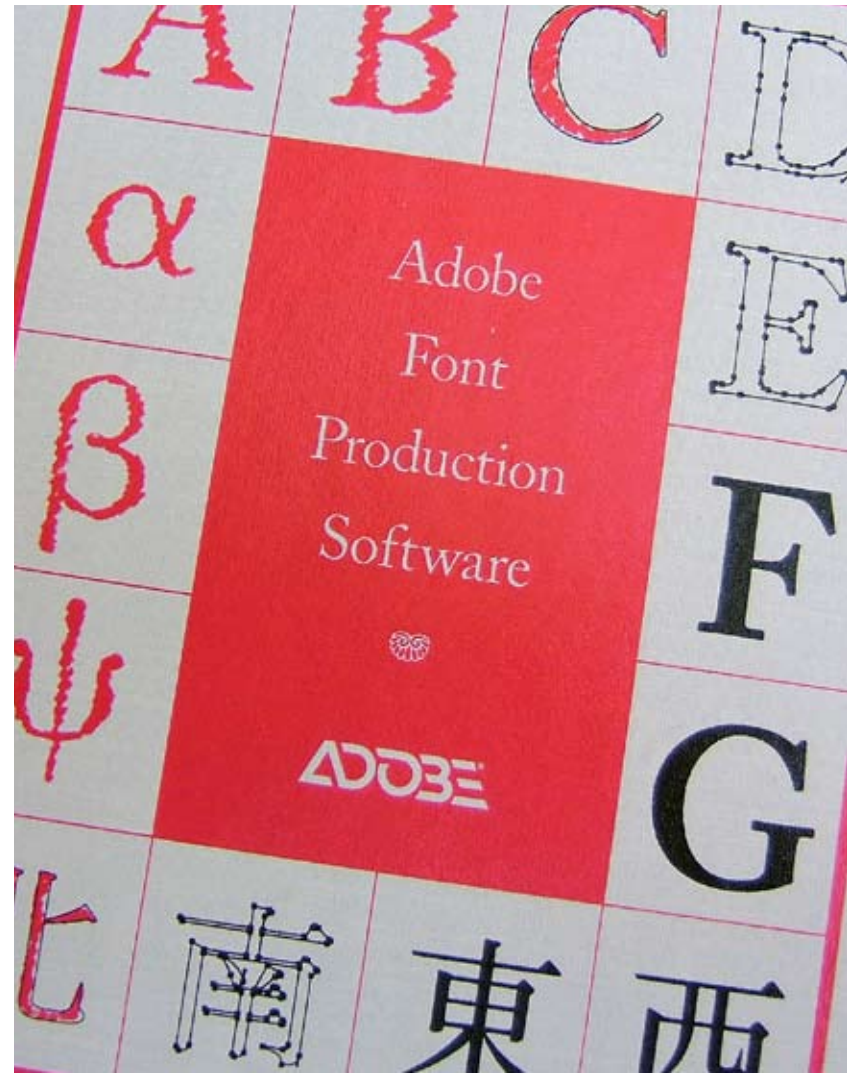


# State-of-the-art font QE

Miguel Sousa

Typeface Designer  
& Font Developer

28 April 2007



# Why? How? What's wrong?

## Font developer A

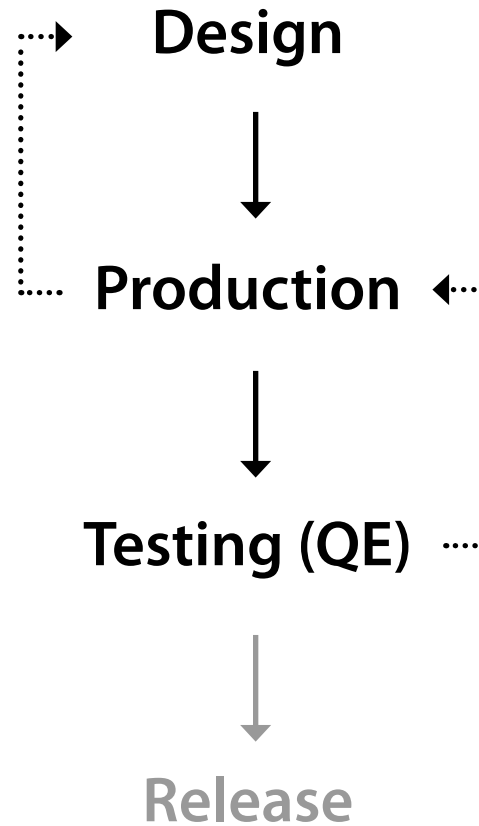
*I have setup my fonts just like Minion, but when I generate OTFs they don't work correctly. What am I doing wrong?*

## Font developer B

*I can't get a family of 8 fonts to arrange correctly in the font menu. And two of them don't even show up. It's driving me crazy. HELP!*

This presentation, and the previous — *Font production at Adobe* —, aim to answer these questions. Font development is **not** a trivial activity, but at Adobe we don't use any trade secret processes, and the tools for producing sound fonts are all publicly available.

# Font Development at Adobe



If it's not tested,  
**it's broken!!**

# Types of font testing

## White-box

Testing fonts based on knowledge of their internal structure

- outlines
- tables
- file structure

## Black-box

Testing fonts based on knowledge of their expected behavior

- installation
- rendering
- behavior in different applications

## Workflow

Test real-world scenarios in multiple applications and environments/platforms

- trivial and intricate situations
- cross-platform
- multilingual
- interaction with other software

# White-box tools

## **CheckOutlines** (in AFDKO)

This tool thoroughly checks all the glyph-outlines, and reports problems such as:

- no points at extremes
- overlapping points and paths
- extremely sharp angles
- reverse paths
- coincident BCPs
- other outline anomalies

Example:

- Outputs a list of glyph names, and problems found in the glyph's outlines

```
checkoutlines font.otf > output.txt
```

## **Microsoft Font Validator**

Tool for testing TrueType and OpenType fonts, evaluating how closely they adhere to the specification. It is particularly useful for validating the font's tables, but it's unable to rasterize CFF glyph-outlines. Requires Windows.

**Notice:** Not all warnings and error messages reported will be "real" or fatal issues. Font Validator may flag errors for well-built fonts, so its output requires some amount of informed analysis.

# White-box tools (cont.)

## **tx** (in AFDKO)

Tool for analyzing and converting the glyph-outline data of TrueType fonts and all OpenType font formats.

It can output the results in several formats, including PDF.

### Examples:

- Outputs the font's *Top- and Private Dictionary*

```
tx -dump -0 test.otf
```

- Generates an interactive PDF that displays the font's glyph set, and a glyph per page showing outlines and hinting "zones"

```
tx -pdf -1 test.otf > glyphset.pdf
```

## **spot** (in AFDKO)

This tool reports the font's OpenType data. OT tables can be output in a human-readable way (feature file syntax), and some, like GPOS and GSUB, can be shown graphically, by outputting a PostScript file.

### Examples:

- Outputs the font's OS/2 table

```
spot -t OS/2 test.otf
```

- Outputs the font's name table

```
spot -t name=3 test.otf
```

- Generates a PostScript file showing a graphic representation of the substitutions in the *liga* feature present in the font

```
spot -P liga test.otf > liga.ps
```

## White-box tools (cont.)

### **CompareFamily** (in AFDKO)

This is the most important tool of the kit. It consists of a collection of tests which cover all the things that the Adobe Type Department has ever done wrong, more than once. It is **the only tool that tests font data across all the faces in a given type family**. It checks names, style-linking, GSUB and GPOS features, BASE and OS/2 tables, glyph hints and font level hints, and several other assorted checks. Altogether it comprises 55 tests — 21 family-wide and 34 single font tests. (If a given test yields no report, it indicates there is no problem with that aspect of the font, or of the family.)

### Examples:

- Executes all the tests to the fonts inside the folder, including 3 reports — menu name, metrics name and Panose number —, and outputs the results to a text file

```
comparefamily -d /Folder_TestFont -rn -rm -rp > output.txt
```

- Executes single font test number 1 (Length overrun check for name ID 18. Max 63 characters, must be unique within 31 chars.), outputting the results to the window

```
comparefamily -d /Folder_TestFont -st 1
```

# Black-box testing

## Tests

- Does the font install/uninstall?
- Do the glyphs render on screen?
- Are the fonts correctly arranged on the font menu?
- Do the fonts style-link?
- Are the glyphs correct?
  - glyph's shape
  - advanced width and positioning
  - size and weight
  - alignment of diacritics
- DSIG verification
- Virus scan

## Tier 1 applications

- Acrobat
- FrameMaker
- Illustrator
- InDesign
- MS Word
- PageMaker
- Photoshop
- QuarkXPress
- TextEdit

## Tier 2 applications

- After Effects
- CorelDRAW
- Flash
- Freehand

# Workflow testing

## Cross-platform tests:

- Insert all characters and test different languages
- Test in foreign system configurations
- Mix all the fonts, and together with other font families
- Generate PDF files, and confirm embedding
- Process through Distiller
- Print to PostScript and non-PostScript printers
- Print from PDF
- Print at various sizes and arrangements
- Verify hinting
- Confirm the kerning is working
- Proof OpenType features
- Test glyph-augmentation through SING

## Related links

### **AFDKO (Adobe Font Development Kit for OpenType)**

- [http://partners.adobe.com/public/developer/opentype/afdko/topic\\_overview.html](http://partners.adobe.com/public/developer/opentype/afdko/topic_overview.html)

### **Microsoft Font Validator**

- <http://www.microsoft.com/typography/FontValidator.msp>

### **OpenType specification version 1.4**

- <http://www.microsoft.com/typography/otspec/>
- [http://partners.adobe.com/public/developer/opentype/index\\_spec.html](http://partners.adobe.com/public/developer/opentype/index_spec.html)

All AFDKO tools are cross-platform (Macintosh and Windows), and work with CJK fonts. The package contains several other tools not covered by this presentation, some of which are specific to CJK font development.

**Better by Adobe.™**