Tampa Java User Group

Breeding 10x Developers

With Developer Productivity Engineering

Who is this guy?

Java Champions

Warner River Bees

codemonkey.fm

Apache Directory

openstack

THE APACHE SOFTWARE FOUNDATION

Maven™

Sonatype Nexus

source: Silicon Valley

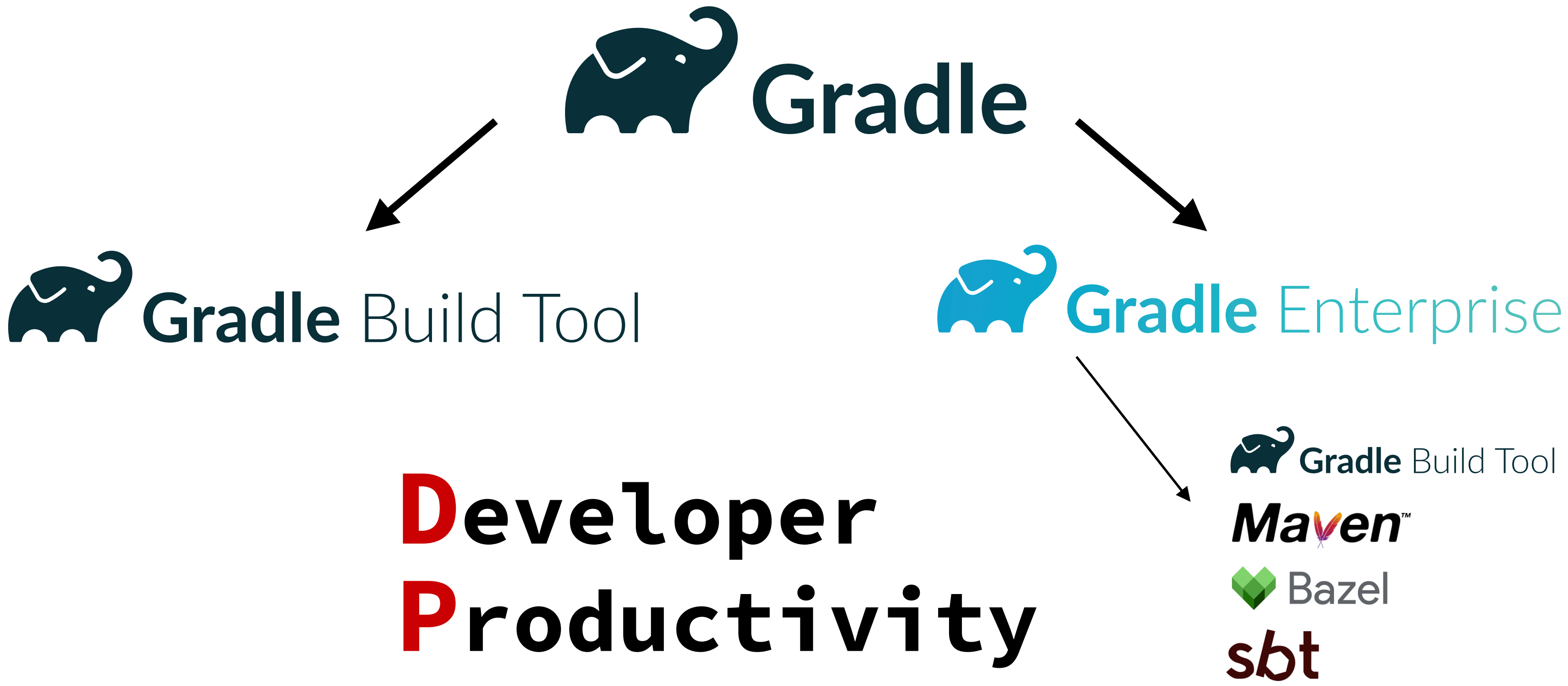
Gradle

VS

Gradle Build Tool

VS

Maven™

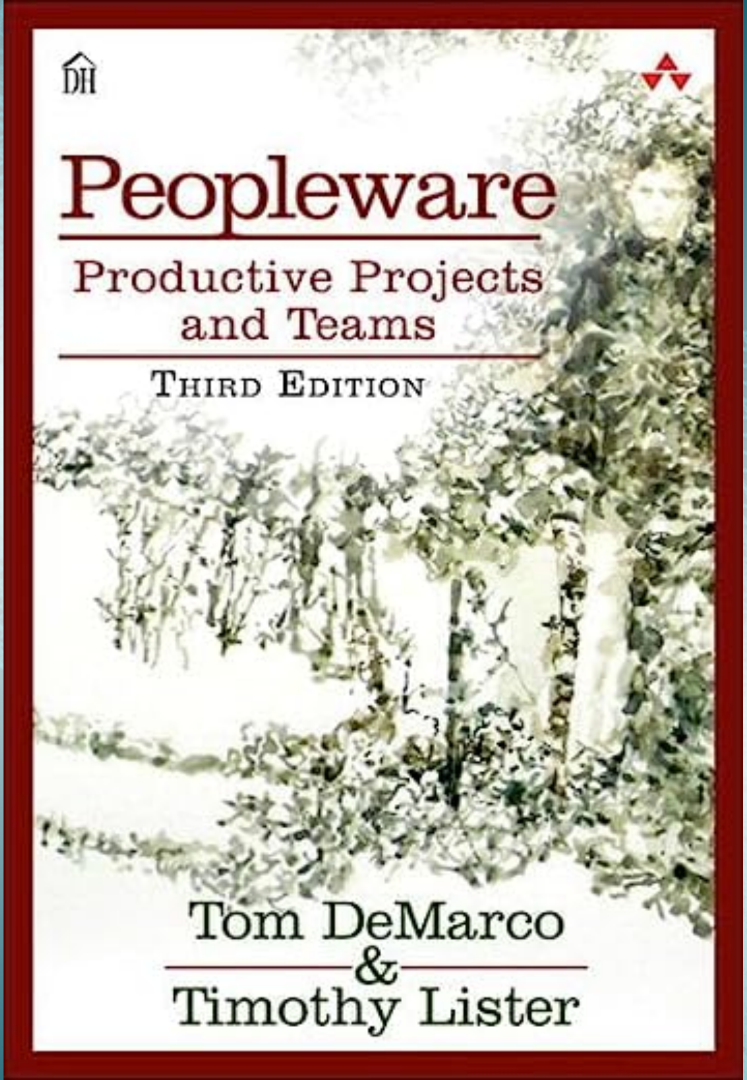


Developer
Productivity
Engineering

10X ENGINEER

EL CHUPACABRA

Myth Origin (probably)
The Coding War Games



Peopleware

Productive Projects and Teams

THIRD EDITION

Tom DeMarco & Timothy Lister

The “best” programmers outperformed the worst by **roughly a 10:1 ratio**
There were some interesting “non-factors”:

Language
Years of Experience
Number of Defects
Salary

What Mattered?

- Paired programmers from the same organization **performed at roughly the same level**
- The average **difference was only 21%** between paired participants
- They didn't work together on the task, but they **came from the same organization**
- **The best organizations performed 11.1x better than the worst**

"While this productivity differential among programmers is understandable, there is also a 10 to 1 difference in productivity among software organizations."

-Harlan D. Mills, Software Productivity

The best performers are clustering in some organizations while the worst performers are clustering in others.

Some companies are doing a lot worse than others.

Something about their environment and corporate culture is failing to attract and keep good people or is making it impossible for even good people to work effectively.

Average performance of those in the top quarter was 2.6 times better than that of those in the bottom quarter.

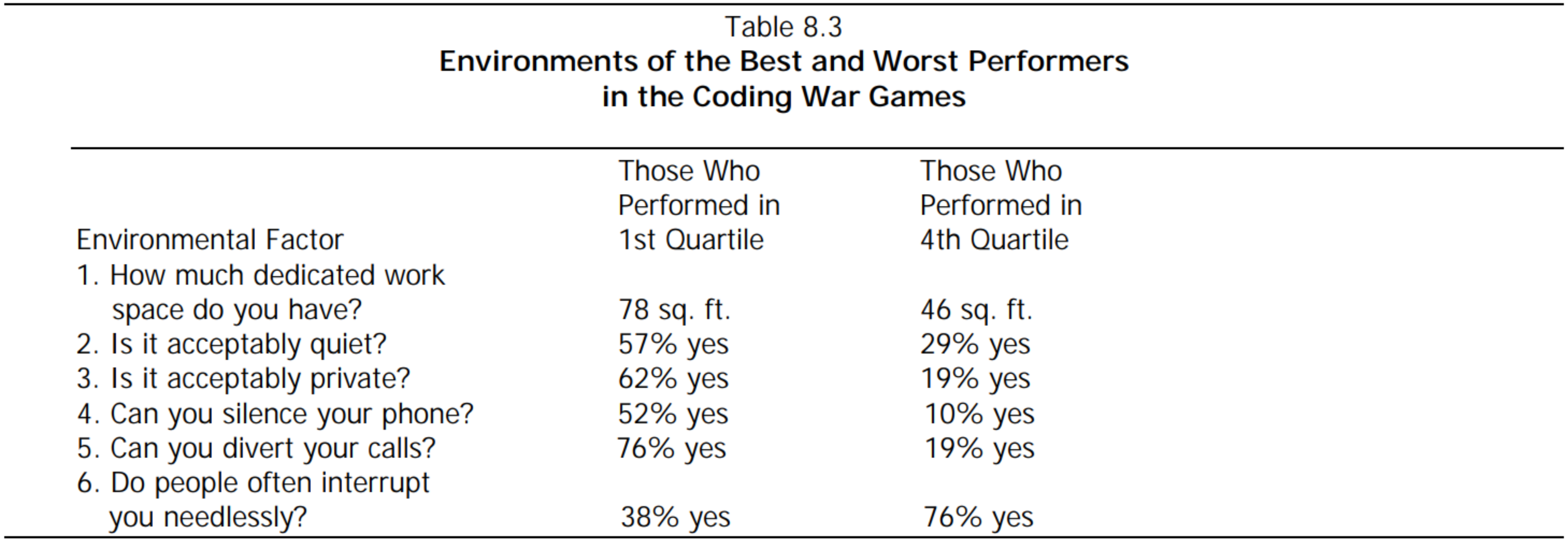
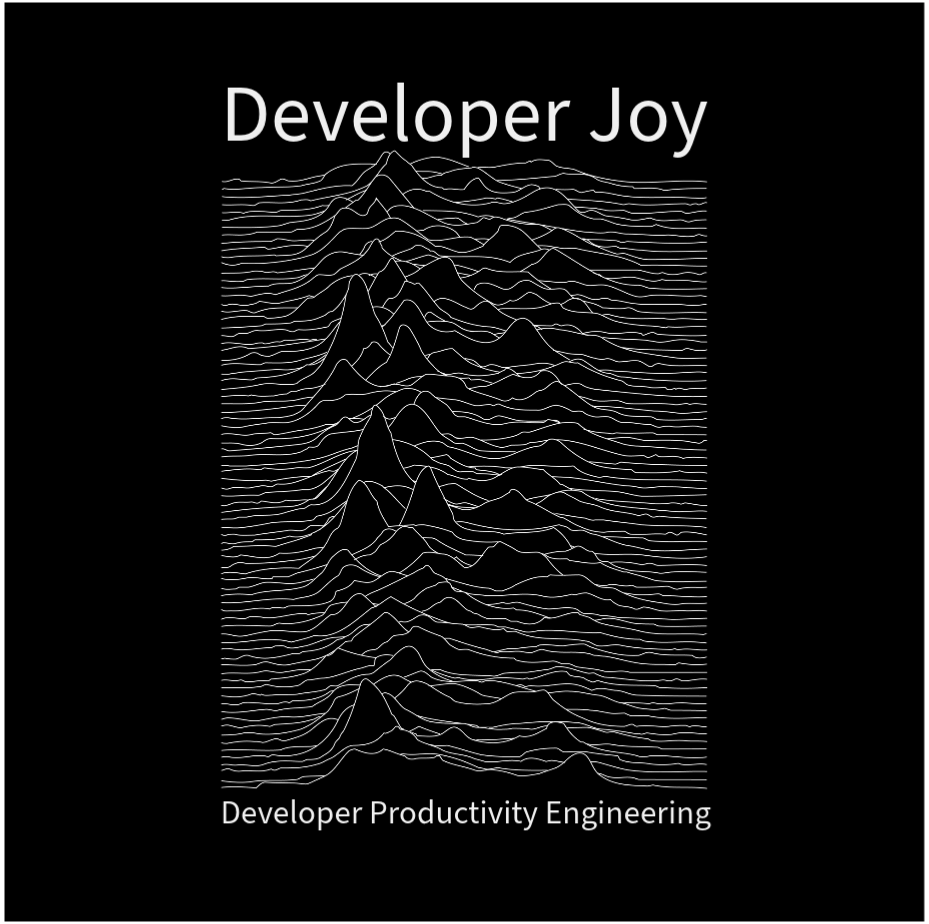


Table 8.3
Environments of the Best and Worst Performers in the Coding War Games

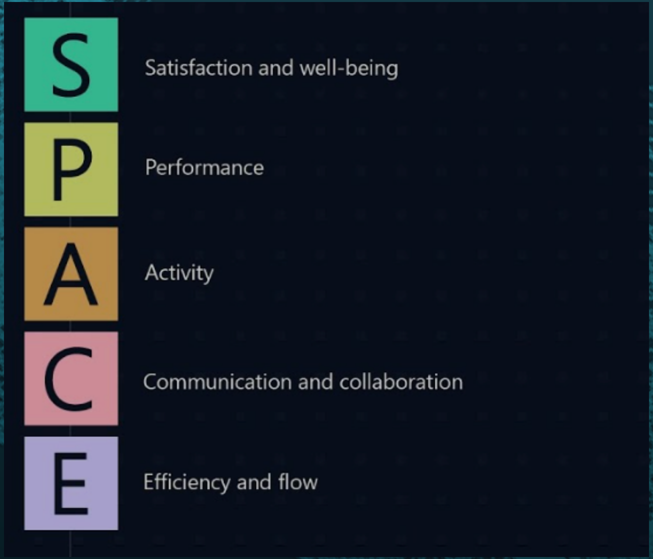
Environmental Factor	Those Who Performed in 1st Quartile	Those Who Performed in 4th Quartile
1. How much dedicated work space do you have?	78 sq. ft.	46 sq. ft.
2. Is it acceptably quiet?	57% yes	29% yes
3. Is it acceptably private?	62% yes	19% yes
4. Can you silence your phone?	52% yes	10% yes
5. Can you divert your calls?	76% yes	19% yes
6. Do people often interrupt you needlessly?	38% yes	76% yes

Though the phrase had not yet been coined, increased productivity came down to developer experience.

10x Organizations are Manufactured, Not Born


Developer Joy
Developer Productivity Engineering

… But Most Organizations Aren't Aligned



In a study dated April 27, 2022, between Microsoft and the University of Victoria in British Columbia, Developers and Managers were surveyed on their interpretation of the SPACE framework

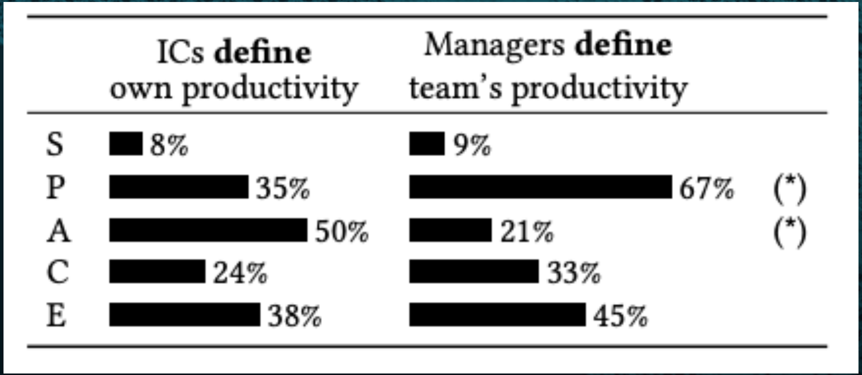
When surveyed with the following questions, **Developers and Managers answered much differently**

Developers

When thinking about your work, how do you define productivity?

Managers

When thinking about your team, how do you define productivity?



	ICs **define** own productivity	Managers **define** team's productivity	
S	8%	9%	
P	35%	67%	(*)
A	50%	21%	(*)
C	24%	33%	
E	38%	45%	

https://arxiv.org/pdf/2111.04302.pdf

DevOps, 12-Factor, Agile, etc, have still **not captured all bottlenecks**, friction, and obstacles to throughput

Many are **hiding in plain sight**, in the developer experience itself

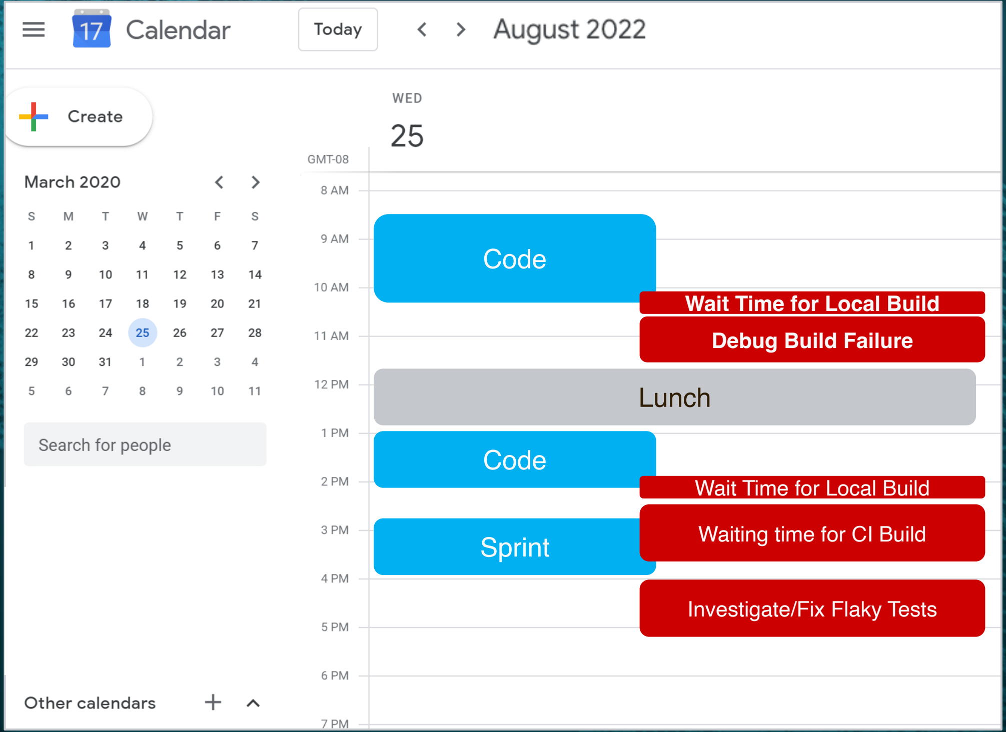
A 10x organization should think about reducing build and test feedback times, and improving the consistency and reliability of builds

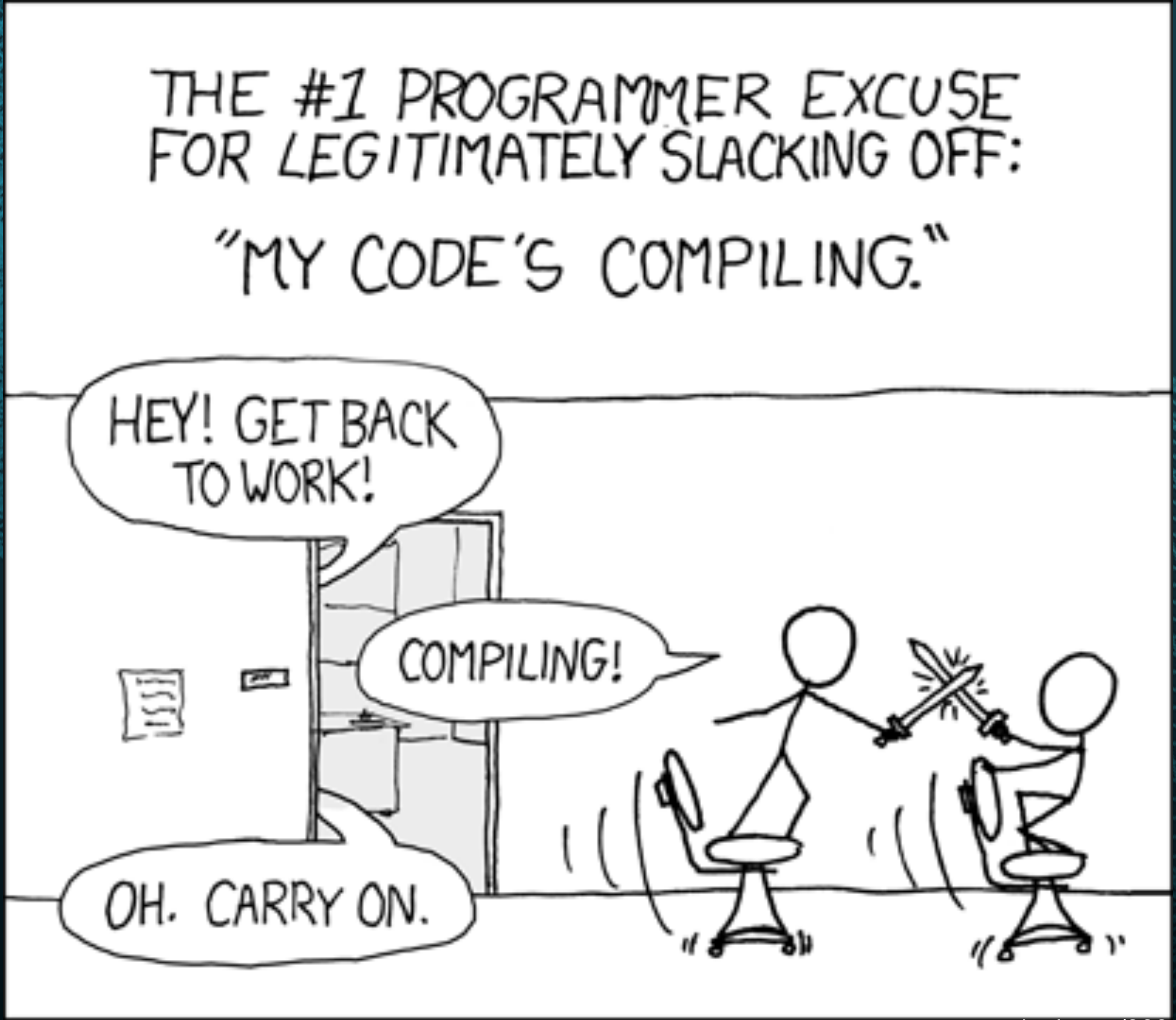
The only initiatives that will positively impact performance are ones which **increase throughput while simultaneously decreasing cost.**

It’s Time for Developer Productivity Engineering

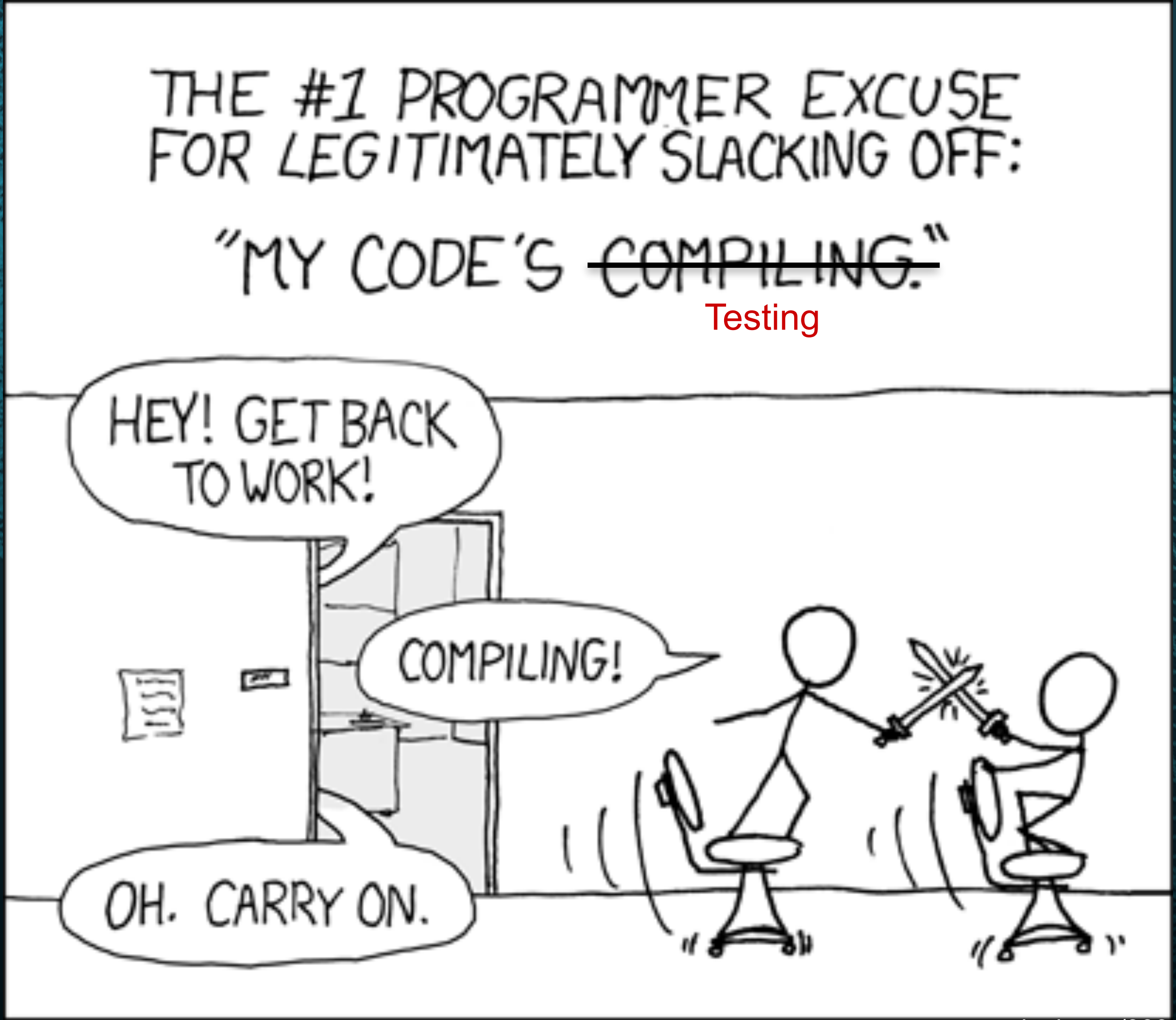
1970s+	1980s+	1990s+	2000s+	2010s+	2020+
JIT Manufacturing	Business Process Reengineering	Change management	Agile, Lean Six Sigma	DevOps	DPE

EXIT

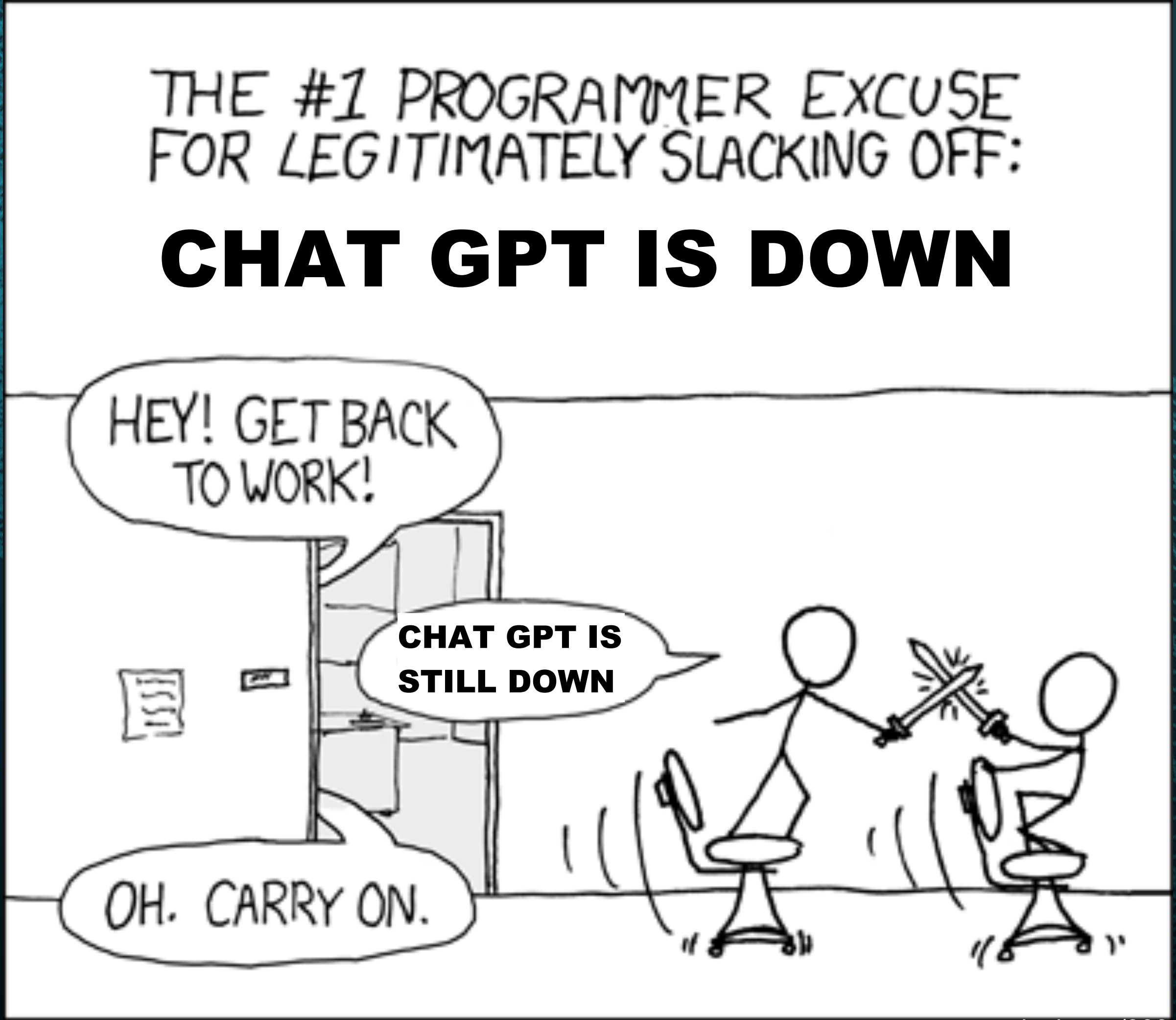

Calendar
Today
August 2022
Create
March 2020
S M T W T F S
Search for people
Other calendars
WED
25
GMT-08
8 AM
9 AM
10 AM
11 AM
12 PM
1 PM
2 PM
3 PM
4 PM
5 PM
6 PM
7 PM
Code
Wait Time for Local Build
Debug Build Failure
Lunch
Code
Wait Time for Local Build
Waiting time for CI Build
Sprint
Investigate/Fix Flaky Tests


THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY CODE'S COMPILING."
HEY! GET BACK TO WORK!
COMPILING!
OH. CARRY ON.

xkcd.com/303

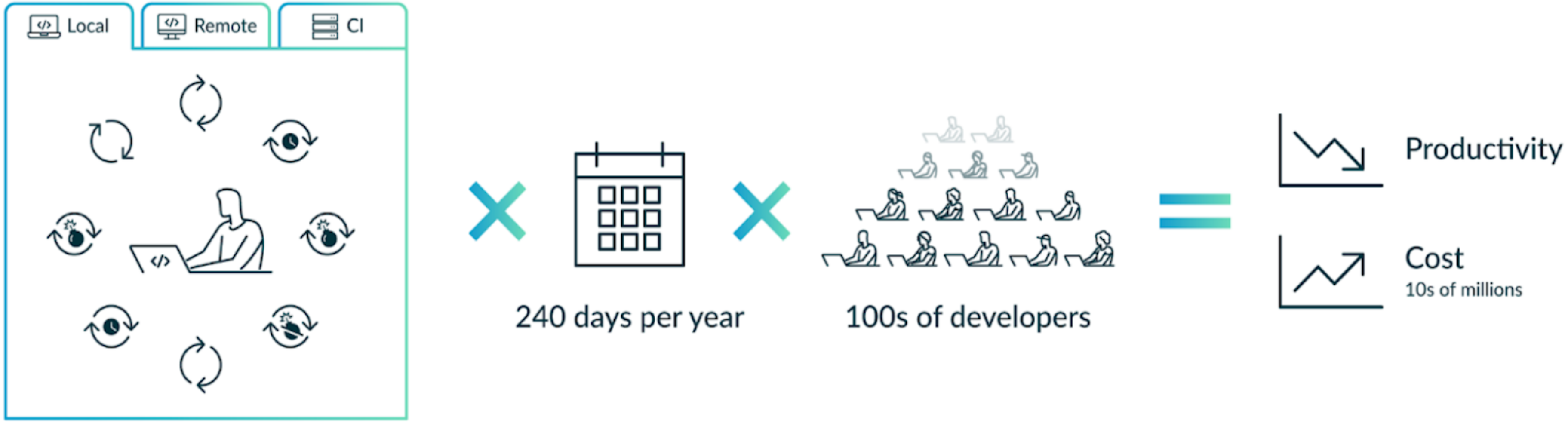

THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY CODE'S ~~COMPILING.~~"
Testing
HEY! GET BACK TO WORK!
COMPILING!
OH. CARRY ON.

xkcd.com/303


THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:
CHAT GPT IS DOWN
HEY! GET BACK TO WORK!
CHAT GPT IS STILL DOWN
OH. CARRY ON.

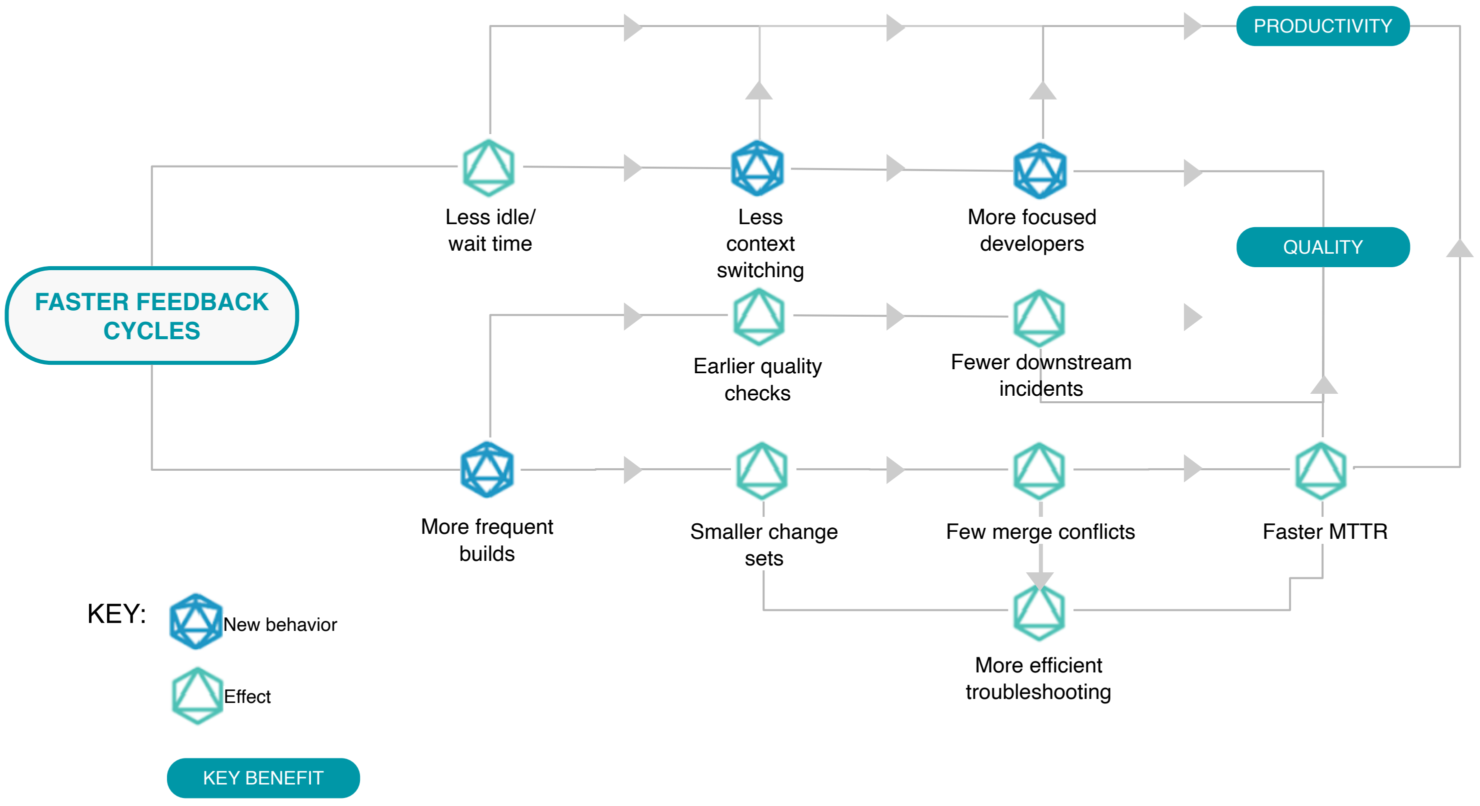
xkcd.com/303

What Problems Does DPE Solve?



- This takes too long!
- This takes too long to fix
- This should have been observable

The anatomy and importance of fast feedback cycles



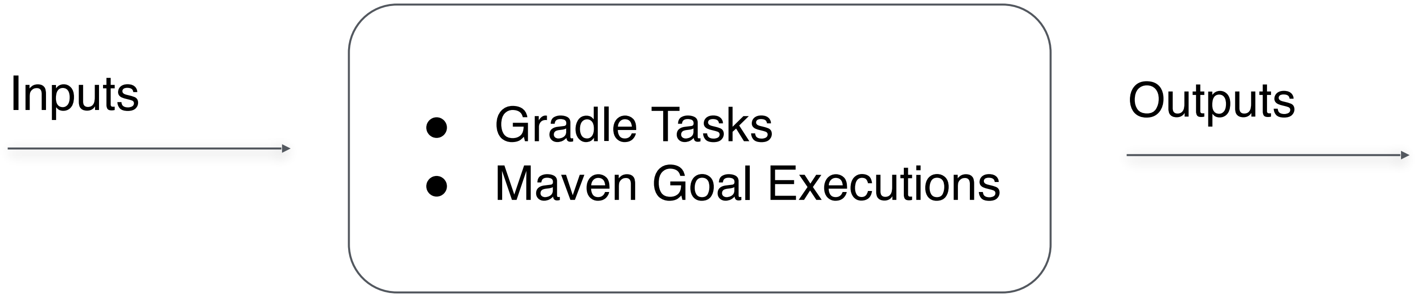
Build caching delivers *fast build and test feedback cycles*

Build Caching

- Introduced to the Java world by Gradle in 2017
- Maven has an open source build cache too
- **Used by leading technology companies** like Google and Facebook
- Can support both **user local and remote caching** for distributed teams

- Build caches are **complementary to dependency caches**, not mutually exclusive:
  - A dependency cache caches **fully compiled dependencies**
  - A build cache accelerates **building a single source repository**
  - A build cache caches build actions (e.g. Gradle tasks or Maven goals)

What is a Build Cache?



When the inputs have not changed, the **output can be reused** from a previous run.

Cache Key/Value Calculation

The ***cacheKey*** for Gradle Tasks/Maven Goals is based on the Inputs:

```
cacheKey(javaCompile) = hash(sourceFiles,
                             jdk version,
                             classpath,
                             compiler args)
```

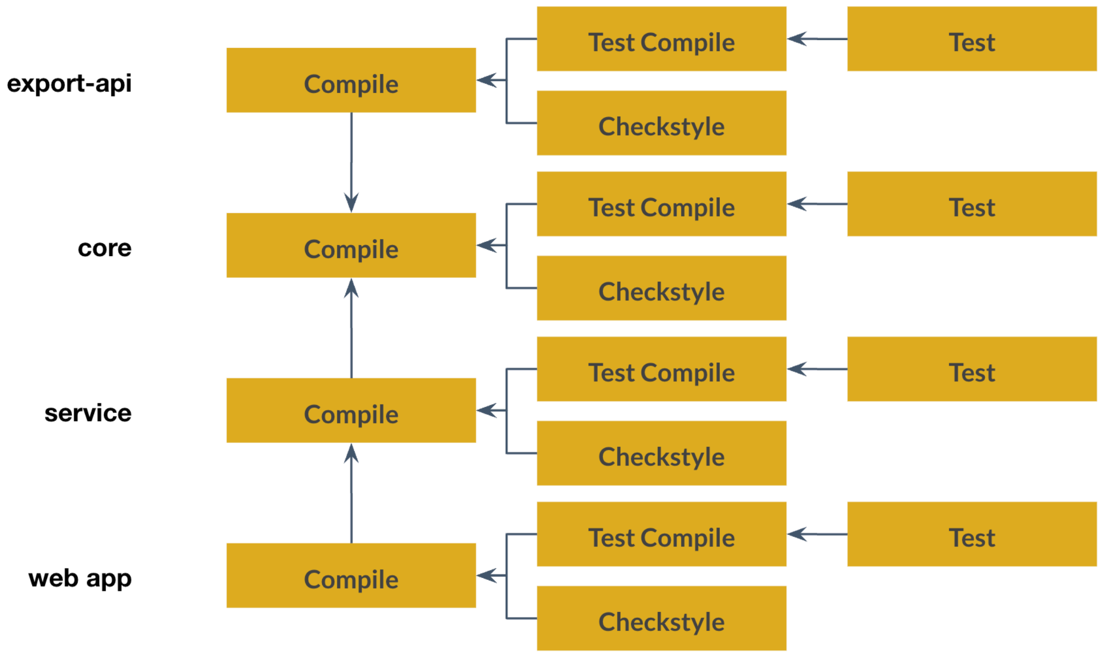
The ***cacheEntry*** contains the output:

```
cacheEntry[cacheKey(javaCompile)] = fileTree(classFiles)
```

For more information, see:

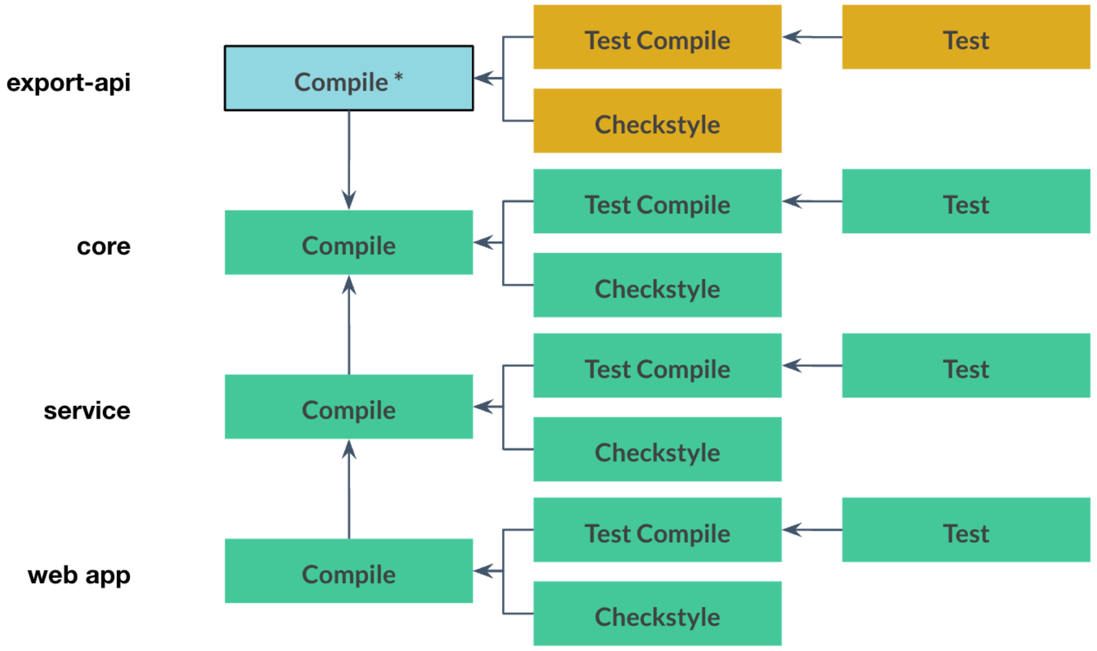
https://docs.gradle.org/current/userguide/build_cache.html

When not using the build cache, with Maven any change will require a **full build**. For Gradle this is the case when doing clean builds and switching between branches.



Task/Goal needs to be executed
Task/Goal retrieved from cache

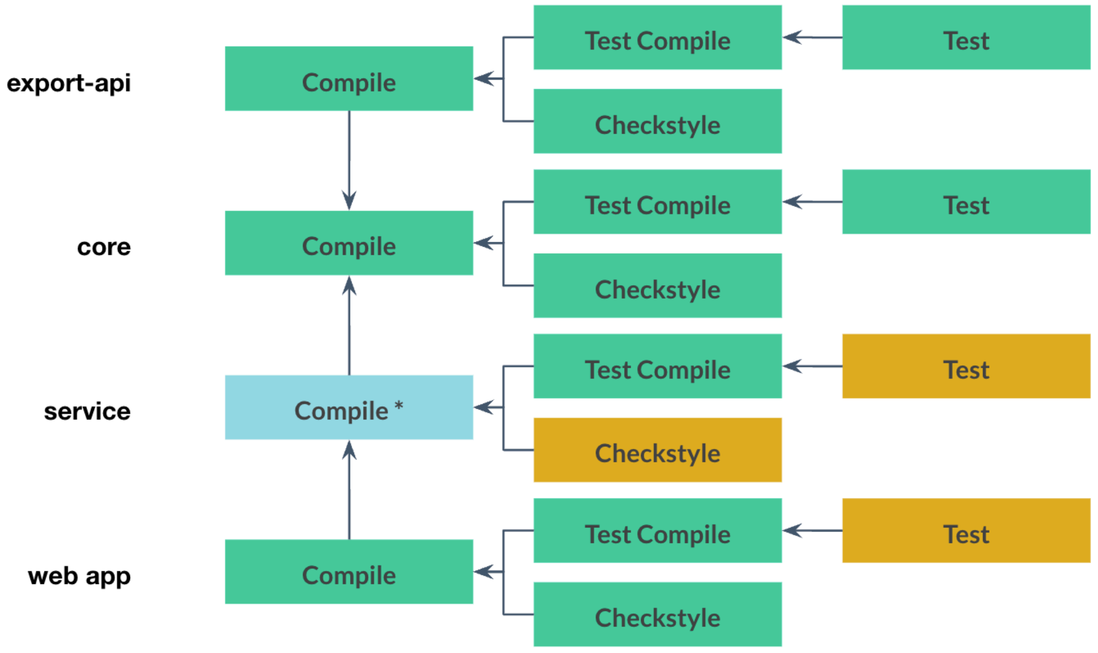
Changing an **public method** in the **export-api** module



🟨 Task/Goal needs to be executed

🟩 Task/Goal retrieved from cache

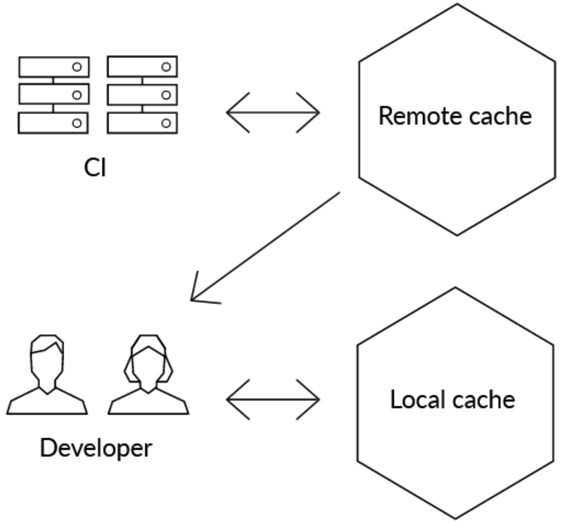
Changing an **implementation detail** of a method in the **service** module



🟨 Task/Goal needs to be executed

🟩 Task/Goal retrieved from cache

Remote Build Cache



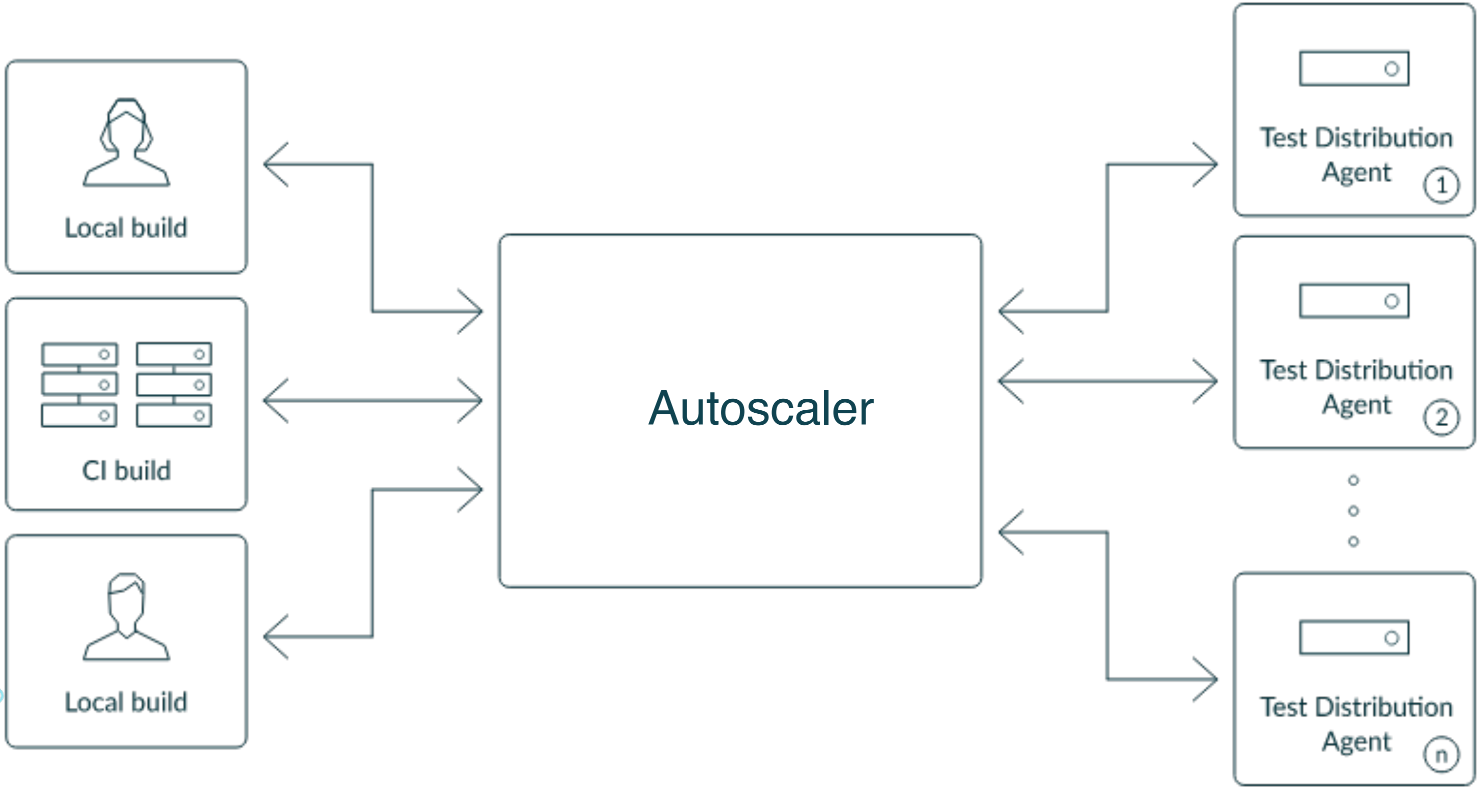
- Shared among different machines
- Speeds up development for the whole team
- Reuses build results among CI agents/jobs and individual developers

Reproducible Builds

reproducible-builds.org

Test distribution *can make tests even faster*

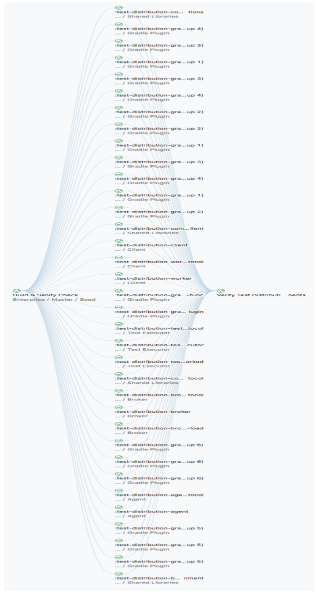
How it works


Local build
CI build
Local build
Autoscaler
Test Distribution Agent 1
Test Distribution Agent 2
Test Distribution Agent n

Existing solutions - CI fanout

Test execution is distributed by manually partitioning the test set and then running partitions in parallel on several CI nodes.

```
pipeline {
  stage('compile') { ... }
  parallelStage('test') {
    step {
      sh './gradlew :testGroup1'
    }
    step {
      sh './gradlew :testGroup2'
    }
    step {
      sh './gradlew :testGroup3'
    }
  }
}
```


Build & Sanity Check
Enterprise / Master / Seed
:test-distribution-co... tions
... / Shared Libraries
:test-distribution-gra... up 4)
... / Gradle Plugin
:test-distribution-gra... up 3)
... / Gradle Plugin
:test-distribution-gra... up 1)
... / Gradle Plugin
:test-distribution-gra... up 3)
... / Gradle Plugin
:test-distribution-gra... up 4)
... / Gradle Plugin
:test-distribution-gra... up 2)
... / Gradle Plugin
:test-distribution-gra... up 2)
... / Gradle Plugin
:test-distribution-gra... up 1)
... / Gradle Plugin
:test-distribution-gra... up 3)
... / Gradle Plugin
:test-distribution-gra... up 4)
... / Gradle Plugin
:test-distribution-gra... up 1)
... / Gradle Plugin
:test-distribution-gra... up 2)
... / Gradle Plugin
:test-distribution-com... lient
... / Shared Libraries
:test-distribution-client
... / Client
:test-distribution-wor... tocol
... / Client
:test-distribution-worker
... / Client
:test-distribution-gra... -func
... / Gradle Plugin
:test-distribution-gra... lugin
... / Gradle Plugin
:test-distribution-test... tocol
... / Test Executor
:test-distribution-tes... cutor
... / Test Executor
:test-distribution-tes... orked
... / Test Executor
:test-distribution-co... tocol
... / Shared Libraries
:test-distribution-bro... tocol
... / Broker
:test-distribution-broker
... / Broker
:test-distribution-bro... -load
... / Broker
:test-distribution-gra... up 6)
... / Gradle Plugin
:test-distribution-gra... up 6)
... / Gradle Plugin
:test-distribution-gra... up 6)
... / Gradle Plugin
:test-distribution-age... tocol
... / Agent
:test-distribution-agent
... / Agent
:test-distribution-gra... up 5)
... / Gradle Plugin
:test-distribution-gra... up 5)
... / Gradle Plugin
:test-distribution-gra... up 5)
... / Gradle Plugin
:test-distribution-b... nment
... / Shared Libraries
Verify Test Distributi... nents

See https://builds.gradle.org/project/Gradle for an example of this strategy

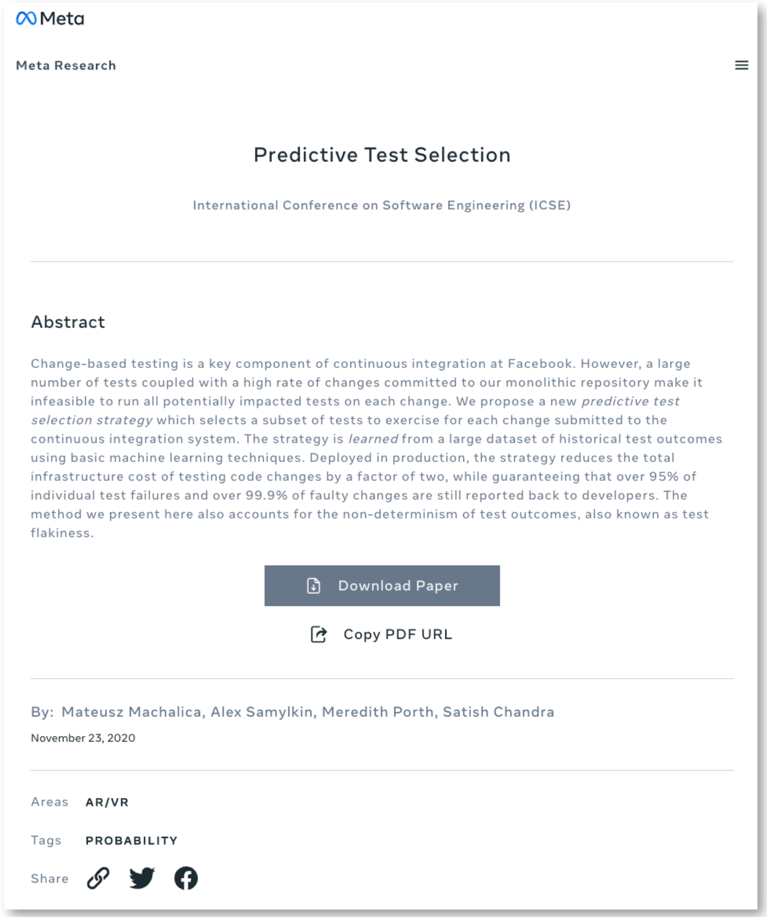
Assessment of existing solutions

- **Build Caching** is great in many cases but doesn't help when test inputs have changed.
- **Single machine parallelism** is limited by that machine's resources.
- **CI fanout** does not help during local development, is inefficient (in particular on ephemeral CI agents or without build cache), requires manual setup and test partitioning, and result collection/aggregation

NETFLIX

Netflix reduced a 62-minute test cycle time down to just under 5 minutes!

***Predictive Test Selection** leads to greater efficiencies*


Meta

Meta Research

Predictive Test Selection

International Conference on Software Engineering (ICSE)

Abstract

Change-based testing is a key component of continuous integration at Facebook. However, a large number of tests coupled with a high rate of changes committed to our monolithic repository make it infeasible to run all potentially impacted tests on each change. We propose a new *predictive test selection strategy* which selects a subset of tests to exercise for each change submitted to the continuous integration system. The strategy is *learned* from a large dataset of historical test outcomes using basic machine learning techniques. Deployed in production, the strategy reduces the total infrastructure cost of testing code changes by a factor of two, while guaranteeing that over 95% of individual test failures and over 99.9% of faulty changes are still reported back to developers. The method we present here also accounts for the non-determinism of test outcomes, also known as test flakiness.

Download Paper

Copy PDF URL

By: Mateusz Machalica, Alex Samylkin, Meredith Porth, Satish Chandra

November 23, 2020

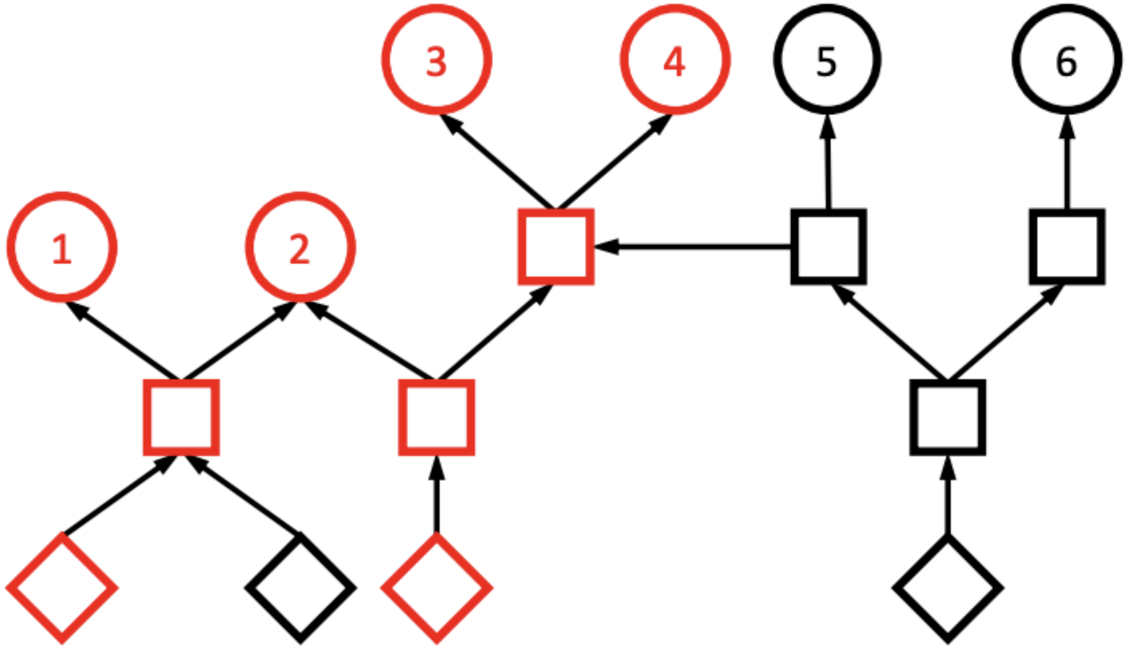
Areas AR/VR

Tags PROBABILITY

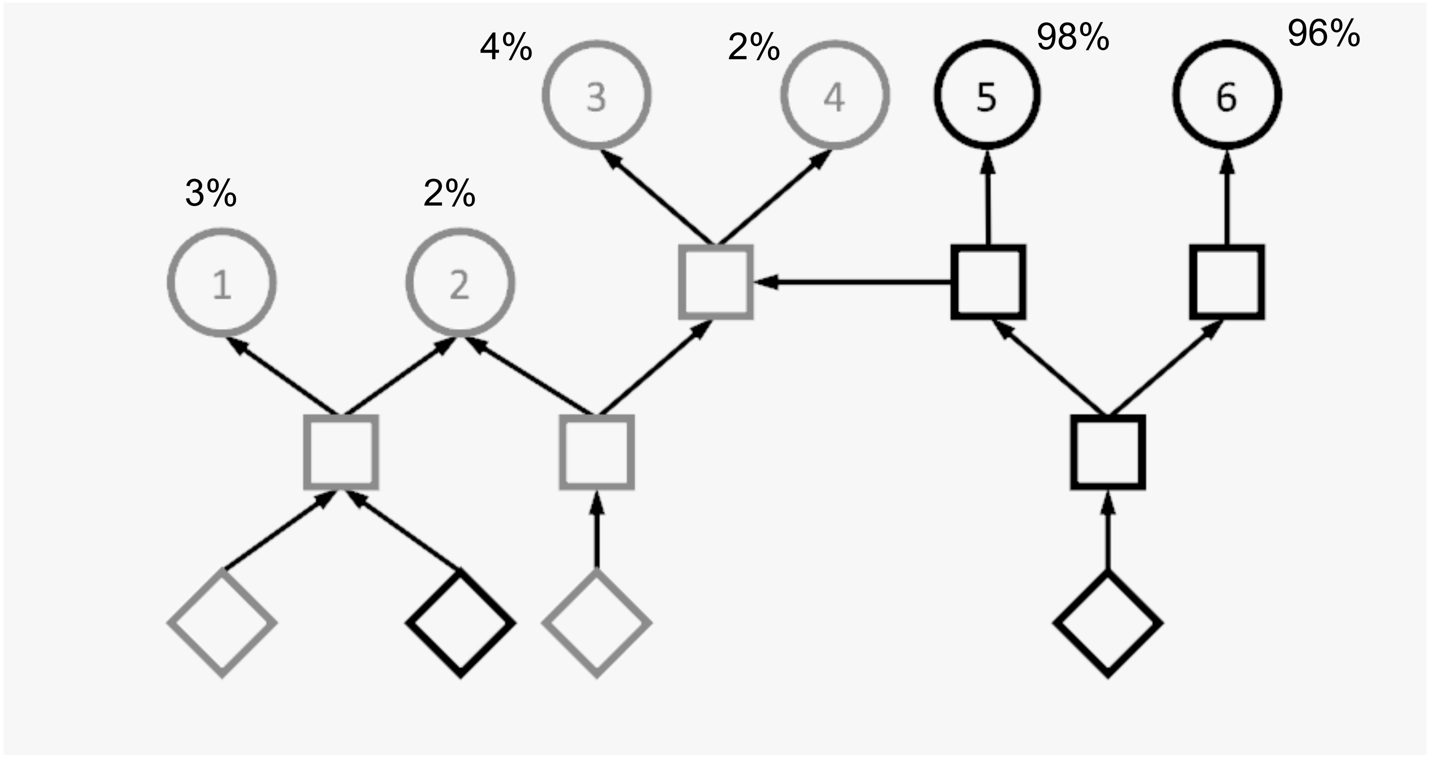
Share

https://research.facebook.com/publications/predictive-test-selection/

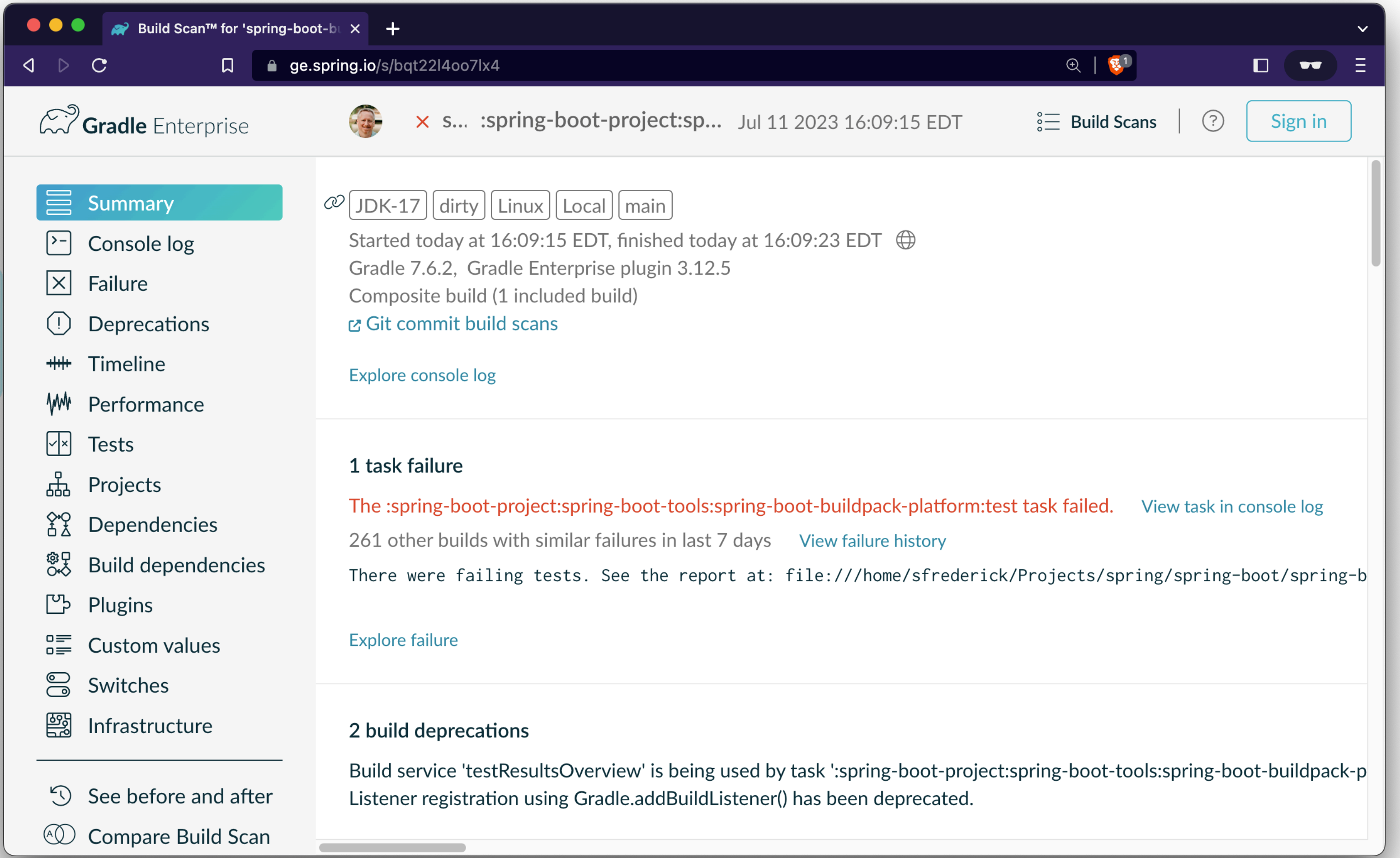
Conventional Test Selection Approach



Predictive Test Selection Approach



***Build Scans** speeds up troubleshooting*



Started today at 16:09:15 EDT, finished today at 16:09:23 EDT
Gradle 7.6.2, Gradle Enterprise plugin 3.12.5
Composite build (1 included build)
Git commit build scans

Explore console log

1 task failure

The :spring-boot-project:spring-boot-tools:spring-boot-buildpack-platform:test task failed. View task in console log
261 other builds with similar failures in last 7 days View failure history

```
There were failing tests. See the report at: file:///home/sfrederick/Projects/spring/spring-boot/spring-b
```

Explore failure

2 build deprecations

Build service 'testResultsOverview' is being used by task ':spring-boot-project:spring-boot-tools:spring-boot-buildpack-p
Listener registration using Gradle.addBuildListener() has been deprecated.

Without focus, problems can sneak back in…

- Infrastructure changes
  - Binary management
  - Caching
  - CI agents
- New annotation processors or versions of annotation processors
- Build logic configurations settings
  - Build tool version and plugins
  - Compiler and/or Memory settings
- Code refactoring
- New office locations
- Without observability, it is impossible to have a great and fast developer experience.

"You can observe a lot by just watching"

- Yogi Berra, Catcher and Philosopher

Performance Insights

Are you tracking local build and test times?

~~Is the cycle fast enough?~~

Is the cycle as **fast as it can possibly be?**

GRADLE ENTERPRISE CUSTOMER STATISTIC

DPE Fosters Developer Joy

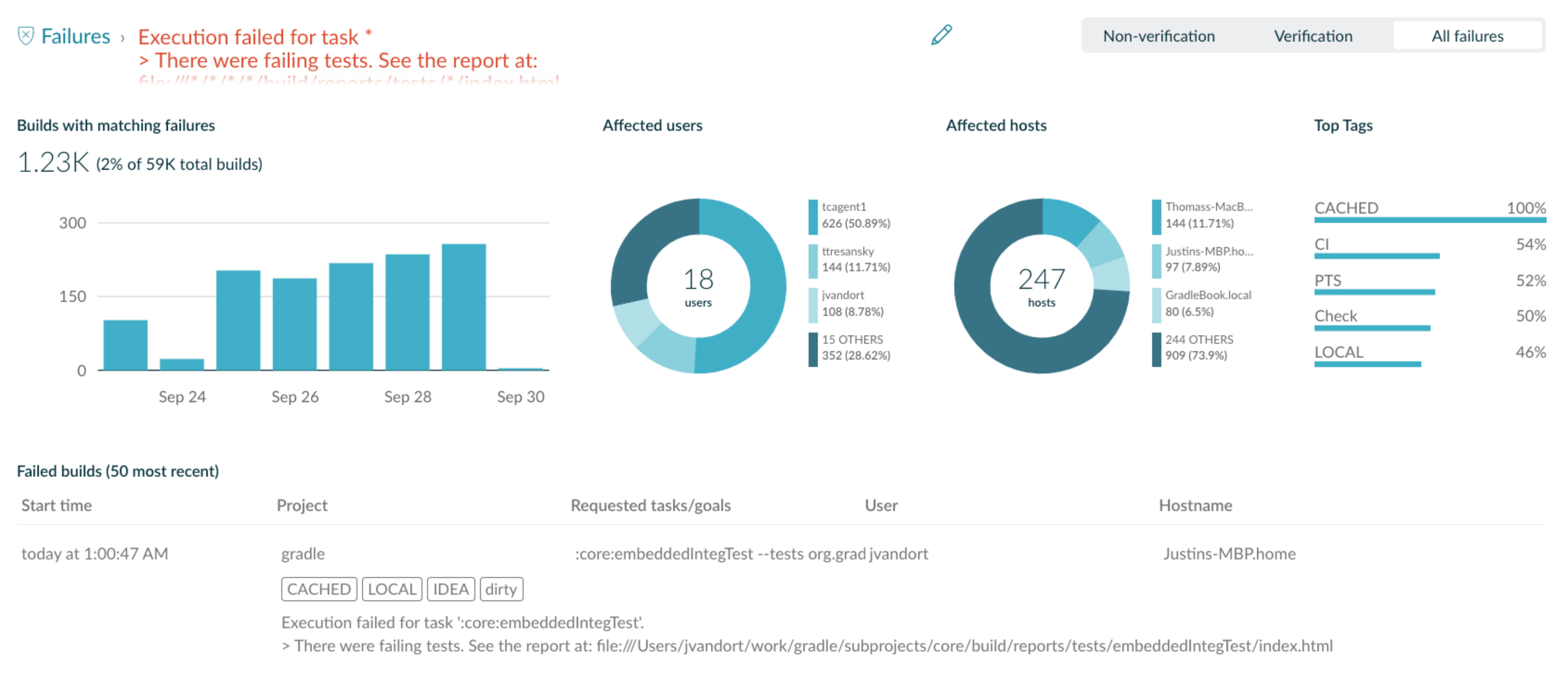
84% of surveyed users agree that DPE's impact on their toolchain makes their job more enjoyable.

Source: TechValidate survey of 51 users of Gradle Enterprise

✓ Validated Published: Jul. 2, 2023 TVID: 930-05A-A5F

Gradle Enterprise | TechValidate by SurveyMonkey

DPE Organizations Eliminate Avoidable Failures



Failures › Execution failed for task *
> There were failing tests. See the report at:
file:///*/*/*/*/build/reports/tests/*/index.html

Non-verification | Verification | All failures

Builds with matching failures
1.23K (2% of 59K total builds)

Affected users
18 users
tcagent1 626 (50.89%)
ttresansky 144 (11.71%)
jvandort 108 (8.78%)
15 OTHERS 352 (28.62%)

Affected hosts
247 hosts
Thomass-MacB... 144 (11.71%)
Justins-MBP.ho... 97 (7.89%)
GradleBook.local 80 (6.5%)
244 OTHERS 909 (73.9%)

Top Tags

CACHED	100%
CI	54%
PTS	52%
Check	50%
LOCAL	46%

Failed builds (50 most recent)

Start time	Project	Requested tasks/goals	User	Hostname
today at 1:00:47 AM	gradle	:core:embeddedIntegTest --tests org.grad	jvandort	Justins-MBP.home

CACHED LOCAL IDEA dirty

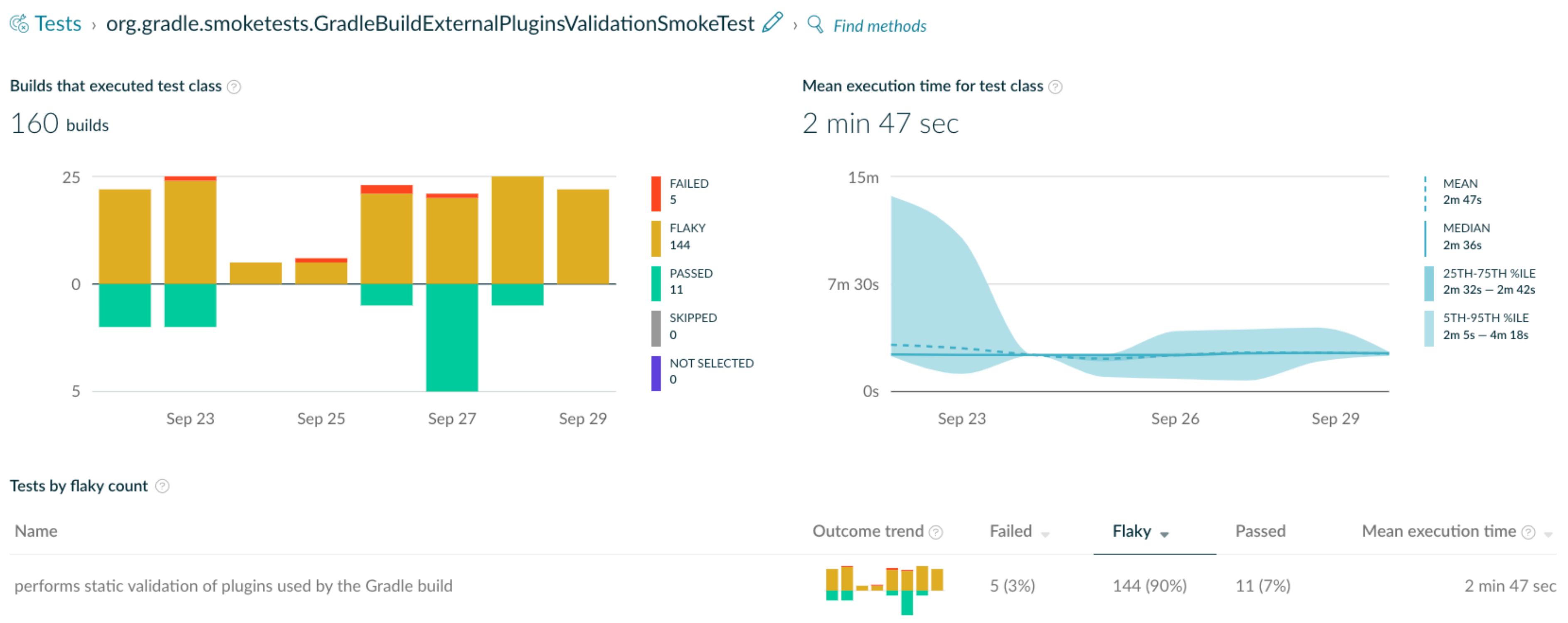
Execution failed for task ':core:embeddedIntegTest'.
> There were failing tests. See the report at: file:///Users/jvandort/work/gradle/subprojects/core/build/reports/tests/embeddedIntegTest/index.html

DPE Organizations Track Failure Rates

Dealing with Flaky Tests

The test is flaky. What do you do now?

a. Try it again
b. Re-run it
c. Re-run it again
d. Ignore it and approve PR
e. All of the above


Tests › org.gradle.smoketests.GradleBuildExternalPluginsValidationSmokeTest › Find methods

Builds that executed test class

160 builds

Mean execution time for test class

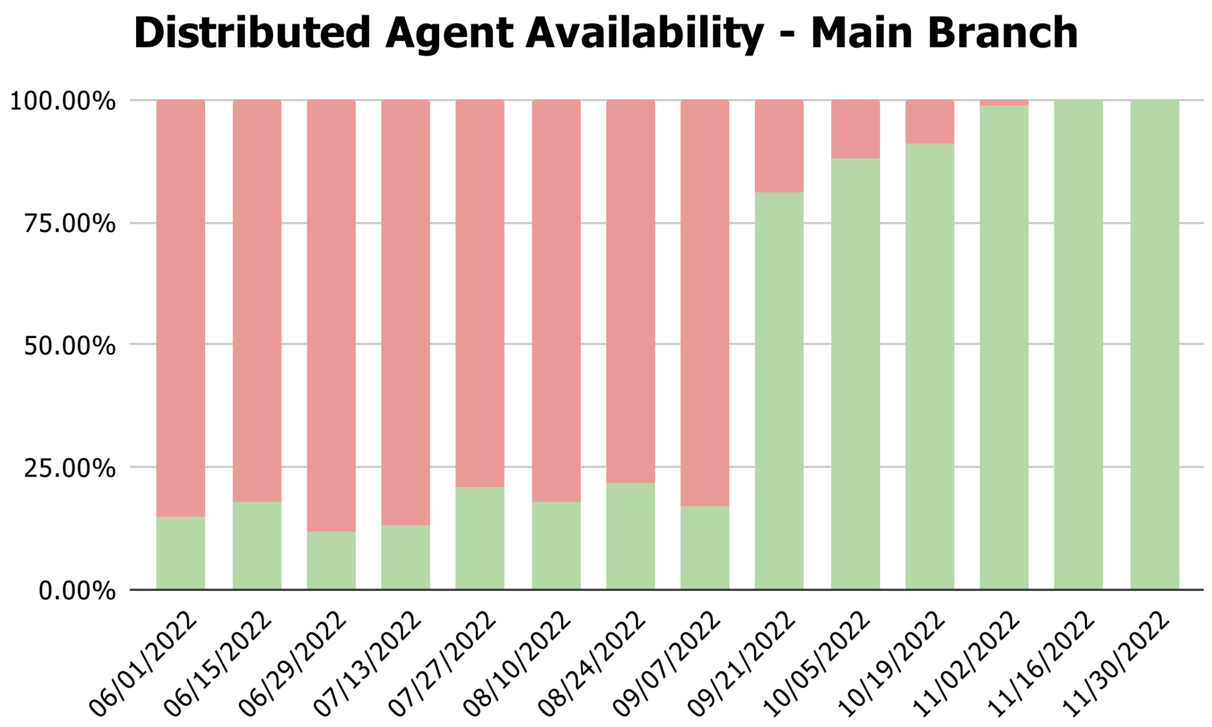
2 min 47 sec

Tests by flaky count

Name	Outcome trend	Failed	Flaky	Passed	Mean execution time
performs static validation of plugins used by the Gradle build		5 (3%)	144 (90%)	11 (7%)	2 min 47 sec

DPE Organizations Analyze Flaky Tests

All Of This Will Improve CI



DPE Will Become Standard Practice

Because the World Should Foster Developer Joy

Questions?

BrianDemers

bdemers

Learn more & get free swag

THANKS!!