Web-käyttöliittymän hyväksymistestauksen priorisointi painotetun verkon avulla

Diplomityön seminaariesitys
25.11.2019
Jukka Pajarinen

Yleiskatsaus

1. Web-käyttöliittymien hyväksymistestauksen automatisoimisen mahdollistava järjestelmä.
2. Painotettua verkkoa hyödyntävä, toistettavissa oleva hyväksymistestauksen priorisointimenetelmä.

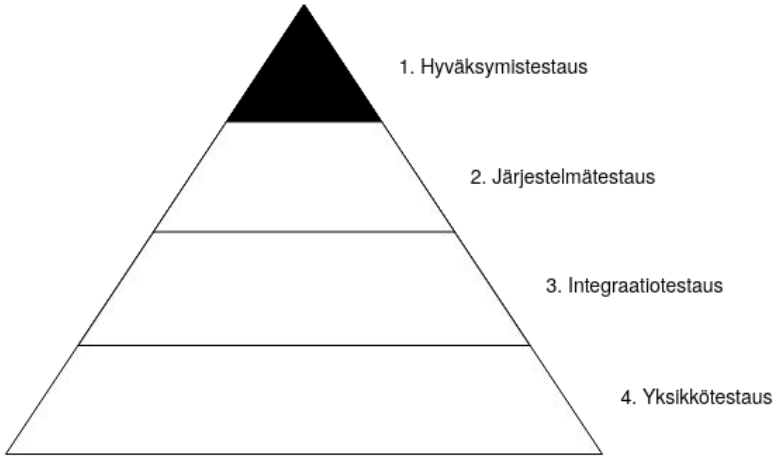
Motiivi

- Diplomityö WordDive nimiselle yritykselle.
- Web-sovelluksen hyväksymistestausta ei oltu vielä toteutettu.
- Koodimuutoksien jälkeen jatkuva tarve käyttöliittymän toimintojen verifioimiseen.
- Testitapauksien priorisoimiseen konkreettinen tarve.
- Testiautomaation rakentamiseen rajallinen määrä resursseja.

Testausjärjestelmä

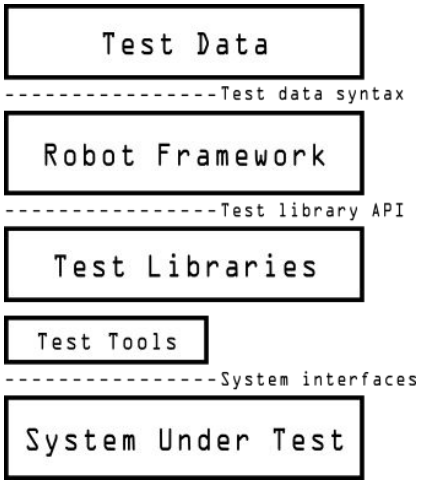
Testauksen tasot

- Testausjärjestelmä mahdollistaa hyväksymistestauksen automatisoimisen.
- Konkreettinen tarve käyttöliittymän testaamisessa.



Työkalut 1/5: Robot Framework

- Testausalusta hyväksymistestaukseen ja hyväksymistestausvetoisen kehitykseen.
- Helposti ymmärrettävä testitapauksien syntaksi.
- Kattavat testiraportit.
- Hyvä tuki ulkoisille kirjastoille.



Työkalut 2/5: Selenium

- Työkalu ja kirjastokokoelma verkkoselainten automatisoimiseen.
- Ensisijaisesti web-sovelluksien automatisoimiseen testaustarpeisiin.
- Seleniumin avulla voidaan esimerkiksi **syöttää tekstiä tekstikenttään** tai **klikata painiketta** web-sovelluksessa.
- SeleniumLibrary on Robot Frameworkiin saatava kirjasto, joka tarjoaa avainsanat Seleniumin käyttämiseen testitapauksissa.

Työkalut 3/5: Xvfb

- Xvfb = X Virtual Framebuffer.
- X-näyttöpalvelimen protokollan toteuttava virtuaalinen näyttöpalvelin.
- Ei tulosta mitään näytölle vaan toimii kuten X-näyttöpalvelin, mutta tietokoneen muistissa.
- Mahdollistaa päätteettömän testauksen mille tahansa GUI-sovelluksille.
- Mahdollisuus ottaa myös kuvankaappauksia.

Työkalut 4/5: Docker

- Sovelluksien säiliöintityökalu (englanniksi: containerization).
- Mahdollisuus rakentaa, koota ja ajaa säiliöiden muotoon konfiguroituja sovelluksia.
- Säiliöiden pystyttäminen uusiin ympäristöihin vaivatonta.
- Dockerfilen avulla voi luoda säiliöidyn sovelluksen.
- Docker-composen avulla voi luoda verkon joka koostuu useista säiliöistä.
- Valmiissa testausjärjestelmässä toimii “liimana”.

Työkalut 5/5: GoCD

- Jatkuvan integroinnin ja jatkuvan julkaisemisen palvelinohjelmisto.
- Koonti, testaus ja julkaisuputkien rakentamiseen.
- Palvelin ja agentit arkkitehtuuri.
- Mahdollistaa toistuvan testauksen ohjelmistotuotantoprosessissa.
- Tallettaa testausjärjestelmän testiraportit ja ruudunkaappaukset.

Priorisointimenetelmä

Priorisointiongelma

- Miten ohjelmiston hyväksymistestaus voidaan priorisoida?
- Tärkeät toiminnot pitää testata huolellisemmin.
- Ajan myötä ohjelmiston toiminnot muuttuvat ja testaus vanhenee.
- Kustannussyyt ja resurssien optimointi.
- Tärkeät ongelmat pitäisi löytää aikaisin.
- Testitapauksien suorittaminen pitäisi järjestää.

Priorisointiin vaikuttavat muuttujat

m	Muuttuja	Etumerkki	Asteikko
1	Liiketoiminnallinen arvo	$+$	1 - 10
2	Liiketoiminnallinen visio	$+$	1 - 10
3	Negatiivinen käyttäjäpalaute	$+$	1 - 5
4	Käyttötapauksien määrä	$+$	10 · suhde
5	Siirtymien määrä	$+$	10 · suhde
6	Positiivinen käyttäjäpalaute	$-$	1 - 5
7	Muutosherkkyys	$-$	1 - 10
8	Toteuttamisen kompleksisuus	$-$	1 - 5
9	Toteutuksen virheherkkyys	$-$	1 - 5
10	Omat lisämuuttujat	$\pm$	1 - 10

Painofunktiot

1. Prioriteettifunktio, yhdistää priorisointiin osallistuvat muuttujat.

$$p(v) = \sum_{i=1}^{5} m_i - \sum_{j=6}^{9} m_j \pm m_{10}$$

2. Solmun painofunktio, hyödyntää prioriteettifunktiota ja käänteislukuja.

$$\alpha(v) = \begin{cases} p^{-1}(v) & p(v) > 0 \\ 1 & p(v) \leq 0 \end{cases}$$

3. Kaaren painofunktio, hyödyntää prioriteettifunktiota ja käänteislukuja.

$$\beta(e_{xy}) = \begin{cases} (p(v_x) + p(v_y))^{-1} & p(v_x) + p(v_y) > 0 \\ 1 & p(v_x) + p(v_y) \leq 0 \end{cases}$$

Esimerkkiverkko

n	**Näkymä**	**Siirtymät**	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	$p(n)$
A	Kirjautuminen	B	10	10	0	2	1	0	5	5	5	8
B	Pelivalikko	A, C, D, G	8	10	1	2	4	4	5	5	5	6
C	Asetukset	A, B	4	6	5	2	2	2	5	5	5	2
D	Peli	B, E, G	10	10	4	2	3	4	4	5	5	11
E	Tulokset	B, D, F	6	8	0	2	3	5	5	4	5	2
F	Onnittelu	B, E	1	8	0	0	2	2	5	2	5	-3
G	Ohje	B, D	1	10	2	0	2	0	8	0	0	7

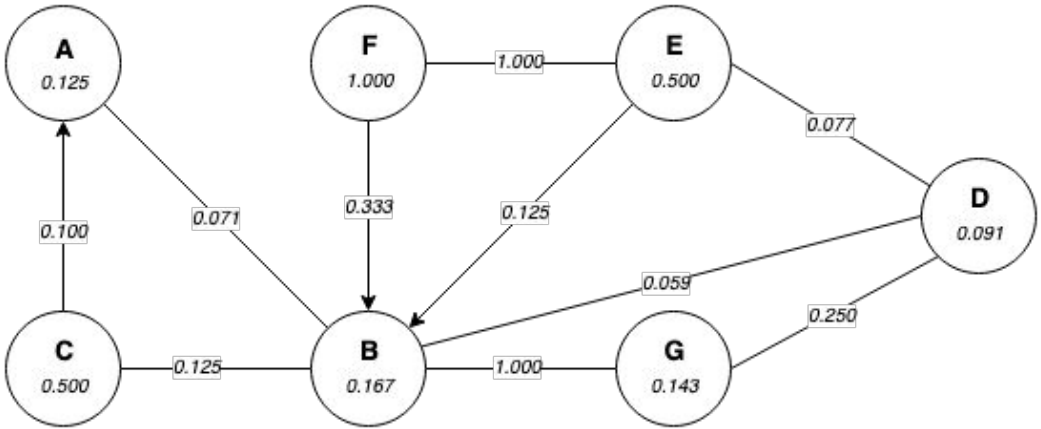
Painomatriisi

- Käytetään painotetun verkon luomiseen.
- Kuvaa kaarien prioriteetteja.

$$M_G \approx \begin{array}{c} G \\ v_A \\ v_B \\ v_C \\ v_D \\ v_E \\ v_F \\ v_G \end{array} \begin{array}{c} \begin{array}{ccccccc} v_A & v_B & v_C & v_D & v_E & v_F & v_G \end{array} \\ \begin{pmatrix} \infty & 0.071 & 0.100 & 0.053 & 0.100 & 0.200 & 0.067 \\ 0.071 & \infty & 0.125 & 0.059 & 0.125 & 0.333 & 1.000 \\ 0.100 & 0.125 & \infty & 0.077 & 0.250 & 1.000 & 0.111 \\ 0.053 & 0.059 & 0.077 & \infty & 0.077 & 0.125 & 0.250 \\ 0.100 & 0.125 & 0.250 & 0.077 & \infty & 1.000 & 0.111 \\ 0.200 & 0.333 & 1.000 & 0.125 & 1.000 & \infty & 0.250 \\ 0.067 & 1.000 & 0.111 & 0.250 & 0.111 & 0.250 & \infty \end{pmatrix} \end{array}$$

$$\beta(e_{AB}) = (p(v_A) + p(v_B))^{-1} = (8+6)^{-1} = \frac{1}{14} \approx 0.071$$

Painotettu verkko ennen leikkauksia



Leikkauksien tekeminen

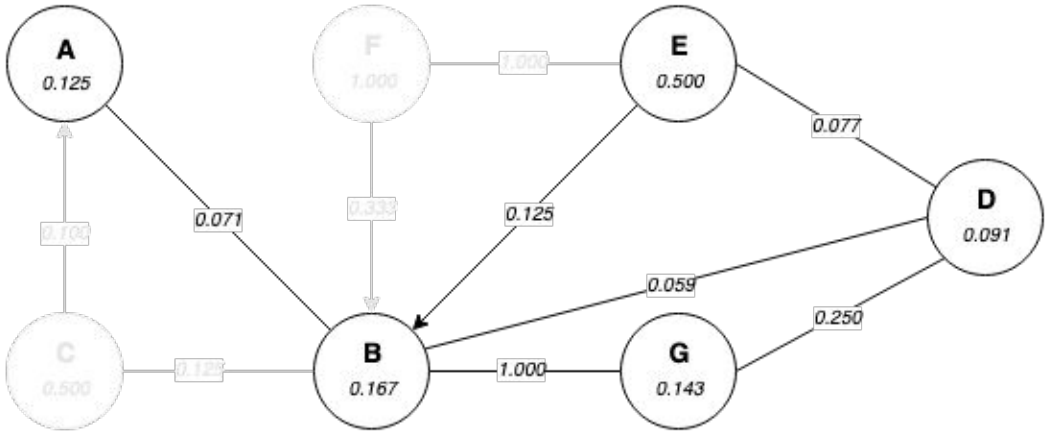
- Valitaan kattavuus, joka toimii rajana leikkauksien lopettamiseen. $0 \leq c \leq 100$,

Toistetaan n-kertaa:

- Poistetaan verkosta löytyvä eristetty solmu. $d_G(v) = 0$
- Poistetaan verkosta löytyvä sillattu solmu. $d_G(v) = 1 \wedge \alpha(v) > \frac{1}{|V(G)|} \cdot \sum_{v \in V(G)} \alpha(v)$
- Poistetaan verkosta sopiva alhaisimman prioriteetin solmu.

$$d_G(v) < max\{d_G(x) | x \in V(G)\} \wedge \alpha(v) > \frac{1}{|V(G)|} \cdot \sum_{v \in V(G)} \alpha(v)$$

Painotettu verkko leikkauksien jälkeen



Verkon ja testitapauksien yhteys

- Solmut ovat näkymiä.
- Näkymiä varten on tarkoitus muodostaa testikokoelmia.
- Yksittäisen solmun prioriteetti vastaa testikokoelman prioriteettia.
- Testitapaukset tekstikokoelman sisällä ovat keskenään samaa prioriteettia.

Evaluointi

Evaluointi 1/2: Testausjärjestelmä

+ Docker säiliönnin avulla tuki myös manuaaliselle testitapauksien ajamiselle
+ Docker säiliöinnin takia toteutus ei ole sidottu CI-palvelimeen
+ Xvfb virtualisoinnin avulla voidaan uusia verkkoselaimia lisätä helposti

- Xvfb virtuallisoinnin avulla ei voida testata Windows ympäristöön saatavia GUI- ohjelmia
- Robot framework takaa helpon luettavuuden kenelle tahansa, mutta ohjelmistokehittäjälle rajatun tuntuinen

Evaluointi 2/2: Priorisointimenetelmä

+ Priorisointimenetelmän toistettavuus
+ Menetelmän avulla on mahdollista priorisoida näkymiä ja siirtymiä
+ Dijkstran algoritmin avulla voidaan selvittää verkosta prioriteetiltaan korkein polku, eli näkymät joiden testikokoelmat tulisi suorittaa ensimmäisenä.
+ Painotettu verkko voidaan piirtää, joka kasvattaa ymmärrystä järjestelmästä

- Testikattavuuden päättäminen näkymä- ja siirtymäperusteisesti voi olla haastavaa
- Menetelmän käyttö soveltuu testikokoelmien, ei testitapauksien, priorisointiin
- Menetelmän käyttö ei ole järkevää jos käyttöliittymä on yksinkertainen

Kiitos!

Kysymyksiä?