Markus Asikainen

ASIAKASVAATIMUSTEN PRIORISOINTI OSANA VAATIMUSTENHALLINTAA

JYVÄSKYLÄN YLIOPISTO
TIETOJENKÄSITTELYTIETEIDEN LAITOS
2013

TIIVISTELMÄ

Asikainen, Markus Petteri
Asiakasvaatimusten priorisointi osana vaatimustenhallintaa
Jyväskylä: Jyväskylän yliopisto, 2013, 24 s.
Tietojärjestelmätiede, kandidaatin tutkielma
Ohjaaja: Jauhiainen, Eliisa

Ohjelmistotuotannossa asiakasvaatimukset määrittävät ne ohjelmiston edellytykset ja rajoitukset, joiden avulla ohjelmisto pyrkii täyttämään heidän tarpeitaan reaalimaailmassa. Asiakasvaatimuksia tunnistetaan ja hallitaan vaatimustenhallintaprosessissa, jonka tavoitteena on varmistaa, että kehitettävä ohjelmisto täyttää asiakkaan ja käyttäjän vaatimukset ja odotukset. Ohjelmistoprojekteissa tunnistetaan usein huomattavasti enemmän vaatimuksia kuin ohjelmiston toteuttamisessa voidaan huomioida. Jotta asiakasvaatimuksista voidaan erottaa kaikkein tärkeimmät vaatimukset, vaatimukset on asetettava tärkeysjärjestykseen eli priorisoitava. Vaatimusten priorisoimiseksi on olemassa lukuisia tekniikoita, joiden monimutkaisuus ja kehittyneisyys vaihtelevat suuresti.

Tässä tutkielmassa käsitellään asiakasvaatimusten priorisointia osana vaatimustenhallintaa. Tutkielmassa kuvataan mitkä asiat vaikuttavat ohjelmistotuotannossa asiakasvaatimusten priorisointiin sekä millaisia priorisointitekniikoita on olemassa asiakasvaatimusten priorisoimiseksi. Tutkimusmenetelmänä käytetään kirjallisuuskatsausta. Tutkimuksen lähdeaineistona käytetään vaatimusten priorisointia koskevia tieteellisiä artikkeleita, kahta julkaistua pro gradutyötä, yhtä diplomityötä sekä muutamia vaatimusten priorisointia käsitteleviä kirjoja.

Tutkielman johtopäätöksinä todetaan, että on olennaista ymmärtää laajasti vaatimusten priorisointia määrittäviä asioita, jotta priorisointitekniikoita voidaan soveltaa parhaiten ohjelmistoprojektissa vallitsevaan viitekehykseen nähden. Priorisointitekniikoiden yksityiskohtaisen tuntemuksen (algoritmi, laskentakaava) sijaan on merkityksellisempää tiedostaa, mikä priorisointitekniikka sopii parhaiten kuhunkin priorisointitarpeeseen. Vaikka priorisointitekniikoiden soveltaminen itsessään voi olla hyvinkin suoraviivaista, lisää sidosryhmien tarpeiden, odotusten ja tärkeyksien merkitysten erilaisuuden huomioon ottaminen ja harmonisointi huomattavasti priorisoinnin monimutkaisuutta. Ohjelmistoprojekteissa on tärkeää sopia etukäteen, millä periaatteilla priorisointia tullaan tekemään ja on tärkeää sitouttaa kaikki sidosryhmät yhteisten periaatteiden noudattamiseen.

Asiasanat: Vaatimusten priorisointi, Vaatimusten hallinta, , asiakasvaatimukset, sidosryhmät, priorisointitekniikka, Analytical Hierarcy Process, Numerical Assignment

ABSTRACT

Asikainen, Markus Petteri
Customer requirements prioritization in requirement management
Jyväskylä: University of Jyväskylä, 2013, 24 p.
Information Systems Science, Bachelor's thesis
Supervisor: Jauhiainen, Eliisa

In software production, customer requirements are defining the conditions and limitations, which the software should help to meet in the real world. Customer requirements are identified and managed in the requirement management process and the aim for this is to ensure, that the developed software meets customer and user requirements and expectations. In software-projects there is usually identified significantly more requirements than can be implemented to the actual software. So that customer requirements can be distinguished among the most important requirements, we have to prioritize the requirements. For requirement prioritization there are several techniques, which complexity and sophistication vary greatly.

This thesis addresses the prioritization of customer requirements as part of the requirements management. The thesis describes what factors will affect the production of the software when prioritizing customer requirements, as well as what kind of prioritization techniques exist for prioritizing customer requirements. The research method used is the literature review. The scientifical sources used in this thesis are scientific articles, three master's thesis and few books concerning the requirements prioritization and requirement management.

Based on this study it is found, that it is essential to understand how the prioritization techniques can be best applied in a software project to the prevailing reference frame. Instead of having detailed knowledge of (an algorithm, a formula) prioritization techniques, it is more significant to be aware of what prioritization technique is best suitable to each priority need. Although the prioritization process can be very straightforward, more stakeholder needs, expectations and different kinds of importance increase the complexity of prioritization. Before starting the prioritization process, it is important to agree in advance the principles of prioritization and to commit all stakeholders together with these principles.

Keywords: prioritization, requirements management, customer requirements, stakeholders, prioritization technique, Analytical Hierarcy Process, Numerical assignment

KUVIOT

TAULUKOT

SISÄLLYS

1 JOHDANTO

Tietotekniikassa vaatimusmäärittelyllä (Requirement Engineering, RE) tarkoitetaan vaatimusten analysointiin ja dokumentointiin liittyvä toiminta, jossa ohjelmistolle asetettavat vaatimukset määritellään, hallitaan ja testataan systemaattisesti (Möttönen 2009, Thayer & Thayer 1997). Kotonyan ja Sommervillen (1998) mukaan vaatimustenhallinta on prosessi jäsenneltyjä toimintoja, joilla pyritään johtamaan, vahvistamaan ja ylläpitämään vaatimusdokumentaatiota sekä ymmärtämään ohjelmistolle asetettuja vaatimuksia.

Vaatimustenhallinnan (Requirement Management, RM) tavoitteena on varmistaa, että kehitettävä ohjelmisto täyttää asiakkaan ja käyttäjän vaatimukset (Möttönen 2009, Kauppinen ym., 2004). Lehtola (2003, Haikala & Märijärvi, 2000) toteaa ohjelmistolle asetettavien asiakas- ja käyttäjävaatimusten lausuvan julki ne ohjelmiston edellytykset ja rajoitukset käyttäjän ja asiakkaan näkökulmasta, joiden avulla ohjelmisto pyrkii täyttämään jonkin heidän tarpeensa reaalimaailmassa. Haikala ja Märijärvi korostavat kuitenkin, etteivät asiakas- ja käyttäjävaatimukset ota kantaa siihen, millainen ohjelmisto nämä vaatimukset täyttäisi.

Ohjelmistoprojekteissa tunnistetaan usein huomattavasti enemmän vaatimuksia kuin mitä ohjelmiston toteuttamisessa voidaan huomioida. Vaatimusten määrää rajoittavia tekijöitä ovat esimerkiksi ohjelmiston rakentamiseen käytettävissä oleva aika ja raha. (Berander & Andrews, 2005.)

Priorisointi tarkoittaa vapaasti suomentaen asioiden asettamista järjestykseen niiden tärkeyden perusteella (The Free Dictionary, 19.12.2012,). Vaatimusten priorisoinnilla tarkoitetaan ohjelmistolle asetettavien vaatimusten luokittelua olennaisiin vaatimuksiin: 1.vaatimukset, jotka on sisällytettävä ohjelmistoon, eli varsinaiset vaatimukset 2. Hyödylliset ominaisuudet, jotka vähentävät ohjelmiston tehokkuutta jos ne jätetään pois 3. Toivottavat ominaisuudet, jotka tekevät ohjelmistosta entistä toivottavamman tietyille sidosryhmille (Firesmith, 2004, Sommerville, 1997). Kallion (2008) mukaan priorisointi lukeutuu osaksi vaatimustenhallintaa:

> Vaatimukset voivat aluksi olla toteamuksia ohjelmiston tavoitteista ja käyttäjien toiveita luonnollisella kielellä ilmaistuina, mutta toiveista ja toteamuksista on muokattava virallisempia määrityksiä (NATURE Team 1996). Vaatimuksia täytyy valita, priorisoida, rajata ja niistä täytyy neuvotella (van Lamsweerde 2000). Ohjelmistokehitysprojekteissa on yleensä enemmän vaatimusehdotuksia kuin mitä aikaresurssit sallivat ja mistä asiakkaat ovat halukkaita maksamaan (esim. Karlsson 1996). Siksi vaatimusten priorisointi on keskeinen osa vaatimustenhallintaa. (Kallio, 2008, s. 12-13.)

Priorisoinnilla pyritään tunnistamaan vaatimusjoukosta kaikkein tärkeimmät vaatimukset ja erottamaan nämä kriittiset vaatimukset toissijaisista vaatimuksista. Vaatimusten priorisoinnissa joudutaan huomioimaan esimerkiksi vaatimusten tärkeyttä, riskejä ja toteutuskustannuksia. Priorisointia toteuttavat tyypillisesti ohjelmiston kehittämiseen liittyvät sidosryhmät, kuten asiakkaat ja käyttäjät, kehittäjät tai edellä mainittujen edustajat.

Tässä tutkielmassa tullaan tarkastelemaan tietojärjestelmille asetettavien asiakasvaatimusten priorisointia osana vaatimustenhallintaa. Tutkielmassa asiakasvaatimuksen määritelmää lähestytään asiakkaan ohjelmistotuotannon sidosryhmäroolin kautta. Keskeisiä käsitteitä ovat vaatimusten hallinta ja priorisointi. Tutkielmassa pyritään löytämään vastaukset seuraavaan tutkimuskysymykseen: Miten asiakasvaatimuksia priorisoidaan osana vaatimustenhallintaa? Tutkimuskysymys jakautuu edelleen alikysymyksiksi: 1. Mitkä asiat vaikuttavat ohjelmistotuotannossa asiakasvaatimusten priorisointiin? 2. Millaisia priorisointitekniikoita on olemassa asiakasvaatimusten priorisoimiseksi? Tutkimusmenetelmänä käytetään kirjallisuuskatsausta. Tutkimuksen lähdeaineistona käytetään vaatimusten priorisointia koskevia tieteellisiä artikkeleita, kahta julkaistua pro-gradutyötä, yhtä diplomityötä sekä muutamaa vaatimustenhallintaa ja vaatimusten priorisointia käsittelevää kirjaa.

Tutkielma jakautuu kahteen osaan. Ensimmäisessä osassa (Luku 2) käsitellään vaatimusten tunnistamista, sidosryhmien merkitystä priorisoinnissa, vaatimusten jaottelua sekä priorisointiin liittyviä haasteita. Toisessa osassa (Luku 3) käsitellään priorisointitekniikoiden soveltamista ja kuvataan tarkemmin kahta eri lähestymistavan priorisointitekniikkaa. Analytical Hierarchy Process edustaa vaatimusten parivertailua ja on toteutukseltaan erittäin monimutkainen tekniikka. Numerical Assignment puolestaan edustaa vaatimusten ryhmittelyä ja on toteutukseltaan erittäin yksinkertainen tekniikka. Viimeisessä luvussa esitellään yhteenveto, jossa kuvataan tutkimustulokset, tutkielman tärkeimmät johtopäätökset sekä suositukset mahdollisista jatkotutkimusaiheista.

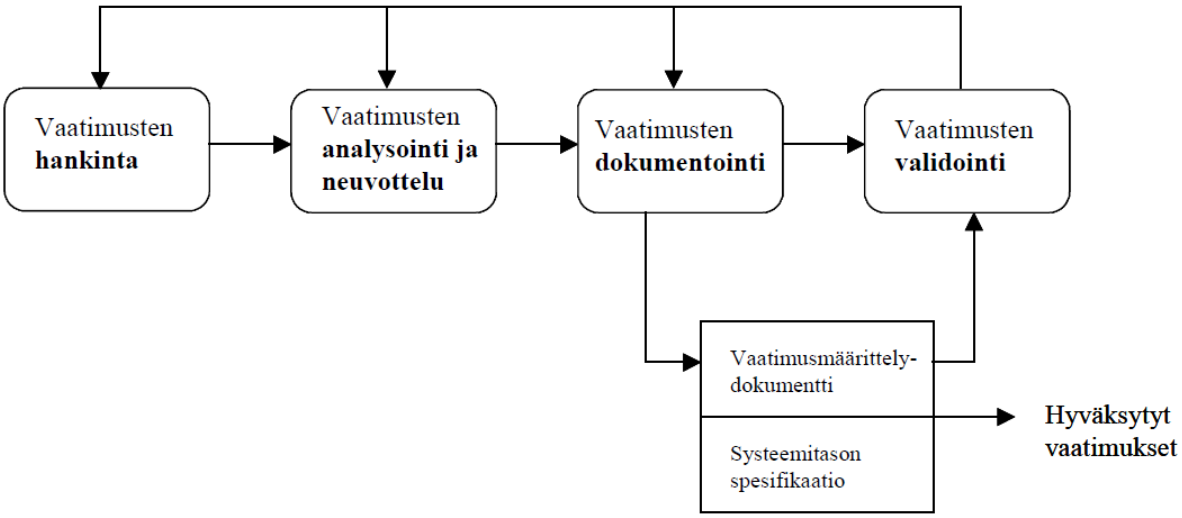
2 VAATIMUSMÄÄRITTELY JA VAATIMUSTEN PRIORISOINTI

Tässä luvussa kuvataan vaatimusten priorisointiin vaikuttavia asioita kuten syitä, jotka puoltavat priorisoinnin toteuttamista. Lisäksi luvussa käsitellään yleisimpiä priorisointiin liittyviä haasteita ja pyritään vastaamaan erityisesti kysymykseen, miksi vaatimuksia on tarpeen priorisoida ohjelmistoprojekteissa.

2.1 Vaatimusten tunnistaminen

Onnistunut ohjelmistokehitys edellyttää tehokasta vaatimusmäärittelyä ja sisäisten ja ulkoisten sidosryhmien tarpeiden huomioon ottamista. Tämä aiheuttaa suuren haasteen ICT-alan yrityksille onnistuneen muutostenhallinnan toteuttamiseksi ohjelmistokehitysprosessin aikana. (Möttönen, 2009.) Lehtolan (2003, Kotonya & Sommerville, 1998) mukaan vaatimusten määrittely- ja hallintaprosessilla tarkoitetaan jäsentynyttä toimintajoukkoa, jonka avulla luodaan, hyväksytään ja ylläpidetään vaatimusmäärittelydokumentaatiota (Kuvio 1).

Kotonyan ja Sommervillen (Lehtola, 2003, Kotonya & Sommerville, 1998) karkeapiirteinen prosessimalli jakautuu neljään osaan: vaatimusten hankintaan, vaatimusten analysointiin ja neuvotteluun, vaatimusten dokumentointiin sekä vaatimusten validointiin. Näitä toimintoja ei kuitenkaan voida ajatella täysin erillisinä, vaan toiminnot ovat osittain päällekkäisiä ja niiden välillä tapahtuu paljon iterointia. Eri sidosryhmien tulisi analyysivaiheessa neuvotella, mitkä vaatimuksista otetaan mukaan toteutussuunnitelmaan ja mitkä jätetään projektin ulkopuolelle.



Kuvio 1: Vaatimusten määrittely- ja hallintaprosessi. (Lehtola, 2003, Kotonya & Sommerville, 1998)

Kuitenkin vain harvoilla organisaatioilla on käytössään tarkasti määritelty ja standardisoitu vaatimusten määrittely- ja hallintaprosessi. Myös priorisoinnin toteuttamisessa esiintyy suuria eroavaisuuksia organisaatioiden välillä. (Lehtola, 2003, Kotonya & Sommerville, 1998.)

Ohjelmistotuotannossa käytetään nykyään kehitysmenetelminä perinteisten lähestymistapojen (esimerkiksi vesiputousmallin) sijasta yhä enemmän ketteriä menetelmiä, jonka johdosta vaatimusmäärittelyssä tunnistetut vaatimukset muuttuvat koko projektin ajan ja niihin palataan toistuvasti. Esimerkiksi Scrum-menetelmässä kehitystiimi työskentelee tiiviisti yhdessä asiakkaan kanssa vaatimusten tunnistamiseksi ja priorisoimiseksi. Tunnistetut vaatimukset kerätään priorisoituun listaan (tuotteen työlista). Laadittuun listaan voidaan lisätä myös uusia vaatimuksia tai muuttaa olemassa olevien vaatimusten järjestystä. Ketterissä menetelmissä kehitystiimi ja asiakas keskustelevat ohjelmistosta ja arvioivat yhdessä sen ominaisuuksia ja toimintoja. Keskustelun aikana asiakas saa antaa palautetta ohjelmistosta ja esittää uusia vaatimuksia. (Ruuska, 2012, Overhage ym., 2011.)

Liiketoimintaympäristöjen muutokset vaikuttavat ohjelmistoille asetettaviin vaatimuksiin. Kehitysprojekteissa aikataulut ovat pitkiä ja suuret vähitellen kehittyvät ohjelmistot vaativat kuukausien tai vuosien kehitystyön. Pitkistä aikatauluista ja yrityksen tarpeiden muutoksista johtuen vaatimukset edellyttävät uudelleen läpikäyntiä ja tarvittaessa muutoksia vaatimusten prioriteetteihin. (Firesmith, 2004.)

Monissa tapauksissa eri sidosryhmät asettavat saman vaatimuksen eri prioriteetille, mikä muodostuu ohjelmiston kehittäjälle haasteeksi. Sommervillen ja Sawyerin mukaan sidosryhmien erilaiset näkemykset tärkeysjärjestyksestä saattavat heijastella ryhmien erilaisia tarpeita tai vaihtoehtoisesti vain erilaisia havaintoja samasta tilanteesta (Lehtola, 2003).

Vaatimustenhallintaa koskevassa tutkimuksessa on alettu suuntautumaan myös markkinavetoisen vaatimustenhallinnan tutkimiseen, jossa ohjelmistolle asetetut vaatimukset kohdistuvat massamarkkinoille suunnattuihin paketoituihin ohjelmistotuotteisiin (”off-the-shelf” -tuotteet). Markkinavetoinen vaatimustenhallinta poikkeaa mittatilausohjelmistojen (”bespoke” -tuotteet) vaatimustenhallinnasta. Mittatilausohjelmistojen kehittämisen yhteydessä kehittäjät ovat aktiivisessa vuorovaikutuksessa tiettyjen ennalta tiedossa olevien sidosryhmien kanssa. Markkinavetoisessa vaatimustenhallinnassa kaikki sidosryhmät eivät ole ennalta tiedossa vaatimustenhallinnan jakaantuessa eri puolille maailmaa. Markkinavetoisen vaatimustenhallinnan tavoitteena on tunnistaa vaatimuksia ja niiden prioriteetteja koskevat yhtäläisyydet ja erot eri markkinasegmenttien välillä. (Regnell, Höst, Natt, Dag, Beremark & Hjelm, 2000.)

Koska pakettiohjelmistojen kaikki mahdolliset käyttäjät ja käyttäjäryhmät eivät ole ennalta tiedossa, on ongelmaa pyritty ratkaisemaan käyttämällä ”persoonat” –menetelmää. Tämän menetelmän juuret ovat alun perin markkinoinnissa. Tässä menetelmässä mallinnetaan kuvitteellisten persoonien markkinasegmenttejä antamalla näille persoonille erilaisia määreitä kuten nimi, ammatti, omaisuus, ikä, sukupuoli, sosiaalistaloudellinen asema. Näiden kuvitteellisten persoonien käytön tulee kuitenkin perustua reaalimaailman ihmisistä kerättyihin tietoihin. Kuvitteellisten persoonien avulla pyritään rajaamaan vaatimuksia ohjelmistolle, eli mitä ohjelmiston tulee tehdä ja mitä sen ei tule tehdä. (Berander & Andrews, 2005.)

2.2 Sidosryhmät ja vaatimusten jaottelu

Ohjelmistotuotannossa päätöksentekoa tukevilla menetelmillä on merkittävä rooli oikeiden päätösten tekemiseksi ja oikeiden tuotteiden kehittämiseksi. Vaatimusten priorisointi on tunnistettu yhdeksi tärkeimmistä aktiviteeteistä näiden oikeiden päätösten aikaansaamiseksi. (Berander, Khan & Lehtola, 2006.)

Wiegersin (2009) mukaan ohjelmistokehityksessä sidosryhmät voidaan jakaa projektin kannalta sisäisiin ja ulkoisiin sidosryhmiin. Sisäisiä sidosryhmiä ovat esimerkiksi projektipäällikkö, kehittäjät, testaajat sekä vaatimusten analysoijat. Ulkoisia toimijoita ovat mm. asiakkaat, jotka valitsevat, määrittelevät tai rahoittavat ohjelmiston kehitystä. Lisäksi asiakkaiksi lukeutuvat ohjelmiston käyttäjät, sertifioijat, auditoijat sekä markkinointia, valmistusta tai myyntiä koskevat tukiryhmät.

Vaatimusten priorisointi on Beranderin ja Andrewsin (2005) mukaan prosessi, jossa:

- Päätetään sidosryhmien toimesta järjestelmän ydinvaatimukset
- Suunnitellaan ja valitaan toivottavin joukko vaatimuksia ohjelmistojen onnistuneen käyttöönoton varmistamiseksi
- Rajataan projektin laajuutta, mikäli kohdataan ristiriitaisia odotuksia tai rajoitteita esimerkiksi aikataulun, budjetin, resurssien, toteutusajan tai laadun suhteen

- Tasapainotetaan liiketoiminnan hyötyjä suhteessa jonkin vaatimuksen toteuttamisesta aiheutuviin kustannuksiin
- Tasapainotetaan vaatimusten vaikutuksia ohjelmistoarkkitehtuuriin ja tuotteen jatkokehittämisestä aiheutuviin kustannuksiin
- Valitaan osajoukko vaatimuksista ja tuotetaan kuitenkin järjestelmä, joka tyydyttää asiakkaita.
- Arvioidaan tulevan järjestelmän asiakastyytyväisyyttä
- Tavoitellaan teknistä etumatkaa ja optimoidaan mahdollisuuksia menestyä markkinoilla
- Minimoidaan uudelleen tehtävän työn tarvetta ja aikataulun lipsumista (suunnittelun vakauttaminen)
- Käsitellään ristiriitaisia vaatimuksia, keskitytään neuvotteluprosessiin ja ratkaistaan
- erimielisyyksiä eri sidosryhmien kesken
- Maksimoidaan aiheutuvien kustannusten osalta kunkin vaatimuksen arvo

Beranderin ja Andrewsin (2005) mukaan sidosryhmien tulisi priorisoinnin lopputuloksena määrittää vaatimusten tärkeys, eli mitkä vaatimukset ovat sidosryhmän kannalta tärkeimpiä ohjelmiston toteutukselle. Tärkeys voi kuitenkin olla erittäin monitahoinen käsite, sillä kunkin sidosryhmän kokema tärkeys määrittyy sen mukaan, minkälaisia intressejä ja painotuksia vasten kukin sidosryhmä vaatimuksia tarkastelee. Jollekin sidosryhmälle tärkeys voi merkitä esimerkiksi ohjelmiston kiireellistä toteuttamista, mutta jollekin toiselle sidosryhmälle tärkeintä voi olla tuotearkkitehtuurille asetettavat vaatimukset tai strateginen merkittävyys yritykselle.

Pelkän vaatimusten tärkeyden määrittämisen lisäksi on vaatimusten priorisointia koskevassa päätöksenteossa syytä ottaa huomioon myös muita tekijöitä, jotka vaikuttavat ohjelmistoa koskevaan tyytyväisyyteen. Priorisoinnissa voidaan tärkeyden lisäksi arvioida mahdollisia seuraamuksia. Mitä seuraa mikäli jokin vaatimus ei täyty? Seuraamus ei kuitenkaan ole suora vastakohta vaatimuksen tärkeydelle, koska esimerkiksi jättämällä toteuttamatta jokin ohjelmistostandardi voidaan aiheuttaa merkittäviä seuraamuksia, vaikka itse vaatimuksen poisjäämisellä ohjelmistotototeutuksesta ei olisikaan merkitystä asiakkaalle. Sama periaate koskee ehdottomia vaatimuksia, joita käyttäjät pitävät itsestäänselvyytenä, mutta joiden puuttuminen voisi tehdä tuotteesta soveltumattoman markkinoille. (Berander & Andrews, 2005.)

Mikäli vaatimusten priorisoinnissa painotetaan vain yhtä valittua osa-aluetta, on helppoa päättää, mikä osa-alue on lopputuloksen kannalta toivottavin. Esimerkiksi voimme valita auton pelkästään sen huippunopeuden perusteella. Kun päätöksenteossa huomioidaan yhden osa-alueen sijasta useita osa-alueita kuten kustannukset, voivat asiakkaat voivat muuttaa mieltään vaatimusten prioriteeteista. Kustannusten huomioimisen myötä korkean prioriteetin vaatimukset voivat muuttua vähemmän tärkeiksi, mikäli ne todetaan erittäin

kalliiksi toteuttaa. Ohjelmiston toteutuskustannuksiin vaikuttavat mm. vaatimusten monimutkaisuus, aiemmin tehdyn ohjelmistokoodin uudelleen hyödyntämismahdollisuudet, testaus- sekä dokumentointitarpeet. Myös kustannuksia voidaan tarvittaessa priorisoida, mutta useimmiten priorisoinnin näkökulmasta riittää yksinkertaisesti todellisten kustannusten arviointi. (Berander & Andrews, 2005.)

Wiegers (2009) toteaa, että sidosryhmien vaatimuksia koskevat intressit menevät ristiin ohjelmistoprojekteissa ja siksi on tärkeää, että kaikille sidosryhmille muodostuu alusta alkaen selkeä käsitys tärkeyden merkityksestä kullekin sidosryhmälle. Tämän vuoksi priorisoinnissa käytettävät tärkeysperiaatteet, eli ”pelisäännöt”, tulee sopia yhteisesti ennen varsinaisen priorisoinnin aloittamista. Yleisesti suositellaan, että priorisointia koskevan päätöksenteon tueksi sidosryhmät koostavat yhteistyössä listauksia priorisoinnissa huomioon otettavista asioista, kukin sidosryhmä omiin tarpeisiinsa ja odotuksiinsa pohjautuen. Sidosryhmien yhteistyössä tunnistamat huomioitavat asiat tulee aina sovittaa vallitsevaan vaatimusviitekehykseen, jotta priorisoinnista saadaan irti suurin mahdollinen hyöty. (Berander & Andrews, 2005.)

Vaatimusten prioriteetteja koskevassa päätöksenteossa on priorisointiin vaikuttavien asioiden yhdistäminen tarpeen, jotta voidaan tehdä päätös toteutetaanko jokin vaatimus välittömästi, myöhemmin tai ei lainkaan. Esimerkiksi kustannus-arvo lähestymistavassa (cost-value approach) sekä arvoa (tärkeyttä sidosryhmälle) että toteuttamisen edellyttämiä kustannuksia priorisoidaan ja valitaan toteuttavaksi ne vaatimukset, jotka antavat eniten vastinetta rahalle. Suositussa Planning Game (PG) priorisointitekniikassa lähestymistapana on työpanoksen (työn kustannusten) ja riskien priorisointi sekä tasapainottelu näiden kahden tekijän välillä. Onnistuneen priorisoinnin varmistamiseksi priorisoinnin toteuttajalta edellytetään tietämystä ja kykyä tiedostaa mahdolliset priorisoinnissa huomioitavat asiat sekä tavat näiden asioiden yhdistämiseksi käytännön tasolla. (Berander & Andrews, 2005.)

Ohjelmistotuotannossa sidosryhmien välinen vuorovaikutus ja erilaiset intressit vaikuttavat merkittävästi vaatimusten tulkintaan. Vaatimusten tulkinnassa ongelmaksi muodostuvat useimmiten ristiriitaiset näkemykset vaatimusten sisällöstä ja prioriteetista. Tyypillisesti vaatimuksia tulkitaan jo pelkästään yksittäisen organisaation sisällä eri tavalla riippuen siitä, kuka henkilö priorisointia kulloinkin käsittelee. Epäonnistuneiden ohjelmistoprojektien yhteydessä on todettu puutteita kehittäjien ja asiakkaiden kanssa tapahtuvassa vuorovaikutuksessa ja tästä havainnosta on johdettavissa aktiivisen vuorovaikutuksen merkitys vaatimusten priorisoinnissa. Ohjelmiston kehittäjien ja asiakkaiden välisellä aktiivisella vuorovaikutuksella varmistetaan vaatimusten tulkinnan yhdenmukaisuus kaikkien ohjelmiston kehittämiseen osallistuvien sidosryhmien kesken. (Möttönen, 2009.)

2.3 Priorisoinnin haasteita

Vaatimusten priorisoinnissa kohdataan haasteita, jotka liittyvät ohjelmistolle asetettaviin vaatimuksiin, ohjelmistoprojektin aika-, raha- ja henkilöstöresursseihin. Lisäksi ohjelmistotuotannossa esiintyvät sanasto-ongelmat ja päätöksenteon vaikeus tuovat haasteita vaatimusten priorisointiin.

Vaatimuksiin liittyvät haasteet

Vaatimusten suuri määrä aiheuttaa Firesmithin (2004) mukaan haasteita priorisoinnille, koska laajoissa ohjelmistoissa voi olla jopa tuhansia yksittäisiä vaatimuksia ja näiden vaatimusten yhdenmukainen käsittely on hankalaa. Tästä johtuen prioriteetit ryhmitellään useimmiten pienempiin hallittaviin kokonaisuuksiin.

Lähes poikkeuksetta ohjelmistolle asetettavat vaatimuksia on kuvattu vaatimusmäärittelyn aikana eri abstraktiotasoilla ja tämä on omiaan aiheuttamaan haasteita priorisoimiselle. Tämä johtuu siitä, että korkeammalla abstraktiotasolla olevilla vaatimuksilla on taipumus saada korkea prioriteetti; esimerkiksi pohdittaessa autolle asetettavia vaatimuksia, ei auton kojelaudassa olevaa lamppua voida verrata siihen, toteutetaanko autoon tavaratila. Useimmat asiakkaat luultavasti asettavat tavaratilan korkeammalle prioriteetille kuin kojelaudassa olevan lampun, mutta jos jonkun pitää vertailla pelkästään tavaratilan lamppua kojelaudan lamppuun, kojelaudan lamppu saattaa saada korkeamman prioriteetin. Siksi on tärkeää, että vaatimuksia ei sekoiteta eri abstraktiotasojen välillä. (Berander, 2007.)

Koska vaatimusten luonnetta määrittää pakollisuus johtaa se usein tilanteeseen, jossa jotkut sidosryhmät uskovat, että kaikilla vaatimuksilla tulisi olla sama korkein prioriteetti. Vaikka ohjelmistolle asetetut todelliset vaatimukset ovat pakollisia tietyllä ajanhetkellä, ne eivät välttämättä ole kaikille sidosryhmille samalla tavalla nykyhetkessä yhtä tärkeitä tai arvokkaita.

Ohjelmiston tai järjestelmän vaatimukset eivät useinkaan vastaa liiketoiminnan tarpeita riittävällä tasolla - ohjelmistoprojekteissa koetaan hankalaksi hankalaa tunnistaa vaatimusten prioriteettien ja liiketoiminnallisten tärkeyksien välisiä suhteita. On todettu, että laadullisten vaatimusten huomiointi projekteissa jää liian vähäiseksi, vaikka ne ovat suoraan sidoksissa järjestelmän arkkitehtuuriin, kehitys- ja ylläpitokustannuksiin, järjestelmän saatavuuteen, yhteen toimivuuteen, suorituskykyyn, siirrettävyyteen, luotettavuuteen, turvallisuuteen sekä käytettävyyteen. Näille laadullisille vaatimuksille annetaan valitettavan usein aivan liian matala prioriteetti. Niitä ei ole ohjelmistoprojekteissa useimmiten määritellä ollenkaan tai niiden määrittely on muutoin jäänyt epämääräiseksi. Mikäli laatuvaatimuksia ei ole määritetty oikein tai ollenkaan, ei niitä luonnollisestikaan voida priorisoida. (Firesmith, 2004.)

Vaikka yleisesti ohjelmistoprojekteissa myönnetään, että teoriassa erilaiset vaatimukset voivat saada erilaisia prioriteetteja, ajaudutaan kuitenkin liian usein tilanteeseen, jossa edelleen 85 - 90 % vaatimuksista on luokiteltuna korkealle prioriteetille. Tämä luokitteluongelma vähentää priorisoinnista saatavia

hyötyjä (Firesmith, 2004, Wiegers, 1999). Prioriteettien tulisi jakautua kohtuullisen tasaisesti, mutta liian tiukka pakolla tehty prioriteettien erottelu voi myös aiheuttaa ongelmia (Firesmith, 2004, Wiegers, 2000).

Resursseihin liittyvät haasteet

Firesmith (2004) toteaa vaikeaksi määrittää, kuinka projektien rajallisia resursseja tulisi käyttää kunkin vaatimuksen toteuttamiseksi. Priorisoitaessa suuria määriä vaatimuksia kohdataan usein projektin aikatauluun ja ajankäyttöön liittyviä haasteita, koska sidosryhmien väliset vaatimusneuvottelut voivat muodostua hyvin pitkiksi. Pienissä ohjelmistoalan yrityksissä vaatimusten prioriteettien käsittelyn edellyttämä aika ja henkilöresurssitarve voivat olla yrityksen selviytymisen kannalta kriittisiä. Tähän voidaan kuitenkin vaikuttaa priorisointiprosessin ketteryydellä. (Azar, Smith & Cordes, 2007.) On valitettavan yleistä, että vaatimusmäärittelyn osuus projektin budjetista jää pieneksi. Vaatimusmäärittely saa harvoin yli 4%:n osuutta projektin budjetista, vaikka useat tutkimukset osoittavat, että vaatimusmäärittelyyn panostamalla voidaan menestyä 3-4 kertaa paremmin sekä budjetin että aikataulussa pysymisen suhteen. Ohjelmistoprojekteissa vaatimustenhallintaan liittyvät toiminnot, mukaan lukien priorisointi, määritellään ennalta tietyssä ajanjaksossa suoritettaviksi (time-boxing). Tyypillisesti niihin käytetty työaika jää kuitenkin liian vähäiseksi ja varsinaista käytännön työtä joudutaan priorisoimaan. (Firesmith, 2004.)

Sanasto-ongelmat ja päätöksenteon vaikeus

Yksi keskeinen ongelma vaatimusten priorisoinnissa on yhteisen sanaston puuttuminen. Ohjelmistotuotannossa ei ole mitään yhteistä sanastoa, jolla voitaisiin ilmaista kullekin prioriteettitasolle ominaiset piirteet. Edelleenkin eri tutkijat käyttävät erilaista sanastoa kuvaamaan mitä näkökulmia priorisoinnissa tulisi ottaa huomioon (esim. tärkeys, kustannukset). Tämä on johtanut ohjelmistokehitystä koskevissa tutkimuksissa sanastojen sekaannukseen; lähes poikkeuksetta tutkijat käyttävät erilaisia sanoja kuvaamaan suositeltuja lähestymistapoja kunkin prioriteettitason sisällä. Sanasto-ongelmat tulevat esille erityisesti empiiristen tutkimusten alueella, jossa olisi tarpeen jäsentää tutkittavaa aihe-aluetta ja ehdottaa mitä sanastoa kussakin yhteydessä tulisi käyttää. (Berander, 2007.)

Ohjelmistoprojekteissa on vaikeaa saada asiakkaita päättämään, mitkä vaatimuksista ovat lopulta tärkeitä. Yhteisymmärryksen aikaansaamiseksi Sommerville ja Sawyer (Lehtola, 2003) kannattavat ns. ”keskustelevia priorisointitapoja”, joissa saatetaan eri sidosryhmät saman pöydän ääreen päättämään vapaamuotoisesti prioriteeteista. Vaatimusten priorisoinnissa on aina kyse kompromisseista eri sidosryhmien erilaisten tarpeiden ja odotusten välillä. Mikäli projektissa on yksi tai useampia vahvoja sidosryhmiä, on näiden vahvojen sidosryhmien toiveita usein vaikea laiminlyödä – ”eli kun iso asiakas käskee hyppäämään, ohjelmiston toimittaja hyppää”. Myös kaikesta valittavat sidosryhmät saavat usein ohjelmistoprojekteissa tahtonsa läpi. Näitä haasteellisia tilanteita varten tulee projektissa olla sovittuna jäsennelty tapa käsitellä eri si-

dosryhmiä. Sidosryhmiä tulee priorisoida eri tekijöiden perusteella, jolloin kunkin sidosryhmän priorisointivaatimuksille voidaan asettaa erilaisia painoarvoja. Näitä painotustekijöitä voivat olla esimerkiksi sidosryhmän tulos edelliseltä vuodelta tai sidosryhmän koko suhteessa muihin projektiin osallistuviin sidosryhmiin nähden. Sidosryhmien prioriteettien muodostamisessa voidaan hyödyntää samoja priorisointitekniikoita kuin vaatimusten priorisoinnissa. (Berander & Andrews, 2005.)

Wiegersin mukaan myös ohjelmistokehittäjät välttelevät joskus priorisointia, koska he eivät halua myöntää, etteivät kykene tuottamaan kaikkia toiminnallisuuksia käytettävissä olevan ajan puitteissa. Käyttäjät puolestaan saattavat vältellä priorisointia, koska he pelkäävät suunnittelijoiden rajaavan toteutusta toteuttamalla korkeimman prioriteetin vaatimukset ja jättämällä alemman prioriteetin vaatimuksia kokonaan toteuttamatta. Lehtola (2003) viittaa Wiegersin näkemykseen, jonka mukaan käyttäjät saattavat olla pahoine aavistuksineen oikeassa, mutta vielä huonompana vaihtoehtona nähdään tilanteet, joissa vaatimukset priorisoi sellainen henkilö, joka tuntee käyttäjän tarpeet huonosti.

3 PRIORISOINTITEKNIIKOIDEN KÄYTTÖ

Tässä luvussa käsitellään kahta priorisointitekniikkaa, joista Analytical Hierarchy Process (AHP) edustaa vaatimusten pareittain vertailua ja on erittäin monimutkainen. Numerical Assignment:ssa (Grouping) puolestaan ryhmitellään vaatimuksia ja sitä pidetään erittäin helppona tekniikkana. Luvussa käsitellään myös priorisointitekniikoiden soveltamista ja sopivimman priorisointitekniikan valintaa.

3.1 Tekniikoita vaatimusten priorisointiin

Vaatimusten priorisoimiseksi on olemassa lukuisia priorisointitekniikoita, jotka eroavat toisistaan kehitysasteensa, monimutkaisuutensa sekä vaatimusten ryhmittelyn osalta. Ohjelmistoprojekteissa priorisointitekniikoiden avulla vaatimusten priorisointi voidaan toteuttaa määrämuotoisesti. Priorisointitekniikoiden erilaisuuden kuvaamiseksi tähän alalukuun on valittu kaksi toisiinsa nähden erilaista priorisointitekniikkaa.

Analytical Hierarchy Process (AHP)
Analytical Hierarchy Process on alun perin Thomas Saatyn kehittämä järjestelmällinen päätöksentekomenetelmä, jota on alettu aikanaan soveltaa vaatimusten priorisointitekniikkana. Analytical Hierarchy Processissa ohjelmiston vaatimusten prioriteetit muodostetaan vertaamalla kaikkia mahdollisia hierarkkisesti luokiteltu vaatimuspareja toisiinsa, jotta voitaisiin löytää korkeimman prioriteetin vaatimukset ja tunnistaa niiden laajuus koko vaatimusmassasta. Tässä priorisointitekniikassa vaatimusten arviointi ja luokittelu perustuu siihen, että kullekin vaatimukselle annetaan arvo asteikolla yhdestä yhdeksään (1-9), jossa yksi (1) edustaa yhtä tärkeää vaatimusta ja yhdeksän (9) edustaa kaikkein tärkeintä vaatimusta. Kutakin vaatimusparia verrataan vain yhden kerran ja vertailtavien vaatimusparien kokonaismäärää kullakin hierarkiatasolla voidaan kuvata yhtälöllä n x (n-1) / 2, missä n kuvaa vaatimusten määrää. Tämä laskukaava johtaa siihen, että vaatimusten määrän kasvaessa vertailtavien vaatimusparien määrä kasvaa suhteessa räjähdysmäisesti. (Berander & Andrews, 2005.)

Tutkimusten mukaan Analytical Hierarchy Process ei ole soveltuva priorisointitekniikka suurien vaatimusmäärien käsittelemiseksi ja tutkijat ovatkin yrittäneet löytää tapoja vertailtavien vaatimusparien lukumäärän vähentämiseksi. Käyttämällä erilaisia variaatioita tästä tekniikasta on saatu vertailtavien vaatimusparien lukumäärää vähennettyä jopa 75 %. Alkuperäisessä tekniikan toteutusmuodossa tarpeettomatkin vaatimusparien vertailut aiheuttivat vaatimusten johdonmukaisuuden vähenemistä ja hankaloittivat mahdollisten virheiden tunnistamista.

Mikäli Analytical Hierarchy Process –tekniikkaa käytettäessä vähennetään vertailtavien vaatimusparien lukumäärää, vähenee samalla myös tarpeettomien vaatimustenparien määrä. Tämä vaikuttaa kykyyn tunnistaa ristiriidassa olevia vaatimuksia. Käytettäessä muita priorisointitekniikoita, johdonmukaisuuden vertailutarvetta ei ole, koska jokaista vaatimuksia voidaan suoraan verrata toiseen vaatimukseen ja johdonmukaisuus on näin aina varmistettu. Tutkimukset osoittavat, että henkilöt jotka priorisoivat käyttäen Analytical Hierarchy Processia päätyvät yleensä epäluotettaviin lopputuloksiin, koska prosessin kontrolli menetetään, kun verrataan vaatimuksia vain pareittain. (Berander & Andrews, 2005.)

Priorisoinnin lopputulokset esitetään tässä tekniikassa painotettuna kunkin vaatimuksen ja sen prioriteetin esittävänä luettelona. Mikäli priorisoinnissa ei käytetä tekniikan hierarkista ominaisuutta, puhutaan tällöin yleisemmin tekniikan ”litteästä” soveltamisesta. Vaatimusten pareittainen vertailu ei kuitenkaan ole mitenkään ainutlaatuinen käsittelytapa Analytical Hierarchy Processille, vaan sitä hyödyntävät myös muut priorisointitekniikat kuten kuplalajittelu. Tällöin pareittainen vertailu toteutetaan enemmän käytännön toimena kuin varsinaisena arviointitekniikkana. (Berander, 2007.)

Numerical assignment (Grouping)

Numerical assignment on yleisin priorisointitekniikka ja sitä suositellaan käytettäväksi RFC 2119- ja IEEE 830-1998 –ohjelmistokehitysstandardeissa. Tekniikka perustuu vaatimusten ryhmittelyyn ja jakamiseen eri prioriteettiryhmiin. Prioriteettiryhmien lukumäärä voi vaihdella, mutta yleensä käytetään kolmea eri ryhmää: korkea (high), keski (medium) ja matala (low). On tärkeää, että kunkin prioriteettiryhmän merkitys vallitsevassa viitekehyksessä kuvataan sellaisessa muodossa, että ohjelmistokehitykseen osallistuvat sidosryhmät ymmärtävät niiden merkityksen ja voivat sitoutua niihin. Käyttämällä pelkästään termejä korkea, keski tai matala aiheutetaan tyypillisesti sekaannusta sidosryhmille. Tästä syystä on tarpeen määritellä sidosryhmille vaatimusten eri prioriteettien merkitykset yleisemmässä muodossa. Yleensä ohjelmistokehityksessä päädytäänkin luokittelemaan prioriteettiryhmät kriittisiin sekä vakio- tai valinnaisiin vaatimuksiin. Tämä luokittelu näyttää tutkimusten mukaan olevan erityisen tärkeää silloin, kun sidosryhmillä on erilaisia näkemyksiä siitä, mitä korkeilla, keski- tai matalan prioriteetin tasoilla tarkoitetaan. Selkeä määritelmä kunkin prioriteettiryhmän merkityksestä minimoi ongelmia. (Berander, 2007.)

Berander (2007) toteaa, että vaikka Numerical assignment on priorisointitekniikkana melko yksinkertainen, kohdataan sen käytön yhteydessä usein yleisempi vaatimusten priorisointia koskeva ongelma, jossa sidosryhmät ajattelevat kaikkien vaatimusten olevan kriittisiä. Yksi vaihtoehto tämän ongelman korjaamiseksi on asettaa rajoituksia sille, kuinka monta vaatimusta voidaan ryhmitellä kuhunkin prioriteettiryhmään. Voidaan esimerkiksi asettaa ehto, jonka mukaan kunkin prioriteettiryhmän tulee sisältää vähintään neljännes (25 %) vaatimuksista.

Rajoitusten asettaminen saattaa Beranderin mukaan (2007) aiheuttaa ongelmia priorisoinnista saatavan hyödyn näkökulmasta, koska sidosryhmät ovat tällöin pakotettuja jakamaan vaatimukset tiettyihin prioriteettiryhmiin. Rajoitusten aiheuttamista hyvistä tai huonoista vaikutuksista ei kuitenkaan ole olemassa empiiristä näyttöä. Priorisoinnin lopputuloksena vaatimusten prioriteetit esitetään tässä tekniikassa järjestysluvuittain asteikolle sijoitettuna. Numerical assignment –tekniikan käytön yhteydessä on kuitenkin tärkeää huomioida, että kaikilla tiettyyn prioriteettiryhmään sijoitetuilla vaatimuksilla on sama prioriteetti, eikä mikään yksittäinen vaatimus saa omaa ainutlaatuista prioriteettia.

3.2 Priorisointitekniikoiden soveltamisesta

Käytettävä priorisointitekniikka on aina riippuvainen ohjelmistoprojektin tarpeista. Sopivinta priorisointitekniikkaa valittaessa onkin arvioitava, mitä priorisoinnilla odotetaan kyseisessä ohjelmistoprojektissa saavutettavan ja millä tasolla vaatimusten välisiä tärkeyseroja halutaan kuvata.

Priorisointitekniikan valinta

Ohjelmistokehityksessä vaatimusten priorisoimiseksi on otettu käyttöön useita tekniikoita, jotka toimivat kukin eri mitta-asteikoilla keskittyen eri näkökulmiin. Tekniikoiden kehittyneisyysaste vaihtelee suuresti korkean tason menetelmäkuvauksista yksityiskohtaisiin priorisointialgoritmeihin. (Berander, 2007.) Riippumatta valitusta priorisointitekniikasta, vaatimusten priorisoimisessa on aina kyse vaatimusmäärittelyssä syntyneiden vaatimuksien käyttämisestä syötteinä varsinaiselle priorisointiprosessille. Useimmiten näille vaatimustiedoille tehdään joitakin kuhunkin priorisointitekniikkaan liittyviä laskelmia ennen vaatimusten välisten prioriteettien esittämistä. (Berander, Khan & Lehtola, 2006.)

Sopivinta priorisointitekniikkaa valittaessa on arvioitava, millä tarkkuustasolla priorisointi on tarpeen tehdä. Yleisenä ohjeena on todettu, että priorisointitekniikkaa valittaessa tulisi pyrkiä valitsemaan yksinkertaisin priorisointitarpeen täyttävä tekniikka. Kehittyneempiä ja monimutkaisempia priorisointitekniikoita tulisi käyttää silloin, kun on tarpeen ratkaista erimielisyyksiä tai tukea priorisoinnilla kriittisimpiä päätöksiä. (Berander, 2007.) Berander ja Andrews (2005) mukaan sopivimman priorisointitekniikan valinnassa voidaan vertailla tekniikoiden ominaisuuksia. Taulukossa 1 esitetään ote kahden priorisointitekniikan ominaisuuksista.

TAULUKKO 1: Ote priorisointitekniikoiden yhteenvedosta. (Berander & Andrews, 2005)

Tekniikka	Asteikko	Rakeisuus	Monimutkaisuus
Analytical Hierarchy Process	suhteellinen	hienojakoinen	erittäin monimutkainen
Numerical Assignment	järjestys (korkea, keski, matala)	karkea	erittäin helppo

Taulukossa 1 asteikolla ilmaistaan vaatimusten välisten tärkeyserojen mittakaavaa priorisointitekniikoissa. Taulukossa Analytical Hierarchy Process –priorisointitekniikan kohdalla suhteellinen asteikko ilmaisee, onko jokin vaatimus tärkeämpi kuin jokin toinen vaatimus. Tarkoituksena on, että kukin vaatimus saa lukuarvon. Numerical Assignment –priorisointitekniikassa kullekin vaatimukselle asetetaan ennalta määritellystä arvojoukosta (korkea, keski, matala) jokin tietty arvo. Priorisointitekniikan rakeisuus ilmaisee, kuinka laajasti kyseisessä tekniikassa voidaan huomioida eri tekijöitä. Monimutkaisuus kuvaa kyseisen tekniikan helppokäyttöisyyttä.

Kehittyneempien priorisointitekniikoiden käyttö vaatii enemmän aikaa, joten mahdollisimman yksinkertaisen tekniikan käytöllä voidaan varmistaa kustannustehokkuutta. Priorisointitekniikkaa valittaessa tulisikin päättää, kuinka "nopea ja likainen" ("quick and dirty") priorisoinnin lähestymistapa voi olla ilman, että päätöksenteon laatu kärsii. Päätöksenteon kannalta olennaisia tietoja ovat esimerkiksi vaatimuksen toteuttamiseen liittyvä kustannusarvio tai seuraamukset vaatimuksen toteutumattomuudesta. Markkinoilla on olemassa useita kaupallisia työkaluja, jotka helpottavat kehittyneempien priorisointitekniikoiden käyttöä (esim. Analytical Hierarchy Process). Lisäksi on mahdollista rakentaa yksinkertaisia apuvälineitä eri priorisointitekniikoiden käyttämiseksi. Tällaisia apuvälineitä voivat olla esimerkiksi taulukkolaskentaohjelmat. (Berander, 2007.)

Priorisointitekniikoiden yhdistäminen priorisointiprosessissa

Joitakin priorisointitekniikoita on mahdollista yhdistää, jolloin priorisoinnista voidaan saada helpompaa ja tehokkaampaa. Esimerkiksi priorisointitekniikoiden yhdistämistä sovelletaan Planning Game (PG) –priorisointitekniikassa osana eXtreme Programming (XP) –ohjelmistokehitysmenetelmää. Planning Gamessa vaatimukset jaetaan ensin prioriteettiryhmiin ja tämän jälkeen kunkin prioriteettiryhmän sisällä asetetaan vaatimukset arvojärjestykseen (ranking). (Berander & Andrews, 2005.)

Toisena esimerkkinä priorisointitekniikoiden yhdistämisestä voidaan mainita vaatimusten triage, jossa vaatimusten prioriteetin arviointi rinnastetaan lääketieteelliseen riskinarvioon sairaaloissa. Lääketieteellisessä triagessa lääkintähenkilökunta jakaa potilaat kolmeen ryhmään: 1. Potilaat, jotka kuolevat joka tapauksessa saivat he hoitoa tai eivät 2. Potilaat jotka jatkavat normaalia elämää saivat he hoitoa tai eivät 3. Potilaat, joille hoidosta on merkittävästi hyötyä.

Vertauskuvana edellä kuvattuun lääketieteelliseen riskinarvioon nähden ohjelmistotuotannossa tunnistetaan vaatimuksia, jotka on sisällytettävä ohjelmistoon, esimerkiksi alustan vaatimukset. Yleensä tunnistetaan vaatimuksia,

joita tuotteen ei selvästikään tarvitse täyttää, esimerkiksi erittäin vahvasti optioiksi lukeutuvat vaatimukset. Jäljelle jäävät ne vaatimukset, jotka tarvitsevat enemmän huomiota. Käytännössä kukin vaatimus on määritetty yhteen näistä kolmesta prioriteettiryhmästä kuten Numerical Assignment -tekniikassa. Ne vaatimukset, jotka tarvitsevat enemmän huomiota priorisoidaan millä tahansa muulla tekniikalla kuten esimerkiksi käyttämällä Analytical Hierarcy Processia. Näin kaikkia vaatimuksia ei tarvitse käsitellä kehittyneemmällä priorisointitekniikalla ja tämä mahdollistaa priorisoinnin vähemmällä vaivalla. (Berander & Andrews, 2005.)

4 YHTEENVETO

Tässä tutkielmassa on esitelty tieteellisen lähdemateriaalin pohjalta asiakasvaatimusten priorisointia osana vaatimustenhallintaa. Lisäksi tutkielmassa on tuotu esille joitakin keskeisiä priorisointiin vaikuttavia asioita ja käsitelty priorisointitekniikoiden käyttöä. Tutkielmassa on kuvattu myös vaatimusten priorisoinnissa esiintyviä haasteita.

Tutkielmassa haettiin vastausta tutkimuskysymykseen: Miten asiakasvaatimuksia priorisoidaan osana vaatimustenhallintaa? Tutkimuskysymys jakautui edelleen alikysymyksiksi: 1. Mitkä asiat vaikuttavat ohjelmistotuotannossa asiakasvaatimusten priorisointiin? 2. Millaisia priorisointitekniikoita on olemassa asiakasvaatimusten priorisoimiseksi?

Ohjelmistotuotannossa ohjelmistolle asetettavat asiakasvaatimukset ovat kaiken projekteissa tehtävän työn perusta riippumatta siitä, käytetäänkö ohjelmiston kehittämisessä ketteriä vai perinteisempiä lähestymistapoja. Mikäli asiakasvaatimuksia ei osata tunnistaa ja käsitellä oikein projektin aikana, on projektin lopputulos aina huono ja asiakas on tyytymätön - hoidetaan projekti muilta osin kuinka hyvin tahansa.

Firesmith (2004) toteaa, että ohjelmistotuotannossa on mahdotonta toteuttaa kaikkia vaatimuksia - ei ainakaan yhteen tiettyyn ohjelmistoversioon. Ohjelmistoon toteuttavat toiminnallisuudet edellyttävät aina päätöksentekoa, jossa ohjelmistoon kohdistuvia vaatimuksia verrataan projektissa käytettävissä oleviin resursseihin (henkilöresurssit, aika, raha) ja niiden asettamiin rajoitteisiin. Tämän päätöksenteon tukena käytetään vaatimusten priorisointia, jonka tavoitteena on asettaa vaatimukset tärkeysjärjestykseen.

Vaatimusten tärkeysjärjestyksen perusteella voidaan tehdä lopulliset valinnat vaatimuksista, jotka on sisällytettävä ohjelmistoon sen elinkaaren aikana. Ennen kuin vaatimusten lopullinen tärkeysjärjestys voidaan muodostaa, on otettava huomioon lukuisia eri asioita, jotka vaikuttavat varsinaisen ohjelmiston toteuttamiseen. Näitä priorisointia ohjaavia asioita ovat esimerkiksi kunkin vaatimuksen tärkeys (merkitys) kullekin sidosryhmälle, vaatimuksen toteuttamisesta tai toteutumattomuudesta koituvat seuraamukset, toteuttamiskustannukset sekä tekniset- ja markkinariskit.

Ohjelmistokehitykseen liittyy useita eri sidosryhmiä, joilla kullakin on erilaisia vaatimuksia koskevia intressejä, näkemyksiä ja odotuksia. Asiakkaat ovat vaatimusten kannalta keskeisin sidosryhmä, koska heidän tarpeitaan varten ohjelmistoa kehitetään. Sidosryhmien näkökulmien yhdistäminen ja ohjelmistoon toteutettavia toiminnallisuuksia koskevan yhteisymmärryksen löytäminen ovat yksi priorisoinnin haasteellisimmista tehtävistä. Kysymys on aina vaatimusten tulkinnasta, tulkinnan kontekstista sekä vaatimuksia koskevasta neuvottelusta. Jotta vältetään vaatimusten epäselvyydet vaatimusten tulkinnassa, kannatetaan yleisesti aktiivista ja avointa vuoropuhelua eri ohjelmistoprojektin sidosryhmien kesken. Tällä varmistetaan, että sidosryhmien erilaiset tarpeet on tehty näkyviksi kaikille osapuolille.

Vaatimusten priorisoimiseksi on kehitetty kehitysasteeltaan ja monimutkaisuudeltaan eritasoisia tekniikoita, joita soveltamalla vaatimuksia koskevasta monimutkaisesta päätöksenteosta voidaan saada määrämuotoisempaa ja dokumentoitua. Tutkielmassa esitelty Analytical Hierarchy Process –tekniikka mahdollistaa moniulotteisen vaatimusten priorisoinnin, mutta sitä pidetään monimutkaisena ja työläänä. Toinen tutkielmassa esitelty tekniikka Numerical Assignment (Grouping) on hyvin yksinkertainen ja yleisimmin käytetty.

Priorisointitekniikkaa valittaessa tulisi pyrkiä valitsemaan sellainen yksinkertaisin priorisointitarpeen täyttävä tekniikka, jonka käyttö ei heikennä päätöksenteossa tarvittavien tietojen laatua. Kehittyneempiä ja monimutkaisempia priorisointitekniikoita tulisi käyttää vain silloin, kun on tarpeen ratkaista erimielisyyksiä tai tukea priorisoinnilla kriittisimpiä päätöksiä (Berander, 2007.). Tutkielmassa on kuvattu esimerkein, kuinka priorisointitekniikoiden soveltamisessa voidaan vaatimusten ryhmittelyn avulla yhdistää eri tekniikoita. Yhdistämällä tekniikoita voidaan vähemmän tärkeitä vaatimuksia priorisoida kevyemmällä tekniikalla ja vastaavasti tärkeimpiä vaatimuksia monimutkaisemmalla tekniikalla.

Tutkielman aihe-alueelta löytyy melko paljon tieteellistä lähdemateriaalia, mutta niiden taso ja diskurssi vaihtelevat suuresti. Tutkielman lähdeaineistossa esiintyy paljon samojen asioiden esittelyä kirjoittajasta riippumatta. Syväluotaavampi tieteellinen pohdinta esimerkiksi eri sidosryhmien käsittelystä ohjelmistoprojektin sisällä jää lähdeaineistossa vähäiseksi, vaikka juuri tätä osa-aluetta koskevia tutkimustuloksia tarvittaisiin ohjelmistoprojekteissa käytännön työn tueksi. Joidenkin vaatimusten priorisointia koskevien tutkimusten mukaan (Lehtola, Kauppinen ja Kujala, 2004) tutkituissa organisaatioissa ei yhdessäkään kyetty tarkemmin kuvaamaan miten prioriteetteja tulisi tehokkaasti määrittää, muokata tai viestiä niistä projektin jäsenille.

Lähdemateriaali antaa kuitenkin lähtökohtia ja teemoja vaatimusten priorisointia koskevan omaehtoisen pohdinnan käynnistämiseksi. Kuten on todettu, ei ole olemassa ainoastaan yhtä ”hyvää” vaatimustenmäärittelyprosessia. Mahdollisia tapoja järjestää vaatimusten määrittelyä ja hallintaa on useita, eikä käytäntöjen soluttaminen sellaisinaan yhdestä organisaatiosta toiseen ole yksinkertaista. Kotonya ja Sommerville (1998) suosittelevat, että jokaisen organisaation tulisi kehittää oma prosessinsa, joka soveltuu yrityksessä kehitettävien järjes-

telmien suunnitteluun, organisaatiokulttuuriin ja ihmisten kokemustasoon vaatimusten määrittelyssä ja hallinnassa. (Lehtola, 2003, Kotonya & Sommerville 1998.) Tämä toteamus pätee myös vaatimusten priorisoinnin toteuttamiseen organisaatioissa.

Tutkielman johtopäätöksinä voidaan todeta, että on olennaista ymmärtää laajasti vaatimusten priorisointia määrittäviä ja ohjaavia asioita, jotta olemassa olevia priorisointitekniikoita voidaan soveltaa parhaiten kussakin ohjelmistoprojektissa. Priorisointitekniikoiden yksityiskohtaisen tuntemuksen (algoritmi, laskentakaava) sijaan on olennaisempaa tiedostaa, minkä tyyppinen priorisointitekniikka sopii parhaiten kuhunkin priorisointitarpeeseen ja millä tavalla siinä voidaan huomioida kunkin sidosryhmän tarpeita ja tärkeyksiä.

Vaikka priorisointitekniikoiden soveltaminen itsessään voi olla hyvinkin suoraviivaista, lisää sidosryhmien tarpeiden, odotusten ja tärkeyksien merkitysten erilaisuuden huomioon ottaminen ja harmonisointi huomattavasti priorisoinnin monimutkaisuutta. On tärkeää määrittää etukäteen, millä periaatteilla priorisointia tullaan tekemään ja sitouttaa kaikki sidosryhmät yhdessä sovittujen periaatteiden noudattamiseen. Tämän lisäksi on vielä huomioitava projektissa vaikuttavat muut rajoitteet kuten resurssit ja niiden riittävyys.

Mikäli samassa ohjelmistoprojektissa on asiakkaina ja käyttäjinä useita eri toimijoita, nousee vaatimusten priorisointi ja harmonisointi esimerkiksi hankinnan näkökulmasta entistä merkityksellisemmäksi. Näissä monitoimijaprojekteissa tietty osa tunnistetuista vaatimuksista voi muodostua kehitettävän ohjelmiston yhteisiksi ydinvaatimuksiksi osan vaatimuksista palvellessa vain yhtä sidosryhmää. Priorisoinnin määrämuotoisuudella ja koko priorisointiprosessin johtamisella pitäisi kuitenkin tavoitella tilannetta, jossa kukin sidosryhmä kokee olevansa lopulta voittaja (win-win -tilanne), vaikka sidosryhmän omista vaatimuksista on useimmiten jouduttu kompromissien vuoksi tinkimään.

Julkishallinnossa on käynnissä useita tietojärjestelmähankkeita, jotka toteutetaan viranomaisyhteistyössä. Julkishallinnossa vaatimusten priorisoinnin toteuttamisperiaatteet ja priorisointiprosessin määrämuotoisuus vaihtelevat projektien välillä, joten vaatimusten priorisointitoiminnan kehittämiselle löytyy vahva tilaus. Tämän tutkielman toivotaan osaltaan antavan syötteitä priorisoinnin kehittämiselle julkishallinnossa. On tärkeää huomioida, että vaatimusten priorisoinnilla voidaan rajata hankinnan kohdetta ja vaikuttaa merkittävästi toteutettavan ohjelmiston kustannustehokkuuteen.

Monitoimijaprojekteissa tapahtuvaa vaatimusten priorisointia on tutkittu tutkielman lähdemateriaalin perusteella verrattain vähän. Tämän osa-alueen tutkiminen on perusteltua erityisesti priorisointiin liittyvän vaatimusten luokittelun, sidosryhmien erilaisten tarpeiden ja tärkeyksien huomioimisen sekä monitoimijaprojektiin soveltuvien priorisointitekniikoiden käytön kannalta.

LÄHTEET

Azar, J., Smith, R. K. & Cordes, D. (2007). Value-oriented requirements prioritization in a small development organization. *Software, IEEE 24* (1), 32-37.

Berander, P. & Andrews, A. (2005). *Requirements Prioritization.*

Berander, P. & Ber, P. (2007). *Evolving prioritization for software product management.*

Berander, P., Khan, K. A. & Lehtola, L. (2006). *Towards a research framework on requirements prioritization. SERPS 6*, 18-19.

Farlex Inc., (2012). The Free Dictionary By Farlex: haku sanalla ”prioritization”. Haettu 19.12.2012 osoitteesta www.thefreedictionary.com/prioritization

Firesmith, D. (2004). *Prioritizing requirements. Journal of Object Technology 3* (8), 35-47.

Haikala, I, Märijärvi, J, (2000). *Ohjelmistotuotanto.* 7. uudistettu painos. Helsinki: Satku – Kauppakaari Oyj.

Kallio, J. (2008). *Vaatimustenhallinta ja sen kehittäminen ohjelmiston elinkaaren näkökulmasta.* Tietojärjestelmätieteen pro gradu –tutkielma. Jyväskylän yliopisto.

Kauppinen, M., Vartiainen, M., Kontio, J., Kujala, S. & Sulonen, R, (2004). *Implementing requirements engineering processes throughout organizations: success factors and challenges. Information and Software Technology* 46: 937–953.

Kotonya, G. & Sommerville, I. (1998). *Requirements Engineering: Processes and Techiques*. New York: John Wiley & Sons Ltd.

Lehtola, L. (2003). *Käyttäjä- ja asiakasvaatimusten priorisointi: Priorisointikäytännöt*. Diplomityö. Helsingin Teknillinen korkeakoulu.

Lehtola, L., Kauppinen, M. & Kujala, S. (2004). *Requirements prioritization challenges in practice. Product focused software process improvement* , 497-508.

Möttönen, M. (2009). *Requirements engineering : linking design and manufacturing in ICT companies.* Oulu: University of Oulu.

Overhage, S., Schlauderer, S. & Birkmeier, D. (2011). What Makes IT Personel Adopt Scrum? A Framework of Drivers and Inhibitors to Developer Acceptance. Teoksessa R., H. Sprague, Jr. (toim.), *Proceedings of the 44th Annual Hawaii International Conference on System Sciences.* Kolua, Kauai, Hawaii, 4–7 January. (s. 1–10). Los Alamitos: IEEE Computer Society.

Regnell, B., Höst, M., Natt och Dag, J., Beremark, P. & Hjelm, T. (2000). Visualization of agreement and satisfaction in distributed prioritization of market requirements. *Proceedings of 6th International Workshop on Requirements Engineering: Foundation for Software Quality*. Citeseer.

Ruuska, T. (2012). *Vaatimusmäärittely ketterässä ohjelmistokehityksessä.* Tietojärjestelmätieteen pro gradu –tutkielma. Jyväskylän yliopisto.

Sommerville, I. & Sawyer,P. (1997). *Requirements Engineering: A Good Practice Guide.* New York: John Wiley & Sons.

Taylor, J. (2004). *Managing information technology projects : applying project management strategies to software, hardware, and integration initiatives.* New York: American Management Association.

Thayer, R. & Thayer, M. (1997). *Software Requirements Engineering Glossary, in: Thayer R & Dorfman M (Eds.), Software Requirements Engineering, Second ed., IEEE Computer Society Press.* Los Alamitos, CA, USA: 489–528.

Wiegers, K. E. cop. (2006). *More about software requirements : thorny issues and practical advice.* Redmond (WA): Microsoft Press.

Wiegers, K.(2000). *Software Testing & Quality Engineering*.