Testausprosessin kehittäminen: Xray-laajennuksen käyttö testauksen raportoinnissa

Virve Liimatta

2020 Laurea

Laurea-ammattikorkeakoulu

Testausprosessin kehittäminen: Xray-laajennuksen käyttö testauksen raportoinnissa

Virve Liimatta

Tietojenkäsittely

Opinnäytetyö

toukokuu, 2020

Laurea-ammattikorkeakoulu **Tiivistelmä**
Tietojenkäsittely
Tradenomi (AMK)

Virve Liimatta
Testausprosessin kehittäminen: Xray-laajennuksen käyttö testauksen raportoinnissa
Vuosi 2020 Sivumäärä 74

Opinnäytetyössä selvitettiin, kuinka Xray-testauslaajennuksen avulla raportoidaan järjestelmätestauksen tila ketterässä ohjelmistokehitysprojektissa projektikohtaisten tarpeiden mukaisesti. Työ on laadullinen kehittämistutkimus, joka toteutettiin Ruokaviraston toimeksiannosta. Tietoperustana opinnäytetyössä oli ketterä ohjelmistokehitys ja ohjelmistotestaus. Näihin pohjautuen työssä kuvattiin testausprosessi esimerkkinä käytetyssä ELTE-projektissa.

Työssä hyödynnettiin myös kvalitatiivisen tutkimuksen menetelmiä kuten teemahaastattelua, lomakekyselyä sekä osallistuvaa havainnointia raportoinnin tarpeiden löytämiseksi. Tutkimuksessa tunnistettiin kolme raportoinnin tasoa. Yleistasolla halutaan yleiskuva testauksesta, sprinttikohtaisella tasolla tietyn ajanjakson tilanne ja vikaraporttikohtaisella tasolla vikaraporttien määrä ja niiden elinkaari.

Testausprosessin kuvauksen, tunnistettujen raportointitarpeiden sekä Xray-testauslaajennuksen kartoituksen pohjalta kehitettiin erilaisia raportointitapoja. Jokaiselle tasolle kehitettiin oma raportointimuotonsa toimeksiantajan käyttämää Xray-testauslaajennusta hyödyntäen.

Työ keskittyi kuvaamaan testausta ja sen raportointia tietyn projektin näkökulmasta, mutta projektin voidaan sanoa edustavan Ruokavirastossa toteutettavia ketteriä ohjelmistokehitysprojekteja. Tällöin tulokset ovat yleistettävissä koko organisaation tasolle.

Opinnäytetyötä tullaan hyödyntämään kahdella tapaa. Kehitetyt raportointitavat otetaan käyttöön esimerkkiprojektissa, mutta opinnäytetyön pohjalta tullaan myös laatimaan Ruokavirastolle ohjeistus Xray-testauslaajennuksen käytöstä raportoinnin välineenä. Ohjeistuksen laatimista tai raportointitapojen käyttöönottoa ei kuvata opinnäytetyössä.

Asiasanat: järjestelmätestaus, ohjelmistotestaus, ketterä ohjelmistokehitys, testausprosessi

Laurea University of Applied Sciences **Abstract**
Degree Programme in Business Information Technology
Bachelor's Thesis

Virve Liimatta

Development of Testing Process: the Use of Xray Testing Management Tool in Reporting

Year 2020 Pages 74

The purpose of this Bachelor's thesis was to develop a way of reporting the status of software testing with Xray test management tool in an agile software development project. This qualitative development study was commissioned by Finnish Food Authority.

The theoretical background section included agile software development and software testing. The methods of theme interview, survey and participant observation were used to discover the demands for reports. The ELTE project was used as an example project.

During the study three levels of reports were recognized. In the general level the interest was in the overview of testing. In the sprint level the focus was the success of testing during a certain time period. The third level concentrated on number and life cycle of bugs.

The description of the testing process, the recognized needs for reports and the mapping of Xray test management tool were used as a basis for the reports developed for each of the levels. Xray was used to compose these reports in accordance with the commissioner's request.

The reports will be used in the ELTE project. Moreover, this thesis will be used as a basis for the guidelines developed for the commissioner of the use of Xray. However, the feedback from the project or the development of guidelines is not depicted in this thesis.

Keywords: system testing, software testing, agile software development, testing process

Sisällys

1 Johdanto

Tässä opinnäytetyössä etsitään parhaita käytänteitä ketterässä ohjelmistokehitysprojektissa tapahtuvaan testaukseen, sen seurantaan ja raportointiin toimeksiantajan käytössä olevan ohjelmiston avulla. Toimeksiantajana opinnäytetyössä on Ruokavirasto, jolla on käytössä JIRA-tehtävienhallintaohjelmisto sekä sen testauksen hallintaan tarkoitettu laajennus Xray. Testauslaajennuksen käyttö on kuitenkin verrattain vähäistä, ja Ruokavirastossa halutaan tietää, miten sen avulla voidaan kehittää testausprosessia erityisesti testauksen seurannan ja raportoinnin näkökulmasta.

Tarkastelun kohteena on Ruokaviraston ELTE-projekti, jota käytetään esimerkkinä Xray:n käytöstä ketterässä ohjelmistokehitysprojektissa. Tavoitteena on, että kerätty tieto olisi yleishyödyllistä eli yleistettävissä hyödynnettäväksi muihin Ruokaviraston projekteihin.

Opinnäytetyössä kuvataan projektin testauskäytänteitä ja kartoitetaan projektissa olevia raportoinnin tarpeita. Työssä myös tutkitaan Xray-laajennuksen ominaisuuksia ja arvioidaan niiden soveltuvuutta tarpeiden mukaiseen raportointiin. Tavoitteena on löytää projektille parhaat käytänteet testauksen tilan ja kattavuuden raportointiin Xray-ohjelmiston avulla.

Opinnäytetyön pohjalta on tarkoitus tuottaa Ruokaviraston ohjelmistotuotannon käsikirjaan yleinen ohjeistus testauskäytänteistä ja raportoinnista JIRA:n ja Xray:n avulla. Ohjeistuksen tuottamista ei kuitenkaan käsitellä tämän opinnäytetyön puitteissa. Aihetta ehdotettiin Ruokaviraston puolelta ja se liittyy oleellisesti opinnäytetyön tekijän työnkuvaan manuaalitestaajana ELTE-projektissa.

2 Opinnäytetyön lähtökohdat

Tässä luvussa esitellään opinnäytetyön tutkimusongelma, aiheen rajaus, toimeksiantaja sekä kehittämisympäristönä käytettävä esimerkkiprojekti. Toimeksiantajana on Ruokavirasto, jossa on meneillään noin 30 kehitysprojektia ja ylläpidossa noin 150 tietojärjestelmää, joiden yhteydessä käytetään JIRA-tehtävienhallintaohjelmistoa. Noin 40 tietojärjestelmän kehittämis- tai ylläpitokohdetta käyttää JIRA:n lisäksi Xray-testauslaajennusta, mutta laajennuksen käyttöaste vaihtelee järjestelmäkehittämisten välillä. (Myllyluoma 2019.)

2.1 Tutkimusongelma ja kehittämistavoitteet

Opinnäytetyön tutkimuskysymyksenä on, kuinka Xray-laajennuksen avulla voidaan raportoida testauksen tila ketterässä ohjelmistokehitysprojektissa. Tutkimuskysymykseen vastataan kehittämällä yhden projektin käyttöön raportointitapoja Xray:ta käyttäen.

Tämän saavuttamiseksi opinnäytetyössä selvitetään, millainen on ketterän ohjelmistokehitysprojektin testausprosessi. Tarkoituksena on siis kuvata, miten testaus toteutetaan Xray:n avulla. Raportoinnin kehittämistä varten tulee tunnistaa, millaista tietoa testauksesta halutaan projektissa. Jotta raportointi voidaan toteuttaa Xray-laajennuksen avulla, pitää selvittää ohjelmiston yleisperiaatteet sekä raportoinnin mahdollistavat ominaisuudet. Kunnollinen ohjelmiston tuntemus on avuksi, kun raportointimalleja kehitetään projektia varten.

Xray:n käyttöä halutaan lisätä toimeksiantajana toimivassa Ruokavirastossa ja opinnäytetyön tavoitteena on edesauttaa löytämään parhaat käytänteet testauksen raportointiin ohjelman avulla. Opinnäytetyön pohjalta laaditaan yleiskäyttöinen ohjeistus Xray:n käytöstä Ruokaviraston kehittämisprojekteille, mikä tukisi kyseisen ohjelmiston käytön yleistymistä. Ohjeiden laatimista ei kuitenkaan käsitellä tämän opinnäytetyön puitteissa, vaan keskitytään kartoittamaan ohjelmiston ominaisuuksia ja kehittämään käytänteitä esimerkkinä toimivan projektin käyttöön.

2.2 Aihealueen rajaus

Opinnäytetyö käsittelee yhdessä ketterässä ohjelmistokehitysprojektissa tapahtuvaa testausta ja siinä projektissa ilmeneviä tarpeita testauksen raportoinnille. Testauskäytänteiden ja raportointitarpeiden voidaan kuitenkin olettaa olevan yleistettävissä, sillä Ruokavirastossa on olemassa käytänteet projektinhallintaan, ohjelmistokehitykseen ja laadunvalvontaan, jotka antavat raamit Ruokavirastossa toteutettaville ohjelmistokehitysprojekteille. Näitä käytänteitä noudatetaan myös esimerkkiprojektissa. Tämän vuoksi, vaikka tulosten katsotaan olevan projektikohtaisia, ne ovat silti hyödynnettävissä koko organisaation käyttöön.

Opinnäytetyössä käsiteltävä testaus on järjestelmätestausta. ELTE-projektissa tapahtuvaa muuta testausta ei käsitellä mainintaa enempää. Rajaus johtuu siitä, että prosessia halutaan kehittää erityisesti Xray:n käytön suhteen, eikä muuta testausta raportoida tähän ohjelmistoon. Testausprosessin kuvaamisessa keskitytään manuaalisesti toteutettavaan järjestelmätestaukseen. Automaatiotestaus esitellään vain pääpiirteittäin. Raportoinnin kehittämisessä kuitenkin otetaan huomioon molemmat testausmuodot, sillä myös automaatiotestauksen tulokset raportoidaan Xray-laajennukseen.

2.3 Toimeksiantaja

Opinnäytetyön toimeksiantaja on Ruokavirasto. Vuoden 2019 alussa Elintarviketurvallisuusvirasto (Evira), Maaseutuvirasto (Mavi) ja osa Maanmittauslaitoksen tietotekniikan palvelukeskusta (Mitpa) yhdistettiin uudeksi virastoksi nimeltään Ruokavirasto. Virasto kuuluu Maa- ja metsätalousministeriön hallinnonalaan, ja maanlaajuisena sillä on lähes tuhat työntekijää noin 20 paikkakunnalla. Viraston päätoimipaikka sijaitsee Seinäjoella, mutta suurin toimipaikka on Helsingin Viikissä. Ruokaviraston toiminnan tavoitteena on ihmisten, eläinten ja kasvien terveyden varmistaminen, maaseudun elinvoimaisuuden tukeminen sekä tietojärjestelmien kehittäminen ja ylläpito. (Ruokavirasto 2019.)

Elintarvikkeiden turvallisuus ja laatu, eläinten terveys ja hyvinvointi, kasvinterveys sekä maa- ja metsätalouden tuotannossa käytettävät lannoitevalmisteet, rehut ja kasvinsuojeluaineet sekä lisäysaineistot kuten siemenet ja taimiaineistot ovat asioita, joiden edistäminen, valvonta ja tutkimus on Ruokaviraston vastuulla. Tämän lisäksi Ruokavirasto on EU:n maksajavirastona vastuussa EU:n maataloustuki- ja maaseuturahastojen varojen käytöstä Suomessa. Sekä EU-tukien että kansallisten tukien (kuten viljelijätukien, hanke-, yritys- ja rakennetukien sekä markkinatukien) toimeenpano on Ruokaviraston vastuulla. Näiden päätehtävien lisäksi Ruokavirastossa kehitetään ja ylläpidetään maaseutuelinkeinohallinnon tietojärjestelmiä sekä oman toimialan rekistereitä. Ruokavirasto myös kehittää sähköisiä asiointipalveluita ja tuottaa maa- ja metsätalousministeriön hallinnonalan virastoille ja laitoksille sekä muille julkishallinnon tahoille tietohallinnon palveluita. (Ruokavirasto 2019.)

2.4 Kehittämisympäristö

Eläinten terveyden hallinta eli ELTE-hankkeen päämääränä on eläinten terveyden hallinnan tehokkaan toimintamallin määrittely sekä tiedonkulkua, toimintaa ja toiminnan kehittämistä tukevien tietojärjestelmien käyttöönotto. Ensisijaisesti on tarkoitus varmistaa, että vastustettavien eläintautien ja eläinten lääkitsemisen valvonnassa tarvittavat hallintatoimet onnistuvat viranomaisketjussa. (Pajala 2019.)

Hankkeeseen kuuluvan ELTE-projektin tavoitteena on toteuttaa ja ottaa käyttöön eläinten terveyden hallintajärjestelmä nimeltään ELTE. Sähköisellä järjestelmällä eläinlääkintähuollon viranomaiset voivat saada, hallinnoida ja raportoida tietoja eläintautiepäilyistä ja -tapauksista, seurannan järjestämisestä, eläinten lääkinnän valvonnasta ja lakisääteisestä eläinten terveysvalvonnasta. ELTE-projektiin kuuluu ELTE-hankkeen kaksi ensimmäistä osa-aluetta, jotka ovat tautitapauksen ja epidemian hallinta sekä rajoittavat määräykset. ELTE-tietojärjestelmälle on määritelty MVP (Minimum Viable Product), jonka ensimmäinen versio on tarkoitus julkaista syksyllä 2020. (Pajala 2019.)

2.5 Keskeiset käsitteet

Käsitteet on ryhmitelty aihealueittain, jotta toisiinsa liittyvät termit seuraavat toisiaan.

JIRA tehtävienhallintaohjelmisto

Issue JIRA:ssa käytettävä nimitys tehtävälle, esimerkiksi taskille, bugille, kehitysehdotukselle

Epic JIRA:n laajin issue, projektissa vastaa suuruudeltaan järjestelmän käyttötilannetta

Story JIRA:n issue, epiciä pienempi, mutta voidaan yhä jakaa pienemmiksi tehtäviksi

Hakusuodatin JIRA:n tallennettu haku, filter

JQL JIRA Query Language, JIRA:n kyselykieli, jolla voi tehdä monimutkaisia hakuja

Labeli JIRA:n issueihin liitettävissä oleva kenttä, jolla voidaan merkitä issueita tietyn tyyppisiksi, esimerkiksi testattaviksi

Ohjausnäkymä näkymä, jolle voi kerätä JIRA:sta tarvitsemiansa tietoja vempainten avulla, dashboard

Vempain ohjausnäkymän pienoisohjelma, joka näyttää määritellyt tiedot tietyllä tavalla, gadget

Xray JIRA:n testauksenhallintaan tarkoitettu laajennus

Test Xray:n issue, testitapaus, johon on määritelty testin suorittamisen vaiheet

Test Execution Xray:n issue, testin suoritussuunnitelma, sisältää 1-n kappaletta testejä

Test Plan Xray:n issue, testisuunnitelma, sisältää 1-n kappaletta testien suoritussuunnitelmia

Requirement Status Xray:n tuoma kenttä vaatimustyyppisiin issueihin, vaatimuksen kattavuusstatus, eli kuvaa sitä, miten vaatimus on katettu testeillä. Voi olla OK, NOK, NOTRUN, UNCOVERED, UNKNOWN

Testrunstatus Xray:n tuoma kenttä testeihin ja testien suoritussuunnitelmiin, testiajon status. Voi olla PASS, FAIL, TODO, EXECUTING, ABORTED

Scrum Ketterän ohjelmistokehityksen viitekehys

Järjestelmätestaus Kokonaiseen järjestelmään kohdistuvaa testausta testiympäristössä

Regressiotestaus Uudelleentestaus, jossa tarkistetaan jo testattujen osien toimivuus uudessa versiossa

Tutkiva testaus Järjestelmää kokeilevaa testausta ilman testitapauksia

Häiriö Testauksessa löydetty ero odotetun ja todellisen lopputuloksen välillä, bugi, defect

Vikaraportti Testauksessa löydetystä häiriöstä tehty raportti

Teemahaastattelu Haastattelu, joka etenee teemojen, ei tarkkojen kysymysten varassa

Kysely Lomake, jolla kysytään samaa asiaa samalla tavalla suurelta joukolta vastaajia

Osallistuva havainnointi Tiedonkeruumenetelmä, jossa tutkija on osallisena toiminnassa, josta tekee havaintoja

Confluence Ohjelmisto tiedon yhteisölliseen tuottamiseen ja jakamiseen

3 Ketterä ohjelmistokehitys

Ohjelmistokehityksessä käytetyssä perinteisessä vesiputousmallissa ohjelmistotuote valmistuu vasta kaikkien projektin vaiheiden jälkeen. Ketterä ohjelmistokehitys pyrkii välttämään tällaisessa tilanteessa mahdollisesti esiintyvät ongelmat, kuten että alun perin suunniteltu ohjelmisto on jo valmistuessaan vanhanaikainen tai ei sittenkään vastaa haluttua toteutukseltaan. Tyypillisesti ketterässä ohjelmistokehityksessä valmistuu jokaisen sprintin eli kehitysvaiheen jälkeen käyttökelpoinen tuote, jota sitten seuraavalla sprintillä kehitetään eteenpäin. Näin ohjelmiston tilaaja näkee jatkuvasti, mitä on tulossa ja pystyy tekemään radikaalejakin muutoksia alkuperäisiin suunnitelmiin.

Ketterien menetelmien pohjalla on vuonna 2001 tehty julistus, jonka kirjoitti 17 ohjelmistokehityksen menetelmien muutosta ajavaa henkilöä. Tavoitteena oli löytää parempia tapoja tekemiseen. Julistuksessa on neljä arvoa ja 12 periaatetta. Arvojen mukaan yksilöitä ja kanssakäymistä tulisi arvostaa enemmän kuin menetelmiä ja työkaluja, toimiva ohjelmisto on tärkeämpi kuin kattava dokumentaatio, asiakasyhteistyö tuottaa paremman tuloksen kuin sopimusneuvottelut ja muutoksiin vastaaminen auttaa kehitystyötä etenemään paremmin kuin suunnitelmassa pitäytyminen. (Agile Manifesto 2001.)

Periaatteissa mainitaan esimerkiksi sellaisia asioita kuin että ohjelmasta toimitetaan toiveiden mukaisia versioita kehityksen aikaisessa vaiheessa, säännöllisesti ja lyhyellä aikavälillä. Muutokset otetaan vastaan kaikissa vaiheissa ja tavoitteena on yhteistyö asiakkaan hyväksi. Keskustelu on tehokkain tiedon välitystapa, ja toimiva ohjelmisto kertoo eniten työn

edistymisestä. Tiimit ovat omatoimisia ja itseorganisoituvia, ja kehittävät säännöllisesti omaa toimintaansa. (Agile Manifesto 2001.)

3.1 Scrum

Tässä opinnäytetyössä tarkastellaan ELTE-projektin toimintaa, joka perustuu osittain Scrum-viitekehykseen. Scrum on iteratiivis-inkrementaalista eli toistavaa ja lisäävää, ja se nojaa kolmeen peruspilariin, joita ovat läpinäkyvyys, tarkastelu ja sopeuttaminen. Läpinäkyvyys on sitä, että kaikilla on yhteisymmärrys tavoiteltavasta lopputuloksesta. Tarkastelulla tarkoitetaan sitä, että tuotoksia ja työn edistymistä arvioidaan, jotta poikkeamat eivät jää huomaamatta. Sopeuttaminen tapahtuu tarkastelun tuloksena, kun prosessissa tai tuotteessa huomataan jotain, mitä ei voida hyväksyä. (Schwaber & Sutherland 2017, 4-5.)

Tuoteomistaja, kehitystiimi ja scrummaster muodostavat Scrum-tiimin. Tiimi on itseohjautuva ja monitaitoinen, eli tiimi ei tarvitse ulkopuolista ohjausta tai osaamista työn suorittamiseen. Tuoteomistaja vastaa kehitysjonon hallinnasta tuottaen sinne selkeästi ilmaistuja tehtäviä riittävällä tarkkuudella haluttuun järjestykseen. Tuoteomistaja tekee kaikki kehitysjonoa koskevat päätökset. Kehitysjonon muuttaminen tuotteeksi on kehitystiimin tehtävä. Kehitystiimi tuottaa yhdessä sovitun valmiin määritelmän mukaisen tuotteen joka sprintissä. Scrummasterin tehtävänä on edistää ja tukea Scrum-menetelmän käyttöä tiimissä. (Schwaber & Sutherland 2017, 5-7.)

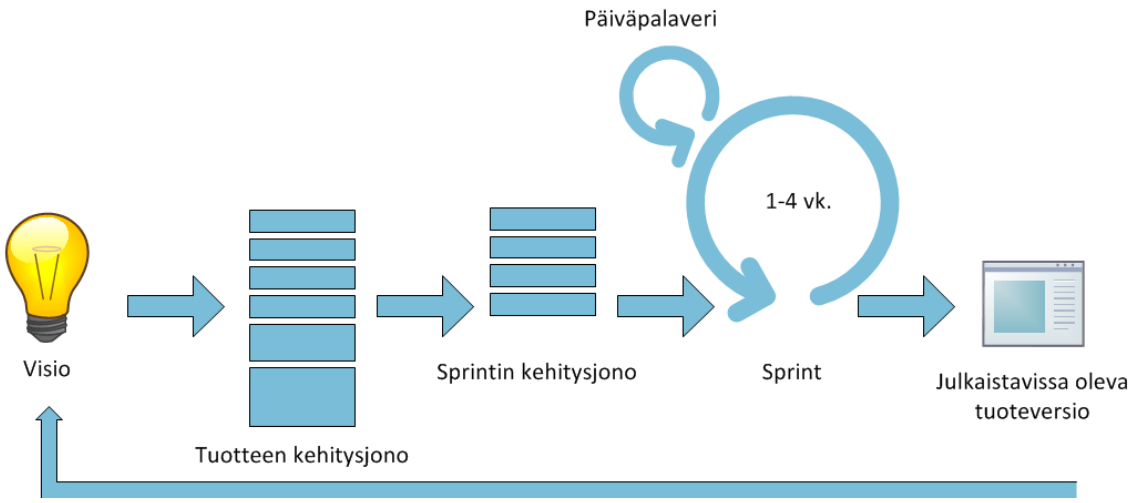
Scrumin tapahtumia ovat sprintti, sprintin suunnittelu, päivittäispalaveri, sprintin katselmointi ja sprintin retrospektiivi. Sprintti on enintään kuukauden pituinen ajanjakso, jonka aikana kehitystyö tapahtuu ja jonka tuloksena on käyttökelpoinen tuote. Sprintti sisältää muut Scrumin tapahtumat. Sprintin suunnittelussa päätetään tiimin kesken, mitä on mahdollista toimittaa sprintin aikana ja miten työ toteutetaan. Päivittäispalaverin pituus on enintään 15 minuuttia ja siinä käydään läpi seuraavan 24 tunnin tehtävät. Tavoitteena on hyötyä työn edistymisen tarkastelusta. Sprintin katselmoinnissa tiimi tarkastelee kehitettyä työtä ja mahdollisesti esittelee tuotosta sidosryhmille tavoitteenaan lisätä vuoropuhelua ja saada palautetta. Retrospektiivissä keskustellaan sprintin sujumisesta ja mahdollisista kehityskohteista toiminnan suhteen. Tavoitteena on tunnistaa hyvät toimintatavat ja suunnitella parannuksia. (Schwaber & Sutherland 2017, 9-14.)

Scrumin tuotoksia ovat tuotteen kehitysjono, sprintin kehitysjono ja inkrementti. Kehitysjonoon on listattuna kaikki tuotteeseen toteutettavat ominaisuudet, toiminnot, vaatimukset, parannukset ja korjaukset. Sprintin kehitysjonossa on sprintin aikana toteutettaviksi valittuja kohtia kehitysjonosta. Inkrementillä tarkoitetaan kehitettyä tuotetta eli se on summa aiemmissa sprinteissä valmistuneita ja sprintin aikana valmistuvia kehitysjonon kohtia. (Schwaber & Sutherland 2017, 15-17.)

3.2 Ruokaviraston ohjelmistotuotantomalli

Ruokavirastolla on IT-palveluiden kehittämiseen suositeltu ohjelmistotuotantomalli, joka perustuu kokemukseen Maa- ja metsätalousministeriön hallinnonalalla tehdyistä projekteista, yleisiin alan hyviin käytäntöihin ja ohjelmistotuotannon teoriaan. Mallin ydin on Scrum-menetelmä, joka on laajennettu hyväksi havaituilla kuvauksilla ja tekniikoilla. Scrum-menetelmän määrittelemät dokumentit ovat tuotteen kehitysjono ja sprintin tehtävälista, muiden dokumenttien tuottamiseen pitäisi olla selkeä tarve. Tavoitteena on ennemmin projektin henkilöiden hyvä kommunikaatio kuin kattava dokumentaatio. (Turpeinen 2019.)

Kuviossa 1 kuvataan tavoiteprosessi, jonka mukaan ohjelmistotuotantoprojektien halutaan toteutuvan. Tuotteen kehitysjonosta valitaan sprintin kehitysjonoon asioita, jotka toteutetaan sprintin aikana ja lopputuloksena on julkaisukelpoinen versio, minkä jälkeen sykli alkaa alusta.



Kuvio 1: Ohjelmistotuotannon prosessi (Turpeinen 2019.)

Tuotteen kehitysjonoon kuvataan järjestelmän vaatimukset sekä kaikki ominaisuudet, toiminnot, parannukset ja korjaukset. Kehitysjono on siis järjestetty lista järjestelmässä mahdollisesti tarvittavista asioista, sitä päivitetään jatkuvasti tuoteomistajan toimesta ja työstetään yksityiskohtien, työmääräarvioiden ja prioriteettien osalta yhdessä kehitystiimin kanssa. Ruokavirastossa tuotteen kehitysjonoa ylläpidetään JIRA-tehtävienhallintaohjelmistossa ja tekstimuotoinen dokumentaatio tuotetaan Confluenceen. (Turpeinen 2019.)

3.3 Ruokaviraston projektinhallintamalli

Ruokavirastossa on käytössä projektinhallintamalli, joka on yleismalli kaikille virastossa toteutettaville projekteille. Malli on yhdistelmä SFS-ISO 21500 standardia ja Ruokaviraston edeltäjien Eviran, Mitpan ja Mavin projektityön ohjeistuksia ja käytäntöjä. (Leppälä 2019.)

Kansainvälinen SFS-ISO 21500 -standardi antaa ohjeita projektinhallinnasta ja se sopii niin julkisten kuin yksityisten organisaatioiden käyttöön kaikentyyppisiin projekteihin sovellettavaksi niiden monimutkaisuudesta, koosta tai kestosta riippumatta. Standardista löytyvät yleistasoiset kuvaukset käsitteistä ja prosesseista, jotka muodostavat projektinhallinnan hyvät käytännöt. (SFS 2019.)

SFS-ISO-standardin mukainen termi ohjelma (rinnakkaistermi hanke) tarkoittaa koordinoidusti johdettua, määräaikaista kokonaisuutta, joka koostuu projekteista ja ohjelman tehtävistä, ja jonka tavoitteena on merkittävä muutos (strateginen päämäärä). Projektisalkku koostuu projekteista, jotka on laitettu tärkeysjärjestykseen ja jotka ovat samanaikaisesti käynnissä. (Leppälä 2019.)

Projekti määritellään ajallisesti rajatuksi ja ainutkertaiseksi tehtäväkokonaisuudeksi, jonka tavoitteena on määritelty lopputuotos. Tavoiteltavat hyödyt, tavoitteet ja tuotokset sekä aikataulu- ja kustannustavoitteet määritellään. Projektille nimetään tilapäinen organisaatio, joka on erillinen linjaorganisaatiosta. Ruokavirastossa on hyvin erilaisia projekteja eikä kaikkia kehitystoimenpiteitä toteuteta projekteina vaan hoidetaan linjatyönä. (Leppälä 2019.)

Projektinhallintaan liittyy ohjausryhmä, projektiryhmä ja kehitystiimi. Ohjausryhmä on projektin omistajan johtama ryhmä, joka varmistaa tarvittavat resurssit sekä ohjaa ja tukee projektityötä. Projektiryhmää johtaa projektipäällikkö, ja ryhmän pääasiallisena vastuualueena on projektin konkreettinen toteutus. Kehitystiimi kehittää tuoteversion omien työmenetelmiensä avulla. (Leppälä 2019.)

Projekti jaetaan kuuteen vaiheeseen, ja jokaisen vaiheen alussa ja lopussa on porttipäätöksentekopiste, jolloin päätetään vaiheen päätöksestä ja projektin etenemisestä seuraavaan vaiheeseen. Projektin vaiheita ovat tarve, valmistelu, suunnittelu, ohjaus ja toteutus, päättäminen ja jälkiseuranta. (Leppälä 2019.)

4 Ohjelmistotestaus

Ohjelmistotestauksen tavoitteena on varmistaa, että toteutettava ohjelmistotuote on toivotun kaltainen ja sen ominaisuudet toimivat tarkoituksenmukaisesti. Testauksen avulla tarkistetaan, että tehty työ vastaa sitä, mitä pitikin tehdä sekä tunnistetaan, milloin tuotos eroaa suunnitellusta. (Kasurinen 2013, 10.) Testauksen avulla arvioidaan myös tehdyn tuotteen laatua ja vähennetään riskejä tai pienennetään niiden vakavuusastetta. (ISTQB 2018, 12.)

Testauksen tarkoitus on tuoda esiin häiriöt, joita ohjelmistossa olevat viat ovat aiheuttaneet (ISTQB 2018, 13). Häiriöllä tarkoitetaan sitä, että testin odotettujen ja todellisten tulosten välillä on eroavaisuus. Jos testaaja huomaa poikkeaman tuloksissa, häiriöstä tuotetaan

vikaraportti. Jos taas häiriö jää huomaamatta, kyseessä on väärä negatiivinen tulos. (ISTQB 2015, 52.) Häiriö ei aina johdu ohjelman viasta, vaan se voi johtua testiympäristöstä, testiaineistosta tai jopa testaajasta. Tällöin kyseessä on väärä positiivinen tulos, ja vaikka tällaisten häiriöiden raportoimista halutaan välttää, ei vikojen raportoimatta jättämistä pidä kannustaa. Silloin vianlöytämisen tehokkuus laskee. (ISTQB 2015, 53.)

ISTQB:n (ISTQB 2018) mukaan testauksessa on seitsemän perusperiaatetta. Ensimmäinen periaate on, että testauksella voidaan osoittaa, että virheitä on olemassa, mutta ei sitä, ettei niitä ole. Eli vaikka testauksessa ei löydetä vikoja, se ei silti tarkoita sitä, ettei niitä ole olemassa. Toinen periaate on, että kaikkea ei voida testata. Kaikkia mahdollisia vaihtoehtoja on mahdotonta testata, joten riskianalyysin, testaustekniikoiden ja priorisoinnin avulla pyritään kohdistamaan testaus oikein. Kolmas periaate on, että aikaisin aloitettu testaus säästää aikaa ja rahaa, sillä virheiden korjaaminen aikaisessa vaiheessa on yleensä halvempaa ja helpompaa. Neljännen periaatteen mukaan viat kasaantuvat, eli pieni ohjelmanosa aiheuttaa suurimman osan löydetyistä vioista. Nämä testauksen aikana löydetyt tai riskianalyysissa ennustetut vikakeskittymät auttavat kohdentamaan testausta ongelma-alueelle. Viidennen periaatteen mukaan testejä ja testiaineistoja tulee ajan myötä muuttaa tai vaihtaa. Testeille käy nimittäin kuin hyönteismyrkylle; niiden teho heikkenee, kun testejä toistetaan uudelleen ja uudelleen muuttamattomina. Kuudes periaate on, että tilanne vaikuttaa testaukseen. Testausta tehdään eri tavalla eri tilanteessa, esimerkiksi erilaisille ohjelmistoille on erilaiset testauskriteerit. Viimeisen periaatteen mukaan virheiden poissaolo on harhaluulo, sillä kuten kaksi ensimmäistä periaatetta osoittavat, kaikkia virheitä ei voida löytää eikä kaikkea voida testata. Tämän lisäksi se, että virheitä ei ole, ei takaa sitä, että järjestelmä on onnistunut, sillä se voi silti olla hankalakäyttöinen tai ei täytä käyttäjien tarpeita. (ISTQB 2018, 15.)

4.1 Testauksen tasot

Ohjelmistotestaus voidaan jakaa neljään tasoon: yksikkö-, integrointi-, järjestelmä- ja hyväksymistestaukseen. Yksikkötestauksessa testataan ohjelmiston yksittäisiä komponentteja ja integrointitestauksessa taas varmistetaan, että nämä yksittäiset komponentit toimivat yhdessä. Järjestelmätestauksessa testataan kokonaista järjestelmää ja sen ominaisuuksia testiympäristössä, kun taas hyväksymistestauksessa käytetään aitoa tai aidonkaltaista ympäristöä. Hyväksymistestaus suoritetaan ennen järjestelmän käyttöönottoa. (Juvonen 2018, 26-28.) Tässä opinnäytetyössä käsiteltävä testaus on järjestelmätestausta.

Järjestelmätestauksessa testauksen kohteena ovat järjestelmän ominaisuudet ja kokonaisvaltainen käyttäytyminen. Siinä tutkitaan järjestelmän toimintoketjuja alusta loppuun saakka sekä tarkkaillaan niiden aikana tapahtuvaa ei-toiminnollista käyttäytymistä. Tavoitteena on varmistaa, että järjestelmä toimii suunnitellun ja määritellyn mukaisesti, tarkistaa sen valmiusaste ja laatu, pienentää riskejä sekä estää vikojen eteneminen tuotantoon. Testauksen

suoritusympäristön pitäisi vastata mahdollisimman paljon järjestelmän todellista toimintaympäristöä. Järjestelmätestauksen suunnittelun materiaalina käytetään järjestelmän vaatimusmäärittelyjä, riskianalyysiraportteja, käyttötapauksia ja käyttäjätarinoita, järjestelmän käyttäytymismalleja, tilakaavioita ja käyttöohjeita. (ISTQB 2018, 31.)

4.2 Testausmenetelmät

Testausprosessi on tilannesidonnainen, mutta voidaan sanoa, että se yleisesti muodostuu testauksen suunnittelusta, testauksen seurannasta ja hallinnasta, testianalyysista, testien suunnittelusta, testien valmistelusta, testien suorituksesta, testauksen edistymisen ja tulosten raportoinnista sekä testauksen päättämisestä. Ketterässä kehityksessä suunnittelun ja toteutuksen lisäksi myös testaus suoritetaan iteratiivisesti, joten nämä vaiheet toistuvat tarpeiden ja tilanteiden mukaan sovellettuina. (ISTQB 2018, 16-17.)

Erilaisia testausmenetelmiä käytetään järjestelmän eri elinkaarivaiheissa. Tässä opinnäytetyössä keskitytään järjestelmän kehitysvaiheeseen, joten yleensä alkuvaiheessa, ylläpidossa tai juuri ennen julkaisua käytettäviä testausmenetelmiä ei esitellä.

Testaus voidaan jakaa staattiseen ja dynaamiseen testaamiseen. Staattisessa testauksessa järjestelmää testataan ilman sen käyttämistä eli sitä tutkitaan esimerkiksi analysoimalla koodia. Dynaamisessa testauksessa järjestelmää sen sijaan käytetään ja sen toimintoja seurataan. Staattista testausta voidaan tehdä jo järjestelmän suunnitteluvaiheessa ja siinä löydetään suoraan järjestelmässä olevia vikoja, ei vikojen aiheuttamia häiriöitä. Dynaamisessa testauksessa tarkastellaan järjestelmän kokonaisuuden tai sen osien toimintaa käytännössä. (Kasurinen 2013, 65.) Tässä opinnäytetyössä käsiteltävä testaus on dynaamista.

Testaus voi olla joko mustan laatikon, lasilaatikon tai harmaan laatikon testausta. Mustan laatikon testaus tarkoittaa sitä, ettei järjestelmän sisäistä toimintaa tarkastella, vaan järjestelmälle annetaan syötteitä ja katsotaan, mitä se antaa palautteeksi. Jotta voidaan varmistua siitä, että syötteet ovat oikeanlaisia ja lopputulos on toivotunlainen, tulee käyttää mahdollisimman monenlaisia syötteitä, myös haitallisia tai virheellisiä, jotta virhetilanteet voidaan tarkistaa. (Kasurinen 2013, 65-66.) Opinnäytetyössä tarkasteltavassa projektissa suurin osa testauksesta tehdään mustan laatikon testauksena.

Jos testauksen aikana tarkastellaan myös järjestelmän sisäistä toimintaa, kyseessä on lasilaatikkotestaus. Järjestelmälle siis annetaan syötteitä ja sen lisäksi, että katsotaan järjestelmän antamaa palautetta, tarkastellaan samalla, mitä järjestelmän sisällä tapahtuu. Lasilaatikkotestauksessa varmistetaan, että järjestelmä näyttää toimivan oikein ja että oikea tulos ei ole pelkästään sattumaa. Tällaisessa testauksessa löytyvä vika pystytään jäljittämään lähdekoodin tasolle saakka, mutta testaus vaatii ohjelmointityön ja järjestelmän logiikan tuntemusta. (Kasurinen 2013, 67.)

Näiden kahden testauksen yhdistelmä on harmaan laatikon testaus, jossa yhdistyy molempien parhaat puolet. Menetelmällä pyritään samaan aikaan testaamaan, että järjestelmä täyttää sille asetut vaatimukset ja että sen lähdekoodi on toimiva. (Kasurinen 2013, 68.)

Regressiotestaus tarkoittaa suunnilleen samaa kuin uudelleentestaus, eli kun järjestelmän jotain osaa muutetaan, halutaan muutoksen jälkeen varmistaa, että jo aiemmin testatut osat toimivat yhä halutulla tavalla. Regressiotestaukseen voidaan valita tilanteen mukainen, sopivaksi katsottu kokonaisuus, eli testaus voidaan keskittää tiettyyn toimintoon, jonka toimivuuden epäillään vaarantuneen muutoksen takia, tai sitten koko järjestelmä voidaan testata muutoksen jälkeen. Regressiotestauksen tärkein tehtävä on tarkistaa, etteivät korjatut viat enää esiinny ja ettei mitään uutta ole rikkoontunut. Se ei ole testauksen taso tai menetelmä, vaan yleisnimi testaukselle, joka tähtää järjestelmän uuden version toimivuuden varmentamiseen. (Kasurinen 2013, 68-69.) Ketterässä kehittämisessä regressiotestaukselle on koko ajan tarvetta, sillä joka sprintin jälkeen tuotetaan uusi tuote, jonka pitää toimia halutulla tavalla. Regressiotestaus monesti pyritään automatisoimaan, jotta manuaalisessa testauksessa voidaan enemmän keskittyä uusien ominaisuuksien testaamiseen.

Tutkivaa testausta ei projektissa tehdä mitenkään tarkoituksenmukaisesti, mutta järjestelmää kokeillaan myös testien tekemisen ulkopuolella joko testitapausten suunnittelun tai järjestelmään tutustumisen takia. Tämä testausmenetelmä ei perustu järjestelmän testaamiseen minkään mallin tai suunnitelman mukaan, vaan lähestymistapa hyödyntää testaajien ammattitaitoa ja ymmärrystä riskeistä eli häiriöiden mahdollisuuksista tietyissä järjestelmän osissa. (Kasurinen 2013, 74.) Tällainen menetelmä, joka hyödyntää testaajan osaamista, intuitiota ja kokemusta vastaavanlaisista toteutuksista, on kokemuspohjainen menetelmä (ISTQB 2014, 38). Tiedon keräys vioista ja tuotteesta sekä testaustyön suunnittelu voidaan toteuttaa tutkivan testauksen avulla. Testejä laaditaan ja suoritetaan dynaamisesti ja samalla raportoidaan niiden tuloksia. Sitä käytetään usein muun testauksen lisänä. (ISTQB 2014, 40.) Tällaista testausta ei kuitenkaan pystytä dokumentoimaan eikä sen kattavuutta seuraamaan, minkä lisäksi tapahtuneiden häiriöiden jäljittäminen ja toistaminen saattaa olla hankalaa. (Kasurinen 2013, 74.) Tällä kuitenkin voidaan löytää häiriöitä, joita määrittelypohjaiset testaustekniikat eivät löydä (ISTQB 2014, 41).

Tutkiva testaus eroaa ad hoc -testauksesta sillä, että tavoitteena voi olla löytää uusia testitapauksia tai etsiä ongelma-alueita. Ad hoc -testaus on suunnittelematonta ja dokumentoimatonta testausta, jossa testataan pikaisesti, että kaikki näyttää toimivan. (Kasurinen 2013, 74.) Koska tutkivaa testausta ei ole mahdollista raportoida systemaattisesti, sitä ei oteta huomioon opinnäytetyössä raportointikäytänteitä luotaessa. Tällä menetelmällä kuitenkin löydetään välillä häiriöitä, joten tämän esittely häiriöiden raportointiin liittyen on oleellista.

4.3 Vikaraportti

Testauksen aikana ilmenneestä häiriöstä kirjoitetaan vikaraportti. Raportin tarkoituksena on välittää tietoa kuvaamalla häiriön löytymishetken tilanne, sen toistamiseen tarvittavat askeleet ja aineisto, sekä odotettu ja todellinen lopputulos. (ISTQB 2014, 55.) Hyvän vikaraportin tunnusmerkkejä on, että se on täydellinen, tiivis, tarkka ja objektiivinen. Täydellisellä tarkoitetaan sitä, että siitä löytyy kaikki tarvittava tieto eikä mitään ole jätetty pois, kun taas tiivis raportti ei sisällä epäoleellisia tietoja. Raportti on tarkka, kun siinä kuvaillaan selkeästi sekä odotettu että todellinen tulos ja kerrotaan askeleet, joilla tilanteen voi toistaa. Objektiivisuus ilmenee, kun raportti on kirjoitettu ammattimaisesti ja faktoihin keskittyen. Vikaraportin muoto kannattaa määritellä mahdollisimman tarkkaan, jotta sen täyttäminen ja niistä luotavien yhteenvetoraporttien laatiminen on helppoa. (ISTQB 2014, 54.)

Vikaraportti kirjoitetaan ensisijaisesti siksi, jotta vika saadaan korjattua, mutta sen sisältämiä tietoja voidaan käyttää myös vikojen luokitteluun, riskianalyysin laatimiseen tai testausprosessin kehittämiseen. (ISTQB 2014, 55.) Vikaraporttiin kerätään tietoja kolmea tarkoitusta varten; vikaraportin hallintaa, projektin laadun ja testauksen edistymisen arviointia sekä testausprosessin arviointia varten. Kerätyistä tiedoista voidaan tehdä esimerkiksi vikatiheysanalyysi, löydettyjen ja ratkaistujen vikojen trendianalyysi tai vikojen keskimääräisen elinajan analyysi. (ISTQB 2015, 54.)

Vikaraportointiin kuuluu keskeisesti vikojen luokittelu, jota käytetään vikojen ryhmittelyyn, testauksen tai kehityselinkaaren tehokkuuden arviointiin sekä mielenkiintoisten suuntausten tunnistamiseen. Luokitteluun vaikuttaa usein vikaraportin elinkaarivaihe. Vastalöydetty vika voidaan luokitella toistettavuuden, arvioidun syyn, oireiden, vian syntymisvaiheen tai löytämisvaiheen mukaan. Vikaluokkia voidaan lisätä vikaraportin tutkimisen jälkeen, sille voidaan antaa esimerkiksi alkuperäissyy eli miksi vika syntyi tai lähde eli missä vika on syntynyt. Lopuksi vika voidaan luokitella käsittelyn jälkeen lopputuloksen tai korjaustoimenpiteen mukaan. (ISTQB 2014, 55-56.)

Vakavuuden ja kiireellisyyden mukaan luokitteleminen on hyvin yleistä. Voidaan myös arvioida vikojen vaikutukset aikatauluun, kustannuksiin, riskeihin, turvallisuuteen tai laatuun, ja luokitella niiden perusteella. Tällaiset luokittelut voivat auttaa päättämään, miten pian vika otetaan korjattavaksi. Jos taas halutaan seurata vikojen elinkaarta, silloin lopputuloksen mukaan luokitteleminen on hyvä vaihtoehto. Jokainen organisaatio ja projekti voi käyttää omaan toimintaansa sopivia luokitteluja. Vikoja ei kannata luokitella monen aspektin mukaan, tällöin vikaraportin täyttäminen monimutkaistuu. Räätälöidyt, tarkkaan mietityt ja yhdenmukaiset luokitteluarvot sen sijaan tuovat hyötyä projektille. (ISTQB 2014, 56.)

4.4 Testauksen seuranta ja raportointi

Testausta seurataan, jotta siitä saadaan kerättyä tietoa, annettua palautetta sekä tuotua testausprosessia näkyväksi. Tiedonkeruu voi olla automaattista tai manuaalista, ja tietojen avulla arvioidaan testauksen edistymistä tai testauksen päätöskriteerien täyttymistä. Raportointi voi myös auttaa priorisoimaan tiettyjen ohjelman osioiden testaamista tai muuttamaan testausaikataulua. (ISTQB 2018, 63.) Testauksen raportoinnin tehtävänä on vetää yhteen ja antaa tiedoksi testaustehtäviin liittyviä tietoja niin testauksen aikana kuin sen jälkeenkin.

Testauksen aikainen raportti on testauksen edistymisraportti ja testauksen jälkeistä kutsutaan testauksen yhteenvetoraportiksi. Tyypillisesti näiden raporttien sisältö vaihtelee organisaation vaatimusten, projektin ja sen elinkaarivaiheen mukaan. Tämän lisäksi raportointi tulee räätälöidä kohderyhmälle sopivaksi, eri sidosryhmät tarvitsevat eritasoista raportointia. Testauksesta raportoidaan yleensä yhteenveto suoritetusta testauksesta sekä mahdolliset poikkeamat, edistymistä haittaavat seikat sekä vikoihin, testeihin ja kattavuuteen liittyvät tiedot. (ISTQB 2018, 64.)

Erilaisia mittareita voidaan käyttää testaustoiminnan arviointiin ja seuraamiseen, kunhan mittari on toistettava, tarkka, vertailukelpoinen ja taloudellinen. Toistettavalla mittarilla saadaan aina sama tulos, kun käytössä on sama mittausväli ja sama menetelmä. Mittari on taas tarkka, jos tulos on järkevä ja hyödynnettävissä oleva mitattavasta suureesta. Vertailukelpoista mittaria voidaan käyttää eri projektissa tai järjestelmissä. Lopuksi, mittari on taloudellinen, kun sitä varten tarvittavien tietojen hankkiminen ja koostaminen tuo enemmän hyötyä, kuin mitä siitä on työtä. (Kasurinen 2013, 162.) Mittareita valitessa tulisi kiinnittää huomiota siihen, että ne tuottavat tuloksia tehdystä työstä eli niiden keräämä tieto kertyy automaattisesti työn tekemisestä ja työkalujen käyttämisestä. (Kasurinen 2013, 167.)

ISO/IEC 29119 -standardissa mittarit jaotellaan seitsemään luokkaan sen mukaan, mikä on mittarin käyttämän tiedon lähde. Mittarit voivat perustua tietoon, jota saadaan raakadatasta eli projektin perustiedoista kuten budjetista, työajanseurannasta tai testitapaustietokannasta, työn suunnitteluun liittyvistä suunnitelmista, projektin senhetkisestä tilasta kertovista tilastoista, järjestelmän kattavuudesta kertovasta datasta, testituloksista, järjestelmän luotettavuudesta eli toimintavarmuudesta kertovista tilastoista tai testauksen tehokkuudesta kertovista arvioista. (Kasurinen 2013, 162-167.)

Mittarit voidaan myös jakaa projekti-, tuote-, prosessi- ja ihmismittareihin. Projektimittareilla arvioidaan testauksen edistymistä lopetuskriteereihin nähden, tuotemittareilla mitataan jotain tuotteen ominaisuutta, prosessimittareilla testausprosessin kyvykkyyttä ja ihmismittareilla ihmisten kyvykkyyttä. (ISTQB 2012, 37.)

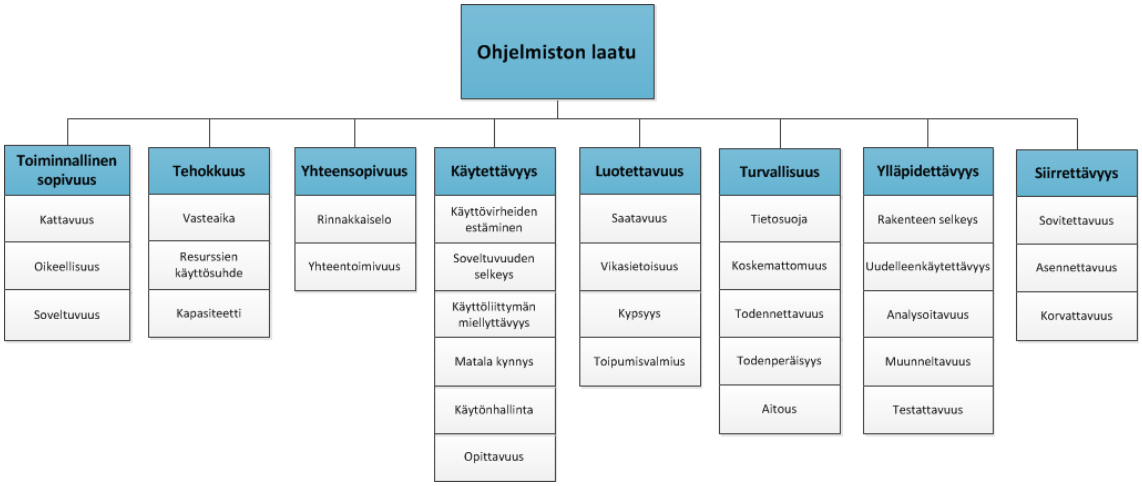
Mittareita valitessa tulee muistaa, että ne tulee määritellä projektin, prosessin ja/tai tuotteen tavoitteiden mukaan keskittyen oleellisimpiin, toisiaan tasapainottaviin ja täydentäviin mittareihin. Mittareiden tietojen kerääminen ja tulosten raportointi kannattaa automatisoida mahdollisimman pitkälle, ja tuloksia kannattaa seurata sekä analysoida niissä ilmeneviä poikkeamia. Raportoinnissa tulee ottaa tavoitteeksi, että tieto on välittömästi ymmärrettävissä. Raportoinnissa voidaan keskittyä esittämään tilanne tiettynä ajankohtana tai tarkastella sen kehitystä. Raportoitavat tiedot kannattaa myös katselmoida ja validoida, jotta voidaan varmistaa tietojen oikeellisuus ja ilmentämä tilanne. (ISTQB 2012, 38.)

Mittaritiedon keräämisen tarkoituksena on arvioida testauksen etenemistä aikataulun ja budjetin suhteen, testattavan järjestelmän sen hetkistä laatua, valittujen testaustapojen riittävyyttä sekä testitapausten tehokkuutta tavoitteisiin suhteutettuina. (ISTQB 2018, 63.) Testauksen edistymistä voidaan seurata riskien, vikojen, testien, kattavuuden tai luottamuksen suhteen. Riskien suhteen voidaan arvioida, kuinka suuri osa riskeistä on katettu testeillä ja kuinka paljon on vielä testaamatta. Vikojen suhteen voidaan tarkastella löydettyjen vikojen määriä, niiden ominaisuuksia tai tarkastella löydettyjen ja ratkaistujen vikojen suhdetta. Testeistä voidaan seurata esimerkiksi suoritusmääriä tai onnistumisprosentteja. Kattavuutta voidaan mitata testien kattamien vaatimusmäärittelyjen tai koodikattavuuden suhteen. Luottamusta mitataan usein subjektiivisesti, mutta kattavuutta voidaan käyttää myös hyväksi sen arvioinnissa. (ISTQB 2012, 38-39.)

Testauksesta kerätystä tiedosta ja havainnoista ei kuitenkaan ole hyötyä, ellei niitä käytetä toiminnon ohjaamiseen ja kehittämiseen. (Kasurinen 2013, 168.) Testauksen hallinnan tehtävänä on pitää huoli, että testaus saavuttaa sille asetetut tavoitteet. Jos raportointi paljastaa poikkeamia testaussuunnitelmasta, tulee tarpeen mukaan tarkastaa testien prioriteetit, testauksen päätöskriteerit, lisätä resursseja tai jopa lykätä julkaisupäivää. (ISTQB 2015, 41.)

4.5 Ruokaviraston ohjeistus laadunhallintaan

Ruokavirastolla on olemassa yleisohjeistus laadunhallintaan. Tietojärjestelmän ja ohjelmistojen laadusta puhuttaessa käytetään ISO/IEC 25010:2011 -standardin mukaista laatumallia (kuvio 2), jossa laatu jaetaan kahdeksaan piirteeseen, jotka taas jakautuvat useisiin alipiirteisiin. Jokainen Ruokaviraston projekti määrittelee itse, mitkä piirteistä ovat relevantteja heidän toimintansa kannalta ja miten. Laatupiirteitä ovat toiminnallinen sopivuus, tehokkuus, yhteensopivuus, käytettävyys, luotettavuus, turvallisuus, ylläpidettävyys ja siirrettävyys. (Turpeinen 2019.)



Kuvio 2: ISO/IEC 25010:2011 laatumalli (Turpeinen 2019.)

Laadunvarmistuksen tekniikoina käytetään testausta, katselmointeja ja käytettävyyden arviointia. Testausta tehdään projektin testaussuunnitelman mukaan ja testitapausten pohjana ovat käyttäjätarinat ja skenaariot. Testaus ei ole erillistä toimintaa vaan sen aikana tehdään yhteistyötä määrittelijöiden ja toteuttajien kanssa, ja sitä tehdään samaan tahtiin muun työn kanssa. Testaukseen liittyvän informaation tulee olla kaikkien projektin osapuolien saatavilla ja nähtävissä, joten testitapaukset ja testihavainnot kirjataan projektin käytössä olevaan tehtävienhallintaohjelmaan, josta ne ovat myös helposti raportoitavissa. (Turpeinen 2019.)

Ruokavirastossa katselmointeja on kolmea tyyppiä: dokumentaation, ohjelmakoodin ja sprintin katselmointeja. Katselmoijan asiantuntemus on katselmoinnin perusta, sillä hän kirjaa virheet ja puutteet sekä esittää parannusehdotuksia ja kysymyksiä oman kokemuksensa pohjalta. Projektin vastuulla on hoitaa sovitut parannukset ja korjaukset. (Turpeinen 2019.)

Käytettävyyden arvioinnin tueksi Ruokavirastossa on olemassa dokumenttipohja, jossa on esitelty mahdollisia arvioinnin kriteerejä ja huomioitavia asioita. Projektille on määritelty laatuvaatimuksia käytettävyyden suhteen, ja arviointi tehdään näiden pohjalta. (Turpeinen 2019.)

5 Tutkimusmenetelmät

Tässä luvussa kuvataan työssä käytettäviä tutkimusmenetelmiä. Opinnäytetyö on kehittämistutkimus, jonka tarkoituksena on tuottaa ”käytännön työelämään käyttökelpoisia ratkaisuja, joiden toimivuus yleensä myös varmistetaan” (Kananen 2013, 15). Kehityskohteen löytäminen ei siis riitä, vaan tutkimus vaatii toimintaa, muutoksen. Tätä varten pitää löytää syyt sekä kehittää keinot muutokseen.

5.1 Kvantitatiivinen ja kvalitatiivinen tutkimus

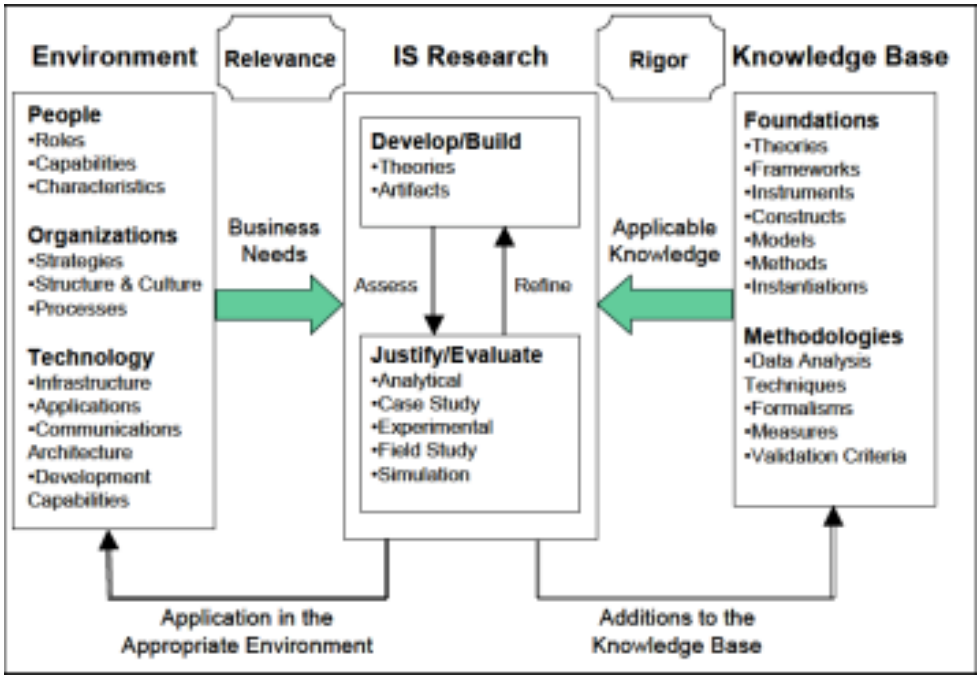
Kvantitatiivinen (määrällinen) ja kvalitatiivinen (laadullinen) tutkimus voidaan erottaa toisistaan tutkimuskäytänteiden tai periaatteellisten kysymysten perusteella. Tutkimussuuntaukset eivät kuitenkaan ole toistensa vastakohtia, vaan niitä voidaan käyttää täydentämään toisiaan. (Hirsjärvi et al. 2007, 132.) Kvalitatiivinen tutkimus pyrkii kuvaamaan kohdettaan todellisen elämän näkökulmasta sekä mahdollisimman kokonaisvaltaisesti, kaikki sen aspektit huomioon ottaen. Tällainen tutkimus pyrkii ennemmin löytämään uusia tutkimusväittämiä kuin todentamaan jo löydettyjä. (Hirsjärvi et al. 2007, 157.) Todentamiseen pyrkii kvantitatiivinen tutkimus, jossa korostuvat aiemmat tutkimukset ja niistä löydetyt johtopäätökset. Kvantitatiivisessa tutkimuksessa keskitytään keräämään määrällistä aineistoa ja testaamaan tutkimushypoteeseja aineiston pohjalta. (Hirsjärvi et al. 2007, 136.)

Tämä opinnäytetyö sisältää kvalitatiivista tutkimusta. Tarkastelussa on yhden projektin toiminta tarkemmalla tasolla, jolloin tutkimusaineisto ei ole laaja eikä tutkittavia tapauksia ole kuin yksi, josta pyritään tekemään yleispäteviä johtopäätöksiä. Tutkittavaan projektiin pyritään perehtymään yksityiskohtaisesti ja kokonaisvaltaisesti.

5.2 Tietojärjestelmien suunnittelutiede

Tietojärjestelmien suunnittelutiede (Design Science Research) on innovatiivisten artefaktien luomista tosimaailman ongelmien ratkaisemiseksi (Hevner 2015). Suunnittelutieteen voidaan sanoa koostuvan rakentamisesta ja arvioinnista. Rakentaminen on prosessi, jossa luodaan tiettyä tarkoitusta varten jokin artefakti (konstruktio, malli, metodi tai toteutus). Arvioinnissa mietitään, miten artefakti palvelee tehtäväänsä. (Järvinen 2006, 26.) Artefakti voi olla uusi ratkaisu uuteen ongelmaan, tuttu ratkaisu uuteen ongelmaan, uusi ratkaisu tuttuun ongelmaan tai tuttu ratkaisu tuttuun ongelmaan (Hevner 2015).

Rakentamiseen ja arviointiin vaikuttavat merkitystä antava ympäristö sekä perusteet antava tietoperusta. Alla olevasta kuvasta (kuvio 3) näkee, mitä ympäristö ja tietopohja tarkoittavat, miten ne vaikuttavat suunnittelutieteeseen ja miten suunnittelutiede vaikuttaa niihin. Tutkimuksella on konteksti, jokin liike-elämän tarve, johon haetaan ratkaisua suunnittelutieteen keinoin. Tietopohja auttaa tutkimuksen teossa tarjoten teorioita, metodeja tai malleja, joiden avulla voidaan rakentaa eli kehittää artefakti. Artefaktia voidaan arvioida tai kehittää eteenpäin, ja tässä työvälineenä ovat taas tiedot ympäristöstä ja tietoperustasta. Kehitetty artefakti vaikuttaa sekä ympäristöönsä ratkaisuna ongelmaan että tietoperustaan lisäämällä tietoa kohteesta. (Hevner 2015.)



Kuvio 3: Suunnittelutieteen määritelmä (Carroll 2018)

5.3 Haastattelu ja kysely

Haastattelu on käyttäytymis- ja yhteiskuntatieteiden käytetyin menetelmä. Sitä voidaan kuvata keskusteluksi, jolla on tarkoitus. Se on kielellistä vuorovaikutusta, jolla kerätään mielipiteitä, tietoja, käsityksiä tai uskomuksia. Haastattelu on joustava menetelmä, joka soveltuu moniin käyttötarkoituksiin ja jolla on mahdollista saada syvällistä tietoa tutkittavasta asiasta. (Hirsjärvi & Hurme 2001, 11.)

Menetelmän joustavuus ilmenee siitä, miten tiedonhankintaa voidaan suunnata tilanteen mukaan. Lisäkysymyksillä voidaan selvittää vastausten takana olevia motiiveja tai tarkentaa epäselväksi jäänyttä lausuntoa. (Hirsjärvi & Hurme 2001, 34.) Informaation kerääminen on haastattelun päämäärä, joten se on ennalta suunniteltua, tavoitteellista toimintaa, joka tapahtuu haastattelijan ehdoilla tai hänen johdollaan. (Hirsjärvi & Hurme 2001, 42.)

Teemahaastattelussa ei käytetä yksityiskohtaisia kysymyksiä vaan keskustelu etenee aihepiirien mukaan. Tällöin haastattelu on puolistrukturoitu, keskustelua ohjaavat teemat on valittu etukäteen, mutta kysymysten tarkan muodon ja järjestyksen puuttuminen vapauttaa haastattelun tutkijan näkökulmasta ja antaa tilaa haastateltavalle. (Hirsjärvi & Hurme 2001, 47-48.) Teemat toimivat haastattelijan muistilistana ja ohjaavat keskustelua (Hirsjärvi & Hurme 2001, 66). Valmiit teemat turvaavat tarvittavan tiedon saannin, mutta toisaalta ne antavat myös liikkumavaraa tilannekohtaisten ratkaisujen tekoon. (Hirsjärvi & Hurme 2001, 102-103.)

Kyselyä käytetään yleensä, kun aineistoa kerätään standardoidusti. Tällä tarkoitetaan sitä, että asiaa kysytään kaikilta vastaajilta samalla tavalla, jotta vastaukset ovat keskenään vertailukelpoisia. Kyselyllä kerätty aineisto voi olla hyvinkin laaja, vastaajia voi olla paljon ja lomakkeella voidaan kysyä useita asioita. Laajan aineiston käsittely on yleensä kvantitatiivista ja sitä varten on kehitetty tilastollisia analysointitapoja ja raportointimuotoja, mikä helpottaa tutkijan työtä. Kyselyihin liittyy myös heikkouksia, sillä ei voida tietää, ottivatko vastaajat kyselyn vakavasti, tuliko kysymysten ja vastausten tulkinnassa väärinkäsityksiä tai millainen on kyselyn kato eli vastaamattomuus. Hyvän lomakkeen luominen myös vaatii aikaa, tietoa ja taitoa tutkijalta. (Hirsjärvi et al. 2007, 188-190.)

Lomakkeissa käytetään yleensä kolmea eri kysymysmuotoa; avoimia kysymyksiä, monivalintakysymyksiä tai asteikkoihin perustuvia kysymyksiä. Avoimissa kysymyksissä on vain kysymys ja vastaukselle on varattu tyhjää tilaa. Tällainen kysymys antaa tilaa vastaajien näkemyksille, joita tutkija ei ole osannut aavistaa. Monivalinnoissa on valmiit vastausvaihtoehdot, joista voidaan antaa valita yksi tai useampi. Vaihtoehdot ovat tutkijan valmiiksi antamia ja hänen kontrolloitavissa. Asteikkoihin perustuvissa kysymyksissä esitetään väitteitä, joihin vastaaja valitsee tutkijan valitsemalta asteikolta vastauksen, esimerkiksi, että onko hän täysin samaa vai täysin eri mieltä väittämän kanssa. (Hirsjärvi et al. 2007, 193-195.)

5.4 Osallistuva havainnointi

Kyselyillä ja haastatteluilla voidaan tutkia, mitä ajatellaan, tunnetaan ja uskotaan. Jotta voidaan tutkia, mitä todella tapahtuu ja toimitaanko kuten sanotaan, tarvitaan tiedonkeruumenetelmäksi havainnointi. Havainnoinnilla saadaan välitöntä ja suoraa tietoa yksilöiden, ryhmien tai organisaation toiminnasta ja käyttäytymisestä. Voidaan sanoa sen olevan todellisen elämän ja maailman tutkimista, mutta toisaalta havainnoija voi häiritä asioiden luonnollista kulkua. Havainnointi vie tutkijalta paljon aikaa, mutta se voi tuottaa mielenkiintoista ja monipuolista aineistoa. (Hirsjärvi et al. 2007, 207-212.) Havainnoinnilla on hyvä hankkia tietoa esimerkiksi työtehtävistä tai niiden suorittamiseen liittyvistä prosesseista. Toiminnan kuvaileminen voidaan kokea hankalaksi, ja havainnoinnilla tunnistetaan myös niin kutsuttu hiljainen tieto, jota ei ehkä muuten jaeta. (Kananen 2013, 88-89.)

Havainnointimenetelmät voidaan jakaa kahteen lajiin, systemaattiseen ja osallistuvaan havainnointiin. Systemaattisessa havainnoinnissa havainnoija on ulkopuolinen ja havainnointi tapahtuu tarkasti rajatuissa tiloissa. Havainnot tehdään ja tallennetaan tarkasti ja systemaattisesti erilaisten apukeinojen avulla, kuten tarkistuslistojen tai arviointiskaalojen avulla. Osallistuvassa havainnoinnissa havainnoija osallistuu tutkinnan kohteiden toimintaan näiden ehdoilla. Havainnoijan osallistumisaste vaihtelee, hän voi yrittää olla tutkittavan ryhmän täydellinen jäsen tai tehdä tutkittaville selväksi olevansa tekemässä havaintoja ryhmässä. (Hirsjärvi et al. 2007, 207-212.)

5.5 Reliabiliteetti ja validiteetti

Reliabiliteetti tarkoittaa sitä, että tutkimuksen mittaustulokset ovat toistettavissa, jolloin tutkimuksen tulokset eivät ole sattumanvaraisia. Kyseessä on siis tutkimuksen luotettavuus. On olemassa useita tapoja todeta tutkimuksen reliaabelius. Tutkimus on reliaabeli, jos kaksi arvioijaa saa saman tuloksen samasta aineistosta tai jos samalla tutkimustavalla saadaan kahdella kertaa samat tulokset. (Hirsjärvi et al. 2007, 226.) Tutkimusprosessin dokumentoinnin lisäksi on siis tärkeää kirjata myös syyt ja perustelut tehdyille valinnoille. Tämä auttaa koko työn oikeellisuuden arvioinnissa. Niin tutkimusprosessin vaiheet, lähtö- ja lopputilanteen kuin menetelmien kuvaaminen ja kirjaaminen on tärkeää dokumentaatiota reliabiliteetin kannalta. (Kananen 2013, 116-117.) Tämän opinnäytetyön reliabiliteetti varmistetaan sillä, että tutkimuksen kulku kuvataan riittävän hyvin, jotta tutkimus on toistettavissa. Selostus auttaa myös lukijaa arvioimaan, onko tulosten tulkinta oikeanlaisella pohjalla.

Tutkimuksen validiteetti tarkoittaa sitä, että tutkittava asia on juuri se, mitä halutaan tutkia. Voidaan siis sanoa, että kyseessä on tutkimuksen pätevyys. Eli että tutkimusmenetelmät tuottavat informaatiota siitä asiasta, mitä oli tarkoitus tutkia. (Hirsjärvi et al. 2007, 226.) Tämä tarkoittaa sitä, että haastattelukysymykset ja lomake tulee laatia mahdollisimman hyvin ja pitää kysymykset yksinkertaisina, jottei väärinymmärryksiä tapahdu, mikä aiheuttaisi vääristymän datassa. Pätevyyttä parantaa myös useamman menetelmän käyttäminen. Tutkimuksen luotettavuutta arvioitaessa kannattaa muistaa, että pystyy perustelemaan, todistamaan ja näyttämään toteen sen, minkä esittää. Myös vaihtoehtoja pitää muistaa punnita ja tuoda esille oma näkemys, sillä se mitä opinnäytetyössä ei mainita, ei ole sitä arvioidessa olemassa. (Kananen 2013, 122.)

6 Xray-testauslaajennus

Xray on JIRA-tehtävienhallintaohjelmiston laajennus. JIRA on projektinhallinnan työkalu, jota käytetään tehtävien ja bugien kirjaamiseen ja seurantaan sekä projektin kokonaisuuden hallintaan. JIRA tukee ketterää ohjelmistokehitystä, sen avulla hallinnoidaan sprintin kulkua ja kehitysjonoa. Ruokavirastossa sovelluskehityksessä käytetty JIRA-projekti siirtyy järjestelmän muutoksenhallinnan työkaluksi, kun järjestelmä viedään tuotantokäyttöön ja se siirtyy kehityksestä ylläpidettäväksi. Tällöin JIRA:a käytetään muutosten ja virhekorjausten hallintaan, seurantaan ja raportointiin. (Kormano 2019.)

Xray-testauslaajennuksen on kehittänyt Xpand IT ja se on julkaistu vuonna 2014. Xray on tarkoitettu testauksen hallintaan, ja sitä voidaan käyttää koko testauksen elinkaaren ajan, sillä se sisältää työkalut niin testauksen suunnitteluun kuin testien luomiseen, suorittamiseen ja raportointiin. Xray tuo JIRA:an uusia tehtävätyyppejä, joiden avulla manuaalista ja

automaattista testausta suoritetaan. Xray on integroitu JIRA:an saumattomasti, joten sen tuomia tehtävätyyppejä voidaan käsitellä samalla tavalla kuin JIRA:n omia. (Freire 2019a.)

Xray:n voi myös yhdistää jatkuvan integraation työkaluihin kuten Jenkins, ja automaattisia testejä sekä niiden tuloksia voidaan tuoda sen kautta. Xray:lla voidaan myös muodostaa raportteja, joilla seurataan testauksen laatua ja kattavuutta. (Freire 2019a.)

6.1 Tehtävätyypit

Xray:n mukana JIRA:an tulee viisi uutta tehtävätyyppiä (issue), joiden nimet ovat Test, Pre-Condition, Test Set, Test Plan ja Test Execution (Freire 2019b). Alla on nämä esitelty tarkemmin ja jokaisen edessä on kuvake, josta tyypin voi ohjelmassa nopeasti tunnistaa. Tehtävätyypeille on opinnäytetyössä kehitetty suomenkielisiä käännöksiä, mutta projektin keskusteluissa niihin viitataan enemmänkin englanninkielisillä nimityksillä, joten opinnäytetyössä on usein mainittu tehtävätyypin alkuperäinen nimitys suluissa. Samoin kuin issue on käännetty tehtäväksi opinnäytetyössä, vaikkei se ole vakiintunut nimitys Ruokavirastossa.

Test tarkoittaa testitapausta, joka on joko manuaalinen (manual) tai automaattinen (generic). Manuaalisen testin suorittajana on käyttäjä, kun taas automaattisen testin suorittajana ja tulosten arvioijana on ulkopuolinen työkalu. (Moreira 2019a.) Test-tehtävässä on testitapauksen kuvaus, suorittamisen vaiheet (stepit), niiden syötteet ja toiminnot sekä odotettu lopputulos. (Freire 2019b.)

Pre-Condition on esiehto, joka pitää täyttää, ennen kuin varsinaisen testin vaiheisiin edetään. Samaa esiehtoa voidaan käyttää useammassa testissä, jolloin vältytään kirjoittamasta samoja vaiheita yhä uudelleen. (Freire 2019b.) Tämän pitää olla samaa tyyppiä kuin testi, johon se linkitetään, eli manuaalinen tai automaattinen. (Moreira 2019b.)

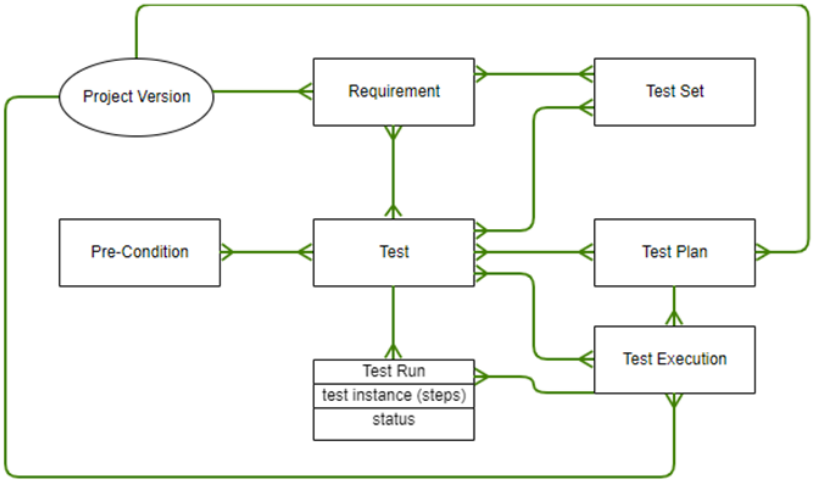
Test Set eli testijoukko koostuu useasta testitapauksesta, jotka on ryhmitelty jollain loogisella tavalla. Sama testi voi kuulua useampaan testijoukkoon. (Freire 2019b.) Testijoukon avulla testit voidaan ryhmitellä vaikka tietyn testattavan komponentin tai aspektin (esimerkiksi tietoturva) mukaan. (Moreira 2019c.) Test Set -tehtävän sijaan testit voi järjestää myös Test Repositoryssa hierarkkiseen järjestykseen (G 2017a).

Test Execution on testin suoritussuunnitelma, joka voidaan määrätä jonkun käyttäjän toteutettavaksi (Freire 2019b). Test Execution -tehtävällä ryhmitellään käyttäjän määrittelemä kokoelma testejä. Overall Execution Status on testin suoritussuunnitelmassa oleva värillinen edistymispalkki, josta näkee yhdellä silmäyksellä läpi menneet, epäonnistuneet ja vielä suorittamatta olevat siihen liittyvät testit. Tähän tehtävään määritellään, millä versiolla testit suoritetaan, milloin testaus aloitetaan ja lopetetaan sekä missä ympäristössä testit

suoritetaan. (Braz 2019.) Testejä voidaan suorittaa ad hoc -tyyppisesti, jolloin testien suoritussuunnitelma luodaan suoraan testistä ja se suoritetaan heti. (G 2017b.)

Test Plan on testisuunnitelma, joka sisältää ne testit ja testien suoritussuunnitelmat, jotka on suunniteltu toteutettavaksi tietyn ajan sisällä. (Freire 2019b.) Tämän tehtävän avulla voidaan visualisoida ja hallinnoida testisuunnitelmaan liitettyjä testejä ja testien suoritussuunnitelmia. Tehtävässä on Overall Execution Status-palkki, josta nähdään testien statusten visuaalinen yhteenveto siitä, kuinka monta testiä on mennyt läpi, kuinka monta epäonnistunut ja kuinka monta on tekemättä. Testisuunnitelmassa testit voidaan järjestää haluttuun järjestykseen, ja siitä on helppoa luoda uusia testien suoritussuunnitelmia, esimerkiksi testeistä, jotka eivät ole menneet hyväksytysti läpi. (Moreira 2019d.)

Alla on kaavio Xray:n tehtävien (issue) riippuvuuksista (kuvio 4). Termi requirement tarkoittaa vaatimustyyppistä tehtävää. Testi liittyy vaatimukseen ja voi kuulua useampaan testijoukkoon, testisuunnitelmaan tai testien suoritussuunnitelmaan, sekä sillä voi olla useampi esiehto ja testiajo. Testien suoritussuunnitelma sen sijaan liittyy yleensä vain yhteen testisuunnitelmaan, mutta testisuunnitelmalla voi olla useampi testien suoritussuunnitelma.



Kuvio 4: Entity Relationship Diagram (Freire 2019b)

Test run tarkoittaa testiajoa eli kun testi suoritetaan, siitä muodostuu test run, jolla on oma status. Testin tila kertoo, menikö testi läpi (PASS) vai epäonnistuiko (FAIL), odottaako testi suorittamista (TODO) vai ollaanko sitä suorittamassa (EXECUTING) tai onko testi keskeytetty (ABORTED). Kaikki projektiin liittyvät testit löytyvät Xray:n Test Repositorysta. (Freire 2019b.)

6.2 Vaatimuksen kattavuusstatus

JIRA-projekti pitää määritellä asetuksista Requirement-projektiksi, jotta tietyt JIRA:n tehtävätyypit määritellään vaatimuksiksi Xray:n näkökulmasta. Tällöin ne saavat kustomoidun kentän nimeltä Requirement Status (vaatimuksen kattavuusstatus), josta voidaan nähdä testikattavuus ja jonka perusteella voidaan muodostaa raportteja. (G 2017c.) Status kertoo sen, onko vaatimukseen liitetty testejä sekä miten testit ovat onnistuneet. Siihen vaikuttavat liittyvien testien tulosten lisäksi vaatimuksen alitehtävien kattavuusstatus eli alitehtäviin liitetyt testit. (Freire 2019c.)

Kuviossa viisi näkyy, miten testien tilat vaikuttavat vaatimuksen kattavuusstatukseen (Requirement Status). Status voi olla OK, NOK, NOTRUN, UNKNOWN tai UNCOVERED. Vihreä OK tarkoittaa sitä, että vaatimus on onnistuneesti katettu: siihen on liitetty testejä ja kaikki siihen liittyvät testit ovat PASS-tilassa (eli menneet onnistuneesti läpi). Punainen NOK eli ’not ok’ tarkoittaa sitä, että vaatimuksen kattavuus ei ole kunnossa: ainakin yksi siihen liitetty testi on FAILED-tilassa (testi ei ole mennyt läpi). Keltainen NOTRUN tarkoittaa sitä, että vaatimusta ei ole kokonaan katettu: ainakin yksi siihen liitetty testi on tilassa TODO tai ABORTED (suorittamatta tai keskeytetty), mutta että yksikään testi ei ole FAILED-tilassa. Sininen UNCOVERED-status tarkoittaa sitä, että vaatimukseen ei ole liitetty yhtään testiä. Viimeinen status on harmaa UNKNOWN, jolloin ainakin yksi vaatimukseen liitetyistä testeistä on UNKNOWN-tilassa, mutta yksikään ei ole FAILED-tilassa. (Freire 2019c.) Tämä on harvinainen poikkeustilanne, jota ei ole kuvattu kuviossa viisi, jossa olevat statukset on otettu projektin JIRA:sta. Kuviossa näkyvä ELTE MVP on version nimi.

Requirement Status	Testien tilat
ELTE MVP - OK	Kaikki vaatimukseen linkitetyt testit ovat PASS
ELTE MVP - NOK	Ainakin yksi vaatimukseen linkitetty testi on FAIL
ELTE MVP - NOTRUN	Ainakin yksi vaatimukseen linkitetty testi on TODO/ABORTED/EXECUTING
ELTE MVP - UNCOVERED	Vaatimukseen ei ole linkitetty yhtään testiä

Kuvio 5: Requirement statuksen (vaatimuksen kattavuusstatus) ja testien tilojen yhteys

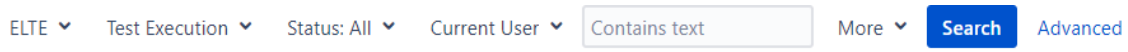
7 Xray:n raportointimahdollisuudet

Opinnäytetyössä on tarkoituksena kehittää raportointitapoja, joiden käyttö perustaa Xray-laajennuksen käyttöön. Tämän takia laajennuksen mahdollisuudet siinä suoritetun testauksen raportointiin käydään läpi. Ennen kuin raportoinnista voidaan puhua, tulee miettiä sitä, miten ohjelmistossa olevasta datasta saadaan vain relevantti aineisto esiin. Aineiston rajaamisessa ohjelmassa käytetään hakua ja näitä hakuja voi tallentaa hakusuodattimiksi. Haettua aineistoa voidaan raportointimielessä tarkastella pelkästään hakutuloksina tai tulokset voidaan näyttää ohjausnäkymällä eri vempaimilla tai viedä jopa toiseen ohjelmaan. Xray myös tarjoaa omia raportteja, joilla testauksen tilaa ja vaatimusten kattavuutta voidaan tarkastella.

7.1 Hakusuodattimet

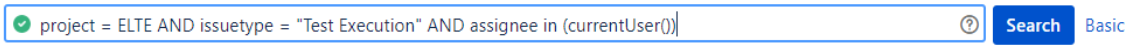
Jotta Xray:n kautta tehtävä raportointi näyttää todellisen tilanteen ja kuvaa oikeaa asiaa, pitää raportoinnissa käytettävä aineisto rajata huolellisesti. Tätä varten tarvitaan tallennettuja hakusuodattimia (filters). Hakusuodattimia kannattaa luoda ja tallentaa projektikohtaisesti projektin käytänteiden mukaisesti. Ne käyvät monenlaiseen raportointiin suoraan sellaisenaan käytettäväksi, tai sitten niistä voidaan kopioida kyselylauseke. Hakusuodattimia voidaan luoda perushaussa ehtojen avulla, mutta jos halutaan tehdä monimutkaisempia hakuja, joudutaan käyttämään JIRA Query Language:a (JQL). JQL on JIRA:n oma kyselykieli, jolla voidaan muodostaa monimutkaisiakin kyselylausekkeita. Xray tuo tähän kyselykieleen vielä omia, laajennuskohtaisia funktioita.

Perushaussa on muutama hakuehto, joihin voi valita arvon, ja hakea ehtoja vastaavia tehtäviä. Esimerkkihaku näkyy kuviossa 6, projektiksi on valittu ELTE, tehtävätyypiksi testien suoritussuunnitelma ja kirjautunut käyttäjä tehtävän suorittajaksi (assignee).



Kuvio 6: JIRA:n perushaku

Yksinkertainen JQL-kysely koostuu kentästä (field), operaattorista ja yhdestä tai useammasta arvosta tai funktiosta. Kenttä tarkoittaa JIRA:ssa olevia kenttiä, kuten status tai tehtävätyyppi tai labeli. Kuviossa 7 on sama haku kuin kuviossa 6, mutta nyt JQL-lausekkeena. JIRA auttaa JQL-lauseen muodostamisessa ehdottaen vaihtoehtoja samalla, kun hakua kirjoittaa. JIRA:n ja Xray:n dokumentaatiosta löytyy ohjeita kentistä ja erilaisista funktioista, joilla saa tarkennettua hakua.



Kuvio 7: JQL-lause

7.2 Ohjausnäkymän vempaimet

JIRA:ssa voi luoda ohjausnäkymän (dashboard), jolle voi kerätä haluamaansa tietoa. Tieto tuodaan ohjausnäkymälle vempainten (gadgets) avulla. Ohjausnäkymiä voi olla useampia ja niitä voi jakaa muille. Ohjausnäkymä on hyödyllinen apuväline, koska sen avulla pystyy seuraamaan monia itseä kiinnostavia tai omaan työskentelyyn vaikuttavia asioita.

JIRA tarjoaa monia hyödyllisiä vempaimia, mutta kaikki niistä eivät sovellu testauksen raportointiin. Siksi tässä esitellään vain niitä, jotka on arvioitu raportoinnin kannalta oleellisiksi. Esittelyt pidetään lyhyinä ja käytettäväksi valittuja vempaimia esitellään tarkemmin raportoinnin esittelyn yhteydessä. Käyttökelpoiset vempaimet voidaan ryhmitellä neljään ryhmään, tehtävien listaukseen tai niiden välisten suhteiden visualisointiin käytettäviin, tehtävien iästä ja elinkaaresta kertoviin, testeihin liittyviin sekä Xray:n raportteihin.

Hakusuodattimen mukaisen aineiston visualisointiin ja listaukseen käytettäviä vempaimia on kolme. **Filter Results** näyttää hakusuodattimen tulokset taulukkona, jossa näytettäviä sarakkeita voidaan määritellä asetuksista. **Pie Chart** on piirakkakuvio, jolla voi visualisoida tallennetun hakusuodattimen mukaista aineistoa valitun tiedon mukaan ryhmiteltynä. **Two Dimensional Filter Statistics** -vempaimella voi tarkastella aineistoa taulukossa kahden tiedon mukaan järjestettynä.

Tehtävien elinkaareen ja ikään liittyviä vempaimia on neljä. Niissä määritellään tarkasteltavan ajanjakson pituus ja ryhmittely. **Average Age Chart** näyttää tehtävän iän eli kuinka monta päivää ratkaisematon tehtävä on ollut avoimena. **Created vs. Resolved** -vempaimen avulla voi tarkastella tiettynä ajanjaksona luotuja ja ratkaistuja tehtäviä. **Recently Created Chart** -vempaimessa voi näyttää tietyn ajanjakson sisällä luodut tehtävät, vempaimen palkki näyttää myös, kuinka monta noista luoduista on ratkaistu tähän mennessä. **Resolution Time** kertoo, kuinka pitkään tiettynä ajanjaksona ratkaistut tehtävät olivat avoinna.

Testeihin liittyviä vempaimia on viisi. **Tests list** -vempain listaa kaikki hakusuodattimen mukaiset testit ja niiden statukset. **Test Runs List** näyttää haettuihin testien suoritussuunnitelmiin (test executions) liittyvät testien suoritukset (test runs) ja niiden tiedot. **Test Runs Summary** näyttää testien suoritusten (test runs) statukset taulukkona tai pylväskaaviona esimerkiksi testeittäin ryhmiteltynä. **Test Evolution** -vempaimella voi nähdä testien kehityksen tiettyjen testien suoritussuunnitelmien aikana. **Overall Test Results** -vempaimella voi tarkastella testejä ja niiden läpimenoa joko version tai tietyn testisuunnitelman puitteissa.

Kaksi Xray:n tarjoamista valmiista raporteista voidaan tuoda vempaimena ohjausnäkymälle, sellaisia ovat **Historical Daily Requirement Coverage** ja **Overall Requirement Coverage**. Molemmat raportit esitellään tarkemmin luvussa 7.3.1.

7.3 Valmiit Xray:n raportit

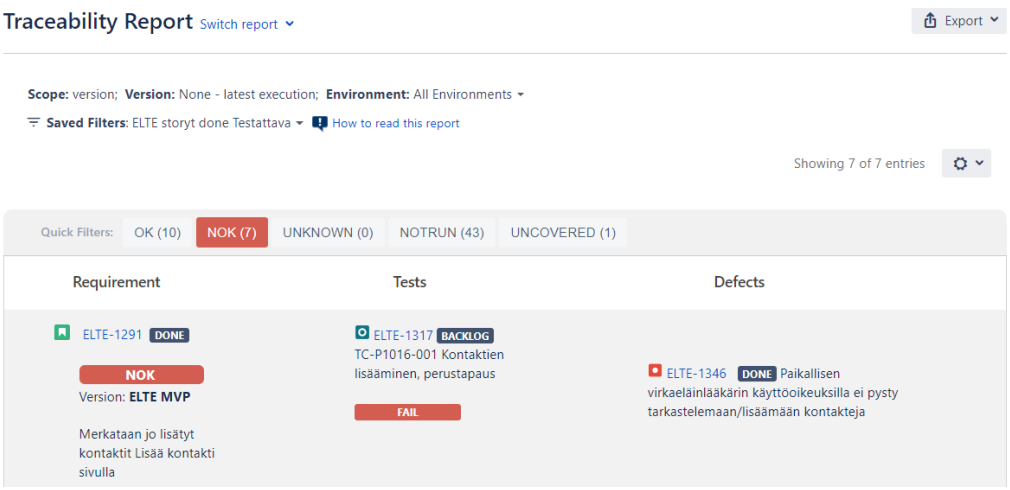
Xray-laajennus tarjoaa erilaisia testaukseen liittyviä raportteja, joita voidaan generoida tehtävien pohjalta. Raportteja on yhteensä kuusi, kolme niistä liittyy vaatimusten kattavuusstatuksen (requirement status) tarkasteluun ja kolme testien suorittamiseen.

Raportteja voidaan rajata niin, että tarkasteltava aineisto on halutunlaista. Analyysin laajuudeksi voidaan valita jokin versio tai testaussuunnitelma, ja tätä valintaa voidaan vielä tarkentaa suodattamalla tulokset halutun ympäristön mukaan. (Duarte 2019a.) Eli raportin saa koostettua esimerkiksi tietystä ohjelmaversiosta tai tietyn testisuunnitelman mukaan ja katsottua vain tiettyyn testiympäristöön liittyvät tulokset. Raporteissa käytettyä aineistoa voidaan myös suodattaa muutaman valmiiksi ehdotetun parametrin tai jos nämä valinnat eivät riitä rajaamaan haluttua dataa, voidaan materiaalin rajaukseen käyttää aiemmin luotua, tallennettua hakusuodatinta. (Duarte 2019a.)

Raportteja voidaan ladata CSV-tiedostona tai niistä voidaan jakaa linkki, jonka avulla joku toinen voi generoida raportin. Sivun URL-osoitteessa kerrotaan raportin määrittelyt ja linkkiä avatessa ne otetaan raportin rajauksiksi (Duarte 2019b). Jotta raportin saa auki, pitää vastaanottajalla olla oikeudet tarkastella kyseisen projektin tehtäviä.

7.3.1 Vaatimusten kattavuuteen liittyvät raportit

Vaatimuskattavuutta voi tarkastella tämänhetkisen tilanteen tai päivittäisten tietojen mukaan tai vaatimuksen kattavuutta voi jäljittää aina testisuoritukseen ja bugiin saakka. Jäljitettävyysraportista (**Traceability Report**, kuvio 8) voidaan nähdä yhteys vaatimuksen ja siihen liitettyjen testien, testiajojen ja vikaraporttien (defect) välillä. Sen avulla voidaan jäljittää ja analysoida, mikä testi tai bugi vaikuttaa vaatimuksen kattavuusstatukseen (requirement status). (Duarte 2019a.)



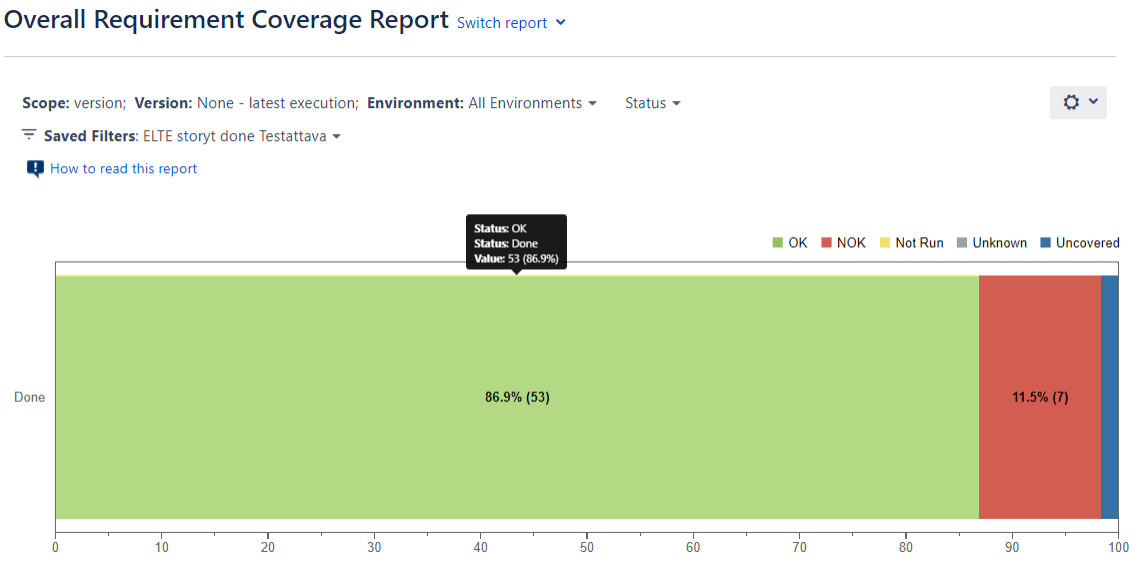
Kuvio 8: Jäljitettävyysraportti

Raportissa näkyy vaakarivillä vaatimus, siihen liittyvät testit, testien suoritukset (test runs) ja viimeisenä testiin liitetyt vikaraportit. Testien suoritukset voidaan tarvittaessa piilottaa, sillä niitä voi olla häiritsevän paljon ja raportin tulkinta vaikeutuu. (Duarte 2019a.)

Raporttia voi suodattaa vaatimuksen kattavuusstatuksen mukaan (OK, NOK, UNKNOWN, NOT-RUN ja UNCOVERED). Näin voidaan nopeasti suodattaa näkyviin esimerkiksi vain ne vaatimukset, joihin liittyy epäonnistuneita testejä (kuten kuviossa 8) tai vaatimuksia, joihin ei vielä ole liitetty yhtään testiä. (Duarte 2019a.) Raportissa näkyy NOTRUN-tilassa nekin vaatimukset, jotka oikeastaan ovat OK-tilassa, mutta niihin liittyviä testejä on jossain toisessa ympäristössä suorittamatta. Vaatimuksella pitää olla jokaisessa ympäristössä läpimennyt testi, jotta se on OK tämän raportin mukaan, mikä on aiheuttanut projektissa hämmennystä.

Vaatimuskattavuusraportin (**Overall Requirement Coverage Report**) avulla voidaan visuaalisesti tarkastella vaatimusten kattavuutta projektissa. Raportissa näkyy vaatimustyyppisten tehtävien tila niiden vaatimuksen kattavuusstatuksen (requirement status) mukaan. Raportista voidaan tarkastella, kuinka suuri osa projektin vaatimuksista on katettu testeillä, kuinka suuresta osasta testit ovat menneet läpi tai epäonnistuneet ja kuinka suuri osa vaatimuksista on vielä kattamatta kokonaan. (Duarte 2019b.) Jotta raportti antaa tarpeellista tietoa, tulee vaatimukset rajata vain niihin, joita projektissa pyritään kattamaan testeillä.

Vaatimuskattavuusraportissa tehtävät voidaan ryhmitellä visuaalisesti usean vaihtoehdon mukaan. Ohjelmassa valmiina olevia ovat prioriteetti, status, päätös ja komponentti. Ryhmittelyn voi tehdä myös erilaisten kustomoitujen kenttien kautta. (Duarte 2019b.) ELTE-projektissa ei kuitenkaan ole käytössä kustomoituja kenttiä ja tarkasteltavan aineiston voi ryhmitellä mielekkäästi vain statuksen mukaan (kuvio 9).

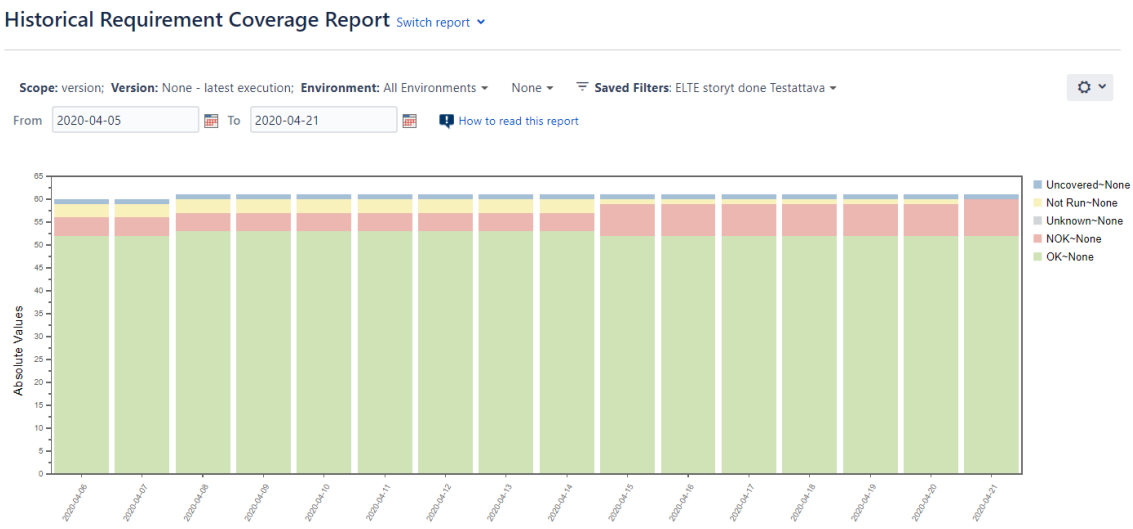


Kuvio 9: Vaatimuskattavuusraportti

Raporttia on mahdollista tarkastella tarkemmin klikkaamalla haluamaansa palkkia. Raportti listaa kaikki palkkiin liittyvät tehtävät (kuviossa 10 on listattuna kuvion 9 sinisen palkin tehtävät), niihin liitettyjen testien määrät kuten myös läpimenneet, epäonnistuneet ja suorittamattomat testit, sekä vaatimuksen valmiusasteen (completeness) prosentteina. Raportissa olevat vaatimukset ja testit ovat linkkejä, joten niitäkin pääsee tarkastelemaan lähemmin. (Duarte 2019b.)

Kuvio 10: Vaatimuskattavuusraportin lisätiedot

Historiallinen vaatimuskattavuusraportti (**Historical Requirement Coverage Report**, kuvio 11) näyttää päiväkohtaiset historiatiedot vaatimusten kattavuuden kehityksestä. Sen avulla voidaan siis analysoida kattavuusstatuksen muutosta päivätasolla. (Duarte 2019c.)



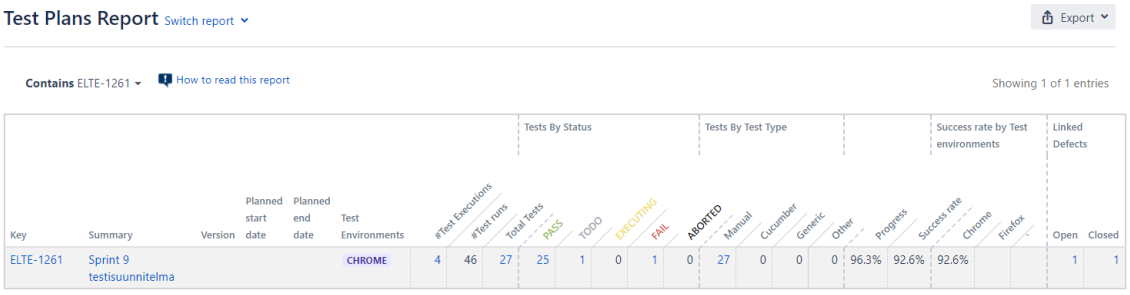
Kuvio 11: Historiallinen vaatimuskattavuusraportti

Aineistoa voi suodattaa samalla tavalla kuin muissakin raporteissa, ja lisäksi tässä raportissa on sama ryhmittelytoiminto kuin vaatimuskattavuusraportissa. Uutena asiana tässä raportissa on datan aikavälin valinta, eli tässä kattavuusstatusta tarkastellaan tietyllä aikajanalla, jossa näkyy päivittäiset testit. Raportti koostuu kahdesta allekkaisesta pylväskaaviosta, joista ylemmässä ovat absoluuttiset arvot ja alemmassa normalisoidut arvot. Raportti on interaktiivinen,

eli palkista saa lisätietoja viemällä hiiren osoittimen sen päälle, ja selitteestä (legend) saa de-aktivoitua osia raportista. (Duarte 2019c.)

7.3.2 Testien suorittamiseen liittyvät raportit

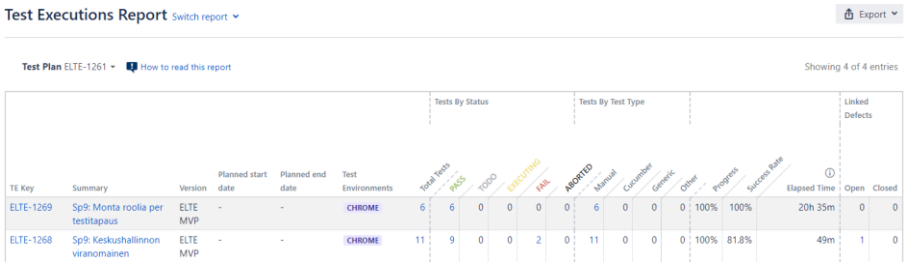
Testisuunnitelmaraportissa (**Test Plans Report**, kuvio 12) näytetään lista hakuehtojen mukaan haetuista testisuunnitelmista ja niihin liittyvistä tiedoista. Raportissa on suunnitelmiin sisältyvien testien suoritussuunnitelmien (test execution) ja testisuoritusten (test run) määrät, testien tilat (PASS, TODO, EXECUTING, FAIL, ABORTED), testien tyyppien (manuaalinen vai automaattinen) määrät, edistymis- ja onnistumisprosentit sekä suunnitelmaan linkitetyt bugit ja niiden statukset. Edistymisprosentti (progress) kertoo niiden testien määrän, jotka ovat lopullisessa tilassa eli joko PASS tai FAIL, ja onnistumisprosentti (success) niiden määrän, jotka ovat onnistuneet (tilassa PASS). (Duarte 2019d.)



Kuvio 12: Testisuunnitelmaraportti

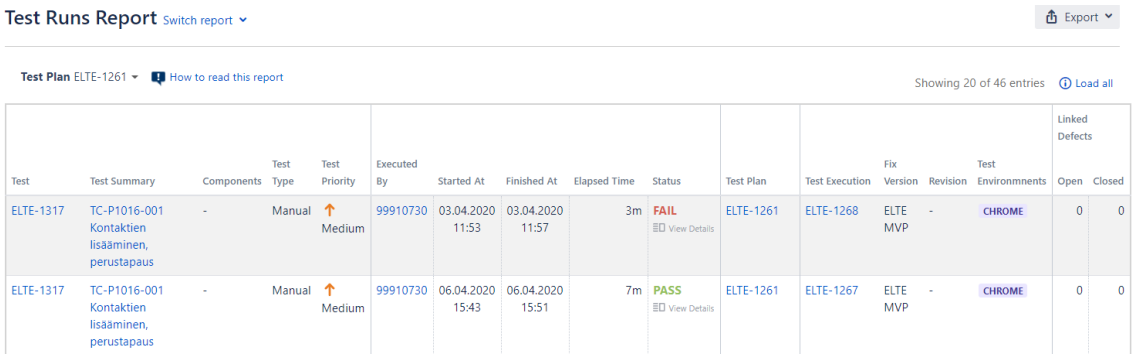
Tämän raportin avulla voidaan arvioida testisuunnitelmaa suhteessa itseensä sekä muihin testisuunnitelmiin. Jotkut raportin arvoista ovat linkkejä ja niiden kautta pääsee tarkastelemaan tarkempia tietoja. (Duarte 2019d.)

Testien suoritussuunnitelmaraportti (**Test Executions Report**, kuvio 13) on samantyylinen kuin testisuunnitelmaraportti. Raportissa on testien määrät, niiden statukset ja tyypit, edistymis- ja onnistumisprosentit sekä linkitetyt vikaraportit. Lisäksi testien suoritussuunnitelman testien suorittamiseen kulunut aika ja testiympäristö. Tätä raporttia on mahdollista suodattaa samalla tavalla, ja testejä, vikaraportteja ja testien suoritussuunnitelmia pääsee tarkastelemaan linkeistä. (Duarte 2019e.)



Kuvio 13: Testien suoritussuunnitelmaraportti

Testien suoritusraportti (Test Runs report, kuvio 14) kertoo testien suorituksen aikaiset tiedot, eli mikä testi oli kyseessä, kuka sen suoritti, milloin ja kuinka kauan siihen meni, mikä oli testin tulos, testausympäristö ja liitettiinkö siihen jokin vikaraportti. (Duarte 2019f.)



Kuvio 14: Testien suoritusraportti

Tämän raportin hyödyt tulevat esiin, kun tuloksia suodatetaan etsityn tiedon mukaisesti. Hakusuodattimia voivat olla esimerkiksi testiympäristö, testisuunnitelma tai testisuoritusten tila. Raportin avulla on hyvä tarkastella esimerkiksi tietyn testisuunnitelman epäonnistuneita testejä. (Duarte 2019f.)

7.4 Confluence ja Xray integraatio

Confluence on tarkoitettu tiimeille tiedon tuottamiseen yhteistyöllä sekä sen jakamiseen. Sisältöä voidaan tuottaa työtilojen, sivujen ja blogien muodossa, ja käyttäjät voivat tuottaa sisältöä yhdessä tai kommentoida toisten tuottamaa tekstiä. (Atlassian 2020.) Kaikki työtilaan käyttöoikeudet saaneet pääsevät luomaan sivuja ja muokkaamaan niitä. Sivuille pääsee tuottamaan sisältöä WYSIWYG-editorin avulla sekä liittämään erilaisia makroja integroitavissa olevista eri järjestelmistä. Tällainen ohjelmisto mahdollistaa yhteisöllisen tiedon tuottamisen ja käsittelyn. Se sopii hyvin opiskeluun, opetukseen, tutkimukseen, hallinnointiin, tuotekehitykseen ja viestintään. (Wikipedia 2019.)

Ruokavirastossa kaikki projektiin liittyvä dokumentaatio löytyy Confluence-pohjaisesta Prikasta. Työtilaan tuotetaan projektinhallinnan dokumentaatiot, kuten esimerkiksi

käyttötarinat, määrittelyt ja projektisuunnitelma. Projektin työtila muuttuu järjestelmän käyttöönoton myötä ylläpidon työtilaksi. (Kormano 2019.)

Confluence ja JIRA ovat saman yrityksen tuotteita, joten ne on suunniteltu täydentämään toisiaan ja ne ovat integroitavissa toisiinsa. Tämän takia JIRA:n issueita voi lisätä Confluence-sivulle tai Confluence-sivusta voidaan luoda JIRA:an uusi tehtävä. (Atlassian 2020.) Confluencessa on myös mahdollista käyttää erilaisia makroja. Ruokavirastolla on käytössä neljä erilaista JIRA:an liittyvää makroa. Makroissa on käännetty JIRA:n termit suomenkielelle ja välillä hieman epäonnistuneesti, sillä esimerkiksi tehtävää (issue) nimitetään ongelmaksi.

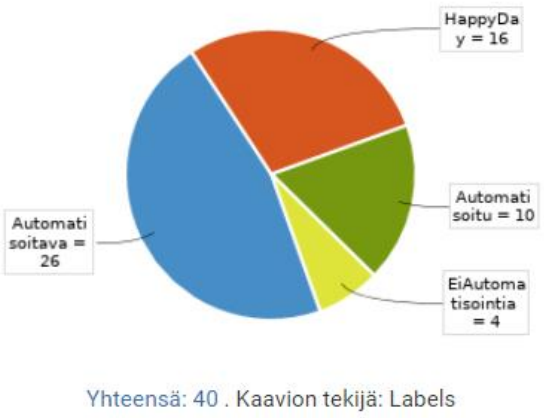
Confluence-sivulle voi tuoda yhden JIRA-issuen tai hakuehtojen mukaisen listan tehtävistä. Tämä toiminto löytyy makroista nimellä **Lisää Jira-ongelma/suodatin**. Hakuehdoksi voi laittaa vaikka vain yhden tehtävän avaimen tai pitkän JQL-lauseen, ja tulokset näytetään sivulla taulukkona, johon voi valita sarakkeet tarpeiden mukaan. Kuviossa 15 on JQL-lauseen mukaan haettuna kaikki vikaraportit.

Avain	Yhteenveto	T	Luotu	Päivitetty	Tila	Resoluutio	Ratkaistu
ELTE-1475	Pitopaikan yhteyshenkilöstä ei näytetä nimeä		toukokuuta 07, 2020	toukokuuta 07, 2020	BACKLOG	Unresolved	
ELTE-1452	Työjono ja kartalla olevien pitopaikkojen näkyydet eivät päivity roolin muutoksen myötä		huhtikuuta 30, 2020	huhtikuuta 30, 2020	DONE	Done	huhtikuuta 30, 2020
ELTE-1451	Yksikön poista hallintopäätöksellä ei onnistu		huhtikuuta 30, 2020	toukokuuta 06, 2020	BACKLOG	Unresolved	
ELTE-1430	Yksiköiden ulkoasu hallintopäätöksellä vaihtelee		huhtikuuta 27, 2020	huhtikuuta 27, 2020	BACKLOG	Unresolved	
ELTE-1414	Kontaktien lisäys: ongelmia karttaa siirrettäessä		huhtikuuta 21, 2020	huhtikuuta 27, 2020	BACKLOG	Unresolved	

Näytetään 5 / 39 ongelmaa Päivitä

Kuvio 15: Issuelistaus

Ympyräkaavioon määritellään jokin aineisto, esimerkiksi JQL-lausekkeella ja kaavion tekijäksi valitaan tieto, jonka mukaan ympyrän jako halutaan nähdä. Kuviossa 16 on aineistona kaikki valmiit testitapaukset ja ne on jaoteltu labelin mukaan.



Kuvio 16: Piirakkakuvio

JIRA:ssa Created vs. Resolved -nimellä oleva ohjausnäkymän vempain voidaan tuoda Confluence-sivulle makrolla, jonka nimi on **Luotu tai Ratkaistu** (kuvio 17). JIRA:n vempaimeen verrattuna tämä sivulle tuotava makro antaa vähemmän informaatiota, sillä siitä ei saa mitään tietoja hiiren osoittimen avulla, kuten vempaimesta. Kaavio tulisi selittää sivulla sekä kertoa, mistä aineistosta se on tuotettu.

Kuvio 17: Luotu tai Ratkaistu -kaavio

Confluence-sivulle voi myös liittää **Kaksiulotteinen**-nimisen makron, jossa aineiston voi näyttää kahden tiedon mukaan jaoteltuna taulukossa. Kuviossa 18 on esitetty kaikki testitapaukset jaoteltuna manuaalisiin ja automaatiotesteihin sekä niiden testien suoritusstatuksen mukaan.

	TestRunStatus			
Test Type	FAIL	PASS	TODO	T:
Generic	0	13	0	13
Manual	3	36	1	40
Total Unique Issues:	3	49	1	53

Näytetään 2 / 2 tilastoa.
Näytä Jirassa

Kuvio 18: Kaksiulotteinen tilasto

8 Testaus ELTE-projektissa

Tässä luvussa kuvataan ELTE-projektissa sovitut käytänteet testauksen ja Xray:n käytön suhteen. Projekti alkoi joulukuussa 2019 ja Xray:n käyttö oli kaikille asianosaisille uutta. Projektissa on kuitenkin alusta asti tähdätty hyödyntämään Xray:n antamat mahdollisuudet. Valmiita käytänteitä ei siis ollut, vaan ne ovat muotoutuneet projektin edetessä ja niitä kehitetään jatkuvasti.

Käytänteitä on kehitetty enimmäkseen yhteistyönä testaustiimin kesken. Koska opinnäytetyötä varten Xray-laajennukseen tutustuminen tarkemmalla tasolla oli tärkeää, tutustumisen myötä sen käytössä on jouduttu olemaan tarkempia. Olemme sopineet erilaisista merkintäkonventioista ja tehtävien tilojen muutoksista, jotta aineistoa on voinut rajata raportointia varten.

Esimerkiksi Testattava ja NoSystemTest -labelit olivat käytössä alusta asti, joten automatisoitavaksi tarkoitettuja testejä päätettiin alkaa merkitä myös labeleille. Labeleihin turvauduttiin myös myöhemmin, kun huomattiin, että tarvitaan jotain, jolla erottaa ohjelmaan luodut, mutta ei vielä käytössä olevat testitapaukset testauksessa pyörivistä testitapauksista.

Joistakin merkintäkonventioista on pitänyt keskustella koko projektin kesken. Tällainen tapaus oli esimerkiksi se, että kaikki kehityksessä olevat epic-tasoiset tehtävät ei ollut otettu työn alle. Projektissa on vielä etsinnässä parhaat käytänteet testauksen aikana heränneiden kysymysten käsittelyyn ja siihen, miten saadaan merkittyä käyttötilanteista tarkoituksella tekemättä jätetyt poikkeuskäsittelyt.

Yksi esimerkki Xray:hin liittyvien testikäytänteiden kehityksestä projektin aikana on testien jako testien suoritussuunnitelmiin. Aiemmin testit oli jaoteltu toimintojen mukaan ja jokaisessa suoritussuunnitelmassa oli mukana kaikkien eri roolien testit. Tämän kuitenkin huomattiin olevan raskaskäyttöinen systeemi. Helpointa on tehdä useampi eri toiminto samalla käyttäjäroolilla peräjälkeen kuin vaihtaa roolia ja kirjautua aina uudelleen järjestelmään. Näin ollen testejä tehdessä joutui hyppimään testien suoritussuunnitelmasta toiseen, jolloin oli myös vaikea tarkistaa, että kaikki kyseisellä roolilla suoritettavat testit oli tehty ennen kuin vaihtoi oikeudet toiseen rooliin. Nykyisin testit on jaoteltu käyttäjäroolien mukaan ja automatisoinnin myötä osa testeistä tehdään vain yhdellä roolilla.

8.1 Testauksen tavoite

ELTE:n testauksen tavoitteena on varmentaa, että järjestelmä toimii määriteltyjen käyttötilanteiden mukaisesti, ja on sille asetettujen laatuvaatimuksien mukainen. Sprintit kestävät kolme viikkoa ja aina sprintin alussa testausvastaava luo Xray:hin sprinttikohtaisen testisuunnitelman. Sprintissä testataan edellisen sprintin aikana kehitettyjä ominaisuuksia, jotka ovat niin valmiita, että niitä pystytään testaamaan kokonaisena käyttötilanteena. Myös aiemmin toteutetut toiminnallisuudet testataan joka sprintissä riittävän kattavasti, eli mukana on myös regressiotestausta. (Britschgi 2019a.) Sprintissä kehitetty versio tuodaan sprintin lopuksi testiympäristöön, jossa testaus tapahtuu. Testauksen toteuttaa manuaalitestaaja Xray:hin tehtyjen testitapausten ja hänelle osoitettujen testien suoritussuunnitelmien mukaan.

Yllä kuvattu koskee manuaalisesti toteutettavaa järjestelmätestausta. Yksikkö- ja integraatiotestauksesta vastaa järjestelmän toimittaja, eli kehittäjät varmistavat itse järjestelmän

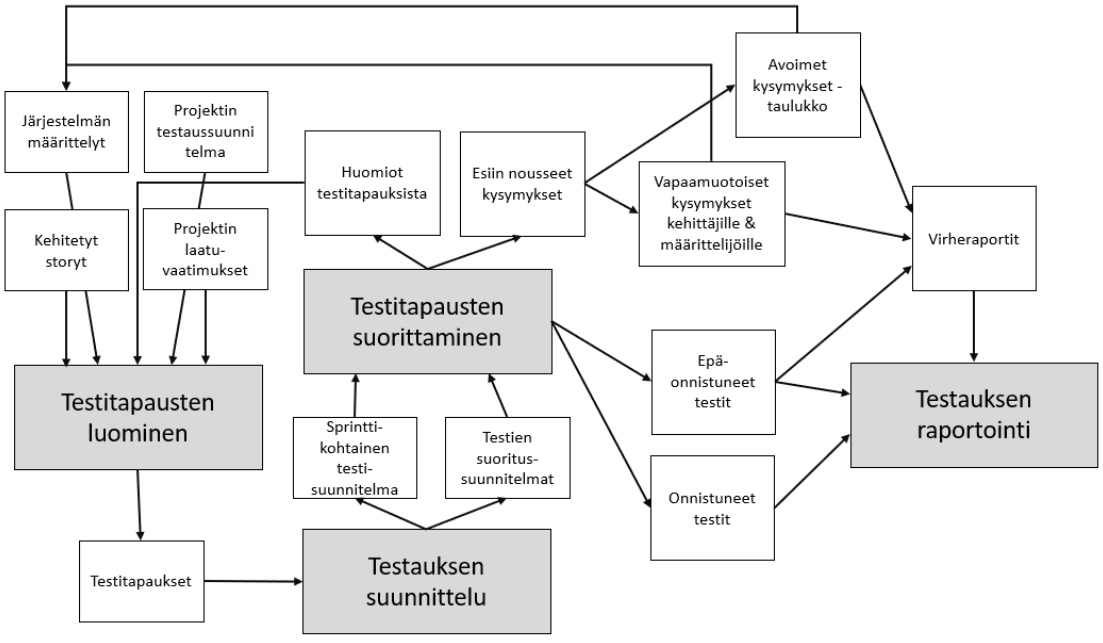
toimivuuden niiltä osin. (Britschgi 2019a.) Tätä testausta ei käsitellä opinnäytetyössä, sillä tätä tekevät kehittäjät eri ohjelmiston avulla, eikä tuloksia raportoida Xray:n kautta tai muutenkaan muulle projektilaisille.

Projektissa suoritetaan myös automaatiotestausta, sillä regressiotestaus pyritään automatisoimaan mahdollisimman pitkälti. Automaatiotesteillä pyritään kattamaan useimmin toistuvat käyttötilanteet, niiden happy day -skenaariot, mutta myös vaihtoehtoisia käyttötilannekulkuja automatisoidaan. Automaatiotestauksessa varmistetaan, että käyttöliittymä toimii odotetulla tavalla ja näytettävä data vastaa tietokannassa olevaa dataa. Tavoitteena on löytää mahdolliset kehitettyjen muutosten aiheuttamat virheet toiminallisuudessa. (Turunen 2020.)

Projektissa tapahtuu myös tutkivaa testausta. Tällainen testaus ilmenee JIRA:ssa vain löydettyjen bugien muodossa, eli osa löydetyistä bugeista ei ole luotu testitapausten suorittamisen yhteydessä. Tutkivaa testausta toteutetaan eniten sprintin alussa, kun uusi kehitysversio tuodaan testiympäristöön ja sen käyttöön tutustutaan.

8.2 Testausprosessi

Projektissa sprinttikohtaisesti toistuva testausprosessi kulkee testitapausten luomisesta testauksen suunnitteluun, jonka jälkeen testitapaukset suoritetaan ja lopuksi suoritettu testaus raportoidaan. Alla olevassa kuviossa 19 nämä päätoiminnot on merkitty harmaalla laatikolla ja nuolilla havainnollistetaan pienempien laatikoiden suuntaa eli ne joko vaikuttavat toimintoon tai ovat sen toiminnon tuotoksia. Kuvio ei ole prosessikaavio vaan visuaalinen esitys testauksen vaiheista.



Kuvio 19: ELTE:n testausprosessi

Testitapausten luomiseen (käsitellään luvussa 8.3.) vaikuttavat järjestelmän määrittelyt, järjestelmän kehityksessä tehdyt story-tyyppiset tehtävät, projektille määritellyt laatuvaatimukset sekä projektin yleinen testaussuunnitelma. Tämän vaiheen tuotoksena on Xray:hin kirjoitetut testitapaukset. Nämä testitapaukset ovat sitten seuraavan vaiheen eli testauksen suunnittelun lähtökohtana. Testitapaukset järjestetään testien suoritussuunnitelmiin ja sprinttikohtaiseen testisuunnitelmaan Xray:ssa. Tämän jälkeen alkaa testitapausten suorittaminen, joka käsitellään luvussa 8.4. Suorittamisen tuotoksina voidaan pitää onnistuneita ja epäonnistuneita testejä, huomioita testitapauksien muotoseikoista sekä esiin nousseita kysymyksiä. Huomiot käsitellään ja testitapaukset päivitetään niiden perusteella. Kysymykset viedään avointen kysymysten taulukkoon (käsitellään luvussa 8.6.) tai niitä kysytään vapaamuotoisesti muilta projektilaisilta. Kysymykset saattavat vaikuttaa järjestelmän määrittelyihin tai niistä tehdään virheraportteja. Virheraportteja luodaan myös epäonnistuneiden testien pohjalta. Niiden luonti käsitellään luvussa 8.5. Viimeinen osuus kaaviosta eli testauksen raportointi on kehitteillä tämän opinnäytetyön puitteissa. Ennen opinnäytetyön yhteydessä kehitettyjä raportteja tapahtunutta raportointia käsitellään luvussa 9.3.

ELTE:ssä käytössä oleva testausprosessi vastaa perusperiaatteiltaan yleistä testauksen prosessia, sen muotoutumiseen ovat vaikuttaneet testaustiimiläisten aiemmat kokemukset, Ruokaviraston käytänteet määrittelyjen, ohjelmistokehityksen ja laatuvaatimusten osalta sekä käytetyn Xray-laajennuksen piirteet. Prosessia on kehitetty aina, kun on tunnistettu ongelmia. Esimerkiksi juuri testauksen aikana nousseet kysymykset, jotka välillä menevät hieman toteutusta ja määrittelyä pidemmälle, näiden käsittelyä on kehitetty suullisista kysymyksistä selkeään taulukkoon, jossa kysymyksien käsittelyä voidaan seurata.

8.3 Testitapaukset

ELTE-projektissa kehitettävän järjestelmän määrittelyissä tunnistetut käyttötilanteet viedään JIRA:an epic-tasoisiksi tehtäviksi. Käyttötilanteiden yksityiskohtaiset määrittelyt elävät järjestelmän kehittämisen ajan, sillä kehitysprosessin aikana tunnistettuja tarpeita ja poikkeuskäsittelysääntöjä viedään koko kehitysprosessin ajan määrittelyn kautta kehittämiseen ja testaamiseen.

JIRA:ssa epic-tasoisiin tehtäviin liitetään aina sprinttikohtaista tekemistä story-tasoisten tehtävien muodossa. Storyt voivat olla esimerkiksi selvittelytöitä, joiden puitteissa haetaan ymmärrystä toteutukseen tulevasta ominaisuudesta ja sen toteuttamisen vaatimuksista. Ne voivat myös olla hyvin teknisiä tehtäviä, kuten uuden tietokantataulun tai tallennussääntöjen luontia. Ne storyt, jotka halutaan testata, liittyvät jonkin määrittelyissä mainitun ominaisuuden toteutukseen, kuten toimintojen näkyvyyteen tietyille rooleille, tapauskohtaisten tietojen näyttämiseen tai toimintojen suorittamiseen.

Käytänteenä on, että aina sprintin lopussa, kun sprintin aikana toteutetut storyt merkataan valmiiksi eli ne siirretään done-tilaan, käy projektin testausvastaava ne yksitellen läpi. Hän merkitsee jokaisen tehdyn storyn joko labelilla **NoSystemTest** tai **Testattava** sen mukaan, tuleeko storyn toteutus testata. Projektilla on muutama tarkistusfiltteri, joilla voidaan tarkistaa, että jokaisella done-tilan storylla on jompikumpi labeli.

Tiivistetysti voidaan siis sanoa, että ELTE-projektissa testaus keskittyy JIRA:ssa done-tilassa oleviin story-tason tehtäviin, joihin on manuaalisesti lisätty labeli ’Testattava’. Tätä varten on luotu koko projektille näkyvä filtteri, jonka nimi on **ELTE storyt done Testattava**, ja jonka JQL-lause on ’**project = ELTE AND issuetype = Story AND status = Done AND labels = Testattava**’. Tämä filtteri on lähtökohtana melkein kaikelle testauksen raportoinnille, koska sillä pystytään rajaamaan aineisto juuri niihin storyihin, joita testataan.

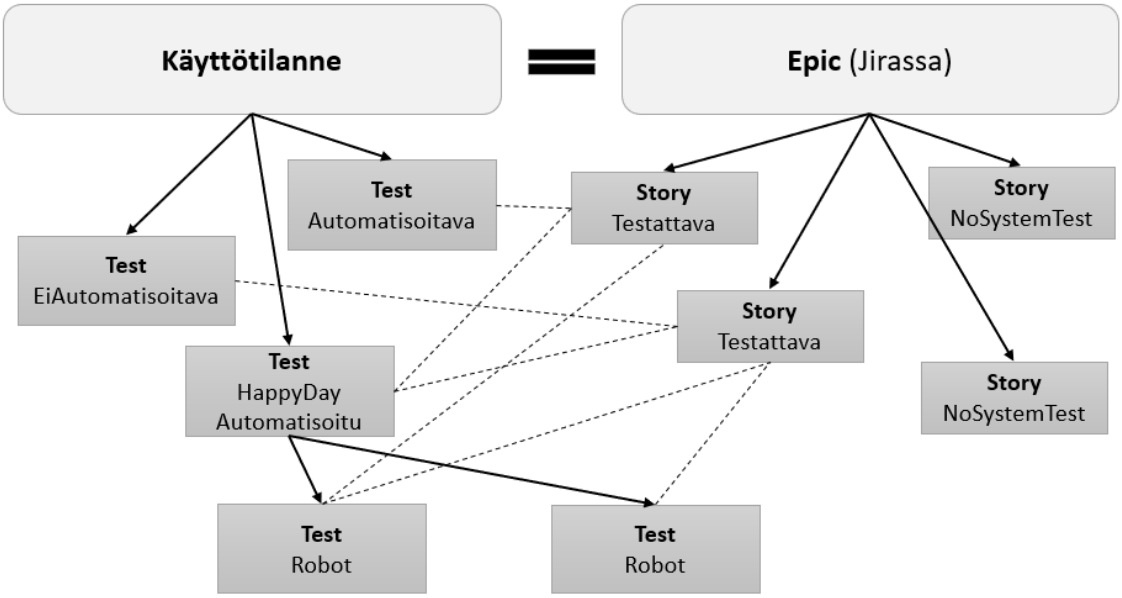
Testauksen aikana suoritettavia testitapauksia ei kuitenkaan suunnitella yksittäisten storyjen pohjalta, vaan ne pohjautuvat käyttötilannekuvauksiin ja niiden vaatimusmäärittelyihin. Jokaisesta käyttötilanteesta tehdään sellainen testitapaus, Happy Day -skenaario, jossa käyttäjä suorittaa kuvatun toiminnon onnistuneesti ilman vaihtoehtoisia tai poikkeuksellisia tapahtumakulkuja. Nämä testitapaukset on merkitty labelilla ’HappyDay’ ja niihin on myös kirjattu tiedossa olevat vaihtoehtoiset tai poikkeukselliset tapahtumien kulut sekä virhetilanteet. Näistä toteutetaan uusia testitapauksia, kun toteutus näille poikkeustilanteille on valmistunut. (Britschgi 2019a.)

Testitapaus tehdään Xray:hin test-tyyppisenä tehtävänä, jolloin siihen voidaan kirjoittaa yleinen kuvaus, johon usein kirjoitetaan käytettävät käyttäjäroolit, sekä kuvata testin suorittaminen askel kerrallaan sekä askelten suorittamiseen tarvittavat syötteet. Kun suunniteltu testitapaus on luotu, se linkitetään sopiviin, ennalta testattavaksi merkittyihin storyihin. Testitapaus tehdään tietyn käyttötilanteen mukaan, joten siihen linkitettävät storyt ovat käyttötilannetta vastaavan epic-tehtävän storyja. Tämän linkityksen ansiosta storyjen ja epic-tehtävien vaatimuksen kattavuusstatus (requirement status) alkaa päivittyä testien suorittamisen jälkeen.

Jokainen luotu testitapaus käydään testaustiimin kesken läpi ja arvioidaan, kannattaako testi automatisoida vai ei. Jos nähdään, että testi on kannattavaa automatisoida, sille annetaan labeli ’Automatisoitava’. Kun manuaalista testitapausta vastaava automaatiotesti on valmistunut ja otettu käyttöön, Automatisoitava-labeli vaihdetaan Automatisoitu-labeliksi. Manuaalista testiä voi vastata yksi tai useampi automaatiotesti. Automatisoitu-labelista näkee myös sen, että testitapaus voidaan jättää pois manuaalitestauksesta.

Uusimpana labelina projektiin lisättiin EiAutomatisointia-labeli niille testitapauksille, joiden automatisointia ei ainakaan tässä kohtaa nähdä järkeväksi. Eli jokaisella testitapauksella pitäisi olla joko Automatisoitava, Automatisoitu tai EiAutomatisointia -labeli.

Tämä edellä kuvattu testien ja storyjen yhteys on nähtävissä kuviossa 20. Eli käyttötilannekuvausta vastaa epic-tason tehtävä JIRA:ssa, ja epic-tehtävään linkitetyt storyt linkitetään sitä vastaavaa käyttötilannetta testaaviin testeihin. Luodut testit merkataan automatisoitava tai automatisoitu -labeleilla, ja tehdyt automaatiotestit linkitetään samoihin storyihin kuin niitä vastaavat manuaalitestit. Automaatiotesteissa on labeli Robot, joka kertoo testin tyypin.



Kuvio 20: Testien liittyminen storyihin ja toisiinsa ELTE:ssä

Testitapauksia päivitetään sitä mukaa, kun käyttötilanteen toteutus laajenee. Jokaisen sprintin alussa käydään läpi edellisen sprintin aikana toteutetut storyt ja suoritetut testit. Näin tarkistetaan, onko suoritetut testit yhä käyttökelpoisia vai onko niiden vaiheisiin tullut muutoksia, esimerkiksi käyttöliittymän muutosten takia. Kun taas toteutettuja storyja käydään läpi, mietitään, että tarvitseeko niiden pohjalta laajentaa suoritettavia testejä tai luoda kokonaan uusia testitapauksia.

8.4 Testien suorittaminen

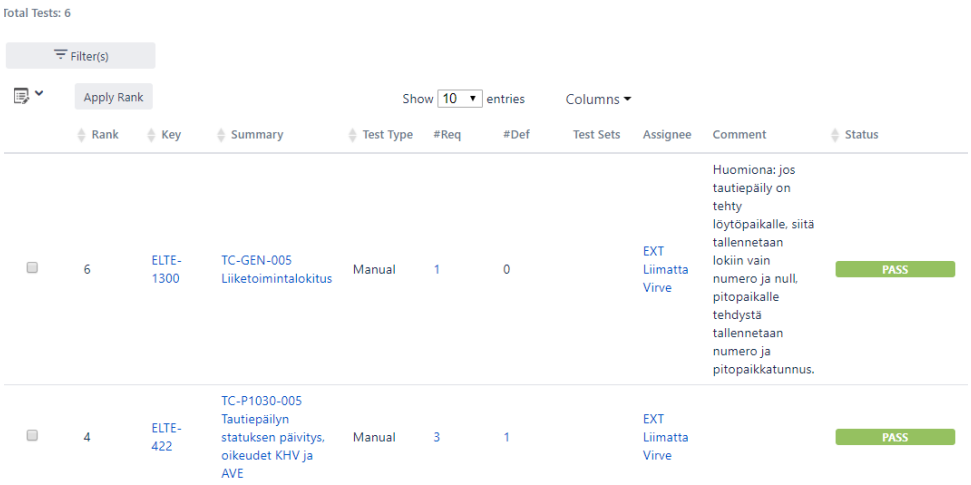
Jokaisen sprintin alussa luodaan sprinttikohtainen testisuunnitelma Xray:n Test Plan -tyyppisenä tehtävänä. Testisuunnitelmaan liitetään testejä ja niistä koostettuja testien suoritussuunnitelmia (test execution). Testisuunnitelmalla on Overall Execution Status -palkki (kuvio 21), joka kertoo, miten testisuunnitelmaan liitettyjen testien suorittaminen on sujunut.

Kuvio 21: Testisuunnitelman Overall Execution Status -palkki

Testit on jaettu testien suoritussuunnitelmiin niin, että kaikki tietyllä käyttäjäroolilla suoritettavat testit ovat samassa (kuvio 22). Automatisoinnin myötä osa testeistä tehdään vain yhdellä roolilla.

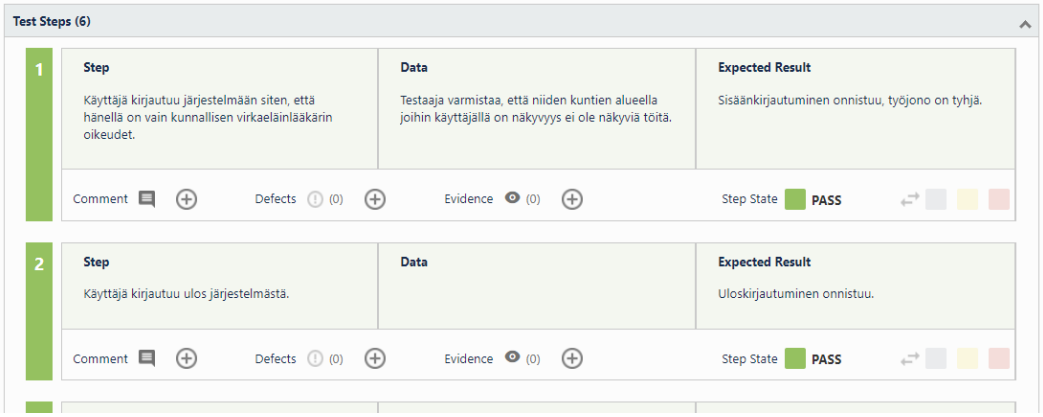
Kuvio 22: Esimerkki testisuunnitelman sisältämistä testien suoritussuunnitelmista

Kun sprinttikohtainen testisuunnitelma on valmis, testausvastaava osoittaa sen sisältämät testien suoritussuunnitelmat manuaalitestaajan työtehtäviksi. Testien suoritussuunnitelmassa (kuvio 23) testit ovat järjestettynä loogiseen järjestykseen, jossa ne voidaan suorittaa. Myös yhdellä testien suoritussuunnitelmalla on samanlainen edistymistä kuvaava palkki kuin testisuunnitelmalla.



Kuvio 23: Esimerkki testien suoritussuunnitelman testeistä

Testi suoritetaan testien suoritussuunnitelmasta valitsemalla testin kohdalta ’execute’. Tällöin testitapaus avautuu ja testin voi suorittaa seuraamalla yksittäisiä vaiheita (kuvio 24). Jos stepin suorittaminen onnistuu, sen tilaksi asetetaan PASS ja siirrytään seuraavaan.



Kuvio 24: Esimerkki testin stepeistä

Jos jokin testin vaihe ei onnistu, stepin tilaksi laitetaan FAIL ja kommenttiin kirjoitetaan huomio siitä, mikä meni vikaan. Jos vika on jo aiemmin raportoitu, steppiin voidaan liittää tehty vikaraportti (defect). Jos taas vika on uusi, stepistä voidaan luoda vikaraportti, jolloin se linkittyy automaattisesti niihin vaatimuksiin, joihin testi on linkitetty. Vikaraportin luomisesta lisää luvussa 8.4. Jos testissä yksikin steppi on FAIL-tilassa, silloin koko testin tilaksi tulee FAIL.

Joissain testitapauksissa on vaiheita, joita ei vielä tämänhetkisellä toteutuksella voi suorittaa tai joiden toteutuksen tiedetään olevan vielä tekemättä. Näitä vaiheita ei laiteta FAIL-tilaan, vaan PASS ja kommenttiin kirjoitetaan, että tämä vaihe ohitetaan. Steppikohtaiset kommentit kannattaa myös kopioida koko testin kommentteihin, jotta ne näkyvät testien suoritussuunnitelman testilistauksessa.

Kun kaikki testiin kirjatut vaiheet on suoritettu, siirrytään takaisin testien suoritussuunnitelmaan ja valitaan seuraava testi. Kun taas testien suoritussuunnitelmasta on tehty kaikki testit, siirrytään seuraavaan suoritussuunnitelmaan, kunnes kaikki testisuunnitelmaan liitetyt on käyty läpi.

8.5 Vikaraportti

Projektissa vikaraportit luodaan JIRA-tehtävienhallintajärjestelmään samalla pohjalla, kuin muutkin tehtävät. Tämä pohja löytyy liitteestä 2. Vikaraportin teko aloitetaan joko samalla tavalla kuin muidenkin Create-toiminnolla tai sitten testin suorittamisen yhteydessä Create defect -toiminnolla. Vikaraportin tehtävätyyppi on Bug ja siihen täytettävät kentät ovat summary, assignee, description, priority, sprint, labels, environment ja attachments.

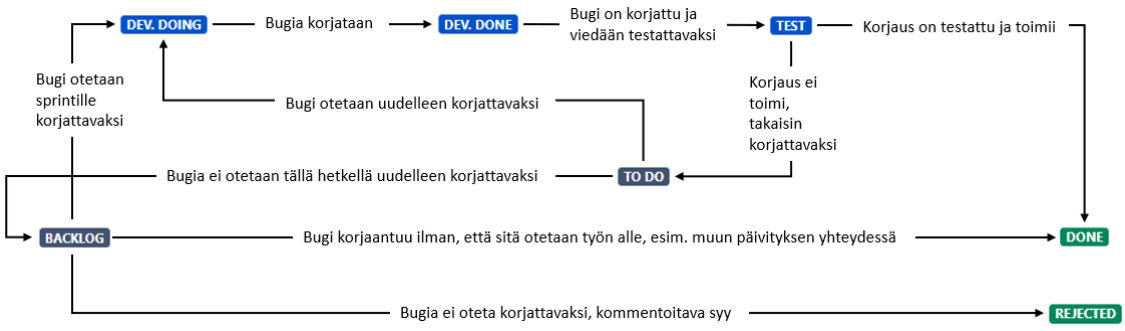
Summary on vikaraportin otsikko, eli lyhyt kuvaus ongelmasta. (Britschgi 2019b.) Pyritään siihen, että otsikko kertoo mahdollisimman yksiselitteisesti raportoidun ongelman. Näin

vikaraportti löytyy helposti listalta. Assignee on testauksessa löytyneillä vioilla aina tuoteomistaja, joka arvioi vian vakavuuden ja priorisoi sen.

Description-kenttään tulee tarkempi kuvaus, jossa kerrotaan, että mitä on tapahtunut, miten tilanne toistetaan ja mitä olisi pitänyt tapahtua. Nämä asiat tulee kuvata niin selkeästi, askel askeleelta, jotta tilanne pystytään toistamaan niiden pohjalta. (Britschgi 2019b.) Jos vikaraporttia on tarpeen päivittää, päivitetyn tekstin yhteyteen kirjoitetaan päivämäärä. Tietoja voidaan päivittää myös kommentteihin, mutta vikaraportin kuvaus ja otsikkoa kannattaa pitää ajantasaisena ja tarkkana, jottei kaikkia kommentteja ole tarpeen lukea.

Muut täytettävät kentät ovat vapaaehtoisia. Näistä oleellisin on attachments eli liitteet, johon voidaan liittää näyttökuvia tai muuta vikaan liittyviä kuvia selventämään raporttia. Environment-kohtaan voidaan laittaa, millä selaimella vika on todettu. Priority on automaattisesti medium ja sen muuttamisesta vastaa tuoteomistaja. Sprint-kenttään tulee se sprintti, jolloin vika korjataan eli se päivitetään silloin, kun vikaraportti otetaan käsittelyyn. Tämän päivittää scrummaster sprintin suunnittelussa sovitun perusteella. Labels-kenttään voi kirjata jonkun sanan, jonka perusteella haluaa luokitella löydettyjä vikoja, tällä hetkellä käytössä on vain käli-labeli, jolla merkitään käyttöliittymään liittyvät vikaraportit.

Kun vikaraportti luodaan, sen statukseksi tulee Backlog. Alla olevassa kuviossa 25 on vikaraportin elinkaari alkutilasta (backlog) lopputilaan, joka on joko done (käsitelty) tai rejected (hylätty). Vika otetaan joko korjattavaksi (dev. doing), se hylätään (rejected) esimerkiksi duplikaattina tai vika korjaantuu jossain muussa yhteydessä, jolloin vikaraportti merkitään käsitellyksi (done). Jos vika otetaan korjattavaksi, se siirretään korjauksen jälkeen testattavaksi ja testaaja joko vahvistaa, että korjaus toimii ja vaihtaa statukseksi done tai jos korjaus ei toimi, niin hän vaihtaa tilaksi to do, jolloin vika otetaan uudelleen korjattavaksi tai siirretään backlogille odottamaan myöhempää ajankohtaa.



Kuvio 25: Vikaraportin statusmuutokset

8.6 Avoimet kysymykset

Testitapauksia luotaessa tai testejä suorittaessa saattaa herätä kysymyksiä järjestelmän käyttäytymisestä. Tällaisia avoimia kysymyksiä varten on luotu taulukko projektin työtilaan, jonne kysymykset ja niihin saadut vastaukset voidaan kirjata, kuten kuviosta 26 näkee.

kysymys	vastaus	bugi?	päivitys	tila
Jos työlista on järjestetty jonkin tiedon mukaan, pitopaikkojen järjestys ei pysy samana, jos käyt jossain tautiepäilyssä ja tallennat sen.	**Käytettävyysongelma. Vietävä backlogille, tehdään bugi.**	ELTE-1372	Bugi luotu 9.4.	BUGI
Hallintopäätökset ja kielivalinnat - käyttäjän kielen mukaan vai eläintenpitäjän kielen mukaan?	**Helpoin ratkaisu: tehdään hallintopäätökset aina molemmilla kielillä.**			MÄÄRITTELY
Hallintopäätöksen tyypin valinta - se ainoa valittava erottuu todella huonosti Windows-koneilla (sekä Firefox että Chrome).	**MVP:ssä vain yksi tyyppi, käyttöliittymän yksinkertaistaminen siis mahdollista joten tämä**		18.3. uudessa versiossa vain yksi hallintopäätöstyyppi valikossa	KORJATTU

Kuvio 26: Avoimien kysymysten taulukko

Alun perin kysymykset listattiin sivulle pelkkänä listana, mutta tällöin lisäselvitykset saattoivat unohtua eikä kysymyksiin koskaan palattu. Nykyisen taulukon värillisistä tilamakroista näkee nopeasti, onko kysymykseen syytä palata vai onko se loppuun käsitelty. Taulukko ei kuitenkaan ole vielä tehokäytössä, sen läpikäyntiä ja päivitystä ei ole aikataulutettu mitenkään. Taulukon jatkokehittely olisi myös tarpeen, nyt kysymykset on ryhmitelty sprinteittäin, mutta ehkä hyödyllisempää olisi laittaa kysymyksen yhteyteen päivämäärä.

9 Raportoinnin tarpeet ELTE-projektissa

Tässä luvussa kuvataan, millaisia tarpeita testauksen raportoinnin suhteen löydettiin tutkimuksen aikana. Projektissa esiintyviä raportoinnin tarpeita etsittiin haastattelulla, kyselyllä sekä havainnoimalla projektin edetessä esiin tulleita kysymyksiä.

Teemahaastattelulla kartoitettiin raportoinnin tarpeita haastattelemalla tiimin raportoinnista eniten kiinnostunutta henkilöä eli projektipäällikköä. Uusien raportointitarpeiden tunnistamiseen taas käytetään verkossa lähetettävää lyhyttä kyselyä. Menetelmän valintaa puolsi se, että sillä saadaan nopeasti ja vaivattomasti aineistoa. Yleinen ongelma menetelmässä on kato, mutta riski päätettiin ottaa. Kyseessä on kuitenkin suhteellisen pieni joukko vastaajia ja kyselyn lähettäjä ei ole kasvoton tutkija, vaan toinen tiimiläinen. Tällaisen ajatuskulun perusteella oli uskottavaa, että melkein kaikki tulisivat vastaamaan kyselyyn, jos kysymykset pidetään selkeinä, lyhyinä ja niiden määrä vähäisenä. Mukaan liitettiin saatekirje, jossa

selvennettiin kyselyn tärkeyttä lähettäjälle ja painotettiin, että kaikkiin kysymyksiin ei ole tarpeellista vastata.

9.1 Projektipäällikön haastattelu

Pääasiana opinnäytetyötä tehdessä pidettiin sitä, että otetaan selvää projektipäällikön tarpeista, eli kuinka hän haluaa raportointia kehitettävän. Tämä selvitettiin haastattelemalla projektipäällikköä teemahaastattelun keinoin.

Haastattelussa oli kolme teemaa: raportoinnin tarpeet, raportoitavat asiat ja raportoinnin tavat. Tarpeista haluttiin tietää, että kenelle ja kuinka usein projektipäällikkö raportoi. Raportoitavista asioista taas, että mitä asioita halutaan raportoinnin kautta saada tietää, mikä kiinnostaa testauksessa. Ja raportoinnin tavoista kysyttiin, että onko jotain toiveita raportoinnin muodoista, millä tavalla sitä halutaan tarkastella.

Projektipäällikön pitää raportoida projektin etenemisestä ohjausryhmälle noin kerran kuukaudessa olevassa kokouksessa. Tällöin läpikäytävää asiaa on niin paljon, että testauksen tilanne tulee raportoida hyvin tiiviistetysti ja nopeasti ymmärrettävässä muodossa. Visuaalisuus olisi avuksi, silloin tilanteen voisi näyttää ja selittää nopeasti. Tärkeintä olisi, että testauksen yleistilanne olisi esitettävissä nopeasti käsitettävässä muodossa. (Pajala 2020.)

Tuotantoon mennessä tarvitaan ohjausryhmän hyväksyntä projektin seuraavalle vaiheelle, jolloin testauksestakin tulee tuottaa laajempi raportti sekä antaa tarkempia tietoja. Tuotantoon siirryttäessä virheille tarvitaan myös statukset, jotka kertovat niiden hyväksyttävyysasteen (ovatko virheet kriittisiä vai lieviä, ja kuinka monta niitä on). (Pajala 2020.)

Haastattelussa tuli ilmi, että eniten testauksen raportoinnissa projektipäällikköä kiinnostaa tietää, mitä on testattu. Raportoinnista halutaan vastauksia kysymyksiin kuten onko kaikki kehitetty työ testattu tai missä vaiheessa testaus menee. Esimerkiksi tieto siitä, mitkä toiminnot on testattu ja mitkä ovat vielä testaamatta olisi kiinnostavaa. (Pajala 2020.)

Toinen kiinnostuksen kohde oli virheet, mitä virheitä on löydetty ja miten ne on käsitelty. Käsittelyllä tarkoitetaan sitä, että onko ne päätetty korjata vai hylätäänkö epäoleellisina ja että milloin korjattavat virheet menevät kehityksen työlistalle. (Pajala 2020.)

Tässä yhteydessä haastattelussa nousi esille huoli virheenkäsittelyn pitkästä ajasta. Kehitys tehdään sprintissä yksi, sprintissä kaksi kehitetty versio viedään testipuolelle ja testataan siellä. Tällöin löydetään virhe. Virhe mahdollisesti otetaan korjattavaksi sprintille kolme ja sprintillä neljä virheen korjaus sitten testataan. Jos taas korjaus ei toimi, se menee sprintillä viisi uudelleen kehitykseen ja testataan taas sprintillä kuusi. Tällöin storyn läpimenoajaksi tulee yhteensä yhdeksän viikkoa, mikä on aika pitkä aika. (Pajala 2020.)

Projektipäällikön mielestä raportointitavaksi joku kirjallinen pohja voisi olla hyvä, jonkunlainen raportti, joka kertoo tämänhetkisen tilanteen. Kiinnostavaa olisi myös eteneminen, jonkinlainen aikajana, josta näkee, miten tila on muuttunut. Myös virheen käsittelyaika voisi kiinnostaa. (Pajala 2020.)

Eli yhteenvetona haastattelusta voidaan sanoa, että on tarvetta visuaaliselle, tiiviille ja ymmärrettävälle kuvaukselle tämänhetkisestä tilanteesta, jonka voi esittää ohjausryhmässä. Kiinnostavia asioita ovat testien kattavuus (eli mitä on testattu ja mitä ei) sekä virhekäsittely.

9.2 Kysely tiimiläisille

Jotta saadaan selville, onko ELTE-projektissa muita tahoja, joita kiinnostaa testauksen tilanteen seuraaminen, suoritettiin pienimuotoinen kysely lähettämällä sähköinen lomake ELTE:n kehityksessä aktiivisesti mukana oleville henkilöille. Kysely suoritettiin Google Docs -palvelussa ja kyselyn kysymyksen löytyvät liitteestä 1. Kyselyssä kysyttiin, onko vastaaja seurannut testausta ja miten, kiinnostaisiko testauksen tilanteesta saada tietoa ja missä muodossa, sekä yleisesti JIRA:n käyttötavoista. Kyselyllä haluttiin kartoittaa, onko tiimissä tunnistamattomia tarpeita testauksen raportoinnille ja miten näihin tarpeisiin voidaan vastata.

Kyselyyn saatiin kahdeksan vastausta, joista seitsemän kertoi seuranneensa ELTE:n testausta. Vastauksia tuli mukavasti tiimissä eri rooleissa toimivilta henkilöiltä, mukana oli kehittäjiä, sisältötiimin henkilöitä sekä testaustiimin jäseniä. Kyselyllä saatiin siis monipuolinen kuva kiinnostuksen kohteista. Enimmäkseen testausta oli tähän saakka seurattu yleisellä tasolla ja suurin osa oli kiinnittänyt eniten huomiota löydettyihin virheisiin. Tietoa testauksesta oli saatu palavereista ja JIRA:sta.

Kaikki kyselyyn vastanneet halusivat saada tietoa testauksen tilasta. Eniten kiinnostivat arvattavasti virheraportit, niiden tila ja kiireellisyys. Myös testauksen (erityisesti automaatiotestauksen) kattavuus ja testien suhde tehtyihin story-tyyppisiin tehtäviin kiinnosti.

JIRA koettiin hyväksi välineeksi, joten sen käyttö pääasiallisena raportointiväylänä vaikutti sopivalta ajatukselta. JIRA:n käytöstä yleisesti saatiin selville, että moni käytti hakusuodattimia löytääkseen tiettyjä kokonaisuuksia tai yksittäisiä tehtäviä nopeammin. Vastanneista muutama mainitsi myös käyttävänsä ohjausnäkymiä ja valmiita raportteja.

Yhteenvetona kyselystä voidaan sanoa, että JIRA:n käyttö raportoinnissa vaikutti perustellulta. Tietotarpeista taas voidaan mainita, että informaatiota vikaraporteista ja testien kattavuudesta kaivattiin enemmän.

9.3 Havainnoidut tarpeet

Opinnäytetyön alkutilanteessa projektissa ei ollut erillistä testauksen raportointia. Testausta pystyi seuraamaan JIRA:n kautta tarkastelemalla sprinttikohtaisia testisuunnitelmia tai muodostamalla Xray:n tarjoamia raportteja, mutta projektilaisia ei opastettu niiden käyttöön. Testaus ilmeni monille tiimiläisille siten, että virhetilanteista keskusteltiin usein kehittäjien ja tuoteomistajan ja määrittelijän kanssa yksityisesti tai palavereissa ennen kuin niistä tehtiin vikaraportteja. JIRA:an kirjattujen vikaraporttien seuraaminen oli myös monille tiimiläisille mitä luultavimmin tuttua aiemmista projekteista.

Projektissa pidetään testaustiimin ja projektipäällikön kesken kerran viikossa tapaaminen, jossa jokainen kertoo vapaamuotoisesti, mitä viimeisen viikon aikana on tehnyt, mitä tekee seuraavaksi ja onko ollut joitain haasteita tai ongelmia, joiden ratkaisemiseksi tarvitaan toimenpiteitä. Projektipäällikkö kirjaa keskustelusta Prikkaan lyhyen koosteen, johon voidaan sitten seuraavalla viikolla palata ja tarkistaa, onko asia ratkennut. Palaveri on 15-30 minuutin mittainen, eikä siihen toivota mitään virallisempaa rakennetta tai täytettävää raporttia.

Manuaalitestaajana opinnäytetyön tekijä on omatoimisesti raportoinut sprintin aikana suoritetusta testaamisesta testausvastaavalle. Tämä on tehty yleensä koostamalla vapaamuotoinen sähköposti, jossa käsitellään testien tekemisessä ilmenneitä haasteita, jotka tarkoittavat esimerkiksi testitapauksissa olevia sanamuotoja, välisteppien puuttumista, testien linkityksiä story-tehtäviin sekä muita käytännön asioita, joita testejä suorittaessa tulee esiin. Myös epäonnistuneet testit käydään läpi, eli kuinka monta niitä oli ja miksi ne epäonnistuivat. Näiden lisäksi kirjataan ylös kaikki testauksen aikaiset huomiot ja kysymykset. Nämä ovat asioita, jotka eivät varsinaisesti olleet testeissä eli testin steppien mukaan testi meni läpi, mutta tapahtui jotain outoa tai samalla jotain muuta huomioitiin. Joskus nämä huomiot saattavat johtaa bugiraportin kirjoittamiseen. Näitä kerätään myös luvussa 9.4. käsiteltyyn avointen kysymysten taulukkoon.

Yhteenvetona, näitä osallistuvan havainnoinnin kautta löydettyjä tarpeita voitaisiin sanoa olevan sprinttikohtainen tieto siitä, miten testaus on sujunut ja miksi jotkin testit ovat epäonnistuneet. Tämän lisäksi luodut virheraportit ja esiin nousseet kysymykset on nähty tarpeellisiksi raportoida eteenpäin.

9.4 Yhteenveto raportoinnin tarpeista

Haastattelulla, kyselyllä ja havainnoimalla kerätyt raportoinnin tarpeet voidaan jakaa kahteen kategoriaan. Jako on tehty listaamalla tarpeet ja ryhmittelemällä niitä analysoimalla niiden ominaisuuksia.

Ensimmäinen kategoria on raportin sisältöön liittyvät tarpeet. Raportoinnin avulla halutaan saada tietää testauksen yleistilanne eli mitä on testattu, mitä on testaamatta ja miten testit liittyvät kehitettyyn työhön. Toinen kiinnostavaksi koettu asiasisältö on virheet ja niiden käsittely.

Toisessa kategoriassa on raportoinnin muotoon liittyvät tarpeet. Näitä ovat visuaalisuus ja tiivis esitysmuoto. Raportin tulisi olla nopeasti selitettävissä ja sisäistettävissä. Toinen tärkeä asia on aika. Ensinnäkin, testauksen nykyhetki kiinnostaa, mikä on tilanne tällä hetkellä, mutta myös se, miten tilanne on kehittynyt. Samoin kuin kiinnostaa, mitä tietyn ajanjakson aikana on tapahtunut.

10 Raportoinnin tasot ja tasojen raportit

Testauksen raportointi jaettiin tasoihin, sillä raportoinnilla nähtiin olevan monta käyttötarkoitusta eikä kaikki tieto kiinnosta kaikkia testauksesta kiinnostuneita. Esimerkiksi kehittäjää kiinnostaa enemmän löydetyt virheet, jotka loppujen lopuksi tulevat mahdollisesti näkymään heidän työjonossaan.

Aineistonkeruumenetelmillä saadut tiedot listattiin paperille ja hahmoteltiin laajemmiksi kokonaisuuksiksi. Näitä kokonaisuuksia analysoimalla aineistosta erotettiin kolme kategoriaa. Ryhmittelyssä huomioon otetut tiedot koskivat sitä, mitä testauksesta halutaan tietää. Analysointia auttoivat yleiset tiedot testauksesta ja etenkin luvussa 4.4. esitetyt ajatukset testauksen raportoinnista.

Yksi selvä kategoria oli testausprosessin yleiskuvaus eli kuinka paljon tehdystä kehitystyöstä on testattu, paljonko testejä eli mitä testataan ja miten. Tämä nimettiin yleistasoksi. Toinen selvästi erottuva kategoria oli sprinttikohtaiset tiedot siitä, miten testaus on mennyt ja mitä ongelmia on havaittu. Näiden tietojen katsottiin liittyvän erityisesti siihen, mitä on tapahtunut tiettynä aikavälinä, joten tämä nimettiin sprinttitasoksi. Kolmanneksi tietotarpeeksi erottui tiedot virheraporteista, kuinka paljon niitä on, millaisia ja millainen on niiden elinkaari. Tämä taso nimettiin bugitasoksi.

Tämän jaottelun jälkeen jokaiseen tasoon kehitettiin omat raportointikäytänteet. Raportteja kehittäessä pidettiin mielessä kaikki kerätty tieto raportoinnin muodosta ja pyrittiin valitsemaan parhaimmat tavat välittää haluttu tieto.

10.1 Raportoinnin hakusuodattimet

Kuten jo aiemmin Xray-laajennuksen kartoituksen yhteydessä on mainittu (luvussa 7.1.), jotta Xray-laajennuksesta saadaan raportointiin relevanttia aineistoa, pitää sitä rajata

hakusuodattimien avulla. Siksi projektiin luotiin useita hakusuodattimia, joilla voidaan hakea tiettyjä asiakokonaisuuksia.

Hakusuodattimien nimeämiseen kehitettiin kaava, joka on helppo oppia ulkoa ja jota pystyy hyödyntämään mahdollisesti muihinkin projekteihin. Nimen tuli olla mahdollisimman lyhyt ja silti kuvaava. Monesti hakusuodattimen nimi on ainoa tieto, joka raportin yhteydessä lukee, joten sen tulisi tukea kaavion tulkintaa ja antaa mahdollisimman paljon informaatiota.

Kaavana on **projektin nimi + issue type + status/label/muu kuva**us. Kolme viimeistä voivat olla kaikki mukana hakusuodattimessa, ne voivat puuttua kokonaan tai niitä on vain yksi. Kaavan ensimmäinen osa on siis projektin nimi ELTE ja toisena on haettavien tehtävien tyyppi (issuetype, esimerkiksi storyt tai testit). Kaavan loppuosa on valinnainen, hakusuodatinta kuvaamaan voi käyttää tehtävän statusta, siihen liitettyä labelia ja/tai muuta vapaamuotoista lyhyttä kuvausta, joka kertoo, miten hakujoukkoa on rajattu. Hakusuodattimet kirjoitettiin taulukkoon (kuvio 27) projektilaisten nähtäville projektin työtilassa olevalle sivulle. Nimi siis koostuu taulukon viidestä ensimmäisestä sarakkeesta, taulukosta löytyy selite sekä suora linkki tehtävälistaukseen JIRA:ssa. Linkkinä toimii JIRA:n antama numerotunniste hakusuodattimelle, joka saattaa näkyä JIRA:ssa esimerkiksi, kun aiemmin luotua raporttia generoi uudelleen. Listasta voi siis tarkistaa myös, mitä hakusuodatinta on viimeksi käyttänyt.

PROJEKTI	ISSUETYPE	STATUS	LABEL	MUU KUVAUS	filtterin selite	Linkki Jiraan (filtterin numero)
ELTE	storyt	done	Testattava	req status UNCOVERED	ELTEn story-issuet, joiden status on done, joilla on label Testattava, ja joiden requirement status on UNCOVERED eli niihin ei ole liitetty vielä yhtään testejä. Tarkistusfiltteri, hakutuloksia ei pitäisi olla	15349
ELTE	testplan			all	Kaikki ELTEn testisuunnitelmat (test plan)	15339
ELTE	testit			all	Kaikki ELTEn test-issuet, jotka ovat valmiita ja testauksessa käytössä	14707
ELTE	testit		Kesken		ELTEn test-issuet, jotka on merkitty labelilla Kesken, eli jotka ovat vielä keskeneräisiä	15315
ELTE	testit		Automatisoitava		ELTEn test-issuet, jotka on merkitty labelilla Automatisoitava, eli joista tehdään automaatiotesti	15310
ELTE	testit		Automatisoitu		ELTEn test-issuet, jotka on merkitty labelilla Automatisoitu, eli joista on tehty automaatiotesti ja joka on käytössä	15311
ELTE	testit		EiAutomatisointia		ELTEn testi-issuet, jotka on merkitty labelilla EIAutomatisointia, eli joista ei ainakaan tämän hetkisen näkemyksen mukaan tehdä automaatiotestejä	15713
ELTE	testit			manuaali	Kaikki ELTEn testi-issuet, jotka ovat manuaalisti testattavia eli joiden test type on manual	15302
ELTE	testit			automaatio	Kaikki ELTEn testi-issuet, jotka ovat automaatiotestattavia eli joiden test type on generic	15303
ELTE	bugit			all	Kaikki ELTEn bugi-issuet	15309
ELTE	bugit	done			Kaikki ELTEn käsitellyt bugit, joiden status on done (eli jotka on korjattu)	15307

Kuvio 27: Projektin käytössä olevia hakusuodattimia

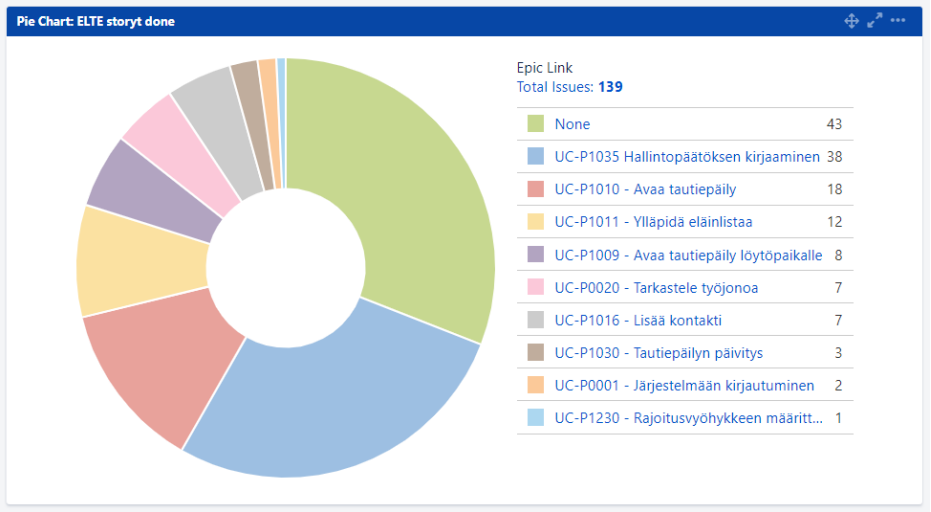
10.2 Yleistaso

Yleistasolla halutaan raportoida se, kuinka paljon projektissa on testattavaa, kuinka paljon siitä on testattu tai testaamatta, sekä kuinka monta testitapausta on luotu ja kuinka monta niistä on automatisoitu tai tullaan automatisoimaan. Tällä tasolla tarkastellaan myös kehitystiimin työtä, eli kuinka monta story-tyyppistä tehtävää on toteutettu kuhunkin käyttötilanteeseen liittyen. Prosenttimäärät ja vertailukelpoiset luvut ovat tärkeitä. Esimerkiksi, kuinka monta prosenttia tehdyistä storyista on merkitty testattavaksi, kuinka monta testiä on

suhteessa storyihin ja mikä on testauksen automatisoinnin aste, eli kuinka monta manuaalisista testitapauksista on automatisoitu.

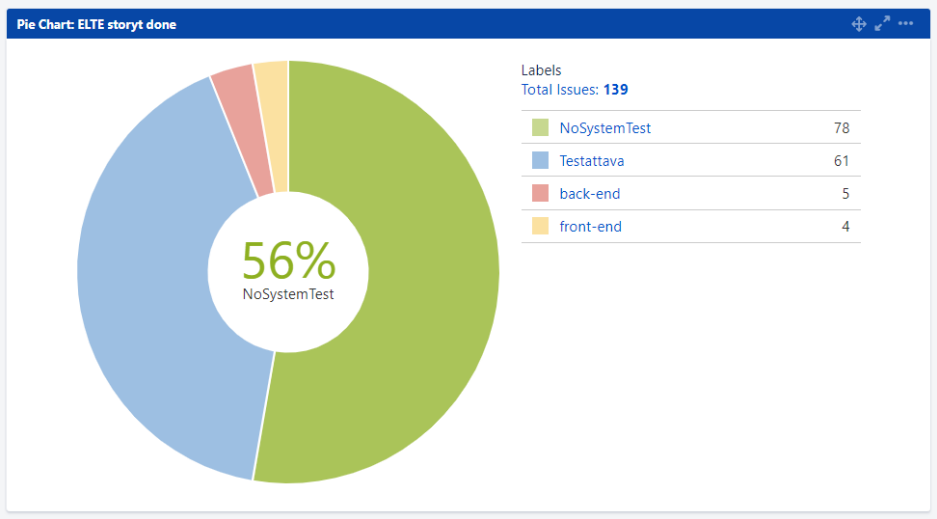
Tätä tasoa varten luodaan JIRA:an ohjausnäkymä (dashboard) nimeltä **ELTE testaus yleisraportti**. Näkymässä käytetään kolmea erilaista vempainta; Pie Chart, Filter Results ja Overall Requirement Coverage. Seuraavaksi esitellään kaaviot yksitellen.

Kuviossa 28 näkyvät kaikki done-tilassa olevat storyt lajiteltuina niihin linkitettyjen epic-tasoisten tehtävien mukaan. Tästä voidaan nähdä, miten paljon työtä mikäkin käyttötilanne on vaatinut tai missä vaiheessa kehitystä käyttötilanne on. Kuviossa esimerkiksi näkyy vain yksi story, joka liittyy käyttötilanteeseen UC-P1230, koska sitä on vasta alettu työstämään. Sen sijaan käyttötilanteeseen UC-P1035 liittyy eniten storyja, se onkin hyvin suuritöinen käyttötilanne PDF-tiedoston generoimisineen.



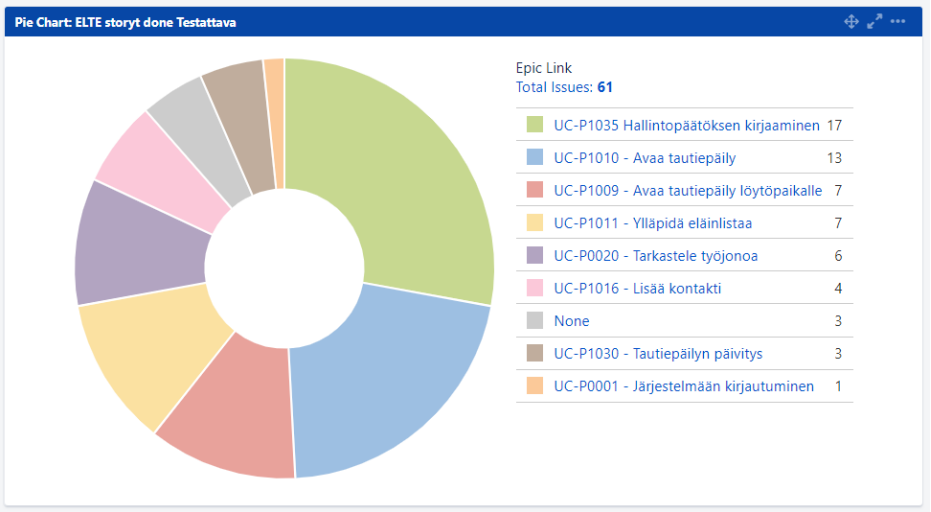
Kuvio 28: Pie Chart - ELTE storyt done – epic link

Piirakkakuviossa 29 kerrotaan, kuinka monta tehdyistä storyista on merkattu testattavaksi (labelilla Testattava) ja kuinka montaa ei voida järjestelmätestauksen yhteydessä testata (labeli NoSystemTest). Kuvion visuaalista puolta hieman hämärtää se, että storyilla voi olla useampi kuin yksi labeli, jolloin piirakkakuvio ei kuvaa 100% storyista vaan yhdeksän storya on kuvattuna kahteen kertaan. Kuviosta nähtävät storyjen luku- ja prosenttimäärät kuitenkin pitävät paikkansa. Kuviota voidaan parantaa keskustelemalla tiimin kesken back-end ja front-end -labeleista. Niitä ilmeisesti käytettiin vain muutamassa sprintissä, jolloin voisi olla, että ne voitaisiin poistaa vanhoista storyista, koska niillä ei nykyisin ole merkitystä sprintin storyjen kannalta.



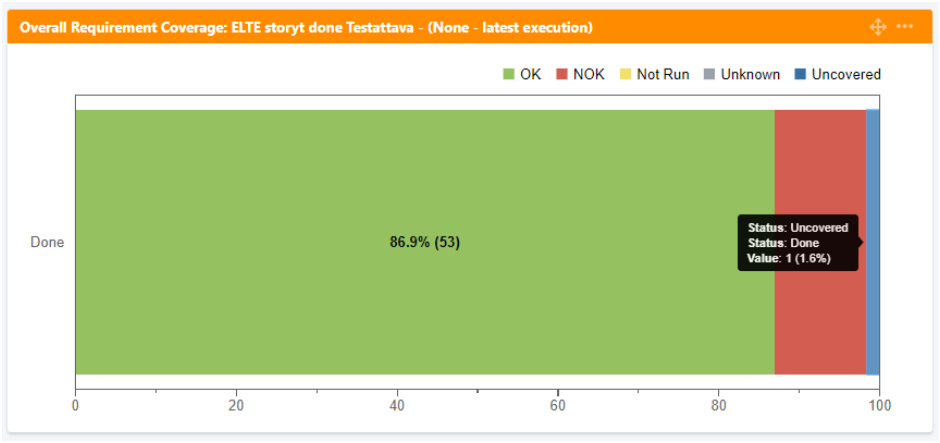
Kuvio 29: Pie Chart - ELTE storyt done – labels

Kuvio 30 on verrannollinen ensimmäiseen kuvioon (kuvio 28), jossa kaikkia valmiita storyja tarkasteltiin epic-tehtävän linkityksen mukaan. Tässä tarkasteltavien storyjen joukkoa on rajattu vain niihin valmiisiin storyihin, jotka on merkitty testattavaksi.



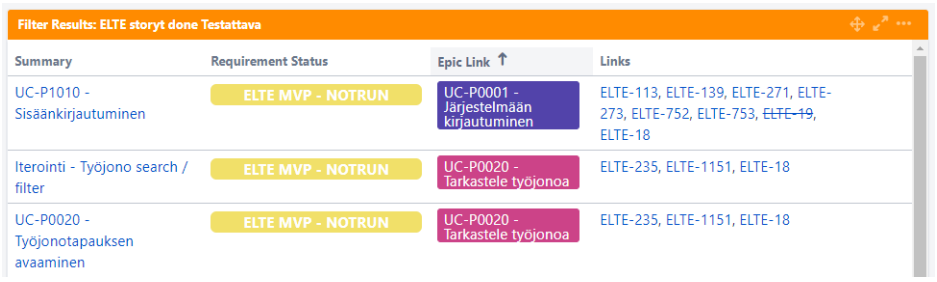
Kuvio 30: Pie Chart - ELTE storyt done Testattava – epic link

Overall Requirement Coverage -vempaimessa (kuvio 31) näkyy, mikä on storyjen vaatimuksen kattavuusstatus (requirement status). Tällä raportoinnin tasolla kiinnostavin osuus on nähdä ne storyt, jotka ovat UNCOVERED-tilassa, eli joihin ei ole vielä liitetty yhtään testiä. Yleensä nämä ovat kesken olevalla tai juuri loppuneella sprintillä valmistuneita storyja.



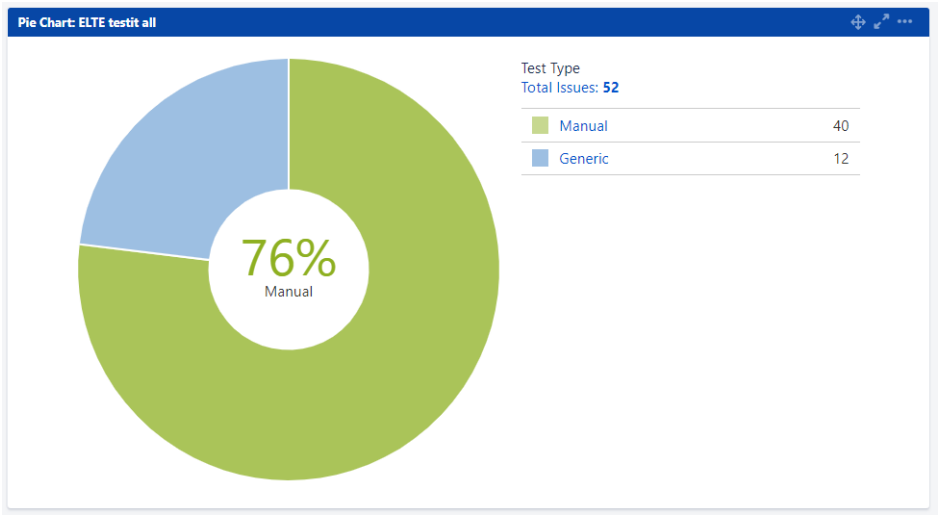
Kuvio 31: Overall Requirement Coverage - ELTE storyt done Testattava

Seuraavassa vempaimessa (kuvio 32) listataan kahdessa edellisessä käytetty aineisto tarkemmin. Tehtävistä näytettävät tiedot on hyvä valita tarkkaan, tälle raportille on valittuna Requirement Status, Epic Link ja Linked Issues. Viimeisin tuo valitettavasti listaan myös muutkin kuin storyyn linkitetyt testit, jotka olisivat olleet kiinnostavimpia.



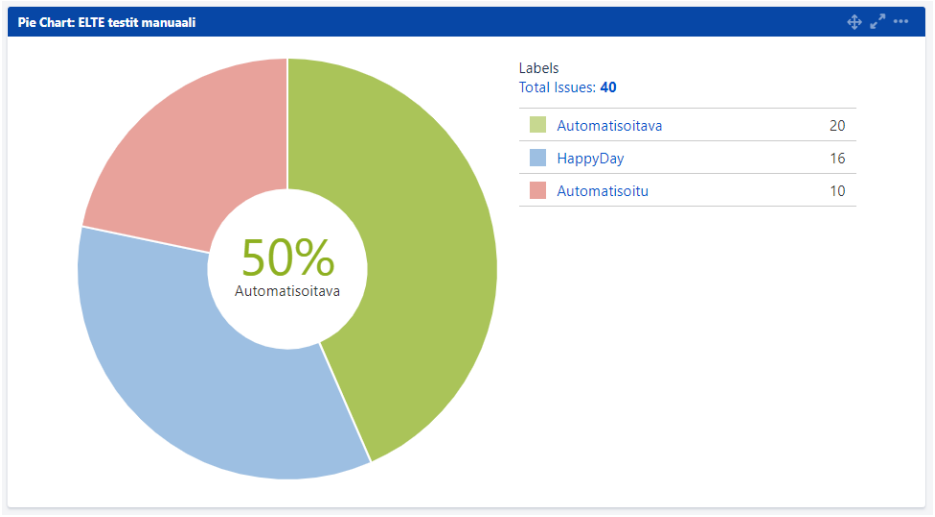
Kuvio 32: Filter results - ELTE storyt done Testattava

Tällä raportoinnin tasolla tarkastellaan testejä enemmänkin niiden määrien mukaan, ei niinkään, että miten testit ovat onnistuneet. Se asia liittyy seuraavan tason raportointiin. Mutta tällä tasolla halutaan kuitenkin tarkastella, kuinka monta testitapausta on luotu projektiin, kuinka monet niistä on automaatiotestejä ja kuinka monet manuaalitestejä. Siksi kuviossa 33 on kaikki valmiit testitapaukset jaoteltuna testin tyypin mukaan, eli kaaviosta näkee, kuinka iso osa testeistä on manuaalisia ja automaattisia.



Kuvio 33: Pie Chart - ELTE testit all

Koska manuaalitestit ovat automaatiotestien pohjana, kuviossa 34 kuvatussa kaaviossa tarkastellaan sitä, kuinka monta manuaalitestiä on merkitty automatisoitavaksi ja kuinka monta niistä on automatisoitu. Tässä kaaviossa on taas sama huono puoli kuin aiemmin eli samalla tehtävällä voi olla useampi labeli. Osaan manuaalitesteistä on laitettu labeli HappyDay merkitsemään sitä, että kyseisessä testissä testataan käyttötilanteen onnistunutta kulkua. Kaaviosta puuttuu myös määrällisesti ne testitapaukset, joita ei nähdä tarpeelliseksi automatisoida, sillä projektissa ei ole sovittu mitään labelia, jolla nämä harvat testit merkattaisiin. Tämä kuitenkin olisi potentiaalisesti mielenkiintoista informaatiota ja tulisi ottaa esille projektissa.



Kuvio 34: Pie Chart - ELTE testit manuaali

Seuraavassa vempaimessa (kuvio 35) on samanlainen tarkempi listaus manuaalisista testitapauksista kuin aiemmin oli testattavista storyista. Linkitetyissä tehtävissä on tällä kertaa niin

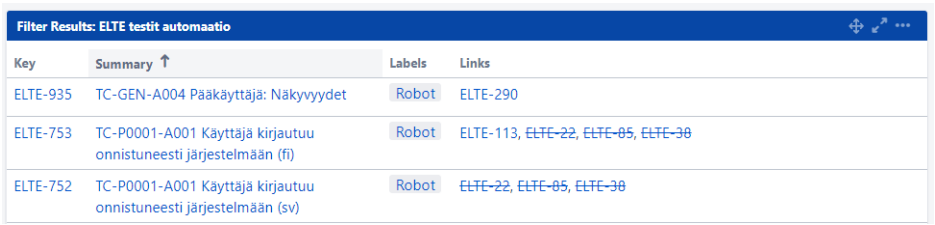
automatisoidut testitapaukset kuin testiin linkitetyt storyt. Valmiit storyt kuitenkin tunnistaa siitä, että niiden avain on yliviivattu.

Filter Results: ELTE testit manuaali

Key	Summary	Labels	Links
ELTE-287	TC-GEN-001 Paikallinen virkaeläinlääkäri: Näkyvyydet	Automatisoitava	ELTE-934
ELTE-288	TC-GEN-002 Alueellinen virkaeläinlääkäri: Näkyvyydet	Automatisoitava	ELTE-1342
ELTE-289	TC-GEN-003 Keskushallinnon viranomainen: Näkyvyydet	Automatisoitava	ELTE-1340
ELTE-290	TC-GEN-004 Pääkäyttäjä: Näkyvyydet	Automatisoitu	ELTE-935

Kuvio 35: Filter Results - ELTE testit manuaali

Ohjausnäkymän viimeinen vempain (kuvio 36) on edellisen listauksen kanssa muuten identtinen, mutta siinä on manuaalisten testien sijaan kaikki automaatiotestit.



Filter Results: ELTE testit automaatio

Key	Summary	Labels	Links
ELTE-935	TC-GEN-A004 Pääkäyttäjä: Näkyvyydet	Robot	ELTE-290
ELTE-753	TC-P0001-A001 Käyttäjä kirjautuu onnistuneesti järjestelmään (fi)	Robot	ELTE-113, ~~ELTE-22~~, ~~ELTE-85~~, ~~ELTE-38~~
ELTE-752	TC-P0001-A001 Käyttäjä kirjautuu onnistuneesti järjestelmään (sv)	Robot	~~ELTE-22~~, ~~ELTE-85~~, ~~ELTE-38~~

Kuvio 36: Filter Results - ELTE testit automaatio

Ohjausnäkymän voi jakaa koko projektin kesken, koska ne on määritelty projektille näkyviksi asetuksista. Näkymän huono puoli on se, ettei sitä voi lähettää linkkinä muille, vaan sen hakeminen ja suosikiksi merkitseminen pitää neuvoa muille. Raporttia voitaisiin vielä kehittää löytämällä tapa jaotella testitapaukset niiden testaaman käyttötilanteen mukaan, jotta voisi arvioida testien laatua ja riittävyyttä käyttötilanteen testaamiseksi. Toinen jatkokehitysidea voisi olla testitapausten määrän kasvu ajan myötä, tästä voisi olla mielenkiintoista nähdä jonkinlaista tilastoa. Tämä asia jätettiin hautumaan raportin kehitysvaiheessa, sillä määriä ei voi seurata testitapausten luontipäivämäärän perusteella, vaan testitapaus otetaan käyttöön niin, että siitä poistetaan labeli. Tämä vaikeuttaa siis tilanteen tunnistamista ohjelmistotasolla.

10.3 Sprinttitaso

Sprinttikohtaisella tasolla testausta tarkastellaan enemmän testien suorittamisen näkökulmasta. Tasolla keskitytään siihen, miten hyvin sprintin aikana suoritettavaksi suunnitellut testit ovat menneet läpi. Perustana on siis sprinttikohtainen testisuunnitelma ja tarkastelun kohteena siihen liitettyjen testien läpimenoprosentit sekä sprintin aikana avatut ja suljetut bugit. Tasoon liittyvä raportointi tulisi luoda joka sprintin yhteydessä uudelleen, jotta aiempien

sprinttien raportteihin voidaan palata. Tämä tuotti ongelmia, sillä oli haastavaa löytää tapa, jolla saadaan talletettua tietyn hetken tilanne.

Tälle tasolle päätettiin luoda Prikkaan oma sivu, koska tarkoituksena oli saada jokaisesta sprintistä oma raporttinsa, eikä ole mitenkään järkevää luoda omaa ohjausnäkymää jokaiselle sprintille. Prikka-sivulle pystytään kuitenkin tuomaan tehtäviä ja käyttämään JQL-lauseita aineiston rajaamiseen.

Ensimmäisenä sivulle tulee linkki sprintin testisuunnitelmaan (kuvio 37). Suunnitelmaa voi tarkastella joko yhtenä tehtävänä, Test Plan Board:ina tai Test Plan -raporttina, joten linkki jokaiseen näihin kolmeen vaihtoehtoon tarjotaan sivulla. Tämän lisäksi Test Plan -tehtävä näytetään omana listana sivulla. Valitettavasti Confluencessa oleva JIRA:n makro ei tarjoa testejä linkkeinä kuten ohjausnäkymällä oleva filter results -vempain. Testisuunnitelman statuksen prosenttimääriä ei myöskään pyöristetä. Tämän takia sivulla oleva lista ei ole ihan niin käyttökelpoinen kuin se voisi olla.

Sprintin 9 testaussuunnitelma. Voit tarkastella suunnitelmaa myös Jiran issuena, Test Plan Board:ina tai Test Plan -raporttina.

Avain	Yhteenveto	Tests Associated With A Test Plan	Test Plan Status
ELTE-1261	Sprint 9 testisuunnitelma	[ELTE-141, ELTE-140, ELTE-287, ELTE-305, ELTE-306, ELTE-362, ELTE-312, ELTE-364, ELTE-288, ELTE-289, ELTE-401, ELTE-235, ELTE-412, ELTE-407, ELTE-422, ELTE-421, ELTE-420, ELTE-494, ELTE-493, ELTE-492, ELTE-1029, ELTE-1030, ELTE-1031, ELTE-1032, ELTE-1300, ELTE-1317, ELTE-1446]	PASS: 25 (92.5925925925926%) FAIL: 1 (3.7037037037037037%) TODO: 1 (3.7037037037037037%)

Kuvio 37: Testisuunnitelman linkit sivulla

Seuraavaksi listataan manuaalisiin testeihin liittyvät testien suoritussuunnitelmat. Tämän makron (kuvio 38) JQL-lause on **project = ELTE AND issuetype = "Test Execution" AND "Test Plan" = ELTE-1261**. Testisuunnitelman avain on ainoa tieto, joka tulee päivittää aina sprinttikohtaista sivua luotaessa. Sarakkeiksi makrossa valittiin avain, tehtävän otsikko, Tests associated with a Test Execution ja Test Execution Status. Test Execution Status on pyöristetty kahden desimaalin tarkkuuteen toisin kuin Test Plan Execution Status edellisellä listalla. Mutta yhäkään testeihin ei tarjota suoria linkkejä.

Sprintin test executionit

Avain	Yhteenveto	Tests Association With A Test Execution	Test Execution Status
ELTE-1269	Sp9: Monta roolia per testitapaus	[ELTE-401, ELTE-420, ELTE-421, ELTE-422, ELTE-364, ELTE-1300]	PASS: 6 (100.0%)
ELTE-1268	Sp9: Keskushallinnon viranomainen	[ELTE-289, ELTE-305, ELTE-306, ELTE-312, ELTE-362, ELTE-407, ELTE-412, ELTE-492, ELTE-493, ELTE-494, ELTE-1317]	PASS: 9 (81.82%) FAIL: 2 (18.18%)
ELTE-1267	Sp9: Alueellinen virkaeläinlääkäri	[ELTE-288, ELTE-305, ELTE-306, ELTE-312, ELTE-362, ELTE-407, ELTE-412, ELTE-492, ELTE-493, ELTE-494, ELTE-1030, ELTE-1031, ELTE-1032, ELTE-1317]	PASS: 13 (92.86%) FAIL: 1 (7.14%)
ELTE-1266	Sp9: Kunnallinen virkaeläinlääkäri	[ELTE-140, ELTE-141, ELTE-235, ELTE-287, ELTE-305, ELTE-306, ELTE-312, ELTE-362, ELTE-407, ELTE-412, ELTE-492, ELTE-493, ELTE-494, ELTE-1029, ELTE-1317]	PASS: 14 (93.33%) FAIL: 1 (6.67%)

4 ongelmaa Päivitä

Kuvio 38: Sprintin testien suoritussuunnitelmat

Automaatiotestien osalta käytetty JQL-lause on monimutkaisempi, sillä siinä tulee määritellä aikaväli, jolla luodut testien suoritussuunnitelmat otetaan listalle, kun aiemmassa aineistoa rajaamaan riitti testisuunnitelman avain. Automaatiotestit eivät ole sprintin testisuunnitelmaan liitettyjä, vaan ne ajetaan Jenkinsissä ja niiden ajotulokset tuodaan Xray:hin vain kerran sprintin aikana. Testien suoritussuunnitelmat erotetaan muista samana aikana luoduista määrittelemällä tehtävän raportoija. Listan (kuvio 39) JQL-lause on project = ELTE AND issuetype = "Test Execution" AND Reporter = "automel8@ruokavirasto.fi" AND (created >= 2020-03-18 AND created <= 2020-04-08). Tässäkin tapauksessa päivämäärät tulee muuttaa jokaisen sprintin mukaisiksi. Sarakkeiksi valittiin samat kuin manuaalitestauksen testien suoritussuunnitelmien listalla, jotta listat ovat yhtenäiset.

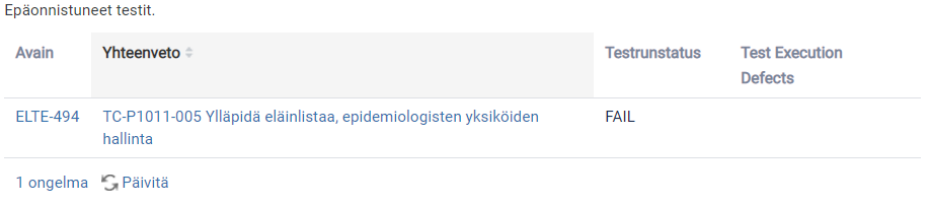
Sprintin aikana ajetut testiautomaatiotestit

Avain	Yhteenveto	Tests Association With A Test Execution	Test Execution Status
ELTE-1335	Automatisoitu testiajo testille ELTE-758	[ELTE-758]	PASS: 1 (100.0%)
ELTE-1325	Automatisoitu testiajo testille ELTE-273	[ELTE-273]	PASS: 1 (100.0%)
ELTE-1324	Automatisoitu testiajo testille ELTE-752	[ELTE-752]	PASS: 1 (100.0%)
ELTE-1323	Automatisoitu testiajo testille ELTE-753	[ELTE-753]	PASS: 1 (100.0%)
ELTE-1322	Automatisoitu testiajo testille ELTE-271	[ELTE-271]	PASS: 1 (100.0%)

5 ongelmaa Päivitä

Kuvio 39: Sprintin aikana ajetut automaatiotestit

Sprintin aikana epäonnistuneet testit esitetään samanlaisella listalla kuin suoritetut testien suoritussuunnitelmat (kuvio 40). Listan JQL-lause on project = ELTE AND issuetype = Test AND "Test Type" = Manual AND (labels = EMPTY OR labels != Kesken) AND TestRunStatus = "ELTE-1261 - FAIL". Tässäkin tarvitsee muuttaa vain ELTE-1261 vastaamaan sprintin testisuunnitelmaa. Tällä listalla yhdeksi sarakkeeksi valittiin Test Execution Defects, jossa on linkki vikaraporttiin, joka on linkitetty epäonnistuneeseen testiin. Tätä listaa tarkastellessa huomaa, että koska testrunstatus tulee aina viimeisimmästä suorituksesta, listalla näkyy vain yksi epäonnistunut testi, vaikka niitä sprintin aikana oli useampikin, niistä osa kuitenkin onnistui eri käyttäjäroolilla tai eri testimateriaalilla. Näissä tapauksissa pitäisi aina sprintin lopussa pitää huoli, että tilasto kuvastaa oikeaa tilannetta, eli tuo testi pitäisi suorittaa uusiksi niin, että epäonnistunut suoritus jäisi uusimmaksi.



Epäonnistuneet testit.

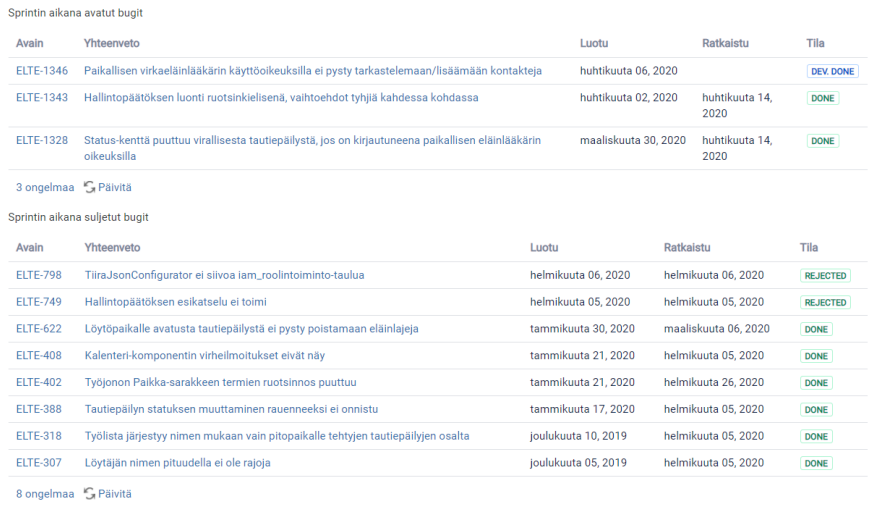
Avain	Yhteenveto	Testrunstatus	Test Execution Defects
ELTE-494	TC-P1011-005 Ylläpidä eläinlistaa, epidemiologisten yksiköiden hallinta	FAIL	

1 ongelma Päivitä

Kuvio 40: Epäonnistuneet testit

Lopuksi raportissa tarkastellaan sprintin aikana avattuja ja suljettuja vikaraportteja (kuvio 41). Avattujen vikaraporttien listan JQL-lause on Project = ELTE AND Type = Bug AND (created >= 2020-03-18 AND created <= 2020-04-08), jossa myös tulee joka sprintin osalta määritellä päivämäärät. Koska listaan halutaan kaikki tuona aikana avatut, ei niiden statusta määritellä JQL-lauseessa, jottei jätetä pois sittemmin suljettuja vikaraportteja. Sarakkeita ovat vikaraportin avain ja otsikko, luontipäivämäärä, ratkaisupäivämäärä, status. Mukana on ratkaisupäivämäärä ja status sen takia, että vikaraportti on saatettu ratkaista jo sprintin aikana tai sitten sen jälkeen.

Suljettujen bugien listan JQL-lause taas on Project = ELTE AND Type = Bug AND (resolved >= 2020-02-05 AND resolved <= 2020-04-08). Tässä listassa on samat tiedot kuin avatuissa bugeissa, jotta listat ovat verrannollisia toisiinsa.



Sprintin aikana avatut bugit

Avain	Yhteenveto	Luotu	Ratkaistu	Tila
ELTE-1346	Paikallisen virkaeläinlääkärin käyttöoikeuksilla ei pysty tarkastelemaan/lisäämään kontakteja	huhtikuuta 06, 2020		DEV. DONE
ELTE-1343	Hallintopäätöksen luonti ruotsinkielisenä, vaihtoehdot tyhjiä kahdessa kohdassa	huhtikuuta 02, 2020	huhtikuuta 14, 2020	DONE
ELTE-1328	Status-kenttä puuttuu virallisesta tautiepäilystä, jos on kirjautuneena paikallisen eläinlääkärin oikeuksilla	maaliskuuta 30, 2020	huhtikuuta 14, 2020	DONE

3 ongelmaa Päivitä

Sprintin aikana suljetut bugit

Avain	Yhteenveto	Luotu	Ratkaistu	Tila
ELTE-798	TiiraJsonConfigurator ei siivoa iam_roolintoiminto-taulua	helmikuuta 06, 2020	helmikuuta 06, 2020	REJECTED
ELTE-749	Hallintopäätöksen esikatselu ei toimi	helmikuuta 05, 2020	helmikuuta 05, 2020	REJECTED
ELTE-622	Löytöpaikalle avatusta tautiepäilystä ei pysty poistamaan eläinlajeja	tammikuuta 30, 2020	maaliskuuta 06, 2020	DONE
ELTE-408	Kalenteri-komponentin virheilmoitukset eivät näy	tammikuuta 21, 2020	helmikuuta 05, 2020	DONE
ELTE-402	Työjonon Paikka-sarakkeen termien ruotsinnos puuttuu	tammikuuta 21, 2020	helmikuuta 26, 2020	DONE
ELTE-388	Tautiepäilyn statuksen muuttaminen rauenneeksi ei onnistu	tammikuuta 17, 2020	helmikuuta 05, 2020	DONE
ELTE-318	Työlista järjestyy nimen mukaan vain pitopaikalle tehtyjen tautiepäilyjen osalta	joulukuuta 10, 2019	helmikuuta 05, 2020	DONE
ELTE-307	Löytäjän nimen pituudella ei ole rajoja	joulukuuta 05, 2019	helmikuuta 05, 2020	DONE

8 ongelmaa Päivitä

Kuvio 41: Sprintin aikana avatut ja suljetut bugit

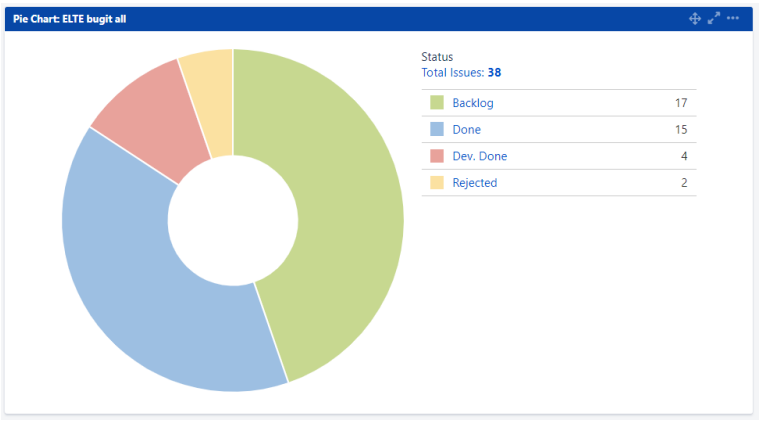
Sprinttikohtainen raportti on helppo kopioida joka sprintille omaksi kokonaisuudekseen ja päivittää makrot näyttämään sprintin tietoja. Prikka-sivu on myös helppo jakaa muille tiimiläisille ja sen päivittämisestä (eli kun uuden sprintin makrot päivitetään sivulle) tulee sivua seuraaville sähköpostilla ilmoitus. Tämä takaa sen, ettei sivu painu unholaan, vaan sitä tullaan aina sprintin lopuksi tarkastelemaan.

10.4 Bugitaso

Kolmas taso keskittyy löydettyihin virhetilanteisiin. Tässä tarkastelun kohteena olisi siis kirjatut virheraportit eli bugit. Tähän liittyy edellisen tason epäonnistuneet testit, varsinkin jos ne ovat aiheuttaneet bugien kirjaamista, mutta mukana on myös muuten järjestelmästä löytyneet virheet/epäkohdat. Monesti vikaraportteja kirjoitetaan huomioista, jotka ovat löytyneet testien ulkopuolella. Tällä tasolla mielenkiinnon kohteena olisi vikaraporttien määrät, ratkaisuprosentit ja elinkaari, etenkin kuinka kauan bugit ovat avoinna.

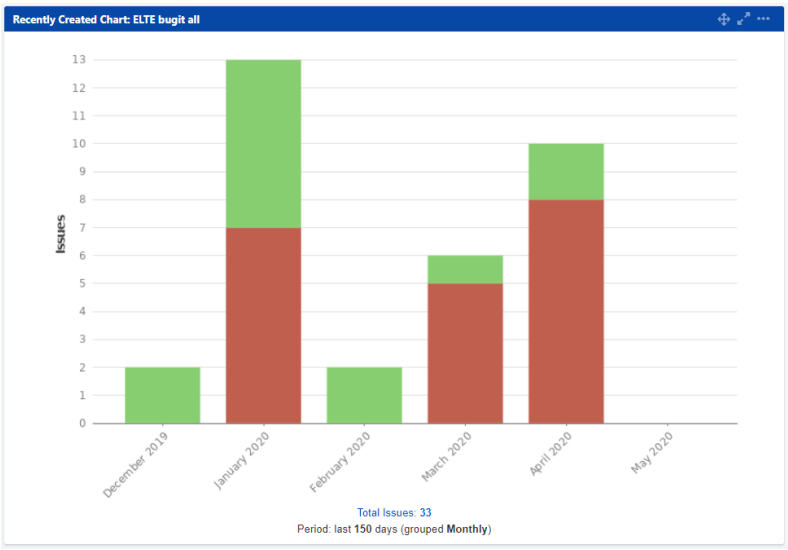
Raportin tulisi olla jatkuvasti päivittyvä ja kuvastaa aina sen hetkistä tilannetta. Siksi projektille luotiin erillinen ohjausnäkymä (dashboard) nimeltään **ELTE testaus bugit**. Kaikkiin raportteihin määriteltiin sama ajanjakso eli 150 päivää ja vikaraportit ryhmiteltiin kuukausittain, jotta ne olisivat vertailukelpoisia keskenään.

Ensimmäisenä vempaimena ohjausnäkymällä on piirakkakuvio (kuvio 42), jossa on kaikki projektissa luodut vikaraportit luokiteltuina niiden statuksen mukaan. Kuviosta näkee hyvin bugien kokonaismäärän sekä yhä avoinna olevat eli ne, joiden status ei ole done tai rejected.



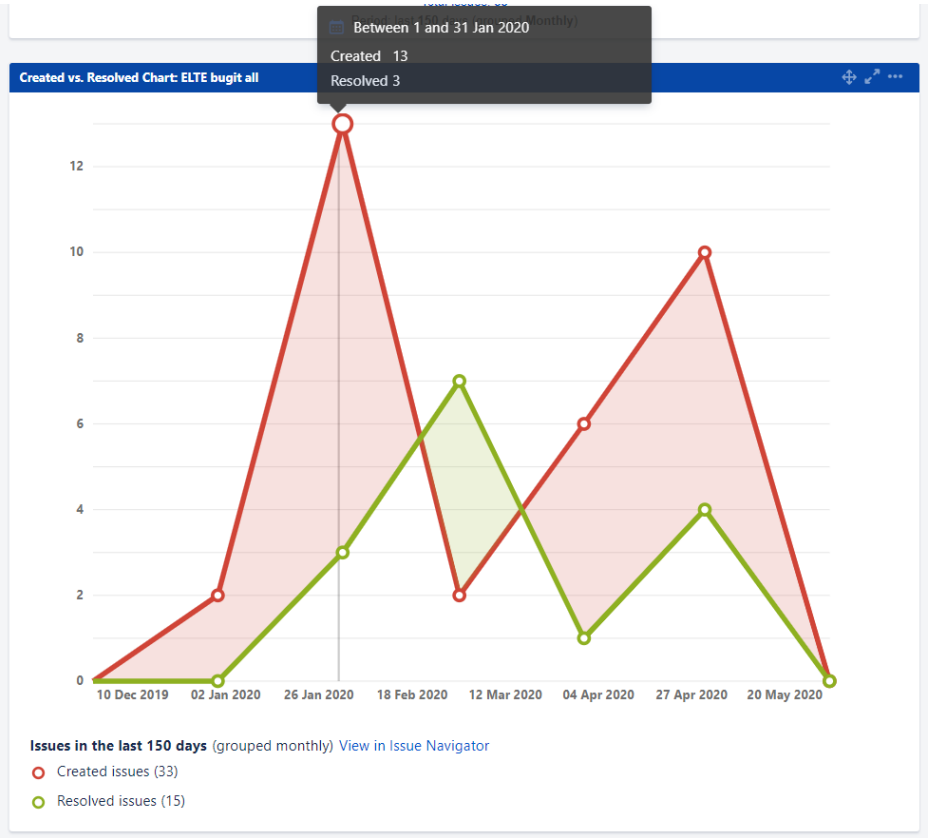
Kuvio 42: Pie Chart - ELTE bugit all - Status

Kuviossa 43 on käytetty vempainta, joka näyttää kuukausittain luotujen bugien määrän. Palkissa vihreä osuus on ne siinä kuussa luodut bugit, jotka on tähän mennessä ratkaistu. Niitä ei siis ole ratkaistu siinä kuussa, vaan nykyhetkeen mennessä. Eli esimerkiksi kaikki helmikuussa ja joulukuussa luodut bugit on ratkaistu. Parhaiten tämä tilasto toimii siinä, että tästä näkee, milloin vikaraportteja on eniten luotu ja miten kaukaa on yhä avoinna olevia bugeja.



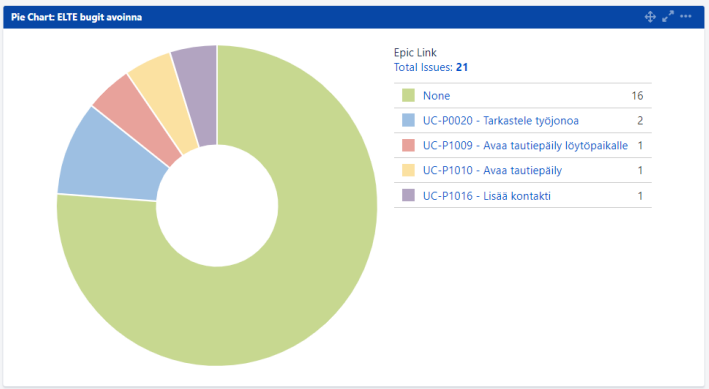
Kuvio 43: Recently Created Chart - ELTE bugit all

Edelliseen kaavioon verrannollinen on kuviossa 44 esitelty Created vs. Resolved Chart. Tässä kaaviossa näkyy samat määrät kuukausittain luotujen vikaraporttien osalta, mutta kaaviosta näkee myös, kuinka monta vikaraporttia on ratkaistu sen kuukauden aikana.



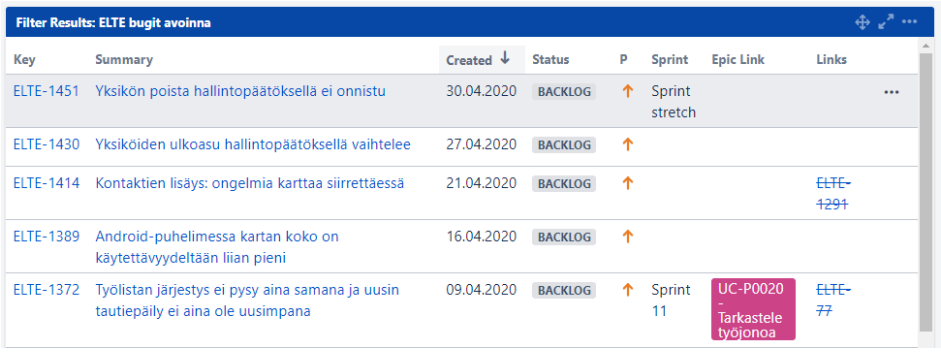
Kuvio 44: Created vs. Resolved Chart - ELTE bugit all

Seuraavaksi keskitytään tällä hetkellä avoinna oleviin vikaraportteihin. Nopea ja visuaalinen kaavio avoinna olevista bugeista on piirakkakaavio (kuvio 45), jossa vikaraportit on ryhmitelty linkitetyn epic-tehtävän mukaan. Kaikkiin bugeihin ei tuota linkkiä ole laitettu, joten vempain ei anna niin paljon informaatiota kuin voisi. Tämä vempain kaipaa siis jatkokehitystä.



Kuvio 45: Pie Chart - ELTE bugit avoinna

Avointen bugien listauksessa (kuvio 46) halutaan nähdä bugin luontipäivämäärä sekä sen tämänhetkinen status, eli onko vikaraportti yhä backlogilla vai korjauksen alla. Valitettavasti listaan ei saa tietoa siitä, kuinka monta päivää bugi on ollut avoinna. Listassa on myös mukana bugin prioriteetti, vaikka se tällä hetkellä on kaikilla bugeilla sama. On kuitenkin ollut puhetta, että testausvastaava alkaa käydä bugeja aktiivisemmin läpi tuoteomistajan kanssa, ja he priorisoivat bugeja paremmin, jotta niiden eriarvoisuus tulee paremmin esille. Osa bugeista nimittäin on vähäpätöisiä ja tulevat luultavasti korjautumaan, kun käyttöliittymäsuunnittelua tehdään. Sprint-sarakkeessa näkyy sprintin numero, jos bugi on nostettu backlogilta sprintille ja on näin priorisoitu korjattavaksi. Kaksi viimeistä saraketta on linkkejä bugiin liittyviin epic-tehtäviin, storyihin ja testeihin. Näiden merkityksestä tulee käydä projektin kesken keskustelua, sillä kuten näkyy, niiden käyttö ei ole yhdenmukaista, vaan välillä linkitys puuttuu kokonaan.



Filter Results: ELTE bugit avoinna

Key	Summary	Created	Status	P	Sprint	Epic Link	Links
ELTE-1451	Yksikön poista hallintopäätöksellä ei onnistu	30.04.2020	BACKLOG	↑	Sprint stretch		
ELTE-1430	Yksiköiden ulkoasu hallintopäätöksellä vaihtelee	27.04.2020	BACKLOG	↑			
ELTE-1414	Kontaktien lisäys: ongelmia karttaa siirrettäessä	21.04.2020	BACKLOG	↑			~~ELTE-1291~~
ELTE-1389	Android-puhelimessa kartan koko on käytettävyydeltään liian pieni	16.04.2020	BACKLOG	↑			
ELTE-1372	Työlistan järjestys ei pysy aina samana ja uusin tautiepäily ei aina ole uusimpana	09.04.2020	BACKLOG	↑	Sprint 11	UC-P0020 - Tarkastele työjonoa	~~ELTE-77~~

Kuvio 46: Filter Results - ELTE bugit avoinna

Suljetuista bugeista on samanlainen vempain kuin avoimista bugeista (kuvio 47), mutta tässä on valittu hieman erilaisia sarakkeita. Luontipäivämäärän lisäksi löytyy bugin sulkemisen päivämäärä. Valitettavasti yhäkään tällä vempaimella ei saada näkyviin aikaa, jonka bugi on ollut avoinna.

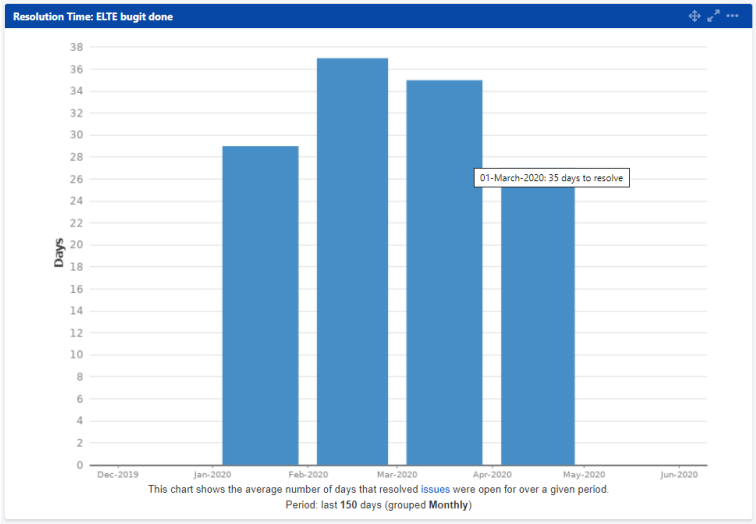
Filter Results: ELTE bugit done

Key	Summary	Created	Resolved	Links
ELTE-1452	Työjono ja kartalla olevien pitopaikkojen näkyydet eivät päivity roolin muutoksen myötä	30.04.2020	30.04.2020	
ELTE-1343	Hallintopäätöksen luonti ruotsinkielisenä, vaihtoehdot tyhjiä kahdessa kohdassa	02.04.2020	14.04.2020	~~ELTE-277~~, ~~ELTE-279~~, ELTE-678
ELTE-1328	Status-kenttä puuttuu virallisesta tautiepäilystä, jos on kirjautuneena paikallisen eläinlääkärin oikeuksilla	30.03.2020	14.04.2020	~~ELTE-255~~

Kuvio 47: Filter Results - ELTE bugit done

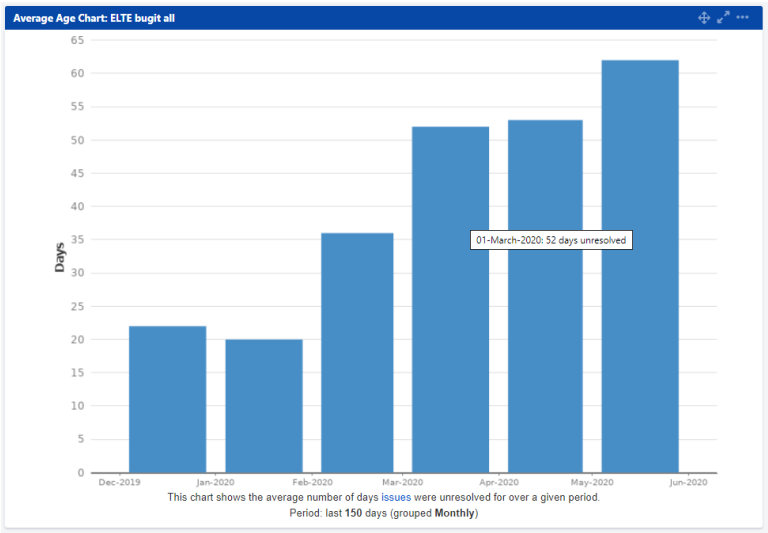
Suljettujen bugien elinikä on yksi kiinnostavimmista asioista, joihin tällä seuraavalla vempaimella yritetään vastata. Resolution Time -nimisellä vempaimella (kuvio 48) voidaan tarkastella kussakin kuussa suljettujen bugien keskimääräistä elinikää. Palkissa ovat kaikki

kyseisessä kuussa ratkaistut bugit ja keskiarvo niiden aukiolopäivistä. Palkkia klikkaamalla pääsee tarkastelemaan ratkaistuja bugeja, tuossa kaavion osoittamassa tapauksessa oli vain yksi bugi, joka oli ollut auki 35 päivää.



Kuvio 48: Resolution Time - ELTE bugit done

Viimeinen kaavio (kuvio 49) näyttää hyvin samanlaiselta kuin edellinen, mutta tässä tarkastellaan kaikki bugeja, niin avoimia kuin suljettuja ja hylättyjä ja niiden keskimääräistä elinikää. Eli keskimääräisesti kuinka kauan kunakin kuukautena ovat vikaraportit olleet avoinna. Tässä tilastossa on siis mukana myös yhä avoinna olevat bugit suljettujen lisäksi. Kaaviosta voi nähdä trendin, että jotkut bugit ovat yhä pidempään ja pidempään auki ja ne nostavat keskiarviota kuukausi kuukaudelta korkeammalle.



Kuvio 49: Average Age Chart - ELTE bugit all

Tämä ohjausnäkymälle rakennettu raportti tuo eniten uutta tietoa projektille, sillä aiemmin ei ole tarkasteltu vikaraporttien elinkaaren pituutta tai avoinna olevien bugien ikää. Raportissa on kuitenkin samat haasteet kuin yleistason ohjausnäkymällä, eli se on hiukan hankala saada kaikkien projektin henkilöiden käyttöön, koska suoraa linkkiä raporttiin ei voida antaa.

11 Yhteenveto ja johtopäätökset

Opinnäytetyössä selvitettiin, kuinka Xray-testauslaajennuksen avulla voidaan raportoida järjestelmätestauksen tila ketterässä ohjelmistokehitysprojektissa projektikohtaisten tarpeiden mukaisesti. Työ toteutettiin Ruokaviraston toimeksiannosta ja testausta tutkittiin ELTE-projektissa, joka toimi esimerkkinä ketterästä ohjelmistokehitysprojektista.

Tietoperustana opinnäytetyössä oli ketterä ohjelmistokehitys ja ohjelmistotestaus. Näihin pohjautuen työssä kuvattiin projektin testausprosessi. Työssä hyödynnettiin myös kvalitatiivisen tutkimuksen menetelmiä kuten teemahaastattelua, lomakekyselyä sekä osallistuvaa havainnointia raportoinnin tarpeiden löytämiseksi. Näiden lisäksi Xray-testauslaajennuksen mahdollisuudet kartoitettiin tutkimalla ohjelman käyttöä.

Opinnäytetyössä tunnistetut projektin raportointitarpeet jaettiin kolmeen tasoon; yleiseen, sprinttikohtaiseen ja virhetilanteisiin keskittyvään raportointiin. Yleistasolla halutaan tietoa testauksen kattavuudesta ja testimääristä, sprinttitasolla halutaan saada kuva, miten testaus on sujunut sprintin aikana ja bugitasolla halutaan tarkastella tarkemmin vikaraportteja sekä niiden elinkaarta ja elinikää. Kahdelle tasolle kehitettiin jatkuvasti ajan tasalla pysyvä raportti, joka toteutettiin JIRA:n ohjausnäkymänä vempainten avulla. Sprinttikohtaisessa raportissa haluttiin saada talteen tietyn ajankohdan tilanne, joten raportti toteutettiin Confluence-sivulle makrojen avulla. Tämä raportti on helppo kopioida ja päivittää vastaamaan aina uusinta sprinttiä. Raporttien kehittämisellä vastattiin opinnäytetyön tutkimuskysymykseen, joka oli, että kuinka Xray-laajennuksen avulla voidaan raportoida testauksen tila ketterässä ohjelmistokehitysprojektissa.

Tietojärjestelmien suunnittelutieteen periaatteiden mukaisesti tässä opinnäytetyössä siis haettiin ratkaisua liike-elämän tarpeeseen. Tarpeena oli tuottaa tietoa testauksen tilasta Xray-ohjelmiston avulla. Tätä tarvetta vastaamaan luotiin artefakti eli raportointimalli ketterän ohjelmistokehitysprojektin käyttöön. Artefaktin tarkoituksena oli antaa projektissa ilmenevien tarpeiden mukaista informaatiota testauksesta. Artefaktin kehittämiseen vaikutti ympäristö eli projekti, jonka tarpeita tutkittiin, sekä ohjelmisto, jonka mahdollisuuksia kartoitettiin. Tietopohjana kehittämiselle oli ketterän ohjelmistokehityksen ja ohjelmistotestaukset teoriat. Opinnäytetyössä ei käsitelty artefaktin eli raportointimallien arviointia eli käytännön palaute mallien toimivuudesta puuttuu. Mallit kuitenkin otetaan käyttöön esimerkkinä

olleessa ELTE-projektissa ja niiden kehittämistä on tarkoitus jatkaa käytännön kokemuksen ja ohjelman tutkimisen perusteella.

Opinnäytetyön aikana tuli selväksi, että ajantasaisen ja oikeita asioita kuvaavan raportoinnin saavuttamiseksi tarvitaan yhteistyötä, kommunikaatiota ja sovittuja käytänteitä, joita noudatetaan. Kaikille tulee olla selvää, miten asioita merkitään tehtäviin, mitkä kentät tulee täyttää ja miten tehtävät tulee linkittää toisiinsa. Raportoinnin laadun ylläpitämiseksi taas tarvitaan tarkistuksia ja korjauksia, joilla pidetään huolta, että raportti kertoo yhä oleellisen. Tärkeää on myös tuntea testauksen käytänteet, miten testitapaukset rakennetaan ja mihin ne liittyvät, jotta aineistoa osaa rajata oikein ja osaa arvioida, mitä asioita kannattaa esittää raportissa.

Opinnäytetyön reliabiliteetin osalta tutkimuksen kulku on kuvattu työssä tarkalla tasolla. Tiedonkeruumenetelmät ja niillä kerätyt tiedot on esitelty työssä. Esimerkiksi havainnoitu testausprosessi on dokumentoitu tarkasti ja esitettyä asiaa on selvennetty kaavioiden avulla. Kyselyn kysymykset ja teemahaastattelun teemat on esitelty työssä hyvin ja niillä kerätyt vastaukset on esitelty pääpiirteittäin. Tämä todentaa myös työn validiteettia ja sen arviointia. Kysymykset ovat nähtävissä ja niiden laatua voidaan arvioida, sekä miettiä, tutkittiinko sitä, mitä oli tarkoituskin. Kehitetyn raportoinnin esittelyosuudessa ei pohdittu vaihtoehtoja, mutta tehdyt valinnat perusteltiin ja vaihtoehdot kuitenkin tulivat ilmi ohjelmiston mahdollisuuksien kartoitusvaiheessa, joka on dokumentoitu työhön.

12 Jatkokehitysehdotukset

Vaikka opinnäytetyö keskittyi kuvaamaan testausta ja sen raportointia tietyn projektin näkökulmasta, voidaan projektin sanoa edustavan Ruokavirastossa toteutettavia ketteriä ohjelmistokehitysprojekteja. ELTE-projektissa nimittäin seurataan Ruokaviraston ohjelmistokehitys- ja projektinhallintamalleja sekä noudatetaan viraston ohjeistusta laadunhallintaan. Tämän perusteella opinnäytetyön tulokset ovat siis yleistettävissä koko organisaation tasolle.

Opinnäytetyön tuloksia tullaankin hyödyntämään kahdella tapaa. Kehitetyt raportointitavat otetaan käyttöön esimerkkiprojektissa, mutta opinnäytetyön pohjalta tullaan myös laatimaan Ruokavirastolle ohjeistus Xray-testauslaajennuksen käytöstä raportoinnin välineenä.

Projektissa jatketaan raportoinnin kehitystä käytön aikana esiin tulevien huomioiden perusteella. Raportit ollaan vasta nyt ottamassa koko projektin käyttöön sprintin vaihtuessa, joten palautetta ei olla vielä saatu. Tavoitteena on pitää huolta, että raportit pysyvät ajantasaisina. Myös palautteen keruu on tärkeää, raporteista tulisi olla hyötyä koko projektille ja niitä halutaan kehittää mahdollisten ehdotusten mukaisesti.

Keskusteluja yleisohjeistuksen laatimisesta osaksi Ruokaviraston ohjelmistotuotannon käsikirjaa jatketaan Ruokaviraston kanssa. Opinnäytetyöprosessin aikana on ilmennyt kiinnostusta ELTE-projektin testaukseen ja raportointiin muiden projektien puolelta. Tämän takia on myös mahdollista, että opinnäytetyön tuloksena kehitettyjä raportointitapoja esitellään tai muita opastetaan samankaltaisten raporttien luomisessa. Tavoitteena on jakaa projektissa kertynyttä osaamista, jotta tiedosta on laajemminkin hyötyä.

Lähteet

Painetut

Hirsjärvi, S. & Hurme, H. 2001. Tutkimushaastattelu, teemahaastattelun teoria ja käytäntö. Helsinki: Yliopistopaino.

Hirsjärvi, S. Remes, P. & Sajavaara, P. 2007. Tutki ja kirjoita. 13. painos. Helsinki: Kustannusosakeyhtiö Tammi.

Juvonen, R. 2018. Ohjelmistoprojektin sudenkuopat ja miten ne vältetään. Helsinki: Books on Demand.

Järvinen, P. 2006. Onko innovaatioiden suunnittelu tiedettä? Lehdessä Systeemityö 2/2006, sivut 25-27.

Kananen, J. 2013. Case-tutkimus opinnäytetyönä. Jyväskylä: Jyväskylän ammattikorkeakoulu.

Kasurinen, J. 2013. Ohjelmistotestauksen käsikirja. Jyväskylä: Docendo.

Sähköiset

Atlassian. 2020. Confluence: What is Confluence? Viitattu 7.5.2020. https://confluence.atlassian.com/confeval/confluence-evaluator-resources/confluence-what-is-confluence

Braz, H. 2019. Test Execution. Viitattu 15.11.2019. https://confluence.xpand-it.com/display/public/XRAY/Test+Execution

Carroll, N. 2018. Design Thinking / Science. Viitattu 14.11.2018. https://noel-carroll.com/research-2/design-thinking/

Duarte, D. 2019a. Traceability Report. Viitattu 28.11.2019. https://confluence.xpand-it.com/display/public/XRAY/Traceability+Report

Duarte, D. 2019b. Overall Requirement Coverage Report. Viitattu 28.11.2019. https://confluence.xpand-it.com/display/public/XRAY/Overall+Requirement+Coverage+Report

Duarte, D. 2019c. Historical Requirement Coverage Report. Viitattu 28.11.2019. https://confluence.xpand-it.com/display/public/XRAY/Historical+Requirement+Coverage+Report

Duarte, D. 2019d. Test Plans Report. Viitattu 28.11.2019. https://confluence.xpand-it.com/display/public/XRAY/Test+Plans+Report

Duarte, D. 2019e. Test Executions Report. Viitattu 29.11.2019. https://confluence.xpand-it.com/display/public/XRAY/Test+Executions+Report

Duarte, D. 2019f. Test Runs Report. Viitattu 29.11.2019. https://confluence.xpand-it.com/display/public/XRAY/Test+Runs+Report

Freire, S. 2019a. Xray Documentation Home. Viitattu 14.11.2019. https://confluence.xpand-it.com/display/public/XRAY/Xray+Documentation+Home?key=XRAY

Freire S. 2019b. Terms and Concepts. Viitattu 14.11.2019. https://confluence.xpand-it.com/display/public/XRAY/Terms+and+Concepts

Freire, S. 2019c. Understanding coverage and the calculation of Test and requirement statuses. Viitattu 5.12.2019. https://confluence.xpand-it.com/display/public/XRAY/Understanding+coverage+and+the+calculation+of+Test+and+requirement+statuses

G, C. 2017a. Organizing. Viitattu 14.11.2019. https://confluence.xpand-it.com/display/public/XRAY/Organizing

G, C. 2017b. Executing. Viitattu 15.11.2019. https://confluence.xpand-it.com/display/public/XRAY/Executing

G, C. 2017c. Requirement Projects. Viitattu 27.11.2019. https://confluence.xpand-it.com/display/public/XRAY/Requirement+Projects

International Software Testing Qualifications Board (ISTQB). 2018. Sertifioitu Testaaja, Perustason sertifikaattisisältö. Tulostettu 9.12.2019. http://www.fistb.fi/sites/fistb/files/liitteet/CTFL%202018%20Sertifikaattisisa%CC%88lto%CC%88%2020181010-1%20Valmis.pdf

International Software Testing Qualifications Board (ISTQB). 2015. Sertifioitu Testaaja, Jatkotason sertifikaattisisältö - Testauspäällikkö. Tulostettu 26.3.2020. http://www.fistb.fi/sites/fistb/files/liitteet/Advanced%20Syllabus%202012%20-%20TM%20Final.pdf

International Software Testing Qualifications Board (ISTQB). 2014. Sertifioitu Testaaja, Jatkotason sertifikaattisisältö - Testausasiantuntija. Tulostettu 26.3.2020. http://www.fistb.fi/sites/fistb/files/liitteet/Advanced%20Syllabus%202012%20-%20TA%20FIN%2020140513%20FINAL.pdf

Moreira, I. 2019a. Test. Viitattu 14.11.2019. https://confluence.xpand-it.com/display/public/XRAY/Test

Moreira, I. 2019b. Pre-Condition. Viitattu 14.11.2019. https://confluence.xpand-it.com/display/public/XRAY/Pre-Condition

Moreira, I. 2019c. Test Set. Viitattu 14.11.2019. https://confluence.xpand-it.com/display/public/XRAY/Test+Set

Moreira, I. 2019d. Test Plan. Viitattu 15.11.2019. https://confluence.xpand-it.com/display/public/XRAY/Test+Plan

Ruokavirasto. 2019. Mikä on Ruokavirasto? Viitattu 7.11.2019. https://www.ruokavirasto.fi/tietoa-meista/mika-on-ruokavirasto/

Suomen Standardisoimisliitto SFS ry. 2012. SFS-ISO 21500. Viitattu 13.12.2019. https://sales.sfs.fi/fi/index/tuotteet/SFS/ISO/ID2/2/198650.html.stx

Wikipedia. 2019. Confluence. Viitattu 15.4.2020. https://fi.wikipedia.org/wiki/Confluence

Julkaisemattomat

Britschgi, J. 2019b. Bugien raportointi. Viitattu 1.4.2020. Ruokaviraston Prikka.

Britschgi, J. 2019a. Testaussuunnitelma. Viitattu 23.3.2020. Ruokaviraston Prikka.

Haapanen-Hurri, A. 2019. Laadunvarmistuksen tarkistuslista. Viitattu 31.3.2020. Ruokaviraston Prikka.

Kormano, M. 2019. JIRAn ja Confluencen käyttöpolitiikka. Viitattu 7.5.2020. Ruokaviraston Prikka.

Leppälä, S. 2019. Projektien hallintamalli. Viitattu 13.12.2019. Ruokaviraston Prikka.

Myllyluoma, M. 2019. Sähköpostikeskustelu 22.11.2019. Ruokavirasto. Helsinki.

Pajala, M. 2019. ELTE-tietojärjestelmän projektisuunnitelma. Viitattu 7.11.2019. Ruokaviraston Prikka.

Pajala, M. 2020. Projektipäällikön haastattelu 14.1.2020. Ruokavirasto. Helsinki.

Turpeinen, J. 2019. Ohjelmistotuotannon käsikirja. Viitattu 31.3.2020. Ruokaviraston Prikka.

Turunen, P. 2020. Automaatiotestaussuunnitelma. Viitattu 13.5.2020. Ruokaviraston Prikka.

Kuviot

Taulukot

Liitteet

Liite 1: ELTEn testauksen raportoinnista -lomake

ELTEn testauksen raportoinnista

Teen opinnäytetyötä Jiran/Xrayn käytöstä testauksen raportoinnissa ja haluaisin kuulla muiden projektilaisten toiveita/ehdotuksia/tarpeita sen suhteen. Kaikenlaiset ideat ja vinkit aiemmista projekteista otetaan vastaan. Kaikkiin kysymyksiin ei tarvitse vastata ja lyhyetkin vastaukset auttavat eteenpäin.
T. Virve

1. Oletko seurannut ELTEn testausta?

   - ◯ Kyllä
   - ◯ En

2. Jos vastasit kyllä, miten olet seurannut sitä?

   _____

3. Kiinnostaisiko saada tietoa testauksen tilanteesta?

   - ◯ Kyllä
   - ◯ Ei

4. Millaista tietoa kaipaisit testauksesta? Mikä asia kiinnostaisi, mitä tietoa haluaisit nähdä ja mistä/miten lukisit sitä mieluiten ja miten usein?

   _____

5. Koska testaus kiinnostaa varmasti eri tavalla eri tehtävissä olevia, niin jos haluat kertoa, voit laittaa tähän millaisessa roolissa toimit ELTEssä.

   _____

6. Tiedätkö, mikä on requirement status tai oletko kiinnittänyt siihen mitään huomiota Jirassa?

   _____

7. Käytätkö muuten Jirassa...

   - ☐ filttereitä
   - ☐ dashboardeja
   - ☐ valmiita raportteja

8. Selvennä hieman, miten käytät Jirassa filttereitä/dashboardeja/valmiita raportteja, mitä hyötyä niistä on?

   _____

9. Onko jotain muita vinkkejä Jiran/Xrayn käyttöön?

Liite 2: Vikaraportin luonti