O design de software morreu?

Eder Ignatowicz
Senior Architect, Dextra

There are known knowns; there are things we know that we know.
There are known unknowns; that is to say, there are things that we now know we don't know.
But there are also **unknown unknowns** – there are things **we do not know we don't know**. ”

United States Secretary of Defense, Donald Rumsfeld

@ederign

Software Craftsman @ Dextra

Instrutor Dextraining

Professor na Faccamp e Unisal

Editor líder na InfoQ Brasil

Pai da Dora <3

Galera Ponta Firme e que Manda bem

Dextra

Projetos Desafiadores

Qualidade de Vida

Ambiente de Melhoria Contínua


Green Field

Unit / Micro Testing

Unit / Micro

Testing

TDD

Unit / Micro

Testing

TDD

Refactoring

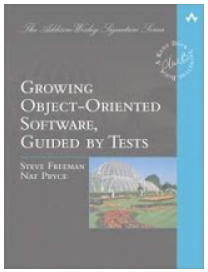
Top Coder

Project Euler

Dojos

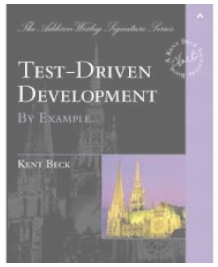
UVa

Katas



dev camp 2013

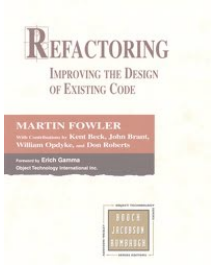
industrial logic



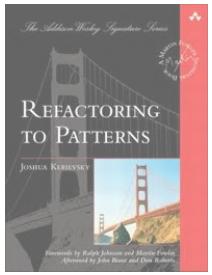
dex training

Estudo

code school
learn by doing



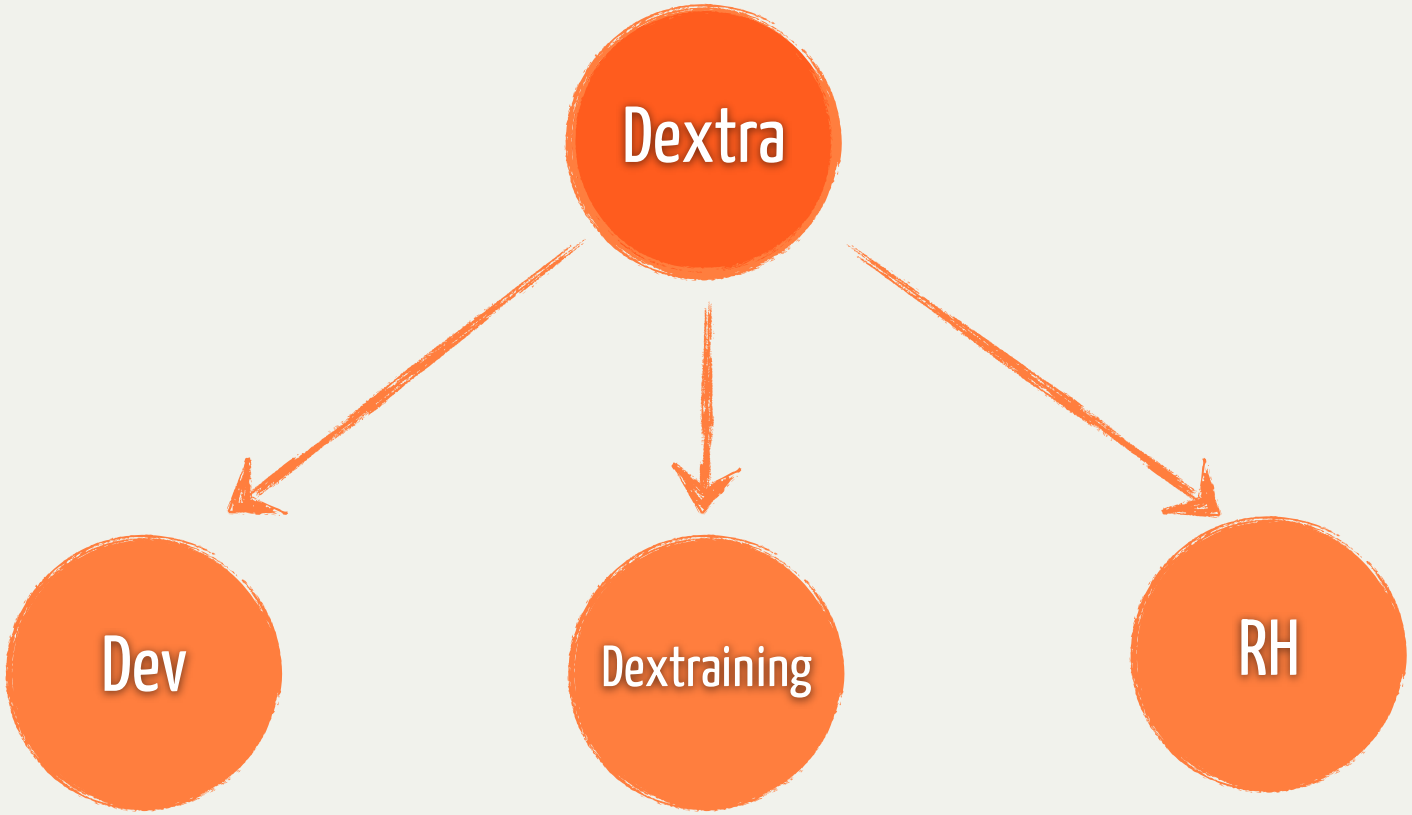
InfoQueue
Enterprise Software Development Community

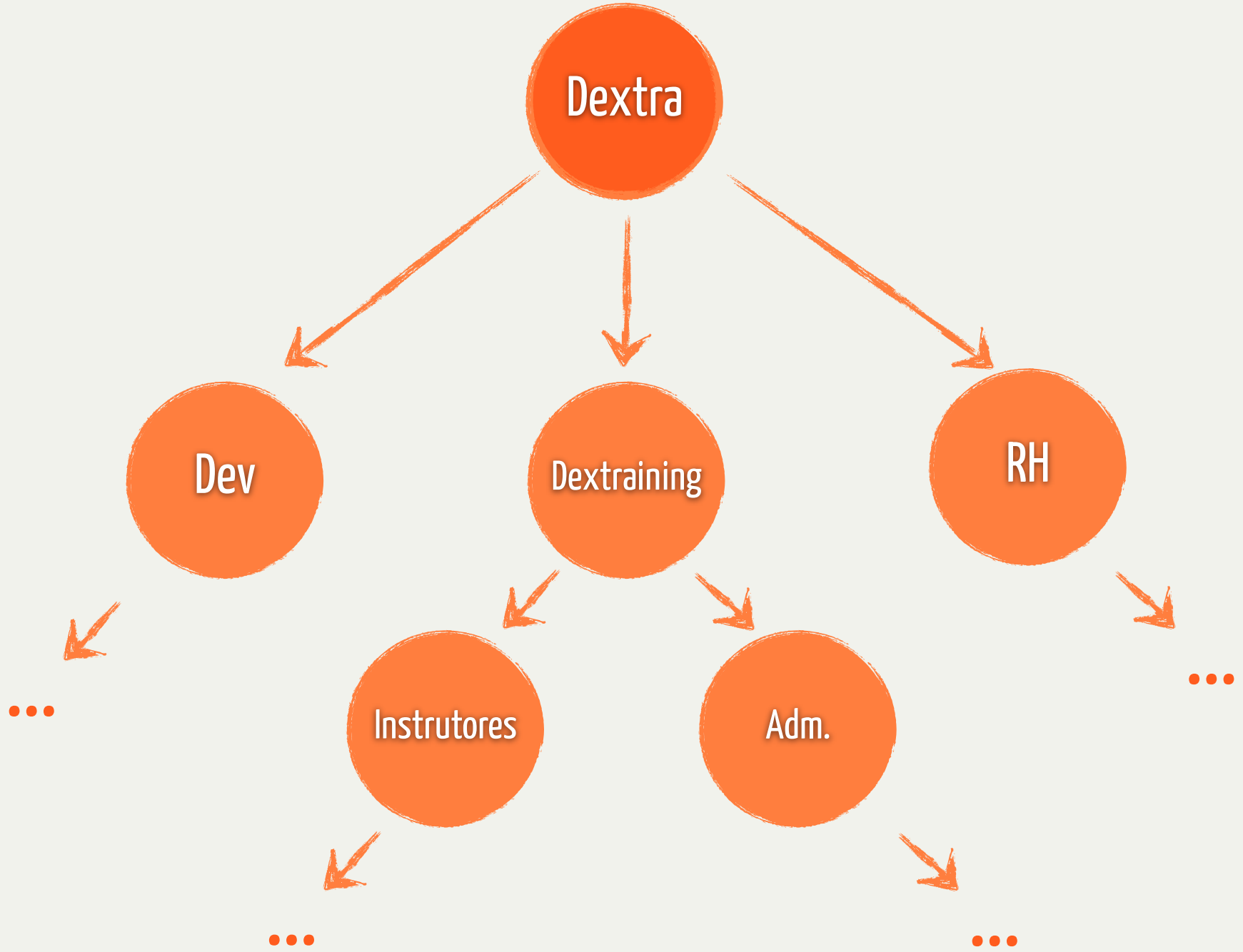


stackoverflow


MAKE GIFS AT GIFSOUP.C

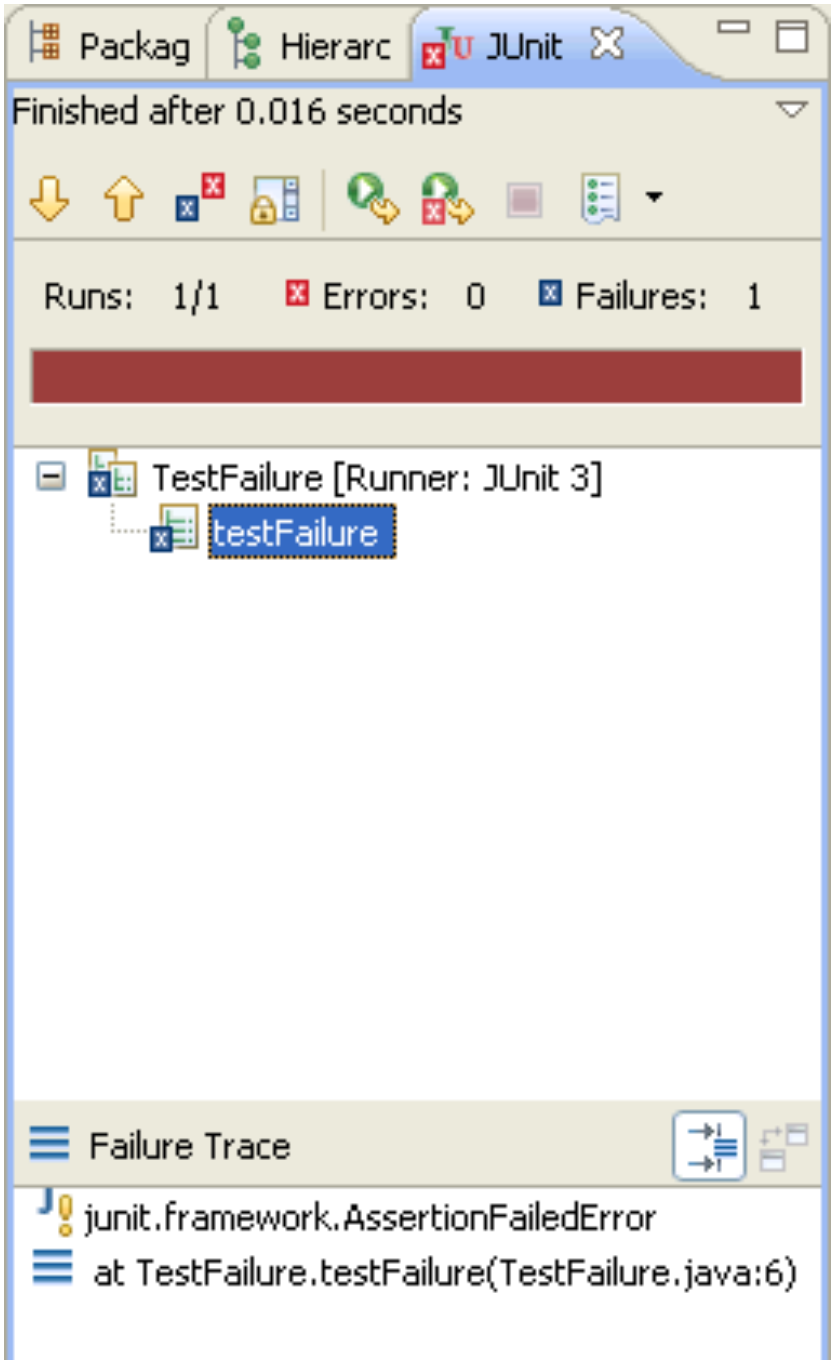
Sistema Legado


Dextra
Dev
Dextraining
RH


Dextra
Dev
Dextraining
RH
Instrutores
Adm.

TDD !!! <3

```
    @Before
     public void setup() {
          avo = CentroCustoTestHelper.geraCentroCusto("Avo", null);
          pai = CentroCustoTestHelper.geraCentroCusto("Pai", avo);
          filho1 = CentroCustoTestHelper.geraCentroCusto("Filho 1", pai);
          filho12 = CentroCustoTestHelper.geraCentroCusto("Filho 12", filho1);
          filho121 =CentroCustoTestHelper.geraCentroCusto("Filho
121",filho12);
     }
     @Test
     public void getFilhos() {
          assertEquals(filho1.getFilhos().size(), 2);
          assertEquals(avo.getFilhos().size(), 4);
     }
```


Packag
Hierarc
JUnit
Finished after 0.016 seconds
Runs: 1/1
Errors: 0
Failures: 1
TestFailure [Runner: JUnit 3]
testFailure
Failure Trace
junit.framework.AssertionFailedError
at TestFailure.testFailure(TestFailure.java:6)

```
public List<CentroCusto> getFilhos() {
		Usuario usuario = webService.getUsuarioLogado();
		if (usuario.isPj()) {
			UsuarioPj usuarioPj = (UsuarioPj) usuario;

			if (usuarioPj.podeAcessar(this)) {
				return getFilhosNaoExcluidos();
			} else {
				List<CentroCusto> filhosVisiveis = new
ArrayList<CentroCusto>(getFilhosNaoExcluidos());

				Iterator<CentroCusto> i = filhosVisiveis.iterator();
				while (i.hasNext()) {
					CentroCusto filho = i.next();
					if ((!usuarioPj.podeAcessar(filho)) || (filho.estaExcluido())) {
						i.remove();
					}
				}

				return filhosVisiveis;
			}
		} else {
			return getFilhosNaoExcluidos();
		}
	}
```

WTF ?!?

```
public List<CentroCusto> getFilhos() {
		Usuario usuario = webService.getUsuarioLogado();
		if (usuario.isPj()) {
			UsuarioPj usuarioPj = (UsuarioPj) usuario;

			if (usuarioPj.podeAcessar(this)) {
				return getFilhosNaoExcluidos();
			} else {
				List<CentroCusto> filhosVisiveis = new
ArrayList<CentroCusto>(getFilhosNaoExcluidos());

				Iterator<CentroCusto> i = filhosVisiveis.iterator();
				while (i.hasNext()) {
					CentroCusto filho = i.next();
					if ((!usuarioPj.podeAcessar(filho)) || (filho.estaExcluido())) {
						i.remove();
					}
				}

				return filhosVisiveis;
			}
		} else {
			return getFilhosNaoExcluidos();
		}
	}
```


```
public List<CentroCusto> getFilhos() {
        Usuario usuario = webService.getUsuarioLogado();
        if (usuario.isPj()) {
            UsuarioPj usuarioPj = (UsuarioPj) usuario;

            if (usuarioPj.podeAcessar(this)) {
                return getFilhosNaoExcluidos();
            } else {
                List<CentroCusto> filhosVisiveis = new
ArrayList<CentroCusto>(getFilhosNaoExcluidos());

                Iterator<CentroCusto> i = filhosVisiveis.iterator();
                while (i.hasNext()) {
                    CentroCusto filho = i.next();
                    if ((!usuarioPj.podeAcessar(filho)) || (filho.estaExcluido())) {
                        i.remove();
                    }
                }

                return filhosVisiveis;
            }
        } else {
            return getFilhosNaoExcluidos();
        }
    }
```


Brownfield

O design de software morreu? aka "Agile matou o Design?"

Eder Ignatowicz
Senior Architect, Dextra

Big Design Upfront

Waterfall

Junte um grupo de pessoas

Junte um grupo de pessoas

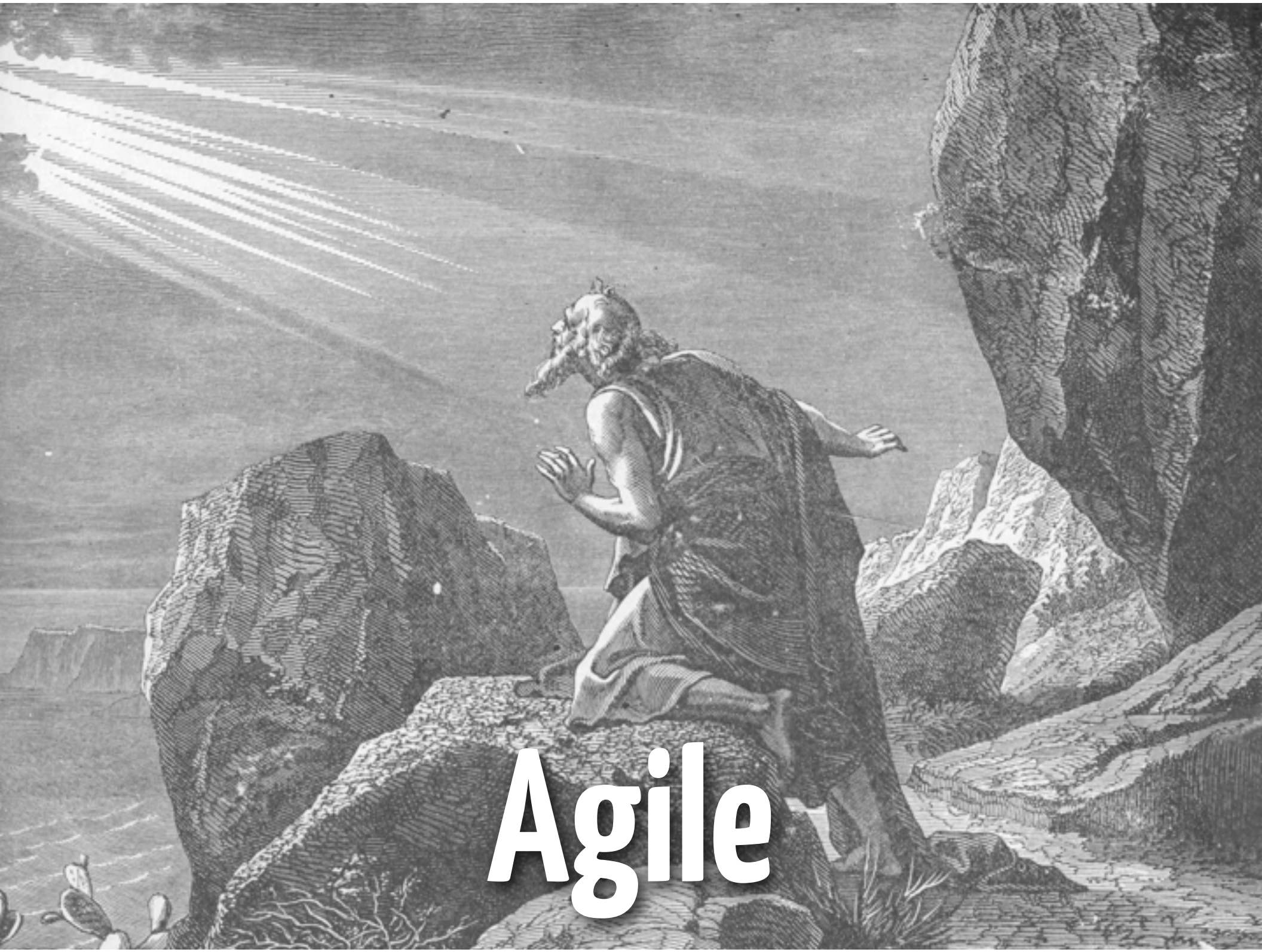
Discutam Hipóteses

Junte um grupo de pessoas

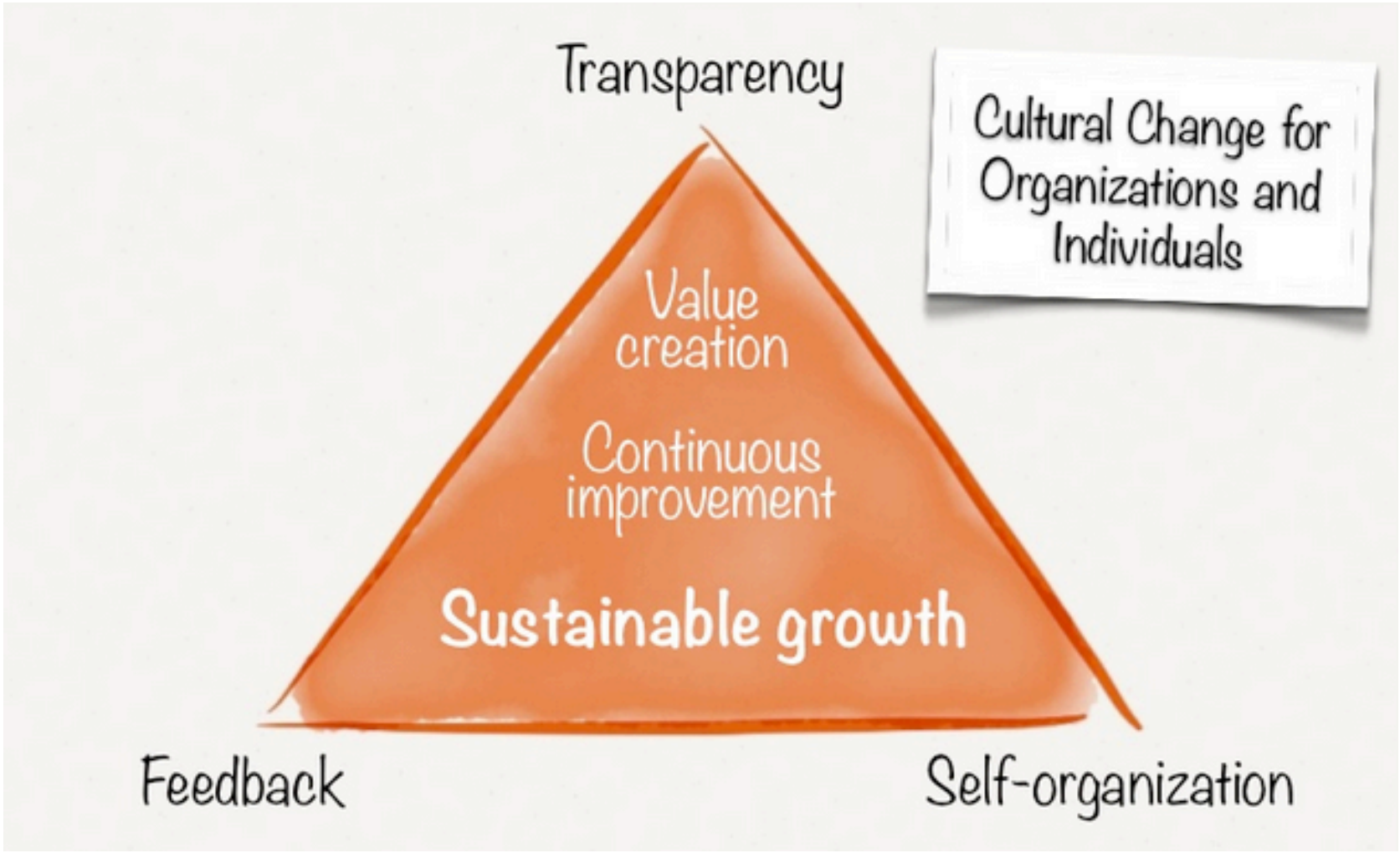
Design!!!

Discutam Hipóteses

“Não existem Requisitos de Software, mas sim hipóteses.”


Agile

Agile Triangle



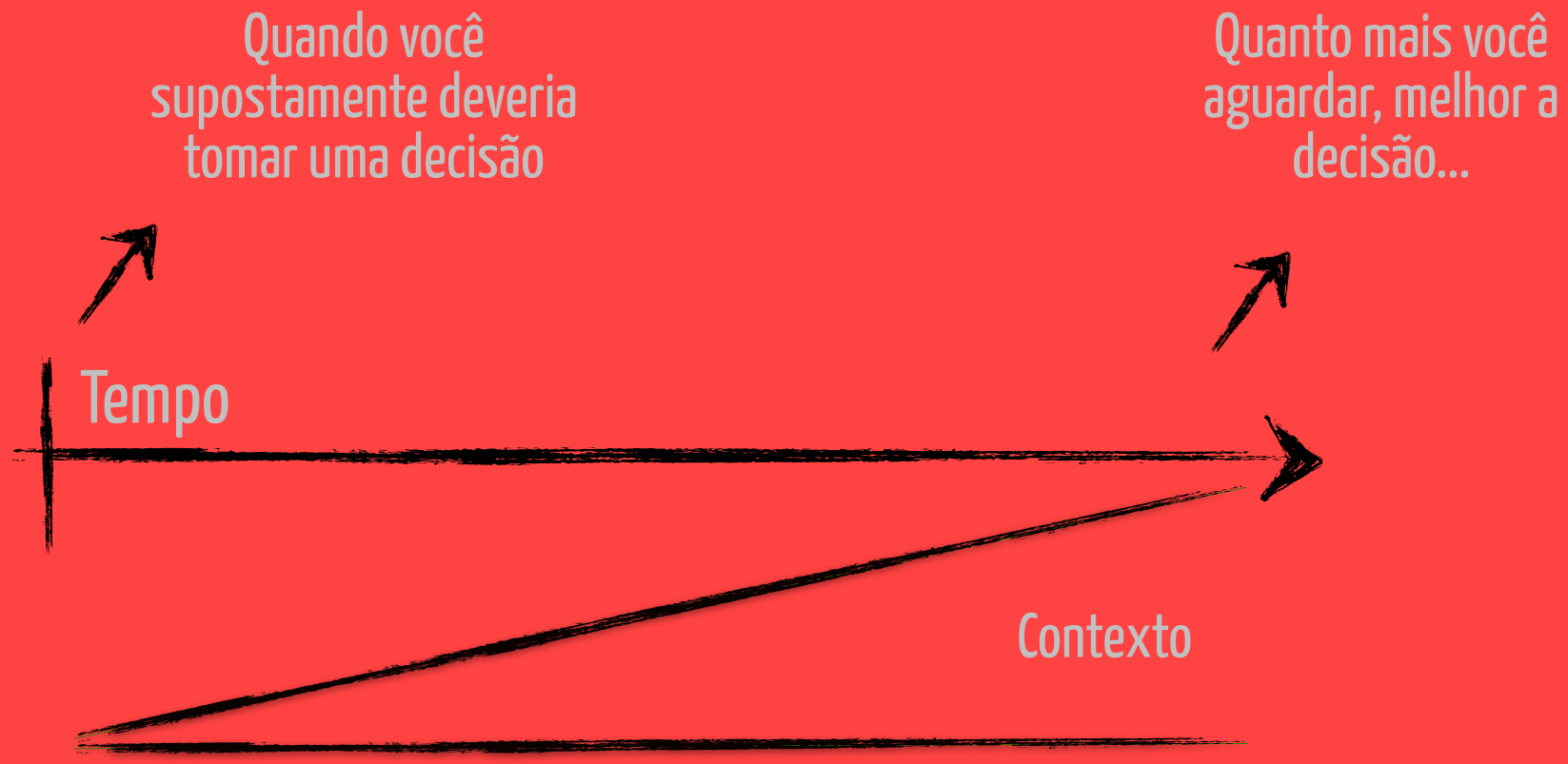
Responder rapidamente ao negócio

Entrega sustentável

Agile

Last Responsible Moment

Last Responsible Moment



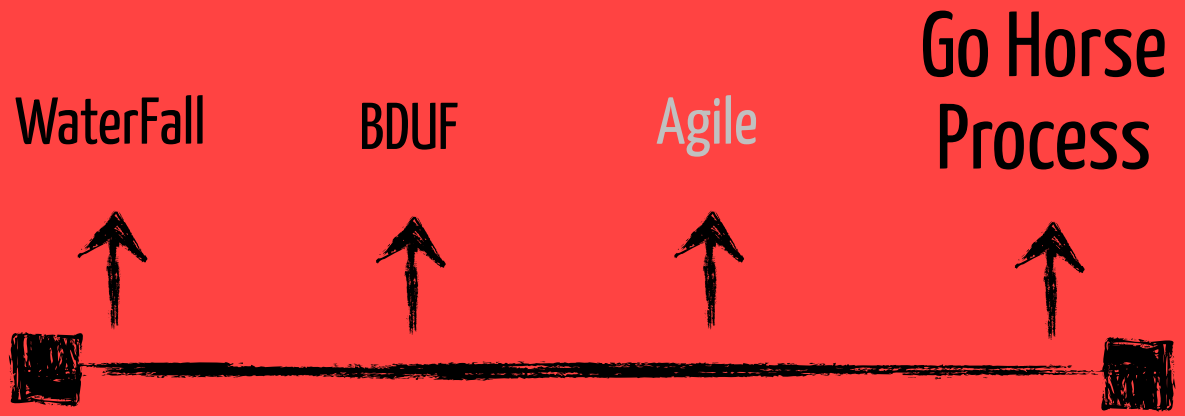
Geração de Valor

Lean Equilíbrio

Engenharia de Software Ágil


Mainstream

Espectro do Design



Go Horse Process

“Vamos deixar deste jeito que funciona, depois refatoramos e as estruturas e os patterns emergirão”

Dívida Técnica

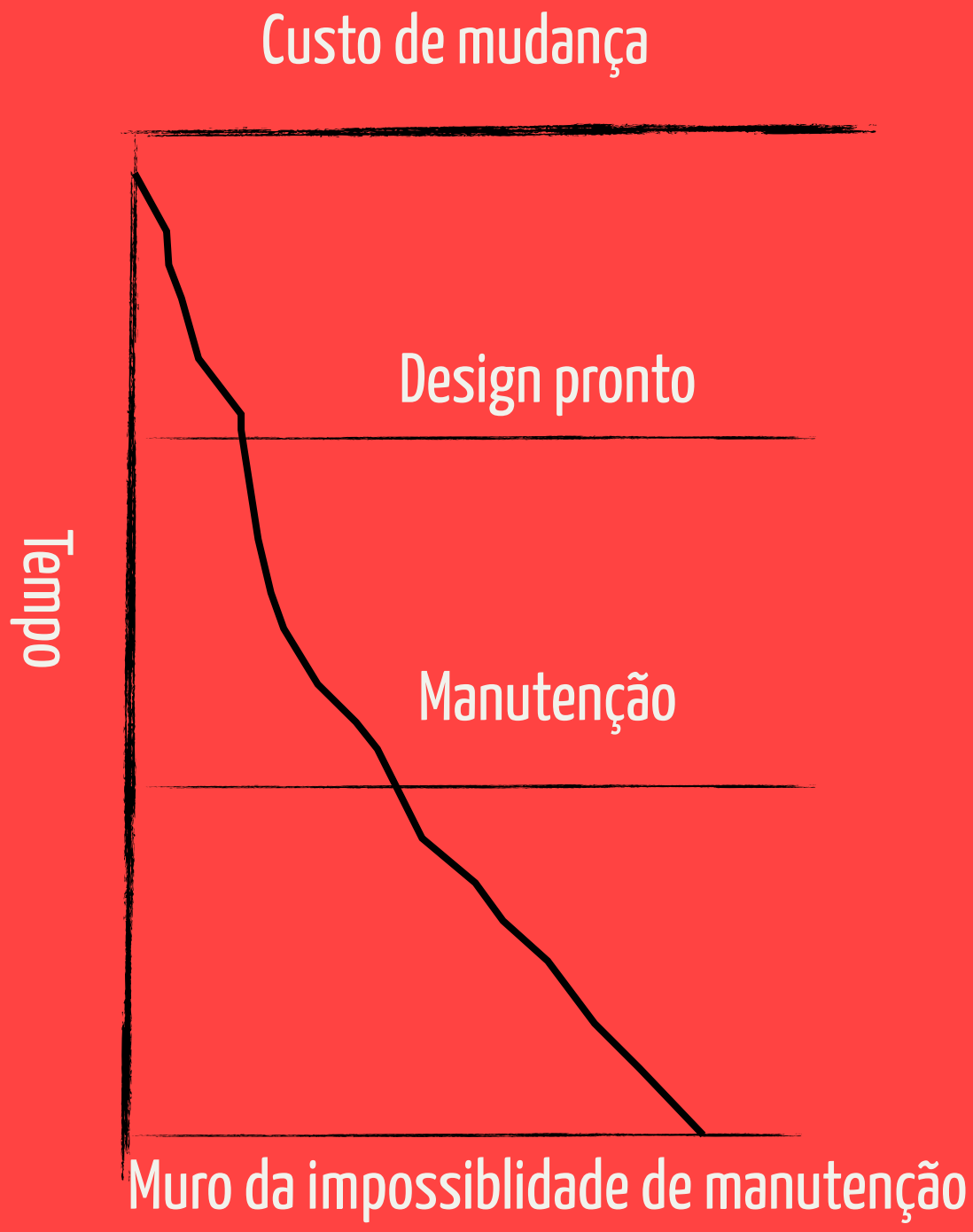
Software Apodrece...

“

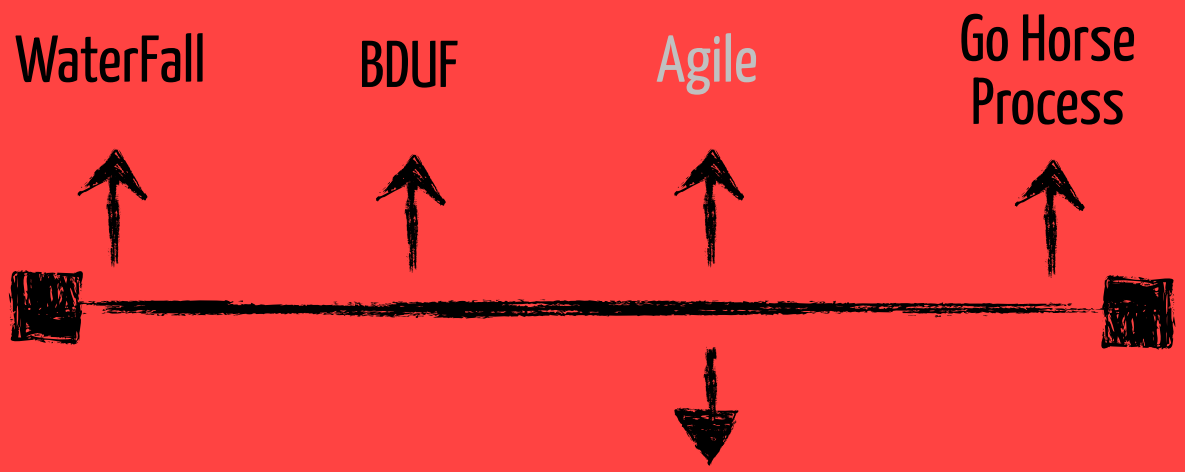
Se você desenvolve iterativamente e não aplica práticas de engenharia ágil, sua base de código morrerá em 2 ou 3 anos...
Não se preocupar com o design em um processo iterativo torna o projeto insustentável

”

James Shore


Custo de mudança
Design pronto
Manutenção
Tempo
Muro da impossiblidade de manutenção

Espectro do Design



Evolutionary Design

Refactoring

Testing

Evolutionar y Design

Continuos Integration

Refactoring

Encapsulate Field

Replace Conditional with polymorphism

Extract Method

Move Method

Refactoring

State/Strategy

Rename Method

Generalize Type

Pull Up / Push Down

```
public List<CentroCusto> getSubCentrosDeCusto() {
        Usuario usuario = getUsuarioLogado();
        if (usuario.isPj()) {
                return getSubCentrosDeCustoUsuarioPJ(usuario);
        } else {
                return getTodosSubCentrosDeCusto();
        }
 }

 private List<CentroCusto> getSubCentrosDeCustoUsuarioPJ(Usuario usuario) {
        if (usuario.podeAcessar(this)) {
                return getTodosSubCentrosDeCusto();
        } else {
                return getSubCentrosDeCustoVisiveisUsuario(usuario);
        }
 }
```

Domain Driven Design

Code == Design

“Refactoring is not the same activity as redesign, where the programmers take a conscious decision to change a large-scale structure. That said, having taken a redesign decision, a team can use a refactoring techniques to get to new design incrementally and safely.”

Growing Object Oriented Software Guided by Tests

MicroTesting

```
    @Before
     public void setup() {
          avo = CentroCustoTestHelper.geraCentroCusto("Avo", null);
          pai = CentroCustoTestHelper.geraCentroCusto("Pai", avo);
          filho1 = CentroCustoTestHelper.geraCentroCusto("Filho 1", pai);
          filho12 = CentroCustoTestHelper.geraCentroCusto("Filho 12", filho1);
          filho121 =CentroCustoTestHelper.geraCentroCusto("Filho
121",filho12);
     }
     @Test
     public void getFilhos() {
          assertEquals(filho1. getSubCentrosDeCusto().size(), 2);
          assertEquals(avo. getSubCentrosDeCusto().size(), 4);
     }
```

```
public List<CentroCusto> getSubCentrosDeCusto() {
        Usuario usuario = webService.getUsuarioLogado();
        if (usuario.isPj()) {
            return getSubCentrosDeCustoUsuarioPJ(usuario);
        } else {
            return getTodosSubCentrosDeCusto();
        }
    }

    private List<CentroCusto> getSubCentrosDeCustoUsuarioPJ(Usuario usuario) {
        if (usuario.podeAcessar(this)) {
            return getTodosSubCentrosDeCusto();
        } else {
            return getSubCentrosDeCustoVisiveisUsuario(usuario);
        }
    }
```

Integração Web Service Legado

```
public List<CentroCusto> getSubCentrosDeCusto() {
        Usuario usuario = getUsuarioLogado();
        if (usuario.isPj()) {
            return getSubCentrosDeCustoUsuarioPJ(usuario);
        } else {
            return getTodosSubCentrosDeCusto();
        }
    }

    private List<CentroCusto> getSubCentrosDeCustoUsuarioPJ(Usuario usuario) {
        if (usuario.podeAcessar(this)) {
            return getTodosSubCentrosDeCusto();
        } else {
            return getSubCentrosDeCustoVisiveisUsuario(usuario);
        }
    }
```

```
public class CentroCustoTestHelper {
    public static class CentroCustoFake extends CentroCusto {
        public CentroCustoFake(String nome, Produto produto) {
            super(nome, produto);
        }
        @Override
        public Usuario getUsuarioLogado() {
            return new UsuarioPj("dorinha@ederig.com");
        }
    }
    public static CentroCusto geraCentroCusto(String nome, CentroCusto pai) {
        CentroCusto centroCustoFilho = new CentroCustoFake(nome, null);
        if (pai != null) {
            pai.adicionarFilho(centroCustoFilho);
            centroCustoFilho.setPai(pai);
        }
        return centroCustoFilho;
    }
}
```

Sobrecarga!

“Things like TDD and Continuous Integration have fundamentally changed the safety in development.”

Joshua Kerievsky

TDD é difícil em sistemas legados

Your micro tests TALKS!

Design bad smells nos testes unitários...

Você quer testar um método privado

Talvez sua classe tenha muitas repsonsabilidades

Uma mudança no código quebra muitos testes

Violações do Open/Closed Principle

Instanciar uma classe necessita instanciar 20 classes

Alto acoplamento

Testar é complicado por causa do framework X

Baixa separação da lógica do domínio

Você quer utilizar um framework de Mock

Violações da Lei de Deméter

Difícil Mockar uma classe

Abstração fraca

É difícil chamar um método ou instanciar uma classe

Muitas responsabilidades em uma classe

Muitos asserts para um método

Violações do SRP, método muito longo

Como praticar design evolutivo?

Como criar sistemas resistentes a mudanças?

Como criar sistemas capazes de
acomodar constantes e contínuas mudanças?

Como prover crescimento sustentável?

Testing/CI

Refactoring

Equilíbrio

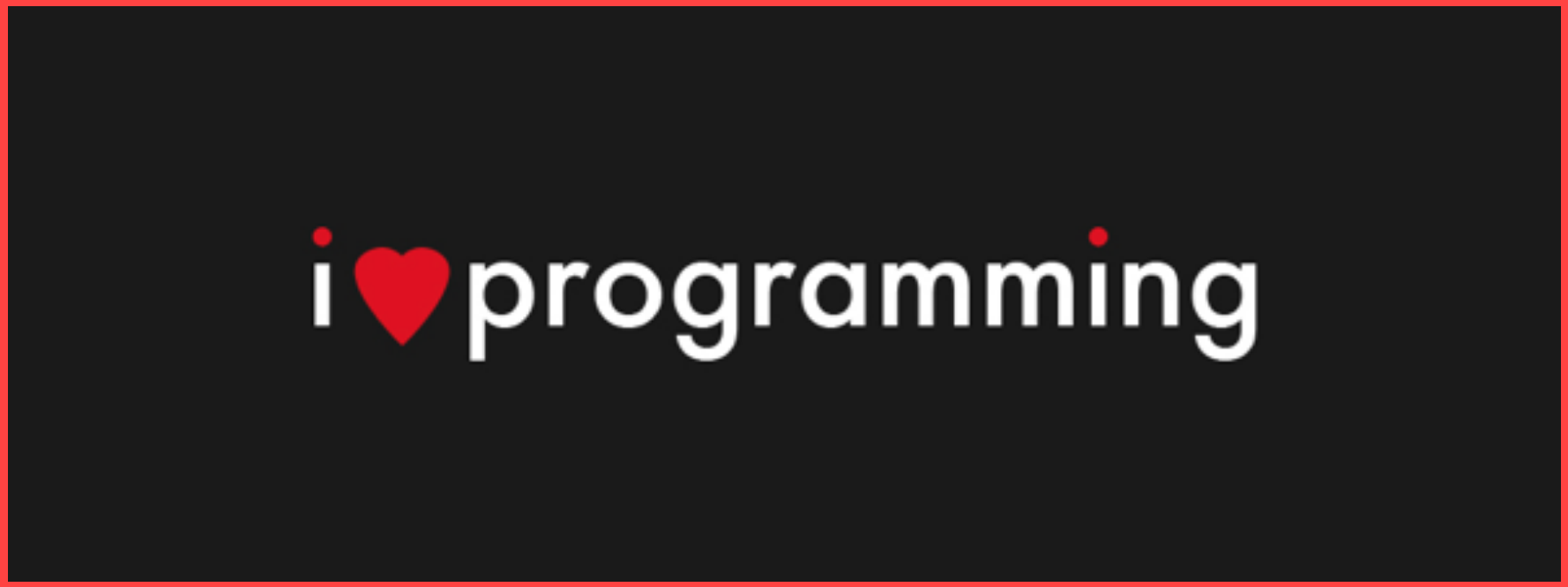
Geração de Valor

Esqueça o barato,
Esqueça os recursos

Você colhe o que planta


Aumente o SEU sarrafo

Conviva com os melhores



Viva com os apaixonados, os craftsman.

Viva uma cultura de
Aprendizado Colaborativo

Porque sem Refactoring, Testes e CI

Não existe design evolutivo

E sem design evolutivo

Não existe Desenvolvimento Ágil

@ederign

Software Craftsman @ Dextra
(Arquitetura, NoSQL, Devops, QA)

Instrutor Dextraining

Professor na Faccamp e Unisal

Editor líder na InfoQ Brasil