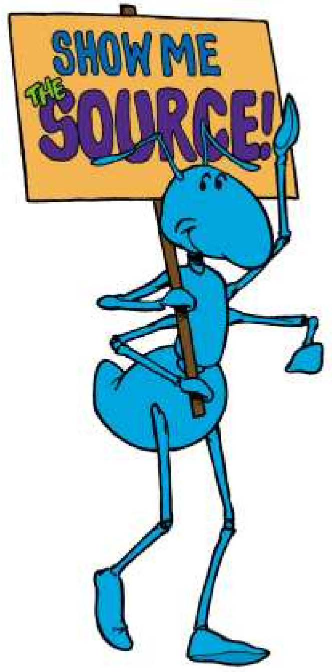
AbiWord 2.0 - "The Wrath Of Dom"

by Martin Sevior and Dom Lachowicz



Abstract

AbiWord is a cross platform Word Processor and is GNOME's word processor.

The future of AbiWord depends on 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 and AbiWord team resolving to tackle the many tough issues that lie ahead.

Outline

Brief History of AbiWord

How it works

What we did since 1.0

Cool new stuff in 2.0

Demonstration of AbiWord-2.0

Questions?

History of AbiWord

Started by some people who are now SourceGear in 1998.

Jeff Hostetler, Eric Sink, Paul Rohr, Bob Sievers

Abandoned except for Paul and Bob in 1999.

Dom and Martin took over as maintainers in 2000.

1.0 released in 2002.

2.0 Real Soon Now$^{(TM)}$.

AbiWord's Goals and Ambitions

"Word Processing for Everyone"

AbiWord aims to be the word processor of and for the masses.

The most powerful, easiest and most pleasant way to write stuff.

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:

A unique approach to cross-platform application development

A very robust import/export architecture, with a profound focus on inter-operability with the many existing word processors on the market today

An ever-growing array of plugins, capable of everything from providing new image loaders to inline translations to dictionaries and thesauri.

AbiWord Internals

Is a C++ program with a large effort to abstract and re-use code.

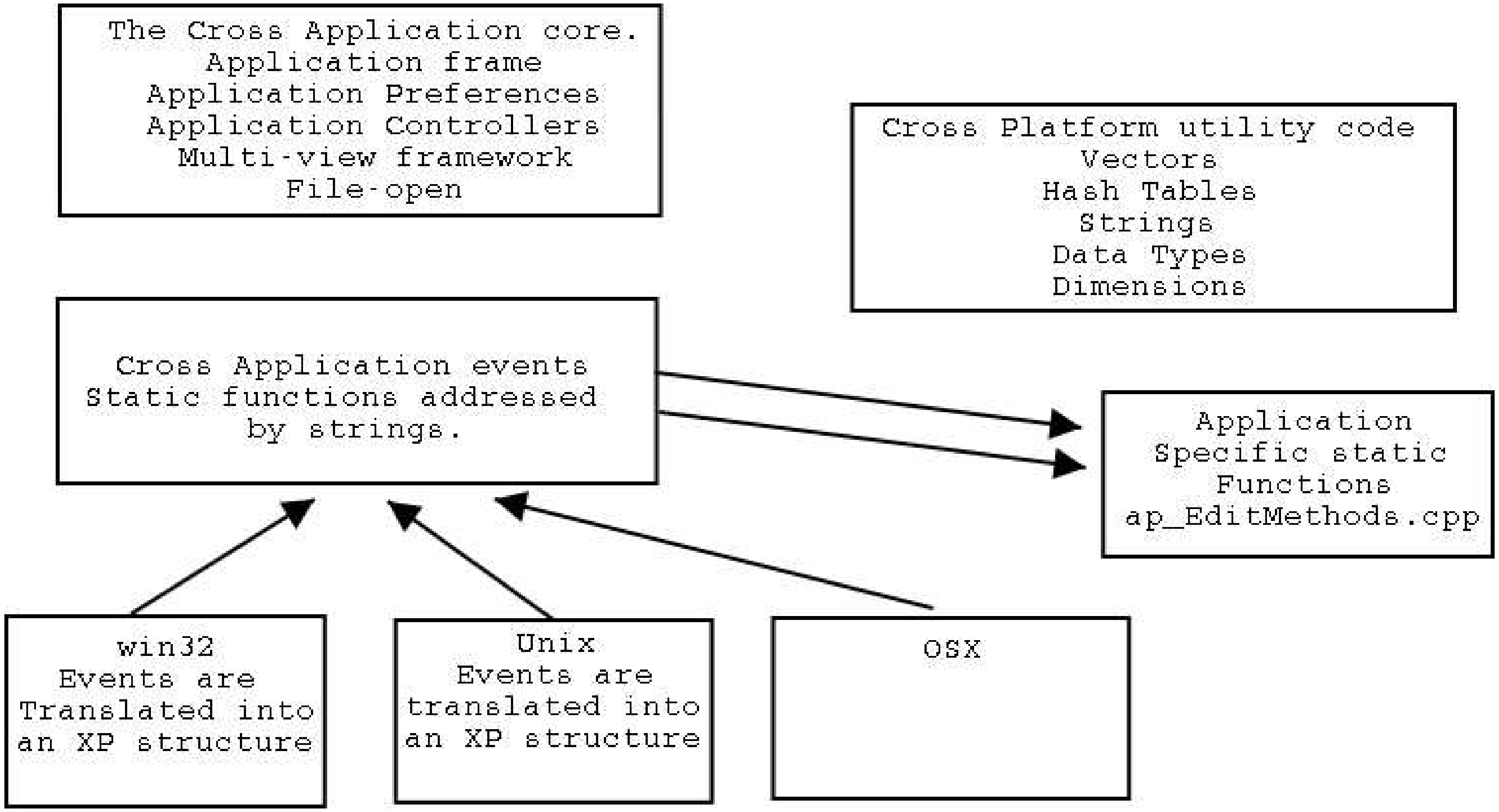
It has a cross Application Architecture (xap), cross platform architecture (xp)

AbiWord was designed to be the first of a full suite of applications.

A large fraction of the code was identified as being useful to other applications.

This code was placed in (xap) classes.

Examples include the event logic, much of the code for dealing with windows and frames, many dialogs.


The Cross Application core.
Application frame
Application Preferences
Application Controllers
Multi-view framework
File-open
Cross Platform utility code
Vectors
Hash Tables
Strings
Data Types
Dimensions
Cross Application events
Static functions addressed
by strings.
Application
Specific static
Functions
ap_EditMethods.cpp
win32
Events are
Translated into
an XP structure
Unix
Events are
translated into
an XP structure
OSX

The code snippet opposite shows how events are mapped into actions.

(This case the Backspace command)

The events mapped to static functions makes it very easy to extend AbiWord.

One simply defines new static functions and adds them to the list methods.

Makes it easy to extend write plugins.

The menu and toolbar code is similar.

Menu and toolbar items are defined in XP structures and then built with platform code.

Menus and Toolbars are easily configurable.

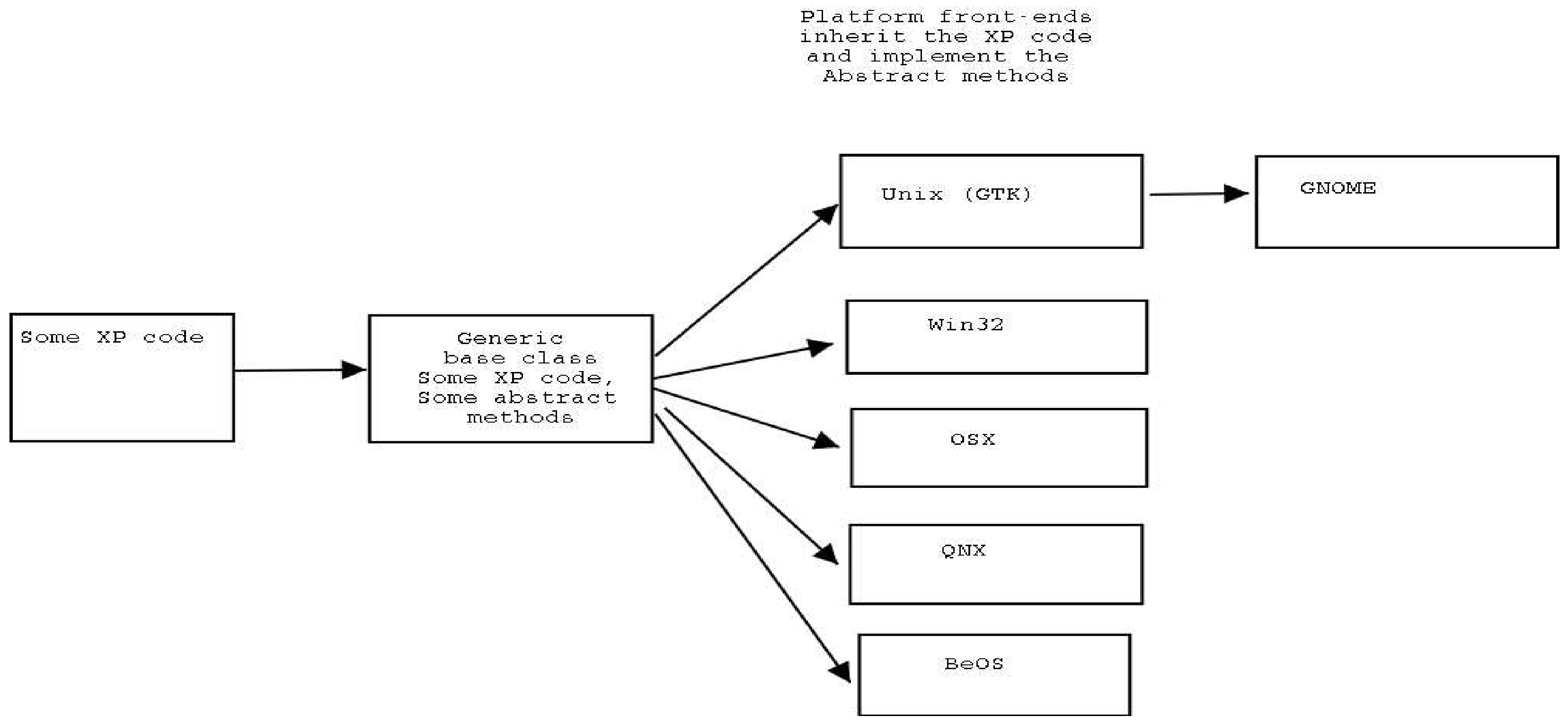
```
  static EV_EditMethod_Fn delLeft;
.
.
static EV_EditMethod s_arrayEditMethods[] =
{
#if defined(PT_TEST) || defined(FMT_TEST) || defined(UT_TEST)
    EV_EditMethod(NF(Test_Dump),            0,  ""),
    EV_EditMethod(NF(Test_Ftr),         0,  ""),
#endif


    // a
    EV_EditMethod(NF(activateWindow_1),     0,      ""),
    EV_EditMethod(NF(activateWindow_2),     0,      ""),
    .
   EV_EditMethod(NF(delLeft),              0,  ""),
}
.
     #define Defun1(fn)  bool F(fn)(AV_View*   pAV_View,
EV_EditMethodCallData * /*\pCallData*/)


Defun1(delLeft)
{
    CHECK_FRAME;
    ABIWORD_VIEW;
    pView->cmdCharDelete(false,1);
    return true;
}
```

The Cross Platform Architecture

AbiWord makes extensive use of abstract virtual methods in base classes to implement platform specific code.


Platform front-ends inherit the XP code and implement the Abstract methods
Some XP code
Generic base class Some XP code, Some abstract methods
Unix (GTK)
GNOME
Win32
OSX
QNX
BeOS

Example - The Graphics class

The base class gr_Graphics has both virtual methods and real code.

The code snippet opposite is from the header file of the gr_Graphics base class.

"findNearestFont" looks up a hash table of cached fonts implemented for all platforms.

The other methods in the class are pure virtual which are implemented in the platform code.

The code for drawing with the Win32 API is totally different from GTK.

```
virtual GR_Font*  getGUIFont() = 0;

virtual GR_Font*  findFont(const char* pszFontFamily,
                           const char* pszFontStyle,
                           const char* pszFontVariant,
                           const char* pszFontWeight,
                           const char* pszFontStretch,
                           const char* pszFontSize) =0;
static const char* findNearestFont(const char* pszFontFamily,
                           const char* pszFontStyle,
                           const char* pszFontVariant,
                           const char* pszFontWeight,
                           const char* pszFontStretch,
                           const char* pszFontSize);

virtual void      drawChars(const UT_UCSChar* pChars,
                           int iCharOffset,
                           int iLength,
                           UT_sint32 xoff,
                           UT_sint32 yoff,
                           int* pCharWidths = NULL) = 0;
```

The net result of the approach used by AbiWord is that the application is a native application on each target platform

pure win32 program on win32, pure Gtk on Unix

can easily implement GNOME architecture to overide Gtk as needed.

Complete separation of GUI from backend.

Leads to a modular and stable platform despite huge complexity.

Downside is that platform developers are needed.

Quite resource intensive.

Text and other objects in AbiWord

AbiWord uses a Model-View-Controller architecture.

The "Model" is the PieceTable.

The "View" are the Layout classes.

The "Controller" is split between document level manipulations and manipulations on the view via the insertion point.

The PieceTable is a doubly-linked list of objects that contain pointers to text, data and const char * text strings which represent attribute-value pairs.

These attribute-value pairs can be arbitrarily complex.

Represent the attributes of the text (font, font-size, sub/superscript etc)

Represent the attributes of the Text structures like (paragraphs, sections, table, cells etc)

You can get a feel for this by browsing *.abw files. (These are basically XML-ized dumps of the piecetable.

This little snippet of an *.abw shows how a simple 2x2 table is defined.
<table> represents a table structure fragment
<cell ....> represents a cell structure.
Notice the list of props=.....
These are key-value pairs.
The values assigned to bot-attach, top-attach etc tell the Layout classes where to place the cell in the table

```
<table>
<cell props="bot-attach:1; left-attach:0; right-attach:1; top-attach:0">
<p style="Normal">Cell 1</p>
</cell>
<cell props="bot-attach:1; left-attach:1; right-attach:2; top-attach:0">
<p style="Normal">cell 2</p>
</cell>
<cell props="bot-attach:2; left-attach:0; right-attach:1; top-attach:1">
<p style="Normal">cell 3</p>
</cell>
<cell props="bot-attach:2; left-attach:1; right-attach:2; top-attach:1">
<p style="Normal">cell 4</p>
</cell>
</table>
```

Layout Classes

There is only one Piecetable per document but each document can have any number of views.

The Views are created by the Layout classes, which interpret the attribute-value pairs, text and data in the PieceTable and create formatted text on the screen.

Each Structure fragment in the PieceTable has a vector of pointers to corresponding Layout classes.

Changes to the PieceTable are recorded in Change Record classes.

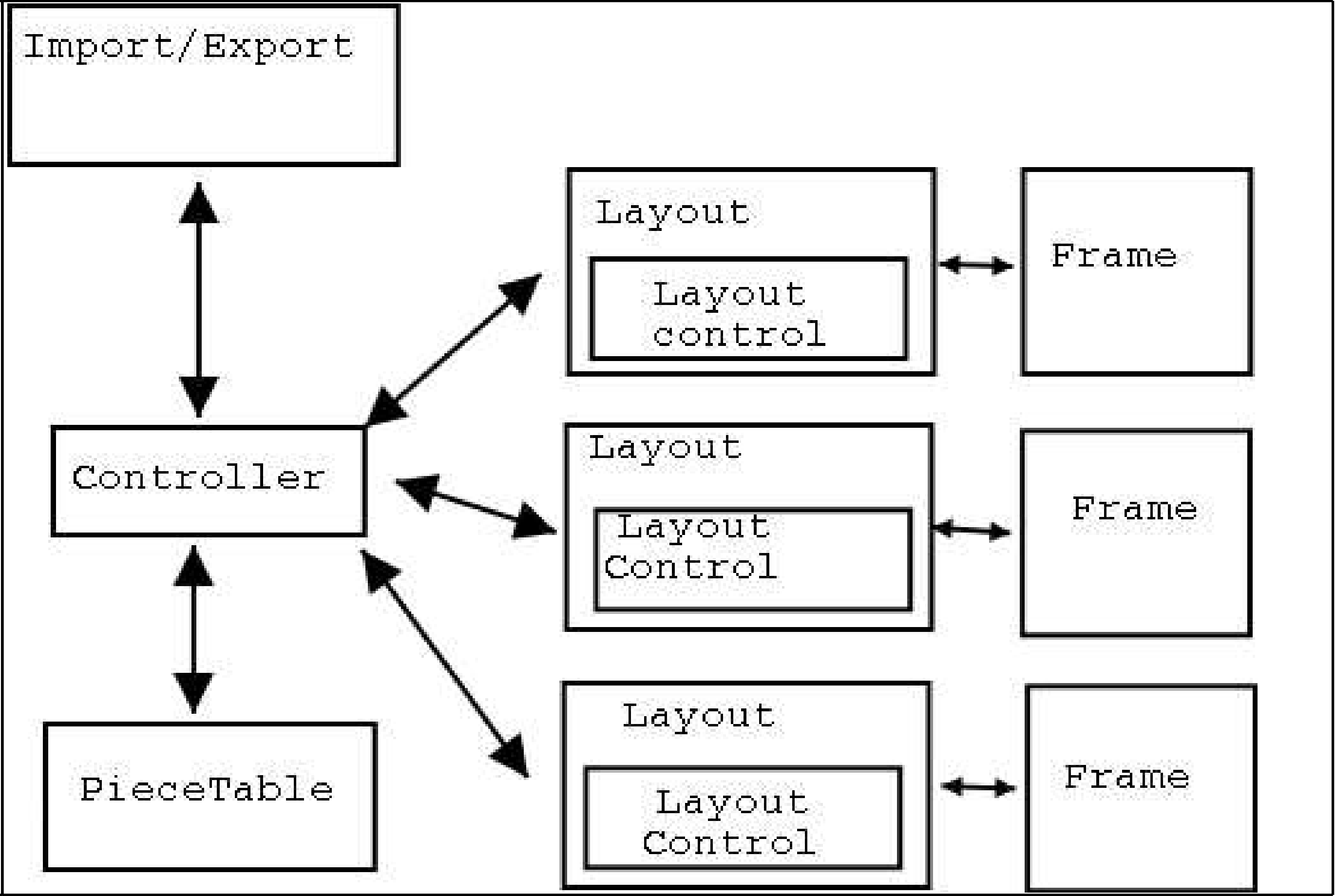
These are both recorded in a Stack for Undo and broadcast out the Layout classes.

The MVC arrangement of AbiWord

Documents are imported/exported directly through the controller

Users interact firstly with the frame, then the layout control.

PieceTable changes are broadcast to all Layouts via the controller



What we've done since 1.0

Refactored the Layout classes to enable complex structures like tables and footnotes.

Extended the PieceTable to allow these structures.

Ported to GTK 2.0, 2.2, 2.3, GNOME2

Use FreeType via XFT2 to draw hardware accelerated anti-aliased text.

Use FontConfig to find all the fonts on a users system (also allows TTF's)

- Other big internal changes to simplify our hacking life (Single Units, Cursor Class, Frame Refactoring).
- OSX Port!
- Us libglade for dialog generation (HIG compliance)

All these are internal changes to enable our new features.

Many of these were hugely de-stablizing.

Still ironing out bugs.

But this hard work is a good investment.

New features will flow more easily now.

What's new for Users in 2.0?

Tons of big new features and bug fixes

- Tables! (Able to reuse GTK Table algorithm)
- Footnotes.
- Endnotes.
- Mail-Merge.
- New and vastly improved importers/exporters.

- Can import and work correctly with very complex documents from MS Word, RTF, Word Perfect.
- Export to HTML. (Now a nice little WYSIWYG HTML generator)
- Revision Marks.
- Cool new and improved plugins. (Text summarizer, Python Plugin, Google Search, Import/Exporters)
- Nautilus View/Bonobo Control

GTK2 interface.

HIG compliant UI

Lots of polish

Nice new art-work from XD2

Font Preview before selection.

Resize images interactively

Drag and drop images

Selection color from GTK theme

Interactive Insert Table widget.

More non-Modal dialogs

The Command line

AbiWord embraces the Unix platform by providing numerous command-line options.

These enable power-users to get more done quickly and...

Provide an easy to-use way to do server-side document manipulation.

Some examples of AbiWord "Back Office" usage are:

- As a Document conversion server
  - eg: serve up MS Word documents to web browsers as HTML
- As a Print server:

Do Mail Merge and generate postscript for printing on the fly.

Convert documents to PDF

Manipulate AbiWord programmatically via a console-like interface (AbiCommand plugin)

No need for a running X server!

Extend AbiWord with plugins.

AbiWord has a powerful plugin architecture. Some of the existing plugins are:

aiksaurus	A powerful and unique thesaurus.
abigimp	Use the GIMP image processing software to manipulate images
Text Summarizer	Choose this option to summarize the text of your document.
Wikipedia	Select some text and search the wikkipedia online encyclopedia.
freetranslation	Automatically translate text. your text into the language of your choice.
abicommand	Powerful and extensible command line interface to AbiWord.
gypsython	Use a python interface to interface directly AbiWord's internal API.
urldict	Select a word and then search an online dictionary for it's meaning.
gda	Interface into the GNOME database program Mergeant.
scripthappy	Run external programs and scripts from within AbiWord.
import/export filters	filters for applix,bzip2 abiword files, clarisworks, coquille, docbook, eml, Human readable ascii text, html, kword, latex, mif, OpenWriter, pdb (Palm), psion, pw (Pathetic writer), sdw, t602, wml, wordperfect,xhtml, xsl-fo

Embed AbiWord.

AbiWord-2.0 can be used as embeddable bonobo component.

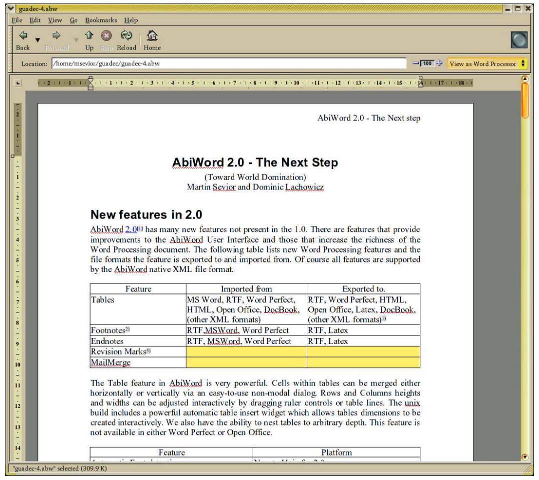
The Component is fully editable.

This gives GNOME programmers a powerful word processor they can plug into their application with just a few lines of C!

Can also be used as Viewer.

As Nautilus View it allows quick browsing of documents.

Might be worth making part of the GNOME platform


guadec-4.abw
File Edit View Go Bookmarks Help
Back
Forward
Up
Stop
Reload
Home
Location: /home/msevior/guadec/guadec-4.abw
100
View as Word Processor

AbiWord 2.0 - The Next step

AbiWord 2.0 - The Next Step

(Toward World Domination)
Martin Sevior and Dominic Lachowicz

New features in 2.0

AbiWord 2.0[1] has many new features not present in the 1.0. There are features that provide improvements to the AbiWord User Interface and those that increase the richness of the Word Processing document. The following table lists new Word Processing features and the file formats the feature is exported to and imported from. Of course all features are supported by the AbiWord native XML file format.

Feature	Imported from	Exported to.
Tables	MS Word, RTF, Word Perfect, HTML, Open Office, DocBook, (other XML formats)	RTF, Word Perfect, HTML, Open Office, Latex, DocBook, (other XML formats)1)
Footnotes2)	RTF,MSWord, Word Perfect	RTF, Latex
Endnotes	RTF, MSWord, Word Perfect	RTF, Latex
Revision Marks3)		
MailMerge		

The Table feature in AbiWord is very powerful. Cells within tables can be merged either horizontally or vertically via an easy-to-use non-modal dialog. Rows and Columns heights and widths can be adjusted interactively by dragging ruler controls or table lines. The unix build includes a powerful automatic table insert widget which allows tables dimensions to be created interactively. We also have the ability to nest tables to arbitrary depth. This feature is not available in either Word Perfect or Open Office.

Feature	Platform

"guadec-4.abw" selected (309.9 K)

The Future of GNOME Office

Much effort has gone into porting existing applications to the tremendous Gnome 2 platform. AbiWord and the other GNOME Office applications are no exceptions. Not only have these applications been "ported' to the new framework, but they were also 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

LibGSF

AbiWord 2.0 strives to utilize many of these technologies to its advantage.

Gnome Office

The AbiWord team has worked with the Gnumeric team in order to create a very solid, robust, full-featured office applications for the GNOME desktop. We endeavor to work on:

Better support for popular MS Windows graphic types (WMF and EMF included)

Better support for popular MS Windows file formats (DOC, XLS, ...), using LibGSF

Cross-Application embeddability

Shared dialogs and other controls (toolbars, menus, font selection dialogs, etc...)

Improved font handling

Improved printing support

The Future.

There are many additional features that we intend to add to AbiWord.

- Handle equations and maths.
- Additional table features (Select columns, split cells, sort tables, text to tables, repeated rows on pages, copy and paste rows and columns).
- Do genuine text wrapping around objects.
- Position objects on pages.
- Positioned Text frames.
- Paragraph borders.
- Draw on top on background images.
- More sophisticated Lists.
- Embedded bonobo and (maybe mozilla) objects.
- Genuine Scalable Graphics support.
- More and improved import/export filters. (Martin Sevior intends to build a latex importer)
- Finish HIG compliance.
- Auto-correction framework

- Auto-completion
- Grammar checker
- Text substituion: eg. (c) => ©

Use libgsf which will enable use to use the large variety of gnome-vfs backends. With this in place we can stream documents to and from AbiWord using a variety of internet protocols.

With these in place and continued improvements in our filters we should meet 99% of everyones Word Processing needs.

(Certainly meet mine!)

How you can help AbiWord

You can help AbiWord development through a number of ways:

- Use it! Tell others to use it. Send *.abw docs to people!
- File bugs and perform regression and stress tests
- Help us out with maintaining Bugzilla
- Help out with "bug-days"
- Help answer questions on the abiword-user list
- Write user and developer documentation

Translate or update an existing translation

Contribute code

Sponsor a developer

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-2.0 is an extremely useful program already.

Elegant design, very fast.

Easy to use

Get's out of your way

Being deployed at 10,000 seats at Crowell Systems and their clients

Developers are absolutely committed to continued development.

Even faster progress after 2.0

Questions/Comments