Engineering for Quality: Using Brainpower rather than Willpower

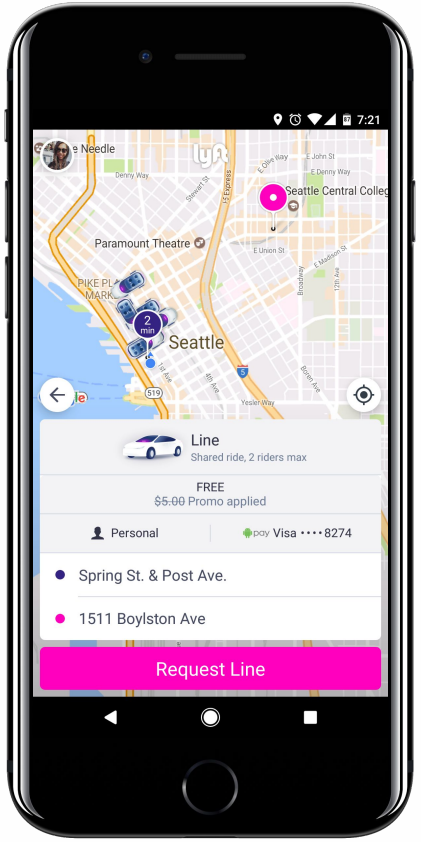
First, a little background

Who am I?

- Senior Engineering Manager at Lyft
- Previous experience at Google, Apple, Twitter, and a few startups
- Managed teams from 3 - 75 people

Lyft in Context

- US-based ridesharing company
- Explosive engineering team growth
- We started with a good basis — strong company culture




BE
YOURSELF


UPLIFT
OTHERS

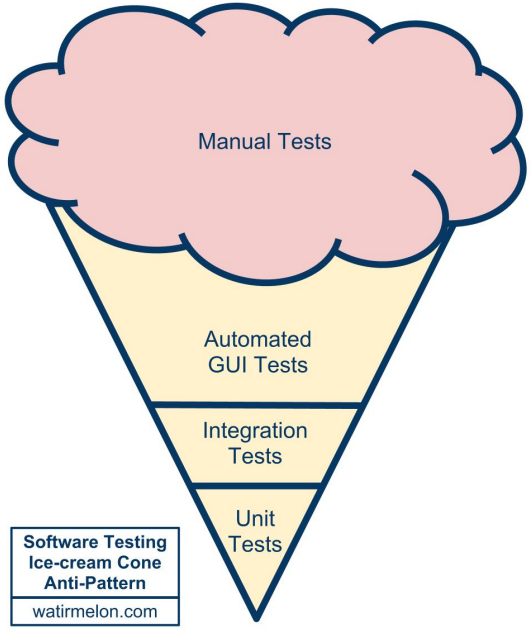

MAKE
IT
Happen

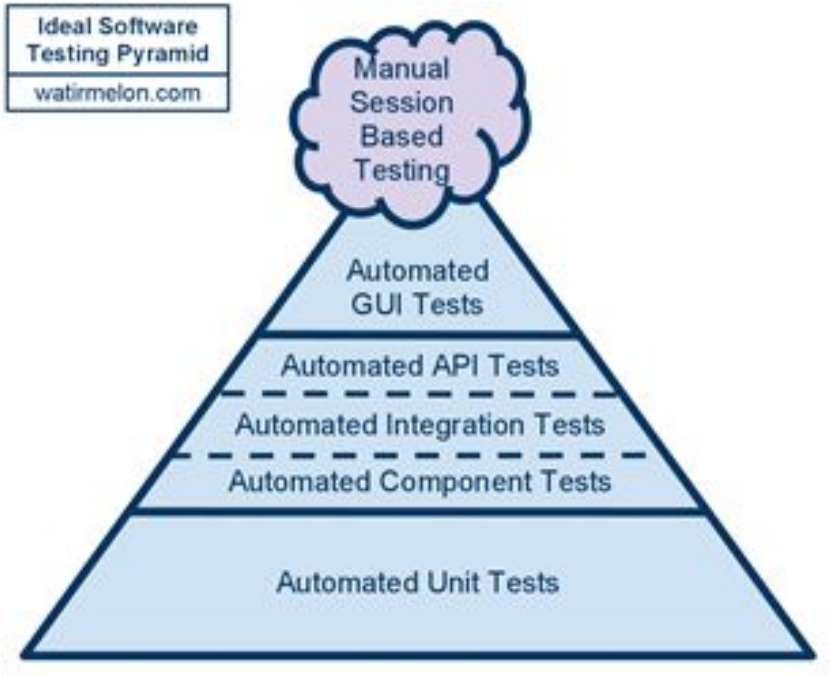

CREATE
FEARLESSLY

The starting point

- Never had tons of testers, but what we had were primarily manual
- Very little investment in automation
- Builds got tested, but chaos ensued

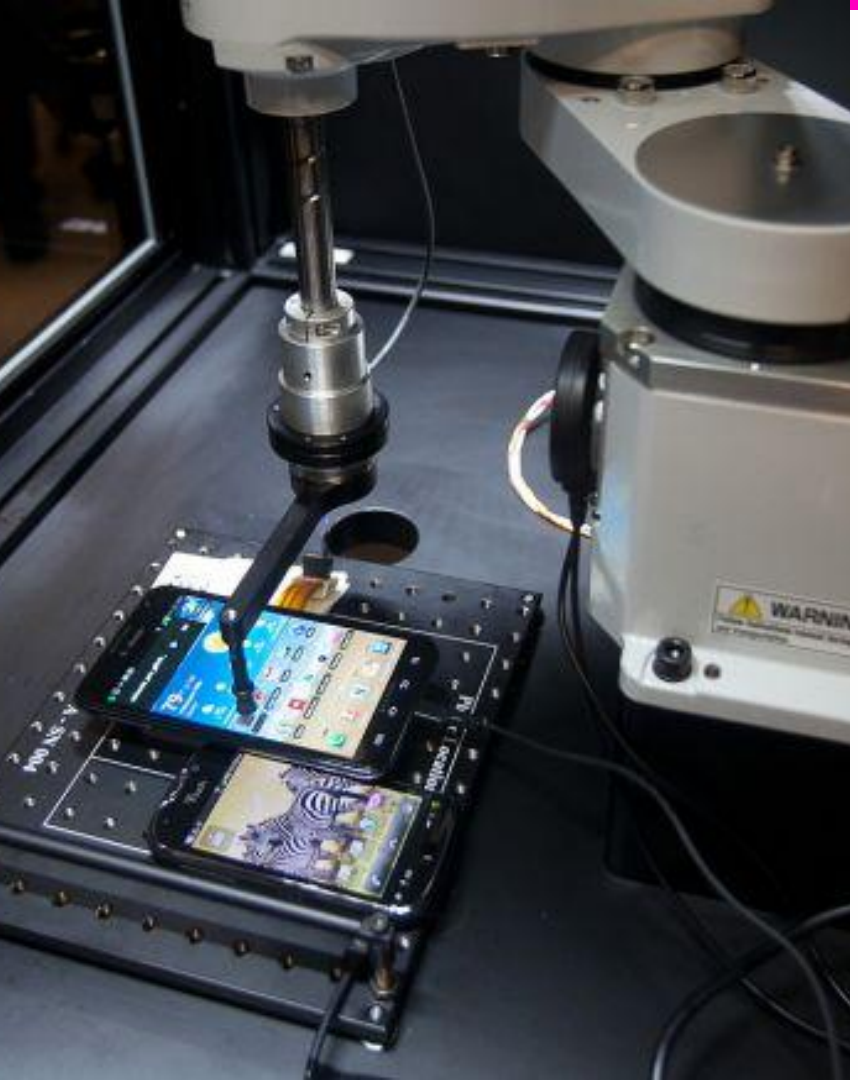
Testing ice cream cone vs testing pyramid





Our Tech Stack

- Microservice architecture for service side: python, go, c++
- iOS app written in Swift
- Android app written in RxJava
- Three (!) CI systems: Travis (Android), CircleCI (iOS), Jenkins (services) — though all submissions go through submit queue on Jenkins
- Github and associated webhooks
- Continuous deployment for services



Will Automation Save Us?

- Everyone wants automation
- Few people know why
- Overarching Philosophy:

Automate process rather than dictating it.

- The challenges:
  - coordination between many backend microservices — there are almost as many services as devs!
  - running a Onebox
  - test cases are very location dependent
  - tests need to be coordinated between driver and passenger — or multiple passengers in a Line ride
  - free-for-all in the developer process

Automation is a neutral good

- Cognitive science tells us

> *"In almost any domain where an individual repeatedly performs an activity, over time, the activity becomes increasingly automated."*

- The cost of this is that once a human "automates" a task, it's less conscious.
- Freeing humans from the more rote tasks results in better testing overall

Cognitive Neuroscience of Human Systems, Forsythe et al, CRC Press 2015

Step 1: Motivate the Team

- Lyft has a distributed QA model — no formal centralized Test leadership
- Establish a strong “umbrella team”
- Explain clearly how some things like test automation and regression test outsourcing are not in conflict with manual testing.
- Morph the team from gatekeepers into facilitators
- UI Automation written in python using Appium
  - running on Sauce Labs infrastructure for real devices and emulators
- simulated rides service for system tests

Step 2: Educate the development team

- Development teams are under pressure to get things shipped fast.
- Establish reliability of the system first
- Unit test jailing
- Integration test speed and reliability
- Testing enough
- Debugging broken tests
- Multiple layers of validation, even before UI Testing occurs.
- Make the process fun: emoji-based submit queue actions

Emoji-Based Actions

To initiate submission, instead of clicking the green merge button in your PR, type *submit this please* (or 🚀) in a PR comment. Your PR will be policy checked and then merged into master. All further commands and any issues that arise should be clearly explained by the submit queue bot in further PR comments. The rest of this document goes into further details on what all of these commands do.

If you would like Submit Queue to submit your PR after all tests pass, type *submit this please when tests pass* (aliases: 🚀 ✅, 💨)

- **submit this please and override checks** (🚀 🙏) This command can be used to bypass submit queue policy checks and immediately commit your PR to master. Note that any skipped checks will be written into the master history. Current checks include:
  - PR tests have passed
  - PR has +1
  - PR is not locked
  - PR is considered mergeable by GitHub

  (aliases: 🎰)
  - **submit this please when tests pass and override checks** (💨 🙏) This is the same as rocket pray but ignores everything *other than* tests passing.
- **test this please** (🔨) `$test_name` Re-runs all failing or pending tests. Takes in an optional test name which is everything to the right of `test-`. For example, to re-run `base-pr-test-unit`, type `:hammer: unit`. (aliases: 🍔)

By the way, WTH Samsung?!

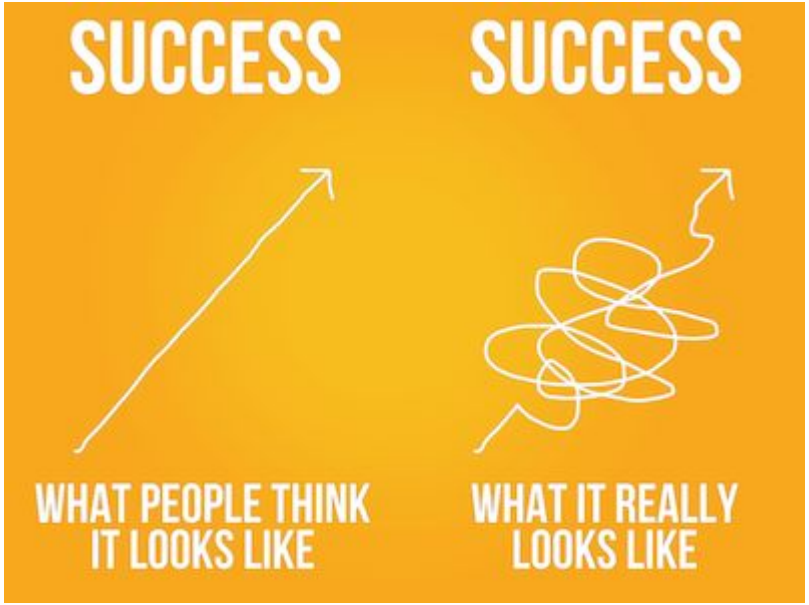
🍪 Cookie

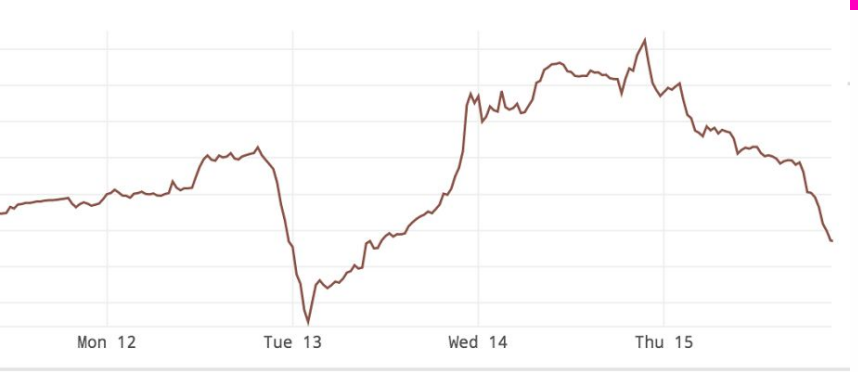
A cookie of any description, though appears as a chocolate chip cookie on most major platforms. On Samsung devices, this emoji displays as two saltine crackers instead of a cookie.

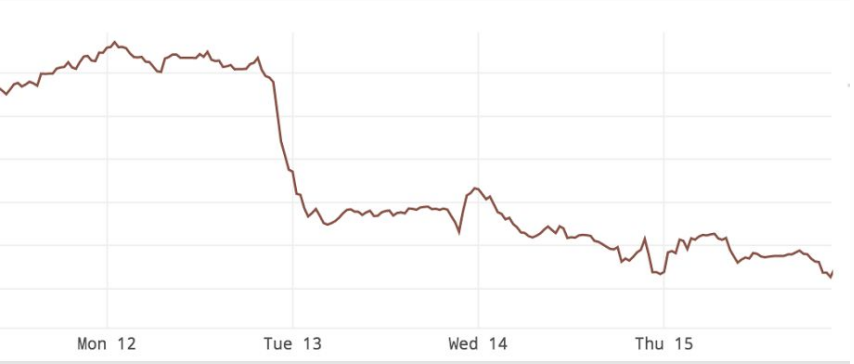
Apple	
Google	
Microsoft	
Samsung	

Step 3: The way forwards sometimes looks like a step backwards.

- We had a 1 week release cadence but it was untenable. So we went to a 2 week cadence.
- Adding an extra RC week allowed the dev team to better test before committing
- Beta was stronger, RC proved unnecessary over time







Step 4: Measure, iterate, repeat.

- We measure all the things (that make sense)
- Developer Productivity Metrics:
  - Number of times 🚀 🙏 (:rocket: :pray:) used
  - Number of times developers had to :hammer:
  - Time from PR to deploy, including interim steps (Time to first comment, time to first +1, time to merge to master)
- Test metrics
  - Bugs found in each release over time from each vector (Testlio, automation, internal manual, Beta)
  - Run time for the UI automation on each platform
  - Test reliability -- flakiness metric
- Release Metrics
  - Number of hotfixes, rollbacks
  - Number of issues found in production

What's Next?

- Native UI Automation for iOS and Android
- Moving our lab infrastructure in-house
- Next generation CI
- Further smoothing of the flow from PR check in to release

Questions?