



# Engenharia de Software (BCC35E)



Igor Scaliante Wiese



@IgorWiese



igor.wiese@gmail.com | igor@utfpr.edu.br |

igorwiese.com

# Licença do material

---

Este Trabalho foi licenciado com uma Licença



**Atribuição-NãoComercial-  
Compartilha Igual 4.0 Internacional  
(CC BY-NC-SA 4.0)**

Mais informações visite

[https://creativecommons.org/licenses/by-nc-sa/4.0/deed.pt\\_BR](https://creativecommons.org/licenses/by-nc-sa/4.0/deed.pt_BR)

# ENGENHARIA DE SOFTWARE MODERNA

Princípios e práticas para  
desenvolvimento de software  
com produtividade



MARCO TULIO VALENTE



## Vinicius Garcia (He/Him) · 1º

Associate Professor at Centro de Informática (UFPE), Associate  
Research Fellow at SoftexRecife, Timbaleiro e Filho de Gandhi

Fala sobre #data, #vscode e #freecodecamp

Recife, Pernambuco, Brasil · [Informações de contato](#)



Universidade Federal de  
Pernambuco



Fast MBA

---

# Processos

Missão 02

# Poor President Obama...

**HealthCare.gov** Learn Get Insurance Log in Español

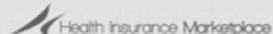
Individuals & Families Small Businesses All Topics  Search

## The System is down at the moment.

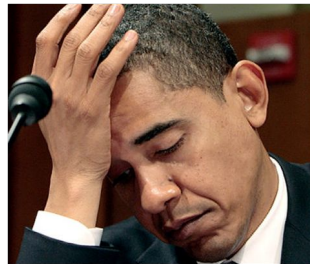
We're working to resolve the issue as soon as possible. Please try again later.

Please include the reference ID below if you wish to contact us at 1-800-318-2596 for support.

Error from: [https://www.healthcare.gov/marketplace/global/en\\_US/registration%23signUpStepOne](https://www.healthcare.gov/marketplace/global/en_US/registration%23signUpStepOne)  
Reference ID: 0.cdd73b17.1380636213.edae7e9

 **181** DAYS LEFT TO ENROLL

<b>OCT</b> <b>1</b>	Open Enrollment Begins	<b>JAN</b> <b>1</b>	Coverage Can Begin	<b>MAR</b> <b>31</b>	Open Enrollment Closes
------------------------	------------------------	------------------------	--------------------	-------------------------	------------------------



1. sprints + iterations ≠ agile  
healthcare.gov nunca foi testado de fato
2. um 'sistema' RESULTA de um 'SISTEMA'  
{não havia como 'parar' healthcare.gov...}
3. TESTE tem que ser parte da ENTREGA  
e não apagar incêndios depois da 'entrega'
4. de nada adianta INFORMATIZAR O CAOS  
se os processos não estão bem definidos...
5. não há sistema PERFEITO; nem POR LEI...  
governos tendem a insistir nisso; **#FAIL!**...
6. sistemas -online, high profile- são ALVOS  
healthcare.gov nunca foi olhado assim

# ~1968: trazer disciplina de engenharia para software

---

- Por que a construção de software não é como construir uma ponte? Qualidade e custo previsíveis e seguros...
  - (embora 90% dos projetos de infraestrutura estejam atrasados/acima do \$)
- “Plan-and-Document”
  - Antes da codificação, o gerente de projeto faz o planejamento
  - Escreve documentação detalhando todas as fases do plano
  - O progresso é medido em relação ao plano
  - As mudanças no projeto devem ser refletidas na documentação e possivelmente para (re)planejar



# 1º Processo de Desenvolvimento: Waterfall (1970), 5 fases

---

- 5 fases do ciclo de vida Waterfall ou Processo de Desenvolvimento em Cascata (A.K.A. “Big Design Up Front” ou BDUF)
  - Análise e definição de requisitos
  - Projeto de sistema e software
  - Implementação e teste de unidade
  - Integração e teste de sistema
  - Operação e manutenção
- Primeiro modelo a organizar as atividades de desenvolvimento
- Uma fase tem de estar completa antes de passar para a próxima.
- Saídas das fases são acordadas contratualmente!
- Todas as fases envolvem atividades de validação





# Com o que cascata funciona bem?

---

- Especificações que não mudam: ônibus espacial da NASA, controle de aeronaves...
- Muitas vezes, quando o cliente vê, quer mudanças
- Muitas vezes, depois de construído a primeira vez, os desenvolvedores aprendem o caminho que eles deveriam ter seguido



# Essa foi uma abordagem bem-sucedida para o desenvolvimento de SW?

- ~420KLOC,  $\leq 1$  erro nas últimas 3 versões
- 260 SW engineers trabalhando para 1 das 4 fábricas de SW para receber o maior “maturity rating” do **SW Engineering Institute@CMU**
- Adesão rigorosa ao Cascata
  - ex: Mudança de 6 KLoC = 2500 pais de documentos



They Write the Right Stuff. FastCompany, 1996.  
Photo: Flickr user Matthew Simantov, CC-BY-SA

## Essa foi uma abordagem bem-sucedida para o desenvolvimento de SW?

---

- E os usuários exclamaram com uma risada e uma provocação: “**É exatamente o que pedimos, mas não o que queremos.**” —Anonymous
- O ponto forte do software é sua capacidade de evoluir - **sua capacidade de mudar e se adaptar** - mas isso não é adequado para abordagens de “*big design up front*” (BDUF) ou abordagens “*top down*”

# Problemas do modelo cascata

---

- **Particionamento** inflexível do projeto em estágios
  - Dificulta a resposta aos requisitos de mudança do cliente.
- Documentos “**completamente elaborados**” são necessários para fazer as transições entre estágios
- Adequado somente quando os requisitos são bem compreendidos e quando as **mudanças são raras**
  - Poucos sistemas de negócio têm requisitos **estáveis**.

# Como lidar com mudanças?

---

*“Planeje jogar [implementação] fora; porque você irá, de qualquer maneira.”*

Fred Brooks, Jr. (1999 Turing Award winner)

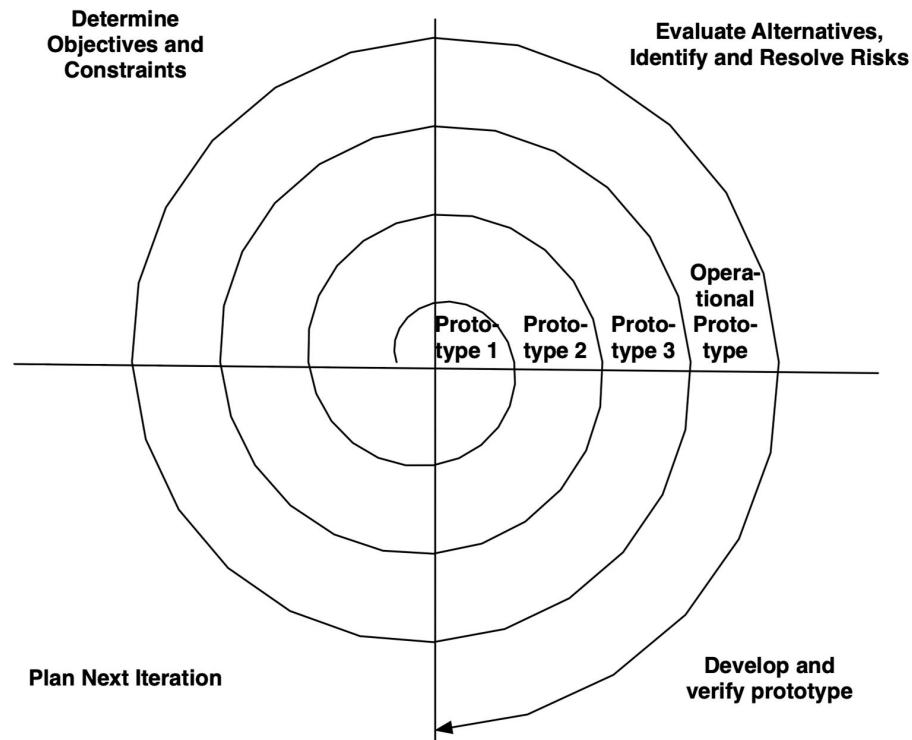
Muitas vezes, após a construção da primeira versão, os desenvolvedores aprendem a maneira correta que deveriam ter construído.



(Photo by Carola Lauber of SD&M [www.sdm.de](http://www.sdm.de). Used by permission under CC-BY-SA-3.0.)

# Ciclo de Vida Espiral (1986)

- Use protótipos para obter feedback do cliente até a versão "final" criada
  - cada "iteração" oferece um novo protótipo
  - iterações podem estar distantes no tempo
- Variante posterior: Rational Unified Process
  - sobrepõe os estágios de planejamento e execução de várias iterações



# Espiral Good & Bad

---

- Iterações envolvem o cliente antes do produto estar completo
- Reduz as chances de mal entendidos
- Gerenciamento de Riscos no ciclo de vida
- Monitoramento do projeto facilitado
- Cronograma e custo mais realista através do tempo



- Iterações longas de 6 a 24 meses
- Tempo para os clientes mudarem de ideia
- Muita documentação por iteração
- Muitas regras para seguir, pesado para todo projeto
- Custo alto do processo
- Complicado de atingir metas de investimento e marcos no cronograma





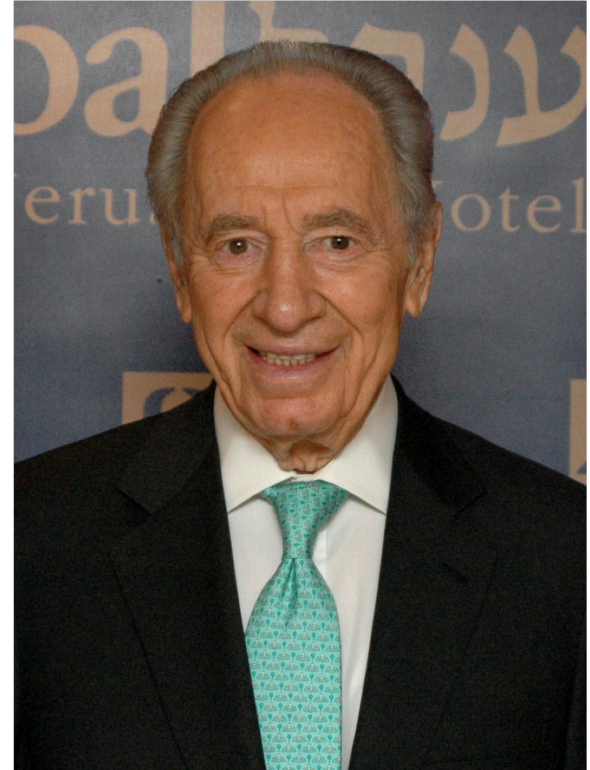
# Peres's Law

---

“Se um problema não tem solução, pode não ser um problema, mas um fato, que não deve ser resolvido, mas deve ser lidado com o tempo.”

— Shimon Peres

(vencedor do Prêmio Nobel da Paz de 1994)



(Photo Source: Michael Thaidigsmann, put in public domain,  
See [http://en.wikipedia.org/wiki/File:Shimon\\_peres\\_wjc\\_90126.jpg](http://en.wikipedia.org/wiki/File:Shimon_peres_wjc_90126.jpg))

## Processo Alternativo?



- P&D requer extensa documentação e planejamento
- P&D depende de um gerente experiente
- Podemos criar software efetivamente sem planejamento e documentação cuidadosos?
- Como evitar o “basta apenas cortar”?

# Agile Manifesto, 2001

---

- “We are uncovering better ways of developing SW by doing it and helping others do it.

Through this work we have come to value

- **Individuals and interactions** over processes & tools
  - **Working software** over comprehensive documentation
  - **Customer collaboration** over contract negotiation
  - **Responding to change** over following a plan
- That is, while there is value in the items on the right, we value the items on the left more.”

# Agile lifecycle

---

- Aceita as mudanças como um fato da vida: melhoria contínua vs. fases
- Os desenvolvedores refinam continuamente o protótipo em funcionamento, mas incompleto, até que os clientes fiquem satisfeitos, com feedback do cliente a cada **iterações** (1-2 semanas)
- **Todos os elementos do ciclo de vida em todas as iterações**
- O Agile enfatiza o **Test-Driven Development (TDD)** para reduzir erros, especificando **User Stories** para validar os requisitos do cliente, **Velocity** para medir o progresso

## Velocidade de um time



- Número de story points que o time consegue implementar em um sprint
- Definição de story points é "empírica"

# Story points



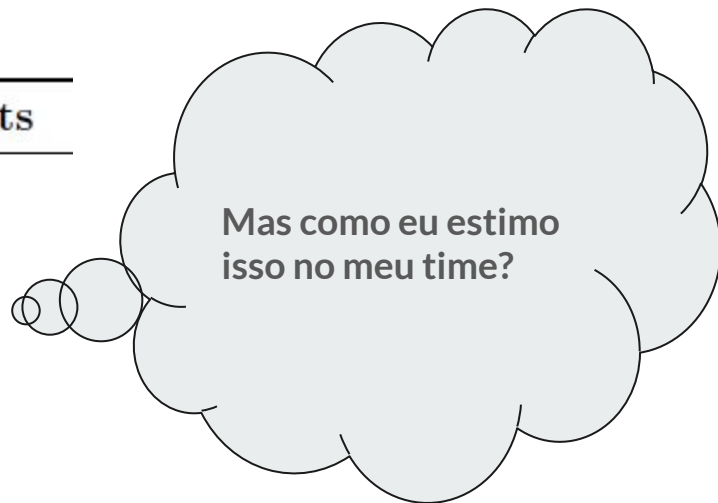
Unidade (inteiro) para comparação do tamanho de histórias. Exemplo de escala: 1, 2, 3, 5, 8, 13

História	Story Points
Cadastrar usuário	8
Postar perguntas	5
Postar respostas	3
Tela de abertura	5
Gamificar perguntas e respostas	5
Pesquisar perguntas e respostas	8
Adicionar tags em perguntas e respostas	5
Comentar perguntas e respostas	3

# Story points

Unidade (inteiro) para comparação do tamanho de histórias. Exemplo de escala: 1, 2, 3, 5, 8, 13

História	Story Points
Cadastrar usuário	8
Postar perguntas	5
Postar respostas	3
Tela de abertura	5
Gamificar perguntas e respostas	5
Pesquisar perguntas e respostas	8
Adicionar tags em perguntas e respostas	5
Comentar perguntas e respostas	3





## “Extreme Programming” (XP), uma versão do ciclo de vida ágil (Kent Beck et al.)

---

- If **short iterations are good**, make them **as short as possible** (weeks vs. years)
- If **simplicity is good**, always do the **simplest thing** that could possibly work
- If **testing is good**, test all the time. Write the test **just before** writing the code to be tested.
- If **code reviews are good**, **review code continuously**, by **programming in pairs**, taking turns looking over each others' shoulders.

### Gestão



Cliente  
presente



Time  
coeso



Ritmo  
sustentável



Posse  
coletiva

### Projeto



Metáfora  
de sistema



Projeto  
simples

### Planejamento



Jogo do  
Planejamento



Histórias de  
usuário



Testes de  
aceitação



Pequenas  
entregas



Spikes de  
Planejamento

### Codificação



Padrão de  
codificação



TDD



Integração  
contínua



Reunião diária  
em pé



Programação  
em par



Refatoração

### Valores



Comunicação



Simplicidade



Feedback



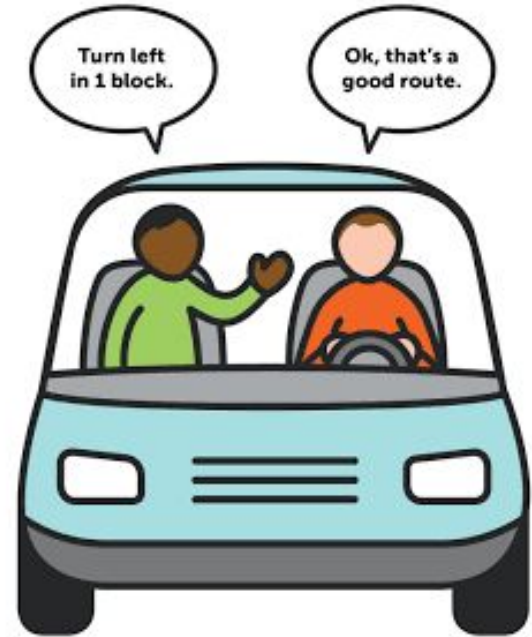
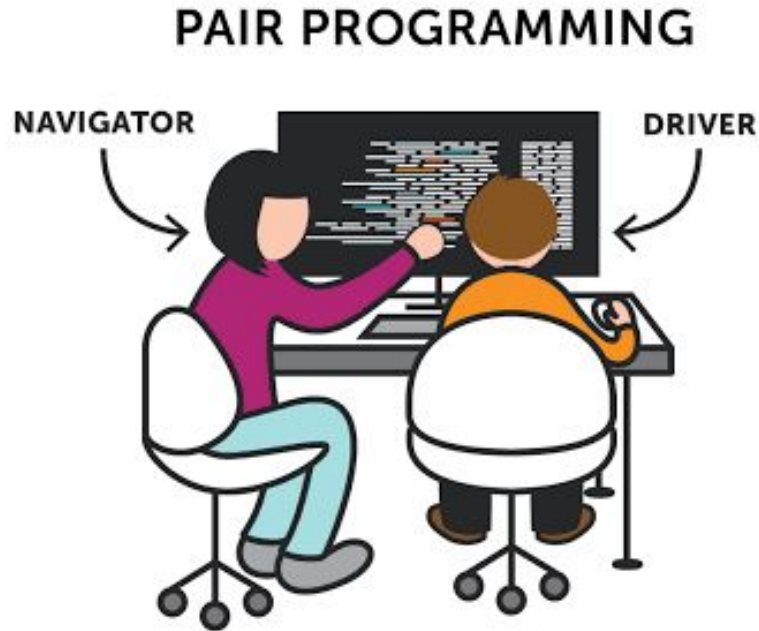
Coragem



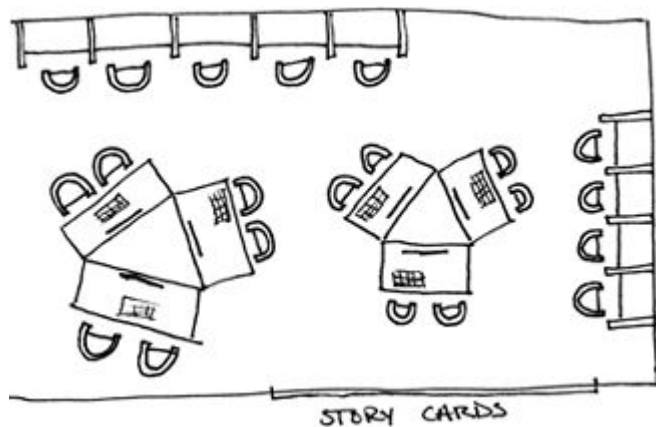
Respeito

# Pair Programming

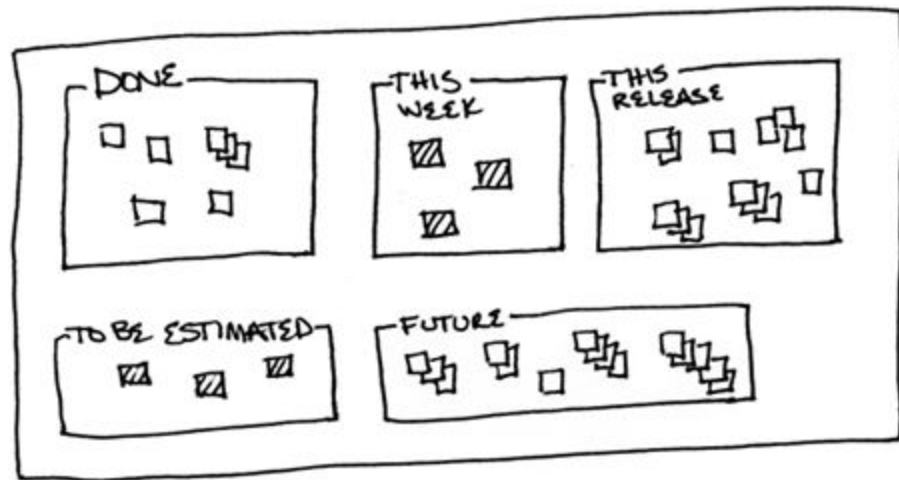
---



# Ambiente de Trabalho



Ambiente de trabalho



Cartazes para "visualizar trabalho" em andamento

# Contratos com Escopo Aberto

---

- Escopo fechado
  - Cliente define requisitos ("fecha escopo")
  - Empresa desenvolvedora: preço + prazo
- Escopo aberto
  - Escopo definido a cada iteração
  - Pagamento por homem/hora
  - Contrato renovado a cada iteração

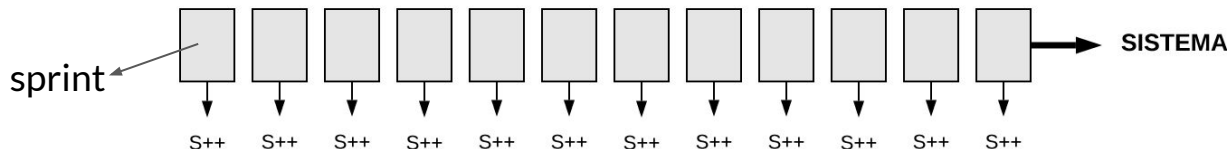
# Contratos com Escopo Aberto

---

- Exige maturidade e acompanhamento do cliente
- Vantagens:
  - Privilegia qualidade
  - Não vai ser "enganado" ("entregar por entregar")
  - Pode mudar de fornecedor

# Scrum vs XP

- Proposto por Jeffrey Sutherland e Ken Schwaber (OOPSLA 1995)
- Scrum não é apenas para projetos de software
  - Logo, não define práticas de programação, como XP
- Scrum define um "processo" mais rígido que XP
  - Eventos, papéis e artefatos bem claros
  - Como em qualquer método ágil, desenvolvimento é dividido em sprints (iterações)
  - Duração de um sprint: até 1 mês, normalmente 15 dias





# O que se faz em uma sprint?

---

- Implementa-se algumas **histórias dos usuários**
- Histórias = funcionalidades (ou features) do sistema
- "Template" = "Como [persona], eu [quero], [para que]."

Postar Pergunta
<i>Um usuário, quando logado no sistema, deve ser capaz de postar perguntas. Como é um site sobre programação, as perguntas podem incluir blocos de código, os quais devem ser apresentados com um layout diferenciado.</i>

Bem simples, deve caber em um post-it

# Quem escreve as histórias?

- Product Owner (PO)
  - Membro (papel) obrigatório em times Scrum
  - Especialista no domínio do sistema
- Suas funções?
  - Escrever histórias dos usuários
  - Explicar histórias para os devs, durante o sprint
  - Definir "testes de aceitação" de histórias
  - Priorizar histórias
  - Manter o backlog do produto



# Backlog do Produto



- Lista de histórias do usuário, que foram escritas pelo PO
- Duas características:
  - Priorizada: histórias do topo têm maior prioridade
  - Dinâmica: histórias podem sair e entrar, à medida que o sistema evolui
- Quais histórias vão entrar no próximo sprint?
  - Essa decisão é tomada no início do sprint
  - Em uma reunião chamada de planejamento do sprint
  - PO propõe histórias que gostaria de ver implementadas
  - Devs decidem se têm velocidade para implementá-las

# Resumindo Scrum



- Elementos principais
  - Sprint (evento)
  - PO e Devs (papeis)
  - Backlog do produto (artefato)
- Importante!
  - Em um time scrum, todos têm o mesmo nível hierárquico
  - PO não é o "chefe" dos Devs
  - Devs têm autonomia para dizer que não vão conseguir implementar tudo que o PO quer em um único sprint

# Agenda de Iteração



## 1ª Semana

Reunião com Cliente	Planejamento da Iteração	Produção
Reunião de retrospectiva		

## 2ª Semana

Produção	Auto-avaliação
Reuniões diárias (standup meeting)	

# Lista de tarefas no planejamento da iteração



## Postar Pergunta

*Um usuário, quando logado no sistema, deve ser capaz de postar perguntas. Como é um site sobre programação, as perguntas podem incluir blocos de código, os quais devem ser apresentados com um layout diferenciado.*

- Projetar e testar a interface Web, incluindo leiaute, CSS templates, etc.
- Instalar banco de dados, projetar e criar tabelas.
- Implementar a camada de acesso a dados.
- Instalar servidor e testar framework web.
- Implementar camada de controle, com operações para cadastrar, remover e atualizar perguntas.
- Implementar interface Web.

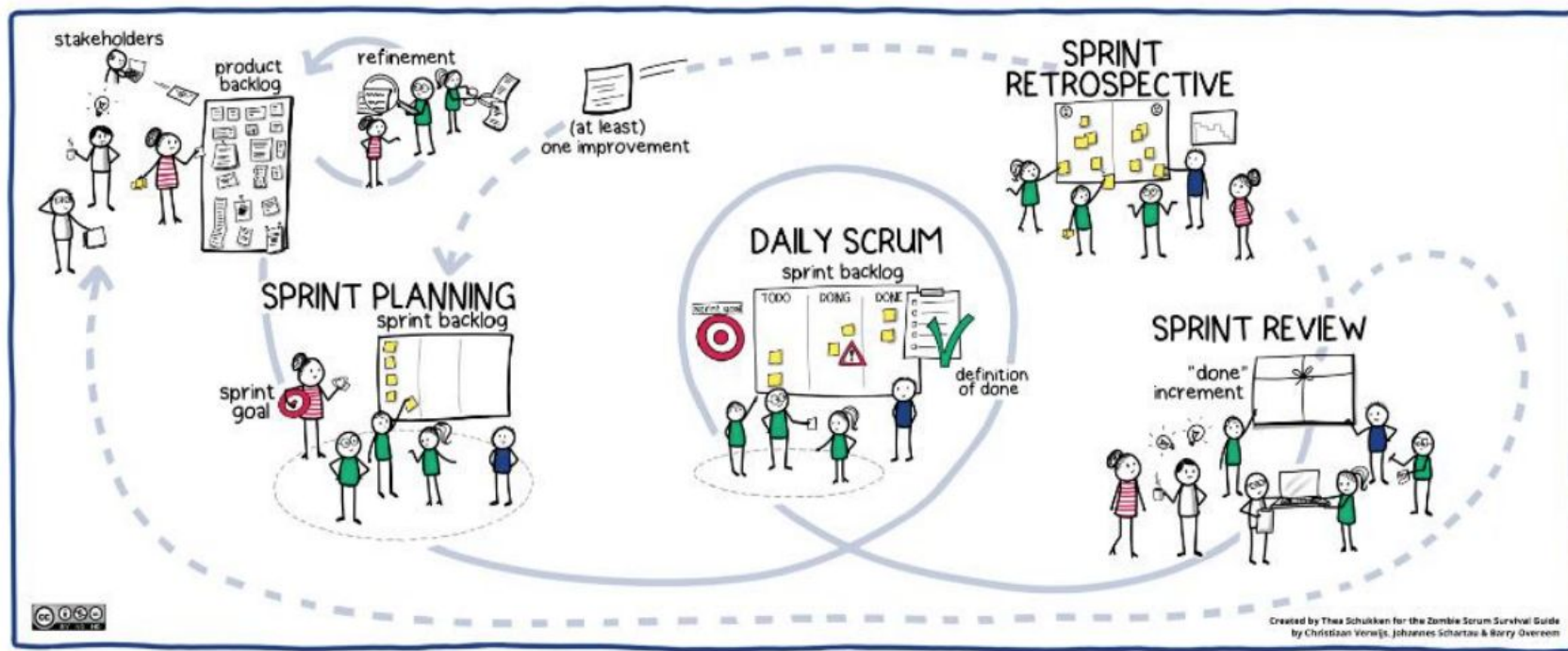
# Atividades importantes

---



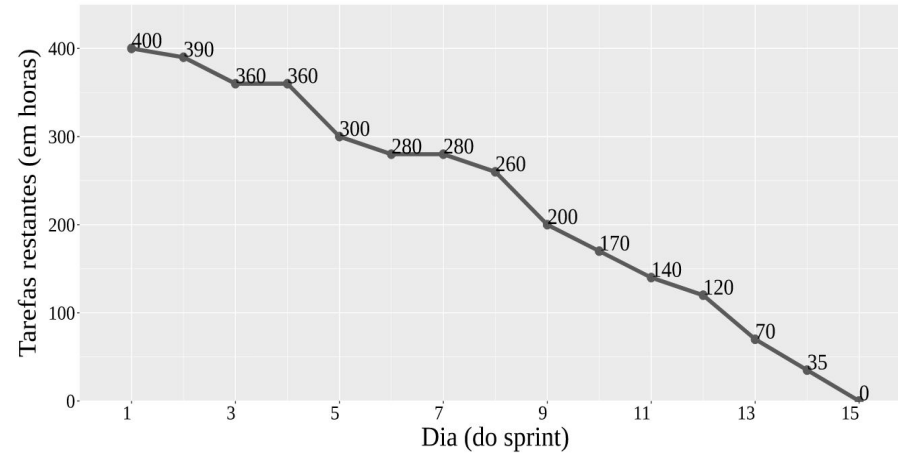


# Visão geral



# Kanban + Gráfico de Burndown

Backlog	To Do	Doing	Testing	Done
<div><div></div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div></div>



Fonte: Engenharia de software moderna

# Kanban



Backlog	Especificação WIP		Implementação WIP		Revisão de Código WIP	
H3	em espec. H2	especificadas T6 T7 T8 T9	em implementação T4 T5	implementadas T3	em revisão T2	revisadas T1

# Kanban

Backlog	Especificação WIP		Implementação WIP		Revisão de Código WIP	
H3	em espec. H2	especificadas T6 T7 T8 T9	em implementação T4 T5	implementadas T3	em revisão T2	revisadas T1

Backlog	Especificação WIP		Implementação WIP		Revisão de Código WIP	
H3	em espec.	especificadas T8 T9 <u>T10 T11 T12</u>	em implementação T4 T5 <u>T6</u> <u>T7</u>	implementadas	em revisão <u>T3</u>	revisadas T1 <u>T2</u>

# Sim: Dirigido a Plano; Não: Ágil

---

- É importante ter uma especificação e projeto?
- Os clientes não estão disponíveis para feedback?
- O Sistema a ser desenvolvido é grande?
- O Sistema é complexo (ex. tempo real)?
- Vai ter um ciclo de vida longo?
- Está utilizando ferramentas “ruins”?
- O time está geograficamente distribuído?
- A cultura do time é orientada a documentação?
- O Time tem um perfil “fraco” de desenvolvimento?
- O sistema está sujeito a regulamentação externa?

# Pergunta



Qual afirmação é **VERDADEIRA**?

- A. A grande diferença entre Ágil e DP é que Ágil não usa requisitos
- B. A grande diferença entre Ágil e DP é medir o progresso de encontro a um plano
- C. Você pode construir apps SaaS utilizando Ágil mas não com DP
- D. A grande diferença entre Ágil e DP é a construção de protótipos e a interação com os clientes durante o processo

# Pergunta

---

Qual afirmação é **VERDADEIRA**?

- A. A grande diferença entre Ágil e DP é que Ágil não usa requisitos
- B. A grande diferença entre Ágil e DP é medir o progresso de encontro a um plano
- C. Você pode construir apps SaaS utilizando Ágil mas não com DP
- D. A grande diferença entre Ágil e DP é a construção de protótipos e a interação com os clientes durante o processo

# Concluindo...



- o SW pode evoluir para atender às necessidades do cliente ao longo do tempo
  - ... se o processo de desenvolvimento abraçar a mudança
- Software é um esporte em equipe (daí a importância de processos)
  - Luis André Barroso, brasileiro que é atualmente VP de Engenharia do Google (vídeo ~1.5 min): [https://youtu.be/S7A6SYI\\_nbc](https://youtu.be/S7A6SYI_nbc)
- Ágil é uma maneira de abraçar a mudança, sendo uma evolução dos modelos Plan&Doc do Big Design Up Front (BDUF)