5. Laskarit

Viides sykli

Syklin teemat:

- *Riskienhallinta ketterissä prosesseissa*
- *Laatuyhteenveto*

Syklin tuotos: Valmiin ohjelmiston ja tehdyn dokumentaation luovutus

Riskienhallinta on oleellinen osa perinteisten projektien projektinhallintaa. Miten riskienhallinta sopii ketteriin prosesseihin?

Laatuyhteenvedossa ryhmä kertoo, miten laadunvarmistus toteutui projektissa ja miten se näkyy ohjelmiston laadussa.

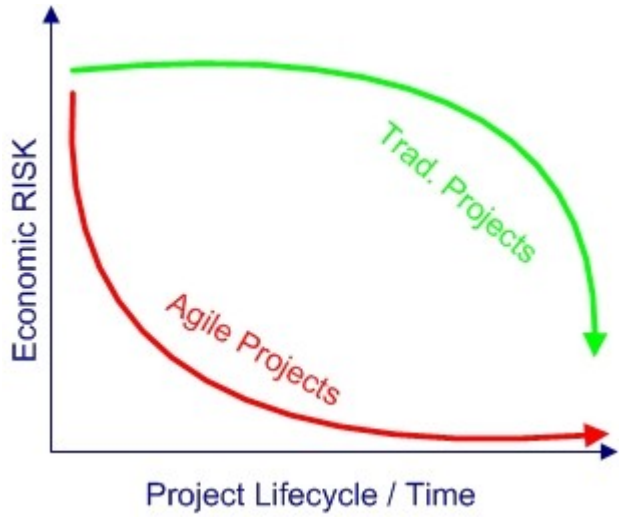
Viidennen syklin lopuksi ohjelmisto ja siitä tehty dokumentaatio luovutetaan harjoitusryhmän ohjaajalle. Työstä ei tarvitse tehdä käyttöohjetta.

Riskienhallinta ketterissä prosesseissa

Riskienhallinnan avulla pyritään vähentämään vahingollisten tapahtumien todennäköisyyttä ja vaikutusta projektiin. Iteratiivisen luonteensa takia riskienhallinta sisältyy osittain jo valmiiksi projektin elinkaareen.

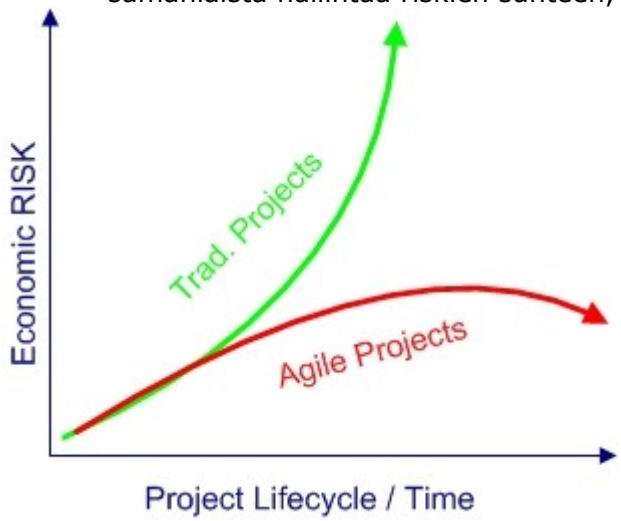
Ketterät menetelmät eivät yleensä tarjoa suoria ratkaisuja riskien hallintaan. Riskien hallinta otetaan huomioon jatkuvalla kommunikoinnilla sekä läheisellä asiakasyhteistyöllä, mutta suoranaisia käytänteitä riskien kartoittamiseen tai arviointiin ei yleensä ole tarjolla.

http://scrum.boulevart.be/?p=58



- Sekä ketterässä, että perinteisissä (esim. vesiputous) prosessimalleissa on projektin alussa korkea taloudellinen riski
- Ketterissä projekteissa asiakkaan kanssa ollaan tekemisissä jatkuvasti projektin aikana, mikä vähentää riskejä, koska asiakkaalta saadaan palautetta ja projektia voidaan korjata/säädellä sen mukaisesti.

- Perinteisissä projekteissa, joissa tuote toimitetaan projektin lopussa, ei ole samanlaista hallintaa riskien suhteen, koska asiakas ei osallistu kehitystyöhön.



- Perinteisissä projekteissa mitä pidemmälle projekti etenee sitä kalliimmaksi muutosten tekeminen tulee.
- Ketterissä prosesseissa muutoksia on helppo tehdä.

http://agile101.net/2009/07/28/12-principles-of-risk-management-pmbok-with-an-agile-slant/

Burn-down chartin avulla on kätevä seurata projektin etenemistä ja siitä näkyy hyvin, jos tahti hidastuu.

Vastuu riskien ehkäisemisestä ja lieventämisestä jakautuu koko ryhmälle, mikä on hyvä, koska kaikki vastuu ei jää projektipäällikölle. Tällöin mahdollisia riskejä havaitaankin helpommin.

Ketterissä menetelmissä on hyvä tehdä selväksi kaikille tiimin jäsenille heidän roolinsa riskienhallinnassa, ja pitää huoli että kaikki riskien hallinnan osat ovat jonkun vastuulla.

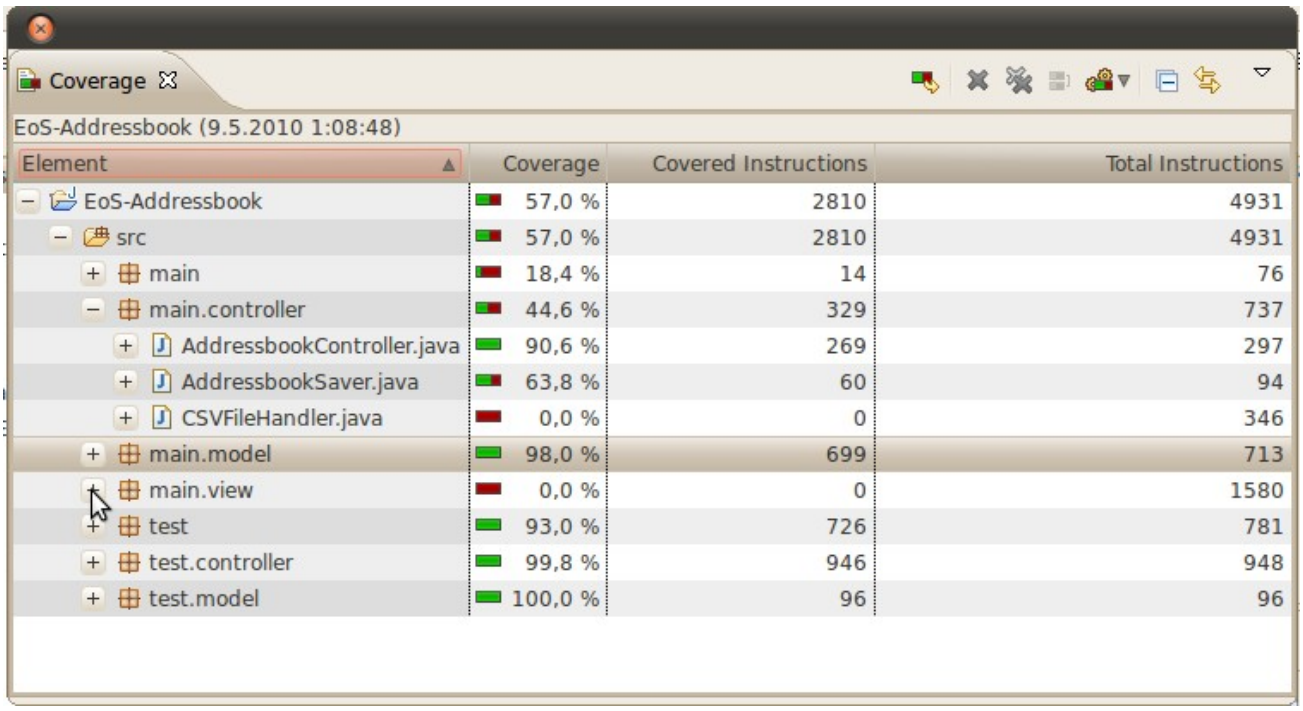
Risk Boardissa pidetään kirjaa mahdollisista riskeistä. Daily Scrumissa on hyvä käydä läpi potentiaaliset riskeihin liittyvät asiat ja päivittää Risk Boardia jos tarvetta on. Lisäksi tietenkin projektin ja sprinttien alussa tulee ottaa huomioon riskienhallinta.

Laatuyhteenveto

Kursivoidut kohdat on kopioitu laadunvarmistussuunnitelmasta.

1. Testausstrategia
Kaikki koodi yksikkötestataan paitsi käyttöliittymän osalta (85 % testikattavuus). Käyttöliittymän toiminta varmistetaan järjestelmätestauksessa.
Yksikkötestaus tehdään samaan aikaan koodauksen kanssa. Järjestelmätestaus tehdään jokaisen uuden toiminnallisuuden valmistuttua.
Yksikkötestien kattavuutta seurataan. (ECLEMMA(.org))

Testausstrategiaa on noudatettu hyvin. Testikattavuus on pysynyt projektin aikana 85% yläpuolella (lukuunottamatta käyttöliittymää). Päätimme kuitenkin olla yksikkötestaamatta CSVFileHandler -luokkaa, koska se vaatisi paljon työtä ja järjestelmätestauksella totesimme, että kyseinen luokka toimii odotetulla tavalla. Jos ko. luokkaa ei oteta huomioon, niin testikattavuus on yli 85%.



2. Aikataulu
Aikataulu määritetään jokaisen sprintin alussa. Sprintin lopussa määritetään tehty act-työmäärä ja tarkistetaan jäljellä oleva työmäärä.

Projekti on pysynyt hyvin aikataulussaan. Kaikki asiakkaan toivomat ominaisuudet ovat valmistuneet ajallaan. Työmääriä on seurattu säännöllisesti Google Docsissa.

3. Projektinhallinnan laadunvarmistus
3.1. Projektin seuranta ja ohjaus
Säännölliset suunnittelukokoukset viikon alussa. Viikon lopulla pidetään tarvittaessa lyhyet tarkistuskokoukset.

Kokouksia on pidetty säännöllisesti suunnitelman mukaan. Tämän takia yhteistyö tiimin jäsenten välillä toimi hyvin ja jos oli jotain epäselvyyksiä niin ne saatiin ratkaistua yhdessä.

3.2. Proseduurit ja työohjeet
Työohjeet tallennetaan google.docs-sivustolle.
Lähdekoodit dokumentoidaan käyttäen runsaasti koodin toimintaa kuvaavia kommentteja.

Lähdekoodin dokumentointia ei ole niin paljon kuin alunperin suunniteltiin. Päätimme, että kommentoimme vain epäselvät osat, koska koodi on suurelta osin luettavaa. Tunnukset on nimetty kuvaavasti ja ohjelman rakenne on yksinkertainen. Vertaisarvioinnin avulla valvottiin ohjelmakoodin luettavuutta.

3.3. Versionhallinta ja dokumenttien valvonta
Versionhallinta toteutetaan git-versionhallintasovelluksella. gitHub:ssa sijaitsee keskusrepository.
Dokumentit ylläpidetään GoogleDocs:n avulla. Sivuilla sijaitsevat mm. työohjeet ja backlogit.

Versionhallinnan käyttö Gitillä on sujunut hyvin. Tarvittavat ohjeet ovat olleet koko ajan saatavilla Google Docsissa.

3.2. Projektin laadunvarmistuksen metriikat
Backlogin avulla seurataan kunkin resurssin taskeja, työmääriä ja valmiusastetta.

Product- ja Sprint-backlogien avulla on jatkuvasti seurattu tehtävien etenemistä - onnistuneesti.

3.3. Katselmointi
Kaikille luokille tehdään vertaisarviointi, jossa kolleega käy läpi versionhallintaan laitettuja luokkia.

Kaikille versionhallintaan tehdyille lisäyksille/muutoksille on tehty samantien vertaisarvioinnit. Tämä on vaikuttanut positiivisesti koodin laatuun.

4. Standardointi
Sovelluskehityksessä käytetään junit-testimallia. Sovellus rakennetaa arkkitehtuurisesti käyttäen MVC-mallia.

JUnit on mahdollistanut testauksen sujumisen helposti ja nopeasti. MVC-mallin noudattaminen on tuonut hyvän arkkitehtuurin ohjelmalle.

Ohjelmistossa käytettävät laadun mittarit

Toiminnallisuus (Functionality)
Backlog ylläpitää toivottuja toiminnallisuuksia, joiden tulee olla kaikkien toteutettuna tai kuitattuna (esim. totetutetaan versiossa X.x).

Backlogien käyttö on taannut toiminnallisuuksien valmistumisen ajallaan.

Luotettavuus (Reliability)
Kattavat yksikkötestit (85 %, paitsi käyttöliittymäkomponenteille).

Yksikkötestit ovat lähes koko projektin aikana olleet tavoitetasolla. Se on tuonut luotettavuutta.

Käytettävyys (Usability)
Käytettävyyttä mitataan asiantuntija-arvioinnilla.

Käytettävyyttä mitattiin jatkuvasti uusien toiminnallisuuksien valmistuttua asiantuntija-arvionnein (tiimin jäsenten suorittamina). Ohjelman käytettävyys on kehitystiimin mielestä hyvä.

Tehokkuus (Efficiency)
Järjestelmätestauksen yhteydessä mitataan operatioiden vasteajat.

Vasteajat ovat olleet koko ajan hyvin pienet eikä niihin ole tarvinnut kiinnittää erityistä huomiota.

Ylläpidettävyys (Maintainability)
Java-tyylin mukaista koodia.
Yritetään välttää spagettikoodia ja purkkaratkaisuja.

Koodi on hyvän Java-tyylin mukaista ja luettavaa.

Siirrettävyys (Portability)
Ohjelmisto testataan useilla eri alustoilla jokaisen sprintin aikana. Alustoina ovat ainakin Linux, Windows ja MacOSX.

Ohjelma toimii hyvin Linuxissa Windowsissa ja MacOSX:ssä.

Prosessin laadun mittarit

Tuottavuus koodirivien määrä viikoittain.

Koodirivien määrä viikoittain jäi laskematta, mutta tuottavuus pysyi mielestämme hyvänä koko projektin ajan. Tämä näkyy siinä että kaikki toiminnallisuudet valmistuivat ajallaan. Tämän kokoisessa projektissa ei katsottu tarpeelliseksi laskea viikoittaista tuotettua rivien määrää. Teimme päätöksen, että tätä mittaria ei käytetä.

Asiakkaan raportoimien virheiden määrä per 1000 riviä kohden.

Asiakas ei raportoinut virheitä.

Työtehokkuuden osalta mitataan työaikaa suhteessa kokonaistyöaikaan.

Tätä mittaria ei käytetty, koska kokonaistyöaikaa ei kerätty harjoitustyön luonteen vuoksi. Työtehokkuuden kanssa ei kuitenkaan ilmennyt ongelmia.

Sprintissä sovittujen taskien toteutuminen aikataulussa.

Sprintissä sovitut taskit toteutuivat aikataulussa.