Testausraportti

Koskelo

Helsinki 15.12.2004

Ohjelmistotuotantoprojekti

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Kurssi

581260 Ohjelmistotuotantoprojekti (6 ov)

Projektiryhmä

Tom Bertell
Johan Brunberg
Lauri Liuhto
Eeva Nevalainen
Harri Tuomikoski

Asiakas

Teemu Kerola

Johtoryhmä

Juha Taina
Turjo Tuohiniemi

Kotisivu

`http://www.cs.Helsinki.FI/group/koskelo/`

Versiohistoria

Versio	Päiväys	Tehdyt muutokset
0.1	22.11.2004	Ensimmäinen versio
0.2	26.11.2004	Lisätty rungot kaikenlaisille testitapauksille
1.0	15.12.2004	Lopullinen versio

Sisältö

1 Johdanto

Tämä on ohjelmistotuotantoprojekti Koskelon testausraportti. Projekti on laajentanut Assari-projektissa jo laajennettua eAssari tehtäväjärjestelmää tuottamalla uusia tehtävätyyppejä.

Testauksella pyrittiin erityisesti löytämään virheitä Koskelossa tuotetuissa ohjelmistokomponenteissa ja mahdollisesti selvittämään onko ratkaisemattomien virheiden syynä korjaamaton virhe ulkoisissa osajärjestelmissä.

2 Yksikkötestaus

Yksikkötestit suoritettiin hyödyntäen JUnit testausympäristön versiota 3.8.1. Yksikkötestauksessa pyrittiin mahdollisimman suureen kattavuuteen ja testiluokkien kattavuutta arvioitiinkin laitoksen tutkimusprojektina kehitetyn Rita-ohjelmiston avulla.

2.1 Composer-osajärjestelmä

2.1.1 TTK91TaskCriteria

Testien kirjoittajat: Eeva Nevalainen ja Harri Tuomikoski

Kohde: TTK91TaskCriteria.java

Kuvaus: Testit suoritettiin virheettömästi. Testit kattavat yli 90% ohjelmariveistä, kattamattomiksi jäivät throw-lausekkeet.

Kutakin konstruktoria testattiin monenlaisilla merkkijonoilla ja testattiin, että ne ottavat oikean osa syöteestä oikeaan muuttujaan.

2.2 Displayer-osajärjestelmä

2.2.1 StaticTTK91Displayer

Testien kirjoittaja: Tom Bertell

Kohde: StaticTTK91Displayer.java

Kuvaus: Testit suoritettiin virheettömästi.

Luokan toiminnallisuuden toteuttavaa getSetting-metodia testattiin monilla yleisillä testiparametreilla ja tutkittiin, että tuloksena syntynyt html-koodi oli syötteiden mukaista. Testaamisen apuna käytettiin TestCache-luokkaa, joka toimi testin tietokantana.

2.2.2 FillInTTK91Displayer

Testien kirjoittaja: Tom Bertell

Kohde: FillInTTK91Displayer.java

Kuvaus: Testit suoritettiin virheettömästi.

Luokka testattiin samoilla tavoin kuin luokka StaticTTK91Displayer, mutta lisäksi testattiin, että koodissa oleva aukon paikka on oikea. Aukon paikkoina testattiin: ennen koodia, koodin jälkeen ja keskellä koodia. Testaamisen apuna käytettiin TestCache-luokkaa, joka toimi testin tietokantana.

2.2.3 TTK91DisplayerUtils

Testien kirjoittaja: Eeva Nevalainen

Kohde: TTK91DisplayerUtils

Kuvaus: Testit suoritettiin virheettömästi. Testit kattoivat 100% luokan ohelmariveistä.

2.3 Analyser-osajärjestelmä

2.3.1 TTK91AnalyserUtils

Testien kirjoittaja: Tom Bertell

Kohde: getTTK91Taskoptions()

Kuvaus: Testit suoritettiin virheettömästi.

Metodia testattiin erilaisilla merkkijonoilla sekä null-arvoilla. Testaamisen apuna käytettiin TestCache-luokkaa, josta testaamisessa käytetyt parametrit haettiin.

2.3.2 TTK91FeedbackComposer

Testien kirjoittaja: Tom Bertell

Kohde: TTK91FeedbackComposer.java

Kuvaus: Testit suoritettiin virheettömästi.

Luokkaa oli työläs testata, koska se tuottaa kymmeniä rivejä html-koodia. Tämän vuoksi luokkaa testattiin vain osalla parametreista ja jätettiin kattavampi testaaminen integraatiotestaukseen, jolloin oikea tulos on helppo todeta selaimessa.

3 Integraatiotestaus

Järjestelmään ei kohdistunut juurikaan integraatiotestausta sekä alkuperäisten suunitelmien puutteellisudeen että projektin loppupuolen aikataulun kireyden vuoksi.

3.1 Composer

Composerista testattiin, että SyntaxChecker luo oikeanlaisen TaskOptions-olion, joka voidaan myös tallentaa merkkijonoina tietokantaan. Käytännössä testaus suoritettiin luomalla esimerkkitehtävä ja käsin lukemalla tietokannan sisältöä. Tehtävien tietokantaan tallentumista testattiin myös analyserin yhteydessä, joka sisältää komponentin, joka palauttaa tehtävänannon tietokannasta.

3.2 Displayer

Displayeriin ei käytännössä kohdistunut ollenkaan integraatiotestausta. Varsinainen testaus tehtiin testaamalla yksittäisten metodien paluuarvoja JUnitilla, kuten jo kuvattu, sekä järjestelmätestausvaiheessa erilaisilla oikeilla tehtävänannoilla.

3.3 Analyser

Analyserin integraatiotestit käsittivät hyvin pintapuoleisesti testatun yhteistyön eri olioiden kesken. Testaus jäi pintapuoleiseksi ohjelmistotuotantoprojekteissa tavanomaisen ajanpuutteen vuoksi. On kuitenkin oletettavaa, että suurin osa sisäisen logiikan virheistä olisi havaittu analyserin järjestelmätesteissa, joissa tutkittiin kunkin kriteerin tarkistuksen toimivuutta.

4 Järjestelmätestaus

Suuri osa Koskelon testauksesta suoritettiin järjestelmätestauksena, sillä Koskelon valmiit komponentit liittyivät kukin yksitellen eAssariin.

4.1 Composer

Composerissa varsinaisesti testattavia osia ovat StaticTTK91Composer, FillinTTK91Composer, TTK91SyntaxChecker ja TTK91TaskParser. Composerien kohdalta erityisesti testattavaa oli, että ne palauttavat kenttien alkuarvot oikein tehtävää editoitaessa ja laittavat eteenopäin oikean Event-arvon. Tämän toiminnalisuuden lisäksi varsinaiset Composer.jsp:t ovat staattisia lomakkeita.

TTK91SyntaxCheckerissä testattavaa oli virheellisen datan hylkääminen, oikeanlaisen datan hyväksyminen ja oikeanlaisen palautelomakkeen tulostaminen. Palautelomakkeelta tarkistettavaa oli myös oikeiden palautetekstien palauttaminen editeventin yhteydessä.

4.1.1 Tehtävänmäärittely

Tehtävänmäärittelyä on testattu seuraavilla syötteillä, jotka pitäisi hyväksyä:

- Tehtävälomakkeella voi laatia onnistuneesti tyhjän tehtävän
- Tehtävänlomakkeella voi syöttää kunkin yksittäisen kriteerin onnistuneesti, lukuunottamatta tiettyjä erityistapauksia kriteerivaatimuksissa.
- Tehtävälomakkeella voi laatia tehtävän, jossa kaikki kriteerit on määritelty, ja ne ovat oikein.

4.1.2 Virheellinen tehtävänmäärittely

Tehtävänmäärittelyä on testattu seuraavilla virheellisillä syötteillä, joista palataan takaisin tehtävänmäärittelysivulle:

- Yritetty antaa syötteinä jotain muuta kuin pilkulla erotettuja lukuja
- Yritetty luoda täyttötehtävää, jossa ei ole määritelty täytettävää osaa tai, jossa on jätetty jompi kumpi täytettävää osaa määrittelevistä sulkeista pois.

- Yritetty luoda täyttötehtävää, josta puuttuu malliratkaisu, ja jossa ei näin ollen ole määritelty täyttöosaa.
- Yritetty antaa ohjelman maksimikoko, käskyjen maksimimäärä ja optimaalinen koko jonain muuna kuin kokonaislukuna.
- Yritetty valita vertailu simuloituun malliratkaisuun ja silti jätetty malliratkaisu tyhjäksi
- Yritetty käyttää konekäskyinä jotain muita merkkijonoja kuin TTK91-käskyjä
- Yritetty käyttää konekäskyjen erottimena jotain muuta kuin puolipistettä.
- Yritetty syöttää kriteerejä, joiden syntaksi ei ole määritellyn ulkoasun (a>b); mukainen, lukuunottamatta sulkeita ja whitespacea.
- Yritetty syöttää kriteerejä, joiden oikeanpuoleinen vertailtava ei ole luku, kun on valittu vertailu simuloituun lopputilaan.
- Yritetty syöttää virheellistä muistiviitteiden määrään liittyvää kriteeriää, käytännössä kaikki muut muodot kuin vertailuoperaattori + luku.
- Yritetty syöttää virheellisiä arvoja näyttötulosteille sekä virheellisesti eroteltuja näyttötulosteita.

4.1.3 Tehtävän editointi

Seuraavia tehtäväneditoinnin ominaisuudet on todettu toimiviksi

- Kriteerien lisäys jo olemassaolevien seuraksi
- Poistettu osa kriteereistä
- Kokonaisen kriteeriryhmän tyhjennys
- Virheellisten kriteerien hylkääminen kuten uuden tehtävänmäärittelyn kanssa

4.1.4 Palautekentät

Seuraavat palautekenttien ominaisuudet on testattu toimiviksi tehtävän muokkauksen yhteydessä:

- Tyhjien palautteiden palauttaminen (ei tulostu ”null”)
- Yksittäisen palautteen palauttamista (palautuu oikeaan kenttään)

- Kaikkien palautteiden palauttamista
- Kentän tyhjennys (Palautteen saa poistettua)

4.2 Displayer

Displayereitä testattiin muutamalla erilaisella tehtävänannolla. Staattinen displayer ei varsinaisesti tee muuta, kuin esitäytä opiskelijan vastauksen takaisin lomakeeseen, joten sen kohdalla testtavaa ei ollut paljoa. Täyttötehtävän diplayer sen sijaan piilottaa mallivastauksesta osan, joten testattiin, että piilotettava osa on oikea. Diplayerit testattiin lähinnä JUnitilla ja silmämääräisesti katsomalla, että palautettu html-osa on oikeanlainen.

Displayerin järjestelmätestaus tehtiin samaan aikaan kuin analyserin. Kun analyserille syötettiin erilaisia ratkaisuyrityksiä, testattiin samalla displayeria. Displayertest2.jsp:hen lisätyt testitehtävät toimivat samalla myös Koskelon displayerien testimateriaalina.

4.3 Analyser

Analyserin järjestelmätestejä varten lisättiin testitehtäviä Diplayertest2.jsp:hen. Analyserissa testattavia osia olivat kunkin määritellyn kriteerin tutkiminen, laadullisen palautteen käsittely sekä palautteen muodostaminen.

4.3.1 Yksittäisen kriteerin toimiminen

- Maksimimäärä suoritetuille käskyille. Testattu toimivan, tämä hyödyntää suoraan Titokoneen ominaisuutta suorittaa enintään n käskyä.
- Suoritettujen käskyjen hyväksymisraja. Testattu toimivan, haetaan suoraan Titokoneelta. Oletetaan Titokoneen palauttavan oikean lukumäärän.
- Optimaalinen käskyjen lukumäärä. Testattu toimivan, käytetään samaa arvoa kuin suoritettujen käskyjen hyväksymisraja.
- Muistiviitteiden määrä. Testattu toimivan, käytettyjen muistiviitteiden määrä voidaan hakea suoraan Titokoeesta. Oletetaan Titokoneen palauttavan oikean lukumäärän.
- Sallitut käsky. Testattu toimivan ainakin yksittäisillä käskyillä ja yksittäisillä laadullisilla käskyillä.
- Kielletyt käskyt. Testattu toimivan ainakin yksittäisillä käskyillä ja yksittäisillä laadullisilla käskyillä.

- Muistikriteerit. Eivät toimi täysin, ks. kohta ”Symboliset muistiviitteet”. Tietyn muistipaikan tutkivat kriteerit toimivat.
- Rekisterikriteerit. Yksittäiset tavalliset ja laadulliset rekisterikriteerit on testattu toimivan.
- Tulosteet. Tulosteet on testattu toimivan.

4.3.2 Rekistriin arvon testaaminen, tulostuksen testaaminen

Tehtävänanto:
Hae muuttujasta X arvo 15 rekisteriin R1 ja tulosta se näytölle.

Tehtävän kriteerit:

- Suoritettujen käskyjen maksimimäärä: 500
- Hyväksymisen yläraja: 4
- Ihannekoon yläraja: 3
- Vertailu valmiisiin kriteereihin
- Vaadittu käsky: LOAD
- Kielletty käsky: JUMP
- Rekisterien sisältö: R1 == 15
- Muistiviitteiden määrä: >= 1
- Tulosteet näytölle: (0,15)

Hyväksyttävä ratkaisu:
X DC 15
LOAD R1, X
OUT R1, =CRT
SVC SP, =HALT

Havainnot:
Tehtävä voidaan ratkaista ja kriteerien täyttyminen tutkitaan oikein.

4.3.3 Simuloitujen tulosteiden vertaaminen

Tehtävänanto:
Hae muuttujasta X arvo 15 rekisteriin R1 ja tulosta se näytölle.

Tehtävän kriteerit:

- Syötteet: 1,2,3
- Piilotetut syötteet: 4,5,6
- Vertailu simuloituun lopputilaan
- Tulosteet näytölle: (0,0), (1,1), (2,2)

Hyväksyttävä ratkaisu:
in r1, =KBD;
out r1, =CRT;
in r1, =KBD;
out r1, =CRT;
in r1, =KBD;
out r1, =CRT;
svc sp, =HALT;

Havainnot:

- Usean outputin vertailu simuloidun malliratkaisun outputteihin toimii.
- Myöskin yksinkertaiseen malliratkaisuun vertailu toimii.
- Salaiset syötteet annetaan oikein ohjelmalle, ja niihin vertailu toimii.

4.3.4 Muistikriteerien testaus

Tehtävänanto:
Lue kolme lukua syötteenä ja tallenna ne muistiin.

Tehtävän kriteerit:

- Syötteet: 1,2,3
- Piilotetut syötteet: 4,5,6
- Vertailu simuloituun lopputilaan

- Muistikriteerit: (10==10),(11==11),(12==12),(13==13),(14==14)
- Vaadittu käsky: IN

Hyväksyttävä ratkaisu:
X DS 3;
IN R1, =KBD;
IN R2, =KBD;
IN R3, =KBD;
LOAD R4, =X;
STORE R1, X;
ADD R4, =1;
STORE R2, X;
ADD R4, =1;
STORE R3, X;
SVC SP, =HALT;

Havainnot:

- Usean muistipaikan yhtäsuuruusvertailu toimii.
- Vertailu kaksilla syötteillä malliratkaisuun toimii.
- Symboliset muistiviitteet eivät edelleenkään toimi.
- Määriteltyjen kriteeriryhmien palaute tulee näkyviin vaikka on tyhjä.

5 Loppukäyttäjillä testatut testitapaukset

Loppukäyttäjiksi testaamaan järjestelmää värvättiin Ilari Nieminen ja Kirsi Jokisalo.

5.1 Testauskerta 1

Testaaja: Ilari Nieminen

Valvoja: Johan Brunberg

Testaajalle annettiin järjestelmän käyttöohje ennen testiä. Järjestelmän toimintaa ei demottu erikseen.

5.1.1 TTK-91-tehtävän laatiminen

Mitä testaaja teki: Laati yksinkertaisen TTK-91-ohjelmointitehtävän, jossa noudetaan muistipaikasta luku ja tulostetaan se näytölle.

Havaitut virheet:

- eAssarin Tehtävänasetukset-sivu pyytää harhaanjohtavasti tehtävänantoa, jonka opiskelijalle näkyvä muotoilu annetaan TTK-91-tehtävätyypeissä kuitenkin vasta seuraavalla sivulla.
- Tehtävänasetukset-sivusta ei selvästi käy ilmi, että suurin osa tehtävän tiedoista on tarkoitus antaa vasta seuraavilla sivuilla.
- Sopivalla tavalla virheellinen kriteerimäärittely tuotti HTTP Status 500 -virheen, jota valvoja ei saanut jälkeenpäin toistumaan.
- Logiikka, jolla kiellettyjen käskyjen käyttämisestä tai käyttämättä jättämisestä tulostettavat palautteet määräytyvät, oli hankala hahmottaa.

Testaajan kommentit: Olisi toivonut mahdollisuutta käsitellä kriteereitä ja niiden palautetekstejä samanaikaisesti. Käyttöohjeen screenshot-esimerkin olisi pitänyt iskeä silmään aikaisemmin.

Testitapaukseen tulkinta: Vakavia puutteita ei havaittu.

5.1.2 Täydennystehtävän laatiminen

Mitä testaaja teki: Laati täydennytstehtävän, jossa summataan lukuja nollaan asti.

Havaitut virheet:

- Käyttöohje ei kerro, että myös opiskelijalle annettu valmis ohjelmakoodi lasketaan analysoidessa mukaan.

Testaajan kommentit: ”Kokeilemalla ja erehtymällä toimintalogiikka selviää.”

Testitapaukseen tulkinta: Järjestelmän käytön oppiminen vaatii jonkin verran rohkeutta ja uteliaisuutta kokeilla eri toimintoja.

5.2 Testauskerta 2

Testaaja: Kirsi Jokisalo

Valvoja: Johan Brunberg

5.2.1 Staattisen tehtävän ratkaiseminen

Mitä testaaja teki: Ratkaisi tehtävän, jossa oli noudettava luku muistista ja tulostettava se näytölle.

Havaitut virheet: (ei havaittu)

Testaajan kommentit: Suora linkki TTK-91-kielen speksiin olisi näppärä.

Testitapaukseen tulkinta: Hyvin toimii, ei ongelmia.

6 Korjaamattomat virheet ja parannusehdotukset

Tämä luku käsittää kaikki järjestelmän testauksessa havaitut virheet, joita ei korjattu.

6.1 Koskelon komponenttien virheet

6.1.1 Puuttuvat merkkijonot

Projektiryhmä ei lisännyt ruotsinkielisiä merkkijonoja tehtävänmäärittelyä varten tietokantaan. Virhe on korjattavissa tekemällä ruotsinkieliset käännökset olemassaolevista suomen- ja englanninkielisistä merkkijonoista ja lisäämällä ne tietokantaan.

6.1.2 Dynaamisen tehtävätyypin puuttuminen

Ryhmä ei toteuttanut dynaamista tehtävätyyppiä ajanpuutteen vuoksi.

6.1.3 Englanninkielisen tehtävänmäärittelyn jumittuminen

Englanninkielinen ja miksei muunkinkielinen, tehtävänmäärittely jumittuu, jos palautekentissä tai aikaisemmin käyttää merkkiä '. Tämä johtuu Assarilta periytyvästä virheestä "Tietokantasyntaksin vaarallisuus".

6.2 Titokoneelta periytyvät virheet

6.2.1 Symboliset muistiviitteet

Titokoneen rajapinnan toteutus on vajavainen symbolitaulun osalta. Käytännössä siis, pyynnöllä getSymbolTable() saa ulos vain tyhjän HashMapin, joka ei sisällä käytettyjä symboleita. Tästä myöskin seuraa, että Koskelon toiminta symbolisten muistiviitteiden kanssa on testaamatonta.

Tästä seuraa, että muistikriteerejä määriteltäessä ei ole mahdollista määritellä kriteeriä, jossa on symbolinen muistiviite niin, että muistikriteeri tulisi ikinä täytetyksi. Käytännössä muistikriteeri jää täyttymättä koska Titokoneen symbolitaulusta ei löydy tarvittavaa symbolia.

6.2.2 Muistiviitteiden määrälaskuri

Titokone ei ilmeisesti käytä musitiaan aivan kuin sen ohjelmakoodista olettaisi, joten pelkkä RandomAccessMemoryyn lisätty laskuri, joka laskee getValue-metodin kutsut ei riitä muistiviitteiden laskemiseksi. Tämän seurauksena lasketaan vain suoritettujen konekäskyjen hakuun menevät muistiviitteet. Tämä ongelma saattaa olle sukua symbolisten muistiviitteiden virheelle.

6.3 Assarilta periytyvät virheet

6.3.1 Tehtävien editointinäkymä vaatii javascriptiä

Käytännössä editointinäkymän linkit ovat muotoa:

```
<a href='javascript: document.TASK373.event.value=4,
document.TASK373.submit();'
onClick='document.TASK373.event.value=9;'>
>></a>
```

vastaava olisi käyttäen HTTPGet-dataa ilman javascriptiä suunnilleen:

```
<a href="formintarget?event=9&taskid=373&event=4">
```

Tämä vaatisi myös http-get requestin käsittelyn lisäyksen TaskDefinitionControlleriin ja muutos ei välttämättä ole näin yksinkertainen. Kuitenkin javascriptin käyttö tässä kohtaa on täysin turhaa ja vain vähentää järjestelmän tukea vanhemmille selaimille.

6.3.2 Tehtävän nimen muuttaminen

Tehtävän nimeä ei ilmeisesti voi muuttaa vaikka nimi onkin editoitavissa tehtävänmäärittelysivulla.

6.3.3 Tehtävänlaadinnan ”Puuttuvat pilkku”

Jotkin merkkijonot aiheuttavat erikoisia Oraclen virheilmoituksia tehtävänmäärittelyssä. Nämä virheet eivät pääse käyttäjälle asti, vaan asia ilmenee käytännössä koko tehtävänmäärittelyn jumittumisena.

Tämä käytös havaittiin ainakin luotaessa englanninkielistä tehtävää ja johtuu ilmeisesti käytetystä lainausmerkistä. Tämän virheen seurauksena myöskin jo osittain onnistuneesti määritelty tehtävä tyhjenee tehtävää editoitaessa.

6.3.4 Tietokantasyntaksin vaarallisuus

Mahdollisesti virhe ”puuttuva pilkku” aiheutuu tästä virheestä. Käytännössä siis Assarin toteuttamat tietokantalausekkeet käyttävät Javan Statement-olioita tietokantakyselyihin tekemättä minkäänlaisia tarkistuksia merkkijonoille, jotka lisäävät tietokantalausekkeisiin. Näitä lausekkeita ei myöskään ilmeisesti täysin suoraan voida vain muuttaa PreparedStatementeiksi korjaamatta myöskin Assarin koodia, jossa oletetaan allaolevan kyselyn muodostuvan Statement-lauseista. Erityisesti tietokantaan tallentavien lauseiden osalta olisi syytä miettiä joko erikseen merkkijonojen sanitointia ennen niiden lisäystä Statementtiin tai muun tietokantakyselylogiikan korjausta.

Tästä virheestä periytyy myös se, että Koskelon tekstikenttiin syötetyssä datassa sopivassa järjestyksessä esiintyvät tietokantamerkkijonot voivat aiheuttaa ennalta odottamattomia ilmiöita tietokannassa.

6.3.5 Tehtävien tallentaminen tietokantaan

Projektin aikana käytössä ollut tehtävien tallentamisjärjestelmä lisää uuden tehtävän vain tietokannan task-tauluun. Jotta tehtävä voitaisiin näyttää, pitäisi tehtävä lisätä myös taskinmodule-tauluun. Taskinmodule-taulu liittää tehtävän tiettyyn kurssiin ja moduuliin. Projektin aikana lisäsimme tehtävät käsin taskinmodule-tauluun kurssille courseX1 ja moduuliin moduleX1.

6.3.6 Tietokannan nimen korjaus käsin

Asennuksen jälkeen luokkaan DatabaseBase pitää muuttaa tietokannan asetukset, jonka jälkeen luokka on käännettävä.

6.4 Esille tuleet korjausehdotukset

6.4.1 Symboliset muistiviitteet

Ryhmä kokee, että yksin tätä virhettä on turha lähteä korjaamaan Titokoneesta, vaan koko Titokone tarvitsisi keskittynyttä korjausta ja jatkokehitystä. Erityisesti, kun kyseinen virhe ei ole ainoa tunnettu Titokoneen virhe, on syytä harkita, että koko Titokoneen kääntäjä kirjoitettaisiin uudestaan. Tällä hetkellä kääntäjä on kooltaan massiivinen ja käytännössä erittäin vaikeasti ylläpidettävä. Erityisesti kääntäjää uusiksi kirjoitettaessa olisi huomioitava sen myöhempi helppo ylläpidettävyys ja toimintalogiikan parempi dokumentointi.

6.4.2 Dynaamisen tehtävätyypin lisäys

Käytännössä dynaaminen tehtävätyyppi voinee hyödyntää ainakin hyvin pitkälle Koskelon toteuttamaa analyseria. Periaatteessa lisäys vaatinee jonkin esikäsittelijän lisäämisen, joka parsii malliratkaisun suoritettavaan muotoon. Tämän jälkeen varsinainen toteutettu analyser kykenee tekemään vertailut, kunhan sille on annettu kriteerit myöskin paikattuina dynaamisen osan kohdalta.

Ennen dynaamisen tehtävän toteutusta on syytä miettiä tarkasti miten käytännössä taulukon esittäminen ohjelmakoodissa tehdään. Koskelo ei keksinyt tähän yleispätevää ratkaisua ja ongelma osoittautuikin arvioitua hankalammaksi.

6.4.3 Assarin tietokannankäsittely

Koko eAssari-järjestelmän tietokantakyselyt on syytä käydä tarkasti läpi. Koskelo ei ole tuottanut omia tietokantakyselyitä, joten tietokannankäsittelystä johtuvat virhee Koskelossa eivät ole ryhmän omaa tuotosta. Todennäköisesti suuri osa tietokantavirheistä saataisiin korjattua siirtymällä käyttämään PreparedStatementtia Statementin sijaan tietokantaan tallennettaessa dataa.

eAssarissa tietokantaa käsitellään usean luokan kautta, jonka seurauksena järjestelmän tietokantayhteyksiä on hankala hahmottaa ja ylläpitää. Kyselyiden toimintaa suositellaan korjattavaksi ja järjestettäviksi yhteen komponenttiin.