Comments in Git repositories analysis

How has the number of TODO tags evolved within the past years

C. Blanquet, K. Dousse, D. Mazzoleni
HES-SO Master of Science in Engineering
University of Applied Sciences and Arts
Lausanne, Switzerland

***Abstract*—The code repository is one of the most important tools in software development. Its utilization has become so popular that the overwhelming majority of developers' team uses it daily. The ability to add comments in the source code is a primary source of documentation that is widely used. The point of this paper is to analyze the evolution of the comments using data acquired on public Git repositories.**

Keywords — Software engineering; git; code repository; comments; todo tags

I. INTRODUCTION

As of today, it is common that developers work in team. Their work, in particular the source code, has to be shared while allowing the whole team to work on it. A bunch of tools have been developed and are widely used to share those resources, to manage the versions and to make backups. These tools use the principle of centralized repository.

Nowadays, Git[1] is the most used of them. It allows each developer to have a local version of the source code and to manage occasional conflicts with other developers when their respective versions are merged. A tremendous amount of Git repositories allow a public reading access, which is a great source of data.

In the past decades, a lot of software development methodologies, styles of coding, average team size, approaches and design patterns have emerged. The way the source code is structured has evolved with these various principles and guidelines. Furthermore, each developer might have his own style of coding.

To fulfill requirements such as having a comprehensive code and maintainability, the presence of comments in the source code is a necessity. In this project, we analyzed the evolution of the comments in the code, in particular the amount of "todo" tags.

II. THE PROBLEM

The quality of code committed to a git repo can greatly vary. Between enterprises and individuals, there are a lot of ways to get things done. However, that does not mean all methods are equally good. The use of comments to explain parts of the code is usually viewed as a quality. It's a quick way to ease the understanding of complicated parts for when other developers read a file, or even your own code weeks later. A trend that took up is to use the keyword "TODO" to mark that some functionality is not completely finished. The goal is to let others know quickly where to work, and what is there to do in order to continue the project. Nowadays, text editors and IDEs are capable of detecting these tags and displaying them in a list to show what are the tasks to do, just like the usual "To Do list".

These tags are obviously committed with the rest of the code. Today, the open-source community is quite large and a lot of projects show their code to the public, even amongst some of the most powerful Computer Science companies. GitHub[8] has become a common place to share such projects. For example, Microsoft works on numerous large-scale open source projects hosted there. Some of their most popular are:

- Visual Studio Code[2], which is a lightweight code editor inspired by Visual Studio.
- TypeScript[3], which is a language that compiles into JavaScript.

At the time of writing, both projects have around 20'000 commits shared between 200 to 300 contributors. This is only an example to show the scale of that kind project: Most huge companies (Google[4], Twitter[5], Netflix[6], Facebook[7], etc…) have their share of open sourced projects.

All of that data is publicly available. This makes it a very interesting data source, and maybe there are valuable information to take from the coding styles and principles of that kind of project.

We want to take a look inside all these lines of code to see if there are some trends in the commenting styles of these projects. More precisely here, we're looking at the "To do" comments tags present in these files. Their primary purpose is to tell that something still has to be implemented. Thus, we can imagine that they are sparse in mature projects, and more present in newer projects, or at least in versions that are heavily in development.

From there, we'll ask ourselves some questions: What projects have the highest amount of "TODO" in their code base? In what kind of projects are they more prevalent? Are there trends based on the language used, or maybe the age of a project? Can we observe a link between the number of "TODO" and the commits close to a release version?

III. OUR SOLUTION

Our goal is to retrieve statistics about how companies and project teams use comments and “TODO” tags. Our approach is straightforward, as all necessary data is available easily. We will clone the repositories we judge relevant, and scan them to find the information we want.

What we want to do is look into different states of a project (called “commits” on git), and count the number of “TODO” in all the files at that state. We’ll start from old commits of a project, and continue further in time, checking out commits incrementally to see how the number of “TODO” evolves over the time. As some projects have a huge number of total commits containing a huge number of files, the computing time can be quite high. Thus, we plan to only check some states we find interesting.

In order to analyze the state of a repository, multiple steps are required. We need to split the job into parts; here is how we picture the main phases of the work to do:

- Clone the requested repositories on the local disk.
- Extract all the comments from different versions of the files
- Parse the extracted comments and look for the “TODO” tags
- Aggregate the received data and display it.

Each of these four tasks will be invoked sequentially by a fifth component that will manage the flow of the whole execution. We called the whole project “probeurre”, as in “prober”.

We use Docker[9] to ease the execution of tasks. Each task will be developed as a Docker image, and will then be run in a Docker container. The utilization of Docker allows us to reduce the coupling between our system’s components, and to execute each task individually. The figure 1 below represents the system’s components that will be deployed on Docker containers.

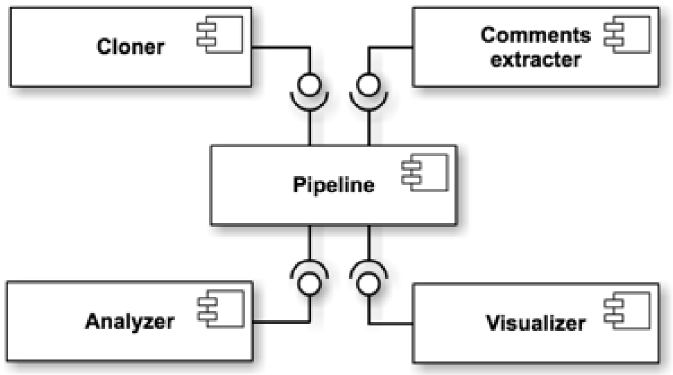


Fig. 1. System’s components deployed on independent Docker containers

Each Docker image also uses a different set of technologies to perform its own work, which allows us to adapt the number of working containers for each step of the process. In the future we could, for example, replace the current analyzer component with some other analyzers. Those new analyzers would consider different aspects of the comments than the “TODO” tags.

Because we used Docker and each container is independent, we’d simply need to add new versions of our analyzer in the pipeline to do the work. Thus our final pipeline is composed of five different Docker images, which are named after what their goal is:

- **Cloner**: takes Git Repositories URLs and clone them locally.
- Comments **Extracter**: browses the cloned Git repository, parse its files and extract all comments lines.
- **Analyzer**: takes the extracted comments, look for “TODO” tags and count them.
- **Visualizer**: aggregates the results from the analyzer and display them.
- **Pipeline**: creates the other containers and allow them to communicate between each other.

As we use different set of tools each Docker image, here is a list of them and how they are being used to contribute to the pipeline:

- Cloner uses git to clone the repositories and python 3 to automate the process.
- Comments Extracter uses an npm package ‘multilang-extract-comments’ that extracts comments from a wide variety of languages, and nodejs to run it.
- Analyzer uses nodejs to count the tags and output them in JSON format efficiently
- Visualizer uses python to generate HTML pages with the results.

The figure 2 illustrates the dataflow processed in the pipeline.

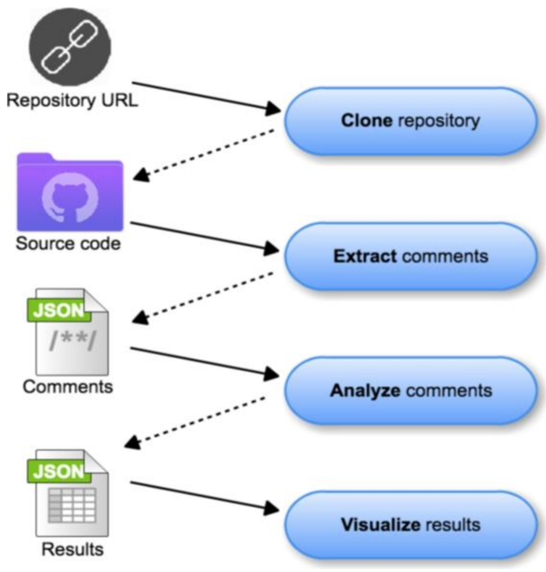


Fig. 2. Pipeline’s workflow

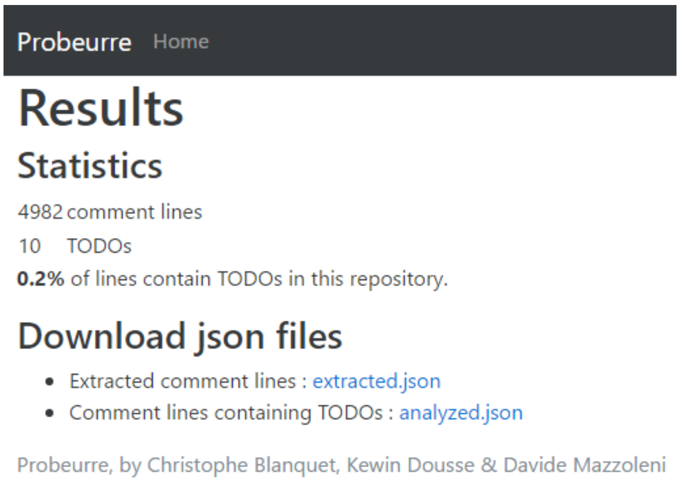
IV. Results

We ran the pipeline on some selected repositories that have enough comments to be interesting. The amount of data being super large, we decided to compare two popular repositories from two different companies:

- **React**[11]
  JavaScript library developed by Facebook used to build user interfaces. Many web developers have been seduced by its declarative view, portability and component-based architecture. Its Git repository has more than **1000** contributors, 8500 commits and 60 releases.
- **AngularJS**[12]
  JavaScript Framework developed by Google used to develop MVC front-end applications. Its main features have been very successful and provide a pleasant way to develop web-clients. Some of its most popular features are: data binding, directives to manage the DOM, dependencies injection and many others. This framework has been really successful and became one of the bests of its category in a few years only. Its Git repository has more than 1500 contributors, 8500 commits and 180 releases.

Even if the two repositories can seem quite different, they mainly use JavaScript and that's the reason we chose them. We wanted to be sure that we were able to retrieve most of the comments by selecting repositories that use well-known languages.

By running our tool, the final result comes out if the "visualizer"; it gives the results in both the standard output, and runs a server on the port 5555 to be able to see the results in a remote web browser when necessary.
The figure 5 shows a final view of the visualizer, at the end of the pipeline, for a given commit.


Probeurre Home

Results

Statistics

4982 comment lines

10 TODOs

0.2% of lines contain TODOs in this repository.

Download json files

- Extracted comment lines : extracted.json
- Comment lines containing TODOs : analyzed.json

Probeurre, by Christophe Blanquet, Kewin Dousse & Davide Mazzoleni

Fig. 3. Example of visualization result

We selected 5 different dates to "snapshot" and compare results: These correspond to releases of important React versions, and will be compared with the master commit of AngularJS at that time. These five dates are:

- 05.05.2017 : An intermediate release of React
- 08.04.2016 : React v15.0.0
- 29.03.2016 : React v0.14.8
- 20.02.2014 : React v0.9.0
- 30.05.2013 : React v0.3.0

The figures 3 and 4 show a comparison of both the total number of comment lines and the total number of TODOs between React and AngularJS.

The tables 1 and 2 show the data from which the graphs were built.

TABLE I. Sample of data for the number of comments line

Comment lines	05.2013	02.2014	03.2016	04.2016	05.2017
React	1'850	2'567	3'759	3'738	4'090
AngularJS	3'399	2'652	4'517	4'532	4'982

TABLE II. Sample of data for the number of TODO tags

TODOs	05.2013	02.2014	03.2016	04.2016	05.2017
React	40	47	117	114	231
AngularJS	7	7	11	11	10

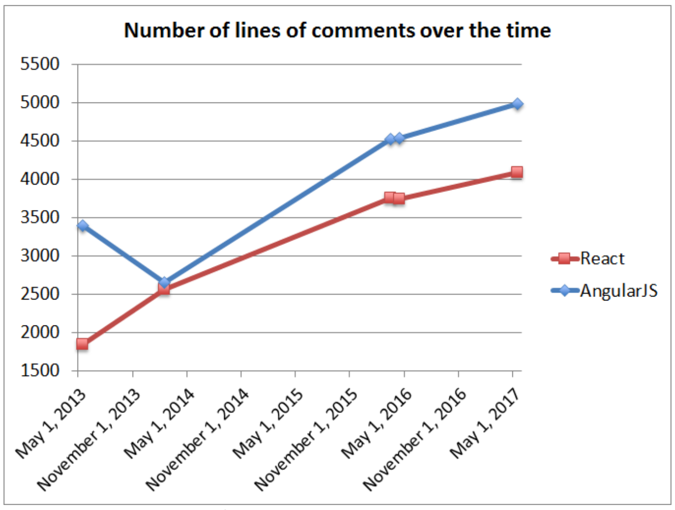


Fig. 4. Representation of the # comments over the time

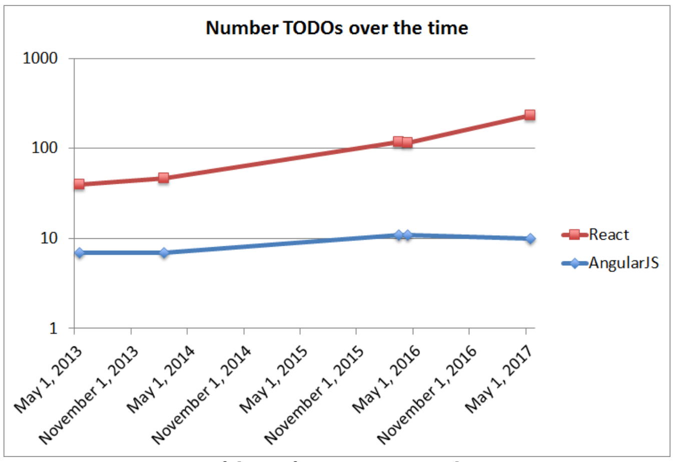


Fig. 5. Representation of the # of TODO tags over the time

Just by looking at the graphs, we can see that even by taking two repositories, both have a really different approach about how to use both comments, and TODOs.

Knowing that React commit labelled v0.3.0 is the first commit of the project that's public on their GitHub repository, we can see that it reflects on the number of comment lines, but not necessarily on the number of TODO tags; a few months later, React doubled its number of comment lines but the amount of TODOs didn't change much.

This is different from the approach taken by AngularJS, which seem to ignore TODO tags. Only a dozen of them are found in their huge codebase, which lets us think that either most of the code is cleaner that React, or that they simply don't want to use that tool. This is also quite surprising considering that we chose the dates for React's big releases, which we thought would contain less TODOs than usual.

V. Conclusion

The goal of this project was to run a pipeline of operations on some large-scale projects of companies, and retrieve correct and significant statistics from it. Each component is executed using autonomous Docker containers, to increase the reusability of each image. We ended up with 4 different images corresponding to the 4 stages we identified, plus one extra image that takes care of running the 4 others. This makes up for a total of also 5 different Git repositories.

The whole pipeline does the necessary steps to extract the data corresponding to one commit in a repository of our choice, which lets the possibility to run it on the dates we think are interesting and the repositories we want.

Although the final result could be displayed in a prettier way, we think that the main goal of the project is reached, as our tool have been used successfully on two large repositories of different companies, and gave us significant results.

One thing to note is that the data we retrieved and analyzed here are only one point of view over a git repository. It would be interesting to take these results and run other analysis like how many lines of code does projects contain in total, or even take into account whole new data sources : We could for example try to find correlations between numbers of comments and TODOS, and the number of issues reported on GitHub, or even the number of questions asked on Stack Overflow[14] related to the subject.

In conclusion, we're happy to have brought some statistics about popular git repositories, and to have been able to do this using up to date technologies. We think it's an interesting point of view over what companies code and how they do it, but we should not forget that this is only one side of the spectrum and that numerous other analysis like this could be done to improve the conclusions we found.

References

[1] Git, *Git*, https://git-scm.com
[2] GitHub Inc., *Visual Studio Code*, https://github.com/Microsoft/vscode
[3] GitHub Inc., *TypeScript*, https://github.com/Microsoft/typescript
[4] GitHub Inc., *Google*, https://github.com/Google
[5] GitHub Inc., *Twitter Inc.*, https://github.com/Twitter
[6] GitHub Inc., *Netflix Inc.*, https://github.com/Netflix
[7] GitHub Inc., *Facebook*, https://github.com/Facebook
[8] GitHub Inc., *The world's leading software development platform*, https://github.com
[9] Docker Inc., *Build, Ship and Run Any App*, Anywhere, https://www.docker.com
[10] JSON, *Introducing JSON*, http://www.json.org
[11] Facebook, *React*, https://facebook.github.io/react/
[12] Google, *AngularJS*, https://angularjs.org/
[13] GitHub Inc., *Release v0.3.0 · facebook/react*, https://github.com/facebook/react/releases/tag/v0.3.0
[14] Stack Overflow, *Stack Overflow*, https://stackoverflow.com