

# **Information Management in Open Source Communities**

**Tomás Aguado Gómez**

## **Information Management in Open Source Communities**

by Tomás Aguado Gómez

Copyright © 2005 Tomás Aguado and Telefónica Investigación y Desarrollo

Permission is granted to copy, distribute and/or modify this document under the terms of the *GNU Free Documentation License*, Version 1.2 or any later version published by the Free Software Foundation with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. You may obtain a copy of the *GNU Free Documentation License* from the Free Software Foundation by visiting their Web site (<http://www.fsf.org>) or by writing to: Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

### Revision History

Revision 0.1 8 May 2005 Revised by: TAG  
OpenOffice.org inclusion

# Table of Contents

<b>Preface .....</b>	<b>v</b>
<b>1. MySQL.....</b>	<b>1</b>
1.1. Why Dual Licensing?.....	1
1.1. MySQL Dual License .....	2
1.1.1. Evolution .....	3
1.1.2. Licensing Model .....	3
1.1.3. Model effectiveness .....	3
1.2. Open Source Community Tools .....	4
1.3. Dual Licensing: The Virtuous Cycle.....	5
1.4. Advantages and disadvantages of Dual Licensing .....	6
1.5. Conclusions .....	6
<b>2. OpenOffice.org.....</b>	<b>8</b>
2.1. Historical Background .....	8
2.2. Components .....	8
2.2. Licensing .....	8
2.3. OpenOffice.org.org Organization.....	9
2.2. How to contribute? .....	10
2.2. Joining OpenOffice.org.....	10
2.2.1. Issue Tracker .....	10
2.2.1.1. Finding and Sending Issues.....	10
2.2.1.2. Issue Handling.....	11
2.2.1.3. Issue Tracker and Contributors.....	15
2.2.1.4. Alternatives.....	16
2.2. Sun and OpenOffice.org .....	16
2.3. Conclusions .....	17
<b>References.....</b>	<b>20</b>

# List of Tables

2-1. OpenOffice.org community evolution .....	16
---	----

# Preface

Developing software using an open source community introduces great advantages but poses serious problems. The first aim of this book is to show how these problems are solved by the most known and important communities. In the first part of the book four successful Open Source communities are analyzed. From their organization, desirable and common solutions for open source development trouble will be inferred.

The creation of an open source community is often hand-made and differs a lot from one case to another, our objective is to extract from all.

This analysis will be focused on how companies and enterprises profit from their open source communities, and the tools they put at the community disposal. Each tool will be analyzed with the aim of designing the organization of an hypothetical ideal community.

From the organization of this ideal community, a business model can be extrapolated. The second objective of the book is formalizing the creation of a business model using an open source development community.

Open Source software is modifying the traditional software development model; closed proprietary software business model has been always related to the sale of the product license for everyone who wanted to use the application. Generally, the resulting product is not well adapted to the customers, or if it does, its costs are prohibitive.

Traditionally Open Source development has always been associated with "altruism", but nowadays this myth is to be forgotten, there are several quality proven business models integrated in the Open Source community. But the question still remains, how can a company profit out of something totally *free*.

The solution will be thoroughly analyzed, and examples will be exposed; summarizing the key is *added services*. Besides the application more services can be offered:

- Staff training in using the software acquired
- Personalized applications, and future adaptations
- Migration and Integration services
- Professional Support

Here another question arises again, How can those added services make up for the software development and extra services costs?. This point has to be cleared, because in principle we have in one hand proprietary software development where money is earned mainly with license fees, on the other hand we have free software development, where the software has to be built too but the profits come only from added services.

The solution is reducing software development costs using a development community. The company that creates an open source development community, must follow some guidelines in order to assure normal operation:

- Community maintenance
- Active participation from the company engineers in the community
- Equip the community with tools aimed to assure the quality of the project

# Chapter 1. MySQL

MySQL ([www.mysql.com](http://www.mysql.com)) is the most popular Open Source database. Free, easy to use and continuously improved MySQL has become a widely used alternative to other high cost databases; although it's not the only free database manager its simplicity has taken MySQL to a leading status in free software as well as in commercial environment.

MySQL is developed, distributed, and supported by MySQL AB. MySQL AB is a commercial company, founded by the MySQL developers, that builds its business by providing services around MySQL. MySQL AB makes money on both support services and commercial licenses, using their profits to create new product development and to expand MySQL business. The company was founded in October 2001, with private investments which are used to solidify their business model and build a base of escalable growth.

MySQL AB, like many other second generation open source companies, follows "Dual Licensing" business model. Thus, users may choose MySQL AB products under both commercial and the GPL<sup>1</sup> License. GPL License enables users to download MySQL software (unsupported) for free, modify it to fit their needs and distribute it without paying anything, however GPL users must follow all the GPL rules, so if they use MySQL to develop an application, the complete source code for this application must also be open and available for redistribution. Users that do not want to release the source code for their applications may use the commercial license in which case they get a supported product, choosing between different support tiers.

## 1.1. Why Dual Licensing?

In MySQL there is technically one product core, but two licenses: one for free distribution and one for proprietary use. Dual license differs from pure free software in several ways:

- Development group is not allowed to start competing products from the source code (forks), because copyright is held by the original author developer and changing the product license requires the full ownership of the source code.
- If a software product created using the free license is embedded in another product, the resulting application must be under free license too.

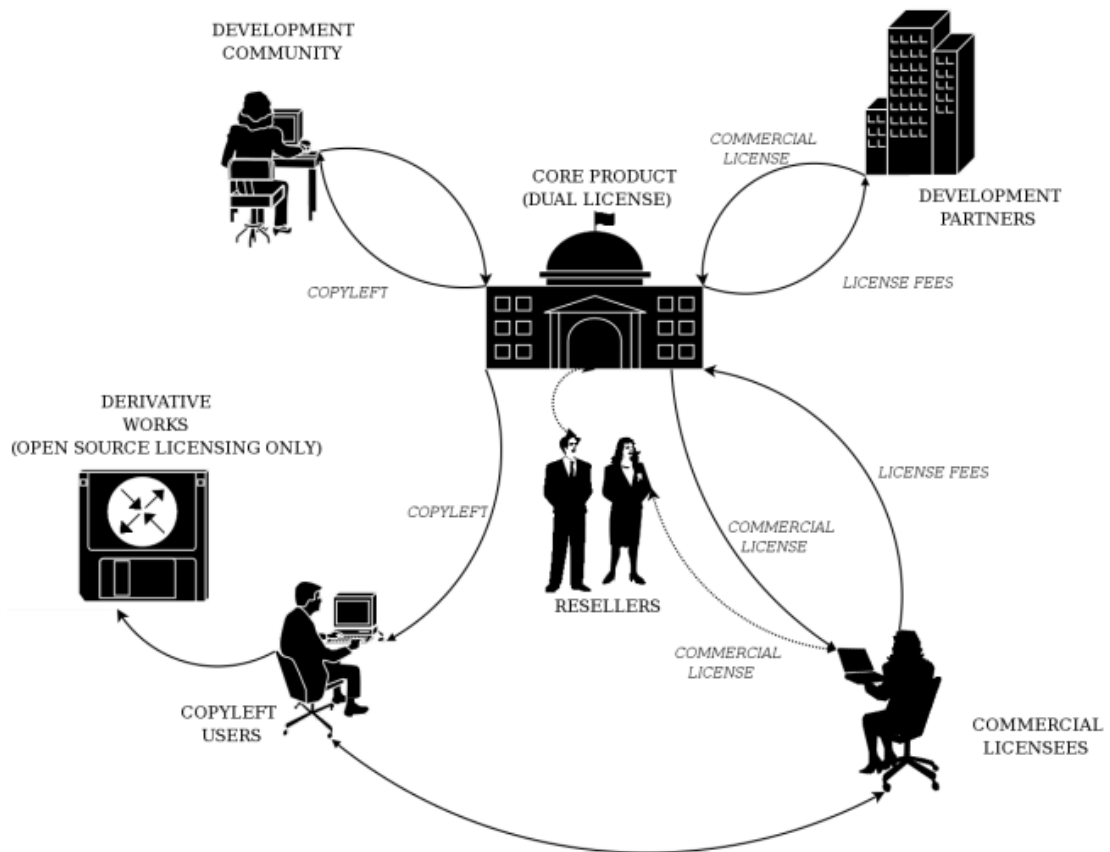
Similar to dual-licensing is triple-licensing where you release your work under three licenses. For example, the Mozilla (<http://www.mozilla.org>) organisation had a project for triple-licensing its web browser under GPL, LGPL<sup>2</sup> and MPL<sup>3</sup>.

By selling a commercial license, companies using dual licenses are able to staff a full-time dedicated development team and are able to provide high quality support even directly from the developers, of course this is a "non-free" service, and it's provided by almost every company of this kind. By licensing products under open source licenses, they are also an active part of the open source community.

This community plays an important role in ensuring the stability and quality of each of those products. The contribution given by this community differs from one project to another. The most common contribution is "beta-testing", open source software developers around the world actively participate in beta testing cycle detecting and fixing bugs.

As a result, the final software product reaches commercial stability more quickly and it's more strongly tested than standard frameworks. Other communities, like *OpenOffice.org* allow development community to add features, contribute to a core module, etc...

Figure 1-1. Dual Licensing business model



End users are divided in copyleft users and customers; the company "dual-licenses" the core product, thus copyleft users may use the software product for free with a copyleft license, and customers may exploit the same core product with a commercial license, sold by the same company who developed the software or by resellers. If a copyleft licensee wants to use the copylefted product in terms which exceed the copyleft license limits, it will be possible for him to buy a commercial license.

As we said, the development community has different degrees of involvement depending on the company which double licenses the software. In MySQL AB, there is not a typical development community, MySQL is developed, supported and marketed by MySQL AB, users may report bugs, test the software or help other MySQL users using the means MySQL AB provides.

Development is directed inside the company, and can hardly be developed by third parties. In 2001, another company tried a fork, but failed as they were not able to control the software development. Nowadays, all contributions are checked and rewritten by MySQL AB developers thus the ownership of the product is not affected

In projects like *OpenOffice.org* users who want to contribute with new code, have two options: JCA<sup>4</sup> or CA<sup>5</sup>. Under the JCA, both Sun and the contributor retain full rights to use, modify, and redistribute the copyrighted work. Under the CA form, the contributor's rights are transferred to Sun and not shared with the contributor

## 1.1. MySQL Dual License

### 1.1.1. Evolution

Almost none of the companies which use double licensed products started using that model, and MySQL is not an exception. At the start, in 1995, MySQL was released under its own license terms. That license allowed limited free distribution and usage of the product with a strong copyleft term on Unix based systems. The business model was essentially dual licensing on Unix-based systems and proprietary on Windows.

Soon, the Linux-based version became very popular on the Internet, and that was the key fact for the future members of MySQL AB (it didn't exist by those years) to turn to the dual license model; in 2000 the free license was changed into GNU GPL on all platforms, and it has remained like this so far.

This decision attracted even more users for the product, and nowadays MySQL has a leading position. In 2001, the company MySQL AB was founded to own the copyright of the database with its partners.

### 1.1.2. Licensing Model

As we said before MySQL AB products are licensed by a commercial license or by GNU GPL. Commercial license is necessary when you intend to distribute MySQL AB software, for example:

- A packaged application contains MySQL, and customers install it in their own machines
- MySQL is not included in the packaged application but it is required by the software to work properly
- Selling Hardware including MySQL

### 1.1.3. Model effectiveness

Companies like MySQL or Trolltech have achieved success in establishing a business model based in dual licenses. The main income for MySQL AB comes from proprietary license sales, but there are many more income sources for MySQL AB:

- Commercial high quality support for MySQL provided by the MySQL developers themselves.
- Consulting services.
- Advertising on their web page.
- Partnering and training programs.

By commercial support, customers get their problems solved directly by the engineers who code MySQL, of course it is a non-free service but this system has turned out to be good because customers receive suggestions they can't get from ordinary support staff.

MySQL AB has a worldwide partner program that covers training courses, consulting and support, publications, plus reselling and distributing MySQL and related products. MySQL AB Partners get visibility on the <http://www.mysql.com/> web site and the right to use special versions of the MySQL trademarks to identify their products and promote their business.

MySQL offers a set of training courses to build database solutions and offers the chance of getting a MySQL certification registering for and special exam. Thus, MySQL incomes are not based only in

commercial licenses but in many other aspects, that makes MySQL AB a stronger and more independent company.

## 1.2. Open Source Community Tools

We have different mailing lists like in many other free software organizations. Problems posted in these lists are solved by MySQL employees but almost everytime by other MySQL users, so thats the way MySQL users get in touch:

- Announce
- Mysql
- Bugs
- Internals
- Mysqldoc
- Benchmarks
- Packagers
- Java
- Win32
- Myodbc
- Gui-tools
- Cluster
- Dotnet
- Plusplpus
- Perl
- MaxDB
- MySQL Cluster

As we said most answers on the mailing list come from volunteers, and sometimes problems are not solved because questions are not made the right way. The kind of answers to technical questions depends as much on the way questions are asked as on the difficulty of developing the answer; a nice and useful document explains how to ask questions in a way that is likely to get a satisfactory answer, written by Eric Raymond and Rick Moen it is available at <http://www.catb.org/~esr/faqs/smart-questions.html>.

In addition to the various MySQL mailing lists, it's possible to find experienced community people on IRC<sup>6</sup>, MySQL AB recommends `#mysql` channel in freenode network

The latest free community support resource are the forums at <http://forums.mysql.com>, as the mailing lists they are grouped in general categories:

- Migration
- MySQL Usage
- MySQL Connectors

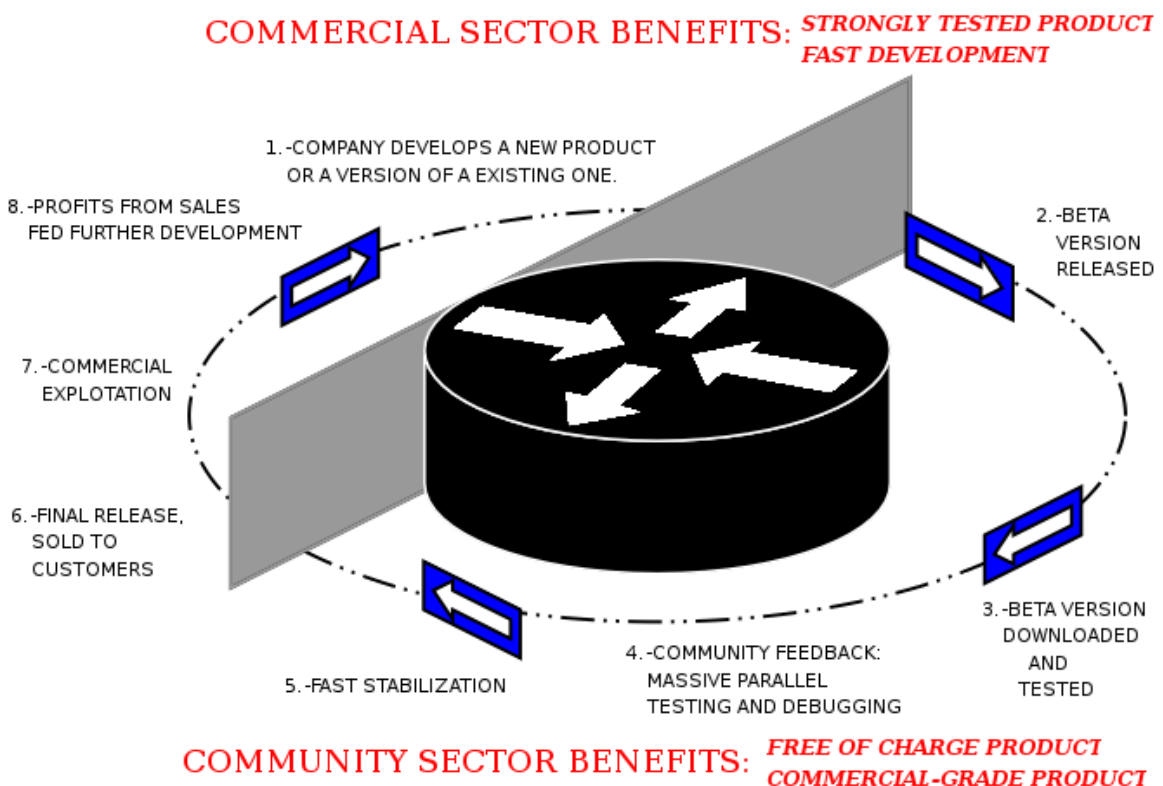
- MySQL Technology
- Business

In <http://forums.mysql.com> we can use almost 50 different web based forums, where end users and developers get in touch and sort out different problems which vary from MySQL installation in different operating systems to security and replication of MySQL database.

Forums are free, but they are not suitable to solve critical problems because this system is far from "real-time" and it is not guaranteed. For better quality support one option is to pay for support from MySQL AB, which will put you in direct contact with MySQL developers.

### 1.3. Dual Licensing: The Virtuous Cycle

Figure 1-2. The Virtuous Cycle



This cycle is based in "Quid pro quo"(something for something) between commercial and open source segments. In the first stage a company, for example MySQL AB releases a product under GPL license, this fact has an immediate effect as open licenses do more for a software project than . The GPL allows MySQL AB to penetrate millions of sites that would never have otherwise known about it. The fact of releasing the product under GPL is what commercial segmente "gives" to open source segment.

Having thousands of users is a great advantage for every software company; even if only 1% of the users take part in bug fixing, testing or developing new features (if possible), as this testing is done parallelly, this development stage is shorter and more detailed, but it's of paramount importance that the huge amount of contributions is processed correctly.

Once a tested and almost free of bugs product is available, the company has a stable product which is released under a commercial license too, the company earns money from the commercial software fees. Profits obtained are invested in new releases, and the cycle starts again.

## 1.4. Advantages and disadvantages of Dual Licensing

**Disadvantages:** Some developers do not like this business model, because they do not want to hand over the copyright of the code they write just to let the company earn money at the expense of their work. It's confusing for readers and for other contributors. It puts up a roadblock to continuing contribution, and makes them read lots of documents to figure out what their rights and responsibilities are. Dual License model has been criticized by free software purist because a company that has bought software core under a commercial license will be able to adapt it and left the product closed, because commercial licenses do not force the company to distribute the source code.

**Advantages:**

Almost all the advantages of dual licensing have been described along the chapter. The engine that makes the Dual License Virtuous Cycle spin is that everyone involved (both commercial and Open Source participants) get more from using Dual Licensed products than what they put into them.

**Commercial users:**

- Battle Tested Software
- High Quality Support

**Open Source users:**

- Free Software
- HNew Open Source Communities
- Open Source Software is revitalized

## 1.5. Conclusions

Dual license does not only fill the gap between free and commercial licenses, it gathers the advantages of both kinds of licenses, and as a result of this dynamic unity more improvements are exposed. Benefits coming for commercial customers make possible to afford to develop and improve the product involved in the Virtuous Cycle thanks to the staff who makes up the company.

MySQL delivers more than 40,000 new product copies to their user community every single day. As a result of the huge community, their software undergoes rigorous and extensive "battle-testing" and bug fixing. This refinement is carried out in parallel and costs in this stage are enormously reduced for MySQL AB.

Using dual licenses, the user is free to choose which one is best for him, he can switch from one to the other and even mix the two models. Technically you get the same software in both cases, the only differences between free and commercial versions are the legal implications and the assurance of service.

MySQL as strong example of second-generation open source companies, has established sustainable business on open source software. This business model has gained support in the IT industry, MySQL provides dual-license software to some of the largest customers in the world including NASA, IBM, Google, Sharp...

MySQL AB is organized virtually with people in a dozen countries around the world. They communicate extensively over the Net every day with one another and with their users, supporters, and partners. This data flow and the contributions coming from cooperative users must be managed efficiently. Information management is crucial, not being able to administer this field is disastrous as we saw in the fork attempt in MySQL.

MySQL development is directed by MySQL AB staff, users community may fix bugs and test software, but all the code is written by MySQL AB staff and almost all the dataflow is managed internally, mailing lists and forums apart. This centralization has advantages and disadvantages. As all the code is processed internally by MySQL dataflow management is simple but this centralization makes MySQL development pace slower than other distributed development projects.

Thus, companies like MySQL solve pressing problems present in open and commercial licenses combining and applying different license policies; but many other important problems are not directly faced. The most important is how to channel and license the contributions coming from end users; MySQL seems to solve this reducing the cooperation options given to the development community, and that's wasting smart and free contributions.

A good choice is investing people and money in successfully setting up a set of tools which make easy for end cooperative users to collaborate, sending whatever they want to contribute with, and making easy for main developers to "process", adapt and include those new features.

This is maybe the heel of achilles of MySQL, the more tools are given to the development community, the more contributions are received, and being able to manage those contributions means faster development and extra features for both open source and commercial users, this means money for MySQL AB and higher quality applications for everyone using the software.

## Notes

1. GNU General Public License
2. GNU Lesser General Public License
3. Mozilla Public License
4. Joint Copyright Assignment
5. Copyright Assignment
6. Internet Relay Chat

# Chapter 2. OpenOffice.org

As OpenOffice.org says in their mission statement, the aim of the project is *To create, as a community, the leading international office suite that will run on all major platforms and provide access to all functionality and data through open-component based APIs and an XML-based file format.*

OpenOffice.org can be used in GNU/Linux and many Unix-like operating systems, Microsoft Windows, Solaris, Mac OS X and OS/2.

## 2.1. Historical Background

In 1986 at the age of 16 Marco Börries founded StarDivision, this company started to develop StarOffice, to compete with Microsoft Word. In 1996, they made a Linux version of their product available as a free binary-only download. It had some key advantages over the other word processors then available for Linux.

The company and the rights to StarOffice were acquired by Sun Microsystems in 1999 as Sun was seeking to compete with Microsoft Office too, in June of 2000 Sun offered StarOffice 5.2 as a free download for personal use. Sun open-sourced almost all the Star-Office code, the project based in that source-code was named OpenOffice.org and is developed by both Sun and the open source community. Similar exercise to MySQL, OpenOffice.org products are dual-licensed.

When Sun wants to release the commercial version, they take the current version of the OpenOffice.org code base, integrate proprietary and third-party code modules, and sells the resulting package commercially.

OpenOffice.org uses a dual license strategy for the source code and a separate documentation license for most documents. The source-code licenses are the GNU GPL and the SISSL<sup>1</sup>. The document license is the Public Document License.

## 2.2. Components

OpenOffice.org works transparently with many file formats, including those from Microsoft Office; this one-way compatibility has made OpenOffice.org a rival to take into account for Microsoft. Each of the components of OpenOffice.org is able to import its counterpart of Microsoft:

*Writer* is a word processor similar to Microsoft Word, it includes extra features like direct export to PDF from a writer document.

*Calc* is similar to Microsoft Excel, as *Writer* it includes extra components and imports Excel file format.

*Base* is a relational database, including support for Microsoft Access, MySQL and ODBC.

*Impress* is Microsoft PowerPoint counterpart, as extra functionality it can export to Macromedia Flash.

*Draw* is a vector graphics editor; as a free alternative to programs like CorelDraw it offers a powerful and easy tool to create 3d shapes, Bezier Curves...

## 2.2. Licensing

Compared with MySQL, OpenOffice.org offers a wider set of tools to get and process contributions, this enables OpenOffice.org to allow contributions not only in bug fixing and testing, but in adding new code to the OpenOffice.org suite.

OpenOffice.org is dual licensed, thus if a commercial version includes end users contribution, Sun must have somehow copyright of those contributions. This problem is solved requiring everyone who wants to contribute source code or other materials which are intended to be integrated with the OpenOffice.org.org product to fill the JCA<sup>2</sup> before contributing. Once fulfilled, under JCA both the contributor and Sun holds full rights over the copyrighted work.

This shared copyright prevent contributors from losing the copyrights over the work they have done and Sun can redistribute the code under commercial or free licenses.

## 2.3. OpenOffice.org.org Organization

In comparison with MySQL, OpenOffice.org exposes an estructure where every member of the community can take part in project doing whatever he wants to, thus decisions must be supported by most of the members of the community. OpenOffice.org general project is composed of subprojects. Every subproject counts on project members, contributors and one leader, a member of OpenOffice.org community may only lead one project. Subprojects are divided in three main categories: *Accepted*, *Native Lang* and *Incubator*.

OpenOffice.org.org is governed by the CC<sup>3</sup>. This council arbitrates in conflicts, and offers a forum for community users. All the discussions of the CC are public and available for all the OpenOffice.org project members.

CC also is responsible of resource allocation and strategic planning. Thus CC must research suitable fund sources, and work with Sun to design and start the terms of this relationship for the benefit of the OpenOffice.org.org project.

CC must also manage relations with sponsors and the public, acting as the community voice for Sun and for the general public and media. CC must also administer relationship not only with donors and external sponsors, but with other Open Source projects too.

Another important project in OpenOffice.org.org is QA Project. The aim of this project is to assure the quality of all the OpenOffice.org version released. QA makes sure every code line of OpenOffice.org makes what it is supposed to do, thus QA Team uses tools to process all the contributions and bugs report/fixing arriving from the community. They also prioritize all the issues coming from the community, thus developers can focus on the most urging problems.

OpenOffice.org.org comprises many public projects, there are three main categories where we can find an active project:

- Accepted
- Native-Lang
- Incubator

*Accepted* projects stage includes almost all the technical projects and Marketing. *Native-Lang* houses those projects involved in the localization and support of OpenOffice.org.org, the aim of *Native-Lang* is to create a community, where OpenOffice.org people involved in OpenOffice.org development can share information and resources, no matter where they are from.

*Incubator* space was created for users to test ideas, whatever they want to create: coding, documentation,... it works like a gateway for projects hoping to become part of the *Accepted* category; this category is very important because of the high rate of mortality of open source projects in early stages.

In OpenOffice.org all global decisions are made democratically. Each *Accepted* project leader has one vote, and *Native Lang* project has has one vote on the whole.

## 2.2. How to contribute?

OpenOffice.org offers users to contribute in whatever they want to, differently from MySQL, users can add new code and develop new modules of OpenOffice.org, thus new tools are needed to process and integrate those contributions.

### 2.2. Joining OpenOffice.org

Users can join de the development community as subscribers to the different mailing lists, as members of the overall project, or as members of a particular project. First of all potential contributors must subscribe to OpenOffice.org project as registered users giving both username and password to access their personal start page.

Subscribing and registering are not the same thing, subscribers may only subscribe to OpenOffice.org mailing lists if they are not prepared to join and help, registering gives full access to contribute.

Registered users can log in OpenOffice.org.org home page and enter their start page. This start page enables users to be full contributors for OpenOffice.org project. The first concept that comes up once in your personal start page is *issue*.

An *issue* is a bug or a problem discovered in both OpenOffice.org.org applications and infrastructure. Users cand report and query issues. An important tool used to track issues used by OpenOffice.org.org organization is *Issue Tracker*(formerly called IssueZilla).

#### 2.2.1. Issue Tracker

OpenOffice.org Issue Tracker is based in Mozilla bug tracking tool: BugZilla. Issue Tracker, developed by *Collab.net* is more generalized, thus contributors may track not only bugs but many other different kinds of activities related to the project:

- Bugs
- Enhancements
- Features
- Tasks
- Paches

Issue Tracker is nowadays used but old-fashioned, it was hosted at tigris but no more releases have appeared, and the project is no longer there. It will be described as an example of efficient contributions processing.

All the development documentation for each project is held in a unique database. For each project, users have a link *Issue tracking* that displays the database issues for that project.

##### 2.2.1.1. Finding and Sending Issues

When a user founds a bug, for example, before submitting it he should search in the project database if an already reported issue handles the bug he is trying to submit. Every issue has a unique issue number so you cam jump directly to it if you know the number. If the contributor doesn't know this number, he may carry out more general searchs, specifying a search filter based on keywords defined by him in both the Summary and the Description of the Issue.

If the bug is not handled by any issue, contributors may wait a little more before creating and submitting the new issue. Users may check if the bug or patch he is trying to report is not fixed in a more recent version, OpenOffice.org may have evolved from the version the issue is reported to.

If a bug found is not managed by any issue, and it was not fixed in the latest version of OpenOffice.org, it is time to submit a new issue. There are some rules that must be followed for a efficient issue processing:

- One problem - one issue
- Provide a meaningful summary
- Provide step-by-step descriptions
- Provide sample documents, if possible
- Use Attachments reasonably

Issues must only include one problem, thus they are handle atomically and problems are not lost in the way, as issues may be "attacked" by different users, the more problems included in an issue, the more difficult is the issue to get solved

If both the description and the the summary are described specifically, other contributors may distinguish between different issues, make sure that the problem related by the issue isn't yet handled, and judge the importance of the issue.

It's important to describe exactly the workflow preceding the bug, doing this the user moves the problem he is trying to describe to the correct context, so other contributors trying to solve that issue will surely save time. If the problem is related to a particular document, users should attach it to the issue if it is not too large.

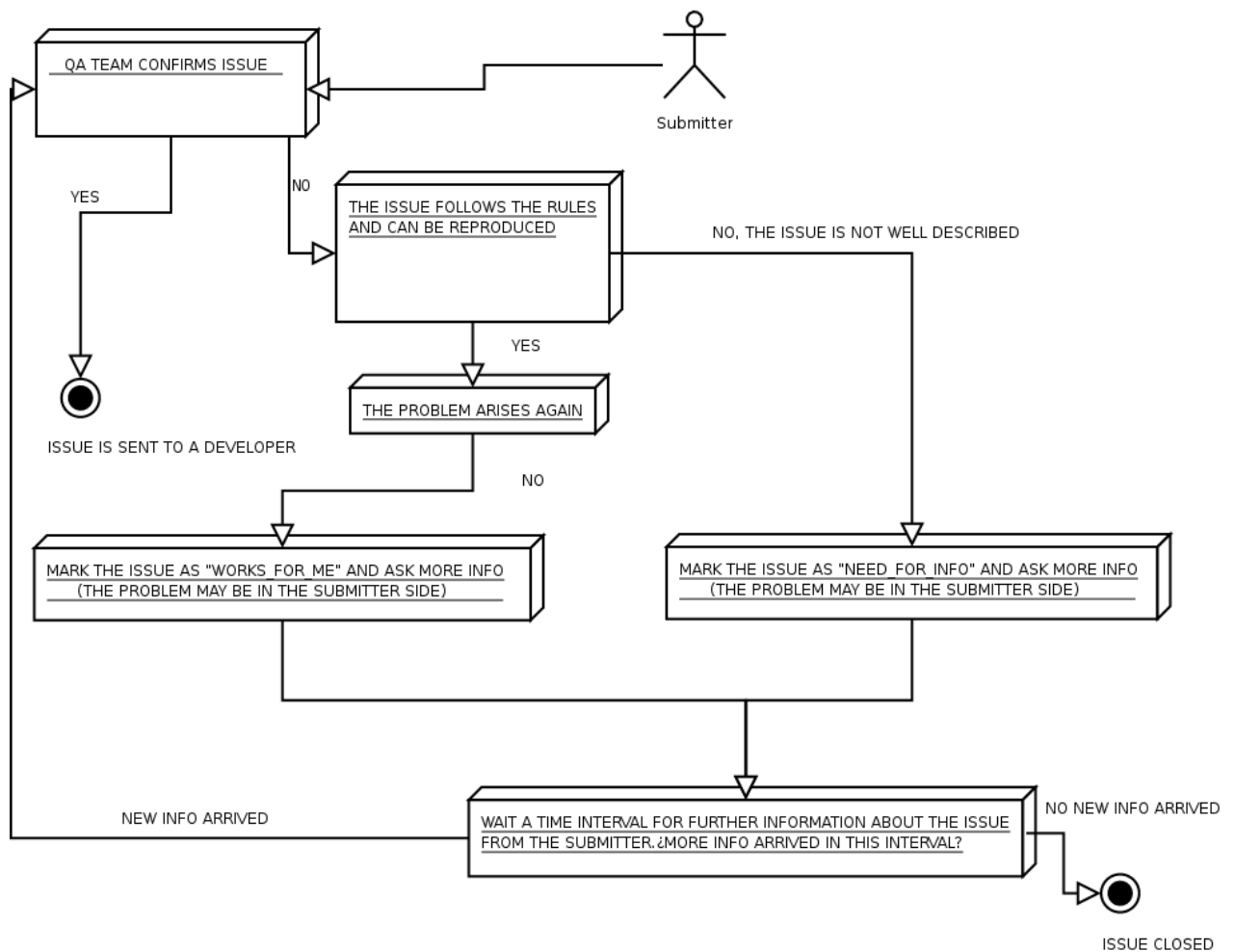
### 2.2.1.2. Issue Handling

Once an issue has been submitted it must be "processed"; first of all the QA Team must ensure that the issue follows the rules mentioned above; if it does, the bug jumps to the fixing and verifying stage.

When the user submits an issue, the first stage to cover, is being confirmed by the QA Team if necessary. If an issue is reported by someone who does not have "Project Issue Tracking - Change" permission, another user having this permission must confirm this issue, until this is done the issue has the *UNCONFIRMED* label as status. If the issue can be processed directly, the status is changed to *NEW* and it is sent to the developer directly, if it doesn't, it must be checked that the issue is well-formed.

Bad descriptions and problems that do not reemerge require further information from the user who submitted the issue, if there's no more info coming from the contributor to process the issue correctly, the issue is closed.

Figure 2-1. Issue Handling



Issue handling in OpenOffice.org.org, generalized it could be applied to any community

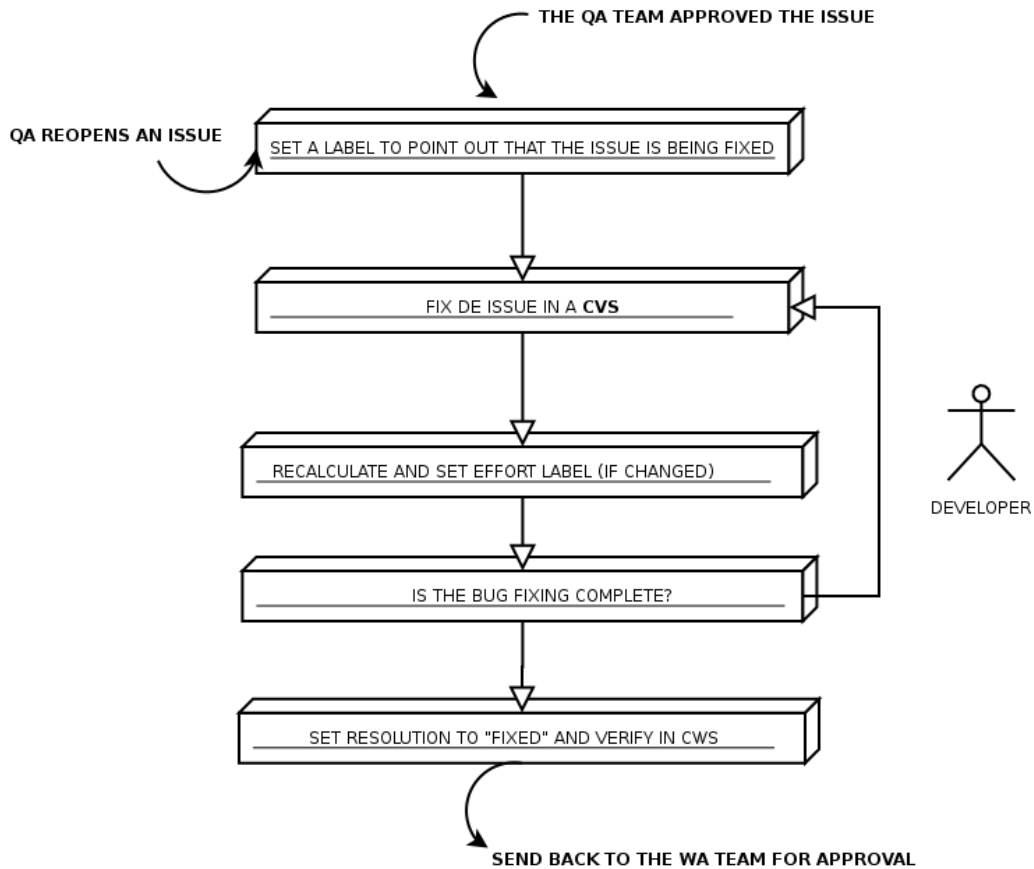
Here a question may be posed, Who assigns issues?. Having "Project Issue Tracking - Change" permission enables a member of the community to validate issues, create an issue directly with the *NEW* label and to assign issues to other contributors, the contributor may accept to solve the issue or he may reassign it to other developer, everytime this is done the status label of the issue must be changed to *NEW*.

Let's deal with the status labels in depth, we have different kinds of labels with different meanings, *NEW* and *UNCONFIRMED* labels are explained above, both are referred to opened projects like *STARTED* (the project is assigned but not yet resolved) and *REOPENED* (the issue was once resolved, but the correct solution was not given)

Solved issues have status labels too, *RESOLVED* (developer resolved the bug and is waiting for QA approval), *VERIFIED* (QA approves the resolution) and *CLOSED* (as we saw before, the issue is dead). When a issue is closed, it has attached a resolution value, *FIXED* means that issued has been tested and checked, *INVALID*, *WONTFIX* (irreparable issues), *LATER* (the issue will no ve fixed in this version of the product, *REMIND* (means the same as *LATER*, but focused on the fact that in coming version of the product if could be fixed), *DUPLICATE* (the problem described by the issue is already managed by an existing issue and *WORKSFORME* which means that doesn't emerge following the steps described by the

contributor.

**Figure 2-2. Issue Handling**



Issue handling in OpenOffice.org.org, generalized it could be applied to any community

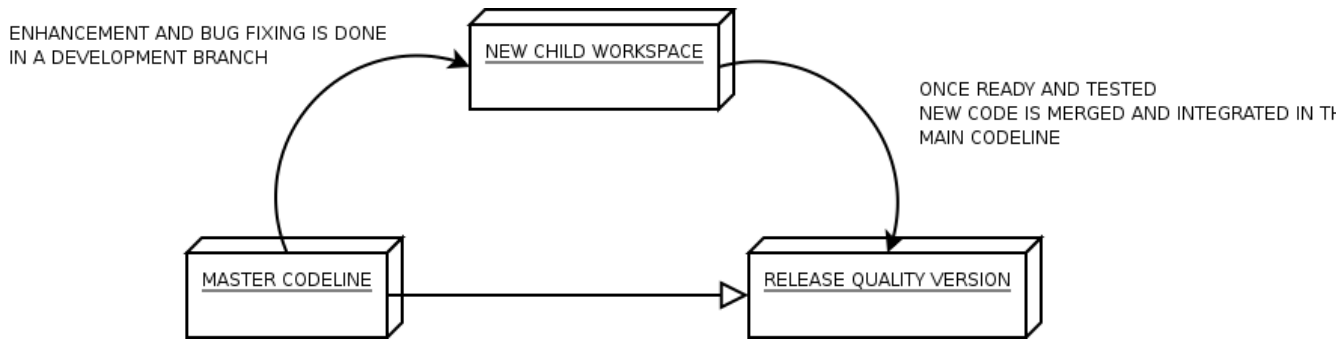
Once the QA Team accepts the issue, what comes to the developer is a precise and complete description of what was wrong in OpenOffice.org suite, and what is even more important for him, that there's a certain problem behind the issue.

Once the developer has accepted an issue, he must change the label of the project to *ACCEPTED*.

OpenOffice.org uses Control Version Systems to carry out parallelly the development of the feature or bug fix we are trying to develop. Releases of OpenOffice.org are implemented on CVS branches, development of OpenOffice.org more significant releases is done on head of the CVS tree, while maintenance of older version is done on branches, which are called OpenOffice.org codelines or master-branches. New versions in this master-branches do not appear everyday, but within regular milestones.

OpenOffice.org is a large project, up to a hundred developers are working on it simultaneously, and the aim of OpenOffice.org is to have always a milestone version with assured quality ready to release, regarding this, thus no commits are made to the master-codeline. Adding new features or bug fixing must be implemented in a *development brach*, this means that all the changes are made on a cvs branch, and once the fix or the feautre are completed and testes on at least two of the main platforms supported by OpenOffice.org, it is merged with the source code again, these branches are called environment child workspaces

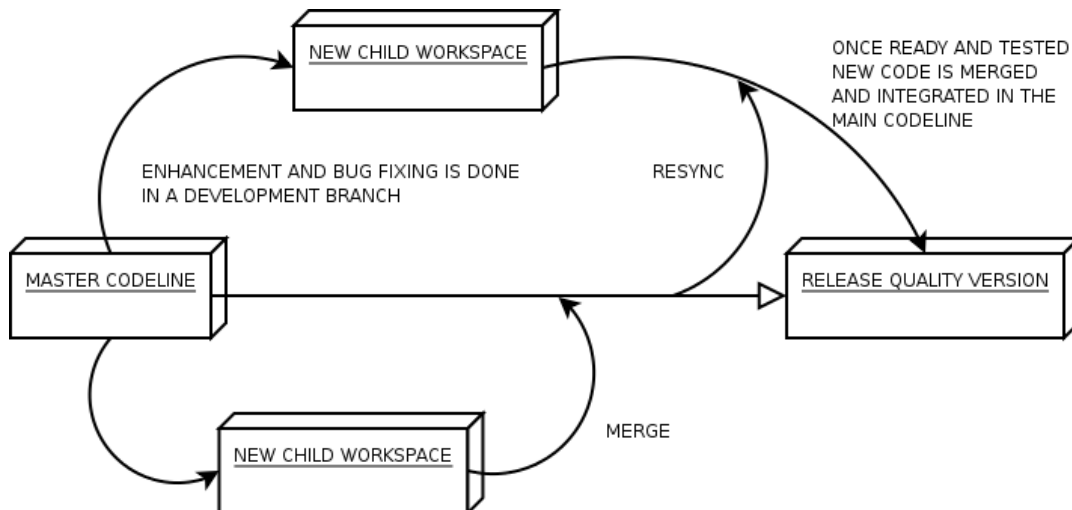
**Figure 2-3. Using CVS in OpenOffice.org**



New code in OpenOffice.org is not committed to the main codeline until its finished and tested

As seen before, OpenOffice.org is a large project and many development branches are kept in parallel over a single master-codeline. This means we can have a common problem with the CVS called "repeated merges". This problem is solved with a resynchronization command; this command assures that all conflicts will be solved not in the master-codeline but in the development branch (child workspace), thus main-codeline quality objective is met.

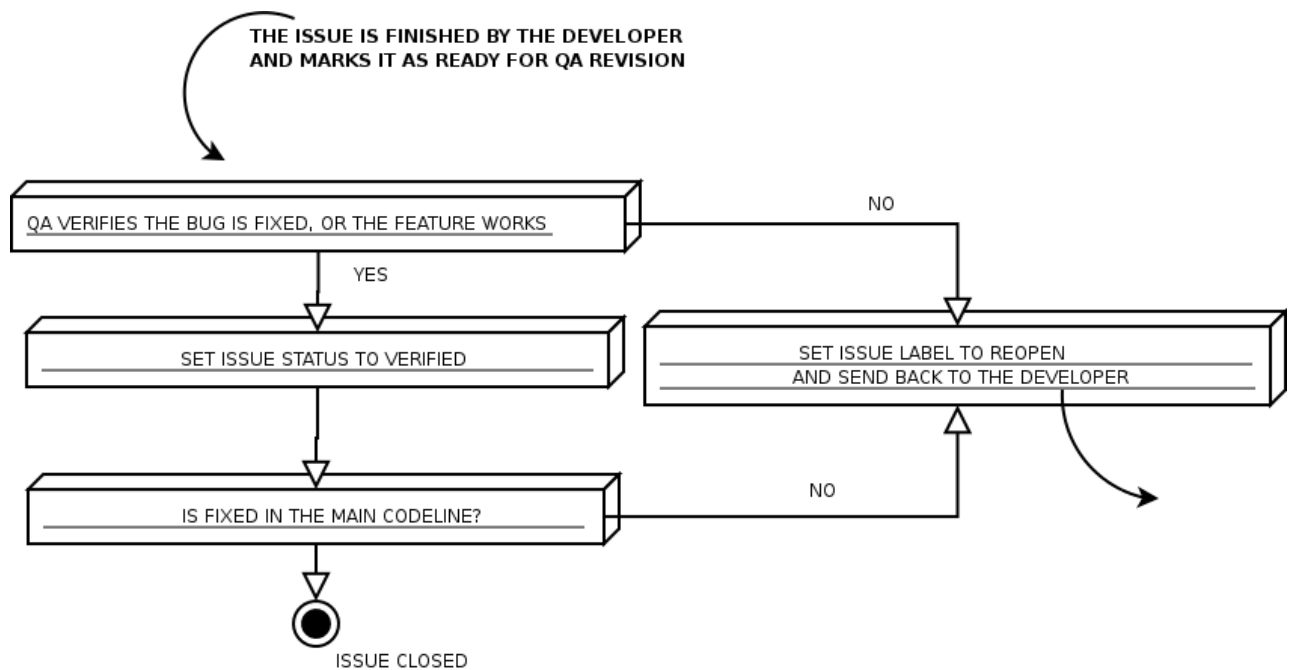
**Figure 2-4. Resynchronization**



New code in OpenOffice.org is not committed to the main codeline until its finished and tested

When the developer is finished with the work he is doing in the child workspace and he has tested and verified the code, his work is reintegrated in the main codeline and the log history of the branch is written in the trunk.

Figure 2-5. QA final verification



QA validates the enhancement or the fix

QA must verify if the new code works, once checked QA team marks the issue as closed and verified. Notice that the code must work in the main-codeline, and in the next release of OpenOffice.org so the fix is available for all end users, if it doesn't in both cases the issue is forwarded to the developer, and the cycle starts again.

### 2.2.1.3. Issue Tracker and Contributors

When a contributor joins a project that uses IssueTracker, he gets automatically an IssueTracker account to access and submit issues in the scope of the project he has joined. Each registered user can customize his IssueTracker preferences. One important property desirable in every bug-tracking tool and present in IssueTracker is automated change notification. Contributors can decide whether they are notified or not when a change is made to an issue related to him. This relationship between developers and issues can be established by several ways: submission, ownership, voting or listed in the CC. Notification options can be configured, and the possible values are:

- Receive email notification of all the changes made to the issues related to the contributor
- Receive email notification of all the changes made to the issues related to the contributor except those made by the contributor himself
- Receive email notification of all the changes made to the issues which the user is listed on the CC line
- No email notifications at all

Another filtering parameter are other users. This way contributors can receive email notifications of the changes made by other contributors. This filter can be used to track the activities of a project member or a group of members.

The last user related property are tracking permissions; permissions on a project depend on the role the contributors is playing:

- Observers can create new issues but modify only those. They can comment any issue
- Developers can modify all issues, but only content or code, if they want to alter other characteristics they must be granted additional permissions
- Project Initiators have full access including changing other contributors permissions

#### 2.2.1.4. Alternatives

As we said before, IssueTracker (IssueZilla) is not further developed, but it has been chosen like a good example to show how to process the information coming from the community. So, What other trackers are available?. First of all we have *Bugzilla* (IssueTracker's parent), it is open source bug system implemented using freely available open source tools, this fact has made of Bugzilla the most important and used bug tracker, available in [www.bugzilla.org](http://www.bugzilla.org).

Bugzilla design principles have been the base for other open source trackers:

- Developed with Open Source and to serve Open Source applications
- Maintain speed and efficiency at all costs
- Follow standars wherever possible

Another implicit principle is to concentrar Bugzilla's development on being a bug tracker. As seen before, BugZilla falls short if a project wants to manage all the contributions coming from his community, thus an option could be modifying BugZilla's code to generate a more general tracker, the same way IssueZilla was born

Lately a new option has turned out, *SCARAB* hosted in [scarab.tigris.org](http://scarab.tigris.org), a highly customizable *Artifact* tracking system. An *Artifact* is the next stage after issues. Issues in IssueTracker were a closed set aspects where the community could contribute. *SCARAB* has fully and free customizable and unlimited numbers of "Issues", this way they are called *Artifacts*.

Thus, *SCARAB* is a tool that can be tuned to track and managed not only bugs, but whatever we want to community to contribute with. It is under heavy development now, but soon it will be a essential tool in every serious open source project.

## 2.2. Sun and OpenOffice.org

OpenOffice represents one of the larger enterprise-sponsored open-source projects in the world. Sun is the enterprise behind OpenOffice.org, but What is Sun role in this project?.

As seen in previous chapters Sun initiated OpenOffice.org project donating the source code from StarOffice and creating OpenOffice.org open-source community. Sun engineers develop OpenOffice.org code and contribute with the code to the community. This effort has created a serious alternative to Microsoft Office, the community development power has increased from OpenOffice.org foundation in 2000.

**Table 2-1. OpenOffice.org community evolution**

<b>OpenOffice.org</b>	<b>2001</b>	<b>2002</b>	<b>2003</b>	<b>2004</b>
Downloads	5.000	1 Million	8.5 Million	+20 Million
Mirrors	0	6	64	+100
CD Distributors	1	12	70	+250
Community Size	3.500	70.000	107.000	+150.000
Native Lang Projects	1	6	38	60
Weekly Posts	600	1500	3000	3500

OpenOffice.org is growing at a high rate, the infrastructure deployed to process all the contributions given by the community have made it possible.

But, what are the differences between StarOffice and OpenOffice.org?, how does Sun profits from the community?. OpenOffice.org shares the code base with StarOffice, what makes StarOffice different from OpenOffice.org is support and third party licensed software that can't be included in OpenOffice.org, as Sun is not allowed to open source it.

Microsoft Office has been the leading office suite for decades, thus Sun realized it was no sense in developing a new and different office suite because most of the potential clients would desist in buying Openoffice.org suite, as they would have to spend great efforts in training their staff on their own.

The solution was obvious, OpenOffice.org from its very beginning has been developed as a free replacement of Microsoft Office. It tastes and smells like MS Office, but it is open and free. But where is the Bussines model here for Sun?. Sun sells its commercial version of OpenOffice including third party proprietary software as StarOffice. OpenOffice source code is developed by the community to a large extent, Sun can invest big part of the efforts in support and training for consumers and businesses, the main handicap when migrating from a deep implantated suite, like MS Office.

Along with this, Sun includes in the "pack" migration and deployment of the new product. Thus, every aspect of a hypothetical migration from MS Office is covered by Sun. Why should a company change his suite? For a fraction of the cost of a common office suite, you get training for your staff, a free and supported product, migration and deployment from the previous system, It would be worth even if it costed the same as an MS Office license.

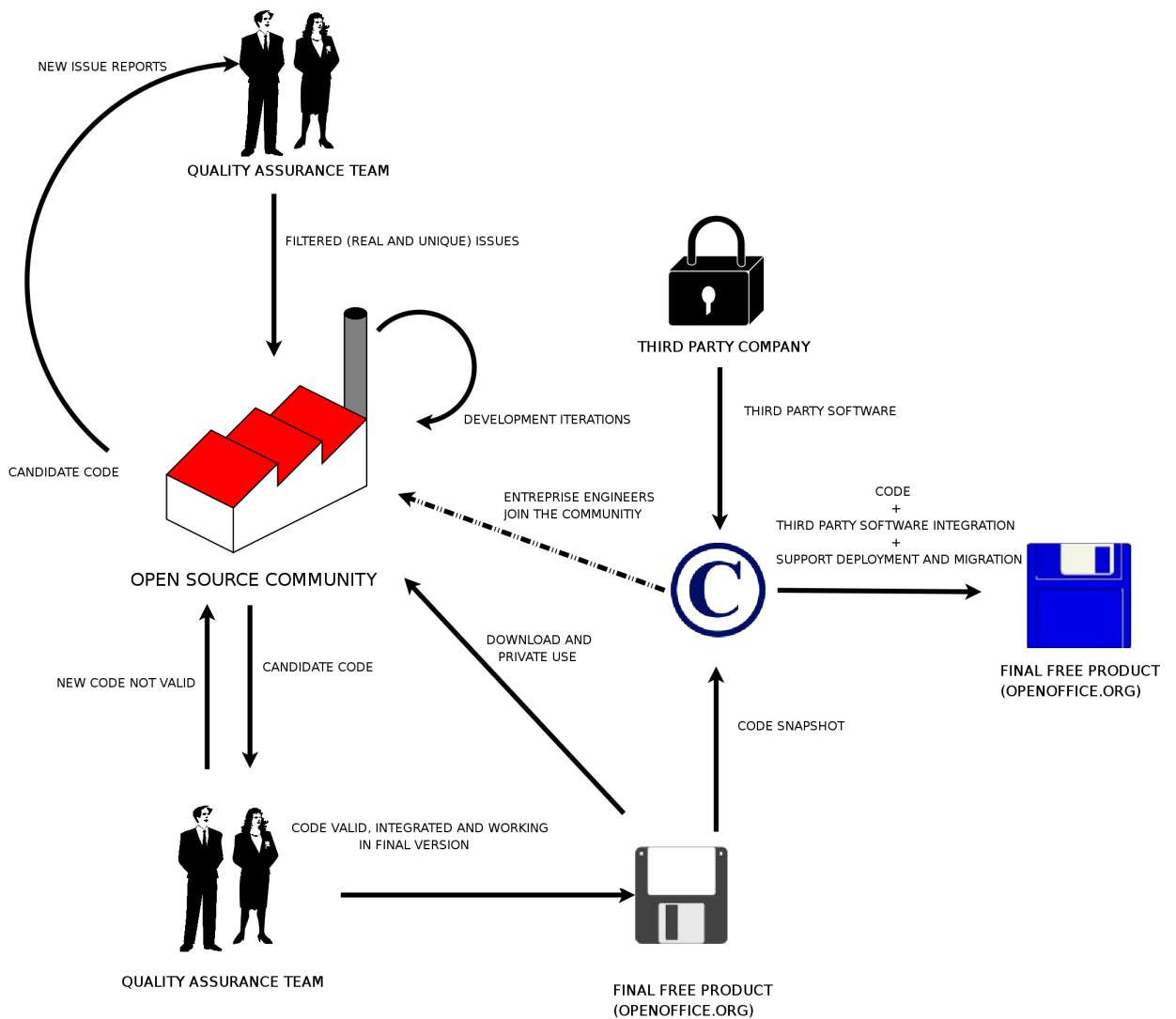
## 2.3. Conclusions

OpenOffice.org has improved from its very beginning, nowadays it is an alternative that must be taken into account. There are several desirable characteristics for creating an open source community that can be extracted from OpenOffice.org organization.

OpenOffice.org development is centralized in the community, its role is far more important than in MySQL, where contributors may only report bugs. This community-focus strengthens and improves the advantages seen in community development, like parallel and intensive test and bug fixing (in this case carried out by the community itself). What comes out in OpenOffice.org is parallel development. Tools must be given to the community in order to organize, prioritize and process contributions. This point is crucial, the more tasks the community does, the more and better tools mus be given to it.

When multiple developers are involved in a project the quality and operation of the software must be assured, even more in distributed projects. In OpenOffice.org we have a clear example of this in the QA Project, which examines every modification or improvement before validating the new version.

Figure 2-6. OpenOffice.org structure



Projects like QA works both in the frontend and the backend of the community. In the frontend they work as a filter to prioritize and eliminate all unnecessary issues. Developers in the community work only in real and unresolved bugs or improvements. In the backend quality assurance projects validate all the community contributions before including the modified version and validating the fix and closing the issue.

Projects like QA are absolutely necessary in active an participative communities, all the work done must be checked and the community developers must be prevented from wasting time in already solved problems or in user-side problems.

## Notes

1. Sun Industry Standards Source License
2. Joint Copyright Assignment

3. Community Council

# References

- Monera Daroqui, Fernando. "Modelos de negocios viables alrededor del Software Libre". November, 2002  
URL: <http://www.opensistemas.com/prensa/clipping/>
- Sanchez Mariscal, Alvaro. *Modelos de negocio basados en Linux y software libre* February, 2004  
URL: <http://es.bea.com/beanews/2004/febrero/index.jsp>
- Hammerbeck, George. *The business case for free software*. December, 2002  
URL: [http://builder.com.com/5100-6401\\_14-1046041.html](http://builder.com.com/5100-6401_14-1046041.html)
- MySQL staff. *The MySQL AB Company*. December, 2002  
URL: <http://ftp.iranscience.net/pub/databases/mysql/company/>
- Active-Venture.com. *Overview of the MySQL Database Management System*. June, 2003  
URL: <http://mysqld.active-venture.com/What-is.html>
- Active-Venture.com. *Overview of MySQL AB*. June, 2003  
URL: [http://mysqld.active-venture.com/What\\_is\\_MySQL\\_AB.html](http://mysqld.active-venture.com/What_is_MySQL_AB.html)
- Active-Venture.com. *The Business Model and Services of MySQL AB*. June, 2003  
URL: [http://mysqld.active-venture.com/MySQL\\_AB\\_business\\_model\\_and\\_services.html](http://mysqld.active-venture.com/MySQL_AB_business_model_and_services.html)
- Sleepycat Software. *Leading Open Source Software Companies MySQL AB, Sleepycat Software and Trolltech AS Prove Strength of Dual-License Model*. March, 2004  
URL: <http://www.sleepycat.com/company/releases/040316.shtml>
- Sleepycat Software. *Leading Open Source Software Companies MySQL AB, Sleepycat Software and Trolltech AS Prove Strength of Dual-License Model*. March, 2004  
URL: <http://www.sleepycat.com/company/releases/040316.shtml>
- BarbcockCharles. *Open Source, Part 2*. March, 2004  
URL: <http://www.informationweek.com/story/showArticle.jhtml?articleID=18402795=1>
- BarbcockCharles. *Unplugged: Mårten Mickos, CEO MySQL AB*. July, 2003  
URL: <http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2914410,00.html>
- MySQL AB staff. *About MySQL AB*. January, 2005 URL: <http://www.mysql.com/company/>
- KuchinskasSusan. *The Both-Source Way to Open Source Revenues*. March, 2004  
URL: <http://www.internetnews.com/dev-news/article.php/3327101>
- OpenOffice.org. *OpenOffice.org Homepage*. April, 2005 URL: <http://www.openoffice.org/>
- OpenOffice.org. *Quality Assurance Project homepage* April, 2005 URL: <http://qa.openoffice.org/>
- MockusAudris T.FieldingRoy HerbslebJames A *Case Study of Open Source Software Development: The Apache Server*. January, 2004  
URL: <http://www.research.avayalabs.com/user/audris/papers/apache.pdf>
- HiserSam *The Power of Mozilla Firefox and OpenOffice on Windows*. October, 2004  
URL: <http://linux.sys-con.com/read/46494.htm>
- González BarahonaJesús RoblesGregorio Seoane PascualJoaquín. *Introducción al software libre*. September, 2003 URL: [curso-sobre.berlios.de/introsobre/sobre.pdf](http://curso-sobre.berlios.de/introsobre/sobre.pdf)