Haaga-Helia
ammattikorkeakoulu Oy

Graafisen käyttöliittymän testauksen suunnittelu

Kaisa Naumova

Opinnäytetyö
Tietojenkäsittelyn
koulutusohjelma
2021

Tiivistelmä

Haaga-Helia
ammattikorkeakoulu Oy

Tekijä(t)
Kaisa Naumova

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Raportin/Opinnäytetyön nimi
Graafisen käyttöliittymän testauksen suunnittelu

Sivu- ja liitesivumäärä
51 + 5

Ohjelmistoja on nykyään lähes kaikissa laitteissa ja niiden lisääntyminen on johtanut siihen, että myös ohjelmoijia on enemmän. Eri tasoisten ohjelmoijien myötä myös virheiden mahdollisuus ohjelmistoissa on lisääntynyt ja johtanut bugien määrän kasvamiseen. Koska uusia ohjelmistoja halutaan käyttöön nopealla tahdilla, mahdollisuus siihen, että kaikkia virheitä ei huomata koodausvaiheessa, on lisääntynyt. Tämä lisää hyvin suunnitellun testausprosessin tarpeellisuutta.

Tämän opinnäytetyön aiheena on graafisen käyttöliittymän testauksen suunnittelu. Opinnäytetyössä käsitellään testauksen suunnittelua, testaussuunnitelman tarkoitusta ja sisältöä, käyttöliittymän toiminnallisuuksien testaamista, käyttöliittymän käytettävyyttä sekä käytettävyyden testaamista. Jokaisesta aiheesta esitellään ensin tietoperusta ja sen jälkeen aihetta käsitellään teoriaa seuraavassa empiiriaosuudessa.

Opinnäytetyön taustana on erään yrityksen kulunvalvonnan ja työajanseurannan järjestelmän uuden web-pohjaisen käyttöliittymän testaustarve. Yritys haluaa testata käyttöliittymän toiminnallisuudet ja käytettävyyden ennen kuin käyttöliittymä annetaan asiakkaalle pilotoitavaksi. Testauksen aikana suoritetaan kahdenlaista testausta: toiminnallisessa testauksessa etsitään järjestelmään mahdollisesti jääneet bugit, ja käytettävyystestauksessa halutaan saada tietoa käyttöliittymän käytettävyydestä ja käyttökokemuksesta. Tämä opinnäytetyö on toiminnallinen ja opinnäytetyön tuloksena syntyi käyttöliittymän testaukselle testaussuunnitelma sekä testauksessa käytettävät testauslomakkeet.

Opinnäytetyön laajuudesta on rajattu pois itse testaus ja testauksen tulokset sekä tulosten analysointi. Opinnäytetyössä keskitytään testauksen suunnitteluun ja testausdokumenttien laadintaan. Teoriaosuudesta on rajattu pois automaatiotestauksen eri työkalut, sillä nyt suunniteltavaa käyttöliittymän testausta ei ole suunniteltu suoritettavaksi automaatiotyökaluja käyttäen.

Asiasanat
Testaus, testaussuunnitelma, käytettävyys, käyttöliittymä, testaussuunnittelu

Sisällys

1 Johdanto

Uuden järjestelmän kehitys lähtee aina siitä ajatuksesta, että järjestelmälle on tarvetta. Uuden järjestelmän toivotaan auttavan käyttäjäänsä hoitamaan tarvittava toiminto aikaisempaa tehokkaammalla ja helpommalla tavalla. Ja olisi tietenkin hyvä, jos järjestelmän käyttämisestä jäisi sen käyttäjälle myös hyvä mielikuva, joka saisi käyttäjän palaamaan käyttämään järjestelmää myös uudestaan. (Filenius 2015, osa 6.)

Ohjelmistoja löytyy nykyään lähes kaikista laitteistoista, joita ihminen on luonut. Ohjelmistotuotteiden lisääntyminen on johtanut siihen, että myös ohjelmoijia on enemmän, jonka vuoksi kaikkia ohjelmistoja ei ole ohjelmoinut kokenut ohjelmoija. Sen myötä myös virheiden mahdollisuus ohjelmistoissa on lisääntynyt ja johtanut bugien määrän kasvamiseen. Nykyään aika on rahaa ja uusia ohjelmistoja halutaan käyttöön nopealla tahdilla, mikä lisää mahdollisuutta, että kaikkia virheitä ei huomata. (Ratilainen 2019.)

Tämän vuoksi testaus on ohjelmistolle välttämättömyys. Jos testaus on hyvin suunniteltu, voi se nostaa ohjelmiston tason keskinkertaisesta hyväksi. Huonosti suoritella testauksella voi olla jopa laatua heikentävä vaikutus. Sen vuoksi on tärkeää, että ohjelmistoprojektiin kuuluu myös hyvin suunniteltu testausprosessi, joka on juuri kyseistä projektia koskeva. (Ratilainen 2019.)

Testauksen tavoitteena on löytää järjestelmässä piilevät viat ennen tuotteen julkaisua. Mahdollisimman ehjä ja viaton järjestelmä parantaa asiakkaiden luottoa järjestelmää tarjoavaan yritykseen, joten testaukseen kannattaa panostaa. Testausta ei kuitenkaan kannata lähteä suorittamaan ilman huolellista suunnittelua.

Ohjelmistotestauksesta on luotu yhtenäisiä linjauksia, joita tulisi noudattaa. International Software Testing Qualifications Board (ISTQB) on instituutio, joka on kerännyt ohjelmistotestaukseen liittyvät yksityiskohdat yhteen ja luonut sertifikaatin, jonka testaajat ympärimaailman voivat suorittaa. ISTQB:n sertifikaatin suomen kielisestä sisällöstä vastaa Finnish Software Testing Board (FiSTB), jonka kautta testaukseen liittyviä sertifikaatteja voi suorittaa.

Tämän opinnäytetyön aiheena on graafisen käyttöliittymän testauksen suunnittelu. Toimeksiantaja haluaa pysyä tuntemattomana, joten yrityksestä käytetään nimitystä Yritys Oy ja käyttöliittymästä nimitystä Käyttöliittymä X. Opinnäytetyön tavoitteena on selvittää

- mitä tulee ottaa huomioon käyttöliittymän testausta suunniteltaessa
- miten käyttöliittymän toiminnallisuuksia ja käytettävyyttä testataan
- mitä käyttöliittymän testaussuunnitelma sisältää

- miten Yritys Oy:n Käyttöliittymä X:n testaus suoritetaan

Alikysymyksinä opinnäytetyössä käsitellään

- miten testaus sisältyy ohjelmistojen elinkaarimalleihin
- mitä eri tasoja ohjelmistojen testausta on
- millä eri tekniikoilla käyttöliittymää voidaan testata
- mitä tarkoitetaan käyttöliittymän käytettävyydellä

Opinnäytetyön tuloksena syntyy testaussuunnitelma Yritys Oy:n Käyttöliittymä X:n testaamiseen sekä testauksessa käytettävät testauslomakkeet.

Tämän opinnäytetyöprosessin suunnittelun seurauksena testattava Käyttöliittymä X on työajanseurantaan ja kulunvalvontaan tarkoitettu selainpohjainen käyttöliittymä. Tämä käyttöliittymä on tarkoitettu korvaamaan nykyinen palvelimelle asennettava käyttöliittymä. Tarkoituksena on tässä vaiheessa pitää käyttöliittymän work flow samanlaisena kuin se on nykyisessä käyttöliittymässä, ja tehdä vain teknologinen hyppäys. Lisäksi käyttöliittymään yhdistetään uusi web-raportoinnin työkalu. (Kuronen 3.12.2020.)

Yritys Oy haluaa testata Käyttöliittymä X:n toiminnallisuudet ja käytettävyyden ennen kuin käyttöliittymä annetaan asiakkaalle pilotoitavaksi. Testauksen aikana suoritetaan kahdenlaista testausta: toiminnallisessa testauksessa etsitään järjestelmään mahdollisesti jääneet bugit ja käytettävyystestauksessa halutaan saada tietoa käyttöliittymän käytettävyydestä ja käyttökokemuksesta. Käyttöliittymän toiminnallisuuksien testaus ja käytettävyyden testaus on eroteltu teoriaosuudessa, jotta molemmista testaustavoista saadaan teoreettinen pohja testaamiselle.

Käyttöliittymä X:ää on koodattu valmiiksi näyttö kerrallaan ja sitä on jo alettu testaamaan kehitystiimin pyynnöstä muutamien henkilöiden toimesta ennen nyt tehtävän suunnitelman valmistumista. Käyttöliittymän testausta varten ei ole tehty ennestään selkeää suunnitelmaa, jonka vuoksi testausta ei suoriteta systemaattisesti eikä sitä dokumentoida järjestelmällisesti. Nyt tehdyn testaussuunnitelman avulla varmistetaan, että käyttöliittymän jokainen osa tulee varmasti testattua, ja toisaalta vältetään turhaa työtä estämällä päällekkäisten testausten mahdollisuuden selkeästi eri henkilöille jaettujen osioiden muodossa. Testauksen tulokset myös dokumentoidaan testauslomakkeiden ja vikatikettien avulla yhdenmukaisesti.

Opinnäytetyö on tyypiltään toiminnallinen. Opinnäytetyöprosessin aikana luon Yritys Oy:lle Käyttöliittymä X:n testaamista varten testaussuunnitelman sekä testauksessa käytettävät tehtävälomakkeet. Lisäksi suoritan testattavien näyttöjen osioinnin ja osioiden jaon testaajien kesken. Testaussuunnitelman sisällössä esiintyvällä testauksen suunnittelijalla

tarkoitetaan minua. Opinnäytetyö on aseteltu niin, että teoriaa sisältävät pääluvut ja empiriaa sisältävät pääluvut vuorottelevat.

Opinnäytetyön laajuudesta on rajattu pois itse testaus ja testauksen tulokset sekä tulosten analysointi. Opinnäytetyössä keskitytään testauksen suunnitteluun ja testausdokumenttien laadintaan. Teoriaosuudesta on rajattu pois automaatiotestauksen eri työkalut, sillä nyt suunniteltavaa käyttöliittymän testausta ei ole suunniteltu suoritettavaksi automaatiotyökaluja käyttäen.

Varsinainen lopullinen testaussuunnitelma on luottamuksellinen, mutta testaussuunnitelman sisältö on esitelty luvussa 3. Opinnäytetyö ei sisällä kuvia testattavasta käyttöliittymästä.

2 Testauksen suunnittelu ja valmistelu

Testauksen näkyvin osa on itse testaus, mutta testauksen suunnittelu ja tulosten analysointi ovat hyvin tärkeitä osia testauksen onnistumisen kannalta. Testausta ei voi suorittaa ennen kuin on päätetty miten, milloin ja mitä testataan. Testaus on erilainen joka kerta, joten ilman huolellista suunnittelua ei testauksen suorittaminen ole mahdollista. Suunnittelun aikana luodaan kartta siitä, kuka tekee, mitä ja missä vaiheessa. Suunnittelun aikana määritellään testauksen tavoitteet ja valitaan keinot, joiden avulla tavoitteita lähdetään saavuttamaan. Nämä tavoitteet ja valitut keinot kirjoitetaan ylös testaussuunnitelmaan. (Morgan 2019, Luku 1.) Tässä luvussa esittelen mitä kaikkea suunnitteluprosessiin kuuluu. Testaussuunnitelmaan paneudun syvemmin luvussa 3.

2.1 Suunnittelun aikana päätettävät asiat

Testausprosessi voidaan jakaa seuraaviin tehtävien pääryhmiin: testauksen suunnittelu, testauksen seuranta ja hallinta, testianalyysi, testien suunnittelu, testien valmistelu, testien suoritus ja testauksen päättäminen. Nämä tehtäväryhmät toteutetaan yleensä iteratiivisesti osittain päällekkäin ja osaa niistä jatkuvasti, varsinkin ketterää elinkaarimallia noudatettaessa. Testauksen suunnitteluun kuuluvien tehtävien aikana määritellään, mitkä ovat testauksen tavoitteet ja millä lähestymistavoilla nämä tavoitteet saavutetaan. Tavoitteiden ja lähestymistapojen valinnassa tulee ottaa huomioon testauksen rajoitteet, ja miettiä, mitkä testaustekniikat ja -tehtävät soveltuvat kyseiseen testaukseen, ja millä aikataululla testaus tulisi suorittaa, jotta saavutetaan sovitut aikataulut. (FiSTB 2018, 17.)

Testauksen suunnittelu on testausta valmistelevan henkilön tärkein tehtävä. Testauksen suunnittelu varmistaa, että ennen testausta on olemassa lista tehtävistä ja merkkipaaluista, joita vasten testauksen edistymistä voidaan verrata, ja näin saadaan tietoa testauksen edistymisestä. Suunnittelun aikana myös määritellään testauksen laajuus ja siihen käytettyjen resurssien suuruus. Keskustelussa tunnistetaan ja sovitaan myös testauksen tavoitteet ja rajoitukset aikaan, laatuun ja kustannuksiin liittyen. Tavoitteet määrittävät sen, milloin testaus on valmis. (Thompson 2019, luku 5.)

Suunnitteluun kuuluu myös aikataulujen muodostaminen, joskin aikatauluja voidaan muokata projektin edetessä, mikäli projektin muut osat niin vaativat (Morgan 2019, luku 1). Aikataulua muodostettaessa on hyvä ottaa huomioon myös valmisteluihin kuluva aika. Suunnitelman ja aikataulun tulee myös mennä yhteen valitun elinkaarimallin kanssa. Aikatauluun kannattaa varata aikaa myös riskianalyysille. (Black 2016, luku 1.3.) Suunnittelun aikana tulisi miettiä testauksen aikataulua, joko niin, että kukin tehtävä on

aikataulutettu kalenteriin, tai niin, että tehtävien suorittamiselle on annettu tietty aikahaarukka. (Thompson 2019, luku 5.)

Testauksen suunnittelu on hieman erilaista riippuen siitä, millaista elinkaarimallia ohjelmiston suunnittelussa käytetään. Peräkkäismalleissa testien suunnittelu voi alkaa jo elinkaaren alussa siinä vaiheessa, kun ohjelmiston vaatimusmäärityksiä muodostetaan. Jos kuitenkin käytössä on iteratiivinen elinkaarimalli, ajoittuu suunnittelu jokaisen syklin alkuun. Tällöin suunnitteluun ei ole käytettävissä projektin alussa valmiiksi mietittyjä määrityksiä, vaan testaajat saavat jokaisen syklin alussa sillä hetkellä voimassa olevat käyttäjätarinat. Testaajien tulisikin olla mukana muokkaamassa käyttäjätarinoita iteraatiota suunniteltaessa. Samalla tulisi tehdä riskien analysointia ja arviointia. Testauksen suunnittelu ja implementointi tapahtuvat juuri ennen kuin itse testaus tapahtuu. (Black 2016, luku 1.3.) Testauksen sijoittumisesta osaksi elinkaarimallia kerrotaan lisää opinnäytetyön luvussa 5.

Testauksen suunnittelijan olisi hyvä miettiä projektipäällikön kanssa, mitä testataan, mitä rooleja testaukseen kuuluu, ja kuka suorittaa testaustehtävät. On suunniteltava ja sovittava, milloin ja miten testaustehtävät suoritetaan, kuinka testauksen tuloksia arvioidaan, ja määritellä milloin testaus lopetetaan. Ennen testausta tulee myös suunnitella, miten testaus dokumentoidaan, mihin testauksen tulokset kerätään ja kenen vastuulla on testauksen tulosten analysointi. Testauksen hallinnointiin liittyen tulee päättää, mitä mittareita vasten testauksen tuloksia vertaillaan, ja millaiset prosessit testauksen monitorointiin ja valvontaan luodaan. (Thompson 2019, luku 5.)

Testauksen valmisteluihin kuuluu myös testausympäristön valmistelu. Ennen testausta tulee konfiguroida testiympäristö, testata sen toiminta ja varmistaa, että tarvittava tuki on saatavilla. Kannattaa myös testata ohjelman asennus testiympäristöön. (Black 2016, luku 1.3.) Suunnittelussa kannattaa muistaa, että se on jatkuvaa toimintaa, joka jatkuu koko testausprojektin ajan. Riskien ja muutosten osuessa kohdalle, suunnitelmaa ja suunnittelua tulee muokata näitä riskejä ja muutoksia vastaaviksi ja huomioiviksi. Pohjasuunnitelma on lyöty lukkoon projektin alussa, joten muutokset käsitellään projektin muutosprosessin kautta. (Thompson 2019, luku 5.)

2.2 Testausstrategia ja lähestymismalli

Testauksen suunnittelun aikana tulee ottaa huomioon valittu elinkaarimalli ja luodaan sen pohjalta testausstrategia. Testausstrategian avulla voidaan varmistaa, että kullekin testaustasolle on määritelty aloitus- ja lopetuskriteerit. (Thompson 2019, luku 5.) Aloitus- ja lopetuskriteerien selitykset ja sisällöt esittelen luvussa 3.

Testausstrategia on tuote- tai organisaatiotasoinen kuvaus testausprosessista ja testauksen lähestymistavasta. FiSTB (FiSTB 2018, 60–61) esittelee tyypillisimpiä testausstrategioita ja niiden piirteet on esitelty taulukossa 1. Testausstrategia voi olla analyyttinen, mallipohjainen, metodinen, prosessia noudattava, reaktiivinen, ohjattu tai regressiota välttävä. Usein näitä testausstrategioita yhdistellään, jotta löydetään tiettyyn testaukseen sopiva lähestymistapa. Lähestymistapa muokkaa yleisestä testausstrategiasta tiettyä projektia varten sopivan. Testauksen lähestymistapaa käytetään pohjana testaustekniikoiden, testaustasojen ja testityyppien valinnassa sekä aloitus- ja lopetusehtojen määrittelyssä. Lähestymistapaa valittaessa tulee pohtia riskejä, käytössä olevia resursseja ja osaamista, järjestelmän luonnetta, testauksen tavoitteita sekä lakeja ja säädöksiä.

Taulukko 1. Tyypillisiä testauksen lähestymistapoja

STRATEGIA	KUVAUS
Analyyttinen strategia	Luottaa jonkin vaikuttavan asian analysointiin (esim. riskien)
Mallipohjainen strategia	Pohjaa testit johonkin malliin
Metodinen strategia	Luottaa aiemmin käytettyihin testeihin
Prosessia noudattava strategia	Luottaa standardien kanssa käytettäviin prosesseihin
Reaktiivinen strategia	Testaus perustuu tapahtumiin reagointiin enemmän kuin etukäteen suunnitteluun
Ohjattu strategia	Testauksen ohjaamiseen käytetään asiantuntijoiden konsultointia
Regressiota välttävä strategia	Tarkoituksena välttää olemassa olevien testausmateriaalien uudelleen käyttöä

2.3 Testitapaukset

Testin suunnittelussa testattava tilanne muunnetaan testitapaukseksi. Suunnittelussa täytyy päättää, millä tasolla testitapaukset halutaan dokumentoida. Hyvin konkreettiset testitapaukset tuottavat tulokseksi yksityiskohtaisia syötteitä ja oletettuja vastauksia, kun taas loogiset ja korkeatasoiset testitapaukset tuottavat sääntöjä, joiden avulla syötteitä voidaan muodostaa tai joiden avulla vastauksia voidaan arvioida. Testitapauksia suunnitellessa tulee myös päättää, mitä tekniikkaa testitapausten luomiseen käytetään. Onko kenties tarvetta käyttää jotain työkalua testitapausten luontiin, ja onko kyseinen työkalu käytettävissä. Lisäksi tulee miettiä, missä järjestyksessä testit tulisi suunnitella ja toteuttaa. (Black 2016, luku 1.5.)

Testauksen suunnittelija käyttää yleensä testitapausten suunnittelussa pohjana testauksen olosuhteita. Hänen tulee kuitenkin ottaa huomioon myös testausstrategiassa ja testaussuunnitelmassa olevat rajoitukset ja ohjeet. Kaikkia testauksen olosuhteita ei kuitenkaan tarvitse kääntää testitapauksiksi, varsinkaan, jos tarkoitus on käyttää reaktiivista testaustapaa. Reaktiivisessa testauksessa testaaja reagoi siihen tietoon, mitä järjestelmä tuo eteen. Tärkeää olisi myös tietää, milloin testi on onnistunut ja milloin ei. Testauksesta ei ole hyötyä, jos testauksen päätteeksi ei pystytä toteamaan menikö testi läpi. (Black 2016, luku 1.5.)

Testitapauksen perustietoihin kuuluvat testattavan käyttöliittymätapahtuman kuvaus, testauksen olosuhteet ja oletettu tulos. Testitapausten hallinnan helpottamiseksi voi olla hyvä lisätä esimerkiksi linkkejä määrittelydokumenttiin tai vikojen seurantajärjestelmään. Testauksen helpottamiseksi on hyvä käyttää muutamaa hyväksi todettua tapaa. Ensiksi; on hyvä pitää testauksessa käytettävä data erillään testitapauksista, jolloin niiden hallinnointi on helpompaa. Myös olosuhteiden kuvaus kannattaa laittaa erilliseen dokumenttiin, jolloin testitapausta ei tarvitse muokata, vaikka olosuhteet muuttuisivatkin. (Ranorex 2021.)

Toiseksi; on hyvä tehdä testitapauksista modulaarisia, jolloin niitä voidaan suorittaa missä järjestyksessä tahansa. Tällöin testaus muistuttaa paremmin aitoa käyttötilannetta, koska käyttäjät eivät käytä järjestelmiä aina siinä järjestyksessä kuin niiden kehittäjät olettavat. Kolmas vinkki on, että kannattaa luoda testitapauksia, joissa oletetaan sekä onnistumisia että epäonnistumisia. Onnistuneissa testitapauksissa vahvistetaan käyttöliittymän toimintaa ja epäonnistuneissa vahvistetaan järjestelmän reaktiot vääriin syötteisiin. (Ranorex 2021.)

Toimenpiteet voidaan kuvata joko hyvin yleisesti tai yksityiskohtaisesti. Se miten yksityiskohtaisesti toimenpiteet kuvataan, riippuu siitä, miten kokeneita testaajat ovat. Mitä tutumpi järjestelmä on testaajille, sitä vähemmän sitä tarvitsee kuvata. Myös sillä on vaikutusta, miten usein käyttöliittymästä tulee uusia versioita. Jos versioita päivitetään tiheästi, voi liian yksityiskohtaisten testitapausten päivittäminen olla työlästä. Mikäli järjestelmässä navigointi on hyvin vapaata, on myös eri testitapausten kirjoittaminen hankalampaa, koska mahdollisia reittivaihtoehtoja on niin monta. (Ranorex 2021.)

Suunnittelun aikana syntyvistä dokumenteista keskeisin on testaussuunnitelma. Seuraavassa osiossa kerron tarkemmin, mikä on testaussuunnitelman tarkoitus, ja mitä se sisältää.

3 Testaussuunnitelman tarkoitus ja sisältö

Testauksen suunnitteluun vaikuttavat monet asiat organisaation käyttämästä testausstrategiasta ja -politiikasta, käytettyyn ohjelmistokehityksen elinkaarimalliin ja menetelmiin. Lisäksi se miten laajaksi testaus on suunniteltu, mitä sillä halutaan tavoittaa, millaisia riskejä ja rajoitteita testaukseen liittyy, kuinka kriittisesti testauksen tuloksiin suhtaudutaan ja millaisia resursseja testaukseen on käytettävissä. Testaussuunnitelma kerää nämä testaukseen vaikuttavat asiat yhteen ja linjaa tehtävät toimenpiteet. (FiSTB 2018, 59.) Tässä luvussa esittelen, miksi testaussuunnitelma tulisi tehdä, ja mitä sen tulisi sisältää.

3.1 Testaussuunnitelman tarkoitus

Testaussuunnitelma sisältää korkealla tasolla kerrottuna, mitä testausaktiviteetteja on suunniteltu. Testaussuunnitelma muodostetaan normaalisti projektin alkuvaiheissa ja sitä muokataan projektin edetessä. Testaussuunnitelma sisältää riittävästi tietoa, jotta sen pohjalta voidaan muodostaa testausprojekti. Kullekin testaustasolle voidaan yleisen testaussuunnitelman lisäksi muodostaa yksityiskohtaisemmat testaustasokohtaiset suunnitelmat. (Thompson 2019, luku 5.)

Graafisen käyttöliittymän testaussuunnitelma määrittää rajat testausprojektille. Ennen testitapausten kirjoittamista on tärkeää olla testaussuunnitelma, joka identifioi saatavilla olevat resurssit ja priorisoi testaukseen valittavat alueet. Tämän tiedon avulla testaustiimi voi luoda testauskirjan ja testata skenaarioita, testitapauksia ja skriptejä. (Ranorex 2021.)

Testaussuunnitelma ohjaa testausta haluttuun suuntaan. Hyvin kirjoitettu testaussuunnitelma edesauttaa testauksen onnistumista. Testaussuunnitelman avulla pystytään myös jakamaan tietoa testauksesta muille sidosryhmille, jolloin testauksesta tulee läpinäkyvämpää kaikille osapuolille. Testaussuunnitelma ei ole muuttumaton dokumentti, vaan sitä voidaan muokata olosuhteiden muutosten mukana. (Software Testing Help 2021.)

3.2 Testaussuunnitelman sisältö

Testaussuunnitelman sisällön laajuus ja yksityiskohtien tarkkuus riippuu siitä, missä vaiheessa projektia ja testaussuunnittelua testaussuunnitelma muodostetaan. Testauksen suunnittelua tehdään läpi koko ohjelmiston elinkaaren. Suunnittelusta voidaan muodostaa yksi kokonaistestaussuunnitelma, jokaisesta testaustasosta voidaan tehdä oma suunnitelmansa (esimerkiksi järjestelmätestaus- ja hyväksymistestaussuunnitelmat) tai eri

testaustyypit voidaan jakaa omiin suunnitelmiinsa (esimerkiksi käytettävyystestaus- ja suorituskykytestaussuunnitelmat). (FiSTB 2018, 59.)

Testaussuunnitelman sisältö vaihtelee tapauskohtaisesti ja suunnitelmaan sisällytettävät osa-alueet riippuvat lähteestä, joskin pääpiirteet ovat kaikissa samat. Testaussuunnitelmaan voidaan sisältää erilaisten testauksen suunnittelutehtävien dokumentointi. Näitä tehtäviä on lueteltu FiSTB:n Perustason sertifikaattisisällössä. Niihin kuuluu muun muassa testauksen laajuuden, tavoitteiden ja riskien määrittely, testauksen yleisten lähestymistapojen määrittely sekä testaustehtävien integrointi ja koordinointi ohjelmiston elinkaaren tehtävien kanssa. Testaussuunnitelmaa tehtäessä tulee päättää, mitä testataan, millaisia henkilöstö ja muita resursseja tarvitaan eri testaustehtävien suorittamiseen, sekä kuinka testaustehtävät tullaan suorittamaan. Suunnitelmaan sisältyy testianalyysin, testien suunnittelun, valmistelun ja suorituksen sekä arviointitehtävien aikatauluttaminen, joko määräpäiviksi (esimerkiksi peräkkäismallisessa kehityksessä) tai joka iteraation tilanteen perusteella (esimerkiksi iteratiivisessa kehityksessä). Suunnitelmaan lisätään tieto, mitä mittareita testauksen seurannassa ja hallinnassa käytetään. (FiSTB 2018, 59.)

Ranorex (2021) jaottelee testaussuunnitelman sisällön seuraaviin osa-alueisiin: odotettu testausaikataulu, tarvittavat resurssit (henkilöstö, fyysiset laitteet, virtuaaliset tai pilvipohjaiset serverit ja työkalut kuten automaatiosovellus), testiympäristö (tietokone, mobiililaite, selain), testattavat ”workflowt” ja sovelluksen visuaalinen design, käytettävyys ja suoritus sekä suunnitellut testaustekniikat (käsikirjoitettu testaus, tutkiva testaus ja käyttäjätestaus). Lisäksi testaussuunnitelmassa tulisi tulla esiin testauksen tavoite, mukaan lukien kriteerit koko testauksen onnistumiselle tai epäonnistumiselle. Software Testing Help-sivuston (Software Testing Help 2021) mukaan, mitä yksityiskohtaisempi testaussuunnitelma on, sitä todennäköisempää testauksen onnistuminen on.

Testaussuunnitelman voi olla tehdä mille vaan alustalle. Excelin käyttö on aika yleistä testauksen etenemisen seurantaan. Testattavien alueiden valinnan voi tehdä monella tapaa. Jos määrittelydokumentit ovat saatavilla, voi niitä käyttää avuksi alueiden valinnassa. Jos määrittelydokumentteja ei ole saatavilla, tai jos ne ovat puutteellisia, voidaan alueiden valintaa varten pitää brainstorming -session. Testattavia alueita voivat olla esimerkiksi visuaalinen design, toiminnallisuudet, sovelluksen suoriutuminen, käytettävyys ja määritysten mukainen toiminta. (Ranorex 2021.)

3.3 Aloitus- ja lopetuskriteerit

Yksi tärkeimmistä testaussuunnitelmaan kirjattavista asioista on kunkin vaiheen aloitus- ja lopetuskriteerit. Aloitusehtojen avulla määritellään, milloin annettu testitehtävä voidaan aloittaa. Näihin tehtäviin voi lukeutua myös sen suunnittelu, milloin testitehtävien suunnittelu ja itse testaus voi kullakin tasolla alkaa. Aloitusehdot määrittelevät kutakin tehtävää edeltävät olosuhteet. Esimerkkinä tyypillisestä aloitusehdosta voi olla, että testaukseen liittyvät vaatimukset, käyttäjätarinat ja mallit ovat testaajien saatavilla. Toinen aloitusehto voi olla, että testausympäristö ja testaustyökalut toimiva ja ovat valmiina käytettäväksi. Ehdoissa voi olla myös maininta, että testissä käytettävä data on saatavilla ja oikeaa. Yksi tyypillinen aloitusehto on myös, että testausta edeltävä toiminto on saatu loppuun ja saavuttanut lopetusehtonsa. (Thompson 2019, luku 5.)

Lopetusehdot määrittelevät sen, milloin kukin testausaktiviteetin voidaan sanoa olevan valmis tai milloin se tulisi lopettaa. Lopetusehdot voidaan määritellä kaikille testausaktiviteeteille tai tietylle testauksen tasolle. Yksi tyypillisistä lopetusehdoista on, että kaikki suunnitellut testaukset on suoritettu. Toinen lopetusehto voi olla, että testauksissa on saavutettu riittävä kattavuus. Lopetusehtoihin voi myös kuulua maininta, että testauksessa löydetyt puutteet ovat määriteltyjen rajojen puitteissa. Lopetusehtoihin voi myös laittaa kustannuksiin tai aikatauluun liittyvän rajan. Lopetusehdot tulisi määritellä mahdollisimman varhaisessa vaiheessa ohjelmiston elinkaarta, joskin niiden tulisi elää ja muuttua projektissa tapahtuvien muutosten mukana. (Thompson 2019, luku 5.)

4 Käyttöliittymä X:n testaussuunnitelma

Esittelen tässä luvussa Käyttöliittymä X:n testaussuunnitelman sisällön. Varsinainen testaussuunnitelma on luottamuksellinen. Testaussuunnitelman sisällössä minun tittelini on testauksen suunnittelija. Testaussuunnitelman pohjaksi otin Software Testing Help-sivustolta ladattavissa olevan testaussuunnitelmapohjan ja muokkasin sitä Yritys Oy:n tarpeisiin sopivaksi.

4.1 Testaussuunnitelman rakenne ja sisältö

Testaussuunnitelman aluksi on johdanto-osio. Johdannossa kerrotaan, että testaussuunnitelma esittelee Käyttöliittymä X:n ja Web raportoinnin testauksen olosuhteet ja testaukseen vaikuttavat asiat. Johdannossa mainitaan myös, että dokumentti sisältää testausstrategian, testauksen suorittamisen strategian sekä kuvauksen siitä, miten testausta hallinnoidaan.

Johdannon jälkeen tulee projektin esittely. Esittelyssä kerrotaan aluksi, että Käyttöliittymä X ja Web Reporting yhdistetään testauksessa yhdeksi kokonaisuudeksi ja niistä käytetään yhdessä nimitystä Käyttöliittymä X. Seuraavaksi esitellään testattava järjestelmä näin:

> Käyttöliittymä X on työajanseurantaan ja kulunvalvontaan tarkoitettu selainpohjainen käyttöliittymä. Käyttöliittymä X on tarkoitettu korvaamaan nykyinen palvelimelle asennettava käyttöliittymä. Tarkoituksena on tässä vaiheessa pitää käyttöliittymän work flow samanlaisena kuin se on nykyisessä käyttöliittymässä ja kehittää vain teknologiaa eteenpäin. Lisäksi käyttöliittymään yhdistetään uusi web-raportoinnin työkalu. (Kuronen 3.12.2020.)

Projektin esittelyssä kuvataan myös projektin vaiheet ja vaiheiden sisällöt. Vaiheita on yhteensä kolme. Ensimmäisessä vaiheessa tehdään nykyisen kaltaiset näytöt niin, että ne toimivat perustasolla, mikä tarkoittaa sitä, että tiedon haku, muokkaus ja tallennus onnistuu. Tässä vaiheessa ei vielä ole mitään erityisiä validointeja. Tätä testaussuunnitelmaa koskeva testaus tapahtuu tämän vaiheen lopussa. Toisessa vaiheessa korjataan esille tulevat virheet ja lisätään käsittelysäännöt. Käsittelysäännöt tarkoittavat esimerkiksi nykyistä Clientia vastaavia sisällöllisiä rajoitteita. Tämän vaiheen jälkeen on pilotointi asiakkaalle. Kolmannessa vaiheessa ryhdytään lisäämään uusia toimintoja ja näkymiä.

4.2 Testausstrategia

Projektin esittelyn jälkeen avataan testausstrategian sisältö. Testausstrategiaan kuuluu testauksen tavoitteiden esitteleminen sekä testauksen oletusten ja periaatteiden esittelyä.

Testauksen tavoitteena on varmistaa Käyttöliittymä X:n toimivuus oletusten mukaisesti sekä löytää toimimattomat ominaisuudet. Testauksessa halutaan myös saada tietoa käyttöliittymän käytettävyydestä. Testauksessa vahvistetaan testauksessa suoritettavien tehtävien avulla käyttöliittymän toiminnallisuuksien oikeellisuus sekä etsitään mahdolliset viat ja korjausta vaativat kohdat. Testauksen seurauksena saadaan tietoa käyttöliittymän testauksen aikaisesta kehitystasosta sekä muutostarpeista käytettävyyden osalta.

Testauksesta on olemassa joitakin yleisiä oletuksia sekä toiminnallisuustestaukselle ja käyttökokemuksen testaukselle ominaisia oletuksia. Yleiset oletukset on esitelty kuviossa 1.

Kuvio 1. Testauksen yleiset oletukset

Toiminnallisuustestaukselle eriteltyinä oletuksina on, että testauksessa käytetään sekä käyttöliittymään etukäteen syötettyä dataa että testauksen aikana syötettyä dataa, ja että testaus koskee ainoastaan Käyttöliittymä X:n Web Client- käyttöliittymää.

Käytettävyyden testaamisen oletuksia kolme. Ensimmäinen oletus on, että testauksen aikana suoritettavat tehtävät tehdään käyttäen Käyttöliittymä X:n Web Client-käyttöliittymää. Toinen oletus on, että käyttäjäkokemuksen testaus suoritetaan etukäteen annettuja tehtäviä suorittamalla sekä tehtävien aikaista käyttöä arvioimalla ja kolmantena oletuksena on, että testauksessa suoritettavat tehtävät on selkeästi esitelty.

Seuraavaksi testaussuunnitelmassa kerrotaan, mitä on testauksen aikana käytettävä data ja millainen on testausympäristö. Testauksessa käytetty data on kehitystiimin syöttämää dataa Käyttöliittymä X:n Web Clientin ja Web raportoinnin käyttöliittymän palvelimelle. Testaustiimillä ei ole pääsyä kyseiselle palvelimelle. Testauksen aikana testaustiimi syöttää käyttöliittymän kautta lisää dataa palvelimelle ja käyttää tätä dataa edelleen testauksen aikana.

Jokainen testaaja suorittaa testauksen haluamallaan selaimella ja merkitsee käytetyn selaimen testauslomakkeelle. Testaajat suorittavat testauksen yksin ilman seurantaa kullekin testaajalle sopivana ajankohtana työajallaan muiden töiden välissä. Tämä vastaa käyttöliittymän oletettua käyttöympäristöä.

4.3 Testauksen laajuus ja tasot

Seuraavaksi testaussuunnitelmassa määritellään toiminnallisuustestauksen ja käytettävyystestauksen laajuudet ja tasot. Toiminnallisuustestauksen tarkoituksena on löytää viat Käyttöliittymä X:n Web Clientin toiminnallisuuksissa. Toiminnallisuustestauksen laajuus on testauksen alkaessa käytössä olevat näytöt ja niiden yhteydet toisiinsa. Testauksesta on rajattu pois käyttöliittymän taustalla toimiva palvelin sekä koodi.

Testauksen suorittaa Yritys Oy:n ohjelmistotuen ja pääkäytön henkilöistä koostuva testaustiimi, yhteensä kahdeksan henkilöä. Testauksen aikataulu on joustava. Ensimmäiselle kierrokselle on annettu testausaikaa kolme-neljä viikkoa. Toiselle kierrokselle on varattu aikaa kolme viikkoa. Testauksen voi suorittaa osissa useampana päivänä.

Testauksessa suoritetaan jokaisella testattavalla näytölle ensimmäisellä kierroksella viisi, toisella kierroksella kuusi, tehtävää, jotka on kirjattu ylös testauslomakkeelle. Testauksen tulokset kirjataan ylös niin ikään kyseiselle lomakkeelle. Testauksessa löytyneiden vikojen korjaus kuuluu kehitystiimille. Testaajat kirjaavat vioista ja parannusehdotuksista tiketit Jira-järjestelmään.

Käytettävyystestauksen tarkoituksena on saada tietoa käyttöliittymän käytettävyydestä. Käytettävyyttä testataan toiminnallisuustestaukseen osallistuvan tiimin toimesta toiminnallisuustestauksen ohessa sekä kahden pelkästään käytettävyyttä testaavien henkilöiden toimesta. Tätä erillistä käytettävyyden testausta varten tehdään erillinen tehtävälomake, jossa on tarkemmin määritellyt tehtävät. Käytettävyyden testauksessa ei käydä läpi kaikkia näyttöjä.

Erillistä käytettävyystestausta järjestetään ainoastaan yksi kierros. Testauksen aikana suoritettavat tehtävät kirjataan erilliselle tehtävälomakkeelle, johon testaaja kirjoittaa tehtävien edistymisen ja tehtävien suorittamisen aikana tekemänsä huomiot käyttöliittymän ulkonäöstä ja käytön helppoudesta. Käytettävyystestauksen ajankohta on testaajan päätettävissä. Koko testi suoritetaan yhdellä kertaa.

4.4 Suoritusstrategia

Testaussuunnitelman seuraavassa osiossa esitellään testauksen suoritusstrategia. Se pitää sisällään testauksen aloitus- ja lopetusehdot, testaussyklien kuvauksen, testauksen mittarit sekä vikojen raportoinnin.

Aloitusehdot tarkoittavat niitä olosuhteita, jonka jälkeen testaus voidaan aloittaa. Käyttöliittymä X:n testauksen aloitusehdot ovat:

- Testauksen suunnittelun valmistuminen
- Testattavien näyttöjen jakaminen testaajien kesken
- Testauslomakkeiden valmistuminen
- Testauslomakkeille pääsyn salliminen lomakkeiden säilytyspaikassa M-filesissa
- Edellisen kierroksen päivitysten valmistuminen

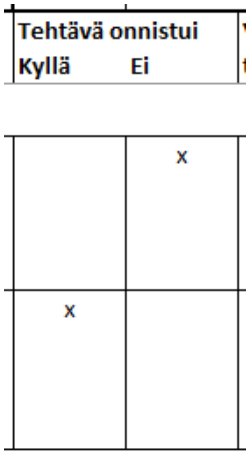
Lopetusehdot ovat ne toivotut olosuhteet, joihin toivotaan pääsevän testauksen seurauksena. Käyttöliittymä X:n testauksen lopetusehdot ovat:

- Kaikki testaustoimenpiteet on kirjattu ylös
- Ei avoimia kriittisiä vikoja
- Kaikki korjattavissa olevat puutteet on korjattu ja raportoitu Jiraan
- Kaikki loput testitiimin huomiot on joko selitetty tai kirjattu ylös jatkokehitysvaiheeseen

Toiminnallisuustestauksessa on kaksi sykliä. Molemmilla kerroilla testataan kaikki näytöt läpi sekä Web Clientin että Web raportoinnin puolelta. Testauksen suorittavat molemmilla kerroilla samat testaajat. Testauksessa noudatetaan samaa osiojakoa molemmissa

sykleissä. Toista testauskierrosta varten annetaan testaajille puhtaat lomakkeet testaustulosten kirjaamista varten. Toiselle testikierrokselle on lisätty yksi tehtävä. Käytettävyyden testauksessa on vain yksi kierros.

Testauksen onnistumisen mittarina käytetään oletetun lopputuloksen toteutumista. Esimerkiksi haut tuottavat odotuksen mukaisen tuloksen, ja väärän tiedon syöttäminen antaa odotusten mukaisen vastauksen. Testauslomakkeeseen merkitään jokaisen tehtävän kohdalle tehtävän läpimenoon liittyen joko ”Kyllä” tai ”Ei” (kuva 1).



Kuva 1. Testauslomakkeen tehtävän läpimenon kuittaus

Mikäli toiminnallisuustestauksen aikana suoritetun tehtävän tekeminen ei onnistu oletusten mukaisesti, tai mikäli tehtävän tekemisen aikana huomataan puutteita, tulee näistä kirjata ylös vikatiketti ohjelmistokehitykselle. Vikojen seurantaan käytetään Jira-järjestelmää. Jokaisesta viasta tai parannusehdotuksesta kirjataan oma tikettinsä. Tiketille määritellään vian taso seuraavasti:

- Tehtävän suorittamisen estävä vika: tyypiksi valitaan ”Bug” ja tasoksi ”Major”
- Osion parannusehdotus: tyypiksi valitaan ”Story” ja tasoksi ”Minor”

Vikojen korjaamisen edistymistä raportoidaan Jira-tiketeille. Tiketin tila muutetaan vian korjauksen edetessä seuraavasti: To do -> In review -> Done -> Closed.

4.5 Testauksen hallinnan prosessi

Viimeisenä pääosiona testaussuunnitelmassa on testauksen hallinnan prosessin kuvaus. Se pitää sisällään testauksen hallinnan työkalujen esittelyn, testauksen suunnitteluprosessin kuvauksen, testauksen suoritusprosessin kuvauksen, riskien analysoinnin, kommunikointisuunnitelman esittelyn sekä eri roolien odotusten esittelyn.

Testauksen hallinnan työkaluja ovat toiminnallisuustestauksen Excel-pohjaiset testauslomakkeet, käytettävyystestauksen Word-lomake, M-files, joka toimii lomakkeiden hallinnointijärjestelmänä, sekä Jira-järjestelmä vikojen raportointiin.

Toiminnallisuustestauksessa suoritettavat tehtävät annetaan testaajille Excel-pohjaisilla testauslomakkeilla. Testauksen aikana suoritetut askeleet ja testauksen löydökset kirjataan kyseisille lomakkeille (liite 2). Web Clientin näytöt on jaettu eri osioihin (liite 1.) ja osiot on ryhmitelty toisiinsa liittyviksi kokonaisuuksiksi siten, että kussakin Excel-tiedostossa on kuudesta kahdeksaan lomaketta. Yksi lomake vastaa aina yhtä Web Clientin näyttöä. Näytön ylälaitaan kirjataan testaajien nimet, testauksen päivämäärä, käytetty selain sekä näytön nimi. Jokaiselle lomakkeelle on kirjattu testauksen aikana suoritettavat tehtävät sekä jätetty tilaa näyttökohtaisille tehtäville. Suoritettavat tehtävät ovat: Tiedon haku, tiedon lisäys, tiedon muokkaus, tiedon poisto, näytöstä toiseen siirtyminen ja väärän tiedon syöttäminen (toisella kierroksella).

Kunkin tehtävän kohdalle merkitään myös mahdollisesta viasta tai parannusehdotuksesta kirjattu Jira-tiketin numero. Kehitystiimi käyttää Jiraa vikojen korjaamisen edistymisen seurantaan. Tikettien edistymisestä tulee ilmoitus tiketin kirjanneelle testaajalle.

Käytettävyystestausta varten on tehty erillinen Word-lomake (Liite 3.), johon kirjattu seitsemän tehtävää. Tehtävien suorittamisen aikaiset askeleet kirjataan lomakkeelle ja niiden onnistumista arvioidaan asteikolla yhdestä neljään. Lomakkeen lopussa on järjestelmän käytettävyyteen liittyen kysymyksiä. Kysymykset ovat seuraavat:

1. Mitä mieltä olet käyttöliittymän ulkonäöstä?
2. Mitä mieltä olit käyttöliittymän käytön helppoudesta?

Testauksen jälkeen lomakkeet tallennetaan M-filesiin ja niiden lukuoikeus annetaan kehitystiimille.

Testauksen suunnitteluprosessin aluksi testauksen suunnittelija ja Web Clientin tuoteomistaja sekä kehitystiimi käyvät läpi testauksen tavoitteet ja pohtivat testauksen aikana suoritettavia tehtäviä. Testauksen suunnittelija luo testauksessa käytettävät testauslomakkeet sekä toiminnallisuustestaukseen että käytettävyystestaukseen. Testauksen suunnittelija kirjaa lomakkeisiin testauksen aikana suoritettavat tehtävät. Testauksen suunnittelija myös jakaa testattavat näytöt eri osioihin (liite 1.) ja jakaa osiot testaajille siten, että kutakin näyttöä testaa kaksi testaajaa. Testitiimi koostuu molemmilla kierroksilla samoista henkilöistä, ja testaajat testaavat samat näytöt molemmilla kierroksilla. Testauksen suunnittelija tallentaa testauslomakkeet M-filesiin ja antaa

lomakkeiden luku- ja muokkausoikeudet testaajille. Testauskierroksen lopussa testauksen suunnittelija antaa lomakkeiden lukuoikeudet myös kehitystiimille.

Ennen testausta tuoteomistaja luo käyttäjätunnukset ja salasanat kaikille testaajille. Testaajat varmistavat, että heillä on pääsy Käyttöliittymä X:ään.

Testauksen suoritusprosessin aluksi kehitystiimi ilmoittaa, milloin käyttöliittymä on valmiina testauskierroksen käynnistymiseen. Testaajille on jaettu tieto siitä, kuka testaa mitäkin näyttöä. Testaajat saavat itse valita itselleen sopivimman ajankohdan testaukseen. Toiveena olisi kuitenkin, että ensimmäiseen kierrokseen kuluisi korkeintaan neljä viikkoa ja toiseen kierrokseen korkeintaan kolmeviikkoa.

Testaajan aloittaessa testaamisen, valitsee hän testattavaa näyttöä koskevan Excel-tiedoston M-filesista ja avaa sen. Lomake on tällöin varattu muokattavaksi kyseiselle testaajalle. Testaaja käy läpi kyseisen näytön lomakkeelle kirjatut tehtävät ja kirjaa tehtävien edistymisen lomakkeelle. Hän merkkaa myös onnistuiko tehtävän suorittaminen vai ei. Mikäli tehtävän suorittaminen ei onnistu, kirjataan viasta tiketti Jira-järjestelmään ja tiketin numero kirjataan kyseisen tehtävän kohdalle lomakkeelle. Testaamisen aikana voidaan ottaa myös kuvakaappauksia ja liittää niitä sekä lomakkeelle että mahdolliseen Jira-tikettiin.

Kun testaaja ei tarvitse lomaketta enää, tai hän pitää testaamisesta taukoa, hän vapauttaa Excel-tiedoston seuraavan testaajan käyttöön. Testauksen suunnittelija seuraa lomakkeiden tilaa M-filesista ja pyytää testaajia sulkemaan auki jääneet tiedostot, mikäli huomaa niiden jääneen varatuksi testauksen keskeytyessä.

Testaukseen liittyvistä riskeistä ja riskejä ennaltaehkäisevistä toimista on koottu lista alla olevaan taulukkoon 2.

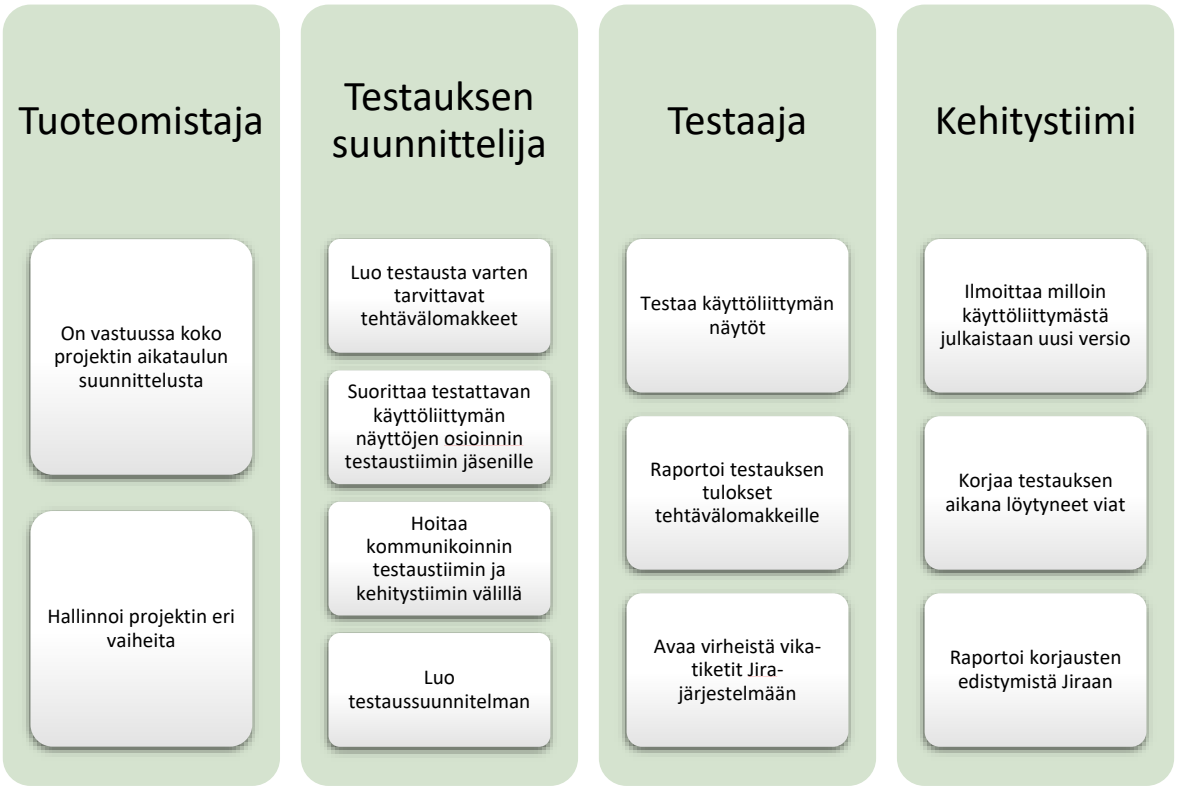
Taulukko 2. Testauksen riskit ja riskien ennaltaehkäisy

Riski	Todennäköisyys	Vaikutus	Ennaltaehkäisevä suunnitelma
Aikataulun viivästyminen	Korkea	Korkea	• Testauksen suorittamiselle annetaan tarpeeksi pitkä aikaväli • Testattava alue osioidaan niin, että yhdelle testaajalle ei tule liikaa testattavaa • Testausta varten tehdään selkeät lomakkeet, joiden avulla testaustulosten kirjaaminen käy sujuvammin
Resurssien saatavuus (Testaajat suorittavat testausta oman työnsä ohella; 1. kierros osuu joulukiireiden ja lomien aikaan, 2. kierros talvilomien aikaan)	Keskiverto	Korkea	• Esimies on tietoinen testauksen viemästä ajasta • Testauksen tarpeellisuutta korostetaan viestinnässä • Lomien vaikutus otetaan huomioon aikatauluarviota tehtäessä
Kaikkia vikoja ei löydetä	Keskiverto	Korkea	• Testitehtävät tehdään niin, että kaikkia osioita testataan moneen kertaan • Testaajat käyttävät aikaa eri testiskenaarioiden läpikäymiseen
Löytyy vika, joka estää testaamisen jatkumisen	Keskiverto	Korkea	• Kehitystiimi suorittaa oman testaamisensa ennen toiminnallisuustestausta
Testaajat eivät pääse kirjautumaan sisään käyttöliittymään	Alhainen	Korkea	• Kaikkien testaajien käyttäjätunnukset testataan ennen testauksen alkamista • Käyttöliittymän toimiminen varmistetaan kehitystiimin toimesta ennen testausta

Seuraavana osana testauksen hallinnan prosessia esitellään kommunikointisuunnitelma. Testauksen aikaista kommunikointia varten luodaan Zoomiin chat-ryhmiä. Yksi ryhmä on testauksen suunnittelijan, tuoteomistajan ja kehitystiimin välinen ryhmä ja toinen ryhmä on testauksen suunnittelijan ja testaustiimin välinen ryhmä. Zoomin chat-ryhmissä jäsenet voivat tiedottaa asioista ja kysyä kysymyksiä testaukseen liittyen.

Testauslomakkeiden päivittäminen tapahtuu M-files tiedonhallintaohjelmassa. M-filesissa näkyy, kenellä lomake on kulloinkin päivitettävänä tai kuka on tehnyt viimeisimmät muutokset dokumenttiin. Sähköpostia ei käytetä lomakkeiden lähettämiseen.

Viimeisenä testaussuunnitelmassa on avattu eri roolien odotuksia. Käyttöliittymä X:n testaukseen liittyen voidaan tunnistaa neljä eri roolia: tuoteomistaja, testauksen suunnittelija, testaaja ja kehitystiimi. Testaussuunnitelmassa on kerrottu, mitä odotuksia kunkin testaukseen osallistuvan osapuolen rooliin kuuluu. Roolien odotukset on listattu kuviossa 2.



Kuvio 2. Roolien odotukset

Seuraavassa luvussa avaan toiminnallisuustestauksen teoreettista taustaa.

5 Ohjelmistokehitys ja testaus

Ohjelmiston testauksella on arvoa ainoastaan silloin, kun se tuottaa arvoa jollekin toiselle osalle ohjelmiston kehityksen elinkaaressa. Se, miten testaus sopii elinkaareen, tulisi olla määritelty testaustrategiassa riippumatta siitä mitä elinkaarimallia noudatetaan. Elinkaarimallilla on kuitenkin vaikutusta siihen, kuinka testaus suoritetaan ja kuinka testaussuunnittelija osallistuu ohjelmiston kehitysprosessiin. (Black 2016, luku 1.2.)

Seuraavaksi esittelen, miten testaus sisältyy eri ohjelmistojen elinkaarimalleihin ja mitä eri tasoja testaamisella voi olla. Lopuksi vielä kerron testaamiseen yleisesti käytettävistä tekniikoista.

5.1 Testaus osana ohjelmiston elinkaarta

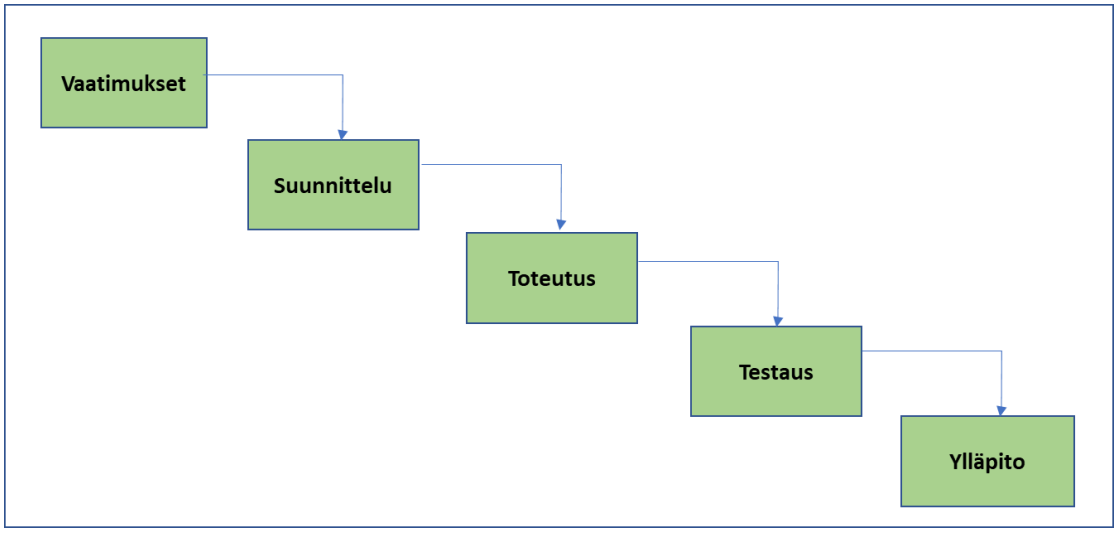
Jokaisella ohjelmistolla on elinkaari ja kaikki ohjelmistoprojektit käyvät läpi samat vaiheet. Nämä vaiheet ovat vaatimusten kerääminen/muodostaminen, suunnittelu, koodaus, testaus, julkaisu ja ylläpito. Tämän elinkaaren läpikäymiseen käytetään yleensä jompaakumpaa projektihallinnan mallia, suunnitelmavetoista mallia tai ketterän kehityksen mallia. (Dooley 2017, luku 2.) FiSTB (FiSTB 2018) käyttää näistä malleista nimitystä peräkkäismalli ja iteratiivinen malli.

Testauksen onnistumisen kannalta testausprosessin limittyminen muihin elinkaaren prosesseihin on avainasemassa (Black 2016, luku 1.2). Ohjelmistokehityksen elinkaarimallissa on lueteltuna ohjelmistokehitysprojektin aikana tehtävät erityyppiset tehtävät, sekä kuvattu, mikä on tehtävien looginen ja ajallinen yhteys toisiinsa. Eri elinkaarimallit vaativat erilaisen lähestymistavan testaukseen. Aikaisen testausperiaatteen mukaisesti testaustehtävien tulisi alkaa elinkaaren alkuvaiheissa riippumatta siitä mitä elinkaarimallia noudatetaan. FiSTB luettelee hyvään testaukseen kuuluvia ominaisuuksia, jotka kuuluvat jokaiseen ohjelmistokehityksen elinkaarimalliin (FiSTB 2018, 25):

- Jokaista kehitystehtävää vastaa jokin testaustehtävä
- Jokaiselle testaustasolle on luotu erityisesti sille tärkeitä testaustavoitteita
- Jokaisen testaustason testien analysointi ja suunnittelu tehdään jo siihen liittyvien toteutustehtävien aikana.
- Testaajat ovat mukana keskusteluissa, joissa määritellään ja tarkennetaan vaatimuksia ja suunnitelmia. He osallistuvat myös tuotosten katselmointiin.

Seuraavaksi avaan, miten testaus sisältyy näihin kahteen eri elinkaarimalliin, eli peräkkäis/suunnitelmavetoiseen malliin ja iteratiiviseen/ketterän kehityksen malliin.

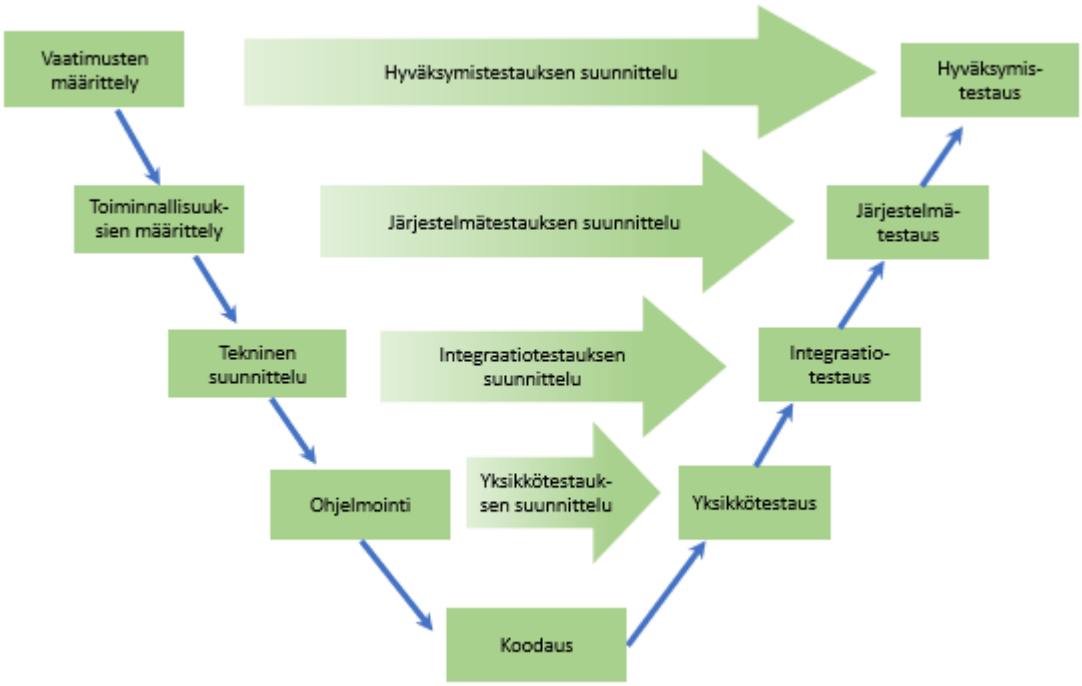
Suunnitelmavetoisista malleista ensimmäinen ja perinteisin on vesiputousmalli (kuvio 3). Vesiputous on hyvin perinteinen metodi, joka on saanut vaikutteita muista aikanaan luoduista prosesseista, joita käytettiin tekniikan alalla. Näille prosesseille oli yhteistä se, että ne muodostuivat jaksoista, jotka seurasivat toisiaan niin, että edellinen jakso suoritettiin aina loppuun ennen seuraavaan siirtymistä. Vesiputousmallissa syntyy paljon dokumentaatiota, koska jokaisesta vaiheesta tehdään hyväksyntädokumentti, mikä tekee vesiputousmallista usein raskaan ja hitaan kehitysmallin. Projektin aikana hyödynnetään monesti Ganttin kaavioita, joiden avulla seurataan projektin etenemistä. (Crookshanks 2015, luku 4.)



Kuvio 3. Vesiputousmalli

V-mallissa testausprosessi on mukana koko projektin ajan. V-mallissa kutakin toteutusvaihetta vastaa oma testaustasonsa (kuvio 4). Testit suoritetaan peräkkäin järjestyksessä, tosin joissain tilanteissa saattaa esiintyä päällekkäisyyksiä. (FiSTB 2018, 25.)

Kuten vesiputousmallissa, V-mallin vasemmassa reunassa on elinkaaren vaiheet toisiaan seuraavassa järjestyksessä. Vaiheet ovat vaatimusten määrittely, toiminnallisuuksien määrittely, tekninen suunnittelu, ohjelmointi ja lopuksi koodaus. Mallin keskiosa kuvastaa sitä, kuinka kussakin vaiheessa tulisi aloittaa tietyn testaustason suunnittelu. Mallin oikeassa reunassa on kaikki testaustasot. Kullakin testaustasolla testaus pohjautuu sitä vastaavan elinkaaren vaiheen määrityksiin. Sen seurauksena testauksessa pystytään keskittymään kulloinkin saatuihin yksityiskohtiin ja toiveena on virheiden löytyminen mahdollisimman aikaisin. V-mallin ollessa peräkkäismalli, kunkin vaiheen tulee olla suoritettuna ennen seuraavaan siirtymistä. Alkuperäisten määrittelyjen tulee olla huolella tehtyjä, koska mikäli asiakkaan toiveita ei ole ymmärretty oikein, ei tätä tiedetä kuin vasta testauksen viimeisessä vaiheessa. (Samaroo 2019, luku 2.)



Kuvio 4. V-malli (Samaroo 2019, kuvaa 2.2 mukaillen)

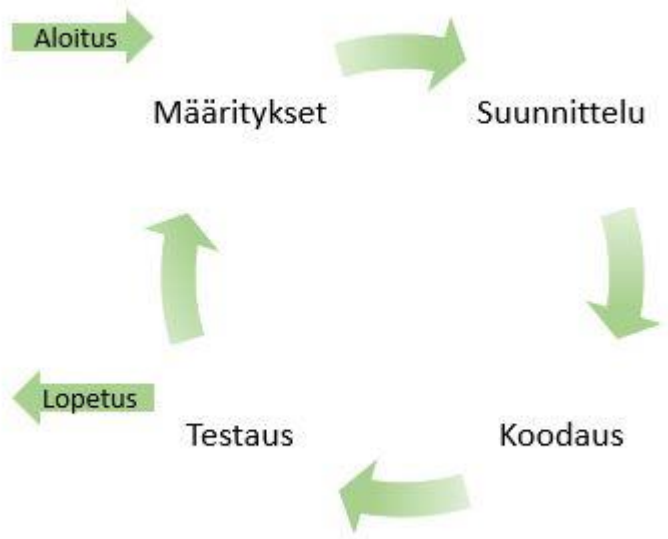
Peräkkäisissä elinkaarimalleissa oletuksena on, että projektitiimi määrittelee vaatimukset projektin alkuvaiheessa ja hallinnoi muutokset vaatimuksiin loppuprojektin aikana. Tällaisissa tapauksissa erillinen testaustiimi voi noudattaa vaatimuspohjaista testausstrategiaa. Testaustiimi voi aloittaa testauksen suunnittelun projektin alkuvaiheessa analysoimalla vaatimuksia ja tunnistamalla testauksen olosuhteet. Testauksen suunnittelun ja vaatimusten analysoinnin seurauksen saatetaan löytää puutteita vaatimuksissa, jolloin testauksesta tulee ennaltaehkäisevä toiminto. Vikojen testaus alkaa myöhemmässä vaiheessa elinkaarta järjestelmätestauksen yhteydessä. (Black 2016, luku 1.2.)

Suunnitelmavetoisessa mallissa on ketteriin malleihin verrattuna selkeämmin määritellyt vaiheet ja enemmän vaatimuksia kunkin vaiheen loppuun suorittamiselle ennen seuraavaan siirtymistä. Näissä malleissa vaaditaan enemmän dokumentaatiota jokaisessa vaiheessa ja varmistusta vaiheen lopputuloksen valmistumisesta. Ketterissä malleissa vaiheet sekoittuvat toisiinsa ja niissä on yleensä vähemmän dokumentaatiota. (Dooley 2017, luku 2.) Seuraavaksi esittelen pääpiirteet ketterän kehityksen malleista.

Ketterän ohjelmakehityksen malleissa pääperiaatteena pidetään, että kehityksen aikaiset muutokset eivät ole poikkeuksellisia, vaan hyvin todennäköisiä. Sen sijaan, että luotettaisiin yksityiskohtaisiin dokumentteihin ja standardoituihin, toistettaviin prosesseihin,

ketterässä mallissa käytetään toistuvaa palautteenantoa ja adaptointia, joiden avulla suunnittelua ja kehitystä ohjataan kohti lopullista päämäärää. Yksi suosituimmista ketteristä viitekehyksistä on Scrum. Scrumissa sovelluksen toimitusta hallinnoidaan lyhyiden ja ennakoitavien iteratiivisten jaksojen avulla. Tavoitteena on jokaisen kehityksen syklin aikana tuottaa joku toimiva toiminto osana lopullista suunnitelmaa. Näistä sykleistä käytetään nimeä ”sprint” ja jokaisen sprintin aikana on tarkoitus tuottaa jotain, mitä asiakas pystyy testaamaan. (Crookshanks 2015, luku 4.)

Inkrementaalisessa mallissa ohjelmiston ominaisuuksia laajennetaan pala kerrallaan. Vaatimusten määrittely, järjestelmän suunnittelu, toteutus ja testaus suoritetaan osissa. Iteratiivisessa kehityksessä ohjelmiston kehitys tapahtuu kierroksina, joiden aikana suoritetaan ohjelmiston ominaisuuksien määrittelyä, suunnittelua, toteutusta ja testausta (kuvio 5). Iteratiivisen kehityksen mallien aikana testausta tapahtuu päällekkäin muiden tehtävien kanssa ja läpi koko kierroksen. Parhaassa tapauksessa ohjelmiston jokainen ominaisuus tulee testatuksi useilla eri testaustasoilla edetessään kohti julkaisua. Monissa iteratiivisissa projekteissa noudatetaan itseorganisoitumista, joka muuttaa myös testaustyön organisointia ja testaajien ja toteuttajien suhteita. (FiSTB 2018, 26.)



Kuvio 5. Iteratiivinen malli (Samaroo 2019, kuvaa 2.3 mukaillen)

Toisin kuin suunnitelmapohjaisissa elinkaarimalleissa, testaajat eivät saa listaa vaatimuksista projektin alussa. Käyttäjätarinat määritellään suunniteltaessa julkaisua, ja ne todennäköisesti muuttuvat. Testaustiimi saa lopulliset käyttäjätarinat jokaisen sprintin alussa. Vaatimusten analysoinnin sijaan, testaajien kannattaa muodostaa riskipohjainen testausstrategia, joka keskittyy laatuun vaikuttavien riskien tunnistamiseen ja priorisointiin. Suunnitellessa iteraatioita, testaajien tulisi olla mukana muokkaamassa käyttäjätarinoita,

koska samalla voidaan hyödyntää testaamisen ennaltaehkäisevää roolia vikojen etsimisessä. Testien suunnittelu ja implementaatio tapahtuu juuri ennen testin suorittamista. Vikojen etsiminen alkaa projektin alkuvaiheessa samalla kun testien suorittaminen joka tasolla alkaa ensimmäisestä sprintistä ja jatkuu toistuvissa, lyhyissä kierroksissa läpi koko projektin. (Black 2016, luku 1.2.)

Yksi iso ero peräkkäismallien ja iteratiivisten mallien välillä on siinä, kuinka nopeasti ohjelmisto saadaan käyttöön. Peräkkäismallien tuottama ohjelmisto sisältää kaikki ominaisuudet, joita siihen on suunniteltu sisällytettävän, mutta ohjelmiston toimittamiseen käyttäjille voi kulua kuukausia tai jopa vuosia. Inkrementaalisten ja iteratiiviset mallien avulla voidaan saada aikaan käyttökelpoinen ohjelmisto viikoissa tai jopa päivissä, joskin kaikkia vaatimuksia vastaavan tuotteen toimittamiseen kuluu kuukausia tai jopa vuosia. (FiSTB 2018, 25–26.)

Ohjelmistokehityksen elinkaarimalleja voidaan myös käyttää yhdessä siten, että eri projektin vaiheessa käytetään eri mallia. V-mallia voidaan käyttää taustajärjestelmien kehittämiseen ja testaamiseen, ja ketteriä kehitysmalleja siinä vaiheessa, kun kehitetään front-end puolta ja testataan käyttöliittymiä sekä niiden toiminnallisuuksia. (FiSTB 2018, 26.)

5.2 Testauksen tasot

Testaustasoja määrittäviä ominaisuuksia ovat testauksen tavoitteet, testitapausten laatimiseen käytetty pohjamateriaali, mitä ollaan testaamassa, tyypilliset häiriöt ja viat sekä valitut lähestymistavat ja vastuut. Testiympäristön tulee olla testaustasolle sopiva. (FiSTB 2018, 27.) Ohjelmiston testaaminen jaetaan lähteestä riippuen kolmeen tai neljään eri tasoon. Dooley (Dooley 2011; Dooley 2017) käyttää kolmea tasoa, jotka ovat yksikkötestaus, integraatiotestaus ja järjestelmätestaus. FiSTB (FiSTB 2018) sekä Filapova ja Vilão (Filipova & Vilão 2018) lisäävät hyväksymistestauksen neljänneksi tasoksi. Nämä eri tasot ovat nähtävissä myös kuviossa 4, jossa esitellään ohjelmistokehityksen V-malli. Seuraavaksi kerron, mitä näillä eri tasoilla tarkoitetaan.

Yksikkötestauksen suorittaa ohjelmiston tekijä. Tällaisessa testauksessa ei yleensä testata käyttöliittymää. Koska testauksessa tehdään yksikkötestausta, testaaja tietää miltä datan on tarkoitus näyttää, ja mitä arvoja testauksesta pitäisi tulla tulokseksi. (Dooley 2011, luku 14; Dooley 2017, luku 16.) Yksikkötestaus tehdään yleensä ohjelmiston kehitysvaiheessa, koska silloin testaus voidaan suorittaa ennen kuin se on liitetty muihin osiin ohjelmistoa. Yksikkötestauksen hyötynä on, että sen avulla löydetään koodin ongelmat jo hyvin aikaisessa vaiheessa kehitystä. Sen ongelmana taas on, että testin

tekijä on yleensä koodin kirjoittaja, ja hänen voi olla vaikeaa muodostaa testiä ohjelmiston oikean käytön mukaiseksi. (Filipova & Vilão 2018, luku 7.)

Yksikkötestauksen tavoitteena on pienentää riskiä, todentaa, että järjestelmä käyttäytyy kuten on suunniteltu ja määritelty, kasvattaa luottamusta komponentin laadusta, löytää mahdollisia vikoja komponentista sekä estää vikoja siirtymästä ylemmille testaustasoille. Yksikkötestaus tehdään erillään muusta järjestelmästä. Sillä voidaan testata toiminnallisuuksia, ei-toiminnallisia ominaisuuksia ja rakenteellisia ominaisuuksia. Tyypillisimmin testauksen kohteena ovat komponentit, yksiköt, moduulit, koodi, luokat ja tietokantamoduulit. Yksikkötestauksessa löydettyjä vikoja voivat olla väärä toiminnallisuus, tietovirtaongelmat ja virheellinen koodi tai logiikka. Vikojen löytymistä ei aina raportoida erikseen, vaan ne korjataan heti löydettäessä. Vikojen raportoinnin avulla voidaan kuitenkin kerätä tärkeää tietoa perusanalyysiä ja prosessikehitystä varten. (FiSTB 2018, 27–28.)

Integraatiotestauksen suorittaa yleensä erillinen testausorganisaatio. Tässä testauksessa testataan eri tasoja ja moduuleja, jotka ovat kytköksissä toisiinsa. Tarkoituksena on testata käyttöliittymiä, jotka hoitavat näitä kytköksiä. Testaajat muodostavat testit tuntien käyttöliittymät, mutta eivät tietäen miten kukin moduuli taustalla on tehty. Voi sanoa, että testaajat ovat tällöin samassa asemassa kuin varsinaiset käyttöliittymän käyttäjät. Integraatiotestauksen tavoitteena on saada selville, miten uusi moduuli toimii aiemman koodin kanssa mahdollisten virheiden löytämiseksi. (Dooley 2011, luku 14; Dooley 2017, luku 16.) ”Big bang”-testaus on yksi integraatiotestauksen muodoista. Siinä odotetaan, että kaikki toisiinsa liittyvät moduulit ovat valmiita ennen kuin aletaan kirjoittamaan testejä ja testaamaan. Tämän testauksen huono puoli on siinä, että testausta joudutaan odottamaan pitkään, jos yksikin moduuleista viivästyy. (Filipova & Vilão 2018, luku 7.)

Integraatiotestauksen tavoitteet ovat samankaltaisia yksikkötestaamisen kanssa, mutta painopiste on rajapinnoissa. Integraatiotestauksen tyypillisiä kohteita ovat alijärjestelmät, tietokannat, infrastruktuuri, rajapinnat ja mikropalvelut. Integraatiotestauksen voi edelleen jakaa kahteen eri tasoon: komponentti-integraatiotestaus ja järjestelmäintegraatiotestaus. Komponentti-integraatiotestauksessa keskitytään komponenttien välisen vuorovaikutuksen ja rajapintojen testaamiseen. Tätä testausta suoritetaan yksikkötestauksen jälkeen, ja se on yleensä automatisoitu. Komponentti-integraatiotestauksen suorittaa yleensä kehittäjät. Komponentti-integraatiotestauksen seurauksena löydettyjä tyypillisiä vikoja ovat aineistoon liittyvät ongelmat, rajapintojen sopivuuteen ja rajapintakutsuihin liittyvät ongelmat, komponenttien väliseen viestintään

liittyvät häiriöt ja niiden käsittelyyn liittyvät ongelmat sekä komponenttien väliseen tiedonkulkuun liittyvät väärät oletukset. (FiSTB 2018, 29.)

Järjestelmäintegraatiotestauksessa testataan järjestelmien, pakettien ja mikropalveluiden välisten vuorovaikutuksia ja rajapintoja. Järjestelmäintegraatiotestausta voidaan käyttää myös ulkoisten organisaatioiden vuorovaikutusten ja rajapintojen testaamiseen. Järjestelmäintegraatiotestausta voidaan suorittaa yhtä aikaa käynnissä olevan järjestelmätestauksen kanssa tai sen jälkeen. Järjestelmäintegraatiotestauksen suorittaa yleensä testaajat, mielellään sellaiset, joilla on käsitys järjestelmän arkkitehtuurista ja jotka ovat olleet mukana integraation suunnittelussa. Järjestelmäintegraatiotestauksen seurauksena löydettyjä vikoja ovat järjestelmien välisten viestirakenteiden epäyhdenmukaisuudet, aineistoon liittyvät ongelmat, rajapintaongelmat, järjestelmien väliseen viestintään liittyvät ongelmat sekä tietoturvasäädösten noudattamiseen liittyvät ongelmat. (FiSTB 2018, 29–30.)

Järjestelmätestauksessa suoritetaan koko järjestelmän testaus. Testauksen suorittaa yleensä ulkopuolinen testausorganisaatio. Testi suunnitellaan tietämättä mitään siitä, miten ohjelman on suunniteltu toimivan tai miten se on kirjoitettu. Järjestelmätestauksessa pyritään varmistamaan, että järjestelmä toimii niin kuin sen on suunniteltu toimivan. (Dooley 2011, luku 14; Dooley 2017, luku 16.)

Järjestelmätestauksen tavoitteena on koko järjestelmän käyttäytymisen ja ominaisuuksien testaaminen. Testauksen aikana käydään läpi kokonaisia toimintoketjuja. Testauksella halutaan varmistaa, että järjestelmä on valmis ja se toimii suunniteltujen ja määriteltyjen vaatimusten mukaisesti. Samalla etsitään mahdollisia vikoja. Järjestelmätestauksen tuloksena syntyy usein tietoa, jonka pohjalta sidosryhmät tekevät päätöksen järjestelmän julkaisusta. Tästä syystä testiympäristön tulisi olla mahdollisimman samanlainen kuin kohde- tai tuotantoympäristö. (FiSTB 2018, 31–32.)

Järjestelmätestauksen kohteita ovat sovellukset, ohjelmistojärjestelmät, käyttöjärjestelmät sekä järjestelmän kokoonpano ja siihen liittyvä aineisto. Testauksen suorittavat yleensä riippumattomat testaajat. Määrittelyihin liittyvät puutteet voivat johtaa vääriin testituloksiin (sekä positiivisiin että negatiivisiin), jotka kuluttavat aikaa ja estävät vikojen löytymisen. Näiden tilanteiden estämiseksi testaajien tulisi osallistua mahdollisimman aikaisessa vaiheessa käyttäjätarinoiden tarkentamiseen tai staattisen testauksen tehtäviin. Järjestelmätestauksessa löydettäviä vikoja ovat tyypillisesti laskutoimitusten virheet, järjestelmän toiminnallinen tai ei-toiminnallinen virheellinen käyttäytyminen, koko

toimintaketjun läpiviemiseen liittyvät häiriöt, tuotantoympäristössä toimiseen liittyvät häiriöt sekä järjestelmä- ja käyttöohjeiden mukaisen käytön häiriöt. (FiSTB 2018, 31–32.)

Hyväksymistestaus (acceptance testing) on viimeinen testaus ennen järjestelmän julkaisemista. Hyväksyntätestauksessa testataan ohjelmistosovelluksen määritysten ja implementoinnin yhteensopivuutta, ottaen huomioon myös liiketaloudelliset vaatimukset ja sopimukset. Testauksen voi suorittaa kahdessa vaiheessa ja kahden eri tiimin voimin. Ensimmäisessä vaiheessa testauksen suorittaa ohjelmistoa kehittävän yrityksen sisäinen tiimi, joka ei kuitenkaan ole mukana itse järjestelmän kehittämisessä. Toisen vaiheen testauksen suorittaa yrityksen ulkopuolinen ryhmä, mahdollisesti asiakasyrityksen sisältä. Siinä tapauksessa kyseessä on Asiakashyväksyntätestaus (Customer Acceptance Testing) tai Käyttäjähyväksyntätestaus (User Acceptance Testing), jos testaajat ovat loppukäyttäjiä. Tästä voidaan käyttää myös nimitystä Beta-testaus. (Filipova & Vilão 2018, luku 7.)

Hyväksymistestauksessa keskitytään järjestelmätestauksen tavoin koko järjestelmän käyttäytymiseen ja ominaisuuksiin. Tavoitteena on varmistaa, että järjestelmä toimii, kuten sen on tarkoitus, se on valmista ja sen toiminnallinen ja ei-toiminnallinen käyttäytyminen ovat määritysten mukaisia. Hyväksymistestauksen perusteella voidaan arvioida, voidaanko järjestelmä julkaista ja luovuttaa loppukäyttäjän käyttöön. Hyväksymistestauksen tavoitteena ei ole niinkään etsiä vikoja, joskin niitä saattaa tulla eteen. (FiSTB 2018, 33–34.)

Hyväksymistestauksen kohteena on muun muassa järjestelmän kokoonpano ja sen aineisto, liiketoimintaprosessit, toipumisjärjestelmät, toiminnalliset ja ylläpitoprosessit, järjestelmän lomakkeet ja raportit sekä tuotantoaineisto. Hyväksymistestaus voi olla muodoltaan käyttäjän hyväksymistestausta, käyttöön soveltuvuuden hyväksymistestausta, alfa- ja betatestausta tai sopimuksiin ja sääntelyihin perustuvaa hyväksymistestausta. Hyväksymistestauksen suorittaa yleensä sidosryhmien edustajat, kuten asiakas, liiketoiminnan käyttäjät, tuoteomistajat tai järjestelmän operaattorit. Hyväksymistestauksessa tyypillisesti löydettyjä vikoja ovat järjestelmän työkulkujen ja vaatimusten ristiriidat, liiketoimintasääntöjen toteutus ei vastaa sovittua, järjestelmä ei ole sopimukseen ja säädöksiin kirjattujen vaatimusten mukainen, ei-toiminnalliset häiriöt, suoritustehokkuuden riittämättömyys kuormitusta kasvattaessa sekä se, että toiminta ei ole tuetun alustan mukaista. (FiSTB 2018, 33–34.)

5.3 Testauksen tekniikat

Testaukseen valittava tekniikka riippuu monesta eri testaukseen liittyvästä tekijästä, kuten järjestelmän tyyppi, vaatimukset, testauksen tavoitteet, resurssit, ohjelmistokehityksen elinkaarimalli sekä aikaisempi kokemus käytettävistä tekniikoista. Yleensä testaajat käyttävät useampaa testaustekniikkaa testitapauksia luodessaan saadakseen parhaan mahdollisen tuloksen testauksesta. Osa tekniikoista soveltuu kaikille testauksen tasoille, kun taas jotkut tekniikat vain johonkin tiettyyn tilanteeseen ja tasolle. FiSTB jakaa testaustekniikat kolmeen luokkaan: mustalaatikkotekniikat, lasilaatikkotekniikat ja kokemuspohjaiset tekniikat. (FiSTB 2018, 50.) Seuraavaksi avaan mitä näillä eri tekniikoilla tarkoitetaan.

Mustalaatikkotekniikat ovat käyttäytymiseen perustuvia tekniikoita. Mustalaatikkotestauksessa ei kiinnitetä huomiota järjestelmän rakenteeseen, vaan siinä keskitytään testattavana olevan järjestelmän syötteisiin ja tuloksiin. Testit muodostetaan pohjamateriaalin analyysin pohjalta, joka voi koostua esimerkiksi ohjelmistovaatimuksista, määrittelyistä, käyttötapauksista ja käyttäjätarinoista. Testauksen kattavuus riippuu siitä, mitä kaikkea osia pohjamateriaalista käytetään testien suunnittelemiseen ja mitä tekniikoita testauksessa sovelletaan. Mustalaatikkotekniikoita voidaan hyödyntää sekä toiminnallisissa että ei-toiminnallisissa testauksissa. Tavoitteena on saada selville, onko vaatimusten ja toteutuksen välissä aukkoja tai vaatimuksissa poikkeamia. (FiSTB 2018, 50–51.)

Mustalaatikkotestaus voi olla joko toiminnallista tai ei-toiminnallista. Toiminnallisessa testauksessa varmistetaan, että järjestelmä on aiemmin määriteltyjen vaatimusten mukainen. Ei-toiminnallisessa testauksessa on tarkoitus tarkistaa, kuinka järjestelmä toimii suurella määrällä dataa ja paineen alla. Toiminnallisessa testauksessa voi olla kyse käyttöliittymän virheistä, virheistä tietokannan rakenteessa tai käyttäytymisessä, vääristä implementoinneista tai puuttuvista toiminnoista. (Filipova & Vilão 2018, luku 7.)

Mustalaatikkotestauksen etuna on, että koska testaajalla ei ole tietoa siitä, mitä on koodattu, pystyy hän paremmin osoittamaan epäjohdonmukaisuudet määrittelyjen ja varsinaisen toteutuksen välillä. Testauksen voi myös suorittaa henkilö, jolla ei ole kokemusta koodaamisesta. Varjopuolena on, että testauksessa on lähes mahdotonta testata jokaista mahdollista navigointipolkua, joten osa mahdollisista puutteista voi jäädä huomaamatta. (Filipova & Vilão 2018, luku 7.)

Lasilaatikkotekniikat ovat rakenteeseen perustuvia tekniikoita. Lasilaatikkotestauksessa analysoidaan testattavana olevan järjestelmän arkkitehtuuri, yksityiskohtaiset

suunnitelmat, sisäinen rakenne tai koodi. Näiden tietojen pohjalta muodostetaan testattavat tilanteet, testitapaukset ja testiaineisto. Odotetut lopputulokset määritellään alkuperäisten määrittelyjen perusteella. Testauksen kattavuus riippuu testaukseen valittavista rakenteen osista. (FiSTB 2018, 50–51.) Dooley (2011;2017) kertoo, että yksikkötestauksessa käytetään juuri lasilaatikkotekniikkaa.

Kokemuspohjaiset tekniikat perustuvat kehittäjien, testaajien ja käyttäjien kokemukseen ja tietämykseen järjestelmän odotetusta käytöstä, sen ympäristöstä, todennäköisistä vioista ja niiden jakaumasta. Näiden kokemusten avulla tehdään testien suunnittelu, valmistelu ja suoritus. Kokemuspohjaisia tekniikoita käytetään usein yhdessä muiden tekniikoiden kanssa. (FiSTB 2018, 51.)

Testaukseen valittava taso ja tekniikka riippuu siis testattavasta ohjelmistosta. Seuraavassa luvussa kerron, kuinka Yritys Oy:n Käyttöliittymä X:n toiminnallisuuksia on suunniteltu testattavaksi.

6 Käyttöliittymän toiminnallisuuksien testaus Yritys Oy:ssä

Yritys Oy:n testattava käyttöliittymä X on työajanseurantaan ja kulunvalvontaan tarkoitettu selainpohjainen käyttöliittymä. Uudesta käyttöliittymästä käytetään nimitystä Web Client. Web Client on tarkoitettu korvaamaan nykyinen palvelimelle asennettava käyttöliittymä sellaisenaan. Tarkoituksena on tässä vaiheessa pitää käyttöliittymän work flow samanlaisena kuin se on nykyisessä käyttöliittymässä ja tehdä vain teknologinen hyppäys. Lisäksi käyttöliittymään yhdistetään uusi web-raportoinnin työkalu. (Kuronen 3.12.2020.)

Ennen testausta on suurin osa työajanseurantaan liittyvistä näytöistä tehty ryppäänä niin, että näyttöjen haut toimivat, tietoja pystytään lisäämään, muokkaamaan, tallentamaan ja poistamaan. Järjestelmään ei ole kuitenkaan vielä tehty kulunvalvonnan osuutta, vaan se tullaan tekemään sen jälkeen, kun työajanseurannan ja raportoinnin osuus on pilotoitu asiakkaalla ja todettu valmiiksi. (Kuronen 3.12.2020.)

Käyttöliittymä X:n kehittäminen on jaettu kolmeen vaiheeseen. Vaiheet ja niiden sisällöt on esitelty taulukossa 3.

Taulukko 3. Käyttöliittymän kehityksen vaiheet ja toimenpiteet

Vaihe	**Vaiheen toimenpiteet**
1	– Tehdään nykyisen kaltaiset näytöt niin, että ne toimivat perustasolla, jolloin tiedon haku ja esille otto, tiedon muokkaus ja tiedon talletus onnistuu – Tässä vaiheessa mitään erityisiä validointeja ei vielä ole
2	– Korjataan esille tulleet virheet – Lisätään käsittelysäännöt eli sisältöä rajoittavat koodaukset – Tämän vaiheen jälkeen pilotointi asiakkaalle
3	– Uusien toimintojen ja näkymien lisääminen

Tässä nyt suoritettavassa testauksessa ollaan vaiheen yksi lopussa. Testauksella halutaan varmistaa, että kaikkien näyttöjen haut toimivat, näyttöihin saadaan lisättyä tietoa, tiedot säilyvät ja niitä pystytään muokkaamaan ja että näytöistä pystytään poistamaan tietoa. Testauksessa löytyvistä virheistä pyydetään tekemään tiketti kehitystiimille. (Kuronen 3.12.2020.)

Testausryhmän koko on 10 henkilöä. Kahdeksan testaajista tutkii toiminnallisuuksia ja etsii mahdollisia virheitä, ja kaksi testaajista suorittaa pelkästään käytettävyyteen liittyvää testausta. Testaajat tekevät testausta oman työnsä ohessa itse valitsemanaan ajankohtana eikä testausta ole valvottu. Testaajien kanssa pidetyssä palaverissa

päätettiin, että jokaista näyttöä testaa kaksi henkilöä. Testauksen edistyminen ja testauksen tulokset eivät sisälly tähän opinnäytetyöhön.

6.1 Testauksen valmistelu

Aloitin testauksen valmistelun käymällä läpi Web Clientin tähän saakka koodatun sisällön ja kirjaamalla ylös kaikki päänäytöt ja näyttöjen alla olevat välilehdet Excel-taulukkoon. Siitä sain käsityksen, kuinka monta testattavaa näyttöä kaikkinensa on. Näyttöjä on 25 kappaletta. Sen jälkeen aloin miettimään, mikä olisi paras testaustapa tälle käyttöliittymälle. Koska testauksen suunnitteluun ei ollut mahdollista käyttää pitkää aikaa, eikä testaukseen ollut käytössä laajoja resursseja, tuli minun miettiä, mikä olisi nopein ja tehokkain tapa saada testaus organisoitua ja tulokset kerättyä yhteen. Jo ennen kuin testauksen suunnittelu tuli minun tehtäväkseni, oli päätetty, ketkä testausta lähtevät tekemään. Näiden käytössä olevien resurssien myötä päätin, että testaus suoritetaan manuaalisesti käyttöliittymän näyttöjä läpikäymällä. Päätin tehdä testaamista varten lomakkeen, josta löytyvät testauksen aikana suoritettavat testitapaukset (käytän näistä myöhemmin nimitystä ”tehtävät”), ja johon testaajat voivat kirjata testauksen aikana suorittamansa työvaiheet ja testauksen tulokset.

Valmistin testaustilanteita varten lomakkeen Exceliin (Liite 2). Muokkasin jokaista näyttöä varten oman lomakepohjan valmiiksi ja ryhmittelin eri näyttöjen lomakkeet viiteen eri tiedostoon aihepiireittäin. Kussakin tiedostossa on viidestä kahdeksaan lomaketta. Jokaisen dokumentin alussa on näyttöjen osiointi, jossa näkyy, kenen on tarkoitus mitäkin näyttöä testata (Liite 1.). Lisäksi jokaisessa tiedostossa on yhdellä välilehdellä malli tehtävälomakkeen täyttöä varten.

Loin ensimmäistä testauskierrosta varten lomakkeeseen viisi testitehtävää tuoteomistajan kanssa käydyn keskustelun pohjalta. Nämä viisi jokaiselle osiolle yhteistä päätehtävää ovat: tiedon haku, tiedon lisääminen, tiedon muokkaus, tiedon poisto sekä ikkunasta toiseen siirtyminen. Toiselle kierrokselle lisäsin vielä tehtävän ”virheellisen tiedon syöttäminen”. Lisäksi jätin tilaa näyttökohtaisille tehtäville. Esimerkki lomakkeella olevasta tehtävästä näkyy kuvassa 2.

Kommentit toiminnoista		Tehtävä onnistui Kyllä	Ei
Tehtävä 1.	**Tiedon haku**		
	Testaaja 1. *(Tähän ensimmäinen testaaja kirjoittaa mitä hän on hakenut ja millä hakusanoilla. Sen jälkeen hän kertoo onnistuiko haku vai ei ja oliko hän tyytyväinen hakuun.)*		x
	Testaaja 2. *(Tähän toinen testaaja kirjoittaa oman hakunsa ja sen kokemukset. Jos ensimmäinen testaaja on raportoinut virheen, ei samaa virhettä raportoida Jiraan uudestaan. Virheen voidaan kuitenkin todeta toistuvan.)*	x	

Kuva 2. Esimerkki testauslomakkeen tehtävästä

Testaajat kirjaavat Excel-lomakkeeseen, mitä he tekevät tehtävien suorittamisen aikana. Jokaisen tehtävän kohdalle merkitään, onnistuiko tehtävän suorittaminen vai jouduttiinko testauksen tuloksena kirjaamaan virhetiketti kehitystiimille Jira-järjestelmään (kuva 3). Mahdollisen tiketin numero lisätään dokumentille sille varattuun kohtaan. Virheen taso pyydetään myös arvioimaan. Tasoja on vain kaksi kehitystiimin toivomuksesta ja tasot noudattavat Jira-järjestelmän tasoja.

	Virheen tasojen selitykset	
	1	Korkea
	2	Alhainen

	Tehtävä onnistui Kyllä	Ei	Virheen taso	Jira-tiketit
		x	1	ABCD-1234
	x			

Kuva 3. Esimerkki virheen raportoinnin merkitsemisestä lomakkeelle.

Pohdimme tuoteomistajan ja testaajien kanssa, mikä olisi kätevin kanava testauslomakkeiden käytölle ja niiden jakamisella kehitystiimille testauksen jälkeen. Päädyimme tallentamaan testauslomakkeet Yritys Oy:llä käytössä olevaan M-files tiedonhallintaohjelmaan.

6.2 Testauksen eteneminen

Testaajan aloittaessa testaamisen, valitsee hän testattavaa näyttöä koskevan Excel-tiedoston M-filesista ja avaa sen. Tiedosto on tällöin varattu muokattavaksi kyseiselle testaajalle. Testaaja käy läpi kyseisen näytön lomakkeelle kirjatut tehtävät ja kirjaa tehtävien edistymisen lomakkeelle. Hän merkkaa myös onnistuiko tehtävän suorittaminen vai ei. Mikäli tehtävän suorittaminen ei onnistu, kirjataan viasta tiketti Jira-järjestelmään ja tiketin numero kirjataan kyseisen tehtävän kohdalle lomakkeelle. Testaamisen aikana voidaan ottaa myös kuvakaappauksia ja liittää niitä sekä lomakkeelle että mahdolliseen Jira-tikettiin.

Kun testaaja ei tarvitse lomaketta enää, tai hän pitää testaamisesta taukoa, hän vapauttaa Excel-tiedoston seuraavan testaajan käyttöön. Testauksen suunnittelija seuraa lomakkeiden tilaa M-filesista ja pyytää testaajia sulkemaan auki jääneet tiedostot, mikäli huomaa niiden jääneen varatuksi testauksen keskeytyessä.

Toiminnallisuustestausta käydään läpi kaksi kierrosta. Molempien kierrosten aikana testataan kaikki näytöt läpi sekä Web Clientin että Web raportoinnin puolelta. Kunkin näytön testauksen suorittaa molemmilla kerroilla sama testaaja, jolloin ensimmäisellä kierroksella tehtyjä havaintoja ja niiden kehitystä on helpompi seurata. Toista testauskierrosta varten testaajille annetaan tyhjät lomakkeet testaustulosten kirjaamista varten.

Kehitystiimi ilmoittaa, kun käyttöliittymä on valmiina testauskierroksen käynnistymiseen. Testaajille on jaettu tieto siitä, kuka testaa mitäkin näyttöä osiointi-dokumentin avulla. Testaajat saavat itse valita itselleen sopivimman ajankohdan testaukseen. Toiveena olisi kuitenkin, että ensimmäiseen kierrokseen kuluisi korkeintaan neljä viikkoa ja toiseen kierrokseen korkeintaan kolme viikkoa.

6.3 Käytetyt tekniikat ja testauksen taso

Testaustekniikkana tässä käyttöliittymän testaamisessa käytetään sekä toiminnallista mustalaatikkotekniikkaa että kokemuspohjaista tekniikkaa. Kuten FiSTB (FiSTB 2018) kertoi mustalaatikkotekniikasta, tämän tyypin tekniikassa keskitytään järjestelmään syötettäviin syötteisiin ja niiden tuottamiin tuloksiin. Mustalaatikkotekniikkaan kuuluvasti, Käyttöliittymä X:n testaajilla ei ole mahdollisuutta nähdä käyttöliittymän koodia eikä siltä osin tietoa siitä, miten liittymä on tarkoitettu toimimaan. Testauksessa käytetty data on kehitystiimin syöttämää dataa Käyttöliittymä X:n Web Clientin ja Web raportoinnin käyttöliittymän palvelimelle. Testaustiimillä ei ole pääsyä kyseiselle palvelimelle.

Testauksen aikana testaustiimi syöttää käyttöliittymän kautta lisää dataa palvelimelle ja käyttää tätä dataa edelleen testauksen aikana. Testaajat arvioivat sitä, reagoiko käyttöliittymä niin kuin sen kuuluisi siihen syötettyihin tietoihin ilman, että he seuraisivat, miten järjestelmä on koodattu reagoimaan.

Testaajat ovat tottuneet käyttämään päivittäisessä työssään sitä käyttöliittymää, johon Käyttöliittymä X pohjautuu. Sen vuoksi testauksessa hyödynnetään myös kokemukseen pohjautuvaa tekniikkaa. Testauksen aikana testaajat käyttävät Käyttöliittymä X:ää niin kuin ovat tottuneet käyttämään nykyistä clientia, ja varmistavat, että Käyttöliittymä X reagoi samalla tavalla, kuin nykyinen client niihin komentoihin, joihin sen kuuluukin reagoida vastaavalla tavalla.

Testaustasoltaan nyt suunniteltu testaus on järjestelmätestausta. Ennen tätä testausvaihetta kehitystiimi on tehnyt omat yksikkötestauksensa koodiin liittyen ja käyttöliittymän yhteys taustalla olevaan palvelimeen on myös testattu integraatiotestauksessa kehitystiimin toimesta. Kuten FiSTB (FiSTB 2018) mainitsee, on tässä testauksessa tarkoitus testata koko järjestelmän toimivuutta ja etsiä samalla mahdollisia vikoja. Tässä testauksessa testausympäristö on niin aito kuin mahdollista, sillä testaajat ovat henkilöitä, jotka tulevat käyttämään käyttöliittymää myös omassa työssään ja suorittavat testauksen oman työnsä ohella. Testauksen onnistumisen mittarina käytetään oletetun lopputuloksen toteutumista eli esimerkiksi haut tuottavat odotuksen mukaisen tuloksen, ja väärän tiedon syöttäminen antaa odotusten mukaisen vastauksen.

Toiminnallisuustestauksen lisäksi nyt suunnitellussa testauksessa suoritetaan Käyttöliittymä X:n käytettävyyden testausta. Seuraavassa luvussa avaan käytettävyyteen liittyvää teoriataustaa, ja luvussa 8 Käyttöliittymä X:n käytettävyyden testausta.

7 Käyttöliittymän käytettävyys ja sen testaus

Uuden järjestelmän kehitys lähtee aina siitä ajatuksesta, että järjestelmälle on tarvetta. Uuden järjestelmän toivotaan auttavan käyttäjäänsä hoitamaan tarvittava toiminto aikaisempaa tehokkaammalla ja helpommalla tavalla. Ja olisi tietenkin hyvä, jos järjestelmän käyttämisestä jäisi sen käyttäjälle myös hyvä mielikuva, joka saisi käyttäjän palaamaan käyttämään järjestelmää myös uudestaan. Tässä kohtaa tulee kuvaan järjestelmän käyttöliittymän käytettävyys.

7.1 Käyttöliittymän suunnittelu

Ennen käyttöliittymiä, tietokoneiden käyttö perustui pitkälti muistettaviin komentoihin, joita syötettiin komentokehotteeseen. Käyttöliittymien tultua käyttöön, ohjelmien käyttö ei perustu enää niinkään muistettaviin komentoihin, kuin tunnistettaviin toimintoihin. Käyttöliittymien etuna on myös peruuttaminen, jolloin virheellisen toiminnon sattuessa käytön jatkaminen on helpompaa. Yleisimmin käytössä oleva käyttöliittymätyyppi on WIMP käyttöliittymä (esimerkiksi Windows tai macOS). WIMP on lyhenne ja tulee sanoista ikkunat (windows), kuvakkeet (icons), valikot (menus) ja osoittimet (pointers). Ikkuna on keino, jolla laitteen graafisen näytön resurssit voidaan jakaa monelle sovellukselle yhtä aikaa. Kuvake on kuva tai symboli, joka edustaa jotain asiaa tai sovellusta. Valikko on lista komennoista tai vaihtoehdoista, joista voi valita. Osoittimilla tarkoitetaan välinettä, jonka avulla käyttöliittymään ohjataan, eli yleisimmin hiirtä tai sormea. (Benyon 2019, 290.)

Käyttöliittymä koostuu kaikesta niistä järjestelmän osista, jonka kanssa käyttäjä joutuu kosketuksiin, oli se sitten fyysisesti, aistillisesti tai käsitteellisesti. Käyttöliittymän suunnittelussa tulee ottaa jokainen näistä kolmesta näkökulmasta huomioon, sillä niiden yhteisvaikutuksesta muodostuu kokemus käyttöliittymän käytöstä. (Benyon 2019, 288.)

Ihmiset ovat fyysisesti vuorovaikutuksessa järjestelmän kanssa esimerkiksi mahdollisten painettavien nappien, kosketettavan näytön tai liikuteltavan ja klikattavan hiiren kursorin tai vieritettävän pyörän välityksellä. Aistien näkökulmasta suunnitellessa tulee ajatella, että ihmiset ovat järjestelmän kanssa vuorovaikutuksessa näkemänsä, kuulemansa ja kosketuksen kautta. Liittymän visuaalinen suunnittelu kohdistuu siihen, mitä käyttäjät näkevät ja huomaavat näytöllä. Nappien tulee olla tarpeeksi suuria, jotta ne huomataan ja niiden tulee olla nimettyjä niin, että niiden tarkoitus ymmärretään. Käyttäjiä tulee myös ohjeistaa, jotta käyttöliittymän käytöstä ei tule epäselvyyttä. Sellaisten näyttöjen sisällöt, joissa on paljon tietoa, tulee miettiä tarkkaan, jotta tiedon merkitys tulee selväksi. (Benyon 2019, 288.)

Graafisen käyttöliittymän suunnittelussa visuaalinen suunnittelu on olennaisessa roolissa. Suunnittelijoiden tulee ottaa huomioon, että käyttäjillä on jo uutta käyttöliittymää käyttäessään mielikuva siitä, mitä toimintoja minkäkin näköinen napin tai symbolin alta tapahtuu. Uusille, ei niin yleisille toiminnoille kannattaa valita erilainen symboli, kuin yleisimmin käytössä olevat symbolit. (Benyon 2019, 302.)

Graafinen.com (2015) on luetellut kymmenen muistisääntöä toimivan käyttöliittymän suunnitteluun. Tärkeimpänä sääntönä sivusto mainitsee, että on tärkeää tuntea käyttöliittymän käyttäjä, eli henkilöt, joille käyttöliittymää suunnittelee. Suunnittelun pohjana tulee olla ajatus siitä, että käyttäjä pystyy käyttöliittymän avulla tekemään mahdollisimman helposti ja ilman monimutkaisia toimintoja sen, mitä hän on käyttöliittymällä tullutkin tekemään. Toinen sivuston tarjoama vihje on, että käyttöliittymän kannattaa noudattaa samoja periaatteita, kuin on yleisestikin käytössä internetin suosituimmilla sivustoilla. Näin käyttäjälle tulee sellainen olo, että hän hallitsee käyttöliittymän käytön jo ensimmäisellä kerralla. Käyttöliittymän tulisi myös olla johdonmukainen, eli nappien ja valikkojen tulee näyttää samalta läpi käyttöliittymän ja niiden toimintojen tulee pysyä samoina.

Käyttöliittymän tulisi olla johdonmukainen myös siinä, että tärkeimmät asiat ovat helposti löydettävissä ja ne tulevat esiin käyttöliittymässä monessa kohdassa. Tähän vaikuttavat käyttöliittymässä käytettyjen elementtien koot, värit ja sijoittelu. On myös hyvä, jos käyttöliittymä antaa palautetta käyttäjälle sekä onnistuneista tilanteista että väärin menneistä painalluksista. Palaute voi tulla joko visuaalisina reaktioina tai tekstinä. Tärkeintä olisi, että käyttäjälle ei jää epäselväksi onnistuiko hän toiminnassaan. Käyttöliittymän tulisi kuitenkin myös sallia käyttäjän virheet ja antaa mahdollisuuden palata takaisin päin tai muuttaa esimerkiksi lomakkeelle syötettyä tietoa. (graafinen.com 2015.)

Käyttöliittymän selkeän visuaalisen ilmeen lisäksi myös kirjoitetulla tekstillä on vaikutusta siihen, mikä on käyttöliittymän käyttökokemus. Selkeää, yksinkertaista ja käyttäjälle hyödyllistä tekstiä sisältävä käyttöliittymä on miellyttävä käyttää ja antaa käyttäjälle tunteen, että hän puhuu samaa kieltä käyttöliittymän kanssa. Käyttöliittymäsuunnittelun yksi tärkeimmistä oivalluksista on se, että mitä yksinkertaisempi ja huomaamattomampi käyttöliittymä on, sitä parempi käyttökokemus. Suunnittelussa tulee siten pitää mielessä, että aina tulisi miettiä, onko kaikki toiminnot tarpeellisia, vai voisiko käyttöliittymän tehdä vielä yksinkertaisemmaksi. Suunnittelussa parhaimmat tulokset löytyvät kokeilemalla ja pyytämällä palautetta käyttäjiltä. (graafinen.com 2015.)

7.2 Käytettävyyden suunnittelu

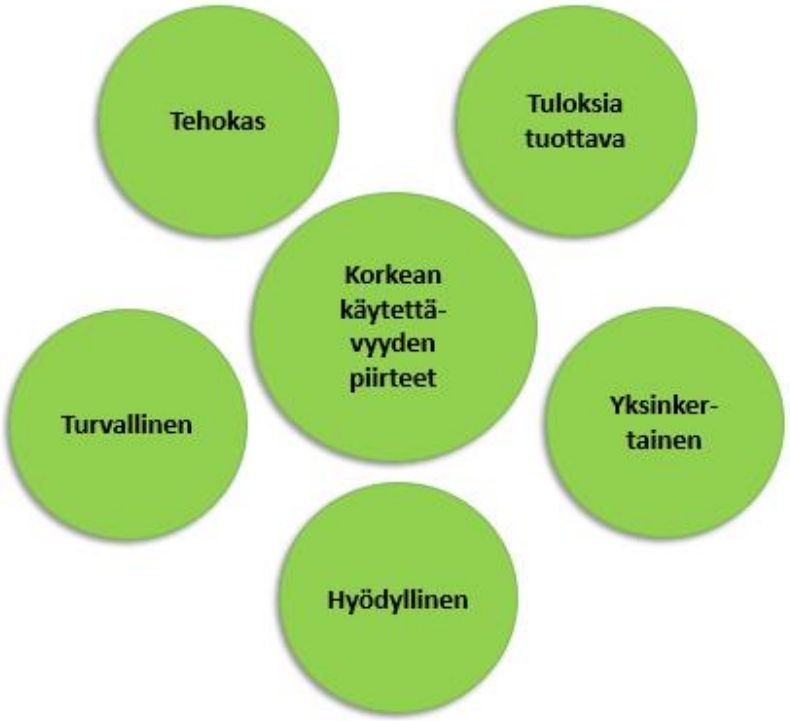
Tuotteen käytettävyydestä on muodostettu ISO standardi 9241–11, jonka mukaan tuotteen käytettävyydellä tarkoitetaan tuotteen tarkoituksenmukaisuutta, sen tehokkuutta ja tuotteen käyttäjien tyytyväisyyttä määriteltyjen tavoitteiden saavuttamiseen tietyssä ympäristössä. (Niemelä 2020.)

Käytettävyydellä tarkoitetaan sitä, miten helppoa käyttöliittymän käyttäminen on. Se on käyttöliittymän sisältämä laatu. ISO 9241-11 mukaan, käytettävyys on se laajuus, jolla tuotetta voidaan käyttää tiettyjen käyttäjien toimesta saavuttaakseen tietyt tavoitteet odotetusti, tehokkaasti ja tyydyttävästi tietyssä kontekstissa. Standardista tulee esille se, että pelkkä käyttöliittymän tehokkuus ja nopeus ei ole tärkeää, vaan myös se tunne, mikä käyttäjälle jää liittymän käytöstä. Käyttöliittymän käytettävyyden arviointi pohjautuu siten sekä objektiivisiin mittareihin, kuten kuinka usein tehtävä onnistuu tai kuinka paljon aikaa tehtävän suorittamiseen kuluu, mutta myös subjektiiviseen näkemykseen yksilön käyttöliittymän käytöstä. Tähän vaikuttaa liittymän suunnittelu, värimaailma ja käytön jälkeinen tunne onnistumisesta. Kun tiedetään, mistä käyttäjät pitävät tai eivät pidä, ja mikä tuotteen käytössä on helppoa tai vaikeaa, pystytään myös arvioimaan käyttöliittymän käytettävyyttä, löydetään korjattavat alueet ja vältettävät mallit ja pystytään vertailemaan käyttöliittymään kilpaileviin liittymiin. (De Bleecker & Okoroji 2018.)

Käytettävyyden merkitystä on korostettu käyttöliittymien suunnittelussa parinkymmenen vuoden ajan, siitä asti, kun tietokoneiden ääressä tehtävä työ lisääntyi. Siinä vaiheessa huomattiin, että käyttöliittymien suunnittelulla oli vaikutusta sekä tehokkuuteen että työtyytyväisyyteen. Alettiin kiinnittää huomiota elementtien sijoitteluun näytöllä, värityksiin ja osiosta toiseen siirtymisen helppouteen. Haastavaksi käytettävyyden suunnittelun on tehnyt se, että käytössä on hyvin erilaisia järjestelmiä ja vaadittavia toimintoja on lukuisia. Myös käyttöliittymiä käyttäviä henkilöitä on hyvin paljon ja kaikilla on omat mieltymyksensä. Sen vuoksi on vaikeaa suunnitelma järjestelmää, joka olisi kaikkien mielestä hyvä. (Filenius 2015, osa 1.)

Benyon (2019) on listannut piirteitä, jotka ovat yleisiä järjestelmillä, joiden käytettävyys on korkealla tasolla. Nämä piirteet näkyvät kuviossa 6. Ensimmäiseksi hän mainitsee, että järjestelmä on tehokas ja sitä käyttävät ihmiset pystyvät tekemään asioita käyttäen sopivan määrän paneutumista. Toiseksi järjestelmä on tuloksia tuottava, mikä tarkoittaa, että se sisältää oikeat toiminnot ja tietosisällön, ja ne on organisoitu oikealla tavalla. Kolmanneksi piirteeksi Benyon mainitsee järjestelmän yksinkertaisuuden, joka tarkoittaa, että järjestelmän käyttö helppo oppia ja sen käytön muistaa myös hetken kuluttua. Seuraavaksi tärkeäksi piirteeksi hän nostaa esiin järjestelmän turvallisuuden, jolloin

järjestelmää on turvallista käyttää sillä sisällöllä mihin sitä on tarkoitus käyttää. Viimeinen käytettävyyttä nostava piirre on järjestelmän hyödyllisyys, jolla viitataan siihen, että järjestelmä tekee ne asiat, jota varten sitä halutaan käyttää.



Kuvio 6. Korkean käytettävyyden piirteet

Käytettävyyden saavuttamiseksi suunnitteluun tulee ottaa ihmiskeskeinen lähestymistapa ja testaus tulee ottaa keskeiseksi osaksi suunnittelua. Suunnittelun tulee perustua neljään pääperiaatteeseen: painopiste käyttäjään ja tehtäviin alusta asti, empiirinen mittaaminen, iteratiivinen suunnittelu ja integroitu käytettävyys. (Benyon 2019, 110.)

7.3 Käytettävyyden testauksen tavoitteet

Käytettävyystestausta tehdään, jotta voidaan seurata kuinka käyttäjät todella käyttävät käyttöliittymää, sen sijaan, että vaan oletettaisiin, kuinka sitä tullaan käyttämään. Testauksen tavoitteena on löytää kohdat, jossa käyttö kompastelee ja samalla kuulla käyttäjien mielipiteitä käyttöliittymästä, mikä on heidän mielestänsä hyvää, ymmärtää käyttäjien päätöksiä ja lopulta käyttää kerättyä tietoa tuotteen parantamiseksi. (De Bleecker & Okoroji 2018.)

Vaikka projektia suunniteltaisiin ja valmisteltaisiin kuinka huolella tahansa, saatetaan silti julkaista järjestelmä, joka on täynnä virheitä tai jossa on toimimattomia osia. Syynä on usein se, että käyttöliittymän testaus on ollut vajaata. Monessa tapauksessa testauksen

suorittaa yhä ne henkilöt, jotka ovat vastuussa käyttöliittymän suunnittelusta ja toteutuksesta, ensin itse koodaajat toimittajapäässä ja sitten määrittelyyn osallistuneet henkilöt asiakkaan päässä. Testaus on näissä tapauksissa vajaata, koska testaavat henkilöt ovat tietoisia siitä, mihin käyttöliittymän olisi tarkoitus pystyä. Näitäkin testauksia tulee toki tehdä, mutta tärkeää on myös lisätä testaukseen täysin ulkopuolisia henkilöitä, jotka ovat potentiaalisia loppukäyttäjiä. Tärkeää on, että käyttöliittymän lopputestaajalla ei ole käsitystä siitä, mihin projektin aikana on pyritty ja mitä kehittämisen aikana on tapahtunut. Tällaiseen testaukseen ei tarvita montaa henkilöä, jo yhdenkin käyttäjän kommentit auttavat aikaisemmin huomaamatta jääneiden puutteiden ja virheiden parantamisessa. (Filenius 2015, osa 6.)

Käytettävyystestausta voidaan käyttää tehokkaasti etenkin iteratiivisessa ohjelmistokehityksessä, jossa tuotetta pyritään kehittämään eteenpäin asteittain. Käytettävyystestausta voidaan käyttää jatkona toimintalogiikan tarkistuksen ja käyttöliittymän virheiden etsinnän jälkeen. Siinä vaiheessa käyttäjät pääsevät mukaan tuotteen kehittelyyn. Käytettävyystestausta voidaan suorittaa myös järjestelmälle, joka on jo käytössä siinä vaiheessa, kun järjestelmää halutaan uusia ja parantaa. Testauksen avulla vanhasta järjestelmästä voidaan kerätä hyväksi havaitut ominaisuudet ja säilyttää ne uudessa versiossa. Toisaalta taas huonoksi havaittuja ominaisuuksia voidaan karsia pois. (Niemelä 2020.)

Käyttökokemusta voidaan arvioida kolmelta eri tasolta: asiantuntija-, osallistuja- tai data analyysitasolta. Asiantuntijoiden tekemissä arvioinneissa löydetään yleensä käytettävyyteen tai käyttökokemukseen liittyvät ongelmat nopeasti, mutta heiltä saattaa mennä ohi yksityiskohdat, jotka oikeiden käyttäjien mielestä ovat hankalia. Osallistujametodeja tulee siksi käyttää jossain vaiheessa kehitysprosessia, jotta käyttäjiltä saadaan oikeaa palautetta. Molempia asiantuntija- ja osallistujatason metodeja voidaan käyttää valvotuissa ympäristöissä, kuten käytettävyyslaboratorioissa, tai sitten arviointeja voidaan suorittaa tilanteissa, joissa kokemus on realistisempaa. Jos oikeita käyttäjiä ei ole mahdollista saada arvioimaan käytettävyyttä, suunnittelijat voivat kehitellä käyttäjäroolit, jotka kerrotaan käytettävyyden testaajille. Data-analyysiä voidaan käyttää hyödyksi käytettävyyden arvioinnissa sen jälkeen, kun järjestelmä tai palvelu on julkaistu. (Benyon 2019, 239–240.)

Käytettävyystestaus on keskeinen tapa sen määrittämisessä saavuttavatko käyttäjät tavoitteensa. Suurin osa käytettävyystesteistä sisältää jonkin näköisen yhdistelmän näistä osa-alueista: läpimenon määrä, virheet, tehtävien läpimenoaika, tehtävien suorittamisen

tyytyväisyysaste, avun saannin mahdollisuus ja lista käytettävyysongelmista, kuten niiden yleisyys ja vakavuusaste. (Lewis & Sauro 2012, luku 2.)

Käytettävyyden testaamisen tavoitteena on tunnistaa syyt, jotka estävät järjestelmän sujuvan ja luotettavan käyttämisen. Perinteinen tapa testata käytettävyyttä on käyttää testejä, jotka on luonut käytettävyyden asiantuntijat. Alimmalla testaamisen tasolla tavoitteena on vahvistaa, että asiakkaan edustajat pystyvät tekemään järjestelmällä sen, mitä järjestelmän suunnittelijat ovat tarkoittaneet järjestelmällä tehtävän. Kehityksen ja tuotannon tasolla tavoitteena on ymmärtää miten loppukäyttäjät käyttävät järjestelmää silloin, kun he tekevät oikeita töitään. (Kenett, Harel & Ruggeri 2018, 358.) Seuraavaksi kerron millaisia menetelmiä käytettävyyden testaukseen voidaan käyttää.

7.4 Käytettävyyden testauksen menetelmiä

Käytettävyystestejä on yleisesti katsoen kahdenlaisia: käytettävyysongelmien etsiminen ja ratkaiseminen (formatiivinen) sekä sovelluksen käytettävyyden kuvaaminen eri mittareita apuna käyttäen **(**summatiivinen).

Formatiivisessa testauksessa kuvataan ongelmaa ja muodostetaan parannusehdotuksia. Vaikka tarkoituksena onkin löytää ja korjata mahdollisimman monta ongelmaa, voidaan silti myös hyödyntää kvantitatiivista datakeruuta vertailemalla ongelmien toistuvuutta ja vakavuutta. Samalla voidaan myös käyttää mittareita määrittämään, kuinka kauan käyttäjillä kesti suorittaa haluttu tehtävä ja antaa arvosana tehtävän suorittamisen helppoudelle. (Lewis & Sauro 2012, luku 2.)

Käytettävyysongelmien testauksen voi aloittaa siinä vaiheessa, kun järjestelmästä on saatu tehtyä toimiva versio. Kaikkien ominaisuuksien ei vielä tarvitse olla toiminnassa, mutta niiden osioiden, joiden toimintoja testataan, tulisi toimia tarkoitetulla tavalla. Tällaista testausta kutsutaan formatiiviseksi testaamiseksi, koska se auttaa muokkaamaan luonnosta tiettyyn suuntaan. (Benyon 2019, 241–242.)

Summatiivista arviointia käytetään silloin, kun halutaan testata ja esitellä valmiin järjestelmän suoritusta. Mittarina voidaan käyttää yrityksen sisäisiä mittareita, ISO 9241:n mukaisia käytettävyysstandardeja tai asiakkaan määrittelemiä vaatimuksia järjestelmän käytettävyydestä (esimerkiksi toiminnon suorittamiseen kuluva aika). (Benyon 2019, 242.) Tyypillisesti on olemassa kahdenlaisia summatiivisia testejä: benchmark ja vertailevia. Benchmark -testauksessa on tarkoitus selvittää, kuinka käytettävä sovellus on verrattuna johonkin määrättyyn kiintopisteeseen. Benchmark-testit antavat vihjeitä siitä mitä käyttöliittymässä tulisi korjata ja myös antaa pohjan, jota vasten verrata jatkokehityksessä

suunniteltuja muutoksia. Vertailevassa summatiivisessa testauksessa on mukana useampi sovellus tai käyttöliittymä. Tässä testauksessa voidaan verrata esimerkiksi nykyistä versiota aikaisempaan versioon tuotteesta tai kilpailevaan tuotteeseen. Vertailevassa testauksessa samat käyttäjät voivat yrittää suorittaa samoja tehtäviä eri tuotteilla tai useampi testaajaryhmä testaa jokaista tuotetta. (Lewis & Sauro 2012, luku 2.)

A/B testauksessa suunnittelijat voivat testata kahden eri suunnitelman välillä siinä vaiheessa, kun järjestelmä on jo toiminnassa. Ensiksi julkaistuun suunnitelmaan tehdään kontrolloituja muutoksia ja näiden eri suunnitelmien toiminnasta kerätystä datasta analysoidaan, kumpi suunnitelma on toimivampi. (Benyon 2019, 241, 246.)

Osallistavassa suunnittelumallissa sidosryhmät auttavat suunnittelijoita asettamaan tavoitteet testaukselle. Sidosryhmien käyttäminen apuna suunnitteluvaiheessa on hyödyllistä lopullisen käyttöönoton ja käytön näkökulmasta. (Benyon 2019, 242.)

Käyttökokemuksen testauksessa data-analyysiä käytetään hyväksi järjestelmän suorituksen ja käyttäjien käyttäytymisen mittaamisen avulla. Data-analyysiä tarjoavat palvelut voivat kerätä ja antaa sivustojen ja sovellusten suunnittelijoille dataa palvelun käytön aikaisista tapahtumista. Sen avulla voidaan selvittää, mitä kautta sivustolle on tultu ja mitä laitetta käyttäjä käyttää, mitä sivustolla on tehty, kauanko sivustolla on vietetty aikaa ja missä järjestyksessä sivuja selattiin. Lisäksi voidaan analysoida, onko sivustolla käyty enemmän johonkin tiettyyn aikaan päivästä, kuukaudesta tai vuodesta. Yksi hyödyllinen data-analyysiä tarjoavien toimittajien työkaluista on kuumakartta. Kuumakartta näyttää mitä osaa sivusta käyttäjät klikkaavat eniten tai mikäli käytössä on silmien liikettä seuraava sovellus, mihin käyttäjän katse kohdistuu sivulla. (Benyon 2019, 242–246.)

Datan avulla tehtävässä käytettävyyden testaamisessa voidaan käyttää apuna käytettävyyden ongelman indikaattoreita (Usability Problem Indicators, UPIs). ”UPI” on ilmentymä, joka ennakoi potentiaalista käyttäjäongelmaa. Tyypillisiä ”UPI”:ta ovat käyttäjän tekemät toiminnot, joissa hän palaa takaisin edelliseen toimintoon, esimerkiksi navigoimalla takaisin edelliseen ruutuun, painamalla ”Undo” tai ”Cancel”-nappeja, valitsemalla toisen ruudun päävalikosta tai etsimällä vihjeitä ”Help”-napin takaa. (Kenett, Harel & Ruggeri 2018, 360.)

Käyttäjäkokemuksen voi lukea negatiiviseksi tai positiiviseksi sen mukaan ilmenikö käytön yhteydessä joku ”UPI”. Kenett, Harel & Ruggeri (2018, 361) avaa kuinka käytettävyyttä voidaan arvioida saatujen ”UPI” tulosten mukaan esimerkiksi näytön lataukseen ja näytön lukemiseen kuluneen ajan perusteella. Tuloksissa näytön lataukseen kestävän ajan tulisi

olla suhteessa näytöllä olevan sisällön määrään ja näytön lataukseen kuluneen ajan tulisi vaikuttaa käyttökokemukseen. Liian pitkään kuluneen latausajan tulisi tuottaa negatiivinen käyttökokemus. Tutkimusten mukaan myös liian lyhyt latausaika saattaa johtaa negatiiviseen käyttökokemukseen. Näytön lukemiseen käytetty aika voi vaikuttaa käyttökokemukseen. Käyttäjät saattavat pysyä näytöllä pitkään, koska sillä on paljon dataa tai koska heidän lukemansa teksti on vaikeasti ymmärrettävissä. Näytön lukemiseen käytetyn ajan tulisi olla suhteessa näytöllä olevaan tekstin määrään. Tiedon etsimiseen näytöltä käytetty aika voi vaikuttaa käyttökokemukseen.

Helppo, suhteellisen nopea ja tehokas järjestelmän arviointitapa on käyttökokemuksen tai käytettävyyden asiantuntijan käyttäminen järjestelmän testaamisessa. Asiantuntija ei voi kuitenkaan korvata oikeiden ihmisten käyttämistä testauksessa, mutta asiantuntijoiden käyttäminen voi olla paikallaan varsinkin suunnittelun alkuvaiheessa. Asiantuntijat pystyvät identifioimaan mahdollisia ongelmia kokemuksensa perusteella. Asiantuntijoiden voidaan antaa vaan yleisesti tutustua suunnitelmaan, mutta parhaiten heidän kokemuksestansa on hyötyä, kun etukäteen valitaan joku tietty lähestymistapa. Yksi yleinen tapa asiantuntija-arvioinnille on skenaariopohjainen testaus, jossa asiantuntija kulkee läpi ehdotetun skenaarion. Toinen vaihtoehto on, että asiantuntija omaksuu jonkun käyttäjäpersoonan. (Benyon 2019, 246.)

Heuristisessä arvioinnissa asiantuntija käy läpi käsillä olevaa suunnitelmaa ja vertaa sitä listaan, jossa on lueteltu hyvän suunnitelman periaatteet. Läpikäytäviä heuristiikkoja voi olla useita, joista valita, riippuen siitä tehdäänkö yleistä arviointia vai johonkin tiettyyn alueeseen liittyvää. Parhaimmassa tapauksessa käyttöliittymän käy läpi useampi asiantuntija, joista jokainen kirjaa ylös löytämänsä ongelmat ja käyttöliittymään liittyvät heuristiikat ja ehdottaa ratkaisunsa ongelmien korjaamiseksi. Vaikeusasteikon lisääminen arviointiin auttaa havainnoimaan ongelman vakavuutta. (Benyon 2019, 246–247.)

Yleinen tapa käyttöliittymäongelmien vaikutusten arviointiin on antaa ongelmalle numero, joka vastaa ongelman vakavuutta (Lewis & Sauro 2012, luku 2):

1. Ongelma estää tehtävän suorittamisen loppuun
2. Ongelma aiheuttaa hidastusta tai turhautumista
3. Ongelmalla on suhteellisen pieni vaikutus tehtävän suorittamiseen
4. Suositus ongelman parantamiseksi

Käytettävyyttä on testattu perinteisesti laboratorio-olosuhteissa. Testauksessa testihenkilö käyttää päätelaitetta ja suorittaa sen avulla testattavan toimenpiteen. Testauksen etenemistä tarkkaillaan sekä teknisesti että asiantuntijoiden toimesta. Testin aikana tehtyjen havaintojen avulla tehdään johtopäätökset siitä, mikä ratkaisussa on toimivaa, ja mitä osa-alueita olisi tarpeen kehittää lisää. Käyttäjän reaktioita testitilanteen aikana

voidaan seurata monella tavalla. Yksi vaihtoehto on seurata, miten käyttäjä operoi päätelaitteen kanssa, ja miten hän navigoi sovelluksessa. Sen lisäksi käyttäjän reaktioista voidaan kerätä tietoa seuraamalla silmänliikkeitä ja jopa mittaamalla sykettä. Silmien liikkeitä tutkimalla on löydetty toistuvia käyttäytymismalleja, kuten se mitä liikerataa noudattaen käyttäjän katse useimmiten liikkuu näyttöruudulla. (Filenius 2015, osa 4.)

Jos testauksessa halutaan selvitä vähemmällä työllä ja kustannuksilla, voidaan käyttää palvelun testaamiseen haamukäyttäjiä. Haamukäyttäjän testaamisessa on kuitenkin se ongelma, että hänelle on annettu valmis tehtävä, jonka mukaisesti hän etenee käyttöliittymässä. Tehtävät on taas valmistanut henkilö, joka tietää miten järjestelmän on tarkoitus toimia. Sen vuoksi haamukäyttäjien testituloksiin kannattaa suhtautua kriittisesti, koska ne eivät täysin vastaa aidon käyttäjän käyttökokemusta. Aidon asiakkaan tullessa käyttämään palvelua ensimmäistä kertaa voivat hänen odotuksensa käyttöliittymän toimivuudesta olla ihan erilaiset kuin suunnittelijoiden. Aito käyttäjä ei välttämättä tiedä edes mitä häneltä odotetaan, kun hän tulee käyttöliittymän ääreen. Tämä loppukäyttäjän käytöksen ja odotusten arvaamattomuus on syy siihen miksi asiakaslähtöinen palveluluiden ja järjestelmien suunnittelu on niin vaikeaa. (Filenius 2015, osa 4.)

Seuraavassa kappaleessa käsittelen Käyttöliittymä X:n käytettävyyden testausta.

8 Käyttöliittymä X:n käytettävyyden testaus

Käyttöliittymä X:n tuoteomistaja Kuronen sanoi hänen kanssaan pitämässäni alkukeskustelussa, että nyt kehitettävän käyttöliittymän toiminnallisuuksien testaamisen lisäksi myös käyttöliittymän käyttäjäkokemusaspekti on tärkeä. Kehitystiimi haluaa tietää, onko näytöt selkeitä, onko ne muutokset, mitä on tehty siirryttäessä nykyisestä clientista Web Clientiin hyviä, ja onko client käytettävyydeltään oikeanlainen. Suurin ero nykyisen clientin ja Web Clientin välillä on hakutoiminnoissa ja niiden muutoksesta halutaankin testauksen myötä saada kommentteja. Testaajilta halutaan myös saada tietoa siitä, onko hakutoimintoihin lisätyt hakukriteerit hyödyllisiä, ja puuttuuko hakutoiminnoista vielä jotain tarpeellista. (Kuronen 3.12.2020.) Tässä luvussa kerron, miten Käyttöliittymä X:n testaus on suunniteltu toteutuvan.

8.1 Kaksinkertainen testaus

Käyttöliittymä X:n käytettävyyden testaus on formatiivista testausta. Testauksen tarkoituksena on auttaa käyttöliittymän kehitystiimiä muokkaamaan käyttöliittymästä käytettävyydeltään mahdollisimman toimiva ratkaisu. Testausta suoritetaan kahdella eri tavalla: toiminnallisuustestauksen yhteydessä sekä erillistä käytettävyyden testauslomaketta käyttämällä.

Ensimmäiset käytettävyyteen liittyvät testaukset tapahtuvat toiminnallisuustestauksen yhteydessä. Käyttöliittymän toiminnallisuuksien testaajaryhmä kirjoittaa käyttämälleen Excel-lomakkeelle toiminnallisuuteen liittyvien virheiden lisäksi kommentteja myös siitä, mitä mieltä he ovat uuden käyttöliittymän käytettävyydestä. Koska toiminnallisuuksien testaaminen aikana testaaja suorittavat manuaalisesti tehtäviä, joita käyttöliittymällä oikeastikin tehdään, pystyvät he myös samalla arvioimaan sitä, kuinka käytettävä uusi käyttöliittymä on. Toiminnallisuustestauksen aikana suoritettavat tehtävät ovat vain hieman suuntaa antavia siinä, mitä käyttöliittymällä tehdään testauksen aikana. Tämä mahdollistaa erilaisten polkujen valinnan tehtäviä suoritettaessa, ja kasvattaa sen mahdollisuutta, että testauksessa löytyy oikeita käytettävyyteen liittyviä ongelmia. Kuten ISO 9241-11 standardissa sanotaan, käytettävyyden määritykseen ei liity pelkästään käyttöliittymän tehokkuus ja nopeus, vaan myös käytön yhteydessä syntynyt tunne käyttöliittymän käytön helppoudesta on tärkeää.

Toiminnallisuustestausta suorittavan ryhmän käytettävyystestauksen lisäksi Käyttöliittymä X:lle suunniteltiin Fileniuksen (2015) kuvaamaa haamutestausta mukaileva käytettävyystestaus. Tätä testausta varten valmistin tehtävälomakkeen (Liite 3), joka sisältää tehtäviä, joita testaaja suorittaa käyttöliittymän avulla. Käytettävyyttä testaa

näiden lomakkeiden avulla kaksi henkilöä, jotka eivät ole mukana toiminnallisuuden testaamisessa. Kumpikaan henkilöistä ei ole testausvaiheessa käyttänyt Käyttöliittymä X:ää edeltävää clientia muutamaa kertaa enempää, joten heillä ei ole suuria ennakko-odotuksia siitä, mitä missäkin vaiheessa tulisi tapahtua. Seuraavaksi kuvailen, kuinka käytettävyystestauksen suunnitellaan etenevän tehtävälomakkeen avulla.

8.2 Käytettävyystestauksen eteneminen

Käytettävyystestausta suoritetaan usein laboratorio-olosuhteissa, kuten Filenius (2015) kertoo käytettävyystestauksista kirjoittaessaan. Käyttöliittymä X:n käytettävyystestauksessa ei kuitenkaan käytetä testauksen seurantaa, vaan kumpikin testaajista suorittaa testauksen itsenäisesti. Syy tähän on aikatauluihin liittyviä. Testaus suoritetaan työaikana muun työn ohessa, ja testaajat käyttävät testaukseen itse valitsemansa määrän aikaa. Ennen testausta tuoteomistaja tekee testaajille tunnukset, joilla he pääsevät kirjautumaan käyttöliittymään, ja varmistaa käyttöliittymän toimimisen testaajien verkkoympäristössä.

Käytettävyystestauksen lomakkeella on yhteensä seitsemän tehtävää. Tehtävissä pyydetään etsimään tietoa käyttöliittymästä, ja lisäämään ohjeissa lukevat tiedot järjestelmään käyttöliittymän avulla. Testauksen aikana testaaja kirjoittaa tehtävän suorituksen etenemisen ja tehtävien suorittamisen aikana tekemänsä huomiot lomakkeelle. Lopuksi kunkin tehtävä n onnistuminen arvioidaan asteikolla yhdestä neljään. Arvioinnissa päätin ottaa mallia Lewis & Sauron (2012) ongelmien kategorioinnista. Lomakkeen asteikko on esitelty lomakkeella seuraavanlaisesti:

1. Onnistui odotetusti
2. Onnistui, mutta ei täydellisesti
3. Tehtävän suorittaminen oli hankalaa/tehtävää jouduttiin muuttamaan
4. Tehtävä jäi suorittamatta

Lomakkeella on lisäksi kaksi kohtaa tehtävien suorittamisen jälkeisiä kommentteja varten. Ensimmäisessä kommenttikentässä kysytään testaajien mielipiteitä käyttöliittymän ulkonäöstä ja toiseen kommenttikenttään pyydetään kommentteja käyttöliittymän käytön helppouteen liittyen. Kuten graafinen.com (2015) sivulla mainitaan, tulee käyttöliittymän suunnittelun pohjautua ajatukseen, että käyttäjä pystyy käyttöliittymän avulla tekemään mahdollisimman helposti ja ilman monimutkaisia toimintoja sen, mitä hän on käyttöliittymällä tullutkin tekemään. Haluamme saada testaajilta kommentteja siihen, noudattaako Käyttöliittymä X tällä hetkellä tätä periaatetta. Käytettävyystestausta järjestetään tässä vaiheessa ainoastaan yksi kierros, mutta todennäköisesti käytettävyyttä testataan jollain muulla tavalla vielä ennen lopullista julkaisua.

9 Pohdinta

Tässä viimeisessä opinnäytetyön osassa käyn läpi prosessin aikana tekemiäni havaintoja, oppimiani asioita ja työn tuloksia. Aluksi pohdin, saatiinko opinnäytetyön avulla vastauksia johdannossa kerrottuihin tutkimuskysymyksiin, sen jälkeen käsittelen opinnäytetyön aikana tehtyjä tuotoksia ja prosessin aikana oppimiani asioita. Lopuksi esitän kehitysehdotuksia Yritys Oy:lle.

9.1 Tutkimuskysymyksiin vastaaminen

Tämän opinnäytetyön tavoitteena oli selvittää, mitä tulee ottaa huomioon käyttöliittymän testausta suunniteltaessa, miten käyttöliittymän toiminnallisuuksia ja käytettävyyttä testataan, mitä käyttöliittymän testaussuunnitelma sisältää ja miten Yritys Oy:n Käyttöliittymä X:n testaus suoritetaan. Lisäksi asetin alikysymyksiksi, miten testaus sisältyy ohjelmistojen elinkaarimalleihin, mitä eri tasoja ohjelmistojen testausta on, millä eri tekniikoilla käyttöliittymää voidaan testata ja mitä tarkoitetaan käyttöliittymän käytettävyydellä. Näihin kysymyksiin lähdin etsimään vastauksia eri lähdekirjallisuutta tutkimalla ja opinnäytetyöprosessin ajalle suunniteltuja tuotoksia tekemällä.

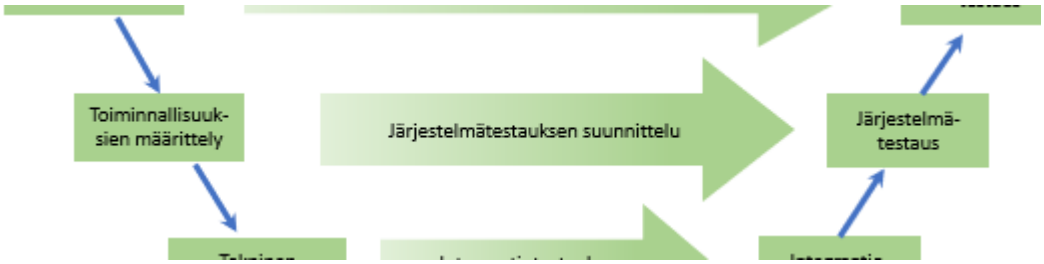
Ensimmäiseksi lähdin etsimään testauksen suunnitteluun ja testaussuunnitelmaan liittyvää teoriaa. Ohjelmistojen ja järjestelmien testaukseen liittyviä lähteitä läpikäydessäni olin yllättynyt, että testauksen suunnittelemisesta ja testaussuunnitelman sisällöstä löytyi niin vähän tietoa. Monesti testaussuunnitelma kuitattiin kirjallisuudessa vain dokumenttina, joka on hyvä olla olemassa, mutta se, mitä suunnitelman tulisi sisältää, oli hyvin lyhyesti kuitattu, jos ollenkaan. Siksi minunkin opinnäytetyössäni testaussuunnitelmaan liittyvät lähteet ovat pääasiassa nettilähteitä. FiSTB:n testaajan perustason sertifikaattisisällössä (FiSTB 2018) oli hyvin kerrottu, että ennen kuin testausta aletaan suunnittelemaan, tulisi testaukseen liittyvät asiat kirjata ylös joko yleiseen, tai jokaisen testaustason omaan testaussuunnitelmaan. Samassa dokumentissa oli myös lueteltu, mitä testaussuunnitelman olisi hyvä sisältää. Halusin kuitenkin myös löytää mallin, jonka pohjalta voisin lähteä Käyttöliittymä X.n testaussuunnitelmaa luomaan. Päätin luottaa Software Testing Help-sivuston tekemään pohjaan. Vertasin pohjaa FiSTB:n luettelemaan sisältöön ja totesin sen sopivaksi. Näiden lähteiden avulla sain selkeän kuvan siitä, mitä testauksen suunnittelussa tulee ottaa huomioon, ja mitä tietoa minun tulee kerätä varsinaiseen testaussuunnitelmaan.

Yhtenä testauksen suunnittelun osana on teoriassa mainittu testauksen aikataulun muodostaminen. Käyttöliittymä X:n testauksen aikataulu määräytyi kuitenkin yrityksen puolelta eikä ollut siten minun tekemäni suunnittelun osana. Aikataulun suunnittelun

tärkeys testauksen valmistelussa tuli kuitenkin hyvin esille. Erityisesti pidin hyvänä huomiota siitä, että myös testauksen suunnitteluun varattu aika tulee ottaa huomioon aikatauluja suunniteltaessa.

Testauksen eri rooleihin liittyvää lähdekirjallisuutta oli hankala verrata toisiinsa, koska jokainen käytti hieman eri titteleitä ja työnjakoa. Sen perusteella oli hankala päätellä, kutsuisinko omaa rooliani testauspäälliköksi vai testauksen suunnittelijaksi. Päädyin kuitenkin jälkimmäiseen, koska se tuntui tähän projektiin sopivammalta. Luulenkin, että jokaiseen testaukseen liittyy erilaisia tehtäväjakoja ja roolituksia organisaatiosta riippuen. Koska jokainen testaus, testattava järjestelmä ja projekti on erilainen, täytyy testausta suunniteltaessa miettiä tarkkaan, mitä rooleja kyseiseen projektiin kuuluu, ja mitkä ovat rooleille asetetut odotukset.

Seuraava tutkimuskysymys, johon lähdin etsimään vastausta, liittyi käyttöliittymän toiminnallisuuksien testaamiseen. Tähän testaamiseen liittyy vahvasti eri ohjelmistojen elinkaarimallit. Ne olivat minulle tuttuja jo opiskelujen aikaisilta kursseilta, mutta nyt etsin tietoa ja kiinnitin huomiota erityisesti testaamisen sijoittumisesta eri elinkaarimalleihin. Nyt suoritettu Käyttöliittymä X:n testaus oli järjestelmätestausta. Yleisesti käytössä olevassa V-mallissa järjestelmätestauksen suunnittelu olisi tapahtunut jo käyttöliittymän toiminnallisuuksia määriteltäessä (kuva 4). Siinä vaiheessa, kun tulin mukaan tähän projektiin, ja aloin suunnittelemaan testausta, oli käyttöliittymän toiminnallisuudet jo päätetty ja käyttöliittymä koodattu lähes valmiiksi. En siksi voinut olla mukana määrittelyvaiheessa ja hyödyntää niitä tietoja järjestelmätestauksen suunnittelussa.



Kuva 4. Järjestelmätestauksen suunnitteluvaihe V-mallissa

Käyttöliittymä X:n kehittämisessä noudatetaan iteratiivista elinkaarimallia, tarkemmin Scrumia. Kuten kerroin kappaleessa 5.1, iteratiivisessa mallissa kehitys etenee sykleittäin sprinttien muodossa. Nyt tehdyt testaukset eivät sinällään olleet osana näitä sprinttejä, mutta testausten avulla kehitystiimi sai arvokasta tietoa, jonka avulla se pystyi suunnittelemaan tarvittavia kehitystoimia sprinteille.

Testauksen eri tasoista ja menetelmistä löytyi hyvin tietoa kirjallisuudesta. Pystyin näiden eri lähteiden avulla tunnistamaan, että nyt tehtävä toiminnallisuustestaus oli toiminnallista mustalaatikkotekniikkaa, jossa taustalla oleva koodi ei ole testaajien tiedossa. Lisäksi testauksessa käytettiin hyödyksi testaajien kokemusta Käyttöliittymä X:ää edeltävän käyttöliittymän käytöstä, joten kokemuspohjainen tekniikka sopi hyvin myös tähän testaukseen.

Kolmas opinnäytetyöni aihe liittyi käyttöliittymän käytettävyyteen. Käyttöliittymän suunnittelusta en löytänyt oikein tuoretta lähdekirjallisuutta, aiheesta oli ehkä kirjoitettu enemmän vuosituhannen vaihteen jälkeen kuin viime vuosina. Löysin kuitenkin pääpiirteet siitä, mitä käyttöliittymän suunnittelussa on hyvä ottaa huomioon. Käytettävyys on aiheena laaja ja kattaa muutakin kuin käyttöliittymän käytettävyyden. Siihen liittyen löytyi hyvin tietoa ja monipuolisesti lähdekirjallisuutta. Kaikissa lähteissä oli mainittu ISO 9241-11 standardi, jota pidin merkkinä siitä, että lähteissä puhuttiin samasta asiasta. Standardin mukaan tuotteen tulee olla tehokas, odotusten mukainen ja toimia tyydyttävästi. Se, että käytettävyydestä on tehty ISO-standardi kuvaa mielestäni asian tärkeyttä. Käyttöliittymä X:n käytettävyystestausta suoritettiin tämän opinnäytetyön suunnitteluun liittyvässä testauksessa kahdessa eri osassa; ensin toiminnallisuustestauksen yhteydessä ja sitten erillisen lomakkeen avulla. Valittu testausmenetelmä oli formatiivinen, jonka tarkoituksena oli antaa tietoa käyttöliittymän kehittämiseksi. Käyttöliittymä X:n kehittämisen ollessa vielä kesken, oli nyt suunnitellut käytettävyystestaukset riittäviä, ja niistä saatiin hyvin tietoa jatkokehitystä varten.

Kaiken kaikkiaan opinnäytetyön tulokset vastasivat mielestäni hyvin asetettuihin tutkimuskysymyksiin. Seuraavaksi käsittelen opinnäyteyön tuloksena syntyneitä testauslomakkeita ja testaussuunnitelmaa. Lisäksi pohdin opinnäytetyöprosessin aikana oppimiani asioita.

9.2 Tuotokset ja opitut asiat

Tämän toiminnallisen opinnäytetyön tuotoksena syntyi lomakkeet Käyttöliittymä X:n toiminnallisuuksien ja käytettävyyden testaamiseen, sekä kyseisen käyttöliittymän testauksen testaussuunnitelma. Testauslomakkeita käytettiin Käyttöliittymä X:n testaamisessa, ja niiden kerrottiin helpottaneen kovasti testaustulosten keräämistä.

Toiminnallisuustestauksen lomakkeelle valitsin pohjaksi Excel-tiedoston ja käytettävyystestauksen lomakkeelle Word-dokumentin. Excel oli mielestäni onnistunut valinta toiminnallisuustestauksen lomakkeiden pohjaksi, koska sitä pystyi hyödyntämään hyvin lomakkeen muodostamisessa, ja eri näyttöjen lomakkeet sai helposti kerättyä

sopiviin kokonaisuuksiin. Ajattelin myös, että mikäli kehitystiimi haluaa tehdä jonkinlaisen makron esimerkiksi epäonnistuneiden testaustehtävien keräämiseksi, se onnistuu Excelin avulla. Excel mahdollisti myös monen toisiinsa liittyvän näytön testauslomakkeiden keräämisen yhteen tiedostoon, joka helpotti näyttöjen osioinnissa, ja jakamisessa testaajien kesken. Käytettävyystestauksen lomakkeen pohjaksi valitsin taas Word-dokumentin, koska sain muotoiltua haluamani kysymykset hyvin Wordin taulukkoon. Lomakkeesta tuli mielestäni selkeä ja helppo käyttää.

Yrityksen aikataulun vuoksi jouduin tekemään testauslomakkeista ensimmäiset versiot jo ennen kuin olin ehtinyt perehtyä täysin testaamisen teoreettiseen taustaan. Testauksen ensimmäinen kierros toteutettiin näitä lomakkeita käyttäen. Lomakkeet saivat kuitenkin paljon kiitosta testaajilta ja kehitystiimiltä, joten en tehnyt juurikaan muutoksia lomakkeisiin toiselle testauskierrokselle. Lisäsin ainoastaan yhden tehtävän, joka koski väärän tiedon syöttämistä. Ensimmäisellä testauskierroksella järjestelmässä ei olisi itseasiassa ollutkaan vielä tarvittavia koodauksia, jotka olisivat tuottaneet oikean vastauksen väärää tietoa syötettäessä, joten tämän tehtävän lisääminen vasta toiselle kierrokselle oli ihan osuva.

Testaussuunnitelmalle halusin löytää mallipohjan ja löysinkin sellaisen Software Testing Help -sivustolta. Keräsin testaussuunnitelmaan Käyttöliittymä X:n testaamiseen liittyviä asioita teoriataustasta löytyneiden kohtien mukaan. Yritys Oy:llä ei ollut käytössä aikaisemmin kirjallista testaussuunnitelmaa, joten nyt tekemääni testaussuunnitelmaa on mahdollista käyttää pohjana myös muissa tulevissa projekteissa.

Teoriaosuutta kirjoittaessani huomasin, että olisi ollut parempi, jos olisin ehtinyt perehtyä teoriaan ennen kuin jouduin tekemään lomakkeet, niin kuin tarkoitus olisikin opinnäytetyöprosessissa. Olisin voinut keskustella tuoteomistajan kanssa enemmän testauksen taustoista ja halutusta lopputuloksesta, kun olisin tiennyt tarkemmin, mitä tietoa tarvitsen. Nyt jouduin tekemään lomakkeet oikeastaan ainoastaan tuoteomistajan kanssa käymäni alkukeskustelun pohjalta, jossa hän mainitsi, että Käyttöliittymä X:n toiminnallisuudesta testataan tiedon haku, lisäys, muokkaus ja poisto. Sinänsä tämäkin tieto riitti jo nyt suoritettuun testaukseen, mutta voi olla, että testauksesta olisi saanut vielä enemmän irti, jos olisin ehtinyt paneutua teoriaan ennen lomakkeiden tekemistä ja testauksen alkamista.

Olen opintojeni aikana opiskellut käytettävyyteen liittyviä teemoja, joten teorian se puoli oli tuttu ennestään. En kuitenkaan hyödyntänyt Käyttöliittymä X:n käytettävyystestauksen suunnittelussa usein käytettävyystestauksessa käytettäviä käyttäjäprofiileja ja käyttäjätarinoita, koska nyt tehtävässä testauksessa testaajat olivat itse potentiaalisia

käyttäjiä. Käyttöliittymän käyttötarkoitus ja tausta olivat heille sen vuoksi tuttuja eikä tilanteen esittelylle käyttäjätarinoiden kautta ollut sen vuoksi tarvetta.

Käytettävyystestaukseen ei tässä vaiheessa laitettu kovinkaan paljon resursseja, koska käyttöliittymää on tarkoitus vielä jatkokehittää käytettävyyden osalta. Huomasin kuitenkin jo nyt tehdyn käytettävyystestauksen aikana, että tarkkailtu testaustilanne, jossa testaaja olisi kertonut ääneen, mitä tekee, olisi saattanut antaa enemmän tietoa käytettävyydestä, kuin testaus, jossa testaaja itse kirjoittaa testauksen aikana tekemänsä huomiot.

Opinnäytetyöprosessin aikana opin paljon uutta testauksen eri tasoista ja menetelmistä. En ole opintojeni enkä työurani aikana ollut aikaisemmin ohjelmistokehityksen kanssa tekemisissä. Sain lähdekirjallisuutta lukiessani kuitenkin hyvän käsityksen, miten monella eri tasolla ohjelmistojen kehitys tapahtuu, ja miten monelta eri kannalta uusien ohjelmistojen suunnittelussa on asioita katsottava. Pelkkä hyvin kirjoitettu koodi ja sen testaus ei riitä, vaan kaikessa tulee ottaa huomioon ohjelmiston lopullinen käyttäjä.

9.3 Kehittämisehdotukset

Opinnäytetyöhöni pohjautuvat kehitysehdotukset liittyvät oikeastaan täysin käytettävyyden testaamiseen. Tuoteomistaja Kuronen kertoi alkuhaastattelussa 3.12.2021, että Käyttöliittymä X:n kehitys on jaettu kolmeen vaiheeseen ja opinnäytetyön aikana suunniteltu testaus tapahtui Käyttöliittymä X:n kehityksen ensimmäisen vaiheen lopussa. Käyttöliittymän pilotointi asiakkaalle tapahtuu vasta seuraavan kehitysvaiheen jälkeen. Käyttöliittymän kehitystyötä on tarkoitus jatkaa pilotoinnin jälkeen.

Käytettävyyden osalta järjestelmässä on siis vielä tapahtumassa. Tuoteomistaja Kuronen sanoikin, että tässä vaiheessa ei ole vielä pystytty tekemään parannusta kaikkiin toimintoihin, vaan toiminnot ovat samoja kuin nykyisessä clientissa. Siinä vaiheessa, kun käyttöliittymän käytettävyyttä aletaan kehittää eteenpäin, suosittelisin suorittamaan lisätestausta. Yksi vaihtoehto olisi esimerkiksi summatiivinen Benchmark-testaus, jossa käyttöliittymän käytettävyyttä mitataan johonkin määrättyyn kiintopisteeseen vertailemalla. Toinen vaihtoehto voisi olla vertaileva summatiivinen testaus, jossa otetaan vertailukohdaksi muutama kilpailijoiden käyttöliittymä. (Lewis & Sauro 2012, luku 2.) Samalla voisi löytyä ideoita Käyttöliittymä X:n kehittämiseksi. Ja kuten mainitsinkin, myös seuratusta käytettävyystestauksesta voisi olla hyötyä.

Data-analyysin hyödyntämistä käyttöliittymän käytettävyyden tutkimisessa kannattaisi miettiä myös Käyttöliittymä X:n kohdalla, varsinkin pilotoinnin aikana. Käyttäjien käytön aikana tekemiä toimia voisi seurata dataa keräämällä, ja käytettävyyttä arvioida eri

mittareiden avulla. Tällaisia voisivat olla esimerkiksi aika, jonka käyttäjä viipyy eri kohdissa käyttöliittymää, tai se missä kohti käyttäjä joutuu palaamaan takaisin. Dataa arvioimalla saisi arvokasta tietoa vielä kehitettävistä kohdista.

Koska Yritys Oy:llä on omia käyttöliittymiä ja järjestelmiä, olisi hyvä, jos yrityksellä olisi testausta varten yleinen testausstrategia ja valmiiksi mietityt mahdolliset lähestymistavat. Testausstrategian pohjalta olisi helpompi lähteä miettimään kehitettävien tuotteiden testaamista. Tällainen testaukseen liittyvä toimintamalli voisi olla Yritys Oy:n seuraava kehityskohde. Tämän strategian ja lähestymistavat voisi kirjata ylös yleiseen testaussuunnitelmaan, jonka pohjata olisi helppo lähteä tekemään projektikohtaisia testaussuunnitelmia. Nyt tekemääni testaussuunnitelmaa voisi käyttää kyseisen yleisen testaussuunnitelman pohjana.

Lähteet

Benyon, D. 2019. Designing user experience: a guide to HCI, UX and interaction design. Pearson Education Ltd

Black, R. 2016. Advanced Software Testing -Vol. 1. Rocky Nook.

De Bleecker, I. & Okoroji, R. 2018. Remote Usability Testing. Packt Publishing.

Dooley, J. 2011. Software Development and Professional Practice. Apress. USA.

Dooley, J. 2017. Software Development, Design and Coding. Apress. Galesburg, Illinois.

FiSTB 2018. Sertifioitu Testaaja. Perustason sertifikaattisisältö. International Software Testing Qualifications Board. Luettavissa: http://www.fistb.fi/sites/fistb/files/liitteet/CTFL%202018%20Sertifikaattisisa%CC%88lto%CC%88%2020181010-1%20Valmis.pdf. Luettu: 30.1.2021

Filenius, M. 2015. Digitaalinen asiakaskokemus: menesty monikanavaisessa liiketoiminnassa. Docendo Oy.

Filipova, O., Vilão, R. 2018. Software Development From A to Z.

Graafinen.com. 2015. Hyvä käyttöliittymä – 10 muistisääntöä. Luettavissa: https://www.graafinen.com/suunnittelu/digi/hyva-kayttoliittyma-10-muistisaantoa/ Luettu: 28.3.2021

ISTQB 2015. ISTQB:n testaussanasto v. 2.3. Englanti – Suomi. Luettavissa: https://tivia.fi/fistb-testi/wp-content/uploads/sites/30/2020/12/istqb_sanasto_2015-04-30-2.3-ENG-FI_EI_Kokeeseen.pdf. Luettu: 30.12.2020

Kuronen, J. 3.12.2020. Tuoteomistaja. Yritys Oy. Haastattelu. Helsinki.

Lewis, J. R., Sauro, J. 2012. Quantifying the User Experience: Practical Statistics for User Research.

Morgan, P. 2019. Luku 1. The Fundamentals of Testing. Teoksessa: Hambling, B. 2019. Software Testing, An ISTQB-BCS Certified Tester Foundation guide, Fourth edition. BCS Learning & Development Ltd.

Niemelä, H. 2.6.2020. Sovelluksen käytettävyyden testaaminen. Verkkolehti SeAMK. Luettavissa: https://lehti.seamk.fi/alykkaat-ja-energiatehokkaat-jarjestelmat/sovelluksen-kaytettavyyden-testaaminen/. Luettu: 9.12.2020

Ranorex 2021. Beginner's guide to User Interface testing. Luettavissa: https://www.ranorex.com/resources/testing-wiki/gui-testing/ Luettu: 28.12.2020.

Samaroo, A. 2019. Luku 2. Life Cycles. Teoksessa: Hambling, B. 2019. Software Testing, An ISTQB-BCS Certified Tester Foundation guide, Fourth edition. BCS Learning & Development Ltd.

Ratilainen, T. 2019. Bugien metsästys kannattaa. Finnish Information Processing Association TIVIA. Luettavissa: https://tivia.fi/2019/12/23/bugien-metsastys-kannattaa/ Luettu 27.2.2021

Software Testing Help, 2021. Test Plan Tutorial: A Guide To Write A Software Test Plan Document From Scratch. Luettavissa: https://www.softwaretestinghelp.com/how-to-write-test-plan-document-software-testing-training-day3/ Luettu: 31.1.2021

Thompson, G. 2019. Luku 5. Test Management. Teoksessa: Hambling, B. 2019. Software Testing, An ISTQB-BCS Certified Tester Foundation guide, Fourth edition. BCS Learning & Development Ltd.

Liitteet

Liite 1. Näyttöjen osiointi

Näyttöjen osiointi

Käyttöliittymä X

Näyttö	Testaaja 1.	Testaaja 2.
Hakemiston otsikko 1		
Näyttö 1	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 2	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 3	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 4	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 5	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 6	*1. testaajan nimi*	*2. testaajan nimi*
Hakemiston otsikko 2		
Ryhmän otsikko	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 7	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 8	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 9	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 10	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 11	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 12	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 13	*1. testaajan nimi*	*2. testaajan nimi*
Ryhmän otsikko	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 14	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 15	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 16	*1. testaajan nimi*	*2. testaajan nimi*
Ryhmän otsikko	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 17	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 18	*1. testaajan nimi*	*2. testaajan nimi*
Ryhmän otsikko	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 19	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 20	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 21	*1. testaajan nimi*	*2. testaajan nimi*
Ryhmän otsikko	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 22	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 23	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 24	*1. testaajan nimi*	*2. testaajan nimi*
Hakemiston otsikko 3		
Näyttö 25	*1. testaajan nimi*	*2. testaajan nimi*
Näyttö 26	*1. testaajan nimi*	*2. testaajan nimi*

Liite 2. Toiminnallisuustestauksen lomake

WEB CLIENT TESTAUS malli

Päivämäärä:	8.12.2020/ 12.12.2020	**Virheen tasojen selitykset**	
Testaaja:	*Ensimmäisen testaajan nimi/ Toisen testaajan nimi*	1	Korkea
Selain:	*Ensimmäisen testaajan käyttämä selain/ Toisen testaajan käyttämä selain*	2	Alhainen
Osio:	*Testattavan näytön nimi*		

Kommentit toiminnoista		**Tehtävä onnistui** Kyllä	Ei	**Virheen taso**	**Jira-tiketit**
Tehtävä 1.	**Tiedon haku**				
	Testaaja 1. *(Tähän ensimmäinen testaaja kirjoittaa mitä hän on hakenut ja millä hakusanoilla. Sen jälkeen hän kertoo onnistuiko haku vai ei ja oliko hän tyytyväinen hakuun.)*		x	1	ABCD-1234
	Testaaja 2. *(Tähän toinen testaaja kirjoittaa oman hakunsa ja sen kokemukset. Jos ensimmäinen testaaja on raportoinut virheen, ei samaa virhettä raportoida Jiraan uudestaan. Virheen voidaan kuitenkin todeta toistuvan.)*	x			
Tehtävä 2.	**Tiedon lisäys**				
	Testaaja 1. *(Tähän ensimmäinen testaaja kirjoittaa mitä tietoa hän on lisännyt järjestelmään ja miten hän on edennyt tehtävässä. Lisäksi hän kertoo onnistuiko tiedon lisäys vai ei ja oliko hän tyytyväinen niihin vaiheisiin, joita joutui tekemään.)*				
	Testaaja 2. *(Tähän toinen testaaja kirjoittaa mitä tietoa hän on lisännyt järjestelmään ja miten hän on edennyt tehtävässä. Lisäksi hän kertoo onnistuiko tiedon lisäys vai ei ja oliko hän tyytyväinen niihin vaiheisiin, joita joutui tekemään.)*				
Tehtävä 3.	**Tiedon muokkaus**				
	Testaaja 1. *(Tähän ensimmäinen testaaja kirjoittaa mitä tietoa hän on muokannut järjestelmässä ja miten hän on edennyt tehtävässä. Lisäksi hän kertoo onnistuiko tiedon muokkaaminen vai ei ja oliko hän tyytyväinen niihin vaiheisiin, joita joutui tekemään.)*				
	Testaaja 2. *(Tähän toinen testaaja kirjoittaa mitä tietoa hän on muokannut järjestelmässä ja miten hän on edennyt tehtävässä. Lisäksi hän kertoo onnistuiko tiedon muokkaaminen vai ei ja oliko hän tyytyväinen niihin vaiheisiin, joita joutui tekemään.)*				
Tehtävä 4.	**Tiedon poisto**				
	Testaaja 1. *(Tähän ensimmäinen testaaja kirjoittaa mitä tietoa hän on yrittänyt poistaa järjestelmässä ja miten hän on edennyt tehtävässä. Lisäksi hän kertoo onnistuiko tiedon poistaminen vai ei ja oliko hän tyytyväinen niihin vaiheisiin, joita joutui tekemään.)*				
	Testaaja 2. *(Tähän toinen testaaja kirjoittaa mitä tietoa hän on yrittänyt poistaa järjestelmässä ja miten hän on edennyt tehtävässä. Lisäksi hän kertoo onnistuiko tiedon poistaminen vai ei ja oliko hän tyytyväinen niihin vaiheisiin, joita joutui tekemään.)*				

Tehtävä 5.	**Ikkunasta toiseen siirtyminen**				
	Testaaja 1. *(Käyttöliittymässä edetään joissakin toiminnoissa ikkunasta toiseen. Tähän ensimmäinen testaaja kirjoittaa kokemuksensa ikkunasta toiseen siirtymisestä, oliko siirtyminen helppoa ja loogista etc.)*				
	Testaaja 2. *(Tähän toinen testaaja kirjoittaa kokemuksensa ikkunasta toiseen siirtymisestä, oliko siirtyminen helppoa ja loogista etc.)*				

Tehtävä 6. (vain toisella kierroksella)	**Väärän tiedon syöttäminen, mitä tapahtui?**				
	Testaaja 1. *(Käyttöliittymässä on hyvä testata myös sitä, antaako käyttöliittymä syöttää tietoa, joka ei ole kyseiseen kohtaan sopivaa. Tähän ensimmäinen testaaja kirjoittaa, mitä tietoa hän on yrittänyt syöttää ja kuinka käyttöliittymä reagoi väärään tietoon.)*				
	Testaaja 2. *(Tähän toinen testaaja kirjoittaa, mitä tietoa hän on yrittänyt syöttää ja kuinka käyttöliittymä reagoi väärään tietoon.)*				

Tehtävä 7.	**Muu toiminto, mikä?**				
	Testaaja 1. *(Joissakin näytöissä voi tulla esiin myös muita tehtäviä, kuin kaikille näytöille yhteiset tiedon haku, tiedon lisäys, tiedon muokkaus ja tiedon poisto. Tähän ensimmäinen testaaja voi kirjoittaa minkä toiminnon hän on testannut ja mitä tehtävän aikana tapahtui.)*				
	Testaaja 2. *(Tähän toinen testaaja voi kirjoittaa lisäämänsä tehtävän.*				

Kommentit ulkonäöstä

Testaaja 1. *(Tähän ensimmäinen testaaja kirjoittaa kommentinsa näytön ulkonäöstä, selkeydestä, paikikkeiden sijainnista, niiden koosta etc.)*
Testaaja 2. *(Tähän toinen testaaja kirjoittaa kommentinsa näytön ulkonäöstä, selkeydestä, paikikkeiden sijainnista, niiden koosta etc.)*

Liite 3. Käytettävyystestauksen lomake

Käyttöliittymä X käytettävyystestaus

Testaaja:	
Päivämäärä:	
Selain:	

Onnistumisen arvosanojen selitykset	
1	Onnistui odotetusti
2	Onnistui, mutta ei täydellisesti
3	Tehtävän suorittaminen oli hankalaa/tehtävää jouduttiin muuttamaan
4	Tehtävä jäi suorittamatta

Tehtävä	**Suoritus** (Kuvaa mitä teet missäkin vaiheessa)	**Onnistuminen (laita X)**	
Etsi Henkilötiedot -näytöstä Tessa Testaaja		1	
		2	
		3	
		4	
Lisää Tessalle tunniste ja työsuhteen alkamispäiväksi 1.12.2020 ja päättymispäiväksi 1.5.2021. Tallenna		1	
		2	
		3	
		4	
Lisää Tessalle työaikaryhmä ja -kaavio Tuntikeräily Tallenna.		1	
		2	
		3	
		4	
Lisää Tessalle esimieheksi Esimies Elli ja 1. Varahyväksyjäksi Varahyväksyjä Valtteri Tallenna		1	
		2	
		3	
		4	

Etsi kaikki Communication & Marketing osastoon kuuluvat henkilöt		2	
		3	
		4	
Etsi mitä pitää sisällään työaikaryhmä Tuntikeräily		1	
		2	
		3	
		4	
Lisää Erikoispäivä Pitkäperjantai, työaikaan tyyppi 3, kulunvalvontaan Ei vaikutusta. Keksi itse päivämäärä.		1	
		2	
		3	
		4	

Mitä mieltä olet käyttöliittymän ulkonäöstä?
Mitä mieltä olit käyttöliittymän käytön helppoudesta?

Liite 4. Testaussuunnitelma (Luottamuksellinen liite)