

Jukka Pajarinen

# **WEB-KÄYTTÖLIITTYMÄN HYVÄKSYMISTESTAUKSEN PRIORISOINTI PAINOTETUN VERKON AVULLA**

Informaatioteknologian ja viestinnän tiedekunta

Diplomityö

Joulukuu 2019

# TIIVISTELMÄ

Jukka Pajarinen: Web-käyttöliittymän hyväksymistestauksen priorisointi painotetun verkon avulla  
Diplomityö  
Tampereen yliopisto  
Tietotekniikan DI-ohjelma  
Joulukuu 2019

---

<Lisää teksti tähän>

Avainsanat: hyväksymistestaus, painotettu verkko, priorisointi, jatkuva integraatio, testiautomaatio

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck -ohjelmalla.

# ABSTRACT

Jukka Pajarinen: Web User Interface Acceptance Testing Prioritization with a Weighted Graph  
Master's Thesis  
Tampere University  
Degree Programme in Information Technology  
December 2019

---

<Add text here>

Keywords: acceptance testing, weighted graph, prioritization, continuous integration, test automation

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

# ALKUSANAT

<Lisää teksti tähän>

Tampereella, 31. joulukuuta 2019

Jukka Pajarinen

# SISÄLLYSLUETTELO

1	Johdanto . . . . .	1
2	Tutkimusasetelma . . . . .	2
2.1	Tutkimuskysymykset . . . . .	2
2.2	Tutkimusmenetelmä . . . . .	2
3	Testiautomaatio . . . . .	4
3.1	Testiautomaation tarkoitus . . . . .	4
3.2	Testauksen tasot . . . . .	4
3.2.1	Yksikkötestaus . . . . .	4
3.2.2	Integraatiotestaus . . . . .	5
3.2.3	Järjestelmätestaus . . . . .	5
3.2.4	Hyväksymistestaus . . . . .	5
3.3	Testiautomaatio prosessina . . . . .	5
3.4	Testitapauksien määrittäminen . . . . .	5
3.5	Testitapauksien priorisointi . . . . .	6
3.6	Web-käyttöliittymien erityispiirteet . . . . .	6
3.7	Testiautomaatio jatkuvan integraation osana . . . . .	6
3.8	Hyväksymistestausvetoinen kehitys . . . . .	6
4	Verkkoteoria . . . . .	7
4.1	Matemaattisten verkkojen tarkoitus . . . . .	7
4.2	Suuntaamaton verkko . . . . .	7
4.3	Suunnattu verkko . . . . .	7
4.4	Painotettu verkko . . . . .	7
4.5	Ketjut ja polut . . . . .	7
4.6	Lyhimmän polun ongelma . . . . .	7
4.6.1	Dijkstran algoritmi . . . . .	7
4.6.2	Bellman–Ford algoritmi . . . . .	8
4.6.3	A* algoritmi . . . . .	8
5	Priorisointi painotetun verkon avulla . . . . .	9
5.1	Priorisointiin vaikuttavat muuttujat . . . . .	9
5.2	Painofunktio priorisointiin . . . . .	9
5.3	Testitapauksien koostaminen näkymistä . . . . .	9
5.4	Painotetun verkon rakentaminen . . . . .	9
5.5	Korkeiden prioriteettien polut . . . . .	9
5.6	Testikattavuuden arviointi . . . . .	9
6	Testauksen suunnittelu ja toteutus . . . . .	10

6.1	Sovelluskehykset ja työkalut . . . . .	10
6.1.1	Docker . . . . .	10
6.1.2	GoCD . . . . .	10
6.1.3	Robot Framework . . . . .	10
6.1.4	Selenium . . . . .	10
6.2	Jatkuva integraatio ja julkaisuputki . . . . .	10
6.3	Painotettu verkko ja priorisoidut polut . . . . .	10
6.4	Testitapauksien toteuttaminen . . . . .	10
6.5	Seuranta ja raportointi . . . . .	11
7	Yhteenveto . . . . .	12
	Lähteet . . . . .	13
	Liite A Esimerkkiliite . . . . .	14

## KUVALUETTELO

1.1	<Lisää kuvateksti tähän.> . . . . .	1
-----	-------------------------------------	---

## TAULUKKOLUETTELO

1.1	<Lisää taulukkoteksti tähän.> . . . . .	1
-----	---	---

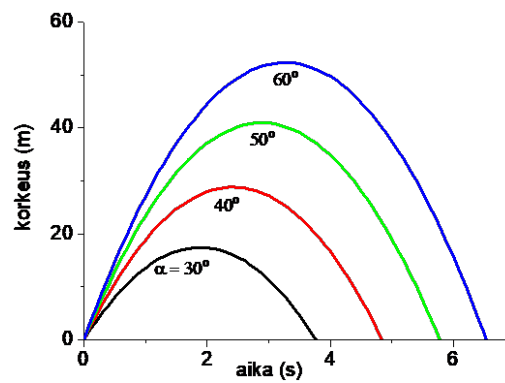


## LYHENTEET JA MERKINNÄT

lyh1 Lyhenne 1

# 1 JOHDANTO

Tässä luvussa ...



**Kuva 1.1.** <Lisää kuvateksti tähän.>

**Taulukko 1.1.** <Lisää taulukkoteksti tähän.>

	Otsikko 1	Otsikko 2
Otsikko 3	Teksti 1	Teksti 2
Otsikko 4	Teksti 3	Teksti 4

(Nawar ja Ragheb 2014)

(Zhang et al. 2007)

## 2 TUTKIMUSASETELMA

Tässä luvussa esitetään diplomityön tutkimuskysymykset sekä käytetty tutkimusmenetelmä. Tutkimuskysymykset liittyvät vahvasti yhteiseen priorisoinnin teemaan, johon tässä työssä erityisesti paneudutaan. Lisäksi työn lopussa on myös toteutuksellinen osuus, joka on tehty diplomityön asiakasyrityksen tarpeita varten. Toteutuksellisessa osuudessa on paljon muutakin sisältöä, joka on varsinaisen priorisointiteeman ulkopuolella, mutta pysyy kuitenkin työn kokonaiskontekstissa.

### 2.1 Tutkimuskysymykset

Tutkimuksen tarkoituksena on pohjimmiltaan tarkoitus löytää ja kehittää toistettavissa oleva menetelmä hyväksymistestauksen testitapauksien priorisoimiseen. Testitapauksien laatimisen yleisenä ongelmanakohtana on erityisesti niiden priorisointi, joka usein johtaa liian suppean tai kattavan testiautomaation rakentamiseen. Tutkimuskysymykset on laadittu siten, että niihin vastaaminen antaa ratkaisun edellä mainittuun testiautomaation ongelmaan.

Työlle asetettiin seuraavat tutkimuskysymykset:

- T1: *Miten painotettua verkkoa voidaan käyttää testitapauksien priorisoimiseen?*
- T2: *Mitkä muuttujat vaikuttavat web-käyttöliittymän hyväksymistestauksen testitapauksien priorisointiin?*
- T3: *Kuinka prioriteetein painotetusta verkosta valitaan toteutettavat testitapaukset?*
- T4: *Miten painotetun verkon avulla tehty priorisointi liitetään yhteen jatkuvan integraation ja testiautomaation kanssa?*

### 2.2 Tutkimusmenetelmä

Tutkimuskysymyksiin vastaamiseksi työn tutkimusmenetelmäksi valittiin ankkuroitu tutkimus, jota mielessä pitäen tutkimuksessa käytettävä aineisto kerättiin. Tarkoituksena oli muodostaa uudenlainen teoreettinen näkökulma tutkivaan asiaan, siihen liittyvää aineistoa tarkastelemalla ja jäsentämällä. Tutkimusaiheen suunnittelun alkuvaiheessa huomattiin, että olemassa olevat käsitteelliset mallit ja teoria ei ole vakiintunutta ja jäsenneiltyä, joka antaa lisää painoarvoa työn tekemiselle. Tutkittavaan asiaan liittyvää aineistoa muun muassa priorisoinnin ja painotettujen verkkojen osalta on saatavilla runsaasti, jo-

ten kyseessä ei ole niin sanottu kokonaisutkimus, vaan aineiston valitseminen perustuu harkintaan. Strategisesti työ kuuluu teoreettisen tutkimuksen piiriin ja se on myös niin sanottu perustutkimus, jossa aikaisemmin saatavilla olevaa tietoa kootaan ja jäsennellään uudestaan järkeviin tutkimuskysymyksiin vastaaviin kokonaisuuksiin. Tutkimusosuus ja teorian muodostaminen diplomityöstä on pyritty pitämään kvalitatiivisena, eli saatavilla olevaa ja teemaan liittyvää aineistoa kerättiin pitäen tutkimuksen paino määrän sijaan laadussa. Aineiston hallintaan käytettiin tietokoneohjelmistoa, jossa aineisto kategorisoi-  
ttiin eri loogisiin kokonaisuuksiin muun muassa priorisoinnin ja painotetun verkon osalta. Kerätyn aineiston avulla pyrittiin luomaan mahdollisimman vahva teoreettinen pohja tutkimuskysymyksiin vastaamiseksi mahdollisimman kattavasti.

## 3 TESTIAUTOMAATIO

Tässä luvussa pyritään esittämään perusteet ja tarvittavat tiedot testiautomaatiosta, jotka liittyvät työn laajempaan teoreettiseen kehykseen. Testiautomaation perusteiden ymmärtämistä tarvitaan työn myöhemmässä vaiheessa, jossa esitetään varsinainen testitapauksien priorisointi painotetun verkon avulla.

### 3.1 Testiautomaation tarkoitus

Testiautomaation tarkoitus on pohjimmiltaan mahdollistaa ohjelmistotuotteen jatkuva ja vaivaton laadunvarmistus, nyt ja tulevaisuudessa. Ohjelmistojen testaamisella itsessään pyritään löytämään ohjelmistotuotteesta virheitä, anomalioita ja varmistamaan että se toimii asetettujen vaatimusten mukaisesti. Testauksen automatisoiminen vapauttaa henkilöresursseja manuaalisesta testaamisesta muihin tuotantotehtäviin sekä parantaa toistuvien testien luotettavuutta poistamalla manuaalisessa testauksessa tapahtuvat inhimillisen virheet. Laadunvarmistuksen osalta testiautomaatiolla voidaan kattaa erilaisia ohjelmistotuotteen laadullisia ominaisuuksia, kuten toiminnallisuus, luotettavuus, käytettävyys, tietoturva, tehokkuus, ylläpidettävyys ja siirrettävyys [ISO 9126].

### 3.2 Testauksen tasot

Testauksen tasoja on lukuisia ja usein ohjelmistojen testaamiseksi on suositeltavaa käyttää eri tasojen yhdistelmää. Ohjelmistotestaus usein jaotellaan kolmeen erillaiseen menetelmään, jotka myös vaikuttavat eri testauksen tasojen käytettävyyteen. Erilaisia menetelmiä ovat mustalaatikkotestaus, harmaalaatikkotestaus ja valkoolaatikkotestaus, jotka eroavat toisistaan yleisesti ottaen siinä, otetaanko tieto ohjelmistotuotteen sisäisestä toteutuksesta mukaan testaamiseen.

#### 3.2.1 Yksikkötestaus

Yksikkötestauksen ajatuksena on testata ohjelmistotuotteen lähdekoodista löytyviä yksiköitä, kuten luokkia, funktioita tai moduleita. Yksikkötestaus toteutetaan ohjelmiston toteuttavia pienempiä yksikköjä vastaan. Yksikkötestaus eroaa muista testauksen tasoista siinä, että sen suorittavat ohjelmistokehittäjät tai ohjelmiston lähdekoodiin perehtyneet henkilöt. Yksikkötestauksella pyritään varmistamaan, että ohjelmiston pienimmät osat toi-

mivat tarkoituksenmukaisella tavalla. Yksikkötestausta hyödynnetään usein myös ketterien menetelmien aihepiirissä, jossa ohjelmistotuotantoa toteutetaan niin sanotulla testivetoisella kehityksellä. Testivetoisessa kehityksessä ohjelmistokehittäjät laativat ensisijaisesti yksiköiden yksikkötestit ennen niiden toteuttamisen aloittamista.

### **3.2.2 Integraatiotestaus**

Integraatiotestauksen ajatuksena on testata ohjelmistotuotteen toteuttavien eri komponenttien yhteentoimivuutta niiden rajapintojen osalta. Integraatiotestaus toteutetaan ohjelmiston suunnitelmaa ja mallia vastaan. Integraatiotestauksen onnistuminen luo perustan ohjelmiston toimimiseen ja koostamiseen kokonaisuena, eri komponenteista koostuvana järjestelmänä. Integraatiotestauksen yhteydessä puhutaan usein myös niin sanotusta savutestauksesta, jonka tarkoituksena on koostaa päivittäinen koontiversio ohjelmistosta ja testata sen kriittisten komponenttien yhteentoimivuus.

### **3.2.3 Järjestelmätestaus**

Järjestelmätestauksen ajatuksena on testata kokonaista ja toimivaa järjestelmää, yhtenä suurena yksikkönä. Järjestelmätestaus toteutetaan usein eräänlaisena tulikokeena, erityisesti ohjelmiston vaatimuksia vastaan. Järjestelmätestaukseen liittyy laajasti erilaisia testattavia laadullisia ominaisuuksia, kuten esimerkiksi toiminnallisuus, luotettavuus, käytettävyys, tietotruvu, tehokkuus ja siirrettävyys.

### **3.2.4 Hyväksymistestaus**

Hyväksymistestauksen ajatuksena on varmistaa toteutettavan ohjelmiston vaatimusten toimivuus erityisesti käytännön tilanteissa. Hyväksymistestaus toteutetaan ohjelmiston toimintoja kuvaavaa vaatimusmäärittelyä vastaan. Hyväksymistestaus on tarkoituksenmukaista laatia sellaiseen muotoon joka testaa lopullisten käyttäjien toimintaa vastaavia käyttötilanteita. Testiautomaatio on erittäin hyödyllinen hyväksymistestausen osalla, koska sillä voidaan automatisoida ohjelmiston validointi ja hyväksyminen sekä estää puutteellisesti toimivan ohjelmiston julkaiseminen.

## **3.3 Testiautomaatio prosessina**

Testiautomaation prosessiin kuuluu erilaisia artefakteja, joita luodaan testausprosessin eri vaiheissa. Eri vaiheita ovat kronologisessa järjestyksessä ovat testisuunnitelma, skenaariot, testitapaukset ja seuranta.

## **3.4 Testitapauksien määrittäminen**

<Yleiset heuristiikat>

### 3.5 Testitapauksien priorisointi

Testitapauksien priorisointi on kustannussyistä tai resurssien optimoinnin kannalta erittäin tärkeää. Ohjelmistotestauksessa on hyvä tiedostaa, että ohjelmistotuotetta ei usein voida testata täydellisesti, joka nostaa esiin tarpeen tärkeimpien testitapauksien löytämisestä. Testitapauksia voidaan priorisoida monella tavalla, joihin tämä diplomityö tuo yhden uuden painottua verkkoa hyödyntävän lähestymistavan.

<Painotetun verkon hyödyntäminen>

<Muut priorisointitavat>

### 3.6 Web-käyttöliittymien erityispiirteet

<Testitapauksien määrittämiseen vaikuttavat erityispiirteet>

<Priorisointiin vaikuttavat erityispiirteet>

### 3.7 Testiautomaatio jatkuvan integraation osana

<Lisää teksti tähän>

### 3.8 Hyväksymistestausvetoinen kehitys

Hyväksymistestausvetoinen kehityksen (englanniksi: ATDD, Acceptance test-driven development) tarkoituksena, kuten testausvetoisessakin kehityksessä on toteuttaa ohjelmistotuotannollinen prosessi testaaminen edellä. Tämä tarkoittaa käytännössä, sitä että ohjelmistokehittäjät laativat ohjelmiston vaatimusten ja suunnitelman mukaisia iteratiivisesti suoritettavia testitapauksia, ennen niitä käyttävän varsinaisen ohjelmakoodin toteuttamista. Hyväksymistestausvetoisessa kehityksessä luodaan ennen toteutusta tarvittavat ohjelmiston asiakasvaatimuksia palvelevat hyväksymistestit, joiden ohjelmiston on tarkoitus läpäistä. Tarvittavat ohjelmiston hyväksymistestit suoritetaan iteratiivisesti ohjelmistokehitysprosessin aikana, ja se tarkoittaa käytännössä jatkuvan integraation ottamista käyttöön ohjelmistokehityksessä. Hyväksymistestausvetoinen kehitys on erittäin hyödyllinen ohjelmistokehityksessä käytetty menetelmä, sillä kehitysvaiheessa on aina tarkasti tiedossa vastaako ohjelmiston silloinen tila asiakasvaatimuksia ja kuinka hyvin. Hyväksymistestausvetoisessa kehityksessä toteutettavat hyväksymistestit testaavat ohjelmistoa kokonaisuutena järjestelmänä tarkoituksen mukaisesti siten kuten se esiintyy loppukäyttäjille.

## **4 VERKKOTEORIA**

Tässä luvussa käsitellään työhön keskeisesti kuuluvan verkkoteorian perusteet käydään huolellisesti läpi erityisesti työssä käytettävät osat.

### **4.1 Matemaattisten verkkojen tarkoitus**

<Lisää teksti tähän>

### **4.2 Suuntaamaton verkko**

<Lisää teksti tähän>

### **4.3 Suunnattu verkko**

<Lisää teksti tähän>

### **4.4 Painotettu verkko**

<Lisää teksti tähän>

### **4.5 Ketjut ja polut**

<Lisää teksti tähän>

### **4.6 Lyhimmän polun ongelma**

<Lisää teksti tähän>

#### **4.6.1 Dijikstran algoritmi**

<Lisää teksti tähän>



### 4.6.2 Bellman–Ford algoritmi

<Lisää teksti tähän>

### 4.6.3 A\* algoritmi

<Lisää teksti tähän>

## **5 PRIORISOINTI PAINOTETUN VERKON AVULLA**

Tässä luvussa ...

### **5.1 Priorisointiin vaikuttavat muuttujat**

<Lisää teksti tähän>

### **5.2 Painofunktio priorisointiin**

<Lisää teksti tähän>

### **5.3 Testitapauksien koostaminen näkymistä**

<Lisää teksti tähän>

### **5.4 Painotetun verkon rakentaminen**

<Lisää teksti tähän>

### **5.5 Korkeiden prioriteettien polut**

<Lisää teksti tähän>

### **5.6 Testikattavuuden arviointi**

<Lisää teksti tähän>

## 6 TESTAUKSEN SUUNNITTELU JA TOTEUTUS

Tässä luvussa ...

### 6.1 Sovelluskehykset ja työkalut

<Lisää teksti tähän>

#### 6.1.1 Docker

<Lisää teksti tähän>

#### 6.1.2 GoCD

<Lisää teksti tähän>

#### 6.1.3 Robot Framework

<Lisää teksti tähän>

#### 6.1.4 Selenium

<Lisää teksti tähän>

### 6.2 Jatkuva integraatio ja julkaisuputki

<Lisää teksti tähän>

### 6.3 Painotettu verkko ja priorisoidut polut

<Lisää teksti tähän>

### 6.4 Testitapauksien toteuttaminen

<Lisää teksti tähän>

## 6.5 Seuranta ja raportointi

<Lisää teksti tähän>

## 7 YHTEENVETO

<Lisää teksti tähän>

<Mihin suuntaan testitapausten priorisointi on menossa>

<Menetelmän järkevyys ja toistettavuus>

## LÄHTEET

- Nawar, M. N. ja Ragheb, M. M. (2014). Multi-heuristic Based Algorithm for Test Case Prioritization. *Computational Science and Its Applications – ICCSA 2014*. Toim. B. Murgante, S. Misra, A. M. A. C. Rocha, C. Torre, J. G. Rocha, M. I. Falcão, D. Taniar, B. O. Apduhan ja O. Gervasi. Lecture Notes in Computer Science. Springer International Publishing, 449–460. ISBN: 978-3-319-09156-3.
- Zhang, X., Nie, C., Xu, B. ja Qu, B. (lokakuu 2007). Test Case Prioritization Based on Varying Testing Requirement Priorities and Test Case Costs. *Seventh International Conference on Quality Software (QSIC 2007)*. Seventh International Conference on Quality Software (QSIC 2007), 15–24. DOI: 10.1109/QSIC.2007.4385476.

## A ESIMERKKILIITE

<Lisää teksti tähän>