# Testauksen toimintamallin kehitys ja soveltaminen Xray-testienhallintasovelluksessa

**Miika Santala**

2024 Laurea

**Laurea-ammattikorkeakoulu**

# Testauksen toimintamallin kehitys ja soveltaminen Xray-testien-hallintasovelluksessa

Miika Santala

Tietojenkäsittely

Opinnäytetyö

Marraskuu, 2024

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

Miika Santala
**Testauksen toimintamallin kehitys ja soveltaminen Xray-testienhallintasovelluksessa**
Vuosi 2024 Sivumäärä 49

Opinnäytetyön päämääränä oli kehittää toimeksiantajalle testauksen toimintamalli valmisohjelmistojen testaukseen sekä osoittaa mallin soveltaminen Xray-testienhallintasovelluksen avulla. Toimintamallista oli tavoitteena muodostaa systemaattinen sekä käytännönläheinen kuvaus testauksen toteuttamiselle toimeksiantajan olemassa olevien testauskäytänteiden tueksi.

Toimintamallin kehitys perustui toiminnalliseen kehitykseen, jossa mallin kehitystyö muodostui sen työstämisestä sekä jatkuvasta mallin käytännön soveltuvuuden arvioinnista todellisen testaustarpeen myötä. Mallin kehityksessä otettiin huomioon sen sovitettavuus ketteriin toimintatapoihin ja yhteensopivuus toimeksiantajan toimintaympäristöön osallistuvan havainnoinnin sekä toimeksiantajan edustajan puolistrukturoidun teemahaastattelun perusteella.

Kehitetyn toimintamallin ratkaisut perustuivat ohjelmistotestauksen yleispäteviin periaatteisiin sekä malleihin, joita opinnäytetyössä kartoitettiin perehtymällä aiheeseen liittyvään teoriaan ja käytänteisiin. Mallin kehitetyssä ratkaisussa huomioitiin toimeksiantajan määrittelemät vaatimukset ja tarpeet, kuten helppokäyttöisyys sekä lähestyttävyys, joiden myötä esitetyissä ratkaisuissa pyrittiin riittävään esitettävään yksinkertaisuuteen.

Opinnäytetyön lopputuloksena muodostui toimeksiantajan käyttöön manuaalisen testauksen toimintamalli sekä lopputulokseen liittyvä dokumentaatio. Valmis toimintamalli kykeni parantamaan edellytyksiä toimeksiantajan organisaatiossa tapahtuvan testauksen läpinäkyvyyden, yhdenmukaisuuden sekä rakenteisen toteutustavan lisäämiseksi. Toimintamallin tarkkuus vastasi tasoa, jossa se kykeni riittävän tarkasti määrittelemään, kuinka testaus tulisi toteuttaa kokonaisuutena huomioimalla toimeksiantajan moninaiset testauksen käyttötarpeet erilaisissa järjestelmissä.

Kehitettyyn toimintamalliin käytetyissä ratkaisuissa huomiotiin jatkokehityksen mahdollisuus esimerkiksi automaatiotestauksen suhteen. Valikoidut ratkaisut mahdollistavat toimintamallin jatkokehittämisen automaatiotestauksen suuntaan käyttäen apuna BDD- eli Behavior Driven Development -tyyppistä testausta etenkin Cucumber-testauskehystä hyödyntäen.

Asiasanat: Ohjelmistotestaus, testauksen toimintamalli, Xray

**Laurea University of Applied Sciences** **Abstract**

Degree Programme in Business Information Technology
Bachelor of Business Administration

Miika Santala

**Development of a Testing Framework and Its Application in the Xray Test Management Tool**

Year 2024 Pages 49

The objective of this thesis was to develop a testing framework for the commissioner to test off-the-shelf software and to display the applicability of the developed model using the Xray test management tool. The goal of the framework was to provide a systematic and practical description of testing to support the commissioner's existing testing practices.

The development of the framework followed the principles of functional development as the design process involved iterative refinement and continuous evaluation of the model's practical applicability based on actual testing needs. The compatibility with agile methodologies and model's alignment with the commissioner's operational environment were considered during the development, based on the information gathered through participatory observation and a semi-structured thematic interview with the commissioner's representative.

The solutions of the developed framework were derived from universal principles and models of software testing, identified through a theoretical review of relevant theories and practices. The commissioner's specific requirements and needs for the framework, such as ease of use and approachability, were taken into account with the aim of achieving sufficient simplicity in the proposed solutions.

The outcome of the thesis was a framework for manual testing along with documentation for the commissioner's use. The completed framework has the potential to enhance transparency, consistency, and structure of testing in the commissioner's organizational testing processes. The framework's level of detail was adequate to provide clear guidelines on how testing should be conducted as a whole while accommodating the commissioner's diverse testing needs across different kinds of systems.

The solutions applied in the completed framework also took into account future development possibilities, such as automated testing. The selected solutions enable further enhancement of the framework towards automated testing, utilizing Behavior-Driven Development (BDD) methods, specifically with the Cucumber testing framework.

Keywords: Software testing, testing framework, Xray

Sisällys

# 1 Johdanto

Ohjelmistoilla on suuri merkitys nykypäivän yhteiskunnan kaikilla osa-alueella ja tämän seurauksena ohjelmistoihin liittyy merkittäviä yhteiskunnallisia riskejä. ICT-alan modernia roolia kuvastaa alan tuottamien päästöjen määrä, jotka ovat vähintään samaa tasoa kuin lentoliikenteen tuottamat päästöt, ja kattavat noin kymmenen prosenttia koko maailman sähkönkulutuksesta (Hallamaa 2023). Ohjelmistoilla on lisäksi keskeinen rooli osana maailmantaloutta, josta kertoo muun muassa se, että vuonna 2024 maailman viisi markkina-arvoltaan suurinta yritystä olivat erilaisia ohjelmistojakin tuottavia teknologiayrityksiä: Nvidia, Apple, Microsoft, Amazon sekä Alphabet eli Googlen emoyhtiö (Companiesmarketcap 2024).

Ohjelmistojen merkittävä rooli sekä levinneisyys lisäävät tarvetta varmistua, että niiden käytössä ja toiminnassa esiintyy mahdollisimman vähän ongelmia. Suunnittelemattomasti käyttäytyvä ohjelmisto voi aiheuttaa merkittävää haittaa laaja-alaisesti organisaatioille ja yhteiskunnalle muun muassa mainehaittana, rahallisina menetyksinä sekä pahimmillaan jopa ihmisten terveyttä uhkaavina haittoina. Esimerkkinä vuoden 2024 heinäkuussa erään tietoturvayhtiön levittämä ohjelmistovirhe aiheutti maailmanlaajuisesti ongelmia muun muassa lentoliikenteessä sekä sairaaloissa (Kajander, Kavander & Hirvonen 2024).

Ohjelmistotestauksella on oleellinen rooli ohjelmistojen odottamattomasta toiminnasta johtuvien vikojen sekä haittojen ehkäisyssä. Testaus auttaa vähentämään ohjelmistovirheiden aiheuttamien haittojen riskiä arvioimalla ohjelmistojen laatua sekä etsimällä ohjelmistoihin liittyviä ohjelmistovirheitä (FISTB 2023). Ohjelmistotestauksen avulla organisaatiot voivat saavuttaa erinäisiä hyötyjä, kuten ohjelmistoihin liittyvä parempi riskienhallinta, toimintavarmuutta, kustannussäästöjä, yhteensopivuutta erinäisten vaatimusten kanssa sekä käyttömukavuutta (IEEE 2024).

Tämä opinnäytetyö keskittyy käsittelemään ohjelmistotestausta toimeksiantajalle kehittävän ratkaisun muodossa. Toimeksiantajan järjestelmien toiminnan kannalta ohjelmistotestauksella on kokonaisvaltaisesti kriittinen rooli ja toimeksiantaja on kyseisen roolin tunnistanut jo aiemmin. Tämän seurauksena toimeksiantajalla on intressejä huolehtia asianmukaisista toimintatavoista sekä menettelyistä testauksen suhteen, jotta riskejä ongelmien syntymiselle voidaan hallita ja vähentää.

Opinnäytetyötä määrittelee kaksiosainen tutkimuskysymys, joka ohjaa opinnäytetyötä ratkaisuunsa toiminnallisen kehitystyön sekä sen aikaansaannoksen käytännön soveltamisen kautta. Tutkimuskysymyksen ensimmäiseksi osaksi muotoutuu kysymys, minkälainen ohjelmistotestauksen ratkaisu voidaan kehittää vastaamaan toimeksiantajan testaustarpeita valmisohjelmistojen suhteen. Tutkimusongelmaan liittyvästä ensimmäisestä tutkimuskysymyksestä

saadaan johdettua tutkimuskysymyksen toinen osa, joka kysyy, miten kehitettyä ratkaisua voidaan soveltaa testienhallintasovelluksessa toimeksiantajan toimintaympäristössä.

Opinnäytetyö vastaa tutkimuskysymyksen ensimmäiseen osaan kehittämällä toimeksiantajalle ohjelmistotestaukseen toimintamallia, joka keskittyy erityisesti valmisohjelmistojen testauskäytänteiden parantamiseen. Opinnäytetyö pyrkii tuottamaan toimeksiantajalle systemaattisen sekä käytännönläheisen toimintamallin testausprosessin hallintaan toimeksiantajan toimintaympäristössä. Tutkimuskysymyksen jälkimmäistä osaa opinnäytetyö käsittelee soveltamalla kehitettyä mallia Xray-testienhallintasovelluksen käyttöön.

Toimeksiantajan toiveesta toimeksiantajan nimeäminen on jätetty pois opinnäytetyön raporttiosuudesta. Linjauksen myötä tässä raportissa jotakin organisaatiokohtaisia kuvauksia on jätetty kuvaamatta tai niitä on muutettu geneerisempään muotoon yksityiskohtien peittämiseksi.

# 2 Ohjelmistotestaus

Ohjelmistotestaus voidaan nähdä töinä tai toimenpiteinä, joiden avulla pyritään varmistamaan testattavan kohteen toimivuus suunnitellusti osoittamalla kohteelle määriteltyjen vaatimusten ja testauskattavuuden täyttyminen. Vaatimusten tai kattavuuden epäonnistunut saavuttaminen voi tarkoittaa virhettä tai puutetta testattavassa kohteessa. Tällöin testauksen tuloksista voidaan päätellä, ettei testattava kohde toimi odotetulla tavalla jostain syystä. Ohjelmistotestaus toimintana pyrkiikin vähentämään todennäköisyyttä sille, että testattava kohde toimisi odottamattomasti. (FISTB 2023, 15; Kasurinen 2014, luku 1; Spillner & Linz 2021, 7-8.)

Ohjelmistotestaus on mahdollista mieltää sekä tarkistavaksi että tutkivaksi toiminnaksi, sillä ohjelmistotestauksen avulla kyetään varmistamaan, että tunnetut syy-seuraussuhteet etenevät testattavassa kohteessa halutulla tavalla testauksen myötä tehtyjen tarkistuksien avulla. Ohjelmistotestauksen tutkiva aspekti puolestaan pyrkii löytämään testattavasta kohteesta tuntemattomia syy-seuraussuhteita, joilla voi olla vaikutusta sen toimintaan esimerkiksi aikaan saamalla odottamattomia lopputuloksia tai tapahtumakulkuja. (Kakkonen & Rytkönen 2023, 16-17.)

Ohjelmistotestauksen avulla testausta suorittava taho, testaaja, pyrkii laajentamaan ymmärrystään ja tietouttaan testattavasta kohteesta. Testauksen avulla hankittu ymmärrys ja tieto voivat auttaa arvioimaan sekä kohentamaan testattavan kohteen laatua esimerkiksi toimintavarmuuden, tietoturvallisuuden tai muun laatua kuvaavan vaatimuksen näkökulmasta. Testauksen myötä saatua informaatiota kyetään lisäksi hyödyntämään testattavan kohteen, sen käyttöympäristön tai käyttäjien toiminnan kehittämiseen, jotta testattava kohde saavuttaisi

sille asetetut vaatimukset erinäisillä testauksen tasoilla. (Rytkönen & Kakkonen 2023, 16; Spillner & Linz 2021, 7.)

Ohjelmistotestaus lähtee oletuksesta, että olemassa ei ole virheetöntä ohjelmistoa tai järjestelmää. Ohjelmistot ja järjestelmät ovat monimutkaisia kokonaisuuksia, joiden käyttötavat sekä -tarpeet vaihtelevat jokaisen käyttäjän, käyttöympäristön tai -tavan mukaisesti. Tämä johtaa siihen, että testattavan kohteen kaikkien mahdollisten toimintatapojen tunnistaminen on käytännössä mahdotonta, sillä kaiken kattavaan testaamiseen vaadittavat resurssit olisivat suhteettoman suuret. Esimerkiksi testauksen lähestymistavan ja resurssoinnin kannalta tällä oletuksella on merkitystä, jotta kyetään tunnistamaan todelliset keinot, jotka vaikuttavat testauksella tavoiteltaviin päämääriin. Tunnistamalla riittävät toteutettavissa olevat testauksen keinot suhteessa testattavaan kohteeseen, voidaan testauksen rajauksesta johtuvia riskejä muun muassa odottamattomille seurauksille pienentää sekä hallita. (Rytkönen & Kakkonen 2023, 17-18; ASTQB 2024.)

Ohjelmistotestauksen suhteen tärkeää tunnistaa muitakin perusolettamuksia testauksen mahdollisuuksista, kuten että ISTQB:n testauksen perusperiaatteita mukaillen ohjelmistotestauksen avulla voidaan osoittaa, että testattavassa kohteessa on virheitä, muttei sitä, että virheitä ei olisi (ASTQB 2024). Tämän periaatteen ymmärtämisellä on merkitystä, kun arvioidaan esimerkiksi testattavaan kohteeseen liittyviä hyväksyttäviä riskejä sekä niiden rajaamisesta testauksen ulkopuolelle. Lisäksi periaatteella on merkitystä, kun tavoitellaan testauksen päämäärää vähentää riskiä testattavan kohteen suunnittelemattomalle toiminnalle.

Testaukseen käytössä olevien resurssien rajallisuutta taklataan ohjelmistotestauksessa testauksen suunnittelussa lähestymällä testausta tietyistä lähtökohdista. Valittujen lähtökohtien perusteella testattavasta kohteesta tunnistetaan testaukseen kohteet, kuten yksittäiset komponentit tai integraatiot, riittävän testauskattavuuden saavuttamiseksi. Tunnistettujen kohteiden perusteella laaditaan testitapaukset, jotka kuvaavat tai testaavat riittävissä määrin kaikkia mahdollisia testattavia kohteita. Tällöin on arvioitava riski mahdollisuudelle, että jotakin oleellista jää testaamatta sekä sille, milloin kyseinen riski on hyväksyttävä. (Rytkönen & Kakkonen 2023, 17-18.)

Ohjelmistotestaus liittyy oleellisesti organisaatioissa tapahtuvaan laadunvarmistukseen, sillä testaus ja laadunvarmistus ovat vahvasti toisiinsa liittyviä toimintoja. Ne ovat toinen toistaan tukevia keinoja parantaa testattavan kohteen laatua pienentämällä riskiä ohjelmiston virheille ja puutteille. Laadunvarmistus toimintana pyrkii testausta laaja-alaisemmin varmistamaan, että organisaatiossa noudatetaan määriteltyjä prosesseja, joiden avulla tavoitellaan laadullisten vaatimusten täyttymistä. Testaus puolestaan tukee laadunvarmistusta, sillä se parantaa osaltaan todennäköisyyttä, että nämä vaatimukset saavutetaan. (Spillner & Linz 2021, 26.)

Ohjelmistotestaukseen, ja etenkin testaustyöhön, liittyy oleellisesti eri järjestöt, jotka tarjoavat muun muassa sertifikaatteja sekä teoriatietoa ohjelmistotestaukseen liittyen. Yksi merkittävä alan järjestö on ISTQB eli International Software Testing Qualifications Board. Kyseessä on kansainvälinen järjestö, joka vastaa tietyistä kansainvälisesti laajasti käytössä olevista ohjelmistotestaukseen liittyvien sertifikaattijärjestelmien kehityksestä sekä ylläpidosta. ISTQB:n sertifikaatit ovat hyvin tunnettuja sekä arvostettuja, ja niiden sisältö voidaan siten liittää ohjelmistotestauksen parhaisiin käytänteisiin. (ISTQB 2024.)

## 2.1 Testaustasot

Ohjelmistotestaus voidaan mieltää osaksi ohjelmistokehitystyötä, vaikka testausta voidaan toteuttaa irrallaan kehitystyöstä esimerkiksi valmisohjelmistojen suhteen. Ohjelmistotestaus voidaan jakaa neljäksi testaustasoksi, joista jokainen vastaa testattavan ohjelmistosuunnittelun arkkitehtuurin tasoja testauksen näkökulmasta: yksikkö-, integraatio-, järjestelmä- ja hyväksymistestaus. Ohjelmistoarkkitehtuurin tasot puolestaan koostuvat ohjelmiston osista, joihin kohdistuu kehitystyön mukaiset tietyt vaatimukset. Tästä muodostuu testaustasojen liitos ohjelmistokehityksen arkkitehtuuriin ja siihen liitettyihin ohjelmistolle määriteltyihin vaatimuksiin. (Spillner & Linz 2021, 58; Umar 2020.)

Yksikkötestaus kohdistuu ohjelmistoarkkitehtuurin alimmalle tasolle eli ohjelmiston kehitykseen. Yksikkötestaus testaa ohjelmiston yksittäisen komponentin, olion tai funktion toimintaa tai rakennetta, ja se toteutetaan usein jo ohjelmointivaiheessa ohjelmoijan toimesta. Yksikkötestaus pyrkii varmistamaan, että ohjelmiston yksittäinen osa toimii itsenäisesti irrallaan muista ohjelmiston osista ja sitä kautta varmistamaan, että osa on myöhemmin käytettävissä laajemmassa kokonaisuudessa. (Kasurinen 2014, luku 3; Spillner & Linz 2021, 58-59.)

Integraatiotestaus on ylemmän tason testausta yksikkötestaukseen nähden. Integrointi tarkoittaa, että esimerkiksi ohjelmiston yksittäiset osat yhdistetään kokonaisuudeksi, joka voi muodostua kahdesta tai useammasta ohjelmiston osasta. Nimensä mukaisesti integraatiotestaus testaa osien toimivuutta keskenään integraation myötä. Integraatiotestauksen tavoitteena on varmistua, että ohjelmiston osat toimivat yhdessä halutulla tavalla. (Kasurinen 2014, luku 3; Spillner & Linz 2021, 66.)

Järjestelmätestaus on integraatiotestausta korkeampi testaustaso, jonka pyrkimyksenä on osoittaa, että osista koottu ohjelmisto täyttää kokonaisuutena sille asetetut määritykset testiympäristössä. Järjestelmätestaus ottaakin testauksessa huomioon testattavan kohteen lisäksi testiympäristön. Järjestelmätestauksessa testiympäristö on ohjelmiston lopullisesta sijoitusympäristöstä, kuten tuotantoympäristöstä, erillinen. Testiympäristön tulisi tarpeen mukaan kuitenkin jäljitellä lopullista ympäristöä mahdollisimman tarkasti, jotta testaus etenisi kohti hyväksymistestausta. (Kasurinen 2014, luku 3; Spillner & Linz 2021, 74-75.)

Hyväksymistestaus on testaustasoista korkein ja toteutetaan siinä vaiheessa, kun kaikki alemman tason testaukset on hyväksytysti suoritettu. Hyväksymistestaus pyrkii osoittamaan, että ohjelmisto vastaa sille asetettuja vaatimuksia sekä määrityksiä sen lopullisessa käyttötarkoituksessaan ja ympäristössään. Yleensä hyväksymistestauksen hyväksyy sellainen taho organisaatiosta, joka vastaa esimerkiksi liiketoimintaan liittyvistä päätöksistä, kun alemmilla testauksen tasoilla testauksen onnistumisen arviointi kuuluu enemmän esimerkiksi testaajille. (Kasurinen 2014, luku 3; Spillner & Linz 2021, 77.)

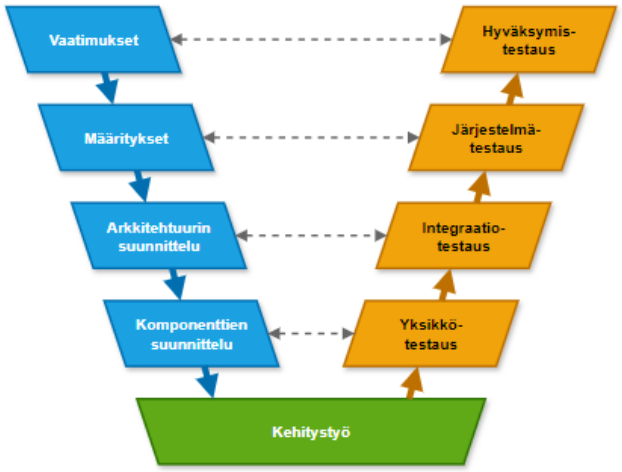
## 2.2 Testausmallit

Ohjelmistotestauksen kuvaamiseen on olemassa erilaisia testausmalleja, jotka lähestyvät testausta eri tavoin. Testausmalleja ovat ainakin perinteinen vesiputousmalli, V-malli sekä erinäiset iteratiiviset ja inkrementaaliset mallit (Spillner & Linz 2021, 50-54). Testausmallit lähestyvät testauksen organisointia, aikataulusta sekä toteutusta toisistaan poikkeavin tavoin. Tämä on otettava testauksessa huomioon, jotta testausmallin yhteensovittaminen organisaatiossa muutoin käytössä oleviin toimintamalleihin onnistuu.

Perinteisessä vesiputousmallissa testaus suoritetaan osana ohjelmistokehitystyötä varsinaisen ohjelmiston kehitystyön jälkeen. Mallin mukaan ohjelmistotestaus ei varsinaisesti tue itse kehitystyötä, vaan toimii eräänlaisena varmistuksena, että ohjelmisto toimii hyväksytysti ennen kuin ohjelmiston kehitystyö määritellään päättyneeksi. (Spillner & Linz 2021, 50.)

Iteratiiviset ja inkrementaaliset mallit ovat moderneja kehityksen malleja, joissa testaus tapahtuu sykleittäin kasvavina tai kehittyvinä kokonaisuuksina. Jokaisen syklin lopussa ohjelmistoa ja sen testausta arvioidaan vaatimuksiin sekä kehitystä ohjaaviin toiveisiin nähden. Tämän arvioinnin pohjalta seuraavassa syklissä kehitystä ja testausta ohjataan enemmän haluttuun suuntaan. (Spillner & Linz 2021, 54.)

Testauksen V-malli on laajasti yleispätevä malli toteuttaa testausta. V-malli noudattaa aiemmin kuvattuja testauksen tasoja osoittaen samalla testaustasojen riippuvuuden ohjelmiston arkkitehtuurista. V-malli korostaa, että kehitystyö ja testaus ovat lopputuloksen kannalta yhtä merkityksellisiä osuuksia ohjelmiston kehitystä. Lisäksi mallin vahvuutena on, että se validoi sekä varmistaa testattavan kohteen vaatimuksiinsa nähden kullakin testauksen tasolla. (Spillner & Linz 2021, 51-54.)




Kuvio 1: Testauksen V-malli (mukaillen Kasurinen 2014, luku 3)

## 2.3 Testausprosessi

Ohjelmistotestaus kattaa prosessina joukon erinäisiä toimenpiteitä, joista esimerkiksi testitapauksien suorittaminen ja testaustulosten raportointi ovat vain osa. Testaukseen kuuluu oleellisesti lisäksi testauksen suunnittelu, testisuoritusten ja -tulosten analysointi, testitapauksien laatiminen sekä erilaiset dokumentoinnin työt. (Spillner & Linz 2021, 8.)

Laadukkaan testauksen toteuttamiseen tarvitaan useampi testaukseen liittyvä toiminto, jotka sidotaan osaksi testauksen aikataulutusta. Spillnerin ja Linzin (2021, 27-28) mukaan ISTQB määrittelee testausprosessille seuraavat toiminnot:

- Suunnittelu
    - Monitorointi sekä hallinta
        - Analysointi
        - Suunnittelu
        - Implementointi
        - Suoritus
        - Valmistuminen

Testausprosessi alkaa suunnittelusta, joka kattaa koko prosessin alusta loppuun kaikkine prosessin muine toimintoineen. Sen sijaan testauksen monitoroinnin sekä hallinnan toiminto kulkee muiden toimintojen ohessa samanaikaisesti, lukuun ottamatta suunnittelun toimintoa. Muutoin testaus etenee prosessina edellä kuvatun luettelon mukaisesti testattavan kohteen analysoinnista testauksen valmistumiseen. (Spillner & Linz 2021, 28.)

## 2.4 Testauksen lähestymistavat

Testausta voidaan lähestyä erilaisista lähestymistavoista, jotka määrittävät, millaisista näkökulmista esimerkiksi testauksen suunnittelua, testitapauksia sekä -suorituksia toteutetaan. Lähestymistapoja on useita, eivätkä ne välttämättä ole toinen toisiaan poissulkevia. Toisin sanoen yhden testausprojektin aikana testausta voidaan lähestyä useiden lähestymistapojen avulla. (Rytkönen & Kakkonen 2023, 164.)

Testauksen lähestymistapoja ovat muun muassa:

- **Testitapauksiin perustuva testaus**, joka on perinteinen tapa lähestyä testausta. Lähestymistavassa testaukseen laaditaan testitapauksia, joiden kuvaus kertoo, mitä tulee testata ja mitä tulee odottaa tulokseksi. Lähestymistavan mukaan testaaja suorittaa testitapaukset manuaalitestauksena, joten lähestymistavan suhteen on tärkeää dokumentoida testitapaukset riittävällä tarkkuudella, jotta voidaan varmistaa testattavan kohteen toiminta halutulla tavalla. Lisäksi testitapauksia on oltava riittävästi, jotta testaus vastaa vaadittua testauskattavuutta. (Rytkönen & Kakkonen 2023, 164-165.)
- **Riskipohjainen testaus** korostaa testauksessa keskittymistä riskiarvioon, jolla kartoitetaan testattavaan kohteeseen liittyviä riskejä sekä näistä aiheutuvia haittoja. Lähestymistapa priorisoi korkean riskin toimintojen tai ominaisuuksien testitapaukset ensimmäisenä ja määrittää ne ehdottomasti suoritettaviksi. Riskittömiin tai vähäriskisiin testitapauksiin puolestaan kiinnitetään vähemmän huomiota eikä niitä välttämättä testata lainkaan. Lähestymistavan kannalta on tärkeää tunnistaa testattavan kohteen toiminnot ja ominaisuudet laajasti sekä riittävän tarkasti, jotta testaus kykenee keskittymään oleellisiin testauksiin testauskattavuuteen nähden. (Kasurinen 2014, luku 8; Rytkönen & Kakkonen 2023, 170-171.)

Testauksen muita lähestymistapoja ovat muun muassa regressiotestaus, joka testaa aiemmin toimiviksi todettuja toiminnallisuuksia uudelleen esimerkiksi versiovaihdon yhteydessä, sekä tutkiva testaus, joka pyrkii tutkivan näkökulman kautta maksimoimaan testattavasta kohteesta löytyvät viat. (Rytkönen & Kakkonen 2023, 166, 173.)

## 2.5 Testaustavat

Ohjelmistotestausta voidaan suorittaa useilla eri testaustavoilla sekä -menetelmillä, joista jokainen lähestyy testausta hieman eri näkökulmista ja tavoittein (IEEE 2024). Mahdollisia testaustapoja on lukuisia, joista esimerkkeinä toimivat aiemmin testaustasoina mainitut testaukset, kuten integraatio- tai hyväksymistestaus. Muita testaustapoja ovat esimerkiksi stressitestaus, suorituskykytestaus ja funktionaalinen testaus. (IBM 2024.)

Testaustapoja kuvaava testauksen laatikkomalli lähestyy testausta sen mukaan, mitä testattavan kohteen sisäisestä toiminnasta tiedetään tai halutaan testata. Laatikkomallin mukaisia testaustapoja ovat lasilaatikko- sekä mustalaatikkotestaukset ja näiden välimuotona harmaalaatikkotestaus. Testaustapoina lasilaatikko- ja mustalaatikkotestauksia voidaan pitää ylätason testaustapoina, joiden mukaisesti tarkemmin kohdistuvia testaustapoja suoritetaan. (Anwar & Kar 2019; Kasurinen 2014, luku 4.)

Lasilaatikkotestauksessa testaaja tuntee testattavan kohteen sisäiset ominaisuudet, jolloin testaaja voi tarkastella sekä testata kohteen sisäisiä rakenteita sekä toiminnallisuuksia. Testaustasoista lasilaatikkotestausta voidaan suorittaa yksikkö-, integraatio- sekä järjestelmätestauksena, mutta usein testaus on yksikkötestausta, joka keskittyy testaamaan etenkin testattavan kohteen sisäisiä toimintoja. Harmaalaatikkotestaus puolestaan tuntee osittain testattavan kohteen rakenteesta ja toiminnasta, jolloin testausta voidaan tehdä esimerkiksi käyttäjän näkökulmasta siten, että kohdistetaan testattavat käyttäjän toimet tiettyihin ohjelmiston sisäisiin rakenteisiin. Tällä tavoin testausta voidaan kohdentaa mustalaatikkotestausta tarkemmin. (Anwar & Kar 2019.)

Mustalaatikkotestaus keskittyy testaamaan testattavan kohteen toimintaa ilman tietoa sen sisäisestä toiminnasta tai rakenteesta, keskittyen etenkin ohjelmiston käyttäjän näkökulmaan. Mustalaatikkotestaus edellyttää, että testattava kohde on siinä määrin toimiva, että sitä voidaan testata loppukäyttäjänä. Käytännössä testaustavassa testattavalle kohteelle annetaan jokin syöte, jonka perusteella tarkastellaan kohteen tuottamaan tulosta, jolloin kohteen sisäinen toiminta pysyy testaajalta tuntemattomana. (Anwar & Kar 2019; Kasurinen 2014, luku 4.)


Kuvio 2: Hahmotelma mustalaatikkotestauksesta (mukaillen Kasurinen 2014, luku 3)

Eräs tapa lähestyä testausta on syötteiden ja odotettujen tulosten kautta, jolloin testaus voidaan määritellä positiiviseksi ja negatiiviseksi testaukseksi. Positiivisessa testauksessa tavoitteena on varmistaa, että testattava kohde tuottaa halutun sekä hyväksyttävän tuloksen annetulla syötteellä. Tämä vahvistaa, että testattava kohde toimii odotetulla tavalla käyttäjän näkökulmasta. Negatiivisessa testauksessa puolestaan arvioidaan, miten testattava kohde reagoi epätavallisiin tai virheellisiin syötteisiin, sekä miten kohde reagoi tällaisessa tilanteessa esimerkiksi toimintakykynsä puolesta. Negatiivisen testauksen myötä voidaan saada arvokasta tietoa testattavan kohteen sisäisten rakenteiden toimimattomuudesta käyttäjän toimien kautta. (BairesDev 2024; Spillner & Linz 2021, 63-64.)

Testauksen laadukkuuden sekä kattavuuden kannalta testaustavan valinnalla on merkitystä, sillä tietyillä testaustavoilla voidaan sekä testata että arvioida spesifejä vaatimuksia. Täten testauksen vaatimusten sekä päämäärien saavuttamiseksi voi olla tarpeen hyödyntää testauksessa useita erilaisia testaustapoja, sillä tällöin testaaja voi kyetä paremmin varmistamaan, että kaikki testaukseen liittyvät vaatimukset täyttyvät tarpeen mukaisesti. (Rytkönen & Kakkonen 2023, 133.)

## 2.6 Testauksen suorittaminen

Testaus voidaan suorittaa joko manuaalisesti tai automatisoidusti. Manuaalitestauksessa testaaja suorittaa testit itse testitapauksiin perustuen, kun automaatiotestauksessa testaus määritellään koodikielellä, kuten pythonilla tai javalla, erilliselle testaustyökalulle, joka puolestaan suorittaa testauksen. Testaustavan näkökulmasta jotkin testausmenetelmät ja lähestymistavat ovat helpommin sekä laajemmin automatisoitavissa kuin toiset, mikä voi vaikuttaa testaustavan valintaan. Esimerkiksi testitapaukset, jotka ovat säännönmukaisia, toistettavissa kerta toisensa jälkeen tai ovat nopeammin suoritettavissa skriptien avulla, soveltuvat hyvin automaatiotestaukseen. (AnAr 2023; IEEE 2024; SmartBear 2024.)

Manuaali- ja automaatiotestaukseen pätevät testauksen kannalta samat päämäärät. Testauksen suoritusta varten tarvittavat tiedot, työkalut, materiaalit, dokumentaatiot ja niin edelleen täytyy olla saatavissa testisuorituksen jälkeenkin, jotta tarpeen mukaan testaus voidaan uusia. Lisäksi testaussuoritus itsessään on kyettävä dokumentoimaan riittävissä määrin, jotta sen perusteella voidaan tehdä erinäisiä johtopäätöksiä testattavasta kohteesta sekä testauksen sujumisesta. (Spillner & Linz 2021, 37-38.)

Manuaalitestauksen etuina automaatiotestaukseen nähden on toteutustavan yksinkertaisuus ja helppous, sekä manuaalitestauksen käyttäjälähtöisyys. Lisäksi manuaalitestauksen suhteen testaajan tai testitapausten laatijan ei ole tarve perehtyä automaatiotestauksen työkaluihin ja toteuttaa sekä ylläpitää testausta niiden avulla. (Kasurinen 2014, luku 4).

## 2.7 Testauksenhallinta ja testauksenhallinta työkalut

Testauksenhallinta on kokonaisvaltaista testaukseen liittyvää toimintaa, joka kattaa testauksen koko elinkaaren. Testauksenhallinta käsittää testauksen organisoinnin, suunnittelun, resurssien hallinnan, testauksen ohjauksen sekä testausroolien määrittelyn. Lisäksi testauksen seuranta, hallinta ja raportointi ovat osa testauksenhallintaa. Tavoitteena testauksenhallinnalla on tehostaa testauksen resurssien käyttöä sekä varmistaa, että testaus tukee testattavan kohteen laadunvarmistusta sekä muita tavoitteita mahdollisimman tehokkaasti. (IEEE 2024.)

Testauksenhallinnassa käytetään apuna testauksen hallinnan työkaluja, joiden tarkoituksena on mahdollistaa testauksen seuranta ja ohjaus kokonaisvaltaisesti koko testausprosessin ajan. Työkalun on kyettävä esimerkiksi tarjoamaan tietoa testitapauksien etenemisestä sekä testitapauksien tilasta, mahdollisten testauksessa havaittujen virheiden määristä sekä näiden vakavuudesta. (Kasurinen 2014, luku 5.)

Xray on testauksenhallinnan työkalu, jonka toimii yhdessä Atlassian Jira-projektienhallintasovelluksen kanssa. Xray ei ole osa Atlassianin sovellustarjontaa, vaan erikseen hankittava lisäosa Jiraan. Nykyisin Xray on osa Idera-yrityksen tuoteperhettä. (Idera 2021.)

Xray:n avulla testaus voidaan kokonaisuudessaan toteuttaa tiketteinä Xray:n toiminnallisuuksien tukemana Jiran projekteihin sidottuna. Xray:n käyttöönoton myötä Jiraan saadaan käytettäväksi uusia tikettityyppejä testaukseen liittyen, sekä muita testauksen hallinnan ominaisuuksia, kuten testitapauksiin liittyviä hakemistorakenteita, raportointia sekä testauksen seurantaa. (Atlassian 2024a.)

Xray:n käyttöönotto tuo Jiraan seuraavat uudet tikettityypit:

- Testisuunnitelma (Test Plan)
- Testijoukko (Test Set)
- Testitapaus (Test)
- Ennakkoehto (Pre-Condition)
- Testisuoritus (Test Execution)

Xray:n pohjalla toimiva Atlassian Jira on selaimella käytettävä projektinhallintasovellus. Jiran avulla organisaatio voi seurata ja hallita erityyppisiä tehtäviä sekä projekteja erityisesti liittyen ohjelmistokehitykseen. Jira on kokonaisvaltainen työkalu edellä mainittujen tarpeiden hallintaan. (Jira 2024.)

# 3 Kehitys- ja tutkimusmenetelmät

Opinnäytetyö on toteutukseltaan toiminnallinen opinnäytetyö, joka tähtää toimeksiantajan toiminnan kehittämiseen tämän toimintatapoja muokkaamalla. Opinnäytetyö kokonaisuutena käsittää raportin sekä toiminnallisen toteutuksen eli toimeksiantajalle tuotetun ratkaisun. Opinnäytetyön raportti käsittelee teoreettisen osuuden, jossa muodostetaan opinnäytetyön aihetta käsittelevä tietoperusta, ja johon opinnäytetyön toiminnallinen osuus pohjautuu. Raporttiosuus sisältää lisäksi kuvauksen toiminnallisen osuuden tuotoksesta. Toiminnallinen osa opinnäytetyöstä on kehitystyö, joka tähtää konkreettisen ratkaisun luomiseen toimeksiantajan organisaation käyttöön. (Haaga-Helia AMK 2022; HAMK 2024.)

Opinnäytetyö lähestyy kehitystyötä iteratiivisen toimintatavan kautta. Iteratiivisuudella tarkoitetaan projektin keston rajaamista kokonaisuutta pienempiin työvaiheisiin, jotka pyrkivät toteuttamaan ennalta sovittuja toiminnallisuuksia tai ominaisuuksia kokonaisuudesta. Yleensä iteraatioita toteutetaan yhdessä jonkin ketterän menetelmän avulla. Ketteriä menetelmiä ovat muun muassa Agile, Lean ja DevOps, joista jokainen lähestyy tietyn projektin hallintaa omista näkökulmistaan sekä vahvuuksistaan. (Erlund, Aalto-Setälä, Hynönen, Lilja, Lindfors, Nevasalo, Salminen & Turunen 2022; Atlassian 2024g.)

Opinnäytetyössä toteutettu iteratiivisuus ei noudata tiukasti tiettyä ketterää menetelmää, vaan keskittyy vaiheittain etenevään ratkaisun kehittämiseen. Ratkaisun kehityskohteet perustuvat alkukartoituksen jälkeen ratkaisun sovellettavuuden arviointiin ja siihen liittyviin huomioihin, sekä toimeksiantajan kanssa käytäviin palautekeskusteluihin.

Opinnäytetyö pyrkii selvittämään tutkimuskysymykseensä vastauksena ratkaisun pääasiassa kahta laadullista tutkimusmenetelmää sekä näihin pohjautuvaa laadullista analyysia hyödyntäen. Toimeksiantajan näkemyksiä, tarpeita sekä vaatimuksia selvitetään kehitettävää ratkaisua varten haastattelututkimuksen avulla. Osallistuvaa havainnoinnin avulla saatuja tuloksia hyödynnetään kehitettävän ratkaisun sitomisessa toimeksiantajan organisaation toimintatapoihin, olemassa oleviin käytänteisiin sekä luomaan käsitys lähtökohdista, joiden päälle ratkaisu kehitetään.

Valitut tutkimusmenetelmät mahdollistavat opinnäytetyön lähtökohtien ja tavoitteiden rajaamisen tutkimuskysymykseen nähden opinnäytetyön kannalta järkeväksi kokonaisuudeksi. Tutkimusmenetelmien avulla saatujen tulosten perusteella opinnäytetyön tutkimuskysymykseen kehitettävässä vastauksessa eli kehitettävässä ratkaisussa voidaan keskittyä erityisesti manuaalitestauksen menetelmiin, jotka soveltuvat valmisohjelmistojen ja niiden käyttöönottojen testaukseen käyttäjälähtöisesti.

## 3.1 Haastattelututkimus

Haastattelututkimus on aktiivinen, kanssakäymiseen perustuva tapa kerätä tietoa kahden tai useamman osallistujan välisen vuorovaikutuksen perusteella. Arkikeskustelusta poiketen, tutkimushaastattelu tähtää ainakin jonkin päämäärään, jossa haastattelun osapuolilla on tietyt roolit. Yleisesti ottaen haastattelija toimii osapuolena, joka pyrkii saamaan tietoa asiantuntijalta eli haastateltavalta. (Hyvärinen, Nikander & Ruusuvuori 2017, luku 1.)

Haastattelututkimuksessa esitetään tavanomaisesti kolmen tyyppisiä kysymyksiä. Ensimmäisenä tutkimusta varten määritellään vähintään yksi tutkimuskysymys, joka ohjaa haastattelututkimuksen kulkua. Toista kysymystyyppiä edustavat itse haastattelukysymykset, joista varsinainen haastattelu koostuu. Kolmas kysymystyyppi on aineistokohtaiset kysymykset, jotka

pyrkivät luomaan vastauksia haastattelututkimuksen perusteella tutkimuskysymyksiin. (Hyvärinen ym. 2017, luku 1.)

Haastattelututkimus voidaan jakaa kolmeen päätyyppiin: strukturoitu, puolistrukturoitu sekä avoin haastattelu. Näistä ensimmäinen haastattelutyyppi noudattaa tarkimmin rajattua kaavaa kysymyksien ja haastattelun rakenteen suhteen. Usein strukturoidut haastattelut ovat malliltaan kyselytyyppinen lomakehaastattelu, jossa haasteltava vastaa valmiiksi muodostettuihin kysymyksiin vastausvaihtoehtojen perusteella. Avoin eli strukturoimaton haastattelu puolestaan on muodoltaan mahdollisimman vapaamuotoinen. Avoin haastattelu ei kuitenkaan voi olla täysin vapaamuotoinen, sillä oletuksena on, että haastattelijalla on jokin aihe, jota hän haluaa haastattelulla tutkia. Puolistrukturoitu haastattelu on kahden edellä mainitun haastattelutyypin niin sanottu välimuoto, jossa haastattelu noudattaa tiettyä rakennetta sekä kysymysten määrittelyitä, mutta sallii vapaamuotoisen vastauksen. (Saaranen-Kauppinen & Puusniekka 2006a; Hyvärinen ym. 2017, luku 1.)

Teemahaastattelu on muodoltaan lähimpänä puolistrukturoitua haastattelututkimusta, jossa perusideana on, että haastattelija määrittelee kysymysten sijaan tiettyjä tutkittavia teemoja, joihin liittyen vuorovaikutusta ohjataan. Teemahaastattelu antaa paljon tilaa ja vapautta haastateltavan puheelle, joka edellyttää haastattelijalta riittävää perehtyneisyyttä käsiteltäviin teemoihin, jotta keskustelua voidaan ohjata tarkoituksenmukaisesti. (Saaranen-Kauppinen & Puusniekka 2006b; Hyvärinen ym. 2017, luku 1.)

## 3.2 Osallistuva havainnointi

Osallistuva havainnointi voi olla tyypiltään aktiivista tai passiivista. Aktiivinen osallistuva havainnointi tarkoittaa, että tutkija osallistuu ja on vuorovaikutuksessa tutkimansa kohteen tapahtumiin, ja siten vaikuttaa tutkimuskohteeseensa. Passiivisessa osallistuvassa havainnoinnissa tutkija seuraa tutkimaansa kohdetta olematta sen kanssa vuorovaikutuksessa, jolloin tutkija ei vaikuta tutkimuskohteeseen suoraan. (Saaranen-Kauppinen & Puusniekka 2006c.)

Osallistuvasta havainnoinnista saatava hyöty perustuu tutkittavasta kohteesta saatuihin havaintoihin ja niiden arviointiin. Havainto itsessään ei ole tietoa, vaan eräänlainen johtolanka, joita tutkimalla sekä tarkkailemalla tutkija ymmärtää tutkittavaan kohteeseen liittyviä merkityksiä. Näiden merkitysten pohjalta tutkija voi oppia ymmärtämään tutkittavaa asiaa ja muodostamaan siihen liittyvää uutta tietoa. Havaintojen muodostamiseen liittyy oleellisesti tutkijan tuntemus tutkittavasta kohteesta, jotta tämä voi muodostaa kohteesta asianmukaisia havaintoja. Virheelliset tai esimerkiksi väärään näkökulmaan sidotut havainnot voivat muodostaa tutkijalle virheellisiä käsityksiä tutkittavasta kohteesta ja siten vaikuttaa tutkimuksen tuloksiin. (Valli 2018, luku 1.)

### 3.3 Laadullinen analyysi

Laadulliselle tutkimukselle on tyypillistä, että tutkimusaineistoa kertyy monessa eri vaiheessa, useista eri havaintolähteistä sekä koko tutkimusprosessin ajan. Laadullista analyysia varten tutkija kerää aktiivisesti valittuja tutkimusmetodeja hyödyntäen havaintoja tutkittavasta kohteesta, jotka analyysissa puretaan osiin. Puretuista osista tutkija pyrkii muodostamaan tulkinnan kokonaisuudesta, jonka perusteella tutkija voi kyetä muodostamaan uskottavan ja asianmukaisen kuvan tutkittavasta kohteesta. Oleellista on näkökulma, kuten tutkimusongelman muotoilu, josta tutkija havainnointeja lähestyy. Kyseisen näkökulman tulisi perustua teoreettiseen viitekehykseen, jotta analyysissa korostuu tieteellisyys. (Puusa & Juuti 2020, luku 9.)

Laadullisen analyysin tavoitteena on luoda tutkittavasta kohteesta perusteltu tulkinta sekä saada aikaan johtopäätöksiä. Laadulliselle analyysille tunnusomaista on tutkimusmetodien aikaansaamien havaintojen luokittelu esimerkiksi teemojen mukaisesti, josta voi olla hyötyä johtopäätöksien teossa sekä perustelussa. Lisäksi luokittelun avulla tutkija voi kyetä löytämään tutkimusaineistosta tiettyjä säännönmukaisuuksia, joiden tulkinnasta voidaan saada oivalluksia tutkittavasta kohteesta. Johtopäätöksien sekä tulkintojen luomiseen vaikuttaa lisäksi analyysin lähestymistapa, joka voi pohjautua teoriaan, jolloin teoreettinen tieto ohjaa analyysien tekemistä. Aineistolähtöinen analyysi puolestaan lähtee muodostamaan käsitystä tutkittavasta kohteesta tutkimusaineiston pohjalta. (Puusa & Juuti 2020, luku 9.)

## 4 Testauksen toimintamalli

Opinnäytetyön toiminnallinen osuus käsitti testauksen toimintamallin kehityksen toimeksiantajan hyödynnettäväksi. Toimintamallin kehityksen lähtökohdat toimeksiantajan näkökulmasta pohjautuivat haastattelututkimuksen ja osallistuvan havainnoinnin myötä saatuihin johtopäätöksiin sekä tuloksiin, joita arviointiin opinnäytetyötä varten laadullisen analyysin kautta.

Toimintamallin kehitystä ohjasi toimeksiantajan halu kehittää tietyn organisatorisen ryhmän testauskäytänteitä ja niiden avulla tukea valmisohjelmistojen käyttöönottoa sekä järjestelmien laadunvarmistusta. Kehitettävässä ratkaisussa huomiotiin työlle asetettujen tavoitteiden sekä tarpeiden lisäksi toimeksiantajan organisaatiossa entuudestaan käytössä olevia testauskäytänteitä ja toimeksiantajan monimuotoinen toimintaympäristö. Lopputuloksen tarkoituksena oli aikaan saada toimeksiantajan organisaation käyttöön soveltuva ratkaisu, jonka käyttöönotto olisi mahdollisimman vaivaton.

## 4.1 Lähtökohdat toimeksiantajan haastatteluun perustuen

Opinnäytetyöhön kuului toimeksiantajan edustajalle teetetty teemahaastattelu. Haastattelun tarkoituksena oli kartoittaa toimeksiantajan näkemyksiä sekä toiveita testauskäytänteiden kehittämiselle toimeksiantajalle. Haastattelun teemat käsittelivät testauksen strategista lähestymistä, organisaation valmiuksia testaukselle, testauksella saavutettavia toivottuja hyötyjä sekä toteutettavan työn vaatimuksia niin sanotusti työn tilaajan näkökulmasta.

Haastattelussa nousi esiin toimeksiantajan halu kehittää testauksen nykytilaa määrämuotoisen sekä suunnitelmallisen testauksen suhteen. Haastattelun perusteella kävi ilmi, että toimeksiantajan organisaatiossa testausta on tehty entuudestaan laajasti ja kattavasti, mutta toimeksiantajan tahtotilana oli saada testausta laaja-alaisemmin läpinäkyvämmäksi sekä testauksen dokumentointia rakenteisemmaksi.

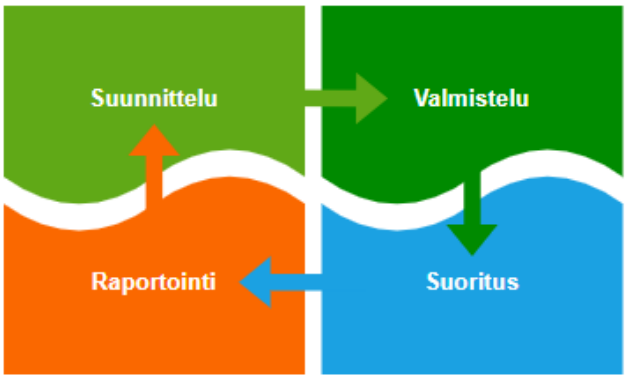
Opinnäytetyön toivottiin kehittävän toimeksiantajalle testaukseen ohjeistettu toimintamalli, jonka haluttiin olevan helppokäyttöinen sekä käyttäjälähtöinen, jotta mallin mahdollinen käyttöönotto olisi toimeksiantajan organisaatiossa mahdollisimman sujuvaa. Lisäksi mallin toivottiin kykenevän lisäämään ei-testaajien suorittaman testauksen dokumentointia sekä kannustavan laajemmin dokumentoituun testaukseen.

Toimeksiantajan näkemyksen mukaisesti, kehitettävässä toimintamallissa tavoiteltavia arvoja olivat laadunvarmistus, luotettavuus sekä läpinäkyvyys. Toimeksiantajan vision mukaan edellä mainittuja arvoja tulisi pyrkiä saavuttamaan huomioimalla, että toimintamalli ohjaisi korostamaan testauksen suunnitelmallisuutta, tasalaatuisuutta sekä määrämuotoisuutta.

Toimeksiantaja nosti haastattelussa esille, että toimintamallin toimivuus käytännössä tulisi osoittaa opinnäytetyössä jollakin tasolla, jotta kehitetty malli vastaisi sille asetettuja tavoitteita sekä vaatimuksia. Haastattelun myötä päädyttiin siihen, että sovellettavuuden havainnointi tulisi tehdä jollakin testienhallintasovelluksella, jonka käyttöä voisi testata ja hyödyntää mahdollisesti laaja-alaisemminkin.

## 4.2 Toimintamallin kuvaus

Opinnäytetyön toimintamallissa testausprosessia käsitellään neljänä vaiheena, jotka kuvataan vaihe vaiheelta etenevänä, iteratiivisena prosessina. Vaiheittain etenevän kuvauksen tarkoituksena on havainnoida testausprosessi yleisellä tasolla sekä luoda prosessille helposti hahmotettava etenemistapa. Iteratiivisuus sitoo testausprosessin paremmin nykyaikaisiin ketteriin työskentelymenetelmiin sekä mahdollistaa toimintamallin implementoinnin monenlaisissa käyttötarkoituksissa.




Kuvio 3: Kuvaus nelivaiheisesta testausprosessista

Toimintamallissa testausprosessi on esitetty seuraavina vaiheina:

- **Suunnitteluvaiheena**, joka käsittää testauksen suunnittelun sekä määrittelyn. Suunnitteluvaiheen tarkoituksena on suunnitella testaus kokonaisuudessaan siten, että se kattaa testattavan kohteen testauksen esimerkiksi määritysten ja testauksen tavoitteiden kannalta riittävissä määrin. Suunnitteluvaiheessa käsiteltyjä asioita kuvataan testisuunnitelmassa.
- **Valmisteluvaiheena**, jossa testisuunnitelman pohjalta laaditaan testitapaukset sekä niihin liittyvät ennakkoehdot sekä testijoukot. Lisäksi valmisteluvaiheeseen kuuluu testiympäristön sekä -aineiston valmistelu, jotta nämä ovat hyödynnettävissä testitapausten suorituksessa. Valmistelut suhteutetaan käytettävissä oleviin resursseihin suunnitelman pohjalta, jotta varmistetaan testisuunnitelman toteutuminen.
- **Suoritusvaiheena**, jolloin testitapaukset suoritetaan ja testauksesta saatuja tuloksia arvioidaan testauksen suunnitelmaan nähden. Arvioinnin perusteella voidaan muun muassa päättää tarve testitapauksien uudelleensuoritukselle tai joidenkin testitapauksien uudelleen suunnittelulle sekä valmistelulle.
- **Raportointivaiheena**, jossa luodaan testauksesta eri tasoisia raportteja, jotka voivat kohdistua esimerkiksi testauksen kokonaisvaltaiseen etenemiseen sekä kattavuuteen.

Toimintamallin kehityksen pohjana on käytetty Xray:n tapaa kuvata testausprosessi (Xray 2023). Xray:n tarjoamaa kuvausta prosessista on yksinkertaistettu sekä sovellettu iteratiiviseen suuntaan opinnäytetyössä kehitettyä toimintamallia varten.

### 4.2.1 Suunnitteluvaihe

Opinnäytetyönä kehitetty toimintamalli kuvaa testauksen suunnittelussa huomioitavia seikkoja, tarkoituksenaan ohjata testaajaa muodostamaan käsitys, ja sen pohjalta kuvaus, kuinka testausta aiotaan toteuttaa, sekä mitä testaus kattaa verrattuna testattavan kohteen vaatimuksiin. Lisäksi suunnitelman laatimisessa ohjataan ottamaan kantaa olemassa oleviin

resursseihin sekä toteutuksen aikatauluun. Suunnitteluvaiheen päätyttyä, testaajalla tulisi mallin mukaisesti olla ymmärrys testauksen etenemisestä.

Malli tarjoaa testaajalle ohjeistuksen testauksen suunnitteluvaiheeseen, mutta se ei ota tarkasti kantaa kaikkiin suunnittelussa huomioitaviin osa-alueisiin, kuten resursseihin, testiaineistoihin tai -ympäristöihin. Tämä johtuu siitä, että mallin on tarkoitus olla yleisluontoinen ohjeistava malli sekä käytettävissä erinäisissä toimintaympäristössä ja tilanteissa, joissa tarpeet testauksen suhteen voivat vaihdella huomattavasti testattavasta kohteesta riippuen.

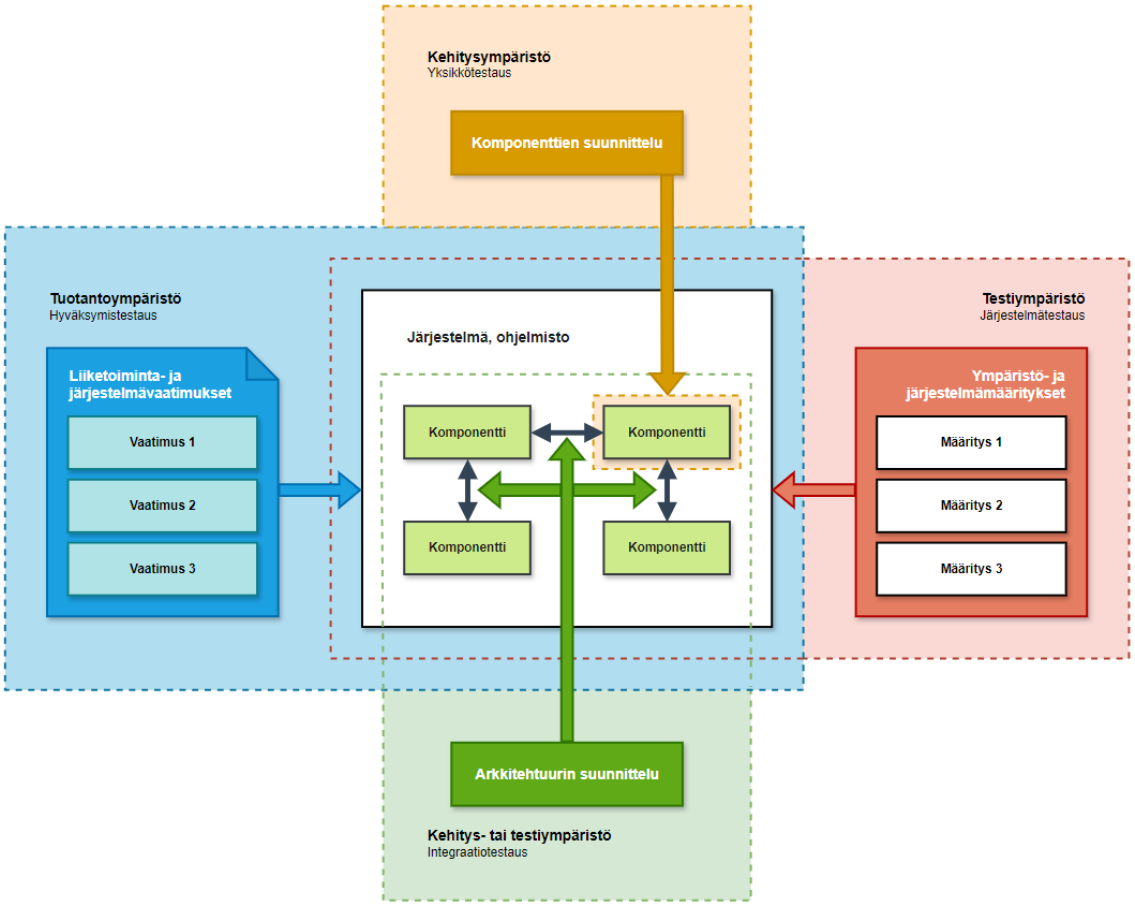
Opinnäytetyön toimintamallissa testattava kohde huomioidaan pääasiassa valmisohjelmistona tai järjestelmäkokonaisuutena, jonka osana jokin valmisohjelmisto toimii. Malli ei kuitenkaan rajaa pois sen käyttömahdollisuuden ulottamista muunlaisten kohteiden testaamiseen. Oleellista mallin mukaisessa testauksessa on testattavan kohteen ja sen osa-alueiden riittävän tarkkarajainen tunnistaminen sekä määrittely.

Toimintamallin mukaan testattavasta kohteesta on kyettävä tunnistamaan osa-alueet, toiminnallisuudet ja käyttötarkoitukset, jotta testaus voidaan kohdistaa tarpeenmukaisesti sekä kattavasti. Malli ohjaa tunnistamaan testattavan kohteen esimerkiksi dokumentaatioon perustuen. Testattavan kohteen määrittelyssä malli ohjaa huomioimaan vaatimukset, joita testaukselle asetetaan.

Toimintamallissa testattavaa kohdetta käsitellään mustalaatikkotestauksen näkökulmasta, joka on luonteva valinta valmisohjelmistojen testauksessa. Tällä on merkitystä etenkin testauksen suunnittelun kannalta, jotta testauksessa pyritään keskittymään oikeanlaisten testitapausten laatimiseen sekä asianmukaiseen testaustasoon.

Toimintamalli johdattelee testaajaa hahmottamaan testattavan kohteen kokonaisuutena, johon kohdistuu testaustasosta riippuen erilaisia lähestymistapoja sekä vaatimuksia. Mallissa testattava kohde koostuu yhdestä tai useammasta komponentista sekä eri testausympäristöistä. Eri testiympäristöt voivat kohdistaa suoritettavan testauksen testattavalle kohteelle sekä kohteen eri osa-alueille testiympäristön sekä testattavan kohteen erillisten tarpeiden mukaisesti. Malli kuvaa kunkin testaustason erityispiirteitä ja huomioon otettavia asioita yleisellä tasolla keskittyen etenkin järjestelmä- ja integraatiotestaukseen.

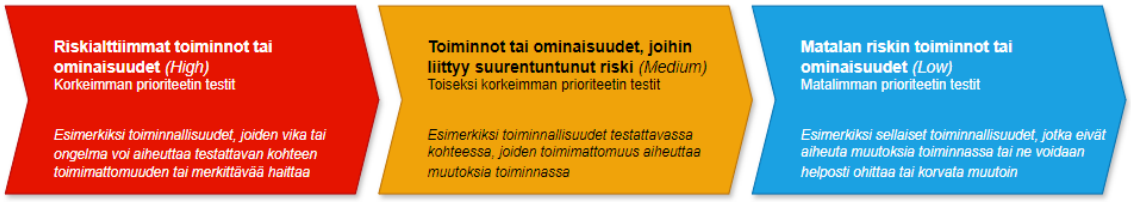
Testaustasojen huomiointi toimintamallissa auttaa testaajaa hahmottamaan selkeämmin testauksen kohdistumisen vaatimuksiin nähden. Tämän ymmärtäminen auttaa testaajaa sovittamaan testauksen priorisointia asianmukaisesti vaatimuksia sekä määrityksiä vasten.




Kuvio 4: Hahmotelma toimintamallin mukaisesta testaustason kohdistumisesta

Toimintamalli lähestyy testausta testitapauksiin perustuvana testauksena riskilähtöisen testauksen kautta. Riskilähtöinen lähestyminen kohdistaa testaukseen käytettävät rajalliset resurssit siten, että testauksessa pyritään keskittymään niiden tehokkaaseen käyttöön. Käytännössä tämä tarkoittaa, että testaus keskittyy painottuen suurimpien ongelmien poistamiseen sekä pääominaisuuksien testaamiseen.

Kehitetyn toimintamallin mukaisessa lähestymistavassa testaukselle luodaan kolme prioriteettiluokkaa testattavan kohteen vaatimuksiin ja testauksen tavoitteisiin perustuvien riskien pohjalta. Priorisointia arvioidaan siten, että testattavasta kohteesta tunnistetaan osa-alueet tai ominaisuudet, joissa virheet aiheuttavat suurimman riskin ohjelmiston toimivuuden, tietoturvallisuuden tai jonkin muun vaatimuksen näkökulmasta.



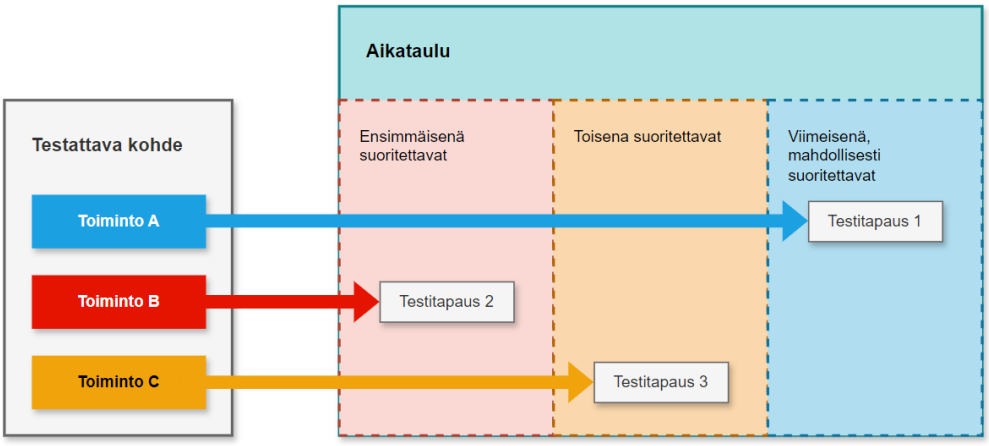

Kuvio 5: Toimintamallin mukaiset kolme prioriteettiluokkaa

Korkeimman tason prioriteettiluokkaan määritellään kuuluvaksi sellaiset testattavat ominaisuudet tai toiminnallisuudet, joissa virheet voivat aiheuttaa merkittävän haitan testattavan kohteen toiminnalle tai jopa estää toiminnan kokonaan.

Keskitason riskiksi mallin mukaan määritellään esimerkiksi sellaiset toiminnot, joihin liittyvät häiriöt sekä virheet voivat vaikuttaa testattavan kohteen toimintaan, tai joiden toiminta riippuu korkeimman riskitason toiminnoista.

Malli käyttää kolmantena luokkana matalan tason riskiluokkaa, johon lasketaan kuuluvaksi sellaiset ominaisuudet sekä toiminnot, joiden ei tunnisteta vaikuttavan testattavan kohteen toimintaan, tai joihin kohdistuva riski on helposti ohitettavissa. Matalan tason riskiluokkaan voidaan määritellä lisäksi sellaiset testitapaukset, jotka kuuluisivat korkeammalle tasolle, mutta niihin liittyvän riskin arvioidaan syystä tai toisesta, kuten olosuhteista tai kohteen käyttökohteesta riippuen, olevan vähäinen.

Toimintamallin mukaisesti prioriteettiluokat sidotaan resursseihin siten, että korkeimmalle prioriteettiluokalle varataan eniten resursseja sekä niihin liittyvät testaukset ajoitetaan suoritettavaksi pääsääntöisesti ensimmäisenä. Keskitason prioriteettiluokkaan kuuluvat testaukset aikataulutetaan ensimmäisen luokan jälkeen suoritettaviksi ja matalimman luokan testitapaukset tarpeen sekä resurssien mukaisesti.




Kuvio 6: Hahmotelma mallin mukaisten prioriteettiluokkien aikataulutuksesta

### 4.2.2 Valmisteluvaihe

Toimintamallin mukaisesti testaukseen valmistautuminen käsittää testitapauksien laatimisen sekä suunnitteluvaiheeseen perustuvien valmisteluiden tekemisen muun muassa testiympäristön sekä -aineiston suhteen. Malli ohjaa tunnistamaan testitapauksiin liittyvät mahdolliset ennakkoehdot, testitapauksen etenemisen sekä tavoitellut lopputulokset eli testitapauksien hyväksymiskriteerit. Näiden suhteen apua voidaan hakea testattavalle kohteelle asetetuista vaatimuksista, asiantuntijalta sekä suunnitteluvaiheen perusteella.

Testausta valmisteltaessa mallin perusteella, testiympäristö on otettava testauksen tarpeiden mukaisesti huomioon ympäristön asettamien rajoitteiden sekä toiminnallisuuksien puitteissa sekä mallissa esitellyn lähestymistavan näkökulmasta. Malli ohjaa testaajaa huomioimaan testitapauksien laatimisessa ympäristökohtaiset erityispiirteet sekä ympäristökohtaiset tavoitellut lopputulokset.

Toimintamallin valmisteluvaihe pyrkii huomioimaan testijoukoissa ja niiden märittelyssä testattavan kohteen osa-alueet, testaustyypit sekä erityisesti testiympäristöt. Jos testitapauksia laaditaan useaan eri testiympäristöön, malli ohjaa luomaan näille testitapauksille ympäristökohtaiset testijoukot. Lähestymistapa on hyödyllinen muun muassa silloin, kun testisuoritukset tehdään ympäristökohtaisesti, sillä testijoukot voidaan liittää testisuorituksiin kokonaisuuksina.

Opinnäytetyössä toimintamallin testitapausten laatimista on lähestytty Given-When-Then (GWT) -mallin kautta. GWT-mallin mukaiset testitapaukset noudattavat Behavior Driven Development (BDD) -tyyppistä eli käyttäytymisvetoisuuteen perustuvaa kehitystapaa. Malli antaa puolistrukturoidun ja siten johdonmukaisen tavan kirjoittaa testitapauksia, jotka seuraavat käyttäjälähtöisesti seurattavaa syy-seuraussuhdetta. (Cucumber 2024; RobotFramework 2024.)

GWT-mallin avulla voidaan rakentaa käyttäytymistä kuvaavia testitapauksia käyttäen kolmea lausetta. Näistä ensimmäinen lause käsittää testitapauksen oletus- tai lähtötilanteen (Given) ennen varsinaista testin suoritusta. Seuraavassa lauseessa (When) testitapaukselle määritellään jokin tilanne tai tapahtuma, joka laukaisee testitapauksen suorituksen. Toista lausetta voidaan lähestyä lauseenalulla "Kun". Kolmas lause (Then) määrittelee tilanteen, jossa testitapaus saavuttaa oletetun lopputuloksensa eli hyväksymiskriteerin. GWT-malli mahdollistaa lisäksi edellä mainittujen kolmen lauseen täydentämisen käyttämällä täydennyslausetta (And), jolla voidaan asettaa tarkentavia kuvauksia kullekin päälauseelle. (Cucumber 2024; RobotFramework 2024.)

Taulukko 1: Testattavana ominaisuutena tiedoston poisto järjestelmästä

| **Given-lause** | **When-lause** | **Then-lause** |
|---|---|---|
| Käyttäjä on kirjautunut järjestelmään, *ja* käyttäjällä on tarvittavat käyttöoikeudet tiedoston käsittelyyn | *Kun* käyttäjä yrittää poistaa tiedoston | *Niin* järjestelmä poistaa tiedoston, *ja* antaa käyttäjälle ilmoituksen poiston onnistumisesta |

GWT-mallin avulla testitapauksista voidaan kirjoittaa suhteellisen yleisluontoinen kuvaus ja lähestyä siten rakenteellista testausta helpottaen testitapauksien suunnittelua sekä hahmottamista. Kyseinen näkökulma palvelee opinnäytetyön kehitystyön lähtökohtien vaatimuksia, kuten käytännönläheistä testauksen suunnittelua sekä toteutusta.

Opinnäytetyön ei ole tarkoitus ohjata käyttämään GWT-mallia suoraan kuvaamaan kirjoitettavien testitapauksien rakennetta. Lisäksi opinnäytetyön esittämän mallin ei ole tarkoituksena ohjeistaa tarkalleen GWT-tyyppisten testitapausten kirjoittamista GWT-mallia tukeviin testienhallintasovelluksiin.

Opinnäytetyön malli ohjaa testaajaa laatimaan testitapauksia sekä positiivisina että negatiivisina testauksina, jotta testaus olisi kattavaa sekä toisi esille mahdollisia virheitä tai ongelmia. Lisäksi toimintamalli käsittelee vain manuaalisen testauksen luomista eikä mallissa käsitellä automaatiotestauksen laatimista.

### 4.2.3 Suoritusvaihe

Toimintamallin mukainen suoritusvaihe käsittää testitapausten varsinaisen suorittamisen. Tähän vaiheeseen kuuluu lisäksi testisuoritusten etenemisen seuranta testausprosessin näkökulmasta, mikä käytännössä tarkoittaa testitapausten suoritusjärjestyksen asettamista testaajan määrittelemien prioriteettien mukaisesti. Suoritusvaiheen keskiössä on varmistaa, että kriittisimmät testitapaukset suoritetaan ensin, jotta saadaan nopeasti tietoa merkittävistä vioista ja mahdollisista ongelmista sekä mahdollisista entuudestaan tunnistamattomista riskeistä.

Mallin mukainen testaus edellyttää, että testaaja arvioi testisuorituksia sekä niiden laatua. Tämä tarkoittaa, että testaaja tarkastelee testien onnistumista sekä tunnistaa mahdolliset virheet ja arvioi, missä määrin testitapaukset täyttävät määritellyt odotukset, sillä tarpeen vaatiessa testitapauksien suoritus pitää uusia.

Testitapausten uudelleen suoritus on oleellinen osa suoritusvaihetta. Uudelleen suoritus voi tulla kyseeseen esimerkiksi, kun testin suorituksessa ilmenee ongelmia tai testitulokset eivät vastaa odotettuja lopputuloksia. Testaajan tekemät arvioinnit ohjaavat sitä, onko yksittäisiä

testitapauksia tarpeen korjata tai jatkokehittää, tai onko tarvetta palata suunnittelu- ja valmisteluvaiheeseen testauksen uudelleen määrittelemiseksi.

Toimintamallin mukainen suoritusvaihe etenee neljässä päävaiheessa:

1. **Testitapausten suorituksen suunnittelu riskilähtöisesti,** jossa vaihe keskittyy testisuoritustikettien suunnitteluun ja laatimiseen valmisteluvaiheen priorisointiin perustuen
2. **Testitapausten suoritus,** jolloin varsinainen testauksen suorittaminen tapahtuu prioriteettiluokan mukaisessa järjestyksessä
3. **Testitapauksien tulosten arviointi**, jossa testaaja arvioi kunkin suorituksen tulokset sen suhteen, miten hyvin testitapaus täyttää sille asetetut tavoitteet
4. **Testitapauksien mahdollinen uudelleensuoritus tai uudelleen laatiminen**, silloin kun on tarve uusia testitapauksia. Tarvittaessa testitapaukset suoritetaan uudelleen testattavan kohteen muutoksien jälkeen tai toisinaan testitapauksen muokkauksen jälkeen

Lisäksi toimintamalli ohjaa ottamaan huomioon testiympäristöt testitapauksien suorittamisen aikana. Eri testiympäristöissä suoritetuilla testeillä voi olla merkittäviä eroja, joten on tärkeää, että mahdolliset ympäristökohtaiset tekijät huomioidaan huolellisesti. Jos testiympäristöjä on useita, testien toteutuksessa tulee varmistaa, että jokaisen ympäristön erityispiirteet otetaan huomioon testitulosten tarkkuuden ja luotettavuuden varmistamiseksi.

Toimintamallin mukainen testisuorituksen uusinta käsittää toimenpiteet, joilla olemassa oleva testitapaus suoritetaan uudelleen sellaisenaan. Uusinnan tavoitteena on todentaa, että testattavaan kohteeseen tehdyt muutokset eivät ole vaikuttaneet testin lopputulokseen, tai että tehdyt korjaukset ovat poistaneet aiemmin havaitun vian, joka johti alkuperäisen testin epäonnistumiseen. Malli ohjaa testaajaa siihen, että testisuorituksen uusinta tehdään muuttamatta itse testitapausta. Tällöin uusitun suorituksen tulokset ovat suoraan vertailukelpoisia alkuperäisen testisuorituksen tulosten kanssa.

Testisuoritus voidaan joutua uusimaan silloin, kun testitapaus on ollut alun perin virheellisesti määritelty tai kattavuudeltaan riittämätön. Tällöin ei varsinaisesti uusita testisuoritusta, vaan palataan testitapauksen valmisteluvaiheeseen. Kun testitapaus on päivitetty, tehdään uusi testisuoritus päivitetystä testitapauksesta. Jos itse testitapausta joudutaan muuttamaan uusinnan yhteydessä, tulee varmistaa, että sen kattavuus vastaa alkuperäistä testitapausta. Näin voidaan vertailemalla todeta, että testitapauksen uudelleen määrittely korjaa alkuperäisen testitapauksen uusinnan syyn.

Toimintamalli käsittelee testisuorituksen uusintaa lisäksi muista syistä kuin alkuperäisen testisuorituksen epäonnistumisesta johtuen. Esimerkiksi regressiotestauksessa voidaan haluta

suorittaa aiemmin onnistuneet testitapaukset uudelleen varmistaakseen, että uudistetun järjestelmän osat toimivat edelleen odotetusti.

### 4.2.4 Raportointivaihe

Toimintamallin mukainen raportointivaihe toimii seuraajana suoritusvaiheelle. Sen pääasiallisena tavoitteena on tuoda esiin testauksesta saadut havainnot sekä nostaa esille havaitut keskeiset huomiot testauksesta. Raportointivaiheessa testaaja voi tarkastella esimerkiksi testauksen etenemistä ja valmistumista, joko testisuorituksiin tai testisuunnitelman toteutumiseen perustuen.

Malli käsittelee raportointia erityisesti testauksen kokonaisvalmistumisen ja löydettyjen virheiden kautta. Tavoitteena on ohjata testaajaa tunnistamaan raportointia käyttäen, miten hyvin testaus on noudattanut testisuunnitelmaa. Samalla malli ohjaa varmistamaan, että testausresurssit on käytetty optimaalisesti, jotta testauksen vaatimukset ja kattavuus saavutetaan prioriteettien sekä aikataulun mukaisesti. Erityisesti epäonnistuneet testitapaukset nostetaan esille, koska ne tarjoavat arvokasta tietoa testattavan kohteen toimivuudesta, vaatimusten täyttymisestä sekä testauksen laadusta.

Raportointivaiheen kautta pyritään lisäämään testauksen läpinäkyvyyttä ja vahvistamaan testattavan kohteen laadunvarmistusta. Malli ohjaa tunnistamaan ja raportoimaan testauksessa ilmenneet pullonkaulat, ongelmakohdat ja mahdolliset puutteet, jotta iteratiivinen testausprosessi voidaan toteuttaa tehokkaasti. Lisäksi raportointi tarjoaa keinoja muodostaa kokonaiskuva testauksen edistymisestä, erityisesti ketterissä toimintamalleissa, jolloin jokaisen iteraation jälkeen voidaan arvioida testauksen tilanne ja edistyminen kokonaisuudessaan sekä parantaa testausta seuraavaan iteraatioon.

## 5 Toimintamallin toteuttaminen Xray-testienhallintasovelluksessa

Opinnäytetyössä kehitetyn toimintamallin soveltuvuutta arvioitiin toimeksiantajan työlle asetettuun vaatimukseen nähden, jonka perusteella toimintamallia tuli kyetä hyödyntää jonkin testienhallintasovelluksen avulla. Tämän vaatimuksen täyttämiseksi opinnäytetyössä toimintamallia sovellettiin käytäntöön Xray-testienhallintasovellusta käyttäen hyödyntämällä erillistä testaustarvetta tapausesimerkkinä toimintamallin mukaisesti toteutusta testauksesta. Xray:n valitseminen testienhallintasovellukseksi perustuu muun muassa Atlassian Jiran suosioon projektinhallintasovelluksena ja Xray:n suoraan integroitavuuteen Jiran kanssa (Atlassian 2022).

Hyödyntämällä mallin soveltamista Xray:n kanssa, kehitettävää toimintamallia kyettiin arvioimaan sekä kehittämään konkreettisen käyttökokemuksen perusteella. Näiden kokemusten perusteella toimintamallista oli mahdollista saada parempi käsitys sen käytännön toimivuuden

näkökulmasta, sekä mallin kyvystä integroitua toimeksiantajan organisaation testausprosesseihin.

Opinnäytetyön raporttiosuus käsittää edellä kuvatusta kokonaisuudesta näkemyksen siitä, miten kehitettyä mallia voidaan soveltaa Xray:ssä. Varsinaisen testauksen tapausesimerkin käsittely on rajattu raportin ulkopuolelle.

## 5.1 Suunnitteluvaiheen toteutus

Testisuunnitelma (Test Plan) on Xray:n tikettityyppi, joka muodostaa testauksesta suunnitelmallisen kokonaisuuden. Testisuunnitelmaan voidaan liittää osaksi muut testaukseen käytettävät Xray-tiketit, ja sen avulla voidaan seurata suunnitelmaan kuuluvien testitapauksien statuksia, testisuorituksien määrää sekä suoritusten etenemistä. Lisäksi testisuunnitelman avulla voidaan parantaa testauksen kokonaisuuden hallittavuutta. Testisuunnitelmatiketti on rakenteisen testauksen toteutuksen kannalta oleellinen komponentti. (Xray 2024a.)

Toimintamallin mukainen suunnitteluvaihe toteutetaan sekä määritellään Xray:ssa testisuunnitelmatikettinä. Malli ohjaa lisäksi testaajaa tekemään suunnittelluvaiheen pohjalta erillisen dokumentaation, johon testisuunnitelma tarkemmin kirjataan. Opinnäytetyön kannalta esitetyn toimintamallin soveltaminen Xray:n avulla keskittyy pääasiassa ohjaamaan testaajaa testisuunnitelmatiketin sekä vaatimusmäärittelyiden luonnissa. Malliin liitetty ohjeistus käsittelee näiden vaiheiden toteuttamisen kirjallisena ohjeena, joissa kuvataan täytettävät tiketin kentät.

Malli ohjaa täyttämään testisuunnitelmatiketille tarpeelliset tiedot, joiden pohjalta tiketin avulla voidaan osoittaa testauksen suunnitelmallisuus. Tiketin luonnissa otetaan huomioon suunnitteluvaiheessa läpikäydyt seikat liittyen muun muassa testattavaan kohteeseen, testauksen tasoon, testauksen lähestymistapaan sekä resurssointiin. Tarkoituksena on saada testisuunnitelmatiketin määrityksistä riittävän kattavat sekä kuvaavat, jotta tiketti itsessään voisi toimia mahdollisimman riittävänä kuvauksena testauksen suunnitelmasta sekä sen toteuttamisesta.

Osittain suunnitteluvaiheen toteutus edellyttää, että testaaja tai testauksen määrittelijä on perehtynyt toimintamallin kuvaukseen kokonaisuudessaan ennen tikettien luomista. Esimerkiksi testaajalla täytyy olla ymmärrys Jiran komponenttien sekä merkintöjen käytöstä, sillä näitä hyödynnetään testisuunnitelmatiketin määrittelyissä.

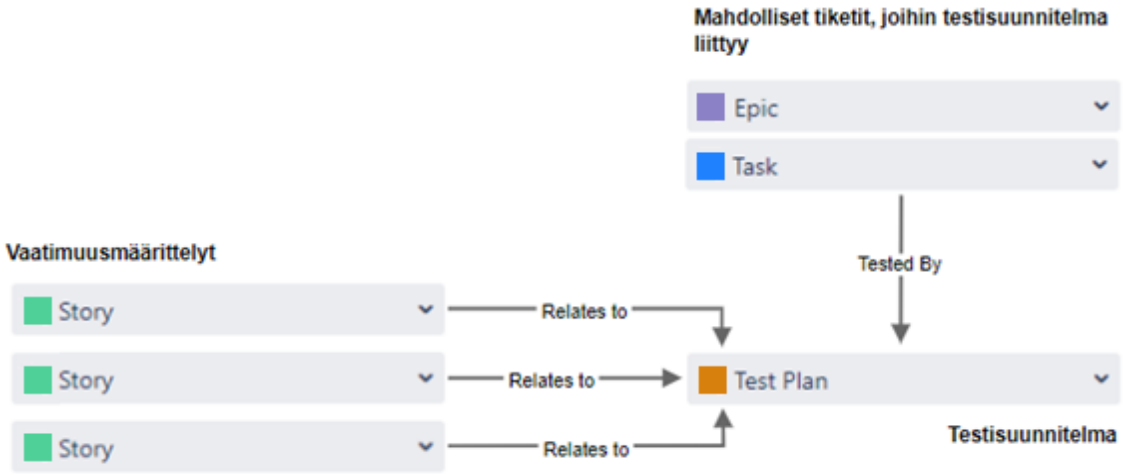

Kuvio 7: Esimerkki testisuunnitelmatiketin määrittelyistä

Testisuunnitelman kattavuuden ja vaatimusten kannalta on oleellista luoda Jiraan Story- eli tarinatikettejä, jotka kuvaavat testauksen vaatimuksia suhteessa riskilähtöiseen testauksen lähestymistapaan käyttäjän näkökulmasta (Atlassian 2024e). Malli ohjaa testaajaa luomaan kolme erillistä tarinatikettiä, joista kukin vastaa yhtä mallissa esitettyä prioriteettitasoa sekä niihin liittyviä testattavan kohteen vaatimuksia. Näiden tarinatikettien avulla voidaan tarkentaa testisuunnitelman sisältöä sekä varmistaa, että siihen liitettävät testitapaukset kattavat kaikki oleelliset riskit ja vaatimukset. Tarinatiketit näkyvät muun muassa testisuunnitelmatiketillä vaatimuksina (Requirements), mikä luo selkeän yhteyden testitapauksien ja ennakkoehtojen sekä testauksen tavoitteiden välille.

Toimintamalli käsittelee tarinatikettien suhteen etenkin prioriteettiarvon määrittelyä tiketille kuvaavan tiketin nimen lisäksi. Tarinatikettien suhteen oleellista on kuvata prioriteettia kuvaava vaatimustaso eli esimerkiksi korkeimman tason tekstitapauksiin liitettävät vaatimukset. Toimintamalli käyttää yksinkertaisena lähestymisenä vain kolmea tikettiä kuvaamaan tunnistettuja riskejä, joten vaatimukset on määriteltävä riittävän yleisluontoisesti. Myöhemmin mallin valmisteluvaiheessa näitä vaatimusmäärittelyitä voidaan tarkentaa testijoukkojen avulla.

Toimintamallin toteutuksen kannalta Xray:n avulla, on tärkeää määritellä Jira- ja Xray-tikettien väliset yhteenliittymät eli linkitykset. Xray osaa luoda omien toiminnallisuuksien puitteissa käyttäjän määrittelemät linkitykset, mutta Jiran ja Xray:n tikettien välille testaajan on erikseen määriteltävä linkittymiset ja linkityksen tyyppi. Mallin ohjeistus ottaa tähän kantaa sekä ohjaa testaajaa luomaan oikeat linkitykset tikettien välille.

Testisuunnitelma voi lopulta muodostua laajaksi ja monimutkaiseksi erilaisten tikettien kokonaisuudeksi. Esimerkiksi kehitystöistä voidaan Jirassa luoda erillisiä Task- eli tehtävätikettejä, jotka kuvaavat tehtävää työtä (Atlassian 2024b). Laajemmat tikettikokonaisuudet voidaan puolestaan yhdistää Epic-tyyppiseksi tiketiksi, johon voidaan liittää useita tehtävätikettejä sekä muita tikettityyppejä (Atlassian 2024b). Näiden tikettien linkittäminen testisuunnitelmatikettiin luo sille lisäarvoa sekä tarjoaa testaajille että projektin sidosryhmille selkeän kokonaiskuvan testauksen tilasta ja edistymisestä. Tarkoituksena on varmistaa, että testauksessa kaikki olennaiset vaatimukset ja riskit on otettu huomioon vaaditulla tavalla.

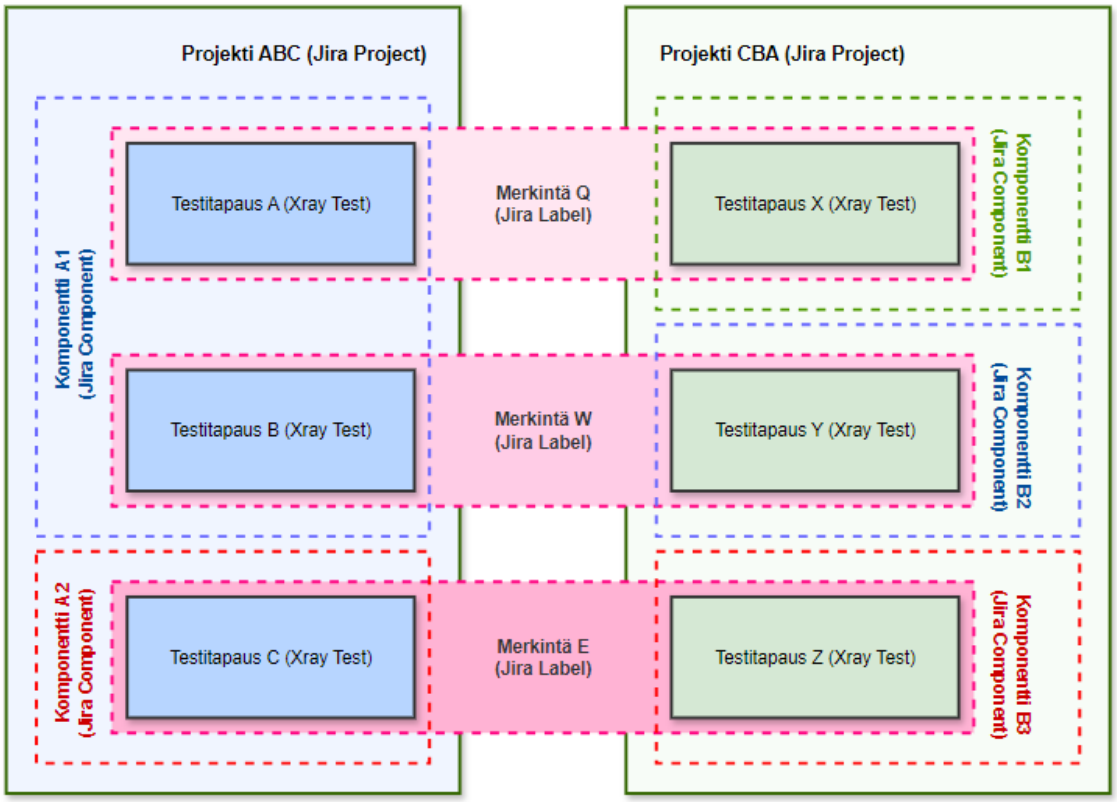



Kuvio 8: Toimintamallissa esitettyjen tikettien linkittyminen keskenään

Mallin mukaiseen suunnitteluvaiheeseen kuuluu lisäksi testaukseen liittyvien tikettien organisoinnin mahdollisuuksien luominen sekä Jiralle että Xray:lle. Jiran näkökulmasta organisointia hallitaan komponenttien (Components) ja merkintöjen (Labels) avulla. Toimintamalli ohjaa testaajaa luomaan Jiran komponentit ja merkinnät tarpeen mukaisesti suunnitteluvaiheessa, jotta kyseiset toiminnallisuudet ovat käytettävissä myöhemmin testauksen toteutuksen aikana.

Jiran komponentit mahdollistavat Jira-projektin ryhmittelyn pienempiin kokonaisuuksiin merkitsemällä tiketit erikseen määriteltyihin projektisidonnaisiin komponentteihin (Atlassian 2024c). Jiran merkinnät puolestaan mahdollistavat tikettien merkitsemisen, jota voidaan käyttää organisoinnissa sekä tikettien etsinnässä merkintään kohdistuvien suodatinehtojen perusteella (Atlassian 2021a).

Opinnäytetyön malli suosittaa testauksen hallinnan tehostamista komponenttien ja merkintöjen avulla. Jos Jiran projektilla ei ole entuudestaan määriteltyjä komponentteja testaukselle, mallin mukaan tulee luoda erillinen testauskomponentti. Tämä komponentti mahdollistaa testausmäärien seurannan ja helpottaa testitapausten erottelua muista Jira-tiketeistä.




Kuvio 9: Hahmotelma Jiran komponenttien ja merkintöjen välisistä eroista

Merkinnät helpottavat testitapauksien ja ennakkoehtojen ryhmittelyä eri kategorioihin, mikä mahdollistaa sujuvan haun ja seurannan koko testausprosessin ajan Jiran toimintoja hyödyntäen. Malli korostaa, että merkintöjen käyttö on organisaationlaajuista, kun komponentit ovat Jira projektikohtaisia. Lisäksi komponentit ja merkinnät eivät ole toisistaan riippuvaisia Jiran näkökulmasta. Toimintamallin mukaisen merkintöjen käytön tarkoituksena on tuottaa lisätietoa Jiran komponentteihin.

Toimintamallin käyttö Jirassa edellyttää, että Jiraan on luotu projekti. Jiran projektit ovat kokonaisuuksia, joihin määritellään kuuluvaksi projektiin liittyviä tikettejä, kuten tehtävätikettejä. Tyypillisesti projekti luodaan esimerkiksi kuvaamaan jotain organisaatiossa tapahtuvaa kokonaisuutta, kuten kehitysprojektia tai palvelua. Malli ei kuitenkaan käsittele erikseen Jira-projektin luomista, vaan olettaa, että sellainen on olemassa Jirassa ennen testauksen aloitusta. (Atlassian 2024e.)

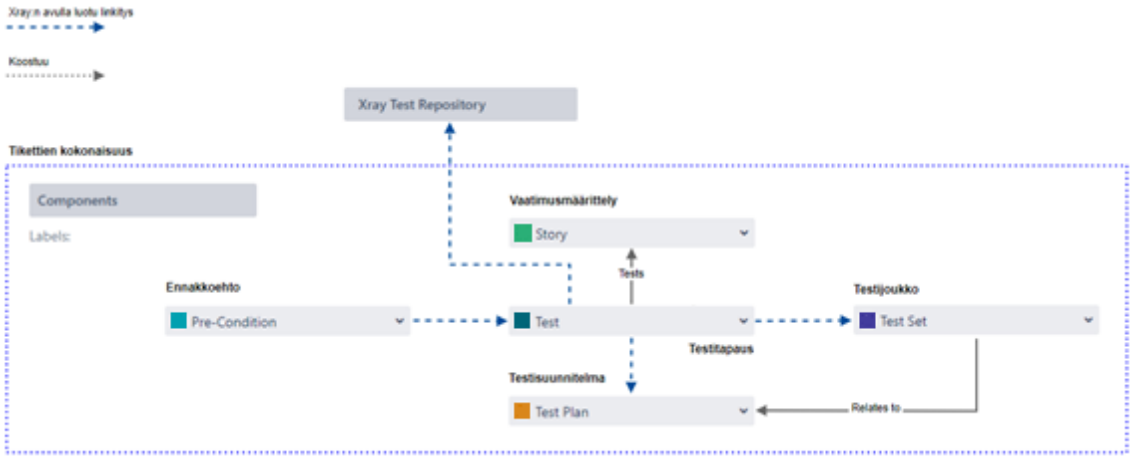
## 5.2 Valmisteluvaiheen toteutus

Opinnäytetyössä esitelty testauksen valmisteluvaihe selkeyttää testausprosessin hallintaa Xray:ssa. Mallin tarkoituksena on opastaa askel askeleelta testaukseen liittyvien tikettien luomiseen, testijoukkojen organisoimiseen ja ennakkoehtojen asettamiseen. Valmisteluvaiheen toimet pyrkivät varmistamaan Xray:n näkökulmasta, että testauksen kattavuus vastaa

suunnitteluvaihetta, sekä että testauskokonaisuus saa loogisen rakenteen. Mallissa esitettyjen vaiheiden avulla voidaan testikokonaisuuksia hallita tehokkaammin, dokumentoida testit selkeästi sekä varmistaa niiden jäljitettävyys Xray:n avulla. Toimintamalli tarjoaa järjestelmällisen lähestymistavan testauksen valmisteluun varmistaen, että kaikki tarpeelliset tiketit sekä muut Jiran määritykset luodaan riittävällä tarkkuudella ennen testauksen aloittamista.

Toimintamallin valmisteluvaiheessa keskiössä ovat Xray:n tikettityypeistä testitapaukset (Test), ennakkoehdot (Pre-Condition) sekä testijoukot (Test Set), jotka Xray:n sekä Jiran tarjoamien linkityksien avulla sidotaan osaksi testisuunnitelmakokonaisuutta. Malli ohjaa testaajaa huomioimaan, että Xray:n avulla määritetyt linkitykset näkyvät osana Xray:n mukana tuomia näkymiä, kun puolestaan Jiran linkitykset ja niiden tyypit ovat testaajan itse määrittelemiä ja näkyvät pääasiassa Jiran toiminnallisuuksina. Lisäksi malli huomioi, että Xray:ssa organisointi perustuu testijoukkotiketteihin sekä testiarkistoihin (Test Repository) ja malli käsittelee edellä mainittujen toiminnallisuuksien toteutuksen osana testauksen valmistelua.

Toimintamalli ohjaa testaajan manuaalisesti määrittämään Xray- ja Jira-tikettien välille linkitykset. Esimerkiksi testitapauksien ja vaatimusmäärittelyiden eli Jiran tarinatikettien tehtävä linkitys on tärkeä osa toimintamallin prosessia. Kyseinen linkitys automaattisesti määrittelee tarinatiketin testitapauksen vaatimukseksi, joka näkyy testisuunnitelmatiketillä.




Kuvio 10: Toimintamallissa esitellyt valmisteluvaiheen tiketit ja toiminnot

Opinnäytetyössä esitelty testauksen valmisteluvaihe ohjaa testaajaa vaihe vaiheelta suorittamaan testauksen valmistelutoimenpiteet Jirassa sekä Xray:ssa seuraavassa, suuntaa antavassa järjestyksessä:

1. Testijoukot
2. Ennakkoehdot
3. Testitapaukset
4. Testiarkistot

Testijoukko koostuu testitapauksista, jotka voidaan ryhmitellä jonkin loogisen kriteerin perusteella joukoksi keskittyen puhtaasti testitapauksien organisointiin (Xray 2024b). Testijoukon avulla testitapaukset voidaan ryhmitellä esimerkiksi testaustyyppien perusteella, kuten suorituskyky-, käytettävyys- tai turvallisuustestaukset ja niin edelleen. Xray:ssa yksi testitapaus voi kuulua useaan testijoukkoon saman aikaisesti. Testisuunnitelmaan verrattuna, testijoukko on huomattavasti yksinkertaisempi kokonaisuus, joka ei ota kantaa siihen kuuluvien testitapauksien etenemisestä tai muihin testaukseen liittyviin yksityiskohtiin.

Testijoukkojen heikkoutena voidaan nähdä se, että testijoukot mahdollistavat vain testitapausten listauksen. Käytännössä tämä tarkoittaa, että testitapauksia ei voida määritellä kuuluvaksi jokin testijoukon alitasolle eli testijoukon sisäiselle testijoukolle. Tällainen organisointi toteutetaan Xray:ssa testiarkistojen avulla. Lisäksi testijoukkojen avulla ei kyetä seuraamaan testitapausten etenemistä, sillä ne tarjoavat vain tiedon siitä, mitkä testitapaukset joukkoon kuuluvat.

Testijoukkojen vahvuutena on, että niiden perusteella voidaan helposti määritellä testisuorituksia, jotka pohjautuvat testijoukkoon määriteltyihin testitapauksiin. Lisäksi testijoukkoja voidaan käyttää esimerkiksi priorisointiin tarvittavien testattavan kohteen osa-alueiden määrittelyyn ja täten tuoda testaukseen halutunlaisia rajauksia sekä rakenteita.

Toimintamalli ohjaa ottamaan testattavan kohteen osa-alueet huomioon testijoukkojen avulla, joko osa-alueina tai suoritettavien testitapausten ryhmittelyn perusteella muun muassa testiympäristön tai testausta määrittävien tekijöiden perusteella. Esimerkiksi tietyt testaustavat voivat toimia loogisina testijoukkojen määrittelyperusteina. Lisäksi malli ohjaa huomioimaan testijoukkojen määrittelyssä vaatimusmäärittelyt siten, että testijoukot rakennetaan kattamaan yksittäistä vaatimusmäärittelyä eli tarinatikettiä kerrallaan.


Kuvio 11: Esimerkki toimintamallin mukaisesti määritellystä testijoukkotiketistä

Testitapauksien ennakkoehdot ovat keskeinen osa mallin mukaista testausta, vaikka niiden käyttö ei ole välttämätöntä. Ennakkoehto määrittelee sellaiset ehdot ja asiat, joiden on toteuduttava, tai jotka otettava huomioon, ennen kuin testitapausta voidaan suorittaa. Yhtä ennakkoehtoa voidaan käyttää ehtona usealle eri testitapaukselle. Tyypiltään ennakkoehdot voivat olla testitapauksia vastaavalla tavalla manuaalisia, geneerisiä Cucumber-kehyksen mukaisia, mutta esitetty malli keskittyy käyttämään vain manuaalista tyyppiä. (Xray 2024c.)

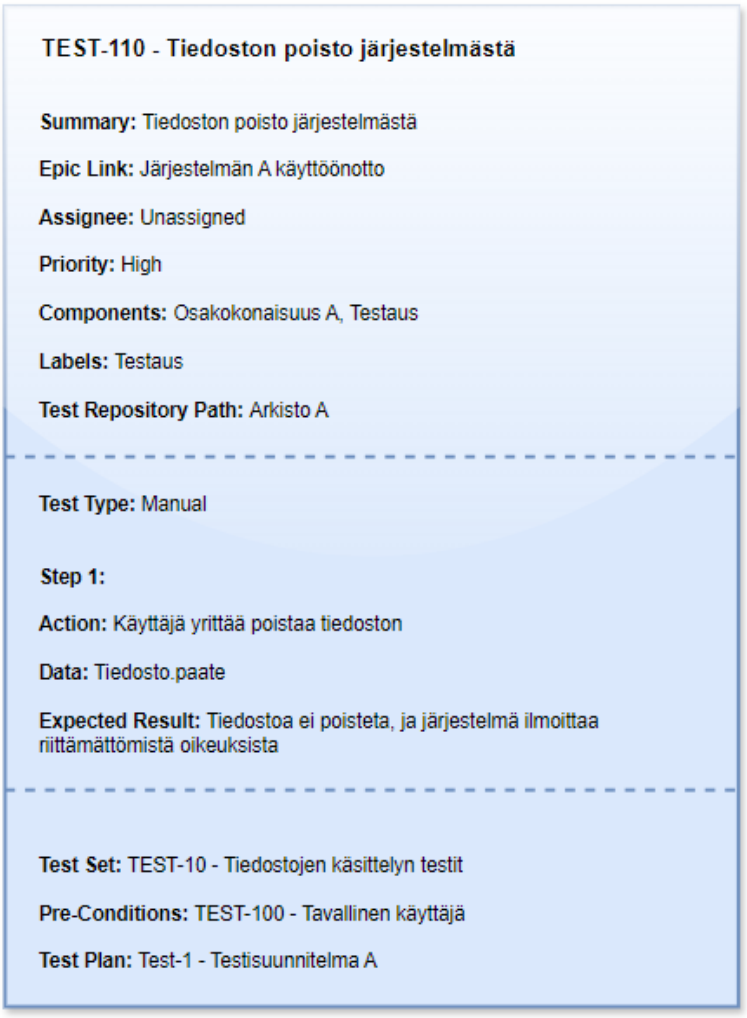
Opinnäytetyön malli ohjaa laatimaan ennakkoehdot GWT-mallin mukaisesti. Tällöin GWT-mallin Given-osa kuvaa alkutilannetta, jonka tulee olla voimassa ennen testin suoritusta. Mallin mukaisesti, jos testitapaukselle ei ole tarvetta määritellä Given-lausetta, jätetään ennakkoehtotiketti luomatta.


Kuvio 12: Esimerkki toimintamallin pohjalta luodusta ennakkoehtotiketistä

Testitapaukset muodostavat testauksen ytimen ja kuvaavat yksityiskohtaisesti testattavat toiminnallisuudet strukturoidulla tavalla. Toimintamalli opastaa luomaan testitapaustiketit valmisteluvaiheen esittelemän laatimistavan mukaisesti. Xray:n mukainen toimintapa jäsentelee jokaisen testitapauksen koostuvaksi yhdestä tai useammasta testivaiheesta (Test Steps). Nämä vaiheittain etenevät tiketit dokumentoivat jokaisen testauksen suoritusvaiheen riittävän yksiselitteisesti, jotta testitapaus vastaa sille määriteltyjä vaatimuksia. (Xray 2024d.)

Mallin mukaiset testitapaukset kirjoitetaan Xray:n manuaalitestaustyyppinä, jolloin jokainen testivaihe määritellään GWT-mallin mukaisesti käyttäjän näkökulmasta testitapausperusteisena lähestymistapana. Malli ohjaa testaajaa kirjoittamaan testitapaukset huomioiden seuraavat Xray:n testivaiheen määritykset:

- **Action**: Testivaiheen aloitus, joka vastaa GWT-mallin mukaista *When*-osaa. Tämä kuvaa konkreettisen toiminnon suoritusta tai tapahtumista, kuten "Kun käyttäjä yrittää poistaa tiedoston."
- **Data**: Mahdollinen tarvittava syöttödata testin aikana, kuten tiedoston nimi tai käyttäjän syöte. Kyseiselle määritykselle ei ole GWT-mallissa erillistä kohtaa, mutta määrityksen määrittelyyn käytetään apuna valmisteluvaiheen osuutta, jossa valmistellaan testiaineisto
- **Expected Result**: Odotettu lopputulos eli GWT-mallissa esitetty *Then*-osa. Esimerkinä lopputulos "Tiedostoa ei poisteta, ja järjestelmä ilmoittaa riittämättömistä oikeuksista."



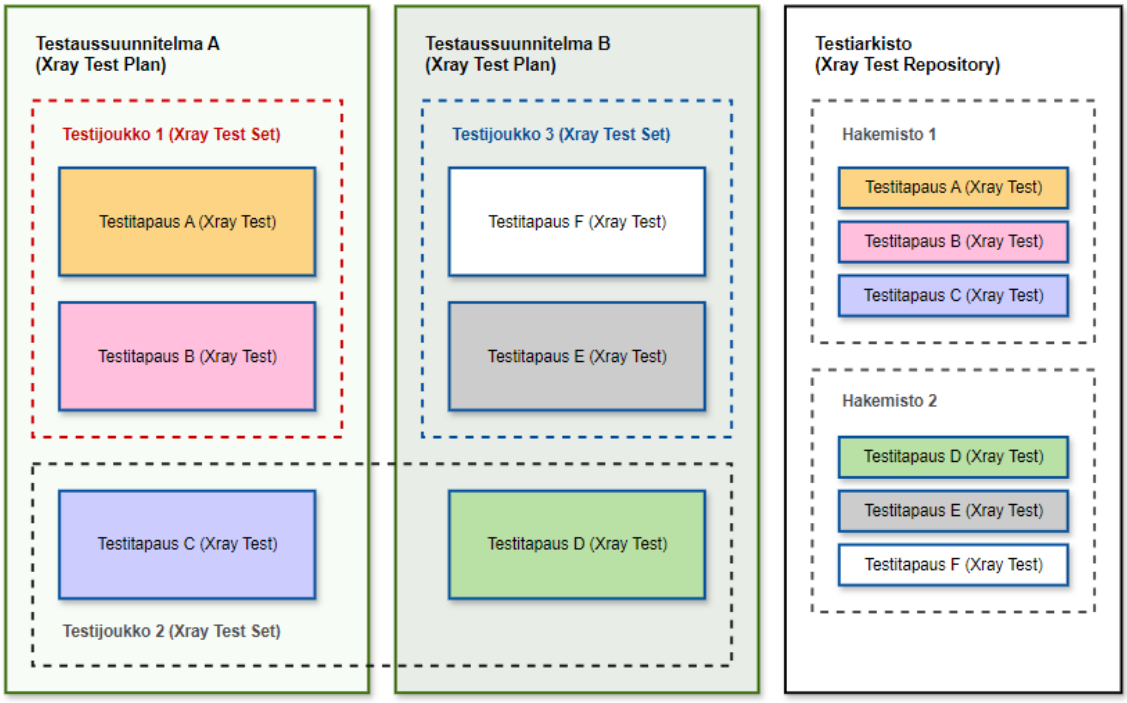

Kuvio 13: Esimerkki mallin mukaisesta testitapaustiketistä

Testitapaukset määritellään toimintamallin mukaan kuuluvuksi testijoukkoihin. Tämän lisäksi testitapauksiin linkitetään erikseen sen prioriteettia sekä vaatimuksia kuvaava vaatimusmäärittely eli tarinatiketti, jotta se näkyy automaattisesti testisuunnitelmatiketillä kyseisen testitapauksen vaatimuksena.

Toimintamallin mukaisesti testiarkistot (Test Repository) on tärkeä Xray:n ominaisuus testauksen hallinnassa ja organisoinnissa. Testiarkistojen avulla testitapaukset voidaan järjestellä Jiran projektikohtaisiin hakemistoihin ja näiden alihakemistoihin helpottaen suurempien sekä monimutkaisempien testauskokonaisuuksien hallintaa, etenkin jos projektissa on useita eri testauskokonaisuuksia ja niihin kuuluvia testisuunnitelmia. (Xray 2024d.)

Testiarkistojen käyttö parantaa testien organisointia ja helpottaa testaajien pääsyä tarpeellisiin testitapauksiin. Testiarkistot mahdollistavat testitapausten selkeän ryhmittelyn, mikä selkeyttää testausprosessia ja vähentää epäselvyyksiä mahdollisten muiden testauskokonaisuuksien suhteen.

Testiarkistot luovat hierarkkisen hakemistorakenteen, jota ei ole mahdollista toteuttaa pelkästään Jiran tai Xray:n tikettien avulla. Hierarkkinen rakenne lisää testitapauksien organisoinnin mahdollisuuksia sekä syvyyttä. Lisäksi testiarkistot tukevat testauksen toistettavuutta ja seurantaa, koska niiden avulla voidaan varmistaa, että testitapaukset ovat helposti löydettävissä projektin edetessä.

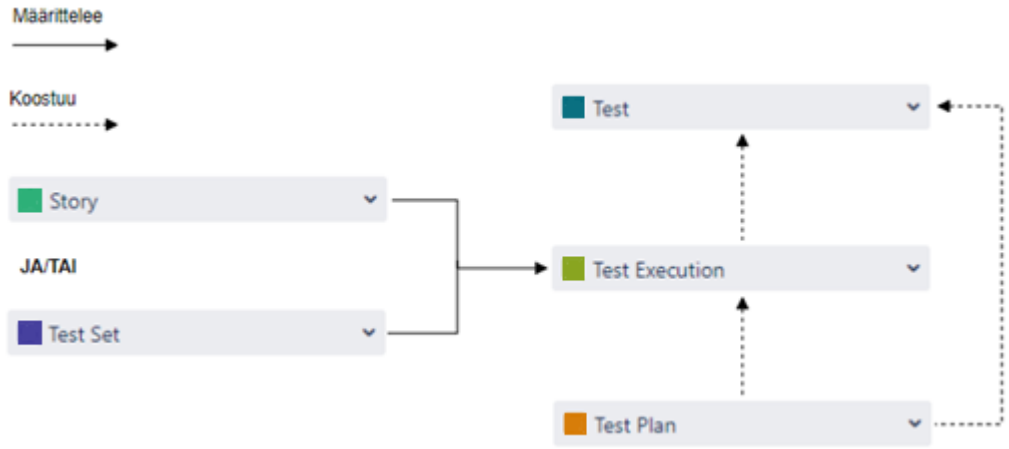



Kuvio 14: Esimerkki testijoukkojen ja testiarkiston välisestä organisoinnin eroavaisuudesta

Valmisteluvaiheen toteutuksessa otetaan lisäksi huomioon suunnitteluvaiheessa luotuja Jiran keinoja organisoida tikettejä komponentteihin sekä määritellä tiketeille lisätietoa merkintöjen avulla. Käytössä olevat komponentit sekä merkinnät merkitään mallin mukaisesti jokaiselle luodulle valmisteluvaiheen tiketille.

## 5.3 Suoritusvaiheen toteutus

Testisuoritus (Test Execution) on Xray:n tikettityyppi, johon voidaan liittää yksi tai useampi testitapaus muodostaen suoritettavien testitapauksien kokoelman. Testisuoritustiketti muodostaa yhteyden testitapauksen ja itsensä välille testiajona (Test Run), jolloin tiketistä muodostuu listaus siihen liitetyistä testitapauksista sekä näiden suorittamisen tilasta. Testiajo itsessään ei ole erillinen tikettityyppi, vaan toiminnallinen osa testisuoritustikettiä, joka kuvaa testitapauksen suoritusta. (Xray 2024e; Xray 2024f.)

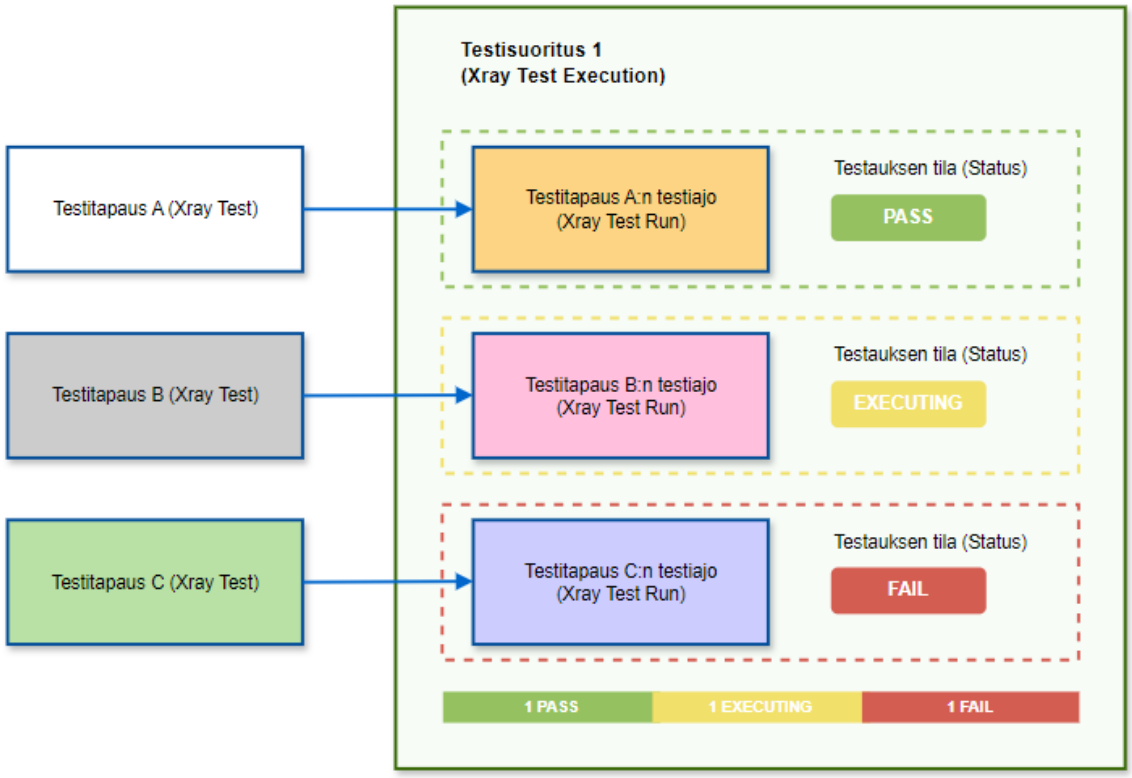



Kuvio 15: Toimintamallissa esitetyt suoritusvaiheen tiketit

Testisuoritustikettiin määriteltyjen testitapauksien suorittaminen muuttaa testiajon tilaa sen mukaan, missä vaiheessa suoritus on. Esimerkiksi, kun testitapaus on saatu suoritettua, voi testiajon tila eli status olla muun muassa PASS tai FAIL, testisuorituksen lopputuloksesta riippuen. Yksittäisen testiajon tila vaikuttaa koko testisuoritustiketin tilaan, sillä testisuoritustiketti seuraa siihen kuuluvien testiajojen etenemistä ja perii suoritusten statukset osaksi laajempaa kuvausta testisuoritustiketin testisuorituksista. (Xray 2024f.)

Xray:n (2024f) määritelmänä testisuorituksien etenemistä kuvataan seuraavasti väreillä:

- **PASS** (vihreä teksti) eli testaus on suoritettu onnistuneesti
- **FAIL** (punainen teksti) eli testaussuoritus on epäonnistunut
- **EXECUTING** (keltainen teksti) eli testaus on aloitettu, mutta kesken
- **TODO** (harmaa teksti) eli testaus odottaa suorittamista
- **ABORTED** (musta teksti) eli testaus on keskeytetty



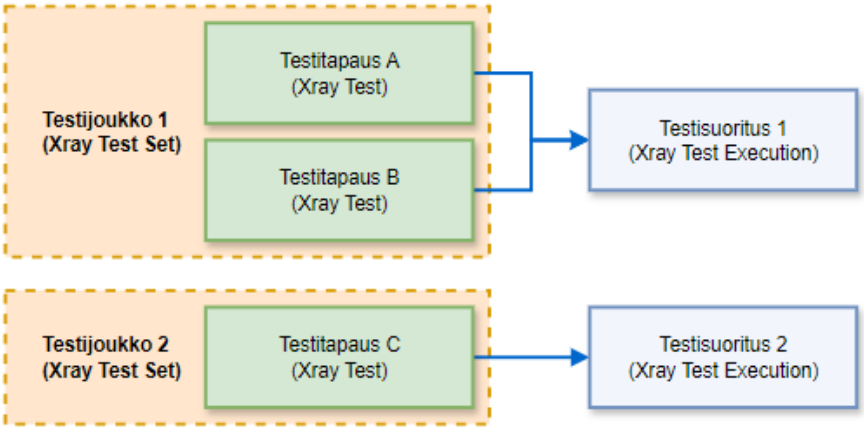

Kuvio 16: Hahmotelma testisuoritustiketin, testitapauksien ja testiajojen yhteydestä toisiinsa sekä testauksen tilaan

Xray:ssa yksi testitapaus voidaan liittää vain kerran testisuoritukseen tarkoittaen, että testitapaustiketti voidaan kerran suorittaa yhdellä testisuoritustiketillä. Jos testitapaus halutaan ajaa uudestaan, on luotava uusi testitapaus, joka liitetään olemassa olevaan tai uuteen testisuoritustikettiin. Vaihtoehtoisesti voidaan luoda uusi testisuoritus, joka yhdistetään uusittavaksi määriteltyyn testitapaukseen. Toimintamalli käsittelee testauksen uudelleen suorittamista testisuoritusten ja -tapausten uusimisen kautta.

Toimintamalli ohjaa suunnittelemaan testisuoritukset prioriteettiperusteisesti, mikä tarkoittaa testitapauksien suorittamista tärkeysjärjestyksessä. Mallin mukaan kaikki korkeimman prioriteetin testit voidaan ryhmittää yhteen testisuoritukseen. Tällainen lähestymistapa tekee priorisoitujen testitapausten seurannasta selkeää, sillä kaikkia suorituksia, niin onnistuneita kuin epäonnistuneita, voidaan tarkastella yhdestä näkymästä. Tämä helpottaa testitapauksien kokonaisvaltaisen suorituksen statuksen seurantaa ja raportointia riskiperusteiseen lähestymistapaan pohjautuen.

Malli korostaa, että edellä kuvattu lähestymistapa ei välttämättä ole ihanteellinen tilanteissa, joissa on suuri määrä erilaisia testitapauksia, tai jos tarvitaan yksityiskohtaisempaa tietoa testien suorituksista muilla perusteilla kuin prioriteetin mukaan. Tällaisissa tapauksissa malli ohjaa luomaan testisuoritukset esimerkiksi testijoukkojen perusteella ja liittämään

vaatimusmäärittelyt erikseen testitapauksiin. Edellä kuvattu toimintatapa ei kuitenkaan jätä huomioimatta prioriteettiin perustuvaa aikataulusta ja suoritusta, vaan organisoi testauksen tarkemmaksi hajautettuun kokonaisuuteen.




Kuvio 17: Hahmotelma testisuoritustiketin muodostamisesta testijoukkoon perustuen

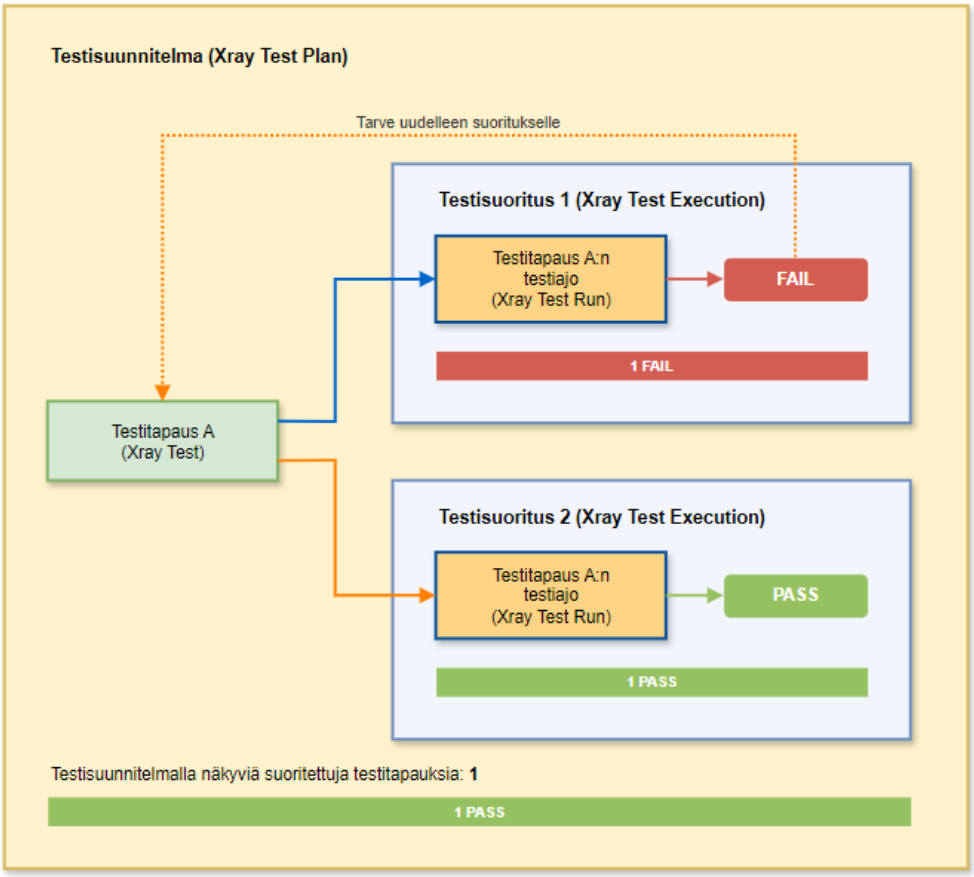
Mallin mukaisesti testisuorituksia voidaan suunnitella ja jaotella esimerkiksi seuraavilla perusteilla:

- **Osa-aluekohtainen jaottelu** ohjaa jakamaan testisuoritukset testattavan kohteen ominaisuuksien mukaan. Lähestymistapa on hyödyllinen esimerkiksi silloin, kun testattavan kohteen osa-alueeseen kohdistuu useita eri prioriteettiluokkia.
- **Testitapauskohtainen lähestymistapaa** mukaillen, jolloin jokaiselle testitapaukselle luodaan oma yksittäinen testisuoritustiketti. Tämä lähestymistapa on hyödyllinen lähinnä silloin, kun testitapauksia on vain muutamia ja ne eroavat toisistaan merkittävästi.
- **Testiympäristökohtainen suoritusten järjestäminen** ohjaa testaajaa hyödyntämään testisuoritustiketin testiympäristö eli Test Environment -kenttää testisuoritusten hallinnassa, jolloin testit voidaan suorittaa ja raportoida eri testiympäristöjen mukaisesti. Tämä on hyödyllistä, jos halutaan kohdentaa testisuoritusten seurantaa testiympäristön perusteella prioriteetin lisäksi.

Toimintamallin pyrkimyksenä on tarjota joustavat keinot testisuoritusten organisointiin, jolloin testaaja voi valita parhaiten soveltuvan tavan testaukseen ja sen tarpeisiin.

Testisuoritusten uusimisessa toimintamalli ohjaa huomioimaan, että Xray:ssa yhden testitapauksen voi suorittaa yhdessä testisuoritustiketissä kerran. Toisin sanoen, testitapausta ei voida suoraan suorittaa uudelleen käyttäen samaa testisuoritustikettiä. Tämän seurauksena testitapauksen uusimista varten pitää luoda joko uusi testisuoritus tai kokonaan uusi testitapaus.

Mallin mukaisesti testauksen seurannan, raportoinnin sekä toistettavuuden näkökulmasta uuden testisuorituksen luominen on ensisijainen tapa uusia testitapauksia. Tällöin testauksien uusinnat suunnitellaan mallin mukaisesti siten, että uusittavia testejä varten luodaan kattavuudeltaan alkuperäistä testisuoritusta vastaava testisuoritustiketti, mutta uudelleen nimettynä. Seurauksena yksittäinen testitapaus kuuluu lopulta kahteen testisuoritukseen ja suorituskohtaiseen testiajoon, joista kummallakin voi olla eri tai sama lopputulos testitapauksen suorituksesta. Lisäksi uuden testisuorituksen käyttö lähestyy iteroivaa toimintatapaa, jossa kunkin iteraation testisuorituksia voidaan seurata mielekkäästi testisuoritusten sekä testisuunnitelman kautta.

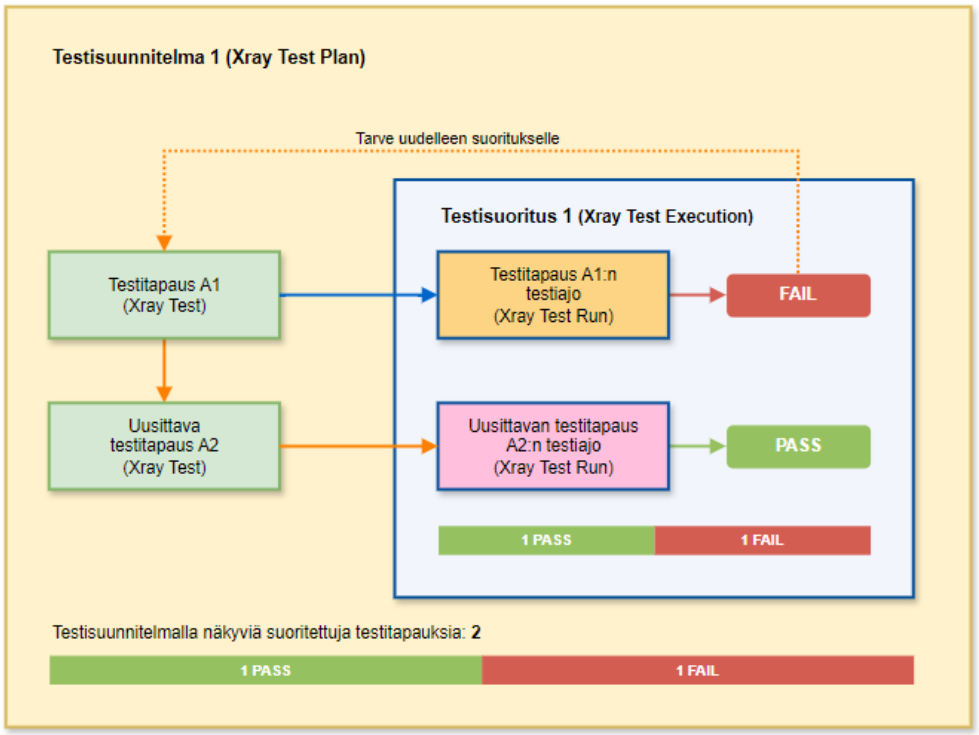



Kuvio 18: Hahmotelma testisuorituksen uusinnasta

Etuina uuden testisuoritustiketin käytössä ovat muun muassa se, että testitapauksien määrän seuranta sekä hallinta on suoraviivaisempaa ja selkeämpää, sillä testitapauksien määrä ei vaihtele uusintojen mukaan. Lisäksi kustakin testisuorituksesta voidaan nähdä suoraan, kuinka monta testitapausta siihen on liitetty ja minkälaisia tuloksia testeistä on saatu. Testisuorituksen uusinta mahdollistaa, että testitapaukset ovat suoraan toistettavissa uusinnan avulla, sillä testitapaus pysyy muuttumattomana. Jos uusittavia testitapauksia on useita, tikettien kokonaishallittavuus pysyy helpompana, sillä testaukseen liittyvien tikettien määrä pysyy rajatumpana, koska yhdellä suoritustiketillä voidaan uusia lukuisia testitapauksia.

Joissakin tilanteissa testisuoritusten uusintaa voidaan haluta lähestyä uuden testitapaustiketin luomisen kautta käyttäen olemassa olevaa testisuoritusta sekä alkuperäisen että uuden testitapauksen suorittamiseen. Tällainen tilanne voi esimerkiksi muodostua, kun halutaan säilyttää historiatieto alkuperäisen testitapauksen suorittamisesta ja jättää alkuperäinen testiajon status nähtäviin. Testitapausta uusittaessa voi lisäksi olla tarpeen pohtia sekä testitapauksen että -suorituksen samanaikainen uusiminen, jolloin mahdollinen virhetilanne alkuperäisellä suorituksella jää selkeämmin dokumentoiduksi testienhallintasovellukseen.

Toimintamalli ohjaa testitapauksen uusinnassa huomioimaan, että tällöin raportoitujen testitapauksien määrä testisuoritustiketillä kasvaa. Esimerkiksi, jos yhdellä testisuoritustiketillä on tarkoitus suorittaa käytännössä yksi testitapaus, näkyy uusi testitapaustiketti käytetyllä testisuoritustiketillä siten, että testitapauksia on suoritettu kaksi. Jos ensimmäinen testitapaus epäonnistuu ja uusintatestaus onnistuu, testisuoritus raportoi, että puolet testitapauksista on epäonnistunut ja toinen puoli onnistunut. Tämä toiminnallisuus voi hankaloittaa testauksen kokonaisuuden seurantaa, sillä testisuorituksilta ei nähdä suoraan, kuinka monet testitapauksista ovat olleet uusintoja, tai kuinka moni alkuperäinen testitapaus on onnistunut.




Kuvio 19: Hahmotelma testitapauksen uusinnasta

Toimintamalli käsittelee lisäksi testisuoritusten seurannan mahdollisuuksia huomioimalla, että Xray:ssa on useita erinäisiä keinoja seurata testisuoritusten edistymistä. Yksi keinoista on tarkastella testisuoritukseen määriteltyjen testitapausten testiajojen tilaa suoraan testisuoritustiketiltä. Tällöin tilanteessa, jossa testisuoritus on rakennettu vastaamaan suoraan

vaatimusmäärittelyä, voidaan suoraan arvioida, kuinka hyvin testaus on suorituksen osalta vastannut testaukselle asetettuja vaatimuksia. Lisäksi testisuoritustiketin kautta voidaan tarkastella tarkemmin yksittäisen testitapauksen testivaiheiden onnistumista erillisen testiajokohtaisen näkymän kautta.

Testisuunnitelmatiketin avulla testaaja voi seurata testauksen edistymistä suunnitelmatasolla siihen liitettyjen testitapausten sekä -suoritusten perusteella. Suunnitelmatason seurannalla testaaja, tai muu taho, voi siten tarkastella suunnitelman kokonaisvaltaista toteutumista ja tehdä testauksen toteutumisen perusteella erinäisiä arvioita testaukseen liittyen muun muassa kattavuuden suhteen. Lisäksi Xray mahdollistaa testisuoritusten tarkastelua testitapauksen kautta, jolloin testaaja voi tarkastella yksittäiseen testitapaukseen liitettyjen testiajojen onnistumista. Tätä ominaisuutta hyödyntämällä testaaja voi helposti arvioida esimerkiksi testitapausten suoritusten määrää yksittäistä testitapausta kohden ja saada käsityksen uusintojen määrän tarpeesta, jotta tietty lopputulos saavutettiin testauksen avulla.

## 5.4 Raportointivaiheen toteutus

Xray:ssa on sisäänrakennettu raportointiominaisuus, jonka avulla voidaan generoida valmisraportteja Jiran projektikohtaisesti testissuunnitelma, -suoritus tai -ajotasolla. Generoitujen raporttien suodatinominaisuudella voidaan raportteja kohdentaa ja rajata tarpeenmukaisesti. Tässä suhteessa esimerkiksi Jiran merkintöjen käytöstä voidaan saada lisähyötyä. Kehitetty toimintamalli käsittelee Xray:n valmisraportteja edellä mainittujen kolmen tason kautta sekä ohjaa testaajaa huomioimaan niiden väliset erot.

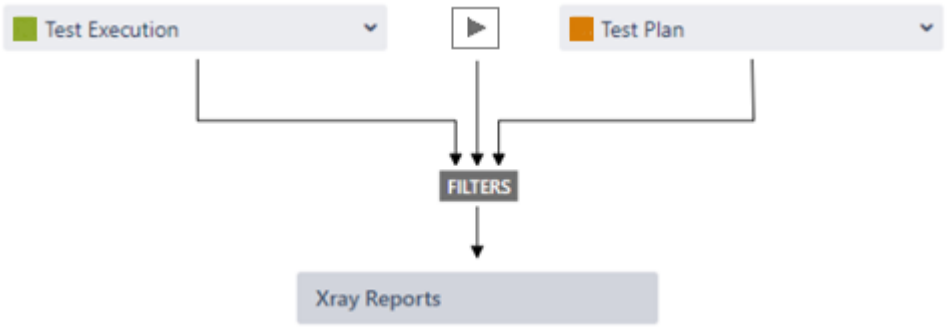
Toimintamalli ei käsittele erillisten räätälöityjen raporttien laatimista tai muita Jiran tai Xray:n raportointimahdollisuuksia. Mallissa muut raportointimahdollisuudet, kuten Atlassian Confluencen ja JQL:n (Jira Query Language), on esitetty erillisen maininnan tasolla. Tässä yhteydessä mainittu Confluence on Atlassianin yhteistyöalusta tiedon jakamiseen sekä dokumentointiin (Atlassian 2024f). Confluence itsessään mahdollistaa tietojen hakemisen Jirasta, joten sen avulla kyetään toteuttamaan erinäisiä ratkaisuja raportoinnin suhteen.

Toimintamalli käsittelee Xray:n valmisraportteja ja niiden generoimista seuraavista Xray:n testaustyyppisistä tiketeistä:

- Raportti testisuunnitelmista (Test Plans Report)
- Raportti testisuorituksista (Test Executions Report)
- Raportti testiajoista (Test Runs Report)

Jokainen edellä mainitusta raporteista käsittelee testausta Jiran projektin sisäisesti, jolloin raportilta voi olla nähtävissä useamman testisuunnitelman tiedot käytettäessä testisuunnitelmakohtaista raportointia, jos projektiin on määritelty useita testisuunnitelmatiketteiä.

Testauksen malli ohjeistaa esimerkiksi suodattimien käyttöön, jotta raportista saadaan nostettua esille etsityt tiedot halutunlaisista testauskokonaisuuksista.




Kuvio 20: Kuvaus raportoinnin yhteydestä testaustiketteihin

# 6 Johtopäätökset

Opinnäytetyö toteutui toiminnallisena kehittämistyönä, jossa toimeksiantajalle kehitettiin kehittävän toiminnan avulla ratkaisu testaustarpeisiin. Kehittämistyön taustalla hyödynnettiin osallistuvaa havainnointia ja haastattelututkimusta sekä näihin perustuvaa laadullista analyysia tarkoituksenaan ohjata tiedon tuottamista toimeksiantajan toimintaympäristöön soveltuvaksi. Lisäksi kehitystyön suuntaa määriteltiin ohjelmistotestauksen teoreettisen tiedon lähtökohdista, joiden avulla opinnäytetyön ratkaisuja sidottiin yleisesti hyväksyttyyn tietoon.

Opinnäytetyön työstämisessä korostui kehitettävän mallin jatkuva arviointi sekä iteratiivisuus toimintamallin käyttökokemukseen perustuen. Käytännössä tämä tarkoitti, että toimintamallin pohjalta luotiin tapausesimerkkinä testattava kokonaisuus valittuun testienhallintasovellukseen. Tapausesimerkin myötä saatujen käyttökokemusten perusteella toimintamallia jatkokehitettiin jatkuvasti yhdessä toimeksiantajan edustajan kanssa tämän antaman palautteen perusteella.

Toimintamallin toteutus muuttui opinnäytetyön työstämisen aikana kehityssyklien myötä sovellettavuutensa ja malliin rakennettujen yksityiskohtien suhteen tarkemmaksi. Kehitysvaiheen jatkuva sovellettavuuden arviointi yhdistettynä kehityksen syklisyyteen osoittautui toimivaksi ratkaisuksi, jolla toistettavuuden kautta mallin käyttöön saatiin varmuutta sekä rakennettua riittävän yksityiskohtaiset kuvaukset toimenpiteiden toteutukselle. Lisäksi iteratiivisen lähestymistavan myötä erinäiset virheet ja puutteet toimintamallin ratkaisuissa kyettiin havaitsemaan suhteellisen nopeasti, ja ne saatiin korjattua varhaisessa vaiheessa.

Opinnäytetyön työstämisen yhteydessä havaittiin, että toimintamallin rakentaminen erityisen tarkkarajaiseksi tuotti haasteita toimeksiantajan toimintaympäristön moninaisuuden vuoksi.

Toimintamallista piti muodostaa muodoltaan sellainen, että se olisi käytettävissä toimeksiantajan organisatorisen ryhmän sisällä useissa käyttötarpeissa. Tämän myötä mallin alkuperäinen tarkempi rajaus esimerkiksi järjestelmätestaukseen jäi pois kehityksen myötä.

## 6.1 Lopputuloksen arviointi

Opinnäytetyön lopputuloksena kehitetty testauksen toimintamalli onnistui useamman kehityssyklin päätteeksi vastaamaan opinnäytetyön tutkimuskysymyksen ensimmäiseen osaan. Toimintamalli, ja siitä luotu kuvaus toimeksiantajan toimintaympäristössä tehtävälle testaukselle, muodostivat tavoitteiden mukaisesti suhteellisen yksinkertaisen ohjeistuksen siitä, kuinka mallia voidaan hyödyntää osana toimeksiantajan testauskäytänteitä. Opinnäytetyö kykeni lisäksi tuottamaan vastauksen tutkimuskysymyksen toiseen osaan osoittamalla kehitetyn testausmallin sovellettavuuden Xray-testienhallintaohjelmiston kanssa.

Opinnäytetyön toiminnallinen tuotos kokonaisuudessaan kykeni luomaan toimeksiantajalle konkreettisen tuotoksen ja sitä kautta luomaan hyödyllistä tietoa organisaation käyttöön. Opinnäytetyön lopputulosta kyettiin hyödyntämään tapausesimerkin myötä toimeksiantajan testaustarpeessa, joka konkreettisesti osoitti kehitetyn ratkaisun käyttömahdollisuudet jatkossa. Lisäksi toimeksiantajan jatkosuunnitelmana oli ottaa ratkaisu käyttöön sellaisenaan.

Toimintamallin käyttö Xray-testienhallintasovelluksen kanssa kykeni osoittamaan toimeksiantajalle toimintamallin sovellettavuuden kyseisen sovelluksen kanssa. Lisähyötynä toimeksiantaja sai havainnollistavan esimerkin Xray:n ja Jiran käyttömahdollisuuksista testauksen hallinnan suhteen.

## 6.2 Jatkokehitys

Tässä raportissa käsiteltiin aiemmin mainintana Cucumber-kehystä, joka luo puitteet automaatiotestaukselle käyttäytymiseen perustuvaa kehitystapaa hyödyntäen. Opinnäytetyössä kehitetty malli kykenisi toimimaan pohjana toimeksiantajan toimintaympäristöön kehitettävälle tavalle tehdä automaatiotestausta Cucumber-kehystä käyttäen. Kehitetty toimintamalli toimisi osittain suoraan Cucumber-kehyksen kanssa. Lisäksi kehitetty malli soveltuisi käytettäväksi Xray-testauksenhallintasovelluksessa suoraan, joka tukee Cucumber-tyyppistä testausta, joten mallia jatkokehittämällä voitaisiin luoda toimintamalli Xray;n avulla toteutettaville Cucumber-testauksille.

Toimintamallin kehityksen myötä nousi esille huomio, että mallin kehitys tasapainotteli jossain määrin riittävän ja riittämättömän tarkkuuden kanssa, sillä toimeksiantajan tarpeet testaukselle saattoivat vaihdella huomattavasti toimeksiantajan toimintaympäristöjen sekä käyttötarpeiden mukaan. Tämän perusteella mallin jatkokehitys siihen suuntaan, että sitä

voitaisiin hyödyntää laajemmin ja erilaisissa testaustarpeissa, voisi tuoda toimeksiantajalle lisäarvoa opinnäytetyön myötä kehitystä mallista.

Toimintamallin teknisestä näkökulmasta mallia voisi kehittää esimerkiksi siten, että se sisältäisi videoituja toimintaohjeita tietyille suoritettaville toiminnoille testienhallintasovelluksen suhteen. Näiden toimintojen sanallinen kuvaaminen mallissa tuotti osaltaan haasteita ja samalla mallin ohjeistuksen yksinkertaisuus kärsi. Mallin houkuttelevuuden sekä käytettävyyden kannalta, yksityiskohtaiset sanalliset selitykset ja kuvaukset tehtävistä toiminnoissa voivat olla haitaksi. Tällöin selkeä ja nopea tapa osoittaa tehtävät toimet esimerkiksi videoinnin avulla voisivat toimia ratkaisuna.

Lähteet

AnAr. 2023. Top 3 Programming Languages for Automation Testing. Viitattu 2.11.2024. https://anarsolutions.com/programming-languages-for-automation-testing/

Anwar, N. & Kar, S. 2019. Review Paper on Various Software Testing Techniques & Strategies. Global Journal of Computer Science and Technology: Software & Data Engineering 19 (2). Global Journals.

ASTQB. 2024. ISTQB Foundation Level – Seven Testing Principles. Viitattu 25.8.2024. https://astqb.org/istqb-foundation-level-seven-testing-principles/

Atlassian. 2021a. Using labels in Jira. Viitattu 28.7.2024. https://community.atlassian.com/t5/Jira-articles/Using-labels-in-Jira/ba-p/1782833

Atlassian. 2021b. Using labels in Jira. Viitattu 3.11.2024. https://community.atlassian.com/t5/Jira-articles/Using-labels-in-Jira/ba-p/1782833

Atlassian. 2022. 13 reasons why Jira is so popular. Viitattu 19.10.2024. https://community.atlassian.com/t5/Jira-articles/13-reasons-why-Jira-is-so-popular/ba-p/1922986

Atlassian. 2024a. Xray Test Management for Jira. Viitattu 20.10.2024. https://marketplace.atlassian.com/apps/1211769/xray-test-management-for-jira?hosting=cloud&tab=overview

Atlassian. 2024b. What are issue types?. Viitattu 28.7.2024. https://support.atlassian.com/jira-cloud-administration/docs/what-are-issue-types/

Atlassian. 2024c. What are Jira components?. Viitattu 28.7.2024. https://support.atlassian.com/jira-software-cloud/docs/what-are-jira-components/

Atlassian. 2024d. What is a Jira project?. Viitattu 3.9.2024. https://support.atlassian.com/jira-software-cloud/docs/what-is-a-jira-software-project/

Atlassian. 2024e. Stories, epics, and initiatives. Viitattu 19.10.2024. https://www.atlassian.com/agile/project-management/epics-stories-themes

Atlassian. 2024f. Confluence. Viitattu 20.10.2024. https://www.atlassian.com/software/confluence

Atlassian. 2024g. What is Lean project management?. Viitattu 3.11.2024. https://www.atlassian.com/agile/project-management/lean-vs-agile

BairesDev. 2024. Positive & Negative Testing Compared: Strategies & Methods. Viitattu 1.9.2024. https://www.bairesdev.com/blog/positive-and-negative-testing/

Companiesmarketcap. 2024. Largest Companies by Marketcap. Viitattu 4.11.2024. https://companiesmarketcap.com/

Cucumber. 2024. Gherkin Reference. Viitattu 13.1.2024. https://cucumber.io/docs/gherkin/reference/

Erlund, K., Aalto-Setälä, M., Hynönen, K., Lilja, J., Lindfors, A., Nevasalo, T., Salminen, J. & Turunen, J. 2022. IT2022 - käytännön käsikirja. E-kirja. Helsinki: Kauppakamari.

Haaga-Helia AMK. 2022. Ohje toiminnalliselle opinnäytetyölle. Viitattu 3.11.2024. https://www.haaga-helia.fi/sites/default/files/file/2024-01/toiminnallinen-ont-ohje.pdf

Hakala, T. J. 2024. Laadullisen tutkimuksen ABC. E-kirja. Helsinki: Gundeamus.

Hallamaa, T. 2023. ICT-alan päästöt ovat jopa lentoliikennettä isommat – nyt ala etsii vihreää koodia. YLE. Viitattu 4.11.2024. https://yle.fi/a/74-20038610

HAMK. 2024. Opinnäytetyö. Viitattu 3.11.2024. https://www.hamk.fi/opiskelijalle/opintojen-suunnittelu/opinnaytetyo/

Hyvärinen, M., Nikander, P. & Ruusuvuori, J. (toim.). 2017. Tutkimushaastattelun käsikirja. E-kirja. Tampere: Vastapaino.

IBM. 2024. What is software testing?. Viitattu 1.9.2024. https://www.ibm.com/topics/software-testing

Idera. 2021. Idera, Inc. Acquires Xray and Xporter. Viitattu 20.10.2024. https://www.idera-racorp.com/pressreleases/acquires-xblend

IEEE. 2024. The Importance of Software Testing. Viitattu 1.9.2024. https://www.computer.org/resources/importance-of-software-testing

ISTQB. 2024. Welcome to ISTQB. Viitattu 19.10.2024. https://www.istqb.org/

Jira. 2024. Welcome to Jira. Viitattu 19.10.2024. https://www.atlassian.com/software/jira/guides/getting-started/introduction#what-is-jira-software

Kajander, R., Kavander, A. & Hirvonen, S. 2024. Ohjelmistovirhe sotki maailmalla lentoja, maksuja ja sairaanhoitoa – suomalaisasiantuntija: ”Ihan uudella levelillä”. YLE. Viitattu 4.11.2024. https://yle.fi/a/74-20100448

Kasurinen, J. P. 2014. Ohjelmistotestauksen käsikirja. E-kirja. Jyväskylä: Docendo.

Juuti, P. & Puusa, A. 2020. Laadullisen tutkimuksen näkökulmat ja menetelmät. E-kirja. Helsinki: Gaudeamus.

RobotFramework. 2024. BDD (Behavior Driven Development). Viitattu 14.1.2024. https://docs.robotframework.org/docs/testcase_styles/bdd

Rytkönen, M. & Kakkonen, K. 2023. Act 2 Lead. Ohjelmistotestauksen johtamisen käsikirja. E-kirja. Tuusula: Ketterät Kirjat Oy.

Saaranen-Kauppinen, A. & Puusniekka, A. 2006a. Strukturoitu ja puolistrukturoitu haastattelu. KvaliMOTV. Viitattu 4.11.2024. https://www.fsd.tuni.fi/menetelmaopetus/kvali/L6_3_3.html

Saaranen-Kauppinen, A. & Puusniekka, A. 2006b. Teemahaastattelu. KvaliMOTV. Viitattu 4.11.2024. https://www.fsd.tuni.fi/menetelmaopetus/kvali/L6_3_2.html

Saaranen-Kauppinen, A. & Puusniekka, A. 2006c. Osallistuva havainnointi. KvaliMOTV. Viitattu 4.11.2024. https://www.fsd.tuni.fi/menetelmaopetus/kvali/L6_4_2.html

Sertifioitu testaaja. Perustason sertifikaattisisältö. 2023. FISTB. Viitattu 2.11.2024. https://fistb.fi/wp-content/uploads/sites/30/2023/09/Kaannos-20230901.pdf

SmartBear. 2024. Test Automation Best Practices. Viitattu 3.11.2024. https://smartbear.com/learn/automated-testing/best-practices-for-automation/

Spillner, A. & Linz, T. 2021. Software Testing Foundations. A Study Guide for the Certified Tester Exam. Foundation Level. ISTQB Compliant. E-kirja. Saksa: dpunkt.verlag.

Umar, M. A. 2020. A Study of Software Testing: Categories, Levels, Techniques, and Types. Viitattu 28.8.2024. DOI:10.36227/techrxiv.12578714.v1

Valli, R. 2018. Ikkunoita tutkimusmetodeihin 1. E-kirja. Jyväskylä: PS-kustannus.

Xray. 2023. Test Process. Viitattu 10.1.2024. https://docs.getxray.app/display/XRAY-CLOUD/Test+Process

Xray. 2024a. Test Plan. Viitattu 28.7.2024. https://docs.getxray.app/display/XRAY-CLOUD/Test+Plan.

Xray. 2024b. Test Set. Viitattu 4.8.2024. https://docs.getxray.app/display/XRAY-CLOUD/Test+Set

Xray. 2024c. Precondition. Viitattu 4.8.2024. https://docs.getxray.app/display/XRAY-CLOUD/Precondition

Xray. 2024d. Test. Viitattu 4.8.2024. https://docs.getxray.app/display/XRAYCLOUD/Test

Xray. 2024e. Test Repository. Viitattu 5.8.2024. https://docs.getxray.app/display/XRAY-CLOUD/Test+Repository

Xray. 2024f. Test Execution. Viitattu 7.9.2024. https://docs.getxray.app/display/XRAY-CLOUD/Test+Execution

Xray. 2024g. Test Runs. Viitattu 7.9.2024. https://docs.getxray.app/display/XRAY-CLOUD/Test+Runs

Kuviot



Taulukot