Increase in efficiency of free software projects through information management

Robert Schuster
Working group Software Engineering
Freie Universität Berlin

2005-05-17

Abstract

Free and Open Source Software is an enduring undertaking. However this does not mean that this development model has no flaws. This paper will present a problem of the F/OSS development process that results from insufficient care for information management. I will outline the factors that lead to this problem and propose a lightweight process enhancement to cope with it. This enhancement will

introduce a role named “mediator” - a person whose task it is to make it easier for new developers to enter the project and support the knowledge transfer between developers. The role is then implemented in the project GNU Classpath and evaluated by it’s developers with the help of a survey. The key aspects of mediation are summarized and abstracted in form of a short manual which is targetted for use by any F/OSS project.

Contents

1 Introduction

After more than 20 years of Free and Open Source Software (F/OSS) development it should have been proved that this model is enduring. I leave the question whether F/OSS will ever dominate the software market to someone else and focus on lowering their entry level and improving the maintainability of these ever-evolving software projects.

As F/OSS projects get older it gets more difficult for newcomers to join it and they are less manageable for a single person. Part of this problem is that the development process is seldom documented consistently (e.g. archival of architectural decisions). This thesis will demonstrate "mediation" as one solution to get these defficiency under control.

Bigger F/OSS projects consist of a distributed team of developers which often crosses timezones and cultural borders. While this may be good for creativity and balance between interests this makes project management

more difficult. The problems manifest themself when making an appointment or when a debate gets hot because of different cultural tempers.

Usually F/OSS developers are motivated intrinsically which means they do programming for the fun of it. Again this is quite good for the actual result but it means that less amusing work like writing documentation gets neglected. Furthermore being on his own means that a developer can completely chose his development environment and tools.

Communication and knowledge transfer is dominated by mailing list and Internet Relay Chat (IRC) usage. These systems are not designed for information management and make it hard to use it for them as the project's library.

The goal of this thesis is to define a light-weight process enhancement, which minds the factors stated above and heightens the efficiency of the project. The enhancement named "mediation" will introduce the role of a project member who explicitly cares about the project's information, writes important issues (e.g. outcome of a discussion) down and makes them available for future reference by new and long-established developers.

I will install "mediation" in the project GNU Classpath[1] where the enhancement will be tested and qualitative advantages (as well as disadvantages) recorded.

Eventually the key ideas of the process enhancement will be abstracted and compiled as a set of guidelines for other projects as well.

1.1 Disambiguation

For this thesis I use the term Free and Open Source software (F/OSS). However in general I prefer the shorter and older (1984) free software definition[2] and not the newer term "open source" which was coined 1998 by the Open Source Intiative and describes itself as

> "a marketing program for free software"[3].

[1] http://classpath.org

[2] http://www.gnu.org/philosophy/free-sw.html

[3] http://opensource.org/advocacy/faq.php - How is "open source" related to "free software"?

Finally the project which is presented as part of this thesis describes itself as a free software project and it would be hard to find the reason for it's existence by being

> "on solid pragmatic grounds"

only, as the OSI says it.

1.2 Categories and conditions of F/OSS projects

As a base work for the presentation of my process enhancement I will categorize F/OSS projects and describe their development and sometimes social conditions. In later sections I will make back references to the issues explained here.

1.2.1 Categories

This section will give you an overview of categories of F/OSS projects which is simplified to fit in the context of this thesis. Using the following terms allows me to quickly describe the characteristics of a project at a later time.

Single Person Projects

Any F/OSS project that has only member who is responsible for the development of the software is a "single person project". Undertaking of this kind tend to be short-lived because when the initial motivation of the founder goes away no developer is left to take over the maintainership. While one may be tempted to think that this is a great loss for the F/OSS community which makes it less productive, maintaining a non-critical software for a while is a valuable experience for newcomers: The project founder learns the basics of administration like setting up a source code respository, maintaining mailing-lists and a project homepage. This knowledge can then be useful when participating in an established and bigger project.

A single person project may evolve into some of the other project forms when more developers get interested and join it. Eventually there are a

few projects which become successful but stay maintained by a single person for a long time. Examples of this kind are the QEmu[4] multiple CPU emulator and the cdrecord tools by Jörg Schilling[5].

Community-based project

A large number of applications serve the need of the free and opensource community and have been created as some members of the community felt the need for such a program. Sometimes a group of developers gathers around a piece of source code that was once proprietary and got released by its copyright holder.

Projects that belong to this category make up the backbone of the F/OSS community because of their large amount and the wealth of knowledge that is contained in them. A main characteristic of these projects is that there is not much commercial interest. That leads to very informal project management styles where every aspect relies purely on social interactions of its members. One can safely say that they are the most free and independent projects.

Organised community project

Since the early days of the F/OSS development people have organised themselves in larger communities where an individual project is part of the main goal of this community. These communities usually provide a common code guide, documentation rules and guidelines for project management.

The oldest communities have been formed around the BSD and the GNU project. In the recent times we can see the development of the Apache, Debian, KDE an Mozilla communities. While some of them have formed legal entities like the Free Software Foundation[6], the Apache Software Foundation[7] or the Mozilla Foundation[8] others remain an informal group but are nevertheless known in the whole F/OSS community.

[4]http://www.qemu.org
[5]http://cdrecord.berlios.de
[6]http://www.fsf.org
[7]http://www.apache.org/foundation
[8]http://www.mozilla.org/foundation

An important fact about these groups is that they are mutually dependent (e.g. Apache web server running on OpenBSD, being compiled by the GCC) and often developers dedicated to one group work partly on other (e.g. porting GNU software to the BSD platform). Software projects belonging to such a group usually inherit their guidelines[9] and are thus more organised then their completely unbound counterparts. As an example the GNU projects publishes and maintains their "GNU Coding Standards"[10]. These guidelines not only manifest itself as documents but find their way into GNU applications, too: The Automake program is used to simplify the construction of software[11]. In the default operating mode it expects a certain set of files which is defined in the GNU coding standards (see Automake Manual[12]) and can only be overridden by a command line switch named "–foreign".

Company controlled projects

In the last years several closed-source applications have been opened up by their companies in the hope of having synergy effects by approaching the F/OSS community. Taking Mozilla as a prominent example we have seen that it takes some time and commitment by the copyright holder for a released project to get adopted and developed by the community. As long as the software stays under control of their company there is a high risk, that the traditional development process dominates. One of these problems is having face to face meetings of employees instead of organising an appointment on IRC or some other form of communication which includes all project members.

On April 20th 2005 a news article on computerworld.com[13] reported that the well-known OpenOffice.org[14] project suffers from lack of volunteer contributors. Besides 50 developers from Sun Microsystems only 4 active free contributors have been counted. Later one of those volunteers commented his experience with the OO.o development team in the following way:

[9] This is often an acceptance criteria.
[10] http://www.gnu.org/prep/standards
[11] Compilation, library building, installation, ...
[12] http://sources.redhat.com/automake/automake.html#Strictness
[13] http://www.computerworld.com/developmenttopics/development/story/0,10801,101210,00.html?source=x
[14] http://www.openoffice.org

> "Since all the main coders work at Sun, you pretty much stand no chance in hell of doing work on core components, except bugfixing. So, for example, don't expect to sign up to the mailing lists and have any clue what people are working on. Don't expect to be informed of major changes coming down the line unless you have somebody on the inside to give you the scoop. Don't expect to get involved in design discussions, don't expect to have any input on scheduling, don't expect to be consulted about anything except when you're going to fix bugs in your code, don't expect to gain influence in the project over time as you become an established, respected developer. In short, don't expect anything that you would normally expect from an open source project."[15]

While I do not claim that his stance is the only reason for OO.o's and other company guided project's lack of volunteers, it gives an indication on a problem which might be analysed separately.

1.2.2 Environment and conditions

F/OSS projects are very diverse in nature and it makes no sense to define clean borders to get them categorized. However by looking at the environment and the conditions in a number of F/OSS projects one will meet recurring properties. By digging deeper into this topic you will see that projects belonging to any of the bigger groups like GNU or Apache share some characteristics.

Communication

The main medium for communication is still the mailinglist. It is easy to setup, hardly needs administration and there is no project hosting software that does not support them. However the differences between the begin with how a project sets up the mailinglists. Smaller projects usually have a common list for users and developers. Most bigger projects start with a developer list, a user list and another one that broadcasts the commit messages of the revision system (e.g. CVS).

[15] http://developers.slashdot.org/article.pl?sid=05/04/20/2157235&tid=185&tid=102&tid=8 - Post: "I guess I'm one of the four"

The manners on a mailinglist are usually defined informal. One of the bigger differences to traditional company meetings is that the participants of a mailinglist are not forced to read it. It is not unusual that someone has not read a particular discussion or question and is therefore not up-to-date with the latest advancements or decisions. This happens more often when there is high traffic on a list, of course.

One of the good aspects of mailinglists is that they are usually publicly archived.

Internet Relay Chat (IRC) is the another important medium for communication which allows developers and users to quickly get in touch and discuss imminent problems.

What is nice for development is used for socialisation, too. It is not a good idea to underrate the importance of community members chatting about topics such as news, the role of software in the human society or just talk about their families. The maintainer of the later to be introduced project GNU Classpath deliberately set up the project's IRC channel to foster socialisation besides benefitting from the advantages for development. On "Planet Classpath"[16] developers write not only about software but do film reviews as well.

Getting in contact with the developers via IRC is helpful for newcomers, too. Although technically possible IRC meetings are not usually archived. The reason for this is that the discussions are much more informal and sometimes less development-centric as on the mailinglist and the developers prefer the kind of privacy the absence of public archival gives.

Decision making

The process of getting to a result or outcome on a debatable topic is largely undefined. In [Ste00] we get to know that the coordination of discussions is done through social conventions which have to be learned by experience.

A usual discussion starts with a request on the mailing list and every member having an opinion tells what he or she thinks about it. The outcome may be clear when enough striking arguments have been told. Sometimes the maintainer has to intervene to stop "flame wars" or, on

[16] http://planet.classpath.org - A web page where the GNU Classpath developer web blogs are syndicated.

another day, a discussion simply dies because of lack of interest. One kind of critical direction in a discussion is taken when it evolves into a "bikeshed"[17] discussion.

One important difference to decisions traditionally made in software projects is that they are not explicitly written down. Everyone who took part in the discussion may be informed about the outcome. This especially problematic for newcomers because they neither know the outcome nor have they a clue that such a discussion has ever taken place. The mailing list archive helps finding these discussion but it should be noted that it may be difficult to understand its context.

Tool usage

Traditionally the tools used by F/OSS debelopers are born from the community itself. The highly successful Concurrent Versions System (CVS[18]) started as a set of shell scripts which where later rewritten as a real application. New functionality and features where added as the need for the arised. Even Subversion[19] which is today treated as CVS genuine successor has its roots in the community because it was designed to overcome the problems people had with CVS.

Another well-known application is Emacs which has a long-history and still today is used by many developers even there are viable alternatives as KDevelop, Anjuta or Eclipse. And last but not least there is still a living community around one of the oldest editors namely vi[m].

As the development tools are freely chosen by their perceived usefulness, their users are unlikely to adopt newer or better-marketed tools. It is considered bad behavior forcing someone to use a specific program to do a certain task. That said developers like automatisation on a low and easily controllable way. The way considered the easiest is often writing scripts in a language like Perl or Bash[20] script. When wanting to introduce a better kind of working one has to keep in mind this attitude towards tool usage.

[17]The word is taken from an analogy which says that few things to be discussed when something as complicated as a nucler power plant has to be built but a huge debate arises around something simple like a bikeshed.

[18]http://www.cvshome.org

[19]http://www.subversion.org

[20]http://www.gnu.org/software/bash

1.2.3 Definition of success

In order to enhance a F/OSS project I needed to know how it defines success. As the F/OSS community works a little different from the commercial world it is not trivial to define or work out what these measures of success are.

In [CAH03] the authors work out draft ideas about success in traditional commercial software and F/OSS projects. Furthermore they do an interesting experiment by asking the question about F/OSS success measures on Slashdot[21], a well-known news site among F/OSS developers, and analysing the answers. The cited paper does not claim to have found the ultimate answer to F/OSS project's success measures and I do not so either. However it gave me a direction and some values which underline the assumptions.

Furthermore the Slashdot experiment demonstrated to me that I have to keep in mind that parts of the success definition are inherently subjective: It makes no sense to define external project success requirement as "GNU/Linux has to reach a market share of 50% on desktop systems by the end of 2005." when an individual developer values it as success, that he can work with the hardware device he has just written a driver for.

As you see success measurement of F/OSS project is a very interesting and debatable topic. However I consider going more into detail is not benefecial to my work. From the list of measures given by Crowston et al I picked three from which I can say that they are explicitly targetted by my approach. The following sections will present these measures and discusses their importance.

Developer count

F/OSS projects are considered open-ended. Software as we understand today evolves and with this steady evolution goes the need for developers who actively contribute to the project. It should be clear that even the most dedicated amongst the participants of a F/OSS project may leave one day to do something else. A simple search on an online search platform for the phrase "needs new maintainer" will quickly reveal numerous hits to mail archives where a new maintainer to an existing F/OSS project is requested.

[21] http://slashdot.org

While actively asking for a new caretaker is one form to prevent that a project gets dormant I will aim at getting it interesting enough that new contributors find their way into the project before the last developer steps out.

Level of activity

The level of mailing list and RCS commit activity may be considered as a good measure of success but I suggest to be careful here: While a high activity is hardly considered harmful a low or suspended activity should not be equated with the project's death.

As an example the gplflash[22] project was left unmaintained for 4 years before a developer volunteered and took over it's maintainership. There is even a whole effort called Unmaintained Free Software[23] whose task is to list F/OSS projects that have no active developers and hopefully find a new caretaker for it.

Even though we know learned that F/OSS projects may be reborn I will address the level of activity as a success indicator that should be kept at a high level.

Developer satisfaction

Although this measure showed up very strongly in the analysis in [CAH03] it is in itself not definite what actually provides the satisfaction. However out of common sense I can construct certain situations where many will agree, that it will have a satisfying effect on the developer:

- A user of the F/OSS program or library thanks it's author personally.
- A developer implemented a complicated feature that finally worked the way it was intended.
- The program or library starts to serve the need it was developed for.

While my approach to leveraging F/OSS development is not actively influencing the users it will aim for giving the satisfaction of the kind of the second and third example to the developers.

[22] http://gplflash.sf.net
[23] http://www.unmaintained-free-software.org

2 Introducing mediation

Einführung zum Abschnitt

2.1 Which difficulties exist?

We have seen that discussion on the mailing list are held informal and are fundamentally different to a meeting in the commercial environment, where it is crucial for the projects advancement to get to a concrete decision. In F/OSS project we have the following properties:

- There is no force to get to a conclusion to a certain point of time.
- If none of the participant has a clever idea the discussion remains without a result.
- If opposing opinions clash upon each other and no consens can be reached there will be no consistent result. Furthermore there is no administration that forces to reach that conclusion.

The usage of simple communication means like mailing-lists and IRC has technical obstacles: Responds to emails may have a delay from some minutes to several days. In contrast to traditional (face-to-face) discussions where the memory of the participants is generally fresh, email-based discussion bear the risk of simply forgetting former utterances. This even more likely when a participant follows a minor branch of the discussion. However it is exceptionally good that email discussions are publicly archived.

Regarding IRC we will notice that statements are presented in list form. Overlapping answers make it hard to not to lose the plot.

Finally one of the major drawbacks is that at the end of the discussion only their participants know about the conclusion. Even if someone writes another mails summarizing the outcome this message is buried in the archive after a few weeks. This is especially bad for persons who join the project after the decision is made.

2.2 The idea of mediation

The goal of my process enhancement is to stem the problems mentioned above. I therefore define a role, whose task is to be attentive in critical situations and makes sure that valuable information does not get lost.

Later I will explain which are these situations and which criterias should be used when selecting what to record. These ideas will be consciously left vague as variation occurs within each project and I assume that after a certain time of familiarization it becomes clearer what the project values as important and what not. Finally I will present the Wiki as a system that lends itself for information collection.

In F/OSS projects it is usual that its members have a high level of freedom to chose how they contribute. Besides programmers people who care about the project's homepage are gladly viewed. It is up to the participant in how many fields he gets involved. The mediator role will be just another remit.

In other words I am simply going to add another component to the F/OSS development process. Furthermore I will in combining familiar technology and tools to something new, I follow a principle that is used to many F/OSS developers from the Unix tools.

2.3 Selected tasks

Within the scope of this thesis I have selected the following taks, which appeared to me as being the most promising.

Help for beginners

If someone joins a project, all decision made in the past are not visible for this person. In efforts of great age, changing developers and high count of participant mistakable and confusing implementations are possible. Some design decisions may be known to the older participant as historically justified but was never written down. The mediation effort therefore tries to collect information for beginners, which will make it easier for them to get used to the project and effectively lowers the entry barrier.

Achieve an overview about the project

Due to the voluntary collaboration nobody can demand to have an overview on the state of individual parts of the project. Even the maintainer, who may traditionally be regarded as responsible for this, is in no way obliged to be familiar with every section of the project. It will be up to the mediation effort to scan the relevant communication channels for specialist

knowledge and developer decisions and keep this in form of a summary: On the mailing list it is usual that developers announce to do a certain work. Since this commonly is not a reason for a bigger discussion such news easily perishs and falls into oblivion. It is then the task of the mediator to keep hold of such announcements and update them accordingly.

Support decision making

Dicussions on the mailing list do not always lead to a precise result or do not cover all possible cases of a problem. To enhance this situation the mediator should raise a topic when these unclarites occur and therefore aim at clarification.

Raise consciousness for the mediation effort

The difficulty of the mediation is largely dependent on the support of the remaining project members. In an ideal world a mediator would not be needed because every participant would collect relevant information on itself. However I regard this approach as not being able to reach consensus. Clues in emails that someone wants a certain issue being kept would make the mediator's job easier. It is therefore a minor task to convey the sense of mediation and to get the participants to interact with the mediator.

2.4 Project properties supporting the success of the mediation role

The mediation effort is not applicable to every project. Some have found different possibilities to cope with the problems or their personal structure makes it hard to apply mediation. In the following I present certain project properties and why their appearance makes spurs the mediation effort.

Project size/complexity

The mediation effort makes sense if a software project contains multiple modules that may evolve independently from each other. In this situation the mediator cares for a better overview.

No constraints on the choice of work

The freedom to chose on what to work on is an important requirement that most F/OSS projects provide. What it makes interesting for mediation is that this freedom allows a developer to leave his tracks on very distinct parts of the source code. It is likely that he does not really understand the design of the piece of code. Mediation can help here by providing the decision that led to the design of a particular module.

Little formalism

When I speak of formalism for F/OSS projects I mean guidelines on how to document a discussion and whether software design papers are in use. In the F/OSS community many project do not have this kind of formalism or only feature a little amount of it. Although being a quiete fast process in the beginning problems evolve at a later point when written information would be more helpful. This is were the mediator comes into play to distill that information from the project's archived communication.

2.5 Related works

Mediation is by far not the only idea to enhance the development process of F/OSS projects. This section presents other works which focus on information management but follow a different approach.

Hipikat

Hipikat is an Eclipse plugin to automatically process project data from various repositories. It is targetted to newcomers of Java projects and was tested in the Eclipse community. The tool is able to read search requests and retrieves its information from the source code repository (CVS), the issue management software (Bugzilla), the project's mailing-list and newsgroup.

Being a good tool for it's projected goal it is not able to build a repository of information that can be read like documentation. As time goes by the amount of search results gets bigger and the every user has to find out the history of the project itself. Apart from that it is a Java application that

depends on Eclipse. Forcing F/OSS developers to use this platform conflicts with my goal to non-intrusively enhance the development process.

Kerneltraffic[24]

Kerneltraffic is a project which monitors the development mailing-lists of F/OS software projects. The authors scan the posts for interesting events and discussions in order to summarize the content. These summaries are usually published on a weekly schedule and are available in multiple data formats. Currently kerneltraffic actively monitors the famous Linux kernel mailing list and the developer mailing list of the Wine project. The purpose of kerneltraffic differs largely from what the mediation effort wants to achieve. While the focus of the former is on publishing news, mediation is centered on the own project solely.

Kernelnewbies[25]

Kernelnewbies is a whole project dedicated on teaching programmers about operating systems kernels in a way that the participant can fix problems in it themselves. The project mainly focusses on the linux kernel but accepts others, too. It features a homepage with FAQ page, a mailing list, a Wiki and an IRC channel. Kernelnewbies is pretty close to the mediation effort but there are some major distinctions:

- it is separated from the development project
- it focusses on operating system kernel development only
- it addresses new developers

Linux Kernel Janitors[26]

The Linux Kernel Janitors are a kind of support team for the kernel developers. While others implement new features and drivers, the janitors clean up the source code of older modules. The project is meant for new

[24]http://www.kerneltraffic.org
[25]http://www.kernelnewbies.org
[26]http://www.kerneljanitors.org

developers which want to get in touch with kernel development. As janitors these people can do small and straightforward tasks and thereby learn how code for the linux kernel has to be written. Like Kernelnewbies this project focusses on new developers only and it is not meant to build a database of development information over time. However an interesting aspect is that this kind of mediation contains practical work.

3 Implementing the mediator

3.1 Introduction to GNU Classpath

GNU Classpath is an effort to write a clean-room implementation of the Java class library and distribute it under a Free Software license. The software does not work as a stand-alone product and has to be combined with a runtime (Java virtual machine) instead. Classpath is used in projects from classical Java virtual machines, over bindings to other languages (.NET, Oberon, Scheme) to complete Java-based operation systems. The ultimate goal is that several runtimes can use Classpath out of the box without modifying it.

GNU Classpath was founded in 1998 and has about 50 developers from which are 30 actively working on it. The number of developers working voluntarily for the project is predominant while others are employees of F/OSS friendly IT firms (ie. Red Hat).

A special aspect of Classpath is that it's developers are often involved in dependent projects. That means that their work on Classpath can be seen as a cooperation between these projects. The effect of this circumstance is that many different requirements are posed towards Classpath because each project has different conditions.

3.2 Why GNU Classpath has been chosen for exemplifying mediation

With its foundation being 7 years ago the project nearly promises to have burried major design decisions in its sourcecode. With developers changing over time the knowledge about these implementation was lost. Besides the age there was a big focus change from the time where GNU

Classpath supported only a single virtual machine to today's state where it is used by around 10 different projects.

From the statistics I learned that there are 20 former contributors. This means that understanding someone else's code and intentions is getting important for new developers.

Since Java packages are usually quite independent from each other, their development can be done without much arrangement between the contributors. The developer may get the impression easier that explaining his intentions when implementing a public API is not important.

3.3 Formulation of the invitation mail

I decided to write an invitation mail which described the process enhancement of mediation and how it should be applied to Classpath. In this vein I hoped to receive feedback on my plans which could be helpful. The invitation mail was first send to Classpath's maintainer Mark Wielaard to make sure that the portrayed approach was understandable to anyone who was not involved in the planning phase.

I composed the invitation mail with hindsight to the circumstances described in section 1.2.2 on page 8. Since I could not assume that the addressee know about the usual terms of software engineering I avoided their use. I was prepared to receive some opposition or at least lack of understanding and therefore clarified my intentions using examples. Futhermore I uttered my respect to the ensuring of everyone's privacy and made suggestion for anonymisation since there may be persons who attach special importance to this.

As I developed for GNU Classpath myself, I described problems with the project from this perspective.

Reaction Mark Wielaard I

Mark Wielaard is maintainer of GNU Classpath since 2003 and his answer was very clear in favor of the mediation effort as well as my scientific study of this. I was a bit surprised by this reaction and thought there will be more reservations and problems of understanding with my approach.

Besides his approval he told me that my assumption about voluntary work holds for GNU Classpath. He said he has done the first steps to bring the developers together on a social level by creating two IRC channels.

One of them is used for developers, users and other interested persons of GNU Classpath and GCJ[27] which act as a rally point for questions and problems with the software.

Reaction Andrew John Hughes (AJH)

AJH has expressed positively about my plans but notes that he considers GNU Classpath not being a regular F/OSS project with scientific or commercial background. In his opinion the work of the mediator is more suited to someone who does not actively program. Since GNU Classpath cannot accept source code from developers having seen Sun's implementation, persons which are tainted in this regard have the possibility to do the non-programming tasks. He thinks of this as some kind of selection "by policy" although this has not been used much so far. AJH thinks that one of Classpath's main difference to other F/OSS projects is that it's development team does not fluctuate.

Reaction Mark Wielaard II

Mark answered to AJH expression and defended the position that GNU Classpath is a rather regular F/OSS project because code acception policies are in use at the Linux kernel and Apache Software Foundation, too.

Reaction Michael Koch (MK)

MK is a Classpath developer since 2002 who is known for his high quantity of contributions and work on a wide variety of modules. MK said that he is in favor of the mediation idea and expressed his concerns about the problems of beginners. In his opinion the needed information exists but cannot be found easily. Furthermore he thinks it is hard for beginners to figure out what they can work on. When applying mediation MK wants to have assurance that this will not hinder the experienced developers doing their job.

[27] A part of the GNU Compiler Collection and sister project of GNU Classpath.

Other

The remaining mails dealt with the distinctive feature of having an imperative proof of the origin of the sourcecode. This proof was always mandatory for GNU projects but is evolving for other projects, too.

3.4 Conclusion

Despite my concerns the idea of mediation was generally accepted. I had expected more opposition. AJH sees less fluctuation on Classpath than on other projects, however in __LINK!__ we learn that the successfull F/OSS projects feature a stable core group of developers. The points addressed by the answers gave me some hints on how to fine tune the mediation effort. Michael Koch's concerns to not to hinder the experienced programmers should remember me not to send too much mails to the mailing list. Finding a place to start was mentioned and coincendentally was my problem too before I joined GNU Classpath. The mediation effort should therefore not forget about this problem.

3.5 Task and procedure of Classpath's mediator

The mediator wants to collate the knowledge which is spreaded on single developers and thereby make it accessible to all of the project members. The mediator feels responsible for this job in particular but should not impose a restriction to modify the data collected by him. This way another project member can change something that was misinterpreted or needs an update on its own. For this purpose a Wiki seems to be a viable option.

In the following paragraphs I explain the mediator's work and what the benefits of this are. Later on I will study these guidelines in practice.

Collate support and information for beginners

The mediator collects data which is of importance for beginners to successfully get familiar with the project. Part of this information may be coding guidelines and conventions or patch commit rules. One can think of this data as a list of frequently asked questions (FAQ) for new developers.

In detail the mediator scans the usual communication channels for project centered and beginner relevant information. Questions on the mailing list asked by beginners are a good way to find this information, too.

If something interesting was found the mediator should summarize the problem and put it into the database.

Support finding a solution to unanswered and periodical recurring questions

Sometimes it happens that a certain problem is addressed multiple times by one or more persons over a longer period without getting to a conclusion. The mediator's job is to support getting to a discussion about the problem in order to reach an agreement. This outcome can then be added to the database.

At first identifying the recurring topics is the major task. It is up to the mediator how he deals with that. Sometimes it is another project member who remembers the an earlier discussion and alludes to this in one of his answers. Careful reading of the mailing list is therefore very important for the task.

When the topic has been identified the mediator should pose a request to discuss the topic. This request should suport for the addressed persons by summarizing what the problem is and what the current conclusion is. Links to former discussion should be added as well. The mailing list archive can be helpful for this.

After the discussion has taken place and an outcome it should be put to the database along with a notice to the mailing list.

Summarize and process decisions

Dicussions on the smaller and bigger implementation problems are common on the developer mailing list. Sometimes bigger changes to the source-code have to be done in steps and the intermediary changes are announced. The mediator should detect such mails and put the relevant information into the database.

In detail his task is to follow discussion and remember items which were granted agreement. When the discussion reached the point where an outcome is clear this should be summarized and added to the database.

With a notification in form of a mail about this new entry the other developers can then check the validity of the summary.

Maintenance of the database

The base idea is that the collected data ages and may get outdated as the development of the project goes on. To be of use for the developers it is neccessary to keep the data up to date.

One way to do this is to pay attention to mails or IRC chats about an already recorded topic and update the entries accordingly.

However I consider it to be much better if a developer knows that there is something written about a topic he is working on. This way the developer can update the issue on its own when something has changed.

In order to inform the other developers about issues dealing with their work, the mediator sends an announcement about the newly added data to the mailing list. As a side effect the affected persons can check whether the information was summarized correctly and may change it if not.

3.6 Further ideas

Besides the tasks presented above there are more topics which might be included in the mediation effort. As time for the experiment was limited I decided to let them out at first although they might be interesting for GNU Classpath, too.

Collection of long-term goals

Real meetings at yearly F/OSS developer conventions are sometimes used to discuss and make long-term plans. By writing them down as mediation data this information can help developers to find out where the project is heading.

Evolve project's development policies

Community projects with no further ties to a larger organisation have to find their own policies regarding topics like the release interval, the definition of a release critical bug, patch commit rules or coding style. I think

it is obvious that for such projects it is quite handy to write these policies down to make them available for newcomers.

3.7 Guidelines for the daily work

The mediator role is meant as a process enhancement which should be integrated in an existing software development process. To increase acceptance in the project the additional work should not hinder the regular participants. Furthermore the principle of voluntary work should not be undermined. Therefore I suggest the following rules.

No force on collaboration

A key aspect of F/OSS projects is that members do their work and contribution voluntarily. Developers react in a negative way [Ste00] when they are ordered to work on a certain problem.

The mediator should therefore animate the other developer to do active contributions to mediation but should not enforce this. I consider it important to write this down as part of the self-conception of the mediator.

No force to use additional software

Developers in F/OSS projects have their very own belief which software they use for a certain task. A solution where someone is obliged to use a specific (or new) tool will certainly be rejected. In the presented example project the mediator will only use existing and well-known applications and I strongly encourage this for others, too.

3.8 Dealing with problematic situations

The mediator is a job that deals with people's reactions and depends largely on their commitment. It is likely that conflicts or problems will arise some time and the following paragraphs present guidelines how to deal with such a situation.

Lack of interest on a conclusion

It is not seldom that a discussion on the mailing list ends before it has really begun. Sometimes people simply do not know about a topic or miss a question because it got buried between other posts.

The mediator should balance whether an unfinished discussion warrants another request and formulate one when neccessary. He can use the reactions on this request as an indicator whether the topic is of general interest which should be put into the database. If no conclusion can be reached the topic can be considered not being important.

Contrary opinions until the end

When discussions in technical communities can reach no consensus and the only outcome is a solomonic answer which for instance leaves the answer to the one who fixes the problem first.

There is not much to do what the mediator can do in such a case but writing down the problem's nature and its suboptimal solution.

Subjectivity

When summarizing information from mailing list posts there is always the risk of displacing someone's opinion or presenting the circumstance improperly. This is a problem because the summaries of discussion should be considered as it's consens and not the mediator's personal opinion.

For errors in the source code F/OSS project heavily rely on peer-review and I see no reason why this should not work for the mediation effort as well. By making sure that everyone else besides the mediator has write-access to the database the risk of recording something wrong or improper can be reduced. By this means the mediator's influence on the final.

3.9 Chosing the tools

After the mediation idea was clearly formulated and the contact with GNU Classpath was established the missing component was a mean to be used as a database for the collected information.

Wiki

There are strong reasons to use a Wiki for the collation of mediation data: To access the data only an internet connection and a standard web browser is needed to bring the user in the reader as well as the editor position. User accounts are optional and are a a mean of convenience to make it easier to track changes thereby the administrative overhead is very low.

The retrieval of old versions of a document is supported by the versioning feature which most Wiki systems have.

Finally the special Wiki formatting syntax can be learned very quickly or can at least be imitated from the data that was already written. For simple changes the special syntax is not even needed what makes the Wiki usage as simple as a standard text editor. Many developers know Wikis already because of the work done by Ward Cunningham and Wikipedia.

The ubiquitous and barrier-free editing capabilities of the Wiki turned out to be of great help for editing the mediation data immediately after something interesting was said on IRC.

Nevertheless the Wiki system has some flaws and should be noted. One problem is that the pages can be edited by everyone and allows them to deface them. We faced this problem in the beginning when several pages where filled with Wiki spam and we had to use the history function to revert the changes.

Normally Wikis are organised as a net of links between the various pages. Applying this organisation to the mediation Wiki would have made it more difficult to find relevant information. I have chosen to use the Wiki as a kind of dynamic homepage and it remains to be seen whether this style is generally accepted upon the developers.

Another concern is that the data in the Wiki sometimes duplicates other information sources like the "README" file, the hacking guide and the project's homepage and administration system (Savannah). I consider this a kind of struggle of responsibility whose outcome depends on the project member's critics.

The Wiki is a very flexible tool and can be tailored to a wide area of uses and turns out to be handy for the basic needs of the mediation effort. However it gets problematic when the number of articles rises. Then there is no mean to easily group or order them alphabetically.

Subdirectory inside the CVS

One of my first ideas was to use a special subfolder inside the source directory and manage a set of HTML or TexInfo files inside it. My intention was that every developer should have a copy of the mediation data when he checks out the sources from CVS allowing him to use it locally.

However for the mediation effort the database had to suppport frequent and small changes. This kind of editing quickly gets tedious with CVS because its setup is optimized for code changes: Every committed change results in a acknowledgment mail on a special mailing list and the description of the change has to be written into a special separate file (the "ChangeLog").

Another problem is that publishing the mediation data would require additional work: The data from the repository had to be converted to HTML and then uploaded to the project server after each change.

Project management system

It has to be noted that GNU Classpath is hosted by the Savane project management system which is a fork of the Sourceforge software. This system provides things like a bug, patch and task tracker, a mailing list and a system to publish news.

While this platform is invaluable for the technical part of F/OSS development it does not provide a mean to support mediation. Listing, organising and (re-)editing of small articles is not a feature of that system. The core problem is that the tracker facilities have too much options and configuration options that distract the reader from the written content. Furthermore each change to an entry would mean that another post gets attached. That is bad because I want to be able to modify an existing article.

However with some effort it would be possible to add a subset of Wiki features into the project management software.

3.10 Implementation and problems

I chose a Wiki-based system because it seemed like the most fitting but not without looking at the alternatives. A general introduction and criticism to the Wiki system can be found in [DRAG03].

The F/OSS world has numerous Wiki systems and as the focus is not on finding the best available tool (or create it myself) I decided to use MoinMoin[28] for practical reasons: It features versioning and was already installed use on the target host for a licensing[29] discussion.

Structure

The initial structure of the Wiki was designed to use a small number of single pages in order to minimize the spread of information. The main page linked to pages describing mediation and the mediation Wiki. Another three pages were used to list articles which are called issues, to the following topics:

- information for beginners
- developer decisions
- current development topics

Overview

In order to find an issue a macro of the MoinMoin Wiki was used that creates a table of contents on each page. The entries consist of the issue titles and link to their respective issue.

Search capability

With MoinMoin it is not possible to implement a search capability that searches the content of the issues. However it features a general search function that simply scans the pages.

Form capability

The issues have a fixed format and it would have been nice to use some kind of form based input system for it. While this feature is not present in MoinMoin other Wiki systems like TWiki[30] have it.

[28] http://moinmoin.wikiwikiweb.de/
[29] http://developer.classpath.org/licensing
[30] http://www.twiki.org

Cross-linking

In each issue's body text I embedded links to the sources of relevant information and each issue has a field for references where I placed links that deal with the topic or problem. Usually these links point to Classpath' mailing list archive or to various places on the web where technical information is kept.

Variability

The look and the used fields of the issues are not strictly defined. In the beginning the issues had more fields for administrative data but I soon considered them dispensable because their impaired the ease of use. Although I did not receive critics on the layout of the issues the editing hints for the medation Wiki asked exactly for that.

3.11 Announcement

The official announcement of the Wiki was done on January 16th 2005 using [[http://projects.mi.fu-berlin.de/w/bin/view/SE/ThesisFOSSIMList#Ank_ndigung][this]
mail. At this time the base structure of the Wiki was ready: It contained some issues, a page that described editing in the mediation Wiki and another one that dealt with the goals and uses of the mediation effort. In this state working with the Wiki was possible.

4 The mediation manual

So far the mediation effort was only Classpath-specific and had no chance of being transfered to other projects. To reach this goal I wrote the mediation manual as a set of project-independent guidelines. In a question and answer style I provided the basic ideas of mediation and how it is supposed to work in a F/OSS project.

Since the manual was supposed to be presented to other project members I kept the number of pages low and focussed on quickly getting to the main practical aspects of mediation.

I directed the manual towards project members as well as people who do not have such an afilliation. In order to publish the mediation manual I

sent an announcement to the development mailing lists of several F/OSS projects.

4.1 Announcement I

In this paragraph I portray my experience with the announcement of the mediation manual using the developer mailing lists of free software projects.

4.1.1 Procedure

I wrote a small mail that contains a small presentation of the topic, a link to the mediation manual and several ways to contact me. Around 50 projects from Sourceforge have been selected by looking which of them met the following criterias:

- Project is in alpha or beta state.
  Sourceforge allows projects to classify their development state (planning, alpha, beta, mature, ..). I chose the alpha and beta states because these are projects where sourcecode is present as opposed to projects in the planning state.
- At least 3 or more members.
  Having 3 or more members makes sure that a certain amount of communication between the developers is needed.
- Being founded before January 2004[31].
  By requiring a minimum age I want to make it more likely that the project created a certain amount of historical data. Having experienced 12 months of development is a reasonable amount of time for a project to evolve.
- At least one release between 2003 and 2005.
  As the interest is on projects which are alive the existence of an release makes it more likely that someone is still actively working on it.

[31] At the time of this writing this meant 12 months ago.

4.1.2 Reactions

The first reactions afters sending my announcement came from 20 mailing list servers which forbade me to send mails withouth being registered. However such systems allow that the list moderator manually permits the mail to go on the list. I hoped that most moderators took a senseful decision since I had no bad intentions. In the end 12 list moderators allowed the mail to pass while 8 mails where lost and another 30 reached their target without any problems at all.

The human answers turned out mostly good. There where 3 developers who said being interested and sent me a number of syntactical and grammatical corrections.

The maintainer of wxGlade told me that he thinks that mediation is a good idea but regrets that his project has no stable members and he cannot take another role for his project.

A developer of the Syllable operating system effort told me that two people in the project are doing something that is comparable to mediation. Hereupon I contacted him to get to know more about this work. I will present his answers below. A little discussion went on in the PearPC project whose outcome I present below as well.

The NHibernate project rated the mediation approach as not being very helpful. However they considered using a Wiki because it seemed to be a good idea for them.

There where two less friendly reactions: One of them complained about the manual expects having set white as the default background color and another one found my well-meant mediation manual announcement worse than spam.

Communication with Brent P. Newhall from Syllable

Brent Newhall is a developer of the Syllable project which aims to write an easy to use desktop operating system. As Newhall told me he and Michael Saunders are doing something comparable to mediation I got interested and asked him questions about this work.

I thereby got to know that he is doing the medation role voluntarily and without any special decision of the core developer team which he does additionally to his programming work.

Newhalls work consists mainly in the writing of a system documentation for the operating system. This documentation is predominantly directed at the user and not meant for project internal discussion. Newhall writes the documentation in a Wiki at whereas he accepts comments and changes from other people, too. The results of this work are available online[32]. The work done by Newhall is not exactly mediation but he does a part of it: His documentation describes system programming with Syllable which obviusly lowers the entry barrier for new developers.

Newhall mainly receives feedback for his work from the project's mailinglist. Besides helpful suggestions he sometimes receives documentation contributions and a few mails from beginners who told him that his work made it easier for them to get into the project's details.

I asked Newhall about the time the work consumes and he estimated, that he spends several hours per week with fixing the documentation.

Finally I wanted to know how ardous he thinks this work is. In his opinion this was a nonsensical question because he thinks that the level of difficulty depends on a person's previous knowledge. He told me that he is a documentation writer by trade and thereby it is easy for him to do this work for the Syllable project. However he thinks that there are a lot of programmers which will find this a difficult task.

I asked Newhall about the work done by Saunders and found out that he is writing developer mailing list summaries similar to the one made by Kerneltraffic. Saunders selects interesting discussions of the last month and comments their content. If the mail contains a request to participate he caters on this and forwards it to his readers. The results of his work are put on his homepage[33].

Reactions of the PearcPC developers

From the PearPC project I received the answer that mediation may be very helpful to them because they see a big discrepancy between the knowledge of their developers and their users. I got to know that they have been some small attempts to document the current state of development using a forum thread and a Wiki. However the developers enganged with this work suffered from lack of time whereby its slowed down. In the end

[32]http://www.other-space.com/sub

[33]http://msa.section.me.uk/sdn

the PearPC team likes the idea of having the development process documented but their volunteers lack of time and new ones have not been found.

4.2 Conclusion

Measured by the number of mails I send I had expected more reactions. However the answers I got are mostly positive and the idea of mediation was presented to a bigger public without any noteworthy opposition. The ones who have read my manual now know about the idea of a mediator for F/OSS projects and I count this as a success.

5 Analysis of the practical implementation

Nearing the end of my study I wanted to receive feedback from the developers of GNU Classpath towards mediation in order to analyse my work. I therefore framed a questionnaire which had to be filled out online using the phpESP[34] software.

The survey aims at finding out the developers' thoughts about mediation as a theoretical concept as well as its actual implementation which I have done. An important aspect of the questionnaire is to have comparable result. Most questions therefore offer 5 fixed answers constituting the varying degrees of agreement. However in order to get more detailed insight about the developer's attitude a number of questions have to be answered with free text.

The survey's questions have been divided into the following categories:

- Knowledge of the developer about the mediation effort.
- Valuation of the mediator practical work.
- Self-assessment of the developer's participation.
- Valuation of the mediation Wiki and the topics chosen.

[34] http://phpesp.sourceforge.net

5.1 Results

The questions about the understanding and knowledge about the mediation effort revealed that while a majority is quite well informed what it is but only a narrow majority knows how to support the mediator. Consequently only a few developers expressed that they know how to do mediation themselves.

The free text section showed that their is a big disparity between well-informed developers and others who do not know anything about mediation. The extreme cases formulate as such:

> "I believe Classpath developers have been kept fairly well informed."

The exact opposite manifests itself simply with this words:

> "I don't know what it is".

A possible answer to why this happened gives the following response:

> "I think I missed the introduction of this effort (because of absence)".

Most developers agreed that mediation helps solving problems and that it is necessary when a software project reaches a certain level of complexity. In conformance with these answers most developers found that the time spend on mediation was not lost to programming. However when it comes to active contributions byte the developers itself there is only a slight majority which thinks that this would be a good idea.

As the current form of medation aimed at helping and involving developers only thr respondents expressed their wish to have less technical weekly news and information resources targetted at end users.

A bit disillusioned where the results of the participation questions: More than half of the respondents had never written a new issue, edited an existing one, answered mediation related questionn or posed a proposal for a new topic.

The usage of the wikie was generally appreciated positively. Though a slight discomfort was measurable because of its the less optimal search

mechanism. A proposal named usingh a WebDAV repository for mediation because it it has better search capabilities. Clear encouragement got the decision to have no discussions in the Wiki.

The final category of questions, a valuation of the mediation topics, brought some interesting ideas: While no one complained about the chosen topics the free text answers demonstrated that there is a need to extend the mediation work. Respondendt who want to extend the target audience asnwered like this:

> "I think we could do a better job at engaging the non-technical audience that's willing to help, [...]."

Others expressed the wish to integrate other forms of data:

> "It would have been fine if the mediator had more agressively added the task list, faq, vm integration guide and GNU Classpath Hacker Guide into the mediation effort"

or

> "Overall architecture, who's working on what, who needs what sort of help, licensing FAQ, schedule and priorities."

6 Conclusion and perspective

The practical expirement showed me that there was much less resistence than expected. Instead the answers of the survey reinforce that mediation is beneficial for the project, that it helps new and long-established developers and that it does no harm to the development process. Furthermore Classpath's members wish to broaden the scope of the mediation effort that it covers a wider audience and more topics.

However some developers have not been informed well about mediation and I therefore plan to reannounce the effort along with some requested changes. The aim is to make sure that every developer knows about mediation and how to support it.

The Wiki proved to be a practical all-purpose tool that worked well for the mediation effort. To overcome the less optimal search function

an integration with the project hosting software might be an interesting option.

My initial design of the issue layout was more complicated: It contained more fields of mandatory information which became hard to maintain for the daily work.

Contrary to my expectations it was less often needed to start discussions on controversial topics. A good source for new issues where explicit requests on the mailing list or discussions on IRC.

By participating in GNU Classpath's development I taught myself how to use certain tools (e.g. GNU M4, Autotools). Theses studies where necessary to work as a team more effectively. Mediation can only help here by showing newcomers what applications should be mastered.

The following incident made more limits of mediation visible: At one point I had to compile and install a snapshot version of the GCC compiler suite and test its Java features. How to solve the small problems that evolve when doing this for the first time should be learned by experience. IRC proved to be a good mean to receive answers and practical help quickly from more experienced developers.

With the mediation manual a project independent set of guidelines have been written. While I received mostly positive feedback about it, a practical test is still outstanding. Furthermore it might be interesting to integrate similar approaches like the kernel janitors, mentors or summary writers to the mediation effort or consider mediation right from the beginning of a F/OSS project.

A Invitation mail for GNU Classpath

Hi fellow GNU Classpath developers,

for some time now I am participating in this project fixing bugs and adding functionality mainly to the java.beans package. Despite my good knowledge of the Java language, participation in development communities and especially the GNU community, is virgin soil to me. As a result I sensed a steep learning curve when I started helping Classpath. In the last weeks I found myself asking a lot of questions on topics which I think are common knowledge for a fair amount of you.

The problematic cases range from specific tool usage over project plans to general policies. I know there is a lot of tool documentation on the net

and a hacking guide for Classpath which is enough for the fundamental stuff like CVS usage or coding guidelines but what I think is still left untouched are questions like:

- What is the outcome of discussions?
- Whom can I ask directly for specific questions?
- What is the general direction of the project? (or: What is considered old stuff which should be avoided in favour to newer decisions?)

Ideas on making this situation better with the intent to make the project participation more enjoyful circled in my head and found their way on a sheet of paper. It was clear to me that the realization of this would need a dedicated effort which cannot be burdened onto someone's shoulders without intrinsic motivation.

After all I am a computer science student who got recently interested in software engineering and was seeking a topic for his semester thesis. I approached the SE group at my department in order to do an academic work around my initial ideas and got positive reponse.

Now I'd like to ask you if you welcome my effort to enhance our project and use the experiences gained from that for academic work.

The following paragraphs describe the planned enhancements (Criticism and comments are welcome):

My basic assumption is that the development process of an unstructured Free software project should be enhanced by non-invasive methods: Any means that make the developer's participation work less comfortable should be avoided. Academic projects with similar goals as mine have largely relied on producing tools which did not get adopted. In contrast to that my approach is based around a role that I call 'mediator'.

In short the mediator's goal in a F/OSS project is to take care that no idea is lost. To be more specific these are some of the things the mediator should do:

- Collect information about project member's interests (e.g. package responsibility).
- Remind of certain events: release, urgent documentation updates, long-term goals.

- Keep an eye on the project documentation and guides.
- Be an active guide for newcomers.
- Dig up or re-introduce ideas which otherwise would get lost in mailing-list conversations.
- Write down the results of decisions and ToDo items.

It should be noted that I consider that some of these tasks require a lot of sensitiveness. A bad formulated ToDo list entry or project decision can lead to unfriendly and heated debate. Furthermore the mediator does not have any higher privileges: Changes to every recorded statement can be made by each project member and the mediator does not make decision, but rather collects them.

The usual work of the mediator will consist of an in-depth study of the mailing-list (archive) but also other communication channels like Classpath's blog area and IRC. Apart from that he stays in touch with the other members and updates the respective documents. The initial effort will be on collecting the existing and upcoming data and finding a suitable way to organize it.

The duration of my thesis is limited to 3 months. At the end of this time we can look at the results of my work and poll whether to continue it or not.

I hope this introduction gave you enough information to get a picture of what I want to do. As stated above criticism and comments are welcome.

Privacy: I respect everyone's privacy but its likely that I will take quotes for my thesis from the mailing-lists (which is already publicly archived) and perhaps from IRC conversations. In the latter case I will address the involved persons and anonymize their statements if they want me to do that.

B Announcement mail of the mediation Wiki

Hi,

the last days I have been entering data for a Wiki on developer.classpath.org/mediation . This place is going to supplement the Hacker's guide, the mailing list and

the homepage (FAQ) by providing useful information about developer decisions. Another whole page deals with issues that might be interesting to new hackers on GNU Classpath.

The Wiki is the most visible part of a work I call mediation. It has a page that explains this work and its aims in detail: http://developer.classpath.org/mediation/Media

There are no obligations on you attached to this work. Involvement is encouraged and appreciated but not enforced. The mission page has more details about this.

If you have questions to anything of the above feel free to ask.

cu Robert

C Mediation Manual

This short manual presents a small set of guidelines for Free and Open Source (FOSS) projects that should lead to a better perceived liveliness and progress. It targets programmers, maintainers and persons currently not involved in a project but willing to participate. The ideas presented here are no rocket-science and you decide on your own how much of it you want to adopt. The general idea is to have a special person - called mediator - who manages and takes care of the project's informations.

Motivation

In Autumn 2004 I joined the GNU Classpath[35] project which is a free implementation of the Java class library. I have a good knowledge in Java and already sent patches to a few free software projects but was never involved in such an undertaking like Classpath.

The project exists since 1998 and has a large amount of source code and numerous developers helped the effort so it was a bit hard for me to get into the game. A major stopper was that I did not know about the project's future plans, where work was needed and what design decisions have been made in the past. Additionally I found it unsatisfying asking questions which the next developer joining after me will ask again.

Soon I started thinking about a way to tackle this problem and how other projects as well could profit from my considerations. The result of

[35] http://gnu.org/software/classpath

that work is presented here.

How do I know that a mediator is a good idea for my project?

If your project has one of the following problems then a mediator might be the right person to add to your project:

- Only a small number of new developers are able to become members of the project because of the complexity of the codebase and their lack of understanding of the project's state.
- Active development is hindered because programmers do not really know what their peers are working on and how the puzzle parts fit together.
- Certain topics are discussed once in a while but no progress seems to take place in their regard.
- Lots of stuff was done, lots of stuff has still to be done but no one knows how far each and every piece has gotten and where it would be good to get started.

Why would I want to solve these problems?

The declared goal is to minimize these problems because such a situation can kill the members' motivation to invest time for their voluntary work. A developer may feel the work as cumbersome and then loses interest. The mediator is going to help the project to cope with these problems.

Why do you call the role "mediator"?

The term mediator is normally used in the context of conflict resolution and means the person who manages a conflict between affected parties. In the context of FOSS projects the conflict to manage is that certain persons have special knowledge or insight while others do not. My position is that it would be better for both parties if the knowledge gets more widespread.

Alright, so what is the mediator all about?

The mediator in a Free and Open Source Software project watches the communication inside a project and compiles the most essential bits and pieces into a concise form. This means that the mediator pays attention to mails and discussions on IRC even if he is not involved or directly affected by these.

An important aspect of his daily work is to look out for unfinished discussions or unanswered questions. Besides that the mediator should apply the 'newbie developer view' to find out what could be important for him. Finally finding out disparities like the big plan that comes up every there and then but was never done is a good trait.

Can you be more specific about what the mediator could actually do?

Watch discussions

Discussions that are held on mailing-lists are the ideal source for information that should be recorded. A mediator should watch all of them and decide which are relevant to be recorded. When the final word was spoken and a result is clear he should summarize the outcome and make it publicly available (for instance on a Wiki).
It is a good idea to allow comments and modifications on the summaries because it is likely that somebody does not like the way it was written down. If a new discussion about the same topic arises it should be clear that the summary has to be updated.

Find out when the same question is asked frequently

Recurring questions from users as well as fellow developers (especially newcomers) are no anomaly these days. What you could learn from them is that they indicate an informational gap that should be closed. If the question has not been answered satisfactory the mediator should look up relevant information from earlier discussions, write a question to the mailing list that displays the problem and its current state. If a result can be achieved that should be summarized and made public by the mediator.

Identify information that is relevant for newbies

In order to make it easier for new developers the mediator should look out for information on coding guidelines or style and commit policies. Ideally these should be available in form of a file in the project folder as many projects do already. This way anyone working on the code has the style guidelines at hand.

Besides this the mediator should look out for implementation pitfalls like documentation that speaks contrarily to what is done in reality. The mediator should explain which variant is the right one and how one is supposed to cope with the problem.

In long-living projects it's likely that it carries a legacy because of an earlier design decision. Maybe there are two internal APIs having the same functionality and it's not clear for newcomers how to handle this. The best thing would be to deprecate and remove one of them but this is sometimes not (yet) possible and in the meantime new developers should be at least aware of the problem. Again that is something the mediator should look out for and describe the problem in a public summary.

Find out fellow developers wishes

By reading emails of developers thoroughly you can often spot indications of wishes. These are sometimes expressed when someone fixed a problem but is not 100% comfortable with the current solution. The mediator should pay special attention to these utterances and get in contact with the developer to find out whether this aspect is important enough to get recorded. After the mediator made the idea publicly available others can review and/or tackle the problem.

Ok, but what about difficulties when doing the mediating?

It's clear that when interacting with people things do not always go round as easily as it should. There could be the problem that the mediator does not receive a real answer. If there is a lack of answers the topic might no be that relevant and the mediator should drop it. Then it's possible that the developers reach no compromise. In this case the mediator might summarize the opinions instead of a concise outcome.

Another problem is that a technical question might be to demanding for the mediator. In this case he should simply publish a draft summary and present it to the developers who know the topic better. If it's wrong there will be complaints and if it is too shallow others will ask for more information which the mediator can then add to reinforce the draft summary.

Can you give some practical examples for something a mediator has done?

Here are examples from the mediator's work at the GNU Classpath project.

Recurring question that was made available for newcomers afterwards

A developer who had seen the source code of Sun's Java class library is considered tainted and cannot work on the code in GNU Classpath because of the risk of copyright infringement claims. The FAQ contains a short entry that tells this but there was no other source of information. Newcomers asked whether they are tainted and if so what they could do instead of coding.

In one case a developer was already waiting for a definitive answer for about three months before he reminded the team of the issue. The mediator then send a mail, stating the problem ("What is a tainted developer allowed to work on") and containing answers from earlier mails found in the archive, to the list. The topic was then discussed again and a comprehensive outcome was available later. The mediator then put a summary of the discussion in the Wiki.

Coding style disparity that was found and added to the newcomer's information

While working on the code the mediator noticed documentation tags which where not documented in the FAQ or the developer guidelines. At first the mediator used the tags as they seemed to be used without questioning their meaning. However after a while he found out that the tags are used differently depending on who edited a certain file. The situation was unclear and so he posed a question to the mailing-list. Although two

developers answered the outcome was still not definite because they uttered contradictory. Nevertheless the mediator added a summary about the outcome of the discussion and put it in the Wiki. A few days later one of the developers had read that summary and complained about its content. After having had a small discussion on IRC with both developers the remaining bits could be solved and the summary was updated.

How much time does it require to be a mediator?

The person adopting this role decides on his own how much time is invested. It should be no fulltime job although the beginning might be an exception. The mediator should be supported by his fellow developers who provide him with information, answer questions and tell him occasionaly what is important information that should be recorded.

Why do you think the mediator should use a Wiki?

A Wiki is a really simple and powerful tool: It is to learn how to edit and everyone has equal rights when doing it. It can be used from nearly all Internet-connected computers and you get a version management for free. Finally if the current mediator leaves, somebody else can take up the work, without any need for new passwords etc.

How can I take action?

It depends on your status: If you are a maintainer or core developer you probably have enough work to do so that you do not want to take the mediator role yourself. That means you should find someone who want this job by filling a request form, adding a note to your project page or simply asking on your mailing list.

You should think about the technical requirements like installing a Wiki. Maybe you do not like a Wiki and instead use something else (e.g. letting mediator work on HTML pages in the repository).

Additionally the mediator should get to know where the important information will appear. Most projects use mailing lists but some have a Wiki instead, others rely heavily on IRC talk.

However if you are not yet involved with a project but would like to be then tell the projects maintainer or core group that you are willing to contribute as a mediator. Pointing them to this manual or using it as a base for your invitation mail is a good idea.

Are there any project characteristics that make it likely that a mediator will work?

A team of voluntary developers and a big amount of sourcecode.

The project should be typically community driven in contrast to enterprise driven. The latter might be more resistant against using what the mediator provides them. Besides that the mediator's effort is hindered if the project members have real-life meetings to make a design decision where he cannot attend. Projects led by one developer are problematic because there is no communication between team members which the mediator could improve.

Settled design phase.

The history of the project should be long enough that design decisions are burrowed in the code. Furthermore due to developer fluctuation certain parts of the project may decay or bitrot because no one knows how these are done or understands them any more after certain developers left. Such a situation provides the informational gap which can be closed by the mediator.

References for the Mediation Manual

- Visit GNU Classpath's mediation Wiki[36].
- The Linux kernel spawned several interesting projects which share the mediation idea:
  Kernel Janitors[37], Kerneltraffic[38], Kernelnewbies[39]

[36] http://developer.classpath.org/mediation
[37] http://www.kerneljanitors.org
[38] http://kerneltraffic.org/kernel-traffic/latest.html
[39] http://kernelnewbies.org

- A janitor effort[40] for Inkscape

D Announcement template for mediation manual

Dear %projectname% developers,

I wrote some guidelines that should help FOSS projects getting more lively and lowering the barrier for new developers to join. You can find them in form of a small manual here http://projects.mi.fu-berlin.de/w/bin/view/SE/ThesisFC

These ideas are the result of work for my bachelor thesis and have been used successfully at the GNU Classpath project. If the topic is of interest to you, I would be happy to receive criticism and comments concerning the manual or the general idea.

For further discussion I have set up a mailing-list (use http://lists.spline.inf.fu-berlin.de/mailman/listinfo/mediation_manual to subscribe or mediation_manual@lists.spline.berlin.de to post unsubscribed). Please send your feedback to this list but if you have reasons to contact me directly then just reply to this mail. In case you answer to your project's mailing list please CC me.

Best regards
Robert Schuster

E Survey

1. How long have you been working on GNU Classpath?

Less than a year	18.2%	2
Less than two years	9.1%	1
More than two years	72.7%	8

2. I know what the mediation effort is about.

[40] http://www.inkscape.org/cgi-bin/wiki.pl?InkscapeJanitors
[41] http://creativecommons.org/licenses/by-sa/2.0

I strongly disagree	18.2%	2
I weakly disagree	0%	0
I weakly agree	45.5%	5
I strongly agree	36.4%	4

3. I know how to support the mediator.

I strongly disagree	18.2%	2
I weakly disagree	18.2%	2
I weakly agree	36.4%	4
I strongly agree	27.3%	3

4. I know how to do the mediation work myself.

I strongly disagree	36.4%	4
I weakly disagree	27.3%	3
I weakly agree	36.4%	4
I strongly agree	0%	0

5. In which way could I have been informed better about the mediator and the mediation effort?

   - "don't know, I think I missed the introduction of this effort (because of absence). Maybe this is the only thing that I could have needed."
   - "I believe that as Classpath developers have been kept fairly well informed of the mediation effort. Notifications of the progress with this task have appeared on the Classpath mailing list, and the meditation wiki, developed as part of this, has been regularly updated. The latter has proved invaluable for keeping track of current development tasks and opinions, especially when it is not always possible to regularly check through other less organized mediums such as IRC or the mailing list. It is also extremely benefical to have a permanent record of such, and to be able to direct people to this system for further help. It also ensures that information is not lost, which seems to have been the case before, with conversations frequently being effectively re-run on the mailing list."
   - "I don't know what this is."

- "I haven't had time to contribute to Classpath lately; I saw the email thread about mediation and I would refer to that in the online archives if I were planning on contributing something again that might need mediation"
- "I'm not actively contributing to classpath at this time, so it would help if I read everything on the mailing list."
- "It works seamlessly and well, so that I think it fulfills its role veryu nicely."
- "Perhaps same status reports from time to time sent to the mailinglist. E.g. with access statistics for the mediation wiki."
- "weekly or bi-weekly updates to the mailinglist on what was summarized/added. (Regular, but not too often!)"

6. Mediation helps to solve problems which emerge because work on free software projects is unconstrained (eg. no force to code every day, no force to read all mails on the list, ...).

I strongly disagree	9.1%	1
I weakly disagree	0%	0
I weakly agree	27.3%	3
I strongly agree	45.5%	5
I don't know	18.2%	2

7. Mediation is necessary when a project reaches a certain level of complexity (eg. number of developers, age or amount of source code).

I strongly disagree	9.1%	1
I weakly disagree	9.1%	1
I weakly agree	27.3%	3
I strongly agree	36.4%	4
I don't know	18.2%	2

8. The time consumed on mediation should be better spend on programming.

I strongly disagree	36.4%	4
I weakly disagree	36.4%	4
I weakly agree	0%	0
I strongly agree	18.2%	2
I don't know	9.1%	1

9. Every developer of the project should actively contribute to the mediation effort (eg. add information on his/her current work and its state of affairs).

I strongly disagree	18.2%	2
I weakly disagree	9.1%	1
I weakly agree	36.4%	4
I strongly agree	18.2%	2
I don't know	18.2%	2

10. Mediation results in a data base that will be very helpful for the project in the future.

I strongly disagree	0%	0
I weakly disagree	0%	0
I weakly agree	36.4%	4
I strongly agree	27.3%	3
I don't know	36.4%	4

11. Thanks to the data collected by the mediation effort I have a better overview about the work in progress.

I strongly disagree	9.1%	1
I weakly disagree	18.2%	2
I weakly agree	27.3%	3
I strongly agree	18.2%	2
I don't know	27.3%	3

12. The additional questions asked by the mediator add noise to the mailinglist.

I strongly disagree	54.5%	6
I weakly disagree	27.3%	3
I weakly agree	0%	0
I strongly agree	0%	0
I don't know	18.2%	2

13. Mediation helps new developers.

I strongly disagree	0%	0
I weakly disagree	0%	0
I weakly agree	9.1%	1
I strongly agree	72.7%	8
I don't know	18.2%	2

14. Mediation helps long-established developers.

I strongly disagree	0%	0
I weakly disagree	0%	0
I weakly agree	63.6%	7
I strongly agree	27.3%	3
I don't know	9.1%	1

15. Other projects should have a mediator following the example of GNU Classpath.

I strongly disagree	0%	0
I weakly disagree	0%	0
I weakly agree	9.1%	1
I strongly agree	36.4%	4
I don't know	54.5%	5

16. Who could also benefit from mediation who has not been targetted yet?

    - "Engaging users to participate into building the knowledge pool, which would matter more for non-developer-oriented projects. GNU Classpath (and the associated runtimes) are a little special as they target as rather specific group of users and developers with a high degree of technical expertise, but I think that the expereince with the mediation effort shows that even in such an envorinment the social interaction aspects of the collaborative work can be improved by .having someone look after loose ends, and trying to help drive conversations to conclusions. For engaging end-users, something like 'GNU Classpath Weekly News' would be an interesting project to get the nice work done within the GNU Classpath family of runtimes exposed and presented towards a larger, less technical-oriented audience."
    - "I don't know."

- "I think I don't understand the question right? Does 'Who' mean, 'Which other projects'? Then I think I don't have much overview of the internals of other projects..."
- "I think the current process already covers most roles."
- "The newcomer is able to find resources to get them sorted, which (most importantly) are easy to update, allowing this to be done on a regular basis. The traditional static webpages for Classpath tend to not see this."
- "Infrequent developers can quickly become acquainted with current developments."
- "Regular developers can get a quick insight into what others are doing, and co-ordinate efficiently."
- "The active users of GNU classpath, the VM implementors are not really targeted. They are involved but just because they are involved in GNU classpath itself."

17. In order to be an attractive idea for other projects, which things should the mediator do in a different way?

    - "don't know"
    - "For GNU Classpath we already had an irc channel and blog aggregator/planet. I think setting up these kind of "social" infrastructures will help for other projects."
    - "I can't really say. I've been doing similar work on Kaffe when I started out, and having a good mediator is very useful for getting a good grass-roots community going. It is less of a classical management role, than helping the herd of cats help manage themselves :)"
    - "I don't know."
    - "I think the current process is appropriate for GNU Classpath. It may not be directly adaptable to another project, due to certain nuances within the existing development process. That said, the overall concept and most of what has been achieved here can be easily transplanted. It would be interesting to see how mediation deals with a larger developer base, a different rate of development, etc."

- "More powerful search mechanisms."
- "The mediator should have more intensive contact with the people and be more active. Currently the mediation work in GNU classpath is pretty passive. He watches mailinglists and IRC and updates the mediation Wiki. When more in contact with the people I think it will help even more because the problems can be better digged up."

18. How many new issues have you written for the wiki?

None	72.7%	8
One	27.3%	3
Two	0%	0
Three	0%	0
More than three	0%	0

19. 20. How often have you edited an existing issue in the wiki?

None	54.5%	6
One	18.2%	2
Two	9.1%	1
Three	9.1%	1
More than three	9.1%	1

20. How often have you answered questions posed by the mediator when he was gathering information for an issue?

None	63.6%	7
One	9.1%	1
Two	9.1%	1
Three	9.1%	1
More than three	9.1%	1

21. How often did you came up with the suggestion to add something to the wiki?

None	72.7%	8
One	0%	0
Two	27.3%	3
Three	0%	0
More than three	0%	0

22. What keeps you from dealing with the mediator and his work?

    - "don't know, nothing special probably"
    - "He mostly just picks up the things that are interesting already. He is doing it as a volunteer and doesn't need guiding."
    - "I'm not sure what the mediator is or why I'd need it."
    - "It's not a very high priority for me as yet."
    - "Kaffe.org keeps me busy all the time :("
    - "My coding emphasis shifted away from having time on Classpath before the mediation efforts started"
    - "Nothing. The mediator is a very good and needed service."
    - "Simply time, and that my own contributions to the project have been infrequent since the beginning of this year (due to an academic project I'm working on). As I work more on Classpath, no doubt my contributions will increase. There is no problem with the actual process of doing so other than this."
    - "Too little time and lazyness."

23. Chosing a wiki for the collation of data was an appropriate decision.

I strongly disagree	0%	0
I weakly disagree	0%	0
I weakly agree	45.5%	5
I strongly agree	45.5%	5
I don't know	9.1%	1

24. Locating certain information from the mediation wiki is easy.

I strongly disagree	0%	0
I weakly disagree	18.2%	2
I weakly agree	27.3%	3
I strongly agree	27.3%	3
I don't know	27.3%	3

25. The decision to keep the wiki free of discussions is appropiate.

I strongly disagree	0%	0
I weakly disagree	0%	0
I weakly agree	27.3%	3
I strongly agree	45.5%	5
I don't know	27.3%	3

26. Which technology would you prefer instead of a wiki and why?

    - "A WebDAV repository, preferably with autoversioning. It would make it easier to search and store different types of content."
    - "Having started to use a wiki for one of my own academic projects in about the same time period, I can definately see the advantages of using a wiki and find it a very appropriate medium for this process. With traditional web pages, there is an inherent deterrent to adding material, in that the person who wants to make the changes has to work out how to access and upload a new version, as well as needing to know how to edit the existing HTML. The wiki removes these restrictions and means that, if someone has an idea, they can simply go and add it to the wiki with little fuss. I also second the idea of not putting discussion on the wiki; there are already appropriate conduits in place for this (IRC, the mailing list) and the wiki is then clearer for new developers and interested parties."
    - "I don't know. 1 I think a wiki does a pretty good job of providing a flexible structure to work with."
    - "I think a Wiki is very good for mediation work. With a wiki several people can contribute easily and make the life of the mediator more easy."
    - "The Wiki was nice since it allowed others to participate. But the Wiki didn't really have a "wiki-nature" the pages were a bit long at times. More issues could have been split up into their own separate pages. That would have made it easier to point someone at just one issue. Currently some subissues have horribly long URLs. On the other hand that might have scattered the data too much and wouldn't have given a broad overview. Some more experimentation with the form would be nice. But

the current wiki is nice because it can be adapted easily to other forms or representing or breaking up the issues."

- "Wiki is perfect. I have also used this in other projects and think that Wikis are perfectly suited for bringing together distributed teams."

27. The topics currently managed (developer decisions, first steps, current issues) have been chosen reasonably.

I strongly disagree	0%	0
I weakly disagree	0%	0
I weakly agree	18.2%	2
I strongly agree	45.5%	5
I don't know	36.4%	4

28. Which additional mediation related topics should be managed?

    - "I don't know ATM. Such things should be decided on demand. At the moment it is ok as it is."
    - "I don't know."
    - "I think this is an appropriate set, giving that less dynamic topics are already covered by the static web page (news items and events, etc.) These three adequately deal with previous discussions, newcomers and keeping abreast of developments, respectively."
    - "I'd love to contribute a 'from bug to bug report and patch submission' step by step guide for Kaffe to the wiki when I have time to finish one, and a list of 'other things you can do to help'. I think we could do a better job at engaging the non-technical audience that's willing to help, but doesn't know how yet, for example by providing a consistent narrative for the existance of free runtimes and their advantages over non-free ones. I get asked that one a lot :)"
    - "It would have been fine if the mediator had more agressively added the task list, faq, vm integration guide and GNU Classpath Hacker Guide into the mediation effort. But I only missed a good integration with the task list. Current issues only showed

what was being worked on. But not really what should be worked on if someone had time and volunteered to do it."

- "Most people starting GNU classpath have problems with installing and how to choose a VM suitable for their GNU classpath version (release or –cvs). There should be some more detailed instructions in how to install GNU classpath with the VMs that support using a standalong version of GNU classpath. And it should be descibed which VMs don't support this, have their own copy of GNU classpath and why."
- "Overall architecture, who's working on what, who needs what sort of help, licensing FAQ, schedule and priorities"

29. This space is for suggestions, notes on the survey and/or your answers.

    - "I'm sorry that this survey was kind of a bust for me but I don't really know what the mediation pages are or what they're for."
    - "Mainly an excellent job is being done. I did note that the odd item on the wiki had not been updated since its initial introduction (locales and generics spring to mind, as areas I've worked on), although the mediator is not necessarily to blame for this. I guess there is still some way to go until developers are used to documenting what is said and done more fully."
    - "None so far."
    - "Thanks for doing this. It really was got to get some things "on paper" which I believe have helped some people quickly get an overview and start with the GNU Classpath project (and community). You showed a good style by subjectively writing about the different subjects that we discussed in the project. And I never had the feeling that you took 'sides' on some issue. Maybe that comes naturally to you, but it was really appreciated."

References

[CAH03] Crowston, Annabi, and Howison. Defining Open Source Software Project Success. *Proceedings of ICIS 2003*, December 2003. http://floss.syr.edu/publications/icis2003success.pdf.

[DRAG03] Davor Cubranic, Reid Holmes, Annie T.T. Ying, and Gail C. Murphy. Tools for light-weight knowledge sharing in open-source software development. 2003. http://opensource.ucc.ie/icse2003/3rd-WS-on-OSS-Engineering.pdf.

[Ste00] Steffen Evers. An Introduction To OpenSource Software Development. 2000. http://user.cs.tu-berlin.de/ tron/opensource/opensource.pdf.