Viikko 7, teht. 4

Ryhmätyö ohjelmistokehityksessä -referaatti

Ohjelmistokehitys on hyvinkin paljon työskentelyä ryhmässä, kun nykyään tarvitaan niin monenlaisia teknisiä taitoja, ettei yksin niistä voi selvitä. Lisäksi ohjelmistokehitys on nopeutunut ja ryhmässä saa enemmän aikaan. Ryhmätöissä tarvitaan myös kommunikointitaitoja ja ne ovat oleellinen osa ohjelmistokehityksen etenemiseen.

Ketterissä menetelmissä työ yleensä pilkotaan pienempiin osiin. Tässä keskitytään scrumiin ja extreme programmingiin (XP), joissa molemmissa on samoja piirteitä. Jälkimmäisessä kehutaan esim. pariohjelmointia, kun koodin laatu paranee ja molemmat voivat opettaa toista. Molemmissa lyhyet työskentelyjaksot (iteraatiot) ovat tärkeitä eli tehdään työ pienissä paloissa ja jokaiselle palalla on selkeä tavoite. Scrum painottaa myös tiimin tasa-arvoisuutta.

Tietenkin myös tiimin jäsenten personalla on väliä. Hyvinä piirteinä pidetään: joustavuus/mukautuvuus, hyvät kommunikointitaidot, älykkyys sekä mukavuus. Ryhmätyöskentelyä voi myös kehittää ja sitä pitäisi tehdä

Viikko 7, teht. 7

Ohjelmistoarkkitehtuurin sisällyttäminen ketteriin ohjelmistotuotantomenetelmiin -referaatti

Ohjelmistoarkkitetuurin merkitys on muuttunut ketterien menetelmien johdosta. Alun suunnitteluprosessi ei ole yhtä raskas kuin aikaisemmin, mutta toisaalta se aiheuttaa ongelmia sen suhteen, ettei välttämättä ole tarpeeksi selkeitä suuntaviivoja toteutukselle, esimerkiksi XP kieltää tuottamasta koodia, jota ei tarvita kyseisessä iteraatiossa.

Ketterässä ohjelmistokehityksessä suunnittelun voi jakaa karkeesti kolmeen osaan: Julkaisun suunnittelu, iteraation suunnittelu, kehitysjakson aikana tapahtuva suunnittelu. Viimeiseen on vähän virallisia ohjeita ja siinä luotetaan enemmän tiimin itseohjaukseen.

Joskus suunnitelmattomuus voi johtaa suuriin ongelmiin ja pahimmassa tapauksessa koko sovellus saatetaan joutua aloittamaan alusta. Kaikille projekteille ei sovi samanlaiset kehitystavat. Arkkitehtuurin sisällyttämistä voidaan parantaa esim. sprint-0:lla (aluksi käydään “kunnollinen” suunnitteluvaihe), arkkitehtuuritarinoilla (kuten user storyt, mutta kehittäjät tekevät nämä), arkkitehtuurijaksoilla (kokonaisia iteraatioita varataan arkkitehtuurille), suunnittelupiikeillä (iteraatiossa varataan jakso arkkitehtuurille) ja arkkitehtuuri erillisenä prosessina (esim. eri tiimi vastaa arkkitehtuurista).

Viikko 7, teht. 8

Metriikat käytänteiden tukena ohjelmiston laadun arvioimisessa -referaatti

Virheet ohjelmistossa voivat käydä kalliiksi. Metriikoita tutkiessa on todettu niiden tuoma lisäarvo, varsinkin kun yhdistellään erillaisia metriikoita. Muutamia hyödyllisiksi todettuja metriikoita ovat: koodikirnu, verkkoanalyysi, testikattavuus ja mutaatiotestaus. Koodikirnulla arvioidaan ohjelmiston muutoksien vaikutusta ohjelmiston virheherkkyyteen. Verkkoanalyysillä tutkitaan ohjelmiston komponenttien riippuvuuksien vaikutusta ohjelmiston virhealttiuteen. Testikattavuudella ja mutaatiotestauksella analysoidaan ohjelmiston lähdekoodin testien tehokkuutta

sekä laadukkuutta.

Muutama pointti testikattavuudesta: 100% kattavuuteen ei yleensä kannata pyrkiä. Testikattavuus on erittäin hyödyllinen erityisesti uusille kehittäjille ja etätyötä tekeville. Yleensä ohjelmiston kanssa alusta alkaen olevat kehittäjät tekevät kattavampia testejä kuin ylläpitäjät tai myöhemmin mukaan tulleet.

Käyttäjän eli ohjelmoijan metriikoita ovat: ketterä kehitys, pariohjelmointi, testilähtöinen kehitys. Viimeiseen liittyen, tutkimukset tuottavuuden kannalta ovat olleet ristiriitaisia. Hyvän koodin laatuattribuuteiksi voidaan luetella muun muassa lähdekoodin kapselointi (algoritmien yksityiskohtien piilottaminen), koheesio (komponenttien yksi vastuu), riippuvuuksien vähäisyys, toistettavuus, testattavuus ja selkeys.

Viikko 7, teht. 9

Johtaminen perinteisissä ja ketterissä ohjelmistoprojekteissa -referaatti

Ohjelmistoprojektien luonne on, että niillä on määritelty tavoite, ne ovat väliaikisia, niillä on määritelty alku ja loppu, kertaluonteisia, niillä on budjetti ja tietty tiimi. On olemassa ohjaavaa sekä voimaannuttavaa johtamista, yleensä jälkimmäistä pidetään parempana, erityisesti kun kyseessä on kokenut tiimi, mutta ohjaavaa tyyliä saatetaan tarvita esimerkiksi kokemattomien ohjelmoijien kanssa. Myös kaks erillaista perinteistä tiimirakennetta liittyy tavallaan johtamistyyleihin, on pääohjelmoijajohtoinen tiimi (ohjaava) ja “egoton” tiimi (voimaannuttava).

Ketterä projektijohtaminen on muuttanut asetelmaa. Siinä kommunikaatio ja yhteistyö ovat tärkeitä sekä asiakas halutaan ottaa mukaan tiimiin. Projektipäällikön rooli on tässä pienentynyt.

Scrumissa on jaettu johtajuus. Scrum master on vain muodollinen jokapäiväinen projektipäällikkö (johtaa esim keskusteluita) ja product owner on projektipäällikkö joka päättää projektin jatkosta sekä siitä mitä ominaisuuksia ohjelmistoon halutaan, muuten hän pysyy lähes poissa.

Ketterien tiimien ohjelmoinnin ulkopuolisia rooleja: mentori, koordinoija, puolestapuhuja, markkinoija ja terminaattori.

Viikko 7, teht. 10

Jatkuva eksperimentointi ohjelmistokehityksen tukena -referaatti

Ohjelmistotuotannossa lean-ajattelu tarkoittaa, että ainoa tehtävä on tavoittaa ja palvella asiakkaita. Tähän liittyviä periaatteita on seitsemän: optimoi kokonaisuus, älä haaskaa, rakenna laadukkaasti, opi jatkuvasti, tuota nopeasti, osallista jokainen, kehity koko ajan.

Rakenna, mittaa, opi ovat peruspalikoita lean-ajattelun toteutuksessa.

Fagerholm et al. tarkoittavat jatkuvalla eksperimentoinnilla kenttäkokeisiin perustuvaa lähestymistapaa ohjelmistokehitykseen. Kokeita tehdään relevanttien sidosryhmien, yleensä asiakkaiden ja käyttäjien, ja mahdollisesti myös muiden sidosryhmien kuten sijoittajien, kolmannen osapuolen kehittäjien tai ohjelmistoympäristökumppaneiden kanssa.

Epätietoisuus voi erityisesti aiheuttaa ongelmia, esim epäselvyys ominaisuuksia sisällössä ja kehitys ilman vahvistusta. Tällöin tuote ja asiakkaan tarpeet eivät aina kohtaa. Tämän takia lean-ajattelu on hyväksi kun tehdään ominaisuuksia joiden hyödyllisyys perustuu konkreettiseen tietoon arvausten sijaan.