Modern Code Review: A Case Study at Google

Sadowski, Söderberg, Church, Sipko, Bacchelli — ICSE-SEIP 2018

Small, fast, one reviewer, and it’s not about bugs.

Colin Wong **Abhinav Gonthina** **Aryan Gupta**

44

engineers surveyed. All 44 said code review is valuable. Two said it found a bug.

So why do it?

Review was introduced to make code teachable

- An early employee brought it in so developers would write code others could understand — code as a teacher for whoever reads it next, and a second person familiar with every piece of it.
- **Today's four reasons:** education, maintaining norms, gatekeeping, accident prevention. Finding bugs is welcome, not the point.
- **The education effect shows up in the logs:** first-year developers get more than twice as many comments per change, and it falls with tenure.

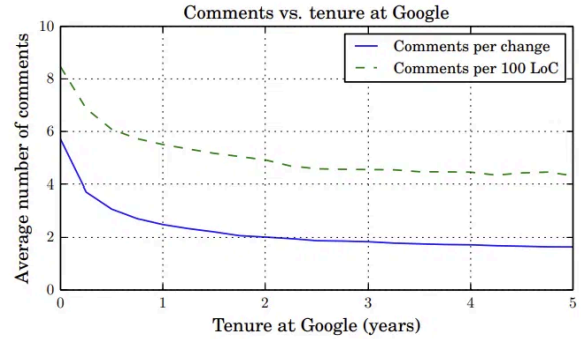


Figure 2: Reviewer comments vs. author's tenure at Google

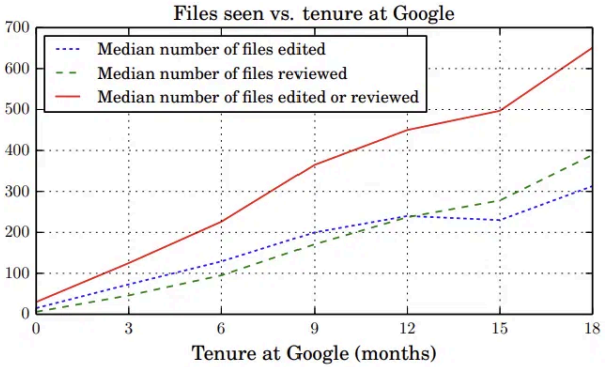


Figure 3: The number of distinct files seen (edited or reviewed, or both) by a full-time employee over time.

What it looks like, in numbers

9 million changes, 25,000+ developers, January 2014 to July 2016.

	Google	Everyone else
Median change size	24 lines	44 (AMD) to 263 (Lucent)
Median reviewers	1 fewer than 25% have more	2 the convergent practice
Median time to finish review	under 4 hours	14.7 – 24 hours
Time to first feedback	<1 hour small, ~5 hours very large	—
Rounds of back-and-forth	80%+ need at most one	—
Reviewing time per developer	3.2 h/week	6.4 h/week self-reported, OSS

Three things make one reviewer enough

- **Ownership.** Every directory has owners, and an owner has to approve any change to it before it lands.
- **Readability certification.** Every change must be written or reviewed by someone certified in that language, so style and norms are somebody's formal job.
- **Static analysis inside the tool.** Tricorder runs 110 analyzers and posts results as comments, so humans aren't spending the review on formatting.
- **Small changes as a cultural norm.** Comments per change climb with size, peaking around 12.5 at roughly 1,250 lines.

Pawtograder, last 30 merged PRs

Read-only gh queries on pawtograder/platform, as of Oct 19, 2026. Bots excluded from every “human” count.

	Google	Pawtograder
Median change size	24 lines	468 lines about 20× bigger
90th percentile size	—	2,048 lines
Median time to first human review	under 1 hour	0.8 hours comparable
90th percentile, first human review	—	52.8 hours
Approvals	1 reviewer must approve	28 of 30 merged with zero

Bigger doesn’t buy more review

PR size	PRs	Median review comments	Got any human review
under 50 lines	6	0	**0 of 6**
50 – 500	9	2	3 of 9
500 – 2,000	12	10.5	11 of 12
over 2,000	3	9	2 of 3

Comments rise with size, then flatten past about 2,000 lines — the same shape as Google’s peak around 1,250.

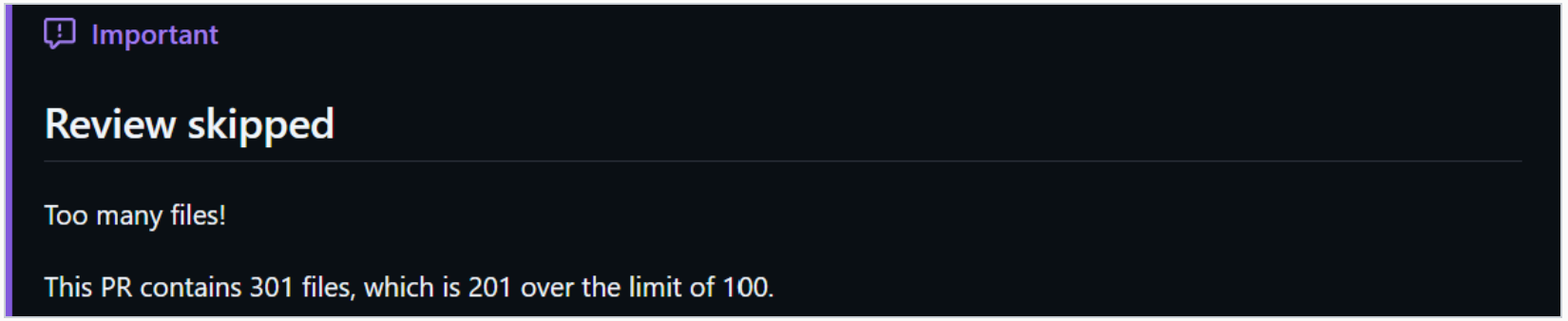
Past a certain size, review gives up

#814
exam grading

Lines	9,495
Files	55
Inline comments	159
First human review	after ~10 days
Open	135 days

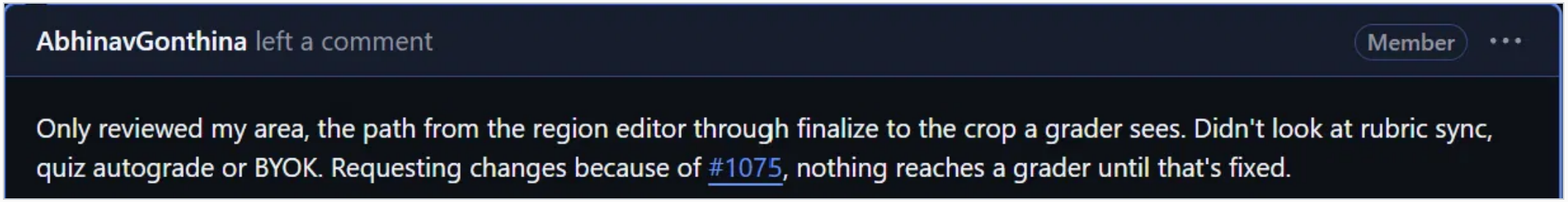
#1050
bug reporter

Lines	40,279
Files	307
Inline comments	8
First human review	same day
Open	17 days



What I saw reviewing #814

- On 9,495 lines across 55 files, I could only review my slice — the answer-region path. Nobody reviewed the whole thing.
- Without the foreign-key fix, finalize failed for every batch and the end-to-end test timed out.
- With the fix, the end-to-end test **passed** — while every answer's text was empty. A green test on a broken result.
- Big PRs hide bugs. That is exactly the paper's case for small changes, on our code.



Four reasons to hold the result loosely

- **One company.** The authors say so themselves: results may not generalize. Google has tooling and a culture almost nobody else has.
- **The survey is small and self-selected.** 44 respondents, volunteers, inside the company — social desirability cuts in one direction.
- **“Valuable” was never measured.** It’s self-reported satisfaction. Defect rates were not compared with and without review.
- **Correlation, not causation.** Small changes and fast reviews travel together; nothing here shows which one produces the other.

Interviewees were recruited by snowball sampling, starting from people the authors already knew.

Three answers to “how many reviewers?”

	Reviewers	What holds it up
Google	1	Ownership, readability certification, static analysis in the tool
Rigby & Bird	2	The convergent practice across OSS, AMD, and Microsoft
Pawtograder, on paper	2	One teammate, one from another team — per the First Implementation Ticket
Pawtograder, in practice	~1 human	The maintainer, plus AI review bots

We have none of Google’s three supports — and we have agent-written code and 468-line PRs.

Google gets away with one reviewer because of ownership, readability, and static analysis. We have none of those, plus agent-written code and big PRs.

Should we push for Google-style small PRs with one reviewer, or is our two-reviewer setup the right call?

One reviewer, small PRs

Two reviewers