Copied from: AACE Proceedings of ED-MEDIA 2004. Edited by L. Cantoni & C. McLoughlin June 21-26, 2004; Lugano, Switzerland

My Webpage can Speak Many Languages

Tomasz Müldner, Fei Wang and Darcy Benoit
Jodrey School of Computer Science, Acadia University
Wolfville, NS, Canada B4P 2R6
Email: {tomasz.muldner,056330w,darcy.benoit}@acadiau.ca

Abstract: Growing globalization generates interest in internationalized software that can be localized to various languages. This paper describes a system that can be used to create an internationalized website which shows the Curriculum Vitae (CV) for a faculty member (creator). The final product is a website that "*can speak many languages*", i.e. a website that will initially be displayed in one language but will have the option to be displayed in a variety of languages chosen by the creator. In addition, the creator is able to specify one or more selections of data to be shown (for example, only journal publications), and one or more of formats these data are to be rendered in (such as HTML or PDF).

1. Introduction

In the past, most software programs could only "*speak one language*". For example, software developed in the UK could speak English, while the same software developed in China could only speak Chinese. (In this paper, a *language* means the natural language used in the **H**uman-**C**omputer **I**nterface (**HCI**). Unless it is clear from the context, we always say "a programming language" when we refer to a language used for programming.) Therefore, two or more versions of the same program that differ in HCI language might require completely different implementations, resulting in error-prone duplications of the original code. This situation was unacceptable for two reasons. Firstly, there are many multi-language countries, such as Canada, where English and French are two official languages. Secondly, with growing globalization, products are often developed in one country and shipped to several other countries. This forced software developers to rethink the software development process and tackle the issue of internationalization. *Internationalization* of a product means that the product can be adapted to various languages without making any changes to the architecture. *Localization* of the internationalized product refers to the adaptation of this product to a specific *locale*, which describes the language. For example, an internationalized Java calendar that has been localized to French will show dates in the format used in France. Due to the length of the terms *internationalization* and *localization*, the short, mnemonic terms of I18N and L10N are used respectively. The shortened terms are names based on the number of letters between the first and last letter of each word.

Product internationalization is difficult because it goes beyond simple functional suitability of a program and considers the many facets of HCI language. Various languages use different alphabets and scripts, spacing rules, directions, date and currency formats, sort orders, etc. Most programs have a Graphical User Interface (GUI) that is built with standard widgets, such as menus and dialog boxes. Localized versions of internationalized programs not only have to provide appropriate translations of interface menus and prompts, but the standard widgets must be able to handle all different HCI languages properly. In particular, widgets must be able to handle issues such as word length, word positioning, font differences and other such items. Such a "dynamic" interface requires that the implementation of the GUI can no longer be hard coded in the source code of the program; instead it must be parameterized to allow different versions of different languages to be plugged in without modifications to the program architecture.

Interest in internationalization is growing rapidly, with more applications being internationalized regularly. One example of this is the Hotel Reservation System, see HRS (2003), where the user can choose one of 25 available languages. Many companies, such as EXCEL Translations (2003) specialize in internationalizing existing applications. Several programming languages and APIs provide support for internationalization, such as the Java JDK 1.4, see Java (2001), JSPs, see Seshadi (2003) and NetBeans from SUN, see NetBeans (2003). However, there are few internationalized personal web pages or educational applications, with some exceptions such as the Mozilla, see Mozilla (1998) and Opera, see Opera (2003) web browsers. Both browsers have a core binary that is able to function by loading a separate file that contains the appropriate information for a localized interface. They each provide language files for over 20 different languages, allowing for an easy switch in the interface language. Finally,

Webmail, see Webmail (2003) is a popular email client running from any browser, whose user can choose one of over 20 languages.

Internationalization can be applied to the existing software or to software presently under development. In this paper, we describe the internationalization process applied to the *development* of a website. As a specific example of this general process, we describe the internationalization process using a system called **I**nternationalized **F**aculty **W**ebsite (**IFW**). This system can be used to create an internationalized website which shows the Curriculum Vitae (CV) for a faculty member. The design of IFW uses **S**eparation **O**f **C**oncerns (**SOC**) to separate tasks that require different types of technical expertise. A user of IFW (the creator of the webpage) does not need to have any technical background and need only follow a series of GUIs in a host language such as English. The English-speaking creator enters her or his CV data, selects one or more languages for the website to be displayed, and then forwards the project to the IFW administrator. The administrator is responsible for finding translators for the task, verifying the completed translation and making the final product available to the creator. The final product is a website that "*can speak many languages*", i.e. a website that will initially be displayed in one language but will have the option to be displayed in a variety of languages chosen by the creator of the website. In addition, the creator is able to specify one or more selections of data will be shown (for example, only journal publications), and one or more of selections of formats these data are to be rendered in (such as HTML or PDF). All translations are permanently stored and can be reused in future translation tasks. The implementation of IFW uses various recently developed software tools based on Java and XML, such as JAXB, versioning of XML documents, XML databases and relational databases with XML support.

This paper is organized as follows. Section 2 briefly describes some related work, and then Section 3 describes the functionality and implementation of the Internationalized Faculty Website. Finally, Section 4 provides conclusions and describes the future work.

2. Related Work

In this section, we describe major issues related to the internationalization process and the translation process, and the support for the internationalization given by XML and Java.

2.1 Major Issues

Internationalization of a system requires the identification of all data that can be shown to the user and may have different values under different locales. These data, known as *resources*, include user messages, page headers and trailers, button labels, etc. In addition, there may be many specific formatting problems. For example, translated strings can vary significantly in size. Consider the example borrowed from Raetzmann & de Young (2003), in which the English text "Authorized User List" consists of 21 characters, while the corresponding German text consists of 32 characters: "Liste der berechtigten Benutzer". Text sizes can cause other problems, such as the limited space associated with a box label. Extra characters should not be displayed to the left of the box, but should be displayed above the box. Standard formatting issues such as date and currency formats must also be considered. For a description of software that provides some tools to support this kind of formatting see Section 2.3.

2.2 Translation Process

At the time of writing this paper, Google, see Google (2003) and other sites provide English translation of small text fragments or entire Web pages between several languages. A user may choose to set the Google homepage to one of more than 100 interface languages. Microsoft Word 2002 is able to perform automatic translation between Chinese, English, French, German, Italian, Japanese, Korean, Portuguese, Russian, and Spanish. Figure 1 shows several Microsoft Word translations of the English sentence "The interest for internationalization is growing rapidly." We also show a translation back to English for each translated sentence.

L'intérêt pour l'internationalisation se développe rapidement
The interest for internationalization develops quickly
Das Interesse für Internationalisierung wächst schnell.
The interest in internationalization grows fast.
兴趣为国际化迅速显现出
The interest rapidly appears for the internationalization

Figure 1: Microsoft Word translations

Using a free upgrade from WorldLingo, see WordLingo (2003) it is possible to translate between many other languages. While not perfect, these translations can help the translator to perform the required task. Repeated translations of the same string should be avoided in order to make the translation process more efficient. For example, translations of common phrases such as "Press any key to continue" should be stored for future reuse. It is expected that as translation systems become more accurate, we will be able to automate much of the translation process.

Computer-assisted translation uses **T**ranslation **M**emory (**TM**) systems that typically consist of a translator module, an editor module, and a database of terms. TM software stores language segments translated by translator in a database for future reuse. Translators working on text segments can invoke fuzzy searches for these segments and use the results retrieved from the database. Some well known companies offering TM systems are Déjà Vu (2003), the Translator's Workbench from Trados (2003), and the STAR Transit (2003). TM software typically uses the **T**ranslation **M**emory e**X**change (**TMX**) format, which is a standardized XML document type for storing collections of segments in multiple languages. For more information on TMX, see Lisa (2003).

Using TM software for translations has both advantages and drawbacks. Firstly, TM software views the source text as a collection of text units called *segments*. Segments may range in size from simple text strings to paragraphs. The technique used to break up the text into segments is called *segmentation*. Segmentation may remove the context in which the text segment appeared, resulting in an incorrect translation. An example in Savourel (2001) shows that the English word "Help" translates to two different French words depending on the context in which the word appears. The common solution to the context problem is to use a verification phase in which the translator reads, verifies and possibly corrects the translation. The second problem with TM systems is that they are expensive, both in terms of the price of the software and the need to hire specialized personal able to use these systems. (More on automatic translation in Dennett (1995)).

2.3 XML and Internationalization

There are many advantages of using XML, see XML (2003) data for internationalization:

1) support for Unicode
2) Separation Of Concerns (SOC): describe content rather than formatting
3) ease of converting to various formats, including HTML, VoiceXML, etc.
4) support for mixing several languages
5) help to avoid repetitions by storing in, and retrieving from, databases
6) support for specifying translatable text

Below, we elaborate on these advantages. Re 1). XML supports ISO-10464 Unicode, see Unicode (2003) and can handle all languages in the world. By default, an XML document is using UTF-8 encoding, but it can be changed to other encodings; for example:

```
<?xml version='1.0' encoding='utf-16' ?>
```

Re 2) and 3). XML data only describe the context, and there are many examples of XSLT templates, Fitzgerald (2003) used to convert XML to other formats for storage and rendering, such as HTML; for example see OmniFormat (2003).

Re: 4). An XML document with multiple languages may look as follows:

```
<p xml:lang="pl"> noc </p>
<p xml:lang="de"> nacht </p>
```

Above, the xml:lang attribute is a standard attribute, whose values specifies the current language. The corresponding XPath, see XPath (2003) function is called lang(), and it can be used in XSLT templates. An example of such a function would be to recover all elements written in Polish. The reason this feature is useful is that some text may appear in English *and* another language, such as publication titles. Re 5). At the time of writing this paper,

there are few full-fledge XML databases. However, most existing databases provide tools to extract XML from relational tables (see Section 3.4.1). Re 6). The translatable text can be specified in several ways. First, there may be special tags to specify elements that are translatable, but this is difficult to do for complex elements, see XML FAQ (2003). Second, all translatable text may be external to XML documents, and referenced using entities or XInclude. The problem with this approach is that entities create a runtime overload, and as of now the support for XInclude is not readily available. In our system, see Section 3, we store data in a database. For more guidelines for creating XML documents for internationalization see XML FAQ (2003) and Rajgopalan (2003).

2.4 Java Support for Internationalization

Java provides strong support for internationalization, see SUN (2002), and Deitsch & Czarnecki (2001). In the Java SDK 2, internationalization resources are grouped into *resource bundles*. The internationalized Java program can be then use resource bundles to read the data for the current locale. Displaying data for a different local involves only specifying the locale; no change is required for the code of the program. If you use resource bundles and decided to use TMX , you will have to implement a conversion from bundles to TMX, see Itagaki (2000).

2.5 Dynamic and Static Content for Internationalization

In general, websites have static, dynamic or mixed content. Dynamic content has many advantages, allowing for content to be retrieved from a database for display. Dynamic content also allows for the separation of presentation from data and logic. Data can be displayed selectively with different renderings, depending on the data. Various technologies exist that support dynamic website content, such as Java Server Page, JSP, see Hall (2001). Resource bundles described in the previous section can also be used with JSP, see Seshadri, G. (2000). Resources are retrieved from the bundle during the initialization of JSPs and stored in the session of these pages. In addition, you can specify in JSPs the required Unicode encoding.

In the next section, we describe the functionality and implementation of the Internationalized Websites.

3. Internationalized Websites

3.1 Introduction

This section describes the internationalization process applied to the development of a website using the IFW system. The system is used to take **C**urriculum **V**itae (**CV**) and produce a webpage that is able to "speak many languages".

The *interface language* is the HCI language that appears in IFW's GUIs. The current version of our system uses English as an interface language. The language in which the CV data are entered does not have to be the same as the interface language. The *primary language* is the default language selected for the final product. For example, if the primary language is Spanish, then initial access to the website will be in Spanish. The website will also be available in the other languages selected by the creator when using the IFW system.

After the creator entered her or his CV data, they select various options to affect the creation of the final product. These options include the selection of:

- the primary display language for the website
- one or more secondary languages available for display
- data to be shown (all data, only journal publications, etc.)
- formats in which data can be rendered (HTML, PDF, PostScript, etc.).

As an example, the website may be initially displayed using Polish as the primary language, allowing the user to switch to one of these secondary languages: Chinese, English, German or French. In addition, the user can display all the data, or only the journal and conference publications, and select HTML, PDF or PostScript as the format in which these data can be rendered. It should be noted that the product of IFW can be run off any kind of a Web server. Figure 2 shows one of the GUIs used by the creator to select a source language, one or more target languages and one or more rendering formats.

Figure 2. GUI used by the creator

The creator of the website does not have to perform any tasks which require technical background, as these tasks are delegated to other IFW users, as described in the next section.

3.2 Users of IFW

The design of IFW uses SOC to separate tasks that require different type of technical expertise. There are four kinds of IFW users:

- creators (enter CV data, select formats, etc.)
- administrators (maintain accounts, forward documents, etc.)
- translators (translate documents submitted by administrators)
- verifiers (verify translations submitted by administrators)

IFW is a distributed system, consisting of a central server (IFW server) and users accessing the IFW server through the Internet. Any browser can be used to access the IFW system. Users must have accounts on the IFW server. Administrators of the IFW server create, modify and delete accounts for all users and maintain repositories of available translators and verifiers. There are two possible approaches to translation of documents. **Single source translation** involves selecting one source language and translating from the source language to other languages. **Multiple source translation** involves translating text from any one language to any other language. The former approach requires availability of translators from a single source language to many other languages, while with the latter approach it may be easier to find required translators. We use multiple source translation for this reason. For example, with *multiple source translation*, one translator may translate a document from English to French, and then another translator can translate the French version into German. This allows us to translate a document from English to German even if we do not have a translator that can do the direct translation.

In order to aid in the process of selecting translators, each translator has a *profil*e that lists pairs of languages; each language pair consists of the source language (a language to translate *from*) and the target language (the language to translate *to*). A translator's profile may include (English, French) and (English, German), while another translator's profile may include (French, Polish) and (Polish, French). We do not assume that the ability of translating from one language to another is reflexive. Similarly, each verifier has a profile listing all pairs of languages that this verifier can verify translations from and to. Administrators use repositories of translators and verifiers to determine the languages to include on the list of *IFW-available languages*. In the above example, IFW-available languages for English are French, German and Polish. For French, the only IFW-available language is Polish. The verifier is able to accept a translation (possibly with minor corrections), or reject it. Note that all data, their translations and status (new, translated or accepted) are permanently stored on the IFW server. The translators and verifiers can re-use previously accepted translations. Translators and verifiers can request accepted translations of a standard CV e.g. from English to Polish translations of just expertise keywords from German to Chinese. The workflow for creating internationalized website can be summarized as follows:

1. The creator asks the administrator to create an account for her or him.
2. The creator uses the account to access the IFW server and enters her or his CV data.
3. The creator selects one or more languages in which this website can be displayed and forwards the request to the administrator. Also, the creator selects data to be shown and formats for display.
4. The administrator submits translations tasks to translators.
5. When any translation is completed, it is returned to the administrator who forwards it to the verifier.

6. Once all translations are verified, the creator is notified and asked to choose one or more final products, with a specified primary language, choices of secondary languages, choices of data to be shown, and formats in which data are can be rendered.

Figure 3 shows the workflow for creating the internationalized website. The administrator may reject a request to create an internationalized website if there are no available translators or verifiers for the requested translations. Also, the administrator may reject a request if the tool to produce the required rendering is not available (for example, a request to create an RTF format). However, the creator may suggest to the administrator the availability of such a tool that the administrator can add to the IFW server.

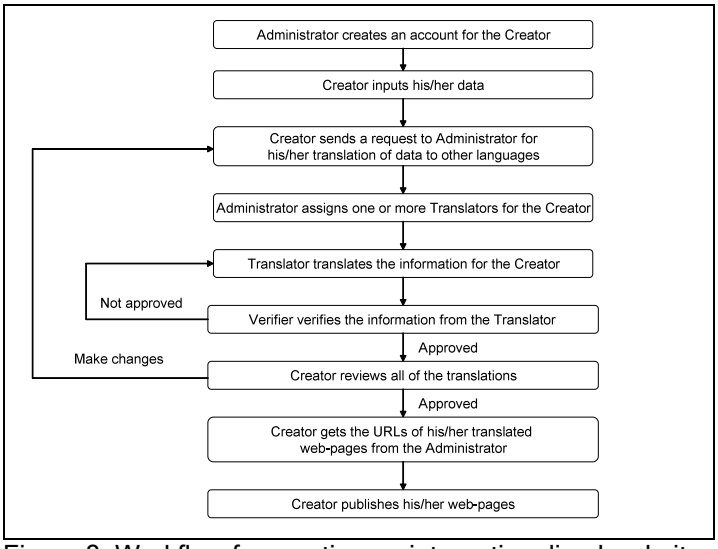

Figure 3. Workflow for creating an internationalized website

Various administrative tasks may be automated, such as assigning submissions to translators. The translated text is permanently stored in the IFW server with no repetitions, allowing for reuse as needed. The owner is able to modify data and resubmit them for translation.

3.3 Final Product and Maintenance of CV Data

In the previous sections, we described the process of creating an internationalized website. This section provides a more detailed description of what the final product – an internationalized website – looks like, and what the specific requirements are on the system serving this website (besides the standard requirement of a web server).

The final product is a website, containing one default webpage (displayed using a primary language), which can be used to:

- choose a secondary language to display CV data
- choose which data will be displayed
- choose a format for display.

The final product may appear in one of two available kinds:

- (i1) transient. All translations are stored in the database, and webpages accessible from the default page are dynamic. The website has to provide the same functionality as the IFW server; see Section 3.4.3.
- (i2) persistent. All webpages are static, generated by retrieving data from the database, and applying all transformations.

From the perspective of the client, who is accessing the website, there is no difference in what kind of final product is used. However, choosing a specific kind influences requirements on the server side, and efficiency of the website.

The generation process is performed by the IFW administrator. This process is time consuming and therefore should be performed only if the CV data are not to be frequently changed. On the other hand, a persistent product,

consisting entirely of static webpages, reduces the overhead caused by creating dynamic webpages, and does not impose any additional requirements on the website server. Therefore, the persistent product can be copied from the IFW server to any site with the standard web server. A transient product requires a website with the server that can handle servlets, access a database, etc. (see Section 3.4.3).

Maintenance of CV data, such as adding a new publication, requires a submission of the request to the administrator. Once translations of new data are preformed and verified, the transient product is available. However, if semi-persistent or persistent requested, then the administrator will generate this product.

3.4 Implementation

IFW is implemented with the help of several recently developed software tools using Java and XML; specifically, JAXB, databases (either native XML or relational that support XML), and versioning of XML documents. This section briefly describes the implementation; for a more details, see Müldner, T., Wong, F., and Benoit, D. (2003). Internally, all data are stored in XML. Specifically, we use the XML File Interchange Format (XLIFF); see XLIFF (2003). The advantage of the latter technique is that it is database-independent and there are various translations tools available that use XLIFF. Note that it is easy to convert XML to XLIFF using XSLT stylesheets; see Savourel (2001). The XLIFF format looks roughly like this:

```
<trans-unit>
  <target> … </target>
  <target> … </target>
</trans-unit>
```

The creator enters data by working as a IFW client (see below), and these data are stored in the database on the IFW server. To implement the programs used by all kinds of users, such as creators or translators, we use JAXB, see Ort & Mehta (2003), which is a Java technology that makes it easy to read, modify and write back XML data. JAXB hides from the internal XLIFF representation of data and shows appropriate GUIs to create data, translate them, etc. The text extracted from the database is handed over to one or more translators, who will be provided with the text, which is extracted from the translation fields (values of the target elements in the above example).

3.4.1 Database

IFW can use any database, which can create XML files. The current implementation uses the DB2 from IBM. IFW will access the database to perform several of its basic tasks, such as:

- building a localized CV for a specific language
- maintenance of the database, such as adding new faculty members, removing and updating data for existing faculty members
- performing translations and verification tasks.

3.4.2 XML Extender for DB2

The DB2 XML Extender provides much of the functionality needed to deal with XML documents. Included as a part of the XML extender are functions designed to work with XML documents, allowing for the retrieval of individual elements or entire XML documents. DB2 allows the XML documents to either be stored as external files or as character data within the database. XML documents can be translated into relational data and back, allowing relational queries to be posed against XML documents as well as allowing traditional relational data to be transformed into XML documents. In conclusion, DB2 provides all of the functionality of a relational DBMS while including sufficient XML functionality to manage XML data.

3.4.3 IFW Client and Server

An IFW client can use any web browser. The IFW server requires the following functionality: handling servlets, access to a database, JDK 1.4, and Java Web Services Development Pack, WSDP.

4. Conclusions and Future Work

This paper described the design and implementation of an IFW system that can create internationalized websites. This website can be used to display CV data in one or more languages and formats. Since the current state of automated translation systems requires human intervention, this version of IFW uses no automated translation. However, its design, in particular the use of XLIFF, makes it possible to include future versions of automated translation systems with no major changes of the IFW architecture.

Future work includes internationalizing the IFW system. Once the initial system is created in a single interface language, the system will be used on itself in order to generate new versions of the interface. IFW interface pages will always be kept up-to-date with any translators that may be available to the system. As new translators are added and new languages are added, the interface will be translated to represent the new languages available for translation. In addition, our future work involves experimenting with other relational databases, such as Microsoft SQL Server (2000) that uses a schema language XML Reduced (XDR). Also, we are planning on comparing relational databases and pure XML databases, such as Xindice, see Xindice (2003). We will experiment with XML versioning systems, see delta XML (2003) to compare new translations and accepted translations to retrieve parts that have to be corrected. Finally, we will extend the functionality of IFW, to support webpages displayed with more than one language.

References

Deitsch, A., & Czarnecki, D. (2001). *Java Internationalization.* O'Reilly.

Déjà Vu (2003). http://www.atril.com/

Delta XML (2003). http://www.deltaxml.com/

Dennett, G., (1995). Translation Memory: Concept, products, impact and prospects. South Bank University http://www.star-uk.co.uk/About_us/People/Gerald_Dennett/msc.pdf

EXCEL Translations (2003). http://www.xltrans.com/ser_tra.html

Fitzgerald, M. (2003). *Learning XSLT.* O'Reilly.

Google (2003). http://www.google.com

Hall, M. (2001). *Core Servlets and Java Server Pages.* Sun Microsystems Press/Prentice Hall PTR.

HRE (2003). Hotel Reservation Service, http://www.hrs.de/

Itagaki, M. (2000). Use XML as a Java Localization Solution. http://www.fawcette.com/Archives/premier/mgznarch/xml/2000/05win00/mi0005/mi0005.asp

IBM (2004). DB2 XML Extender Webpage, http://www-3.ibm.com/software/data/db2/extenders/xmlext/

Java (2002). http://java.sun.com/j2se/1.3/docs/guide/intl/index.html

LISA (2003). The Localization Industry Standards Association. http://www.lisa.org/tmx/

Mozilla project (1998). http://www.mozilla.org/docs/refList/i18n/

Müldner, T., Wong, F. and Benoit, D. (2003). Internationalization of Websites. Technical Report 2003-04, Acadia University, 2003

NetBeans (2003). http://www.netbeans.org/

OmniFormat (2003). http://www.omniformat.com/

Ort, E. & Mehta, B. (2003). Java Architecture for XML Binding (JAXB) http://developer.java.sun.com/developer/technicalArticles/WebServices/jaxb/

Raetzmann, M. & de Young, C. (2003). Galileo Computing Software Testing and Internationalization. TeriR@lemoine-international.com

Rajgopalan, S. (2003). Software Internationalization: A Holistic View. Advisor. http://portalsadvisor.com/doc/12841

RWS (2003). http://www.translate.com/locales/en-US/companyinfo.html

Savourel, Y. (2001). XML Internationalization and Localization. SAMS.

Seshadri, G. (2000). Internationalize JSP-based Websites. Java World, http://www.javaworld.com/javaworld/jw-03-2000/jw-03-ssj-jsp.html

SUN (2002). Internationalization. http://java.sun.com/j2se/1.3/docs/guide/intl/index.html

Trados (2003). http://www.trados.com/

Transit (2003). http://www.star-ag.ch/eng/software/sprachtech/transit.html

Unicode (2003). http://www.unicode.org/

Webmail (2003). http://www.webmail.co.za

WordLingo (2003). http://www.wordlingo.com/

Xindice (2003). http://xml.apache.org/xindice/

XLIFF (2003). XLIFF 1 Specification. http://www.oasis-open.org/committees/xliff/documents/xliff-specification.htm

XML (2003). http://www.wordlingo.com/

XML FAQ (2003). XML Internationalization and Localization FAQ. http://www.opentag.com/xmli18nfaq.htm

XPath (2003). http://www.w3.org/TR/xpath