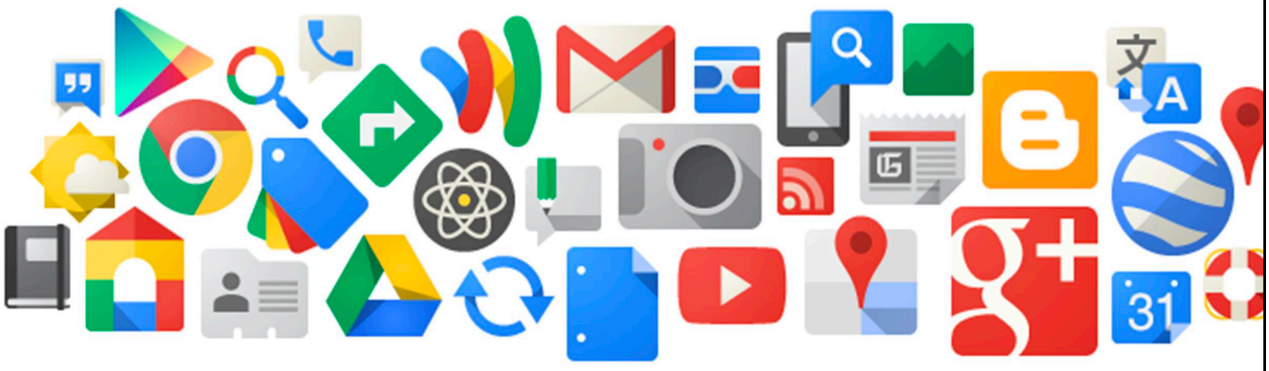
Software Engineering at Google

CS 515, Spring 2020

Laura Moreno
lmorenoc@colostate.edu

1

Google

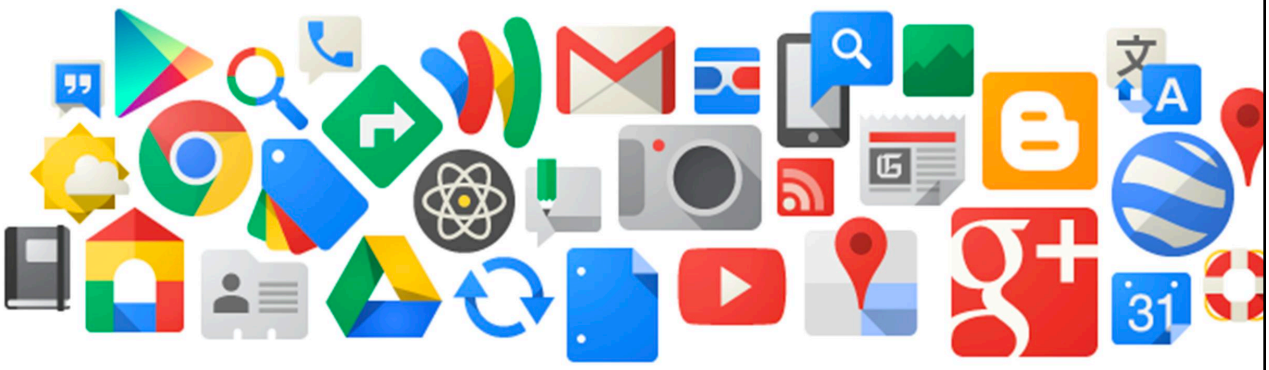


Reasons for success?

- Enlightened leadership
- Great people
- High hiring bar
- Financial strength
- Excellent software engineering practices

2

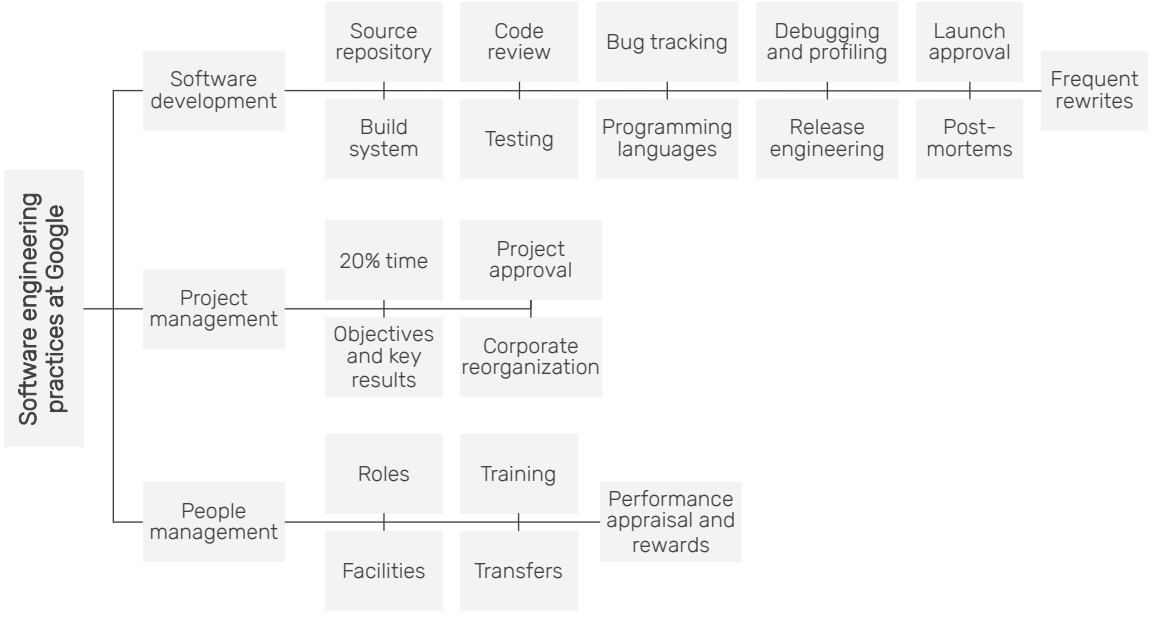
Google



Reasons for success?

- Enlightened leadership
- Great people
- High hiring bar
- Financial strength
- Excellent software engineering practices

3



4

Software development

5

The source repository

Most of Google's code is stored in a single source-code repository

- Accessible to all software engineers at Google
- Exceptions: Chrome (OSS) and Android (OSS)

As of January 2015

- **86** TB repository
- **1** billion files
- **9** million source code
- **2** billion lines of source code
- **35** million commits
- **40K** commits per workday

6

The source repository ::: Permissions

Write access to the repository is controlled

- Only the listed owners of each subtree of the repository can approve changes to that subtree

In general, any engineer can:

- Access any piece of code
- Check it out and build it
- Make local modifications and test them
- Send changes for review by the code owners
- Check in (commit) those changes, if an owner approves

Engineers are encouraged to fix anything that they see is broken

7

The source repository ::: Practices

Almost all development occurs at the "head" of the repository, not on branches

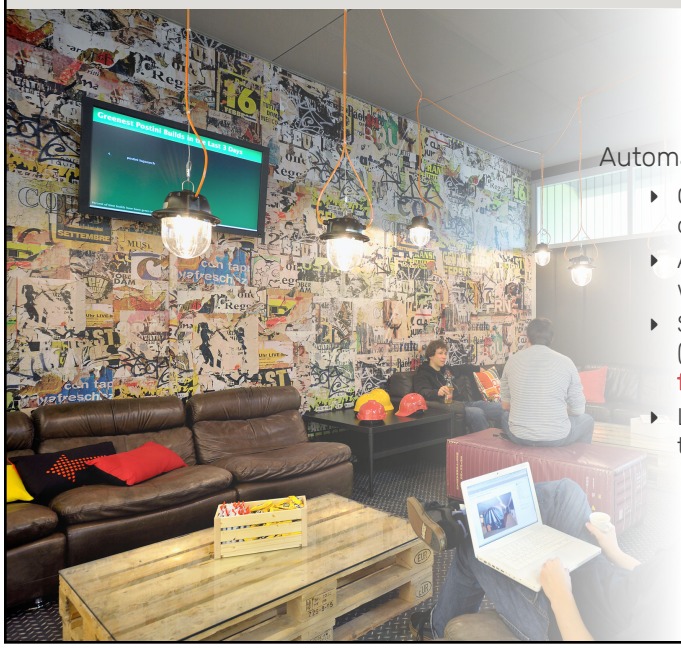
- Integration problems are identified early
- Merging work is minimized
- It is easier and faster to push out security fixes

Code ownership

- Each subtree of the repository has a file with its owners' ids (at least 2)
- Optionally, subdirectories also inherit owners from their parent directories
- The owners of each subtree control write access to that subtree
- Changes to a subtree can be made by any Googler but must be approved by an owner

8

The source repository ::: Practices



Automated systems run tests frequently

- Often after every change to any file in the transitive dependencies of the test
- Author and reviewers are notified of any change for which the tests failed
- Some teams have displays with color-coded lights (successful build and all test passing, some tests failing, broken build)
- Larger teams have a "build cop" who ensures that the tests pass at head
  - Works with authors of offending changes to fix any problems or to roll back

9

The build system

Blaze

- Compiles and links software and for running tests
- Provides standard commands that work across the whole repository
- Any Googler can build and test any software in the repository

How does Blaze work?

- Interprets "BUILD" files written by programmers
- Build files consist of high-level declarative build specifications
  - Build rules specifying for each entity, its name, its source files, and the libraries or other build dependencies
- Maps each build rule to a set of build steps
- Uses Google's distributed computing infrastructure, which allows to build extremely large programs quickly or to run thousands of tests in parallel

10

The build system ::: Practices

Build results are cached "in the cloud"

- If another build request needs the same results, the build system will reuse them

Incremental rebuilds are fast

- The build system stays resident in memory so that for rebuilds it can incrementally analyze just the files that have changed since the last build

Presubmit checks

- Use of tools to automatically run a suite of tests
- Each subtree of the repository can contain a configuration file that determines which tests to run and when (at review time, at commit time, or both)
- Tests can be synchronous (good for fast-running tests); or asynchronous, with the results emailed to the review discussion thread

Colorado State

@lymorchoc

12

Code review

Web-based code review tools integrated with email (e.g., Rietveld)

How do these tools work?

- Author requests a review
- Reviewers are notified by e-mail
- Reviewers view side-by-side diffs and comment on them
- Email notifications are sent when reviewers submit their review comments

Potential issues

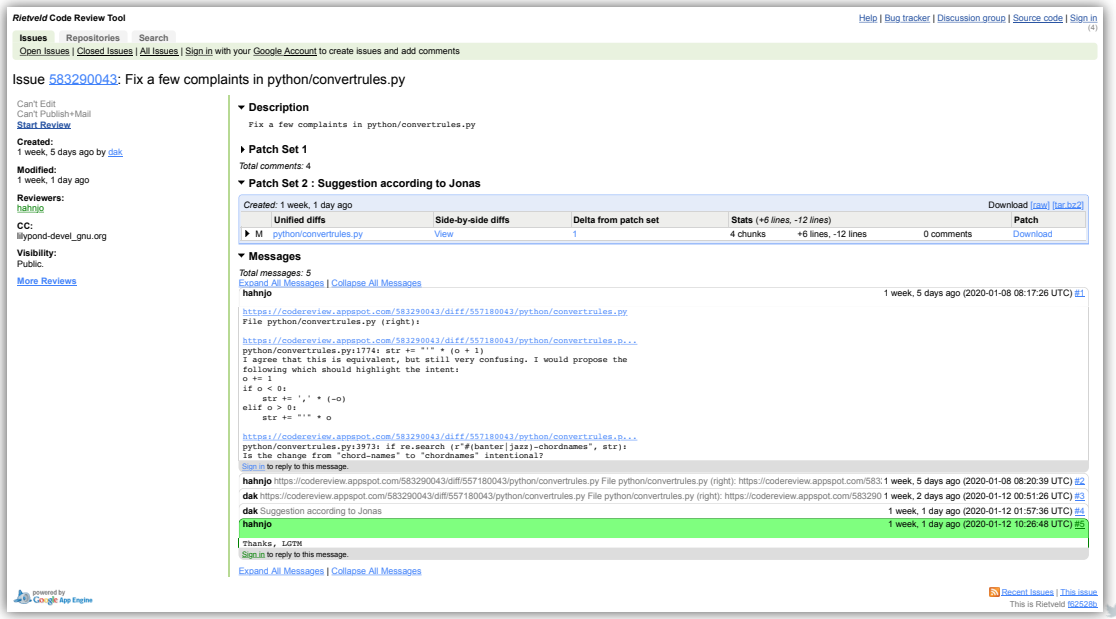
- Reviewers are too slow to respond or overly reluctant to approve changes, which could slow down development

Colorado State

@lymorchoc

13

Code review ::: Rietveld



14

Code review ::: Practices

All changes to the main source code repository must be reviewed by at least one other engineer

- If a change author is not an owner of the modified files, one of the owners must review the change
- Exception: an owner of a subtree can commit an urgent change to that subtree before it is reviewed, but a reviewer must still be named, and the change author and reviewer will get automatically nagged about it until the change has been reviewed and approved

Tools for automatically suggesting reviewer(s) for a given change

- Based on ownership and authorship of the modified code, history of recent reviewers, and number of pending code reviews for each potential reviewer
- At least one of the owners of each subtree which a change affects must review and approve that change… also, the author is free to choose reviewer(s)

15

Code review ::: Practices

Code review discussions are automatically copied to a mailing list designated by project maintainers

- Anyone is free to comment on any change
- When a bug is discovered, it's common to track down the change that introduced it and to comment on the original code review thread to point out the mistake
- It is possible to send code reviews to several reviewers and then to commit the change as soon as one of them has approved

There is an experimental section of the repository where the code review is not enforced

- Code running in production must be in the main section of the repository
- In practice, engineers often request code reviews for code in experimental

Engineers are encouraged to keep each individual change small

- Larger changes are broken into a series of smaller changes that a reviewer can easily review in one go
- This makes it easier for the author to respond to major changes suggested during the review of each piece

16

Testing

Strongly encouraged and widely practiced at Google

- All code used in production is expected to have unit tests
- The code review tool will highlight if source files are added without tests
- Code reviewers require tests for any change that adds new functionality
- Testing can be automatically enforced as part of the code review and commit process

Testing types

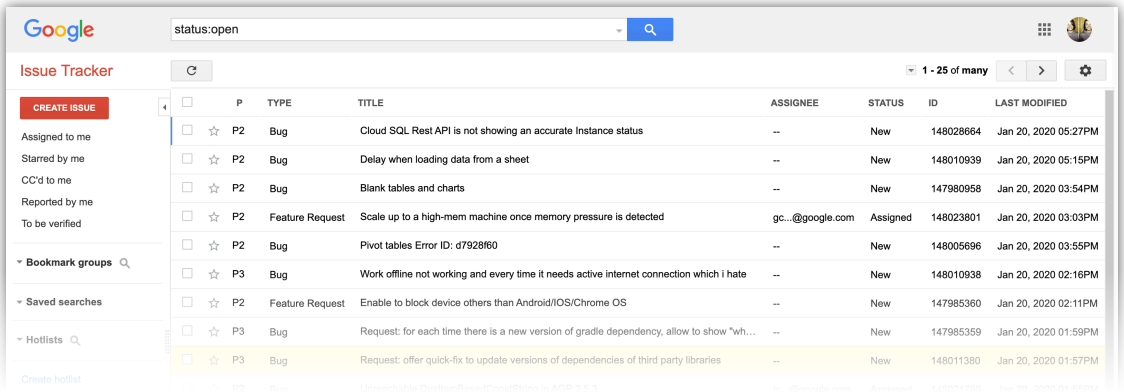
- Unit testing (mocking frameworks are popular)
- Integration testing
- Regression testing
- Load testing (prior deployment)

Automated tools for measuring test coverage

- Results are integrated as an optional layer in the source code browser

17

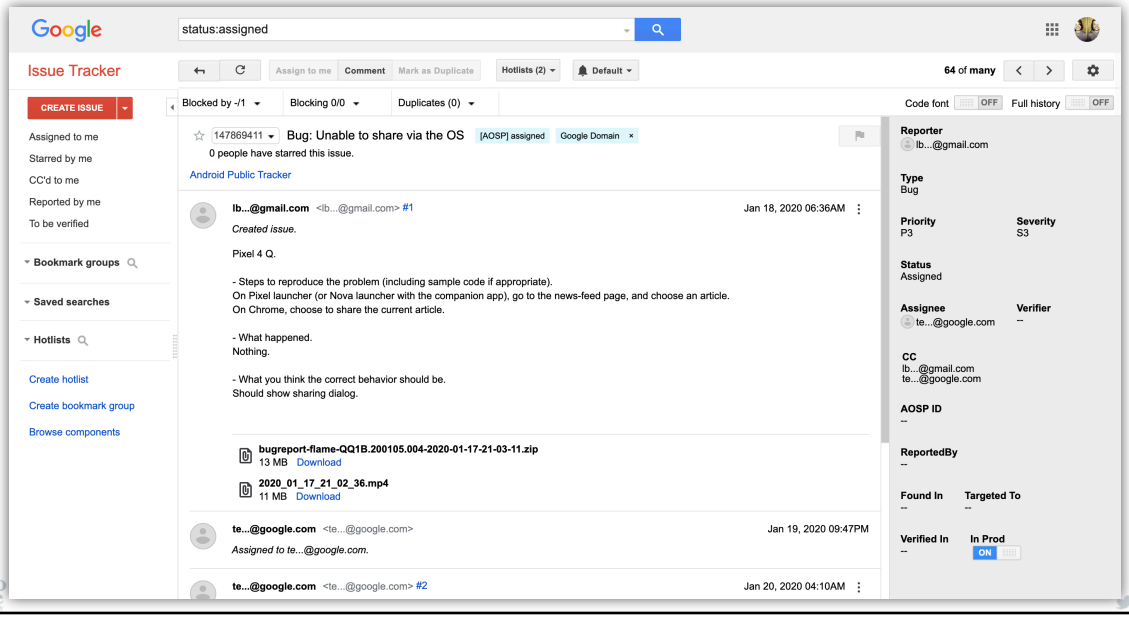
Bug tracking



Google Issue Tracker (aka Buganizer)

- Tracks issues: bugs, feature requests, customer issues, and processes (e.g., releases or clean-up efforts)
- Bugs are categorized into hierarchical components
- Each component can have a default assignee and default email list to CC
- When sending a change for review, engineers are prompted to associate the change with an issue number

Bug tracking ::: Google Issue Tracker



Bug tracking ::: Practices

Bug management differs among teams

- Some teams regularly scan through open issues in their component(s), prioritizing them and where appropriate assigning them to an engineers
- Some teams have an individual responsible for bug triage, others do bug triage in their regular team meetings
- Some teams make use of labels on bugs to indicate whether bugs have been triaged, and which release(s) each bug is targeted to be fixed in

20

Programming languages (PLs)

Officially-approved programming languages at Google: C++, Java, Python, Go, and JavaScript

- Minimizing the number of PLs reduces obstacles to code reuse and programmer collaboration

Style guides for each language

- Ensure that code across the company is written with similar style, layout, naming conventions, etc.

Company-wide readability training process

- Experienced engineers train other engineers in how to write readable, idiomatic code in each PL
- How? By reviewing a substantial change or series of changes.
- Each change that adds non-trivial new code in a given PL must be approved by someone who has passed this "readability" training process in that PL

21

Programming languages (PLs)

Many specialized domain-specific languages are used for particular purposes (e.g. the build language)

Interoperation between PLs is done mainly using Protocol Buffers

- Domain-specific language for specifying structured data
- Compiler takes in such descriptions and generates code in C++, Java, Python, for constructing, accessing, serializing, and deserializing these objects

Commonality of process is key to making development easy, even with an enormous code base and a diversity of languages

- Single set of commands to perform all the usual software engineering tasks (e.g., check out, edit, build, test, review, commit, file bug report, etc.)

22

Debugging and profiling tools

Google servers are equipped with tools for debugging running servers

- In case of a server crash, a signal handler will automatically dump a stack trace to a log file, as well as saving the core file
- If the crash was due to running out of heap memory, the server will dump stack traces of the allocation sites of a sampled subset of the live heap objects

There are also web interfaces for debugging that allow examining incoming and outgoing RPCs, changing command-line flag values, resource consumption, profiling, etc.

- These tools greatly increase the overall ease of debugging to the point where it is rare to fire up traditional debuggers

23

Release engineering

A few teams have release engineers, but often the work is done by regular software engineers

Releases are done frequently for most software

- Weekly or fortnightly releases are a common goal, and some teams even release daily
- Helps to keep engineers motivated and increases overall velocity by allowing more iterations
- It is only possible by automating most of the release engineering tasks

Typical release process

- A fresh workspace is synced to the change id of the latest "green" build and a release branch is created
- The release engineer selects additional changes to be "cherry-picked"
  - The software is rebuilt from scratch and the tests are run
  - If any tests fail, changes are made to fix the failures and are cherry-picked onto the release branch
- When the tests pass, the built executable(s) and data file(s) are packaged up
- The packaged build is loaded onto a "staging" server for integration testing by a small set of users
- The build is rolled out to one or more "canary" servers that process some of the live production traffic
- The release is rolled out to all servers in all data centers

24

Launch approval

The launch of any user-visible change or significant design change requires approvals from a few people outside of the core engineering team

Approvals are required to ensure that the code complies with legal requirements, privacy requirements, security requirements, reliability requirements, business requirements, etc.

The internal launch approval tool is used to track the required reviews and approvals and ensure compliance with the defined launch processes for each product

25

Post-mortems

Documents required when there is a significant outage of any of our production systems

Post-mortems describe systems' incidents, including title, summary, impact, timeline, root cause(s), what worked/what didn't, and action items

- Focus: problem and how to avoid it in future

Sections

- Impact: effect of the incident, in terms of duration of outage, number of lost queries and revenue
- Timeline: events leading up to the outage and the steps taken to diagnose and rectify it
- What worked/what didn't: detection practices, concrete actions, lessons learnt

26

Frequent rewrites

Most software at Google gets rewritten every few years

This consumes a large fraction of Google's resources, but also has some crucial benefits:

- It cuts away all the unnecessary accumulated complexity that was addressing requirements which are no longer so important
- It is a way of transferring knowledge and a sense of ownership to newer team members, which is crucial for productivity
- It encourages mobility of engineers between different projects, which helps to encourage cross-pollination of idea
- It helps to ensure that code is written using modern technologies and methodologies

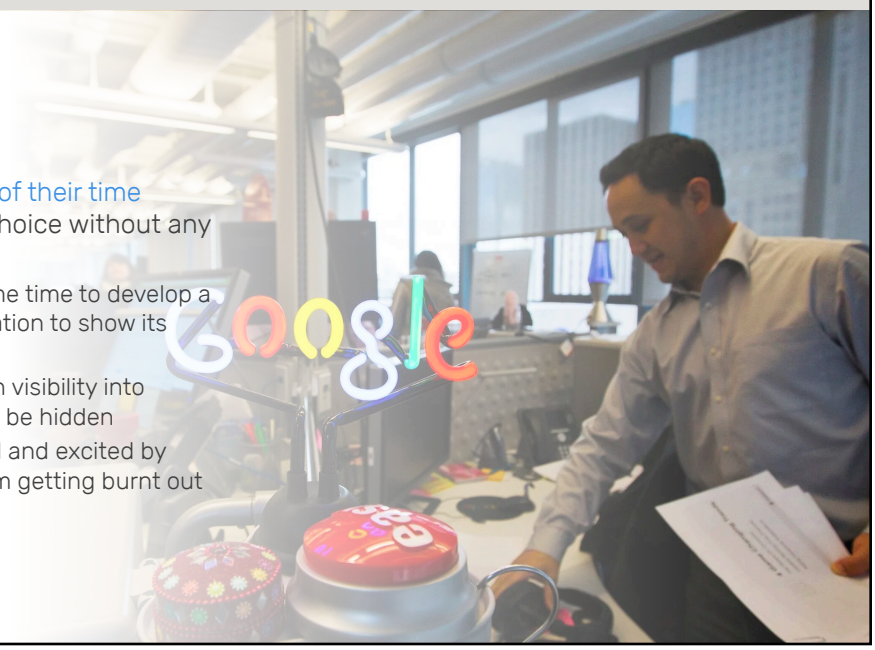
27

Project management

20% time

Engineers can spend up to 20% of their time working on any project of their choice without any approval

- Anyone with an idea has some time to develop a prototype, demo, or presentation to show its value
- It provides management with visibility into activity that might otherwise be hidden
- It keeps engineers motivated and excited by what they do, and stops them getting burnt out



Objectives and key results (OKRs)

Individuals and teams are required to document their goals and to assess their progress towards them

Teams set quarterly and annual objectives with measurable key results

- At every level of the company, going all the way up
- Goals for individuals and small teams should align with the higher-level goals
- At the end of each quarter, progress towards the key results is recorded and each objective is given a score from 0.0 to 1.0 (0 to 100% completion)
- Results are visible across Google (with some exceptions) but not used directly for performance appraisal

OKRs should be set high

- The target average score is 65%, i.e., a team is encouraged to set as goals about 50% more tasks than they are likely to accomplish
- If a team scores significantly higher than that, they are encouraged to set more ambitious OKRs for the following quarter

OKRs provide a key mechanism for communicating what each part of the company is working on

30

Project approval

There is no well-defined process for project approval or cancellation

Managers are responsible and accountable for the projects their teams work on

- In some cases, decisions are made in a bottom-up fashion, with engineers being given freedom to choose which projects to work on, within their team's scope
- In other cases, decisions are made in a top-down fashion, with executives or managers making decisions about which projects will go ahead, which will get additional resources, and which will get cancelled

31

Corporate reorganizations

Occasionally an executive decision is made to cancel a large project

- Engineers who had been working on that project may have to find new projects or new teams

There have been some "defragmentation" efforts (i.e., projects that are split across multiple geographic locations are consolidated into a smaller number of locations)

- Engineers are generally given freedom to choose their new team and role from within the positions available in their geographic location
- In the case of defragmentation, they may also be given the option of staying on the same team and project by moving to a different location

Merging/splitting teams and changes in reporting chains seem to be frequent

32

People management

33

Roles

Small number of different roles within engineering

- Separate the engineering and management career progression ladders
- Separate the tech lead role from management
- Embed research within engineering
- Support engineers with product managers, project managers, and site reliability engineers (SREs)
- Within each role, there is a career progression possible, with a sequence of levels, and the possibility of promotion to recognize performance at the next level

34

Roles ::: Main roles

Engineering manager

- Always manage people (3~30, 8~12 is most common)
- Often former SWEs
- Have considerable technical expertise and people skills
- Perform coaching with career development
- Do performance evaluation part of the hiring process

Software engineer (SWE)

- Most people doing software development work have this role
- Hiring bar for software engineers is very high

Research scientist

- Hiring criteria are very strict, and the bar is extremely high
- Requires exceptional research ability (great publication record and ability to write code)
- Many people in academia who would be able to qualify for a SWE role would not qualify for a Research Scientist role
- Evaluated on their research contributions, including their publications

Site reliability engineer

- Do maintenance of operational systems

Product manager

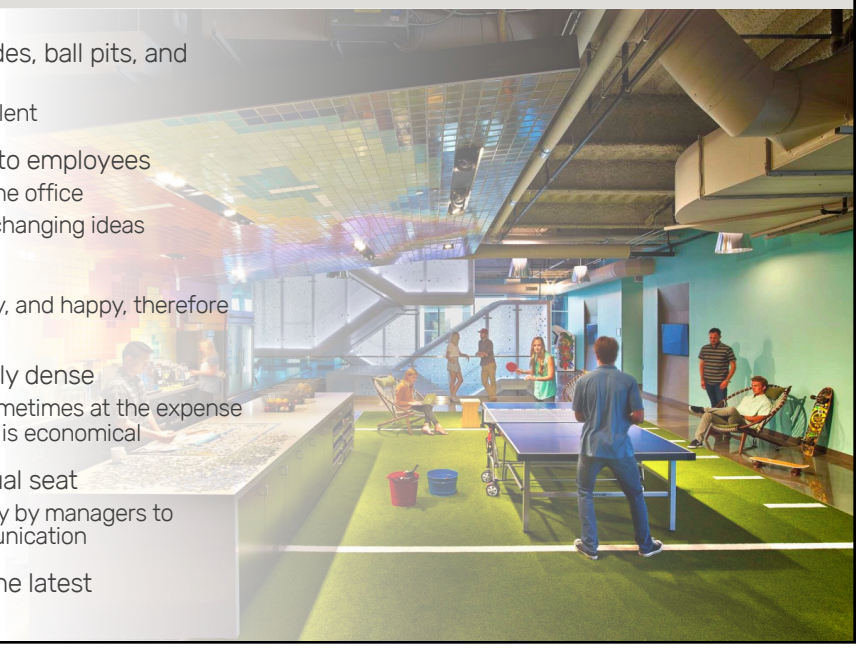
- Responsible for the management of a product
- Coordinate the work of SWEs, ensuring that everything needed is in place to produce a high-quality product
- Do not write code themselves but work with SWEs to ensure that the right code gets written.

Program manager / Technical program manager

- Program managers manage projects, processes, or operations
- Technical program managers require specific technical expertise relating to their work, e.g., linguistics for dealing with speech data

35

Facilities ::: There's no "free" lunch



Famous for its fun facilities with slides, ball pits, and games rooms
- Helps attract and retain good talent

Cafes and "microkitchens" are free to employees
- Encourage Googlers to stay in the office
- Act as an informal space for exchanging ideas

Gyms, sports, and on-site massage
- Help keep employees fit, healthy, and happy, therefore productive

Seating is open-plan, and often fairly dense
- Encourages communication, sometimes at the expense of individual concentration, and is economical

Employees are assigned an individual seat
- Seats are re-assigned frequently by managers to facilitate and encourage communication

Meeting rooms are equipped with the latest technologies

Colorado State

40

Training

Google encourages employee education in many ways:
- Mandatory initial training course for new Googlers ("Nooglers")
- "Codelabs" for technical staff (SWEs and research scientists)
- Online and in-person training courses
- Support for studying at external institutions

Each Noogler is usually appointed an official "mentor" and a separate "buddy" to help get them up to speed

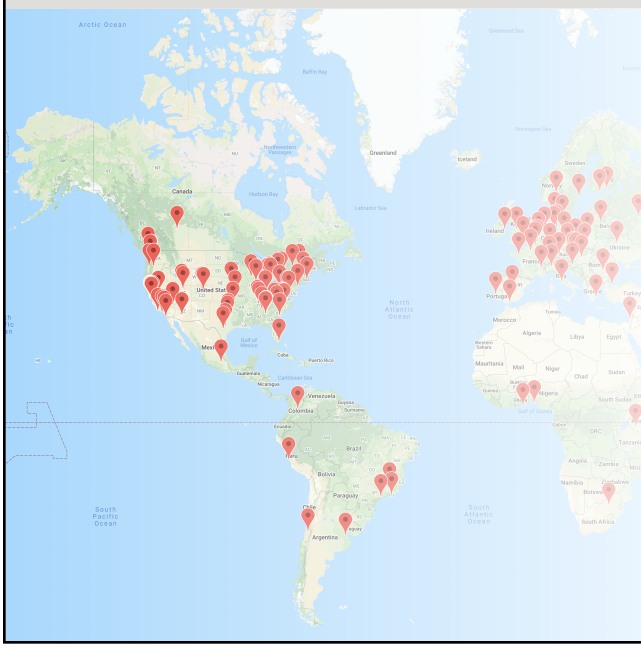
Unofficial mentoring via regular meetings with manager, team meetings, code reviews, design reviews, and informal processes.

Colorado State

@lymorchot

41

Transfers



Transfers between different parts of the company are encouraged

- Help spread knowledge and technology across the organization
- Improve cross-organization communication

Transfers between projects and/or offices are allowed for employees in good standing after 12 months in a position

Software engineers are encouraged to do temporary assignments in other parts of the organization

42

Performance appraisal and rewards

Feedback is strongly encouraged

- Any employee can nominate any other for a "peer bonus" ($100) up to twice per year
- Employees can also give "kudos", i.e., formalized statements of praise for good work
- Managers can award bonuses
- Employees get annual performance bonuses and equity awards

Promotion process is very careful and detailed

- Nomination by self or manager, self-review, peer reviews, manager appraisals
- Decisions are made by promotion committees, and the results can be subject to further review

Poor performance is handled with manager feedback

- If needed, a performance improvement plan is set
- If that fails, termination is possible, but in practice this is extremely rare

Manager performance is assessed with feedback surveys

- Employee fill in a survey about the performance of their manager twice a year
- The results are anonymized and aggregated and then made available to managers

43

References

F. Henderson, "Software Engineering at Google." (2017)

M. Greiler, "Code Reviews at Google are lightweight and fast." (2019)

J. Whittaker, J. Arbon, J. Carollo, "How Google Tests Software." Addison-Wesley. (2012)

Various Googlers, "Google Engineering Tools" (2011)