Machine Learning for Software Engineering

Alberto Bacchelli

Machine Learning for SE: Day plan

When	What
9:00 - 10:00	Introductions and overview
10:30 - 11:15	PyDriller on Software Ownership and Quality
11:15 - 12:00	**GHTorrent on Pull Request Acceptance**
13:00 - 15:00	Mining Unstructured Software Data
15:30 - 17:00	Machine Learning and Mobile Apps

Machine Learning for Software Engineering

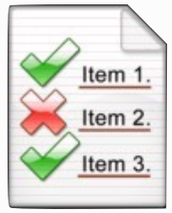
GHTorrent on Pull Request Acceptance

Machine Learning for SE — Workflow

software engineering tasks helped

software development

software maintenance



software testing

...

empirical evidence

what is a good bug report?

what is the impact of code ownership on quality?

is pair programming useful?

machine learning and software analysis techniques

classification

program analysis

clustering

parsing

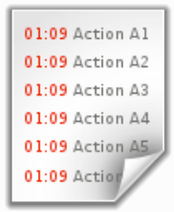
...

The software development process generates much data

Versioning system

Development environment

Issue tracking system



Program execution

Debugger application

Emailing system

...

Machine Learning for SE — Strong academic impact

An Exploratory Study of the Pull-based Software Development Model

Georgios Gousios
Delft University of Technology
Delft, The Netherlands
G.Gousios@tudelft.nl

Martin Pinzger
University of Klagenfurt
Klagenfurt, Austria
martin.pinzger@aau.at

Arie van Deursen
Delft University of Technology
Delft, The Netherlands
Arie.vandeursen@tudelft.nl

ABSTRACT

The advent of distributed version control systems has led to the development of a new paradigm for distributed software development; instead of pushing changes to a central repository, developers pull them from other repositories and merge them locally. Various code hosting sites, notably Github, have tapped on the opportunity to facilitate pull-based development by offering workflow support tools, such as code reviewing systems and integrated issue trackers. In this work, we explore how pull-based software development works, first on the GHTorrent corpus and then on a carefully selected sample of 291 projects. We find that the pull request model offers fast turnaround, increased opportunities for community engagement and decreased time to incorporate contributions. We show that a relatively small number of factors affect both the decision to merge a pull request and the time to process it. We also examine the reasons for pull request rejection and find that technical ones are only a small minority.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement—*Version control*; D.2.9 [**Software Engineering**]: Management—*Programming teams*

General Terms

Management

Keywords

pull-based development, pull request, distributed software development, empirical software engineering

1. INTRODUCTION

Pull-based development is an emerging paradigm for distributed software development. As more developers appreciate isolated development and branching [7], more projects, both closed source and, especially, open source, are being migrated to code hosting sites such as Github and Bitbucket with support for pull-based development [2]. A unique characteristic of such sites is that they allow any user to clone any public repository. The clone creates a public project that belongs to the user that cloned it, so the user can modify the repository without being part of the development team. Furthermore, such sites automate the selective contribution of commits from the clone to the source through pull requests.

Pull requests as a distributed development model in general, and as implemented by Github in particular, form a new method for collaborating on distributed software development. The novelty lays in the decoupling of the development effort from the decision to incorporate the results of the development in the code base. By separating the concerns of building artifacts and integrating changes, work is cleanly distributed between a contributor team that submits, often occasional, changes to be considered for merging and a core team that oversees the merge process, providing feedback, conducting tests, requesting changes, and finally accepting the contributions.

Previous work has identified the processes of collaboration in distributed development through patch submission and acceptance [23, 5, 32]. There are many similarities to the way pull requests work; for example, similar work team structures emerge, since typically pull requests go through an assessment process. What pull requests offer in addition is process automation and centralization of information. With pull requests, the code does not have to leave the revision control system, and therefore it can be versioned across repositories, while authorship information is effortlessly maintained. Communication about the change is context-specific, being rooted on a single pull request. Moreover, the review mechanism that Github incorporates has the additional effect of improving awareness [9]; core developers can access in an efficient way all information that relates to a pull request and solicit opinions of the community ("crowd-source") about the merging decision.

A distributed development workflow is effective if pull requests are eventually accepted, and it is efficient if the time this takes is as short as possible. Advancing our insight in the effectiveness and efficiency of pull request handling is of direct interest to contributors and developers alike. The goal of this work is to obtain a deep understanding of pull request usage and to analyze the factors that affect the efficiency of the pull-based software development model. Specifically, the questions we are trying to answer are:

RQ1 How popular is the pull based development model?

RQ2 What are the lifecycle characteristics of pull requests?

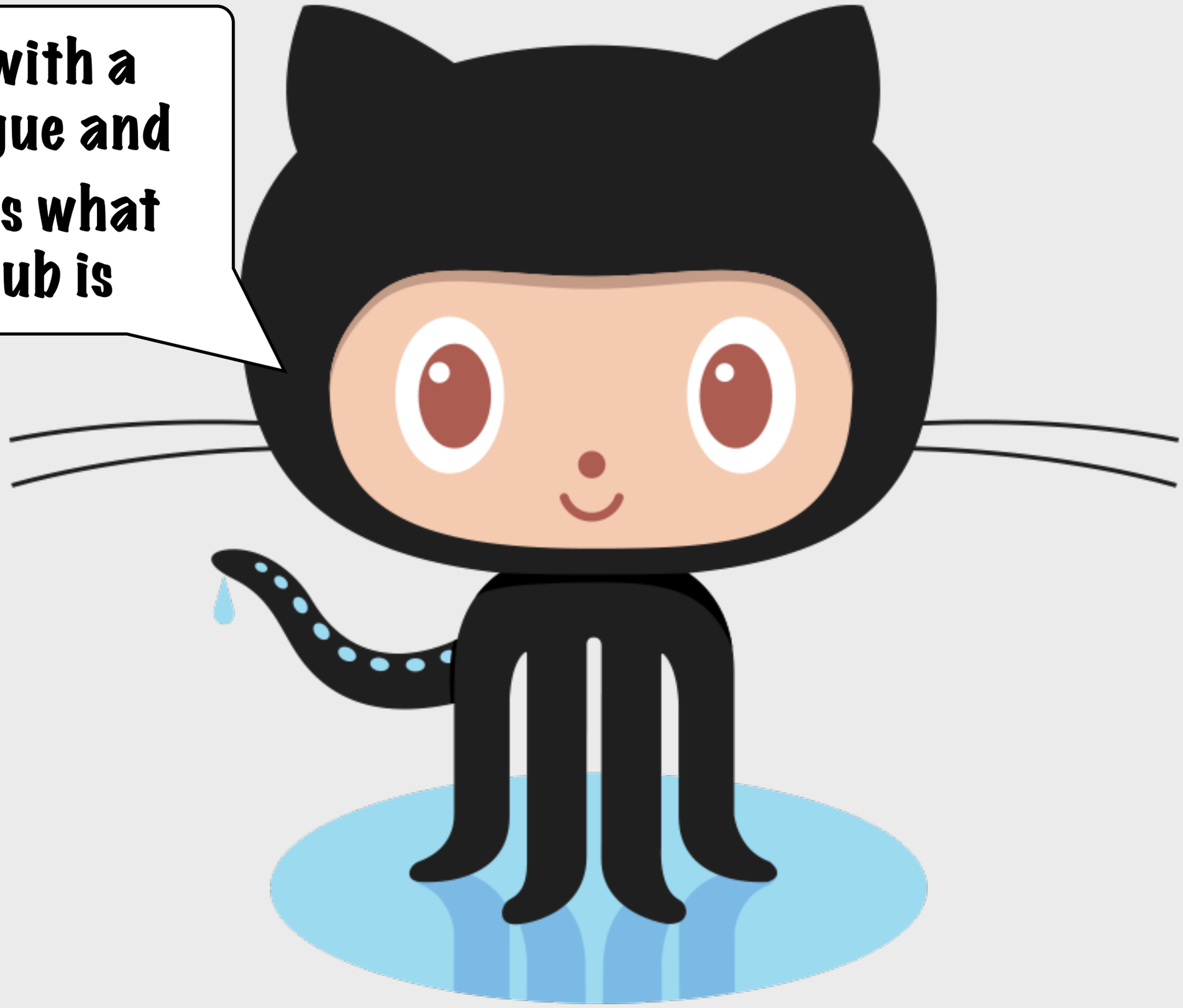
RQ3 What factors affect the decision and the time required to merge a pull request?

RQ4 Why are some pull requests not merged?

Our study is based on data from the Github collaborative development forge, as made available through our GHTorrent project [16].

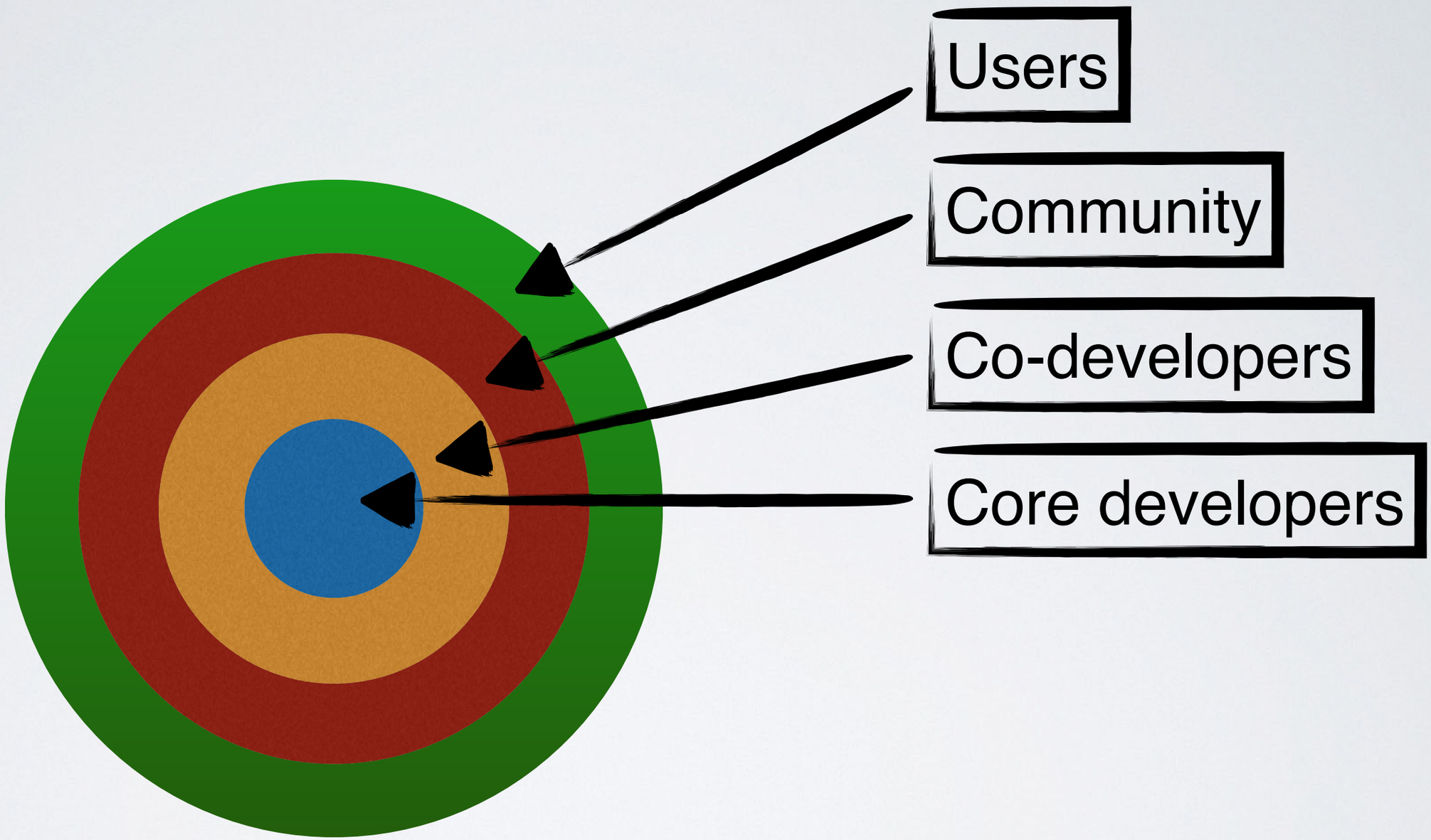
What is GitHub?

Pair with a colleague and discuss what GitHub is

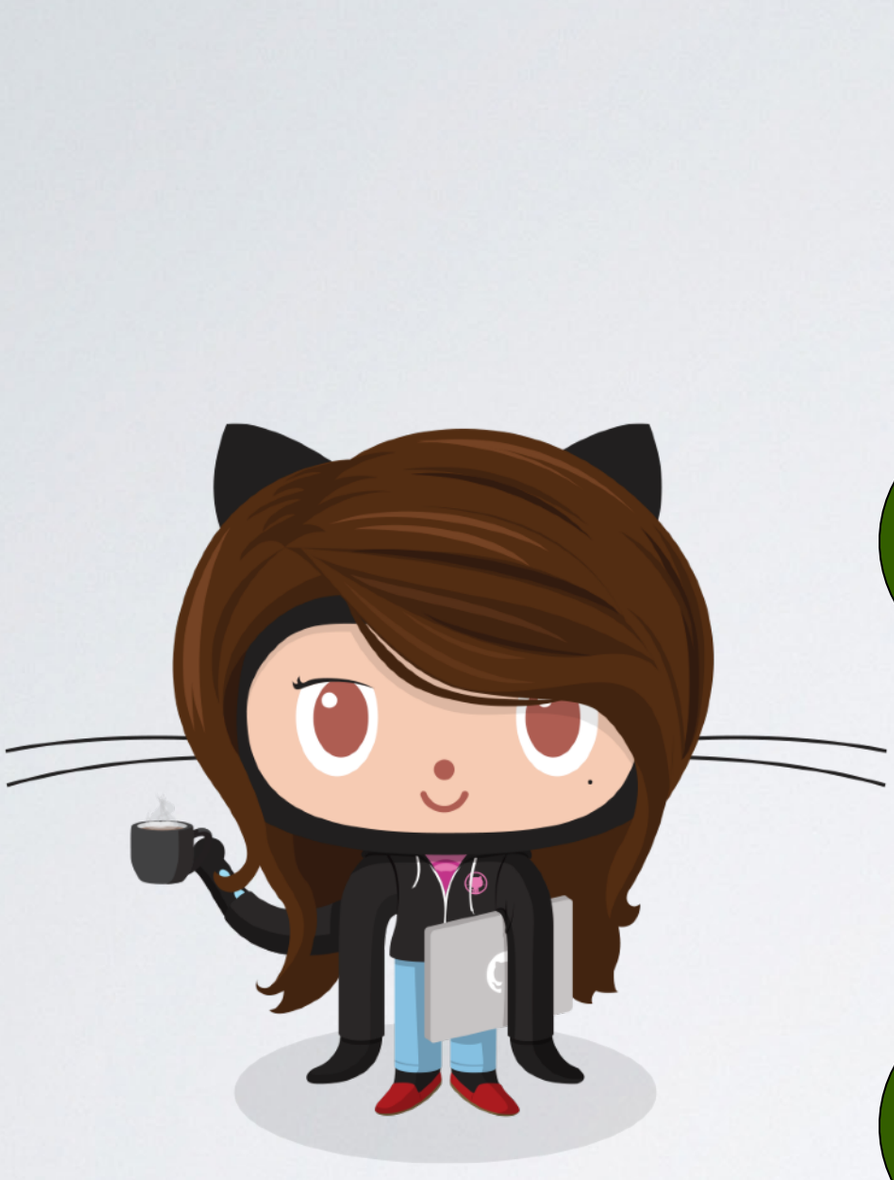


What is GitHub? The pull based development model

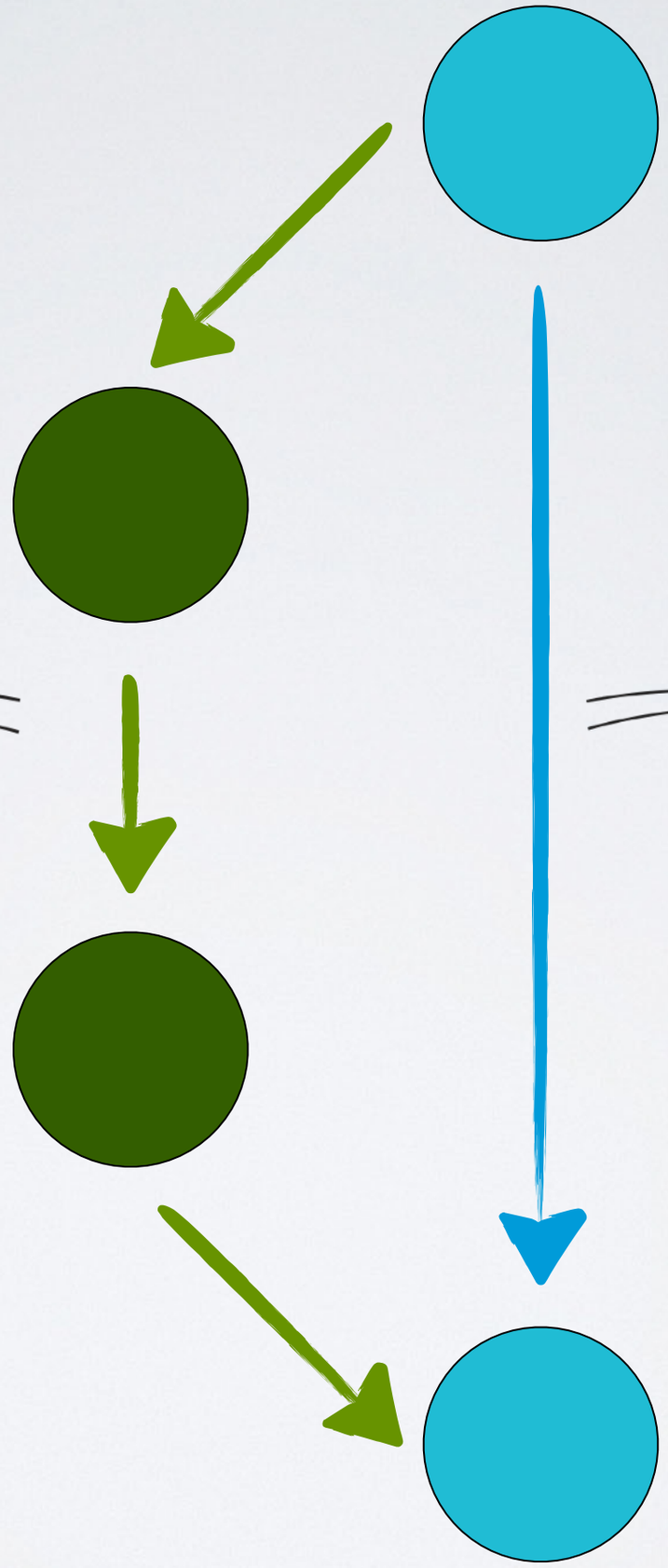
Open-source communities: The onion model



GitHub's pull-based development model

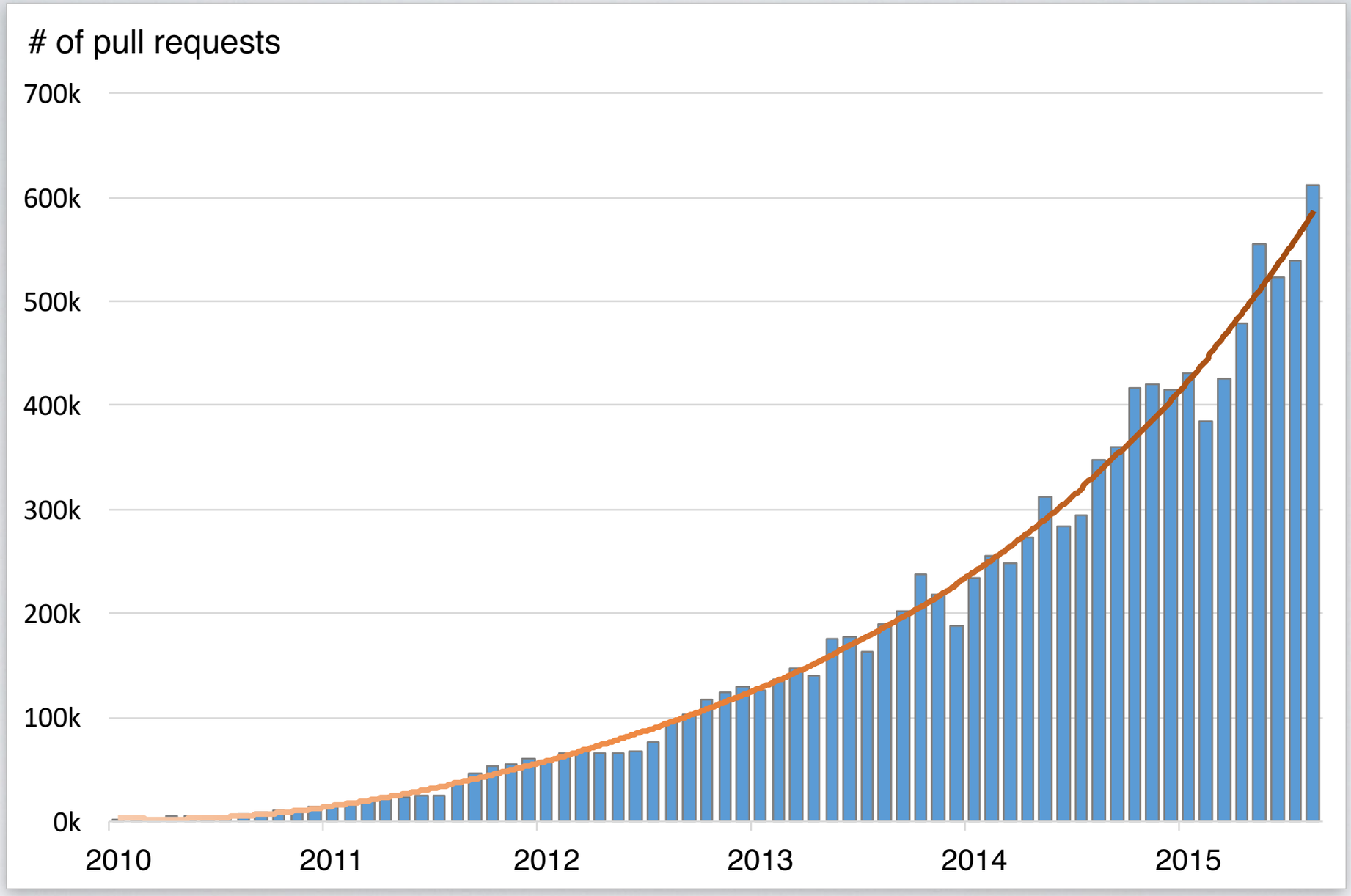


contributor



core developer (integrator)

GitHub's pull-based development model: A success story



projects with > 1 integrators

55% shared repository

45% use pull requests

2016

135k repos/month

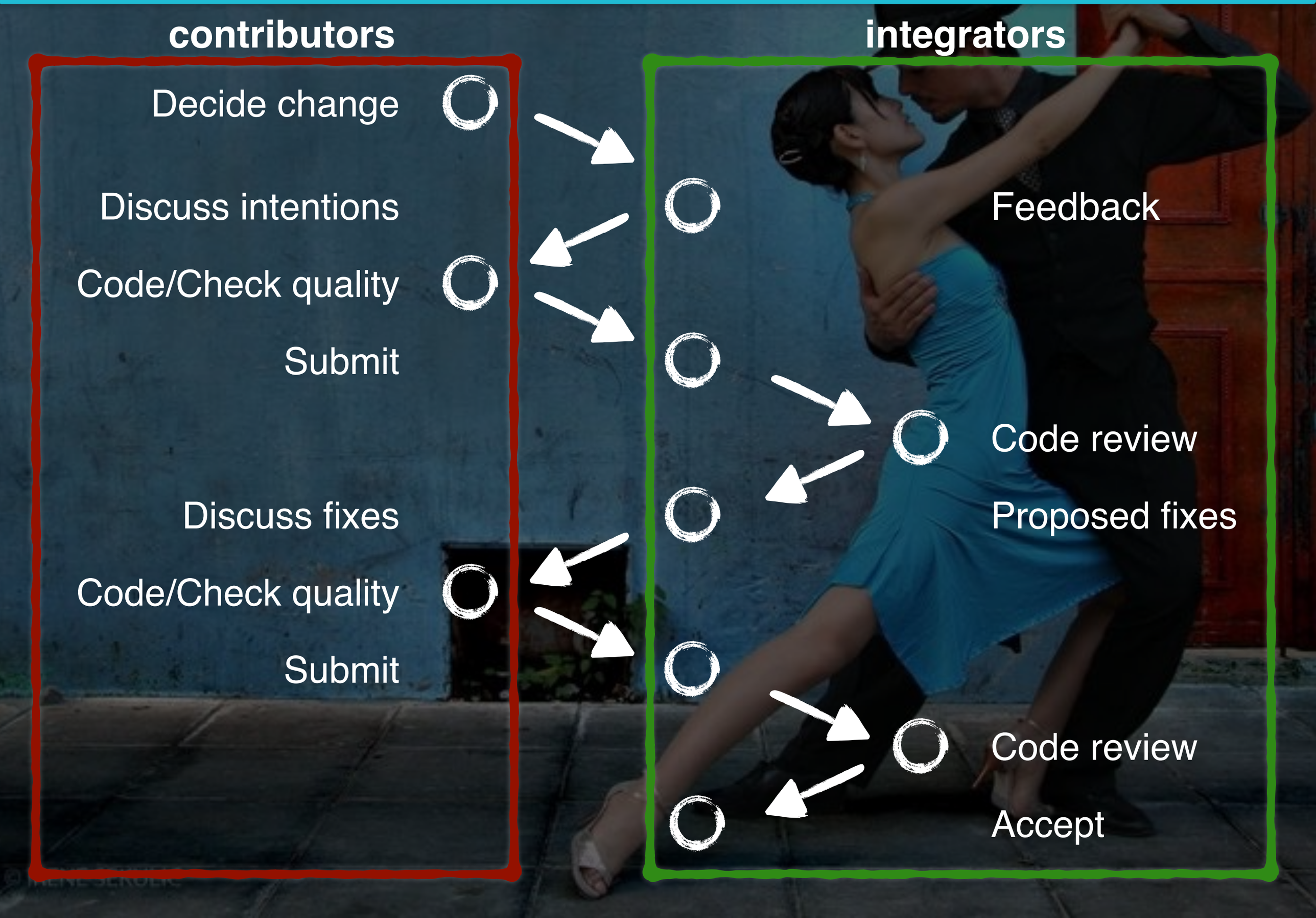
600k pull-requests/month

The integrators

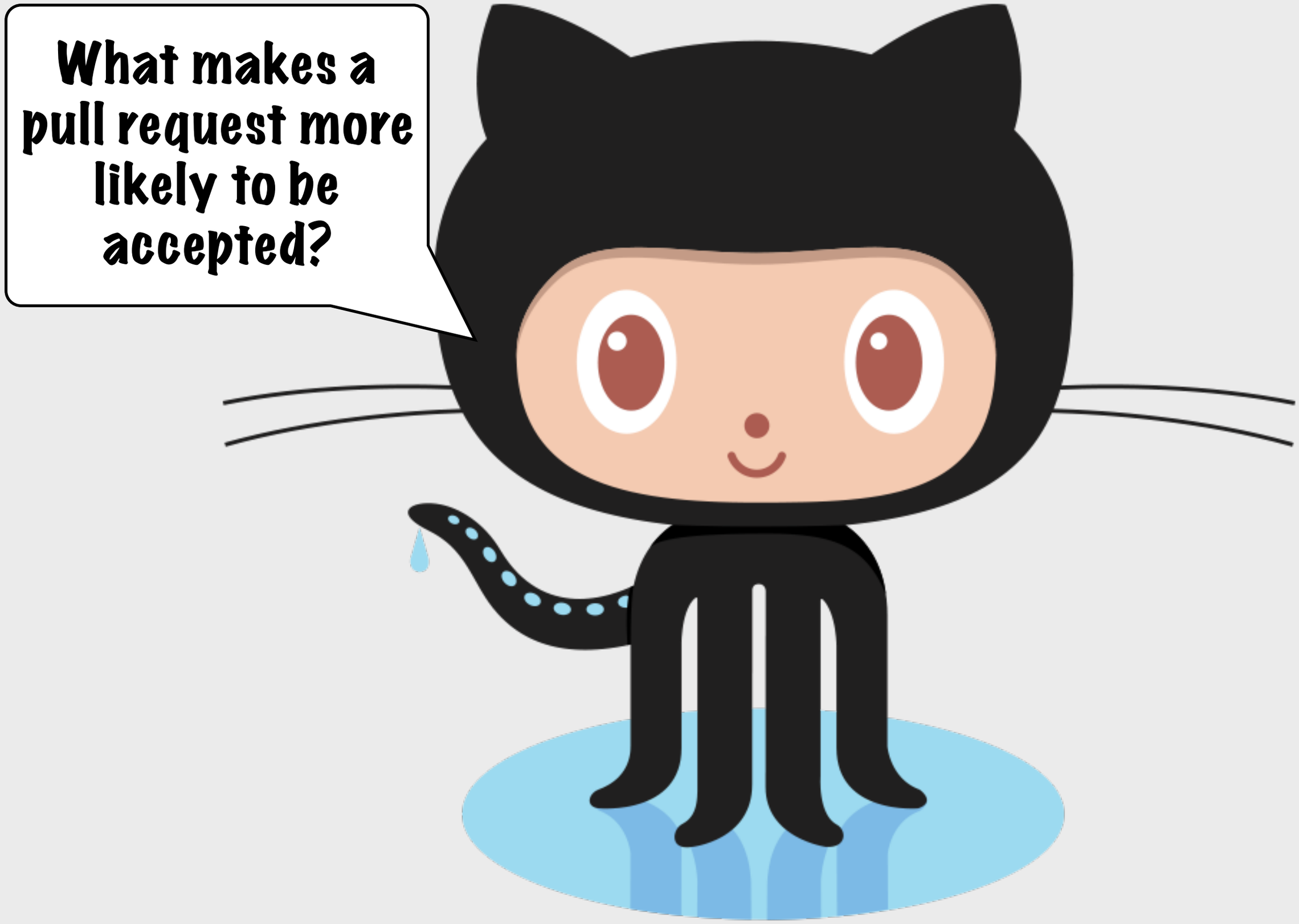
The contributors


© IRENE SEKULIC

The contributor's perspective



GitHub's pull-based development model



GHTorrent

http://ghtorrent.org