HAAGA-HELIA
ammattikorkeakoulu

Päiväkirjaopinnäytetyö mobiilisovelluksen testaajana ja laadunvarmistajana

Aleksi Monaco

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2015

HAAGA-HELIA
ammattikorkeakoulu

Tiivistelmä

20.11.2015

Tekijä(t)
Aleksi Monaco

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
Päiväkirjaopinnäytetyö mobiilisovelluksen testaajana ja laadunvarmistajana.

Sivu- ja liite-sivumäärä
59 + 0

Opinnäytetyön otsikko englanniksi
Thesis report journal as a mobile app tester and quality assurance engineer

Opinnäytetyö on kirjoitettu portfoliomaisessa päiväkirjamuodossa, jossa kuvataan opiskelijan arkea ohjelmistotestaajana ja laadunvarmistajana pienessä yrityksessä. Se koostuu kymmenestä seurantaviikosta, jossa merkintöjä kirjoitetaan maanantaisin, keskiviikkoisin ja perjantaisin työsuhteen osa-aikaisuuden vuoksi. Jokaisen seurantaviikon lopussa on analyysi, jossa käydään ohjelmistotestauksen osa-alueita läpi yrityksessä tehdyn työn näkökulmasta.

Opiskelija työskentelee pienessä yrityksessä, joka myy unentarkkailulaitteita, jotka toimivat älypuhelimen avulla. Koska yrityksellä ei ole ollut aikaisempaa testaajaa, opiskelija palkattiin tätä asemaa varten. Työtehtävät koostuvat pääasiassa yrityksen Android- ja iOS-mobiilisovellusten testaamisesta ja laadunvarmistuksesta. Viikoittaisten analyysien tavoitteena on parantaa yrityksen testausprosesseja ja toimintatapoja ammattikirjallisuuden ja muiden lähteiden avulla, mikä lopulta parantaa yrityksen tuotteen laatua. Opinnäytetyön toisena tavoitteena on myös kehittää opiskelijan omaa testausosaamista ja työtapoja.

Lopputuloksena työstä syntyi yritykselle testausprosessi ja se antoi opiskelijalle kokemusta ohjelmistotestauksesta, mobiiliratkaisun julkaisuprosessista ja erilaisista testaustyökaluista, mistä on hyötyä tulevaisuudessa. Tulos antoi myös hyvän pohjan testausprosessien jatkokehitystä varten.

Asiasanat
Testaus, laadunvarmistus, kehittäminen

HAAGA-HELIA
University of Applied Sciences

Abstract

20 November 2015

Author(s)
Aleksi Monaco

Degree programme
Business Information Technology

Report/thesis title	**Number of pages and appendix pages**
Thesis report journal as a mobile app tester and quality assurance engineer	59 + 0

This thesis report is written in a journal format. It describes the author's daily work as a software tester and quality assurance engineer at a small company. It consists of 10 weekly periods where daily entries are written every Monday, Wednesday and Friday, due to the part-time nature of the work contract. At the end of every week, there is an analysis, where different aspects of software testing are analysed from the perspective of the author's work.

The author works at a small company that sells sleep monitors that work with the user's smart phone. The company hasn't had testers previously, thus the student was hired for this position. The main responsibilities are testing and quality assurance of the company's Android and iOS mobile apps. The goal of the weekly analyses are to improve the company's testing processes and best practices through professional literature and other sources. In the end this will improve the overall quality of the company's product. The other goal is to develop the student's professional skills and practices in software testing.

As a result, the study has provided the company with a high quality testing process and has given the author experience in software testing, mobile solution releasing process and different testing tools, from which the author will benefit in the future. The result will also give a solid testing process knowledge base that which can be further developed.

Keywords
Software testing, quality assurance, development

Sisällys

1 Johdanto

Opinnäytetyö kirjoitetaan päiväkirjamuodossa 10 viikon ajanjaksolla syksyllä 2015. Päiväkirjamerkintöjen kirjoittaminen aloitetaan maanantaina 31.8.2015 ja viimeinen merkintä kirjoitetaan perjantaina 7.11.2015. Merkintöjä kirjoitetaan arkiviikon joka toisena päivänä johtuen osa-aikaisesta työsuhteesta ja koulun samanaikaisuudesta. Jokaisen viikon lopuksi kirjoitetaan viikoittainen analyysi kuluneesta viikosta.

Työtehtävät koostuvat pääasiassa yrityksen Android- ja iOS-mobiilisovellusten testaamisesta ja laadunvarmistuksesta. Työssä vaaditaan tarkkuutta, loogista päättelykykyä ja ohjelmointiosaamista sekä kärsivällisyyttä ja viestintätaitoja, sillä testaajan on pystyttävä viestittämään kaikki löydökset muille kehittäjille. Android- ja iOS-mobiiliohjelmointiosaamisesta on paljon hyötyä, mutta testaajalla on oltava hallussa erilaiset testaustyökalut, sillä niiden hyödyntäminen parantaa työtehokkuutta. Ohjelmointiosaamista vaaditaan ohjelmointivirheiden löytämiseen ja manuaalisten testien automatisointiin.

Minulle ei ole aikaisempaa ohjelmistotestauskokemusta, joten valitsin Lessons Learned in Software Testing: A Context-Driven Approach –kirjan lähdemateriaaliksi. Kirjaa on suositeltu aloitteleville testaajille, ja siinä esitetään yleisiä kompastuskiviä, joita testaajat kohtaavat. Kirjan avulla pyritään kehittymään sekä parantamaan työtehokkuutta. Toiseksi lähteeksi valitsin Beautiful Testing: Leading Professionals Reveal How They Improve Software –kirjan, joka ehdottaa erilaisia testausvaihtoehtoja ja korostaa asiakaslähtöistä testausta. Viimeiseksi ammattikirjallisuudeksi valitsin Software Testing –kirjan, joka sisällöltään kattaa ohjelmistotestauksen kaikki osa-alueet. (Sachin 2012.) Viikoittaisessa analyysissa pyritään analysoimaan viikon työtavat ja miten niitä voitaisiin parantaa.

Yrityksessä työskentelee noin 20 henkilöä. Päätoimisto sijaitsee Suomessa, mutta yrityksellä on myös pienempi toimisto Yhdysvalloissa. Suurin osa työntekijöistä työskentelee Suomessa. Yritys myy unen tarkkailulaitteita, joita laitetaan esimerkiksi petauspatjan alle. Laite kerää yön aikana sen käyttäjälle erilaisia tietoja unen laadusta. Tuote toimii iOS- ja Android-älypuhelinten kanssa. Sisäisesti yritys koostuu pääosin kahdesta tiimistä: tuote- ja markkinointitiimistä. Tuotetiimissä kehitetään tuotetta ja tehdään käyttäjätutkimuksia. Markkinointitiimissä markkinoidaan ja myydään tuotetta sekä hoidetaan asiakaspalvelua. Työskentelen Suomen toimistossa tuotetiimissä. Yrityksessä on töissä useampi ulkomaalainen, joten yrityksen virallinen työkieli on englanti. Työ vaatii osaltani paljon viestintää markkinointitiimin kanssa, esimerkiksi käyttäjäpalautteiden käsittelyssä.

1.1 Käsitteet

Crittercism – Työkalu, jolla seurataan mobiilisovellusten suorituskykyä ja kaatumisilmoituksia.

FogBugz – Työkalu, jolla merkitään ja hallinnoidaan sovelluksissa esiintyviä ohjelmointivirheitä.

Hyväksyntätestaus – Testausvaihe, jolla varmistetaan, että sovelluksen osa vastaa määritystä.

iTunes Connect – Applen palvelu, jolla voidaan julkaista mobiilisovelluksia.

Manuaalinen testaus – Testausprosessi, jossa testaaja käy läpi erilaisia testitapauksia liittyen sovellukseen.

Mixpanel – Työkalu, jolla seurataan käyttäjien vuorovaikutuksia web- and mobiilisovellusten kanssa.

Regressiotestaus – Testausvaihe, jossa varmistetaan, että uudet muutokset sovellukseen eivät ole rikkoneet muita osia.

Sentry – Työkalu, jolla voidaan seurata palvelimen lähettämiä lokimerkintöjä.

Xcode – Applen kehittämä kehitysympäristö, jossa kehitetään OS X - ja iOS-sovelluksia.

Yksikkötestaus – Testausmenetelmä, jolla testataan lähdekoodin yksittäisiä osia.

2 Lähtötilanteen kuvaus

2.1 Oman nykyisen työn analyysi

Työtehtäviini kuuluu pääasiassa yrityksen iOS- ja Android-sovelluksen testaaminen ja laadunvarmistaminen sekä yrityksen pilvipalvelimen ylläpitäminen. iOS- ja Android-sovellus käyttävät yhteistä pilvipalvelinta tietojen tallentamiseen ja jakamiseen. Lisäksi yrityksellä on sivutuotteena web-sovellus, jossa visualisoidaan samaa mobiilisovellusten tarjoamaa dataa laajemmin. Työhön kuuluu erilaisten yrityksen valitsemien työkalujen käyttöä. Laadunvarmistajana vastuuni on pitää huolta, että ennen jokaista uutta sovellusversiota, se on varmistettu toimivaksi laajalla hyväksyntätestauksella.

Työskentelen pääasiassa yhdessä tuotteen kehittäjien kanssa ja aina kun uusia ominaisuuksia tai korjauksia on toteutettu, vastaan sen lopullisesta testaamisesta. Kehittäjä on tietenkin vastuussa oman toteutuksensa testaamisesta, mutta varmistan lopullisella testauksella sen toimivaksi tai virheen korjatuksi. Lopullinen testaus koostuu yleensä manuaalisesta testauksesta, joka tehdään oikeilla mobiililaitteilla. Kehityksessä käytetään versionhallintaa ja kun versionhallinnan päähaaraan on lisätty uusia ominaisuuksia tai korjauksia, luon uuden sovellusversion. Yleensä luonnin yhteydessä ajan vielä kaikki yksikkötestit, ja sen jälkeen varmistan yleisellä tasolla sovelluksen toimivuuden. Kun uusi sovellusversio toimii halutulla tavalla, julkaisen sen yrityksen sisäiseen testaukseen. Sisäiseen testaukseen kuuluu kaikki yrityksen työntekijät, ja sillä pyritään estämään kaikki suurimmat ohjelmointivirheet – mikäli niitä esiintyy. Jos sovellusversiossa on jotain pielessä, saan ilmoituksen ja varmistan, että minä tai joku muu kehittäjistä korjaa asian. Toivon mukaan sisäisessä testauksessa ei esiinny vakavia ongelmia, jolloin sovellusversio julkaistaan ulkoiseen testaukseen, johon kuuluu yrityksen valitsemia ulkopuolisia aktiivisia käyttäjiä. Testauksessa käyttäjät ilmoittavat kokemuksiansa joko virallisessa Facebook-ryhmässä tai sähköpostilla. Prosessin jälkeen uusi sovellusversio on valmis julkaistavaksi kaikkien käyttäjien saataville.

Vaikka testattavaa on paljon uutta sovellusversiota julkaistaessa, on aikaisempaa versiota ylläpidettävä ja ohjelmointivirheitä kirjattava. Mobiilisovellusten suorituskyvyn ja kaatumisilmoitusten seuraamiseen käytän Crittercism-ohjelmaa. Ohjelman avulla pystyn selvittämään mitkä ovat yleisimmät ohjelmointivirheet ja virhetilanteet. Mikäli joitakin virheitä esiintyy useasti, teen virhetilanteesta raportin ja keskustelen asiasta kehittäjien kanssa.

Ohjelmointivirheiden raportoimiseen käytän FogBugz-nimistä ohjelmaa, jolla luodaan ja priorisoidaan erilaisia ohjelmointivirheisiin ja käytettävyysongelmiin liittyviä raportteja.

Kaikki isoimmat ja kiireellisimmät virheet korjataan ensimmäiseksi. Mobiilisovellukset käyttävät pilvipalvelinta tietojen tallettamiseen ja siihen tehdään usein muutoksia tarpeiden mukaan, siksi palvelimen tilaa täytyy myös seurata. Seuraan palvelimen tilaa Sentry-ohjelmalla, joka ilmoittaa kaikista palvelimen uusista lokitiedoista sähköpostilla.

Olen huomannut, että työtehtävässä tarvitaan tarkkaavaisuutta. Varsinkin testaamisen aikana on tärkeää, että käyn läpi mahdollisimman monta käyttötilannetta ja kirjaan tarkasti löydökset. Ilman kunnollista dokumentaatiota on vaikea priorisoida pahimmat korjattavat virheet. Asioita voi jäädä helposti huomaamatta ja siksi testaamisessa tulee olla kärsivällinen. Ennen testaajan roolia, työskentelin yrityksessä harjoittelevana ohjelmistokehittäjänä ja olen huomannut, että ohjelmointiosaamisesta on hyötyä virheiden löytämisessä. Voidakseen työskennellä kehittäjien kanssa, tulee ymmärtää erilaisia termejä, joita käytetään ohjelmointimaailmassa.

Työkalujen tehokas käyttö on tärkeää, sillä niiden avulla testausprosessi sujuu tehokkaammin. Selviytyäkseni työtehtävässä minun tulee kehittyä testaajana. Kokemusta kertyessä testaaminen helpottuu: on helpompi löytää pienemmätkin virheet ja niiden syyt. Testausprosessin parantamiseksi on tärkeää keskustella muiden työntekijöiden kanssa, pitää aktiivista viestintäkanavaa asiakastuen kanssa ja ymmärtää käyttäjän tarpeita.

Yrityksellä ei ole aikaisemmin ollut toimivaa testausprosessia ja minut palkattiin syksyksi juuri sen kehittämistä varten. Minulla ei ollut vankkaa työkokemusta testauksesta, mutta kovalla työnteolla uskon kehittäväni yritykselle testausprosessin. Yhdessä esimieheni ja muiden kehittäjien kanssa kehitämme prosessia, mutta lopulta minä tehtäväni on muodostaa kokonaisuus. Työ vaatii kuitenkin jonkin asteista ohjeistusta kokeneemmalta kehittäjältä.

Jotta voin kehittyä, aion lukea ohjelmistotestaukseen liittyvää kirjallisuutta. Ohjelmistokehittäjät eivät varsinaisesti ole testaajia, mutta heillä on tarpeeksi kokemusta testauksesta ja he voivat ohjata minua. Aikaisemman työsuhteen perusteella minulla on tuotteesta ja sen takaa toimivasta lähdekoodista paljon kokemusta, joka helpottaa testaamista. Uskon, että syksyn loppupuolella olen itsevarmempi ja kokeneempi testaaja.

Olen edelleen opiskelija ja varsinainen työhistoria IT-alalla alkoi vasta kouluni työharjoittelun merkeissä. Roolini yrityksessä on muuttunut, mutta työympäristö on pysynyt lähes samanlaisena. Työtehtävät ovat lähinnä muuttuneet. Voisi siis sanoa, että olen ammatillisesti vielä erittäin varhaisessa vaiheessa ja kehitettävää on. Minulle on tärkeää, että ymmärrän mahdollisimman suuren osan yrityksen toiminnasta ja löydän oman roolini IT-

alalla. Testaajana olen vastuussa siitä, että käyttäjät ovat tyytyväisiä tuotteeseen ja että he jatkavat sen käyttöä.

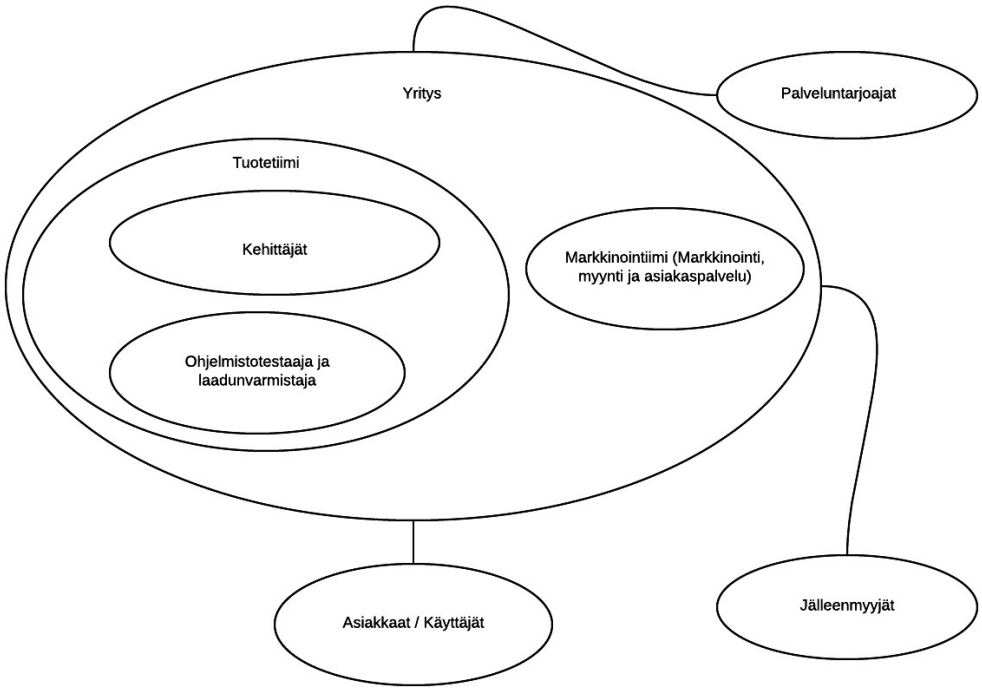
Yleensä kysyn paljon neuvoja ja ohjeita, varsinkin kun aikaisempaa laadunvarmistus- tai testausprosessia ei ole. Aiemmin ohjelmistokehittäjänä työtehtävät olivat yksinkertaisempia ja suoraviivaisempia ja työnkuvaani kuului enemmän työtehtäviä. Nyt työtehtävät ovat vaativampia ja vastuuta on enemmän.

Ennen tavoitteenani oli kehittyä ohjelmoijana ja varmistaa koodini laatu. Edelleen haluan jatkoni kannalta kehittyä ohjelmoijana, mutta haluan erityisesti kehittyä testaajana. Testausprosessi on monivaiheinen ja useilla ohjelmistoyrityksillä on erikseen palkatut testaajat. Minulla on paljon opittavaa testausmenetelmistä ja tavoista, mutta tästä lähtökohdasta on hyvä kehittyä. Tavoitteena on kehittää prosessi, joka luo paremman tuotteen yritykselle.

2.2 Sidosryhmät työpaikalla

Yrityksen päätoimisto koostuu kahdesta tiimistä, tuote- ja markkinointitiimistä, kumpaakin johtaa nimetty henkilö. Tuotetiimi koostuu lähinnä minusta ja muista kehittäjistä. Tiimin johtava henkilö on myös esimieheni. Markkinointitiimi koostuu markkinoinnista, myynnistä ja asiakaspalvelusta. Asiakaspalvelun tehtävänä on vastata asiakaspalautteeseen ja välittää palautteita tuotetiimille tarpeen mukaan. Yrityksellä on toimisto myös Yhdysvalloissa, jossa sijaitsee yrityksen toimitusjohtaja ja Yhdysvallan markkinoista vastaava myyntipäällikkö.

Tuotetta myydään suoraan yrityksen verkkokaupasta, mutta tuotteitta on tarjolla myös jälleenmyyjiltä. Asiakkaat ovat lähinnä yksityiskuluttajia.



Kuva 1. Yrityksen sidosryhmät ohjelmistotestaajan näkökulmasta

Sisäisiin sidosryhmiin kuuluvat muut kehittäjät, joiden kanssa teen työtä. Kehittäjien lisäksi teen yhteistyötä asiakaspalvelun kanssa, joka kuuluu markkinointitiimiin. Ulkoiset sidosryhmät ovat tuotetta käyttävät asiakkaat, testaustyökaluja tarjoavat yritykset ja jälleenmyyjät.

2.3 Vuorovaikutustaidot työpaikalla

Eniten yhteistyötä teen yrityksen sisäisten sidosryhmien kanssa. Päivittäin olen samassa tilassa kuin koko tuotetiimi. Markkinointitiimi on viereisessä huoneessa ja siten helposti lähestyttävissä. Yrityksen periaatteena on, ettei missään ole ovea kiinni. Se mahdollistaa helpon kommunikoinnin. Kommunikointiin yrityksessä käytetään myös Flowdock-nimistä pikaviestintäohjelmaa, mutta koska suurin osa työntekijöistä on samassa toimistossa, helpoin tapa kysyä apua ja keskustella on kasvokkain.

Yrityksen virallinen työkieli on englanti, mutta joskus puhutaan suomea edellyttäen, että keskustelupiirissä kaikki osaavat sitä. Kaikki lähdekoodi ja dokumentit kirjoitetaan siten luonnollisesti englanniksi. Asiakaspalvelu palvelee myös englannin kielellä.

Tuotetiimissä käydään läpi ne uudet ominaisuudet, jotka minun tulisi testata ja minä puolestani kerron mitä tulisi korjata. Virhetilanteet tai huonot käyttökokemukset tulevat joko

minulta tai käyttäjiltä asiakaspalvelun kautta. Käyttäjät voivat olla joko yrityksen sisäisiä tai ulkoisia.

3 Päiväkirjaraportointi

Päiväkirjamerkinnät tehdään viikon jokaisena maanantaina, keskiviikkona ja perjantaina. Niiden tarkoitus on kuvata päivän tavoitteet ja samalla arvioida omaa kehitystä.

3.1 Seurantaviikko 36

Maanantai 31.8.2015

Maanantai aloitettiin yrityksen yhteisellä stand up –kokouksella. Jokainen työntekijä sai kertoa kuluneiden viikkojen työtehtävistä ja saavutuksista sekä samalla kertoa kyseisen viikon tulevat työtehtävät. Kokouksen ideana on pitää kaikki työntekijät ajan tasalla muiden tekemisistä. Yrityksen suhteellisen pienen koon vuoksi se pysyy lyhyenä ja mielekkäänä. Luonnollisesti isommassa yrityksessä tämän tyyppinen kokous ei olisi mahdollista. Itse kerroin uudesta työsuhteestani ohjelmistotestaajana, sillä virallisesti aloitin roolissa elokuun loppupuolella. Kerroin myös tarkemmin työtehtävistäni tulevalla viikolla. Tehtäviin kuului lähinnä virheiden korjaamista ja ohjelmisto-osien testausta.

Yhteisen kokouksen jälkeen pidimme tuotetiimin kesken suunnittelukokouksen viikon työtehtävistä. Samalla pidimme retrospektiivin tyylisen kokouksen, jossa pohdittiin parin edellisen viikon saavutuksia ja työtapoja. Suunnittelukokouksessa tiimi määräsi jokaiselle työntekijälle oman osa-alueen tuotteen kehitysprosessin kehitettäväksi edelleen. Osa-alueeni liittyi testaukseen – tarkemmin regressiotestaukseen. Viikon tavoitteeksi sovimme osaltani, että loppuviikkoon mennessä esitän ehdotukseni siitä, kuinka parantaisimme regressiotestausta uutta sovellusversiota luodessa.

Viime viikolla olin julkaissut sisäiseen testaukseen uuden iOS-sovellusversion, joka sisälsi lähinnä pieniä korjauksia ja sovelluksen japaninkielisen käännöksen. Aamupäivällä tarkistin Crittercism-ohjelmalla kyseisen version tilanteen. Huomasin, että yhtä virhettä esiintyi selkeästi muita virheitä useammin. Päätin kirjoittaa tilanteesta raportin, mutta vasta myöhemmin, sillä minulla oli viime viikon lopulla jäänyt ison virheen korjaaminen kesken.

Loppupäivän käytin virheenkorjaukseen, joka liittyi mobiilisovelluksen tietojen jakamiseen pilvipalvelimelle. Ongelma oli lähinnä, että osa tiedoista ei tallentunut palvelimelle oikein ja se ilmeni, kun sovellukseen tehtiin muutoksia hiljattain. Voisi sanoa, että tehtävä oli reg-

ressiotestausta, vaikka uudet muutokset oli jo julkaistu käyttäjille. Testausta tietojen jakamiseen olisi pitänyt tehdä tarkemmin, jotta ongelma olisi huomattu ajoissa. Tehtävä vaati iOS-sovelluksen lähdekoodin tutkimista. Päivän päätteeksi löysin mahdollisen virheen aiheuttavan koodin ja kirjoitin siitä sähköpostin muotoisen raportin kahdelle muulle kehittäjälle, jotta he voivat seuraavana päivänä keksiä oikean ratkaisun. Samalla kun tutkin virhettä, huomasin että osa iOS-sovelluksen yksikkötesteistä eivät läpäisseet iOS 7 –käyttöjärjestelmällä, joten korjasin testit. Olen tyytyväinen päivän saavutuksiin, mutta olisi tärkeää muistaa tehdä yhtä asiaa kerrallaan. Työtehtävät pysyvät paremmin hallinnassa, jos niitä tekee systemaattisesti.

Keskiviikko 2.9.2015

Heti aamulla pidimme yrityksen yhteisen bi-weekly-kokouksen. Se pidetään kahden viikon välein ja siinä käymme tarkemmin läpi kunkin tiimin suunnitelmia pidemmällä aikavälillä. Ero maanantain pidettävään kokoukseen on, että maanantaisin jokainen työntekijä kertoo itsenäisesti omista työtehtävistään, kun taas bi-weekly-kokouksessa tiimien johtajat kertovat oman tiimin saavutuksista ja tulevaisuuden suunnitelmista. Tuotetiimin suunnitelmiin kuului nykyisen sovelluksen käytettävyyden parantaminen, joka tarkoittaa osaltani virheiden korjaamista ja käyttäjäpalautteiden tutkimista.

Kokouksen jälkeen tein pienempiä tehtäviä: lisäsin uuden käyttäjän iOS-sovelluksen ulkoiseen testaukseen ja korjasin sovelluksen sisäisen palautteen lähettämisen. Oli käynyt ilmi, että integraatio asiakastukijärjestelmän ja mobiilisovelluksen kanssa ei toiminut. Koska Sentry-ohjelma lähettää ilmoituksen sähköpostiini uusista virheistä pilvipalvelimella, huomasin virheen lähes saman tien ja sen avulla pystyin myös korjaamaan virheen mahdollisimman nopeasti. Integraatio on elintärkeä asiakaspalautteiden keräämiseen, sillä palautteet lisätään suoraan asiakaspalautejärjestelmään, johon asiakastuki pystyyn vastaamaan suoraan.

Maanantaina korjasin yksikkötestejä iOS-sovelluksessa, joten pyysin muita kehittäjiä katsomaan tekemiäni muutoksia. Yrityksessä on tuotekehityspuolella tapana katselmoida kaikki kehittäjien tekemät koodimuutokset siten, että toinen kehittäjä katselmoi ja tarvittaessa kommentoi ja ehdottaa korjauksia koodiin. Minusta tämä on yksi hyvä tapa parantaa koodin laatua ja olen ollut tyytyväinen siihen. Muiden kiireiden vuoksi kukaan ei vielä ehtinyt katselmoimaan koodini, joten viimeistään perjantaina lisään muutokseni iOS-sovellukseen.

Maanantaina tein raportin löytämästäni virheestä iOS-sovelluksessa. Se liittyi tietojen jakamiseen pilvipalvelimelle. Kävin tänään siitä keskustelua muiden kehittäjien kanssa ja mietimme kuinka ja kuka virheen korjaa. Päätimme, että koska olin jo tutustunut ongelmaan tarkemmin, korjaisin ohjelmointivirheen. Päivän loppupuolella aloitin virheen korjauksen ja löysin siihen mahdollisen ratkaisun. Perjantaina aion todennäköisesti pyytää muutoksia arviotavaksi ja käydä keskustelua ratkaisusta. Emme pitäneet tänään tuotetiimin päivittäistä stand up-kokousta, joka kyllä olisi mielestäni ollut aiheellista pitää. Pidämme sen maanantaisin viikon suunnittelukokouksen yhteydessä, mutta muina päivinä ne pidetään aamulla, kun kaikki ovat saapuneet toimistolle. On tärkeää pitää kaikki ajan tasalla, mutta välillä kiireiden ja muiden kokousten vuoksi stand up-kokousta ei välttämättä ehditä pitämään.

Perjantai 4.9.2015

Päivän tavoitteena oli julkaista uusi sovellusversio iOS:lle. Siihen kuuluisi löytämäni virheen korjaus sekä muita pieniä ominaisuuksia. Aamulla korjasin virheiden jakamiseen liittyvän ongelman, jonka jälkeen toinen kehittäjä katselmoi sen. Koska kävimme jo aikaisemmin keskustelua ratkaisusta, koodiin ei tarvinnut enää tehdä muutoksia. Vaikka olen työtehtävältäni ohjelmistotestaaja, myös minua pyydetään tekemään ohjelmointitehtäviä johtuen pienestä henkilöstön määrästä.

Pidimme aamulla päivittäisen stand up-kokouksen, jossa omalta osaltani kerroin, että tavoitteena oli julkaista uusi sovellusversio. Vaikka en ole päivittäin toimistolla, muut tuotetiimin jäsenet pystyvät pitämään kokouksen jokaisena työpäivänä. Kokouksen jälkeen lisäsin tekemäni muutokset versiohallinnan päähaaran. Koska muutos liittyi ohjelmointivirheeseen, merkitsen FogBugz-ohjelmaan virheen korjatuksi selityksen kanssa. Jokainen korjattu virhe tulisi merkitä ohjelmaan, sillä näin virhelista pysyy ajan tasalla ja kaikki tietävät mitä muita virheitä tulisi korjata.

Maanantaina sovittiin, että perjantaihin mennessä selvittäisin kuinka voisin parantaa ohjelmiston regressiotestausta. Koska yhdellä tuotetiimin jäsenellä on aikaisempaa kokemusta regressiotestauksesta edellisessä työpaikassa, kävin hänen kanssaan keskustelua ja yhdessä mietimme parannusehdotusta. Päädyimme lopputulokseen, että jonkinlainen testaushallintajärjestelmä olisi hyvä olla testitapauksien hallintaa varten. Ohjelmaan tulisi pystyä lisäämään testitapauksia ja käymään niitä läpi sekä merkitä testien tuloksia ylös. Vaihtoehtoja on varmasti monia, mutta kollega ehdotti TestLodge-ohjelmaa, joka olisi mahdollista integroida FogBugz-ohjelman kanssa. Mietimme kollegan kanssa testauspro-

sessia ja päädyimme tulokseen, jossa jokaisen kehittäjän vastuulla on testata oma toteutus. Uuden julkaisun yhteydessä tehtäväni olisi tehdä testausta suuremmalla laajuudella. Ennen uuden version julkaisua varmistaisin sovelluksen toimivuuden hyväksyntätestauksella, johon kuuluu regressiotestausta ja yleiseen toimivuuden testaamista. Yleisessä testauksessa käytäisiin sovelluksen kaikki tärkeimmät ominaisuudet läpi yleisellä tasolla. Osiin, joihin on tehty muutoksia, tehtäisiin regressiotestausta. Regressiotestauksessa käytäisiin testitapauksia läpi ominaisuuteen liittyen. Maanantaina esittelen ehdotukseni ja keskustellemme siitä tuotetiimin kanssa.

Päivän päätteeksi loin uuden sovellusversion ja kokeilin hyväksyntätestaus-mallia heti käytännössä. Ennen julkaisemista sisäiseen testaukseen varmistin sovelluksen toimivuuden yleisellä tasolla ja tein regressiotestausta. Muutoksia oli erittäin vähän, joten regressiotestausta ei tarvinnut paljon tehdä. Esimerkiksi kehittäjä oli toteuttanut mobiilisovelluksen herätyskello-ominaisuuteen äänenvoimakkuuden osoittimen käyttöliittymään. Koska ominaisuus liittyi herätyskelloon, kävin testitapaukset siihen liittyen. Tällä hetkellä testitapaukset on luotu Excel-muodossa, mutta tulevaisuudessa ne voitaisiin käydä järjestelmän avulla läpi. Lopuksi julkaisin uuden version sisäiseen testaukseen.

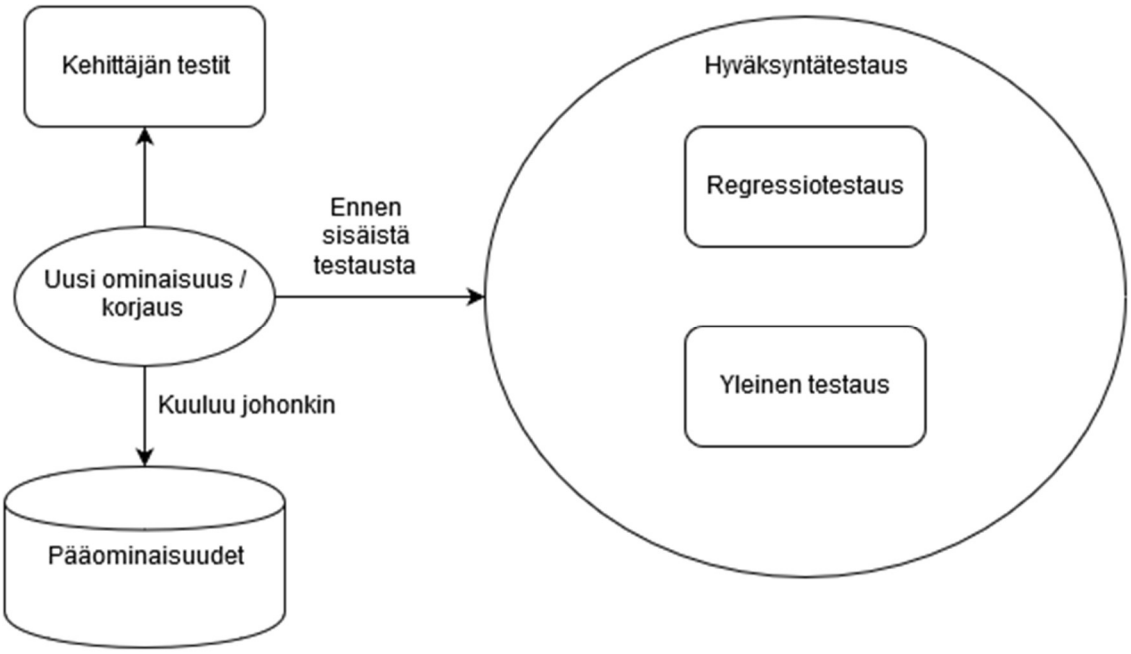
Viikkoanalyysi

Tämä oli toinen viikkoni yrityksen virallisena ohjelmistotestaajana. Tavoitteena tälle viikolle oli harjoitella ja kehittää päätuotteen eli mobiilisovelluksen testausprosessia. Viikossa se ei synny, mutta viikon lopussa näkemys laadukkaasta prosessista oli selkeämpi. Viikon pääaiheina oli ohjelmointivirheraporttien hallinta ja sovelluksen hyväksyntätestaus, joita tullaan työstämään useampi viikko. Aiheet ovat sovelluksen laadun kannalta ensiarvoisia ja tehtäväni on pitää huolta siitä, että niissä edistytään.

Virheraporttien hallinta on erityisen tärkeää, sillä raportit ovat testaajien ensisijaisia tuotoksia (Pettichord, Bach & Kaner 2002, 65). Hyvästä raportista hyötyvät kaikki, sillä se antaa pohjan virheen korjaamiselle. Kaikki tiedostetut virheet ovat raportoitu FogBugz-ohjelmalla, siten että jokaisella tuotteen osa-alueella on oma listansa. Esimerkiksi Android-, pilvipalvelin- ja iOS-virheet ovat kukin omilla listaloillaan. Haasteena ei ole mielestäni pitää kirjaa tiedetyistä virhetilanteista, vaan muistaa pitää ne ajan tasalla ja priorisoida listat siten, että kriittisimmät virheet korjataan ensin. Ohjelman avulla virheiden tilaa voi muuttaa – onko virhe korjattu vai onko korjaus jo julkaistu. Testaajan vastuulla ei ole raportoida kaikki mahdolliset virhetilanteet, vaan jokaisella yrityksessä työskentelevällä tulisi olla mahdollisuus lisätä raportti virheestä (Pettichord 2002, 68). Kaikki eivät esimerkiksi näe

sovelluksen käyttöliittymäelementtejä samalla tavalla – jonkun mielestä painikkeen tulisi olla sivun vasemmassa reunassa, ei oikeassa reunassa.

Viikon toisena pääaiheena oli sovelluksen regressiotestaus, josta minun on tarkoitus esitellä parannusehdotus maanantaina. Regressio tarkoittaa uusien muutoksien yhteydessä sovelluksen muuttumattomien osien testausta (Software Testing Help 2015). Kävin prosessin parantamisesta keskustelun toisen työntekijän kanssa.



Kuva 2. Alustava laadunvarmistusprosessi yrityksen mobiilisovellukselle

Kun kehittäjä on toteuttanut uuden ominaisuuden tai korjannut virheen, hänen tulisi testata muutoksensa. Tarkoituksena ei ole testata koko sovellus, vaan oman koodin testaus esimerkiksi yksikkötesteillä ja manuaalisella testauksella. Kehittäjän on pystyttävä toteamaan, että muutos tai ominaisuus toimii halutulla tavalla. (Balasubramanyam.) Muutos tai ominaisuus on myös kuuluttava johonkin koko sovelluksen pääominaisuuteen. Sillä tarkoitetaan korkeamman taso kokonaisuutta, joka voisi olla yrityksen tuotteessa esimerkiksi herätyskello tai käyttäjän tietojen synkronointi pilveen. Mikäli ominaisuus on itsessään kokonaisuus, joka ei varsinaisesti kuuluun mihinkään muuhun osaan, siitä tehdään uusi pääominaisuus. Muutoksien liittäminen johonkin pääominaisuuteen auttaa testaajaa hyväksyntätestauksessa, sillä yleensä kaikkea ei ole mahdollista testata joka kerta, ja regressiotestaaminen on helpompi kohdistaa muutamaan osaan.

Koska kehittäjä on tehnyt oman osuuden testauksesta ja merkinnyt mihin pääominaisuuteen muutos on liitetty, voin tehdä muutamaan osaan tarkempaa regressiotestausta ja

käydä muut osat läpi yleisellä tasolla. Tarkemmassa regressiotestauksessa käyn läpi kaikki pääominaisuuden käyttötapaukset ja merkitsen tulokset ylös. Koska sovellukseen tehdään jatkuvasti korjauksia tai parannuksia tämän tyylinen testaaminen on tärkeää (Software Testing Help 2015). Mikäli ongelmia esiintyy tai testit eivät mene läpi, sovellusta ei julkaista sisäiseen testaukseen. Koska tulevaisuudessa testit ovat yksitoikkoisia, yksi vaihtoehto olisi automatisoida kaikki mahdolliset testitapaukset. Tämä on aihe, jonka aion nostaa esille ja josta on jonkin verran käyty keskustelua. Se vaatii kuitenkin hyviä ja selkeitä testitapauksia, muuten automatisointi ei auta ja siitä tulee sekavaa (Pettichord 2002, 98–99).

Viikon tavoitteet sain ainakin osittain saavutettua, mutta laadunvarmistuksen kehittäminen on jatkuva vaihe, jonka lopputulos voidaan saavuttaa tulevaisuudessa. Vaikka regressiontestauksen parantaminen on vastuullani, on hyvä, että pystyn jakamaan ajatuksia ja keskustelemaan aiheesta muiden kanssa. Päätökset tehdään kuitenkin yhdessä.

3.2 Seurantaviikko 37

Maanantai 7.9.2015

Perinteisesti aloitin maanantain osallistumalla yrityksen yhteiseen stand up –kokoukseen. Kerroin tavoitteestani tälle viikolle ja raportoin viime viikon saavutuksista. Perjantaina julkaisin uuden iOS-sovellusversion yrityksen sisäiseen testaukseen, joten työntekijöillä oli ollut viikonloppu aikaa käyttää sitä. Tämän viikon tavoitteena on jatkaa uuden versiokandidaatin julkaisuprosessia, sillä versio täytyy saada App Store –sovelluskauppaan käyttäjän saataville. Myöhemmin oli vielä luvassa tuotetiimin keskeinen suunnittelukokous, jonka jälkeen tavoitteena oli saada versioehdokas sisäisestä ulkoiseen testaukseen.

Aamukokouksen jälkeen ehdin vielä ennen suunnittelukokousta keräämään muutamalta työntekijältä palautetta sovelluksen viikonloppuisesta käytöstä. On hyvä liittää laadunvarmistusprosessiin vaihe, jossa kysytään jo testauksen sisäisessä vaiheessa palautetta suullisesti. Haastattelin kahta työntekijää, josta yhdellä ei ollut mitään poikkeavaa mainittavaa, mutta toisella oli palautetta uudesta muutoksesta käyttöliittymään. Palaute oli pikemminkin idea siitä, miten muutoksen olisi voinut tehdä toisella tavalla. Kirjasin palautteen, jotta se otetaan myöhemmin huomioon.

Päivällä pidimme suunnittelukokouksen, jossa mietimme ja priorisoimme työtehtäviä tiimin kesken. Koska tehtäväni liittyvät tuotteen testaukseen ja laadunvarmistukseen, minulle ei varsinaisesti annettu tehtäviä kuten muille, koska ne ovat selkeät jo etukäteen. Yrityksellä

on menossa projekti, jossa iOS-sovellukselle kehitetään mahdollisuus käyttää sitä Apple Watch –älykellon kanssa. Tehtävät liittyivät lähinnä sen projektin edistämiseen, mutta osallistun myös projektin loppuvaiheessa testaamiseen. Tehtäväni on varmistaa, että kellon integrointi sovellukseen toimii halutulla tavalla.
Päivän lopuksi kävin viimeisiä testitapauksia läpi, jotta saisin sovelluksen ulkoiseen testaukseen. Ulkoiseen testaukseen kuuluu noin 200 yrityksen valitsemia henkilöitä, jotka ovat halunneet mukaan testaajiksi. Niin sanotut beta-testaajat pääsevät etukäteen kokeilemaan sovellusta ennen kuin se julkaistaan ja yleensä testaajilta saadaan hyvää ja rakentavaa palautetta ja virhetilanteita. Tämä estää virheellisen version julkaisun. Testaamisen jälkeen lähetin iTunes Connect –ohjelman kautta kutsuja testaajille, jotta he pääsevät kokeilemaan sovellusta. Seuraavaksi testaajilta kysytään sopivan ajan kuluttua palautetta, jonka mukaan tehdään sovellukseen joko muutoksia tai se julkaistaan viralliseen sovelluskauppaan. Tänään oli tarkoitus esittää parannusehdotus regressiotestaukselle, mutta valitettavasti kaikilla oli kiireitä kokouksien ja muiden tehtävien takia, joten pidän esityksen myöhemmin mahdollisimman pian.

Keskiviikko 9.9.2015

Beta-testaajien palautteen mukaan loin tänään pikaversion, jossa oli pieniä muutoksia. Esimerkiksi muutamasta kielestä oli tarkoituksella jätetty pois käännös, sillä käännösyrityksillä voi mennä pari päivää käännöstilauksissa. Versiota ei luonnollisesti tarvinnut erikseen enää testata, vaan versio meni suoraan ulkoisten käyttäjien saataville. Päivän tavoitteena oli viimeistellä julkaistava versio ja miettiä tulisiko siihen enää mitään uusia korjauksia tai muutoksia.

Koska uudessa versiossa sovellukseen lisättiin japanin kielen tuki, App Storeen täytyi päivittää kuvakaappaukset sovelluksesta. Käytännössä otin kehitysympäristön avulla kaappauksia sovelluksen viidestä tärkeimmästä kohdasta, mutta olen ottanut niitä aikaisemmin usealle kielelle, joten pyrin ottamaan samanlaiset kuvat kaikille kielille. Tehtävä ei itsessään ole haastava, mutta kuvat täytyy ruudun kokojen takia ottaa kaikille iPhone-malleille. Lopuksi kuvat lisätään ohjelman avulla mukaan uuteen sovellusversioehdokkaaseen.

Seuraava tehtäväni oli testata uutta tulevaa ominaisuutta, joka liittyy sovelluksen tietojen jakamista iOS-puhelimen omaan terveyssovellukseen. Tähän osa-alueeseen ei varsinaisesti löytynyt ajettavia testitapauksia, joten ennen kuin aloin testaamaan ominaisuutta, minun tuli miettiä ja kirjoittaa testitapauksia. On hyvä regressiotestauksen kannalta, että jokaiselle sovelluksen osa-alueelle löytyy funktionaalisia testejä ja tietenkin tilanteen vaatiessa niitä tulee lisätä ja päivittää. Funktionaaliset testit ovat erilaisia testitapauksia, jotka

varmistavat, että ominaisuus toimii niin kuin on suunniteltu. Ne koostuvat nimestä, suoritettavista kohdista, variaatioista ja lopputuloksista eri versioilla. Nimi kuvaa mitä testissä tehdään ja mikä on haluttu lopputulos ja sitä käydään läpi kohdittain eri variaatioilla, joita voi olla esimerkiksi eri älypuhelimen mallit tai käyttöjärjestelmät. Jokaiselle sovellusversiolle ja variaatiolle, jolle testit on tehty, merkitään onko se hyväksytysti läpäisty.

Päivän päätteeksi sain vielä esittää parannusehdotukseni regressiotestaukselle ja sain palautetta siihen. Ensimmäisen viikon viikkoanalyysissä kävin prosessin läpi eikä muilla ollut mitään sitä vastaan. Koska yrityksellä ei ole ollut virallista regressiotestausprosessia muut kuuntelivat ehdotustani mielenkiinnolla ja tukivat sitä. Testitapaushallinta-ohjelman käytöstä muut suosittelivat ottamaan selvään mitä kaikkia etuisuuksia ohjelma toisi ja harkitsemaan käyttöön ottamista. Tällä hetkellä kaikki testitapaukset on tallennettu taulukkolaskentaohjelmaan ja tulevaisuudessa testitapaukset voivat paisua isoiksi. Ohjelma tekisi testitapausten läpikäyntiä mielekkäämpää paremman käyttöliittymän avulla ja testit pysyisivät järjestelmässä tallessa.

Perjantai 11.9.2015

Sain tehtäväksi ennen työpäivän alkua hakea toimistolle uusi iOS-laite, jota voidaan käyttää sovelluksen testauksessa. Pääasiassa iOS-laite tulee minun käyttööni, sillä syksyllä yritys julkaisee ison ominaisuuden, joka lisää toiminnallisuutta Apple Watch –älykellolle. Koska itselläni on käytössä iOS-laite, joka ei tue kelloa, yritys osti iPhone 5S –älypuhelimen. Olen suuressa roolissa ominaisuuden testauksessa, joten on tärkeää, että pääsen testaamaan prototyyppejä mahdollisimman aikaisessa vaiheessa.

Keskiviikkona kirjoitin testitapaukset uudelle tulevalle ominaisuudelle, joten päivän tavoite oli tehdä testit ja raportoida poikkeuksista. Testien ajaminen itsessään ei ole varsinaisesti vaikeaa, mutta niiden kirjoittamisessa täytyy olla luova. Täytyy miettiä kaikkia eri vaihtoehtoja miten ominaisuutta voi testata. Koska kyseessä on terveystietojen jakaminen, testitapauksia voi olla kolme: Joko kaikki tiedot tai vain osa tiedoista jaetaan. On myös mahdollista, ettei mitään tietoja jaeta. Käyttäjä voi itse päättää mitä tietoja jaetaan, joten käyttötapauksia on mietitty käyttäjän näkökulmasta. Testeissä on tärkeää, että käydään mahdollisimman monta käyttötilannetta. Ei ole olemassa yleistä kaavaa kuinka paljon testausta tulisi tehdä, vaan jokainen tilanne täytyy arvioida erikseen (Pettichord 2002, 170).

Suurin osa testeistä meni läpi, mutta yksi käyttötapaus ei täyttynyt. Testitapauksilla tehdään tarkat step-by-step –kuvaukset ja ominaisuus ei siltä osalta selkeästi toiminut. Suori-

tin ensin muut testit loppuun ja lopuksi kirjoitin kattavan raportin virheestä. Kuten olen aikaisemmin maininnut, raportin tulee olla selkeä ja siinä tulee olla ohjeet kuinka ongelman voi toistaa. Näin kehittäjät pystyvät itse toistamaan ongelman ja sitä kautta myös korjaamaan sen. Raportissa tulee olla myös versio, jossa ongelma on ilmennyt. Jos versiota ei mainita, se vähentää raportin laatua, koska kehittäjä ei voi arvata missä ja milloin se on ilmennyt. Päivän lopuksi ilmoitin vielä kehittäjille tuloksista ja mainitsin virhetilanteesta sekä raportin olemassaolosta.

Viikkoanalyysi

Viikon aikana pääsin harjoittelemaan testitapausten kirjoittamista sekä sisäisen ja ulkoisen testauksen hallinnointia. Ennen uuden sovellusversion julkaisua yritys on aikaisemmin käyttänyt sisäistä ja ulkoista testausta, jolloin toinen henkilö on ollut vastuussa siitä. En ole aikaisemmin ollut varsinaisesti vastuussa tästä prosessista, mutta kehittäjänä se oli tuttu prosessi. Koska se on koettu toimivaksi, en katsonut että sitä tulisi muuttaa millään tavalla. Tietenkin aktiviinen palautteen kerääminen ja tehokas eteneminen julkaisuprosessissa auttaa sovelluksen uusien ominaisuuksien ulos julkaisua. Beta-testauksen avulla sovellus validoidaan ennen julkaisua tosi-elämässä sovellusta käyttävien käyttäjien avulla (Patton 2001, 244).

Olen miettinyt miten ja miksi ottaisin käyttöön testitapaushallinta-ohjelman – mitä hyötyjä ja haittoja se toisi nykyiseen prosessiin. Waterhousin (2013) mukaan testitapaushallinta-ohjelman hyödyt voidaan jakaa kolmeen osa-alueeseen: hallinta, jäljitettävyys ja automaatio. Manuaaliset testitapaukset pysyvät paremmin hallinnassa mikäli ne on jäsennelty oikein. Samaa rakennetta käyttävien testitapausten muuttaminen vaatii muutoksen vain yhdessä paikassa. Tämä on selkeä etu verrattuna Excel-muotoiseen ratkaisuun, sillä olen jo huomannut että nykyisissä testitapauksissa on toistoa vaikka sitä ei ole paljon. Lisäksi jokaisen ominaisuuden funktionaaliset testit ovat omissa tiedostoissa, joka vaatii monen ikkunan aukipitämistä samanaikaisesti, kun taas ohjelmassa tarvitaan vain yksi ikkuna, jossa sijaitsee kaikki testitapaukset.

Taulukko 1. Testitapausten merkintämatriisi taulukkolaskentaohjelmalla yrityksessä

Test name	Requirements	Steps	Operational variations	Software version (Build number)	...

Yksi iso ongelma, joka tuli ilmi pitämäni esityksen jälkeen, oli testitulosten merkitseminen nykyisessä muodossaan. Jokaiselle versiolle luodaan oma sarake ja ongelmaksi muodostuu kun niitä kerääntyy useampi, jolloin taulukosta tulee erittäin pitkä. Ohjelman avulla usean testiajon tulokset näkyvät selkeämmin ja mahdollisesti niistä on tarjolla erilaisia metriikoita ja kuvaajia (Waterhouse 2013). Vaikka taulukot eivät toistaiseksi ole niin pitkiä, ongelma voi tulla vastaan lähiaikoina.

Ohjelman avulla testitapaukset voitaisiin helposti jäljittää johonkin pääominaisuuteen. Excel-tiedostot ovat erinäisiä kappaleita, jotka toki sijaitsevat samassa hakemistossa, mutta eivät ole liitetty mihinkään paitsi nimen avulla. Viimeinen Waterhousin (2013) nimeämä hyöty on testien automatisointi, joka ei ainakaan TestLodge-ohjelmassa ole suoranaisesti mahdollista. Testien automatisointi vaatii paljon suunnittelua ja työtä, joka kannattaa aloittaa jo kun tuotetta ei ole vielä edes toteutettu (Pettichord 2002, 124). Tämä ei ole enää yrityksessä mahdollista, mutta on hyvä pitää automaatio mielessä, sillä se helpottaisi testaajan työmäärää.

Sain luvan kokeilla ohjelmaa, mikäli se on minusta järkevää, joten olen päättänyt tilanteen vaatiessa ottaa TestLodge-ohjelman käyttöön. Kiireen keskellä voi olla haastavaa perehtyä uuteen järjestelmään, mutta viimeistään siinä vaiheessa kun taulukoista on enemmän haittaa kuin hyötyä, otan ohjelman käyttöönoton prioriteetiksi.

Kokeilin tällä viikolla ensimmäistä kertaa palautteen keräämistä sovelluksen sisäisessä testauksessa. Vaikka sitä on kerätty aikaisemmin, on tärkeää tehdä siitä virallinen vaihe, jonka avulla poistetaan turhat ohjelmointivirheet ulkoisilta testaajilta. Vaikka yritän löytää kaikki mahdolliset virheet, on tavallista, että en välttämättä näe niitä kaikkia ja kun sovellusta testaa mahdollisimman moni, virheet löydetään helpommin. Sen lisäksi, että ihmiset näkevät asiat erilailla, jokainen käyttää ja testaa sovellusta eritavalla. Toinen voi esimerkiksi painaa eri painikkeita eri kohdissa sovellusta. (Patton 2001, 243.) Kun sovellus julkaistaan ulkoiseen testaukseen, palautteen kerääminen täytyy tehdä järjestelmällisemmin, koska kaikkia beta-testaajia ei ole mahdollista haastatella suullisesti. Yritys on perustanut Facebook-ryhmän, johon testaajille ilmoitetaan uusista versioista ja johon he voivat antaa palautetta. Toinen vaihtoehto antaa palautetta on lähettää sähköpostia tai ottaa yhteyttä asiakastukeen. Jokainen käyttäjän palaute ei välttämättä ole suoranaisesti virhe sovelluksessa, vaan se voi olla käytettävyysongelma, joka on käyttäjän näkökulmasta virheellinen.

Ensi viikon tavoite on saada ulkoisessa testauksessa oleva versio App Store –kauppaan ja testata älykelloprototyyppiä. Aikataulu on tiukka, mutta jos aikaa on, perehdyn tarkemmin TestLodge-ohjelmaan ja siihen mitä kaikkia sen avulla voi tehdä. On tärkeää pitää

huolta siitä, että sovelluksen laadun kannalta jokaista testausvaihetta tullaan jatkossa noudattamaan. On erityisen tärkeää tehdä funktionaaliset testit ja kerätä käyttäjäpalautetta.

3.3 Seurantaviikko 38

Maanantai 14.9.2015

Ennen perinteistä aamukokousta, asensin ensimmäisen prototyypin älykellosovelluksesta. Koska sovellus kehitetään syksyllä ilmestyvälle watchOS 2:lle, prototyypin asennus vaatii uuden iOS 9 –käyttöjärjestelmän beta-version asennuksen. Asensin ensin uuden käyttöjärjestelmän, sitten itse prototyypin. Yrityksessä on käytössä muutama älykello, joten sain testikäyttöön yhden niistä. Aamukokouksessa kävimme jälleen läpi viikon tavoitteet ja tulevat tehtävät. Viikon tavoitteenani on saada uusi iOS-versio julkaistua ja testata älykelloprototyyppiä sitä mukaan kun sitä toteutetaan. Lisäksi viikon aikana tulisi lisätä Android-versioon käännöksiä, jotka on toteutettu iOS, muttei Android-versioon. Yritys keskittyy tällä hetkellä enemmän iOS- kuin Android-versioon, koska iOS-version käyttäjiä on enemmän.

Kokouksen jälkeen otin tehtäväksi tutkia Android-version nykyisen tilanteen. Japanin käännökset olivat suurimmaksi osaksi toteutettu lukuun ottamatta yhtä sovelluksen osaa, joka tarjoaa erilaisia neuvoja unen laadun parantamiseen. Koska tämä oli toteutettu iOS:lle, pystyin helposti pienen ohjelman avulla konvertoimaan tiedoston Androidiin sopivaan muotoon. Tehtävänäni oli myös tarkastaa kyseiset käännökset ja samalla huomasin, että tuetulle kielelle hollannille ei ollut käännöksiä. Tämä oli kuitenkin tiedossa ja kollega lupasi toimittaa käännökset minulle mahdollisimman pian. Kun käännökset on koottu, voin tehdä Androidista uuden version. Aasian markkinoilla kumppanit ovat kyselleet näitä lisäyksiä, joten deadline versiolle asetettiin alustavasti perjantaiksi.

Loppupäivän kirjoitin alustavia testitapauksia älykellosovellukselle. Huomasin, että koska sille oli kirjoitettu vaatimusmäärityksdokumentti, johon oli kuvattu käyttötapauksia, minun oli paljon helpompi kirjoittaa testitapauksia niiden perusteella. Tietenkin käyttötapauksissa ei oteta huomioon erikoisia tilanteita, joita olisi hyvä testata. Lokakuussa julkaistavassa sovelluksessa on kaksi pääominaisuutta, jotka täydentävät nykyistä iOS-sovellusta, joten jaoin testitapaukseni näiden ominaisuuksien perusteella. Testitapauksissa käytin aikaisempaa pohjaa, jota olen käyttänyt muiden luomieni testitapausten kirjoittamiseen. Mietin erilaisia testitapauksia, jotka voivat olla poikkeuksellisia, mutta kuitenkin mahdollisia. Kysyin

toiselta kehittäjältä määrityksiä, miten kussakin erikoistilanteessa sovelluksen tulisi käyttäytyä, mutta valitettavasti hän ei osannut vastata tähän, sillä kyseessä on prototyyppi ja kaikkia mahdollisia skenaarioita ei ole vielä mietitty. Aion vielä kysyä mielipiteitä kirjoittamastani testitapauksista, jotta saan parannusehdotuksia esimerkiksi puuttuvista tapauksista.

Keskiviikko 16.9.2015

Olen parin päivän aikana testannut älykellosovellusta omalla ajallani ja aamulla raportoin kehittäjälle löytämäni virheet, jotka liittyivät lähinnä käyttöliittymäelementteihin. Prototyyppi on tarkoituksella jätetty keskeneräiseksi, joten monet mainitsemani virheet tiedettiin jo. Varsinaisia testitapauksia ei vielä tässä vaiheessa kannata tehdä. Ne vasta kannattaa tehdä vasta kun kokonaisia ominaisuuksia on valmiina. Aamulla osallistuin myös kahden viikon välein pidettävään kokoukseen, jossa käytiin lähitulevaisuuden suunnitelmia tiimeittäin.

Uusi iOS-versio julkaistaan huomenna, joten tämän päivän tehtävänä on tehdä viimeisiä hyväksyntätestejä. Vielä ennen uuden version varsinaista julkaisua ei ole kehitetty tarkkaa prosessia, mutta koin, että käyn vielä sovelluksen pääominaisuudet läpi eri laitteilla ja käyttöjärjestelmillä. Koska pienin tuettu iOS-käyttöjärjestelmä on 7, niin kävin sovellusta manuaalisesti ja systemaattisesti läpi kyseisellä käyttöjärjestelmällä aina sisäänkirjautumisesta mittauksen aloitukseen. Koska lähes kaikilla beta-testaajilla on uusi laite, on tärkeää testata itse vanhemmat versiot, sillä eri käyttöjärjestelmät ja laitteet voivat syystä tai toisesta käyttäytyä eri tavalla.

Lopullisen manuaalisen testauksen lisäksi kävin läpi beta-testaajien antamia palautteita toisen työntekijän kanssa läpi. Palautteet kerättiin sähköpostikyselyllä, jossa beta-testaajilta kysyttiin kokemuksia uudesta versiosta verrattuna edelliseen. Monet pitivät uusista ominaisuuksista. Osa ehdotti parannuksia nykyiseen sovellukseen ja yhdellä käyttäjällä versio ei toiminut ollenkaan. Minun kannaltani tämä ei-toimiva yksilö oli kiinnostavin ja tutkin virhettä tarkemmin. Koska käyttäjän mukaan sovellus lakkasi toimimasta jokaisen yön aikana, tutkin Crittercism-ohjelman avulla löytyisikö käyttäjän sähköpostia vastaavaa kaatumisilmoitusta. Valitettavasti en löytänyt virheestä viittaavaa ilmoitusta, joten sovimme toisen työntekijän kanssa, että hän kysyisi käyttäjältä tarkempaa kuvausta tilanteesta.

Maanantaina tein alustavat testitapaukset älykellon testaamiselle ja tarkoituksena oli keskustella siitä muiden tuotetiimissä työskentelevien kanssa. Sovin perjantaiksi pieni muotoi-

sen kokouksen, jossa käydään yleisesti läpi älykellosovelluksen testausprosessia ja samalla saan tarkennuksia epäselväksi jääneisiin testitapauksiin. Koska suurimmalla osalla beta-käyttäjistä ei ole Apple Watch –älykelloa, olisi hyvä täydentää beta-testaajiksi myös kellon käyttäjiä.

Perjantai 18.9.2015

Eilen esimieheni oli tarkoituksena julkaista uusi iOS-versio App Store –kauppaan, mutta muiden kiireiden takia hän ei ehtinyt. Koska versio julkaistaan tämän päivän päätteeksi, minulla oli aikaa tehdä laajempaa manuaalista testausta. Testausta ei voi tehdä koskaan liikaa ja sen avulla voin entistä varmempana näyttää vihreätä valoa versiolle. Julkinen liikenne oli tänään lakossa koko päivän, joten emme viettäneet työpäivää perinteisesti toimistolla, vaan sen sijaan Helsingin keskustan alueella, joka toi työntekoon omat haasteensa.

Toimistolla on mielestäni parempi keskittyä omaan tekemiseen, kun taas yleisessä kahvilassa, johon keräännyimme tuotetiimin kanssa, tilan puute ja taustaäänet voivat häiritä työskentelyä. Lakko tuli yllättäen ja olimme sopineet tuotetiimin kanssa kokouksia, joita on aina parempi pitää kasvotusten. Loppujen lopuksi työskentely sujui hyvin ja tein viimeisiä manuaalisia testauksia lähes koko päivän.

Pidimme päivän aikana sopimani kokouksen, jossa käsiteltiin laatimiani testitapauksia yrityksen tulevalle älykellosovellukselle. Keskustelin lähinnä työntekijän kanssa, joka on keskittynyt tuotesuunnitteluun. Neuvottelimme siitä, mikä olisi jokaisen tapauksen haluttu lopputulos. Keksimme myös uusia poikkeustapauksia, joita itselleni ei ollut tullut mieleen ja lisäsimme ne testitapauslistalle. Esimerkiksi tilanne, jossa joko kellosta tai älypuhelimesta loppuu akku - miten sovelluksen tulisi käyttäytyä silloin. Keskustelimme myös yleisesti testitapauksista ja millaisia niiden kannattaisi olla. Käyttäjät antavat paljon palautetta tuotteesta ja vaikka funktionaalisesti testit menevät läpi, käyttäjän kannalta tuotteessa voi kuitenkin olla puutteita tai virheitä. Testitapauksia tulisi päivittää säännöllisesti ja niitä kannattaa miettiä käyttäjäkeskeisemmin. Tämä liittyy enemmän käyttäjäkokemuksen parantamiseen. Vaikka sen parantaminen ei varsinaisesti vastuullani, minunkin on hyvä miettiä testejä käyttäjän kannalta ja miten testitapaukset koetaan heidän näkökulmastaan.

Varsinaisen julkaisun tekee esimieheni, mutta päivän päätteeksi täytin vielä lopulliset tiedot liittyen uuteen julkaisuun, esimerkiksi julkaisupäivämäärä ja mitä manuaalisia testejä versiolle on tehty. Raporttiin tulee ulkoisesta testauksesta saatujen palautteiden yleisimmät ja kriittisimmät kommentit sekä Mixpanel-ohjelmasta saatuja analytiikkatilastoja, joita

voi olla esimerkiksi sovelluksen käyttökerrat erityisellä laitteella. Tilastojen avulla voidaan päätellä onko versiota testattu tarpeeksi. Vaikka varsinainen julkaisu tehdään painiketta napsauttamalla, on hyvä, että sen tekee esimieheni eli tuotetiimiä johtava työntekijä.

Viikkoanalyysi

Viikon päätavoite oli saada uusi iOS-versio ja viikon viimeisenä työpäivänä se saavutettiin. Olen tyytyväinen tulokseen. Viikon aikana oli tarkoitus myös kehittää ja testata pieniä ominaisuuksia Android-versiolle ja tämä tavoite tuli osittain saavutettua, mutta kuten olen aikaisemmin maininnut, Android ei ole yrityksen prioriteetti. Kun uusi sovellusversio julkaistaan toimivuuden varmistamiseksi, sitä seurataan eri työkalujen avulla.

Perjantaina tein viimeiset hyväksyntätestaukset sovellukselle ja seuraavan kolmen työpäivän aikana on tarkoitus seurata versio tilannetta tuotannossa. Sitä seurataan Crittercism-ohjelmalla, jolla nähdään millaisia virhetilanteita käyttäjille voi tulla ja miten paljon niitä esiintyy. Mikäli yhtä erityistä virhettä esiintyy useasti, sen korjaaminen täytyy ottaa käsittelyyn mahdollisimman nopeasti. Koska versio on testattu hyvin, mitään suuria yllätyksiä ei pitäisi esiintyä. Analytiikkatilastoja voidaan seurata, mutta asiakastukeen tulevat palautteet ovat tärkeässä roolissa. Kolmen päivän aikana on tarkoitus seurata asiakastukeen lähettyjä palautteita ja niiden avulla analysoida, onko jokin vialla sovelluksessa.

Software Testing Help –sivuston artikkeli (2015) kuvailee miten ohjelmiston julkaisuprosessia voi parantaa erilaisin menetelmin. Sivusto ehdottaa, että juuri julkaisun jälkeen tuotetta kehittävä ryhmä voi pitää tilanneraporttikokouksen, jossa käytäisiin läpi version tuomat hyödyt ja haitat sekä keskusteltaisiin vakavimmista tiedostetuista ohjelmointivirheistä. Yrityksen tuotetiimin kohdalla ajankohta nopealle kokoukselle voisi olla seuraava työpäivä, mutta toisaalta sen pitäminen olisi kuitenkin mielestäni järkevää heti, mikäli versiossa tulee ilmi tapauksia, josta tulisi keskustella. Seurannan kannalta artikkelissa (2015) kehotetaan pitämään huolta, että siihen tarkoitetut työkalut on määritetty oikein. Yrityksen tuotteen tapauksessa on tärkeää, että Crittercism-, Mixpanel- ja Sentry-ohjelmat ovat toimivia ja ne raportoivat tietoja uuden version tapahtumista. Julkaisua suositeltiin tehtävän viikon ensimmäisenä työpäivänä, koska tieto mahdollisista ongelmista saadaan näin nopeammin. Viikonlopun aikana käyttäjille voi esiintyä vakaviakin ongelmia, jolloin siihen päästään kunnolla käsiksi vasta maanantaina – ensi kerralla tämä on hyvä pitää mielessä.

Kehittämäni prosessi hyväksyntä- ja regressiotestaukselle on toiminut tehokkaasti, sillä nyt olen paremmin tietoinen siitä, miten uusia julkaisuehdokkaita tulisi testata. Kun tiedän mihin kohtiin sovellusta on tehty muutoksia, pystyn sen mukaan tekemään manuaalisia

testejä. Haastavaa mielestäni on sovelluksen testaaminen käyttäjän silmin, sillä olen itse kehittänyt ja monta kertaa testannut sitä. Käyttäjän odotukset ja tarpeet on ymmärrettävä, sillä se määrittää lopulta tuotteen laadun (Patton 2001, 58). Tiedän miten kukin näkymä toimii ja suurilta osin miten se on teknisesti toteutettu. Käyttäjällä taas voi tulla vastaan sovelluksesta ongelma tai epäselvyys, joka olisi minulle nyt selkeä. Hyvä testaaja ajattelee teknisesti, luovasti, kriittisesti ja käytännöllisesti. Teknisellä tasolla testaajan on ymmärrettävä faktat järjestelmästä ja osattava käyttää siihen liittyviä työkaluja hyödykseen sekä kyettävä luovasti keksimään erilaisia testausskenaarioita. Testaaja löytää vain ongelmat, jotka hän voi itse kuvitella. Testaajan on myös pystyttävä asennoitumaan kriittisesti erilaisista ideoihin ja toteuttamaan erilaiset positiiviset asiat käytännössä, esimerkiksi erilaisten testityökalujen käyttö. (Pettichord 2002, 15.) Jatkossa minun on ajateltava testaustani laajemmin – millaisia parannuksia tuotteelle voi tehdä, jotta käyttötapauksista, ja siten testitapauksista, tulee käyttäjien näkökulmasta luontevia ja järkeviä. Käytännössä tämä voi tarkoittaa omien testitapausten päivittämistä.

Opin joka viikko uutta ohjelmistotestauksesta ja pyrin aina parantamaan tuotteen testausta yhdessä tuotetiimin kanssa. Hyvän testaajan täytyy oppia ajattelemaan kuin sellainen (Pettichord 2002, 15). En ole aikaisemmin hallinnoinut julkaisun jälkeistä prosessia, joten ensi viikolla seuraan tarkasti uutta versiota esitetyillä menetelmillä. Ehdotan seuraavalla kerralla, mikäli mahdollista, että julkaisu tehtäisiin, jos ei maanantaina, niin alkuviikosta, jolloin pääsen nopeammin seuraamaan julkaisua. Yleisellä tasolla olen oppinut iOS-sovellusversion julkaisun kokonaisuutena, jolloin seuraavalla kerralla omalta osaltani prosessi tulee olemaan nopeampi ja tehokkaampi.

3.4 Seurantaviikko 39

Maanantai 21.9.2015

Sovimme viikonlopun aikana tuotetiimin kanssa, että maanantaina pitäisimme heti aamusta, suunnittelukokouksen tälle viikolle. Apple Watch –sovelluksen tulee olla valmis ja julkaistu käyttäjille lokakuun alussa ja paljon on tehtävää, joten jaoimme kullekin jäsenelle työtehtäviä prioriteettien mukaan. Meillä oli jälleen koko yrityksen yhteinen kokous, jossa kerrottiin edellisen viikon saavutuksista ja kyseisen viikon tavoitteet. Suurin osa työajastani tulee menemään älykellon testaamiseen, julkaisuhallinnointiin ja laadunvarmistukseen, ja jos aikaa jää työskentelen uuden Android-version julkaisun parissa.

Koska perjantaina uusi versio julkaistiin App Store –kauppaan, katsoin aamukokouksien jälkeen version tilanteen työkalujen avulla. Toistaiseksi mitään vakavaa ei ole tullut vastaan, mutta seuraan tilannetta kuitenkin tarkasti ainakin tämän viikon aikana, sillä kaikki eivät vielä todennäköisesti ole asentaneet uusinta versiota. Yrityksen yksityisessä Facebook-ryhmässä käyttäjä ilmoitti, että sovellus ei toimi uudella iOS 9 –käyttöjärjestelmällä. Olen todennut sovelluksen toimivaksi kyseisellä käyttöjärjestelmällä, joten uskon, että ongelma johtuu jostain muusta. Kysyin käyttäjältä lisätietoja ongelmasta ja käyttämästään sovellusversiosta. Tuotetta on saatavilla kahta eri mallia, joista kumpikin käyttää eri Bluetooth-versiota. Käyttäjän vastauksen perusteella voin paremmin analysoida mistä ongelma voi johtua.

Sain päivän tehtäväksi päivittää kaikkien Apple Watch –älykelloa käyttävien työntekijöiden puhelimet ja kellot uusin käyttöjärjestelmäversioihin. Yrityksen käytössä on useampi kello, joka vaativat päivityksen juuri julkaistuun watchOS 2 -käyttöjärjestelmän beta-versioon, jossa on paljon korjauksia. Prototyyppi-sovellus vaati näitä korjauksia toimiakseen. Koska muutamat yrityksen työntekijöistä asuvat ja tekevät töitä Yhdysvalloista käsin, kirjoitin vielä päivän päätteeksi ohjeet miten he voivat asentaa käyttöjärjestelmät itse.

Keskiviikkona tarkoitus on julkaista ensimmäinen versio älykellosta ulkoiseen testaukseen, joten vaikka tiistaina en ole toimistolla, suoritan tekemiäni testitapauksia sovellukselle joko tiistaina tai keskiviikkona. Testitapaukset on kirjoitettu jo viime viikolla, joten testaus tulee menemään suoraviivaisesti. Koska kyseessä ei ole valmis sovellus, kirjoitan raportin löytämistäni virheistä ja havainnoista.

Keskiviikko 23.9.2015

Päivän tavoitteena on aikataulullisesti julkaista ensimmäinen versio sovelluksesta, joka sisältää älykellosovelluksen. Ulkoiseen testaukseen julkaiseminen tässä vaiheessa on hyvä tapa saada beta-käyttäjiltä palautetta, jonka perusteella sovellukseen voidaan tehdä muutoksia. Varsinkin kellon testaamiseen liittyy moni asia, koska kellosovelluksen täytyy toimia yhdessä iOS-sovelluksen kanssa. Tietoa lähetetään kellon ja puhelimen välillä edes takaisin ja kaikenlaisia yhteysongelmia voi esiintyä.

Päivän tavoiteeni oli tehdä täysi manuaalinen testaus tekemilläni testitapauksilla ja päivän lopussa kirjoittaa testiraportti, jonka avulla kehittäjät saavat hyvän kuvan siitä, millä tasolla sovellus toimii eri tilanteissa. Aamulla töihin tultuani asensin beta-versioehdokkaan sovelluksesta, jonka avulla voin käydä testitapauksia läpi. Löysin hyvinkin monta käytettävyy-

teen liittyvää ongelmaa, josta jokaisesta tilanteesta otin kuvakaappauksen, jonka voin lopuksi lisätä testiraporttiin. Kello luo uutta dataa, joka integroidaan iOS-sovelluksessa olevaan dataa. Testauksen aikana löysin vakavan ohjelmointivirheen, joka ylikirjoittaa puhelimessa olevaa dataa, joten kävimme ongelmaa yhdessä läpi toisen kehittäjän kanssa.

Koska määräaika beta-versiolle on tänään, loin uuden version, jossa oli korjaus löytämälleni virheelle. Korjauksen avulla pystyin suorittamaan loput testeistä ja päivän lopuksi kirjoitin uusimman version pohjalta testiraportin. Yrityksessä ei ole aikaisemmin ollut tapana kirjoittaa virallisia raportteja koska johtopäätöksiä tehtiin aikaisemmin suullisesti ja testitapaustuloksien perusteella. Raportissa kuvailin mitkä tapaukset suoritin, mitkä niistä läpäisivät ja mitkä eivät, ja lisäksi kirjoitin sanallisen kuvauksen kokemuksesta, johon lisäsin mukaan kuvakaappauksia tilanteista, joissa oli mielestäni virheitä. Sain hyvää palautetta raportista. Kirjallisuuden ja internetin avulla voin miettiä loogisen pohjan, jolla voidaan muodostaa jatkossa yhteneväisiä raportteja.

Beta-julkaisemisessa oli ongelmia ja yritin vielä illalla selvittää mistä ongelma johtuu. Keskustelupalstalta paljastui, että muillakin iOS-kehittäjillä oli ollut viime aikoina sama ongelma, joka viittasi siihen, että Applella oli teknisiä vikoja. Toistaiseksi täytyy odottaa, että Apple korjaa ongelman, jotta testiversion julkaiseminen ulkoisille testaajille on mahdollista.

Perjantai 25.9.2015

Koska aikataulu on kiireinen, työskentelin myös torstai-iltana älykellosovelluksen kanssa. Beta-versio saatiin torstaiaamuna ongelmien jälkeen yrityksen sisäiseen testaukseen ja tehtäväni oli varmistaa, että versio voidaan julkaista ensimmäistä kertaa ulkoiseen testaukseen. Manuaalisen testauksen tuloksena totesin, että älykellosovellus ei riko muita osia sovelluksesta ja se voidaan julkaista ulkoiseen testaukseen. Yritys oli jo myöhässä tästä ensimmäisestä beta-versiosta, joten aikaa sisäiselle testaukselle ei jäänyt, joka ei välttämättä ole hyvä asia.

Aamulla ennen älykellosovelluksen palautteiden keräämistä, tarkistin App Storen –version tilanteen Crittercism-ohjelmalla. Mitään suurta virhettä ei ollut esiintynyt, mutta yhtä virhettä oli selkeästi muita yleisemmin. Ensin selvitin mistä ongelma voi johtua ja vasta sen jälkeen keskustelin siitä kehittäjän kanssa. Ongelma oli hyvin tekninen ja monimutkainen, sillä ohjelman tuottaman raportin perusteella ei voinut selvittää, mistä virhe on peräisin. Yleensä virheistä kirjoitetaan raportti, mutta koska raporttiin vaaditaan ohjeet toistamiselle ja kuvailua, päätin keskustella siitä mieluummin kokeneemman kehittäjän kanssa. Hän sai

paremman kuvan ongelmasta ja päätimme, että hän ottaa korjaamisen tehtäväksi myöhemmin.

Työntekijältä tuli palautetta, että beta-versiossa oleva sovellus ei käynnistynyt vanhemmilla iOS-käyttöjärjestelmillä. Koska kohdistin manuaalisen testauksen uusimpaan käyttöjärjestelmään kelloa varten, en käynyt läpi vanhinta tuettua käyttöjärjestelmää. Virallisena testaajana se oli virhe, mutta onneksi lähes jokaisella beta-testaajalla on uusin tai melkein uusin iOS-käyttöjärjestelmä. Päivän aikana sain testauksen avulla selville mistä virhe johtuu ja tein siitä kehittäjälle virheraportin ja hän sai virheen korjattua nopeasti.

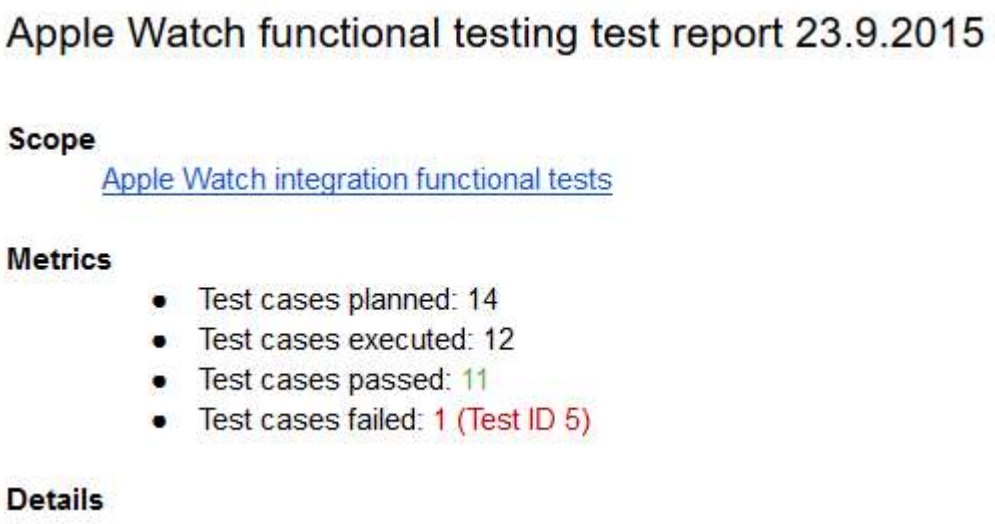
Lisäsin keskiviikkona tekemääni testiraporttiin kuvakaappauksia, jonka avulla priorisoitiin korjaukset nykyiselle beta-versiolle. Kehittäjiltä meni korjauksiin koko päivä. Korjaukset olivat lähinnä pieniä käyttöliittymään liittyviä muutoksia. Päivän päätteeksi loin uuden sisäisen version, johon kuuluu paljon korjauksia, mukaan lukien vanhemman käyttöjärjestelmän virheen korjaaminen. On hyvä, että korjaus saatiin tänään, sillä näin työntekijöillä on viikonloppu aikaa testata kellon toimintaa. Maanantaihin mennessä beta-käyttäjiltä saadaan myös palautetta, jonka avulla voidaan tehdä lisää korjauksia. Virallinen julkaisupäivä kellosovellukselle on lokakuun alussa, joten onneksi aikaa on. Tosin seuraavat viikot tulevat olemaan kiireisiä.

Viikkoanalyysi

Verrattuna muihin työviikkoihin ohjelmistotestaajan roolissa, tämä viikko oli minulle kaikkein kiiresin ja työteliäin. Päätyötehtäväni oli älykellosovelluksen täysi manuaalinen testaus ja regressiotestaus, sillä se integroitiin nykyiseen iOS-sovellukseen. Ilmeni, että kaiken testaaminen ei ole mahdollista ja minun täytyi viikon aikana priorisoida se mitä kaikkea haluan testata, jotta beta-versiosta tulee mahdollisimman eheä. Nykyisessä beta-versiossa on mahdollisesti virheitä, mutta ne liittyvät enemmän käytettävyyteen ja sovelluksen visuaaliseen ilmeeseen. Minun tehtäväni oli varmistaa, että sekä iOS- että älykellosovelluksen pääominaisuudet toimivat ja mikään uusi osa ei riko nykyisiä ominaisuuksia.

Ensimmäistä kertaa tällä viikolla kirjoitin tekemälleni testitapauksille raportin, jossa kuvailin löydöksiä ja merkitsin tuloksia istunnosta. Aiemmin yrityksessä ei ole varsinaisesti tehtyä dokumenttipohjaisia raportteja, johon merkitään esiin tulleita asioita testauksesta. Raportin luonnin aikana mietin millaisia asioita ja mihin muotoon merkinnät tulisi kirjoittaa ja kiireen takia en ehtinyt ottamaan mitään pohjaa käyttöön.

Jotta virallisen testidokumenttipohjan voi ottaa käyttöön, täytyy ymmärtää jokainen siinä esiintyvä kohta. Miksi ne ovat siinä ja milloin kohtia voi poistaa? Jos pohjaa ei sisäistä ja monet kohdat ovat epäselviä testaajalle, sitä ei kannata ottaa käyttöön. (Pettichord 2002, 131.) Viikon aikana en käyttänyt mitään pohjaa, vaan sen sijaan pohdin mitä kohtia haluan tuoda esille ja kirjoitin raportin yksinkertaiseen Google Sheet –dokumenttiin, joka on kaikkien työntekijöiden saatavilla yhteisessä Google Drive -pilvipalvelimella. Sain hyvää palautetta raportistani ja mielestäni sen kannalta sisältö on tärkein aspekti, mutta ulkoasun täytyy olla selkeä ja ytimekäs. Kehittäjät voivat hyvän formatoidun raportin pohjalta saada hyvän kuvan kaikista testeistä, jotka eivät mene läpi ja miksi ne epäonnistuvat. Tarkoitus olisi jatkossa tehdä epäonnistuneista tapauksista virheraportti FogBugz-ohjelmaan, jonka avulla kehittäjät voivat korjata virheen.


Apple Watch functional testing test report 23.9.2015

Scope

Apple Watch integration functional tests

Metrics

- Test cases planned: 14
- Test cases executed: 12
- Test cases passed: 11
- Test cases failed: 1 (Test ID 5)

Details

Kuva 3. Yrityksen ensimmäinen testiraportti

Ensimmäisessä testiraportissa mainitsin otsikolla, mikä on testauksen päätarkoitus ja milloin se on tehty. Tärkeää mielestäni on määrittää mitkä funktionaaliset testit ajettiin, näyttää tilastoja niiden kokonaismäärästä ja onnistumisprosentti sekä kertoa sanallisesti, mitkä ja miksi jotkut testit epäonnistuivat. Lopuksi vielä lisäsin kuvakaappauksia tilanteista, jotka eivät varsinaisesti olleet virheitä, mutta parannusehdotuksia käyttöliittymään. Pyrin jäljittämään raportin niin, että lisäsin linkin testitapauksiin, jotka suoritin.

Software Testing Helpin (2015) artikkelissa kuvataan 12 askelta tehokkaaseen testiraporttiin:

1. Dokumentin tarkoitus
2. Testattavan kohteen kuvaus
3. Testauslaajuus
4. Tilastot

5. Testausmenetelmät
6. Testausympäristö ja työkalut
7. Opitut asiat
8. Suositukset
9. Parhaat käytänteet
10. Poistumiskriteerit
11. Johtopäätös
12. Määritelmät ja lyhenteet

Ottaen huomioon yrityksen koko ja koska yrityksessä on yksi ohjelmistotestaaja, kaikkia näitä kohtia ei ole mahdollista, eikä kannata, joka kerta täyttää raportissa. Huomasin listasta, että tekemäni raportti täyttää muutaman osan, esimerkiksi merkitsin siihen testauslaajuuden eli mitkä testit suoritetaan ja tilastoin tulokset. Riippuen testauslaajuudesta voidaan jatkossa lisätä myös muita kohtia, mutta kohdat 1., 3., 4., 5. ja 11. tulisi olla pakollisia. Testauksella täytyy olla jokin tarkoitus. Laajuus täytyy määrittää ja tulokset on merkittävä sekä jonkin asteiseen johtopäätökseen on päästävä, mutta tilanteen vaatiessa voidaan esimerkiksi erikoiset testausympäristöt määrittää.

Pattonin (2001, 278–280) mukaan on yleisesti neljä järjestelmää, jossa voi säilyttää testitapauksia ja niiden tuloksia: päässä, paperilla, taulukkolaskentaohjelmassa ja siihen tarkoitetulla ohjelmalla, jossa on oma tietokanta. Tämän kokoisessa yrityksessä ja tuotteessa omassa päässä tiedon säilyttäminen ei ole järkevää ja yksinkertaiset tekstipohjaiset dokumentit voivat mennä nopeasti sekaviksi. Säilytän tällä hetkellä taulukossa kaikkia testitapauksia, johon merkitään tulokset ohjelmistoversioittain. Raportit kirjoitin tekstimuotoisena. Viimeinen vaihtoehto, jossa käytetään testaukseen tarkoitettua ohjelmaa, raportit voitaisiin jäljittää niiden testitapauksiin ja se selkeyttäisi kokonaistilannetta. Aikaisemmilla viikoilla mainitsin TestLodge-ohjelman, jonka avulla testitapaukset ja tulokset säilytetäisiin samassa paikassa ja niiden visualisointi olisi selkeämpää.

Puhuimme viikonloppuna esimieheni kanssa siitä, että mikäli aikaa on, kokeilen TestLodge-ohjelmaa testaustyökaluna. Ensimmäinen askel on siirtää taulukkolaskentaohjelmassa sijaitsevat testitapaukset ohjelmaan, mutta koska vasta kokeilen ohjelmaa, siirrän vain muutaman setin ensin. Ohjelmassa on myös mahdollista saada testitulokset manuaalisen testauksen lopputuloksena, joten aion verrata ohjelman tuottamaa raporttia omaan kirjoittamaani raporttiin. Oletukseni on, että ohjelman generoima raportti ei riitä, vaan tarvitaan myös sanallinen kuvaus testi-istunnosta. Ensimmäinen raportti oli karkea, mutta tärkeintä on tehdä siitä mahdollisimman ytimekäs, joten vaikka Software Testing Helpin (2015) artikkelissa suositellaan 12. eri askelta, pyrin löytämään tasopainon tarvittavan askel määrään suhteen, jotta raportti ei paisu liian monimutkaiseksi.

3.5 Seurantaviikko 40

Maanantai 28.9.2015

Päivän päätehtävänä oli varmistaa, että uusi iOS-sovellusversio, joka sisältää älykellosovelluksen, on valmis laitettavaksi Applen katselmoitavaksi. Beta-versio on ollut ulkoisessa testauksessa koko viikonlopun ja sitä on testannut aktiivisesti myös kaikki yrityksen työntekijät. Viikoittaisessa aamukokouksessa kerroin viikon tavoitteeni, johon kuuluu uuden version manuaalisen testaaminen lisäksi esiintyvien ohjelmointivirheiden kirjaaminen ja Android-sovellusversion luonti.

Viikonlopun aikana muilta työntekijöiltä oli tullut ideoita uusista testitapauksista liittyen älykellosovellukseen. Ne kannatti lisätä muiden joukkoon. Kun muut testaavat sovellusta, voi tulla esiin uusia näkökulmia sovelluksesta, joita itse ei ole tullut ajatelleeksi. Yksi työntekijöistä oli ilmoittanut virheellisestä tilanteesta, jossa käyttöliittymä oli käyttäytynyt oudosti, joten kirjasin virheen ylös kehittäjälle korjattavaksi.

Koska iOS-versio oli laitettava tänään Applen katselmoitavaksi ja ensi viikolla sovelluksen täytyy olla App Store –kaupassa, keräsin kaikki kriittisimmät ohjelmointivirheet pienille paperilapuille, jotta voisimme kehittäjien kanssa miettiä mitkä kannattaa ja mitkä ehditään korjaamaan. Kaikkia virheitä emme saisi korjattua, joten valitsimme tärkeimmät kokonaisuutta ajatellen. Päivän aikana kehittäjät saivat viimeisiä lisäyksiä ja korjauksia valmiiksi ja testasin niitä eri laiteilla ja käyttöjärjestelmillä. Suunnittelemissani prosesseissa kehittäjän tehtävänä on testata omat ominaisuudet ja korjaukset, mutta kiireiden vuoksi autoin muita saamaan prosessin nopeammaksi. Koska käännökset älykellosovelluksen saatiin vasta tänään käännösyritykseltä, jouduin myös testaamaan kaikki lokalisoidut kielet. Minulla oli aikaa myös kirjoittaa kaikille uuden version testanneille beta-käyttäjille sähköposti, jossa kysyin kokemuksia sovelluksesta, varsinkin jos jollakin oli mahdollisuus testata älykellon kanssa. Sähköpostissa kysyin kokemuksia sovelluksen toimivuudesta uuden iOS 9 –käyttöjärjestelmän kanssa, mutta kerään palautteet vasta seuraavana työpäivänä, kun suurin osa käyttäjistä on ehtinyt vastaamaan.

Päivän loppupuolella loin viimeisen sovellusversion, joka tulisi menemään Applen katselmoitavaksi. Se sisälsi lähinnä korjauksia verrattuna edelliseen, mutta koska aika ei riittänyt, osaa ohjelmointivirheistä ei ehditty korjaamaan. Sovellukseen tehtiin paljon visuaalisia

muutoksia ja liitettiin älykellosovellus, tehtävänä oli ottaa App Store –kauppaan uudet kuvakaappaukset. Tehtävä vie oman aikansa päivästä, mutta olen tyytyväinen siihen, että saimme lopulta sovelluksen julkaisukuntoon ja arvioitavaksi aikataulun mukaan.

Keskiviikko 30.9.2015

Päivä alkoi perinteisellä bi-weekly-kokouksella, jossa jälleen käytiin sekä markkinointitiimin, että tuotetiimin saavutukset, suurimmat haasteet ja suunnitelmat seuraaville viikoille. Tuotetiimi kertoi maanantain kohokohdasta, jossa uusi iOS-sovellusversio lähetettiin Applen arvioitavaksi. Suunnitteilla on lähettää parin päivän julkaisun jälkeen virheiden korjausversio, jota aiotaan työstää tällä viikolla.

Kokouksen jälkeen ennen tuotetiimin kesken pidettävää suunnittelukokousta, ehdin vastaamaan muutamaan beta-käyttäjän lähettämään palautteeseen. Kehujen lisäksi, muutamat käyttäjät ilmoittivat erilaisista virhetilanteista testiversiota käyttäessä. Jotkut virheistä olivat jo tiedostettuja ja niitä tullaan korjaamaan lähitulevaisuudessa. Ne saadaan mahdollisesti korjattua seuraavaan versioon mennessä. Yksi uusi virhe löytyi, joten kirjoitin siitä virheraportin. Aamupäivällä pidimme tuotetiimin kanssa suunnittelukokouksen, jossa kävimme läpi viikon työtehtävät ja tavoitteet. Kuten aikaisemmin mainitsin, uusi versio julkaistaan ensi viikon alussa ja tarkoituksena on lähettää pari päivää sen jälkeen Applen katselmoitavaksi versio, joka sisältäisi lähinnä käyttöliittymäparannuksia ja virheiden korjauksia. Vaikka julkaisun yhteydessä virheiden määrä voi kasvaa, monet virhetilanteet on jo tiedostettu ja niitä ei ole ehditty korjaamaan aikataulun vuoksi.

Suunnittelukokouksen jälkeen sain päivän työtehtäväksi luoda uuden Android-version, jota on jo pitkään toivottu. Versio sisältää käännöksiä muutamalle uudelle kielelle, jotka olivat jo osaksi toteutettu. Lisäsin loput käännöksistä ja pienen testauksen jälkeen laitoin version Android-ryhmän sisäiseen testaukseen. Vaikka tein ensimmäistä kertaa uutta Android-versiota, prosessi tuntui tutulta. Koska käännökset tulee vielä tarkistaa, lähetin apk-tiedoston työntekijälle, joka välittää sen henkilöille, joiden äidinkielenä on yksi käännetyistä kielistä. Verrattuna iOS-julkaisuprosessiin, uusissa Android-versioissa ei tarvitse odottaa viikkoa, jotta se saadaan viralliseen kauppaan.

Ehdin Android-tehtävän ohella luoda yritykselle tunnukset TestLodge-testienhallintaohjelmaan. Päätin kokeilla ohjelmaa, sillä olen huomannut manuaalista testausta tehdessä, että työskentelyyn liittyy paljon eri ikkunoihin siirtymistä ja kokonaisuutena prosessi on sekava. Ohjelma tuo mukanaan paljon eri ominaisuuksia, joita voin hyödyntää, esimerkiksi metriikoiden automaattinen generointi ja muihin palveluihin integroiminen. Kokeilun vuoksi

siirsin yhden testitapaustaulukkoni ohjelmaan ja yllättävää oli, että se tarjosi vaihtoehdon, jolla taulukot voidaan tuoda siihen automaattisesti. Sillä voidaan myös luoda testisuunnitelmia, joten voisin virheiden korjaus versiota varten laatia suunnitelman ja lisätä muut tarvittavat testitapaukset.

Perjantai 2.10.2015

Torstaipäivällä laitoin Android-version sisäisestä testauksesta ulkoiseen testaukseen. Muutoksia oli erittäin vähän eikä ohjelman logiikkaa muutettu, joten vaikka versio oli ollut vain yhden päivän sisäisesti testauksessa, se voitiin laittaa beta-käyttäjien saataville. Beta-version saatavuuden viestittäminen tapahtuu Google+-ryhmän avulla. Lisäsin ryhmän viestiketjuun viestin uudesta versiosta ja pyysin käyttäjiä kokeilemaan sitä seuraavana yönä. Perjantaipäivän tavoitteena oli kerätä palautetta ja tarkistaa version tilanne, jotta sen saisi mahdollisimman nopeasti markkinoille.

Aamupäivällä tuotetiimin daily stand up –kokouksen jälkeen tarkistin Android-version tilanteen Mixpanel-ohjelman avulla. Otin selvää, kuinka moni oli käyttänyt sovellusta mittaukseen, niin että varsinaisia unituloksia oli saatu. Beta-käyttäjiä on huomattavasti vähemmän kuin iOS-versioissa. Koska käyttäjiä oli kymmeniä, pystyin toteamaan, ettei versiossa ollut rikkoontunut mikään. Google+-ryhmään ei ollut tullut mitään palautetta käyttäjiltä, joten myöhemmin päivällä julkaisin beta-version tuotantoon. Prosessi on Android-versioissa huomattavasti nopeampi kuin iOS-versiossa, jossa Apple kanssa joutuu odottamaan viikon, kun taas Google Play Store-kauppaan beta-version saa kahdessa tunnissa. Päivitys oli tärkeä Aasian markkinoille, sillä siinä oli tärkeän kielilisäys, jonka avulla käyttäjiä saadaan enemmän.

Yrityksessä on aloittanut uusi työntekijä asiakastuessa ja minun tehtäväni oli varmistaa, että hänellä on mahdollisuus kirjoittaa virheraportteja Fogbugz-ohjelmaan. Päivän tavoitteena oli lisätä työntekijä ohjelmaan ja näyttää hänelle kuinka raportti kirjoitetaan. Muiden kiireiden vuoksi en ehtinyt perehdyttämään häntä, mutta sovimme että ensi viikolla käydään läpi, mitä hyvään raporttiin kuuluu kirjata.

Päivän aikana jatkoin TestLodge-ohjelman käyttöönottoa siirtämällä uusia testitapauksia siihen Google Sheets –tiedostoista. Kokeneempi kehittäjä suositteli kokeilemaan toista testihallintatyökalua rinnakkain TestLodgen kanssa. Minusta idea oli hyvä, sillä siten saan laajemman näkemyksen työkaluista, joten päätin ottaa maanantaina kokeiluun toisen sovelluksen. Koska olin siirtänyt ohjelmaan muutaman testitapauksen, päivän lopussa lähdin kirjoittamaan testisuunnitelmaa tulevalle iOS-versiolle. Uusi versio julkaistaan ensi tiistaina

ja sitä seuraava virheiden korjausversio kahden viikon päästä. Testisuunnitelmassa pohdin mitä kaikkea minun tulisi testata ja millä työkaluilla. En ole aiemmin kirjoittanut suunnitelmaa, mutta onneksi ohjelmassa oli valmiita alaotsikoita, joiden avulla pystyin täyttämään tarvittavat kohdat. Suunnitelman tulisi heijastaa tulevaa versiota niin, että testausta tehdään vain sovelluksen niihin osa-alueisiin, joihin tehdään muutoksia. Jatkan suunnitelman tekoa maanantaina.

Viikkoanalyysi

Viikon tavoitteeksi asetin itselleni uuden Android-version luomisen ja tulevan iOS-version testaamisen. Viikon lopuksi sain sopivasti Android-version viralliseen Google Play Store –kauppaan, johon olen tyytyväinen. Versiota on kysytty jo pidemmän aikaa, joten sen olisi voinut tehdä jo aikaisemmin, sillä muutokset olivat pieniä ja prosessi on suhteellisen nopea. Pääsin viikolla perehtymään uuteen TestLodge-ohjelmaan ja tarkoituksena on ottaa myös vertailtavaksi toinen testihallintatyökalu, jotta voin päättää kumman ottaa käyttöön. En ole varsinaisesti vielä käyttänyt ohjelmaa, mutta minulla on optimistinen asenne työkalua kohtaan, koska sen avulla voin parantaa testausprosessia.

TestLodge on verkkopohjainen testihallintatyökalu, jolla voidaan hallinnoida eri projekteihin liittyviä testaus vaatimuksia, suunnitelmia, tapauksia ja tuloksia. Jokaisella tiimin jäsenellä on mahdollisuus käyttää sitä mistä tahansa ja sitä on helppo käyttää yksinkertaisen käyttöliittymän takia. (Software Testing Help 2015.) Tarkoituksen on ollut ottaa ohjelma käyttöön kollegan ehdotettua sitä muutama viikko sitten. Tällä viikolla loin yritykselle viralliset tunnukset ja pääsin hyvin alkuun. Jotta ohjelmaa voi käyttää tehokkaasti, on ymmärrettävä mitä kaikkea sillä voi tehdä ja miten. Alustavasti olen ajatellut käyttää ohjelmaa testitapausten ajamista varten. Tähän liittyy myös testisuunnitelmien kirjoittaminen ennen varsinaista manuaalista testausta.

Software Testing Helpin (2015) artikkelissa kuvataan kahdeksan äskettäistä parannusta TestLodge-ohjelmaan käyttäjäpalautteiden perusteella:

1. Integrointi erilaisiin virheseurantajärjestelmiin
2. Vaatimusseuranta
3. Kuvien liittäminen
4. Toiminnan kuvaajat
5. Edistyksellisen testiajon luonti
6. Sisällön kopiointi
7. Versiointi
8. Uudelleenkäytettävät testitapaukset

Yrityksessä on käytössä Fogbugz-ohjelma, joka on yksi listatuista virheseurantajärjestelmistä, jonka voi integroida ohjelmaan. Kun jokin testitapaus ei mene läpi, integraatio luo Fogbugz-ohjelmaan suoraan virheraportin. Se kuulostaa järkevältä ja en näe syytä miksi sitä ei kannataisi tehdä. Mikäli johonkin testitapauksen halutaan visuaalisesti esittää lopputulos, kuvan lisääminen kertoo enemmän kuin teksti. Ajan myötä testitapaukset muuttavat ominaisuuksien ja vaatimusten mukana, joten on hyvä, että testitapausten eri versioita voidaan säilyttää. Edellisessä ratkaisussa Google Sheet –tiedostot säilytettiin pilvessä ja niitä voi kuka tahansa muokata ilman, että edellinen versio säilyy. Ohjelman pääominaisuuksien lisäksi ohjelma tarjoaa lukuisia muita ominaisuuksia, joita voin tarvittaessa ottaa käyttöön.

Testisuunnitelma on joukko ajatuksia, jotka ohjaavat tulevaa testausprosessia (Pettichord 2002, 233). Aiemmin yrityksessä, esimerkiksi viimeisimmässä julkaisussa, olen ajanut funktionaalisia testejä lähinnä omien päätelmien mukaan muutoksien perusteella. Alustavassa yrityksen laadunvarmistusprosessissa, jonka loin työsuhteen alussa, ajoin testejä uusimpien muutosten perusteella. Olen edelleen sitä mieltä, että tämä on järkevä tapa työskennellä, mutta testisuunnitelman laatiminen etukäteen tekisi lopullisesta testauksesta systemaattisempaa, varsinkin kun tulevat korjaukset sovellukseen tiedetään jo etukäteen.

Koska viimehetken muutokset ovat mahdollisia, tarkoituksena on suunnitella prosessia, jossa tämän tyyppiset muutokset eivät kaada koko testisuunnitelmaa (Pettichord 2002, 168). Pettichord (2002, 169) ehdottaa erilaisia ratkaisuja, joilla testisuunnitelmasta saadaan joustava. Yleisen tason testitapauksia tulisi myös tehdä. Ne voidaan aina suorittaa, ohjelman muuttuessakin. Sen sijaan, että tehtäisiin pitkiä testausdokumentteja, niistä tulisi tehdä mahdollisimman ytimekkäitä. Testitapauksia ei tulisi yhdistää käyttöliittymään, ellei testi ole tarkoitettu käyttöliittymän testaamiseen, koska se voi muuttua usein. Kokeneempi kehittäjä oli aikaisemmin maininnut, että testitapauksia ei tulisi kirjoittaa yksityiskohtaisesti, sillä muuten niitä joutuu ylläpitämään jatkuvasti.

Testisuunnitelmassa käyn läpi testauksen kohteen, testattavat ja ei-testattavat osat, hyväksymiskriteerit, aikataulun, työkalut ja riskit. Suunnitelman yksi tavoite on viestittää muulle tuotekehitystiimille, mitä testaaja on tekemässä (Patton 2001, 252). Toinen tavoite on tehdä omasta työskentelystä systemaattisempaa ja aloittaa se ajoissa – näin virheet löydetään ajoissa.

Ensi viikolla jatkan suunnitelman laatimista ja toivottavasti saan sen jo valmiiksi alkuviikosta. Seuraavan viikon tavoitteena on myös kokeilla toista testihallintatyökalua, jotta voin

vertailla sitä TestLodge-ohjelmaan. Virallinen julkaisupäivä uudelle iOS-versiolla on tiistaina, joten silloin on minun vastuullani olla tarkkana ja seurata julkaisua eri työkaluilla. Mikäli virheitä esiintyy pystyn ajoissa kertomaan siitä kehittäjille. Vaikka testaisin sovelluksen huolellisesti, voi aina olla mahdollista, että joku käyttäjistä käyttää sitä eri tavalla ja löytää jotain, joka on jäänyt minulta huomaamatta.

3.6 Seurantaviikko 41

Maanantai 5.10.2015

Viikoittaisessa yrityksen yhteisessä stand up –kokouksessa kerroin uudesta Android-versiosta, jonka julkaisin perjantaina Google Play Store –kauppaan. Tämän viikon tavoite on saada uusi iOS-versio julkaistua mahdollisimman pian ja aloittaa aikaiset testaukset sitä seuraavaksi tulevalle versiolle.

Päivän aamukokousten jälkeen istuimme yhdessä esimieheni kanssa katsomaan julkaistavan iOS-version tilannetta. Perjantaina myöhään illalla Apple oli hyväksynyt version sellaiseen kuntoon, että yrityksen vastuulla on julkaista se. Tarkistimme kaikki tiedot liittyen julkaisuun, kuten kuvakaappaukset ja sovelluskuvaukset eri kielille. Pienien päivitysten jälkeen julkaisin version App Store –kauppaan. Sovellus ei näy välittömästi kaupassa, mutta tunnin sisällä se oli jo käyttäjien saatavilla. Virallinen julkaisupäivä sovellukselle on tiistaina 6.11., jolloin kaikki markkinointimateriaali lähetetään käyttäjille ja medialle.

Koska olin saanut jo perjantaina TestLodge-ohjelman osittain toimintakuntoon, kutsuin muut tuotetiimin jäsenet siihen käyttäjiksi. Ohjelmassa on mahdollisuus rajoittaa oikeuksia rooleihin, mutta en nähnyt tässä vaiheessa sitä tarpeelliseksi. Etsin myös toisen samankaltaisen ohjelman, jota voin kokeilla TestLodge-ohjelman rinnalla. Testrail on testihallintaohjelma, jossa idea on sama kuin TestLodge-ohjelmassa. Loin tunnukset ja tutustuin pikaisesti ohjelman käyttöliittymään, mutta en ehtinyt vielä perehtymään siihen tarkemmin. Koska olen päässyt toiseen ohjelmaan melko hyvin sisälle, käytän ensisijaisesti sitä seuraavissa testisessioissa. Älykellosovellukseen tehdään parhaillaan korjauksia ja tehtäväni oli tänään ajaa TestLodge-ohjelmalla kaikki siihen liittyvät testitapaukset. Päivän aikana keskustelimme kehittäjien kanssa uusista esille tulleista testitapauksista, joten sain listaan pari uutta tapausta. Tavoitteena oli saada tulokset suoritetuista testeistä, jotka kertovat visuaalisesti muille kehittäjille enemmän kuin taulukossa esitettynä. Tarkoituksena on kuvastaa punaisena kaikki kohdat, jotka vaativat korjausta ja lopputestauksessa nämä kohdat voidaan ajaa uudestaan ja muuttaa ”vihreiksi”.

Päivän lopuksi sain vielä tehtäväksi testata muutosta, jossa ohjelmakoodia tehtiin uudelleen. Testien tarkoituksena oli todistaa, että vaikka koodin tehtiin muutoksia, sovellus toimii kuitenkin moitteettomasti. En ehtinyt tekemään testejä loppuun, joten keskiviikon tavoitteena on luoda, ajaa ja dokumentoida kaikki tekemäni testit sovellukselle.

Keskiviikko 7.10.2015

Koska tiistaina oli suuri julkaisu uudelle iOS-versiolle, asetin aamupäivällä itselleni tehtäväksi tarkistaa version tilanteen Crittercism-ohjelmalla. Uusia käyttäjiä oli luotu huomattavan suuri määrä ja sen huomasi myös ohjelmasta, sillä uusia virhetilanteita oli ilmestynyt. Kuten olen aikaisemmin maininnut, että vaikka kuinka hyvin testaan, eri käyttäjät käyttävät sovellusta erilaisilla laiteitta ja eri tavalla, joten on luonnollista, että uusia ennakoimattomia tilanteita esiintyy sovelluksessa. Kaikki uudet tilanteet olivat suurin osa yksittäisiä tapauksia tai pienellä osalla käyttäjistä, joten mitään varsinaisia toimenpiteitä en toistaiseksi tehnyt. Katson tilanteen uudestaan seuraavana työpäivänä ja käyttäjäpalautteiden mukaan.

Maanantaina aloitin koodimuutoksen testaamisen, joten päivällä dokumentoin testin, jolla varmistin, että ohjelma toimii muutoskohdassa niin kuin sen pitäisi. Tällaista erityistestitapausta en tuonut testihallintaohjelmaan, vaan kirjoitin sen yksinkertaisesti Excel-tiedostoon. Mielestäni joko testihallintaohjelmassa on kohta erityisille tapauksille tai ne jätetään sieltä kokonaan pois, jolloin ohjelmassa olisi vain yleisiä testitapauksia liittyen johonkin ominaisuuteen.

Suoritin ensimmäiset testitapaukset TestLodge-ohjelmalla maanantaina, josta kirjoitin lopulta testiraportin. Vaikka lopputulokset kertovat miten testit ovat menneet, mielestäni on hyvä kirjoittaa myös suullinen kuvaus testisessiosta. Olen myös viikon aikana tutustunut Testrail nimiseen ohjelmaan, joka tarjoaa käytännössä samat ominaisuudet kuin TestLodge, mutta enemmän mahdollisuuksia. Koska tiistaina kehittäjät tekivät uuden testiversion sovelluksesta, joka sisälsi korjauksia virheisiin, päätin testata tämän version Testrail-ohjelmalla. Näin sain laajemman käsityksen toisesta ohjelmasta.

Vaikka TestLodge on käyttöliittymältään yksinkertainen, Testrail tarjoaa enemmän mahdollisuuksia muokata omaa testausprosessia. Esimerkiksi Testrail testitapauskokoelmiin voi luoda alitapausryhmiä, johon tapaukset voidaan sijoittaa. Tämä mahdollistaa testien järkevän ryhmittelyn, joka lopulta helpottaa ja tehostaa testausprosessia, sillä testatessa voidaan suorittaa osa pääominaisuuksien aliryhmien testitapauksista. Testrail-ohjelmasta sai ammattimaisemman kuvan, kuin TestLodgesta juuri muokattavuuden takia. Opin, että on hyvä vertailla työkaluja ja työtapoja, jotta saan itselleni paremman tavan työskennellä.

Lopulta kirjoitin testisessiosta raportin kehittäjille, jotta he saavat tiedon mikäli virheitä ei enää esiinny. Toistaiseksi kirjoitan raporttini tavalliseen tekstidokumenttiin ja yksi tapa olisi hyödyntää ohjelmaa järkevän raportin generoimiseen. En ole virallisesti valinnut Testrailiä yrityksen tulevaksi testihallintaohjelmaksi, mutta se on vahva ehdokas. Perjantaina jatkan ohjelmaan perehtymistä ja älykellon manuaalista testausta.

Perjantai 9.10.2015

Päivän ensimmäinen tehtävä oli tarkistaa tiistaina virallisesti julkaistun version tilanne. Keskiviikkona ei tullut mitään vakaavaa vastaan ja tänään tarkistettuani tilanteen päädyin lopputulokseen: vaikka erilaisia virhetilanteita oli jälleen ilmestynyt, jokainen niistä oli tapahtunut alle kymmenelle käyttäjälle. Aktiivisia käyttäjiä on useita tuhansia, joten en toistaiseksi perehtynyt virheisiin muiden kiireellisten työtehtävien takia.

Päivän tavoite oli saada seuraava versio Applen katselmoitavaksi, joten tehtäväni oli testata kehittäjien tekemiä korjauksia. Yleensä ennen versiohallinnan päähaaraan liittämistä, kehittäjät testaavat omat korjauksensa, mutta koska niitä ei ollut vielä liitetty päähaaraan eikä minulla ollut kokonaista versiota testattavana, testasin eri korjauksia haaroittain. Yrityksen periaatteena on, että jokaisessa versionhallinnan haarassa tehdään yhtä asiaa, joten luonnollisesti jokainen korjaus oli omalla haarallaan.

Testauksen aikana tuli ilmi useita käyttötilanteita, joissa sovellus ei toiminut kuten se on määritetty. Varsinkin älykellosovelluksessa tuli vastaan yhteysongelmia, kun sovellusta käytti eri tavoin. Testaajana tehtäväni on myös pakottaa erilaisia virhetilanteita erilaisin keinoin, esimerkiksi katkaisemalla yhteys kelloon. Kun uusia käyttötapauksia tuli vastaan, lisäsin niistä Testrail-ohjelmaan testitapauksia.

Aamulla sain palautetta Testrail-ohjelmasta, johon olen viime aikoina merkinnyt testitulokseni. Oman mielipiteeni lisäksi muut kehittäjät olivat sitä mieltä, että Testrail on selkeämpi kuin TestLodge. Ohjelmassa olen laittanut tavoitteeksi uuden iOS-version julkaisun, johon olen liittänyt testitapauksia, joita tulisi suorittaa julkaisuun mennessä. Kehittäjät näkevät helposti yleisen kuvauksen, jossa on kuvaajaan merkitty kuinka moni testeistä on läpäisty. Kun sivua katsoo tarkemmin, siinä kuvataan mitkä testeistä ovat ja eivät ole läpäisty. Testitulokset ovat siten dynaamisia, että niitä päivitetään sitä mukaa kun tilanne muuttuu.

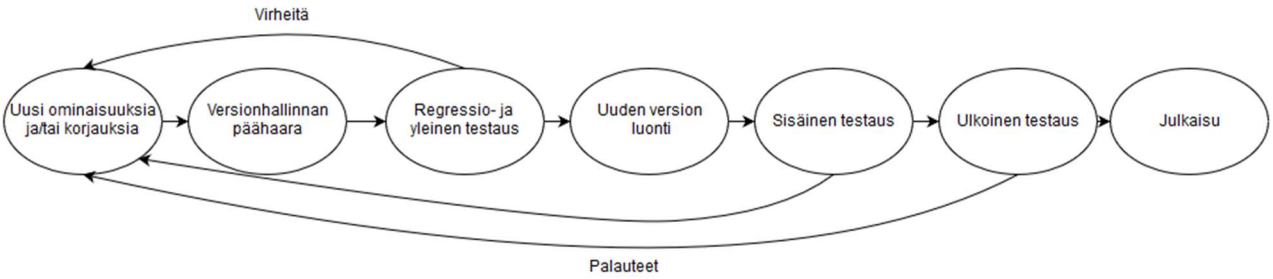
Päivän lopuksi loin uuden version, johon ei saatu mukaan kaikki haluttuja korjauksia, joten versiota ei laiteta vielä tänään Applen katselmoitavaksi. Koska korjauksia on kuitenkin toteutettu osa, lopullisen testaukseen jälkeen versio voidaan julkaista sisäiseen ja lopulta

ulkoiseen testaukseen, jotta saadaan beta-käyttäjiltä nopeasti palautetta. Ensi viikolla on tarkoituksena saada heti alkuviikosta lopullinen versio Applen katselmoitavaksi, jotta se saadaan App Store –kauppaan.

Viikkoanalyysi

Viikolle oli selkeät tavoitteet, johon kuului uuden iOS-version julkaisu ja sitä seuraavan version asettaminen Applen katselmoitavaksi, jotta sekin saadaan seuraavalla viikolla App Store –kauppaan. Julkaisu tehtiin jo maanantaina, vaikka virallinen julkaisupäivä oli tiistaina ja seuraavaa versiota ei toistaiseksi saatu vielä katselmointiin, mutta se saatiin sisäiseen testaukseen ja ensi maanantaina se saadaan Applelle.

Tiukassa aikataulussa olen oppinut, kuinka toimia kovan paineen alla ja miten saada työtehtävät tehtyä systemaattisesti. Olen yrityksessä oppinut syksyn aikana kuinka mobiiliratkaisuja ja uusia versioita julkaistaan. Vaikka julkaisut eivät ole aina menneet täysin suunnitelman mukaisesti, toistamalla prosessia mahdollisimman monta kertaa, pystyn jatkossa työskentelemään tehokkaammin. Jos prosessiin ottaa systemaattisen lähestymistavan ja kehittää taitojaan, paineen alla työskentely helpottuu (Haden 2012).



Kuva 4. Yrityksen mobiiliratkaisun julkaisuprosessi

Kun kehittäjät ovat toteuttaneet ja testanneet uusia ominaisuuksia ja korjauksia, ne liitetään projektin versionhallinnan päähaaraan. Jossakin tapauksissa olen osallistunut toteutusvaiheessa ominaisuuden testaukseen, mutta niissä tapauksissa testaus on hyvin rajattu yksittäisen kohtaan. Yleensä kehittäjä suorittaa sopivan määrän testejä, jolla todetaan ominaisuus toimivaksi (Pettichord 2002, 163). Tällä hetkellä yrityksessä näitä testejä ei aina varsinaisesti dokumentoida niin, että niitä olisi mahdollista suorittaa uudelleen. Ensimmäistä kertaa tällä viikolla kirjoitin yksityiskohtaisen testin, jonka tarkoituksena oli varmentaa iOS-sovellukseen tehtyä muutosta. Jatkossa olisi viisasta pitää jonkin asteista dokumentaatiotasoa tilannekohtaisesti, mutta uskon että tärkeintä tällä hetkellä on pitää huolta siitä, että jokaiselle sovelluksen pääosalle on funktionaaliset testit.

Ennen uuden version luontia, tehtäväni on suorittaa tarvittavat regressiotestit ja varmistaa, että sovellus toimii yleisellä tasolla. Varsinkin tiukalla aikataululla olen huomannut, että kaikkea ei ehdi testata. Tämän takia on tärkeää osallistua ominaisuuksien testaamiseen kehityksen alkuvaiheessa. Pettichordin (2002, 148) mukaan kehittäjät haluavat puhua työstään ja testaajan tehtävänä on kysyä heiltä kysymyksiä liittyen heidän työhönsä, jotta hän saa paremman kuvan testattavasta ominaisuudesta. Minun selkeä etuni yrityksessä on, että työskentelin aikaisemmin harjoittelevana kehittäjänä, joten pystyn yleensä teknisellä tasolla keskustelemaan esimerkiksi virheeseen liittyvästä ratkaisusta. Ohjelmointikielen tunteminen auttaa keskusteluissa. Osallistuin ennen kehittäjänä pääosin iOS-sovelluksen kehittämiseen, joten koodin lukeminen auttaa minua, eikä minun edes tarvitse aina kysyä kehittäjältä mitä muutoksia on tehty.

Kun versioehdokas on hyväksytty testauksella, siitä tehdään virallinen numeroitu kokonaisuus joko Xcode- tai Eclipse-kehitysympäristöllä ja se voidaan julkaista sisäiseen testaukseen. Jos versio on testattu hyvin, tulee sisäisestä testauksesta harvoin sellaista palautetta, että sitä ei voitaisi julkaista ulkoiseen testaukseen. Käyttäjiä on enemmän ulkoisessa testauksessa, joten palautetta on kerättävä sähköpostilla. Kun versio on todettu julkaisukelpoiseksi se julkaistaan App Store – tai Google Play Store –kauppaan.

Yrityksessä ei ole neuvoteltu miten usein versioita tehdään, mutta viikoittain niitä voi tulla useampia. Jos versioita tulee liian tiheään tahtiin, testaajan on vaikea hallinnoida niitä (Pettichord 2002, 161). Olen huomannut, että testauksesta ei tule niin systemaattista, jos sitä tekee kiireessä. Viime aikoina kun julkaisuille on asetettu tarkat päivämäärät, ymmärrän että jostain on aina karsittava, jotta saadaan julkaisu sovitusti markkinoille.

Perjantaina, kun uusi versio oli tarkoitus luoda, työskentelin läheisesti kehittäjän kanssa, jolla oli keskeneräisiä korjauksia. Pettichord (2002, 175) suosittelee paritestausta, jolloin testauksesta tulee sujuvampaa. Testasin ominaisuutta, jolloin pystyin välittömästi kertomaan mikä siinä oli vielä vikana. Kehityksestä tuli näin dynaamisempaa ja kehittäjän työstä helpompaa, sillä hänellä oli hieman vaikeuksia keksiä sopivaa ratkaisua käyttötilanteille.

Ensi viikon maanantaina on vielä aikaa tehdä viimeisiä testauksia julkaisuehdokkaalle, mutta päivän lopuksi se on saatavana aikataulun mukaisesti katselmointiin. Tehtäväni silloin on kerätä palautetta sisäisestä testauksesta ja varmistaa vielä sovelluksen yleinen toimivuus. Koska aikaa ei ole, versio laitetaan poikkeuksellisesti maanantaina myös ulkoiseen testaukseen. Sovellukset ovat yleensä Applella noin viikon katselmoitavana, joten

jos ulkoisessa testauksessa tulee jotain kriittistä esille, versiota voidaan vielä viikon aikana korjata ennen julkaisua.

3.7 Seurantaviikko 42

Maanantai 12.10.2015

Sisäisessä testauksessa oleva uusi iOS-versio laitettiin viikonloppuna ulkoiseen testaukseen, jotta testaajilla olisi aikaa käyttää sitä. Sovellus on käynnissä yön aikana, joten olisi ollut liian myöhäistä laittaa se ulkoiseen testaukseen tänään. Päivän tavoite oli määrittää oliko sovellus julkaisukelpoinen. Viikoittaisessa kokouksessa mainitsin tämän päivän tavoitteesta ja mitä tulevina päivinä tulisin tekemään. Tuotetiimissä alettaisiin jo miettimään mitä seuraavaan iOS-versioon tulee.

Käyttäjä oli julkaissut beta-testaajien Facebook-ryhmään kommentin, jossa hän sanoi, että uusin beta-versio ei toiminut uusimmassa iOS-käyttöjärjestelmässä. Käyttäjä vakuutti, että sovellus lopetti toimintansa viiden minuutin sisällä ohjelman käynnistyttyä. Koska iOS-sovelluksessa on Crittercism-integraatio, pystyin helposti käyttäjätietojen perusteella katsomaan, mitä sovelluksessa on tapahtunut. Ohjelmasta löysin käyttäjän tapahtumat, mutta käyttäjälle ei oltu kirjattu virhetilanteita. Tänään en ehtinyt, mutta kysyn seuraavana päivänä käyttäjältä tarkempaa kuvausta tapahtuneesta.

Vaikka uusinta iOS-versiota ei oltu julkaistu, pidimme tuotetiimin kesken suunnittelukokouksen siitä, mitä seuraavaan versioon tehdään. Kaikkea ei ehditty toteuttamaan tulevaan versioon, joka menee tänään Applen katselmoitavaksi, joten ne siirtyvät loogisesti sitä seuraavaan versioon. Valmistauduin kokoukseen ottamalla selvää Crittercism-ohjelmasta; mitkä ovat suurimmat virhetilanteet App Store –kaupassa olevassa versiossa. Vaikka isoimmassa virheessä oli enintään noin kymmenen käyttäjää, ne tulisi korjata. Keräsin kolme suurinta virhettä ja esitin ne koko tuotetiimille ja lopulta niistä tehtiin työtehtävät kehittäjille.

Koska julkaisu ulkoiseen testaukseen tehtiin iltapäivällä, kehittäjillä oli päivällä aikaa tehdä viimeiset muutokset. Testaamiseen täytyy varata aikaa ja korjauksia voidaan jatkaa loputtomasti. Kun sovittiin, että kehittäminen loppuu, loin uuden version. Käytin loppupäivän testaamiseen, johon kuului yleinen testaus eri laitteilla ja käyttöjärjestelmillä sekä älykelloosuuden funktionaalinen testaaminen. Uusimman version testaamiseen olen käyttänyt viime aikoina Testrail-ohjelmaa. Oli hyvä, että kokeilin kahta eri työkalua, sillä valitsin niistä selkeämmän ja monipuolisemman ja tulen jatkossa käyttämään sitä. Päivän aikana

luotu versio laitettiin suoraan Apple katselmoitavaksi, mutta koska versio on yleensä siellä viikon, se on beta-käyttäjien saatavilla. He tulevat viikon aikana käyttämään sitä ja heiltä tullaan vielä keräämään erikseen palautetta, jolla vielä todetaan versio julkaisukelpoiseksi.

Keskiviikko 14.10.2015

Päivä aloitettiin perinteisellä kahden viikon välein pidettävällä bi-weekly-kokouksella. Tuotetiimissä kerrottiin lähitulevaisuuden suunnitelmista, johon kuului uuden iOS- ja Android-version julkaiseminen. Yritys on pitkään keskittynyt Apple-alustan kehittämiseen ja Android on jäänyt sivurooliin. Ensi viikosta lähtien yrityksessä aloittaa ulkoinen konsultti, joka tulee työskentelemään uuden Android-version kanssa. Isoin ongelma tällä hetkellä on, että esimerkiksi ulkoasultaan sovellusversiot eri alustoille ovat hyvin erilaiset.

Päivän tavoitteena oli tehdä viimeisiä testejä uudelle iOS-versiolle, sillä se on Applen katselmoitavana, joten sen aikana on hyvä vielä tarkemmin varmistaa sovelluksen toimivuus laajemmalla testauksella. Ennen ulkoiseen testaukseen julkaisua testasin sovellusta eri versioilla yleisellä tasolla. Tänään oli aikaa valita oikeat testitapaukset ja käydä niitä läpi Testrail-ohjelmalla.

Lounaan yhteydessä pidin koko yritykselle lyhyen esittelyn Testrail-ohjelmasta ja sen ominaisuuksista. Esittelin kuinka olen ohjelmaa käyttänyt ja miten se auttaa yrityksen tuotteen laadunvarmistuksessa. Vaikka ohjelma voi olla esimerkiksi markkinointitiimille osittain hyvin tekninen, ohjelma antaa yleisen kuvauksen tuotteen nykytilanteesta. Ohjelmaa tarjoaa lukuisia muita ominaisuuksia, mutta yritys ei välttämättä hyödy osasta, joten jatkossa aion vielä tutustua näihin ominaisuuksiin ja valita hyödylliset ominaisuudet, jotta voin parantaa testausprosessin tehokkuutta entisestään. Esittelyn lopuksi käytiin keskustelua, jossa moni kysyi ohjelmasta tarkemmin ja pyrin vastaamaan heille oman tietämykseni mukaan.

Päivän lopuksi lähetin kaikille uuden iOS-version testaajille sähköpostia, jossa kysyin kokemuksista ja ongelmista, joita joku on mahdollisesti kohdannut. Sähköpostin lähettäminen on tarpeellinen vaihe, jotta kaikki palautteet voidaan huomioida virallista julkaisua tehdessä. Perjantaina luen kaikki palautteet ja viestitän tarvittaessa tuotetiimille esiin tulleista ongelmista.

Perjantai 16.10.2015

Päivän tavoitteena oli saada uusi julkaistava iOS-versio sellaiseen kuntoon, että kun Apple on katselmoinut sen, julkaisu voidaan tehdä lähes saman tien. Tähän kuului lopputestien ja raportin tekeminen sekä julkaisuun vaadittavien tietojen täyttäminen.

Aamulla ennen iOS-version viimeistelyyn siirtymistä, katsoin beta-käyttäjien lähettämät palautesähköpostit läpi. Lähetin sähköpostit keskiviikkona, joten tänään oli mielestäni hyvä aika katsoa ne vaikka palautteita voi tulla myöhemminkin. Yleensä kaikki, jotka haluavat antaa palautetta, antavat sitä melko nopeasti. Suurimmalla osalla käyttäjistä ei ollut mitään erikoista sanottavaa, mutta muutamat heistä kuvailivat tilanteita, joissa sovellus ei toiminut. Varsinkin yksi palaute oli erittäin mielenkiintoinen. Sen tuloksena päädyin itse kokeilemaan tapausta omilla laitteilla ja havaitsin saman virheen, joten loin uuden virheraportin Fogbugz-ohjelmaan. Virhe estää oleellisen osan sovelluksesta toimimasta, joten ehdotin sen korjaamista seuraavissa versioissa. En ollut itse ajatellut aikaisemmin käyttäjän kuvaamaa tilannetta, mikä todistaa sen, että beta-testauksesta on todella hyötyä.

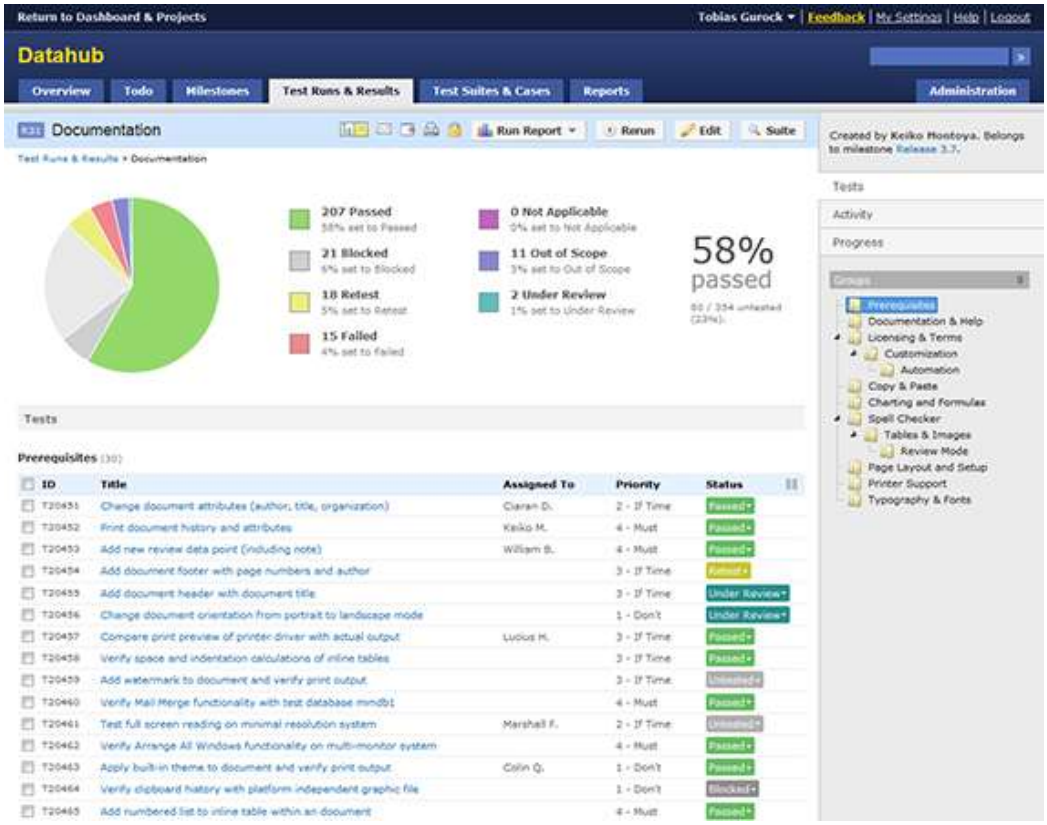
Sähköpostien läpikäymisen jälkeen suoritin loput keskiviikkona jääneistä testeistä. Mitään vakaava ei testien aikana esiintynyt, mutta pieni osa testeistä ei läpäisseet niille asetettuja vaatimuksia. Monet niistä ovat ennalta tiedettyjä virheitä, joita on suunnitteilla korjata, mutta ne on hyvä merkitä Testrail-ohjelmaan kehittäjien ja muiden nähtäväksi. Kun testit oli suoritettu, loin ohjelman avulla raportin, jossa saadaan kaikki uuteen versioon liittyvät tiedot. Ohjelma mahdollistaa visuaalisen kuvaustavan, jossa voidaan helposti nähdä testien yleiskuvaus, kun taas Excel-tiedostoista ei varsinaisesti saanut yhdellä vilkaisulla selkeää kuvaa esimerkiksi testien määrästä. Raportin perusteella voidaan päättää onko versioehdokas valmis julkaistavaksi.

Ohjelman generoiman raportin lisäksi kirjoitin tekstidokumentin, jossa sanallisesti avasin testisessiota ja kuvailin kaikki virheet, jotka löysin – uudet ja vanhat. Dokumentissa on linkki ohjelman raporttiin, joten ne täydentävät toisiaan niin, että toisessa on kuvaukset ja toisessa on tilastot ja tulokset. Tärkeimpiä tietoja ovat testien läpäisyprosentti ja suoritetut testit ja niiden lukumäärä käyttöjärjestelmittäin.

Päivän lopuksi täytin julkaisuun tarvittavat tiedot, jotka lisätään tiedostoon aina uuden julkaisun yhteydessä. Siihen merkitään Mixpanel-ohjelmasta saadut käyttäjämetriikat, käyttäjäpalautteet, Crittercism-ohjelmassa suurin esiintyvä virhe ja lyhyt kuvaus suoritetuista testeistä. Ensi viikolla versioehdokas palaa katselmoinnista toivon mukaan hyväksyttynä, joten koska kaikki on muuten valmista, julkaisu voidaan tehdä heti kun se halutaan markkinoille.

Viikkoanalyysi

Viikon tavoite oli saada uusi julkaistava iOS-versio ulkoisesta testauksesta Applen katselmointiin ja varmistaa, että se on julkaisukelpoinen. Se on hyvä saada markkinoille mahdollisimman nopeasti, sillä se sisältää kriittisiä korjauksia sovellukseen – varsinkin älykello-osaan. Olen huomannut, että testaus- ja julkaisuprosessista on tullut selkeämpää, kun sen on muutaman kerran kokenut laadunvarmistajan näkökulmasta. Kun prosessilla on selkeä rakenne, minun on helppo antaa muille tuotetiimin jäsenille yleiskuvaus tuotteen tilanteesta.



Kuva 5. Testitulosten esitys Testrail-ohjelmassa

Viikon aikana selvitin tarkemmin, kuinka käyttää Testrail-ohjelmaa ja sen tarjoamia ominaisuuksia. Virallisesti tällä viikolla valitsin kyseisen ohjelman yrityksen käyttöön, vertailtuani kahta eri ohjelmaa. Kalashnikov (2014) vertailee blogissaan kolmea eri testihallintatyökalua, josta kaksi ovat samat, joita itse kokeilin. Hän kuvailee, että testitulosten esittäminen Testrail-ohjelmassa on kattavampi kuin TestLodge-ohjelmassa, sillä tärkeät tiedot tuodaan selkeämmin esille. Kun itse kokeilin ohjelmia, huomasin että testitapaukset ovat mahdollista ryhmitellä Testrail-ohjelmassa myös aliryhmiin, mikä antaa testaamiselle systemaattisen rakenteen. Testiajoja luodessa en aina halua valita ryhmän kaikkia testejä, esimerkiksi regressiotestejä tehdessä, koska haluan keskittyä sovelluksen erityiseen

osaan. Sekä Testrail että TestLodge sopivat pienien ja keskisuurien projektien testaamiseen ja ne tarjoavat selkeän käyttöliittymän lisäksi lukuisia ominaisuuksia (Kalashnikov 2014).

Kun olin suorittanut kaikki valitsemani testit iOS-versiolle, Testrail esittää tulokset hyvin selkeästi. Pystyin muokkaamaan ohjelman niin, että jokaiseen testitapaukseen pystyin lisäämään kentän, joka kertoi millä versiolle se on suoritettu. Ohjelma mahdollistaa testisessioiden luomisen eri käyttöjärjestelmille ja laitteille, joten käytännössä koska yrityksen tuote tukee kolmea viimeisintä iOS-käyttöjärjestelmää, loin kolme omaa testiajoa, jotka yhdistettiin lopulliseen raporttiin. Ohjelma ottaa huomioon kaikki testitapaukset ja antaa selkeän yleiskuvan, mutta myös hyvän yksityiskohtaisen kuvauksen, esimerkiksi mitkä testeistä eivät läpäisseet iOS8-käyttöjärjestälmällä.

Viikon aikana selvitin, kuinka esitän lopullisen raportin julkaistavasta iOS-versiosta. Valitsin sekä Testrail-ohjelman että itse tuottamat raportit, jotta kehittäjän ja muut tuotetiimin jäsenet saavat mahdollisimman kattavan kuvan version tilanteesta. Raporttia valmistellessa siihen tulisi liittää lista tuotteen korjaamattomista virheistä, sillä näin estytään ikäviltä yllätyksiltä (Pettichord 2002, 187). En kirjoittanut varsinaista listaa versiossa esiintyvistä virheistä, mutta ohjelman ”punaiset” testit puhuvat puolestaan ja kaikki virheet merkitsen Fogbugz-ohjelmaan ja kaikki kriittiset virheet kerron kehittäjille erikseen. Raporttia kirjoittaessa on tärkeää kuvailla testitulosten faktat, eikä ilmaista omia subjektiivisia kantoja, esimerkiksi käyttöliittymäratkaisuista. On vaikeaa arvioida tuotteen markkinakelpoisuus pelkällä testauksella, joten testaaja tulisi kuvailla ainoastaan se minkä hän tietää varmaksi. (Pettichord 2002, 187.) Olen pyrkinyt aina raporteissa antamaan realistisen kuvan niillä tiedoilla, jotka olen saanut testisessioista, jotta kaikki saavat oikean kuvan tuotteesta.

Keskiviikkona aloitin valitsemani lopputestit, jolla hyväksytän version julkaistavaksi omasta puolestani. Työpäivän päättyessä keskeytin session ja jatkoin sen loppuun perjantaina. Työskentelen samassa tilassa muiden tuotetiimin jäsenien kanssa ja taustameteli, esimerkiksi keskustelu voi häiritä keskittymistä. Session aikana testaajien tulisi olla keskittyneitä itse asiaan, jos heitä keskeytetään jatkuvasti, heidän tulee työskennellä lyhyissä jaksoissa. Sessioihin kannattaa varata kuitenkin suojattu aikaväli, jolloin heitä ei häiritä. (Pettichord 2002, 177.) Testauksen aikana olen hoitanut välillä myös muita työtehtäviä, kuten sähköpostiviesteihin vastaamista, joten jatkossa minun tulisi keskittyä ainoastaan testaukseen, jotta loppuraportista tulee kattava ja pystyn huomaamaan kaikki virheet.

Ensi viikolla yrityksessä aloittaa ulkoinen ohjelmistokehittäjäkonsultti, joka tulee työskentelemään Android-version kanssa. Osallistun siten uuden Android-version julkaisuun testauksen kannalta. Tavoitteena ei ole toteuttaa suuria teknisiä uudistuksia, vaan lähinnä käytettävyyden ja käyttöliittymän tekeminen yhtenäiseksi iOS-version kanssa.

3.8 Seurantaviikko 43

Maanantai 19.10.2015

Aamulla ennen yrityksen yhteistä kokousta, tarkistin Applen katselmoinnissa olevan version tilanteen, eikä versio ollut valmis vielä julkaistavaksi. Lisää palautetta beta-käyttäjiltä ei ollut myöskään tullut. Kokouksessa kerroin viime viikon saavutuksesta uuden iOS-version kannalta ja tämän viikon työtehtävistä, joihin liittyy sekä Android- että iOS-sovellusten kanssa työskentelyä.

Päivän tavoitteena oli saada kaikki viimeiset testitapauskokoelmat Testrail-ohjelmaan. Koska viime viikolla kokeilin ohjelmaa, en nähnyt tarpeelliseksi tuoda kaikkia tapauksia ohjelmaan, joten valitsin vain tarvittavat tapaukset testausta varten. Nyt kun ohjelma tulee yrityksen viralliseen käyttöön, on hyvä tuoda kaikki Google Drive –pilvipalvelusta löytyvät tapaukset, jotta jatkossa ne ovat saatavilla kun niitä tarvitaan. Päivitin osan tapauksista vastaamaan sovellusten nykyistä tilannetta. Osa oli vanhempia, joten jätin ne kokonaan pois. Niistä jotka otin ohjelmaan, tein varmuuskopion, jotka jätin pilvipalveluun, siltä varalta, että niille tapahtuisi jotain ohjelmassa.

Koska iOS-versioon on tehty viime aikoina paljon muutoksia – myös business loogisia muutoksia, työntekijä, joka on käyttänyt molempia versioita datan keräämiseen, oli löytänyt viikonloppuna paljon eroavaisuuksia. Osa tiedoista näkyi eri tavalla Android-versiossa, vaikka käytännössä sama data on saatavilla yrityksen pilvipalvelun kautta. Android-version kehitys on jätetty tiedostetusti myöhemmäksi, mutta yrityksen lähitulevaisuuden suunnitelmina on päivittää se. Itse asiassa iltapäivällä saapui ulkoinen konsultti, joka aloitti työskentelynsä yrityksessä. Loppupäivästä vertailin tarkasti molempia versioita käyttäen itse kerättyä dataa. Huomasin testatessa samat virheet, kuin toinen työntekijä, joten tein havainnoista virheraportin kuvakaappausten kanssa. Tilanne, jossa käyttäjä käyttää sekä Android- että iOS-versiota on melko harvinaista, mutta erittäin tärkeää, että data. jota yritys tarjoaa, on yhteneväistä ja että se esitetään mahdollisimman samanlaisella tavalla.

Keskiviikko 21.10.2015

Ensimmäinen tehtävä päivälle oli tarkistaa katselmoinnissa olevan iOS-version tilanne, sillä se oli ollut noin viikon verran siellä. Aamulla se oli kuitenkin valmis julkaistavaksi, joten päivän tavoitteena oli ottaa esimiehen kanssa hetki ja varmistaa kaikki julkaisuun liittyvät asiat läpi. Sovin hänen kanssa ajankohdan iltapäivästä, jolloin keskustelisimme esimerkiksi testauksessa esiin tulleista huomioista.

Aamupäivällä huomasin Crittercism-ohjelmasta tiheästi esiintyviä poikkeustilanteita liittyen viimeisimpään julkaistuun iOS-versioon. Poikkeustilanteet eivät varsinaisesti kaada sovellusta niin, että se sammuisi, vaan ne ovat yleensä käyttäjille huomaamattomia. Kehittäjille se viestii tilanteista, jossa sovellus käyttäytyy eri tavalla kuin sen on tarkoitus. Olin aikaisemmin kirjoittanut suurimmista virheistä raportit ja poikkeukset liittyivät ohjelman mukaan selkeästi näihin virheisiin, joten päivitin virheraportit sisältämään lisätietoa. Tämän avulla kehittäjät saavat laajemman kuvan virheestä, jolloin se on myös helpompi korjata.

Päivän aikana kävin keskustelua asiakastuen kanssa liittyen virhetilanteisiin, joita pieni osa käyttäjistä on kokenut viime aikoina. He olivat saaneet palautteita, joissa sovellus ei ole toiminut uuden iOS9-käyttöjärjestelmän kanssa. Ongelmia on esiintynyt normaalia enemmän, joten näitä tapauksia varten asiakastuki oli pyytänyt tarkempaa kuvausta yhdeltä käyttäjältä. Virhetilan pohjalta laadittiin dokumentti. Tehtävänäni oli sen perusteella kokeilla, josko pystyisin toistamaan virheen luotettavalla tavalla. Virhetilanteita esiintyessä on tärkeää saada käyttäjiltä kaikki mahdolliset tiedot liittyen sovellusversioon ja laitteeseen. Se auttaa virheen tutkimisessa. Käyttäjän käyttötapaukset olivat kuitenkin dokumentin mukaan hyvin tavalliset ja selkeät – ainoa eroavaisuus oli käyttäjän iOS-käyttöjärjestelmässä. Hän käytti uutta beta-versiossa olevaa iOS-käyttöjärjestelmää. Päätin asentaa saman version testilaitteeseen, jolloin pyrin simuloimaan käyttöympäristön, mutta toistaiseksi mitään havaintoja en tehnyt.

Loppupäivän käytin esimiehen kanssa uuden version julkaisuun, jossa kävimme vielä suullisesti version tilanteen läpi ulkoisen testauksen ja funktionaalisten testien kannalta. Päädyimme siihen lopputulokseen, että versio on julkaisukelpoinen eikä se rikkonut mitään aikaisempia ominaisuuksia. Päivän lopuksi versio olikin jo kaikkien käyttäjien saatavilla.

Perjantai 23.10.2015

Koska keskiviikkona julkaistiin uusi iOS-versio App Store –kauppaan, tehtäväni oli tarkistaa version tilanne Crittercism-ohjelmalla, oliko uusia ongelmia syntynyt. Uusia virhetilanteita ei toistaiseksi ole vielä esiintynyt, mutta luonnollisesti virheet, joita ei ole vielä korjattu

esiintyi uudestaan tässäkin versiossa. Olen kirjannut kaikki nämä virheet Fogbugz-järjestelmään ja niiden korjaus on lisätty tehtäväksi seuraavaan julkaisuun. Seuraavan kerran tarkistan tilanteen uudestaan, mutta epäilen, ettei mitään suuria muutoksia tule sovelluksen virheisiin.

Yrityksessä aloitti noin kuukausi sitten uusi työntekijä asiakastuessa ja sovimme torstaina, että pidämme lyhyen session, jossa kävisimme läpi miten ja minne kirjoitetaan sovellukseen liittyviä virheitä Fogbugz-ohjelmaan. Aamupäivällä katsoimme yhdessä miten ohjelma toimii ja minkälainen on hyvä raportti. Tarkoituksena on, että kenellä tahansa yrityksessä on mahdollisuus kirjoittaa raportti – varsinkin asiakastuessa työskentelevällä henkilöillä. Kerroin, että raportissa tulisi olla sovelluksen versio, jossa virhe tapahtui, lyhyt kuvaus virhetilanteesta ja mahdollisesti kohdat, joilla se voidaan toistaa - riippuen virheestä. Aina ei välttämättä ole selkeää tapaa toistaa, mutta korostin että kaikki mahdolliset tiedot liittyen virheeseen tulisi lisätä, esimerkiksi kuvakaappaukset. Hyvän raportin avulla kehittäjällä on kattava lähtökohta, jolla virhe voidaan korjata.

Koska yrityksellä on, Yhdysvaltojen jälkeen, eniten käyttäjiä Suomessa, on päätetty, että sovellus tullaan kääntämään myös suomen kielelle. Käännösyritykselle on lähetetty käännettävät osat, mutta tehtävänäni Androidin kohdalla oli lisätä valmiiksi tarvittavat tiedostot, johon tulisi lopulliset käännökset. Olen aikaisemmin lisännyt Android-versioon uusia kieliä, joten tehtävä annettiin luonnollisesti minulle.

Päivän lopuksi loin uuden iOS-version, joka sisälsi osan suunnitelluista korjauksista. Väliversioita on hyvä luoda, jotta ne saadaan sisäiseen testaukseen, jossa virheet voidaan validoida korjatuksi. Ennen testaukseen lisäämistä, suoritan kaikki tarvittavat regressiotestit, joita en tänään vielä ehtinyt tekemään, joten suoritan ne maanantaina.

Viikkoanalyysi

Viikon päätavoite oli saada Applen katselmoinnissa oleva iOS-versio App Store –kauppaan. Oli tärkeä saada se käyttäjille mahdollisimman nopeasti, sillä se sisältää oleellisia korjauksia juuri julkaistuun älykellosovellukseen. Viime aikoina kehitys on keskittynyt hyvinkin paljon älykellosovellukseen, koska se oli suuri lisäys tuotteelle, mutta tällä viikolla julkaistun version jälkeen kehitys tulee keskittymään enemmän sekä iOS- että Android-sovellukseen. Tällä viikolla yrityksessä aloitti ulkoinen konsultti, joka työskentelee Android-version kanssa. Kehitys on vielä melko alussa, joten testausta ei ole vielä tarvittu.

Orientoin perjantaina uudelle työntekijälle, kuinka yrityksen virheenseurantajärjestelmää käytetään ja kuinka kirjoitetaan hyvä virheraportti. Vaikka kuka tahansa voi kirjoittaa virheraportteja, on tärkeää että minäkin luen ne. Testausvastaavan on hyvä lukea muiden kirjoittamia virheraportteja, koska se auttaa saamaan paremman kuvan tuotteen tilanteesta, henkilökunnan vahvuuksista ja yrityksen sisäisestä kommunikaatiosta (Pettichord 2002, 191). Yritys pitää lyhyitä kokouksia asiakkaiden palautteiden mukana tulleista ongelmista, joten on tärkeää pitää suullinen kommunikaatioyhteys asiakastukeen, sillä muuten jatkossa voi helposti esiintyä duplikaattivirheraportteja tai epäselviä tapauksia. Pettichord (2002, 191) esittää kysymyksiä, joiden avulla saadaan selville kuinka hyvin raportti on kirjoitettu:

- Onko raportti itsessään hyvin kirjoitettu?
- Onko ongelma esitetty yksinkertaisella tavalla?
- Onko jokin kohta jätetty epäselväksi, joka selviäisi lisätestauksella?
- Ovatko testit, jolla virhe löydettiin rutiininomaisia tai oivaltavia?
- Oliko virhe vaikea löytää?
- Mikä on raportin sävy?
- Ymmärsikö ohjelmoija raportin perusteella virheen?

Mielestäni on tärkeää, että ohjelmoija, joka korjaa virheen, ymmärtää ongelman, muuten raportti on itsessään turha. Koska olen yrityksen ainoa testaaja, en voi vaatia muilta joka kerta täydellisiä raportteja, mutta on pyrittävä yksinkertaisuuteen ja varmistettava, että esimerkiksi käyttäjän esittämä tapaus on ohjelmointivirhe sovelluksessa, joka tuo palautteiden keräämisessä omat haasteensa. Pettichordin (2002, 72) mukaan testaajien tulisi myös raportoida käytettävyyteen liittyvistä virheistä, vaikka joidenkin muiden mukaan testaaja ei saisi puuttua käyttöliittymään. Yleensä, jos huomaan epäloogisuutta sovelluksen ulkoasussa, esitän mielipiteeni esimiehelle tai ohjelmistosuunnittelijalle, ja päädymme joko johtopäätökseen, että sitä tulisi muuttaa tai sen kuuluu olla niin kuin se on toteutettu.

Viime aikoina minulle on kerätty virhetilanteita, jotka käyttäjien mukaan liittyvät uuteen iOS9-käyttöjärjestelmään. Olen yrittänyt selvittää näitä tapauksia, mutta yleensä olen tullut siihen johtopäätökseen, että sovellus toimii kuten pitää. Haasteena on kirjoittaa näistä tilanteista virheraportteja, kun selkeästi käyttäjän kuvauksen perusteella sovellus ei toimi. Kun yrittää itse toistaa käyttäjän kuvatuilla kohdilla, sovellus toimii. Yleensä pyydän vielä asiakastuesta tarkempaa kuvausta ja kaikkia mahdollisia lisätietoja käyttöympäristöstä, esimerkiksi laitemalli, sovellusversio ja käyttöjärjestelmä.

Toisena haasteena, jonka olen laadunvarmistajan roolissa huomannut, on virhelistan priorisointi, sillä luonnollisesti kaikkeen ei kehittäjiltä riitä aika. Patton (2001, 283) esittää syitä miksi kaikkia ohjelmointivirheitä ei voida korjata:

1. Aikaa ei ole tarpeeksi
2. ”Se ei ole virhe, vaan ominaisuus”
3. Virhe on liian virhealtis korjattavaksi
4. Korjaaminen ei ole sen arvoista

Täten mielestäni virheiden priorisointi on tärkeää, jotta saan käyttäjille mahdollisimman hyödyllisiä korjauksia, jotka parantavat tuotteen käyttökokemuksia. Tehtäväni testaajana on ottaa esille kaikkein tavallisimmin esiintyvät virheet. Hyvänä kriteerinä olen käyttänyt Crittercism-ohjelmaa, joka selkeästi esittää virheiden esiintymismäärän iOS-sovellusversioittain. Kaikki ohjelman tuottamat raportit eivät aina kerro tarpeeksi esiintyvästä virheestä, joten jos epäselvää virhettä esiintyy vain muutamalle käyttäjälle tuhansista käyttäjistä, korjaaminen ei ole sen arvoista.

Ensi viikolla pyrin jatkamaan iOS-version ja mahdollisesti Android-version kanssa, mikäli kehitystä tehdään niin pitkälle, että on valmiita ominaisuuksia joita pystyn testaamaan. Koska iOS-versioon tehdään edelleen käytettävyysparannuksia ja virheen korjauksia, on tärkeää, että suoritan oikeat regressiotestit. Näin voin varmistaa, että mitään olemassa olevia ominaisuuksia ei rikkoonnu.

3.9 Seurantaviikko 44

Maanantai 26.10.2015

Aamu alkoi yrityksen yhteisellä stand-up-kokouksella, jossa kerroin lähisuunnitelmista liittyen iOS- ja Android-versioon. Tällä viikolla on suunnitelmissa saada App Store -kauppaan uusi iOS-versio, jota on kehitetty siitä lähtien kun sitä edellinen versio laitettiin Applen katselmoitavaksi. Lisäksi Android-versioon kehitetään uusia ominaisuuksia, jonka avulla käyttöliittymä saadaan yhteneväiseksi iOS-version kanssa.

Päivällä asiakastuesta pyydettiin lisäämään uusi käyttäjiä Android beta-testaajiksi. Prosessi ei ole poikkeuksellinen. Käyttäjältä pyydetään sähköpostiosoite, jonka avulla hänet lisätään yrityksen Google+-ryhmään. Kun käyttäjä on kutsuttu siihen, hänelle on oikeus ladata sovelluksen viimeisimmät beta-versiot.

Perjantaina loin väliversion uudelle iOS-versiolle ja tänään tavoitteena oli suorittaa regressiotestejä riippuen siitä, mihin kohtaan sovellusta on tehty muutoksia. Versioon tulee vielä

tällä viikolla lisää korjauksia ja suomenkielinen lokalisointi, mutta on hyvä ajaa testejä jo tässä vaiheessa, että versio saadaan nopeammin sisäisen ja ulkoisen testauksen kautta Applen katselmoitavaksi.

Päivän lopuksi käännösyritykseltä tuli sovelluksen suomenkieliset käännökset, jotka toinen kehittäjä lisäsi iOS-versioon. Tehtäväni oli käydä koko sovellus läpi ja tarkistaa käännökset. Huomasin, että sovelluksen yksi osa oli jäänyt kokonaan kääntämättä ja monet kohdat olivat virheellisesti käännetty. Haasteena käännösyrityksellä oli varmasti kontekstinpuute eli he eivät tienneet minkälaisiin tilanteisiin käännettävät merkkijonot tulevat. Keskiviikkona kirjoitan kaikki virheelliset käännökset ja tilanteet ylös ja kerron kehittäjälle, joka tekee tarvittavat korjaukset.

Keskiviikko 28.10.2015

Tiistaina uusi iOS-versio laitettiin yrityksen sisäiseen testaukseen. Kaikki työntekijät voivat käyttää sitä päivän verran. Tänään oli tavoitteena saada versio Applen katselmointiin ja samalla ulkoiseen testaukseen. Yleensä ennen kuin versio laitetaan katselmoitavaksi, sitä pidetään noin kahden päivän verran ulkoisessa testauksessa, jotta kaikilta suurimmilta virheiltä vältytään. Vversion oli tarkoitus olla jo maanantaina katselmoinnissa.

Tehtäväni oli täydentää uuteen versioon liittyviä tietoja, ottaa uusia kuvakaappauksia sovelluksen suomenkielisestä versiosta ja tehdä keskenjääneet regressiotestit loppuun. Koska käännökset täytyy saada myös Android-versioon myöhemmin, lähetin tänään käännettävät tiedostot käännösyritykselle. Tarkistin regressiotestejä tehdessä samalla suomenkielisen version mahdollisia virheitä.

Koska viime iOS-julkaisusta oli lähiaikoina, olen joka työpäivä tarkistanut Crittercism-ohjelmasta, ettei siinä ole esiintynyt uusia virhetilanteita. Huomasin, että samoja korjaamattomia virheitä esiintyi siinäkin. Päivän lopuksi julkaisin version ulkoiseen testaukseen ja se laitettiin Applen katselmoitavaksi. Katselmoinnin aikana teen vielä lopullisen laajemman hyväksyntätestauksen, jotta voidaan varmistaa, että versio on julkaisukelpoinen.

Perjantai 30.10.2015

Päivän tavoitteena oli varmistaa, että kaikki liittyen seuraavaan iOS- ja Android-version julkaisuun ovat etenemässä oikeaan suuntaan. Applen katselmoitavana on vieläkin keskiviikkona testattu iOS-versio, mutta koska versio laitettiin silloin myös ulkoiseen testauk-

seen, se on myös beta-versioiden katselmoinnissa. Beta-version katselmointi kestää huomattavasti vähemmän kuin App Store katselmoinnissa ja tapana onkin yrityksessä laittaa versio ensin ulkoiseen testaukseen. Tärkeää oli tänään myös tarkistaa nykyisen App Store –kaupassa olevan version tilanne työkaluilla.

Asiakastuki oli edellisenä päivänä lisännyt uusia virheraportteja Fogbugz-järjestelmään, joten kävin tänään läpi raportteja. Koska koko tuotetiimin ovat käyttäjiä järjestelmässä, siirsin osan nykyisistä virheistä kehittäjiä vastuulle suunniteltujen julkaisun mukaan. Kehittäjät eivät aina muista merkata korjauksia järjestelmään, joten kävin läpi niitä ja tarvittaessa päivitin niiden tilat. Yksi virhe liittyi käännösvirheeseen sovelluksessa, joten ehdotin sen kirjoittajalle, että lisää tapauksesta kuvakaappauksen, mutta virheestä ilmoittaneella käyttäjälle ei ollut siitä kuva. Kehitysympäristön avulla pystyn helposti ottamaan mistä vaan sovelluksen osasta kuvan ja tallentamaan sen suoraan tietokoneeni työpöydälle, mikä tekee kuvankaappauksesta helppoa.

Suomen kielen käännöksistä oli unohdettu yksi sovelluksen tärkeä osa, joten tänään lähetin Excel-tiedoston käännösyritykselle. Se sisälsi englannin kielellä käännettävät merkkijonot. Pidimme päivällä pienen kokouksen käyttäjien viimeaikaisista vaikeimmista virhetapauksista tuotetiimin ja asiakastuen kanssa. He ovat raportoineet erilaisia tilanteita, joissa sovellus lakkaa yhtäkkiä toimimasta ja sammuu. Yrityksellä on käytössä Crittercism-työkalu, jonka pitäisi kirjata kyseiset tilanteet. Keskustelimme tarkemmin aiheesta ja yksi työntekijöistä mainitsi, että esimerkiksi Crittercism ei tilastoi käyttöjärjestelmää aiheutuneista virheistä, kuten älypuhelimen muistin loppumisesta johtuvaa sovelluksen sammumista. Kokouksen lopputulos oli, että yksi kehittäjistä alkoi selvittää asiaa, jotta näille yksittäisille käyttäjille saataisiin sovellus tulevaisuudessa toimimaan.

Ensi viikon päätehtävänä on saada uusi iOS-versio App Store –kauppaan lopullisen hyväksyntätestauksen kautta. Viimeistelen suomen kieliset käännökset sekä iOS- että Android-versiolle.

Viikkoanalyysi

Viikon tavoite oli saada uusia iOS- ja Android-versioiden julkaisuja eteenpäin ja iOS-version kohdalla Applen katselmointiin. On tärkeää saada versio katselmointiin, sillä yleensä se kestää noin viikon. Aikataulujen takia tällä viikolla poikkeuksellisesti iOS-versio laitettiin samalla hetkellä sekä Applen että beta-versio katselmointiin, jotta saadaan luvattu suo-

menkielinen tuki sovellukselle. Ensi viikolla versio saadaan alkuviikosta beta-käyttäjille, joten ennen varsinaista julkaisua on mahdollista, että uusia ongelmia tulee esiin. Koska suoritin kaikki vaadittavat regressiotestit, esiintymisen todennäköisyys on pieni.

Tällä viikolla huomasin suomen kielisten käännösten tarkistuksessa, että kaikki käännökset eivät kuullosta itse sovelluksessa hyvältä. Kun käännettävät merkkijonot lähetetään käännösyritykselle ja sitä kautta kääntäjälle, hän ei voi assosioida kääntämiään lauseita ja sanoja kontekstin mukaan. Koska sovellus on käännetty usealle kielelle, mukaan lukien kiina, korea ja japani, yrityksen on vaikea validoida tekstit sovellukselle sopivaksi. Välillä on tullut esimerkiksi Aasian markkinoilla jälleenmyyjiltä korjauksia käännöksiin, joka taas vaikuttaa sovelluksen luotettavuuteen ja uskottavuuteen. Patton (2001, 159) painottaa, että on tärkeää testaajan kyetä luotettavasti validoimaan käännökset. Se tarkoittaa käytännössä sitä, että hänen tulisi olla sujuva käännetyssä kielissä. Tämä on haaste yrityksessä, sillä kukaan ei virallisesti osaa mitään muuta lokalisoituja kieliä kuin englantia ja suomea. Koska sovelluksessa käsitellään uneen liittyviä käsitteitä, huomasin, että esimerkiksi englannin kielen sana ”sleep” voi tarkoittaa sekä verbiä että substantiivia. Kun taas suomen kielessä ne ovat eri sanoja: nukkua ja uni. Tästä johtuen olenkin pohtinut, miten kiinan käännökset ovat toimineet käyttäjille, sillä merkeillä voi olla monia merkityksiä. Yksi vaihtoehto on tarkistuttaa käännökset toiselta kääntäjältä ja mielellään sovelluksen kanssa – näin kääntäjän on helpompi valita oikea sana.

Patton (2001, 164) kehottaa, että ohjelmakoodissa ei tulisi olla kovakoodattuja merkkijonoja, koska muuten niitä ei voi kääntää teknisellä tavalla järkevästi. Onneksi Applen mobiilisovelluksissa pystyy helposti määrittämään tiedosto, josta voidaan johtaa useampi lokalisaatio. Toinen haaste olisi päivämäärät, kellonajat ja aikavyöhykkeet, mutta sovelluksen pystyy helposti asettaa käyttämään puhelimen järjestelmässä määritettyjä asetuksia.

Tällä viikolla huomasin edistystä virheraporttien tekemisessä, sillä asiakastuesta tuli omaaloitteisesti uusia käyttäjiltä saatuja virheitä. Pienetkin virheet hämmentävät käyttäjiä ja vähentävät luottoa koko sovellukseen (Pettichord 2002, 74). Ne voivat sovelluksen kohdalla olla kirjoitus-, laskuvirheitä, virheellisesti esitettyjä kaaviota tai käyttöliittymäelementtien vääränlaisia sommitteluja. Yrityksessä on useammin isompia ja tärkeämpiä ominaisuuksia kehitettävänä, joten pienet korjaukset jäävät yleensä myöhemmäksi, vaikka niiden korjaaminen ei tulisi viedä paljon aikaa kehittäjältä. Minun tehtäväni on tuoda esille selkeästi, mitkä virheet tulisi tuotteen laadun kannalta korjata mahdollisimman pian niin, että kaikki aika ei mene niiden korjaamiseen. Mitä enemmän virheitä kerääntyy sovellukseen, sitä pienempi käyttäjän luotto tuotteeseen tulee olemaan (Pettichord 2002, 75).

Olen huomannut, että asiakkaat raportoivat useasti virhetilanteita, joita on joko erittäin vaikea toistaa. Pettichord (2002, 77) suosittelee kuitenkin, että vaikka virheitä ei saisi toistettua, ne tulisi aina raportoida. Hän korostaa, että virhe tulisi raportoida silloin, kun sitä on oikeasti yritetty toistaa järkevällä tavalla. Olen pyytänyt asiakastuesta mahdollisimman paljon tietoa käyttäjän sovellusympäristöstä, kuten laitteen malli, käyttöjärjestelmä ja sovelluksen versio. Kävimme tällä viikolla keskustelua perjantaina, että esiin tulleet ongelmat voivat johtua myös itse puhelimesta, esimerkiksi tilanne, jossa muisti loppuu. Kaikki Crittercism-työkalun keräämät kaatumiset eivät aina raportoidu ohjelmaan, joten olen kehittämässä tapaa, jolla saisimme tiedon siitä, mikäli sovellus on äkkinäisesti sulkeutunut puhelimen tai muun syyn takia. Ei-toistettavissa virheitä raportoidessa, tulee mainita että sen toistaminen ei ole onnistunut (Pettichord 2002, 78). Jatkossa päivitän raporttini niin, että siinä tulee selkeästi ilmi, ettei virhettä pystytä luotettavalla tavalla toistamaan.

3.10 Seurantaviikko 45

Maanantai 2.11.2015

Viikko alkoi yrityksen perinteisellä yhteisellä stand-up-kokouksella, jossa omalta osaltani kerroin tämän viikon tavoitteista. Ulkoisessa testauksessa on uusin iOS-versio, joka hyväksyttiin beta-testattavaksi perjantaina myöhään illalla ja tulee toivon mukaan Applen katselmoinnista tällä viikolla hyväksytysti takaisin julkaisua varten. Tämän viikon tavoite on tehdä viimeiset hyväksyntätestaukset ja kerätä beta-käyttäjiltä palaute, jotta saadaan jälleen varmuus tuotteen laadusta.

Kehittäjät ovat tehneet kovasti töitä erilaisten virheiden ja käytettävyystekijöiden kanssa, joten perjantaina tulisi laittaa Applen katselmoitavaksi jälleen uusi iOS-versio. Viime aikoina yrityksen tavoitteena on tehdä iteratiivisesti pieniä virheen korjausjulkaisua, jotta käyttäjille saadaan mahdollisimman nopeasti uusia korjauksia, joka myös luo hyvää kuvaa kehityksestä käyttäjille. Osuuteni on siis tärkeä tällä viikolla, sillä joudun testaamaan useampaa iOS-versiota. Aloitin tänään päivän suunnittelemalla mitä testejä ehdin tekemään, jotta saadaan tällä hetkellä katselmoinnissa oleva versio julkaistua. Haasteena aikataulullisesti on työrytmini, sillä työskentelen maanantaisin, keskiviikkoisin ja perjantaisin, joten minun on mietittävä tarkasti, miten monta testiä ehdin tekemään, jotta voidaan hyvällä varmuudella antaa vihreää valoa uudelle julkaisuehdokkaalle.

Suunnittelun jälkeen aloitin saman tien funktionaalisten testien tekemisen. Ennen julkaisua tehdään laajempi hyväksyntätestaus, joten päätin, että jatkossa yritän pitää testit sa-

mana eri käyttöjärjestelmillä. Loin Testrail-ohjelmassa uuden testisuunnitelman, johon valitsin sopivat testit, joten testien suorittaminen on itsessään helppoa. Ennen version yrityksen sisäiseen testaukseen laittamista, suoritan yksityiskohtaisemmat regressiotestit, jotka vaihtelevat tehtyjen muutosten mukaan. Toimintatapana aion tehdä niin, että aloitan yhdestä käyttöjärjestelmästä ja suoritan sen jokaisen valitun testitapauksen, ennen kun siirryn seuraavaan käyttöjärjestelmään.

Haasteena viikko tuo myös Android-version testaamisen ja osuuteni siinä, sillä iOS vie itsessään paljon aikaa. Ulkoinen konsultti on muiden kehittäjien kanssa työstänyt sitä ahkerasti ja tällä viikolla pitäisi saada luotua uusi versioehdokas. Tilannetta helpottaa se, että muutokset ovat lähinnä käyttöliittymään liittyviä, joka helpottaa testausta. Minulla on käytössä vain yksi Android-käyttöjärjestelmää käyttävä laite, joten pyrin keskustelemaan tällä viikolla siitä, miten sitä tulisi testata.

Keskiviikko 4.11.2015

Aloitin aamun kutsumalla uusia käyttäjiä yrityksen iOS beta-ryhmään asiakastuen antaman listan mukaan. Lista koostui enimmäkseen käyttäjistä, joilla on viime aikoina ollut ongelmia sovelluksen kanssa. Mikäli heillä esiintyy ongelmia, voin tarkkailla sovelluksen aktiviteettejä tarkemmin, kun käyttäjiä ei ole beta-vaiheessa vielä niin paljon. Pystyn esimerkiksi Crittercism-ohjelmasta löytämään helpommin käyttäjiä vastaavat kaatumisilmoitukset, joiden avulla heitä voidaan asiakastuen kautta auttaa paremmin.

Käännösyritys oli toimittanut hiljattain Android-versioon soveltuvat suomen kielen käännökset, jotka päivän aikana lisäsin versionhallintaan. Koska iOS-versiolle saadaan pian suomen kielen lokalisaatio, voi olla että ennen varsinaista uutta Android-versiota teemme väliversion, joka sisältäisi vain käännökset. Loppujen lopuksi se on esimieheni päätös, kuinka Android-versiossa edetään.

Käytin suurimman osan päivästä Applen katselmoinnissa olevan iOS-version funktionaaliseen testaamiseen. Tavoitteena oli saada se tänään julkaistua, sillä se hyväksyttiin tiistaina katselmoinnista, joten se oli painiketta vaille julkaistu. Koska olin maanantaina suunnitellut hyvin mitkä testit suoritan, pystyin keskittymään ainoastaan niiden tekemiseen. Päivän lopuksi generoin Testrail-ohjelman avulla raportin, johon minulla oli myös mahdollisuus lisätä sanallinen kuvaus, joten minun ei tarvinnut kirjoittaa erillistä dokumenttia. Kun kaikki löytyy yhdestä paikasta, muiden on myös helpompi löytää kaikki tarvittavat tiedot. Raportin perusteella annoin versiolle hyväksynnän, joten enää jäi esimieheni tehtäväksi painaa viimeistä painiketta.

Koska perjantaina on tarkoituksena saada jälleen uusi iOS-versio Applen katselmointiin, suunnittelin vielä päivän lopuksi mitkä testit teen perjantaina. Testit ovat tyypiltään regressiomaisia sillä laajempi hyväksyntätestaus tehdään ennen varsinaista julkaisua. Tähän versioon tehtiin koodimuutoksia, joissa toiminnallisuus säilyy samana, mutta sisäistä rakennetta parannettiin. Minun tehtäväni on perjantaina käydä sovellus läpi niin hyvin kuin pystyn annetussa ajassa, jotta voidaan varmistaa, että kaikki sovelluksen ominaisuudet säilyvät ehjinä.

Perjantai 6.11.2015

Uusi iOS-versio julkaistiin keskiviikkona App Store –kauppaan, joten tänään aloitin aamun tarkistamalla kyseisen version tilanteen Crittercism-ohjelmalla. Mitään kriittisiä virheitä ei esiintynyt, mutta perinteisesti tarkoitus on aktiivisesti seurata viikon ajan minkälaisia virheitä käyttäjillä voi esiintyä. Muuten päivän tavoitteena oli saada sitä seuraava iOS-versio Applen katselmointiin, joten tein suurimman osan päivästä regressiotestausta, jotka olin suunnitellut keskiviikkona.

Asiakastuen pyynnöstä lisäsin jälleen uusia käyttäjiä iOS beta-ohjelmaan ja osalle ei ollut tullut sähköpostikutsua, joten jouduin lähettämään ne uudestaan. Jatkossa näiden käyttäjien toimintaa sovelluksessa voidaan seurata tarkemmin.

Sain apua testaukseen toiselta kehittäjältä, jolla oli pieni hetki aikaa auttaa manuaalisten testien loppuun saamisessa. Koska testattavaan versioon tehtiin paljon koodimuutoksia, regressiotestaaminen on tällä kertaa astetta suurempi. Käytin loppupäivän näiden testien tekemiseen, jonka lopputuloksena versio saatiin onnistuneesti Applen katselmointiin. Ulkoinen konsultti sai myös alustavan Android-implementaation, jossa oli osa sovituista muutoksista, joten siitä tehtiin yrityksen sisäiseen testaukseen väliversio. Ensi viikon tavoitteena on kerätä palautetta, sillä konsultti oli hieman epävarma siitä, kuinka toteutus toimii. Tulevaisuudessa on mietittävä kuinka eri alustojen testaaminen hoidetaan, koska iOS itsessään vie oman aikansa.

Viikkoanalyysi

Viikon tavoitteena oli saada yrityksen iOS-versioita eteenpäin ja niissä mielestäni onnistuttiin, sillä keskiviikkona julkaistiin yksi versio ja sitä seuraava saatiin Applen katselmointiin perjantaina. Toinen tavoitteista oli kuitenkin saada myös Android-väliversio sisäiseen testaukseen. Sovittiin, että se tehtäisiin mikäli aikaa jää. Huomasin tällä viikolla kuinka aikaa

vievää manuaalisten testien suorittaminen on - varsinkin kun tulisi testata useampaa versiota. Tilannetta vaikeuttaa, että työskentelen osa-aikaisesti yrityksessä, mikä taas vie viikon testausajasta. Olemme keskustelleet useamman työntekijän kanssa, mukaan lukien esimieheni kanssa, kuinka tätä osaa testausprosessia tulisi tehostaa. Käytännössä tämä tarkoittaa ainakin osan testien automatisointia ja tarkan testisuunnitelman laatimista, jossa olisi vain oleellisimmat testitapaukset.

Patton (2001, 220-221) esittää testityökalun ja automaation keskeiset periaatteet:

- Nopeus
- Tehokkuus
- Tarkkuus
- Resurssien vähentäminen
- Simulaatio
- Sinnikkyys

Luonnollisesti jos ainakin osan tämän hetkisistä testeistä saisi automatisoitua niin, että esimerkiksi kone suorittaisi niitä, se säästäisi aikaa testaajalta eli minulta, jotta voin keskittyä testeihin, joihin tarvitaan ”ihmisen” kosketus. Tämä on suoraan sidonnainen testauksen tehokkuuteen, sillä kun suoritan manuaalisia testejä, en pysty tekemään muita juttuja, kun taas automatisointi mahdollistaisi useamman työtehtävän tekemisen. Olen myös huomannut, että kun testausta tekee useampia tunteja yhtäjaksoisesti, keskittyminen ei ole niin tarkkaa ja silloin ei välttämättä huomaa kaikkia virheitä tai outouksia sovelluksessa. Kun yhden käyttöjärjestelmään voi mennä 50 testitapausta ja nykyään käyttöjärjestelmiä on kolme, testitapauksia voi joutua tekemään hyvin monta. Lisäksi koneet eivät koskaan väsy ja työkalun käyttö vähentää pitkällä aikavälillä yrityksen käyttämiä resursseja testaamiseen, koska koneet tekevät osan työstä tehokkaasti.

Testiautomaation tavoitteena on vähentää testausresursseja, paljastaa regressio-ohjelmointivirheitä ajoissa ja havaita virheellisiä muutoksia uusissa versioissa (Pettichord 2002, 94). Kun saan kaikki virheet tiedostettua mahdollisimman nopeasti, se ennaltaehkäisee entisestään virheellisten versioiden julkaisemisen. Virheet voidaan raportoida lähes saman tien, jotta korjaukset saadaan aloitettua nopeammin.

Ongelmaksi voi kuitenkin esiintyä huononlaatuinen automaatio. Pettichord (2002, 102) varoittaa kirjassaan erilaisista ongelmakohdista, joita siinä voi esiintyä. Testit eivät välttämättä tee, sitä mitä testaajaa kuvittelee niiden tekevän. Sen takia on myös tärkeää, että testaaja ymmärtää ohjelmoinnista sen verran, että pystyy saamaan realistisen kuvan siitä, mitä automaatiotestit tekevät. Testit eivät välttämättä ole ajan myötä enää kiinnostavia, mikä tarkoittaa, että niitä ei tarvitsisi enää suorittaa, esimerkiksi jos sovelluksen jokin osa

muuttuu radikaalisti. Testien kattaus verrattuna koko sovellukseen voi olla vähäinen, sillä testien määrä ei kerro kuinka hyvin jokin ominaisuus testataan, joten kattavuus olisi hyvä tarkistaa.

Lardinois (2015) ilmoittaa artikkelissaan, että Amazon on julkaissut palvelun nimeltä AWS Device Farm, jonne kehittäjät voivat lähettää mobiilisovelluksensa, ja niitä testataan oikeilla laitteilla. Yrityksessä olemme aikaisemmin keskustelleet kyseisestä palvelusta, ja se olisi yksi vaihtoehto, jolla saisimme osan sovelluksen ominaisuuksista automatisoitua. Sovelluksessa käytetään erillistä Bluetooth-laitetta, joten se tuo omat haasteensa automaatioon, sillä Amazonin robotit eivät voisi tätä osaa testata. On varmasti muitakin yrityksiä, jotka tarjoavat samankaltaista palvelua, mutta tärkeintä tällä hetkellä yrityksessä on käydä keskustelua ja valita sopiva vaihtoehto, jolla saataisiin entisestään tehostettua testausta. Yksi vaihtoehto, jota voisin suositella yrityksen entisenä ohjelmoijana, on kirjoittaa omia koodipätkiä, jolla käytäisiin sovellusta läpi.

4 Pohdinta ja päätelmät

Työsuhteen alussa olin hyvin optimistinen, mutta varovainen tulevasta syksystä yrityksen ohjelmistotestaajana ja laadunvarmistajana. Korostin, että minulla ei ole aikaisempaa kokemusta ohjelmistotestauksesta, kuitenkin täytyy ottaa huomioon, että yritys ja sen tuote olivat tulleet tutuksi työharjoittelun aikana. Minulla oli alustavasti hyvä pohja tuotteen uusien versioiden julkaisuun, mutta varsinaista testausta en ollut aikaisemmin tehty. Havaitsin jo heti ensimmäisten viikkojen aikana, kuinka yritys on jättänyt tärkeät työtavat pienemmälle huomiolle, erityisesti mitä tulee regressiotestaukseen. Viimeisten viikkojen aikana olin jo muodostanut rutiinin testauksesta, varsinkin funktionaalisesta testauksesta, jonka avulla pystyin jo paljon tehokkaammin antamaan testituloksia ja raportteja liittyen uusiin julkaisuihin. Asetin tavoitteeksi luoda prosessin, jonka avulla pystyn parantamaan tuotteen laatua niin, että siinä olisi vähemmän virheitä kuin aikaisemmin ja kaikki havaitut virheet otettaisiin nopeasti käsittelyyn. Mielestäni onnistuin luomaan yritykselle järkevän prosessin, jonka avulla tavoitteet saavutettiin, mutta aina on parannettavaa. Tärkeää oli kuitenkin antaa pohja tulevaisuutta varten.

Kuten aikaisemmin mainitsin, regressiotestausta ei oltu yrityksessä oikeastaan tehty. Pystyin muiden työntekijöiden avustuksella luomaan alustavan prosessin, jonka avulla kyettiin huomaamaan kaikki kriittisimmät ohjelmointivirheet ennen yrityksen sisäiseen testaukseen julkaisua. Viimeisillä viikoilla hyödynsin tätä prosessia ja koin sen myös hyödylliseksi siinä mielessä, että se antoi mahdollisuuden ajatella regressiotestausta järjestelmällisesti. Yhdessä vaiheessa käsittelin useampaa sovellusversiota samanaikaisesti, joten ilman selkeää prosessia työstäni olisi tullut paljon vaikeampaa. Koska olin kehittäjille esitellyt tämän laadunvarmistusprosessin, sain aina tarvittavat tiedot siitä, mihin kohtiin sovellusta oli tehty muutoksia.

Barber (2007) kertoo artikkelissaan, kuinka on lähes mahdotonta testata kaikki sovelluksen käyttötapauksia. Seurantaviikkojen aikana havaitsin saman ilmiön, sillä vaikka olisin halunnut käydä jokaisessa julkaisussa kaikki testitapaukset, minun täytyi priorisoida ne niin, että kaikkein oleellisimmat suoritetaan. Tämä oli tärkeä havainto, ja saatoin aluksi ajatella testitapausten läpikäyntiä naiivisti. Lopuksi valitsin jokaista App Store julkaisua varten vakituiset testitapaukset, jotka kävisin aina läpi. Regressiotestausta varten valitsen tarkasti muutoksia vastaavat testit, joilla varmistetaan, että ne eivät aiheuttaneet virheitä sovelluksen pääominaisuuksissa. Tulevaisuudessa pystyn ajattelemaan ajan käyttöä realistisemmalla tavalla.

Yksi konkreettisimmista muutoksista yrityksen testausprosessia oli uuden testitapaustyökalun käyttöönotto. Keskustelimme aiheesta yhdessä muiden tuotetiimien jäsenien kanssa, mutta lopulta minä valitsin työkalun. Opin tässä, kuinka tärkeää on löytää omalle tekemiselle tehokkain ratkaisu, sillä ennen työkalua kirjoitin testitapauksia ja niiden tuloksia Excel-muotoiseen tiedostoon. Vaikka aluksi tämä vaikutti hyvältä vaihtoehdolta, huomasin hyvin nopeasti, miten tiedostojen hallinta oli hajanaista ja monimutkaista. Ennen kuin ryhdyin kokeilemaan uutta työkalua, kokeneempi kehittäjä suositteli kahden työkalun välistä vertailua, jotta saan mahdollisimman hyvän kuvan markkinoilla olevista vaihtoehdoista. Totesin tämän menetelmän hyväksi ja uskon, että jatkossa pystyn käyttämään samantyylistä vertailua. Esimerkiksi jos tulevaisuudessa jatkan kehittäjänä ja minun tulisi valita uusi kehitysympäristö, valitsen kahden tai useamman vaihtoehdon ennen kuin teen valintani.

Testitapaustyökalun avulla pystyin parantamaan testausprosessia, mutta samalla myös selkeyttämään sitä muiden tuotetiimin jäsenten näkökulmasta. Vaikutti siltä, että kenelläkään ei ollut selkeää kuvaa, esimerkiksi siitä, mikä oli sovellusversion testiläpäisyprosentti tai kuinka moni epäonnistui. Testrail-ohjelman avulla kaikilla tuotetiimin jäsenillä on mahdollisuus saada selkeämpi kuva, siitä mikä on jokaisen testauksessa olevan versio tilanne. Huomasin itsekin, kuinka visuaalinen käyttöliittymä toi selkeyttä ja mielekkyyttä tekemiseen. Testiraporttien luominen nopeutui myös huomattavasti, koska ohjelma tuotti sen automaattisesti, kuten oli tavoitteenani. Vastuulleni jäi vain sanallisen selityksen kirjoittaminen. Oli kiinnostavaa huomata, kuinka raportin sai helposti muotoiltua haluamaansa muotoon.

Yrityksellä on pitkään ollut käytössä järjestelmä ohjelmointivirheiden kirjaamiseen ja esittämiseen nimeltä Fogbugz. Työsuhteen alussa koin, että järjestelmän ensisijainen ylläpitäminen oli minun vastuullani, mutta virheiden kirjaaminen ei tulisi olla vain minun vastuullani. Tästä syystä tuloksen ohjeistin uutta työntekijää, kuinka järjestelmää käytetään ja miten kirjoitetaan hyvä virheraportti. Otin periaatteeksi, että kaikkien yrityksen työntekijöiden tulisi kirjata löytämiään virheitä sovellukseen liittyen. Merkitsin raporttiin erityisiä kohtia, joita siinä tulisi olla, jotta siitä olisi varsinaista hyötyä virheitä korjaaville kehittäjille. Samalla kehityin itse testaajana, koska etsin kirjallisuudesta hyviä käytäntöjä raporttien kirjoittamisen suhteen ja käytäntöjen avulla pystyin sitten ohjeistamaan muita. Tavoitteena oli tehdä Fogbugz-järjestelmästä hyödyllinen ja käyttäjäystävällinen, jotta tuotetiimi saa realistisen ja ajantasaisen kuvan sovelluksissa esiintyvistä virheistä. Seurantaviikkojen aikana huomasin haasteen, joka koski virheiden priorisointia. Oli tärkeää korjata heti ensimmäiseksi kriittisimmät virheet. Suurena apuna listan priorisoinnissa oli Crittercism-ohjelma,

jonka avulla pystyin selkeästi esittämään, mitkä ongelmat vaikuttavat käyttäjiin eniten versioittain. Tietenkään kaikki virheet eivät ohjelmassa esiinny, esimerkiksi käyttöliittymään liittyvät virheet. Opin kuitenkin, että oleellinen osa prosessia oli tutkia yleisimmät virheet ja esittää ne tuotetiimille, jotta ne saadaan suunnitellusti seuraavaan sovellusversioon.

Testausprosessia on selkeästi kehitettävä edelleen minimoimalla manuaalinen työ, jotta testaajana pystyn mahdollisesti nopeuttamaan testausta. Tällä hetkellä käytän paljon aikaa funktionaalisten testien tekemiseen, joka tarkoittaa sitä, että useamman sovellusversion hallinta voi olla haastavaa. Viimeisten seurantaviikkojen aikana huomasin, kuinka sekä iOS että Android-version testaaminen tulisi tulevaisuudessa viemään paljon aikaa, joten olisi loogista pyrkiä automatisoimaan ainakin osa funktionaalisista testeistä. Esitin viimeisen seurantaviikon analyysissä yhden vaihtoehdosta, AWS Device Farmista, joka siirtäisi manuaalisen työn roboteille. Luonnollisesti tämä on aihe, josta tulen keskustelemaan useamman työntekijän ja esimieheni kanssa. Joka tapauksessa tämä on vaihe, joka vaatii minulta myös ohjelmointiosaamista, koska prosessi voi vaatia omien koodipätkien tekemistä. Esimerkiksi avoimen lähdekoodin työkalun käyttäminen iOS testitapausten tekemiseen vaatii ohjelmointiosaamista. On olennaista, että kirjoitetaan miten ohjelman tulisi testitapauksessa käyttäytyä. Olen pääasiassa kuitenkin kiinnostunut ohjelmoinnista ja minut palkattiin yritykseen harjoittelevaksi kehittäjäksi, joten uskon, että onnistun tehtävässä.

Alkutavoitteisiin nähden uskon, että syksyn aikana olen kehittynyt testaajana ja yrityksen testausprosessissa on huomattu selkeitä parannuksia. Sen parantaminen on jatkuva tehtävä, mutta uskon, että olen luonut vakaan pohjan ja seuraavat edistysvaiheet ovat jo suunnitteilla. Tärkeää on ylläpitää ja jatkuvasti miettiä kehitysideoita, jotta laadunvarmistusprosessista saadaan entistä vakuuttavampi. Huomasin, että ammattikirjallisuudesta sain paljon hyviä näkemyksiä, joiden avulla pystyin esittämään uusia ideoita yrityksen prosessiin. Koska analyyseissä olin käynyt yhden osa-alueen kattavasti läpi, pystyin seuraavalla viikolla saman tien ottamaan opit käyttöön. Koin, että analyysit selkeästi tukivat kehitystäni ja nostivat itsevarmuutta parannusehdotuksien tekemiseen. Tulevaisuudessa pyrin myös erilaisten artikkelien ja kirjallisuuden avulla vahvistamaan osaamistani ja niiden avulla miettimään, mikä on paras ratkaisu pienelle yritykselle. Seuraava tavoite prosessissa on funktionaalisten testien automaatio, joten tätä varten pyrin löytämään työkaluja ja kirjallisuutta aiheeseen liittyen. Opinnäytetyön antamien analyysien pohjalta on selkeää kuinka jatkaa eteenpäin.

Lähteet

Barber S. 2007. Software performance testing: You can’t test everything. Luettavissa: http://searchsoftwarequality.techtarget.com/tip/Software-performance-testing-You-cant-test-everything. Luettu 11.11.2015.

Capucci F. & Gumaste V. 2013. Test Case Writing. Luettavissa: https://university.utest.com/test-case-writing-creation/. Luettu 10.9.2015.

Doan A. 2015. Automated mobile app functional testing with Appium. Luettavissa: http://www.techinsight.io/review/devops-and-automation/automated-functional-testing-of-mobile-apps-using-appium/. Luettu 14.11.2015.

Goucher A. & Riley T. 2009. Beautiful Testing: Leading Professionals Reveal How They Improve Software.

Haden J. 2012. Be Graceful Under Pressure: 7 Tips. Luettavissa: http://www.inc.com/jeff-haden/how-to-be-graceful-under-pressure.html. Luettu 10.10.2015.

Hornreich H. 2014. iOS UI Automated Testing Tools. Luettavissa: http://appleprogramming.com/blog/2014/02/07/ios-ui-automated-testing-tools/. Luettu 14.11.2015.

Kalashnikov A. 2014. Tools Battle: SpiraTest, TestRail and TestLodge. Luettavissa: http://sysgears.com/articles/tools-battle-spiratest-testrail-and-testlodge/. Luettu 17.10.2015.

Lardinois F. 2015. Amazon Launches AWS Device Farm, Lets Developers Test Android And Fire OS Apps On Real Devices. Luettavissa: http://techcrunch.com/2015/07/09/amazon-launches-aws-device-farm-lets-developers-test-android-and-fire-os-apps-on-real-devices/#.xbleflx:a6Ru. Luettu 7.11.2015.

Marick B. 1997. Classic Testing Mistakes. Luettavissa: http://www.exampler.com/testing-com/writings/classic/mistakes.html. Luettu:11.11.2015.

Murali Balasubramanyam. The Best Practices For Regression Testing. Luettavissa: http://www.softwarespecialists.com/best-practices-everyone-implement-regression-testing/. Luettu 3.9.2015.

Patton R. 2001. Software Testing. Sams Publishing. Yhdysvallat.

Pettichord B., Bach J. & Kaner C. 2002. Lessons Learned in Software Testing: A Context-Driven Approach. Wiley Computer Publishing. New York.

Reminnyi S. 2015. Developing Test Automation Scripts and Automation Frameworks. Luettavissa: http://www.infoq.com/articles/test-scripts-frameworks. Luettu: 11.11.2015.

Sachin 2012. 8 Best Software Testing Books QA Engineer / Tester Must read. Luettavissa: http://www.fromdev.com/2012/04/8-best-software-testing-books-every-qa.html. Luettu: 29.8.2015.

Software Testing Help 2015. A Simple 12 Steps Guide to Write an Effective Test Summary Report [with A Sample Report for Download]. Luettavissa: http://www.softwaretestinghelp.com/test-summary-report-template-download-sample/. Luettu: 26.9.2015.

Software Testing Help 2015. How TestLodge Test Management Improved with Your Feedback. Luettavissa: http://www.softwaretestinghelp.com/testlodge-test-management-tool-review/ . Luettu: 3.10.2015

Software Testing Help 2015. How to Improve the Test Release Process For Successful Bug Free Software to Production. Luettavissa: http://www.softwaretestinghelp.com/how-to-improve-the-test-release-process/. Luettu: 18.9.2015.

Software Testing Help 2015. What is Regression Testing? Regression Testing Tools and Best Practices. Luettavissa: http://www.softwaretestinghelp.com/regression-testing-tools-and-methods/. Luettu: 10.9.2015.

Thomson C. 2014. 5 Steps to prioritizing when you can't test everything. Luettavissa: http://www.te52.com/testtalk/2014/09/25/5-steps-prioritizing-cant-test-everything/. Luettu: 11.11.2015.

Waterhouse G. 2013. Advantages of using Test Managements tools. Luettavissa: http://www.testingthewaterhouse.com/2013/05/advantages-of-using-test-management.html. Luettu: 12.9.2015.