Semantic Web - Metadata editor

Ohjelmistotuotantoprojekti, kesä 2002
Ohjelmistotuotantoryhmä 1, Meedio

http://www.cs.Helsinki.FI/group/meedio

Mikko Apiola (M.A.)
Ari Inkovaara (A.I.)
Miikka Junnila (M.J.)
Justus Karekallas (J.K.)
Pekko Parikka (P.P.)

Helsinki 11. heinäkuuta 2002
Testaussuunnitelma
Helsingin yliopisto

Tietojenkäsittelytieteen laitos

Versiohistoria

Versio	Pvm	Laatija	Kommentti
0.1	10.7.2002	Pekko Parikka	Alustava versio kommentoitavaksi
0.2	11.7.2002	Justus Karekallas	Tarkistettu ja korjattu versio

Sisältö

Liitteet

1 Johdanto (P.P.)

Tämä testaussuunnitelma tukee seuraavia projektin tavoitteita:

- Tunnistaa ne ohjelmiston osat, jotka pitää testata
- Esittelee testauksen vaatimukset.
- Kuvaa testausstrategiaa
- Luettelee testitapaukset

Testauksessa testataan Meedio ryhmän määrittelydokumentissa määritellyt ensimmäisen prioriteetin mukaiset toiminnot. Testausdokumentti on tarkoitettu ohjeeksi ja avuksi ryhmälle testausta varten, ei pikkutarkasti seurattavaksi.

1.1 Testattava tuote (J.K.)

Tuote on editoriohjelma, jolla tarkistetaan, että XML-tiedosto on validissa muodossa. Validoituun tiedostoon lisätään RDF-tietoa. Editoriohjelma on tarkoitettu museotyöntekijälle, joka käsittelee museon esinetietokannasta muodostetun XML-tiedoston sisältämiä esine-kortteja ja hän muokkaa esineisiin liittyvät tiedot oikeamuotoisiksi ja lisää esineisiin museon esinetietojen ontologian mukaisen luokittelun.

2 Ympäristövaatimukset (J.K.)

Luvussa kuvataan testauksessa käytettävä laitteisto ja ohjelmat, joilla testaus suoritetaan sekä testauksessa käytettävä materiaali.

2.1 Laitteisto (J.K.)

Ohjelmistoa testataan seuraavalla kokoonpanolla:

1) Luokka D327, kuussaari

- Käyttöjärjestelmä: Red Hat Linux 6.1
- Prosessori: 233Mhz Pentium
- Keskusmuistia: 128Mb
- Levytilaa: 3,77 Gb

2) Luokka D326

- Käyttöjärjestelmä: Linux, Windows 2000
- Prosessori: 1333MHZ Athlon
- Keskusmuistia: 528MB
- Levytilaa:

2.2 Ohjelmisto (J.K.)

Ohjelmistoa testataan seuraavissa käyttöjärjestelmissä:

-Windows 2000

-Linux

Molemmissa käyttöjärjestelmissä täytyy olla Jakarta Tomcat asennettuna.

Windows ympäristössä testauksessa käytetään Microsoft Explorer5.5- ja Netscape Navigator4.7x-selaimia. Linux ympäristössä ohjelmisto testataan Netscape Navigator4.7x-selaimella.

2.3 Turvallisuus (J.K.)

Testaus ei saa kaataa yliopiston palvelimia tai muita järjestelmiä.

2.4 Apuvälineet (J.K.)

Testauksessa käytetään Finnish Museums Online (FMO) keskushallinnolta saatavaa materiaalia Materiaali koostuu museoesineiden ontologian kuvaavasta RDF-skeemasta ja esineitä kuvaavista XML-korteista sekä korttien mukana toimitettavasta tai ryhmän itsensä tekemästä XML-skeemasta. Tämän lisäksi testaukseen voidaan myös käyttää Yliopistomuseon XML-kuvatietokantaa ja Meedioryhmän siihen tekemää XML-skeemaa sekä RDF-skeemaa.

3 Henkilöstö ja vastuualueet (P.P.)

Testauksen suorittavat projektiryhmän jäsenet:

- Mikko Apiola
- Justus Karekallas
- Pekko Parikka
- Miikka Junnila
- Ari Inkovaara

Testauksen vastuualueet jakaantuvat seuraavasti:

Tehtävä	***Vastuuhenkilö 1***	***Vastuuhenkilö 2***
Testauksen suunnittelu	Justus Karekallas	Pekko Parikka
Testauksen johto, yksikkötestaus	Justus Karekallas	Pekko Parikka
Testauksen johto, integrointitestaus	Ari Inkovaara	Pekko Parikka
Testauksen suoritus	Kaikki ryhmän jäsenet	

Tehtävä	*Vastuuhenkilö 1*	*Vastuuhenkilö 2*
Testauksen tarkastus	Kaikki ryhmän jäsenet	
Testauksessa ilmenneiden virheiden korjaaminen	Ko. osajärjestelmän tekijä.	

4 Testattavat toiminnot (J.K.)

Tässä luvussa kerrotaan kaikki ne ohjelmiston toiminnot, jotka testataan. Toiminnot on jaoteltu omiin lukuihin sen mukaan mihin moduuliin ne kuuluvat.

4.1 Käyttöliittymän toiminnot (J.K.)

1. XML-kortin tietojen havainnollinen näyttäminen
2. XML-kortin tietojen muokkaus
3. XML-kortin virheellisten kenttien korostus
4. Ontologian esittäminen dynaamisena hierarkkisena listana
5. Semantiikan liittäminen XML-korttiin
6. Kortin tallennus
7. XML-tiedoston tallennus
8. Käsiteltävän XML-tiedoston valmiiden korttien tallennus
9. Kuvan näyttäminen käyttäjälle
10. Tiedoston sisältämien korttien näyttäminen havainnollisesti

4.2 XML-käsittelijän toiminnot (J.K.)

1. XML-Skeeman lukeminen
2. XML-kortin lukeminen
3. XML-kortin validoiminen halutun skeeman mukaisesti
4. Tarkkojen tietojen antaminen XML-kortista, ja sen oikeellisuudesta

4.3 RDF-käsittelijän toiminnot (J.K.)

1. Konfiguraatiotiedostossa määritellyn RDF(S)-muotoisen ontologian lukeminen sisäiseen tietorakenteeseen
2. Jena-oliomallin mukaisen rakenteen muodostaminen RDF(S)-ontologiasta
3. Edellisen kohdan mukaisen rakenteen antaminen käyttöliittymälle
4. Tietojen antaminen käyttöliittymälle kohdan 4. mukaisista luokista
5. Ontologian ilmentymän tallennus XML-korttiin
6. Uusien instanssien vertaaminen olemassa oleviin instansseihin
7. Dokumenttien validiuden syntaktinen tarkastaminen

4.4 Tiedostonkäsittelijän toiminnot (J.K.)

1. XML-tiedoston kopioiminen järjestelmän käyttöön
2. Kortin tilatietojen lisääminen/muuttaminen (käsittelyssä / ei käsitelty / valmis)
3. Käsiteltävien XML-korttien ID-tiedojen, otsikoiden ja tilatietojen antaminen
4. Kortin antaminen XML-tiedostosta
5. Kortin tallentaminen XML-tiedostoon
6. Tiedoston tallentaminen haluttuun paikkaan
7. Valmiiden korttien tallentaminen haluttuun paikkaan tiedostosta
8. Käsittelyssä olevan XML-tiedoston tuhoaminen

4.5 Ohjaimen toiminnot (J.K.)

1. Tiedostonimien antaminen
2. Käsittelyssä olevien XML-tiedostojen nimien antaminen
3. XML-tiedoston avaaminen
4. Viimeksi käsitellyn XML-tiedoston avaaminen
5. XML-skeeman lukeminen ja kenttien linkittäminen
6. XML-kortin validoiminen ja kortin tietojen linkittäminen
7. Rikastetun XML-kortin tallentaminen
8. Ontologian juuriluokkien antaminen
9. Luokan aliluokkien antaminen
10. Luokan antaminen
11. Instanssikyselyiden tekeminen
12. Liitettyjen tietojen antaminen
13. Valmiiden XML-korttien tallentaminen
14. XML-tiedoston tallentaminen
15. XML-tiedoston poistaminen
16. Luokkapolkujen antaminen

4.6 Vaadittava tulosaineisto (P.P.)

Testauksesta saadaan tuloksena testausdokumentti. Testausdokumentissa on liitteenä virhelista kommentteineen. Jokaisesta testistä on liitteenä myös kirjanpito eli loki.

5 Testausmenettely ja testitapaukset (P.P.)

Ohjelmiston jokainen moduuli testataan erillisenä komponenttina. Komponentit testataan "white box" -menetelmällä, jotta voidaan testata niiden sisäistä toimivuutta. ”White box” –testauksen suorittaa kunkin moduulin tekijä. Testauksessa on tarkoitus käyttää haarakattavuutta, eli jokaisen ehtolauseen kaikki vaihtoehdot käydään läpi. Tämä takaa sen, että jokainen moduulin koodirivi tulee suoritettua ainakin kerran. Tämän lisäksi kukin moduuli testataan ”black box” –menetelmällä. Tätä varten jokaiselle komponentille määrätään erityinen testaaja, joka on eri henkilö kuin kyseisen komponentin tekijä. ”Black box” –menetelmässä testataan moduulin ulkoista toimintaa ilman, että tunnetaan sen sisäistä rakennetta.

5.1 Testitapausluokat (J.K.)

Testitapaukset luokitellaan tärkeysjärjestykseen. Nämä tärkeysluokat ovat:

A) Erittäin tärkeä

Järjestelmän toiminnan testaamisen kannalta välttämätön testitapaukset.

B) Vähemmän tärkeä

Testitapaukset, jotka eivät vaikuta keskeisesti järjestelmän toimintaan.

Testitapauksille annetaan virheluokitukset seuraavan jaotuksen mukaan:

1. Vakava virhe.
   Virhe kaataa koko järjestelmän.

2. Keskimääräinen virhe.
   Käyttäjälle aiheutuu haittaa sattuneesta virheestä.

3. Lievä virhe.
   Virheen aiheuttama vahinko on pieni, eikä se vaikuta järjestelmän toimintaan.

Virheet korjataan siten, että korjaus aloitetaan vakavista virheistä. Kun vakavat virheet on korjattu, siirrytään keskimääräisiin virheisiin. Lopuksi korjataan lievät virheet, mikäli aika riittää.

5.2 Kattavuus (P.P.)

Kaikista osajärjestelmistä on tarkoitus testata kaikki ensimmäisen prioriteetin toiminnot mahdollisimman kattavasti.

Tarkoitus on, että käyttöliittymän toimintoja testatataan kutakin vähintään kaksi kertaa eri arvoilla. Kaikki muut testit automatisoidaan.

Muuttujia testataan erilaisilla arvoilla - sekä oikeilla että virheellisillä, jotta nähdään miten ne toimivat eri tilanteissa.

Seuraavassa muutamia esimerkkejä testattavista arvoista:

- String
  - Tyhjä
  - Merkki
- integer
  - ylivuoto
  - alivuoto
  - positiivinen arvo
  - negatiivinen arvo
  - tyhjä
- boolean
  - oikea arvo (true)
  - väärä arvo (false)
- raja-arvot, silmukat
  - annetaan arvoja, joilla testataan esim. silmukan toimivuus; jos raja-arvona on 10, testataan ainakin arvoilla 9, 10, 11

Muiden muuttujien testausarvoja ei käydä läpi tässä dokumentissa vaan ne jätetään testaajien arvioitaviksi.

5.3 Käyttöliittymän testaus (P.P.)

Käyttöliittymässä testataan kaikki 1. prioriteetin toiminnot . Käyttöliittymän toimintojen testauksen yhteydessä testataan myös onko käyttöliittymä helppokäyttöinen ja tehokas. Käyttöliittymän toimintoja testataan sekä oikeilla että väärillä syötteillä. Käyttöliittymän testaamisessa käytetään apuna asiakkaan edustajia, joiden avulla testataan käyttöliittymän käytettävyyttä.

1. XML-kortin tietojen havainnollinen näyttäminen
   - Tietoja sisältävien kenttien generointi ja ulkoasu
   - XML-kortin tietojen vastaanottaminen XML-käsittelijältä

2. XML-kortin tietojen muokkaus
   - Kenttien editoinnin toiminta
   - Täyttöohjeiden esittäminen
   - Muutettujen tietojen välittäminen XML-käsittelijälle

3. XML-kortin virheellisten kenttien korostus
   - Värikoodaus
   - XML-käsittelijän ja käyttöliittymän välinen kommunikointi

4. Ontologian esittäminen dynaamisena hierarkkisena listana
   - Ulkoasu
   - RDF-käsittelijän luoman rakenteen vastaanottaminen
   - Rakenteen tulkitseminen ja piirtäminen

5. Semantiikan liittäminen XML-korttiin
   - Semantiikan oikeellisuus
   - Kommunikointi RDF-käsittelijän kanssa

6. Kortin ja XML-tiedoston tallennus
   - Tallennuksen suorittaminen
   - Tallennettavan kortin välitys tiedostonkäsittelijälle

7. KUVAN NÄYTTÄMINEN KÄYTTÄJÄLLE

- **KUVAN SIJAINTI KÄYTTÖLIITTYMÄSSÄ**
- Kuvan piirtäminen
- Kuvan sijaintipaikan selvittäminen

8. Tiedoston sisältämien korttien näyttäminen havainnollisesti

- **AVATUN TIEDOSTON ESITTÄMINEN**
- Tiedoston sisältämien korttien esittäminen
- Kommunikointi tiedostonkäsittelijän kanssa

5.4 XML-käsittelijän testaus (P.P.)

XML-käsittelijästä testataan kaikki siihen liittyvät 1. prioriteetin toiminnot.

1. XML-Skeeman lukeminen
   - Skeeman lukeminen
   - Luetun skeeman tarkistus

2. XML-kortin lukeminen
   - Kortin lukeminen
   - Luetun kortin tarkistus

3. XML-kortin validoiminen halutun skeeman mukaisesti
   - Kortin vertaaminen skeemaan
   - Kortin validiksi muuttaminen
   - Validoidun kortin tarkistus

4. Tarkkojen tietojen antaminen XML-kortista, ja sen oikeellisuudesta
   - Kortista käyttöliittymälle generoitava rakenne
   - Kortin tietojen oikeellisuuden kuvaaminen

Testauksessa käytetään JUnit framework –testausympäristöä (luku 5.10).

5.5 RDF-käsittelijän testaus (P.P.)

RDF-käsittelijästä testataan kaikki siihen liittyvät 1. prioriteetin toiminnot.

1. Konfiguraatiotiedostossa määritellyn RDF(S)-muotoisen ontologian lukeminen sisäiseen tietorakenteeseen
   - Ontologian sijainnin selvittäminen
   - RDF-skeeman lukeminen
   - Luetun skeeman tarkistaminen

2. Jena-oliomallin mukaisen rakenteen muodostaminen RDF(S)-ontologiasta
   - Rakenteen oikeellisuus
   - Rakenteen muodostaminen tietorakenteeseen luetusta ontologiasta

3. Edellisen kohdan mukaisen rakenteen antaminen käyttöliittymälle
   - Rakenteen ulkoasu
   - Rakenteen oikeellisuus

4. Tietojen antaminen käyttöliittymälle kohdan 3. mukaisista luokista
   - Luokkarakenteessa liikkuminen
   - Luokkarakenteen osarakenteiden ulkoasu

5. Ontologian ilmentymän tallennus XML-korttiin
   - Ilmentymän liittäminen korttiin
   - Päivitetyn XML-kortin tarkistus

6. Uusien instanssien vertaaminen olemassa oleviin instansseihin
   - Instanssikyselyn tekeminen RDF-tietokantaan
   - Kyselyn tulosten käsittely
   - Kyselyn jälkeinen toiminta

7. Dokumenttien validiuden syntaktinen tarkastaminen

Testauksessa käytetään JUnit framework –testausympäristöä (luku 5.10).

5.6 Tiedostonkäsittelijän testaus (P.P.)

Tiedostonkäsittelijästä testataan kaikki siihen liittyvät 1. prioriteetin toiminnot.

1. XML-tiedoston kopioiminen järjestelmän käyttöön
   - Tiedoston lukeminen konfiguraatiotiedoston osoittamasta hakemistosta
   - Tiedoston kopioiminen järjestelmän hakemistoon
   - Tiedoston läpikäyminen XML-korttien tunnistenumeroiden selvittämiseksi sekä tunnistenumerot ja korttien tilatiedot sisältävän tiedoston tallennus
   - Tiedoston avaaminen järjestelmän käyttöön

2. Kortin tilatietojen lisääminen/muuttaminen (käsittelyssä / ei käsitelty / valmis)
   - Tilatiedon muuttaminen XML-tiedostossa olevaan korttiin
   - Tilatiedon muuttaminen korttien tilaa ja tunnistenumeroita säilyttävään taulukkoon

3. Käsiteltävien XML-korttien ID-tiedojen, otsikoiden ja tilatietojen antaminen
   - Otsikon määrittelevän attribuutin lukeminen konfiguraatiotiedostosta
   - Annetulla ID-numerolla varustetun kortin otsikon lukeminen XML-tiedostosta
   - ID-numeron ja tilatiedon lukeminen niitä säilyttävästä taulukosta

4. Kortin antaminen XML-tiedostosta
   - Kortin etsiminen tiedostosta
   - Kortin lukeminen tiedostosta

- Luetun kortin tarkistaminen

5. Kortin tallentaminen XML-tiedostoon
   - Oikean tallennuskohdan etsiminen tiedostosta
   - Entisen kortin päälle kirjoitus

6. Tiedoston tallentaminen haluttuun paikkaan
   - Annettuun hakemistoon tallennus

7. Valmiiden korttien tallentaminen haluttuun paikkaan tiedostosta
   - Valmiiden korttien tunnistaminen
   - Annetuun hakemistoon tallennus

8. Käsittelyssä olevan XML-tiedoston tuhoaminen
   - Tiedoston tuhoaminen annetulla nimellä

Testauksessa käytetään JUnit framework –testausympäristöä (luku 5.10).

5.7 Ohjaimen testaus (P.P.)

Ohjaimesta testataan kaikki siihen liittyvät 1. prioriteetin toiminnot. Testauksella pyritään selvittämään, että ohjain toimittaa oikeanlaista tietoa oikeasta paikasta oikeaan paikkaan. Tieto ei saa muuttua matkalla tiedon muokkaajalta tiedon vastaanottajalle. Testauksessa kiinnitetään erityisesti huomio siihen, että tietoa vastaanotetaan suunnitelluilla parametreilla. Vastaavasti myös eteenpäin toimitettavan tiedon on vastattava niitä parametreja, jotka on määritelty vastaanottavan osapuolen rajapinnassa. Mikäli tietoa muokataan ennen vastaanottajalle toimittamista, on muokkauksen suorittavan osion toiminta testattava.

Testauksessa käytetään JUnit framework –testausympäristöä (luku 5.10).

5.8 Ohjelman kaatumisen testaus (P.P.)

Testataan miten ohjelma kestää kaatumista. Tutkitaan miten käyttöliittymä reagoi, kun ohjain ajetaan alas sekä miten muu ohjelmisto reagoi, kun käyttöliittymä ajetaan alas. Mitään muuta ei testata.

5.9 Suorituskyvyn arviointi

Järjestelmän suorituskykyyn liittyvät testit, esim.

- Aikaan liittyvät suorituskyvyt – vasteajat

- Kuormitustesti - järjestelmän toiminta korkealla kuormituksella (esim. iso ontologia, paljon kortteja sisältävän xml-tiedoston avaaminen)

5.10 Junit framework –testaus (A.I. & J.K.)

Junit framework-testausta käytetään liitettäessä ohjelmiston eri osia toisiinsa. Tarkoituksena on tunnistaa aiemmin testattujen osien yhdistämisessä syntyvät virheet. Junit framework-testauksen avulla voidaan automatisoida ohjelmiston testaus. Testien suoritus, tulosten tallennus ja virheiden raportointi kuuluvat kaikki Junit framework-testauksen perustoimintoihin. Testaajan pääasialliseksi tehtäväksi jää testin suorittavan testiohjelman laatiminen. Testiohjelmat laaditaan luokkina, jotka määrittelevät testattavat asiat ja testausmenetelmät. Laadittujen luokkien ilmentymien avulla toteutetaan halutun tyyppiset testiohjelmat. Testiohjelmia voidaan ajaa Junit framework-ympäristön tarjoaman graafisen käyttöliittymän avulla useita samanaikaisesti.

Meedio-ryhmän metadataeditori-projektissa Junit framework-testausta on tarkoitus käyttää osajärjestelmien luokkien yhdistämisen yhteydessä sekä eri osajärjestelmien liittämisessä toisiinsa. Jokaisen osajärjestelmän laatija vastaa suunnitelemansa osion Junit-testauksen suunnittelusta. Osajärjestelmien integrointivaiheen testaussuunnitelman laatimiseen osallistuvat kaikki ryhmän jäsenet.

5.11 Testauksen tehtäväjärjestys

Valmiita moduuleja voidaan testata samanaikaisesti. Integrointitestaus aloitetaan tiedostonkäsittelijän ja ohjaimen yhteistyön testaamisesta. Tästä jatketaan siten, että liitetään XML-käsittelijä ohjaimeen. Seuraavaksi lisätään RDF-käsittelijä ohjaimeen. Lopuksi liitetään käyttöliittymä ohjaimeen ja testataan koko järjestelmää. Tämän jälkeen tehdään validointitestaus.

6 Integrointitestaus (P.P.)

Integrointitestauksessa varmistetaan järjestelmän eri moduulien ja kaikkien niihin liittyvien luokkien toiminta yhdessä. Integrointitestauksessa käytetään ”black box” -menetelmää. Kunkin luokan syötteiden ja tulosteiden oikeellisuus varmistetaan jakamalla niiden arvoalueet ekvivalenssiluokkiin, joista muodostetaan testiaineisto. Tämän jälkeen toteutetaan luokille testiaineiston mukaiset testitapaukset. Meedioryhmän metadataeditori testataan alhaalta ylös -menetelmällä. Tällöin ohjelmiston toteutus ja testaus aloitetaan luokkakaavion alimmista luokista. Järjestelmästä vielä puuttuvien alempia luokkia kutsuvien luokkien toimintaa simuloidaan erikseen koodattavilla ajuriluokilla. Näiden ajuriluokkien tulosteista voidaan todentaa kutsutun luokan tulosteen vastaavuus määriteltyyn.

7 Validointitestaus (P.P.)

Validointitestaus tehdään kun integrointitestaus on valmis ja siinä havaitut virheet korjattu. Validointitestauksen tarkoituksena on tarkastaa täyttääkö metadataeditori sille vaatimusmäärittelyssä asetetut vaatimukset. Tämä tapahtuu suorittamalla joukko testitapauksia "black box" –menetelmällä. Testitapauksista selviää täyttääkö ohjelmisto sille asetetut toiminnallisuus-, suorituskyky- ja muut vaatimukset sekä sen että dokumentaatio on oikeellinen.

8 Viitteet (J.K.)

- Junit framework, lisätietoa: http://www.juni.org
- XML (Extensible Markup Language), lisätietoa: http://www.w3.org/XML/
- XML-Skeema, lisätietoa: http://www.w3.org/XML/Schema
- RDF (Resource Description Framework), lisätietoa: http://www.w3.org/RDF/
- Java Server Pages, lisätietoa: http://java.sun.com/products/jsp/index.html
- JSP+tag libraries, lisätietoa: http://java.sun.com/products/jsp/taglibraries.html
- Jakarta Tomcat, lisätietoa: http://jakarta.apache.org/tomcat/index.html
- Sax (Simple API for XML), lisätietoa: http://www.saxproject.org/
- Dom (Document object model), lisätietoa: http://www.w3.org/DOM/
- Jena, lisätietoa: http://www.hpl.hp.com/semweb/jena-top.html
- HTML (Hypertext Markup Language), lisätietoja: http://www.w3.org/MarkUp/
- CSS (Cascading Style Sheets), lisätietoja: http://www.w3.org/Style/CSS/
- Java Applet, lisätietoja: http://java.sun.com/
- Java Servlet, lisätietoja: http://java.sun.com/