P- 1229

KUMARAGURU COLLEGE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DEBUGGING GNOME

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
HANNAH REBEKAH.F
REG.NO:0137S0032

GUIDED BY
Mr.C.RAJANKRUPA, M.C.A., 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 GNOME

Done By

HANNAH REBEKAH.F

Reg. No. 0137S0032

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 ..29/09/04..........

[illegible]
Internal Examiner

[illegible]
External Examiner

DECLARATION

I here by declare that the project work entitled

"DEBUGGING GNOME"

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

Mr.C.Rajankrupa M.C.A

Name of Candidate	Register Number
Hannah Rebekah.F	**0137S0032**

Signature of Candidate: [illegible]

Date: 24/09/04

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 Department**, Kumaraguru College of Technology, giving me this golden opportunity to carry out my project work successfully.

I like to express my deep sense of gratitude to **MR. K.R.Baskar Project coordinator, CSE Department.** I am grateful for his inspiring guidance and wholehearted support, which he extended to me throughout the course of this project.

My sincere thanks are offered to **Mr.C.Rajankrupa 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 owe my gratitude to **Mr. Rajiv Ranjan, Team leader, Novell Software Development Limited,** for rendering permission to carry out my project work in this esteemed concern.

I am extremely thankful to **Mr. Chenthill Palanisamy & Mr.Siva, Software Engineers, Novell Software Development Limited,** for their kind help and guidance for my project. I am grateful for their intensive training and also for lending me their equipments.

I owe my gratitude to my parents who have encouraged me a lot towards the successful completion of this project work.

SYNOPSIS

The project works on debugging and enhancing the GNOME source code that is developed by high professionals all over the world for the sake of bug-free and user-friendly environment. A bug is picked up from the www.bugzilla.gnome.org where the bugs are posted. Once the bug is solved and patched to the site, the resolution is turned on to *FIXED* if the maintainer approves to it. The enhancements also provide ease in operation. This project and its further developments would surely enable millions of users around the globe to work in an easy and efficient operating environment.

This project involves in providing a bug free operating environment for the technical and cultural world of LINUX, an operating system that is highly valued for its openness, user flexibility, design simplicity, and its rich collaborative historical client and simplified printing configuration, as well a video conferencing system legacy. In this project we have solved bugs and provided enhancements for various modules in the GNOME 2.8 that is due for release. The new GNOME desktop offers several new features, such as a universal instant messaging.

The following are some of the bugs that were solved in this project:

- Task date & time order is incorrect.
- The option to delete all the appointments at a time is not present.
- Users have requested a menu item, which functions like "expunge" in the mailer.
- Ctl+n to create new compose windows.
- Change welcome message.
- Strange English in address book message.
- Typo in string: Word "be" missing.

- OK button active in "Select source" with no calendar selected.

This project is part of the Novell Open Source Internship Program (NOSIP) started by Novell, Bangalore to attract more people, especially students to the open source community. Novell Open Source Internship Program is for final or penultimate year students and who want to work on Open Source project, under the guidance of Novell.

The objective of this program is to encourage the open source initiatives in the student community in India. It is well known that the most significant contributions to the Open Source product development come from students in Europe and America. India at this points lags in this.

Open Source Software development is very good for students because they get the chance to work with some of the most skilled engineers and do the most challenging product development work. They also get the public recognition and acknowledgement of their contributions, which is very useful in the long term. Most professional courses also have a mandatory requirement to do an industry project.

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

GNOME involves programming in the GObject System and uses GNOME technologies like Bonobo, gtk etc.

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.
- Code compilation.
- Obtaining the final design.
- Testing the system.
- Implementing the system.
- Documenting the project.

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 following few lines gives a brief introduction on Linux and GNOME environment.

About 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.

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.

NetworkinGg 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.

About GNOME: -

GNOME is a UNIX and Linux desktop suite and development platform. The GNOME (GNU Network Object Model Environment) project was started in August 1997 by *Miguel de Icaza* as an attempt to provide a free software desktop for the GNU/Linux operating system. Like many other open source projects, they use the World-Wide Web and e-mail for intercommunication, the GNU tool suite (GCC, make, e-macs, gdb, for software development), CVS (for software configuration management), and *bugzilla* (for bug tracking) as their main development platform.

The project currently involves around five hundred developers. In 1997, the first tarball is posted (version 0.10). Version 1.0 was released in March 1999, which is integrated into Red Hat Linux as its default desktop. Version 1.4 is the latest version, released in 2001, and version 2.0 is in beta testing. GNOME is composed of a large collection of programs and libraries, comprising millions of lines of code.

Novell.

ORGANISATION PROFILE

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, Border Manager 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.

Novell Bangalore understands 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, and 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

2002 EDITORS' CHOICE LINUX

"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

E TREME

"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

The project requires 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
- Red Hat Linux 7.3
- 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.

The project is done on the GNOME desktop. The project can be done only on the above-mentioned OS. If any other OS is used,(ie Fedora Core) some of the components for doing the project are likely to be missing.

2.2 Existing System

The GNOME 2.6 Desktop is the existing version of the ever popular, multi-platform free desktop environment.

The main components of the existing version of GNOME: -

- The file manager
- Control center
  - Keyboard layouts
  - Desktop background
- Panel applets
  - Network monitor
  - Keyboard indicator
  - Keyboard accessibility status

The applications of GNOME are: -

- Web browser
- GNOME meeting
- Abi word

The utilities of GNOME are: -

- PDF viewer
- Sound recorder
- Volume control
- Search for files
- Text editor
- Keyboard accessibility status
- Screen reader and magnifier
- On-screen keyboard
- Dasher
- Lockdown
- Online help
- Vector graphics

The project currently involves around five hundred developers. In 1997, the first tar ball is posted (version 0.10). Version 1.0 was released in March 1999, which is integrated into Red Hat Linux as its default desktop. Version 2.6 is the latest version; released version 2.8 is in beta testing. GNOME is composed of a large collection of programs and libraries, comprising millions of lines of code.

The current beta version of GNOME consists of hundreds of bugs. These bugs cause a lot of inefficiency in the desktop.

The following are some of the sample bugs: -

Bug#	Component	Status	Short Summary
57320	Calendar	NEW	Week view not persistent

Bug#	Component	Status	Short Summary
56889	Calendar	NEW	OK button active in "Select source" with no calendar selected

Bug#	Component	Status	Short Summary
52679	Miscellaneous	NEW	Change welcome message

Bug #	Component	Status	Short Summary
469850	Calendar	NEW	Evolution freezes when running different operations involving Calendar or Tasks.

Bug #	Component	Status	Short Summary
56787	Tasks	NEW	Unable to view tasks or appointments.

Bug #	Component	Status	Short Summary
50703	Calendar	NEW	An appointment is not displayed correctly.

Bug #	Component	Status	Short Summary
58400	Shell	NEW	Deleting folder has wrong title.

Bug #	Component	Status	Short Summary
45084	Calendar	NEW	Crash: moused over the application

Bug #	Component	Status	Short Summary
60853	Calendar	NEW	No alarm/reminder for birthdays

All the bugs are filed in the bug tracking system called the Bugzilla bug tracking system. These bugs are picked up from his tracking system and the bugs can be worked upon. Once the bugs are fixed the patches are reported to the "patch list". The patches are surveyed upon and if the patches cause no further problems the patches are added to the source code.

This project involves in providing a bug free operating environment for the technical and cultural world of LINUX, an operating system that is highly valued for its openness, user flexibility, design simplicity, and its rich collaborative historical client and simplified printing configuration, as well a video conferencing system legacy. In this project we have solved bugs and provided enhancements for various modules in the GNOME 2.8 that is due for release.

2.3 Proposed System

Looking to GNOME 2.8 and beyond

GNOME operates on time-based release philosophy, an attempt to continuously provide the best of the developers' efforts to users as quickly as possible. Six months after GNOME 2.8 will feature integrated email, contacts, calendar and multimedia functionality, and further advances in accessibility, usability and internationalization.

Getting Involved

The core of GNOME's success is its many volunteers, both users and developers. As a user, your contribution can be as simple as filling good bug reports. You can file bugs in bugzilla using the simple bug assistant, If you want to contribute more, you can join our activate bug-squad.

For developers, there is much exciting progress to be made in any of our active developer groups – Accessibility, Documentation, Usability, Translation, Web, Testing, Graphics and Desktop & Platform Development. Many of these sub-projects have web pages on developer.gnome.org. Choosing a role that suits you may be difficult at first, but here is a guide to help you make your decision. Helping on GNOME can be an incredibly satisfying experience, allowing you to meet a wide range of motivated, skilled, and helpful people all working towards a unified goal.

The user will be able to perform the following functions: -

- Users will learn to use your program faster, because interface elements will look and behave the way they are used tc.
- Novice and advanced users alike will be able accomplish tasks
  - Quickly and easily, because the interface won't be confusing or make things difficult.
  - Your application will have an attractive look that fits in with the rest of the desktop.
  - Your application will continue to look good when users change desktop themes, fonts and colors.

- Your application will be accessible to all users, including with disabilities or special needs.

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 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 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.

3.2.2 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 that 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 are 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 interfaces 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, and Guile.

3.2.3 CORBA

CORBA allows programs to communicate regardless of their programming language, the platforms and machines they are running on. The communication between heterogeneous programs is guaranteed due to the IIOP protocol. IIOP is a TCP/IP based protocol specified by the OMG consortium. In CORBA, an object is an instance of a class, which is defined by a set of operations, attributes and exceptions. Every object provides well-defined services through its interface, and every object has a reference by which it is identified in a distributed environment.

3.2.4 Object Request Brokers

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.

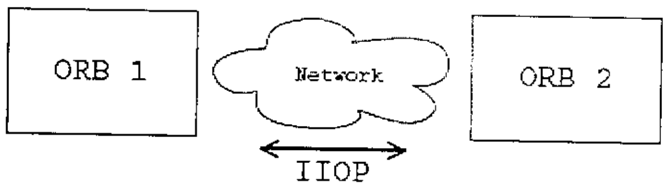


Figure 2.1: ORB interoperability

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.

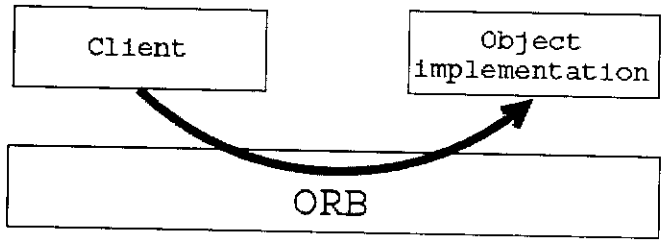


Figure 2.2: An ORB communicating a request to a target object implementation

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.

3.2.5 BONOBO

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.

Let's suppose that a developer wants to write a mail client program that is capable of dealing with attached files. The program should then be able to view images, run videos, render HTML documents, etc. In this case, the use of components should be considered. For example, there is no need for the program to know how to plot images. This functionality should be provided by a component like the image viewer Eye of GNOME (EOG).

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 to manipulate 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, Bonobo is a set of CORBA interfaces and their default implementation in C.

3.2.6 ORBIT

ORBit is the CORBA implementation that was written for the GNOME project. All the Bonobo IDL interfaces are implemented using the ORBit ORB. GNOME developers had considered the use of two ORBs before Elliot Lee and Dick Porter decided to write a new one. They considered Xerox's ILU but they couldn't get Xerox to modify its licensing terms. Then, they adopted MICO for a while but its performance was quite poor.

ORBit is a CORBA 2.2 compliant ORB. It was designed to have a high performance, which means low memory usage and high speed. ORBit is written in C and runs over multiple platforms. It has many language bindings including C, Perl and C++.

Most of the time Bonobo components will be running on the same process. However they will be using ORBit for communicating. This seems a little bit awkward since CORBA is primarily used for distributed objects in a networked environment. Sending IIOP messages from the stub to the skeleton in order to access a local object does not help performance. Actually, ORBit treats the case of a local object separately. The shortcut is indeed included in the stub of every object.

4. SYSTEM DESIGN

4.1 Input design

Bug 1: -

Bug #	Component	Status	Short Summary
65051	Tasks	NEW	Task date & time order.

Description of Problem: -

Task date & time order is incorrect, generally task start day will mention and then end date is given. But while creating task its vice versa on creating assigned task.

Steps to reproduce the problem: -

1. Launch application.
2. Create new assigned task
3. Say start date September 10, 2004 and end date September 1, 2004

Actual Results: -

Task is sent.

Expected Results: -

As start date is grater than end date it should not able to send new task, at least it should give a warning.

TOOLS USED: GNU Debugger (GDB)

LANGUAGES USED: GTK+, BONOBO

COMMANDS USED: export, find, grep, vi, configure, make and make install.

PLATFORM USED: RED HAT LINUX 9.0

OPERATING ENVIRONMENT: GNOME.

Bug 2: -

Bug #	Component	Status	Short Summary
16253	Tasks	NEW	Expunge menu in Tasks.

Description of Problem: -

Speaking of menus, users have also requested a menu item which functions like "expunge" in the mailer; that is, which permanently erases all completed tasks. This should be added to the Edit menu.

Steps to reproduce the problem: -

1. Launch application.
2. View edit menu.

Actual Results: -

'Expunge' not found.

Expected Results: -

'Expunge' option should be added.

TOOLS USED: GNU Debugger (GDB)

LANGUAGES USED: GTK+, BONOBO

COMMANDS USED: export, find, grep, vi, configure, make and make install.

PLATFORM USED: RED HAT LINUX 9.0

OPERATING ENVIRONMENT: GNOME.

Bug 3: -

Bug #	**Component**	**Status**	**Short Summary**
6592	Miscellaneous	NEW	ctl+n to create new compose windows

Description of Problem: -

Pressing ctrl-n on when a compose window is open doesn't make another compose window you have to put the focus on the main window if you want to open another window.

Steps to reproduce the problem: -

1. Launch application.
2. Open new compose window.
3. Press ctl+n.

Actual Results: -

New compose window does not opened.

Expected Results: -

It might be helpful if pressing ctrl-n when having a compose window also worked.

TOOLS USED: GNU Debugger (GDB)

LANGUAGES USED: GTK+, BONOBO

COMMANDS USED: export, find, grep, vi, configure, make and make install.

PLATFORM USED: RED HAT LINUX 9.0

OPERATING ENVIRONMENT: GNOME.

Bug 4: -

Bug #	**Component**	**Status**	**Short Summary**
60336	Calendar	NEW	The option to delete all the appointments at a time is missing.

Description of Problem: -

I am having some 15 contacts for a particular. I tried to delete all the contacts at a time. But does not provide option to delete the all the appointments at once.

Steps to reproduce the problem: -

1. Launch application.
2. Delete appointments.

Actual Results: -

All appointments are not deleted.

Expected Results: -

All appointments must be deleted at once.

TOOLS USED: GNU Debugger (GDB)

LANGUAGES USED: GTK+, BONOBO

COMMANDS USED: export, find, grep, vi, configure, make and make install.

PLATFORM USED: RED HAT LINUX 9.0

OPERATING ENVIRONMENT: GNOME.

Bug 5: -

Bug #	Component	Status	Short Summary
52679	Evolution	NEW	Change welcome message

Description of Problem: -

Welcome message needs to point to http://support.ximian.com for all technical support questions.

Steps to reproduce the problem: -

1. Launch application.
2. View welcome message.

Actual Results: -

Welcome message does not point to http://support.ximian.com.

Expected Results: -

Welcome message needs to point to http://support.ximian.com.

TOOLS USED: GNU Debugger (GDB)

LANGUAGES USED: GTK+, BONOBO

COMMANDS USED: export, find, grep, vi, configure, make and make install.

PLATFORM USED: RED HAT LINUX 9.0

OPERATING ENVIRONMENT: GNOME.

4.2 Database Design

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. bute 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 to note when bugs have test cases (using the keyword testcase).

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.

Wontfix

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

4.3.1 GNOME

The two main modules that make up the GNOME architecture namely ORBit and Bonobo basically relies on CORBA. ORBit is nothing but an implementation of CORBA while Bonobo relies on CORBA for its communication layer.

GNOME is Free Software and part of the GNU project, dedicated to giving users and developers the ultimate level of control over their desktops, their software, and their data. GNOME understands that usability is about creating software that is easy for everyone to use, not about piling on features.

The GNOME project provides two things: The GNOME desktop environment, an intuitive and attractive desktop for end-users, users, and the GNOME development platform, an extensive framework for building applications that integrate into the rest of the desktop. GNOME's community of professional and volunteer usability experts have created Free Software's first and only Human Interface Guidelines, and all core GNOME software is adopting these principles. Free Software is about enabling software freedom for everyone, including users and developers with disabilities. GNOME's Accessibility framework is the result of several years of effort, and makes GNOME the most accessible desktop for any UNIX platform.

GNOME is used, developed and documented in dozens of languages, and we strive to ensure that every piece of GNOME software can be translated into all languages. GNOME strives to be an organized community, with a foundation of several hundred members, usability, accessibility, and QA teams, and an elected board. GNOME releases are defined by the GNOME

Release Team and are scheduled to occur every six months. Perhaps more than anything else, GNOME is a worldwide community of volunteers who hack, translate, design, QA, and generally have fun together.

Bonobo is the component model that derived from the GNOME project. GNOME is a free desktop environment as well as a suite of powerful applications like the spreadsheet *Gnumeric*, the word processor AbiWord or the file manager Nautilus.

Several companies like Ximian, Red Hat, SuSe, Mandrake Soft and Sun Microsystems distribute GNOME. Theoretically, GNOME can run on any UNIX-like platform supporting the X Windows System. Here are some of the platforms that were successfully tested: Linux (any recent distribution), Solaris, FreeBSD and NetBSD. Even though GNOME is based on the X Windows System, it is not tied to a specific window manager.

From the developer's point of view, GNOME presents a powerful framework for software development. Indeed, GNOME makes use of cutting edge technologies. Among the modules that make up the GNOME architecture are:
- GTK+: The toolkit for creating user interfaces.
- ORBit: A thin and fast implementation of CORBA.
-Bonobo: The architecture for creating reusable software components and compound documents.
- OAF: A framework for activating CORBA servers.

4.3.2 GTK+

The Bonobo distribution consists of a collection of CORBA IDL interfaces and their default implementation. The implementation is in C and is based on the GTK+ library. More specifically, it uses the GTK+ object system, which allows writing C programs with an object oriented programming approach.

GTK+ stands for the GIMP Toolkit. Originally, it was developed for the GIMP, but now it is being used in many applications and constitutes the basis for GNOME's user interface.

GTK+ is essentially a library for creating graphical user interfaces. It is built on top of the GLib library which enhances programs portability. GTK+ is completely written in C, but many language bindings are available including C++, Perl and Python. GTK+ was written with an object-oriented approach based on the GTK+ object system. GNOME adopted GTK+ mainly because of its flexible and convenient programming interface, but also because of its licensing terms.

4.3.3 GLIB LIBRARY

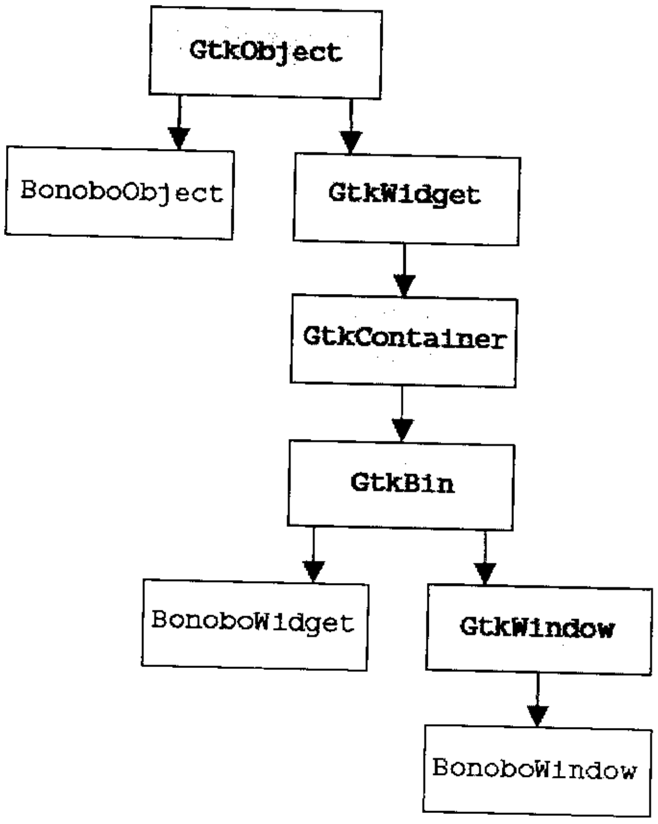
The GLib library is written in C and contains some convenient functions and definitions. For instance, the GLib library defines basic types, memory allocation routines and useful macros like those dealing with type conversion and byte order. It also provides useful data structures like arrays, hash tables and linked lists, and includes utility functions like those related to strings.

GLib replaces many standard and common C language constructs. These replacement functions and definitions make applications easier to write and insulate them from the details of different operating systems. In other words, the advantages of using the GLib library are to increase the portability of applications and libraries, and to provide some utility functions.

4.3.4 The GTK+ object system

GtkObject forms the basis of the GTK+ object system. It plays a similar role as the Object class in Java. Every GtkObject is made up of two structures, one structure that represents an object, and another that represents a class.

A **Gtk object** is an object that inherits from GtkObject. Sometimes, when there is no confusion, a Gtk object will simply be called an **object**. BonoboObject, GtkWidget and GtkButton are all examples of Gtk objects. Figure 2.3 shows a hierarchical tree of some Gtk objects, starting from the first object, which is GtkObject. The white rectangles represent some Gtk objects used in the Bonobo distribution.



A Hierarchical Tree of some Gtk Objects

4.3.5 The GTK+ type system

GTK+ has also a type system that is used by the object system to check the validity of casts. Every Gtk object has a type, and every type is associated with a unique identifier represented by an unsigned integer. The identifier is assigned by the type system when the Gtk type unique function is called.

The Gtk type unique function registers the new type with the type system. It takes the type of the parent object as its first argument. This allows the type system to make a hierarchy of types, which can be used to check the validity of casts. The validity test is done by calling the Gtk type is a function, which takes two identifiers as arguments and verifies that one identifier is the other's ancestor. When writing a Gtk object, it is common to have a function that registers the Gtk object with the system. The registration is made on the first call of the function. On subsequent calls, the function simply returns the type's identifier of the Gtk object.

These functions allow the initialization of the object and class structures. The Gtk objects that inherit from BonoboXObject define their get type function by using the BONOBO X TYPE FUNC FULL macro, which is defined in the bonobo xobject.h file.

4.3.6 Building GNOME

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 ... ]
```

a) Global Options:

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

b) 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

c) 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

d) 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

- **Concurrent Version System (CVS)**

CVS (Concurrent Version System) began life as a front end to RCS developed in the early 1990s, but the model of version control it uses was different enough that it immediately qualified as a new design. Modern implementations don't rely on RCS.

Unlike RCS and SCCS, CVS doesn't exclusively lock files when they're checked out. Instead, it tries to reconcile non-conflicting changes mechanically when they're checked back in, and requests human help on conflicts. The design works because patch conflicts are much less common than one might intuitively think.

The interface of CVS is significantly more complex than that of RCS, and it needs a lot more disk space. These properties make it a poor choice for small projects. On the other hand, CVS is well suited to large multideveloper efforts distributed across several development sites connected by the Internet. CVS tools on a client machine can easily be told to direct their operations to a repository located on a different host.

The open-source community makes heavy use of CVS for projects such as GNOME and Mozilla. Typically, such CVS repositories allow anyone to check out sources remotely. Anyone can, therefore, make a local copy of a project, modify it, and mail change patches to the project maintainers. Actual write access to the repository is more limited and has to be explicitly granted by the project maintainers. A developer who has such access can perform a commit option from his modified local copy, which will cause the local changes to get made directly to the remote repository.

An example for this is a well-run CVS repository, accessible over the Internet, at the GNOME CVS site. This site illustrates the use of CVS-aware browsing tools such as Bonsai, which are useful in helping a large and decentralized group of developers coordinate their work.

The social machinery and philosophy accompanying the use of CVS is as important as the details of the tools. The assumption is that projects *will* be open and decentralized, with code subject to peer review and inspection even by developers who are not officially members of the project group.

Just as importantly, CVS's non-locking philosophy means that projects can't be blocked by a lock if a programmer disappears in the middle of making some changes. CVS thus allows developers to avoid the "single person point of failure" problem; in turn, this means that project boundaries can be fluid, casual contributions are relatively easy, and projects are not required to have an elaborate hierarchy of control.

The CVS sources are maintained and distributed by the FSF. CVS has significant problems. Some are merely implementation bugs, but one basic problem is that your project's file namespace is not versioned in the same way changes to files themselves are. Thus, CVS is easily confused by file renamings, deletions, and additions. Also, CVS records changes on a per-file basis, rather than as *sets* of changes made to files. This makes it harder to back out to specific versions, and harder to handle partial check-ins. Fortunately, none of these problems are intrinsic to the non-locking style, and they have been successfully addressed by newer version-control systems.

♦ STEPS INVOLVED IN BUILDING MODULES

Basically there are three packages involved in building modules. They are

- Autoconf.
- Automake.
- Libtool.

Autoconf, *Automake* and *Libtool* are packages for making your software more portable and to simplify building it--usually on someone else's system. Software portability and effective build systems are crucial aspects of modern software engineering practice. It is unlikely that a software project would be started today with the expectation that the software would run on only one platform. Hardware constraints may change the choice of platform, new customers with different kinds of systems may emerge or your vendor might introduce incompatible changes in newer versions of their operating system. In addition, tools that make building software easier and less error prone are valuable.

Autoconf is a tool that makes your packages more portable by performing tests to discover system characteristics before the package is compiled. Your source code can then adapt to these differences.

Automake is a tool for generating `Makefile's--descriptions of what to build--that conform to a number of standards. Automake substantially simplifies the process of describing the organization of a package and performs additional functions such as dependency tracking between source files.

Libtool is a command line interface to the compiler and linker that makes it easy to portably generate static and shared libraries, regardless of the platform it is running on.

◆ <u>How To Run Configure And Make</u>

A package constructed using Autoconf will come with a `configure' script. A user who wants to build and install the package must run this script in order to prepare their source tree in order to build it on their particular system. The actual build process is performed using the make program.

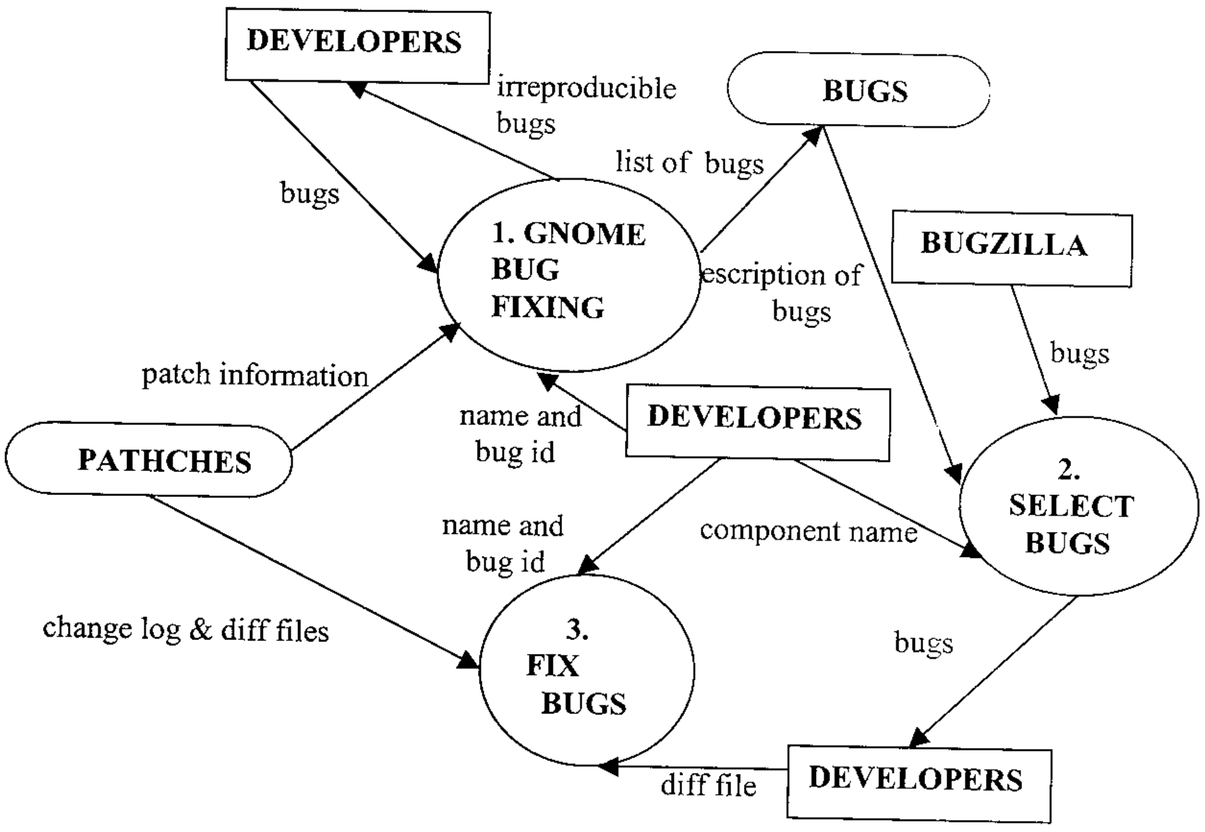
The ‘configure’ script tests system features. For example, it might test whether the C library defines the time_t data type for use by the time( ) C library function. The ‘configure’ script then makes the results of those tests available to the program while it is being built.

This chapter explains how to invoke a ‘configure’ script from the perspective of a user--someone who just wants to take your package and compile it on their system with a minimum of fuss. It is because Autoconf works as well as it does that it is usually possible to build a package on any kind of machine with a simple configure; make command line.

♦ <u>**About Makefile**</u>

A 'Makefile' is a specification of dependencies between files and how to resolve those dependencies such that an overall goal, known as a *target*, can be reached. 'Makefile's are processed by the make utility. Other references describe the syntax of 'Makefile's and the various implementations of make in detail. This chapter provides an overview into 'Makefile's and gives just enough information to write custom rules in a 'Makefile.am'

DFD: -


DEVELOPERS
irreproducible bugs
BUGS
list of bugs
bugs
1. GNOME BUG FIXING
escription of bugs
BUGZILLA
bugs
patch information
name and bug id
DEVELOPERS
PATHCHES
2. SELECT BUGS
name and bug id
component name
change log & diff files
3. FIX BUGS
bugs
diff file
DEVELOPERS

5. SYSTEM TESTING AND IMPLEMENTATION

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.

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 com mands 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.

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.

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

a) 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, process should present in the code.

b) Encapsulation

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

c) 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.

d) 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.

e) 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

6. Conclusion

Open Source Software development is very good for students because they get the chance to work with some of the most skilled engineers and do the most challenging product development work. They also get the public recognition and acknowledgement of their contributions, which is very useful in the long term. Most professional courses also have a mandatory requirement to do an industry project.

Novell provides an opportunity for the students to do this project under the guidance of Novell engineers on the open source products supported by Novell. This could be done through students having an internship program with Novell. Novell intends to help these students in completing their curriculum project. As part of that, Novell provides all the necessary technical guidance and support to the students.

In a release GNOME 2.6 marks the fruit of six months of hard work of all the hackers, maintainers, translators, testers, usability team, and accessibility team, the GNOME community has done it again: GNOME 2.6.0 continues the high standards in the areas of usability, accessibility and internationalization that our users expect from the world's free software desktop.

The GMOME 2.6 Desktop is the latest release of the ever popular, multi-platform free desktop environment.

GNOME 2.6 desktop has several interesting new features and hundreds of bug fixes. These make day-to-day usage much easier; with noticeable improvements in the user interface, general stability and speed, and the built-in help system. You can learn more in the following sections.

GNOME provides a user-friendly environment that "just works" for everyday users, without excess complexity or obscure features. At the same time we provide the rich flexibility experienced developers demand.

The GNOME 2.6 Desktop is the latest release of the ever popular, multi-platform free desktop environment.

GNOME 2.6 desktop has several interesting new features and hundreds of the bug fixes. These make day-to-day usage much easier; with noticeable improvements in the user interface, general stability and speed, and the built-in help system.

GNOME includes powerful features such as world-class smooth text rendering, a first-class accessibility infrastructure, and a complete internationalization infrastructure that includes support for bi-directional text.

7. SCOPE FOR FUTURE DEVELOPMENT

Open source projects are becoming very popular nowadays. Open source projects entirely depend upon one's interest. If people continue to show more interest these projects have a good scope.

Speaking of GNOME, GNOME is being used a lot these days. With hundreds of bug fixes and several enhancements the open source communities have made it a big success. With more efforts each and every person will know about the GNOME desktop in the forth-coming years.

GNOME operates on time-based release philosophy, an attempt to continuously provide the best of the developers' efforts to users as quickly as possible. Six months after GNOME 2.8 will feature integrated email, contacts, calendar and multimedia functionality, and further advances in accessibility, usability and internationalization.

There will be several more bug fixes and many more enhancements which will make GNOME more user friendly. Within a few more months a bug free GNOME will come into existence.

8. REFERENCES

References: -

http://gnomebangalore.org/

http://gnome.org/

http://gnome.org/about/

-> About gnome

http://www.gnome.org/bounties/

->Desktop integration boundaries

http://gnome.org/start/2.6/notes/

->Release notes on GNOME

http://gtk.org/

->About GTK+

http://gtk.org/tutorial/sec-messagedialog.html

->About GTK+ message dialog

http://gtk.php.net/apps/rate.php?appid=234&rate=2&thecat=100&thesubcat=&offset

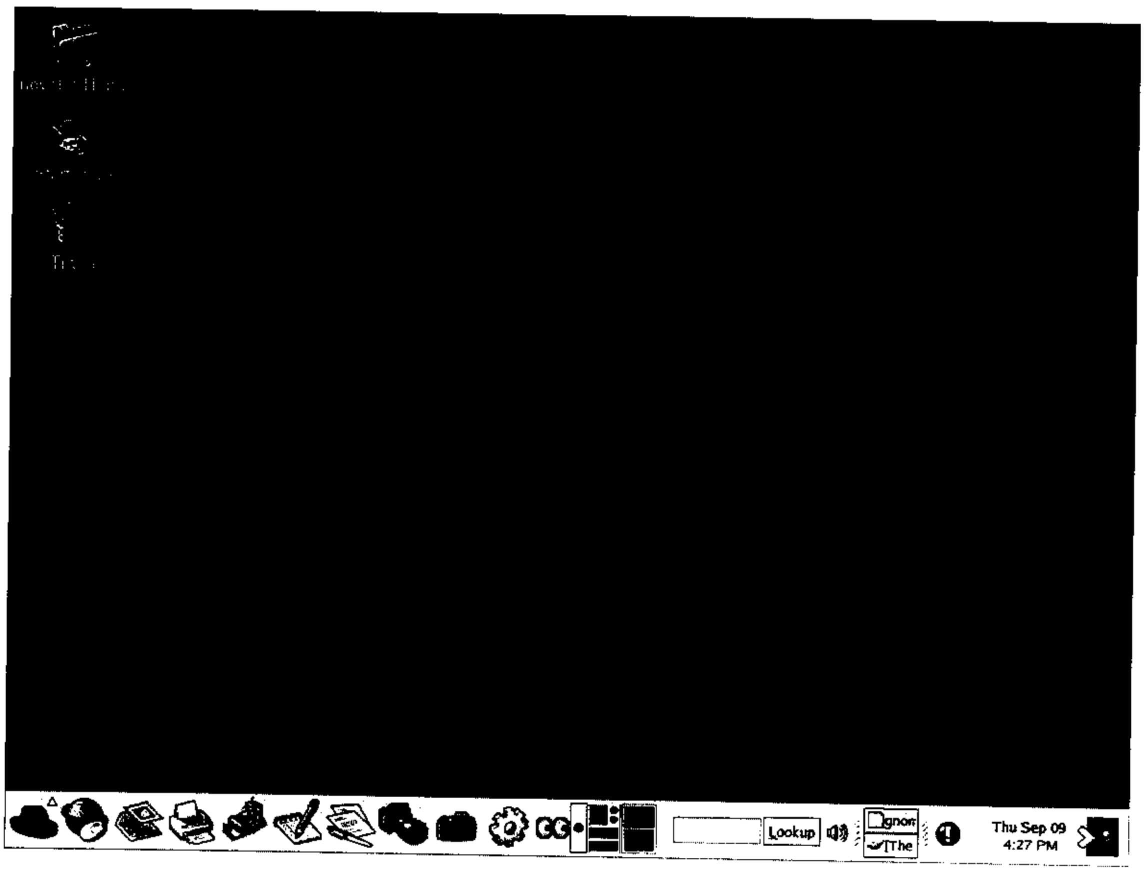
->About GTK+ applications

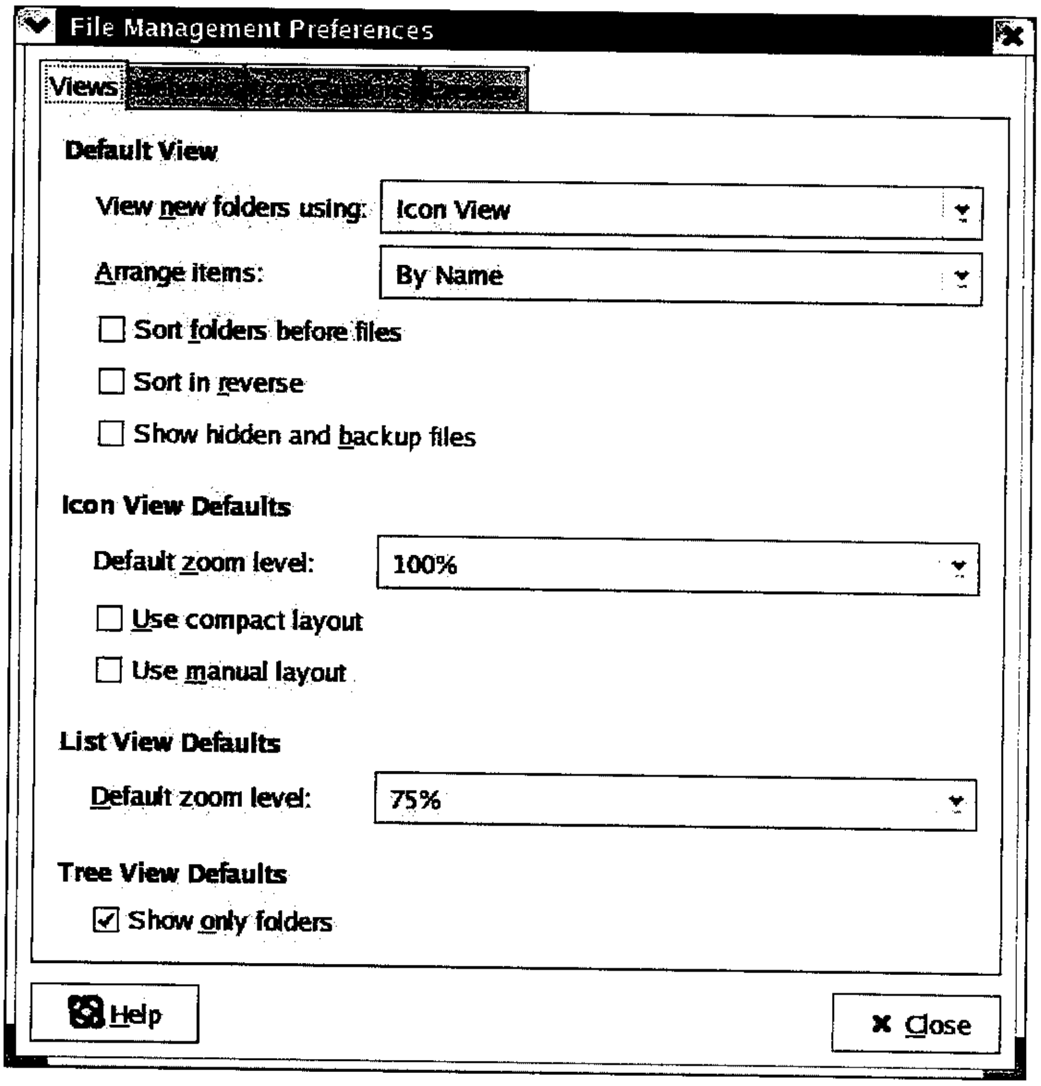
http://www.gnomebangalore.org/?q=node/view/45

->NOSIP

9. APPENDIX

9.2 SAMPLE SCREENS




File Management Preferences

Views

Default View

View new folders using: Icon View

Arrange items: By Name

☐ Sort folders before files

☐ Sort in reverse

☐ Show hidden and backup files

Icon View Defaults

Default zoom level: 100%

☐ Use compact layout

☐ Use manual layout

List View Defaults

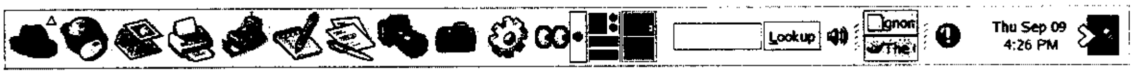
Default zoom level: 75%

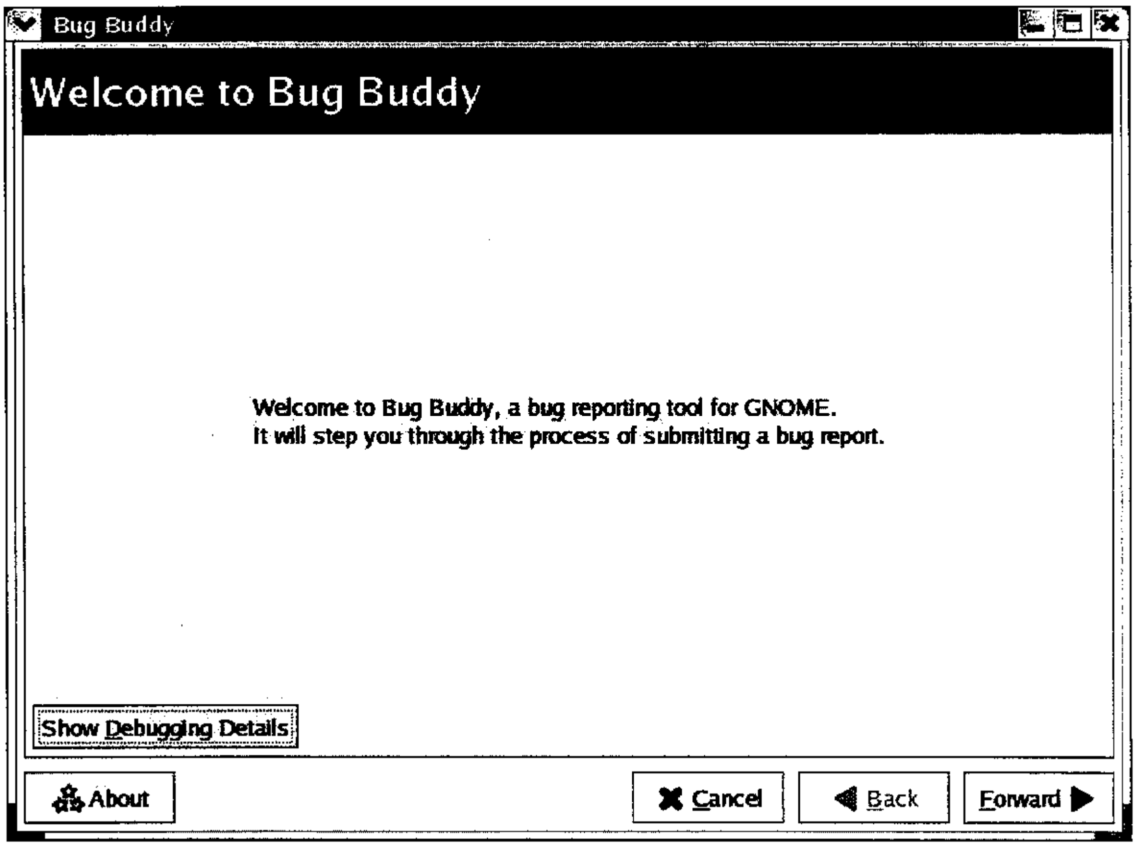
Tree View Defaults

☑ Show only folders

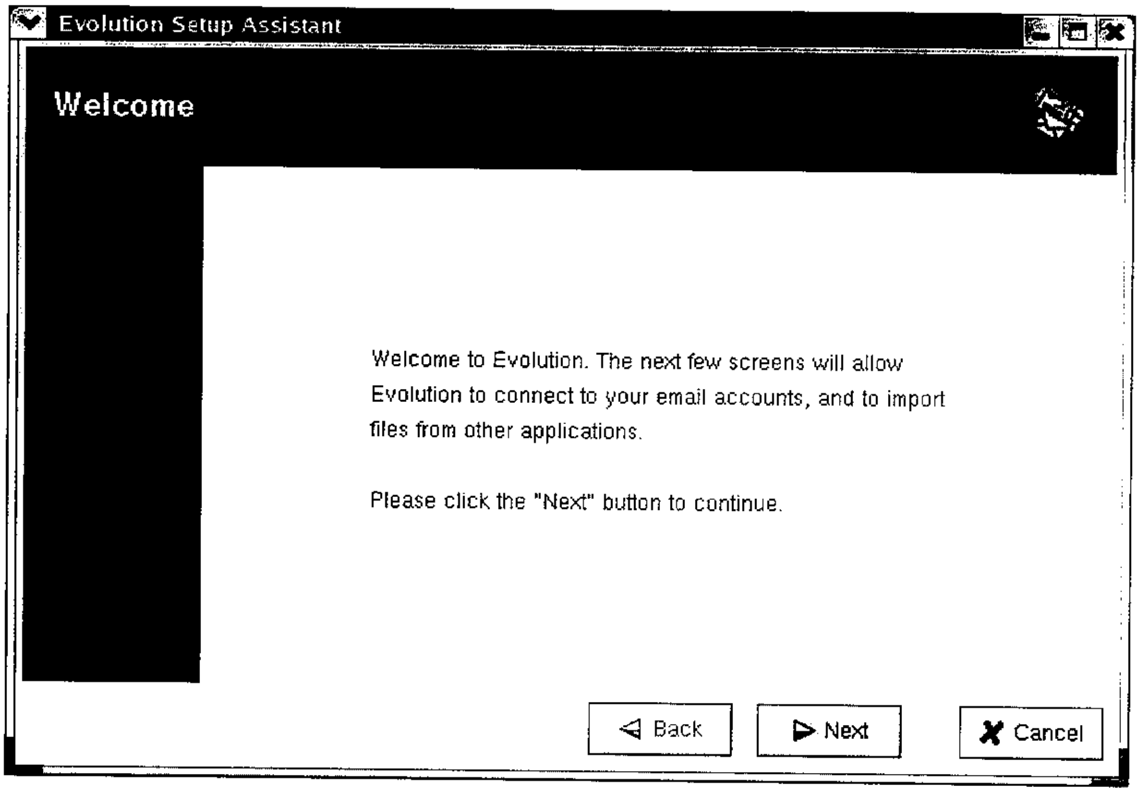
Help

Close


Lookup
Ignore
Thu Sep 09
4:26 PM


Welcome to Bug Buddy

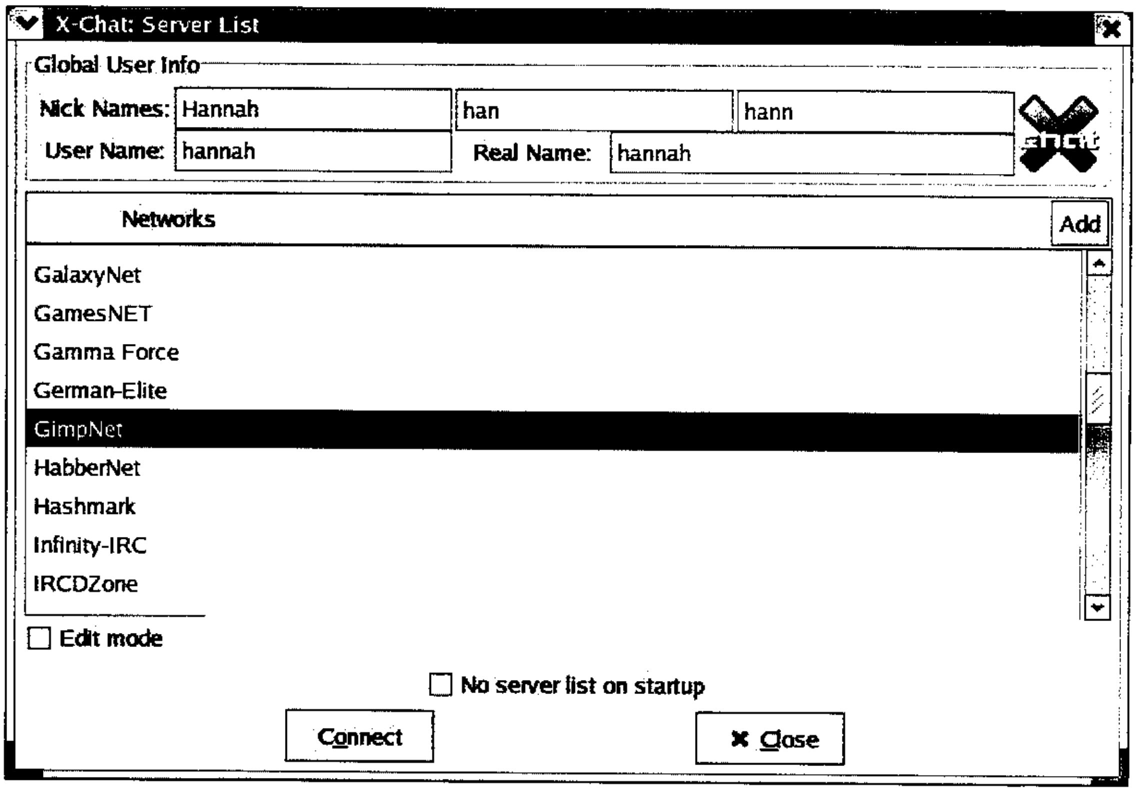
Welcome to Bug Buddy, a bug reporting tool for GNOME.
It will step you through the process of submitting a bug report.



Welcome

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.


X-Chat: Server List

Global User Info

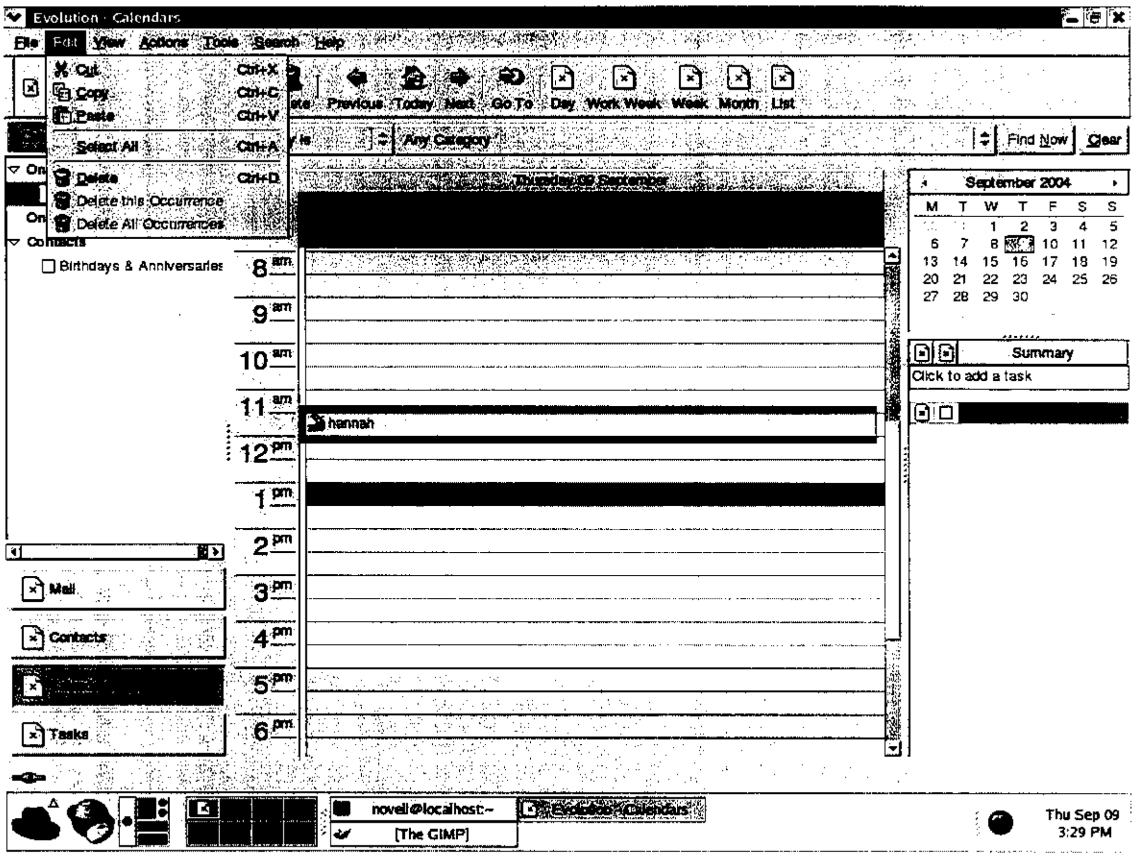
Nick Names:	Hannah	han	hann
User Name:	hannah	Real Name:	hannah

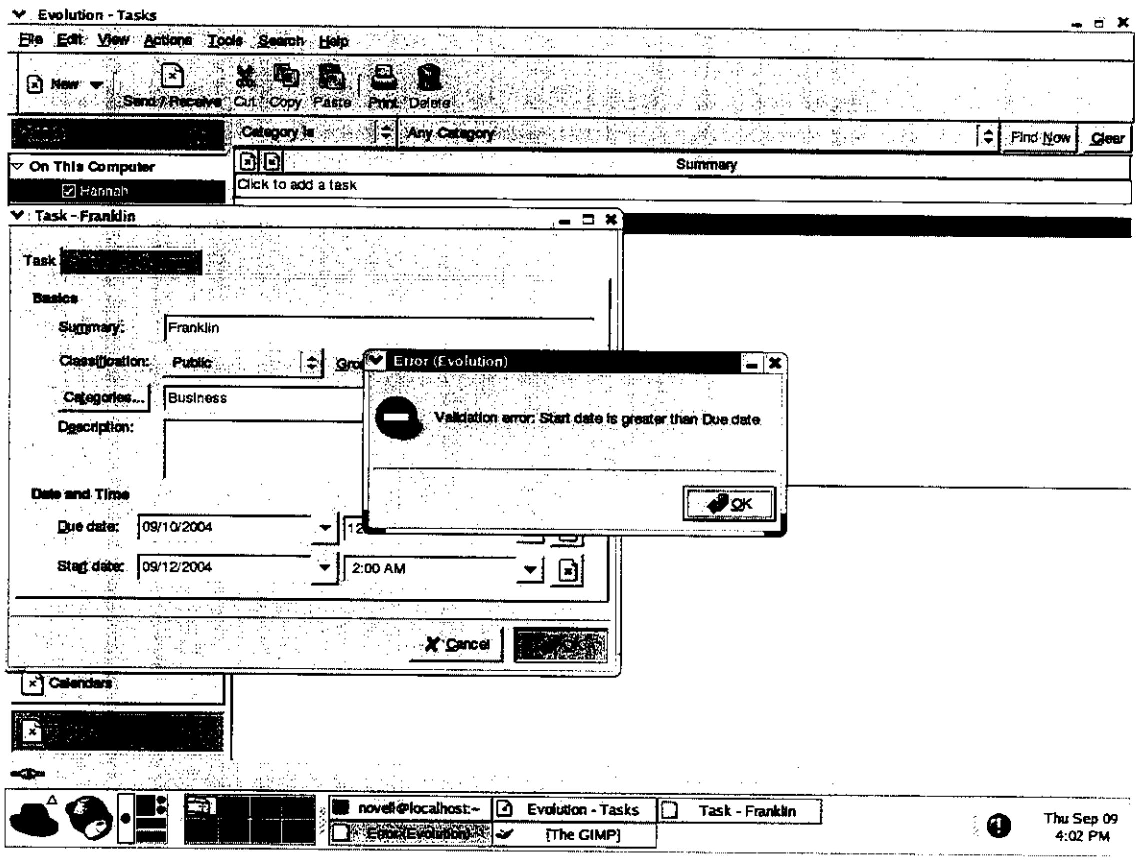
Networks

GalaxyNet
GamesNET
Gamma Force
German-Elite
GimpNet
HabberNet
Hashmark
Infinity-IRC
IRCDZone

☐ Edit mode

☐ No server list on startup


Evolution - Calendars
File
Edit
View
Actions
Tools
Search
Help
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Select All Ctrl+A
Delete Ctrl+D
Delete this Occurrence
Delete All Occurrences
Previous
Today
Next
Go To
Day
Work Week
Week
Month
List
Any Category
Find Now
Clear
September 2004
M T W T F S S
Contacts
Birthdays & Anniversaries
hannah
Summary
Click to add a task
Mail
Contacts
Tasks
novell@localhost:~
[The GIMP]
Evolution - Calendars
Thu Sep 09
3:29 PM



Evolution - Tasks

File Edit View Actions Tools Search Help

New | Send / Receive | Cut | Copy | Paste | Print | Delete

Category is | Any Category | Find Now | Clear

On This Computer

Hannah

Summary

Click to add a task

Task - Franklin

Task

Basics

Summary: Franklin

Classification: Public

Categories... Business

Description:

Date and Time

Due date: 09/10/2004

Start date: 09/12/2004 2:00 AM

Cancel

Calendars

Error (Evolution)

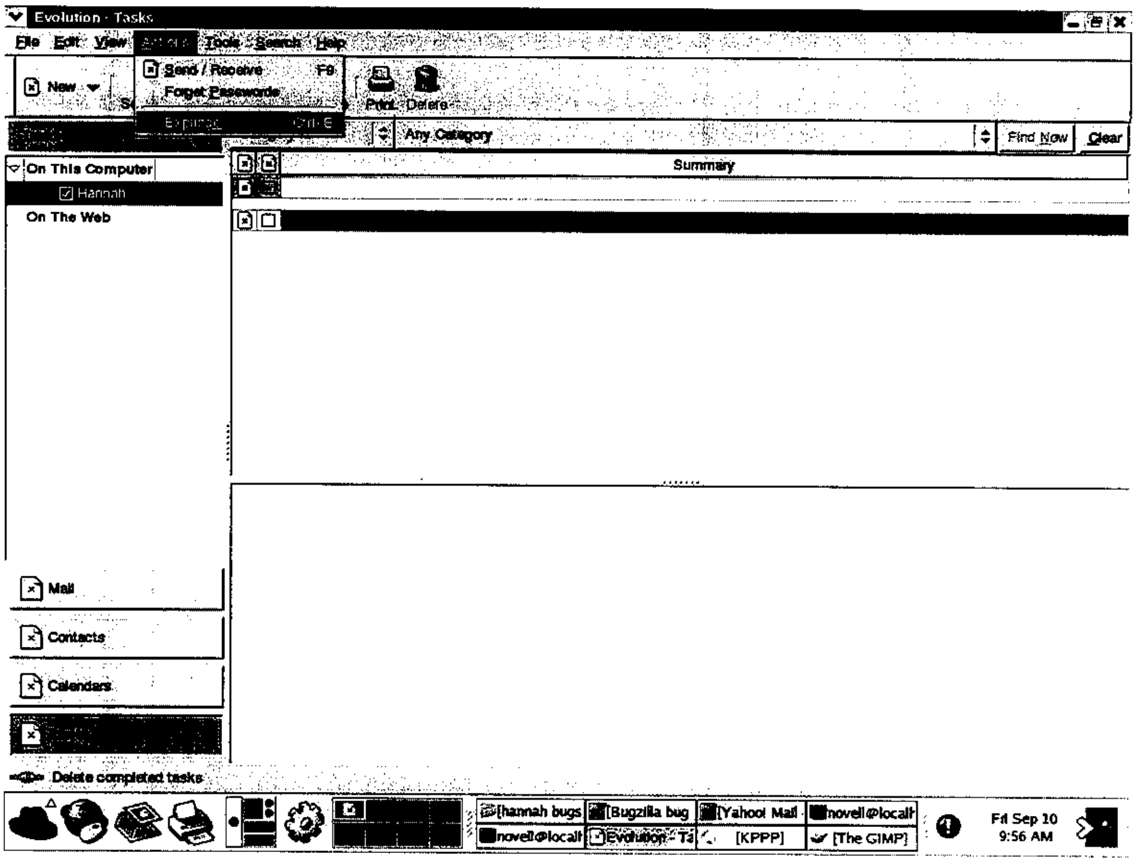
Validation error: Start date is greater than Due date

OK

novell@localhost:~ | Evolution - Tasks | Task - Franklin

Error (Evolution) | [The GIMP]

Thu Sep 09
4:02 PM


Evolution - Tasks
File
Edit
View
Actions
Tools
Search
Help
Send / Receive F9
Forget Passwords
Expunge Ctrl-E
New
Print
Delete
Any Category
Find Now
Clear
Summary
On This Computer
Hannah
On The Web
Mail
Contacts
Calendars
Delete completed tasks
[hannah bugs
[Bugzilla bug
[Yahoo! Mail
novell@localh
novell@localh
Evolution - Ta
[KPPP]
[The GIMP]
Fri Sep 10
9:56 AM

9.3LIST 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.

9.2 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