Asiakaslähtöinen ohjelmistosuunnittelu ketterien menetelmien yhteydessä

31.12.2021

Ketterillä ohjelmistonkehitysmenetelmillä tarkoitetaan menetelmiä, joissa ohjelmistokehitys tehdään pieninä tehtävinä tai työjaksoina ja vaatimuksia ja määrittelyä tarkennetaan koko kehitysprojektin ajan. Tällöin projektin aikana ilmeneviin muutoksiin voidaan reagoida nopeasti ja keskittyä aina olennaisiin asioihin. Tyypillisiä ketteriä menetelmiä ovat mm. Scrum ja Kanban. Scrum sopii ennalta sovittujen kehitysjaksojen (Sprint) vuoksi hyvin uuden ohjelmiston kehittämiseen ja Kanban on käytössä tyypillisesti sellaisissa tilanteissa, joissa kehitysjakson tehtäviä ei voida kiinnittää esimerkiksi tiimille tulevien satunnaisten ylläpitotehtävien vuoksi.

Tehtävälista

Molemmissa edellä mainituissa menetelmissä käytetään jatkuvasti projektin aikana päivitettävää tehtävälistaa (Product backlog), jossa päällimmäisenä ovat aina prioriteetiltaan tärkeimmät tehtävät. Tarkoituksena on, että keskitytään aina olennaisimpiin tehtäviin. Toisaalta prioriteetit voivat muuttua projektin aikana esimerkiksi saadun palautteen perusteella.

Loppukäyttäjän tarpeet voivat tällaisessa tehtävälistassa kadota, kun tehtäväkuvauksessa keskitytään ohjeistamaan suunnittelijaa. Viesti siitä mitä loppukäyttäjä todella tarvitsee ei välity ja väärinymmärrysten mahdollisuus kasvaa.

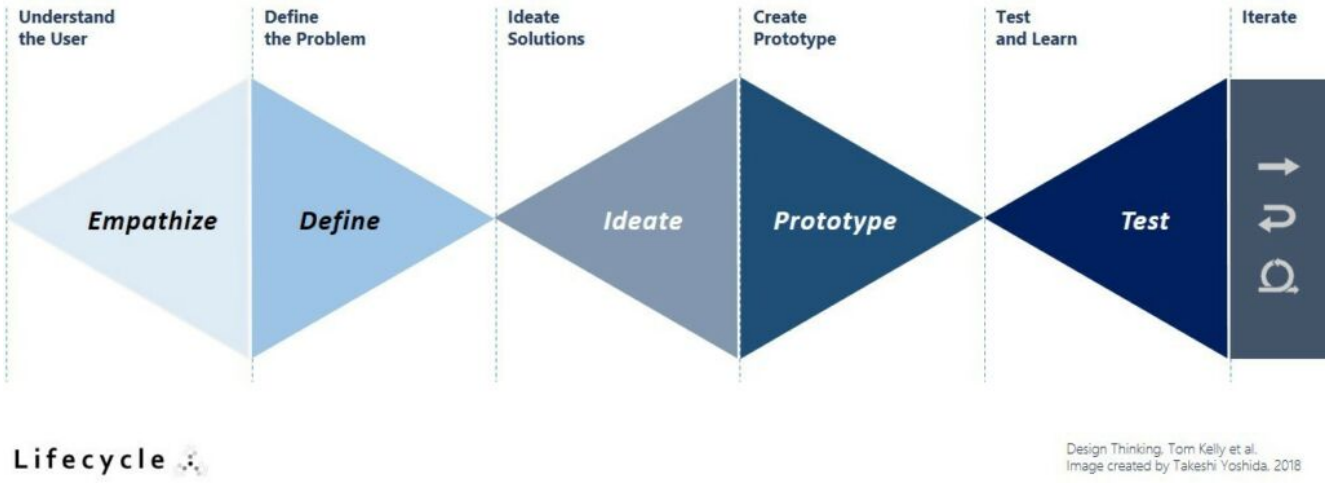
Suunnitteluajattelu lähtökohtana

Yksi asiakaslähtöisyyden parantamiseen ketterien menetelmien yhteydessä esitetty toimintatapa on suunnitteluajattelu, joka tarjoaa työkaluja ja menetelmiä tuotteen käyttäjän ymmärtämiseen ja asiakaslähtöiseen ongelmanratkaisuun.

Perinteisissä ohjelmistokehitysmenetelmissä loppukäyttäjän näkökulmaa on usein tuotu suunnitteluun mukaan pohtimalla käyttötapauksia eli kuka ohjelmistoa tulee käyttämään ja miten. Suunnitteluajattelussa tätä viedään pidemmälle pohtimalla perusteellisemmin loppukäyttäjän näkökulmaa, pyrkimällä asettumaan tämän asemaan.

Kuviossa 1 on kuvattu suunnitteluajattelun prosessia. Prosessi koostuu viidestä eri vaiheesta: Käyttäjän tarpeiden havainnoiminen (Empathize), Käyttäjän ongelmien määrittely (Define), Ratkaisujen ideointi (Ideate), Prototyypin rakentaminen (Prototype) ja Ratkaisujen testaaminen (Test). Prosessi etenee lineaarisesti, mutta

aikaisempiin vaiheisiin voidaan palata, jos todetaan että ratkaisu ei toimi tai huomataan että on ymmärretty jotain väärin.



Kuvio 1. Suunnitteluajattelun prosessi.

Täsmällisemmät tehtäväkuvaukset

Ohjelmistokehityksessä toimitusketju voi toisinaan olla hyvin pitkä. Esimerkkinä yritys A myy ohjelmistotuotetta asiakasyrityksille, joiden työntekijät ovat sovelluksen loppukäyttäjiä. Yritys A ostaa ohjelmistokehityksen yritykseltä B, jolla edelleen voi olla alihankkijoita, joiden työntekijät suorittavat varsinaisen ohjelmistokehityksen. Tällöin ohjelmiston loppukäyttäjän ja ohjelmistokehittäjän välillä on monta porrasta ja tiedonkulun ongelmat voivat johtaa siihen, että ratkaisut eivät täysin vastaa sitä, mitä ketjun loppupäässä odotetaan.

Ajatuksena on, että prosessia käytetään soveltuviin tehtävälistan tehtäviin ja saadaan niiden tehtäväkuvausta laajennettua vastaamaan paremmin todellista tarvetta. Menetelmää voidaan käyttää esimerkiksi kehitysjaksojen välissä parantamaan seuraavaan kehitysjaksoon tulevien tehtävien kuvauksia. Näin voidaan varmistaa, että seuraavaksi toteutettavien ominaisuuksien kanssa mennään varmasti oikeaan suuntaan ja loppukäyttäjän tarpeet tulevat huomioiduiksi.

Asiakaslähtöisessä ohjelmistosuunnittelussa oleellista on se, että ymmärretään ohjelmiston loppukäyttäjän tarpeet ja osataan asettua suunnittelutyössä loppukäyttäjän asemaan. Ohjelmistoprojekteja toisinaan vaivaavat epäonnistumiset vältetään saumattomalla yhteistyöllä ja halulla ymmärtää asiakkaan puolesta ratkottavat ongelmat perin pohjin.

Juha Yli-Hemminki
Lehtori
SeAMK

Lähteitä:

Try Design Thinking + Scrum. The Lean & Agile Practitioner | by Takeshi Yoshida | MediumThe New User

Story Backlog is a Map – Jeff Patton & Associates (jpattonassociates.com)Design Thinking Bootleg — Stanford d.school