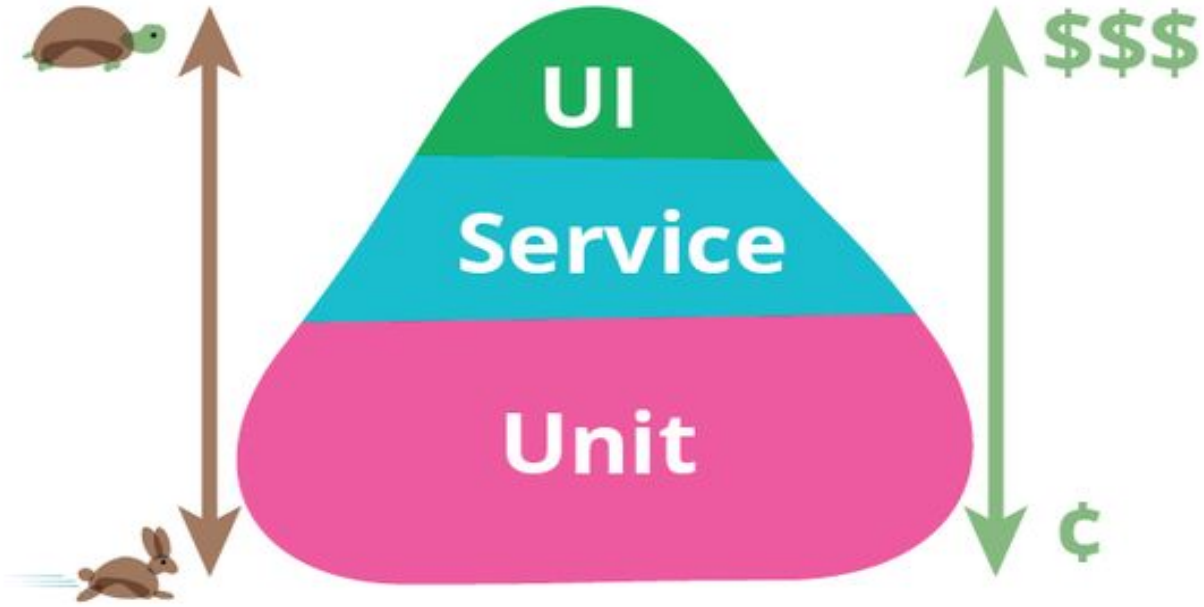
Faculdade IMPACTA

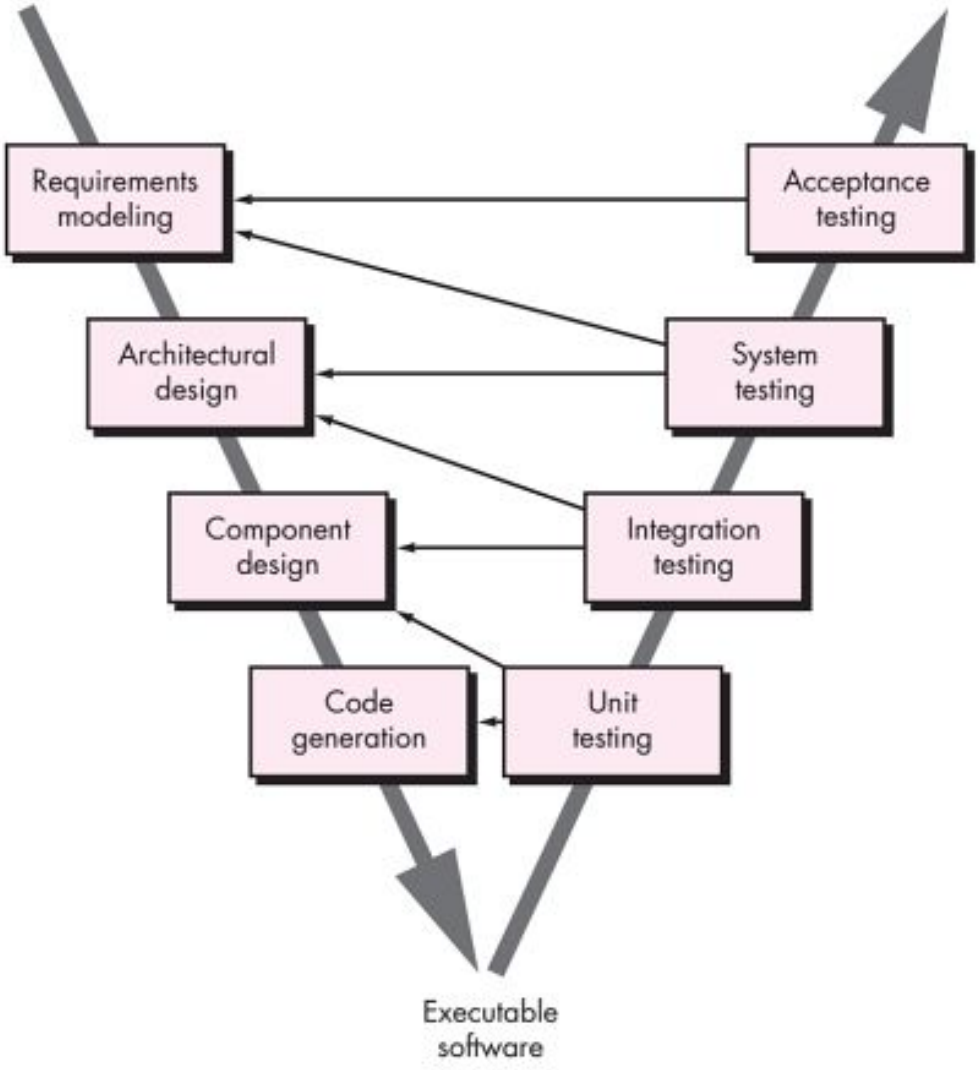
Automated Software Testing

João Henrique Victorino da Silva

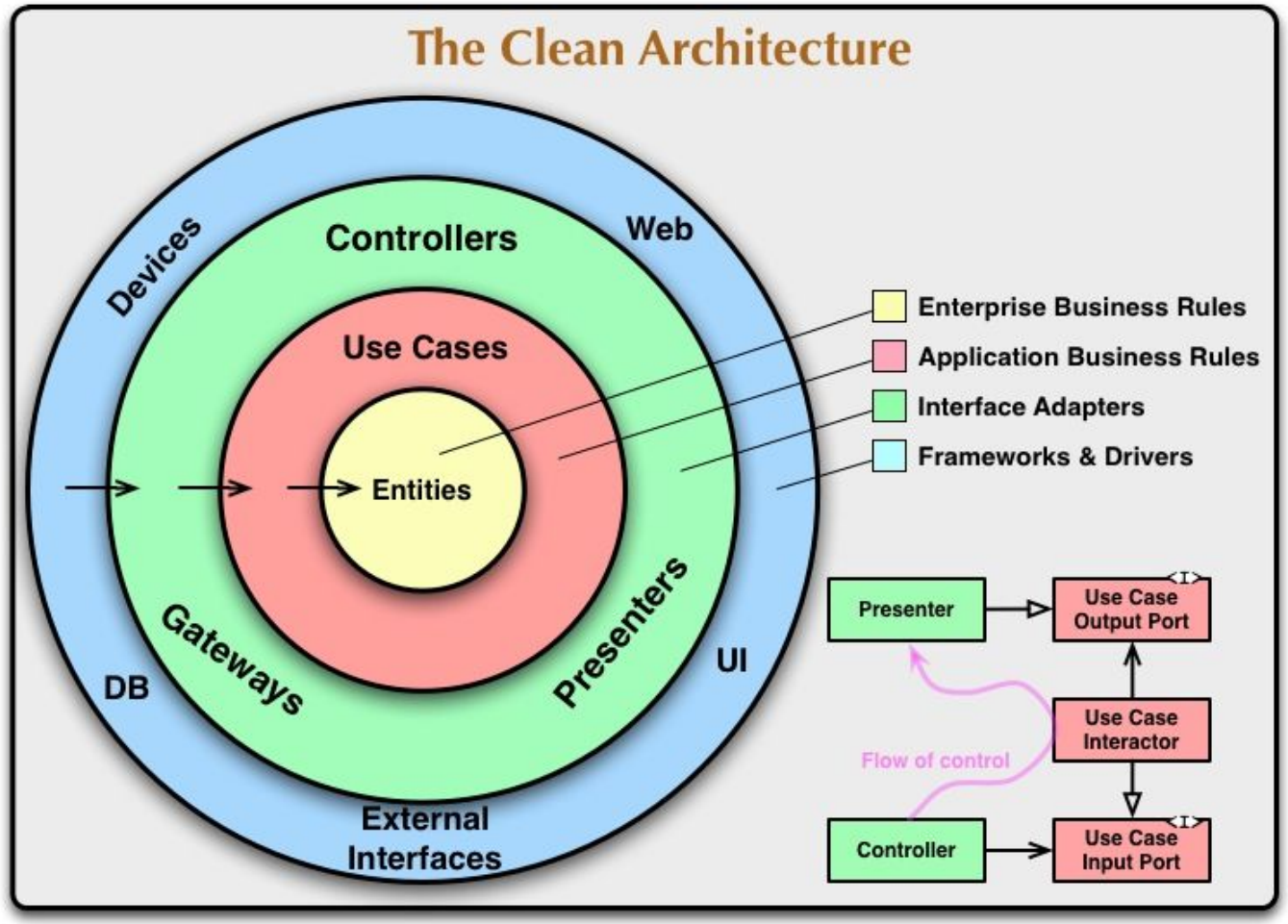
Pirâmide de testes



V-Model


Requirements modeling
Acceptance testing
Architectural design
System testing
Component design
Integration testing
Code generation
Unit testing
Executable software

Clean Architecture

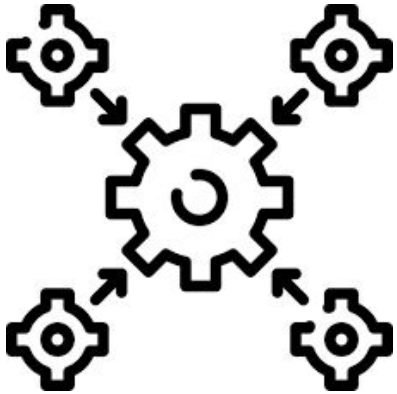


Granularidade dos testes

- **Testes de unidade**, classes e métodos isolados
- **Testes de integração**, um componente ou mais
- **Teste de sistema**, todo o sistema, relação entre componentes e infraestrutura (RNF)
- **Teste de aceitação**, todo o sistema, relação entre componentes e infraestrutura (RF)

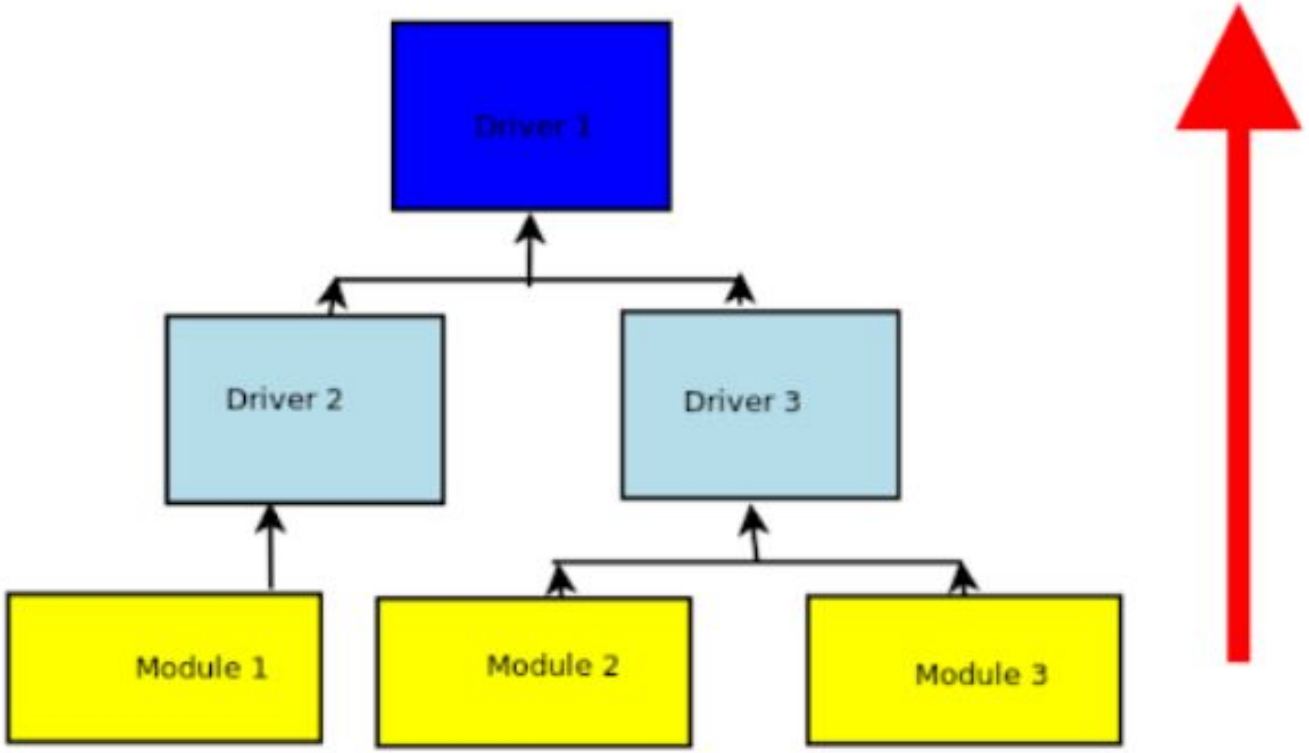
Testes de integração

- Testar a relação/colaboração entre classes e componentes
- Se necessário ainda exclua a infraestrutura
  - Ainda será necessário usar mocks para outras classes ou componentes
- Pode testar recursos externos para garantir qualidade
- Explorar exceções que não puderam ser tratadas durante o teste de unidade



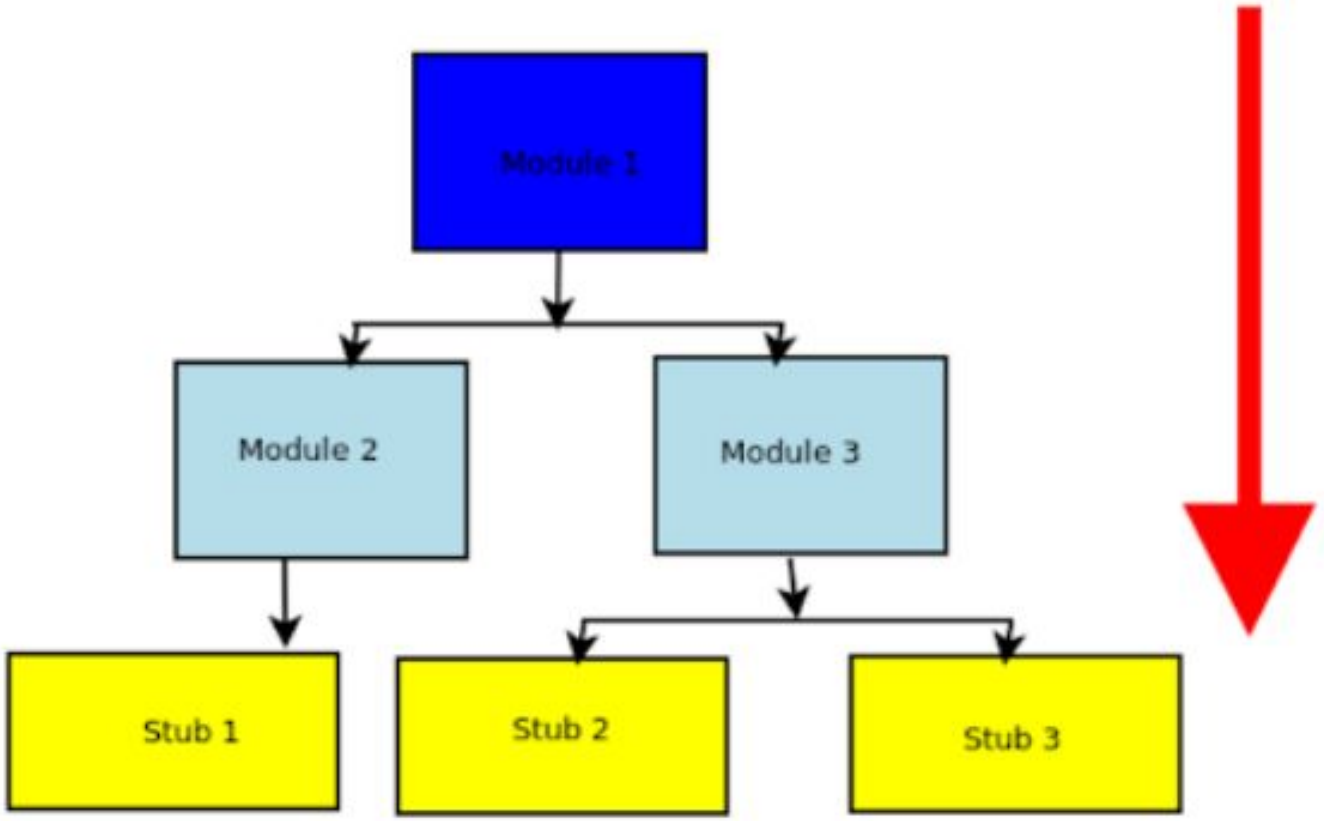
Testes de integração

Bottom-up



Testes de integração

Top-down



Criando a camada de aplicação

https://gist.github.com/joaovictorino/fd80a207c35bef460cee34291022a3ca

(20 minutos)

Validando fluxos alternativos de aplicação

https://gist.github.com/joaovictorino/50abe117aaa52633b88dec92b730b8fc

(15 minutos)

Refatorando o teste de aplicação

https://gist.github.com/joaovictorino/52600cfea9e651fab9212e9a385bd175

(10 minutos)

Dependências externas

- Fontes de dados
  - A fonte de dados mais comum é o banco de dados, mas essa dependência torna seu teste vulnerável e lento
- Virtualização de recursos
  - Quando seu componente depende de algo externo ao sistema, como uma integração via API ou elementos de infra

- **Mocks, Stubs, Fakes e Spies!!!**

Dublês de testes

Tipos de dublês

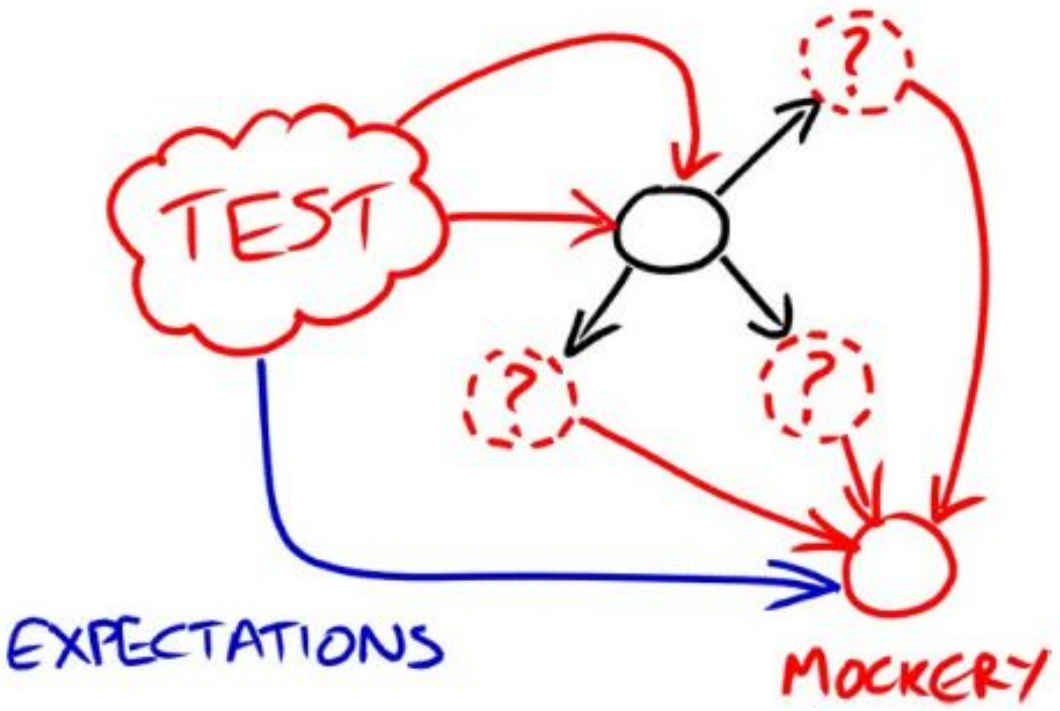
- Fakes
  - Implementação funcional, por exemplo, usar um banco de dados em memória, mas não é usado em produção.
- Dummies
  - Objeto criado mas nunca utilizado, existem apenas para cumprir contratos.
- Stubs
  - Responde exatamente aquilo que é necessário para o teste (retorno fixo). Normalmente não responde para aquilo que não foi programado.

Tipos de dublês

- Spies
  - São stubs que registram alguma informação quando são chamados (Serviço de email que guarda no sistema de arquivos os emails enviados)
- Mocks
  - São objetos pré programados com expectativas que formam uma especificação das chamadas que eles devem receber

Frameworks de mocks

- Frameworks criam os tipos dinamicamente
- Normalmente implementam mais de um tipo de dublê
- Posso verificar internamente no Mock se o estado está correto



Fakes

```
export class MemoriaContaRepositorio implements Repositorio<Conta, string> {
    private _dicionario: Map<string, Conta>;
    constructor(){
        this._dicionario = new Map<string, Conta>();
    }
    public buscar(numero: string): Conta | undefined {
        return this._dicionario.get(numero);
    }
    public adicionar(entidade: Conta): void {
        this._dicionario.set(entidade.numero, entidade);
    }
}
```

Mock

```
test("transferir com sucesso", () => {
    const { repositorio, contaOrigem, contaDestino } = criarMock();
    const transferenciaServico = new TransferenciaServico(repositorio);
    const dto = new TransferenciaDTO("123456", "654321", 100.0);
    const recibo = transferenciaServico.transferir(dto);

    expect(repositorio.buscar).toBeCalledTimes(2);
    expect(repositorio.adicionar).toBeCalledTimes(2);
    expect(repositorio.buscar("123456")!.saldo).toBe(4900.0);
    expect(repositorio.buscar("654321")!.saldo).toBe(5100.0);
    expect(recibo.length).toBe(6);
});
```

Trabalhando com mocks

https://gist.github.com/joaovictorino/61ad34bc4c5cc8d1048394381bb3757d

(10 minutos)

Cobertura de código

Acima de 80% é caro, então teste os módulos mais críticos/complexos

```
/**
 * @param price   The price to set.
 */
public void setPrice(String price) throws RecipeException{
    int amtPrice = 0;
    try {
        amtPrice = Integer.parseInt(price);
    } catch (NumberFormatException e) {
        throw new RecipeException("Price must be a positive integer");
    }
    if (amtPrice >= 0) {
        this.price = amtPrice;
    } else {
        throw new RecipeException("Price must be a positive integer");
    }
}
```

Jest com cobertura de código


```
$ npx jest --coverage
```

Jest com cobertura de código

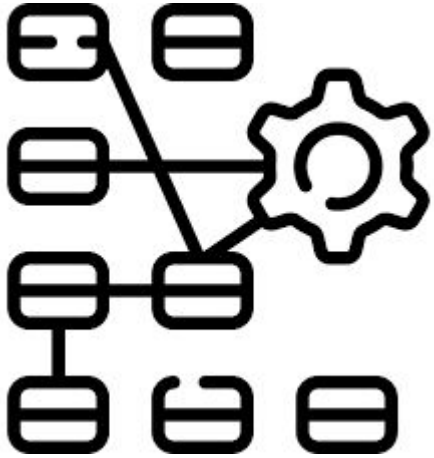
https://gist.github.com/joaovictorino/223c12ce0c3c3f185674362fba571667

(5 minutos)

Limitar a complexidade

Reduzir a complexidade

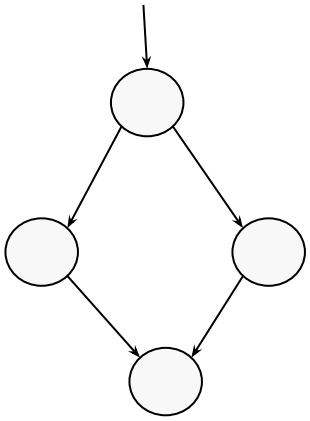
1. Reduzir acoplamento
2. Aumentar coesão
3. Criar interfaces bem definidas
4. Reduzir a complexidade das classes



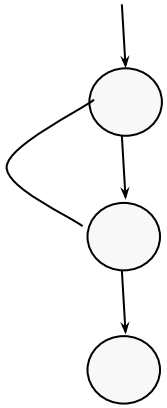
Complexidade Ciclomática

Thomas J. MacCabe - 1976

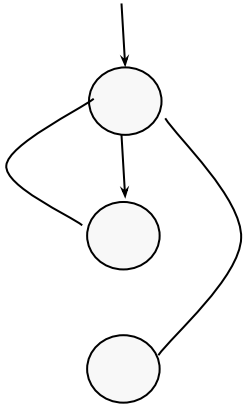
Mede a complexidade de um código a partir da quantidade de caminhos independentes existentes.



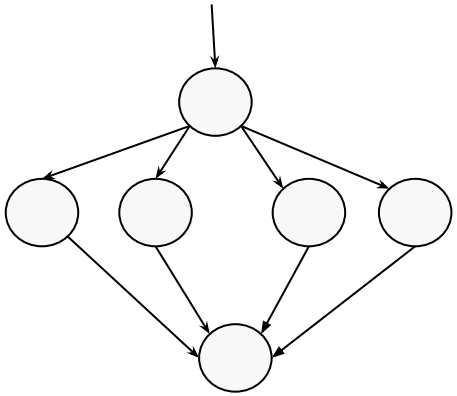
if-then-else



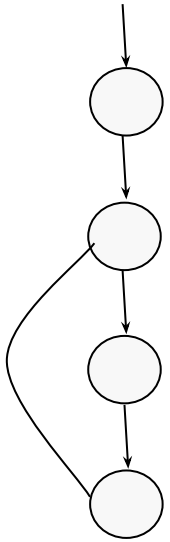
do-until



while



case

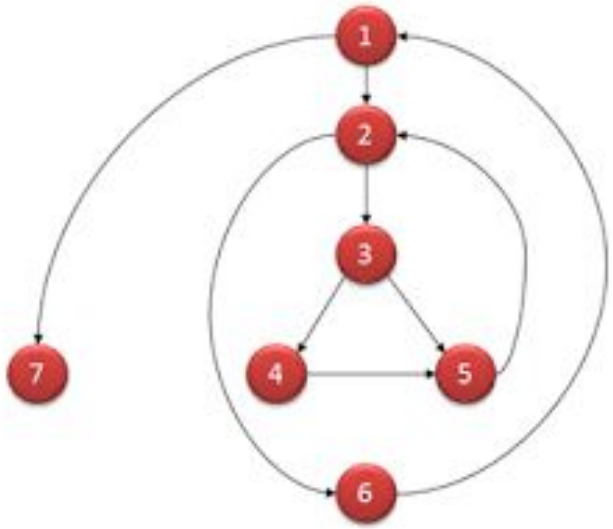


for

Complexidade Ciclomática

Thomas J. MacCabe - 1976

- Para obter a métrica, o grafo de controle é um grafo direcionado:
  - Nós representam os blocos
  - Arestas representam o fluxo de controle de um bloco para o outro.



Complexidade Ciclomática

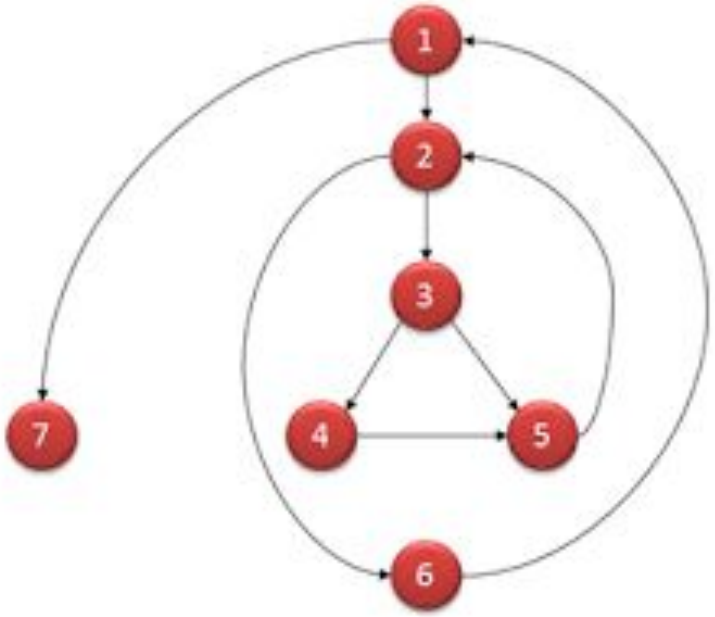
Thomas J. MacCabe - 1976

A complexidade de MacCabe pode ser expressa como segue

$$M = E - N + 2$$

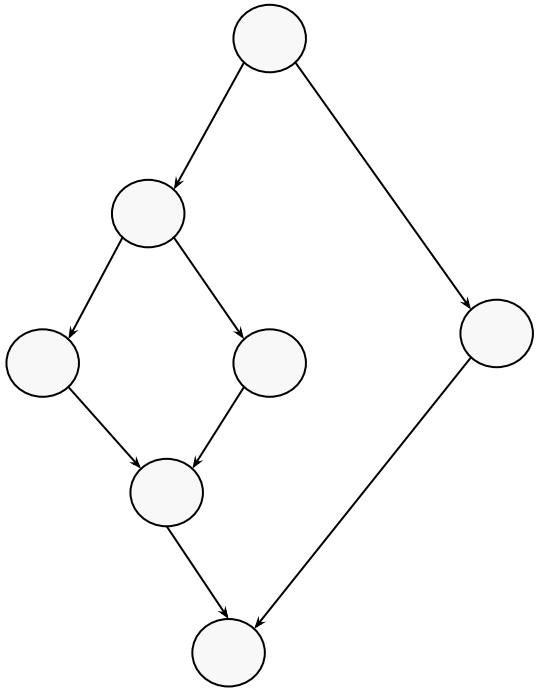
onde,

E: número de arestas do grafo
N: número de nós do grafo



Complexidade Ciclomática

Thomas J. MacCabe - 1976



```
if a == b:
    if a < b:
        menor = a
    else:
        menor = b
else:
    menor = 0
```

N = 7

E = 8

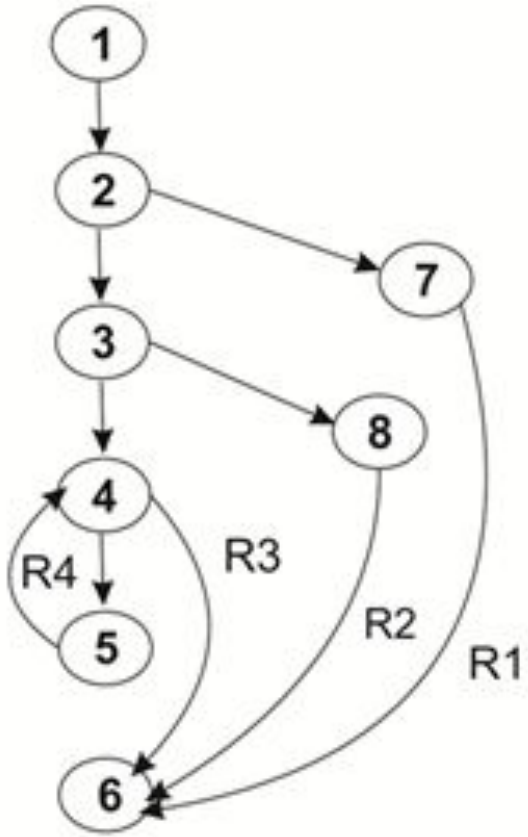
M = 8 - 7 + 2 = 3

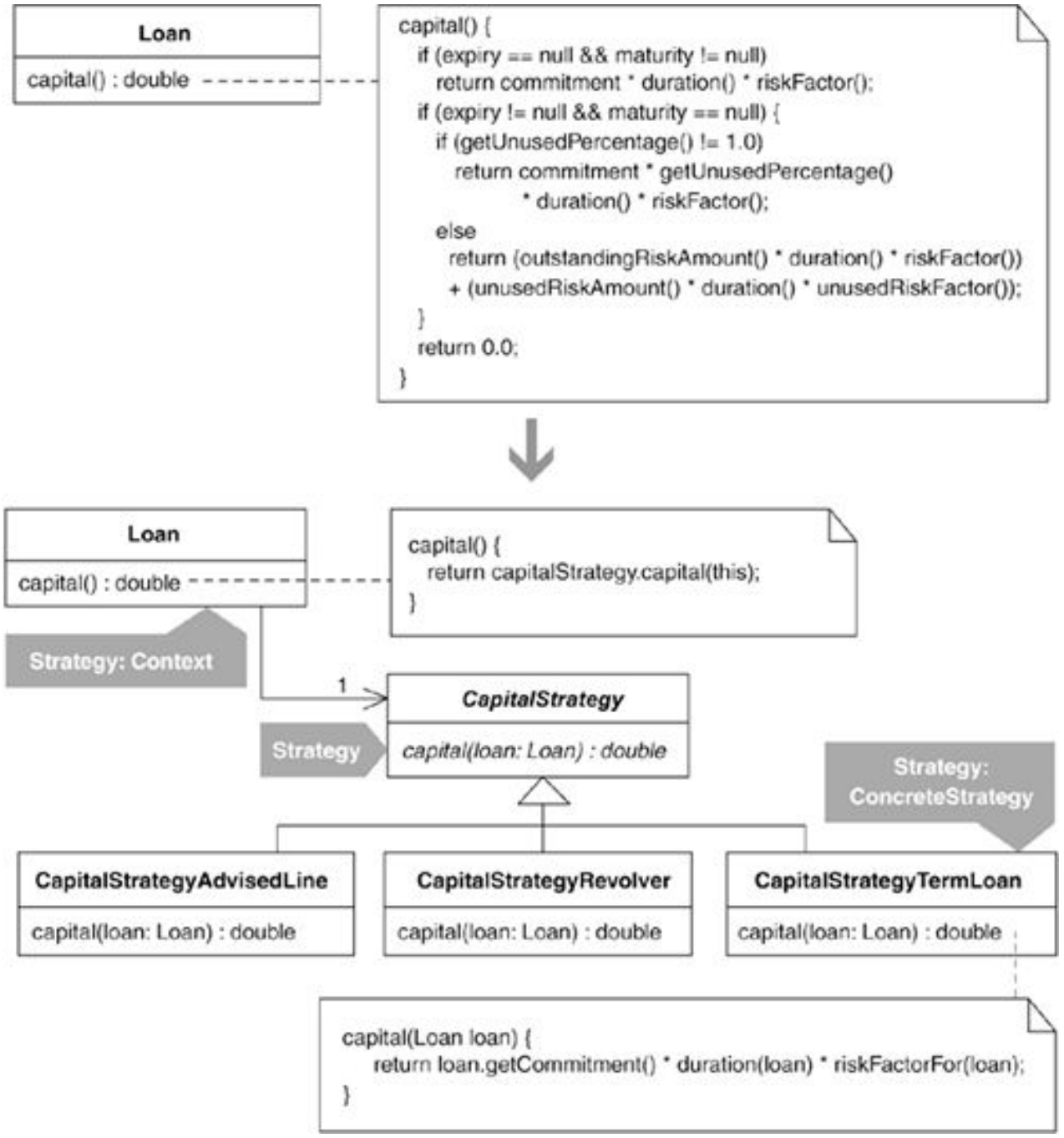
Benefícios da métrica de complexidade

Complexidade Ciclomática	Avaliação de Risco
01-10	Programa simples, sem muito risco
11-20	Complexo, risco moderado
21-50	Complexo, programa com alto risco
maior que 50	Programa quase impossível de ser testado (Altíssimo risco)

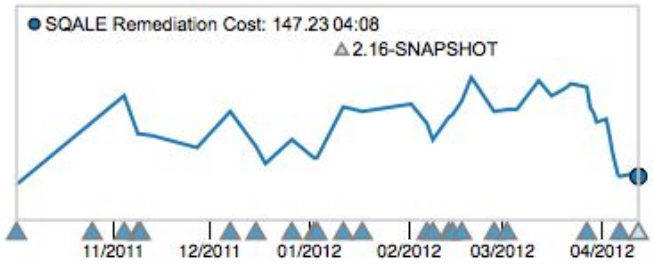
Complexidade ciclomática

```
        int fib (int n) {
            int a = 1;
1           int b = 1;
            int c = 2;

2           if (0 == n)
                return 0;  ←7
3           if (n < 3)
                return 1;  ←8

4           while (n-- > 2) {
                c = a + b;
5               a = b;
                b = c;
            }

6           return c;

        }
```




Loan
capital() : double
capital() {
if (expiry == null && maturity != null)
return commitment * duration() * riskFactor();
if (expiry != null && maturity == null) {
if (getUnusedPercentage() != 1.0)
return commitment * getUnusedPercentage()
* duration() * riskFactor();
else
return (outstandingRiskAmount() * duration() * riskFactor())
+ (unusedRiskAmount() * duration() * unusedRiskFactor());
}
return 0.0;
}
Loan
capital() : double
capital() {
return capitalStrategy.capital(this);
}
Strategy: Context
1
CapitalStrategy
capital(loan: Loan) : double
Strategy
Strategy: ConcreteStrategy
CapitalStrategyAdvisedLine
capital(loan: Loan) : double
CapitalStrategyRevolver
capital(loan: Loan) : double
CapitalStrategyTermLoan
capital(loan: Loan) : double
capital(Loan loan) {
return loan.getCommitment() * duration(loan) * riskFactorFor(loan);
}

SonarQube



Violations
1,148

Rules compliance
96.7%

Blocker	0	
Critical	0	
Major	637	
Minor	178	
Info	333	

Alerts : Skipped unit tests > 0.

Lines of code
62,406
110,305 lines
23,838 statements
1,195 files

Classes
1,300
171 packages
7,029 methods
1,348 accessors

Code coverage
68.7%
69.3% line coverage
67.0% branch coverage

Unit test success
100.0%
0 failures
0 errors
2,597 tests
3 skipped
4:08 min

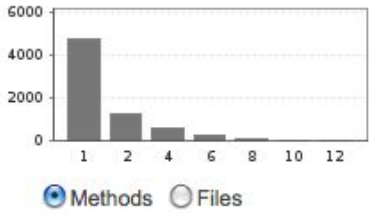
Comments
7.7%
5,208 lines
26.9% docu. API
4,272 undocu. API

Duplications
0.8%
924 lines
49 blocks
35 files

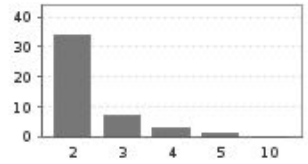
Package tangle index
10.1%
> 56 cycles

Dependencies to cut
35 between packages
51 between files

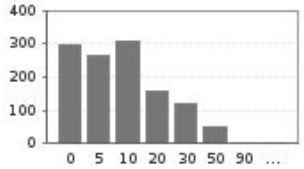
Complexity
1.9 /method
10.5 /class
11.4 /file
Total: 13,592



LCOM4
1.1 /class
3.8% files having LCOM4>1



Response for Class
15 /class



Rodando o SonarQube com Docker

https://gist.github.com/joaovictorino/85cbecc3d430a43e21ba80ccba5f78dc

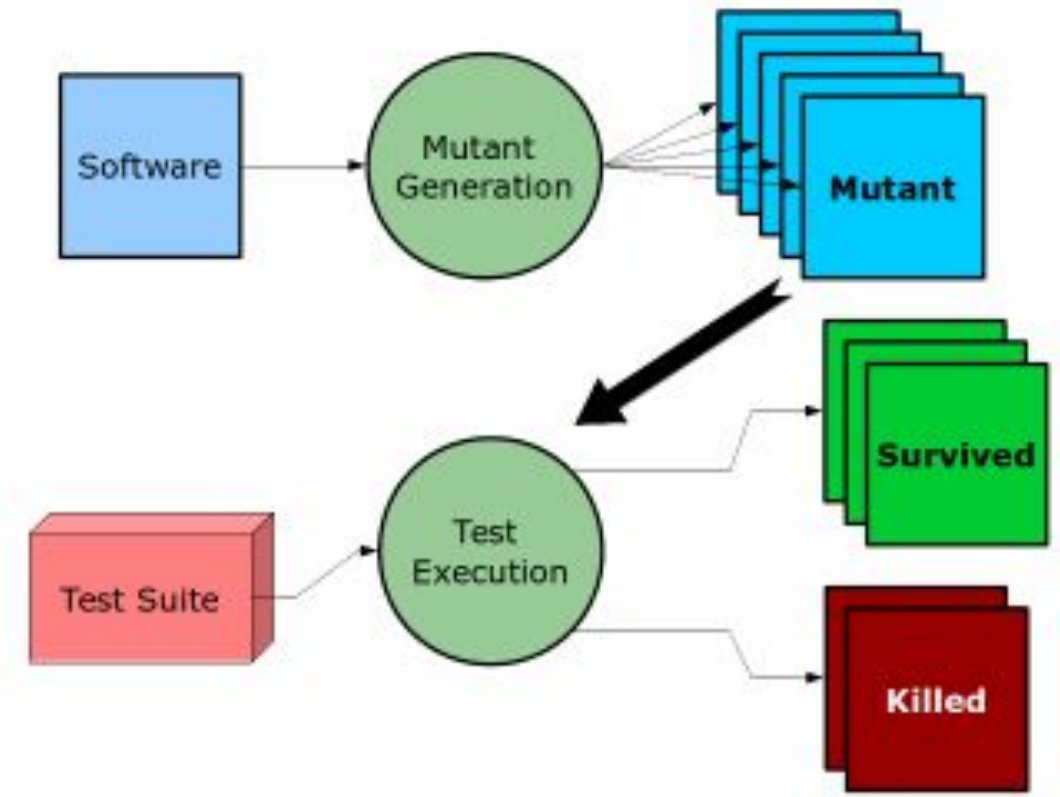
(10 minutos)

Analisando nosso código

https://gist.github.com/joaovictorino/e21f9d556c5e028940c92ddb5a790550

(10 minutos)

Testes mutantes



Stryker

https://stryker-mutator.io/

Executando testes de mutação

https://gist.github.com/joaovictorino/23d6a53f634c141c34d068db01236a88

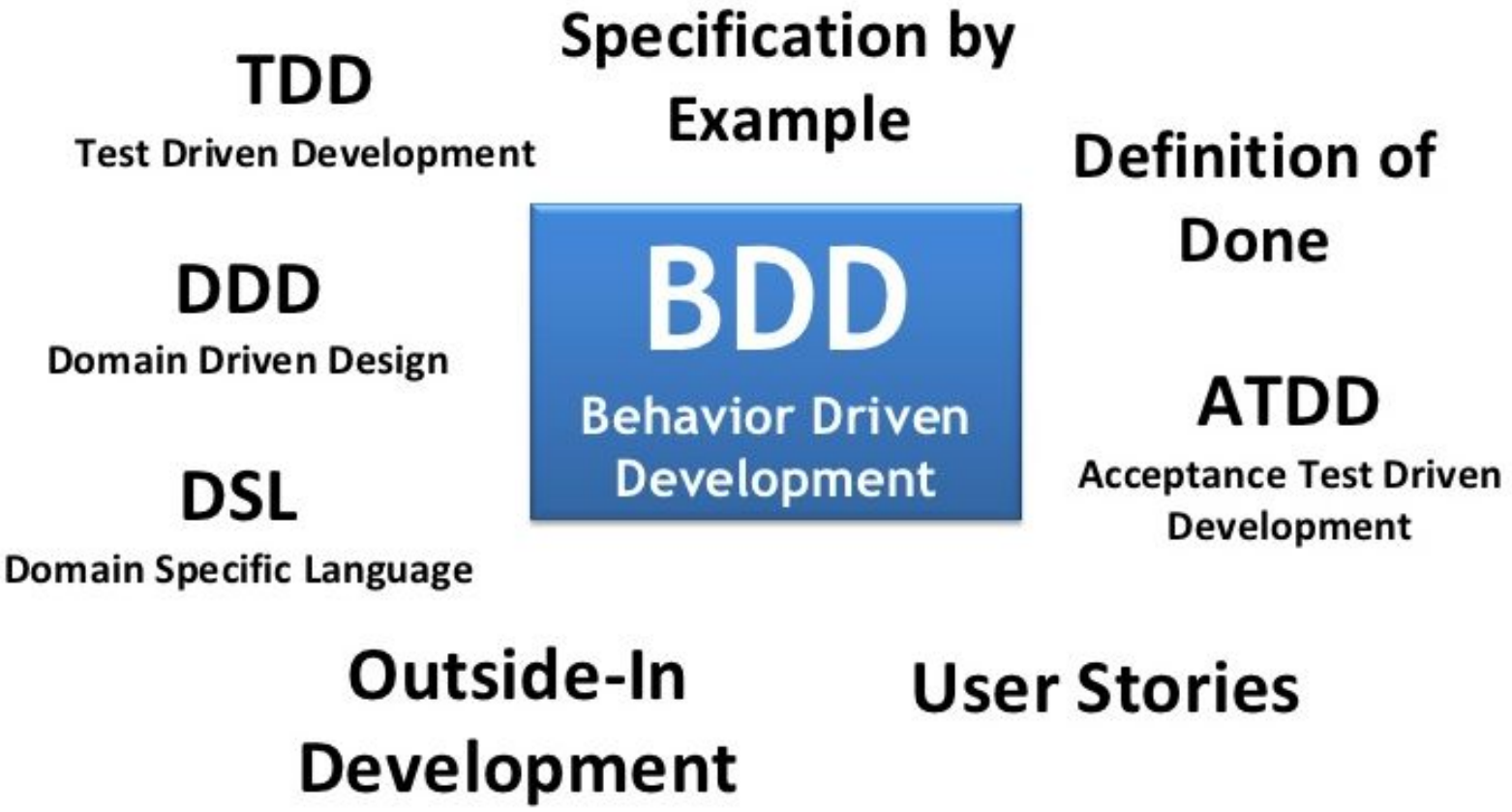
(10 minutos)

Corrigindo nossos testes

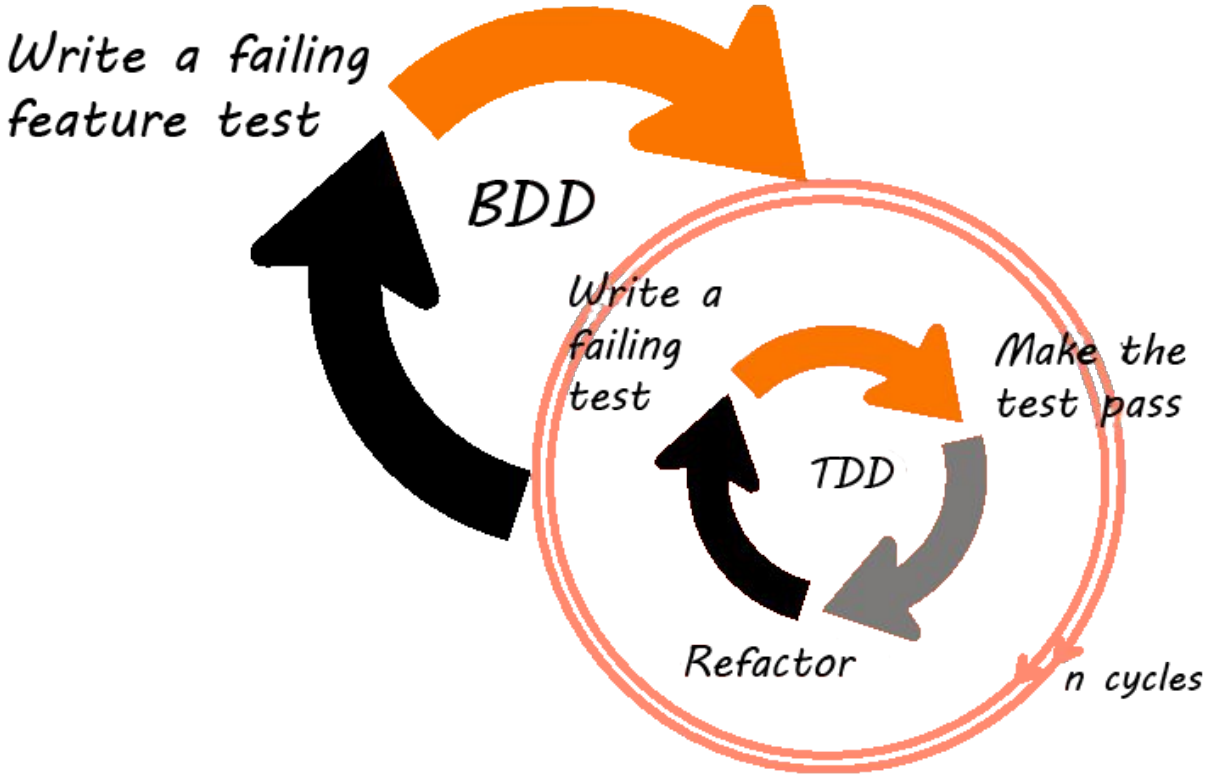
https://gist.github.com/joaovictorino/be42a717ecdd69cc1a29768401899b72

(10 minutos)

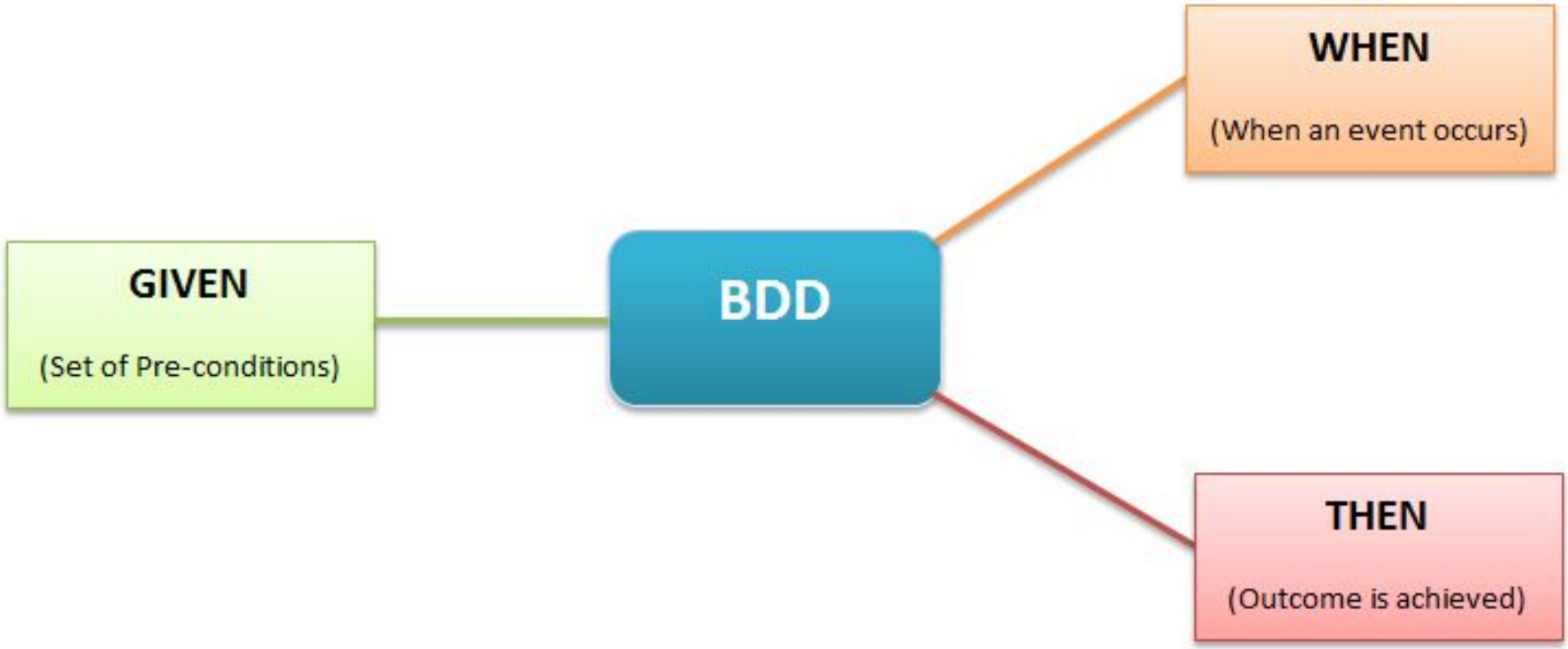
Behavior Driven Development (BDD)



TDD e BDD


Write a failing feature test
BDD
Write a failing test
Make the test pass
TDD
Refactor
n cycles

Gherkin


WHEN
(When an event occurs)
GIVEN
(Set of Pre-conditions)
BDD
THEN
(Outcome is achieved)

BDD

1. Write story

Plain text

```
Scenario: A trader is alerted of status

Given a stock and a threshold of 15.0
When stock is traded at 5.0
Then the alert status should be OFF
When stock is traded at 16.0
Then the alert status should be ON
```

2. Map steps to Java

POJO

```
public class TraderSteps {
    private TradingService service;   // Injected
    private Stock stock; // Created

    @Given("a stock and a threshold of $threshold")
    public void aStock(double threshold) {
        stock = service.newStock("STK", threshold);
    }
    @When("the stock is traded at price $price")
    public void theStockIsTraded(double price) {
        stock.tradeAt(price);
    }
    @Then("the alert status is $status")
    public void theAlertStatusIs(String status) {
        assertThat(stock.getStatus().name(), equalTo(status));
    }
}
```

Cucumber

https://cucumber.io/

Gherkin

```
Funcionalidade: Transferir valores entre contas
   Validar a transferência de valores entre contas

   Cenário: Transferir valores com sucesso entre duas contas
      Dada conta "<origem>" com saldo <saldoO> e a conta "<destino>" com saldo <saldoD>
      Quando a conta "<origem>" transferir <valor> para a conta "<destino>"
      Então o saldo da conta "<origem>" deve ser <resultO> e a conta "<destino>"
      <resultD>

Exemplos:
    | origem | destino | valor  | saldoO  | saldoD | resultO | resultD |
    | 123456 | 654321  | 100.0  | 5000.0  | 5000.0 | 4900.0  | 5100.0  |
    | 987654 | 321654  | 1000.0 | 2000.0  | 100.0  | 1000.0  | 1100.0  |
    | 987654 | 321654  | 2500.0 | 20000.0 | 0.0    | 17500.0 | 2500.0  |
```

Typescript

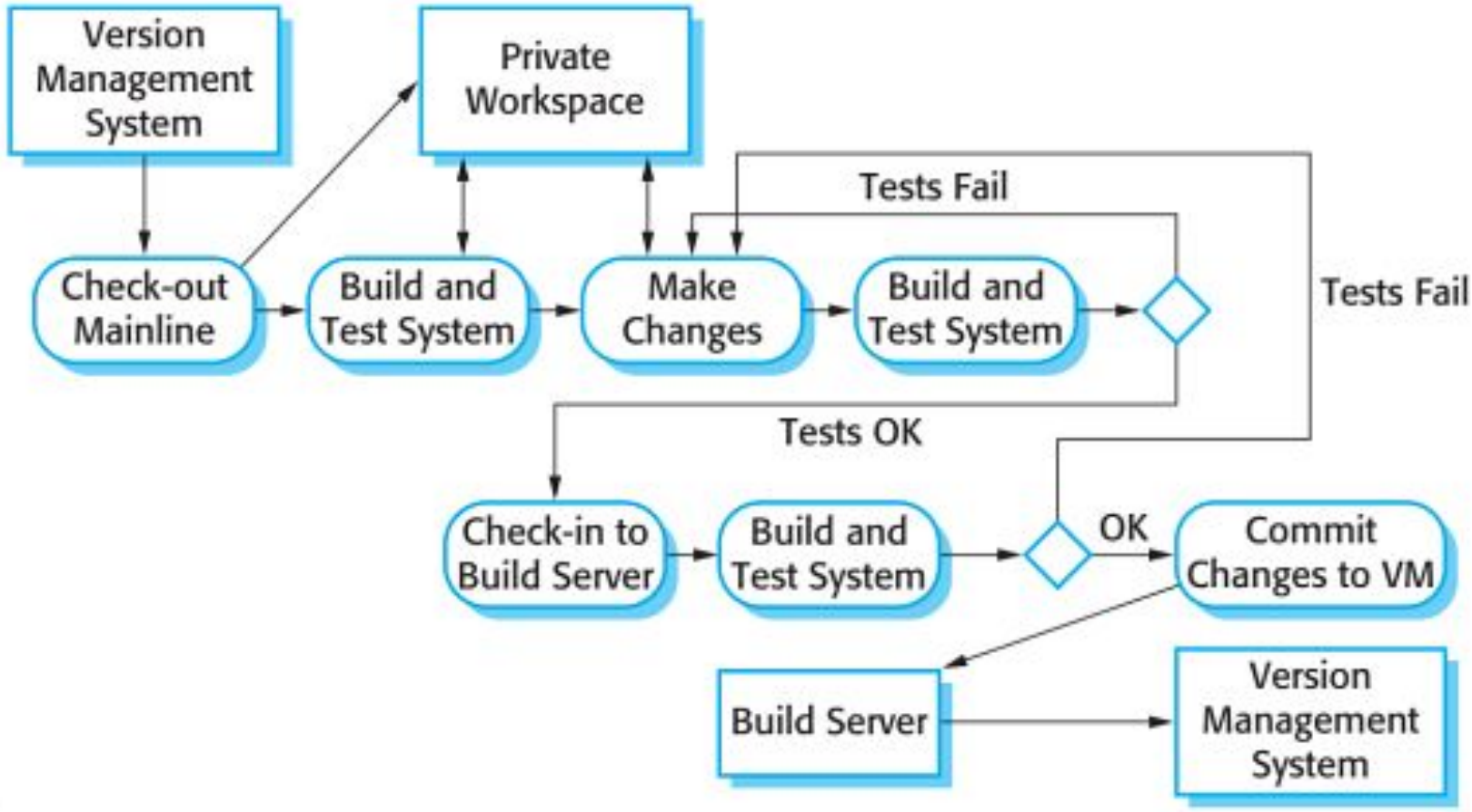
```
@binding()
class TransferenciaServicoStep {

    @given("conta {string} com saldo {float} e a conta {string} com saldo {float}")
    public dada(origem: string, saldoO: number, destino: string, saldoD: number): void {
        ...
    }

    @when("a conta {string} transferir {float} para a conta {string}")
    public quando(origem: string, valor: number, destino: string): void {
        ...
    }

    @then("o saldo da conta {string} deve ser {float} e a conta {string} {float}")
    public entao(origem: string, resO: number, destino: string, resD: number): void {
        ...
    }
}
```

Testando com BDD

https://gist.github.com/joaovictorino/bd4db6b5c2a15eea8675099b343e7c62

(20 minutos)

Integração contínua



Integração contínua

- Cada nova versão submetida ao gerenciador de versão iniciará uma bateria de testes
- Excluir elementos de infraestrutura
- Testes rápidos de serem executados
  - Testes de unidade
  - Testes de integração

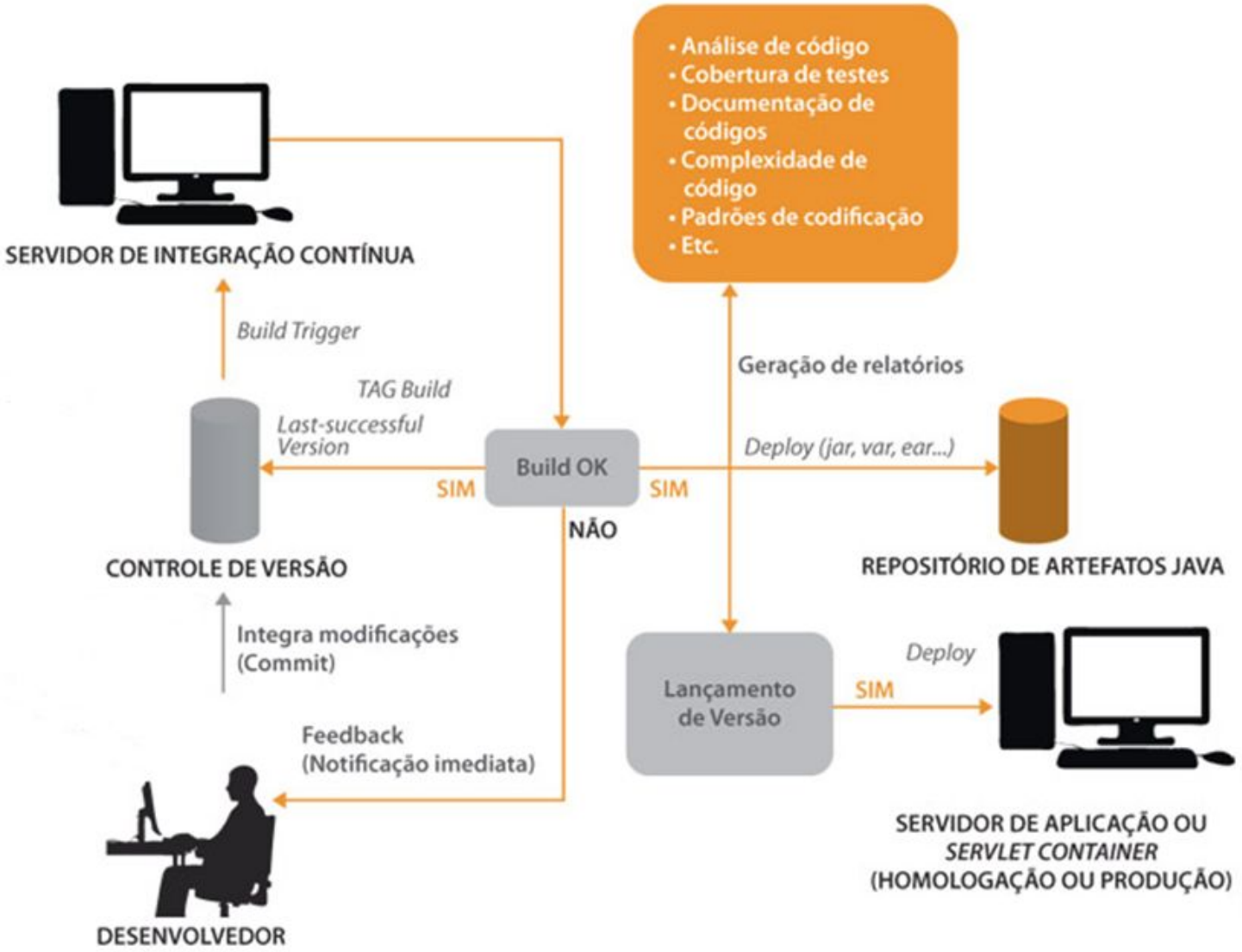
Testes de fumaça (smoke tests)

Subindo o código para o GitHub

https://gist.github.com/joaovictorino/da7cf1cf8cc42f76188525372c43ae9e

(10 minutos)

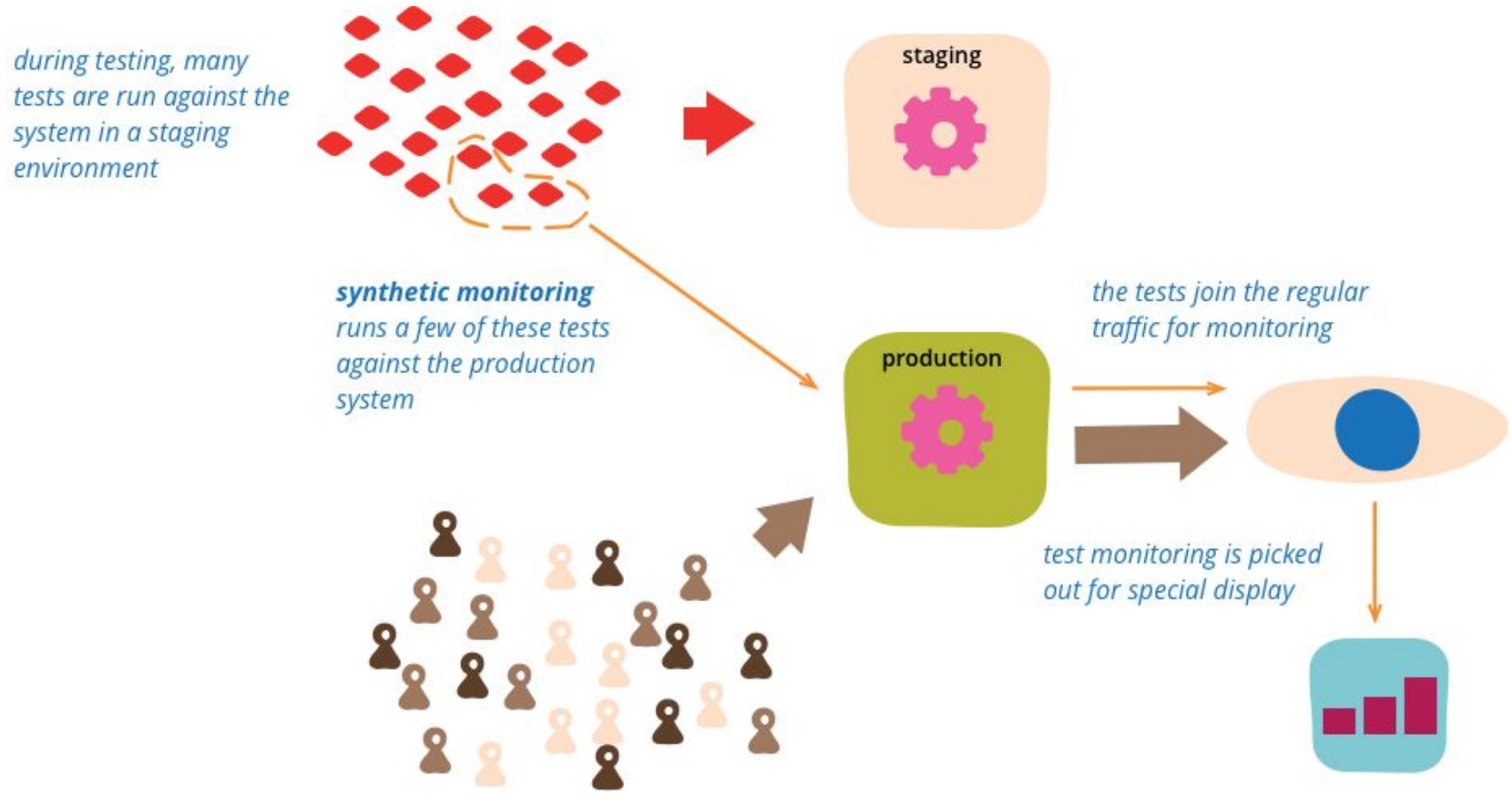
Entrega contínua



Entrega contínua

- Pode ser executado em período agendado, builds noturnos
- Testes mais demorados e completos
  - Testes de unidade
  - Testes de integração
  - Testes de aceitação
  - Testes de sistema

Monitoramento/testes sintéticos



https://martinfowler.com/bliki/SyntheticMonitoring.html

Jenkins

https://www.jenkins.io/

Subindo o Jenkins com Docker

https://gist.github.com/joaovictorino/cc2a8531dc7dbc24a3d668e08c5967ba

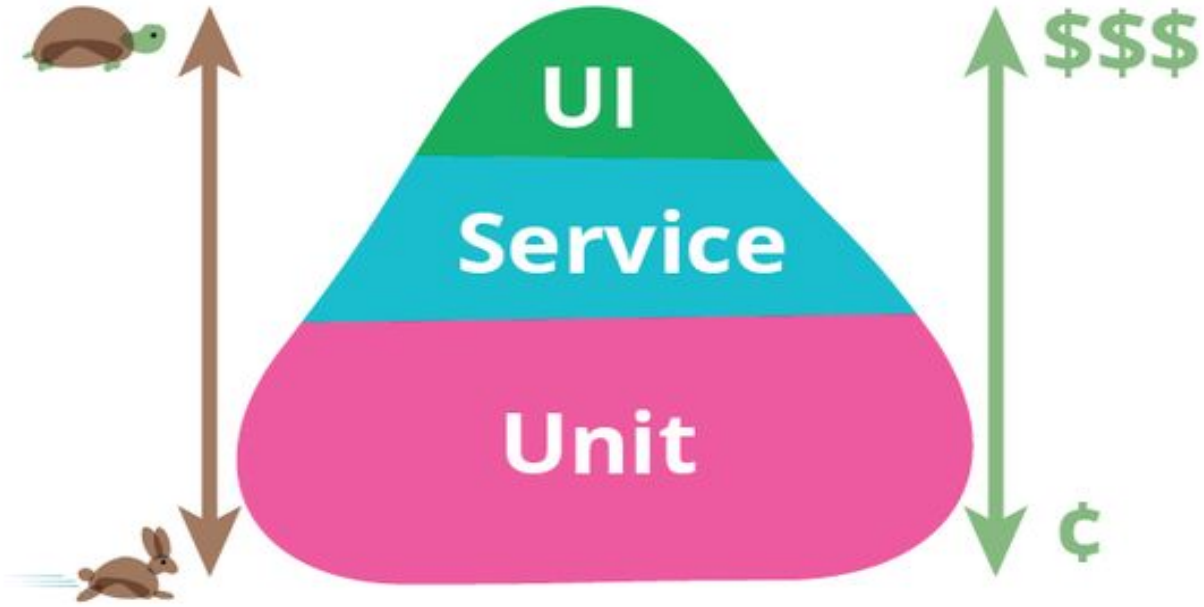
(10 minutos)

Executando os testes no Jenkins

https://gist.github.com/joaovictorino/b406ff57b0cb688453f0220438dcae03

(10 minutos)

Pirâmide de testes



Faculdade IMPACTA

BOA NOITE!

Referência no ensino de Tecnologia em **Graduação**, **Pós**, **EAD**, **MBA**, **cursos livres** e **treinamentos empresariais**. Lidera iniciativas em Inteligência Artificial, Ciência de Dados, Robótica, Engenharia da Computação, Big Data, UX e Transformação Digital. Criou a incubadora Impacta Open Startup, exclusiva para seus estudantes. É campeã na formação de times para Hackathons, eventos de inovação patrocinados por gigantes: NASA, IBM, Deloitte, Shell, Santander, Itaú, Globo e Fiesp. É uma das marcas mais admiradas pela Comunidade Tech da América Latina. Atua no mercado desde 1988 e formou mais de 1 milhão de pessoas e certificou mais de 25 mil empresas de diversas áreas da Economia. Ver mais em: impacta.edu.br e impacta.com.br