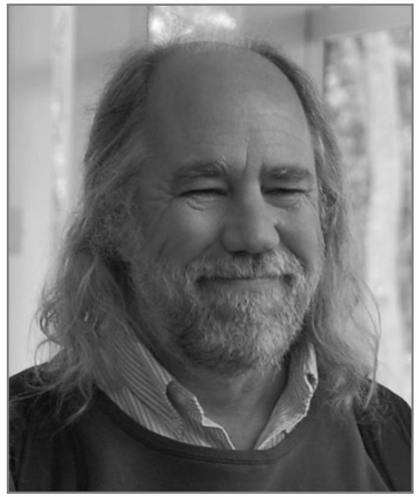
Builds

Yegor Bugayenko

Lecture #21 out of 24
80 minutes

The slidedeck was presented by the author in this YouTube Video



Grady Booch

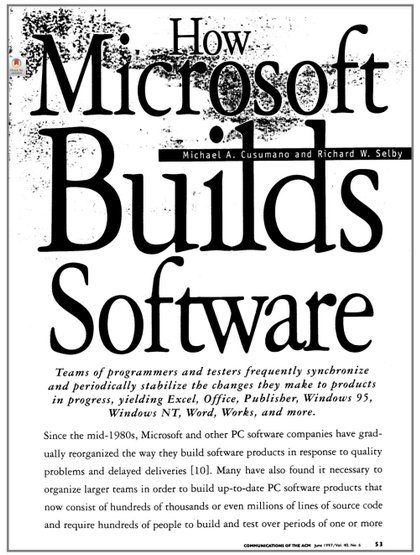
"In general, there may be more internal releases to the development team, with only a few executable releases turned over to external parties. The internal releases represent a sort of continuous integration of the system and exist to force closure on some key system areas."

— Grady Booch, Robert A. Maksimchuk, Michael W. Engle, Bobbi J. Young, Jim Connallen, and Kelli A. Houston. *Object-Oriented Analysis and Design With Applications*. Addison-Wesley, 1994. doi:10.5555/1407387

Michael Cusumano

“Regardless of how often individual developers check in their changes to the source code, a designated developer, called the project build master, generates a complete build of the product on a daily basis using the master version of the source code.”

— Michael A. Cusumano and Richard W. Selby. How Microsoft Builds Software. *Communications of the ACM*, 40(6):53–61, 1997. doi:10.1145/255656.255698



How Microsoft Builds Software

Michael A. Cusumano and Richard W. Selby

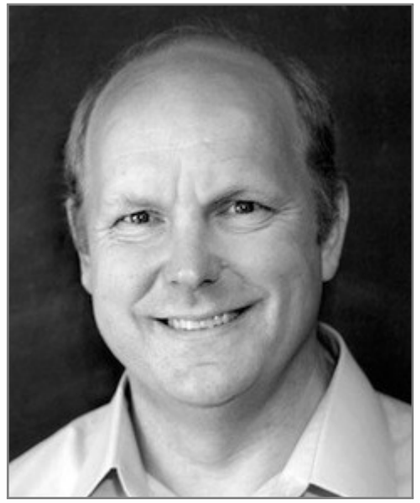
Teams of programmers and testers frequently synchronize and periodically stabilize the changes they make to products in progress, yielding Excel, Office, Publisher, Windows 95, Windows NT, Word, Works, and more.

Since the mid-1980s, Microsoft and other PC software companies have gradually reorganized the way they build software products in response to quality problems and delayed deliveries [10]. Many have also found it necessary to organize larger teams in order to build up-to-date PC software products that now consist of hundreds of thousands or even millions of lines of source code and require hundreds of people to build and test over periods of one or more

COMMUNICATIONS OF THE ACM June 1997/Vol. 40, No. 6 53

"[In Microsoft], the rule is that if developers check in code that 'breaks' the build by preventing it from completing the recompilation, they must fix the defect immediately."

— Michael A. Cusumano and Richard W. Selby. How Microsoft Builds Software. *Communications of the ACM*, 40(6):53–61, 1997. doi:10.1145/255656.255698



Kent Beck

“Developers need freedom to make changes where they make the most sense. Therefore, integrate and test several times a day. Throw away unintegrated code after a couple of days and start over. Ignore code ownership. Program in pairs. Don’t integrate without unit tests.”

— Kent Beck. Extreme Programming: A Humanistic Discipline of Software Development. In *Proceedings of the International Conference on Fundamental Approaches to Software Engineering*, pages 1–6. Springer, 1998. doi:10.1007/bfb0053579

Martin Fowler

"For most projects, the XP guideline of a ten minute build is perfectly within reason. Most of our modern projects achieve this. It's worth putting in concentrated effort to make it happen, because every minute chiseled off the build time is a minute saved for each developer every time they commit."

— Martin Fowler. Continuous Integration. http://martinfowler.com/articles/continuousIntegration.html, 2006. [Online; accessed 07-02-2024]

Jez Humble

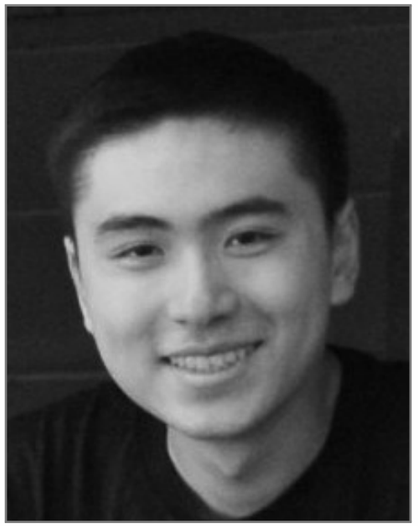
"Automation is the key. It allows all of the common tasks involved in the creation and deployment of software to be performed by developers, testers, and operations personnel, at the push of a button."

— Jez Humble and David Farley. *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation.* Pearson Education, 2010. doi:10.5555/1869904

Bogdan Vasilescu

"Our main finding is that continuous integration improves the productivity of project teams, who can integrate more outside contributions, without an observable diminishment in code quality."

— Bogdan Vasilescu, Yue Yu, Huaimin Wang, Premkumar Devanbu, and Vladimir Filkov. Quality and Productivity Outcomes Relating to Continuous Integration in GitHub. In *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering*, pages 805–816, 2015. doi:10.1145/2786805.2786850



Kai Huang

"Our results show there are good reasons for the rise of CI. Compared to projects that do not use CI, projects that use CI: release twice as often, accept pull requests faster (1.6x), and have developers who are less worried about breaking the build."

— Michael Hilton, Timothy Tunnell, Kai Huang, Darko Marinov, and Danny Dig. Usage, Costs, and Benefits of Continuous Integration in Open-Source Projects. In *Proceedings of the 31st International Conference on Automated Software Engineering*, pages 426–437, 2016. doi:10.1145/2970276.2970358

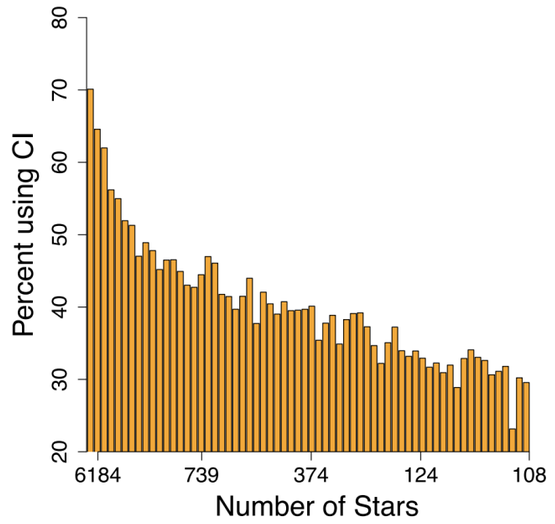


Figure 1: CI usage of projects in GitHub. Projects are sorted by popularity (number of stars).

"In the most popular (starred) group, 70% of projects use CI. As the projects become less popular, the percentage of projects using CI declines to 23%. Observation: Popular projects are more likely to use CI."

Source: Michael Hilton, Timothy Tunnell, Kai Huang, Darko Marinov, and Danny Dig. Usage, Costs, and Benefits of Continuous Integration in Open-Source Projects. In *Proceedings of the 31st International Conference on Automated Software Engineering*, pages 426–437, 2016. doi:10.1145/2970276.2970358

Table 3: CI usage by programing language. For each language, the columns tabulate: the number of projects from our corpus that predominantly use that language, how many of these projects use CI, the percentage of projects that use CI.

Language	Total Projects	# Using CI	Percent CI
Scala	329	221	67.17
Ruby	2721	1758	64.61
Go	1159	702	60.57
PHP	1806	982	54.37
CoffeeScript	343	176	51.31
Clojure	323	152	47.06
Python	3113	1438	46.19
Emacs Lisp	150	67	44.67
JavaScript	8495	3692	43.46
Other	1710	714	41.75
C++	1233	483	39.17
Swift	723	273	37.76
Java	3371	1188	35.24
C	1321	440	33.31
C#	652	188	28.83
Perl	140	38	27.14
Shell	709	185	26.09
HTML	948	241	25.42
CSS	937	194	20.70
Objective-C	2745	561	20.44
VimL	314	59	18.79

"Languages that have the highest CI usage are also dynamically-typed (e.g., Python and JavaScript). One possible explanation may be that in the absence of a static type system which can catch errors early on, these languages use CI to provide extra safety."

Source: Michael Hilton, Timothy Tunnell, Kai Huang, Darko Marinov, and Danny Dig. Usage, Costs, and Benefits of Continuous Integration in Open-Source Projects. In *Proceedings of the 31st International Conference on Automated Software Engineering*, pages 426–437, 2016. doi:10.1145/2970276.2970358

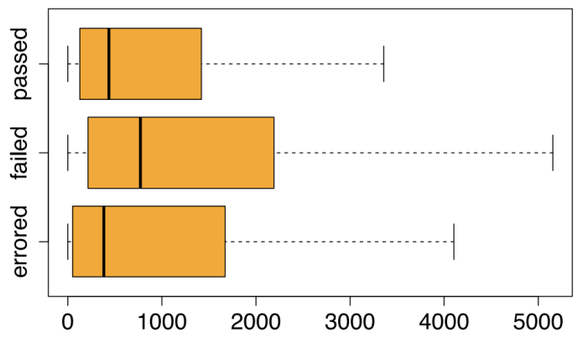


Figure 5: Build time distribution by result, in seconds

"The average build time is just under 500 seconds. Errored builds are those that occur before the build begins (e.g., when a dependency cannot be downloaded), and failed builds are those that the build is not completed successfully."

Source: Michael Hilton, Timothy Tunnell, Kai Huang, Darko Marinov, and Danny Dig. Usage, Costs, and Benefits of Continuous Integration in Open-Source Projects. In *Proceedings of the 31st International Conference on Automated Software Engineering*, pages 426–437, 2016. doi:10.1145/2970276.2970358

Michael Hilton

"Developers use CI to guarantee quality, consistency, and viability across different environments. However, adding and maintaining automated tests causes these benefits to come at the expense of increased time and effort."

— Michael Hilton, Nicholas Nelson, Timothy Tunnell, Darko Marinov, and Danny Dig. Trade-Offs in Continuous Integration: Assurance, Security, and Flexibility. In *Proceedings of the 11th Joint Meeting on Foundations of Software Engineering*, pages 197–207, 2017. doi:10.1145/3106237.3106270

Table 2: Barriers developers encounter when using CI

Barrier	Broad	Focused
B1 Troubleshooting a CI build failure	50%	64%
B2 Overly long build times	38%	50%
B3 Automating the build process	34%	26%
B4 Lack of support for the desired workflow	31%	42%
B5 Setting up a CI server or service	27%	29%
B6 Maintaining a CI server or service	27%	40%
B7 Lack of tool integration	26%	12%
B8 Security and access controls	21%	14%

"When a CI build fails, some participants begin the process of identifying why the build failed. Sometimes, this can be fairly straightforward..."

Source: Michael Hilton, Nicholas Nelson, Timothy Tunnell, Darko Marinov, and Danny Dig. Trade-Offs in Continuous Integration: Assurance, Security, and Flexibility. In *Proceedings of the 11th Joint Meeting on Foundations of Software Engineering*, pages 197–207, 2017. doi:10.1145/3106237.3106270

Figure 1: Maximum acceptable build time (minutes)

"[Fowler, 2006] suggests most projects should try to follow the XP guideline of a 10-minute build. When we asked our 523 participants what is the maximum acceptable time for a CI build to take, the most common answer was also 10 minutes."

Source: Michael Hilton, Nicholas Nelson, Timothy Tunnell, Darko Marinov, and Danny Dig. Trade-Offs in Continuous Integration: Assurance, Security, and Flexibility. In *Proceedings of the 11th Joint Meeting on Foundations of Software Engineering*, pages 197–207, 2017. doi:10.1145/3106237.3106270

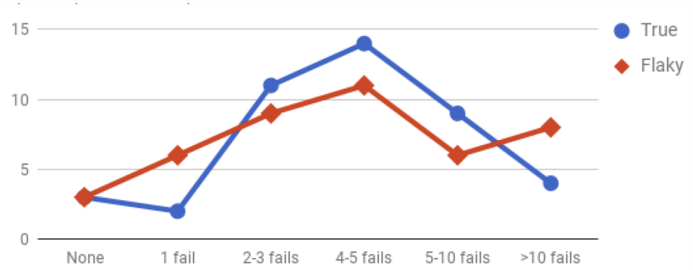


Figure 6: Flaky vs True test failures reported by Pivotal developers (N=42)

"Pivotal developers experienced similar numbers of flaky and true CI failures per week. However, for the largest category, >10 fails a week, there were twice as many flaky failures as true failures."

Source: Michael Hilton, Nicholas Nelson, Timothy Tunnell, Darko Marinov, and Danny Dig. Trade-Offs in Continuous Integration: Assurance, Security, and Flexibility. In *Proceedings of the 11th Joint Meeting on Foundations of Software Engineering*, pages 197–207, 2017. doi:10.1145/3106237.3106270

John Micco

"Unfortunately, across our entire corpus of tests, we [in Google] see a continual rate of about 1.5% of all test runs reporting a 'flaky' result."

— John Micco. Flaky Tests at Google and How We Mitigate Them. https://testing.googleblog.com/2016/05/flaky-tests-at-google-and-how-we.html, may 2016. [Online; accessed 25-02-2024]

Google Thoughts About Flaky Tests

- Almost 16% of our 4.2M tests have some level of flakiness
- 84% of transitions from Pass -> Fail are from ‘flaky’ tests
- We spend up to 16% of our compute resources re-running flaky tests
- Certain people/automation more likely to cause breakages (oops!)
- Certain languages more likely to cause breakages (sorry)

Source: John Micco. The State of Continuous Integration Testing at Google. *ICST*, 2017

Carmine Vassallo

"In total, the 349 projects underwent 116,741 builds, of which 30,792 (26%) failed. It is interesting to notice how the percentage of build failures is approximately the same in OSS and ING."

— Carmine Vassallo, Gerald Schermann, Fiorella Zampetti, Daniele Romano, Philipp Leitner, Andy Zaidman, Massimiliano Di Penta, and Sebastiano Panichella. A Tale of CI Build Failures: An Open Source and a Financial Organization Perspective. In *Proceedings of the International Conference on Software Maintenance and Evolution (ICSME)*, pages 183–193. IEEE, 2017. doi:10.1109/ICSME.2017.67

Thomas Rausch

"Process metrics have a significant impact on the build outcome in 8 of the 14 projects on average, but the strongest influencing factor across all projects is overall stability in the recent build history. For 10 projects, more than 50% (up to 80%) of all failed builds follow a previous build failure."

— Thomas Rausch, Waldemar Hummer, Philipp Leitner, and Stefan Schulte. An Empirical Analysis of Build Failures in the Continuous Integration Workflows of Java-Based Open-Source Software. In *Proceedings of the 14th International Conference on Mining Software Repositories*, pages 345–355. IEEE, 2017. doi:10.1109/msr.2017.54

TABLE I
NAME, DESCRIPTION AND METADATA OF PROJECTS USED AS RESEARCH SUBJECTS

Name	Description	Age VCS*	Commits	Committers	Age CI*	Builds	Build freq.*	Fail ratio
Apache Storm	Distributed computation framework	1961	11656	253	647	5472	8.5	0.69
Crate.IO	Scalable SQL database	1390	13722	69	1024	21864	21.4	0.63
JabRef	Application for managing BibTeX databases	1407	16851	112	1046	9615	9.2	0.29
Butterknife	Android dependency injection library	1426	894	114	1426	1220	0.9	0.34
jcabi-github	Object-oriented wrapper of Github API	1179	2543	60	1024	1316	1.3	0.45
Hystrix	Fault tolerance library for distributed systems	1532	2321	103	880	1228	1.4	0.48
Openmicroscopy	Microscopy data environment	4364	55571	59	1501	16383	10.9	0.19
Presto	Distributed SQL query engine for big data	1635	23500	308	1180	19112	16.2	0.49
RxAndroid	RxJava bindings for Android	1264	495	74	880	728	0.8	0.16
SpongeAPI	Minecraft plugin API	874	3692	213	874	8835	10.1	0.24
Spring Boot	Java application framework	1561	11300	566	1276	10051	7.9	0.28
Square OkHttp	HTTP+HTTP/2 client for Android and Java	1651	3832	263	1576	7439	4.7	0.49
Square Retrofit	HTTP client for Android and Java	2336	1640	231	1576	3040	1.9	0.20
WordPress-Android	WordPress for Android	2697	21548	66	1191	15025	12.6	0.14

* in days

Source: Thomas Rausch, Waldemar Hummer, Philipp Leitner, and Stefan Schulte. An Empirical Analysis of Build Failures in the Continuous Integration Workflows of Java-Based Open-Source Software. In *Proceedings of the 14th International Conference on Mining Software Repositories*, pages 345–355. IEEE, 2017. doi:10.1109/msr.2017.54

TABLE II
DESCRIPTION AND FREQUENCY OF ERROR CATEGORIES

Label	Projects	Description
testfailure	12	An automated test did not pass
compile	12	Compilation error
git	12	VCS interaction error, e.g., worker fails to fetch code
buildconfig	11	Faulty build config, e.g., syntax error in pom.xml
crash	11	Build environment crashed or exceeded time limit
dependency	11	Dependency error, e.g., invalid version number
quality	10	Coding-rule violation during code inspection
unknown	9	Errors without a clearly identifiable cause
itestfailure	4	An automated integration test failed
doc	3	Documentation issue, e.g., undocumented methods
license	3	License criteria not met, e.g., missing license headers
compatibility	2	API incompatibility, e.g., due to version conflict
androidsdk	1	Android SDK-related error, e.g., download failed
buildout	1	Specific build error of Python submodule in Crate.IO

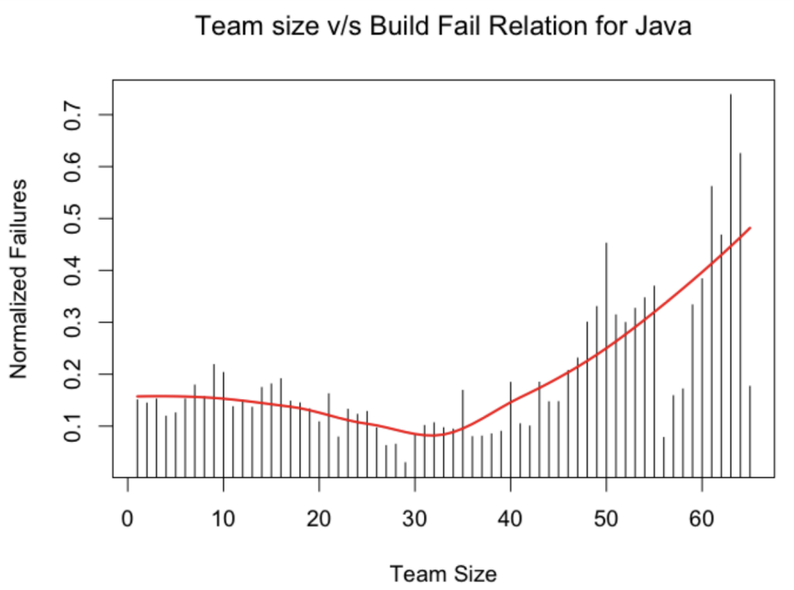
Source: Thomas Rausch, Waldemar Hummer, Philipp Leitner, and Stefan Schulte. An Empirical Analysis of Build Failures in the Continuous Integration Workflows of Java-Based Open-Source Software. In *Proceedings of the 14th International Conference on Mining Software Repositories*, pages 345–355. IEEE, 2017. doi:10.1109/msr.2017.54

"On average, 41% of builds fail because of test failures... On average, 30% of errors occur in the first half of the build runtime. The later half is dominated by 70% testfailures. Together with a build-retry approach, testfailures can cause long delays in the feedback loop."

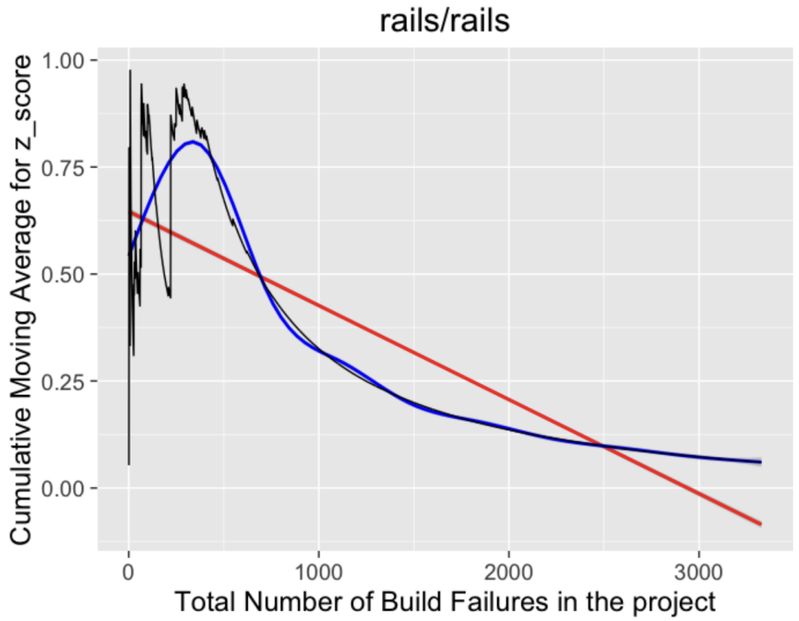
Romit Jain

"We observed that team size generally increases the number of build failures. The interesting observation here is the minima after which there is a positive curve."

— Romit Jain, Saket Kumar Singh, and Bharavi Mishra. A Brief Study on Build Failures in Continuous Integration: Causation and Effect. In *Proceedings of the Progress in Advanced Computing and Intelligent Engineering*, pages 17–27. Springer, 2018. doi:10.1007/978-981-13-0224-4_2



Source: Romit Jain, Saket Kumar Singh, and Bharavi Mishra. A Brief Study on Build Failures in Continuous Integration: Causation and Effect. In *Proceedings of the Progress in Advanced Computing and Intelligent Engineering*, pages 17–27. Springer, 2018. doi:10.1007/978-981-13-0224-4_2



Source: Romit Jain, Saket Kumar Singh, and Bharavi Mishra. A Brief Study on Build Failures in Continuous Integration: Causation and Effect. In *Proceedings of the Progress in Advanced Computing and Intelligent Engineering*, pages 17–27. Springer, 2018. doi:10.1007/978-981-13-0224-4_2

> On October 16, 2018,
> GitHub launched “Actions”

Mehdi Golzadeh

> “Together with Travis, GHA covers more than 80% of all usages. Moreover, in only 18 months GHA has overtaken all other CIs in popularity.”

— Mehdi Golzadeh, Alexandre Decan, and Tom Mens. On the Rise and Fall of CI Services in GitHub. In *Proceedings of the International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 662–672. IEEE, 2022. doi:10.1109/SANER53432.2022.00084

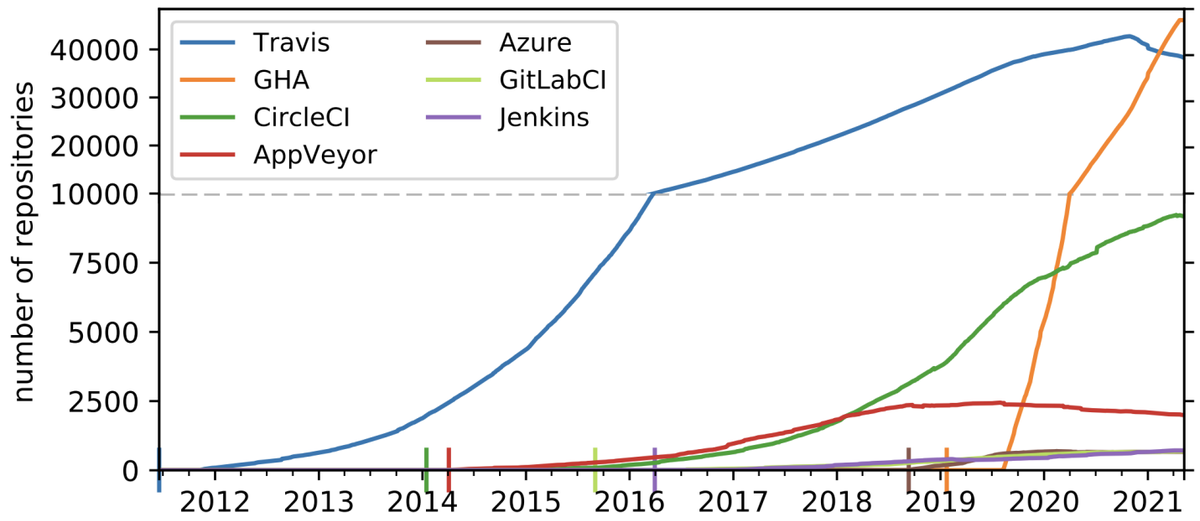


Fig. 2: Number of repositories using a specific CI.

Source: Mehdi Golzadeh, Alexandre Decan, and Tom Mens. On the Rise and Fall of CI Services in GitHub. In *Proceedings of the International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 662–672. IEEE, 2022. doi:10.1109/SANER53432.2022.00084

Wing Lam

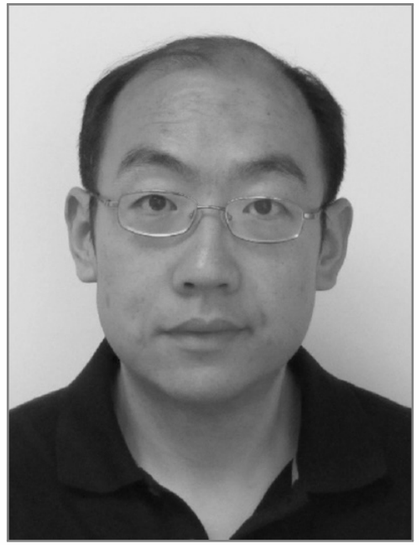
"We find that 53% of flaky tests detected in CI runs are not detected in isolation."

— Wing Lam, Stefan Winter, Angello Astorga, Victoria Stodden, and Darko Marinov. Understanding Reproducibility and Characteristics of Flaky Tests Through Test Reruns in Java Projects. In *Proceedings of the 31st International Symposium on Software Reliability Engineering (ISSRE)*, pages 403–413. IEEE, 2020. doi:10.1109/issre5003.2020.00045

Owain Parry

"The causes and associated factors of flaky tests:
1. Asynchronicity and Concurrency, 2. Platform Dependencies, 3. Floating-Point numbers,
4. Order-dependent tests, 5. Unordered Collections,
6. Shared access to static fields, 7. Timeouts, 8. I/O and Network, 9. Algorithmic Non-determinism."

— Owain Parry, Gregory M. Kapfhammer, Michael Hilton, and Phil McMinn. A Survey of Flaky Tests. *ACM Transactions on Software Engineering and Methodology*, 31(1):1–74, 2021. doi:10.1145/3476105



Pei Liu

"We start by collecting a set of 84,475 open-source Android apps from the most popular three online code hosting sites, namely Github, GitLab, and Bitbucket. We then look into those apps and find that only around 10% of apps have leveraged CI/CD services, i.e., the majority of open-source Android apps are developed without accessing CI/CD services."

— Pei Liu, Xiaoyu Sun, Yanjie Zhao, Yonghui Liu, John Grundy, and Li Li. A First Look at CI/CD Adoptions in Open-Source Android Apps. In *Proceedings of the 37th International Conference on Automated Software Engineering*, pages 1–6, 2022. doi:10.1145/3551349.3561341

Shanto Rahman

"FlakeSync works by identifying a critical point, representing some key part of code that must be executed early w.r.t. other concurrently executing code, and a barrier point, representing the part of code that should wait until the critical point has been executed. FlakeSync can modify code to check when the critical point is executed and have the barrier point keep waiting until the critical point has been executed, essentially synchronizing these two parts of code for the specific test execution."

— Shanto Rahman and August Shi. FlakeSync: Automatically Repairing Async Flaky Tests. In *Proceedings of the 46th International Conference on Software Engineering (ICSE)*, page 920. IEEE, 2024

References

Kent Beck. Extreme Programming: A Humanistic Discipline of Software Development. In *Proceedings of the International Conference on Fundamental Approaches to Software Engineering*, pages 1–6. Springer, 1998. doi:10.1007/bfb0053579.

Grady Booch, Robert A. Maksimchuk, Michael W. Engle, Bobbi J. Young, Jim Connallen, and Kelli A. Houston. *Object-Oriented Analysis and Design With Applications*. Addison-Wesley, 1994. doi:10.5555/1407387.

Michael A. Cusumano and Richard W. Selby. How Microsoft Builds Software. *Communications of the ACM*, 40(6):53–61, 1997. doi:10.1145/255656.255698.

Martin Fowler. Continuous Integration. `http://martinfowler.com/articles/continuousIntegration.html`, 2006. [Online; accessed 07-02-2024].

Mehdi Golzadeh, Alexandre Decan, and Tom Mens. On the Rise and Fall of CI Services in GitHub. In *Proceedings of the International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 662–672. IEEE, 2022. doi:10.1109/SANER53432.2022.00084.

Michael Hilton, Timothy Tunnell, Kai Huang, Darko Marinov, and Danny Dig. Usage, Costs, and Benefits of Continuous Integration in Open-Source Projects. In *Proceedings of the 31st International Conference on Automated Software Engineering*, pages 426–437, 2016. doi:10.1145/2970276.2970358.

Michael Hilton, Nicholas Nelson, Timothy Tunnell, Darko Marinov, and Danny Dig. Trade-Offs in Continuous Integration: Assurance, Security, and Flexibility. In *Proceedings of the 11th Joint Meeting on Foundations of Software Engineering*, pages 197–207, 2017. doi:10.1145/3106237.3106270.

Jez Humble and David Farley. *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation*. Pearson Education, 2010. doi:10.5555/1869904.

Romit Jain, Saket Kumar Singh, and Bharavi Mishra. A Brief Study on Build Failures in Continuous Integration: Causation and Effect. In *Proceedings of the Progress in Advanced Computing and Intelligent Engineering*, pages 17–27. Springer, 2018. doi:10.1007/978-981-13-0224-4_2.

Wing Lam, Stefan Winter, Angello Astorga, Victoria Stodden, and Darko Marinov. Understanding Reproducibility and Characteristics of Flaky Tests Through Test Reruns in Java Projects. In *Proceedings of the 31st International Symposium on Software Reliability Engineering (ISSRE)*, pages 403–413. IEEE, 2020. doi:10.1109/issre5003.2020.00045.

Pei Liu, Xiaoyu Sun, Yanjie Zhao, Yonghui Liu, John Grundy, and Li Li. A First Look at CI/CD Adoptions in Open-Source Android Apps. In *Proceedings of the 37th International Conference on Automated Software Engineering*, pages 1–6, 2022. doi:10.1145/3551349.3561341.

John Micco. Flaky Tests at Google and How We Mitigate Them. `https://testing.googleblog.com/2016/05/flaky-tests-at-google-and-how-we.html`, may 2016. [Online; accessed 25-02-2024].

John Micco. The State of Continuous Integration Testing at Google. *ICST*, 2017.

Owain Parry, Gregory M. Kapfhammer, Michael Hilton, and Phil McMinn. A Survey of Flaky Tests. *ACM Transactions on Software Engineering and Methodology*, 31(1):1–74, 2021. doi:10.1145/3476105.

Shanto Rahman and August Shi. FlakeSync: Automatically Repairing Async Flaky Tests. In *Proceedings of the 46th International Conference on Software Engineering (ICSE)*, page 920. IEEE, 2024.

Thomas Rausch, Waldemar Hummer, Philipp Leitner, and Stefan Schulte. An Empirical Analysis of Build Failures in the Continuous Integration Workflows of Java-Based Open-Source Software. In *Proceedings of the 14th International Conference on Mining Software Repositories*, pages 345–355. IEEE, 2017. doi:10.1109/msr.2017.54.

Bogdan Vasilescu, Yue Yu, Huaimin Wang, Premkumar Devanbu, and Vladimir Filkov. Quality and Productivity Outcomes Relating to Continuous Integration in GitHub. In *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering*, pages 805–816, 2015. doi:10.1145/2786805.2786850.

Carmine Vassallo, Gerald Schermann, Fiorella Zampetti, Daniele Romano, Philipp Leitner, Andy Zaidman, Massimiliano Di Penta, and Sebastiano Panichella. A Tale of CI Build Failures: An Open Source and a Financial Organization Perspective. In *Proceedings of the International Conference on Software Maintenance and Evolution (ICSME)*, pages 183–193. IEEE, 2017. doi:10.1109/ICSME.2017.67.