Teksti: Mika Koski

Miten lunastaa ketteryyden arvolupaukset?

Mika Koski on Trinil Oy:n Senior Consultant, jolla on 13 vuoden kokemus ohjelmistonkehityksestä eri rooleissa. Nykyisin hän toimii konsulttina ketterään kehityksen projekteissa. Trinil Oy on asiantuntijayritys, joka on keskittynyt projektinhallinnan ja sovelluskehityksen asiantuntijapalveluihin.

Moni ketterä ohjelmistoprojekti on kohdannut loppunsa ennenaikaisesti, vaikka luvassa piti olla hyvää, halvalla ja nopeasti. Käyttäjäkeskeiset suunnittelumenetelmät lupaavat parempaa yhteiseloa ohjelmistojen ja käyttäjien välille, mutta lopputuloksena saattaa olla näennäisen älykkäitä toimintoja epäyhtenäisesti toteutettuna. Miksi arvolupaukset jäävät lunastamatta?

Ohjelmistot ovat verkkopankkien, digiboxin ja kännykän myötä tulleet osaksi lähes jokaisen ihmisen arkipäivää. Muutos on huima niistä ajoista, jolloin ohjelmistojen käyttäjät olivat valkotakkisia miehiä ilmastoiduissa konesaleissa. Ohjelmistojen arkipäiväistyminen on tuonut myös uusia vaatimuksia ohjelmistojen tekijöille. Nykyään ohjelmiston käyttäjä voi olla kuka tahansa, missä tahansa ja käyttäjällä ei ole välttämättä minkäänlaista käyttökoulutusta ohjelmiston käyttöön. Kaikkien entisten laatuhyveiden lisäksi, tulee nykyaikaisen ohjelmiston olla helppokäyttöinen ja käyttökokemukseltaan miellyttävä. Miellyttävä verkkopalvelu tuottaa lisämyyntiä nykyisten asiakkaiden joukossa ja kaikkein parhaimmassa tapauksessa saa nämä suosittelemaan palvelua uusille käyttäjille.

Myös ohjelmistoteknologia on muuttunut sitten valkotakkisten miesten. Nykyiset ohjelmat eivät koostu enää pääohjelmasta ja aliohjelmista, jotka ladataan (kooltaan) suureen keskuskoneeseen, ajetaan ja sitten poistetaan uuden ajon tieltä. Nykyisissä ohjelmistoissa temmeltävät oliot, jotka heittelevät toisilleen poikkeuksia jopa tuhansia kilometrejä yhden ainoan transaktion aikana. Matka laskuaan maksavan verkkopankkiasiakkaan selaimesta, pankin oman tietojärjestelmäviidakon läpi perusjärjestelmään ja sieltä takaisin jonkin toisen pankin perusjärjestelmään saattaa mennä yli kahdenkymmenen arkkitehtuurikerroksen läpi.

Edelläkuvatussa toimintakentässä ohjelmistojen tekeminen on vaikeaa ja kallista. Vaikeaa siksi, että eri järjestelmiä ja integraatiopisteitä on paljon. Kallista siksi, että ohjelmistoprojektin aikana ehtii muuttua niin tekninen- kuin liiketoimintaympäristö. Muutos maksaa aina.

Ketterät kehitysmenetelmät lupaavat testattuja, laadultaan parempia ohjelmistoja. Lisäksi nämä saadaan tuotantoon haukkumaan hintaansa takaisin ennätysajassa ja vielä budjetin puitteissa. Kun tähän toimintamalliin lisätään vielä käyttäjäkeskeisen suunnittelun menetelmät, jolloin saadaan miellyttävä ja selkeä käyttöliittymä sekä järjestelmän ja käyttäjän välinen vuorovaikutus hiottua huippuunsa, niin mitäpä muuta voisi edes toivoa. Oletko kuitenkaan ollut mukana tai vieressä todistamassa helppokäyttöisten ohjelmistojen syntyä ketterien menetelmien avulla? Oletko kenties kohdannut projekteja, joissa on soudettu ja huovattu ilman tietoa siitä, mitä on saatu aikaan, missä nyt ollaan ja mikä on päämäärä? Jos haluat välttää tämän, käy läpi ennen seuraavaa projektia oheinen ”tarkistuslista”, jolloin sekä ketterän kehityksen että käyttäjäläheisen suunnittelun arvolupaukset toteutuvat todennäköisemmin.

Perusasiat

Perusasiat ovat aina yksinkertaisia. Ketterässä kehityksessä tulee julkaista säännöllisesti, lyhyin väliajoin ohjelmiston osia, jotka voidaan antaa jollekin käytettäväksi. Ohjelmiston käyttö voi olla oikea asiakasjulkaisu, sisäinen henkilökuntajulkaisu tai julkaisu kehitystiimeistä riippumattomille testi- tai laatuorganisaatiolle. Yhtä kaikki, jos projekti ei julkaise ensimmäisestä kuukaudesta lähtien säännöllisesti, tulee pysähtyä miettimään miksi. Karrikoiden voidaan sanoa, että tarkoitus pyhittää keinot. Vaikka prosessit ja menetelmät olisivat muuten vajavaisia, kertoo kyky julkaista palan ohjelmistoa siitä, että kyseisessä projektissa on jotkut asiat huomattavasti paremmin hallussa kuin sellaisessa projektissa, joka käyttää hienoa menetelmää, mutta ei pysty julkaisemaan mitään.

Harjoittelu ja toisto ovat ketterän kehityksen kulmakiviä, sillä näin saadaan nopealla aikataululla käytyä läpi pienimuotoinen ohjelmistopro-

jekti. Samalla toimintaan syntyy tietty rytmi ja aikaa myöden prosessissa mukanaolevat ihmiset oppivat mitä heidän tulee tehdä aina kussakin vaiheessa sykliä. Näin he pystyvät kehittämään itseohjaustaan sekä yksilöinä että tiiminä.

Lyhyin välein julkaistavat ohjelmistopalat tukevat myös käyttäjäläheisen kehityksen tarpeita. Virheiden korjaaminen ja parannusehdotusten toteuttaminen on halvinta silloin, kun näitä päästään tekemään mahdollisimman aikaisin. Parannuskohteita pystytään taas parhaiten löytämään antamalla oikea ohjelmisto oikeiden käyttäjien käyttöön. Tästä lisää seuraavassa tarkistuslistan kohdassa.

Sidosryhmät

Käyttäjäläheistä ohjelmistoa ketterin menetelmin ei voi tehdä ilman ns. maksavaa asiakasta, joka antaa projektille aina kulloisenkin suunnan ja sitoumuksen paljonko tiettyyn toiminnallisuuteen ollaan (vielä) valmiita investoimaan. Yli kvartaalirajan menevissä projekteissa on aina muutoslähteenä liiketoiminnalliset strategiamuutokset, jotka edellyttävät myös liiketoimintaa edustavien ihmisten osallistumista projektinohjaukseen. Jos todetaan, että ”meidän pankki hakee ensi kvartaalista lähtien kasvua talletuksilla”, niin tällöin kilpailevien liiketoimintasektorien toimintoihin ei aseteta seuraavissa iteraatioissa niitä kaikkein suurimpia panoksia, vaan saatetaan tyytyä aiemmassa iteraatiossa tehtyyn ”karvalakkimalliin”. Suunnanvaihto voidaan tehdä iteraation vaihtuessa, koska tavoite on saada tehtyä tuotantokelpoisia ja testattuja toimintoja jokaisessa iteraatiossa.

Liiketoiminnan lisäksi tarvitaan myös aito loppukäyttäjä, joka määrittää käytettävyyden minimiehdot kullekin toiminnolle ja verifioi sen, että toiminnallisuus kattaa ne toiminnot, jotka hän tarvitsee päästäkseen tavoitteisiinsa palvelun käyttäjänä.

Jos projektilla ei ole riittävää näkyvyyttä niin liiketoimintaan kuin käyttäjiin ihmisinä, joiden kanssa tulee voida käydä ajatustenvaihtoa ja jotka osallistuvat projektiin käyttäen koekaniineina vielä keskeneräistä palvelua, on vaarana, että projekti maalaa itsensä nurkkaan. Todellisen liiketoiminta- ja loppukäyttäjävuorovaikutuksen puute johtaa useimmiten siihen, että projekti alkaa itse luoda käsityksiä siitä, mitä heidän halutaan tekevän ja millainen valmiin palvelu tulisi olla. Lopputuloksena saattaa olla ketterästi metsään lipsahtanut ”turhake”, mutta vielä todennäköisemmin lopputuloksena on liian monimutkainen ja kallis palvelu, jonka jää kesken.

Muuta asiat numeroiksi

Liiketoiminnan ja käyttäjien mukanaolo ei yksistään riitä onnistuneeseen priorisointiin. Erilaisista painotuksista johtuen, saadaan pelkästään subjektiivisilla tai kvalitatiivisilla mittareilla usein tuloksia, joissa saattaa olla sisällä merkittäviä virhelähteitä projektiohjauksen kannalta. Näitä virhelähteitä voidaan pienentää muuttamalla subjektiiviset arviot numeroiksi, jotka pohjautuvat joko liiketoiminnallisiin tunnuslukuihin tai muihin mitattaviin asioihin.

Useat ketterän kehityksen menetelmät painottavat liiketoiminnallisen lisäarvon tärkeyttä. Projektin olemassaolon edellytys on se, että projekti tuottaa mahdollisimman paljon ja tehokkaasti liiketoiminnalle välineitä liiketoiminnan harjoittamiseen. Joissain tapauksissa lisäarvo voidaan laskea helposti esim. tietyn toiminnallisuuden toteutuksesta saatavien hyötyjen ja toteuttamattomasta toiminnallisuudesta aiheutuvien kustannusten summana. Tällöin saadaan konkreettiset ”kepit ja porkkanat”, jotka seuraavat eri priorisointiskenaarioita.

Myös käytettävyys tulee muuttaa numeroiksi. Usein kuulee puhuvan ”hyvästä ja huonosta” käytettävyydestä ilman sen tarkempaa määrittelyä, mitä itse käytettävyys on. Käytettävyys ei ole jokin tietty kohta käyttöliittymässä, jokin hieno ratkaisu, vaan käytettävyys tulee määritellä käyttäjän näkökulmasta ja löytää jokin mitattava suure ilmentämään sitä. Käytettävyydelle voidaan löytää suureita, kuten opittavuus, muistettavuus, virheiden määrä ja transaktion läpimenoaika. Tämänkaltaisen operationalisoinnin jälkeen olemme aivan eri tasolla käyttäjäläheisessä suunnittelussa kuin perinteisellä ”ravista hihasta” –menetelmällä. Vaara siitä, että ”käytettävyys” alkaa elämään omaa elämää pienenee.

”Myös käytettävyys tulee muuttaa numeroiksi.”

Kaikkein haastavin muunnos numeroiksi tehdään rajahyödyn kohdalla, kun yritetään vastata kysymykseen ”paljonko tästä kannattaa maksaa”. Useasti tämän tyyppinen tarkastelu voi olla niin vaativa ja luonteeltaan akateeminen, että on parasta pitäytyä siinä, että muistetaan panoksen ja siitä saadun hyödyn välisen suhteen olevan epälineaarinen. Käytetään siis tavallista heinänte-kojärkeä, jos investoinnit niin ominaisuuksiin kuin niiden käytettävyyteen alkavat tuntua kohtuuttomilta suhteessa havaittavaan hyötyyn.

Aloita yksinkertaisesti (ja jatka samaa rataa loppuun asti)

Jos mahdollista, aloita sekä ketterän kehityksen että käyttäjäläheisen toimintamallin harjoittelu pienessä projektissa. Näin saadut hyödyt ovat suuremmat, kun toimintaan osallistuvat henkilöt pääsevät lähemmäksi tapahtumien ydintä ja kaikki ovat tietoisia lähes kaikista asioista, joita ympärillä tapahtuu. Tärkeää on olla tietoinen myös niistä asioista, joita ei itse suorita, koska tämä tieto saattaa auttaa ymmärtämään paremmin oman osuuden merkityksen ”isossa kuvassa” ja siten edesauttaa itseohjautuvuuden kehit-

tymistä. Vasta sen jälkeen, kun pieni ydin osaa toimia uudessa viitekehyksessä on aika skaalata projektia vastaamaan kokonaistarvetta.

Toinen asia, jossa kannattaa olla varovainen, on teknisten ratkaisujen toteutus. Mitä vaikeampi arkkitehtuuri ja kovempi pyrkimys yleiskäyttöiseen, laajennettavaan ja helposti ylläpidettävään ratkaisuun, sitä suurempi on riski epäonnistua. Tämä koskee erityisesti kunkin toiminnallisuuden ensimmäisiä iteraatioita. Nyrkkisääntönä voisi sanoa, että ensimmäisessä iteraatiossa kannattaa asettaa tavoitteeksi saada aikaan, jotain joka toimii ja vasta sitten murehtia siitä, että oliko ratkaisu tyylikäs. Refaktorointi, ohjelman sisäisen rakenteen parantaminen toiminnan pysyessä samana, ei ole merkki epäonnistuneesta suunnittelusta, vaan siitä, että näkemys tarkentuu jokaisessa iteraatiossa.

Yksinkertaisuus on valttia myös ohjelmistotuotantoprosessissa ja siinä tehtävissä suunnittelukuvauksissa. Useimmiten paras ratkaisu on alkaa ns. puhtaalta pöydältä ts. tehdä jokin pieni kokonaisuus ensimmäisessä lyhyessä iteraatiossa ilman prosessia ja prosessin vaatimia artifakteja. Ainoa sääntö tällöin on, että julkaisu tai demo tapahtuu iteraation päätteeksi. Kehitystiimi tulee tunnistamaan viimeistään kolmannessa iteraatiossa ne prosessivaiheet ja suunnittelukuvaukset, joita se todella tarvitsee työssään. Näin vältymme pöhöttyneeltä toimintamallilta, jossa useimmat roolit, dokumentit ja prosessien vaiheet ovat kuolleita kirjaimia.

Lähtökohtaisesti ketterät ja käyttäjäläheiset menetelmät ovat haastavia projektin muutoshallinnan kannalta. Molemmat suosivat iteraatioita ja toivottavat tervetulleiksi käyttäjäpalautteesta saatavat muutokset. Näin ollen muutoshallinnan on oltava kunnossa. Tärkein väline muutoshallintaan on ”product backlog”, joka pitää sisällään kaiken tekemättömän työn, myös muutokset ja korjaukset aiemmin tehtyihin toimintoihin. Product backlog on pidettävä aina kunnossa, sillä se on projektin projektisuunnitelma ja muutoksenhallintaväline samanaikaisesti. Ilman product backlogia ei tiedetä mitä on saatu aikaan, missä nyt ollaan ja mitä tehdään seuraavaksi.

Pelkästää product backlog ei riitä muutoshallinnan onnistumiseen. Epäyhteneväisyyksistä johtuvia muutoksia, oli ne sitten peräisin erilaisista käyttöliittymäratkaisuista tai useamman toteuttajan kirjoittamasta erilaisista koodirakenteista, tulee minimoida pitämällä säännöllisesti koodi- ja käyttöliittymäkatselmointeja. Toinen tärkeä mekanismi turhien (=ei käyttäjälähtöisten) muutosten välttämiseen on ottaa projektissa käyttöön heti alusta alkaen riittävän tarkat tyyli- ja ohjelmointiohjeet. Tyyliohjeissa määritellään miltä jokin asia käyttöliittymässä näyttää ja millä tavoin se reagoi käyttäjän toimiin. Ohjelmointiohjeet taas määrittelevät, miten ohjelmat tulisi toteuttaa ja koodi dokumentoida. Näin vähennämme itse aiheutetun muutospaineen määrää.

Yhteenveto

Jos havaitsit, että tarkistuslistalla oli asioita, jotka projekteissasi eivät ole toteutuneet, niin prosessin kehittäminen todennäköisesti kannattaa. Pidä kehittämistyössä kuitenkin jalat maassa ja pää kylmänä. Terve järki vie pidemmälle kuin kirjaviisaus ja fundamentalismi.