Sulautettujen järjestelmien

ketterä käsikirja

Lehtonen, Tuomivaara, Rantala,
Känsälä, Mäkilä, Jokela, Könnölä,
Kaisti, Suomi, Isomäki & Ylitolva

Sulautettujen järjestelmien

ketterä käsikirja

Teijo Lehtonen, Seppo Tuomivaara, Ville Rantala, Marja Känsälä,
Tuomas Mäkilä, Tero Jokela, Kaisa Könnölä, Matti Kaisti,
Samuli Suomi, Minna Isomäki ja Marko Ylitolva

Turun yliopisto
Työterveyslaitos

TEKES

ISBN: 978-951-29-5837-5 (sid.)
978-951-29-5838-2 (pdf)

Toimitus: Kaisa Könnölä ja Minna Isomäki
Taitto: Marko Ylitolva ja Tuomas Mäkilä
Kuvitus: Tuomas Mäkilä
Web-toteutus: Samuli Suomi

Painopaikka: Painosalama Oy
Turku 2014

Esipuhe

Ketteryys tarkoittaa joustavaa, nopeaa ja hallittua reagointia käsillä olevaan tilanteeseen. Vaikka ketterä kehitys vastaa tuottavuuden haasteisiin, edistää se myös hyvin toteutettuna henkilöstön hyvinvointia. Ketterien menetelmien käyttö sulautettujen järjestelmien kehityksessä ei ole ollut yhtä yleistä kuin ohjelmistokehityksessä. Menetelmien edut ovat kuitenkin hyödynnettävissä myös sulautettujen järjestelmien kehityksessä. Ketteryyden omaksuminen ei ole ilmaista, vaan vaatii niin organisaatiolta kuin yksilöiltäkin panostusta. Työ on kuitenkin palkitsevaa, kun siinä pääsee alkuun.

Tämä käsikirja ketterien menetelmien käyttöönoton tueksi sulautettujen järjestelmien kehityksessä on syntynyt Turun yliopiston ja Työterveyslaitoksen tutkijoiden yhteistyönä Tekesin rahoittamassa *AgiES: Agile and Lean Product Development Methods for Embedded ICT Systems* -projektissa. Työn ovat mahdollistaneet hankkeeseen osallistuneet ketteryyttä soveltavat ja käyttöönottavat yritykset sekä muut kumppanit. Haluamme kiittää hyvästä yhteistyöstä tutkijoita, kumppaneita, rahoittajia sekä muita hanketta edistäneitä tahoja.

Käsikirjan käyttöohje

Käsikirjan tavoitteena on toimia johdantona, tiedonhaun lähtökohtana sekä konkreettisten käytännön ohjeiden antajana monille erityyppisille lukijoille kehitystiimin jäsenestä aina organisaation johtoon. Kirjan jokainen aukeama on kirjoitettu omaksi kokonaisuudekseen, jolloin lukija voi poimia tietoa ketteryydestä ja sen käyttöönotosta tarpeensa mukaan. Tekstissä ohjataan asiasta lisää kertovaan kirjallisuuteen, jota täydentää kirjan loppuun kerätty kirjallisuusluettelo. Kirjan lopussa on myös sanasto, joka esittelee kirjassa käytetyt yleisimmät ketteryyden käsitteet englanninkielisine käännöksineen.

Kirja jakautuu neljään värein eroteltuun pääosioon:

- Ensimmäinen osio johdattaa ketteriin menetelmiin, sulautettujen järjestelmien suunnitteluun sekä työhyvinvointiin. Osio tarjoaa näistä teemoista ne perussisällöt, jotka on kirjan sisällön kannalta keskeistä ymmärtää.

- Toinen osio johdattaa ketterien menetelmien soveltamiseen sulautettujen järjestelmien suunnittelussa. Osio käsittelee ketteryyteen valmistautumista, itse käyttöönottoa, työkalujen hyödyntämistä sekä muutostilanteen hallintaa. Osio päättyy ketteryyden elinkaarimallien esittelyyn. Mallit opastavat kolmannen tekniikkakorttiosion sisältöön ja käyttöön.

- Kolmas osio koostuu 17 kaksipuoleisesta tekniikkakortista, jotka on ryhmitelty ketterää prosessia, ketteryyden perustekniikoita, dokumentointia ja työkaluja, vastuita ja rooleja sekä sulautetun kehittämisen omia käytäntöjä esitteleviin kortteihin. Kukin kortti esittelee yhden ketteryyteen liittyvän käytännön tai työkalun. Tekniikat esitellään mahdollisimman konkreettisesti sekä havainnollistetaan käytännön esimerkeillä. Tekniikoiden toteuttamisohjeet on koottu helposti muistettaviksi resepteiksi.

- Viimeinen osio esittelee AgiES-projektin sekä sen osana toteutetut tapaustutkimukset projektin kumppaniyritysten tuotekehitystiimeissä. Yritysesimerkit kertovat, miten yrityksissä lähdettiin käytännössä kehittämään tiimien toimintaa ketterämpään suuntaan ja miten yritykset ovat kokeneet ketteryyden omassa toiminnassaan.

Käsikirja on julkaistu myös nettisivustona osoitteessa embedded.utu.fi/kasikirja. Sivustolta löytyy paitsi kirjan sähköinen versio, myös linkkejä lisämateriaaleihin sekä mahdollisuus osallistua keskusteluun ketteristä menetelmistä ja niiden soveltamisesta.

Toivomme, että tästä käsikirjasta on apua tiimeille ja organisaatioille heidän pyrkiessään kehittämään toimintaansa joustavammaksi, sujuvammaksi ja tuottavammaksi kestävällä tavalla.

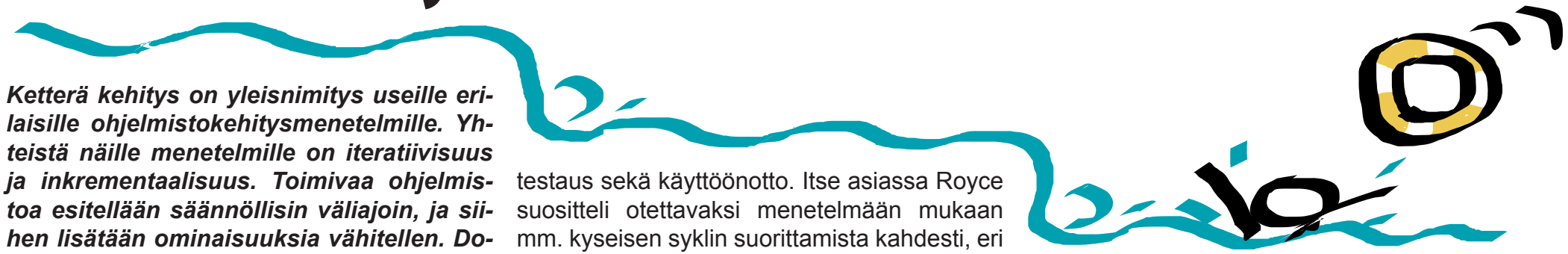
Syyskuussa 2014,

Teijo Lehtonen ja Seppo Tuomivaara

Sisällys

Johdanto

Ketterä kehitys



Ketterä kehitys on yleisnimitys useille erilaisille ohjelmistokehitysmenetelmille. Yhteistä näille menetelmille on iteratiivisuus ja inkrementaalisuus. Toimivaa ohjelmistoa esitellään säännöllisin väliajoin, ja siihen lisätään ominaisuuksia vähitellen. Dokumentaatiossa keskitytään olennaiseen ja suunnitelmat muokkautuvat kehityksen edetessä. Vuorovaikutus johtoportaan, kehittäjien ja asiakkaan välillä on sujuvaa ja tiivistä.

Historia

Yleisesti ajatellaan, että iteratiiviset ja inkrementaaliset menetelmät ovat tulleet käyttöön 2000-luvun vaihteessa. Kuitenkin niillä on pohjansa jo kauempana ohjelmisto- ja tuotekehityksen historiassa.

Jo 1930-luvulla esitettiin syklien käyttöä laadunvarmistuksessa. Iteratiivisia ohjelmistonkehitysmenetelmiä oli käytössä myös mm. NASAssa ja IBM:ssä 1960- ja 1970-luvuilla.

1970-luvulla Winston Royce julkaisi artikkelin, jossa hän esitteli kokemuksiinsa perustuvan kehitysmenetelmän. Tässä *vesiputousmallissa* työvaiheet seuraavat toisiaan: vaatimusten määrittely, analyysi, suunnittelu, ohjelmointi, testaus sekä käyttöönotto. Itse asiassa Royce suositteli otettavaksi menetelmään mukaan mm. kyseisen syklin suorittamista kahdesti, eri vaiheiden välisiä iteraatioita sekä asiakkaan ottamista mukaan suunnittelutyöhön. Yleisesti mallia kuitenkin seurattiin virheellisesti yksinkertaisessa muodossaan, jonka jo Royce totesi artikkelissaan riskialttiiksi.

1970- ja 1980-luvuilla vesiputousmalli oli pääasiallinen ohjelmistokehitysmenetelmä. Rinnalla kuitenkin esiintyi jo monenlaisia iteratiivisempia menetelmiä. 1990-luvulla vesiputousmallia alettiin pitää ongelmallisena ja kehitettiin paljon erilaisia kevyempiä menetelmiä. Joissakin näistä taustalla oli tuotantoteollisuudesta tullut Lean-ajattelutapa (s. 8).

Ketteryyden alkuperänä pidetään useimmiten *ketterän ohjelmistokehityksen julistusta* (Agile Manifesto), jonka 17 ohjelmistoalan ammattilaista kirjoittivat helmikuussa 2001 Utahissa. Julistus oli oikeastaan kokoelma niistä arvoista, jotka 1990-luvulla uusia menetelmiä kehittäneet ohjelmistoammattilaiset olivat kokeneet hyviksi. Julistus sisältää arvojen lisäksi 12 periaatetta.

Ketterän ohjelmistokehityksen julistuksen arvot

Löydämme parempia tapoja tehdä ohjelmistokehitystä, kun teemme sitä itse ja autamme muita siinä. Kokemuksemme perusteella arvostamme:

Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja

Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota

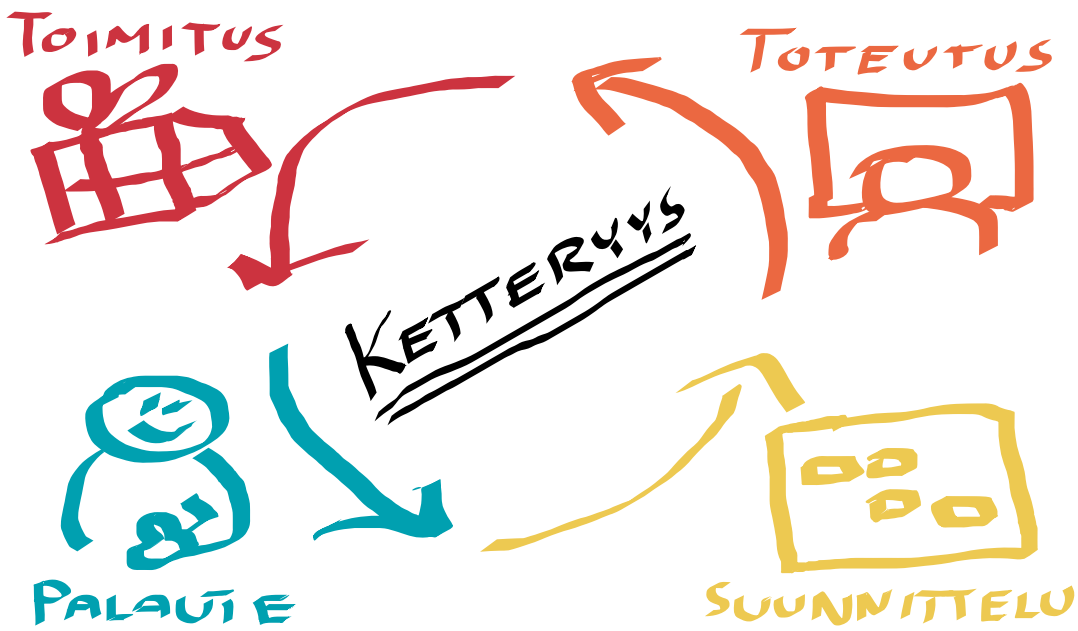
Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja

Vastaamista muutokseen enemmän kuin pitäytymistä suunnitelmassa

Jälkimmäisilläkin asioilla on arvoa, mutta arvostamme ensiksi mainittuja enemmän.

Arvot

Arvot muodostavat pohjan ketterille kehitysmenetelmille ja niitä tulisi soveltaa silloin, kun menetelmä tai käytäntö ei suoraan vastaa ratkottavaan ongelmaan.



Yksilöt ja kanssakäyminen

Ketterissä menetelmissä annetaan kehittäjille vastuuta omasta työstään ja oman työnsä suunnittelusta. Toimivaan kommunikaatioon kiinnitetään huomiota niin oman tiimin sisällä kuin tiimin ulkopuolellakin. Tiimi sopii yhdessä työtehtävistä ja kehittää työmenetelmiään jatkuvasti. Tiimi ottaa kantaa paitsi tekniseen sisältöön, myös siihen, mitä on realistista saada toteutettua tietyssä ajassa. Käytännön työ pohjautuu toistensa kanssa kommunikoiviin yksilöihin, joita käytössä olevat työkalut ja menetelmät tukevat.

Toimiva ohjelmisto

Pääasiallinen kehityksen mittari ketterissä menetelmissä on säännöllisesti toimitettava toimiva ohjelmisto. Työ paloitellaan pieniin paloihin siten, että ohjelmistoon lisätään ominaisuuksia inkrementaalisesti ja jokainen lisätty osa on tehty ja testattu toimivaksi asti. Dokumentaatio ei ole itseisarvo, vaan tukee työn tekemistä.

Asiakasyhteistyö

Asiakas otetaan mukaan suunnitteluun koko ohjelmistokehityksen ajan sen sijaan, että alussa tehtäisiin tarkka sopimus ja vasta projektin lopussa asiakas näkisi valmiin tuotteen. Tiimi esittää asiakkaalle säännöllisin väliajoin työnsä tuloksia ja saa asiakkaalta palautetta. Näin kaikki osapuolet pyrkivät koko ajan kohti yhteistä tavoitetta. Tavoite muokkautuu matkan varrella täyttämään paremmin asiakkaan tarpeet, kun molempien osapuolien ymmärrys kehitettävästä tuotteesta lisääntyy.

Muutokseen vastaaminen

Kehitystyön aikana tuotteen vaatimukseen tulee muutoksia. Projektin edetessä sekä tiimi, että asiakas ymmärtävät paremmin, miten asiakkaan tarpeet saadaan parhaiten täytettyä. Myös ympäristö saattaa muuttua ja asettaa uusia vaatimuksia projektille. Alkuperäiset suunnitelmat eivät ole yksityiskohtaisia ja tarkkoja, vaan niitä muokataan projektin edetessä muutosten osoittamaan suuntaan. Suunnitelmia tehdään kullakin ajan hetkellä vain niin pitkälle kuin on tarpeellista työn etenemisen kannalta.

Lisää aiheesta

Beck K. ym.: *Agile Manifesto*. http://agilemanifesto.org/.

Larman C.: *Agile and Iterative Development: A Manager's Guide*.

Esimerkki: Scrum

Scrum on yleisin tällä hetkellä käytetyistä ketteristä menetelmistä. Se tarjoaa muutamia sääntöjä, jotka keskittyvät ihmisten väliseen vuorovaikutukseen ja yhteistyöhön. Scrumin kulmakivenä ovat itseohjautuvat tiimit, päivittäisen työn etenemisen seuranta sekä tarkat ennalta määrätyt kevyet seremoniat. Vaikka Scrumin perusperiaatteet ovat yksinkertaisia, sen käytännöt vaikuttavat merkittävästi tiimin toimintaan ja työskentelytapoihin.

Prosessi

Scrumissa tuote kehittyy vähitellen yleensä 1–4 viikon mittaisten iteraatioiden (s. 35) aikana. Näitä jaksoja kutsutaan *sprinteiksi*.

Sprintin aikana tiimi tapaa *päivittäisissä tilannepalavereissa* (s. 43), joiden tarkoituksena on päivittää tiimin jäsenten työn eteneminen ja poistaa siinä mahdollisesti esiintyvät ongelmat. Jokaisen sprintin alussa järjestettävässä *suunnittelupalaverissa* (s. 37) tiimi yhteisesti valitsee työtehtävät, jotka se sitoutuu sprintin aikana tekemään. Sprintin tuotokset katselmoidaan sprintin lopuksi *katselmointipalaverissa* (s. 39), jossa esitellään toimiva tuote sprintin aikana valmistuneiden ominaisuuksien kautta.

Työtehtäviä hallinnoidaan *iteraation kehitysjonolla* sekä *tuotteen kehitysjonolla* (s. 45). Scrum tarjoaa lisäksi työkalun itse prosessin kehittämiseen. Jokaisen sprintin lopuksi pidetään *retrospektiivi* (s. 41), jossa tiimi käy läpi edellistä toteutettua sprinttiä ja miettii, kuinka tiimin työskentelytapoja voisi kehittää entistä paremmiksi.

Tiimi ja roolit

Scrum-tiimi työskentelee itsenäisesti ja itseorganisoituvasti (s. 51). Tiimi voi itse päättää työskentelymenetelmistään ja työnjaostaan. Vaikka Scrum-tiimissä ei olekaan tavanomaisista hierarkiaa, siinä on kuitenkin kolme oleellista roolia.

Tuoteomistaja
Yksi tiimin jäsenistä toimii tuoteomistajana (s. 53). Tämä voi olla myös asiakas, mutta yleensä tuoteomistajan tulisi olla täysipäiväinen tiimin jäsen. Hän on vastuussa tuotteen kehitysjonon luomisesta, ylläpidosta ja priorisoinnista. Hän valitsee sprintin tavoitteet tuotteen kehitysjonosta jokaiseen uuteen sprinttiin sekä katselmoi työn tuotoksen iteraation katselmointipalaverissa asiakkaan kanssa.

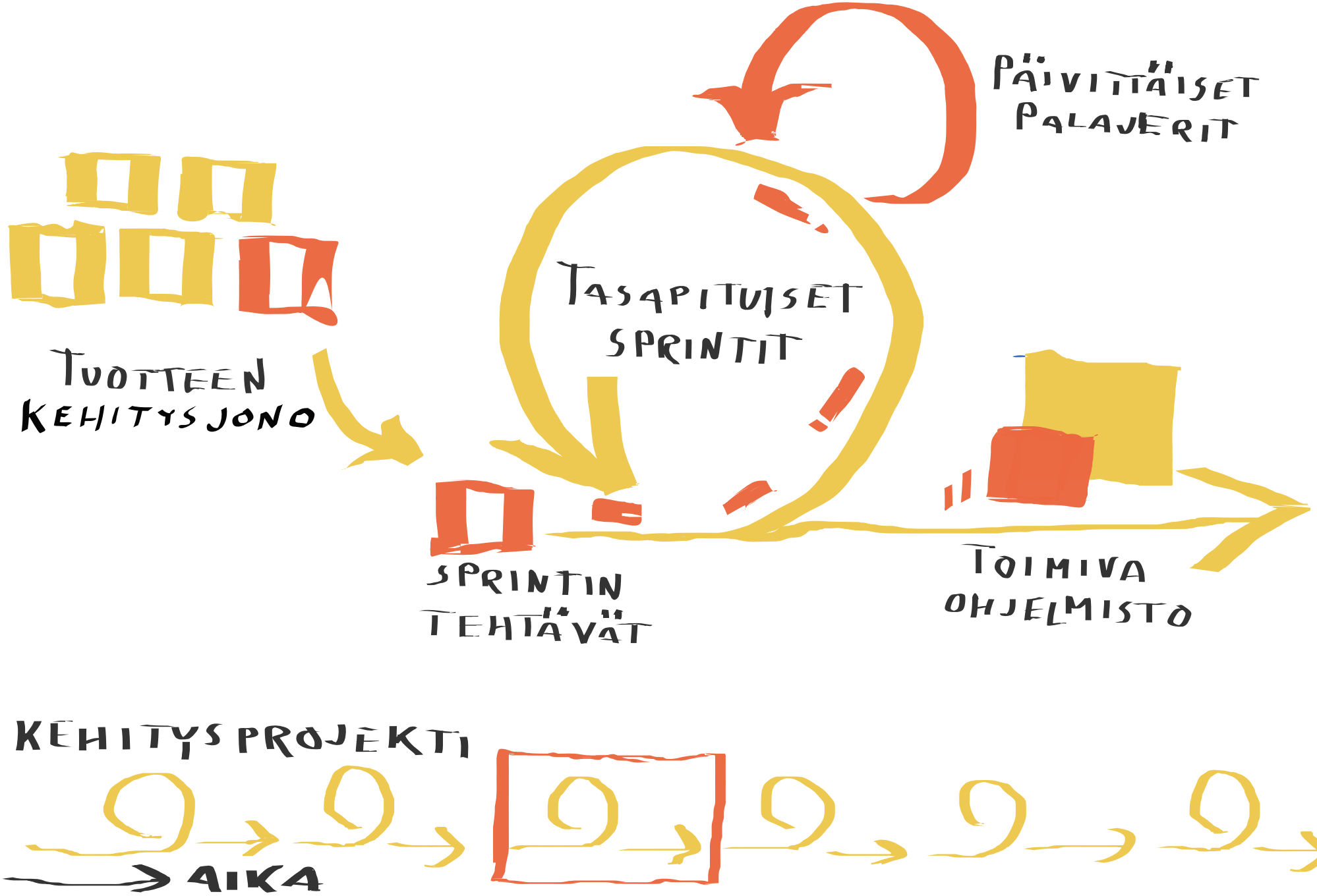
Scrummaster
Yksi tiimin jäsenistä toimii tiimin fasilitoijana (s. 52), scrummasterina. Hän huolehtii siitä, että tiimi noudattaa Scrumin periaatteita ja käytäntöjä. Lisäksi hän varmistaa työrauhan tiimille sprintin ajaksi ja poistaa työn etenemisen esteet niiden ilmaantuessa. Hän myös yleensä organisoi päivittäiset pikapalaverit, iteraation katselmointipalaverin, iteraation suunnittelupalaverin sekä retrospektiivin.

Scrum-tiimin jäsen
Tiimit jäsenet saavat keskittyä sprintin aikana itse kehitystyön tekemiseen. Jokaisessa sprintissä tiimi toteuttaa iteraation kehitysjonoon valitut tehtävät, jotka on yhdessä valittu tehtäviksi.

Lisää aiheesta

Schwaber K., Sutherland J.: *The Scrum Guide*. http://bit.ly/ZgrC7Z.

Schwaber K., Beedle M.: *Agile Software Development with Scrum*.

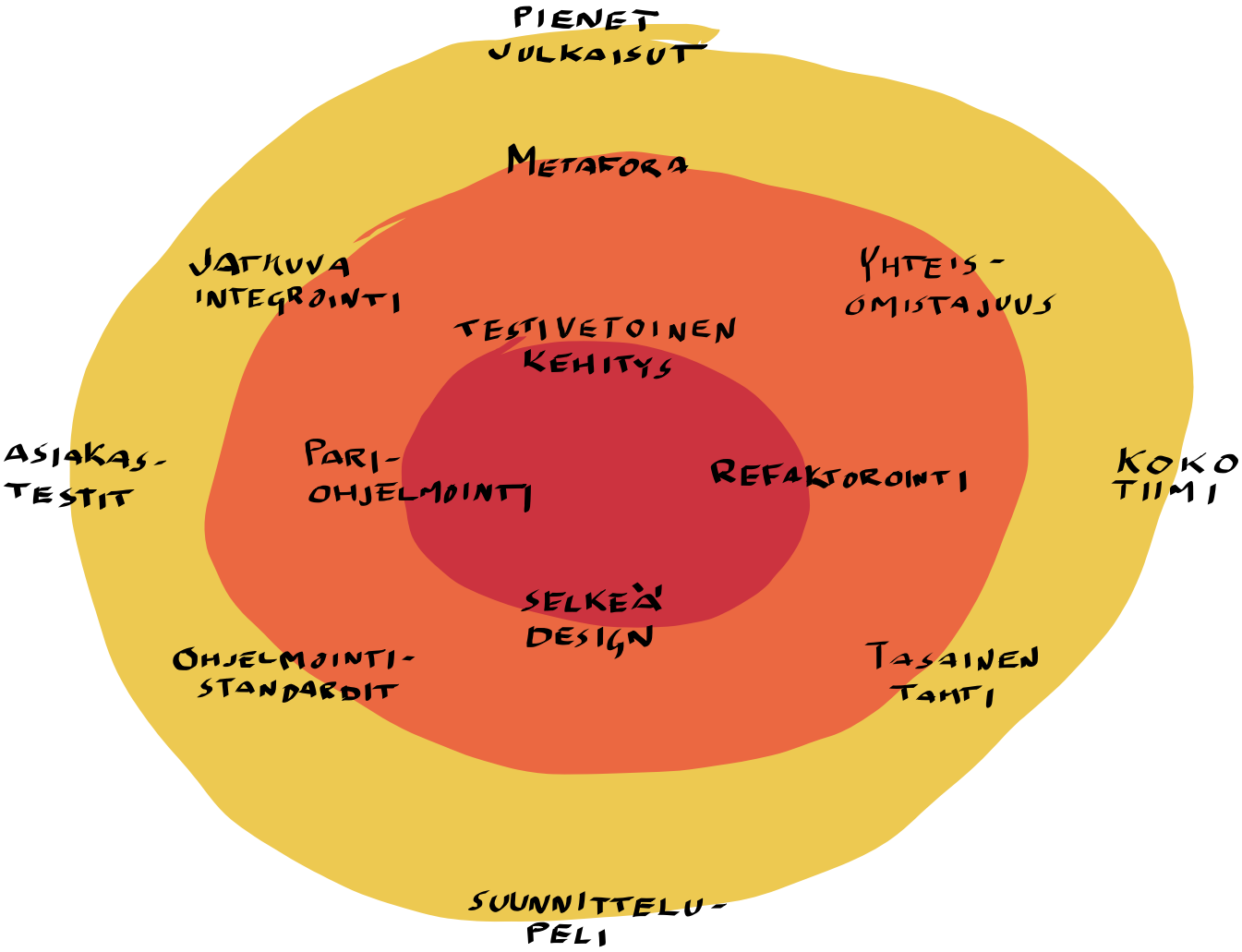

PÄIVITTÄISET PALAVERIT
TASAPITUISET SPRINTIT
TUOTTEEN KEHITYSJONO
SPRINTIN TEHTÄVÄT
TOIMIVA OHJELMISTO
KEHITYSPROJEKTI
AIKA

Esimerkki: Extreme Programming (XP)

Extreme Programming (XP) painottaa muiden ketterien menetelmien tapaan mukautuvuutta enemmän kuin ennustettavuutta. Menetelmän tarkoitus on parantaa sekä ohjelmiston laatua että tiimin kykyä vastata asiakkaan vaatimusten muuttumiseen.

XP:ssä useiden toistuvien ohjelmistojulkaisujen ja lyhyiden kehityssyklien tarkoitus on parantaa tuottavuutta ja tarjota tarkastuspisteitä kehityssyklien välissä. Iteraatioiden välillä uudet asiakasvaatimukset voidaan käydä läpi välittömästi ja tuoda mukaan kehitystyöhön. XP on tarkoitettu melko pienille projektitiimeille, joissa iteraation pituus vaihtelee yhden ja kolmen viikon välillä. Tämän lisäksi XP painottaa tasaista työtahtia, jossa ylitöiden tekoa pääsääntöisesti vältetään.

XP perustuu viiteen ydinarvoon sekä niiden pohjalta luotuihin periaatteisiin ja käytäntöihin. XP:n arvot ja käytännöt on suunniteltu siten, että muutoksen kustannus pysyy projektin ajan suunnilleen samana, eikä kasva eksponentiaalisesti ohjelmistoprojektin edetessä. Jos tähän tavoitteeseen päästään, tiimi voi toimia ketterästi toteuttaen haluttuja muutoksia sujuvasti myös projektin loppupuolella.



XP:n käytäntöjä

Kehittäjien päivittäiset käytännöt

Kehittäjät työskentelevät *pareittain* tarjoten arvokasta palautetta toisilleen ja kehittävät samalla laajempaa ymmärrystä järjestelmän toiminnasta. Ohjelmisto pidetään mahdollisimman *yksinkertaisena* ja sitä parannetaan jatkuvasti *refaktoroimalla*. *Testivetoisessa kehityksessä* (s. 63) ohjelma rakennetaan yksikkötestien kautta, jolloin kaikki tuotettu koodi on toimivaa. Näin kasvatetaan asiakkaan ja kehittäjien luottamusta ohjelmistoa ja sen laatua kohtaan.

Tiimin työtä tukevat käytännöt

Ohjelmisto kehitetään yhteisen kielen, *metaforan* kautta. Tämän ja *jatkuvan integroinnin* avulla järjestelmä on koko ajan toimiva. Tiimillä on yhteiset *ohjelmointistandardit* ja *tiimi omistaa yhteisesti kaiken koodin*, jolloin käytännössä kuka tahansa pareista voi tehdä tarvittavat muutokset. Tämän lisäksi tiimin tavoitteena on työskennellä *tasaisella tahdilla* noin 40 tunnin työviikolla ilman merkittävää ylityötä. Tiimillä on selkeät pelisäännöt, jotka auttavat kohti nopeaa ja laadukasta toteutusta.

Kokonaisuuden hallinnan käytännöt

Koko tiimi työskentelee yhdessä samassa tilassa asiakkaan kanssa saaden jatkuvaa palautetta niin muilta tiimiläisiltä kuin asiakkaaltakin. *Suunnittelupelissä* otetaan huomioon asiakkaan laatimien käyttäjätarinoiden tekniset, aikataululliset ja liiketaloudelliset seikat ja valitaan mitkä käyttäjätarinat toteutetaan iteraation aikana. Jokaisen lyhyen iteraation lopuksi on olemassa toimintakuntoinen *pieni julkaisu*. Asiakas määrittelee *asiakastestit*, jotka tuotteen on läpäistävä ennen kuin sen voidaan sanoa täyttävän asiakasvaatimukset. Niin asiakas kuin kehittäjätkin tietävät koko tuotekehityksen ajan tuotteen realistisen tilanteen ja näin tuote saadaan vastaamaan sitä, mitä asiakas tarvitsee.

Lisää aiheesta

Wells D.: *Extreme Programming – a gentle introduction*. http://bit.ly/1tuWFpx

Beck K.: *Extreme Programming Explained: Embrace Change, 2nd Edition*.

XP:n arvot

Kommunikaatio – Käytäntöjen on tarkoitus pitää sekä tiimin sisäinen että tiimin ja sidosryhmien välinen kommunikaatio sujuvana.

Yksinkertaisuus – On edullisempaa tehdä aluksi yksinkertaisin toteutus, jota voidaan joutua muokkaamaan myöhemmin, kuin tehdä heti monimutkainen ratkaisu, jota ei välttämättä ikinä tarvita.

Palaute – Konkreettista palautetta saadaan kehitettävältä järjestelmältä yksikkötestien muodossa, asiakkaalta hyväksymistestien muodossa sekä tiimin sisällä tiiviin yhteistyön kautta. Palautteen kautta tuote kehittyy oikeaan suuntaan.

Rohkeus – Äärimmilleen viedyt käytännöt vaativat rohkeutta. Tiimi kertoo rohkeasti tuotteen todellisen tilanteen. Ohjelmiston osia poistetaan ja kirjoitetaan uudelleen tarpeen tullen. Uusia lähestymistapoja kokeillaan parhaan lopputuloksen saavuttamiseksi.

Kunnioitus – Tiimin jäsenellä on oikeus ottaa vastuu omasta työstään ja tehdä se parhaaksi näkemällään tavalla. Tiimi kunnioittaa asiakkaan mielipiteitä ja asiakas tiimin asiantuntemusta.

Esimerkki: Kanban ja Lean

Lean on johtamisoppi, jossa pyritään poistamaan hukkaa ja kehittämään toimintaa jatkuvasti. Lean-ohjelmistokehitys on Lean-ajattelun soveltamista ohjelmistojen kehitykseen. Kanban on menetelmä, jonka avulla organisaatio voi muuttaa nykyisiä prosessejaan Leanin periaatteiden mukaisesti.

Lean-ajattelun keskeisin periaate on hukan eliminointi, joka tarkoittaa kaikkea resursseja kuluttavaa työtä, joka ei tuota asiakkaalle lisäarvoa. Hukan eliminoimiseksi tuotteen arvoketju on tehtävä näkyväksi.

Kanbanin yhtenä merkittävänä tavoitteena on optimoida arvoketju. Työvuon seurannalla voidaan määrittää miten arvo kulkeutuu järjestelmän läpi. Tarkastelemalla arvoketjua ja sen kehitystä voidaan ongelmakohdat löytää ja hakea niihin ratkaisuja. Jotta prosessia voidaan lähteä parantamaan, se pitää ymmärtää. Siksi prosessi pitää tehdä näkyväksi ja julkiseksi. Ilman selkeää ymmärrystä prosessista, siihen liittyvien ongelmien ratkaisu ei ole tehokasta.

Kanban-taulu

Yksi tapa työvuon visualisointiin on käyttää Kanban-taulua. Kanban-taulu koostuu sarakkeista ja työtehtävistä jokaisessa sarakkeessa. Yksinkertaisimmillaan taulussa on kolme saraketta, jotka kuvaavat työvaiheita: tehtävät työt, käynnissä olevat työt sekä tehdyt työt. Sarakkeet kuvaavat työprosessia ja sen eri vaiheita. Työtehtävät liikkuvat taululla työn etenemisen mukaan (s. 59).

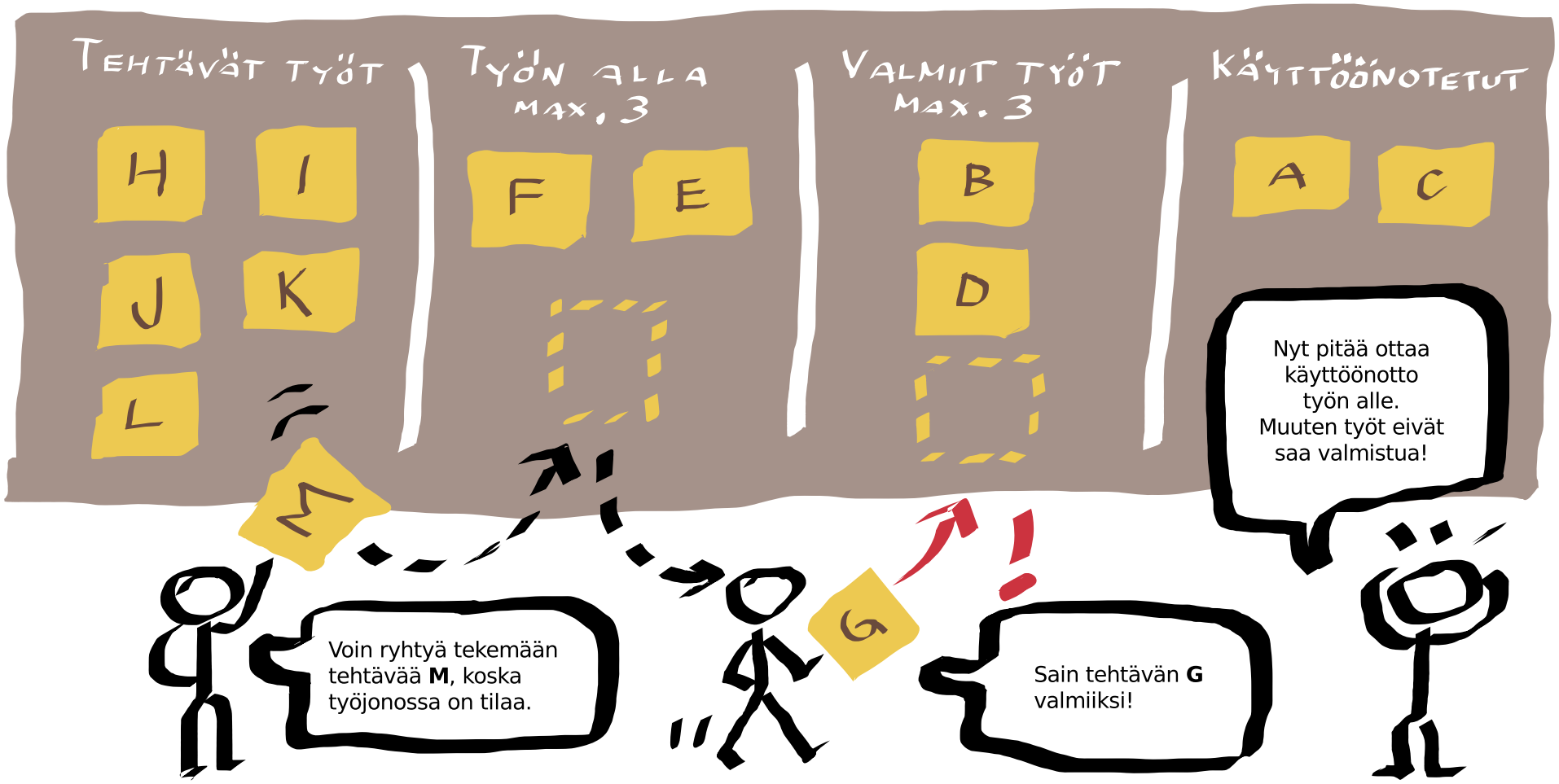
Päällekkäisten työtehtävien rajoitus erottaa Kanbanin merkittävästi muista menetelmistä. Jokaisen vaiheen samanaikaisesti käynnissä olevan työn rajoittaminen estää liikatuotannon ja paljastaa prosessin pullonkaulat. Työtehtäviä voidaan ottaa kuhunkin vaiheeseen vasta, kun tilaa on. Tämä eroaa merkittävästi perinteisestä prosessista, jossa jokaisen työvaiheen tarvittava tuotantomäärä yritetään ennustaa. Rajoitteet tuovat nopeasti esiin ongelmakohdat työvuossa, jonka jälkeen niitä on mahdollista ratkaista.

Kanbanin kolme ydinperiaatetta

1. Aloita millä pystyt.
2. Hae vähittäistä muutosta.
3. Kunnioita nykyisiä prosesseja, rooleja ja velvollisuuksia.

Lean-ohjelmistokehityksen periaatteet

1. **Poista hukka** – Älä tee ominaisuuksia varmuuden vuoksi, varmista nopea toteutus vaatimuksesta testaukseen.
2. **Rakenna laatu osaksi tuotetta** – Kirjoita toimivia testejä vaatimusten sijaan ja integroi jatkuvasti.
3. **Luo tietoa** – Opi uutta kokeilujen kautta, haasta nykyiset tavat ja mahdollista nopea reagointi muutoksiin.
4. **Lykkää sitoumusta** – Tee myöhäiset muutokset mahdollisiksi tekemällä rajoittavia päätöksiä vasta viime hetkellä.
5. **Toimita nopeasti** – Vältä listojen ja jonojen käyttöä ja tee toimivia versioita lyhyin väliajoin työstäen vain muutamia asioita kerrallaan.
6. **Kunnioita ihmisiä** – Kun tiimillä on mahdollisuus vaikuttaa omaan työhönsä, on se sitoutunut yhteisen tavoitteen saavuttamiseen. Tiimin, johdon ja asiakkaan välillä vallitsee keskinäinen kunnioitus.
7. **Optimoi kokonaisuus** – Keskity kokonaisuuteen, toimita koko tuote ja mittaa tuloksellisuutta tuotteen hyödyllisyyden ja asiakkaan tyytyväisyyden kautta.



Lisää aiheesta

Anderson D.: *Kanban – Successful Evolutionary Change for Your Technology Business*.

Poppendieck M., Poppendieck T.: *Implementing Lean Software Development: From Concept to Cash*.

Sulautettujen järjestelmien määritelmä, historia ja kehitysmenetelmät

Määritelmä

Sulautettu järjestelmä on jonkin tehtävän suorittamiseen suunniteltu laite, joka sisältää elektroniikkaa, mahdollisesti mekaniikkaa sekä laitteelle erityisesti suunniteltua ohjelmistoa. Sulautetun järjestelmän suunnittelu edellyttää eri alojen kehittäjien tiivistä yhteistyötä.

Sulautetussa järjestelmässä ohjelmallisen osuuden on tarkoitus olla sulautuneena laitteen toimintaan siten, ettei laitteen käyttäjän tarvitse välttämättä tiedostaa sen luonnetta ja olemassaoloa. Sulautetuille ohjelmistoille on usein ominaista reaaliaikainen toiminta. Käyttöliittymän tulee reagoida lähes välittömästi ja useat laitteiden toiminnot edellyttävät ennustettavaa, reaaliaikaista toimintaa. Monelle sulautetulle järjestelmälle on myös ominaista käytettävissä olevan energian rajallisuus johtuen muun muassa virtalähteenä käytettävistä akuista. Nämä piirteet aiheuttavat omat haasteensa laitteiden suunnitteluprosessille.

Sulautetun järjestelmän koko voi vaihdella hyvin pienestä ja yksinkertaisesta laitteesta, kuten sähköhammasharjasta, suuriin teollisuusrobotteihin ja vaikka lentokoneisiin. Sulautetut järjestelmät muodostavat keskeisen osan monen arkipäiväisen laitteen näkyvästä kehityksestä. Esimerkiksi suuri osa autoteollisuudessa viime vuosina esitellyistä ominaisuuksista, kuten ajonvakautusjärjestelmistä, perustuu sulautettuihin järjestelmiin.

Historia

Monet sulautetut järjestelmät ovat laitteita, joiden toteuttaminen sellaisenaan on tullut mahdolliseksi vasta tekniikan kehittymisen ja sulautetun järjestelmän käsitteen myötä. Sulautettujen järjestelmien historian alkuhetkeä on täten vaikea määritellä. Laitteet, jotka sisältävät mikrokontrollereja ja niiden avulla ohjelmoimalla toteutettuja toimintoja, voidaan nähdä sulautettujen järjestelmien varhaisina edustaijina.

Tekniikan ja sen mukana sulautettujen järjestelmien kehittymisen myötä sulautetut järjestelmät lähestyvät ominaisuuksiltaan yleiskäyttöisiä tietokonejärjestelmiä. Monet laitteet ovat uudelleen ohjelmoitavissa ja niihin voidaan jälkikäteen helposti lisätä uusia ohjelmallisia ominaisuuksia.

“Sulautettu järjestelmä: Laite, joka sisältää elektroniikkaa, mekaniikkaa sekä laitteelle erityisesti suunniteltua ohjelmistoa.”

Kehitysmenetelmät

Sulautettujen järjestelmien kehityksessä perinteinen lähestymistapa on niin sanottu vesiputousmalli (s. 2). Tyypillinen sulautetun järjestelmän kehitysprojekti alkaa järjestelmän määrittelyllä ja suunnittelulla. Järjestelmäsuunnittelussa määritellään järjestelmän laitteisto- ja ohjelmisto-osat, joita aletaan suunnitella rinnakkaisissa, toisistaan pääosin erillisissä prosesseissa. Osien valmistuttua ne integroidaan yhteen ja järjestelmä testataan kokonaisuutena.

Sulautettujen järjestelmien suunnittelussa voidaan noudattaa myös niin sanottua yhteissuunnittelua, jossa laitteistoa ja ohjelmistoa kehitetään yhdessä ja yhtä aikaa. Useissa tapauksissa osa-alueet kuitenkin muodostavat omat niin monimutkaiset ja laajat kokonaisuutensa, ettei täydellinen yhteissuunnittelu ole mahdollista.

Sulautettujen järjestelmien osa-alueiden – ohjelmisto, elektroniikka ja mekaniikka – väliset riippuvuussuhteet asettavat vaatimuksia ja rajoituksia, jotka on otettava tarkasti huomioon. Sulautettujen järjestelmien suunnittelulle tyypillistä on tilanne, jossa elektroniikka tarvitsisi ohjelmistoa ja ohjelmisto elektroniikkaa, jotta kumpaakin osa-aluetta saataisiin edistettyä ja testattua. Tästä syystä ohjelmisto- ja laitteistokehitys tekevät yleensä yhteistyötä myös erillisten kehitysprosessiensa aikana.

Lisää aiheesta

Catsoulis J.: *Designing Embedded Hardware*.

White E.: *Making Embedded Systems: Design Patterns for Great Software.*

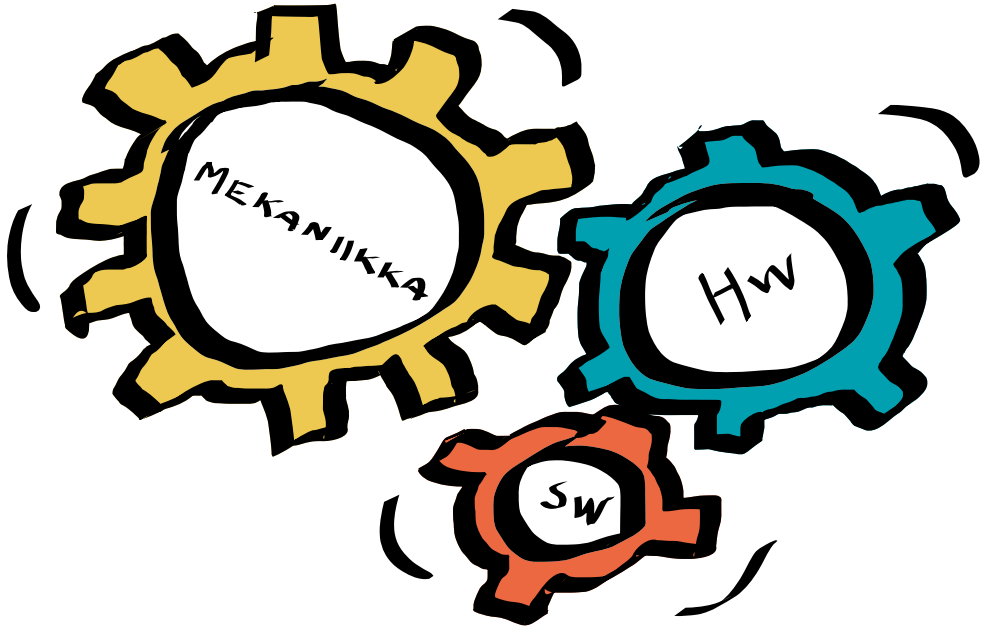
Ketteryyden haasteet ja mahdollisuudet sulautettujen järjestelmien suunnittelussa

Ketterät menetelmät on suunniteltu ensisijaisesti ohjelmistokehityksen lähtökohdista ja sen tarpeisiin. Ketterien menetelmien taustalla oleva ketteryyden filosofia on puolestaan peräisin valmistavan teollisuuden piiristä. Monet ketterälle kehitykselle ominaiset piirteet nojaavat ohjelmistokehitykselle luonteenomaiseen tekemiseen ja niistä yleisesti esitetyt esimerkit tulevat ohjelmistokehityksen maailmasta.

Ketterät menetelmät tarjoavat sulautettujen järjestelmien suunnitteluun mahdollisuuksia, joilla tuotekehitysprosessia saadaan tehostettua kehittäjien työhyvinvointia tukien. Kuitenkin on havaittavissa myös haasteita jotka pitää ratkaista ennen kuin laaja-alainen ketteryyden soveltaminen on mahdollista.

Kokonaiskuvan ymmärrys

Ketterät menetelmät painottavat tehokasta tiedonkulkua ja läpinäkyvää organisaatiota. Ne helpottavat tuotekehityksen kokonaiskuvan ja tuotekehityksessä vallitsevien riippuvuussuhteiden ymmärtämistä. Sulautettujen järjestelmien suunnittelussa eri osa-alueille erikoistuneet kehittäjät osallistuvat saman järjestelmän suunnitteluun. Perinteinen näkökulma (s. 10) jakaa järjestelmän osiin kehittäjien teknisen osa-alueen perusteella ja jokainen vastaa omasta alueestaan. Tällaisessa lähestymistavassa käy helposti niin, että kokonaiskuvan ymmärtäminen vaarantuu ja oman työn vaikutus muiden työhön unohtuu. Ketterät menetelmät ohjaavat jaettuun vastuunkantoon ja kokonaisuuden pilkkomiseen (s. 49) ennemmin toiminnallisten osien kuin teknisten osaamisalueiden perusteella. Näin eri osaamisalueiden osaajat työskentelevät yhdessä ja oppivat ymmärtämään ratkaisujensa vaikutukset kokonaisuuden kannalta.



Projektin etenemisen näkyvyys

Perinteisessä toimintaympäristössä projektin etenemisen seuranta voi olla hyvinkin haastavaa yksittäiselle kehittäjälle eikä hän välttämättä edes osaa vaatia sitä tai olla kiinnostunut siitä. Projektien etenemistä seurataan usein taloudellisesta näkökulmasta, mikä ei tyypillisesti juurikaan hyödytä kehittäjää omassa työssään. Ketterillä menetelmillä tavoitellaan parempaa ja kattavampaa näkyvyyttä koko projektin tilasta. Säännöllisillä tilannepalavereilla (s. 43) kehittäjät päivittävät toisilleen oman tilanteensa, mikä myös helpottaa aikaista tarttumista ongelmakohtiin. Hyvä näkyvyys projektin tekniseen etenemiseen helpottaa oman työn organisointia ja sujuvuutta sekä motivoi etenemään kohti yhteistä tavoitetta.

Monialainen kehitysympäristö

Sulautettujen järjestelmien suunnittelussa tuotekehitystiimeille on tyypillistä, että jokaisella tiimin jäsenellä on omat erityisosaamisalueensa. Nämä saattavat olla seurausta esimerkiksi koulutustaustasta, aiemmasta työkokemuksesta tai syvällisestä perehtyneisyydestä johonkin aihepiiriin. Ketterissä kehitysmenetelmissä on monia piirteitä, jotka tähtäävät osaamisen laajentamiseen. Kyseiset piirteet perustuvat siihen oletukseen, että kaikki tiimin jäsenet kykenevät hoitamaan tai tarvittaessa omaksumaan lähes kaikki tiimin tehtävät. Sulautettujen järjestelmien kehityksessä tiimit muodostuvat sekä ohjelmisto- että laitteistosuunnittelijoista, ja näistä vielä esimerkiksi laitteistopuoli voi jakautua elektroniikka- ja mekaniikkasuunnittelijoihin. Perehtyminen toisten tiiminjäsenten osaamisalueisiin helpottaa kokonaisuuden ja riippuvuussuhteiden ymmärtämistä. Täydellinen tehtävien kierrätys ei kuitenkaan sulautettujen järjestelmien suunnittelussa ole tarkoituksenmukaista niin kauan kuin tiimeissä on mukana hyvin erilaisille osa-alueille erikoistuneita kehittäjiä.

Suunnitelmallisuuden tarve

Ketterässä kehityksessä painotetaan suunnitelmien tarkentumista projektin edetessä. Tämä asettaa haasteita laitteistosuunnittelulle, jossa suunnitelman muuttuminen testattavaksi laitteeksi saattaa vaatia usean viikon työpanoksen. Ketterää laiteprojektia aloitettaessa on syytä kartoittaa, mitkä asiat pitää päättää missäkin vaiheessa projektia ja muodostaa näistä projektin sisäiset etapit. Etappien puitteissa voidaan edetä ketterien periaatteiden mukaisesti.

Tavoittamattomat asiakkaat

Monet sulautetut järjestelmät ovat tuotteita, joita myydään massamarkkinoille tai suunnitteluvaiheessa vielä tuntemattomille asiakkaille. Näin ollen tiivis asiakasyhteistyö tuotekehityksen aikana ei aina ole mahdollista siinä määrin kuin ketterät menetelmät siihen kannustavat. Näissä tilanteissa kehittäjien on hyvä toimia yhteistyössä myyntiorganisaation kanssa tunnistaakseen potentiaalisten ostajien tarpeet.

Vaihtelua sietämättömät ominaisuudet

Sulautetuille järjestelmille on ominaista laitteiston ja ohjelmiston kiinteä riippuvuussuhde. Tämä aiheuttaa tuotekehitystyöhön haasteita, joissa molempien osapuolien on toimittava ennakoitavasti, jotta kokonaisuus toimii halutusti. Järjestelmissä olevien rajapintojen aikainen määrittely on tärkeää, jotta rajapinnassa toimivien kokonaisuuksien suunnittelu saadaan käyntiin. Kehittäjien välisen kommunikaation ja läpinäkyvyyden merkitys on tässäkin kohdassa erittäin tärkeää, jotta pienetkin muutokset saadaan kaikkien tietoon mahdollisimman pian ja turhan työn teko voidaan minimoida.

Lisää aiheesta

Kaisti M., Rantala V., Mujunen T., Hyrynsalmi S., Könnölä K., Mäkilä T., Lehtonen T.: A*gile methods for embedded systems development - a literature review and a mapping study*.

Ketterän kehityksen arvot ja periaatteet sulautetuissa järjestelmissä

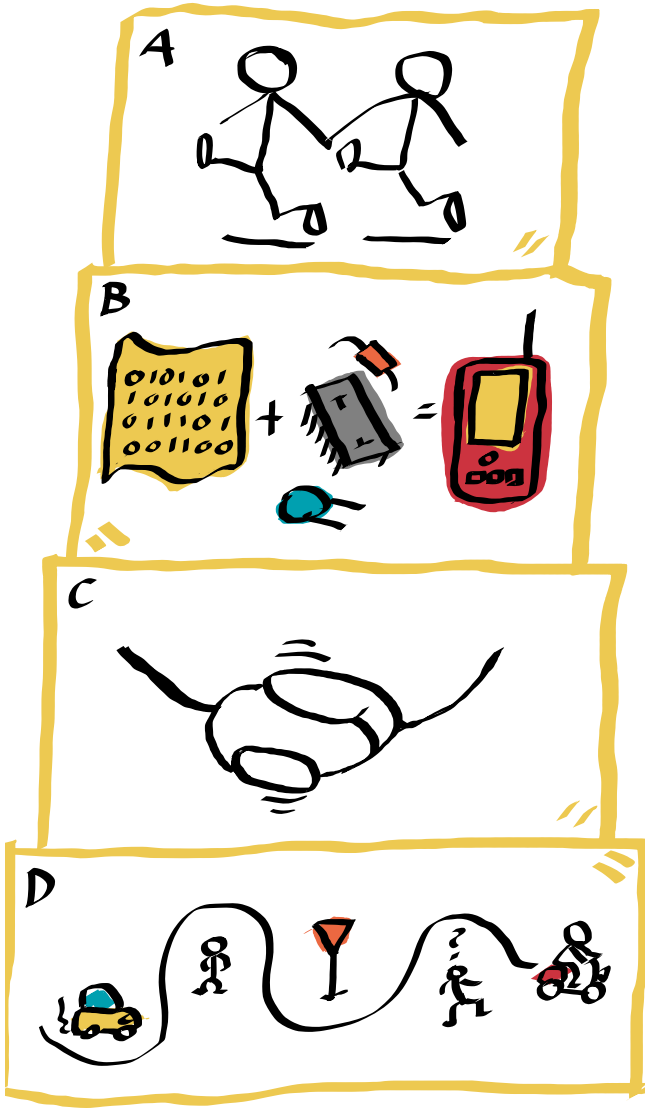
Arvot ja periaatteet luovat pohjan ketterälle kehitykselle. Muokkaamalla näitä ketterän ohjelmistokehityksen julistuksen neljää arvoa ja kahtatoista periaatetta voidaan vastata sulautettujen järjestelmien kehittämisen erikoispiirteiden haasteisiin.

A) Yksilöt ja vuorovaikutus

Sulautettujen järjestelmien kehittämisessä laitteisto, ohjelmisto ja mekaniikka muodostavat tiiviin kokonaisuuden. Koska kukin osa-alue vaikuttaa toisiinsa, kehittäjien välisen vuorovaikutuksen merkitys korostuu. Järjestelmän suunnittelussa on mukana eri alojen asiantuntijoita, jolloin ymmärrys oman työn merkityksestä muille on tärkeää. Tiiviissä vuorovaikutuksessa mahdolliset väärinymmärrykset tulevat esiin ja saadaan selvitettyä nopeasti, jolloin vältetään turhaa työtä. Ketteriin periaatteisiin kuuluvat myös luottamus yksilöihin ja tiimeihin sekä ohjeet tehokkaampaan kanssakäymiseen eri osapuolten kesken. Tavoitteena on myös kestävä työtahti koko projektin ajan, sekä tiimin säännöllinen toimintatapojen kehittäminen.

B) Toimiva tuote

Toimivaa tuotetta on sulautetuissa järjestelmissä mahdotonta toimittaa lyhyin, säännöllisin väliajoin, koska uusien laitteistojen valmistaminen on hidasta ja kallista. Toimivan tuotteen tulee kuitenkin olla se, mihin pyritään. Tuotteen sijaan iteraatioiden lopuksi voidaan esitellä erilaisia demonstraatioita ja simulaatioita tuotteesta, pyrkien joka kerta kokonaisuuteen, johon on lisätty jotain uutta edelliseen kertaan nähden. Aina kun mahdollista, esitellään kuitenkin toimiva tuote. Näin päästään konkreettisiin välitavoitteisiin. Asiakas näkee tuotteen edistymisen ja kykenee antamaan palautetta, jonka mukaan tuote kehittyy. Laadun merkitys korostuu sulautetuissa järjestelmissä – fyysistä laitetta ei kyetä päivittämään kuten pelkkää ohjelmistoa, ja usein ohjelmistonkin päivittäminen on haastavaa. Tällöin tuotteen on oltava laadukas ja toimiva heti ensimmäisestä toimitetusta versiosta alkaen.



C) Asiakasyhteistyö

Sulautettujen järjestelmien kehityksessä loppuasiakas on usein kauempana kehittäjästä verrattuna ohjelmistokehitykseen. Esimerkiksi jos kehitettävä järjestelmä on osa suurempaa järjestelmää, voi asiakas olla joku, joka käyttää tätä järjestelmän osaa. Asiakkaan tarpeiden tyydyttäminen ja muutosten hyödyntäminen asiakkaan eduksi ovat yhtä tärkeitä sulautetuissa järjestelmissä kuin ohjelmistokehityksessäkin. Asiakkaalta saatava palaute säännöllisesti esitetyistä demonstraatioista tai keskeneräisestä tuotteesta auttavat kehittämään tuotetta oikeaan suuntaan.

D) Muutokseen vastaaminen

Erityisesti laitteiston puolella suuret muutokset loppuvaiheessa projektia ovat kalliita, niin tuotantokulujen kuin muutosten aiheuttamien kerrannaisvaikutusten vuoksi. Kuitenkin sulautetun järjestelmän kehittäminen on usein pitkä prosessi ja ymmärrys halutusta tuotteesta kasvaa projektin aikana. Yksinkertaisin ratkaisu tuotekehityksen alkuvaiheissa saattaa aiheuttaa rajoituksia jatkokehityksen aikana. Tämän vuoksi päätökset pitäisi tehdä mahdollisimman myöhään ja pitää mielessä myös yleiskäyttöisyys sekä päätöksistä aiheutuva lisätyön määrä. Suunnitelman tulisi sulautetussa järjestelmässäkin olla vain niin yksityiskohtainen, kuin siinä kohtaa projektia on välttämätöntä.

Ketterän kehityksen periaatteet *sovellettuna* sulautettuihin järjestelmiin

(Alkuperäisistä periaatteista muokatut osat on esitetetty kursiivilla.)

1. Tärkein tavoitteemme on tyydyttää asiakas toimittamalla tämän tarpeet täyttäviä *demonstraatioita, jotka johtavat hyödylliseen tuotteeseen* aikaisessa vaiheessa ja säännöllisesti. (B,C)

2. *Vältämme rajoittavien päätösten tekoa salliaksemme muutokset* myös kehityksen myöhäisessä vaiheessa. *Näin muutosta voidaan hyödyntää* asiakkaan kilpailukyvyn edistämiseksi. (C,D)

3. Toimitamme *demonstraatioita, jotka johtavat hyödylliseen tuotteeseen* säännöllisesti, parin viikon tai kuukauden välein, ja suosimme lyhyempää aikaväliä. (B)

4. Liiketoiminnan edustajien ja ohjelmistokehittäjien tulee työskennellä yhdessä päivittäin koko projektin ajan. (A)

5. Rakennamme projektit motivoituneiden yksilöiden ympärille. Annamme heille puitteet ja tuen, jonka he tarvitsevat ja luotamme siihen, että he saavat työn tehtyä. (A)

6. Tehokkain ja toimivin tapa tiedon välittämiseksi kehitystiimille ja tiimin jäsenten kesken on kasvokkain käytävä keskustelu. (A)

7. *Demonstraatiot ja toimiva tuote* ovat edistymisen ensisijainen mittari. (B)

8. Ketterät menetelmät kannustavat kestävään toimintatapaan. Hankkeen omistajien, kehittäjien ja käyttäjien tulisi pystyä ylläpitämään työtahtinsa hamaan tulevaisuuteen. (A)

9. Teknisen laadun ja *tuotteen* hyvän rakenteen jatkuva huomiointi edesauttaa ketteryyttä. (B)

10. *Yksinkertaisuus ja yleiskäyttöisyys* – tekemättä jätettävän työn maksimointi *samalla minimoiden tehtävä työ* – on oleellista. (D)

11. Parhaat arkkitehtuurit, vaatimukset ja suunnitelmat syntyvät itseorganisoituvissa ja *tiivistä yhteistyötä tekevissä* tiimeissä. (A)

12. Tiimi tarkastelee säännöllisesti, kuinka parantaa tehokkuuttaan, ja mukauttaa toimintaansa sen mukaisesti. (A)

Lisää aiheesta

Kaisti M., Mujunen T., Mäkilä T., Rantala V., Lehtonen T.: *Agile Principles in the Embedded System Development*.

Ketterän kehityksen työhyvinvointilupaukset

Työhyvinvointi tarkoittaa, että työ on mielekästä ja sujuvaa turvallisessa, terveyttä edistävässä sekä työuraa tukevassa työympäristössä. Organisaation kestävä menestys edellyttää työhyvinvointiin panostamista sekä strategiassa että käytännön toiminnassa. Työhyvinvoinnin kehittämiseen tulee kiinnittää huomiota työmenetelmien jatkuvan parantamisen yhteydessä. Oikein sovellettuna ketterän kehityksen periaatteet ja toimintakäytännöt ylläpitävät ja edistävät työhyvinvointia parantaessaan ja sujuvoittaessaan työtä.

“Työhyvinvoinnin kannalta ketteryydessä keskeistä on ymmärrettävä, mielekäs ja hallittava työ.”

Mitä työhyvinvointi on?

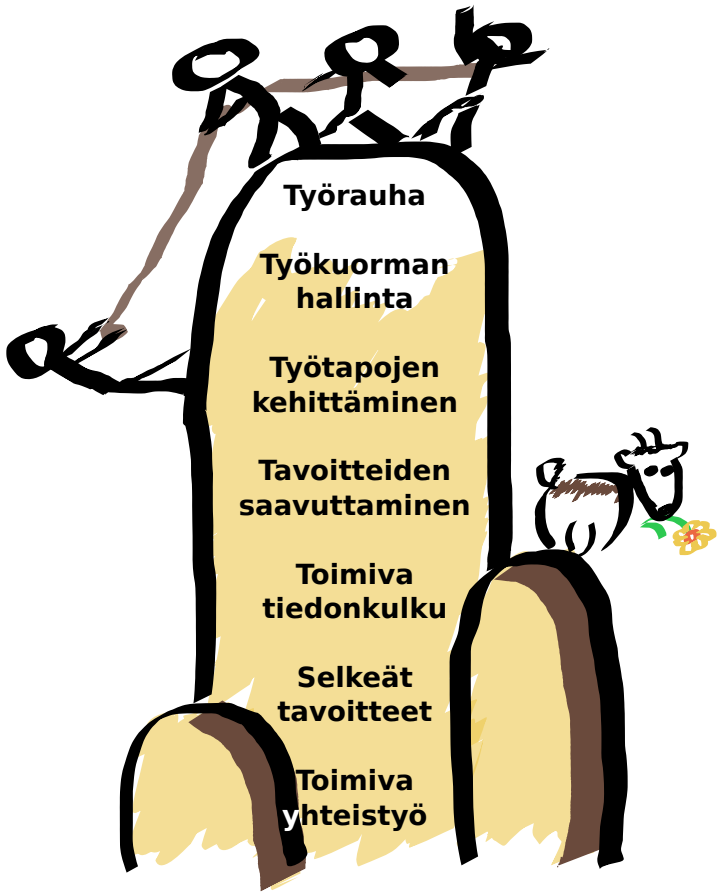
Henkilöstöön, työympäristöön, työyhteisöön, työprosesseihin ja johtamiseen liittyvät tekijät vaikuttavat monin tavoin työhyvinvointiin. Työ, työyhteisö ja organisaatio koetaan hyvänä, kun viestintä ja tiedonkulku on toimivaa, ilmapiiri on avoin ja asiallinen sekä työn sisältöön, suorittamiseen ja olosuhteisiin pystyy vaikuttamaan. Myös työtovereiden, esimiehen ja organisaation antama tuki on tärkeää hyvinvoinnin ja työn sujumisen kannalta.

Työ koetaan mielekkääksi, kun sillä on selkeät tavoitteet, työhön kohdistuvat odotukset ovat tiedossa, työn liittyminen organisaation perustehtävään ymmärretään ja työtä arvostetaan.

Johtamisen laadulla on merkitystä. Sen toimivuus ja oikeudenmukaisuus syntyvät muun muassa oikein annetusta palautteesta. Työhyvinvoinnin ylläpitäminen ja edistäminen on tietoista, jatkuvaa ja pitkäjänteistä toimintaa arjen työn sujuvoittamiseksi osana organisaation muuta toimintaa. Toimivat käytännöt motivoivat ja kannustavat työntekijöitä. Ne toimivat hyvinvointia ylläpitävänä puskurina työstä nousevaa kuormitusta vastaan.

Työhyvinvoinnin vaikutukset

Työhyvinvoinnin edistäminen organisaatiossa on kannattava investointi. Toimenpiteiden välittömiä vaikutuksia ovat sairaspoissaolojen, työkyvyttömyyseläkkeiden ja Kela-maksujen väheneminen. Välillisiä vaikutuksia puolestaan ovat työajan käytön tehostuminen ja tuottavuuden lisääntyminen. Työn laadun on todettu paranevan ja palvelu- sekä prosessi-innovaatioiden lisääntyvän. Hyvinvoiva ja motivoitunut henkilöstö on pettämätön resurssi kovenevassa kilpailussa.



Miten ketteryys parantaa työhyvinvointia?

Työn mielekkyys kasvaa

Ketterien periaatteiden mukaan projektit rakennetaan motivoituneista yksilöistä, joille tarjotaan riittävä taustatuki ja toimintavapaus. Tiimin sisällä voidaan valita, kuka jäsenistä tekee yksittäisiä tehtäviä eli jäsenet saavat *hyödyntää vahvuuksiaan* ja *opetella uutta*.

Suuren projektin jakaminen osiin ja suoritettavien tehtävien valitseminen yhdessä *selkeyttävät työn tavoitteita* ja työntekijään kohdistuvia odotuksia. Priorisoinnin kautta työ saadaan mielekkääksi, isoon *kokonaisuuteen* ja sen eteenpäin viemiseen *liittyväksi*. Kehittämisessä pyritään tuottamaan toimivia tuotteen osia mahdollisimman usein ja lyhyellä aikavälillä. Tällainen lyhyen aikavälin *tavoitteiden saavuttaminen* lisää työn mielekkyyttä ja hallittavuutta.

Ketteriin menetelmiin kuuluu *yhteistyön tekeminen* tavoitteiden saavuttamiseksi, avun ja tuen tarjoaminen sekä aloitteellisuuteen rohkaiseminen. Työn mielekkyyttä lisää myös se, että *asiakkaan tyytyväiseksi tekeminen* on toiminnan johtava periaate.

Työssä kuormittuminen saadaan hallintaan

Ketterissä periaatteissa pyritään yksinkertaisuuteen – tekemättä jätetyn työn maksimointiin. Säännöllisen priorisoinnin kautta *työkuormaa hallitaan* ja *työmäärän arviointi helpottuu*. Mahdollisuus vaikuttaa työtehtävien määrittelyyn auttaa hallitsemaan myös työtehtävien vaatimusten ja työntekijän osaamisen tasapainottamisessa.

Työn pilkkominen, tehtävien priorisointi, vaikutusmahdollisuudet, tehdyn työn arviointi ja uudelleen suuntautuminen riittävän lyhyen iteraation jälkeen auttavat toteuttamaan ketterien periaatteiden kestävää kehitystä. *Työrytmi tasaantuu* ja intensiivijaksot korvautuvat terveellä päivittäisellä paineella: Pääsääntöisesti pyritään seuraamaan normaalia työviikkoa ja ylitöiden sijasta karsimaan ominaisuuksia.

Lisää aiheesta

Aura O., Ahonen G., Ilmarinen J.: *Strategisen hyvinvoinnin tila Suomessa 2011.*

Janhonen M.: *Tasapainoinen tiedon jakaminen ja tiimityön laatu*.

Lindström K., Leppänen A. (toim.): T*yöyhteisön terveys ja hyvinvointi*.

Rauramo P.: *Työhyvinvoinnin portaat – Viisi vaikuttavaa askelta*.

Työ ja prosessit muuttuvat sujuvammiksi

Työn *keskeytysten hallinta* edistää sujuvuutta. Ketterissä menetelmissä organisoidaan työhön systemaattisesti jaksoja, jolloin kehittäjillä on *työrauha* ja he voivat keskittyä käsillä olevaan työhön. Asiakas ei voi pyytää tekemään ylimääräistä iteraation aikana.

Työntekijöiden itseohjautuvuus ja *tiedonkulun tehostuminen* sujuvoittavat niin ikään työtä. Parhaat lopputulokset saavutetaan, kun tiimit ovat itseorganisoituvia ja liiketoiminnan edustajat ja tuotteen kehittäjät työskentelevät yhdessä päivittäin. Tehokkaimpana tiedonvälityksen tapana on kasvotusten tapahtuva kommunikaatio. Osallisuus ja jaettu johtajuus toteutuvat esimerkiksi siinä, että koko tiimi osallistuu asiakastilanteisiin.

Ketterissä periaatteissa painotetaan *jatkuvaa huomiota korkeaan laatuun*. Tiimin jäsenet miettivät tasaisin väliajoin kuinka tiimi voisi *kehittyä sisäisesti* ja toimia paremmin. Tavoitteena ovat työn tehostuminen ja selkeytyminen. Tiimit tunnistavat turhat työkäytännöt, jotta niistä voidaan luopua. *Uskallus tarttua epäkohtiin* ja *uudet innovaatiot* lisäävät työhyvinvointia!

Sulautettujen järjestelmien ketterät kehitysmenetelmät

Ketteryyteen valmistautuminen

Ketteryyden käyttöönotto kannattaa aloittaa tutustumalla ketteryyden ajatuksiin ja erilaisiin ketteriin kehitysmenetelmiin sekä selvittämällä, minkälaisena kehittäjät kokevat tämänhetkisen tuotekehitysprosessin. Ketteryyden käyttöönottoa kannattaa pohtia myös siltä kannalta, mitä hyötyä ketteryydestä kyseiselle tiimille tai projektille voisi olla ja miksi nykyistä kehitysprosessia halutaan muokata. Koska ketteryys lähtee tiimistä itsestään, tiimi otetaan alusta asti mukaan suunnittelemaan ketteriä toimenpiteitä.

Nykytilanteen kartoitus

Ennen ketteryyden käyttöönottoa on hyvä kartoittaa yrityksen tuotekehitysprosessin nykytilanne. Kartoitus voidaan aloittaa käymällä mahdollista dokumentaatiota läpi. Koska dokumentaatio ei aina vastaa todellisuutta, prosessin todellinen vaikutus käytännön työn tekemiseen pyritään saamaan selville esimerkiksi haastatteluin ja kyselyin. Tärkeää on selvittää kokemuksia työnteosta kaikilla tasoilla johtoportaasta kehittäjiin.

Kartoituksen avulla täsmennetään nykyiset haasteet ja hyvin toimivat tavat päivittäisessä työnteossa. Hyvin toimivia tapoja ei kannata hylätä, vaan pohtia miten ne saataisiin sovitettua osaksi ketterämpiä toimintatapoja.

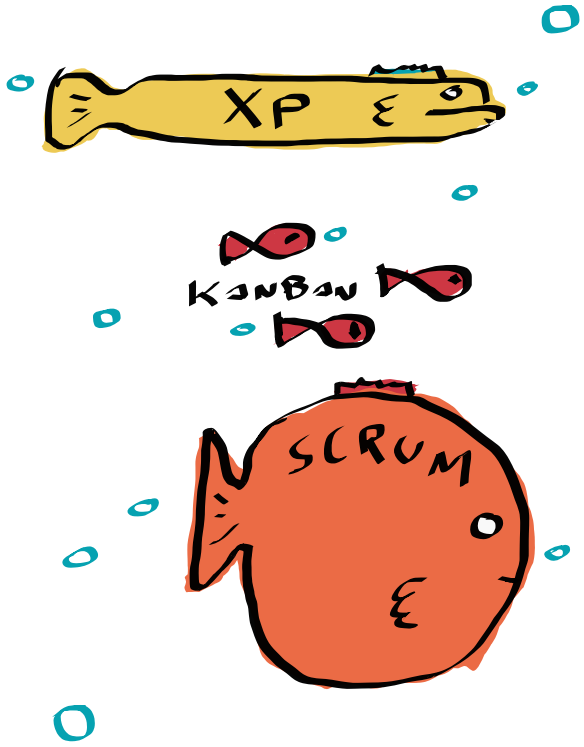
Miksi tuotekehitysprosessia halutaan muuttaa?

Kun ketteryyttä halutaan ottaa käyttöön, syynä ovat usein haasteet nykyisessä tuotekehityksessä. Esimerkiksi vaikka projekteille on määritelty tarkat aikataulut, projektit voivat olla kutakuinkin aina myöhässä. Joissakin projekteissa on saatettu projektin loputtua huomata, että toteutetut asiat eivät olekaan niitä, mitä asiakkaat kaipaisivat. Myöhästyminen tai epäoleellisten asioiden toteutus on saattanut aiheuttaa jopa projektien perumisia.

Jotta tuotekehitysprosessin muuttaminen onnistuisi, nykyistä tilannetta pitää haluta parantaa. Motiiveja tulee pohtia paitsi yrityksen johdon, myös yksittäisten työntekijöiden kannalta. Ketteryys ei välttämättä sovi kaikkiin organisaatioihin.

Ketteryyden ymmärtäminen

Ketteriä menetelmiä on useita, esimerkkeinä tässä käsikirjassa esitellyt Scrum, XP ja Kanban. Kaikkiin tilanteisiin pelkästään käytännöt eivät tarjoa ratkaisuja, joten ketteryyden arvojen ja periaatteiden ymmärtäminen on myös tärkeää.



Ketteryyteen kannattaa tutustua monipuolisesti esimerkiksi kirjallisuuden ja internetin avulla. Alan lukuisat konferenssit ja koulutukset ovat yksi väylä ketteryyteen tutustumiseen. Myös asiantuntevia konsultteja voidaan käyttää ketteryyden käyttöönoton apuna.

Käyttöönotettavien ketterien menetelmien valitseminen

Kun lähtötilanne tunnetaan, valitaan mitä ketteriä menetelmiä tai käytäntöjä otetaan käyttöön. Menetelmistä ja käytännöistä päättää viime kädessä tiimi itse. Käyttöönotettaviin menetelmiin vaikuttaa kartoituksessa löydettyjen asioiden lisäksi myös projektityön luonne. Esimerkiksi turvallisuuskriittisissä järjestelmissä on otettava huomioon jäljitettävyys.

Valitut menetelmät ja käytännöt toimivat vain lähtökohtana. Aluksi voi olla parempi omaksua jokin prosessi melko kurinalaisesti ja oppikirjan mukaan, joskin tiimille sopivaksi muokaten. Esimerkiksi erilaiset palaverikäytännöt ovat helppoja omaksua alkuvaiheessa. On kuitenkin tärkeää ymmärtää myös niiden taustalla oleva filosofia ja periaatteet, jotta ne eivät jää tyhjiksi seremonioiksi, joiden välillä palataan vanhaan toimintatapaan. Ketteryyden omaksuttuaan tiimi voi soveltaa sitä yhä enemmän itselleen sopivaksi. Tiimin tulee tarkastella koko ajan työskentelytapojaan ja muuttaa niitä toimivammiksi.

Ketteryyden käyttöönotto

Tiimin vastuulla on käytössä olevien menetelmien tehokas käyttäminen sekä niiden kehittäminen parantamaan tiimin työskentelyä. Tiimi tarvitsee kuitenkin tukea johdolta ja ymmärrystä niin asiakkaalta kuin muultakin organisaatiolta. Vaikka uusia toimintatapoja voidaan ottaa käyttöön nopeastikin, onnistunut muutos toimintakulttuurissa ei tapahdu hetkessä.

Tiimi toteuttamassa muutosta

Ketteriä menetelmiä käyttöönotettaessa keskeisessä asemassa on tiimin jäsenten aktiivinen osallistuminen muutoksen suunnitteluun ja toteuttamiseen. Tiimi tuntee parhaiten oman työnsä ja on näin asiantuntija myös muutoksen suunnittelussa. Yhteisen suunnittelun kautta kaikki ymmärtävät, miksi työtapoja muutetaan ja mitä hyötyä muutoksilla tavoitellaan. Näin tiimi sitoutuu muutoksen toteuttamiseen, jolloin muutoksella on edellytykset onnistua.

Ketteriä käytäntöjä sovellettaessa muutos tulee omakohtaiseksi, kun tiimi ja sen jäsenet muokkaavat käytäntöjen toteuttamista parhaimmaksi näkemällään tavalla. Esimerkiksi tilannepalaverien tiheyttä kannattaa miettiä: kuinka tiheästi käydyt keskustelut tuottavat tarvittavaa uutta tietoa ja käyttääkö tiimi työaikaansa pelkästään tähän projektiin, vai kuluuko sitä myös muihin projekteihin tai ylläpitotyöhön. Käytäntöjä mukauttaessa otetaan huomioon niistä saatava hyöty, kuten toimiva tiedon jako, ja niiden toteuttamisen kulut, kuten palavereihin kuluva aika.

Tukea muutokselle

Muutoksia tukemaan voidaan tiimin sisältä valita fasilitoija (s. 52), jonka tehtävänä on huolehtia, että sovituista asioista pidetään kiinni. Fasilitoijan ei tarvitse olla tiiminvetäjä, sillä yhtenä tiimiläisenä hän pystyy tuomaan ketteryyttä tiimiin sisältä päin. Tärkeintä fasilitoijalle on, että hän suhtautuu positiivisesti ketteryyden käyttöönottoon.

Johdon ja esimiesten tehtävänä on ohjata muutosta organisaation toiminnan kannalta mielekkääseen suuntaan ja mahdollistaa muutos osoittamalla sille tarvittavat resurssit sekä tukensa muutosta organisoiville. Johdon ja esimiesten tehtävänä on lisäksi yhdistää organisaation erilliset muutosprosessit ja muutokset hallittavaksi ja toimivaksi kokonaisuudeksi.

Ilman johdon tukea tiimi kokee olevansa yksin ja turhautuu. Organisaation sisällä tiimin uusia työskentelytapoja ei arvosteta ja tiimin työ vaikeutuu. Tiimillä on haasteita vakuuttaa asiakasta työskentelytapojensa toimivuudesta, jos tiimi ei ole varma niiden jatkuvuudesta. Jos ketteryyttä halutaan toteuttaa täyspainoisesti, myös muun organisaation ja ulkoisen asiakkaan ymmärrys ketterästä toimintatavasta on tärkeää.

Muutosvasteen nopea saavuttaminen

Muutoksen hyödyllisyyden kokemista edistävät 1) muutoksen vaikutusten nopea arviointi ja 2) välittömät työn sujuvuutta sekä tavoitetta parantavat korjaustoimenpiteet. Uusien käytäntöjen soveltamista kannattaakin seurata yhdessä tiimin sekä muiden asianosasten kanssa. Mitkä käytännöt ovat sellaisia, jotka jäävät helposti toteutumatta? Mikä on syy tämän takana – onko käytäntö vain aluksi haastava käyttää vai eikö se sovellu lainkaan työympäristöön? Jos käytäntö seurannan jälkeen todetaan toimimattomaksi, se voidaan yhdessä sopimalla muuttaa tai poistaa.

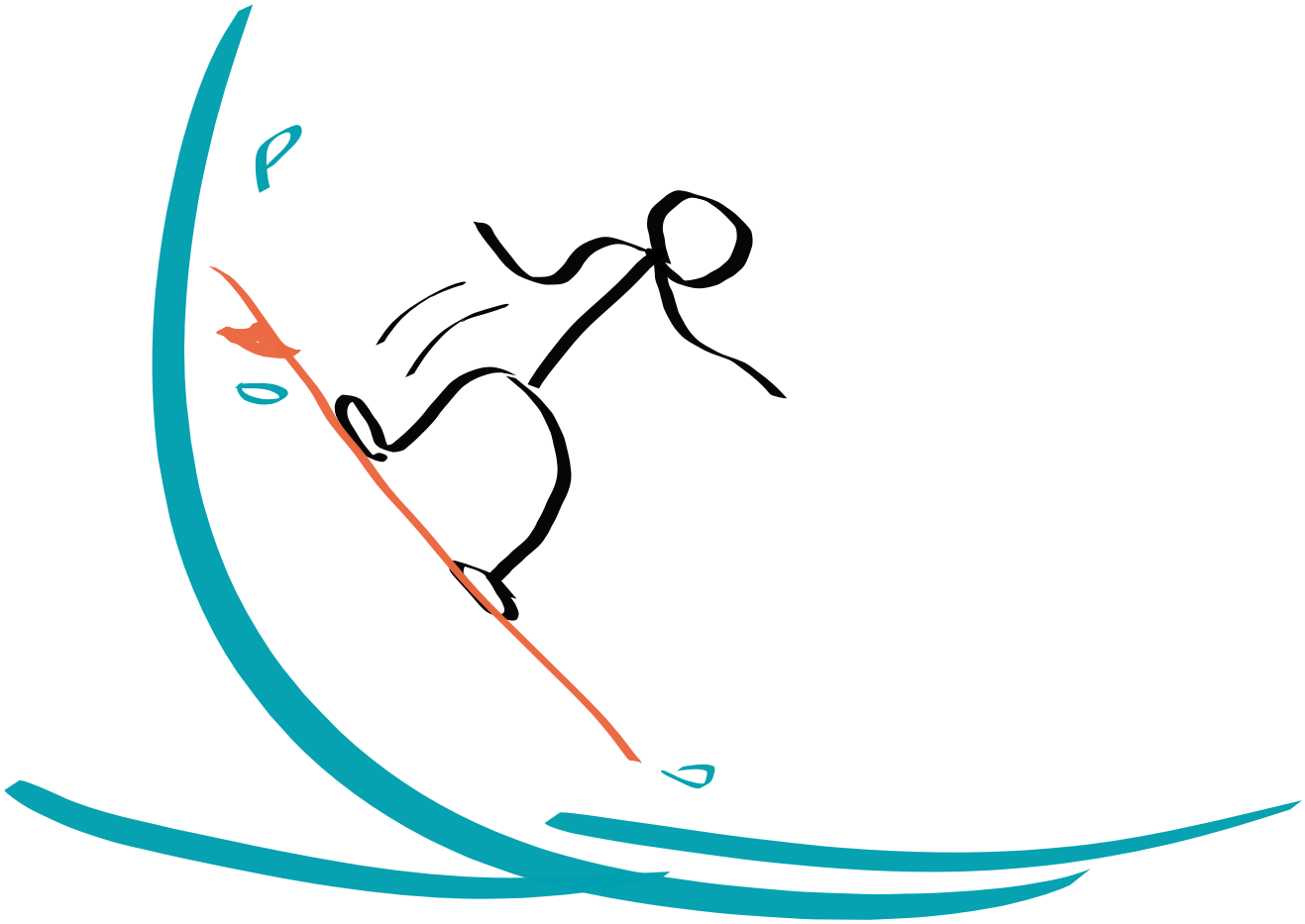
Uuden toimintakulttuurin ja ajattelutavan luominen

Uuden toimintakulttuurin ja ajattelutavan oppiminen on pitkäaikainen ja tinkimättömyyttä vaativa prosessi. Se saattaa viedä aikaa puolesta vuodesta pariin vuoteen. Sovittujen käytäntöjen vakiinnuttua on aika miettiä, miten voidaan edelleen kehittää ketterää toimintatapaa ja edistää uutta ajattelutapaa. Voidaan

miettiä esimerkiksi, löytyisikö uusia käytäntöjä ja työkaluja vielä entisestään parantamaan työskentelytapoja. Ketteryydelle ominaista onkin jatkuva kehittyminen ja mukautuminen muuttuviin olosuhteisiin.

Uuden toimintakulttuurin rakentamiseen tarvitaan kannustusta ja tukea. Kun tiimiin luotetaan, se kokeilee ja muuttaa toimintatapojaan rohkeasti. Kun tiimillä on valtaa päättää toimintatavoistaan, sillä on mahdollisuus oppia myös yrityksen ja erehdyksen kautta. Myös erehdykset on sallittava ja niitä on käsiteltävä syyllistämättä ketään. Toimintatavat ovat huonoja, eivät ihmiset.

Ketteryyden arvoihin ja periaatteisiin pohjaava ajattelutapa toimii oppimisen ohjenuorana. Joskus tiimille tulee vastaan tilanteita, joihin ratkaisua ei löydy suoraan mistään olemassa olevasta käytännöstä. Tällöin tiimin tulee pohjata ketteryyden arvoihin ja ratkaista tilanne näiden arvojen pohjalta.



Työhyvinvoinnin edistäminen muutoksessa

Työhyvinvoinnin ja ketteryyden käyttöönoton edistämisessä keskeistä on luoda yhteistä ymmärrystä työstä ja työyhteisön toiminnasta. Uudet innovaatiot ovat työhyvinvoinnin kannalta sekä mahdollisuus että uhka. Usein ne tuovat työhön mielekkyyttä ja sujuvoittavat sitä. Toisaalta vanhaa toimintatapaa on vaikea muuttaa, joten ketteriä toimintatapoja toteutettaessa on tärkeää varmistaa myös työhyvinvoinnin toteutuminen.

Työn kehittämisellä työhyvinvointia

Työn sujuvuus syntyy työn tavoitteiden yhteisestä ymmärtämisestä, yhteistyöstä, työnjaon sekä työvälineiden toimivuudesta. Työn muutokset aiheuttavat häiriöitä ja jännitteitä sekä työn sujumattomuutta. Syitä työn sujumattomuuteen ovat uusien toimintatapojen oppimisen hitaus, vanhoista luopumisen vaikeus sekä uusien ja vanhojen toimintatapojen sekoittuminen keskenään.

Häiriöt aiheuttavat turhautumista ja kuormittumista. Onnistuminen itselle tärkeällä työn osa-alueella tuottaa hyvinvointia, kun taas epäonnistumista seuraa kuormitus, vaikka työmäärä olisi kohtuullinen. Muutokset tuovat usein aluksi mukanaan epätasapainoa, kaaosta ja hallitsemattomuuden tunnetta.

Työhyvinvoinnin edistäminen on työn kehittämistä siten, että häiriöt työssä vähenevät. Työn sujuvuus ja työhyvinvointi paranevat, kun kehitetään yhdessä uusia sujuvia toimintamalleja korvaamaan vanhoja käytäntöjä. Työn kehittämisen pohjaksi tarvitaan häiriöiden synnyn selvittämistä ja ymmärtämistä sekä tulevien kehitysmahdollisuuksien hahmottamista.

Työntekijät, esimiehet ja johto kehittävät työtä ja työyhteisön toimintaa tarkastelemalla työtä. He luovat yhteistä ymmärrystä erityisesti työn muuttuvasta kohteesta eli asiakkaasta ja tämän tarpeista.

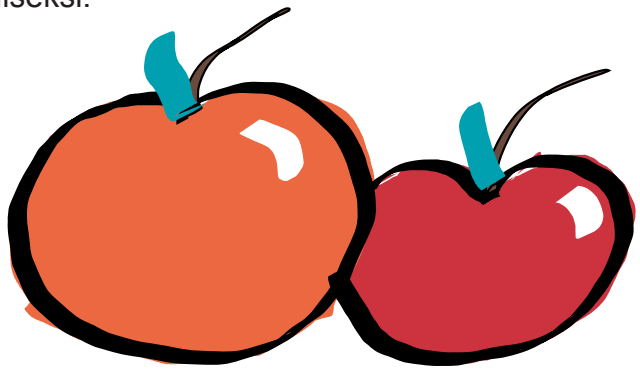
Työntekijöiden ja johdon työhyvinvointia edistää parhaiten se, että heillä on mahdollisuus osallistua oman työnsä ja organisaation toiminnan kehittämiseen, oppia yhdessä.

Työn kehityshaasteiden analysoimiseksi tarvitaan yhteisiä välineitä työprosessista saatavan tiedon jäsentämiseksi. Yhteinen kehittäminen edellyttää myös perinteisten hierarkkisten rajojen murtamista. Toimivilla johtamiskäytännöillä ja -työkaluilla vältetään ennustamattomuuden ja kompleksisuuden aiheuttamaa kaaosta.

Muutoksen vastustaminen kuuluu muutokseen

Muutoksen vastustaminen ilmenee muutoksen ja sitä ajavien kritisointina, informaation torjumisena, välttelynä ja vähättelynä. Muutokseen voidaan suhtautua välinpitämättömästi tai hyökkäävästi vanhaa puolustaen ja itseä suojellen. Muutosvastarinta estää uuden oppimista. Yleensä lähes puolet henkilöstöstä kuuluu muutoksen hitaisiin omaksujiin, joilla on kriittinen asenne muutosta kohtaan.

Muutosjohtaminen on suunniteltua, järjestelmällistä ja hallittua muutoksen läpivientiä. Muutoksen tekeminen yhdessä vaatii aikaa ja muutoksen herättämien ajatusten sekä tunteiden käsittelyä. Ketteryyteen kuuluvat osallisuus ja vaikutusmahdollisuudet tekevät muutoksen omakohtaiseksi. Tällöin uusien käytäntöjen ja ajattelutavan omaksuminen on helppoa ja jatkuva kehittäminen tulee mahdolliseksi.



Työhyvinvoinnin haasteet ketteryyttä kehitettäessä

Pyrittäessä tiimityössä moniosaamiseen ja osaamisen jakamiseen luontaiset osaamiserot voivat johtaa osaajien ylikuormittumiseen. Individualismi ja erikoistuminen voivat olla haaste tiimin tavoitteiden saavuttamisen ja moniosaamisen soveltamisen kannalta. Yksilö voi kokea autonomiansa ja asemansa uhatuksi. Lisäksi tiimin sisäisen kommunikaation, vuorovaikutuksen sekä tiedonkulun edistäminen on haasteellista.

Jatkuva asiakastoiveisiin reagointi ja tavoitteiden muokkaaminen altistavat kuormittumiselle sekä tuottavat sekaannusta työnjakoon, rooleihin ja tavoitteisiin. Joustavuuden ja jatkuvan muutoksen vaatimus voi heijastua asiantuntijan työhön epävarmuutena häiriten hallinnan tunnetta.

Tiimin autonomia voi tulla uhatuksi, jos asiakkaat vaikuttavat tiimin tekemisen yksityiskohtiin. Huomiota onkin kiinnitettävä siihen, ettei iteraatioon tule liikaa lisätyötä, kuten lisätoiveita johdolta tai odottamattomia virheidenkorjauksia, jotka haastavat tiimin yhteishengen. Oman työn arvostus kärsii, jos havaitsee tehneensä ns. turhaa työtä.

Ketterien menetelmien toteuttaminen vaatii asiantuntijoilta itseorganisoituvuutta ja itsensä johtamista. Osaamisen puutteet näillä alueilla voivat johtaa työssä kuormittumiseen. Jaetussa johtajuudessa epävirallinen vallankäyttö ja tunteet voivat vaikeuttaa asiakaskeskeistä tulosjohtamista.

Johdon liiallinen puuttuminen ja epäselvät odotukset voivat vaikeuttaa ketterän toimintatavan käyttöönottoa. Ketterän muutoksen aikaansaamisen haasteiksi on tunnistettu myös johdon tuen puute sekä huoli johdon kontrollin, suunnittelun ja dokumentaation häviämisestä sekä perinteinen organisaatiokulttuuri ja -rajat.

Ketterät menetelmät muutosjohtamisen periaatteiden mukaan toteutettuna antavat mahdollisuuden vastata näihin haasteisiin.

Lisää aiheesta

Launis K., Schaupp M., Koli A., Rauas-Huuhtanen S.: *Muutospajaohjaajan opas*.

Multanen L., Bredenberg K., Koskensalmi S., Lauttio L.-M., Pahkin K.: *Parempi työyhteisö – avaimia kehittämiseen*.

Schaupp M., Koli A., Kurki A.-L., Ala-Laurinaho A.: *Yhteinen muutos – työhyvinvointia työtä kehittämällä*.

Työhyvinvoinnin varmistaminen ketteryyden käyttöönotossa

1. Muutos toteutetaan yhdessä. Muodostakaa yhteinen käsitys muutoksen tarpeesta, tavoitteista ja toteuttamisesta.
2. Tarkkailkaa tiimin jäsenten kuormittumista ja puuttukaa siihen.
3. Keskustelkaa tavoitteet, työnjako ja roolit selviksi aina, kun työssä ilmenee häiriöitä.
4. Tunnistakaa osaamistarpeet ja huolehtikaa riittävästä osaamisesta.
5. Muistakaa avoin ja asiallinen kommunikaatio.
6. Kritiikki ja arviointi kuuluvat kehittämiseen, kuten positiivinen palautekin.
7. Ottakaa johto mukaan ongelmien ratkaisemiseen.
8. Antakaa tiimille työrauha tavoitteen saavuttamiseksi.
9. Tukekaa tiimin jäseniä vastuunotossa ja oma-aloitteisuudessa.

Työkalut kehitysjonon hallintaan

Fyysiset työkalut ketteryyden keskiössä

Vaikka ketteryydessä painotetaan yksilöiden ja kanssakäymisen tärkeyttä menetelmiin ja työkaluihin verrattuna, työkaluratkaisuilla voi olla avainasema kehitysprosessin sujuvuudessa. Ketterän kehityksen näkökulmasta työkalujen tulisi olla mahdollisimman yksinkertaisia ja tarkoituksenmukaisia sekä tukea suoraan yksilöiden välistä kommunikaatiota.

Varsinaisten kehitystyökalujen lisäksi käytännön tuotekehityksessä tarvitaan aina jonkinlaisia projektinhallintaan liittyviä työkaluja. Ketteryyden ytimessä olevat projektinhallintatehtävät on pyritty minimoimaan ja vastuu niistä jakamaan koko tiimille. Ensisijaisesti ketterässä kehityksessä hallintatyökaluja tarvitaankin kehitystyötehtävien hallintaan. Yksinkertaisin ratkaisu tämän kaltaisiin tehtäviin on ottaa käyttöön jokin fyysinen työkalu. Kynän ja paperin voimaa ei väheksytä ketterissä menetelmissä, esimerkiksi Kanban-taulu (s. 8) ja iteraation kehitysjono (s. 45) voidaan hyvin koostaa edullisesti ja vaivattomasti käyttäen esimerkiksi kynää ja paperia (s. 59). Ketteryyden oletuksena on, että tiimi työskentelee samassa tilassa, jolloin tehtävätaulu on aina käden ulottuvilla ja toimii tiimin luontaisena kokoontumispaikkana.

Ohjelmistoratkaisut

Kehityshistorian jäljitettävyys on usein tärkeää sulautettuja järjestelmiä kehitettäessä, erityisesti terveydenhuoltoon tai muihin turvallisuuskriittisiin sovelluksiin liittyvissä projekteissa. Lisäksi kehitys on tyypillisesti jakautunut useammalle tiimille, jotka koostuvat taustoiltaan erilaisista ihmisistä ja voivat olla hajautettuja. Tällöin laajaa kehitysjonoa täytyy hallita jonkin ohjelmistotyökalun voimin kynän ja paperin sijaan.

Erilaisia kehitysjonon ja muun ketterän prosessin hallintaan suunniteltuja ohjelmistoja on tarjolla paljon – ilmaisista kalliisiin, ja kevyistä työkaluista hyvinkin laajoihin järjestelmiin.

Tyypillistä on, että sulautettujen järjestelmien kehitystä ei ole työkaluissa otettu huomioon: ketteryys mielletään edelleen pitkälti ohjelmistokehityksen piirteeksi. Vastaavasti myös käytössä oleva kehitysprosessi voi olla hyvinkin räätälöity, jolloin tiettyyn menetelmään pohjautuvien työkalujen hyödyntäminen voi olla kankeaa.

Monet sulautetun kontekstin vaatimukset, esimerkiksi useiden projektien ja tuotteiden yhtäaikaisuus, ovat yleisiä myös ohjelmistokehitysprojekteissa. Nämä ovat kuitenkin vieläkin tavallisempia sulautettuja järjestelmiä kehitettäessä. Varsinaisia erityisvaatimuksia ovat muun muassa ohjelmisto- ja laitteistokehityksen toisistaan riippuvien työtehtävien hallinta käyttäjäystävällisellä tavalla – tämä saattaa vaatia työkalusta riippuen joustavuutta tai luovuutta.

Ketteryyttä omaksuttaessa on järkevää aloittaa mahdollisimman vaivattomilla ja saatavilla olevilla työkaluratkaisuilla (esimerkiksi taulukkolaskentaohjelmilla tai virheidenhallintajärjestelmän liitännäisillä) ja vaihtaa myöhemmin erilliseen järjestelmään työkalutarpeiden selkiytyessä.

Lisää aiheesta

Suomi S.: *Project Management Tools in Agile Embedded Systems Development*.

* Vuorovaikutuksellisuus puutteellista verrattuna fyysisiin työkaluihin

Fyysiset työkalut

Esim. kynä, paperi, tehtävätaulu, tarinakortti

+ Halpoja hankkia, helppoja käyttää, herättävät keskustelua

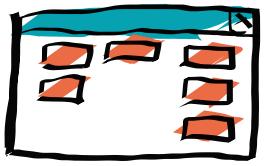
- Etäkäyttö ja historian seuranta haastavaa

Taulukkolaskimet

Esim. Excel, OpenOffice

+ Valmiiksi käytössä, runsaasti valmiita pohjia kehitysjonon hallitsemiseen, helppo räätälöityvyys

- Manuaalisen työn määrä, *

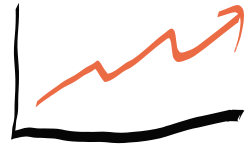


Kehitysjonon ja päivittäisen ketterän työn hallintaohjelmistot

Esim. ScrumWise, TinyPM

+ Käytön helppous, paljon valinnanvaraa

- Muokattavuus, on-site-lisenssit harvinaisia, paljon valinnanvaraa, *



Ketterät ALM-työkalut (Application Lifecycle Management)

Esim. Rally, VersionOne, Jira Agile

+ Räätälöitävyys ja ominaisuuksien määrä tekevät mahdolliseksi hyvin erilaisten prosessien seurannan, jäljityksen ja raportoinnin

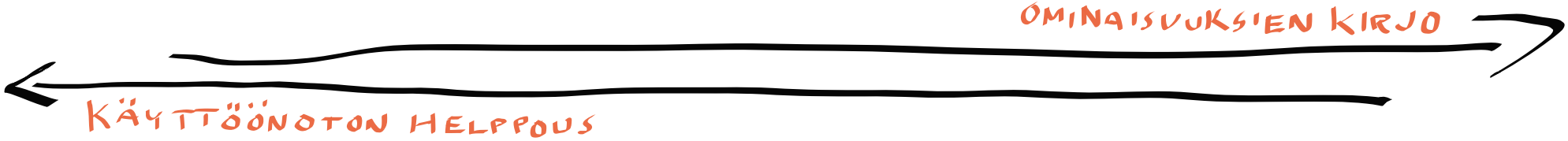
- Lisenssit, konfigurointi, ominaisuuksien määrä saattaa haitata käytettävyyttä, *

Pluginit

Esim. Agilo for Trac, Scrum for Trello, Redmine-Scrumbler

+ Voidaan hyödyntää olemassaolevaa työkalua ketteryyteen siirryttäessä, usein ilmaisia

- Konfiguroinnin työläys ja tuotetuen puute, *



Alkuvaiheen ketterä elinkaarimalli

Tällä ja seuraavalla aukeamalla esitellyt elinkaarimallit toimivat eräänlaisena sisällysluettelona käsikirjan tekniikkakatalogiin auttamalla valitsemaan, mitä tekniikoita kannattaa valita käyttöön missäkin vaiheessa. Alkuvaiheen ketterä elinkaarimalli esittelee AgiES-tutkimusprojektissa (s. 72) muodostetun suosituksen ketteryyden käyttöönottoa aloitteleville organisaatioille.

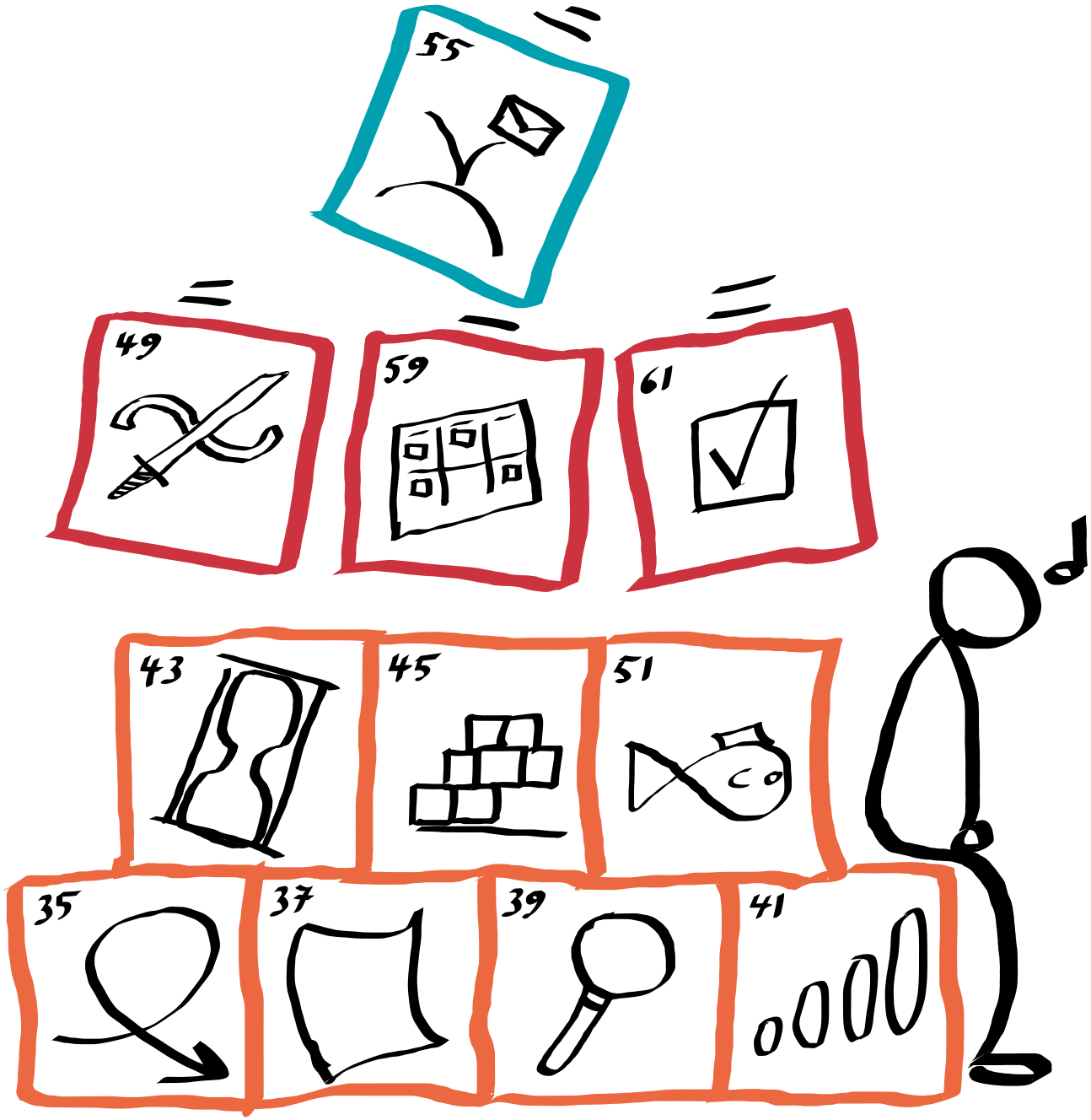
Käsikirjan tekniikkakatalogissa esitellään monipuolisesti sulautettujen järjestelmien ketterään kehitykseen soveltuvia käytäntöjä. Tarkoituksena ei ole tarjota valmista mallia, vaan kannustaa organisaatioita ja jopa yksittäisiä tiimejä hakemaan itselleen parhaiten toimivat menetelmät. On huomattava, että yksittäisten käytäntöjen lisäksi organisaatiossa pitää hallita ketteryyden arvot ja periaatteet. Lisäksi erilaiset ketterät menetelmät – kuten Scrum (s. 4) ja XP (s. 6) – tarjoavat pitkälle hiottuja malleja ketteryyden toteuttamiseen. Kuitenkin on hyvä pitää mielessä, että nämä ohjelmistokehitykseen suunnitellut menetelmät vaativat soveltamista sulautettujen järjestelmien kehittämiseen.

Alkuvaiheen elinkaarimallin tekniikat

Motivaatio ketterään kehitykseen löytyy käytössä olevien menetelmien haasteista, joita ovat tyypillisesti kommunikaation, tiimityön ja yhteisen priorisoinnin puutteellisuus. Vesiputousmaisesta kehityksestä ketteriin menetelmiin siirtyvässä sulautettuja järjestelmiä kehittävissä organisaatioissa ensimmäisenä käyttöönotettavat ketterät käytännöt ovatkin usein hyvin samanlaisia. Suurin osa ensivaiheen käytännöistä liittyy iteratiivisten rakenteiden tuomiseen jokapäiväiseen työhön.

Iteratiivisuutta edistäviä käytäntöjä ovat: *Iteraatio* (s. 35), *iteraation katselmointipalaveri* (s. 39), *iteraation suunnittelupalaveri* (s. 37) ja *säännölliset tilannepalaverit* (s. 43). Iteratiivisten käytäntöjen lisäksi on äärimmäisen tärkeää, että organisaatio harjoittelee alusta lähtien tiimien *itseorganisoituvuutta* (s. 51) ja tähän liittyvää *retrospektiivia* (s. 41). Näin voidaan varmistaa, että menetelmät mukautuvat tiimin tarpeisiin eikä toisin päin. Näiden ketteryyden perusrakenteiden lisäksi alkuvaiheen elinkaarimalliin kuuluu *kehitysjonon* (s. 45) käyttöönotto, jolla opetellaan ketterää tapaa seurata työn tekemistä. Tämä voi erota merkittävästi suunnitelmalähtöisten prosessien dokumentoinnista – riippuen siitä, kuinka tunnollisesti organisaatio on työtään dokumentoinut.

Yllä kuvattujen, lähes kaikille organisaatioille keskeisten käytäntöjen lisäksi ketteryyden käyttöönottovaiheessakin voi olla syytä miettiä muutamien lisäkäytäntöjen omaksumista. *Ketteryyden tarkistuslista* (s. 61) on yksinkertainen työkalu, joka konkretisoi käyttöönottovaiheessa ketteryyden käsitteitä ja sopii erityisesti organisaatioihin, joiden henkilöstössä on hyvin vähän aiempaa kokemusta ketteryydestä. Mikäli kehitysjonon käsite tuntuu abstraktilta *tehtävätaulu* (s. 59) tarjoaa tekniikan kehitysjonon toteuttamiseksi. Erityisesti laitteistopainotteisessa kehittämisessä voi *töiden paloittelu* (s. 49) iteraatioihin osoittautua haastavaksi. Tähänkin haasteeseen tekniikkakatalogi tarjoaa ratkaisuja. Joissakin tapauksissa ketteryyden alkuvaiheeseen voi soveltua myös *keskeytysten hallinta* (s. 55). Vaikka kyseessä onkin periaatteessa edistyneempi tekniikka, siihen perehtyminen on hyödyllistä työympäristössä, jossa suurin osa kehittäjistä työskentelee useammissa projekteissa samanaikaisesti tai työhön liittyy poikkeuksellisen paljon ylläpito- ja tukitöitä.



Alkuvaiheen elinkaarimallin hyödyt

Alkuvaiheen elinkaarimalli pyrkii toteuttamaan ketteryyden kannalta keskeiset käytännöt, jotka antavat tekemiselle oikean rakenteen ja tukevat ketteryyden periaatteiden ymmärtämistä. Valituilla iteratiivisilla tekniikoilla, esimerkiksi vaatimusten jäädyttämisellä iteraatioiden ajaksi, pyritään tasaisempaan työkuormaan ja työrauhan takaamiseen iteraatioiden ajaksi. Ketteryyden kautta pyritään lisäämään työn läpinäkyvyyttä ja kehittämään kommunikaatiota avoimempaan suuntaan. Tätä tavoitetta valituista tekniikoista edistävät iteraation katselmointipalaveri, jossa myös asiakkaat voivat olla mukana, ja päivittäiset kehittäjien keskeiset tilannepalaverit. Ketterässä kehityksessä kehittäjien oma vastuu ja samalla myös vaikutusvalta omaan työhön korostuu. Tätä tuetaan erityisesti kehitysjonon, itseorganisoituvuuden sekä retrospektiivien kautta. Näitä tekniikoita soveltaessaan kehittäjät päättävät itse mm. iteraatioon otettavista töistä ja projektissa hyödynnettävistä menetelmistä ilman vahvaa ylhäältä tulevaa ohjausta.

Kun alkuvaiheen elinkaarimalli on saatu toimimaan, se tarjoaa hyvän pohjan kehittyneempien ketterien tekniikoiden soveltamiseen sekä ketteryyden räätälöimiseen organisaation itsensä näköiseksi.

Edistynyt ketterä elinkaarimalli

Kun ketterät menetelmät on tuotu onnistuneesti organisaatioon ja kehitystiimit alkavat ymmärtämään ketteryyden periaatteita, voidaan ryhtyä ottamaan käyttöön edistyneempiä ketteriä tekniikoita. Näissä tekniikoissa korostuvat sulautettujen järjestelmien kehityksen erityispiirteet. Kun alkuvaiheen tekniikat muistuttivat eri organisaatioissa paljon toisiaan, edistyneen elinkaarimallin tapauksessa lähdetään rakentamaan juuri organisaation tarpeita palvelevia käytäntöjä. Tätä kautta voidaan erottua kilpailijoista ja saada ketteryyden kautta todellista kilpailuetua.

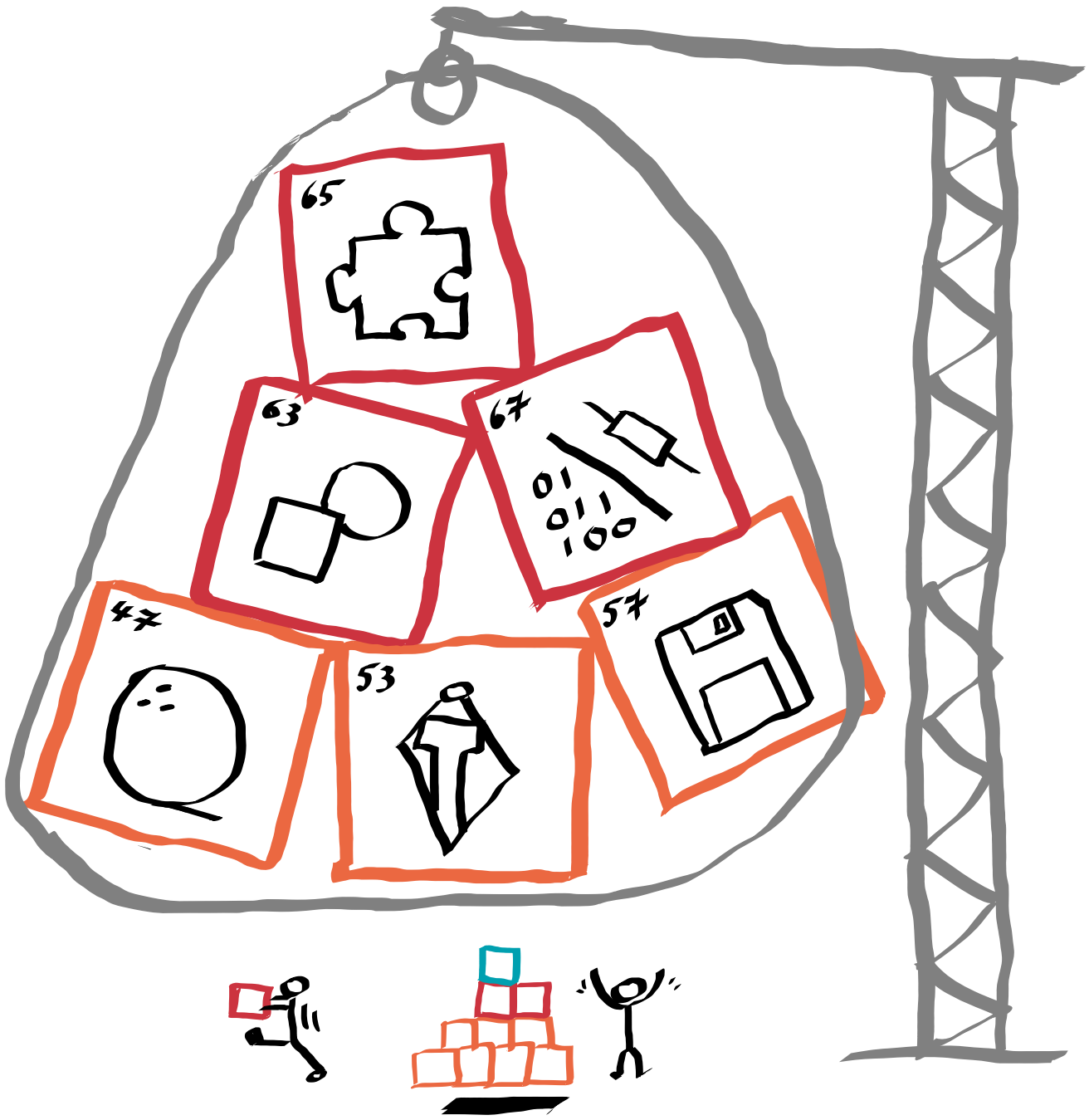
Edellisellä aukeamalla esiteltiin alkuvaiheen ketterä elinkaarimalli, jossa rakennettiin ketteryyden perusteita tekemällä työtä iteratiiviseksi, antamalla tiimeille vastuuta omasta työstään itseorganisoituvuuden kautta ja ottamalla askeleita ketterän dokumentoinnin suuntaan kehitysjonoa hyödyntämällä. Lisäksi esiteltiin muutamia hieman pidemmälle meneviä tekniikoita, jotka sopivat tiettyihin kehitysympäristöihin.

Kun perusteet ovat kunnossa kehitystiimeissä, voidaan lähteä ottamaan käyttöön edistyneempiä tekniikoita ja räätälöimään ketteriä menetelmiä enemmän organisaation näköisiksi. Kiirehtiä ei kannata, sillä monet perustekniikoista luovat pohjaa hienostuneemmille tekniikoille. Käytännössä edistynyt elinkaarimalli sisältää lähes kaikki käsikirjan tekniikkakortit ja kuvaa yhdenlaisen ideaalitilanteen, jolla ketterää kehitystä voidaan hyödyntää sulautettujen järjestelmien kehityksessä täysipainoisesti.

Uudet tekniikat voidaan jakaa kahteen kategoriaan. Ensimmäisen kategorian tekniikat täydentävät jo käyttöönotettuja. *Valmistellun vaatimuksen ja valmiin ominaisuuden määritelmät, sekä hyväksymiskriteerit* (s. 47) tarkentavat kehitysjonon käyttöä ja nopeuttavat iteraatioiden välissä pidettävien palaverien työskentelyä. *Tuoteomistaja* (s. 53) tuo uuden roolin itseorganisoituvan tiimin rinnalle toimimaan tuotteiden ja asiakkaiden rajapinnassa. Tämä on tärkeää erityisesti sulautetuissa järjestelmissä, joissa asiakas on harvoin yksi yksittäinen taho, vaan tuotteita tehdään massamarkkinoille. *Ketterän työkaluohjelmiston valitseminen* (s. 57) puolestaan auttaa hyödyntämään digitaalisia työkaluja aiemmin esiteltyjen kehitysjonojen organisointiin.

Ensimmäisen kategorian tekniikat ovat tuttuja jo ohjelmistokehityksen puolelta, joskin tekniikkakatalogissa niitä pyritään esittelemään sulautettujen järjetelmien näkökulmasta. Sen sijaan toisen kategorian tekniikat – *testivetoinen kehitys* (s. 63)*, modulaarinen suunnittelu* (s. 65) ja *kaksikohdesuunnittelu* (s. 67) – ovat sellaisia, joissa sulautettujen järjestelmien vaatimukset otetaan kokonaisvaltaisesti huomioon. Samalla ne ovat myös tämän käsikirjan teknisimpiä osioita. Ketteryyden käytön lisääntyessä sulautettujen järjestelmien saralla nämä edistyneet tekniikat ovat alttimpia kehittymään edelleen.

Uusia tekniikoita omaksuttaessa on hyvä myös miettiä, voidaanko jättää jotain alkuvaiheen tekniikoita pois organisaation oppiessa ketteryyttä. Vaikka monet esitetyistä tekniikoista saattavat ajan saatossa muuttua rutiiniksi on lähes kaikilla paikkansa kokonaisuudessa. Oikeastaan ainoa tekniikka, jonka merkitys alkuvaiheen jälkeen vähenee on *ketteryyden tarkistuslista* (s. 61), sillä sen keskeinen ajatus on konkretisoida alkuun ehkä vaikeita ketteryyden käsitteitä.



Mitä edistyneiden tekniikoiden jälkeen?

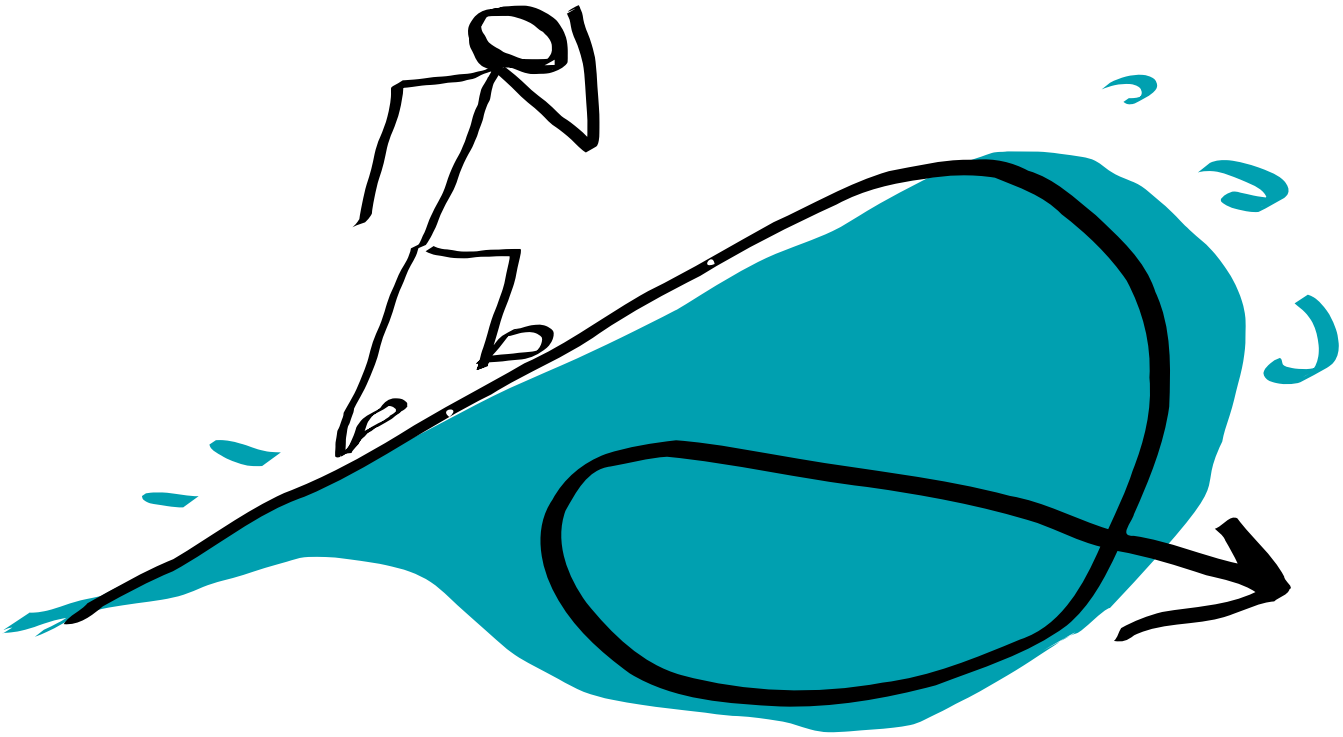
Edistyneessä elinkaarimallissa luetellaan lähes kaikki tekniikkakatalogin tekniikat. Tämä lista ei kuitenkaan ole koko totuus ketterien menetelmien hyödyntämisestä. Ketterien menetelmien yleistyessä myös sulautettujen järjestelmien parissa kehitetään uusia tekniikoita ja näistä on entistä enemmän tietoa saatavissa. Tällöin onkin tärkeää pysyä ajantasalla menetelmien kehityksessä internetin, kirjallisuuden ja alan tapahtumien kautta. Kun yrityksen henkilöstö alkaa olla vihkiytynyttä ketterän kehityksen arvoihin ja periaatteisiin, myös omien käytäntöjen kehittäminen on täysin mahdollista.

Ketteryyden käyttöönotto ei ole koskaan valmis. Jokaisella organisaatiolla ja jopa tiimillä on oma, itselleen sopiva tapa tehdä kehitystyötä, johon vaikuttaa paitsi tiimin sisäinen dynamiikka myös ympäristön (esim. asiakas, tuote, liiketoiminta) luomat vaatimukset. Näin ollen esimerkiksi liiketoimintaympäristön muuttuessa todennäköisesti myös kehitysmenetelmien tulee muuttua. Vaikka ympäristön suhteen saavutettaisiinkin kohtuullisen muuttumaton tilanne, aina on mahdollista parantaa toimintatapoja tai vähintäänkin seurata tarvetta muutokseen. Onneksi ketterät menetelmät tarjoavat valmiita tekniikoita työtapojen kehittämiseen retrospektiivien ja itseorganisoituvuuden kautta. Kun näistä tekniikoista ei tingitä, varmistetaan ketterän menetelmän sopivuus organisaation kulloiseenkin tilanteeseen.

Tekniikkakatalogi

Ketterä prosessi

Iteraatio



Ongelma: Työskentelytahti vaihtelee projektin eri vaiheissa työmäärän ja kiireen lisääntyessä projektin loppua kohti. Toisaalta tavoite saattaa olla kaukainen tai epäselvä, jolloin projekti tuntuu loppumattomalta.

Ratkaisu: Lyhyet työskentelyjaksot eli iteraatiot rytmittävät projektia ja jakavat työmäärää tasaisemmin. Lisäksi jokaisella iteraatiolla on oma välitavoitteensa, joka auttaa pilkkomaan projektia pienemmiksi, helpommin ymmärrettäviksi etapeiksi.

Tavoite: Työkuorma jakaantuu tasaisemmin koko projektin ajalle, lyhyen aikavälin suunnittelu ja seurattavuus paranevat sekä projektiin saadaan välitavoitteita.

Iteraation tarkoituksena ketterässä kehityksessä on tuottaa jokaisen lyhyen jakson päätteeksi jonkinlainen toimiva lopputuote tai tuotteen osa. Sulautetuissa järjestelmissä tämä lopputuote ei välttämättä ole konkreettinen fyysinen tuote, vaan myös erilaiset demonstraatiot toimivat iteraation tuotoksina. Jokaisen iteraation tarkoituksena on tuottaa hieman edellistä valmiimpi lopputulos. Iteraatiot seuraavat toisiaan, kunnes tuote on valmis. Iteratiivinen kehitys mahdollistaa reagoimisen asiakkaan toiveisiin, nopean suunnan muuttamisen sekä toimivan tuotteen toimittamisen nopeasti.

Iteraatiolla on hyvä olla sovittuna jokin tavoite. Tavoite kuvaa mihin kyseisen iteraation aikana keskitytään. Näin projekti ja sen päätavoite tuntuvat konkreettisemmilta ja järkevimmiltä. Välitavoitteen olemassaolo ja sen mukanaan tuoma tunne pienistä onnistumisista motivoi työntekijää. Iteraation tavoite antaa työrauhaa ja mahdollistaa laadun pitämisen korkeana.

Iteraatio aloitetaan yleensä suunnittelemalla iteraation tavoite, sekä mitä pitää tehdä, jotta tähän tavoitteeseen päästään. Tämä tapahtuu usein suunnittelupalaverissa (s. 37). Suurin osa ite-

raatiosta kuluu välitavoitteen toteuttamiseen ja toteutumista voidaan seurata esimerkiksi kehitysjonon (s. 45) kautta. Iteraation lopuksi on hyvä tarkastella esimerkiksi katselmointipalaverissa (s. 39), miten tavoitteen saavuttaminen lopulta onnistui. Näin opitaan muodostamaan realistisempia välitavoitteita.

Iteraatio ei saa olla liian lyhyt, muttei toisaalta liian pitkäkään. Hyvä iteraation pituus on yleensä 1–4 viikkoa. Iteraatio on sopivan pituinen silloin, kun se kyetään suunnittelemaan riittävän yksityiskohtaisesti, mutta toisaalta tavoitteet tuntuvat vielä riittävän suurilta. Pituus kannattaa säilyttää samana iteraatiosta toiseen, sillä tämä helpottaa suunnittelua, kun ymmärrys yhteen iteraatioon mahtuvasta työmäärästä lisääntyy. Lisäksi tasapituiset iteraatiot luovat toimivia rutiineja.

Jakamalla koko projektin työmäärä pienempiin iteraatioihin kasvatetaan ymmärrystä siitä, mitä voidaan saada aikaiseksi tietyn ajanjakson aikana. Tällöin ymmärretään paremmin suunniteltujen aikataulujen realistisuus ja voidaan laskea ja seurata tiimin työn nopeutta, jolloin aikatauluarviot osuvat paremmin oikeaan. Työn vaiheistus ja jakaminen sekä lyhyen aikavälin suunnitelmat takaavat sen, että projektin alkupuolellakin on konkreettisia tavoitteita ja työkuorma tasoittuu tasaisemmin koko projektin ajalle.

Parhaimmillaan iteratiivinen työskentely rauhoittaa työn tekemistä, mahdollistaa keskittymisen ja johtaa myös työntekijän näkökulmasta kontrolloidumpaan toimintaan. Osatavoitteiden toteuttamisen kautta valmiiksi saatavat työt lisäävät osaamisen ja onnistumisen tunnetta sekä työn mielekkyyttä.

Käytännössä koettua

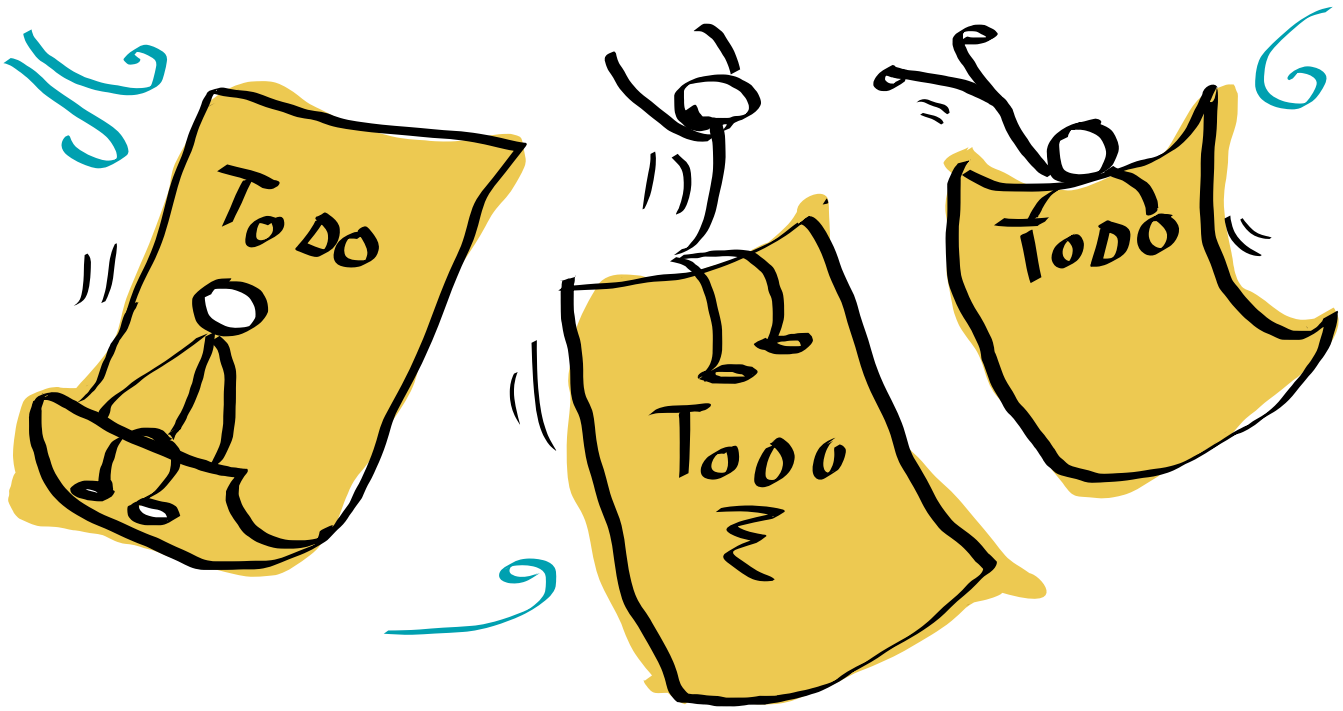
Eräs projekti tuntui määrättömän pitkältä ja tavoite kaukaiselta. Tämän vuoksi otettiin käyttöön iteraatiot. Ensimmäiseksi iteraatioksi valittiin seitsemän viikkoa, koska tällöin oli projektissa muutenkin tulossa konkreettinen välitavoite ja se haluttiin hyödyntää iteraatioon välitavoitteeksi. Tämän iteraation päätteeksi todettiin, että yksityiskohtainen suunnitelmallisuus oli mahdotonta näin pitkälle aikavälille. Sen vuoksi sovittiin, että jatkossa iteraatiot kestävät tasan neljä viikkoa, ja näin kyettiin suunnittelemaan paremmin, mitä iteraation aikana toteutetaan.

Vinkkejä

- Iteraatio ja sen toteuttamiseen liittyvät rutiinit auttavat työn hallinnassa ja kuormituksen säätelyssä. Iteraation käyttöönotto on oppimisprosessi, jossa alkuun voi ilmetä ongelmia iteraatioon mahtuvan työmäärän arvioinnissa ja siinä, miten liikaan työhön suhtaudutaan jakson aikana. Tällaisessa tilanteessa on huomioitava se, että liiallista työkuormaa ei lähdetä selvittämään työn laatua laskemalla tai tekemällä ylitöitä.
- Iteraation aikana on myös tärkeää säilyttää työrauha ja välttää lisäämästä tarpeettomasti epävarmuutta jakson tavoitteen pitävyydestä. Kaikkiaan iteratiivinen työskentely vaatii kokonaistavoitteen eli vision siitä mihin suuntaa yksittäisillä jaksoilla pyritään. Visio on tärkeä muun muassa silloin, jos iteraation kuluessa joudutaan miettimään jakson työmäärän uudelleen mitoittamista.

Ketterä prosessi

Iteraation suunnittelupalaveri



Ongelma: Kehitystiimillä ei ole selkeää käsitystä siitä, mitä työtehtäviä tulisi tehdä seuraavaksi ja miten omat työt linkittyvät toisten töihin.

Ratkaisu: Järjestetään säännöllinen ja maksimissaan yhden työpäivän mittainen suunnittelupalaveri ennen jokaisen iteraation aloittamista.

Tavoite: Iteraation kehitysjonoon on määritettynä työtehtävät, jotka ovat realistisesti toteutettavissa seuraavan iteraation aikana, sekä iteraatiolla on tavoite, joka toteutuu nämä suunnitellut työtehtävät tekemällä.

Suunnittelupalaverissa on ideana, että tiimi selvittää itselleen kaksi asiaa: iteraation (s. 35) tavoitteen ja iteraation työtehtävät. Suunnittelupalaveri on tyypillisesti aikarajattu: esimerkiksi Scrumissa (s. 4) tilaisuudelle suositellaan käytettävän aikaa 30 päivän iteraatioiden tapauksessa 8 tuntia, muuten tilaisuus olisi hyvä pitää vieläkin lyhyempänä.

Ennen tilaisuutta tuotteen kehitysjonon (s. 45) tulee olla priorisoitu ja ajan tasalla. Priorisoinnin hoitaa tuoteomistaja (s. 53) keskustelemalla tiimin ja asiakkaan kanssa. Kaikkien paitsi ensimmäisen iteraation tapauksessa tuotteen kehitysjonoa on juuri päivitetty iteraation katselmointipalaverissa (s. 39), jolloin kehitystiimillä on tuoreessa muistissa kokonaiskuva tuotteen kehityksen etenemisestä.

Suunnittelupalaverin alussa käydään läpi korkeimman prioriteetin vaatimukset siten, että kaikilla on yhteinen ymmärrys vaatimuksista. Tuoteomistajalla on tässä tärkeä rooli hänen vastatessaan tiimin kysymyksiin kohtien tarkoituksesta ja yksityiskohdista. Yhteisen ymmärryksen myötä tiimi kykenee valitsemaan ja arvioimaan kuinka paljon vaatimuksista pystytään toteuttamaan iteraation aikana.

Valituista vaatimuksista pilkotaan tehtäviä (s. 49), jotka ovat tyypillisesti korkeintaan muutaman työpäivän mittaisia ja aina yhden kehittäjän tehtävissä. Tiimi tuntee oman työnsä parhaiten ja suunnittelee näin yhdessä miten vaatimukset saadaan käytännössä toteutettua (s. 51). Tehtävien pilkkomisen yhteydessä vaatimusten työmääräarviot saattavat päivittyä, ja onkin tärkeää tarkistaa vielä lopuksi, että iteraatioon on valittu sopiva työmäärä.

Iteratiivisessa kehityksessä pyritään tuottamaan jokaisessa iteraatiossa uutta, toimivaa toiminnallisuutta tuotteeseen. Tämän vuoksi suunnittelupalaverissa määritellään myös iteraation tavoite. Se auttaa ajattelemaan iteraation työtehtäviä osana suurempaa kokonaisuutta irrallisten tehtävien sijaan.

Iteraation suunnittelupalaverin on tarkoitus olla hyvin keskusteleva tilaisuus. Tuoteomistaja ei määrää iteraation työtehtäviä, vaan niiden valinta on kehitystiimin päätettävissä.

Ohjelmisto- ja laitteistotehtävien muodostaminen

Monessa sulautettuja järjestelmiä kehittävässä yrityksessä ohjelmisto- ja laitteistotiimeille kohdistuvat vaatimukset ovat tyypillisesti hyvin erilaisia ja niiden pilkkomiseen tehtäviksi ei välttämättä tarvita koko monilukuisen tiimin läsnäoloa. Tällöin voi olla perusteltua jakaa suunnittelupalaveri niin, että ensin molemmat tiimit muodostavat yhdessä iteraation kehitysjonon ja tavoitteen, ja töiden pilkkominen tapahtuu kahdessa erillisessä palavereissa.

Resepti

1. Varaa kehitystiimille aikaa suunnittelupalaveria varten.
2. Käy läpi iteraation tärkeimmät tavoitteet ja kehitysjonon kohdat tuotteen näkökulmasta.
3. Valitse mitkä ja miten monta kohtaa kehitysjonosta otetaan mukaan iteraatioon.
4. Muodosta suurista kehitysjonon kohdista hallittavan kokoisia (½–2 työpäivää) tehtäviä tiiviisti keskustelemalla.
5. Varmista, että tiimillä on valittuna sopiva työmäärä ja selkeä kuva siitä, mitä tehdään seuraavaksi, miksi se tehdään ja mitä keinoja tiimi aikoo siihen käyttää.

Vinkkejä

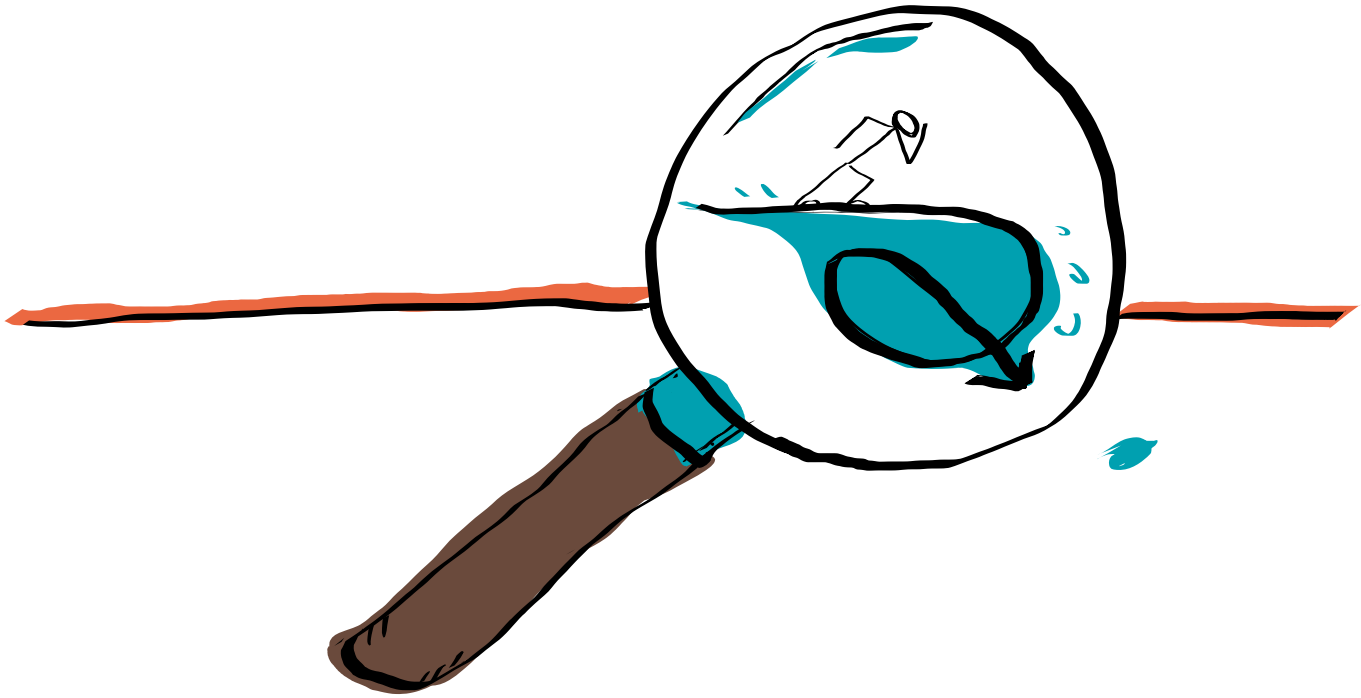
- Tuoteomistajan pitkän tähtäimen tuotevisio sekä hyvin muodostetut vaatimukset ovat suuressa roolissa suunnittelupalaverin sujuvuuden kannalta.
- Hajautettujen tiimien kohdalla tulee varmistaa teknisin apuvälinein mahdollisimman häiriötön kommunikaatio sekä järjestää mahdollisuuksien mukaan myös kasvokkaisia tapaamisia.
- Kokonaisen työpäivän kestävän suunnittelukokouksen järjestäminen työpaikalla voi olla vaikeaa muiden projektien ja työtehtävien aiheuttaessa keskeytyksiä ja häiriöitä. Ratkaisuna voidaan lyhentää iteraation pituutta ja samalla vastaavasti suunnittelupalaverin kestoa. Tilaisuus voidaan toteuttaa myös toimiston ulkopuolella poissa häiriölähteistä.

Lisää aiheesta

Stevens P.: *The Effective Sprint Planning Meeting*. http://bit.ly/1xGSnle

Schwaber K., Sutherland J.: *The Scrum Guide* (s. 8-10). http://bit.ly/ZgrC7Z

Iteraation katselmointipalaveri



Ongelma: Projektin etenemisestä ei ole tarkkaa tietoa, eikä asiakas pysty vaikuttamaan projektin etenemiseen.

Ratkaisu: Säännöllisin väliajoin iteraatioiden välissä asiakkaan kanssa pidetään palaveri, jossa tarkastellaan projektin edistymistä. Tarkoituksena on esittää projektin eteneminen esittelemällä tuotteen valmiita ominaisuuksia.

Tavoite: Esitellään projektin etenemistä säännöllisin väliajoin ja saadaan asiakkaalta välitöntä palautetta projektin etenemisestä sekä suunnasta, johon projektia seuraavan iteraation aikana viedään.

Iteraation katselmointi on jokaisen iteraation päätteeksi pidettävä palaveri, jossa käydään läpi, mitä iteraation aikana on saatu aikaan ja verrataan saavutuksia iteraation suunnittelupalaverissa (s. 37) tehtyihin tavoitteisiin. Palaverissa esitellään projektin edistymistä asiakkaalle. Tavoitteena on saada esiteltäväksi jokaisen iteraation päätteeksi jokin toimiva ominaisuus. Tämä tarkoittaa, että iteraation lopussa lisätty toiminnallisuus on toteutettu ja testattu.

Asiakkaan kanssa käydään yhdessä läpi tuotteen kehitysjono (s. 45) ja asiakas pääsee esittämään toiveensa tuotteen vaatimusten tärkeysjärjestyksestä. Vaatimukset voidaan määritellä valmiiksi vain yhdessä asiakkaan kanssa ja näin varmistutaan siitä, että molemmilla osapuolilla on yhteinen ymmärrys mitä kyseisen vaatimuksen onnistunut toteutus tarkoittaa (s. 47). Iteraation katselmointi ei ole pelkästään tilannepalaveri ja tuotteen esittelemisen tarkoituksena on myös saada asiakkaalta palautetta sekä vahvistaa yhteistyötä.

Iteraation katselmointi pidetään tarkoituksella hyvin epämuodollisena. Osallistujat ovat tyypillisesti tuoteomistaja, tiimi, johto, asiakas ja mahdollisesti tarpeen mukaan muiden projektin

kehittäjiä. Osallistujien kutsuminen tapahtumiin kannattaa miettiä tilannekohtaisesti esiteltävän tuotteen mukaan. Asiakkaan tai hänen edustajansa läsnäolo on kuitenkin tilaisuuden ydin.

Ideaalitilanteessa tiimi on saanut kaikki tuotteen kehitysjonosta tuodut vaatimukset toteutettua, mutta tärkeintä on saavuttaa asiakkaan korkeimman prioriteetin vaatimukset. Myös keskeneräiset vaatimukset on hyvä käydä läpi ja keskustella siitä, minkä takia niitä ei saatu toteutettua.

Yleisenä sääntönä palaverin pituudelle voidaan pitää yhtä tuntia jokaista iteraation viikkoa kohden. Esimerkiksi Scrumissa (s. 4) katselmoinnin pituudeksi määritellään 4 tuntia kuukauden sprinttiin. Jos tiimissä on fasilitoija (s. 52), hän organisoi palaverin tai varmistaa muuten, että se järjestetään, ja varmistaa, että tiimi pysyy aikataulussa.

Iteraation katselmoinnin lopputuloksena on päivitetty tuotteen kehitysjono seuraavaa suunnittelupalaveria varten. Kehitysjonoa tulisi tarkastella kuitenkin myös kokonaisuutena, eikä pelkästään seuraavaa iteraatiota varten.

Käytännössä koettua 1

Eräässä yrityksessä havaittiin, että iteraation katselmoinnin toteuttaminen johdatti tiimin sekä sen yksilöt ottamaan aiempaa paremmin vastuuta tehtävien toteuttamisesta. Avoimuuden saavuttaminen oli kuitenkin aluksi haasteellista ja vaati tottumista siihen, että tehty ja tekemätön työ tulee näkyväksi. Lisäksi oli opeteltava puhumaan ikävistä asioista ja ilmaisemaan selkeästi, jos jonkin asian ei nähty toteutuvan odotetusti. Iteraation katselmoinnin kautta myös yhteistyötahojen sekä sisäisen asiakkaan ymmärrys tiimin resursseista parani ja tilaukset sekä vaatimukset tulivat realistisemmiksi.

Käytännössä koettua 2

Asiakkaan yhteisymmärryksen löytämiseksi koettiin, että on tärkeää määritellä yhteiset ominaisuuksien hyväksymiskriteerit. Nämä kriteerit pakottivat määrittelemään vaatimukset tarkasti ja yksiselitteisesti. Ennen vaatimuksen toteutusta kriteerit piti määrittelyn lisäksi hyväksyttää asiakkaalla. Tämän avulla voitiin myös ratkaista mahdolliset konfliktitilanteet.

Resepti

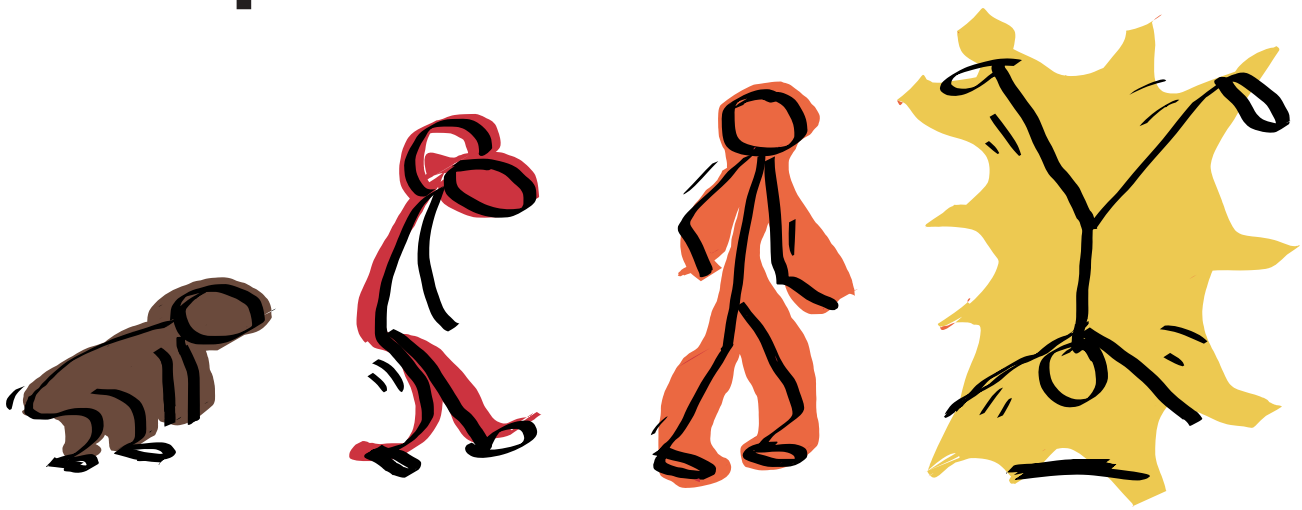
1. Käy läpi iteraation tavoitteet ja niiden toteutuminen.
2. Esittele tuotteen senhetkinen tilanne, esimerkiksi demonstraation tai simulaation kautta.
3. Käy läpi iteraation vaatimukset, sekä toteutuneet, että mahdollisesti syystä tai toisesta kesken jääneet.
4. Priorisoi tuotteen kehitysjono.
5. Käy läpi seuraavan iteraation tavoitteet.

Lisää aiheesta

Schwaber K., Sutherland J.: *The Scrum Guide* (s. 11). http://bit.ly/ZgrC7Z

Ketterä prosessi

Retrospektiivi



Ongelma: Tiimin työskentely ei tunnu tehokkaalta, eikä vie projektia eteenpäin. Tiimin jäsenet kokevat tiimin toimintatavat hankaliksi, mutta heillä ei ole keinoa, jolla parantaa työskentelyään yhdessä.

Ratkaisu: Järjestetään säännöllisesti tilaisuuksia, joissa tehtävänä on tarkastella yhdessä nykyisiä toimintatapoja, sekä erityisesti muokata menetelmiä toimivammiksi lisäämällä, poistamalla ja muuttamalla toimintatapoja.

Tavoite: Saadaan jokaisen tiimin jäsenen ääni kuuluviin siten, että tiimi itse päättää toimintatapojensa muokkaamisesta ja muuttamisesta. Jokaisessa retrospektiivissä päätetään yhdessä muutama käytännön toimenpide, sekä sovitaan niiden toteuttamisvastuusta ja seuraamisesta. Näin työskentelytavat kehittyvät koko ajan vastaten tiimin kokemuksia ja tarpeita.

Retrospektiivin alussa on hyvä kerrata, miksi olemme täällä ja mikä on tilaisuuden tavoite. Aluksi tarkastellaan sitä, mitä ja miten asioita on tehty valitun ajanjakson aikana ja lopuksi suunnitellaan muutoksia työskentelymenetelmiin ja sovitaan käytännön toimenpiteistä.

Retrospektiivillä voi olla jokin teema tai tarkasteltavan ajanjakson tapahtumia voidaan käsitellä yleisesti. Retrospektiivin osanottajien määrä voikin vaihdella aiheesta riippuen siten, että joskus tiimi kokoontuu yksinään ja joskus mukana on myös esim. myynnin, markkinoinnin tai asiakkaan edustajia. Mahdollinen teema voidaan tuoda esiin jo itse kutsussa. Kutsu voi sisältää myös ennakkotehtäviä, joiden avulla saadaan retrospektiivin osallistujat pohtimaan aihetta etukäteen.

Retrospektiivillä pyritään luomaan mahdollisimman monipuolinen kuva nykyisten työskentelymenetelmien toteutuksesta ja toimivuudesta. Usein hyödynnetään yksilö- pari- tai ryhmätöitä, joiden tuloksia esitellään muille. Näin saadaan helpommin myös hiljaisempien jäsenien ääni kuulumaan.

Valittaessa toteutettavia muutoksia huomio kannattaa kiinnittää siihen, että muutokset ovat riittävän pieniä, selkeitä ja toteutettavissa olevia. Iso ongelma on hyvä pilkkoa pienemmiksi toteuttamiskelpoisiksi askeliksi. On myös hyvä valita joku vastuulliseksi huolehtimaan, että muutokset toteutetaan. Edelleen on tärkeää seurata esimerkiksi seuraavassa retrospektiivissä, että sovitut muutokset on todella viety käytäntöön. Jos näin ei ole tapahtunut on hyvä pohtia, miksi – onko muutos ollut liian laaja, epämääräisesti määritelty vai jäikö vastuullinen valitsematta.

Retrospektiiville ominainen ajankohta on iteraation (s. 35) lopussa. Myös projektin lopussa on hyvä järjestää retrospektiivi, jotta opitaan, mikä on mennyt hyvin ja mitä voitaisiin muuttaa seuraavaan projektiin.

Retrospektiivin tarkoitus ei ole tarkastella varsinaisen työn tuloksia, vaan erityisesti työskentelymenetelmiä. Ketterien menetelmien periaatteena on, että tiimi itse saa päättää omista työskentelytavoistaan, eikä näitä sanella ylhäältäpäin. Retrospektiivissä käytävän keskustelun tulee olla luottamuksellista, jossa myönteisiä ja kielteisiä kokemuksia toimintatavoista voidaan jakaa avoimesti. Työskentelyn tehostamisen lisäksi retrospektiivillä kehitetään työtä sujuvammaksi ja hallittavammaksi ja sen tuloksena yhteinen ymmärrys työn tekemisen tavoista lisääntynyt.

Vinkkejä

- Joskus voi tulla eteen tilanne, jossa retrospektiivissä ei tule esiin uutta asiaa. On kuitenkin suositeltavaa, että retrospektiiveilla olisi säännöllinen ajankohta, vaikka se olisi harvemminkin. Jos niitä järjestetään ”tarpeen mukaan”, retrospektiivien pitämisestä tingitään helposti kiireisinä aikoina tai kun ihmiset kyllästyvät niihin pitkissä projekteissa. Näin voi käydä etenkin, jos retrospektiiveillä ei ole nimettyä järjestäjää tai tällä on niin paljon muita tehtäviä, ettei ehdi huolehtia järjestämisestä.
- Joskus kyseessä voi olla haaste, jota tiimi ei kykene itse ratkaisemaan. Tällöin retrospektiivin kehitysehdotus voi olla esimerkiksi viestin vieminen eteenpäin siitä, mikä helpottaisi tiimin toimintaa.
- Jos tiimille on nimetty fasilitoija (s. 52), hän voi huolehtia, että tiimiläiset järjestävät retrospektiivin itseorganisoituvasti ja hän voi myös auttaa retrospektiivin käytännön järjestelyissä. Hyvien retrospektiivien vetämistä tukevat vetäjän omistautuminen ja hyvät fasilitointitaidot.
- Toteutusvinkki: Yksi ryhmätyövaihtoehto on listata pienryhmissä hyvät ja huonot puolet sekä parannusehdotukset. Nämä kerätään yhteen ja parannusehdotuksista äänestetään käyttöönotettavat.

Lisää aiheesta

Derby E., Larsen D.: *Agile Retrospectives, Making Good Teams Great*.

Resepti

1. Valitse tilaisuuden järjestäjä, joka on vastuussa tilaisuuden suunnittelusta, vetämisestä, järjestelyistä sekä asianosaisten kutsumisesta tilaisuuteen. Järjestäjä voi lähettää muille myös ennakkotehtäviä. Järjestäjän rooli on sujuvoittaa retrospektiivin kulkua, järjestäjän ollessa itse mahdollisimman objektiivinen. Järjestäjän rooli voidaan sopia myös tiimin sisällä kiertäväksi.
2. Järjestäjä avaa tilaisuuden muistuttamalla siitä, mikä on tilaisuuden tavoite.
3. Tarkastellaan erilaisten tehtävien avulla toimintatapoja.
4. Suunnitellaan tehtävien avulla mahdollisia muutoksia toimintatapoihin.
5. Valitaan yhdessä mitä muutoksia toteutetaan ja varmistetaan, että valitut toimenpiteet ovat selkeitä, helposti toteutettavissa ja että joku on niistä vastuussa. Sovitaan myös miten toteutumista seurataan.
6. Keskustellaan hetki siitä, miten retrospektiiviä voisi parantaa seuraavaan kertaan.

Käytännössä koettua

Eräässä tiimissä säännölliset tilannepalaverit koettiin turhiksi. Retrospektiivissä tarkasteltiin aluksi ongelmaa: huomattiin, että kaikki eivät olleet aina paikalla, eivätkä muistaneet välttämättä edes ilmoittaa poissaoloistaan. Ratkaisuna sovittiin, että jatkossa jokaisen tilannepalaverin päätteeksi tarkistetaan seuraavan tilannepalaverin ajankohdan sopivuus kaikille. Tarvittaessa ajankohtaa siirretään hieman myöhemmäksi, jos poissaolon syynä on esimerkiksi juuri samaan ajankohtaan osuva toinen palaveri.

Säännölliset tilannepalaverit

Ongelma: Tieto ei kulje tiimien sisällä ja projektien kokonaisuuksien ymmärtäminen sekä etenemisen seuraaminen on haastavaa.

Ratkaisu: Järjestetään säännöllisesti tilannepalaveri, jossa jokainen osallistuja lyhyesti kertoo, mitä on tehnyt edellisen tilannepalaverin jälkeen, mitä suunnittelee tekevänsä ennen seuraavaa tilannepalaveria ja mitä mahdollisia ongelmia hänellä tällä hetkellä on. Ongelmia ei ratkaista palaverin aikana, vaan sovitaan, ketkä osallistuvat niiden ratkaisuun.

Tavoite: Kaikki tiimin työntekijät saavat selkeän yleiskuvan projektin tilanteesta ja siitä, mitä kukin on tekemässä. Näin osataan suhteuttaa oma tekeminen tiimin tekemiseen ja ymmärretään omien töiden vaikutus muiden tekemisiin.

Säännölliset tilannepalaverit ovat lyhyitä kokouksia, joiden tarkoituksena on koota kehitystiimi säännöllisesti yhteen, välittää tietoa kehitystiimin sisällä, pitää koko tiimi tietoisena projektin tilanteesta ja etenemisestä sekä helpottaa ongelmien aikaista tunnistamista sekä ratkaisemista.

Tuotekehitystiimin palaverikäytäntöjä suunniteltaessa on hyvä miettiä tilannepalaverien kokoonpanoa, kestoa, ajankohtaa sekä järjestämistiheyttä. Palaverien kokoonpano voi vaihdella pienestä, yhden asian ympärillä työskentelevästä ryhmästä kokonaiseen tuotekehitystiimiin. Koko tiimin yhteisien palaverien etu on tiedon kulku mahdollisimman laajasti. Toisaalta suuremmalla ryhmällä palaverit ovat luonnollisesti pidempiä ja osa käsiteltävistä asioista ei koske kaikkia läsnäolijoita. Tässä kohtaa kuitenkin kannattaa harkita kaikkien työajan käyttämistä palaveriin, jotta maksimoidaan laaja tiedonkulku ja odottamattomien ongelmien nopea esille nouseminen.

Säännöllisten tilannepalaverien kesto määräytyy siten, että jokaiselle osallistujalle varataan noin minuutti omien asioidensa kertomiseen. Näin ollen palaverit pysyvät kohtuullisen pituisina, ei-

vätkä kuormita liikaa. Esimerkiksi Scrum-menetelmässä (s. 4) päivittäisiä tilannepalavereja suositellaan pitämään seisten. Tällöin tiimi pysyy vireämpänä, eivätkä palaverit venähdä niin helposti. Hyvä ajankohta tilannepalavereille on esimerkiksi aamupäivällä yhdeksän ja kymmenen välillä.

Tilannepalaverien järjestämistiheys voi vaihdella yhdestä viiteen kertaan viikossa riippuen tiimin rakenteesta ja tehtävän työn luonteesta. Hyvä lähtökohta voi olla joka toinen päivä pidettävä palaveri, jotta kerrottavaa asiaa ei kerry yhteen palaveriin liikaa, mutta toisaalta palaverien väliin jää aikaa tehdä suunniteltuja tehtäviä.

Käytännössä koettua 1

Eräässä tiimissä havaittiin, että harvempi aikataulu ja suuremmat ryhmät toimivat paremmin kuin pienet ryhmät ja tiheä aikataulu. Pienissä ryhmissä tiedonsiirtoa ei syntynyt, koska kaikki tiesivät jo lähimpien työkavereidensa tekemiset. Toisaalta iso ryhmä ja lyhyt palaveriväli voivat johtaa turhautumiseen, kun samoja asioita kuulee toistuvasti. Pilottiyrityksessä todettiin, että isommalla ryhmällä kaksi kertaa viikossa pidetyt palaverit eivät venyneet liian pitkiksi ja rasittaviksi ja toisaalta esille nousi ongelmia, jotka eivät olleet paljastuneet päivittäisissä keskusteluissa.

Vinkkejä

- Äänekkäimmät henkilöt saattavat dominoida palavereissa. Varmista tasapuolinen ajankäyttö kaikkien kesken. Erityisesti hajautetuissa tiimeissä varmistu siitä, että myös hiljaisempien ääni tulee kuuluviin.
- Kasvokkaisen kommunikaation niukkuus on hajautettujen tiimien haaste. Kiinnitä erityistä huomiota hyvien hajautettujen palaverikäytäntöjen luomiseen ja järjestä toisinaan myös hajautetulle tiimille kasvokkaisia tapaamisia.

Käytännössä koettua 2

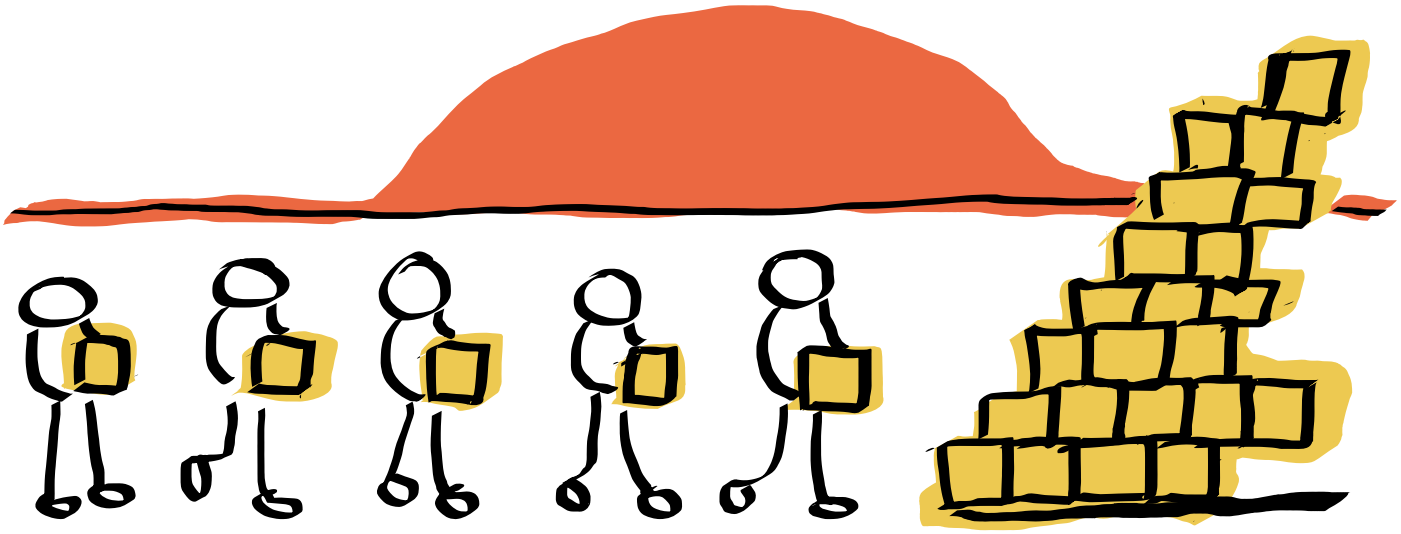
Toisessa, hajautetusti työskentelevässä tiimissä videoneuvottelun avulla toteutetut pikapalaverit koettiin erittäin tärkeiksi. Raskaaksi ja pitkiksi koettuja palavereja kehitettiin tehokkaammiksi ja kevyemmiksi. Paras tulos saavutettiin, kun sovitusta aikataulusta pidettiin kiinni ja jokaiselle annettiin vain lyhyt aika tilannekatsaukseen.

Pikapalaverin nähtiin lisäävän työskentelyn läpinäkyvyyttä, kommunikaatiota ja vuorovaikutuksen avoimuutta. Palaverissa saatu tieto lisäsi oman työn sujuvuutta ja sijoittamista kokonaisuuteen. Mahdollisuus kertoa heti kohtaamistaan ongelmista ja jatkotyöskentelyn selkeys niiden ratkaisemiseksi vähensivät haastavien tehtävien kuormittavuutta.

Resepti

1. Määrittele palaverien kokoonpanot.
2. Määrittele palaverien järjestämistiheys, kesto ja ajankohdat.
3. Valvo, että sovituista palaverikäytännöistä pidetään kiinni.
4. Kerää kokemuksia ja kehitä käytäntöjä niiden mukaisesti. Kohdissa 1 ja 2 määriteltyjä yksityiskohtia voidaan säätää kokemuksien karttuessa.

Kehitysjono



Ongelma: Työn organisointi, priorisointi ja seuranta tiimin sisällä ja sen yhdistäminen asiakkaan toiveisiin ei ole sujuvaa.

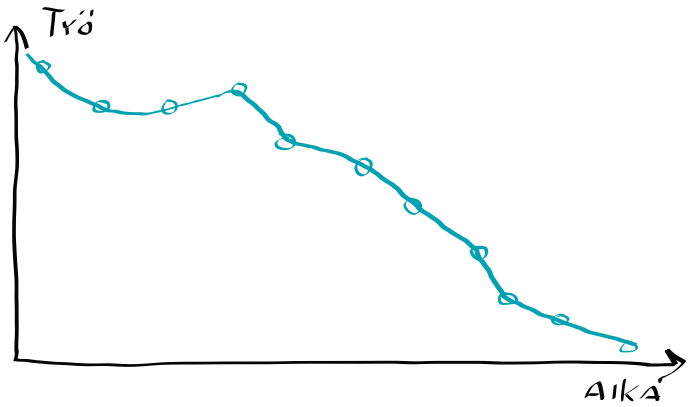
Ratkaisu: Kehitysjonojen avulla voidaan tiimin työ organisoida ja työn etenemistä seurata asiakkaan toiveiden mukaan. Tuotteen kehitysjonon avulla huomioidaan asiakkaan toiveet sekä organisoidaan projektikokonaisuus. Iteraation kehitysjonon avulla jaetaan tuotteen kehitysjonosta valitut tärkeimmän prioriteetin kohdat tehtäviksi, jotka tiimi toteuttaa iteraation aikana.

Tavoite: Projektin tavoitteet sekä eteneminen tulevat selkeäksi ja näkyväksi niin, että tiimi ja asiakas tietävät molemmat projektin tilanteen koko kehityksen ajan.

Kehitysjonoja on olemassa kahdenlaisia – tuotteen kehitysjonoja ja iteraation kehitysjonoja. *Tuotteen kehitysjono* kuvaa tuotekokonaisuuden ja täten näyttää mitä asiakkaalle lopulta toimitetaan. *Iteraation kehitysjono* puolestaan kuvaa iteraation aikana toteutettavia tehtäviä.

Tuotteen kehitysjonoa hallinnoi tyypillisesti tuoteomistaja. Tuoteomistaja (s. 53) on vastuussa tuotteen arvon maksimoimisesta sekä kehitystiimin työstä. Tuotteen kehitysjonoon kirjataan selkeästi vaatimukset, jotka tuotteessa tulee olla. Tuoteomistaja priorisoi vaatimukset ja antaa riittävän yksityiskohtaiset tiedot iteraation suunnittelupalaverin (s. 37) aikana, jotta tiimi voi arvioida vaatimusten työmääriä ja toteuttaa niistä mahdollisimman monta seuraavan iteraation aikana. Onnistuneen tuotteen kehitysjonon hallinnan kannalta on tärkeää, että tuoteomistaja kommunikoi tiiviisti kehittäjien kanssa, vaikka hän tekee itse viime kädessä päätökset. Tuotteen kehitysjonon kautta vaatimukset siirtyvät kontrolloidusti kehittäjätiimin toteutettavaksi.

Jokaista iteraatiota varten luodaan uusi iteraation kehitysjono suunnittelupalaverin aikana. Iteraation kehitysjonoon kehitystiimi luo jokaiselle käyttäjätarinalle yksityiskohtaiset tehtävät ja arvioi jokaiseen tehtävään kuluvan ajan. Suuret tehtävät pilkotaan pienimmiksi niin, että ne voidaan suorittaa yhden tai kahden päivän aikana (s. 49). Tehtäviä ei määrätä tiimin ulkopuolelta, vaan tiimi ja sen jäsenet itse päättävät työtehtävät ja vastuut. Tiimin itseorganisoitumisen (s. 51) toteutumisen kannalta on tärkeää, että tiimi saa arvioida itse tekemänsä työmäärän iteraation aikana. Iteraation kehitysjonoa päivitetään säännöllisesti iteraation aikana tehdyn työmäärän ja arvioidun jäljellä olevan työmäärän suhteen.



Käytännössä koettua 1

Eräässä yrityksessä käytettiin kolmea erilaista kehitysjonoa: tuotteen kehitysjonoa, etappikehitysjonoa ja iteraation kehitysjonoa.

Etappikehitysjono oli lisätty kahden kehitysjonon tueksi helpottamaan ulkoisten aikataulujen ja etappien vaikutusta kehitystyöhön. Etappikehitysjono toimi tuote- ja iteraation kehitysjonojen välissä vaatimustasolla. Siihen siirrettiin tuotteen kehitysjonosta ne vaatimukset, jotka tulee olla toteutettuja ennen tiettyä ulkoisesti määrättyä aikatauluetappia.

Projektin etenemistä seurattiin avoimien ja suljettujen vaatimusten avulla käyttäen etapin edistysmiskäyrää, joka päivitettiin aina iteraation katselmoinnissa.

Etappikehitysjonon pohjalta laadittiin iteraation kehitysjono. Siinä olevien tehtävien estimaatit päivitettiin päivittäin ja iteraation edistymistä seurattiin tilannepalavereissa edistymiskäyrän avulla. Iteraation kehitysjonoa tai iteraation edistymiskäyrää käytettiin vain iteraation etenemisen seurantaan, ei projektitasolla.

Edistymiskäyrä

Oleellinen työkalu tehdyn ja jäljellä olevan työmäärän visualisointiin on edistymiskäyrä. Edistymiskäyrän vaaka-akseli kuvaa aikaa ja pystyakseli jäljellä olevaa työmäärää. Työtehtävät saadaan kehitysjonosta. Edistymiskäyrää voidaan hyödyntää niin iteraation kuin tuotteen kehitysjonon tukena. Estimoitujen tehtävien valmistuttua jäljellä olevaa työmäärää päivitetään, joka näkyy työn vähenemisenä edistymiskäyrässä.

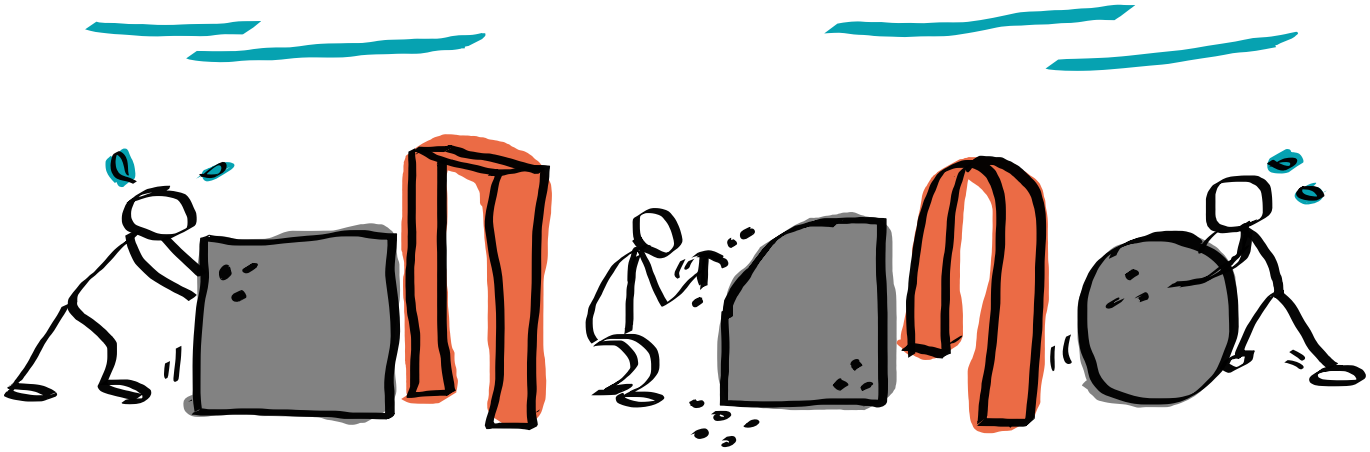
On tyypillistä, että työmäärä iteraation tai tuotekehityksen aikana kasvaa. Kaikkea toteutettavaa toiminnallisuutta ei yleensä osata ennakoida ja sitä lisätään tuotekehityksen aikana. Edistymiskäyrän avulla tiimi näkee työn etenemisen päivittäin koko projektin ajalta. Tällöin on mahdollista löytää myös tiimin tuotekehitysvauhti, joka kuvaa työskentelynopeutta, joka auttaa parantamaan työn suunnittelua ja estimoinnin tarkkuutta.

Käytännössä koettua 2

Eräässä yrityksessä havaittiin, että järjestelmien ylläpitotehtävät tulivat työprosessiin kehitysjonojen ulkopuolelta ja aiheuttivat suunnitellun työn aikarajojen ylittämispainetta, kuormittumista ja työrauhan häiriintymistä. Onkin tärkeää, että uudet työtehtävät tulevat suunnitteluprosessin ja kehitysjonon kautta. Yllättävien tehtävien kohdalla ennakointi on kuitenkin haasteellista. Tuoteomistajan ja tiimin kesken käytiin tärkeysjärjestyksen varmistava keskustelu aina, kun uusi tehtävä otettiin työn alle.

Tuotteen kehitysjono lopullisine tavoitteineen koettiin erittäin tärkeäksi myös työmotivaation, optimaalisen kuormituksen ja työn tehokkuuden säilymisen kannalta.

Valmistellun määritelmä, valmiin määritelmä ja hyväksymiskriteerit



Ongelma: Tiimi joutuu ottamaan iteraatioihin vaatimuksia, joissa on paljon tulkinnanvaraa, ja vastaavasti toteutetut ominaisuudet herättävät keskustelua niiden valmiusasteesta.

Ratkaisu: Määritellään projektille seuraavat säännöt: 1) valmistellun määritelmä, joka määrittelee vaatimusten minimitarkkuuden ja 2) valmiin määritelmä, joka määrittelee yhteiset minimikriteerit ominaisuuksien hyväksymiselle. Lisäksi keskeisille vaatimuksille määritellään hyväksymiskriteerit, jotka kertovat, millä ehdoilla vaatimus voidaan todeta toteutetuksi.

Tavoite: On olemassa yhteiset pelisäännöt, jotta kaikilla kehitystyön osapuolilla on selkeä näkemys tavoitteista ja tulkinnanvaraisuus vältetään.

Ketterässä kehityksessä painotetaan kevyttä dokumentointia ja vaatimusten nopeaa hallintaa. Yleinen tapa tämän toteuttamiseen on rakentaa tuotteelle kehitysjono (s. 45), jonka jokainen kohta kuvaa tuotteen keskeisiä ominaisuuksia esimerkiksi käyttäjätarinoiden avulla. Kehitysjonon kohtia voidaan lisätä, muokata ja poistaa melko vapaasti iteraatioiden välillä. Saavutetulla joustavuudella on myös kääntöpuolensa, sillä kehitysjonon kohtien kirjoittaminen vaatii osaamista ja kokemusta.

Monesti ongelmia syntyy kahdessa kohtaa: 1) kun kehitysjonon kohtia tulkitaan ja puretaan iteraation tehtäviksi ja 2) kun kehittäjät päättävät, milloin kehitysjonon kohta voidaa tulkita tehdyksi. Yhdessä nämä ongelmat johtavat siihen, että iteraatioiden välissä asiakkaan tai tuoteomistajan (s. 53) ja kehitystiimin näkemykset tuotteen valmiusasteesta eivät kohtaa. Tämän vuoksi kertaalleen työstettyjä kehitysjonon kohtia joudutaan palauttamaan takaisin kehitysjonoon. Tämä vaikuttaa kielteisesti kehitystyön osapuolten välisiin henkilösuhteisiin ja estää työn etenemisen järkevän seurannan. Ratkaisuna ongelmaan on määritellä tarkemmat säännöt töiden hyväksymiseen ja vastaavasti kehitysjonon kohtien kirjoittamiseen.

Hyväksymiskriteerit
Lähtökohtaisesti kaikkien kehitysjonon kohtien tulisi olla yksiselitteisiä, jolloin edelläkuvatun kaltaisilta ongelmilta vältyttäisiin. Käytännössä tähän on kuitenkin hyvin vaikeaa päästä ilman, että joustavuus ja ketteryys kohtien käsittelyssä menetetään. Hyväksymiskriteerit ovat yksiselitteiset, mielellään mitattavat kriteerit, joiden täytyttyä kyseinen kehitysjonon kohta voidaan todeta valmiiksi. Hyväksymiskriteerit eivät välttämättä kuvaa kaikkia mitattavia ominaisuuksia, vaan nostavat esille hyväksynnän kannalta keskeiset seikat, eräänlaisen minimin riman ylitykselle. Hyväksymiskriteerit liittyvät läheisesti testivetoiseen kehitykseen (s. 63) varsinkin, jos ne voidaan ilmaista formaalissa muodossa.

Valmiin ominaisuuden määritelmä
Hyväksymiskriteerit kuvaavat yksittäisen kehitysjonon kohdan hyväksymisedellytykset. Lisäksi on olemassa monia projektikohtaisia kriteereitä, jotka koskevat kaikkia tai lähes kaikkia työjonon kohtia. Näitä voivat olla esimerkiksi ominaisuuksien testausta, integrointia tai dokumentointia koskevat määritykset. Jotta vältetään näiden samojen asioiden kirjaaminen kerta toisensa jälkeen, kehitystiimi voi määritellä yhden yhteisen listan, jossa kuvataan yleinen valmiin määritelmä.

Valmistellun vaatimuksen määritelmä
Usein ongelmien lähtökohta on liian löysästi tai moniselitteisesti kirjoitettu vaatimuskuvaus. Valmistellun määritelmä kuvaa ne seikat, jotka kehitysjonon kohdasta tulee löytyä, jotta se voidaan ottaa johonkin iteraatioon toteutettavaksi. Mikäli kehitysjonon kohta ei täytä määritelmää, se palautetaan kirjoittajalle. Valmistellun määritelmässä voidaan kuvata esimerkiksi, missä muodossa vaatimus tulee kirjata ja mitä asioita vaatimusta varten tulee olla selvitettynä.

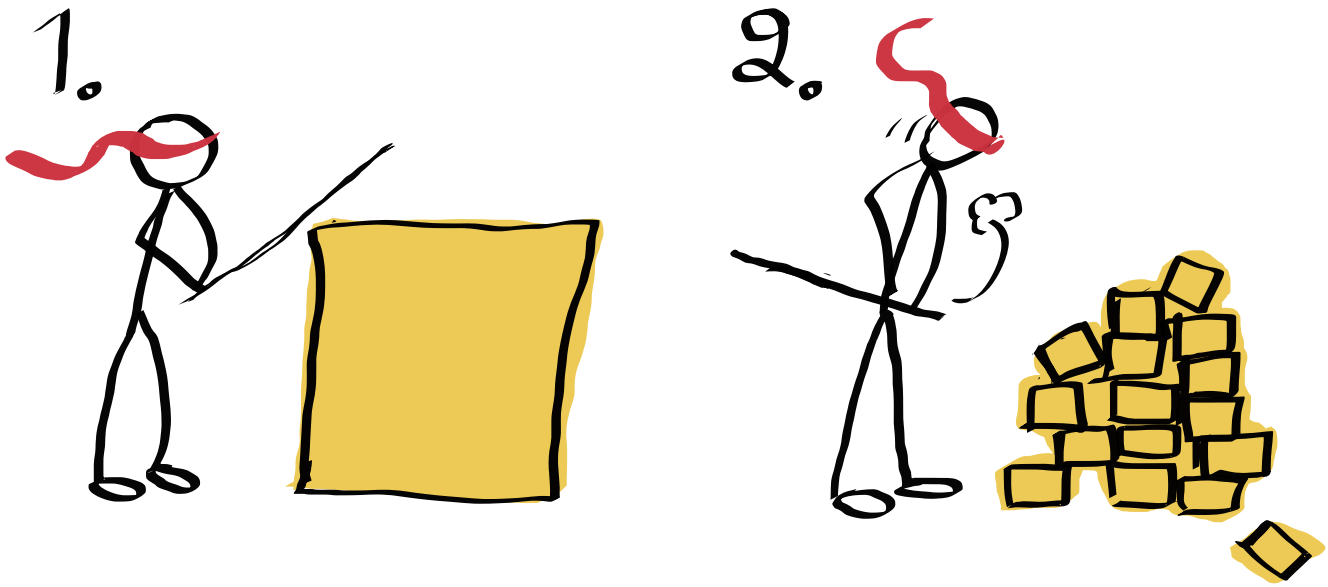
Hyväksymiskriteerit päättää tuoteomistaja tai asiakas. Sekä valmiin määritelmän että valmistellun määritelmän päättää kehitystiimi mahdollisesti yhdessä tuoteomistajan ja asiakkaan kanssa. Määritelmät tulee ilmaista lyhyenä ja selkeänä listana, jotta niiden toteutuminen on helposti tarkistettavissa tarvittaessa. Suositeltavaa on kirjoittaa listat isoille tauluille, jotka ovat jatkuvasti kehitystiimin nähtävissä. Mikäli listat eivät ole helposti saatavissa, niiden noudattaminen unohtuu helposti.

Valmiin ja valmistellun määritelmä sekä hyväksymiskriteerit ovat niin yleiskäyttöisiä tekniikoita, ettei niiden käyttö sulautettujen järjestelmien kehityksessä juurikaan eroa puhtaasta ohjelmistokehityksestä.

Esimerkki: Valmiin määritelmä
Kehitysjonon kohdan tulee olla:
- Testattu asianmukaisesti
- Dokumentoitu joko kommentein (SW) tai erillisessä tekstitiedostossa (HW)
- Integroitu testipalvelimelle tai HW-suunnitelmaan
- Tallennettu versionhallintaan, jotta se voidaan todeta valmiiksi.

Esimerkki: Valmistellun määritelmä
Kehitysjonon kohdan tulee:
- Olla toteutettavissa yhdessä iteraatiossa
- Kuvata selkeästi kohdan tuottama liiketoiminta-arvo
- Olla otsikoitu napakasti (2-3 sanaa)
- Kuvata keskeisimmät hyväksymiskriteerit yksiselitteisesti
- Olla toteutettavissa itsenäisesti riippumatta muista kehitysjonon kohdista, jotta se voidaan ottaa iteraatioon toteutettavaksi.

Töiden paloittelu



Ongelma: Työt on jaettu tehtäviin, mutta tehtävien koko on niin suuri, että niiden toteuttaminen kestää useamman iteraation tai kuukauden ajan. Tehtävien jakaminen pienempiin, yhden iteraation aikana toteutettaviin osiin tuntuu mahdottomalta.

Ratkaisu: Töiden paloitteluun sopivankokoisiin työtehtäviin kiinnitetään huomiota jo töiden suunnitteluvaiheessa. Paloittelua ei tehdä aina ominaisuus kerrallaan, vaan paloittelussa otetaan huomioon ominaisuuden toteuttamisen iteraatiota pidempi aika ja ominaisuuden toteuttaminen paloitellaan tarvittaessa osiin.

Tavoite: Työt saadaan paloiteltua sen kokoisiksi tehtäviksi, että tehtävät pystytään toteuttamaan yhden iteraation aikana.

Sulautettujen järjestelmien suunnittelussa on mukana monen eri alan asiantuntijoita, jolloin kokonaiskuvan hallitseminen on tärkeää. Pienikin muutos voi aiheuttaa muutostarvetta moneen eri alueeseen. Tämän vuoksi suunnittelu lähteekin usein liikkeelle sulautetun järjestelmän kokonaisarkkitehtuurista. Vaikka kokonaisuutta hallitaankin, voidaan myös sen tekeminen pilkkoa pienempiin työtehtäviin: mitä kokonaiskuvasta tarvitaan juuri tässä vaiheessa projektia, mikä on oleellista, ja mikä osuus voidaan suunnitella tarkemmin vasta myöhemmin.

Asiakkaan vaatimus on usein hyvinkin korkealla tasolla. Ketterässä kehityksessä näistä asiakkaan vaatimuksista muodostetaan usein käyttäjätarinoita, jotka kertovat, miten asiakas haluaisi käyttää tuotetta. Sulautettujen järjestelmien kehittämisessä kehitysjonon kohdat eivät välttämättä ole käyttäjätarinoita kuvaten käyttäjän toimintaa, vaan voivat kuvata esimerkiksi sulautetun järjestelmän toiminnallisuutta. Joka tapauksessa näiden kehitysjonon kohtien tulee olla sen kokoisia, että ne kyetään toteuttamaan yhden iteraation (s. 49) aikana. Tyypillisesti kehitysjonon kohta jaetaan useammaksi tehtäväksi, ja käytännössä yhden tehtävän toteuttaminen on yhden henkilön tehtävänä yhden iteraation aikana. Näin useampi kehittäjä osallistuu saman kehitysjonon kohdan toteutukseen. Koska tehtävät liittyvät samaan ominaisuuteen, ovat ne usein hyvin riippuvaisia toisistaan.

Laitteiston suunnittelussa yhden ominaisuuden toteuttaminen vie pitkään, joskus jopa koko projektin ajan. Tällöin yhden iteraation kehitysjonon kohdaksi ei ole mahdollista ottaa koko ominaisuuden toteuttamista. Ominaisuuden toteuttaminen voidaankin jakaa osiin – karkeasti määriteltynä esimerkiksi piirikaavion suunnittelu on yhtenä kohtana, piirikuvion asettelu toisena kohtana, valmistus mahdollisesti kolmantena kohtana ja lopullisen piirin testaus neljäntenä kohtana. Nämä kaikki kehitysjonon kohdat voidaan ajoittaa eri iteraatioihin siten, mikä on toteuttamisen kannalta järkevää.

Ketterässä ohjelmistokehityksessä voidaan toteuttaa asiakkaan vaatimukset yksi ominaisuus kerrallaan. Näin pystytään priorisoimaan tärkeimmät ominaisuudet ja toteuttamaan ne valmiiksi asti kerrallaan. Samankaltaiseen toteutukseen voidaan laitteistopuolella pyrkiä modulaarisella suunnittelulla (s. 65). Näin voidaan hallita pienempiä kokonaisuuksia, priorisoida moduuleja ja helpottaa uudelleenkäyttöä. Modulaarisuuden lisäksi voidaan käyttää esimerkiksi emulaattoreita ja valmiita piirikortteja, jolloin voidaan edetä toteutuksessa nopeammin, vaikka lopullista tuotetta ei vielä pystytäkään valmistamaan.

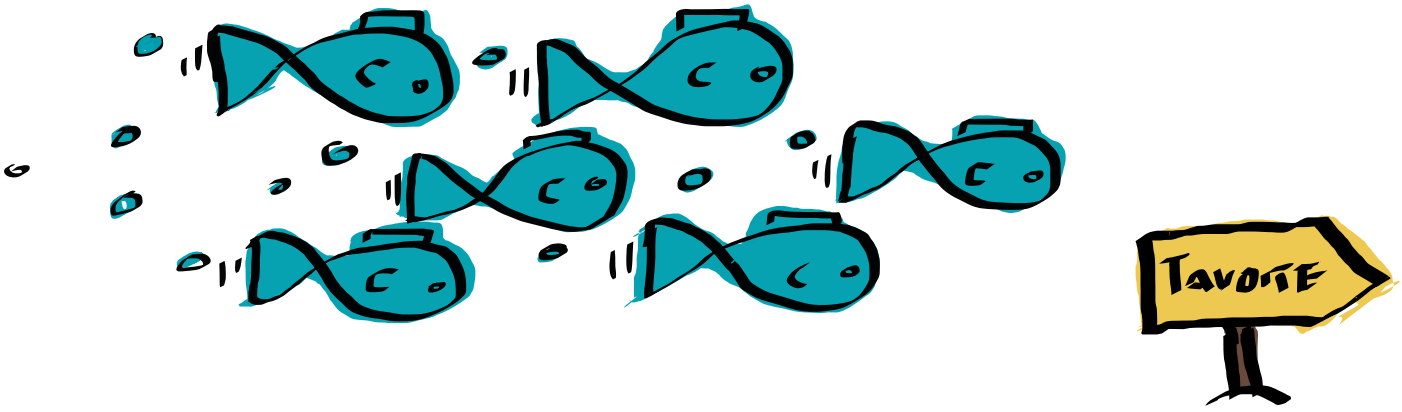
Jakamalla ominaisuuksien toteuttamista osiin saadaan tehtävistä helpommin hallittavia ja seurattavia. Näin ymmärretään paremmin, missä kohtaa suunnittelutyö etenee ja siihen voidaan tehdä muutoksia asiakkaan toiveiden mukaisesti. Tehtäviä suunniteltaessa tulee otettua huomioon myös tehtävien väliset riippuvuussuhteet, kuten se, mitä ohjelmistosta tarvitaan laitteiston testaukseen ja päinvastoin.

Tehtävien mukauttaminen iteraatioihin tekee työstä läpinäkyvämpää ja täsmentää työmäärän arviointia. Näin myös tiimi- ja työntekijäkohtainen työkuorma pysyy tasaisena, epävarmuus hallittavana ja kestävä tuottavuus turvataan. Huomioitavaa on se, että töiden paloittelu ja työmäärän arvioinnin tarkkuus iteraatiolle kehittyvät kokemuksen myötä. Ilman iteraation aikana valmiiksi saatavaa osakokonaisuutta tiimin työskentely on haastavaa. Osatavoitteet antavat työhön aikaan saamisen kokemuksen ja lisäävät työn mielekkyyttä, kun tavoitteet ovat selkeitä ja mielessä pysyviä.

Esimerkki

Asiakasvaatimus 1:	Haluan laitteessa olevan WLANin.
Kehitysjonon kohta 1.1:	WLAN käynnistyy ja se havaitaan.
Tehtävä 1.1.1:	Varmista WLAN –komponentin kytkennät ja niiden toimivuus.
Tehtävä 1.1.2:	Ohjelmoi ajuri, jonka kautta WLANin päälläolo havaitaan.
Tehtävä 1.1.3:	Testaa WLANin käynnistyminen ja havaitseminen.
Kehitysjonon kohta 1.2:	…
Tehtävä 1.2.1:	…
…	…

Itseorganisoituva tiimi



Ongelma: Tiimi ei pääse itse vaikuttamaan työnsä sisältöön tai työmääräänsä. Tämä aiheuttaa haasteita mm. sovituissa aikatauluissa pysymisessä. Tiimi toteuttaa valmiit tekniset määritelmät ottamatta kantaa toteutustapaan.

Ratkaisu: Tiimi organisoi itse sisäisesti työnsä ja määrittelee tekniset ratkaisut asiakkaan vaatimusten toteuttamiseksi. Tiimissä kierrätetään töitä niin osaamisensiirron kuin työmäärien tasaamisen vuoksi.

Tavoite: Tiimi voi itse päättää miten ja kuka tiimistä toteuttaa asiakkaan vaatimukset. Tiimi saa palautetta ratkaisuistaan ja pystyy näin parantamaan lopputulosta asiakkaan hyväksi.

Itseorganisoituvalla tiimillä on valtaa toteuttaa asiakkaan vaatimukset parhaaksi näkemällään tavalla. Vaatimukset järjestetään asiakkaan kanssa sovittujen prioriteettien mukaisesti ja toteuttamisen tekniset ratkaisut jäävät tiimin päätettäväksi. Tiimi ottaa huomioon teknisten vaatimusten lisäksi myös muut, esimerkiksi taloudelliset, rajoittavat tekijät. Kun tiimi työstää vaatimuksia yhdessä asiakkaan kanssa, tiimi pystyy vaikuttamaan myös vaatimusten toteuttamiseen tarvittavan työmäärän arviointiin.

Itseorganisoituva tiimi päättää itse miten toteuttaa vaatimukset ja kuka on vastuussa mistäkin osuudesta. Näin tiimi kykenee tarjoamaan parhaan mahdollisen osaamisensa ja hyödyntämään yksilöiden vahvuuksia. Toisaalta suunnitellessaan ja jakaessaan yhdessä toteutusta tiimin jäsenet oppivat uusia asioita toisiltaan ja osaaminen siirtyy. Itseorganisoituva työskentely on asiantuntijatyötä tekevälle motivoiva tapa työskennellä sillä siinä hän pääsee käyttämään kaikkea osaamistaan ja työskentelemään ryhmässä luovasti yhteisen tavoitteen eteen.

Tiimin tärkein prioriteetti on toteuttaa asiakkaan vaatimukset asiakkaan parhaaksi, ja tämä toteutetaan tiiviissä yhteistyössä asiakkaan kanssa. Asiakkaalta pyydetään palautetta ja palautteen kautta tuotteesta saadaan entistä toimivampi. Koska tiimi on oman työnsä paras asiantuntija, tiimi arvioi itse kuinka suuri on kunkin vaatimuksen toteuttamiseen kuluva työmäärä. Näin aikataulut saadaan pohjaamaan paremmin todellisuuteen. Vaatimusten toteutusten työmäärää seurataan, ja tätä kautta tiimi oppii paremmin arvioimaan uusien vaatimusten työmääriä.

Sulautettujen järjestelmien suunnittelussa tiimiläiset ovat usein oman alueensa asiantuntijoita, eivätkä välttämättä kykene tekemään täysin toistensa työtä. Vaikka tämä jakaakin automaattisesti tehtäviä tietyille henkilöille, itseorganisoituvassa tiimissä on mahdollista oppia ja opetella uusia asioita muiden tukemana.

Hyvin toimivassa itseorganisoituvassa tiimissä ei ole aseman luomia hierarkioita, vaan kaikilla on yhtäläinen mahdollisuus vaikuttaa toteutukseen ja työskentelytapoihin. Toteutusta seurataan ja vastuuta kannetaan yhdessä. Ketterälle tiimille on ominaista, että ketteriä menetelmiä ei sovelleta käyttöönoton jälkeen suoraan oppikirjan mukaan, vaan käytäntöjä tiimille sopivaksi soveltaen. Yhteinen vastuu vähentää yksilön kokemaa vastuun ja epäonnistumisen kuormittavuutta sekä vaikutusmahdollisuuksien myötä parantaa työssä oppimista. Kaikki tämä parantaa tiimin työtä ja tuottavuutta. Myös onnistumiset ovat yhteisiä, mikä lisää työn mielekkyyttä.

Joissakin tapauksissa itseorganisoituvuus ei ole tuottanut toivottuja tuloksia. Kun työntekijät ovat saaneet keskenään päättää mitä tekevät on syntynyt kuppikuntia sekä epätervettä työn siirtelyä, kun on töitä, joita kukaan ei halua tehdä. Tärkeitä asioita itseohjautuvassa tiimissä ovat jäsenten tasavertainen kohtelu, osaamisen arvostus, kuuntelu ja kyky rakentaviin konflikteihin.

Itseorganisoituvan tiimin johtaminen

Vaikka itseorganisoituvassa tiimissä ei ole hierarkiaa, esimiesten ja johdon rooli on tärkeä. Heidän tehtävänään on tukea tiimiä varmistamalla, että tiimistä löytyy tarvittava tietotaito toteutukseen. Esimiehen tehtävänä on myös auttaa tiimiä ratkomaan ongelmia ja poistamaan tiimin toiminnassa havaittuja sisäisiä ja ulkoisia esteitä. On myös tärkeää, että esimiehellä on riittävästi aikaa tiimille. Tiimi tarvitsee tukea myös asiakasvaatimusten hallintaan ja priorisointiin – esimerkiksi esimiehen taholta. Ei kuitenkaan ole välttämättä hyvä ratkaisu, että esimies osallistuisi tekniseen toteutukseen. Esimies toimii myös informaation jakajana tiimin ja tiimin ulkopuolisten välillä.

Siirtyminen itseorganisoituvaan työskentelyyn vaatii aikaa ja yhteisiä keskusteluita toiminnan periaatteista sekä henkilön, joka ottaa vastuun tiimin toiminnan ohjaamisesta.

Lisää aiheesta

Huusko, L.: Työpaikkana tiimi – *Miten tiimi kasvaa vastuuseen?*

Fasilitoija tiimityötä helpottamaan

Tiimissä kannattaa valita fasilitoija, jonka tehtävänä on varmistaa, että tiimissä noudatetaan yhdessä sovittuja toimintatapoja. Fasilitoijaksi kannattaa valita ketteristä toimintatavoista kiinnostunut henkilö, joka tukee näin tiimin sisältä sekä ketterien toimintatapojen käyttöönottoa että ketterän filosofian toteutumista. Tiimin sisältä valitulla fasilitoijalla on kokonaiskuva tiimin tilanteesta.

Fasilitoijan tehtävänä on helpottaa ketterien toimintatapojen käyttöönottoa ja huolehtia siitä, että sovituista käytännöistä pidetään kiinni. Hän huolehtii myös tarvittaessa palaverien ja muiden käytäntöjen järjestelyistä. Fasilitoijan rooliin voi kuulua myös esteiden poistaminen tiimin tieltä, kuten tiimin työrauhan turvaaminen. Joissakin tilanteissa fasilitoija toimii tiimin kontaktihenkilönä tiimin ulkopuolelle ja kommunikoi erityisesti tuoteomistajan (s. 53) kanssa.

Itseorganisoituvassa tiimissä fasilitoijan rooli voi myös olla kiertävä, jolloin jokainen on vuorollaan vastuussa toimintatapojen toimivuudesta ja niiden kehittämisestä yhdessä muun tiimin kanssa. Kypsässä tiimissä tiimiläiset itse hoitavatkin useita fasilitoijan rooliin kuuluvia tehtäviä.

Fasilitoijan roolia ei tule yhdistää rooliin, jonka kanssa se on ristiriidassa. Tällaisia ovat esimerkiksi esimiehen rooli auktoriteettisuhteen takia tai tuoteomistajan rooli, sillä tuoteomistajan rooli on tuottaa tiimille vaatimuksia. Lisäksi on tärkeää, että fasilitoijalla on riittävästi aikaa tiimille.

Tuoteomistaja

Ongelma: Asiakasrajapinta ja kehitettävään tuotteeseen liittyvä päävastuu on epämääräinen erilaisissa yrityksen projekteissa.

Ratkaisu: Tiimille valitaan täyspäiväinen tuoteomistaja, joka on päävastuussa tuotteesta, sen kehitysjonon hallitsemisesta ja sidosryhmien kanssa kommunikoinnista.

Tavoite: Kokopäiväisenä henkilönä tuoteomistaja voi pitää huolen vastuistaan ja toimia linkkinä tiimin ja ulkomaailman välillä.

Tuoteomistaja on yksi yleensä täysipäiväinen henkilö, joka on vastuussa tuotteen arvon maksimoinnista ja kehitystiimin työstä. Tuoteomistajan käytännön työtehtäviin kuuluu näin erityisesti tuotteen kehitysjonon (s. 45) hallinnointi ja eri sidosryhmien kanssa kommunikointi. Scrum-menetelmässä (s. 4) tuoteomistaja on yksi kolmesta nimetystä roolista.

Tuotteen kehitysjonon hallintaan kuuluu etenkin kehitysjonon kohtien tarkoituksenmukaisuuden ja selkeyden varmistaminen sekä niiden priorisointi tuotteen arvon maksimoinnin näkökulmasta. Itseorganisoituvuutta (s. 51) silmällä pitäen kehitystiimi voi itsekin huolehtia monista kehitysjonoon liittyvistä tehtävistä, mutta päävastuu kehitysjonosta säilyy silti tuoteomistajalla. On kuitenkin tärkeää, että tuoteomistaja keskustelee tuotteen kehitysjonosta tiimin kanssa. Tätä keskustelua käydään luontevasti iteraation suunnittelupalavereissa (s. 37).

Tuoteomistajalla on hyvä olla erilaisia projektinhallinnan taitoja ja tarpeeksi teknistä tietämystä tuotetta koskevien päätösten arvioimiseksi. Usein tuoteomistajaksi valikoituu aiemmin projektipäällikön tehtäviä suorittanut henkilö. Tällöin täytyy pitää huolta, että henkilö on sisäistänyt perinteisen projektipäällikön ja tuoteomistajan roolien väliset erot: ketterässä tuotekehitysprosessissa tiimi on itse vetovastuussa tekemistään sen sijaan, että heitä ohjattaisiin ylhäältä päin. Muutenkin hyvä ketteryyden tuntemus on tuoteomistajalle välttämätöntä.

Yleisesti tuoteomistajalla tulisi olla selkeä visio kehitettävästä tuotteesta, minkä vuoksi tuoteomistaja on luonnollinen vastuullinen tuotteen kehitysjonon pääasialliseksi ylläpitäjäksi. Tuoteomistajan on hyvä kehittää tuotevisiota yhdessä asiakkaan kanssa. Pidemmän tähtäimen tuotevisio lisää näkyvyyttä yli yhden iteraation.

Usein sulautettuja järjestelmiä tehdään massamarkkinoille, jolloin asiakas ei välttämättä ole konkreettinen. Tällöin asiakasnäkökulma on yrityksen sisällä ja tuoteomistajan kommunikaatio yrityksen muihin ulkoisiin ja sisäisiin sidosryhmiin on tärkeää. Toisaalta konkreettisen asiakkaan tapauksessa ollaan usein tilanteessa, jossa asiakas ei pysty osallistumaan esimerkiksi iteraation katselmointipalavereihin (s. 39). Tällöin tuoteomistaja voi toimia asiakkaan edustajana. Vastavuoroisesti asiakkaan kanssa kommunikoidessaan tuoteomistaja edustaa tiimiään.

Tuoteomistajan ja tiimin välinen toimiva yhteistyö ja selkeä työnjako ovat tärkeitä. Osuvat työmääräarviot ja toimiva tuotteen kehitysjono edistävät tätä. Läpinäkyvyys ja nopea palautesykli vaativat nopeaa reagointia tuoteomistajalta ja tiimiltä.

Resepti

1. Listaa sopivia kandidaatteja tuoteomistajaksi yrityksestä ja sen ulkopuolelta. Kriteereitä ovat etenkin tekninen tietämys kehitettävästä tuotteesta, projektinhallinnalliset ja kaupalliset taidot sekä hyvä ketteryyden tunteminen. Tuoteomistajan rooli on haastava edellyttäessään sekä teknistä että liiketoiminnallista osaamista.
2. Valitse sopiva henkilö. On tärkeää, että tuoteomistajalla on sekä aikaa että oikeus päättää tuotteen vaatimuksista.
3. Tuoteomistajan tulee jatkuvasti seurata tuotteen kehitystä ja osallistua ketteriin tilaisuuksiin. Näistä tärkeitä ovat etenkin suunnittelu- ja katselmointipalaverit:

- Iteraation suunnittelupalaverissa tuoteomistaja on vastuussa, että tiimiläisillä on tarpeeksi ymmärrystä kehitysjonon tärkeimmistä kohdista

- Katselmointipalaverissa tuoteomistaja päättää joko yksin tai yhdessä asiakkaan kanssa kehitysjonon kohtien sulkemisesta.

Voiko asiakas itse toimia tuoteomistajana?

Asiakasta voidaan harkita tuoteomistajan rooliin joissain erityistilanteissa, mutta useimmiten tämä ei ole suositeltavaa. Tuoteomistajan tehtävä vaatii teknisen osaamisen lisäksi laaja-alaisesti muita taitoja, jotta tuotteen kehitysjonon rakentaminen ja priorisointi olisi mahdollista. Lisäksi täyspäiväinen osallistuminen on suositeltavaa. Asiakkaalla ei tyypillisesti ole optimaalisinta näkökulmaa tähän tehtävään.

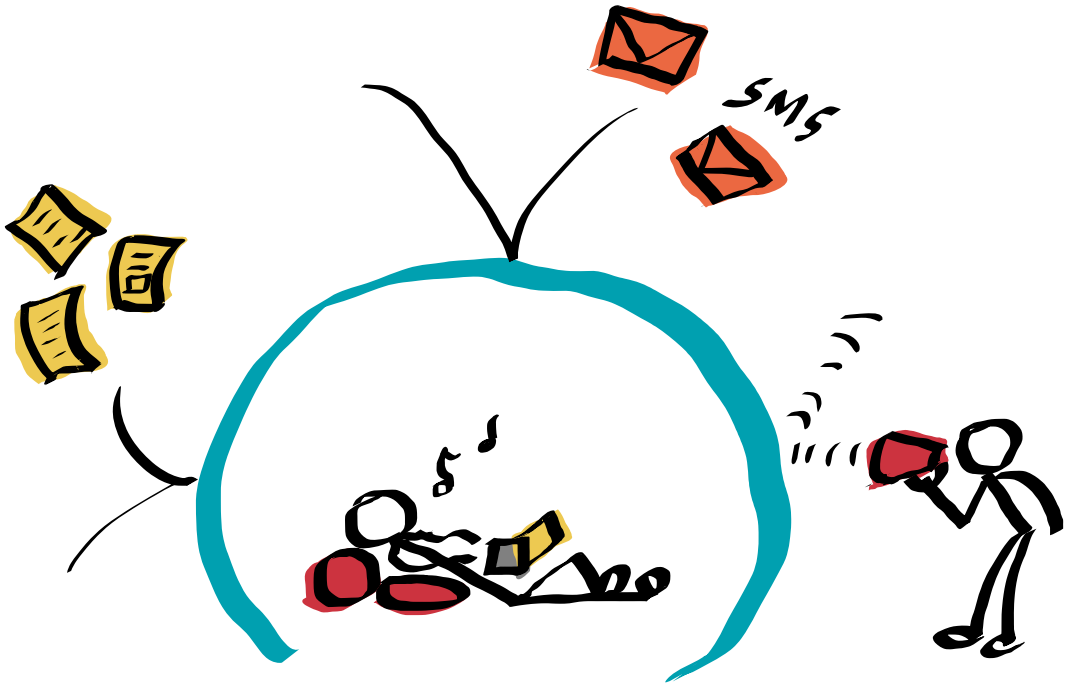
Lisää aiheesta

Clifford J.: *Transitioning to Scrum: Selecting the Product Owner*. http://bit.ly/1ISgVTk

McConnell P.: *Your Client Isn't Your Product Owner*. http://bit.ly/WH32vd

Schwaber K., Sutherland J.: *The Scrum Guide* (s. 5). http://bit.ly/ZgrC7Z

Keskeytysten hallinta



Ongelma: Tiimin ulkopuolelta tulevat pyynnöt, ohjeet ja kysymykset keskeyttävät työskentelyn ja rikkovat työrauhan.

Ratkaisu: Sovitaan yhdessä protokolla, jonka mukaisesti toisten työskentely keskeytetään ja keskeytykset käsitellään.

Tavoite: Työntekijät saavat työrauhan keskittyä tehokkaasti tekemiseensä ilman, että ulkoiset häiriöt jatkuvasti katkaisevat työskentelyn.

Hektisessä tuotekehitysympäristössä täydellisen työrauhan saaminen on haastavaa ja mm. vanhoja projekteja koskevat korjaus- ja ylläpitopyynnöt keskeyttävät työskentelyn säännöllisesti. Työn alla olevan projektin ulkopuolelta tulevat pyynnöt vaikeuttavat oman työn organisointia, koska ilmaantuvien lisätehtävien määrää ei pystytä arvioimaan eikä suunniteltujen työtehtävien tekemiseen käytössä oleva aika ole täysin ennakoitavissa. Yhteinen käytäntö keskeytysten hallintaan ja käsittelyyn helpottaa työn organisointia sekä tehostaa työn edistymistä kokonaisuutena. Keskeytysten hallintaa mietittäessä on syytä muistaa, että ulkopuolelta tulevat häiriöt ovat usein kokonaisuuden kannalta yhtä oleellisia kuin varsinainen työn alla oleva projekti. Paras lopputulos saavutetaan löytämällä tasapaino työn alla olevan projektin ja lisätehtävien hoidon välillä.

Keskeytysten hallinta on oleellinen osa ketteryyden toteuttamista sekä työskentelyn pitämistä mielekkäänä ja sujuvana sekä tuottoisana. Liian jäykät toimintatavat kuitenkin vievät mahdollisuuden nopeaan reagointiin, helppoon muutokseen ja luontaisten muutostarpeiden kohtaamiseen.

Keskeytysten luokittelu

Yksi tapa keskeytysten hallintaan on sopia yhteinen luokitteluasteikko projektin ulkopuolelta tuleville keskeytyksille. Luokittelussa määritellään keskeytyksille kategoriat esimerkiksi niiden toteuttamiseen kuluvan ajan ja tuloksen kriittisyyden perusteella. Kategoriasta riippuen tehtävä voidaan ottaa työn alle heti, se voidaan lisätä kehitysjonoon tehtäväksi esimerkiksi seuraavan viikon aikana tai se voidaan jättää odottamaan sopivaa hetkeä. Tärkeää on, että keskeytyksiin kuluva aika huomioidaan työmääräarvioissa, esimerkiksi iteraation kehitysjonossa (s. 45).

Keskeytysten tuottajien ohjeistaminen

Miten tahansa keskeytysten käsittely toteutetaan, on tärkeää muistaa kertoa käytännöistä laajasti kaikille potentiaalisille keskeytysten tuottajille. Ohjeistuksesta on hyvä selvitä mahdollinen luokittelu, joka vaikuttaa tehtävän valmistumisaikatauluun. Siitä on hyvä myös selvitä toiveet tehtävänannon sisällöstä, jotta tehtävän tekijä pääsee mahdollisimman tehokkaasti toteuttamaan annettua tehtävää. Keskeytysten tuottajien on myös tiedettävä, kehen ottaa yhteyttä. Vastaanottajan rooli voi olla pysyvä tai kiertävä.

Resepti

1. Kartoita, kuinka paljon projektiin osallistuvat joutuvat keskeyttämään projektin parissa työskentelyään ulkopuolelta tulevien keskeytysten johdosta.
2. Määrittele työtehtävät niin, että ulkopuolelta tulevien pyyntöjen hoitamiseen keskittyvät tehtävät määritellään selkeästi ja organisaatiolla on yhdessä sovitut käytännöt keskeytysten hoitamiseksi.
3. Tiedota tiimin soveltamista käytännöistä laajasti, jotta keskeytysten tuottajat osaavat toimia niiden mukaisesti.

Ylläpitotunti

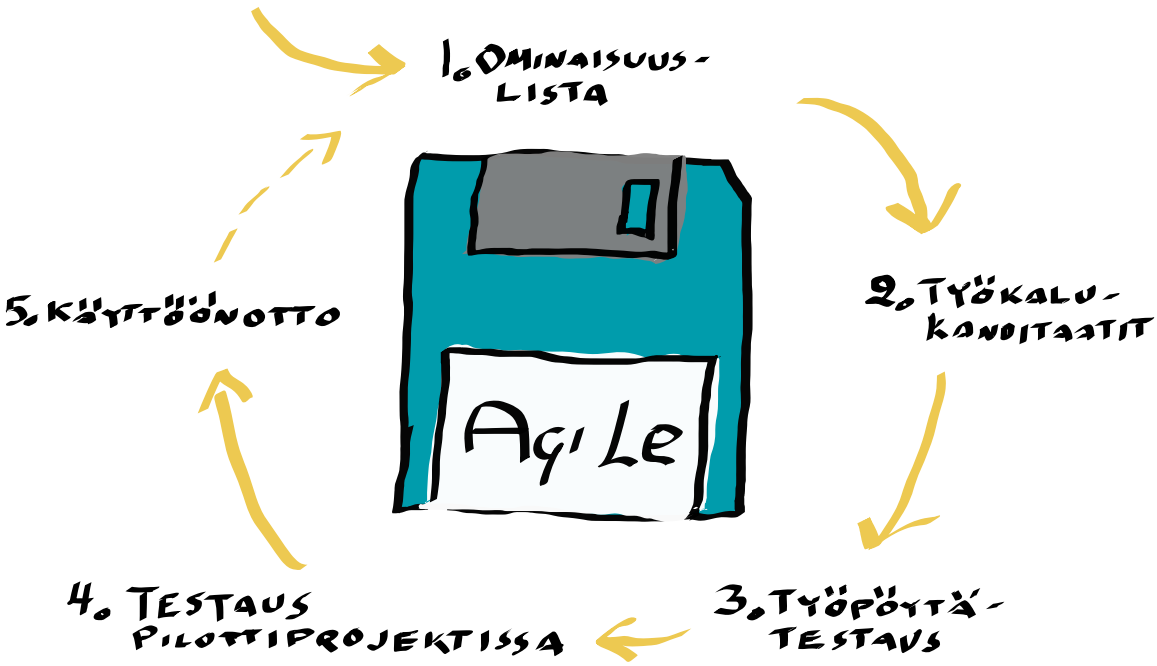
Tehtävissä, joihin liittyy huomattavia määriä vanhoihin projekteihin liittyviä tehtäviä ja joissa nämä tehtävät ilmaantuvat satunnaisesti pitkin käynnissä olevaa projektia, voi olla tarpeen ottaa käyttöön ylläpitotunti. Ylläpitotunti on kiinteä aika, joka on säännöllisesti, esim. päivittäin tai viikoittain, varattu työntekijöiden kalenterista eikä sille ole ennalta suunniteltu sisältöä. Ylläpitotunti on varattu vain ja ainoastaan ilmaantuneiden, vanhoja projekteja koskevien tehtävien hoitamiseen. Jos tällaisia tehtäviä ei ole jonain päivänä, jatketaan ylläpitotunnin aikana kuitenkin normaalia tuotekehitystyötä.

Käytännössä koettua

Eräässä organisaatiossa tuotekehitystiimin ulkopuolelta tulevat pyynnöt ja lisätarpeen ilmaukset tulivat kontrolloimattomasti puhelimen, pikaviestiohjelman, sähköpostin ja ruokapöytäkeskustelujen kautta satunnaisesti kaikille tiimin jäsenille. Tämä aiheutti paljon lisäselvitysten tarvetta, jatkuvaa työn ja työnjaon uudelleen määrittelyä sekä vei aikaa yhteisiltä kokouksilta. Työskentely oli kyllä ketterää, mutta arvaamatonta ja siten työntekijöitä kuormittavaa.

Ratkaisuna oli se, että pyyntöjen vastaanotto toteutettiin keskitetysti. Keskeytykset käsitteli yksi henkilö. Näin muut saivat työrauhan. Tiimin työtekijöille määriteltiin häiriötön päivittäinen työskentelyaika (klo 9-15), jolloin oli oman tarpeen mukaan mahdollisuus sulkea eri viestintäkanavat. Tuotteeseen pyydetty uusi piirre meni normaaliin suunnitteluun ja päätyi seuraavan jakson kehitysjonoon hyväksi katsotuin prioriteetein.

Ketterän työkaluohjelmiston valitseminen



Ongelma: Joko kehitysjonon ylläpito ja työtehtävien seuranta on haastavaa yksinkertaisia apukeinoja käyttäen tai nykyinen projektinhallintatyökalu ei mukaudu ketterään kehitysmenetelmään

Ratkaisu: Valitaan ketterään sulautettujen järjestelmien kehitykseen soveltuva työkaluohjelmisto.

Tavoite: Mahdollistetaan sujuva kehitysjonon hallinta mukautetussa kehitysprosessissa, jossa laitteiston ja ohjelmiston työtehtävien välillä on monenlaisia riippuvuuksia.

Sopivan työkalun valitseminen tiimille ei ole triviaali tehtävä. Vaikka työkalun käyttötarpeet olisivatkin selvillä yleisesti, niin varsinaisia työkaluja tarkastellessa ei ole aina selvää, mitkä ominaisuudet työkalu oikeasti toteuttaa.

Ketterän kehitysprosessin hallitsemiseen on tarjolla useita erilaisia työkaluvaihtoehtoja (s. 26), joiden avulla ketteryyttä voidaan tukea. Useimmat näistä ratkaisuista ovat kuitenkin suunniteltuja ketterään ohjelmistokehitykseen, jonka vuoksi laitteiston ja ohjelmiston toisistaan riippuvien työtehtävien hallitseminen saattaa vaatia joko luovuutta tai muokkauksia työkaluun. Se, miten työtehtävien riippuvuuksia tulisi lopulta hoitaa, on tiimin ja organisaation itse ratkaistava. Esimerkiksi useimmissa virheidenhallintajärjestelmissä voidaan tikettiin eli virheenkorjaustehtävään liittää linkkejä toisiin, kyseisen tiketin edistymiseen vaikuttaviin tiketteihin. Tämä ominaisuus on tuettuna myös monissa laajemmissa ketterissä työkalusovelluksissa.

Laitteiston ja ohjelmiston toisistaan riippuvien työtehtävien hallinnoimisessa vieläkin tärkeämmäksi osaksi voi muodostua visualisointi. Voi olla esimerkiksi suuri apu nähdä yhdellä silmäyksellä tilannekatsaus siitä, miten jokin vaatimus on

edennyt ohjelmiston suhteen laitteistokehitykseen verrattuna. Ongelmana on se, ettei tällaisia ominaisuuksia ole valmiina työkaluissa. Taulukkolaskentaohjelmistoa tai itse tehtyjä liitännäisiä käyttäen halutunlaisen visualisoinnin muodostaminen on kuitenkin mahdollista.

Sopiva työkaluohjelmisto on niin riippuvainen sitä käyttävästä tiimistä ja kehitysprosessin luonteesta, että tarkkoja valintaohjeita on mahdoton muodostaa. Tärkein periaate onkin, että kehitystiimillä itsellään on merkittävin rooli työkalun valinnassa: ketteryyden periaatteiden mukaisesti itseorganisoituvalla tiimillä (s. 51) tulisi olla tieto siitä, mitkä ominaisuudet ovat välttämättömiä kehityksen kulun kannalta.

Kun päätös uuden työkalun tarpeesta on tehty, tulisi järjestelmällisesti selvittää kriittisimmät työkalulta vaaditut ominaisuudet sekä tiimin että organisaation näkökulmasta. Erilaisista lähteistä kerättyjen potentiaalisten työkaluehdokkaiden listaa voi olla huomattavasti helpompi karsia muiden ihmisten kokemusten perusteella kuin työkaluvalmistajien markkinointimateriaalin avulla. Kapealla listalla myös testattavien työkalujen määrä pysyy kurissa. Tätä listaa kannattaa myöhemmin parannella työkalujen testaamisesta saatujen kokemusten perusteella ja tarvittaessa laajentaa alkuperäisten työkalukandidaattien kokoelmaa.

Mikäli jokin testatuista työkaluista vaikuttaa pätevältä, se kannattaa ottaa oikeaan projektiin testikäyttöön varhaisessa vaiheessa. Aina sopivaa työkalua ei löydy, jolloin joudutaan palaamaan aikaisempiin vaiheisiin. Joissakin tilanteissa työkalutarpeet ovat niin omalaatuisia, että vaihtoehdoiksi jää valita jokin työkalu, joka täyttää suurimman osan vaatimuksista ja mahdollistaa laaja-alaisen muokkaamisen joko omilla tai kolmannen osapuolen liitännäisillä.

Kehitysjonon hallinnan sujuvuus on erittäin tärkeää muun ketteryyden toteutumisen kannalta. Jatkuvat työkalun käyttöön liittyvät ongelmat vievät aikaa paitsi yksittäiseltä työntekijältä, myös muulta tiimiltä, kun esimerkiksi suunnittelupalavereissa (s. 37) aikaa käytetään työtehtävien suunnittelun sijaan työkaluongelmien puimiseen. Samasta syystä kynnys työkalun käyttämiseen tulee olla mahdollisimman matala: mikäli ihmiset kokevat työkalun vaivalloiseksi tai turhaksi, suunnittelu- ja katselmointipalaverien (s. 39) hyöty vähenee merkittävästi.

Keskeiset periaatteet

1. Kehitystiimin tulee ohjata valintaprosessia.
2. Ketteryyden pilottivaiheessa kannattaa olla avoin uusille työkalukäytännöille, vaikka yleisesti työkalun tulisikin soveltua tuotekehitysprosessiin eikä toisinpäin.
3. Kehitysjonon sulava hallinta on merkittävä edellytys ketterän kehityksen sujuvuudelle.
4. Työkalun käytöstä pitää pyrkiä tekemään mahdollisimman esteetöntä.
5. Työkalut auttavat estimoinnin hallitsemisessa mutta eivät yksinään paranna estimaatteja – eivätkä ikinä tee niistä täydellisiä.

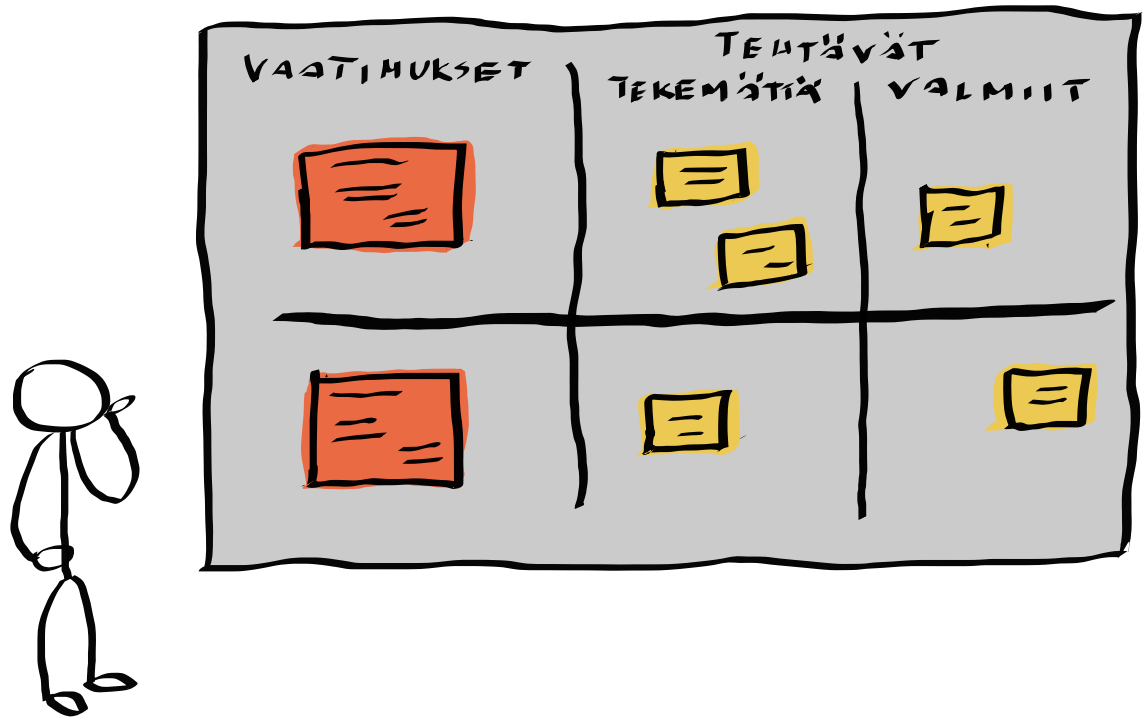
Käytännössä koettua

Yrityksessä oli tarve hallita toteutettavien ominaisuuksien edistymistä piirin eri lohkoissa. Koska työkalun valintaan ei ollut käytettävissä valtavasti resursseja, päädyttiin hallitsemaan kehitysjonoa itse tehdyllä taulukkolaskentapohjalla. Tämä mahdollisti kehityksen visualisoinnin ilman isoa panostusta valmiiden työkaluvaihtoehtojen kartoitukseen.

Lisää aiheesta

Suomi S.: *Project Management Tools in Agile Embedded Systems Development*.

Iteraation tehtävätaulu



Ongelma: Iteraation työtehtävät eivät ole kaikkien tuotteen kehitykseen osallistuvien näkyvissä ja kaikkien helposti muokattavissa.

Ratkaisu: Muodostetaan muutamasta sarakkeesta koostuva tehtävätaulu, jonka vasempaan laitaan kerätään iteraatiolle suunniteltuja kehitysjonon kohtia kuvaavia kortteja. Niiden oikealle puolelle sijoitetaan näistä pilkottuja tehtäväkortteja, joita kullekin tehtävälle vastuulliset kehittäjät siirtävät kehitystyön edetessä seuraavaan sarakkeeseen. Kortin sijainti vaakasuunnassa kuvaa tehtävän sen hetkistä tilaa.

Tavoite: Pidetään kirjaa tehdyistä ja tekemättömistä tehtävistä, mahdollistetaan kehityksen edistymisen tarkkailu pelkällä vilkaisulla ja muodostetaan luonnollinen paikka, jossa kehitystyöstä voidaan keskustella.

Tehtävätaulut ovat esimerkkejä helppokäyttöisistä, edullisista ja vuorovaikutukseen kannustavista fyysisistä ketteristä työkaluista. Erilaisia tehtävätauluja käytetään monissa eri yhteyksissä: esimerkiksi Kanban-menetelmän keskiössä on kanban-taulu (s. 8), jolla voidaan visuaalisesti rajoittaa käynnissä olevan työn määrää.

Iteraation tehtävätaulu on kätevä tapa sekä suunnitella että seurata iteraation työtehtävien edistymistä. Iteraation suunnittelupalaverissa (s. 37) kehitystiimi valitsee iteraation vaatimukset ja lisää nämä ensimmäiseen sarakkeeseen kehitysjonokortteina. Tiimi edelleen muodostaa niistä tekemättömien tehtävien sarakkeen tehtäväkortteja niin, että jokaisen kehitysjonon kohdan tehtävät ovat omalla rivillään. Iteraation käynnistyttyä kukin kehittäjä siirtää valitsemiaan kortteja eteenpäin tehtävien sen hetkistä tilaa kuvaamaan.

Sarakkeita voi olla useampia, ja niiden tulee kuvata tyypillisiä tiloja tehtävien etenemiselle. Erityisesti tehtäväkorttien siirtämiselle valmiit-sa-

rakkeeseen tulee olla sovittu yhteiset kriteerit, esimerkiksi testien läpäiseminen (s. 47). Kun yksittäisen kehitysjonon kohdan kaikki tehtäväkortit ovat valmiit-sarakkeessa, katsotaan vaatimuksen olevan valmis katselmointiin ja tuoteomistajan (s. 53) tai asiakkaan hyväksyttäväksi. Jokaisen kehittäjän tulee luonnollisesti myös huolehtia, että omat tehtäväkortit ovat ajan tasalla – muuten taulu ei toimi.

Kehitysjonokortti sisältää kyseessä olevan kehitysjonon kohdan kuvauksen, jonka avulla voidaan kortti päättää otettavaksi iteraatioon ja vastaavat tehtäväkortit muodostaa. Korttiin merkityn työmääräarvion avulla voidaan valita kuinka monta kehitysjonokorttia iteraatiossa kyetään toteuttamaan. Vastuullisen valitseminen kehitysjonokorttiin auttaa varmistamaan, että kaikki tehtävät tulevat tehtyä, vaikka tehtävien tekijöitä samassa vaatimuksessa olisi useita.

Tehtäväkortti sisältää tehtävän nimen ja kuvauksen. Kuvauksen tulee olla riittävän yksityiskohtainen, jotta kehittäjä ymmärtää tarkasti mitä hänen tulee toteuttaa. Tärkeä tieto on myös kyseiselle tehtävälle arvioitu työmäärä, joka on tyypillisesti merkitty tunteina. Kortin tietoja päivitetään tarvittaessa iteraation edetessä.

Tehtävätaulu tarjoaa tiimille yksinkertaisen tavan seurata työn edistymistä ja toimii lisäksi keskeisenä kokoontumispaikkana esimerkiksi säännöllisissä tilannepalavereissa (s. 43) sekä suunnittelu- ja katselmointipalaverissa (s. 39). Verrattuna yhtä lailla varsin nopeasti käyttöönotettavaan taulukkolaskinohjelmistossa ylläpidettävään iteraation kehitysjonoon, tehtävätaulu on näkyvämpi ja välittömästi kaikkien muokattavissa. Fyysisenä työkaluna se voi olla hyvä keskustelun herättäjä ja toimivampi ratkaisu kuin jokin sähköisistä vastineista. Aina fyysinen tehtävätaulu ei kuitenkaan ole vaihtoehto, esimerkiksi hajautetun tiimin tai seurannan erityisvaatimusten takia. Silloin vaihtoehtoina ovat muun muassa ketterät projektinhallintaohjelmistot ja niiden sähköiset tehtävätaulut (s. 26).

Resepti

1. Valitse tehtävätaululle keskeinen paikka seinältä, jonne kaikilla tiimiläisillä on vaivaton pääsy.
2. Tee taulu seinään jollain helpolla käytössä olevalla menetelmällä (esim. tussitaulu, maalarinteippi).
3. Täytä taulun ensimmäinen sarake suunnittelupalaverissa tuotteen kehitysjonon kohtia edustavilla korteilla. Muodosta samassa tilaisuudessa tekemättömät tehtävät vastaavien kehitysjonokorttien oikealle puolelle.
4. Iteraation edetessä jokaisen kortin tehtäväkseen ottanut siirtää tehtäväkortteja niiden tilaa kuvaaviin sarakkeisiin.
5. Kun vaatimuksen kaikki tehtävät ovat valmiit, on vaatimus valmis katselmointiin.

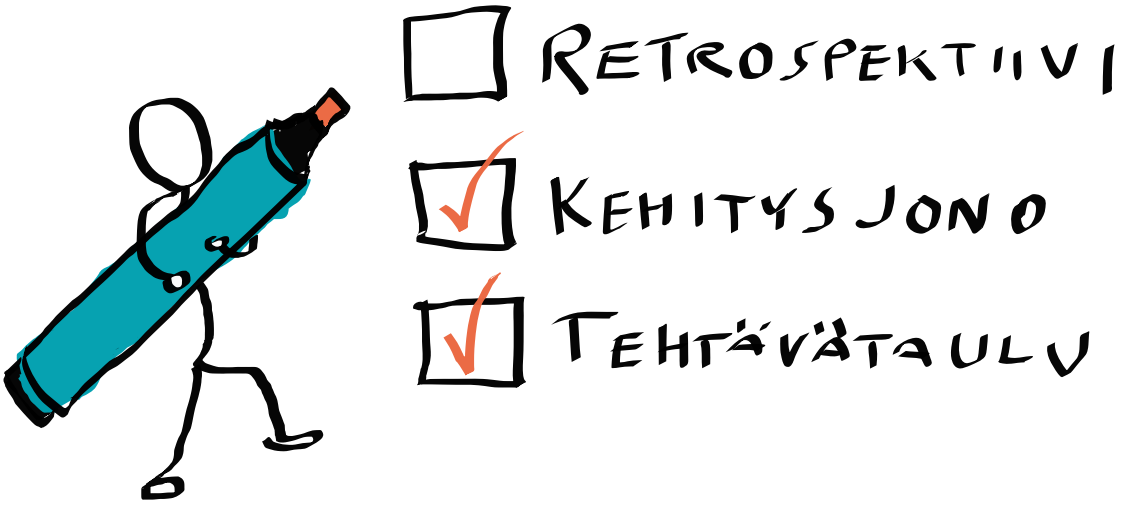
Käytännössä koettua

Eräässä yrityksessä tehtävätaulu toimi työn seuraamisen ja tiedonvälityksen kanavana sekä tiimin työntekijöiden keskuudessa että tiimistä ulospäin. Yhdellä vilkaisulla näki missä tuotteen kanssa oltiin menossa ja mikä asia oli kullakin työn alla. Taulu sijaitsi avokonttorissa keskeisellä paikalla ja jokainen havaitsi, kun työtoveri käveli päivittämään taulua. Usein tilanteissa syntyi keskustelua. Näkyvä paikka edisti myös päivittämisen muistamista. Informaatiota siirtyi hyvin, kun tehtävät oli pilkottu riittävän pieniksi ja kehitysprosessin todelliset vaiheet olivat näkyvissä taulun sarakkeissa.

Lisää aiheesta

Perry T.: Drifting Toward Invisibility: *The Transition to the Electronic Task Board*.

Ketteryyden tarkistuslista



Ongelma: Erityisesti käyttöönottovaiheessa ketteryys on tiimille liian abstrakti käsite. Erilaisia tekniikoita kokeillaan, mutta kenelläkään ei ole selkeää kuvaa siitä, mitkä tekniikat toimivat ja mitkä eivät.

Ratkaisu: Tiimi valitsee yhdessä ne tekniikat, joihin se haluaa sitoutua. Tekniikat kirjataan tarkistuslistaan. Joka retrospektiivin yhteydessä tiimi käy läpi, mitkä tekniikat ovat olleet todellisuudessa käytössä, miten ne ovat toimineet ja onko niitä tarpeen kehittää. Tarvittaessa tarkistuslistaa muokataan paremmin tiimin tarpeita vastaavaksi.

Tavoite: Tiimille syntyy yhteinen tekemisen tapa. Käytetyt tekniikat konkretisoituvat ja niiden käytön kehittäminen dokumentoidaan systemaattisesti.

Ketteryyden tarkistuslista on yksinkertainen työkalu, jolla konkretisoidaan ketteryyden käyttöönottoon liittyvät asiat ja seurataan käyttöönoton etenemistä. Tarkistuslistaan kirjataan otsikkotasolla valitut ketterät käytännöt ja tarvittaessa tarkennetaan lyhyesti, miten kukin käytäntö kehitystiimissä tulkitaan. Tarkistuslistan muodostaminen riippuu siitä, miten ketteryyttä yleisemmin otetaan organisaatiossa käyttöön. Parhaimmassa tilanteessa tiimi saa itse päättää, mitä käytäntöjä se lähtee ensimmäisessä ketterässä projektissaan kokeilemaan. Vaikka käytännöt osin tulisivatkin valmiiksi annettuna, tiimin on kuitenkin hyvä kirjoittaa tarkistuslista projektin alussa yhdessä, jotta varmistetaan, että valitut käytännöt tulkitaan tiimin sisällä samalla tavalla. Tarkistuslista voidaan toteuttaa hyvin yksinkertaisesti esim. PowerPoint-kalvona, tekstitiedostona tai tarkistuslistalle pyhitetyllä tussitaululla.

Vaikka käytäntöjen läpikäynti projektin aluksi tiimin kesken onkin suositeltavaa, täysi hyöty tarkistuslistasta saadaan, kun sitä päivitetään aktiivisesti projektin aikana. Luonnollisin ajankohta tähän on retrospektiivin (s. 41) yhteydessä, kun tiimin työmenetelmien toimivuutta muutenkin arvioidaan. Tarkistuslista käydään tiimin kesken läpi käytäntö käytännöltä pyrkien vastaamaan seuraaviin kysymyksiin:

Onko käytäntöä todella noudatettu edellisen tarkastelun jälkeen?

Millä tavoin käytäntöä on noudatettu?

Koetaanko käytäntö niin hyödylliseksi, että sen soveltamista jatketaan?

Millä toimenpiteillä varmistetaan käytännön noudattaminen?

Miten käytäntöä voitaisiin vielä parantaa entisestään?

Näin tarkistuslistan avulla varmistetaan, ettei tiimi pidä käytössä turhia käytäntöjä ja tarvittava tuki käytäntöjen noudattamiselle järjestetään. Tarkistuslistaan on toki mahdollista ottaa mukaan myös uusia käytäntöjä, mikäli tämä koetaan tarpeelliseksi. Aloitteleva ketterä tiimi voi käyttää tarkistuslistaa eräänlaisena toivelistana, johon listataan tiimin hyödyllisinä pitämät käytännöt. Toivelista priorisoidaan ja käytäntöjä otetaan käyttöön sopivissa paloissa projektin edetessä. Näin aloittelevan tiimin voi olla helpompi tutustua uusiin käytäntöihin. Toki samalla on hyvä muistaa, että ketterät käytännöt muodostavat kokonaisuuden, joten priorisointi on syytä tehdä ketteryyteen perehtyneen henkilön kanssa ja käytännöt ottaa käyttöön tai hylätä kohtuullisen ripeällä tahdilla.

Mikäli tarkistuslistan kukin versio säilytetään, projektin lopuksi saadaan kohtuullisen objektiivinen kuva siitä, miten ketteryys on projektin aikana kehittynyt, mitkä käytännöt ovat tiimin työn kannalta osoittautuneet mielekkäiksi ja mitkä on projektin kuluessa hylätty. Tätä tietoa voidaan hyödyntää seuraavissa projekteissa tai kun ketteryyden käyttöönottoa laajennetaan organisaatiossa uusiin tiimeihin.

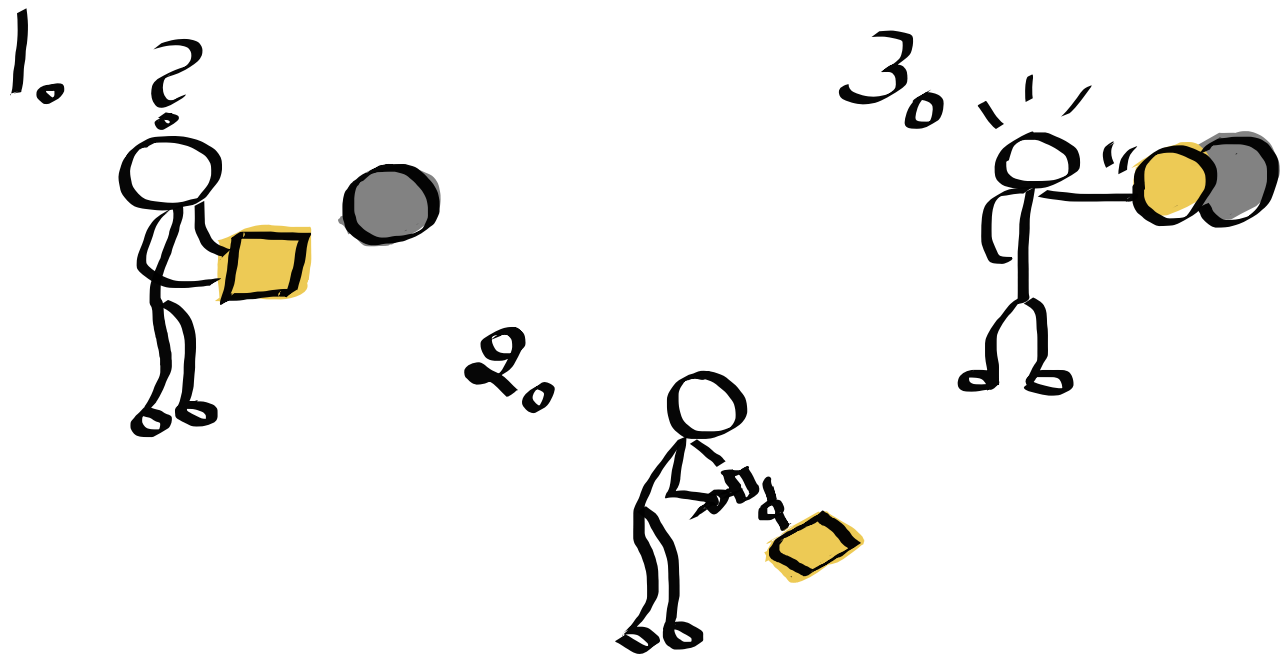
Resepti

1. Laatikaa tiimin kesken ensimmäinen versio tarkistuslistasta. Käyttäkää tähän riittävästi aikaa.
2. Päivittäkää tarkistuslista säännöllisesti esim. retrospektiivin yhteydessä. Tähän vaiheeseen kuluu yleensä 15–30 min.
3. Toteuttakaa päivityksen yhteydessä kirjatut toimenpiteet käytäntöjen käyttöönoton tukemiseksi.
4. Projektin lopuksi voidaan käydä läpi tarkistuslistan kehittyminen projektin aikana seuraavia projekteja varten.

Käytännössä koettua

Tarkistuslistan ideaa voi soveltaa tilanteen mukaan. Eräässä yrityksessä ei listattu kaikkia käytäntöjä, vaan retrospektiiveissä nostettiin esille keskeisimmiksi tai haastavimmiksi kootut ketterät käytännöt. Jokaisesta käytännöstä kirjattiin ylös edellisen iteraation ajalta hyvät ja huonot puolet sekä kehitysehdotukset seuraavaan iteraatioon. Seuraavalla kerralla tarkastelu toistettiin ja verrattiin tuloksia edellisen kerran kehitysehdotuksiin. Tätä toistettiin, kunnes keskeiset käytännöt todettiin tiimissä enimmäkseen toimiviksi.

Testivetoinen kehitys



Ongelma: Suunnitteluprosessissa tehtäviä virheitä ei havaita ajoissa jolloin tuotteen laatu, kustannusarviot ja aikataulut kärsivät.

Ratkaisu: Testivetoisessa kehityksessä ominaisuuksien kehittäminen aloitetaan aina määrittelemällä testit lopulliselle, valmiille ominaisuudelle. Kehityksen edetessä ominaisuuksia testataan aina näitä testejä vasten.

Tavoite: Vältetään projektin aikaiset harha-askeleet ja käytetään aika alusta asti tehokkaasti oikeiden asioiden tekemiseen. Varmistetaan uusien ominaisuuksien ja toteutusten oikeellisuus mahdollisimman aikaisessa vaiheessa.

Testivetoinen kehitys on paitsi yksi iteratiivisen ja inkrementaalisen kehityksen muoto, myös yksi XP:n käytännöistä (s. 6). Testivetoisessa kehityksessä suunniteltujen ominaisuuksien oikeellisuus pyritään varmistamaan automatisoiduilla testeillä, jotka määritellään vaatimusten perusteella ennen varsinaisen ominaisuuden toteutusta. Testivetoinen kehitys tähtää yksinkertaisiin toteutuksiin, jotka tekevät juuri sen, mitä niiden on suunniteltukin tekevän, eivätkä mitään muuta.

Testivetoisessa kehityksessä uuden ominaisuuden suunnittelu aloitetaan määrittelemällä tälle ominaisuudelle automatisoidut testit. Kehitettävä ominaisuus saattaa myös olla jonkin olemassa olevan ominaisuuden uusi kehitysversio. Laitteistoa suunniteltaessa suunniteltu testi ei välttämättä ole automatisoitu mutta sisältää keskeiset testit ja mittaukset, jotka toimivan laitteen ominaisuuden pitäisi toteuttaa.

Alkutilanteessa uudet testit luonnollisesti tuottavat virheilmoituksen tai muutoin virheellisen tuloksen. Tämä on itse asiassa tärkeää kokeilla heti alkuvaiheessa. Jos uusi testi läpäistään ilman testattavaan ominaisuuteen tehtyjä muutoksia, uusi testi on hyödytön, eikä se toimi tarkoituksessaan.

Testivetoinen kehitys on luonteeltaan iteratiivista. Jokainen iteraatio aloitetaan testien laadinnalla, jonka jälkeen kehitetään ensin toivottua uutta ominaisuutta, ja tämän jälkeen testataan ominaisuus alkuvaiheessa laadituilla testeillä. Tavoitteena alkuvaiheessa on läpäistä uudet testit mahdollisimman vähällä vaivalla. Vasta tämän jälkeen jatketaan ominaisuuden kehittämistä pidemmälle laatimalla testit, toteuttamalla ja testaamalla. Näin varmistetaan toteutuksen pysyminen mahdollisimman yksinkertaisena.

Testivetoisessa suunnittelussa yksittäisen ominaisuuden, jolle laaditaan oma testinsä, koko on suositeltavaa pitää mahdollisimman pienenä. Kokonaisvaltaisten testien suunnittelu pienelle yksikölle on yksinkertaisempaa ja virheiden paikantaminen niiden ilmetessä helpompaa. Lisäksi dokumentoinnin tarve on vähäisempi, koska pienten yksiköiden toiminta on helpommin käsitettävissä.

Resepti

1. Määrittele, minkälaisen ominaisuuden haluat.
2. Laadi ominaisuudelle testit. Varmista, että nykyinen järjestelmä ei läpäise testiä.
3. Laadi pienin mahdollinen toteutus, joka läpäisee testit.
4. Kehitä ominaisuutta eteenpäin kohti lopullista muotoaan.

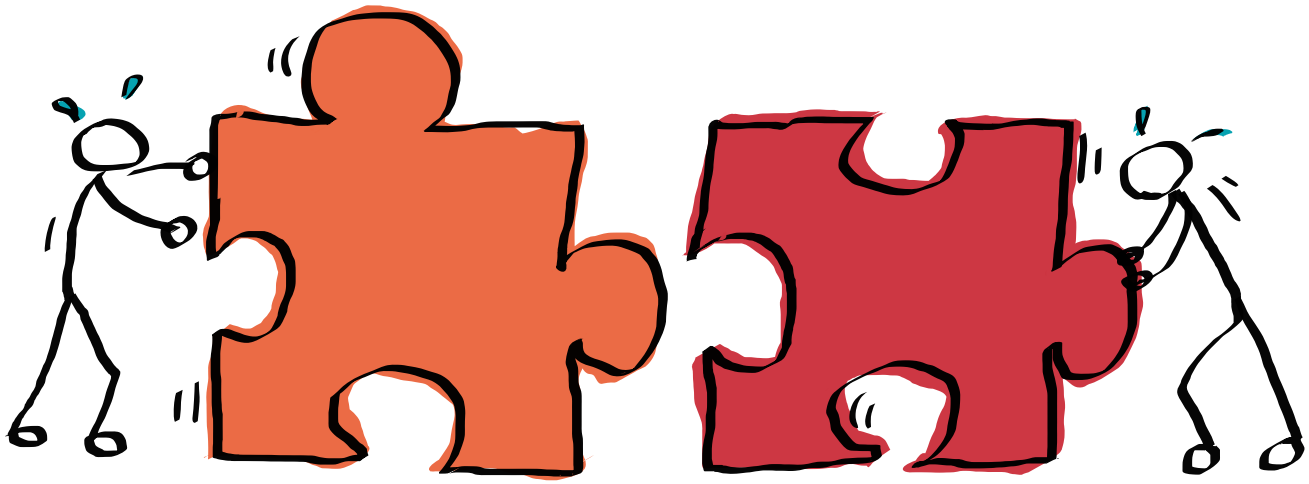
Testivetoisen suunnittelun nähdään tuovan monia hyötyjä sulautettujen järjestelmien ohjelmistojen kehitykseen. Testivetoinen kehitys ohjaa suunnittelua modulaarisempaan suuntaan, jolloin tuotosten yleiskäyttöisyys lisääntyy ja elinkaari pitenee. Lisäksi menetelmällä voidaan varmistaa, että kaikki laitteen ominaisuudet tulevat myös testatuksi, eivätkä ohjelmistoon päätyneet, sinne kuulumattomat ominaisuudet aiheuta laitteen toimintaan häiriöitä. Testivetoisella kehityksellä vähennetään näin turhan työn tekemistä ja parannetaan suunniteltavan järjestelmän laatua. Testivetoinen kehitys mahdollistaa myös aikaisen ja jatkuvan regressiotestauksen, jolla varmistetaan, että yhteen osaan tehdyt muutokset eivät aiheuta uusia virheitä ohjelmiston tai laitteiston muissa osissa. Testien kautta saadaan nopeasti varmuus muutoksen vaikutuksista ominaisuuden ja koko järjestelmän toimintaan, jolloin kynnys pienten muutosten tekemiselle madaltuu.

Tuotekehityksen alkuvaiheissa testivetoinen kehitys voi hyötyä kaksikohdesuunnittelusta (s. 67), jolloin lopullisen laiteympäristön omainen testaaminen saadaan aloitettua jo suunnittelun alkuvaiheissa.

Lisää aiheesta

Grenning J.: *Test Driven Development for Embedded C*.

Modulaarinen suunnittelu



Ongelma: Suunniteltavien laitteiden osien uudelleenkäyttö on vaikeaa ja testaaminen hankalaa ennen kuin koko laite on valmis.

Ratkaisu: Modulaarisessa suunnittelussa laite jaetaan useaan pienempään itsenäiseen osaan, joiden väliset rajapinnat on tarkasti määritelty.

Tavoite: Mahdollistetaan laitteen osien itsenäinen kehitys ja testaus sekä tehokas uudelleenkäyttö.

Nykyaikaiset sulautetut järjestelmät ovat laajoja ja monimutkaisia kokonaisuuksia, joiden kehittäminen vaatii paljon resursseja ja aikaa. Järjestelmien suunnittelu ja hallitseminen yhtenä kokonaisuutena vaatii tarkkaa ja huolellista suunnittelua. Näin suurten kokonaisuuksien hallitseminen ketterässä suunnitteluprosessissa on haasteellista, koska luonnolliset iteraatiot (s. 35) venyvät helposti turhan pitkiksi.

Modulaarinen suunnittelu ei ole käsitteenä uusi, eikä se ole varsinaisesti ketterä tuotekehitysmenetelmä. Modulaarisella suunnittelulla voidaan kuitenkin helpottaa ketterien periaatteiden soveltamista esimerkiksi laitteistosuunnittelussa.

Modulaarinen suunnittelu on tunnettua myös ohjelmistokehityksen puolella. Modulaarisessa ohjelmoinnissa ohjelmiston toiminnot erotetaan omiksi itsenäisiksi moduuleikseen. Ohjelmoinnissa modulaarisuudella parannetaan ohjelmiston ylläpidettävyyttä. Moduulit toteuttavat omat osansa kokonaisuudesta ja yhdistyvät toisiinsa tarkoin määriteltyjen rajapintojen kautta. Näin uusia moduuleja voidaan helposti lisätä, ja vanhoja moduuleja poistaa tai päivittää. Modulaarisella suunnittelulla parannetaan järjestelmän osien uudelleenkäytettävyyttä. Itsenäisesti määriteltyjen, toteutettujen ja dokumentoitujen osien uudelleenkäyttö on suoraviivaisempaa kuin osien irrottaminen suuresta järjestelmästä ja liittäminen osaksi toista järjestelmää.

Sulautettujen järjestelmien suunnittelussa modulaarinen lähestymistapa helpottaa ketterien käytäntöjen noudattamista. Moduulien kanssa toimiessa yksittäisten tehtävien koko saadaan pidettyä kohtuullisen pienenä, ja työn luonnolliset iteraatiot ovat sopivan mittaisia. Modulaarisuus helpottaa myös testaamista. Moduulien kehitysvaiheessa niiden toiminta saadaan testattua itsenäisesti. Tämän jälkeen voidaan siirtyä testaamaan niitä osana suurempaa järjestelmää, jolloin testaaminen painottuu lähinnä moduulien rajapintojen toimintaan.

Sulautettujen järjestelmien suunnittelussa modulaarisen suunnittelun pohjana voi toimia niin sanottu alustalähtöinen suunnittelu. Järjestelmien pohjana käytetään yleiskäyttöistä alustaa, joka sisältää järjestelmän pääarkkitehtuurin ja keskeisimmät ominaisuudet. Alustan päälle lisätään uusia ominaisuuksia, jotka osaltaan toteuttavat järjestelmän toiminnallisuuden.

Lisää aiheesta

Cordeiro L., Barreto R., Oliveira M.: T*owards a Semiformal Development Methodology for Embedded Systems*.

Alustalähtöinen suunnittelu

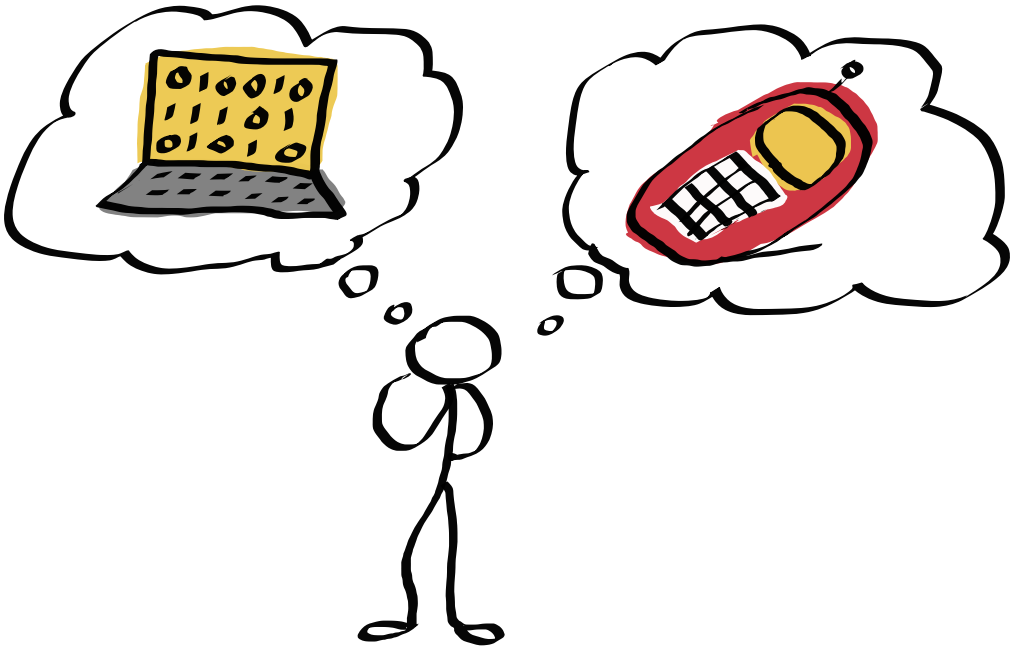
Alustalähtöistä suunnittelua voidaan pitää modulaarisen suunnittelun erikoistapauksena. Suunniteltaessa useita samankaltaisia laitteita, tuotekehitystiimille muodostuu käsitys järjestelmien arkkitehtuurin keskeisistä piirteistä ja ominaisuuksista. Tämän käsityksen pohjalta voidaan suunnitella alusta, joka toimii lähtökohtana tuleville uusille järjestelmille. Alusta sisältää laitteissa käytetyn arkkitehtuurin, keskeisimmät ominaisuudet sekä kattavat rajapinnat uusien ominaisuuksien lisäämiselle.

Kun yleiskäyttöinen alusta on suunniteltu, voidaan uuden järjestelmän tuotekehitys aloittaa nopeasti ja kustannustehokkaasti helposti muokattavan ja paljon toiminnallisuutta sisältävän alustan avulla.

Yleiskäyttöiseksi suunniteltuun alustaan saattaa kertyä paljon toiminnallisuutta, jota ei tarvita jokaisessa projektissa. Toisaalta, koska alustasta halutaan yleiskäyttöinen, kyseiset ominaisuudet kuitenkin halutaan sisällyttää alustaan, jolloin uusien tuotteiden suunnittelu alustan pohjalta tulee nopeammaksi ja halvemmaksi.

Laaja ominaisuuksien kirjo näkyy alustan koossa ja komponenttien määrässä, mikä rajoittaa sen käyttöä suurissa tuote-erissä. Pienissä sarjoissa, prototyypeissä sekä tuotteissa, joihin tarvitaan asiakaskohtaisia muokkauksia, alustalähtöinen suunnittelu voi tuoda selkeitä kustannussäästöjä uudelleenkäytettävyyden avulla. Yleisesti tällaiset alustat koostuvat ohjelmoitavasta prosessorista ja logiikasta, jonka ympärille kerätään paljon ominaisuuksia sekä liittimiä ja väyliä, joiden avulla alustaan voidaan helposti liittää eri toiminnallisuutta sekä moduuleja.

Kaksikohdesuunnittelu



Ongelma: Sulautetun järjestelmän ohjelmistokehitys joutuu odottamaan laitteistokehityksen etenemistä ennen kuin ohjelmistokehityksessä päästään etenemään.

Ratkaisu: Kaksikohdesuunnittelussa sulautetun järjestelmän ohjelmistosta kehitetään toiminnallisesti yhteneviä versioita yhtäaikaisesti sekä lopulliselle laitteelle että jollekin yleiskäyttöiselle alustalle.

Tavoite: Mahdollistetaan täysipainoinen ohjelmistokehitys ja testaus ennen kuin lopullinen laitteisto on valmis.

Sulautettujen järjestelmien kehitys on monen osa-alueen tiivistä yhteistyötä. Järjestelmän ohjelmisto-, laitteisto- ja mekaniikkaosat suunnitellaan tyypillisesti rinnakkaisissa prosesseissa yhtä aikaa toistensa kanssa. Nämä osa-alueet eivät kuitenkaan ole toisistaan riippumattomia, vaan yhdellä osa-alueella tehdyt ratkaisut vaikuttavat myös muihin alueisiin. Osa-alueiden väliset riippuvuussuhteet saattavat myös aiheuttaa tilanteita joissa jokin osa-alue ei pääse etenemään, koska se tarvitsee toisen osa-alueen tuloksia esimerkiksi testaustarkoituksiin.

Sulautetut järjestelmät muodostuvat tyypillisesti juuri kyseistä järjestelmää varten suunnitelluista ohjelmistosta, laitteistosta ja mahdollisesti mekaniikasta. Ohjelmiston testaamiseen tarvitaan laitteistoa ja laitteiston testaamiseen ohjelmistoa. Elektroniikan ja mekaniikan välillä vallitsee vastaava riippuvuussuhde. Ohjelmiston ja laitteiston välisestä suhteesta muodostuu pullonkaulaongelma, jossa kumpaakaan osa-aluetta ei päästä täysipainoisesti kehittämään ja ennen kaikkea testaamaan ennen kuin toinen osa-alueista on saavuttanut riittävän tason.

Kaksikohdesuunnittelu on yksi ratkaisu ohjelmiston ja laitteiston väliseen pullonkaulaongelmaan. Kaksikohdesuunnittelussa järjestelmän ohjelmistoa kehitetään yhtäaikaisesti sekä lopulliselle laitteistolle että yleiskäyttöiselle PC-tietokonealustalle. Käytännössä tähän tarvitaan kaksi kääntäjäohjelmistoa, toinen PC-tietokoneelle ja toinen kehitettävälle sulautetulle laitteelle. Yleis-

käyttöisellä tietokoneella voidaan käyttää erinäisiä simulaatioympäristöjä lopullisen laitteen mallintamiseen. Tällainen simulaatioympäristö muodostaa varsinaisen laitteen ohella toisen kaksikohdesuunnittelun kohteista. Laitteiston ominaisuuksia voidaan myös emuloida hyvinkin yksinkertaisilla ohjelmallisilla toteutuksilla, jotka esimerkiksi syöttävät ohjelmistolle laitteen sensorien keräämää tietoa.

Ohjelmiston suunnittelu sekä järjestelmän omalle laitteistolle että yleiskäyttöiselle PC-tietokoneelle helpottaa laitteistoriippuvuuksien ymmärrystä ja tekee usein ohjelmistosta yleiskäyttöisempää. Kahdelle eri alustalle suunniteltaessa työmäärä saattaa projektin alkuvaiheessa hieman kasvaa. Useimmiten parantuneen yleiskäyttöisyyden myötä vastaavasti ohjelmiston elinkaari pitenee ja kokonaistyömäärä vähenee ohjelmiston ollessa helpommin käytettävissä myös tulevissa laitteissa.

Resepti

1. Suunnittele ja kehitä sulautetun järjestelmän ohjelmistot yhtä aikaisesti sekä sulautetulle järjestelmälle että PC-tietokoneelle.
2. Kiinnitä huomiota ohjelmiston yleiskäyttöisyyteen ja pitkään elinkaareen.
3. Täydennä ohjelmiston aukkoja sitä mukaa kun sulautetun järjestelmän laitteiston suunnittelu etenee.

Kaksikohdesuunnittelu helpottaa laitteen ohjelmiston toimintojen monipuolista testaamista heti suunnitteluprosessin alusta alkaen. Sulautetulle järjestelmälle ja PC-tietokoneelle kehitettävät versiot pidetään toiminnallisesti yhtenevinä. Näin ominaisuuksia voidaan testata PC-tietokoneella ennen kun järjestelmälle kehitettävä laitteisto on tähän riittävällä tasolla.

Laitteistosuunnittelulle kaksikohdesuunnittelun hyödyt tulevat esiin laitteiston testaamisessa. Kaksikohdesuunnittelun myötä molemmat osa-alueet pääsevät etenemään suunnitteluprosessissa todellisesti rinnakkain, jolloin laitteistosuunnittelun edetessä laitteiston testaamiseen on käytettävissä mahdollisimman pitkälle kehitettyjä ohjelmistoversioita.

Merkittävä osa sulautettujen järjestelmien ohjelmistoista liittyy laitteen käyttöliittymään. Nämä ominaisuudet tyypillisesti eivät ole tiukasti alustaan sidottuja, mikä edesauttaa kaksikohdesuunnittelun soveltamista erityisesti käyttöliittymäsuunnitteluvaiheessa.

Lisää aiheesta

Samek M.: *Dual Targeting and Agile Prototyping of Embedded Software on Windows*. http://bit.ly/1tJu6Zn

Grenning J.: *Test Driven Development for Embedded C*. Pragmatic Bookshelf.

Yritysesimerkit

AgiES – tapaustutkimuksia sulautetusta ketteryydestä ja työhyvinvoinnista

AgiES-projektissa tutkittiin ketteryyden kehittämistä ja työhyvinvointia erityisesti sulautettujen järjestelmien suunnittelussa. Tutkimuskohteina oli neljä yritystä ja tutkimuskysymyksiä oli kaksi – ensimmäinen keskittyi työhyvinvointiin ketteryydessä ja toinen ketterien käytäntöjen käyttöönottoon sulautettujen järjestelmien kehittämisessä.

AgiES – Agile and Lean Product Development Methods for Embedded ICT Systems

http://embedded.utu.fi

Tekesin rahoittama hanke 2012–2014.

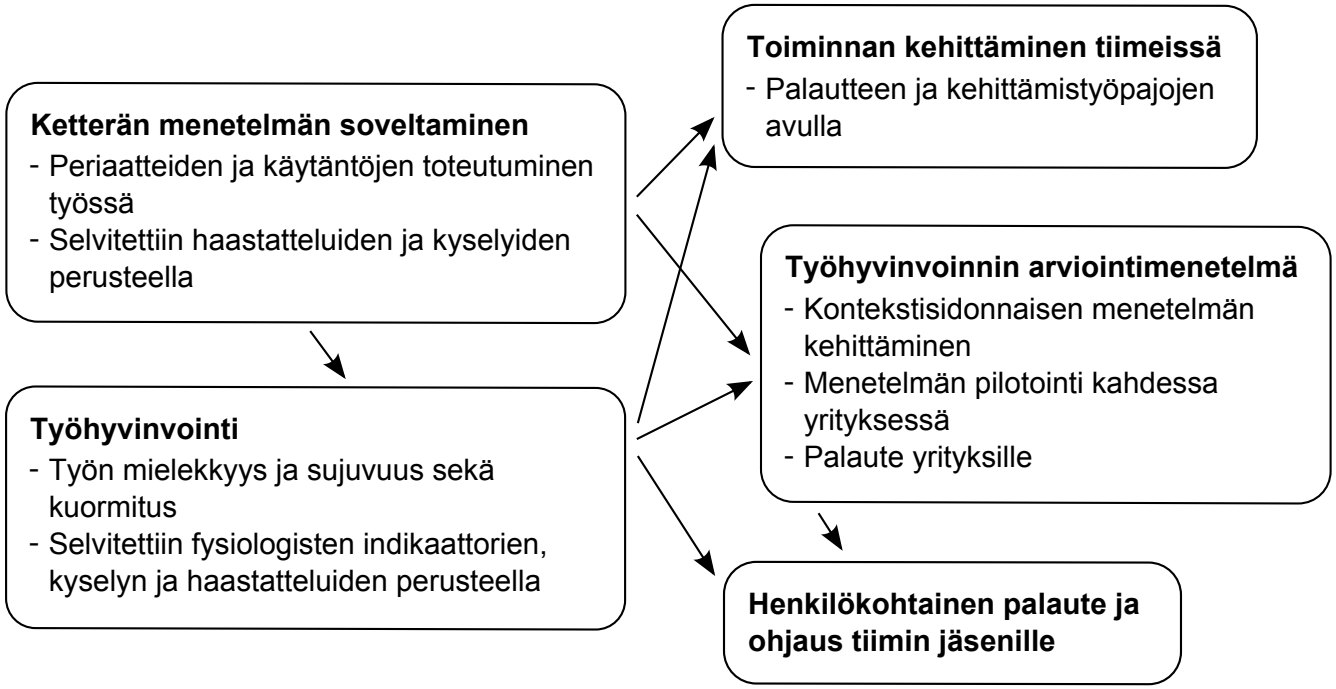
Yhteistyöyritykset: Finnish Software Measurement Association FiSMA ry, Lindorff Finland Oy, Neoxen Systems, Nextfour Group Oy, Nordic ID Oy ja Ericsson Oy.

Muut kumppanit: Agile Finland ry, Suomen tuotannonohjausyhdistys STO ry, TEK Tekniikan akateemiset, Turku Science Park Oy, Yrityssalo Oy, Turku Agile Group ry sekä professori Norio Tokumaru (Nagoya Institute of Technology, Japan).

Työhyvinvointi ja ketteryys

Ketteryyden yhteyttä työhyvinvointiin kartoitettiin ketteryyttä hyödyntävän ohjelmistoyrityksen, Lindorffin, kolmessa tiimissä haastatteluilla, fysiologisilla mittauksilla, sekä kyselyllä. Tutkimuksen tuloksia käsiteltiin tiimin kanssa kehittämistyöpajoissa. Lisäksi fysiologisista mittaustuloksista annettiin henkilökohtainen palaute.

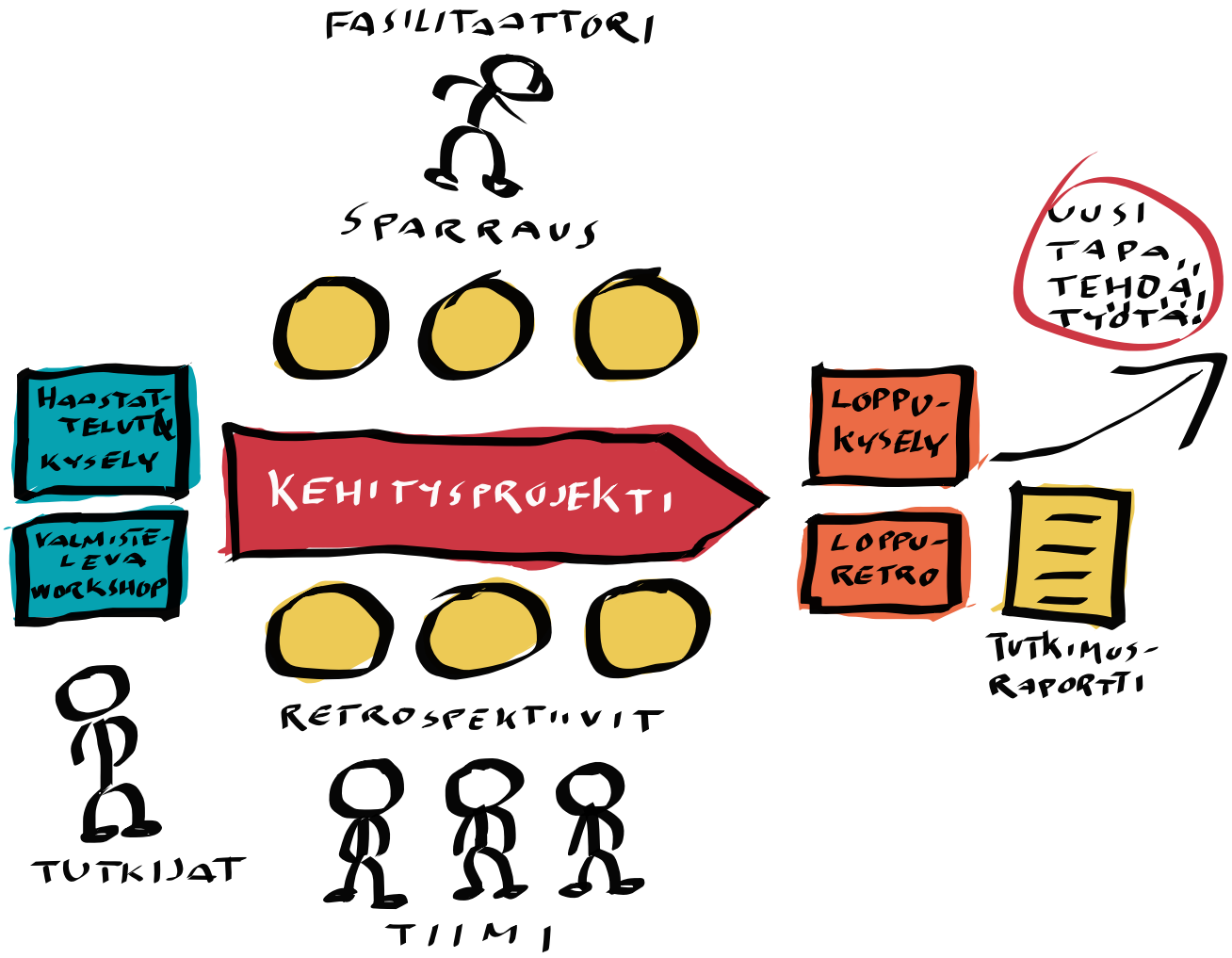
Kehitettyä arviointimenetelmää pilotoitiin fysiologisten mittausten ja kyselyn osalta kahdessa sulautettuja järjestelmiä kehittävässä yrityksessä, Ericssonilla ja Nextfourilla, jotka olivat AgiES-projektin myötä ottaneet käyttöön ketteriä menetelmiä. Tuloksista järjestettiin palautetilaisuudet.



Ketterien menetelmien kehittäminen sulautettuihin järjestelmiin

Ketterien menetelmien kehittäminen kolmessa erilaisessa sulautettujen järjestelmien suunnittelua tekevässä yrityksessä (Ericsson, Nextfour ja Nordic ID) aloitettiin esiselvitysprosessilla, johon kuului haastatteluja osalle tuotekehityshenkilökunnasta sekä kyselyt koko AgiES-projektiin osaa ottavalle tuotekehityshenkilökunnalle. Haastatteluista ja kyselyistä saatujen tulosten perusteella valittiin yhdessä yrityksen edustajien kanssa kehittämiskohteet, joita lähdettiin parantamaan projektin aikana.

Tiimistä valittiin fasilitoija (yksi tai useampi), jonka tehtävänä oli tukea tiimiä sisältäpäin uusien menetelmien käyttöönotossa sekä johdosta sponsori, joka sitoutui tukemaan tiimiä. Menetelmiä kehitettiin yhteistyössä sponsorin ja fasilitoijien kanssa, ja myös tiimi otti osaa menetelmäkehitykseen erityisesti retrospektiivien kautta. Projektin lopuksi toteutettiin kyselyt ja haastattelut uudelleen, jotta saatiin tietoa siitä, miten ketteryyden käyttöönotto oli sujunut.



Ericsson

ERICSSON

Ericsson on kansainvälinen telekommunikaatiojärjestelmiä valmistava yritys. Ericssonilla ketterien menetelmien kehittämiseen sulautettujen järjestelmien suunnittelussa osallistui noin 15-jäseninen tuotekehitystiimi, joka työskentelee modeemien digitaaliosien parissa. Tiimi jakautui erityisosaamisalueiden perusteella neljään pienempään ryhmään, mutta suurimpaan osaan kehitysprojektin toiminnasta se osallistui yhtenä tiiminä. Tiimistä valittiin projektin fasilitoijiksi (s. 52) kaksi suunnittelijaa, joiden lisäksi tiimin esimies oli aktiivisesti mukana työskentelykäytäntöjen kehittämisessä.

Esiselvitysprosessi ja ketteryyden käyttöönotto

Lähtötilanne tuotekehitystiimissä selvitettiin haastatteluilla ja kyselyllä sekä tutustumalla yrityksen prosessista laadittuun dokumentaatioon. Esiselvitysten perusteella suurimmat ongelmat olivat suunnittelijoiden välisessä kommunikaatiossa, laadittavassa dokumentaatiossa sekä yhteisten työskentelykäytäntöjen määrittelyssä. Näihin ongelmiin lähdettiin etsimään ratkaisuja ketteristä menetelmistä kootuilla ja muokatuilla käytännöillä.

Valittuja ketteriä käytäntöjä kehitettiin pilottiprojektissa, joka alkoi marraskuussa 2013 ja jatkui maaliskuuhun 2014. Käytäntöjä valmisteltiin yhteistyössä fasilitoijien ja tiimin esimiehen kanssa syys-marraskuussa 2013. Huolellisen valmistelun ansiosta monet yksityiskohdat saatiin kohdalleen, joten pilotoinnissa päästiin etenemään tehokkaasti jo sen alkumetreiltä lähtien.

Pilotoidut käytännöt

Ericssonilla otettiin käyttöön *iteraatiot* (s. 35). Iteraation pituudeksi valikoitiin jo alkuvalmisteluissa kaksi viikkoa ja se todettiin sopivaksi pilotin aikana.

Iteraation tehtävät suunniteltiin *suunnittelupalaverissa* (s. 37), joka pidettiin iteraation ensimmäisenä maanantaina iltapäivällä. Lisäksi tarvittaessa voitiin järjestää toinen suunnittelupalaveri iteraation toisena maanantaina. Toisessa suunnittelupalaverissa oli mahdollisuus suunnitella iteraation toisen viikon tehtävät uudelleen, jos oli ilmennyt suuria muutoksia projektin sisällössä tai jouduttiin paneutumaan kiireellisiin ongelmiin.

Katselmointipalaveri (s. 39) järjestettiin iteraation ensimmäisenä maanantaiaamupäivänä. Tässä tilaisuudessa käytiin läpi edelliselle iteraatiolle suunnitellut tehtävät ja kirjattiin, mitkä niistä oli tehty ja mitkä piti siirtää tuleviin iteraatioihin.

Katselmointipalaverin jälkeen pidettävissä *retrospektiiveissä* (s. 41) useimmiten käytettiin protokollana ryhmätyömallia, jossa retrospektiivin aluksi kerrottiin päivän aihe ja jaettiin tiimi 2–4 hengen ryhmiin. Ryhmät saivat 10–15 minuuttia aikaa miettiä vastauksia päivän retrospektiivin kysymyksiin. Tämän jälkeen vastaukset koottiin ja sovittiin miten niiden pohjalta mahdollisesti muutettiin yhteisiä käytäntöjä.

Säännöllisiä tilannepalavereja (s. 43) järjestettiin projektin alkuvaiheessa joka toinen päivä niin, että tiimi oli jaettu kolmeen ryhmään. Jokaisen tiimiläisen tuli osallistua oman ryhmänsä palaveriin ja muiden ryhmien palavereihin sai osallistua oman kiinnostuksensa mukaan. Kyseinen järjestelmä koettiin heti aluksi turhan raskaaksi eikä ryhmäjako tuntunut toimivan, joten palaveritiheyttä harvennettiin ja ryhmät yhdistettiin. Lopulta päädyttiin ratkaisuun jossa tilannepalaveri järjestettiin joka torstai sekä joka toinen maanantai.

Tiimi hallinnoi tehtäviään *kehitysjonon* (s. 45) kautta. Tiimin tehtävät oli koottu yhteiseen kehitysjonoon, joka oli toteutettu Excel-taulukkona ja tallennettu yhteiseen versionhallintajärjestelmään. Kehitysjono oli näkyvillä aina yhteisissä tilaisuuksissa. Taulukoissa oli eriteltynä tiimin täydellinen kehitysjono sekä senhetkisen iteraation kehitysjono. Iteraation kehitysjonoista selvisi jokaisen tehtävän vastuuhenkilö ja tehtävän senhetkinen tila.

Keskeytysten hallinnalla (s. 55) lähdettiin hakemaan keinoa rajoittaa tiimin ulkopuolelta tulevia häiriöitä. Pilotin alkuvalmisteluissa päädyttiin ottamaan käyttöön keskeytysten luokittelu ja ylläpitotunti. Jokainen tiimin ulkopuolelta tuleva keskeytys luokiteltiin sen tyypin ja kiireellisyyden mukaan ja pääsääntöisesti joko suoritettiin ylläpitotunnilla tai siirrettiin kehitysjonoon odottamaan seuraavaa suunnittelupalaveria. Ylläpitotuntia varten varattiin tunti jokaisen työpäivän lopusta ylläpitotehtäviin ja tiimin ulkopuolelta tulevien pyyntöjen hoitamiseen. Näin rauhoitettiin päivät varsinaisen projektin toteutukselle ja otettiin jo suunnitteluvaiheessa huomioon näiden ennustamatta tulevien lisätehtävien hoitamisen ajan tarve.

Havainnot ja kokemukset

Valituista käytännöistä keskeytysten hallinnasta sekä ylläpitotunnista päätettiin luopua pilottiprojektin puolessa välissä. Muut käytännöt havaittiin tiimin toimintaan sopivammiksi ja voimavaroja päätettiin kohdentaa niiden kehittämiseen. Keskeytysten hallintaa ja ylläpitotuntia ei kuitenkaan lopullisesti hylätty, vaan niiden käyttöön ja kehittämiseen voidaan palata tulevien projektien yhteydessä.

Käytäntöjen kehittämistä jatkettiin kahdeksan kaksiviikkoisen työskentelyiteraation ajan. Pilottiprojektin kesto määräytyi projektin, jonka yhteydessä käytäntöjä kehitettiin, keston mukaan. Pilottiprojektin lopputuloksena kehitettyjen käytäntöjen soveltamista päätettiin jatkaa tiimin seuraavassa tuotekehitysprojektissa.

Iteratiivisen tuotekehitysprosessin ja sen mukanaan tuomien palaverikäytäntöjen koettiin parantaneen tiimin sisäistä kommunikaatiota. Projektin eteneminen ja vaihe olivat helpommin ymmärrettävissä ja tietoisuus tiimin sisäisistä riippuvuussuhteista parani. Pilottiprojektin myötä tarve tiimin sisäiselle dokumentaatiolle on parantuneen kommunikaation ansiosta vähentynyt.

Nextfour

NEXTFOUR GROUP

Nextfour on sulautettujen järjestelmien suunnitteluun ja kehitykseen erikoistunut yritys. Nextfour kehittää tuotteita erityisesti lääketieteen ja teollisuuden tarpeisiin kattaen ohjelmisto-, elektroniikka- sekä mekaniikkasuunnittelun.

Nextfourilla tehtiin noin kahden kuukauden mittainen tuotekehitysprojekti, jossa kokeiltiin ketteriä käytäntöjä sulautettujen järjestelmien tuotekehityksessä. Projektiin osallistui kaksi elektroniikkasuunnittelijaa, neljä ohjelmistosuunnittelijaa sekä sisäinen asiakas.

Esiselvitysprosessi ja ketteryyden kehittäminen

Yrityksen lähtötilanne kartoitettiin haastatteluilla, kyselytutkimuksella sekä tutustumalla yrityksen prosesseihin. Esiselvityksessä keskityttiin käytettyyn tuotekehitysprosessiin, asiakasrajapintaan, testaukseen sekä palautteeseen. Tämän perusteella päätettiin lähteä selkeyttämään tuotekehitysprosessia, joka jo lähtötilanteeltaan oli melko ketterä, muttei riittävän selkeä tai yhtenäinen.

Projektin alussa yrityksessä oli jo käytetty ketteriä menetelmiä. Käytäntöinä iteraation suunnittelu oli tuttu ja projektinhallintatyökalu Trac oli yleisesti käytössä kehitysjonon hallinnassa. Tämän lisäksi projekteissa oli käytössä kolme erilaista kehitysjonoa, jotka ohjasivat tuotteen kehitystä.

Ketteriä käytäntöjä kokeiltiin syksyllä 2013. Käytäntöjä käytettiin kahden viikon iteraatioissa noin kahden ja puolen kuukauden ajan. Toteutettu projekti oli oma kokonaisuutensa. Ennen projektin alkua vanhaa kehitysprosessia muokattiin määrittelemällä uudet projektivaiheet. Käytännössä vanhaa prosessia selkeytettiin ja yksinkertaistettiin liittämällä tuotteen verifiointi kiinteäksi osaksi itse tuotekehitystä, kuten jo käytännössä oli aiemmin osittain tehty.

Pilotoidut käytännöt

Kehitystiimi otti käyttöön *päivittäiset tilanne-palaverit* (s. 43), jotka pidettiin joka päivä samaan aikaan. Palaveri pidettiin lyhyenä ja tekemisen etenemistä seurattiin päivittäin iteraatiokohtaisen *edistymiskäyrän* (s. 46) avulla.

Jokaisen iteraation päätteeksi pidettiin *katselmointipalaveri* (s. 39), jossa käytiin läpi iteraation aikana saavutetut tavoitteet. Palaverissa projektin etenemistä esiteltiin tuoteprototyypin avulla asiakkaalle silloin, kun se oli mahdollista. Vaatimuslista käytiin läpi asiakkaan kanssa ja lista priorisoitiin asiakkaan toiveiden mukaisesti kehitystiimin avulla. Vaatimukset suljettiin tai hyväksyttiin asiakkaan kanssa yhteisesti sovittujen kriteerien avulla. Projektin etenemistä seurattiin iteraation katselmoinnissa etappikehitysjonon avulla.

Iteraation *suunnittelupalaveri* (s. 37) oli jo projektin alussa vakiintunut ja hyvin toimiva käytäntö. Siksi käytäntöön haettiin vain pieniä parannuksia. Jokaiselle vaatimukselle määrättiin omistaja, joka oli kyseisestä vaatimuksesta vastuussa. Omistaja jakoi vaatimuksen työtehtäviin ja arvioi alustavasti tehtävien suorittamiseen vaadittavan ajan. Tämä arvio täsmennettiin yhteisesti tiimissä ja käytettiin työmääräestimaattina suunniteltaessa iteraation kehitysjonoa.

Retrospektiivi (s. 41) otettiin käyttöön uutena käytäntönä. Siinä keskityttiin tiimin työskentelytapojen parantamiseen. Tiimin jäsenet pohtivat, mitä tehtiin hyvin ja mitä käytäntöjä kannattaa jatkaa. Lisäksi arvioitiin, mitä pitäisi parantaa ja mikä estää tiimiä toimimasta parhaalla mahdollisella tavalla. Arviointien tukena käytettiin *ketteryyden tarkistuslistaa* (s. 61). Listan avulla oli mahdollista käydä tehokkaasti läpi kaikki käytännöt, joita oli sovittu kokeiltavan tai joita harkittiin otettavan kokeiltavaksi.

Yleisesti tunnettujen *tuote- ja iteraation kehitysjonojen* (s. 45) lisäksi projektiin oli jo valmiiksi vakiintunut käyttöön *etappikehitysjono*. Etappi- sekä tuoteen kehitysjonoon kerättiin tuotteen vaatimukset. Tuoteen kehitysjonosta otettiin osa vaatimuksista etappikehitysjonoon. Etappikehitysjonon aikataulu määräytyi ulkoisista tekijöistä, kuten yleisistä tuotteen esittelytilaisuuksista. Tällöin yhteen etappikehitysjonoon kerättiin ne vaatimukset, jotka olivat kyseisellä ajanjaksolla tärkeimmät. Iteraation kehitysjono tarjosi näkyvyyttä projektin etenemisestä iteraation sisällä.

Projektissa oli jo valmiiksi käytössä iteraatiokohtainen *edistymiskäyrä*. Merkittävä uudistus oli ottaa mukaan myös etapin edistymiskäyrä, jossa näkyi ainoastaan kyseisen etapin aikana suoritetut ja jäljellä olevat vaatimukset. Tämän avulla koko projektiin saatiin huomattavasti parempi näkyvyys.

Projektinhallintatyökalu ohjasi tiimin toimintaa vahvasti. Työkalua kehitettiin projektin aikana joustavammaksi ja yksinkertaisemmaksi käyttää. Merkittävin muutos työkaluun tuli projektin seurantaan liittyen; koko projektin näkyvyys vaatimustasolla edesauttoi ymmärtämään tehdyn ja tekemättömän työn määrän.

Havainnot ja kokemukset

Tiimin itseorganisoituvuus ei toiminut projektin alussa johtuen epäselvistä vastuun rajoista. Tämä esti tiimiä toimimasta tehokkaasti, eikä muutoksia saatu tehtyä hyvin projektin alussa. Projektin aikana tilanne muuttui ja tiimi pystyi tekemään itsenäisiä päätöksiä omista toimintatavoistaan. Tämän seurauksena oleellisimmat ja suurimmat muutokset saatiin projektissa tehtyä ja ne selkeyttivät tiimin toimintaa sekä lisäsivät tyytyväisyyttä tiimissä. Suurin muutos oli se, että työmäärää kyettiin seuraamaan koko projektin laajuudessa. Lisäksi säännölliset palaverit selkeyttivät projektin etenemistä ja iteraation katselmoinnissa asiakkaan tarpeet saatiin tehokkaammin tiimin tietoon.

Nordic ID

Nordic ID on RFID-tiedonkeruulaitteita ja niihin pohjautuvia kokonaisratkaisuja kehittävä salolainen yritys. Ketteryyden käyttöönottoprojektina Nordic ID:lla toimi jo aloitettu yritykselle tyypillinen tuotekehitysprojekti, joka jatkui vielä menetelmien kehittämisen jälkeen. Projektiin osallistui työntekijöitä ohjelmisto- ja laitteistotiimeistä, projektipäällikkö sekä tuotekehityspäällikkö, yhteensä noin 10 henkilöä. Projektin mekaniikkakehitys oli pääosin ulkoistettu.

Esiselvitysprosessi ja ketteryyden käyttöönotto

Esiselvitysprosessi suoritettiin haastatteluilla ja kyselyllä, joilla kartoitettiin, miten työntekijät kokivat nykyiset työskentelykäytännöt. Haastatteluihin valikoitiin erilaisissa rooleissa työskenteleviä. Kyselyyn vastasi koko yrityksen tuotekehityksessä työskentelevä henkilöstö. Kyselyiden ja haastatteluiden perusteella haasteina koettiin tuotekehitysprosessin puutteellisuus, asiakkaan tarpeiden ja yrityksen vision näkyminen käytännön työnteon tasolla, hallintatyökalujen ja dokumentoinnin hyödyntäminen sekä projektin tiimityön haasteet.

Esiselvitysprosessissa nousseista asioista keskusteltiin tutkijoiden ja yrityksen edustajien kesken. Projekti, jossa menetelmiä oli tarkoitus kehittää, oli jo käynnissä ja haasteet asiakkaan tarpeisiin ja yrityksen visioon liittyivät enemmän projektin alkuun. Siksi menetelmäkehityksessä päätettiin keskittyä prosessin parantamiseen sekä työkalujen ja dokumentoinnin parempaan käyttöön. Tätä kautta haettiin parannuksia myös tiimityön haasteisiin.

Menetelmien kehitystä tehtiin yhteistyössä tutkijoiden kanssa noin seitsemän kuukauden ajan. Tänä aikana käytäntöjä hiottiin toimivammiksi niin tutkijoiden ja fasilitoijien (s. 52) välisten palaverien kuin retrospektiivienkin (s. 41) aikana.

Pilotoidut käytännöt

Lyhyempiä konkreettisia välitavoitteita pitkään projektiin päätettiin lähteä hakemaan *iteraatioiden* (s. 35) kautta. Iteraatioiden pituudeksi suunniteltiin kuukaudesta kahteen ja tavoitteena oli sitoa iteraatiot joihinkin muihin käytännön etappeihin, kuten laitteiston toteutuskierroksiin. Tästä syystä ensimmäisen iteraation pituudeksi valittiin seitsemän viikkoa. Se koettiin kuitenkin suunnitelmallisuudeltaan liian haastavaksi, ja tämän jälkeen päätettiin iteraatioiden pituudet pitää tasan neljässä viikossa.

Projektin tilanteen ja muiden työtehtävien näkyvyyttä parantamaan otettiin käyttöön työkalut *tuotteen ja iteraation kehitysjonojen hallintaan* (s. 45). Työkalut kehittyivät projektin aikana: aluksi käytössä oli vain *iteraation kehitysjono* kunkin työntekijän työn alla olevista tehtävistä jo aiemmin käytössä olleessa virheidenhallintyökalussa. Tehtävien teko tähän työkaluun koettiin haastavaksi, koska tehtävät luotiin tyhjästä perustuen aiempaan kokemukseen ja kunkin suunnittelijan omaan arvioon projektin tilanteesta ja seuraavista vaiheista. Tämän vuoksi iteraation kehitysjonon rinnalle päätettiin ottaa taulukko- ja tekstimuotoiset *tuotteen kehitysjonot* sekä iteraation tavoitteet. Siten voitiin myös seurata koko projektin etenemistä. Iteraation kehitysjonoon otettiin projektin aikana mukaan myös työmääräarviot, koska ne

helpottivat valitsemaan oikean määrän työtehtäviä yhteen iteraatioon.

Tuotteen kehitysjonon korkeampaa tasoa käytiin läpi yhdessä aina iteraation loppupuolella seuraavaa iteraatiota ajatellen. Tällöin ennen iteraation alussa ollutta *suunnittelupalaveria* (s. 37) kukin työntekijä valmistautui uuteen iteraatioon laatimalla oman osuutensa kehitysjonon tehtävistä. Tehtävälistat käytiin yhdessä läpi suunnittelupalaverissa. Näin varmistettiin sopiva työmäärä ja se, että toisistaan riippuvat työtehtävät tulivat tehdyksi saman iteraation aikana. Iteraation lopuksi *katselmoinnissa* (s. 39) arvioitiin koko projektin osallistujien kesken, miten suunnitelmat olivat toteutuneet ja miksi sekä mitä oli saatu aikaiseksi.

Iteraatioiden aikana pidettiin *viikoittaiset tilannepalaverit* (s. 43). Tuotekehityksessä monella myös muut tehtävät, kuten ylläpitotyöt, veivät paljon aikaa. Todettiin, että kerran viikossa olevat palaverit riittävät. Myös tilannepalavereissa olivat mukana kaikki projektin työntekijät. Tilaisuuden kesto oli puolesta tunnista tuntiin. Työtehtävät käytiin läpi kehitysjonotyökalua hyödyntäen. Projektin alussa työkalua ei muistettu pitää ajan tasalla, mutta projektin loppuun mennessä kaikki päivittivät työtehtäviensä tilanteen työkaluun – viimeistään juuri ennen palaveria.

Jokaisen iteraation päätteeksi pidettiin myös *retrospektiivi*. Suurimmaksi osaksi retrospektiivit olivat *ketteryyden tarkistuslista* -pohjaisia (s. 61). Kun retrospektiivin pohja pidettiin samana, keskustelut alkoivat pyöriä samojen teemojen ympärillä, eikä uusia ideoita enää syntynyt. Tämän vuoksi välillä retrospektiivin aiheena oli jokin muu haastavaksi koettu asia, kuten dokumentointi ja ylläpito. Retrospektiiveissä sovitut muutokset työskentelytapoihin pidettiin pieninä ja toteuttamiskelpoisina, ja niiden toteutumista käytiin läpi aina seuraavassa retrospektiivissä.

Havainnot ja kokemukset

Projektinhallintatyökalut tuntuivat aluksi haastavilta, mutta niiden käyttöön totuttiin nopeasti ja niistä koettiin olevan hyötyä. Vaikka palaverien määrä on lisääntynyt, ovat ne osoittautuneet hyödyllisiksi, ja palaverien ajankäyttö on tehostunut käytäntöjen tultua tutummiksi. Ketterien menetelmien käyttöönoton jälkeen ymmärrys muiden työstä sekä tietyn ajanjakson aikana tehtävästä työmäärästä on lisääntynyt. Suunnittelijat ottavat enemmän vastuuta niin työmääräarvioista kuin työtehtävien aikataulutuksestakin. Yhteistyön myötä tiiminjäsenten luottamus toisiinsa on kasvanut, ja dokumentaation tarve on pienentynyt kommunikaation parantumisen myötä.

Lindorff Finland

Lindorff Finland on asiakas- ja luottosuhteisiin keskittyvien palvelujen tarjoaja, joka tuottaa luotonhallinnan ja maksamisen palveluja, kuten luottopäätöksiä, laskutusta, perintää ja saatavien ostoa. Lindorffissa tutkimukseen ja kehittämiseen osallistui ohjelmistokehityksestä vastaava yksikkö ja sen eri tuotteita tai tuotteiden osa-alueita kehittävät kolme kehitystiimiä, joista yksi jakautui useampaan alatiimiin.

Lähtötilanne selvitettiin kahden tiimin esimiehen ja yhden alatiimin fasilitoijan (s. 52) haastatteluilla. Lisäksi käytiin läpi organisaatiossa olemassa olevan henkilöstökyselyn työhyvinvointiin liittyviä tuloksia kolmen tiimin osalta. Lindorffilla tutkittiin tiimien kokemuksia ketterien menetelmien soveltamisesta, koettuja ketteryyden ja työhyvinvoinnin yhteyksiä sekä tiiminjäsenten kuormituksen vaihteluita yhden iteraation aikana. Tutkimuksen tuloksena tunnistettiin, miten ketteryyttä voidaan soveltaa työhyvinvointia tukien ja mitä työhyvinvointia haastavia tekijöitä on otettava huomioon ketteriä menetelmiä sovellettaessa.

Tutkimuksen suorittaminen

Tutkimuksessa kerättiin tietoa kolmen kehittäjätiimin jäseniltä, heidän esimiehiltään ja tuoteomistajilta (s. 53) haastatteluilla, fysiologisilla stressimittauksilla ja kyselyllä. Ensimmäisenä aineistonkeruumenetelmänä tutkimuksessa olivat teemahaastattelut. Tiiminjäsenten ja esimiesten haastatteluissa pääpaino oli kokemuksissa ketterien käytäntöjen toteuttamisesta, koetussa työhyvinvoinnissa sekä tulkinnoissa näiden välisistä yhteyksistä. Tuoteomistajien haastatteluissa taas painottuivat koettu tyytyväisyys ketterien käytäntöjen toteuttamiseen ja yhteistyöhön kehittäjätiimin kanssa sekä näiden väliset yhteydet. Kaikissa haastatteluissa teemoina olivat ketterien menetelmien kehitysnäkymät ja -tarpeet oman työn kannalta.

Toiseksi fysiologisilla stressimittauksilla selvitettiin iteraation aikaista vireys- ja stressitasoa sekä palautumista. Tiiminjäsenten fysiologista kuormittumista tarkasteltiin iteraation alussa, keskivaiheilla ja lopussa. Näin voitiin fysiologisen stressivasteen avulla testata ketterien menetelmien antamaa lupausta tasaisesta kuormittumisesta iteraation aikana. Fysiologiset stressimittaukset käsittivät sykevälivaihtelun ja kortisolihormonin tason.

Kolmantena tutkimusmenetelmänä oli verkkokysely työhyvinvoinnista ketterässä kontekstissa. Kyselyn teemoja olivat ketteryyden toteutuminen omassa työssä, tiimissä ja organisaatiossa, työn voimavara- ja kuormitustekijät sekä työn imu ja palautuminen. Haastattelun ja kyselyn pohjalta voitiin tutkia tiimien ketteryyttä ja sen ilmenemismuotoja sekä työhyvinvointia työn sujumisen, mielekkyyden ja kuormittumisen näkökulmista. Eri aineistoilla saatiin tietoa ketterien menetelmien soveltamisen ja periaatteiden esiintymisen sekä työhyvinvoinnin yhteyksistä.

Tutkimuksen tulokset palautettiin osallistujille kehittämistyöpajoissa, joissa niitä tulkittiin yhdessä tiimin jäsenten kanssa tiimin työskentelyn kehittämiseksi. Työpajat tarjosivat tietoa ketterän toimintatavan periaattein toimivan

tiimin jäsenten jäsennyksistä, ketterän työskentelyn periaatteista ja niiden soveltamisesta omassa työssä. Työpajojen tavoitteena oli lisäksi ketterien toimintamallien ja työtapojen kehittäminen työhyvinvointia tukevasti.

Tutkimustulokset

Ketterät palaverikäytännöt osoittautuvat helposti omaksuttaviksi ja kommunikaatiota lisääviksi. Palaverikäytäntöjen työhyvinvointia tukevan soveltamisen kannalta on tärkeää hajautettujen tiimien palaverikäytäntöjen kehittäminen, osallistujien aktiivisuuden tasapuolisuuden varmistaminen sekä palaverien pitäminen riittävän kevyinä, fokuksessa ja aikataulussa. *Iteratiivisuus ja inkrementaalisuus* lisäävät työn mielekkyyttä ja motivaatiota pienten välitavoitteiden saavuttamisen sekä nopean asiakastarpeisiin reagoinnin myötä. Lyhyet iteraatiot tasaavat koko projektin aikaista työpainetta ja antavat työrauhaa. Fysiologiset stressimittaukset kuitenkin osoittavat iteraation aikaisen kuormittumisen vaihtelevan. Kuormittuminen oli iteraation alussa ja lopussa hieman koholla verrattuna iteraation keskivaiheen kuormittumiseen.

Itseorganisoituvat tiimit (s. 51) kehittävät ketteriä menetelmiä omaan tarpeeseensa sen sijaan, että toteuttaisivat niitä täysin oppikirjan mukaan. Työhyvinvointia lisäävänä ja työpainetta tasaavana koetaan se, että tiiminjäsenet jakavat toistensa kanssa sekä haasteet että onnistumiset. Jakamista tukee tiiminjäsenten moniosaajuus. Sen sijaan yhteen henkilöön kiinnittyvä erikoisosaajuus on haasteellista jakamisen kannalta. Avokonttori, fyysinen tehtävätaulu ja ohjelmistotyökalut tukevat tiimin ketterää toimintaa parantamalla tiedonkulkua ja jäsentämällä yhteistä tehtävää. Ketteryyden toteutumisen kannalta on tärkeää, että tiimillä on päätäntävaltaa työjakson työmäärään ja myös työn toteuttamistapoihin, ts. heillä on mahdollisuus soveltaa ketteriä menetelmiä haluamallaan tavalla.

Toimiva yhteistyö liiketoiminnan ja tuoteomistajien kanssa on edellytys sille, että kehittäjätiimi pystyy tuottamaan lisäarvoa ja ennustettavuutta liiketoiminnalle. Tähän kuuluu molemminpuolinen läpinäkyvyys ja nopea palautesykli sekä liiketoiminnan osallistuminen katselmuksiin ja keskustelu tuotteen kehitysjonosta. Tuoteomistajan rooli (s. 53) on haastava vaatiessaan sekä teknistä että liiketoimintaosaamista. Roolin onnistumisen kannalta oleellista on riittävä aika ja oikeus tehdä rooliin kuuluvat päätökset. On tärkeää, että liiketoiminta, myynti mukaan lukien, sekä asiakkaat ovat omaksuneet ketterän kehittämistavan. Selkeä ja yhteinen tuotevisio sekä toimiva vuorovaikutus asiakkaan kanssa ylläpitävät ja edistävät kehittäjien motivaatiota. Riippuvuudet muihin tiimeihin ja asiakkaisiin saattavat haastaa tasaisen työtahdin aiheuttaen toisinaan odottelua ja toisinaan ylityön tarvetta.

Ketterän filosofian ja periaatteiden ymmärtäminen, käytäntöjen toteuttaminen sekä jatkuvan parantamisen ylläpitäminen ovat keskeisiä mutta myös haasteellisia ketterän toiminnan osa-alueita. Esimerkiksi retrospektiivien (s. 41) toteuttaminen säännöllisesti ja niin, että ne koetaan hyödyllisiksi, on haastavaa. Kiire ja kyllästyminen vaikuttavat tähän. Niin tuoteomistajan, fasilitoijan kuin ketterän tiimin esimiehenkin roolissa on tärkeää, että heillä on riittävästi aikaa kehittäjätiimille. Ketteryyttä tukevan esimiehen, johdon ja organisaation on hyvä sitoutua näkyvästi ketteryyteen sekä tukea erityisesti kehittäjätiimin itseohjautuvuutta.

BA Group

BA Group on ohjelmistokehitykseen, järjestelmäintegrointiin sekä näitä tukeviin menetelmiin ja työkaluihin erikoistunut yritys. Yrityksellä on kokemusta useammista toimeksiannoista, joissa on asiakkaan toimintatapojen lisäksi kehitetty niitä tukevia työkaluja.

Kehitystyökalut määrittelevät ruohonjuuritason toimintatavat ja kehittäjien käyttämän termistön. Tätä kautta työkalujen merkitys päivittäiseen työhön ja sen menetelmiin on huomattava. Toisaalta ketterät menetelmät poikkeavat merkittävästi perinteisistä työtavoista, joita varten suurin osa nykyisistä kehitystyökaluista on rakennettu. Näin ollen työkalut ohjaavat käyttäjäänsä tiettyihin työkäytäntöihin ja samalla työkäytännöt tarvitsevat tuekseen tietynlaisia työkaluja. Optimaalisessa tilanteessa työkalu tukee tiimin jäseniä valitun kehitysmenetelmän noudattamisessa.

Ketteryyttä tukevan työkalun valinnasta on kirjoitettu tämän käsikirjan aiemmissa osioissa. Yleisten kehitystyökalujen vaikutus työmenetelmiin on vaikeammin todettavissa, sillä parhaiten tämän näkevät työkaluja käyttävät työntekijät eli yleensä kehittäjät. Ketterät menetelmät helpottavat työkalujen ja työmenetelmien huomioonottamista itseorganisoituvuuden (s. 51) sekä retrospektiivien (s. 41) kautta. Vaikka tiimille tulee lisävastuuta menetelmien ja työkalujen sovittamisesta, sillä on myös paras tieto sovitustyön toimivaan toteuttamiseen.

On huomattava, että myös valitut alustat voivat tukea ketterän kehittämisen periaatteita. Alusta voi esimerkiksi tarjota valmiit työkalut testivetoisen kehittämisen (s. 63) tarvitsemille yksikkötesteille. Lisäksi alustavalinnalla voidaan vaikuttaa siihen, miten sujuvaa ohjelmistojen ja laitteistojen yhteensovittaminen on. Tästä esimerkkinä ovat tietyt funktionaaliset kielet, joiden avulla on aiempaa helpompi rakentaa teollisen internetin laajoja järjestelmiä (laajemmin esineiden internet, internet of things).

Myös sulautetuissa järjestelmissä toimintakenttä muuttuu teknologian kehittyessä. Aiemmin mainitun teollisen internetin lisäksi sulautetut järjestelmät muodostavat yhä laajempia järjestelmiä ja kytkeytyvät osaksi pilvipalveluita sekä big data -ratkaisuja. Tällöin on tärkeää, että työkalut pystyvät tukemaan hyvin eri tyyppistä työtä aina elektroniikan suunnittelusta reaktiivisten web-sovellusten toteuttamiseen.

Neoxen Systems

Neoxen Systems on varsinaissuomalainen ohjelmistotalo, joka on erikoistunut oppilaitoksille suunnattuihin tietojärjestelmiin. Yritys toimii useissa verkostoissa ja monesti ratkaisut syntyvät yhteistyössä verkostokumppanien kanssa, jotka voivat edustaa monipuolisesti teknisiä asiantuntijoita, toimialan asiantuntijoita ja tilaajia.

Tiiviissä kumppaniverkostossa toimiminen muistuttaa ketterän kehityksen näkökulmasta paljon sulautettujen järjestelmien kehittämistä. Eri toimijoilla on erilaiset toimintatavat ja osaamisprofiilit, mutta toiminnan tavoite on silti kaikille osapuolille sama. Oleellista on löytää sellainen aikataulutus, jota kaikki osapuolet voivat sujuvasti noudattaa. Lisäksi on tärkeää määritellä iteraatioiden välituotteet sellaisiksi, että kaikki osapuolet voivat niiden kautta osoittaa työn etenemisen. Partneriverkon hallinta ja tehokas kommunikaatio eri osapuolten kesken voi joskus olla haastavaa verkostossa, jossa toimijat edustavat eri organisaatioita. Tähänkin ketteryys voi tuoda ratkaisumalleja säännöllisten palaverikäytäntöjen ja kokoukset tehokkaina pitävien toimintamallien kautta.

Verkostossa tuotteen tai palvelun ominaisuuksien päättäminen voi olla joskus haastavaa, koska kaikkien osapuolten näkemykset tulee ottaa huomioon. Ketteryydessä korostetaan lisäksi muutosten hyväksymistä ja niihin reagoimista, mikä entisestään lisää joustavan vaatimustenhallinnan merkitystä. Verkostossa voidaan jakaa rooleja niin, että tietyt kumppanit vastaavat asiakasrajapinnan seuraamisesta ja ottavat tuoteomistajan (s. 53) roolia. Lisäksi tuotekehitysvastuita voidaan jakaa modulaarisesti (s. 65) niin, että keskeisimmät suuntaviivat lyödään lukkoon koko verkoston kesken ja yksityiskohdat päätetään kustakin moduulista vastaavassa kehitystiimissä.

Neoxen Systemsillä on pitkä kokemus ohjelmistokehityksen työmenetelmistä. Ketteryys ei saa olla itsetarkoitus, vaan työmenetelmät tulee valita tilanteen mukaan. Erityisesti sulautettujen järjestelmien kehityksessä menetelmävalinnan tulisi tukea erilaisten tiimien työtä ja mahdollistaa erilaiset työvuot. Menetelmävalinta ei ole aina välttämättä kehittäjistä kiinni, vaan myös asiakkaiden todelliset tarpeet ja osin myös ennakkokäsitykset vaikuttavat asiaan. Esimerkiksi julkisella sektorilla hankintojen toteuttamista ketterinä projekteina usein vasta harjoitellaan. Onneksi asiakkaiden tietämys ja osaaminen ketteristä menetelmistä on viime vuosina kasvanut, joten myös ketterät menetelmät ovat tulleet varteenotettavaksi vaihtoehdoksi yhä useammassa projektissa.

Neoxen Systemsin esimerkki osoittaa, että ajatukset ketteryyden ulottamisesta yli organisaatiorajojen ovat mahdollisia. Kun pohditaan, miten erilaiset organisaatiot voivat työskennellä yhteisen tavoitteen eteen, voidaan oppia paljon sulautettujen järjestelmien kehittämisestä, jossa samat haasteet ilmenevät eri mittakaavassa.

Lisätiedot

Kirjallisuusluettelo

Sulautetut järjestelmät

Catsoulis J.: Designing Embedded Hardware. O'Reilly Media (2005).

White E.: Making Embedded Systems: Design Patterns for Great Software. O'Reilly Media (2011).

Ketterä kehitys

Abrahamsson P., Salo O., Ronkainen J., Warsta J.: *Agile Software Development Methods: Review and Analysis*. VTT Publications 478 (2002).

Beck K., Beedle M., van Bennekum A., Cockburn A., Cunningham W., Fowler M., Grenning J., Highsmith J., Hunt A., Jeffries R., Kern J., Marick B., Martin R., Mellor S., Schwaber K., Sutherland J., Thomas D.: *Agile Manifesto*. http://agilemanifesto.org/ (2001).

Larman C.: *Agile and Iterative Development: A Manager's Guide*. Addison-Wesley (2003).

Larman C., Basili V.: *Iterative and Incremental Development: A Brief History*. Computer, Volume 36 Issue 6 (June 2003).

Ketterän kehityksen menetelmät ja käytännöt

Anderson D.: *Kanban – Successful Evolutionary Change for Your Technology Business*. Blue Hole Press (2010).

Beck K.: *Extreme Programming Explained: Embrace Change, 2nd Edition*. Addison-Wesley (2004).

Clifford J.: *Transitioning to Scrum: Selecting the Product Owner*. http://bit.ly/1lSgVTk (21.01.2010).

Derby E., Larsen D.: *Agile Retrospectives, Making Good Teams Great*. Pragmatic Bookshelf (2006).

Grenning J.: *Test Driven Development for Embedded C*. Pragmatic Bookshelf (2013).

McConnell P.: *Your Client Isn't Your Product Owner*. http://bit.ly/WH32vd (02.11.2012).

Perry T.: *Drifting Toward Invisibility: The Transition to the Electronic Task Board*. AGILE '08 Conference (August 2008).

Poppendieck M., Poppendieck T.: *Implementing Lean Software Development: From Concept to Cash*. Addison-Wesley Professional (2006).

Schwaber K., Beedle M.: *Agile Software Development with Scrum*. Prentice Hall (2001).

Schwaber K., Sutherland J.: *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game*. http://bit.ly/ZgrC7Z (July 2013).

Stevens P.: *The Effective Sprint Planning Meeting*. http://bit.ly/1xGSnle (14.1.2009).

Wells D.: *Extreme Programming – a gentle introduction*. http://bit.ly/1tuWFpx (8.10.2013).

Ketterä kehitys sulautetuissa järjestelmissä

Tutkimusryhmän kirjallisuuskatsaus, jossa käydään läpi keskeisimmät akateemiset tutkimukset aiheesta:

Kaisti M., Rantala V., Mujunen T., Hyrynsalmi S., Könnölä K., Mäkilä T., Lehtonen T.: A*gile methods for embedded systems development - a literature review and a mapping study*. EURASIP Journal on Embedded Systems 15 (November 2013).

Muut lähteet:

Cordeiro L., Barreto R., Oliveira M.: *Towards a Semiformal Development Methodology for Embedded Systems*. International Conference on Evaluation of Novel Approaches to Software Engineering (May 2008).

Kaisti M., Mujunen T., Mäkilä T., Rantala V., Lehtonen T.: *Agile Principles in the Embedded System Development*. XP2014 Conference (May 2014).

Samek M.: *Dual Targeting and Agile Prototyping of Embedded Software on Windows*. http://bit.ly/1tJu6Zn (12.04.2013).

Suomi S.: *Project Management Tools in Agile Embedded Systems Development*. http://bit.ly/1rVOTc9 (2014).

Työhyvinvoinnin kehittäminen ja ketteryys

Aura O., Ahonen G., Ilmarinen J.: *Strategisen hyvinvoinnin tila Suomessa 2011* (2011).

Huusko, L.: Työpaikkana tiimi – *Miten tiimi kasvaa vastuuseen?* Edita Publishing (2007).

Janhonen M.: *Tasapainoinen tiedon jakaminen ja tiimityön laatu*. Hallinnon tutkimus, 28 (4) (2009).

Launis K., Schaupp M., Koli A., Rauas-Huuhtanen S.: *Muutospajaohjaajan opas*. Tykes, Raportteja 71, http://bit.ly/1qggrHR (2010).

Lindström K., Leppänen A. (toim.): *Työyhteisön terveys ja hyvinvointi*. Työterveyslaitos: Helsinki (2002).

Marchenko A., Abrahamsson P.: Scrum in a Multiproject Environment: *An Ethnographically-Inspired Case Study on the Adoption Challenges*. AGILE '08 Conference (August 2008).

Multanen L., Bredenberg K., Koskensalmi S., Lauttio L.-M., Pahkin K.: *Parempi työyhteisö – avaimia kehittämiseen*. Työterveyslaitos: Helsinki (2005).

Palomäki P.: *Software Industry Transformation from Products to Services*. ICSOB 2010: First International Conference on Sofware Business (June 2010).

Rauramo P.: *Työhyvinvoinnin portaat – Viisi vaikuttavaa askelta*. Työturvallisuuskeskus: Helsinki (2012).

Schaupp M., Koli A., Kurki A.-L., Ala-Laurinaho A.: *Yhteinen muutos – työhyvinvointia työtä kehittämällä*. Työterveyslaitos: Helsinki (2013).

Sanasto

Alustalähtöinen suunnittelu *(platform-based design)* – Suunnittelutapa, joka pohjaa yleiskäyttöiseen kehitysalustaan (s. 66).

Edistymiskäyrä *(burndown chart)* – Kuvaaja tehdyn ja jäljellä olevan työmäärän visualisointiin (s. 46).

Fasilitoija – Tiimin jäsen, jonka tehtävänä on varmistaa, että tiimissä noudatetaan yhdessä sovittuja toimintatapoja (s. 52).

Fysiologiset stressi- ja palautumismittaukset – Mittauksia työtekijän vireys- ja stressitasosta sekä palautumisesta. Näitä pystytään selvittämään sykevälivaihtelun ja kortisolihormonin tason avulla.

Hyväksymiskriteerit *(acceptance criteria)* – Ehdot, joilla vaatimus voidaan todeta toteutetuksi (s. 47).

Iteraatio – Lyhyt muutaman viikon työskentelyjakso, joka tuottaa jonkinlaisen toimivan lopputuoteen tai tuotteen osan. Iteraation synonyymeinä käytetään myös termejä sykli, kehitysjakso ja erityisesti Scrumissa sprintti (s. 35).

Iteraation kehitysjono *(sprint backlog)* – Tapa esittää jokaisen kehitysjonon kohdan yksityiskohtaiset tehtävät. Jokaista iteraatiota varten luodaan uusi iteraation kehitysjono iteraation suunnittelun aikana (s. 45).

Itseorganisoituvuus – Tiimillä on valta ja samalla vastuu oman päivittäisen työnsä organisoinnista (s. 51).

Jatkuva integrointi *(continuous integration)* – Eri kehittäjien tekemää ohjelmistokoodia integroidaan jatkuvasti yhteen ja testataan automaattisilla testeillä. Yksi mm. XP:n (s. 6) käytännöistä.

Kaksikohdesuunnittelu *(dual targeting)* – Järjestelmän ohjelmistoa kehitetään yhtäaikaisesti sekä lopulliselle laitteistolle että esim. PC-pohjaiselle simulaatioalustalle (s. 67).

Kanban *(看板)* – Ketterä kehitysmenetelmä, jonka tavoitteena on optimoida työvuo (s. 8).

Katselmointipalaveri *(iteration review)* – Jokaisen iteraation päätteeksi pidettävä palaveri, jossa esitellään, mitä iteraation aikana on saatu aikaan ja verrataan saavutuksia iteraation suunnittelupalaverissa tehtyihin tavoitteisiin (s. 39).

Kehitysjono *(backlog)* – Tapa tiimin työn organisoimiseen listan avulla ja työn etenemisen seuraamiseen asiakkaan toiveiden mukaan (s. 45).

Kehitysjonokortti – Kortti, jonka avulla kuvataan kehitysjonon kohtaa tehtävätaululla (s. 59).

Kehitysjonon kohta *(backlog item)* – Yhden iteraation aikana toteutettavissa oleva uusi toiminnallisuus tuotteeseen (s. 45).

Ketteryyden tarkistuslista *(agile checklist)* – Ketteryyden käyttöönottovaiheessa esimerkiksi retrospektiiveissä käytettävä tapa käydä läpi valittuja käytäntöjä ja niiden toimivuutta (s. 61).

Käyttäjätarina *(user story)* – Tapa esittää kehitysjonon kohdat käyttäjälähtöisesti (s. 49).

Lean – Tuotekehitysfilosofia, jolla tavoitellaan hukan minimoimista. Tästä on johdettu myös Lean-ohjelmistokehitysmenetelmä (s. 8).

Modulaarinen suunnittelu – Suunnittelutapa, jossa laitteen toiminnallisuus jaetaan pieniin itsenäisesti toteutettaviin osiin, joiden rajapinnat on tarkasti määritelty (s. 65).

Retrospektiivi *(retrospective)* – Tilaisuus, jossa tarkastellaan, miten asioita on tiimissä tehty. Lopuksi kehitetään parannuksia työskentelymenetelmiin ja sovitaan käytännön toimenpiteistä (s. 41).

Scrum – Ketterä menetelmä, jossa on tarkkaan määritellyt seremoniat, yleisin tällä hetkellä käytetyistä ketteristä menetelmistä (s. 4).

Suunnittelupalaveri *(iteration planning)* – Palaveri, jossa tiimi selvittää itselleen kaksi asiaa: iteraation tavoitteen ja iteraation työtehtävät (s. 37).

Tehtävä *(task)* – Kehitysjonon kohdasta jaettu osa, jonka yksi kehittäjä pystyy toteuttamaan maksimissaan yhden iteraation aikana (s. 49).

Tehtäväkortti – Kortti, joka kuvaa tehtävän etenemistä tehtävätaululla (s. 59).

Testivetoinen kehitys *(test-driven development, TDD)* – Menetelmä, jossa ominaisuuksien kehittäminen aloitetaan aina määrittelemällä testit lopulliselle, valmiille ominaisuudelle (s. 63).

Tilannepalaveri *(daily stand-up)* – Lyhyt palaveri, jonka tarkoituksena on koota kehitystiimi säännöllisesti yhteen, välittää tietoa kehitystiimin sisällä, pitää koko tiimi tietoisena projektin tilanteesta ja etenemisestä sekä helpottaa ongelmien aikaista tunnistamista sekä ratkaisemista (s. 43).

Tuoteomistaja *(product owner)* – Henkilö, joka on vastuussa tuotteen arvon maksimoinnista ja kehitystiimin työstä ja hallinnoi tuotteen kehitysjonoa (s. 53).

Tuotteen kehitysjono *(product backlog)* – Lista, joka kuvaa tuotekokonaisuuden ja täten näyttää mitä asiakkaalle lopulta toimitetaan (s. 45).

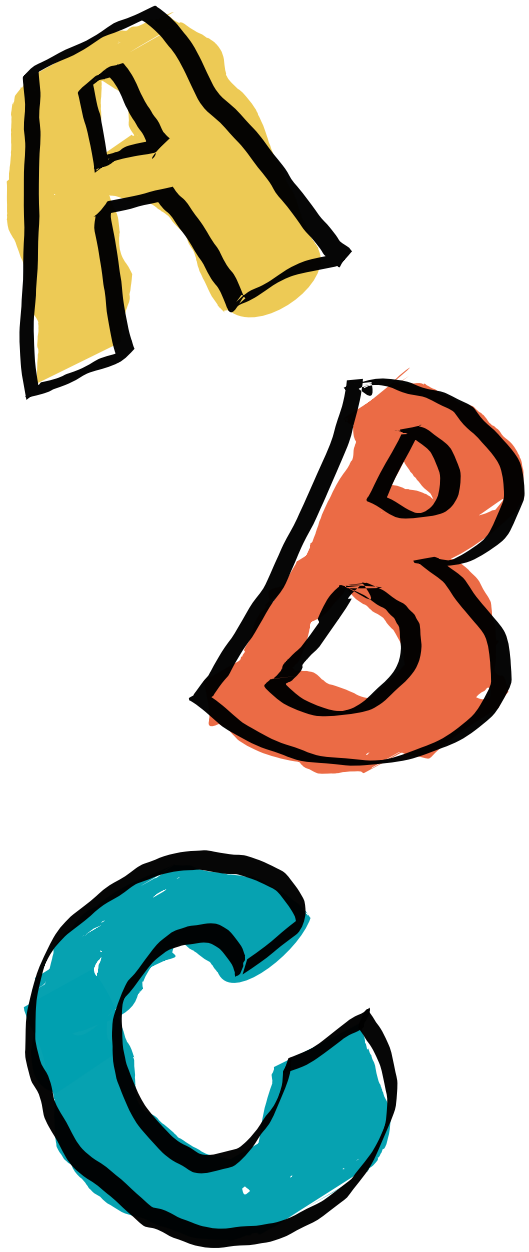
Työvuo *(work flow)* – Työn askeleiden eteneminen, jonka arvoa pyritään maksimoimaan esim. Kanban -menetelmässä (s. 8).

Valmiin (ominaisuuden) määritelmä *(definition of done, DoD)* – Yhteisten minimikriteerien määritelmä ominaisuuksien hyväksymiselle (s. 47).

Valmistellun (vaatimuksen) määritelmä *(definition of ready, DoD)* – Vaatimuksen minimitarkkuuden määritelmä (s. 47).

XP (Extreme Programming) – Ketterä menetelmä, joka perustuu tarkasti määriteltyihin käytäntöihin (s. 6).

Yhteissuunnittelu *(co-design)* – Menetelmä, jossa laitteistoa ja ohjelmistoa kehitetään yhdessä ja yhtä aikaa.



Teijo Lehtonen | Samuli Suomi | Tero Jokela | Matti Kaisti
Marko Ylitolva | Ville Rantala | Minna Isomäki | Kaisa Könnölä | Tuomas Mäkilä | Seppo Tuomivaara | Marja Känsälä

Sulautettujen järjestelmien ketterä käsikirja kertoo, mitä ketterät menetelmät ovat ja miten niitä sovelletaan sulautettuja järjestelmiä rakennettaessa. Kirja avaa käytännön tekniikat ja kuvaa tosielämän kokemuksia. Aihetta tarkastellaan paitsi työmenetelmien myös työhyvinvoinnin näkökulmasta.

Turun yliopisto
University of Turku

Technology
Research
Center

Työterveyslaitos

Tekes

Mukana kirjan julkaisussa:

FISMA
Finnish Software Measurement Association

Agile Finland

SCIENCE PARK TURKU

TEK
TEKNIIKAN AKATEEMISET

ISBN: 978-951-29-5837-5 (sid.), 978-951-29-5838-2 (pdf)

http://embedded.utu.fi/kasikirja