P-1228

KUMARAGURU COLLEGE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DEBUGGING EVOLUTION

PROJECT WORK DONE AT
NOVELL SOFTWARE DEVELOPMENT (I) Pvt. Ltd.
BANGALORE.

PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF M.Sc SOFTWARE ENGINEERING (APPLIED SCIENCE) OF BHARATHIAR UNIVERSITY, COIMBATORE.

SUBMITTED BY
FAZLUL HUQ.R
REG.NO:0137S0031

GUIDED BY
Miss.P.PARAMESWARI, M.C.A., M.Phill., Lecturer,
Department of Computer science and Engineering

KUMARAGURU COLLEGE OF TECHNOLOGY

(Affiliated to Bharathiar University)

Department of Computer science and Engineering

Coimbatore – 641 006

CERTIFICATE

This is to certify that the project work entitled

DEBUGGING EVOLUTION

Done By

FAZLUL HUQ.R

Reg. No. 0137S0031

Submitted in partial fulfillment of the requirements for the award of the degree M.Sc Applied Science Software Engineering of Bharathiar University.

[illegible]
Professor and Head

[illegible]
Internal Guide

Submitted for the University examination held on 29-09-2004

[illegible]
Internal Examiner

[illegible]
External Examiner

DECLARATION

I here by declare that the project work entitled

"DEBUGGING EVOLUTION"

done at

NOVELL SOFTWARE DEVELOPMENT LIMITED, BANGALORE.

submitted to

Kumaraguru College of Technology

(Affiliated to Bharathiyar University)

in partial fulfillment of the requirements of the award of the Degree of M.Sc (APPLIED SCIENCE – Software Engineering) is a report of work done by me during the period of study in Kumaraguru College of Technology, Coimbatore-641006.

Under the supervision of

Miss.P.Parameswari.M.C.A.,M.Phil.

Name of Candidate	Register Number
Fazlul Huq.R	**0137S0031**

Signature of Candidate: [illegible]

Date : 29|09|04.

Place : Coimbatore.

ACKNOWLEDGEMENT

I deem it as great pleasure to place my deep sense of gratitude and indebtedness to **Dr.K.K.Padmanaban, B.Sc(Engg.), M.Tech., Ph.D., Principal, Kumaraguru College of Technology** for giving me the opportunity to undertake this project

I am grateful to, **Dr.S.Thangasamy, Ph.D., Professor and Head of the CSE Department**, Kumaraguru College of Technology, giving me this golden opportunity to carry out my project work successfully.

My sincere thanks are offered to **Mr. K.R.Baskaran Project coordinator and Miss.P.Parameswari, lecturer of CSE Department** for the encouragement and support bestowed on me as my project guide .I am very much indebted to her for the suggestions and guidance extended in successfully completing the project.

I thank all my faculties whose diligent efforts have led me to complete the project successfully.

I owe my gratitude to **Mr. Rajiv Ranjan, Team leader**, and my external mentors **Mr. Chenthill Palaniswamy, Mr. Siva** of Novell Software Development (I) Pvt. Ltd., for rendering permission to carry out my project work in this esteemed concern and the help rendered by them.

I wish to express my sincere thanks to people who have contributed a lot towards the successful completion of this project work.

Novell Software Development (I) Pvt. Ltd.
Regd. Office : 49/1 & 49/3, Garvebbhavi Palya,
7th Mile, Hosur Road,
Bangalore - 560 068. INDIA
Ph : +91-80-2573-1856 Fx : +91-80-2573-1870

Novell

CERTIFICATE

This is to certify that R. Fazlul Huq student of M.Sc (software Engineering) of Kumaraguru College of Technology, Coimbatore has been working on the "`Evloution Develoment Project`" since 15th May 2004 under the guidance of Novell .

He has been regularly interacting with the mentor.

The project is partially completed and will end on 4th of November 2004. His performance has been found to be satisfactory till date.

Date: 23rd of September 2004

[illegible]

Parag Goel
Manager – Software Engineering
Novell Software Development (I) Private Ltd

SYNOPSIS

The project entitled “**Debugging Evolution**” is a testing project. It is done in **LINUX**, where **GTK+** programming is used.

“**Debugging Evolution**” is the part of the **GNOME** project. This involves ‘**Fixing of Bugs**’ in Evolution.

Evolution, from Novell, is the world's most popular and powerful collaboration solution for Linux and UNIX systems. It seamlessly integrates e-mail, calendaring, scheduling, contact management, and task lists

Each Evolution component has to implement a canonical Evolution::Component **CORBA** interface; when the shell starts up it queries **Bonobo** Activation asking for a list of all the Bonobo components that support that interfaces, activates them, and embeds them.

Evolution is a project to provide integrated mail, addressbook and calendaring functionality to the **GNOME desktop**

Evolution is the most popular integrated mail, addressbook and calendaring application in Linux. Apart from being one of the most successful open source projects, Evolution also provides connectors for groupware applications like Microsoft Exchange, Groupwise and SuSe OpenExchange.

Working in Evolution would provide you a strong foundation in Gnome technologies as well as Internet standards for Calendaring and Mailing like POP, IMAP, VCards and ICAL. Current work in Evolution includes providing APIs for Desktop developers to integrate calendaring and groupware services with their applications.

Ximian Evolution is a main tool to implement evolution. Ximian Evolution is a tool to help you get your work done. It can help you work in a group by handling email, address and other contact information, and one or more calendars. It can do that on one or several computers, connected directly or over a network, for one person or for large groups.

Ximian Connector 1.4 has added the **Out of Office** feature and an automatic configuration tool for large Exchange deployments. Previous versions of Ximian Connector for Microsoft Exchange added support for the **Flag for Followup** feature, public folders, delegation, direct booking, and more.

With Ximian Evolution, you can accomplish your most common daily tasks faster. For example, it takes only one or two clicks to enter an appointment or an address card sent to you by email, or to send email to a contact or appointment. Ximian Evolution makes displays faster and more efficient, so searches are faster and memory usage is lower. People who get lots of mail will appreciate advanced features like VFOLDERS, which let you save searches as though they were ordinary mail folders.

The scope of the evolution project is to achieve the following features:

- **Better Fonts**

  Ximian Evolution 1.4 uses the same font smoothing technology as the rest of your GNOME 2 desktop.

- **Cleaner Shutdowns**

  We have fixed several bugs that caused Evolution to continue to occupy system resources when it was not running

- **Mail Composer HTML Improvements**

  The message composer has continued to improve

- **New Graphics and Icons**

Evolution involves programming in the GObject System and uses Gnome technologies like Bonobo, gtk etc. Working in Evolution would provide you a strong foundation in Gnome technologies as well as Internet standards for Calendaring and Mailing like POP, IMAP, VCards and ICAL. Current work in Evolution includes providing APIs for Desktop developers to integrate calendaring and groupware services with their applications.

Evolution is to make the tasks of storing, organizing, and retrieving your personal information easier, so you can work and communicate with others. That is, it's a highly evolved groupware program, an integral part of the Internet-connected desktop.

This project aims at creating a better and a more user-friendly version of Evolution-1.5.

CONTENTS

1. INTRODUCTION

1.1 Project Overview

This project is being done to make the tasks of storing, organizing, and retrieving your personal information easier, so you can work and communicate with others.

Computer programs or operating systems for which the source code is publicly available are referred to as open-source software. Inherent in the open source philosophy is the freedom of a distributed community of programmers to modify and improve the code. Thus outside programmers can improve, or use it themselves. This includes fixing bugs, improving performance, and adding features. The most widely known example of open-source software is the Linux operating system

The Evolution is done as an Open Source Project and it become the first to offer comprehensive Linux solutions for the enterprise and give organizations a secure, reliable and mature Linux foundation.

Ximian Evolution is the most widely used Linux personal and workgroup information management software, seamlessly integrating email, collaboration, calendaring, meeting scheduling, contact management, and task lists using open standards. Now included in virtually every Linux distribution, Evolution is the standards graphical email client and collaboration front end for Linux.

Currently, evolution supports Posted tasks and the above feature extends it to support scheduled Group Tasks. The enhancements would involve GUI support for the feature as well as backend support in evolution-data-server. The feature shall be developed for the GroupWise backend though the GUI needs to be generic enough for other back ends that may want to implement this feature.

Evolution involves programming in the GObject System and uses Gnome technologies like Bonobo, gtk etc. Working in Evolution would provide you a strong foundation in Gnome technologies as well as Internet standards for Calendaring and Mailing like POP, IMAP, VCards and ICAL. Current work in Evolution includes providing APIs for Desktop developers to integrate calendaring and groupware services with their applications.

The steps that were followed while starting the project:

- System study of the requirements.
- Reference of the studies made.
- Building the source from the Open source CD.
- Updating the sources through the web.
- Creating a new path for Evolution.
- Running evolution from that path.
- Testing of the design through the operation.
- Making the necessary changes.
- Obtaining the final design.
- Coding the project.
- Code compilation.
- Testing the system.
- Implementing the system.
- Documenting the project.

LIST OF ABBERIVATIONS

- **GNOME** - GNU Network Object Model Environment
- **KDE** – K Desktop Environment
- **GCC** – GNU C++ Compiler
- **CVS** – Concurrent Version System
- **CORBA** – Common Object Request Brober Architecture
- **IIOP** – Internet Inter-ORB Protocol
- **TCP/IP** – Transport Control Protocol/Internet Protocol
- **OMG** – Object Management Group
- **ORB** – Object Request Broker
- **QA** – Quality Analysis
- **PDF** – Portable Document Format
- **HTTP** – Hyper Text Transfer Protocol
- **IDL** – Interface Definition Language
- **MICO** -MICO Is CORBA
- **GIMP** – GNU Image Manipulation Program
- **RCS** – Revision Control System
- **SCCS** – Source Code Control System
- **SSH** – Secure Shell
- **URI** – Uniform Resource Identifier
- **POSIX** – Portable Operating System Interface
- **GGU** – GNOME Ghost Script
- **GTK** -GIMP Tool Kit

LIST OF DEFINITIONS

- **The GNU Debugger (GDB):** The purpose of a debugger such as GDB is to allow you to see what is going on inside another program while it executes or what another program was doing at the moment it crashed.

- **Grep:** Searches the named input files (or standard input if no files are named, or the file name - is given) for lines containing a match to the given PATTERN. By default, grep prints the matching lines.

- **Find:** Searches the directory tree rooted at each given file name by evaluating the given expression from left to right, according to the rules of precedence, until the outcome is known (the left hand side is false for and operations, true for or), at which point find moves on to the next file name.

- **Process status (ps):** Gives a snapshot of the current processes. The use of top displays a repetitive update of this status.

- **Link (ln):** Create a link to the specified TARGET with optional LINK_NAME. If LINK_NAME is omitted, a link with the same base name as the TARGET is created in the current directory. When using the second form with more than one TARGET, the last argument must be a directory; create links in DIRECTORY to each TARGET. Create hard links by default, symbolic links with --symbolic. When creating hard links, each TARGET must exist.

- **Vim:** is a text editor that is upwards compatible to Vi. It can be used to edit all kinds of plain text. It is especially useful for editing programs.

Novell.

Novell Software Development (I) Pvt. Ltd., Bangalore is a subsidiary of Novell Inc, USA. Novell Bangalore has contributed significantly in making Novell's one Net Vision a reality by taking complete engineering responsibility for many critical components and products including eDirectory, ZENworks, DNS/DHCP, TCP/IP, BorderManager IPv6, SMS, NFS, Groupwise Gateways and DirXML. While developing these Net Services software products on Linux and Netware, Novell Bangalore has acquired depth in skill, competency and 'know-how' in handling high-end core technology. This competency has been successfully built and constantly enriched by Novell Bangalore.

We at Novell Bangalore understand that it is best to have a local community in India, focusing on Linux desktop. Novell is setting up a 40 member team in Bangalore, working exclusively as part of the Linux desktop community. This community would participate in Linux desktop development called Ximian Desktop, GNOME, Open Office and Mono, making it the most suitable desktop for global and local needs. If you think that you can contribute in any manner, you are most welcome to join this community. An initiative of this nature requires contributions from various areas including programming, testing, documentation, artwork, translation, domain enterprise.

Awards Won by

Novell.

4 out of 5 rating from PC Magazine

PC Magazine gives Ximian® Desktop overall 4 stars, and says "Evolution was intuitive, stable, and easy to set up. It does indeed provide an elegant alternative to Microsoft Outlook."

Linux Journal Editor's Choice Award for best Communications Tool

"The unthinkable is happening," the Linux Journal editors wrote. "Linux gurus are dropping text-based mailers for a GUI mailer called Evolution. Besides mail, Evolution also offers a calendar, address book with LDAP integration, and to-do list."

Golden Tuxie Award for best Graphical Email Client

"Evolution is definitely our top banana," said the editors of Linux Magazine, describing "a powerful end-user GUI environment that makes the other 800 pound gorilla from Redmond look like a monkey's uncle."

Evolution Offers Outlook Experience

"We were impressed with the interface and improvements in Evolution [1.2] since we reviewed the program in its initial release last year. When teamed with Version 1.2 of Ximian's Connector for Microsoft Corp.'s Exchange 2000, Evolution does a good job of standing in for Outlook as an Exchange client on machines that are running Linux or Solaris."

2. SYSTEM STUDY AND ANALYSIS

2.1 Software Requirement Specification

Novell Evolution is supported on the following distributions of software's:

- SUSE LINUX 9.0
- SUSE LINUX 8.2
- SUSE LINUX Desktop 1.0
- Red Hat Linux 9.0
- Red Hat Linux 8.0 (see note below)
- Red Hat Linux 7.3 (see note below)
- Linux Mandrake 9.1
- Sun Solaris 8 (SPARC)(requires Sun GNOME 2)

Notes:

- All distributions are for 32-bit Intel architecture processors and clones, i.e. ix86, AMD, unless specified.
- If your distribution is not listed, it is unsupported at this time. We do not support beta, testing, or unstable versions of any distribution. For example, Red Hat Rawhide, Red Hat 9.0.93 (Severn), Debian Sid, and Mandrake Cooker are not supported.
- Novell Evolution 1.4 and Evolution Connector 1.4 are supported on Red Hat 7.3 and Red Hat 8.0 only when

➢ <u>BONOBO</u>

BONOBO is the component model used in the GNOME project. It is largely inspired from Microsoft's Object Linking and Embedding 2 (*OLE2*). However, Bonobo relies on *CORBA* for its communication layer whereas OLE2 relies on COM/DCOM.

Large GNOME applications like the spreadsheet *Gnumeric*, the file manager *Nautilus* or the mail client and calendar Evolution make heavy use of Bonobo.
Other GNOME applications like the postscript viewer, the *Portable Document Format* (PDF) viewer or the SVG viewer also use Bonobo but in a smaller degree. Obviously, components built for these applications may well be used in future applications.

Bonobo is the framework that allows GNOME developers to create:

- **Components** which are reusable pieces of software. Components help reducing the complexity of applications by reducing the amount of information a programmer needs to know about the system. They also allow the programmer to tackle a single problem in order to provide a specific functionality. More importantly, they are easy to reuse and debug. Furthermore, components present well defined interfaces and interactions: The magic behind component programming is in the "interfaces" that a component exports to the world. Each one of these interfaces is a "socket" into which other components and applications can plug into. In Bonobo, interfaces are described in IDL and interactions rely on CORBA. CORBA can be considered as the glue between the cooperating components.

- **Embeddable documents** which are documents that can be inserted or embedded into other documents.

- **Controls** which are graphical components. They are embedded in applications called containers.

- **Scripts** which allow manipulating an application through its Bonobo components. This is possible since components export their CORBA interfaces. An example of scripting would be to manipulate a Gnumeric spreadsheet. Technically speaking, Bonobc is a set of CORBA interfaces and their default implementation in C.

➢ <u>OBJECT REQUEST BROKERS</u>

An Object Request Broker (ORB) is an implementation of the CORBA specifications. Many ORBs are available. They differ in many ways. Some of them are free, others are not. Figure 2.1 shows that different ORBs spread over the network are able to interoperate via IIOP. An ORB encapsulates the inherently heterogeneous aspects of the networks and operating systems.

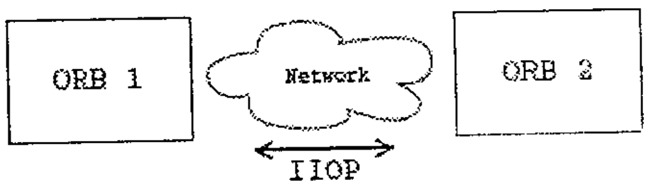


Figure 2.1: ORB interoperability

Thus, an ORB is a middleware because Middleware hides the underlying details from the application. It does this by providing a common set of services across all the platforms on which it lives.

An ORB can also be considered as a layer above the network and the operating system. Technically speaking, an ORB allows a client to transparently invoke an operation on a server object (Figure 2.2). The object can be either on the same machine or across the network. Furthermore, the client need not to know the details of the object, namely its programming language, its operating system, or any other aspect that is not reflected in its interface. Briefly, an ORB makes the communication between a client and a server easier.

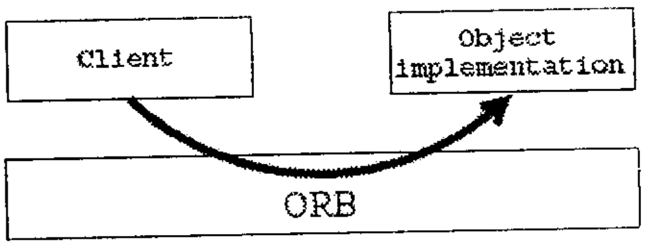


Figure 2.2: An ORB communicating a request to a target object implementation

An ORB should accept requests, transport them to the appropriate object and possibly return results. In others words, it is responsible for the transport, marshaling and unmarshaling of messages. Marshaling consists of encoding messages and putting them on the wire, while unmarshaling consists of decoding them from the wire and placing them in an address space.

As mentioned earlier, there are many differences between ORBs. Among these differences are the performance in terms of speed and memory usage, the language bindings provided and the CORBA services available.

Here is a list of some known ORBs:

- Free ORBs: Red Hat's ORBit, Sun's Java IDL, Xerox's ILU and MICO.
- Vendor ORBs: IONA's Orbix, IONA's OrbixWeb2, OOC's ORBacus3, and Inprise's VisiBroker4.

2.2 Existing System

Evolution, from Novell, is the world's most popular personal and workgroup information management solution for Linux and UNIX systems. The software seamlessly integrates email, calendaring, contact management, and task lists, in one easy-to-use application. Evolution is powerful collaboration software that connects to popular corporate communications architectures like Microsoft Exchange, Novell GroupWise, and other messaging systems. Novell Evolution supports a broad range of leading Linux distributions.

With the additional installation of Evolution Connector for Microsoft Exchange, Novell Evolution functions as Microsoft Exchange 2000 or 2003 client, with access to scheduling, mail, public folders and global address book features.

Evolution 1.2, which will be available during Q3 2004, will deliver important new features including:

- integrated connectivity to Novell GroupWise
- integrated connectivity to Microsoft Exchange
- improved offline support for IMAP accounts
- numerous calendar improvements,
- support for S/MIME, enhanced contact management
- Gaim instant messaging integration
- Improved desktop integration.

2.3 Proposed System

Evolution 1.5

Novell® EvolutionTM 1.5.94 is the e-mail client included with the Novell Linux Desktop 1.0. Evolution can retrieve e-mail messages from various e-mail systems.

This release of Evolution includes the following new features:

- ♦ Support for NNTP, S/MIME, GroupWise®, and Exchange servers SP2 or later (in addition to the IMAP, LDAP, POP, and SMTP servers supported in previous versions)
- ♦ Calendars now display as a single view with different overlays
- ♦ Web calendars can be used as well as local calendars
- ♦ Spam filtering

2.3.1 What's New in Connector 1.5Connector Features?

Evolution Connector supports the following basic Microsoft Exchange features:

➢ **Mail**

♦ **Viewing Mail in Exchange Folders**

Mail stored on the Exchange server is visible in the Mail and Exchange tools in Evolution.

♦ **Sending E-Mail via Exchange Protocols**

You might use the Microsoft Exchange mail transport protocol to send e-mail. Make sure that the address you have entered as your e-mail address is exactly the one that the Exchange server has on file. This may be "yourname@exchange-server.your_domain.com" rather than "yourname@your_domain.com".

- **Calendar**

  - **Meeting Request/Proposal**

    Allows Evolution users to schedule meetings and view attendee availability for other users (Evolution or Outlook users) on Exchange.

  - **Adding iCalendar Meeting Requests to the Calendar**

    If you receive an iCalendar meeting request and add it to your calendar, it is saved to your Exchange calendar.

- **Contacts**

  - **Address Completion**

    Supported for your Exchange Contacts folder. Not yet supported for the Global Address List.

  - **Adding vCards to the Address Book**

    If you receive a VCard attachment and click Save in Address Book, it is saved to your Exchange address book.

    New Address Book entries can be created on Exchange from received e-mail messages with a single click

3. PROGRAMMING ENVIRONMENT

3.1 Hardware Configuration

The following is a rough guideline of some hardware requirements for Linux. You do not have to follow them directly, but this list should give you a rough idea of what's required:

- An Intel 80386 or better CPU (the faster and more powerful the better, of course). You don't need a math coprocessor, although it's strongly recommended as it speeds up a lot of graphics operations, especially under X. If you have an 80386 chip, 80387 math coprocessors are available separately and are installed in a socket on your motherboard. If you have a 80486 processor, the math coprocessor is on the 486 chip itself. (The exception is the 80486SX, which is a 486 chip without the coprocessor components.) Pentium and Pentium Pro CPUs have the coprocessor built in.

- If you don't have a math coprocessor, the Linux kernel will emulate floating-point math for you. If you do have one, however, floating-point math will be handled by the hardware, which for some applications is a real plus.

- Your system must be either an ISA, EISA, PCI, or local bus architecture machine. These terms specify how the CPU communicates with hardware, and are a characteristic of your motherboard. Most existing systems use the ISA bus architecture.

- At least 4MB of RAM.

- Memory is speed, so if you have more RAM you'll thank yourself for it later. If you're a power user, 8MB should be more than enough for most applications. If you want to run X Window, your system will require at least 8MB of RAM.

- A hard drive with space available for installing Linux. The amount of space required depends on the amount of software you're installing and how much free space you wish to leave yourself. You can install Linux in very small amounts of disk space, but a realistic minimum is about 150MB. For a full system with X and development tools, much more is required. The complete installation can use up 250MB, with more useful for data files.

- A Hercules, CGA, EGA, VGA, or Super VGA video card and monitor. In general, if your video card and monitor work under MS-DOS or Microsoft Windows, then Linux should be able to use them without any problem. However, if you're going to use the X Window system (either Metro-X or Xfree86), some video configurations are not supported.

3.2 Description of Software's and Tools used

Linux is a popular operating system which finds its major utility in the field of networking. **GNOME (GNU network object model environment)** is an operating environment used in Linux. This chapter gives a brief introduction on Linux and GNOME environment. It also explains in detail about the significance of this project.

3.2.1 History of LINUX

Linux is an operating system that was initially created as a hobby by a young student, *Linus Torvalds*, at the University of Helsinki in Finland. Linus had an interest in Minix, a small UNIX system, and decided to develop a system that exceeded the Minix standards. He began his work in 1991 and released version 0.02 and worked steadily until 1994, when version 1.0 of the Linux Kernel was released. The kernel, at the heart of all Linux systems, is developed and released under the GNU General Public License and its source code is freely available to everyone. It is this kernel that forms the base around which a Linux operating system is developed. There are now literally hundreds of companies and organizations and an equal number of individuals who have released their own versions of operating systems based on the Linux kernel.

Linux has completed a decade of development. Today, Linux is one of the fastest growing operating systems in the history. From a few dedicated fanatics in 1991-92 to millions of general users at present, it is certainly a remarkable journey. The big businesses have 'discovered' Linux, and have poured millions of dollars into the development effort, denouncing the anti-business myth of the open-source movement. IBM Corporation once considered the archenemy of open-source hacker community, has come forward with a huge fund for development of open source Linux based solutions. But what's really amazing is the continuously increasing band of developers spread throughout the world who work with a fervent zeal to improve upon the features of Linux. The development effort is not, as many closed-sourced advocates accuse, totally engulfed with chaos.

A well-designed development model supervised by some *maintainers* is adopted. Along with this, there are thousands of developers working to port various applications to Linux.

Networking support in Linux is advanced and superior to most other Operating Systems. Since the people developing Linux collaborated and used the Internet for their development efforts, networking support came early in Linux development. As an Internet server, Linux is a very good choice, often outperforming Windows NT, Novell and most UNIX systems on the same hardware. Linux is frequently chosen by leading businesses for superior server and network performance.

Through the efforts of developers of desktop management systems such as *KDE* and GNOME, office suite project *OpenOffice.org* and the *Mozilla* web browser project, to name only a few, there are now a wide range of applications that run on Linux and it can be used by anyone regardless of his/her knowledge of computers.

3.1.2 The GTK+ Tool

GTK+ is a multi-platform toolkit for creating graphical user interfaces. Offering a complete set of widgets, GTK+ is suitable for projects ranging from small one-off projects to complete application suites.

GTK+ is free software and part of the GNU Project. However, the licensing terms for GTK+, the GNU LGPL, allow it to be used by all developers, including those developing proprietary software, without any license fees or royalties.

GTK+ is based on three libraries developed by the GTK+ team:

- ♦ **GLib** is the low-level core library that forms the basis of GTK+ and GNOME. It provides data structure handling for C, portability wrappers, and interfaces for such runtime functionality as an event loop, threads, dynamic loading, and an object system.
- ♦ **Pango** is a library for layout and rendering of text, with an emphasis on internationalization. It forms the core of text and font handling for GTK+-2.0.
- ♦ The **ATK** library provides a set of interfaces for accessibility. By supporting the ATK interfaces, an application or toolkit can be used with such tools as screen readers, magnifiers, and alternative input devices.

GTK+ has been designed from the ground up to support a range of languages, not only C/C++. Using GTK+ from languages such as Perl and Python (especially in combination with the Glade GUI builder) provides an effective method of rapid application development.

GTK is essentially an object oriented application programmer's interface (API). Although written completely in C, it is implemented using the idea of classes and callback functions (pointers to functions).

There is also a third component called GLib which contains a few replacements for some standard calls, as well as some additional functions for handling linked lists, etc. The replacement functions are used to increase GTK's portability, as some of the functions implemented here are not available or are nonstandard on other Unixes such as g_strerror(). Some also contain enhancements to the libc versions, such as g_malloc() that has enhanced debugging utilities.

In version 2.0, GLib has picked up the type system which forms the foundation for GTK's class hierarchy, the signal system which is used throughout GTK, a thread API which abstracts the different native thread APIs of the various platforms and a facility for loading modules.

As the last component, GTK uses the Pango library for internationalized text output. This tutorial describes the C interface to GTK. There are GTK bindings for many other languages including C++, Guile, Perl, Python, TOM, Ada95, Objective C, Free Pascal, Eiffel, Java and C#. If you intend to use another language's bindings to GTK, look at that binding's documentation first. In some cases that documentation may describe some important conventions (which you should know first) and then refer you back to this tutorial. There is also some cross-platform APIs (such as wxWindows and V) which use GTK as one of their target platforms; again, consult their documentation first.

If you're developing your GTK application in C++, a few extra notes are in order. There's a C++ binding to GTK called GTK--, which provides a more C++-like interface to GTK; you should probably look into this instead. If you don't like that approach for whatever reason, there are two alternatives for using GTK. First, you can use only the C subset of C++ when interfacing with GTK and then use the C interface as described in this tutorial. Second, you can use GTK and C++ together by declaring all callbacks as static functions in C++ classes, and again calling.

GTK using its C interface. If you choose this last approach, you can include as the callback's data value a pointer to the object to be manipulated (the so-called "this" value). Selecting between these options is simply a matter of preference, since in all three approaches you get C++ and GTK. None of these approaches requires the use of a specialized preprocessor, so no matter what you choose you can use standard C++ with GTK.

This tutorial is an attempt to document as much as possible of GTK, but it is by no means complete. This tutorial assumes a good understanding of C, and how to create C programs. It would be a great benefit for the reader to have previous X programming experience, but it shouldn't be necessary. If you are learning GTK as your first widget set, please comment on how you found this tutorial, and what you had trouble with. There are also C++, Objective C, ADA, Guile and other language bindings available, but I don't follow these.

This document is a "work in progress". Please look for updates on http://www.gtk.org/. I would very much like to hear of any problems you have learning GTK from this document, and would appreciate input as to how it may be improved.

4. SYSTEM DESIGN

4.1 Input design

Bug #	Product - Version	Component	Status	Short Summary
48126	Evolution – 1.5.90	Tasks	NEW	Various options not available from menu bar

Description of Problem

All menu options should be available from the main menu bar. Some items, such as Assign Task and Forward as iCalendar, are available from the menu bar after you open a task. Others, such as Open task, are available from a right-click popup menu only.

Steps to reproduce the problem

1. Invoke Tasks.
2. Create several tasks.
3. Review the main menu. Right-click on any task
to review the popup menu.

Actual Results

Several menu items are available on the right-click popup menu only, or are available in the menubar after you open a task.

Expected Results

All menu items should be available from the menu bar.

Tools used	: GNU Debugger (GDB)
Languages used	: GTK+, BONOBO
Commands used	: export, grep, vi, configure, make & make install
Platform used	: Red Hat Linux 9.0
Operating environment	: Evolution

Bug #	Product - Version	Component	Status	Short Summary
48133	Evolution - unspecified	Calendar	NEW	Peculiar popup menu in Scheduling tab

Description of Problem

If you create a meeting and click on the Options or Auto pick button in the Scheduling tab, the Popup menu partially covers the button.

Steps to reproduce the problem

1. Invoke Calendar.
2. Choose File > New > Meeting.
3. Select the Scheduling tab.
4. Click on the Options button or Auto pick button.

Actual Results

Popup menu partially covers the button.

Expected Results

Popup menu should display below button.

Tools used	: GNU Debugger (GDB)
Languages used	: GTK+, BONOBO
Commands used	: export, grep, vi, configure, make & make install
Platform used	: Red Hat Linux 9.0
Operating environment	: Evolution

Bug #	Product - Version	Component	Status	Short Summary
61849	Evolution - 1.5.9	Contacts [was: Address book]	NEW	Right-click menu says "New Address book," "New" menu says "Contacts Group"

Description of Problem

New->Contacts Group Contacts tool, right-click in shortcut bar, "New Address book." Change "Address book" to "Contacts Group".

Steps to reproduce the problem

1. Invoke Contacts.
2. Choose File > New > Contacts Group.

Actual Results

It is "Address Book"

Expected Results

Change "Address book" to "Contacts Group".

Tools used	: GNU Debugger (GDB)
Languages used	: GTK+, BONOBO
Commands used	: export, grep, vi, configure, make & make install
Platform used	: Red Hat Linux 9.0
Operating environment	: Evolution

Bug #	Product - Version	Component	Status	Short Summary
16524	Evolution - 1.0.x	Miscellaneous	NEW	Not all buttons have tool tips

Description of Problem

I've noticed that some buttons do not seem to have tool tips and no other visible indication of what they actually do. For example, there's a button with an X in between the trash can and the "display previous" message buttons in the mail browser. However, this button does not seem to work at all and there is no tool tip indicating what it is.

Actual Results

Not all the buttons have tool tips.

Expected Results

Give tool tips for all the buttons.

Tools used	: GNU Debugger (GDB)
Languages used	: GTK+, BONOBO
Commands used	: export, grep, vi, configure, make & make install
Platform used	: Red Hat Linux 9.0
Operating environment	: Evolution

4.2 Database Design

4.2.1 Bugzilla Bug Database

This bug database is used to track the bugs in open source software projects, including GNOME, Evolution, Red Carpet, and Setup Tools.

- **To report a bug...**
- **To search for existing bugs...**

Bugzilla is a database for bugs. It lets people report bugs and assigns these bugs to the appropriate developers. Developers can use Bugzilla to keep a to-do list as well as to prioritize, schedule and track dependencies.

Not all 'bugs' are bugs. Some items in the database are known as *Enhancement Requests* or *Requests For Enhancement* (RFE). An RFE is a bug whose severity field is set to 'enhancement'. People often say 'bug' when they mean 'item in Bugzilla', so RFEs often wind up being called bugs.

Enter the tasks you're planning to work on as enhancement requests and Bugzilla will help you track them and allow others to see what you plan to work on. If people can see your flight plan, they can avoid duplicating your work and can possibly help out or offer feedback.

ANATOMY OF A BUG

Bugs and RFEs are composed of many fields. Some of them are described here.

- **Component**

  The Mozilla application is composed of many different components such as the networking library, javascript, and the layout engine. But your bug to Some components are very similar. For example, problems with table layout should be assigned to HTML Tables, not to Layout The component you choose determines which person Bugzilla assigns the bug to.

- **Status Whiteboard**

  The Status Whiteboard is used for writing short notes about the bug.

- **Keywords**

  This field is used to store various keywords. For instance, the Bug a thon uses it to note when bugs have test cases (using the keyword test case).

TARGET MILESTONE

The Mozilla project uses target milestones to plan Mozilla's development process. If a bug is marked mozilla1.3, it means an assigned developer picked Mozilla 1.3 as his/her estimate of the earliest milestone at which that bug might be resolved. This field should only be set by the person responsible for the bug.

DEPENDENCY

If a bug can't be fixed until another bug is fixed, that's a dependency. For any bug, you can list the bugs it depends on and bugs that depend on it. Bugzilla can display a dependency graphs which shows which the bugs it depends on and are dependent on it.

ATTACHMENT

Adding an attachment to a bug can be very useful. Test cases, screen shots and editor logs all can help pinpoint the bug and help the developer reproduce it. If you fix a bug, attach the patch to the bug. This is the preferred way to keep track of patches since it makes it easier for others to find and test. To make a patch, you need to generate a diff file containing the differences between the file with your changes and the original file in the repository.

To generate a patch file enumerating changes for all files in the current directory try cvs diff -u >mypatch.diff. To apply a patch, go to the proper directory and patch < bugpatch.diff.

LIFE CYCLE OF A BUG

What happens to a bug when it is first reported depends on who reported it. New Bugzilla accounts by default create bugs which are UNCONFIRMED - this means that a QA (Quality Assurance) person needs to look at it and confirm it exists before it gets turned into a NEW bug.

When a bug becomes NEW, the developer will probably look at the bug and either accept it or give it to someone else. If the bug remains new and inactive for more than a week, Bugzilla nags the bug's owner with email until action is taken. Whenever a bug is reassigned or has its component changed, its status is set back to NEW. The NEW status means that the bug is newly added to a particular developer's plate, not that the bug is newly reported.

Those to whom additional permissions have been given have the ability to change all the fields of a bug (by default, you can only change a few). Whenever you change a field in a bug it's a good idea to add additional comments to explain what you are doing and why. Make a note whenever you do things like change the component, reassign the bug, create an attachment, add a dependency or add someone to the CC list. Whenever someone makes a change to the bug or adds a comment, the owner, reporter, the CC list and those who voted for the bug are sent an email (unless they have switched it off) showing the changes to the bug report.

RESOLUTIONS

When a bug is closed it's marked RESOLVED and given one of the following resolutions.

Fixed

A fix for this bug has been checked into the tree and tested by the person marking it FIXED.

Invalid

The problem described is not a bug, or not a bug in Mozilla.

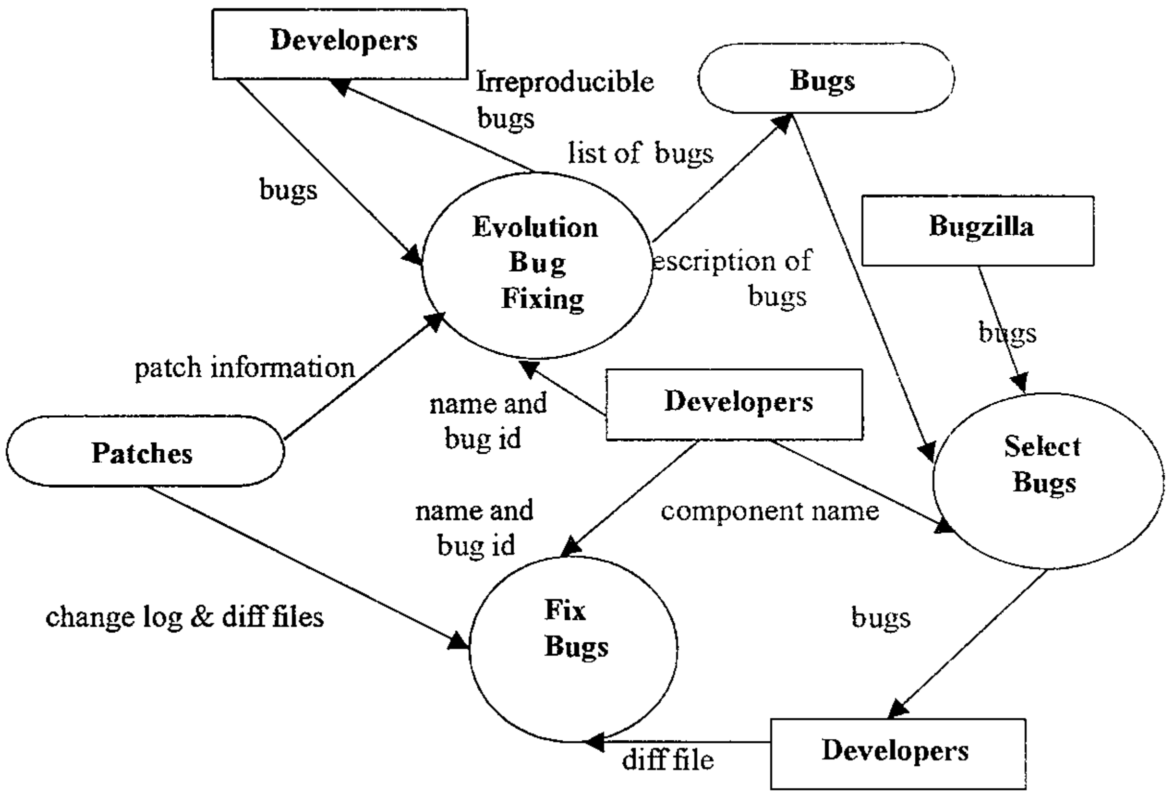
Won fix

The problem described is a bug which will never be fixed, or a problem report which is a "feature", not a bug.

Duplicate

The problem is a duplicate of an existing bug. Marking a bug duplicate requires the bug number of the duplicating bug and will add a comment with the bug number into the description field of the bug it is a duplicate of.

4.3 Process Design



This design explains the way to build Evolution in GNOME Operating Environment. It also explains in detail about the different types of packages used for building modules in GNOME Environment.

One of the easiest ways to build GNOME 2 from source is by using the **vicious-build-scripts**. These build scripts are in CVS and here is a small guide to using these scripts.

Two other ways to building GNOME 2 are by using **GARNOME** (for tarballs), **jhbuild** (for CVS) and **eazel-hacking**.

➢ Jhbuild

Jhbuild is shell script for updating and building the recent codes for the modules present in GNOME. The support tools are installed using the bootstrap before building GNOME. With jhbuild commands it is possible to get the latest source of the GNOME from the CVS repository. Jhbuild must be updated every month to add the new modules included in GNOME. The usage of jhbuild is as follows:

```
jhbuild [ -f config ] command [ options ... ]
```

♦ **Global Options:**

-f, --file=CONFIG	specify an alternative configuration file
--no-interact	do not prompt for input

➢ Commands:

♦ gui	build targets from a gui app
♦ update	update from cvs
♦ updateone modules	update a fixed set of modules.
♦ build [opts...] [modules]	update and compile (the default)
♦ buildone [options...]	modules build a single module
♦ run program [arguments...]	run a command in the build environment
♦ shell	start a shell in the build environment

♦ bootstrap	build required support tools.
♦ list [opts ...] [modules]	list what modules would be built

➢ **Options valid for the update, build and buildone commands:**

♦ -s, --skip=MODULES	treat the given modules (and deps) as up to date
♦ -t, --start-at=MODULE	start building at the given module

➢ **Options valid for the build and buildone commands:**

♦ -a, --autogen	Always run autogen.sh
♦ -c, --clean	run make clean before make
♦ -n, --no-network	skip cvs update

5. SYSTEM IMPLEMENTATION AND TESTING

5.1 System Testing

Software testing is an important element of the software quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of software system element and the costs associated with a software failure are motivating forces for well planned, through testing.

5.1.1 Testing Objectives

There are several rules that can serve as testing objectives. They are,

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding an undiscovered error.
- A successful test is one that uncovers an undiscovered error.

If testing is conducted successfully according to the objectives stated above, it will uncover errors to the working according to the specification that performance requirements appear to have been met.

5.1.2 The GNU Debugger (GDB):

After the bug has been selected from the bug database the particular program for changing the code is found. After this the appropriate changes are made in the code, make install is done. If there is any error during make install the program is debugged with the GDB.

The purpose of a debugger such as GDB is to allow you to see what is going on inside another program while it executes or what another program was doing at the moment it crashed.

Apart from the GDB after the bug is fixed the bug is tested manually by restarting GNOME. If the bug is fixed the patch is sent else the code is again debugged by using the GDB.

GDB can do four main kinds of things (plus other things in support of these) to help you catch bugs in the act:

- ♦ Start your program, specifying anything that might affect its behavior.
- ♦ Make your program stop on specified conditions.
- ♦ Examine what has happened, when your program has stopped.
- ♦ Change things in your program, so you can experiment with correcting the effects of one bug and go on to learn about another.

You can use GDB to debug programs written in C, C++, and Modula-2. Fortran support will be added when a GNU Fortran compiler is ready.

GDB is invoked with the shell command gdb. Once started, it reads commands from the terminal until you tell it to exit with the GDB command quit. You can get online help from gdb itself by using the command help.

5.2 System implementation

The term implementation has different meanings, ranging from the conversion of the basic application to a compatible replacement of the computer system. Implementation is used here to mean the process converting a new of a revised system design in to an operational one. During the implementation stage we convert the detailed code in a programming language.

5.2.1 Goals for Implementation stage

The first goal of implementation is to provide a faithful translation of design. The choice of a language should be pragmatic, governed by mixture of theoretical needs and practical constraints. Good software should avoid any gap between design and code. This is particularly important for reuse of a component or for maintenance work that will require tracing the connection of design to code.

5.2.2 Characteristics of Implementation

- **Abstraction**

  Abstraction deals with the ability of an implementation to allow the programmer to ignore the portion of detail that is not important at the current level of consideration. Each of the three kinds of abstraction-control, data and process should present in the code.

- **Modularization**

  Modularization requires as partitioning the implementation with each abstraction occupying its own separate and identifiable unit.

- **Encapsulation**

  While implementation a design, care should be taken to truly hide within a module.

♦ **Verification**

Assertions used during formal verification of the detailed design should be included as comments in the source code.

Implementation is the stage of the project when theoretical design is turned into a working system. The processing activities in this project fall into two categories. One is file sending and other is file receiving.

♦ **Installation**

After complementing development of the project, the software will be installed in the client side computer. The system should be running on operating system of at least Red Hat.

For the installation of the software the setup of the software has to be created which will help us to install all the components used in the project and with the help of which only the work can run successfully. The setup wizard will setup the product. This will automatically include all the files to setup kit. The database entry and updating should be done manually. Since we place the pages in the network server there is a chance to miss or damage the pages due to the trespassing. So, we have to keep the backup copies of the setup files to required number of floppies or CD's or even in the hard disks and run the file call setup which will install the entire required component to computer. These pages are stand alone and don't need development software to access it.

♦ **Maintenance and Training**

Maintenance of the software is one of the major steps in the computer animation. Software which is developed by the engineer, should undergo maintenance process in regular interval of time goes on new problems arise and it must be corrected accordingly. Maintenance and enhancements are a long-term process. If the problem is derived or upgraded, then also the software should be changed.

In this project, the maintenance is carried over by the staff of the company. Since, they are the key parsons to develop this project they know clearly about the project and coding structures. So they will change the coding whenever required. Regarding the project maintenance, the changes will occur then there according to the conditions.

Various types of maintenance that can be made are

- Corrective maintenance
- Adaptive maintenance
- Prefecture maintenance
- Reverse engineering
- Reengineering

The staffs in the company are parted in each and every level of the project. So, they don't need any training in the software. During the development process, they sit and entered each and every entry to test the project. They themselves used this as the opportunity to take training in the software. So, extra training is not needed for the users.

6. CONCLUSION

Ximian has moved rapidly to spur industry adoption of the Linux platform and the GNOME desktop. In August 2000, Ximian helped drive the formation of the GNOME Foundation, which today includes industry leaders Sun Microsystems, IBM, Red Hat, HP, SuSE and others. In August 2003, Ximian was acquired by Novell, Inc. becoming a fundamental component of Novell's Linux strategy.

Now with the backing, endorsement and resources of Novell, the Ximian Group continues to play a central role in the open source community, providing leadership and core technology to key open source projects and industry groups, including GNOME, the Free Software Foundation, a community initiative to develop an open source, Linux-based version of the Microsoft.NET development platform.

This project aims at creating a better and a more user-friendly version of Evolution-1.5.

7. SCOPE FOR FUTURE DEVELOPMENT

The currently shipping version of Evolution is Evolution 1.5, which will be available during Q3 2004, will deliver important new features including:

- integrated connectivity to Novell GroupWise
- integrated connectivity to Microsoft Exchange
- improved offline support for IMAP accounts
- numerous calendar improvements,
- support for S/MIME, enhanced contact management
- Gaim instant messaging integration
- Improved desktop integration.

Evolution is to make the tasks of storing, organizing, and retrieving your personal information easier, so you can work and communicate with others. That is, it's a highly evolved groupware program, an integral part of the Internet-connected desktop.

8. REFERENCES

http://gnomebangalore.org/

http://gnome.org/boundaries/ - Desktop integration Boundaries

http://gtk.org/ - About the GTK+ tool

http://codeblogs.ximian.com/blogs/evolution/archives/000233.html - Brain-read

http://cvs.gnome.org/viewcvs/evolution-sharp/ - evolution-sharp

http://gnome.org/projects/evolution/ - evolution project web page

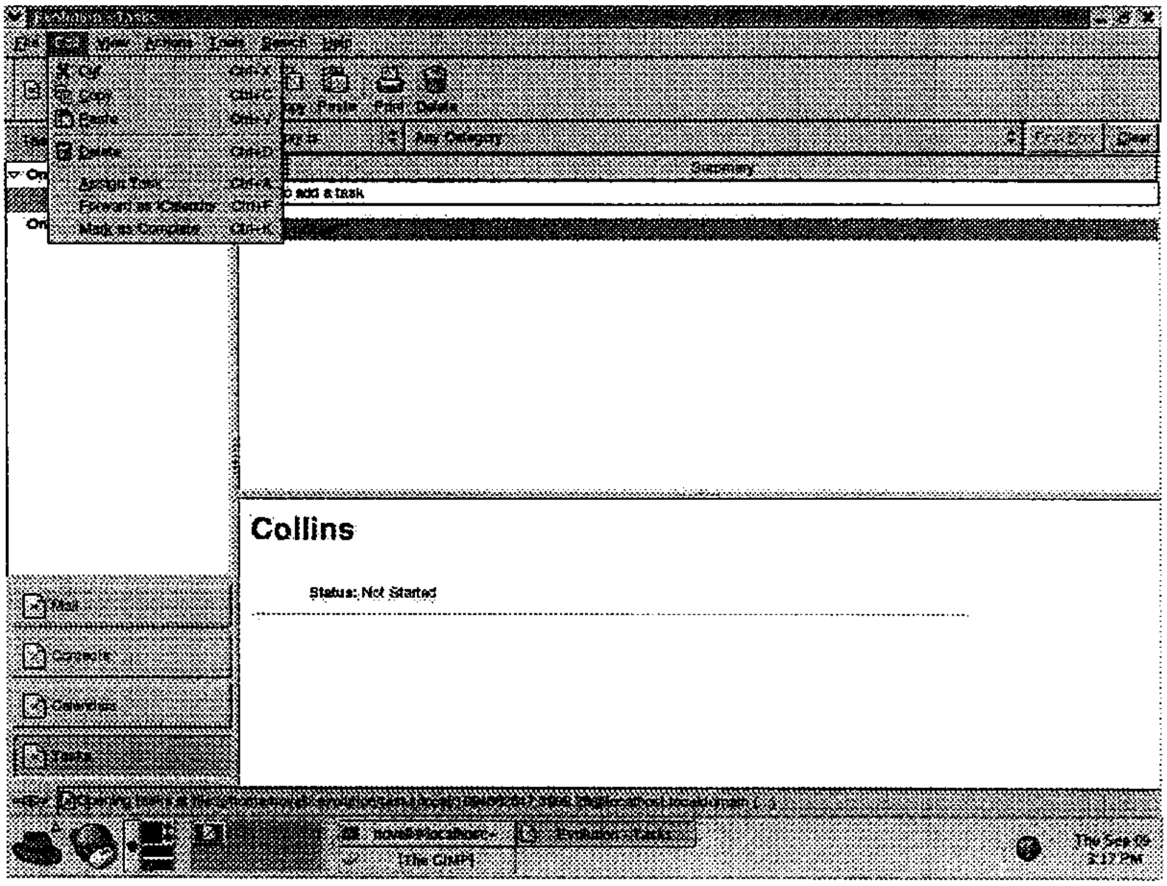
http://gnome.org/projects/evolution/arch.shtml-evolution architecture

http://gnome.org/projects/evolution/doc/c154.html - evolution guide

9. APPENDIX

9.1 Sample Screen Shots

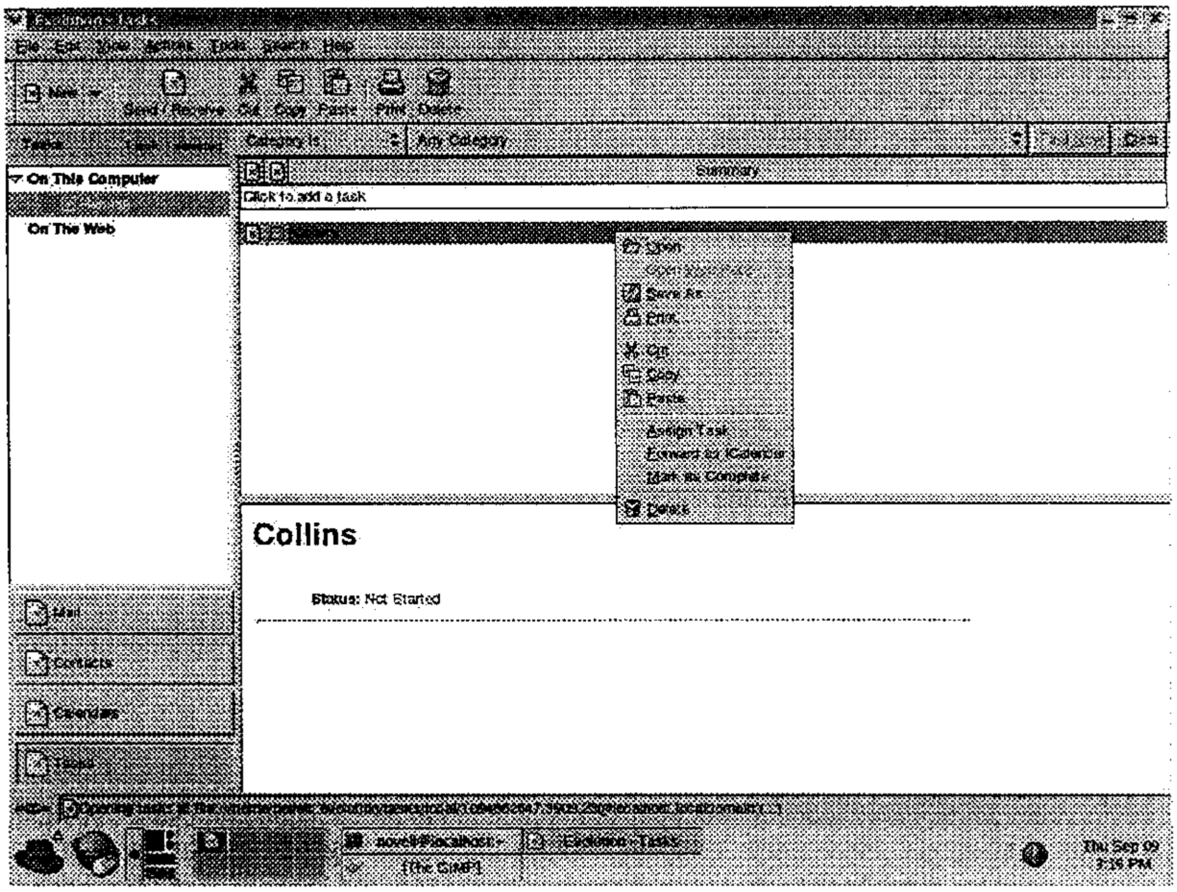
Evolution - Tasks



Collins

Status: Not Started

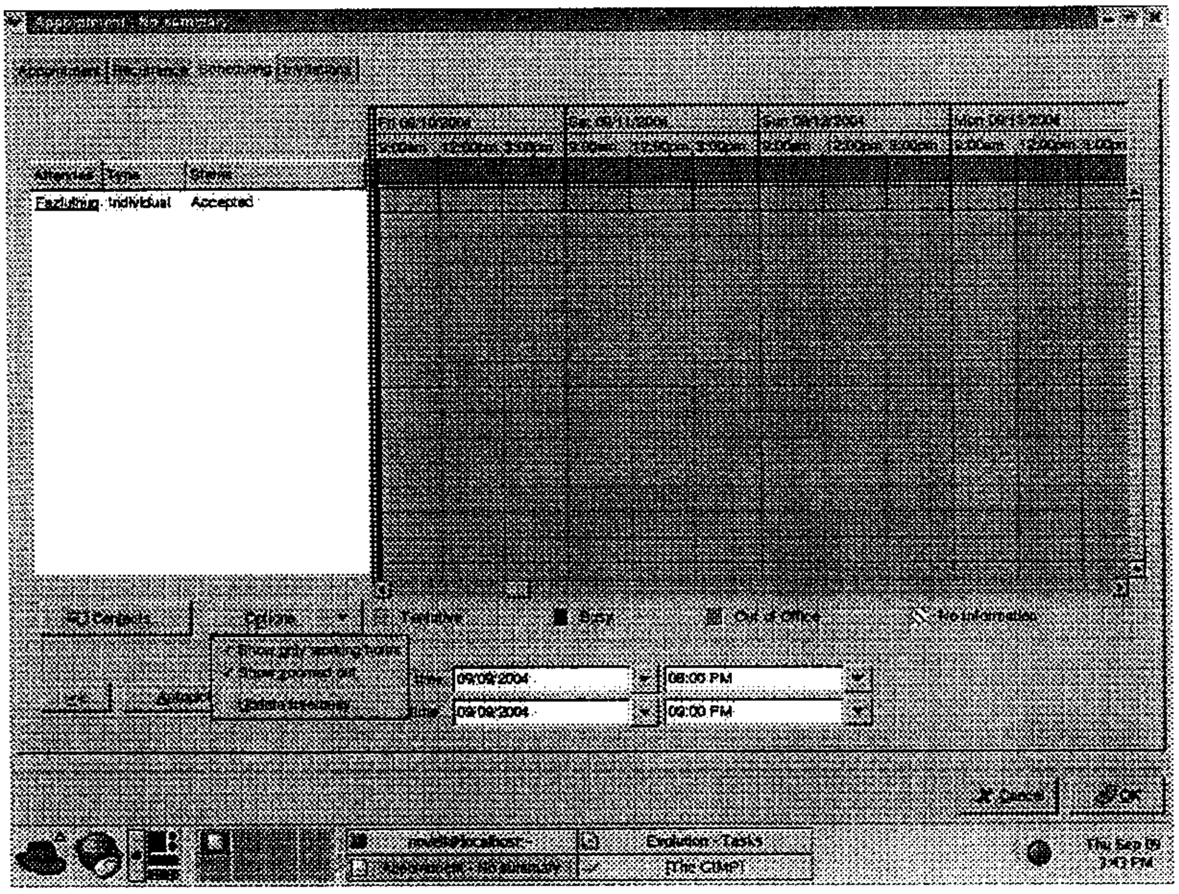
Evolution -Tasks



Collins

Status: Not Started

Appointments



Appointment | Recurrence | Scheduling | Invitations

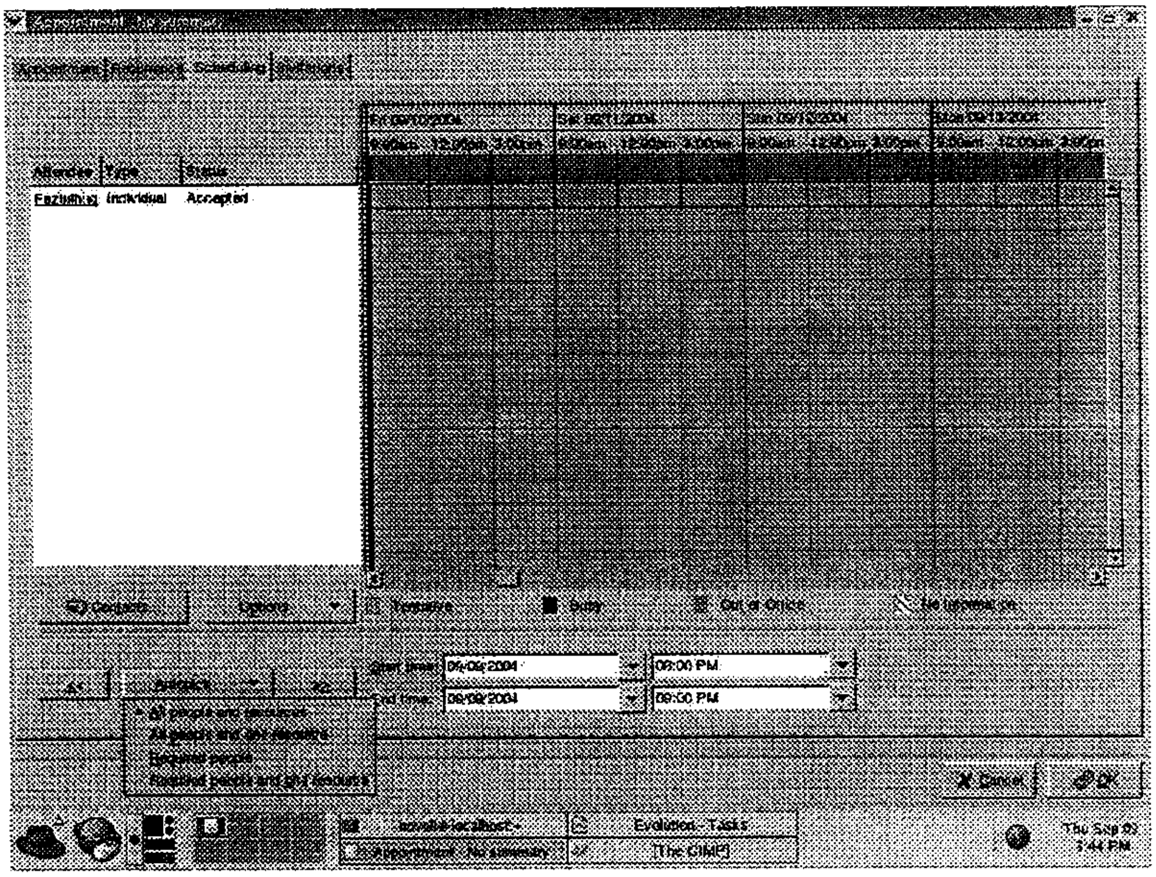
Attendee	Type	Status
Fazlulhuq	Individual	Accepted

Fri 09/10/2004 | Sat 09/11/2004 | Sun 09/12/2004 | Mon 09/13/2004

Start time: 09/09/2004 08:00 PM

End time: 09/09/2004 09:00 PM

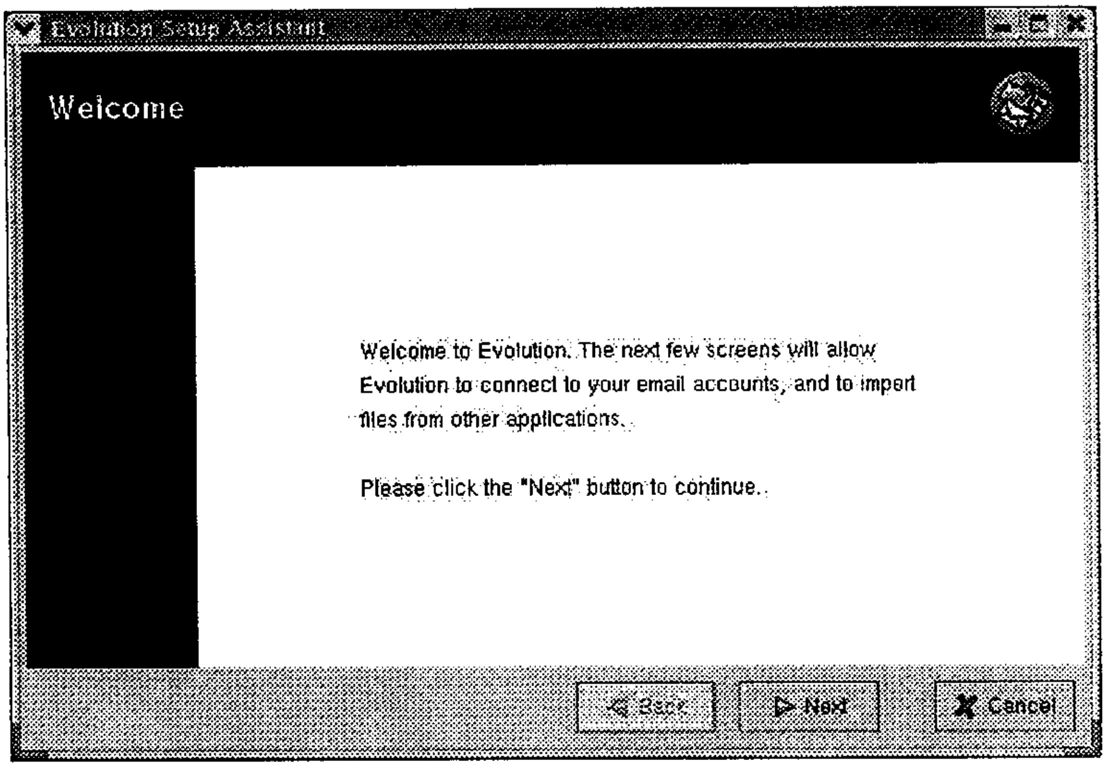
Appointments



Attendee	Type	Status
Faziath(s)	Individual	Accepted

Start time: 09/09/2004 08:00 PM
End time: 09/09/2004 09:00 PM

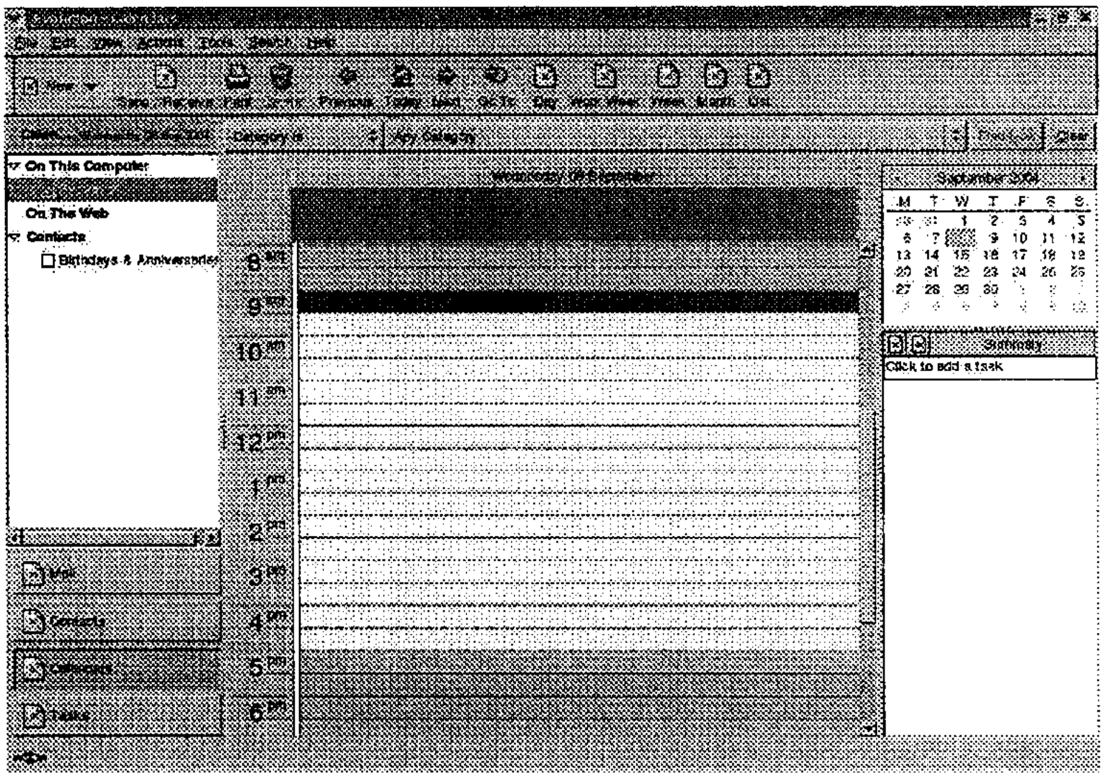
Evolution Setup Assistant



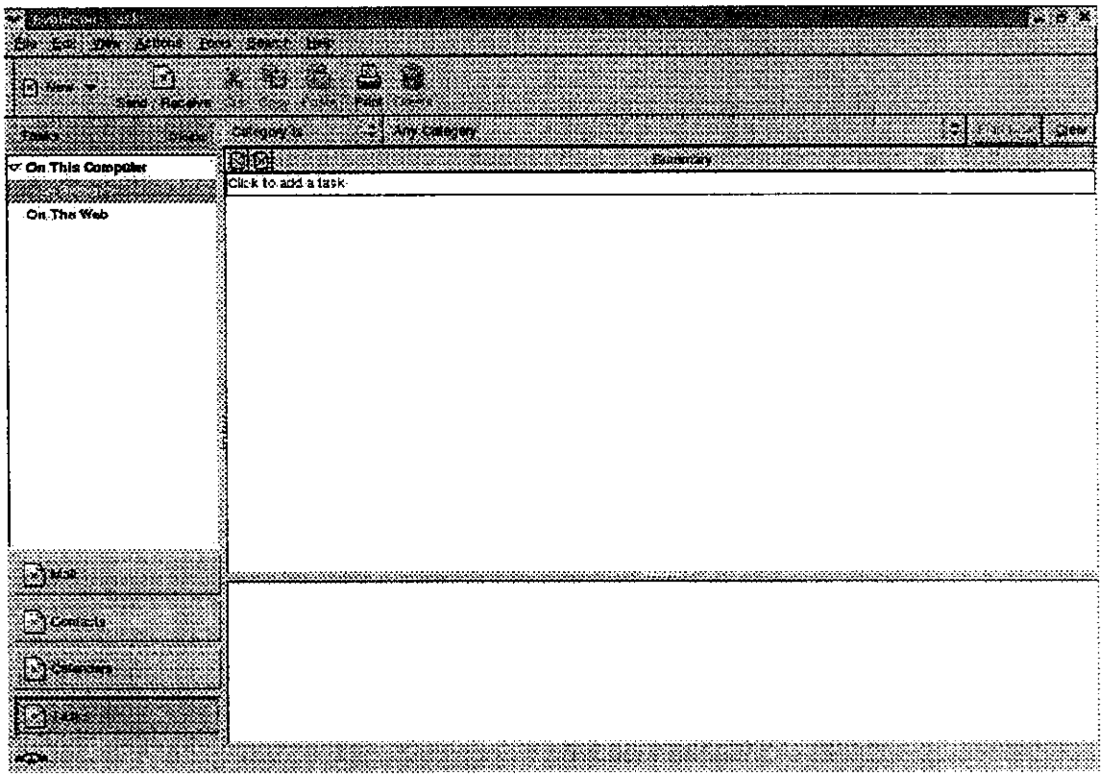
Welcome to Evolution. The next few screens will allow Evolution to connect to your email accounts, and to import files from other applications.

Please click the "Next" button to continue.

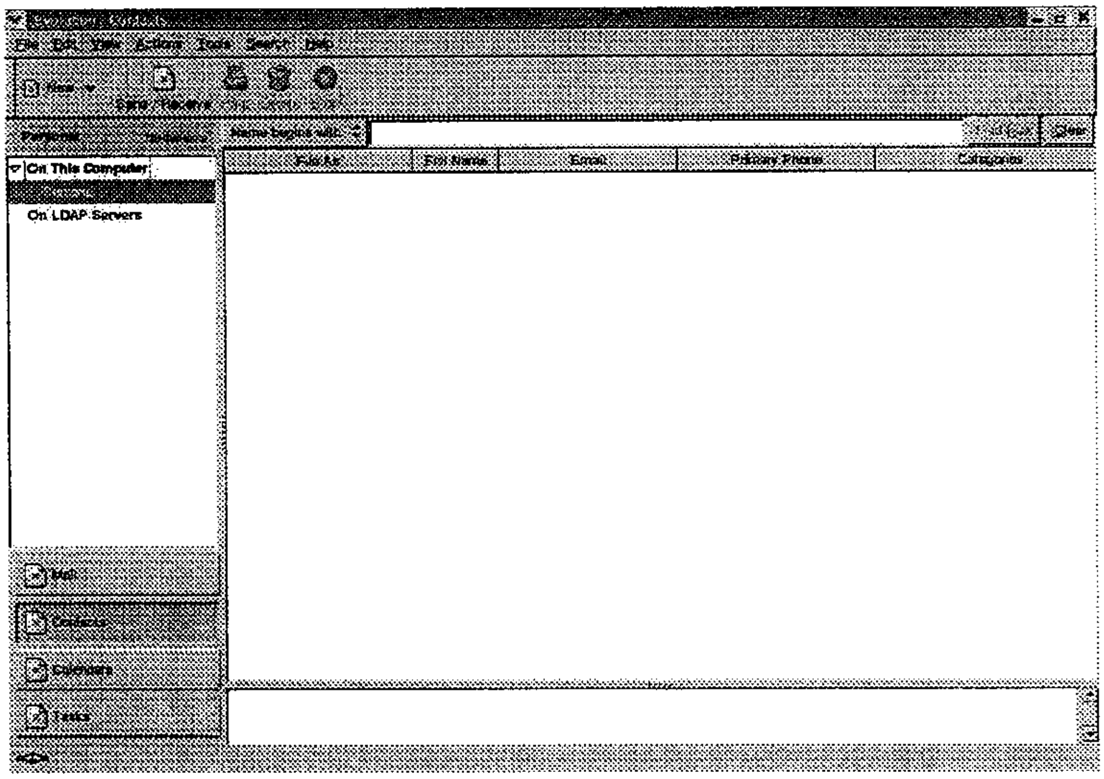
Evolution - Calendar



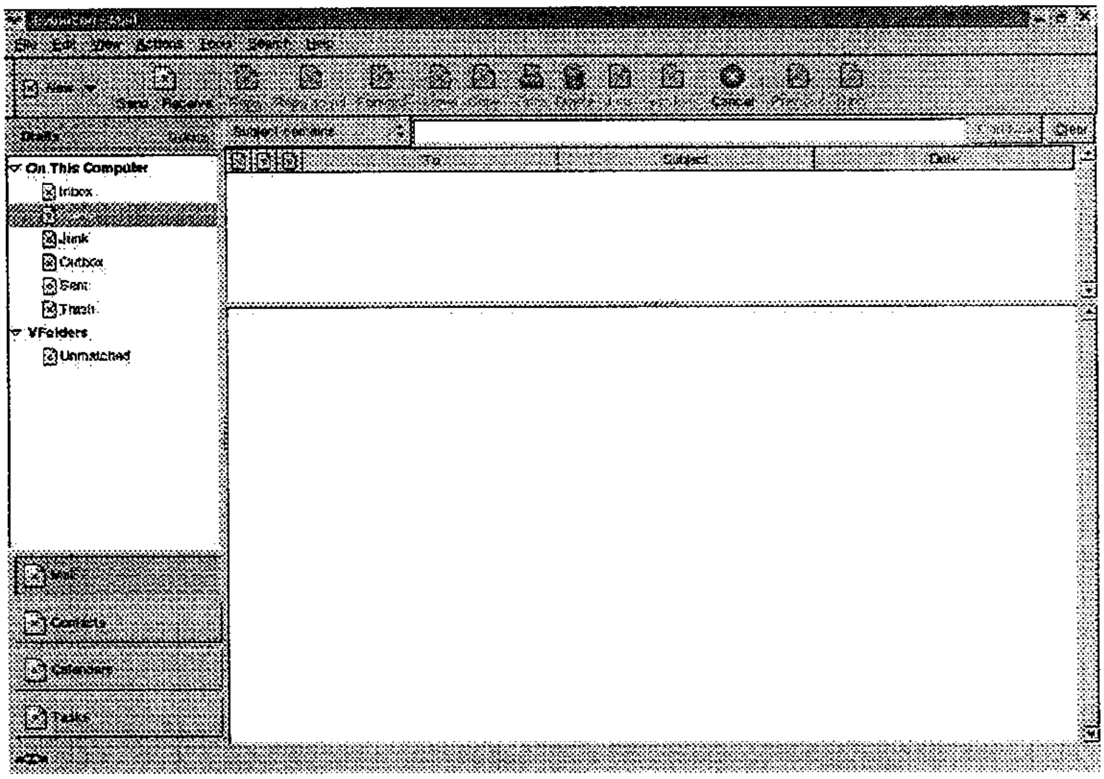
Evolution - Tasks



Evolution - Contacts



Evolution – Mail



File Edit View Actions Tools Search Help
New
Send / Receive
Cancel
Drafts
Subject contains
Clear
To
Subject
Date
On This Computer
Inbox
Junk
Outbox
Sent
Trash
VFolders
Unmatched
Mail
Contacts
Calendars
Tasks