AbiWord:

A Cross–Platform GNOME Office Component

Dom Lachowicz <doml@appligent.com>
Hubert Figuière <hfiguiere@teaser.fr>

Abstract

AbiWord is a cross platform Word Processor, but it is also GNOME's word processor. The future of AbiWord is definitely related to the future of the GNOME project as a whole, and to GNOME Office in particular.

The future of GNOME Office looks promising, with the leading hackers of the Gnumeric, AbiWord, and Guppi team resolving to tackle the many tough issues that lie ahead.

AbiWord's Goals and Ambitions

AbiWord aims to be the word processor of and for the masses. AbiWord does its best to inter–operate with the user's native environment and its existing applications, toolkits, and file formats while providing an incomparable level of quality, service, and performance. AbiWord achieves this through:

1. A unique approach to cross–platform application development
2. A very robust import/export architecture, with a profound focus on inter–operability with the many existing word processors on the market today
3. An ever–growing array of plugins, capable of everything from providing new image loaders to inline translations to dictionaries and thesauri.

Glossary

Here is a short list of acronyms we use that may be using and confusing:

- **AP**: designates application framework code
- **XAP**: designates cross−application framework code
- **XP**: designates cross−platform code

The Future of AbiWord

At the time of this writing, AbiWord is nearing its **1.0** milestone release. While this is a very important milestone for the team and the project, it is only the first of many steps toward achieving the complete feature set that a user expects and deserves. The next milestone release – **1.2** – will add several long awaited features. Here's a peek at what the developers have on their white–boards:

Tables

Table support has probably been our most requested feature and something that we will definitely address in our 1.2 series. This involve reworking our text formatting engine. That leaves entirely in the XP code. Changing it will benefit to all platforms quite immediately.

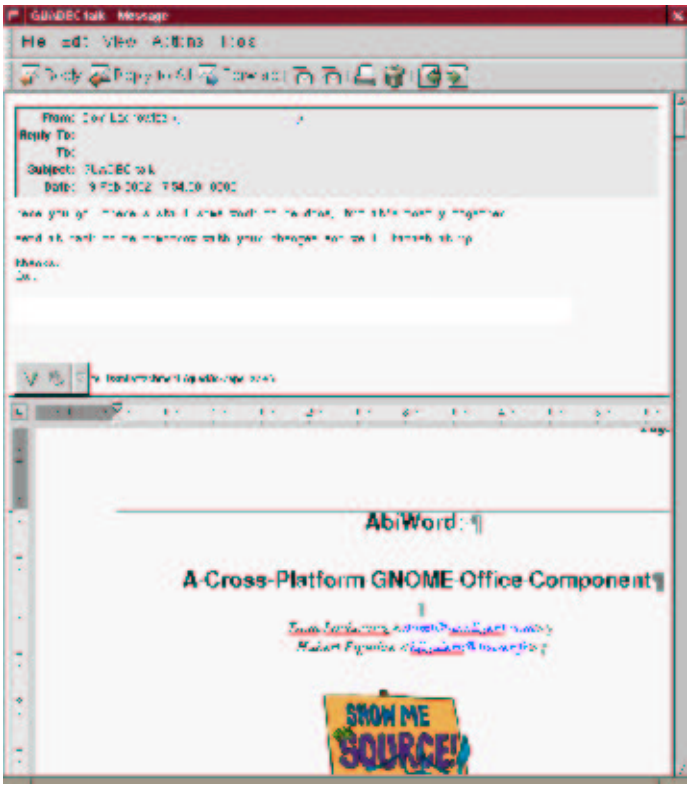
Compound object embedding

After tables, object embedding is the feature that most users ask for. With AbiWord 1.2 it should be possible to embed and edit equations, spreadsheets, graphs, and other data

AbiWord as an embeddable component

OEOne recently demonstrated that AbiWord can be embeded as a XPCOM component inside of their Mozilla–based desktop. We'd like to leverage Bonobo and OLE2 to make AbiWord an embeddable component. The über–cool side effect – you'll be able to do inline previews of AbiWord or MS–Word documents inside of Evolution and Nautilus.

Here are a few application examples of AbiWord embedding.



The biggest problem will be to keep embedding and embeddability as much in the XP world as possible.

The Future of GNOME Office

With GNOME 2.0 just recently released, much effort is going into porting existing applications to this tremendous new platform. AbiWord and the other GNOME Office applications are no exceptions. Not only are these applications being "ported' to the new framework, but they are also being re−architected to leverage the many wonderful new controls, widgets, and technologies that it provides. Some of the more important technologies are:

- Bonobo
- GConf
- GTK+2

AbiWord 1.2 will strive to utilize many of these technologies to its advantage.

The AbiWord team will also be working closely with the Gnumeric team in order to create a very solid, robust, full−featured, and well−integrated office environment for the GNOME desktop. We endeavor to work on:

- An escher drawing model (new canvas)
- Better support for popular MS Windows graphic types (WMF and EMF included)
- Better support for popular MS Windows file formats (DOC, XLS, ...), including LibOLE2 − "The child that nobody wanted"
- Cross−Application embedability
- Shared dialogs and other controls (toolbars, menus, font selection dialogs, etc...)
- Improved font handling
- Improved printing support

AbiWord's Cross–Platform Ambitions

AbiWord has been designed from the ground up as a cross–platform application that would run on every imaginable platform.

Why cross platform?

This question might sound strange, but why would anyone develop a free (libre) software product to run on non free platforms as well as on free platforms? The AbiWord team does this because we whole–heartedly endorse **both** freedom of software and freedom of choice. If either choice or necessity demands that a user must use proprietary operating environment, s/he is still free to use AbiWord there.

The framework

The AbiWord team has developed its own cross–platform C++ framework. Abi's framework is much unlike other cross–platform developments like OpenOffice, Mozilla or even Qt. Unlike these others, Abi's framework is purposefully designed to integrate well with a user's native system, primarily because Abi leverages OS system services to its advantage. In layman's terms, Abi will use native controls, widgets, filesystem drivers, plug–in frameworks, printers, etc... instead of providing its own. This, of course, is not without its costs. The platform maintainers ultimately have more code to maintain, but the result (and our ultimate asset) is that Abi really is a native application everywhere it runs. It looks and behaves just like any other application on your desktop.

The approach taken is to separate interfaces from implementation. We provide a set of cross–platform interfaces and abstract classes which are then implemented in terms of native controls.

What we do:

- Abstract drawing canvas: we don't reimplement our own rasterizer but instead adapt our calls to draw onto native canvases
- I18N and L01N: Our framework is highly adaptable and portable to a large variety of languages and locales, including Arabic, Hebrew, and CJK locales.
- Event mapper/dispatcher: AbiWord's native classes trap native events and actions and then dispatch them back to XP classes. Key presses, mouse motions, and other user–interaction events are bound from the platform event system to the framework events.
- Abstract controls: The AbiWord framework abstracts or implements some important widgets such as font combo boxes, status bars, and rulers.
- Activate and manage dialogs
- ... and of course provide a cross–platform text formatter and management on top of that.

What we don't do:

- Manage the widget creation and layout. The reason is that widget management is mostly dependent of the toolkit, and each toolkit offer tools to do it easily. And since we don't...
- ...Rewrite a complete graphics toolkit like some others that we're familiar with <*cough*> GTK+, QT, OpenOffice, Mozilla <*/cough*>. This is too much work for too little benefit, and the drawbacks are huge. Look at the overhead of XUL in Mozilla, the size of the framework in OpenOffice and you'll see the benefits of our approach.
- Render fonts: since this is considered a graphic primitive, we let each platform do that by itself.

This means that the platform implementor has to provide:

- Bindings between GUI events and the XP framework
- GUI layout for the word processor frame
- Ready made dialogs
- Graphics primitives
- Other platform specific features (fonts, printing, ...)

Despite (or perhaps because of) all of that, we are highly portable. For instance, we run on more platforms than OpenOffice does. For that matter, only "Hello World" and Mozilla run on more platforms than AbiWord does ;–)

The ultimate catch

Our framework is not only cross–platform, but it aims to be cross–application as well. So this means that many of the lower–level classes are potentially available to any new application. These parts will "just work™" in their new application for **every** platform that AbiWord supports. This includes such things as:

- Toolbars
- Menus
- Graphics and font handling
- Printing
- A multitude of dialogs

Question: why not port an app like Gnumeric to other platforms using the Abi framework?

How does that work?
First we separated the framework in 2 parts:

1. The cross–application framework (**XAP**)
2. The application framework (**AP**)

The text engine is separate as it is a really independent engine. The cross–application framework provides general purpose classes that could be useful for any application. The application framework specializes those classes to match the application specific needs.
Here is an example, the frame (ie the main window). The classes inheritance scheme is as follows:

`XAP_Frame -> AP_Frame`

XAP_Frame, an interface class, loads its own platform implementation class, and AP_Frame can (a possibly will) do the same. AP_Frame implementation class should inherit from XAP_Frame implementation which inherit from a base virtual implementation class

`XAP_FrameImp -> XAP_GnomeFrameImp -> AP_GnomeFrameImp`

Those implementation classes are only used as a delegate or proxy that will abstract all the platform specific implementation code. We have bidirectional relations between the implementation and the XP class.

The architecture is slightly different from what we do in version 1.0. Version 1.0 framework is not as clean as it sounds, and even if it features lot of XP code, it partly lacks separation of interface and implementation.

How to Help AbiWord

You can help AbiWord development through a number of ways:

1. Help us out with maintaining Bugzilla
2. File bugs and perform regression and stress tests
3. Help out with "bug−days"
4. Help answer questions on the abiword−user list
5. Write user and developer documentation
6. Translate or update an existing translation
7. Contribute code
8. Sponsor a developer
9. Port Abi to your favorite platform (eg: Qt/KDE, MacOS 9)

All of this and more is explained at http://www.abisource.com/howtohelp.html

Conclusions

AbiWord 1.2 is poised to become one of the best word processors ever and run on an unparalleled number of platforms and architectures. It will be more than ever an integral part of GNOME Office.

Questions?