Teksti: Risto Nevalainen

Risto Nevalainen, vanhempi neuvonantaja, FiSMA ry.
Risto Nevalainen on toiminut pitkään systeemityön menetelmien ja laadun parissa. Hän toimi Sytyke ry:n puheenjohtajana 1980-luvun lopulla.

Ketterästi tehtyä ja laadukasta – onnistuuko?

Nykyään hypetetään paljon ketterästä kehityksestä. Se sopii jotenkin luontaisesti ohjelmistokehittäjän ajatteluun, ja kokemukseni mukaan koetaan melkein poikkeuksetta positiiviseksi. Mutta ei sekään ole tae projektin ja toimituksen onnistumisesta ja tuloksen laadusta. Ketteryys tuo mieleeni ajat, jolloin seinätekniikka koki läpimurron ja siihenkin kohdistui melkeinpä ylimaallisia odotuksia. Paljon niissä onkin samaa – mm. tiimityö, nopearytmisyys ja nopeasti jotakin valmista. Nykyinen ohjelmistotekniikka ja hyvät välineet mahdollistavat sen, mikä jäi parikymmentä vuotta sitten vielä haaveeksi. Nyt on ilo nähdä ohjelmistotuotannon ottavan ison askeleen kohti ammattimaista tekemistä ja korkeaa ennustettavuutta tekemisen ja tuloksen osalta.

1. Johdantoa

Kun näin Sytykkeen jäsenviestin tämän lehden teemasta ”hypetystä tervejärkisesti”, se kolahti minuun heti. Mieleeni tuli pikimmiten ketterä kehittäminen ja siihen kohdistunut hypetys – joka on mielestäni valtaosiltaan ansaittua. Siksipä halusin tehdä tämän (puolustus)kirjoituksen, muistellen samalla että vastaavanlaista hypetystä on ollut jo tietotekniikan alkuajoista saakka. Olen nähnyt useita satoja eri tavoin hallittuja projekteja arvioijaurani aikana, niistä parikymmentä ketterän kehityksen periaatteiden mukaisesti. Ja aina olen nähnyt saman: porukka on tavattoman innostunutta eikä koe minkäänlaista tarvetta palata menneeseen. Jotakin erinomaisuutta siinä siis täytyy olla.

Käsittelen ketterän kehittämisen hypetystä kolmessa asiassa: 1) tuottaako ketterä kehittäminen parempaa laatua kuin esim. V-malliin perustuva kehitystapa? 2) ovatko ohjelmistotuotantoon kehitetyt kyvykkyys- ja kypsyysmallit ristiriidassa vai samansuuntaisia ketterän kehityksen kanssa? ja 3) onko Scrum-tyyppinen projektijohtaminen jotenkin erilaista kuin perinteisempi peräkkäiseen tekemiseen tai iteraatioihin perustuva malli? Alustava vastaukseni kysymyksiin 1 ja 2 on pääosin myönteinen ja kysymykseen 3 varauksellinen.

Tarkoitan ketteryydellä (agile) tässä artikkelissa teknisen tekemisen ja projektinhallinnan muodostamaa kokonaisuutta, esim. Scrum-tyyppinen projektinhallinta yhdistettynä muutamaan tai useimpiin XP-periaatteisiin. Esimerkiksi käyköön pienehkö kehitysprojekti, jossa on 4 – 5 kahden viikon iteraatiota kiivaimman kehityksen aikana sekä sitä ennen ja jälkeen kuukauden tai parin esisuunnittelu ja järjestelmätestaus, päättyen onnistuneeseen toimitukseen. Oleellisimpia XP-periaatteita ovat jo määritysten tekemisen rinnalla aloitettu testauksen suunnittelu, jatkuva integrointi kunkin iteraation sisällä ja niiden välillä sekä pari/tiimityöskentely toteutuksessa.

2. Tuottaako ketterä kehitys parempaa laatua?

Olen siis pääasiassa sitä mieltä, että ketterä kehitys tuottaa parempaa laatua kuin muut elinkaarimallit. Tai kyseessä onkin joukko ohjelmistokehityksen periaatteita pikemmin kuin mikään uusi malli. Mielestäni parempi laatu on tärkein yksittäinen ketteryyttä puoltava tekijä. Juju on siinä, että lopputulos saadaan nopeasti näkyväksi ja kriittisimmät osat ovat mukana useissa tekemisen ja testaamisen kierroksissa. Ne ehtivät vakiintua ja täydentyä projektin aikana. Ehkä jotakin muuta vähemmän tärkeää jää tekemättä, mutta se onkin yksi ketteryyden perusajatuksista. ”Customer on your side” periaate tarkoittaa, että tehdään asiakastarpeen mukaisia ratkaisuja ja vakiinnutetaan ne nopeasti. Ja mikäs sen parempi tapa kuin tehdä lopputulosta alusta alkaen, esim. toimiva proto jota täydennetään ja kasvatetaan.

Laatua mitataan usein virheiden määrällä per testausvaihe. Tässä suhteessa ketterä kehitys voi olla huononnäköistä aluksi, koska vajaan tuotteen testauksessa eli toteutuksen alkuvaiheessa löytyy väkisinkin paljon virheitä. Mutta lopputulos ratkai-

see, eli isoja uusia virhekasaumia ja yllätyksiä ei pitäisi enää tulla loppuvaiheessa. Virheiden määrä voi olla siksikin suuri, kun testaus on näkyvää työtä jo toteutuksen rinnalla.

Sudenkuoppana paremmalle laadulle on määritysten liiallinen epämääräisyys. Tämä voi johtua siitä, että asiat ovat oikeasti avoinna eikä vaatimuksia siis tiedetä taikka sitten asiakastarpeet ovat turhan epämääräisiä ja jatkuvasti muuttuvia. Tämä johtaa monenkertaiseen tekemiseen ja raskaaseen iterointiin, eikä työ tahdo edistyä – päättyen usein pettymykseen ja projektin lopettamiseen. Uusi projekti tehdään sitten jollakin koetellummalla tavalla ja ketteryys muuttuu kirosanaksi. Asiakaskin valittaa, että tämä sitoo liikaa heidän avainhenkilöidensä aikaa eikä tunnu tehokkaalta.

Vaatimusten epämääräisyys on toisaalta peruste käyttää ketteriä menetelmiä. Aloitetaan sieltä, mikä jotenkin tunnetaan ja työn aikana opitaan loput. Perinteisen kiinteähintaisen projektin kannalta tämä on kuitenkin mahdoton tilanne, kun projektin laajuudesta ei ole mitään tietoa.

Koetan havainnollistaa toimittajan ja asiakkaan vuorovaikutusta oheisella kaaviolla (kuva 1), jossa vasemmalla puolella on toimittajan kyvykkyys (kuvassa valittu kolmostaso), ja oikealla asiakkaan vastaava kyvykkyys. Tilanteessa A toimittaja ei kykene vastaamaan asiakkaan koviin ja selkeisiin vaatimuksiin ja on siis juurisyy ongelmille ja huonolle laadulle. Tilanteessa B on hyvä tasapaino, eli ammattimainen toimittaja on hiukan parempi kuin ainutkertaista järjestelmää haluava asiakas. Tilanteessa C taasen asiakas on ongelma, kun sen kyvykkyys on heikko ja toiminta siten ennustamatonta. Toimittaja on silloin ikäänkuin jojon päässä, koettaen venyä milloin mihinkin.

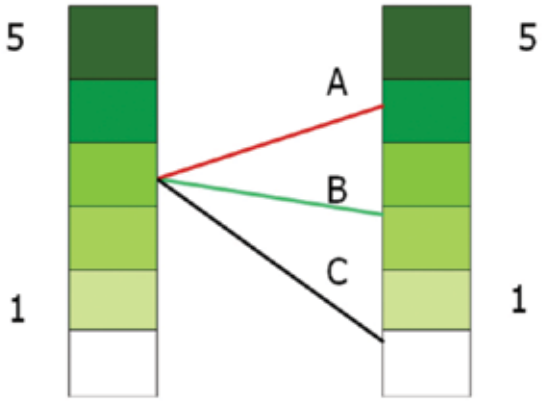
3. Onko ketterä kehitys kypsyysmallien mukaista vai niiden wvastaista?

Minulta kysytään usein, onko ketterä kehitys jotenkin uutta ja jopa vastakkaista esim. tunnetun kypsyysmallin CMMI vaatimuksille. Vastaavia muita malleja ovat esim. SPICE kyvykkyysmalli ja monilla toimialoilla käytössä olevat elinkaarimallit. Vastaan aina, että ei ole vaan ketterä kehitys on hyvinkin kyvykkyys- ja kypsyysmallien vaatimusten mukaista ja jopa edellytys nousta nopeasti ja tehokkuutta lisäten ylemmille tasoille.

Syynä epämääräiseen, ristiriitaiseen ja mielestäni väärään vastakkainasetteluun on liian puhdasoppinen ja kapea ajattelu puolin ja toisin. Koetan purkaa sitä sekä malleista että ketteryydestä lähtien.

Monet kyvykkyys/kypsyysmallien soveltajat ja arvioijat ottavat mallien vaatimukset liian kirjaimellisesti, eli vaaditaan ne toteutettavaksi sellaisenaan. Mallien tarkoitus on kuitenkin koota hyviä käytäntöjä ja yleisiä vaatimuksia, jotka voidaan toteuttaa monella eri tapaa. Yksi näistä tavoista voi olla mielestäni ketterä kehitys. Otetaan esimerkiksi CMMI-vaatimus ”Perform peer reviews”, joka saa usein heikon arvosanan suomalaisissa ohjelmistoyrityksissä. Monien mielestä se tuo mieleen vain ulkopuolisen tarkastuksen ja lisäbyrokratian, vaikka katselmoinnin toteutustapa voisi olla yhtä hyvin projektiryhmässä tehty paritarkastuskin – täydennettynä vaikkapa yksikkötesteillä. Tällainen katselmointitapa taasen on hyvinkin ketterien periaatteiden mukaista, pidetäänhän edistymisestä ja teknisistä tuloksista joka-aamuinen kokouskin!

Ketterän kehityksen mukaisesti mielestään elävät toimivat usein perusajatusten vastaisesti tai tulkitsevat niitä liikaa edukseen. Otetaan esimerkiksi idea ”documented as needed”. Se tarkoittaa, että tehdään vain tarpeelliseksi koettuja kuvauksia, muut jätetään tekemättä. Käytännön tilanne



Kuva 1: Toimittajan ja asiakkaan periaatteelliset asetelmat keskinäisissä kyvykkyystasoissa. Toimittajan oletettu kyvykkyys vasemmalla ja asiakkaan oikealla.

on usein, että dokumentaatio jää tekemättä tai lopuksi harsitaan jotakin asiakasdokumentaatiota muistuttavaa. Tästä saa tietysti huonon arvosanan CMMI-arvioinnissa, ja mallin vaatimus koetaan byrokratiaksi ja turhan työn teettämiseksi. Mutta ei ”documented as needed” suinkaan tarkoita, että dokumentteja tehtäisiin vain suunnittelijoita itseään varten ja muista viis. Kyllä niitä tarvitaan myöhemminkin mm ylläpidon ja jatkokehityksen tarpeisiin. Eli lopputuloksena on jotakuinkin myös CMMI-vaatimuksia vastaava dokumentointitapa eli en näe tässä mitään ristiriitaa.

Toki monet standardoidut mallit ovat aika raskaita etenkin projektinhallinnan puolella, eivätkä ketterän kehityksen noudattajat pidä niistä. Mutta vika onkin näissä vakioiduissa toimintatavoissa, joissa ei ole otettu tarpeeksi huomioon ketterän kehityksen tarpeita. Ei kahden viikon sprinttiä kannata suunnitella täysimääräisen projektin lailla. Mutta ei se tarkoita, ettei suunniteltaisi lainkaan. Ketterän kehityksen tyypillinen toteutus on pitää kunkin sprintin alussa työpaja, jossa kootaan edellisen kauden tulokset ja suunnitellaan

seuraavan kauden (esim. kahden viikon) työt. Itse asiassa ne suunnitellaan usein hyvinkin tarkasti, ja tekijäporukka niihin sitouttaen.

Olen nähnyt suuren määrän ketterän kehityksen mukaisia projekteja, joissa kokonaisuuden suunnittelu on jäänyt vähälle, kun luotetaan pala palalta tekemiseen. Tämä on yksi sudenkuopista, johtaen useimmiten paljon suurempaan työmäärään ja sprint-askeleisiin kuin oli alunperin aavistuskaan. Eli ei ketterä kehitys poista projektinhallinnan ja ”top down” näkemyksen tarvetta, päinvastoin korostaa sitä jotta kukin yksittäinen askel menee oikeaan suuntaan.

Mallien vaatima mittarointi toteutuu mielestäni erinomaisesti ketterässä kehityksessä. Työmaata seurataan sekä backlog tilanteen että burn-down asteen avulla, sekä tehdään kehityssyklien välissä katselmointeja (niitä kutsutaan usein retrospective review – nimellä). Tämä jos mikä on esimerkiksi CMMI:n vaatimusten mukaista ja on itse asiassa ponnahduslauta tasojen 4 ja 5 vaatimaan mittarointiin. Onkin itse asiassa niin, että ketterän kehityksen mukainen toiminta saa kokonaisuutena parempia tasolukuja prosessiarvioinneissa kuin perinteiset tavat. Tähän vaikuttaa itse systematiikan lisäksi hyvät välineet, joiden avulla tekemistä tuetaan tai automatisoidaan.

”Koko ketterän kehityksen pahimpana sudenkuoppana pitäisin sen suurinta ansiotakin eli nopea-tempoisuutta.”

4. Onko Scrum parempi tapa hallita ohjelmistoprojekteja?

Ketterän kehityksen projektinhallintana on useimmiten Scrum eli hyvin mietitty jaksottainen kehityskausi (usein käytetään nimitystä sprint), jonka aikana panostetaan tiimityöhön, viestintään, näkyvään suunnitteluun ja tiimikeskeiseen kehittämiseen. Kukin sprint sisältää enemmän tai vähemmän sekä määrittelyä, toteutusta, testausta että toimittamista. Tietysti ihan projektin alussa painottuu määrittely, lopussa enemmän testaus, virheenkorjaus ja lopulta täyden toimituksen tekeminen.

No, eihän tuossa kuulosta olevan mitään uutta. Eikä olekaan! Vanha kunnon hyvän suunnittelun periaate toteutuu, ja useimmat meistä näkevät tekemisessä jopa peräkkäisiä vesiputouksen tapaisia vaiheita, nyt vain useampaan kertaan tehtynä. Tältä osin vanhat opit siis toistavat itseään. Jo seinätekniikan kultakaudella 1980-luvulla näimme vastaavia elinkaaria kukkuloineen kuin nyt RUP-mallin kalvoissa.

Hyvänä puolena scrum-menettelyssä pitäisin edistymisen hyvää tietämistä. Hidastempoisemmassa projektissa ei paljastu heti, jos jotakin vaatimusta ei osata toteuttaa. Sitä pallotellaan, ja uskotellaan edistymistä tapahtuvan. Karu piirre scrumissa on, että myös osaamattomuus paljastuu nopeasti eli jotkut meistä kokevat putoavansa kärryiltä. Onhan se toisaalta hyväkin että vastuita joudutaan miettimään nopeasti uusiksi, kunhan se vain tehdään harkitusti ja muitakin kuin teknisen tekemisen taitoja kunnioittaen.

Koko ketterän kehityksen pahimpana sudenkuoppana pitäisin sen suurinta ansiotakin eli nopeatempoisuutta. Koko ajan on joku takaraja lähellä saavuttaa tuloksia. Se panee yrittämään sekä yksilön että tiimin tasolla, onhan yhdessä onnistumisen kokemus hieno asia. Mutta tähänkin voi turtua ja väsyä.

Pari vuotta sitten pidimme FiSMAn piirissä työpajan projektinjohtamisen ja ohjelmistokehityksen kokemuksista. Emme hyväksyneet sinne muita kuin todistuskelpoisia, eli tilastoihin ja analyyseihin perustuvia tuloksia. Pääpuhujamme Khaled-El Emam näytti käyriä, joiden mukaan noin kolmen kuukauden iteraatio näyttäisi olevan kokonaisuuden kannalta paras ja pitkän päälle tehokkain tapa. Tämä on aika kaukana siitä viikon tai kahden rumbasta, jota puhdaslinjaisimmat ketterän kehityksen yritykset käyttävät. Mutta tästä näyttäisi seuraavan burn out ennemmin taikka myöhemmin. Tämä ilmiö ei ole vielä tullut vahvasti esiin suomalaisessa työkulttuurissa, jossa on ollut tilaa omalle vastuunotolle ja räätälöidyille tehtävänkuville.

5. Lopuksi

Olen koettanut yllä karsia ketteryyden hypetystä rajatulta osin ja omiin kokemuksiini perustuen. Kaikista pienistä epäkohdista huolimatta näen sen suurena kehitysaskeleena ohjelmistotuotannossa, ja jopa ikiaikaisten ohjelmoinnin ihanteiden toteutumisena. Muutama vuosi sitten kuuntelin Scott Amblerin palopuhetta ketterän kehityksen ylivoimaisuudesta. Se kuulosti minun korviini aivan siltä, kuin olemme aina haaveilleet tekevämme ohjelmistoja ja systeemityötä. Itse asiassa se kuulosti siltä, miten olen nähnyt systeemityötä tehtävän suomalaisissa yrityksissä jo pitemmän aikaa. Teknologia ja välineet eivät ole vain antaneet periksi aiemmin eivätkä ole olleet tarpeeksi joustavia. Nyt ovat, eli tämä hype on toteutumassa!