SOFTWARE PROCESS IMPROVEMENT AND PRACTICE
Softw. Process Improve. Pract. 2003; **8**: 201–215 (DOI: 10.1002/spip.189)

The GNOME Project: a Case Study of Open Source, Global Software Development

Research Section

Daniel M. German*,†
Software Engineering Group Department of Computer Science University of Victoria Victoria BC, Canada

Many successful free/open source software (FOSS) projects start with the premise that their contributors are rarely colocated, and as a consequence, these projects are cases of global software development (GSD). This article describes how the GNOME Project, a large FOSS project, has tried to overcome the disadvantages of GSD. The main goal of GNOME is to create a GUI desktop for Unix systems, and encompasses close to two million lines of code. More than 500 individuals (distributed across the world) have contributed to the project. This article also describes the software development methods and practices used by the members of the project, and its organizational structure. The article ends by proposing a list of practices that could benefit other global software development projects, both FOSS and commercial. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: software engineering; global software development; open source software; empirical study; management of software projects

1. INTRODUCTION

The three most important factors that differentiate Global Software Development (GSD) from traditional software development are distance, time differences, and cultural differences (Carmel 1999). Several reasons are credited for the growing popularity of GSD, such as the premise of cheaper cost of production, proximity to customers, a larger labor pool, and the natural globalization of software companies, result of their growth, mergers and acquisitions (Carmel and Agarwal 2001, Hersleb and Moitra 2001). At the same time, GSD is considered to be a difficult endeavor. As Hersleb and Moitra stated, 'there is strong evidence [...] that multi-site development tasks take much longer than comparable colocated tasks' (Hersleb and Moitra 2001).

Free and Open Source Software (FOSS) engineering projects have been of particular interest to the GSD research community because they start with the premise that developers are rarely colocated. While it is true that many of these projects are composed of a handful of individuals, several large, successful projects involve a significant number of contributors, ranging from software architects and engineers to software testers and documentation authors. These contributors are often located in different countries. These teams display the three factors that Carmel attributed to GSD: they are separated by a large distance, work in different time zones, and have cultural differences. For example,

* Correspondence to: Daniel M. German, Software Engineering Group, Department of Computer Science, University of Victoria, Victoria BC, Canada
† E-mail: dmgerman@uvic.ca

the MAINTAINERS file in the version 2.4.22 of the Linux kernel contains email addresses from domains corresponding to countries in the Americas, Europe, and Australasia.

An understanding of the FOSS development process can potentially lead towards potential advances in GSD. The summary of the 2nd International Workshop on Global Software Development (Damian 2002) states the need to research how open source development teams deal with issues of trust and personal interrelationships; Elliot and Scacchi have stated that 'studies are needed to understand how people in these (FOSS) projects work together to coordinate software development at a distance and what social worlds arise to assist in this collaboration' (Scacchi 2004).

The FOSS movement has demonstrated that it can create high-quality software. The most distinctive characteristic of FOSS projects is its license agreement, which gives its users certain rights such as the ability to run the software, inspect its source code, modify it and redistribute it, without having to pay for that right. Successful FOSS projects tend to be created by communities that are globally distributed, and there is currently no accepted framework that defines how FOSS is or should be developed in practice (Scacchi 2004). Instead, every project tends to have its own framework, created as a consequence of its developers' experience and the adoption of practices that seem to be successful in other projects.

1.1. Research Questions

This article looks in detail into the organization and software engineering practices followed by the GNOME project, in an attempt to understand how its contributors make it a success of GSD. GNOME, the GNU Network Object Model Environment (gnome.org), is a free (as defined by the General Public License, and therefore an open source) desktop environment for Unix systems. One particular feature of GNOME is that it is being created by hundreds of contributors, making it one of the largest FOSS projects. The objective of this research is to answer the following questions: how do GNOME contributors organize themselves? What are the processes they follow to overcome their dislocation? What type of communication do they use? How do they interact? Are there any practices that they follow that could be used by other teams developing software, both FOSS and commercial?

1.2. Methodology

The author became a contributor to GNOME in 1999, and was the maintainer of one of its modules (ggv, a postscript viewer) for over one year. He contributed patches to this module until 2001 and he has been a member of the GNOME Foundation (described in Section 3.2) since its conception. This article is the result of what an anthropologist would call an immersion study. This study was supplemented by the analysis of the logs of its CVS (Concurrent Versions System) repository and the archives of its 104 mailing lists (German in press, German and Mockus 2003).

2. GNOME

GNOME was started in 1996 by Miguel de Icaza in order to create a free collection of libraries and applications that could make Linux a viable alternative in the desktop. GNOME is composed of three main components:

- An easy-to-use GUI environment.
- An 'office suite' of applications, which includes a spreadsheet, a word processor, and a large set of simpler applications.
- A collection of tools, libraries, and components to develop this environment and its application.

The first version (0.0) was posted in August 1997 and provided only one simple application and a set of libraries. Version 1.0 was released in March 1999, a point at which it was integrated into Red Hat Linux as its default desktop (and continues to be part of it). Version 2.4 is the latest stable version, released in September 2003, and its current official distribution (Charles 1999, de Icaza 2002). GNOME currently comprises almost two million lines of code.

A stable, easy-to-use desktop suite is seen as a requirement for Unix to become a viable alternative to Microsoft or Apple operating systems (Charles 1999, Shirky 2000). As a consequence, many Unix companies are particularly interested in the success of GNOME. GNOME 1.0 was the first version included in Red Hat Linux, and this decision acknowledged GNOME as a strategic collection of

applications for Red Hat's goal to become a desktop operating system. Red Hat set up Red Hat Advance Development Labs to assist in the development of GNOME. Since 1999, Red Hat has dedicated several developers and documenters to the GNOME project (de Icaza 2002).

At the end of 1999, two companies–Eazel and Ximian–were established to continue the development of GNOME components and to sell services around it. Eazel concentrated on the development of the file manager (Nautilus), while Ximian concentrated on the development of a groupware, e-mail tool called Evolution, and on the release of a version of GNOME and its corresponding commercial support. Owing to lack of additional funding, Eazel folded early in 2001, while Ximian (a private company that was created by the founders of the GNOME Project, including Miguel de Icaza), continues to operate and currently has four seats in the board of directors of the GNOME Foundation. Ximian has recruited some of the most active volunteer contributors to the project. One of Ximian's main products are a shrink-wrapped version of GNOME called 'Ximian Desktop', and a subscription-based service for the automatic updating of software in the client's machine. Ximian was bought by Novell in August 2003.

In 2000, Sun Microsystems decided to adopt GNOME as its desktop software for Solaris (replacing the Common Desktop Environment–CDE). Sun has established the 'Sun GNOME Accessibility Development Lab', and its paid employees (who work alongside members of the GNOME community) help define and drive accessibility support in GNOME, as well as contribute writing code. Sun currently supports GNOME under Solaris and plans to include it in future releases of the operating system (Sun Microsystems 2003).

3. ORGANIZATION

In order to handle a project of this magnitude, the code base is divided into *modules*. There are four main groups of modules: (a) required libraries (such as the libraries for GUI, printing, XML processing, CORBA); (b) core applications (such as its applets, an editor, a windows manager, and a configuration tool); (c) applications (such as a spreadsheet, a mail client, a word processor), and (d) others (several dozen modules and growing, these modules represent individual applications that are not considered part of the core of GNOME).

GNOME is not a monolithic application. Rather, it is a large collection of libraries and applications that work and evolve together. There is no single GNOME maintainer who decides the present and future of the project. Instead, each module has its own maintainers, set of developers, development timeline, and objectives. Some modules might have an unofficial industrial sponsor, who employs the core contributors and maintainers of the module to do their work.

Each module has one or more maintainers, who oversee the development of their corresponding module and coordinate and integrate the contributions of other developers to their module (German 2002). These modules are interrelated between themselves, but their relationships are kept to the minimum, so each module can evolve as independently as possible from the rest. The official distribution of version 2.4 contains 76 different modules. This distribution includes only the basic infrastructure of GNOME (modules of type a and b as described above).

Module maintainers serve the roles of leaders for their module. Lerner and Triole (2000) identified the main roles of a leader in an FOSS project as:

- Providing a vision;
- Dividing the project into parts in which individuals can tackle independent tasks;
- Attracting developers to the project;
- Keeping the project together and prevent forking.

The success of an FOSS project is dependent on the ability of its maintainers to divide the project into small parts in which the developers can work with minimal communication between each other and with minimal impact to the work of others (Lerner and Triole 2000). GNOME has been able to attract and maintain good, trustworthy maintainers in its most important modules, and many are employees of companies who sponsored the project.

3.1. Management and Direction

Although not as famous as Linus Torvalds, Miguel de Icaza has been recognized as an important individual in the open source movement. The magazine *Technology Review* selected him as one of the innovators of 1999: 'De Icaza was chosen both for

his accomplishments in the GNOME Project and as a representative of the open-source software movement, which embodies a creative new mode of innovation: a large-scale collaboration over the Internet. People like Miguel are the future of technology' (Linus Torvalds was also included in this list) (The Technology Review 1999). Before starting GNOME, de Icaza was a well-known personality in the open source movement, and this is exemplified by the fact that he is known on a first name basis as Miguel in these circles. This is one of the factors that allowed him to start GNOME and create a critical mass around it.

For many years, GNOME used to be run by a 'legislature' where each of its contributors had a voice, and a vote and the developer's mailing list (developers) was the 'floor' where the issues were discussed. Miguel de Icaza, its founder, served as the 'constitutional monarch' and 'supreme court' of the project, and had the final say on any unsolvable disputes. This model is very similar to the one used by Perl during its first years.

In October of 1999, Miguel de Icaza, along with Nat Freeman, another of the core developers of the GNOME project created Helixcode (now Ximian, and recently bought by Novell), a commercial venture aimed at continuing the development of GNOME, and planning to generate income by selling services around it.

The perception of many contributors was that companies such as Red Hat (that already controlled the development of several of the critical low-level libraries of the project) and Ximian (which had taken over the development of the mail client of the project) could hijack the project and steer it away from its original goals. During 2000, and after months of debate in the mailing lists, the GNOME community decided to create a foundation, similar to the *Apache Software Foundation* (its goal is to oversee the present and future of the Apache project) to take over the direction of GNOME and guarantee that it would continue with its original goals.

3.2. The GNOME Foundation

The mandate of the Foundation is 'to further the goal of the GNOME Project: to create a computing platform for use by the general public that is completely free software'. (The Gnome Foundation 2000).

The Foundation is composed of four entities:

1. Its members. Any contributor to the project can apply for membership. A contributor is defined as 'any individual who has contributed to a non-trivial improvement of the GNOME Project, such as code, documentation, translations, maintenance of project-wide resources, or other non-trivial activities which benefit the GNOME Project. Large amounts of advocacy or bug reporting may qualify one as a contributor, provided that such contributions are significantly above the level expected of an ordinary user' (The Gnome Foundation 2002). The term of the membership is two years, and it is renewable, thus guaranteeing that the Foundation is populated with current contributors.
2. The Board of Directors. The Board of Directors is composed of 11 members, who are expected to be Foundation members and who are democratically elected by the rest of the Foundation members. Not more than four members can belong to the same corporation or organization and this rule has been already enforced several times in the past. Furthermore, the members of the Board are supposed to serve in a personal capacity and not as representatives of their employers. The Board is responsible for making sure that GNOME fulfills its expected goals. As defined by the Foundation's charter, the Board is the primary decision-making body of the GNOME Foundation. The Board meets regularly (via telephone call) to discuss the current issues and take decisions on behalf of the entire community. The minutes of each meeting are then published in the foundation-announce mailing list. Board members are expected to contribute a significant part of their time to the Project.
3. The Advisory Board (composed of corporate and non-profit organizations). Some of the corporate members of the Advisory Board are IBM, MandrakeSoft, Red Hat, and Sun. The Debian Project and the Free Software Foundation are its only two non-for-profit members. Corporate members are expected to pay $10,000 in annual dues, while there are no membership fees for non-for-profit organizations.
4. The executive director, who is a paid employee of the Foundation, is responsible for 'managing

and growing the GNOME foundation as an organization'.

The Foundation fulfills the following four roles (Mueth and Pennington 2002):

1. It provides a democratic process in which the entire GNOME development community can have a voice. The Board is an elected representative of the community and is expected to act on its behalf and best interest.
2. It guarantees that the decisions on the future of GNOME are done in an open and transparent way.
3. It is responsible for communicating information to the media and corporations.
4. It is a legal entity that can accept donations and make purchases to benefit GNOME.

3.3. Contributors

GNOME makes a strong point that coders are not the only contributors to a project. Anybody with a significant contribution can achieve a significant position in the project (ultimately becoming a member of the Board of Directors). The number of active GNOME contributors is difficult to determine. Frequently, a developer might contribute actively during some periods and not at all during others. Furthermore, some contributors might contribute bug reports or patches directly to a developer. By October 2003, the CVS repository had recorded contributions from 569 different user Ids, and the Foundation was composed of 320 members.

3.3.1. The Paid Employees

As we described in German (2002), several companies have been subsidizing the development of GNOME. Red Hat, Sun Microsystems, and Ximian are some of the companies that pay full-time employees to work on GNOME. Paid employees are usually responsible for the following tasks: project design and coordination, testing, documentation, and bug fixing. These tasks are usually less attractive to volunteers. By taking care of these tasks, the paid employees make sure that the development of GNOME continues at a steady pace. Some paid employees also take responsibility (as maintainers) for some of the critical parts of the project, such as gtk+ and ORBit (Red Hat), the file manager Nautilus (Eazel, now bankrupt), Evolution (Ximian), etc. Paid employees contribute not only in the form of code. One of the most visible contributions of Sun employees is the proposal of the GNOME Accessibility Framework, whose goals are to guarantee that GNOME can be used by a vast variety of users, including persons with disabilities. In September of 2002, Sun received the 'Helen Keller Achievement Award' from the American Foundation of the Blind for its GNOME Accessibility Framework (Sun Microsystems 2002).

3.3.2. Volunteers

Volunteers still play a very important role in the project and their contributions are everywhere: as maintainers and contributors to modules, as bug hunters, as documenters, as beta testers, etc. In particular, there is one area of GNOME that is done mainly by volunteers – internationalization. The translation of GNOME is done by small teams of volunteers (volunteers who usually speak the language in question and who are interested in seeing support for their language in GNOME). The volunteers spread over a large spectrum. Some are students, some are software professionals, and some are academics (for example, Dr. Andreas L. Guelzow, Professor at the Department of Mathematical and Computing Sciences, Concordia University College of Alberta is one of the two maintainers of gnumeric). Most of the paid developers in GNOME were, at some point, volunteers. Essentially for the volunteers, their hobby became their job.

3.3.3. Nonprogrammers

As with any other open source project, GNOME is a meritocracy, where people are valued by the quality (and quantity) of their contributions. Several contributors have reached a high status within the project, even if they are not programmers. The best-known examples are Telsa Gwynne and Tuomas Kuosmanen (better known as TigerT). Ms. Gwynne's main contributions have been in documentation; in 2001, she was elected to the Board of Directors. TigerT has designed many of the graphical elements of GNOME, including its logo, and is an important contributor to the GIMP project too.

3.3.4. Attracting New Contributors

One of the challenges of mature FOSS projects is to attract new contributors who can help the current developers, and in some cases, replace them when

they cannot continue working on it. As the project grows, the cost of entry is increased. GNOME, acknowledging this, has attempted several ways to attract people.

- Bug fixing. Like industry, bug fixing is perceived as a way for the newcomer to get acquainted with the source code and the architecture of the project.
- TODO lists. Every module is expected to have a TODO file that lists activities that need to be done.
- GNOME-love mailing list. Created in March of 2001, the description of the mailing list reads 'In the GNOME Love mailing list you will find a helping hand that will help you get up to speed in contributing at any level to this exciting project. [...] Consider GNOME Love to be a place where you will learn new technologies, where you will learn how to work with other contributors and enjoy the passion of software development.' Potential contributors are encouraged to post a message where they state their interests, credentials and request a 'task'. The mailing list coordinators (or other maintainers) will then propose an activity.

3.4. Committees

Given the lack of a single company driving the development according to its business goals, FOSS projects tend to rely on volunteers to do most of the administrative tasks of those projects. The Foundation is responsible for organizing committees to complete tasks the Foundation identifies as important. These committees are headed by Board members. Contributors then volunteer to be members of these committees. Examples of committees are:

- The foundation membership committee is responsible for maintaining the membership list of the Foundation, processing requests for membership, and the annual elections for the new Board of Directors.
- The fund-raising committee is dedicated to find ways to promote donations from individuals and organizations.
- The sysadmin committee is responsible for maintaining the infrastructure of the project, which includes making sure that the machines and software are running as expected.
- The GUADEC committee organizes an annual conference in an effort to get contributors together. This committee is responsible for the entire organization of the conference: requesting proposals for venues, selecting a venue, organizing a program committee, organizing the actual event, etc.
- The Release Team's responsibility is the planning and the coordination of the overall project. Its job is to find a proper schedule for the 'next' release and to make sure that the project actually follows the schedule.

3.5. The Release Team

Because GNOME is a collection of several different modules, each has its own set of core contributors, who might have very different time commitments to the project. Furthermore, each module has its one release schedule. The organization of a release schedule for GNOME is not easy. This task is the main concern of the Release Team that is responsible for the planning and coordination of the overall project. They are responsible for developing, in coordination with the module maintainers, release schedules for each of the different modules, and the schedule of the overall project. They also keep track of the development, making sure that everything stays within schedule. Jeff Waugh, a GNOME Foundation member, summarized the accomplishment of the team and the skills required (in his message of candidacy to the Board of Directors in 2002): '*[The Release Team] has earned the trust of the GNOME developer community, madly hand-waved the GNOME 2.0 project back on track, and brought strong co-operation and 'the love' back to the project after a short hiatus. It has required an interesting combination of skills, from cheer-leading and Maciej-style police brutality to subtle diplomacy and 'networking''*. (Waugh 2002)

A release schedule is composed of the following:

- Regular test release dates, approximately every 2 weeks. The project uses a numbering system similar to Linux. For example, versions 2.1 and and 2.3 are test releases, while 2.0, 2.2 and 2.4 are stable releases. Table 1 lists the main versions of the project and the month they were released.
- Dates for:
  - Feature freeze, after which no more features will be allowed to a given module.

- API freeze, after which no further changes to the APIs can be made.
- User Interface (UI) freeze is the deadline for changes in the UI, and after this date only fixes to major UI bugs are allowed. This allows documenters to start their work without fear of major changes to the UI.
- UI Review and String freeze, after which no UI, or string changes are allowed without confirmation by the Release Team. This allows the internationalization to start (translation of the UI).
- Hard code freeze, which avoids a last minute accident that could risk the stability of a module or the project.

A natural question to ask is 'is the release schedule useful?'. Given that the web site uses CVS for its own version control, it was possible to retrieve the history of changes in the schedule for its 2.4 release. The final schedule for this version was decided on April 20, 2003, when September 10, 2003 was decided as the release date for version 2.4. The schedule was not modified since that date, except for the 'Hard Code Freeze' date, which was delayed by 2 days. Table 2 shows the schedules as defined in April 20 and its final version:

As it was planned, version 2.4 was released on Sept. 10, 2003. The Release Team scheduled that date 5 months in advance, and delivered on time.

Table 1. Main milestones of the GNOME project

Version 0.0	Aug. 1997
Version 1.0	March 1999
Version 1.2	May 2000
Version 1.4	April 2001
Version 2.0	June 2002
Version 2.2	Feb. 2003
Version 2.4	Sept. 2003

Table 2. Release schedule for version 2.4

Proposed	Final	Event
June 9	*same*	Feature and API/ABI freeze
July 7	*same*	UI freeze
August 4	*same*	Strings and Hard UI freeze
August 23	August 25	Hard Code Freeze
September 1	*same*	Release Candidate tarballs due
September 3	*same*	Release Candidate release
September 8	*same*	2.4.0 final tarballs due
September 10	*same*	2.4.0 final release

This is a remarkable feat, especially considering that it involved the synchronization of 72 different modules with no central management authority other than the GNOME Foundation.

4. INFRASTRUCTURE

One of the main requirements for GSD is agreement on a common toolkit, such that each member of the team will have access to it. In a private development, this is not an issue, as it is expected that the organization will provide the necessary software and hardware infrastructure.

This infrastructure required by GNOME can be divided in two main categories: (a) contributor level, which is the hardware and software necessary for a given individual to collaborate with the project; and (b) community level, which is used by all the collaborators to communicate and share their progress.

Given the philosophy behind FOSS and because volunteers have always been perceived as the critical driving force in FOSS, the toolkit of choice is usually FOSS itself (one notable exception to this rule is the use of *bitkeeper* a commercial product for configuration management used by the Linux project, free to be used by Linux developers, with certain restrictions).

GNOME uses, like many FOSS projects, the GNU toolkit (gee, make, autoconf, automake, emacs, vi, etc.), CVS for software configuration management, Bugzilla for bug management, GNU Mailman for its mailing lists, and of course, because its goal is to provide a desktop for Unix, it uses Unix as its development platform. Potential developers, by just installing a recent version of Linux, have an environment that allows them to start contributing to the project. The cost of entry is therefore minimized.

During its conception, GNOME faced a bootstrapping problem: it required a GUI toolkit, but none was available (with a compatible license). In an example of the second principle of the Cathedral and the Bazaar, '*Good programmers know what to write. Great ones know what to rewrite (and reuse)*' (Raymond 1999) GNOME proceeded to reuse the X11 widgets developed for the GNU Image Manipulation Program (GIMP). This library is known as the gtk+ GIMP Toolkit, and GNOME became its maintainer and continues to improve it. GNOME has continued to create libraries and infrastructure

that are needed to fulfill its goals, for example, a UI generator (glade) and a printing library (gnome-print).

At the level of community infrastructure, the project has usually relied upon donation of bandwidth and servers by third parties. Universities have been a major source of resources in this category (frequently without the administration being aware of it). When he started GNOME, de Icaza was a system administrator working for the National Autonomous University of Mexico (UNAM). At that time, the UNAM provided the basic communication infrastructure for the project: a CVS repository, a ftp server and a mail server. Currently, many different organizations provide servers and mirrors that help GNOME. For example, the Academic Computer Club at the Umeå University in Sweden provides the main ftp server for GNOME.

5. REQUIREMENTS

As described in (Scacchi 2002), most FOSS projects do not have a traditional requirement's engineering phase. Specially at the beginning of GNOME, the only stakeholders were the developers who acted as users, investors, coders, testers, documenters, etc., with little interest in the commercial success of the project, but who, at the same time, wanted to achieve respect from their peers for their development abilities. One of the main goals of the developers is to produce software that is used by its associated community.

In particular, we can identify the following sources of requirements in GNOME:

- Vision. One or several leaders provide a list of requirements that the system should satisfy. In GNOME, this is epitomized by the following non-functional requirement: 'GNOME should be completely free software' (free as defined by the Free Software Foundation).
- Reference Applications. Many of its components are created with the goal of replacing similar applications. The GNOME components should have most if not the same functionality as these reference applications. For example, gnumeric uses Microsoft Excel as its reference, ggv uses gv and kghostview, Evolution uses Microsoft Outlook and Lotus Notes.
- Asserted Requirements. In a few cases, the requirements for a module or component are born from a discussion in a mailing list. In some cases, a requirement emerges from a discussion whose original intention was not to do requirement analysis. In other instances (as it is in the case of Evolution), a person posts a clear question instigating discussion on the potential requirements that a tool should have. Evolution was born when several hundred messages were created describing the requirements (functional and non-functional) that a *good* mailer should have had before coding started.
- A prototype. Many projects start with an artifact as a way to clearly state some of the requirements needed in the final application. Frequently, a developer proposes a feature, implements it, and presents it to the rest, who then decide on its value and choose to accept the prototype or scrap the idea (Hissam *et al*. 2001). GNOME started with a prototype (version 0.0) created as the starting point of the project.
- *Post-hoc* requirements. In this case, a feature in the final project is added to a module because a developer wants that feature and he or she is willing to do most of the work, from requirements to implementation and testing. This feature might be unknown by the rest of the development team until the author provides them with a patch, and a request to add the feature to the module.

Regardless of the method used, requirements are usually gathered and prioritized by the leaders of the project, the maintainer or maintainers of the module and potentially the Foundation. A maintainer has the power to decide which requirements are to be implemented and in which order. The rest of the developers could provide input and apply pressure on the maintainers to shape their decisions (as in *post-hoc* requirements). Sometimes a subset of the developers of an FOSS project might not agree with the maintainer's view, and could potentially jeopardize the project, and create what is known as a fork. So far this has not happened within GNOME.

6. COMMUNICATION

Developers are located in many different places all around the world. As described above, some are volunteers and the rest work for different organizations (and even within those organizations, they might still be located in different parts of

the world, as is the case with some Ximian developers). Communication relies on the following infrastructure:

- Mailing Lists. The GNOME project has extensively used mailing lists. These lists have a wide range of purposes: some are intended for final users, some for particular modules, some for announcements, etc. Mailing lists provide a trail of decision making for the project. The GNOME web site lists a total of 104 mailing lists. Some organizations, such as Ximian, provide mailings lists for the modules that it is responsible for (such as the Evolution and GtkHtml). Mailing lists are usually divided into discussion, users, developers, and announcement lists
- IRC: Internet Relay Chat. The 'water-cooler conversations' have been mimicked by using IRC. Developers connect to a common IRC server/channel and wait for other users to connect. The conversation is informal, with no real agenda and no permanent record.
- Web sites. The web sites of the project comprise a large amount of information, intended for every type of contributor to the project (coders, bug reporters, bug hunters, documenters, translators, etc.), for its users, and for any casual visitor.
- GUADEC, the GNOME Conference is a Foundation's effort to get developers together. Its goal is to provide a venue for discussion, interaction, and training. The Foundation attempts to support many of the developers who cannot afford their own travelling expenses. The last GUADEC lasted 5 days and took place in Dublin, Ireland.
- The GNOME Summaries. A summary is published in the GNOME mailing list at regular intervals. It usually contains the most relevant events of the period, links to new or improved documentation, news specific to different GNOME modules, a 'Hacker Activity' section, enumerating the most active modules and the most active developers in the project, and a bug-hunting section, listing the number of current bugs per module including the progress made during the period.

6.1. Conflict Resolution

Like any other social activity, from time to time conflicts between contributors arise. These conflicts are usually resolved between the different conflicting parties by finding a common point of agreement. One of the most potentially problematic issues in GNOME is the definition and evolution of its libraries. During 2001, a long and tiring flamewar erupted within the project, in which contributors were divided in terms of which features should be included in version 2.0. The core of the problem was that some module maintainers and some members of the Release Team had different ideas on what should be and what should not be included in the next release, and one of the changes involved a major architectural modification. It was clear that an official procedure to resolve these types of conflicts did not exist.

In order to avoid future confrontations, a transparent process was created to address the proposal and acceptance of modifications to these libraries. The procedure is detailed in the 'GNOME Enhancement Procedure' document draft (Pennington 2001). This document recommended the creation of the list gep-announce, and a procedure to follow to propose changes 'with wide impact, not to relatively minor changes within a module'. The procedure is as follows:

- Creation of a requirements document GEP (GNOME Enhancement Proposal). Proposals are classified as guidelines and requirements proposals and should include:
  - Name and email address of the owner(s).
  - Date the document was posted to gep-announce.
  - Length of the discussion period, at minimum 1 week.
  - Status of the GEP (Approved, Rejected, Pending).
  - A list of 'responsible owners who will approve/reject the GEP and evaluate the solution proposals.
  - A list of uncontroversial requirements (with brief rationale/explanation as required).
  - A list of controversial requirements (with comments). Requirements by default are considered uncontroversial, but if a requirement is contested by a contributor, it is moved to a 'controversial requirements' section. This section will contain comments, both by the creator of the GEP, and any other contributor.
- Posting the GEP

- Approving the GEP. The GEP is quickly approved if there is no controversy. If there is a controversy, discussion continues for 10 days, followed by a 'last call' lasting 4 days. 'After this period, the responsible maintainers must vote each controversial requirement in or out of the approved requirements list. The vote to keep each controversial requirement is simple majority. After this vote concludes, the 'controversial requirements' section should be split in half, into approved/rejected requirements. No requirements/comments should be deleted from the document, even after they are rejected; they should remain in the 'rejected section' for archival purposes' (Meeks 2001). Once the GEP has been updated to reflect the approved and rejected requirements, the responsible maintainers vote. A two-third majority is needed to approve a GEP. If there is no agreement on who should be included in the maintainers list, this issue is decided by the Board of Directors. Once a GEP is approved, it cannot be modified.

Till date there have been a total of 13 GEPs, but only 2 have been finalized (one rejected and one approved). It is interesting to note that GEP 0, which describes the GEP process, has not been formally approved nor rejected. Regardless of their status, GEPs have provided a formal and transparent procedure in which differences can be aired and solved.

6.2. GUADEC

The GNOME community organizes an annual conference called 'The GNOME User and Developer European Conference' (GUADEC). GUADEC has been successful in bringing together developers, users, and other parties interested in the project, such as policy makers. Table 3 lists the history of GUADEC. GUADEC hosts a technical papers session and tutorials on its technology.

One of the goals of GUADEC is to bring together the core developers in order that they meet face to face, a factor that helps better communication and potential conflict resolution. It also allows them to plan next releases.

Table 3. GUADEC's history

GUADEC I	March 2001	Paris, France
GUADEC II	April 2001	Copenhagen, Denmark
GUADEC III	April 2002	Seville, Spain
GUADEC IV	June 2003	Dublin, Ireland
GUADEC V	2004	Kristiansand, Norway

Part of the money raised by the Foundation (through the program Friends of GNOME) is used to help core developers pay their way to the conference. GUADEC is considered an important place in the decision-making process of GNOME, as it is one of the few opportunities in which contributors can be in the same location at the same time. In fact, the original proposal for the Foundation (the GNOME Steering Committee) was created during GUADEC I.

7. CONTRIBUTORS AND THEIR CONTRIBUTIONS

In order to understand the contributions to the GNOME project, historical data from its CVS repository was used. CVS keeps track of who modifies which file, and the corresponding delta associated with the modification. This change is known as a file revision. CVS keeps information such as who made the revision, when the actual diff of the revision was done, number of lines added, and number of lines removed. softChange (German and Mockus 2003) was used to recover the information from these logs and to enhance it. For instance, CVS does not keep track of which files are modified at the same time. softChange analyses the logs, and rebuilds these groups of files, which are then called Modification Requests (MRs). An MR is a request by a contributor to commit a group of files at the same time. The belief is that if two files are part of the same MR, it is because they are somehow interrelated. Contrary to source code releases, CVS logs provide a very fine grained view to the evolution of the project.

The logs of 62 of the modules (composing the 2.2 distribution of GNOME) were analyzed. In order to provide a more accurate view of the current group of contributors, only the year 2002 was analyzed. It was decided to further narrow the analysis and consider only contributors to the code base (who will be referred as programmers) and MRs involving C files (C is the most widely used language in GNOME, and the number of C files in these modules outnumber the files in the next language – bash – by approximately 50 times).

A total of 185 programmers were identified. Ninety-eight programmers contributed 10 or less MRs, which account for slightly less than 3% of the total MRs. The most active programmer (in terms of MRs) accounted for 7% of the total. The top 10 programmers accounted for 46% of the total MRs. Even though these numbers need to be correlated with the actual number of LOCS or defects (bugs) removed per MR, they point in the direction that a small number of developers are responsible for most of the coding of the project. When taking into account the division of the project into modules, this effect seemed more pronounced. Table 4 shows the top five programmers for some of the most actively modified modules of GNOME.

In an analysis of SourceForge projects, Krishnamurthy found that most FOSS projects are composed of a handful of developers (Krishnamurthy 2002). GNOME is one of the few projects that appears to break this rule, given that more than 500 people have 'write access' to its CVS repository. Zawinsky, a Mozilla developer, provides insight into this phenomenon: 'If you have a project that has five people who write 80% of the code, and a hundred people who have contributed bug fixes or a few hundred lines of code here and there, is that a 105-programmer project?' (as cited in (Jones 2000)).

Evolution was further analyzed. It was composed of approximately 470 kLOCS (Feb. 2003) and almost 50% of the times the source code has been modified, the change can be attributed to one of 5 developers (see Figure 1).

Only 18 contributors accounted each for more than 1% of the total MRs. The largest contributor

Table 4. Top 5 programmers of some the most active modules during 2002. The first column shows the name of the module, the second shows the total number of programmers who contributed in that year, the third shows the anonymized top 5 programmers and the proportion of their MRs with respect to the total during the year. Some programmers contributed actively to more than one module (e.g. programmer 8). In this table, only MRs that included C files are considered

Module	# Progs.	Programmer	Prop. of MRs
glib	24	1	31%
		2	18%
		3	10%
		4	10%
		5	9%
gtk+	48	1	37%
		2	12%
		5	9%
		6	8%
		7	4%
gnome-panel	49	8	42%
		9	12%
		11	6%
		12	6%
		13	6%
ORBit2	11	14	51%
		8	28%
		15	9%
		17	5%
		18	3%
gnumeric	19	20	34%
		21	23%
		22	17%
		23	12%
		24	9%

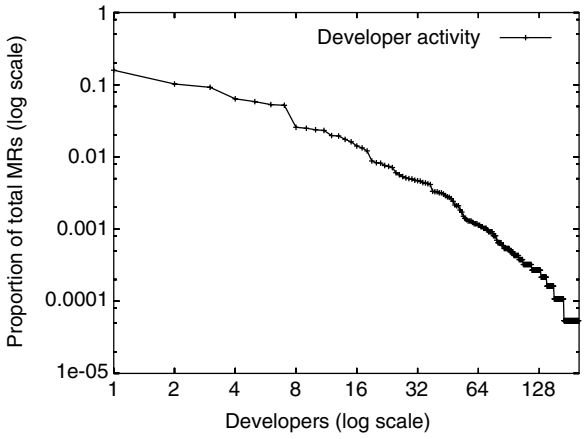

Figure 1. This plot shows the number of CVS MRs committed to Evolution per developer. Each contributor was assigned a number from 1 to 201, which corresponds to the X axis. From a total of 196 developers, 5 account for 47% of the CVS transactions, while 20 account for 81% of the MRs, and 55 have done 95% of them

Table 5. Most active programmers, as a proportion of total MRs

Programmer	Prop.	Accum.
1	0.16	0.16
2	0.10	0.26
3	0.09	0.35
4	0.06	0.42
5	0.06	0.48
6	0.05	0.53
7	0.05	0.58
8	0.03	0.61
9	0.02	0.63
10	0.02	0.65
other	0.35	1.00

is responsible for 16% of the MRs, while at the other side of the spectrum, 32 contributors had one MR only. Furthermore, a total of 48% of the MRs were contributed by only 5 contributors, while 142 contributors contributed just 5% of the MRs (80 contributed a total of 1% of the MRs).

Table 5 shows the 10 most active developers, as a proportion of all MRs. The top 10 appear to be Ximian employees or consultants (see http://primates.ximian.com/). This fact corroborates the hypothesis that private companies (such as Red Hat, Ximian, and Eazel) have had a very important effect on the development of the GNOME project (German 2002). In that respect, it is similar to the Mozilla project where core contributors were employees of Netscape (see (Mockus *et al*. 2002)).

How regularly were contributors participating in Evolution? The number of different contributors by year is depicted in Table 6. After January 2000, in any given month, there was an average of 32 contributors (8.3 stddev, minimum 15, maximum 47) per month to the project.

For a more detailed analysis of Evolution, see (German in press).

8. OBSERVATIONS

Through its history, the GNOME project has been trying to adapt to the needs of its members in order to reach its most important goal (to create free desktop environment for Unix).

The organizational structure of the project is really a combination of two structures. Many of its core developers are paid employees of companies who have a special interest in the success of GNOME. These organizations most likely have a say on what the developer does. On the other hand, within GNOME they belong to its organizational structure. The GNOME organization tends to be flat: the GNOME Foundation at the top, the maintainer of the module in the middle, and the contributors of the module at the bottom. The developer has to act, presumably, taking into account the interests of both organizations.

Table 6. Contributors to Evolution by year. It takes into account only those contributors with CVS write access

Year	Number of contributors
1998	37
1999	54
2000	95
2001	98
2002	79
2003	56

To supplement this flat organizational structure, the Foundation has created 'committees' and 'task forces' that are composed of volunteers and have specific goals. In many ways, the organizational structure of GNOME is reminiscent of the organization of the faculty in a department of a university.

Mockus *et al*. (2002) proposed two hypotheses related to the number of core developers of a project. Hypothesis #1 stated that FOSS projects will have a core of developers no larger than 10 to 15 people who control the code base, and who are responsible for around 80% of the new functionality. Hypothesis #2 reads: 'a strict code ownership policy will have to be adopted to separate the work of additional groups, creating, in effect, several related OSS projects'. In GNOME, both hypotheses are supported. GNOME has been divided into smaller projects that minimize the number of people involved that are involved in each of these subprojects (or modules). The analysis made suggests that fewer than five people are responsible for 70 to 80% of the programming effort of a given module. When a module starts to grow in complexity and the number of core developers grows too, it is either split into two different projects, or submodularized, and then one or two developers take control of each of the given submodules.

Decisions appear to be taken in an open, democratic way. The success of the project appears to depend, at least in part, in not losing its contributors. It appears that for the contributors it is important that they feel that their contributions are taken into account and that they are part of the decision process. The Foundation is the result of that need.

8.1. What can be Learnt from GNOME?

It is not possible to state what exactly makes GNOME a success in GSD. One can only create hypotheses that could be further studied, both in open source projects and commercial ones. On the basis of this study, the author believes that the following practices have the potential to benefit any GSD effort.

- Flatten the organization of the project and allow more participation in the decision-making process. Developers like to feel like they are at the core of the project, not at the bottom of the hierarchical administration.
- Use multiple types of communication. Formal and semiformal communication should include mailing lists, divided into different topics. If somebody asks a question in a mailing list, and the question is answered, then other people can learn from it too. This is particularly important for new comers to the project. Also, mailing lists should be archived and searchable because they become an important source of information. Instant messaging is very useful for one-on-one communication, but it does not replace the 'water-cooler' conversations. IRC, on the other hand, is a gathering place, where conversations among multiple individuals can occur and people can join and leave without explicit invitations. Finally, keep everybody informed by sending regular reports on the current state of development of the project.
- Treat developers as partners in the development process. The license agreement of GNOME (the General Public License) makes developers feel that the community (and by extension, them) owns the product. In traditional companies, employees are issued stock and options to make them feel part of the company and share its potential success. Depending on the field of endeavor of a given organization, an open source license might be a good motivator to the development team. A main challenge for any organization is to find well-motivated people. Many of the developers in GNOME are highly motivated to work on the project because they believe in it and its goals.
- Have a regularly scheduled colocated meeting where the main decisions can be made. This reinforces collaboration and helps during conflict resolution. This meeting should involve both social and technical activities, including presentations on what every individual does and the type of expertise they have.
- Create task forces that work aside the different teams, similar to those created in GNOME. These tasks forces would help the administration understand the current state of the project and detect potential problems faster.
- Set clear procedures and policies for conflict resolution and other potential problems.
- Modularize the product in order to minimize communication, and clearly state how the APIs and interfaces between different modules will be defined, and when they will be frozen.
- Ask developers to document their daily contributions (as part of the configuration management system and/or ChangeLogs) in order that everyone quickly knows what the others are working on.

One has to understand that the requirements of FOSS development are significantly different from commercial software development. What might work for one type of project might not work for the other, and vice-versa.

9. CONCLUSIONS AND FUTURE WORK

After almost 6 years of development, GNOME has demonstrated that it is a success. Its free software nature has created special requirements in the way that the project is organized and managed. Furthermore, GNOME is a project where people employed by different companies and volunteers work together with a common goal. Developers contribute in a wide range of ways (code, testing, bug reports, documentation, artwork, bug-hunting, and system administration) and are located across the world, relying on the ability of its leaders and maintainers to manage the project, on the Internet as its communication channel and with the use of several tools (such as mailing list, web pages, CVS, Bugzilla) to maintain a good enough communication that allows for the project to proceed. Recently, the Foundation has taken the responsibility of giving the project a coherent vision; this is to guarantee that GNOME continues to fulfill its main goal: 'to create a computing platform for use by the general public that is completely free software.'

The proposed hypotheses on how other projects can learn from the GNOME experience need to be evaluated. For example, one could ask a development team to try to use some FOSS communication and documentation methods for part of the year, and use their usual ones during another; then interview the members of the team in order to find out if there was any perceived difference. Some hypotheses (such as flattening the

organizational structure) are more difficult to test because it is difficult to, first, convince the company involved, and second, set up a control group. It will be also valuable to observe other FOSS and closed source projects, and see if their successes or failures can be correlated to these hypotheses, or to new ones.

ACKNOWLEDGEMENTS

This research was supported by the Natural Sciences and Engineering Research Council of Canada, and the Advanced Systems Institute of British Columbia. The author would like to thank the GNOME development team and especially the anonymous reviewers whose helpful and detailed comments made this a better paper.

REFERENCES

Carmel E. 1999. *Global Software Teams*. Prentice Hall: Upper Saddle River, New Jersey.

Carmel E, Agarwal R. 2001. Tactical approaches for alleviating distance in global software development. *IEEE Software* **18**(2): 22–29.

Charles J. 1999. Linux support ranges from GUI to Big Blue. *Computer* **32**(5): 20–22.

Damian D. 2002. Workshop on global software development. *ACM SIGSOFT Software Engineering Notes* **5**(7): 65–71.

de Icaza M. 2002. GNOME History. The GNOME Project, Boston, MA, http://primates.helixcode.com/~miguel/gnome-history.html.

German DM. 2002. The evolution of the GNOME Project. *Proceedings of the 2nd Workshop on Open Source Software Engineering*, The GNOME Project, Boston, MA.

German DM. Using software trails to rebuild the evolution of software. *Journal of Software Maintenance and Evolution: Research and Practice*. Wiley: New York; in press.

German DM, Mockus A. 2003. Automating the measurement of open source projects. *Proceedings of the 3rd Workshop on Open Source Software Engineering*. University College Cork: Cork Ireland, 63–68, http://opensource.ucc.ie/icse2003.

Hissam S, Weinstock C B, Plakosh D, Asundi J. 2001. Perspectives on Open-Source Software. Technical Report CMU/SEI-2001-TR-019, Software Engineering Institute, Carnegie Mellon University: Pittsburg, MA.

Hersleb JD, Moitra D. 2001. Global software development. *IEEE Software* **18**(2): 16–20.

Jones P. 2000. Brooks' law and open source: The more the merrier? Does the open source development method defy the adage about cooks in the kitchen? (IBM developer Works). IBM Corporation: White Plains, New York, ftp://www6.software.ibm.com/software/developer/library/merrier.pdf.

Krishnamurthy S. 2002. Cave or Community? An empirical examination of 100 mature open source projects. *First Monday* **7**(6) (http: //www.firstmonday.org/issues/issue7-6/krishnamurthy/index.html).

Lerner J, Triole J. 2000. *The Simple Economics of Open Source*, Working Paper 7600. National Bureau of Economic Research: Cambridge, MA, http://papers.nber.org/papers/w7600.

Meeks M. 2001. ACTION GEP 0: Process for GNOME Enhancement Proposal. The GNOME Project, Boston, MA, http://developer.gnome.org/gep/gep-0.html.

Mockus A, Fielding RT, Herbsleb J. 2002. Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology* **11**(3): 1–38.

Mueth D, Pennington H. 2002. GNOME Foundation FAQ. The GNOME Project, Boston MA, http://foundation.gnome.org/faq.html.

Pennington H. 2001. GNOME Enhancement Procedure. The GNOME Project, Boston, MA, http://ometer.com/policy.html.

Raymond E. 1999. *The Cathedral & the Bazaar*. O'Reilly & and Associates: Sebastopol, CA.

Scacchi W. 2002. Understanding the requirements for developing open source software systems. *IEEE Software* **149**(1): 24–39.

Scacchi W. 2004. Free and open source development practices in the game community. *IEEE Software* **21**(1): 59–66.

Shirky C. 2000. Linux for the End User–Phase 1. *Linux Journal* **8**(74).

Sun Microsystems 2002. Sun Microsystems is honored with the Helen Keller Achievement Award for its leadership in accessibility advancements. Press Release, http://www.sun.com/smi/Press/sunflash/2002-09/sunflash.20020924.1.html.

Sun Microsystems 2003. GNOME 2.0 FAQ. Sun Microsystems, http://www.sun.com/software/star/gnome/faq/generalfaq.html.

The GNOME Foundation 2000. GNOME Foundation Charter Draft 0.61. The GNOME Project, Boston, MA, http://foundation.gnome.org/charter.html.

The GNOME Foundation 2002. GNOME Foundation Bylaws. The GNOME Project, Boston, MA, http://foundation.gnome.org/bylaws.pdf.

The Technology Review 1999. The 1999 TR100. *Technology Review* (http://www.technologyreview.com/articles/99/11/tr1001199.asp).

Waugh J. 2002. Candidacy: Jeff Waugh. GNOME Foundation Announce Mailing List. The GNOME Project, Boston, MA, http://mail.gnome.org/archives/foundation-announce/2002-November/msg00008.html.