 6			
33	Business requirement 5	Business requirement 7			
34	Business requirement 5	Business requirement 8			
35	Business requirement 6	Business requirement 7			
36	Business requirement 6	Business requirement 8			
37	Business requirement 7	Business requirement 8			

Kuvio 27. Liiketoimintavaatimusten priorisointi.

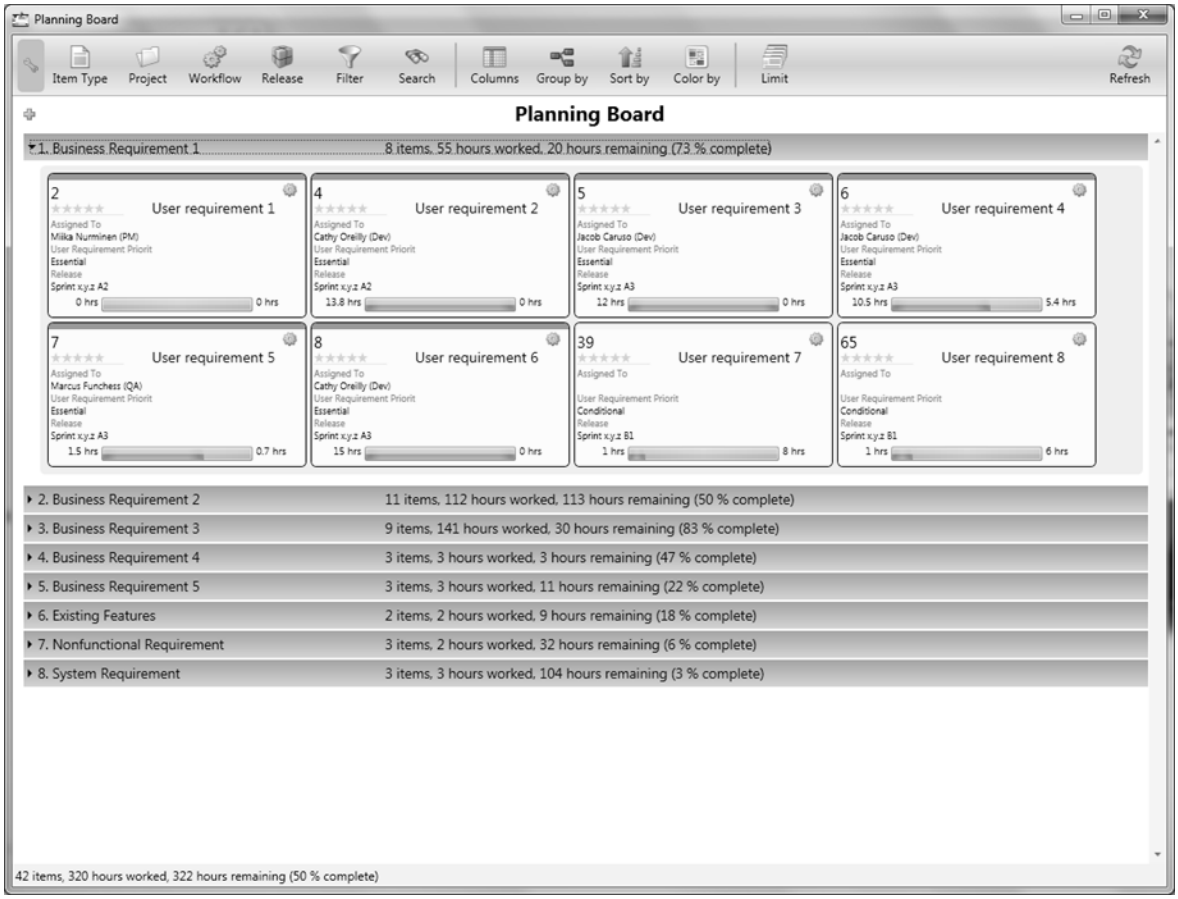
Liiketoimintavaatimusten priorisointi suoritettiin kuviossa 27 kuvatun matriisin avulla kesäkuun alusta alkaen. Varsinainen priorisointimatriisi toteutettiin liiketoiminnan näkökulmasta siten, että vältyttiin murtolukujen käytöltä. Toisaalta liiketoimintavaatimusten välille haluttiin todellisia eroja, joten liiketoiminnan henkilöt koulutettiin lähtökohtaisesti käyttämään intensiteettejä yksi, viisi ja yhdeksän suorittaessaan pareittaisia vertailuja liiketoimintavaatimuksille. Lisäksi haluttiin olla varmoja, että liiketoiminta pystyi keskittymään vaatimusparien arviointiin sen sijaan, että he olisivat hahmotelleet rivien ja sarakkeiden merkitystä pareittaisessa vertailussa. Tuotehallinta organisoi liiketoimintayksikkökohtaiset työpajat, joissa liiketoimintavaatimukset priorisoitiin. Osa liiketoimintayksiköistä halusi suorittaa priorisoinnin henkilökohtaisesti ja osa keskitetysti. Tämä oli mahdollista, koska tulokset konsolidoitiin keskitettyyn laskenta-Exceliin. Työpajoissa oli esillä tuotesuunnitelma, jossa esiteltiin priorisoitaviin liiketoimintavaatimuksiin liittyviä välttämättömiä käyttäjävaatimuksia. Työpajoihin osallistuneet henkilöt pystyivät näin ollen näkemään helpommin, mitä liiketoimintavaatimus tarkoitti käytännössä.

Työpajojen jälkeen tuotehallinta konsolidoi kaikki liiketoimintavaatimusten priorisoinnissa käytetyt Excel-taulukot yhteen Exceliin, jonka avulla suoritettiin liiketoimintavaatimusten prioriteettien laskenta analyyttisen hierarkiaprosessin laskentasääntöjen mukaisesti. Kesäkuun puolessa välissä järjestettiin työpaja kaikille liiketoiminnan edustajille keskitetysti, jossa esiteltiin liiketoimintavaatimusten priorisoinnin tulokset. Työpajassa todettiin ja päätettiin, että liiketoimintavaatimusten prioriteetit olivat oikeat, mutta niitä voitiin myös muuttaa tarvittaessa. Työpajan jälkeen liiketoimintavaatimukset linkitettiin tuotekehitysprojekteihin kuvion 28 mukaisesti. Käytännössä tämä tarkoitti sitä, että kaikki tuotesuunnitelmalla olleet liiketoimintavaatimukset tulivat läpinäkyviksi tuotekehitykseen, koska kaikki liiketoimintavaatimukset linkitettiin tuotekehitysprojekteihin niiden prioriteetin mukaisessa järjestyksessä.



Kuvio 28. Liiketoimintavaatimusten linkitys tuotekehitysprojekteihin.

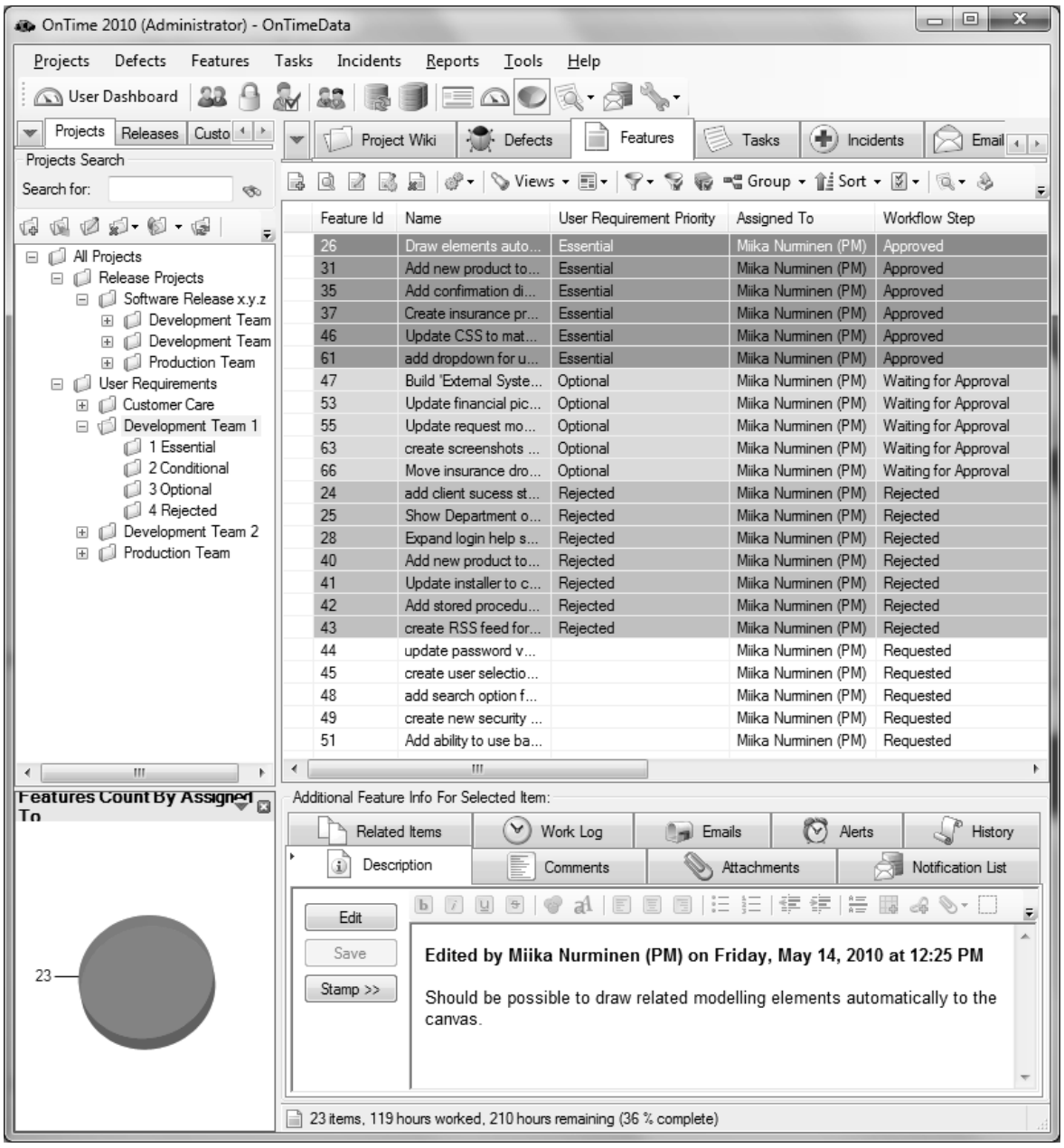
Kohdeyrityksen käyttämään vaatimustenhallintajärjestelmään luotiin kuviossa 29 esitetty suunnittelunäkymä, jonka avulla liiketoimintavaatimuksiin liitettyjen käyttäjävaatimusten edistymistä oli helppo seurata. Suunnittelunäkymien avulla liiketoimintavaatimuksien sisältämiä käyttäjävaatimuksia pystyttiin tarkastelemaan esim. julkaisu-, tiimi- tai syklikohtaisesti. Kohdeyrityksessä oli jo entuudestaan käytössä vaatimusten työmäärien arviointi, jonka avulla esim. tuotejulkaisujen resurssivaatimuksia pystyttiin arvioimaan tehokkaasti.



Kuvio 29. Liiketoimintavaatimusten suunnittelunäkymä.

6.2.4.3 Käyttäjävaatimusten priorisointi

Tuotehallinta vastasi käyttäjävaatimusten priorisoinnista tiimikohtaisesti siten, että tuotteiden omistajat suorittavat vastuullaan oleviin tuotteisiin liittyvien käyttäjävaatimusten priorisoinnin luokittelemalle ne asianomaisiin prioriteettiluokkiin. Käyttäjävaatimusten priorisointi toteutettiin kohdeyrityksen käytössä olleella vaatimustenhallintajärjestelmällä, joka on esitetty kuviossa 30.



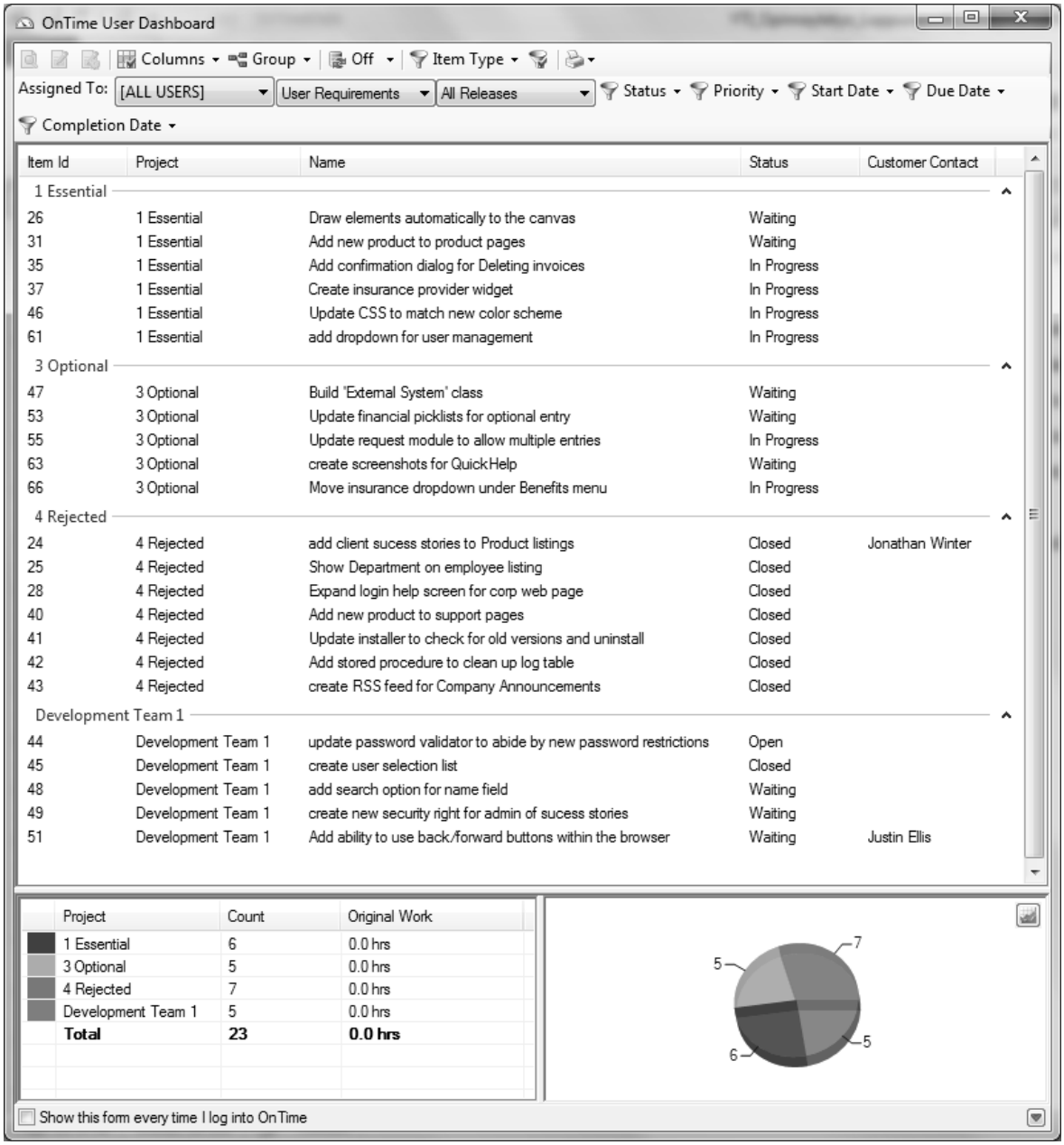
Kuvio 30. Käyttäjävaatimusten priorisointi.

Kohdeyrityksen keskitetty asiakastukitiimi keräsi käyttäjävaatimukset keskitetysti kaikilta sidosryhmiltä. Lisäksi tuotehallinta ja tuotekehitys pystyivät lisäämään uusia käyttäjävaatimuksia tarvittaessa. Kaikki uudet käyttäjävaatimukset ohjattiin tiimikohtaiseen rakenteeseen, joka oli tuotekohtaisesti tuotteenomistajan hallinnoima. Osa käyttäjävaatimuksista oli niin isoja kokonaisuuksia, että tuotteenomistajat pilkkoivat ne helpommin toteutettaviin osakokonaisuuksiin, jotka linkitettiin toisiinsa hyödyntämällä vaatimustenhallintajärjestelmän relaatioita. Osa käyttäjävaatimuksista oli niin pieniä, että ne voitiin toteuttaa sellaisenaan. Kohdeyrityksen vaatimustenhallintajärjestelmässä hyödynnettiin työnkulkutoiminnallisuutta, jonka avulla kontrolloitiin käyttäjävaatimusten käsittelyä.

Käyttäjävaatimukset luokiteltiin kolmeen eri prioriteettiryhmään tai ne hylättiin. Tuotteenomistaja priorisoi käyttäjävaatimukset välttämättömiksi, ehdollisiksi ja valinnaisiksi, jonka jälkeen tuotteenomistaja kiinnitti käyttäjävaatimukset tiettyyn tuotejulkaisuun. Tuotejulkaisuihin kiinnitettiin ainoastaan välttämättömiä ja ehdollisia käyttäjävaatimuksia. Käyttäjävaatimusten priorisointi perustui tuotesuunnitelmalla oleviin liiketoimintavaatimuksiin. Käyttäjävaatimus luokiteltiin lähtökohtaisesti valinnaiseksi tai se hylättiin, mikäli sitä ei pystytty kytkemään tuotesuunnitelman liiketoimintavaatimuksiin. Yleisin syy käyttäjävaatimuksen hylkäämiselle oli se, että ohjelmistovaatimus oli jo olemassa mutta sitä ei ollut vielä toteutettu. Käyttäjävaatimukset priorisoitiin välttämättömiksi, mikäli todettiin, ettei liiketoimintavaatimus ole hyväksyttävissä ilman kyseenomaista käyttäjävaatimusta. Muussa tapauksessa käyttäjävaatimus katsottiin ehdolliseksi. Uusia käyttäjävaatimuksia kirjattiin järjestelmään noin 20 kappaletta kuukaudessa, joten tuotteiden omistajat priorisoivat niitä sitä mukaa, kun ne olivat kirjattu järjestelmään.

Tehokas prioriteettiryhmien hyödyntäminen mahdollisti keskittymisen olennaiseen eli välttämättömiin käyttäjävaatimuksiin. Välttämättömät käyttäjävaatimukset olivat ns. ydintoiminnallisuuksia, jotka oli toteutettava, ennen kuin voitiin aloittaa seuraavan liiketoimintavaatimuksen toteuttaminen. Ehdollisia käyttäjävaatimuksia toteutettiin lähtökohtaisesti vasta sen jälkeen, kun kaikki välttämättömät käyttäjävaatimukset oli jo toteutettu.

Kohdeyrityksen käyttämä vaatimustenhallintajärjestelmä mahdollisti erilaiset käyttäjäkohtaiset näkymät käyttäjävaatimusten hallintaan. Kuviossa 32 on kuvattu yhden kehitystiimin vastuulla olevien käyttäjävaatimusten hallintanäkymä, jota tuotteen omistaja näki helposti käyttäjävaatimusten tilan tietyn tuotejulkaisun suhteen.



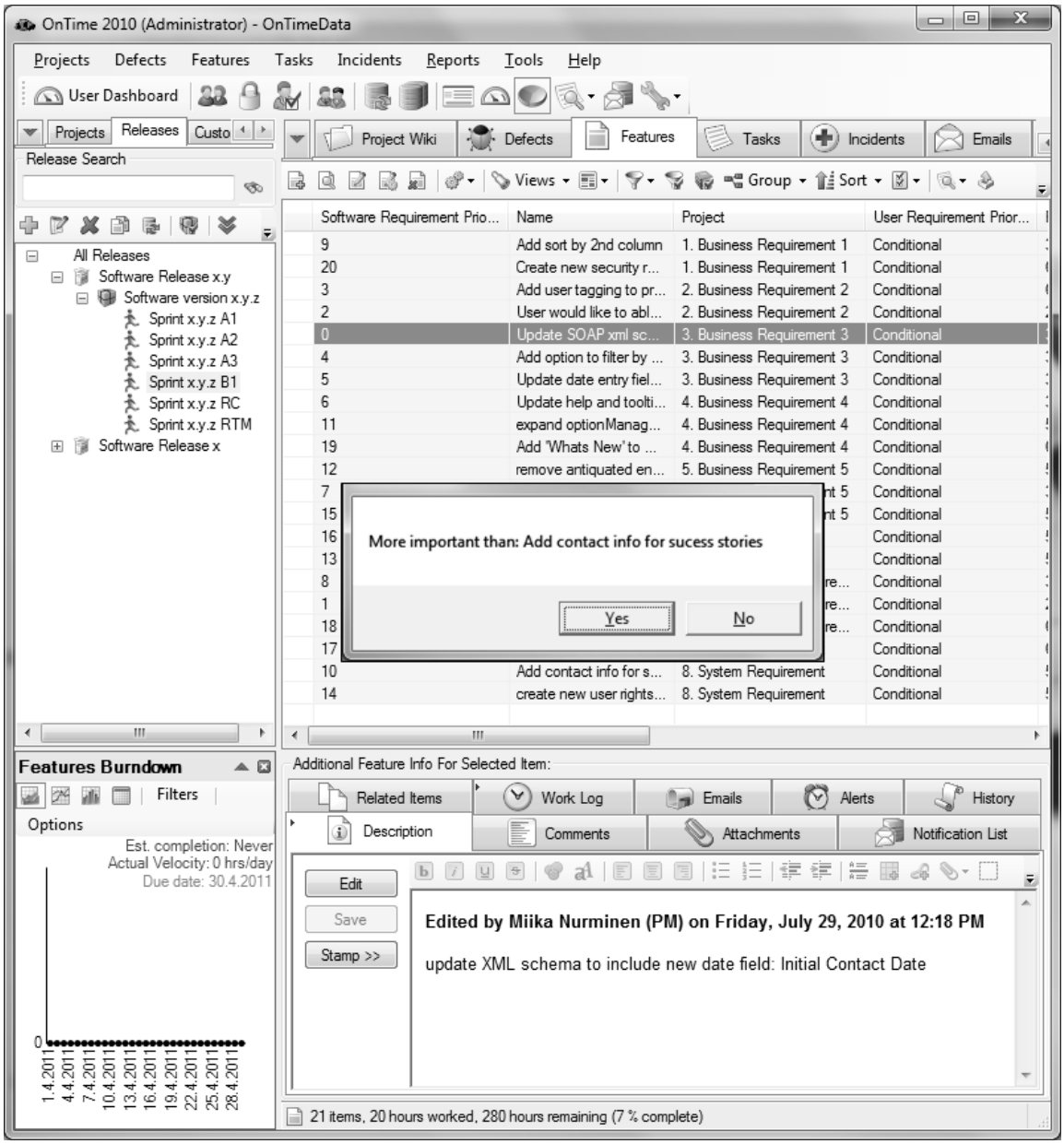
Kuvio 32. Käyttäjävaatimusten käsittely.

6.2.4.4 Ohjelmistovaatimusten priorisointi

Yksittäiset ohjelmistovaatimusspesifikaatiot toteutettiin projekteihin kytketyissä kehityssykleissä ketterän ohjelmistokehityksen mukaisesti. Kehityssyklien suhde projekteihin on esitetty kuviossa 30. Kehityssyklien kesto oli neljä viikkoa ja yhdessä kehityssyklissä oli mukana ohjelmistovaatimuksia eri projekteista. Osa kehityssykleistä saattoi sisältää ainoastaan ehdollisia vaatimuksia, jos kaikki välttämättömät vaatimukset oli jo toteutettu, kuten kuviosta 30 voi todeta. Kaikkia ehdollisia vaatimuksia ei kuitenkaan ehditetty välttämättä toteuttamaan kyseisessä syklissä, joten tuotteenomistaja priorisoi kuvion 30 esittämässä tapauksessa ehdolliset vaatimukset tasapainotetun puolitushaun (BST) avulla. Puolitushaku toteutettiin kohdeyrityksen käyttämään vaatimustenhallintajärjestelmään siten, että se kohdistettiin käyttöliittymässä valittuun

kehityssykliin, kuten kuvion 33 esimerkissä on nähtävissä. Puolitushakuproseduuri tunnisti valitusta kehityssyklistä ohjelmistovaatimukset, joiden prioriteetti oli nolla eli määrittelemättä. Tämän jälkeen se haki valitusta kehityssyklistä puolitushakupuun keskellä olevan ohjelmistovaatimuksen, johon verrattiin ohjelmistovaatimusta, jonka prioriteetti oli nolla. Sama toistettiin noudattamalla tasapainotetun puolitushaun periaatetta, kunnes vertailtavalla ohjelmistovaatimuksella oli prioriteettiluku.

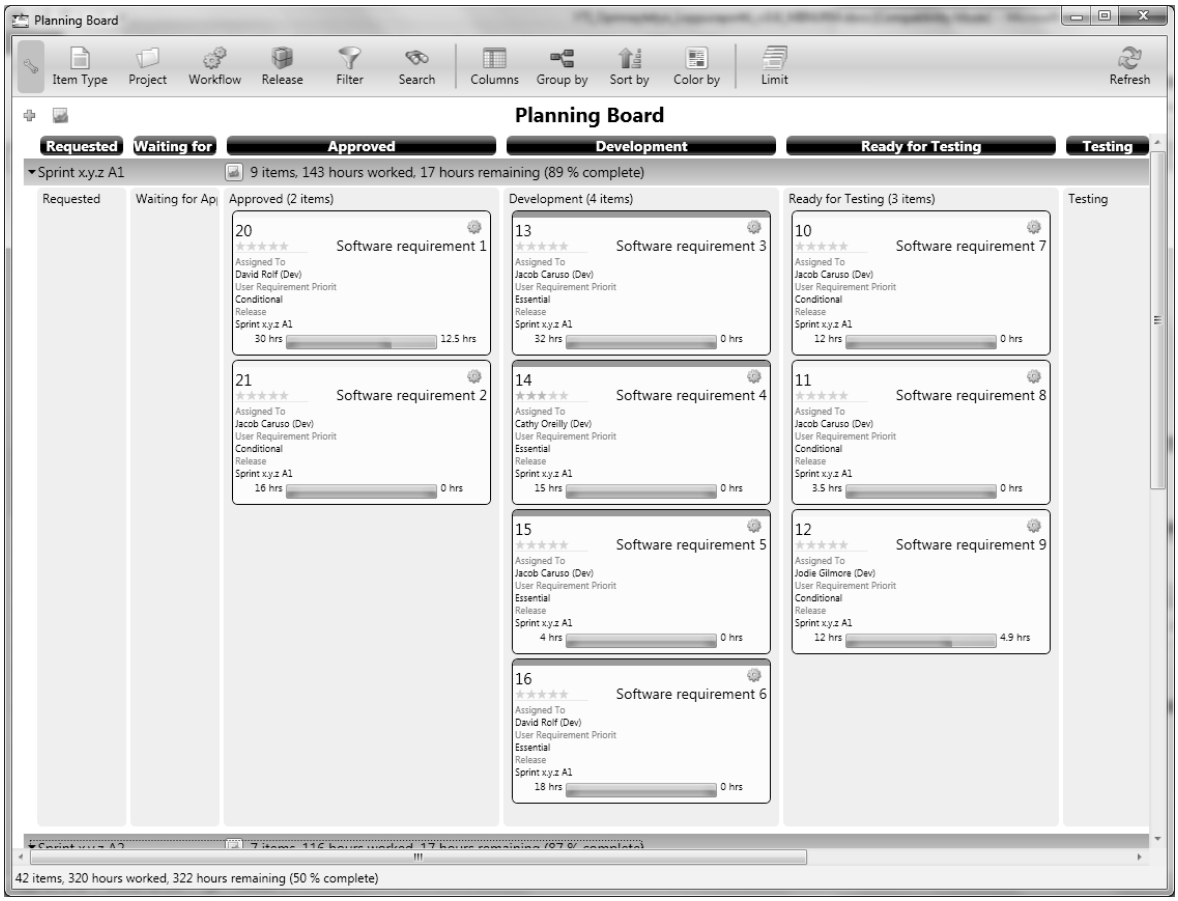
Edellä kuvattu ohjelmistovaatimusten priorisointi auttoi tuotekehittäjiä olennaisesti valitsemaan seuraavaksi toteutettavan vaatimuksen, koska kehityssyklin sisältämät ohjelmistovaatimukset olivat valmiiksi priorisoituna koko ajan. Lisäksi menetelmää voitiin hyödyntää liiketoiminnan kanssa, jos joku liiketoiminnan edustajista yritti saada pikaisesti jonkin tietyn ohjelmistovaatimuksen toteutukseen seuraavaan ohjelmistojulkaisuun.



Kuvio 33. Ohjelmistovaatimusten priorisointi.

Ohjelmistovaatimusten priorisoinnissa oli tärkeää hahmottaa ehdollisten ja valinnaisten ohjelmistovaatimusten suurpiirteinen järjestys, sillä niiden määrä saattoi olla erittäin suuri, joka johti siihen, että niiden käsittelyyn kului valtavasti tuotehallinnan resursseja.

Käyttäjävaatimuksiin linkitettyjä ohjelmistovaatimuksia oli mahdollista tarkastella erilaisten näkymien avulla kohdeyrityksen käyttämässä vaatimustenhallintajärjestelmässä. Kuviossa 34 on kuvattu käyttäjävaatimusten työnkulku tietyssä syklissä. Ohjelmistovaatimusdokumentti sisälsi yksityiskohtaiset toiminnalliset ja tekniset määritellyt, jotka ohjelmistokehittäjät toteuttivat ohjelmistotuotteisiin.



Kuvio 34. Ohjelmistovaatimusten työkulku julkaisusyklissä.

6.3 Toiminta

Tässä tutkimuksessa hyödynnettiin konstruktiivista tutkimuskehystä etsittäessä vastausta tutkimuksen pääongelmaan eli siihen voidaanko ohjelmistotuotteelle esitetyt vaatimukset priorisoida systemaattisemmin ja läpinäkyvämmin kohdeyrityksessä. Tutkimuksessa käytetyn konstruktiivisen tutkimuskehyksen tehokas hyödyntäminen pohjautui pääongelman tueksi määriteltyihin osaongelmiin, jotka jäsensivät tutkimuksen käytännön toteutuksen.

Kohdeyrityksen nykytila ohjelmistotuotteiden vaatimusten priorisoinnin suhteen selvitettiin tuotekehityksen prosessianalyysillä ja vaatimusten priorisointiin osallistuvien henkilöiden avoimilla haastatteluilla. Prosessianalyysin tuloksista voitiin todeta, että vaatimusten priorisoinnissa ei käytetty systemaattista menetelmää. Vaatimusten priorisointiin osallistuvien henkilöiden haastattelut osoittivat, että vaatimusten priorisointi ei ollut läpinäkyvää eri sidosryhmille.

Kirjallisuuskartoituksen avulla tutkimuksessa selvitettiin millaisia suosituksia ja ratkaisuja on olemassa ohjelmistotuotteiden vaatimusten priorisointikäytännöille. Kirjallisuuskartoituksen avulla selvitettiin vaatimusten priorisoinnin suhde ohjelmistotuotantoon ja vaatimusmäärittelyyn. Perehtyminen ohjelmistotuotantoon ja vaatimusmäärittelyyn yleisellä tasolla mahdollisti mm. yleisesti käytettyjen käsitteiden hyödyntämisen laadittaessa käsitemallia.

Aikaisempiin tutkimuksiin perehtyminen ja syventävät kirjallisuuskartoitukset kertoivat, millaisilla menetelmillä ohjelmistotuotteiden vaatimuksia voidaan priorisoida kohdeyrityksessä. Aikaisemmat tutkimukset keskittyivät pääasiallisesti todistamaan priorisointimenetelmien virheettömyyttä, tehokkuutta, jne. Tässä tutkimuksessa menetelmien hyödyntämien algoritmien tarkastelu ei ollut niin olennaista, koska vaatimusten systemaattisessa ja läpinäkyvässä priorisoinnissa oli pikemmin kyse yhteisen toimintakulttuurin jalkauttamisesta. Toisaalta aikaisemmat tutkimukset antoivat arvokasta tietoa mm. priorisointimenetelmien helppokäyttöisyydestä, opittavuudesta, tulosten ymmärrettävyydestä ja skaalattavuudesta, joilla oli suuri merkitys myös toimintakulttuurin näkökulmasta.

Käytännössä tutkimuksen aikana todettiin, että ohjelmistotuotteiden vaatimuksia priorisoitiin kohdeyrityksessä kolmella eri tasolla: 1) liiketoimintavaatimusten priorisointi strategisella tasolla, 2) käyttäjävaatimusten priorisointi taktisella tasolla ja 3) ohjelmistovaatimusten priorisointi operatiivisella tasolla. Tutkimuksessa toteutettu priorisointimalli pystyttiin sovittamaan ja mukauttamaan edellä esitetyille tasoille niin, että vaatimuksia pystyttiin käsittelemään systemaattisesti ja läpinäkyvästi eri tasoilla ja tasojen välillä. Lisäksi vaatimusten priorisointi kytkettiin tiiviimmin kohdeyrityksen strategiaprosessiin tuotesuunnitelman avulla.

6.3.1 Rakentaminen

Edellä kuvattujen rakentamisvaiheiden perusteella voitiin todeta, että ohjelmistovaatimusten priorisoinnissa käytettävät käsitteet pystyttiin tunnistamaan ja sovittamaan kohdeyrityksen toimintaan. Käsitteiden avulla muodostettiin käsitemalli vaiheistuksen pohjaksi.

Ohjelmistovaatimusten priorisoinnin vaiheistus toteutettiin hyödyntämällä käsitemallia siten, että vaatimukset olivat jäljitettävissä strategiselta tasolta operatiiviselle tasolla ja päinvastoin. Priorisointimallin vaiheet pystyttiin toteuttamaan käyttöympäristöön hyödyntämällä tarkoituksen mukaisia työkaluja ja tietojärjestelmiä, kuten kohdeyrityksen entuudestaan käyttämää vaatimustenhallintajärjestelmää. Näin ollen rakentamisvaiheessa onnistuttiin konseptoimaan käsitteet, määrittelemään käsitemalli, vaiheistamaan tehtävät ja toteuttamaan priorisointimalli käyttöympäristössään.

6.3.2 Arviointi

Tutkimuksessa määritellyt käsitteet ja käsitemalli antoivat kohdeyritykselle yhtenäisen termistön ohjelmistotuotteen vaatimusten priorisoinnille. Käsitteitä hyödynnettiin strategisen tason päätöksenteossa priorisoitaessa liiketoimintavaatimuksia tuotesuunnitelmalle, taktisen tason käyttäjävaatimuksien priorisoinnissa linkitettäessä välttämättömät käyttäjävaatimukset liiketoimintavaatimuksiin ja operatiivisella tasolla priorisoitaessa välttämättömät ohjelmistovaatimukset tehokkaasti toteutukseen. Yhteinen termistö auttoi keskustelemaan vaatimuksista eri tasoilla ja eri sidosryhmien välinen kommunikaatio parantui olennaisesti, koska pystyttiin puhumaan vaatimuksista niiden oikeassa merkityksessä.

Tässä tutkimuksessa kehitetty priorisointimalli vaiheistettiin prosessikuvauksen avulla, jotta se pystyttiin kommunikoimaan eri sidosryhmille tehokkaasti. Prosessikuvaus lisättiin osaksi kohdeyrityksen tuotekehityksen toimintaprosesseja. Prosessin eri vaiheita pystyttiin hyödyntämään tehokkaasti priorisointimallin eri tasoilla. Prosessi kytkettiin kohdeyrityksen strategiaprosessiin, jotta se pystyttiin vaiheistamaan strategiasyklin mukaisesti. Tämä mahdollisti vaatimusten priorisoinnin entistä tiiviimmän kytkemisen itse strategiatyöhön ja se toi mukaan läpinäkyvyyttä strategiasta operatiiviselle tasolle.

6.3.3 Teoretisointi

Tässä tutkimuksessa kehitetty priorisointimalli perustui olemassa olevien ohjelmistovaatimuksiin liittyvien käsitteiden ja niiden välisten relaatioiden mukauttamiseen kohdeyrityksen omaan toimintaan. Lisäksi eri vaatimusten priorisointiin liittyviä päätöksenteon tasoja pyrittiin käsittelemään strategisesta, taktisesta ja operatiivisesta näkökulmasta, jotta kehitettävä priorisointimalli säilyisi riittävän yksinkertaisena. Priorisointimalli ja sen sisältämät abstraktiotasot kytkettiin kohdeyrityksen käytössä olleeseen strategiaprosessiin ja operatiivisen tason tuotekehitysprosessiin. Lisäksi tuotehallinnan merkitys korostui entisestään taktisella tasolla, jossa päätettiin millaisilla käyttäjävaatimuksilla liiketoimintavaatimukset voidaan saavuttaa parhaiten.

Priorisointimallin sisältämät tehtävät olivat suoritettavissa, koska ne oli vaiheistettu johdonmukaisesti eri sidosryhmille. Priorisointimenetelmiin sisältyvät työvaiheet toteutettiin priorisointimenetelmien määrittelemien työvaiheiden mukaisesti. Priorisointimallia voitiin hyödyntää käytännössä, koska sen eri osat kytkettiin käytännön priorisointityöhön. Tuotehallinnalla oli suuri rooli priorisointimallin käytännön toteutuksessa, koska se on osallisena kaikissa eri priorisointivaiheissa jollain tavoin. Toisaalta oli toivottavaa, että juuri tuotehallinnalle syntyy mahdollisimman kattava kokonaiskuva vaatimuksien erillisistä tasoista ja niiden välisistä riippuvuuksista. Tuotehallinta alusti uuden tuotesuunnitelman karkean tason liiketoimintavaatimuksilla hahmoteltaessa pidemmän aikavälin tuotevisiota. Tuotehallinta sijoitti johtoryhmän esittämät strategiset liiketoimintavaatimukset tuotesuunnitelmalle. Tuotehallinta organisoi työpajat eri liiketoimintayksiköille ja keräsi toivotut liiketoimintavaatimukset yhteen. Tuotehallinta hahmotti ja linkitti alustavat käyttäjävaatimukset liiketoimintavaatimuksille. Tuotehallinta organisoi liiketoimintavaatimusten priorisointityöpajat ja työpajat joissa esiteltiin priorisoinnin tulokset. Tuotehallinta priorisoi välttämättömät käyttäjävaatimukset linkittämälle ne liiketoimintavaatimuksiin. Tuotehallinta priorisoi välttämättömät ohjelmistovaatimukset mahdollisimman tehokkaasti toteutukseen ja linkitti ne käyttäjävaatimuksiin. Tuotehallinta organisoi liiketoimintavaatimusten katselmoinnit, joissa keskusteltiin yksittäisistä liiketoimintavaatimusten taustalla olevista ohjelmistovaatimuksista. Tuotehallinnan suuri rooli ja kokonaisvastuu auttoivat lähtökohtaisesti priorisointimallin käyttöönotossa ja käytössä kohdeympäristössään.

6.3.4 Perustelu

Käytännössä priorisointimallista syntyi tärkeä työkalu tuotehallinnalle ja sen sisältämiä yksittäisiä priorisointitehtäviä suoritettiin kohdeyrityksessä päivittäin, jolloin voitiin todeta niiden myös toimivan käytännössä. Käytännössä priorisointimallin toimivuus voitiin todeta tutkimuksen aikana siten, että tuotekehityksessä oli toteutuksessa ainoastaan välttämättömiä ohjelmistovaatimuksia, jotka olivat linkitetty käyttäjävaatimusten kautta tuotesuunnitelmalla olleisiin liiketoimintavaatimuksiin. Priorisointimallia voitiin hyödyntää päivittäisessä vaatimusten priorisoinnissa, koska sen avulla vaatimukset olivat jäljitettävissä jokaisen eri tason läpi ylhäältä alas ja alhaalta ylös. Toisaalta siitä muodostui entistä tärkeämpi kommunikoinnin väline vaatimusten priorisointiin osallistuville sidosryhmille ja se olisi tuonut esiin esim. ohjelmistovaatimukset, jotka olivat toteutettu ilman linkkiä tuotesuunnitelmaan.

7 Diskussio

Ohjelmistotuotteelle esitettyjä vaatimuksia oli mahdollista hallita tehokkaasti priorisointimallin avulla. Mallin avulla voitiin arvioida, mitkä vaatimukset otetaan mukaan ohjelmistotuotteen seuraavaan julkaistavaan versioon ja mitkä vaatimukset voitiin jättää toteutettavaksi myöhemmin.

Tutkimuksessa kehitetyn vaatimusten priorisointimallin avulla voitiin varmistaa, että ohjelmistotuote vastasi liiketoiminnan esittämiin vaatimuksiin. Priorisointimallin sisältämän systemaattisen ja läpinäkyvän priorisointiprosessin avulla varmistettiin, että kaikki vaatimusten priorisointiin osallistuvat henkilöt pystyivät kertomaan oman näkökulmansa, vaatimukset saatiin priorisoitua ja pystyttiin hyväksymään toteutettavat vaatimukset. Priorisointimallin avulla ohjelmistotuotteille esitetyt vaatimukset pystyttiin priorisoimaan riittävän yksinkertaisesti ja nopeasti eri priorisointitasoilla.

7.1 Yhteenveto

Tässä tutkimuksessa vaatimusten priorisointia lähestyttiin vaatimusten priorisointiin osallistuvien erilaisten sidosryhmien näkökulmasta. Tutkimuksessa havaittiin, että eri sidosryhmillä oli erilainen näkökulma vaatimusten priorisointiin. Strategisella tasolla suoritettava priorisointi perustui yleensä koko markkinoiden esittämiin liiketoimintavaatimuksiin ja priorisoinnin perusteena oli yleensä tulevaisuuden ennustaminen. Taktisella tasolla suoritettava priorisointi pyrki huomioimaan myös olemassa olevien asiakkaiden ja loppukäyttäjien vaatimukset, jotka eivät välttämättä edustaneet koko markkinoiden näkemystä, sillä niiden taustalla oli yleensä jokin todellinen ohjelmistotuotteeseen liittyvä huoli, murhe, toive tai haave. Operatiivisella tasolla suoritettava priorisointi perustui yleensä seuraavan ohjelmistoversion mahdollisimman kustannustehokkaaseen julkaisemiseen. Operatiivisella tasolla pyrittiin identifioimaan välttämättömät ohjelmistovaatimukset, jotka synnyttivät mahdollisimman paljon lisäarvoa asiakkaille ja loppukäyttäjille.

Edellä esitetyt kolmen näkökulmaa olivat ratkaisu tälle tutkimukselle. Niiden avulla tutkimuksen pääongelmaa pystyttiin lähestymään hyödyntämällä kolmea erillistä abstraktiotasoa. Tutkimuksessa suoritetussa kirjallisuuskatsauksessa havaittiin, ettei käytössä ole yhtä tiettyä priorisointimenetelmää, jota voisi hyödyntää yleiskäyttöisenä priorisointimenetelmänä kaikilla edellä esitetyillä tasoilla. Näin ollen tutkimuksen kehittämistehtäväksi muodostui priorisointimallin rakentaminen edellä esitetyille tasoille. Eri tasoille löydettiin tarkoituksenmukainen priorisointimenetelmä, joka perustui käytettyjen resurssien suhteesta saavutettuihin hyötyihin. Eri tasoilla

käytetty priorisointimalli antoi kohdeyritykselle systemaattisuuden ja läpinäkyvyyden ohjelmistovaatimusten priorisointityöhön.

Tutkimuksessa hyödynnetty konstruktiivinen tutkimuskehys tuki priorisointimallin kehittämistä erittäin hyvin. Tutkimuskehys jäsensi priorisointimallin rakentamisen niin, että priorisointimalli pystyttiin kehittämään evolutionäärisesti. Tutkimuskehys mahdollisti vaatimustenhallintaan yleisesti liittyvien käsitteiden jäsentämisen ja mallintamisen. Lisäksi tutkimuskehyksen avulla pystyttiin liittämään tarkoituksenmukaiset priorisointimenetelmät priorisointimallin vaiheisiin. Priorisointimenetelmien sisältämien valmiiden vaiheiden avulla koko priorisointimalli pystyttiin toteuttamaan tehokkaasti käyttöympäristöön.

Tutkimus lisäsi tietämystä tutkittavalla osa-alueella erityisesti kohdeyrityksen toimintakulttuurin näkökulmasta. Priorisointimallin systemaattinen hyödyntäminen paransi koko vaatimustenhallintaan liittyviä perusedellytyksiä jäsentämällä yhteisen termistön kohdeyrityksen käyttöön. Lisäksi eri sidosryhmät ymmärsivät priorisointimallin myötä muiden sidosryhmien näkökulmia selvästi paremmin, koska malli antoi läpinäkyvyyttä vaatimusten priorisointiin. Toisaalta kaikki ymmärsivät entistä paremmin ohjelmistotuotteisiin toteutettavien ominaisuuksien merkityksen kokonaisuuden näkökulmasta, mikä piti sisällään kohdeyrityksen markkinat, asiakkaat ja loppukäyttäjät.

Tutkimuksen tulokset olivat yleistettävissä ja siirrettävissä, koska tutkimuksessa kehitetty priorisointimalli perustui prosessimalliin, jonka eri vaiheet hyödynsivät valmiita priorisointimenetelmiä. Lisäksi prosessimallin vaiheet oli jäsennetty ohjelmistovaatimustenhallinnassa yleisesti käytettävien käsitetasojen avulla. Tämä tutkimus tuki sitä olettamusta, ettei ole olemassa yhtä tiettyä priorisointimenetelmää, jonka avulla voidaan priorisoida kaikki ohjelmistotuotteeseen liittyvät vaatimukset. Toisaalta tutkimus tuki sitä olettamusta, että systemaattinen ja läpinäkyvä priorisointi liittyi myös yrityksen toimintakulttuuriin.

7.2 Johtopäätökset ja toimenpidesuositukset

Tässä tutkimuksessa nousi esille ohjelmistotuotteen julkaisusyklin pituuden suhde vaatimusten priorisointiin. Optimaalinen julkaisusyklin pituus mahdollistaisi ainakin teoriassa ainoastaan välttämättömien vaatimusten toteuttamisen seuraavaksi julkaistavaan ohjelmistoversioon. Olettamus perustui siihen, että tiettynä ajanhetkenä vanhemmat välttämättömät vaatimukset eivät tuntuneet enää niin välttämättömiltä, kuin ne olivat tuntuneet aikaisemmin. Näin ollen optimaalisen julkaisusyklin löytäminen on erittäin olennaista, jotta vaatimusten priorisoinnista voidaan saada kaikki mahdollinen hyöty.

Vaatimusten prioriteettien kommunikoinnissa markkinoille, asiakkaille ja loppukäyttäjille pitää olla erittäin huolellinen, koska kaikilla on omat intressit ja oma näkökulma puuttuvien ominaisuuksien suhteen. Markkinat vertailevat puuttuvia ominaisuuksia kilpailijoiden ohjelmistotuotteisiin ja asiakkaat sekä loppukäyttäjät vaativat ominaisuuksia reaktiivisesta näkökulmasta sen jälkeen, kun ovat törmänneet johonkin ohjelmistotuotteen rajoitteeseen. Prioriteettien julkistamisen sijaan on järkevämpää keskittyä toteutettujen ominaisuuksien tehokkaaseen kommunikointiin.

7.3 Jatkotutkimus- jatkokehittämisehdotukset

Priorisointimallin toimivuutta käyttöympäristössään olisi mahdollista ja suotavaa tutkia esim. toimintatutkimuksen näkökulmasta. Tässä tutkimuksessa rakennettiin ensimmäinen tuotantokäyttöön toteutettu priorisointimalli kohdeyritykselle painottamalla vaatimusten priorisointiin liittyviä vaiheita ja eri priorisointitasojen läpinäkyvyyttä. Toisaalta priorisointimalli toteutettiin prosessimallin avulla ja prosessimallin liittyy olennaisesti jatkuvan parantamisen käsite.

Tuotesuunnitelma osoittautui myös tässä tutkimuksessa erinomaiseksi kommunikaatiovälineeksi eri tasoilla. Tuotesuunnitelma oli käytössä kohdeyrityksessä jo entuudestaan, mutta siihen liittyvää teoriatietämystä, käytäntöjä ja malleja olisi mahdollista parantaa entuudestaan.

Tutkimuksen sisältämän kehittämishankkeen tuloksena syntyneen priorisointimallin toteuttamisessa käytettiin useita eri työkaluja. Tuotesuunnitelma laadittiin käyttämällä kohdeyrityksen omaa ohjelmistotuotetta, liiketoimintavaatimukset priorisoitiin MS Excelin avulla ja käyttäjä- ja ohjelmistovaatimukset priorisoitiin vaatimustenhallintajärjestelmän avulla. Kokonaisuus oli tuotehallinnan vastuulla mutta toteutusympäristöä olisi mahdollista kehittää helpommin hallittavaksi.

7.4 Tutkimusprosessin evaluaatio

Tutkimuksen alkuvaiheessa kehittämistehtävän erottaminen itse tutkimuksesta olisi saattanut hieman helpottaa mm. tutkimussuunnitelman laatimista. Tutkimussuunnitelman ja kehittämistehtävän projektisuunnitelman käsitteleminen yhdessä määrittelydokumentissa aiheutti haasteita erityisesti silloin, kun työskentelyssä oli pidempiä taukoja.

Tutkimuksen kannalta olennaisinta oli tutkimuksen tehokas rajaaminen. Tutkimuksen rajaaminen ohjelmistotuotteen vaatimusmäärittelyyn sisältyvään vaatimusten priorisointiin ja vaatimusten priorisoinnin apuna käytettäviin menetelmiin mahdollisti konkreettisen priorisointi-

mallin kehittämisen tehokkaasti. Idean, johtoajatuksen ja tutkimusongelman kiteyttämisen jälkeen itse kehittämishankkeen toteuttaminen oli helpoin vaihe tutkimuksessa.

Tutkimusraportin kirjoittaminen pitkittyi monen eri asian summana, joten voidaan todeta, että konstruktiivisen tutkimuskehyksen vahvuus on sen käytännönläheisessä tutkimusotteessa, joka takaa konkreettiset lopputuotokset toteutusympäristöön.

Lähteet

Ainasoja, A. 16.6.2010. Kansainvälisen liiketoimintayksikön johtaja. QPR Software Oyj. Haastattelu. Helsinki.

Ambler, S. 2005. The Agile System Development Life Cycle (SDLC). Luettavissa: http://www.ambysoft.com/essays/agileLifecycle.html. Luettu: 2.10.2010.

Aurum, A. & Wohlin, C. 2005. Engineering and Managing Software Requirements. Springer. Berlin.

Bebensee, T. & Weerd, I.& Brinkkemper, S. 2010. Binary Priority List for Prioritizing Software Requirements. Requirements Engineering: Foundation for Software Quality, REFSQ 2010, LNCS 6182, s. 67-78.

Bell, J. & Gupta, G. 1993. An Evaluation of Self-adjusting Binary Search Tree Techniques. Software - Practice and Experience, 23, 4, s. 369-382.

Berander, P. & Khan, K. & Lehtola, L. 2006. Towards a Research Framework on Requirements Prioritization. Sixth Conference on Software Engineering Research and Practice in Sweden, 2006, s. 39-48.

Carlshamre, P. & Regnell, B. 2000. Requirements Lifecycle Management and Release Planning in Market-Driven Requirements Engineering Processes. Database and Expert Systems Applications, 11, s. 961 – 965.

Charette, R. 1989. Software Engineering Risk Analysis and Management. McGraw-Hill Book Company. New York.

Cohn, M. 2006. Agile Estimating and Planning. Prentice Hall. Upper Saddle River, New Jersey.

Cohn, M. 2008. User Stories Applied. For Agile Software Development. 12. Painos. Addison-Wesley. Boston.

Bray, I.2002. An Introduction to Requirements Engineering. Addison-Wesley. Harlow.

Davis, A. 2005. Just Enough Requirements Management: Where Software Development Meets Marketing. Dorset House Publishing. New York.

Erkheikki, M. 18.6.2010. Suomen liiketoimintayksikön johtaja. QPR Software Oyj. Haastattelu. Helsinki.

Haikala, I. & Märijärvi, J. 2004. Ohjelmistotuotanto. 10. painos. Talentum. Helsinki.

Hamlet, D. & Maybee, J. 2001. The Engineering of Software. Technical Foundations for the Individual. Addison Wesley Longman. Boston.

Heinonen, L. 17.6.2010. Palveluliiketoiminnan päällikkö. QPR Software Oyj. Haastattelu. Helsinki.

Hilska, T. 8.6.2010. Tuotepäällikkö. QPR Software Oyj. Haastattelu. Helsinki.

Hirsjärvi, S., Remes, P. & Sajavaara, P. 2009. Tutki ja kirjoita. 15. painos. Tammi. Helsinki.

IEEE Std 830-1998. IEEE Recommended Practice for Software Requirements Specifications. The Institute of Electrical and Electronics Engineers, Inc. New York. 31.

Järvinen, A. & Järvinen, P. 2004. Tutkimustyön metodeista. Tiedekirjakauppa TAJU. Tampere.

Junkkonen, K. 10.6.2010. Teknologiajohtaja. QPR Software Oyj. Haastattelu. Helsinki.

Karlsson, J. & Ryan, K. 1997. A Cost-Value Approach for Prioritizing Requirements. IEEE Software, 14, 5, s. 67 – 74.

Karlsson, J. & Wohlin, C. & Regnell, B. 1998. An Evaluation of Methods for Prioritizing Software Requirements. Information and Software Technology, 39, 14-15, s. 939-947.

Kotonya, G. & Sommerville, I. 2003. Requirement Engineering. Processes and Techniques. Fourth edition. Wiley. Chichester.

Laamanen, K. 2004. Johda liiketoimintaa prosessien verkkona. Ideasta käytäntöön. 5. painos. Suomen Laatukeskus Oy. Helsinki.

Lehto, T. 15.6.2010. Liiketoiminnan kehitysjohtaja. QPR Software Oyj. Haastattelu. Helsinki.

Lehtola, L. 2006. Providing value by prioritizing requirements throughout software product development - State of practice and suitability of prioritization methods. Helsinki University of Technology. Helsinki.

Lehtola, L. & Kauppinen, M. 2004. Empirical Evaluation of Two Requirements Prioritization Methods in Product Development Projects. Software Process Improvement - Lecture Notes in Computer Science, 2004, 3281, s. 161-170.

March, S. & Smith, G. 1995. Design and natural science research on information technology. Decision Support Systems, 1995, 15, s. 251-266.

McConnell, S. 2002. Ohjelmistotuotannon hallinta. IT Press. Helsinki.

McConnell, S. 2004. Professional Software Development. Addison-Wesley. Boston.

Merriam-Webster. 2010 a. Priority. http://www.merriam-webster.com/dictionary/priority. Viitattu 27.6.2010.

Merriam-Webster. 2010 b. Prioritize. http://www.merriam-webster.com/dictionary/prioritize. Viitattu 27.6.2010.

Ojasalo, K. & Moilanen, T., & Ritalahti, J. 2009. Kehittämistyön menetelmät. WSOYpro. Helsinki.

Pressman, R. 2000. Software Engineering. A Practioner's Approach. Viides painos. McGraw-Hill Publishing Company. Berkshire.

Racheva, Z. & Daneva, M. & Buglione, L. 2008. Supporting the Dynamic Reprioritization of Requirements in Agile Development of Software Products. IWSPM '08 Proceedings of the 2008 Second International Workshop on Software Product Management, 10.1109, IWSPM.2008.7, s. 49-58.

Robertson, S. & Robertson, J. 1999. Mastering the Requirements Process. Addison-Wesley. Harlow.

Schwaber, K. 2004. Agile Project Management with Scrum. Microsoft Press. Redmond.

Siipola, T. 9.6.2010. Tuotteen omistaja. QPR Software Oyj. Haastattelu. Helsinki.

Sommerville, I. 1996. Software Engineering. Fifth edition. Addison-Wesley Publishing Company. Wokingham.

Sommerville, I. & Sawyer, P. 1997. Requirements Engineering. Wiley. Chichester.

Steinberg, D. & Palmer, D. 2004. Extreme Software Engineering. A Hands-On Approach. Pearson Education. Upper Saddle River.

Stevens, S. 1946. On the Theory of Scales of Measurement. Science, 103, 2684, s. 677-680.

Virtanen, T. 15.6.2010. Tuotekehitysjohtaja. QPR Software Oyj. Haastattelu. Helsinki.

Walraet, B. 1991. A Discipline of Software Engineering. Elsevier Science Publishers B.V. Amsterdam.

Wiegers, K. 1999. First Things First: Prioritizing Requirements. Luettavissa: http://www.processimpact.com/articles/prioritizing.html. Luettu: 15.2.2010.

Wiegers, K. 2003. Software Requirements. Toinen painos. Microsoft Press. Redmond.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B. & Wesslén, A. 2000. Experimentation in Software Engineering. An Introduction. Kluwer Academic Publishers. Dordrecht.

Liitteet

Liite 1. Priorisointimenetelmien kuvaus.

Menetelmä	Kuvaus	Tavoite	Luokittelu	Tuotokset
AHP (Analytic Hierarchy Process)	1. Vaatimusten suunnittelijat katselmoivat vaatimukset (kattavuus ja yksiselitteisyys) 2. Asiakkaat ja käyttäjät suorittavat pareittaisen vertailun (kunkin vaatimuksen suhteellisen arvon) 3. Ohjelmistosuunnittelijat suorittavat pareittaisen vertailun arvioidakseen kunkin vaatimuksen suhteelliset toteuttamiskustannukset 4. Ohjelmistosuunnittelijat laskevat kunkin vaatimuksen suhteellisen arvon ja toteuttamiskustannukset (hinta-arvo – diagrammi) 5. Sidosryhmät hyödyntävät hinta-arvo – diagrammia (käsitekartta - vaatimusten analysointi)	Muutamia vaatimuksia Tärkeimmät vaatimukset	Suhteellinen arvo	Suhteellinen arvo ominaisarvon perusteella
Wiegersin menetelmä	1. Samalla käsitetasolla olevien vaatimusten listaus 2. Vaatimusten arviointi suhteessa arvon tuottoon asiakkaalle (1 pieni arvo - 9 suuri arvo) 3. Haitan arviointi asiakkaalle ellei vaatimusta toteuteta (1 pieni haitta -9 suuri haitta) 4. Lasketaan arvot asiakkaalle vaatimuskohtaisesti (tekijöitä mahdollista painottaa) 5. Arvioidaan vaatimusten toteuttamisesta aiheutuvat suhteelliset kustannukset (1 matala - 9 korkea) 6. Arvioidaan vaatimusten toteuttamisesta aiheutuvat suhteelliset tekniset ja muut riskit (1 pieni riski - 9 korkea riski) 7. Saadut arvot syötetään lomakkeelle ja lasketaan saatujen arvojen perusteella prioriteetit kullekin vaatimukselle	Kymmeniä vaatimuksia Julkaisusuunnitelma	Vertailuluku näkökohtia yhdistämällä	Yksittäinen vertailuluku prioriteetille prioriteetti = arvo % / (kustannus % + riski %)
IEEE Std 830-1998	1. Välttämätön - Ohjelmisto ei ole hyväksyttävä, ellei näitä vaatimuksia ole toteutettu sovitulla tavalla 2. Ehdollinen - Laajentaisi ohjelmistotuotteen ominaisuuksia, mutta ei tekisi tuotteesta mahdotonta hyväksyä 3. Valinnainen - Joukko toimintoja, jotka saattaisivat olla hyödyllisiä	Satoja vaatimuksia Erotellaan tärkeimmät muista	Arvo ryhmittelemällä	Yksittäinen vertailuluku prioriteetille luokituksen avulla
100-Point Method (100P)	1. Jokainen osallistuja saa 100 pistettä jostain arvosta 2. Pisteillä on tarkoitus ostaa tarjolla olevia vaatimuksia 3. Jokainen osallistuja kirjoittaa esim. paperille minkä arvoiseksi arvostavat tietyn vaatimuksen 4. Pisteet summataan vaatimuksittain 5. Korkeimmat pisteet saaneet vaatimukset arvotetaan tärkeimmiksi vaatimuksiksi	Muutamia vaatimuksia Tärkeimmät vaatimukset	Arvo äänestämällä ja panostamalla	Yksittäinen vertailuluku prioriteetille äänestyksen ja panostuksen avulla
Binary Search Tree (BST)	1. Kaikki vaatimukset kerätään yhteen pinoon. 2. Valitaan yksi vaatimus juurinoodiksi. 3. Valitaan toinen vaatimus ja verrataan sitä juurinoodin vaatimukseen. 4. Vähemmän tärkeä rinnastetaan juurinoodin vasemmalle puolelle ja tärkeämpi oikealle puolelle. Uusi vaatimus asetetaan lapsinoodiksi vasemmalle tai oikealle puolelle riippuen prioriteetista. Vaiheet kolme ja neljä toistetaan, kunnes kaikki on vertailtu. 5. Puurakennetta voidaan hyödyntää niin, että tärkeimmät vaatimukset ovat listojen alussa ja vähemmän tärkeät listojen lopussa.	Paljon vaatimuksia	Näkökulma kuten arvo ryhmittelemällä	Yksittäinen vertailuluku prioriteetille BST-algoritmin avulla

Liite 2. Priorisointimenetelmien helppous.

Menetelmä	**Syötteet**	Aika	Helposti opittava	Helppokäyttöisyys	Tulosten ymmärrettävyys	Houkuttelevuus
AHP (Analytic Hierarchy Process)	Käyttäjien aktiviteetit	Vaatii paljon aikaa jos vertailtavia vaatimuksia on paljon	Suhteellisen monimutkainen laskenta. Kumpi vertailtavista vaatimuksista on tärkeämpi ja missä suhteessa: 1 Samanarvoiset 3 Hienokseltaan arvokkaampi 5 Huomattavasti arvokkaampi 7 Hyvin suuresti arvokkaampi 9 Äärimmäisesti arvokkaampi	Helpompaa kuin irrallisten vertailu-lukujen käyttäminen Murtolukujen käyttäminen voi olla hieman haastavaa (5 pystyrivi - 1/5 vaakarivi)	Tulokset helposti eroteltavissa – jokaiselle vaatimukselle suhteellinen arvo ominaisarvon perusteella	Käyttö sellaisenaan eri matriisien avulla saattaa aiheuttaa turhautumista
Wiegersin menetelmä	Käyttäjien aktiviteetit	5min. / vaatimus / asiakasedustaja 5min. / vaatimus / suunnittelija	Kyllä Jokainen vaatimus arvioidaan neljästä näkökulmasta: hyöty (1 pieni arvo - 9 suuri arvo) haitta (1 pieni haitta -9 suuri haitta) kustannus (1 matala - 9 korkea) riski (1 pieni riski - 9 korkea riski)	Vaatimusten suhteellisten arvojen määrittäminen saattaa olla hankalaa ilman ennalta määriteltyjä perusteita - mitä tietty arvo välillä 1-9 tarkoittaa	Tulokset helposti eroteltavissa - jokaisella vaatimuksella tärkeydestä ja toteuttamiskustannuksista riippuva vertailuluku	Helppoa hallita esim. MS Excelissä
IEEE Std 830-1998	Käyttäjien aktiviteetit	3min. / vaatimus	Kyllä Jokaiselle vaatimukselle määritellään prioriteettiluokka	Selkeä koska perustuu loogiseen ajatteluun	Yksinkertainen ja selkeä tulos Lista ei ole käyttökelpoinen jos esim. välttämättömiä vaatimuksia on erittäin suuri määrä	Käyttö sellaisenaan on jo varmasti kaikkien tiedossa eli lopputulos ei juuri houkuttele
100-Point Method (100P)	Käyttäjien aktiviteetit	3min. / vaatimus	Kyllä Jokainen vaatimus arvotetaan	Selkeä koska perustuu loogiseen ajatteluun	Yksinkertainen ja selkeä tulos	Helppo ja nopea tapa saada aikaiseksi valintoja
Binary Search Tree (BST)	Käyttäjien aktiviteetit	1min. / vaatimus	Kyllä Jokaiselle vaatimukselle haetaan karkea arvo	Selkeä koska perustuu loogiseen ajatteluun	Yksinkertainen ja selkeä tulos	Helppo ja nopea tapa saada aikaiseksi valintoja

Liite 3. Priorisointimenetelmien käytettävyys.

Menetelmä	Tarkkuus	Vikasietoisuus	Skaalautuvuus
AHP (Analytic Hierarchy Process)	Suhdeasteikko Vertailtavien vaatimusten tulisi kuulua samaan osakokonaisuuteen Erittäin tarkka ja luotettava	Erittäin vikasietoinen, voidaan tehdä tarkistuksia	Realistinen pienelle määrälle vaatimuksia n(n-1)2 10 vaatimusta = 45 vertailua 20 vaatimusta = 190 vertailua 100 vaatimusta = 4950 vertailua
Wiegersin menetelmä	Välimatka-asteikko Perustuu priorisoijien kykyihin arvioida hyötyjä, haittoja, kustannuksia ja riskejä Semikvantitatiivinen menetelmä joka ei ole matemaattisesti tiukka Wiegersin mukaan tulisi käyttää ainoastaan ”neuvoteltaviin” vaatimuksiin jotka eivät ole välttämättömiä	Vahvuudet eivät ole matemaattisessa pätevyydessä Välimatka-asteikko (1-9) - annetuilla arvoilla ei saisi operoida kuin ne olisi annettu suhdeasteikolla	Neljä vertailulukua jokaiselle vaatimukselle: - hyöty - haitta - kustannus - riski 10 vaatimusta = 40 vertailua 20 vaatimusta = 80 vertailua 100 vaatimusta = 400 vertailua
IEEE Std 830-1998	Järjestysasteikko Subjektiivinen, epätarkka, luokitukset karkeita, täysin henkilö riippuvainen Vaatimusten välillä on järjestys mutta ei suhteita Voidaan suorittaa karkea priorisointi ja jatkaa tietyn ryhmän osalta esim. AHP:llä	Vaatimukset valitaan subjektiivisesti ja yleensä ainoastaan välttämättömät toteutetaan Liiketoiminta priorisoi ilman kehittäjiä: 85% välttämättömiä, 10% ehdollisia, 5% valinnaisia	Yksi vertailuluku jokaiselle vaatimukselle: 10 vaatimusta = 10 vertailua 20 vaatimusta = 20 vertailua 100 vaatimusta = 100 vertailua
100-Point Method (100P)	Järjestysasteikko Subjektiivinen, epätarkka, luokitukset karkeita, täysin henkilöriippuvainen Sisältää jonkinlaista vertailua joten vaatimusten välille syntyy jonkinlaiset suhteet	Vaatimukset valitaan sen perusteella mitkä ovat osallistujalle tärkeimpiä Toimii vain ensimmäisellä kierroksella koska toisella kierroksella pisteet on nähtävillä ja tämä voi johtaa politikointiin	Teoriassa pitäisi vertailla kaikki keskenään, mutta käytännössä käy helposti niin että mietitään annettavia pisteitä yksittäiselle vaatimukselle
Binary Search Tree (BST)	Järjestysasteikko Subjektiivinen, epätarkka, luokitukset karkeita, henkilöriippuvainen Sisältää jonkinlaista vertailua joten vaatimusten välille syntyy jonkinlaiset suhteet	Vaatimukset valitaan subjektiivisesti ja vaatimukset saadaan oikealle ”hehtaarille” mikäli niitä on paljonn	n log n 10 vaatimusta = 33 vertailua 20 vaatimusta = 86 vertailua 100 vaatimusta = 664 vertailua