OULUN AMMATTIKORKEAKOULU

Janne Kanniala

LIHASTEN VÄSYMISEN MITTAAMINEN JA KÄYTTÖLIITTYMÄN AUTOMAATIOTESTAUS

LIHASTEN VÄSYMISEN MITTAAMINEN JA KÄYTTÖLIITTYMÄN AUTOMAATIOTESTAUS

Janne Kanniala
Opinnäytetyö
Kevät 2017
Tietotekniikan koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, hyvinvointiteknologian suuntautumisvaihtoehto

---

Tekijä:Janne Kanniala
Opinnäytetyön nimi: Lihasten väsymisen mittaaminen ja käyttöliittymän automaatiotestaus
Työn ohjaaja: Jukka Jauhiainen, Kaisa Orajärvi, Esa Juntura, Juha Heikkinen ja Johanna Varanka
Työn valmistumislukukausi ja -vuosi: Kevät 2017 Sivumäärä: 54

Vuodesta 2014 lähtien tietotekniikan koulutusohjelmassa on ollut kokeella koosteopinnäytetyöt. Tämä opinnäytetyö koostuu kahdesta osasta. Ensimmäisestä 5 opintopisteen osakokonaisuudesta, joka valmistui vuoden 2015 keväällä ja toisesta 10 opintopisteen osakokonaisuudesta, joka valmistui vuoden 2017 keväällä.

Opinnäytetyön ensimmäisessä osassa tehtiin selvitystyö liittyen lihasten väsymiseen, palautumiseen ja lihaksen väsymiseen liittyviin mittausmenetelmiin. Työn tavoitteena oli selvittää tahdonalaisesti liikutettavien lihasten väsymistä suorituksen aikana ja sen jälkeen.

Opinnäytetyön toinen osa toteutettiin Piimega Oy:lle, joka on oululainen ohjelmistotalo. Työssä käytettiin Piimegan Total Commerce -toiminnanohjausjärjestelmää. Total Commerce on suunniteltu kaupoille, joilla myyntiä tapahtuu niin verkkokaupassa kuin kivijalkamyymälässä. Toiminnanohjausjärjestelmään on rakennettu Magento-verkkokauppa-rajapinta.

Toisen osan tavoitteena oli toteuttaa sellainen testiympäristö, joka pyörittää testitapahtumia mahdollisimman vähällä ylläpidolla. Ympäristön on tarkoitus säästää kehittäjien aikaa testausprosessista. Testausympäristö piti myös automatisoida.

Työssä käytettiin hyväksi Microsoftin käyttöliittymätestaukseen tarkoitettua Coded UI Tests -testityökalua, joka mahdollistaa käyttöliittymätestien nopean tekemisen. Testityökalu vaatii toimiakseen Visual Studioon Enterprise lisenssin. Testausympäristön automatisoinnista vastaa Jenkins. Jenkins on avoimeen lähdekoodiin perustuva Continuous Integration- palvelin, joka ohjaa sille määriteltyjen tehtävien suoritusta. Testiympäristö kommunikoi Jenkinsin kanssa PsExeci:n avustuksella, jonka avulla on mahdollista suorittaa komentorivikehotteita etänä.

Testipalvelimelle asennettiin paikallinen SQL-palvelin, jossa säilytetään testaukseen tarvittavaa dataa. Testipalvelin lähettää sähköpostiviestejä testien lopputuloksista. Testausympäristö saatiin kokonaisuudessaan toteutettua aikataulussa ja se on otettu käyttöön.

Asiasanat: lihasten rasittuminen, palautuminen, käyttöliittymätestaus, automatisoitu testipalvelin

ABSTRACT

Oulu University of Applied Sciences
Degree programme in Information Technology and Telecommunication, Medical Engineering

---

Author: Janne Kanniala
The names of parts of the thesis: Measurement of muscle fatigue, Automated testing for user interface
Supervisor(s):Jukka Jauhiainen, Kaisa Orajärvi, Esa Juntura, Juha Heikkinen ja Johanna Varanka
Term and year when the thesis was submitted: Spring 2017 Number of pages: 54

The thesis consist of two separate parts. This type of method has been experimenting since 2014. The first part of the thesis is five credits and it completed in spring 2015. The second part was completed in the spring of 2017 and has a total of ten credits.

The purpose of the first part was to investigate muscle fatigue, recovery and measurements of fatigue. The aim of the first thesis was to clarify fatigue of skeletal muscle before and after training.

The second part of the thesis was done for Piimega Oy which is a software company from Oulu. This thesis was made for Piimegas Total Commerce- ERP product. The system is designed to meet the needs of modern retail organizations, where transactions take place both in stand-alone stores and in web stores. Total Commerce is a lightweight ERP solution with its Magento e-commerce interface.

The purpose of the second part was to create automated testing environment which is running test cases with minimal maintenance. Environment saves time from the testing process.

Tests are created with Microsoft Coded UI test tool which allows user interface testing. Microsoft Coded UI requires Visual Studio Enterprise license. Jenkins takes care of automation in testing environment. Jenkins is an open source Continuous Integration- server which leads all tasks that are given to it. Testing environment communicates with Jenkins via PsExec. PsExec allows to execute processes on other systems.

Testing environment has its own local database. Database is used for store data which ERP uses. Environment sends e-mails automatically from test results and it has been introduced.

Keywords: muscle fatigue, recovery, user interface testing, automated testing environment

SISÄLLYS

1 JOHDANTO

Tietotekniikan koulutusohjelmassa on vuodesta 2014 lähtien ollut kokeilussa koosteopinnäytetyö. Koosteopinnäytetyössä normaali 15 opintopisteen opinnäytetyö on jaettu kolmeen osaan. Työ oli mahdollista suorittaa kahdessa tai kolmessa osassa. Kahden osan työssä tehtiin ensiksi yksi viiden opintopisteen työ ja yksi kymmenen opintopisteen työ. Ensimmäisessä osassa oli tarkoitus kerätä tietoa haluamastaan aiheesta, jota olisi voitu soveltaa muissa osissa. Tässä työssä ensimmäinen ja toinen osa eivät kuitenkaan liity toisiinsa. Tämän opinnäytetyön osat ovat valmistuneet keväällä 2015 ja keväällä 2017.

Työn ensimmäisessä osassa käsitellään lihasten väsymistä ja työ on pääosin teoriaosuuteen keskittynyt. Työ keskittyy pääasiassa lihaksiston väsymiseen liittyvien termien selittämiseen. Työssä tarkastellaan erilaisia lihaksiston väsymisestä johtuvia tiloja, palautumista ja elektromyografiaa. Elektromyografia on tutkimusmenetelmä, jonka avulla rekisteröidään lihasten sähköisiä toimintoja ja aktiopotentiaaleja. Ensimmäisen osan tavoitteena oli selvittää tahdonalaisesti liikutettavien lihasten väsymistä suorituksen aikana ja sen jälkeen. Ensimmäisen osan aiheen sain koululta. Työn ensimmäinen osa valmistui keväällä 2015.

Työn toisen osan tilaajan toimi oululainen ohjelmistoalan yritys Piimega Oy. Työn osassa käsitellään käyttöliittymän automaatiotestausta. Työn tavoitteena oli toteuttaa automaattinen testausympäristö Total Commerce ERP -tuotteelle. Ympäristön toteutuksessa huomioitavaa oli, että sen piti toimia mahdollisimman vähällä ylläpidolla ja säästää kehittäjien aikaa testausprosessista. Toisen osan aihe sai alkunsa, kun tilaaja siirtyi keskitettyihin versiopäivityksiin Total Commerce ERP:stä.

2 ENSIMMÄISEN OSAN ESITTELY

Opinnäytetyön ensimmäisessä osassa (liite 1) keskitytään lihaksiston väsymiseen liittyvään teoriaan. Työn tarkoituksena oli tutustua erilaisiin lihaksiston väsymiseen liittyviin tapahtumiin ja seurauksiin. Tarkasteltavia asioita työssä ovat myös harjoittelun ja palautumisen vaikutukset lihaksistoon. Työssä myös otettiin selvää yleisistä lihaksiston väsymisen mittausmenetelmistä.

Opinnäytetyön aikana sain hyvän yleiskäsityksen siitä, mitä lihaksistossa tapahtuu harjoittelun seurauksena. Työn aikana myös sain paljon lisätietoa lihaksen toiminnasta urheilusuorituksen aikana ja jälkeen. Työssä tarkastellaan lihasten väsymystä monesta erilaisesta näkymästä. Kuinka liiallinen harjoittelu vaikuttaa lihaksistoon ja mitä seurauksia liiallisesta harjoittelusta tulee, jos ei anneta lihaksiston palautua kunnolla.

Tavoitteena oli selvittää tahdonalaisesti liikutettavien lihasten väsymistä suorituksen aikana ja sen jälkeen. Työssä luodaan kokonaiskuva, mitä lihaksistossa tapahtuu sen väsyessä ja palautuessa. Lihaksen väsymisestä kuvataan sen yleisimmät tapahtumat. Palautumisesta kuvataan lihaksen palautumisprosessi ja mitä seurauksia voi olla, jos lihaksisto ei palaudu kunnolla.

3 TOISEN OSAN ESITTELY

Opinnäytetyön toisen osan (liite 2) aiheena oli käyttöliittymän automaatiotestaus. Työssä toteutettiin automaattinen testausympäristö, joka suorittaa testitapahtumia itsenäisesti. Työssä käydään läpi ohjelmistojen testausta yleisesti, käyttöliittymän testaukseen tarkoitetun työkalun ominaisuuksia, käyttöliittymätestin ohjelmointi, testaukseen tarvittavia muita työkaluja ja testattavaa ohjelmistoa.

Työssä tehtäviin käyttöliittymätestien tekemiseen käytettiin hyväksi Microsoftin Coded UI Tests -testityökalua. Työkalun käyttöä ja sen tarjoamia ominaisuuksia on kuvattu työssä. Testausympäristön toteuttaminen vaati myös monien muiden työkalujen opettelemista. Näitä työkaluja ovat Jenkins, PsExec ja SQL-palvelin. Käytettävistä työkaluista ja niiden ominaisuuksista on kerrottu tarkemmin työssä.

Työn tavoitteena oli saada syventävää tietoa ohjelmistojen testauksesta, toteuttaa automatisoitu testausympäristö, dokumentoida testitapahtumat ja syventää ohjelmointiosaamista. Testiympäristön toteutuksessa huomioitavaa oli, että sen pitää olla toiminnassa mahdollisimman vähällä ylläpidolla ja säästää kehittäjien aikaa testausprosessista. Testausympäristö saatiin kokonaisuudessaan toteutettua.

4 YHTEENVETO

Työn ensimmäiseen osaan sain aiheen koululta. Tartuin mielelläni aiheeseen, koska urheilu ja lihaksisto ovat kiinnostaneet aina minua. Työstä sain paljon irti ja opin, mitä muutoksia lihaksistossa tapahtuu harjoittelun seurauksena. Palautumiseen sain myös erilaista näkökantaa, kuinka huono palautuminen voi vaikuttaa jaksamiseen ja leposykkeeseen. Opin myös erilaisista lihaksiston väsymiseen liittyvistä mittausmenetelmistä.

Työn toisessa osassa tilaajana toimi oululainen ohjelmistoalan yritys Piimega Oy. Olin tehnyt harjoittelun Piimegalle ja opinnäytetyön tekeminen sinne oli luonnollinen jatkumo. Työ oli kiinnostava tehdä ja mielenkiintoa riitti koko opinnäytetyön ajan. Erityisesti testausympäristöä tehdessäni minua motivoi nähdä omat työni tulokset. Työssä käsitellään käyttöliittymän automaattista testaamista ja ohjelmistotestaukseen liittyvää teoriaa. Toteutettu testausympäristö on ohjelmoitu Visual Basic-kielellä. Testausympäristö on otettu tilaajalla käyttöön ja sitä on tarkoitus jatkokehittää tilaajan verkkosivujen testaamiseen.

Koosteopinnäytetyö ei toteutunut kohdallani parhaalla mahdollisella tavalla, koska työn ensimmäinen ja toinen osa eivät liittyneet toisiinsa. Jaettaessa opinnäytetyö pienempiin kokonaisuuksiin ensimmäisessä osassa pitäisi keskittyä työn teoriaan, jota käytettäisiin sitten hyväksi työn myöhemmissä vaiheissa.

LIITTEET

Liite 1 Lihasten väsymisen mittaaminen

Liite 2 Käyttöliittymän automaatiotestaus

LIITE 1

OULUN AMMATTIKORKEAKOULU

Janne Kanniala

LIHASTEN VÄSYMISEN MITTAAMINEN

LIHASTEN VÄSYMISEN MITTAAMINEN

Janne Kanniala
Opinnäytetyö
Kevät 2015
Tietotekniikan koulutusohjelma
Oulun ammattikorkeakoulu

SISÄLLYS

1 JOHDANTO

Työ on ensimmäinen osa kaksiosaista opinnäytetyötä. Ensimmäisessä osassa pääpainona oli lihasten väsymiseen liittyvä teoria.

Oman fyysisen suorituskyvyn parantaminen on nykypäivänä kasvava trendi. Yhä useammat ihmiset kuntoilevat päivittäin. Kuntoilussa on ollut useiden vuosien ajan apuna sykemittareita, joilla ei kuitenkaan voi mitata lihaksen väsymistä.

Lihaksisto jaetaan kolmeen lihaskudostyyppiin, joita ovat sileä lihas, sydänlihas ja poikkijuovainen lihas. Poikkijuovainen lihas on näistä tahdonalaisesti ohjattava. (1.) Opinnäytetyön tarkoituksena on selvittää tahdonalaisesti liikutettavan lihaksen väsyminen lihasta kuormittavan suorituksen aikana ja suorituksen jälkeen.

Lihaksisto väsyy ihmisen kuormittaessa sitä, mikä voidaan todentaa suorituskyvyn heikkenemisenä. Rankassa suorituksessa syntyy maitohappoja. Maitohappojen seurauksena lihaksisto happamoituu, ja tämä happamuus estää lihaksen energiansaantia.

Lihaksen rasittavuutta voidaan tutkia elektromyografian (EMG) avulla. Siinä kuvataan lihasten ja hermojen välisiä sähköisiä ilmiöitä.

2 LIHASTEN RASITTUMINEN

Lihas väsyy ollessaan jännityksessä tai suorittaessaan toistuvia supistuksia kuormituksen alaisena. Väsymisen voi kuitenkin suoraan yhdistää lihasten voimantuoton heikkenemiseen ja epämiellyttävään kivun tunteeseen lihaksessa. Lihaskudokseen kohdistuneella lihasväsymisellä viitataan hermo-lihasjärjestelmän maksimaaliseen tai optimaaliseen suorituskyvyn heikkenemiseen. Väsymiseen vaikuttavat mm. hermoston väsyminen, häiriöt lihassupistuksessa, solujakauma, energiavarastojen riittävyys ja lihaksen pH:n lasku. (2.)

Lihasväsymystä voidaan mitata laboratorioissa suoritettavilla isometrisillä lihasjännityksillä. Väsymystutkimuksessa lihaksen lämpötila ja lihaskudoksen paine ovat väsymykseen vaikuttavia päätekijöitä. Lihasta voidaan tutkia kehon ulkopuolella ottamalla näytepala halutusta lihaksesta. Tutkimalla lihasta koepalan avulla on otettava huomioon, että lihas ei välttämättä käyttäydy samalla tavalla kuin kehon sisällä. Lihasta voidaan tutkia myös kehon sisältä käsin. Elektromyografiamittaus on esimerkiksi kehon sisäistä lihaksen tutkimista. (3.)

Yksi keskeinen asia, mitä lihaksessa tapahtuu, kun se väsyy, on maksimaalisen voimantuottokyvyn lasku ja kyky ylläpitää normaalia voimatasoa. Maksimivoima voi laskea jopa 50 % väsymyksen seurauksena. Minuutin mittaisen maksimaalisen lihasjännityksen seurauksena lihasta on todella vaikea jännittää 30 sekuntia lihasjännityksen jälkeen. Muita lihasväsymyksen aiheuttamia oireita ovat voimantuottonopeuden ja rentoutumiskyvyn hidastuminen. Säännöllisellä lihasharjoittelulla voi yrittää minimoida voimantuottonopeuden hidastumista. (3.)

Voimaharjoituksessa, jossa treenataan lihaksisto väsymykseen asti, lihaksen fosfokreatiini varastot palavat melkein täysin loppuun ja glykogeenipitoisuus vähenee noin 40 %. Fosfokreatiinien tehtävä kehossa on olla lihaksiston ja hermoston energian lähde. Fosfokreatiinista 95 % sijaitsee ihmisen luustolihaksistossa. Kehossa fosfokreatiinin fosfaattiryhmä siirretään entsyymin avulla (kreatiinikinaasi) ADP:lle, jonka tuotoksena reaktiosta muodostuu kreatiinia ja ATP:tä. Kehossa on fyysisen suorituksen jälkeen korkea kreatinikinaasitaso, ja laktaattikonsentraatio arvot ovat koholla. (2.)
Ihmisen treenatessa lihaksiaan lihaksistossa tapahtuu mikrovaurioita. Mikrovaurioiden seurauksena kasvutekijöiden määrä solun sisällä lisääntyy ja proteiinisynteesi kiihtyy. Kehon korjatessa vauriot lihaksen pinta-ala kasvaa.

Edellä mainittujen ilmiöiden lisäksi lihasten väsymiseen vaikuttavat myös perinteiset tekijät, kuten sukupuoli, lihaksen verenkierto, ikä ja aineenvaihdunta-aineet. (3.)

2.1 Sentraalinen väsyminen

Sentraalinen väsyminen tarkoittaa sitä, että lihakseen saapuvien aktiopotentiaalien määrä laskee normaali tilasta. Sentraalinen väsyminen johtuu siitä, että keskushermoston kyky ärsyttää motoneuroneita laskee fysiologisten syiden vuoksi. Useat hormonit (adrenaliini, noradrenaliini), joita vapautuu urheillessa ehkäisevät sentraalista väsymystä. (3.)

2.2 Periferinen väsyminen

Suurin osa lihasväsymyksestä johtuu periferisestä väsymisestä. Periferistä väsymistä aiheuttaa muutokset lihaskalvoissa ja T-putkijärjestelmän depolarisaation jälkeisissä prosesseissa. Elektrodien avulla mm. elektromyografiassa voidaan mitata lihassolukalvon impulsseja ja potentiaalimuutoksia. (3.)

Lihassolukalvon laskenut kyky siirtää aktiopotentiaaleja solukalvoja pitkin johtuu ATP:n saatavuuden heikkoudesta ja siitä riippuvaisten natrium-kalium-pumppujen toiminnan muutoksista. Pumppujen toiminnan muutoksista seuraa, ettei natriumia pääse poistumaan lihassolusta ja kaliumia ei pääse lihassoluun. (3.)

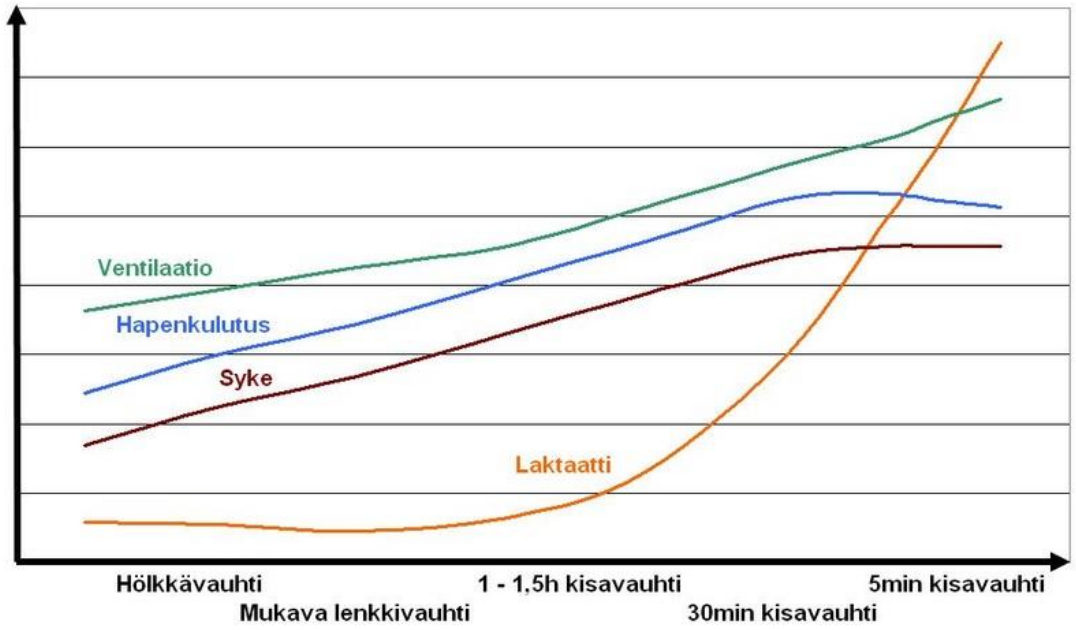
2.3 Lihaksen toiminta

Urheilusuorituksen aikana lihaksiston verenkierto kasvaa viisinkertaiseksi ja hapenkulutus kymmenkertaiseksi. Rasituksen aikana lihaksisto hankkii energiansa glykogeeneistä ja rasvahapoista. Heikkokuntoisella ihmisellä lihaksen energiansaanti koostuu enimmäkseen glykogeeneistä. Hyväkuntoisella puolestaan palaa enemmän rasvahappoja ja glykogeenejä jää säästöön. Valkuaisaineiden ja aminohappojen osuus energiantuotannosta on suhteellisen pieni (5–10 %). Lihassolussa lisääntyy mitokondrioiden määrä rasituksessa. (4.)

2.3.1 Laktaatti

Harjoiteltaessa tarpeeksi kovaa lihaksistoon muodostaa laktaattia (”maitohappoa”). Harjoittelu voidaan jakaa kahteen kuormituskategoriaan laktaatin avulla, aerobisen ja anaerobiseen. Aerobinen harjoittelu tarkoittaa sitä, että syke pysyy alle 70 % maksimisykkeestä koko ajan. Anaerobinen kyn-

nys tarkoittaa sykerajoilla kuvattuna 85–90 % maksimisykkeestä. Laktaatin vaikutus näihin harjoittelumuotoihin on se, että aerobisessa harjoittelussa laktaatti pystytään polttamaan pois, mutta anaerobisessa keho ei enää pysty polttamaan sitä ja näin ollen väsymys on ennemmin tai myöhemmin edessä. (5.) (Kuva 1.)



KUVA 1. Laktaatin muodostuminen erilaisilla juoksuvauhdeilla (6.)

Lyhyen palautumisen aikana lihaksisto pystyy muuttamaan osan laktaatista energiaksi. Palautumisen pituus määrittää sen, kuinka paljon laktaatista voidaan tuottaa energiaksi ja kuinka korkeaksi laktaatin tilavuus veressä kasvaa.

Laktaateilla on myös hyödyllisiä vaikutuksia, mm. veren kohonneen laktaattipitoisuuden myötä kasvuhormonivaste kasvaa. (2.)

2.3.2 Viivästynyt lihaskipu

Tarpeeksi kovan rasituksen jälkeen elimistön lihaksisto on ”kipeänä”. Tämä kipu on seurausta lihasvaurioista. Yleensä kipu alkaa harjoittelun seuraavana päivänä. Tällöin lihassoluista vapautuu valkuaisaineita ja entsyymejä. Lihaksistonvauriot korjaavat erilaiset solut ja kudoksen muutokset voivat säilyä muutamia viikkoja, mutta täydellinen korjautuminen voi kuitenkin kestää jopa kolme kuukautta. (7.)

3 PALAUTUMINEN

Kuormituksen jälkeen lihaskudos tarvitsee lepoa palautuakseen rasituksesta ja saavuttaakseen taas parhaan mahdollisen suorituskyvyn. Palautumisen aikana elimistö poistaa lihakseen kertyneitä kuona-aineita ja pyrkii täyttämään kuormituksesta johtuvaa energiavajetta. Lihaskudoksen palautuminen ei yleensä käynnisty heti rasituksen jälkeen ja suorituskyvyn lasku voi jatkua vielä hetken aikaa. Keskeinen tekijä palautumisen kannalta on kudoksen verenkierron voimakkuus. Kuona-aineet pääsevät veren mukana kudoksesta pois ja energiaa saadaan kuljetettua rasittuneisiin kudoksiin. (8.)

Maksimaalisen urheilusuorituksen jälkeen elimistön ATP-varastot laskevat 30–40 % ja KP-varastot lähes 100 %. ATP-varastot palautuvat nopeimmin kuormituksen jälkeen (1–3 min), KP-varastot palautuvat 50 % puolen minuutin aikana. Ainoastaan maksimaalisessa intervalliharjoittelussa palautuminen on hitaampaa. (2.)

Urheilusuorituksen jälkeen ruokailulla on tärkeä merkitys. Hiilihydraattipitoinen ravinto lisää maksan verenkiertoa ja uusien glykogeenivarastojen valmistus voi alkaa lihaksistossa ja maksassa. Glykogeenivarastot täyttyvät kahdessa vaiheessa. Ensimmäistä vaihetta kutsutaan anaboliseksi ikkunaksi, joka on 1h–1,5h treenin jälkeen. Glykogeenia kertyy tällöin lihaksistoon nopeasti. Ruokailu olisi hyvä ajoittaa tälle ajanjaksolle, koska tällöin elimistön lihakset ottavat ravintoa hyvin vastaan. Anabolisen ikkunan aikana insuliinia ei tarvita siirtämään glykogeenejä lihakseen. Glykogeenin siirtymisnopeus anabolisessa ikkunassa on 12– 30mmol/l tunnissa. Anabolisen ikkunan ulkopuolella nopeus on 3mmol/l tunnissa. (4.)

Kevyellä jälkiverryttelyllä voidaan nopeuttaa laktaatin poistumista hieman. Jälkiverryttelyn intensiteetti on hyvä pitää välillä 30–60 % maksimisuorituksesta. Myös kylmäkäsittelyn on todettu nopeuttavan palautumista. (8.)

3.1 Rasitusvammat

Lihaskudos sopeutuu hyvin siihen kohdistuvaan kuormitukseen, jos kuormituksen teho, kesto ja intensiteetti ovat sopivia. Kuitenkin liiallinen pitkä-aikainen kuormitus voi aiheuttaa kudosvaurioita.

Lihasten väsyessä niiden iskunvaimennuskyky heikkenee ja kontaktiin liittyvät iskuvoimat välittyvät eri tavalla jakautuen, ja kontaktista seuraavat iskuvoimat voivat kohdistua luuhun. Myös lihasepätasapaino ja lihaskireys voivat altistaa luiden ja pehmytkudoksien rasitukselle. Rasitusvammoja voi välttää hyvällä lihashuollolla ja oikeanlaisella harjoittelulla. (8.)

3.2 Stressi ja uni

Stressi vaikuttaa ihmisen kehoon negatiivisesti. Stressi voi aiheuttaa univajetta, heikentää immuniteettiä, lisätä sydänperäisen kuoleman riskiä ja astmaan sairastuvuutta. Stressi kuluttaa energiavarastoja, hidastaa ruuansulatusta ja nostaa verenpainetta. Jos ihmiskeho kärsii stressistä, lihaksien palautumiskapasiteetti kärsii. Palautumista häiritsee stressin aiheuttamat univajeet ja kohonnut sydämen syke. Korkean stressin omaavalla urheilijalla isometrisestä voimasta palautui vain 38,2 %, kun taas matalan stressitason urheilijalla palautuminen oli 60,4 %. (2.)

Uni on terveyden kannalta perusedellytys. Se vaikuttaa vireystilaan ja mielialaan. Huippu-urheilijan elimistö kehittyy harjoitteiden välisen palautumisjakson aikana. Mitä paremmin palaudut, sitä parempi on harjoituksen tulos ja seuraavan harjoituksen teho. (9.)

Unenpuute vaikuttaa lukemattomiin tekijöihin, jotka ovat kriittisiä palautumiselle. Näitä tekijöitä ovat mm. kudosten korjaus- ja uudistusprosessi, sokeriaineenvaihdunta, hormonaalinen järjestelmä ja ruokahalu. Unen laatua voi puolestaan peilata ylikuntoon, jota käsitellään myöhemmin tässä työssä.

3.3 Ylikunto

Urheilijalle ylikunto voi iskeä missä vaiheessa elämää tahansa. Ylikunnossa olevan henkilön elimistöä on rasitettu enemmän, kuin mistä sillä on mahdollisuudet palautua. Ylikuntoa voi esiintyä akuuttina ja kroonisena. Akuuttisessa muodossa palautuminen kestää yleensä vähemmän (noin 1–4 viikkoa), kuin kroonisessa (noin 3kk–1v). (10.)

Ylikunto vaikuttaa negatiivisesti urheilijan suorituskykyyn. Sen tavallisimpia oireita ovat väsymys, kohonneet sykearvot, alttius rasitusvammoille ja immuniteetin heikkeneminen. (10.)

4 ELEKTROMYOGRAFIA

Elektromyografia on tutkimusmenetelmä, jonka avulla rekisteröidään lihasten sähköisiä toimintoja ja aktiopotentiaaleja. Se perustuu aktiopotentiaalivirtojen rekisteröintiin, joista seuraa lihassolukalvolla varautuneiden ionien konsentraatiomuutoksia. Aktiopotentiaali varaus näyttää lihaksen sen hetkisen kuormitusasteen. EMG-signaali kerätään kehon omien elektrodien avulla.

Hermo- ja lihassolukalvolla vallitsee jännite-eroja, jotka johtuvat ionikonsentraatioista. Solukalvo on varautunut eri lailla solun sisä- ja ulkopuolelta. Varauksen suuruus riippuu solukalvolla vallitsevasta tilasta. Solukalvolla esiintyy kolme erilaista tilaa, kun tarkastellaan tilannetta aktiopotentiaalin kannalta. Nämä tilat ovat lepopotentiaali, depolarisaatio ja repolarisaatio.(2.)

EMG-mittausta suunniteltaessa kannattaa miettiä tarkkaan, mitä halutaan selvittää ja millaista informaatiota lihaksesta halutaan saada. Mittauksen yleisin tarkoitus on selvittää, onko lihas aktiivinen silloin, kun sen pitäisi olla, tai onko lihas aktiivinen silloin, kun sen ei pitäisi olla. Lisäksi tarkkaillaan, onko aktiivisuus normaalia vai katkonaista.

5 YHTEENVETO

Opinnäytetyöni tarkoitus oli antaa tietoa lihasten väsymisestä, palautumisesta ja lihaksen väsymisen mittausmenetelmästä. Aihe itsessään herätti kiinnostusta minussa, koska olen urheillut paljon elämäni aikana ja olen kiinnostunut mitä kehossa tapahtuu solu ja lihastasolla urheiltaessa. Seuraavassa opinnäytetyön vaiheessa on tarkoitus tehdä mittauksia elektromyografian avulla. Odotan innolla sitä, että pääsee itse tekemään mittauksen, näkemään aktiopotentiaalieroja ja analysoimaan tuloksia.

Tavoitteenani oli opinnäytetyön alussa selvittää tahdonalaisesti liikutettavien lihasten väsymistä suorituksen aikana ja sen jälkeen. Mielestäni onnistuin tässä hyvin ja lisää tekstiäkin tuli mm. mittausmenetelmästä ja palautumisesta. Opin paljon uutta lihaksistossa tapahtuvissa asioista suorituksen aikana, ja kuinka monia tapahtumia suorituksen aikana oikeastaan tapahtuu.

Olen tyytyväinen työn lopputulokseen. Seuraavassa vaiheessa pyrin panostamaan enemmän työn suunnitteluvaiheeseen, että saisin punaisen langan pidettyä paremmin kasassa. Aiheesta löytyy paljon tietoa, ja sitä on helppo lähteä laajentamaan suuntaan jos toiseenkin.

LÄHTEET

1. solunetti: Yleistä lihaskudoksesta. 2006. Solunetti. Saatavissa: http://www.solunetti.fi/fi/histologia/lihaskudos/. Hakupäivä 14.04.2015.
2. Haikarainen, Timo 2013. Kuinka kovaa treenaat? Osa 2: näin arvioit palautumistasi ja muokkaat treenisi sen mukaan – Haikarainen. Saatavissa: https://lihastohtori.wordpress.com/2013/07/22/kuinka-kovaa-treenaat-osa-2/. Hakupäivä 20.04.2015.
3. Kauranen, Kari 2014. Lihas – rakenne, toiminta ja voimaharjoittelu. Tampere: Liikuntatieteellinen Seura ry.
4. Lihaksen rakenne. 2008. Ravintokirja.fi. Saatavissa: http://www.ravintokirja.fi/Teoriatietoa_lihasten_toiminnasta_2008.pdf. Hakupäivä 20.04.2015.
5. Malinen, Sanna 2013.Sykerajat ja harjoittelu. Saatavissa: http://www.fitlandia.fi/sykerajat-ja-harjoittelu/. Hakupäivä 20.04.2015.
6. Erilaisten juoksuvauhtien (tehojen) vaikutukset elimistöön. 2015. Uusimaa juoksee. Saatavissa: http://www.uusimaajuoksee.fi/uusimaa-juoksee-hanke/valmennustietoa/osa-1-erilaisten-juoksuvauhtien-/. Hakupäivä 18.04.2015.
7. Hulmi, Juha 2014. Lihasarkuus – no pain no gain voimaharjoittelussa?. Saatavissa: https://lihastohtori.wordpress.com/2014/05/27/lihasarkuus-no-pain-no-gain-voimaharjoittelussa/. Hakupäivä 20.04.2015.
8. Vuori, Ilkka – Taimela, Simo – Kujala, Urho 2005. Liikunta lääketiede. Helsinki: Kustannus Oy Duodecim.
9. Kehittyminen tapahtuu levossa. 2016. Suomen Olympiakomitea. Saatavissa: http://www.huippu-urheilija.fi/urheileminen/lepo_ja_palautuminen/. Hakupäivä 18.04.2015.
10. Kallio, Tapio 2008. Kuntoilijan itsehoito-opas. Jyväskylä: WSOYpro/Docendo-tuotteet

LIITE 2

OULUN AMMATTIKORKEAKOULU

Janne Kanniala

KÄYTTÖLIITTYMÄN AUTOMAATIOTESTAUS

KÄYTTÖLIITTYMÄN AUTOMAATIOTESTAUS

Janne Kanniala
Opinnäytetyö
Kevät 2017
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

SISÄLLYS

1 JOHDANTO

Opinnäytetyön tilaajana toimi oululainen ohjelmistoalan yritys PiiMega Oy. PiiMega on perustettu vuonna 1998 ja siellä työskentelee noin 50 IT-alan ammattilaista. Toimipisteitä on kaksi ja ne sijaitsevat Oulussa ja Vantaalla. Opinnäytetyö liittyy tilaajan Total Commerce ERP-tuotteeseen, joka on verkkokauppoihin integroitu toiminnanohjausjärjestelmä. Suoritin myös opintoihini liittyvät harjoittelut PiiMegalle, joten opinnäytetyön tekeminen yritykseen oli luonnollinen jatkumo.

Testausjärjestelmä päätettiin uudistaa, kun tilaaja siirtyi keskitettyihin versiopäivityksiin Total Commerce ERP:stä. Työn tarkoituksena oli selvittää sopiva testausmenetelmä ja saada rakennettua testausympäristö Total Commerce ERP -tuotteelle.

Selvitystyön jälkeen päädyttiin Microsoftin Coded UI Test -työkaluun. Työkalu mahdollistaa käyttöliittymätestauksen ja suhteellisen nopeasti tehtävät testitapahtumat. Työkalun valinnassa myös merkittävä tekijä oli, että tilaajan toiminnanohjausjärjestelmää kehitetään jo entuudestaan Microsoftin Visual Studio-ohjelmalla.

Ympäristön on tarkoitus säästää ohjelmiston kehittäjien aikaa testausprosessista, toimia mahdollisimman vähällä ylläpidolla ja estää mahdollinen ohjelmiston virheellinen toiminta loppukäyttäjällä. Testiympäristö myös automatisoitiin ajamaan testit jokaisen versionhallintaan tehtävän päivityksen jälkeen. Automatisoinnin hoitaa avoimeen lähdekoodiin perustuva Jenkins. Jenkins oli jo entuudestaan käytössä PiiMegalla. Testausympäristöä on tarkoitus tulevaisuudessa jatkokehittää tilaajan verkkosivustojen testaukseen.

2 OHJELMISTOTESTAUS YLEISESTI

Ohjelmistojen testaus on mitä tahansa toimintaa, joka tähtää kehittämään ohjelmiston kykyä kohdata sille asetetut vaatimukset. Testaamisella voidaan osoittaa, että ohjelmisto täyttää kaupalliset ja tekniset vaatimukset, testattu ohjelmisto toimii halutulla tavalla ja ohjelmisto voidaan implementoida halutuilla ominaisuuksilla. Testauksen avullakin on kuitenkin hankala havaita kaikkia testattavan ohjelmiston virheitä. (1.)

Testauksen tarkoitus on löytää testattavasta ohjelmistosta virheet ja estää virheellisen ohjelmiston päätyminen loppukäyttäjälle. Testaus ei kuitenkaan sulje pois käyttäjäkohtaisia virheitä, mutta sillä voidaan taata ohjelmiston toiminta määritetyissä olosuhteissa oikein. Testausta käytetään myös varmistamaan, että testattava ohjelmisto ei tee mitään, mitä sen ei haluta tekevän. (1.)

2.1 Testauksen historia

Ohjelmistotestauksen alkuna voidaan pitää 1950-lukua. Testauksen historia voidaan jakaa viiteen erilaiseen vaiheeseen. Nämä vaiheet ovat seuraavat:

- Virheenjäljitys (1956) jolloin testaus oli oikeastaan sama asia kuin debuggaus
- Havainnollistaminen (1957–1978), joka oli aika, jolloin testattiin, että ohjelma vastaa sille asetettuja spesifikaatioita
- Hajottaminen (1979–1982), jolloin testattiin toteutuksessa olevia virheitä
- Arviointi (1983–1987), jolloin pyrittiin löytämään virheitä vaatimuksissa, suunnittelussa ja toteutuksessa
- Ennaltaehkäisy alkoi vuonna 1988 ja se on viides ja viimeinen vaihe. Testauksen tarkoituksena on virheiden ehkäisy vaatimuksissa, suunnittelussa ja toteutuksessa. (2.)

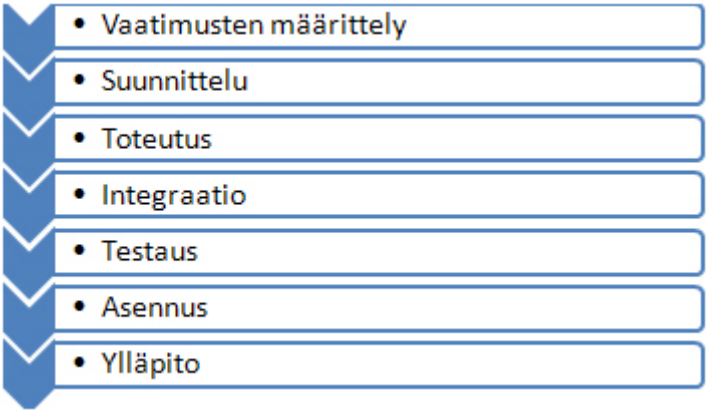
2.2 Testaus ohjelmiston kehitysprosessissa

Yleensä ohjelmistotestaus toteutetaan ohjelmistotuotantoprosessin loppupäässä. Tämä antaa kuitenkin harhaanjohtavan kuvan siitä, kuinka ohjelmiston testausta tulisi suorittaa. Testausta pitäisi

suorittaa koko ohjelmiston kehitysprosessin ajan aina määrittelyvaiheesta prosessin loppuun saakka.

2.2.1 Testaus vesiputousmallissa

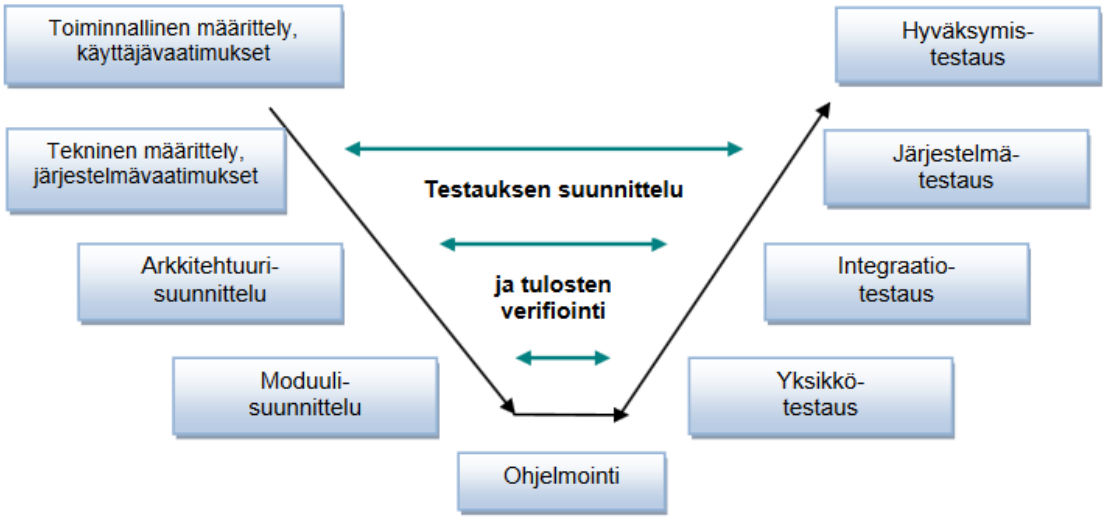
Vesiputousmallilla tarkoitetaan ohjelmistotuotannon prosessimallia, jossa vaiheet kulkevat vesiputouksen lailla tasolta toiselle. Malli on peräisin vuodelta 1970, jolloin sen esitteli Winston W.Royce. Vesiputousmallin vaiheet ovat järjestelmävaatimukset, ohjelmistovaatimukset, analyysi, suunnittelu, ohjelmointi, testaus ja käyttöönotto (kuva 1). Vesiputousmalli on ohjelmistotestauksen kannalta huono malli, koska siinä testaus suoritetaan vasta kehityksen jälkeen. Testauksen ollessa viimeinen vaihe se jää usein liian vähälle huomiolle. Myös testauksen avulla löydetyt suunnitteluvirheet ja viat tulevat kalliiksi ja aikaa vieväksi korjata, kun ne löydetään viimeisenä. (3.)



KUVA 1. Testaus vesiputousmallissa (4.)

2.2.2 Testaus V-mallissa

Myöhäisessä vaiheessa aloitetun testauksen ongelmia korjaamaan on kehitetty testauksen V-malli (kuva 2). Tässä mallissa ohjelmiston testaamiseen aletaan ottaa kantaa jo projektin alkuvaiheessa. V-mallin etuna vesiputousmalliin on, että testauksen suunnittelu aloitetaan heti projektin alussa. Tästä syystä projektin myöhästyessä V-mallissa ei karsita testaukseen käytettävää aikaa, niin kuin vesiputousmallissa kävisi. Ohjelmiston laatu myös paranee jatkuvan testauksen seurauksena. (3.)



KUVA 2. Testaus V-mallissa (3.)

2.2.3 Testaus ketterissä projekteissa

Ketterillä menetelmillä tarkoitetaan keveitä ja nopeasti muutoksiin reagoivia ohjelmistokehitysmenetelmiä. Ketterät menetelmät eivät varsinaisesti tunne testaajan roolia, vaan tiimin jäsenet ovat moniosaavia ja tiimi on itseohjautuva. Tämä tarkoittaa sitä, että tiimi itse päättää, kuinka kehitysjakson tehtävät organisoidaan ja jaetaan. Ketterissä projekteissa laadunvarmistus ja testaus on yleensä automatisoitu siten, että tiimillä on koko ajan tarkka käsitys ohjelmistossa olevista virheistä tai poikkeamista. Monet ovat myös sitä mieltä, että ketteryyttä ei voi aidosti soveltaa ilman automatisoitua testausta osana ohjelmistokehitysprosessia. (5.)

2.3 Regressiotestaus

Regressiotestauksella varmistetaan, että ohjelmistoon tehdyt muutokset eivät riko järjestelmää, vaan se toimii edelleen vaatimusten ja määrittelyjen mukaisesti. Regressiotestit ovat paljon resursseja vieviä, minkä seurauksena regressiotestausta suoritetaan yhä enemmän automaattisesti. Regressiotestit olisi pidettävä mahdollisimman vähäisenä, kuitenkaan vähentämättä testattavan alueen kattavuutta. Testejä olisi hyvä tehdä jokaisen virheen korjauksen jälkeen liittyen korjattuun virheeseen. Jos virheeseen on kirjoitettu monta erilaista testitapahtumaa, pitäisi vähemmän tehokkaasta testistä hankkiutua eroon. (6.)

3 MICROSOFT CODED UI TESTS -TESTITYÖKALU

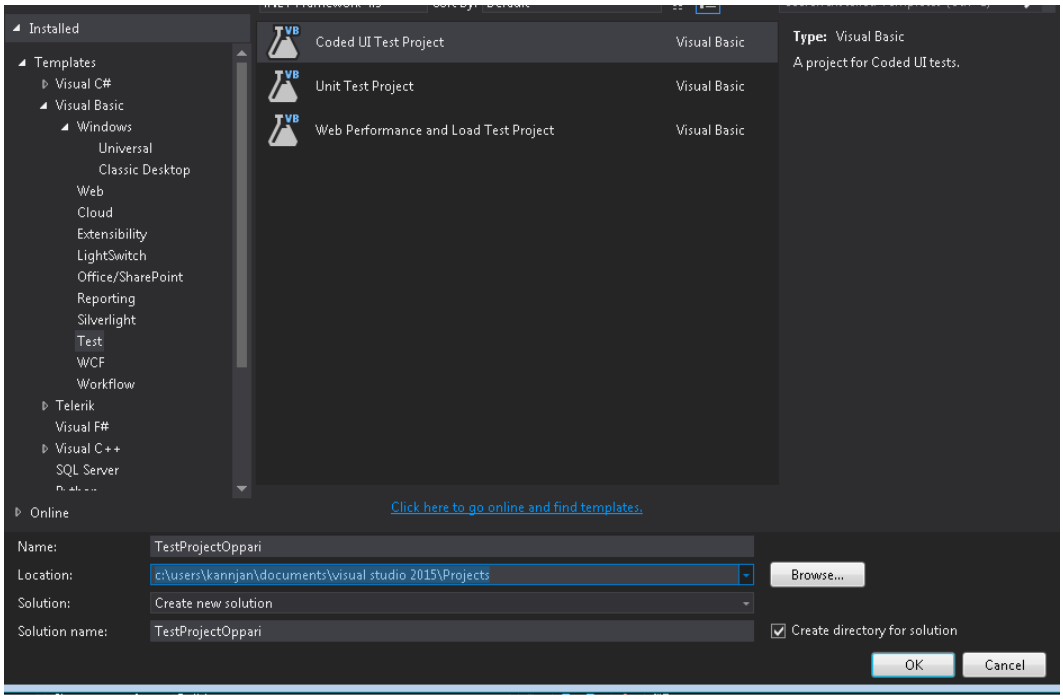
Testien tekemiseen käytetään Microsoftin omaa testityökalua nimeltään Coded UI Tests, joka sisältyy Visual Studion Enterpriseen. (7.) Työkalulla on mahdollista tehdä käyttöliittymätestejä suoraan Visual Studion avulla. Testien kulkua on mahdollista seurata Visual Studiosta, josta näkyy reaaliajassa meneekö testi läpi vai ei. Testien ajaminen Coded UI Tests -työkalulla on nopeaa, minkä seurauksena testejä on mahdollista ajaa useita kertoja päivässä. Testien tekemiseen tarvitaan Visual Studioon maksullinen Enterprise-lisenssi.

3.1 Testin suunnittelu

Ensimmäinen vaihe testien tekemisessä on testitapauksen suunnittelu. Yleensä tähän liittyy jokin tehty uusi ominaisuus ohjelmistoon, jonka perusteella testi tehdään. Suunnittelussa dokumentoidaan testi tapahtumakohtaisesti vaihe kerrallaan läpi ja kirjataan haluttu lopputulos ylös. Testien tarkistaminen(assertointi) kannattaa myös huomioida, koska testin nauhoittaminen joudutaan keskeyttämään tarkistuksen tekemisen ajaksi. Suunnittelussa huomioitavaa on myös, että pidetään testitapahtuman nauhoitus mahdollisimman lyhyenä. Tämän seurauksena testejä on helpompi ylläpitää, testattavan ominaisuuden muuttuessa koko testiä ei tarvitse nauhoittaa uudelleen ja testin epäonnistuessa on helpompaa paikantaa missä kohtaa virhe on tapahtunut.

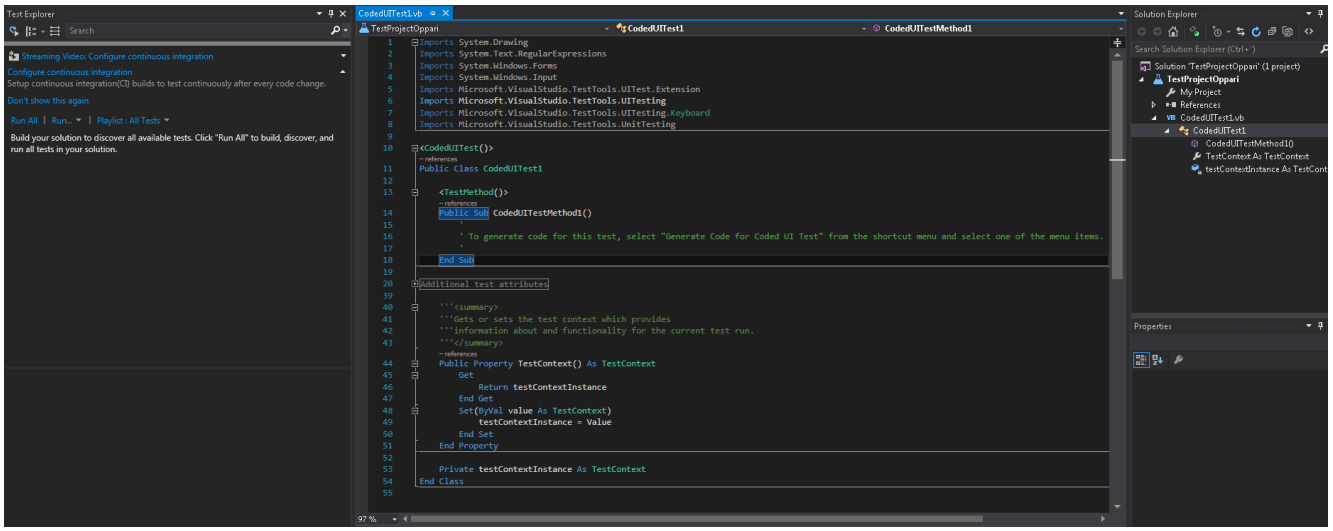
3.2 Coded UI Test-projektin luominen

Testiprojektin luominen tapahtuu Visual Studion käyttöliittymän kautta. Valitaan Visual Studiossa vasemmalta ylhäältä File -> New -> Project. Tämän jälkeen Visual Studioon ilmestyy erilaisia projektin aloituspohjia. Valitaan näistä Test -> Coded UI Test Project ja nimetään projekti halutulla nimellä (kuva 3).



KUVA 3. Projektin luonti

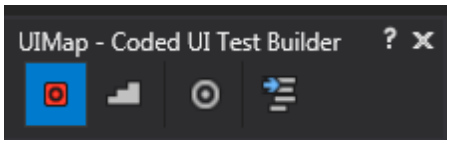
Tämän jälkeen Visual Studio generoi CodedUiTest.vb nimisen luokan ja yhden testimetodin. Testimetodin voi halutessaan nimetä uudelleen. Metodi kannattaa nimetä mahdollisimman kuvaavasti sillä se toimii jatkossa testin nimenä (kuva 4).



KUVA 4. Testiprojektin pohja

3.3 Käyttöliittymätestin tekeminen

Testin tekeminen tapahtuu Visual Studion Test Builderin avulla. (7.) Test Builderissa on mahdollista nauhoittaa testejä, tarkastella nauhoitetun testin steppejä, lisätä tarkistuksia (assertion) ja generoida nauhoitetun testin koodi (kuva 5).

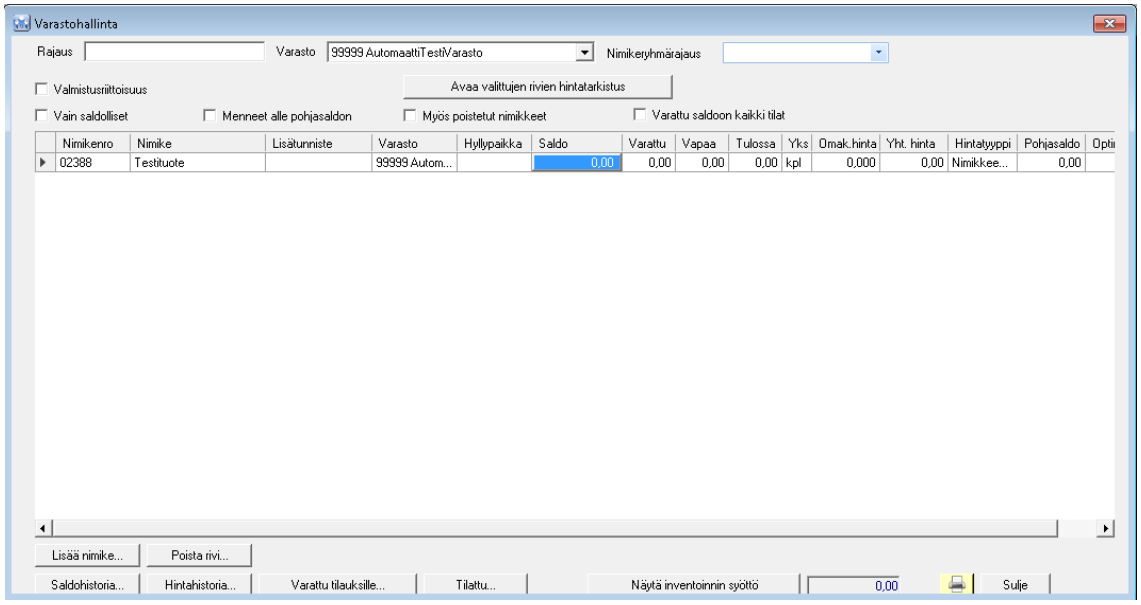


KUVA 5. Test Builder

Testin nauhoittamisen jälkeen Test Builder luo UImap.uitest-tiedoston, joka sisältää nauhoitetun testin vaiheet.

3.4 Testien ohjelmointi

Testien ohjelmointi tapahtuu pääasiassa nauhoittamalla testitapahtumia Coded UI Test Builderin avulla. Joitakin testien kohtia ei pystynyt suoraan nauhoittamaan, vaan ne ohjelmoitiin itse. Tässä tarkastellaan yhtä testiä, joka on ohjelmoitu käsin. Kuvassa 6 nähdään yksi testissä käytettävistä käyttöliittymänäkymistä.



KUVA 6. Testissä käytettävä käyttöliittymänäkymä

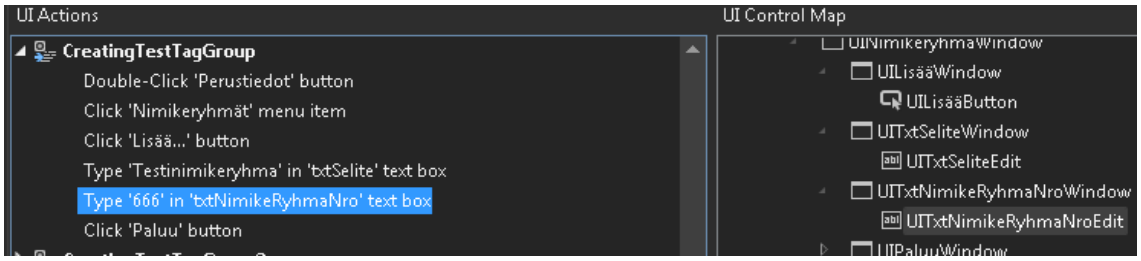
Tarkasteltavasta koodista (liite 1) nähdään testimetodin nimi (= SpecifyStorageUpdateBasedOnProductGroup), joka toimii myös testin nimenä. Metodin ensimmäisellä rivillä määritellään query-niminen muuttuja, joka on tekstityyppiä. Queryn arvoksi asetetaan palautuvan tietokantakyselyn arvo, ja tämän arvon ollessa true testin suoritus keskeytetään. Muutoin testi avaa testattavan Total Commerce ERP:n ja alkaa suorittamaan testiä. Testin vaiheet on kuvattu mahdollisimman kuvaavasti, että niitä voitaisiin mahdollisesti käyttää useampaan kertaan eri testitapahtumissa.

Tarkasteltaessa testin UIMap.vb-luokan aliohjelmaa ChangeStockValueForProduct nähdään, miten testin tämä vaihe on tehty (liite 2). Testissä määritellään aluksi kaikki painikkeet ja kentät, mitä siinä käytetään. Määrittelyn jälkeen seuraava vaihe testin suorittamisessa on saldo-kentän arvon tarkastelu. Saldo-kentän sen hetkinen arvo tallennetaan muuttujaan talteen, ja tätä muuttujaan tallennettua arvoa verrataan tämän jälkeen lukuun 50. Muuttujaan arvon ollessa 50 kasvatetaan muuttujan arvoa yhdellä. Muuttujan arvon tarkastuksen jälkeen testi suorittaa saldo-kentän klikkaamisen hiirellä. Tämä aukaisee uuden ikkunan (kuva 7), jossa Uusi saldo-kenttään asetetaan edellä tallennetun muuttujan arvo. Kuvassa 7 näkyvässä ikkunassa Syy-kenttä on joko käytössä tai pois käytöstä. Testissä pitää suorittaa SQL-kysely, jonka perusteella Syy-kenttään joko syötetään muutoksen syy tai ei. Testin lopuksi muutetaan Uusi saldo-kentän arvo Muuta-painikkeella ja muutettu arvo päivittyy tietokantaan.

KUVA 7. Saldon muutos -ikkuna

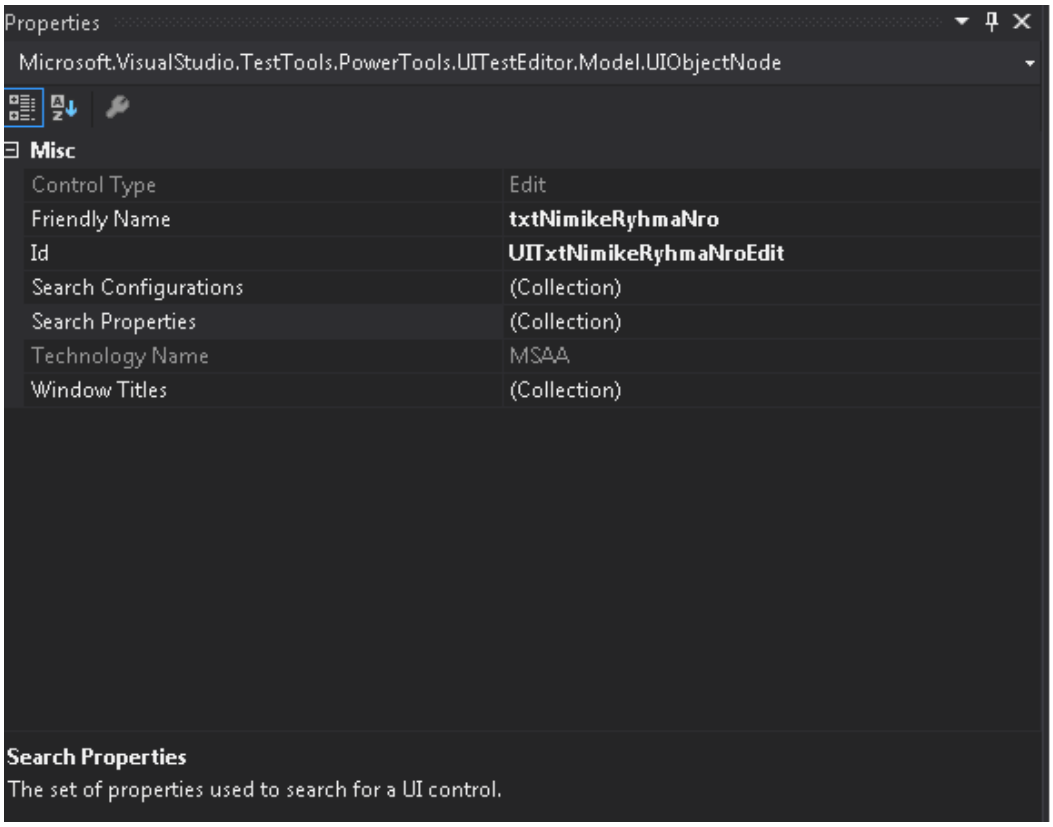
3.5 UIMap.uitest-tiedosto

UIMap.uitest-tiedostosta nähdään kaikki nauhoitetut testimetodit ja testien vaiheet. Nauhoitettujen testien vaiheita voi muokata UIMap.uitest-tiedoston avulla suoraan Visual Studion tarjoamassa käyttöliittymässä (kuva 8).



KUVA 8. UIMap.uitest-tiedosto

Kuvassa 8 nähdään yhden nauhoitetun testin vaiheet. Tästä näkymästä voi testin vaiheita jälkikäteen poistaa, asettaa viivettä eri vaiheiden suorituksen välille, luoda uuden metodin haluamastaan testin vaiheesta ja siirtää testin UIMap.vb-luokkaan. Näkymästä näkee myös kaikki testeissä käytetyt käyttöliittymäkomponentit. Tästä näkymästä voi myös tarkastella testeissä käytettävien käyttöliittymäkomponenttien ominaisuus-arvoja. Properties-näkymästä (kuva 9) pystyy mm. määrittelemään käyttöliittymäkomponenttien hakuun käytettäviä arvoja, joiden avulla testejä ohjataan.



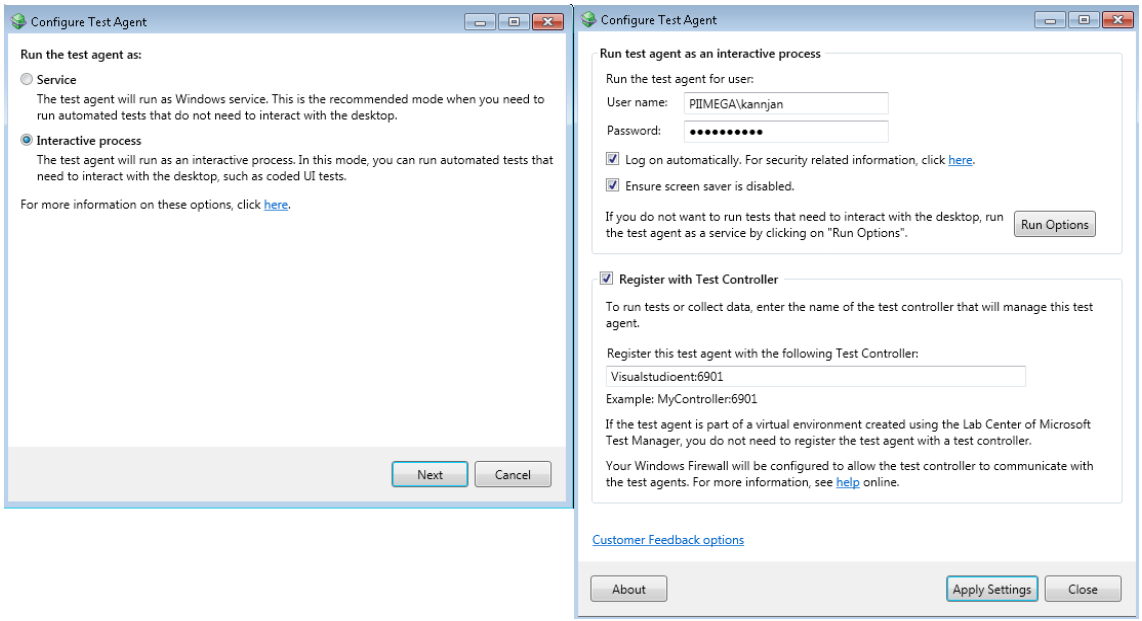
KUVA 9. Properties-näkymä

3.6 UIMap.vb-luokka

Edellä käydystä UIMap.uitest-näkymästä testejä on mahdollista siirtää UIMap.vb-luokkaan. Luokassa on mahdollista aloittaa testien ohjelmointi alusta tai kustomoida UIMap.uitest-näkymästä tuotuja testejä. Total Commerce ERP:n asiakaskohtaisista asetuksista riippuvat testit pitää tehdä tässä luokassa, koska asiakaskohtaisten asetusten selvitykseen tarvitaan SQL-kyselyitä.

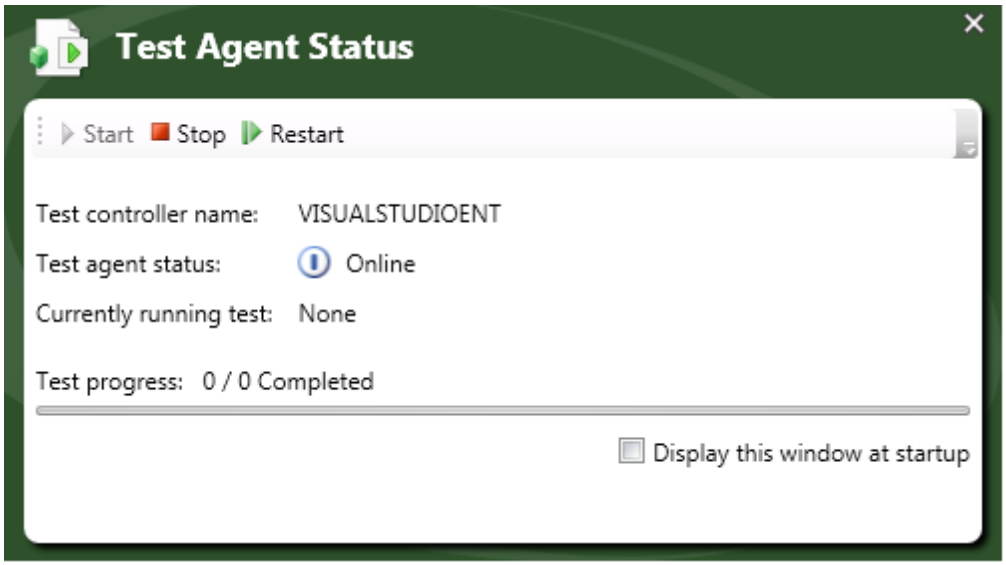
3.7 Testiagentit

Visual Studioon kehitettyjen testiagenttien tehtävänä on mahdollistaa testien ajaminen etänä. (8.) Agentit asennettin ja konfiguroitiin testipalvelimelle (kuva 10).



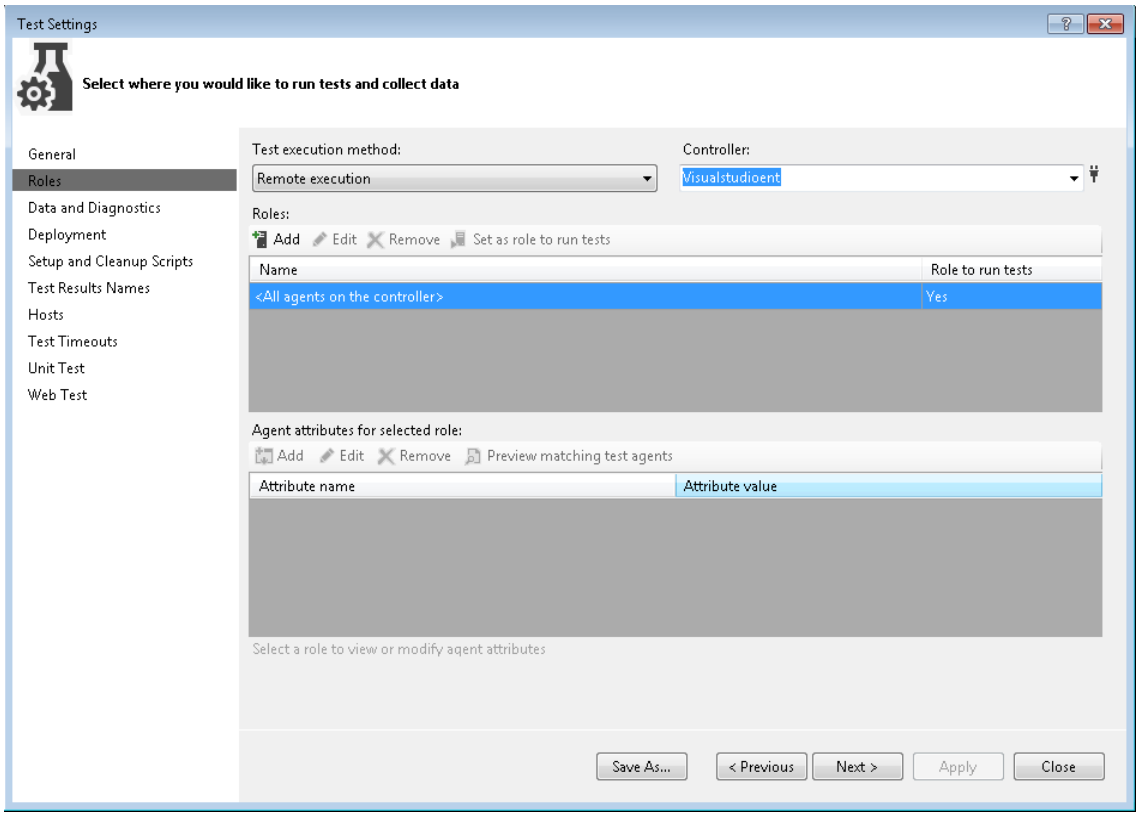
KUVA 10. Agenttien konfigurointi

Agenttien ollessa käytössä näytölle ilmestyy Test Agent Status -ikkuna, josta näkee agentin tilan. Näkymästä on mahdollista pysäyttää agentit, käynnistää uudelleen agentit ja nähdä sen hetken suoritettava testi (kuva 11).



KUVA 11. Test Agent Status -näkymä

Agenttien toimiminen edellytti myös projektin asetuksien konfigurointia. Projektiin lisättiin TestSettings-niminen tiedosto, johon asetettiin testit ajettavaksi etänä agenttien avustuksella (kuva 12).

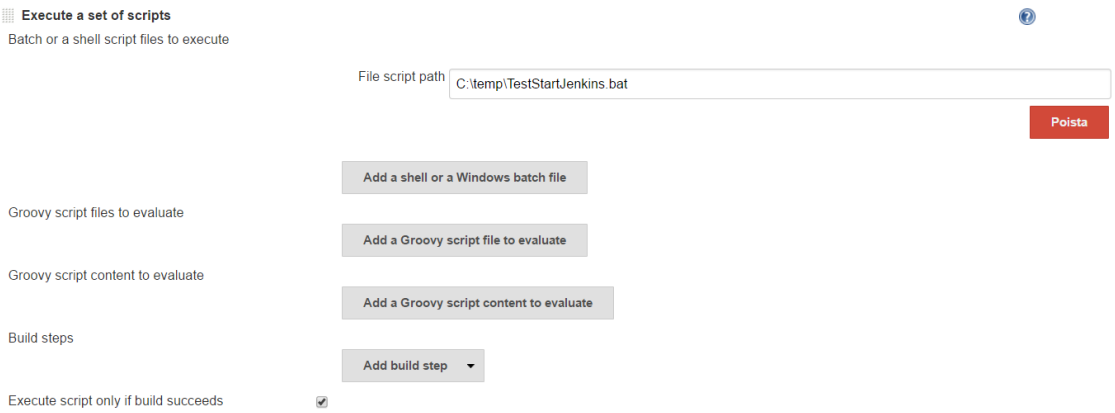


KUVA 12. Testiprojektin TestSettings-tiedosto

4 MUUT KÄYTETYT MENETELMÄT

4.1 Jenkins

Jenkins on avoimeen lähdekoodiin perustuva ilmainen Java-pohjainen Continuous Integration-palvelinohjelma, joka ohjaa sille määriteltyjen tehtävien suoritusta. (9.) Jenkins mahdollistaa versionhallinnan koodimuutosten seurannan, ja näyttää jokaisen versionhallintaan tulleen muutoksen kuvauksen. Jenkinsiin löytyy monipuoliset liitännäiset muun muassa työssä käytettävä Post build task-plugin, jonka avustuksella testit ajetaan Jenkinsin viimeisenä vaiheena (kuva 13). (10.)



KUVA 13. Jenkinsin PostBuildTask-näkymä

Jenkinsin hyödyistä voidaan myös mainita sen helppo konfigurointi, sillä se tarjoaa suoran käyttöliittymän. (10.)

4.2 PsExec

PsExec on Microsoftin tarjoama työkalu, jonka avulla voidaan suorittaa komentorivikomentoja joko paikallisesti tai etänä. (11.) PsExecin avulla Jenkins-palvelimelta annetaan testipalvelimelle komento käynnistää testit. Testien suoritus tapahtuu TestStartJenkins.Bat- nimisen tiedoston ajamisella. Komennossa –accepteula hyväksyy käyttäjäehdot. Seuraavana annetaan testipalvelimen osoite ja –u parametrin arvoksi syötetään testipalvelimen käyttäjätunnus ja –p puolestaan salasana (kuva 14).

```
"C:\temp\PsExec.exe" -accepteula \\visualstudioent -u                -p          "C:\TestStart.Bat"
```

KUVA 14. TestStartJenkins.Bat- tiedoston sisältö

TestStartJenkins.Bat tiedoston ajamisen seurauksena ajetaan TestStart.Bat- niminen tiedosto, joka käynnistää testit MsTest.exe:n avulla. TestStart.Bat- tiedoston sisältö kuvassa 15.

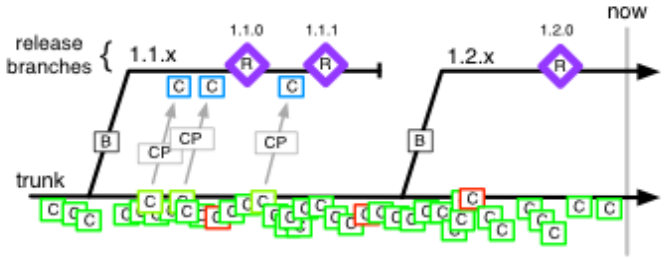
```
"C:\Program Files\Microsoft Visual Studio 14.0\Common7\IDE\MSTest.exe" /testcontainer:"C:\Source\TMEAutomaattinenTestaus\TMEAutomaattinenTestaus\bin\Debug\TestausDokumentoinninPohjalta.dll" /testsettings:"C:\Source\TMEAutomaattinenTestaus\TMEAutomaattinenTestaus\TestSettings1.testsettings"
```

KUVA 15. TestStart.Bat- tiedoston sisältö

Tiedostossa ajetaan Microsoftin MSTest.exe ja tälle ajettavalle tiedostolle annetaan testcontainer-parametrin arvoksi ajettavan testiprojektin polku ja testikohtaiset asetukset.

4.3 Versionhallinta

Tilaajalla käytössään oleva versionhallinta ohjelma on TortoiseSVN. (12.) Se on ilmainen avoimeen lähdekoodiin perustuva Windows-käyttöliittymä versionhallintaan. TortoiseSVN mahdollistaa versionhallintaan tehtävien muutosten tarkastelun ja näyttää, kuka muutoksen on siirtänyt versionhallintaan. TortoiseSVN mahdollistaa myös muutosten palauttamiset. Sillä pystyy myös haarauttamaan versionhallinnan kehityshaarasta julkaistavan version haaraan. Näitä haaroja kutsutaan nimellä Trunk ja Branch. Trunk toimii kehityshaarana ja Branch julkaistuna versiona (kuva 16). Automaattinen testaus on käytössä kehityshaarassa ja ennen Branchista julkaistavaa versiota käydään sille käsin ajamassa testit.

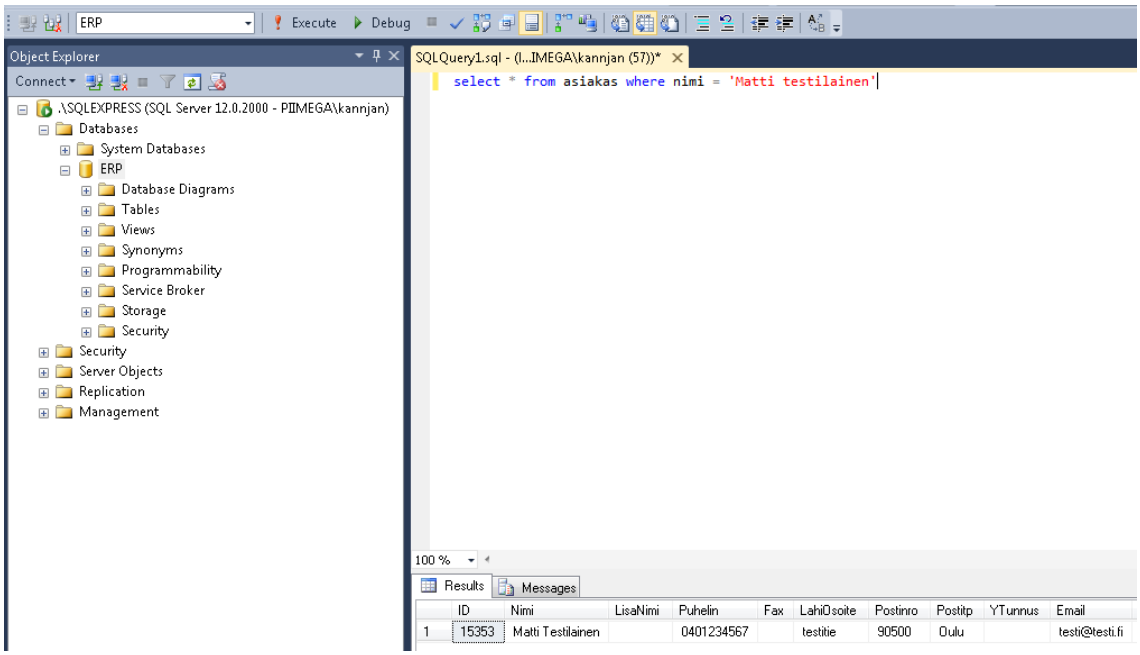


KUVA 16. Trunkin toimiessa kehityshaarana Branchista tehdään julkaistavat versiot. (13.)

4.4 SQL-Server

Lyhennys SQL tulee sanoista Structured Query Language, joka on IBM:n kehittämä standardoitu kyselykieli. (14.) SQL:n avulla on mahdollista suorittaa hakuja, tehdä muutoksia ja lisätä tietoa tietokantaan. Työssä testipalvelimelle asennettiin paikallinen SQL-palvelin, jonka tarkoituksena on säilyttää kaikkea sitä dataa, mitä ERP käyttää toiminnassaan. Palvelimelle luotiin yksi tietokanta, joka vastaa yhden Total Commerce ERP:n asiakkaan tietoja. Tietokantaan on mahdollista ylikirjoittaa dataa, jos halutaan vaihtaa asetuksia joita vasten ERP:n testausta suoritetaan.

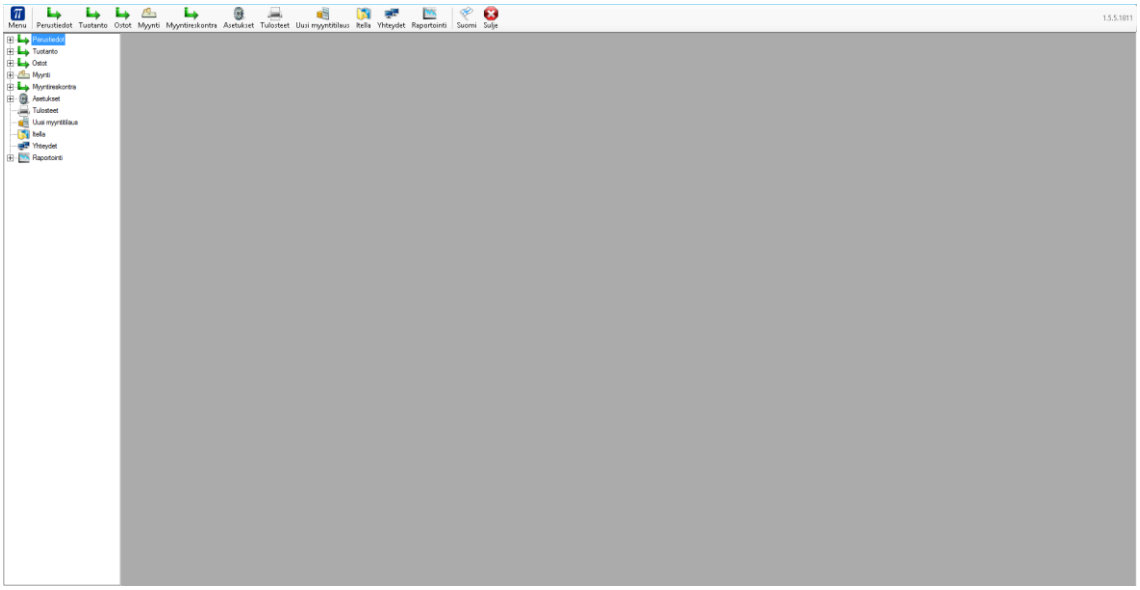
SQL-palvelimen hallintaan käytettiin Microsoftin työkalua nimeltään SQL Server Management Studio. (15.) Työkalu tarjoaa suoran käyttöliittymän SQL-palvelimen hallintaan. Käyttöliittymän avulla voidaan mm. suorittaa erilaisia SQL-kyselyitä ja ottaa varmuuskopioita tietokannoista. Management Studion käyttöliittymästä vasemmalta nähdään suoraan sen hetkiseen tietokantaan olevan yhteyden nimi. Yhteyden alapuolella nähdään kaikki tietokannat, joita kyseisellä palvelimella on. SQL-kyselyn suorittaminen tapahtuu New Query-painiketta painamalla, jolloin aukeaa tekstikenttä, mihin kyselyn voi kirjoittaa. Kyselyn suoritus tapahtuu painamalla Execute-painiketta tai F5 ja kyselystä palautuvat arvot tulevat suoraan näkyviin Management Studioon. Management Studion käyttöliittymä näkyy kuvassa 17.



KUVA 17. Management Studio

5 TOTAL COMMERCE ERP:N TESTAUS

Total Commerce ERP on toiminnanohjausjärjestelmä, joka on suunniteltu kaupoille, joilla myyntiä tapahtuu niin verkkokaupassa kuin kivijalkamyymälässä. Toiminnanohjausjärjestelmään on rakennettu Magento-verkkokauppa-rajapinta (16.), jonka avulla data verkkokaupan ja toiminnanohjausjärjestelmän välillä liikkuu. Toiminnanohjausjärjestelmän avulla voidaan helposti mm. seurata myyntiä, varastosaldoja ja tuotetietoja. Kuvassa 18 nähdään ERP:n aloitusnäkymä.



KUVA 18. ERP:n aloitusnäkymä

5.1 Dokumentointi

Ennen testausympäristön kehittämisen aloittamista testitapahtumat dokumentoitiin. Dokumentoinnissa otettiin huomioon kategoria, mihin testi liittyy, testitapauksen nimeäminen, testitapauksen vaiheet, testitapauksen odotettu tulos ja testitapauksen tuloksen varmistaminen. Dokumentaatio kasvaa koko ajan testitapahtumien määrän kasvaessa. Työn tilaajan käyttöön jäi testitapahtumiin liittyvä dokumentaatio. Kuvassa 19 nähdään yhden testitapauksen dokumentointi.

Kategoria	Testitapaus	Stepit	Odotettu tulos	Varmistetaan
Alustus	Testihenkilön generointi verkkokaupasta XML:n avulla	1. Valitse Yhteydet 2. Aja manuaalisesti ajo "Hae tilaukset verkkokaupasta" 3. Käytä verkkokaupan ftp-yhteyttä	Testeissä käytettävä henkilö luotu ERP:iin	Tarkistetaan tapahtumaloki, että OK

KUVA 19. Testitapauksen dokumentaatio

5.2 Testiprojektin alustaminen

Testiprojektin alustaminen tapahtuu määrittelemällä muuttujat testien lukumäärälle, hyväksytyille ja hylätyille testeille, versionumerolle, ohjelman käynnistymisen polulle, testituotteelle ja SQL-yhteydelle (kuva 20).

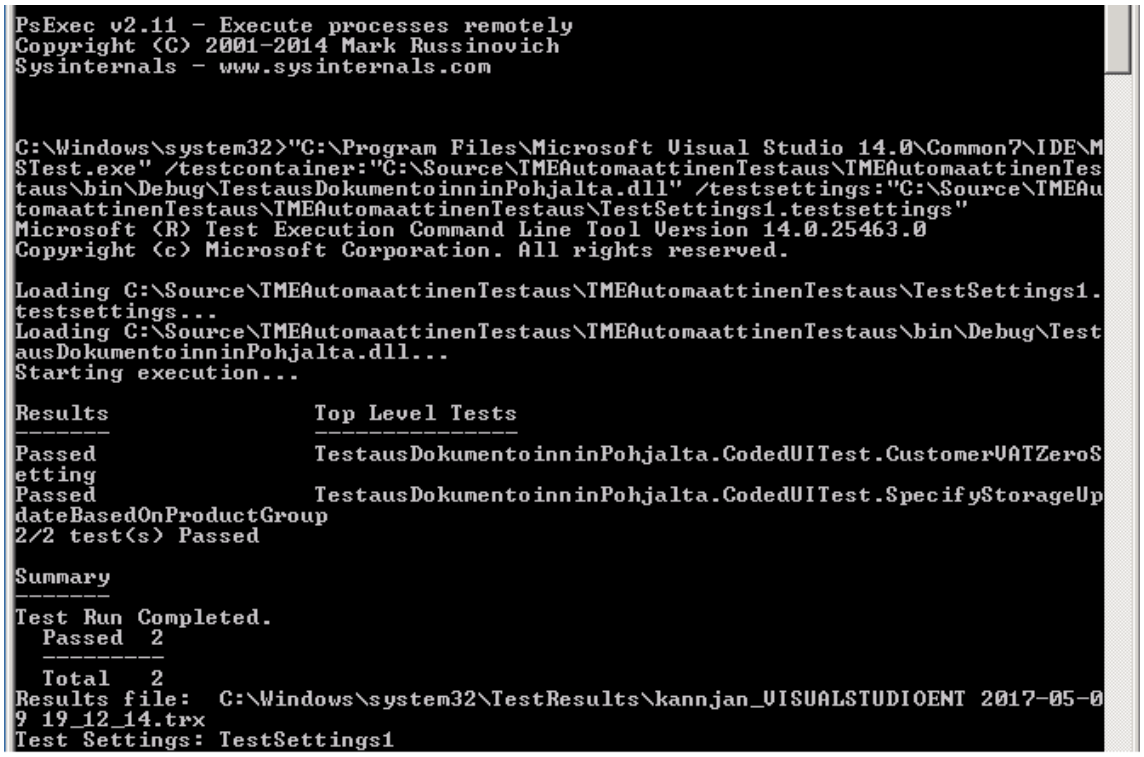
```
Private _testCount As Integer
Private _passedTests As Integer
Private _failedTests As Integer
Private _VersionValue As String
Private apppath As String = "C:\Demo\Total ERP\Trunk\TotalCommerce.exe"
Private testProductSKU As String = "02388"
Public Shared ConnectionString As String = "Data Source=.\SQLEXPRESS;Initial
```

KUVA 20. Testiprojektin alustus

Testien käynnistyessä myös suoritetaan aliohjelma ClassInit, joka suorittaa SQL-skriptin ja luo tyhjän lokitiedoston. SQL-skripitin tarkoituksena on asettaa ennalta määritellyt tietokanta-arvot testejä varten sopiviksi. Lokitiedostoon kirjataan suoritettujen testien tulos eli onko testi mennyt läpi vai ei.

5.3 Jenkinsin tehtävä testauksessa

Jenkinsin tehtävänä testauksessa on kääntää automaattisesti versionhallinnassa oleva lähdekoodi ja ajaa testiprojekti. Jenkins ajaa testiprojektin jokaisen versionhallintaan tehtävän koodipäivityksen jälkeen. Jenkins myös automaattisesti siirtää sen hetken ajetun version versionhallinnasta testipalvelimelle. Testipalvelimen versio Total Commerce ERP:stä päivittyy siis sitä mukaa, kun koodipäivityksiä tehdään versionhallintaan ja näin ollen järjestelmästä on aina testipalvelimella uusin versio testattavissa. Jenkins kommunikoi testipalvelimen kanssa PsExec:in avulla ja käskee testipalvelinta ajamaan testausprojektin (kuva 21).



KUVA 21. Jenkins-palvelin lähettää kutsun testipalvelimelle testien aloittamiseksi

5.4 Testiprojektin kääntäminen

Testipalvelimen saadessa Jenkinsiltä käskyn testien suorittamisesta alkaa testiprojektin kääntäminen. Visual Studio kääntää testiprojektin ja suorittaa ohjelmoidut testit käyttäen hyväksi testiagentteja. Testiprojektin kääntämisen aikana testipalvelimelle ei ole suotavaa mennä, koska se voi aiheuttaa virheitä lopputulokseen.

Testiprojektin kääntäminen aloitetaan hakemalla FTP-rajapinnan kautta testihenkilöt ja testituotteet. ERP hakee FTP:ltä XML-tiedoston, joka sisältää tiedot henkilön ja tuotteen tiedoista. ERP lukee nämä tiedot XML-tiedostosta ja tallentaa ne testiympäristön tietokantaan. Testihenkilöä ja testituotetta käytetään jokaisessa testissä, jossa testihenkilö tai testituote on pakollinen.

Käyttöliittymätestin lopussa on yleensä testin validointi, jolla tarkistetaan, että saatu lopputulos on oikea. Validointi voidaan suorittaa mahdollisuuksien mukaan suoraan käyttöliittymästä tai tietokantakyselyiden avulla. Käyttöliittymän validoinnissa voidaan lisätä suoraan TestBuilderista assertion, jolla tarkistetaan, että testin lopputulos on haluttu. Validoinnilla on vaikutusta siihen, meneekö testi läpi vai hylätäänkö se.

5.5 Testien ajon jälkeen

Testien ajojen jälkeen testien tuloksista koostetaan tekstitiedosto. Tekstitiedoston generoinnissa käytetään hyväksi TestCleanup-attribuuttia, joka suoritetaan jokaisen testin ajon jälkeen. Testin ajon jälkeen suljetaan ERP ja kasvatetaan ajettujen testien lukumäärää yhdellä. TestCleanupissa suoritetaan myös aliohjelma WriteLogs, jonka tehtävänä on tekstitiedoston kirjoitus (kuva 22).

```
<TestCleanup()> Sub MyTestCleanup()
    _testCount = _testCount + 1
    Me.UIMap.CloseERP()
    If TestContext.CurrentTestOutcome.ToString = "Failed" Then
        _failedTests = _failedTests + 1
        WriteLogs(TestContext.TestName, "Fail")
    Else
        WriteLogs(TestContext.TestName, "Success")
        _passedTests = _passedTests + 1
    End If
End Sub

2 references
Public Sub WriteLogs(Testname As String, status As String)
    Dim AppPath As String = String.Format("C:\Users\kannjan\Desktop\Log.txt")
    Dim filu As FileInfo = New FileInfo(AppPath)
    Dim writer As StreamWriter = filu.AppendText
    writer.WriteLine(Testname + "      " + status + "      " + "Aika: " + DateTime.Now.ToString())
    writer.Close()
End Sub
```

KUVA 22. Testin lopputuloksen tarkastelu ja tekstitiedoston kirjoitus

Tekstitiedosto sisältää tiedon suoritetun testin nimestä, testin lopputuloksesta ja ajankohdasta jolloin testi on ajettu (kuva 23).

```
PurchaseOrderResend      Success      Aika: 6.4.2017 10:02:31
SpecifyStorageUpdateBasedOnProductGroup      Success      Aika: 6.4.2017 10:04:17
```

KUVA 23. Tekstitiedosto testien lopputuloksesta

Tämä tiedosto lähetetään halutuille henkilöille sähköpostilla käyttäen SmtpClient-luokkaa (liite 3). Lähetettävän sähköpostin otsikoksi on määritelty testien lukumäärä, hyväksyttyjen ja hylättyjen testien lukumäärä ja Total Commerce ERP:n versionumero. Sähköpostiviestin liitteenä on edellä käyty tekstitiedosto (kuva 24).

KUVA 24. Testitulokset sähköpostiin

6 AUTOMATISOIDUN TESTAUSYMPÄRISTÖN HYÖDYT JA HAITAT

Automaatiolla saavutettavien testien hyötyjä ovat toistettavuus, säästetty aika ja virheiden minimointi. Testitapahtumia voidaan ajaa monia kertoja päivässä ja testitapahtuman pysyessä kokoajan samanlaisena ei käyttäjäkohtaisia virheitä pääse tapahtumaan. Testien toistettavuus myös parantaa ohjelman laatua. Automatisoinnin avulla työaika, jota kului ennen manuaaliseen testaukseen, voidaan nyt käyttää muuten. Automatisointi ei kuitenkaan korvaa kokonaan manuaalista testaamista. Testiympäristön avulla myös virheiden havaitsemisen todennäköisyys kasvaa. Jatkuvan testauksen seurauksena virheitä on helpompi havaita aikaisessa vaiheessa ohjelmankehityksen elinkaarta. Virheiden korjaaminen tulee kalliimmaksi, mitä pidemmällä ollaan ohjelmankehityksessä. (17.)

Automaatiotestauksella on myös haasteensa. Testausympäristö voi tuoda väärää turvallisuudentunnetta, testiympäristöä on ylläpidettävä ja testaus on implementoitava yrityksen testauskulttuuriin. Turvallisuudentunteella tarkoitetaan sitä, että luotetaan liikaa testijärjestelmän tuloksiin. Testien mennessä onnistuneesti läpi voi ohjelma siltikin sisältää virheitä, ja tämän seurauksena ei pidä sokeasti luottaa testijärjestelmän lopputuloksiin. Testiympäristöä on ylläpidettävä mahdollisten ohjelmisto muutosten seurauksena. Käyttöliittymän muuttuessa siihen kohdistuva testitapahtuma pitää tehdä uudestaan. Tämän seurauksena testitapahtumat pyritään pitämään mahdollisimman lyhyenä. Testijärjestelmä pitää myös tuoda osaksi henkilökunnan arkea. Aikaa on varattava testitapahtumien tekemiseen ja kouluttaa henkilökunta käyttämään testaukseen vaadittavia työkaluja. (17.)

7 YHTEENVETO

Opinnäytetyön tavoitteena oli toteuttaa tilaajan käyttöön automaattinen testausympäristö. Entuudestaan tilaajalla ei ollut käytössä testiympäristöä, vaan kaikki testaaminen tehtiin käsin. Testiympäristön toteutuksessa huomioitavaa oli myös, että sen pitää olla toiminnassa mahdollisimman vähällä ylläpidolla. Testausympäristöön tehtiin myös testitapahtumat testattavan tuotteen tärkeimmistä ominaisuuksista ja testitapahtumat dokumentoitiin tilaajalle. Ympäristö saatiin kokonaisuudessaan toteutettua tilaajan käyttöön ja testitapahtumia on tällä hetkellä noin 40 kappaletta.

Käyttöliittymän automaatiotestaus on aiheena hyvin laaja ja erilaisia testaukseen tarkoitettuja työkaluja löytyy paljon. Työssä käytettävään Coded Ui Test -työkaluun päädyttiin, koska se on saatavilla Visual Studioon ja Visual Studiolla kehitettiin jo entuudestaan Total Commerce ERP:tä. Valintaan vaikutti myös, että Coded Ui Test -työkalu mahdollistaa selaimen avulla internetsivujen käyttöliittymä testauksen. Tulevaisuudessa on tarkoitus toteuttaa testaus myös tilaajan asiakkuudenhallinta järjestelmään. Asiakkuudenhallinta järjestelmä on erillinen ohjelma internetissä.

Suurimmaksi ongelmaksi automatisoidussa testiympäristössä osoittautui testien ylläpidettävyys. Käyttöliittymien muuttuessa saattaa testitapahtumasta tulla epävalidi, jolloin testitapahtuma joudutaan tekemään uudelleen. Testitapahtuman uudelleen tekemisen seurauksena työaikaa joudutaan käyttämään turhaan, että saadaan testitapahtumasta taas validi.

Opinnäytetyön aikana sain syventävää tietoa testaamisesta ja kuinka tärkeää testauksen dokumentointi on. Työn aikana hyvin suunniteltu testitapahtuma osoittautui jo puoliksi tehdyksi. Jenkinsistä minulla ei ollut entuudestaan kokemusta, mutta sen selkeän käyttöliittymän avulla se osoittautui helppokäyttöiseksi. Jenkinsistä löytyi myös paljon dokumentaatiota, joka helpotti Jenkinsin käytön opettelua. Työn aikana laajensin myös ohjelmointiosaamistani ja testien ohjelmointi oli oikeastaan hieman erilaista kuin normaali VB-ohjelmointi. Haasteita käyttöliittymätestien ohjelmointiin toi se, että suurin osa internetissä dokumentoiduista testeistä oli kirjoitettu C#-kielellä, jonka seurauksena joutui hieman soveltamaan. Kokonaisuudessaan testausympäristö saatiin toteutettua valmiiksi aikataulussa ja se on otettu tilaajalla jo hyvin käyttöön.

LÄHTEET

1. Ohjelmiston testaaminen. 2017. Wikipedia. Saatavissa: https://fi.wikipedia.org/wiki/Ohjelmiston_testaaminen. Hakupäivä 07.05.2017.
2. Software Testing Techniques. Technology Maturation and Research Strategy. Lu Luo. Saatavissa: http://www.cs.cmu.edu/~luluo/Courses/17939Report.pdf. Hakupäivä 08.05.2017.
3. Sainio, Laura 2010. Ohjelmistotestauksen menetelmät ja työvälineet. Opinnäytetyö. Saimaa: Saimaan ammattikorkeakoulu, tietotekniikan koulutusohjelma. Saatavissa: https://www.theseus.fi/bitstream/handle/10024/12297/Sainio_Laura_liite1.pdf. Hakupäivä 08.05.2017.
4. Vesiputousmalli. 2017. Wikipedia. Saatavissa: https://fi.wikipedia.org/wiki/Vesiputousmalli. Hakupäivä 07.05.2017.
5. Sininen meteoriitti. Ketteryys haltuun: Ketterän kehityksen yleiset periaatteet. Saatavissa: https://www.meteoriitti.com/2013/06/06/ketteryys-haltuun-ketteran-kehityksen-yleiset-periaatteet/. Hakupäivä 22.05.2017.
6. Regression Testing Visual Studio .NET 2013. 2017. Microsoft Developer Network. Saatavissa: https://msdn.microsoft.com/en-us/library/aa292167%28v=vs.71%29.aspx. Hakupäivä 22.05.2017.
7. Visual Studio Enterprise. Microsoft. Saatavissa: https://www.visualstudio.com/vs/enterprise/. Hakupäivä 25.05.2017.
8. Installing and Configuring Test Agents and Test Controllers. Microsoft. Saatavissa: https://msdn.microsoft.com/en-us/library/dd648127(v=vs.120).aspx. Hakupäivä 12.05.2017.
9. Jenkins. Saatavissa: https://jenkins.io/download/. Hakupäivä 25.05.2017.
10. Meet Jenkins. 2016. Jenkins. Saatavissa: https://wiki.jenkins-ci.org/display/JENKINS/Meet+Jenkins. Hakupäivä 08.05.2017.
11. Windows Sysinternals. Microsoft. 2016. Saatavissa: https://technet.microsoft.com/en-us/sysinternals/bb897553.aspx. Hakupäivä 25.05.2017.
12. TortoiseSVN. 2016. Saatavissa: https://tortoisesvn.net/downloads.html. Hakupäivä 25.05.2017.

13. Google's vs Facebook's Trunk Based Development. 2014. Paul Hammant. Saatavissa: https://paulhammant.com/2014/01/08/googles-vs-facebooks-trunk-based-development/. Hakupäivä 12.05.2017.
14. Codd, E.F. 1970. A Relational Model of Data for Large Shared Data Banks.
15. Download SQL Server Management Studio (SSMS). Microsoft. 2017. Saatavissa: https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms. Hakupäivä 25.05.2017.
16. Magento. Saatavissa: https://github.com/magento. Hakupäivä 25.05.2017.
17. Fewster, Mark – Graham, Dorothy. 1999. Software Test Automation. Iso-Britannia: Ashford Colour Press.

LIITTEET

Liite 1 Testissä käytettävät aliohjelmat

Liite 2 UiMap.vb-luokka

Liite 3 Sähköpostin lähettäminen

TESTITAPAHTUMASSA KÄYTETTÄVÄT ALIOHJELMAT LIITE 1/1

```
''' <summary>
''' TME-1117
''' </summary>
<TestMethod>
0 references
Public Sub SpecifyStorageUpdateBasedOnProductGroup()

    Dim query As String = Me.UIMap.DB_GetStringValue("SELECT Arvo FROM yVakio WHERE ID = 337")
    If query.ToLower().Equals("false") Then
        ApplicationUnderTest.Launch(apppath)
        Me.UIMap.LoginToERP()
        Me.UIMap.FetchFTPProductAndOrder()
        Me.UIMap.CreateWhAndToWebStoreWH()

        Me.UIMap.OpenStockManagementForProduct("02388")
        Me.UIMap.ChangeStockValueForProduct("02388", 50)
        Me.UIMap.OpenStockManagementForProduct2("02389")
        Me.UIMap.ChangeStockValueForProduct("02389", 50)
        Me.UIMap.CloseVarastohallintaForm()

        Me.UIMap.CreatingTestTagGroup()
        Me.UIMap.CreatingTestTagGroup2()

        Me.UIMap.ChangeGroupForTestProduct()
        Me.UIMap.ChangeGroupForTestProduct2()

        Me.UIMap.TestGroupSelectWarehouseFixed()

        Me.UIMap.TakeOffWebStoreStorage()

        Me.UIMap.MoveWarehouseValueToWebStore()
        Me.UIMap.WarehouseLogAssertion()
    End If

End Sub
```

UIMAP.VB-LUOKKA


```
5 references | 0/4 passing
Public Sub ChangeStockValueForProduct(sku As String, amount As Object)
    Dim uITxtUusiSaldoEdit As WinEdit = Me.UISaldonmuutosWindow.UITxtUusiSaldoWindow.UITxtUusiSaldoEdit
    Dim uIMuutaButton As WinButton = Me.UISaldonmuutosWindow.UIMuutaWindow.UIMuutaButton

    Dim saldoCell As WinCell = FindNthDatagridviewRowCellByName(DirectCast(Me.UIPiiMegaTotaliHelpFinWindow.UIVarastohallintaWindow.UIDgvNimikeVarastoWindow.UIDataGridViewTable, WinTable), 0, "Saldo Row 0")

    If saldoCell IsNot Nothing Then
        Dim currentCellValue As String = saldoCell.Value

        Dim newValue As Integer = 50
        Integer.TryParse(amount.ToString(), newValue)

        Dim currentValue As Decimal = 0D
        Decimal.TryParse(currentCellValue, currentValue)

        If currentValue = newValue Then newValue += 1

        Mouse.Click(saldoCell, New Point(48, 11))

        'Type '50' in 'txtUusiSaldo' text box
        uITxtUusiSaldoEdit.Text = newValue.ToString()

        ' Get yVakio 316 Value
        Dim val As String = DB_GetStringValue("SELECT Arvo FROM yVakio WHERE ID = 316")

        If val.ToLower().Equals("true") Then
            Dim uITxtSyyEdit As WinEdit = Me.UISaldonmuutosWindow.UITxtSyyWindow.UITxtSyyEdit
            uITxtSyyEdit.Text = Me.StockControlChangeCauseParams.UITxtSyyEditText
        End If

        'Click 'Muuta' button
        Mouse.Click(uIMuutaButton, New Point(30, 8))
    End If
End Sub
```

SÄHKÖPOSTIN LÄHETTÄMINEN

LIITE 3/1

```
<ClassCleanup()> Public Shared Sub SendResultsEmail()
    Dim filu As String = String.Format("C:\Users\kannjan\Desktop\Log.txt")
    Dim attachment As Attachment = New Attachment(filu)
    Const DEFAULTSMTPPORT As Integer = 25I
    Using smtp As New SmtpClient()
        smtp.Host = "smtp.dnainternet.net"
        smtp.Port = DEFAULTSMTPPORT
        smtp.EnableSsl = False

        Using msg As New MailMessage()
            With msg
                .From = New MailAddress("testit@piimega.fi")
                .To.Add(New MailAddress("janne.kanniala@piimega.fi"))
                .Subject = String.Format("Automaattisten testien tulokset. Testejä: {0}, Passed: {1}, Failed: {2}. ERP:n versio: {3}",
                .Body = String.Format("Liitteenä löytyy testitulokset, testit ajettu: {0}", DateTime.Now)
                .Attachments.Add(attachment)
            End With
            Try
                smtp.Send(msg)
            Catch ex As Exception
            End Try
        End Using
    End Using
End Sub
```