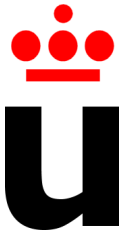

UNIVERSIDAD
REY JUAN CARLOS

Master in Free Software

Academic Year 2012/2013

Dive into openSUSE Project and it's aspects

Autor: Athanasios-Ilias Rousinopoulos

Tutor: Dr. Gregorio Robles

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

As for the community aspect I have contributed to the openSUSE Project for the last 2 years. Additionally, I have participated in conferences where I presented openSUSE to the attendee's, as an openSUSE Ambassador. In addition to tha I took part in GSOC 2012 with openSUSE Project. During this period of time I developed a plugin for openSUSE Connect Framework. As for the academic aspect I believe that the knowledge acquired along the Master in Free Software and my research experience during my work in Libresoft will help me to mine, analyze (with focus in software development teams and projects) the openSUSE Project. My experience and contribution gained from both aspects will bring to the light very interesting results and show to the public

1.2 Structure of the document

1.3 Related Technologies

The openSUSE project [1] is a worldwide effort that promotes the use of Linux everywhere. openSUSE creates one of the world's best Linux distributions, working together in an open, transparent and friendly manner as part of the worldwide Free and Open Source Software community. The project is controlled by its community and relies on the contributions of individuals, work-

[1] `http://www.opensuse.org/en/`

ing as testers, writers, translators, usability experts, artists and ambassadors or developers. The project embraces a wide variety of technology, people with different levels of expertise, speaking different languages and having different cultural backgrounds. The openSUSE Project's effort and software relies on the openSUSE Distribution[2].The openSUSE distribution is a stable, easy to use and complete multi-purpose distribution. It is aimed towards users and developers working on the desktop or server. It is great for beginners, experienced users and ultra geeks alike, in short, it is perfect for everybody! The latest release, openSUSE 12.3, features new and massively improved versions of all useful server and desktop applications. It comes with more than 1,000 open source applications. openSUSE is also the base for SUSE's award-winning SUSE Linux Enterprise [3]. Furthermore openSUSE has many subporjects and working teams as in the software development part (Education,Factory,ARM,Medical,Tumbleweed etc) and as in the other parts (Artwork,Wiki,Translation).

[2] http://en.opensuse.org/Portal:Distribution

[3] https://www.suse.com/products

Chapter 2

Objetives

Objective 1 : Mine and analyze of mailing-lists related to software development projects

Objective 2 : Mine and analyze source code repositories related to software development , by also genereting statistics in these repositories

Objective 3 : Mine and analyze the bug-tracking system of openSUSE Project (related to software and openSUSE distribution development) by also genereting statistics in the bug-tracking system

Objective 4 : Estimate the cost of analyzed source code repositories and openSUSE Projects by using the COCOMO Model [1].

Objective 5 : Measure the participation of volunteers and developers in openSUSE Project. With the term 'participation' we define :

- Commtis in source code repositories
- Number of Mails in mailing-lists
- OBS [2] commits.

Furthermore as for the participation will be followed some principals defined by [1, Chapter 7]

Objective 6 : Contribute data to FLOSSMetrics for future studies

Objective 7 : Contribute to VizGrimoire and MetricsGrimoire projects by sharing source code patches and modules developed during Msc Thesis

[1] http://en.wikipedia.org/wiki/COCOMO

[2] http://openbuildservice.org/

Chapter 3

Design

Tools and software :

- CVSAnaly [1] - Tool that extracts information out of source code repository logs and stores it into a database.
- MailingListStats [2] - Command line based tool used to analyze mboxes
- Gitstats [3] - Statistics generator for git repositories
- Bicho [4] - Command line based tool used to analyze bug tracking systems
- SLOCCOUNT [5] - Set of tools for counting physical Source Lines of Code (SLOC)
- VizGrimoireR [6] - Extract software metrics for CVSAnaly , Bicho , MailingListStats
- VizGrimoireJS [7] - Create dashboards and reports from VizGrimoireR
- Karma2python [8] - Karma Plugin (openSUSE Connect) in Python
- Python (v 2.7.3) [9]

[1] https://github.com/MetricsGrimoire/CVSAnalY
[2] https://github.com/MetricsGrimoire/MailingListStats
[3] http://gitstats.sourceforge.net/
[4] https://github.com/MetricsGrimoire/Bicho
[5] http://www.dwheeler.com/sloccount/
[6] https://github.com/VizGrimoire/VizGrimoireR
[7] https://github.com/VizGrimoire/VizGrimoireJS
[8] https://github.com/athanrous/karma2python
[9] http://python.org/ftp/python/2.7.3/Python-2.7.3.tgz

- Scipy (v 0.10.1-3.1.2) [10]
- Matplotlib [11]

3.1 Tools instalation

As we our software and design is based on the tools mentioned before at this Chapter we will see the installation instructions for each tool.The purpose of the Section **??** is to show the installation process of the tools in openSUSE 12.2. Concerning openSUSE we have to mention that not all the package dependencies are available in the default installation of the operating system.As a result the aim of this Section is to contribute all this knowledge to the openSUSE Community for furthre study and improvement.

3.1.1 CVSAnaly

CVSAnalY has the following dependencies [12]:

- RepositoryHandler (see 3.1.1)

  RepositoryHandler Installation

  - We install the following dependencies

    ```
    sudo zypper in git-svn curl
    libcurl4 python-pycurl cvs cvsps git-cvs
    ```

  - In the folder where you downloaded RepositoryHandler run the following command:

    ```
    sudo python setup.py install
    ```

- cvs (optional, for CVS support)
- subversion (optional, for SVN support)

[10] http://software.opensuse.org/package/python-scipy
[11] http://software.opensuse.org/package/python-matplotlib
[12] http://metricsgrimoire.github.io/CVSAnalY/

- git (optional, for Git support)
- Python MySQLDB (optional but recommended)
- Python SQLite (optional)

After installing all the dependencies in the folder where you downloaded CVSAnaly run the following command:

```
sudo python setup.py install
```

3.1.2 MailingListStats

- Install the requirements

  ```
  sudo zypper in python-mysql python-psycopg2
  mysql-community-server
  mysql-community-server-client
  mysql-community-server-errormessages
  ```

- Download MailingListStats:

  ```
  git clone https://github.com/MetricsGrimoire/MailingListStats.git
  ```

- In the folder where you downloaded MailingListStats run the following command:

  ```
  sudo python setup.py install
  ```

  Here you can find an example of usage of MailingListStats (you don't have to create the database before running the following command) :

  ```
  mlstats --db-user root --db-password root
  --db-name mlstats_kernel --db-admin-user root
  --db-admin-password root
  http://lists.opensuse.org/opensuse-kernel/
  ```

3.1.3 Gitstats

- We install the prerequisites for Gitstats (as we have already installed Git , and Python we need the last dependcy)

```
sudo zypper in gnuplot
```

- Then browse the

```
\item We download the latest version of Gitstats from here by cloning
\begin{lstlisting}
git clone git://github.com/hoxu/gitstats.gits
```

- You don't have to install Gitstats, just you have to launch as follows : You run:

```
$ ./gitstats /project_folder /output_folder
[the folders have to be different]
```

3.1.4 Bicho

- Download Bicho

```
git clone https://github.com/MetricsGrimoire/Bicho.git
```

- As we installed all the prerequisites for MailingListStats , now we have to install only the following packages (by typping this command in terminal) :

```
sudo zypper in python-beautifulsoup
python-feedparser python-dateutil python-storm
```

- We create a database called

```
bicho_test
```

and then we run the following command :

```
bicho --db-user-out=root
--db-password-out=root --db-database-out=bicho_test
-d 15 -b bg -u
https://bugzilla.libresoft.es/buglist.cgi?product=bicho
```

- Download the following module in order to analyze issue trackers from repositories

```
wget https://launchpad.net/lazr.restfulclient
/trunk/0.13.3/+download
/lazr.restfulclient-0.13.3.tar.gz
bicho --db-user-out=root
--db-password-out=root
--db-database-out=bicho_issue -b github
-u https://api.github.com/repos/VizGrimoire/VizGrimoireJS/issues
--backend-user=[GITHUB USER]
--backend-password=[GITHUB PASS]
```

  In openSUSE we faced the bug mentioned here [13]

3.1.5 SLOCCOUNT

- We add (via terminal) the following repository in our existing ones. It is necessary because SLOCCOUNT is provided by this repository.

```
sudo zypper ar
http://download.opensuse.org/repositories/
devel:/tools/openSUSE_12.2
```

- Then we install the packages

```
sudo zypper in sloccount sloccount-debuginfo
 sloccount-debugsource
```

3.1.6 VizGrimoireR

- Install the dependencies

```
zypper ar http://download.opensuse.org/
repositories/devel:/languages:/R:/patched/openSUSE_12.2/
zypper in R-base
```

[13] https://github.com/MetricsGrimoire/Bicho/issues/66

```
R-base-devel R-patched libmysqlclient-devel
gcc-c++
```

- Configure Mysql

```
PKG_CPPFLAGS="-I /usr/include/mysql"
PKG_LIBS="-L /var/lib/mysql -lmysqlclient"
export PKG_CPPFLAGS="-I /usr/include/mysql"
export PKG_LIBS="-L /var/lib/mysql -lmysqlclient"
```

- Install via R stat

```
[ install.packages("package_name"); ]
```

The following packages (please follow the sequence provided)

```
RMySQL rjson RColorBrewer digest abind
plyr gtable stringr lattice reshape2
scales dichromat munsell colorspace
labeling graph BiocGenerics Rgraphviz
proto MASS ggplot2 DBI rgl
getopt optparse zoo
```

- Install VizGrimoireR

```
git clone https://github.com/VizGrimoire/VizGrimoireR.git
cd VizGrimoireR/
R CMD INSTALL vizgrimoire/
```

3.1.7 VizGrimoireJS

- chmod –R a+x /srv/www/htdocs/VizGrimoireJS/
  chmod –R 777 /srv/www/htdocs/VizGrimoireJS/

3.1.8 Scipy

- In terminal we type

```
sudo zypper in python-scipy
```

3.1.9 Matplotlib

- In terminal we type

```
sudo zypper in python-matplotlib
```

3.1.10 Generate Stats

In order to generate the stats for a specified repository you have to follow the following steps :

- ```
  --db-password root --db-database cvsanaly_obs
  --no-parse --extensions=CommitsLOC,FileTypes
  /path/to/your/repository
  ```

- ```
  python unifypeople.py
  -d cvsanaly_zypper -u root -p root -i no

  python domains_analysis.py
  -d mlstats_zypp -u root -p root
  --db-hostname localhost --db-port 3306
  ```

- ```
  python domains_analysis.py -d
  cvsanaly_zypper -u root -p root
  --db-hostname localhost --db-port 3306
  ```

- ```
  python its2identities.py
  --db-database-its bicho_zypper
  --db-database-ids cvsanaly_zypper
  -u root -p root --db-hostname
  localhost --db-port 3306
  ```

- ```
  R --vanilla --args -d mlstats_zypp -u
  root -p root -i cvsanaly_zypper -r
  people,companies,repositories -s
  2007-03-20 -e 2013-05-03 -o
  /home/zoumpis/git/
  VizGrimoireR/vizGrimoireJS/data/json/
  -g months < mls-analysis.R
  ```

- ```
  R --vanilla --args -d cvsanaly_zypper
  -u root -p root -i cvsanaly_zypper
  -r people,companies,repositories
  -s 2006-10-25 -e 2013-05-19
  -o /home/zoumpis/git/VizGrimoireR/vizGrimoireJS
  /data/json/ -g months < scm-analysis.R
  ```

- ```
  R --vanilla --args -d
  bicho_zypper -u root -p root
  -i cvsanaly_zypper -r people,
  companies,repositories -s 2011-11-10
  -e 2013-05-23 -o /home/zoumpis/git
  /VizGrimoireR/vizGrimoireJS/data/json/ -g months -t
  github < its-analysis.R
  ```

In order to run the R scripts you have to make the appropiate changes according to your needs.
  ```

Chapter 4

Testing and validation

4.1 Zypper Analysis

4.1.1 Introduction

Zypper is the package manager in openSUSE and SUSE distribution. In more details and according to the wiki definition [1] is a command line :package manager, which makes use of libzypp, providing functions like repository access, dependency solving, package installation,etc.Zypper can also handle repository extensions like patches, patterns and products.

4.1.2 Repository Analysis

In order to analyze the Zypper repository we have to define some metrics so to have heterogeneous data and better quality of analysis. We have to point that depending of the kind of analysis you wish to do you have to define the corresponding metrics. ViZGrimoire toolset offers, for instance, analysis specialised in company's contribution on a repository but in our case we focus on the community part. To that end the metrics we define are the following ones:

- Commits per repository
- Authors per repository
- Files

[1] `http://en.opensuse.org/Portal:Zypper`

- Branches
- Lines Added
- Lines removed

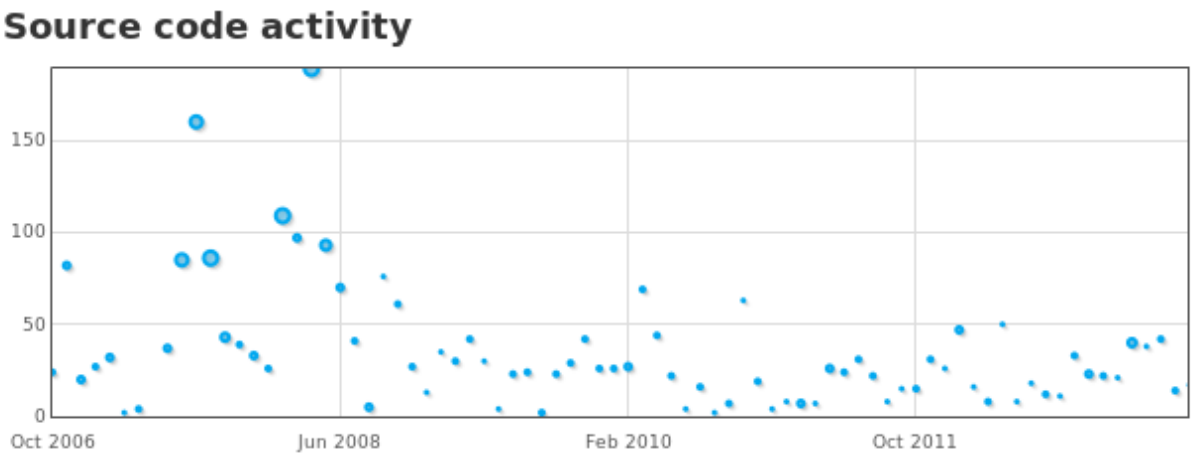

Figure 4.1:

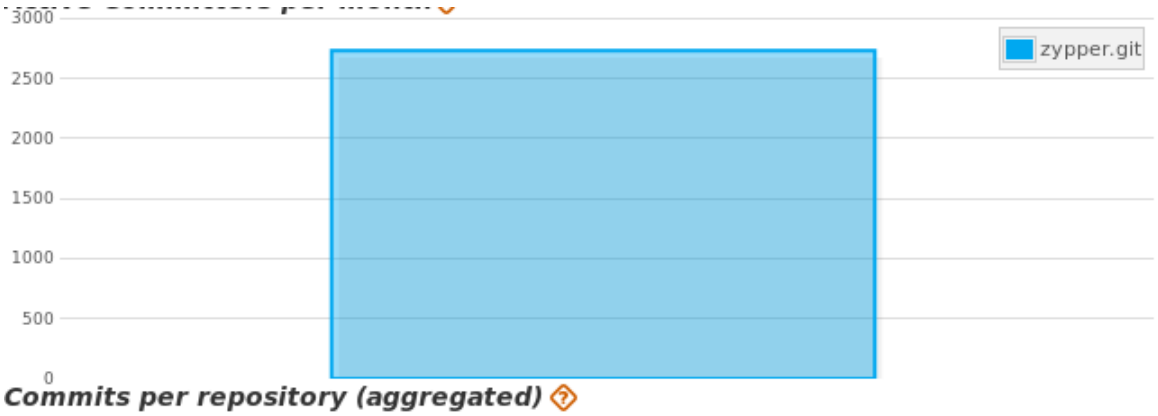

Figure 4.2:

Taking about the commits per repository we have to count in two plots. These plots are 4.1 and 4.2. In 4.1 we see a "Bubble" plot representing the evolution of the commits in Zypper repository during the time. Furthermore we can clearly see that the study period is from

October 2006 until now [in fact is from 25-06-2006 to 12-05-2013] so it is a period of time that overcomes a 6 years period. By focusing more on the plot we can see that the period from October 2006 to January 2008 is the period of time with maximum number of commits. After January 2008 we see that the maximum number of number of commits does not overcomes the 100 and specially the last period [October 2011 - now] this amount is being reduced 50 % . In order to analyze the evolution of the commits we have to included in our analysis 3 more parametres mentioned before, otherwise we will only be able to make an analysis related to the number of the commits . These parametres are *Files, Lines Added, Lines removed*. By including these 3 parameteres in our analysis we can see the impact of the commits in the source code repository. At this point there is a lot of study that could be done, and somehow questions like *How the commits affect in the evolution of the source code repository* or *Concerning the lines removed and added which is the relation with the evolution of the number of files*. At that point, focusing on the analysis, we see that the maximum number of files concerning the plot 4.4 is commited in January 2008 exactly the same month where the period with the maximum number of commits ends. In order to verify a proportional relationship between the number of commits with the 3 parameters mentioned before we have to examine the evolution [during the same period of time] for these metrics. As for the "Lines Added" we can clearly see in the Figure 4.4 that the maximum number of lines added takes places the same month (January 2008) as previously and the same period of time we see in the Figure 4.4 a high amount of lines removed close to January 2008 but the maximum value is being commited almost 2 years later (February 2010). During the same month we can see in the Figure 4.1 that the amount of the commits maintains a stable curvature which means that the developers removed a very high amount of lines with a small amount of commits. Comparing that fact with the relation between the number of commits at the first period and the lines added at the same period we can obviously see that developers and contributors added a high amount of lines by commiting very often and as a result a proportional relationship between commits from developers and lines added by the developers for this period of time. Although this relationship sounds an ideal one we cannot confirm it after the first period of the study.

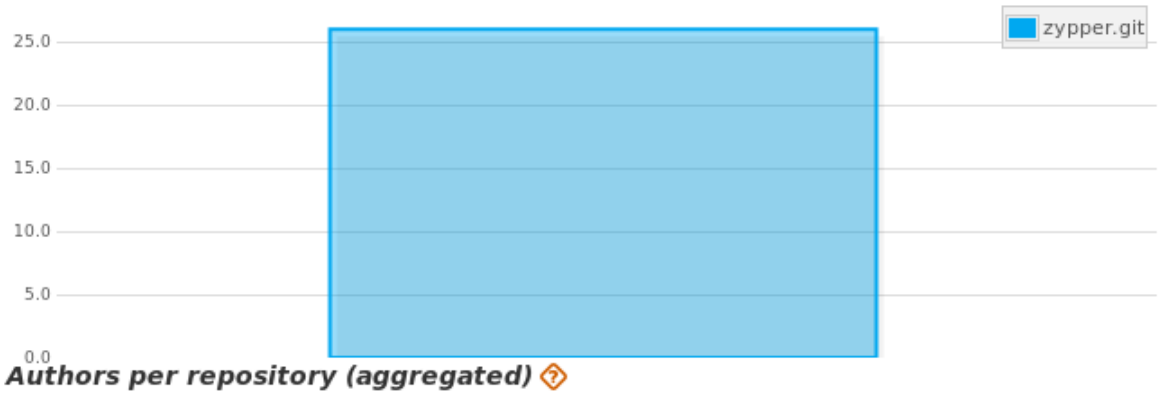

Figure 4.3:

Having a look at the Figure 4.5 we can see that the plots representing the evolution of Commiters and Authors is exactly the same and as a result we have the same amount of Commiters and Authors at Zypper repository. That means that people who are contributing to the repository are the authors of the files as well. Having this fact in our mind we can examine which is the relationship between the evolution [or number] of Commitss in the repository and the Commiters. In the same Figure we see that the curvature of Commits and of Commiters are very similar. For instance we see that the higher amount of commits takes place in January 2008 and the same happens with the number of Commiters [or author in our case]. We can explain this phenomeno with many ways, but under simple conditions we see high activity in the repository during this period of time. Furthermore concerning the number of "Lines Added" (see 4.4) for this period of time we can declare that the more the commiters are the more commits we have and the more lines added we have in our repository. until this part of our analysis we didn't include the number of branches as a factor of our study. Having a look at the Figure 4.5 we can see that at January 2008 our repository has 4 branches and we have the same number of our branches in our repository nearby Octobrer 2011. At October 2011 things are changing a lot. We see that the we have only 3 Commiters (maximum 4) from October 2011 until now and the number of Commits doesn't overcome the 50. So in terms of Commits, Commiters and Author we could say there is a stable curvature.

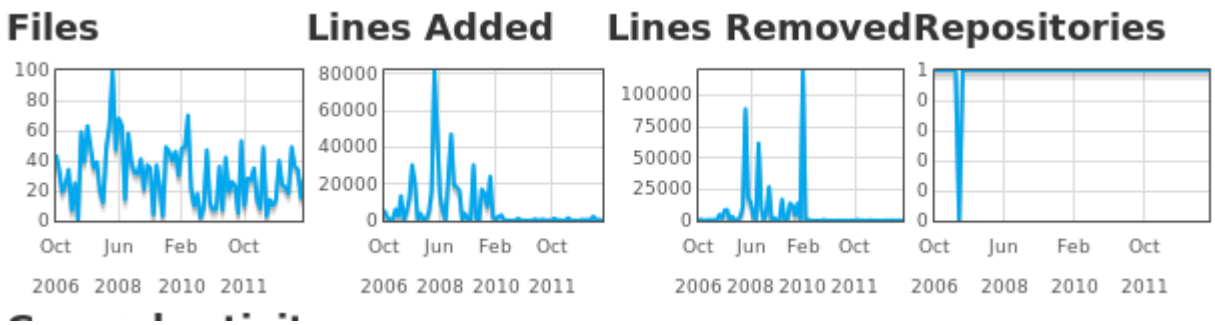

Figure 4.4:

By having a look at the Figure 4.4 we can confirm as well a stable curvature as there is a very low activity in both cases. The facts mentioned before have a special impact on the number of Branches in our repository. After October 2011 the number of Branches is increasing by obtaining the maximum value as well (7 branches). By focusing a bit more on the plots close after October 2011 we had only 3 developers who were mainting 7 branches. Seems that this ratio between commiters and branches hasn't maintained for large period of time and as consquense currently we can talk about 2 commiter who maintain 3 branches in the Zypper repository. At this point many questions and research challenges can come out. Although during the last period the number of Branches obtains the maximum value, the Lines added and Lines Removed do have a low value and the number of Commits as well. Under these consumption we can see that high activity in Commits means high number of commiters in the source code repository, but that's not an absolute deduction as we have to exam the activity related with bug-fixing on the Zypper repository. For sure more research can be done, as part of the further work on how the activity is splited between the branches the developers and how the commits are being spread between the developers. Answering this questions we will be able to define who is the core developer in this repository during a concrete period of study. Currently in terms of anonymity we cannot publish developers names and the information that we have (offered by the tools) provide us information about the Top Authors (the last year) and the Top Authors during our period of study. So in order to answer to these questions this kind of metrics (e.g Top Author per Period) have to be defined in ViZGrimoireR and ViZGrimoireJS. For the time being and according to Figures 4.3 and 4.2 we have 26 Authors during the whole period

of study and 2734 commits done. Although is not part of our metrics defined at the beginning we just define (for general interest) the Mean Commits per Author are 105. In the next case study in Chapter 4.2 we will see which is the Mean Commits per Author for the Open Build Service source code repository. Finally although we know the lines changed (removed, added) currently we don't have any information about the type of code changes that developers made in Zypper repository. As type of changes we can define any kind of changes in source code (functions,class,if statements etc) but also in which programming language these changes have been done.

Figure 4.5:

4.1.3 Mailing list Analysis

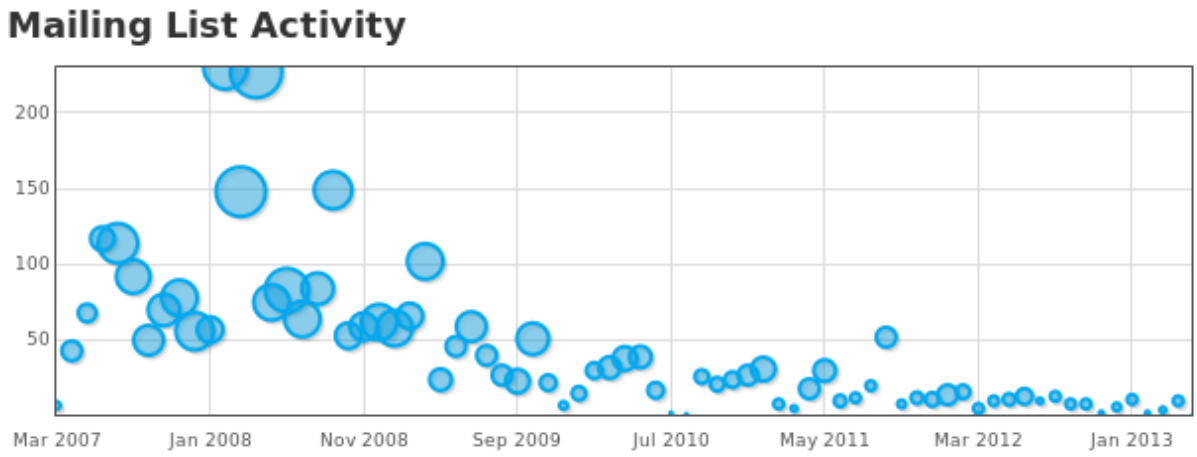

Figure 4.6:

In the Chapter 4.1.2 we saw the analysis of the Zypper source code repository, until now we know what is going on with the evolution of the commits, the lines added and removed on the repository but we don't have any piece of information about the mailing list and how developers (contributors and users) are interacting with. As we did with the source code repository here we define a set of metrics so as to analyse the mailing list.

- Messages sent per repository

- Senders (people) per repository

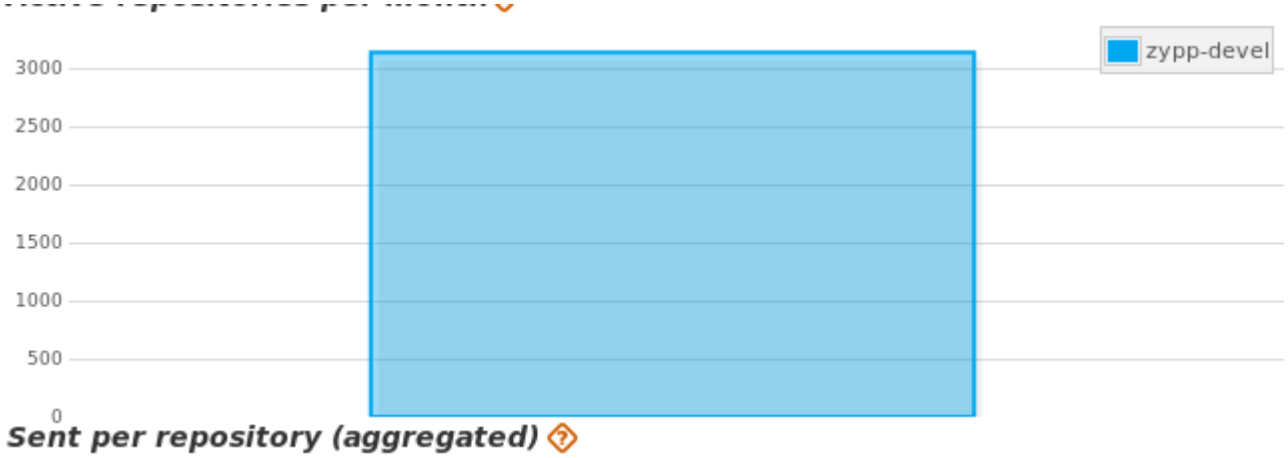

Figure 4.7:

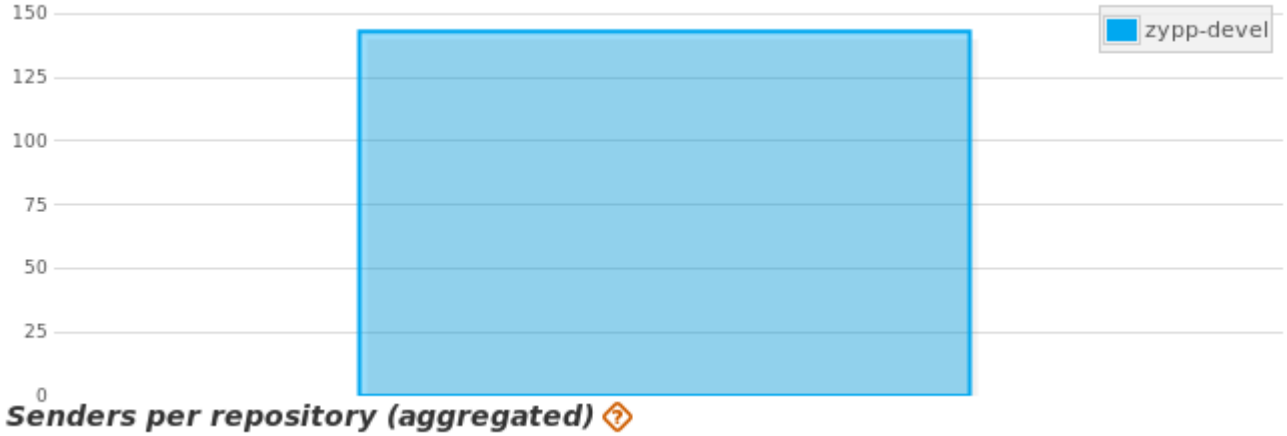

Figure 4.8:

Our period of study, regarding the mailing list, is 9 months shorter in comparison with the source code repository's period of study. To be more concrete in the mailing list we study the activity from 20-03-2007 to 02-05-2013. As our period of study is a bit shorter let's divide it in 3 sub-periods. These sub-periods are :

- March 2007 - November 2008

- November 2008 - May 2011

- May 2011 - May 2013

The division of the periods has been done according to the activity shown in the Figure 4.6. According to Figures 4.7 and 4.8 we have 3144 messages sent in the mailing list by 143 Authors (according to ViZGrimoire terms). Concerning this data we could claim that each Author send approximately 22 messages during 2235 days, but this way is not the appropiate one as we don't know if during the whole period each Author send at least one message. In other words, as we know that our period of study is 73 Months we only have information about total amount of Authors and Messages sent, but we don't have any information about the evolution of authors and the messages sent. In the Figure 4.9 we see the evolution of the senders and the messages sent. Concerning the sub-periods defined before we analyze the evolution of the *Senders* and *Sent*. As for the first period we see that the maximum value of messages sent is being obtatined during this period. In more details during January 2008 and November 2008, for 2 times we see that the messages sent overcome the amount of 200, until January 2008 we see that the maximum amount of messages sent is no bigger than the half of the maximum amount obtained after January 2008. In other words we could claim that the amount increased almost 100 %. At this point as we know what is going on with the evolution of the sent messages we have to see if the senders the messages increased 100 %, so to define a proportional relationship between the messages *Sent* and the *Senders*. As for the *Senders* at the first period we see that at the beggining the senders were 5 and nearby after January 2008 they obtain the maximum value (25) but after that at the end of the sub-period the total senders are 15. Concerning this piece of data we can claim that the number of sender increased approximately 300 % . Furthermore by November 2008 the messages sent were only 50 whereas the amount of messages during January 2008 was 200, in other words the amount of messages decreased 400 % and the amount of senters increased only 20 % between January 2008 and November 2008.

Figure 4.9:

Concerning this piece of information a good approach to evaluate the numbers given is to define a percentage of messages sent per sender during these two months (January 2008 and November 2008). As for January 2008 approximately this percentage is 12/50 = 0.24 or 24 % and in November 2008 is 15/55 = 0.27 or 27 % . Concerning these percentages we could state that the activity in the mailing list remained stable at this period. By having a better look at the Figure 4.9 in the *Sent* plot the curvature is being carectised by many ups and downs whereas the *Senders* plot's curvature is more stable and with less ups and downs we could claim that the participants that mailed during this period of time and then stoped mailing, send a little amount of messages at a short period of time (1 or 2 months maximum). We could explain the motives and the reasons of this fact but we are not able to claim that people who posted in the mailing list are the same people who commited in the source code repository. In other words the data are not homogeneous one since we are talking about different authors and commiters. By mining the data from our database we produced the following tables [4.1.3 and 4.2.2], refering to Top Authors and Top Senders in the source code repository and in the mailing list. By having a better look at the tables we can see that only 50 % of the Top Authors and the Top Senders are the same [M.A ,J.K ,D.M ,J.R , S.K]. We cannot define a state for the activity in the source code repository that could be applied to the mailing list as well. The reasons of this stack are :

- The two periods of study are not the same.
- High and low activity in both cases doen't always happen at the same period of time.
- People high-rated in the Top Author list are not high rated in Top Senders list. This also happens vice versa.

Concering these facts we cannot claim, for instance, that the people who sent a high number of mails the same ones commited at the same way during this period of time.

Top Author	Num.of Commits
J.K	1627
M.A	620
M.V	113
D.M	99
J.R	87
D.H	59
T.G	43
K.K	24
G.M	23
S.K	7

Top Sender	Messages Sent
M.A	474
D.M	376
J.K	306
K.K	277
M.S	168
M.M	164
S.V	98
S.K	84
S.S	73
J.R	72

Although we can claim a general fact for source code repository and the mailing list we have to examine the activity of people who are both Top Authors and Top Senders the same period of time (in our case will be January 2008 to November 2008)

Name	Commits	Messages
J.K	661	137
M.A	9	146
D.M	33	125
J.R	76	62
S.K	7	56

According to Table 4.1.3 we see that people like J.K commited much more than posted in the mailing list and people like M.A posted more in the mailing list than commited in the repository. This Table prove us that for this period of time we cannot define a general rule for the activity. In other words we cannot say for all the developers that commited more than posted and vice versa. Furthermore due bug in ViZGrimoire dashboard we are not able to see who are the Top Closers and Top Openers. To that end we cannot see the activity of the *common people* in mailing list,source code repository and issue tracker and then extract a general fact. By associating the activity in mailing lists source code repositories and issue trackers we are able to see if and in what way developers interact with these kind of tools. Furthermore as the social coding is the core of the evolution of each Open Source project studying the activity in an Open Source project has to include an analysis of the source code repository.

4.1.4 Issue tracker analysis

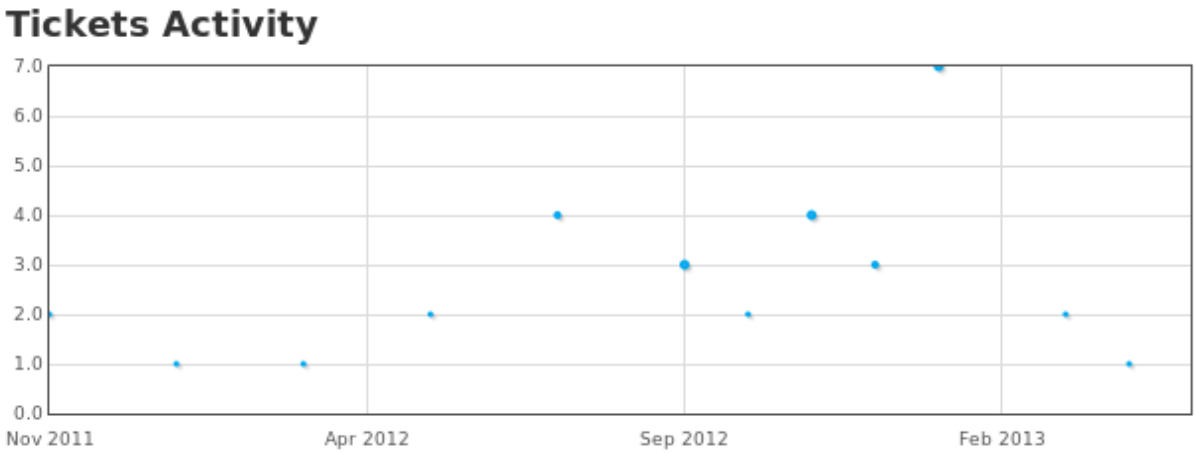

Figure 4.10:

On the one hand mailing lists are the traditional method and way to establish communication between developers and users and on the other hand the issue trackers offer a *communication channel* between contributors and developers by providing comments and commits. So part of our analysis is the activity in the zypper issue tracker. Having a look at the Figure 4.10 we see that the issue tracker period of study is even more shorter than the mailing list's period of study. So by making the assumption that the names of the Top Closers and Top Openers are available we couldn't analyze properly the activity in the issue tracker in comparison with the mailing list and the source code repository. To be more concrete let's see the activity in the Figures 4.1 , 4.6 and 4.10 from August 2012 to May 2013. In the first 2 plots (as we examined before) the activity is lower in comparison with the rest of the period of study. As a result now we examine the same period for the issue tracker. By having a look at the plot 4.11 we see that the highest value of closed tickets is being obtained during this period of study [August 2012 - May 2013]. According to the plot we can clearly see that in August 2012 the number of tickets closed and the number of closers is zero. So during this month ther is no activity in the issue tracker. After August 2012 we see the activity increasing rapidly in the closers part but also in the tickets closed. Although the activity is being increased in both parts we cannot define a proportional relationship between the Closers and the t*ickets closed*. In more details between September

2012 the tickets closed have a different curvature in comparison with the *closers* line. The *closers* line increase more sharply and remains more stable (in the maximum) in comparison with the *closed* line. That means that for the period of time which the *closers* remained stable the same people closed less issues in comparison with the rest of the period of study. Maybe this "frozen" activity is a result of many reasons but as we don't have any information about the closers name and data we cannot claim or generate a fact for this short period of time.

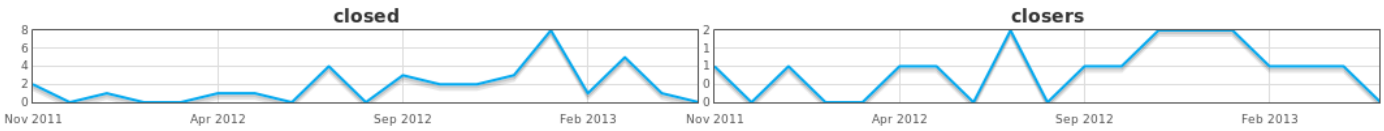

Figure 4.11:

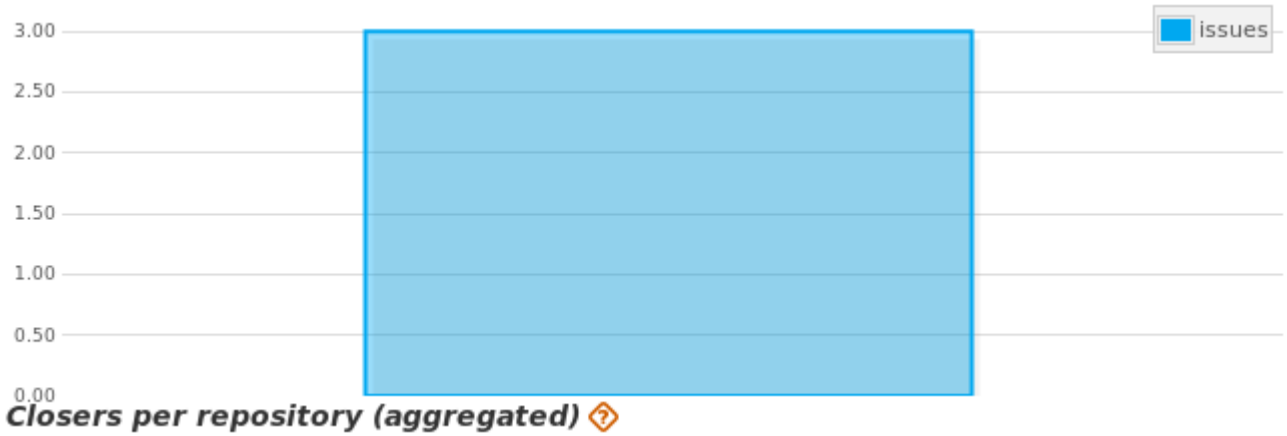

Figure 4.12:

Figure 4.13:

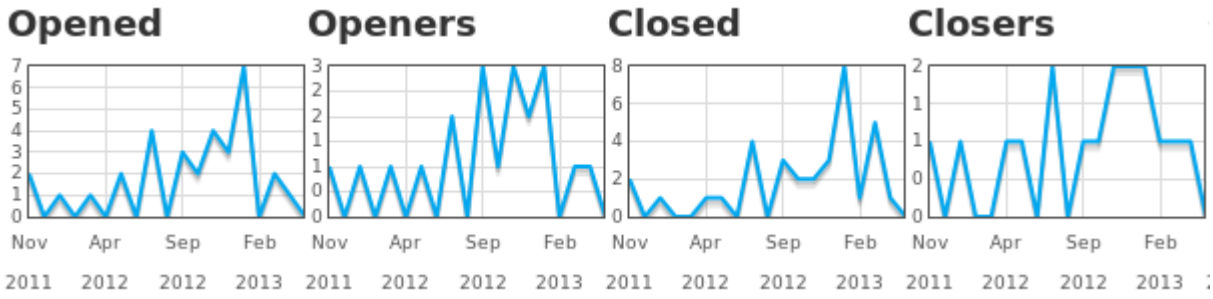

Figure 4.14:

Concerning the plot 4.14 we can see two new metrics related to issue tracker. These metrics are :

- Tickets opened
- Tickets openers

As previously we analyzed the *closers* and the *closed* lines. The upcoming question is what is going with the evolution of the opened tickets and the persons who opened tickets from November 2011 to May 2013. As the iteraction with an issue tracker includes the state of opening and closing of tickets it is impossible to exclude the study of evolution of the *open*

and *openers* . As we did in the analysis of the mailing list and the analysis of the source code repository we will focus on the period from August 2012 to May 2013. Concerning the plot 4.14 for the tickets opened we see that from August 2012 the number of opened tickets is increasing sharply when close to February 2013 obtains the maximum value (7). From the *Openers* part we see a slightly different evolution. In more details between August 2012 and February 2013 the Openers obtain the maximum value (3) three times when all this period of time the minimum amount of people who opened a ticket is 1 and the maximum amount of people is 3. By having a look at the *opened* we see that the minimum amount of tickets opened is 2 and the maximum 7. When for the first time the openers gain the maximum value (September 2012) the number of opened tickets is only 3. Furthermore the second time that the openers obtain the maximum value the tickets opened are only 4. To that end the last time when the openers obtain the maximum value the open 7 tickets, which can be concerned as the highest activity during all the period of study. We could claim that the team of people of openers does not remain stable and when a new member involves in the team is not productive at once. Unfortunately as ViZGrimoire dashborad does not provide any information about the data of closers and openers we are not able to approve or decline this kind of theory. As a result our analysis for Zypper Issue tracker stops here.

4.2 OBS Analysis

4.2.1 Introduction

Before diving into the analysis of the OBS repository, we have to have a look at the definition of OBS. According to its definition *The Open Build Service (OBS) is a generic system to build and distribute binary packages from sources in an automatic, consistent and reproducible way. You can release packages as well as updates, add-ons, appliances and entire distributions for a wide range of operating systems and hardware architectures* [2]. In addition to that definition the OBS offers to users of any distribution the possibility to search for built packages for their distribution. Furthermore as for the developers it is an efficient place to build up groups and work together through its project model [3]. As a consequence people from other Free/Open

[2] `http://openbuildservice.org/about/`

[3] `https://build.opensuse.org/`

Source projects contribute in the development of the Open Build Service, communicate via the mailing list or submit issues in the issue tracker. So our study is more open in comparison with the Zypper case.

4.2.2 Repository Analysis

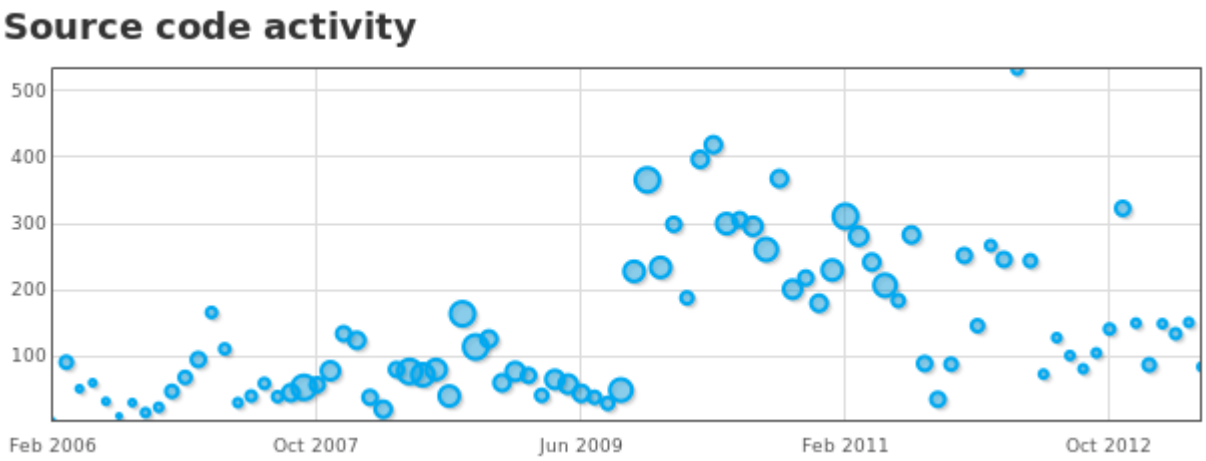

Figure 4.15:

As we did with Zypper repository in the Section 4.1, in this section we will analyze the Open Build Service source code repository. According to 4.15 we see that the activity in the Open Build Service repository begins at February 2006 until May 2013. We divide our period of study into 3 sub-periods. These sub-periods are :

- February 2006 - January 2009
- January 2009 - October 2012
- October 2012 - May 2013

Apart from the division of the our period of study, in order to analyse the activity in the OBS source code repository we define the following metrics :

- Commits per repository

- Commiters per repository
- Authors per repository
- Files
- Branches
- Lines Added
- Lines Removed

Figure 4.16:

Considering the analysis of commits we have to point out that the amount of commits in OBS repository is even 6 times higher rather than Zypper repository (12411 in OBS and 2735 in Zypper).As our analysis cannot stand without focusing in the metrics defined before, in the Figure 4.16 we can see the evolution of commits by the passing of the time. For the first sub-period of study [February 2006 - January 2009] we see that the maximum value of commits that is being obtained, doesn't overcome the 200. Apart from that the *Commits* line maintain its curvature stable with normal ups and downs. In general in the majority of the months during this subperiod the number of commits is less or equal to 100. To be more concrete only 4 times the *Commits* overcomes the limit of 100 commits by proving the stable curvature of the line. The upcoming question related to this stability is the interaction of the developers and contributors during the first sub-period. As we did in Zypper case in order to have a more concrete view about the developers and contributors activity we have to see the evolution of the metrics *Lines*

Added , *Lines Removed* and *Files*. The rest of the metrics defined before (*Authors per repository* , *Branches*) are also part of our analysis.

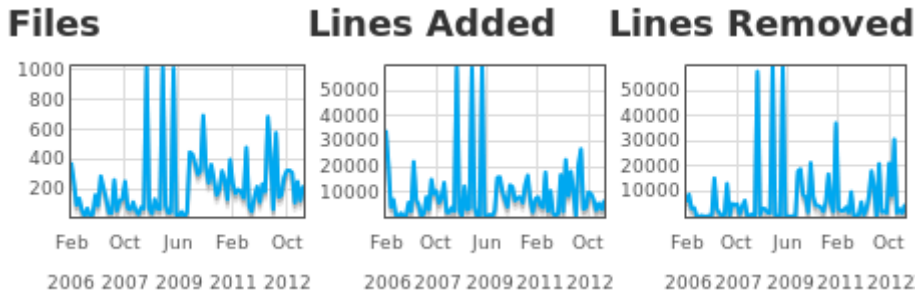

Figure 4.17:

In the Figure 4.17 we have a clear image of the number of *Files* , *Lines Added* and *Lines Removed*. The aim of this study is to associate, where possible, the activity being done by *Commiters* and *Authors* during the passing of the months by measuring this activity in *Commits*, *Files* , *Lines Added* and *Lines Removed* and number of *Branches*. Coming back to the first sub-period and especially in the evolution of amount of *Commiters* and of *Authors* we see that both *Commiters* and *Authors* obtain the same maximum value (12) and the curvature of the two lines is very similar. Under this consumption there are no differences between the evolution of the *Commiters* and *Authors* during the first sub-period. Having this fact in our mind it is very interesting to examine the evolution of *Lines Added* and *Lines Removed*. From the first view these three lines look the same. By focusing the amount obtained by each line is different. First of all for the first sub-period the *Files* obtains only once the maximum value (1000). For the rest of the sub-period the number of *Files* doesn't overcome the amount of 400 and only 3 time overcomes the limit of 200. The rest of the months the amount of files is less or equal to 200. As for the lines added we see by June 2008 and December 2008 obtains the maximum value. The rest of the period (except Janary 2007) the *Lines Added* line doesnot overcomes the limit of 20.000. By having a look at the *Lines Removed* we see that only 2 times overcome the limit of 10.000 in June 2008 and December 2008, which means the same as with *Lines Added* line. So for the first sub-period these two lines have similar curvate. Concerning the evolution of the *Commiters* and *Authors* we have 8 Commiters at January 2008 with 59625 *Lines Added* and 57846 *Lines Removed*. Furthermore considering the low amount of Commiters (or Authors)

during this sub-period and the high amount of *Lines Added* and *Lines Removed* we can clearly define a high activity per developer, as we have few developers and thousands of Lines. Apart from that we have to point out that all these changes in Lines are being done in less than 2 Branches so the evolution is very centralized and each commiter is being characterized by high activity and contribution. By focusing in the second sub-period [January 2009 - October 2012] we see that according to 4.15 we see that the commit activity changes a lot. To be more concrete we see that the lower limit of commits is 200, when in the previous sub-period was the upper-limit. Approximately we can see that the commit activity is being done in the are of 200 to 400 commits. By focusing more on the plot we will see that only at one case in March 2012 we have 533 commits. Furthermore there are cases between February 2011 and October 2012 where the number is lower than 200. Although this fact we could characterize the activity for this sub-period higher than the previous sub-period. Currently our piece of information is only related to the commits activity. As we did in the previous sub-period we analyze this kind of activity with the evolution of amount of *Commiters*, *Authors Files Added*, *Files Removed* and *Branches*. As for the *Commiters* and according to Figure 4.16 we see that the line maintains its curvature between June 2009 and February 2011 like between October 2007 and June 2009. In more details the minimum value of *Commiters* is 6 [June 2009] and the maximum one is 12 [February 2011]. We spot changes in the activity after February 2011 and before October 2012. During this period we see that the *Commiters* by February 2011 are 12 the *Authors* are 14 and by the passing of the months we see that the amount of *Authors* and *Commiters* is being decreased. To be more concrete the lowest amount of *Commiters* is 4 and *Authors* is 4 as well [June 2012]. Concering the lines of *Commits*, *Branches* and the Figure 4.17 by June 2012 we have 4 Commiters and 4 Authors who made 128 commits added 20765 lines removed 21062 lines by interacting with 2 Branches and 579 Files. Furthermore by examining the same metrics when *Commiters* and *Authors* obtain the maximum value [February 2011] we see that 12 *Commiters* and 14 *Authors* made 311 Commits, interacted with 237 Files added 8057 Lines and removed 3288 Lines while interacting with 3 Branches of source code. According to that data we see that in June 2012, although the amount of commiters is 70 % less, the activity is even more higher. In addition to that we clearly see that Authors and Commiters added almost 3 times more lines than in February 2011, removed 7 times more lines. Although in total they commited 2 times less [128 commits in one case and 311 in other] propotional the activity per

Commiter/Author is higher. So we see that we cannot claim a propotional logic between the amount of commiters and the amount of commits or between the amount of lines added and commiters and the lines removed for instance. As we refer to Authors and Commiters it is very usefull to see the activity of *Authors* and *Commiters* in February 2011 and October 2012 with more details and then define the core development team. According to the following table the core development team is defined in the Table 4.2.2 [the names in this table differ from the names used in Table 4.1.3]

Name	Total Commits
A.S	185
S.P	135
S.K	59
M.S	38
T.S	9

According to Table 4.2.2 we see that from 17 *Commiters* and 22 *Authors* during the months February 2011 and October 2012 only 5 people form the core development team. In other words that means there are developers who committed only in February 2011 or only in October 2012 but with a very low amount of commits. Apart from that sub-period, we have to find who are the Top Authors table during all the period of study [February 2006 - May 2013].

Name	Total Commits
A.S	3707
S.K	1919
S.P	1715
M.S	1506
A.B	658
T.S	627
M.M	328
D.M	227
T.Sr	214
T.So	182

By comparing the Table 4.2.2 and the Table 4.2.2 we see that 4 of the members of the core team are listed in the first 4 positions as Top Authors during all the period of study. That means that these 4 people had a continiusly high contribution and activity in the Open Build Service source code repository. The last member of the team [T.S] is also listed in the list of Top Authors but comes 6th in the ranking. Concerning the rest of the people listed in Top Authors table we can claim that also T.S has a continiously high activity during our period of study.

4.2.3 Mailing list Analysis

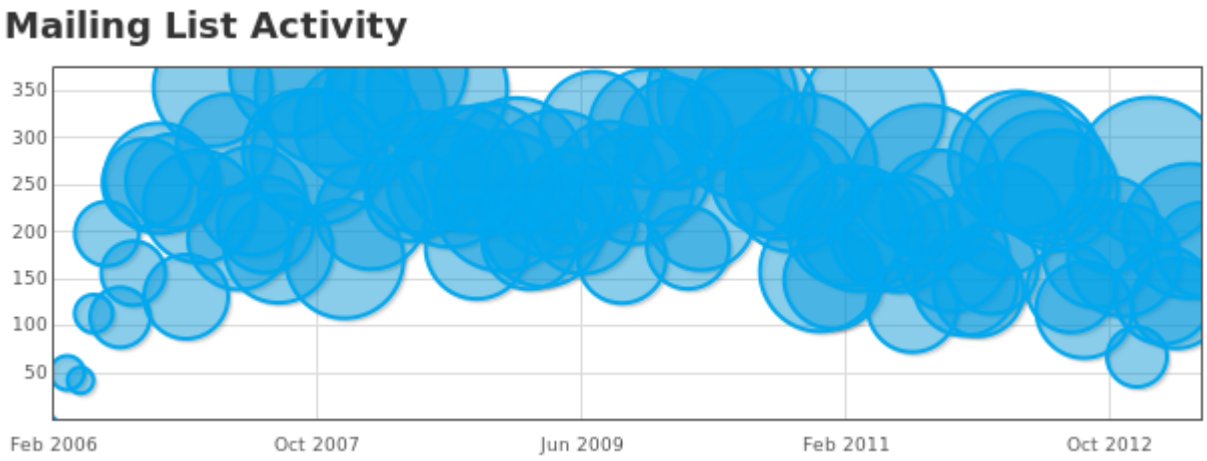

Figure 4.18:

Apart from the analysis of the source code repositority activity, part of our analysis for the Open Build Service project is the analysis of the mailing list called *opensuse-buildservice* [4], which developers, contributors and users are using in order to communicate. By having a first look at the Figure 4.18 we see that the activity is even more higher in comparison with the Figure 4.6 as wee see more people interacting with the mailing list for each month. At this point we have to mention that the radius of each buble symbolizes the people who are senting e-mails in the list, so the biggest the radius is the more people they interact with the mailing list. Furthermore we have to point out that the period of study in the MailingList and the period of study in the source code repository is almost the same [in case of the repository is 21-02-2006 to 16-05-2013 and

[4] `http://lists.opensuse.org/opensuse-buildservice/`

in mailing list is 22-02-2006 to 22-05-2013]. In order our analysis to be more concrete and specific we define the following metrics :

- Messages sent per repository
- Senders (people) per repository

By defining these metrics we are able to see if people who sented messages are also interacted with the source code repository at the same time. By providing a simple plot with a piece of data does not assure a concrete analysis and furthermore it is impossible to associate data from the mailing list with the source code repository.

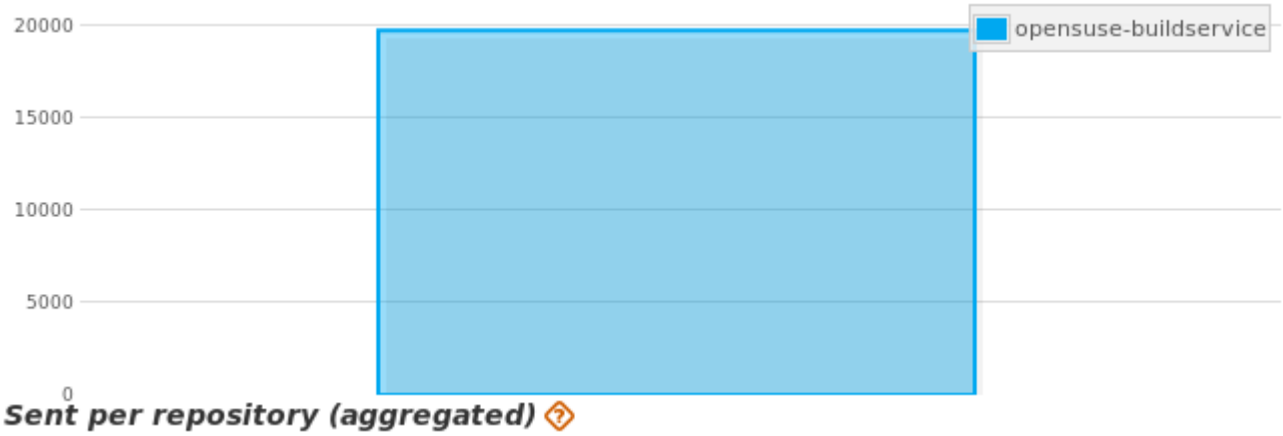

Figure 4.19:

In the Figure 4.19 we see that the total amount of sent messages is almost 20.000 [19.718 messages] and by comparing this figure with Figure 4.7 we can clearly define that the activity in opensuse-buildservice repository is higher. Apart from the messages sent in the repository we have to see how many people sent all this amount of messages.

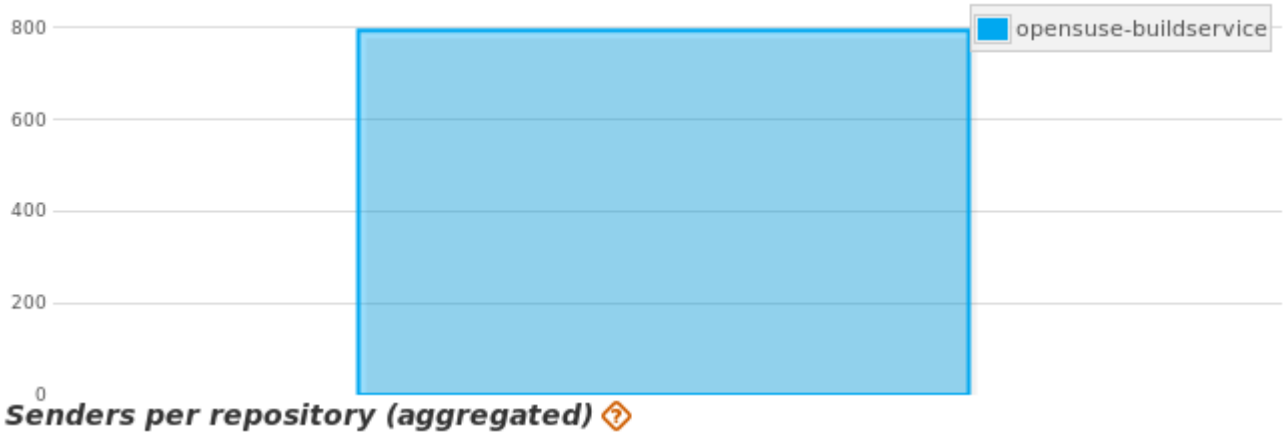

Figure 4.20:

So in the Figure 4.20 we see that the people who sent messages are almost 800 [794] which means 6 times more than the amount of people who sent messages in the Zypper mailing list. Apart from the amount of *Senders* and *Messages Sent* it is very interesting to analyze the evolution of these metrics.

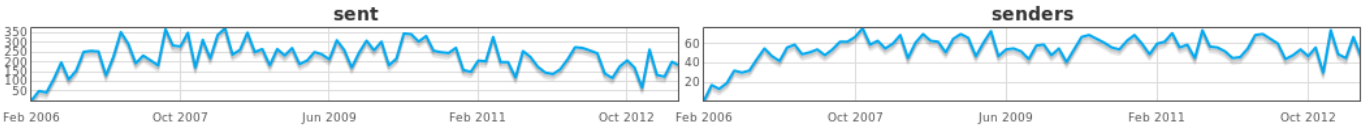

Figure 4.21:

In the Figure 4.21 we see the evolution of messages sent and senders. By having a look at the curvature of both lines we see that the *Sent* line has more ups and downs in comparison with the *Senders* line. As as result we will divide our period of study into two sub-periods. These sub-periods are:

- February 2006 - June 2009
- June 2009 - May 2013

We divide into these two sub-periods in order to split into two approximately equal pieces the whole period of our study. As for the first period of study we that in most of the cases the messages sent have a lower limit of 150 and an upper one 350. In only 3 cases [March 2006, April 2006,July 2006] the amount of messages is less than 150. That means that from the beggining of the study period the amount messages sent is high, and even more higher in comparison with the Figure 4.9 where the high activity is being maintained until the beggining of the year 2009. As we have high activity during the first sub-period we have to examine the amount of people who posted in the mailing list during the same sub-period. It is more important to see how the people involved into the mailing list, interact with it. By interaction we mean messages sent, but also if somebody left the mailing list or have not posted for a period of time. In contrast with the *Sent* line the *Senders* line increases a bit more slowly and with less ups and downs. By having a better look at the *Senders* line we see that from May 2006 until October 2007 the amount of *Senders* has an upper limit of 67 people and a lower limit of 19 people. In other words we see that 48 people interacted with the mailing list during a 16 months period of time. By running the following sql query in our database :

```
select count(*), first_date
from messages where first_date >
'2006-05-01' and first_date < '2007-11-01';
```

The result is 4095 messages sent from 48 people. Approximately we could say that each of them sent 85 messages during a 16 months period or in other words 5 messages per month. Although this kind of mean calculation seems the ideal one we cannot accept it as a clear proof as in order to extract a conclusion like that we have to examine the following parameters:

- Who are these 48 people and if are they different
- All these people posted at least one message per month during the 16 month period, or posted a high quantity of messages for some months and for the rest of them did not post

Until now we focused on the first part of the sub-period by analysing the evolution of messages *Sent* and *senders*. As for the second part we see that the messages sent maintain the upper and lower limit that we defined before [150 to 350] without any exception of less or more messages sent. So the curvature of the *Sent* line is more stable in comparison with the first part of

the sub-period by maintaing although many ups and downs. It is also interesting to see what is going on with the *Senders* line for the same period of time. As for the *Senders* line the lower limit is 45 and the upper is 76 people. Apart from the limits we see that the curvature of the *Senders* line is more stable and with less ups and down in comparison with the *Sent* line. To be more concrete in October 2007 the people who sent messages were 67 and in June 2009 54. So by calculating the difference are only 13 people less whereas in the first part of the sub-period are 48 people. That may happens because the first part of the sub-period is the beggining of the activity so more people use to be subscribed in a mailing list and at least post a minimum number of messages. By the evolution of time this amount of people becomes more stable and somehow we can define the core team [as we did in the Subsection 4.2.2]. Before defining the core team we have to examine the second sub-period as for the *Sent* and the *Senders* prospective. As for the messages sent we see that the curvature of the lines differs in comparison with the first sub-period. Especially we see more ups and downs during the whole period but also the lower and the upper limit changes. As for the limits the upper one is 347 messages sent, where in the first period is 352 and as for the lower limit is 67 messages sent. The difference between the two sub-periods is that in the second sub-period the line instead of rising up is slowing down and with many ups and downs. As for the senders line , things are changing. The upper limit in this case is 74 people and the lower limit is 30 people. In comparison with the first sub-period [67 people-upper limit,19 people-lower limit] we see that the lower limit increased 36 % which means at least more people posted in the mailing list during the second sub-period. As for the upper limit increased only 7 %. As now for the month with the maximum amount of messages Sent [April 2010] we see that is 347 messages sent by 67 people. Furthermore having a look at the month with the less amount of messages sent [December 2012] is 67 messsages sent by 30 people. By comparing these data we see an even more higher activity in the first case and more centralized activity in the second one. So we know that the double amount of people posted 5 times more messages in April 2010 where in December 2012 approximately each contributor sent 2 messages. In order to see the centrelization of the messages we have to examine many factors. This kind of analysis can extract a lot of conclusions and facts but is also part of future work.

As we studied the evolution of messages *Send* , *Senders* now we have to associate the activity in the mailing list with the activity in the source code repository. As for the activity in

the source code repository we list here the Top Authors during the whole period of study. The table is equal to the Table 4.2.2 we listed before

Name	Total Commits
A.S	3707
S.K	1919
S.P	1715
M.S	1506
A.B	658
T.S	627
M.M	328
D.M	227
T.Sr	214
T.So	182

Name	Total Messages
A.S	2832
M.L.s	792
P.P	510
M.Hu	412
S.K	348
D.S	324
M.Mk	320
D.L	293

By having a look at the two tables we see that from the Top Authors only two people posted also in the mailing list [A.S,S.K]. For these two people we have to point out that they are part of the core development team (according to Table 4.2.2) so it is very interesting to see their interaction in both sides. As for A.S we that the amount of commits is higher than the messages sent [3707 commits and messages 2832] and as for S.K we see that has also higher amount of commits [1919 commits and messages 348]. So for these period we see that the members of the core development team but also listed in Top Senders they commit more than they message. By running the same analysis in the Zypper case and according to the Table 4.2.2 and Table

4.1.3 we see that the two people who are part of the core development team committed more than posting messages in the Zypper period of study. To be more concrete [J.K] made 1627 *Commits* and send 306 *Messages* and [M.A] made 620 *Commits* and send 474 *Messages*. To that end we see that the core members of the development team they use more the social coding for communicating rather than the traditional way which is the mailing list.

4.2.4 Issue tracker analysis

As we did with Zypper we analyze the activity in the issue tracker of Open Build Service.

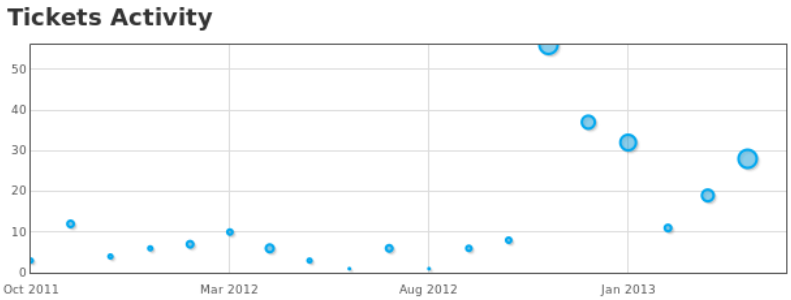

Figure 4.22:

As we see in Figure 4.22 the period of study for the issue tracker is slightly different from the source code repository and mailing list's one. To be more concrete the period of study is from October 2011 to May 2013. We divide the period of study into two sub-periods. These periods are :

- October 2011 - April 2012
- April 2012 - May 2013

Before focusing with the analysis we have to define the following metrics :

- Tickets opened
- Tickets openers

Figure 4.23:

According to the Figure 4.23 we see that during the first period the activity is very low as in the majority of months the tickets handled does not overcome the limit of 10. Only in one case the number of tickets overcomes the limit of 10. In other words we have a low activity but we do not have any information about the people who interacted with the Issue tracker during this period. By having a look at the ticket openers we see that during the first period and in the majority of the months does not overcome the limit of 5 people. So by combining these data, we could say that approximately 5 people opened 10 tickets so in other words each opener opened 2 tickets. More interesting becomes the study of the second period. Especialy we see that after August 2012 the amount of opened tickets rises up to the maximum value at November 2012 [56]. In November 2012 the amount of openers obtains the maximum value as well [14]. So we see that the same month is the point in the time where both *Tickets Opened* and *Openers* obtain the same value. Approximately we could say that each opener opened 4 tickets. Moving forward in the plot we see that by February 2013 we have 11 tickets opened and 5 openers. By calculating approximately we see that each opener opened 2 tickets. So by comparing the two months we see that in the first month the activity is higher. Apart from that we see that the curvature of the lines *Openers* and *Opened* is not the same. As for the *Openers* we see that the line has more ups and downs in comparison with the *Opened* line. Furthermore we see that by April 2013 we have 14 Openers and 28 Tickets opened. By comparing the activity of April 2013 and November 2012 we see that approximately in November 2012 we have 4 tickets opened per opener and in April 2013 we have 2 tickets opened per opener. So we see that the same amount of people opened less tickets. Unfortunately due error of ViZGrimoire dashboard mentioned in 4.1.4 we do not have any more concrete information about the openers and closers. To that end we stop our analysis here.

4.3 Sloc - Gitstats analysis

Project	Age	Total Files	Total Lines of Code	Total Commits	Authors
Zypper	2398 days	154	36142	2219	26
Open Build Service	2643 days	1505	180712	11086	90

Project	SLOC	Person-Months	Months	Effort/Schedule	Estimated Cost ($)
Zypper	19,704	4.57	0.95 (11.45)	4.79	617,908
Open Build Service	82,881	20.67 (248.08)	1.69 (20.32)	12.21	2,792,668

By using the SLOCCount we count the Source Lines of Code (SLOC) but also the effort made by the developers and the Estimated Cost as well. By having a look at the Table 4.3 and the Table 4.3 we see that for Zypper the total Source Lines of Code are 19.704 [SLOCCount estimation] and 36142 [Gitstats estimation]. As we make the an estimation of the Cost we accept as valid the Total Source Lines of Code provided by SLOCCount. Furthermore during our analysis of the source code repositories the Gitstats generated inaccurate piece of data and that's why we finally used the tools provided by ViZGrimoire for our analysis. Coming back to the analysis of Zypper repository we see that the total Source Lines of Code are 19.704 the Estimated Cost is 617.908 $. As for the Open Build Service we see that the Total Source lines of Code are almost 4 times more than Zypper's one. Furthermore we see that the Cost estimation for Open Build Service repository is 2,792,668 $ which is almost 4 times more than the Zypper's one. Empirically comes out that the more the lines are the higher the Estimated cost is. By using SLOCCount we can see if a project is "value-for-money" and make further study on the productivity of the developers. The primary goal of this Master Thesis is not to analyse the productivity of the developers but to test and use tools that can measure the activity of the developers and provide a cost estimation of the project.

Chapter 5

Conclusions

This Master Thesis tried to dive into the aspects of openSUSE Project. We have to point out that it was very difficult to define the appropiate criteria for our analysis as the openSUSE Project has many aspects and is rapidly grown project. From our previous experience and interaction with the openSUSE Project and after a lot tests we analyzed two repositories, two mailing lists and two issue trackers. Defining the conditions of the analysis was another difficult decision as we had to follow the goals defined before.

5.1 Problems

Before having a look at the goal it is very important to see (briefly) the problems that we faced during our work.

5.1.1 Analysis

- At the beginning our goal was to analyze 3 source code repositories (Kernel,Zypper,Open Build Service). Due to a bug while running CVSAnaly2 [1], we analyzed 2 of the 3 source code repositories.

- It was difficult to find source code repositories that fullfil the Analysis conditions. For instance repositories in gitorious can be mined and analyzed without any problem but gitorious doesn't offer by default a corresponding issue tracker for each repository.

[1] `https://github.com/MetricsGrimoire/CVSAnalY/issues/31`

- Many of the repositories hosted in Github although they have a high number of commits they do not offer an issue tracker.
- We didn't focus on the measurement of the participation part due to the problems mentioned before.

5.1.2 Software

- The programming languages we are using (Python,R) are the main basis of the tools. During the installation of the tools some package dependencies and packages are not provided for the openSUSE distribution.
- The tools VizGrimoireR and VizGrimoireJS were not well documented and only tested in Debian-based distributions.

As we faced the problems mentioned in the Subsections 5.1.2 and 5.1.1 we tried many solutions in order to overcame them. Finally here we provide the solution in the problems. The steps in order to overcome and solve the problems defined before are the following ones :

- Studying the provided how-to's and documentation very carefully
- When facing a problem, search (on the net) for possible solutions.
- Testing of each solution until we find the appropriate one.
- Compiling from scratch packages and dependencies not provided for openSUSE.
- Bug reporting for issues and runtime errors while using the tools.
- Contacting developers (via IRC, mail, or in person)
- Documenting the installation of the tools, so as to be used by the openSUSE Community or from the academic world.

At this point we have to analyze if the goals set are fullfilled finaly. By having a look at the Chapter 2 let's see which of the goals achieved. According to our work done we achieved the Objectives 1,2,3,4. As for the Objective 5 we were not able to achieve it as we had to overcame the problems with the tools and analysis mentioned before. As a result the Objective

5 is part of future work and research. Moving forward to the Objectives we managed to fullfill the Objectives 6 and 7, as we have documented the whole process with success and for future study and usage by the academic world and open source community in general.

5.2 Lessons Learned

As this is a Master Thesis work in this point we have to mention the knowledge and the lessons learned from the Master in Free Software. The following Courses [and all the materials provided set the basis of our work :

- Project Evaluation [2]
- Project Management [3]
- Development Tools [4]

Apart from the Courses my personal work and involvement in the Libresoft Research Group had a significant impact in the Master Thesis. Furthermore I would like to thank in personal (apart from my Prof. Gregorio Robles) the lecturer Daniel Izquierdo, Prof.Israel Herraiz and Alvaro del Castillo [developer in Bitergia [5] for their help and their advices during working on my Master Thesis.

5.3 Future work

As we proposed before more analysis can be done in the productivity of the core team of the developers by analysing the cost estimation of their commits. Furthermore research can be done on the reasons for starting contributing on an open source project or for stop contributing in an open source project.In addition to that the center of the study has to move from the traditional methods to the social coding tools (repositories) as the core members of the development team commit more than sending messages in the corresponding mailing list. Apart from this part of future work the following points could be part of future work but also academic research :

[2] http://docencia.etsit.urjc.es/moodle/course/view.php?id=125
[3] http://docencia.etsit.urjc.es/moodle/course/view.php?id=128
[4] http://docencia.etsit.urjc.es/moodle/course/view.php?id=124
[5] http://bitergia.com/

- After the solution of all the bugs and issues reported, mine and analyze the Kernel repository.
- Focus more on the "Measuring participation" part and the social media interaction of developers and contributors.
- Apply the same methodology in other FOSS distributions and bring to light various deductions
- Compare the package manager of openSUSE with other rpm based package managers by following the same methodology.
- Compare Debian based package manager (apt-get) with an rpm one (zypper or yum).
- Research on the developers activity (e.g bugs solved, commits, participation in repositories, core team of developers etc)

Apendix A

Apendix 1

Bibliography

[1] Jono Bacon .*The Art of Community: Building the New Age of Participation*.O'Reilly, May 2012