# Why Google Stores Billions of Lines of Code in a Single Repository

Colin MacMaster, Alex Claycomb, Alex Namkung, Inti Espejo, Michael Koch, Nandha Chokkan

## Google Develops Code at a Massive Scale

Their repository includes:

- 86 TB of data
- 2 Billion lines of code
- 9 million unique source files
- shared access by 25000 developers
- 16,000 manual changes a day
- 24,000 changes by automated systems
- Serves 500,000 read queries per second

Commits Per Week (2010-2015)

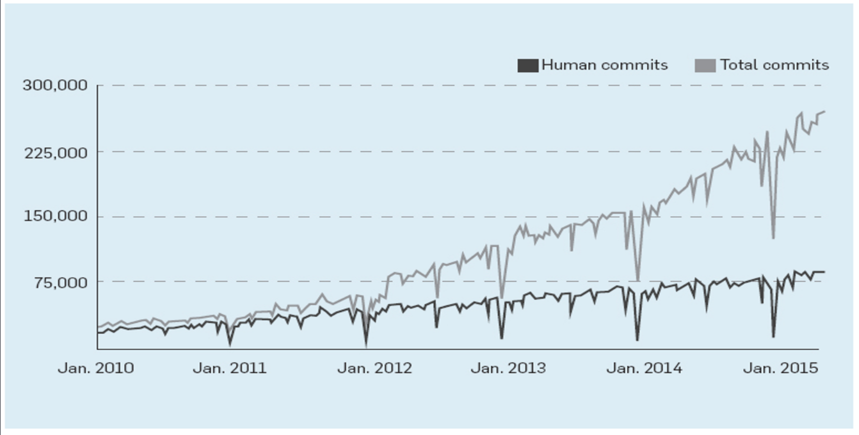
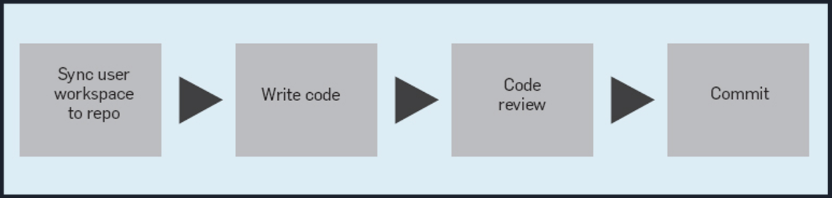


Figure 3. Commits per week.

## Piper

- Google's main method for storing their source code in one large repository
- Distributed across 10 Google data centers
  - Before Piper, Google used one Perforce instance
- Logs read and write access, and supports file-level access control for security
- Works very similar to Git:
  - Developer creates a local copy of files (like a local clone of a repository in git)
  - Updates to Piper can be pulled and merged into local work (like a fetch and merge in git)
  - Files can be committed and pushed to the repository after code review



Simplified Piper Workflow

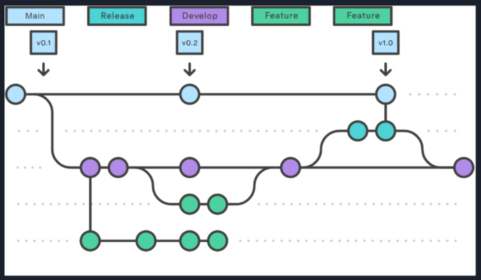
## Clients in the Cloud (CitC)

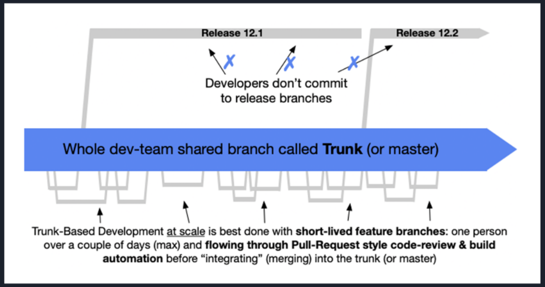
- Main way of developing for applications stored on Piper
- All files stored in a cloud-based linux-only environment
- All Piper files can be browsed, but only changes are stored in CitC
- Workspaces are available across machines
- Features that ease development:
  - Auto-commit for code review
  - Simple edits to Piper files directly from code browser

Ultimately, Piper and CitC allow Google and other large software companies to keep their source code in monolithic repositories.

# Trunk Based Development

- Piper and Citc were heavily influenced by Trunk Based Development
- Changed to the most recent version of code or the “trunk” are made in a single, serial ordering
- Avoids the painful process of merging together large branches of code
- Because of this development process, branches are rarely used at Google and are only used for releases and snapshots
- Flag flips are used instead to easily switch off new implementations that may have problems
- Trunk based development also allows Google to easily test and experiment with performance while the code is being written



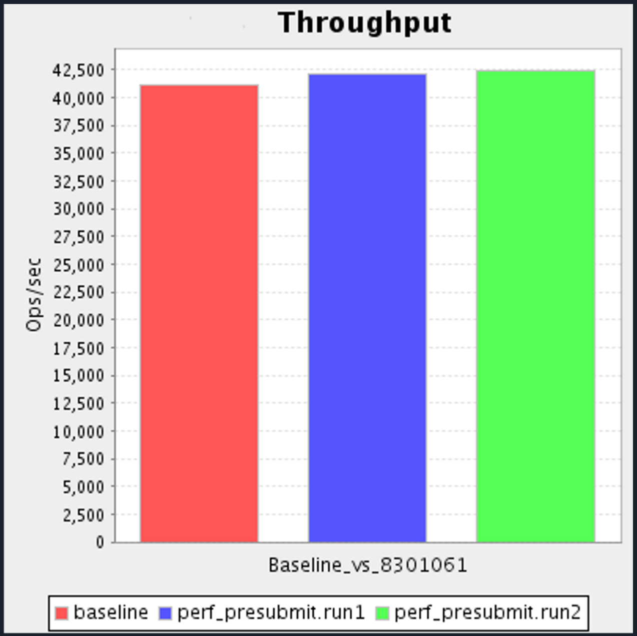

Release 12.1
Release 12.2
Developers don't commit to release branches
Whole dev-team shared branch called **Trunk** (or master)
Trunk-Based Development at scale is best done with **short-lived feature branches**: one person over a couple of days (max) and **flowing through Pull-Request style code-review & build automation** before “integrating” (merging) into the trunk (or master)

## Google Workflow

- Due to the nature of trunk-based development on such a massive scale, several practices and support systems are required.

Technical

- Automated testing infrastructure initiates a rebuild of all affected dependencies on almost every change
  - Changes that result in breakage are automatically undone
- Google's "Presubmit" infrastructure can identify breakage without committing to the repository
  - Presubmit analyses are run for all changes
  - Code owners can create their own custom analyses to target certain directories



Example presubmit report at Google

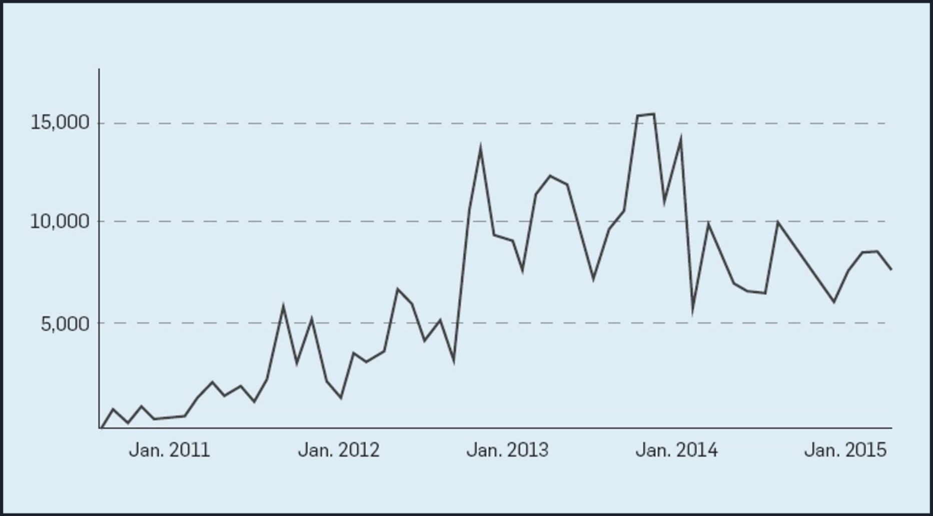
## Google Workflow

<u>Cultural</u>

- Code quality is maintained by reviewing all code that is committed to the repository
  - CitC and Google formatting rules make it easy for managers to quickly review employee commits
  - “Tricorder” system also provides automated analytics on coding quality

<u>Cleanup</u>

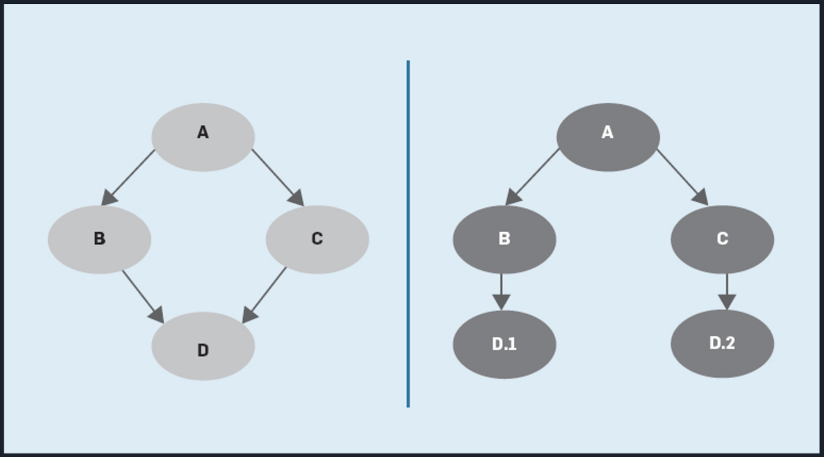
- Teams of Google members frequently undertake wide reaching code cleanup changes to maintain the health of the codebase
  - “Rosie” system splits a large patch into smaller pieces which can be reviewed, tested, and submitted independently



Rosie commits per month

## Advantages

- Unified versioning
    - “Single source of truth”
- Large amount of code available for sharing and use
    - Easy to include code across directories
- Atomic changes
    - Major changes impacting 100s, 1000s of files can be done in a single operation
- Fluid team boundaries
    - All code stays in same repo, regardless of project ownership
- Easy to understand how individual source files fit in

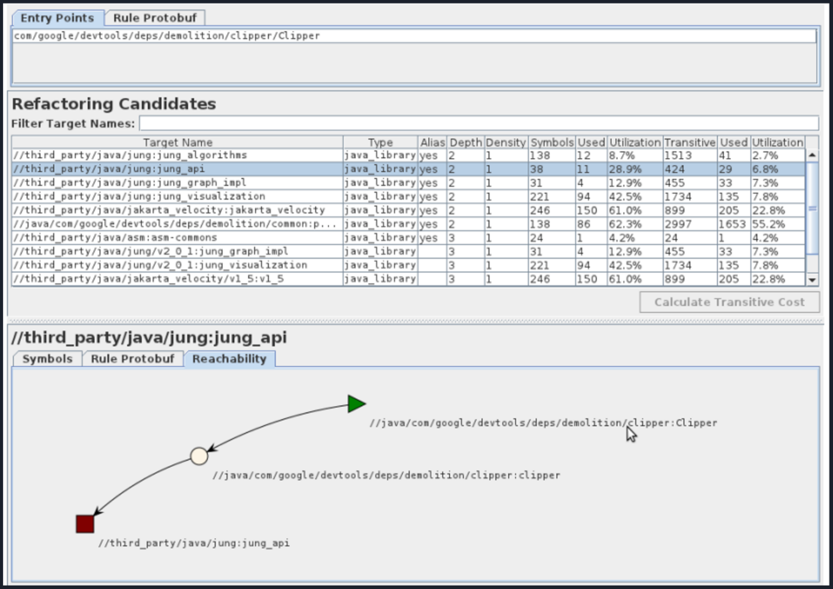


The “diamond dependency” problem - avoided by Google’s monolithic codebase

Source: https://cacm.acm.org/magazines/2016/7/204032-why-google-stores-billions-of-lines-of-code-in-a-single-repository/fulltext

## Costs and Trade-Offs

- Tooling Investment
    - Requires tools for development and execution
- Increased Codebase Complexity
    - Difficulties with code discovery and unnecessary dependencies
        - Binary bloating, build breakages
- Code Health Investment
    - Support refactorings, codebase-wide clean-ups, and modernization efforts



com/google/devtools/deps/demolition/clipper/Clipper

**Refactoring Candidates**

Filter Target Names:

| Target Name | Type | Alias | Depth | Density | Symbols | Used | Utilization | Transitive | Used | Utilization |
|---|---|---|---|---|---|---|---|---|---|---|
| //third_party/java/jung:jung_algorithms | java_library | yes | 2 | 1 | 138 | 12 | 8.7% | 1513 | 41 | 2.7% |
| //third_party/java/jung:jung_api | java_library | yes | 2 | 1 | 38 | 11 | 28.9% | 424 | 29 | 6.8% |
| //third_party/java/jung:jung_graph_impl | java_library | yes | 2 | 1 | 31 | 4 | 12.9% | 455 | 33 | 7.3% |
| //third_party/java/jung:jung_visualization | java_library | yes | 2 | 1 | 221 | 94 | 42.5% | 1734 | 135 | 7.8% |
| //third_party/java/jakarta_velocity:jakarta_velocity | java_library | yes | 2 | 1 | 246 | 150 | 61.0% | 899 | 205 | 22.8% |
| //java/com/google/devtools/deps/demolition/common:p... | java_library | yes | 2 | 1 | 138 | 86 | 62.3% | 2997 | 1653 | 55.2% |
| //third_party/java/asm:asm-commons | java_library | yes | 3 | 1 | 24 | 1 | 4.2% | 24 | 1 | 4.2% |
| //third_party/java/jung/v2_0_1:jung_graph_impl | java_library |  | 3 | 1 | 31 | 4 | 12.9% | 455 | 33 | 7.3% |
| //third_party/java/jung/v2_0_1:jung_visualization | java_library |  | 3 | 1 | 221 | 94 | 42.5% | 1734 | 135 | 7.8% |
| //third_party/java/jakarta_velocity/v1_5:v1_5 | java_library |  | 3 | 1 | 246 | 150 | 61.0% | 899 | 205 | 22.8% |

**//third_party/java/jung:jung_api**

Symbols
Rule Protobuf
Reachability
//java/com/google/devtools/deps/demolition/clipper:Clipper
//java/com/google/devtools/deps/demolition/clipper:clipper
//third_party/java/jung:jung_api

Clipper: Dependency Refactoring Tool

## Alternatives

Google has been considering a move from their proprietary software to Git as their main version control system

Some teams (Android, Chrome) are already using it to supplement their use of the main repository

The main drawback for google is that Git functions much better with many small repositories rather than one large one, so they would have to split their repository

Google is experimenting with Mercurial a system that is much more similar to Git than their proprietary system

## Question

Do you think this strategy is beneficial for the giant that Google has become, or is it more useful for engineers to be able to manage their projects on a smaller scale?