Teksti: Tomi Juhola

Käyttäjäkeskeinen suunnittelu ja ketteryys

– Ristiriitaiset lähestymistavat vai arvokas synergia

Käyttäjäkeskeinen suunnittelu (User-centered design) ja ketterä ohjelmistokehitys ovat yleisesti käytössä modernissa ohjelmistokehityksessä. Molemmat ovat iteratiivisia ja nostavat asiakkaan ja käyttäjät keskeiseen rooliin ohjelmistotuotantoprosessissa. Yhteisistä päämääristä huolimatta ketterien menetelmistöjen ja käyttäjäkeskeisen suunnittelun yhdistäminen on harvinaista. Ovatko lähestymistavat siis keskenään ristiriidassa vai eikö niiden yhdistämisestä seuraa riittävää hyötyä?

Käyttäjäkeskeinen suunnittelu on 1980-luvun puolivälissä kehitetty kokoelma ohjelmakehityskäytäntöjä ja myös oma ohjelmistokehitysfilosofiansa. Kantava ajatus on käyttäjän tuominen keskeiseksi osaksi ohjelmistokehitysprosessia. Käyttäjä yhdistetään prosessiin haastattelemalla tulevia käyttäjiä, tarkkailemalla heitä työssään sekä tutkimalla ja analysoimalla käyttäjäryhmän tietoja. Käyttäjälle myös näytetään prototyyppejä järjestelmästä ja pyydetään häntä tekemään prototyypeillä toimintoja. Näistä tilanteista saatujen palautteiden, tietojen ja havaintojen avulla löydetään uusia vaatimuksia ohjelmistolle, parannuksia käyttöliittymään ja ymmärretään eri ominaisuuksien tärkeys paremmin.

Yhteneväisyyksiäkin toki löytyy näistä malleista. Molemmat haluavat luoda useita prototyyppejä, joista pyydetään palautetta ja tämän perusteella jatketaan kehitystyötä eteenpäin.

Suurin osa tästä työstä tehdään ennen varsinaisen ohjelmistokehitystyön aloittamista, joka taas toteutetaan iteratiivisesti pitäen kohdekäyttäjät jatkuvasti ohjelmistokehittäjien mielissä. Tässä auttaa esim. persoonien määrittäminen, joka perustuu keskeisten käyttäjäryhmien tunnistamiseen ja niiden personoimiseen luomalla kuvitteellisia esimerkkikäyttäjiä. Esimerkkikäyttäjistä kehittäjät tietävät lähes kaiken: perhesuhteet, harrastukset, taidot ja rajoitukset. Esimerkiksi Yahoo!:n web-sovellusten suunnittelussa nämä esimerkkihenkilöt ovat keskeisessä roolissa päätettäessä tuoteen ominaisuuksien tarpeellisuudesta, käyttöliittymän monipuolisuudesta ja jopa toteutustekniikasta.

Ketterä ohjelmistokehitys on yhteisnimitys kevyille ja muutokseen mukautuville ohjelmistokehitysprosesseille. Ketterille menetelmistöille tärkeimmät arvot on määritelty Ketterän ohjelmistokehityksen manifestissa (Agile manifesto). Tärkeimpiä asioita ohjelmistokehityksessä ovat yksilöt ja näiden välinen vuorovaikutus, toimiva ohjelmisto, yhteistyö asiakkaan kanssa sekä muutokseen vastaaminen. Näiden arvojen pohjalta olisi hankala suoraan lähteä kehittämään ohjelmistoja, joten manifeston yhteydessä on myös määritelty 12 periaateetta, jotka ohjaavat tarkemmin työskentelyä ohjelmistokehitysprojekteissa.

Ketteryys vs. käyttäjäkeskeisyys

Ketterät menetelmistöt, eteenkin Extreme programming painottavat usein etukäteissuunnittelun merkityksettömyyttä ja sen aiheuttamaa hukkatyötä ja muutosjäykkyyttä. Käyttäjäkeskeinen suunnittelu taas lähtee ajatuksesta, että ennenkuin lähdetään tekemään riviäkään toteutusta, tulisi olla tarkka kuva kohdekäyttäjistä ja jo ensimmäiset käyttöliittymäluonnokset.valmiina. Kent Beck, Extreme programmingin kehittäjä, piti tätä etukäteissuunnittelua käyttäjäkeskeisen suunnittelun suurimpana ongelmakohtana, koska se tekee käytettävyyssuunnittelijoista projektien pullonkaulan ja samalla keskittää päätöksentekoa.

Ketterät menetelmistöt myös painottavat dokumentaation keveyttä, jopa niinkin paljon, että valloilla on yleinen harhakäsitys dokumenttien puuttumisesta kokonaan ketteristä ohjelmistokehitysprojekteista. Käyttäjäkeskeinen suunnittelu taas pohjaa pitkälti erilaisiin tutkimusdokumentteihin, käyttöliittymäsuunnitelmiin ja esimerkkihenkilöjulisteisiin.

Yhteneväisyyksiäkin toki löytyy näistä malleista. Molemmat haluavat luoda useita prototyyppejä, joista pyydetään palautetta ja tämän perusteella jatketaan kehitystyötä eteenpäin. Käyttäjäkeskeinen suunnittelu hakee tämän palautteen käyttäjiltä, Extreme programming pyytää palautetta asiakkaalta, joka vastaa myös käyttäjien tarpeista ja Scrum hakee tämän palautteen jokaisen sprintin eli iteraation jälkeisestä demo-tilaisuudesta, jossa voi olla paikalla sekä liiketoiminnan edustajia että varsinaisia käyttäjiä. Palaute tulee siis ketterissä menetelmistöissä useimmiten asiakkaan vastuulliselta edustajalta, joka toimii parhaansa mukaan välittääkseen käyttäjien tarpeet, kun taas käyttäjäkeskeinen suunnittelu hakee palautetta varsinaisilta loppukäyttäjiltä. Molempien mallien samanaikainen käyttö tuo luonnollisesti enemmän

palautetta ja antaa paremman kuvan niin asiakkaan kuin loppukäyttäjien tarpeista.

Sekä käyttäjäkeskeisessä suunnittelussa että ketterissä menetelmistöissä on käyttäjä tai asiakas yhtenä keskeisimpänä roolina kehitysprosessissa. Heidän tulee ottaa osaa ohjelmistokehitykseen, jotta kehittäjät tuottavat ohjelmiston, joka vastaa heidän eksplisiittisiä sekä implisiittisiä tarpeitaan mahdollisimman hyvin. Jotta tätä ohjelmistoa voidaan tarkistaa tasaisin väliajoin, ovat molemmat lähestymistavat iteratiivisia.

Ohjelmistot sopiviksi palautteella

Palaute on siis molemmille hyvin keskeinen ohjelmistokehitystä ohjaava tekijä. Ketterät menetelmistöt aloittavat kehityksen tärkeimmistä korkean tason vaatimuksista, tulkitsevat niitä sekä kyselevät asiakkaalta lisätietoja ja luovat näiden perusteella jo heti ensimmäisessä iteraatiossa toimivan ohjelmiston, joka annetaan asiakkaalle testattavaksi. Asiakkaan palautteen mukaan ohjelmistoa muokataan ja rakennetaan edelleen, kunnes päästään asiakkaan tarpeet täyttävään sovellukseen. Erityisen tuskallista tämä saattaa olla käyttöliittymän osalta, jonka hiominen tyydyttävään muotoon voi kestää melko pitkään. Myös käyttöliittymän ja konseptien yhtenäisyyden kannalta tämä malli vaatii erityistä tarkkaavaisuutta. Tässä suhteessa käyttäjäkeskeinen suunnittelu voi avittaa ketteriä menetelmistöjä.

Käyttäjäkeskeisen suunnittelun käyttö projektin alussa mahdollistaa yhtenäisen käyttöliittymän suunnittelun, tekemättä kuitenkaan suuria ohjelmistokehitystä rajoittavia suunnittelupäätöksiä. Sen sijaan että iteroiden ja testaten etsitään käytettävyydeltään oikeanlaista käyttöliittymää, voidaan tämä saada suurinpiirtein kohdalleen heti ensimmäisestä iteraatiosta alkaen. Esimerkiksi käyttäen paperiprototyyppejä käyttöliittymien hahmottelemiseen ja käyttäjien kanssa kommunikoimiseen voidaan nopeasti saada selville käyttöliittymän yleinen rakenne. Paperiprototyypithän eivät ole turhia ja ohjelmiston kannalta merkityksettömiä dokumentteja vaan arvokkaita ja testattavia käytettävyysmalleja, joiden pohjalta voidaan helposti luoda toimiva käyttöliittymä.

Persoonien määrittelemisen tehoa on myöskin kokeiltu ketterissä ohjelmistokehitysprojekteissa ja kokemukset ovat olleet lupaavia. Näkyvillä paikoilla sijaitsevien persoonien on nähty muistuttavan kehittäjiä keille he ovat ohjelmistoa tekemässä. Samalla päämäärä ja kokonaiskuva pysyy selkeämpänä. Nämä esimerkkikäyttäjät myöskin helpottavat päätöksentekoa, koska kehittäjillä on tarkempi kuva käyttäjistä ja heidän tarpeistaan. Toisaalta nämä persoonat on melko helppo unohtaa teknisten asioiden ollessa päällimmäisenä, joten persoonien määritteleminen ei yksinään varmista, että kehittäjillä on loppukäyttäjä mielessä päätöksiä tehdessään.

Sudenkuopista synergiaan

Kokemukset näiden menetelmien yhdistämisestä ovat olleet positiivisia, joskin luonnollisesti myös ongelmia on havaittu. Yksi yleisimmistä ongelmista näiden kahden menetelmän yhdistämisessä on ollut käytettävyyssuunnittelijoiden ja kehittäjien välinen kommunikointi. Molemmilla tahoilla on havaittu taipumusta suojautua toisen ryhmän tekemiä päätöksiä vastaan. Kehittäjien tehdessä päätöksi käyttöliittymän tiimoilta vastustavat käytettävyyssuunnittelijat automaattisesti näitä. Sama pätee myös toisinpäin ja tämä valtataistelu johtaa helposti haitalliseen ”me ja ne muut” ajatteluun.

Jotta nämä menetelmät toimisivat mahdollisimman hyvin yhdessä, tarvitsee näihin muutamiin ongelmakohtiin kiinnittää huomiota. Ensinnäkin kehittäjien ja käytettävyyssuunnittelijoiden välistä eroa tulisi kaventaa ja parantaa näiden kommunikaatiota. Ihanteellista olisi luonnollisesti, jos kehittäjätkin hallitsisivat käytettävyyssuunnittelun periaatteet. Toiseksikin käyttäjien sitouttamiseen tulisi panostaa. Käyttäjät pitäisi saada mukaan lähes koko kehitysprojektin ajaksi ja heiltä täytyy saada arvokasta palautetta käytettävyydestä.

Erilaisilla projekteilla on erilaiset tarpeet. Tämä on hyvä ottaa huomioon käyttäjäkeskeistä suunnittelua ja ketteriä menetelmistöjä yhdistettäessä. Joissakin projekteissa käyttäjäkeskeisestä suunnittelusta ei ole hyötyä. Joissakin projekteissa ei ole aikaa tehdä kattavaa tutkimusta käyttäjistä, vaan ensisijaisena tarkoituksena on tuoda mahdollisimman nopeasti ohjelman ensimmäinen versio markkinoille. Joissakin projekteissa on hyvä suunnitella 90% käyttöliittymästä valmiiksi etukäteen, toisissa taas riittää karkea malli 60%:sta käyttöliittymää. Joskus taas ei ole mahdollisuutta saada oikeita loppukäyttäjiä testaamaan ohjelmistoa ja paperiprototyyppejä. Käyttäjäkeskeisen suunnittelun käytölle on oltava selkeä tarve, jotta sitä kannattaa yrittää yhdistää ketteriin menetelmistöihin. Ilman tarpeiden ymmärtämistä voi tämä tekniikka hankaloittaa ohjelmistokehitystä.

Käyttäjäkeskeinen suunnittelu ja ketterät ohjelmistokehitysmenetelmät ovat siis yhteensovitettavissa olevia tekniikoita. Niiden oikeanlainen yhdistäminen tuo usein lisäarvoa asiakkaalle, vähentää ominaisuuksien uudelleentyöstämistä ja helpottaa ohjelmistokehitystyötä. Eri tekniikoiden tuntemusta tosin vaaditaan, jotta niitä voidaan käyttää tehokkaasti yhdessä. Kaikenkaikkiaan jokaisessa ohjelmistokehitysprojektissa tulisi pääasiallisena tavoitteena olla asiakkaan ja käyttäjien tarpeiden täyttäminen sekä parhaan mahdollisen ohjelmiston kehittäminen -- ei puhtaiden kehittämisprosessien orjallinen noudattaminen, vaan prosessinkin ketterä räätälöinti kulloisenkin projektin tavoitteet parhaiten täyttäväksi.

Käyttäjäkeskeisen suunnittelun käyttö projektin alussa mahdollistaa yhtenäisen käyttöliittymän suunnittelun, tekemättä kuitenkaan suuria ohjelmistokehitystä rajoittavia suunnittelupäätöksiä.

Käyttäjäkeskeinen suunnittelu ja ketterät ohjelmistokehitysmenetelmät ovat siis yhteensovitettavissa olevia tekniikoita.