Jussi Koutonen

KETTERIEN MENETELMIEN SKAALAAMINEN – KÄYTÄNNÖN KOKEMUKSIA

Tietojärjestelmätieteen kandidaatin tutkielma
9.2.2009

Jyväskylän yliopisto

Tietojenkäsittelytieteiden laitos

Tiivistelmä

Koutonen, Jussi Veli Ilmari

Tietojärjestelmätiede, järjestelmäkehityksen suuntautumisvaihtoehto, ohjelmistotuotannon syventymiskohde
Ketterien menetelmien skaalaaminen – käytännön kokemuksia
Jyväskylä, Jyväskylän yliopisto, 09.02.09
Kandidaatin tutkielma, 27 sivua.

Ketteriä menetelmiä hyödynnetään enenevissä määrin ohjelmistokehityksen alalla. Niistä on saatu monia myönteisiä kokemuksia. Erityisesti laadun paraneminen ja työhön kuluvan ajan lyheneminen houkuttavat hyödyntämään ketteriä menetelmiä.

Vaikka ketterät menetelmät soveltuvat lähinnä pieniin projekteihin pyritään niitä hyödyntämään myös suurissa projekteissa skaalaamalla. Skaalaamiseen ei ole hopealuotia. Tämä tutkielma on kirjallisuuskatsaus käytännön kokemuksiin, joita on saatu ketteriä menetelmiä skaalattaessa. Tutkielmassa ei ole keskitytty erityisesti XP:hen ja Scrumiin mutta suosionsa takia ketteristä menetelmistä käsitellään juuri näitä.

Lähdemateriaalin perusteella havaittiin kaksi merkittävintä osa-aluetta ketteriä menetelmiä skaalattaessa. Ryhmien hallintaan ja sitä tukeviin toimiin liittyvät ongelmat ovat merkittäviä skaalattaessa ketteriä menetelmiä, koska ketterät menetelmät eivät itsessään tue useita ryhmiä. Vastauksena ongelmiin nähtiin ryhmien omatoimisuuden ja yhteistyön lisääminen. Toinen merkittävä osa-alue ovat ihmiset. Kun ketteriä menetelmiä skaalataan suuriin projekteihin, ihmisiltä vaaditaan erilaisia kykyjä, kuten yrittäjähenkisyyttä.

AVAINSANAT: ketterät menetelmät, ketterien menetelmien skaalaaminen, XP, Scrum

Sisällysluettelo

1 Johdanto....4

2 Yleiskuva ketteristä menetelmistä....6

3 Käytännön kokemuksia ketteristä menetelmistä....9
3.1 Ryhmien hallinnointi....10
3.1.1 Tuotteen työlista....10
3.1.2 Pyrähdys....11
3.1.3 Scrumien scrum-kokous....12
3.1.4 Riippuvuuksien hallinta ja ryhmien koordinointi....14
3.1.5 Jatkuva integrointi....17
3.2 Ihmiset....18
3.2.1 Ryhmä....18
3.2.2 Oikeat ihmiset – asenne vai kokemus....19
3.2.3 Pariohjelmointi....20

4 Yhteenveto....21

Lähteet....23

Liite - Käännökset....26

1 JOHDANTO

Ketterät menetelmät tarjoavat työkaluja, joiden avulla selvitä ohjelmistokehityksen nopeassa tahdissa muuttuvassa ympäristössä, jolle ovat ominaisia muuttuvat vaatimukset ja tiukat aikataulut(Lan Cao & Ramesh, 2007). Ketterät menetelmät soveltuvat pienille kehitysryhmille ja pienen mittakaavan projekteihin, joissa on alle 20 kehittäjää ja yksi tai useampi paikan päällä oleva asiakas (Reifer, Maurer, & Erdogmus, 2003).

On havaittu, että ketterät menetelmät parantavat ohjelmistojen laatua ja lyhentävät niiden kehittämiseen vaadittua aikaa ja kustannuksia (Reifer, 2002). Muun muassa tämän takia ketterät menetelmät ovat herättäneet sekä tutkijoiden että ohjelmistokehitysalan ammattilaisten huomion (P. Abrahamsson, Warsta, Siponen, & Ronkainen, 2003; Mann & Maurer, 2005) ja luoneet painetta yhä suuremmille yrityksille hyödyntää ketteriä menetelmiä(Lindvall et al., 2004; Salo & Abrahamsson, 2008).

Useat organisaatiot tutkivatkin mahdollisuuksia hyödyntää ketteriä menetelmiä niin niille ominaisessa mittakaavassa kuin skaalattuna suurempia projekteja varten. Vielä ei kuitenkaan ole mallia, jonka avulla skaalaaminen onnistuisi helposti (Reifer et al., 2003). Ketterien menetelmien skaalaamisesta on kirjoitettu alan julkaisuissa ja myös erilaisia raportteja käytännön kokemuksista löytyy jonkin verran.

Skaalaaminen on menetelmän soveltamista, räätälöintiä ja muokkaamista suunniteltua suurempaan ohjelmistoprojektiin. Tyypillisesti projekti, jossa hyödynnetään ketteriä menetelmiä, koostuu alle 20 kehittäjästä ja yhdestä tai useammasta paikan päällä olevasta asiakkaasta (Reifer et al., 2003). “Suuressa” ohjelmistoprojektissa on joko yli 30 henkilöä tai yli kolme ryhmää, joiden yhteinen päämäärä on tuottaa kohdeohjelmisto (Moore & Spens, 2008). Tarkkaa

määritelmää skaalaamiselle ohjelmistokehityksen alalla ei ole, mutta edellä esitetty lienee riittävä tämän tutkielman tarpeisiin.

Tässä tutkielmassa pureudutaan ketterien menetelmien skaalaamiseen ja erityisesti skaalaamisesta saatuihin kokemuksiin käytännön näkökulmasta. Tarkoituksena on muodostaa näkemys keskeisistä seikoista, jotka nousevat esille ketteriä menetelmiä skaalattaessa. Tutkielma on kirjallisuuskatsaus 2000-luvulla aiheesta julkaistuihin tutkimuksiin.

Seuraavassa luvussa esitellään Ketterä manifesti, jossa esitellään ketterien menetelmien taustalla vaikuttavat neljä arvoa ja kaksitoista periaatetta. Koska käytännössä koskaan ei skaalata puhtaasti vain yhtä menetelmää saati hyödynnetä sitä koko laajuudessaan, ei eri menetelmien tarkka kuvaaminen ole oleellista. Näin myös siksi, että julkaisuissa on hämärtynyt eri ketterien menetelmien rajat ja yleensä viitataan vain ketteryyteen. Kolmannessa luvussa esitellään käytännön kokemuksia. Luku on jaettu kahteen alalukuun, joissa on edelleen omia alalukujaan. Kahtiajaossa perusteena on ollut alalukujen liittyminen joko ryhmien hallinnointiin tai ihmisiin ja heidän ominaisuuksiinsa.

Tutkielman lopussa liitteenä olevan käännökset ovat omiani, koska ei ole standardeja käännöksiä termeille, joita tutkielmassa on runsaasti. Termit on käännetty, koska tutkielman kielen tulee olla suomea siitäkin huolimatta, että tavallisesti käytettäisiin englanninkielisiä termejä ketteristä menetelmistä puhuttaessa. Lisäsin käännökset mahdollisten väärinkäsitysten välttämiseksi ja helpottamaan lukemista.

2 YLEISKUVA KETTERISTÄ MENETELMISTÄ

Tässä luvussa kuvataan ketterien menetelmien ominaiset piirteet. Nämä piirteet heijastelevat Ketterää manifestia (Agile Alliance, 2001), jossa luetellaan ketterien menetelmien neljä arvoa:

Yksilöitä ja vuorovaikutusta mieluummin kuin prosesseja ja työkaluja

Toimivaa ohjelmistoa mieluummin kuin kattavaa dokumentaatiota

Yhteistyötä asiakkaan kanssa mieluummin kuin sopimusneuvotteluja

Muutoksiin vastaamista mieluummin kuin suunnitelman seuraamista

Vaikka arvostemmekin käsitteitä, jotka ovat oikealla puolella, arvostamme vasemmalla puolella olevia käsitteitä enemmän (Agile Alliance, 2001). Ketterä toiminta painottaa ohjelmistokehittäjien keskinäistä suhdetta, yhteisöllisyyttä ja inhimillistä roolia vastakohtana laitosmaisille prosesseille ja kehitystyökaluille. Tiimien sisäiset suhteet, työympäristön ilmapiiri ja muut tiimihengen nostatuskeinot ovat keskeisessä asemassa ketterissä menetelmissä. Ihmisten lisäksi ohjelmistotuotannossa tärkeää on itse tuotos ja tuotteen pääkomponentti, toimiva järjestelmä, siksi toimivaa ohjelmistoa arvostetaan enemmän kuin kattavaa dokumentaatiota. Kolmannen arvon pääsanoma on se, että yhteistyö asiakkaan kanssa on yksi kriittisistä menestystekijöistä ohjelmistoprojekteissa, koska aktiivisen yhteistyön kautta asiakas voi auttaa tiimiä ymmärtämään kehitettävän ohjelmiston toiminnallisia vaatimuksia. Neljäs arvo edellyttää, että kehittämisryhmän jäsenet ovat valmiita tekemään muutoksia ja olemassa olevat sopimukset on tehty sellaisella työkalulla, joka tukee ja mahdollistaa muutosten tekemisen, koska sellaisen järjestelmän toimittaminen asiakkaalle, joka toteuttaa ei-tärkeitä vaatimuksia, on turhaa. (Airaksinen, 2004)

Ketterässä manifestissa luetellaan myös 12 periaatetta, joita ketterät menetelmät

pyrkivät toteuttamaan.

Korkein prioriteettimme on tyydyttää asiakkaan tarpeet toimittamalla varhaisessa vaiheessa ja jatkuvin aikavälein toimivia ohjelmistoversioita

Hyväksy vaatimusten muutokset, jopa kehityksen myöhäisissä vaiheissa. Ketterät prosessit valjastavat muutoksen asiakkaan kilpailueduksi.

Julkaise toimiva ohjelmisto useasti, muutamasta viikosta pariin kuukauteen.

Liikemiehien ja kehittäjien tulee työskennellä yhdessä päivittäin projektin ajan.

Rakenna projektit motivoituneiden ihmisten ympärille. Anna heille heidän tarvitsemansa ympäristö ja tuki ja luota heihin saadaksesi työ tehdyksi.

Suorituskykyisin ja tehokkain menetelmä tiedon kuljettamiselle kehitystiimin sisällä on kasvokkain tapahtuva viestintä.

Toimiva ohjelmisto on päämittari edistymiselle.

Ketterät prosessit tukevat kestävää kehitystä. Sponsorien, kehittäjien ja käyttäjien tulisi pystyä pitämään tasainen tahti ennalta määräämättömän ajan.

Jatkuva huomio tekniseen laadukkuuteen ja hyvään suunnitteluun parantaa ketteryyttä.

Yksinkertaisuus – tekemättömän työn maksimoimisen taito – on oleellista.

Parhaat arkkitehtuurit, vaatimukset ja suunnitelmat hioutuvat itseohjautuvissa ryhmissä.

Tasaisin väliajoin ryhmän on pohdittava, kuinka tulla tehokkaammaksi, ja toimittava pohdintojen lopputuloksen mukaisesti

Vaikka ketterillä menetelmillä on yhteinen tausta ketterässä manifestissa, ovat

ne toisistaan poikkeavia ja saattavat painottaa ohjelmistokehityksen eri osa-alueita. Siinä missä Scrum (Schwaber & Beedle, 2001) on pääasiasssa projektinhallinta työkalu, XP (Beck, 1999) on kokoelma hyväksi havaittuja periaatteita ja käytäntöjä (P. Abrahamsson, Salo, Ronkainen, & Warsta, 2002). Tästä syystä on huomattava, että eri ketteriä menetelmiä voidaan hyödyntää yhtä aikaa. Kahta eri menetelmää hyödyntävässä organisaatiossa menetelmien rajat hämärtyvät ja usein viitataankin vain ketteryyteen. Mahdollisesti tästä johtuen julkaisuissa oli suoria asiavirheitä; pariohjelmointia väitettiin Scrumin käytännöksi vaikka se kuuluu XP:hen (Lee, 2008).

Ketterien menetelmien ominaisia piirteitä ovat Abrahamssonin määritelmän mukaan inkrementaalisuus, yksinkertaisuus, mukautuvuus ja yhteistyöpainotteisuus. Inkrementaalisuus tarkoittaa pieniä ohjelmajulkaisuja ja nopeita, lyhyitä kehityssyklejä. Yksinkertaisuudella tarkoitetaan menetelmän oppimisen helppoutta ja muokattavuutta sekä selkeää määrittelyä. Mukautuvuus on menetelmien ominaisuus, joka mahdollistaa viime hetken muutoksien tekemisen ja muutoksiin reagoimisen. Yhteistyöpainoitteisuus viittaa ketterissä menetelmissä vallitsevaan läheiseen kanssakäymiseen asiakkaiden ja kehittäjien välillä. (P. Abrahamsson et al., 2002)

3 KÄYTÄNNÖN KOKEMUKSIA KETTERISTÄ MENETELMISTÄ

Ketterien menetelmien skaalaamiseen ei ole olemassa kaiken kattavaa tai edes alkeellista mallia. Vuonna 2003 järjestettiin Kanadassa ketterien menetelmien skaalaamiseen keskittynyt työpaja, jossa alan ammattilaiset keskustelivat aiheeseen liittyvistä ongelmista. Osallistujat kertoivat kokemuksistaan ja havainnoistaan ongelmiin liittyen. Pätevää ratkaisua ketterien menetelmien skaalaamiseen ei kuitenkaan syntynyt mutta sellaisen tarpeellisuudesta osallistujat pääsivät yhteisymmärrykseen (Reifer et al., 2003). Huomiota tukevat myös Nokian organisaation kokemukset, joiden mukaan XP:n hyödyntäminen suuressa organisaatiossa ilman menetelmän räätälöimistä on yleensä mahdotonta (Lindvall et al., 2004).

Joissakin tapauksissa organisaatiot ovat hyödyntäneet erilaisia malleja siirtymässään perinteisistä menetelmistä ketteriin menetelmiin. Nämä mallit eivät ole kuitenkaan auttaneet varsinaisessa skaalaamisessa vaan organisaation muutoksessa tai siirtymisen kypsyyden määrittelyssä. Eräs organisaatio siirtyi vesiputous -mallista ketteriin menetelmiin käyttäen Tuckmanin mallia ja sen eri tasoja siirtymän kypsyyden mittarina. Mallin käyttö helpotti eri ratkaisujen seurauksien havainnollistamisessa; esimerkiksi henkilöstön lisääminen pudotti ryhmän kypsyyden alemmalle tasolle (Lee, 2008). Toisessa organisaatiossa oli organisaation muutokseen pureutuvan kirjallisuuden pohjalta tunnistettu kahdeksan kriittistä menestystekijää, jotka ovat avainasemassa muutoksen onnistumiseksi. Hieman ristiriitaisesti organisaation muutoksesta kertovassa raportissa todettiin, että käytettiin onnistumisen mittarina sitten käyttöönotettuja XP:n käytäntöjä tai menestystekijöitä siirtymä voitaisiin tulkita epäonnistuneeksi mutta käytännössä toimivaksi (Spayd, 2003). Muutamissa raporteissa, kuten (Kniberg & Farhang, 2008), ketterät menetelmät on otettu onnistuneesti käyttöön projektin pelastamiseksi, kun perinteisiä menetelmiä

käyttämällä projekti on ajautunut umpikujaan eikä sen hylkääminen ole tullut kysymykseen.

Yhteisiä kaikille käytännön kokemuksia raportoiville tutkimuksille on niissä esiintyvät ongelmakohdat ja merkittävimmät havainnot. Ilmeisimpiä ongelmakohtia ketterien menetelmien skaalaamisessa ovat ainakin riippuvuuksien hallinta, jatkuva integrointi ja kommunikaation säilyminen ryhmien välillä, jotka kaikki osaltaan vaikuttavat ryhmien hallinnointiin. Merkittävin yksittäinen osa-alue ketteriä menetelmiä skaalattaessa ovat ihmiset, joita jo ketterässä manifestissakin painotetaan (Moore & Spens, 2008). Kokemuksia näiltä osa-alueilta valotetaan seuraavaksi omissa alaluvuissaan.

3.1 Ryhmien hallinnointi

3.1.1 Tuotteen työlista

Tuotteen työlista (engl. product backlog) määrittelee kaiken, mitä nykytietämyksen perusteella on tehtävä lopullisen tuotteen valmistamiseksi. Listassa jokaiselle tehtävälle määritellään prioriteetti ja arvioidaan työhön kuluva aika. Eri tehtävien väliset riippuvuussuhteet on nähtävissä tuotteen työlistassa. Työlistaa päivitetään ja tarkennetaan jatkuvasti. (P. Abrahamsson et al., 2002)

Millerin raportissa kerrotaan kolmikerroksisesta puurakenteesta, jonka yksittäiset lehtisolmut ovat kehitettäviä ominaisuuksia eli työlistan tehtäviä. Lehtisolmut ovat rakenteen kolmannella tasolla, jotka kutsutaan ominaisuushaaraksi (engl. feature brach). Kun ominaisuus on valmis, koodi integroidaan aina edeltävän tason kokonaisuudeksi kunnes saavutetaan juurisolmu, josta lopullinen tuote rakennetaan. Haaroittamisen vuoksi lehtisolmujen vanhempiin tulleet muutokset integroidaan lehtisolmuun joka toinen viikko. (Miller & Carter, 2007)

Tarinaseinä (engl. story wall) on yksi tuotteen työlistan muoto. Tarinakortti (engl. story card) on paperi, jossa määritellään tehtävä, tehtävän prioriteetti ja mahdolliset riippuvuussuhteet. Tarinaseinän on oltava fyysisesti olemassa ja seinälle pitää sijoittaa kaikki tarinakortit, jotta ryhmän ja projektin edistymisen hahmottaminen on helppoa. Leen raportissa kokeiltiin myös tarinaseinän toteuttamista taulukkolaskentatiedostona mutta se havaittiin huonoksi juuri hahmottamisongelmien ja puutteellisten tietojen takia. (Lee, 2008)

Tuotteen työlistan yksityiskohtaisuus on muutenkin tuottanut ongelmia. Lyonin raportissa projektilla oli liikaa yksityiskohtia, kun yksittäisten ryhmien työlistat oli koottu yhdeksi isäntälistaksi. Tällöin yksi ihminen ei mitenkään kyennyt hahmottamaan työlistaa saati ohjaamaan ryhmiä sen perusteella. Asiaa pahensi tehtävien suorittamiseen kuluvan ajan hankala arviointi, kun ryhmille ei ollut yhteistä nopeuden mittaria. Ongelma ratkaistiin lopulta siten, että ryhmillä oli omassa työlistassaan ryhmän tehtävät yksityiskohtineen ja jokaisella tehtävällä sitä vastaava kohta isäntälistassa. Ryhmissä oli tuotteen omistaja, joka huolehti työlistan tehtävien priorisoinnista, ja ryhmä arvioi tehtäviin kuluvan ajan, jota sitten verrattiin ryhmän keskimääräiseen nopeuteen, jotta arviot olisivat ryhmien välillä vertailukelpoisia. Isäntälistasta ryhmien tuotteen omistajat saattoivat tarkistaa, mitkä olivat koko projektin prioriteetit. (Lyon & Evans, 2008)

3.1.2 Pyrähdys

Pyrähdys (engl. sprint) on Scrumin käytäntö ja nimitys yksittäiselle ohjelman rakentamiseen tarkoitetulle iteraatiolle. Pyrähdyksessä on seuraavat osat: pyrähdyksen suunnittelukokous (engl. sprint planning meeting), pyrähdyksen työlista (engl. sprint backlog) ja päivittäinen scrum-kokous (engl. daily scrum meeting). Pyrähdyksen lopussa pidetään pyrähdyksen katselmus (engl. sprint review meeting). Yhden pyrähdyksen pituus on tavallisesti viikosta neljään

viikkoon. (P. Abrahamsson et al., 2002)

Päällekkäisyyksien välttämiseksi voidaan ryhmille määrätä saman pituiset pyrähdykset, jotka alkavat ja päättyvät samaan aikaan ja jotka nimetään loogisesti. Tällä helpotetaan projektin rytmitystä ja ryhmien välistä kommunikaatiota, samalla vältetään väärinymmärrykset aikataulutuksen suhteen. (Lyon & Evans, 2008)

Pyrähdysten jatkaminen, koska tehtäviä ei saatu suoritettua pyrähdyksen aikana, on osoittautunut huonoksi vaihtoehdoksi. Pyrähdyksen jatkaminen aiheutti vain lisää viivästymistä, koska ryhmän jäsenet eivät enää tarkalleen tienneet, mitä heidän tulisi tehdä tehtävien suorittamiseksi. Sen sijaan tehtävät siirretään seuraavaan pyrähdykseen tai tehtävistä karsitaan osia. (Mann & Maurer, 2005)

3.1.3 Scrumien scrum-kokous

Edellisessä aliluvussa mainittu päivittäinen scrum-kokous on scrumien scrum-kokouksen pohjana. Päivittäinen scrum-kokous on lyhyt ryhmän jäsenten kokous, jossa heistä jokainen kertoo, mitä he ovat tehneet viime kokouksen jälkeen, mitä he aikovat seuraavaksi tehdä ja mitä mahdollisia esteitä heidän työlleen on havaittavissa. Havaittuihin ongelmiin puututaan varsinaisen kokoukseen jälkeen niiden toimesta, joita asia koskee. Kokouksen pääasiallinen tarkoitus onkin varmistaa, että ryhmä on ajan tasalla ja varma tehtävistään. (P. Abrahamsson et al., 2002)

Scrumien scrum-kokous on keino useamman kuin kolmen ryhmän hallitsemiseksi. Kokouksen tarkoituksena on mahdollistaa ryhmien välinen kommunikointi. Erityisesti kokouksessa tulisi keskittyä riippuvuuksiin ja integrointiin. Kokoukseen osallistuu vähintäänkin yksi henkilö jokaisesta ryhmästä. Osallistuvan henkilön tulisi olla joku muu ryhmästä kuin tuotteen

omistaja tai scrum-ohjaaja. Ryhmän tulisi itse päättää, kuka osallistuu kokoukseen, perustuen jäsenten vahvuuksiin ja kulloinkin kokouksessa käsillä oleviin ongelmiin tai aiheisiin. (Cohn, 2007)

Scrumien scrum-kokouksen kesto ja kokousten määrä viikossa vaihtelee. Cohn ehdottaa, että kesto ja kokousten määrä tulisi olla ryhmien itse päätettävissä. Artikkelissaan hän suosittelee tavallisesta scrum-kokouksesta poikkeavaa, pitempää kestoa ja kokousten määrän vähentämistä päivittäisestä. Cohn perustelee näkemystään scrumien scrum-kokouksen poikkeavuudella scrum-kokouksesta. Sen sijaan, että ongelmat ratkaistaan kokouksen ulkopuolella, ne ratkaistaan yhteistyössä kokouksen aikana, koska useita ryhmiä koskevat ongelmat uhkaavat koko projektin etenemistä. Ongelmia seurataan ongelmalistalla, jota päivitetään kokouksissa. (Cohn, 2007)

Grewalin raportissa organisaatio hyödynsi onnistuneesti Scrumien scrum-kokouksia vaikkakin hieman eri tavalla kuin Cohn sitä ehdottaa. Kokoukseen osallistui aina ryhmän vetäjät, eikä kokoukseen osallistujaa vaihdeltu. Ryhmän vetäjät sopivat keskenään ryhmien koordinoinnista ja kertoivat omille ryhmilleen pääasiat muiden ryhmien toimista. Jos ryhmä havaitsi, että he joutuvat jollakin osa-alueella työskentelemään yhteistyössä toisen ryhmän kanssa, heidän oli itse mentävä keskustelemaan asiasta toisen ryhmän kanssa. Tämä tapa osoittautui paremmaksi kuin aiemmin kokeiltu ryhmän jäsenen hoitama jatkuva raportointi muille ryhmille, koska raporteissa oli harvoin hyödyllistä tietoa. (Grewal & Maurer, 2007)

Scrumien scrum-kokousten hyödyntäminen ei välttämättä yksistään ole riittävä ratkaisu ryhmien koordinoimiseen ja ketterien menetelmien skaalaamiseksi tarvitaan myös muita toimia. Tai ryhmien koordinoinnissa voi käyttää aivan muita keinoja, kuten Lyonin raportissa esitellään. (Lyon & Evans, 2008)

3.1.4 Riippuvuuksien hallinta ja ryhmien koordinointi

Riippuvuuksien hallinnalla tarkoitetaan, että sellaiset tilanteet, joissa ryhmien toimilla on vaikutuksia muiden ryhmien toimiin, pyritään välttämään tai ratkaisemaan mahdollisimman sulavasti. Yleensä ratkaisu ongelmatilanteiden välttämiseen on ryhmien koordinointi. Koordinoinnin toteutus on mahdollista monin eri tavoin.

Riippuvuuksien hallintan ja ryhmien koordinoinnin merkitys kasvaa, kun ryhmien määrä lisääntyy. Toisaalta suuri määrä ryhmiä voi tarkoittaa myös monimutkaisempaa järjestelmää, mikä entisestään lisää riippuvuuksien hallinnan merkitystä, koska todennäköisyys että useat ryhmät työstävät samaa lähdekoodia tai hyödyntävät samoja ominaisuuksia kasvaa. Riippuvuuksien hallintaan ja ryhmien koordinointiin ei kiinnitetä huomiota ketterissä menetelmissä, koska projektit oletetaan pieniksi ja ryhmien määrä vähäiseksi. (Babinet & Ramanathan, 2008)

Kahden ryhmän toimien välisestä riippuvuudesta voi koitua ongelmia, jos ensimmäisen ryhmän tekemätön työ estää toisen ryhmän töiden tekemisen tai ryhmän tekemä muutos koodiin vaatii muutoksia myös toisen ryhmän työstämään koodiin. (Babinet & Ramanathan, 2008)

Ryhmien välisen kommunikoinnin tukeminen on tärkeää, koska ketterät menetelmät eivät ota kantaa ryhmien välisestä kommunikoinnista ja koordinoinnista syntyviin ongelmiin. Nokian organisaatiossa havaittiin, että eräs ratkaisu ongelmaan on pyrkiä vähentämään tarvetta ryhmien väliselle kommunikoinnille. Tämä onnistuu, jos yksittäinen ryhmä kehittää itsenäistä osajärjestelmää ja sijaitsee fyysisesti yhdessä paikassa. Kokemus on kuitenkin osoittanut, että tällaiseen tilanteeseen pääseminen ja arkkitehtuurin säilyttäminen on hankalaa. Vääjäämättömät muutokset arkkitehtuuriin muuttavat tilannetta ja kommunikaatiolle onkin tarvetta. Toinen

lähestymistapa on ryhmien välisten työpajojen jatkumo, jossa ryhmien jäsenet kokoontuvat yhdessä ratkaisemaan useampia ryhmiä koskettavat ongelmat. (Lindvall et al., 2004)

Microsoftin organisaatiossa järjestelmän osittaminen toteutettiin haaroittamalla (engl. branching). Haaroittamisessa jokaiselle ryhmälle annetaan omavarainen tilannevedos tarvittavista ulkopuolisista riippuvuussuhteista, työkaluista ja lähdekoodista (Jacob, Rodriquez, & Barry,). Tilannevedoksen ansiosta osajärjestelmää voidaan kehittää omaan tahtiin ja muista osajärjestelmistä riippumatta. Ajoittain tilannevedos kuitenkin täytyy päivittää, jotta se on muun koodin kanssa ajantasalla. Haaroittamisessa eristetään yksittäinen ominaisuus (engl. feature), jota ryhmä työstää. (Miller & Carter, 2007)

Babinetin raportissa kuvaillaan Salesforce.com -organisaation lähestymistapa riippuvuuksien hallintaan. Lähestymistapa muistuttaa suuresti scrumien scrum-kokouksien hyödyntämistä mutta sisältää myös muita välineitä. Organisaatio pyrkii luomaan läpinäkyvyyttä projektiin, tarjoamaan työkaluja ryhmien väliseen tiedon jakamiseen ja yhteistyöhön, korostamaan ryhmien oma-aloitteisuutta riippuvuuksien hallitsemisessa ja hyödyntämään automatisointia mahdollisimman paljon. Lisäksi havaittiin, että riippuvuuksien hallinan on oltava jatkuvaa eikä riitä, että riippuvuudet huomioidaan vain pyrähdyksen alussa tai pyrähdystä edeltävässä suunnitteluvaiheessa. (Babinet & Ramanathan, 2008)

Salesforce.comilla on erityinen aloituspalaveri, johon kaikki projektin ryhmät osallistuvat. Palaverissa esitellään, mitä ryhmät suunnittelevat tekevänsä, joten palaveria ennen ryhmien on suunniteltava toimensa. Tällä halutaan varmistaa, ettei yksikään ryhmä ole vailla suunnitelmaa, koska riippuvuuksien hallinnan kannalta on vaikeaa selvittää riippuvuuksia, jos jollakin ryhmällä on puuttellinen näkemys tulevista tehtävistä. Lisäksi palaverilla pyritään

takaamaan, että kaikki ryhmät tietävät, mitä muut tulevat tekemään. Tämä edesauttaa ryhmien välistä omatoimista riippuvuuksien selvittämistä. Myöhemmin näihin alustaviin suunnitelmiin tulleita muutoksia käsitellään pyrähdysten kaitselmuksissa, jotta riippuvuuksista pysytään ajan tasalla. (Babinet & Ramanathan, 2008)

Aloituspalaveriin liittyy pian sen jälkeen pidettävä riippuvuuksien tunnistamisharjoitus, jossa kunkin ryhmän edustajat yhdessä kartoittavat ryhmien väliset riippuvuudet. Harjoituksessa käytetään sovittua merkintätapaa ja lopputuloksena syntyy dokumentti, jossa projektin ryhmien väliset riippuvuudet ovat selkeästi nähtävillä. Pyrähdyksen aikana pidetään katselmuksia, joissa riippuvuuksia ja mahdollisia muutoksia seurataan. (Babinet & Ramanathan, 2008)

Tilanteista, joissa ryhmät omatoimisesti, keskenään yhteistyössä selvittävät riippuvuuksista mahdollisesti koituvat ongelmat, selvitään yleensä menestyksekkäästi. Vastaavasti yhteistyön ja omatoimisuuden puuttuessa kohdataan suuria ongelmia. (Moore & Spens, 2008)

Salesforce.comilla on virtuaalinen arkkitehtuurista vastaava ryhmä, joka koostuu muiden ryhmien jäsenistä. Virtuaalinen ryhmä huolehtii, että järjestelmän arkkitehtuuri säilyy ehjänä. Merkittävät, arkkitehtuuriin olennaisesti vaikuttavat muutokset tulee hyväksyttää virtuaalisella ryhmällä, jotta vaikutukset järjestelmään tai muiden ryhmien työhön ovat selvillä ja ryhmä kykenee koordinoitumaan muiden ryhmien kanssa. (Babinet & Ramanathan, 2008) Myös Knibergin raportissa mainitaan, että arkkitehtuurin vaatiessa muutoksia he väliaikaisesti nimittivät yhden henkilön vastuualueeksi huolehtia teknisistä muutoksista arkkitehtuuriin ja listasivat arkkitehtuurin muutokset arkkitehtuurin työlistaan (Kniberg & Farhang, 2008). Muita ryhmäkoordinoitia helpottavia tehtäviä ovat scrumien scrum-kokous (engl.

scrum-of-scrums) ja kevyt viikkoraportti, jossa kerrotaan riippuvuuksista, riskeistä ja mahdollisista esteistä. (Babinet & Ramanathan, 2008)

Riippuvuuksien hallinta on keskeinen ongelma suurissa projekteissa, jotka hyödyntävät ketteriä menetelmiä. Ongelman ratkaisut perustuvat joko riippuvuuksien eliminoimiseen kokonaan tai ryhmien välisen yhteistyön ja aloitteellisuuden korostamiseen. Lähdemateriaalin perusteella näyttäisi siltä, että parhaaseen lopputulokseen päästään, kun pyritään eliminoimaan riippuvuudet ja jäljelle jäävät ratkaistaan ryhmien välisellä yhteistyöllä. Ketterien menetelmien mahdollistaman muutoksiin reagoimisen vuoksi riippuvuuksia tulee myös seurata jatkuvasti. Lisäksi pienissä projekteissa hyödyllinen ulkoisten häiriötekijöiden huomiotta jättäminen ja keskittyminen vain oman ryhmän toimiin ei ole mahdollista suuren mittakaavan projektissa (Moore & Spens, 2008).

3.1.5 Jatkuva integrointi

Jatkuva integrointi (engl. continous integration) on käytäntö, jossa valmis koodi integroidaan välittömästi muuhun koodiin, jolloin saadaan uusi ohjelmakooste. Integroitaessa kaikki testit ajetaan ja koodin tulee läpäistä ne kaikki, jotta se hyväksytään. Järjestelmä kasataan useita kertoja päivässä. Myös jatkuva integrointi on XP:n käytäntö. (P. Abrahamsson et al., 2002)

Käytännössä on huomattu, että esimerkiksi joka yö tapahtuva kasaus ei ole riittävä. Leen raportissa mainitaan, että yksikkötestejä ei läpäisty oikein juuri koskaan ja palautteen saamisessa kesti yksi päivä (Lee, 2008). Ohjelmistokoosteen rakentaminen käsin on aikaavievää ja virhealtista (Kniberg & Farhang, 2008). Jatkuva integrointi kannattaa automatisoida, jotta palaute saadaan nopeasti ja korjaavat toimet voidaan aloittaa. Automatisoitu integrointi vähentää integrointiin käytettävää aikaa ja työmäärää merkittävästi, lisäksi automaattiset testit parantavat laatua (Babinet & Ramanathan, 2008). Mitä

suuremmaksi ohjelmisto kasvaa ja mitä enemmän sen eri osien välillä on riippuvuuksia sitä suurempi merkitys toimivalla, automaattisella integrointiprosessilla on (Moore & Spens, 2008). Jatkuva integrointi helpottaa vakaan tahdin ylläpitämistä ja toisaalta auttaa pienentämään riippuvuuksista johtuvia ongelmia (Kniberg & Farhang, 2008)..

Lindvallin raportissa mainitaan, että jatkuva integrointi paransi ohjelmistoon tehtävien muutosten laatua, koska ohjelmakoosteeseen tulleet virheet havaittiin nopeasti (Lindvall et al., 2004). Lee huomauttaa, että nopea palautteen saaminen on tärkeää, joten ohjelmakoosteen tilasta tulee olla tarvittava tieto helposti, nopeasti saatavilla. Rikkinäisen ohjelmakoosteen korjaaminen on tärkein tehtävä (Lee, 2008). Moore raportoi, että kun ryhmiä ei saatu sitoitumaan rikkinäisten ohjelmakoosteiden korjaamiseen vaadittiin johdon puuttuminen projektin toimiin ja lähdekoodin jäädyttäminen. Lopulta jouduttiin luomaan erillinen ohjelmakoosteesta vastaava ryhmä, jotta ohjelmakoosteet läpäisivät testit ja muiden ryhmien työ saattoi jatkua (Moore & Spens, 2008).

Lindvallin raportissa myös kerrottiin jatkuvan integoinnin ja koodin uudelleen muotoilun aiheuttaneen monia ongelmia, kun organisaatiossa oli muutoksenhallintakomitea (engl. change control board, CCB) (Lindvall et al., 2004). Ketterien menetelmien hyödyntäminen ja perinteisiin menetelmiin kuuluvan muutoksenhallintakomitean säilyttäminen vaikuttaa ristiriitaiselta. Koodin uudelleen muotoilu eli refaktorointi (engl. refactoring) tarkoittaa ohjelmakoodin parantamista muun muassa yksinkertaistamalla (P. Abrahamsson et al., 2002).

3.2 Ihmiset

3.2.1 Ryhmä

Leen raportista käy ilmi, että ryhmän ollessa hajautunut eri huoneisiin tai

kerroksiin kommunikointi ja XP-käytäntöjen hyödyntäminen ovat vaikeaa. Kun ryhmä siirtyi työskentelemään sille omistettuun työtilaan, lähekkäin istuvien kommunikaatio osoittautui hyödylliseksi ja ongelmien ratkaiseminen nopeutui, koska ryhmän jäsenet oppivat paitsi keskustelemalla myös kuuntelemalla muita. (Lee, 2008)

Microsoftilla yhden toiminnallisuuden tuottanut ryhmä hajoitettiin ja jäsenistä koottiin uusi ryhmä seuraavaa toiminnallisuutta varten. Myöhemmin kuitenkin havaittiin, että ryhmän säilyttäminen ennallaan tehostaa toimintaa etenkin, jos ryhmän tehtävät koskevat koko projektin ajan samaa osa-aluetta. (Miller & Carter, 2007)

3.2.2 Oikeat ihmiset – asenne vai kokemus

Mooren mukaan ketterät menetelmät poikkeavat perinteisistä ohjelmistokehitysmenetelmistä merkittävästi. Voidaan väittää, että ketterät menetelmät vaativat aivan erilaista osaamista ja erilaisia kykyjä kuin perinteiset menetelmät. Moore havaitsi, että juuri oikeiden ihmisten löytäminen oli suurin haaste ketteriä menetelmiä hyödyntävässä, suuressa ohjelmistoprojektissa. Kokemus ketteristä menetelmistä ei takaa, että yksilö soveltuu suuren mittakaavan ketterään projektiin. Sen lisäksi, että henkilö sisäistää ketterien menetelmien keskeiset periaatteet ja käytännöt, hänen tulee osata ajatella projektia myös oman ryhmänsä ulkopuolella ja tehtä töitä yhteisen hyvän eteen. Suuren mittakaavan ketterä projekti vaatii oman ajattelutapansa. (Moore & Spens, 2008)

Moore havaitsi, että suuren mittakaavan projektissa menestyksekkäitä olivat ryhmät, joiden jäsenillä oli tiettyjä ominaisuuksia. Näitä ominaisuuksia ovat kyky ajatella ja toimia oman ryhmän rajojen ulkopuolella, kyky omistautua projektiin parhaan lopputuloksen aikaan saamiseksi kuin yrittäjä, tahto tuoda julki ikävät tosi asiat ja käsitellä ne loppuun asti sekä kyky hyväksyä muutokset

ja luottaa muiden tekemiin päätöksiin, joihin itse ei ole voinut vaikuttaa. (Moore & Spens, 2008)

3.2.3 Pariohjelmointi

Pariohjelmoinnissa (engl. Pair programming) kaksi ihmistä ohjelmoi yhdessä (P. Abrahamsson et al., 2002). Se on erinomainen tapa jakaa tietoa ryhmän sisällä mutta toisaalta se usein herättää aluksi muutosvastarintaa.

Leen raportissa pareja seurattiin matriisilla, josta näkyi ongelmalliset parit ja hyvin toimivat parit. Matriisin avulla pyrittiin eliminoimaan ongelmat ja toisaalta hajauttamaan hyvin toimiviakin pareja tiedon siirtymisen vuoksi. Käytännössä pareista silloisen alueen expertti tukee oppivassa osassa olijaa, joka kirjoittaa varsinaisen koodin. Hyvin toimivaksi käytännöksi havaittiin myös, että parista ensimmäinen huolehtii yksikkötestin ja toinen ohjelmakoodin kirjoittamisesta. (Lee, 2008)

4 YHTEENVETO

Ketteriä menetelmiä pyrkivät nykyään hyödyntämään niin pienet kuin suuretkin organisaatiot. Ketterien menetelmien skaalaamiseksi ei ole olemassa hopealuotia mutta lähdemateriaalista löytyi kirjava joukko tapoja, joilla ketteriä menetelmiä on onnistuneesti skaalattu suurten projektien tarpeisiin. Jokainen organisaatio skaalasi menetelmiä omalla tavallaan mutta perusongelmat ja niiden ratkaisut olivat periaatteessa samoja organisaatiosta riippumatta. Skaalattaessa havaittiin kaksi osa-aluetta, jotka tuottivat ongelmia.

Suurimmat haasteet ketterien menetelmien hyödyntämisessä suurissa projekteissa juontuvat ryhmien hallinnoinnista. Suuri määrä ryhmiä vaikeuttaa näiden koordinointia, riippuvuuksien hallintaa ja monimutkainen ohjelmisto ohjelmakoosteiden kasaamista. Scrumien scrum-kokous on ensimmäinen askel ryhmien hallinnoimiseen. Havaittujen ongelmien ratkaisu ja ennaltaehkäiseminen ryhmien välisellä yhteistyöllä nousee keskeiseen asemaan yhdessä sujuvasti rullaavien tukitoimintojen, kuten automatisoidun jatkuvan integroinnin, kanssa. Ryhmien yhteistyötä voi helpottaa pyrähdysten synkronoinnilla siten, että pyrähdysten alussa ja lopussa on helppo järjestää kaikkia ryhmiä koskevat kokoukset. Suurissa projekteissa ohjelmiston koko tuo myös ongelmia mukanaan. Suuri koko vaikeuttaa tuotteen työlistan ylläpitämistä ja sen järkevää esittämistä. Tästä havaittiin, että tuotteen työlistan on oltava fyysinen objekti, jotta sitä on helppo hallinnoida. Työlistan yksityiskohtien tulee palvella tarkoitusta, joten ryhmillä voi olla tarkemmat tehtävien kuvaukset työlistassa kuin projektin tuotteen omistajalla.

Toinen tärkeä tekijä ovat ihmiset ja heidän muodostamansa ryhmät. Oikeanlaisten ihmisten löytäminen suuriin ketteriä menetelmiä hyödyntäviin ohjelmistoprojekteihin voi olla haastavaa, koska sellaisiin sopivien ihmisten käytöstä ja ominaisuuksia ei pidetä sopivina perinteisissä

ohjelmistoprojekteissa. Suurissa projekteissä, joissa ketteriä menetelmiä hyödynnetään, tulisikin ryhmien jäsenillä olla tiettyä yrittäjähenkeä, jotta he sitoutuvat työhön ja työskentelevät omatoimisesti tarpeen vaatiessa. Kuten ketterissä menetelmissä tavallisesti myös suuriin projekteihin skaalattuna ryhmän tulee työskennellä yhteisessä tilassa, jossa kommunikaatio on helppoa. Ryhmän hitsautuminen toimivaksi kokonaisuudeksi on myös tärkeää, joten ryhmän jäsenten vaihtaminen tai ryhmän hajottaminen ei ole suositeltava vaihtoehto. Edelleen ryhmän toimintaa voi tehostaa esimerkiksi seuraamalla matriisin avulla toimivia pariohjelmointi pareja.

Tässä tutkielmassa havaittiin kaksi merkittävää osa-aluetta, jotka tuottavat haasteita ketteriä menetelmiä skaalattaessa. Sekä ryhmien hallintaan että ihmisiin liittyen löydettiin monia ratkaisuja, joita esiteltiin edellä. Jatkossa voitaisiin tutkia, löytyykö ketterien menetelmien skaalaamiseen liittyen muita merkittäviä osa-alueita, jotka tässä tutkielmassa vielä jäivät käsittelemättä. Toisaalta voisi olla tarpeellista osoittaa empiirisesti tässä tutkielmassa esiteltyjen havaintojen todenperäisyys kattavammassa otannassa, koska tässä tutkielmassa lähdemateriaalina ei ollut empiiristä tutkimusta vaan se painottui käytännön kokemuksiin eikä mittauksiin.

LÄHTEET

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile software development methods: review and analysis. VTT Electronic). *VTT Publications,* (478)

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

Agile Alliance. (2001). *Agile manifesto.* http://agilemanifesto.org/

Airaksinen, T. (2004). Ketterät menetelmät. (Bachelor's thesis, University of Jyväskylä).

Babinet, E., & Ramanathan, R. (2008). *Dependency management in a large agile environment.* Agile, 2008. AGILE '08. Conference

Beck, K. (1999). *Extreme programming explained: Embrace change*. Reading, Mass.,: Addison-Wesley.

Cohn, M. (2007). *Advice on conducting the scrum of scrums meeting.* http://www.scrumalliance.org/articles/46-advice-on-conducting-the-scrum-of-scrums-meeting

Grewal, H., & Maurer, F. (2007). *Scaling agile methodologies for developing a production accounting system for the oil & gas industry.* AGILE 2007

Jacob, J., Rodriquez, M. & Barry, G. *Microsoft team foundation server branching guidance.* Retrieved 18 May, 2007, from http://www.codeplex.com/BranchingGuidance

Kniberg, H., & Farhang, R. (2008). *Bootstrapping scrum and XP under crisis A story from the trenches.* Agile, 2008. AGILE '08. Conference

Lan Cao, & Ramesh, B. (2007). *Agile software development: Ad hoc practices or sound principles?.* IT Professional

Lee, E. C. (2008). *Forming to performing: Transitioning large-scale project into agile.* Agile, 2008. AGILE '08. Conference

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

Lyon, R., & Evans, M. (2008). *Scaling up pushing scrum out of its comfort zone.* Agile, 2008. AGILE '08. Conference

Mann, C., & Maurer, F. (2005). *A case study on the impact of scrum on overtime and customer satisfaction.* Agile Conference, 2005. Proceedings

Miller, A., & Carter, E. (2007). *Agility and the inconceivably large.* AGILE 2007

Moore, E., & Spens, J. (2008). *Scaling agile: Finding your agile tribe.* Agile, 2008. AGILE '08. Conference

Reifer, D. J. (2002). How good are agile methods? *Software, IEEE, 19*(4), 16-18.

Reifer, D. J., Maurer, F., & Erdogmus, H. (2003). Scaling agile methods. *Software, IEEE, 20*(4), 12-14.

Salo, O., & Abrahamsson, P. (2008). *Agile methods in european embedded software development organisations: A survey on the actual use and usefulness of extreme programming and scrum.* Software, IET

Schwaber, K., & Beedle, M. (2001). *Agile software development with scrum.*

Prentice Hall PTR.

Spayd, M. K. (2003). *Evolving agile in the enterprise: Implementing XP on a grand scale.* Agile Development Conference, 2003. ADC 2003.

LIITE - KÄÄNNÖKSET

aloituspalaveri = release kick-off

haaroittaminen = branching

jatkuva integrointi = continous integration

ketterät menetelmät = agile methods

muutostenhallintakomitea = change/configuration control board

ohjelmakooste = build

ongelmalista = problem/issue backlog

pyrähdyksen katselmus = sprint review meeting

pyrähdyksen suunnittelukokous = sprint planning meeting

pyrähdyksen työlista = sprint backlog

pyrähdys = sprint

päivittäinen scrum-kokous = daily scrum meeting

ryhmän vetäjä = team lead

scrumien scrum-kokous = scrum of scrums

scrum-ohjaaja = scrum master

scrum-ryhmä = scrum team

tarinakortti = story card

tarinaseinä = story wall

testilähtöinen ohjelmistokehitys = test-driven development

tuotteen omistaja = product owner

tuotteen työlista = product backlog