Code Review at Google

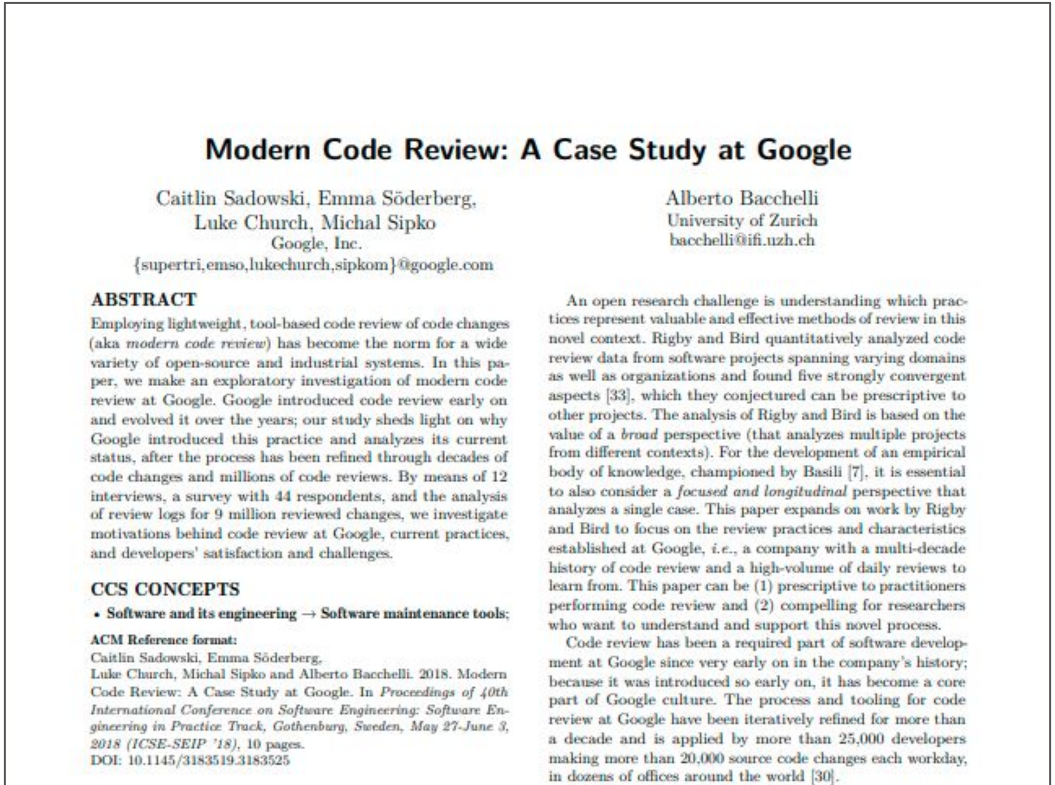
Emma Söderberg
Dept. of Computer Science

LUND UNIVERSITY

LTH
FACULTY OF ENGINEERING

Software@LTH Oct 8, 2020

ICSE 40TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING MAY 27 - JUNE 3 2018 GOTHENBURG, SWEDEN



Modern Code Review: A Case Study at Google

Caitlin Sadowski, Emma Söderberg,
Luke Church, Michal Sipko
Google, Inc.
{supertri,emso,lukechurch,sipkom}@google.com

Alberto Bacchelli
University of Zurich
bacchelli@ifi.uzh.ch

ABSTRACT

Employing lightweight, tool-based code review of code changes (aka *modern code review*) has become the norm for a wide variety of open-source and industrial systems. In this paper, we make an exploratory investigation of modern code review at Google. Google introduced code review early on and evolved it over the years; our study sheds light on why Google introduced this practice and analyzes its current status, after the process has been refined through decades of code changes and millions of code reviews. By means of 12 interviews, a survey with 44 respondents, and the analysis of review logs for 9 million reviewed changes, we investigate motivations behind code review at Google, current practices, and developers' satisfaction and challenges.

CCS CONCEPTS

- **Software and its engineering → Software maintenance tools**;

ACM Reference format:
Caitlin Sadowski, Emma Söderberg,
Luke Church, Michal Sipko and Alberto Bacchelli. 2018. Modern Code Review: A Case Study at Google. In *Proceedings of 40th International Conference on Software Engineering: Software Engineering in Practice Track, Gothenburg, Sweden, May 27-June 3, 2018 (ICSE-SEIP '18)*, 10 pages.
DOI: 10.1145/3183519.3183525

An open research challenge is understanding which practices represent valuable and effective methods of review in this novel context. Rigby and Bird quantitatively analyzed code review data from software projects spanning varying domains as well as organizations and found five strongly convergent aspects [33], which they conjectured can be prescriptive to other projects. The analysis of Rigby and Bird is based on the value of a *broad* perspective (that analyzes multiple projects from different contexts). For the development of an empirical body of knowledge, championed by Basili [7], it is essential to also consider a *focused and longitudinal* perspective that analyzes a single case. This paper expands on work by Rigby and Bird to focus on the review practices and characteristics established at Google, *i.e.*, a company with a multi-decade history of code review and a high-volume of daily reviews to learn from. This paper can be (1) prescriptive to practitioners performing code review and (2) compelling for researchers who want to understand and support this novel process.

Code review has been a required part of software development at Google since very early on in the company's history; because it was introduced so early on, it has become a core part of Google culture. The process and tooling for code review at Google have been iteratively refined for more than a decade and is applied by more than 25,000 developers making more than 20,000 source code changes each workday, in dozens of offices around the world [30].

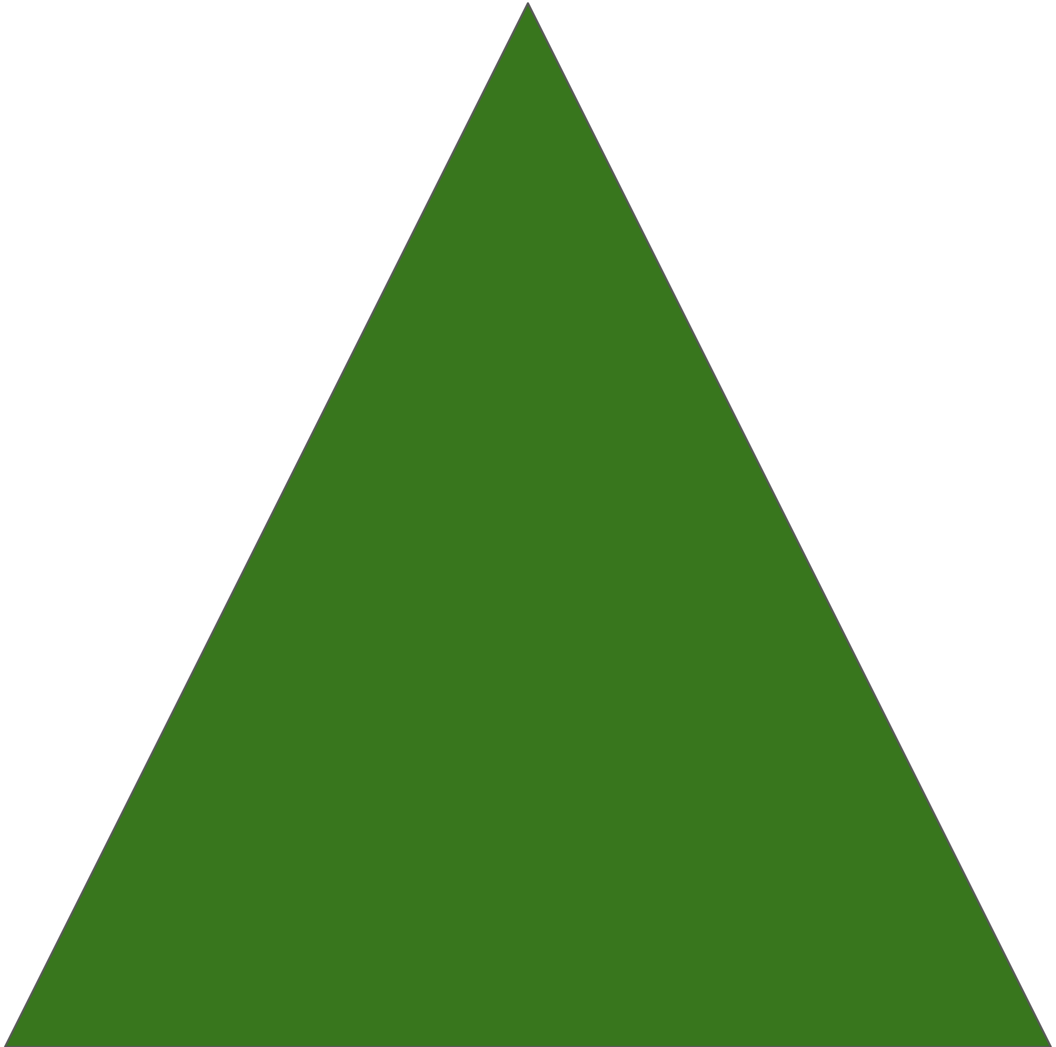
The Google Context

Very big mono repo. (1B+ files, 35M+ commits)

Custom developer infrastructure: Piper, CitC, Blaze, Critique, Tricorder, …

Global development, 25K+ engineers

Why Google Stores Billions of Lines of Code in a Single Repository? CACM, Vol 59(6):78-87, 2016.

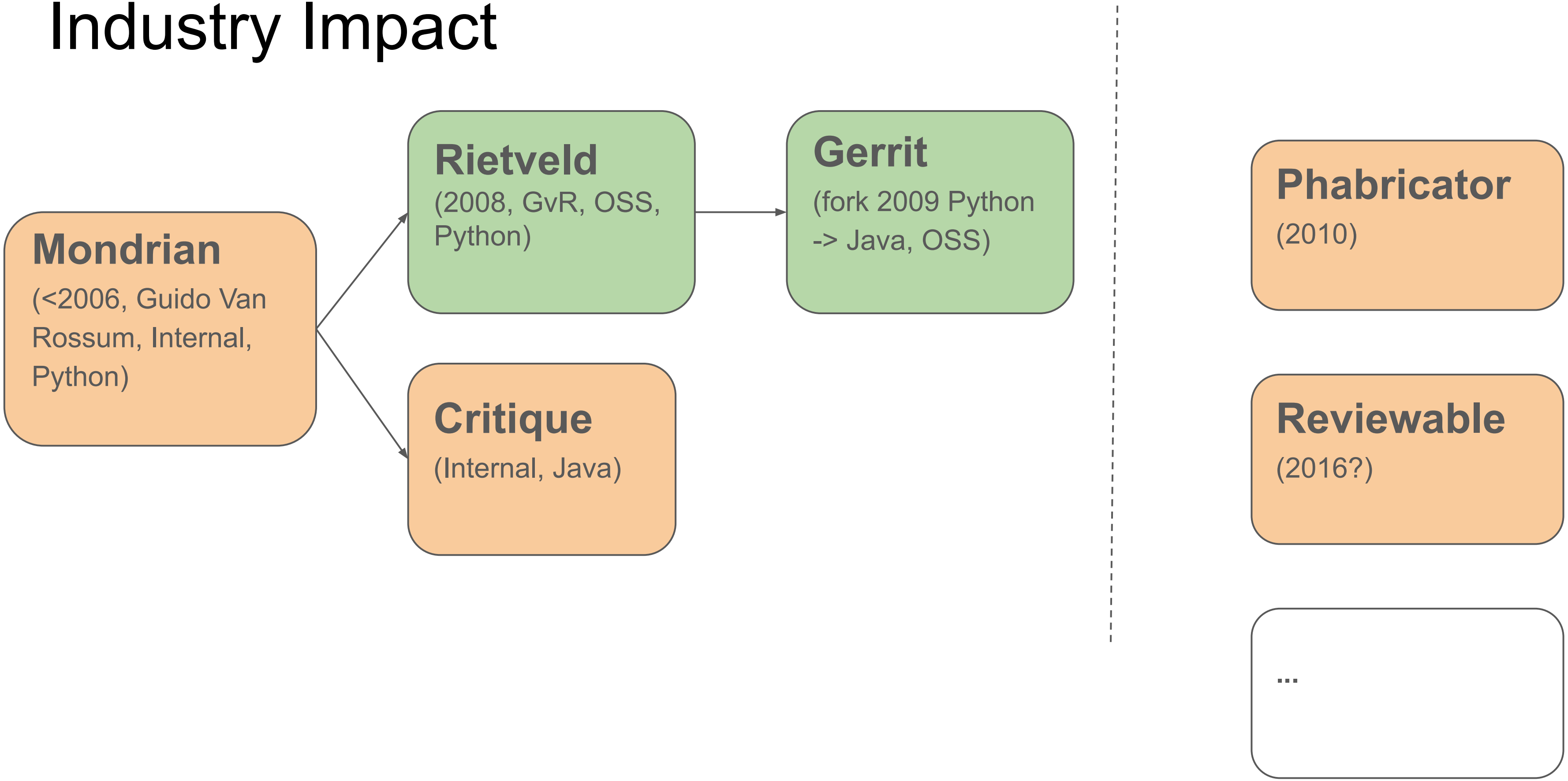


History of Code Review @ Google

When it started and why?

- Research code base (prototyping) -> production code base
- Future engineers need to be able to read
- Goal: Knowledge sharing
- Discovered benefits:
  - Enforce style & design
  - Ensure proper tests
  - Security / Oversight

Industry Impact



Mondrian
(<2006, Guido Van Rossum, Internal, Python)

Rietveld
(2008, GvR, OSS, Python)

Gerrit
(fork 2009 Python -> Java, OSS)

Critique
(Internal, Java)

Phabricator
(2010)

Reviewable
(2016?)

...

The Code Review Process

The steps of the process: creating, previewing, commenting, addressing feedback, approving (LGTM)

The rules of the process: at least one reviewer, readability, ownership

Quantifying the process:

- Avg. 3 authored changes / week per developer
- Median 4 changes reviewed / week
- Median latency < 4 hours
- Median LOC changed is 24
- < 25% changes have more than one reviewer

The Developers View

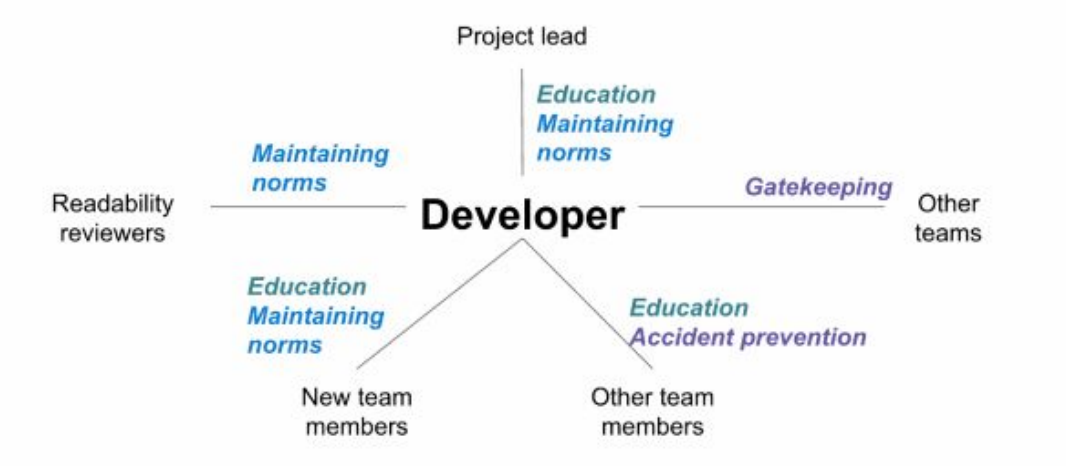


Figure 1: Relationship diagram describing which themes of review expectations appeared primarily within a particular author/reviewer context.

Effects

Knowledge spreading

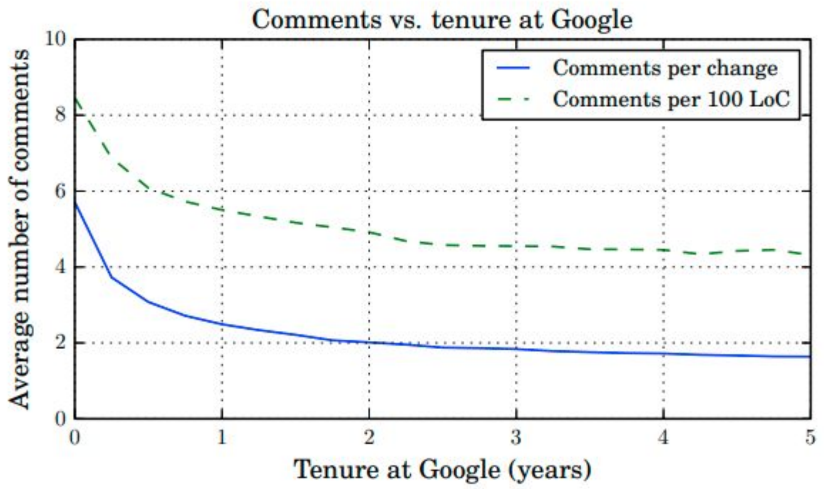


Figure 2: Reviewer comments vs. author's tenure at Google