



Kuva 1. Fordin legendaarinen T-malli, joka muutti maailman ja ohjelmistokehityksen.

Lisää onnistumisia Scrumilla

Scrum on uusimpien tutkimusten valossa maailman yleisin ohjelmistokehityksen viitekehys. Scrum sopii erityisen hyvin muutosten hallintaan, riskien nopeaan minimoimiseen ja käyttäjien hyödyn maksimoimiseen. Mutta mistä oikeastaan on kysymys ja voiko Scrumia hyödyntää muussakin projektityössä kuin ohjelmistokehityksessä?

Teksti: Lare Lekman

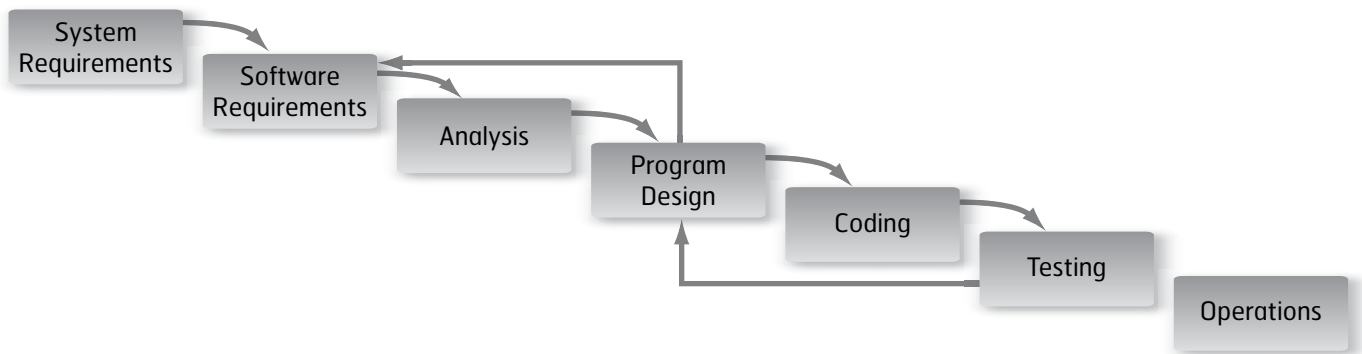
Ensin oli fordismi ja vesiputous

Henry Ford lahjoitti ihmiskunnalle 1913 ensimmäisen sarjatuotetun auton, T-Fordin, johon tehdastyöläiselläkin oli varaa. Aikansa teollinen ihme perustui 84 kokoonpanopisteeseen, joiden avulla työvaiheet voitiin erottaa toisistaan ja kouluttaa te-

hokkaasti. Tämän filosofian, *fordismin*, pohjalta kehitettiin sittemmin modernit tehtaot ja liukuhihnat.

T-Fordin sarjatuotannon kankeus ei kuitenkaan sallinut poikkeuksia työnkulkuun ja jopa auton väri säilyi samana lähes 14 vuotta.

“Any customer can have a car painted any color that he wants so long as it is black.” – Henry Ford



Kuva 2. Winston W. Roycen 1970 esittelemä prosessimalli, joka sittemmin nimettiin vesiputoukseksi.

Nuoren ohjelmistoteollisuuden etsiessä sopivaa prosessia oli luontevaa ottaa mallia valmistavan teollisuuden menestyksestä. Fordismin vaikutuksesta syntyi ohjelmistokehityksen vesiputousmalli, jonka perusteet Winston Royce esitteli 1970.

Vesiputousmallissa työvaiheet seuraavat toisiaan peräkkäin, kuten Fordin liukuhihnalla. Itse Winston Royce tosin varoitti, etteivät esimerkiksi ohjelmiston määrittely, suunnittelu, toteutus ja testaus käytännössä valmistu kerralla. Niihin on palattava uudestaan. Ohjelmistoteollisuus kuitenkin rakastui selkeältä vaikuttavaan malliin, jonka avulla työvaiheet voitiin myydä, toteuttaa ja laskuttaa tarvittaessa erikseen.

Valitse menetelmä projektin mukaan

Vesiputousmallista poiketen Scrum olettaa ratkaistavan ongelman monimutkaiseksi eli sellaiseksi, jossa tuntematonta on enemmän kuin tunnettua.

Harkitessasi Scrumin käyttämistä esimerkiksi perinteisen PMBOK-ohjeistuksen sijaan arvioi ensin, ovatko kaikki projektiin vaikuttavat mekanismit etukäteen tunnettuja vai riippuuko työn lopputulos uuden tiedon hankkimisesta ja kehitystiimin yhteistyöstä projektin aikana. Scrumia ei alun perin kehitetty etukäteen tarkasti tunnetun työn ohjaamiseen. Siihen tarkoitukseen sopinee paremmin PMBOK.

“Scrumista on vaikeaa olla pitämättä; käytämme sitä luontaisesti ollessamme selkä seinää vasten.” – Jim Coplien

Ohjelmistokehityksessä taas epäselvät vaatimukset, muuttuva teknologia, toteutuksen riskit sekä kommunikoinnin virheet ovat arkipäivää. Tällaisen kompleksisuuden ohjaamiseen soveltuu parhaiten empiriinen prosessimalli.

Scrumilla on kolme tukijalkaa

Scrumin empiriisen prosessin kolme tukijalkaa ovat läpinäkyvyys, työn tulosten säännöllinen tarkasteleminen ja työn sopeuttaminen toleransseihin, kun tulokset ovat toleranssien ulkopuolella.

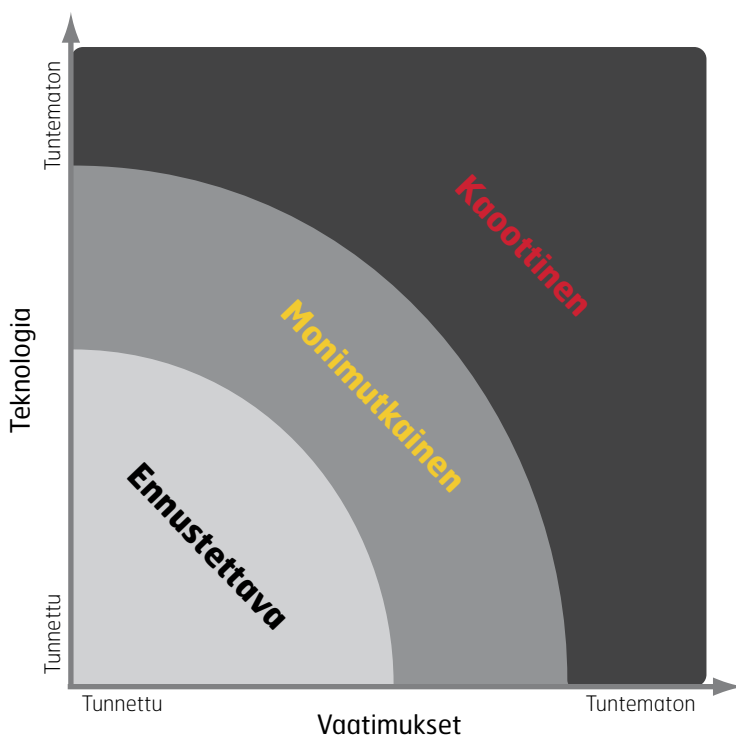
Läpinäkyvyys on kriittisen tärkeää, jotta työn etenemistä voidaan tarkastella. Esimerkiksi rakennusmestarin olisi mahdollista tehdä päätöksiä, jos rakenteilla oleva talo olisi piilossa kivimuurin takana. Tämän vuoksi Scrumissa työn tuloksia tarkastellaan esittelemällä tuotteen muuttuneet piirteet jokaisen 1-4 viikon mittaisen iteraation eli sprintin lopussa. Näin saatavan ajantasaisen ymmärryksen ja palautekeskustelun avulla voidaan siirtyä seuraavan sprintin suunnitteluun todellisuuden perustuvan näkemyksen tukemana.

Scrum vaatii harjoittelua

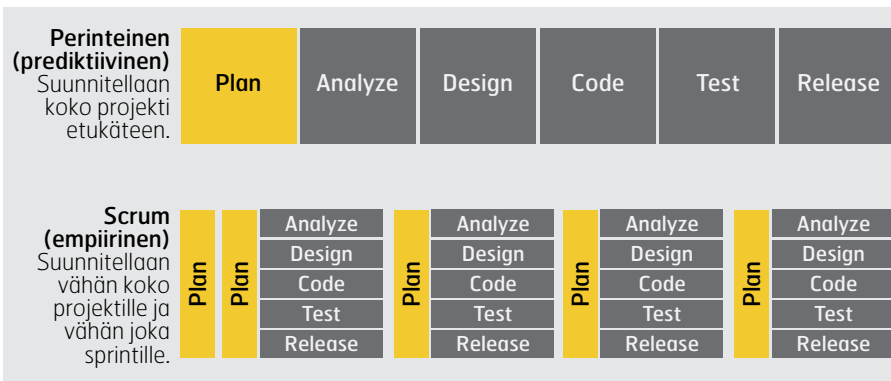
Scrumiin siirryttäessä suurimpia muutoksia ovat:

1. Aikarajatut kehitysjaksot eli sprintit.
2. Tuoteomistajan tekemä vaatimusten säännöllinen priorisointi.
3. Työmenetelmien säännöllinen arviointi ja kehittäminen *retrospektiiveissä*.

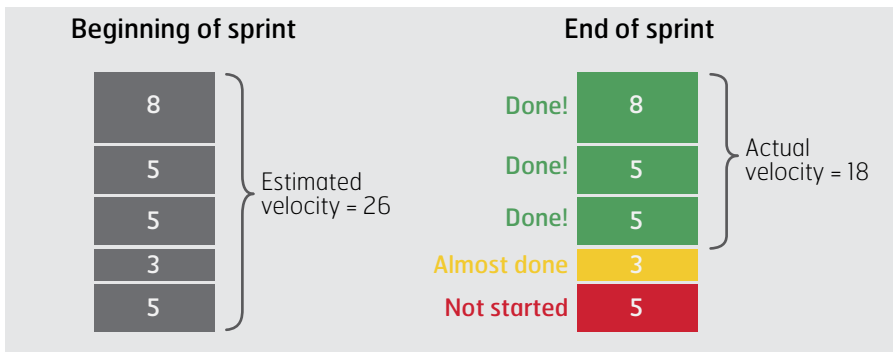
Vaatimusten priorisoinnista ja siten tuoteomistajan menestyksestä vastaa tuoteomistaja, joka valitsee kehitystiimin tuella tärkeimmät vaatimukset kuhunkin alkavaan sprinttiin.



Kuva 3. “When the process is too complicated for the defined approach, the empirical approach is the appropriate choice.”



Kuva 4. Oikea-aikainen suunnittelu pienentää projektin riskejä ja kasvattaa arvoa loppukäyttäjille.



Kuva 5. Sprinttiin valitut ja sen aikana täysin valmiiksi saadut vaatimukset. 18 työpistettä on kehitystiimin todellinen nopeus.

“Scrum muistuttaa jalkapalloa. Sen säännöt oppii nopeasti, mutta menestyksellä pelaaminen vaatii harjoittelua.” – Ken Schwaber

Sprintin aikana siinä toteutettavia vaatimuksia ei enää vaihdeta tai lisätä, vaan ainoastaan tarkennetaan. Näin kehitystiimi saa keskittyä sovittuun työhön sen sijaan, että muutoksia “lypsetään” kehitystiimiltä milloin tahansa.

Sprintin pituus kannattaa valita sellaiseksi, etteivät vaatimukset ehdi sprintin aikana muuttua, mutta kehitystiimi ehtii saada vähintään yhden vaatimuksen *täysin valmiiksi*. On syytä kiinnittää erityistä huomiota *valmiin* määritelmään, jotta kaikilla on yhtenevä käsitys siitä, mitä “valmis” tarkoittaa. Näin myös kehitystiimin antamat aika-arviot ovat yhtenäisempiä.

Sprintin yleisin pituus on 2 viikkoa. Kun sprintin pituus ja kehitystiimin henkilöt vaikiintuvat, työskentelyn rytmi opitaan ja voidaan laskea varsin tarkasti kehityksen *nopeus* eli paljonko työtä tiimi saa yhdessä sprintissä täysin valmiiksi. Esimerkiksi jos kehitystiimi on saanut kussakin edellisessä sprintissä valmiiksi keskimäärin 20 työpisteen arvosta vaatimuksia ja julkaisuun tarvitaan vielä 60 työpisteen verran vaatimuksia, voidaan arvioida, että julkaisuun tarvitaan vielä vähintään $60 / 20 = 3$ sprinttiä.

Jokaisen sprintin lopussa pidetään retrospektiivi, jossa arvioidaan edellisen sprintin työskentelyn vahvuuksia ja heikkouksia ja sovitaan kokeiltavista parannuksista.

Scrum Masterin tärkein tehtävä on pyörittää Scrum-prosessia ja poistaa esteet kehitystiimin työskentelyltä.

Scrumia voi soveltaa monipuolisesti

Scrum on kehitetty ohjelmistojen tuotekehitykseen, mutta se sopii periaatteessa minkä tahansa kompleksin työn ohjaamiseen, jossa tavoitteet voidaan lukita vähintään viikon ajaksi. Scrumia voi myös soveltaa esimerkiksi perinteisempien Prince2- ja PMBOK -projektien toteutusvaiheissa riskien minimoimiseen ja muutosten hallintaan. Koska Scrum on itsessään yksinkertainen eikä tarjoa valmiita ratkaisuja, sitä täydennetään usein insinööritason käytännöillä kuten eXtreme Programming (XP) ja Test-Driven Development (TDD).

Kanban sopii ylläpitoon

Autoteollisuus tarjoaa ohjelmistokehitykselle taas parastaan *Lean Developmentin* muodossa. *Leanissa* on paljon hyödyllisiä käytäntöjä, kuten jätteen eli turhan työn tunnistaminen ja vähentäminen. Ohjelmistokehityksessä eräs *Lean*-sovellus on *Kanban*, jossa ei ole lainkaan sprinttejä, vaan julkaisu tehdään heti, kun saadaan valmiiksi *Minimum Marketable Feature Set*.

Lare Lekman

Kirjoittaja toimii Scrum-kouluttajana ja valmentajana. Professional Scrum Trainer Certified Scrum Coach www.scrumwell.com

Kanban soveltuu erittäin luontevasti ylläpitävään työhön ja muutaman kehittäjän sovelluskehitykseen, kun yllätyksiä on paljon tai niihin tulee reagoida lähes välittömästi. Kun kehitystiimin koko kasvaa 4-9 henkeen tai kehitetään uutta tuotetta, saadaan vastaavasti etua Scrumin sprinttien tavoitteellisuudesta ja selkeästä jaksotuksesta. Ennen Kanbanin kokeilemistä kannattaa omaksua empiirinen ajattelu harjoittelemalla Scrumilla tai hankkia valmennusta.

Lähdeviitteet

Forrester Research (2010) “Agile Software Development is Now Mainstream”. CIO Magazine.

Winston Royce (1970) “Managing the Development of Large Software Systems”.

Babtunde A. Ogunnaiké & W. Harmon Ray (1994) “Process Dynamics, Modeling, and Control”. Oxford University Press.

Ken Schwaber, Jeff Sutherland & Lare Lekman (2010) “The Scrum Guide”, suomenkielinen laitos.

Henrik Kniber (2009) “Kanban vs. Scrum”.

