HAMK
Hämeen ammatti-
korkeakoulu

Testiautomaation hyötyjen todentaminen asiakas- ja potilastietojärjestelmän testaamisessa

Ammattikorkeakoulututkinnon opinnäytetyö

Tieto- ja viestintätekniikka, Insinööri (AMK)

Syksy 2024

Mari Jäntti

HAMK
Hämeen ammatti-
korkeakoulu

Tieto- ja viestintätekniikka		Tiivistelmä
Tekijä	Mari Jäntti	Vuosi 2024
Työn nimi	Testiautomaation hyötyjen todentaminen asiakas- ja potilastietojärjestelmän testaamisessa	
Ohjaaja	Joni Järvenpää	

Potilastietojärjestelmän testaamiseen kohdeyrityksessä käytetään sekä manuaalista testausta että testiautomaatiota. Testiautomaatiota ei kuitenkaan ole vielä saatu laajennettua sovelluskehityksen kaikille tiimeille yhtä laajasti käyttöön, ja testiautomaatiokehityksen priorisoimisen tueksi tarvitsee todentaa sen tuomia hyötyjä sekä jatkuvaan kehittämiseen vaadittavia toimenpiteitä. Tämän toiminnallisen opinnäytetyön tavoitteena oli käytännön esimerkein tarkastella ja todentaa näitä hyötyjä sekä tavoitella käytännön toiminnan kehittämistä ja tehostamista.

Tässä työssä kuvataan testiautomaation nykytilaa sekä sen tulevaisuuden tavoitteita kohdeyrityksessä, joiden tueksi testiautomaation hyötyjä pyrittiin todentamaan. Esimerkiksi valittiin työnkulku potilastietojärjestelmästä, jonka mukaan kuvattiin yhden testitapauksen luonti sekä manuaalisesti että automaattisesti, jotta automaatiotestauksen ja manuaalisen testauksen eroja saatiin kuvattua selkeämmin. Testitapausten luonnin vaiheet, työhön kuluneet ajat sekä testien suorittamiseen kuluneet ajat dokumentoitiin ja niitä vertailtiin keskenään.

Käsittelyyn valittiin myös otanta sosiaali- ja päihdehuollon jo olemassa olevista regressiotesteistä ja niiden ajoajoista. Niistä manuaalisten testiajojen aikoja vertailtiin automatisoitujen testiajojen aikoihin. Lähes poikkeuksetta voitiin todentaa, että yksittäisten testien sekä kaikkien testien yhteenlasketut ajoajat olivat manuaaliajoissa moninkertaiset verrattuna automaatiotestien ajoaikoihin.

Loppuun koottiin lisäksi yhteenveto ja erittely automaatiotesteissä esiintyneistä virheistä, joista oli toteutettu määrätyn ajan aikana seuranta ja tuloksia analysoitu.

Johtopäätöksenä todettiin, että vaikka testiautomaation kehittämiseen kuluu alussa enemmän resursseja ja se aiheuttaa hetkellisesti enemmän kuluja, on siihen panostaminen pitkällä tähtäimellä tärkeää ja kannattavaa, sillä järjestelmän laajetessa manuaalinen testaus tulee viemään entistä enemmän aikaa ja täten myös henkilöresursseja.

Avainsanat	Testiautomaatio, regressiotestaus, potilastietojärjestelmä, järjestelmätestaus
Sivut	24 sivua

HAMK
Hämeen ammatti-
korkeakoulu

Information and Communication Technology — Abstract

Author Mari Jäntti — Year 2024

Subject Verifying the Benefits of Test Automation in Testing the Patient Information System

Supervisors Joni Järvenpää

The goal of this functional thesis was to examine and verify benefits of test automation with practical examples and to aim for the development and enhancement of practical operations. Testing the patient information system of the target company is done using both manual testing and test automation. However, test automation has not yet been extended to all application development teams, and to support the prioritization of test automation development, it is necessary to verify the benefits it brings, and the measures required for continuous development. This work describes the current status of test automation and its future goals in the target company.

As an example, a workflow from the system was chosen, according to which the creation of one test case was described both manually and automatically, so that the differences between automated testing and manual testing could be described more clearly. The stages of creating test cases, the time spent on the work and the time spent on executing the tests were documented and compared.

A sample of the already existing regression tests of social care and substance abuse care and their executing times were also selected for processing. The times of manual test executions were compared to the times of automated test executions. Almost without exception, it could be verified that the combined executing times of individual tests and all tests were many times higher in manual tests compared to the executing times of automation tests. This work also includes a summary and breakdown of the errors that occurred in the automation tests, which had been monitored during the specified period and the results analyzed.

As a conclusion, it was stated that although the development of test automation takes more resources at the beginning and temporarily causes more costs, investing in it is important and profitable in the long run, because as the system expands, manual testing will take even more time and thus also more human resources.

Keywords Test automation, regression testing, patient information system, system testing

Pages 24 pages

Sisällys

Kuvat, taulukot ja kaavat

1 Johdanto

Ohjelmistojen testaus on yleisimmin käytetty menetelmä laadunvarmistukseen ja laadunvalvontaan ohjelmistokehitysorganisaatioissa sekä tärkeä osa ohjelmistonkehitysprosessia. Testauksen avulla on mahdollista saada tietoa ohjelmistotuotteen todellisen ja vaaditun toiminnan eroista. Testauksen kustannuksia ja aikaa voidaan vähentää testiautomaatiolla, jossa testin suorittaminen tapahtuu automaatiotyökalun avulla. (Wiklund ym., 2017)

Tämä opinnäytetyö on suunniteltu tehtäväksi Oy Apotti Ab:lle. Oy Apotti Ab syntyi vuonna 2015 kuntien ja HUSin yhteisestä tahdosta rakentaa maailman ensimmäinen sosiaali- ja terveydenhuollon yhteinen tieto- ja toiminnanohjausjärjestelmä. Apotin tavoitteena on kehittää sosiaali- ja terveydenhuollon palveluita ja yhtenäistää toimintatapoja kustannusten hillitsemiseksi. Sosiaali- ja terveydenhuollon ammattilaisilla on Apotti-järjestelmän myötä käytössään asiakas- ja potilastiedot reaaliaikaisesti palvelu- ja hoitopaikasta riippumatta. (Apotti, 2023)

Apotti- asiakas- ja potilastietojärjestelmän testaamiseen käytetään sekä manuaalista testausta että testiautomaatiota. Testiautomaatiota ei ole kuitenkaan vielä saatu laajennettua käyttöön jokaiseen sovelluskehityksen tiimiin erinäisten haasteiden vuoksi: testiautomaatiota ei ole priorisoitu tarpeeksi eikä yhtiötasolla hahmoteta testauksen vaatiman työmäärän suuruutta, eikä näin myöskään testiautomaation välttämättömyyttä. Näiden haasteiden takia muut järjestelmän kehitystyöt ovat toistuvasti ajaneet testiautomaation ohi. Tästä syystä sovelluskehityksen tiimeille on sovittu yhteiset tavoitteet vuodelle 2024 testiautomaation kehityksen edistämiseksi. Kehityksen priorisoimisen tueksi tarvitsee todentaa testiautomaatiosta saatavia hyötyjä sekä jatkuvaan kehittämiseen vaadittavia toimenpiteitä. Valitut tutkimuskysymykset työn toteuttamisen tueksi ovat:

- Missä testiautomaation lisäarvo näkyy, kun testataan asiakas- ja potilastietojärjestelmää siihen liittyvine integraatiorajapintoineen?
- Mitkä ovat testiautomaation käytön haasteet kohdeyrityksessä?
- Miten paljon jatkuvaa ylläpitotyötä automaatiotestit vaativat?

Opinnäytetyön tekijä työskentelee kohdeyrityksessä sosiaalihuollon avopalveluiden tiimissä sovelluskehittäjänä, ja on oman tiiminsä toinen testiautomaatiovastaava.

2 Ohjelmistotestaus ja työkalut

Ohjelmistotestaus on tapa varmistaa ohjelmiston virheettömyys sekä vastaako tuote odotettuja vaatimuksia. Se sisältää ohjelmiston ja järjestelmän komponenttien suorittamisen yhden tai useamman ominaisuuden arvioimiseksi. Ohjelmistotestauksen tarkoituksena on tunnistaa virheet, puutteet tai puuttuvat vaatimukset verrattuna todellisiin vaatimuksiin. (Okezie ym., 2019) Tyypillisesti testaus luokitellaan kolmeen kategoriaan: Toiminnallinen testaus, ei-toiminnallinen testaus tai suorituskykytestaus sekä ylläpito, johon kuuluu esimerkiksi regressiotestaus. Ohjelmistotestausta voidaan suorittaa manuaalisesti tai sitä voidaan automatisoida testiautomaation avulla. (Geeks for Geeks, 2024)

2.1 Testiautomaatio

Automaattista testausta eli testiautomaatiota voidaan käyttää ohjelmistokehityksessä testausprosessin tehokkuuden ja hyötysuhteen parantamiseen. Testiautomaatiossa testien suorittamiseen käytetään automaatiotyökaluja. Prosessin nopeuttaminen on liikevaihdon ja kilpailuedun kannalta tärkeää, mutta ohjelmiston laadusta ei saa tinkiä. Koska markkinat vaativat jatkuvasti monimutkaisempien ohjelmistojen nopeampaa julkaisua, testiautomaatio on ainoa tapa säilyttää luottamus laatuun ja noudattaa tiukkaa julkaisuaikataulua. (Katalon, n.d.)

Yleensä automaattista testausta on kahta luokkaa: toiminnallista ja ei-toiminnallista. Toiminnalliset testit varmistavat, että ohjelmiston toiminnot toimivat vaatimusten mukaisesti. Ei-toiminnallinen testaus varmistaa niiden vaatimusten täyttymistä, jotka eivät ole suoraan kriittisiä ohjelmiston toimivuudelle, mutta vaikuttavat käyttäjäkokemukseen, esimerkiksi ohjelmiston suorituskyky, skaalautuvuus, tietoturva sekä käytettävyys. (Codecademy, 2021)

Vaikka testiautomaatio vaatii testaajan ylläpitämään testitapauksia, se auttaa viime kädessä parantamaan ohjelmiston laatua, testauksen kattavuutta ja skaalautuvuutta. Automatisoituja testejä on mahdollisuus ajaa enemmän samassa ajassa ja niiden suorittaminen on nopeampaa. Myös ihmisen tekemien virheiden määrä pienenee ja testien tarkkuus paranee.

Kaikkea testausta ei kuitenkaan ole tarkoitus automatisoida, eikä se olisi kannattavaakaan. Ohjelman toiminnot, jotka saattavat muuttua usein, kannattaa testata manuaalisesti, sillä automaatiotestien jatkuva päivittäminen heikentäisi testauksen tehokkuutta. Myös ohjelman uudet toiminnot kannattaa alussa testata manuaalisesti, mikäli muutoksia tai korjauksia vielä tehdään. Lyhyet ja harvemmin ajettavat testit kannattaa myös jättää manuaalisesti testattaviksi. (Gillis, 2023)

2.2 Mustalaatikkotestaus

Mustalaatikkotestaus (Black box testing) on ohjelmistotestausmenetelmä, jossa ohjelmiston toimintoja testataan ilman tietoa sisäisestä koodirakenteesta, toteutuksen yksityiskohdista tai sisäisistä poluista. Mustalaatikkotestaus keskittyy pääasiassa ohjelmistoon syötettyyn tietoon ja ulostuloihin ja se perustuu täysin ohjelmiston määrityksiin ja vaatimuksiin. (Hamilton, 2024)

Mustalaatikkotestauksen avulla voidaan tunnistaa, kuinka ohjelmisto reagoi odotettuihin ja odottamattomiin käyttäjän suorittamiin toimintoihin sekä sen vasteaikaa, käytettävyysongelmia ja luotettavuusongelmia. Ensin täytyy tarkastella järjestelmän vaatimuksia ja spesifikaatioita. Sen jälkeen valitaan kelvolliset syötteet ja niille odotetut ulostulot testiskenaarioon. Testitapaukset rakennetaan ja tämän jälkeen suoritetaan. Testituloksien ulostuloja verrataan odotettuihin ulostuloihin ja mikäli vikoja on ilmennyt, ne korjataan ja testit ajetaan uudelleen. (Hamilton, 2024)

2.3 Regressiotestaus

Regressiotestaus on ohjelmistotestausta, jolla varmistetaan, että viimeisimmät ohjelmiston tai koodin muutokset eivät ole vaikuttaneet haitallisesti ohjelmiston jo olemassa oleviin ominaisuuksiin. Kyseessä on valikoima jo suoritettuja testitapauksia, jotka suoritetaan uudelleen sen varmistamiseksi, että nykyiset toiminnot toimivat kuten niiden pitää. (Hamilton, 2024)

Ilman regressiotestausta pienetkin muutokset ohjelmistossa voivat johtaa kalliisiin virheisiin. Siksi se on järjestelmällinen käytäntö, joka auttaa ylläpitämään ohjelmiston laatua. Regressiotestit voidaan suorittaa manuaalisesti pienissä projekteissa, mutta useimmissa tapauksissa testisarjan toistaminen joka kerta, kun testattavan ohjelmiston päivitys tehdään,

on liian aikaa vievää ja monimutkaista, joten regressiotestit kannattaa automatisoida, mikäli mahdollista. (Hamilton, 2024)

2.4 Meliora Testlab

Meliora Testlab tai lyhyemmin Testlab on responsiivisella käyttöliittymällä varustettu selainpohjainen testinhallintatyökalu, jossa testaus perustuu määriteltyjen testitapausten suorittamiseen. Sen avulla on helppo hallita mitä testataan, mitä on testattu, missä korjattavat ongelmakohdat ovat ja mitä täytyy testata korjausten jälkeen. Testitapaukset ryhmitellään testiajoihin. Testit voidaan myös linkittää vaatimuksiin tai käyttötapauksiin, jotta voidaan seurata, mitä ominaisuuksia on testattu. Testauksen aikana löydettyjä ongelmia voidaan seurata ja niistä voidaan kirjata esimerkiksi havaintoja. (Software Testing Help, 2024)

2.5 Ranorex Studio

Ranorex Studiolla voidaan rakentaa automaatiotestejä ilman ohjelmointiosaamista sen tallennus ja toisto -toiminnon avulla (Record and Replay). Ranorex Studion avulla testaajat voivat tallentaa sovelluksen käytön askeleet ja niihin liittyvät testausvaiheet ja toistaa ne todentaakseen sovelluksen toimivuutta. Käyttäjät voivat suorittaa toimintoja manuaalisesti verkko- tai mobiilisovelluksessa, jotka nauhoitetaan ja tallennetaan testinä. Nauhoitetun testin saa suoritettua napin painalluksella uudelleen. (Ranorex, n.d.)

Ranorex Studiolla saa muokattua tallennettuja toimintoja helposti, lisättyä tekstin tai kuvien tarkistuksia, asetettua parametriarvoja sekä rakennettua data- tai avainsanalähtöisiä testejä. Vaikka Ranorex Studion peruskäyttö ei vaadi ohjelmointiosaamista, se tukee myös vakio-ohjelmointikieliä C# ja VB.NET. Tallenteita voi muokata tai mukautettuja testejä luoda kokonaan käyttäjäkoodissa. (Ranorex, n.d.)

Kohdeyrityksen järjestelmää testattaessa nauhoitettuja tallenteita on aina välttämätöntä muokata ja viimeistellä, jotta testi toimisi mahdollisimman luotettavasti jatkossa, sekä olisi mahdollisimman ylläpidettävä. Tärkeimpinä muokkauksina tallenteisiin on manuaalisesti lisättävä viiveitä vakauden takaamiseksi, koska testattava ohjelmisto ei aina toimi samalla nopeudella kuin testiä luodessa. Lisäksi elementtien polkuja on muokattava mahdollisimman

geneerisiksi, sillä liian tarkat polut voivat aiheuttaa nopeasti ongelmia testiajoissa, etenkin järjestelmän muuttuessa. Tämä tekee testeistä myös ylläpidettävämpiä.

2.6 Jenkins

Jenkins on Java-pohjainen avoimen lähdekoodin automaatioalusta, joka sisältää jatkuvaan integrointiin suunniteltuja laajennuksia. Sitä käytetään ohjelmistoprojektien luomiseen ja testaamiseen. Sen avulla voi myös julkaista ohjelmistoa jatkuvasti käyttämällä erilaisia testaus- ja käyttöönottomenetelmiä. Organisaatiot voivat käyttää Jenkinsiä ohjelmistokehitysprosessin automatisointiin ja nopeuttamiseen. Jenkins sisältää useita kehitystyön elinkaaren aikaisia toimintoja, kuten rakentamisen, dokumentoinnin, testauksen, paketoinnin, käyttöönoton, staattisen analyysin ja paljon muuta. (BasuMallick, 2022)

Kohdeyrityksessä Jenkinsin käyttö on isossa roolissa osana testiautomaatiota. Sillä hallinnoidaan automaatiotestien ajojen aikataulutusta ja se helpottaa testauskuorman tasausta eri testiajoservereiden välillä sekä testitulosten seurantaa.

2.7 Git versionhallinta

Versionhallinta tarkoittaa ohjelmistokoodin muutosten seurantaa ja hallintaa. Versionhallintajärjestelmät ovat ohjelmistotyökaluja, jotka auttavat hallitsemaan lähdekoodin muutoksia. Kun kehitysympäristöt ovat laajentuneet, versionhallintajärjestelmät auttavat ohjelmistotiimiä työskentelemään nopeammin ja ketterämmin. Ne auttavat myös vähentämään kehitysaikaa ja lisäämään onnistuneita käyttöönottoja. (Atlassian, n.d.)

Tänä päivänä ylivoimaisesti eniten käytetty versionhallintajärjestelmä on Git. Se on laadukas ja laajasti tuettu avoimen lähdekoodin projekti. Projektin ylläpitäjät ovat osoittaneet tasapainoista harkintaa ja lähestymistapaa käyttäjien pitkän aikavälin tarpeiden täyttämiseen säännöllisillä julkaisuilla, jotka parantavat käytettävyyttä ja toimivuutta. (Atlassian, n.d.)

Myös kohdeyrityksen automaatiotestien hallinta ja ylläpito on helpompaa versiohallintatyökalua käyttämällä. Sen avulla voidaan helposti erottaa muutoksen alla oleva työ päivittäin ajettavasta testien päähaarasta erilliseen versiohallinnan sivuhaaraan. Tällöin

tehtyjä muutoksia pystyy paremmin seuraamaan ja kontrolloimaan eivätkä niiden kehityksen aikaiset ongelmat näy päivittäin ajettavissa testeissä.

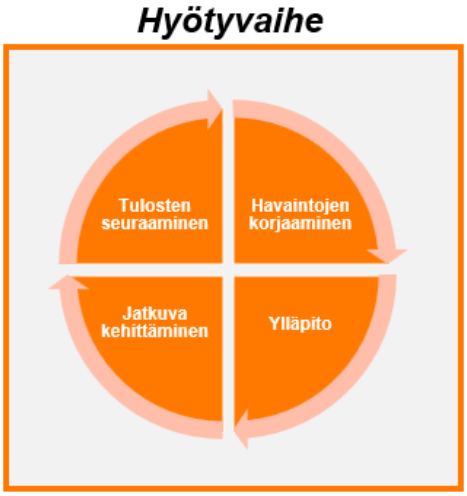
3 Kehittämistyön tavoitteet ja menetelmät

Kohdeyrityksen testiautomaatiokehityksen lyhyen tähtäimen tavoitteena on päästä tilanteeseen, jossa jokaisessa tiimissä on testiautomaatio-osaamista, ymmärrystä sen hyödyistä ja tiimeissä sitoudutaan testiautomaation jatkuvaan kehittämiseen ja ylläpitämiseen. Jotta näihin tavoitteisiin päästään, täytyy tiimeissä toteuttaa testiautomaatioon ja sen työkaluihin perehtyminen, testiautomaatiotyön suunnittelu sekä itse automaatiotestien rakentaminen. Tämä on valmistelevaa vaihetta, jonka kautta saavutetaan pitkän tähtäimen tavoitteet.

Kuva 1. Testiautomaatiokehityksen valmisteleva vaihe, Apotti.

Pitkän tähtäimen tavoitteena saavutetaan niin sanottu hyötyvaihe, jolloin automaatiotestit ovat osana ketterää kehitystä. Tämä vähentää manuaaliseen, toistuvaan testaukseen kuluvaa työaikaa, jolloin toiminta tehostuu. Testaustulosten jatkuva seuraaminen mahdollistaa ongelmien ja virheiden paikantamisen nopeasti eivätkä ne pääse etenemään loppukäyttäjille. Välitön palaute parantaa myös tiimin osaamista ja ymmärrystä tuotteesta. Testauksen laatu paranee, jolloin myös tuotteen laatu paranee.

Kuva 2. Hyötyvaihe kuvattuna, Apotti



Tämä opinnäytetyö toteutetaan toiminnallisena työelämän kehittämistyönä, jonka tavoitteena on todentaa ja selkeyttää ohjelmistotestauksen automatisoinnista saatavia lisähyötyjä käytännön esimerkein sekä arvioida testiautomaation jatkokehitystä. Työn tarkoituksena on olla tukemassa testiautomaation kehityksen edistämistä ja sen priorisoimista, jotta testiautomaatiokehityksen pitkän tähtäimen tavoitteisiin päästäisiin jokaisen sovelluskehityksen tiimin osalta.

Työssä perehdytään testauksessa käytettäviin testaustyökaluihin ja menetelmiin, testitapausten luontiin ja niiden suorittamiseen sekä testitapausten ylläpitoon. Järjestelmän testaamista sekä testaamiseen ja testien ylläpitoon käytettyä työmäärää ja -aikaa tarkastellaan sekä manuaalisen testauksen että testiautomaation näkökulmista. Työssä tarkastellaan myös testiautomaation haasteita asiakas- ja potilastietojärjestelmän testaamisen näkökulmasta ja pohditaan mahdollisia tulevaisuuden kehityskohteita.

Työssä tarkastellaan sosiaalihuollon, päihdehuollon sekä kotihoidon manuaalisesti ajettujen sekä automatisoitujen regressiotestien ajoaikoja. Testien ylläpitotarpeiden seuranta on aloitettu ja esiin nousevat päivitystarpeet sekä niiden juurisyyt dokumentoidaan. Lisäksi toteutetaan esimerkki regressiotestitapauksen luomisesta ja suorittamisesta ensin Meliora Testlabilla, jonka jälkeen sama testitapaus automatisoidaan ja ajetaan Ranorex Studiolla.

4 Projektin toteutus

Tässä luvussa kuvataan kohdeyrityksen järjestelmän regressiotestauksen testaustavat, testitapauksen valintaprosessi sekä testitapauksen luonti ja suoritus. Vertauskohteeksi automatisoitava testitapaus toteutetaan ensin manuaalisesti, johon kulunut työaika sekä testin ajoaika myös dokumentoidaan. Tuloksia tarkastellaan ja niistä kootaan lopuksi yhteenveto. Lopussa on lisäksi esiteltynä yhteenveto testitapauksissa seurantajakson aikana ilmenneistä päivitystarpeista.

4.1 Taustaa

Ennen testiautomaation käyttöönottoa kohdeyrityksessä järjestelmän testaus on suoritettu ainoastaan manuaalisesti käyttämällä Meliora Testlabia. Testlabiin on luotu testitapaukset järjestelmän eri työnkuluista. Testitapauksen askeleet sisältävät tarkat kuvaukset siitä, mitä järjestelmässä tarvitsee tehdä ja mitä lopputuloksia askeleissa suoritettavista toiminnoista odotetaan. Testaajat suorittavat järjestelmässä manuaalisesti testien askeleet ja lopuksi kirjaavat testit läpäistyiksi tai epäonnistuneiksi, jos testauksessa on ilmennyt virheitä. Virheistä kirjataan havainnot, jotka menevät yleensä korjattavaksi eri henkilöille tai tiimeille, kuin itse testaajille. Korjausten jälkeen testit suoritetaan uudelleen. Sama toistuu niin kauan, että testit on saatu hyväksytysti läpäistyä.

Järjestelmän laajentuessa ja toiminnallisuuksien lisääntyessä manuaalinen testaus tulee viemään entistä enemmän aikaa. Järjestelmän regressiotestaus on jatkuvaa työtä, sillä käyttöliittymäpäivityksiä tulee kerran kuukaudessa ja versiopäivityksiä kaksi kertaa vuodessa. Tästä syystä järjestelmän regressiotestejä halutaan automatisoida mahdollisimman paljon. Regressiotestien ajaminen myös laajempien päivitysten ulkopuolella auttaisi kehitysmuutosten aiheuttamien mahdollisten vikatilojen ja sivuvaikutusten nopeammassa kiinnisaamisessa ja paikannuksessa.

4.2 Automatisoitavan testitapauksen työnkulun valinta

Regressiotestauksen automatisointiin on yleensä tarkoituksenmukaista valita sellaiset testitapaukset tai työnkulut, joilla testataan järjestelmän yleisimpiä ja tärkeimpiä toiminnallisuuksia, joiden toimimattomuus aiheuttaisi suuria tai jopa käyttöä estäviä haittoja.

Laajassa järjestelmässä, jota käytetään moniin eri tarkoituksiin, voi tällaisia testitapauksia olla helposti jopa satoja.

Automatisoitavan työnkulun valinnassa täytyy kiinnittää huomiota työnkulun monimutkaisuuteen. Liian monimutkaisia tai pitkiä työnkulkuja voi olla hankalaa tai liian aikaa vievää automatisoida. Pitkiä, monivaiheisia testejä voi olla myös vaikeaa saada täysin stabiileiksi niin, että testiajot eivät epäonnistuisi usein. Yksinkertainen automatisoitu testi voi antaa enemmän lisäarvoa kuin monimutkainen testi, joka on virheherkempi muutoksille. Tästäkin syystä pidemmät testitapaukset kannattaa jakaa pienemmiksi. Myös liian spesifit klikkaukset tai kompleksiset validointiaskeleet voivat aiheuttaa turhia virheitä testiajoihin sekä tehdä testien ylläpitämisestä työlästä, jolloin automatisoinnista saatavat hyödyt jäävät vähäisiksi tai kokonaan saamatta. Myöskään järjestelmään lisättyjen, uusien ominaisuuksien testaamista ei ole kannattavaa lähteä heti automatisoimaan, sillä niihin saattaa lyhyen ajan sisällä tulla vielä päivityksiä.

Esimerkkitestitapausta varten valittiin sosiaalihuollon lähetteen hyväksynnän työnkulku. Työnkulussa tarkistetaan, löytyykö oikea lähete oikeasta työjonosta, jonka jälkeen lähete hyväksytään. Sama työnkulku pätee useampaan eri ilmoitukseen, määräykseen ja hakemukseen, joten testiä voi käyttää useampaan testaukseen. Tämä testitapaus on suunniteltu ajettavaksi sen jälkeen, kun testipotilaalle on ensin luotu hyväksyntää odottava lähete.

4.3 Testitapausten luonti

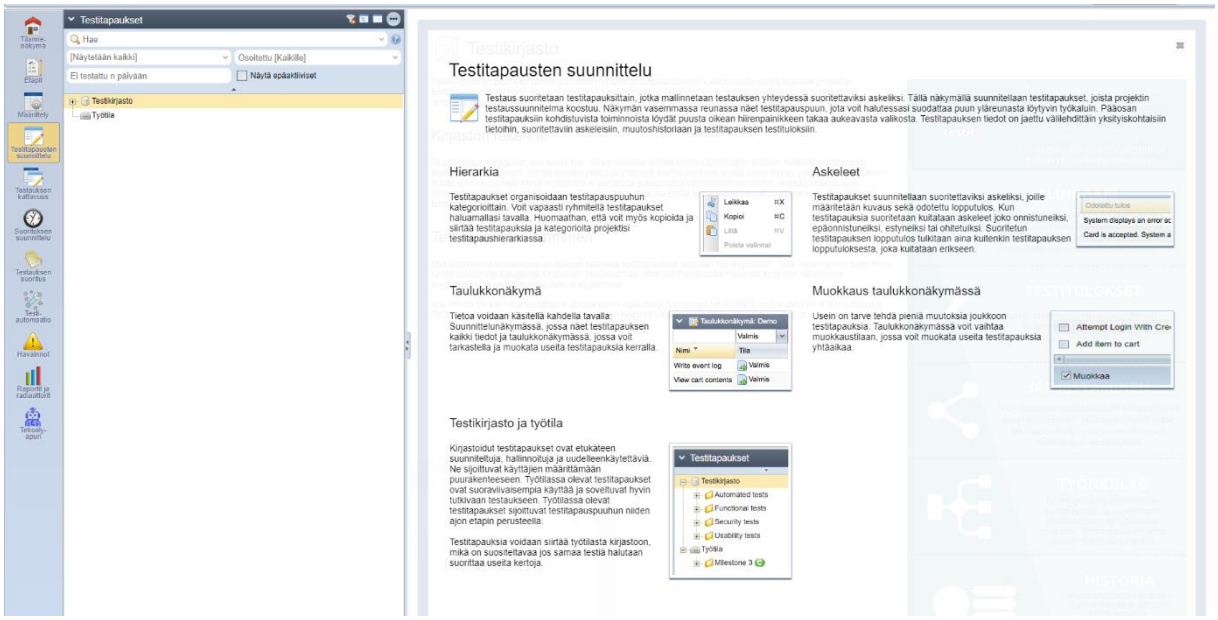
Jotta eroja manuaalisen testauksen sekä automaatiotestien välillä kohdeyrityksessä saataisiin konkreettisemmin esiteltyä, sisällytetään toteutukseen myös manuaalisen testitapauksen luonti ja suoritus. Joidenkin automatisoitujen testien lähteenä on käytetty vanhoja manuaalisia testitapauksia, joten molempiin esimerkkeihin käytetään samaa työnkulkua.

4.3.1 Manuaalisen testitapauksen luonti

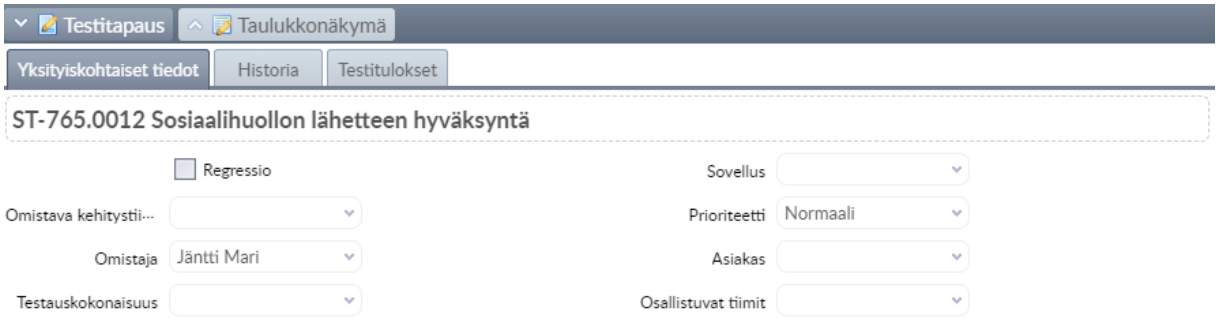
Manuaalisesti ajettava testitapaus luodaan Meliora Testlabiin oikean kansiorakenteen alle. Testitapaus nimetään selkeästi ja työnkulkua kuvaavasti. Testitapaukseen luodaan askeleet,

joihin kirjoitetaan työnkulun mukaisessa järjestyksessä järjestelmässä tehtävät toimenpiteet mahdollisimman selkeästi ja tarpeeksi yksityiskohtaisesti, jotta testin suorittaminen on sujuvaa myös sellaiselle testaajalle, jolle kyseinen työnkulku ei ole entuudestaan tuttu. Esiehtoihin kirjataan tarpeen vaatiessa lisätietoja mahdollisista valmisteluvaiheista, jotka täytyy olla suoritettuna ennen kyseisen testin aloittamista sekä mainita testissä käytettävästä testiasiakkaasta.

Kuva 3. Meliora Testlabin käyttöliittymä.



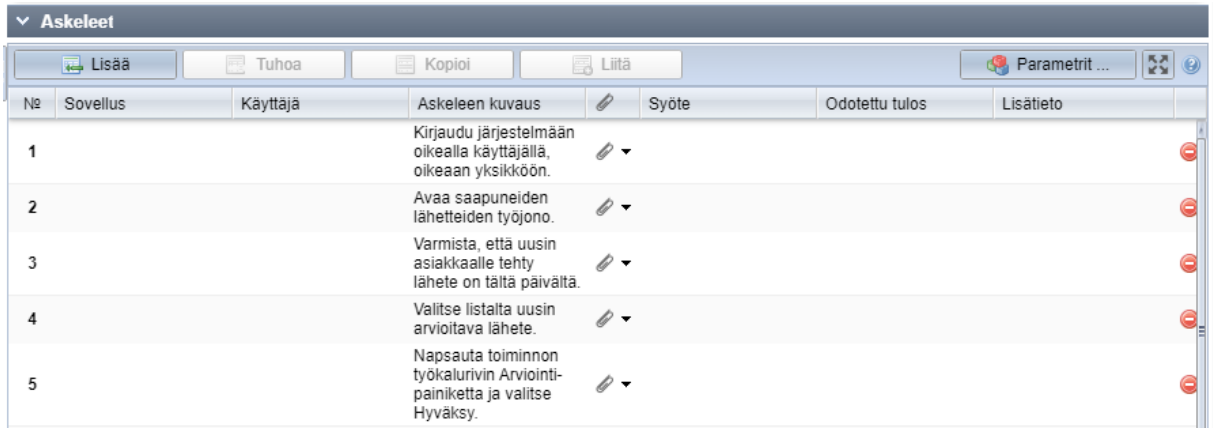
Kuva 4. Testitapauksen tiedot.



Lähetteen hyväksynnän työnkulun testitapauksen askeleiden luonti aloitettiin järjestelmään kirjautumisesta. Seuraava askel ohjeistaa käyttäjää avaamaan saapuneiden lähetteiden työjonon, josta hyväksyttävää lähetettä etsitään. Työjonolta odotetaan löytyvän aiemmin

toisen testitapauksen ajossa luotu, hyväksyntää odottava lähete. Testitapauksen askel ohjeistaa käyttäjää valitsemaan listalta uusimman lähetteen ja varmistamaan, että lähete on tänä päivänä luotu. Tällä tarkistuksella vahvistetaan myös, että edellisen testin suoritus on ollut onnistunut. Jos valitulta testipotilaalta löytyy saman päivän lähete, se hyväksytään. Lähetteen hyväksyntään kuuluu myös ensiarvion tietojen kirjaus, joiden täyttöön testin viimeiset askeleet ohjeistavat.

Kuva 5. Lähetteen hyväksynnän testitapauksen askeleita Meliora Testlabissa

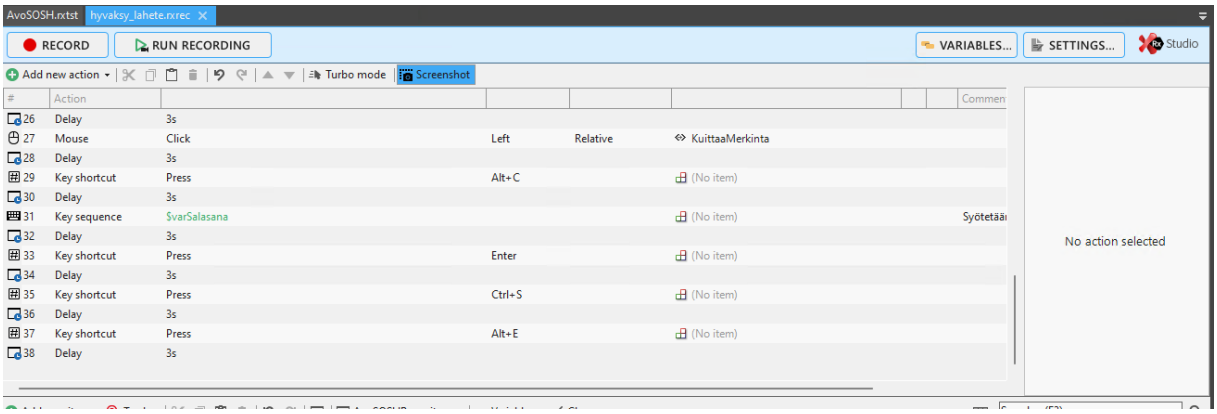


4.3.2 Automatisoidun testitapauksen luonti

Seuraavaksi luodaan automaattisesti suoritettava testitapaus Ranorex Studiolla. Testitapaus luodaan oikean testisalkun (Test Suite) alle. Testitapauksen työnkulun askeleet luodaan nauhoitustoiminnolla (Record). Kun järjestelmä on auki oikeassa kohdassa, josta työnkulkua lähdetään tekemään, nauhoitus laitetaan päälle.

Nauhoitus tallentaa hiiren liikkeet, klikkaukset sekä näppäimistön painallukset. Kun nauhoitus on tehty, tallentuneet askeleet tarkistetaan ja toistotoiminnolla (Replay) nauhoitus ajetaan testin toiminnan varmistamiseksi. Virheellisesti tallentuneet askeleet korjataan ja ylimääräiset, mahdolliset vikapainallukset siivotaan pois. Klikatut elementit tallentuvat testisalkun arkistoon, josta ne ovat käytettävissä myös muissa moduuleissa ja testeissä. Askelten väleihin on tarpeellista asettaa myös stabilointiviiveitä, jotta esimerkiksi järjestelmän lataamisen viiveet eivät aiheuta virheitä testissä. Testitapaukseen saa lisättyä myös validointiaskeleita. Testien askeleisiin saa yhdistettyä koodilla myös taustalla tapahtuvia toimintoja, esimerkiksi päivämäärän tarkistuksen.

Kuva 6. Testitapauksen askeleita Ranorex Studiossa.



Kuva 7. Elementtejä Ranorex Studion testisalkun arkistossa.

Automaatiotestit koostuvat useista moduuleista, jotka sisältävät testissä suoritettavat askeleet. Moduulit kannattaa luoda mahdollisimman yleiskäyttöisiksi, jotta samoja moduuleita voidaan hyödyntää mahdollisimman monissa testeissä. Näin automatisointiprosessi on tehokkaampaa, ja säästytään ylimääräiseltä työltä.

Kun automaatiotesti on valmis ja todettu toimivaksi, testi katselmoidaan ennen lisäystä tuotantoajoihin. Jotta testi menisi katselmoinnista läpi, kaikki testin askeleet on kommentoitava, osiot nimetty selkeästi ja ylimääräiset siivottu pois. On varmistuttava, että data tuodaan testiin hallitusti muuttujien kautta, eikä kovakoodaamalla askeleisiin.

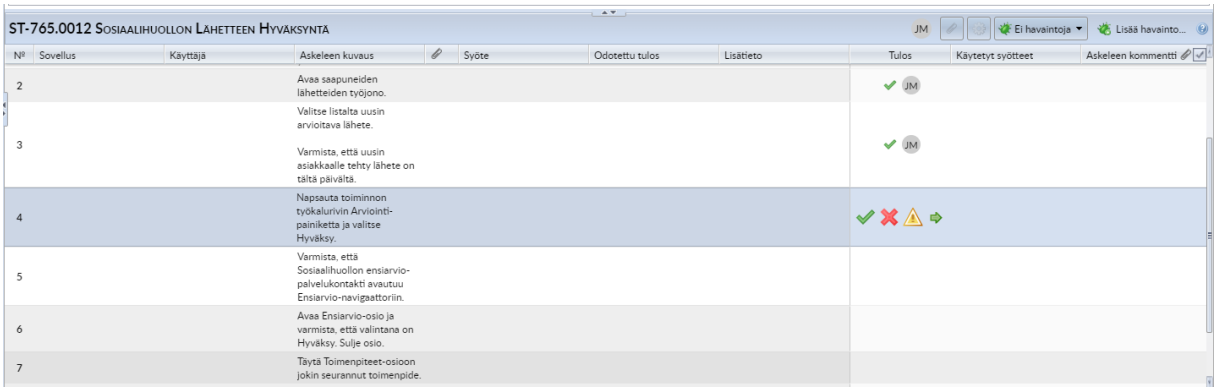
4.4 Testien suoritus

Kun työnkulusta luodut testitapaukset ovat valmiit ja katselmoitu, testit suoritetaan. Tässä luvussa kuvataan sekä manuaali- että automaatiotestien suoritustavat.

4.4.1 Manuaalisen testin suoritus

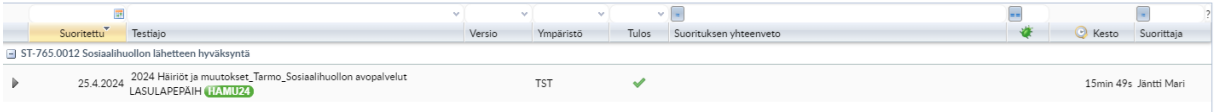
Meliora Testlabilla siirrytään testauksen suorituksen tilaan, jossa testin ajo tapahtuu. Testitapaus lisätään ajoon ja testin askeleet suoritetaan testin mukaisessa järjestyksessä järjestelmässä. Mikäli testin askel saadaan suoritettua vaatimusten mukaisesti, askel merkitään läpäistyksi. Kun testin kaikki askeleet on suoritettu onnistuneesti, testi merkitään valmiiksi. Askeleista tai testin ajosta voi kirjoittaa myös kommentteja. Testin suorittamiseen käytetty aika kirjautuu ajohistoriaan.

Kuva 8. Testitapauksen suoritusta Meliora Testlabissa.



Mikäli jokin testin askel ei tuota haluttua lopputulosta tai askelta ei voi suorittaa hyväksytysti, askel merkitään epäonnistuneeksi ja testin suoritus jää kesken. Epäonnistumisesta kirjataan havainto. Kun havainto on korjattu, testi suoritetaan uudelleen. Jos askeleet tällä kertaa saadaan suoritettua hyväksytysti, testi merkitään läpäistyksi.

Kuva 9. Suoritetun testin tiedot Meliora Testlabissa.

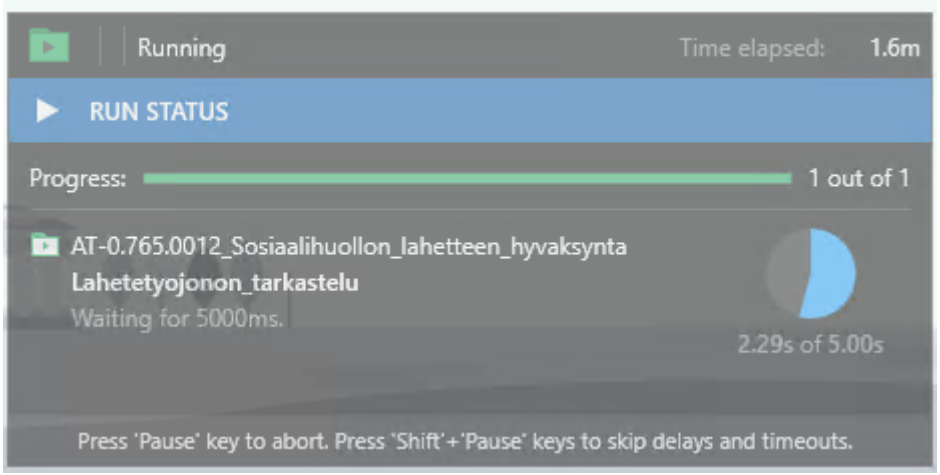


4.4.2 Automaatiotestin suoritus

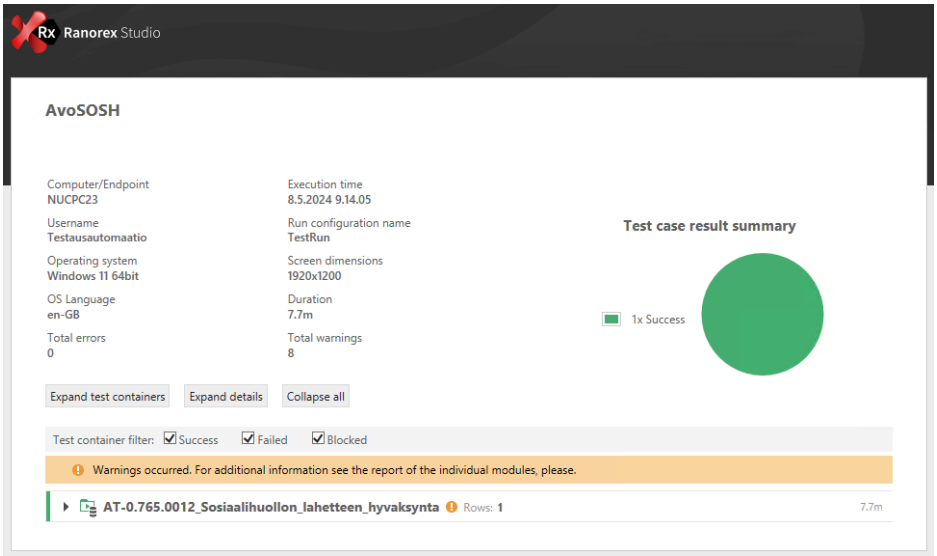
Automatisoitu testi suoritetaan eli ajetaan Ranorex Studiolla. Jenkinsillä hallinnoidaan ajettavia testiajoja, jotka on ajastettu ajettavaksi joka yö. Testiajoja voi käynnistää myös manuaalisesti. Ranorex Studion lokitiedostoihin tallentuu testiajojen raportit. Lokitiedostosta

näkee, onko testi läpäissyt ja mikäli ei, virheilmoituksen sekä tiedon missä kohdassa testiä ajo on epäonnistunut.

Kuva 10. Testiajon statusikkuna.



Kuva 11. Onnistuneesti suoritetun testiajon yhteenvetoikkuna.

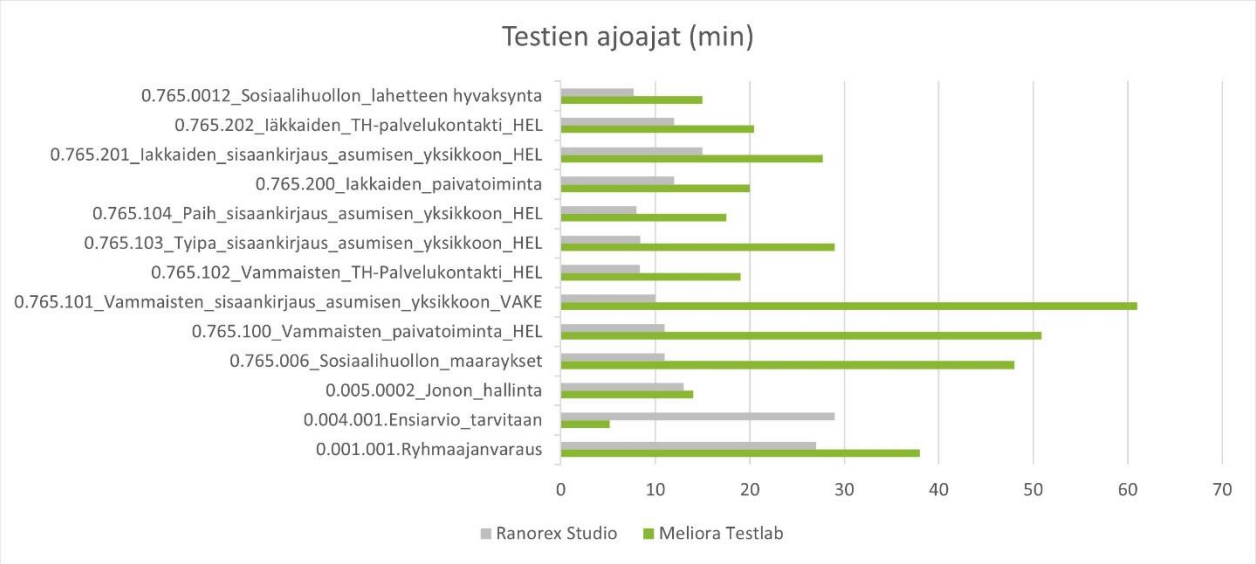


4.5 Tulokset

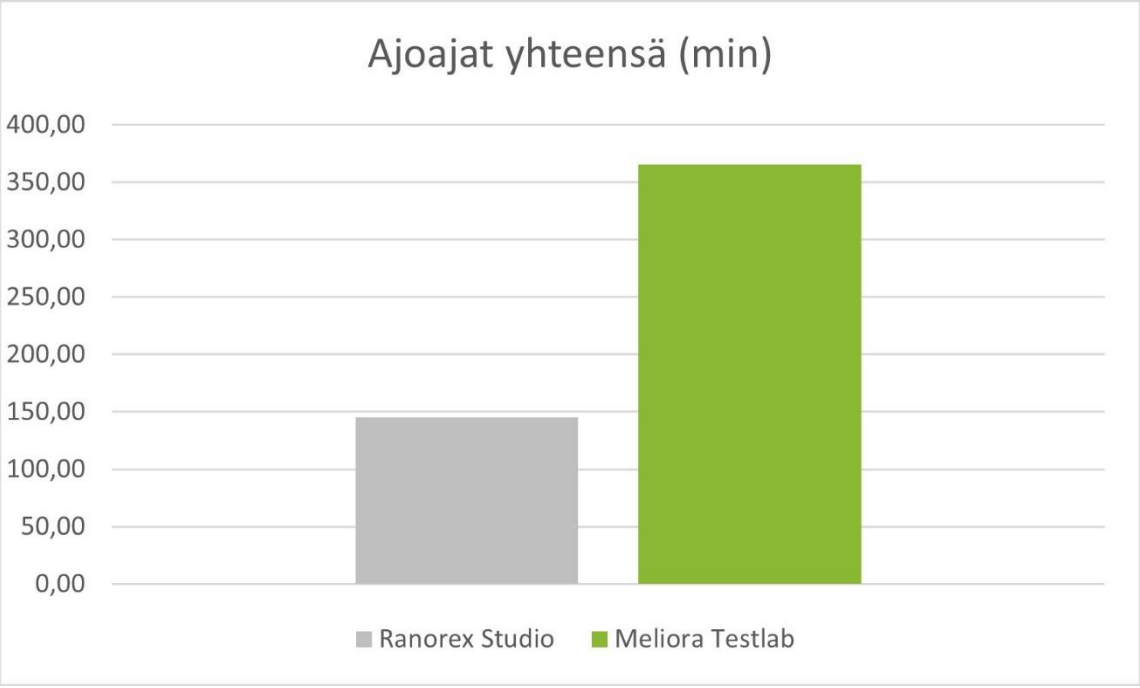
Alla esitellään otantaa sosiaalihuollon sekä päihdehuollon regressiotestien ajoajoista. Jokaisesta testitapauksesta on otettu vertailuun Meliora Testlabilla suoritetun vastaavan tai

lähes vastaavan työnkulun manuaalisen testin ajoaika sekä Ranorex Studiolla suoritetun automaattisen testin ajoaika.

Kuva 12. Sosiaali- ja päihdehuollon regressiotestien ajoajat testeittäin.



Kuva 13. Sosiaali- ja päihdehuollon regressiotestien ajoajat yhteensä.



Testitapausten suorittamiseen kuluneessa ajassa on nähtävillä selkeä ero manuaalisten ja automaatiotestien välillä. Lähes poikkeuksetta testien manuaalinen ajaminen vie

moninkertaisesti aikaa verrattuna automatisoituihin testeihin. Joitain poikkeuksia voi olla, mikäli esimerkiksi työnkulku testissä on pitkä tai automaatiotesti sisältää paljon erilaisia tarkistuksia. Jokaisen manuaalisen testitapauksen ajamiseen tarvitaan myös fyysinen henkilö ajon suorittamaan, joten manuaalinen testaaminen vie myös enemmän resursseja.

Testitapausten luonti manuaalisesti on suoraviivaisempaa ja nopeampaa, koska testitapausten askeleet ja niissä suoritettavat toiminnot kirjoitetaan tekstinä, eivätkä ne sen vuoksi vaadi kuin murto-osan automatisoidun testitapauksen työajasta. Testitapausten automatisointiin menee helposti useampi tunti, etenkin alussa, kun testitapauksia on tehty vähemmän ja työkalujen käyttöä vasta opetellaan. Esimerkkitestitapauksen luonnissa pystyttiin käyttämään jo aiemmin luotua moduulia lähetejonon tarkistukseen, joten testitapauksen valmiiksi saaminen oli jonkin verran nopeampaa, kuin jos testin työnkulku olisi nauhoitettu kokonaan. Testitapauksen automatisointiin käytettyyn työaikaan sisältyi myös testitapauksen viimeistelyyn kuuluvat toimenpiteet eli stabilointiviiveiden asettaminen, testisalkun arkiston siivous, testin osioiden uudelleen nimeäminen, osioiden polkujen siistiminen sekä testitapauksen askelten kommentointi.

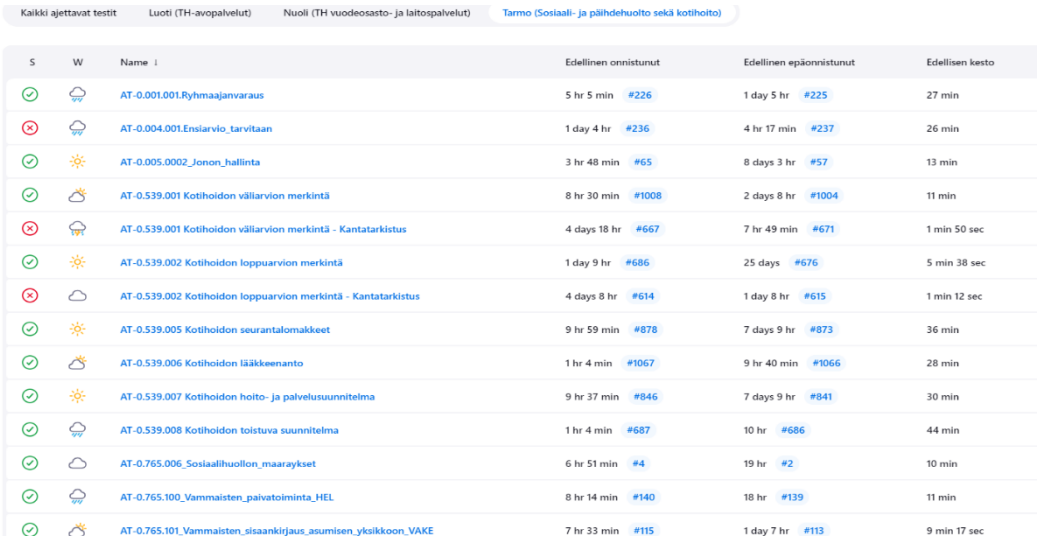
Taulukko 1: Esimerkkitestitapauksen työ- ja ajoajat

Kulunut aika (h)	**Meliora Testlab**	**Ranorex Studio**
Testitapauksen luonti	0,5	4,3
Testitapauksen ajo	0,25	0,13

4.6 Testiajojen seuranta ja ylläpito

Automaatiotestit ajetaan joka yö. Jenkinsin päänäkymästä näkee suoraan jokaisen testiajon tilan, kuinka kauan edellinen testiajo on kestänyt, kuinka kauan edellisestä testiajosta on kulunut ja kuinka kauan edellisestä läpäistystä testiajosta on kulunut. ”Säätila” kuvaa keskiarvoa viimeisen viiden ajon onnistumisista.

Kuva 14. Testiajojen tietoja Jenkinsin käyttöliittymässä.



Kaikki ajettavat testit | Luoti (TH-avopalvelut) | Nuoli (TH vuodeosasto- ja laitospalvelut) | Tarmo (Sosiaali- ja päihdehuolto sekä kotihoito)

S	W	Name ↓	Edellinen onnistunut	Edellinen epäonnistunut	Edellisen kesto
✓		AT-0.001.001.Ryhmaajanvaraus	5 hr 5 min #226	1 day 5 hr #225	27 min
✗		AT-0.004.001.Ensiarvio_tarvitaan	1 day 4 hr #236	4 hr 17 min #237	26 min
✓		AT-0.005.0002_Jonon_hallinta	3 hr 48 min #65	8 days 3 hr #57	13 min
✓		AT-0.539.001 Kotihoidon väliarvion merkintä	8 hr 30 min #1008	2 days 8 hr #1004	11 min
✗		AT-0.539.001 Kotihoidon väliarvion merkintä - Kantatarkistus	4 days 18 hr #667	7 hr 49 min #671	1 min 50 sec
✓		AT-0.539.002 Kotihoidon loppuarvion merkintä	1 day 9 hr #686	25 days #676	5 min 38 sec
✗		AT-0.539.002 Kotihoidon loppuarvion merkintä - Kantatarkistus	4 days 8 hr #614	1 day 8 hr #615	1 min 12 sec
✓		AT-0.539.005 Kotihoidon seurantalomakkeet	9 hr 59 min #878	7 days 9 hr #873	36 min
✓		AT-0.539.006 Kotihoidon lääkkeenanto	1 hr 4 min #1067	9 hr 40 min #1066	28 min
✓		AT-0.539.007 Kotihoidon hoito- ja palvelusuunnitelma	9 hr 37 min #846	7 days 9 hr #841	30 min
✓		AT-0.539.008 Kotihoidon toistuva suunnitelma	1 hr 4 min #687	10 hr #686	44 min
✓		AT-0.765.006_Sosiaalihuollon_maaraykset	6 hr 51 min #4	19 hr #2	10 min
✓		AT-0.765.100_Vammaisten_paivatoiminta_HEL	8 hr 14 min #140	18 hr #139	11 min
✓		AT-0.765.101_Vammaisten_sisaankirjaus_asumisen_yksikkoon_VAKE	7 hr 33 min #115	1 day 7 hr #113	9 min 17 sec

Virheraporteista voi halutessaan asettaa sähköposti-ilmoituksen. Ajojen raporteista näkee tarkemmat tiedot testiajoista. Epäonnistuneiden testiajojen raporteista näkee, missä kohdassa testiä virhe on tapahtunut sekä tarkemman kuvauksen itse virheestä, jolloin ongelman korjaaminen on helpompaa ja nopeampaa.

Kuva 15. Osa virheeseen keskeytyneen testiajon raportista Jenkinsissä.

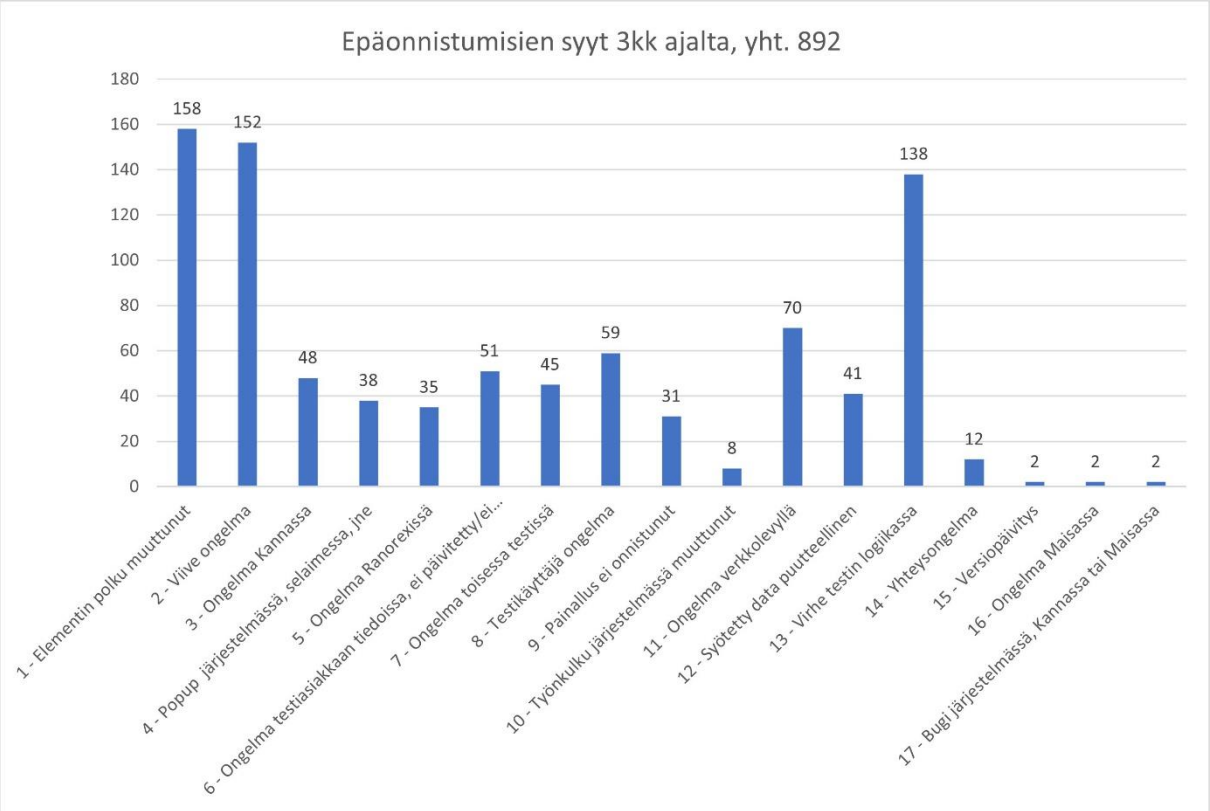
```
$varHakemus = 'lastensuojeluilmoitus'
[2024/04/09 03:26:02.080][Info   ][Keyboard]: Haetaan hakemus
Key sequence from variable '$varHakemus' with focus on 'Hyperspace'.
[2024/04/09 03:26:02.802][Info   ][Delay]: Stabilointi viive
Waiting for 5s.
[2024/04/09 03:26:07.846][Info   ][Keyboard]: Kuitataan haku
Key sequence '{Return}' with focus on 'Hyperspace'.
[2024/04/09 03:26:07.948][Info   ][Delay]: Stabilointiviive
Waiting for 10s.
[2024/04/09 03:26:18.007][Info   ][Mouse]: Hyv►ksyt►►n valittu hakemus
Mouse Left Click item 'Hyperspace.Maaraykset.HyvaksyValinta' at 8;6.
[2024/04/09 03:32:18.118][Info   ][User]: Item 'HyvaksyValinta' was found using Robust
[2024/04/09 03:32:20.351][Error  ][Module]: Failed to find item 'AvoSOSHRepository.Hyp
```

Jenkinsin testituloksia seurataan päivittäin. Mikäli jonkin testin ajo on epäonnistunut, ajoraportti tarkastetaan ja virhe tutkitaan. Mikäli virhe johtuu testistä, testi korjataan

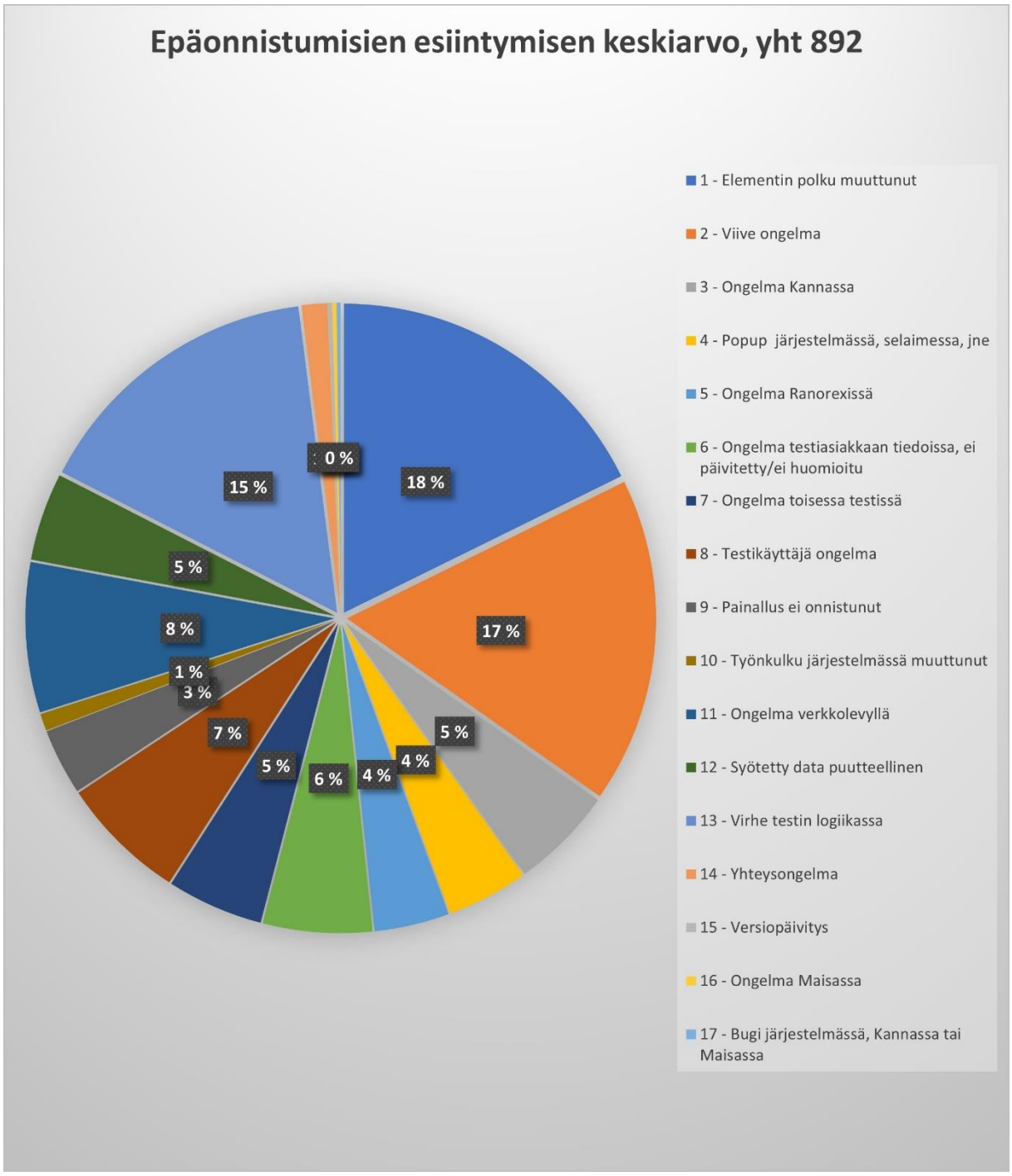
mahdollisimman pian. Mikäli järjestelmän virhe on aiheuttanut testiajon epäonnistumisen, tarkistetaan, onko järjestelmässä ilmennyt korjauksen tarve.

Automaatiotestien tuloksia seurattiin ja kerättiin testausasiantuntijoiden toimesta päivittäin kolmen kuukauden ajan. Tulokset koottiin Jenkinsin ajohistorioista. Tulokset analysoitiin epäonnistuneiden testien lokitiedostoista ja virheet jaettiin 17 eri kategoriaan.

Kuva 16. Testiajojen virheet 3kk:n seurannan ajalta jaettuna kategorioittain, Apotti



Kuva 17. Virhekategorioiden prosentuaaliset osuudet, Apotti



Testien onnistumisprosentin keskiarvo seuranta-ajalta oli 83 %. Päivittäin virheitä oli keskimäärin 11 testissä. Päivittäin ajettujen testien onnistumisprosentin keskiarvo oli 78 %. Kolmen kuukauden aikana virheitä ilmeni yhteensä 892 kpl. Kolme yleisintä seurannan

aikana esiintynyttä syytä (kattavat 51 % ilmenneistä virheistä) olivat elementin muuttunut polku (18 %), ongelma viiveissä (17 %) sekä virhe testin logiikassa (15 %).

Versio- tai muut käyttöliittymäpäivitykset voivat aiheuttaa esimerkiksi jonkin elementin polun muuttumisen niin, ettei elementtiä enää testiä ajaessa löydy ja testi keskeytyy virheeseen. Tämä seuranta aloitettiin versiopäivityksen jälkeen, joka osaltaan vaikutti kyseisten virheiden määrään. Harvoin kuitenkaan työnkulut muuttuvat päivitysten myötä niin radikaalisti, että jokin kokonainen moduuli täytyisi nauhoittaa alusta saakka uudelleen, vaan yleensä yksittäisen elementin päivittäminen riittää.

Virheet testien viiveissä johtuvat enimmäkseen huonosti optimoiduista testeistä, joissa ei joko ole käytetty tarpeeksi viiveitä tai viiveet eivät ole tarpeeksi pitkiä, jolloin järjestelmän tila ei ehdi päivittyä, ennen kuin testi siirtyy seuraavaan askeleeseen. Ajoittain järjestelmässä on esiintynyt myös muuta hitautta, joka on vaikuttanut testiajoihin ja aiheuttanut ajojen epäonnistumisia.

Virheet testien logiikassa taas koostuvat väärällä logiikalla rakennetuista testeistä sekä virheistä, joita on seurannut yhteisten moduulien muokkaamisesta. Yhteisten moduulien muokkauksessa on oltava erityisen tarkka ja muutokset vaativat aina laajempaa testaamista ennen käyttöönottoa.

4.7 Integraatioiden haasteet

Järjestelmän integraatioiden testausta ei toistaiseksi voida täysin toteuttaa testiautomaatiolla. Testiympäristöissä ei ole kaikkia kolmannen osapuolien integraatioita tai olemassa olevat integraatiot eivät ole jatkuvasti päällä, jonka takia testejä ei olisi järkevää ajaa toistuvasti. Osa integraatiotesteistä sisältää myös manuaalisia tarkistuksia järjestelmän ulkopuolelta, jolloin testien automatisointi olisi enemmänkin ylimääräistä työtä. Testiautomaatiota on kuitenkin mahdollista hyödyntää esimerkiksi sanomarakenteiden oikeellisuuden tarkistuksissa.

5 Johtopäätökset ja pohdinta

Siitä huolimatta, että testiautomaation käyttöönotto ja uusien testien luonti veisi alussa enemmän aikaa ja resursseja ja siitä seuraisi hetkellisesti enemmän kustannuksia, on testiautomaatioon investoiminen pitkällä tähtäimellä kannattavaa. Manuaalinen testaus tulee viemään tulevaisuudessa entistä enemmän aikaa, kun järjestelmän toiminnallisuudet laajenevat entisestään. Manuaalisten testien ajoihin tarvitaan aina henkilöresursseja testejä ylläpitävien ja uusia testitapauksia luovien henkilöiden lisäksi. Ja koska automaatiotestit ovat helposti ajettavissa ja tulokset seurattavissa päivittäin, pysyy ohjelmiston laatu parempana, kun ongelmiin on mahdollista puuttua saman tien, jolloin ehkäistään virheiden lisääntymistä.

Automaatiotestit takaavat sen, että järjestelmän testaus tapahtuu aina samalla tavalla. Koska järjestelmä on laaja ja työvaiheita on käytännössä mahdollista tehdä järjestelmässä useammalla eri tavalla, on tärkeää, että järjestelmää testatessa suoritetaan aina sama, oikea työnkulku, jotta vältytään esimerkiksi turhilta havainnoilta ja varmistutaan, ettei mikään oikean työnkulun mahdollinen ongelma jää huomaamatta. Testitapauksiin täytyy sisällyttää työnkulun askeleet mahdollisimman tarkasti. Toisaalta jokaisen mahdollisen virhetilanteen tarkistuksen sisällyttäminen testitapauksiin saattaa tehdä testeistä liian pitkiä ja monimutkaisia. Yksinkertainen automatisoitu testi voi antaa enemmän lisäarvoa, kuin monimutkainen testi, joka on virheherkempi muutoksille. Parhaimman hyödyn saamiseksi testiautomaatiosta on hyvä suunnitella tarkkaan, millaisia työnkulkuja ja testejä on oikeasti kannattavaa automatisoida ja mitä jättää manuaalisesti testattaviksi. Muuten riskinä on, että aikaa ja resursseja kuluu enemmän testien ylläpitoon, kuin itse testaukseen. Pidempiä työnkulkuja kannattaa myös jakaa useammiksi testeiksi, koska pitkistä ja monimutkaisista testeistä voi olla vaikeaa saada toimivia ja vakaita.

Vaikka päivittäisissä testiajoissa ilmenee jonkin verran virheitä, ja testeissä on parantamisen varaa, on kokonaisuus tällä hetkellä suhteellisen stabiili. Testien ylläpitoa helpottaa nauhoitusmoduulien uudelleenkäyttö, jolloin yksittäisen moduulin päivitys riittää, eikä samaa korjausta tarvitse tehdä moneen kertaan. Yhteisten moduulien muokkauksessa täytyy olla kuitenkin huolellinen ja muutokset testata laajemmin, ennen kuin niitä otetaan käyttöön.

Jotta uusien automaatiotestien kehittämisestä saadaan nopeampaa ja tehokkaampaa, testimoduuleita täytyy luoda testikirjastoon tarpeeksi, vähintään järjestelmän perustoiminnoista sekä usein toistuvista toiminnoista. Moduuleista täytyy tehdä

mahdollisimman yleiskäyttöisiä, jotta niitä voidaan sujuvasti hyödyntää uudestaan useampiin eri testeihin, jolloin uusien testien luominen vie vähemmän aikaa. Näin myös testien ylläpito olisi helpompaa, kun samoja päivityksiä ei tarvitse tehdä jokaiseen testitapaukseen erikseen.

Kun testeissä ilmenee virheitä, niihin olisi hyvä pystyä reagoimaan mahdollisimman nopeasti. Testiautomaation kehityssuunnitelmissa on perustaa testiautomaatiosta vastaava ryhmä, jossa on henkilöitä yli tiimirajojen, jolloin jokaisessa tiimissä ei välttämättä olisi enää omaa, määrättyä testiautomaatiovastaavaa. Tämä varmasti edesauttaisi myös sitä, että testeissä ilmenneiden virheiden korjaamiseen löytyisi paremmin aikaa ja resursseja.

Olisi mielenkiintoista analysoida, pystyisikö kehitysympäristössä suoritettavan testiautomaation vaikutusta järjestelmän laatuun tarkastelemaan myös tuotantoympäristön ongelmista raportoitavien häiriötikettien kautta. Tämä vaatisi pidemmän ja perusteellisemman seurannan sekä perehtymisen erilaisiin häiriötiketteihin ja niiden luokituksiin. Tämän työn aikataulujen sekä toteutuksen laajuuden puitteissa kyseistä perehtymistä ja analyysia ei ollut mahdollista toteuttaa. Ehkä tulevaisuudessa tällaistakin seurantaa on mahdollista suorittaa, jotta saadaan lisää konkreettista todennusta testiautomaation hyödyistä.

Lähteet

Apotti (2023).
https://www.apotti.fi/apotti/apotti-yrityksena/

Atlassian (n.d.). *What is version control?*
https://www.atlassian.com/git/tutorials/what-is-version-control

Atlassian (n.d.). *What is Git?*
https://www.atlassian.com/git/tutorials/what-is-git

BasuMallick, C. (2022). *What Is Jenkins? Working, Uses, Pipelines, and Features*, Spiceworks
https://www.spiceworks.com/tech/devops/articles/what-is-jenkins/

Codecademy, T. (2021). *What Is Automation Testing?* Codecademy Blog
https://www.codecademy.com/resources/blog/what-is-automation-testing/

Geeks for Geeks (2024). *Functional Testing – Software Testing*
https://www.geeksforgeeks.org/software-testing-functional-testing/

Gillis, A. (2023). *Automated testing*
https://www.techtarget.com/searchsoftwarequality/definition/automated-software-testing

Hamilton, T. (2024). *What is BLACK Box Testing? Techniques, Types & Example*
https://www.guru99.com/black-box-testing.html

Hamilton, T. (2024). *What is Regression Testing?*
https://www.guru99.com/regression-testing.html

Hamilton, T. (2024). *What is Software Testing?*
https://www.guru99.com/software-testing-introduction-importance.html

Katalon (n.d.). *What is Automation Testing? Definition, How-to, and Best Practices*
https://katalon.com/resources-center/blog/what-is-automation-testing

Okezie, F., Odun-Ayo, I., Bogle, S (2019). *A Critical Analysis of Software Testing Tools* https://iopscience.iop.org/article/10.1088/1742-6596/1378/4/042030/pdf

Ranorex (2024). https://www.ranorex.com/test-automation-tools/

Software Testing Help (2024). *Meliora Testlab Test Management Tool Review* https://www.softwaretestinghelp.com/meliora-testlab-test-management-tool-review/

Test Evolve (n.d.). *Record and Replay Testing vs Scripting: Navigating the Pros and Cons for Your Testing Needs* https://www.testevolve.com/blog/record-and-replay-testing-vs-scripting

Veerappan, K (n.d.). *Measuring the effectiveness of Test Automation,* Zuci Systems blog https://www.zucisystems.com/blog/measuring-the-effectiveness-of-test-automation/

Wiklund, K., Eldh, S., Sundmark, D., Lundqvist, K. (2017). *Impediments for software test automation: A systematic literature review* https://onlinelibrary.wiley.com/doi/full/10.1002/stvr.1639