Kettu-Sovellusprojekti

Sovellusraportti

Henri Koskenranta
Kosti Kuokkanen
Antti Marttila
Terhi Taanonen

Versio: 0.4
Julkinen
28. tammikuuta 2008

Jyväskylän yliopisto

Tietotekniikan laitos

Jyväskylä

Hyväksyjä	Päivämäärä	Allekirjoitus	Nimenselvennys
Projektipäällikkö	__.__.2008		
Tilaaja	__.__.2008		
Ohjaaja	__.__.2008		

Tietoa dokumentista

Tekijät:

- Henri Koskenranta (HK) heolanko@jyu.fi 050-3077318
- Kosti Kuokkanen (KK) koalkuok@jyu.fi 044-5666304
- Antti Marttila (AM) antmatma@jyu.fi 050-3546127
- Terhi Taanonen (TT) ttaanone@jyu.fi 050-3624724

Dokumentin nimi: Kettu-projekti, Sovellusraportti
Sivumäärä: 31
Tiedosto: sovellusraportti.tex

Tiivistelmä: Tämä dokumentti on sovellusraportti Jyväskylän yliopiston tietotekniikan laitoksen Kettu-sovellusprojektille. Dokumentissa kuvataan Kettu-sovellusprojektin jatkokehittämää sovellusta.
Avainsanat: Foxability, ESOK, esteettömyys, Kettu

Versiohistoria

Versio	Päivämäärä	Muutokset	Tekijät
0.1	19.12.2007	Luonnoksen tekeminen aloitettu.	KK
0.2	14.1.2008	Useista luvuista työversiot. Runko hahmottuu.	KK,AM
0.3	17.1.2008	Ensimmäinen katselmoitava versio.	KK
0.4	28.1.2008	Katselmointikorjauksien toteutus.	KK, AM, HK, VI

Tietoa projektista

Kettu-sovellusprojekti jatkokehitti Firefox-selaimen Foxability-laajennosta ja tuotti siihen WCAG 1.0 suosituksen mukaiset esteettömyystestit.

Tekijät:

- Henri Koskenranta (HK) heolanko@jyu.fi 050-3077318
- Kosti Kuokkanen (KK) koalkuok@jyu.fi 044-5666304
- Antti Marttila (AM) antmatma@jyu.fi 050-3546127
- Terhi Taanonen (TT) ttaanone@jyu.fi 050-3624724

Tilaajat:

- Kimmo Aittokallio kimaitt@jyu.fi 014-2602746
- Antti Ekonoja anjoekon@jyu.fi 014-2602746
- Tommi Lahtonen tjlahton@jyu.fi 014-2602746
- Hannu Puupponen Hannu.Puupponen@adm.jyu.fi 014-2603734

Ohjaajat:

- Ville Isomöttönen vilisom@jyu.fi 0400-608130
- Tarmo Friman tatafrim@jyu.fi 044-5815816

Yhteystiedot:

- Sähköpostilistat: kettu@korppi.jyu.fi, kettu_opetus@korppi.jyu.fi
- Projektiarkisto: https://korppi.jyu.fi/list-archive/kettu/ind.html
- Opetusarkisto: https://korppi.jyu.fi/list-archive/kettu_opetus/ind.html
- Työhuone: Ag C225.3 / 014-2604971

Sisältö

1 Johdanto

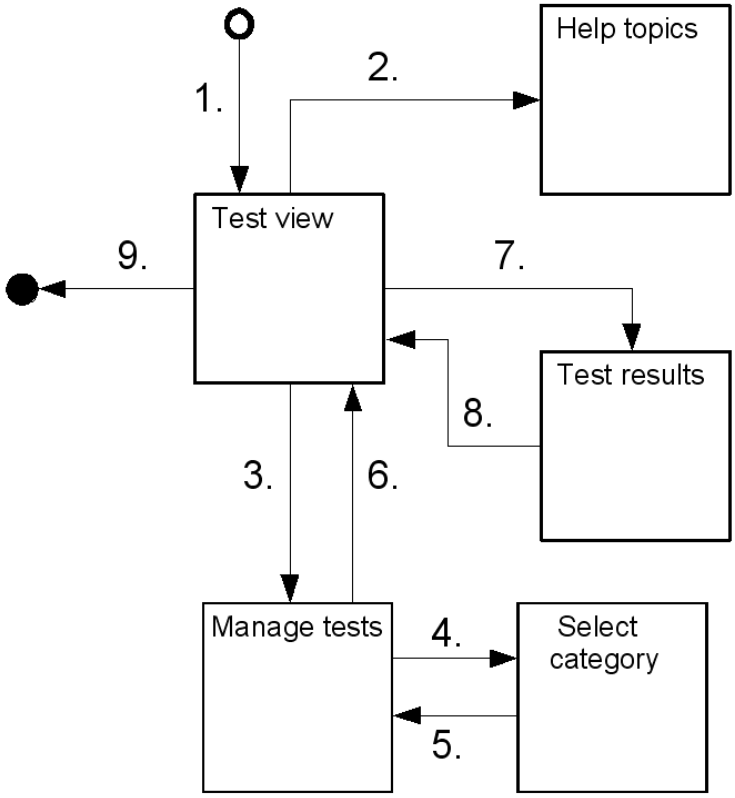
Kettu-sovellusprojekti jatkokehitti Foxability-laajennosta Firefox-selaimeen ja tuotti siihen WCAG 1.0 suosituksen mukaiset esteettömyystestit. Foxability-laajennoksen tarkoituksena on automaattinen verkkosivun esteettömyyden tarkistus.

Kettu-sovellusprojektin tilaajina olivat Jyväskylän yliopiston tietotekniikan laitos yhteistyössä ESOK-hankkeen kanssa. Kettu-sovellusprojektissa jatkokehitettiin Jyväskylän yliopiston tietojenkäsittelytieteiden laitoksen Foxability-projektin tekemää laajennosta, joka pohjautui Jukka Mäntylän pro gradu -tutkielmaan.

Tämä dokumentti on Kettu-sovellusprojektin sovellusraportti. Sovellusraportissa käydään läpi sovelluksen rakennetta, eri osien toimintaa, sekä toteutusratkaisuja. Luvussa 2 esitellään lyhyesti laajennoksen käyttöliittymä. Luku 3 käsittelee laajennoksen rakennetta ja toimintaa. Luvussa 4 esitellään laajennoksen luokat ja luvussa 5 käytettyjä tiedostoja ja tietorakenteita. Luvussa 6 suoritetaan katsaus testaukseen ja luvussa 7 kerrotaan laajennoksen teossa käytetystä ohjelmointiympäristöstä. Lopuksi, luvussa 8, kerrotaan jatkokehitysideoista.

2 Käyttöliittymä

Luvussa esitellään Foxability-sovelluksen käyttöliittymä ja siitä löytyvät toiminnallisuudet. Monipuolisempi kuvaus käyttöliittymän eri ominaisuuksista löytyy käyttöohjeesta. Käyttöliittymän ikkunoiden väliset suhteet ilmenevät kuvassa 2.1



Kuva 2.1: Käyttöliittymän ikkunoiden väliset suhteet.

1. Sovellus käynnistetään.
2. Luetaan käyttöohje.
3. Siirrytään *Manage tests* -ikkunaan lisäämään testejä. Myös testien poisto tapahtuu *Manage tests* -ikkunassa.
4. Valitaan lisättävä testimoduuli. Näkymä siirtyy *Select category* -ikkunaan, jossa valitaan lisättävälle testimoduulille kategoria.
5. Lisätty testimoduuli näkyy puurakenteessa valitun kategorian alla.

6. Lisätyt testit ilmestyvät näkymään myös *Test view* -ikkunaan.
7. Valitaan testejä ja suoritetaan ne, jolloin näkymä siirtyy *Test results* -ikkunaan.
8. *Test results* -ikkunan suljettaessa näkymä siirtyy *Test view* -ikkunaan.
9. Suljetaan Foxability.

2.1 Puurakenne

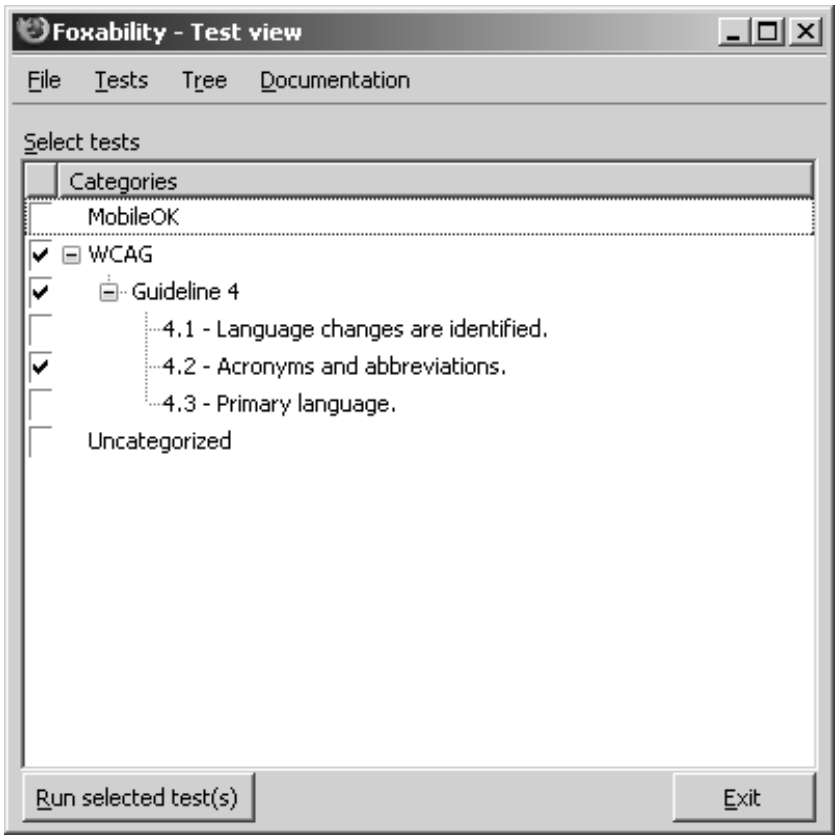
Testit näkyvät kaikissa käyttöliittymänäkymissä samanlaisessa puurakenteessa. Puun juurina ovat kategoriat, oksina moduulit ja lehtinä yksittäiset testit. Kategoria on yläkäsite, joka voi sisältää useita moduuleja. Moduuli on pienin yksikkö, jonka käyttäjä voi Foxabilityyn lisätä. Moduuli voi kuitenkin sisältää useita testejä. Esim. kuvassa 2.2 *WCAG* on kategoria, *Guideline 4* on moduuli ja *4.1 - Language changes are identified.* on yksittäinen testi.

2.2 Test view

Kuvassa 2.2 on *Test view* -ikkuna. Sen tärkeimmät toiminnot ovat testien selaaminen, testien valitseminen ja valittujen testien suorittaminen. Valitut testit suoritetaan painamalla *Run selected test(s)* -painiketta, joka avaa tulokset *Test results* -ikkunaan. *Test view* -näkymästä voi siirtyä *Manage tests* -ikkunaan valitsemalla *File → Manage tests*. Käyttöohje löytyy valinnalla *Documentation → Help topics*. Luvussa 2.1 on esitelty näkymässä käytetty puurakenne.

2.3 Manage tests

Kuvassa 2.3 on *Manage tests* -ikkuna. Sen toiminnot ovat moduulien ja kategorioiden hallinta. Moduulin lisäys tapahtuu painamalla *Add module* -painiketta, jolloin aukeaa normaali tiedostodialogi, josta valitaan lisättävän tiedoston nimi. Tiedoston valinnasta seuraa näkymän siirtyminen *Select category* -ikkunaan. Kategorian ja moduulin poisto tapahtuu aktivoimalla poistettava kategoria tai moduuli ja painamalla *Remove*-painiketta. Luvussa 8.4 käsitellään kategorian poistoon liittyviä ongelmia. Luvussa 2.1 on esitelty näkymässä käytetty puurakenne.



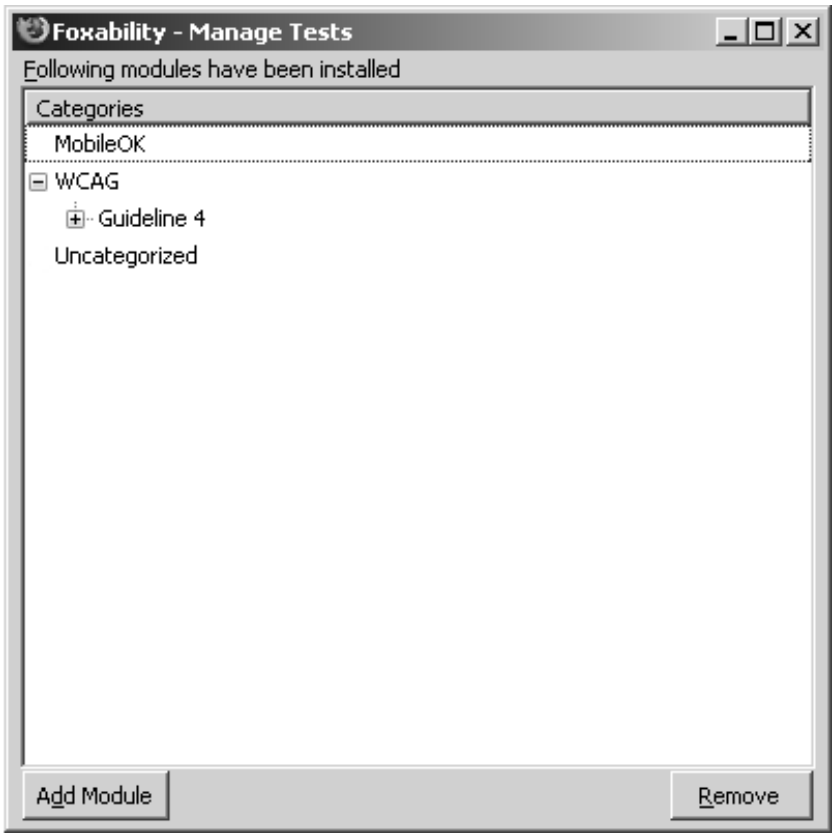
Kuva 2.2: *Test view* -ikkuna.

2.4 Select category

Kuvassa 2.4 on *Select category* -ikkuna, jossa voidaan lisätä valittu testimoduuli haluamaansa kategoriaan tai luoda uusi kategoria.

2.5 Test results

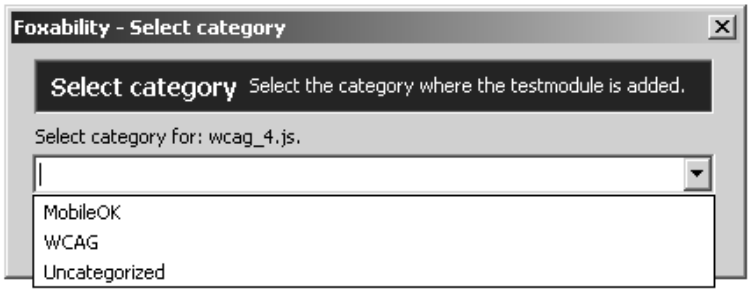
Kuvassa 2.5 on *Test results* -ikkuna, jossa näytetään esteettömyystestien tulokset. Lisäksi ikkunassa on tiedot testatun sivun osoitteesta, tilastoja testien tuloksista, testien nimet ja testien tekijöiden yhteystiedot. Testien tuloksia ja testien tekijöiden yhteystietoja voi halutessaan poistaa tai palauttaa näkyviin valinnoilla *hide/show test results* ja *hide/show module information*.



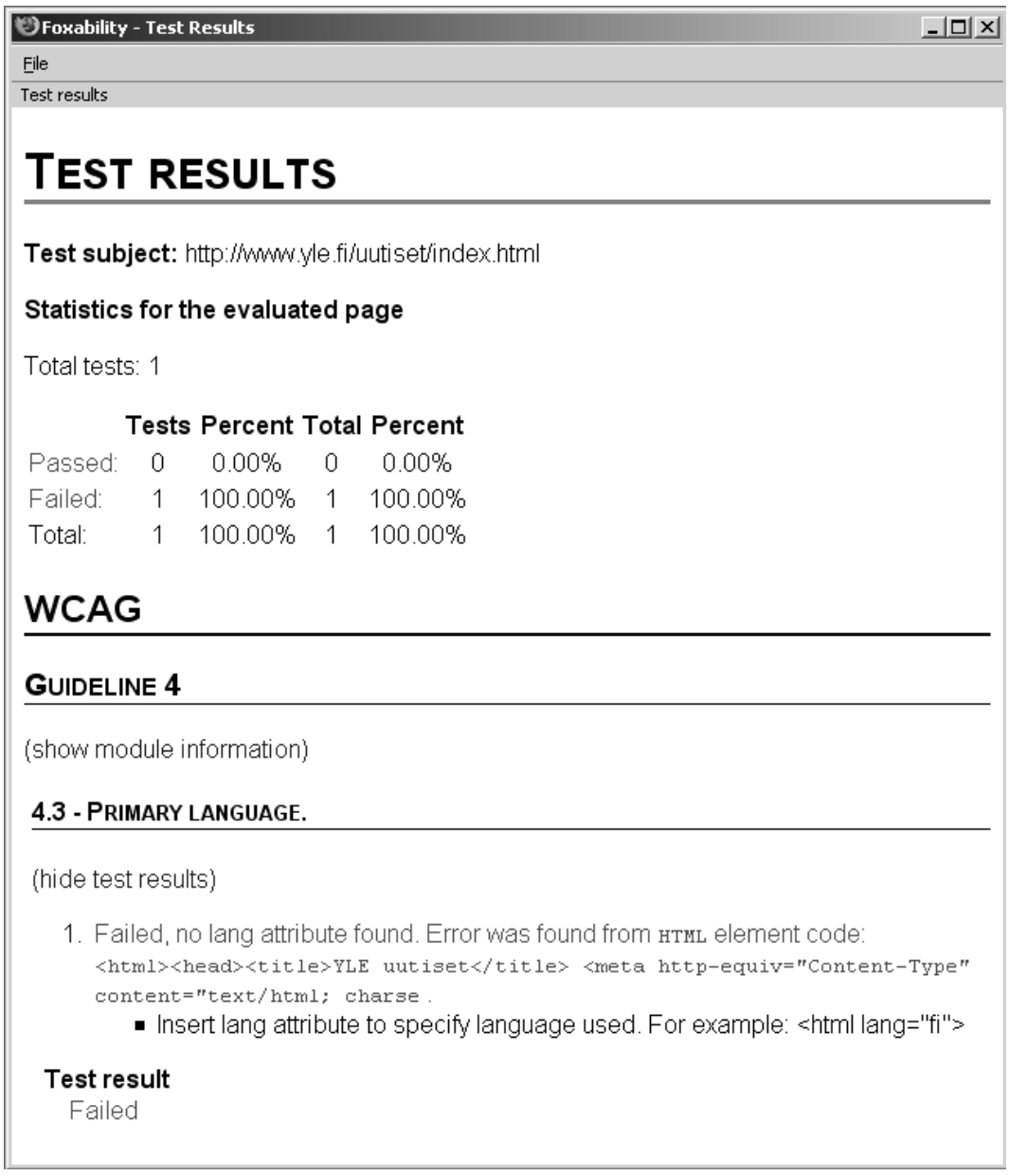
Kuva 2.3: *Manage tests* -ikkuna.

2.6 Käyttöliittymän toteutus

Käyttöliittymä on toteutettu käyttöliittymän kuvauskieli XUL:lla. XUL on XML-muotoinen, järjestelmäriippumaton merkintäkieli. XUL:ia esitellään tarkemmin luvussa 5.2.



Kuva 2.4: *Select category* -ikkuna.

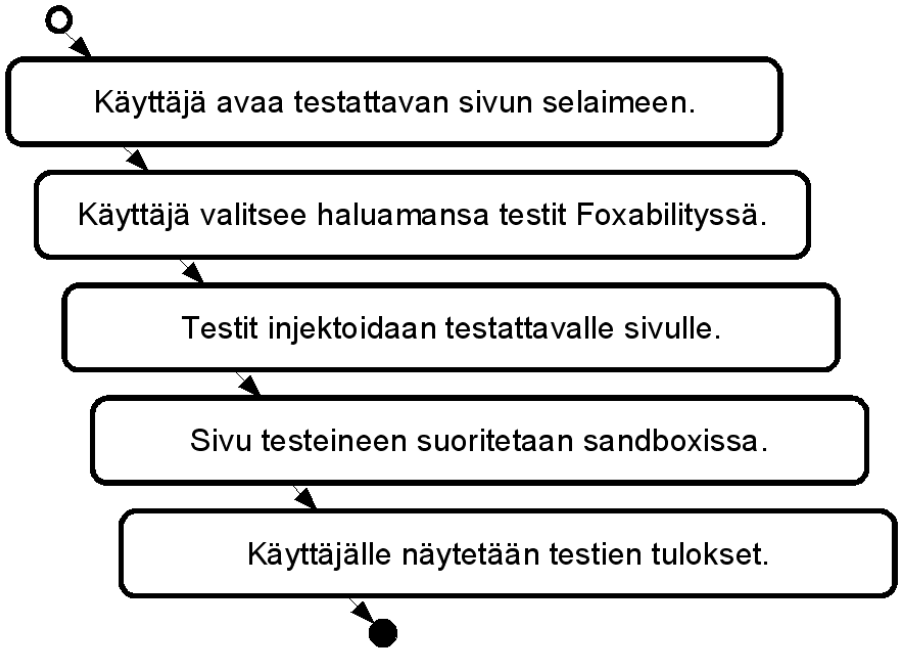


Kuva 2.5: *Test results* -ikkuna.

3 Rakenne ja toiminnallisuus

Tässä luvussa käydään läpi sovelluksen rakennetta, sekä sovelluksen osien toimintaa ja tehtäviä. Kuvassa 3.1 on esitelty Foxabilityn yleinen toimintaperiaate.

3.1 Toimintaperiaate



Kuva 3.1: Foxabilityn yleinen toimintaperiaate.

Foxabilityn tarkoitus on esteettömyystestien ajaminen verkkosivulle ja suoritettujen testien tulosten näyttäminen. Jotta tämä kaikki voitaisiin tehdä, tulee käyttäjän ensin valita testattava sivu. Valinta tapahtuu avaamalla testattava sivu selaimeen ja sen jälkeen siirtymällä Foxabilityyn. Käyttäjän tulee myös valita haluamansa testit, jotka sivulle ajetaan. Valitut testit injektoidaan (ks. luku 3.1.2) testattavaan sivuun, jonka jälkeen testattava sivu viedään sandboxiin (ks. luku 3.1.1), jossa testit suoritetaan. Lopuksi testien tulokset näytetään Foxabilityn avaamassa ikkunassa.

3.1.1 Sandbox

Sandbox on eristetty, tietoturvallinen ympäristö, jossa voidaan suorittaa ohjelmia rajoitetuin oikeuksin. Foxabilityn tapauksessa sandbox takaa käyttäjälle laajennoksen turvallisen käytön, vaikka testi olisi kirjoitettu vahingoittamistarkoituksessa. Rajoitetuista oikeuksista voi olla kuitenkin myös haittaa. Foxability ei pysty tarkistamaan sivuja, jotka ovat käyttäjän valitseman sivun domainin ulkopuolella.

3.1.2 Injektointi

Injektointi tarkoittaa tekniikkaa, jossa ohjelman ulkopuolinen merkkijono lisätään ohjelmaan. Toisin sanoen injektointi on vain hieno termi ohjelman ulkopuolisen tiedon ujuttamiseksi ohjelmaan. Usein merkkijono on lyhyt koodinpätkä ja ohjelma jokin verkkopalvelu. Injektointitekniikkaa käytetäänkin usein verkkopalvelujen haavoittuvuuksien hyödyntämiseen haittatarkoituksessa.

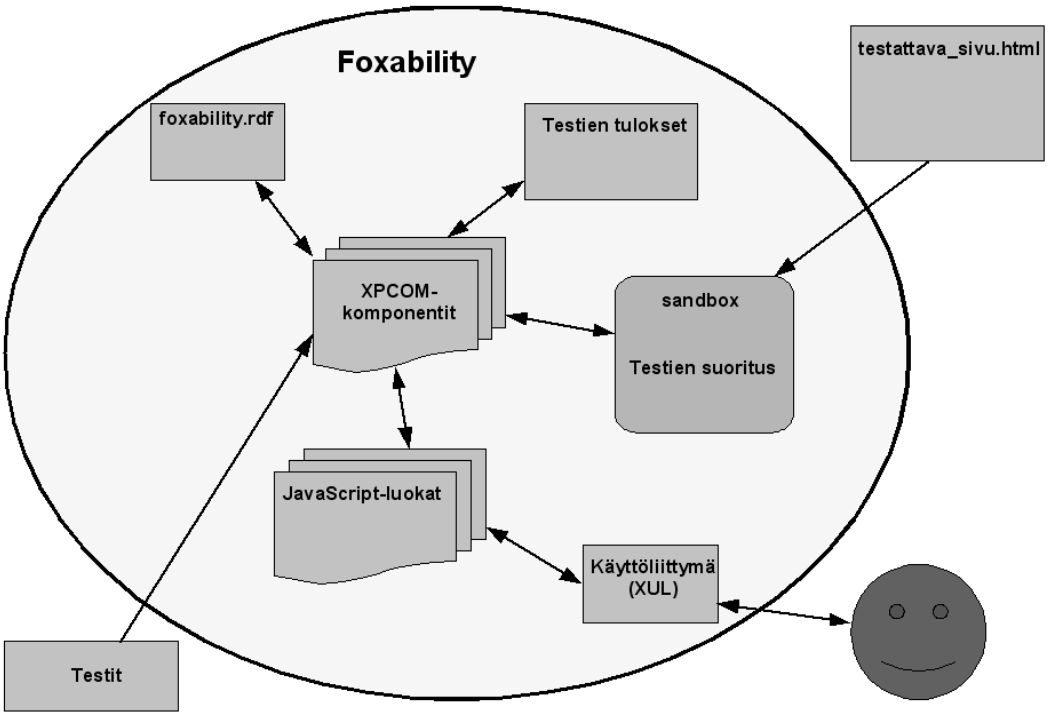
Foxabilityssä injektointitekniikkaa käytetään testien lähdekoodin siirtämiseen testattavalle sivulle.

3.2 Rakenne

Kuvassa 3.2 on esitelty Foxabilityn yleistä rakennetta. Foxabilityn käyttöliittymä on toteutettu käyttöliittymän kuvauskieli XUL:lla (ks. luku 5.2). Varsinaiset toiminnallisuudet ovat kirjoitettu JavaScriptillä.

Gecko on Firefoxin ydin, joka tarjoaa toiminnallisuuksia kuten tietovirtojen käsittelyä, muistinhallintaa jne. Geckon toiminnallisuuksiin päästään käsiksi vain Cross Platfrom Component Object Model eli XPCOM-komponenttien avulla. Tällaisia toiminnallisuuksia, joihin Foxabilityssä tarvitaan XPCOM-komponentteja ovat mm. testien tulosten käsittely, tiedoston käsittely, sekä testien suoritus. XPCOM-komponentit voisi kirjoittaa muillakin kielillä, mutta Foxabilityssä ne on toteutettu JavaScriptillä.

Foxability.rdf tiedostossa pidetään yllä tietoja käyttäjän asentamista testeistä. Itse testit ovat kuitenkin pikemminkin ulkopuolelta tuleva sisältö, kuin valmis osa Foxabilityä.



Kuva 3.2: Foxabilityn rakenne.

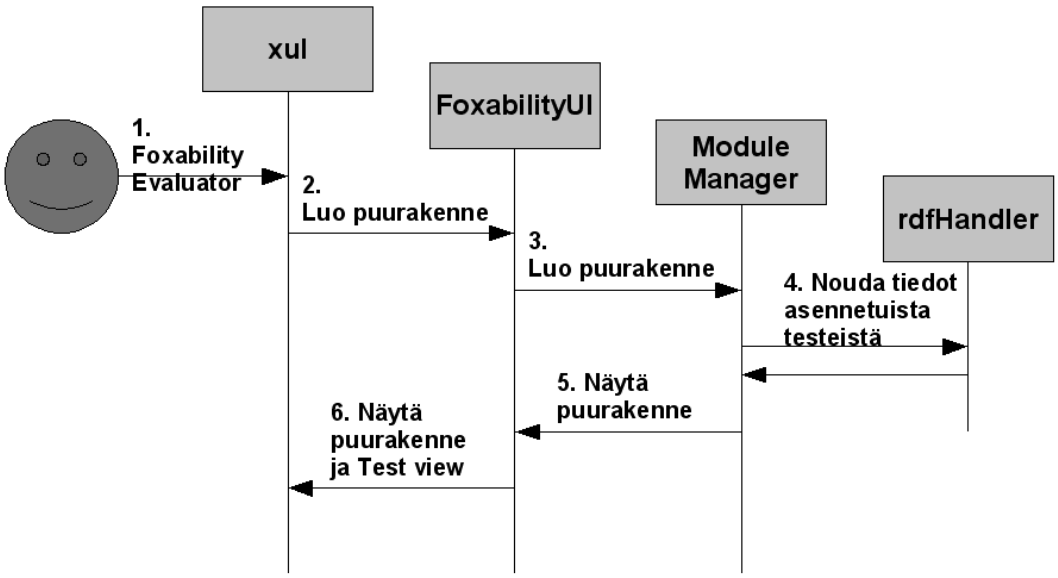
Varsinainen testien ajo tapahtuu sandboxissa, josta on kerrottu lisää luvussa 3.1.1.

4 Luokat

Tässä luvussa kerrotaan Foxabilityn luokista ja niiden keskinäisistä suhteista. Foxabilityn luokkarakenne ja luokat ovat perintöä Foxability-projektista ja Kettu-projektin yhteydessä niiden keskinäisiin suhteisiin ei ole puututtu. Luokkien sisältöjä on kuitenkin muokattu, korjailtu ja laajennettu.

Luvuissa 4.1 - 4.3 esittellään sovelluksen sisäistä rakennetta pääkäyttötapausten avulla. Luvuissa 4.4 - 4.16 on esitelty lyhyet kuvaukset sovelluksen luokista.

4.1 Sovelluksen käynnistyminen



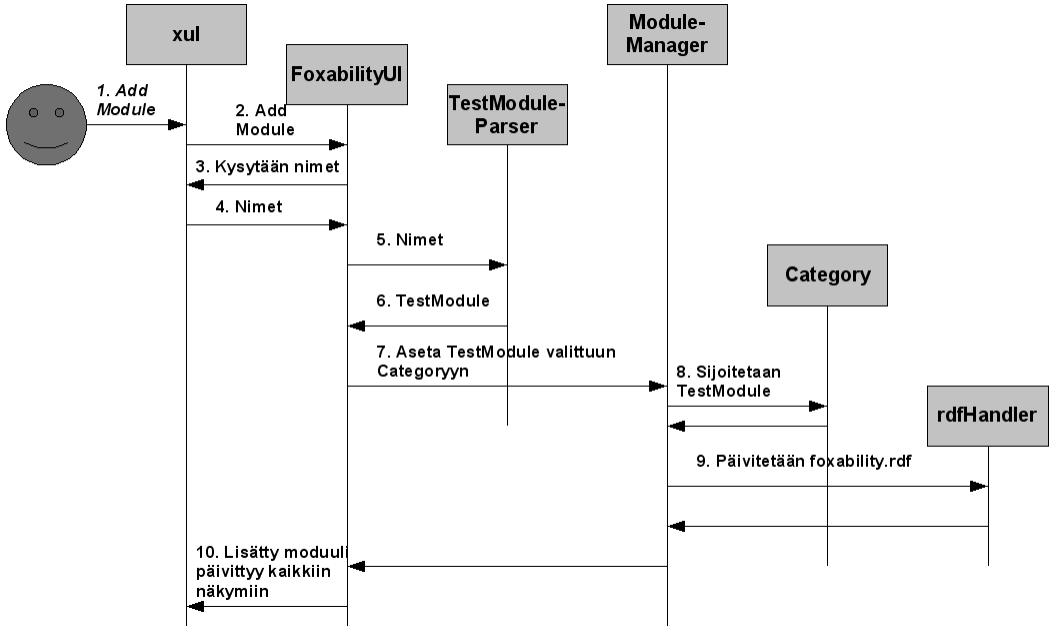
Kuva 4.1: Sekvenssikaavio Foxabilityn käynnistymisestä.

Kuvassa 4.1 on esitetty seuraava tapahtumaketju sekvenssikaavion muodossa.

1. Käyttäjä valitsee *Foxabilit Evaluator* Firefoxin Tools-valikosta.
2. faOverlay.xul-tiedoston oncommand-tapahtumankäsittelijään määritelty aliohjelma suoritetaan. Aliohjelman tehtävä on luoda faMain.xul-tiedoston mukainen ikkuna, jota varten tarvitaan tiedot näytettävästä puurakenteesta.
3. FoxabilityUI luo ModuleManagerin ja pyytää sitä luomaan puurakenteen.

4. ModuleManager luo foxability.rdf-tiedostoon perustuvan puurakenteen
5. Puurakenne palautetaan FoxabilityUI:lle, joka päivittää sen faMain.xulille, jolloin puurakenne tulee näkyville.

4.2 Testin lisääminen



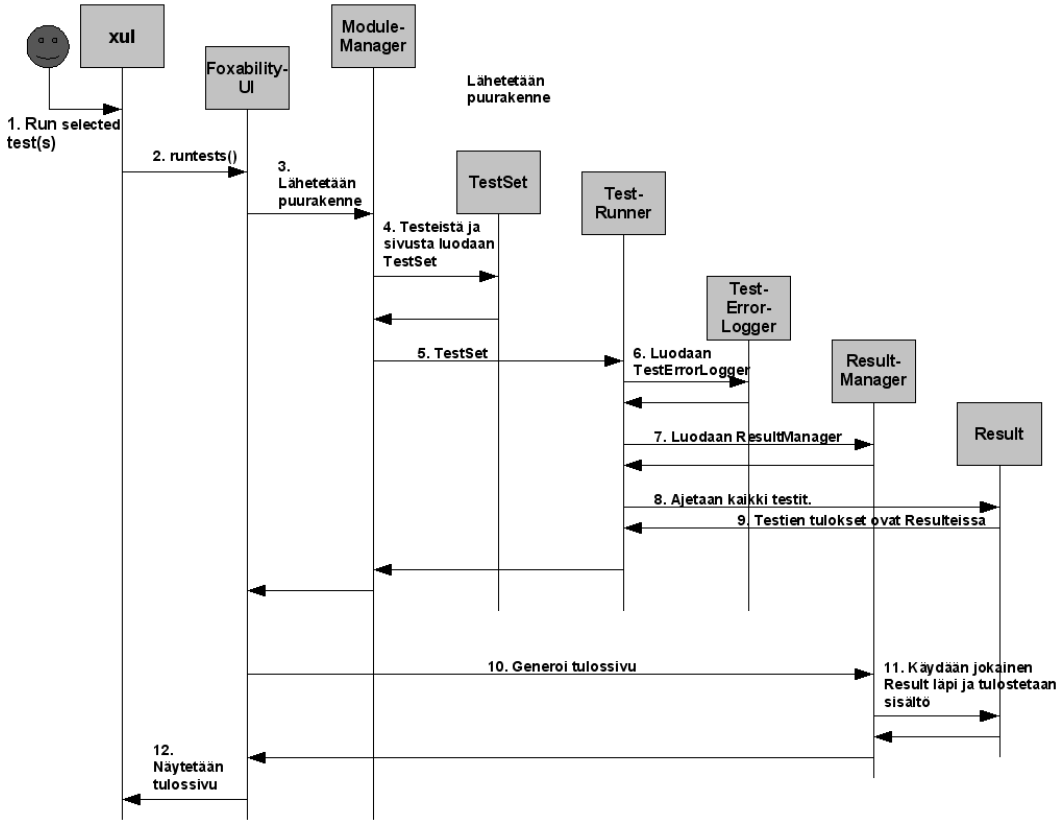
Kuva 4.2: Sekvenssikaavio testin lisäämisestä Foxabilityyn.

Kuvassa 4.2 on esitetty seuraava tapahtumaketju sekvenssikaavion muodossa.

1. Käyttäjä valitsee Foxabilityn *Manage Tests* -näkymässä *Add Module*.
2. faManage.xul-tiedoston oncommand-tapahtumankäsittelijään määritelty aliohjelma suoritetaan.
3. Käyttäjä valitsee lisättävän testimoduulin tiedosto-dialogista, jonka jälkeen käyttäjä syöttää sen kategorian nimen, johon testimoduuli lisätään.
4. Syötetyt nimet palautuvat FoxabilityUI:lle.
5. FoxabilityUI luo TestModuleParserin ja käskeen sen luoda TestModulen annetun testimoduulin nimen perusteella.

6. Luotu TestModule palautuu FoxabilityUI:lle.
7. FoxabilityUI käskee ModuleManagerin asettamaan luodun TestModulen käyttäjän valitsemaan kategoriaan.
8. ModuleManager hakee oikean kategorian ja sijoittaa TestModulen sinne.
9. ModuleManager päivittää foxability.rdf-tiedostoa.
10. Lisätty testimoduuli näkyy nyt kaikissa näkymissä.

4.3 Testin ajaminen



Kuva 4.3: Sekvenssikaavio testin ajamisesta.

Kuvassa 4.3 on esitetty seuraava tapahtumaketju sekvenssikaavion muodossa.

1. Käyttäjä valitsee *Run selected test(s)*.

2. faMain.xul-tiedoston oncommand-tapahtumankäsittelijään määritelty aliohjelma suoritetaan.
3. FoxabilityUI lähettää puurakenteen käyttäjän tekemine valintoineen ModuleManagerille.
4. ModuleManager luo TestSetin valituista testeistä ja testattavasta sivusta.
5. ModuleManager lähettää TestSetin suoritettavaksi TestRunnerille.
6. TestRunner luo TestErrorLoggerin, johon kirjataan testien ajossa mahdollisesti tapahtuvia virheitä. Tässä virheellä ei tarkoiteta testattavassa sivussa olevia virheitä, vaan esim. testissä tapahtuvia poikkeuksia.
7. TestRunner luo ResultManagerin.
8. TestRunner testaa sivun sandboxissa TestModule kerrallaan.
9. Jokaisen testin ajossa luodaan vähintään yksi Result ja ResultManager tallentaa ne.
10. FoxabilityUI käskee ResultManagerin generoida tulossivu.
11. ResultManager tulostaa jokaisen Resultin tulokset tulossivulle.
12. Tulossivu näytetään käyttäjälle.

4.4 FoxabilityUI

FoxabilityUI on Foxabilityn käyttöliittymäluokka. Perinteisten käyttöliittymään liittyvien tehtävien lisäksi se hoitaa myös osan muiden luokkien välisestä kommunikaatiosta. FoxabilityUI:n aliohjelmia kutsutaan XUL:lla kuvatusta ikkunoista osoittamalla painikkeiden yms. tapahtumankäsittelijöille kutsuttavan aliohjelman nimi.

4.5 FA_TestRunner

Tämän prototyypin tärkeimmät tehtävät ovat injektoida testit testattavalle sivulle, luoda sandbox (ks. luku 3.1.1) ja suorittaa sivu siinä. Testien suorituksessa mahdollisesti ilmentyvät virheet käsitellään FA_TestErrorLogger-prototyypissä.

4.6 FA_TestErrorLogger

Tämän prototyypin ainoa tehtävä on raportoida testien ajossa esiintyneet virheet virhekonsoliin. Virheellä tässä yhteydessä ei tarkoiteta testattavasta sivusta löytyvää esteellisyyttä vaan esim. poikkeuksen lentämistä ajettavassa testissä.

4.7 FA_Category

FA_Categoryn tehtävä on hallinnoida moduuleja. Tämä prototyyppi on puurakenteessa esiintyvä kategoria. Puurakenne on esitelty luvussa 2.1.

4.8 FA_TestModule

Uudet testimoduulit toteutetaan perimällä tämä abstrakti prototyyppi. Tämän prototyypin tehtävä on hallinnoida testejä ja säilyttää tiedot testimoduulin tekijöistä. FA_TestModule on puurakenteen moduuli. Puurakenne on esitelty luvussa 2.1

4.9 FA_Test

FA_Test-prototyyppi edustaa yksittäistä testiä. Se tietää oman nimensä ja funktioviitteen varsinaiseen testifunktioon. Kun testi on ajettu, tallentaa prototyyppi FA_Result-muotoisen testin tuloksen itseensä. FA_Test on puurakenteen testi. Puurakenne on esitelty luvussa 2.1.

4.10 FA_rdfHandler

FA_rdfHandler kirjoittaa ja lukee foxability.rdf tiedostoa, jossa ylläpidetään tietoja asennetuista kategorioista, moduuleista ja testeistä.

4.11 FA_Result

Tämän prototyypin tehtävä on tallettaa tietoa testin ajon tuloksesta ja tulos voi olla joko passed, warning, failed tai remark. Lisäksi prototyyppissä on virhe- ja korjausehdotusmerkkijonot. Testin ajon tuloksen ollessa passed kyseiset merkkijonot ovat tyhjiä.

- passed: testattu sivu läpäisi testin moitteetta.
- warning: testattu sivu ei ole testin kannalta moitteeton, mutta sitä ei voi suoraan hylätäkkään.
- failed: testattu sivu ei läpäissyt testiä.
- remark: testatussa sivussa on jokin huomioon otettava asia. Tämä tulos annetaan usein kun testiä ei pysty/osata koneellisesti tarkistaa.

4.12 FA_TestModuleParser

FA_TestModuleParser lukee testitiedostoa ja luo sen perusteella FA_TestModulen.

4.13 FA_TestSet

FA_TestSet-prototyypin tehtävä on valita kaikkien testien joukosta käyttäjän valitsemat testit ajettavaksi. Tämä prototyyppi saa käyttöönsä käyttäjän puurakenteeseen tekemät valinnat ja luo sen tiedon perusteella ajettavien testien joukon.

4.14 FA_ModuleManager

Tämä on yksi Foxabilityn keskeisiä prototyyppejä. Se tietää mitä prototyyppejä kutsua, kun halutaan jokin tehtävä tehdyksi. Kyseisiä tehtäviä ovat foxability.rdf:n hallinta, moduulien hallinta, testien ajaminen, testimoduulien luominen testitiedostoista, sekä testijoukon luominen.

4.15 FA_ResultManager

FA_ResultManager hoitaa testien tuloksien keräämisen, testitilastojen laskemisen, tulossivun generoimisen ja tulossivun näyttämisen käyttäjälle.

4.16 Testimoduuli

Testimoduuli tulee periä FA_TestModule-prototyypistä. Testattavaan sivuun päästään käsiksi DOM-rajapinnan avulla. Tutkittavat elementit valikoidaan usein käyttäen XPath-kieltä. Valikoiduille elementeille tehdään sitten testistä riippuvaisia jatkotutkimuksia. Jokaista tutkittavalta verkkosivulta löytyvää testin kannalta virheellistä tietoa kohtaan luodaan uusi FA_Result-prototyyppi, jossa virheen tiedot siirtyvät eteenpäin. Jos testi ei löydä yhtään virhettä, luodaan lopuksi FA_Result-prototyyppi, jonka tilaksi asetetaan *passed*.

5 Tiedostot ja tietorakenteet

Tässä luvussa kuvataan ohjelman käyttämien tiedostojen rakenteita ja tarkoituksia.

5.1 foxability.rdf

Foxability mahdollistaa testien lisäämisen ja poiston. Jotta Foxability muistaisi asennetut kategoriat ja moduulit käynnistyskertojen välissä, pitää tieto niistä olla tallessa jossakin. Tieto asennetuista testeistä ylläpidetään foxability.rdf tiedostossa. RDF-tiedosto on XML-muotoinen metatiedontallennusstandardi. Kuvassa 5.1 on foxability.rdf-tiedosto yhden testimoduulin lisäämisen jälkeen. Siinä näkyy miten kuvassa 5.4 esiintyvä puurakenne on tallennettu. Seuraavissa kolmessa kappaleessa on käyty läpi kuvan 5.1 pääkohdat.

Ensimmäiset neljä riviä ovat yleisiä XML ja RDF-määrityksiä, jotka ovat kaikkien RDF-tiedostojen alussa. Foxabilityn kannalta oleellinen osa alkaa riviltä 31. Sekvenssi allCategories sisältää kolme valmista kategoriaa testimoduuleille. Aina kun käyttäjä lisää uuden kategorian, tulee se tämän sekvenssin alle listaelementtinä. Jokaista listan elementtiä kohden on yksi Description-elementti. Esimerkiksi WCAG:n Description-elementti löytyy riviltä 39. Se kertoo lähinnä, että WCAG:n kategorianimi on WCAG. Tämä siis tulostuu käyttäjälle sovelluksen käyttöliittymässä.

WCAG:lla on myös oma sekvenssi, joka sisältää sen testit. Tässä tapauksessa sen sisältä löytyy lisäämämme wcag_4.js-testimoduuli. Kyseisellä testimoduulilla on taas vastaavasti oma Description-elementti, joka kertoo yleisiä tietoja modulista, sekä polun tiedostoon, tämä löytyy riviltä 20.

Testimodulilla on myös oma sekvenssi rivillä 15, joka sisältää suoritettavien aliohjelmien nimet. Jokaista aliohjelmaa kohtaan on oma Description-elementti, jossa sijaitsee näyttöön tulostettava merkkijono, sekä kutsuttavan aliohjelman nimi. Tällaiset Description-elementit sijaitsevat riveillä 5, 12 ja 41.

5.2 XUL

XUL on XML-muotoinen järjestelmäriippumaton käyttöliittymän kuvauskieli. Kuvauskielellä muotoillaan käyttöliittymän ulkoasu eli käyttöliittymäkomponenttien

tyypit ja sijainnit. Lisäksi kuvauskielellä osoitetaan käyttöliittymäkomponenttien tapahtumankäsittelijöille kutsuttavat aliohjelmat. Kuvauskieltä on käytetty Foxabilityn kaikkien seitsemän eri näkymän luomiseen. Kuvassa 5.2 on tiedosto faCategory.xul kokonaisuudessaan. Kyseisen tiedoston avulla on kuvattu *Select category* -ikkunan käyttöliittymäkomponentit ja määritelty käyttöliittymäkomponenttien tapahtumankäsittelijöille aliohjelmat.

Kuvassa 5.2 on faCategory.xul-tiedosto, jonka avulla kuvan 5.3 ikkuna on muodostettu. Seuraavassa listassa on käyty kuvan 5.2 pääkohdat läpi.

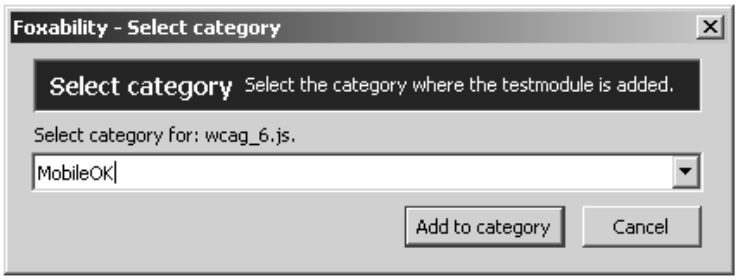
- Rivillä 3 määritellään tiedoston nimi, jossa käytetyt muuttujat ovat määritelty.
- Riveillä 5 ja 6 asetetaan ikkunan id ja title-attribuuteille arvot .
- Rivillä 7 esitellään piirrettävät painikkeet.
- Riveillä 8 ja 9 asetetaan esitellyille painikkeille nimet.
- Riveillä 10-12 asetetaan ikkunan tapahtumankäsittelijöille kutsuttavat aliohjelmat.
- Rivillä 15 lisätään tähän kohtaan sivua fa.js-tiedoston sisältö.
- Riveillä 17-19 luodaan dialogheader-elementti ja annetaan sen attribuuteille arvoja.
- Rivillä 21 luodaan laatikko, johon asetetaan myöhemmin luotava menulista.
- Rivillä 24 luodaan itse menulista, johon kategoriat tulostetaan.
- Rivillä 27 datasources-attribuutti on asetettu nulliksi, sillä se asetetaan fa.js-tiedoston aliohjelmassa.
- Rivillä 28 annetaan viite foxability.rdf-tiedostossa olevaan listaan, josta haetaan menulistan sisältö.
- Rivillä 35 minkä attribuutin arvo haetaan listaelementistä.

```
01<?xml version="1.0"?>
02<RDF:RDF xmlns:NS1="http://foxability.sourceforge.net/"
03         xmlns:NC="http://home.netscape.com/NC-rdf#"
04         xmlns:RDF="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
05  <RDF:Description RDF:about="http://foxability.sourceforge.net/WCAG/C:\MyTemp\kettu\
                              trunk\tests\wcag_4.js/language"
06                   NS1:testName="4.3 - Primary language."
07                   NS1:functionName="language" />
08  <RDF:Seq RDF:about="http://foxability.sourceforge.net/MobileOK">
09  </RDF:Seq>
10  <RDF:Description RDF:about="http://foxability.sourceforge.net/MobileOK"
11                   NS1:categoryName="MobileOK" />
12  <RDF:Description RDF:about="http://foxability.sourceforge.net/WCAG/C:\MyTemp\kettu\
                              trunk\tests\wcag_4.js/accAbbTest"
13                   NS1:testName="4.2 - Acronyms and abbreviations."
14                   NS1:functionName="accAbbTest" />
15  <RDF:Seq RDF:about="http://foxability.sourceforge.net/WCAG/C:\MyTemp\kettu\
                        trunk\tests\wcag_4.js">
16    <RDF:li RDF:resource="http://foxability.sourceforge.net/WCAG/C:\MyTemp\kettu\
                           trunk\tests\wcag_4.js/languageChange"/>
17    <RDF:li RDF:resource="http://foxability.sourceforge.net/WCAG/C:\MyTemp\kettu\
                           trunk\tests\wcag_4.js/accAbbTest"/>
18    <RDF:li RDF:resource="http://foxability.sourceforge.net/WCAG/C:\MyTemp\kettu\
          trunk\tests\wcag_4.js/language"/>
19  </RDF:Seq>
20  <RDF:Description RDF:about="http://foxability.sourceforge.net/WCAG/C:\MyTemp\kettu\
                              trunk\tests\wcag_4.js"
21                   NS1:moduleName="Guideline 4 "
22                   NS1:author="Kosti"
23                   NS1:email="koalkuok@jyu.fi"
24                   NS1:website="http://sovellusprojektit.it.jyu.fi/kettu/"
25                   NS1:filePath="C:\MyTemp\kettu\trunk\tests\wcag_4.js"
26                   NS1:className="WCAGFour" />
27  <RDF:Seq RDF:about="http://foxability.sourceforge.net/Uncategorized">
28  </RDF:Seq>
29  <RDF:Description RDF:about="http://foxability.sourceforge.net/Uncategorized"
30                   NS1:categoryName="Uncategorized" />
31  <RDF:Seq RDF:about="http://foxability.sourceforge.net/allCategories">
32    <RDF:li RDF:resource="http://foxability.sourceforge.net/MobileOK"/>
33    <RDF:li RDF:resource="http://foxability.sourceforge.net/WCAG"/>
34    <RDF:li RDF:resource="http://foxability.sourceforge.net/Uncategorized"/>
35  </RDF:Seq>
36  <RDF:Seq RDF:about="http://foxability.sourceforge.net/WCAG">
37    <RDF:li RDF:resource="http://foxability.sourceforge.net/WCAG/C:\MyTemp\kettu\
                           trunk\tests\wcag_4.js"/>
38  </RDF:Seq>
39  <RDF:Description RDF:about="http://foxability.sourceforge.net/WCAG"
40                   NS1:categoryName="WCAG" />
41  <RDF:Description RDF:about="http://foxability.sourceforge.net/WCAG/C:\MyTemp\kettu\
                              trunk\tests\wcag_4.js/languageChange"
42                   NS1:testName="4.1 - Language changes are identified."
43                   NS1:functionName="languageChange" />
44</RDF:RDF>
```

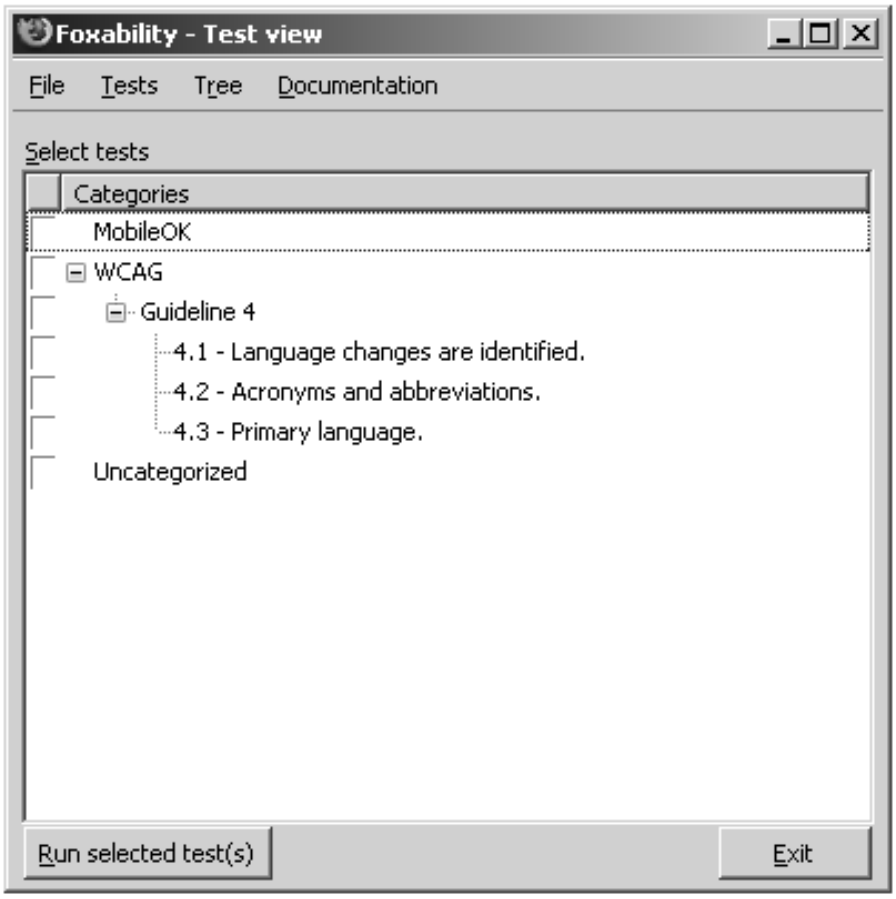
Kuva 5.1: Foxability.rdf.

```
01<?xml version="1.0"?>
02<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>
03<!DOCTYPE dialog SYSTEM "chrome://foxability/locale/foxability.dtd">
04<dialog
05  id="faCategory"
06  title="&category.title;"
07  buttons="accept, cancel"
08  buttonlabelaccept="&category.addCategory;"
09  buttonlabelcancel="&category.cancel;"
10  ondialogaccept="return FoxabilityUI.parseCategoryValue(event);"
11  ondialogcancel="return FoxabilityUI.cancel(event);"
12  onload="FoxabilityUI.faCategory_onLoad();"
13  xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
14
15  <script type="application/x-javascript" src="chrome://foxability/content/fa.js"/>
16
17  <dialogheader
18    title="&category.selectCategory;"
19    description="&category.selectDescription;"/>
20
21  <vbox flex="1">
22    <label id="categoryMenuLabel"/>
23
24    <menulist
25      id="category"
26      editable="true"
27      datasources="rdf:null"
28      ref="http://foxability.sourceforge.net/allCategories">
29      <template>
30        <rule >
31          <conditions>
32            <content uri="?list"/>
33            <member container="?list" child="?category"/>
34            <triple subject="?category"
35                predicate="http://foxability.sourceforge.net/categoryName"
36                object="?name"/>
37          </conditions>
38          <action>
39            <menupopup>
40              <menuitem uri="?category" label="?name" />
41            </menupopup>
42          </action>
43        </rule>
44      </template>
45    </menulist>
46  </vbox>
47</dialog>
```

Kuva 5.2: faCategory.xul-tiedosto.



Kuva 5.3: *Select category -ikkuna*.



Kuva 5.4: *Test view* -ikkuna.

6 Testaus

Testauksen tarkoituksena on varmistaa, että sovellus täyttää sille asetetut toiminnalliset ja laadulliset vaatimukset. Seuraavaksi käydään lyhyesti kuinka testaus käytännössä toteutui.

6.1 Toiminnallisuuksien testaus

Ohjelmointitehtävät Kettu-sovellusprojektissa jakautuvat selvästi kahtia: esteettömyystestien toteuttamiseen ja käyttöliittymän parantamiseen sekä bugien korjaukseen.

Esteettömyystestien kohdalla testin toteuttaja kirjoitti itse testisivuja, joita vastaan esteettömyystestien toimivuutta testattiin. Testisivut pohjautuivat tilaajan kanssa tehtyihin tulkintoihin WCAG 1.0 -suosituksesta. Testisivut ovat saatavilla Kettu-sovellusprojektin cd:ltä. Käyttöliittymään liittyvien ohjelmointitöiden testaus tapahtui kehitettyjen toiminnallisuuksien monipuolisella käyttämisellä.

Kettu-projektin loppuun on suunniteltu Jyväskylän yliopiston Plone-kehittäjille suunnattu koulutustilaisuus. Koulutustilaisuudesta saadaan myös Kettu-ryhmän ulkopuolisia käyttökokemuksia.

6.2 Hyväksyntätestaus

Hyväksyntätestausta tapahtui sekä iteraatioiden välisissä palavereissa että vapaamuotoisemmin projektiryhmän työhuoneessa. Hyväksyntätestaus oli käytännössä sitä, että esteettömyystestin tai käyttöliittymän toimintaa esiteltiin tilaajalle ja tilaaja antoi siitä palautetta. Lopuksi sovelluksen osa joko hyväksyttiin, hyväksyttiin muutoksin, hyväksyttiin kieliasua lukuunottamatta tai ei hyväksytty ollenkaan. Jos hyväksyntätestaus tapahtui vapaamuotoisessa tilaisuudessa projektiryhmän työhuoneessa, merkittiin mahdollinen hyväksyminen seuraavan palaverin pöytäkirjaan.

7 Ohjelmointikäytänteet ja toteutusympäristö

Tässä luvussa kerrotaan Kettu-sovellusprojektissa käytetyistä ohjelmointikäytänteistä, toteusympäristöstä sekä työkaluista.

7.1 Lähdekoodin ulkoasu

```
/**
 * CommentModel
 * Comment Model class doesn't do anything. [jargon]
 *
 * @author kettu http://sovellusprojektit.it.jyu.fi/kettu/
 * @author mina <mina@jyu.fi>
 *
 * @version x.x
 * @extends FA_TestModule
 *
 */

/**
 * Comment model
 * @constructor
 */
function CommentModel() {
  this.name = FA_TestModule.name;
}

CommentModel.prototype = new FA_TestModule;


  getTestByName: function(testName) {
    return this.getTestIndexByName(testName);
  }
};
```

Kuva 7.1: Esimerkki lähdekoodin muotoilukäytänteestä.

Lähdekoodin muotoilussa ja kommentoinnissa jatkettiin samoilla linjoilla, mitä Foxability-projektissa oli käytetty. Eclipse-ohjelmointiympäristö aiheutti hieman ongelmia koodin yhtenäisen muotoilun kanssa. Luokkien ja metodien kommentointi on tehty yllä olevien JavaScriptDoc-käytänteiden mukaan. Lähdekoodi kirjoitettiin englanniksi.

Aliohjelmien ja muuttujien nimet on kirjoitettu pienellä alkukirjaimella siten, että seuraavat sanat ovat kiinni edellisessä ja aloitettu isolla kirjaimella. Tiedostoissa,

jotka ovat peräisin jo Foxability-projektista, ollaan jatkettu Foxability-projektin koodauskäytänteitä.

7.2 Toteutusympäristö

Foxability-sovelluksen jatkokehitys oli luonnollista tehdä samalla kielellä kuin Foxabilityä oli siihen asti tehty, eli JavaScriptillä. Toteutus tehtiin Eclipse 3.2 -ohjelmointiympäristössä. Eclipsessä käytettiin lisäksi kahta laajennosta: Subclipse ja JSEclipse. Subclipse mahdollisti versiohallinnan käytön suoraan Eclipsen kautta ja JSEclipse sai ohjelmointiympäristön ymmärtämään JavaScriptiä.

Firefoxissa käytimme myös kahta laajennosta: Extension Developer Extension ja Web Developer Extension. Extension Developer Extensionilla saa nopeasti työn alla olevan laajennoksen testikäyttöön ja Web Developer Extension on monitoimityökalu verkkosivujen kehityksen tueksi.

7.3 Foxabilityn kehitysversion käyttöönotto

Tässä luvussa neuvotaan, miten Foxabilityn kehitysversion saa asennettua käyttöön.

1. Hae uusin versio Foxabilitystä SourceForgesta osoitteesta:
   `http://foxability.sourceforge.net/`
2. On suositeltavaa luoda oma profiili Firefoxiin, jolle Foxability ym asennetaan. Luo sellainen käynnistämällä Firefox parametrin -p avulla. Start -> Run -> (esim. `C:\polkuFirefoxiin\firefox -p`)
3. Hae Extension Developer Extension osoitteesta:
   `http://ted.mielczarek.org/code/mozilla/extensiondev/`
4. Käynnistä Extension Developer Extension ja valitse työhakemistoksi Foxabilityn extension-kansio.
5. Paina *Install for development* Extension Developer Extensionissa. Käynnistä Firefox uudestaan. Nyt Firefoxin *Tools*-valikkoon on ilmestynyt *Foxability Evaluator* -valinta ja sen valitsemalla Foxability käynnistyy.

8 Jatkokehitysideat

Luvussa käsitellään laajennoksen kehittämisideoita, joita tämän projektin puitteissa ei pystytty toteuttamaan. Lisäksi pohditaan, mitä asioita olisi kannattanut jälkikäteen ajateltuna tehdä toisin.

8.1 Käyttöliittymä

8.1.1 Rivinumerointi

Ehkä tärkein yksittäinen ominaisuus, joka Foxabilityyn tulisi kehittää, on esteettömyystestin löytämän virheen sijainnin rivinumeron näyttäminen. Rivinumeron näyttäminen helpottaisi virheellisten kohtien löytämistä lähdekoodista. Rivinumeroinnin toteuttamisen eteen tehtiin tutkimustyötä, mutta resursseja sen toteuttamiseen ei Kettu-sovellusprojektilla ollut. Rivinumeroinnin implementointi jälkikäteen on hyvin työlästä ja sen helppo toteutus olisi vaatinut asian huomioonottamista jo Foxability-projektin suunnitteluvaiheessa. Lisäksi rivinumerointiin liittyy kohdassa 8.3.1 esitelty ongelma.

8.1.2 Esteettömyys

Esteetön esteettömyystarkistin on periaatteellinen vaatimus, jonka toteuttamiseen Kettu-sovellusprojektin yhteydessä ei osoitettu resursseja. Seuraava kysymys selventää parhaiten miksi laajennoksen tulisi olla esteetön: Onko asiallista huomautella muiden esteellisyydestä, jos itse huomautteleva ohjelmakin on esteellinen?

8.1.3 Lokalisointi

Yksi Kettu-sovellusprojektin päämääristä oli saada laajennos helposti lokalisoitavaksi. Lokalisoida tulisi niin käyttöliittymä kuin myös testit. Sitä silmällä pitäen testeissä merkkijonot ovat kirjoitettu kootusti testitiedostojen alkuun. Käyttöliittymän lokalisointi on pääosin vain kääntämistyötä. Käyttöliittymässä poikkeuksen tekee tulosikkunat, joissa työn määrä voi olla suurempi. Arvioimme lokalisoinnin mahdollistamiseen kuluvan n. 60 h, jonka lisäksi n. 20 h per lokalisoitava kieli.

8.2 Toiminnallisuudet

8.2.1 Lisää testejä

Yksi Kettu-projektin alkuperäinen tavoite oli tehdä MobileOK-testit, mutta niiden toteuttamiseen ei riittänyt resursseja. Lisäksi WCAG 2.0 suositus on ilmestymässä piakkoin.

8.2.2 EARL

Evaluation and Report Language eli EARL on testien tuloksien kuvaamiseen tehty kieli. Jo Foxability-projektin ajoilta juontava tavoite on saada Foxability tallentamaan testien tulokset EARL-muotoon.

8.3 Testit

8.3.1 Firefox muokkaa lähdekoodia

Firefox on niin kettu selain, että verkkosivut näytetään muokattuun lähdekoodiin perustuen. Lähdekoodin muokkaus voi olla harmitonta lopetuselementtien lisäämistä tai harmillista elementtien keskinäisten paikkojen vaihtelua. Lähdekoodia muokataan ilmeisesti sen takia, että DOM-puun rakentaminen vaatii eheän lähdekoodin.

Kun Foxabilityllä tarkistetaan verkkosivuja saa Foxability käyttöönsä tämän muokatun lähdekoodin perusteella rakennetun DOM-puun. Kun testien raportit perustuvat muokattuun lähdekoodiin, voi virheilmoituksissa näkyä lähdekoodin pätkiä, joita alkuperäisessä lähdekoodissa ei ole.

Kohtaan 8.1.1 liittyen, muokattuun lähdekoodiin perustuva rivinumerointi antaisi rivinumerot mahdollisesti väärin.

Jatkokehitysideana testit olisi ehkä syytä perustua alkuperäiseen lähdekoodiin.

8.3.2 Sandboxin asettamat rajoitukset domainin ulkopuolelle menemisestä

Sandboxin rajoituksista johtuen, Foxability ei pysty käsittelemään tiedostoja, jotka sijaitsevat testattavan sivun domainin ulkopuolella. Koska domainin ulkopuoliset tiedostot (esim. CSS) eivät ole harvinaisia, olisi kätevää, jos niitäkin pystyttäisiin käsittelemään.

8.4 Tiedossa olevat bugit

8.4.1 RDF-käsittely

FA_rdfHandler kirjoittaa .rdf-tiedostoa, miten sattuu ja siitä johtuen puurakenne voi joskus näyttää oudolta.

8.4.2 Puurakenne ja testin poisto

Testin poistaminen jättää testin puurakenteen näkyville. Testien tekstejä ei kuitenkaan näy. Erittäin harvinainen, vaikea toisintaa.

8.4.3 Pääikkunan sulkeminen

Jos on lisätty uusi testi laajennokseen ja suljetaan pääikkuna, jäävät muut ikkunat silti auki.

8.4.4 XHTML-tiedoston ylimääräiset merkinnät

Kun tulosraportti tallennetaan XHTML-tiedostoon, siihen ilmestyy ylimääräisiä elementtejä. Ylimääräisiä elementtejä ovat ainakin `<script>`, `<link>` ja `<meta>`.

8.5 Toteutusratkaisujen analyysi eli mitä tekisimme toisin

8.5.1 Toistuvien funktioiden nosto ylempään luokkaan

Useassa testissä käytetään samoja aliohjelmia, joten sellaiset aliohjelmat olisi hyvä nostaa FA_TestModuleen, josta testit peritään. Tällaisia aliohjelmia ovat niin testitulosten raportointiin liittyvät kuin myös useassa testissä tarvittavat yleiset aliohjelmat.

8.5.2 Yleiskäyttöiset funktiot

Joidenkin testien yhteydessä on hyödynnetty toiseen testiin tarkoitettuja aliohjelmia. Tästä on voinut seurata epätäsmällinen virheilmoitus. Virheilmoituksen olisi syytä olla aina mahdollisimman täsmällinen ja perusteleva. Toisin sanoen aliohjelmista pitäisi tehdä mahdollisuuksien mukaan yleiskäyttöisiä, eikä testikohtaisia.

A Termit

EARL Evaluation and Report Language eli EARL on RDF-pohjainen standardi tiedon tallentamiseen, siirtämiseen ja muokkaamiseen.

Esteettömyys Esteettömyydellä tarkoitetaan yleisesti sitä, että tuote tai palvelu on saatavilla yhdenmukaisesti riippumatta asiakkaan fyysisistä, psyykkisistä tai sosiaalisista rajoitteista.

Gecko Gecko on Firefoxin ydin. Gecko tarjoaa monipuolisen ohjelmointirajapinnan, jota käytetään XPCOM-komponenttien avulla.

Injektointi Injektointi tarkoittaa tekniikkaa, jossa ohjelman ulkopuolinen merkkijono lisätään ohjelmaan. Toisin sanoen injektointi on vain hieno termi ohjelman ulkopuolisen tiedon ujuttamiseksi ohjelmaan. Usein merkkijono on lyhyt koodinpätkä ja ohjelma jokin verkkopalvelu. Injektointitekniikkaa käytetäänkin usein verkkopalvelujen haavoittuvuuksien hyödyntämiseen haittatarkoituksessa.

RDF RDF on lyhenne sanoista Resource Description Framework. RDF on XML-muotoinen metatiedontallennusstandardi.

Sandbox Sandbox on eristetty, tietoturvallinen ympäristö, jossa voidaan suorittaa ohjelmia rajoitetuin oikeuksin.

XPath XML Path Language eli XPath on kieli, jolla voidaan valita rakenteisesta dokumentista elementtejä.

XPCOM XPCOM on lyhenne sanoista Cross Platform Component Object Model. XPCOM-komponenttien kautta päästään käyttämään Geckon toiminnallisuuk-

	sia. XPCOM-komponentteja on mahdollista kirjoittaa lukuisilla eri kielillä.
XUL	XML User Interface Language eli XUL on käyttöliittymän kuvauskieli.