Collaboration Networks in Software Development: Perspectives from Applying different Granularity Levels using Social Network Analysis - Research in progress

Miguel Angel Fernandez, Gregorio Robles and Jesus Gonzalez Barahona

GSyC/LibreSoft, Rey Juan Carlos University

(ma.fernandezsa@alumnos, grex@)urjc.es; jgb@bitergia.com

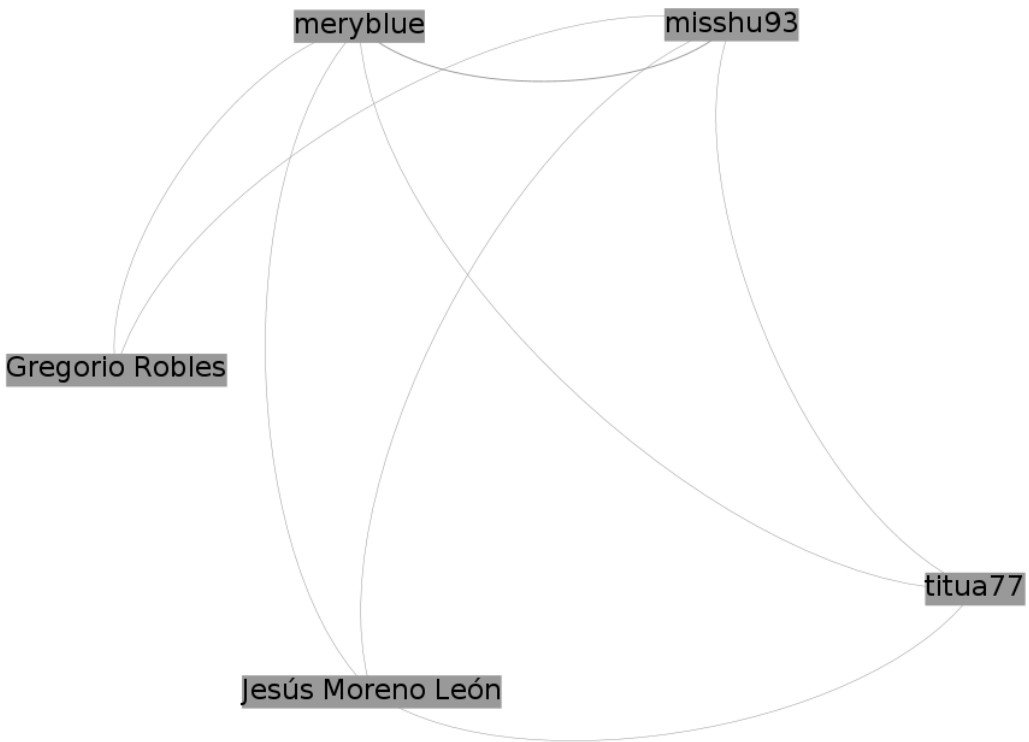
July 5, 2015

Motivation

- Large software projects may involve a lot of developers (Sometimes thousands of them!).
- Our interest is to understand better how developers collaborate and how this interaction evolves over time.
- We opted to study Free/Libre and Open Source Software (FLOSS) projects due to the easy, public data availability in websites like GitHub.

How do we study collaborations?

- Using Social Network Analysis tecniques we get collaboration networks.



In these network graphs:

Nodes = Developers

Two developers (nodes) are connected if they have collaborated together.

Edges = Collaborations

Edges width represents the amount of collaboration (The wider the edge, the greater is the number of interactions between those two nodes).

Up to now...

- In most social network studies the resulting network is based on file/module-based data.
- If there is a collaboration between two developers in the same file/module, these developers are connected.

A different point of view

- When there are tens of files in a module or thousands of lines in a file, did collaboration really exist?
- We think the resulting collaboration network graph depends heavily on the granularity level that is considered.

A different point of view: New-level analysis

- We've been working to obtain collaboration graphs at function/method level.
- In these graphs, two developers collaborate if they have modified the same function in a given time period.
- Excluding large fuctions/methods, we think this new point of view can help us to understand better this analysis.

Methodology: Our tool

- In LibreSoft, our department at Rey Juan Carlos University, we have developed a python script named GraphDataCreator
- That studies changes in a given Git-tracked repository.
- Using the commit history of all contributors in a specified period of time.

Detailed algorithm

Case of study: Gedit

- We used the program to study the evolution of GNOME-text editor Gedit *.
- The considered date range for this study goes from the very beginning of the project to this year.
- To extract data from wide time periods we have developed a super-script that automatically divides large date ranges into smaller periods. * Data extracted from GitHub repository: https://github.com/GNOME/gedit

Summing up...

Date range

- Goes form April 15, 1998 until April 15, 2015.
- Divided into six-month periods

Resulting data

- Two different graphs: for each date range, an in-file and an in-method network.
- Statistic parameters referred to networks, such as betweeness centrality and clustering coefficent.

Multiple Columns

Heading

1. Statement
2. Explanation
3. Example

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer lectus nisl, ultricies in feugiat rutrum, porttitor sit amet augue. Aliquam ut tortor mauris. Sed volutpat ante purus, quis accumsan dolor.

Table

Treatments	**Response 1**	**Response 2**
Treatment 1	0.0003262	0.562
Treatment 2	0.0015681	0.910
Treatment 3	0.0009271	0.296

Table : Table caption

Theorem

Theorem (Mass–energy equivalence)

$E = mc^2$

Verbatim

Example (Theorem Slide Code)

```
\begin{frame}
\frametitle{Theorem}
\begin{theorem}[Mass--energy equivalence]
$E = mc^2$
\end{theorem}
\end{frame}
```

Figure

Uncomment the code on this slide to include your own image from the same directory as the template .TeX file.

Citation

An example of the `\cite` command to cite within the presentation:

This statement requires citation [Smith, 2012].

References

John Smith (2012)

Title of the publication

Journal Name 12(3), 45 – 678.

Titulo

- Punto 1
- Punto 2
- Punto 3

The End