VII ESCOLA REGIONAL DE ENGENHARIA DE SOFTWARE

ERES 2023

Maringá/PR | 6 a 8 de Dezembro

Identificação de testes instáveis

Rafael Rampim Soratto[1]
Marco Aurélio Graciotto Silva[1]

[1]Universidade Tecnológica Federal do Paraná (UTFPR)
Programa de Pós-Graduação em Ciência da Computação
Campo Mourão - PR - Brasil

8 de dezembro de 2023

Função:

- **Verificação** e **validação** do software.
- Aumenta a confiança no código-fonte
- Mantém **segurança do sistema**;
- Garante **comportamento esperado** na **especificação** do produto.

- Normalmente, utiliza-se um código-fonte contendo testes automáticos (Code under test/ CUT) para as classes e funções de um programa (System under test / SUT).
- O teste de regressão garante que **funcionalidades** de um software não sejam **danificadas** por **atualizações**.

Frameworks

- Jesthttps://github.com/jestjs/jest;
- Karma (Descontinuado)https://github.com/karma-runner/karma;
- Mocha https://github.com/mochajs/mocha;
- Cypress https://www.cypress.io/;

Jest Examples

- React [1]
- React Cosmos[2]
- Moleculer[3]
- Next.js[4]
- Gatsby[5]

[1] https://github.com/facebook/react
[2] https://github.com/react-cosmos/react-cosmos
[3] https://github.com/moleculerjs/moleculer
[4] https://github.com/vercel/next.js/
[5] https://github.com/gatsbyjs/gatsby

Flaky:

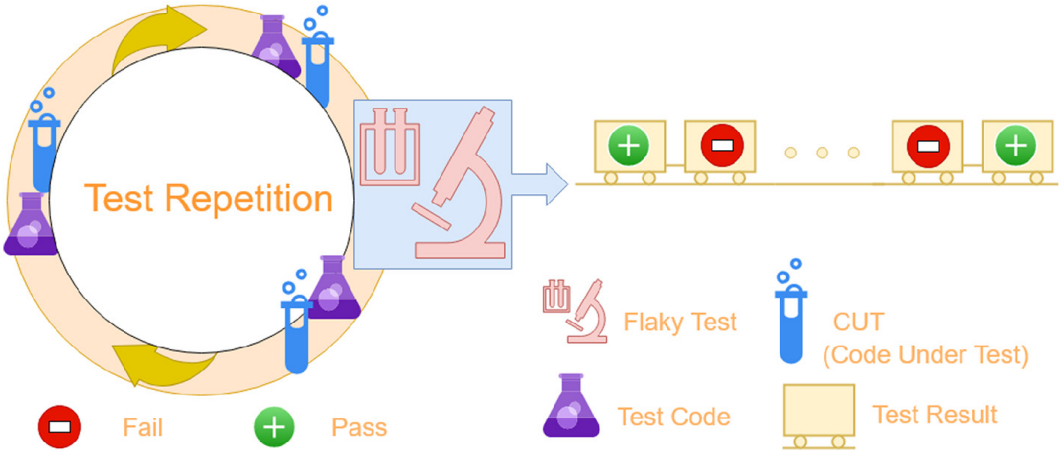
Testes falham em diferentes execuções **sem alteração** do código-fonte (**SUT** / **CUT**);

- Ocorre após **várias execuções** do mesmo código.
- Ocorre geralmente em testes de **interfaces** e **sistemas**.
  - Neles, a dependência entre **componentes, serviços e plataformas** pode ser alta.

Consumo de recurso humano e computacional

Consome **muitos recursos** durante o processo de **deploy** , além de **remover** a confiança na validade dos testes.

Flaky tests identification



Exemplo de correção Flaky Test identificado no Moleculer [6]

Código 1: FLaky relacionado ao serviço externo utilizado no teste (serialização)

```
1  describe("Test MsgPackSerializer", function () {
2    let serializer = new MsgPackSerializer();
3    serializer.init();
4
5    it("should serialize the event packet", function () {
6      const obj = {created_at: new Date("2022-11-06T22:58:47.000Z")
           }
7      const s = serializer.serialize(cloneDeep(obj), P.PACKET_EVENT)
           ;
8
9      expect(s.length).toBe(154); ...
10   });
11 });
```

[6] https://github.com/moleculerjs/moleculer

Código 2: Flaky comment: “Bump up timeout for a flaky test ”

```
2  it(`Focus router wrapper after navigation to regular page (from
       index)`, () {
3      cy.visit(`/`).waitForAPIorTimeout(`onRouteUpdate`, { timeout:
           10000 })

5      cy.changeFocus()
6      cy.assertRouterWrapperFocus(false)
7      cy.navigateAndWaitForRouteChange(`/page-2/`)
8      cy.assertRouterWrapperFocus(true)
9    })
```

Consumo de recursos e perda da confiança nos testes

Consome **muito recurso** durante o processo de **implantação** de um sistema, além de **remover** parte da **confiança** nos testes.

- Em 2015 o time da Microsoft Windows e Dynamics estimam que **5%** das falhas nos testes estão relacionadas aos **flaky**[7].

[7] Herzig e Nagappan, “Empirically Detecting False Test Alarms Using Association Rules”.

- No ano de 2017, foi relatado que **16% dos 4,2 milhões** de testes implementados na Google falharam sem alterações no código, e que eles consumiram de **2% até 16%** de todo o recurso computacional disponível na empresa para execução de testes de regressão[a].

[a] Micco, *The State of Continuous Integration Testing@ Google.(2017)*.

Identificação de instabilidades por reexecução

A abordagem mais comum para identificar uma instabilidade é **executando** o teste diversas vezes.

Problemas relacionados

- **Custo alto** para execução, **atraso** na implantação do software e **falta de confiança** nos testes.
- A falha do teste afeta negativamente a eficácia do teste de regressão e, consequentemente, impacta a evolução do software.

Abordagens para identificação de flaky tests

1. Reexecução;
   - Quanto maior o número, maior a probabilidade de uma instabilidade ocorrer.
2. Análise diferencial de cobertura;
   - Quando o resultado muda de uma execução para outra, é analisado uma diferença entre as execuções.
3. Tests Smells;
   - São utilizados em um modelo de classificação, de forma similar a um vocabulário;
   - Analisa más escolhas nos teste;
   - Desempenho cai consideravelmente no contexto interprojeto.
4. Análise de texto e vocabulários;
   - 100% estática;
   - Precisa de dados reais sobre execuções para realizar novas análises.

1. Escolha um projeto que seja útil pra você. Escolhi o Moleculer (framework de microserviços em NodeJS) [8].
2. Instale as dependências (yarn ou npm i);
3. Verifique no arquivo *package.json* qual comando é utilizado para executar o teste (dentro da chave scripts).
   - No conteúdo do comando você descobrirá qual framework executa os testes.
   - Exemplo : *"test": "jest –coverage",*
4. gere um **build** se for necessário, e execute o comando de teste (geralmente yarn test).
5. Repita o passo anterior até encontrar uma falha.

No repositório está descrito todo o processo de identificação e correção de flaky test em software livre [9].

[8] https://github.com/moleculerjs/moleculer
[9] https://github.com/sorattorafa/flakyrequest

1. Clone o projeto[10] na sua máquina.
2. Instale as dependências e execute o teste pelo menos uma vez.
3. Discuta com os colegas sobre:
   - Existem testes instáveis no projeto?
   - A execução de algum teste falhou na sua máquina e na do colega não?

[10] https://github.com/flakytestresearch/jests-tests-example

ReRun

ReRun é a abordagem mais popular na indústria para detectar falhas em testes.

- Ele reexecuta um conjunto de testes em uma versão de código fixa várias vezes, procurando saídas inconsistentes nas execuções.
- Infelizmente, ReRun é caro e não confiável.

Shaker

O SHAKER é uma técnica mais leve para melhorar a capacidade do ReRun de detectar testes flaky[11].

- Adiciona ruído no ambiente de execução (por exemplo, adiciona tarefas estressoras para competir pela CPU ou memória).

[11] Silva, Teixeira e D'Amorim, "Shake It! Detecting Flaky Tests Caused by Concurrency with Shaker".

- Baseia-se nas observações de que a simultaneidade é uma importante fonte de instabilidade;
- A adição de ruído no ambiente pode interferir na ordenação dos eventos e, consequentemente, influenciar nas saídas dos testes.

Shaker para projetos NodeJS

Implementamos o Shaker para projetos NodeJS que utilizam os frameworks de teste Jest e Karma.

Descrição dos argumentos do Shaker

Args

Short	Full command	Description
-h	–help	show this help message and exit
-o	–output-folder	specify output folder
-sr	–stress-runs	specify number of stress runs
-nsr	–no-stress-runs	specify number of no-stress runs
-tc	–tests-command	specify the command of the test(s) Shaker will execute

Exemplo

./shaker-js/shaker/shaker.py jest “jests-tests-example” -tc “yarn test” -o “jests-tests-example/output” -sr 1 -nsr 1

Executando Shaker no Actions do Github

Caso tenha algum problema em executar localmente, você pode executar na nuvem, utilizando Github Actions.

1. Faça um fork do projeto [12].
2. Vá em Actions e execute o workflow do Shaker.

[12] https://github.com/flakytestresearch/jests-tests-example

1. Clone um dos projetos;
2. Execute localmente ou no actions:
   - Localmente, execute o comando: ./shaker-js/shaker/shaker.py –stress-runs 1 –no-stress-runs 1 jest ”projectname--tests-command ”yarn test--output-folder ”projectname/output”
   - No actions, crie um fork, adicione uma action semelhante ao projeto exemplo, e execute.

VII ESCOLA REGIONAL DE ENGENHARIA DE SOFTWARE

ERES.2023

Maringá/PR | 6 a 8 de Dezembro

Identificação de testes instáveis

Rafael Rampim Soratto[1]
Marco Aurélio Graciotto Silva[1]

[1]Universidade Tecnológica Federal do Paraná (UTFPR)
Programa de Pós-Graduação em Ciência da Computação
Campo Mourão - PR - Brasil

8 de dezembro de 2023