Surveying the Developer Experience of Flaky Tests

Owain Parry[1], Gregory M. Kapfhammer[2], Michael Hilton[3], Phil McMinn[1]

[1]University of Sheffield, UK
[2]Allegheny College, USA
[3]Carnegie Mellon University, USA

What is a flaky test?

- A *flaky test* is a test case that can **pass and fail without changes** to the code under test.
- Flaky tests reduce developer productivity and lead to a loss of confidence in testing.

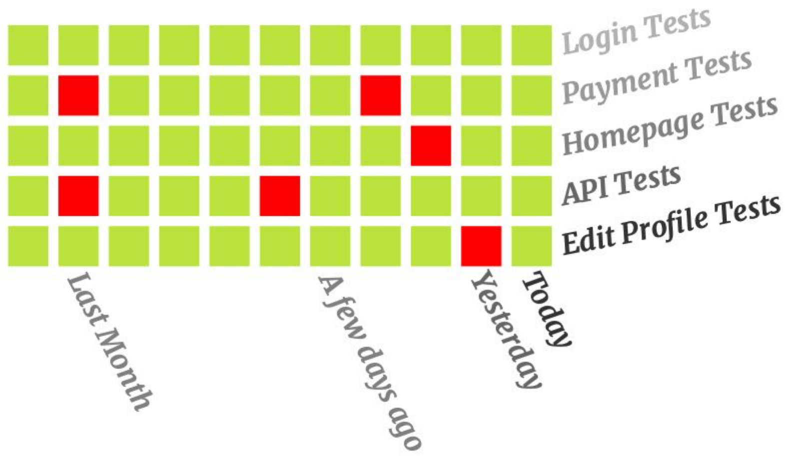


Image courtesy of Brian Graham
https://statagroup.com/articles/flaky-tests

What has been done about flaky tests?

- The past decade has seen an increasing volume of empirical studies on flaky tests *[Luo et. al. 2014]*, *[Throve et. al. 2018]*, *[Romano et. al. 2021]*.
- But there is less focus on the views and experiences of software developers.
- Where previous such studies exist, they focus on specific organizations or self-reported experiences *[Hilton et. al. 2017]*, *[Eck et. al. 2019]*.

What did we do?

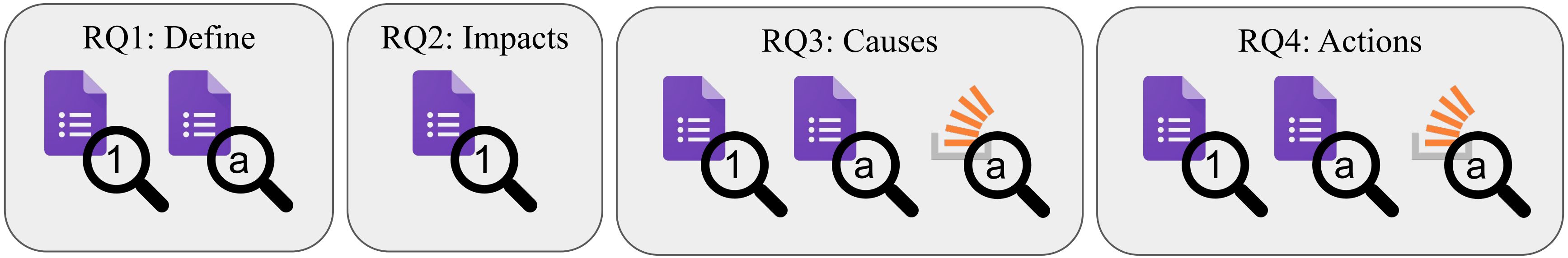
- We set out to learn how developers define and react to flaky tests and to understand developers' experiences of their impacts and causes.
- We deployed a survey on social media and received 170 responses.
- We also analyzed 38 StackOverflow threads about flaky tests.

Our research questions

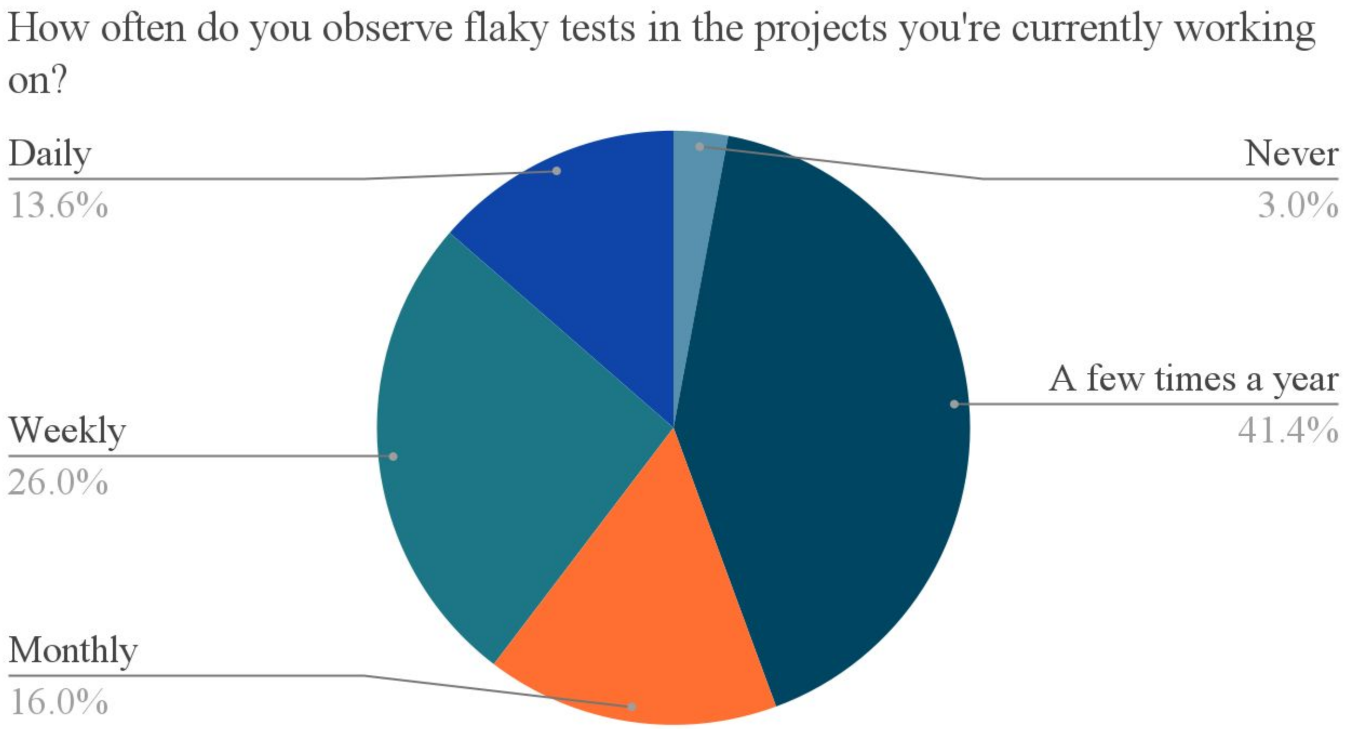
- RQ1: How do developers **define** flaky tests?
- RQ2: What **impacts** do flaky tests have on developers?
- RQ3: What **causes** the flaky tests experienced by developers?
- RQ4: What **actions** do developers take against flaky tests?

Our methodology

- We reviewed published papers and grey literature to design a **survey** of 11 open- and closed-ended questions.
- We collected a dataset of **StackOverflow threads** where a developer asked for help addressing one or more flaky tests and accepted an answer.
- We performed **numerical analysis** on the closed-ended survey questions and **thematic analysis** on the open-ended questions and the StackOverflow threads.

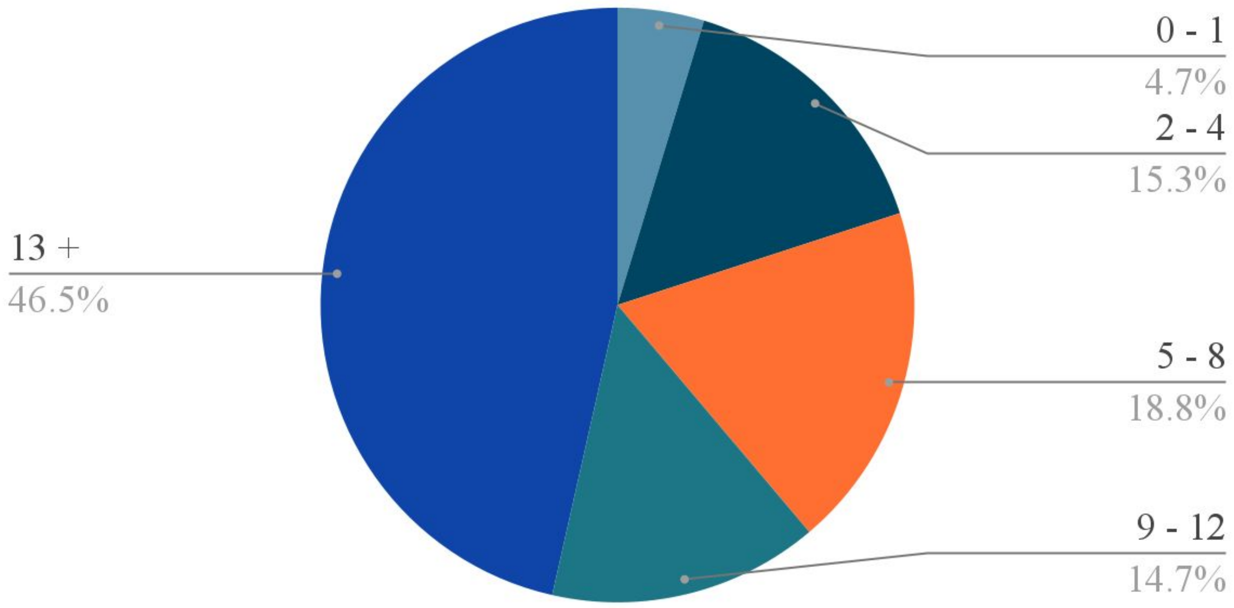


A little bit about our survey respondents

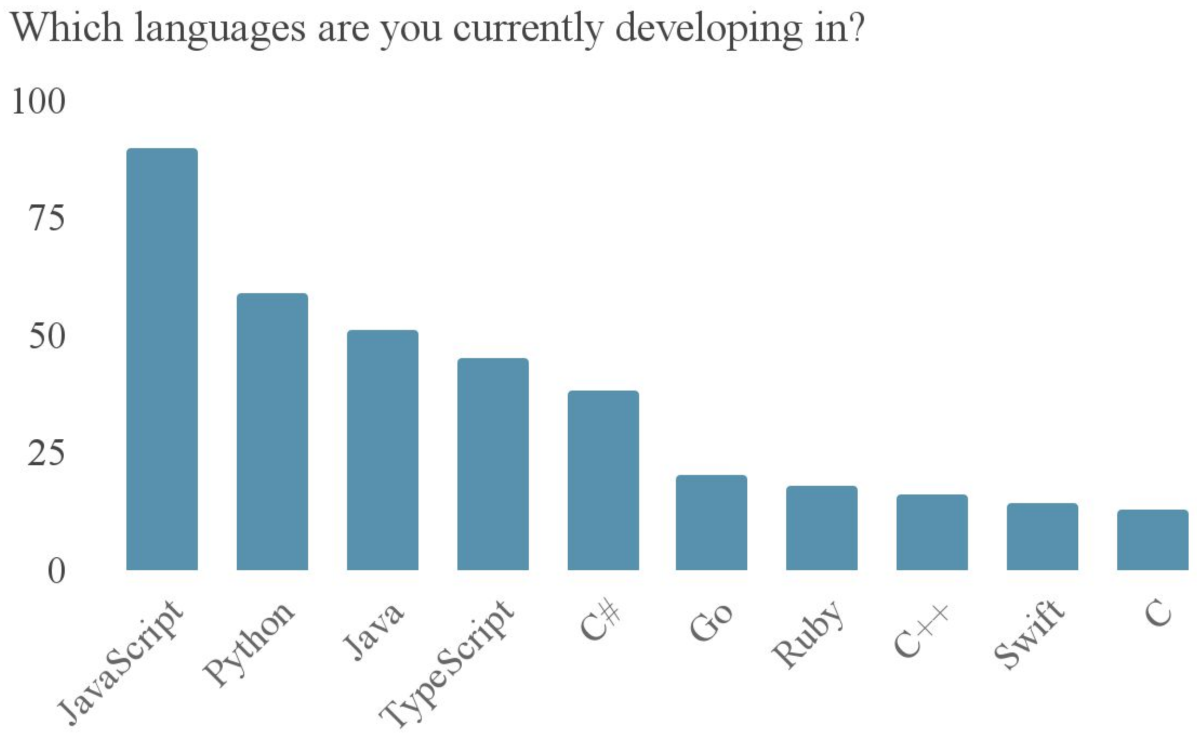


A little bit about our survey respondents

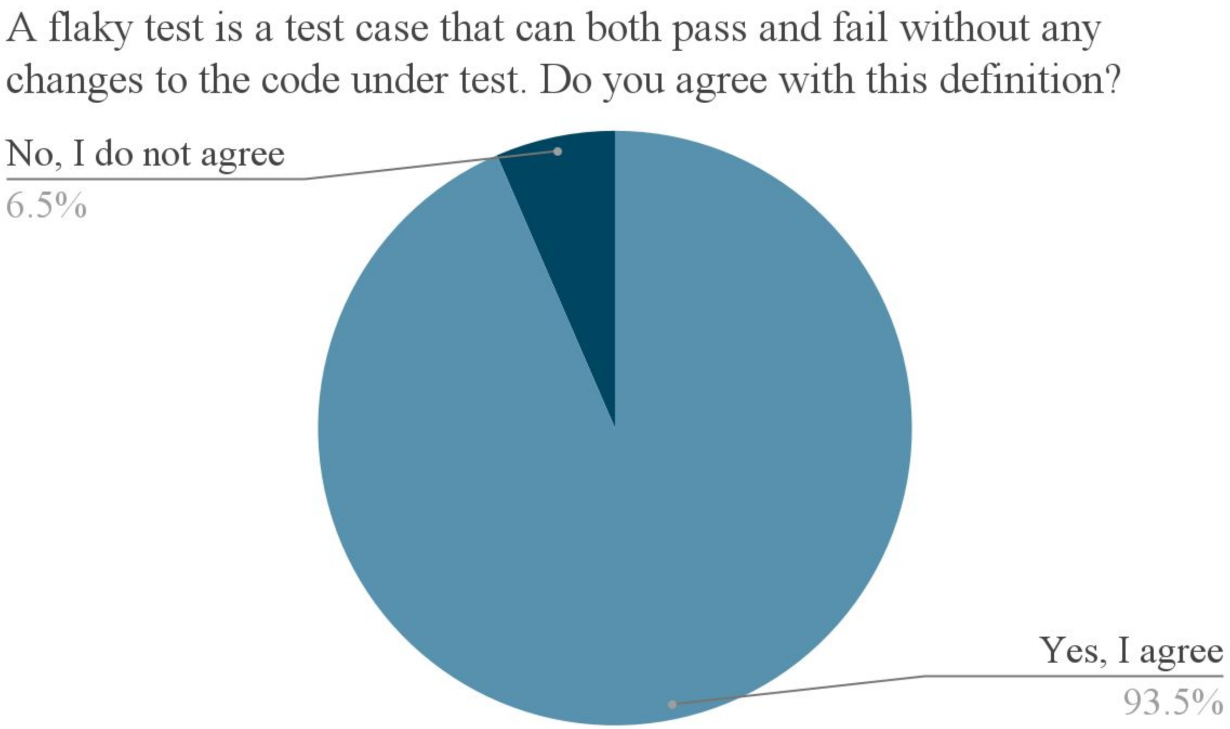
How many years experience do you have in commercial and/or open-source software development?



A little bit about our survey respondents



RQ1: How do developers define flaky tests?



RQ1: How do developers define flaky tests?

- **Beyond code**: The definition extends beyond the test case code and the code that it covers. *“... a flaky test is any test that changes from pass to fail (or vice versa) in different environments”* - P97.
- **Flaky code under test**: A flaky test can indicate that the code under test is flawed, rather than the test case itself. *“... a flaky test is therefore either unreliable itself or it proves the code under test is flawed and unreliable”* - P155.
- **Beyond test outcomes**: A test case can be considered flaky despite having a consistent outcome. *“... this includes pass/fail, but can encompass other aspects such as coverage or test time”* - P58.

RQ2: What impacts do flaky tests have on developers?

To what extent do you agree with the following statements…
(Strongly disagree: 0, Disagree: 1, Agree: 2, Strongly agree: 3)

		Score	*Rank*
Flaky tests reduce the **reliability** of testing.		2.45	4
Flaky tests reduce the **efficiency** of testing.	★	**2.47**	**3**
Flaky tests lead to a loss of **productivity**.	★	**2.50**	**2**
Flaky tests lead to a loss of **confidence** in testing.		2.21	5
Flaky tests hinder **continuous integration** (CI).	★	**2.63**	**1**
Flaky tests make it more likely for you to **ignore** (potentially genuine) test failures.		2.16	6
It is difficult to **reproduce** a flaky test failure.		2.09	7
It is difficult to **differentiate** between a test failure due to a genuine bug and a test failure due to flakiness.		1.76	8

RQ3: What causes the flaky tests experienced by developers?

In the projects you're currently working on, how often have you encountered flaky tests caused by…
(Never: 0, Rarely: 1, Sometimes: 2, Often: 3)

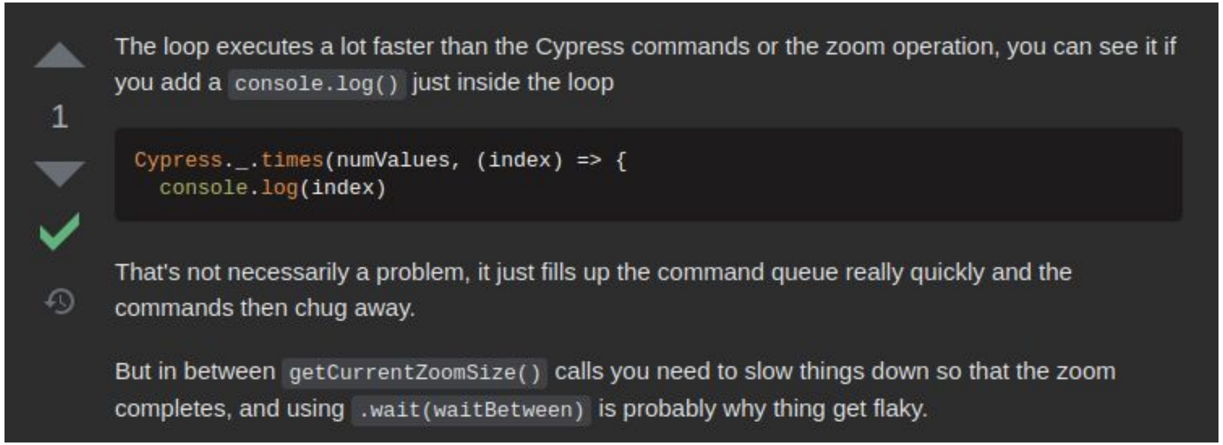
		Score	*Rank*
Not correctly **waiting** for the results of asynchronous calls to become available.		1.30	4
Synchronization issues between **multiple threads** interacting in an unsafe or unanticipated manner.		1.12	5
Tests not properly **cleaning up** after themselves or failing to **set up** their necessary preconditions.	★	**1.69**	**1**
Improper management of **resources** (e.g., not closing a file or not deallocating memory).		0.89	7
Dependency on a **network** connection.	★	**1.44**	**2**
Not accounting for all the possible outcomes of **random** data generators or code that uses them.		0.69	9
Reliance on the local system **time/date**.		1.06	6
Inaccuracies when performing **floating point** operations.		0.48	10
Assuming a particular iteration order for an **unordered collection**-type object (e.g., sets).		0.73	8
Reasons that cannot be precisely determined.	★	**1.32**	**3**

RQ3: What causes the flaky tests experienced by developers?

- **External artifact**: An issue in an external service, library, or other artifact, that is outside the scope and control of the software under test. *"Third-party artifacts, services, or dependencies … which you do not have full control of."* - P8.
- **Environmental differences**: Environmental differences between local development machines and remote build machines. *"Environmental differences in local vs CI like different JVM defaults."* - P21.
- **Host system issues**: Problems regarding the machines running the test suites. "Changes in hardware that the code and tests are running on." - P155.

RQ3: What causes the flaky tests experienced by developers?

- **UI timing**: Test case does not wait for a user interface to be in the correct state.
- **Logic error**: Error in the logic of the test code or the code under test.
- **Shared state**: Test case depends on state shared with other test cases.



1

The loop executes a lot faster than the Cypress commands or the zoom operation, you can see it if you add a `console.log()` just inside the loop

```
Cypress._.times(numValues, (index) => {
  console.log(index)
```

That's not necessarily a problem, it just fills up the command queue really quickly and the commands then chug away.

But in between `getCurrentZoomSize()` calls you need to slow things down so that the zoom completes, and using `.wait(waitBetween)` is probably why thing get flaky.

https://stackoverflow.com/questions/67375506

RQ4: What actions do developers take against flaky tests?

After identifying a flaky test, how often do you…
(Never: 0, Rarely: 1, Sometimes: 2, Often: 3)

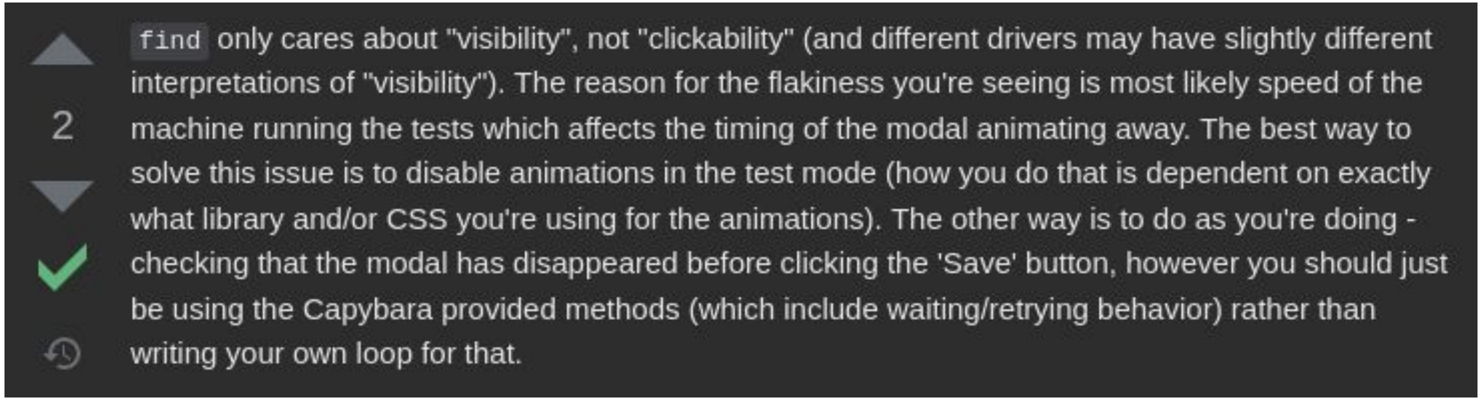
		Score	*Rank*
Take **no action**.		1.19	4
Re-run the build.	★	**2.67**	**1**
Document and defer (e.g., submit an issue/bug report).	★	**1.62**	**3**
Delete the test.		0.94	5
Quarantine the test.		0.77	8
Mark the test to be **skipped** or as an expected failure (e.g., xfail).		0.93	6
Mark the test to be automatically **repeated** (e.g., by using the flaky plugin for pytest).		0.79	7
Attempt to **repair** the flakiness.	★	**2.41**	**2**

RQ4: What actions do developers take against flaky tests?

- **Emotive response**: An expression of anger or some other emotion. *"Get very angry."* - P34.
- **Alert proper person**: Inform other member or members of the development team about the flaky test. *"Tell the person who maintains that codebase."* - P52.
- **Reorder tests**: Adjust the order of the test cases. *"Reorder tests in case they are order-dependent."* - P111.

RQ4: What actions do developers take against flaky tests?

- **Fix logic**: Repair a logic error.
- **Wait for condition**: Add an explicit wait for a condition.
- **Add mock**: Mock out an object or method.


2

`find` only cares about "visibility", not "clickability" (and different drivers may have slightly different interpretations of "visibility"). The reason for the flakiness you're seeing is most likely speed of the machine running the tests which affects the timing of the modal animating away. The best way to solve this issue is to disable animations in the test mode (how you do that is dependent on exactly what library and/or CSS you're using for the animations). The other way is to do as you're doing - checking that the modal has disappeared before clicking the 'Save' button, however you should just be using the Capybara provided methods (which include waiting/retrying behavior) rather than writing your own loop for that.

https://stackoverflow.com/questions/48027118

Anything else you'd like to tell us?

- **Developer culture**: The relationship between flaky tests and testing practices and developer culture. *"It's often and organizational problem …"* - P89.
- **Emotive response**: An expression of anger or other emotion. *"They suck."* - P91.
- **Poor tooling support**: Tooling for handling flaky tests is inadequate or not well known. *"Library support for automatically handling them in Scala is poor or not well popularized."* - P7.

Recommendations

- **Consider beyond code**: The definition of a flaky test should include factors beyond the test case code or the code under test, such as properties of the execution environment.
- **Not completely useless**: Flaky tests may indicate a flaw in the code under test or another aspect of the software system. Therefore, developers should not write them off as completely useless.
- **Impact on CI**: Flaky tests can become an obstacle to the effective deployment of CI. Researchers should consider the creation and evaluation of new approaches to better mitigate this trend.

Recommendations

- **Careful setup/teardown**: Insufficient setup and teardown is a common cause of flaky tests. Developers should exercise particular care when writing setup and teardown methods for their test suites.
- **Identify root causes**: It is difficult to manually determine the root cause of many flaky tests. Researchers should continue to develop automated techniques for this challenging task.
- **Repair promptly**: Developers should to repair flaky tests as soon as possible after identifying them to avoid them accumulating and potentially being ignored.