Testaussuunnitelma

PULSU

Syksy 2008
Ohjelmistotuotantoprojekti

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Kurssi

581260 Ohjelmistotuotantoprojekti (9 op)

Projektiryhmä

Heikki Manninen
Noora Joensuu
Sami Vuorivirta
Joel Kaasinen
Erno Liukkonen

Asiakas

Peter von Etter
Roman Yangarber

Ohjaaja

Peter von Etter

Kotisivu

http://www.cs.helsinki.fi/group/pulsu/

Versiohistoria

Versio	Päivämäärä	Kuvaus
0.1	05.10.2008	Ensimmäinen versio
0.2	03.11.2008	Johdantoa muutettu
0.3	08.11.2008	Lisätty järjestelmätestausosiota
1.0	11.12.2008	Kohta "6 Testit" lisätty

Sisällysluettelo

1. Johdanto

Tämä on ohjelmistotuotantoprojektin Pulsu testaussuunnitelma. Suunnitelman tarkoituksena on varmistaa mahdollisimman toimivan ja testatun ohjelman tekeminen. Testaus on koko ryhmän vastuulla. Testausvastaava on vastuussa siitä että testaus tehdään testaussuunnitelmassa määritellyllä tavalla. Testausta tulee suorittaa toteutuksen rinnalla.

Projektin tarkoituksena on kehittää PULS-järjestelmälle helppokäyttöisempi ja ECDC:n vaatimuksiin soveltuva web-käyttöliittymä. PULS on luonnollisen kielen prosessointijärjestelmä, joka on kehitetty Helsingin yliopiston tietojenkäsittelylaitoksella. Se erottelee faktoja jäsentämättömästä tekstistä ja tallettaa ne tietokantaan. Järjestelmää voidaan käyttää eri skenaarioihin. Tällä hetkellä sitä käytetään pääasiassa tauti- ja liiketalousskenaarioihin. PULSU keskittyy lähinnä tautiskenaarioon, mutta pyrkii tekemään uudesta käyttöliittymästä sellaisen, että muiden skenaarioiden lisääminen on jälkikäteen helppoa.

Ohjelma toteutetaan SBCL-lisp ohjelmointikielellä, jonka interaktiivisuuden vuoksi testaaminen on nopeaa. Lausekattavuudelta vaaditaan vähintään 80%. Testejä on kolmen kaltaisia. Yksikkötestauksessa testataan järjestelmän pienimmät osat. Integrointitestaus on rajapintatestausta. Järjestelmätestauksessa testataan järjestelmän toimivuutta kokonaisuutena.

Jos jossain testausvaiheessa järjestelmästä löytyy virheitä, ne pyritään korjaamaan, jos se on aikataulun puitteissa mahdollista. Jos virheitä ei korjata, ne dokumentoidaan.

Testauksessa käytetään apuna Practical Common Lisp kirjasta löytyvää testaus kehystä. Kaikki testit käydään läpi ajamalla test-all funktio, joka kertoo menivätkö testit läpi.

Lausekattavuus tarkistetaan SBCL:ssä olevalla sb-cover moduulilla.

2. Sanasto

Lausekattavuus - kertoo, kuinka isossa osassa testattavan yksikön lauseita on käyty, kun testitapaukset on suoritettu.
Siirtymä - yhteys kahden lauseen välillä.
Haaraumakattavuus - käydään läpi ohjelman kaikki haarat. Esim if lauseiden aiheuttamat haarat.
osa-arvoalue - yksikön toiminnallisuus samanlaista samalla osa-arvoalueella.
Tynkä - testauskomponentti, joka toteuttaa pienimmän mahdollisen testauksessa vaaditun toiminnallisuuden.
Arvoalueanalyysi - kaikilla testin syötteillä on arvoalue, johon syöte kuuluu.

3 Yksikkötestaus

3.1 Lähestymistapa

Suoritetaan koodauksen yhteydessä. Koodin tekijä suorittaa testit. Kaikki ohjelmarivit testataan. Testitapaukset johdetaan ohjelmakoodinrakenteesta. esim If-lauseiden molemmat haarat tulee testata.

Testauksessa valitaan testiarvoja, jotka ovat tyypillisiä sallittuja syötteitä, osa-arvoalueiden reuna-arvoja sekä null-arvo. Myös käyttäytyminen virheellisillä syötteillä testataan.

Testauksessa suoritetaan toiminnallisuutta ja rakennetta testaavia testejä. Toiminnallisuutta testaavilla testeillä varmistetaan, että testattava yksikkö tekee sen, minkä lupaa. Rakennetta testaavilla testeillä varmistetaan, että testauksessa käydään kaikki ohjelmakoodi läpi, kuten osat joihin päästään vain erikoistapauksissa (esim. tietokantavirhe) ja kuollut koodi (koodi, johon ei päästä lainkaan).

Testit kirjoitetaan kyseisen kooditiedoston loppuun.

3.2 Testattavat kohdat

Testataan lispin pienimmät yksiköt (funktiot) järkevillä testiarvoilla.

Silmukat eivät saa jäädä ikuiseen toistoon normaaleilla syötteillä.

3.3 Hyväksymiskriteerit

Yksikkö katsotaan testatuksi, kun se on testattu tässä dokumentissa määritellyllä tavalla, eikä virheitä enää löydy.

4 Integrointitestaus

4.1 Lähestymistapa

Suoritetaan kun osia liitetään yhteen. Integrointitestauksessa testataan integroitujen yksiköiden rajapintoja. Integrointitestauksessa noudatetaan bottom-up strategiaa, jossa liitetään yksikkötestattuja osia toisiinsa, jolloin saadaan isompia yksiköitä.

Mahdollisia virheitä voivat olla
- Kutsuja ymmärtää rajapinnan väärin.
- Kutsuttava palauttaa väärin tulkitun arvon.
- Rajapintaa käytetään ei-toivotulla tavalla.
- Kutsuja odottaa palvelulta sivuvaikutuksia, jotka eivät toteudu.
- Kutsuttava aiheuttaa sivuvaikutuksia, joita kutsuja ei odottanut.
- Kutsuja aiheuttaa poikkeustilanteen, johon ei varauduttu.
- Kutsuva ja kutsuttava ymmärtävät palvelun syötteiden arvoalueet eri tavoin.

Testit kirjoitetaan kyseisen kooditiedoston loppuun.

Javascriptiä testataan JsUnit testikehystä käyttäen.

4.2 Testattavat kohdat

Kaikki osajärjestelmien väliset rajapinnat testataan.

1. Selvitetään, mitä rajapintojen palveluja integroidut osat vaativat toisiltaan ja tarjoavat toisilleen: siis mistä kohdin osat liittyvät toisiinsa
2. Tehdään kullekin palvelulle arvoalueanalyysi ja valitaan sen perusteella testisyötteet
3. Käytetään rajapintaa annetuilla testisyötteillä kutsujan kautta.

Kun osien liittyminen toisiinsa rajapintojen tarjoamien palveluiden kautta on selvitetty, valitaan sopivat testisyötteet ja rajapinnan toimivuus testataan.

4.3 Hyväksymiskriteerit

Kahden yksikön integrointitestaus on valmis, kun kaikki yksiköiden välinen yhteistyö on testattu, kaikki kutsuvan mahdollisesti generoimat poikkeukselliset syötteet on testattu - eli mahdolliset poikkeustilanteet on suoritettu, kaikki kutsuttavan aiheuttamat sivuvaikutukset järjestelmään on testattu.

5 Järjestelmätestaus

5.1 Lähestymistapa

Järjestelmätestauksessa testataan valmista järjestelmää. Järjestelmä testataan Käyttöliittymiensä kautta. Ei-toiminnalliset vaatimukset testataan toiminnallisten vaatimusten jälkeen. Järjestelmä testaukseen käytetään Selenium IDE ohjelmaa.

5.2 Testattavat kohdat

5.2.1 Testattavat käyttötapaukset

a.1 - Newest-näkymässä käyttäjä selaa unified-ryhmiä, jotka on listattu aikajärjestyksessä uusimmasta lähtien. (Requirements analysis 4.1.2)

a.2 - Newest-näkymässä käyttäjä siirtyy toisille sivuille löytääkseen haluamansa ryhmän.

a.3 - Käyttäjä voi ottaa pois näkyvistä caset jotka on merkattu oikein ryhmitellyiksi.

a.4 - Käyttäjä löytää etsimänsä unified-ryhmän ja näkee sen uusimman casen. Halutessaan hän näkee kaikki ryhmän caset. Käyttäjä valitsee ryhmän jonkin casen ja siirtyy tähän liittyvään dokumenttiin. (Requirements analysis 4.1.2)

b.1 - Käyttäjä voi muuttaa case:a dokumentti näkymässä. (Requirements analysis 4.1.1)

b.2 - käyttäjä voi poistaa case:n dokumentti näkymässä. (Requirements analysis 4.1.1)

b.3 - käyttäjä voi tehdä uuden case:n dokumenttiin. (Requirements analysis 4.1.1)

b.4 - Käyttäjä korjaa/muuttaa case:a dokumentti näkymässä ja merkkaa sen oikein ryhmitellyksi. (Requirements analysis 4.1.1 ja 4.1.3)

b.5 - Käyttäjä vaihtaa väärin ryhmitellyn casen unified-ryhmää. Tämä tehdään dokumentti näkymässä (Requirements analysis 4.1.4)

5.2.2 Testattavat ei-toiminnalliset vaatimukset

c.1 - Järjestelmän tulee reagoida alle 5 sekunnissa 90 prosentissa tapauksista, kun käyttäjiä on noin 10 käyttämässä samaa järjestelmää. (Requirements analysis E4)

c.2 - Sivujen tulee toimia Internet Explorerissa. (Requirements analysis E7)

c.3 - Käyttöliittymän tulee olla paranneltavissa ja tulee olla laajennettavissa toisiin skenaarioihin (Requirements analysis E8)

5.3 Hyväksymiskriteerit

Järjestelmätestaus on riittävä, kun käyttötapaukset ja ei-toiminnalliset vaatimukset on suoritettu onnistuneesti.

6. Testit

Testien tulokset ja järjestelmätestauksen Selenium-testit löytyvät repositoryn trunk/selenium hakemistosta. Testit testaavat kohdan 5.2.1 käyttötapaukset. Testit ajetaan Selenium IDE selain plugin ohjelmalla. Ohjelmalla avataan (Open Test Suite...) tests.selenium tiedosto ja ajetaan se.

7. Testausaikataulu

Yksikkötestausta suoritetaan koodattaessa. Projektin ensimmäisessä vaiheessa integrointi- ja järjestelmätestausta suoritetaan viikoilla 41-45.

Toisessa vaiheessa integrointi- ja järjestelmätestaus suoritetaan viikoilla 48-50