

Engenharia de Software Moderna

Cap. 2 - Processos

Prof. Marco Tulio Valente

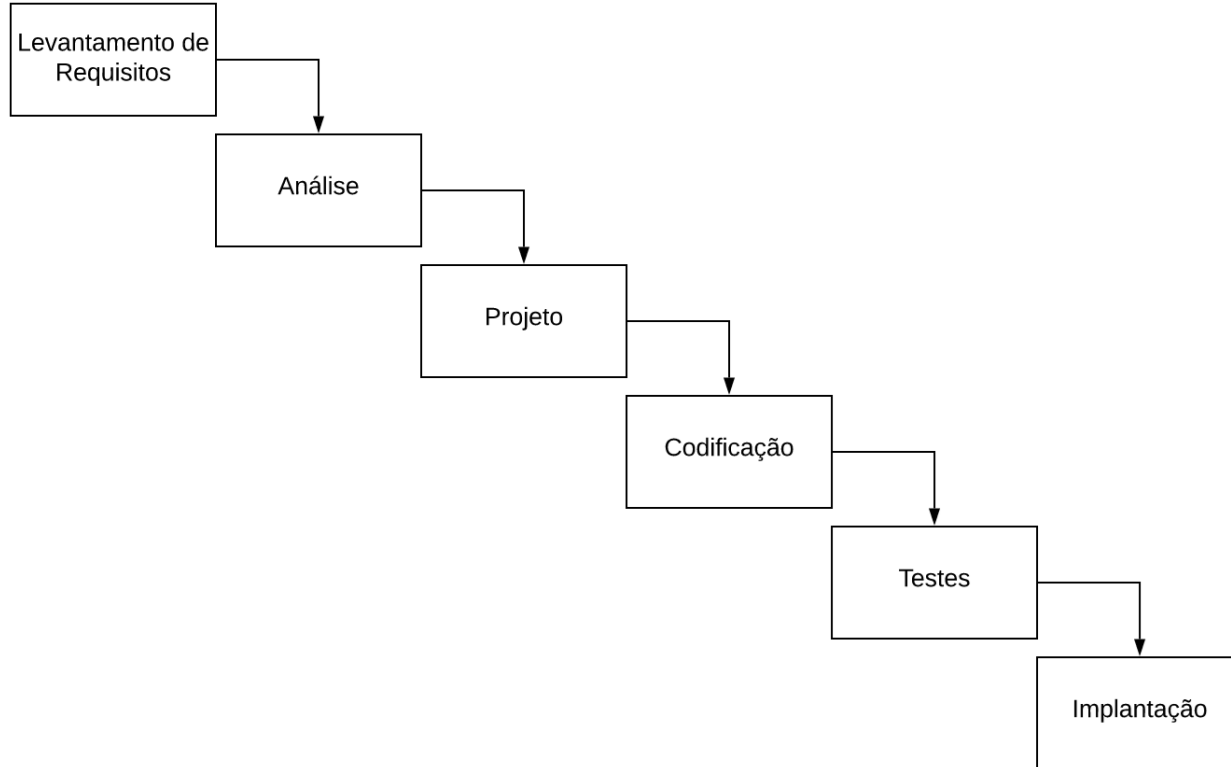
<https://engsoftmoderna.info>, @engsoftmoderna

Licença CC-BY; permite copiar, distribuir, adaptar etc; porém, **créditos devem ser dados ao autor dos slides**

Engenharia Tradicional

- Civil, mecânica, elétrica, aviação, automobilística, etc
- Projeto com duas características:
 - Planejamento detalhado (*big upfront design*)
 - Sequencial
- Isto é: Waterfall (Cascata)
 - Há milhares de anos

Natural que ES começasse usando Waterfall



No entanto: Waterfall não funcionou com software!

Software é diferente

- Engenharia de Software \neq Engenharia Tradicional
- Software \neq (carro, ponte, casa, avião, celular, etc)
- Software \neq (produtos físicos)
- Software é abstrato e "adaptável"

Dificuldade 1: Requisitos

- Clientes não sabem o que querem (em um software)
 - Funcionalidades são "infinitas" (difícil prever)
 - Mundo muda!
- Não dá mais para ficar um ano levantando requisitos, um ano projetando, um ano implementando, etc
- Quando o software ficar pronto, ele estará obsoleto!

Dificuldade 2: Documentações Detalhadas

- Verbosas e pouco úteis
- Na prática, desconsideradas durante implementação
- *Plan-and-document* não funcionou com software – (Documento de Requisitos do Produto)



Manifesto Ágil (2001)

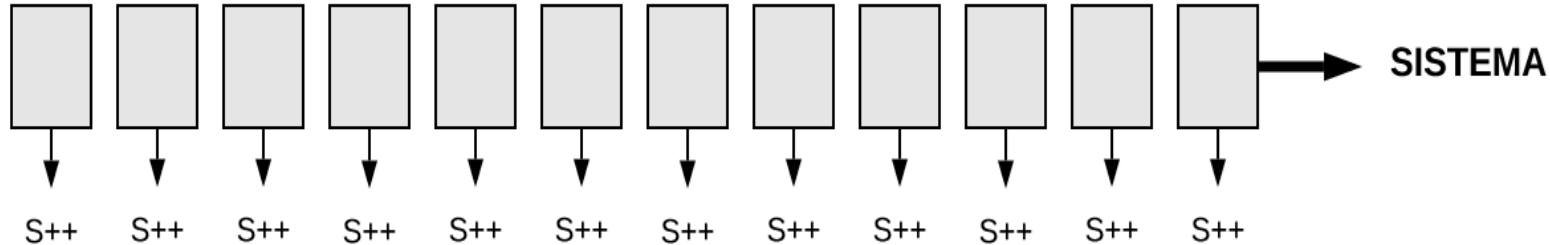


Ideia central: desenvolvimento iterativo

Waterfall



Ágil



Desenvolvimento iterativo

- Suponha um sistema imenso, complexo etc
- Qual o menor "incremento de sistema" eu consigo implementar em 15 dias e validar com o usuário?
- Validar é muito importante!
- Cliente não sabe o que quer!

Reforçando: ágil = iterativo

Outros pontos importantes (1)

- Menor ênfase em documentação
- Menor ênfase em *big upfront design*
- Envolvimento constante do cliente

Outros pontos importantes (2)

- Novas práticas de programação
 - Testes, refactoring, integração contínua, etc

Métodos Ágeis

Métodos Ágeis

- Dão mais consistência às ideias ágeis
 - Definem um processo, mesmo que leve
 - Workflow, eventos, papeis, práticas, princípios etc

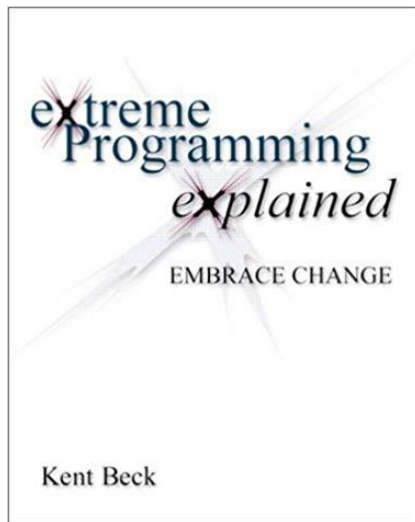
Métodos Ágeis – Exemplos

- Extreme Programming (XP)
- Scrum
- Kanban

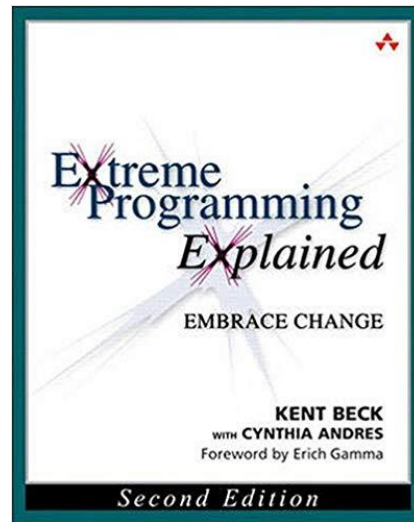
Extreme Programming



Kent Beck



1999



2004

XP = Valores + Princípios + Práticas

Valores

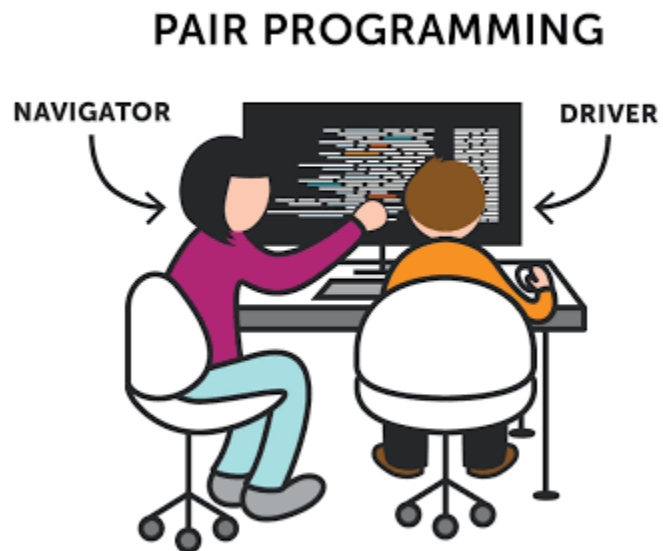
- Comunicação
- Simplicidade
- Feedback
- Coragem
- Respeito
- Qualidade de Vida (semana 40 hrs)

Princípios

- Economicidade
- Melhorias Contínuas
- Falhas Acontecem
- Baby Steps
- Responsabilidade Pessoal

Práticas sobre o Processo de Desenvolvimento	Práticas de Programação	Práticas de Gerenciamento de Projetos
Representante dos Clientes Histórias de Usuário Iterações Releases Planejamento de Releases Planejamento de Iterações Planning Poker Slack	Design Incremental Programação Pareada Testes Automatizados Desenvolvimento Dirigido por Testes (TDD) Build Automatizado Integração Contínua	Ambiente de Trabalho Contratos com Escopo Aberto Métricas

Pair Programming



Estudo com Engenheiros da Microsoft (2008)

- Vantagens:
 - Redução de bugs
 - Código de melhor qualidade
 - Disseminação de conhecimento
 - Aprendizado com os pares
- Desvantagem:
 - Custo

Contratos de Software

- Basicamente, software pode ser desenvolvido:
 - Internamente
 - Externamente (terceirizado), via um contrato
- Contratos de software podem ser de dois tipos:
 - Escopo Fechado
 - Escopo Aberto (defendidos por XP)

Contratos com Escopo Fechado

- Cliente define requisitos ("fecha escopo")
- Empresa desenvolvedora: preço + prazo

Contratos com 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

Scrum

- Proposto por Jeffrey Sutherland e Ken Schwaber

SCRUM Development Process

Ken Schwaber

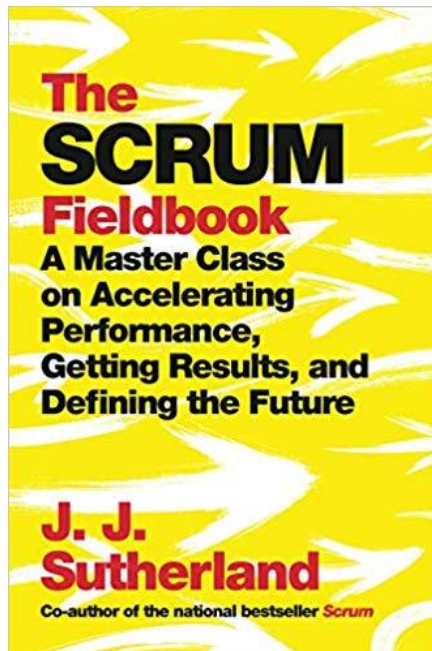
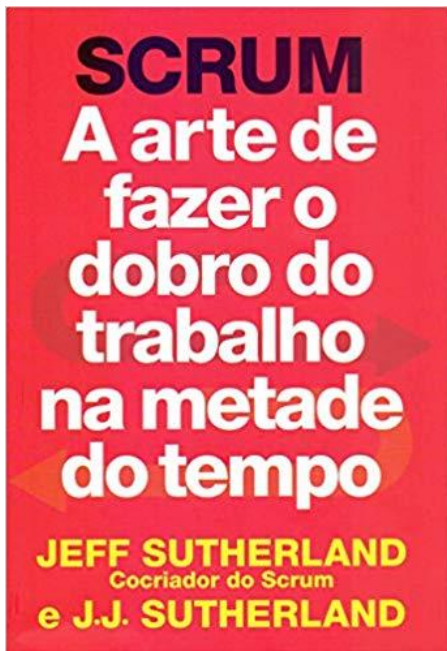
Advanced Development Methods
131 Middlesex Turnpike Burlington, MA 01803
email virman@aol.com Fax: (617) 272-0555

ABSTRACT. *The stated, accepted philosophy for systems development is that the development process is a well understood approach that can be planned, estimated, and successfully completed. This has proven incorrect in practice. SCRUM assumes that the systems development process is an unpredictable, complicated process that can only be roughly described as an overall progression. SCRUM defines the systems development process as a loose set of activities that combines known, workable tools and techniques with the best that a development team can devise to build systems. Since these activities are loose, controls to manage the process and inherent risk are used. SCRUM is an enhancement of the commonly used iterative/incremental object-oriented development cycle.*

KEY WORDS: *SCRUM SEI Capability-Maturity-Model Process Empirical*

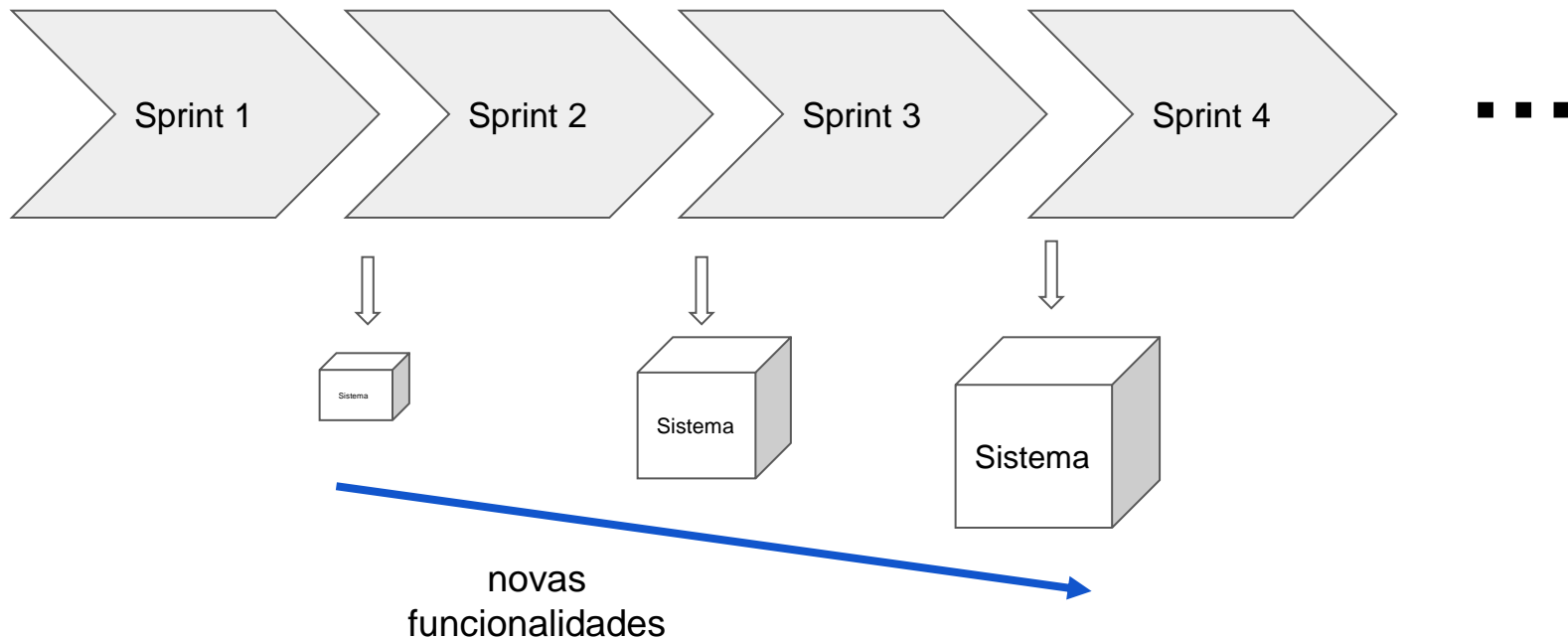
Scrum

- Scrum é uma indústria: livros, consultoria, certificações, etc



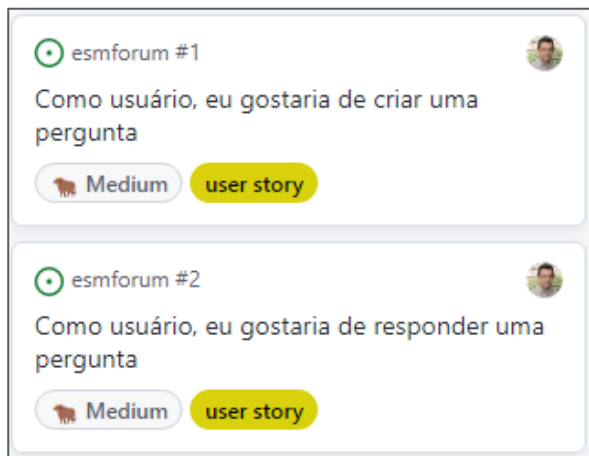
Principal evento: Sprints

- Duração: até 1 mês, normalmente 15 dias



O que se faz em um sprint?

- Implementa-se algumas **histórias dos usuários**
- Histórias = funcionalidades do sistema
- Exemplo: fórum de perguntas e respostas



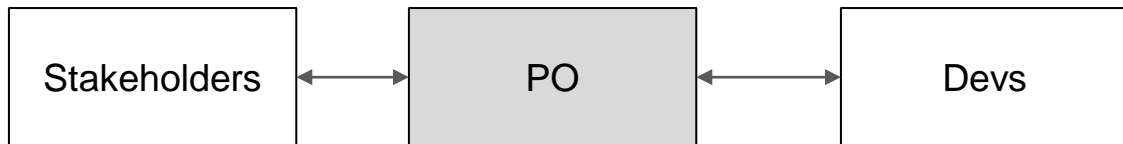
Quem escreve as histórias?

- **Product Owner (PO)**
- Papel obrigatório em times Scrum
- Especialista no domínio do sistema

Antes: Waterfall

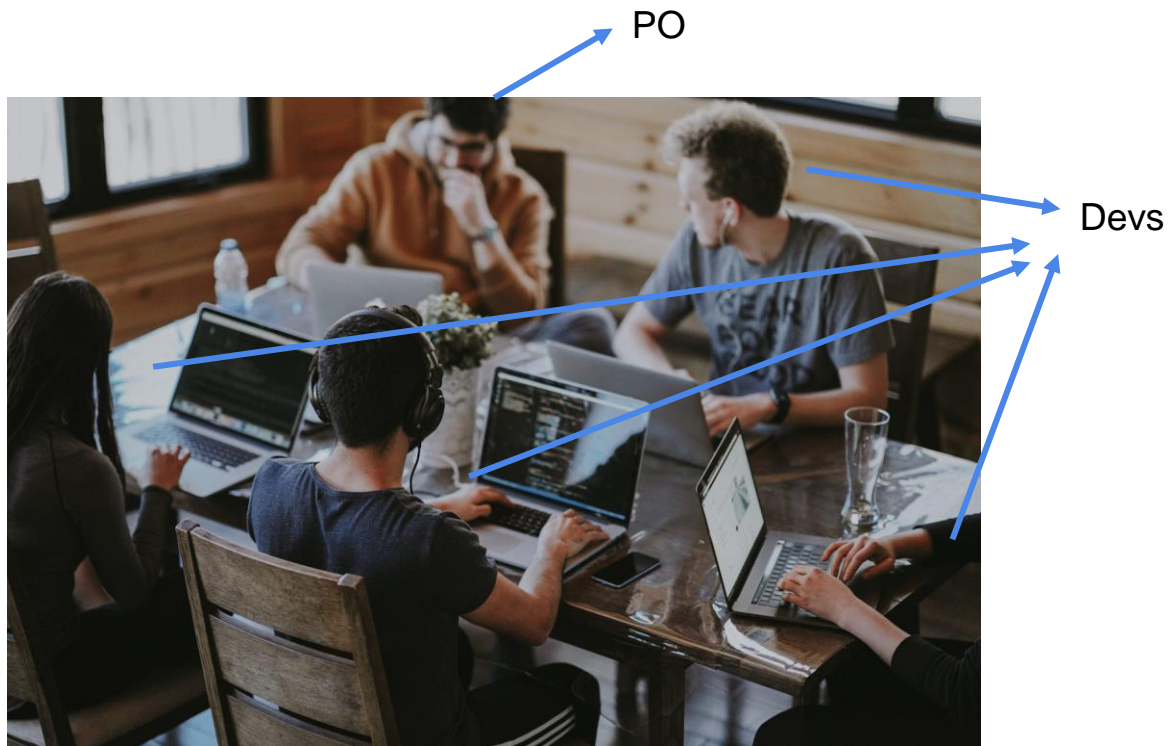


Hoje: Scrum



- Durante sprint, PO explica histórias para devs
- Troca-se documentação formal/escrita por informal/verbal
- Conversas entre PO e devs

Hoje ...



Product Owner senta junto dos desenvolvedores e explica requisitos para eles

Funções de um PO

- Escrever histórias dos usuários
- Explicar histórias para os devs
- Definir "testes de aceitação" de histórias
- Priorizar histórias

Backlog do Produto

- Lista de histórias do usuário
- (e outros itens de trabalho importantes)
- Duas características:
 - Priorizada: histórias do topo têm maior prioridade
 - Dinâmica: histórias podem sair e entrar...

Resumindo

- Iteração: sprint
- Papéis: PO e Devs
- Artefato: backlog do produto

Quais histórias vão entrar no próximo sprint?

- Decisão 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

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

Voltando ao Planejamento do Sprint

- 1a parte da reunião:
 - Definem-se as histórias do sprint
- 2a parte da reunião:
 - Histórias são quebradas em tarefas
 - Tarefas são alocadas a devs

Exemplo: fórum de perguntas e respostas

Backlog do Produto

esmforum #1



Como usuário, eu gostaria de criar uma pergunta



Medium

user story

esmforum #2



Como usuário, eu gostaria de responder uma pergunta



Medium

user story

esmforum #3



Como usuário, eu gostaria de editar e deletar minhas perguntas



Medium

user story

esmforum #4



Como usuário, eu gostaria de editar e deletar minhas respostas



Medium

user story

esmforum #4 ...



Como usuário, eu gostaria de editar e deletar minhas respostas



Medium

user story

esmforum #5



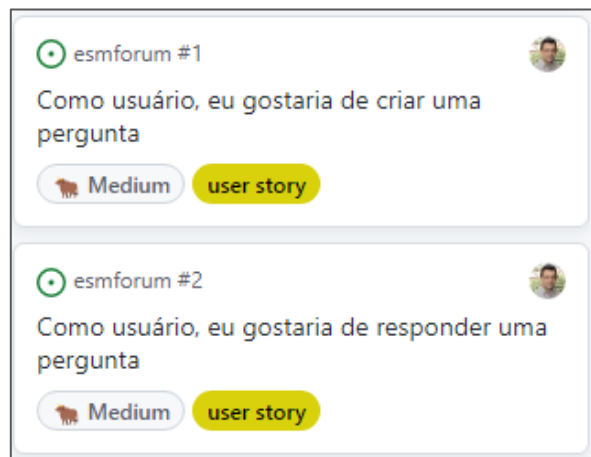
Como usuário, eu gostaria de me cadastrar no sistema e ter uma página de perfil



Large

user story

Histórias do Sprint



Backlog do Sprint

- Lista de tarefas do sprint (com responsáveis e duração)

<input checked="" type="checkbox"/> esmforum #11 Instalar banco de dados e criar primeiras tabelas task	<input checked="" type="checkbox"/> esmforum #12 ... Criar e testar uma primeira rota usando o Express task	<input checked="" type="checkbox"/> esmforum #14 Implementar no backend a lógica de criar e listar perguntas task
<input checked="" type="checkbox"/> esmforum #20 Instalar node.js e Express task	<input checked="" type="checkbox"/> esmforum #13 Implementar versão inicial da tela principal task	<input checked="" type="checkbox"/> esmforum #15 Adaptar tela principal para incorporar a lista e a criação de perguntas task

Sprint está pronto para começar!

Times e Papéis de Scrum

Times Scrum

- Pequenos (time de basquete a um time de futebol)
- 5 a 11 membros, sendo 1 PO e 1 Scrum Master
- Multidisciplinares: devs, designers, cientistas de dados, etc
- Observação: times ágeis são hoje chamados, com frequência, de squads

Scrum Master

- Especialista em Scrum: ajuda o time a adotar Scrum
- Removedor de impedimentos não-técnicos
 - Exemplo: desenvolvedores não têm máquinas boas
- Pode também coletar métricas de processo
- Não é o chefe do time, mas um “líder servidor”
- Pode pertencer a mais de um time

Mais alguns eventos

Reuniões Diárias

- 15 minutos de duração. Cada participante diz:
 - o que ele fez ontem
 - o que pretende fazer hoje
 - e se está tendo alguma dificuldade
- Objetivos:
 - Melhorar comunicação
 - Antecipar problemas

Sprint termina com dois eventos:
Review e Retrospectiva

Revisão do Sprint

- Time mostra o resultado do sprint para PO e stakeholders
- Implementação das histórias pode ser:
 - Aprovada
 - Aprovada parcialmente
 - Reprovada
- Nos dois últimos casos, história volta para o backlog do produto

Retrospectiva

- Último evento do sprint
- Time se reúne para decidir o que melhorar
 - O que deu certo?
 - Onde precisamos melhorar?
- Modelo mental: melhorias constantes
- Não é para "lavar a roupa suja"

Comentário interessante sobre o objetivo de eventos Scrum



Cory House ✓

@housecor



You don't need a daily standup. But you do need to communicate often.

You don't need formal retrospectives. But you do need to regularly discuss improvement opportunities.

You don't need sprints. But you do need to break work down and deploy often.

You don't need a sprint review. But you do need to iterate based on feedback.

You don't need a scrum master. But you do need to assure the things above happen.

12:12 PM · Mar 11, 2023 · **195.1K** Views

181 Retweets **21** Quote Tweets **1,347** Likes

<https://twitter.com/i/web/status/1634572956746776577>

Mais alguns conceitos de Scrum

Time-box






























- Eventos têm uma duração bem definida

Evento	Time-box
Planejamento do Sprint	máximo de 8 horas
Sprint	menos de 1 mês
Reunião Diária	15 minutos
Revisão do Sprint	máximo de 4 horas
Retrospectiva	máximo de 3 horas

Critérios para Conclusão de Histórias (done criteria)

- Critérios internos para considerar histórias prontas
- Também chamados de DoD (Definition of Done)
- Exemplos:
 - Testes de unidade com cobertura $\geq 75\%$
 - Revisão de código por outro dev do time
 - Atualizar documentação (se atualizou API)
 - Teste de performance (para certas histórias)

Scrum Board

Backlog	To Do	Doing	Testing	Done
     	        	       	   	 

Exemplo: projeto da Mozilla (usando GitHub Projects)

Smart Scheduling

Updated 13 days ago

Filter cards

100 Backlog

📄 Bug 1632870 - Store test configuration in the task definition

Added by ahal

📄 Bug 1667401 - Always schedule "new manifests" with manifest based bugbug optimizers

Added by ahal

📄 Investigate Treeherder UI/UX changes for test filtering

Added by armenzg

📄 Handle pushes containing multiple backouts

mozci#204 opened by marco-c

enhancement

📄 When a task/group has inconsistent classifications, use the status of the task/group on the backout to figure it out

mozci#200 opened by marco-c

enhancement

📄 Bug 1635921 - Use |mach try fuzzy| as a means to select configurations with

1 Next up

📄 Reinstate integration tests

mozci#390 opened by marco-c

3 In progress

📄 Build a dashboard to see schedulers results over time (both # of tests and durations)

Added by marco-c

📄 Add a way to get durations of shadow scheduler tasks

mozci#102 opened by ahal

metrics

1 linked pull request

📄 Bug 1639164 - "mach try auto" should select the best platforms to run manifests on

assigned: marco

Added by marco-c

222 Done

📄 Make adr dependency optional

mozci#388 opened by marco-c

📄 Bug 1671422 - Add durations to group_result actions in errorssummary formatter

assigned: ahal

Added by ahal

📄 Cache results from data sources

mozci#368 opened by marco-c

enhancement

1 linked pull request

📄 OOMs while analyzing some pushes

mozci#366 opened by marco-c

bug

1 linked pull request

📄 Add support for getting groups and groups results in the Treeherder data source

mozci#312 opened by marco-c

1 linked pull request

Story Points

Story Points

- Usados para estimar o tamanho de histórias
- Ajudar a definir o que vai “caber” no sprint
- Uso não é obrigatório em Scrum
- Definição de story points é "empírica"

Escala de story points

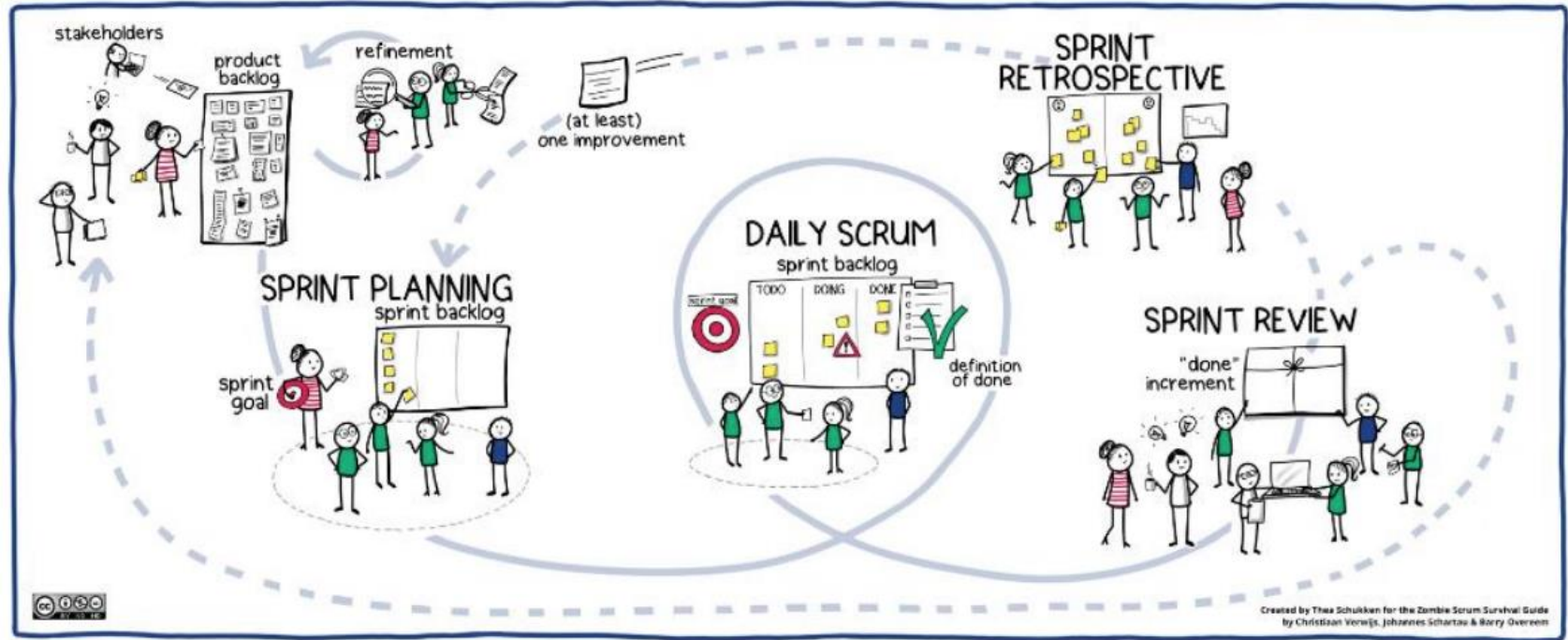
- Mais comum: 1, 2, 3, 5, 8, 13, ...
- Velocidade: número de story points que consegue implementar em um sprint

Exemplo

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

Definido pelos devs
do time

Resumo em 1 slide



Kanban

Kanban

- Origem na década de 50 no Japão
- Sistema de Produção da Toyota
- Manufatura lean, produção just-in time, etc

kanban = "cartão visual"



Kanban em Desenvolvimento de Software



Kanban vs Scrum

- Kanban é mais simples
- Não existem sprints
- Não é obrigatório usar papéis e eventos, incluindo:
 - Scrum master
 - Daily Scrum, Retrospectivas, Revisões
- Time define os papéis e eventos

Kanban

"Grandes" colunas do quadro:
Passos

Backlog	Especificação WIP		Implementação WIP		Revisão de Código WIP	
	em espec.	especificadas	em implementação	implementadas	em revisão	revisadas

1a sub-coluna:
em andamento

2a sub-coluna:
concluídas

Iremos explicar daqui a
pouco o que significa WIP


Fluxo de trabalho (tempo)

Kanban

- Ideia central: sistema pull
- Membros "puxam" trabalho:
 - a. Escolhem uma tarefa para trabalhar
 - b. Concluem tarefa (movem ela para frente no quadro)
 - c. Voltam para o passo (a)




tempo

Backlog	Especificação WIP		Implementação WIP		Revisão de Código WIP	
	em espec.	especificadas	em implementação	implementadas	em revisão	revisadas





tempo





Backlog	Especificação WIP		Implementação WIP		Revisão de Código WIP	
	em espec.	especificadas	em implementação	implementadas	em revisão	revisadas

Backlog	Especificação WIP		Implementação WIP		Revisão de Código WIP	
	em espec.	especificadas	em implementação	implementadas	em revisão	

tempo

Backlog	Especificação WIP		Implementação WIP		Revisão de Código WIP	
	em espec.	especificadas	em implementação	implementadas	em revisão	revisadas

Backlog	Especificação WIP		Implementação WIP		Revisão de Código WIP	
	em espec. 	especificadas	em implementação	implementadas	em revisão	

Backlog	Especificação WIP		Implementação WIP		Revisão de Código WIP	
	em espec.	especificadas    	em implementação	implementadas	em revisão	revisadas



tempo

Backlog	Especificação WIP		Implementação WIP		Revisão de Código WIP	
	em espec.	especificadas □ □ □ □	em implementação	implementadas	em revisão	revisadas

Backlog	Especificação WIP		Implementação WIP		Revisão de Código WIP	
	em espec.	especificadas □ □ □	em implementação □	implementadas	em revisão	revisadas

Backlog	Especificação WIP		Implementação WIP		Revisão de Código WIP	
	em espec.	especificadas □ □ □	em implementação	implementadas □	em revisão	

Exemplo 2

Exemplo 2: Ontem no final do dia

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

Hoje no final do dia:

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

Kanban: Limites WIP

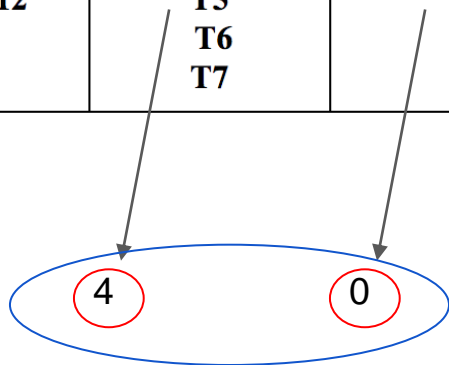
Limites WIP (Work in Progress)

Backlog	Especificação		Implementação		Revisão de Código	
		2		5	3	
H3	em espec.	especificadas T8 T9 T10 T11 T12	em implementação	implementadas T4 T5 T6 T7	em revisão T3	revisadas T1 T2

Limites WIP (Work in Progress)

- Número máximo de tarefas em um passo
- Contando: tarefas em andamento e concluídas

Backlog	Especificação 2		Implementação 5		Revisão de Código 3	
H3	em espec.	especificadas T8 T9 T10 T11 T12	em implementação T4 T5 T6 T7	implementadas	em revisão T3	revisadas T1 T2



Objetivos dos Limites WIP

- Criar um fluxo de trabalho sustentável
 - Evitar que o time fique sobrecarregado de trabalho
 - WIP = "acordo" entre o time e a organização
 - Capacidade de trabalho de um time
- Evitar que o trabalho fique concentrado em um passo

Frase comum em Kanban:

“pare de começar e comece a terminar”

Mais um exemplo: Implementação no limite

Backlog	Especificação (2)		Implementação (5)		Revisão (3)	
			X X	X X X		

- Talvez seja o momento de revisar algumas tarefas

Comentários Finais sobre Kanban

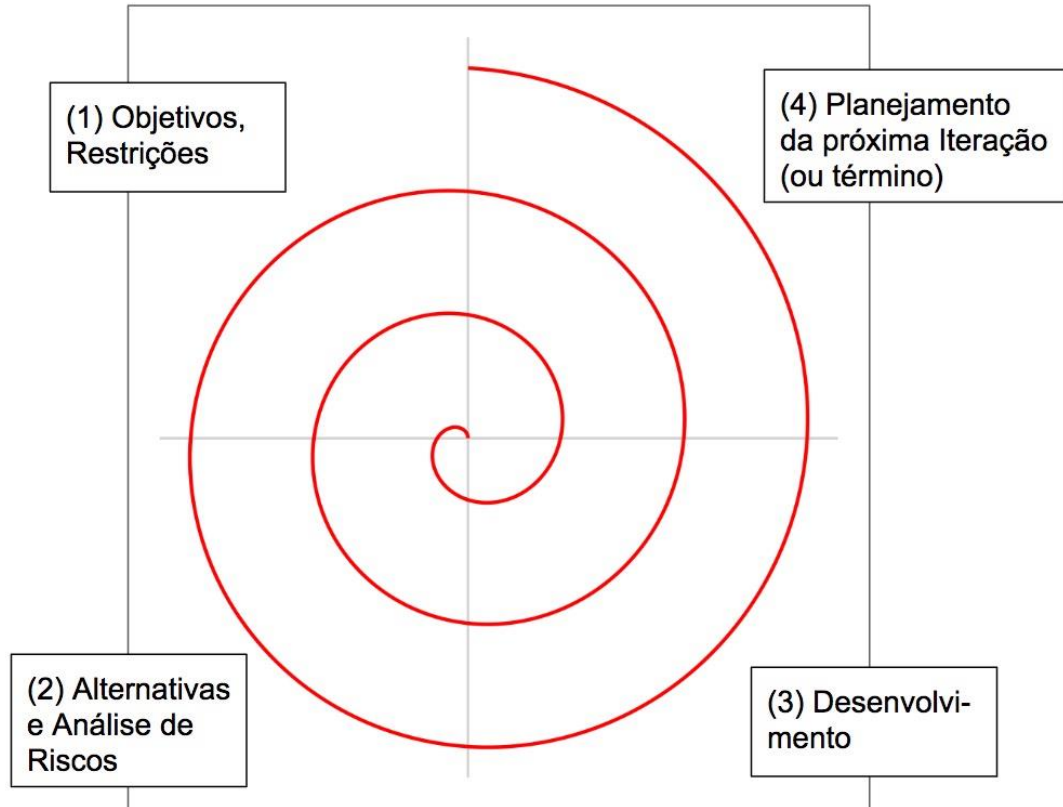
- Kanban é mais simples do que Scrum
 - Kanban é mais adequado para times maduros
 - Talvez, começar com Scrum e depois ir para Kanban
- Kanban é um método evolucionário, pois:
 - Começa-se com o que se faz hoje
 - Vamos entendendo o fluxo atual e seu gargalos
 - E implementando pequenas melhorias, mas graduais

Outros Processos (não ágeis)

Transição de Waterfall para Ágil

- Antes da disseminação dos princípios ágeis, alguns métodos **iterativos** ou **evolucionários** foram propostos
- Transição Waterfall (~1970) e Ágil (~2000) foi gradativa
- Exemplos:
 - Espiral (1986)
 - Rational Unified Process (RUP) (2003)

Modelo em Espiral



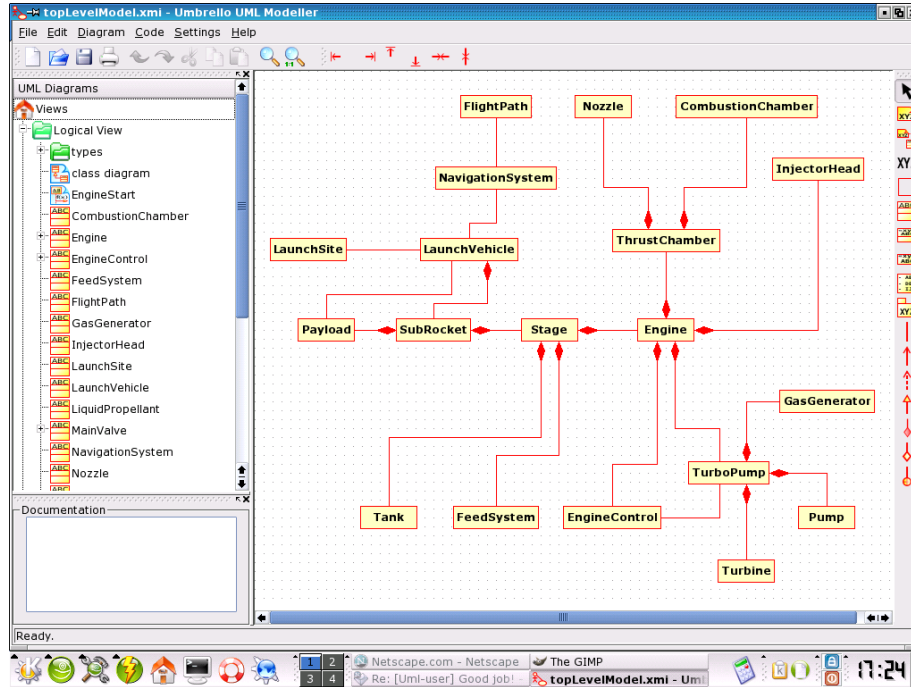
Proposto por Barry Boehm

Iterações: 6 a 24 meses (logo, mais que em XP ou Scrum)

Rational Unified Process (RUP)

- Proposto pela Rational, depois comprada pela IBM
- Duas características principais:
 - Diagramas UML
 - Ferramentas CASE

CASE (Computer-Aided Software Engineering)



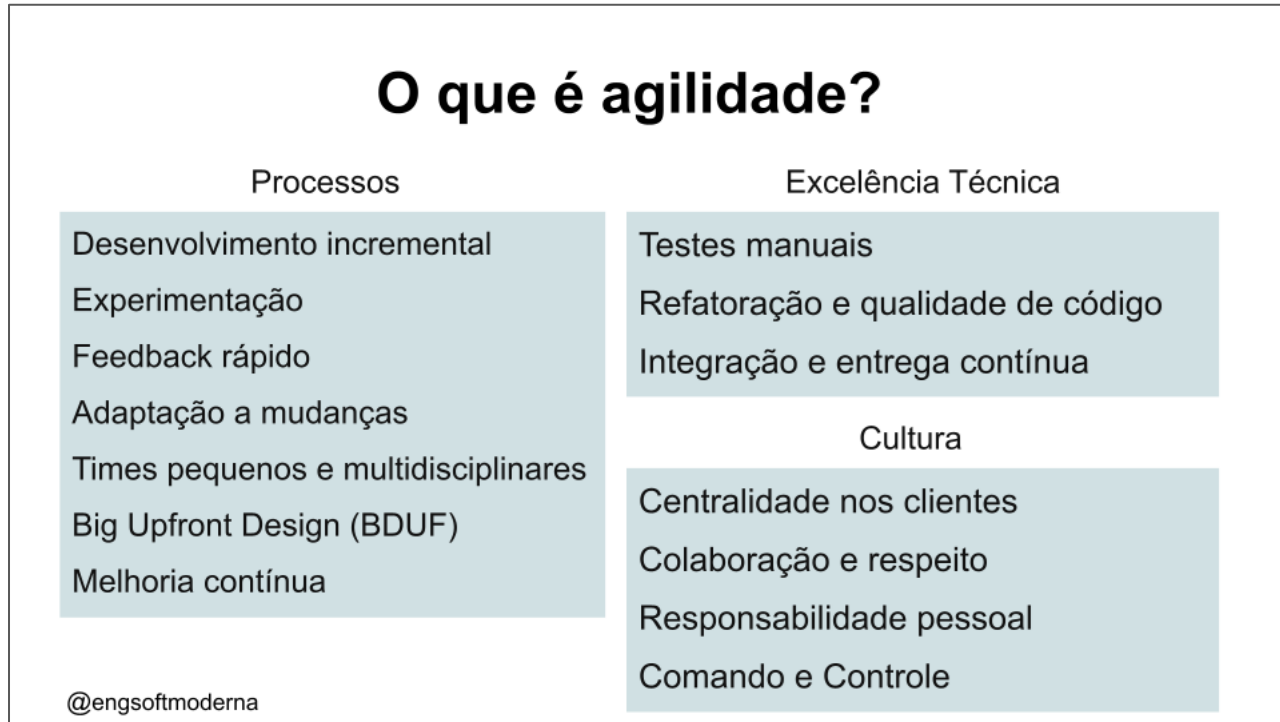
Nome vem de sistemas de CAD
(usados em engenharia tradicional)



Antes de concluir

- Processos não são adotados 100% igual ao manual
 - Bom senso é importante
 - Experimentação é importante

1. O seguinte slide resume métodos ágeis em três áreas: processos, excelência técnica e cultura. No entanto, em cada uma delas, existe uma característica que **NÃO** é compatível com os princípios de agilidade. Indique tais características.



2. Suponha que você trabalha em um Banco X e ficou encarregado pela implantação do PIX no mesmo. Então, sua primeira decisão foi montar um squad para ficar responsável por essa funcionalidade. Você decidiu também que esse squad deveria usar Scrum. Responda então:

(a) Quais profissionais você deve contratar ou convidar para esse squad? Qual a responsabilidade deles?

(b) Suponha que o projeto foi concluído com sucesso. Meses depois, o Banco Central baixou a seguinte regra: transferências via PIX entre 20:00 e 06:00 devem obedecer ao limite de R\$ 1.000,00. Qual seria o papel de cada membro do time na implementação dessa nova regra? Descreva de forma simplificada.

(continuação)

(c) Escreva três histórias de usuários do projeto PIX no Banco X

(d) Escolha uma das histórias anteriores e descreva:

- seus testes de aceitação (também chamados critérios de aceitação)
- seus critérios de conclusão (“done criteria”).

3. Suponha que a FURB pretende migrar para um novo sistema de apoio ao ensino, que irá substituir o DION. Ela está cogitando três estratégias:

(a) construir o novo sistema internamente, usando engenheiros de software que são servidores da universidade.

(b) terceirizar o desenvolvimento com uma fábrica de software.

(c) comprar ou assinar algum produto disponível no mercado.

Suponha que o sistema desenvolvido ou comprado nas três alternativas será desenvolvido usando Scrum. Então, descreva o perfil de PO (Product Owner) mais adequado para cada uma delas.

4. No livro e nas aulas, ao comentar sobre os itens dos backlogs de scrums, demos ênfase a histórias de usuários. Porém, elas não são os únicos itens possíveis em um backlog. Por exemplo, suponha os seguintes tipos de bugs:

(a) Um bug detectado durante a implementação de uma história de um sprint.

(b) Um bug não crítico reportado por um usuário do sistema.

(c) Um bug muito crítico que está impactando diversos usuários do sistema.

Como esses bugs deveriam ser tratados por um time que usa Scrum?

5. Existem quatro variáveis importantes em contratos de software:

- Escopo
- Tempo
- Custo
- Qualidade

Métodos ágeis, principalmente XP, argumentam que é impossível fixar todas essas quatro variáveis por meio de um contrato, pois surpresas sempre vão acontecer durante o projeto. Suponha então um contrato com escopo fechado. Se ocorrer uma surpresa ao longo do projeto, qual dessas variáveis tende a ser sacrificada pela empresa contratada a fim de evitar multas?