# Transitioning a Vastly Multilingual Corporation to Unicode

*Lorna A. Priest*

For 70 years SIL International has been working to study, develop and document the world's lesser-known languages. Many of these languages were previously unwritten and thus the speakers of these languages have often lived on the fringes of society around them. SIL's work typically includes academic research (which can include linguistic analysis of the language and anthropological research), translation and literacy work.

As an organization with 2,000+ linguists, coming from over 35 countries, and working in over 1,200 languages, SIL International had a strong incentive to switch to Unicode encoding. Document sharing can be a nightmare when you have up to 100 different custom-encoded fonts in use in one country! And as industry began to force the use of Unicode (by not supporting legacy solutions) the incentive grew stronger. In 2001 SIL International began a concerted effort to transition the organization to using Unicode.

Many steps were involved in our transition. These included getting computer support people on board; helping upper management to understand the complexities and the need; and developing tools for converting legacy data to Unicode, fonts which cover Latin and Cyrillic repertoires as well as non-Roman fonts, software to use Unicode and training in using the tools of Unicode. We have discovered that training is an ongoing process. We continue to find areas where we need to focus our training efforts. This paper addresses the steps taken, the problems that arose, and the solutions that were found.

## 1. The world as we knew it

### 1.1. "Hacked" fonts

For as long as SIL has been using computers, we have had to create "hacked" fonts because the characters we needed were not in standard fonts. If a linguist wanted a lowercase "barred u" (ʉ) and uppercase "barred U" (Ʉ) they found a codepoint that they didn't need and put their barred u and capital barred U in those positions (see Figure 1).

| ANSI | Codepage 1252 | "hacked" encoding |
|---|---|---|
| 230 | æ | ʉ |
| 198 | Æ | Ʉ |

Figure 1. "Hacked" font

If the example used in Figure 1 was the worst they had done, they would not run into many problems. But, if they happened to choose a non-word building codepoint, then they began to experience problems with lines breaking mid-word. If the application in use had a "smart quotes" feature, and the intended character was given the codepoint for " " or " ' " then it would be converted to an opening or closing quote! This could be especially confusing if

someone decided to encode a special character in an opening quote position! Figure 2 illustrates this.

| ANSI | Codepage 1252 | "hacked" encoding | "smart" quotes "on" |
|---|---|---|---|
| 034 | " | ʉ | “ |
| 039 | ' | Ʉ | ‘ |
| 147 | “ | ɨ | ʉ > ɨ |

Figure 2. "Hacked" font behavior when "smart" quotes are turned on.

If they happened to choose positions that were not case matches, and they allowed their application to "change case," they immediately had the wrong character. Or if "Capitalize first letter of sentences" was turned on, then when the user input ANSI 224 (à) to get "ŋ," Word would "fix" it, changing ANSI 224 (à) to ANSI 192 (À) (displayed as "ʉ"), which was not what was wanted at all (see Figure 3).

| ANSI | Codepage 1252 | "hacked" encoding | Change case |
|---|---|---|---|
| 192 | À | ʉ | ʉ > ŋ |
| 193 | Á | Ʉ | Ʉ > Ŋ |
| 224 | à | ŋ | ŋ > ʉ |
| 225 | á | Ŋ | Ŋ > Ʉ |

Figure 3. "Hacked" font behavior when "Change case" is used.

From Word 97 on, when text (that was formatted with a non-symbol font) was exported using "save as text", some translations occurred. Characters U+0100 and above must get translated into 8-bit some way, and it is done by mapping characters into their nearest ASCII equivalents, where this makes sense, or into "?" if there is no similar ASCII character. For characters in the range U+0020 to U+00FF, you would expect that these would be translated into the 8-bit numerical equivalents, but this is not necessarily the case, as shown by the examples below. These may be a perfectly natural translation unless you have "hacked" your font and put a barred u in place of "®"!

```
00A0 > x20 (non-breaking space > space)
00A9 > x28 x63 x29  (copyright symbol > "(c)")
00AE > x28 x72 x29  (registered symbol > "(r)")
00BC > x31 x2f x34  (fraction one quarter > "1/4")
00BD > x31 x2f x32  (fraction one half > "1/2")
00BE > x33 x2f x34  (fraction three quarters > "3/4")
```

Figure 4. "Hacked" font behavior when "save as text" is used.

## 1.2. Symbol fonts

In order to avoid these kinds of problems, many people encoded their "hacked" fonts as symbol fonts. As soon as Word 97 was released we began seeing problems related to symbol fonts and Unicode. Some of the problems we began experiencing were:

- If text is formatted with a symbol-encoded font and the formatting is changed to a non-symbol-encoded font, the text may appear as empty boxes.

- If a run of text is formatted with a symbol-encoded font, lines will wrap at any point in the run, not only at word boundaries.
- If text is formatted with a symbol-encoded font and the file is saved to Word 6.0/95 or text formats, when the file is reopened, that text will have been changed to question marks.

### 1.3. One font per language or language family or country

Because of the technical limitation of only 255 characters in a font, we had many, many fonts. The best-case scenario was a group of SIL entities in West Africa who did an assessment of what characters were needed and created a font to meet all of the needs in that country or region. The worst case scenario was countries in Latin America that had one font per language. Document sharing was a nightmare if someone forgot to send the font along with the document! Data loss occurred if the font was not archived with a document.

## 2. A brave new world – transitioning to Unicode

So, there was agreement that we had a problem. However, the general consensus was that software manufacturers should not be messing up our data! We had a working solution (we could live with document sharing problems) until they messed up the applications we were using. It took awhile to convince people that this was *our* problem, we had "hacked" the fonts. We shouldn't expect others to fix our problem when we were the ones who had developed non-standard solutions.

Although many developers had seen that Unicode was something we should start using, it wasn't until 1998 or 1999 that it became a concern to a critical mass of technical (computer) people within SIL.

SIL International has a Language Software Board (LSB) which guides policy regarding linguistic-related software development. In 2001, as a result of beginning to see the value of Unicode, a number of recommendations were made to the LSB and were accepted as a corporate strategy[1]. Amongst those recommendations was that all future software development must be Unicode-enabled.

This document stated that accomplishing a transition would require attention in three main areas:

- Software (conversion tools, Unicode-capable language software applications)
- Writing-system resources (encoding conversion mappings, fonts, keyboards, Private Use Area (PUA) character semantics descriptions)
- Support and training

These are discussed in detail below.

Because of the large amount of language data that SIL International has produced over the past 70 years, and continues to produce, our transition to Unicode has had to be staged. The scope and priority in transitioning to Unicode was stated as:

> Because we are not only publishing, but eventually also archiving data, the scope of the transition will initially be to active language projects who will benefit from making the transition for their day-to-day work, then to all active projects, and then finally to all projects in which we have been significantly involved.

[1] This document was titled "Corporate Strategy for Transition to Unicode."

Our transition to Unicode is not limited to the Roman world. The Non-Roman Script Initiative (NRSI) is a department within SIL International which has been primarily responsible for promoting Unicode in the organization. As its name implies, the NRSI is primarily interested in meeting the computing needs of the non-Roman world. However, it became clear that until all, or a majority, of our software was Unicode-enabled, we had no chance of developing non-Roman solutions. In order for our software to be Unicode-enabled, the Roman world had to be convinced about converting to Unicode (the non-Roman world was already convinced of the need for Unicode). Before they could be convinced, resources had to be in place. The NRSI made the decision to delay non-Roman work (committing to provide a Unicode global Roman font) if SIL's Language Software Development team would chose to *not* support legacy fonts in our new suite of linguistic applications called FieldWorks[2] (thereby forcing people to use Unicode). The NRSI felt that this decision would ultimately support our non-Roman needs better.

## 2.1. Software

A large number of SIL members use Microsoft software. Because current Microsoft software has very good support for Unicode, and relatively good support for complex-script rendering, users were encouraged to use Windows 2000 or later versions of Windows. We recommended Word 2002 (or later) for general word-processing needs and WorldPad[3] for those requiring complex-script rendering for private-use (PUA) characters[4].

### 2.1.1. Unicode-capable language software applications

Our 2001 Unicode Transition strategy document mandated that all new SIL software should support Unicode. This particularly included our new suite of applications called **FieldWorks**. In order to encourage people to begin using Unicode, this suite needed to include all of the linguistic analysis capabilities which our legacy applications could do. FieldWorks uses **Uniscribe/OpenType** and **Graphite**[5] rendering capabilities which allow linguists to use whatever solution best suits their needs.

A successor to the Shoebox[6] program, which a number of linguists have used for many years, is called Field Linguist's **Toolbox**[7]. Toolbox is a data management and analysis tool. It is especially useful for maintaining lexical data, and for parsing and interlinearizing text, but it can be used to manage virtually any kind of data. It has been Unicode-enabled using UTF-8.

Another useful tool that was developed by the United Bible Societies is called **Paratext**[8]. Paratext is a Bible translation software program that allows a person to input, edit and check a translation of the Scriptures, based on the original texts (Greek, Hebrew), and modeled on versions in major languages. Paratext is also Unicode compliant.

A solution to our keyboarding needs was released in 2000, a Unicode enabled version of **Keyman**[9]. This is an easily customizable keyboarding utility. We have continued to encourage

---

[2] See http://fieldworks.sil.org.

[3] See http://scripts.sil.org/WorldPadDownload.

[4] Since, by nature of the Private Use Area, industry software does not support the PUA.

[5] See http://scripts.sil.org/RenderingGraphite.

[6] See http://www.sil.org/computing/shoebox.

[7] See http://www.sil.org/computing/toolbox.

[8] See http://paratext.ubs-translations.org.

[9] See http://www.tavultesoft.com.

the development and use of this product as well as encouraging the use of Microsoft's Keyboard Layout Creator (**MSKLC**)[10] where appropriate.

### 2.1.2. Character encoding conversion tools

Our Unicode Transition strategy document also mandated an encoding conversion component that could be used as a stand-alone conversion tool or by other SIL applications. It needed to be able to perform conversion on plain-text data or on Standard Format Marker (SFM) data (with the ability for different SFM fields to undergo different conversions). Similar in concept to XML, SFM is a structured data markup which has been in use within SIL for over 20 years. The encoding conversion component also needed to be able to convert data on the clipboard.

Currently, three tools have been created in assisting the conversion of legacy data to Unicode.

The first of these is called **TECkit**[11]. It was beta released in November of 2000. TECkit is a system for defining and implementing the conversions or mappings between the custom 8-bit encodings used by our legacy solutions and the Unicode standard.

TECkit was the biggest part of the solution for data conversion. In order to use TECkit though, people had to create legacy-to-Unicode mapping files. Since the majority of our legacy fonts had been created using our Encore fonts, a second tool was created called **Encore2Unicode**. This utility could look inside an Encore-derived TrueType font and extract enough information to create a draft mapping file.

TECkit was designed to work with unformatted text. Much of the data SIL linguistics have is formatted, and people did not want to lose that formatting. A system-wide repository for encoding converters and transliterators was created, along with a simple COM interface to select and use a converter from the repository. The repository supports three kinds of converters: TECkit; ICU-based[12]; and CC ("Consistent Changes", a transduction tool that has been widely used within SIL for many years). This third piece to the encoding conversion solution is called **EncCnvtrs**[13]**.**

Without having these in place SIL linguists would be resistant to converting their legacy data to Unicode. The FieldWorks suite also includes tools which make use of TECkit for converting legacy data to Unicode as it is imported into FieldWorks.

### 2.1.3. Software Summary

FieldWorks applications are still being developed. It will be some years before the FieldWorks suite of applications can update or replace all of the legacy applications that are currently in use. FieldWorks may eventually be able to replace Toolbox and Paratext as well, but until that time we already have these Unicode tools available.

The character encoding conversion tools are still considered "proof of concept." The team developing FieldWorks will also continue enhancing the conversion tools in integrally interacting with FieldWorks as well as making them easier to use.

Unicode-enabled publishing applications with sufficient OpenType or Graphite support for complex rendering are still lacking. A Mac solution is **XeTeX**,[14] a typesetting system based on a merger of Donald Knuth's TeX system with Unicode and Mac OS X font technologies. Our goal is to have a complete solution (or solutions) identified by the end of 2005.

---

[10] See http://www.microsoft.com/globaldev/tools/msklc.mspx.

[11] Text Encoding Conversion toolkit. See http://scripts.sil.org/TECkit.

[12] International Components for Unicode. See http://oss.software.ibm.com/icu/.

[13] See http://scripts.sil.org/EncCnvtrs.

[14] See http://scripts.sil.org/XeTeX.

## 2.2. Writing-system resources

The Unicode Transition strategy paper outlined the need for Unicode *fonts* which would contain all the characters needed for our work and the need for *mappings* to convert that data, both structured and unstructured, to Unicode. As mentioned earlier, we chose to focus on providing writing-system resources for the Roman world first, in order to gain greater acceptance for the use of Unicode.

### 2.2.1. Assessment

We found that the writing-system needs in the Roman world was a big gray area. We didn't really have any idea how many legacy fonts were in use within our organization. This meant we didn't know how many encoding systems were out there. The NRSI was assigned the task of assessment.

The first step in the process of assessment was begun even before the Unicode Transition strategy was set in place. In July of 1999 we started talking about a Unicode IPA font[15]. This font was specifically geared toward the International Phonetic Alphabet (used for technical representation of linguistic data), rather than orthographic (used by the speakers of the language). We worked with our linguistics department as well as eliciting suggestions from anyone who was interested in this font. We released a beta Unicode IPA font in the fall of 2000 with at least one maintenance release after that. Subsequent to this release we began to realize we needed one font that covered all of the Latin and Cyrillic IPA *and* orthographic needs throughout the corporation. Early on we decided to include all of the Latin and Cyrillic blocks which were in Unicode, whether we had a documented need for a particular character or not.

The process of inventorying the character and glyph inventory took approximately two years. We requested that each SIL entity appoint a Unicode Transition representative (UT representative). This person would be the one the NRSI would interact with. This person collected all of the fonts from their entity and sent them to the NRSI. With the **Encore2Unicode** utility we were able to, fairly painlessly, inventory all the glyphs in the fonts. Ultimately we had several hundred fonts we worked with. Once this inventory was in a database we could do a frequency count of glyphs, as well as knowing which glyphs mapped to which USV. Through this process we found a number of alternate glyphs for the same character, and we also found between 50-100 characters which were not in Unicode. At this stage we went back to the UT representatives to find out if the character was in actual use. In some cases the character had been tested during orthography development and was never used. We did not include those in our font. Other times there was a definite need for that character. The decision of whether to include non-Unicode characters in our fonts was passed on to the PUA Committee.

### 2.2.2. Creating an SIL PUA committee

The SIL Private Use Area Committee (PUA)[16] was created for developing policy for managed use of the Unicode Private Use Area within SIL, and promoting this policy on behalf of SIL International. Our goal was to coordinate the use of characters in the Private Use Area so that data can be shared easily among SIL groups working in different parts of the world. We see the SIL PUA as a private extension of Unicode proper, requiring the same level of definition and documentation.

A natural result of having a corporate-managed PUA is knowing what characters need to be proposed for additions to Unicode. To date we have 230 characters in our corporate PUA but 123 of those have now been proposed and accepted (or approved for acceptance) into the Unicode standard.

---

[15] See Hosken, 2000.

[16] See http://scripts.sil.org/UnicodePUA.

### 2.2.3. Creating a font

Because we planned to create more than one font we put a lot of effort into designing a "Font Information Database" (FIDB) to contain all the information possible about the characters and glyphs needed. This database has a vast amount of information in it which will be used as we develop other typefaces. The database contains information such as:

- SIL Names for glyphs
- PostScript names
- Unicode encoding and character names
- Alternate glyphs for various characters
- Ligature information
- Font-specific information for glyphs
- A way of grouping glyphs into sets
- PUA characters and their properties
- Attachment points (this information is not currently in the FIDB but the structure is there to handle it)

Many of the glyphs needed were already in our Encore font system and those glyphs did not need redesigning. However, the following steps were still needed:

- Designing new glyphs (using FontLab)
    - Completion of Latin and Cyrillic ranges of Unicode
    - PUA characters not in our Encore fonts
    - Specifying which glyphs could be created as composites and how
- Attachment points
- Writing Smart font code for
    - Alternate glyphs
    - OpenType
    - Graphite
    - AAT

| | |
|---|---|
| *Capital Eng alternates* | Ŋ<br>Ɒ<br>Ɲ |
| *Tone numbers* | ˥ / [115] |

Figure 5. Smart font code: Alternate glyph selection.

| | Font with smarts | Font without smarts |
|---|---|---|
| ɛ + ◌̃ + ◌́ | ɛ̃́ | ɛ̃́ |
| Ɛ + ◌̃ + ◌́ | Ɛ̃́ | Ɛ̃́ |
| g + ◌̩ | g̩ | g̩ |

Figure 6. Smart font code: Stacking diacritics.

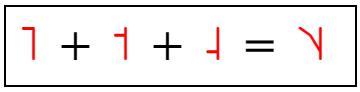

Figure 7. Smart font code: Tonebar ligatures.

A very complex font build process was developed which involves running scripts written in **Python** and **Perl** as well as some manual interaction with **FontLab** menus and dialogs (FontLab is scriptable in Python and all but a few of the scripts run within FontLab). It uses data from FontLab, the Font Information Database (FIDB) in **Access**, attachment point definitions from **Excel**, and Composite definitions from an **XML** file to produce a **TrueType** (TTF) font file with the correct glyphs, encoding, PostScript names, line metrics, and internal strings. Graphite and OpenType code is then inserted into the TTF.

We anticipate this process will ensure future development of other fonts to be straightforward.

### 2.2.4. Testing the font

A test harness was created using Perl scripts and Perl extension modules to test both Uniscribe and Graphite rendering. It tested:

- That every glyph was included for the Unicode ranges specified
- Every glyph for accurate attachment points (with upper and lower diacritics, as well as stacking diacritics)
- Every glyph with surrounding text (for side-bearings)
- Alternate glyph selection

We also held a test cycle with approximately 15 field testers. They were given a list of things to check which can be found in Appendix A.

Issues found during the testing phase were:

- Attachment points in wrong place (ascenders, descenders)
- Stacking diacritic problems
- Side-bearings
- Dot removal, for dotted glyphs with diacritics
- Hinting problems
- Bugs in software (Uniscribe, Graphite and Internet Explorer)

These bugs were fixed, documented or reported to the appropriate people.

### 2.2.5. Mappings

In order for people to convert their structured and unstructured data to and from Unicode, mapping files to convert their data had to be created. Our corporate efforts were put into developing the tools (see Section 2.1.2) and into training (see Section 2.3) to handle the sometimes very complex encoding mappings.

**Basic form:**

*Legacy <> Unicode*

```
0x60 <>   combining_vertical_line_below
0x61 <>   latin_small_letter_a
0x62 <>   latin_small_letter_b
0x63 <>   latin_small_letter_c
0x64 <>   latin_small_letter_d

0x65 <>   U+0065 ; lowercase e

"f"  <>   U+0066

103  <>   U+0067

0x43  <>  latin_small_letter_c combining_cedilla

0x43  <>  U+0063 U+0327

0x6C 0xF2 <> latin_small_letter_l_with_middle_tilde

'ha'  <>  ethiopic_syllable_ha
'hu'  <>  ethiopic_syllable_hu
'hi'  <>  ethiopic_syllable_hi
'he'  <>  ethiopic_syllable_hee
```

Figure 8. Simple TECkit mapping rules.

In the non-Roman world, we have had various types of legacy solutions: glyph-based encodings with dumb rendering, often associated with very complex keyboards, in effect implementing the character/glyph model in the keyboard handler; and also various smart-rendering technologies based on non-Unicode encodings.

Many of these legacy non-Roman solutions, particularly the glyph-based dumb-rendering solutions, have led to a requirement for extremely complex encoding mappings. In effect, the legacy/Unicode mapping has to implement the character/glyph model for the particular script, reversibly, in order to transduce between Unicode *characters* and the legacy "character codes" that actually correspond to glyphs or even glyph fragments.

We are in the process of establishing a repository of encoding mappings which can be used with the conversion tools.

### 2.2.6. Archiving

Archiving is also an issue that we are addressing (or planning to address). Data that has already been archived at some point over the past 70 years is likely encoded in a hacked legacy encoding. Unless the "hacked" font was archived with the data, much of that data may be lost. This data may eventually be converted to Unicode as well, but these will be dealt with as the

need arises, and as personnel and other resources are available. We are just beginning to put policies into place concerning this.

### 2.2.7. Input method resources

Although not part of the strategy paper, the need for good input methods was implicit. Developing keyboards is not a difficult process and rather than put corporate resources into this, we have ensured that training is available for field-support staff to develop these. A respository for these keyboards is also in process.

### 2.2.8. Writing-system resources summary

We discovered that our initial inventory of characters and/or glyphs needed was good but somewhat incomplete. After the first release of the "Doulos SIL" font there were several months where requests trickled in for other characters. We have no doubt that this will continue.

At this point we have only released one global Latin/Cyrillic font. Development of the next publication-quality font has begun. At completion we anticipate our fonts to include serif and sans-serif faces appropriate for linguistic research, literacy development and publishing. We anticipate this to take several more years.

Maintaining a repository for the mapping files and for keyboards will be an on-going process.

## 2.3. Support and Training

The Unicode Transition strategy paper also addressed the need for training. Those areas of our corporation that were involved in Unicode and/or in computer training were instructed to examine the need to provide Unicode-related training for field-support staff. The NRSI does not have a mandate to provide training, but they were the main repository for Unicode knowledge.

As we began to work on this area we realized that our small group of people could not provide all the support and training that the entire corporation needed. The NRSI began to focus their efforts on *training*. We felt that this would provide the necessary infrastructure for the *support* that people would need.

### 2.3.1. Training

Two primary areas of training began to emerge. Firstly the NRSI felt that we should hold a workshop to help our computer support people get up to speed in the area of Unicode. To address this and other issues concerning non-Roman scripts, over a two-year period we began to organize what eventually became known as the "Non-Roman Technical Consultation." This brought together SIL individuals from 18 countries as well as individuals from 11 partnering organizations. It was held September 17-22, 2001 in the UK. In the process of pulling together the agenda for this workshop we realized the need for a resource handbook to give the participants. Out of this came a book called "Implementing Writing Systems."[17] It included chapters on:

- Understanding characters, keystrokes, codepoints and glyphs,
- Character set encoding basics
- Understanding Unicode
- Keyboard design theories
- Rendering technologies

After this workshop we also held one other large workshop and two small ones called "Unicode Transition Workshop". Attending these workshops were people who needed to understand

[17] See http://scripts.sil.org/IWS-TOC.

Unicode for their work as well as people whose job is to do computer training. Much of future training could be handled by these people.

### 2.3.2. Support

Although we felt that support could be handled by the people we trained, we found that the people we had trained still needed a level of support themselves. When they ran into problems they needed someone to come back to. That often was people in the NRSI.

Besides email communication we began to put as much of our combined knowledge on the Internet. The development of our website[18] has greatly aided in the transfer of knowledge to these individuals. This website contains much of our training material, all of the material from the book "Implementing Writing Systems: An Introduction," many of the tutorials that were developed for the various workshops, as well as Unicode fonts, Unicode keyboards, and legacy-to-Unicode mapping files.

### 2.3.3. Support and Training summary

Communication was a huge part of support and training. In addition to the development of the website, we started several internal email mailing lists to inform people about Unicode and also to provide a discussion forum for people who were looking for Unicode solutions.

We have also discovered that "Support and Training" does not just include technical support people. An area we are just beginning to address is the need to train general language workers to understand and work with Unicode-encoded text. Some of this will need to be self-paced training for language workers already involved in linguistic work but training also must begin during their SIL academic training. For instance, learning about orthography development should include an understanding of Unicode.

## 3. Conclusion

We have not finished the process of converting to using Unicode. This will probably take many more years. Some of our linguists are still using Windows 98 and their computers are not capable of running some of the newer software. Others are nearing completion of their work and don't want to switch mid-stream. Some will not be able to switch until there is a good Unicode publishing solution. But most are now aware of Unicode and know that someday they will have to convert their data to Unicode. The pieces are mostly in place for them to convert their data when the time comes.

## 4. References

Constable, Peter. 1999. "Microsoft Office 2000 Beta 2 Preview" NRSI Update #10.

___. July 2002. "Corporate Strategy for Transition to Unicode—Summary of Recommendations to the Language Software Board" NRSI Update #17.

Drescher, Dennis. 2004. "Corporate Transition to Unicode: Roman Font Strategy." Working document.

Hosken, Martin. 1997. "Windows and Codepages" NRSI Update #8.

___. 2000. "IPA: The Future" NRSI Update #12.

Kew, Jonathan. July 2002. "TECkit—new and improved" NRSI Update #17.

[18] See http://scripts.sil.org.

# Appendix A

*The following form was used to solicit feedback from our Doulos SIL font testers.*

Users who don't require dynamically positioned diacritics, ligatures, or alternate glyphs can use most any app to test the font with, but otherwise there are three basic categories of test configurations:

**A. We would like the font to be tested in the following environments (we don't expect you to test them in all environments, choose - then tell us what you are doing):**

1. **Uniscribe testing**: In addition to other kinds of testing, Uniscribe testers should be looking at diacritic placement and ligatures (e.g., tone bars). Uniscribe testers should be aware that base+diacritic combinations that exist in Unicode (and the font) as precomposed chars are handled differently than those that are not (Uniscribe favors the use of the precomposed). (Note however: PUA characters needing special handling, e.g., diacritics, will not work properly in Uniscribe-based apps. Nor do the double-wide diacritics).
    a. Office2003 (Word and Publisher)
    b. Paratext 6 (http://paratext.ubs-translations.org)
    c. NotePad (will only work if you put the Uniscribe DLL from Paratext 6 or Office 2003 in a directory along with a copy of Notepad.exe and then use that Notepad.exe)
2. **Non-Uniscribe OpenType testing:** The only app I'm aware of is InDesign (or perhaps any of the new Creative Suite from Adobe). Adobe apps do not yet handle dynamic diacritic placement. But one thing they permit that Uniscribe-based apps do not is selection of alternate glyphs.
    a. InDesign 2 or CS (can download an evaluation version from: http://www.adobe.com/products/indesign/main.html)
3. **Graphite-testing:** In addition to all the above (from Uniscribe testing and Non-Uniscribe OpenType testing), Graphite should handle the double-diacritics and PUA chars correctly, and should present a menu for changing font features.
    a. WorldPad (http://scripts.sil.org/WorldPadDownload)
    b. Mozilla (http://sila.mozdev.org)
    c. Data Notebook (FieldWorks)

**B. Some ideas on what to do:**

1. Take your branch keyboard
2. Convert it to Unicode (TECkit may be helpful to you: http://scripts.sil.org/TECkit)
3. Using your new keyboard, type in all possible character combinations which your entity uses
4. Check any dotted characters (i, j) with diacritics.
5. “Select All” and “Change Case” to see if case mappings are correct.

**C. Next**

1. Create a TECkit mapping file for your entity (You may want to look at what other entities have done: http://scripts.sil.org/ConversionMaps or check out the tutorials at: http://scripts.sil.org/UnicodeTutorials)
2. Convert three or four legacy-encoded scripture chapters into Unicode (this has the benefit of checking a large portion for text flow, character spacing, word spacing, etc) and/or
3. Create a word list of an entire scripture text (this has the benefit of having all character and diacritic combination needed for this language, but does not show paragraph flow)
4. Import file into one of the above applications
5. If using WorldPad or InDesign
    a. Change font features (i.e. choose an alternate glyph such as other form of eng or glottal stop)
        1) Capital Eng alternates
        2) Rams horn alternate
        3) Tone letters
        4) Cyrillic E alternates
        5) Combining breve Cyrillic form
        6) Vietnamese-style diacritics
        7) Show invisible characters
        8) Barred-bowl forms
        9) Literacy alternates
        10) Small v-hook alternate
        11) Capital Y-hook alternate
        12) Capital N-left-hook alternate
        13) Small ezh-curl alternate
        14) Capital T-hook alternate
        15) Capital H-stroke alternate
        16) Capital R-tail alternate
        17) Small p-hook alternate
        18) Romanian-style alternates
        19) Capital Ezh alternate
        20) Ogonek alternates
        21) Modifier apostrophe alternate
        22) OU alternates
        23) Empty set alternate
6. Check on screen:

a. spacing

b. diacritic positioning

c. ligatures (like tone bars)

d. Check any narrow characters with diacritics that the characters next to them don't collide

7. Print all pages

8. Check printed copy (or get translator to check) for:

a. spacing

b. diacritic positioning

c. Check any narrow characters with diacritics that the characters next to them don't collide

**D. Give us FEEDBACK!**

Please use the following feedback form for reporting problems. Also include a file (Word, WorldPad, InDesign, etc) which demonstrates the problem and a .gif of what you are seeing.

Feedback on Doulos SIL Regular

Name:

Organization:

Operating system used:

Version: Service pack/update:

Applications used:

Name:

Version: Service pack/update:

Name:

Version: Service pack/update:

Name:

Version: Service pack/update:

Language(s) this font was used to represent:

For the following, please mention specific characters or font features as relevant:

What worked well?

What did not work well?

Other comments?