YOU DON'T LOVE ME YET

- **Stephan Bergmann**
  - Software Engineer
- Sun Microsystems, Inc.

YOU DON'T LOVE ME YET
—Roky Erickson

- The story so far:
  - What has changed?
  - Why did you do that?
  - And now its broken!
- The road ahead

OpenOffice.org 3

- ...comes in three layers:
  - Top layer is brand specific (plain OOo, BrOffice, StarOffice, ...).
    - The user should only have to interact with this layer.
  - Middle layer is generic OOo basis.
    - The bulk of the files goes here.
  - Bottom layer is UNO Runtime Environment (URE).
    - Could serve other applications, too.
- Multiple brand layers (OOo 3, SO 9) share a basis layer.
- Multiple basis layers (3.0, 3.1) share a URE layer.

Layers? What Layers?

- For the average user, nothing changes:
  - OOo still comes as a single installation set.
  - All the executables are still found in C:\Program Files\OpenOffice.org 3\program.
  - It still sometimes behaves silly and crashes.

Oh, Layers!

- Only when you dig deeper than the graphical frontend will you notice the changes.
  - “Where is that libvclli.so I want to replace with a freshly built bugfix version?”
  - “I can no longer debug soffice.bin from msdev!”
  - “My Python application that starts OOo in the background no longer works. What a shame.” (“Yes, I know. I'm sorry.”)
  - “I told it to install to C:\MyOOo. It actually installs to C:\MyOOo\OpenOffice.org 3. Keep it short, man!”
  - ...

Why?

- A simple, modular code base would have many benefits:
  - If you want to use only part of it, you should also only need to build, test, ship, and install part of it.
  - If you produce multiple products, you should build, test, ship, and install common parts just once:
    - Different products like URE, OpenOffice.org, OpenOffice.org SDK, BrOffice, StarOffice, StarOffice SDK, StarSuite contain common parts.
    - Some parts are independent of platform (unxlngi6, unxsols4, wntmsci10, etc.).
    - Some parts are independent of locale (en-US, de, ja, etc.).
    - Some parts do not change across versions.

(from OooCon '07)

Because!

- Concrete packages contain full paths, and some are shared across products: the shared parts can only have one path
  - no way to have the basis in /opt/openoffice.org3.0 or /opt/staroffice9
- But can't we keep everything together?
  - multiple products would be thrown together in one directory tree
    - products could not have files with identical names (e.g., soffice)
    - the products' names could not be reflected in the single tree's pathname

(from FOSDEM ’08)

A Means to an End

- It is important to remember that splitting OOo into three layers was a means to an end, not an end in itself.
  - Platforms that have no requirement for the split do not need the split (Windows, Mac OS X).
  - However, it is also considered important to keep OOo and its source code as uniform as possible across the various platforms.

The Good, the Bad, and the Ugly

- On average Unix (Linux, Solaris: ELF, packages like RPM, DEB), the split works well.
  - Ready to reap the benefits.
- On Mac OS X with a single dmg installation file, there comes no benefit from splitting.
  - Still internally has three layers (for consistency).
  - Move to other packaging formats in the future?
- On Windows, splitting apparently works against the platform.
  - DLL hell, here we come!

Packages (Linux, Solaris)

- Different products (OOo, BrOffice, StarOffice, ...) share packages with identical names and identical content.
  - At the moment, packages are still built multiple times.
  - At the moment, should-be identical packages sometimes still vary slightly.
- I just discovered Novell publishes identically named packages with (I assume) slightly different content.
  - Can cause problems if you install multiple products on one machine.
  - How shall we handle this?

ELF Libraries (Linux, Solaris)

- Symbolic links between layers (basis-link, ure-link).
- RPATH $ORIGIN/../basis-link/program and $ORIGIN/../ure-link/lib in OOo ELF files will find libraries.
- Extension UNO components are guaranteed to have available preloaded URE libraries without having to specify where to find them.

Mac OS X Libraries

- Symbolic links between layers (basis-link, ure-link).
- @loader_path/../basis-link/program and @loader_path/../ure-link/lib in Mach-O files will find libraries.
- Extension UNO components are guaranteed to find URE libraries through @executable_path/urelibs symbolic link.

Windows Libraries

- Fake symbolic links between layers (basis-link, ure-link text files).
- Wrapper executables extending PATH ensure EXEs and DLLs (including extension UNO components) find libraries.
  - Prior experiments with mis-using /DELAYLOAD hooks failed.
  - Extending PATH fails when system directories happen to contain DLLs with same names (C:\Windows\libxml2.dll comes before URE layer libxml2.dll from PATH in Windows DLL search order).
    - Mainly a problem with external DLLs (libxml2.dll, python23.dll).

Windows Assemblies

- An assembly bundles one or more DLLs.
  - It has a unique tag (name, version, publicKeyToken).
  - A DLL/EXE has a manifest declaring that it depends on a specific assembly version.
- Private assemblies are part of an application installation.
  - They must reside in the same directory as the EXE.
- Shared side-by-side assemblies are available globally.
  - They must reside in C:\Windows\WinSxS.

Shared Side-by-Side Assemblies

- Turn libxml2.dll into a side-by-side assembly:
  - Add a libxml2.manifest stating “this is version 1.0.0.0 of libxml2.dll used by OpenOffice.org.”
  - Specify /MANIFESTDEPENDENCY when linking against libxml2.dll.
  - When installing OOo, install libxml2 assembly into C:\Windows\WinSxS, instead of installing libxml2.dll into C:\Program Files\OpenOffice.org 3\URE\bin\libxml2.dll.
    - We already do that for the Microsoft.VC90.CRT assembly.
  - During building (and for smoketest OOo installations done while building), ensure libxml2 assembly is present as private assembly in every directory where it is needed.

Really!?

- Do that for all external DLLs (< 40).
- Things become more and more baroque.
- Are we working against the platform? (An executable and its DLLs typically are in the same directory on Windows.)
- Or is this the way the external libraries should already have been packaged in the first place, by their respective maintainers?

Layer Free Windows?

- For some time, we shared layers on Windows, too.
  - Abandoned it as it had no real benefits.
  - Now every product contains private copies of all three layers (similar to Mac OS X).
- Lumping together the three layers would not be trivial, however.
  - For example, unify $URE_LAYER/bin with $BASIS_LAYER/program without symbolic links!
- Cross-platform consistency has its merits, too.

Was It Worth the Pain?

- If nothing else, it forced the code to be more explicit about where it finds things.
  - Similar to the infamous Java class path cleanup (which unearthed gems like Thread.getContextClassLoader).
  - Still, I at least consider being explicit a Good Thing. (“You pedantic party-pooper!”)
- After all, it was a necessary first step for the specific way to modularity we chose to walk on.

Explicit dlopen

- dlopen(“foo”) with no slash in filename works in implementation-defined manner:
  - Relative to calling code's location on Linux, Solaris.
  - Relative to current working directory on Mac OS X.
- rtl_loadModule(“foo”) calls (i.e., dlopen(“foo”) calls) in OOo source code failed for Mac port.
  - Mac port added complicated patch to rtl_loadModule.
- Calls failed on all platforms after layer split.
  - rtl_loadModule now requires a full path.
  - No complicated patch for Mac OS X needed any more.

Onwards

- Get Windows DLL handling under control.
- Reorganize the build process so that packages for multiple products are only built once.
  - That is, finally bring in the harvest.
- Reorganize the build process so that independent parts can be built independently.
  - Can ease the experience of working on OOo code for newcomers and experts alike.
  - Novell is also working on this.

I LOVE YOU SO MUCH IT HURTS

—Floyd Tillman