WebKit Project – A FLOSS report

Simón Pena Placer

January 21, 2010

Contents

1 Introduction

Nowadays, web browsers are among the most used software applications. Even if the leader application in the field is still Internet Explorer, the Open Source browser Mozilla Firefox has experienced an important growth[12]. Mozilla's layout engine[1], Gecko, is used in many other projects, but another competitor has entered the field: WebKit. WebKit is the Open Source layout engine for Apple's browser Safari but, thanks to being Open Source, it is being used in many different projects with important support from large companies (like said Apple, Google or Nokia).

This work was first thought of as a comparison between Gecko and WebKit's engines, but soon it was clear that analyzing both projects at the same time would be too ambitious. Besides of the fact that both are big projects, Gecko is currently too coupled to the rest of the browser, making its analysis difficult to achieve. Due to that, this work will be focused in analyzing WebKit project.

This analysis will be structured in the following way

- First, the WebKit Project will be presented and some information about its goals and history will be shown.
- After that, we will propose a set of objectives this work will try to achieve.
- The tools, techniques and tests used to gather all the information needed about the project will be explained.
- Results will be provided for the actions performed in the previous step.
- The results obtained will be explained, and final conclusions will be exposed.

1.1 WebKit: the project

WebKit is a layout engine designed to allow web browsers to render web pages. The WebKit engine provides a set of classes to display web content

[1]A layout engine, or rendering engine, is software that takes marked up content (such as HTML, XML, image files, etc.) and formatting information (such as CSS, XSL, etc.) and displays the formatted content on the screen.

in windows, and implements browser features such as following links when clicked by the user, managing a back-forward list, and managing a history of pages recently visited. WebKit was originally created in order to be used as the layout engine for Safari, Apple's Web browser, being derived from Konqueror browser's KHTML software library.

WebKit's WebCore and JavaScriptCore components are available under the GNU Lesser General Public License, and the rest of WebKit is available under a BSD-style license[1, 11, 13].

1.1.1 Goals

WebKit project goals are **compatibility**, **standards compliance**, **stability**, **performance**, **security**, **portability**, **usability** and **hackability**.

Keeping the project Open Source is also a goal, but with BSD-like and LGPL-like licenses to remain usable for free and proprietary applications.

Finally, while project's primary goal is rendering content deployed on the Web, it is considered important being able to serve as a general-purpose display and interaction engine[8].

1.1.2 Project's History

WebKit project's history went through different stages, getting progressively more collaborative.

The code initially created for the KDE Projects KHTML and KJS in 1998, was forked in 2002 to create the frameworks WebCore and JavaScriptCore, respectively. There were no public contributions for a year, until 2003, when the first code changes were released[4].

From 2003 to 2005 Apple and KDE collaboration was difficult, as code interchange was done in large amounts, and sometimes it was even necessary to sign NDA agreements to see Apple's code.

However, in 2005 Apple's WebKit development changed. The WebKit team started reversing Apple-specific changes for platform-independent ones, a CVS repository was opened, and KHTML-WebKit collaboration increased. Eventually, on June 7, 2005 Apple open sourced WebKit, opening access to both repository and bug tracking system.

Since then, collaboration with other parties has been increasing, and it is even speculated that KDE team could move from KHTML to WebKit.

See [13] for more information about WebKit's history.

1.1.3 Projects using WebKit

Among the projects using WebKit, there are Apple's Safari, Apple's Mail, Google Chrome / Chromium, Adium, Epiphany and many others. See [9] for a list of projects using WebKit, and [10] for a list of projects using WebKit/GTK+.

1.2 Motivations

This work is meant to be a revision of the WebKit Project. The project health will be studied, the main, most important developers will be identified and the companies that support them will be presented, too. General information about the project itself will be exposed: project size, number of members –both in the mailing list and in the repository–, number of commits...

We will also try to achieve more specific information: identify the onion model in the project, look at code groups evolution, track developers moving between different companies, and point out how derivative projects may influence WebKit –due to an increase in the number of bugs reported, in the amount of commits or other situations.

Finally, temporal analysis will be performed to search for existing trends, answering the following questions:

- Is the project increasing its activity?
- Is the number of developers increasing or decreasing?
- How is the work balanced between the developers?

2 Methodology

For the analysis of this project, we will take into account the following information sources:

- Repository information, extracted from the commit logs.
- Developer mailing list information, extracted from the mailing list archive.

- Unassigned bugs mailing list information, extracted from the mailing list archive[2]
- Code lines ownership information, extracted from the SCM tool.

2.1 Tools used

To gather that information, the following LibreSoft tools were used: guilty[3], mlstats[4], and cvsanaly[5]. For the full set of LibreSoft tools, see [5]

guilty File-oriented tool that retrieves information about line changes, allowing to identify who owns a line of code, which changes have been made and so on. All the information extracted is stored in a database.

mlstats This tool stores mbox-formatted mailing lists information in a MySQL database.

cvsanaly This tool stores the log from a SCM (CVS, SVN or Git) in a MySQL database.

More information about the tools and how they work can be found at [3, 7].

R[6] was used to access the databases generated with the tools exposed above, generating reports and graphs to be included from this report.

2.2 Webkit data sources

The resources used in this work were updated last time on January 10, 2010. The following data sources were used:

- Subversion repository, located at `http://svn.webkit.org/repository/webkit/trunk/`
- Developers mailing list, located at `https://lists.webkit.org/pipermail/webkit-dev/`

[2]Trying to get the information from the project's Bugzilla was unsuccessful, as the bugs's date could not be extracted.

[3]`http://git.libresoft.es/guilty/`

[4]`https://svn.forge.morfeo-project.org/libresoft-tools/mailingliststat/`

[5]`http://git.libresoft.es/cvsanaly/`

- Unassigned bugs mailing list, located at `https://lists.webkit.org/pipermail/webkit-unassigned/`.

2.3 Queries

Some of the queries available from the Melquiades project[6] at Flossmetrics will be used for this work. They can be found at [2]

2.3.1 Quantitative analysis

The first type of queries we will analyze are quantitative, and will give an idea about the size of this project. Queries like obtaining the number of commits, committers, files committed will be performed, with regards to the repository. The mailing list will be analyzed querying the number of mails sent, people subscribed to the mailing list and lifespan of the mailing list. Finally, the unassigned bugs mailing list will be used to check trends in identifying bugs.

Repository

Number of commits This query counts the number of commits in the repository.

```
SELECT COUNT(s.id)
FROM scmlog s;
```

Number of committers This query counts the number of different people committing to the repository.

```
SELECT COUNT(DISTINCT s.committer_id)
FROM scmlog s;
```

Number of files committed This query counts the number of different files handled in the repository.

```
SELECT COUNT(DISTINCT a.file_id)
FROM actions a;
```

[6]`http://melquiades.flossmetrics.org/`

Most commits by user This query groups commits done by the same nickname, showing the number of commits made through the lifespan of the project. It is a good indicative of the relative importance of the committers.

```
SELECT people.name, COUNT(*) AS commit_count
FROM people LEFT JOIN scmlog ON ( people.id = scmlog.committer_id )
GROUP BY scmlog.committer_id
ORDER BY commit_count
DESC LIMIT 20;
```

It should be noted that this query does not handle same users using different nicknames for committing. That will be handled with another query.

Most commits by user last year This query is a modification of the previous one, considering only the commits done during the last year. It can give us some insight of the current state of the project.

```
SELECT people.name, COUNT(*) AS commit_count
FROM people LEFT JOIN scmlog ON ( people.id = scmlog.committer_id )
WHERE year(scmlog.date)=2009
GROUP BY scmlog.committer_id
ORDER BY commit_count
DESC LIMIT 20;
```

Again, this query does not handle same users using different nicknames.

Lifespan of the repository This query retrieves the dates of the first and last commit to the repository, and its lifespan in years.

```
SELECT YEAR(MIN(date)), YEAR(MAX(date)),
YEAR(MAX(date))-YEAR(MIN(date))
FROM scmlog;
```

Mailing list

Number of mails This query counts the number of messages sent to the mailing list.

```
SELECT COUNT(m.message_ID)
FROM messages m;
```

Number of people contributing to the mailing list This query counts the number of people who have sent emails to the mailing list. However, people using different nicknames will be count more than once.

```
SELECT COUNT(DISTINCT p.people_ID)
FROM messages_people p;
```

Most emails by user This query groups emails sent by the same nickname, showing the number of emails sent through the lifespan of the project. It is a good indicative of the relative importance of the poster.

```
SELECT p.username, COUNT(*) as emails
FROM messages_people mp LEFT JOIN messages m
ON (m.message_id = mp.message_id )
LEFT JOIN people p ON (p.people_ID = mp.people_ID )
GROUP BY p.username
ORDER BY emails DESC
LIMIT 20;
```

Most emails by user last year This query is a modification of the previous one, considering only the emails sent during the last year. It can give us some insight of the current state of the project.

```
SELECT p.username, COUNT(*) as emails
FROM messages_people mp LEFT JOIN messages m
ON (m.message_id = mp.message_id )
LEFT JOIN people p ON (p.people_ID = mp.people_ID )
WHERE YEAR(m.arrival_date) = 2009
GROUP BY p.username
ORDER BY emails DESC
LIMIT 20;
```

Users with more than one account This query shows usernames and email addresses for those users who have more than one account participating to the list.

```
SELECT username, email_address
FROM people
```

```
WHERE username IN (
SELECT username
FROM people
GROUP BY username
HAVING COUNT(*) > 1)
ORDER BY username;
```

Lifespan of the mailing list This query retrieves the dates of the first and last commit to the mailing list, and its lifespan in years.

```
SELECT YEAR(MIN(m.arrival_date)), YEAR(MAX(m.arrival_date)),
YEAR(MAX(m.arrival_date))-YEAR(MIN(m.arrival_date))
FROM messages m;
```

Unassigned bugs mailing list

All-time unassigned bugs While this query does not take into account invalid bugs (duplicated, mistakenly reported, etc.), can be used to understand the size of the project

```
SELECT COUNT(*)
FROM messages;
```

2.3.2 Qualitative analysis

The following queries will provide more insight about the behavior and health of the project. Number of commits by date, number of email messages by date, active and inactive users, all-time trend, last 6 months trend or companies influence to the project will be presented.

Repository

People with different usernames on the repository When the repository queries were exposed, it was noted that people committing from different usernames would be taken into account as different people. The following query deals with it, considering they keep the username the same, just changing the domain name.

```
SELECT SUBSTRING_INDEX(p.name,"@",1) AS username,
COUNT(*) AS count
FROM people p LEFT JOIN scmlog ON (p.id = scmlog.committer_id)
GROUP BY username
ORDER BY count DESC
LIMIT 20;
```

Activity by date on the repository The following query shows the monthly number of commits to the repository. It is a good indicator about the project's health, allowing to tell if it is growing, stable or declining.

```
SELECT YEAR(date) AS year, MONTH(date) AS month, COUNT(*) AS count
FROM scmlog
GROUP BY year, month;
```

Average monthly commits by year The following query shows the average number of commits in a month for each of the years of the project's life.

```
SELECT g.year, AVG(g.numcommits)
FROM
(SELECT YEAR(date) AS year, MONTH(date) AS month,
COUNT(id) AS numcommits
FROM scmlog
GROUP BY year, month) g
GROUP BY g.year;
```

Individual commits after November 2007 The following query is very interesting to display a change of policies in the repository access. Since November 2007, people committing to the repository must use a full email address instead of an user name.

```
SELECT p.name, MIN(date), MAX(date), COUNT(*) AS commits
FROM people p LEFT JOIN scmlog l ON (p.id = l.committer_id)
WHERE date >= DATE('2007-11-1')
AND p.name NOT LIKE '%@%'
GROUP BY p.name
ORDER BY commits desc;
```

Activity by company on the repository The following query displays the amount of commits done by developers affiliated to a company.

```
SELECT SUBSTRING_INDEX(people.name,"@",-1) AS company,
MIN(date), MAX(date), COUNT(*) AS commits
FROM people LEFT JOIN scmlog ON (people.id = scmlog.committer_id)
WHERE people.name LIKE '%@%'
GROUP BY company
ORDER BY commits DESC;
```

Activiy by day of week on the repository The following query allows to tell how is the work distributed by the day of the week.

```
SELECT YEAR(date) AS year, DAYOFWEEK(date) AS day,
COUNT(*) AS COMMITS
FROM scmlog
GROUP BY year, day;
```

Commits by time of day on the repository The following query lets us tell how is the work distributed by the time of the day.

```
SELECT YEAR(date) AS year, HOUR(date) as hour,
COUNT(*) AS commits
FROM scmlog
GROUP BY year, hour;
```

It should be noted that the commit time stored is GMT.

Mailing list

Activity by date on the mailing list The following query shows the monthly number of mails sent to the mailing list. It is another indicator of the project's health.

```
SELECT YEAR(arrival_date) AS year, MONTH(arrival_date) AS month,
COUNT(*) AS count
FROM messages
GROUP BY year, month;
```

Activity by company on the mailing list The following query shows the number of emails sent to the mailing list by domain name (filtering out "gmail.com" should limit the results to actual companies).

```
SELECT people.domain_name, COUNT(*) as emails
FROM people LEFT JOIN messages_people
ON (people.people_ID = messages_people.people_ID)
GROUP BY people.domain_name
ORDER BY emails DESC
LIMIT 20;
```

Activity by day of week on the mailing list The following query allows to tell how is the work distributed by the day of the week.

```
SELECT YEAR(date) AS year, DAYOFWEEK(messages.arrival_date) AS day,
COUNT(*) AS emails
FROM messages
GROUP BY year, day;
```

Activity by time of day on the mailing list The following query lets us tell how is the work distributed by the time of the day.

```
SELECT YEAR(date) AS year, HOUR(messages.arrival_date) AS hour,
COUNT(*) AS emails
FROM messages
GROUP BY year, hour;
```

Unassigned bugs mailing list

Unassigned bugs by month The following query shows the number of bugs by month, which gives us an idea about the trends in finding bugs.

```
SELECT YEAR(arrival_date) AS year, MONTH(arrival_date) AS month,
COUNT(message_id)
FROM messages
GROUP BY year, month;
```

Average monthly unassigned bugs by year The following query shows the average number of bugs in a month for each of the project's years.

```
SELECT g.year, AVG(g.numcommits)
FROM
(SELECT YEAR(date) AS year, MONTH(date) AS month,
COUNT(id) AS numcommits
FROM scmlog
GROUP BY year, month) g
GROUP BY g.year;
```

3 Results

In this section, results will be exposed that follow the previously explained methodology. They answer issues like project's size, project's development model, companies supporting development, and most valuable developers–contributors. For that matter, results will be categorized in repository-related, developers' mailing list-related and unassigned bugs' mailing list-related.

3.1 Repository

In order to measure the repository size, a set of variables are taken into account: number of commits, number of committers, number of files under version control, number of lines of code and years of activity. Table 1 presents that information. In figure 1 the distribution of programming languages in the project is shown[7].

Having presented the data about the project size, the following results will help identify the most important committers. Table 2 shows the *All-time* top 20 committers. However, a quick glance at it lets us find several committers sharing their username, but using different email addresses. Results from table 3 show that information corrected, so users get their contribution added up even if they have used different email addresses to commit. We can identify one user, "**darin**", as the one who has done the most commits: 4876.

[7]Number of lines of code and programming language distribution were generated using David A. Wheeler's 'SLOCCount'.

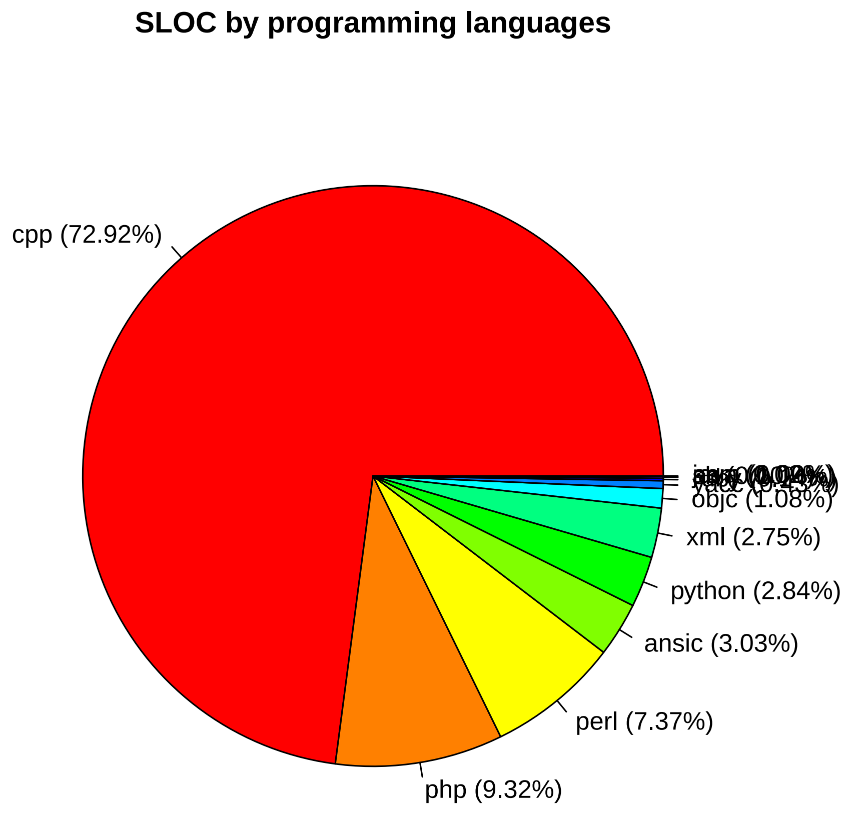

Figure 1: Programming language distribution

Concept	Count
Number of commits	44143
Number of committers	195
Number of files under version control	80860
Number of lines	824955
Years of activity	9

Table 1: Brief summary of the repository's activity

	Committer	Commit count
1	darin	3583
2	hyatt	2158
3	eric@webkit.org	1967
4	mjs	1620
5	hausmann@webkit.org	1174
6	rjw	1104
7	darin@apple.com	1076
8	kocienda	958
9	mitz@apple.com	945
10	mrowe@apple.com	941
11	cblu	885
12	hyatt@apple.com	884
13	eseidel	860
14	weinig@apple.com	796
15	aroben@apple.com	754
16	ggaren	726
17	andersca	715
18	ap@webkit.org	703
19	andersca@apple.com	661
20	sullivan	660

Table 2: Top 20 committers. Multiple accounts ignored

In order to have information about last trends, last year top 20 committers are shown in table 4. Our previously identified *top committer*, "darin", is now located at the 5th place, being "eric@webkit.org" the developer who has

	Committer	Commit count
1	darin	4876
2	hyatt	3042
3	eric	1967
4	mjs	1842
5	hausmann	1417
6	andersca	1376
7	ggaren	1287
8	weinig	1263
9	ap	1252
10	aroben	1164
11	rjw	1104
12	mitz	968
13	kocienda	958
14	mrowe	941
15	oliver	900
16	cblu	885
17	eseidel	860
18	beidson	759
19	sullivan	724
20	adele	695

Table 3: Top 20 committers. Multiple accounts grouped

done the biggest amount of commits last year: 1646. His progression is worth noting, as he was ranked 3^{rd} all-time, with 1967 commits, which means he has done $83,68\%$ of his work last year.

	Committer	Commit count
1	eric@webkit.org	1646
2	abarth@webkit.org	474
3	hausmann@webkit.org	440
4	kov@webkit.org	387
5	darin@apple.com	385
6	mrowe@apple.com	364
7	simon.fraser@apple.com	357
8	mitz@apple.com	334
9	hyatt@apple.com	322
10	oliver@apple.com	320
11	dglazkov@chromium.org	317
12	xan@webkit.org	312
13	weinig@apple.com	274
14	levin@chromium.org	255
15	ggaren@apple.com	246
16	aroben@apple.com	238
17	ap@webkit.org	235
18	andersca@apple.com	231
19	barraclough@apple.com	220
20	pfeldman@chromium.org	213

Table 4: Top 20 committers during 2009

Complete email addresses are presented in table 4. After November 2007, users committing to the repository stopped using just their username, and started using an email address for commits. Table 5 shows who did the last commits using the old username schema, and how after day 12 nobody used the username alone.

After November 2007, it is possible to query the repository to get the number of commits done by users on behalf of their company. If we assume that an *username@domain* is working for a company after which the domain is named, the table 6 shows how the commits are distributed among the companies participating in WebKit project.

	Committer	First contribution	Last contribution	Commit count
1	hausmann	2007-11-07 11:32:07	2007-11-10 23:24:34	77
2	aroben	2007-11-01 02:22:35	2007-11-08 00:45:58	20
3	antti	2007-11-03 01:23:43	2007-11-12 02:54:55	14
4	alp	2007-11-01 19:37:36	2007-11-06 03:53:16	11
5	oliver	2007-11-02 00:30:25	2007-11-12 09:00:29	11
6	eseidel	2007-11-06 05:55:25	2007-11-12 01:34:37	10
7	kmccullo	2007-11-01 01:06:56	2007-11-09 09:29:20	10
8	kevino	2007-11-01 22:36:00	2007-11-06 07:39:22	9
9	mjs	2007-11-01 06:02:30	2007-11-07 08:23:25	9
10	ddkilzer	2007-11-03 07:58:34	2007-11-07 22:54:57	7
11	sfalken	2007-11-08 08:40:08	2007-11-09 18:16:39	7
12	ap	2007-11-01 10:21:30	2007-11-06 19:10:22	7
13	ggaren	2007-11-01 09:36:58	2007-11-05 22:56:09	6
14	weinig	2007-11-01 22:33:33	2007-11-10 00:34:19	5
15	tristan	2007-11-05 23:43:45	2007-11-10 00:51:48	4
16	justing	2007-11-01 00:23:26	2007-11-07 02:13:47	4
17	mitz	2007-11-01 16:30:23	2007-11-02 07:27:29	3
18	hyatt	2007-11-01 04:05:41	2007-11-09 20:57:39	3
19	adele	2007-11-01 20:01:09	2007-11-06 20:53:18	3
20	adachan	2007-11-06 01:02:50	2007-11-06 03:10:05	2
21	bdakin	2007-11-07 06:31:10	2007-11-07 06:31:10	1
22	andersca	2007-11-01 02:16:23	2007-11-01 02:16:23	1
23	honeycutt	2007-11-10 03:28:57	2007-11-10 03:28:57	1

Table 5: Commits without providing an email address after November, 2007

Analyzing the results, we can see how Apple, through *Apple.com* and *WebKit.org* domains, leads the development. Google, through *Chromium.org* – an Open Source project which serves as the foundation for the Google Chrome browser– also has an important contribution, but far from Apple's. Other companies have just minor contributions.

	Company	First contribution	Last contribution	Commit count
1	apple.com	2007-10-25 07:01:13	2010-01-04 13:01:40	12005
2	webkit.org	2007-11-07 06:24:17	2010-01-04 12:37:24	8753
3	chromium.org	2008-10-02 00:34:16	2009-12-31 01:12:15	1831
4	nokia.com	2009-06-19 16:28:38	2010-01-03 20:36:38	63
5	google.com	2009-09-01 20:00:54	2009-12-17 20:18:40	30
6	torchmobile.com	2009-08-13 20:04:25	2009-10-14 17:48:41	10
7	forwardbias.in	2009-11-17 06:39:31	2009-12-24 20:41:36	8

Table 6: Number of commits by the company an user is affiliated to

3.1.1 Temporal Analysis

To determinate how the project is evolving through time, we have analyzed how the distribution of commits varies. We will present a monthly, weekly and daily analysis through the 7 years of life of the project.

Figure 2 presents the monthly analysis. The first subchart depicts the average monthly commits growing with time[8]. The second subchart represents the seasonal variation extracted from the data: each year presents one major peak of activity, with two smaller ones, and two important drops. The drops can be easily matched with holidays, with one being around Christmas time, and the other in the summer. The major peak is located around November, and could be due to the project's life cycle.

Figures 3 and 3 show the weekly and daily analysis, respectively. Together with the monthly exposed before, they show how WebKit's development model is company-led, as work load is concentrated during the office time, between monday and friday. Weekly analysis shows how the peak of activity is concentrated at the beginning of the week, and decreases slowly after that,

[8]The line dropping in the 2010 is due to the data being extracted in the first week of the year

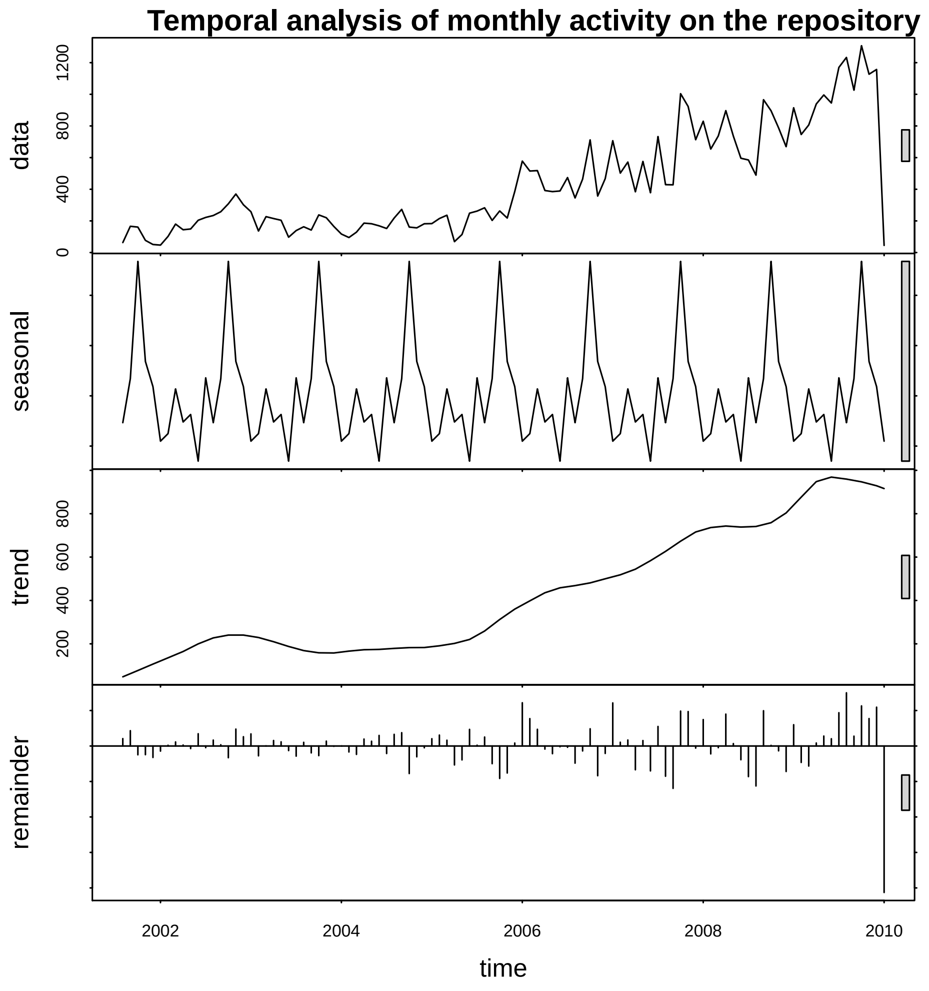

Figure 2: Evolution of yearly activity in the repository

with little to none work done on Saturdays and Sundays. Daily analysis is more interesting, as it shows how the activity increases through the day, decreases a little in the lunch break, and keeps increasing until the office time is over. Then it goes down to almost zero. The bottom line with all of these analysis is that the project is continuously increasing its activity.

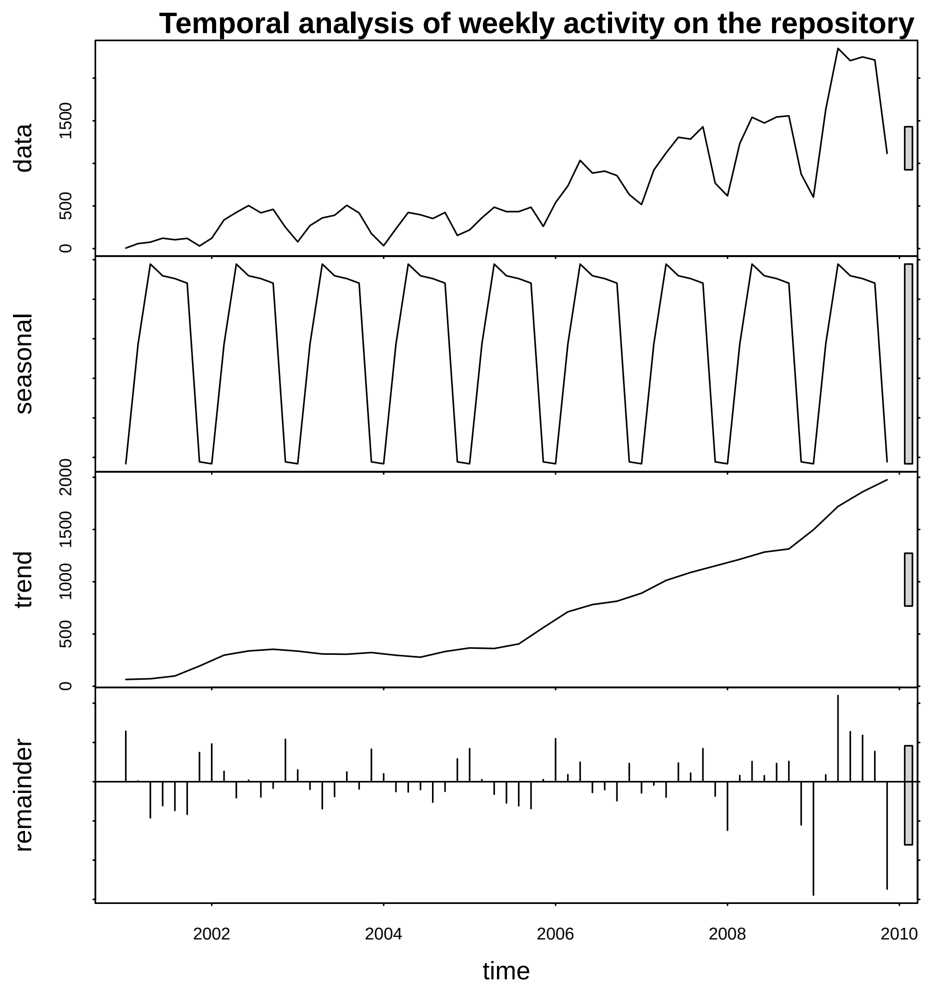

Figure 3: Evolution of weekly activity in the repository

In order to finish the repository analysis, figure 5 shows the Lorenz Curve, which allows us to tell how the work is distributed among the project devel-

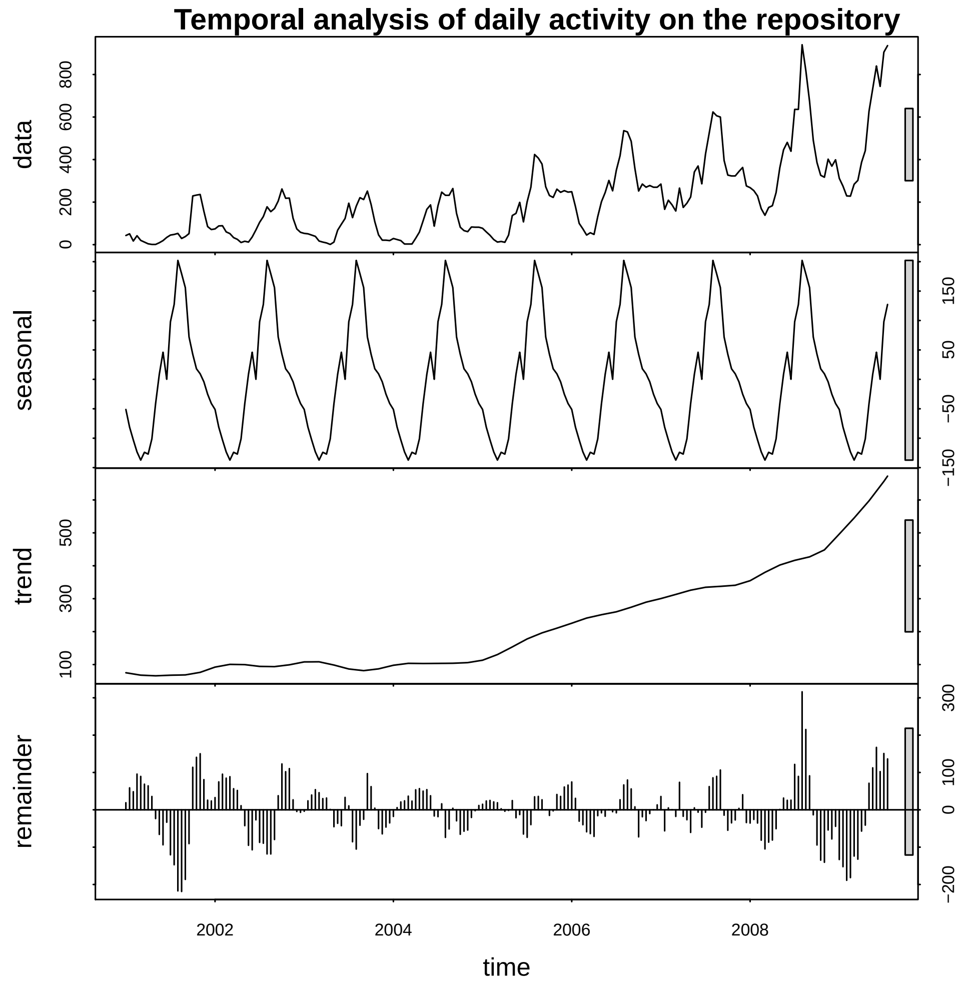

Figure 4: Evolution of daily activity in the repository

opers. The Gini coefficient is 0.7057624. Those results mean that 70% of the work is done by the 20% of the people.

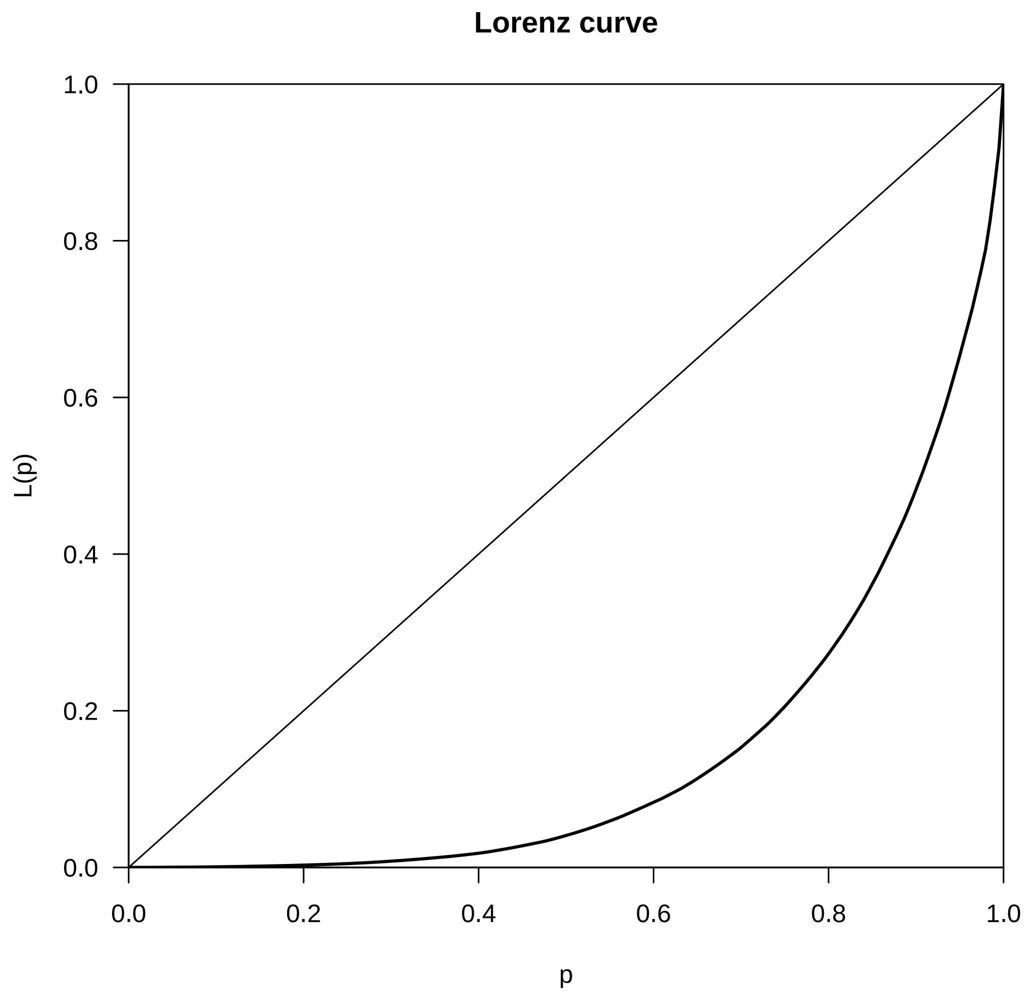

Figure 5: Lorenz curve for the repository

3.2 Developers' mailing list

As we did with the repository, a set of variables were taken into account to evaluate the developer's mailing list. Figure 7 presents the number of emails sent, number of people writing to the list and mailing list lifespan.

Concept	Count
Emails sent to the list	9760
Unique email addresses writing to the list	1199
Different user names writing to the list	1132
Years of activity	5

Table 7: Brief summary of the developers' mailing list

Again, after having presented the mailing list summary, users contribution in the mailing list is measured now. Table 8 shows the *all-time* top 20 posters to the mailing list. We can see the same user "darin" being the one who contributed the most, with 623 emails sent. However, "mjs" is closer here, with 605, just 18 less than him.

Table 9 shows last year's top 20 posters, so we can have current info about the mailing list. In this case, "darin" is still the leader with 288 emails sent, and with "mjs" effectively close. If we search for "eric", we found him with 123 emails sent. As he had sent 170 in the all-time rankings, he has done $72,35\%$ of his contribution last year.

As opposed to the repository, users writing to the mailing list have been sending a full email address all the time. Even if several users had more than one email address, as shown in table 10, the queries used previously have taken it into account.

Thanks to the fact that it is possible to retrieve the full email address for all the project's history, the table 11, which shows emails sent from a given domain address, is really interesting to see which companies supported WebKit's development. Ignoring free available domain names like "gmail.com" or "yahoo.com", Apple (*apple.com* and *webkit.org*), Google (*chromium.org* and *google.com*) or KDE (*kde.org*) are interesting examples about entities interested in WebKit.

3.2.1 Temporal Analysis

Like we did with the repository, temporal analysis of the mailing list evolution through the years is a good tool to tell about WebKit's health. Again, we will present a monthly, weekly and daily analysis.

	Username	Email count
1	darin	623
2	mjs	605
3	ddkilzer	307
4	mrowe	209
5	aroben	206
6	mike.emmel	173
7	eric	170
8	hyatt	165
9	ggaren	160
10	abarth	147
11	pkasting	125
12	bfulgham	115
13	ap	106
14	oliver	102
15	jorlow	98
16	zecke	91
17	kevino	83
18	jackwootton	80
19	jhaygood	73
20	vniles	70

Table 8: Top 20 posters

	Username	Email count
1	darin	288
2	mjs	233
3	abarth	132
4	eric	123
5	pkasting	106
6	jorlow	96
7	ddkilzer	86
8	mrowe	78
9	ggaren	72
10	aroben	69
11	ap	67
12	atwilson	66
13	hyatt	64
14	lastguy	61
15	oliver	54
16	vniles	54
17	bfulgham	48
18	zherczeg	48
19	levin	40
20	zecke	34

Table 9: Top 20 posters during 2009

	Username	Email address
1	aa	aa@google.com
2	aa	aa@chromium.org
3	achats	achats@avvanta.com
4	achats	achats@blarg.net
5	alex	alex@dojotoolkit.org
6	alex	alex@milowski.org
7	alex	alex@milowski.com
8	alex	alex@ialexi.com
9	alex	alex@sdpimail.com
10	alp	alp@atoker.com
11	alp	alp@nuanti.com
12	andersca	andersca@mac.com
13	andersca	andersca@apple.com
14	christian	christian@twotoasts.de
15	christian	christian@plesslweb.ch
16	dan	dan@dancryer.com
17	dan	dan@cdslash.net
18	danw	danw@nekotech.com
19	danw	danw@gnome.org
20	darin	darin@apple.com
21	darin	darin@google.com
22	darin	darin@chromium.org

Table 10: Email addresses for users with more than one email address

	Domain name	Email count
1	apple.com	2329
2	gmail.com	2244
3	webkit.org	665
4	chromium.org	423
5	google.com	337
6	yahoo.com	221
7	mac.com	169
8	kde.org	134
9	kilzer.net	110
10	selfish.org	91
11	inf.u-szeged.hu	86
12	trolltech.com	83
13	theolliviers.com	83
14	nokia.com	73
15	sympatico.ca	65
16	hotmail.com	64
17	hatcher.name	62
18	atoker.com	57
19	Sun.COM	56
20	lkcl.net	55

Table 11: Number of emails sent by company employees

Figure 6 shows how the average monthly activity evolved through the project's life. As in the repository case, there is a major peak, just before the summer time, and peaks after each holiday season. It is interesting to note how Christmas time suppose a complete halt in communications.

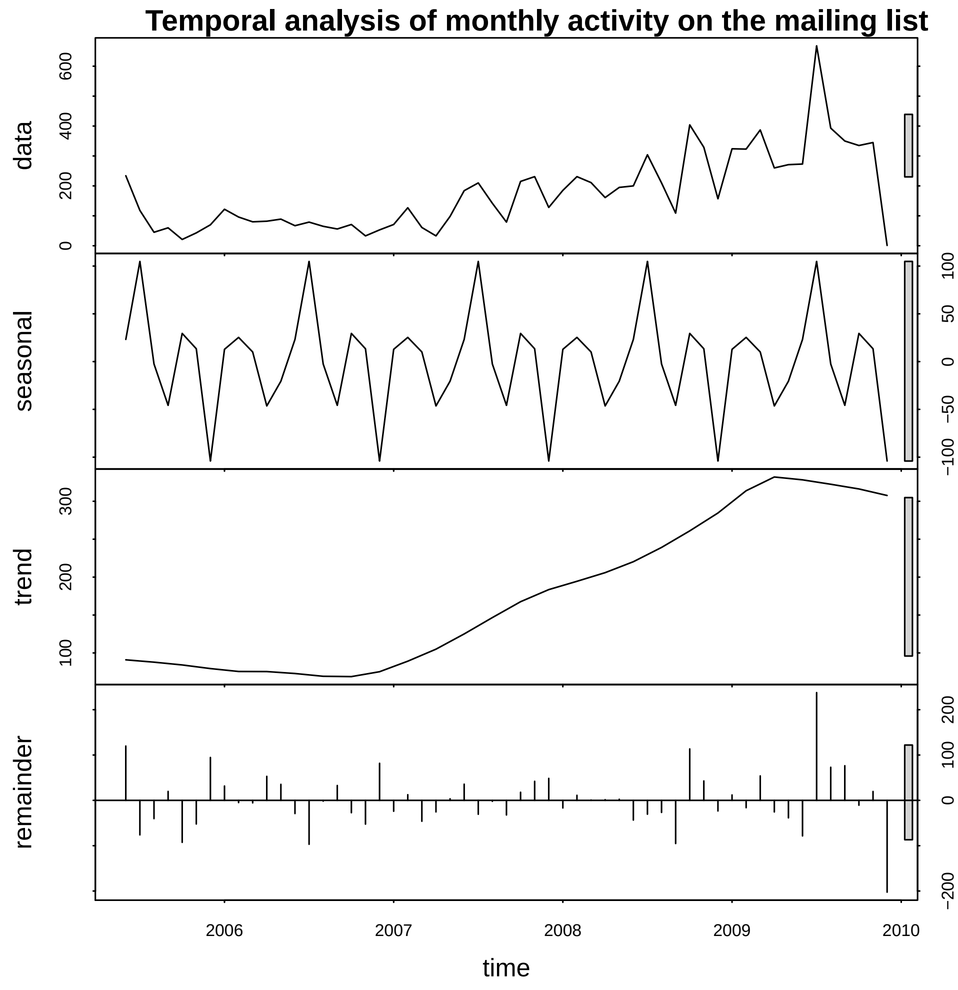

Figure 6: Evolution of yearly activity in the developers' mailing list

Figures 7 and 7 show the evolution in weekly and daily activity, respectively. The weekly one shows an activity which increases through the week, going to almost-zero on Sunday. It is interesting to note that mailing list activity looks bigger than repository activity on Saturdays. Regarding to daily activity, it can be said that there is activity answering them sometime in the morning. After that, it halts around the lunch time, only to increase until the end of the office time. Surprisingly, it looks like there is another peak of activity, maybe at home now, after that break.

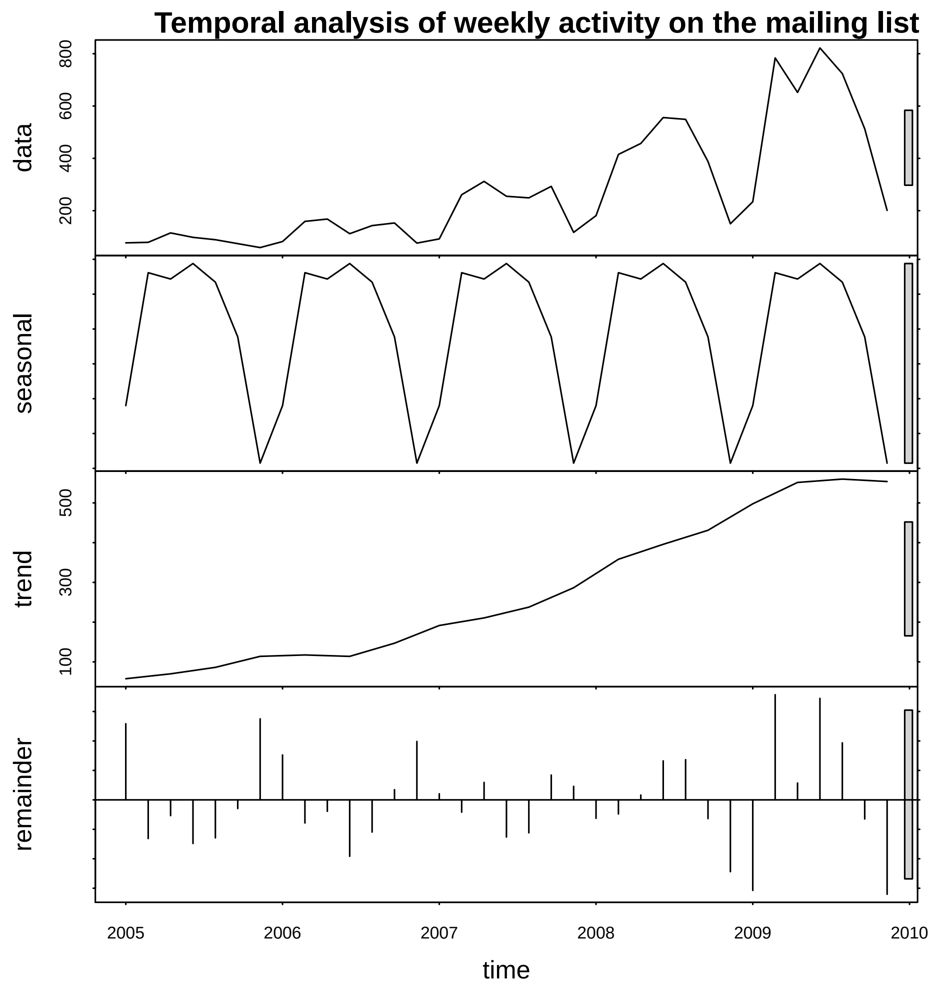

Figure 7: Evolution of weekly activity in the developers' mailing list

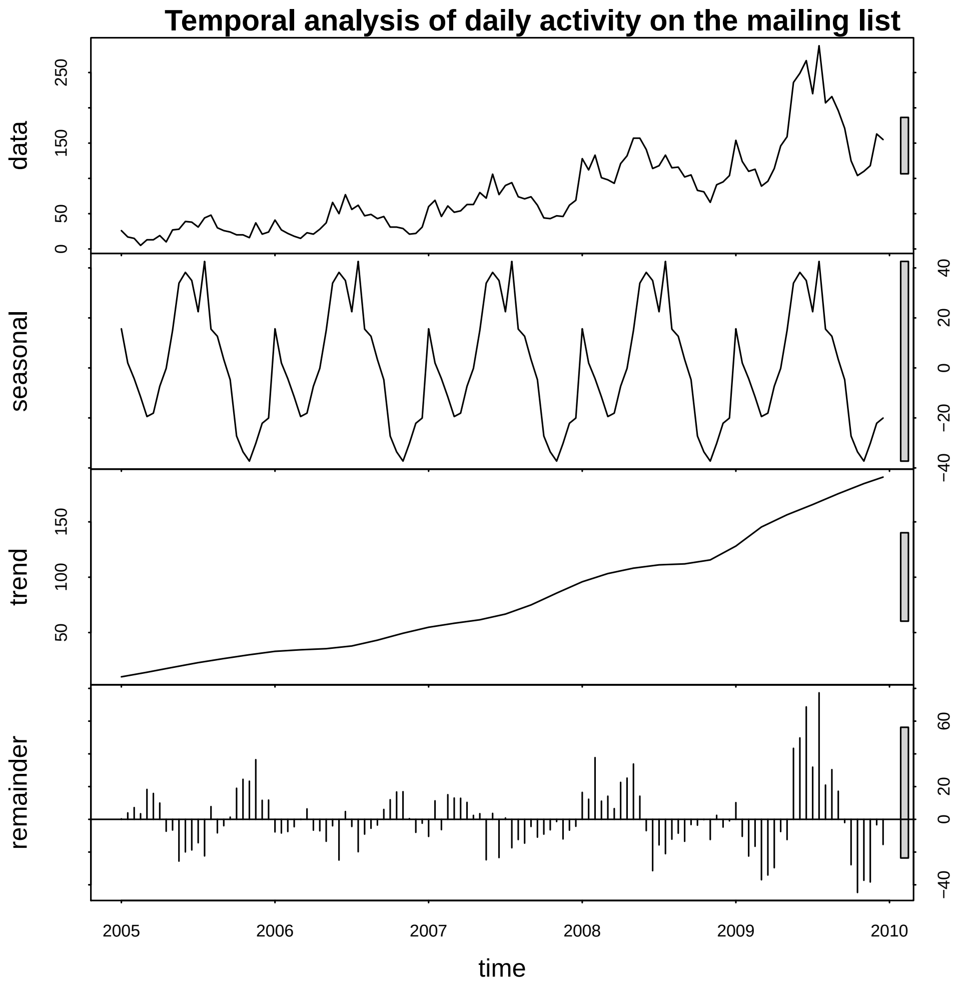

Figure 8: Evolution of daily activity in the developers' mailing list

3.3 Unassigned bugs mailing list

Every time a bug is opened, it gets sent to this mailing list. Updates and changes regarding those bugs also send an email to the list[9]. There are 160342 emails in this list.

Figure 9 shows the average monthly activity in the mailing list. While at the beginning it has not grown as much as the developers mailing list, year 2009 seems to be a great activity explosion. Regarding to the seasonal activity, it looks like *bug hunting* activity also decreases during the Christmas time, just as it was professionally carried, too. After that, there is another drop in activity around February-March, and then it stays more or less constant until Christmas time again. It is interesting, as the expected drop in summer time does not appear so highlighted here.

4 Conclusions and future research

In this report, an analysis about the WebKit Open Source project has been made. Most important contributors –all time and last year's, both committers and mailing list writers– have been identified. Companies behind the development have been confirmed: it was clear that Apple was the main supporter of WebKit, but Google's contribution, while far from Apple's, is also noteworthy. However, Nokia's contribution was not that clear.

Also regarding to companies, it was also made clear that the development is being carried out in an enterprise-mode. Both code development and mailing list handling is being made during the office time, and holiday periods are clear through the project's life.

About project's health and activity, two facts were presented. The Lorenz Curve and Gini coefficient showed us that development was concentrated on a reduced group of people (typically Apple's developers), while temporal series analysis showed us how the average activity was increasing steadily.

However, interesting research could still be done. The first one is related to Gecko: the origin of this work. Is Gecko increasing its activity at the same rate as WebKit? An analysis of Mozilla's code should be carried on to be able to compare both.

Another question is related to milestones in the project. While we identified November 2007 as a milestone when analyzing the repository, searching

[9]`http://webkit.org/contact.html`

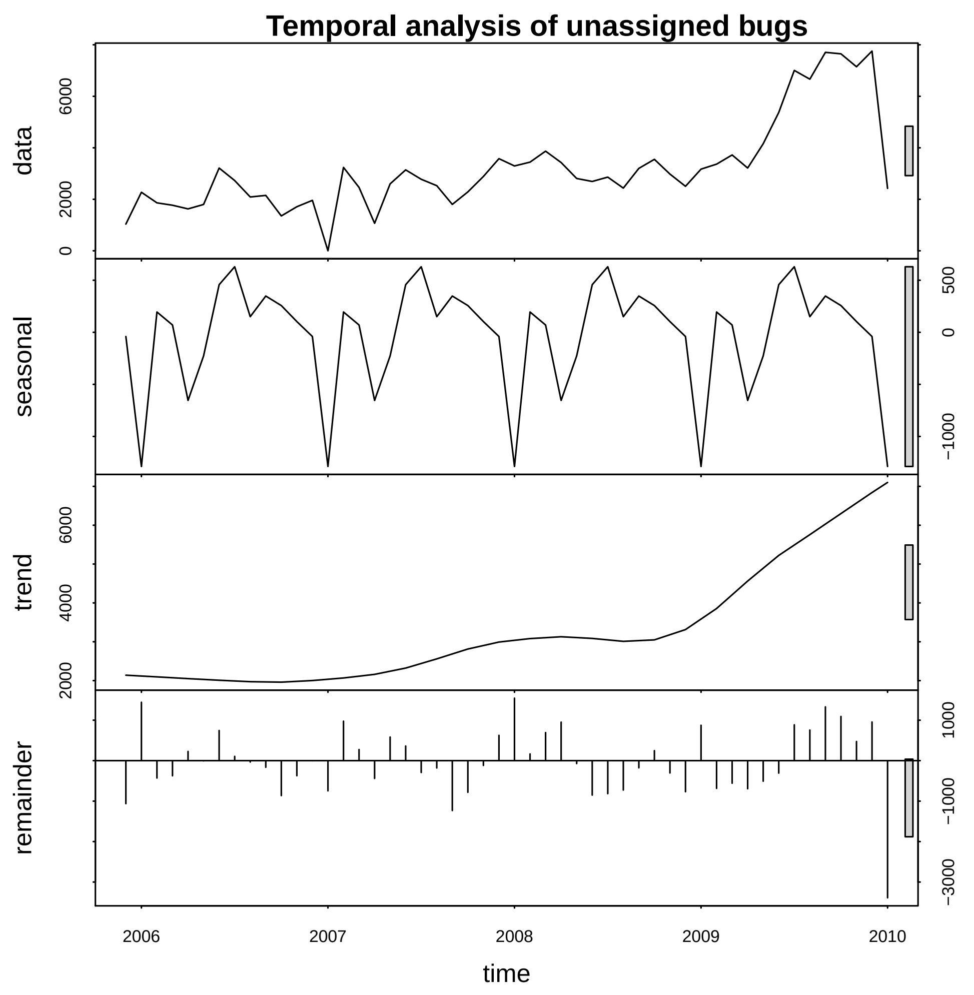

Figure 9: Evolution of the monthly activity on the unassigned bugs mailing list

the Web only showed that the WebKit team achieved HTML5 media support, that Android appeared and chose WebKit and that Committer and Reviewer policy changed[10]

Finally, the analysis could be extended to cover different types of actions in the commits, the Lorenz Curve could be calculated for different groups of developers, and more tests could be done.

[10]Surfin' Safari - The WebKit Blog - 2007 - November `http://webkit.org/blog/date/2007/11/`

References

[1] Apple Developer Connection. Open Source – Internet & Web – WebKit. http://developer.apple.com/opensource/internet/webkit.html, 2009.

[2] Flossmetrics. queries [Melquiades Wiki]. http://melquiades.flossmetrics.org/wiki/doku.php?id=queries, 2009.

[3] Flossmetrics. start [Melquiades Wiki]. http://melquiades.flossmetrics.org/wiki/doku.php, 2009.

[4] Kde.org. (fwd) Greetings from the Safari team at Apple Computer. http://lists.kde.org/?m=104197092318639, 2003.

[5] LibreSoft. start [GSyC/LibreSoft tools]. http://tools.libresoft.es/, 2010.

[6] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2009. ISBN 3-900051-07-0.

[7] MASTER ON FREE SOFTWARE. Practical Approach: Analysing Libre Software Communities. http://gsyc.escet.urjc.es/moodle/mod/resource/view.php?id=2961, 2009.

[8] WebKit. The WebKit Open Source Project – WebKit Project Goals. http://webkit.org/projects/goals.html, 2009.

[9] WebKit. Applications using WebKit – WebKit. http://trac.webkit.org/wiki/Applications%20using%20WebKit, 2010.

[10] WebKit. Applications using WebKit/GTK+ – WebKit. http://trac.webkit.org/wiki/ApplicationsGtk, 2010.

[11] WebKit. The WebKit Open Source Project. http://webkit.org, 2010.

[12] Wikipedia. Usage share of web browsers – Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Usage_share_of_web_browsers, 2010.

[13] Wikipedia. WebKit – Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/WebKit, 2010.