Muutokset ohjelmistokehityksen dokumentointiin yrityksen siirtyessä käyttämään ketteriä ohjelmistonkehitysmenetelmiä

Oulun yliopisto
Tietojenkäsittelytieteiden laitos
Pro gradu - tutkielma
Juha Haapajärvi
2.12.2012

Tiivistelmä

Ketterät ohjelmistonkehitysmenetelmät ovat tämän vuosituhannen puheenaihe ohjelmistojen kehityksen alueella. Monet suuret ohjelmistoalan yritykset tutkivat menetelmien käyttöönottoa. Ketterät ohjelmiston sisältävät paljon uusia elementtejä varsinkin organisatorisen tiedon hallintaan. Ketterissä ohjelmistonkehitysmenetelmissä tiedonkulkua on yritetty parantaa suullisen kommunikoinnin avulla. XP-ohjelmistonkehitysmenetelmässä on käytössä pariohjelmointi, Scrum-ohjelmistonkehitysmenetelmässä on käytössä päiväpalaverit ja jaettu työskentelytila. Myös ketterien ohjelmistonkehitysmenetelmien iteraatiosyklit parantavat kommunikaatiota.

Vaikka ketterissä ohjelmistonkehitysmenetelmissä on pohjimmiltaan ollut alusta saakka paljon tiedon johtamisen elementtejä, käytännössä on menetelmien käyttö keskittynyt vielä ratkaisemaan teknisiä ongelmia ja tiedon johtamisen näkökulmat ovat olleet hyvin toissijaisia. Käytännön tekijät ketterissä ohjelmistotiimeissä ovatkin yleensä pohjakoulutukseltaan suuntautuneet enemmän tekniikkaan kuin johtamiseen tai tiedon johtamiseen. Tutkimuksen mielenkiinto onkin, kuinka ketterät menetelmät käytännössä toteuttavat niihin liitettyjä tiedon hallinnan elementtejä, ja kuinka vuosia perinteiseen dokumentointiin luottaneet yritykset ovat pystyneet muuttamaan toimintatapojaan ketterien menetelmien aatteita hyödyntäviksi

Tässä tutkimuksessa tutkittiin ketterän ohjelmistonkehityksen räätälöityjä prosesseja. Tutkimuksessa paneuduttiin yksinkertaisilla haastatteluilla yritysten kehittämiin ketteriin ohjelmistonkehitysprosesseihin ja siihen, mitenkä prosessit huomioivat ketterien ohjelmiston kehitysmenetelmien taustalla olevat tiedon hallinnan teoriat.

Oliko ketteriin ohjelmistonkehitysmenetelmiin siirtyminen muuttanut tämän tutkimuksen kohteena olleiden yritysten dokumentointi käytäntöjä? Haastattelut kertoivat aika selvästi, että tieto projekteissa välittyy yhä dokumenttien avulla. Yllättävää ei ollut, ettei ketteriin menetelmiin siirtyminen ole oikeastaan tuonut mitään muutosta dokumentoinnin tarpeeseen. Jopa kasvua erilaisten raporttien muodossa ilmeni. Kaksi kolmesta haastatelluista perusteli, että virtuaalitiimit ja offshore-toiminta vaativat korostetun tarkkaa dokumentointia.

Lisäksi oli yleistä, että dokumentointia ei suunniteltu projektikohtaisesti, vaan turvauduttiin vanhoihin käytäntöihin ja malleihin. Dokumentointia tehtiin turhaan, eikä aina edes tiedetty mihin tietoa tullaan käyttämään. Dokumenteissa keskityttiin kertomaan vain suunnitteluratkaisut, eri vaihtoehtojen ja taustojen selvittely oli harvinaista.

Avainsanat

Ketterät menetelmät, dokumentointi, ketterä ohjelmistonkehitys, Rüping

Sisällysluettelo

Tiivistelmä 2
Sisällysluettelo 3
1. Johdanto 5
   1.1 Tutkimusongelma ja tutkimuksen toteutus 6
   1.2 Tutkimuksen tulokset 6
   1.3 Tutkielman rakenne 7
2. Ketterät menetelmät 8
   2.1 Historia 10
   2.2 Nykytilanne 10
   2.3 Ketterä manifesti 10
   2.4 Johdatus Scrum-ohjelmistonkehitysmenetelmään 11
   2.5 XP- ohjelmistonkehitysprosessi 12
   2.6 Ketterä hajautettu (distributed) ohjelmistonkehitys 13
3. Ohjelmiston dokumentointi 15
   3.1 Vaatimusmäärittelyt 15
   3.2 Arkkitehtuuri- ja suunnitteludokumentit 16
   3.3 Tekninen dokumentointi 16
   3.4 Käyttäjädokumentointi 17
4. Tietämyksen johtaminen ketterissä ohjelmiston kehitysmenetelmissä – teoreettinen tausta 18
   4.1 SECI-malli 18
   4.2 Uuden tuotteen kehittämisen peli 19
   4.3 XP perusteet 20
   4.4 Ketterän manifestin periaatteet 20
5. Ketterä dokumentointi: Rüpingin tulkinta 22
   5.1 Dokumentoinnin sisällön kohdentaminen 22
   5.2 Asioiden taustojen ilmaiseminen 23
   5.3 Vastuuhenkilö ja resursointi 24
   5.4 Palautteen saaminen 24
   5.5 Kokonaiskuva, dokumentoinnista tiedottaminen 25
   5.6 Yhteenveto 25
6. Tutkimuksen lähestymistapa 26
   6.1 Tapaustutkimus 26
   6.2 Laadullinen tutkimus 26
   6.3 Teemahaastattelu 27
   6.4 Haastattelujen toteutus 27
   6.5 Tutkimusaineisto 28
7. Haastattelujen tulkinta 29
   7.1 Yritys 1 29
   7.2 Yritys 2 30
   7.3 Yritys 3 31
8. Pohdinta 33
   8.1 Dokumentoinnin sisällön kohdentaminen 34
   8.2 Asioiden taustojen ilmaiseminen 34
   8.3 Vastuuhenkilö ja resursointi 35
   8.4 Palautteen saaminen 35

8.5 Projektin markkinointi ja dokumentin markkinointi 36
9. Loppupäätelmät 37
Lähdeluettelo 39
Liite 1 keskustelurunko: 43

1. Johdanto

Ketterät menetelmät ovat tämän vuosituhannen puheenaihe ohjelmistojen kehityksen alueella. Monet suuret ohjelmistoalan yritykset tutkivat menetelmien käyttöönottoa. Ketterät ohjelmiston sisältävät paljon uusia elementtejä varsinkin organisatorisen tiedon hallintaan.

Ohjelmistoprojektin johtamisen kannalta ei ole mitenkään selvää että kaikilla ohjelmistoprojektin työntekijöillä olisi kaikki projektin toteutukseen tarvittava tieto ja tiedon käytettävyyteen on panostettava. Tietoa on kuitenkin runsaasti. Haaste ohjelmisto tiimissä onkin saada asiakkaalta tuleva tieto kaikkien ohjelmiston kehitystiimin jäsenten käyttöön. Perinteisesti Tayloristisissa menetelmissä tämä on hoidettu massiivisella dokumentoinnilla ja tietokannoilla.

Afile manifesti sanoo ”Toimivia ohjelmia ennen kokonaisvaltaista dokumentaatiota” (Agile Alliance, 2001). Tämän toteuttamiseksi, on ketterissä ohjelmistonkehitysmenetelmissä yritetty parantaa tiedonkulkua suullisen kommunikoinnin avulla. XP-ohjelmistonkehitysmenetelmässä on käytössä pariohjelmointi, Scrum-ohjelmistonkehitysmenetelmässä päiväpalaverit ja jaettu työskentelytila. Myös ketterien prosessien nopeat iteraatiosyklit parantavat kommunikaatiota.

Vaikka ketterissä ohjelmistonkehitysmenetelmissä on pohjimmiltaan ollut alusta saakka paljon tiedon johtamisen elementtejä, käytännössä on menetelmien käyttö keskittynyt vielä ratkaisemaan teknisiä ongelmia ja tiedon johtamisen näkökulmat ovat olleet hyvin toissijaisia. Käytännön tekijät ketterissä ohjelmistotiimeissä ovatkin yleensä pohjakoulutukseltaan suuntautuneet enemmän tekniikkaan kuin johtamiseen tai tiedon johtamiseen. Tutkimuksen mielenkiinto onkin kuinka ketterät menetelmät käytännössä toteuttavat niihin liitettyjä tiedon hallinnan elementtejä ja kuinka vuosia perinteiseen dokumentointiin luottaneet yritykset ovat pystyneet muuttamaan toimintatapojaan ketterien menetelmien aatteita hyödyntäviksi.

Moni argumentti ketterien ohjelmistonkehitysmenetelmien suunnalta on että dokumentointia ei pystytä ylläpitämään ja se harvoin on päivän tasalla. Toisaalta moni vuosituhannen alun tutkimus, on päätynyt lopputulokseen, että dokumentointia käytetään ja se on havaittu hyödylliseksi, vaikka dokumentointi ei olisikaan ihan päivän tasalla. Esimerkiksi Lethbridge, Singer ja Forward (2003).

Ohjelmiston kehitys on tietointensiivinen prosessi. Ohjelmiston kehityksessä yhdistetään asiakkaalta ja tulevalta käyttäjäorganisaatiolta tuleva mitä -tieto ohjelmiston varsinaisen toteutuksen toteuttavan organisaation tekniseen miten -tietoon. Tiedon johtamisella on siis varsin suuri merkitys ohjelmiston kehitysprosessin onnistumisessa. Myös kyky kehittyä, ymmärtää ja oppia asiakkaan tietämystä järjestelmien soveltamisalasta voidaan pitää merkittävänä kilpailuetuna. Ohjelmistoyrityksen arvo muodostuu pitkälti työntekijöiden hallussa olevasta tiedosta eikä koneista ja laitteista kuten perusteollisuudessa. (Bjornson, 2008; Highsmith, 2002; Takeuchi & Nonaka, 1986.)

Dokumentointi edustaa Nonakan ja Takeuchin (1995) tiedon spiraalissa tiedon muuttumista yksilön tiedosta yhteisölliseksi (dokumentin kirjoittaminen) ja toisinpäin (dokumentin tiedon hyödyntäminen)

1.1 Tutkimusongelma ja tutkimuksen toteutus

Monet tietotekniikan yritykset Oulun alueella ovat ottaneet ketterät ohjelmistonkehitysmenetelmät käyttöön. Ketterät ohjelmistonkehitysmenetelmät eivät kuitenkaan ole yksikäsitteisiä malleja, vaan enemmänkin pohjia, jonka mukaan kukin organisaatio luo oman mallinsa sille, miten he toteuttavat ketteriä prosesseja.

Tässä tutkimuksessa tutkittiin räätälöityjä prosesseja. Tutkimuksessa paneuduttiin yksinkertaisilla haastatteluilla yritysten kehittämiin ketteriin ohjelmistonkehitysprosesseihin ja siihen, mitenkä prosessit huomioivat ketterien ohjelmiston kehitysmenetelmien taustalla olevat tiedon hallinnan teoriat.

Tiedonjohtamisen teoriat, mitkä ovat olleet pohjana ketterien ohjelmistonkehitysmenetelmien kehityksessä, ovat tuoneet ketteriin menetelmiin voimakkaasti ajatuksen suullisen kommunikoinnin korostamisesta. Ketterissä ohjelmistonkehitysmenetelmissä korostetaan myös toimivan ohjelman toimittamista suhteessa kokonaisvaltaisen dokumentoinnin korostamiseen.

Tutkimusongelmaa ratkaistaessa haettiin vastausta seuraaviin kysymyksiin:

Ovatko yritykset Oulussa siirtyneet luottamaan suulliseen kommunikointiin vai tehdäänkö yhä perinteisen mallisia laajoja dokumenttikirjastoja?

Korostetaanko toimivaa ohjelmaa vai vaaditaanko kokonaisvaltaista dokumentointia?

Onko ketteriin ohjelmistonkehitysmenetelmiin siirtyminen muuttanut dokumentoinnin tarvetta?

Tutkimus rajattiin käytännön syiden ja resurssirajoitusten vuoksi koskemaan Oulun alueella toimivia ohjelmistoalan yrityksiä, jotka olivat siirtyneet käyttämään ketteriä ohjelmistonkehitysmenetelmiä. Tutkimuksessa rajattiin tarkoituksella ulos käyttäjädokumentointi ja markkinointimateriaali, jotka ovat ketterien ohjelmistonkehitysmenetelmien näkökulmasta vain tekemisen kohteita ja vaatimuksia. Tutkimus oli teemahaastatteluna toteutettu tapaustutkimus, johon oli saatu haastattelut kolmesta Oulussa toimivasta yrityksestä.

Tutkimus pohjautui tiedonjohtamisen teorioihin, joita löytyy ketterien ohjelmistonkehitysmenetelmien takaa. Teemahaastattelujen pohjalta tutkimuksen tulosta analysoitiin Andreas Rüpingin ”Agile Documentation”-kirjassa (2005) esitettyjen ohjeiden (pattern) avulla.

1.2 Tutkimuksen tulokset

Onko ketteriin ohjelmistonkehitysmenetelmiin siirtyminen muuttanut tämän tutkimuksen kohteena olleiden yritysten dokumentointi käytäntöjä? Haastattelut kertoivat aika selvästi että tieto projekteissa välittyy yhä dokumenttien avulla. Yllättävää ei ollut, ettei ketteriin menetelmiin siirtyminen ole oikeastaan tuonut mitään muutosta dokumentoinnin tarpeeseen. Jopa kasvua erilaisten raporttien muodossa ilmeni. Kaksi kolmesta haastatelluista kertoi, että virtuaalitiimit ja offshore-toiminta vaatii dokumentointia.

Dokumentointia ei suunniteltu projektikohtaisesti, vaan turvauduttiin vanhoihin käytäntöihin ja malleihin. Dokumentointia tehtiin turhaan, eikä aina edes tiedetty mihin tietoa tullaan käyttämään. Dokumenteissa keskityttiin kertomaan vain suunnitteluratkaisut, eri vaihtoehtojen ja taustojen selvittely oli harvinaista.

1.3 Tutkielman rakenne

Tutkielman ensimmäinen luku toimii johdantona. Toisessa luvussa käydään läpi ketterien ohjelmiston kehitysmenetelmien taustoja sekä tutustutaan pariin yleisimpään menetelmään Scrum- ja XP-ohjelmistonkehitysmenetelmä hiukan tarkemmin. Kolmannessa luvussa avataan dokumentoinnin merkitystä ohjelmistojen kehitystyössä. Neljännessä luvussa tutustutaan joihinkin ketterien ohjelmistonkehitysmenetelmien taustoilla vaikeuttaneisiin tiedonjohtamisen teorioihin.

Luvussa 5 tutustutaan tämän tutkimuksen kantavana voimana olevaan joukkoon ohjeita Rüping (2005) siitä, miten dokumentointi voitaisiin tehdä kohdennetusti ja tehokkaasti ketterien ohjelmistonkehitysmenetelmien ajatusmaailman mukaisesti.

Luvussa 6 tutustutaan käytettävään tutkimusmenetelmään. Luvussa 7 tulkitaan haastatteluja. Luvussa 8 pohditaan saatuja tuloksia ja lopuksi luvussa 9 esitellään yhteenveto.

2. Ketterät menetelmät

Ketterät ohjelmiston kehitysprosessit ovat saaneet viimeaikoina paljon huomiota osakseen. Ketterät ohjelmiston kehitysprosessit yrittävät tarjota nopeampia ja joustavampia menetelmiä toteuttaa ohjelmistoja (Abrahamsson, Warsta, Siponen, & Ronkainen, 2003). Ketterät ohjelmiston kehitys menetelmät ovat joukko menetelmiä, joilla ei ole yhteistä kantaisää, vaan ketterien ohjelmiston kehitysmenetelmien kehitys on lähestynyt samoja ongelmia useammalta eri suunnalta (Abrahamsson et al., 2003). Scrum ja XP olivat eniten käytetyt ketterät menetelmät (Shine Technologies Pty Ltd, 2003).

Ketterät menetelmät eivät yleensä tarjoa yksikäsitteisiä ohjeita ja neuvoja siihen, miten ohjelmistoprojekti käytännössä pitäisi järjestää, vaan tarjoavat enemmänkin ohjeita (Abrahamsson et al., 2003). Menetelmiä myös yleensä muokataan ja kehitetään kunkin organisaation tarpeiden mukaiseksi. Tässä tutkimuksessa esille tuotavia havaintoja ei voida siis suoraan yhdistää menetelmien virallisiin tapoihin vaan niiden käytännön sovituksiin.

Ketterissä ohjelmistonkehitysmenetelmissä ei ole mitään uutta, samanlaiset käytännöt ovat olleet mukana ohjelmiston kehityksessä 1960-luvulta asti (Merisalo-Rantanen, Tuunanen, & Rossi, 2005).

Ketterän lähestymistavan tarkoituksena on lisätä joustavuutta, ketteryyttä ja olla soveltuvampi nykypäivän ohjelmankehitysympäristöjen kanssa. Tämä ei sovellu niin hyvin isoille globaaleille projektiorganisaatioille, joissa kokonaisuus ei näy ja joilla on useita päällekkäisiä riippuvuuksia, joita ei voida tehokkaasti seurata. Toisaalta mikään ei estä myöskään isoja organisaatioita ottamasta ideoita ja käytänteitä ketteristä menetelmistä. (Rao, Naidu, & Chakka, 2011.)

	Perinteinen	Ketterä
Lähtökohtaoletukset	Systeemi on täysin määriteltävissä ja voidaan toteuttaa huolellisen suunnittelun avulla.	Korkealaatuinen, mukautuva ohjelmisto voidaan kehittää pienien tiimien avulla hyödyntäen jatkuvan suunnittelun periaatteita.
Kontrolli	Prosessilähtöinen	Ihmislähtöinen
Johtamistyyli	Käskytys ja valvonta	Johtajuus ja yhteistyö
Tiedonjohtaminen	Explicit	Tacit
Roolit	Henkilökohtaiset – suosii erikoistumista	Itse organisoituvat tiimit - suosii roolien kierrätystä
Kommunikointi	Formaali	Epäformaali
Asiakkaan rooli	Tärkeä	Kriittinen
Projektin aikataulutus	Tehtävien ja aktiviteettien määräämä	Tuoteominaisuuksien ohjaama
Kehitysprosessi	Elinkaarimalli (Waterfall, Spiral tai joku variaatio)	Evolutionary-delivery - malli
Oletettu organisaation muoto/rakenne	Mekaaninen – byrokraattinen	Orgaaninen, joustava ja osallistumista ja yhteistyötä tukeva
Teknologia	Ei rajoituksia	Suosii oliopohjaisia teknologioita.

Taulukko 1. Vertailu: perinteinen ja ketterä ohjelmistonkehitys (Nerur, Mahapatra, & Mangalaraj, 2005,75).

Taulukossa 1 on listattu perinteisten ja ketterien ohjelmistonkehitysmetodien eroja. Toisin kuin perinteisissä prosesseissa, ketterissä ei odoteta järjestelmän vaatimuksien olevan täysin jäätyneitä toteutuksen alkaessa ja tiukkojen prosessien noudattamisen sijaan ketterät ohjelmistonkehitysmenetelmän luottavat enemmän ihmiseen. ketterissä menetelmissä on myös luonteenomaisena piirteenä lyhyet iteraatiosyklit ja tiimien itseohjautuvuus. (Nerur et al., 2005.)

Ketterien ohjelmistonkehitys menetelmien tutkimus on vielä hyvin lasten kengissä. Dybå ja Dingsøyr (2008) löysivät 1996 tutkimusta, jotka käsittelivät ketteriä ohjelmistonkehitysmenetelmiä. Näistä 36 oli hyväksyttävällä tasolla luotettavuuden ja merkityksen näkökulmasta. Tutkimuksissa oli löydetty monia ketteriä ohjelmistonkehitysmenetelmien etuja ja heikkouksia, mutta tutkimusten todistusaineisto oli huono.

2.1 Historia

Ohjelmistokehitys on perinteisesti perustunut vesiputousmalliin, jossa tavoitteena on ensisijaisesti projektin vieminen läpi tietyssä ajassa ja tietyillä resursseilla. Vesiputousmallissa sovelluksen määrittelytyö tehdään yhteistyössä tilaajan kanssa. Määrittelytyön tulokset dokumentoidaan eli kirjataan tarkasti ylös paperille. Asiakkaan hyväksynnän jälkeen määrittely lyödään lukkoon ja aloitetaan sovelluksen teknisen toteutuksen suunnittelu. Suunnittelutyön tulokset dokumentoidaan ja lyödään lukkoon. Sovellus toteutetaan ja toiminnallisuus testataan. (Royce, 1987.)

Perinteinen Tayloristinen menetelmä on kuitenkin huono monella tapaa. Lyhyessä vaatimusmäärittelyvaiheessa ei voida kirjata kaikkea tilaajaorganisaatiossa olevaa hiljaista tietoa paperidokumentteihin. Myöskään toteuttava tiimi ei pysty antamaan tilaajaorganisaatiolle palautetta mahdollisuuksista, koska toimivaa tuotetta ei päästä esittelemään asiakkaalle kuin ihan projektin lopussa. Projektin sisäinen kommunikaatio on yleensä dokumentteihin perustuvaa, mikä rajoittaa uusien ideoiden syntyä. Perinteinen vesiputousmalli ei myöskään tue vaatimusten uudelleen arviointia ja päivittämistä. (Agile Alliance, 2001; B. Boehm, 2002; Cao & Ramesh, 2008; Chau, Maurer, & Melnik, 2003; Chow & Cao, 2008.)

2.2 Nykytilanne

Tänä päivänä sovellusten toteuttamiseen käytössä oleva aika on pienempi kuin aikaisemmin (Maurer & Martel, 2002):

- Sovelluksille asetettavat vaatimukset muuttuvat nopeassa tahdissa.
- Sovelluksen laadun kriteerit poikkeavat perinteisestä ohjelmistokehityksestä.
- Tuotteen saaminen ajoissa (ensimmäisenä) markkinoille on olennaista.

Myös ohjelmiston vaatimukset ovat monimutkaisempia. Pystyäkseen laatimaan ohjelmiston ohjelmistotiimin on hallittava sovellusalueensa hyvin. (Maurer & Martel, 2002.)

XP on yleisimmin käytetty ketterä menetelmä. 70% projekteista, joissa ketteriä ohjelmistonkehitysmenetelmiä käytettiin, tekivät internet-ohjelmistoja. (Vijayasarathy & Turk, 2008.)

Pienten organisaatioiden lisäksi myös isoissa organisaatioissa on tunnustettu ja löydetty tarve ketterien ohjelmistonkehitysmenetelmien hyödyntämiselle. Ohjelmistonkehitystiimit kohtaa jatkuvasti tarpeen nostaa tuottavuutta samalla kun ohjelmiston laadun pitäisi vähintään pysyä samalla tasolla. Ohjelmiston kehityksessä on normaalia, että kehitystyö pitää aloittaa jo siinä vaiheessa kun vaatimuksista vain osa on selviä. Tämän takia myös isot organisaatiot tutkivat mahdollisuuksia käyttää ketteriä ohjelmistonkehitysmenetelmiä. Toiseen ongelmaan törmätään, kun vaatimukset siirretään kehitystiimille korkeammalla tasolla. Tällöin vaatimuksista on vaikeaa purkaa yksityiskohtainen ohjelmistospesifikaatio. Ohjelmistospesifikaation laadinta on myös hidasta ja resursseja vaativaa jolloin valmiit ovat ajasta jäljessä jo valmistuessaan. Tämä ajaa organisaatioita toimimaan ketterämmin. (Lindvall et al., 2004.)

2.3 Ketterä manifesti

Vuonna 2001 merkittävimpiä ketterän ohjelmistonkehityksen edelläkävijöitä kokoontui Snowbirdin hiihtokeskukseen Utahissa keskustelemaan menetelmiensä yhteisestä perustasta. Tarkoitus oli luoda yhteistä pohjaa ketterille menetelmille ja edistää näin ketterän

ajattelun leviämistä. Tuon kokoontumisen tuloksena he julkaisivat julistuksen nimeltä Agile Manifesto (’Ketterä manifesti’), jota pidetään ketterän kehityksen perusmääritelmänä. Manifestissa määritellään ketterille menetelmille neljä tyypillistä arvoa, sekä 12 periaatetta, joita menetelmät noudattavat. (Agile Alliance, 2001.)

Ketterässä manifestissa on esitetty 4 arvoa joihin ketterät menetelmät perustuvat (Agile Alliance, 2001). Arvot voidaan kääntää seuraavasti:

> ***Yksilöitä ja vuorovaikutusta*** *enemmän kuin prosesseja ja työkaluja*
>
> ***Toimivaa sovellusta*** *enemmän kuin kokonaisvaltaista dokumentaatiota*
>
> ***Asiakasyhteistyötä*** *enemmän kuin sopimusneuvotteluita*
>
> ***Muutokseen reagoimista*** *enemmän kuin suunnitelman noudattamista.*

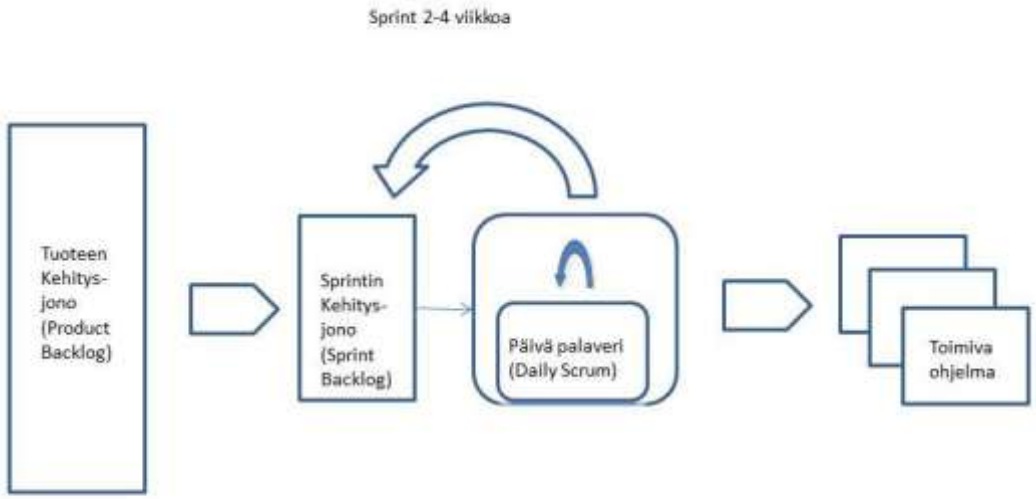
Ketterät ohjelmiston kehitysprosessit ovat siis joukko menetelmiä. Julistuksen mukaan menetelmät kuitenkin omaavat joitain yhteisiä piirteitä, kuten toistuvien julkaisujen toteuttaminen (Agile Alliance, 2001), yleensä jopa viikon välein. Asiakas rajapinnassa olevien ihmisten täytyy työskennellä päivittäin yhdessä toteuttavien ihmisten kanssa (Agile Alliance, 2001).

Perinteinen tapa toteuttaa ohjelmistonkehitysprosessi on, että systeemiasiantuntijat kirjoittavat määrittelydokumentin ja myöhemmin ohjelmistokehitystiimi toteuttaa sen. Kommunikointi perinteisissä malleissa perustuu lähes yksinomaan paperidokumentteihin. Ketterät ohjelmiston kehitysmenetelmät taas välttävät ylimääräisten dokumenttien kirjoittamista ja keskittyvät käyttämään suullista tietoa. Vain tarpeellinen tieto kirjoitetaan eksplisiittiseen muotoon. (Agile Alliance, 2001.)

2.4 Johdatus Scrum-ohjelmistonkehitysmenetelmään

Scrummin tekee erittäin mielenkiintoiseksi historiasta löytyvät nimet jotka ovat varsin tuttuja myös oppivan organisaation teorioissa. Scrummin kehittymisen voidaan sanoa alkaneen artikkelista Takeuchi ja Hirotaka (1986). Tässä artikkelissa kyseiset tutkijat rinnastivat ohjelman kehityksen Rugby-peliin, mistä myös ovat lähtöisin monet Scrummissa käytetyt termit.

Scrum on yksi yleisimmin käytetyistä ketteristä ohjelmistonkehitysmenetelmistä (Shine Technologies Pty Ltd, 2003). Scrum-ohjelmistonkehitysmenetelmä on kuvattu kirjassa (Schwaber, 2002). Kuvassa yksi on yksinkertaistettu kuvaus Scrum-ohjelmistonkehitysmenetelmästä.



Kuva 1. Scrum prosessi (Schwaber, 2002).

Tuotteen suunnittelu ja projektin suunnittelu aloitetaan laatimalla lista vaatimuksista tuotteen kehitysjonoksi (product backlog) nimettyyn dokumenttiin. Vaatimukset priorisoidaan ja jaetaan tuleville sprinteille. Yhden sprintin aikana toteutettavaksi suunnitellut vaatimukset muodostavat sprintin kehitysjonon (sprint backlogin). Scrum- projekti koostuu noin 2-4 viikkoa pitkistä sprinteistä, Jotka ovat lyhyitä iteraatioita. Jokaisen sprintin tavoitteena on toimittaa toimiva kokonaisuus, prototyyppi. Jokainen sprintti alkaa sprintin backlogin päivittämisellä. Sprintin kehitysjono on lista niitä ominaisuuksista, jotka ovat kyseiselle ajanjaksolle valittu tehtäväksi.

Projekti tiimin sisäistä tiedon siirtymistä edistetään päivittäisellä päiväpalaverilla. Päiväpalaveri on nopea epäformaali kokoontuminen, noin 15min (stand-up meeting). Palaverissa käydään nopeasti kunkin projekti tiimin jäsenen töiden tilanne ja mitä kukin tekee seuraavan päivän aikana.

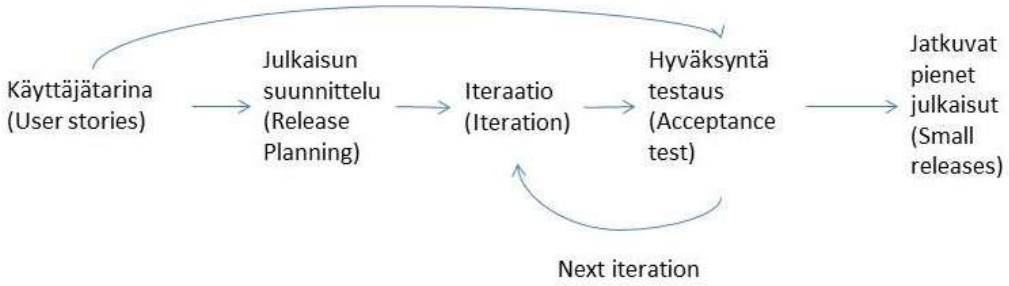
Scrum-projektissa asiakkaan ja tilaajan näkökulmaa edustaa projektiryhmässä henkilö jolla on tuoteomistajarooli (product owner) ja jonka vastuulla on liiketoiminnallisen näkökulman tuominen projektiin. Tuoteomistaja edustaa siis kaikella lailla asiakkaan ja käyttäjän näkökulmaa. (Schwaber, 2002.)

Vaatimusmäärittely scrum-projektissa kirjataan tuotteen kehitysjono nimiseen dokumenttiin, joka sisältää listan halutuista toiminnoista (Schwaber, 2009). Scrummissa korostetaan sitä että ohjelmisto ryhmän tulisi työskennellä samassa tilassa tai ainakin muuten lähekkäin. Joka päiväsen päiväpalaverin tarkoituksena on välittää tietoa siitä kuka tekee mitäkin, ei niinkään teknisiä yksityiskohtia. (Schwaber, 2002.)

2.5 XP- ohjelmistonkehitysprosessi

Toinen yleisimmistä ketteristä ohjelmiston kehitys prosesseista on XP-ohjelmistonkehitysmenetelmä (eXtreme Programming), joka tunnetaan myös yleisesti lyhenteellä XP. XP:n virallisena kuvauksena voidaan pitää kirjaa *Extreme programming*

explained embrace change (Beck, 2000). XP- prosessi on voimakkaasti iteratiivinen. Yksittäinen iteraatio pituudeltaan vain noin viikon luokkaa (Beck, 2000).



Kuva 2. XP-ohjelmistonkehitysmenetelmä (Beck, 2000).

Kuvan kaksi mukaan XP-ohjelmistonkehityksessä vaatimustenmäärittely alkaa laatimalla kortit, joita kutsutaan käyttäjätarinoiksi (user story). Käyttäjätarinat laaditaan yhdessä käyttäjän ja ohjelmiston kehittäjän kanssa. Käyttäjätarinoita käytetään sitten julkaisun suunnittelussa (release planning). XP-ohjelmistonkehitysmenetelmässä käyttäjätarinat ovat myös pohja hyväksymistesteille. Käyttäjätarinoiden luonti ja ylläpito käytännössä muodostaa XP-ohjelmistonkehitysmenetelmän vaatimusmäärittelyn. Jokaisessa iteraatiossa on tarkoituksena saada aikaan toimiva ohjelmisto, jonka toteutetut ominaisuudet testataan ja hyväksytään.

XP nojaa kommunikoinnissa lähes yksinomaan suulliseen yhteydenpitoon. XP-ohjelmistonkehitysmenetelmän mukainen ohjelmistoprojekti ei tarvitse analyysi- tai suunnitteludokumentteja. Toisaalta XP on suunniteltu pienille tiimeille kooltaan kahdesta – kymmeneen ohjelmoijaa. (Briand, 2003.)

XP-ohjelmistonkehitysmenetelmästä on olemassa uudempi versio joka pohjautuu kirjaan Beck ja Andres (2004). Uudemmassa versiossa XP-ohjelmistonkehitysmenetelmässä on alkuperäisen XP:n neljän arvon: kommunikointi, yksinkertaisuus, palaute ja rohkeus lisäksi viides: kunnioitus. Myös käytännössä uusi XP on hyvin erilainen kuin Beck (2000).

2.6 Ketterä hajautettu (distributed) ohjelmistonkehitys

Johdannossaan Braithwaite ja Joyce (2005) kertoo, että vähäinen mutta kasvava tutkimus ketterästä hajautetusta ohjelmistonkehityksestä antaa varsin pessimistisen kuvan. Hyvin usein tutkimuksissa kirjoittajat olettavat, että mikäli ohjelmistonkehitystiimin jäsenet ovat hajallaan, niin ketterien ohjelmistonkehitysmenetelmien kommunikointimenetelmät eivät toimi. (Braithwaite & Joyce, 2005.)

Hajautettu ketterä ohjelmistonkehitys voidaan jakaa seuraaviin luokkiin: (Braithwaite & Joyce, 2005.)

1. Ketterän ohjelmistonkehityksen alihankinta (Agile Outsoursing), missä ketterä ohjelmistonkehitys tapahtuu jossakin sopivassa maassa.
2. Hajanainen ketterä ohjelmistonkehitys (Agile Dispersed Development): mikä on käytössä monissa avoimen lähdekoodin-projekteissa, tiimin jäsenet ovat hajallaan, mutta jatkuvasti yhteyksissä eri kommunikointikanavien kautta.
3. Hajautettu ketterä ohjelmistonkehitys (Distributed Agile Development): Yksi kehitystiimi on hajautettu useammalle paikkakunnalle ollakseen lähellä asiakasta. Tehokas ja tiheä kommunikointi varmistaa, ettei ketterien ohjelmistonkehitysmenetelmien periaatteita rikota.

Lähestymistapoja ketterän hajautetun ohjelmistonkehityksen organisoimiseksi on monia: esimerkiksi Braithwaite ja Joyce (2005) esittää yhden mallin jakaa tavat perustuen Ihmisiin, kommunikointiin ja koodiin.

Hajautettu ohjelmistonkehitys tukeutuu formaaleihin mekanismeihin kuten esimerkiksi yksityiskohtaisiin arkkitehtuurimäärittelyihin. Ketterät ohjelmistonkehitysmenetelmät taas tukeutuvat epämuodollisiin kommunikointitapoihin, kuin muodolliseen dokumentointiin. Tämä luo todellisen haasteen ketterien ohjelmistonkehitysmenetelmien käytölle hajautetussa ohjelmistonkehitysympäristössä. (Ramesh, Cao, Mohan, & Xu, 2006.)

Ramesh et al (2006) toteavat että hajautettu ohjelmistonkehitys myös rajoittaa mahdollisuuksia ohjata etäällä toimivien tiimien toimintaa. Ohjelmiston kehityksen ollessa hajautettu joudutaan ohjaamiseen käyttämään kiinteitä laatuvaatimuksia, kun taas ketterät ohjelmistonkehitysmenetelmät tukeutuvat jatkuvaan keskusteluyhteyteen asiakkaan kanssa.

Ketterät ovat kontrollin suhteen myös hyvin ihmisläheisiä, kun taas hajautettu ohjelmistonkehitys on ollut hyvin prosessikontrolloitua. Ketterissä on myös hyvin merkittävässä asemassa ryhmähenki, joka hajautetussa on taas vaikea luoda. (Ramesh et al., 2006.)

Hajautetussa ohjelmistonkehityksessä dokumentoidut muodolliset sopimukset (agreement) ovat merkittävässä asemassa. Ketterässä ohjelmistonkehityksessä taas luotetaan paljolti suulliseen sopimiseen. (Ramesh et al., 2006.)

Tästä huolimatta Braithwaite ja Joyce (2005) toteaa, että hajautettu ohjelmistonkehitys voi toimia ja menestyä hyödyntäen sekä ketteriä että hajautettuja ohjelmistonkehitysmalleja.

3. Ohjelmiston dokumentointi

Ohjelmiston dokumentointi on monitahoinen asia. Tutkimusaineiston ja heterogeenisyyden takia tässä tutkimuksessa tutustuttiin ohjelmiston dokumentointiin hyvin laajana asiana.

Ohjelmistondokumentointi voidaan määritellä seuraavasti (Wikipedia,s.d.):

Software documentation *or* ***source code documentation*** *is written text that accompanies computer software. It either explains how it operates or how to use it, and may mean different things to people in different roles.*

Projektissa ei pitäisi tuottaa mitään sellaisia dokumentteja tai välituloksia, mitä ei myöhemmin käytetä (Ruuska, 2001)

Dokumentaatio on harvoin ajan tasalla. Arkkitehtuuri ja muu abstrakti dokumentaation tieto on kuitenkin yleensä paikkansa pitävää tai ainakin tarjoaa historiallista näkökulmaa, jota ylläpitäjät voivat hyödyntää. Lisäksi yleensä dokumentaatiota on kuitenkin liikaa ja se on huonosti kirjoitettua. Tämän takia tiedon löytäminen dokumentaatiosta saattaa olla niin paljon resursseja kuluttavaa, ettei sitä edes yritetä. (Lethbridge, Singer & Forward, 2003.)

Tässä luvussa on esitelty tärkeimpiä ohjelmistoprojektin dokumentointi luokkia. Vaatimusmäärittelyt, arkkitehtuuri- ja suunnitteludokumentit ja tekninen dokumentointi määrittelevät ja auttavat ohjelmiston tekemistä. Loppukäyttäjän dokumentit ja markkinointimateriaali ovat sitten ohjelmiston käyttöä ja markkinointia varten.

3.1 Vaatimusmäärittelyt

Vaatimusmäärittelydokumentaatio on kuvaus siitä mitä kyseinen ohjelma tekee tai sen pitäisi tehdä. Vaatimusmäärittelyä käytetään läpi ohjelmiston kehityksen ohjelmiston sisällöstä kommunikoimiseen. Vaatimusmäärittelyt on hyvin usein myös pohja sopimuksille, mitä ohjelmiston pitäisi toteuttaa. Vaatimusmäärittelyjä hyödyntää ja tuottaa kaikki ohjelmistotuotannon toimijat: loppukäyttäjistä ohjelmiston toteuttajiin. (Chih-Wei Ho, Johnson, Williams, & Maximilien, 2006; CMMI Product Team, 2006.)

Perinteisesti, jos vesiputousmallia voidaan siten nimittää, ohjelmiston kehitysprosessi lähtee liikkeelle vaatimusmäärittelystä. Ketterissä ohjelmistonkehitysmenetelmissä vaatimusmäärittely taas jakautuu tyypillisesti koko ohjelmistonkehitysjaksolle. (Cao & Ramesh, 2008.)

Vaatimusmäärittely on prosessi missä pyritään selvittämään mitä ominaisuuksia ohjelmistolla pitäisi olla, missä ohjelmistoa tai järjestelmää käytetään ja kuka käyttää (Endres & Rombach, 2003). CMMI kuvaa vaatimusmäärittelyprosessia seuraavasti: Sidosryhmien tarpeet, toiveet, rajoitukset, rajapinnat, toiminnalliset konseptit ja tuotekonseptit analysoidaan, harmonisoidaan, jalostetaan ja hiotaan joukoksi asiakasvaatimuksia. (CMMI Product Team, 2006.)

Vaatimusmäärittely on ohjelmiston kehitysprosessin suurin virheiden aiheuttaja (B. W. Boehm, McClean, & Urfrig, 1975). Vaatimusmäärittelyn virheriskin vähentämiseksi voidaan käyttää erilaisia menetelmiä kuten prototyyppejä ja mallintamista (Endres & Rombach, 2003).

Ketterien ohjelmistonkehitysmenetelmien kehitysmetodit poikkeavat vaatimusmäärittelyn osalta perinteisestä sikäli, että jos verrataan vaikkapa perinteiseen vesiputousmalliin, jossa ohjelmiston vaatimukset ja projektinsuunnittelu ja hinnoittelu lukitaan hyvin aikaisessa vaiheessa, niin ketterissä ohjelman kehitysprosesseissa tarkoituksella tätä toimintaa jaetaan koko projektin ajalle. Asiakkaan näkökulmasta tämä tarkoittaa muun muassa sitä että he joutuvat allokoimaan omia asiantuntijoitaan huomattavasti perinteistä ohjelmiston kehitysprosessia pitemmälle ajanjaksolle, mutta myös sitä että hinta-arviot ovat summittaisia eikä ohjelmiston toimittaja välttämättä sitoudu niihin. (Cao & Ramesh, 2008; Cao & Ramesh, 2008; Chau et al., 2003; Cohn & Ford, 2003.)

Monesti on vaikea tietää kuinka tarkasti vaatimuksia pitäisi kirjoittaa ylös vaatimusdokumentteihin ja kuinka paljon asioita pitäisi jättää arkkitehtuuri ja toteutus dokumentointiin.

Ketterissä ohjelmistonkehitysmenetelmissä näihin ongelmiin on pyritty vastaamaan nopeilla integraatiosykleillä ja jatkuvalla prototyyppien rakentamisella. Tilaajan ja käyttäjien edustus sidotaan näin koko projektin ajaksi kiinni jatkuvaan vaatimusmäärittelyyn. (Cohn & Ford, 2003; Martin, Biddle, & Noble, 2004; Turk, France, & Rumpe, 2005.)

Vaatimusmäärittelystä kootaan ketterissä tuotekehitysprojekteissa yleensä kehitysjono, joka sitten toimii ketterin prosessien nopeiden iteraatioiden työlistana.

3.2 Arkkitehtuuri- ja suunnitteludokumentit

Arkkitehtuuridokumenttien rooli on ollut ohjelmiston kehityksessä enempi suunnittelua tukeva kuin olemassa olevaa dokumentoiva. Arkkitehtuuridokumentointia käytetään vähempi validointiin ja testaamiseen. Käytännössä arkkitehtuurin dokumentointi on yhtä tärkeää, kun arkkitehtuurin hahmotteleminen ja suunnittelu, sillä jos arkkitehtuuri ei ole ymmärretty oikein ei järjestelmän kehitys voi saavuttaa tavoitteitansa. (Clements, Garlan, Little, Nord, & Stafford, 2003.)

Arkkitehtuuri- ja suunnitteludokumentit kuvaavat systeemiä kolmella tapaa (Clements et al., 2003):

- Kuinka järjestelmä on jaettu pienempiin osiin
- Kuinka järjestelmä on jaettu osiin ja kuinka osat vaikuttavat toisiinsa ohjelmiston toiminnan aikana
- Kuinka järjestelmän osat vaikuttavat ympäristöön ja kommunikoivat ulospäin

3.3 Tekninen dokumentointi

Tekninen dokumentointi on se mitä yleensä ohjelmoijat tarkoittavat termillä ohjelmiston dokumentointi. Rivien välistä on myös tulkittavissa, että kun ketterissä menetelmissä korostetaan suullista kommunikaatiota ja esitetään että kokonaisvaltaista dokumentaatiota ei tarvita, niin puhutaan nimenomaan teknisestä dokumentoinnista. (Agile Alliance, 2001.)

Ohjelmistoa kehitettäessä pelkkä koodi yksin ei ole riittävä vaan se tarvitsee lisäksi hiukan tekstiä jotta se olisi ymmärrettävää. Koodidokumentointi on sikäli haastavaa että sillä on taipumus jäädä muutoksissa jälkeen. (Agile Alliance, 2001.)

3.4 Käyttäjädokumentointi

Käyttäjädokumentointi toisin kuin tekninen dokumentointi on ketterien ohjelmistonkehitysmenetelmien näkökulmasta vain yksi vaatimus toteutettavaksi eikä ole siten kiinnostava tämän tutkimuksen näkökulmasta, paitsi kun kyseessä on ohjelmistokirjasto, jota käytetään osana projektin kohteena olevaa tuotetta. Tällöin käyttäjädokumentointi on käytännössä sama kuin tekninen dokumentointi. (Wikipedia,s.d.; Schwaber,2009; Maurer & Martel, 2002.)

Myös markkinointimateriaali on ketterien ohjelmistonkehitysmenetelmien näkökulmasta vain yksi lopputuote, joten se ei ole tämän tutkimuksen näkökulmasta kiinnostava. . (Wikipedia,s.d.; Schwaber,2009; Maurer & Martel, 2002.)

4. Tietämyksen johtaminen ketterissä ohjelmiston kehitysmenetelmissä – teoreettinen tausta

Ohjelmistotyöhön vaikuttavat samat lait ja teoriat kuin muuhunkin organisatoriseen oppimiseen. Seuraavaksi on esitelty niistä tärkeimmät. Lisäksi tässä yhteydessä on esitelty muutama ohjelmistojen kehitysmenetelmiin, varsinkin ketterien ohjelmistojen kehitysmenetelmien tietämyksen hallintaan liittyvä malli.

Tiedon johtamiseen kuuluu lisäksi joukko teorioita, joita ei tässä yhteydessä käydä tarkemmin läpi. Näitä teorioita ovat esittäneet muun muassa Argyris ja Schön (1996), Senge, Kleiner ja Roberts (1994) ja Pedler, Boydell ja Burgoyne (1989).

SECI-malli (Nonaka & Takeuchi, 1995) on eräs uuden tiedonluonnin perusteorioita. Malli kuvaa tiedon muuttumista ja kehittymistä yksilöllisen, hiljaisentiedon ja yhteisöllisen ja kirjoitetun tiedon muunnosten avulla. Lisäksi tässä luvussa on esitelty Scrummenetelmän eräs perustateorioista Takeuchi ja Nonaka (1986) sekä XP menetelmää tutkiva vähän uudempi teoriapohja Kahkonen ja Abrahamsson, (2003).

4.1 SECI-malli

Nonaka ja Takeuchi (1995) loivat teorian uuden tiedon muodostumisesta tietospiraalin avulla. Tiedon luomisen spiraali ja Ba- käsite muuttivat 1990-luvulla käsityksiä tiedon luomisesta.

Aiemmin tietoa oli lähestytty pääasiassa siltä kannalta, kuinka organisaatiot sopeutuvat muutoksiin ja käsittelevät jo olemassa olevaa tietoa. Nonakan ja Takeuchin malli kuvaa, kuinka uutta tietoa luodaan aktiivisesti ja näin vaikutetaan ympäristöön. Nonaka ja Takeuchi yhdisti ajattelussaan länsimaisen kartesiolaisuuden ja japanilaisen filosofian. Tieto määriteltiin dynaamiseksi inhimilliseksi prosessiksi, jossa on mukana toiminta. Tämä lähestymistapa mahdollisti tiedon luomisen ja innovaatioiden ymmärtämisen uudella tavalla. Nonaka ja Takeuchi jakaa tiedon eksplisiittiseen ja hiljaiseen tietoon. Eksplisiittinen tieto on yleistä, helposti tallennettavissa, siirrettävissä ja esitettävissä. Se on järkiperäistä ja voidaan irrottaa kontekstista. Eksplisiittinen tieto voidaan esittää sanallisesti tai numeraalisesti. Suurin osa tiedosta on kuitenkin hiljaista tietoa. Se on kokemuksellista, henkilökohtaista, kontekstisidonnaista ja vaikeaa jakaa tai ilmaista. Sitä voidaan kuitenkin jakaa henkilökohtaisessa vuorovaikutuksessa. Hiljaisen tiedon kautta avautuu ymmärrys tiedon luomiseen. (Nonaka & Takeuchi, 1995.)

SECI- malli kuvaa tiedon muuttumista hiljaisesta tiedosta, organisatoriseksi tiedoksi ja taas takaisin hiljaiseksi tiedoksi. Kuvassa 3 on esitetty Nonakan ja Takeuchin SECI-mallin periaate.

Kuva 3. Tiedon spiraali (Nonaka & Takeuchi, 1995)**.**

Nonakan ja Takeuchin malli perustuu neljään tiedon muunnokseen. Sosialisaatio eli tiedon muuntuminen tai siirtyminen hiljaisena tietona ihmiseltä toiselle. Ulkoistamisessa tieto muuntuu yksilön hiljaisesta tiedosta organisaation käytettävissä olevaan muotoon, yleensä kirjalliseen muotoon. Kombinaatiossa organisaationaalista tietoa yhdistellään luoden samalla uutta osaamista. Ja viimeisenä mallin muunnoksena on Sisäistäminen jossa organisaation tieto tulee yksilön osaamiseksi.

Tiedon luominen koostuu kolmesta elementistä:

- SECI- prosessi, tiedon luominen tiodon muuttuessa hiljaisesta tiedosta käsitteelliseksi tiedoksi.
- Ba, mikä on tiedon viitekehys
- Tietämyksen kehittymisen tekijät: lähtötieto, tulostieto ja käsittelijät.

Ba-viitekehyksessä hiljainen tieto muuntuu ja vahvistuu tiedon muuttuessa läpi sosialisaation, ulkoistamisen, yhdistämisen ja sisäistämisen. (Nonaka, Toyama & Konno, 2000.)

4.2 Uuden tuotteen kehittämisen peli

Nonakan ja Takeuchin panos ohjelmistojen kehityksen kehittämiseksi alkoi siis jo melkein vuosikymmen ennen heidän varsinaista läpimurtoteoriaa organisatorisen tiedon tutkimuksen saralla. Vuonna 1986 Takeuchi ja Nonaka osoittivat kuusi ominaisuutta jotka pitäisi huomioida ohjelmiston kehitys organisaatiossa:

- Sisään rakennettu epästabiilisuus.
- Itseohjautuvat projektitiimit
- päällekkäiset kehitysvaiheet

- Moni-oppiminen
- Hienovarainen ohjaus
- Organisatorinen opitun siirto.

Sisään rakennettu epävakaus on elementti, joka inspiroi kokeilemaan uutta ja luomaan uusia ideoita ja innovaatioita. Tätä vielä vahvistavat itseohjautuvat projektitiimit, jotka saavat annettujen rajojen sisällä vapaasti toteuttaa ideansa. Projektitiimien ohjaus pitää olla tämän takia hyvin hienovarainen. (Takeuchi & Nonaka, 1986.)

4.3 XP perusteet

Tutkimuksessa Kähkönen ja Abrahamsson (2003) on luotu teoreettinen malli organisaationalisen tiedon luomiselle ketterissä kehitysmenetelmissä. Tutkimus keskittyi lähinnä XP-ohjelmistonkehitysmenetelmään. Tutkimuksen mukaan seuraavissa ketterien ohjelmiston kehitysmenetelmien käytänteissä jalostettiin tietoa muodosta toiseen:

- Suunnittelupeli (planning game)
- Jatkuvat pienet julkaisut
- Yksinkertainen rakenne
- Piikki (spike)
- Pari ohjelmointi
- Refaktorointi
- Metaphor
- Yhteinen avoin työtila
- Ohjelmointistandardit
- Testaus
- Projektin sisäinen asiakas.

Samat käytänteet löytyvät myös muista ketteristä ohjelmankehitysmenetelmistä, ehkä hiukan eri nimellä ja voidaan tiivistää esimerkiksi edellä selitettyyn malliin (Takeuchi & Nonaka, 1986).

4.4 Ketterän manifestin periaatteet

Tämä tutkimuksen analyysi pohjautuu yleisiin ketterien ohjelmiston kehitysmenetelmien takana oleviin ketterän manifestin julistuksiin. Näiden julistusten takana olevat periaatteet voidaan johtaa edellä esitetyistä teorioista. (Mateos-Garcia & Sapsed, 2008.)

Mateos-Garsia ja Sapsed tiivistävät tiedon johtamiseen liittyvät tekijät neljään eri luokkaan.

- Nopeat prototyyppi-iteraatio syklit
- Huomio asiakkaaseen ja rehelliseen kommunikaatioon
- Sisäinen kommunikaatio ja konfliktien ratkaisu
- Itseorganisoituvat tiimit.

Nopeat prototyyppi-iteraatiosyklit ovat perusta ketterille menetelmile. Ketterät ohjelmistonkehitysmenetelmät velvoittavat toteuttamaan toimivia prototyyppejä ohjelmiston ominaisuuksista, jotka voidaan integroida, testata välittömästi nopeiden kehitysiteraatioiden avulla. Ottamalla tämä käytänne käyttöön voidaan jo hyvin varhaisessa kehitysvaiheessa havaita virheitä ja puutteita. Myös ihan uusia mahdollisuuksia on mahdollista löytää, kun prototyyppien avulla voidaan myös asiakasorganisaatiolle opettaa uusia to-

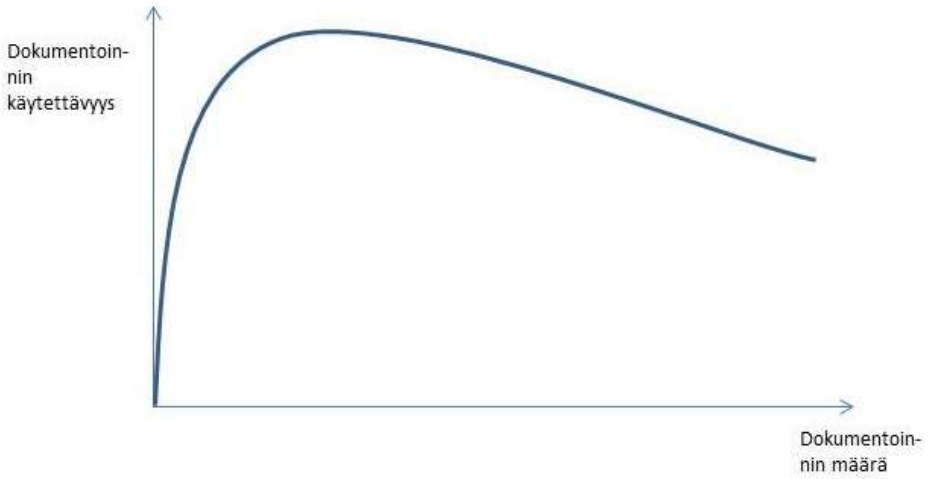
teutusmahdollisuuksia. Yksikehäinen oppiminen (single-loop learnig) kuten Argyris ja Schön (1996) esittävät, mahdollistuu tämän ominaisuuden avulla.

Sisäinen kommunikaatio ja konfliktien ratkaisu on elementti, jota ei ole runsaasti tutkittu. Ketterä ohjelmistonkehitys tehdään yleensä pienissä monitaitoisissa tiimeissä. Tiimien sisäinen kommunikaatio hoidetaan siten, että hiljaisen tiedon siirtymistä voisi tapahtua. Esimerkiksi Scrum käyttää päivittäisiä lyhyitä palavereita, jotta jokainen olisi tietoinen siitä, missä ollaan menossa. XP hyödyntää pariohjelmointia. Nämä yhdessä yhteisen avoimen työskentelytilan kanssa edistävät hiljaisen tiedon siirtymistä. (Abrahamsson, Salo, Ronkainen, & Warsta, 2002; Nonaka & Takeuchi, 1995; Schwaber, 2009.)

Itseorganisoituvattiimit mahdollistavat uusien ideoiden ottamisen käyttöön. Ketterät ohjelmistonkehitysmenetelmät korostavat voimakkaasti tiimin sisäistä tasa-arvoa. Esimerkiksi Scrum-tiimissä on vain yksi jäsen, jolla on formaali tehtävä. Häntä kutsutaan Scrum-masteriksi. Scrum-masterinkin rooli on lähinnä pitää huoli, että scrummin käytänteitä noudatetaan. Tiimin jokaisen jäsen voi siis itse valita tehtävänsä. Toinen tekijä, joka lisää tiimin sisäistä tiedon kulkua ja siten tiimin oppimista on se, että kuka tahansa tiimin jäsen voi ruveta rakentamaan mitä tahansa osaa ohjelmistosta. Tämä taas pakottaa tiimin kommunikoimaan. Ketterät ohjelmiston kehitys menetelmäthän eivät suosineet paperi dokumentteja vaan suullista tiedon välittämistä. Samalla henkilökohtaisen kommunikoinnin yhteydessä voidaan olettaa tapahtuvan myös hiljaisen tiedon siirtymistä. (Abrahamsson et al., 2002; Nerur & Balijepally, 2007; Nerur et al., 2005; Nonaka & Takeuchi, 1995; Schwaber, 2009.)

5. Ketterä dokumentointi: Rüpingin tulkinta

Andreas Rüping 2005 on kuvannut kirjassaan oman tulkintansa siitä mihin ohjelmiston dokumentoinnissa olisi keskityttävä. Vaikka kirjan nimi viittaa ketterän manifestin näkökantaan kevyestä dokumentoinnista, ovat kirjan ohjeet (pattern) päteviä tuotettaessa minkälaista dokumentointia tahansa. Tässä luvussa esitellään keskeiset Rüpingin esittämät ohjeet.



Kuva 4. Dokumentoinnin käytettävyys suhteessa määrään.

Rüpingin mielestä liika dokumentointi ei tuo dokumentointiin enää lisää arvoa, vaan dokumentoinnin käytettävyys alkaa huonontua. Dokumentoinnin laatu on hänen mielestään paljon tärkeämpää kuin määrä. On paljon parempi dokumentoida kevyesti, tarkasti, organisoidusti ja siten että tieto on päivän tasalla. Rüpingin mielestä se että ketterissä korostetaan kevyttä dokumentointia, ei missään mielessä tarkoita sitä että dokumentoinnin laadusta voidaan tinkiä.

Rüping esittelee kirjassaan 50 ohjetta dokumentointiin. Seuraavassa esitellään lyhyet perusteet niistä tärkeimmille.

5.1 Dokumentoinnin sisällön kohdentaminen

Rüping esittää kirjassaan mielipiteen, että jos dokumentin halutaan olevan hyödyllinen, dokumentin sisällön on vastattava lukijan odotuksia.

Kohdennetaanko dokumentit oikealle lukijalle

Projektien dokumentointi kohdistuu monille erilaisille lukijoille: Projektin johto, arkkitehdit, suunnittelijat, ohjelmoijat ja käyttäjät. Eri lukijaryhmillä on erilaiset tarpeet ja erilaiset taustat. Dokumentointia tehtäessä pitäisi miettiä kuka dokumenttia tarvitsee. Mikäli dokumentti ei vastaa lukijan tarpeita, ei hän sitä todennäköisesti lue eikä käytä.

Jokaiselle dokumentille pitää määritellä lukijakunta ja dokumentti pitää kirjoittaa lukijakunta mielessä.

Kohdennettu tietosisältö

Ohjelmiston dokumentointi sisältää tietoa moninaisista asioista, mikä on tyypillisesti kirjattu ylös useisiin dokumentteihin. Mitä tietoa pitäisi kirjata mihinkin dokumenttiin ja kuinka pitkä dokumentin pitäisi olla. Jokaisella dokumentilla pitäisi olla selvä ja määritelty sisältö, joka tekee dokumentista eheän ja suoraviivaisen.

Projektikohtaiset dokumentoinnin vaatimukset

Ohjelmistoprojektit eivät ole toistensa kaltaisia. Pieni projektiryhmä, joka työskentelee samassa tilassa, voi pärjätä ilman dokumentointia. Toisaalta iso useammalla paikkakunnalla toimiva projekti tarvitsee toimiakseen dokumentointia. Ehkä myös monimutkainen design tai sidosryhmien vaateet aiheuttavat dokumentoinnille tarvetta.

Jokaisen projektin pitäisi määritellä projektikohtaiset dokumentointivaatimukset.

Onko dokumentointiportfolio mietitty kunnolla

Yleisen standardidokumentointiprosessin laatimiselle ei ole tarvetta, sillä projektit poikkeavat toisistaan liikaa. Ohjelmistoprojekteilla on kuitenkin monia asioita yhteisiä joten jotain standardisointia on järkevää olla olemassa. Organisaation dokumentaatioportfolio määrittelee mitkä dokumentit voivat olla tarpeellisia ohjelmistoprojektille. Mikäli organisaatiolla on dokumentointiportfolio, projektit voivat valita mitkä dokumentit projekti katsoo tarpeelliseksi.

Onko keskitytty pitkänajan tietoon

Ketterissä ohjelmistonkehitysmenetelmissä korostetaan suullista yhteydenpitoa. Dokumentoinnilla on kuitenkin merkityksensä. Dokumentoinnilla mikä keskittyy tarjoamaan tietoa pitkän aikavälin näkökulmassa, jatkokehitysprojekteille ja ylläpitoon on huomattava merkitys.

5.2 Asioiden taustojen ilmaiseminen

Dokumentointiin olisi liitettävä tieto siitä, kuinka luotettavaa ja pystyvää dokumentissa esitetty tieto on.

Tehdäänkö määrittely yhteistyössä

Jokainen ohjelmistonkehitysprojekti vaatii määrittelyn, joka on tehty yhteistyössä projektiryhmän ja asiakkaan kanssa. Näin voidaan varmistaa että projektin suunta on se minkä asiakas haluaa.

Ovatko suunnitteludokumentoinnissa taustat ilmastu, vaan vain ratkaisu

Mikäli suunnittelu dokumenteissa kerrotaan myös ratkaisuiden perusteet, voidaan luoda parempi pohja tulevaisuuden muutostarpeille.

Ovatko kuvaus ja arviointi erillään

Dokumentissa pitäisi erottaa mikä on olemassa olevan kuvausta ja mikä taas kirjoittajan objektiivista arviointia asiasta.

5.3 Vastuuhenkilö ja resursointi

Dokumentointi on osa projektia jolle on varattava resurssit ja jonka laatua on valvottava.

Onko dokumentointi erillinen tehtävä, jolle on varattu omat resurssit.

Dokumentoinnin tulisi olla oma itsenäinen tehtävä, jonka tekemiseen on varattava aikaa ja henkilöresursseja. Kun dokumentointi on oma itsenäinen tehtävä, sitä voidaan arvioida ja priorisoida suhteessa muihin projekti aktiviteetteihin.

Onko dokumentilla vain yksi vastuuhenkilö

Jos tehtävällä on useita vastuuhenkilöitä, niin tehtävä ei todennäköisesti valmistu koskaan. Projektin jokaisella dokumentilla tulisi olla yksi nimetty vastuuhenkilö. Tämän henkilön ei tarvitse olla dokumentin kirjoittaja, mutta hänen koordinoida muiden henkilöiden panosta dokumentin kirjoittamiseen.

Jatkuva dokumentointi

Kun projektin dokumentaatiota kehitetään kokoajan tasaisin välein, jolloin se vastaa projektin sen hetkistä tilaa. On hyvä käytäntö esimerkiksi päivittää ohjelmiston dokumentointi jokaisen julkaisun yhteydessä.

Kirjoitus ja peilaus

Jos halutaan saada tiimin jäsenistä irti paras panos dokumentoinnin suhteen, tulisi heille antaa aikaa, ei pelkästään varsinaiseen kirjoittamiseen, vaan myös sen tarkastelemiseen mitä dokumenttiin on kirjoitettu. Materiaalin kerääminen ja organisoiminen vaatii luovuutta. On melkein mahdotonta kirjoittaa täydellistä dokumenttia suoraan. Dokumentin kirjoittaminen vaatii useita kierroksia kirjoittamista ja evaluointia.

5.4 Palautteen saaminen

Dokumentoinnin kirjoittaminen on oppimisprosessi. Oppiminen vaatii palautetta.

Katselmointikulttuuri

Dokumentointi voi hyötyä paljon katselmoinneista, mikäli pystytään rakentamaan katselmointikulttuuri, jonka sekä dokumenteista vastuussa olevat, sekä katselmointeihin osallistuvat henkilöt, pitävät mukavana.

Katselmointi ennen toimitusta

Palautteen saaminen dokumentin kirjoittamisen aikana katselmointien kautta on aina eduksi, mutta katselmointi ennen virallista toimitusta on pakollinen. Katselmoinnissa tulisi tarkistella: Täyttääkö dokumentti tavoitteet ja onko sillä käyttöä lukijoille, onko dokumentti teknisesti tarkka ja oikea, ovatko muotokysymykset oikein ja onko dokumentti ymmärrettävä.

Asiakaskatselmointi

Asiakaskatselmoinnit voivat parantaa dokumentin laatua varsinkin mitä tulee sovellusalueen tietämykseen. Samalla voidaan parantaa yhteistoimintaa ja yhteishenkeä.

Puolueeton näkökulma

Dokumentoinnin vastuuhenkilöt voivat saada parhaiten puolueetonta palautetta, ihmisiltä joilla on yleistä tietämystä aiheesta, mutta eivät ole kuitenkaan mukana projektin toiminnassa.

Realistisia esimerkkejä

Tekniset dokumentit ovat joskus hyvinkin abstrakteja. Monet projektin henkilökunnasta eivät ole kovainkaan hyviä asiantuntijoita sovellusalueesta. Abstraktien asioiden ilmaisemiseen konkreettisella tavalla tarvitaan reaalimaailmasta otettuja realistisia esimerkkejä.

5.5 Kokonaiskuva, dokumentoinnista tiedottaminen

Dokumentoinnissa tulisi miettiä, sitä mitenkä uusi työntekijä pääsee projektin mukaan, sitä mitenkä projektin jäsenet ja sidosryhmät saavat tiedon valmistuvista dokumenteista ja siitä mitenkä dokumentteihin tallennettu tieto saadaan koko organisaation saataville.

Kokonaiskuvadokumentti (Bic Picture) , voiko ihmiset tutustua projektiin, ilman yksityiskohtia

Jokaiseen projektiin tulee silloin tällöin uusia työntekijöitä. Uusien projektin jäsenten sisäänajoon tarvitaan dokumentti, mistä saa hyvän yleiskuvan, ilman että joutuu tutustumaan suureen määrään teknisiä yksityiskohtia. Kokonaiskuvadokumentti kertoo yleiskuvan ohjelmiston arkkitehtuurista ja suunnittelu periaatteista.

Markkinointi

Dokumentit saavat enemmän huomiota, jos lukijoille, joille dokumentti on tarkoitettu, aktiivisesti kerrotaan dokumenttien ilmestymisestä ja kehotetaan tutustumaan.

Tulevaisuuden hyötyminen

Vain jos projektidokumentaatio on tehty saataville organisaation laajuisesti, voivat tulevat projektit hyötyä projektin opeista.

5.6 Yhteenveto

Tässä luvussa esiteltiin Rüpingin 50 ohjeesta valitut ohjeet oikeiden asioiden löytämiseksi dokumentteihin ja ohjeet, jotka liittyivät projektin hallintaan ja laadunvarmistukseen. Tutkimuksesta rajattiin pois liian yksityiskohtaisina, loput 30 ohjetta, joista 8 käsittelee yksittäisen dokumentin rakennetta, 8 ohjetta muotoilua ja 14 ohjetta dokumenttien hallintaa ja teknistä organisaatiota.

6. Tutkimuksen lähestymistapa

Tämä tutkimus oli tapaustutkimus joka toteutettiin laadullisena haastattelututkimuksena. Tässä luvussa kerrotaan ensin haastattelututkimuksen käsitteistöä ja lopuksi kerrotaan miten tutkimus on järjestetty.

6.1 Tapaustutkimus

Tapaustutkimus on tutkimusstrategia mikä keskittyy ymmärtämään yksittäisen tutkittavan kohteen sisällä olevia tapahtumia. (Eisenhard, 1998.) Tapaustutkimuksessa voidaan tarkastella yhtä tai useampaa tapausta. Tiedonhankinta tapoina ovat voivat olla esimerkiksi kyselyt, haastattelut, havainnointi ja arkistomateriaalin käyttö (Järvinen P. & Järvinen A., 1998). Eisenhard (1998) mukaan tapaustutkimus sopii hyvin tilanteeseen, joissa tarkasteltavana on uusi, vähän tutkittu kohdealue. Tapaustutkimuksen tavoitteena on ensisijaisesti mahdollisimman monipuolinen kokonaisuuden hahmottaminen (Järvempää & Kosonen,1996.) Tapaustutkimus vastaa kysymyksiin miten ja miksi (Järvinen P. & Järvinen A., 1998).

Aineisto voi olla laadullista tai määrällistä. Tapaustutkimuksessa on ensisijaisesti tavoitteena mahdollisimman monipuolinen kokonaisuuden hahmottaminen yksittäisestä tapauksesta. Tapaustutkimuksessa pyritään myös ymmärtämään tutkimuksen kohteena olevien henkilökohtaisia käsityksiä. (Järvempää & Kosonen,1996.)

Tapaustutkimuksen luotettavuus on suuri tutkittavan tapauksen sisällä, mutta tulosten yleistettävyys on vaikeaa. Tapaustutkimuksen yhteydessä puhutaan analyyttisestä yleistettävyydestä, jolla tarkoitetaan yleistyksiä, jotka pätevät yli tutkitun tapauksen. (Järvempää & Kosonen,1996.)

6.2 Laadullinen tutkimus

Laadullisen eli kvalitatiivisen tutkimuksen tarkoituksena on yleensä kokonaisvaltainen taustojen selvittäminen. Tutkimuksen lähtökohtana kokonaisvaltainen todellisen elämän kuvaaminen ja siinä huomioidaan tosiasia että elämä on moninainen. (Hirsjärvi, Remes, & Sajavaara, 2007.)

Laadullisessa tutkimuksessa suositaan ihmistä tiedon keruun instrumenttina. Tutkijan oma arvonäkökulma vaikuttaa tutkimuksen tuloksiin, koska arvot vaikuttavat siihen miten näemme tutkittavaa ilmiötä. Myös tutkimuksen kohteena olevat henkilöiden henkilökohtainen näkemys pääsee esille. (Hirsjärvi et al., 2007.)

Laadullisessa tutkimuksessa tehdään merkitystulkinta tuotettujen johtolankojen ja käytettävissä olevien vihjeiden pohjalta. Lähteet ovat aina osittain päällekkäisiä ja kertovat versioita samoista asioista, mutta silloinkin valottavat asian eri puolia (Alasuutari, 1999.)

Laadullisen tutkimuksen tyypillisiä piirteitä (Hirsjärvi et al., 2007):

- Kokonaisvaltaista tiedon hankintaa
- Suositaan ihmistä tiedon keruun instrumenttina

- Käytetään induktiivista analyysiä
- Laadullisten metodien käyttö aineiston hankinnassa, tutkittavien ääni pääsee esille
- Kohde joukko voidaan valita tarkoituksen mukaisesti
- Tutkimus suunnitelma muotoutuu tutkimuksen edetessä
- Käsitellään tapauksia ainutlaatuisina.

Laadullinen tutkimus ei ole synonyymi tulkitsevalle tutkimukselle, vaikka ne joskus yhdistetään samaksi käsitteeksi. Laadullinen tutkimus voi tai voi olla ettei ole tulkitsevaa tutkimusta riippuen tutkijan filosofisista olettamuksista. (Klein & Myers, 1999)

6.3 Teemahaastattelu

Eskola ja Vastamäki (2001) kertovat kirjassaan, että kun halutaan tietää, mitä joku ajattelee jostakin asiasta, kaikkein yksinkertaisinta ja usein tehokkainta on tietenkin kysyä sitä häneltä.

Teemahaastattelu on laadullisen tutkimuksen muoto. Kyseessä on eräänlainen keskustelu, joka tapahtuu tutkijan aloitteesta ja ohjauksessa. Keskustelu jossa tutkija pyrkii vuorovaikutuksessa saamaan haastateltavan ilmaisemaan tutkijaa kiinnostavat asiat. (Eskola & Vastamäki, 2001.)

Teemahaastatteluilla on muun muassa seuraavat ominaisuudet:

- Haastattelun avulla saadaan kuvaavia esimerkkejä alueille (Sanford, 1966)
- Haastattelu sopii lomaketta paremmin emotionaalisille ja intiimeille alueille (Sanford, 1966)
- Haastattelussa on suuremmat mahdollisuudet motivoida henkilöitä kun lomaketutkimuksessa.
- Haastateltavalla on haastattelussa enemmän mahdollisuuksia tulkita kysymyksiä, se on menetelmänä joustavampi ja mahdollistaa täsmennykset. (Hirsjärvi & Hurme, 1995)

Haastattelua suunniteltaessa tutkijan on päätettävä millaisia päätelmiä hän aikoo aineistostaan tehdä ja haastattelun suunnitteluun kuuluu myös kannanotto hypoteesien muodostamiseen. Haastattelurunko ei saisi olla liian tarkka vaan sen tulisi olla enemmänkin teema-alueluettelo. (Hirsjärvi & Hurme, 1995.)

6.4 Haastattelujen toteutus

Tutkimus toteutettiin tapaustutkimuksena. Tutkimukseen saatiin haastateltavaksi kolmen organisaation edustajat. Haastateltavat olivat kaikki pienten projektien tai tiimien vetäjiä ja olivat toimineet organisaatiossa jo ennen organisaation siirtymistä käyttämään ketteriä ohjelmistonkehitysmenetelmiä.

Haastattelut toteutettiin teemahaastatteluina. Haastatteluissa, noin tunnin kukin, pyrittiin ensin kartoittamaan kyseisen haastateltavan rooli organisaatiossa, se mitenkä pitkällä ketterien menetelmien käyttöönotto yrityksessä on ja miten yleisesti koetaan siirtyminen ketteriin menetelmiin.

Haastateltavan annettiin aika vapaasti kertoa organisaationsa toiminnasta. Keskustelua hiukan kuitenkin ohjattiin välikysymyksillä pysymään dokumentoinnin näkökulmassa. Lopuksi käytiin kirjan Rüping (2005) ohjeet läpi ja niiden käyttäminen.

Tutkimus toteutettiin haastattelemalla kolmea henkilöä kolmesta yrityksestä. Haastateltavat pyysivät, ettei heidän henkilöllisyyttään ja organisaatiota avata tutkimuksessa enempää kuin on tarpeen.

Monet yritykset ovat hyvin varovaisia kertoessaan sisäisestä toiminnastaan eikä tarkempi haastateltavien yksilöinti ole tutkimuksen tuloksen kannalta tärkeä. Tämän johdosta haastateltavat ja yritykset ovat esitelty tässä vain ominaisuuksien ja roolien avulla.

Haastattelut suoritettiin syksyn 2010 ja kevään 2011 aikana. Ensimmäisessä vaiheessa kyseltiin haastateltavasta taustatietoja, minkälainen ketterä menetelmä heillä on käytössä. Ja kuinka pitkällä he ovat ketteriin menetelmiin siirtymisessä, teemat 1-3. Sitten keskustelussa pureuduttiin tarkemmin muutoksiin mitä ketteriin menetelmiin siirtyminen on aiheuttanut, teemat 4-6. Viimeiseksi keskusteltiin valikoitujen Rüpingin (2005) ohjeiden käytöstä. Haastattelurunko on liitteessä 1.

6.5 Tutkimusaineisto

Tutkimukseen valittiin haastateltavaksi kolme yritystä Oulun alueelta joiden ennakko tietojen mukaan tiedettiin siirtyneen käyttämään ketteriä ohjelmistonkehitysmenetelmiä noin viisi vuotta sitten.

Tutkimukseen luonnollisesti valikoitui iso Oulun alueella toimiva matkapuhelinjätti, samoin kuin kyseisen konserniin kuuluva verkkolaitteita valmistava yritys. Haastateltavista kaksi edusti puhdasoppista toteutustiimiä, joiden tarkoituksena oli siis tuottaa toimiva järjestelmän osa ja yksi haastateltavista oli testausryhmän edustaja.

Haastateltavat henkilöt ovat olleet kyseisen yrityksen palveluksessa jo ennen ketterien menetelmien käyttöönottoa, joten he voivat vertailla toimintaa ennen ja jälkeen ketteriin ohjelmistonkehitysmenetelmiin siirtymisen. Keitä haastateltavat ovat ja mikä on heidän tarkka osastonsa tai toimenkuvansa ei ole tutkimuksen tuloksen takia merkittävä, joten luottamuksellisuussyistä ei heitä tässä esitellä tarkemmin.

Yritys1, suuri matkapuhelinjätti

Kuten edellä tuli esille ei Oulun alueella toimivaa suurta matkapuhelin jättiä voi jättää minkään tietotekniikkasektoria koskevan tutkimuksen ulkopuolelle. Yrityksestä haastateltavana oleva oli vuonna 2010 valmistunut ja toiminut yrityksessä haastatteluhetkellä 7 vuotta laatuun ja testaukseen liittyvissä tehtävissä. Haastateltava toimi haastattelu hetkellä Testaustiimin johtotehtävissä.

Yritys 2, suuri verkkolaitetoimittaja

Toiseksi yritykseksi valikoitui suuri matkapuhelinverkkoihin laitteita valmistava yritys ja sieltä lähinnä tukeasemille käyttöliittymiä tekevä organisaatio. Haastateltavana oli vajaa 10 vuotta ohjelmisto projektin vetotehtävissä toiminut henkilö. Haastatteluhetken rooli oli tuoteomistaja.

Yritys 3, Keskikokoinen suomalainen it-konsultti/alihankkija yritys

Yritys toimii alihankinta organisaationa operatiiviselle organisaatiolle. Yrityksellä on omaa ketteriin ohjelmiston kehitysmenetelmin liittyvää osaamista, ja esimerkiksi haastateltava oli authorisoitu scrummaster (Scrum Master). Lisäksi he tekivät konsultointi työtä ja kehittivät asiakkaan Scrum prosesseja.

7. Haastattelujen tulkinta

Miten ketterien menetelmien lähinnä Scrummin käyttöönotto on vaikuttanut dokumentointikäytäntöihin ja dokumentointiin. Asiaa peilattiin lähinnä aikaisemmin esitettyyn Andreas Rüpingin (2005) ohjeiden avulla.

7.1 Yritys 1

Haastattelu kohdistui haastateltavan testaustiimin toimintaan vaikkakin osa saaduista kommenteista peilasi myös laajemmin kyseisen yrityksen toimintaa.

Yritys oli voimakkaasti siirtymässä ketteriin ohjelmistonkehitysmenetelmien käyttöön. Menetelmiä oli yrityksessä otettu käyttöön noin 2-3 vuoden ajan. Haastateltava kuvaili käytettävän menetelmän olevan Scrum. Kun keskusteltiin menetelmästä tarkemmin, niin Scrum merkitsi heille lähinnä tihentynyttä releasesykliä. Haastateltava kertoi yrityksessä olevan useita Scrum-tiimeja joilla kaikilla oli omat backloginsa. Varsinaista Scrum:n tuoteomistajaa ei ollut, vaan sitä roolia hoiti asiakas projektien päälliköistä koostuva tiimi. Yrityksen Scrum sovellutuksessa tuoteomistajan roolia vastas asiakasprojektien päälliköistä koostuva tiimi. Vaatimukset koostettiin Scrum-tiimien backlogeihin. Siirtyminen ketteriin menetelmiin oli jo vakiintunut, mutta prosessia kuitenkin pyrittiin parantamaan ja tehokkuutta lisäämään. Testaustiimin työtä myös pyrittiin automatisoimaan. Haastateltavan mukaan heillä oli käytössä systeemejä, joiden avulla osa testaussuunnitteludokumentoinnista voitiin toteuttaa jo koodaus vaiheessa lisäämällä koodiin merkkejä joiden avulla voitiin testitapauksia luoda automaattisesti.

Pääosa dokumentoinnista haastateltavan mielestä oli lähinnä testausraportteja ja muita hallinnollisia dokumentteja. Tähän vastaukseen saattoi vaikuttaan merkittävästi haastateltavan rooli testausryhmän vetäjänä. Testausryhmän tehtävä vaati laatimaan huomattavan määrän erilaisia testausraportteja ja metriikoita, joilla pyrittiin osoittamaan laatutaso.

Jo ennen siirtymistä Scrummiin oli haastateltavan mukaan yrityksessä suunta turhan dokumentoinnin poistamiseksi. Automatisointi oli myös muuttanut dokumentointia siten että työkalut tuottivat dokumenteista suuremman osan automaattisesti. Virtuaalitiimit toivat väkisinkin omat vaatimuksensa dokumentointiin ja sisäisen dokumentoinnin tarve oli jopa kasvanut. Dokumentoinnissa sisältö oli kuitenkin muodostunut tärkeämmäksi. Lisäksi epäformaali kommunikointi oli vilkastunut. Dokumentointi oli vaan yksi ”item” backlogissa.

Haastateltava kertoi, että dokumentointikäytänteet olivat hyvin vakiintuneita ja että ”Devaajat” tietävät mitä on tulossa, ja mitä menossa. Tehtävien organisointi ja jakaminen osiin oli auttanut myös dokumentoinnin suunnittelussa.

Haastateltavan näkemys, siitä että kohdennetaanko dokumentit oikealle lukijalle, oli että hänen mielestään siinä onnistutaan kohtuullisen hyvin. Dokumenttien tietosisältöä pidettiin kohtuullisen oikeana. Testaustiimistä kun oli kyse, niin heidän näkökulmastaan oli hyvä, että vaatimukset katselmoitiin testauksen kannalta.

Toisaalta vaikka he eivät varsinaisesti tehneet WORD-dokumentaatiota, niin erilaisia raportteja tehtiin ”devaajien” mielestä liikaa. Haastateltava sanoi aika suoraan, että dokumentaatiota ei ole oikeastaan suunniteltu, vaan niitä tehdään, koska joku pyytää. Dokumentoinnin hyödyntäminen ei ollut tekijän tiedossa.

Haastateltavan mukaan testaustiimin tekemä dokumentointi oli pääsääntöisesti testiraportteja, ja muita tilanneyhteenvetoja. Arvioitaessa, miten tiimi huomioi keskittymisen pitkänajan tietoon, niin haastateltavan näkemyksen mukaan heillä ei siihen ollut tarvetta. Sen sijaan tiimi teki metriikkaa, mutta ei tiedetty miten sitä tullaan käyttämään. Dokumentointi tehtiin pääasiassa yhteistyössä, ja siinä apuna oli myös jokin verran automatiikkaa.

Haastateltava kertoi, että vaikka hänen oma tiiminsä ei varsinaisesti suunniteltudokumentaatiota tehnytkään, että suunnitteludokumentoinnissa on lähinnä tietoa mitä laitteen pitäisi tehdä, ei miksi. Joidenkin asioiden perustelut ja taustat jouduttiin hakemaan koodidokumentoinnista ja jopa koodi kommentoinnista.

Kokonaiskuvadokumentaatiota, minkä avulla voitaisiin helposti ajaa projektiin uusia ihmisiä sisään, ei ollut tehty, vaan uudet projektin jäsenet joutuvat jonkin verran tutkimaan projektin muuta dokumentaatiota. Vaihtuvuus oli kuitenkin vähäistä, joten tämä ei ole ollut ongelma. Wiki-dokumentaatiota kirjoitettiin yleisimmistä kysymyksistä.

Haastateltavan mukaan dokumentoinnin resurssiallokointi oli hoidettu ottamalla dokumentointi ”sprint-itemeiksi”. Dokumentoinnin jatkuva ylläpito ontui haastateltavan mielestä: ”Tietysti tärkeimpiä ylläpidetään, kuten rajapintadokumentit”. Joitakin asioita ylläpidetään vain kevyesti projekti-wikissä. Dokumentoinnissa voi joskus olla Faktat ja omat mielipiteet sekaisin ja joskus omat mielipiteet esitetään faktoina.

Dokumentointi katselmoitiin, jotkin asiat epäformaalisti, mutta ”aina pitää katselmoida”. Myös koodi katselmoidaan. Ennen toimitusta pidetään tarkka ”review”. ”Kieli kohdilleen”. Rüpingin esittämää Dokumentaation markkinointia ei olut, vaan oletettiin että tarvittava tieto löytyy vanhojen dokumentointitapojen avulla.

7.2 Yritys 2

Haastateltava edusti organisaatiota, jonka tehtävänä oli tuottaa hallintaohjelmistoja, joilla tukiasemia testataan ja niille syötetään peruskonfiguraatiotiedot. Organisaatiolla oli voimakas käytettävyysnäkökulma. Varsinainen ohjelmiston toteutustyö tehtiin ”off shore”, mutta Oulussa oli voimakas käytettävyyssuunnittelun osaaminen. Haastateltavan oma projekti ja siihen liittyvä tuote oli jo ylläpitovaiheessa.

Haastattelun kohteena olevalla organisaatiolla oli haastateltavan mukaan haasteena alihankinta ja offshore työskentely. ”Devaaja-tiimit” sijaitsivat pääsääntöisesti ”off shore” (Kiina) , kun taas määrittely ja spesifikaatiotyö tehtiin Oulussa. Tämän johdosta haastateltavan mukaan dokumentointia oli pakko tehdä. Varsinkin Kiina nosti dokumentoinnin vaatimusta. Työskentely kiinalaisten kanssa vaati, että kaikki piti dokumentoida hyvin tarkasti.

Haastateltava kertoi, että yrityksessä oltiin voimakkaasti kehittämässä ketterien kehitysmenetelmien käyttöä mutta siirtymävaihe alkoi olla jo ohi. Ketterien ohjelmistonkehitysmenetelmien käyttöä eri muodoissaan organisaatiossa oli kokeiltu parin- kolmen vuoden ajan.

Haastateltava kertoi, että Scrummin käyttö kohdistui kuitenkin lähinnä ”devaajiin”, ja voimakkaimmin näkyvä elementti oli haastateltavan mukaan päiväpalaverit ja nopeat integrointi syklit. Merkittävä osa suunnittelutyöstä ja suunnitteludokumenteista tehtiin scrummin ulkopuolella. Haastateltava kertoi, että heille vaatimusmäärittely tulee ”korkeammalta taholta”.

Suurin osa tehtävistä dokumenteista olivat käyttöliittymädokumentteja. Teknisiä luokkatason dokumentteja organisaatio teki vähemmän. Feedbackiä toteutuksesta ei ole, tiimi oli tuotteen ylläpitovaiheessa joten kyseessä oli paljolti vain vanhan päälle tekemistä. Uusia ideoita ei kokeilla.

Haastateltavan mukaan hänen organisaationsa ei tehnyt turhia dokumentteja, niiden tietosisältö oli oikea ja ne oli kohdennettu oikealle lukijalle. Dokumentointiportfolio oli ajansaatossa muotoutunut kohdilleen ja haastateltava piti sitä toimivana. Dokumentointi keskittyi pitkänajantietoon.

Dokumentointi on pääsääntöisesti yhden suunnittelijan aikaansaannos. Katselmoinneissa on mukana myös dokumentoinnin käyttäjiä. Dokumentoinnin tekee pääsääntöisesti erillinen tiimi, dokumenteilla on yksi vastuuhenkilö, joka yleensä oli myös dokumentin kirjoittaja, ja niiden ylläpito on jatkuvaa. Kirjoitettuja dokumentteja peilataan jatkuvasti toteutukseen.

Haastattelussa kävi ilmi, että dokumentoinnille oli tyypillistä, että dokumenteissa esitetään vain valitut ratkaisut. Joistain hyvin monimutkaisista asioista tehdään ”studyjä”. Käyttöliittymä dokumentointi sisälsi hyvin tarkkoja realistisia esimerkkejä ja näyttökuvia, joiden avulla voitiin tarkasti hahmottaa mitä haluttiin saada aikaiseksi.

Haastateltava kertoi että heidän prosessien mukaan heillä oli katselmointi aina ennen toimitusta. Dokumentointia ei varsinaisesti markkinoida, mutta dokumentoinnissa on vakio dokumentit eri asioille, joten se helpottaa tiedon löytymistä.

Dokumentoinnissa on huomioitu se että dokumentointia uudelleen käytetään seuraavissa julkaisussa ja jopa uusissa tuotteissa.

Projektiin pääsemistä varten ei ole esittelymateriaalia. Toisaalta projektiin ei ole otettu viimeaikoina uutta henkilökuntaa.

7.3 Yritys 3

Haasteltava edusti organisaatiota, jonka yksi tuote oli ketterien menetelmien ja Scrumin kouluttaminen ja käytön edistäminen. Organisaatio myi ohjelmistoalihankintaa, missä merkittävänä osana oli Scrum-osaaminen. Organisaatiolla oli myös omaa ohjelmistokehitystä ja omia tuotteita. Haastattelussa pyydettiin keskittymään oman organisaation oma ohjelmistokehityksen menetelmien kertomiseen, mutta vastauksissa oli silti paljon vaikutteita asiakkaille tehtävistä projekteista.

Haastattelun kohteena oleva organisaatio kehitti Scrum-prosessia. Prosessi ja Scrum-osaaminen olivat yritykselle myyntituote. Organisaatiolla oli merkittävästi muodollista kouluttautumista ketteriin ohjelmistonkehitysmenetelmiin ja Scrummiin. Toimintatavat ja prosessi kuitenkin vaihtelivat asiakasprojekteittain riippuen asiakkaasta.

Yrityksen oman Scrum-mallin sprintin pituus oli 2 viikkoa. Jokaisen Sprintin jälkeen ”freesataan, katsotaan missä mennään ja sprint planning”. Se, mitä nopeasti haastateltava selitti, vastasi sitä, mitä luvussa 2.4 on Scrum-menetelmästä kerrottu. Vaikka orga-

nisaatio oli varsin kypsä Scrumin käyttäjä, niin organisaatiossa vielä haettiin uusia uria ja kehitettiin omaa prosessiaan.

Haastateltavan mukaan tekniseen dokumentointiin oli pohjia. Asiakasdokumentointi oli yleensä määritelty tuotteen kehitysjonossa. Haastateltavan mielestä dokumentointia ei ollut suunniteltu riittävän hyvin eivätkä tehdyt dokumentit kohdistuneet pääpiirteissään oikealle lukijalle ja dokumenttien ”träkkääminen” ontuu.

Haastateltavna mielestä organisaatiossa tehtiin paljon turhaa dokumentointia eikä dokumentointiportfoliota ollut mietitty kunnolla. Dokumenttien sisältö oli kuitenkin tiivis ja oikea. Kokonaissuunnitelmien puute haittasi dokumentoinnin kohdistamista pitkän ajan tietoon. Haastateltavan mukaan softan oletettu elinkaari vaikutti tarpeeseen kohdistaa dokumentointia pitkänajan tietoon.

Suunnittelupäätösten taustoja oli kirjattu esimerkiksi ”user storeista”. Mutta taustojen kirjaamisessa oli myös pahoja puutteita. Joissakin dokumenteissa saatettiin viitata vanhoihin dokumentteihin, mitkä eivät välttämättä olleet saatavilla, tai ainakaan eivät olleet kyseisen projektin näkökulmasta päivän tasalla.

Haastateltavan mukaan uusien henkilöiden sisään ajoa ei ole huomioitu tekemällä kokonaiskuvadokumenttia, vaan tieto pitää hakea useista dokumenteista tai useilta henkilöiltä.

Se mikä on dokumentin kirjoittajan näkemys asiaan, on joskus hiukan sekaisin sen kanssa mikä on totuus.

Dokumenteilla on yksi vastuuhenkilö, mutta dokumentin kirjoitus on tiimityötä. Dokumentoinnilla oli erillinen tekninen kirjoittaja, mutta nyt kehittäjät hoitavat kaiken dokumentoinnin. Kehittäjiä on hankala saada ymmärtämään ja motivoitumaan. Dokumentointi on jatkuvaa ja iteratiivista ja katselmointi käytännöt olivat käytössä.

Dokumentointia myös markkinoitiin, joten tieto siitä mitä on dokumentoitu on leviää.

8. Pohdinta

Tutkimuksella haluttiin selvittää mitenkä yritykset Oulun alueella olivat räätälöidessään ketteriä ohjelmistonkehitysmenetelmiä käyttöönsä, huomioineet tietämyksen johtamisenteorioista ketteriin menetelmiin kehittyneet piirteet, olivatko yritykset siirtyneet luottamaan suulliseen kommunikointiin ja oliko yritysten dokumentoinnin tarve muuttunut.

Tutkimus oli tapaustutkimus, jossa kohteena oli Oulun alueella toimivat ohjelmistoyritykset, joiden tiedettiin harrastavan ohjelmiston kehitystä ketterien menetelmien avulla. Tutkituissa firmoissa oli ollut Scrum käytössä tutkimushetkellä kahdesta kolmeen vuotta. Ensimmäiset kaksi organisaatiota olivat jo stabiloineet kehitysprosessinsa. Mutta kolmannen firman edustaja, joka myös konsultoi Scrummia, kertoi, että he yhä etsivät parempia tapoja kehittää ohjelmistoja Scrumin avulla.

Prosessit ovat ohjelmistoyrityksille yleensä hyvin yksityisiä, niin tälläkin kertaa. Yritysten käyttämistä menettely tavoista ei siis ollut saatavissa etukäteistietoa. Teemahaastattelu toimii parhaiten tällaisessa tilanteessa. Teemahaastattelu antoi myös mahdollisuuden tulkita kysymyksiä.(Hirsjärvi & Hurme, 1995.)

Ensimmäinen haastattelun kohde ei ollut varsinainen ohjelmistonkehitys tiimi, vaan testausryhmä, jonka rooliin ei kuulunut varsinaisesti ohjelmistodokumenttien teko. Scrum oli voimakkaasti sovellettu. Product tuoteomistajan roolia hoiti asiakasprojektien päälliköistä koostuva tiimi. Varsinainen loppukäyttäjä oli etäinen. Moni tutkimus on kuitenkin todennut, että ketterien ohjelmiston kehitysmenetelmien käytöstä saadaan paras hyöty, mikäli asiakas on läsnä jokapäiväisessä kehitystyössä. (Martin et al., 2004; Turk et al., 2005.)

Kahden ensimmäisen haastatellun organisaation edustajat kertoivat, että heillä tuotekehitys muodostuu virtuaalitiimeistä, mikä aiheuttaa ongelmia, mikäli käytetään oppikirjamaista lähentymistapaa Scrummiin. Molemmissa organisaatioissa perusteltiin virtuaalitiimien käytöllä dokumentoinnin tärkeyttä. Vaatimukset tulivat ulkopuolelta. Toisena haastattelun kohteena oli organisaatio, joka tuotti hallintaohjelmistoja tukiasemia varten. Haastateltava kertoi että heidän organisaationsa oli hajautettu ja pääosa koodauksesta tehdään ulkomailla – Kiinassa.

Braithwaite ja Joyce (2005) toteaakin juuri näin, että hajautetussa ympäristössä ei ketterien ohjelmistonkehitysprosessien kommunikointimenetelmät toimi. Mutta Baitwaite (2005) esittää toisaalta malleja, joiden avulla hajautettu ketterä ohjelmistonkehitys voidaan toteuttaa. Ramesh (2006) nostaa tutkimuksessa hajautetun ohjelmistonkehityksen ja ketterän ohjelmistonkehityksen yhdistämisen hyvin haasteelliseksi toiminnaksi.

Toista yritystä edustava haastateltava kertoi, että vaatimukset heidän tuotteilleen tulee valmiina ulkopuolelta ja että sen lisäksi pääosa suunniteltutyöstä tehdään perinteisesti Scrum-tiimin ulkopuolella. Herää kysymys onko kehitysprosessi edes ketterä, jos dokumentointi tehdään scrummin ulkopuolella, palautetta ei saada, ja vaatimukset tulevat ulkopuolelta (dokumenttina) katso Nerur & Balijepally (2007).

Firma 3 kehitti Scrum-prosesseja. Heidän Scrum oli huomattavan oppikirjamaisempi (Schwaber, 2002) kuin ensimmäisillä kahdella haastattelun kohteena olleena organisaatiolla.

Vaikka ketteriin ohjelmistonkehitysmenetelmiin liittyy myytti, ettei niissä tehdä dokumentaatiota, niin todellisuudessa dokumentaatiota pitää ohjelmistosta toteuttaa, jotta voidaan varmistaa ylläpito (de Souza, Anquetil, & de Oliveira, 2005). Myös asiakkaalle ja eri sidosryhmille tarvitaan dokumentaatiota projektin edistymisestä (Schwaber, 2009).

8.1 Dokumentoinnin sisällön kohdentaminen

"Tehokkain ja toimivin tapa tiedon välittämiseksi kehitystiimille ja tiimin jäsenten kesken on kasvokkain käytävä keskustelu." (Agile Alliance, 2001).

Rüpingin mukaan liika dokumentointi ei tuo lisäarvoa. Dokumentointi pitäisi suunnitella siten että keskitytään tietoon mitä tarvitaan kuukausien tai vuosien päästä, ja pyritään välttämään dokumentoimasta asioita, jotka vanhenevat nopeasti. Dokumentointi pitäisi kohdentaa oikealle lukijalle ja dokumentoinnin asiasisältö pitäisi olla huolellisesti suunniteltu, siten että oikea tieto on nopeasti löydettävissä. (Rüping, 2005.)

Rüping (2005) esittää seuraavat ohjeet:

- Kohdennetaanko dokumentit oikealle lukijalle
- Kohdennettu tietosisältö
- Projektikohtaiset dokumentoinnin vaatimukset
- Onko dokumentointiportfolio mietitty kunnolla
- Onko keskitytty pitkänajantietoon.

Kaksi haastatelluista yrityksistä piti dokumentoinnin kohdentumista varsin hyvänä. Toisaalta vaikka Firma 1 piti dokumentoinnin kohdentumista hyvänä, niin erilaisia raportteja tehtiin koodinkehittäjien mielestä liikaa ja dokumentointia ei ole oikeastaan suunniteltu. Kuitenkin tehtiin paljon dokumentteja, joiden käyttötarkoitusta ei tunnettu tai ymmärretty. Firman 2 edustajan mukaan dokumentointiportfolio oli aikojen saatossa muotoutunut kohdilleen. Kun ottaa huomioon heidän maantieteellisesti hajautetun organisaationsa, niin se että dokumentointia ei ole voitu korvata face-to-face – kommunikaatiolla oli ymmärrettävää ks. Braithwaite ja Joyce (2005), Korkala ja Abrahamsson (2004), Ramesh et al.(2006) ja Salo (2008).

Ketteriin menetelmiin siirtyminen ei siis dokumentoinnin suunnittelun, kohdentamisen ja sisällön osalta ole haastatteluun osallistuneissa kahdesta ensimmäisestä organisaatioissa tuonut muutoksia.

Kolmas haastattelu toi paljon avoimemman näkökulman. Osa dokumenteista oli suunniteltu yhdessä asiakkaan kanssa ja toteutettiin yhtenä projektin sprintin tehtävälistan tehtävänä. Kolmannen haastateltavan mukaan vaikka tekniseen dokumentointiin oli pohjia, niin dokumenttien kohdentaminen ontui hiukan. Dokumentaatioportfoliota ei ollut suunniteltu riittävän hyvin eivätkä dokumentit kohdentuneet oikealle lukijalle.

8.2 Asioiden taustojen ilmaiseminen

Rüpingin (2005) mukaan dokumentoinnissa pitäisi kertoa myös asioiden taustoja, jotta jatkokehityksessä voidaan arvioida miltä osin. Toisaalta hänen mielestään dokumen-

toinnissa pitää pystyä erottamaan tehdyt ratkaisut kirjoittajan mielipiteistä. Näin toimien luodaan parempi pohja jatkokehitykselle.

Rüping (2005) esittää seuraavat ohjeet:

- Tehdäänkö määrittely yhteistyössä
- Onko suunnitteludokumenteissa tausta ilmaistu, vaan vain ratkaisu
- Ovatko kuvaus ja arviointi erillään.

Haastatelluissa yrityksissä oli haastattelujen pohjalta arvioituna yleinen käytäntö kertoa dokumentoinnissa vaan suunnittelupäätökset. Vain joissain harvoissa tapauksissa tehtiin study-dokumentteja. Samoin dokumentoijan henkilökohtaiset mieltymykset suunnittelupäätöstöisten tekemisessä olivat samanarvoisia faktoihin perustuvien ratkaisuiden kanssa. Määrittelytyötä tehtiin jonkin verran yhteistyössä, mutta ainakin toisen haastatellun organisaation kohdalla, organisaatiossa oli yksi vastuuhenkilö jokaiselle dokumentille.

8.3 Vastuuhenkilö ja resursointi

Dokumentointi on vaativa tehtävä, johon pitää varata resursseja kuten muuhunkin ohjelmiston kehityksen tehtäviin. Jotta dokumentin sisältö ja valmistumisaikataulu pysyisivät suunnitelmien mukaisena, tulisi jokaisella dokumentilla olla nimetty vastuuhenkilö, jonka ei kuitenkaan tarvitse olla dokumentin kirjoittaja. Projektin vaatimukset elävät, ja toteutuksessa tulee esiin uusia asioita, joten dokumentoinnin täytyy olla jatkuvaa ja dokumenttien päivittyä projektin edetessä. (Rüping, 2005.)

Rüping (2005) esittää seuraavat ohjeet:

- Onko dokumentointi erillinen tehtävä, jolle on varattu omat resurssit.
- Onko dokumentilla vain yksi vastuuhenkilö
- Jatkuva dokumentointi
- Kirjoitus ja peilaus.

Kaikki haastateltavat kertoivat että kaikki merkittävä dokumentointityö tehtiin ottamalla dokumentit omiksi sprintin kehitysjonon tehtäväksi. Samoin kaikki haastateltavat kertoivat, että heillä dokumenteilla oli nimetty vastuuhenkilö ja dokumentteja päivitetään jatkuvasti. Toisin sanoen Rüpingin ohjeiden mukaista käytäntöä noudatetaan haastateltavien mukaan kaikissa tutkituissa organisaatioissa.

8.4 Palautteen saaminen

Dokumentin kirjoittaminen on oppimisprosessi. Ohjelmistosuunnittelija harvoin on sovellusalueen asiantuntija. Dokumentin esittäminen arvosteltavaksi eri katselmoinneissa on tapa saada palautetta. Dokumentin kirjoittamisen aikana epäviralliset katselmoinnit auttavat saamaan palautetta. Katselmoinnissa pitäisi olla myös näkökulmaa projektin ulkopuolelta ja varsinkin asiakkaan suunnalta. (Rüping, 2005.)

Rüping (2005) esittää seuraavat ohjeet:

- Katselmointikulttuuri
- Katselmointi ennen toimitusta
- Asiakaskatselmointi
- Puolueeton näkökulma.

Kaikilla haastatteluun osallistuneilla oli jonkinlaiset katselmointi käytännöt. Yleensä katselmoinnilla tarkoitettiin kuitenkin jonkin näköistä luovutustarkastusta, missä tarkistettiin dokumentin ulkoasu. Yhdenkään haastateltavan organisaation kulttuuriin ei kuulunut sovellustiedon tuominen mukaan katselmointien kautta. Rüpingin ohjeiden mukaan katselmoinnit eivät ole mekaanisia tarkastustilaisuuksia, vaan katselmointien avulla hankitaan tietoa. Tätä näkökulmaa ei yhdessäkään haastattelussa tullut esille.

8.5 Projektin markkinointi ja dokumentin markkinointi

Projektin markkinointi tai oikeastaan uusien työntekijöiden sisään ajomateriaali on Rüpigin mielestä tärkeä (Rüping, 2005). Haastatellut yritykset olivat laiminlyöneet tämän. Yleensä perusteluna oli se, ettei projektiin odotettu tulevan uutta väkeä.

Dokumentointi on turhaa jos ei tietoa osata etsiä oikeasta dokumentista (Rüping, 2005). Kaksi ensimmäistä haastateltavaa organisaatiota luottivat siihen, että dokumentointi käytäntö on aikojen saatossa vakiintunut ja että projektin työntekijät tietävät mitä dokumentteja on olemassa. Vain viimeinen haastateltava kertoi että heillä mainostetaan dokumenttien valmistumista.

Rüpingin ja ketterien ohjelmistonkehitysmenetelmien näkemystä siitä, että projektilla pitäisi olla lyhyt Kokonaiskuvadokumentti, joka vastaa enemmän kysymykseen kuka tietää kuin miten, ei haastattelun kohteena olevissa organisaatioissa ollut huomioitu. Myöskään dokumenttien markkinointia ei harrastettu, vaan oletettiin että dokumentti osataan etsiä vanhojen käytäntöjen mukaan ja että projektin jäsenet tietävät minkälaisia dokumentteja on olemassa tai tullaa tekemään.

9. Loppupäätelmät

Ketterät ohjelmistonkehitysmenetelmät ovat olleet tulossa tämän vuosituhannen alun ja olleet puheenaihe ohjelmistojen kehityksen alueella. Monet ohjelmistoalan yritykset tutkivat menetelmien käyttöönottoa. Jotkut mieltävät ketterät ohjelmistonkehitysmenetelmät parhaimmaksi asiaksi mitä ohjelmistoteollisuudessa on tapahtunut, kun taas jotkut näkevät ne paluuksi menneisyyteen ja rinnastavat ne hakkeroimiseen.

Ketterissä ohjelmistonkehitysmenetelmissä tiedonkulkua suullisen kommunikoinnin avulla on yritetty parantaa. XP-ohjelmistonkehitysmenetelmässä on käytössä pariohjelmointi, Scrum-prosessissa päiväpalaverit ja jaettu työskentelytila. Myös ketterien prosessien nopeat iteraatiosyklit parantavat kommunikaatiota.

Ketterissä ohjelmistonkehitysmenetelmissä on pohjimmiltaan ollut alusta saakka paljon tiedon johtamisen elementtejä. Toistaiseksi käytännössä on menetelmien käyttö keskittynyt vielä ratkaisemaan teknisiä ongelmia ja tiedon johtamisen näkökulmat ovat olleet hyvin toissijaisia. Käytännön tekijät ketterissä ohjelmistomenetelmiä räätälöitäessä ovatkin yleensä pohjakoulutukseltaan suuntautuneet enemmän tekniikkaan kuin johtamiseen tai tiedon johtamiseen. Tutkimuksen mielenkiinto onkin kuinka ketterät menetelmät käytännössä toteuttavat niihin liitettyjä tiedon hallinnan elementtejä ja kuinka vuosia perinteiseen dokumentointiin luottaneet yritykset ovat pystyneet muuttamaan toimintatapojaan ketterien menetelmien aatteita hyödyntäviksi.

Ketterät menetelmät eivät anna yksityiskohtaisia toimintaohjeita, vaan enemmänkin suuntaviivoja, jolloin toimintatavat joudutaan räätälöimään organisaatiokohtaisesti. Tässä tutkimuksessa tutkittiin näitä räätälöityjä prosesseja. Tutkimuksessa paneuduttiin yksinkertaisilla haastatteluilla yritysten kehittämiin ketteriin ohjelmistonkehitysprosesseihin ja siihen, mitenkä prosessit huomioivat ketterien ohjelmiston kehitysmenetelmien taustalla olevat tiedon hallinnan teoriat.

Oliko ketteriin ohjelmistonkehitysmenetelmiin siirtyminen muuttanut tämän tutkimuksen kohteena olleiden yritysten dokumentointi käytäntöjä? Haastattelut kertoivat aika selvästi että tieto projekteissa välittyy yhä dokumenttien avulla. Myöskään yllättävää ei ollut, ettei ketteriin menetelmiin siirtyminen ole oikeastaan tuonut mitään muutosta dokumentoinnin tarpeeseen. Jopa kasvua erilaisten raporttien muodossa löytyi. Kaksi kolmesta haastatelluista kertoi, että virtuaalitiimit ja offshore-toiminta vaatii dokumentointia.

Dokumentointia ei suunniteltu projektikohtaisesti, vaan turvauduttiin vanhoihin käytäntöihin ja malleihin. Dokumentointia tehtiin turhaan, eikä aina edes tiedetty mihin tietoa tullaan käyttämään. Dokumenteissa keskityttiin kertomaan vain suunnitteluratkaisut, eri vaihtoehtojen ja taustojen selvittely oli harvinaista.

Olivatko yritykset Oulussa siirtyneet luottamaan suullisen kommunikointiin vai tehdäänkö yhä perinteisen mallisia laajoja dokumenttikirjastoja? Haastattelut eivät indikoineet mitään sellaista, että kommunikoinnissa olisi ketteriin ohjelmistonkehitysmenetelmiin siirtymisen myötä muutettu näiltä osin käytäntöjä.

Korostetaanko toimivaa ohjelmaa vai vaaditaanko kokonaisvaltaista dokumentointia? Tähän kysymykseen ei kunnolla saatu vastausta johtuen haastatteluun saatujen organisaatioiden luonteesta. Ketterät ohjelmistonkehitysympäristöt tuntuivat merkitsevän lähinnä nopeaa integrointisykliä, mistä on arvattavissa halu panostaa siihen, että toimitetaan toimivaa ohjelmistoa muille testattavaksi. Dokumentointia ei taasen ollut paineita muuttaa.

Onko ketteriin ohjelmistonkehitysmenetelmiin siirtyminen muuttanut dokumentoinnin tarvetta? Ketteriin ohjelmistonkehitysmenetelmiin siirtyminen ei kuitenkaan ollut aiheuttanut paineita vähentää dokumentointia.

Tutkimus oli tapaustutkimus, johon osallistui kolme yritystä. Monet yritykset kieltäytyivät perustellen kehitysprosessien olevan heidän core-osaamista ja pelkäsivät tiedon luovuttamista.

Haastatteluun saatiin siis vain kolme yritystä, mikä on hyvin pieni määrä tuomaan luotettavaa kuvaa. Haastateltavien aikataulut olivat myös kiireiset ja haastattelulle saatu noin yhden tunnin aika ei ollut riittävä. Teemahaastatteluun olisi kuitenkin pitänyt varata runsaasti aikaa (Hirsjärvi & Hurme, 1995).

Laadullisen teemahaastattelun tekeminen on vaikeaa (Hirsjärvi & Hurme, 1995). Nyt haastattelija ja haastattelun suunnittelija oli ensikertalainen.

Tutkimuksessa tuli esille näkökulma, mitä ei maailmalla ole paljonkaan tutkittu: "Miten ketteriä ohjelmistonkehitysmenetelmiä oikeastaan käytetään?" Paljon on tutkimusta siitä, miten pitäisi tai mikä olisi hyvä tapa, muttei siitä, miten toimitaan. Vaikka haastattelujen lukumäärä jäi tässä tutkimuksessa vähäiseksi, tutkimus tuo näkökulmia siihen, miten paljon ketterien menetelmien prosesseja räätälöidään.

Lähdeluettelo

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). *Agile software development methods: Rewiew and analysis* VTT Finland.

Abrahamsson, P., Warsta, J., Siponen, M. T., & Ronkainen, J. (2003). New directions on agile methods:A comparative analysis. *Proceedings of the 25th International Conference on Software Engineering : Portland.*

Agile Alliance. (2001). *Manifesto for agile software development.* Luettu 26.11.2012, http://agilemanifesto.org/

Alasuutari, P. (1999). *Laadullinen tutkimus* (3. uudistettu painos ed.). Tampere: Osuuskunta Vastapaino.

Argyris, C., & Schön, D. A. (1996). *Organizational learning II, theory, method, and practice.* Reading (MA): Addison-Wesley.

Beck, K. (2000). *Extreme programming explainedembrace change*. Reading (MA): Addison-Wesley.

Beck, K., & Andres, C. (2004). *Extreme programming explained: embrace chang*e. Addison-Wesley Professional.

Bjornson, F. O. (2008). Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used. *Information and Software Technology, 50*(11), 1055-1068.

Boehm, B. (2002). Get ready for agile methods, with care. *Computer, 35*(1), 64-69.

Boehm, B. W., McClean, R. K., & Urfrig, D. B. (1975). Some experience with automated aids to the design of large-scale reliable software. *SIGPLAN Not., 10*(6), 105-113.

Braithwaite, K., & Joyce, T. (2005). XP expanded: Distributed extreme programming. Extreme programming and agile processes in software engineering, 1524-1526.

Briand, L. C. (2003). Software documentation: How much is enough? *Proceedings of the Seventh European Conference on Software Maintenance and Reengineering,* pp. 13-15.

Cao, L., & Ramesh, B. (2008). Agile requirements engineering practices: An empirical study. *IEEE Software, 25*(1), 60.

Chau, T., Maurer, F., & Melnik, G. (2003). Knowledge sharing: Agile methods vs. tayloristic methods. *Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises. WET ICE 2003.*, 302-307.

Chih-Wei Ho, Johnson, M. J., Williams, L., & Maximilien, E. M. (2006). On agile performance requirements specification and testing. *In Agile Conference, 2006,* 6-52.

Chow, T., & Cao, D. (2008). A survey study of critical success factors in agile software projects. *Journal of Systems and Software, 81*(6), 961-971.

Clements, P., Garlan, D., Little, R., Nord, R., & Stafford, J. (2003). Documenting software architectures: Views and beyond. *Proceedings of the 25th International Conference on Software Engineering,* 740-741.

CMMI Product Team. (2006). *CMMI for development, version 1.2. ,* Pittsburgh, USA: Carnegie Mellon Software Engineering Institute

Cohn, M., & Ford, D. (2003). Introducing an agile process to an organization. *Computer, 36*(6), 74-78.

de Souza, S. C. B., Anquetil, N., & de Oliveira, K. M. (2005). A study of the documentation essential to software maintenance. *Proceedings of the 23rd Annual International Conference on Design of Communication: Documenting & Designing for Pervasive Information,* Coventry, United Kingdom. pp. 68-75.

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology, 50*(9–10), 833-859.

Eisenhardt, K. M. (1989). Building theories from case study research. *Academy of management review*, 532-550.

Endres, A., & Rombach, H. D. (2003). *A handbook of software and systems engineering: Empirical observations, laws, and theories.* Harlow,USA: Pearson.

Eskola, J., & Vastamäki, J. (2001). Teemahaastattelu: Opit ja opetukset. In J. Aaltola, & R. Valli (Eds.), *Ikkuoita tutkimusmetodeihin I* (pp. 24). Suomi: PS-kustannus.

Highsmith, J. (2002). *Agile software development ecosystems.* Harlow,USA : Pearson.

Hirsjärvi, S., & Hurme, H. (1995). *Teemahaastattelu* (7. painos ed.). Helsinki: Yliopistopaino.

Hirsjärvi, S., Remes, P., & Sajavaara, P. (2007). *Tutki ja kirjoita* (13. osin uudistettu painos ed.). Helsinki: Tammi.

Järvempää, E. & Kosonen,K. (1996). *Johdatus Tutkimusmenetelmiin ja tutkimuksen tekemiseen*. Espoo:Tekninen korkeakoulu, Tuotantotalouden yksikkö.

Järvinen, P. & Järvinen, A. (1993). *Tutkimustyön metodeista.* Tampere: Tampereen yliopisto

Klein, H. K., & Myers, M. D. (1999). A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS Quarterly, 23*(1), 67-93.

Korkala, M., & Abrahamsson, P. (2004). Extreme programming: Reassessing the requirements management process for an offsite customer. *Software Process Improvement,*12-22.

Kähkönen, T., & Abrahamsson, P. (2003). Digging into the fundamentals of extreme programming building the theoretical base for agile methods. *Proceedings of the 29th Euromicro Conference, 2003,* 273-280.

Lethbridge, T. C., Singer, J., & Forward, A. (2003). How software engineers use documentation: The state of the practice. *Software, IEEE, 20*(6), 35-39.

Lindvall, M., Muthig, D., Dagnino, A., Wallin, C., Stupperich, M., Kiefer, D., et al. (2004). Agile software development in large organizations. *Computer, 37*(12), 26-34.

Martin, A., Biddle, R., & Noble, J. (2004). *The XP customer role in practice: Three studies. In Agile Development Conference, (pp. 42-54). IEEE.*

Mateos-Garcia, J., & Sapsed, J. (2008). Adopting 'Agile' and 'Scrum' practices as 'Organisational becoming': Cases from the UK video games industry. *Euromot 2008: the Third European Conference on Management of Technology,* Nice , France.

Maurer, F., & Martel, S. (2002). Extreme programming: Rapid development for web-based applications. *IEEE Internet Computing, 6*(1), 86-90.

Merisalo-Rantanen, H., Tuunanen, T., & Rossi, M. (2005). Is extreme programming just old wine in new bottles: A comparison of two cases. *Journal of Database Management (JDM), 16*(4), 41-61.

Nerur, S., & Balijepally, V. G. (2007). Theoretical reflections on agile development methodologies. *Communications of the ACM, 50*(3), 79-83.

Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Communications of the ACM, 48*(5), 72-78.

Nonaka, I., & Takeuchi, H. (1995). *The knowledge-creating company: How japanese companies create the dynamics of Innovation. USA:* Oxford University Press.

Nonaka, I., Toyama, R., & Konno, N. (2000). SECI, Ba and Leadership: a Unified Model of Dynamic Knowledge Creation. Long range planning, 33(1), 5-34.

Pedler, M., Boydell, T., & Burgoyne, J. (1989). The learning company. *Studies in Continuing Education, 11*(2), 91.

Ramesh, B., Cao, L., Mohan, K., & Xu, P. (2006). Can distributed software development be agile? *Communications of the ACM, 49*(10), 41-46.

Rao, K. N., Naidu, G. K., & Chakka, P. (2011). A study of the agile software development methods, applicability and implications in industry. *International Journal of Software Engineering and its Applications, 5*(2), 35-46.

Royce, W. W. (1987). Managing the development of large software systems: Concepts and techniques. *ICSE '87: Proceedings of the 9th International Conference on Software Engineering,* Monterey, California, United States. pp. 328-338.

Ruuska, Kai (2001). *Projekti hallintaan*. Jyväskylä:Talentum Media.

Rüping, A. (2005). *Agile documentation: A pattern guide to producing lightweight documents for software projects*. England: John Wiley & Sons Ltd.

Salo, O. (2008). Agile methods in european embedded software development organisations: A survey on the actual use and usefulness of extreme programming and scrum. *IET Software, 2*(1), 58-64.

Sanford, N. (1966). The interview in personality appraisal. *Teoksessa Testing Problems in Perspective.* Toim.Anastasi, Anne. Washington: The American Council of Education,

Schwaber, K. (2002). In Beedle M. (Ed.), *Agile software development with scrum* . Upper Saddle River : Prentice-Hall.

Schwaber, K. (2009). *Agile project management with scrum. USA:* Microsoft Press.

Senge, P. M., Kleiner, A., & Roberts, C. (1994). *The fifth discipline fieldbook.* London: Nicholas Brealey Publishing.

Shine Technologies Pty Ltd. (2003). *Agile Methodologies survey results.* Luettu 1, 2009, http://www.agilealliance.org/system/article/file/1121/file.pdf

Takeuchi, H., & Nonaka, I. (1986). The new new product development game. *Harvard Business Review, 64*(1), 137.

Turk, D., France, R., & Rumpe, B. (2005). Assumptions underlying agile software-development processes. *Journal of Database Management. 16*(4), 62-87.

Vijayasarathy, L., & Turk, D. (2008). Agile software development: A survey of early adopters. *Journal of Information Technology Management, 19*(2), 1-8.

Wikipedia.(s.d.). Software documentation. Luettu 20.11, 2012 http://en.wikipedia.org/wiki/Software_documentation

Liite 1 keskustelurunko:

A. Nopea esittely haastateltavasta

B. Nopea kuvaus käytettävästä ketterästä prosessista

C. Onko siirtymä ketteriin jo vakiintunut vai haetaanko vielä uria?

D. Miten dokumentointi on muuttunut kun yritys on siirtynyt ketteriin (tämä olettaa että haastateltava on ollut yrityksessä useamman vuoden)

   - Onko siirrytty käyttämään face-face
   - Miten on huomioitu jatkuvasti muuttuvat vaatimukset

E. Agile Documentation Patterns: (Andreas Rüping)

Kohdennetaanko dokumentit oikealle lukijalle

1. Kohdennettu tietosisältö
2. Projektikohtaiset dokumentin vaatimukset
3. Onko dokumentointiportfolio mietitty kunnolla
4. Onko keskitytty pitkänajan tietoon
5. Tehdäänkö määrittely yhteistyössä
6. Ovatko suunnitteludokumentoinnissa taustat ilmastu, vaan vain ratkaisu
7. Kokonaiskuva, voiko ihmiset tutustua projektiin, ilman yksityiskohtia
8. Onko kuvaus ja arviointi erillään
9. Realistisia esimerkkejä
10. Onko dokumentointi erillinen tehtävä, jolle on varattu omat resurssit
11. Onko dokumentilla vain yksi vastuuhenkilö
12. Jatkuva dokumentointi
13. Kirjoitus ja peilaus
14. Katselmointi kulttuuri
15. Katselmointi ennen toimitusta

16. Asiakas katselmointi
17. Puolueeton näkökulma
18. Markkinointi
19. Tulevaisuuden hyötyminen