Tuoteomistajan rooli ketterässä ohjelmistokehityksessä

Jaakko Nygrén

Pro Gradu –seminaarin kirjallinen alustus
Helsinki 14.2.2012

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Sisältö

Johdanto

Viidestä viiteentoista prosenttiin ohjelmistoprojekteista keskeytetään epäonnistumisen takia ja vielä useampi projekti myöhästyy tai ylittää budjetin. Robert Charetten artikkelissa Why Software Fails [1] esitellään useita epäonnistuneita projekteja sekä kaksitoista yleistä syytä projektien epäonnistumiseen. Näistä viisi liittyy suoraan haasteisiin projektinhallinnassa sekä liiketoiminnan ja tuotannon kommunikoinnissa. Artikkelin mukaan virheet projektien hallinnassa ovat suurin yksittäinen luokka syitä projektien epäonnistumiseen. Suuri osa epäonnistumisista voitaisiin välttää, mutta vaikka epäonnistuminen saattaa riskeerata koko liiketoiminnan tulevaisuuden, ei sen välttämiseen kiinnitetä tarpeeksi huomiota.

Software Engineering Institute (SEI) on kehittänyt Capability Maturity Model Integration for Development (CCMI-DEV) kypsyysmallin auttaakseen organisaatioita arvioimaan kykyä ohjelmistojen ja palveluiden tuotantoprosessin toteutuksen arviointiin ja parantamiseen [2]. SEI:n vuodelle 2011 julkaisemista mittaustuloksista [3] selviää, että arvosteluun osallistuneiden organisaatioiden keskimääräinen kypsyysaste on alle arvosteluasteikon keskitason. Todellisuudessa ohjelmistoja ja palveluita tuottavien organisaatioiden kypsyysaste on todennäköisesti vielä heikompi, sillä arvioinnissa ovat mukana vain arvion tilanneet organisaatiot. Vaikka projektityön määrä on edellisen vuosikymmenen aikana kasvanut huomattavasti, projektinhallinnan teoria kohtaa jatkuvasti kritiikkiä. Projektinhallinnan menetelmien käytännön toteutuminen eroaa huomattavasti menetelmien määritelmistä [4]. Koska projektit eivät ole yhteneväisiä, projektinhallinnan menetelmiä tulisikin soveltaa projekteihin sopiviksi [5], [6].

Ketterissä ohjelmistokehitysprosesseissa projektin hallinnan tehtävät on jaettu koko kehitystiimille. Asiakkaan edustaja eli tuoteomistaja vastaa budjetista, resursoinnista sekä tuotteen tavoitteista. Kehittäjät ovat vastuussa tehtävien toteutustavan määrittelystä ja toteutuksen keston arvioinnista. Tuoteomistajan tärkeimmät tehtävät liittyvätkin tuotteen kokonaiskuvan muodostamiseen, sen onnistuneeseen viestintään sekä toteutuksen ohjaamiseen käytännössä. Näiden tehtävien puutteellinen suoritus voi johtaa projektin epäonnistumiseen joko teknisesti tai liiketoiminnallisesti [7], [8].

Kuinka tuoteomistajan roolia voidaan kehittää ketterässä ohjelmistokehityksessä? Tässä

tutkielmassa esitellään malli tuoteomistajan roolin optimoimiseksi projektikohtaisesti. Tutkielman toisessa luvussa kartoitetaan ketterän ohjelmistokehityksen pääperiaatteet ja tuoteomistajien keskeisimmät tehtävät kirjallisuudesta. Kolmannessa luvussa tuoteomistajan keskeisimpien tehtävien suoritusta ja merkitystä analysoidaan tapaustutkimuksen menetelmällä kolmessa eri ohjelmistokehitysprojektissa. Neljännessä luvussa kuvaillaan tapaustutkimuksessa esiintyneet tuoteomistajan tärkeimmät tehtävät sekä niiden laiminlyönnin vaikutukset. Lopuksi esitetään malli tehtävien merkityksen arvottamiseksi projektikohtaisesti ja ehdotetaan mallin avulla parannuksia tuoteomistajan tehtävien suorittamiseen yhdessä kolmesta esimerkkiprojekteista.

Tuoteomistaja ketterässä ohjelmistokehityksessä

Ketterän ohjelmistokehityksen pääperiaatteet

Ketterien ohjelmistokehitysmenetelmien pääperiaatteet on koottu yhteen Agile Manifesto:ssa eli ketterän ohjelmistokehityksen julistuksessa. Julistuksessa kehotetaan arvostamaan

- Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja
- Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota
- Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja
- Vastaamista muutokseen enemmän kuin pitäytymistä suunnitelmassa [9]

Tässä luvussa esitellään ketterän ohjelmistokehityksen keskeisimmät pääpiirteet tuoteomistajan roolin kannalta.

Yksilöt ja itseohjautuva ohjelmistokehitystiimi

Ketterässä ohjelmistokehityksessä vastuu projektin onnistumisesta on kehitystiimillä. Kehitystiimi on itseohjautuva ja sillä on perinteisiä menetelmiä laajemmat rajat tehdä luovia ratkaisua päästäkseen tavoitteeseen. Koska vastuu projektin onnistumisesta on tiimillä, myös valta tehdä päätöksiä on oltava tiimissä. Itseohjautuva tiimi ei tarkoita tiimiä ilman projektinhallintaa vaan tiimiä jolla on mahdollisuus vastata muutoksiin ja organisoitua uudelleen tarvittaessa. Tarve muutoksiin tulee eri sidosryhmien

muuttuvista ja tarkentuvista tarpeista. Tiimin ratkaisukykyyn optimaalisen ratkaisun löytämiseksi luotetaan. Tämä korostaa entisestään tiimin jäsenten yksilöllisten, ketteriin menetelmiin soveltuvien tietojen, taitojen ja luonteenpiirteiden merkitystä. Ketteryys vaatii organisaatiolta kykyä luottaa tiimin jäsenten kompetenssiin sekä kykyä tehdä päätöksenteko tiimille näkyväksi [5]. Ketterät ohjelmistokehitysmenetelmät painottavat prosessin aikaista, todelliseen palautteeseen perustuvaa, elävää määrittelyprosessia. Kattavan määrittelydokumentaation sijaan kehitystiimillä on mahdollisuus käyttää luovuuttaan yhteisen päämäärän saavuttamiseksi [7].

Extreme Programming –menetelmässä kehittäjät työskentelevät tiiviisti asiakkaan kanssa muuntaakseen käyttötapaukset toimivaksi koodiksi. Kehittäjien vastuulla on valita vapaasti yksinkertaisin toteutustapa hyväksymiskriteereiden täyttämiseksi, sovittuja ohjelmointistandardeja seuraten. Menetelmän työtavat, kuten pariohjelmointi, kannustavat tiiviiseen yhteistyöhön myös kehittäjien välillä [8].

Aikainen julkaisu ja iteratiivisuus

Ketterissä kehitysprosesseissa pyritään mahdollisimman aikaisessa vaiheessa tuottamaan ja julkaisemaan sidosryhmien kannalta kriittisimmät ominaisuudet. Tämä mahdollistaa sekä tuotteen aikaisemman käyttöönoton, aikaisemman liiketoiminnallisen hyödyn että prosessin aikaisen tuotekehityksen ja asiakasarvon analysoinnin. Aikaisella julkaisulla pyritään myös havaitsemaan tuotteen mahdollinen epäonnistuminen perinteisiä menetelmiä aikaisemmin. Mikäli tuote ei tule onnistumaan, sen toteuttamiseen on käytetty vähemmän resursseja kun epäonnistuminen havaitaan jo kehitysprosessin aikana. Tuotteen varsinaisten loppukäyttäjien sekä muiden sidosryhmien todellisten tarpeiden jatkuvan analysoinnin mahdollistamiseksi tuotteesta pyritään julkaisemaan parannuksia jokaisessa iteraatiossa. Tuotteen elinkaaren vaiheesta riippuen iteraatioissa toteutettavat parannukset voidaan julkaista tuotannossa tai vain rajatulle testiryhmälle [7].

Laajan vaatimusmäärittelyn ja haastavien aikataulujen sijaan Extreme Programming –menetelmällä pyritään esittämään asiakkaalle ja kehitystiimille kysymys ”Miten toteuttaisit tuotteen annetussa ajassa ja resursseilla?”. Kysymys kehottaa valitsemaan yksinkertaisimman mahdollisimman työtavan ongelman ratkaisuun. Se vaatii asiakasta poimimaan tärkeimmät toiminnallisuudet ja antamaan teknisille asiantuntijoille eli

kehittäjille vastuun niiden toteuttamisesta. Kehittäjät tukevat asiakasta arvioimalla tietyn vaatimuksen toteutuksen keston [8].

Pyrkimys täydellisen tuotteen suunnitteluun ennen sen toteutusta johtaa usein kalliiseen ja monimutkaiseen tuotteeseen joka ei toimi oikein [10]. Vaikka kehitysprosessina seurattaisiin jotain ketterää mallia, suunnittelulähtöinen prosessi, jossa pyritään oikeaan ratkaisuun ensimmäisellä yrityskerralla epäonnistuu usein. Tuotesuunnittelun menetelmästä riippumatta vasta varsinainen toteutus tai sen toiminnallinen prototyyppi on testattavissa laajalti tuotteen todellisilla käyttäjillä [7].

Extreme Programming –menetelmä painottaa nopeuttamaan kehitystä ja aikaistamaan tietyn toiminnallisuuden julkaisua pyrkimällä toteuttamaan yksinkertainen mahdollinen toimiva toteutus, joka täyttää hyväksymiskriteerit ja läpäisee testit. Menetelmä tuo tarpeen jatkuvalle refaktoroinnille ja testilähtöiselle ohjelmistokehitykselle. Lopputuloksena pyritään aikaan saamaan mahdollisimman ilmaisuvoimainen, kattavasti testattu ja toimiva tuote. Tulevaisuuden tarpeisiin varautuminen koetaan liian kalliiksi ja riskialttiiksi. Ajankäyttö tällä hetkellä tarpeettoman modulaariseen toteutukseen tekee sovelluksesta monimutkaisemman kuin tarpeellista ja vie resursseja tärkeimpien eli kyseiseen iteraatioon valittujen tehtävien toteuttamiselta [8].

Asiakkaan ja sidosryhmien osallistuminen tuotteen kehitysprosessiin

Toteutettavan tuotteen ainoa tarkoitus on täyttää sen käyttäjien ja asiakkaiden tarve. Tämän mahdollistamiseksi käyttäjät, asiakas ja muut sidosryhmät pyritään osallistamaan prosessiin mahdollisimman tiiviisti [7]. Palautteen saannin ja siihen reagoimisen nopeutta pyritään tehostamaan tuomalla sidosryhmien edustajat ja käytettävyysasiantuntijat tai oikea asiakas ja loppukäyttäjä mukaan tiimiin [5]. Scrum –menetelmässä asiakasta edustava tuoteomistaja käsittelee ja analysoi eri sidosryhmien tarpeet ja priorisoi tuotteen tehtäväjonon. Extreme Programming -menetelmässä asiakas (eng. customer) on jatkuvasti mukana tiimissä ja vastuu työn toteutumisesta on jaettu kehittäjille ja asiakkaalle. Menetelmässä kehittäjien (eng. developer) velvollisuus on välttää liiketoiminnallisiin ratkaisuihin osallistuminen. Asiakkaan velvollisuus on antaa kehittäjille vapaat kädet optimaalisen teknisen ratkaisun löytämiseksi. Projektin etenemiseksi osapuolten on siis tehtävä tiivistä yhteistyötä. Tämä tekee asiakkaan ja kehittäjien suhteen tiiviimmäksi ja tehostaa kommunikaatiota [8].

Oppimissuhde asiakkaan edustajien ja kehittäjien välillä on kaksisuuntainen: jatkuva läsnäolo mahdollistaa nopean reagoinnin asiakkaan tarpeiden tarkentumiseen ja tekee asiakkaalle helposti nähtäväksi miten tiimi on käsittänyt ongelmakentän [5].

Tuotteen määrittelyprosessi, markkinatutkimus ja liiketoiminta-analyysi tehdään perinteisissä ohjelmistokehitysprosesseissa varsinaisen toteutusprosessin ulkopuolella ja yleensä ennen toteutuksen aloittamista. Loppukäyttäjien palaute kerätään markkinatutkimuksessa vasta tuotteen julkaisun jälkeen. Ketterät ohjelmistokehitysmenetelmät suosittelevat tuotesuunnittelun ja markkinatutkimuksen tuomista osaksi kehitysprosessia. Tuotteen kehittäminen aloitetaan mahdollisimman karkeasta visiosta, joka kuvaa mitä tuotteen täytyy välttämättä tehdä täyttääkseen loppukäyttäjän vaatimukset. Tuotteen muodostuminen (eng. Product Discovery) on jatkuva prosessi, jota ruokkii jatkuva asiakkailta ja loppukäyttäjiltä kerättävä palaute. Markkinoiden ja asiakkaan tarpeiden kattava ymmärrys ennen tuotantopäätöksen tekemistä on tärkeää tuotteen onnistumisen kannalta, mutta liian yksityiskohtainen tuotesuunnittelu jättää liian vähän tilaa tuotteen kehittymiselle kehitysprosessin aikana. Toisaalta kuvitelma asiakkaiden tarpeiden ymmärtämisestä ilman kattavaa tutkimusta on usein virheellinen. Ketterillä ohjelmistokehitysmenetelmillä pyritään välttämään molempia ääripäitä osallistamalla asiakas kehitysprosessiin nopeatahtisen julkaisemisen ja jatkuvan todelliseen käyttöön perustuvan asiakasymmärryksen kartoittamisen keinoilla [7].

Ketterien ohjelmistoprojektien menetelmät vaatimusten määrittelyyn

Lao Can ja Balasubramaniam Ramesh julkaisivat empiirisen tutkimuksen tulokset [11] käytännön menetelmistä vaatimusten määrittelyyn kuudessatoista organisaatiossa ketterissä ohjelmistoprojekteissa. Neljässätoista näistä organisaatioista vaatimuksia ei määritelty projektin alussa. Projekti aloitettiin korkean tason vaatimuksista, joilla kehittäjille selvitettiin korkean tason visio ratkaistavasta ongelmakentästä. Iteraatioiden välillä tuoteomistaja tai asiakas tarjosi kehitystiimille tarkemman kuvauksen seuraavaksi toteutettavista käyttötapauksista ja alustavasta toteutussuunnitelmasta. Menetelmän ansiosta suhde asiakkaaseen oli laadukkaampi ja tiiviimpi ja tämän ansiosta vaatimukset kyettiin hahmottamaan selkeästi. Myöhäisen määrittelyn haasteena koettiin vaikeudet arvioida projektin kustannuksia ja kestoa, dokumentaation puute sekä

haasteet ei-toiminnallisten vaatimusten toteutumisessa [11].

Vaatimusten muutoksiin vastaaminen koetaan keskeiseksi osaksi ketteriä kehitysmenetelmiä. Kun muutostarpeisiin vastataan jatkuvan analysoinnin avulla, voidaan paremmin täyttää asiakkaan muuttuvat tarpeet. Tutkimuksessa havaittiin että syklien välillä tehtävän käyttäjäanalyysin perusteella tehtävät muutokset olivat kuitenkin käytännössä hyvin harvinaisia ja pieniä. Vaatimusten muutoksiin reagoinnin eduksi koettiin jatkuvan analysointi- ja suunnittelutyön tuomat edut käyttötapausten laadussa. Tämä osaltaan johti muutostarpeiden vähäisyyteen jo toteutetuissa toiminnallisuuksissa. Haasteena koettiin vaatimusten muutoksista ja myöhäisestä määrittelystä seuraavat puutteet jo tehdyissä arkkitehtuurivalinnoissa sekä tästä seuranneet refaktoroinnin kustannukset [11].

Ketterien menetelmien pyrkimys viedä vaatimusten priorisointi äärimmilleen niiden asiakasarvoon perustuen mahdollistaa asiakkaalle suurimman mahdollisen hyödyn mahdollisimman aikaisessa vaiheessa. Kaikki tutkimuksen organisaatiot priorisoivat vaatimuksia prosessin edetessä ja vaatimusjoukon muuttuessa. Perinteisissä malleissa priorisointi tehdään yleensä projektin alkuvaiheilla perustuen esimerkiksi ominaisuuksien riippuvuussuhteisiin ja kustannuksiin ja ensimmäisessä julkaisussa tähdätään valmiiseen tuotteeseen. Tutkimuksessa havaittiin että ketterissä malleissa prosessin aikainen priorisointi perustui yksinomaan käyttötapausten asiakasarvoon [11].

Reagointi muutokseen ketterässä ohjelmistokehityksessä

Ketterät kehitysmenetelmät yhdistetään kykyyn vastata muutokseen perinteisiä menetelmiä nopeammin prosessin aikana. Artikkelissa Agile Software Development: The Business of Innovation [12] Jim Highsmith ja Alistair Cockburn kehottaa muutosten minimoimisen sijaan keskittymään väistämättömään muutostarpeeseen nopeaan vastaamiseen. Toisessa artikkelissaan [13] Highsmith ja Cockburn viittaavat tutkimukseen 200 ohjelmistokehitysprojektista. Tutkimuksessa todettiin, että käytännössä projektien edetessä primääritavoite ei ole enää alkuperäisessä suunnitelmassa pysyminen vaan asiakkaan vaatimusten täyttäminen. Kun vaatimukset ovat muuttuneet, pyrkimys alkuperäisessä suunnitelmassa ja vaatimusmäärittelyssä pysymiseen tarkoittaisi tietoista pyrkimystä olemaan vastaamatta liiketoiminnan muuttuneisiin tarpeisiin.

Barry Boehm esittelee kirjassaan Software Engineering Economics [14] muutoksen hinta –käyrän, jossa muutoksen hinta nousee eksponentiaalisesti suhteessa sen korjaamista edeltävään aikaan. Muutostarpeiden aikaisen selvittämiseen ja muutosten nopean toimittamisen avulla voitaisiin minimoida kustannusten eksponentiaalinen kasvu [14]. Extreme Programming –menetelmässä painotetaan kuitenkin muutoksen hinnan pysyvyyttä muuttumattomana. Menetelmän mukaan muutos kannattaa toteuttaa vasta kun sitä ehdottomasti tarvitaan. Menetelmässä kehotetaan käyttämään resurssit ominaisuuksiin, jotka tuottavat nopeimmin asiakasarvoa. Näin saatava hyöty kattaa muutoksen tekemisen odottamisesta koituvat lisäkustannukset. Tämänkin mallin tarkoituksena on maksimoida muutostarpeisiin vastaaminen hyväksymällä muutosten väistämättömyys ja tekemällä päätös niihin vastaamisesta asiakkaalle helpoksi ja edulliseksi [8]. Testilähtöisessä ohjelmoinnissa ja pariohjelmoinnissa palautteen käsittely ja muutosten vieminen tuotantoon saattaa tapahtua jopa minuuteissa tai tunneissa, kun taas perinteisissä vaatimusmäärittelylähtöisissä (eng. big requirements up front, BRUF) menetelmissä käsittelyaika voi olla kuukausia [15]. Jatkuvalla sidosryhmiltä kerättävällä palautteella ja sen analysoinnilla pyritään muutostarpeeseen vastaamisen kustannusten minimoinnin suhteessa tuotteesta saatavaan hyötyyn.

Muutoksiin reagoinnin nopeuden lisäksi ketterät menetelmät pyrkivät minimoimaan muutoksista koituvat kustannukset keskittymällä toteutuksen laatuun. Lähteestä riippuen [1] virheiden korjaamiseen käytettävän ajan arvioidaan olevan 40-60% projektin työmäärästä. Tuotantoon viedyn virheen korjaamisen vaatima työmäärä saattaa olla jopa satakertainen verrattuna oikeellisen ohjelmakoodin toteuttamiseen [1]. Ketteristä menetelmistä esimerkiksi Extreme Programming –menetelmä kannustaa mahdollisimman yksinkertaiseen toteutukseen, jatkuvaan laadun parantamiseen sekä aikaiseen virheiden havaitsemiseen [8], [16]. Lyhyt vasteaika muutosten käsittelyssä ja aikainen julkaisu tuovat asiakkaalle realistisen kuvan tuotteen nykytilasta. Perinteisissä tarkasta määrittelystä lähtevissä malleissa asiakkaan on luotettava tuotteen valmistumiseen suunnitelman mukaisesti, koska tuote on nähtävillä vasta prosessin testausvaiheissa [12].

Ohjelmistokehitysprosessin mukautuminen organisaation tarpeisiin

Highsmith ja Cockburn [12] kuvailevat organisaatioita monimutkaisiksi omaksuviksi

järjestelmiksi, joissa itsenäiset yksilöt toimivat valitsemallaan tavalla mukauttaen annettuja ohjeistuksia. Ketterien menetelmien ohjeistukset ovat mukautettaviksi tarkoitettuja ohjeita, kun taas perinteisien menetelmien ohjeistukset pyrkivät kattamaan kaikki erikoistapaukset ja toimimaan parhaiten kirjaimellisesti noudatettuna. Ketterissä menetelmissä tiimi on vastuussa prosessin mukauttamisesta haasteiden edessä, kun taas perinteisissä malleissa ratkaisun voidaan odottaa löytyvän mallista [12].

Ketterät kehitysmenetelmät painottavat tiimin jäsenten välisen kommunikoinnin nopeuden, tehokkuuden ja laadun merkitystä. Extreme Programming suosittelee pariohjelmointia, Crystal ja Scrum tiivistä yhteistyötä ja yhtenäiseksi järjestettyä tiimiä ja Lean Development mahdollisimman kevyttä kommunikointia [7], [8], [12]. Onnistuneemman kommunikoinnin mahdollistamiseksi ketterissä menetelmissä pyritään pitämään tiimi yhdessä fyysisessä paikassa, korvaamaan dokumentit keskustelulla ja tussitauluilla sekä parantamaan tiimin jäsenten välistä suhdetta ja suhtautumista työhön. Toteutuakseen oikein, ketterät menetelmät vaativatkin tiimin jäseniltä ominaisuuksia, joita perinteiset mallit eivät edellyttäneet. Perinteiset menetelmät on suunniteltu standardoimaan ja yhtenäistämään työtavat ja työntekijät, kun taas ketterät mallit pyrkivät hyödyntämään yksilöiden ja yksilöllisten tiimien vahvuudet [5].

Kirjassaan Kanban: Successful Evolutionary Change for Your Technology Business [6], David Anderson kuvaa suhtautumista prosessien mukauttamiseen ketterien ohjelmistomenetelmien alkutaipaleella negatiiviseksi. Mallien taustalla olevat henkilöt eivät tarkkaan tienneet miksi mallit toimivat niin hyvin ja suosittelivat seuraamaan malleja tarkasti tai riskeeraamaan projektin onnistumisen. Nyt trendi on toinen: sekä Anderson että Highsmith ja Cockburn [5] kyseenalaistavat mahdollisuuden löytää yksi prosessi, joka olisi optimaalinen, jos ei kaikkiin, edes useampiin projekteihin. Highsmith ja Cockburn kuvaavat projektikokonaisuutta ”ekosysteemiksi”, joka koostuu hyvin erilaiset taidot ja taustat omaavista yksilöistä työskentelemässä yhdessä fyysisessä tilassa organisaation tapojen ja kulttuurin vaikutuksen alaisena. Prosessin asentamisessa ekosysteemiin on kaksi mahdollisuutta: muuntaa ekosysteemi prosessiin sopivaksi tai prosessi ekosysteemiin soveltuvaksi. Ketterät mallit suosivat näistä jälkimmäistä [5].

Anderson kehottaa kehittämään prosessia etsimällä haasteita prosessissa ja vastaamalla niihin yksitellen mukaillen Eliyahu Goldratt:n kirjassaan The Goal [17] esittelemää Theory of Constraints –teoriaa. Teoria seuraa idioomaa *ketju ei ole vahvempi kuin sen*

heikoin lenkki. Goldratt esittelee viisivaiheisen mallin, jota voidaan soveltaa prosessin kehittämiseen:

1. Tunnista haaste prosessissa
2. Suunnittele ratkaisu haasteeseen
3. Suunnittele ratkaisun käyttöönotto koko organisaatiossa
4. Suorita ratkaisun vaatimat toimenpiteet
5. Jos haaste on saatu ratkaistua, palaa vaiheeseen 1.

Kanban –menetelmässä pyritään tunnistamaan haasteet rajoittamalla työn alla olevien tehtävien määrää (eng. limiting work-in-progress) työvaiheissa. Kun tietty prosessin vaihe kuormittuu, syntyy myöhäisempiin vaiheisiin tyhjiö. Perinteisissä prosesseissa tällaisia tyhjiöitä pyritään välttämään antamalla työntekijöille vaihtoehtoista tekemistä tai siirtämällä vajaakuormalla olevia resursseja muihin tehtäviin. Andersonin mukaan prosessin optimointi kuitenkin edellyttää tyhjiön synnyn. Tyhjiön syntyyn vastataan pullonkaulana olevan ongelman ratkaisemisella. Uuden prosessin valinnan ja sen kirjaimellisen seuraamisen sijaan Anderson suositteleekin kehittämään olemassa olevaa prosessia inkrementaalisesti ja jatkuvasti organisaation tarpeisiin soveltuvaksi [6].

Tuoteomistajan vastuualueet ketterässä ohjelmistokehityksessä

Perinteisessä ohjelmistokehityksessä vastuu tuotteen markkinoille saamisesta on jaettu tuotepäällikölle, tuotemarkkinoijalle ja projektipäällikölle. Ketterissä ohjelmistokehitysprosesseissa rooli on keskitetty asiakkaan edustajalle, jota Scrum –menetelmässä kutsutaan tuoteomistajaksi ja Extreme Programmin –menetelmässä asiakkaaksi. Asiakkaan ja tuoteomistajan rooli on hyvin lähellä toisiaan. Tästä johtuen tutkielmassa käytetään termiä tuoteomistaja vastaamaan asiakkaan edustajan roolia ketterissä kehitysmenetelmissä yleisesti.

Tuoteomistajan tehtävänä on ennen kaikkea muodostaa tuotteen visio, kommunikoida se kehitystiimille ja varmistaa sen toteutuminen käytännössä. Tämän mahdollistamiseksi tuoteomistajalla on organisaation täysi tuki tarvittavien resurssien käyttöön [7]. Tässä luvussa esitellään tuoteomistajan keskeiset tehtävät ja ominaisuudet ketterässä ohjelmistokehityksessä.

Tuoteomistaja Scrum -menetelmässä

Roman Pichler kuvaa kirjassaan Agile Product Management with Scrum [7] tuoteomistajan vaatimuksia näin:

Kuvauksessa keskitytään ennen kaikkea tuoteomistajan roolissa olevan henkilön kykyyn suoriutua tietyistä tehtävistä, ei niinkään työtehtäviin tai vastuualueisiin. Pichler nostaakin tuoteomistajan henkilökohtaiset ominaisuudet keskeiseksi vaatimukseksi työn onnistumiselle suorittamiselle. Tuoteomistajan tehtävien suorittaminen Scrum –menetelmän mukaisesti koetaankin usein liian haastavaksi. Kirjallisuudessa on esitetty ratkaisuksi tehtävien jakamista useammalle henkilölle eli tuoteomistajatiimille [18]. Scrum –menetelmän kehittäjien ylläpitämän Scrum Guide:n määritelmässä kuitenkin painotetaan että tuoteomistaja on yksi henkilö, ei komitea [19]. Kun vastuu tuotteen onnistumisesta sekä teknisesti, liiketoiminnallisesti että asiakkaan tarpeiden täyttämisessä on keskitetty yhdelle henkilölle, vältytään haasteilta eri sidosryhmien tarpeiden kommunikoinnissa. Yksi henkilö hallitsee tuotteen vision ja varmistaa että vision perusteella toteutettava ratkaisu on optimaalinen eri sidosryhmien kannalta [7]. Tämä tekee roolista erittäin tärkeän, mutta myös haastavan.

Scrum Guide [19] määrittelee tuoteomistajan roolin näin:

- Tuoteomistaja ylläpitää tuotteen kehitysjonoa ja varmistaa, että se on kaikkien nähtävissä.
- Tuoteomistaja priorisoi tuotteen kehitysjonon töiden riskin, lisäarvon ja tarpeellisuuden perusteella.
- Tuoteomistaja on yksi henkilö, ei komitea. Tuoteomistajalla voi olla tukenaan komiteoita, mutta tuotteen kehitysjonon prioriteettien muuttamiseksi näiden henkilöiden tulee vakuuttaa tuoteomistaja.
- Organisaation kaikkien henkilöiden tulee kunnioittaa tuoteomistajan päätöksiä. Kenenkään ei sallita pyytää kehitystiimiä työskentelemään muilla prioriteeteilla, eikä kehitystiimien sallita kuunnella ketään, joka väittää toisin.
- Tuoteomistaja ja kehitystiimi miettivät yhdessä mitä toiminnallisuutta tuotteen kehitysjonosta valitaan kehitettäväksi seuraavaksi.

- Tuoteomistaja tarkentaa tuotteen kehitysjonon sisältöä tarvittaessa ja auttaa tiimiä löytämään kompromisseja.
- Tuoteomistaja selvittää kehitystiimin kanssa mitkä kehitysjonon kohdat on saatu valmiiksi ja mitkä ovat vielä kesken.

Tuoteomistaja omistaa tuotteen ja vastaa sen toteutuksen onnistumisesta niin teknisesti, myynnillisesti kuin asiakkaan tarpeiden täyttämisessä. Yksi henkilö ei kuitenkaan voi korvata perinteisissä malleissa liiketoiminnan, markkinoinnin, palvelun ja teknisen toteutuksen suunnittelusta vastanneita henkilöitä. Tuoteomistajan tehtävä onkin edustaa eri sidosryhmiä toisilleen, kuitenkin pyrkien välttämään haasteet ylimääräisestä kerroksesta aiheutuvassa kommunikaatiossa [7]. Vaikka vastuu koko projektin toteutumisesta on tuoteomistajalla, tuoteomistaja rooli vastaa ennen kaikkea kysymykseen *”mitä ollaan rakentamassa?”*. *”Miten rakennetaan?”* on muun kehitystiimin päätettävissä [7], [8].

Asiakas Extreme Programming –menetelmässä

Extreme Programming –menetelmässä asiakkaan edustajaa kutsutaan yksinkertaiseksi asiakkaaksi. Asiakas on osana kehitystiimiä ja vastaa tuotteen liiketoiminnallisia vaatimuksia koskevista päätöksistä. Kehittäjät ovat vastuussa oikeisiin teknisiin ratkaisuihin päätymisestä. Vastuun jakaminen ja tiivis yhteistyö kehittäjien ja asiakkaan välillä pakottaa molemmat osapuolet tiiviiseen keskusteluun. Projektin eteneminen vaatii kommunikointia toisen osapuolen kanssa [8].

Asiakkaan tehtävä on määritellä projektille tavoitteet ja ajaa projektia niitä kohti. Mitä paremmin asiakas onnistuu tehtävässään, sitä todennäköisemmin projekti onnistuu. Kuten Scrum –menetelmässä, asiakas on yksi henkilö. Tämä henkilö voi edustaa useampia sidosryhmiä, mutta vastuu ja valta projektin päätöksistä on aina keskitetty [8].

Kuten tuoteomistajan, asiakkaan tehtävänä on maksimoida projektiin käytettävistä investoinneista saatava hyöty. Tämän aikaansaamiseksi asiakas pyrkii mittaamaan projektissa etenemistä, analysoimaan sidosryhmien tarpeita jatkuvasti, reagoimaan muuttuviin tarpeisiin nopeasti sekä toteuttamaan kehitystiimin kanssa yhteistyössä vain tuotteen tärkeimmät ominaisuudet [8].

Extreme Programmingin ja Scrumin erot liittyvät lähinnä kehittäjien työtapojen

määrittelyyn, joita tässä tutkielmassa ei käsitellä. Poikkeuksena on Scrumin määrittelemät rajoitteet iteraation tai kehityssyklin aikaiselle iteraation sisällön muuttumattomuudelle. Extreme Proramming ei rajoita aloittamattomien työtehtävien muuttamista, joten asiakkaalla on mahdollisuus reagoida muutoksiin tarvittaessa Scrum –menetelmää nopeammin.

Tuotteen kokonaiskuva ja sen viestiminen kehittäjille

Tuotteen kokonaiskuva muodostuu asiakkaan tarpeista (eng. customer needs) ja tuotteen ominaisuuksista (eng. product attributes) jotka ovat kriittisiä olla olemassa asiakkaan tarpeisiin vastaamiseksi ja tuotteen onnistumiseksi. Keskeistä onkin pystyä nostamaan tietyt tarpeet ja ominaisuudet tärkeimmiksi, jotta mahdollisimman aikaisessa julkaisussa voitaisiin täyttää pienin mahdollinen joukko tuotteen onnistumisen kannalta kriittisimpiä ominaisuuksia.

Kattavan ennen projektin aloittamisesta tehtävän vaatimusten keruun ja määrittelyn sijaan ketterät ohjelmistokehitysmenetelmät suosivat prosessin aikaista asiakkaan tarpeiden analysointia. Todelliseen käyttöön ja palautteeseen perustuva jatkuva asiakasarvon kartoittaminen ja siihen perustuva tuotesuunnittelu ja ominaisuuksien priorisointi mahdollistavat asiakasarvon maksimoinnin mahdollisimman aikaisessa vaiheessa. Kattavan määrittelydokumentaation sijaan tuotteesta luodaan yksinkertainen visio, joka ei määrää palvelun varsinaisia ominaisuuksia, vaan kuvaa miten tuote täyttää asiakastarpeen ja tuottaa asiakkaalle mahdollisimman paljon arvoa. Visiolla pyritään vastaamaan kysymysiin kuka tuotetta käyttää, mihin tarpeisiin tuotteella vastataan, mikä on kriittistä näihin tarpeisiin vastaamiseksi, onko tuote mahdollista toteuttaa ja miten sen liiketoiminnallinen hyöty voidaan saavuttaa. Vision puuttuminen tai sen puutteellinen kommunikointi johtaa sidosryhmien irrallisiin ja mahdollisesti ristiriitaisiin tarpeisiin. Nämä täytetään joukolla toiminnallisuuksia, jotka eivät muodosta selkeää ja loogista kokonaisuutta [7].

Tuoteomistajan tehtävä on muodostaa visio tuotteesta sekä hallita ja ylläpitää tuotteen kokonaiskuvaa. Tuoteomistaja pitää huolen, että visio vastaa eri sidosryhmien odotuksia sekä varmistaa että se on sidosryhmille yhteinen. Koska kattavaa määrittelydokumentaatiota ei ole, jaettu visio on edellytys kehittäjien ja sidosryhmien pyrkimiseksi samaan lopputulokseen. Lopputulosta ei ole tarkkaan määritelty vaan

kehitystiimin luovuudelle jätetään tarpeeksi tilaa. Keinot lopputulokseen pääsemiseen muodostuvat kehitysprosessin edetessä [7].

Tuotteen kehitysjonon ylläpito ja priorisointi

Ketterissä kehitysmenetelmissä tuotteen kehitysjono koostuu tuotteen ominaisuuksia kuvaavista käyttötapauksista tai toiminnallisuuksista. Käyttötapaukset kuvaavat asiakkaan tai käyttäjän näkökulmaa tuotteesta tai palvelusta. Menetelmästä riippuen tuoteomistaja voi priorisoida käyttötapaukset iteraatioiden välillä tai jatkuvasti kun tehtävälistalle voidaan ottaa lisää tehtäviä. Käyttötapausten luonti ja priorisointi tehdään keräämällä ja analysoimalla jatkuvasti palautetta liiketoiminnan, loppukäyttäjien, hallinnon, markkinoinnin ja tekniikan edustajilta [12]. Palautetta analysoimalla luodaan tuote joka vastaa optimaalisesti sidosryhmien tarpeita. Puutteellinen tai virheellinen analyysi johtaa tuotteeseen, joka valmistuessaan ei sovellu käyttöön.

Tuotteen kehitysjono on yksinkertainen priorisoitu lista tehtäviä, jotka vaaditaan tuotteen toteutumiseksi käytännössä. Lista voi tarvittaessa pitää sisällään asiakkaiden tarpeiden kartoittamisen ja sekä toiminnalliset että ei-toiminnalliset vaatimukset tuotteelle [7]. Tuoteomistajan keskeisin tehtävä on ylläpitää kehitysjonoa ja saattaa se kaikkien nähtäville [19]. Ketterässä kehityksessä tuotteen kehitysjono korvaa perinteisen vaatimusmäärittelydokumentin. Kehitysjonon kärkipää kuvaa tuotteen tärkeimpiä tehtäviä asiakasarvon ja tuotteen teknisen toteutumisen kannalta.

Tuoteomistajuus kolmessa ohjelmistoprojektissa

Projekti ”A”

Projektin kuvaus

Projektissa toteutettiin Suomen mittakaavassa suuren kuluttajille suunnatun verkkopalvelun uusi versio. Kävijämäärältään palvelu oli julkaisun aikaan TNS-Gallupin Web-sivustojen viikkoluvuissa [20] 25 suurimman palvelun joukossa.

Projektissa seurattiin Scrum –ohjelmistokehitysprosessia. Kehitystiimissä oli yksi tuoteomistaja, tiiminvetäjä ja yhdeksän kehittäjää. Projektiin osallistui lisäksi käytettävyyssuunnittelutoimisto, joka toteutti palvelun konseptin ja graafisen ulkoasun. Iteraation pituus oli neljä viikkoa ja projekti toteutettiin kuudessa iteraatiossa. Iteraatiot

aloitettiin iteraation suunnittelupalaverilla, jossa valittiin seuraavassa iteraatiossa toteutettavat käyttötapaukset. Iteraation etenemistä seurattiin päivittäisissä 15 minuutin tapaamisissa, joissa kehittäjät raportoivat keskeneräisistä töistään. Iteraation päätteeksi iteraation tulokset käytiin läpi ja siirryttiin suunnittelemaan seuraavaa iteraatiota. Projektin kesto oli noin kuusi kuukautta.

Tuoteomistajan rooli projektissa

Tuoteomistajan roolissa ollut henkilö oli tekninen projektipäällikkö, joka vastasi tuotteen kehitysjonon ylläpidosta ja saattamisesta kehittäjien nähtäväksi. Tuotteen varsinaista omistajuutta edusti käytettävyyssuunnittelutoimiston pääsuunnittelija sekä tuotteen varsinainen omistaja eli yrityksen toimitusjohtaja. Tuoteomistajan tehtävänä oli välittää suunnittelijoiden ja muiden sidosryhmien toiveet kehitystiimille. Kehityksen edetessä kehittäjät hakivat konseptiin liittyvät päätökset suoraan käytettävyyssuunnittelijoilta. Ristiriidat tuotteen varsinaisen omistajan ja käytettävyyssuunnittelijoiden näkemyksissä näkyivät aika ajoin valmiissa tuotteessa.

Tuotteen kokonaiskuva oli kehitystiimille melko hyvin selvillä projektin edetessä. Käytettävyyssuunnittelijat viestivät kokonaiskuvan muutoksista ja visiostaan kehittäjille jatkuvasti pidetyissä konseptipalavereissa. Koska tuoteomistaja ei ollut vastuussa konseptista, hän ei aina pystynyt omatoimisesti vastaamaan kehittäjien kysymyksiin. Vaikka tuoteomistaja oli jatkuvasti läsnä, ei tarkennuksia ja kompromisseja käyttötapauksiin saatu tehtyä. Palvelun konseptisuunnitelmien viivästyessä, kehittäjät joutuivat odottamaan tarkennuksia käyttötapauksiin tai päättelemään ratkaisut omatoimisesti.

Tuotteen kehitysjono oli jatkuvasti koko tiimin saatavilla. Kehitysjono ei kuvannut koko tuotetta vaan sitä täydennettiin tarvittaessa ennen iteraatioiden aloittamista. Epic-käyttötapauksia ei ollut määritelty vaan kehitysjono koostui toteutettavista työtehtävistä. Tuote oli määritelty kattavasti ennen toteutuksen aloitusta ja muutoksia määrittelyyn ei projektin edetessä juurikaan tullut. Muutokset olivat lähinnä tarkennuksia käyttöliittymäkuviin ja interaktioihin.

Kehitysjono oli priorisoitu kehitystiimin kanssa yhteistyössä lähinnä teknisten riippuvuuksien perusteella. Projektissa tähdättiin julkaisuun vasta kun koko tuotteen kehitysjono on valmis, joten priorisointia käyttötapausten asiakasarvon perusteella ei

juurikaan tehty. Julkaisun lähestyessä jouduttiin joistain toiminnallisuuksista luopumaan ja aiempien vaiheiden puutteellinen priorisointi aiheutti julkaisun viivästymisen keskeisimpien käyttötapausten ollessa yhä kesken.

Projektissa edettiin Scrum –menetelmän mukaisissa iteraatioissa, joiden sisältö suunniteltiin tuoteomistajan ja kehittäjien yhteistyössä poimimalla käyttötapauksia tuotteen kehitysjonosta. Kehitystiimi pystyi ehdottamaan iteraatioon valittavat käyttötapaukset teknisen toteutuksen vaiheen perusteella. Iteraation sisältö säilyi muuttumattomana toteutusvaiheessa, mutta epätarkat hyväksymiskriteerit ja käyttötapaukset tarkentuivat jatkuvasti toteutuksen aikana. Iteraation tulokset käytiin läpi iteraation päätteeksi, mutta koska käyttötapausten hyväksymiskriteerejä ei oltu määritelty, tehtäviä jouduttiin usein viimeistelemään myöhemmissä iteraatioissa. Iteraatiot koostuivat epäyhtenäisestä joukosta kehittäjien valitsemia ominaisuuksia. Iteraatioilla ei ollut varsinaisia tavoitteita vaan projektissa tähdättiin vasta koko tuotteen valmistumiseen.

Projekti ”B”

Projektin kuvaus

Projektissa toteutettiin prototyyppi kuluttajille suunnatusta tablet-käyttöisestä sovelluksesta projektityönä Helsingin Yliopiston Software Factory –kurssilla.

Projektissa seurattiin iteratiivista ohjelmistokehitysprosessia, jossa oli piirteitä Scrum ja Kanban –menetelmistä. Tiimissä oli kolmen tuoteomistajan tiimi ja kahdeksan kehittäjää. Kehittäjät toimivat tarvittaessa käytettävyyssuunnittelijoina, graafikoina ja tiimivetäjinä. Projektin kesto oli seitsemän viikkoa ja projekti oli jaettu seitsemään yhden viikon iteraatioon. Iteraatiot aloitettiin suunnittelupalaverilla, jossa määriteltiin iteraatiossa toteutettavat käyttötapaukset. Kehittäjät ylläpitivät tehtävien statustietoa yhteisellä taululla. Iteraation kulkua seurattiin päivittäin pidettävällä 15 minuutin tapaamisella. Iteraation jälkeen iteraation tulokset esiteltiin tuoteomistajille.

Tuoteomistajan rooli projektissa

Projektissa asiakkaan roolissa oli tuoteomistajatiimi, joka koostui kolmesta kehittäjien kannalta tasa-arvoisesta henkilöstä. Tuoteomistajatiimin jäsenillä oli keskinäiset roolit: liiketoimintavastaava, tuotevastaava ja asiakassuhdevastaava. Tuotteeseen kohdistuvat

muutostarpeet tulivat kuitenkin tuoteomistajatiimin jäseniltä suoraan kehittäjille ja olivat usein ristiriitaisia. Tuoteomistajatiimin keskinäisissä ristiriitatilanteissa oli epäselvää minkä tuoteomistajan mielipiteitä tai päätöksiä pitäisi noudattaa.

Projektin alkaessa tuoteomistajilla oli korkean tason visio palvelusta ja tuote oli tarkoitus määritellä projektin edetessä. Visiota ei saatu viestittyä kehittäjille ja kokonaiskuva tuotteesta puuttui sekä tuoteomistajilta että kehittäjiltä koko projektin ajan. Tämä johti siihen, että projektilla ei ollut varsinaisia tavoitteita, joissa etenemistä olisi voitu seurata. Selkeän päämäärän puuttuminen johti myös osaltaan pyrkimyksiin virheellisiin tavoitteisiin.

Projektissa edettiin viikon kestoisissa iteraatioissa, joiden sisältö suunniteltiin kehittäjien ja tuoteomistajien yhteistyöllä. Iteraatioilla ei ollut varsinaisia tavoitteita, mutta iteraatioissa pyrittiin tuottamaan aina jokin toiminnallinen lisäys tuotteeseen. Ennen iteraation aloittamista tuoteomistajat määrittelivät käyttötapaukset seuraavaa iteraatiota varten yhteistyössä kehittäjien kanssa. Vain pieni osa kehittäjistä osallistui käyttötapausten ja iteraation sisällön suunnitteluun. Kehittäjät määrittelivät käyttötapauksille hyväksymiskriteerit, mutta kriteerien mukaan toteutetut toiminnallisuudet eivät vastanneet tuoteomistajan odotuksia. Hyväksymiskriteerit olivat epäselviä ja muuttuivat toteutuksen aikana. Iteraation tulokset käytiin läpi iteraation päätteeksi, mutta yksittäisiä tehtäviä ei hyväksytty tai hylätty hyväksymiskriteerien perusteella.

Tuoteomistajat olivat aktiivisesti saatavilla koko projektin ajan. Kehitystiimi ei kuitenkaan saanut tuoteomistajilta tarvittavia tarkennuksia työtehtäviin, koska tuotteen kokonaiskuvaa ei pystytty määrittelemään. Tuotteella ei ollut varsinaista kehitysjonoa eikä sen epic-käyttötapauksia saatu määriteltyä. Koska kehitysjonoa ei ollut olemassa, ei käyttötapausten priorisointia voitu kunnolla tehdä. Epic-käyttötapauksia yritettiin määritellä ja priorisoida ennen iteraation aloittamista siltä osin, kuin seuraavassa iteraatiossa voitaisiin toteuttaa. Laajemmalla tuotteen vision kriittisimpien ominaisuuksien priorisoinnilla pyrittiin määrittelemään minimitaso ominaisuuksia, jotka voitaisiin toteuttaa projektissa. Tämä ominaisuusjoukko kuitenkin muuttui projektin aikana jatkuvasti.

Projekti ”C”

Projektin kuvaus

Projektissa toteutettiin uusi kuluttajille suunnattu verkkopalvelu yli 1000 työntekijän organisaatiossa.

Projektissa oli käytössä iteratiivinen, viikon mittaisiin kehityssykleihin jaettu hyvin kevyesti määritelty ohjelmistokehitysmenetelmä. Kehitystiimiin kuului tuoteomistajan lisäksi tiiminvetäjä ja 13 kehittäjää. Kehitystiimin lisäksi toteutuksesta vastasi käytettävyyssuunnittelutoimiston suunnittelija sekä graafikko. Projektin kesto oli kaksi ja puoli kuukautta.

Tiimin suuren koon vuoksi päivittäisiä status-tapaamisia ei pidetty vaan tiiminvetäjä seurasi töiden etenemistä tiettyjen toiminnallisuuksien kehittämisestä vastaavien kehittäjien kanssa henkilökohtaisesti. Ainoa muodollinen tapaaminen oli viikoittainen läpikäynti, jossa kehittäjät esittelivät parannukset palvelussa tuoteomistajille, toisille kehittäjille sekä muille sidosryhmille. Ennen seuraavan iteraation aloittamista, tiiminvetäjä keskusteli tuoteomistajan kanssa seuraavassa iteraatiossa toteutettavista käyttötapauksista.

Tuoteomistajan rooli projektissa

Tuoteomistajan roolissa oli projektiin nimetty tuotepäällikkö. Tuoteomistajan päätösvaltaa käytti tuoteomistajan lisäksi projektin käytettävyyssuunnittelijat, jotka kommunikoivat muutoksista tuoteomistajan ohitse. Kehittäjille oli välillä epäselvää keneltä tarkennukset käyttötapauksiin ja päätökset muutoksista pitäisi kysyä ja vastaanottaa.

Palvelun visio ja projektin kokonaiskuva viestittiin kattavasti kehittäjille projektin alkaessa sekä kerran projektin aikana. Projektin tultua päätökseen todettiin kokonaiskuvan säilyneen lähes muuttumattomana koko projektin ajan. Palvelu oli määritelty tarkkaan ennen toteutuksen aloittamista käyttöliittymäkuvien muodossa. Toteutusprosessin aikana tulleet muutostarpeet olivat lähinnä pieniä tarkennuksia käyttötapauksissa ja interaktioissa. Tieto pienemmistä muutoksista ei kuitenkaan päätynyt kehittäjille ilman kehittäjien aktiivisuutta. Käytettävyyssuunnittelijat olivat kerran viikossa paikalla vastaamassa kehittäjien konseptia koskeviin kysymyksiin ja tarkennuksiin toteutuksessa.

Projektissa edettiin viikon pituisissa kehityssykleissä. Syklit tähtäsivät aina selkeästi tietyn toiminnallisuuden julkaisuun alpha- tai beta-käyttäjille. Iteraatioiden tavoitteissa pysyttiin hyvin ja iteraation tulokset ja tehtävien valmistuminen käytiin läpi iteraatioiden päätteeksi.

Tuoteomistaja oli paikalla lähes aina ja pystyi nopeasti tekemään tarkennuksia ja kompromisseja toteutettaviin käyttötapauksiin. Tarvittaessa tarkennukset haettiin suoraan käytettävyyssuunnittelijoilta. Ristiriitoja tuoteomistajan ja käytettävyyssuunnittelijoiden näkemyksissä oli harvoin. Vaikka tuoteomistaja oli hyvin saatavilla, ei hän osallistunut käyttötapausten toteutustavan suunnitteluun.

Tuotteen kehitysjonoon oli koottu tuotetta kuvaavat epic-käyttötapaukset. Kehitysjono oli kuitenkin koko projektin ajan puutteellinen ja toteutettavat käyttötapaukset pääteltiin ennen toteutuksen aloittamista käyttöliittymäkuvista usein kehittäjien toimesta. Tarkennukset saatiin tarvittaessa nopeasti, mutta kehitysjonon puuttuminen vaikeutti projektin valmiusasteen arviointia projektin edetessä. Varsinaisia hyväksymiskriteerejä ei oltu määritelty. Kehitysjono oli esillä vain digitaalisesti eikä se ollut kehittäjien aktiivisessa käytössä.

Ennen toteutuksen aloittamista käytettävyyssuunnittelijat olivat määritelleen palvelusta julkaisuun riittävän minimitason, joka oli kuvattu käyttöliittymäkuvin. Varsinaista priorisointia valittujen käyttötapausten välillä ei tehty, sillä projektissa tähdättiin valmiin palvelun julkaisuun.

Havainnot tuoteomistajien tehtävien merkityksestä

Tässä luvussa esitellään tapaustutkimuksessa esitellyissä projekteissa esiintyneet tuoteomistajan tärkeimmät tehtävät, haasteet niiden toteutumisessa sekä tehtävien laiminlyönnin seuraukset. Tiettyjen tehtävien nosto tuoteomistajan tärkeimmiksi tehtäviksi perustuvat henkilökohtaisiin kokemuksiini edellä kuvatuissa projekteissa. Valitut tehtävät ovat tapaustutkimuksessa kerättyjen havaintojen perusteella muokattu joukko tutkielman toisessa luvussa esiteltyjä kirjallisuudesta kerättyjä tuoteomistajan tehtäviä.

Vain yksi päätösvaltainen tuoteomistaja

Tuotteella on yksi tuoteomistaja eikä komitea. Tuotteen kehitysjonon prioriteettien muuttamiseksi muiden tulee vakuuttaa tuoteomistaja.

Kun tuoteomistajia oli useita tai oli epäselvää kuka on varsinainen tuoteomistaja, kehittäjät eivät tienneet kenen vastuulla on tehdä tarkennuksia ja muutoksia käyttötapausten toteutustapaan. Useamman päätösvaltaisen tuoteomistajan ristiriitaiset näkemykset aiheuttivat ylimääräistä työtä ja virheellisiä toteutuksia. Kun yhtä ristiriidatonta päätöstä tai näkemystä ei ollut kehittäjien saatavilla, tuotteen kehitys pysähtyi.

Tunnusmerkit

- Projektin alkaessa tai sen aikana jollekin on epäselvää kuka on varsinainen tuoteomistaja
- Kehittäjät ottavat vastaan muutoksia ja tehtäviä usealta eri henkilöltä
- Kehitettävä tuote on monimutkainen tai epäjohdonmukainen
- Tuoteomistaja ei saa aikaan tarvittavia muutoksia

Ratkaisukeinoja

- Tuoteomistaja nimetään selkeästi. Kehitysjonon muutosten aikaansaamiseksi muiden on vakuutettava tuoteomistaja
- Tuoteomistajalla on organisaation tuki ja valta tehdä päätöksiä vapaasti
- Tiimille ei saa antaa tehtäviä tuoteomistajan ohi

Tuotteen kokonaiskuvan jatkuva kommunikointi

Tuoteomistajalla on jatkuvasti selkeä kokonaiskuva tuotteesta ja se viestitään tiimille aktiivisesti.

Tuotteen selkeän vision puuttuminen aiheutti jatkuvasti pyrkimystä ristiriitaisiin ja virheellisiin tavoitteisiin. Vaikka tavoitteet eivät olisi muuttuneet, kehittäjät olisi kannattanut pitää ajan tasalla projektin nykytilasta. Jos muiden kehittäjien tuotoksista ei viestitä aktiivisesti, on vaarana etteivät kehittäjät tiedä missä vaiheessa projekti ollaan menossa. Ilman tuotteen kokonaiskuvan jatkuvaa kommunikointia, projektin tavoitteet koettiin epäselviksi. Mikäli visio ei ole selkeä edes tuoteomistajalle ja projektin

tavoitteet ovat epäselviä, projektin eteneminen kohti tavoitteita on mahdotonta.

Tunnusmerkit

- Tuoteomistaja ei pysty kuvaamaan korkean tason epic-käyttötapauksia tuotteelle
- Kehittäjille on epäselvää mitä ollaan tekemässä ja minkä ongelman tuote ratkaisee
- Kehitettävä tuote on monimutkainen tai epäjohdonmukainen
- Julkaisujen jälkeen jotkut sidosryhmät eivät ole tyytyväisiä tuotteeseen
- Kehitystiimille ei ole selkeää missä vaiheessa projektia ollaan menossa

Ratkaisukeinoja

- Kaikkien sidosryhminen kartoittaminen, palvelusuunnittelu ja epic-käyttötapausten kirjoittaminen
- Laajan tai monimutkaisen tuotteen pilkkominen useammaksi projektiksi
- Vision viestiminen tiimille jatkuvasti

Iteraatioiden selkeät tavoitteet

Iteraatiolle on määritelty selkeät tavoitteet, jotka voidaan saavuttaa ja joissa etenemistä voidaan mitata.

Iteraation tavoitteiden puuttuminen teki iteraation aikaisen etenemisen seurannan ja iteraation onnistumisen arvioinnin vaikeaksi sekä tiimille että tuoteomistajalle. Kun iteraatio ei muodostanut selkeää kokonaisuutta, ei iteraatioissa saatu toteutettua johdonmukaisia julkaisuja.

Tunnusmerkit

- Kehitystiimin jäsenet eivät ole selvillä iteraation tavoitteista
- Kehitystiimin jäsenet eivät osaa arvioida tullaanko iteraatiossa
- Iteraation jälkeen on epäselvää onnistuttiinko tavoitteissa
- Iteraatiosta ei seuraa tietyn teeman tai kokonaisuuden julkaisu

Ratkaisukeinoja

- Iteraation käyttötapausten selkeä rajaus ja yhteisen teeman määrittely iteraation alussa
- Julkaisun tai iteraation tavoitteiden selkeä määrittely iteraation alussa

Tuoteomistajan aktiivinen osallistuminen tiimin toimintaan

Tuoteomistaja on jatkuvasti läsnä ja mukana tiimin toiminnassa. Tuoteomistaja tarkentaa tuotteen kehitysjonon sisältöä tarvittaessa ja auttaa tiimiä löytämään kompromisseja käyttötapausten toteuttamisessa.

Kun tuoteomistaja ei ollut kehittäjien saatavilla, vastausten saanti kysymyksiin hidastui huomattavasti. Kehittäjät joutuivat tekemään tuoteomistajan vastuulla olevia päätöksiä itsenäisesti ja ottamaan vastaan muutoksia tuoteomistajan ohitse. Tämän seurauksena käyttötapausten toteutus erosi tuoteomistajan näkemyksestä.

Tuoteomistajan aktiivinen läsnäolo olisi nopeuttanut projektin etenemistä, tehostanut kehitystiimin työtä ja varmistanut ettei kehitystiimi etene virheelliseen suuntaan. Kun kehittäjät eivät saaneet käyttötapauksiin tarvittavia tarkennuksia nopeasti tai kun toteutuksen vaatimia kompromisseja käyttötapauksissa ei voitu tehdä, työskentely hidastui eikä valittu ratkaisu usein ollut teknisesti optimaalinen.

Tunnusmerkit

- Käyttötapausten toteuttaminen on haastavaa
- Toteuttajat kokevat valitut ratkaisut epäoptimaalisiksi
- Ongelmia tai ristiriitoja käyttötapauksissa ei saada ratkaistua
- Tuoteomistaja ei ole läsnä eikä osallistu tiimin päivittäiseen toimintaan
- Tuoteomistaja ei tunnu kehitystiimin jäseneltä

Ratkaisukeinoja

- Tuoteomistajan tulee osallistua päivittäiseen toimintaan henkilökohtaisesti
- Tuoteomistajalla on hyvä tekninen osaaminen toteutettavasta palvelusta
- Tuoteomistaja on valmis tekemään kompromisseja käyttötapausten toteutuksesta
- Tuoteomistaja on päivittäin henkilökohtaisesti kehittäjien saatavilla ja tarkentaa

käyttötapauksia tarvittaessa

Kattava kehitysjono kaikkien nähtävissä

Tuoteomistaja ylläpitää koko tuotetta kuvaavaa kattavaa kehitysjonoa ja varmistaa, että se on kaikkien nähtävissä.

Tuotteen kehitysjonoa lukemalla pitäisi saada selkeä kuva koko tuotteesta. Kun kehitysjono ei ollut tiimin nähtävillä tai se ei ollut ajan tasalla, kehittäjät alkoivat päätellä käyttötapauksia itse muun materiaalin kuten käyttöliittymäkuvien perusteella. Tällä tavoin päädyttiin usein tekemään matalan prioriteetin tehtäviä tai virheellisiä toteutuksia ja ylimääräistä työtä.

Tunnusmerkit

- Kehitysjonoa ei päivitetä tai se on puutteellinen
- Kehittäjät tekevät töitä joita ei ole määritelty käyttötapauksina
- Kehitysjono ei ole kehittäjien aktiivisessa käytössä

Ratkaisukeinoja

- Ajan tasalla oleva kattava kehitysjono on kaikkien nähtävillä analogisessa muodossa
- Iteraatioihin otetaan töitä ainoastaan tuotteen kehitysjonon kärkipäästä

Asiakasarvon perusteella priorisoitu kehitysjono

Tuoteomistaja priorisoi tuotteen kehitysjonoa jatkuvasti käyttötapausten asiakasarvon perusteella. Tärkeimmät käyttötapaukset toteutetaan seuraavaksi.

Ketterissä ohjelmistokehitysmenetelmissä on keskeistä pyrkimys tuottaa asiakkaalle lisäarvoa mahdollisimman nopeasti. Tätä tavoitetta ei voida saavuttaa ilman oikein priorisoitua kehitysjonoa. Ilman asiakasarvoon perustuvaa priorisointia päädyttiin jatkuvasti toteuttamaan käyttötapauksia, jotka eivät olleet kriittisiä tuotteen julkaisun kannalta. Tästä seurasi, että tuotteen ensimmäisen version julkaisu viivästyi.

Tunnusmerkit

- Tuotteen kehitysjono ja käyttötapausten prioriteetit pysyvät muuttumattomana

projektin edetessä

- Tuotteen kehitysjono on kehittäjien priorisoima
- Tuoteomistaja ei osaa perustella käyttötapausten keskinäistä tärkeysjärjestystä
- Kehitysjonon kärkipäästä puuttuu tärkeitä käyttötapauksia
- Iteraatiot eivät johda julkaisuihin

Ratkaisukeinoja

- Käyttötapausten arvon analysointi ja kehitysjonon jatkuva priorisointi yhdessä kaikkien sidosryhminen kanssa
- Tuotteen parannusten julkaisu iteraatioiden päätteeksi

Toteutuksen ja iteraation sisällön suunnittelu yhdessä

Tuoteomistaja ja kehitystiimi miettivät yhdessä mitä toiminnallisuutta tuotteen kehitysjonosta valitaan kehitettäväksi. Tuoteomistaja pilkkoo kehitystiimin kanssa epic-käyttötapaukset ennen iteraation aloittamista käyttötapauksiksi, jotka on mahdollista toteuttaa yhdessä iteraatiossa.

Kun tuoteomistaja ja kehittäjät eivät pilkkoneet epic-käyttötapauksia toteutettaviksi käyttötapauksiksi yhdessä, siirtyivät kysymykset käyttötapausten toteutuksesta toteutusvaiheeseen. Kehittäjät joutuivat tulkitsemaan epic-käyttötapauksia tai käyttötapauksia parhaan kykynsä mukaan ilman tuoteomistajan reaaliaikaista palautetta. Toteutuksen suunnittelussa tapahtuneet virheet aiheuttivat ylimääräistä työtä ja virheellisiä toteutuksia.

Kehittäjillä on usein kattavin kuva tuotteen teknisestä nykytilasta. Kun kehittäjät eivät voineet osallistua toteutettavien käyttötapausten suunnitteluun tai vaikuttaa iteraation sisältöön, työt jäivät usein keskeneräisiksi ja valittujen tehtävien toteuttaminen vaikeutui. Tästä seurasi että kehittäjät joutuivat jatkuvasti tekemään töitä joita ei oltu määritelty osaksi iteraation suunnitelmaa.

Tunnusmerkit

- Tuoteomistaja ei ole paikalla kun epic-käyttötapausten tai käyttötapausten toteutusta suunnitellaan
- Kehittäjät kokevat käyttötapausten toteuttamisen haasteelliseksi

- Kehittäjillä on jatkuvasti töitä iteraation suunnitelman ulkopuolelta

Ratkaisukeinoja

- Kehittäjät voivat osallistua käyttötapausten määrittelyyn
- Kehittäjät voivat osallistua käyttötapausten priorisointiin
- Käyttötapausten alustava toteutus suunnitellaan iteraation aloituspalaverissa tuoteomistajan läsnäollessa

Iteraatiossa toteutettavien käyttötapausten hyväksymiskriteerit

Tuoteomistaja määrittelee yhteistyössä tiimin kanssa toteutettaville käyttötapauksille hyväksymiskriteerit.

Kun käyttötapausten hyväksymiskriteerejä ei oltu määritelty tai ne oli määritelty epäselvästi, kehittäjien ja tuoteomistajan kuva käyttötapauksen valmistumisesta oli usein ristiriidassa. Tämä teki iteraatiossa etenemisen arvioinnin mahdottomaksi ja johti usein keskeneräiseksi jääviin toteutuksiin ja toiminnallisuuksiin. Hyväksymiskriteerit tulisi määritellä niin kattavasti, että kun kriteerit täyttyvät, toiminnallisuus on valmis.

Tunnusmerkit

- Valmis toiminnallisuus on tuoteomistajan tai jonkin sidosryhmän mielestä puutteellinen
- Kehittäjä ei osaa sanoa onko tehtävä valmis

Ratkaisukeinoja

- Kattavat hyväksymiskriteerit ovat edellytys käyttötapauksen toteutuksen aloittamiselle
- Hyväksymiskriteerit ovat kaikkien nähtävillä ja ne kuvaavat valmista toiminnallisuutta

Tehtävien säilyminen muuttumattomina iteraation aikana

Toteutettavat käyttötapaukset ja niiden hyväksymiskriteerit ovat tarkkoja ja pysyvät muuttumattomina iteraation aikana. Lisää tehtäviä ei anneta kesken iteraation.

Tehtävänannon muuttuminen toteutuksen aikana häiritsi keskittymistä, hidasti

toteutusta ja aiheutti kehittäjille ylimääräistä työtä. Kun kattavasti määriteltyjä käyttötapauksia ei ollut tuotteen kehitysjonossa, kehittäjät hyväksyivät toteutukseen puutteellisesti määriteltyjä työtehtäviä. Toteutuksen muuttuminen kehityksen aikana aiheutti kehittäjissä turhautumista. Keskeneräisesti määriteltyjen ja muuttuvien käyttötapausten toteutuksen sijaan pullonkaula prosessissa olisi pitänyt tunnistaa ja kehittää käyttötapausten ja hyväksymiskriteerien määrittelyä.

Tunnusmerkit

- Hyväksymiskriteerit ovat epäselviä ja tarkentuvat toteutuksen aikana
- Työt eivät valmistu suunnitellussa aikataulussa
- Tuoteomistajan ja kehittäjän näkemys tehtävän hyväksymiskriteereistä on erilainen
- Tuoteomistaja ei ole tyytyväinen toteutettuihin toiminnallisuuksiin

Ratkaisukeinoja

- Kehitysjonossa on oltava tarpeeksi tarkkaan määriteltyjä, korkean prioriteetin käyttötapauksia hyväksymiskriteereineen
- Hyväksymiskriteereiden on oltava yksiselitteisiä, selkeitä ja pysyttävä muuttumattomina toteutuksen ajan
- Toteutuksessa olevat tehtävät on rauhoitettava toteutuksen ajaksi

Töiden hyväksyntä iteraation päätteeksi

Tuoteomistaja hyväksyy tai hylkää toteutetut työt iteraation päätteeksi.

Tekemätön tai puutteellisesti toteutettu työ täytyy priorisoida uudelleen ja siirtää takaisin tuotteen kehitysjonoon. Kun tuoteomistaja ei hyväksynyt tehtyjä töitä ja keskeneräisiä töitä siirretty takaisin kehitysjonoon, jäi töiden valmiiksi saattaminen kehittäjien vastuulle. Kun tuoteomistaja ei ollut selvillä töiden valmiusasteesta, jäi lopulliseen tuotteeseen paljon keskeneräisiä toiminnallisuuksia.

Tunnusmerkit

- Valmiit toiminnallisuudet eivät toimi kunnolla
- Palvelu ei toimi kunnolla

- Tehtyjä töitä ei hyväksytä ja merkitä valmiiksi
- Kehittäjillä on usein töitä annettujen tehtävien ulkopuolelta

Ratkaisukeinot

- Hyväksymiskriteerien tarkka määrittely
- Hyväksymiskriteerien täyttyminen täytyy varmistaa ennen työn merkitsemistä valmiiksi
- Tehtävien viimeistelytyöt kirjataan uusiksi käyttötapauksiksi ja priorisoidaan uudelleen tuotteen kehitysjonoon

Tuoteomistajan tehtävien priorisointi projektikohtaisesti

Yhteenveto

Lähteet

[1] R. N. Charette, “Why Software Fails,” *Ieee Spectrum*, vol. 42, no. 2005, pp. 42-49, 2005.

[2] Software Engineering Institute, “CMMI® for Development, Version 1.3,” 2010.

[3] Software Engineering Institute, “Published CMMI Appraisal Results,” 2011. [Online]. Available: http://sas.sei.cmu.edu/pars/pars.aspx. [Accessed: 11-Jan-2012].

[4] M. Winter and T. Szczepanek, “Projects and programmes as value creation processes: A new perspective and some practical implications,” *International Journal of Project Management*, vol. 26, no. 1, pp. 95-103, 2008.

[5] A. Cockburn and J. Highsmith, “Agile Software Development: The People Factor,” *Computer*, no. 2001, pp. 131-133, 2001.

[6] D. J. Anderson, *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press, 2010, p. 278.

[7] R. Pichler, *Agile Product Management with Scrum: Creating Products That Customers Love*. Addison Wesley (pear, 2010, p. 133.

[8] *Extreme Programming Pocket Guide [Paperback]*. O’Reilly Media; 1 edition, 2003, p. 80.

[9] “Ketterän ohjelmistokehityksen julistus.” [Online]. Available: http://agilemanifesto.org/iso/fi/. [Accessed: 11-Jan-2012].

[10] *Innovator’s Guide to Growth: Putting Disruptive Innovation to Work*. Harvard Business School Press, 2008, p. 320.

[11] L. Cao and B. Ramesh, “Agile Requirements Engineering Practices: An Empirical Study,” *Engineering*.

[12] J. Highsmith and A. Cockburn, “Agile software development: the business of innovation,” *Computer*, vol. 34, no. 9, pp. 120-127, 2001.

[13] J. Highsmith, C. Consortium, and A. Cockburn, “Agile Software Development: The Business of Innovation,” *Science*.

[14] B. W. Boehm, “Software Engineering Economics,” Oct. 1981.

[15] S. W. Ambler, “Why Agile Software Development Techniques Work: Improved Feedback,” *Best practices for Software Development*, 2006. [Online]. Available: http://www.ambysoft.com/essays/whyAgileWorksFeedback.html.

[16] K. Beck, “Embracing Change with Extreme Programming,” *Computer*, no. 1999, pp. 70-77, 1999.

[17] E. M. Goldratt, *The goal: a process of ongoing improvement*. [Croton-on-Hudson, NY]: North River Press, 1986.

[18] A. de S. Croix and A. Easton, *The Product Owner Team*. IEEE, 2008, pp. 274-279.

[19] K. Schwaber and J. Sutherland, *Scrum Guide - Fi*. 2009.

[20] “TNS | Suomen web-sivustojen viikkoluvut.” [Online]. Available: http://tnsmetrix.tns-gallup.fi/public/. [Accessed: 08-Feb-2012].