Scrumban-menetelmän käyttö ketterässä ohjelmistokehityksessä

Kalle Ilves

Kandidaatintutkielma
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Helsinki, 25. huhtikuuta 2016

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author
Kalle Ilves

Työn nimi — Arbetets titel — Title

Scrumban-menetelmän käyttö ketterässä ohjelmistokehityksessä

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Kandidaatintutkielma	25. huhtikuuta 2016	23

Tiivistelmä — Referat — Abstract

Ketterien menetelmien periaatteisiin kuuluvat yksilöiden arvostaminen, asiakasyhteistyö ja muutokseen reagointi. Ne kehitettiin vaihtoehdoksi perinteisille dokumentaatiokeskeisille sekä raskasrakenteisille tavoille tuottaa ohjelmistoa. Ketterien menetelmien suosio on ollut 2000-luvun alussa suuressa kasvussa. Suuren suosion saavuttaneita menetelmiä ovat Scrum, Kanban sekä niiden yhdistelmät, kuten Scrumban.

Scrumban on ketterän ohjelmistokehityksen menetelmä, joka yhdistää Scrumin ja Kanbanin piirteitä. Scrumban soveltuu prosesseihin, jotka ovat arvaamattomia ja nopeasti muuttuvia. Tällaisissa projekteissa Scrumin periaatteiden noudattaminen on osoittaunut hankalaksi, mutta samaan aikaan Kanban asettaa prosessille liian vähän rajotteita.

Tässä tutkielmassa tutustutaan ketteriin menetelmiin sekä esitellään ketterän ohjelmistokehityksen menetelmistä Scrum ja Kanban. Tämän jälkeen käydään läpi Scrumin ja Kanbanin yhtäläyisyyksiä sekä eroja. Yhtäläisyyksien ja erojen läpikäynnin jälkeen tarkastellaan Scrumbanin saamia vaikutteita Scrumista ja Kanbanista. Lopuksi kerrotaan Scrumbanin käytöstä kahden tapaustutkimuksen avulla ja eritellään käytöstä havaittuja hyötyjä ja käyttöön liittyviä haasteita.

ACM Computing Classification System (CCS):
Software and its engineering Agile software development

Avainsanat — Nyckelord — Keywords
ketterän ohjelmistokehityksen menetelmät, kanban, scum, scrumban

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Sisältö

1 Johdanto

Ketterän ohjelmistokehityksen suosio ohjelmistotuotannossa on kasvanut huomattavasti 2000-luvun alussa [Lin12]. Vuonna 2013 ohjelmistotuotannon ammattilaisille tehdyssä kyselyssä eniten suosiota ketteristä menetelmistä on saavuttanut Scrum, Kanban sekä niiden yhdistelmät, kuten Scrumban [Ver13]. Samaan aikaan perinteisempien menetelmien, kuten Vesiputousmallin suosio on laskussa. Yhtenä tekijänä Vesiputousmallin suosion laskuun voidaan pitää epäonnistuneiden projektien suurempaa lukumäärää ketteriä menetelmiä noudattaviin projekteihin verrattuna [Ver13]. Vuonna 2012 julkaistun raportin mukaan vain 14% Vesiputousmallia noudattavista projekteista onnistui, kun taas ketteriä menetelmiä noudattavissa projekteissa vastaava osuus oli 42% [Ver13].

Tämä kandidaatintutkielma tarkastelee kolmea ketterän ohjelmistokehityksen menetelmää, Scrumia ja Kanbania, sekä niitä yhdistävää menetelmää Scrumbania. Tutkielma esittelee Scrumin ja Kanbanin historiaa ja periaatteita. Scrumin ja Kanbanin erojen määrittelemisen kautta siirrytään käsittelemään Scrumbanin vaikutteita Scrumista ja Kanbanista. Scrumbanin käyttöä ketterässä ohjelmistokehityksessä käsitellään käymällä läpi kaksi tapausesimerkkiä. Tutkimusongelmat on muotoiltu seuraavasti:

- Mitä on ketterä ohjelmistokehitys?
- Mitä on Scrumban? Miten siinä näkyy Scrumin ja Kanbanin erot sekä yhtäläisyydet?
- Miten Scrumbania on sovellettu käytännössä ja mitä kokemuksia siitä on saatu?

Tutkielman rakentuu kolmesta osasta. Toisen luvun alussa annetaan yleiskatsaus ketterästä ohjelmistokehityksesta. Yleiskatsausta seuraa

Scrumin ja Kanbanin käsittely menetelmien historian ja periaatteiden kautta. Kolmas luku määrittelee Scrumin ja Kanbanin eroja ja tuo esille Scrumbanin vaikutteita sekä Scrumista, että Kanbanista. Neljäs luku esittelee kaksi tapausesimerkkiä Scrumbanin soveltamisesta ketterässä ohjelmistokehityksessä. Tapauksien käsittely rakentuu tapauskuvauksista, Scrumbanin käyttöönotosta ja Scrumbanista saatujen hyötyjen käyttöön liittyvien haasteiden esilletuonnista. Tutkielma päättyy yhteenvetoon.

2 Scrum ja Kanban ketterässä ohjelmistokehityksessä

Tässä luvussa esitellään ketterän ohjelmistokehityksen periaatteita sekä tutustutaan Scrumiin ja Kanbaniin. Ensimmäisen aliluvun tarkoitus on selvittää ketterän ohjelmistokehityksen lähtökohdat ja periaatteet. Seuraavat kaksi alilukua käsittelevät Scrumin ja Kanbanin historiaa sekä periaatteita. Kahden viimeisen aliluvun tarkoitus on muodostaa kattavampi kuva ketterien menetelmien käytöstä ja pohjustaa Scrumin ja Kanbanin erojen määrittelemistä.

2.1 Ketterä ohjelmistokehitys

Ketterä ohjelmistokehitys on vaihtoehto perinteisille dokumentaatiokeskeisille sekä raskasrakenteisille tavoille tuottaa ohjelmistoa [man01]. Sen keskeiset arvot esiteltiin vuonna 2001 kirjoitetussa ketterän ohjelmistokehityksen julistuksessa [DC04]. Kirjoittajien kokemuksen perusteella ketterässä ohjelmistokehityksessä tulee arvostaa

- yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja
- toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota

- asiakasyhteistyötä enemmän kuin sopimusneuvotteluja
- reagointia muutokseen enemmän kuin pitäytymistä suunnitelmassa

Yksilöiden ja kanssakäymisen arvostaminen painottaa ohjelmistokehittäjien yhteisöllisyyttä sekä ihmiskeskeisyyttä prosessikeskeisyyden sijaan. Toimivan ohjelmiston arvostamisella tarkoitetaan jatkuvaa uusien toimivien julkaisujen tuottamista tihein väliajoin, mikä vähentää raskasta dokumentaatiota. Vaikka sopimusten merkitys suurissa ohjelmistoprojekteissa on suuri, tulee asiakasyhteistyö asettaa aina sopimuksia suurempaan arvoon. Muutokseen reagoimisella tarkoitetaan sitä, että ohjelmistotuotantoprosessin edetessä kaikkien osapuolien tulee olla valmistautuneita mahdollisiin muutoksiin sekä reagoimaan niihin. [DC04]

2.2 Scrum

Scrum-ohjelmistotuotantomenetelmän tutkimus- ja kehitystyö sai alkunsa 1990-luvun alussa [JS11]. Kehitystyön aloittamisen taustalla vaikuttivat kokemukset, joiden perusteella perinteisen ohjelmistokehityksen menetelmien noudattaminen oli tietyissä projekteissa osoittautunut hankalaksi, jopa mahdottomaksi [DC04]. Peter DeGrace ja Leslie Stahl toivat kirjassaan "Wicked Problems, Righteous Solutions" [DS90] esille perinteisen ohjelmistokehityksen menetelmiin kuuluvan Vesiputousmallin suurimmat ongelmat:

- Vaatimuksia ei täysin ymmärretä ennen kuin projekti on alkanut
- Käyttäjät tietävät, mitä he haluavat vasta kun he näkevät sovelluksen alustavan version
- Sovelluksen vaatimukset muuttuvat usein kehitysprosessin edetessä

- Uudet työkalut ja teknologiat tekevät toteutusmenetelmistä arvaamattomia

DeGracen ja Stahlin julkaisussa esitetään ratkaisuksi Vesiputousmallin ongelmiin *Kaikki kerralla* -mallia (All-at-once). Mallissa oletetaan, että kaikki ohjelmiston osa-alueita, kuten suunnittelua, ohjelmointia ja testaamista tehdään samanaikaisesti. Scrumin skaalautuva, kehitysryhmäkeskeinen *Kaikki kerralla* -malli sai vaikutteita Hirotaka Takeuchin ja Ikujiro Nonakan artikkelista "The New Product Development Game" [JS11]. Artikkelissaan Takeuchi ja Nonaka esittävät ajatuksia itseorganisoituvasta kehitysryhmästä, jossa jokaisella kehitysryhmän jäsenellä on jatkuvasta hyvä yleiskuva ohjelmiston tämänhetkisestä tilasta [TN86]. Itseorganisoituvan kehitysryhmän ajatus on läsnä myös DeGracen ja Stahlin julkaisussa. Siinä onnistunutta kehitysprosessia kuvaillaan rugby-joukkueeksi, jossa itseorganisoituva joukkue pyrkii etenemään kentän poikki tiiviissä yhteistyössä [DS90]. Scrum saikin nimensä rugby-termistä *scrum*, joka viittaa rugbyn aloitusryhmittymiseen [JS11].

Scrum on iteratiivinen ja inkrementaalinen menetelmä. Iteratiivisuus tarkoittaa sitä, että prosessi etenee kiinteän pituisissa osissa eli iteraatioissa. Iteraatioista käytetään Scrumissa nimitystä *sprintti* (sprint). Sprinttien pituus vaihtelee, mutta Ken Schwaber ehdotti alunperin niiden pituudeksi yhdestä kuuteen viikkoa. Inkrementaalisuus taas tarkoittaa sitä, että ohjelmistoon lisätään uusia toiminallisuuksia kehitysjaksojen edetessä ja ohjelmiston ei tarvitse olla valmis ensimmäisen iteraation jälkeen. [JS11]

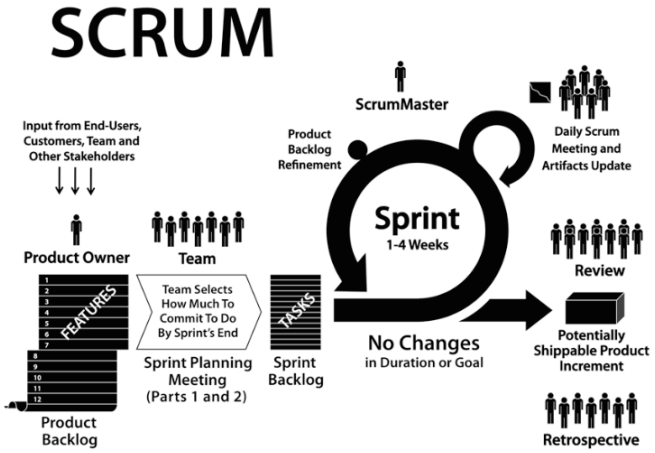
Jokainen sprintti alkaa kehitysryhmän ja asiakkaan välisellä tapaamisella. Tästä tapaamisesta käytetään nimitystä *sprintin suunnittelu* (sprint planning) [JS11]. Sen aikana kehitysryhmä ja asiakas sopivat siitä, mitkä toiminnot kehitysryhmä toteuttaa seuraavan tinsprin aikana. Sprintin suun-

nittelun aikana kehittäjäryhmän tulee myös määrittä *sprintin tavoite* (sprint goal), jonka tulee vastata kysymykseen “Miksi teemme tämän sprintin?” [Kni07].

Sprintin aikana kehitysryhmä pitää *päivätapaamisia* (daily scrum). Tapaamiset ovat lyhyitä, yleensä noin 15 minuutin pituisia ja niiden aikana kehitysryhmän jäsenet keskustelevat työn alla olevista toiminnoista [RF09]. Tapaamisen aikana jokainen kehitysryhmän jäsen kertoo vuorollaan, mitä on tehnyt ja mitä aikoo tehdä seuraavaksi sekä tuo esille mahdollisia esteitä edistymisensä kannalta [JS11].

Jokaisen tinsprin lopussa kehitysryhmän ja asiakkaan välillä pidetään *sprintin arviointitapaaminen*. Sen aikana asiakkaalle esitellään ohjelmistoon sprintin aikana toteutetut toiminnot. Sprintin lopussa pidetään myös *retrospektiivi*, jonka aikana kehitysryhmä ja mahdollisesti asiakas analysoivat sprintin onnistumista tuoden esille parannusehdotuksia työtapoihin tulevia sprinttejä varten. [Kni07]

Kuva 1: Scrum-projektin elinkaari [JS11].



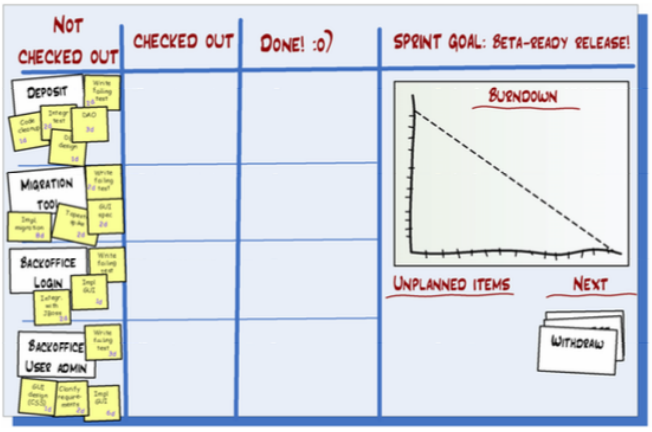
Scrumiin liittyy työn edistymisen säännöllinen tarkastelu sekä tarvittaessa sen sopeuttaminen [JS11]. Tarkastelun ja sopeuttamisen helpotta-

miseksi Scrumissa on määritelty *tuotoksia* (artifacts), joita ovat *tuotteen kehitysjono* (product backlog), *sprintin kehitysjono* (sprint backlog) ja *sprintin edistymiskäyrä* (burndown chart) [Kni07].

Tuotteen kehitysjono on työlista toteutettavan ohjelmiston toimintoja, joita kutsutaan usein *käyttäjätarinoiksi* (user story) [Kni07]. Käyttäjätarinat on listattu asiakkaan antaman prioriteetin mukaan siten, että korkeimman prioriteetin käyttäjätarinat ovat listan kärjessä. Käyttäjätarinaan liitetään toiminnon kuvauksen lisäksi kehitysryhmän tekemä aika-arvio toiminnon toteuttamiseen kuluvasta ajasta [Kni07]. Aika-arviot esitetään yleensä *tarinapisteillä* (story point). Yksi tarinapiste vastaa usein karkeasti yhtä miestyöpäivää [JS11]. Sprintin kehitysjono sisältää kehitysryhmän sprintin suunnittelutapaamisessa valitsemat käyttäjätarinat, jotka kehitysryhmä sitoutuu toteuttamaan sprintin aikana [Kni07]. Käyttäjätarinat jaetaan sprintin kehitysjonossa yksityiskohtaisempiin tehtäviin. Jokaisella tehtävällä on usein tila, joka kertoo sen edistymisestä. Tiloja voi olla esimerkiksi kolme: “kuittaamatta“, “kuitattu“ ja “valmis“. Sprintin edistymiskäyrä kuvaa sprintissä jäljellä olevaa työmäärää ajan funktiona [JS11]. Se rakentuu kehitysryhmän arvioidessa, kuinka paljon heiltä kuluu suurin piirtein aikaa keskeneräisten tehtävien valmistumiseen.

Scrum määrittelee kolmesta kuuteen roolia, jotka yhdessä muodostavat scrum-ryhmän. Ken Schwaber on typistänyt roolien määrän kolmeen: *tuotteen omistajaan* (product owner), *scrum-mestariin* (scrum master) ja *kehittäjään*. Tuotteen omistaja on projektin asiakas tai hänen etujaan edustava henkilö. Hänen vastuullaan on määrittellä tuotteen kehitysjonoon uusia käyttäjätarinoita ja vahvistaa, että sprinttiin valitut käyttäjätarinat on toteutettu halutulla tavalla. Scrum-mestari ohjaa kehitysryhmää prosessin noudattamisessa ja poistaa projektin edistymiseen vaikuttavia esteitä. Ke-

Kuva 2: Esimerkki sprintin kehitysjonosta [Kni07].



hittäjät valitsevat sprintin kehitysjonoon käyttäjätarinoita ja toteuttavat sprintin aikana niissä määritellyt toiminnot. [JS11]

2.3 Kanban

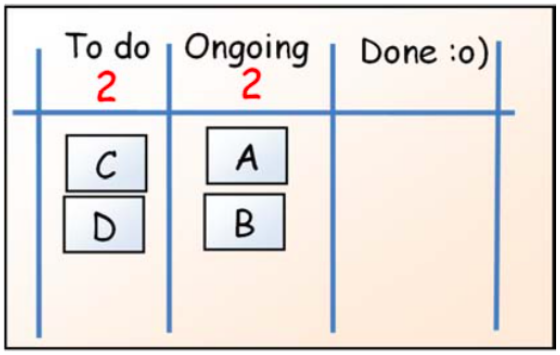
Kanbanin historia ohjelmistotuotantomenetelmänä johtaa juurensa vuoteen 2002 [And10]. Alunperin Kanbania käytettiin tuotantomenetelmänä Toyotan autotehtailla 1950-luvun alussa [JMG03]. Ensimmäinen ohjelmointikehitykseen implementoitu Kanban oli David Andersonin vuonna 2004 Microsoftilla käyttöönottama *Rumpu-Puskuri-Köysi* -menetelmä (Drum-Buffer-Rope), joka on Eliyahu Goldrattin kehittämän *Rajotteiden teorian* (theory of constraints) tuontantosovellus [And10]. Rajotteiden teorian mukaan uuden prosessin tulisi antaa kehittyä eliminoimalla yksi pullonkaula kerrallaan. *Rumpu-Puskuri-Köysi* -menetelmää pyrkii löytämään pullonkaulan ja sen jälkeen lievittämään sitä kunnes se ei enää rajoita tehokkuutta. Sen jälkeen ilmestyy uusi pullonkaula ja sykli toistuu [Gol90]. Anderson ja Rick Garber esittelivät menetelmensä vuonna 2007 *Lean New Product Development* konferenssissa nimellä Kanban. Syynä siirtymisessä Rumpu-Puskuri-Köysi -menetelmästä Kanbaniin oli Kanbanin parempi kyky toipua pullonkaulatilanteiden aiheutta-

mista katkoista. Kanban oli heidän mielestään myös helpompi sanoa, selittää, opettaa ja toteuttaa kuin Rumpu-Puskuri-Köysi -menetelmä [And10].

Kanban ohjaa kehitysryhmää visualisoimaan työnkulkua, rajoittamaan keskeneräisen työn määrää ja jatkuvasti kehittämään käytänteitään [IPF+11]. Andersonin mukaan Kanban kannustaa kehitysryhmää laatimaan tilannekohtaisia ratkaisuja prosessiin sen sijaan, että se kannustaisia seuraamaan tarkasti tiettyä prosessia [And10]. Projektia ei jaeta iteraatioihin, vaan kehitysryhmä voi itse päättää milloin suunnitella tuotetta, keskustella prosessin kehittämisestä ja julkaista tuotteen valmiit toiminnot [HK09].

Työnkulun visualisointi on Kanbanissa tärkeää. Visualisoinnin keskeisessä osassa on *kanban-taulu*, joka sisältää kehitysryhmän tehtävät [IPF+11]. Tehtävät esitetään kortteina, joista käytetään myös nimitystä *kanban*. Kortit siirtyvät kanban-taululla eri jonojen välillä, kunnes ne lopulta valmistuvat. Jonoja voivat olla esimerkiksi “kuittaamatta”, “kuitattu” ja “valmis” [HK09]. Jonoilla on nimen lisäksi koko, jonka avulla rajoitetaan jonossa olevien korttien määrää. Asiakas voi lisätä jatkuvasti uusia kortteja “kuittaamatta”-jonoon, joka toimii puskurina kuittaamattomien ja kuitattujen tehtävien välillä [HK09].

Kuva 3: Esimerkki kanban-taulusta. Jonojen kohdalla olevat numerot kuvaavat niiden kokoa [HK09].



Kanbanissa keskeneräisen työn määrää rajoitetaan kanban-taulun jonojen kokojen määrittelemisellä. Kun jonon korttien määrä saavuttaa sille määritellyn koon, ei jonoon saa lisätä kortteja ennen kuin jokin sen korteista siirtyy siitä pois. Kanban ei määrittele maksimiarvoa keskeneräiselle työmäärälle, vaan kehitysryhmä asettaa sen itse ja muuttaa sitä tarpeen mukaan [HK09]. Jonon koon muuttaminen vaikuttaa *jaksonaikaan* (cycle time), joka kuvastaa aikaa tehtävän aloittamisen ja sen valmistumisen välillä. Lisäksi se vaikuttaa *läpimenoaikaan* (lead time), joka kuvastaa aikaa tehtävän lisäämisestä "kuittaamatta"-jonoon sen valmistumiseen [IPF+11]. Andersonin mukaan suuri jonon koko johtaa pidempään jaksonaikaan, joka johtaa huonolaatuisempaan tuotteeseen [And10]. Pieni jonon koko taas pakottaa kehitysryhmän reagoimaan ongelmiin nopeammin. Sen vuoksi pieni jonon koko johtaa lyhyempään jaksonaikaan [HK09] ja Andersonin mukaan parempilaatuiseen tuotteeseen [And10].

Kanbaniin kuuluu muutamia ketterän kehityksen mukaisia aktiviteetteja. *Päivittäiset tapaamiset* (daily standup meetings) ovat Scrumin päivätapaamisten kaltaisia tapaamisia, jossa jokainen kehitysryhmän jäsen vastaa kolmeen kysymykseen: "mitä saavutit eilen?", "mitä teet tänään?" ja "onko sinulla ongelmia, tai tarvitseko apua?". Tapaamisissa käytetään kanban-taulua, joka antaa vastauksia tapaamisten aikana esitettäviin kysymyksiin. Päivittäistä tapaamista seuraa usein heti *jälkitapaaminen* (after meeting), jonka aikana pienet ryhmät keskustelevat prosessin kehittämisestä [HK09]. Andersonin mukaan jälkitapaamiset ovat tärkeä aktiviteetti prosessin kehittyvän luonteen kannalta [And10].

Priorisointitapaamiset (prioritization meetings) ovat tasaisin väliajoin järjestettäviä tapaamisia, joiden aikana on tarkoitus priorisoida kanban-taulun kuitattujen tehtävien jonoa. Tapaamiseen tulee osallistua ainakin

asiakkaan edustaja ja kehitysryhmä, mutta myös muiden asianomaisten läsnäolo on suotavaa. Priorisointitapaamisten aikana jonojen kokoja muutetaan, joten on tärkeää, että tapaamisia pidetään mahdollisimman tihein väliäajoin [HK09]. Anderson suosittelee, että niitä järjestetään viikottain [And10].

Valmiin ohjelmiston julkaisusta päätetään *julkaisutapaamisissa*, joita pidetään myös tasaisin väliajoin. Tapaamisten aikana päätetään siitä, mitkä osat ovat valmiina julkaisuun ja mitä niiden julkaisemiseksi täytyy tehdä. Julkaisutapaamisen lopputuloksena on julkaisusuunnitelma, joka sisältää suoritusjärjestyksessä olevan sarjan toimenpiteitä [And10].

3 Scrumbanin kehitys Scrumista ja Kanbanista

Nopeasti vaihtuvat vaatimukset tuottavat vaikeuksia Scrumin aikarajoitteisten iteraatioiden noudattamisessa, mutta toisaalta Kanban antaa kehitysryhmälle liikaa vapauksia [Ter12]. Arvaamattomaan kehitysympäristöön on kehitetty Scrumin ja Kanbanin periaatteita yhdistävä menetelmä Scrumban. Tässä luvussa määrittellään Scrumin ja Kanbanin yhtäläisyyksiä ja eroja sekä tarkastellaan Scrumbanin periaatteita ja sen saamia vaikutteita Scrumista ja Kanbanista.

3.1 Scrumin ja Kanbanin yhtäläisyyksiä sekä eroja

Scrum ja Kanban noudattavat ketterää- sekä Lean-ajattelutapaa, jonka tärkein arvo on turhan työn minimointi prosessissa. Molemmat menetelmät korostavat nopeaa muutoksen vastaamista, joka on yksi ketterien menetelmien kulmakivistä. Lean-ajatteluvasta molemmat ovat omaksuneet *Juuri Oikeaan Tarpeeseen* -periaateen (Just-In-Time), jonka nojalla kehitysryhmä “vetää” uutta työtä ollessaan valmiita sen sijaan, että sitä “työnnetään” heille. [HK09]

Scrum ja Kanban noudattavat Lean-ajattelutavan mukaista empiirisyyden ja optimisaation kautta syntyvää jatkuvan kehityksen periaatetta, josta käytetään Lean-terminologiassa nimitystä *Kaizen*. Kaizen-periaatteen nojalla prosessin on tarkoitus muodostua kokeilemisen ja kokemuksen kautta [HK09]. Sekä Andersonin, että Sutherlandin mukaan jokainen ohjelmistotuotantoprosessi on uniikki, joten käytänteitä täytyy mukauttaa tarpeen mukaan [JS11, And10].

Kanbanissa empiriisyys näkyy kehitysryhmän määrittäessä kanbantaulun jonojen kokoja [And10]. Aluksi jonojen kokojen määrittäminen on arvioiden varassa, mutta läpimenoaikaa tarkkailemalla kokoja voidaan muutta. Jos jonojen koot ovat liian pieniä, kehitysryhmässä on jäseniä, joilla ei ole mitään tehtävää. Tällöin läpimenoaika on korkea ja jonojen kokoja kannattaa nostaa matalamman läpimenoajan saavuttamiseksi [HK09].

Scrumissa taas jokainen sprintti on “musta laatikko”, joka vaatii ulkoista ohjausta [JS11]. Kehitysryhmä ei voi tietää varmasti, kuinka monta käyttäjätarinaa he ehtivät toteuttaa sprintin aikana. He tietävät sen vasta, kun heillä on kokemusta edellisistä sprinteistä [HK09]. Kun sprintissä toteutettujen tarinapisteiden määrä on tiedossa, kehitysryhmä pystyy määrittämään *nopeutensa* (velocity). Nopeus arvioi kuinka monta käyttäjätarinaa kehitysryhmä pystyy toteuttamaan sprintin aikana [JS11].

Taulukko 1: Scrumin ja Kanbanin eroja [HK09].

Scrum	Kanban
Iteraatiot ovat aikarajoitettuja.	**Aikarajoitetut iteraatiot ovat mahdollisia.** Kehitysryhmä päättää itse milloin eri aktiviteetteja järjestetään.
Kehitysryhmä sitoutuu tiettyyn työmäärään iteraatiokohtaisesti.	**Sitoutuminen on valinnaista.**
Käynnissä olevaan iteraatioon ei voi lisätä toteutettavia toimintoja.	**Toteutettavia toimintoja voi lisätä jatkuvasti,** kunhan niiden määrä ei riko kanban-taulun "kuittaamatta"-jonolle asetettua kokoa.
Määrittelee kolmesta kuuteen roolia.	**Ei määrittele rooleja.**
Keskeneräisen työn määrää rajoitetaan sprintin sisällä.	**Keskeneräisen työn määrää rajoitetaan kanban-taulun jonojen kokojen määrittelemisellä.**
Sprintin kehitysjonon omistaa yksi kehitysryhmä	**Useammat kehitysryhmät tai yksilöt voivat jakaa saman kanban-taulun.**
Työmäärää arvioidaan ja työmäärän seuraamiseen käytetään sprintin edistymiskäyrää.	**Työmäärän arviointi on valinnaista**, eikä työmäärän seuraamiseen määritellä erityistä kaaviota.
Nopeutta käytetään suunnittelun ja prosessikehityksen mittarina	**Läpimenoaikaa** käytetään suunnittelun ja prosessikehityksen mittarina
Priorisoitu työlista (tuotteen kehitysjono).	**Priorisointi on valinnaista.**

3.2 Scrumbanin vaikutteet Scrumista ja Kanbanista

Scrumille ja Kanbanille on yhteistä se, että ne sallivat eri menetelmien periaatteiden yhdistelemisen ja mukauttamisen erilaisiin ympäristöihin. Käytetyssä menetelmässä saattaa siis olla sekä Scrumin, että Kanbanin piirteitä. Tällöin menetelmästä käytetään nimitystä *Scrumban* [NKMS12]. Scrumbanin tärkein vaikuttaja on Corey Ladas, joka työskenteli Andersonin kanssa Kanbanin kehittämiseksi 2000-luvun alussa.

Ladasin mielestä Kanban ei ole prosessi vaan työkalu, joka ilmentää periaatetta. Kanbania voidaan käyttää prosessin muodostamiseen, joka määrittyy jokaiselle ongelmalle erikseen käytössä oleviin resursseihin pohjautuen. Kanban on Ladasin mukaan työkalu, jota käytetään muiden työkalujen tapaan ongelmanratkaisuun.

Työnkulun virtauksen hallintaan Ladas esittää kanban-taulun jonojen kokojen asettamista, jonka avulla työmenetelmässä saavutetaan joustavuuden ja kontrollin tasapaino. Kanban-taulun yksittäisten tehtävien työmäärän arviointi ei ole tärkeää, vaan tehtävän keskimääräinen työmäärän arviointi. Ladasin mielestä suurin osa Scrumissa tehdyssä työmäärän arvioinnista on turhaa.

Aikarajoitetut iteraatiot eivät Ladasin mielestä sovellu prosesseihin, jotka ovat arvaamattomia ja nopeasti muuttuvia. Työnkulun tulee antaa edetä jatkuvan suunnittelun kautta ilman iteraatioiden suunnittelua. Suunnittelua tehdään tasaisin väliajoin ja siihen liittyy Scrumin mukainen arviointitapaaminen ja retrospektiivi. Päivittäin järjestetään lyhyitä, Scrumin päivätapaamisten mukaisia kehitysryhmän välisiä tapaamisia.

Pitkäaikaista suunnittelua varten Scrumbanissa käytetään *ämpärin kokoista suunnittelua* (bucket size planning). Siinä suunnitelmat jaetaan neljään *ämpäriin*, joista jokainen kuvastaa tietyn aikavälin tavoitteita. Ensimmäinen ämpäri on *yhden vuoden ämpäri* ja se sisältää pitkän aikavälin suurpiirteisiä suunnitelmia. Toinen ämpäri on *kuuden kuukauden ämpäri* ja se sisältää tarkennettuna tärkeimmät vaatimukset. Kolmas ämpäri on *kolmen kuukauden ämpäri* ja siinä tavoitteet on määritelty kättäjätarinoina. Neljäs ämpäri on *nykyinen ämpäri* ja se sisältää käyttäjätarinat jaettuna yksityiskohtaisempiin tehtäviin. Kehitysryhmä valitsee tehtäviä kanban-tauluun kykyisestä ämpäristä. [Mis14]

4 Esimerkkejä Scrumbanin käytöstä

Tässä luvussa esitellään kaksi esimerkkiä Scrumbanin käyttöönotossa ohjelmistotuotantoprojekteissa. Tapausesimerkit on valittu neljän Scrumbanin käyttöön kohdistuvan tapaustutkimuksen joukosta. Tutkielmaan valitut kaksi tapausesimerkkiä valittiin niin, että ne tarjoavat katsauksen mahdollisimman erilaisiin projekteihin, tuovat esille Scrumbanin käyttöön liittyvät hyödyt ja haitat, ja luovat hyvän yleiskuvan Scrumbanin käytöstä sekä Scrumban-prosessin muodostamisesta. Tapaussimerkeissä kerrotaan ensin lähemmin tapauksesta, sen jälkeen Scrumbanin käyttöönotosta ja lopuksi saavutetuista hyödyistä sekä mahdollisista haasteista. Ensimmäisessä tapauksessa kerrotaan Scrumbanin käytöstä laajassa hallinnointi projektissa. Toisessa tapauksessa on kysymyksessä siirtyminen Scrumista Scrumbaniin. Luvun lopussa esitellään päätelmiä tapauksista.

4.1 Scrumista Scrumbaniin: Tapaustutkimus prosessisiirtymästä

Tutkimuksen kohteena oli keskikokoinen ruotsalainen sisällönhallintajärjestelmää kehittävä ja ylläpitävä yhtiö. Yhtiön ohjelmistokehitys oli jakautunut maantieteellisesti Tukholmaan ja Hanoihin. Ruotsissa toimi yksi kehitysryhmä ja Vietnamissa yksi kehitys- ja yksi testausryhmä. Kehitys- ja testausryhmät koostuivat 15-20 työntekijästä. Yhtiö oli suorittanut aikaisemmin prosessisiirtymän Ruotsissa sijaitsevassa toimistossaan. Kommunikaatio-ongelmien välttämiseksi Vietnamissa sijaitsevassa toimistossa toteutettiin prosessisiirtymä Scrumista Scrumbanin vuonna 2011 helmikuu-heinäkuu-välillä. [NKM11]

Vietnamin ryhmien prosessimuutos jaettiin kolmeen peräkkäiseen vaiheeseen: esisiirtymä (pre-transition), prosessisiirtymä (process transition) ja jälkisiirtymä (post-transition). Esisiirtymävaiheessa yritys suunnitteli ja

valmistautui prosessin läpivientiin, prosessisiirtymävaiheessa tehtiin suurimmat muutokset sekä otettiin käyttöön uusi prosessimalli ja jälkitransitiovaiheessa suoritettiin mallin hienosäätöä ja jatkuvaa kehittämistä. [NKM11]

Esisiirtymäviaheessa yhtiössä oli tullut esiin 12 prosessiin liittyvää ongelmaa. Ongelmia olivat muun muassa avoimuuden puute, kommunikaation heikkous, hidas palaute johdolta sekä puutteita prosessin tuntemisessa ja siihen sitoutumisessa. [NKM11]

Scrumbaniin tähtäävän prosessisiirtymän taustalla vaikuttivat Henrik Knibergin ja Mattias Skarinin näkemykset Kanbanista, jotka Natalja Nikitinan ja Mira Kajko-Mattsson tiivistivät kolmeen pääkohtaan:

- Työnkulun visualisointi
- Keskeneräisen työn määrän rajoittaminen
- Läpimenoajan mittaaminen

Prosessisiirtymävaiheen aikana suoritettiin yhteensä kuusi kehitysprosessiin vaikuttanutta muutosta. Ensimmäinen muutos oli siirtyminen Scrumin kehitysjonoista kanban-tauluun. Toinen muutos koski kokouskäytäntöjä, jotka muutettiin vastaamaan tavoiteltavan prosessimallin periaatteita. Eniten kokokäytäntöjen muutos vaikutti päivittäisiin tapaamiisiin ja retrospektiiveihin. [NKM11]

Kolmas muutos koski sprinttien lopettamista. Neljäntenä muutoksena uudelleenjärjesteltiin ryhmien välinen kommunikointi. Vietnamilaiset ryhmät pitivät tästä lähtien yhteisen päivittäiset tapaamiset ja lisäksi edustajat molemmista ryhmistä osallistuivat ruotsalaisen ryhmän päivittäisiin tapaamisiin. [NKM11]

Jälkisiirtymävaiheessa suoritettiin vielä kolme muutosta. Seitsemäntenä muutoksena laaditttiin yhteiset ohjelmointiohjeet (coding guidelines).

Kahdeksantena hienosäädettiin kanban-taulun jonojen kokoja. Alunperin jonojen koko oli viisi, jonka jälkeen niiden koko laskettiin kolmeen ja lopulta ne nostettiin neljään. Yhdeksäntenä muutoksena esiteltiin erilaisia ryhmähenkeä parantavia aktiviteettaja, kuten työtapajatyöskentelyä ja yhteisiä lounaita. [NKM11]

Esisiirtymävaiheessa esiin tulleista ongelmista suurin osa pystyttiin ratkaisemaan joko kokonaan tai osittain prosessitransitiovaiheen tai jälkitransitiovaiheen aikana. Ongelmaa hitaasta palautteesta johdolta siirtyminen Scrumista Scrumbaniin ei onnistunut ratkaisemaan. Lisäksi transitio aiheutti uuden ongelman, joka oli prosessin kasvaminen ja monimutkaistuminen. [NKM11]

Nikitina ja Kajko-Mattson pitävät tärkeimpänä oppina prosessisiirtymästä sitä, että Scrumissa kohdatut ongelmat eivät johtuneet Scrumista, vaan sen huonosta käyttöönotosta yhtiössä. He eivät pidä Kanbanin periaatteiden käyttöönottoa ratkaisuna ongelmiin, vaan sen sijaan jatkuvaa prosessitarkastelua ja -kehitystä. Nikitinan ja Kajko-Matsonin johtopäätös on, että kaksi olennaista tekijää prosesseihin liittyvien ongelmien ratkaisussa ovat jatkuvan prosessikehityksen käytännöt sekä hyvin koulutettu ja sitoutunut henkilöstö. [NKM11]

4.2 Scrumban ylläpitoprojektissa

Tutkimuksen kohteena oli viiden ylläpitäjän ryhmä, jotka olivat vastuussa suuren telekommunikaatiokeskuksen ylläpidosta. Ylläpitoryhmän sisällä oli eri taitoja omaavia yksilöitä, joiden kommunikaatiossa oli ongelmia. Puolet ylläpitoryhmän ajasta kului lyhyiden, kahdesta kolmeen minuuttia kestävien tehtävien tekemiseen. Lyhyet tehtävät ja arvaamattomat vaatimukset tekivät projektista monimutkaisen ja epävakaan. [Ter12]

Projektin prosessin muodostaminen alkoi Scrumin käyttöönotolla, koska se oli käytössä yhtiön muilla ryhmillä. Scrum oli projektissa käytössä viiden viikon ajan viikon pituisissa sprinteissä. Scrumin noudattaminen tuotti kuitenkin projektissa ongelmia, joten seuraavaksi otettiin käyttöön Kanban. Kanbanin ja Scrumin kokeilut sekä niissä havaitut ongelmat johtivat menetelmien periaatteiden yhdistelemiseen. Yhdistelyn seurauksena projektissa otettiin käyttöön Scrumban. [Ter12]

Scrumbanin käyttööntosta ylläpitoprojektissa vastasi Katarzyna Terlecka. Prosessissa otettiin käyttöön Scrumin mukainen tuotteen kehitysjono, mutta sprintin kehitysjono korvattiin kanban-taulun mukaisella kehitysjonolla. Kehitysjono koostui kolmesta jonosta: jonot Windows- ja Linux-kehittäjien tehtäville sekä jono valmiille tai hylätyille tehtäville. Tehtävänä oleville tehtäville asetettiin molemmissa jonoissa Kanbanin mukainen jonon koko, joka rajattiin kymmeneen tehtävään. Tehtävien toteuttamiseen kuluvaa aikaa ei arvioitu, vaan ryhmän nopeus muodostui tehtyjen tehtävien määrästä. [Ter12]

Prosessissa otettiin käyttöön myös Scrumin mukaisia rooleja ja aktiviteettaja. Tuotteen omistaja huolehti tuotteen kehitysjonosta tuottaen tehtäviä ryhmälle. Lisäksi käytössä oli scrum-mestaria vastaava rooli, jonka vastuulla oli huolehtia prosessin noudattamisesta. Sprinteistä luovuttiin ja suunnittelua tehtiin päivittäin päivätapaamisia ja sprintin suunnittelu tapaamisia yhdistelevissä tapaamisissa. Tapaamisten aikana ryhmä valitsi uusia tehtäviä tuotteen kehitysjonosta ja arvioi lyhyesti valmiita tehtäviä. Lisäksi tuotteen omistajalla oli tapaamisen aikana mahdollisuus lisätä uusia tehtäviä tuotteen kehitysjonoon. Muita aktiviteettaja olivat arviointitapaamiset sekä retrospektiivit. Arviointitapaamiset järjestettiin tuotteen omistajan aloitteesta, enintään kahden viikon välein. Retrospektiivejä järjestettiin vähintään

kahden viikon välein. [Ter12]

Terleckan mielestä Scrumin epäonnistuminen ei ollut ryhmän sitoutumattomuuden syytä. Sen sijaan syynä oli hänen mielestään Scrumiin liittyvä sprintin suunnittelu, joka osoittautui mahdottomaksi näin arvaamattomassa projektissa. Sprinttien suunnittelusta päivittäiseen suunitteluun siirtyminen auttoi nopeasti muuttuvissa vaatimuksissa. Scrumin sprinteistä luopuminen johti kuitenkin tehottomuuteen joidenkin kehitysryhmän jäsenien osalta. Terleckan mielestä syynä oli säännöllisen palautesyklin puute. [Ter12]

Kanbanin suurimmat ongelmat olivat ryhmän edistymisen tunteen puute, suunnittelun vähäisyys sekä tuotteen omistajan puuttuminen. Tuotteen omistajan kanssa järjestetyt arviointitapaamiset mahdollistivat uusien tehtävien lisäämisen tuotteen kehitysjonoon sekä valmiiden tehtävien arvioinnin, mitkä toivat ryhmälle edistymisen tunteen. [Ter12]

4.3 Päätelmiä tapauksista

Molemmissa tapauksissa prosessin Kanbanin piirteet mukailevat Ladasin näkemyksiä Kanbanista. Yhteisenä piirteinä tapauksissa ovat työnkulun visualisointi kanban-taululla ja keskeneräisen työn määrän rajoittaminen taulun jonojen kokojen määrittelyllä. Lvussa 4.2 esitellyssä ylläpitoprojektissa kanban-taulun lisäksi käytössä oli Scrumin mukainen tuotteen kehitysjono, jota ei ollut käytössä ensimmäisessä tapauksessa. Molempia tapauksia yhdisti myös aikarajoitettujen iteraatioiden puute prosessissa. Scrumin mukaisia suunnittelutapaamisia, arviointitapaamisia, retrospektiivijä ja päivätapaamisia järjestettiin molemmissa prosesseissa tasaisin väliajoin.

Tapausten välillä oli eroja Scrumin periaatteiden implementoinnissa prosessiin. Ylläpitoprojektissa oli enemmän Scrumin piirteitä kuin

ensimmäisessä tapauksessa. Tuotteen kehitysjonon lisäksi ylläpitoprojektissa oli käytössä Scrumin mukainen tuotteen omistajan sekä scrum-mestarin rooli. Ensimmäisessä tapauksessa rooleja ei määritelty lainkaan.

Scrumbanista saadut kokemukset esitellyissä tapausesimerkeissä olivat pääasiassa positiivisia. Kommunikaatio, yhteistyö ja ryhmähenki kehittyivät molemmissa tapauksissa positiivisesti. Lisäksi useat, erityisesti aikarajoitetuista iteraatiosta johtuvat ongelmat pystyttiin ratkaisemaan Scrumbanin avulla tehokkaasti. Ylläpitoprojektissa Scrumin aikarajoitetuista iteraatioista luopuminen johti kuitenkin joidenkin kehittäjien vastuulla olevien tehtävien hitaampaan valmistumiseen.

Tapausten prosessit olivat arvaamattomia ja vaatimukset muuttuivat nopeasti. Tapausten perusteella tälläisissa prosesseissa Kanbanin tai Scrumin noudattaminen tuottaa ongelmia. Scrumin suurimpana ongelmana on aikarajoitettujen iteraatioiden noudattaminen ja Kanbanin ongelmana rajotteiden puute, joka johtaa ongelmiin kommunikoinnin kanssa. Scrumin palautesyklin ja Kanbanin työnkulun visualisoinnin yhdistämisen avulla pystytään muodostamaan prosessi, joka sopii arvaamattomaan ympäristöön.

5 Yhteenveto

Perinteisten ohjelmistotuotantomenetelmien noudattaminen on osoittautunut useissa projekteissa hankalaksi. Ongelmina perinteisissä lähestymistavoissa on, ettei vaatimuksia täysin ymmärretä projektin alussa sekä se, että vaatimukset muuttuvat jatkuvasti kehitysprosessin edetessä [DS90]. Vaihtoehtona perinteisille ohjelmistotuotantomenetelmille on noussut ketterän ohjelmistokehityksenmentelmiä. Ne painottavat yksilöitä ja yksilöiden vuorovaikutusta, toimivan ohjelmiston merkitystä, asiakkaan merkitystä kehitysprosessin kriittisenä osana ja muutoksiin sopeutuvaa kehitystä [DC04].

Scrum ja Kanban ovat ketterän ohjelmistokehityksen menetelmiä. Niita yhdistää Lean-ajattelutavan mukainen empiirisyyden ja optimisaation periaate, josta käytetään nimitystä Kaizen [HK09]. Kaizen-periaate mahdollistaa Scrumin ja Kanbanin periaatteiden yhdistelemisen projektin tarpeisiin sopivan prosessin muodostamiseksi. Scrumin ja Kanbanin piirteitä yhdistävästä prosessista käytetään nimitystä Scrumban [NKMS12]. Tutkielman tapausesimerkkien perusteella Scrumban soveltuu arvaamattomiin projekteihin, joissa vaatimukset muuttuvat nopeasti. Tälläisia projekteja ovat muun muassa tapauksesimerkeissä esitellyt ylläpitoprojektit.

Vaatimuksien muuttuessa nopeasti Scrumin aikarajoitettujen iteraatioiden noudattaminen tuottaa tapausesimerkkien perusteella hankaluuksia [Ter12, NKM11]. Sprinttien suunnittelun sijaan nopeasti muuttuviin vaatimuksiin pystyy vastaamaan Scrumbanin mukaisella jatkuvalla suunnittelulla [Ter12]. Aikarajoitetuista iteraatioista luopuminen voi kuitenkin tuota myös ongelmia palautesyklin kanssa, jonka puuttuminen voi johtaa kehitysryhmän tehottomuuteen.

Tapausesimerkeissä Kanbanin ongelma on rajotteiden puute, joka saattaa johtaa kommunikaation vähenemiseen kehitysryhmän jäsenten välillä sekä edistymisen tunteen puutteeseen [Ter12]. Kommunikaatiota kehitysryhmän sisällä pystyy lisäämään Scrumin mukaisilla päivätapaamisilla ja retrospektiiveillä. Kehitysryhmän edistymisen tunnetta lisää tasaisin väliajoin asiakkaan kanssa järjestettävät arviointitapaamiset. Kanbanin hyvänä puolena nousee esille työnkulun visualisointi kanban-taulun avulla [Ter12, NKM11].

Jokainen ohjelmistotuotantoprojekti on omalla tavallaan uniikki. Tapausesimerkit osoittivat, että sopivaa prosessia ei välttämättä ole valmiiksi olemassa, vaan se tulee muodostaa kokeilemisen ja kokemuksen kautta.

Scrum ja Kanban eivät ole valmiiksi määriteltyjä prosesseja, vaan työkaluja prosessin kehittämiselle. Scrumban on menetelmä, joka on seurausta Scrumin ja Kanbanin empiirisestä luonteesta. Se on kuitenkin pelkkä työkalu ja kuten kaikki työkalut, se ratkaisee vain tietyn ongelman.

Lähteet

[And10] Anderson, David J.: *Kanban: Successful Evolutionary Change for Technology Organizations.* Blue Hole Press, 2010.

[DC04] David Cohen, Mikael Lindvall, Patricia Costa: *An Introduction to Agile Methods.* Teoksessa *Advances in Computers, Vol. 62*, 2004.

[DS90] DeGrace, P. ja Stahl, L. H.: *Wicked problems, righteous solutions: a catalogue of modern software engineering paradigms.* Yourdon Press, 1990.

[Gol90] Goldratt, Eliyahu M.: *What is this thing called theory of constraints and how should it be implemented?* North River Press, 1990.

[HK09] Henrik Kniberg, Mattias Skarin: *Kanban and Scrum - making the most of both.* Teoksessa *Enterprise Software Development Series*, 2009.

[IPF+11] Ikonen, Marko, Pirinen, Elena, Fagerholm, Fabian, Kettunen, Petri ja Abrahamsson, Pekka: *On the impact of kanban on software project work: An empirical case study investigation.* Teoksessa *Engineering of Complex Computer Systems (ICECCS), 2011 16th IEEE International Conference on*, sivut 305–314. IEEE, 2011.

[JMG03] John M. Gross, Kenneth R. McInnis: *Kanban Made Simple: Demystifying and Applying Toyota's Legendary Manufacturing Process*. AMACOM, 2003.

[JS11] Jeff Sutherland, Ken Schwaber: *The Scrum Papers: Nut, Bolts, and Origins of an Agile Framework*. Scruminc, 2011.

[Kni07] Kniberg, Henrik: *Scrum and XP from the Trenches*. Teoksessa *Enterprise Software Development Series*, 2007.

[Lin12] Linders, Ben: *Evidence of Success of Agile Projects*, 2012. `http://www.infoq.com/news/2012/11/success-agile-projects`.

[man01] manifesto, Agile: *Manifesto for Agile Software Development*, 2001. `http://www.agilemanifesto.org/`.

[Mis14] Miseviciute, Dovile: *Scrumban: on demand vs. long-term planning*, 2014. `www.eylean.com/blog/2014/11/scrumban-on-demand-vs-long-term-planning`.

[NKM11] Nikitina, Natalja ja Kajko-Mattsson, Mira: *Developer-driven big-bang process transition from Scrum to Kanban*. Teoksessa *Proceedings of the 2011 International Conference on Software and Systems Process*, sivut 159–168. ACM, 2011.

[NKMS12] Nikitina, Natalja, Kajko-Mattsson, Mira ja Stråle, Magnus: *From Scrum to Scrumban: A Case Study of a Process Transition*. Teoksessa *Proceedings of the International Conference on Software and System Process*, ICSSP '12, sivut 140–149, Piscataway, NJ, USA, 2012. IEEE Press, ISBN 978-1-4673-2352-9. `http://dl.acm.org/citation.cfm?id=2664360.2664382`.

[RF09] Rubart, Jessica ja Freykamp, Frank: *Supporting daily scrum meetings with change structure.* Teoksessa *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, sivut 57–62. ACM, 2009.

[Ter12] Terlecka, Katarzyna: *Combining Kanban and Scrum – Lessons from a Team of Sysadmins.* 2014 Agile Conference, 0:99–102, 2012.

[TN86] Takeuchi, Hirotaka ja Nonaka, Ikujiro: *The new new product development game.* Harvard business review, 64(1):137–146, 1986.

[Ver13] VersionOne: *8th Annual State of Agile Survey*, 2013. `http://info.versionone.com/state-of-agile-development-survey-ninth.html`.