Break New Ground

Construa testes efetivos através do princípio F.I.R.S.T

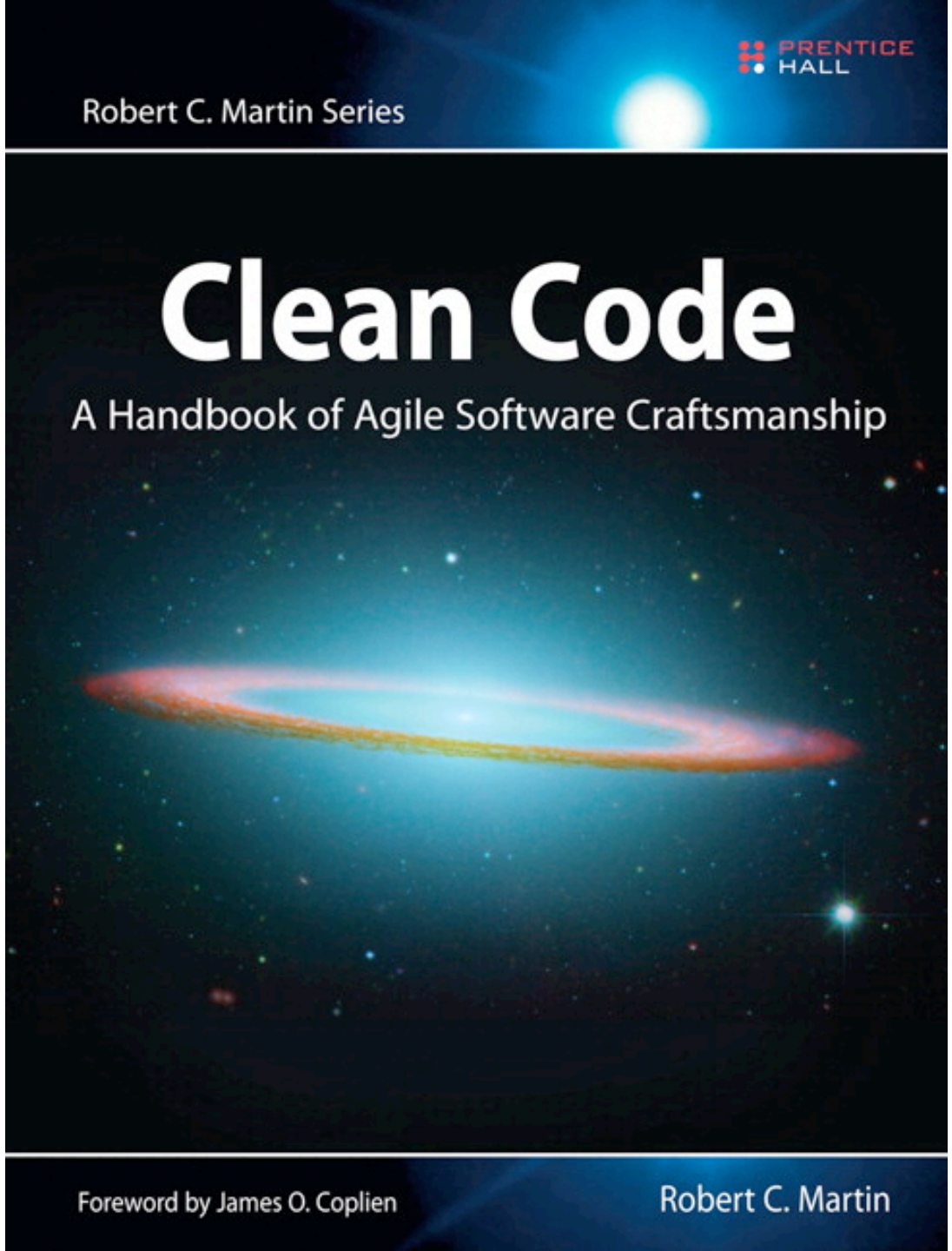
Elder Moraes | @elderjava
Developer Advocate

Abril, 2019

ORACLE

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.


PRENTICE HALL
Robert C. Martin Series
Clean Code
A Handbook of Agile Software Craftsmanship
Foreword by James O. Coplien
Robert C. Martin

FAST

INDEPENDENT

REPEATABLE

SELF-VALIDATING

TIMELY

FAST

- Se os testes forem lentos, você não vai querer executá-los com frequência
- Se você não os executa com frequência, não encontrará problemas cedo o suficiente para corrigí-los facilmente

O que pode fazer um teste ser lento

Execução manual

Validação manual

A própria execução do teste

FAST

- Eliminando a execução manual
  - Frameworks: JUnit, Arquillian, Selenium, Mockito, Wiremock
  - Automações: Maven, Gradle, Jenkins, Vagrant, Ansible, Docker Compose
  - Monitoramento: Grafana, Prometheus, Jaeger

ORACLE

- Eliminando a validação manual
  - Vide SELF-VALIDATING

- Eliminando lentidão no próprio teste
  - Identificação dos pontos de lentidão (banco de dados, integração, rede, etc)
  - Refatoração de código
  - Mocking

INDEPENDENT

- Um teste não deve depender do outro
- Um teste não deve criar as condições para o próximo teste
- Você deve ser capaz de executar cada teste individualmente e executar todos os testes em qualquer ordem
- Quando os testes depende uns dos outros, o primeiro a falhar causa uma reação em cadeia, dificultando o diagnóstico e escondendo defeitos que possam estar adiante

INDEPENDENT

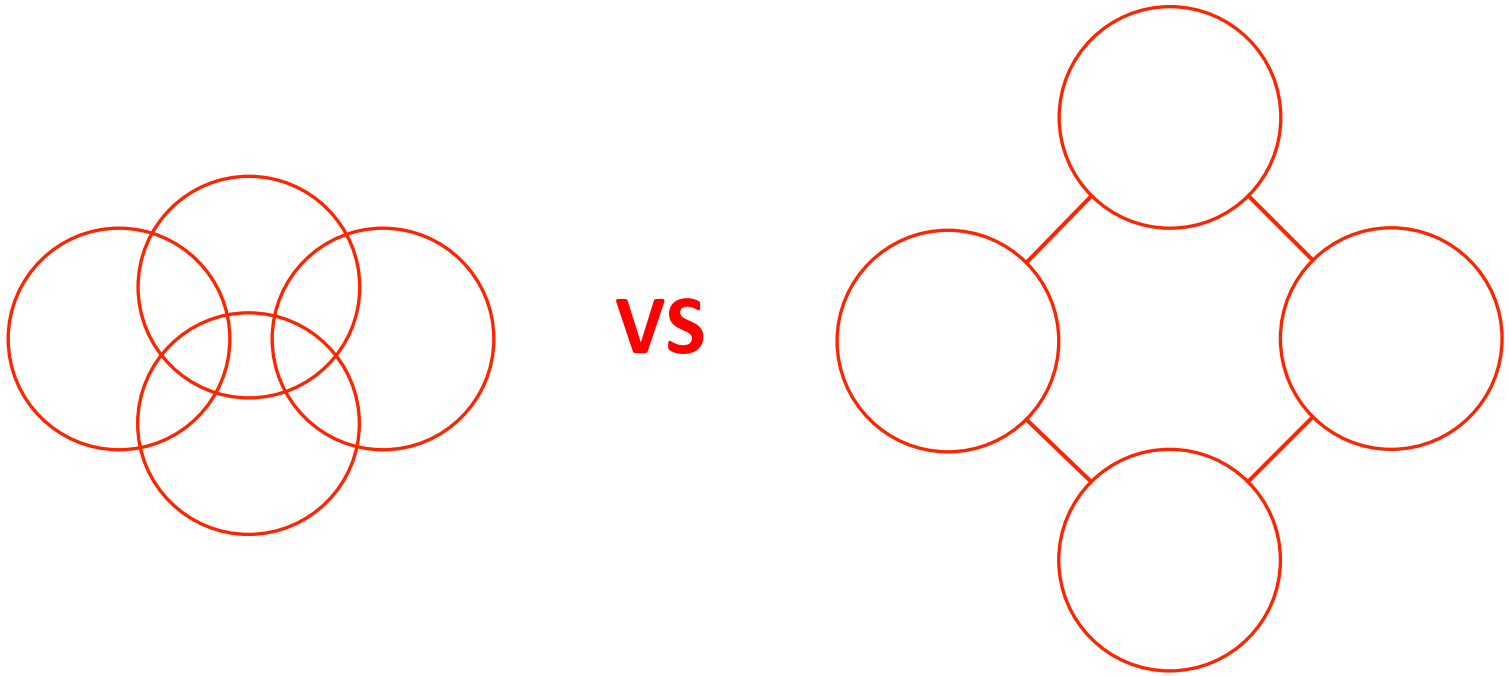
- Como eliminar dependência entre testes
  - Alta coesão
  - DRY (**D**on't **R**epeat **Y**ourself)

"We want to design components that are self-contained: independent, and with a single, well-defined purpose"

- The Pragmatic Programmer

INDEPENDENT

- Como eliminar dependência entre testes
  - Baixo acoplamento



```
public Response addUser(String name,
                        String email,
                        Double salary){
    ...
}
```

```
public Response addUser(User user){
    ...
}
```

REPEATABLE

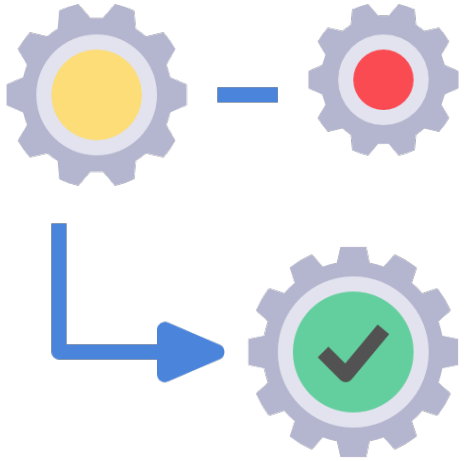
- Testes deve ser reproduzíveis em qualquer ambiente
- Produção, QA, laptop (mesmo sem rede)
- Do contrário, você sempre terá uma desculpa para não executá-los
- Além disso, não conseguirá executá-los quando o ambiente não estiver disponível

REPEATABLE

- Eliminando a dependência de ambientes

Banco de Dados

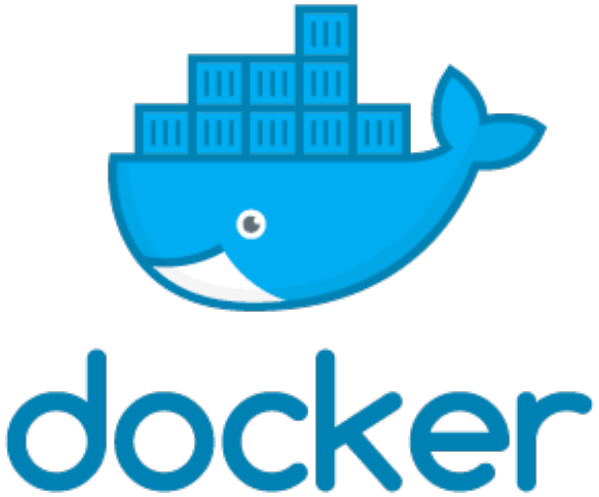
Sistema Operacional

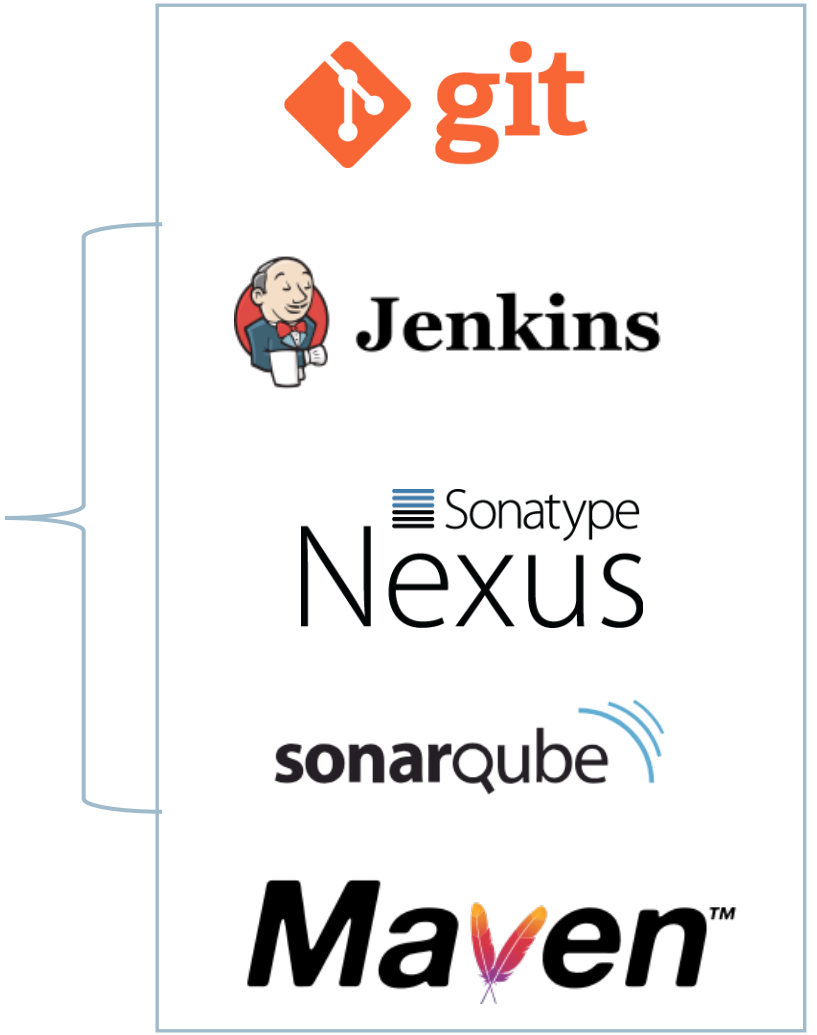


Deploy e Execução

REPEATABLE

- Eliminando a dependência de ambientes





Vagrantfile

SELF-VALIDATING

- Testes devem ser auto-explicativos: **passou/falhou**
- O teste não deve exigir que *você*:
  - Leia um log
  - Compare dois arquivos
  - Veja uma tela
  - Etc…
- Se não forem auto-explicativos, então a falha pode ser subjetiva
- Leva à demora na validação (e falha no princípio FAST)

```java
@Test
public void teste1() {
    Pessoa[] pessoas = repository.getPerson();
    boolean passou = false;

    for (int i = 0; i < pessoas.length; i++) {
        if (pessoas[i].getName().equals("Elder Moraes")) {
            passou = true;
        }
    }

    if (passou) {
        assertTrue(true);
    }
}
```

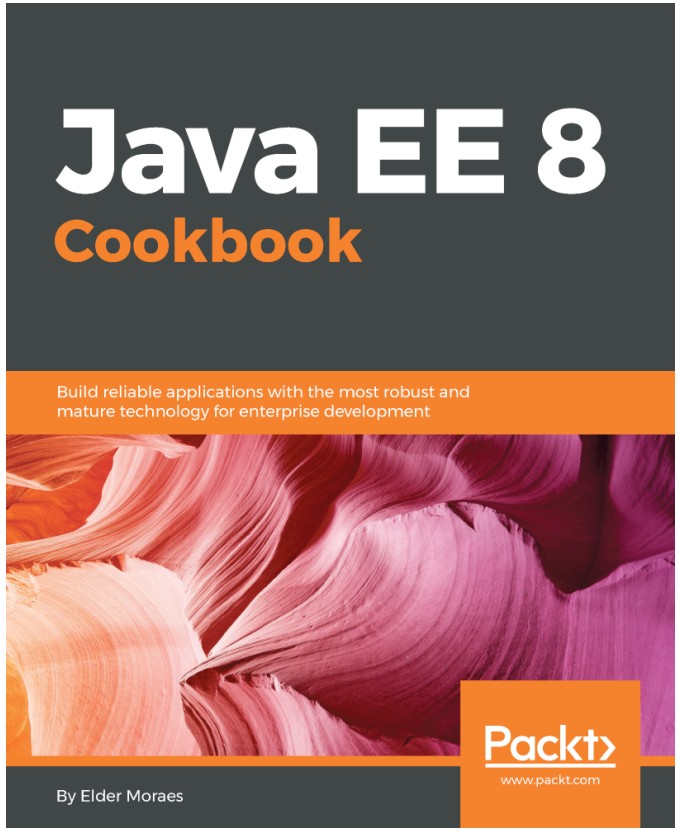
```java
@Test
public void validaRetornoDados() {
    Pessoa pessoaEsperada = new Pessoa("Elder Moraes");
    List<Pessoa> pessoas = repository.getPerson();

    assertThat(pessoas, hasItem(pessoaEsperada));
}
```

TIMELY

- Testes devem ser escritos o quanto antes
- Se você espera um código entrar em produção para escrever o teste, pode acabar descobrindo que é difícil testá-lo
- Ao fazer isso continuamente, logo você terá todo seu código de produção impossível de ser testado

"Tests are **as important**… as the **production** code is. Perhaps they are even **more important**, because tests preserve and enhance the **flexibility, maintainability, and reusability** of the production code."

– Rober C. Martin, "Clean Code"



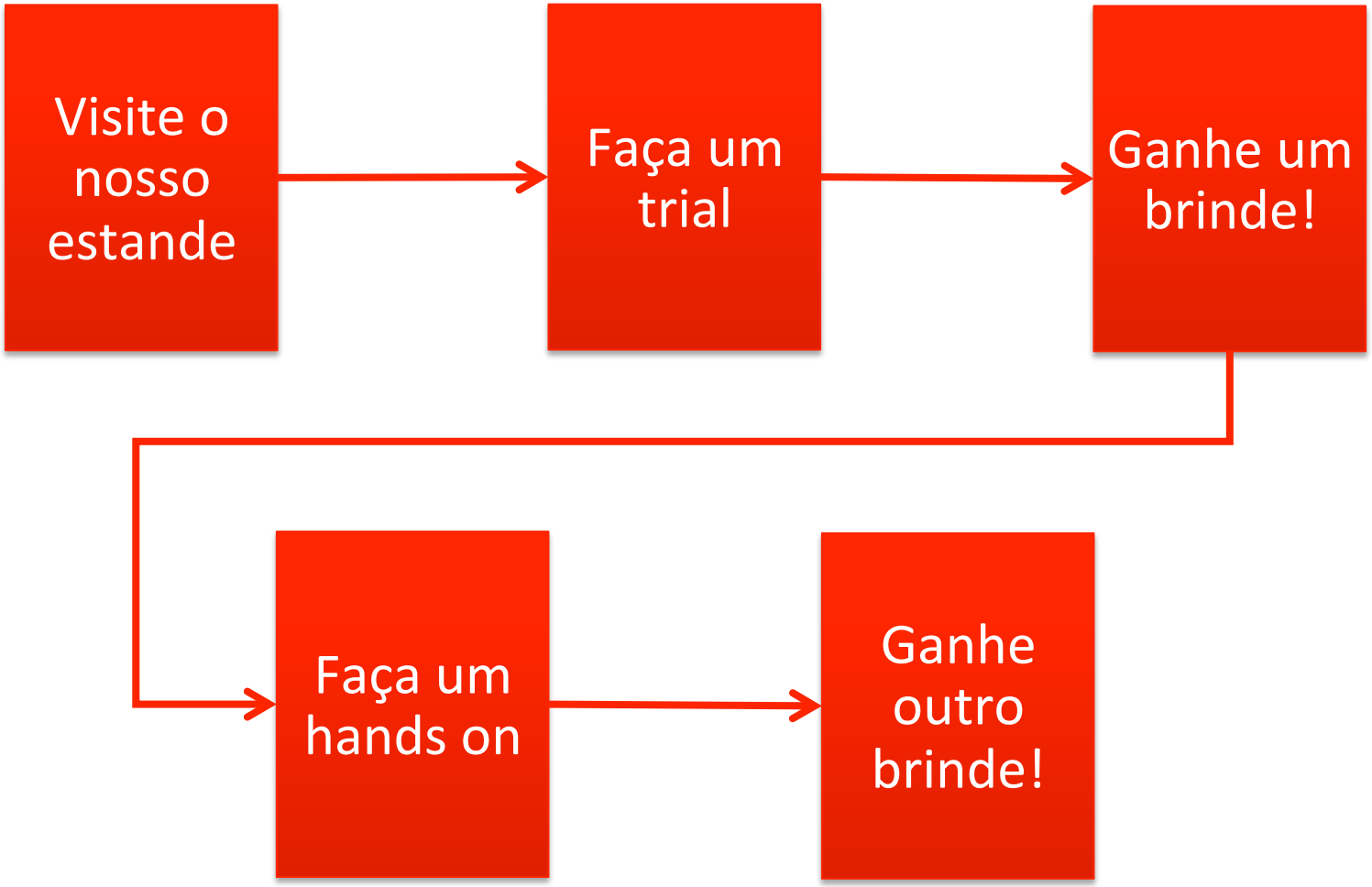
@elderjava

book.eldermoraes.com

ORACLE

Java@Cloud Age

bit.ly/javacloudage



Break New Ground