

Scrum: Ketterien menetelmien markkinajohtaja

Lasse Koskela

Reaktor Innovations Oy

lasse.koskela@ri.fi

Abstrakti

Scrum on yleisimmin käytössä oleva ketterä ohjelmistokehitysmenetelmä, mutta miksi? Omalta osaltaan tilannetta voi selittää prosessimallin kehittäjien aktiivisuudella tai mieleenpainuvalla nimellä, mutta taustalta löytyy myös paljon osuvampia ja järkeenkäyvämpiä syitä. Tässä esityksessä pureudumme Scrum-menetelmän lyhyen esittelyn jälkeen siihen, mikä tekee Scrumista agile-menetelmien oikeutetun markkinajohtajan ja siihen missä markkinan rajat kulkevat?

Scrumin historia

Scrumin juuret juontavat 1990-luvun alkupuoliskolle ja Easel Corporation-nimisen ohjelmistoyrityksen toimitiloihin, joissa ensimmäinen Scrum-projekti syntyi Jeff Sutherlandin, John Scumniotalesin ja Jeff McKennan käsissä vuonna 1993. Easel Corporation ostettiin paria vuotta myöhemmin, mutta Scrum-menetelmä jatkoi kehitystään. Jeff Sutherland ja Ken Schwaber kuvasivat Scrum-menetelmän ensimmäistä kertaa julkisesti vuoden 1996 OOPSLA-konferenssissa.

Menetelmä ei syntynyt tyhjästä vaan ammensi oppejaan Harvard Business Review –artikkelin “The New New Product Development Game” (1986) innovaatioista erityisen tuottavien organisaatioiden tavoista tehdä tuotekehitystä. Itse asiassa termi ”Scrum” mainittiin ensin nimenomaan tuossa artikkelissa—Sutherland ja Schwaber toivat termin ohjelmistokehityksen maailmaan.¹

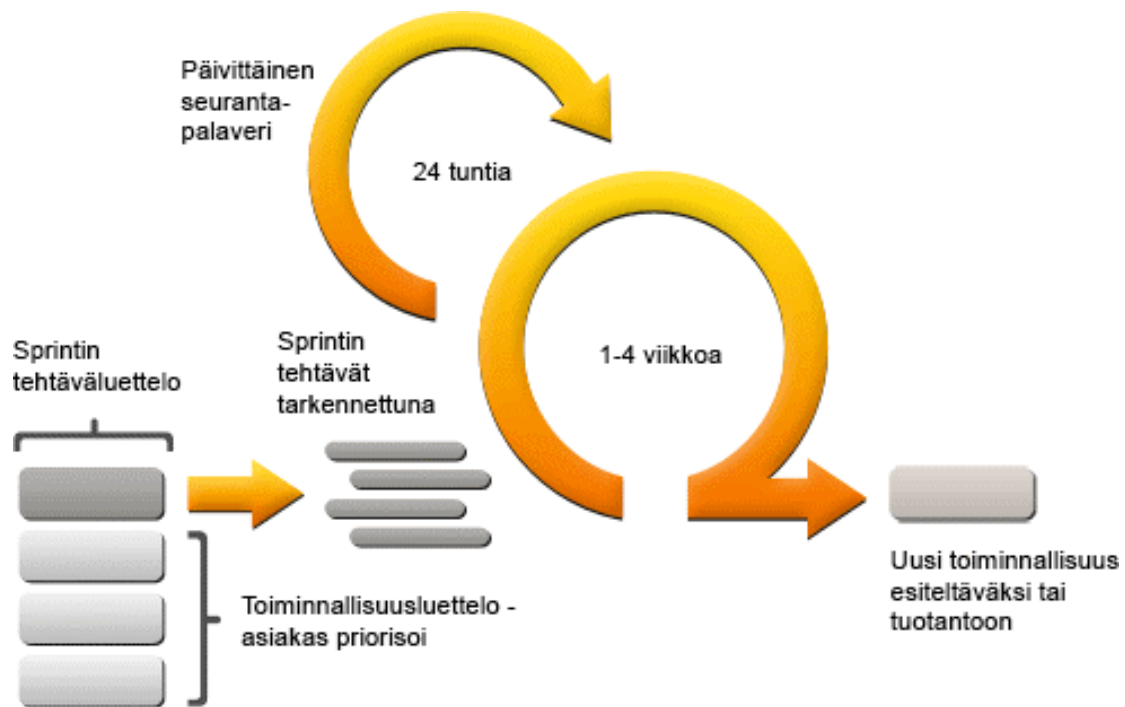
Vuosituhanen vaihtuessa vaihtui myös terminologia. Kent Beck kokosi joukon XP:n kannattajia yhteen huoneeseen Oregonissa vuonna 2000 keskustelemaan eri “kevyiden” menetelmien hyveistä, mikä johti vuotta myöhemmin termin “agile method” sekä itse Agile Manifeston syntymiseen useiden eri menetelmien edustajien yhteisessä tapaamisessa Utahin Snowbirdissä, 2001.² Snowbirdin tapaamiseen osallistuivat myös Jeff Sutherland sekä Ken Schwaber ja Scrumista tuli samalla yksi useista kattotermin “agile method” alle luetuista menetelmistä.

Scrum-prosessi

Scrum on prosessin perspektiivistä erittäin yksinkertainen ja kuvataan usein seuraavankaltaisella kaaviolla.

¹ <http://www.infoq.com/presentations/The-Roots-of-Scrum>

² <http://agilemanifesto.org/>



Kuva 1 – Prosessina Scrum on erittäin yksinkertainen ja voidaan helposti kuvata yhdellä kaaviolla.

Lukien kuvaa 1 vasemmalta oikealle, voimme tunnistaa prosessin tärkeimmät artefaktit sekä syklisen luonteen. Kaikki lähtee liikkeelle asiakkaan priorisoimasta listasta tuotteen halutuista ominaisuuksista eli toiminnallisuusluettelosta (Product Backlog). Jokaisen iteraation (Sprint) alussa tiimi ja asiakas yhdessä valitsevat alkavan Sprintin sisällön, muodostaen Sprintin tehtäväluettelon (Sprint Backlog).

Sprintin aikana, joka yleensä kestää 1-4 viikkoa, tiimi työskentelee parhaimmaksi katsomallaan tavalla toteuttaakseen kaikki Sprintin tehtäväluetteloon valitut ominaisuudet. Sprintin sisällä tiimi synkronoi omaa toimintaansa päivittäisen seurantalaverin avulla. Kunkin Sprintin lopputuloksena tuotteesta syntyy uusi versio, joka voidaan haluttaessa viedä tuotantoon tai esimerkiksi sisäiselle asiakkaalle koekäyttöön.

Tämä sykli jatkuu Sprint kerrallaan niin kauan kuin tuotteen omistaja (Product Owner) näkee hyödylliseksi investoida jatkokehitykseen. Kehityksen jatkuessa tuotteen omistaja työstää jatkuvasti tuotteen toiminnallisuusluetteloa yhdessä tiimin jäsenten kanssa siten, että listan prioriteettijärjestys edustaa kullakin ajan hetkellä parasta mahdollista tietoa.

Tässä tulikin jo nopeasti mainittua monia Scrum-menetelmän ydinosa, mutta katsotaan vielä hieman lähemmin tuotteen ominaisuuslistaa sekä Sprintin sisällä käytettävää tehtävlistaa ja tutustutaan sen jälkeen Scrum-menetelmän määrittelemiin rooleihin.

Scrum-projektin raamattu: Product Backlog

Scrum-menetelmän ydin on dokumentti nimeltä Product Backlog, jota kutsumme paremman käännökseen puutteessa tuotteen ominaisuuslistaksi. Tämä dokumentti on kaikessa yksinkertaisuudessaan lista, joka sisältää tuotteen omistajan kannalta arvokkaita asioita—yleensä rakennettavan tuotteen ominaisuuksia tai piirteitä—prioriteettijärjestyksessä. Ominaisuuksien keskinäinen prioriteetti syntyy tuotteenomistajan päätöksenä joka perustuu ominaisuuden tuomaan lisäarvoon,

kustannusarvioon, sekä mahdollisiin riskitekijöihin tai muuhun epävarmuuteen. Kuva 2 on anonymisoitu esimerkki Microsoft Excel –työkalulla tehdystä tuotteen ominaisuuslistasta.

Product Backlog					0	1	2	3	4	5	6	7	8
Backlog item	Wiki link	Origin	Value	Requirement Area	654	614	614	614	614	614	614	614	614
Item #1	http://wiki/item	Customer 1	1000	Ads	8	0	0	0	0	0	0	0	0
Item #2	http://wiki/item	Customer 2	500	Sales	8	0	0	0	0	0	0	0	0
Item #3	http://wiki/item	Improve	10		4	0	0	0	0	0	0	0	0
Sprint #1					20								
Item #4	http://wiki/item	Customer 1	2000	Ads	4	4	4	4	4	4	4	4	4
Item #5	http://wiki/item	Customer 5	100	Admin	2	2	2	2	2	2	2	2	2
Item #6	http://wiki/item	Customer 3	300	Admin	3	3	3	3	3	3	3	3	3
Item #7	http://wiki/item	Customer 3	200	Ads	13	13	13	13	13	13	13	13	13
Item #8	http://wiki/item	Customer 2	500	News	13	13	13	13	13	13	13	13	13
Item #9	http://wiki/item	Customer 1	200	Sales	21	21	21	21	21	21	21	21	21
Item #10	http://wiki/item	Customer 1	100	News	8	8	8	8	8	8	8	8	8
Item #11	http://wiki/item	Customer 1	400	Ads	50	50	50	50	50	50	50	50	50
Item #12	http://wiki/item	Customer 1	200	News	60	60	60	60	60	60	60	60	60
Item #13	http://wiki/item	Customer 3	400	Admin	50	50	50	50	50	50	50	50	50
Item #14	http://wiki/item	Customer 7	200	Sales	60	60	60	60	60	60	60	60	60
.
Item #30	http://wiki/item	Improve	400	News	50	50	50	50	50	50	50	50	50
Item #31	http://wiki/item	Customer 2	200	Admin	60	60	60	60	60	60	60	60	60
Release #3													
Item #32	http://wiki/item	Customer 1	400	Ads	50	50	50	50	50	50	50	50	50
Item #33	http://wiki/item	Customer 4	200	Admin	60	60	60	60	60	60	60	60	60
Item #34	http://wiki/item	Customer 1	400	News	50	50	50	50	50	50	50	50	50
Item #35	http://wiki/item	Customer 1	200	Sales	60	60	60	60	60	60	60	60	60

Kuva 2 - Esimerkki tuotteen ominaisuuslistasta toteutettuna Microsoft Excel -työkalulla.

Tuotteen ominaisuuslista luodaan projektin alussa sen hetkisen tiedon pohjalta, mutta elää läpi tuotteen koko elinkaaren. Projektin edetessä tuotteen omistaja lisää uusia ominaisuuksia listaan ja järjestää niitä prioriteetin mukaan, poistaa ominaisuuksia jotka ovat käyneet tarpeettomiksi, ja yleisesti ottaen pitää listan ajan tasalla.

Kaikki ominaisuuslistan asiat eivät siis ole yhtä tärkeitä vaan listan kärki on korkeammalla prioriteetissa kuin peräpää ja tiimien kapasiteettia käytetään nimenomaan listan ensimmäisten ominaisuuksien toteuttamiseen—listan kärki siis poimitaan aina Sprintin suunnittelupalaverissa tarkemman tutkiskelun ja sitä myöten toteutuksen alle.

Sprintin suunnittelupalaverin sujuvuus kuitenkin edellyttää, että ominaisuudet on pureskeltu tarpeeksi pieniin palasiin—palasiin joille tiimi pystyy antamaan tarkat työmääräarviot ja jotka ovat sellaisenaan edelleen arvokkaita tuotteenomistajalle. Koko ominaisuuslistaa ei tällaiseen hienojakoisuuteen asti kuitenkaan ole järkevää jalostaa vaan tätä työtä tehdään läpi projektin rinnakkain varsinaisen kehitystyön ohella, muokaten listaa jatkuvasti siten, että ominaisuuslistan kärki on aina pienempää ja paremmin ymmärrettyä tavaraa kuin listan häntä.

Scrum-tiimin päiväkirja: Sprint Backlog

Siinä missä tuotteen ominaisuuslistan eli Product Backlogin voisi mieltää Scrum-projektin pyhäksi kirjaksi ja tuotteen omistajan omaisuudeksi, Sprint backlog puolestaan on puhtaasti Scrum-tiimin oma työkalu. Sprint Backlog, jota joskus kutsutaan myös nimellä Sprintin tehtävälista, on jatke tuotteen ominaisuuslistalta poimituille ominaisuuksille.

Siinä missä tuotteen ominaisuuslistalla kaikki merkitsee jotain tuotteen omistajalle, Sprintin tehtävälistalle päätyvät tehtävät ovat usein puhtaasti teknisiä—asioita joita pitää tehdä jotta haluttu ominaisuus saadaan lisättyä tuotteeseen. Kuva 3 esittelee erään projektin Sprintin tehtäväälistaa, jälleen Microsoft Excel –työkalulla toteutettuna.

		Days Left in Sprint				
		15	13	10	8	
		F				
Who	Description	7/22/2002	7/24/2002	7/26/2002	7/31/2002	
Total Estimated Hours:		554	458	362	270	0
-	User's Guide	-	-	-	-	-
SM	Start on Study Variable chapter first draft	16	16	16	16	
SM	Import chapter first draft	40	24	6	6	
SM	Export chapter first draft	24	24	24	6	
Misc. Small Bugs						
JM	Fix connection leak	40				
JM	Delete queries	8	8			
JM	Delete analysis	8	8			
TG	Fix tear-off messaging bug	8	8			
JM	View pedigree for kindred column in a result set	2	2	2	2	
AM	Derived kindred validation	8				
Environment						
TG	Install CVS	16	16			
TBD	Move code into CVS	40	40	40	40	
TBD	Move to JDK 1.4	8	8	8	8	
Database						
KH	Killing Oracle sessions	8	8	8	8	
KH	Finish 2.206 database patch	8	2			
KH	Make a 2.207 database patch	8	8	8	8	
KH	Figure out why 461 indexes are created	4				

Kuva 3 - Esimerkki Sprintin tehtäväälistasta toteutettuna Microsoft Excel -työkalulla.

Kuvan 3 esimerkki ei ole missään nimessä ideaalinen esimerkki Sprintin tehtäväälistasta. Kyseistä tehtäväälistaa ei nimittäin millään lailla ole sidottu takaisin tuotteen ominaisuuslistaan, mikä näkyy siinä ettei tehtäväälistan perusteella pysty sanomaan esimerkiksi sitä, kuinka monta ominaisuutta on jo toteutettu ja kuinka monta on vielä työn alla.

Tämä ei kuitenkaan ole sinänsä kriittinen puute, että Scrumin yksi perimmäisistä tukijaloista on itseohjautuva tiimi ja Sprintin tehtäväälista on nimenomaan tiimin työväline, jota he käyttävät päästäkseen Sprintin suunnittelupalaverissa sitoutumaansa tavoitteeseen. Se, että kuvan 3 tiimillä ei juurikaan ole näkyvyyttä heidän kokonaisvaltaiseen edistykseensä Sprintin sisällä, on asia jonka he oppivat Scrum-prosessiin sisäänrakennetun palautesyklin kautta—ja muuttavat toimintatapaansa Sprintin lopussa järjestettävän retrospektiivin myötä palaverissa, jossa tiimi erityisesti tarkastelee toimintaansa ja pyrkii kehittämään sitä.

Se, että Scrum jättää käytännössä kaikki työkalut ja tekniikat ”auki” juontaa juurensa juuri siihen, että paras ratkaisu tulee usein tekijöiltä itseltään—ei keskitetyltä prosessiorganisaatiolta—ja luottamukseen siitä, että yksilöt tekevät työtä parhaan tietämyksensä ja taitonsa mukaan.

Yksilöistä puheen ollen, niistä harvoista asioista joita Scrum määrittelee yksi on ns. Scrum-roolit, joita löytyy kolme kappaletta.

Scrum-menetelmän kolme roolia

Siinä missä Sun Microsystems aikoinaan J2EE:n myötä visioi organisaation, jossa sovelluksen kehitystä ja asennusta varten oli määritelty lukuisia rooleja aina useasta erityyppisestä kehittäjästä monenlaiseen asentajaan ja konfiguraattoriin, Scrum määrittelee tasan kolme roolia: tuotteen omistajan, itse tiimin, ja eriskummallisen roolin nimeltä Scrum Master.

Tuotteen omistaja (Product Owner)

Tuotteen omistaja on loppukäyttäjän ja maksavan asiakkaan edustaja. Hänen vastuullaan on, että tiimi työstää liiketoiminnan kannalta oikeita asioita. Hän myös päättää koska ja mitä ominaisuuksia tuotteesta julkaistaan. Yksin hänen ei kuitenkaan tarvitse kaikkea tehdä, koska hänellä on Scrum-tiimi apunaan.

Tiimi

Scrum-tiimi on projektin lisäarvoa tuottava elin, joka muuntaa tuotteen ominaisuuslistaa pala palalta toimivaksi tuotteeksi. Scrum-tiimi on moniosaava (engl. cross-functional) eli koostuu eri funktioiden osaamisesta, joita voivat olla esimerkiksi ohjelmointi, systeemitestaus, käyttöliittymäsuunnittelu ja tekninen kirjoittaminen.

Kuten jo aiemmin viittasimme, Scrum-tiimi on itseohjautuva. Tämä tarkoittaa sitä, että tiimi itse päättää kuinka Sprintin tehtävälistan tehtävät organisoidaan ja jaetaan. Tiimin sisäisiä virallisia rooleja ei ole vaan kuka tahansa voi poimia tehtävälistasta minkä tahansa tehtävän itselleen. Käytännössä yksilöt usein seuraavat omaa erityisosaamistaan, mutta itseohjautuvuus mahdollistaa myös yksilöllisen osaamisalueen laajentamisen ja edesauttaa tiimin yhteistoimintaa pitämällä tavoitteen ja vastuun sen saavuttamisesta yhteisenä.

Scrum Master

Scrum Masterin viisi tärkeintä vastuuta usein listataan seuraavasti:

- Esteiden poistaminen IT:n ja asiakkaan välillä siten, että asiakas ohjaa kehitystä suoraan
- Asiakkaan opettaminen Scrum käytössä tuottojen maksimointiin ja tavoitteiden saavuttamiseen
- Kehitystiimin elinolojen parantaminen tukemalla ja kannustamalla luovuutta ja itseohjautuvuutta.
- Kehitystiimin tuottavuuden kasvattaminen
- Menetelmien ja käytäntöjen parantaminen siten, että jokaisen Sprintin lopputulos on teknisesti tuotantokelpoinen.

Toisin sanoen, Scrum Master on roolina yhdistelmä valmentajaa, kivimuuria ja katujuuraa, jonka tehtävänä on luoda ja ylläpitää ympäristö ja kulttuuri, joka mahdollistaa Scrum-tiimin ja tuotteen omistajan sujuvan yhteistyön.

Pintapuolisesti Scrum Masterin rooli vaikuttaa usein siltä, että hän ainoastaan organisoi Sprintin suunnittelupalaverin, Sprintin päätöspalaverin sekä päivittäisen Scrum-palaverin, mutta pinnan alla rooli sisältää paljon enemmän ja hyvä Scrum Master tarvitseekin paksun nahan, ihmistuntemusta, ja poliittista pelisilmää.

Yllä mainittu ”kivimuurin” rooli viittaa siihen, että Scrum Master joutuu usein suojaamaan tiimiä ulkoisilta häiriöiltä ja minimoimaan näiden vaikutuksen tiimin

tuottavuuteen. Katujyränä Scrum Master puolestaan joutuu toimimaan tilanteissa, missä ympäröivä organisaatio tai sen byrokratia estää tiimiä etenemästä täyden potentiaalinsa mukaisesti.

Siinä missä tuotteen omistajan rooli on usein erittäin työläs, Scrum Masterin rooli on usein erittäin epäkiitollinen.

Eikä siinä kaikki

Kuten juuri totesimme, Scrum määrittelee kolme roolia eikä yhtään enempää. Nämä roolit voivat kuitenkin joskus jakaantua useammalle henkilölle ja niiden lisäksi saatetaan tarvita ylimääräisiä rooleja—ei Scrumin toimivuuden vaan Scrumin ja ympäröivän organisaation sovittamiseksi toisiinsa.

Yksi esimerkki roolien jakaantumisesta ovat esimerkiksi tuoteomistajatiimi, joka muodostuu useammasta tuotehallinnon ammattilaisesta ja/tai mahdollisesti toiminnallisista suunnittelijoista, interaktiosuunnittelijoista, jne. Tällainen tuoteomistajatiimi hoitaa aiemmin kuvatun Product Owner –tontin yhdessä, muodostaen yhden mielipiteen ja tahtotilan. Käytännössä tuoteomistajatiimissä usein tarvitaan yksi puheenjohtaja tai ”päätuoteomistaja”, jonka ääni ratkaisee—etenkin jos tuoteomistajilla on tuotteesta kullakin omat reviirinsä ja heidän kannustimensa eivät ole kiinni koko tuotteen menestyksestä ja kehityksestä.

Vastaavasti Scrum Masterin rooli saatetaan joskus jakaa tiimin jäsenten kesken, usein kiertävänä ”hattuna”, jota kukin vuorollaan kantaa. Näin voidaan toimia esimerkiksi tilanteessa, jossa kukaan tiimin jäsenistä ei ole erityisen innokas toimimaan tässä uudessa roolissa tai jos tiimi näkee hyödylliseksi hajauttaa osaamistaan myös tältä osin.

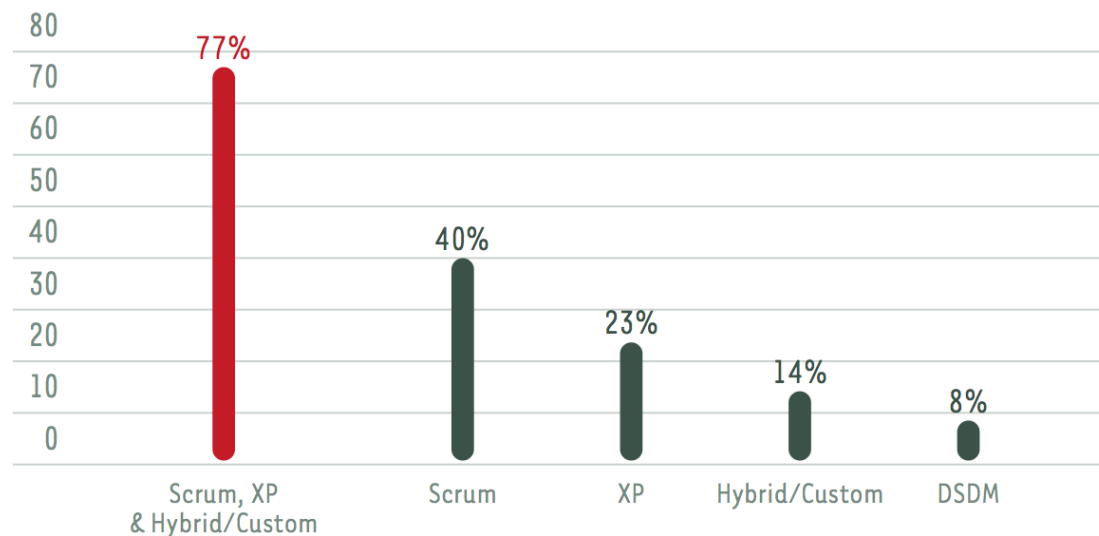
Scrum määrittelemien roolien lisäksi saatetaan siis joskus tarvita muita ”virallisia” rooleja sovittamaan Scrum-projekti ympäröivään organisaatioon. Tällainen tarve saattaa esiintyä esimerkiksi tilanteessa, jossa asiakasorganisaation virallinen toimintatapa edellyttää jokaisella kehitysprojektilla olevan nimetty tekninen arkkitehti tai projektipäällikkö. Useimmiten nämä roolit täytetään kehitystiimin jäsenien toimesta—vaikka roolin edellyttämät ylimääräinen työ tehtäisiinkin koko tiimin voimin.

Olemme nyt raapaisseet lähes joka puolta Scrum-menetelmästä, mutta abstraktissa lupailtiin myös näkemyksiä Scrumin markkinaosuuden suhteen, joten nostetaan perspektiiviä ja selvitetään seuraavaksi hieman eri Agile-menetelmien markkinaosuuksia.

Agile-menetelmien markkinaosuudet

Scrum on vain yksi monista menetelmistä, jotka luetaan käsitteen ”agile method” alle. Muita yleisiä agile-menetelmiä ovat mm. Extreme Programming (XP), DSDM-konsortion Dynamic Systems Development Method (DSDM), Jeff De Lucan Feature-Driven Development (FDD), Tom Gilbin Evolutionary Project Management (Evo), sekä Alistair Cockburnin Crystal-menetelmät. Näillä menetelmillä on omat yksilölliset piirteensä vaikka toteuttavatkin samoja Agile Manifeston arvoja. Vastaavasti eri agile-menetelmien suosio ja levinneisyys vaihtelee merkittävästi.

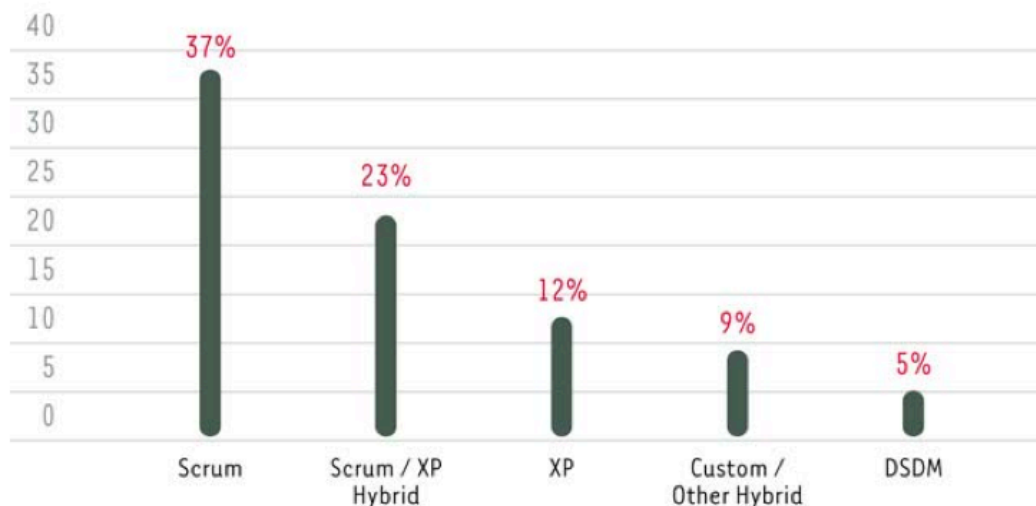
Agile-menetelmien projektinhallintatyökalua kehittävä yritys, VersionOne, teetti vuonna 2006 kyselyn johon vastasi 722 yksilöä 47:stä eri maasta.³ Kysely, jonka otsikko oli "State of Agile Development", sisälsi muunmuassa kysymyksen vastaajan organisaation käyttämästä menetelmästä (ks. kuva 4). Sama kysymys sisältyi yrityksen vuonna 2007 teettämään kyselyyn, johon vastauksia tuli 1700 kappaletta yli 70 maasta, ja sama trendi näkyi vastauksista edelleen (ks. kuva 5).⁴



Which Agile Methodology Do You Follow Most Closely?

Kuva 4 - VersionOne, Inc. teettämä "State of Agile Development" -tutkimus vuodelta 2006 antaa kuvan suosituimpien ketterien menetelmien markkinaosuuksista.

Which Agile methodology do you follow most closely?



Kuva 5 - VersionOne, Inc. teettämä "State of Agile Development II" -tutkimus vuodelta 2007 vahvistaa edellisvuoden kyselyn tuloksia Scrum- ja XP-menetelmien markkinaajoitajuudesta.

Kuten kuvan 4 tuloksista voidaan päätellä, kyselyn vastaajien keskuudessa Scrum ja XP nousivat esille selvästi suosituimpina menetelminä, Scrumin ottaessa selkeän voiton toiseksi jääneestä XP:stä. Vuoden 2007 kyselyn tuloksissa eriteltiin Scrum/XP-

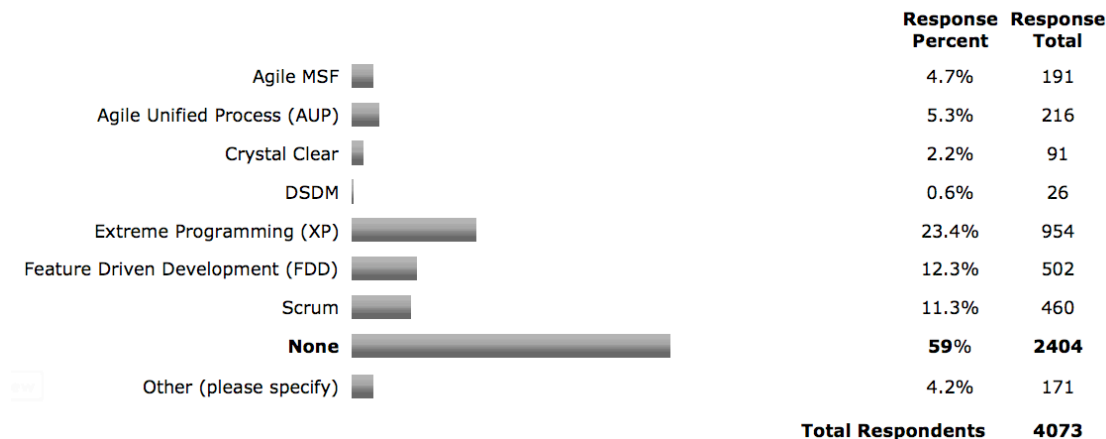
³ <http://www.versionone.com/pdf/StateofAgileDevelopmentSurvey.pdf>

⁴ http://www.versionone.com/pdf/StateOfAgileDevelopmet2_FullDataReport.pdf

hybridit muista hybridi-menetelmistä, mikä edelleen korosti Scrumin markkinajohtajuutta—noin 60% vastaajista ilmoitti käyttävänsä Scrumia joko sellaisenaan tai yhdistäen siihen XP:n käytäntöjä.

Mutta kuinka luotettavia VersionOnen kyselyt ovat? Jos vilkaisemme Scott Amblerin Dr. Dobbs –lehdessä 2006 julkaisemaa kyselyä (ks. kuva 6), tulokset näyttävät jokseenkin erilaisilta.⁵

Which agile methodologies has your organization adopted? (check all that apply)



Kuva 6 - Scott Amblerin Dr. Dobbs -lehdessä julkaisema kysely nostaa XP:n ja FDD:n asemia markkinajohtajuuskilvassa.

Kuvan 6 mukaan vaikuttaisi siis siltä, että XP on kaikkein laajimmin käytössä oleva agile-menetelmä ja että Scrum sijoittuisi vasta kolmanneksi FDD:n ottaessa toisen sijan.

Melkoinen ero, mutta mistä se johtuu? Yksi tekijä on varmasti näiden kyselyiden toisistaan jossain määrin eriävä kohdeyleisö—siinä missä VersionOnen kyselyä markkinoitiin ensisijaisesti sähköpostilistoilla ja VersionOne-työkalun käyttäjien keskuudessa, Scott Amblerin kysely levisi ensisijaisesti Dr. Dobbs –lehden lukijoiden keskuudessa.

Kyselyt eivät siis tunnu kertovan kovin vakuuttavasti eri menetelmien käyttöasteesta, paitsi että suosituin menetelmä vaikuttaisi löytyvän kolmikosta Scrum, XP ja FDD. Pystyisimmekö kenties päättämään jotain eri menetelmien online-yhteisöjen koosta? Sekä Scrum- että XP-yhteisöiden ensisijaiset viestintäfoorumit pyörivät Yahoo!-n sähköpostilistojen muodossa. Näihin rekisteröityneitä käyttäjiä löytyy Scrumin osalta 4,741 ja XP:n osalta 8,568,⁶⁷ mikä tuntuisi vahvistavan XP:n asemaa.

FDD:ltä sen sijaan ei vastaavaa tietoa löydy. FDD:n pääasiallinen web-sivusto, featuredrivendevelopment.com, sisältää keskustelufoorumin jonne on vuosien saatossa kerääntynyt vain noin 1,200 viestiä. Se tuntuu kovin pieneltä verrattuna Scrum- ja XP-sähköpostilistojen kymmeneen tuhansiin viesteihin. XP-listalla niitä on kertynyt jo yli 130,000. Tämä yhdistettynä siihen, että kirjoittaja ei henkilökohtaisesti ole törmännyt yhteenkään FDD-projektiin eikä yhteenkään FDD-käyttäjään, viittaa

⁵ <http://www.ambysoft.com/downloads/surveys/AgileAdoptionRatesSummary.pdf>

⁶ <http://groups.yahoo.com/group/scrumdevelopment/>

⁷ <http://groups.yahoo.com/group/extremeprogramming/>

vahvasti siihen, että Amblerin kyselyn tulokset eivät täsmää omassa ympäristössäni vallitsevaan todellisuuteen.

Ja ehkä siinä onkin kaiken avain—ympäristössä. Kenties meidän pitäis katsoa lähemmäksi kotipesää selvittäessämme agile-menetelmien markkinajohtajaa? Jos finanssialalla IBM on kuningas niin miksei sama ilmiö voisi toteutua myös ohjelmistokehitysmenetelmien osalta? Kun yritän oikein kovasti miettiä, ei mieleen tulti kuin pari hassua projektia jossa käytössä olisi enemmän XP:tä kuin Scrumia muistuttava ketterä menetelmä.

Ehkä Suomi todella on Scrumin kotipiha. Tätä tukee myös se seikka, että yli 600 maailman noin 12,000 sertifioidusta Scrum-käyttäjistä on piskuisesta Suomesta.

Miksi Scrum “myy” paremmin kuin muut?

Scrum on menestynyt hyvin ketterien menetelmien markkinoilla ja etenkin näin koti-Suomessa—siitä ei ole epäilystä. Mielenkiintoisempi kysymys onkin, miksi näin on käynyt? Mikä tekee Scrumista myyvän menetelmän? Vastauksia saattaa olla useita ja esittelen seuraavaksi oman näkemykseni mukaan oleelliset viisi syytä. Mielivaltaisessa järjestyksessä.

Nokia

Näin Suomessa oltaessa, Nokian merkitystä Scrumin levinneisyyden kannalta ei voi myöskään vähätellä. Mitä Nokia edellä, sitä toimittajat perästä. Sama pätee vastaavasti muihin isoihin yrityksiin, jotka työllistävät merkittäviä osia paikallisista IT-alan toimijoista muualla maailmassa.

Sertifiointimalli

Kun alan konferensseissa herää kahvitauolla keskustelu siitä, miksi Scrum on levinnyt niin voimakkaasti viimeisen parin vuoden aikana, ensimmäinen tarjottu tekijä on yleensä Scrum Alliancen sertifiointimalli. Se, että Scrum-menetelmän kylkeen on tarjolla virallisen tahon kehittämä sertifiointikurssi, josta saa virallisen paperin ja nimen julkiselle web-sivulle, on yllättävän merkittävä tekijä isojen IT-talojen koulutus- ja osto-organisaatioiden silmissä.

Meillä Suomessakin on näkyvissä se, että ostajat kysyvät potentiaaliselta toimittajalta, kuinka monta ”CSM:ää” (Certified Scrum Master) heillä on talossa.⁸ Vaikka kyseinen sertifiointikurssi sertifioi vain sen, että henkilö on viettänyt kaksi päivää luokkahuoneessa—ei välttämättä osallistuen aktiivisesti eikä välttämättä sisäistäen mitään kurssin sisällöstä—nähdään se kuitenkin erottavana tekijänä sertifioidun ja ei-sertifioidun konsultin välillä.

Ja hyvästä syystä. Kurssille osallistuminen kuitenkin on tietynlainen indikaattori, vaikkei siihen täysin kannatakaan luottaa osaamisen takuuna. Jos on aikaa haastatella vain yhtä hakijaa Scrum Masterin tehtäviin, kannattaa kymmenestä kandidaatista todennäköisesti kutsua haastatteluun se, joka kurssin on käynyt—jos CV:t muuten ovat tasavertaiset.

Yksinkertaisuus

Vaikka Scrumin sertifiointimallin merkitystä ei voi kieltää Scrumin levinneisyyden edistäjänä, kenties paljon oleellisempi ominaisuus on itse prosessin yksinkertaisuus. Scrumin äärimmäisen yksinkertainen olemus—priorisoitu ominaisuuslista, kolme

⁸ <http://www.scrumalliance.org/training/>

roolia, ja lyhyet iteraatiot—on helppo omaksua. Tuntematon pelottaa, mutta Scrum on helppo ymmärtää ja sitä myöten helpompi hyväksyä.

Matala kynnys

Yksinkertaisuuden myötä myös Scrumin käyttöönoton suhteen kynnys on matalampi kuin monilla muilla ketterillä menetelmillä. Toisin kuin XP, Scrum ei edellytä tiettyjä insinöörikäytäntöjä. Toisin kuin FDD, Scrum ei edellytä tiettyjä suunnittelutekniikoita tai työkaluja. Riittää, että projektia viedään eteenpäin kiinteämittaisissa iteraatioissa, Scrumin kolme roolia on täytetty, ja kehitystä ajetaan priorisoidulla ominaisuuslistalla.

Tämän kolikon kääntöpuoli on ikävä kyllä olemassa—Scrum on helppo ottaa käyttöön muuttamatta entistä toimintamallia juurikaan. Toisin sanoen, Scrumin vaatimat käytännöt on myös helppo maskeerata vanhan prosessin päälle, mikä johtaa pahimmassa tapauksessa Scrumin hylkimiseen epätoimivaksi todettuna—ilman, että organisaatio on kertaakaan nähnyt Scrumia sovellettavan ”oikein.”

Skaalautuvuus

Tärkeimpään kilpakumppaniinsa XP:hen verrattuna Scrumilla on myös myyntivalttinaan selkeä, yksinkertainen ja dokumentoitu skaalautumismalli koko organisaatiolle—ei pelkästään yhdelle projektille tai yhdelle projektitiimille.

Siinä missä XP-kirjallisuus ei juurikaan käsittele skaalautumista tai hajautusta, Scrum-kirjallisuudessa on auttavasti esitelty tapoja soveltaa Scrum-menetelmää useamman tiimin ja hajautetun kehityksen tilanteisiin. Vaikka nämä ratkaisut toimivat yhtä lailla monien muiden menetelmien osana, niiden puute ”virallisissa” lähteissä tuntuu johtavan siihen, että menetelmä X ”ei tue hajautettua kehitystä” tai ”ei sovellu useamman tiimin käyttöön.”

Ja nämä väitteet ovat perättömiä. Agile-menetelmät nimittäin soveltuvat yhtä lailla pieniin ja paikallisiin kuin isoihin ja hajautettuihin projekteihin. Mielenkiintoisempi kysymys kuuluukin, onko tälle soveltuvuudelle olemassa rajoja, ja jos on, niin missä ne kulkevat?

Missä kulkevat markkinan rajat?

Kysyttäessä, mihin Agile-menetelmät—tai erityisesti Scrum—soveltuvat, voimme vastata, ”kaikkein.” Tämä ei kuitenkaan ole kovin hyödyllinen vastaus, sillä eiväthän nämä menetelmät voi olla yhtä sopivia kaikenlaisiin projekteihin ja ympäristöihin.

Muotoilkaamme siis kysymys uudelleen. Kysytään mieluummin, miten eri tekijät vaikuttavat Scrumin käyttöön ja vastataan siihen osissa. Yksinkertaisuuden nimissä yleistän alla Scrumin vertailukohdan eli erilaiset variaatiot prediktiivisistä, vaiheistetuista, suunnitelmavetoisista jne. prosesseista ”vesiputousmalliksi.”

Todellisuudessa kukaan ei käytä vesiputousmallia—tai ei ainakaan myönnä käyttävänsä—mutta kontrasti agile- ja vesiputousmallien välillä on tehokas keino korostaa erojen merkitystä.

Teknologia ja domain

Yksi yleisimmistä vastalauseista moneen muutosehdotukseen on, ”ei toimi meillä, koska meidän teknologiamme/domainimme on paljon haastavampi ja monimutkaisempi.” On aivan totta, että vanha ja kankea teknologia usein hidastaa ohjelmistokehitystä merkittävästi—ja tämä pätee menetelmästä riippumatta.

Vesiputousmallin kaltaisessa prosessissa ”haastavan” teknologian seuraukset saattavat näkyä huonommin, mutta ne ovat siellä ja mitä pidempään nämä seuraukset kasautuvat pimennossa, sitä kaameampi todellisuus usein on kun se vihdoinkin näyttää itsensä.

Projektin koko

Toistaiseksi suurin Scrum-projekti on ollut yli tuhannen hengen mammutti. Suomessa suurin (tietääkseni) on noin 300 henkeä. Pienimmät luonnollisesti ovat 1-2 hengen luokkaa ja valtaosa projekteista sijoittuu jonnekin parin hengen ja muutaman kymmenen väliin.

Mahdollista Scrumin käyttö siis on lähes minkä tahansa kokoisessa projektissa, vaikka prosessin sujuvuus ja toteutus luonnollisesti vaihtelee. Oleellista on kuitenkin, että samat ongelmat, jotka löytyvät Scrumia skaalatessa massiivisiin projekteihin, ovat olemassa esimerkiksi vesiputousmallia käytettäessä—näiden ongelmien seuraukset vain ovat vaikeammin yhdistettävissä niiden alkuperään.

Valaistaan tätä pienellä esimerkillä. Vesiputousmallin mukaan vedettävän projektin kymmenen eri tiimiä on helppo allokoida toteuttamaan kukin omaa komponenttiaan alussa tarkasti määritellyjä rajapintoja vasten. Scrumia käytettäessä joudutaan tiimien tekemistä synkronoimaan jatkuvasti, mikä saatetaan tulkita indikaationa siitä, ettei Scrum sovellu näin isoihin projekteihin. Alussa hyvältä näyttävän vesiputousmallin illuusio kuitenkin hajoaa siinä vaiheessa kun havaitaan muutostarpeita ja tuotetta halutaankin muokata eri suuntaan kuin alunperin oli suunniteltu.

Vesiputousmalli siis käytännössä johtaa myös tiimien välisen intensiivisen synkronoinnin tarpeeseen, mutta kontekstissa, jossa tiimien rakenne ja osaaminen sekä järjestelmän tila on epäoptimaalinen tähän tarkoitukseen—kukin puhuu omaa kieltään eikä tiedä järjestelmän muista osista ja tämän lisäksi eri komponenttien yhteensopivuus ei ole samaa luokkaa kuin parin viikoin välein koko tuotteen integroivassa Scrum-projektissa.

Hajautettu kehitys

Hajautetun kehityksen ensimmäinen sääntö on, ”älä hajauta.” No, tästä riippumatta tilanne usein on se, että yhden tuotteen ohjelmistokehitystä tehdään useassa eri paikassa ja mahdollisesti myös usealla eri aikavyöhykkeellä.

Scrumin implementointi hajautetussa ympäristössä on vaikeaa. Tiimien ollessa hajallaan—ja etenkin kun tuotteen omistaja ei ole jatkuvasti tiimien lähellä—on ilmiselvää, että kommunikaatio ja yhteistyö kärsivät. Tästä saattaa seurata esimerkiksi se, että tiimien on vaikeaa toteuttaa ominaisuuksia sellaisina kuin tuotteen omistaja ne haluaa koska etäisyys osapuolten välillä aktiivisesti estää tarpeellisen interaktion tapahtumista.

Tosiasia on kuitenkin se, että hajautettu kehitys tuo samat puutteet myös vesiputousprojekteihin. Suurin ero on jälleen se, että tiimit etenevät ”väärään suuntaan” pitempään ja jyrkemmin kuin jos koko tuotekehitysorganisaatio istuisi samassa rakennuksessa, samassa siivessä, samassa kerroksessa ja samassa avotilassa.

Ulkoistettu kehitys

Scrum on syntynyt tuotekehityskontekstissa ja sopiikin siihen kaikkein parhaiten, mutta Scrumin iteratiivisen ja inkrementaalisen luonteen tuomat kilpailuedut realisoituvat myös projektitoiminnassa. Vastaavasti se, ovatko kehittäjät yrityksen

omaa henkilökuntaa vai konsultteja, ei toimi erottavana tekijänä Scrumin ja vesiputousmallien välillä. Se, mikä ulkoistetun kehityksen osalta merkitsee jotain, on sopimusmenettely.

Vesiputousmallien pohjalta on näennäisen helppo tehdä toimitussopimuksia, koska projektin lopputuotos on määritelty tarkasti ja hintakin on usein kiinteä. Tämähän ei ota huomioon sitä, että määrittelyyn käytetystä työstä saatava hyöty tippuu erittäin progressiivisesti, mutta kun tätä piilotettua kustannusta on erittäin vaikea mitata, on täysin ymmärrettävää kuinka tästä menettelystä on tullut niin suosittu.

Scrum-projektien osalta toimitussopimukset ovat kinkkisempi kysymys. Toisaalta, tuottavuuden optimointi edellyttäisi erittäin löyhästi määriteltyä sopimusta jottei kumpikaan osapuoli ennenaikaisesti sido omia käsiään. Toisaalta, ostaja helposti näkee tällaisen sopimuksena avoimena shekkinä, mikä ei luonnollisesti näytä hyvältä hänen oman esimiehensä silmin.

Käytännössä itse Scrum toimii erittäin hyvin ulkoistuskontekstissa, vaikkakin sopimusten hierominen saattaa olla vaikeampaa kuin perinteisten fixed-scope/fixed-price -sopimusten. Oleellista on ymmärtää, kuinka sopimukseen kirjoitetut sanat rajoittavat yhteistyön potentiaalia puolin ja toisin ja etsiä sopiva tasapaino.

Muut funktiot

Haluaisin vielä viimeiseksi tuoda esille näkökulman, joka usein jää huomiotta puhuttaessa eri ohjelmistokehitysmenetelmien soveltuvuudesta ja markkinasta—muut funktiot.

Kuten aiemmin tuli jo mainittua, Scrum on erittäin helppo ottaa käyttöön muuttamatta oleellisesti organisaation toimintaa. Tämä ei kuitenkaan ole koko totuus. Scrum nimittäin tuo nopeasti esille kipeitä haavoja ja organisaation epäkohtia kun toiminnan läpinäkyvyys kasvaa.

Vesiputousmalliin on helppo integroida organisaation muita funktioita, kuten markkinointi, myynti, ja strateginen johtaminen. Ehkä eniten siksi, että pitkäaikaisia suunnitelmia on helppo synkronoida. Scrum-projektin näkyvyys on kuitenkin erilainen sillä emme edes yritä ennustaa niin kauas tulevaisuuteen, vaan tarkennamme suunnitelmia liikkuvan horisontin mukana emmekä edes unelmoi tietävämme tulevaisuutta vaan keskitymme katsomaan lyhyemmän horisontin sisälle, optimoiden investointejamme vallitsevan ympäristön ja kokemusperäisen historiatiedon pohjalta.

Tämä ero fokuksessa—suunnitella koko organisaation tasolla pitkälle tulevaisuuteen vai keskittyä parantamaan koko organisaation kykyä vastata muutokseen—on se, minkä ympärille lähes kaikki käsittelemämme ”markkinan rajat” –keskustelun tekijät lopulta nivoutuvat. Scrum on helppo ottaa nimellisesti käyttöön IT-organisaatiossa muuttamatta mitään oleellista toimintatavassa, mutta Scrum on erittäin tuskallinen ottaa käyttöön koko organisaation tasolla ilman muutosta.

Ja tämä ei ole Scrumin heikkous vaan sen suurin voimavara. Scrum nimittäin ajaa meitä muokkaamaan organisaatioitamme parempaan suuntaan—murtaen teennäisiä siiloja ja puskien meitä kohti läpinäkyvää toimintaa ja terveellistä, tehokasta ekosysteemiä.

Yhteenveto

Scrum on kenties suosituin agile-menetelmä Suomessa ja vähintäänkin yksi suosituimpia maailmanlaajuisesti. Suosion taustalla löytyy monia syitä, mukaanlukien menetelmän yksinkertaisuus, matala adoptiokynnys, suurten yritysten esimerkki, sekä integroituvuus koko organisaation lukuisiin eri funktioihin IT:n ulkopuolella. Priorisoituun ominaisuuslistaan, iteratiiviseen ja inkrementaaliseen prosessiin, ja kolmeen rooliin perustuvaa Scrumia käytetään menestyksellisesti kaiken kokoisissa ja eri teknologioilla toteutettavissa projekteissa.

Lasse Koskela

Lasse on Reaktorin menetelmäspesialisti. Hän kiertää kouluttamassa ja valmentamassa asiakasorganisaatioita ketterien menetelmien ja insinöörikäytäntöjen osalta sekä kotimaassa että ulkomailla. Hän on myös kirjoittanut kirjan testivetoisesta kehityksestä (Manning Publications, 2007).