Name:	David Goerig (djg53)
School:	School of Computing
Programme of Study:	Advanced Computer Science (Computational Intelligence)
E-mail:	djg53@kent.ac.uk
Academic Adviser:	DR P Kenny (P.G.Kenny@kent.ac.uk)

Coursework 1: Development processes (CO894)

Chosen subject: Development processes at Google

Source:

Ferguson Henderson article (Google software engineer):

https://arxiv.org/ftp/arxiv/papers/1702/1702.01715.pdf

University of Kent | Computing

Introduction:

I choose to speak about tools and processes used at Google. Google is a well-known company, and the article I used for my research was written in 2017 by a Google software engineer. The main products of the company are of course Google Maps, Google News, Google Translate, Chrome, Android, etc.

Software Development:

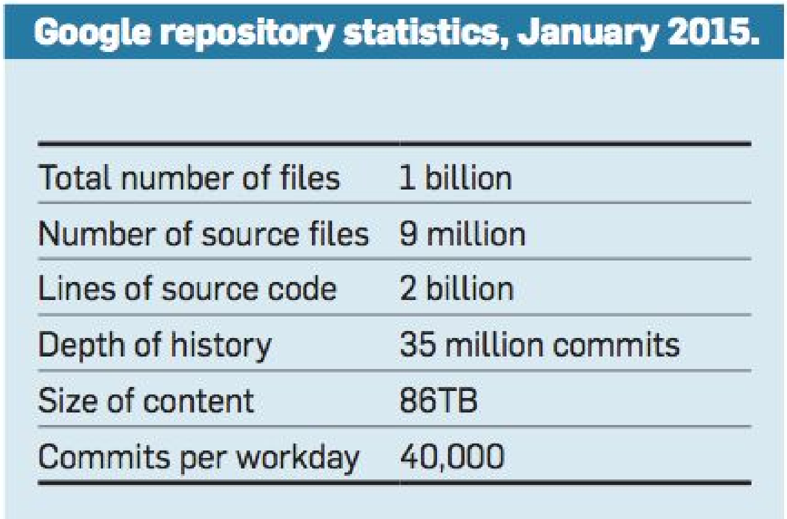
1. Data Storage

A main part of most of software engineering projects is how to store all the data. Google uses a homegrown version-control system (name is not communicated), and most of the code is stored in a single unified source-code repository (it’s called a monolithic-source-management strategy). So most of Google projects are stored in the same repository (it is approximately a 86 terabyte repository).

🔍 My point of view	On one hand, I’m not sure the monolithic-source-management is a good idea, and I did not think it could exist in big companies, and that because many reasons (the visibility of the repository, maintenance / migration problem, etc.). But on the other hand, these articles (https://people.engr.ncsu.edu/ermurph3/papers/seip18.pdf, https://cacm.acm.org/magazines/2016/7/204032-why-google-stores-billions-of-lines-of-code-in-a-single-repository/fulltext) helped me to understand google point of view (for a more collaborative model, because of the maintenance and the cost of the split).

Each directory has a set of owners who must approve the change in their area of the repository.

Google repository statistics (2015):



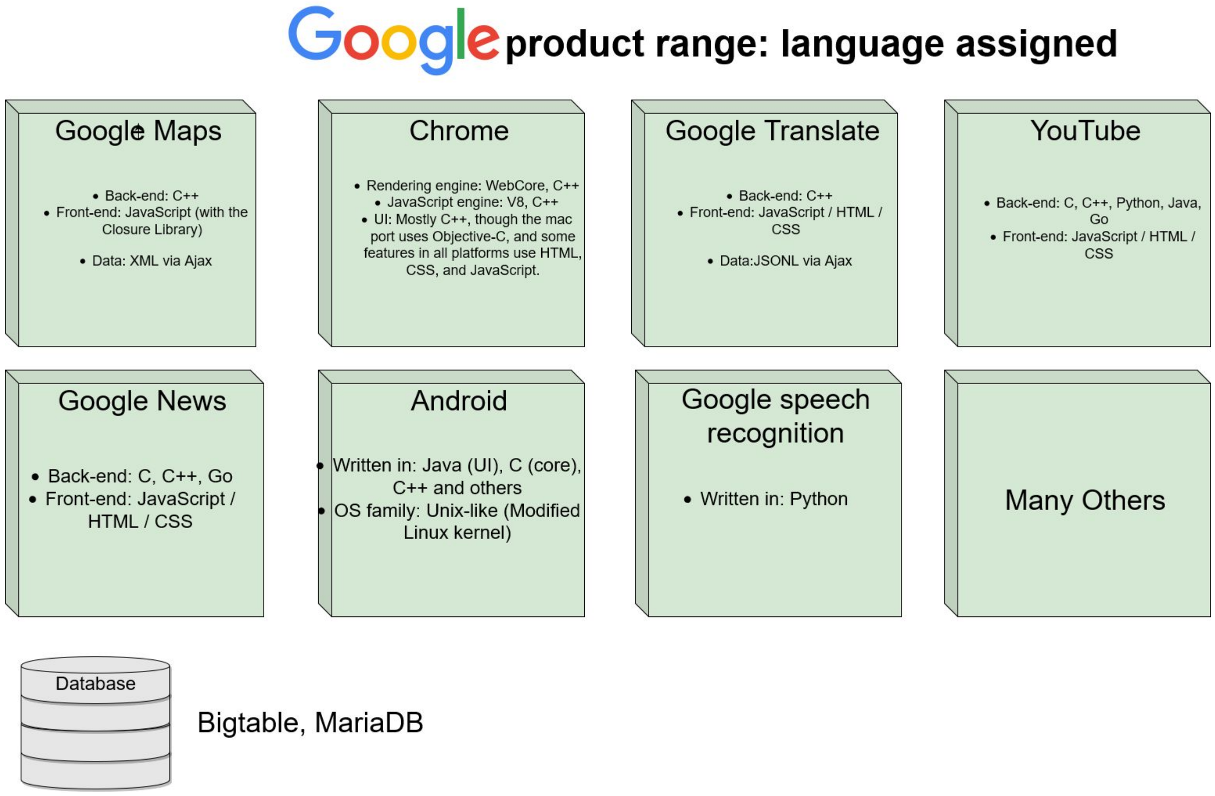
Google repository statistics, January 2015.

Total number of files	1 billion
Number of source files	9 million
Lines of source code	2 billion
Depth of history	35 million commits
Size of content	86TB
Commits per workday	40,000

(source: https://dirtysalt.github.io/html/why-google-stores-billions-of-lines-of-code-in-a-single-repository.html)

2. Used language

At google they use many programming languages: C++, Java, Python, Go or Javascript.



Google product range: language assigned

Google Maps
- Back-end: C++
- Front-end: JavaScript (with the Closure Library)
- Data: XML via Ajax

Chrome
- Rendering engine: WebCore, C++
- JavaScript engine: V8, C++
- UI: Mostly C++, though the mac port uses Objective-C, and some features in all platforms use HTML, CSS, and JavaScript.

Google Translate
- Back-end: C++
- Front-end: JavaScript / HTML / CSS
- Data:JSONL via Ajax

YouTube
- Back-end: C, C++, Python, Java, Go
- Front-end: JavaScript / HTML / CSS

Google News
- Back-end: C, C++, Go
- Front-end: JavaScript / HTML / CSS

Android
- Written in: Java (UI), C (core), C++ and others
- OS family: Unix-like (Modified Linux kernel)

Google speech recognition
- Written in: Python

Many Others

Database

Bigtable, MariaDB

They created their own guidelines for each languages.

My point of view	As in the cpp guide, we can see that they have many rules for each languages. I think that’s something that miss in a lot of enterprises, although it is really important. So every programmers follow the same rules, and the code can be maintained by everyone. See: https://google.github.io/styleguide/cppguide.html for a style example.

3. Code review

All code is reviewed before commit (by humans and automated tooling). Google has built his own web-based code review tools. This tool allows the user to make some “request review” for each part of the code (directly connected to the email). Allow the reviewers to view side-by-side diffs and comment on them.

All change to the main source code must be reviewed by one engineer if the author of the change is the owner of the file. Otherwise the code must be reviewed by two or more engineers to be approved.

Google encourages each engineer to keep each individual change small.

Code review at Google should look for different topics:

- design of the code (well-designed, appropriate to the system)
- functionality
- complexity (Could the code be made simpler?)
- tests (if the code has its unit tests)
- naming (following google style guide)
- comments (clear and useful)
- documentation (relevant or not)

Apart of that, they created a system to pick the best reviewers for each piece of code. It is usually the owner of the file, an expert in the programming language, etc. If you pair-programmed a piece of code with somebody who was qualified to do a good code review on it, then that code is considered reviewed.

Testing:

For the building and testing system Google uses a distributed build system called Blaze (NB: I think that the author made an error and wanted to say “Bazel: https://bazel.build/).
This software is responsible for compiling and linking software, but as well for building tests.

They use this system because of each specify high-level concepts like the C++ (with all the libraries), GO, Java, Python, Javascript, etc.

Apart from that they “strongly encouraged” unit testing. All the code in production needs to have his own unit tests. Each new functionality needs to add new tests to cover it. Load testing prior to deployment is a main topic of google (checking key metrics, particularly latency and error rate). They use different frameworks for the unit test depending on the languages (mainly Bazel, but GTest, Gmock are used too).
They made a unit testing library for C++ and use it for every server: Google test.

Google engineers have tools for automatically running a suite of tests before and during the run of a code review or in order to prepare a commit to the repository. The test to run is determined by a configuration file in each subtree of the repository.

Google uses Burganizer as part of their bug tracking set-up. It’s a tool for tracking issues: bugs, feature requests, customer issues and processes. The tool can categorize bugs into hierarchical components (with default assignee and email to link). Tests and automated checks are performed before and after commiting. Auto-rollback may occur in the case of widespread breakage.

In addition to that, each Google servers are linked with libraries that provide debugging tools. For example, in case of a crash, a signal will automatically create a report, create log files, and depending on the context can restart the server.

Deployment environments:

Unlike many large enterprises, most teams at google let the release engineering work done by regular software engineers.

These releases are done, for most software, weekly, and sometimes even daily, when the automating is made for the normal release engineering tasks of the project. Google chooses to make new versions / releases very frequently, and it is for three main reasons:

- of course the maintainability and the update of the application
- to keep engineer motivated
- to have more feedback on new features.

🔍 My point of view	I find this idea very well, it’s obviously more exciting than waiting for months to see the delivery of his work.

These are the steps of a classic release:

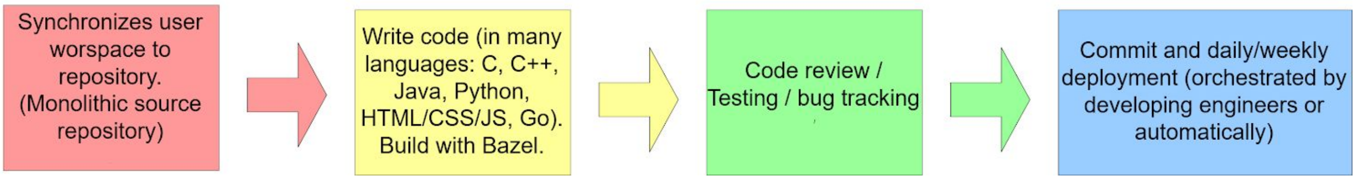
- Find a fresh workspace, or clean an old one.
- Syncing the latest ‘green build’ on a release branch. A green build is the last commit which all the automatic tests passed.
- Building the software.
- Rerun all the tests on the workspace.
- If a test fails, change the release branch with additional changes.
- If all the tests pass, the built executable(s) and data(s) file are packages up.
- Finally the release can be rolled out to all servers in all data centers. (for the traffic and the reliability of services).

A lot of other steps need to occur between the release and the final launch. These steps are numbers of improvements in term of legal, privacy, security, business and reliability requirements. They made an internal launch tool in order to track the required reviews and approvals.

Conclusion:

To conclude, we can say that Google has been able to adapt (and sometimes creating them) to the technologies of its time. And as in the case of the monolithic source repository, they have been able to better adapt their resources to match the enterprise philosophy of the software engineering and to the team management. Here is a diagram summarizing Google's workflow:

Google workflow summary


Synchronizes user worspace to repository. (Monolithic source repository)
Write code (in many languages: C, C++, Java, Python, HTML/CSS/JS, Go). Build with Bazel.
Code review / Testing / bug tracking
Commit and daily/weekly deployment (orchestrated by developing engineers or automatically)