Tutkittua tietoa

T. Dybå, T. Dingsøyr: Empirical Studies of Agile Software Development : A Systematic Review. *Information and Software Technology* 50, 2008, 833-859.

J.E. Hannay, T. Dybå, E. Arisholm, D.I.K. Sjøberg: The Effectiveness of Pair Programming: A Meta-Analysis. *Information and Software Technology* 51, 2009, 1110-1122.

D.I.K. Sjøberg, A. Johnsen, J. Solberg: Quantifying the Effect of Using Kanban versus Scrum: A Case Study. *IEEE Software* 29, 5, 2012, 47-53.

M. Ikonen: Lean Thinking in Software Development: Impacts of Kanban on Projects. PhD thesis, A-2011-4, University of Helsinki, Department of Computer Science, 2011.

S. Mäkinen: Driving Software Quality and Structuring Work through Test-Driven Development. Pro gradu, HY/TKTL, 2012.

V. Kettunen et al.: A Study on Agility and Testing Processes in Software Organizations. In Proc. ISSTA'10. ACM, 2010.

Tutkittua tietoa 1

+ XP-projektit myöhästyvät hieman vähemmän kuin ”perinteiset” projektit
+ XP-projektit ylittävät budjettinsa selvästi vähemmän kuin ”perinteiset” projektit
+ pienten XP-tiimien tuottavuus on useimmiten parempaa kuin vastaavien ”perinteisten” projektiryhmien tuottavuus
+ läsnä oleva asiakas parantaa asiakasyhteistyötä
+ XP-projektien asiakkaat ovat tyytyväisempiä kuin ”perinteisten” projektien asiakkaat
+ XP-projekteissa ohjelmistokehittäjät ovat tyytyväisempiä kuin ”perinteisissä” projekteissa
+ XP toimii parhaiten, kun ohjelmistokehittäjät ovat kokeneita ja hallitsevat sekä toimialan että kehitystyökalut

Tutkittua tietoa 2

+ ketterästi kehitettyjen ohjelmistojen laatu on parempi kuin vastaavien ”perinteisesti” kehitettyjen ohjelmistojen laatu
+ 40 tunnin työviikko vähentää ylitöiden tekemistä

+ testivetoinen kehitys vähentää virheiden määrää
+ testivetoinen kehitys parantaa testauksen koodikattavuutta
+ testivetoinen kehitys pienentää koodin mutkikkuutta (complexity)
+ testivetoinen kehitys lisää ohjelmiston ylläpidettävyyttä
+ testivetoisesti kehitetyn ohjelmiston kokonaiskustannukset ovat pienemmät kuin ”perinteisesti” kehitetyn ohjelmiston, mikäli sen elinkaari on riittävän pitkä

Tutkittua tietoa 3

+ testauksen aloittaminen projektin aikaisessa vaiheessa helpottaa testausresurssien tarpeen arviointia
+ jatkuva ja aikaisin aloitettu testaus antaa enemmän aikaa testauksen tekemiseen
+ jatkuva ja aikaisin aloitettu testaus helpottaa testauksen fokusointia ohjelmiston tiettyihin osiin
+ myös testaus mukautuu helpommin muutoksiin ketterässä kehityksessä

Tutkittua tietoa 4

+ pariohjelmointi parantaa koodin laatua yleisesti ottaen hiukan, kokemattomilla ohjelmoijilla huomattavasti
+ pariohjelmointi nopeuttaa koodin tuottamista
+ pariohjelmointi on parhaimmillaan vaikeissa ja korkeaa laatua vaativissa tehtävissä, joilla ei ole tiukkaa aikataulua
+ ketterät projektit kehittävät opiskelijoiden kommunikointi-, sitoutumis- ja yhteistyötaitoja
+ ohjelmistokehitys on Kanbanilla nopeampaa kuin Scrumilla
+ tehtävät valmistuvat Kanbanilla nopeammin kuin Scrumilla
+ työn tuottavuus on Kanbanilla suurempi kuin Scrumilla
+ Kanban visualisoi hyvin työn etenemistä
+ Kanban tukee tilannekohtaista analyysia ja muutosta

Tutkittua tietoa 5

– työmäärän arviointi menee pahasti metsään XP-projektien alussa (mutta paranee projektien edetessä)
– XP-projekteissa ei kiinnitetä tarpeeksi huomiota ohjelmiston suunnitteluun eikä arkkitehtuuriin
– XP-projektien 40 tunnin työviikko uuvuttaa kehittäjät
– XP ei sovellu suuriin projekteihin
– Scrum-projekteissa pitää kouluttaa myös asiakasta
– Scrumin aikaikkunat ovat keinotekoisia, koska työmäärät yleensä aliarvioidaan ja vastaan tulee ennakoimattomia ongelmia
– Scrumin aikaikkunat johtavat liian nopeasti kehitettyyn ja huonolaatuiseen koodiin
– Scrumin pyrähdyksissä työ jakaantuu epätasaisesti: esimerkiksi testaajilla on pyrähdyksen alussa luppoaikaa ja lopussa liian kiire
– Scrumin järjestämisvaihe (staging) koetaan jätteeksi

Tutkittua tietoa 6

- ketterästi kehitettyjen ohjelmistojen käyttöliittymä on huonompi kuin ”perinteisesti” kehitettyjen ohjelmistojen käyttöliittymä
- ketterissä tiimeissä on vaikeampi vaihtaa ohjelmistokehittäjiä kuin ”perinteisissä” projektiryhmissä
- ketterien projektin onnistuminen ei riipu pelkästään teknisistä vaan myös sosiaalisista taidoista
- aina ketterästi läsnä oleva asiakas stressaantuu ja uupuu

Tutkittua tietoa 7

- testivetoinen kehitys lisää hieman koodin kytkentää (coupling)
- testivetoisesti kehitetyssä ohjelmistossa on lähes yhtä paljon yksikkötestauskoodia kuin varsinaista toiminnallista koodia
- useimmiten testivetoinen kehitys vie enemmän aikaa kuin ”perinteinen” kehitys, lisää kokonaistyömäärää ja kasvattaa kustannuksia
- testivetoinen kehitys vähentää työn tuottavuutta

- pariohjelmointi lisää kokonaistyömäärää
- pariohjelmointi vähentää kokeneiden ohjelmoijien tuottavuutta ja laskee heidän koodinsa laatua
- pariohjelmointi ei sovellu tehtäviin, joilla on tiukka aikataulu, vähäiset resurssit ja korkeat laatuvaatimukset

Tutkittua tietoa 8

- testaukseen erikoistuneille henkilöille on edelleen tarvetta (kehittäjien tekemä testaus ei aina riitä)
- kehityksen rinnalla tehty testaus voi olla vaikea suorittaa ja vaatii erityistä harkintaa ja ohjausta
- dokumentaation puute muodostuu ongelmaksi, jos testausta pitää ulkoistaa tai testaajilla ei ole riittävää kokemusta tuoteympäristöstä
- ketterässä kehityksessä testaajilta vaaditaan parempaa teknistä osaamista

Tutkittua tietoa 9

- testivetoinen kehitys on vaikeaa monille opiskelijoille

- virheitä korjataan Kanbanilla hitaammin kuin Scrumilla
- Kanban ei tarjoa apua ongelmien varsinaiseen ratkaisemiseen
- Kanban-taulu on liian rajoittunut työkalu suuriin projekteihin

± XP, Scrum, lean ja Kanban vaativat johtamista ja hyviä johtajia onnistuakseen; ilman niitä lopputuloksena on useimmiten kaaos ja sekuli