Ohjelmistotuotantomenetelmien kehittyminen 1950-luvulta nykypäivään

Ohjelmistotuotanto on ollut olemassa 1950-luvulta nykypäivään saakka. Termi ohjelmistotuotanto alkoi esiintymään kirjallisuudessa 60-luvun puolivälissä. 70-luvulla tehokkaampien tietokoneiden markkinoille tulo mahdollisti monimutkaisempien ohjelmistojen kehittämisen ja tarve organisoiduille ohjelmistotuotantoprosesseille syntyi.

Eri aikakausilla ohjelmistonkehittäjät ovat toteuttaneet sen hetkisiä parhaina pidettyjä työskentelytapoja. Perinteiset ohjelmistotuotantomallit ovat pohjautuneet suunnitelmavetoiseen ohjelmistokehitykseen, jossa edetään lineaarisesti. Vesiputousmalli on yksi tällainen tuotantomalli ja sen ongelmien paljastuessa inkrementaaliset ja iteratiiviset menetelmät saivat jalansijaa. 2000-luvulle tultaessa kehitettiin niin kutsuttuja ketteriä menetelmiä ja ne ovatkin saaneet jalansijan jo jopa suosituimpana tuotantomallina ohjelmistojen kehittämiseen.

Ohjelmistoarkkitehtuurin sisällyttäminen ketteriin ohjelmistotuotantomenetelmiin

Arkkitehtuurivetoisen ohjelmistotuotannon katsotaan usein olevan ristiriidassa ketterien menetelmien kanssa. Arkkitehtuurin suunnitteluun vaadittu aika nähdään viivykkeenä tärkeimmälle tavoitteelle, toiminnallisuutta sisältävän ohjelmiston toimitukselle asiakkaalle. Ketterissä menetelmissä arkkitehtuurin ajatellaan muotoutuvan kehitystyön edetessä.

Tämä on herättänyt kritiikkiä arkkitehtuuriyhteisön piirissä, jossa ketterien menetelmien mallin ei katsota ottavan tarpeeksi hyvin huomioon kehitysympäristön ja kohdelaitteiston rajoitteita ja sen katsotaan tuovan ikäviä yllätyksiä kehityksen edetessä.

Ketterien menetelmien lisäksi on luotu käytänteitä, joissa arkkitehtuurisia ratkaisuja tehdään rinnakkain varsinaisen sovelluskehityksen kanssa. Tällaisia käytänteitä ovat nollasprintit, suunnittelupiikit ja arkkitehtuuritarinoiden lisääminen käyttäjätarinoiden rinnalle.

Jatkuva eksperimentointi ohjelmistokehityksen tukena

Eksperimentointi on eritoten startup-yritysten käyttämä tapa tuottaa ohjelmistoja, jossa kehitys jatkuu syklissä ja perinteiseen ohjelmistojen kehitysmalliin verrattuna tarkoituksena on hyödyntää jatkuvaa käyttöönottoa, eli toimittaa uusi ominaisuus lyhyellä aikavälillä asiakkaan käyttöön. Myös muunlaiset yritykset voivat saada lisäarvoa tästä tavasta tuottaa ohjelmistoja.

Syklissä ensimmäinen askel on idean löytyminen. Idea on jokin toiminnallisuus, jonka uskotaan tuovan arvoa asiakkaalle ja sitä kautta liikevaihtoa yritykselle. Hypoteesi tarpeesta sille asetetaan. Toiminnallisuudesta luodaan minimituote, joka sisältää vain olennaiset ominaisuudet uuden toiminnallisuuden takaamiseksi. Jatkuvan käyttöönoton mukaisesti tuote toimitetaan asiakkaan käyttöön ja mitataan, miten sovelluksen käyttö muuttuu uuden ollessa implementoitu. Mittaustuloksista riippuen uusi ominaisuus voidaan ottaa pysyväksi osaksi ohjelmistoa, tai se voidaan hylätä.

Eksperimentointi on toimiva tapa tuottaa ohjelmistoa erityisesti, kun sovellus on verkossa toimiva. Asiakkaat saavat uuden version käyttöönsä ilman vaadittavia sovelluspäivityksiä. Yhä suurempi osa sovelluksista on tätä nykyä verkossa toimivia ja eksperimentoinnin asema vankistuu.

Scrumban-menetelmän käyttö ketterässä ohjelmistokehityksessä

2000-luvulla ohjelmistokehityksen suosituimmaksi paradigmaksi on noussut ketterä ohjelmistokehitys. Suosituimmiksi viitekehyksiksi ovat nousseet Scrum ja Kanban. Molemmat noudattavat ketterän kehityksen ja lean-ajattelun ydinajatuksia, nopeaa muutoksen vastaamista ja jatkuvaa kehitystä.

Scrumban on näiden kahden viitekehyksen ominaisuuksia yhdistelevä menetelmä. Sen katsotaan sopivan projekteihin, jotka ovat nopeasti muuttuvan lisäksi arvaamattomia. Scrumbanin kehittäjien mielestä Scrum asettaa liian tiukkoja rajoitteita työskentelyprosessille, eritoten sen rajatulle ajankäytölle, Kanbanin taas ollessa liian rajoitteeton. Scrumban yksinkertaisesti pyrkii löytämään molemmista viitekehyksistä parhaat puolet.

Sopivaa menetelmää tietyn ohjelmiston kehittämiseen ei automaattisesti ole olemassa. Scrum, Kanban ja niiden yhdistelmä eivät ole valmiiksi määriteltyjä prosesseja, mutta oikeissa olosuhteissa ne voivat olla toimiva työkalu prosessin kehittämiseen.