Miniprojekti loppuyhteenveto

Alkutoimet

Ryhmässämme oli neljä jäsentä: Henri Korpela, Kalle Ilves, Ossian Laurinharju ja Toni Rantanen. Yhteistyömme sujui hyvin koko projektin ajan ja ilmapiiri oli hyvä. Projekti toteutettiin *Scrum*-projektin hallinta menetelmää noudattaen. Käytimme ohjelmointi-kielinä käytimme Rubya ja JavaScriptiä. Suunnittelemamme toteutus asiakkaan toiveiden pohjalta oli web-sovellus, joten käytimme Rails-frameworkia. Projekti toteutettiin kolmessa viikon mittaisessa sprintissä, joiden aikana jokainen ryhmänjäsen teki noin neljä tuntia töitä. Aikarajoissa pysyttiin hyvin. Githubia käytettiin versionhallintaan. Githubissa olivat myös linkit itse ohjelmaan herokussa, sekä testien tarkastelun Travisissa. Projektissa tuli käyttää *continius integration*-periaatetta. Tämä toimi hyvin ja herokussa oli alusta asti uusin toimiva versio ohjelmasta. Käytetyistä ohjelman kehitystyökaluista ongelmia aiheutti lähinnä Github. Onnistuimme useaan otteeseen ”rikkomaan” versionhallinnan, mutta onneksi edellinen toimiva versio oli tallessa jonkun ryhmänhenkilön koneella, jolloin koodi pystyttiin helposti palauttamaan.

Ryhmäntoiminnasta

Organisoimme toimintamme niin, että *sprintin* alkaessa jokaiselle jaettiin tarkat tehtävät. Kaikki ryhmänjäsenet kävivät aktiivisesti yliopistolla, joten näimme lähes päivittäin. Kokoonnuimme aina vaihtelevalla kokoonpanolla päivän aikana ja keskustelimme projektiin liittyvistä ongelmista ja projektin edistymisestä. Näin saatiin nopeasti selville, jos jokin hidasti projektin etenemistä. Nämä tapaamiset olivat meidän versiomme *Scurmmin Daily scrum*-palavereista.

Scrummissa ryhmällä on *scrum master*, sekä *product owner* asemassa olevat henkilöt. Meidän ryhmässämme nämä asemat eivät olleet aina selkeät. Kaikki esittivät asiakastapaamisissa omia ehdotuksiaan asiakkaalle. Kalle Ilves toimi suurelta osin ryhmän johtajana ja piti huolta työnedistymisestä. Ongelman ilmetessä ryhmänjäsenistä ratkaisemaan rupesivat kuitenkin aina, sillä hetkellä paikalla olleet. Kaikki pyrkivät pitämään projektin logeja ajan tasalla. Jokainen merkitsi missä vaiheessa oma työ oli menossa ja teki arvion työhön käytettävästä tuntimäärästä.

Ryhmällämme oli perjantaina asiakastapaaminen, jonka jälkeen pidimme lyhyehkön (noin 15 minuuttia) retrospektiivin. Tämä palaveri oli ryhmämme kannalta erittäin hyödyllinen. Sen aikana kävimme läpi mitä parannettavaa edelliseltä *sprinti:tä* oli jäänyt. Lisäksi olimme tällöin aina saaneet asiakkaalta palautteen. Palaute itsessään auttoi määrittelemään oliko, jokin mennyt pieleen.

Ensimmäinen sprint

Ensimmäisessä *sprint:sä*: Laitoimme Rails serverin pystyyn Herokuun ja teimme sovelluksen pohjan. Pääsimme kaikkiin *sprint:lle* asetettuihin tavoitteisiin ja saimme myös hieman seuraavan *sprint:n* töitä aloitettua. Ongelmia oli aiheuttanut *sprintbaglog:n* kirjanpito. Olimme kirjanneet työtunnit *sprintbaglog:iin* väärällä tavalla. Palautteen saatuamme ja tiedostettuamme virheen saimme sen nopeasti korjattua. Testit olivat myös aiheuttaneet

ongelmia. JavaScriptin käyttö sovelluksessa sisällön luomiseen aiheutti vaikeuksia cabypara-testien toiminnalle. Tämä kuitenkin pystyttiin ratkaisemaan konfiguroimisella. Työn jaossa oli myös pieniä ongelmia ensimmäisen sprint:n aikana ja työtehtäviä jouduttiin hieman muuntelemaan *sprint:n* aikana.

Toinen sprint

Toinen *sprint* sujui hyvin. Edellisen *sprint:n* virheet oli korjattu ja *sprint:n* tavoitteisiin päästiin. Ryhmän työtehtävät oli jaettu selkeämmin kuin ensimmäisen *sprint:n* alussa ja kaikille oli selvää mitä kunkin kuului tehdä. Ohjelmaan oli kuitenkin jäänyt pieni virhe viittausten filteröintiin ja olimme ymmärtäneet hieman väärin asiakkaan edellisessä tapaamisessa asettamat toiveet. Esimerkiksi asiakkaan toivoma ominaisuus mahdollisuudesta kopioida valittujen viitteiden BibTex leikepöydälle ei toiminut halutulla tavalla. Jouduimme myös päivittämään ohjelmamme BibTex viitteiden tyyppejä, jotta saimme asiakkaan esimerkki viitteet toimimaan. Asiakas oli kuitenkin tyytyväinen työntuloksiin.

Toisen *sprint:n* aikana aloimme myös korjailla ohjelmaa siihen suuntaan, että useampikäyttäjä voisi käyttää ohjelmaa samanaikaisesti. Esimerkiksi Toisen käyttäjän yrittäessä muokata viitettä, jota toinen käyttäjä on juuri muokannut/poistanut aiheutti virheen. Tämä ja muutamia muita samankaltaisia virheitä korjattiin ja samalla alettiin kiinnittää asiaan huomiota.

Kolmas sprint

Kolmannessa *sprint:ssä* korjasimme toiminnallisuuden asiakkaan toiveiden mukaiseksi saamiemme toiveiden mukaisesti. Kolmannessa *sprint:ssä* törmäsimme ”mystiseen” ongelmaan, jossa moduuli (BibTex parsettaja) ei toiminut testiympäristössä, mutta toimi moitteettomasti kehitys- ja tuotantoympäristössä. Emme saaneet ongelmaa ratkaistua. Tämä johti siihen, että emme voineet kirjoittaa yksikkötestejä kyseiselle moduulille. Testasimme moduulin toimintaa kuitenkin cabypara testien avulla ja varmistimme näin myös testien avulla moduulin toiminnan. Ryhmän työskentely toimi myös hyvin virheen ilmetessä, sillä moduulia toteuttava ryhmänjäsen ilmoitti ongelmasta, jolloin ryhmä pystyi yhdessä miettimään ja etsimään ratkaisua ongelmaan. Päätös testien jättämisestä tehtiin lopulta yhdessä, eikä yhden henkilön mielipiteen perusteella. Lopullisesta ohjelmasta tuli mielestämme hyvä. Kaikkia alun perin määriteltyjä ominaisuuksia ei ohjelmaan saatu toteutettua, mutta annetun ajan puitteissa tulos oli hyvä. Kaikki toteutetut ratkaisut eivät ehkä ole täysin toimivia suurilla datamäärillä, mutta toiminnallisuus on hyvin jaettu komponentteihin ja näin optimointi ja mahdollinen kehitys olisi kuitenkin helposti mahdollista.

Ryhmäntoiminnan arviointia

Projektin aikana ryhmämme toiminta kehittyi kokoajan. Ryhmänjäsenten vastuualueet ja tehtävät selkeytyivät *sprint:en* aikana ja pyrimme säilyttämään samanlaiset tehtävät ryhmänjäsenillä *sprint:en* välillä. Huomasimme että pariohjelmointi toimi joissain tehtävissä erittäin hyvin ja pariohjelmointi olikin yleistä ryhmässämme. Myös ryhmämme kommunikaatio kehittyi. Emme valinneet mitään tiettyä kanavaa, jonka avulla kommunikoimme projektin aikana vaan kanavat muodostuivat itsestään projektin aikana (esim. tapaamispaikat yliopistolla ja *steam chat*). Pyrimme huolehtimaan, että kaikki tiesivät missä pitää olla ja milloin. Tämä onnistuikin hyvin ja kaikki olivat paikalla pakollisissa

tilaisuuksissa. Opimme paljon ryhmätyöskentelystä projektin aikana. Tärkeimpänä parannuksena olisi selvästi yhteisen kommunikointi kanavan päättäminen heti projektin alussa.

Työskentelytavat

Definition of done:iin kuului, että koodin tuli noudattaa *clean code* periaatteita. Tämä toteutui suurilta osin hyvin. Testit olivat kuitenkin etenkin toisen *sprint:n* jälkeen huonossa kunnossa ja sisälsivät paljon toistoa. Tämä kuitenkin korjattiin tekemällä moduuli, jonka avulla koodista saatiin karsittua turha *copy-paste*. Osaan testeistä jäi kuitenkin vielä parantamisenvaraa. Jälkeenpäin testit olisi voinut arkkitehturoida toisin. Löysimme hienon artikkelin, jonka ohjeita noudattaen cabypara testeistä olisi voinut saada mielestämme hyvin arkkitehturoituja (http://www.elabs.se/blog/51-simple-tricks-to-clean-up-your-capybara-tests). Itse ohjelman koodi oli siistiä ja noudatti hyvin *clean code* periaatteita ja oli järkevästi arkkitehturoitu. Esimerkiksi uusien viittaustyyppien ja attribuuttien lisääminen ohjelmaan on kooditasolla helppoa ja muutos täytyy toteuttaa vain yhteen paikkaan. Samoin uusien ominaisuuksien lisääminen oli vaivatonta projektin aikana. Vanhaa koodia ei juurikaan tarvinnut muuttaa.

Yhteenveto

Projektissa opimme paljon. Etenkin *Scrum:ista*. Mielestämme *Scrum* oli hyvin käyttökelpoinen ja toimiva tapa tehdä ohjelmia. Etenkin se että ryhmän ymmärtäessä asiakkaan toiveen hieman väärin saatiin palaute asiakkaalta nopeasti heti seuraavassa asiakastapaamisessa. Lisäksi *daily scrum* -kokous on varmasti arvokas ongelmien tunnistamisessa ryhmäntoiminnassa.