Ximian Evolution™: The GNOME Groupware Suite

Ettore Perazzoli
Ximian, Inc.
ettore@ximian.com, http://www.ximian.com

Abstract

This paper describes Evolution, a complete, extensible groupware application for the GNOME Desktop Environment.

1 Introduction

One of the big gaps between the world of proprietary operating systems (such as Windows and MacOS) and the free software world has been the lack of a decent groupware application to allow users to handle mail, appointments, to-do lists and contacts in an integrated way. While applications such as Lotus Notes and Microsoft Outlook are very popular and widely deployed on the Windows and MacOS platforms, for a long while there was simply no application that could match their functionality on free operating systems such as GNU/Linux and BSD. That is why Evolution was born.

Originally started as the "GNOME Mailer" project by the GNOME community and currently developed by Ximian, Evolution is meant to fill such gap, and provide users with a solution that is competitive, and even better to many respects, than those used in proprietary operating system.

2 Design goals

The following design goals are the foundation for the design and implementation of Evolution:

- *Versatility.* Evolution does all the things that a modern groupware application is supposed to do; it manages contact information, appointments and mail, and integrates this functionality in an easy-to-use, integrated package. Evolution acts as the central point of control for all of the user's communication needs.

- *Compatibility.* Evolution tries to support as many protocols and standards as possible, to facilitate integration into existing environments.

- *Component-based design.* Evolution is not a monolithic application; it extensively uses the Bonobo[1] technology so that its various component can be easily accessed and reused from outside Evolution. Also, the design is extensible so that it is very easy to implement new components or add new functionality on top of the existing framework. There are many ways in which a groupware application can be customized to suit the needs of a company, and Evolution tries to make customization as easy as possible.

- *Integration.* By providing public CORBA interfaces to the core functionality of its components, Evolution can be easily scripted and integrated with other desktop applications.

- *Scalability.* Evolution is meant to handle the needs of today's mail users. It is rather typical, for a hacker, to be subscribed to a number of mailing lists and consequently receive a huge amount of mail; so one of the primary goals of the Evolution mail component is to be able to deal with huge loads gracefully, and give users tools to nicely organize, read and search through their mail in an efficient way.

- *Freedom.* Evolution is free software, released under the GNU General Public License. Its source code is developed on the GNOME CVS

[1] `http://www.ximian.com/tech/bonobo.php3`

server[2] and is publically available at any time to anyone who wants to contribute or simply wants to look at it.

3 Shell and Components

Information in Evolution is organized into folders of different types; Evolution itself is made up of different components, each of which is able to handle a certain type of folder.

The various components are managed by a container component called the *shell*. The shell is what the user actually invokes when he launches the application, and its main purpose is to provide a framework through which the components can live together and cooperate. It also provides all the user interface for the functionality that is not component-specific, such as the shortcut bar, the folder management commands (such as copy, move and rename), drag and drop, and so on.

While the shell handles the framework, it's actually the Evolution components that manipulate and display the contents of the folders. Communication between components and between the components and the shell happens by using Bonobo and consequently CORBA interfaces.

It is interesting to note that the various pieces of Evolution are currently out-of-process components, and that CORBA deals with the inter-process communication nicely and transparently; it would be possible to turn these components into shared libraries without any substantial changes to the code.

3.1 Defining an Evolution Component

The basic CORBA interface that an Evolution component has to implement is called `GNOME::Evolution::ShellComponent`. When the shell is activated, it performs a query through the GNOME component activation service (which is called OAF[3]), looking for all the objects that implement that interface. These objects are then activated and initialized.

[2] A web interface to the GNOME CVS server is accessible at `http://cvs.gnome.org`

[3] Object Activation Framework

Each component can then register a certain number of folder types, and is responsible for handling them. For example, the mail component registers a `mail` folder type, and the addressbook component registers a `contacts` folder type.

Multiple folder types can be registered by the same component; the component is also responsible for creating the physical folders of the type it registers, and for copy, move and delete operations.

3.2 Storages and the Folder Tree

Evolution displays the folder hierarchy as a folder tree on the left side of its windows. The top-level nodes of the folder tree are called *storages*, and they identify a specific source of information. The following are storages that are implemented by the current Evolution components:

- The local storage, which is where all the folders stored on the local file system are.
- The IMAP server storages. Each of the configured IMAP servers gets a storage node which contains all the folders stored on that specific IMAP server.
- The LDAP server storage. This storage contains all the configured LDAP servers.
- The vfolder storage. This contains all the configured virtual folders (virtual folders are discussed in section 4.5).

Storages are either built in the shell (for example, this is the case for the local storage), or are implemented by components.

The main purpose of a storage is to provide a tree of folders and provide the physical URIs for each of them. For example, the `INBOX` folder on IMAP server `imap.foo.bar.net` has to be displayed under the `imap.foo.bar.net` node in the folder tree, and needs to be mapped to the `imap://user@foo.bar.net/INBOX` URI.

3.3 The Local Storage

The shell implements the mapping for the local storage, which allows folders to be saved on a local file

system.

The local folders are all saved in the `evolution` directory in the user's home directory. Folders of different types all live there, and every folder can have one or more subfolders of arbitrary types.

In the case of the local storage, each folder is actually a directory whose name is the same as the name of the folder in the storage. Each directory contains a `folder-metadata.xml` file that contains information about the folder, such as its type and description. "Stock" folders (such as the *Inbox* or the *Drafts* folder) also store additional information there, such as the translations for the name of the folder in various languages. Each folder directory also contains a `subfolder` directory that contains all its subfolders.

Besides the `subfolders` directory and the `folder-metadata.xml` file, the Evolution components are responsible for the rest of the contents of the directory. When opening a folder, Evolution will invoke the `ShellComponent::createView` method on the component, with a `file:` URL to the folder's directory as the argument, and the component will return a Bonobo Control for that view.

4 Evolution Mail

4.1 The Camel Messaging Library

The foundation of the mail component is the Camel messaging library. Loosely based on the design of Sun's JavaMail API, it provides an abstraction layer through which mail messages are accessed.

4.2 Pluggable Architecture

Camel uses URIs such as `mbox://home/ettore/mbox` to locate message folders. The first part of the URI (`mbox` in this case) represents the module that is used to access the folder, and different modules can be implemented easily and installed independently of Evolution.

By using this mechanism, Camel currently supports IMAP, POP-3 and NNTP through the same API. SSL encription is also available for both the IMAP and POP-3 protocols. Moreover, Camel supports different mail storage mechanisms, including a standard UNIX `mbox` format which is the default, and a one-file-per-message MH-like back-end.

New protocols and back-ends can be added by subclassing some specific classes and installing some shared libraries in a specific location on the file system.

4.3 Summarizing and Indexing

In order to speed up access to the mail contained in the local folders, Camel creates a summary file which contains the contents for some of the headers (such as subject, author and recipients). The summary file is read when the folder is first opened, and gets updated as messages get added and removed. By using this method, Camel is able to read and manipulate mbox folders much faster than most other mail clients.

In addition to this, Camel provides a custom indexing library called `libibex`. When mail is imported into a folder, `libibex` is used to create an on-disk hash of all the words, together with pointers to the messages that contain them. When a search is performed on a folder, Evolution can thus just go through the hash and obtain the list of matching messages very quickly.

Evolution makes searching through folders very simple by providing a *quicksearch bar*. The quicksearch bar contains an option menu through which the user can select a criterion (such as "Message Body Contains" or "Sender Contains"), and an entry in which the user can specify a search string; when this functionality gets activated, only the matching messages are displayed in the message list. The user can also further manipulate the search criteria by using an advanced search dialog, and save the search so that it appears as a pre-defined search criterion in a menu on the quicksearch bar.

Camel also enables the application to associate metadata to each message; for example, it is possible to associate keywords, scores or colors to specific mail messages. This functionality allows the user to expand his range of possibilities for mail organization.

4.4 Filters

Camel also provides filtering of messages through arbitrary rules. The rules are expressed as Lisp-like s-expressions, and allow a broad range of possibilities.

Matches are specified by expressions like the following:

```
(header-contains "From" "john.smith")
```

This will match messages that contain the string `john.smith` in the `From` header. Basic matches such as `header-contains` can be combined by using boolean operators, such as `and`, `or` and `not`. For example:

```
(not (or (header-contains
          "To" "ettore@ximian.com")
         (header-contains
          "Cc" "ettore@ximian.com"))
     (header-contains
      "From" "john.smith"))
```

Actions are also specified as s-expressions. For example:

- Copy to a folder:

  ```
  (copy-to "mbox://home/jsmith/somefolder")
  ```

- Move to folder:

  ```
  (move-to "mbox://home/jsmith/somefolder")
  ```

- Delete:

  ```
  (delete)
  ```

- Set the colour:

  ```
  (set-colour "rgb:ff/80/80")
  ```

- Set the flags:

  ```
  (set-system-flag "Answered")
  ```

Actions can also be combined by using `begin`, and other rules can be prevented from being applied to the same message by using `stop`:

```
(begin
 (copy-to "mbox://home/jsmith/Inbox")
 (set-colour "red")
 (move-to "mbox://home/jsmith/Important")
 (stop))
```

These s-expressions are normally hidden to the user; Evolution comes with a user-friendly GUI filter editor that allows editing of the rules in a simple way.

The same syntax is also used for performing searches through the folders.

4.5 Virtual Folders

In addition to "traditional" filtering, Camel also supports *virtual folders*, by using the indexing capabilities provided by `libibex`. A virtual folder (or *vfolder*, for short) is a folder that doesn't physically exist on a storage media; rather, its content is dynamically generated through a search query on one or more physical folders.

Camel is able to make virtual folders work just like regular folders, allowing copying and moving of messages, changing of their attributes and so on. Changes to the messages actually affect the physical folders the messages are contained in; and since resolving the query is very efficient due to the indexing, using vfolders is not much slower than using regular on-disk folders.

Users can use Virtual folders to organize their mail in several different ways; for example, they can track messages from/to a specific person, search for messages containing certain words, look for messages with a certain score or associated to some specific keyword.

Evolution encourages the user to create virtual folders by providing GUI hooks for the most common operations; for example, it is possible to create a virtual folder for messages sent by a certain author by just right-clicking on a message by that author and selecting the "Create Vfolder on Sender" command.

4.6 Mailing List Recognition

Evolution comes with a simple, yet effective system to filter and create vfolders on mailing lists.

Evolution is able to recognize a certain number of typical hidden mailing list headers that identify the mailing list; when a message with such headers gets imported, a mailing list ID gets extracted from them and is put in the message summary, associated with that message.

Examples of mailing list headers that Evolution would recognize are:

```
Sender: owner-mailing-list@site.net
X-Mailing-List: mailing-list
List-Id: mailing list
```

By associating this mailing list ID automatically, Evolution is able to both filter mailing list messages and create vfolders of them, without requiring the user to actually specify the complex header matches manually as it normally happens with tools like `procmail`.

4.7 Encryption

Camel also includes support for S/MIME and PGP based encryption, which are thus natively supported by the mail component.

S/MIME support is based on the `libnss` library originally developed for the Mozilla[4] project.

4.8 Bonobo Integration

By using the Bonobo component technology, the mailer component in Evolution is also able to display various kinds of email contents in-line. For example, if you receive an attachment that is a Gnumeric spreadsheet, Evolution is able to activate and embed an instance of Gnumeric to display the spreadsheet in-line, as if it was integral part of the mail message.

This mechanism works by using the built-in MIME type handing mechanism of GNOME VFS[5] to detect what component(s) can be used to display the contents in-line.

Due to the extensible nature of Bonobo and GNOME VFS, it's very easy to add several kinds of viewers for different types of documents; for example, once the Bonobo integration of OpenOffice[6] is complete, it will be possible to display contents of Word, PowerPoint or Excel documents by using OpenOffice's ability to handle those proprietary formats.

The calendar and addressbook components in Evolution implement some viewer components themselves. For example, the calendar provides a viewer for `iTIP` attachments that are used for sending meeting requests. When a user receives an `iTIP` attachment in a mail message, Evolution activates the corresponding calendar control that displays information about the meeting request, and allows the user to either reject the meeting (by sending a denial message back to the author of the request), or accept and add information about the meeting to her own calendar automatically.

Likewise, the addressbook component implements a viewer for the `vCard` format that is used to describe contact information. The `vCard` control displays the information and also allows the user to add the contact to his contact database by clicking on a button in the control itself.

4.9 The Mail Display and Composition Engine

Evolution uses a library called `GtkHTML` to display and compose mail messages. `GtkHTML` is a simple, lightweight HTML engine which also comes with a powerful editor featuring undo/redo support, configurable keybindings and built-in spell checking (so that it can highlight misspelled words automatically as the user types).

By default, HTML is disabled in the message composer. When HTML is disabled, the rich text attributes don't get displayed (so all text is displayed in the same monospaced font), but automatic formatting of the inserted text still happens automatically. This way, it is possible to align paragraphs, make itemized lists and indent the text, while still sending plaintext messages.

The composer is also able to switch back and forth between HTML and non-HTML mode and than back to HTML again without losing the text attributes that the user has set.

[4] `http://www.mozilla.org`

[5] `http://www.ximian.com/tech/gnome-vfs.php3`

[6] `http://www.openoffice.org`

5 The Wombat

The storage back-end for the Evolution's addressbook and calendar components is provided by a separate component called *the Wombat*. The addressbook and calendar components are simply implemented as views for the data stored into the Wombat.

The Wombat is an out-of-process component; so the Wombat works as a little stand-alone daemon which the addressbook and the calendar connect to.

This data/view split is achieved by registering listeners on the Wombat; a listener is an interface that a Wombat client implements on his side to get notifications about changes in the data. When some data in the Wombat changes, the Wombat goes through all the registered listeners and reports the details of the change to them so that the views can update themselves.

5.1 Advantages of the Wombat

There are several advantages with this solution, compared to a more simple approach based on a monolythic component managing both the data and the view:

- The Wombat is accessible by different applications, and can be accessed without running Evolution.
- Since CORBA bindings are available for various languages including scripting languages, it is very easy to query or make changes on the addressbook's or the calendar's contents by using a scripting language (for example, Perl).
- Access to the data is arbitrated by the Wombat, so there is no need to lock the files or care about conflicting changes. Likewise, since all the changes go through the Wombat, the data is always kept in a consistent state, and buggy 3d party applications cannot corrupt the physical files.
- Because of the data/view split and the fact that the Wombat acts as an independent data repository, it is possible to easily transform the Wombat into a remote server.
- All the folders of the same type can be accessed through a common Wombat interface. For example, it is possible to access LDAP and local folders by using the same addressbook interfaces (see 6).

5.2 The Wombat's Structure

The Wombat is constituted by two subparts:

- The PAS, or Personal Addressbook Server, for handling contact folders.
- The PCS, or Personal Calendar Server, for handling task and calendar folders.

Each of these modules can provide different kinds of backends through a simple pluggable architecture based on dynamically linked modules.

5.3 Palm Synchronization

Evolution also comes with the ability to synchronize the addressbook and calendar folders to a PDA device running the PalmOS operating system.

This feature is based on GNOME's PalmOS synchronization technology, called `gnome-pilot`, which is a generic framework for providing PDA synchronization conduits. `gnome-pilot` provides a small daemon, called `gpilotd`, that watches the cradle device connected to either the serial or the USB port; when the hotsync button on the cradle gets activated, `gpilotd` activates all the conduits, one by one, so that they can perform the synchronization of the data.

Evolution provides addressbook and calendar conduits that simply talk to the Wombat, reading and synchronizing the data through its CORBA interfaces. If Evolution is running when synchronization happens and the user has calendar and/or addressbook views open, they will get updated automatically through the listener mechanism.

6 Evolution Addressbook

The addressbook backend (PAS, see section 5.2) in the Wombat includes support for both LDAP servers and local contact databases.

In the local case, it uses Sleepycat's Berkeley DB library[7]; the information is stored on the database as a set of `vCard`s, indexed by the name. Since the widespread `vCard` format is being used, it's really easy to migrate contact information from/to other applications.

For accessing LDAP servers, it uses the OpenLDAP[8] implementation of the LDAP protocol.

The data of the addressbook can be displayed both in a traditional list view, and in a multicard view similiar to that used in Microsoft Outlook; both views can be printed, using the GNOME printing architecture.

The addressbook also implements address picker controls that allow a user to select one or more addresses from the contacts folders (both local and LDAP), either through a simple entry widget with automatic completion, or through a simple point-and-click picker dialog similiar to that found in other mail applications. These controls are used both in the message composer (to specify the recipients of a message) and in the calendar (to pick attendee addresses in the meeting request dialog).

7 Evolution Calendar

The Evolution calendar allows the user to manage his to-do lists and appointments. Its implementation is based on the `libical` library and the data is saved in the standard `iCalendar` format, which makes it easily interoperable with other similiar applications.

It supports different styles of views and also supports printing of the calendar with a nice layout. The to-do list supports setting priorities, deadlines and categories and is highly configurable.

The calendar also comes with a quicksearch bar that the user can use to look for events and to-do items.

Currently it only supports local calendars, but at some point support for a shared calendar server solution will be added.

Evolution is also provided with a per-user alarm daemon for getting notification of events even when Evolution isn't running. The alarm daemon just connects to the Wombat to get the list of pending events, and then wakes up as soon as the specified time is reached.

8 Importing Tools

Evolution provides a generic Bonobo-based framework for creating importing tools, to facilitate migration from other applications. When the user first runs Evolution, the installed import plugins are activated and check for the canonical locations where data from other applications is stored (for example, `/home/user/mail` for Pine, or `/home/user/nsmail` for Netscape Communicator).

Evolution currently supports automatic importing from the following applications:

- Elm
- Netscape Communicator
- Pine
- GnomeCard
- Generic UNIX `mbox` (which, for example, can be used with Mutt)

Other importers (such as one for Gnus[9], the Emacs mail and news reading application) are being worked on.

9 Conclusion

Evolution 1.0 is scheduled to be released some time in Autumn 2001. Binary snapshots are available

[7] `http://www.sleepycat.com`
[8] `http://www.openldap.org`
[9] `http://www.gnus.org`

through Ximian Red Carpet, the package management application of the the Ximian GNOME desktop. The Ximian GNOME desktop can be downloaded from `http://www.ximian.com/desktop/`.

Alternatively, the Evolution source code can be downloaded from the regular GNOME FTP site (`ftp://ftp.gnome.org/pub/GNOME/unstable/source`) or from the GNOME CVS server; instructions for using the GNOME CVS server are available on the GNOME Developer's web site at `http://developer.gnome.org`.

More information about Evolution is available on the Ximian Evolution home page at `http://www.ximian.com/apps/evolution.php3`.