Opinnäytetyö (AMK)

Tietotekniikka

Sulautettujen järjestelmien ohjelmistotekniikka

2010

Jari Kärkkäinen

KYSELYN RAKENTEEN SUUNNITTELUOHJELMA

– JAXB työpöytäsovelluksessa

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka | Sulautettujen järjestelmien ohjelmistotekniikka

Kesäkuu 2010 | Sivumäärä: 37

Ohjaaja: TkL Jari-Pekka Paalassalo

Jari Kärkkäinen

KYSELYN RAKENTEEN SUUNNITTELUOHJELMA

Tässä työssä suunniteltiin ja ohjelmoitiin kyselyn rakenteen suunnittelemiseen tarkoitettu apuohjelma. Tavoitteena oli helppokäyttöinen ohjelma, jossa käyttäjästä johtuvien virheiden mahdollisuus olisi pieni. Ohjelma tuottaa esityksen kyselyn rakenteesta XML-muotoisena tiedostona, joka voidaan sopivalla vientitoiminnolla siirtää varsinaiseen kyselyn toteutusohjelmaan.

Lisäksi määriteltiin pienryhmälle soveltuvia ohjelmistosuunnittelun menettelytapoja yleisten projektinhallinnan ohjeistusten perusteella soveltaen niitä työn kohteena olevan ohjelman suunnitteluun.

Ohjelma on tehty Java-kielellä hyödyntäen XML-sidontaan tarkoitettua JAXB-rajapintaa. JAXB mahdollistaa helpon tiedostojen käsittelyn ilman, että ohjelmoijan tarvitsee käsitellä XML-dataa suoraan. XML-sidonta soveltuu hyvin ohjelmistoihin, joissa käsitellään lomakemuotoista dataa.

Pienessä ryhmässä voidaan saada hyviä tuloksia, mikäli projektinhallinnalliset asiat mitoitetaan oikein ja käytetään tilanteen mukaan sopivia työkaluja.

ASIASANAT: XML, JAXB, ohjelmistosuunnittelu, Java-ohjelmointi

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Embedded Software

June 2010 | Total number of pages: 37

Instructor: Jari-Pekka Paalassalo , Lic. Tech, Principal Lecturer

Jari Kärkkäinen

SURVEY STRUCTURE DESIGN PROGRAM

In this thesis a survey structure design program was designed and programmed. Target was to create an easy to use program with which chance of user derived errors would be minimized. The program produces an XML-formatted file containing a survey structure. The file can be imported into the actual survey program with an appropriate integration tool.

In addition to the program, some software design practices suitable for small group work were defined. Basis for these practices is in general project management. These practices were implemented when designing the program.

The program was written in Java and XML binding api JAXB was used to ease the work. JAXB enables easy modification of XML data with schema derived classes. XML binding proved to be effective in designing programs handling form data.

KEYWORDS:

XML, JAXB, software design, Java programming

SISÄLTÖ

1 JOHDANTO

Tässä opinnäytetyössä esitellään pääpiirteissään pienryhmässä tehtävä ohjelmisto-suunnitteluprosessi ja sen tulokset.

Tässä opinnäytetyössä pienryhmällä tarkoitetaan yhden tai muutaman henkilön muodostamaa työryhmää. Tällaisia työryhmiä on käytännössä hyvin paljon suomalaisen yrityskentän pirstaleisuuden vuoksi: valtaosa yrityksistä ohjelmistoalallakin on alle 5 hengen yrityksiä [1].

Tässä opinnäytetyössä pienryhmän käsitteen perusajatus on, että pienen ryhmän ei kannata yrittää olla suuri. Suuren ryhmän käytännöt yhdistettyinä pieneen ryhmään tuottavat raskaan byrokratian, mikä itse asiassa pikemmin estää kuin edistää työskentelyä. Luvussa 2 käsitellään yleisiä suunnittelunäkökohtia, jotka voi omaksua pienryhmän käytännöiksi. Ohjelmistokehitysprosessin muodollista kokonaisuutta ei käsitellä, vaan keskitytään ainoastaan niihin osiin, jotka ovat väistämättömiä toimivan ohjelmiston toteuttamisen kannalta.

Tässä työssä kohteena on kyselyn rakenteen suunnitteluun tarkoitettu ohjelma. Ohjelmalla tehdään lomakekyselyn rakenne kysymyksineen ja lomakkeineen XML-muotoiseksi tiedostoksi, joka voidaan siirtää varsinaiseen kyselyohjelmistoon sopivalla apuohjelmalla. Rakenteen muoto ei ole sidoksissa mihinkään kyselyohjelmisto-tuotteeseen, mutta sen referenssitoteutuksena on käytetty LimeSurvey-ohjelmistoa.

Luvussa 3 käsitellään ohjelmassa käytettävän perusteknologian piirteitä. Käytettävä JAXB mahdollistaa helpon XML-käsittelyn, mikä nopeuttaa ja yksinkertaistaa ohjelman kehitystyötä.

Varsinainen kehitystyö käsitellään luvussa 4. Kehitystyössä sovelletaan pienryhmälle soveltuvia yleisiä suunnittelunäkökohtia. Kehitystyön tuloksen, valmiin ohjelman, käsittelyn painopiste on tietomallin ja käyttöliittymän välisessä vastaavuudessa.

2 YLEISET SUUNNITTELUNÄKÖKOHDAT

Suomessa yritykset yleensä ovat hyvin pieniä ja näin on myös ohjelmistoalalla [1][2]. Taulukosta 1 käy ilmi, että pääosassa suomalaisia ohjelmistoalan yrityksiä työskentelee keskimäärin vain yksi henkilö. Näissä yrityksissä suunnittelutyö tehdään pienryhmissä, joissa kaikkia ohjelmistokehitysprosessin vaiheita ja rooleja ei voi täysin erotella. Tässä tarkastelussa keskitytään niihin tekijöihin, joihin tällaisessa pienryhmässä ohjelmistoprojektissa voi keskittyä.

Taulukko 1: Yritykset toimialoittain ja henkilöstön suuruusluokittain 2008 (TOL 2008). Suomen yritykset yhteensä ja toimiala 62 Ohjelmistot, konsultointi ja siihen liittyvä toiminta [1]

Yrityksen henkilöstömäärä	Suomen yritykset		Ohjelmistot, konsultointi ja siihen liittyvä toiminta	
	Yrityksiä	Henkilöstö	Yrityksiä	Henkilöstö
...4	284502	291269	**3873**	**3865**
5...9	18487	118552	384	2494
10...19	9444	125317	252	3325
20...49	5422	161807	165	5018
50...249	2441	245887	84	9128
250...	656	559383	25	17099
Yhteensä	320952	1502215	4783	40929

Yleisesti projekti alkaa jonkin tarpeen havaitsemisesta, määrittelee tarpeen täyttämiseen tarvittavat menettelyt ja päättyy, kun tarve on täytetty. Projekti on siis päättyvä prosessi, jolla pyritään täyttämään jokin tarve. [3]

Jotta projekti todella päättyisi onnistuneesti, tarvitaan sen hallintaa, joka määrittelee tavoitteen ja keinot niiden saavuttamiseksi. Projekti voi päättyä myös epäonnistumiseen esimerkiksi, jos projektin rahoitus loppuu tai alkuperäinen tarve muuten katoaa. Epäonnistunutkin projekti on tärkeää päättää hallitusti, jotta seurauksena ei olisi sekasortoa.

Ohjelmistoprojekti koostuu projektisuunnittelusta ja analyysistä, kehitystyöstä testauksesta sekä käyttöönoton hallinnasta. Vaiheet ovat tyypillisesti ajallisesti toisiaan seuraavia, mutta kuitenkin aina jossain määrin päällekkäisiä. Vaiheiden hallinnassa ja

projektin onnistuneessa loppuun saattamisessa on apuna kehitysvaiheessa projektiryhmän roolijako projektijohtoon, suunnittelijoihin, koodaajiin ja testaajiin. [4][5]

2.1 Projektisuunnittelu ja analyysivaihe

Projektisuunnittelussa päätetään projektin aloittamisesta ja allokoidaan resursseja sen toteuttamiseksi. Samalla määritellään projektin tavoite, esimerkiksi ohjelman valmistaminen. Tavoite tulee määritellä selkeästi ja mahdollisimman yksiselitteisesti. Projektista ei pidä tehdä loppumiseltaan määrittelemätöntä vaan päätetään, että ohjelma valmistetaan vain tiettyyn pisteeseen, joka voi olla toimiva demo tai ensimmäinen julkaisu. [5]

Pienryhmässä projektisuunnittelu yhdistyy loogisesti analyysi- ja suunnitteluvaiheisiin. Analyysivaiheessa selvitetään varsinaiset vaatimukset, jotka lopputuotteen tulee täyttää, sekä tarvittavat tiedot, joita tuote käsittelee. Tällainen yhdistetty projektisuunnittelun pääasiallinen menettely ei ole ryhmän koosta riippuvainen. Joka tapauksessa päälinjojen tulee olla selvillä ja kirjattuina ennen varsinaisen työn alkua.

Suunnittelun aikana ohjelmisto jaetaan osiin, joita voi tarvittaessa kehittää omina projekteinaan, jos kyseessä on todella laaja kokonaisuus. Tällöin päätetään myös muista yleisistä suunnittelunäkökohdista, joiden tarkoituksena on parantaa tuotteen ylläpidettävyyttä ja jatkokehitysmahdollisuuksia.

2.2 Laadun hallinta

Projektinhallinnan yleisessä ohjeistuksessa laadun hallinta toteutetaan testauksella ja katselmuksilla. Pienessä ryhmässä resurssit ovat hyvin rajalliset ja voi olla vaikeaa järjestää ohjelman testaus järkevällä tavalla. Oleellista on kuitenkin pyrkiä järjestelmällisesti löytämään ohjelmakoodissa olevia virheitä. Satunnainen kokeileminen ei vielä ole testausta. Testauksen suorittajan ei pitäisi olla ohjelman kirjoittaja, koska kirjoittaja on sokea omille puutteilleen, mikä vaikuttaa testaukseen vähentäen sen todellista tehokkuutta. [5]

Aina käytettävissä olevat keinot ovat hyviä: ohjelmakoodin selkeä dokumentointi-käytäntö ja koodin läpikäynti. Koodin läpikäyntiin voi soveltaa tarkastusmenettelyn yleistä käytäntöä [5, s. 269]. Pienryhmässä tarkastuskokousten järjestäminen ei kuitenkaan ole järkevää.

Näiden keinojen lisäksi ohjelmaa voi kokeilla kylvämällä virheitä syötteeseen tai ohjelmakoodiin. Virheitä voi tehdä tallenteisiin, kuten tiedostoihin sekä tietokantaan. Tehokasta voi olla myös metodin kehitysvaiheessa syöttää sille virheellistä tietoa poikkeustilateiden käsittelyn todentamiseksi. Tällöin puhutaan ns. virheiden kylvämisestä. [5, s. 296]

Pienryhmässä tehokas roolijakoon perustuva testausmenettely ei ole mahdollista, joten laadun hallinnassa korostuu erityisen huolellisuuden ja tehdyn työn dokumentoinnin merkitys.

2.2.1 Käytettävyys

Ohjelmiston suunnittelussa käytettävyys on tärkeä laatunäkökohta. Mikäli käytettävyyteen ei jo suunnittelussa kiinnitetä huomiota, todennäköisyys projektin epäonnistumisesta kasvaa lopputuotteen huonona menestyksenä kohdeyleisön keskuudessa. [3]

Pienryhmässä kunnollinen käytettävyystestaus ei todennäköisesti ole mahdollista, joten ohjelmoijan pitää pystyä etäännyttämään itsensä tehtävästä työstä ja asetuttava tulevan käyttäjän asemaan. Hyvä lähtökohta etäännyttämiseen on miettiä, mitkä asiat itseä yleensä ohjelmistoissa ärsyttävät ja pyrkiä tekemään omassa ohjelmassa nämä asiat helpommiksi.

Ohjelmistoissa on kuitenkin paljon vakiintuneita käytäntöjä, joita ei pidä ilman tarkkaa harkintaa muuttaa tai jättää pois. Esimerkiksi useissa ohjelmistoissa valikkopalkin ensimmäinen valikko on otsikoituna: ”Tiedosto” ja käyttäjät ovat tottuneet tuosta valikosta löytämään mm. ohjelman lopetuskomennon.

2.2.2 Ohjelmakoodin dokumentointikäytäntö

Ohjelmakoodi dokumentoidaan kirjoitettaessa mieluiten siten, että jokaisen metodin dokumentointi kirjoitetaan ennen metodin toteutuksen kirjoittamista. Metodien dokumentointiin tulee kuvaus metodin toiminnasta, sen parametreista ja palautusarvoista.

Mahdolliset huomiot lisätään käyttäen joko TODO- tai FIXME-merkintöjä. Merkintöjen tarkoituksena on kiinnittää ohjelmoijan huomio myöhemmissä työvaiheissa tarvittaviin varmentamista vaativiin kohtiin.

2.2.3 Ohjelmakoodin läpikäynnit

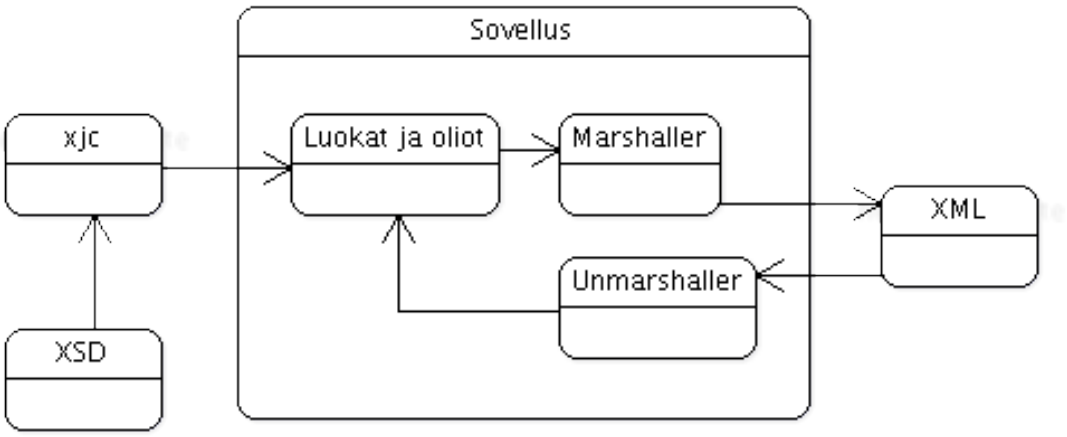
Jokaisen työvaiheen jälkeen tehty ohjelmakoodi käydään lävitse rivi riviltä käyttäen virheiden jäljitintä tarkkaillen kaikkia kriittisiä muuttujia mahdollisten sivuvaikutusten havaitsemiseksi.

Koodin läpikäynnissä tarkistetaan paitsi ohjelmallinen toiminta, myös ohjelman dokumentoinnin ajantasaisuus.

Ohjelmakoodin läpikäyntiin ei pitäisi joutua käyttämään kerrallaan muutamaa tuntia kauempaa aikaa, koska pitkällinen yksityiskohtiin keskittyminen on uuvuttavaa. Pitkittyessään läpikäynti muuttuu tehottomaksi. Läpikäynnin tehokkuuden optimointi vaikuttaa suoraan ohjelman osajakoon ja ajankäyttöön. Paria päivää pidempiä osatoteutuksia ei pidä tehdä.

3 JAXBIN KÄYTTÄMINEN

JAXB on lyhenne sanoista Java Api for XML Binding, mikä käytännössä tarkoittaa Java-luokkien ja XML-skeeman välistä rakenteellisen yhteyden ohjelmallista määrittelyä. Vaikka JAXB on osa Metro Web Services määrittelyä [6], se soveltuu hyvin käytettäväksi myös työpöytäsovelluksissa.



Kuva 1: JAXBin toimintaperiaate.

JAXBin käyttäminen sovelluksessa alkaa määrittelemällä ohjelman käsittelemien tietojen tarpeet. Tietotarpeet esitetään XML-skeemana. Skeeman perusteella generoidaan sitä vastaava luokkarakenne, mitä hyödynnetään ohjelman toteutuksessa. Menettely mahdollistaa helpon XML-käsittelyn ilman, että ohjelmoijan tarvitsee välittömästi muokata XML-tiedostoja tai -objekteja. JAXBin mahdollistama vaivaton tiedostojen lukeminen ja kirjoittaminen erityisesti nopeuttavat tuotekehitystä poistamalla käytännössä kaiken XML-datan suoran käsittelyn. [7][8] (Kuva 1)

3.1 XML-skeema

Skeema on ohjelman tuottamien XML-tiedostojen rakennekuvaus, jossa määrätään tiedoston elementtien tietotyypit ja esiintyminen. [10]

XML-skeemalla ja tietokannan relaatiomallilla on paljon yhteistä. Kumpikin kuvaavat tallennettavan ja käsiteltävän tiedon rakenteen ja tietojen välisiä hierarkkisia yhteyksiä.

3.2 Sidonta

Sidonta (binding) tapahtuu Java-kehitysympäristöön kuuluvalla XML-kääntäjällä xjc. Sidonnassa XML-skeeman perusteella automaattisesti generoidaan Java-luokat, jotka vastaavat skeeman rakenteita.

Sidonta tapahtuu ajamalla xjc konsolissa projektin juurikansiossa. Parametri -p kertoo kääntäjälle, minkä niminen Java-paketti luodaan ja parametri -d kertoo kohdekansion, mihin generoitava paketti tehdään.(Taulukko 2)

Taulukko 2: XML-kääntäjän ajokomento ja sen rakenne. Komento ajettuna konsolissa projektikansiossa.

`xjc -p fi.khii.jaxb.survey data/xml/survey/Survey.xsd -d src`	
Kääntäjä	`xjc`
Paketin nimi	`-p fi.khii.jaxb.survey`
Lähdeskeema	`data/xml/survey/Survey.xsd`
Kohdekansio	`-d src`

Kääntäjälle voi antaa muitakin ohjeita lisäämällä skeemaan tietoja sidottavista tyypeistä ja generoitavista metodeista. [7][9]

3.3 Luokat

Sidonnan avulla muodostetut luokat vastaavat skeeman rakenteita ja niitä voi käyttää kuin mitä tahansa muita luokkia. Sidonnassa luodaan erityinen `ObjectFactory` luokka, mitä käytetään olioiden luomiseen. Luotavat oliot ovat tyypillisesti lähes tyhjiä, eli niiden attribuutteja ei luoda, vaikka skeemassa olisi määritelty oletusarvot. Olioiden täyttämiseen tarvitsee tehdä joko oma metodinsa tai luoda olio lukemalla se valmiista XML-tiedostosta. Kummassakin luontitavassa on omat etunsa ja haittansa.

Luotaessa olio määräämällä sen arvot ohjelmassa etuna on, että vältytään käyttäjästä ja käyttöympäristöstä aiheutuvilta ongelmilta. Tällaisia ongelmia voivat olla esimerkiksi tiedoston tuhoutuminen, sen korruptoituminen tai sen käyttöoikeuksien väärä määrittely. Ohjelma ei enää toimi odotetulla tavalla, jos sen käyttämä tiedosto ei toimi. Lisäksi ohjelman toiminnalle välttämättömien tietojen, kuten käynnistysparametrien, täyttäminen eksplisiittisesti käytännössä poistaa käyttäjästä johtuvan virheen mahdollisuuden.

Toisaalta ulkoinen tiedosto on joustava ja siihen on helppo lisätä elementtejä tarvittaessa. Erityisesti kun käytettävien tietojen määrää ei tiedetä tai tietomäärä muuttuu, tai edellytetään, että vaihtoehtoisia tietoja on oltava helppo lisätä, on ulkoisen tiedoston käyttäminen parempi ratkaisu. Esimerkiksi ohjelman kielitiedostot ovat tällainen tapaus.

3.4 Kirjoittaminen ja lukeminen

Muokatut tiedot kirjoitetaan XML-tiedostoon `javax.xml.bind.Marshaller` luokan olion avulla. Koodiesimerkeistä (ohjelma 1 ja ohjelma 2) nähdään, että tiedoston kirjoittaminen ja lukeminen ovat ohjelmoijan kannalta hyvin yksinkertainen toimenpide. [8]

Tallennusmetodille (ohjelma 1) välitetään parametreina tallennettava JAXBElement-tyyppinen olio, joka sisältää tallennettavan datan. Lisäksi välitetään kohdetiedosto, johon tallennus tapahtuu ja tallennettavan tiedoston lisätiedoksi skeeman sijaintitieto. Metodissa luodaan JAXBContext-olio ja siihen liittyvä Marshaller-olio, joka hoitaa varsinaisen kirjoittamisen.

Ohjelma 1 : XML-tiedoston tallentamiseen käytettävä metodi fi.khii.`jaxb.JaxbHandler` luokassa.

```
protected <T> void marshal(JAXBElement<T> gl, File saveFile, String
schemaLocation) throws JAXBException {

     JAXBContext jc =
     JAXBContext.newInstance(gl.getValue().getClass());

     Marshaller m = jc.createMarshaller();

     m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, new
     Boolean(true));

     m.setProperty(Marshaller.JAXB_SCHEMA_LOCATION,
     schemaLocation);

     m.marshal(gl, saveFile);
}
```

Luettaessa XML-tiedostosta lukemismetodille (ohjelma 2) välitetään tietona, minkä tyyppinen arvo halutaan, tiedosto, josta luetaan, ja luettavan tiedoston skeema. Skeeman merkitys on se, että lukija osaa käsitellä tiedostoa oikein ja tarkistaa sen rakenteen oikeellisuuden. Metodissa luodaan JAXBContext-olio ja siihen liittyvä Unmarshaller-olio, joka huolehtii varsinaisesta lukemisesta.

Ohjelma 2: XML-tiedoston lukemiseen käytettävä metodi `fi.khii.jaxb.JaxbHandler` luokassa.

```
protected <T> T unmarshal(Class<T> class1, File xml, File schema)
     throws SAXException, JAXBException {

     Schema sc = null;

     SchemaFactory sf = SchemaFactory
     .newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);

     sc = sf.newSchema(schema);

     String packageName = class1.getPackage().getName();

     JAXBContext jc = JAXBContext.newInstance(packageName);

     Unmarshaller u = jc.createUnmarshaller();

     u.setSchema( sc );

     JAXBElement<T> doc = (JAXBElement<T>) u.unmarshal(xml);

     return doc.getValue();
}
```

4 OHJELMA

Tehtävänä on toteuttaa kyselyn suunnitteluun apuohjelma, joka muodostaa XML-tiedoston, jossa kuvataan kyselyn rakenne. Työn referenssitoteutuksena on LimeSurvey 1.8+ [11], joka on suhteellisen yleisesti käytössä oleva avoimen lähdekoodin tiedonkeruujärjestelmä. Toteutettava ohjelma ei ole kuitenkaan sidottu LimeSurveyyn, vaan sen muodostama rakenne voidaan siirtää sopivaa integraatio-ohjelmaa käyttämällä myös muihin järjestelmiin.

Ohjelma on hyödyllinen, koska LimeSurveyn käyttöliittymä on suhteellisen monimutkainen. LimeSurveyn käytön opettelu vie oman aikansa, mikä tuottaa paljon turhaa liikennettä palvelimella. Liikennemäärät ovat palvelinvuokrauksessa ja webhotellipalveluissa yleisesti rajoitettuja, mutta laajennettavissa lisämaksusta. Käyttöliittymän opettelu tuottaa ongelmia myös kyselyä suunnitellessa, sillä useita ominaisuuksia ei tarvita perusasioiden hoitamiseen. Nämä asiakkaan kokemat ongelmatilanteet voivat lisätä ylläpidon työtä merkittävästi.

Liiketoiminnalliselta kannalta kyselyn rakenteen suunnittelu www-käyttöliittymässä on myös ongelmallista. Suunnittelu edellyttää pääsyä kyselyn toteuttamiseen, jolloin myös selvästi laskutusperusteena olevat kyselyn asetukset, kuten kyselyn julkaisuaika ja julkaisun pituus voivat tulla asiakkaan muokattaviksi.

4.1 Suunnittelu

Suunnittelun lähtökohdaksi otettiin, että ohjelman käytön on oltava helpommin opittavissa kuin LimeSurvey 1.8+ -käyttöliittymän käyttö. Ohjelma tulee olla myös olla siirrettävissä käyttöjärjestelmien ja koneiden välillä helposti.

Siirrettävyys toteutuu käyttämällä perusteknologiana Javaa siten, että ohjelmaa ei tarvitse asentaa koneelle, vaan sitä voidaan käyttää suoraan USB-tikulta.

Alkuvaiheessa ohjelmaan toteutetaan kyselyn, kysymysryhmän ja kysymyksen luominen. Kysymysten ehdollisuus ja monipuoliset kysymystyypit, sekä lomakkeiden esikatselu jätetään toteuttamatta. Ehdollisuus ja kysymystyypit toteutetaan kuitenkin mallin tasolla XML-skeemaan.

Ohjelman käyttöliittymä toteutetaan alusta alkaen monikielisenä ja ominaisuus monikielisten kyselyiden tekemiseksi toteutetaan mallin tasolla, mutta ei vielä käyttöliittymään.

Pois jätettäviä ominaisuuksia ovat mm. monipuoliset kysymystyypit, kuten kysymysryhmät, joihin voi lisätä omia sarakeotsikoita. Samoin pois jätetään kysymysten ja lomakkeiden ehdollisuus. Ehdollinen kysymys tai lomake näkyy tai piilotetaan valmiissa kyselyssä ehdollisesti jonkin edeltävän kysymyksen vastauksen perusteella. Alkuvaiheen toteutuksesta jätetään pois myös useimmat helppokäyttöisyyteen liittyvät ominaisuudet, kuten pikanäppäimet. (Taulukko 3)

Taulukko 3: Keskeisimmät käytettävyyteen ja toiminnallisuuteen liittyvät ominaisuudet, joita ei alkuvaiheessa toteuteta.

Ominaisuus	Toiminta
Kopiointi	Kopioi kysymysryhmän tai kysymyksen uudeksi kohteeksi. Kopioitaessa myös kaikki kohteen aliominaisuudet kopioituvat.
Poistaminen	Poistaa kysymysryhmän, kysymyksen tai näiden ehdon. Poistettaessa myös kaikki kohteen aliominaisuudet poistetaan.
Järjestely	Kysymyksiä ja kysymysryhmiä voi järjestellä raahaamalla hiirellä suoraan rakennepuussa.
Pikanäppäimet	Tallennusta, kopiointia ja järjestelyä varten pikanäppäinyhdistelmät ctrl+s, ctrl+c, ctrl+x ja ctrl+v. Pikanäppäiminä käytetään vakiintuneita, kaikille käyttäjille tuttuja pikanäppäinyhdistelmiä.
Kyselyn monikielisyys	Käyttäjä voi lisätä kyselylle vaihtoehtoisen kielen. Alkukielinen kyselyrakenne kopioidaan SurveyAdditionalLangs listan elementiksi. Käyttöliittymässä muokkausalueella on välilehti kyselyn vaihtoehtokielistä versiota varten.
Ehdollisuus	Kysymysryhmälle ja kysymykselle voi asettaa ehdollisen näkymisen.
Monipuoliset kysymystyypit	Käyttäjä voi vaihtaa kysymyksen tyyppiä tavallisen sanallisen kysymyksen ja erilaisten valintakysymysten välillä.

4.2 Tietotarpeet

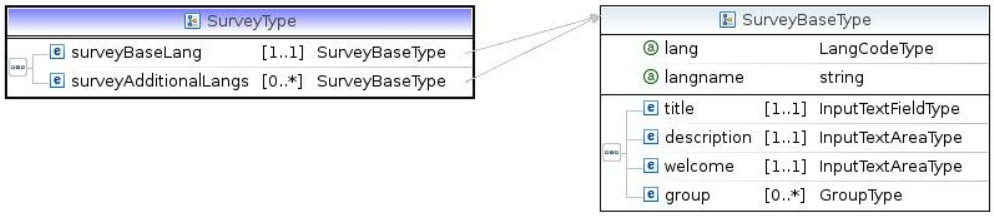
Ohjelma tuottaa kyselyrakennetiedoston XML-muodossa. Tiedon tuontia varten LimeSurvey 1.8+:n toteutetaan myöhemmin integraatio-ohjelma, jolla tiedot saadaan tuotua vaivattomasti. Jotta kyselyn rakenne olisi mahdollinen toteuttaa ohjelmalla, on siihen saatava rakennettua tarvittava tietomalli.

Tietomalli toteutetaan JAXBin avulla, eikä erillistä tietokantaa tarvita.

4.2.1 Kysely

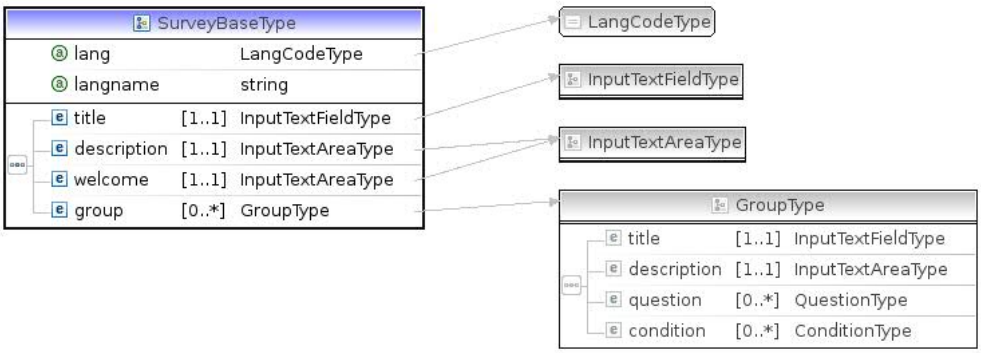
Kysely koostuu kysymysryhmistä, kysymyksistä ja niihin liittyvistä ehdoista. Kyselyyn liittyvä skeema on merkittävästi monimutkaisempi kuin monikielisyyteen tai käynnistysparametreihin liittyvä.

Kysely koostuu peruskielisestä kyselystä ja mahdollisista lisäkielistä. Lisäkielien ei tarvitse olla samoja kuin käyttöliittymässä käytettävissä olevat kielet, vaan muitakin kieliä voi käyttää. Kyselyn ylätason rakenne on esitetty kuvassa 2.



Kuva 2: SurveyType-luokan skeeman rakenne

Kyselyn peruskielinen ja lisäkieliset esitykset ovat kaikki samanlaisia. Kun ohjelmassa lisätään kyselyyn kieliversio, alkuperäinen peruskielinen versio kopioidaan uudeksi olioksi ja tallennetaan SurveyType-olion surveyAdditionalLangs listaan, joka on tyyppiä ArrayList<SurveyBaseType>. Ohjaustietoina ovat tällöin kyselyn kieli ja kielikoodi, jotka ovat SurveyBaseType:n attribuutit lang ja langname. (Kuva 3)

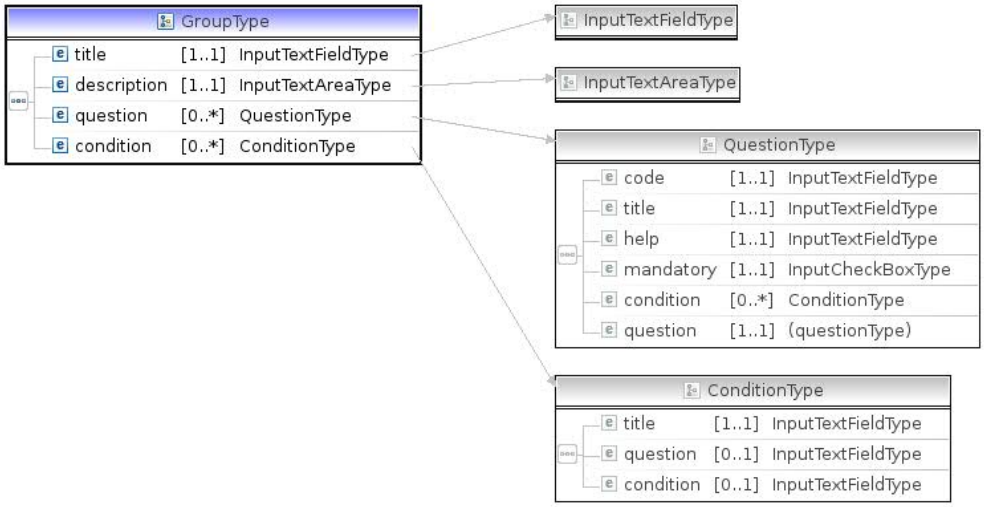


Kuva 3: SurveyBaseType luokan skeeman rakenne

Perustietojen lisäksi kukin kysely sisältää joukon kysymysryhmiä. Kysymysryhmät voi ymmärtää aihepiirin mukaisiksi ryhmittelyiksi, kuten taustatiedot tai asennoituminen johonkin selvitettävään asiaan. Tavallista on esittää kysymysryhmä yhtenä lomakkeena tai lomakesivuna.

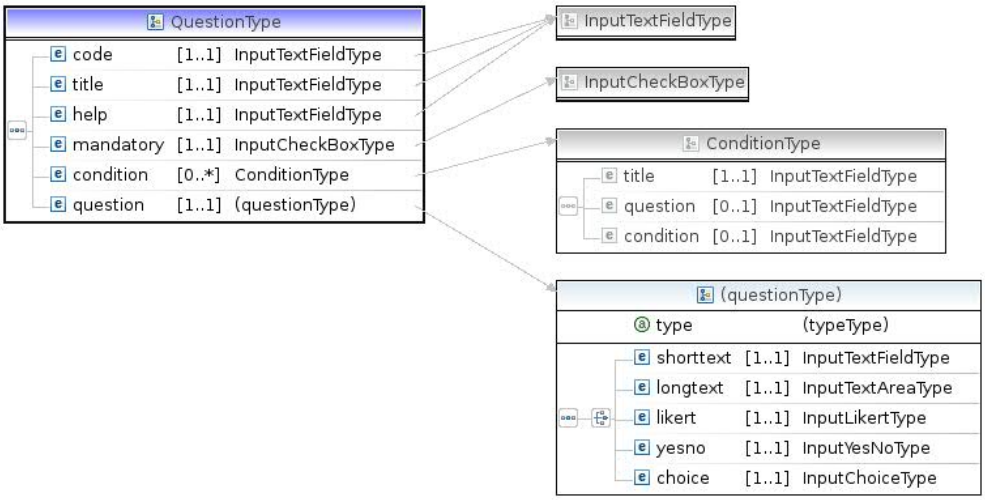
Kysymysryhmiä, jotka ovat GroupType-tyyppisiä voi olla 0 tai enemmän. Kysymysryhmässä on QuestionType-tyyppisiä kysymyksiä samoin 0 tai enemmän. Todellisessa kyselyssä kysymysryhmiä kyselyyn tulee käytännössä aina ainakin yksi ja ryhmään ainakin yksi kysymys, mutta suunnitteluohjelmassa pitää voida tallentaa keskeneräinen kysely, jossa ei välttämättä vielä ole yhtään ryhmää tai kysymystä. (Kuva 4)

Kysymysryhmä sisältää otsikon, kuvauksen ja joukon kysymyksiä. Kysymysryhmälle voi asettaa myös ehtoja, joiden mukaan se esitetään.



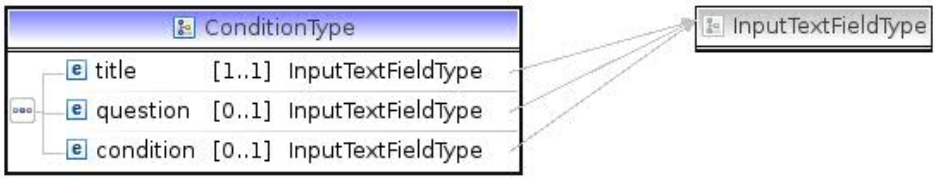
Kuva 4: GroupType luokan skeeman rakenne

Kysymyksen tietoina ovat kysymyskoodi, jota käytetään tietojen erotteluun, kysymyksen otsikko ja avusteteksti. Kysymys voi olla pakollinen ja sillä voi olla ehdollinen näkyvyys. Kysymystyyppi vaihtelee kysymyksen mukaan. Skeemaan on toteutettu vaihtoehdot vain muutamista kysymystyypeistä. (Kuva 5)



Kuva 5: QuestionType luokan skeeman rakenne

Kysymyksille ja kysymysryhmille voi asettaa ohjelmassa ehdollisen näkyvyyden, joten skeemassa on rakenne ehtoa varten. (Kuva 6)



Kuva 6: ConditionType luokan skeeman rakenne

Ehdon rakenteessa question on viite kysymykseen, jonka perusteella ehdollisuus on voimassa. Varsinainen ehdon sääntö on condition-kentässä.

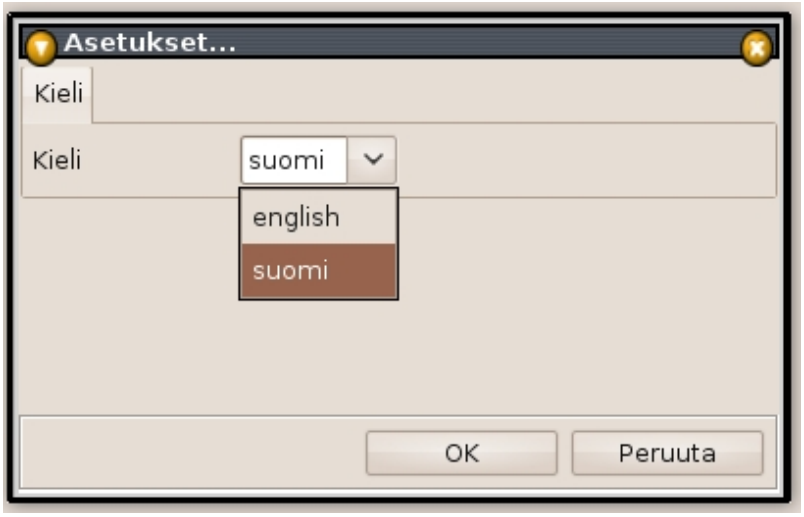
4.2.2 Monikielisyys

Kyselyt pitää voida toteuttaa useilla kielillä. Pelkästään suomalaisia toimijoita palvellakseen kysely pitää voida toteuttaa suomen lisäksi ainakin ruotsiksi ja englanniksi. Näiden mahdollisuuksien heijastuminen myös suunnitteluohjelman käyttöliittymään on luonteva seuraus.

Suunnitteluohjelman käyttöliittymän monikielisyys on toteutettu merkkijonojen korvaamisella. Ohjelmakoodissa sanojen tilalla käytetään avainmerkkijonoja, jotka vastaavat jotain varsinaista sanaa tai tekstiä. Sanat ja tekstit saadaan kulloinkin käytettävää kieltä vastaavasta kielitiedostosta, joka on XML-muodossa. Tiedostossa merkkijonot ovat avain-arvopareina, jotka luetaan yhteen HashMap-olioon `fi.khii.survey.Lang` -luokassa, mistä avainta vastaava arvo haetaan kun merkkijonoa, sanaa tai tekstiä, käytetään osana käyttöliittymää.

Kielitiedostojen XML-skeema on esitetty liitteessä 1 ja liitteessä 2 on esitettynä katkelma kielitiedostosta. Skeemasta näkee, että kielitiedoston rakenne on hyvin yksinkertainen. Perustietoina ovat kieli, kielikoodi ja kielitiedoston tekijä, joiden lisäksi on joukko avain-arvopareja.

Perustiedoista kieli on kielen nimi kyseisellä kielellä ja kielikoodi on kielen kaksimerkkinen ISO 639 -standardin mukainen tunniste [12]. Kielitiedoston tekijän nimi on tiedostossa siksi, että jos kielitiedostoja tekee joku ulkopuolinen, mahdollisesti vapaaehtoinen tekijä, saa hän selvän maininnan ansioistaan tekijänä.



Kuva 7: Asetusten muokkaamiseen käytettävä modaalinen ikkuna. Kuvassa käyttöliittymän kielen valitseminen.

Kielen vaihtaminen tapahtuu ohjelman asetuksista (Kuva 7). Kieliasetus tulee voimaan vasta ohjelman uudelleenkäynnistyksen jälkeen, koska avainten mukaiset todelliset merkkijonot asetetaan osaksi käyttöliittymää kun käyttöliittymäoliot luodaan.

Mikäli kielitiedostosta puuttuu käyttöliittymän merkkijono, kirjoittaa Lang-olio lokitiedostoon ilmoituksen puuttuvasta merkkijonosta ja käyttöliittymässä käytetään

haettua avainmerkkijonoa. Lokitiedostojen luomiseen käytetään `java.util.logging.Logger` ja `java.util.logging.FileHandler` luokkia.

4.2.3 Käynnistysparametrit

Käynnistysparametrien merkitys on pääasiassa parantaa ohjelman käytettävyyttä. Ohjelman lopetuksen yhteydessä tallennetaan tietoja ohjelmaikkunan sijainnista ja sen osien keskinäisistä kokosuhteista. Ohjelmaa uudelleen käynnistettäessä tiedot luetaan tallennetusta tiedostosta ja käytetään uuden olion luomisessa.

Kun ohjelmaikkuna käynnistettäessä ilmestyy näytölle samaan paikkaan ja saman näköisenä kuin se oli ohjelmaa sammutettaessa, voi käyttäjä jatkaa helposti siitä mihin edellisen kerran lopetti.

Ohjelmaa voi käyttää USB-tikulta, mikä mahdollistaa siirtymisen eri koneiden välillä. Siirtymisen aiheuttamien, koneiden asetusten eroista johtuvien ongelmatilanteiden varalta ohjelman voi käynnistää myös oletusasetuksilla, jolloin ikkuna asettuu näytön vasempaan ylänurkkaan. Ohjelma käynnistetään oletusasetuksilla antamalla komento `java -jar JaxbSurvey.jar default`.

Poiketen monikielisyydestä, käynnistysparametrien skeemassa sekä Ini-luokassa kaikki tallennettavat arvot on määrätty eksplisiittisesti. Kielitiedostoissa voi olla ylimääräisiä tai puuttuvia arvoja, mutta käynnistysparametritiedostossa pitää olla juuri määrätyt tiedot. Taulukossa 4 on esitetty tallennettavat käynnistysparametrit, joista osa on määritelty valmiiksi tulevia tarpeita varten ja jotka eivät ole vielä käytössä.

Taulukko 4: Ohjelman käyttämät ja tallentamat käynnistysparametrit.

Parametri	Merkitys	Käytössä
lang	Ohjelman käyttöliittymän kieli.	kyllä
lastFile	Viimeksi muokatun tiedoston nimi.	ei
lastFolder	Viimeksi käytetty tallennuskansio.	ei
windowHeight	Ikkunan korkeus.	kyllä
windowWidth	Ikkunan leveys.	kyllä
windowX	Ikkunan sijainti pikseleinä x-akselilla mitattuna näytön vasemmasta ylänurkasta.	kyllä
windowY	Ikkunan sijainti pikseleinä y-akselilla mitattuna näytön vasemmasta ylänurkasta.	kyllä

Parametri	Merkitys	Käytössä
dividerLocation	Ikkunan rakennepuu- ja editointialueen jakajan sijainti ikkunan vasemmasta reunasta mitattuna.	kyllä
openLastEdited	Kertoo, avataanko seuraavalla käynnistyksellä viimeksi muokattu tiedosto.	ei

Käynnistysparametrien XML-skeema on esitetty liitteessä 3 ja esimerkkitiedosto liitteessä 4.

4.3 Toteutus

Ohjelman toteutuksessa on neljä oleellista osaa: itse kysely, käynnistysparametrit, monikielisyys ja ohjelman kontrollerina toimiva käynnistysohjelma SurveyCreator. Taulukossa 5 on esitetty ohjelman paketointi selityksineen.

Tässä toteutuksen tarkastelussa keskitytään pääasiassa kyselyn XML-skeeman ja ohjelman käyttöliittymäkomponenttien väliseen yhteyteen.

Taulukko 5: Ohjelman paketointi ja luokat. Lihavoituna on korostettuna XML-sidonnassa automaattisesti luodut luokat. Kursiivilla on korostettuna interface-luokat.

Paketti	Luokat	Tehtävä
fi.khii.jaxb	XmlHandler	Huolehtii tiedostojen lukemisesta ja kirjoittamisesta. Paketissa fi.khii.survey olevat Lang, Ini ja Survey laajentavat tätä luokkaa.
fi.khii.jaxb.i18n	**LangType** **Mtype** **ObjectFactory**	Ohjelman monikielistämiseen liittyvät luokat
fi.khii.jaxb.ini	**IniType** **ObjectFactory**	Käynnistysparametrien tallentaminen ja lukeminen
fi.khii.jaxb.survey	**Survey SurveyType** **SurveyBaseType** **GroupType** **QuestionType** **ConditionType** **InputCheckBoxType** **InputChoiceType** **InputDateType** **InputLikertType** **InputTextAreaType** **InputTextFieldType** **ObjectFactory**	Kyselyn rakenteeseen liittyvät luokat Kysymystyyppeihin ja niiden tietoihin liittyvät luokat. Paketissa fi.khii.survey.ui ovat näitä vastaavat käyttöliittymäkomponentit.
fi.khii.survey	SurveyCreator	Ohjelman käynnistys ja kontrollointi.

Paketti	Luokat	Tehtävä
	Survey Lang Ini	
fi.khii.survey.ui	EditPanel EditPanelGroup EditPanelSurvey EditPanelQuestion *Inputfield* InputTextArea InputTextField InputCheckBox SettingsDialog SurveyTree SurveyTreeNode	Käyttöliittymäkomponentit Tiedonsyöttölomakkeet Tiedonsyöttölomakkeiden syötekentät Asetusikkuna Rakennepuu

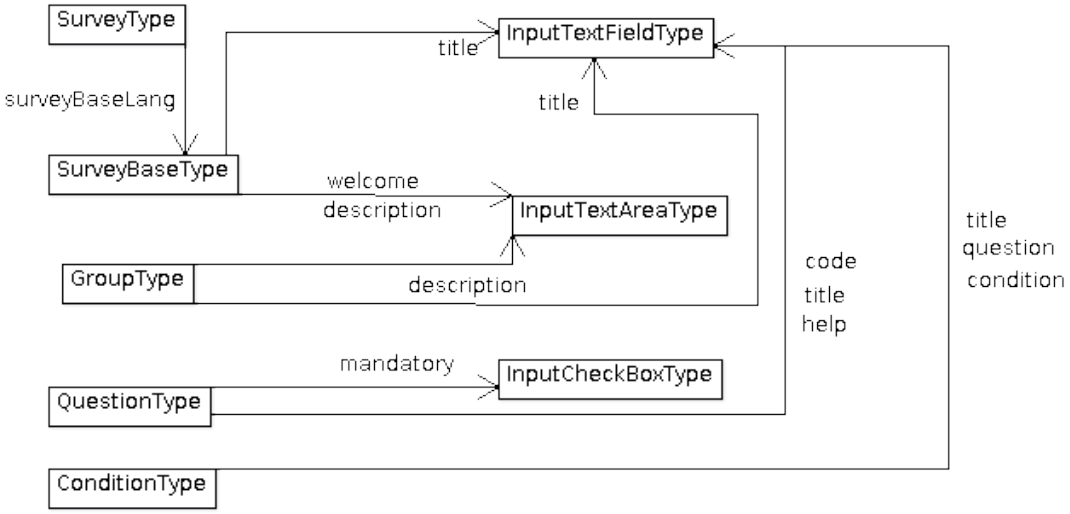
4.3.1 Kyselyn luokkarakenne

Automaattisesti xjc-kääntäjällä generoitu kyselyn luokkarakenne on samanlainen kuin sitä vastaavan XML-skeeman rakenne. Kuvassa 8 on esitetty luokkarakenne ilman luokkien metodeja ja attribuutteja.

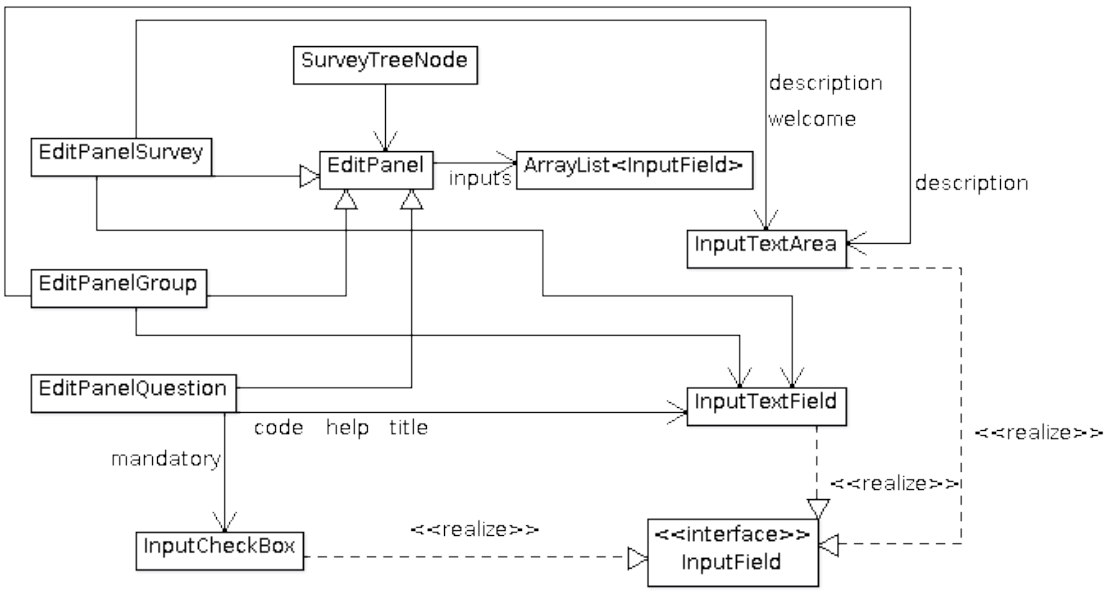
Luokkarakenteesta näkyy myös, että vaikka käytännössä jokaisessa kyselyssä onkin vähintään yksi kysymysryhmä, ei sen olemassaolo ole välttämätöntä.

Käyttöliittymäkomponenttien suhteissa näkyy, että muokkauspaneeleissa, jotka laajentavat EditPanel-luokkaa on joukko InputField-luokkia, jotka vastaavat XML-skeemassa olevia määrittelyjä. (Kuva 9)

Varsinaisena tietovarastona toimii SurveyTreeNode, jolla on ominaisuutenaan EditPanel. XML-data säilytetään SurveyTreeNoden userObject ominaisuudessa.



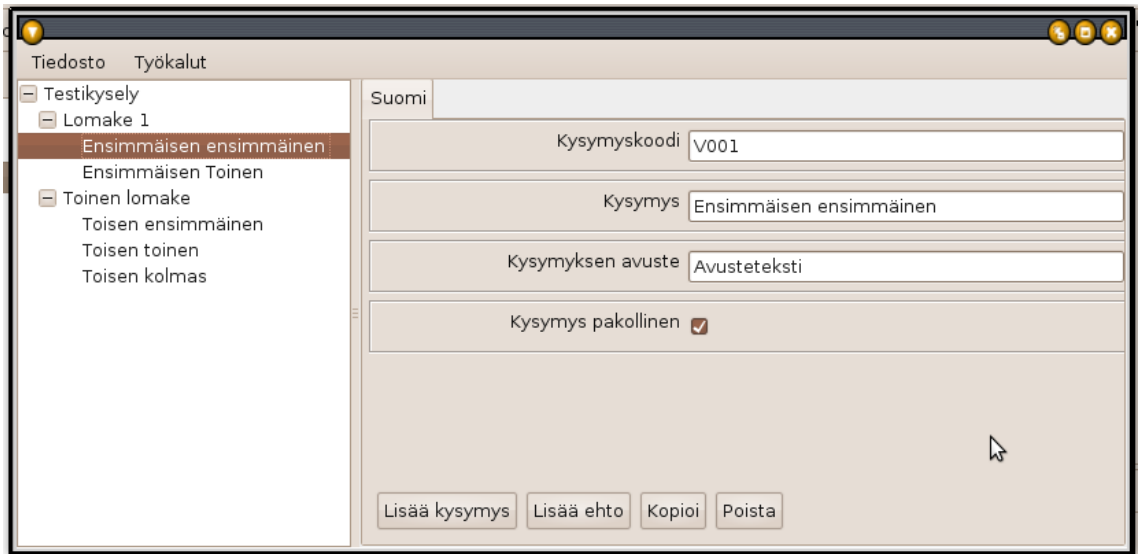
Kuva 8: Paketin fi.khii.jaxb.survey luokkarakenne.



Kuva 9: Paketin fi.khii.survey.ui rakenne SurveyTreeNode näkökulmasta.

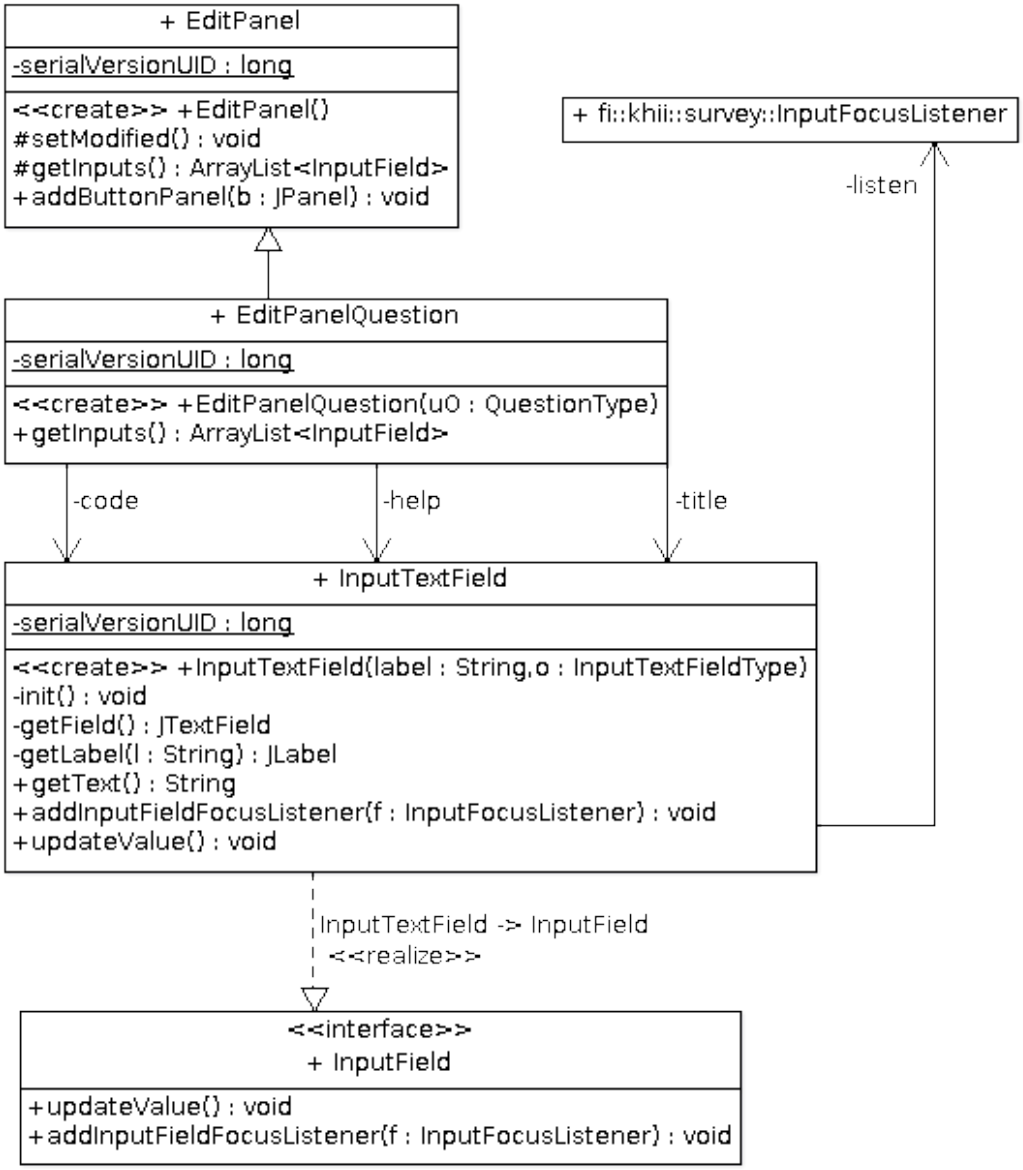
4.3.2 Käyttöliittymä

Ohjelman käyttöliittymä on tehty mahdollisimman yksinkertaiseksi, ilman mitään ylimääräisiä osia. Kyselyn rakenne näkyy puuna ikkunan vasemmassa reunassa. Kun puusta valitaan kohde, kohteen muokkauspaneeli näytetään ikkunan oikeassa reunassa. Muokkauspaneelin alapuolella on painikerivi, josta valitaan valittuun kohteeseen liittyvät toiminnot. (Kuva 10)



Kuva 10: Ohjelman käyttöliittymä. Vasemmalla tietorakenne ja oikealla muokkauspaneli.

Muokkauspaneelissa on joukko InputField-tyyppisiä komponentteja, jotka ovat JPanelin laajennoksia. Komponentteihin liittyy kuuntelija, joka tarkkailee varsinaisen syötekentän kohdistusta. Kohdistuksen siirtyminen pois syötekentästä aiheuttaa FocusEventin, jonka focusLost metodissa ohjataan syötekentän arvo vastaavan XML-datan arvoksi. Näin muutettu tieto tallentuu tietorakenteeseen ilman käyttäjän erillisiä toimia. (Kuva 11)



Kuva 11: Kysymyksen tietojen muokkauspanelin luokkarakenne. Havainnollisuuden vuoksi kuvasta on jätetty pois InputCheckBox-luokka, joka liittyy kysymyksen pakkolisuuden määräämiseen.

4.4 XML-testaus

Ohjelman testaus XML-syötteen osalta toteutetaan tuottamalla virheitä luettaviin tiedostoihin. Testauksen tarkoituksena ei ole ainoastaan löytää ohjelman toiminnassa olevia virheitä, vaan myös dokumentoida virhetilanteessa käyttäytyminen. Virhetilanteen käyttäytymistä verrataan odotettuun normaalitilanteeseen ja vertailun perusteella päätetään jatkokehityksestä.

Kaikkien luettavien tiedostojen osalta testataan tiedoston puuttuminen, tiedostosta puuttuvien tietojen käsittely ja tiedoston väärä muoto sekä skeeman puuttuminen ja virheet. Skeeman virheet ovat mahdollisia lähinnä siirryttäessä vanhasta versiosta uuteen, mikäli skeemaan on tullut muutoksia.

Kirjoitettavien tiedostojen osalta testattiin skeeman puuttuminen ja virheellisyys, tallennuskansion puuttuminen sekä käyttöoikeusvirheet tallennuskansion ja korvattavan tiedoston tapauksissa.

4.4.1 Kielitiedostojen testaus

Kielitiedostot eivät ole ohjelman toiminnan kannalta kriittisiä, eikä niissä olevien puutteiden tai virheiden pitäisi estää ohjelman käyttöä. Testausmenettelyssä löytyi kuitenkin ohjelman toimintaa ja käyttöä merkittävästi haittaavia tilanteita, mihin tulee puuttua jatkokehityksessä. Ilman testausmenettelyä virheitä ei olisi havaittu. (Taulukko 6)

Normaalitoiminnassa kielitiedosto luetaan ohjelmaan lang-kansiosta ja sen tiedot siirtyvät käyttöliittymässä käytettävään HashMap-olioon. Puuttuva kielitiedostokansio ja kielitiedoston väärä muoto estävät ohjelman normaalin käyttämisen. Tämän lisäksi skeeman puuttuminen tai virheet sekä kielitiedoston puuttuminen tuottavat käyttötilanteen, joissa käyttöliittymän merkkijonokorvaus ei voi toimia.

Taulukko 6: Kielitiedostojen testausmenettely. Korostettuna ohjelman käytölle merkittävää haittaa aiheuttavat tilanteet.

Testi	Odotusarvo	Testaustulos
Puuttuva tiedosto	Käyttöliittymässä merkkijonoina ovat haettavien korvausmerkkijonojen avaimet. Puuttuvasta tiedostosta kirjoitetaan lokimerkintä. Muuten normaali toiminta. Puuttuvista tiedoista kirjoitetaan ilmoitus lokitiedostoon.	Ohjelma käynnistyy ja merkkijonojen tilalla ovat niiden avaimet. Lokitiedostojen kirjoitus onnistuu.
Puuttuva kansio	Kuin puuttuva tiedosto.	Asetusikkuna ei avaudu. Muuten odotusten mukainen toiminta. Virhe johtuu siitä, että asetusten kielisivulla pudotusvalikkoon luetaan tiedostojen nimiä.
Puuttuvat tiedot	Puuttuvista tiedoista kirjoitetaan ilmoitus lokitiedostoon.	Korvausmerkkijonoista kirjoitetaan ilmoitus lokiin.

Testi	Odotusarvo	Testaustulos
Väärä muoto	Lukeminen epäonnistuu. Käsittely puuttuvan tiedoston mukainen.	Perustietojen, kieli, kielikoodi ja tekijä puuttuminen estää kielitiedoston lukemisen. Kielen vaihtaminen virheelliseen muotoon tuottaa tilanteen puuttuva kansio mukaisen toiminnan.
Skeeman puuttuminen	Kuin puuttuva tiedosto	Toiminta odotusten mukainen
Skeeman väärä muoto	Kuin puuttuva tiedosto	Toiminta odotusten mukainen

4.4.2 Käynnistysparametrien testaus

Käynnistysparametrien toiminnassa oleellista on, että ne jokaisella käynnistyskerralla ovat olemassa. Kaikissa tallennustiedostoon liittyvissä ongelmissa oletusarvojen tulisi tulla käyttöön. (Taulukko 7)

Testausmenettelyssä löytyi tilanteita, joista on käyttäjälle haittaa. Ohjelma ottaa käynnistyessään käyttöön oletusasetukset, mutta käyttäjä ei saa ilmoitusta tilanteen syystä. Jatkokehityksen kannalta on tärkeää ottaa huomioon tällaiset käyttäjälle näkymättömät virhetilanteet.

Taulukko 7: Käynnistysparametrien testausmenettely. Korostettuna ohjelman käytölle merkittävää haittaa aiheuttavat tilanteet.

Testi	Odotusarvo	Testaustulos
Puuttuva tiedosto, lukeminen	Ohjelma käynnistyy oletusasetuksilla.	Toiminta odotusten mukainen.
Puuttuva kansio, lukeminen	Ohjelma käynnistyy oletusasetuksilla.	Toiminta odotusten mukainen.
Puuttuva tiedosto, kirjoittaminen	Ohjelma sammuu normaalisti, ini-tiedosto luodaan.	Toiminta odotusten mukainen.
Puuttuva kansio, kirjoittaminen	Ohjelma sammuu ja lokiin merkintä virheestä.	Toiminta odotusten mukainen. JaxbException ja FileNotFoundException
Skeeman puuttuminen, lukeminen	Ohjelma käynnistyy oletusasetuksilla. Lokiin merkintä virheestä.	Toiminta odotusten mukainen.

Testi	Odotusarvo	Testaustulos
Skeeman puuttuminen, kirjoittaminen	Ohjelma sammuu normaalisti. Lokiin kirjoitetaan ilmoitus virheestä ja tieto, että seuraavalla käynnistyskerralla käytetään oletusasetuksia.	Lokimerkintää ei tule, mutta ini-tiedosto kirjoitetaan normaalisti. Kirjoitettaessa dataa ei validoida skeemaa vasten.
Puuttuvat tiedot	Ohjelma käynnistyy oletusasetuksilla	Kieliarvon lang tyhjä tai väärä arvo estää käyttöliittymän kielen löytymisen. Käyttäjä voi korjata tilanteen valitsemalla asetuksista käytettävän kielen.
Väärä muoto	Ohjelma käynnistyy oletusasetuksilla	Toiminta odotusten mukainen. Sammutettaessa uusi ehjä ini-tiedosto kirjoitetaan.
Skeeman väärä muoto, lukeminen	Ohjelma käynnistyy oletusasetuksilla	Toiminta odotusten mukainen.
Skeeman väärä muoto, kirjoittaminen	Ohjelma sammuu normaalisti.	Toiminta odotusten mukainen.

4.4.3 Kyselyrakenteen testaus

Kyselyrakenteen testauksessa jouduttiin testaamaan myös hyvin epätodennäköisiä virhetilanteita. Puuttuva tiedosto lukemisen aikana voi toteutua käytännössä vain jos yhteys levylle katkeaa, tai jos joku muu käyttäjä siirtää tai nimeää tiedoston eri nimellä juuri kun tiedostovalitsin on avattu.

Useimmat lukemiseen liittyvät virhetilanteet voivat toteutua vain, jos tallennettuja tiedostoja muokataan jollain muulla editorilla. Tällöin ohjelman ei tarvitse yrittää avata väärin muokattua tiedostoa, vaan pelkkä virheilmoitus riittää.

Kirjoittamisen aikaisia virhetilanteita, jotka liittyvät tiedoston väärään muotoon tai puuttuviin tietoihin ei ole mahdollista simuloida muuttamatta ohjelmakoodia. Näitä tapauksia ei ole sisällytetty testausmenettelyyn.

Kyselyrakenteen testausmenettely on esitetty taulukossa 8. Testauksen aikana ei testitapauksissa havaittu merkittävää haittaa aiheuttavia tilanteita. Tilanne, jossa kirjoittaminen epäonnistuu käyttöoikeusvirheen vuoksi ei kuitenkaan toimi odotetulla tavalla, vaan käyttäjä joutuu itse valitsemaan ”Tallenna nimellä” -vaihtoehdon.

Taulukko 8: Kyselyrakenteen testausmenettely. Korostettuna ohjelman käytölle merkittävää haittaa aiheuttavat tilanteet.

Testi	Odotusarvo	Testaustulos
Puuttuva tiedosto, lukeminen	Käyttäjälle annetaan ilmoitus epäonnistuneesta lukemisesta ja tehdään lokimerkintä.	Toiminta odotusten mukainen.
Puuttuva kansio, lukeminen	Kuin puuttuva tiedosto, lukeminen.	Toiminta odotusten mukainen.
Puuttuva tiedosto, kirjoittaminen	Ohjelma tarjoaa ”Tallenna nimellä” vaihtoehdon.	Toiminta odotusten mukainen.
Puuttuva kansio, kirjoittaminen	Ohjelma tarjoaa ”Tallenna nimellä” vaihtoehdon.	Toiminta odotusten mukainen.
Puuttuvat tiedot, lukeminen	Kuin puuttuva tiedosto, lukeminen.	Toiminta odotusten mukainen.
Väärä muoto, lukeminen	Kuin puuttuva tiedosto, lukeminen.	Toiminta odotusten mukainen.
Skeeman puuttuminen, lukeminen	Käyttäjälle annetaan ilmoitus epäonnistuneesta skeeman lukemisesta ja tehdään lokimerkintä.	Toiminta odotusten mukainen.
Skeeman puuttuminen, kirjoittaminen	Skeemaa ei tarvita kirjoitettaessa. Tiedosto kirjoitetaan normaalisti.	Toiminta odotusten mukainen.
Tiedoston käyttöoikeusvirhe, lukeminen	Kuin puuttuva tiedosto, lukeminen.	Toiminta odotusten mukainen.
Tiedoston käyttöoikeusvirhe, kirjoittaminen	Kuin puuttuva tiedosto, kirjoittaminen.	Ohjelma ilmoittaa tallennuksen epäonnistuvan, mutta ei tarjoa muita vaihtoehtoja.

5 YHTEENVETO

Tässä työssä osoitettiin, että oikealla ohjelmistokomponenttien valinnalla ja kehitystyöryhmän koon kannalta järkevillä suunnittelukäytännöillä voidaan sujuvoittaa ohjelmistokehitysprosessia merkittävästi. JAXBin avulla XML-tiedostojen luominen ja lukeminen on yksinkertaista. XML-skeemalla määritellyn tietorakenteen ja käyttöliittymän välinen yhteys on mutkattomasti toteutettavissa ja näin ohjelman ylläpitokehitys helpottuu.

JAXB ja XML-sidonta soveltuvat hyvin osaksi ohjelmia, joissa käsitellään tietokantamaisia, lomakemuotoisia tietoja ilman välitöntä tarvetta tietokantayhteydelle.

Määriteltyjä kehityskäytäntöjä voidaan soveltaa jatkossa kaikissa ohjelmisto-projekteissa, joissa työryhmän koko on hyvin pieni. Suurissa ryhmissä muodollinen projektiroolijako on kuitenkin mahdollinen ja sitä tulee soveltaa. Oleellista on, että pienen ryhmän ei kannata tekeytyä suureksi, eikä suuren ryhmän pidä ottaa käyttöön pienen ryhmän käytäntöjä. Väärä muodollisuuden skaala johtaa joko hallitsemattomuuteen tai liikaan byrokraattisuuteen, jotka molemmat pysäyttävät kehitystyön.

Työn aikana kyselyn rakenteen suunnitteluohjelma ei valmistunut täysin toiminnalliseksi kokonaisuudeksi. Vain perustoiminnallisuus toteutettiin, mutta ohjelman jatkokehitys on kuitenkin helppoa tehdyn määrittelyn mukaisesti.

LÄHTEET

[1] Tilastokeskus, *Yritysrekisterin vuositilasto, yritystiedot TOL 2008*, [www-dokumentti]. Saatavilla http://www.stat.fi/til/syr/tau.html. (Luettu: 27.4.2010)

[2] Metsä-Tokila, Timo, *Ohjelmistoala*, Toimialaraportti 1/2009, Työ- ja elinkeinoministeriö, 2009.

[3] Wiio, A., *Käyttäjäystävällisen sovelluksen suunnittelu*, Helsinki: Edita Publishing Oy, 2004.

[4] Murch, R., *IT-projektinhallinta*, Helsinki: Edita Publishing Oy, 2002.

[5] Haikala, I. ja Märijärvi, J., *Ohjelmistotuotanto*, Helsinki: Talentum Media Oy, 2006.

[6] Oracle SDN, *Metro Web Services Overview*, [www-dokumentti]. Saatavilla: http://java.sun.com/webservices/ (Luettu: 26.4.2010)

[7] Ort, Ed and Mehta, Bhakti, *Java Architecture for XML Binding (JAXB)* [www-dokumentti]. Saatavilla http://java.sun.com/developer/technicalArticles/WebServices/jaxb/ (Luettu: 26.4.2010)

[8] Laun, Wolfgang, *A JAXB Tutorial* , [www-dokumentti]. Saatavilla: https://jaxb.dev.java.net/tutorial/ (Luettu: 27.4.2010)

[9] Sun Microsystems, *The Java Web Services Tutorial, Customising JAXB Bindings*, [www-dokumentti]. Saatavilla: http://java.sun.com/webservices/docs/2.0/tutorial/doc/JAXBUsing4.html (Luettu: 27.4.2010)

[10] Walkama, P. ja Laakkonen, A., *Inside XML skeema*, Helsinki: Edita Publishing Oy, 2004.

[11] LimeSurvey.org, *LimeSurvey.org - THE Survey software - free and open source!* [www-dokumentti]. Saatavilla: http://www.limesurvey.org/. (Luettu: 26.4.2010).

[12] Library of Congress, *Codes for the Representation of Names of Languages*, [www-dokumentti] Saatavilla: http://www.loc.gov/standards/iso639-2/php/code_list.php (Luettu: 26.4.2010)

LIITTEET

LIITE 1 Ohjelman kielitiedostojen XML-skeema

LIITE 2 Ohjelman suomenkielinen kielitiedosto, katkelma

LIITE 3 Ohjelman käynnistysparametrien XML-skeema

LIITE 4 Ohjelman käynnistysparametrien tallennustiedosto, esimerkki

Ohjelman kielitiedostojen XML-skeema

```
<?xml version="1.0" encoding="UTF-8"?>

<schema xmlns:tns="http://www.khii.fi/xml/I18N"
xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.khii.fi/xml/I18N"
elementFormDefault="qualified">

<element name="langs" type="tns:LangType"></element>

<complexType name="LangType">
     <sequence>
       <element name="lang" type="string"></element>
       <element name="langcode" type="string"></element>
       <element name="author" type="string"></element>
       <element name="token" type="tns:MType"
       minOccurs="0" maxOccurs="unbounded"></element>
     </sequence>
</complexType>

<complexType name="MType">
     <sequence>
       <element name="key" type="string"></element>
       <element name="value" type="string"></element>
     </sequence>
</complexType>

</schema>
```

Ohjelman suomenkielinen kielitiedosto, katkelma

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<langs xmlns="http://www.khii.fi/xml/I18N"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.seitix.com/xml/I18N
../xml/i18n/I18N.xsd">

     <lang>Suomi</lang>
     <langcode>fi</langcode>
     <author>Jari Kärkkäinen</author>

     <token>
       <key>menufile</key>
       <value>Tiedosto</value>
     </token>

     <token>
       <key>menutools</key>
       <value>Työkalut</value>
     </token>

     <token>
       <key>open</key>
       <value>Avaa...</value>
     </token>

     <token>
       <key>new</key>
       <value>Uusi</value>
     </token>
```

Ohjelman käynnistysparametrien xml-skeema.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.khii.fi/xml/Survey/ini"
xmlns:tns="http://www.khii.fi/xml/Survey/ini"
elementFormDefault="qualified">

<element name="ini" type="tns:IniType"></element>

<complexType name="IniType">
     <sequence>
       <element name="lang" type="string"></element>
       <element name="lastFolder" type="string"></element>
       <element name="lastFile" type="string"></element>
       <element name="windowHeight" type="int"
       default="400"></element>
       <element name="windowWidth" type="int"
       default="760"></element>
       <element name="windowX" type="int" default="10"></element>
       <element name="windowY" type="int" default="10"></element>
       <element name="dividerLocation" type="int"
       default="150"></element>
       <element name="openLastEdited" type="boolean"
       default="false"></element>
     </sequence>
</complexType>

</schema>
```

Ohjelman käynnistysparametrien tallennustiedosto, esimerkki.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<ini xmlns="http://www.khii.fi/xml/Survey/ini"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.khii.fi/xml/Survey/ini
../xml/ini/ini.xsd">

     <lang>Suomi</lang>

     <lastFolder></lastFolder>

     <lastFile></lastFile>

     <windowHeight>566</windowHeight>

     <windowWidth>846</windowWidth>

     <windowX>245</windowX>

     <windowY>99</windowY>

     <dividerLocation>203</dividerLocation>

     <openLastEdited>false</openLastEdited>

</ini>
```