Tampereen yliopisto

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

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

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

ALKUSANAT

Tampereella, 31. joulukuuta 2019

Jukka Pajarinen

SISÄLLYSLUETTELO

KUVALUETTELO

TAULUKKOLUETTELO

LYHENTEET JA MERKINNÄT

$\mathbb{N}$	Luonnolisten lukujen joukko
ATDD	Hyväksymistestausvetoinen kehitys, englanniksi: acceptance test driven development
e2e	Päästä päähän testaus, englanniksi: end-to-end

1 JOHDANTO

Tässä luvussa esitetään lyhyesti työn keskeisin sisältö ja rakenne. Lisäksi pohditaan miksi työ on tarpeellinen ja miksi testitapauksien priorisoiminen on ongelmallista.

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 ylikattavan 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 diplomitöissä yleisesti käytetty design science menetelmä. Tarkoituksena oli muodostaa uudenlainen toistettavissa oleva menetelmä tutkimuksen kohteena olevan ongelman ratkaisemiseksi. Tutkimusidean hahmottelemisen jälkeen, valittua tutkimusenetelmää käyttäen ensin määriteltiin tutkimuskysymykset 2.1. Seuraavaksi kartoitettiin ratkaisuvaihtoehdot tutkimuskysymyksiin ja valittiin sopivimmat ratkaisut perustelut esittäen. Lisäksi sitten esitettiin yhtenäinen ratkaisu kokonaisuudessaan tutkittavan aiheen osalta. Asiakasyrityksen ohjelmis-

totuotetta silmälläpitäen toteutettiin kokonaisratkaisu. Lopuksi vielä evaluoitiin ratkaisun toimivuus ja esitetään yhteenveto tutkimuksesta.

Tutkimusaiheen suunnittelemisen alkuvaiheessa huomattiin, että olemassa olevat käsitteelliset mallit ja teoria eivät olleet kovin vakiintunutta ja jäsenneltyä, joka antoi lisää painoarvoa työn tekemiselle. Tutkittavaan asiaan liittyvää aineistoa muun muassa priorisoinnin ja painotettujen verkkojen osalta on saatavilla runsaasti, joten aineiston valitseminen perustui harkintaan. Aineston hallintaan käytettiin tietokoneohjelmistoa, jossa aineisto kategorisoitiin eri loogisiin kokonaisuuksiin muun muassa priorisoinnin ja painotetun verkon osalta. Kerätyn aineiston avulla pyrittiin luomaan mahdollisimman vahva teoreettinen pohja tutkimuskysymyksiin vastaamiseksi mahdollisimman kattavasti.

2.3 Tutkimussuunnitelma

3 TESTIAUTOMAATIO

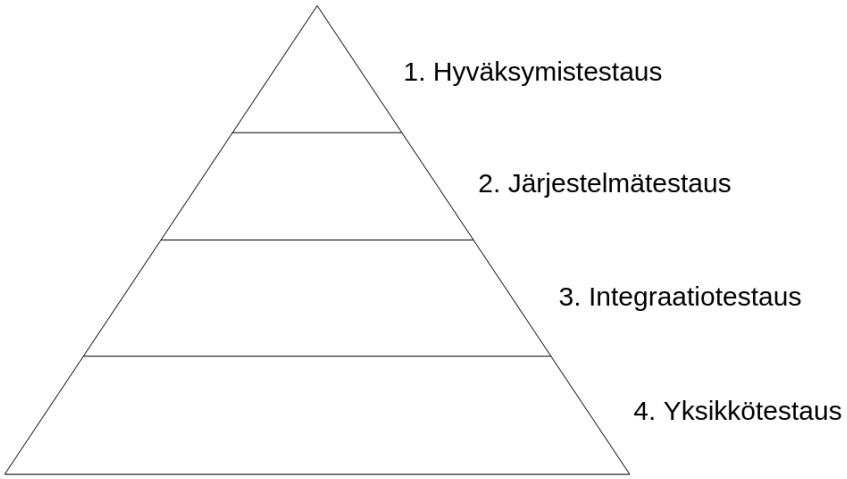
Tässä luvussa esitetään perusteet ja tarvittavat tiedot testiautomaatiosta, jotka liittyvät työn laajempaan teoreettiseen kehykseen. Testiautomaation perusteiden ymmärtämistä tarvitaan työn myöhemmässä vaiheessa, jossa esitetään varsinainen testitapauksien priorisointi painotetun verkon avulla.

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, tehokkuus, ylläpidettävyys ja siirrettävyys (ISO:9126-1 2001).

3.2 Testauksen tasot

Testauksen tasoja on useita ja usein ohjelmistojen kattavaan testaamiseen on suositeltavaa käyttää ohjelmistotuotantoprosessissa eri tasojen yhdistelmää. Ohjelmistotestaus usein jaotellaan kolmeen erilaiseen menetelmään, jotka myös vaikuttavat eri testauksen tasojen käytettävyyteen. Erilaisia menetelmiä ovat mustalaatikkotestaus, harmaalaatikkotestaus ja valkolaatikkotestaus, jotka eroavat toisistaan yleisesti ottaen siinä, otetaanko tieto ohjelmistotuotteen sisäisestä toteutuksesta mukaan testaamiseen. Testauksen tasot voidaan jakaa neljään eri tasoon, jotka usein kuvataan pyramidin muotoon. Pyramidimuodossa alimpana kuvataan yksikkötestaus, joka luo vahvan pohjan kokonaisvaltaiselle testaamiselle. Noustessa pyramidissa ylöspäin testattavana olevan kohteen laajuus kasvaa. Ylimpänä pyramidissa on hyväksymistestaus, joka on tarkoituksellista toteuttaa vaatimusmäärittelyn täyttävää valmista järjestelmää vastaan.



Kuva 3.1. *Testauksen tasot pyramidin muodossa*

3.2.1 Yksikkötestaus

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

sia testattavia laadullisia ominaisuuksia, kuten esimerkiksi toiminnallisuus, luotettavuus, käytettävyys, tietotruva, tehokkuus ja siirrettävyys.

3.2.4 Hyväksymistestaus

Hyväksymistestauksen ajatuksena on varmistaa toteutettavan ohjelmiston vaatimusten toimivuus erityisesti käytännön tilanteissa. Hyväksymistestaus toteutaan ohjelmiston toimintoja kuvaavaa vaatimusmäärittelyä vastaan. Hyväksymistestaus on tarkoituksenmukaista laatia sellaiseen muotoon joka testaa lopullisten käyttäjien toimintaa vastaavia käyttötilanteita. Samassa asiayhteydessä puhutaan usein myös niin sanotusta päästä päähän testauksesta (englanniksi: e2e, end-to-end). Testiautomaatio on erittäin hyödyllinen hyväksymistestausen osalla, koska sillä voidaan automatisoida ohjelmiston validointi ja hyväksyminen sekä estää puutteellisesti toimivan ohjelmiston julkaiseminen.

4 HYVÄKSYMISTESTAUS

Tässä luvussa esitetään perusteet ja tarvittavat tiedot hyväksymistestauksesta..

4.1 Testiautomaatio prosessina

4.2 Testausvetoinen kehitys

4.3 Hyväksymistestausvetoinen kehitys

Hyväksymistestausvetoisen 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. Hyväksysmistestausvetoinen 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 kokonaisena järjestelmänä tarkoituksenamukaisesti, siten kuten se esiintyy loppukäyttäjille.

4.4 Testiautomaatio ja jatkuva integraatio

5 TESTITAPAUKSET

Tässä luvussa esitetään perusteet ja tarvittavat tiedot testitapauksista..

5.1 Mikä on testitapaus?

5.2 Testitapauksien määrittäminen

Yleisiä testitapauksien määrittämiseen käytettäviä heuristiikkoja ovat muun muassa:

- Polut ja tiedostot
- Aika ja päivämäärät
- Numerot
- Merkkijonot
- Yleiset rikkeet
- Muuttujien analyysi
- Kosketuspisteet
- Rajat
- CRUD-toiminnot
- Datan eheys
- Konfiguraatiot
- Katkokset
- Nälkiintyminen
- Samanaikaiset käyttäjät
- Transaktiotulvat
- Riippuvuudet
- Rajaehdot
- Syötetyypit
- Tilan analyysi
- Käyttäjät ja skenaariot

5.3 Web-käyttöliittymien erityispiirteet

Web-käyttöliittymillä on myös omia erityispiirteitä, jotka vaikuttavat testitapauksien laatimiseen.

- Navigointi
- Syötteet
- Syntaksi
- Selainasetukset

5.4 Priorisointiongelma

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 uudenlaisen painottua verkkoa hyödyntävän lähestymistavan.

- Painotetun verkon hyödyntäminen
- Muut priorisointitavat

6 VERKKOTEORIA

Tässä luvussa käsitellään työhön keskeisesti kuuluvan verkkoteorian perusteet käydää huolellisesti läpi erityisesti työssä käytettävät osat. Työssä sovelletaan erityisesti verkkoteorian painotettua verkkoa sekä verkkoteoriassa esiintyvän lyhimmän polun ongelmaan kehitettyjä ratkaisualgoritmeja. Verkkoteoria itsessään on osa diskeettiä matematiikkaa.

6.1 Matemaattisten verkkojen tarkoitus

Matemaattisten verkkojen tarkoituksena on mallintaan parittaisia riippuvuuksia verkkomaisessa objektijoukossa. Verkkoteoriassa peruskäsitteitä ovat itse *verkko* eli *graafi*, joka muodostuu *solmuista* ja niiden välisiä riippuvuuksia esittävistä *kaarista* tai *nuolista*. Verkkoteorialla on lukuisia käytännön sovellutuksia. Verkkoteoriaa sovelletaan muun muassa tietokonetieteissä, kielitieteissä, fysiikan ja kemian sovellutuksissa, sosiaalisissa tieteissä ja biologiassa. Alun perin verkkoteoria katsotaan syntyneen 1700-luvulla esiintyneestä niin sanotusta Königsbergin siltaongelmasta, johon Leonhard Euler esitti todistuksensa.

6.2 Perusmerkinnät ja käsitteet

Verkkoteoriassa käytetään seuraavia perusmerkintöjä:

- $V := \{v_1, v_2, v_3\}$ Solmujoukko joka sisältää *solmut* v_1, v_2 ja v_3.
- $E := \{e_1, e_2, e_3\}$ Kaarijoukko joka sisältää *kaaret* e_1, e_2 ja e_3.
- $\phi(e_1) := \langle v_1, v_2 \rangle$ Solmuja v_1 ja v_2 yhdistävän *kaaren* e_1 kuvaaja.

Verkkojen solmujen välisiä yhteyksiä, eli kaaria esitetään usein myös yhteys- tai painomatriisina.

$$M_G = (a_{ij})_{3\times 3} = \begin{array}{c} G \\ v_1 \\ v_2 \\ v_3 \end{array} \begin{array}{c} \begin{array}{ccc} v_1 & v_2 & v_3 \end{array} \\ \begin{pmatrix} 1 & 1 & 2 \\ 1 & 0 & 0 \\ 2 & 0 & 0 \end{pmatrix} \end{array}$$

Verkkoteoriassa käytetään myös muun muassa seuraavia käsitteitä:

- Solmun asteluku, $d_G(x)$, eli solmuun liittyvien *kaarten* määrä.
- Surkastunut verkko, $E = \varnothing$, eli verkko jossa ei ole *kaaria*.
- Täydellinen verkko, eli jokaista solmuparia $v_1 \neq v_2$ yhdistää ainakin yksi *kaari*.

- Aliverkko, $G_2 \subset G_1$, eli *verkko* G_2 joka koostuu osasta *verkon* G_1 *solmuja* ja *kaaria*.
- Verkon komplementti, G', eli sellainen *verkko*, jossa on kaikki ne *kaaret* joita *verkossa* G ei esiinny.
- Verkon yhtenäisyys, $v_1 \neq v_2, v_1 \rightarrow v_2$, eli jokaiselle solmuparille $v_1 \neq v_2$ on olemassa niitä yhdistävä *kaari*.
- Polku, $\{v_0, v_1, ..., v_n\}, v_0 \rightarrow v_n$, eli *suunnattu solmujono* jota pitkin voidaan kulkea *solmusta* v_0 *solmuun* v_n.
- Eristetty solmu, $d_G(v_1) = 0$, eli *solmu* jonka *asteluku* on nolla.
- Silta, $v_1 \rightarrow v_2, d_G(v_1) = 1 \vee d_G(v_2) = 1$, eli *kaari* johon yhdistyvän *solmun asteluku* on yksi ja jonka poistaminen epäyhteinäistää *verkon*.

6.3 Painotettu verkko

- $\alpha := V(G), E(G) \rightarrow \mathbb{N}$ Painofunktion yleinen kuvaus.

6.4 Verkon leikkaaminen

6.5 Lyhimmän polun ongelma

- $d_G^{\alpha}(v_1, v_2) = min\{\alpha(P)|P : v_1 \rightarrow v_2|v_1, v_2 \in V(G)\}$ Lyhimmän polun ongelma.

6.5.1 Dijkstran algoritmi

7 PRIORISOINTI PAINOTETUN VERKON AVULLA

Tässä luvussa esitetään tutkimuksen tärkein sisältö, eli toistettavissa oleva menetelmä testitapauksien priorisoimiseen. Priorisointia varten esitetään harkintaa käyttäen lähdeaineistosta suodatetut priorisointiin vaikuttavat muuttujat, painofunktio, testitapauksien näkymäperusteinen koostaminen ja painotetun verkon laatiminen. Lisäksi menetelmää käyttäen tuotetun painotetun verkon sisältämää informaatiota käytetään prioriteeteiltaan tärkeiden polkujen löytämiseen ja testikattavuuden arviointiin.

7.1 Priorisointiin vaikuttavat muuttujat

- Liiketoiminnallinen arvo
- Liiketoiminnallinen visio
- Projektin muuttumisen volatiliteetti
- Kehittämisen kompleksisuus
- Vaatimusten taipumus virheellisyyteen

7.2 Painofunktio priorisointiin

Painofunktion yleinen kuvaus verkossa G, solmuille V ja kaarille E.

$$\alpha := V(G), E(G) \rightarrow \mathbb{N}$$

Painofunktio yksittäiselle solmulle v, eli näkymälle.

$$\alpha(v) = value + vision - volatility - complexity - errorness$$

Painofunktio yksittäiselle kaarelle e, eli siirtymälle.

$$\beta(e) = value - volatility - complexity - errorness$$

Painofunktion polulle P solmusta v_1 solmuun v_2.

$$\gamma(P) = \sum_{v \in P} \alpha(v) + \sum_{e \in P} \beta(e)$$

7.3 Käyttöliittymän näkymät ja siirtymät

7.4 Painotetun verkon rakentaminen

$$M_G = (a_{ij})_{3\times 3} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \end{array} \begin{array}{c} \begin{array}{ccc} G & v_1 & v_2 & v_3 \end{array} \\ \begin{pmatrix} 1 & 1 & 2 \\ 1 & 0 & 0 \\ 2 & 0 & 0 \end{pmatrix} \end{array}$$

7.5 Painotetun verkon karsiminen

Painotetun verkon karsiminen eli leikkaaminen on prioriteeillä painotetun verkon tärkeä ominaisuus. Verkkoteorian soveltaminen prioriteettien avulla painotettuun verkkoon on erityisen hyödyllistä, kun verkon kaarissa korkea paino tarkoittaa suurta prioriteettia. Tällaisessa tapauksessa on mahdollista soveltaa lyhimmän polun ongelman ratkaisemiseen kehitettyjä algoritmeja, jolloin ne toimivat etsien alhaisimman prioriteetin polkuja. Lyhimmän polun etsimiseen on tarkoituksenmukaista valita aina aloitus ja lopetuspisteet, joiden välille lyhin polku verkossa voidaan etsiä. Prioriteetein painotetun verkon karsimistarkoitukseen olisi järkevää valita sellaiset aloitus- ja lopetuspisteet, joiden välillä ei näyttäisi olevan korkean prioriteetin solmuja. Voidaan kuitenkin menetellä myös siten, että valitaan aloitus- ja lopetuspisteeksi sellaiset solmut, jotka ovat painoltaan verkon alhaisimmat $v_1 = min(V)$ ja $v_2 = min(V \setminus \{v_1\})$ ja esimerkiksi verrata niiden lyhimmän polun kokonaisprioriteettia muuhun verkkoon.

7.5.1 Dijkstran algoritmin soveltaminen

- Pienimmän prioriteetin solmuparin etsiminen, eli $v_1 = min\{\alpha(V)\}$ ja $v_2 = min\{\alpha(V \setminus \{v_1\}\}$.
- Dijkstran algoritmin käyttö lyhimmän (prioriteetiltaan pienimmän) polun löytämiseen, eli $s = min(\gamma(P)), P \in G$.
- Leikkauksien tekeminen ja toistaminen n-kertaa.

7.5.2 Leikkauksien tekeminen

- Poistetaan yhden yksittäiseen solmuun johtavat sillat, jossa $\alpha(v) < aivanliianalhainen$.
- Poistetaan kaikki yksittäiset eristetyt solmut, eli asteluku on nolla $d_G(X) = 0$.
- Poistetaan Dijkstran lyhimmän polun kaaret, jossa $\beta(e) < liianalhainen$.

7.6 Testitapauksien muodostaminen

8 TESTAUKSEN SUUNNITTELU JA TOTEUTUS

Tässä luvussa esitetään työn perusteella tehty esimerkkitoteutus sekä käytännön sovelluskehyksen ja työkalut. Tämän luvun tarkoituksena on todistaa menetelmän toimivuus oikeassa ohjelmistotuotannon ympäristössä toteuttaen samalla testiautomaatio asiakasyritykselle.

8.1 Sovelluskehykset ja työkalut

8.1.1 Docker

8.1.2 GoCD

8.1.3 Robot Framework

8.1.4 Selenium

8.2 Jatkuva integraatio ja testiautomaatio

8.3 Painotetun verkon käyttö priorisoimiseen

8.4 Testitapauksien toteuttaminen

9 YHTEENVETO

Tässä kappaleessa esitetään yhteenveto tutkimuksen tuloksista ja pohditaan kuinka hyvin toistettavissa oleva kyseinen kehitetty menetelmä on. Tutkimusta verrataan myös muuhun verrattavissa olevaan tutkimusaineistoon aiheesta, ja pohditaan mihin suuntaan testitapauksien priorisoimisen tutkimustyö on menossa.

LÄHTEET

ISO:9126-1 (2001). *ISO/IEC 9126-1:2001*. en. URL: http://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/02/27/22749.html.

A ESIMERKKILIITE