Test doubles

and why they are important

Why Test Doubles?

Some benefits:

- Allows us to isolate the system we are testing from other parts of the system so that we can investigate the behaviour of the system under test
- Increases speed of test
- Is always available
- Avoids undesirable side effects of interfacing dependant systems
- Cheap and easy to implement
- Gives us access to data or states that otherwise might not be available

“Test doubles are indispensable tools for testing across architectural boundaries. Mocking is a good practice in general.”

Robert C Martin

What are Test Doubles?

Functions or code created with the purpose of verifying behaviour of production code. Test doubles are used to limit the system under test so that specific behaviour may be verified.

Stubs

- Responds to calls with pre-programmed output
- Must be set up for each test

Fakes

- Handmade Stubs

Mocks

- Enables verification of calls they are expected to receive.
- Provides a way of verifying behaviour

Spies

- Handmade Mocks

Commands and Queries

Two categories of abstractions with two types of methods:

- **Command** methods changes the state of the system, but does not return or report the state

- **Query** methods tells us about the state of the system - without modifying it.

Stubs vs Mocks

While a Stub simulates real objects with the minimum methods needed for a test, Mock objects are used to check if the correct techniques and paths are applied to the objects

We use Stubs for queries:

- A **query** just returns data and should not affect or change the system under test
- A **stub** returns some pre-programmed output when called in the expected way

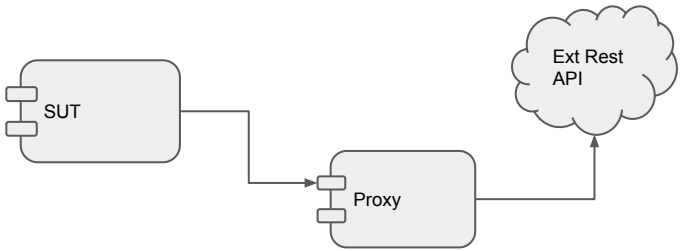
We use Mocks for commands:

- A **command** is a method call that changes the state of the system under test
- A **mock** is used to confirm that a command has been triggered as expected

Guidelines (1)

Use Test Doubles for classes that you own:

- Implement an abstraction layer (proxy) between your System under test (SUT) and external services



Verify as little as possible:

- Specification should be as precise as possible, but not more precise

Tests that verifies too much tends to check implementation details rather than behaviour. This causes the test to be fragile and is more likely to fail for unrelated changes

Guidelines (2)

Do not use Test Doubles for isolated objects:

- Objects that has no collaborators with other objects in the system should not be tested using Mocks or Stubs.
- Aggregates, entities and domain model in general are typically object without collaborators.

Don’t add behavior inside Test Doubles:

- Mock objects should not add any additional complexity to the test environment
- Behaviour should be obvious and self explanatory
- Behaviour added to Mock objects might be a symptom of misplaced responsibilities

Guidelines (3)

Only use Test Doubles for immediate neighbours:

- Objects that has to pass through several other objects to complete behaviour will be fragile due to many dependencies over several abstraction layers.
- In these cases the design might be improved looking for code smells like Feature Envy, Inappropriate Intimacy and Message Chain.

The same class CAN act BOTH as a Stub and a Mock:

- What type of Test Double to use is decided at method level
- Depending on the method we are considering it might be a Stub or a Mock

DevOps Metrics Action

GitHub Action

DevOps Metrics from GitHub

v1.0.2 Latest version

Get DevOps Metrics from GitHub project issues and releases

CodeQL passing | units-test passing | Test Coverage 88%

This GitHub Action will calculate a set of DevOps Research and Assessment (DORA) metrics based on status and dates from commits and issues.

Deploy Rate 3.27 | Lead Time 14.84 | Change Failure Rate 14% | Mean time to Restore 3.3

Inputs

repo

Repository from where to read issues and statuses. List one or more repositories, either as one single string, as an array or all separated by newlines. Valid formats are:

```
repo: my-repo
```

Use latest version

Stars

Star 1

Contributors

Links

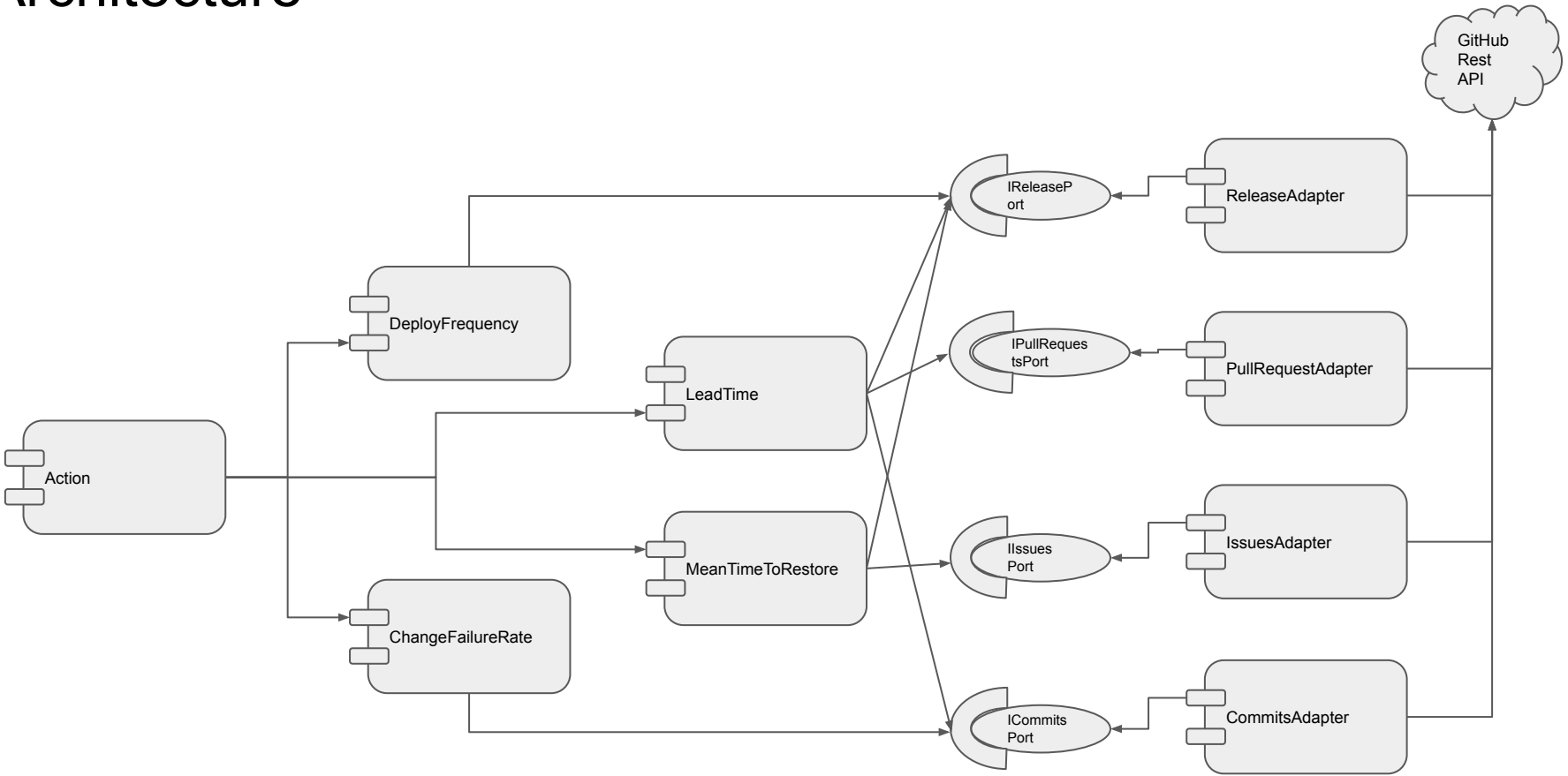
- stenjo/devops-metrics-action
- Open issues 0
- Pull requests 2
- Report abuse

DevOps Metrics from GitHub is not certified by GitHub. It is provided by a third-party and is governed by separate terms of service, privacy policy, and support documentation.

GitHub Action

- Typescript
- Jest testing framework
- 85-90% test coverage

Architecture



Stub

```
2   import {Release} from '../src/types/Release'
3   import fs from 'fs'
4
    Run & Add Only
5   describe('Mocked Release API should', () => {
      Run & Add Only
6     it('return releases', async () => {
7       const r = new ReleaseAdapter(undefined, 'testowner', ['project1'])
8       mockedGetReleasesReturns('./tests/test-data/releases.json')
9
10      const releases: Array<Release> =
11        (await r.GetAllReleasesLastMonth()) as Array<Release>
12
13      expect(releases.length).toBeGreaterThan(0)
14      expect(releases[0].author.type).toBe('Bot')
15      expect(releases.reverse()[0].name).toBe('v0.0.1')
16    })
17  })
18
19  function mockedGetReleasesReturns(file: string) {
20    const getIssuesMock = jest.spyOn(
21      ReleaseAdapter.prototype as any,
22      'getReleases'
23    )
24    getIssuesMock.mockImplementation((): Promise<Release[]> => {
25      return Promise.resolve(
26        JSON.parse(fs.readFileSync(file).toString()) as Release[]
27      )
28    })
29  }
30
```

Testing the release adapter

- Arrange
- Act
- Assert

Helper function:

- Replaces the private function *getReleases* with a mocked function that returns the content of a .json file

Mock

```
121  it('return 8 on pullrequests with base.ref = main and earlier release on other repo', async () => {
122    const pullRequests = [
123      {
124        merged_at: '2023-04-28T17:50:53Z', // 30-22 = 8
125        base: {ref: 'main', repo: {name: 'devops-metrics-action'}},
126        commits_url: 'path/to/commits/1'
127      }
128    ] as PullRequest[]
129    const releases = [
130      {
131        url: 'https://api.github.com/repos/stenjo/other-repo/releases/101411508',
132        published_at: '2023-04-29T17:50:53Z'
133      },
134      {
135        url: 'https://api.github.com/repos/stenjo/devops-metrics-action/releases/101411508',
136        published_at: '2023-04-30T17:50:53Z'
137      }
138    ] as Release[]
139    commitsAdapter.getCommitsFromUrl = jest.fn(
140      (url: string): Promise<Commit[] | undefined> => {
141        return Promise.resolve([
142          {commit: {committer: {date: '2023-04-22T17:50:53Z'}}}
143        ] as Commit[])
144      }
145    )
146    const lt = new LeadTime(pullRequests, releases, commitsAdapter, new Date())
147
148    const leadTime = await lt.getLeadTime()
149
150    expect(commitsAdapter.getCommitsFromUrl).toBeCalled()
151    expect(commitsAdapter.getCommitsFromUrl).lastCalledWith('path/to/commits/1')
152    expect(leadTime).toBe(8)
153  })
154
```

Testing the LeadTime class

- Arrange
- Act
- Assert

Asserts verifies that Mocked function is actually called with expected parameters

Conclusion

- Test Doubles are essential tools for creating robust automated tests that verifies focused functionality
- Mocks and stubs are easily implemented in the test code
- Keeps tests fast, obvious and easily readable, making sure your code quality stays high

References

- **Jest Mock Functions**: https://jestjs.io/docs/mock-function-api
- **Dependency inversion, dependency injection, and unit tests** by Michał Kaczanowicz, https://medium.com/@michakaczanowicz/dependency-inversion-dependency-injection-and-unit-tests-6fbe1c5c9063
- **Lesson 2 - Test Doubles** by Alessandro Di Gioia, Alcor Academy
- **DevOps-metrics-action**: https://github.com/stenjo/devops-metrics-action

Thank you

How to contact me:
sten.johnsen@bouvet.no
@stenjo on Twitter