Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Ohjelmistotuotanto
Leena Uoti

Opinnäytetyö

Mobiilipelien merkkijonojen tarkistussovellus

Työn ohjaaja FM Ville Haapakangas
Työn tilaaja Universomo
Tampere 02/2009

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma, Ohjelmistotuotanto

Tekijä	Leena Uoti
Työn nimi	Mobiilipelien merkkijonojen tarkistussovellus
Sivumäärä	34
Valmistumisaika	Helmikuu 2009
Työn ohjaaja	FM Ville Haapakangas
Työn tilaaja	Universomo

TIIVISTELMÄ

Opinnäytetyön lähtökohtana oli ohjelmoida sovellus, joka auttaa mobiilipelien tekstien käsittelyä käännösvaiheessa ja nopeuttaa käännöksien integrointia peliprojektiin. Työn toimeksiantajana toimi tamperelainen mobiilipelistudio Universomo.

Universomossa mobiilipelien tekstit kirjoitetaan ensin englanniksi, josta ne käännätetään muille kielille. Sovellus generoi englanninkielisistä teksteistä Microsoft Office Excel -taulukon käännöstoimistolle lähetettäväksi. Kun taulukko tulee käännöstoimistolta täytettynä takaisin, se syötetään sovellukselle, joka palauttaa käännökset oikeille paikoilleen. Palautuksen yhteydessä sovellus suorittaa käännöksille tarkastuksia.

Sovellus toteutettiin Java-ohjelmointikielellä. Koska Java ei sisällä sisäänrakennettua tukea Microsoft Officen käyttämille tiedostoformaateille, työssä valittiin käytettäväksi Apache POI -rajapinta mahdollistamaan Excel-tiedostojen ohjelmoimisen.

Opinnäytetyössä käsitellään Microsoft Office Excel -tiedostojen ohjelmoimista Javalla, ja siitä annetaan esimerkkejä sekä kerrotaan vaihtoehtoisista rajapinnoista. Sovelluksen toiminnallisuutta sekä sen testausta kuvataan omissa luvuissaan.

Opinnäytetyön tuloksena on syntynyt toimeksiannon mukainen sovellus. Sovellus on integroitu yrityksessä käytettävään alustaan ja otettu yrityksessä tuotantokäyttöön syksyllä 2008.

Avainsanat: mobiilipelit, merkkijonot, Java, Excel

TAMK University of Applied Sciences
Business Information Systems, Software Engineering

Writer	Leena Uoti
Thesis	Application for Checking Translated Mobile Game Strings
Pages	34
Graduation time	February 2009
Thesis Supervisor	M.Sc. Ville Haapakangas
Co-Operating Company	Universomo

ABSTRACT

The goal of this thesis was to implement an application for handling localization of strings in a mobile game development. The co-operating company and the client of the thesis was Universomo, a Finnish mobile game studio located in Tampere.

In Universomo, the strings of the mobile game are written in English during the production stage. After that, the strings are translated to other languages in a translation agency. The application created in this thesis generates a Microsoft Office Excel workbook which includes the English texts, and the workbook is then sent to the translation agency. After receiving the translations, the application reads through the file and saves the translations to their correct places. While going through the translated strings, the application also performs different checks for the strings.

The application was programmed with Java. As Java does not include support for Microsoft Office file formats, Apache POI API was used to enable the reading and writing of the Excel files with Java.

This thesis deals with different solutions for handling Excel sheets with Java and gives simple programming examples as well. It also covers the application itself and how it works.

A functional application was created as a result of the thesis. The application has been integrated into the company's platform, and has been taken into production use in fall 2008.

Key words: mobile games, strings, Java, Excel

Sisällysluettelo

1 Johdanto

Vuodesta 2007 eteenpäin mobiilipelimarkkinoiden voiton ennustetaan kasvavan yli 10 prosentin vuosivauhdilla vuoteen 2011 asti. IT-alan tutkimusyritys Gartner kertoo mobiilipelimarkkinoiden voiton ylittävän jo 4,5 miljardia dollaria vuonna 2008. (Gartner 2008)

Mobiilipelien tullessa yhä suosituimmiksi, kasvaa myös tarve niiden lokalisointiin. Videopelien lokalisointi tarkoittaa sitä, että pelistä tehdään eri versiot eri maiden markkinoille. Maakohtaisissa versioissa otetaan huomioon kohdemaiden tavat ja kulttuurierot. Yleensä lokalisointi kattaa pelin tekstit ja puhutut ääniraidat, mutta se saattaa ulottua myös pelissä käytettäviin grafiikoihin ja musiikkiin asti.

Mobiilipelien kohdalla lokalisointi tarkoittaa lähinnä pelin tekstien kääntämistä eri kohdemaiden kielille. Tässä työssä mobiilipelin teksteillä tarkoitetaan kaikkia pelissä näkyviä merkkijonoja. Itse pelin tarinaan liittyvien tekstien lisäksi peliin sisältyvät mm. ohjetekstit, mahdolliset lakitekstit, tekijätiedot (Credits-tekstit) ja softkey-näppäinten tekstit.

Opinnäytetyön toimeksiannon tavoitteena oli ohjelmoida toimeksiantajalle sovellus, joka helpottaa pelin merkkijonojen käsittelyä käännösprossesin aikana. Tässä opinnäytetyössä kerrotaan kyseisen sovelluksen teossa käytetyistä tekniikoista sekä kuvataan sovelluksen toiminnallisuutta. Lisäksi kerrotaan kehitystyössä kohdatuista ongelmista ja sovelluksen testauksesta.

1.1 Toimeksiantaja

Opinnäytetyön toimeksiantaja Universomo on vuonna 2002 perustettu tamperelainen mobiilipelistudio. Vuodesta 2007 lähtien yritys on ollut amerikkalaisen pelijulkaisijan THQ:n tytäryhtiö. Universomossa työskentelee kaikkiaan n. 60 työntekijää, ja sillä on toimipaikat Tampereella ja Helsingissä.

Universomo kehittää mobiilipelejä Java-, BREW- ja iPhone -alustoille. Yritys on kehittänyt useita palkittuja mobiilipelejä, kuten Star Wars: The Force Unleashed, Puzzle Quest: Warlords ja 300: The Mobile Game (Universomo 2008).

1.2 Sovelluksen tausta ja tarve

Universomon pelit sisältävät yleensä käännökset viidelle kielelle. Tämä tarkoittaa sitä, että käyttäjä voi pelin Options-valikosta valita haluamansa pelikielen. Eurooppaan toimitettavissa peleissä kielivaihtoehtoina ovat englanti, saksa, espanja, ranska ja italia. Itä-Euroopassa myytävissä peleissä kielinä ovat englannin lisäksi venäjä, turkki, puola ja tsekki. Lisäksi englanninkieliset tekstit lokalisoidaan Amerikkaan ja Iso-Britanniaan erikseen, sillä sanojen kirjoitusasu voi vaihdella.

Tuotantovaiheessa pelin tekstit kirjoitetaan ensin englanniksi. Kun tekstit ovat valmiina, ne lähetetään käännöstoimistolle käännettäviksi kahdeksalle muulle kielelle. Opinnäytetyönä tehty sovellus liittyy tähän tekstien käännösvaiheeseen. Sovellusta käytetään, kun

- tekstit lähetetään käännettäväksi
- saadut käännökset integroidaan peliin.

Yrityksen käyttämässä alustassa pelien merkkijonot tallennetaan xml-tiedostoihin. Sovellus kokoaa näistä tiedostoista käännöstoimistolle lähtevän Excel-taulukon. Taulukko kootaan tietyistä attribuuteista, kuten tunnus, maksimipituus ja itse käännettävä merkkijono. Taulukkoon on varattu jokaiselle merkkijonolle oma kielisarake, johon käännöstoimisto täyttää oikean käännöksen.

Kun Excel-taulukko tulee täytettynä käännöstoimistolta takaisin, se syötetään sovellukselle, ja sovellus palauttaa käännökset takaisin alustaan oikeille paikoilleen. Samalla merkkijonot muutetaan takaisin alustan käyttämiin xml-muotoihin. Palautuksen yhteydessä sovellus suorittaa käännetyille merkkijonoille tarkistuksia, jotka helpottavat ja

nopeuttavat käännösten integrointia projektiin. Tämä säästää aikaa, sillä pelin merkkijonoja ennen ja jälkeen käännösten ei tarvitse enää verrata manuaalisesti toisiinsa.

1.3 Sovelluksen toteutustapa

Sovellus toteutettiin Java-ohjelmointikielellä, joka on Sun Microsystemsin kehittämä laitteistoriippumaton ohjelmointikieli. Sovellus integroitiin yrityksessä käytettävään ohjelmointiympäristöön, jonka kautta sitä käytetään. Sovellus on siis ns. ohjelmalisäke (plug-in).

Microsoft Excel -taulukoiden käsittelyä varten tarvittiin rajapinta, sillä Java ei tue Excelin tiedostomuotoja. Käytettäväksi valittiin Apache POI -rajapintakokoelma, jonka sisältämä HSSF-rajapinta on tarkoitettu Excel-tiedostojen käsittelyyn. Sovelluksen ohjelmoinnissa käytettiin Apache POI:n versiota 2.5.1.

Seuraavassa luvussa kerrotaan tarkemmin eri Excel-tiedostojen käsittelyyn tarkoitetuista rajapintavaihtoehdoista, sekä annetaan perustelut valitun rajapinnan käyttämiseen. HSSF-rajapinnasta kertovassa luvussa annetaan myös ohjelmointiesimerkkejä valitun rajapinnan toiminnasta.

2 Microsoft Office Excel -tiedostot ja Java-ohjelmointi

Microsoft Office on toimisto-ohjelmistopaketti, johon kuuluu mm. Word-tekstinkäsittelyohjelma, Excel-taulukkolaskentaohjelma ja PowerPoint-esitysgrafiikkaohjelma. Nämä ohjelmat käyttävät OLE2 Compound Document -formaattia, joka on Microsoftin kehittämä geneerinen tiedostoformaatti. (PRONOM The Technical Registry - OLE2 Compound Document 2008.)

Java-ohjelmointikielessä ei ole sisäänrakennettua tukea Microsoft Officen tiedostoformaateille. Koska Java ei tue tätä formaattia, sille on kehitetty erillisiä rajapintoja, jotka mahdollistavat OLE2 Compound Document -tiedostojen käsittelyn Java-sovelluksissa.

2.1 Rajapintavaihtoehtoja

Tässä luvussa esitellään kolme rajapintaa, jotka mahdollistavat Microsoft Excel-tiedostojen käsittelyn Java-ohjelmoinnissa. Nämä rajapinnat ovat Apache POI -rajapintakokoelma, JExcel ja JExcelApi. Tässä esiteltyjen vaihtoehtojen lisäksi on olemassa myös muitakin Java-Excel -rajapintoja, mutta niitä ei otettu lähempään tarkasteluun tätä sovellusta tehtäessä.

2.1.1 The Apache POI Project

Apache POI on Java-ohjelmointikielelle kehitetty rajapintakokoelma, joka mahdollistaa Microsoftin OLE2 Compound Document -formaattien käsittelyn Java-sovelluksissa. POI on yksi Apache-ohjelmistosäätiön (*The Apache Software Foundation*) alla toimivia projekteja. Säätiö tukee avoimen lähdekoodin ohjelmistoprojekteja ja niiden kehitystä. Vuodesta 2007 lähtien POI on ollut oma itsenäinen aliprojektinsa Apachen alla. (The Apache Poi Project 2008.)

Apache POI jakaantuu useaan mielenkiintoisesti nimettyyn aliprojektiin, joita ovat mm. seuraavat:

- *Horrible Word Processor Format* (HWPF), joka tarjoaa rajapinnan Microsoft Word -dokumenttien lukemiseen ja kirjoittamiseen.

- *Horrible Slide Layout Format* (HSLF), joka on rajapinta Microsoft PowerPoint -diaesitysten käsittelyyn.
- *Horrible Spread Sheet Format* -rajapinta (HSSF), joka on tarkoitettu Microsoft Excel -taulukkojen lukemiseen ja kirjoittamiseen.

(The Apache Poi Project 2008: Overview.)

Apache POI on avoimen lähdekoodin projekti, joka on ladattavissa ilmaiseksi projektin kotisivuilta. Rajapintakokoelma kuuluu Apache Licence -lisenssin alle. Lisenssi sallii Apachen ohjelmiston käyttämisen henkilökohtaisiin tai kaupallisiin tarkoituksiin. Se sallii myös Apachen ohjelmiston käyttämisen jaettavassa tuotteessa tai paketissa. Lisenssistä tulee kuitenkin liittää kopio mukaan kaikkiin uudelleenjaettaviin tuotteisiin, jotka sisältävät Apachen ohjelmiston.
(The Apache Software Foundation: Apache Licence and Distribution FAQ)

2.1.2 JExcel

JExcel on TeamDevin kehittämä kaupallinen kirjasto, joka tarjoaa rajapinnan Excel-tiedostojen käsittelyyn Java-ohjelmoinnissa. JExcelin avulla voidaan kirjoittaa ja lukea Excel-tiedostoja, mutta sen avulla voi myös sisällyttää Excel-työkirjan Java Swing -applikaatioon. Kuten HSSF, myös JExcel perustuu olio-ohjelmointiin ja rajapinnan rakenne sisältää mm. luokat Workbook ja Worksheet (JExcel Programmer’s Guide 2008).

JExcelin pääominaisuuksia ovat:

- Excel-tiedostojen hallinnointi; työkirjoja ja niihin liittyvää tietoa voi lukea, kirjoittaa ja muokata.
- Kuuntelijoiden asettaminen; työkirjoille voi asettaa kuuntelijoita, jolloin tiedostoon, työkirjaan tai välilehteen tehdyistä muutoksista lähetetään ilmoitus. Ilmoitus voidaan lähettää mm. seuraavista tapahtumista: uusi työkirja luodaan tai avataan, uusi taulukko-välilehti lisätään työkirjaan, työkirja tallennetaan tai suljetaan, sekä jos yksittäinen solu tai useampia soluja valitaan.
- Työkirjan voi sisällyttää Java Swing -applikaatioon aivan kuten minkä tahansa muun Swing-komponentin.

(TeamDev - Java & Microsoft Excel.)

JExcel on maksullinen tuote ja sen hinta määräytyy sen mukaan, ostaako lisenssin yksityinen kehittäjä (Personal licence) vai yritys (Standard licence). Yksityinen kehittäjä voi ostaa lisenssin itselleen silloin, kun ei tee kehitystyötä millekään yritykselle. Yrityksen tulee hankkia yrityslisenssejä yksi jokaista JExceliä käyttävää työntekijää kohden. (TeamDev – JExcel Purchase.)

2.1.3 JExcelApi

Samankaltaisesta nimestä huolimatta JExcelApia ei pidä sekoittaa TeamDevin JExceliin. JExcelApi on vapaan lähdekoodin rajapinta Excel-tiedostojen käsittelyyn Javalla, ja se on saatavilla ilmaiseksi kuten Apache POI -projektikin. JExcelApi käyttää GNU Lesser General Public Licence (LGPL) -lisenssiä.

JExcelApin avulla voi lukea kaikkia Excel 95 – 2003 -tiedostoja. Rajapinta tukee kaavioiden kirjoittamista ja lukemista Excel 97 -versiosta eteenpäin. JExcelApin avulla voi myös muotoilla taulukon ulkoasua sekä liittää ja kopioida taulukkoon kuvia. Kaikkia ominaisuuksia rajapinta ei tue yhtä laajasti, sillä rajapinnan avulla pystyy luomaan uusia taulukko-välilehtiä vain Excel 2000 -tiedostoissa. (JExcelApi 2008.)

JExcelApin kotisivuilta (http://jexcelapi.sourceforge.net/) ei käy ilmi rajapinnan kehittäjää. Käyttäjillä on käyttäjäryhmä Yahoo! Groups -sivustolla, mutta sitä lukeakseen tulee rekisteröityä Yahoo!:n palveluun. Osa informaatiosta, kuten vastauksia joihinkin käyttäjien esittämiin kysymyksiin, on ainoastaan tällä käyttäjäryhmän sivustolla. Rekisteröityminen on ilmaista, mutta sama sisältö olisi hyvä esittää myös suoraan rajapinnan kotisivuilla, sillä sieltä käyttäjät tietoa ensimmäiseksi etsivät.

2.2 Perusteluja rajapinnan valintaan

Vaatimuksena käytettävälle rajapinnalle oli, että sen avulla tulee pystyä lukemaan ja kirjoittamaan Excel-tiedostojen sisältöä, sekä luomaan uusia Excel-tiedostoja. Tiedoston taulukkovälilehtien ulkoasua tulee myös pystyä muotoilemaan. Sovelluksen toteutukseen valittiin käytettäväksi Apache POI -rajapintakokoelma, joka sisältää Excel-tiedostojen käsittelyyn tarkoitetun HSSF-rajapinnan. Syitä tämän rajapinnan valintaan olivat seuraavat:

- JExcelApi pystyy luomaan uusia taulukkovälilehtiä ainoastaan Excel 2000 -tiedostomuodossa, kun taas Apache POI:n HSSF-rajapinta pystyy käsittelemään kaikkia Excelin '97 – 2003 -versioita. Koska uuden taulukon luominen on sovelluksessa keskeinen toiminto, on HSSF näistä parempi vaihtoehto. JExcelin kotisivuilta ja dokumentaatiosta ei löytynyt tietoja rajapinnan tukemista Excel-versioista.
- Kaikkien kolmen rajapinnan dokumentaatio löytyy lähinnä ainoastaan internetistä. Kaikkiin löytyy Javadoc-dokumentaatio rajapintojen kotisivuilta. Apache POI:n ja JExcelin kotisivut tarjoavat myös useita esimerkkejä rajapinnan käyttöön. JExcelApin kotisivut sisältävät myös esimerkkejä, mutta ne eivät ole yhtä kattavia ja selkeitä.
- JExcel on näistä kolmesta rajapintavaihtoehdoista ainoa maksullinen. Apache POI ja JExcelApi ovat molemmat avoimen lähdekoodin tuotteita ja ladattavissa ilmaiseksi. Koska Apache POI sisältää sovelluksen teossa tarvittavat ominaisuudet, ei ole syytä maksaa JExcelin lisenssistä, kun tarvittavan työkalun saa ilmaiseksikin.

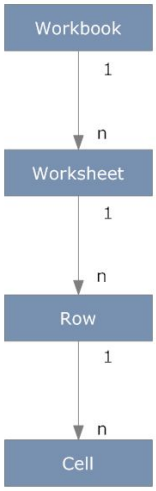
Yksi tärkeä Apache POI:n valintaa puoltava syy oli myös se, että se oli tekijälle jo etukäteen tuttu. Vaikka sovellusta ohjelmoidessa tuli vastaan uusia asioita, ei rajapinnan käytön ja rakenteen opetteluun ei kulunut ylimääräistä aikaa, koska perusasiat olivat jo hallussa.

3 Sovelluksessa käytetty HSSF-rajapinta

HSSF on yksi Apache POI:hin sisältyvistä rajapinnoista. HSSF tukee Microsoft Excelin ’97 (– 2003) tiedostomuotoa. Vuonna 2007 julkaistu viimeisin Office-paketti käyttää uutta Office Open XML (OOXML) -tiedostomuotoa. Uusimman Excelin *.xlsx* -tiedostomuoto ei siis enää perustu OLE2-formaattiin. Opinnäytetyö-sovelluksen ohjelmoinnin aikaan HSSF ei tukenut tätä uutta OOXML-tiedostomuotoa, mutta rajapintaa kehitetään koko ajan, ja tuki myös uusimpiin Excel-tiedostomuotoihin on tulossa. (The Apache Poi Project 2008: POI-HSSF.)

HSSF:n käyttö on helppo omaksua, jos olio-ohjelmointi on tuttua. Oliokielissä, kuten Javassa, olioiden piirteet määritellään olion luokassa. Jokaisella oliolla on oma luokkansa, jonka mukaan olio on luotu. Olio on tämän luokan ilmentymä eli instanssi. Luokka määrittelee, mitä attribuutteja ja operaatioita sen ilmentymillä on. Tämä tarkoittaa, että jokaisella luokan ilmentymällä on samat attribuutit ja operaatiot, mutta ainoastaan attribuuttien arvot vaihtelevat. (Koskimies 2000: 37.)

HSSF-rajapinnassa luokkina ovat mm. HSSFWorkbook, HSSFSheet, HSSFRow ja HSSFCell. Workbook viittaa työkirjaan, joka koostuu yhdestä tai useammasta taulukkovälilehdestä (Worksheet). Taulukossa voi olla useita rivejä (Row) sekä riveillä soluja (Cell). Luokkien suhteet näkyvät kuviosta 1.



Kuvio 1: HSSF-luokkarakenne Guptaa (2003) mukaellen

3.1 Taulukoiden lukeminen

Kun Excel-taulukoita halutaan lukea, täytyy POIFSFileSystem-luokasta luoda uusi olio. POIFSFileSystem-luokan rakentajaan tulee parametrina InputStream-luokan instanssi. Abstraktin InputStream-luokan aliluokasta FileInputStream luodaan instanssi, jonka rakentajaan annetaan luettava tiedosto. Tämän jälkeen FileInputStream-luokan olio välitetään parametrina POIFSFileSystem-luokan rakentajalle.

HSSFWorkbook-luokasta luodaan uusi instanssi, jonka rakentajaan välitetään parametrina juuri luotu POIFSFilesystem-luokan olio. HSSFWorkbook-luokan olion kautta voidaan hakea haluttu taulukko-välilehti, ja vastaavasti taulukon kautta haluttu rivi. Kaikkia taulukon tietoja voidaan hakea käyttämällä luokkien aksessorimetodeita, kuten workbook.getSheet(sheetNum) tai sheet.getRow(rownum). Kuviossa 2 taulukosta luetaan cellText-muuttujaan yhden solun sisältämä teksti. (The Apache Poi Project 2008: The New Halloween Document.)

```
// luodaan FileInputStream-luokan instanssi
FileInputStream file = new FileInputStream("laskut.xls")

// sijoitetaan luotu olio POIFSFileSystem-luokan olion
// konstruktoriin
POIFSFileSystem filesystem = new POIFSFileSystem(file);

// luodaan HSSFWorkbook-luokan instanssi
HSSFWorkbook workbook = new HSSFWorkbook(filesystem);

// haetaan Workbookista haluttu taulukko-välilehti,
// sheetNum viittaa taulukko-välilehden indeksiin
HSSFSheet sheet = workbook.getSheet(sheetNum);

// haetaan taulukosta haluttu rivi, rowNum viittaa rivin
// numeroon.
HSSFRow row = sheet.getRow(rowNum);

// haetaan riviltä haluttu solu, columnNum viittaa
// sarakkeen numeroon
HSSFCell cell = row.getCell(columnNum);

// haetaan cellText-muuttujaan solun sisältämä teksti
String cellText = cell.getStringCellValue();
```

Kuvio 2: Excel-taulukon lukeminen HSSF:n avulla

Tietovirran käsittely voi aiheuttaa IOException-poikkeuksen, johon täytyy varautua poikkeuskäsittelyllä. Selvyyden vuoksi poikkeuskäsittely on jätetty esimerkistä pois.

3.2 Taulukoiden luominen ja muokkaaminen

Excel-taulukon luominen aloitetaan luomalla uusi HSSFWorkbook-luokan olio. Uusi taulukko-välilehti voidaan luoda ainoastaan työkirjan kautta, ja vastaavasti solua ei voi luoda, ennen kuin on luotu rivi. Kuviossa 3 luodaan uusi taulukko, jonka yhteen soluun kirjoitetaan teksti ”Otsikko”.

```
//luodaan Workbook-luokan olio
HSSFWorkbook workbook = new HSSFWorkbook();

// luodaan uusi taulukko-välilehti nimeltään Taulukko 1
HSSFSheet sheet = workbook.createSheet("Taulukko 1");

// luodaan uusi rivi, parametrinä annetaan rivin indeksi
HSSFRow row = sheet.createRow(0)

// luodaan äsken luodulle riville solu, parametrinä
// annetaan halutun sarakkeen indeksi
HSSFCell cell = row.createCell(0);

// lisätään soluun haluttu teksti
cell.setCellValue("Otsikko");

// luodaan uusi tiedosto ja kirjoitetaan tiedostoon
FileOutputStream out = new FileOutputStream("tiedosto.xls");

// suljetaan tietovirta
workbook.write(out);
out.close();
```

Kuvio 3: Excel-taulukon luominen HSSF:n avulla

Excel-taulukoiden muokkaaminen onnistuu yhdistämällä taulukon lukeminen ja sen luominen. Ensin luetaan muokattava taulukko, minkä jälkeen taulukon tietoja voi poistaa ja kirjoittaa halutessaan uudet arvot tilalle.

3.3 Taulukon solujen muotoilu

HSSF sisältää luokan HSSFCellStyle, joka määrittää solujen muotoilun eli tyylin. Tyyli sisältää esimerkiksi tekstin fontin ja siihen liittyvä muotoilut, kuten fontin koon, värin ja tekstin tasauksen. Myös solun reunaviivat määritellään tyylissä. Yksi HSSFCellStyle-luokan olio sisältää yhden tyylin, jota voi käyttää useissa soluissa. Yhdessä työkirjassa voi olla eri soluille useita tyylejä, eli useita HSSFCellStyle-olioita. (The Apache Poi Project 2008. The New Halloween Document)

Uusi tyyli luodaan HSSFWorkbook-olion kautta, ja tällöin tyyli tallentuu sen työkirjan yhteydessä käytettäväksi. Kuviossa 4 luodaan HSSFCellStyle-luokan instanssi, johon asetetaan fontti sekä solun yläreunan viiva. Lopuksi aiemmin luotu solu asetetaan käyttämään tätä tyyliä.

```
// luodaan uusi HSSFCellStyle-olio
HSSFCellStyle style = workbook.createCellStyle();

// luodaan uusi fontti
Font font = workbook.createFont();

// asetetaan fontin kooksi 12
font.setFontHeightInPoints((short) 12);

// asetetaan fontti kursiiviksi
font.setItalic(true);

// liitetään fontti tyyliin
style.setFont(font);

// asetetaan tyyliä käyttävien solujen yläreunaan viiva
style.setBorderTop(style.BORDER_THIN);

// asetetaan solulle tyyli
cell.setCellStyle(style);
```

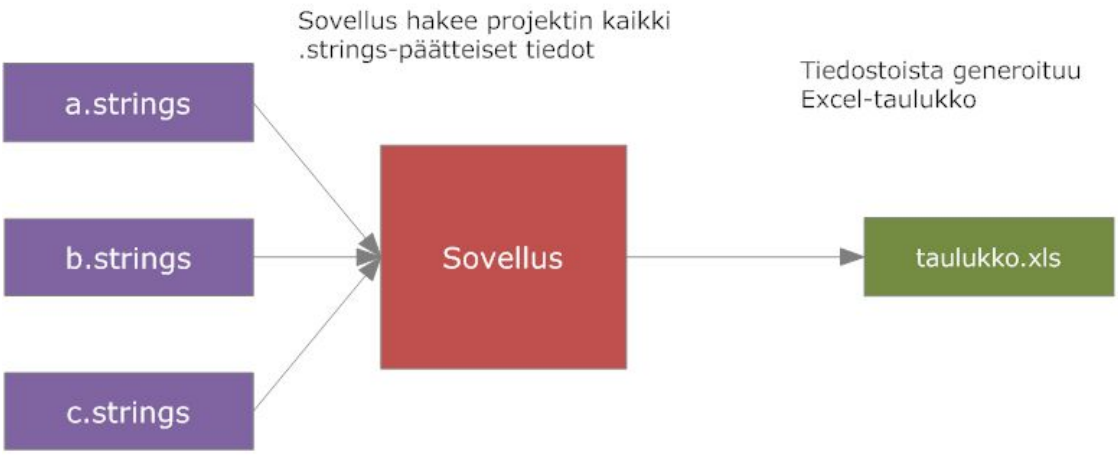
Kuvio 4: Solun muotoilu HSSF:n avulla

4 Sovelluksen toiminta

Sovelluksen toiminta on tässä työssä jaettu kahteen pääosaan, Export- ja Import-toimintoihin. Export-toimintoon liittyy sovelluksen toiminnallisuus siihen asti, kunnes merkkijonotaulukko lähetetään käännöstoimistolle käännettäväksi. Import-toiminto kattaa käännösten integroinnin takaisin projektiin sekä samalla suoritettavat tarkistukset. Seuraavissa aliluvuissa kerrotaan molemmista toiminnoista tarkemmin sekä kuvataan sovelluksen toimintaa esimerkkikäännösten avulla.

4.1 Export-toiminto

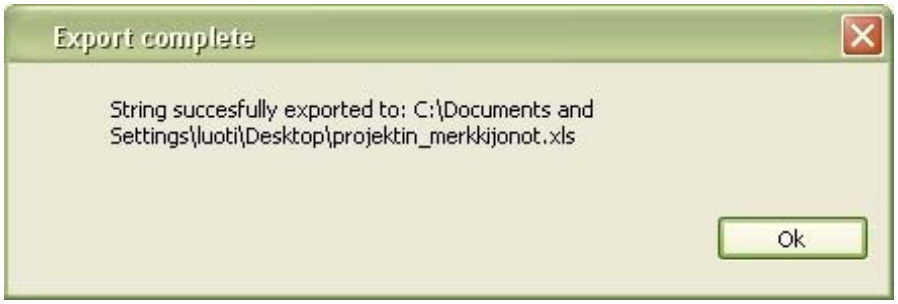
Pelin merkkijonot tallennetaan alustaan .strings-päätteisiin xml-tiedostoihin. Yksi tiedosto sisältää useita samaan aihealueeseen liittyviä merkkijonoja. Näiden tiedostojen merkkijonoista käyttäjä voi Export-toiminnon avulla helposti luoda käännöstoimistolle lähtevän Excel-taulukon. Kuvio 5 havainnollistaa Export-toimintoa.



Kuvio 5: Export-toiminnon kuvaus

4.1.1 Taulukon generointi

Kun käyttäjä painaa Export-painiketta, sovellus etsii kaikki projektiin kuuluvat .strings-päätteiset tiedostot. Näistä xml-tiedostoista kootaan käännöstoimistolle lähtevä Excel-taulukko. Sovellus kysyy käyttäjältä, minne tiedosto tallennetaan, ja käyttäjä voi nimetä tiedoston haluamallaan tavalla. Kun merkkijonotaulukko on onnistuneesti luotu, sovellus ilmoittaa sitä käyttäjälle vielä erikseen, kuten kuviossa 6.



Kuvio 6: Sovellus ilmoittaa merkkijonotaulukon luomisesta.

Taulukkoa generoidessaan sovellus hakee xml-tiedostoista merkkijonot ja niitä kuvailevat attribuutit. Taulukkoon kirjattaviin attribuutteihin kuuluvat merkkijonoa kuvaileva id-teksti, kommentti merkkijonosta, sekä tieto siitä, kuinka monta merkkiä pitkä käännös saa maksimissaan olla. Info-sarakkeiden jälkeen englanninkieliset merkkijonot sijoitetaan taulukkoon ja seuraaviin sarakkeisiin varataan tilat muiden kielten käännöksiä varten. Excel-taulukkoon lisättävät attribuutit näkyvät taulukosta 1.

Attribuutti	**Kuvaus**
ID Text	Symboli merkkijonolle, esim "LEVEL_COMPLETED_TEXT"
Comment	Merkkijonon kuvaus
Max. Length	Merkkijonolle asetettu maksimipituus
en, de, es, fr, it etc.	Itse tekstiresurssi eri kielillä. Merkkijonon jokainen käännös on oma attribuuttinsa.

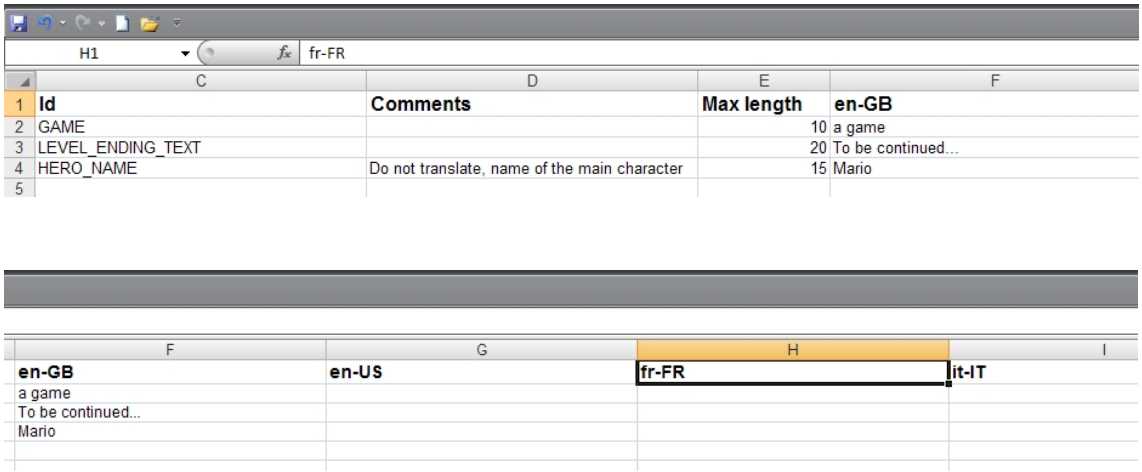
Taulukko 1: Excel-taulukkoon kirjattavat tiedot

Kommentti-sarake voi sisältää merkkijonosta tarkempaa tietoa käännöstoimistolle. Merkkijonojen maksimipituus on myös tärkeä, koska matkapuhelimien näytöissä on

rajallinen määrä tilaa. Jos käännökset ovat liian pitkät, ne eivät tällöin näkyisi pieninäyttöisissä puhelimissa kokonaan.

Esimerkki

Pelissä on kolme merkkijonoa, jotka tulee käännättää. Merkkijonot ovat *"a game"*, *"To be continued..."* ja *"Mario"*. Export-toiminto generoi Excel-taulukon, jossa näkyy jokaisen merkkijonon Id-teksti, merkkijonoon liittyvät kommentit ja merkkijonon maksimipituus. Maksimipituussarakkeen vieressä on merkkijonosarake, jossa on käännettävä teksti englanniksi. HERO_NAME -merkkijonolle on lisätty kommentti, että merkkijonoa *"Mario"* ei tarvitse kääntää muille kielille, sillä se on pelin päähahmon nimi. Kuviossa 7 on Excel-taulukko, jossa näkyvät eri informaatio-ja kielisarakkeet.



	C	D	E	F
1	**Id**	**Comments**	**Max length**	**en-GB**
2	GAME		10	a game
3	LEVEL_ENDING_TEXT		20	To be continued...
4	HERO_NAME	Do not translate, name of the main character	15	Mario
5				

F	G	H	I
en-GB	**en-US**	**fr-FR**	**it-IT**
a game			
To be continued...			
Mario			

Kuvio 7: Excel-taulukon id-, kommentti-, maksimipituus-sarakkeet sekä kielisarakkeita

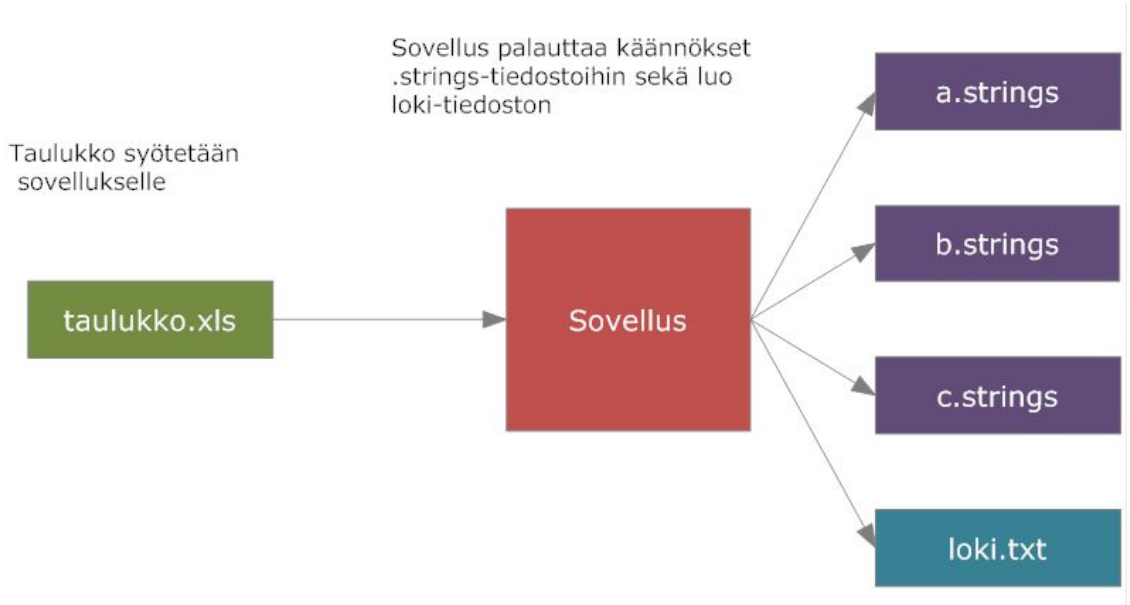
4.1.2 Hidden id -tunnus

Merkkijonojen käännökset palautetaan Import-toiminnolla takaisin oikeille paikoilleen alustaan. Alkuperäisillä merkkijonoilla täytyy siis olla jokin tunniste, jonka avulla käännökset ohjataan oikeille paikoilleen.

Jokainen merkkijono saa numeerisen Hidden id -tunnuksen automaattisesti, kun merkkijono lisätään alustaan. Tämä tunnus tulee myös mukaan Excel-taulukkoon, mutta Hidden id -sarake asetetaan piiloon, jotta se ei ole näkyvissä käännöstoimistolle. On tärkeää, ettei käännöstoimisto (tai kukaan muukaan) pääse muuttamaan tunnusta, jotta käännösten palautus onnistuu. Hidden id -tunnuksen lisäksi merkkijonon alkuperäinen tiedostopolku lisätään Excel-taulukkoon palautusta varten. Myös tiedostopolku on piilotettu, jotta sitä ei pääsisi kukaan vahingossa muuttamaan.

4.2 Import-toiminto

Import-toiminto palauttaa käännetyt merkkijonot oikeille paikoille alustaan, kuten kuvio 8 havainnollistaa. Palautuksen yhteydessä sovellus suorittaa käännöksille tarkistuksia, jotta käännöksiä ei tarvitse käydä manuaalisesti läpi.



Kuvio 8: Import-toiminnon kuvaus

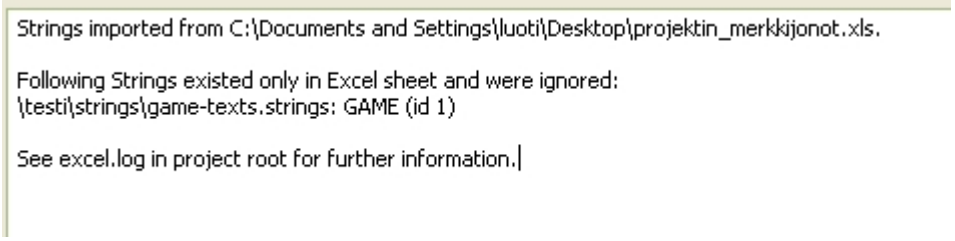
4.2.1 Sovelluksen tekemät tarkistukset

Muuttuneet merkkijonot

Palautuksen yhteydessä saattaa tulla vastaan tilanteita, jossa projektin merkkijonot ovat jo muuttuneet. Tämä tarkoittaa, että käännöstoimistolta saattaa tulla takaisin käännettynä merkkijonoja, joita ei ole enää olemassa alustassa. Pelistä on saatettu poistaa esimerkiksi jokin tarinaan liittyvä teksti. Sovellus vertaa alustassa ja taulukossa olevia merkkijonoja toisiinsa. Vertailuun käytetään merkkijonon alkuperäistä tiedostopolkua, id-tekstiä sekä Hidden id -tunnusta. Jos käännettyä merkkijonoa ei löydy enää palautushetkellä alustasta, tällöin käännöksiäkään ei palauteta alustaan. Sovellus ilmoittaa näiden merkkijonojen tiedot käyttäjälle.

Esimerkki

Pelin sanat *"a game"*, "*to be continued..."* ja "*Mario"* lähetetään käännöstoimistolle. Sinä aikana, kun tekstit ovat käännettävänä, pelistä päätetään poistaa "*a game"* -merkkijono. Kun Excel-taulukko käännöksineen syötetään sovellukselle, sovellus ilmoittaa, että merkkijono puuttuu alustasta. Poistetun "*a game"* -merkkijonon käännöksiä ei myöskään tallenneta alustaan. Ilmoituksessa kerrotaan merkkijonon tiedostopolku, Id-teksti sekä merkkijonon Hidden id, kuten kuviossa 9.

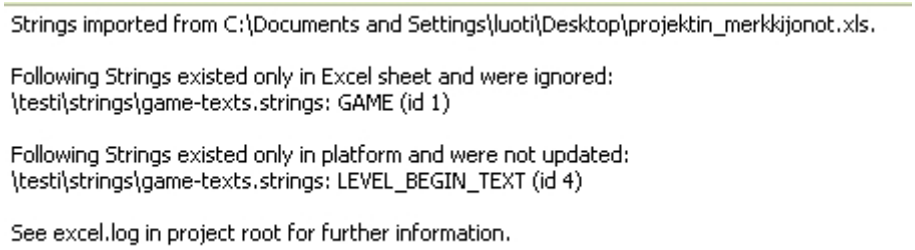


Kuvio 9: Sovellus ilmoittaa merkkijonoista, jotka on poistettu alustasta.

Projektiin on myös saatettu lisätä käännösprosessin aikana uusia merkkijonoja, joihin ei ole käännöstä tulossa. Myös näissä tapauksissa sovellus ilmoittaa käyttäjälle asiasta.

Esimerkki

Merkkijono *"a game"* oli poistettu projektista. Käännösprosessin aikana projektiin on sen lisäksi lisätty uusi merkkijono *"Let's save the princess"*, jonka Id-teksti on LEVEL_BEGIN_TEXT. Kun käännökset palaavat ja ne syötetään sovellukselle, sovellus ilmoittaa, ettei LEVEL_BEGIN_TEXT -merkkijonolle saapunut käännöstä. Käyttäjälle esitettävä ilmoitus näkyy kuviossa 10.



Kuvio 10: Sovellus ilmoittaa puuttuvista käännöksistä.

Liian pitkät käännökset

Mobiilipelien päätelaitteissa, eli kännyköissä, on nykyään hyvin vaihtelevat näytön koot. Pelin merkkijonoille määritellään maksimipituus, jotta tekstit mahtuisivat niille varattuun tilaan pelin käyttöliittymässä. Sovellus vertaa Max length -sarakkeessa määritettyä lukua jokaisen merkkijonon käännöksiin. Jos käännös on liian pitkä, sovellus ilmoittaa siitä käyttäjälle. Käännös kuitenkin tallennetaan alustaan, eli jää käyttäjän vastuulle tarkastaa merkkijono. Tarvittaessa käännöstoimistolta tulee pyytää uusi lyhyempi käännös kyseisestä merkkijonosta.

Numeeriset arvot

Pelin teksteihin saattaa kuulua myös merkkijonoja, jotka koostuvat kokonaan numeroista. Excelin solun desimaaliasetukset saattavat aiheuttaa sen, että esimerkiksi soluun kirjoitettu luku ”0000” muuttuukin luvuksi ”0” kun käyttäjä painaa enteriä. Excel saattaa asetuksista riippuen myös pyöristää desimaalilukuja lähimpään kokonaislukuun.

Tämä voi muodostua ongelmaksi, sillä pelin merkkijonoissa saatetaan tarvita nimenomaan tietyllä tavalla muotoiltua lukua. Jotta numeeristen solujen sisältö voidaan varmistaa oikeaksi, pitää käyttäjän tarkistaa käännöksestä tulevat numeeriset merkkijonot. Jotta tarkistaminen sujuisi nopeasti, sovellus ilmoittaa, mitkä saapuvista merkkijonoista ovat kokonaan numeerisia.

Kielletyt merkit

Käännöstoimistolta tulevissa käännetyissä merkkijonoissa saattaa olla mukana myös erikoismerkkejä. Pelin merkkijonot tulee tarkistaa näiden erikoismerkkien varalta, koska pelin fontteihin ei ole sisällytetty kaikkia mahdollisia merkkejä. Jos pelin tekstit sisältävät erikoisen merkin, jota ei ole määritelty fonteissa, ei tämä merkki näy oikein pelin sisällä.

Sovellukseen pystyy tarpeen mukaan itse määrittelemään mahdolliset kielletyt merkit ja niitä korvaavat sallitut merkit. Kiellettyihin erikoismerkkeihin kuuluu mm. Ellipsismerkki. Ellipsis on merkki, joka näyttää kolmelta pisteeltä, mutta oikeasti nämä kolme pistettä muodostavat vain yhden merkin. Taulukossa 2 on esimerkit kahdesta kielletystä merkistä, sekä näiden merkkien sallittu muoto.

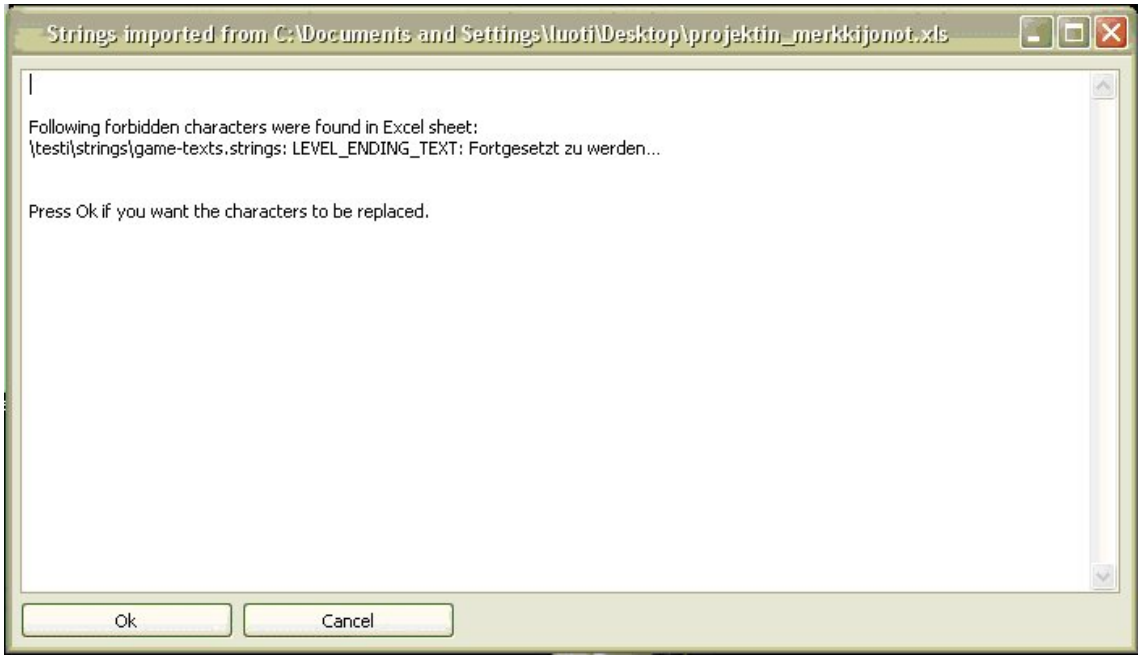
Merkin nimi	**Kielletty**	**Muutettu**
Tavaramerkki	™	(tm)
Ellipsis	… (\u2026)	... (kolme tavallista pistettä)

Taulukko 2: Esimerkit kielletyistä merkeistä

Kun käännökset sisältävä Excel-tiedosto syötetään sovellukselle, sovellus tarkistaa löytyykö käännetyistä merkkijonoista kielletyksi määriteltyjä merkkejä. Jos kiellettyjä merkkejä löytyy, sovellus ilmoittaa merkeistä ja pyytää käyttäjältä vahvistuksen niiden korvaamiseen.

Esimerkki

Saksankielisistä käännöksistä löytyy ellipsis-merkki. Sovellus ilmoittaa merkkijonon Id-tekstin sekä sisällön, kuten kuviossa 11.



Kuvio 11: Sovellus ilmoittaa löytyneistä kielletyistä merkeistä

Jos käyttäjä antaa luvan, sovellus korvaa kielletyt merkit oikeilla. Esimerkissä Ellipsis-merkki korvataan siis kolmella tavallisella pisteellä. Käyttäjä voi myös peruuttaa Import-toiminnon valitsemalla “Cancel”.

4.2.2 Loki-tiedosto

Sovelluksen toteutuksen yhteydessä tuli ilmi, että olisi hyvä tallentaa käyttäjälle näytettävät ilmoitukset vielä erilliseen tiedostoon, josta käyttäjä voi tarkistaa ne myöhemminkin. Sovellukseen toteutettiin toiminto, mikä tallentaa palautuksen yhteydessä annetut virheilmoitukset aina erilliseen tekstitiedostoon. Loki-tiedosto generoituu, vaikka palautuksen yhteydessä ei tulisikaan virheilmoituksia. Tällöin tiedostossa lukee ainoastaan tieto, mistä Excel-tiedostosta käännökset ovat palautettu. Kuviossa 12 näkyy esimerkki loki-tiedoston sisällöstä.

```
Strings imported from C:\Documents and Set-
tings\luoti\Desktop\projektin_merkkijonot.xls.

Following Strings existed only in Excel sheet and were ignored:
\testi\strings\game-texts.strings: GAME (id 1)

Following Strings existed only in platform and were not updated:
\testi\strings\game-texts.strings: LEVEL_BEGIN_TEXT (id 4)

Following forbidden characters were found in Excel sheet:
\testi\strings\game-texts.strings: LEVEL_ENDING_TEXT: Fortgesetzt zu
werden…
```

Kuvio 12: Lokitiedoston sisältö

5 Sovelluksen testaus ja ongelmat

Haikalan ja Märijärven (2002, 282) mukaan ohjelmistojen testaus määritellään *suunnitelmalliseksi virheiden etsimiseksi* ohjelmaa tai sen osaa suorittamalla. Seuraavissa aliluvuissa kerrotaan sovelluksen testauksesta ja sen tuloksista, sekä sovelluksen toteutuksen yhteydessä vastaan tulleista ongelmista.

5.1 Testitapausten valinta

Testitapaukset voidaan valita kahden peruslähestymistavan kautta. **Lasilaatikkotestaus** (white box testing) tarkoittaa, että testitapausten laatija tuntee ohjelmakoodin toiminnan, ja käyttää tätä tietoa hyväkseen testitapauksia laadittaessa. **Mustalaatikkotestauksessa** (black box) ei vaadita ohjelmiston toteutuksen tuntemista. Testitapaukset suunnitellaan ja valitaan testattavan ohjelman spesifikaatioiden perusteella. Kolmantena lähestymistapana on **harmaalaatikkotestau**s (gray box), joka on yhdistelmä kahdesta edellisestä. Testitapaukset tehdään spesifikaatioiden mukaan, mutta samalla hyödynnetään tietoa ohjelman toteutuksesta. (Ohjelmistotuotanto 2002, 289.)

Sovelluksen testaus suoritettiin testitapausten avulla. Testaus suoritettiin harmaalaatikkotestauksen perusteiden mukaan, sillä testitapauksien suunnitteluun osallistuivat ohjelmoijan lisäksi myös henkilöitä, jotka eivät tunteneet tarkasti sovelluksen toteutusta. Sovelluksesta kerättiin kaikkiaan 17 testitapausta, jotka kuvasivat sovelluksen toivottua toiminnallisuutta. Osa testitapauksista on nähtävissä liitteessä 1.

5.2 Testauksen tulokset

Testauksessa kävi ilmi virhe sovelluksen toiminnallisuudessa. Käyttäjällä saattaa olla omalla koneellaan useita eri peliprojekteja. Kun käyttäjä valitsee sovelluksella “Export”, tulee sovelluksen hakea ainoastaan aktiivisena olevan projektin sisältämät merkkijonot ja luoda niistä taulukko. Sovellus kirjoitti kuitenkin kaikkien koneella olevien projektien merkkijonot taulukkoon, ei pelkästään aktiivisen projektin merkkijonot. Tämä virhe ei ollut käynyt ilmi aiemmin toteutuksen yhteydessä, sillä toteutuksen aikana

testiympäristössä oli ollut vain yksi testiprojekti. Virhe huomattiin vasta testauksessa, kun sovellusta ajettiin koneella, johon oli lisätty useampi peliprojekti. Virhe saatiin korjattua kuitenkin helposti, sillä se vaikutti ainoastaan siihen, mitkä tiedostot sovellus hakee, kun se generoi merkkijonotaulukkoa.

Sovelluksen vaatimuksena oli, että käännöksestä palaavat merkkijonot tallentuvat varmasti takaisin omille paikoilleen. Testitapauksissa käytiin tarkasti läpi eri vaihtoehtoja, jotka saattavat sekoittaa Import-toimintoa, kuten jos kahdella .strings-tiedostolla on sama nimi, tai kahdella samassa .strings-tiedostossa sijaitsevalla merkkijonolla on sama id-teksti. Import-toiminto toimi kuitenkin odotetusti, sillä sovellusta toteuttaessa oli varauduttu kyseisiin tilanteisiin antamalla jokaiselle merkkijonolle yksilöllinen Hidden id -tunnus. Hidden id -tunnuksen käyttäminen vertailuun yhdessä merkkijonon tiedostopolun ja id-tekstin kanssa takaavat sen, että Import-toiminnon yhteydessä suoritettavat tarkistukset toimivat ja merkkijonot tallentuvat oikeille paikoilleen.

5.3 Ongelmat sovelluksen toteutuksessa

Sovelluksen toteutuksessa eniten haasteita ilmeni tarvittavan tiedon löytämisessä. Rajapintana käytettävän HSSF:n dokumentointi vaikutti välillä puutteelliselta, sillä kaikkien metodien käytölle ei löytynyt selitystä rajapinnan dokumentoinneista. Tämä johti siihen, että joidenkin metodien käyttö ei sujunut ongelmitta, vaikka tietoa yritettiin etsiä myös Apache POI:n uutisryhmistä ja keskustelupalstoilta.

Yhtenä esimerkkinä voi mainita setHidden() -metodin. Sovelluksen toiminnassa oli vaatimuksena, ettei Excel-taulukkoon tule näkyviin xml-tiedostoissa olevaa Hidden id -tunnusta. Hidden id -tunnus on apuna, kun käännöksiä palautetaan Excel-tiedostosta takaisin xml-tiedostoihin. On tärkeää, ettei Excelissä olevia Hidden id -arvoja muuteta, jotta käännökset palautuvat oikeisiin kohtiin.

Excel-taulukon muotoilut suoritetaan HSSFCellStyle-luokan avulla. HSSFCellStyle-luokalla on metodi setHidden(), joka asettaa tyyliä käyttävät solut piiloon. HSSF:n dokumentointi kuvaa metodin käyttöä ainoastaan lauseella: *"set the cell's using this style to be hidden"* (POI API Documentation: HSSFCellStyle). Metodi ei kuitenkaan toimi-

nut odotetusti, ja sen käyttö kierrettiin lopulta käyttämällä HSSFSheet-luokan metodia setColumnWidth(short column, short width). Tämän metodin avulla Hidden id -kolumnin leveydeksi asetettiin 0, jotta kolumnin solut eivät tule näkyviin taulukkoon. Ratkaisu ei ole paras mahdollinen, mutta ainakin se mahdollisti sen, ettei Hidden id -kolumni ole oletusarvoisesti näkyvissä taulukossa.

6 Sovelluksen jatkokehitysideoita

Vaikka sovellus saatiin valmiiksi tavoitteiden mukaisesti ja se on otettu jo tuotantokäyttöön, löytyy siihen mahdollisia jatkokehitysideoita sekä sovelluksen toiminnallisuudesta että teknisestä toteutuksesta.

6.1 Päivittäminen XSSF-rajapintaan

Sovelluksen toteutuksessa käytettiin apuna Apache POI:n HSSF-rajapintaa. Sovellusta ohjelmoitaessa rajapinta pystyi käsittelemään ainoastaan Excelin vanhoja tiedostomuotoja, eli OLE2-formaattia, jota käytetään Excelin '97 – 2003 -versioissa. Työn kirjoitushetkellä tuli ilmi, että HSSF:n rinnalle on julkaistu XSSF-rajapinta, joka on tarkoitettu Microsoftin uuden OOXML-tiedostomuodon käsittelyyn. XSSF-rajapinnan avulla voi siis käsitellä Excel 2007 -version .xslx-päätteisiä tiedostoja.

Jos sovellus haluttaisiin päivittää niin, että se pystyisi lukemaan ja kirjoittamaan.xlst-tiedostoja, tulisi ohjelmakoodiin tehdä joitain muutoksia. Apache POI:n kotisivujen mukaan päivityksen näkyvämpänä muutoksena on se, että HSSF:n käyttämä luokkakirjasto *org.apache.poi.hssf.usermodel* korvaantuu uudella SS-luokkakirjastolla (*org.apache.poi.ss.usermodel*). Suurin ero näiden kahden välillä on se, että SS-luokkakirjastosta on poistettu luokkien nimistä HSSF-alkuosat. Täten esimerkiksi HSSFWorkbook-luokka on pelkkä Workbook. Muuten sisällöltään uusi luokkakirjasto vastaa vanhaa HSSF-kirjastoa, ja kaikkien toiminnallisuuksien tulisi säilyä ennallaan. (The Apache POI Project: Upgrading to POI 3.5)

Mielestäni sovelluksen päivittäminen vastaamaan uutta rajapintaa ei kuitenkaan ole tarpeellista, ainakaan vielä. Vaikka yrityksessä käytettäisiin Microsoft Office 2007 -pakettia, ja tiedostot tallennettaisiin .xslx-muotoon, saattaa käännöstoimiston käyttämä Excel-versio silti olla vanhempi, jolloin he tallentaisivat tiedostot vanhassa formaatissa. Excel 2007 -versiossa käyttäjä pystyy valitsemaan tiedoston tallennusmuodoksi myös vanhan OLE2-formaatin, eli tiedostopäätteen .xls. Vaikka sovellus ei pysty käsittelemään uusia

OOXML-formaatin tiedostoja, voi käyttäjä silti käyttää Excel 2007 -versiota, kunhan vain tallentaa tiedostot vanhan formaatin mukaisesti.

6.2 Tunnistetietoja sisältävien solujen lukitseminen

Toinen jatkokehitysidea liittyy sovelluksen toiminnallisuuteen. Toteutuksen yhteydessä muodostui ongelmaksi solujen piilottaminen taulukkoon (ks. luku 5). Tällä hetkellä Excel-taulukkoon sijoitettavat merkkijonojen tunnistetiedot, eli Hidden id ja tiedostopolku, ovat piilotettuina taulukossa, mutta näitä soluja ole mitenkään lukittu. Periaatteessa kuka tahansa käyttäjä pystyisi vetämään solut auki ja muuttamaan tai poistamaan tunnistetietoja. Tietojen muuttamisen jälkeen käännösten palautus ei enää onnistuisi, vaan käyttäjä saisi virheilmoituksia merkkijonoista, jotka on jo poistettu projektista, ja joita ei siten tallenneta alustaan.

Sovelluksen jatkokehityksen kannalta olisi hyvä saada nämä solut lukituiksi, jotta voidaan varmistaa, että tiedot pysyvät muuttumattomina. Vielä kuitenkaan tämän toiminnallisuuden puuttumisesta ei ole ollut ongelmaa, vaikka sovellus onkin jo ollut käytössä.

7 Lopuksi

Toimeksiannon tavoitteena oli toteuttaa sovellus, joka helpottaa mobiilipelien tekstien käsittelyä pelin käännösvaiheessa. Toteutettu sovellus saatiin valmiiksi aikataulussa ja se otettiin yrityksessä tuotantokäyttöön syksyllä 2008. Tammikuuhun 2009 mennessä sovelluksella oli käsitelty viiden projektin merkkijonot, ja toimeksiantaja on ollut tyytyväinen toteutettuun sovellukseen. Sovelluksen käyttöönoton myötä käännösten käsittelyssä käytetty aika on lyhentynyt, mikä on eduksi kiireellisten peliprojektien tuotantoaikataulussa. Voidaan katsoa, että toimeksiannon tavoite saavutettiin.

Apache POI -rajapinnasta ei ole saatavilla kirjallista lähdemateriaalia. Sovelluksen toteutusvaiheessa turvauduinkin internetistä löytyvään rajapinnan dokumentaatioon, mutta luin myös kehittäjien ja käyttäjien sähköpostilistoja. Rajapinnan käytöstä on kirjoitettu myös artikkeleja, mutta nämä tarjoavat enemmän taustatietoa kuin varsinaista ohjelmointiapua. Opinnäytetyötä kirjoittaessani käytin hyväkseni rajapintojen kotisivuja, joista sain suurimman osan lähdemateriaalistani. Kirjallisina lähteinä käytin Haikalan ja Märijärven Ohjelmistotuotanto-kirjaa, sekä Koskimiehen Oliokirjaa.

Sovellusta tehdessäni opin paljon käytännön taitoja ohjelmoinnista ja erityisesti Java-ohjelmointikielestä. Olin jo ennen sovelluksen ohjelmoimista tutustunut Apache POI -projektiin ja sen käyttöön Excel-taulukoiden käsittelyssä, mutta tämän sovelluksen tekeminen vahvisti tietojani rajapinnan toiminnasta.

Ohjelmistoja tuotettaessa harvaa sovellusta kirjoitetaan alusta alkaen itse, vaan käytetään hyväksi toisen kirjoittamaa ohjelmakoodia, jota muokataan ja jonka päälle rakennetaan omaa sovellusta. Koska toimeksiantona tehty sovellus integroitiin yrityksessä käytettävään ohjelmointiympäristöön, jouduin lukemaan myös muiden kirjoittamaa koodia ja ottamaan sen huomioon myös omassa tekemisessäni. Jos nyt alkaisin tehdä sovellusta uudelleen, tutustuisin ympäristöön ja valmiina olevaan ohjelmakoodiin laajemmin ja tarkemmin heti projektin alussa, enkä vasta ongelmakohtien niin vaatiessa.

Lähteet

Gartner 2008. Gartner Says Worldwide Mobile Gaming Revenue to Surpass $4.5 Billion in 2008. [online] [viitattu 4.12.2008]. http://www.gartner.com/it/page.jsp?id=706407

Gupta, Samudra 2003. Learn to Read and Write Microsoft Excel Documents with Jakarta's POI. [online] [viitattu 10.10.2008]. http://www.devx.com/Java/Article/17301/0/page/2

Haikala, Ilkka; Märijärvi, Jukka 2002. Ohjelmistotuotanto. 9. painos. Talentum Media Oy.

JExcelApi 2008. [online] [viitattu 19.10.2008]. http://jexcelapi.sourceforge.net/

JExcel Programmer's Guide 1.2 2008. TeamDev Ltd. [online] [viitattu 17.10.2008]. http://www.teamdev.com/jexcel/documentation.jsf

Koskimies, Kai 2000. Oliokirja. Helsinki: Satku – Kauppakaari.

POI API Documentation: HSSFCellStyle. [online] [viitattu 21.11.2008]. http://poi.apache.org/apidocs/org/apache/poi/hssf/usermodel/HSSFCellStyle.html

PRONOM The Technical Registry. OLE2 Compound Document Format 2008. [online] [viitattu 19.10.2008]. http://www.nationalarchives.gov.uk/PRONOM/Format/proFormatSearch.aspx?status=detailReport&id=767

TeamDev - Java & Microsoft Excel. TeamDev Ltd. [online] [viitattu 18.11.2008]. http://www.teamdev.com/jexcel/index.jsf

TeamDev – JExcel Purchase. TeamDev Ltd. [online] [viitattu 20.11.2008]. http://www.teamdev.com/jexcel/purchase.jsf

The Apache Poi Project 2008. [online] [viitattu 19.10.2008]. http://poi.apache.org/

The Apache Poi Project 2008. Overview. [online] [viitattu 19.10.2008]. http://poi.apache.org/overview.html

The Apache Poi Project 2008. POI-HSSF - Java API To Access Microsoft Excel Format Files. [online] [viitattu 19.10.2008]. http://poi.apache.org/hssf/index.html

The Apache Poi Project 2008. The New Halloween Document. [online] [viitattu 4.12.2008]. http://poi.apache.org/spreadsheet/how-to.html

The Apache Poi Project. Upgrading to POI 3.5. [online] [viitattu 5.1.2009]. http://poi.apache.org/spreadsheet/converting.html

The Apache Software Foundation. Apache Licence and Distribution FAQ. [online] [viitattu 10.1.2009]. http://www.apache.org/foundation/licence-FAQ.html#License

Universomo 2008. [online] [viitattu 18.10.2008]. http://www.universomo.com

Liitteet

Liite 1: Ote testitapauksista

TESTITAPAUS 5: Excel-taulukossa on käännöksiä merkkijono(i)sta, jo(i)ta ei enää ole projektissa.
Esivaatimukset: Projektista on Export-toiminnolla generoitu Excel-taulukko
Kuvaus:

1. Poista projektin alta yksi/useampia merkkijonoja
2. Valitse Import ja valitse exportattu Excel-taulukko
3. Varmista, että sovellus ilmoittaa “Following Strings existed only in Excel sheet and were ignored” ja alapuolella listaa poistettujen merkkijonojen nimet ja polut

Odotettu lopputulos: Sovellus ilmoittaa käyttäjälle, jos Excel-taulukossa on merkkijonoja, jotka on poistettu projektista.
Testin tulos: = odotettu lopputulos

TESTITAPAUS 7: Projektissa on merkkijono(ja), johon ei ole tulossa käännöksiä Excel-taulukosta
Esivaatimukset: Projektista on Export-toiminnolla generoitu Excel-taulukko
Kuvaus:

1. Lisää projektin strings-tiedoston alle uusi merkkijono
2. Anna merkkijonolle Id-teksti ja tallenna tiedosto.
3. Valitse Import ja valitse aiemmin exportattu Excel-taulukko
4. Varmista, että sovellus ilmoittaa “Following Strings existed only in platform and were not updated” ja alapuolella listataan uusien merkkijonojen Id-tekstit

Odotettu lopputulos: Sovellus ilmoittaa käyttäjälle, jos projektiin on lisätty merkkijonoja, jotka puuttuvat palautettavasta Excel-taulukosta.
Testin tulos: = odotettu lopputulos

TESTITAPAUS 12: Strings-tiedostossa on kaksi tai useampaa merkkijonoa, joilla on sama Id-teksti.

Esivaatimukset: -

Kuvaus:

1. Avaa projektin strings-tiedosto
2. Muuta kahdelle merkkijonolle sama symbol-nimi ja tallenna
3. Valitse Export ja avaa generoitu Excel-tiedosto
4. Lisää Excel-tiedostoon käännöstekstiä kyseisille merkkijonoille ja tallenna tiedosto
5. Valitse Import-toiminto
6. Varmista, että samannimisten merkkijonojen käännökset tallentuivat oikein.

Odotettu lopputulos: Käännökset tallentuvat oikein, vaikka strings-tiedostossa on kaksi tai useampaa merkkijonoa, joilla on sama Id-teksti.

Testin tulos: = odotettu lopputulos

TESTITAPAUS 15: Excel-taulukosta on tulossa ei-skandinaavisia kirjaimia (esim. Itäkielten aakkoset, aksenttikirjaimet jne.) strings-tiedostoihin.

Esivaatimukset: Projektista on generoitu Excel-tiedosto

Kuvaus:

1. Avaa Projektin Excel-tiedosto ja lisää käännössarakkeisiin erikoiskirjaimia
2. Tallenna tiedosto ja valitse Import
3. Varmista, että erikoiskirjaimet ovat tallentuneet oikein strings-tiedostoihin.

Odotettu lopputulos: Import-toiminto onnistuu, vaikka strings-tiedostoihin tallennetaan erikoiskirjaimia.

Testin tulos: = odotettu lopputulos