

Engenharia de Software

Processos

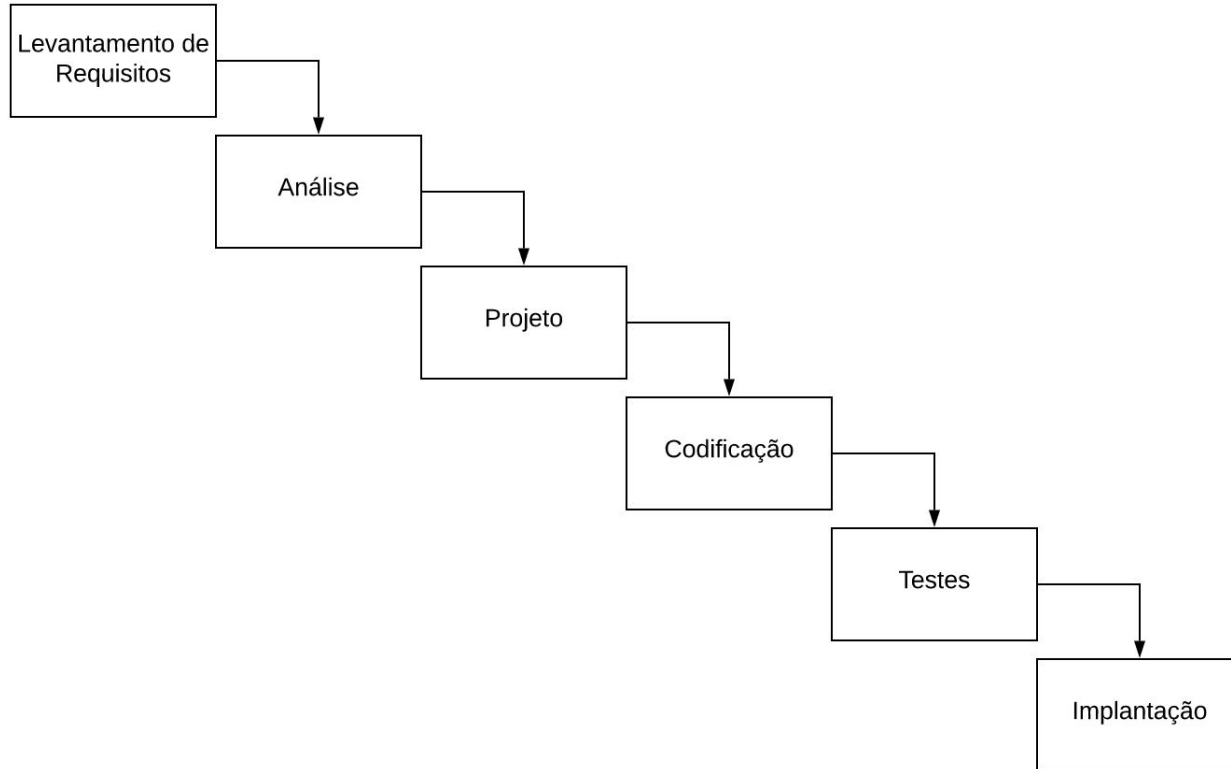
Prof. Paulo Meirelles

paulo.meirelles@ufabc.edu.br

Engenharia Tradicional

- Civil, mecânica, elétrica, aviação, automobilística, etc
- Projeto com duas características:
 - Planejamento detalhado (*big upfront design*)
 - Sequencial
- Isto é: Waterfall
 - 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 1 ano levantando requisitos, 1 ano projetando, 1 ano implementando, etc
- Quando o software ficar pronto,
ele estará obsoleto!

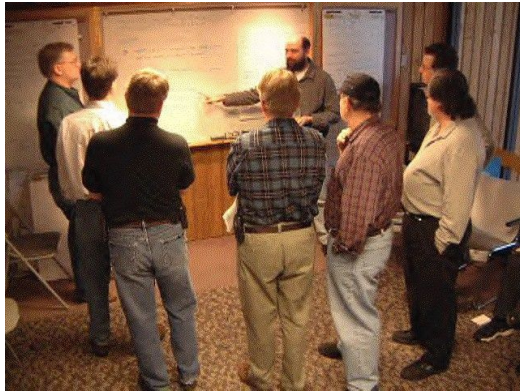
Dificuldade 2: Documentações Detalhadas

- Verbosas e pouco úteis
- Na prática, desconsideradas durante implementação
- *Plan-and-document* não funcionou com software



Manifesto Ágil (2001)

- Encontro de 17 engenheiros de software em Utah
- Crítica a modelos sequenciais e pesados
- Novo modelo: incremental e iterativo





Manifesto para Desenvolvimento Ágil de Software

Estamos descobrindo maneiras melhores de desenvolver software, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo. Através deste trabalho, passamos a valorizar:

Indivíduos e interações mais que processos e ferramentas
Software em funcionamento mais que documentação abrangente
Colaboração com o cliente mais que negociação de contratos
Responder a mudanças mais que seguir um plano

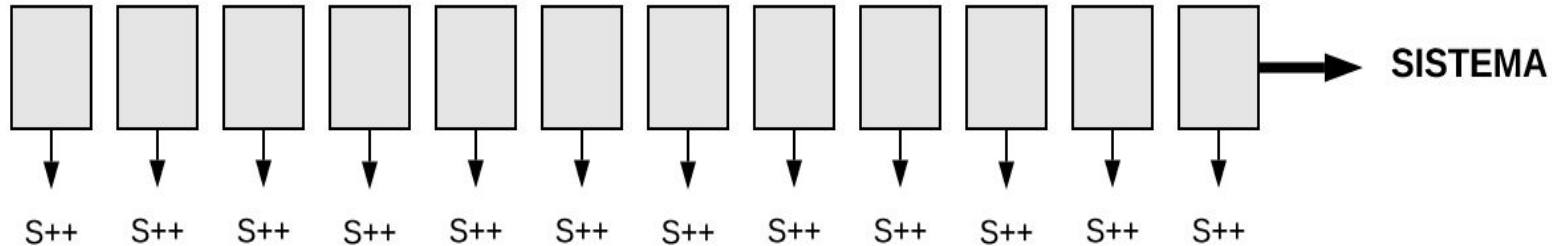
Ou seja, mesmo havendo valor nos itens à direita,
valorizamos mais os itens à esquerda.

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 (*product owner*)

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, papéis, práticas, princípios etc

Métodos Ágeis

- Extreme Programming (XP)
- Scrum
- Kanban

Antes de começar

- Nenhum processo é uma bala-de-prata
- Processo ajuda a não cometer certos "grandes erros"
- Processos não são adotados 100% igual ao manual
 - Bom senso é importante
 - Experimentação é importante

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)

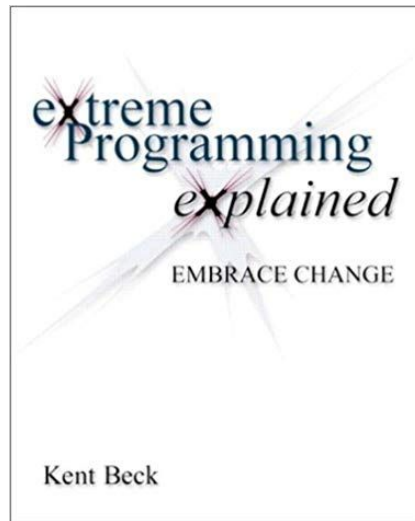
https://youtu.be/S7A6SYI_nbc

Extreme Programming (XP)

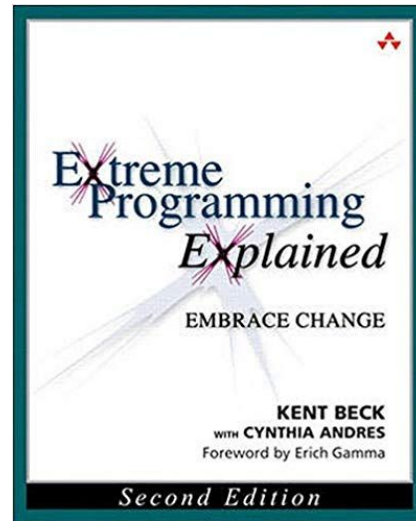
Extreme Programming



Kent Beck



1999



2004

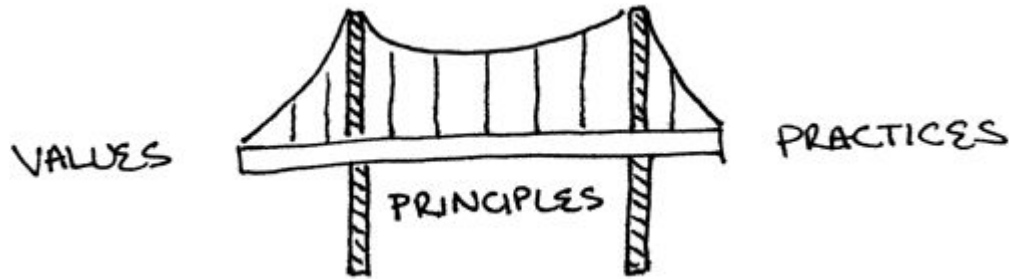
XP = Valores + Princípios + Práticas

Valores

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

Valores ou "cultura" são fundamentais em software!

XP = Valores + Princípios + Práticas



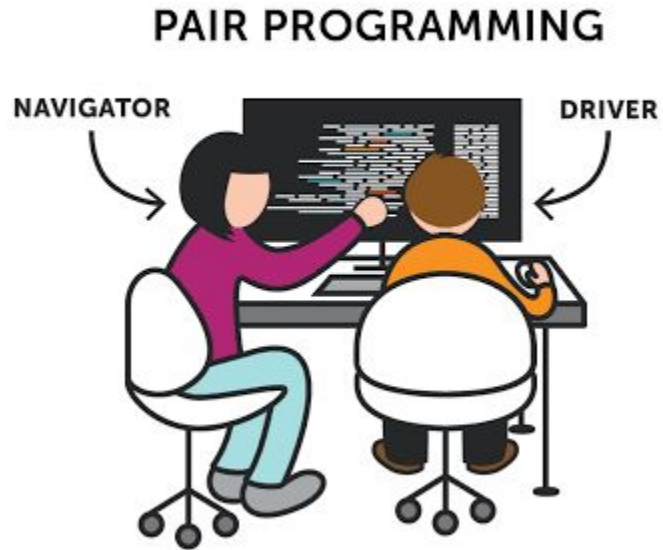
Princípios

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

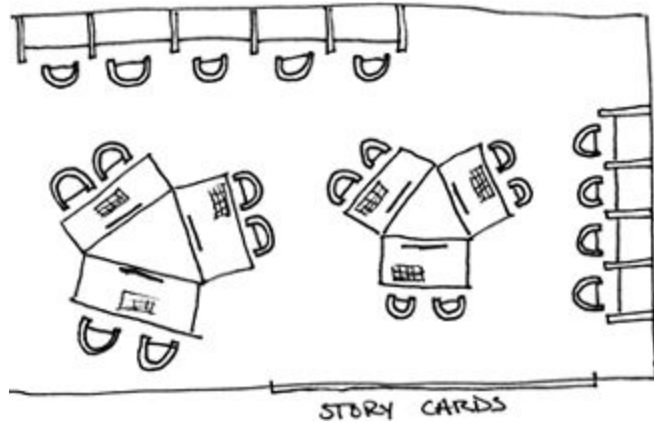
XP = Valores + Princípios + Práticas

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

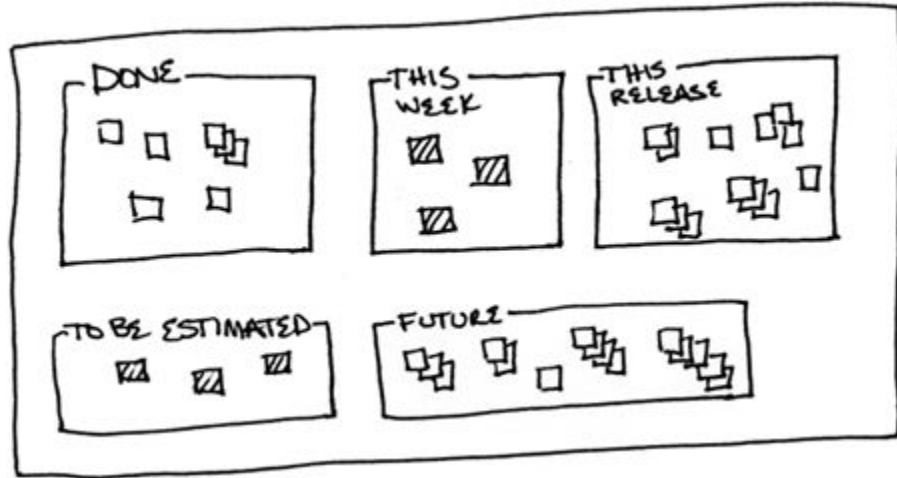
Pair Programming



(2) 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

Scrum

- Proposto por Jeffrey Sutherland e Ken Schwaber (OOPSLA 1995)

SCRUM Development Process

Ken Schwaber

Advanced Development Methods

131 Middlesex Turnpike Burlington, MA 01803

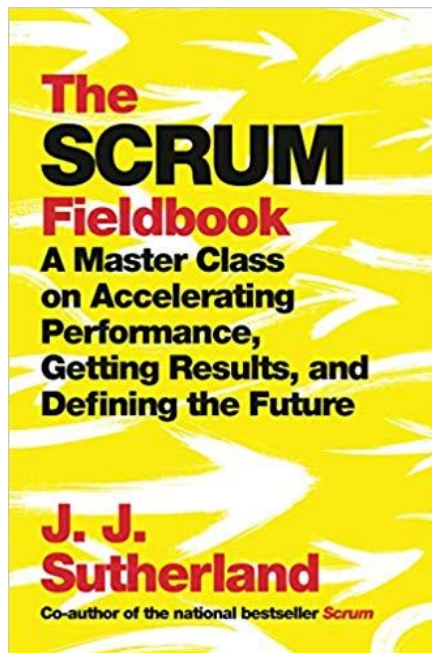
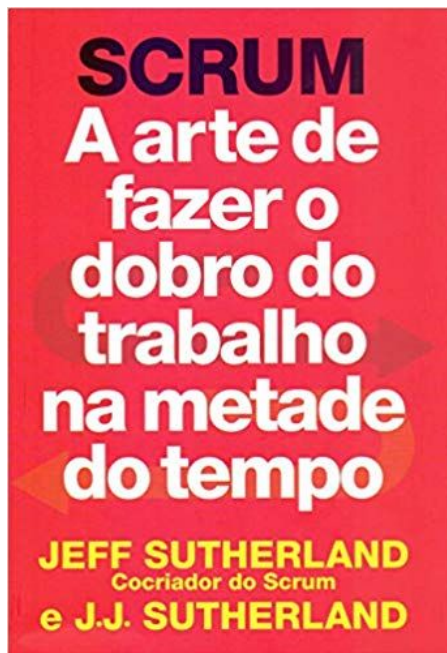
email virman@aol.com Fax: (617) 272-0555

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, marketing

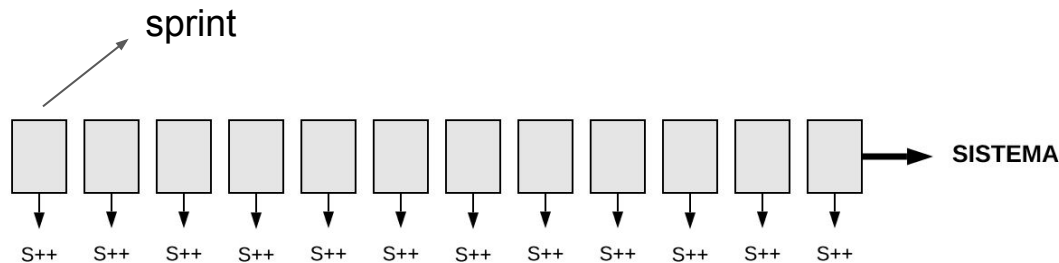


Scrum vs XP

- 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

Principal evento: Sprints

- 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 um sprint?

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

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.

Bem simples, deve caber em um post-it

Quem escreve as histórias?

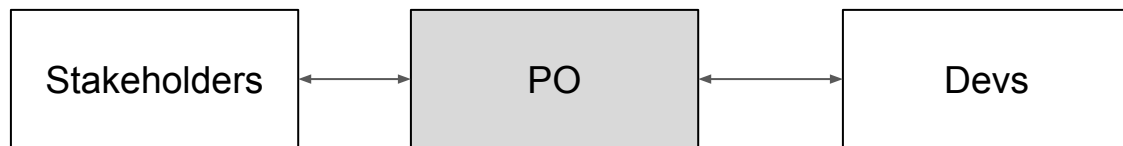
- **Product Owner (PO)**
- Membro (papel) obrigatório em times Scrum
- Especialista no domínio do sistema

Antes (Waterfall)



Observação: stakeholders são os clientes do sistema e qualquer outra parte interessada ou afetada por ele.
Exemplo: departamento jurídico é interessado no módulo de contratos de um sistema

Hoje (Scrum)



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

Funções de um PO

- 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

Resumindo

- Sprint (evento)
- PO e Devs (papeis)
- Backlog do produto (artefato)

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

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:

- História: Postar Perguntas
- Tarefas:
 - Projetar e testar a interface Web, incluindo layout, CSS templates, etc.
 - Instalar banco de dados, projetar e criar tabelas.
 - Implementar a camada de acesso a dados.
 - Implementar camada de controle, com operações para cadastrar, remover e atualizar perguntas.
 - Implementar interface Web.

Backlog do Sprint

- Lista de tarefas do sprint, com responsáveis e duração estimada

Sprint está pronto para começar!

Scrum: Conceitos Complementares

Times Scrum

- Pequenos, do tamanho de "duas pizzas"
- 5 a 11 membros
 - 1 PO
 - 3 a 9 desenvolvedores
 - 1 Scrum Master

Scrum Master

- Especialista em Scrum do time
 - Ajudar o time a seguir Scrum e seus eventos
- Removedor de "impedimentos" não-técnicos
 - Exemplo: desenvolvedores não têm máquinas boas
- Não é o chefe do time
- Pode pertencer a mais de um time

Mais alguns eventos

Reuniões Diárias

- 15 minutos de duração; em pé. 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

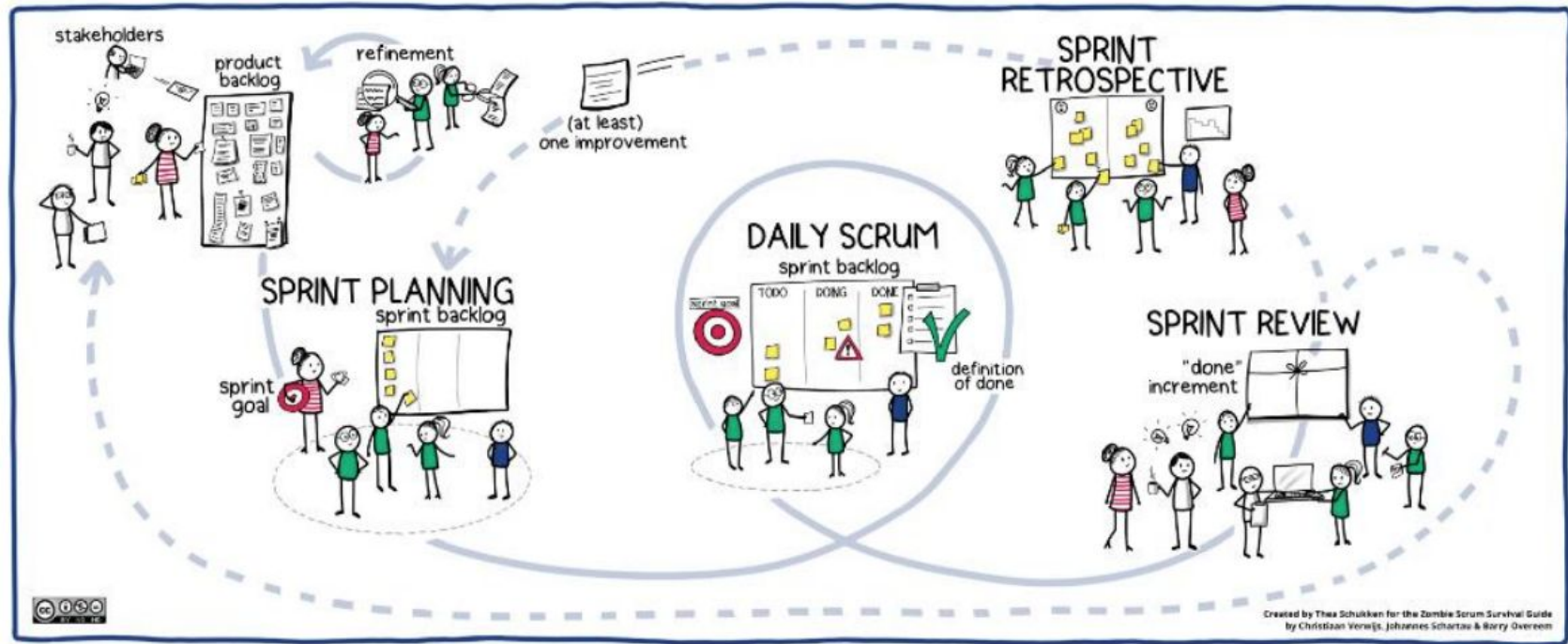
Revisão do Sprint

- 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"

Resumo em 1 slide



Mais alguns conceitos de Scrum

Time-box

- Atividades têm uma duração bem definida

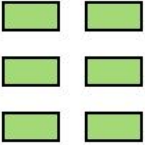
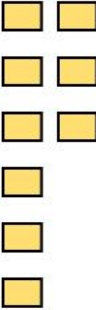

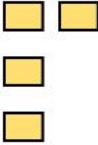

Evento	Time-box
Planejamento do Sprint	máximo de 8 horas
Sprint	menos de 1 mês
Reunião Diária	15 minutos
Revisão do Sprint	máximo de 4 horas
Retrospectiva	máximo de 3 horas

Critérios para Conclusão de Histórias (done criteria)

- Importante: considerar qualidade externa e interna
- Qualidade externa:
 - Testes de aceitação (caixa preta ou funcionais)
 - Testes não-funcionais (ex.: desempenho, usabilidade)
- Qualidade interna:
 - Testes de unidade
 - Revisão de código

Mais alguns artefatos de Scrum

Scrum Board

Backlog	To Do	Doing	Testing	Done
				

Scrum Board



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

Exemplo: GitLab (https://gitlab.com/gitlab-org/gitlab/-/boards)

The image displays three GitLab Kanban boards, each representing a different workflow or project phase. The boards are organized into columns and rows, with each row representing a task. Tasks are labeled with their titles, IDs, and various tags or labels indicating their category, priority, and status.

Board 1: Open (34472 items, 7577 members)

- Task 1:** Disable `ci_disable_validates_dependencies` after 10.3 RC3 release (#257852). Labels: Category:Continuous Integration, devops, verify, group, continuous integration, section, ops.
- Task 2:** Update doc/development/pipelines.md for graphql-verify job (#297030). Labels: Engineering Productivity, documentation, missed:13.8.
- Task 3:** Discuss: Using Dynamic Child Pipelines in GitLab Pipeline (#267491). Labels: Engineering Productivity, ep, pipeline, meta.
- Task 4:** Consider dynamic analysis test mapping to streamline test execution (#222369). Labels: Engineering Productivity, meta.

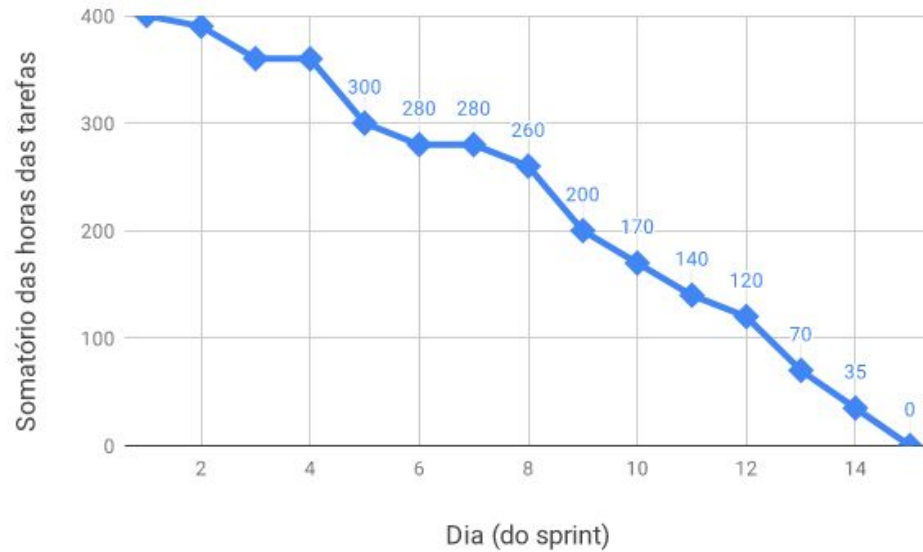
Board 2: workflow in dev (278 items, 418 members)

- Task 1:** Automated (pipeline) test planning for maintenance mode (#266992). Labels: Quality, auto updated, devops, enablement, geo, active, group, geo, missed-deliverable, missed:13.7, missed:13.8, section, enablement.
- Task 2:** Investigate: pipelines' behaviour in maintenance mode (#296533). Labels: Category:Disaster Recovery, Enterprise Edition, GitLab Premium, devops, enablement, geo, active, group, geo, section, enablement.
- Task 3:** BE: Retrieve details for individual Jira issues (#294155). Labels: Category:Integrations, Deliverable, Integration, Jira, atlassian, backend, customer, devops, create, feature, group, ecosystem, section, dev.
- Task 4:** JIRA integration doesn't use the authentication for the proxy provided. (#294155). Labels: Category:Integrations, Integration, Jira.

Board 3: workflow in review (134 / 1 items, 201 members)

- Task 1:** Make it clearer what to do after adding a namespace (#235909). Labels: Category:Integrations, Integration, Jira, Jira, ConnectApp, Technical Writing, devops, create, frontend, group, ecosystem, missed:13.7, priority, 4, section, dev.
- Task 2:** Display feature flag information in Jira (#229347). Labels: Category:Integrations, Deliverable, Integration, Jira, Jira, ConnectApp, atlassian, backend, devops, create, feature, group, ecosystem, missed-deliverable, missed:13.8, priority, 1, section, dev.
- Task 3:** Add a setting to allow/disallow duplicate Maven package uploads (#276882). Labels: Category:Package Registry, Deliverable, Package:P1, Stretch, backend, devops, package, feature, feature, addition, group, package, quad-planning, complete-action, ruby, section, ops.

Burndown chart



Estimativa de Tamanho de Histórias de Usuários

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

Velocidade de um time

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

Quando não usar métodos ágeis?

Adaptar ou não vale a pena usar métodos ágeis

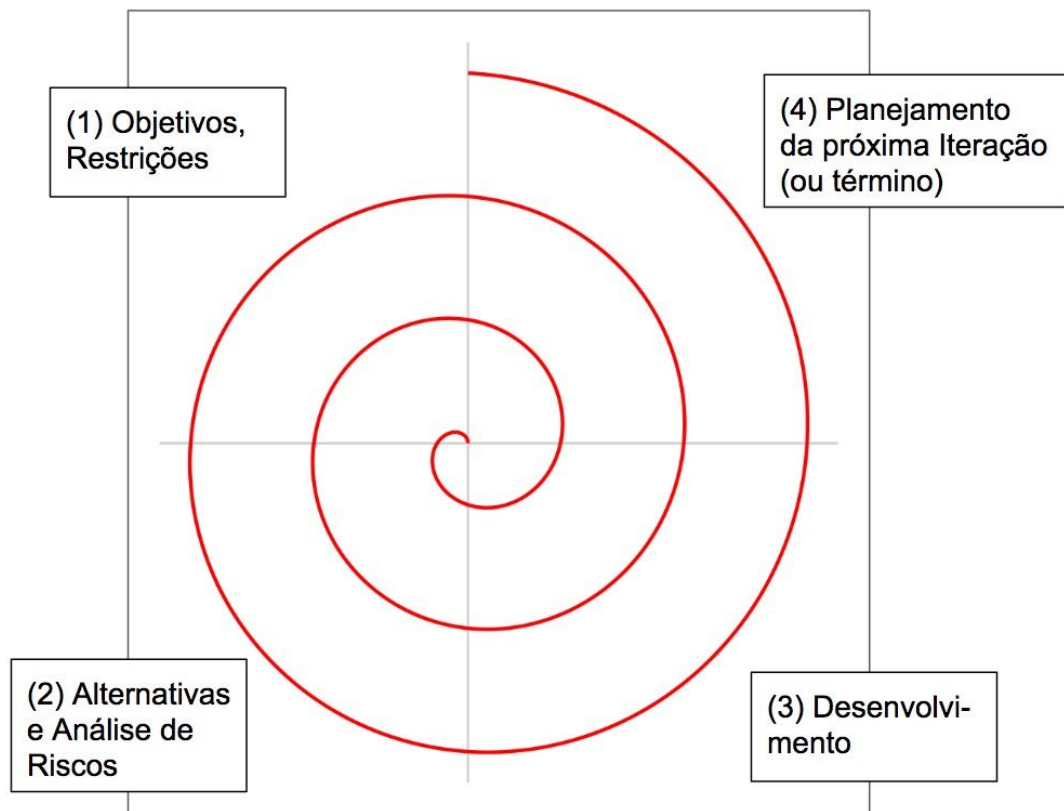
- Condições do mercado são estáveis e previsíveis
- Requisitos são claros no início do projeto e irão permanecer estáveis
- Clientes não estão disponíveis para colaboração frequente
- Sistema semelhante já foi feito antes; logo, já se conhece a solução
- Problemas podem ser resolvidos sequencialmente, em silos funcionais
- Clientes não conseguem testar partes do produto, antes de completo
- Mudanças no final do projeto são caras ou impossíveis
- Impacto de mudanças provisórias pode ser catastrófico

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



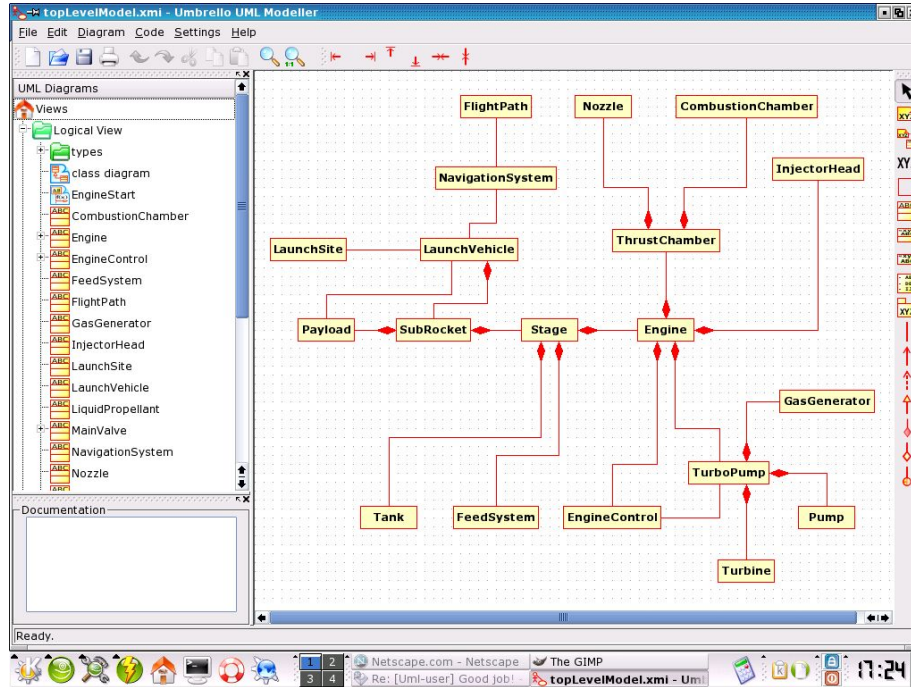
Proposto por Barry Boehm

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

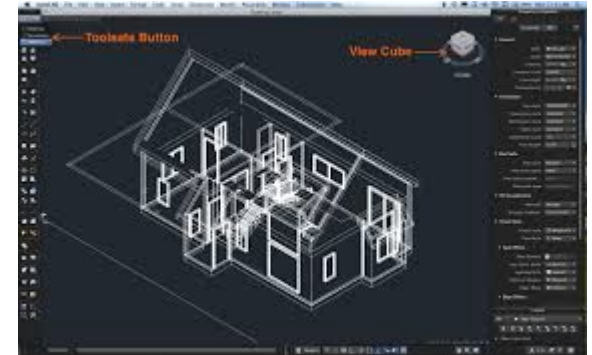
Rational Unified Process (RUP)

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

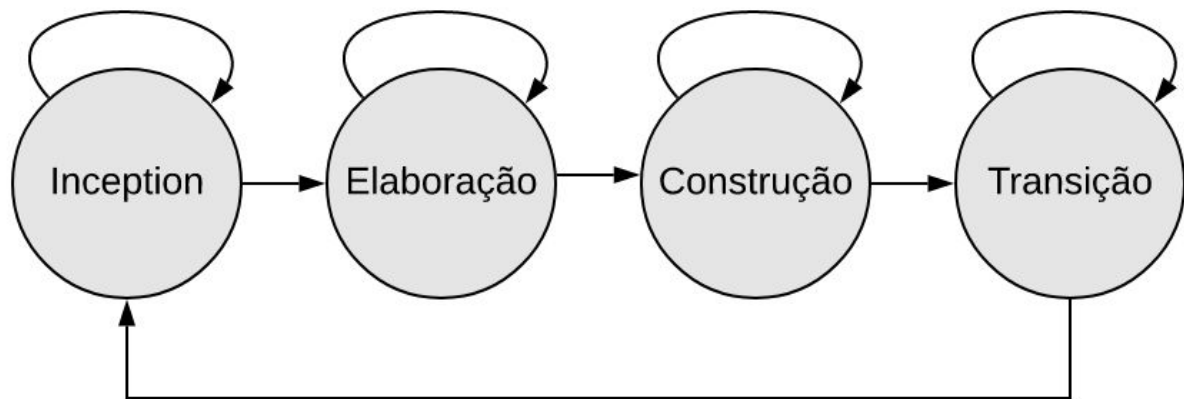
CASE (Computer-Aided Software Engineering)



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



Fases do RUP



- Inception: análise de viabilidade
- Elaboração: requisitos + arquitetura
- Construção: projeto de baixo nível + implementação
- Transição: implantação (deployment)

Engenharia de Software

Processos

Prof. Paulo Meirelles

paulo.meirelles@ufabc.edu.br