Testaussuunnitelma

Kohahdus
Helsinki 20.10.2006

Ohjelmistotuotantoprojekti
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Kurssi

581260 Ohjelmistotuotantoprojekti (6 ov)

Projektiryhmä

Taro Morimoto, Projektipäällikkö
Tuomas Palmanto, Vaatimusmäärittelyvastaava
Mikko Kinnunen, Suunnitteluvastaava
Markus Kivilä, Koodivastaava
Jari Inkinen, Testausvastaava
Paula Kuosmanen, Dokumenttivastaava

Asiakas

Teemu Kerola

Johtoryhmä

Sanna Keskioja

Kotisivu

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

Versiohistoria

Versio	Päiväys	Tehdyt muutokset
0.1	11.10.2006	Ensimmäinen versio
0.2	15.10.2006	Lisätty puuttuvat käyttötapaustestit
0.3	20.10.2006	Korjattu virheitä

Sisältö

1. Johdanto

Kohahdus on järjestelmä automaattisesti tarkastettavien TTK-91-konekielen harjoitustehtävien luomiseen ja ratkaisemiseen. Järjestelmä on tarkoitettu käytettäväksi opetuksen tukena, opetettaessa Tietokoneen toiminta -kurssia. Tietojenkäsittelytieteen opettajat voivat tehdä järjestelmään uusia tehtäviä ja määritellä kuinka ne tarkastetaan automaattisesti. Tietokoneen toiminta -kurssin opiskelijat ja kurssin tehtävistä kiinnostuneet itseopiskelijat voivat ratkaista tehtäviä ja saada palautetta niiden onnistumisesta.

1.1 Tavoitteet

Tämän testausdokumentin tavoitteena on opastaa testausvaihe, siten että Kohahdus-järjestelmä toimii virheettömästi ja toteuttaa vaatimusdokumentin vaatimukset.

Kaikkia virhetilanteita ei välttämättä löydetä testauksen aikana, mutta pyritään siihen että ohjelma suoriutuu annetusta tehtävästä virheettömästi ja mahdollisimman nopeasti. Tässä testaussuunnitelmassa kuvataan, kuinka Kohahdus testataan, jotta päästäisiin yllä kuvattuihin tavoitteisiin. Testauksen suunnitteludokumentissa on käytetty mallina muiden ohjelmistotuotantoprojektien dokumentteja.

1.2 Dokumentin rakenne

Luvussa 2 sanasto.

Luvussa 3 kuvataan testaussrategia: yksikkötestaus, integrointitestaus, järjestelmätestaus.

Luvussa 4 kerrotaan testaukseen käytetettävistä välineistä.

Luvussa 5 kerrotaan, mitä kaikkea halutaan testata.

Luvussa 6 kerrotaan testausaikataulusta.

2. Sanasto

TTK91=Auvo Häkkisen kehittämä ohjelmointikieli, joka läheisesti muistuttaa symbolista konekieltä.

KOKSI=Auvo Häkkisen kirjoittama konekielisimulaattori, joka toteuttaa TTK-91-kielen.

Järjestelmä=Projektimme tuotos, Kohahdus

Ohjelma=Opiskelijan kirjoittama TTK91-ohjelma, eli vastaus johonkin tehtävään

eAssari=Tietokantapohjainen ympäristö ohjelmallisesti tarkastettavien harjoitus- ja koetehtävien suorittamiseen

Titokone=Koski-nimisen Ohjelmistotuotantoprojektiryhmän vuonna 2004 rakentama järjestelmä konekielisten ohjelmien kääntämiseen ja suorittamiseen.

Koski=Vuoden 2004 Ohjelmistotuotantoprojekti joka rakensi konekielen simulaattorin ja debug-ympäristön, eli Titokoneen

Koskelo=Vuoden 2004 Ohjelmistotuotantoprojekti, joka integroi Titokoneen ja eAssarikehyksen yhteen. Ratkaisusta ei tullut kuitenkaan käyttökelpoista, eikä sitä ole otettu käyttöön.

Kriteeri=Sääntö jonkamukaan tehtävän oikeellisuus tarkistetaan. Kriteereitä voi ollamonta yhdelle tehtävälle.

Aihepiiri=Tehtävälle täytyy määritellä aihepiiri, johon tehtävä kuuluu.

JUnit=Testaustyökalu javakielelle

HttpUnit=Testaustyökalu webbisivuille (kuten servletit). Perustuu Junittiin.

JSPUnit=Työkalu erityisesti JSP-sivujen testaamiseen. Perustuu junittiin.

JSP=JavaServer Pages. Dynaamisien sivujen tekoon kehitetty ohjelmointikieli.

TitoTrainer=Kohahduksen tuotoksen nimi

3. Testausstrategia

Ohjelman testauksessa käytetään testauksen V-mallia. Tässä mallissa ideana on aloittaa yksikkötestauksella pienistä ohjelman osista, kuten metodeista ja luokista. Tämän jälkeen siirrytään testaamaan komponenttien yhteistoimintaa pienissä osissa, jotta saadaan selville missä virhe tapahtuu. Kolmannessa vaiheessa testataan osajärjestelmiä kokonaisuuksina, ja lopulta testataan koko järjestelmän toimivuutta.

Yksikkö- ja integrointitestaus pyritään tekemään testiluokilla mahdollisimman vähällä työllä. Järjestelmätestauksessa testataan ohjelmaa suoraan käyttöliittymien kautta, testaten kaikki palvelut, käyttötapaukset ja toiminnot. Projektin pääpainon ollessa käyttöliittymän selkeydessä, voidaan käyttää tämän asian testaukseen Tietokoneen toiminta- kurssin oppilaita. Kun ohjelmaa testaa joku muu kuin projektin jäsen, on paikalla kirjaamassa testaajan kommentteja joku projektilainen.

Koska Kohahdus-projektin järjestelmä joudutaan siirtämään projektin lopuksi asiakkaan toimesta varsinaiseen asiakasympäristöön, on tätä varten tehtävä asennusohjeet. Tämän testaus suoritetaan nykyisessä järjestelmässä poistamalla tietokannan taulut ja ohjelmisto, jonka jälkeen luodaan taulut sekä asennetaan ohjelmisto asiakasympäristöön.

3.1 Yksikkötestaus

Yksikkötestauksessa testataan jokainen luokka ja jsp-sivu. Painopiste on toteutuksen testaamisessa. Luokat testaavat luokan tekijät. Testit kirjoitetaan koodin jälkeen, koska testien kirjoitus sekä testien toteuttavan koodin tekeminen on työlästä eikä aikataulu anna sijaa tälle ratkaisulle.

Yksikkötestaus pyritään suorittamaan lausekattavasti, mikä tarkoittaa, että testattavasta kohteesta käydään muodostetusta suunnatusta verkosta käydään kaikki solmut läpi. Esimerkiksi metodista suunnattu verkko muodostetaan seuraavalla tavalla:

- Metodiin tulo on lähtösolmu ja siitä poistuminen maalisolmu. Molempia on tietenkin vain yksi.
- Kutakin lausetta kuvaa solmu ja siirtymistä lauseesta toiseen kuvaa verkon särmä.
- Ehtolauseen tapauksessa siitä lähtee kaksi särmää.
- Case-lauseessa niin monta särmää kuin case-tapauksia, sekä yksi särmä, joihin mikään vaihtoehto ei sovi.
- Jos lause on return, lähtee siitä särmä maalisolmuun.

Luokasta testataan metodit, paikalliset tietorakenteet ja rajapintojen toteutus. Metodeista testataan koodi, silmukat ja sisäiset tietorakenteet. Lisäksi suoritetaan virhetilanteiden testaus, kuten vaikka yritys hakea tietokannasta henkilön tiedot, jota siellä ei ole. Mahdolliset virhetilanteet luodaan ja tarkistetaan kuinka metodi selviää niistä. Rajapintojen toteutusta testattaessa tarkistetaan, että olion metodit ja attribuutit toimivat yhdessä, jotta ne voivat tuottaa luokan rajapinnan palvelut.

Lausekattavuus voidaan laskea seuraavalla kaavalla:

LK=TL/AL, missä

LK on lausekattavuus,

TL on testeissä käytetyn yksikön lauseiden lukumäärä

AL on kaikkien yksikön lauseiden lukumäärä.

Toisin sanoen lausekattavuus kertoo, kuinka suuressa osassa testattavan yksikön lauseita on käyty, kun testit on suoritettu. Yleisesti tulisi pyrkiä mahdollisimman lähelle 100%:ia, mutta Kohahdus voinee tyytyä 70%:in. Hyväksymiskriteerinä myös että kaikki yksikön toiminnot ja mahdolliset tilat on testattu. Samoin kaikki poikkeustilanteet. Testaamattomia ohjelman osia ei tule liittää testaukseen vaan tulee luoda tynkiä (tai testata ensin ne osat joiden suorittamiseen ei tarvita muita osia).

Yksikkötestaus suoritetaan luokan ohjelmoijan laitteistolla. Laitteistoissa ei ole paljoa eroja, mutta ainakin testausvastaavan koneella voidaan testata, miten ohjelma suoriutuu huonolla prosessoriteholla.

3.2 Integrointitestaus

Integrointitestaus tullaan tekemään ns. Bottom-up -strategialla. Tällöin ei rakenneta runkoa kuten Top-down –strategiassa, vaan yksikkötestattuja osia integroidaan toisiinsa yksi kerrallaan ja näin saadaan hiukan suurempia yksiköitä. Jatketaan integroimista, kunnes lopulta saadaan valmis tuote.

Testauksessa testataan Kohahdus-komponenttien toimintaa keskenään, ja lisäksi testataan niiden toimintaa Eassarin ja Titokoneen kanssa. Toisaalta koska suurin osa Eassarin komponenteista joudutaan toteuttamaan itse, keskittyy Eassarin testaus lähinnä tietokantaan. Titokonetta ei käytetä vielä ensimmäisessä iteraatiossa.

Integrointitestauksessa tarkoitus kuitenkin testata nimenomaan intergroitujen yksiköiden palveluiden yhteistyötä, eli rajapintoja. Testausprosessi menee seuraavasti:

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

Integrointitestauksessa ei pitäisi tulla ilmi muuta kuin rajapintaongelmia, sillä kukin yksikkö on jo testattu erikseen ja siten varmistettu että ne toimivat oikein. Vaikka kaikki yksiköt toimisivat oikein voi tulla ongelmia niiden yhteistyössä.

Integrointitestauksen ongelmia on esimerkiksi se, että kutsuja ymmärtää rajapinnan väärin, tai kutsuttava palauttaa väärin tulkitun arvon. Rajapintaa saatetaan myös käyttää väärällä tavalla. Kutsuja voi odottaa palvelulta sivuvaikutuksia, jotka eivät toteudu, tai kutsuttava aiheuttaa sivuvaikutuksia, joita kutsuja ei odottanut. Kutsuja voi myös aiheuttaa poikkeustilanteet, johon ei oltu varauduttu. Kutsuja ja kutsuttava voivat myös ymmärtää palvelun syötteiden arvoalueet eri tavoin.

Integrointitestauksessa voidaan vaikkapa testata parametrien arvoalueiden äärirajoilla olevia testiarvoja tai antaa osoitinparametreille voidaan antaa null-osoitin. Viestinvälitysrajapinnoille voidaan tehdä rasitustestaus. Proseduraaliselle rajapinnalle voidaan tehdä poikkeuksellisia kutsumisjärjestyksiä, kuten vaikka tiedoston luku ennen sen avaamista.

Kahden yksikön integrointitestaus on valmis, kun kaikki yksiköiden välinen yhteistyö on testattu, kaikki poikkeukset on testattu, ja kaikki mahdolliset kutsuttavan aiheuttamat sivuvaikutukset järjestelmään on testattu. Integrointitestaus on valmis, kun kaikki yksiköt

integroitu yhteen.

3.3 Järjestelmätestaus

Järjestelmätestaus (System testing) tehdään integrointitestauksen jälkeen. Järjestelmää testataan kokonaisuuten, johon kuuluvat ohjelmiston lisäksi laitteisto ja järjestelmän kanssa yhteistyössä toimivat ulkoiset ohjelmat.

Kohahdusta testataan suoraan käyttöliittymän kautta. Luodaan tunnus, luodaan tehtävä, ratkaistaan tehtävä ym. Toteutustapa eli oliopohjaisuus ei ole enää näkyvillä.

Järjestelmän testauksessa pyritään kustakin vaatimuksesta kirjaamaan täyttyykö vaatimus Kohahduksessa. Jos vaatimus ei täyty, täytyy kirjata miten vaatimuksen täyttymättömyys käy ilmi. Kirjataan myös mikäli jotain vaatimusta ei voida havaita tai testata.

3.4 Muu testaus

Käyttöliittymää on tarkoitus testata projektin ulkopuolisilla henkilöillä, jotta saadaan selville kuinka selkeä käytettävyys ohjelmalla on. Mikäli selkeitä epäselvyyksiä ilmenee, on syytä muokata käyttöliittymää selkeämpään muotoon.

Lopullisen testauksen suorittaa asiakas. Erimielisyyksistä pyritään keskustelemaan asiakkaan ja ryhmän kesken, mutta pienet toiminnaliset virheet pyritään lähes varmasti korjaamaan. Projektin ollessa kahdessa iteraatiossa voidaan ottaa asiakkaan kommentit ensimmäisessä demossa, ja korjata mahdollisesti puutteet toisessa iteraatiossa. Lopullisen testauksen voidaan todeta olevan valmis kun kaikki virheet on joko korjattu, tai niiden korjaamatta jättäminen on sovittu asiakkaan kanssa.

4. Testaukseen käytettävät välineet

Testaukseen voidaan käyttää JUnit, HttpUnit ja JSPUnit - testaustyökaluja. Jälkimmäiset perustuvat JUnittiin, ja ne on tehty testaamaan webbisivuja.

4.1 Testauskirjastojen lisääminen Eclipseen

Luodussa projektissa mennään **properties** – sivulle. Valitaan sieltä **Java Build Path** ja libraries – välilehti. Painetaan **Add External JARs**:ia ja haetaan tarvittavat kirjastot (.jar) tiedostoista. Lopuksi painetaan **ok**, jonka jälkeen testejä voidaan alkaa suorittamaan.

4.2 JUnit-testien teko Eclipsessä

Valitaan projektista testattava tiedosto, oikealla napilla **new**, **JUnit testcase**. Eclipse antaa testiluokalle valmiin nimen, ja voidaan valita tekeekö Eclipse testille valmiiksi tyngät metodeista. Testien tekoon löytyy apua osoitteessa www.junit.org.

4.3 Junit-testin rakenne

Alkuun täytyy importoida testikirjasto(t), kuten *import junit.framework.*;* JUnit-testit laajentavat yläluokkaa TestCase, joten ne määritellään muodossa:

public luokannimi extends TestCase {...}

Luokan nimelle syytä antaa jokin Test-päätteinen nimi, kuten vaikka DBHanderTest, jotta ne erottaa normaalista luokista. Luokassa käytettävät yhteiset muuttujat määritellään kuten muissakin luokissa metodien ulkopuolella. Yhteisiin muuttujiin voi vaikkapa sijoittaa testattavan luokan oliot, tosin välttämättä muutujia ei tarvitse olla yhtään.

setUp()- metodissa määritellään ennen kutakin testimetodia suoritettavat alustustoimet. setUp ajetaan automaattisesti ennen kutakin testimetodia. Jos testit esimerkiksi käsittelevät listarannetta ja muokkavat sitä, mutta halutaan, että testien alussa on kaikilla sama lähtötilanne, kannattaa alustus suorittaa setUp-metodissa. Esimerkki setUp metdista:

```
protected protected void setUp() {
        f12CHF= new Money(12, "CHF");
        f14CHF= new Money(14, "CHF");
}
```

tearDown()-metodissa määritetään kunkin testimetodin jälkeen suoritettavat toimenpiteet. Tässä metodissa vapautetaan pysyvät resurssit, jotka setUp-metodissa annettiin. Esimerkiksi suljetaan avatut verkkoyhteydet. tearDown()-metodia ei yleensä tarvita.

Testimetodit määritellään nimeämällä julkinen testimetodi muodossa *testTestinNimi*. Yhdessä testimetodissa kannattaa testata vain yhtä toimintoa eli käytännössä metodia. Testi kirjoitetaan käyttäen JUnitin assert-lauseita ja kirjoittamalla tarvittava määrä muuta koodia. Testien kirjoittajan on syytä varmistaa, että testi kattaa metodin testauksen tarpeeksi hyvin. Esimerkki testimetodista (käyttää ylläolevaa setUp()-metodia):

```
public void testSimpleAdd() {
        Money expected= new Money(26, "CHF");
        Money result= f12CHF.add(f14CHF);
        assertTrue(expected.equals(result));
}
```

assert-lauseita on erilaisia, kuten *assertTrue*, *assertEquals*, *assertNotNull*. Lisää tietoa asserteista löytyy Assertin API:ssa.
http://junit.sourceforge.net/javadoc/junit/framework/Assert.html.

simerkit on otettu seuraavasta artikkelista:
http://junit.sourceforge.net/doc/testinfected/testing.htm.

4.4 Junit-testien ajo eclipsessä

Testimetodeja ei tarvitse kutsua missään luokassa, vaan ajetaan luokka suoraan testinä. Valitaan **Run as**-valikosta JUnit. Testin jälkeen nähdään tiedot ajetusta testistä. *Runs* kertoo ajettujen testimetodien lukumäärän. *Errors* ilmoittaa monessako tapauksessa ohjelman suoritus on keskeytynyt johonkin virheeseen. *Failures* kertoo kuinka monta *assert*-lausetta on mennyt pieleen; tosin ohjelman osa on tällöin suoritettu loppuun. Klikkaamalla virheen antanutta testimetodia nähdään *Failure Trace* – ikkunassa tarkemmin mitä on mennyt pieleen.

4.5 HttpUnit-testit

HttpUnit-testit toimii kuten JUnit-testit eli ne laajentaa *TestCase*:n, niissä on setUp() ja tearDown() ja niissä käytetään myös assert-lauseita. HttpUnittia voidaan käyttää HttpServlet-luokkien testaamiseen. Tosin Kohahdus ei tule käyttämään niitä paljoa. Hyvä opastus aiheesta löytyy seuraavasta linkistä:
http://httpunit.sourceforge.net/doc/tutorial/task1.html.

HttpUnit on kirjastot jotka täytyy importoida testiluokan alkuun. Testeissä luodaan olio ServletRunner, jolla luodaan client, ServletUnitClient. Tälle voidaan suorittaa operaatioita. Seuraavassa esimerkkejä:

```
ServletRunner sr = new ServletRunner( "web.xml" );
ServletUnitClient client = sr.newClient();
client.setAuthorization( "aUser", "pool-admin" );
WebResponse response = client.getResponse( "http://localhost/PoolEditor" );
WebForm form = response.getFormWithID( "pool" );
```

Asserttia käytetään siis jälleen testauksen oikeellisuuden tarkistamiseen, esimerkiksi vaikka:

```
assertEquals( "Away team 0", "", form.getParameterValue( "away0" ) );
```

HttpUnittia voidaan käyttää myös JSP-sivujen testaukseen, mutta vaikeasti saatavien ohjeiden vuoksi on parempi käyttää testityökalua JSPUnit, joka luotu juuri niitä sivuja varten.

4.6 JSPUnit-testit

JSPUnit on HttpUnitin tavoin lisäys JUnittiin. HttpUnit on monipuolisempi ohjelma, mutta samalla vaikeampi käyttää, joten JSPUnit soveltuu projektille hyvin. Luokat löytää sivulta: http://www.dallaway.com/jsptest/.

Testiluokat laajentavat taas TestCase:a. Alkuun on hyvä määritellä luokalle String-tyyppinen muuttuja, jossa on osoite testattavan tai testattavien JSP-sivujen sijainnille. Testimetodeissa luodaan Session-olioita, johon voidaan lisätä Cookie-olioita, tai käyttää luokkaa Response, jolla voi hakea Sessionista tietoja. Assert-lauseilla testataan jälleen kerran sivujen oikeellisuus.

Hyvä esimerkkitesti löytyy linkistä http://www.dallaway.com/jsptest/JSPTestTest.txt.

5 Suoritettavat testitapaukset

5.1 Koodin testaus

Ensimmäisessä iteraatiossa tullaan toteuttamaan tai muokkaamaan luokkia User.java, Task.java, Course.java, DBHandler.java ja Criterion.java. Testauksessa näistä luokista täytyy luoda Junit-testiluokka, ja kirjoittaa kattavat testimetodit, jotka käyvät läpi koko koodin. Samoin JSP-sivujen koodi testataan, mahdollisesti käyttäen JSPUnittia.

5.2 Käyttötapaukset

Kohahduksen vaatimukset koostuvat paljolti käyttötapauksista, joten testaus on hyvä tehdä siltä pohjalta.

HYV=hyväksytytään kyseinen syöte.

5.2.1 Opettajan käyttötapaukset

Kt1. järjestelmään kirjautuminen
Sivu: Aloitussivu (login.jsp)
Testattavat käyttöliittymäkomponentit:
- tekstikentät Username, Password
- dropdown Language, ei tulla toteuttamaan ensimmäisessä iteraatiossa
- painike Sign in

Tekstikentät: Username ja Password

Syöte	***Toivottu tulos***
Oletusarvo: molemmat tyhjiä	*'User name or password incorrect'*
Toinen tyhjä	*'User name or password incorrect'*
Väärä syöte: Teme xxyyzz	*"User name or password incorrect'*
Väärä salasana:Teemu xxyzzz	*"User name or password incorrect'*
Oikea syöte: Teemu xxyyzz	*HYV*

Painike: Sign in

Syöte	***Toivottu tulos***
Painallus	*Järjestelmä siirtää käyttäjän tehtävien selaussivulle*

Kt2. Tehtävien selaaminen
Sivu: Kurssi ja tehtälistaus (teacherTaskList.jsp)
Testattavat käyttöliittymäkomponentit:
- Tekstikenttä Create new course
- painikkeet new course, remove course, new task, edit task, edit as new, remove task(**kt6**),
  log off(**kt8**)

Tekstikenttä: New course

Syöte	*Toivottu tulos*
Välilyönti	*'Virheellinen kurssin nimi'*
Html-koodia "T ito 2006 Linkki"	*'Virheellinen kurssin nimi'*
Oikea "T itou syksy 2006"	*HYV*

Painike: New course

Syöte	*Toivottu tulos*
Painallus	*Luo kurssin tekstikentässä määritellyssä nimessä, näkyy heti sivulla*

Painike: Remove Course

Syöte	*Toivottu tulos*
painallus	*Kysyy varmistuksen poistetaanko kurssi. Mikäli valitaan kyllä kyseinen kurssi poistuu järjestelmästä.*

Painike: New task

Syöte	*Toivottu tulos*
painallus	*Siirtää käyttäjän tehtävänluontisivulle*

Painike: Remova Task

Syöte	*Toivottu tulos*
painallus	*Kysyy varmistuksen poistetaanko tehtävä. Mikäli valitaan kyllä kyseinen kurssi poistuu järjestelmästä*

Painike: Edit task

Syöte	*Toivottu tulos*
painallus	*Siirtää käyttäjän tehtävänluontisivulle, jonka kenttien oletusarvona kyseisen tehtävän arvot*

Painike: Edit as new

Syöte	*Toivottu tulos*
painallus	*Siirtää käyttäjän tehtävänluontisivulle, jonka*

Syöte	*Toivottu tulos*
	kenttien oletusarvona kyseisen tehtävän arvot

Painike: Log off

Syöte	*Toivottu tulos*
Painallus	*Käyttäjä kirjautuu ulos järjestelmästä, siirtyy kirjautumissivulle*

Kt3. Tehtävän lisääminen
Sivu: Tehtäväluontisivu (composer.jsp)
Testattavat käyttöliittymäkomponentit:
- painikkeet Add variable, Save, ja kielletyille/vaadituille käskyille olevat napit
- Radiopainikkeet Fill-In/Programming, UseModel/Criteria
- tekstikentät Task name, Task Description, Inputs, Secret Inputs, Register values (7kpl), Register values for secret input (7kpl), Variable names, Variable values, Variable secret values, Outputs, Secret Outputs, Code size, Code size quality, data area size, data area size quality, stack max size, stack max size quality, executed commands, executed commands quality, memory references, memory references quality, max commands allowed, Palautekentät kaikille kriteereille, Final feedback success, Final feedback failed.
- Dropdownit, jossa vertailut muuttujien ja rekistereiden arvoille.

Painike: Add Variable

Syöte	*Toivottu tulos*
Painallus	*Lisää uuden muuttujan listaan, jolle kaikki kentät määritelty*

Painike: Save

Syöte	*Toivottu tulos*
Painallus	*Tallentaa tehtävän tietokantaan mikäli kaikki kentät kunnossa, siirtää käyttäjän takaisin tehtälistaussivulle, jossa näkyy lisätty tehtävä*

Painikeet: kielletyt/vaaditut käskyt

Syöte	*Toivottu tulos*
Valitaan yksi vaadituksi	*HYV*
Valitaan yksi vaadituksi, yksi kielletyksi	*HYV*
Kaikki kielletty/vaadittu	*HYV*
Yksi kielletty	*HYV*

Syöte	*Toivottu tulos*
Ei yksikään kielletty tai sallittu	*HYV*

Radiopainikkeet: Fill-In/Programming

Syöte	*Toivottu tulos*
Fill-In	*Kaksi ylimääräistä tekstikenttää alku- ja jälkikoodille*
Programming	*Ei ylimääräisiä kenttiä*
Fill-In -> Programming -> Fill-In, Fill-In kentissä tekstiä	*Teksti säilyy muutoksen mukana*

Radiopainikkeet: Use model/Criteria

Syöte	*Toivottu tulos*
Use model	*Tekstikenttä mallikoodille*
Criteria	*Mallikoodin tekstikenttä häviää*
Use model -> Criteria -> Use model, mallikoodissa tekstiä	*Mallikoodin koodi tulee takaisin näkyviin*

Tekstikenttä: Task name

Syöte	*Toivottu tulos*
Oletusarvo, tyhjä	*Virheellinen nimi*
Välilyönti	*Virheellinen nimi*
Html-tekstiä: harjoitus 1	*Virheellinen nimi*
Yksi merkki	*HYV*
Oikea syöte: Harjoitus 1	*HYV*

Tekstikenttä: Task description

Syöte	*Toivottu tulos*
Oletusarvo, tyhjä	*Virheellinen kuvaus*
Välilyönti	*Virheellinen kuvaus*
Html-tekstiä: Laske sitä...	*Virheellinen kuvaus*
Yksi merkki	*HYV*
Oikea syöte: Laske sitä...	*HYV*

Tekstikentät: Inputs ja Secret inputs

Syöte	***Toivottu tulos***
Oletusarvo, tyhjä	*HYV, ei syötteitä*
välilyönti	*HYV, ei syötteitä*
Yksi syöte: 3	*HYV*
Negatiivinen -1	*Virheellinen syöte*
Monta, mutta väärin, 4.4.3 3 5	*Virheellinen syöte*
Muu kuin numero: E	*Virheellinen syöte*
Välit eri lailla: 4,5, 4, 3, 2	*HYV*
Oikea: 3, 5, 7	*HYV*
Eri määrä syötteitä Inputs ja Secret Inputs	*Virheellinen syöte*

Tekstikentät: Register values and Register secret values

Syöte	***Toivottu tulos***
tyhjä	*HYV, ei luoda kriteeriä*
välilyönti	*HYV, ei luoda kriteeriä*
Muu kuin numero, -, r, &	*Virheellinen rekisterin arvo*
Oikea: 4	*HYV*
Numeron jälkeen jotain: 4TT	*Virheellinen rekisterin arvo*
Negatiivinen: -4	*Virheellinen rekisterin arvo*
Nolla: 0	*HYV*

Tekstikentät: Variable names

Syöte	***Toivottu tulos***
Oletusarvo: tyhjä	*HYV, ei luoda kriteeriä*
Numero: -4, 6	*Virheellinen muuttuja*
Pitkä nimi: Muuttis	*HYV*
Ääkkösiä: å	*Virheellinen muuttuja*
Muita merkkejä: e-w	*Virheellinen muuttuja*
Oikea: x	*HYV*
Numero perässä: X3	*HYV?*

Tekstikentät: Variable and secret variable values

Syöte	***Toivottu tulos***
Oletusarvo:tyhjä	*HYV, ei määritellä muuttujaa*
Muu kuin numero, -, r, &	*Virheellinen muuttujan arvo*

Syöte	***Toivottu tulos***
Oikea: 4	*HYV*
Numeron jälkeen jotain: 4TT	*Virheellinen muuttujan arvo*
Negatiivinen: -4	*Virheellinen muuttujan arvo*
Nolla: 0	*HYV*

Tekstikentät: Outputs and secret outputs

Syöte	***Toivottu tulos***
Oletusarvo, tyhjä	*HYV, ei tulosteita*
välilyönti	*HYV, ei tulosteita*
Yksi syöte: 3	*HYV*
Negatiivinen -1	*Virheellinen tuloste*
Monta, mutta väärin, 4.4.3 3 5	*Virheellinen tuloste*
Muu kuin numero: E	*Virheellinen tuloste*
Välit eri lailla: 4,5, 4, 3, 2	*HYV*
Oikea: 3, 5, 7	*HYV*
Eri määrä syötteitä Outputs ja Secret Outputs	*Virheellinen tuloste*

Tekstikentät: Laadulliset kriteerit: Code size, Code size quality, data area size, data area size quality, stack max size, stack max size quality, executed commands, executed commands quality, memory references, memory references quality

Syöte	***Toivottu tulos***
Oletusarvo:tyhjä	*HYV, ei määritellä arvoa*
Muu kuin numero, -, r, &	*Virheellinen kriteerin arvo*
Oikea: 43	*HYV*
Numeron jälkeen jotain: 4TT	*Virheellinen kriteerin arvo*
Negatiivinen: -4	*Virheellinen kriteerin arvo*
Nolla: 0	*HYV*

Tekstikenttä: Max commands allowed

Syöte	***Toivottu tulos***
Oletusarvo:200	*HYV*
Muu kuin numero, -, r, &	*Virheellinen arvo*
Oikea: 43	*HYV*
Numeron jälkeen jotain: 4TT	*Virheellinen arvo*

Syöte	*Toivottu tulos*
Negatiivinen: -4	*Virheellinen arvo*
Nolla: 0	*Virheellinen arvo*
10000 tai yli	*Varoitus, HYV*

Tekstikentät: Palautekentät kriteereille

Syöte	*Toivottu tulos*
Oletusarvo: oletuspalaute	*HYV*
Tyhjä	*HYV, antaa oletuspalautteen*
Yksi merkki	*HYV*
Palaute mutta esim rekisterille ei ole annettu arvoa	*HYV, ei tallenneta kriteeriä*
Välilyönti	*HYV, antaa oletuspalautteen*

Tekstikentät: Final feedback success/Final feedback failed

Syöte	*Toivottu tulos*
Oletusarvo:oletuspalaute	*HYV*
Tyhjä	*HYV, antaa oletuspalautteen*
Välilyönti	*HYV, antaa oletuspalautteen*
Yksi merkki	*HYV*

Dropdownit: Vertailut muuttujien ja rekisterien arvoille

Syöte	*Toivottu tulos*
Oletusarvo =	*HYV*
Muut, <, <=, >, >=, !=	*HYV*

Kt4. Tehtävän muokkaaminen
Sivu: Tehtäväluontisivu (composer.jsp)
Testattavat käyttöliittymäkomponentit:
- Samat mitä tehtävän luonnissa, ei kuitenkaan syytä testata niitä uudestaan. Täytyy kuitenkin katsoa että muutokset tulevat voimaan.

Kt5. Tehtävän tulostaminen
Sivu: ??
Testattavat käyttöliittymäkomponentit:

Kt7. Raporttien katsominen 2. iteraatiossa
Sivu: Tilastosivu (statistics.jsp)
Testattavat käyttöliittymäkomponentit:

5.2.1 Opiskelijan käyttötapaukset

Kt9. järjestelmään rekisteröityminen
Sivu: Rekisteröintisivu (signup.jsp)
Testattavat käyttöliittymäkomponentit:
- tekstikentät First name, Last name, Student number, Social security number, E-mail, User name, Password, Password again.
- painike Sign up

Tekstikenttä: First name

Syöte	***Toivottu tulos***
Oletusarvo: tyhjä	*'Etunimi ei voi olla tyhjä'*
Välilyönti tai useita välejä	*'Etunimi ei voi olla tyhjä'*
Yksi merkki '3'	*HYV*
Html-koodia <l>Taro</l>	*'Virheellisiä merkkejä etunimessä'*
Oikea syöte: Taro	*HYV*

Tekstikenttä: Last name

Syöte	***Toivottu tulos***
Oletusarvo: tyhjä	*'sukunimi ei voi olla tyhjä'*
Välilyönti tai useita välejä	*'sukunimi ei voi olla tyhjä'*
Yksi merkki '3'	*HYV*
Html-koodia <l>Morimoto</l>	*'Virheellisiä merkkejä sukunimessä'*
Oikea syöte: Morimoto	*HYV*

Tekstikenttä: Student number

Syöte	*Toivottu tulos*
Oletusarvo: tyhjä	*'Opiskelijanumero tai sotu puuttuu'*
Tyhjä mutta social security number hyväksytty	*HYV*
Muita kun numeroita: 129as½3u7	*'virheellinen opiskelijanumero'*
Liian lyhyt: 123	*'virheellinen opiskelijanumero'*
Liian pitkä:99999999999999999	*'virheellinen opiskelijanumero'*
Oikea: 047839222	*HYV*

Tekstikenttä: Social security number

Syöte	*Toivottu tulos*
Oletusarvo: tyhjä	*'Opiskelijanumero tai sotu puuttuu'*
Tyhjä mutta student number hyväksytty	*HYV*
Virheellinen muoto 343-gg-333	*'virheellinen sotu'*
Oikea muoto: 170287-123C	*'HYV'*

Tekstikenttä: E-mail

Syöte	*Toivottu tulos*
Oletusarvo: tyhjä	*'Sähköposti ei voi olla tyhjä'*
välilyönti	*'virheellinen sähköpostiosoite'*
Yksi merkki '9'	*'virheellinen sähköpostiosoite'*
html-koodia: taro.morimoto@cs.helsinki.fi	*'virheellinen sähköpostiosoite'*
Oikea syöte: taro.morimoto@cs.helsinki.fi	*HYV*
Virheellinen syöte: taro@	*'virheellinen sähköpostiosoite'*

Tekstikenttä: User name

Syöte	*Toivottu tulos*
Oletusarvo: tyhjä	*'käyttäjänimi ei voi olla tyhjä'*
Välilyönti tai useita välejä	*'käyttäjänimi ei voi olla tyhjä'*
Nimi joka jo kannassa	*'käyttäjänimi on jo varattu'*
Alta 3 merkkiä	*'Käyttäjänimi liian lyhyt'*
2 välilyöntiä, 1 merkki	*'Käyttäjänimi liian lyhyt'*
Yli 10 merkkiä	*'Käyttäjänimi liian pitkä'*
Välilyöntejä alussa tai lopussa, merkkejä alle 10	*'Virheellinen käyttäjänimi'*
Välilyönti nimen keskellä	*'Virheellinen käyttäjänimi'*

Syöte	*Toivottu tulos*
Oikea:ninja	*HYV*

Tekstikentät: Password ja Password again

Syöte	*Toivottu tulos*
Password tyhjä	*'virheellinen salasana'*
Password again tyhjä	*'salasanan varmistus puuttuu'*
Välilyönti salasanan keskellä	*'virheellinen salasana'*
Password ja password again eivät samat	*'virheellinen salasana'*
Password ja password again täsmäävät hyväksytyllä salasanalla	*HYV*
Alle 5 merkkiä	*'virheellinen salasana'*
Välilyöntejä alussa tai lopussa	*'virheellinen salasana'*
Välilyönti tai useita välilyöntejä	*'virheellinen salasana'*
Html-koodia ' salasana '	*'virheellinen salasana'*
Väärä syöte: @$}	*'virheellinen salasana'*
Oikea syöte: passun1	*HYV*

Painike: Sign up

Syöte	*Toivottu tulos*
Painallus	*Järjestelmä siirtää käyttäjän kirjautumissivulle*

Kt10. järjestelmään kirjautuminen
Sivu: Aloitussivu (login.jsp)
Testattavat käyttöliittymäkomponentit:
- tekstikentät Username, Password
- dropdown valikko Course
- painike Sign in, Sign up

Tekstikentät: Username ja Password

Syöte	*Toivottu tulos*
Oletusarvo: molemmat tyhjiä	*'User name or password incorrect'*
Toinen tyhjä	*'User name or password incorrect'*
Väärä syöte: Tarou xxyyzz	*"User name or password incorrect'*
Väärä salasana:Taro xxyzzz	*"User name or password incorrect'*
Oikea syöte: Taro xxyyzz	*HYV*

Dropdown: Course

Syöte	***Toivottu tulos***
Selaaminen	*Näyttää kaikki järjestelmän kurssit*

Painike: Sign in

Syöte	***Toivottu tulos***
Painallus	*Järjestelmä siirtää käyttäjän dropdownissa valitun kurssin tehtävien selaussivulle*

Painike: Sign up

Syöte	***Toivottu tulos***
Painallus	*Siirtää käyttäjän tunnuksenluontisivulle*

Kt11. Tehtävien katselu
Sivu: Tehtäväsivu (studentTaskList.jsp)
Testattavat käyttöliittymäkomponentit:
- painikkeet Log off (**kt13**), Do task

Painike: Log off

Syöte	***Toivottu tulos***
painallus	*Käyttäjä kirjautuu ulos järjestelmästä, siirtyy kirjautumissivulle*

Painike: Do task

Syöte	***Toivottu tulos***
painallus	*Käyttäjä siirtyy valitun tehtävän tehtävänratkaisusivulle*

Kt12. Tehtävien ratkaiseminen 2. iteraatiossa
Sivu: Tehtäväsivu (answer_task.jsp)
Testattavat käyttöliittymäkomponentit:

Kt14. Omien tietojen muokkaus
Sivu: Aloitussivu (edit_profile.jsp)
Testattavat käyttöliittymäkomponentit:
- tekstikentät First name, Last name, Student number, Social security number, E-mail, Old password, New password, New password again.
- Painike Save

Passwordeja lukuunottamatta testaus samalla periaatteella, kuten rekisteröitymisessä. Oletusarvoina vanhat tiedot.

Tekstikentät: Old Password

Syöte	***Toivottu tulos***
Oletus: tyhjä	*'vanha salasana väärä'*
Yksi merkki '9'	*HYV*
välilyönti	*'virheellinen salasana'*
Html-koodia ' salasana '	*'virheellinen salasana'*
Väärä syöte: @$}	*'virheellinen salasana'*
Oikea syöte: passun1, eli se joka oli jo aiemmin määritelty	*HYV*

Tekstikentät: New password ja New password again

Syöte	***Toivottu tulos***
Molemmat tyhjiä, ei muuteta	*HYV*
New password tyhjä, new password again	*'virheellinen salasana'*

Syöte	*Toivottu tulos*
ei tyhjä	
New password again tyhjä, New password ei tyhjä	*'salasanan varmistus puuttuu'*
New password ja New password again eivät samat	*'virheellinen salasana'*
New password ja New password again täsmäävät hyväksytyllä salasanalla	*HYV*
Yksi merkki '9'	*HYV*
välilyönti	*'virheellinen salasana'*
Html-koodia ' salasana '	*'virheellinen salasana'*
Väärä syöte: @$}	*'virheellinen salasana'*
Oikea syöte: passun2	*HYV*

Painike: Save

Syöte	*Toivottu tulos*
Painallus	*Tallentaa tiedot kantaan, Siirtää käyttäjän takaisin studentTaskList.jsp*

5.3 Testien raportointi

Testeistä kirjoitetaan raportti, joista täytyy käydä ilmi vähintään seuraavat asiat:

Testaaja: Taro

Päivämäärä: 18.10.2006

Mitä testattu: Luokka DBHandler metodeineen

Testin kuvaus: Testattiin JUnit-testiluokalla testDBHandler

Odotetut tulokset: Tietokantaoperaatiot onnistuu

Havaitut virheet: Ei ongelmia.

Tulokset: Testi suoritettu onnistuneesti.

Testausraporteista koostetaan Testausdokumentti.

6 Testausaikataulu

Yksikkötestauksessa ei tarvita aikataulua, koska se katsotaan osana koodausta ja suoritetaan sen yhteydessä. Integrointi – ja järjestelmätestaus pyritään aloittamaan viikolla 42. Aikaa ei jää paljon, mutta pyritään, että kaikki saadaan toimimaan viikon aikana.