



Engenharia de Software

Processos de Desenvolvimento de Software

Aula de Hoje

- **Processo de Desenvolvimento de Software**
 - Conceitos Básicos
 - Atividades, tarefas, papéis (*roles*), artefatos, fluxo de trabalho...
 - Mudanças e Melhoria de Processo
- **Modelos de Processo de Software**
 - Waterfall - “Cascata” ou Sequencial
 - Desenvolvimento Incremental e/ou iterativo
 - Modelo Espiral
 - Metodologias Ágeis: Extreme Programming (XP), Scrum, Kanban e Lean
 - Integração e Configuração (Reúso)
 - Modelo de Processo de Negócio
 - *Business Process Model* (BPM); e
 - *Business Process Model Notation* (BPMN).



Aula de Hoje

- **Processo de Desenvolvimento de Software**
 - Conceitos Básicos
 - Atividades, tarefas, papéis (*roles*), artefatos, fluxo de trabalho...
 - Mudanças e Melhoria de Processo
- **Modelos de Processo de Software**
 - Waterfall - “Cascata” ou Sequencial
 - Desenvolvimento Incremental e/ou iterativo
 - Modelo Espiral
 - Metodologias Ágeis: Extreme Programming (XP), Scrum, Kanban e Lean
 - Integração e Configuração (Reúso)
 - Modelo de Processo de Negócio
 - *Business Process Model* (BPM); e
 - *Business Process Model Notation* (BPMN).

Processo de Entrega



<https://unsplash.com/>

Atividades e Tarefas bem definidas!

Introdução: Engenharia Tradicional

- Logística, engenharia civil, mecânica, elétrica, mecatrônica, automobilística e aviação
- Duas características
 - Planejamento detalhado (*big upfront design*)
 - Sequencial

Introdução: Software é diferente

- **Engenharia de Software (ES)** ≠ Engenharia Tradicional
- **Software** ≠ carro, ponte, casa, avião, celular
- **Software** ≠ produtos físicos
- **Software** é abstrato e "adaptável"

Motivação: Processo de Software

- Como **estruturar** o projeto?
- Quais são as **atividades** e fases? (análise, projeto, desen., teste...)
- Como organizar a **comunicação**?
- Quem tem quais **responsabilidades**?
- **Esquecemos** de algo?
- Podemos **prever** o resultado do projeto?
- Como **gerenciar e controlar** o progresso?
- Como compartilhar e obter **experiência**?
- Como sincronizar **hardware e software** desenvolvimento?

Processo é uma receita?



Atividades e Tarefas bem definidas!



<https://unsplash.com/>

Motivação: Processo de Software

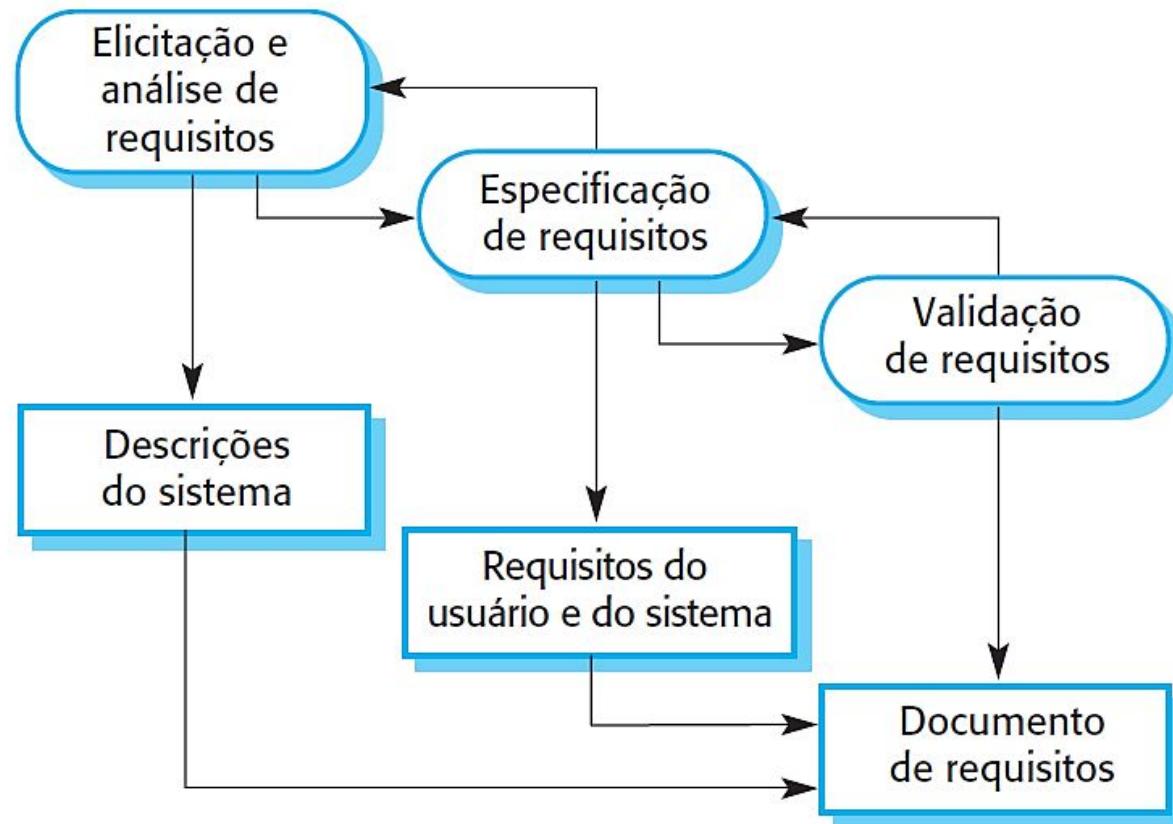
- É necessário considerar processos diferentes
- Diferentes organizações desenvolvem softwares distintos
- Requisitos mudam conforme o contexto
- Stakeholders são heterogêneos
- O processo descrito inclui
 - **Produtos** - saída de uma atividade do processo
 - **Papeis (Roles)** - pessoas envolvidas e suas responsabilidades
 - **Pré e Pós Condições** - suposições verdadeiras antes e depois de uma actividade

Processo de Software: Conceitos Básicos

- Os processos de software reais são um conjunto de atividades estruturadas para desenvolver um sistema de software
 - atividades, tarefas, papéis e um fluxo de trabalho...
- As atividades são colaborativas e gerenciais = objetivo de:
 - especificar, projetar, implementar e testar um sistema de software
- Basicamente, um processo básico possui as atividades
 - Especificação - o que o sistema faz
 - Projeto e Implementação - organização e desenvolvimento
 - Validação - o que o cliente quer
 - Evolução - mudanças e responde as necessidades do cliente

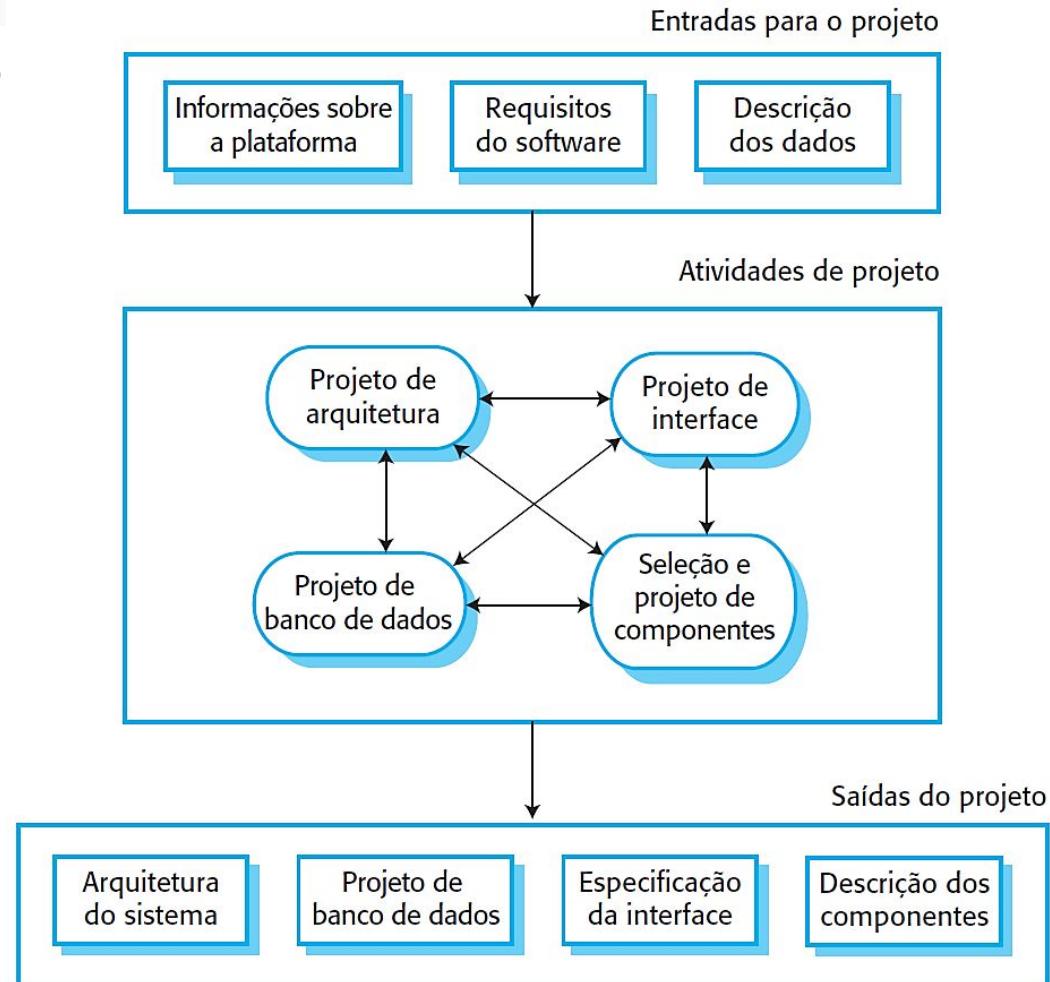
Especificação

- O processo de Engenharia de Requisitos



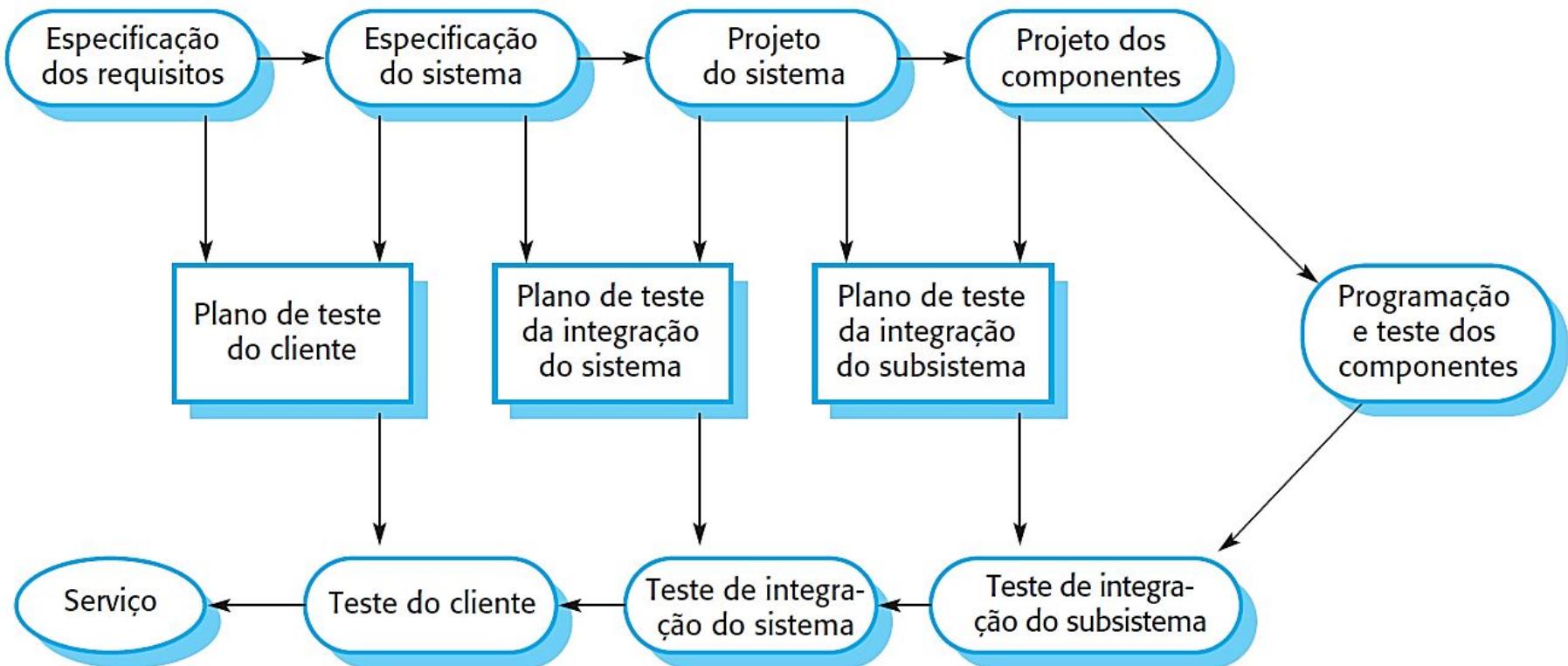
Desenvolvimento (Projeto e Implementação)

- Processo de Projeto



Validação (V-Model)

- Atividades de teste em um processo dirigido por plano



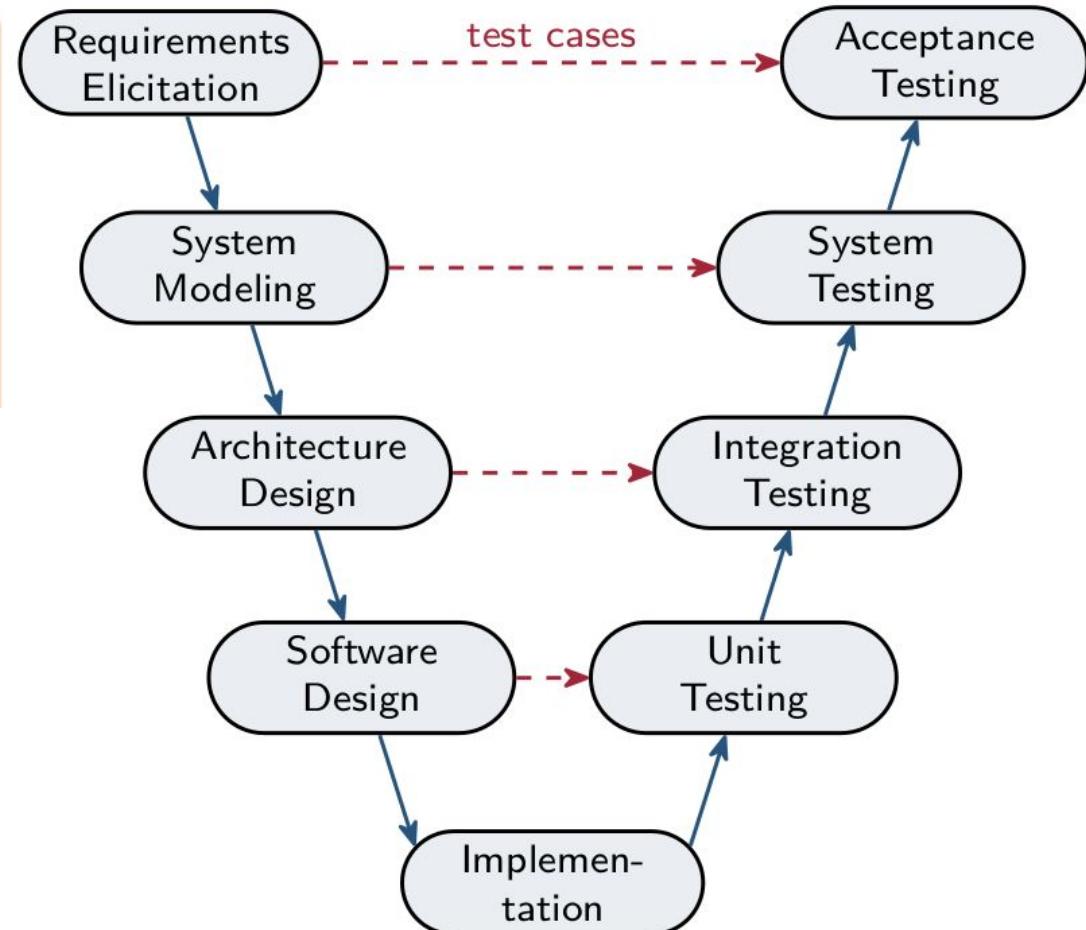
Exemplo: Original V-Model

V-Model

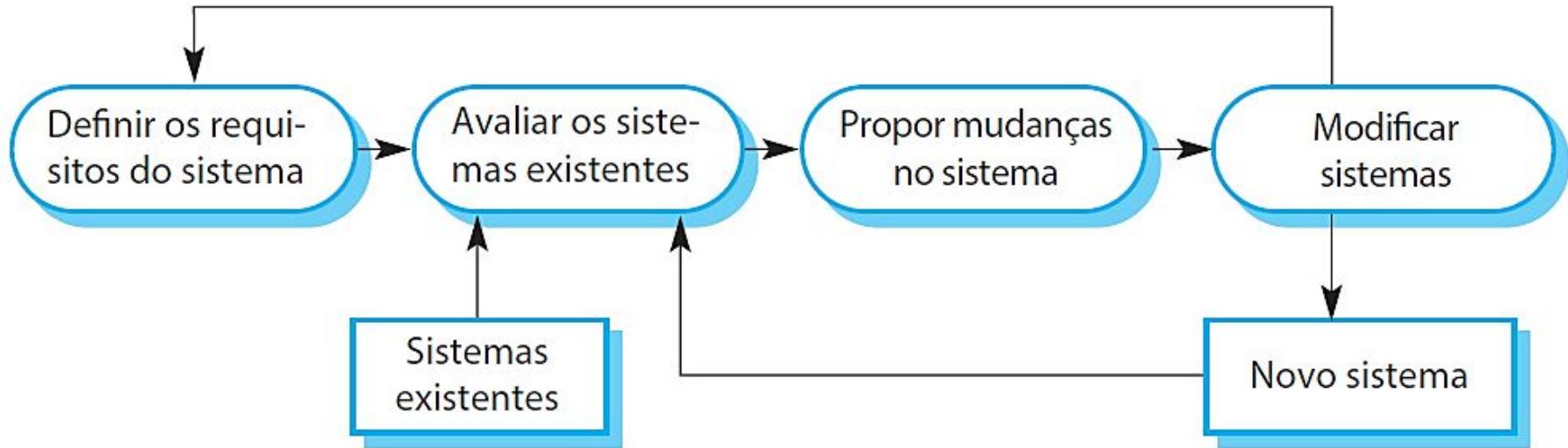
[Ludewig and Licher]

- developed by the German Ministry of Defense ([Verteidigungsministerium](#)) and required since 1992
- extension of the waterfall model: project-aligned activities such as quality assurance, configuration management, project management

(Thüm, 2022)



Evolução

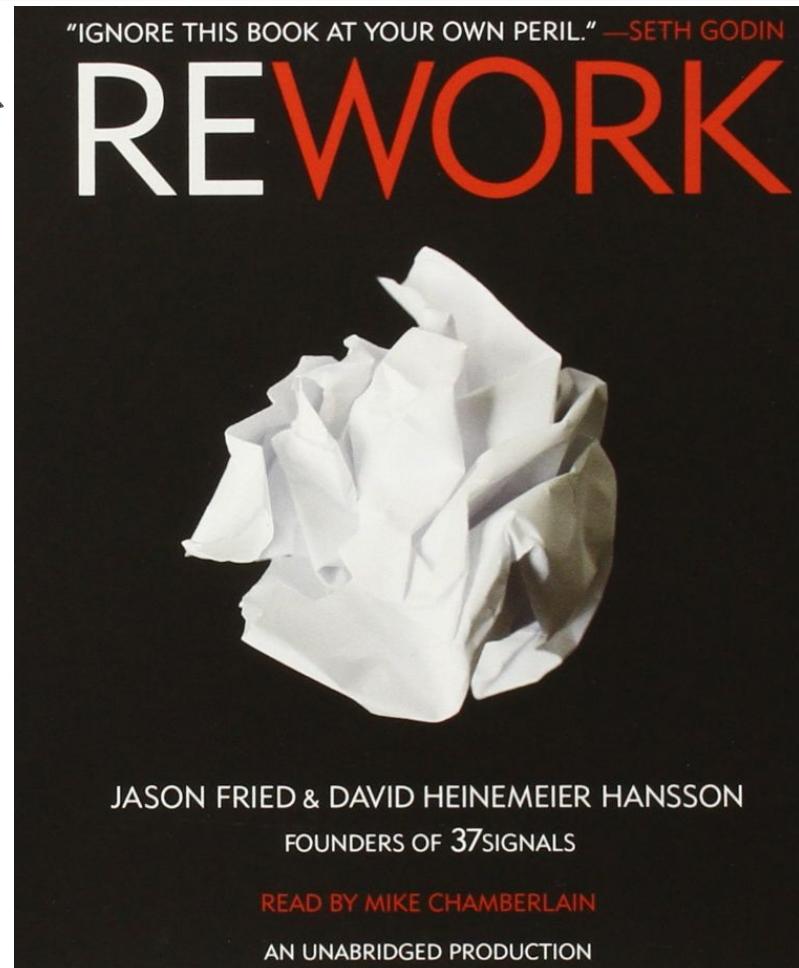


Dúvidas

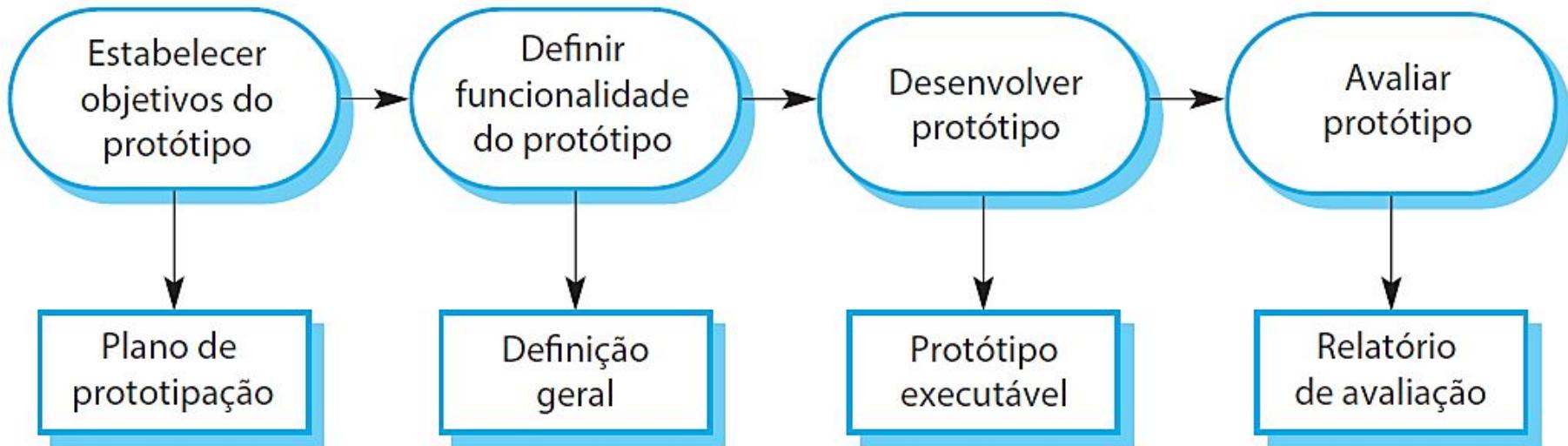


Mudanças

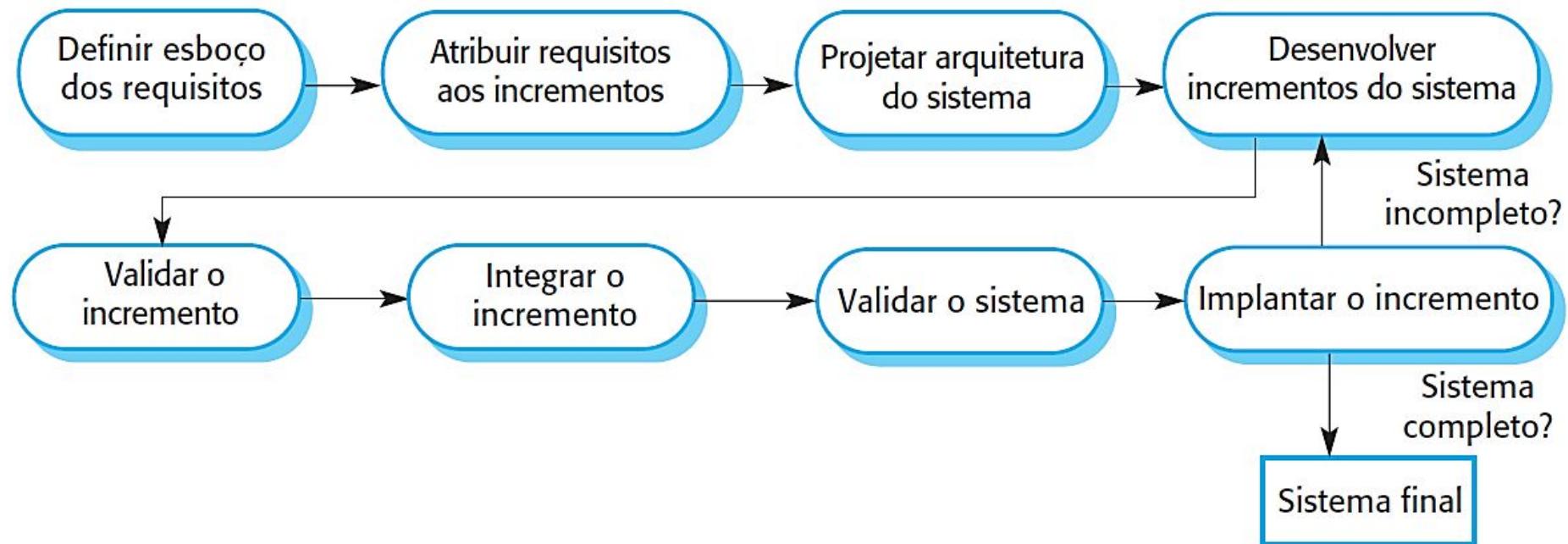
- Reduzir os custos de Retrabalho
 - Antecipação a mudança
 - Tolerância à mudança
- Lidar com mudanças e variações
 - Prototipação
 - Entrega incremental



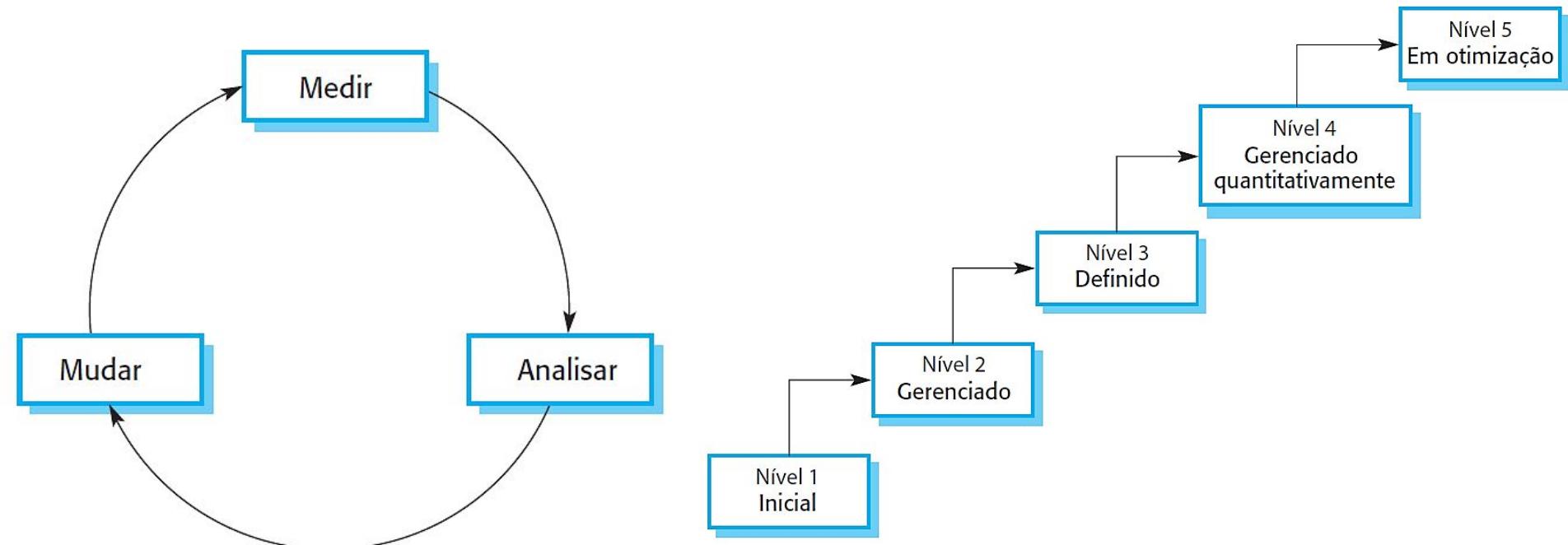
Mudanças: Prototipação



Mudanças: Entrega incremental



Melhoria de Processo: Ciclo e Maturidade



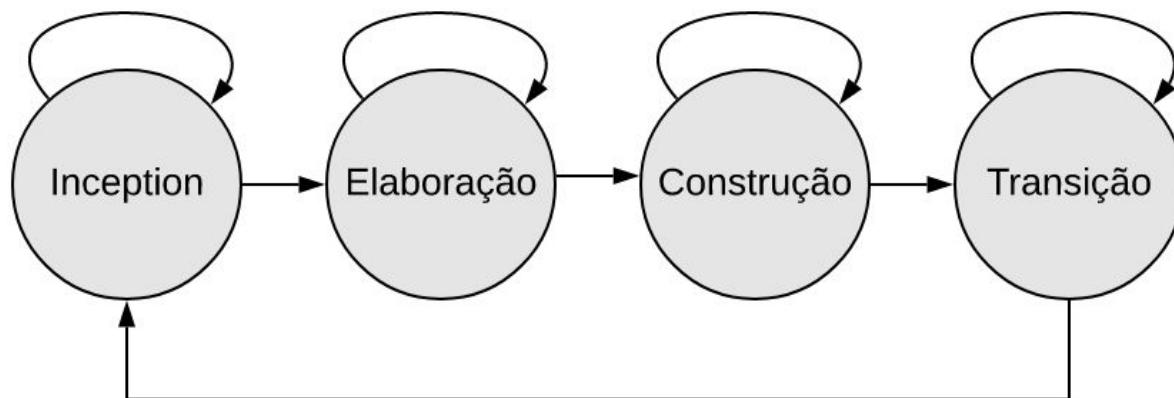


Aula de Hoje

- **Processo de Desenvolvimento de Software**
 - Conceitos Básicos
 - Atividades, tarefas, papéis (*roles*), artefatos, fluxo de trabalho...
 - Mudanças e Melhoria de Processo
- **Modelos de Processo de Software**
 - Waterfall - “Cascata” ou Sequencial
 - Desenvolvimento Incremental e/ou iterativo
 - Modelo Espiral
 - Metodologias Ágeis: Extreme Programming (XP), Scrum, Kanban e Lean
 - Integração e Configuração (Reúso)
 - Modelo de Processo de Negócio
 - *Business Process Model* (BPM); e
 - *Business Process Model Notation* (BPMN).

Um pouco de história... Rational Unified Process (RUP)

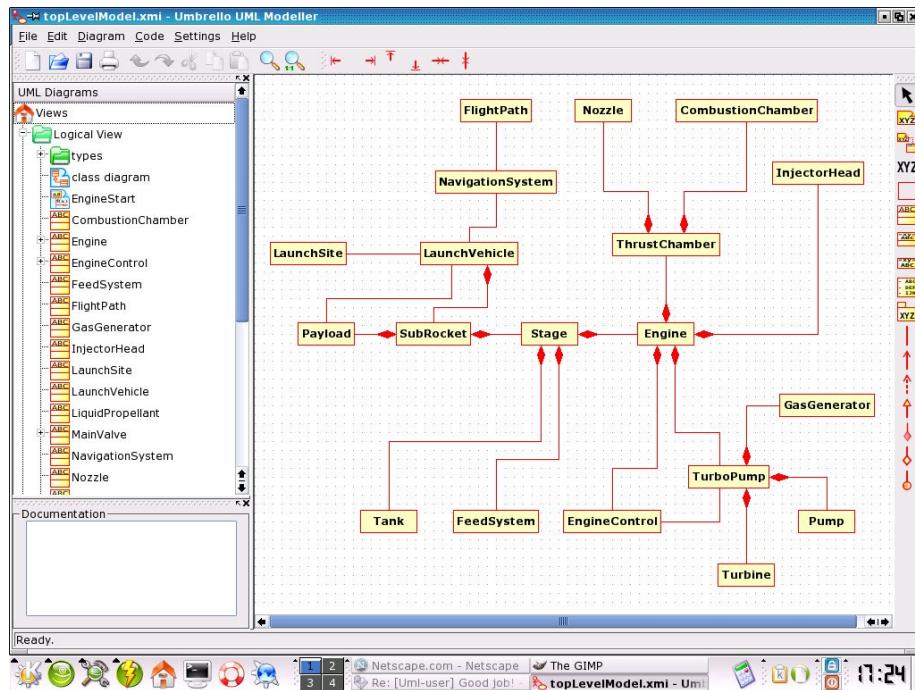
- Proposto pela Rational (IBM). Surge CASE -> UML...



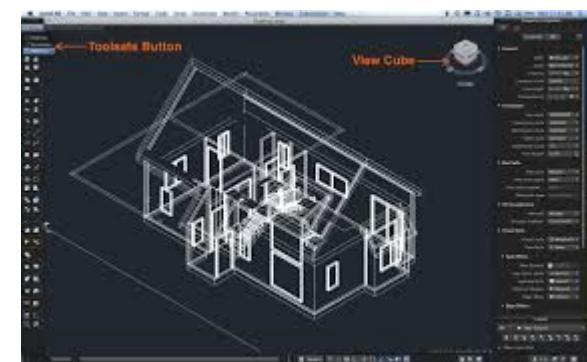
LEGADO!

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

História: Computer-Aided Software Engineering (CASE)



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

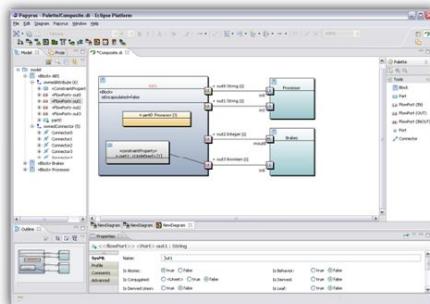


Ferramentas CASE - Open-Source



Applications Technologies

Eclipse Papyrus™ Modeling environment



News

Features

Download

Support

Screenshots

Document

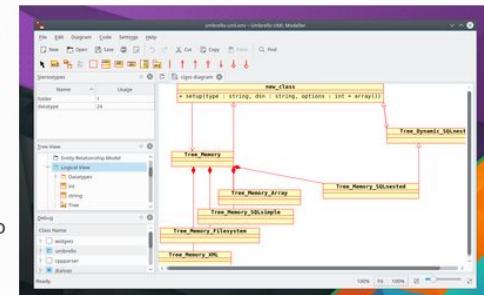
Welcome to Umbrello - The UML Modeller

Umbrello UML Modeller is a Unified Modelling Language (UML) diagram program based on KDE Technology.

UML allows you to create diagrams of software and other systems in a standard format to document or design the structure of your programs.

You may take a look at the [screenshots](#) to see umbrello in action.

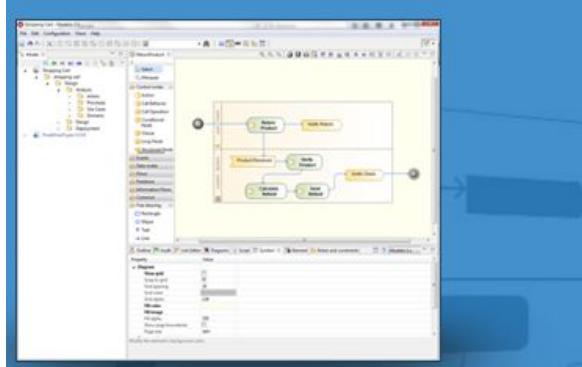
Our [handbook](#) gives a good introduction to Umbrello



MODELIO

The open source extensible modeling environment
supporting:

UML, BPMN, ArchiMate, SysML,



Ferramentas CASE - Comerciais



Products ▾ Pricing ▾ Support ▾ Resources ▾ UML Tutorial Community Download Now ▾

Search...



Login

ENTERPRISE ARCHITECT

Model | Design | Construct | Test | Deploy | Manage

Learn more

Download Now

Pricing

Current Version: 16.1

Ferramentas CASE - Comerciais



MAGICDRAW

AWARD-WINNING BUSINESS PROCESS, ARCHITECTURE, SOFTWARE AND SYSTEM MODELING TOOL WITH TEAMWORK SUPPORT

LOGIN

Designed for Business Analysts, Software Analysts, Programmers, QA Engineers, and Documentation Writers, this dynamic and versatile development tool facilitates analysis and design of Object Oriented (OO) systems and databases. It provides the industry's best code engineering mechanism (with full round-trip support for Java, C++, CL (MSIL) and CORBA IDL programming languages), as well as database schema modeling, DDL generation and reverse engineering facilities.

Intro

Key Benefits

Screenshots

Overview

Editions

Industry standards-compliance and support

MagicDraw supports the UML 2 metamodel, the latest XMI standard for data storage and the most popular programming languages for implementation.

Unlike other UML modeling and architecture environments, MagicDraw makes it easy for you to deploy a Software Development Life Cycle (SDLC) environment that best suits the needs of your business. Our approach to standards and our Open API makes it easy for you to integrate with applications that work together, best supporting the needs of your business. We integrate with many leading products: IDEs, requirements, testing, estimation, MDD, database, and others.



Ferramentas CASE - Comerciais

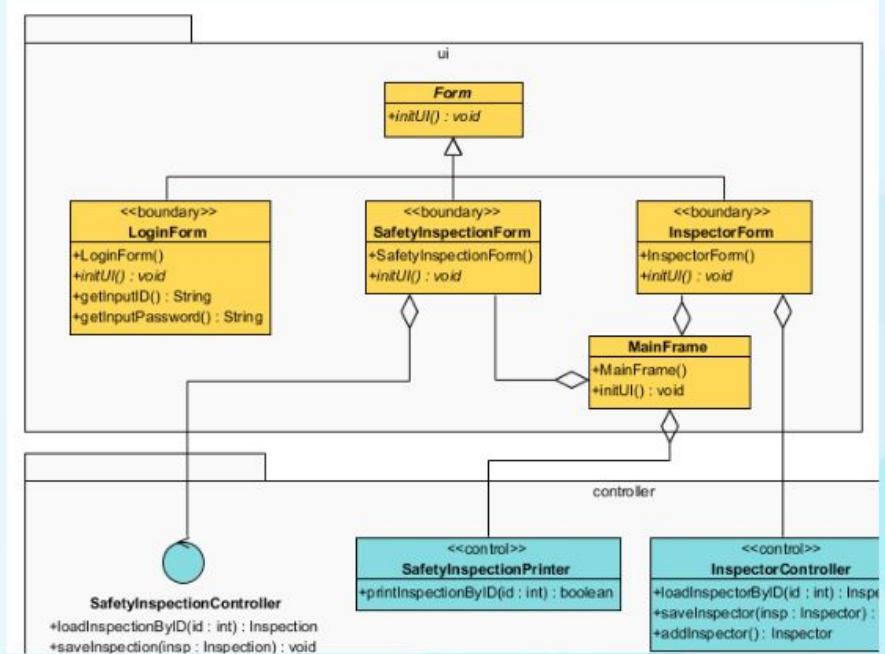


What's New Features ▾ Tutorials Support Pricing Try Now Request Demo VP Online ▾

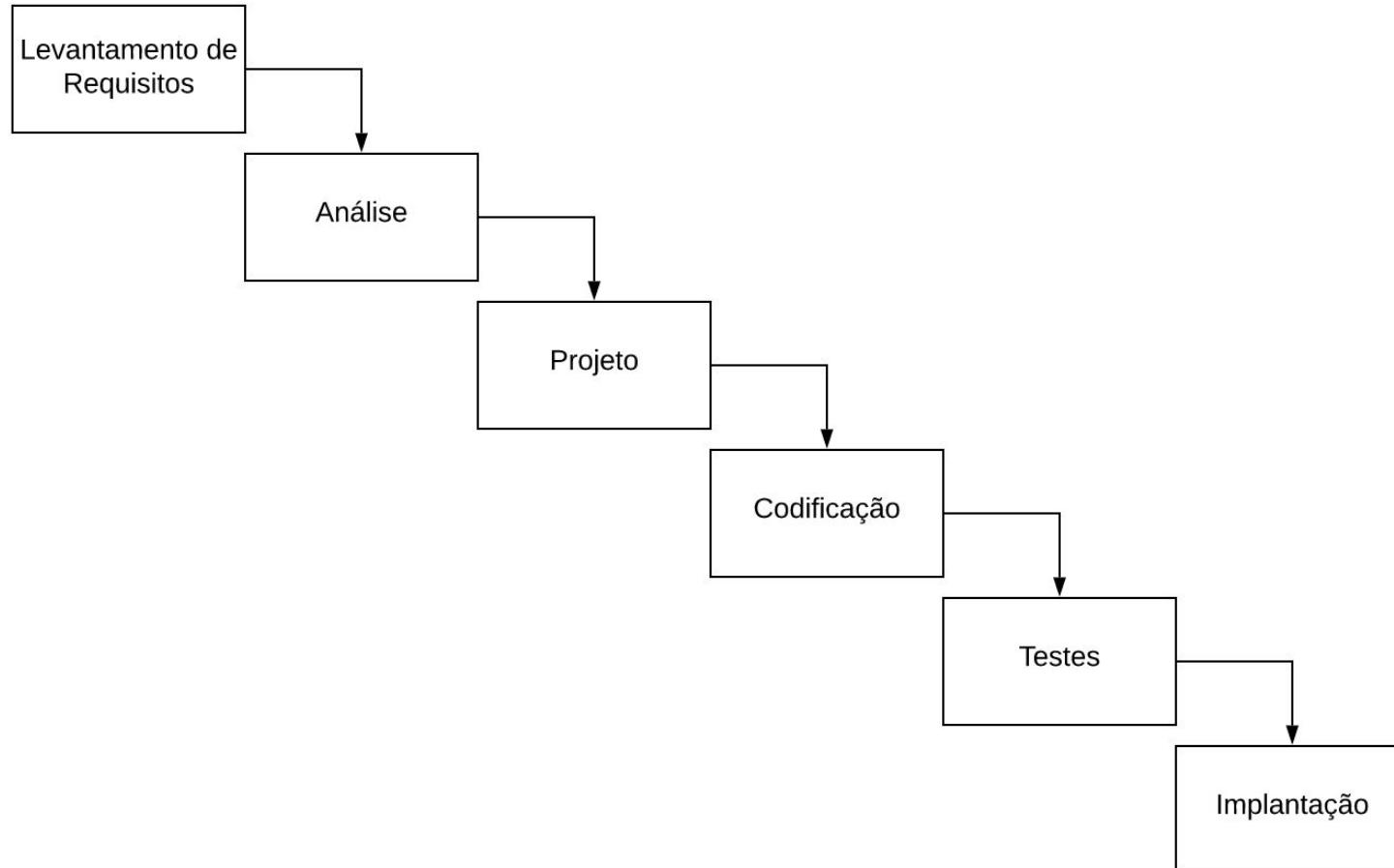
The #1 Development Tool Suite

that drives your project to success

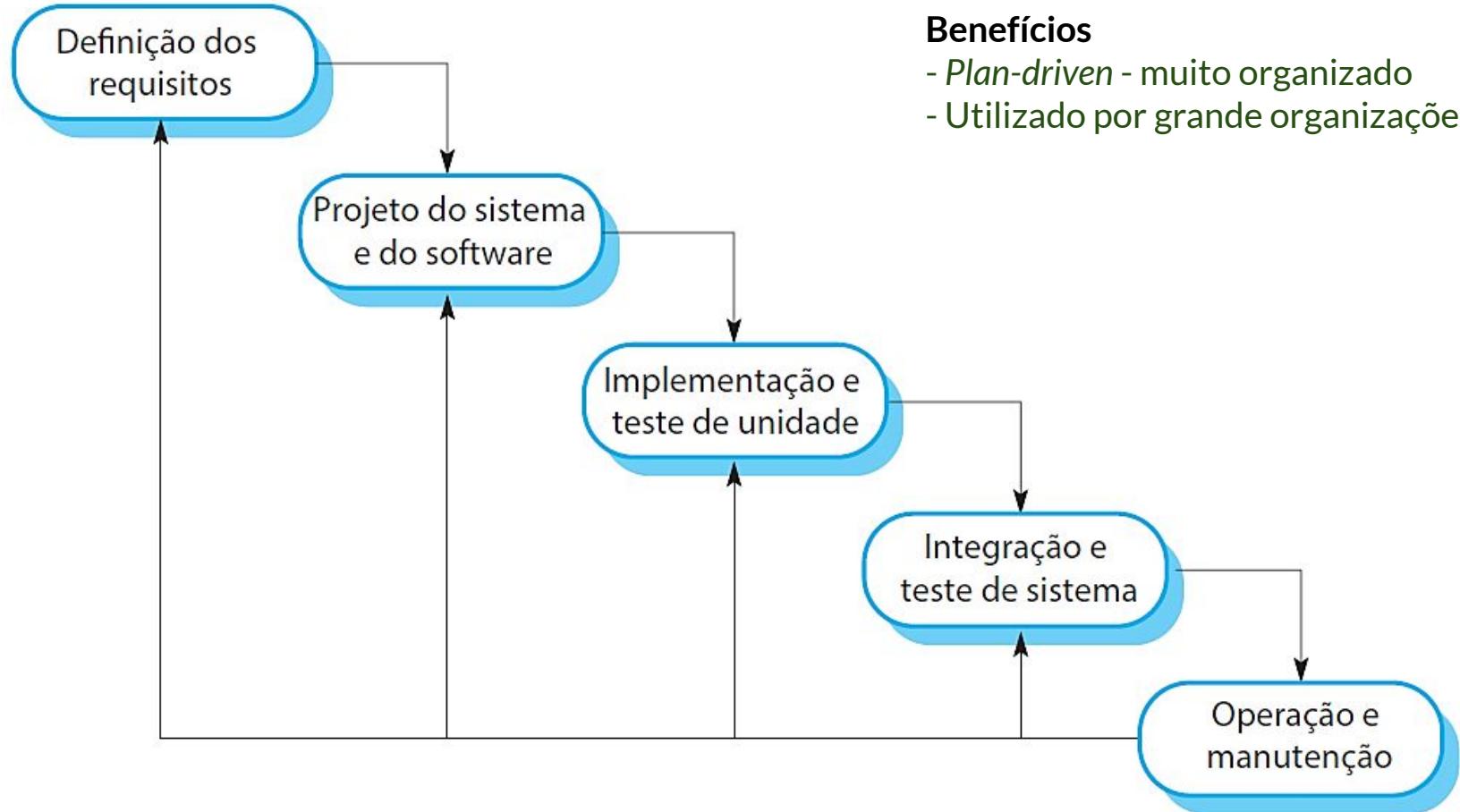
A suite of design, analysis and management tools to drive your IT project development and digital transformation.



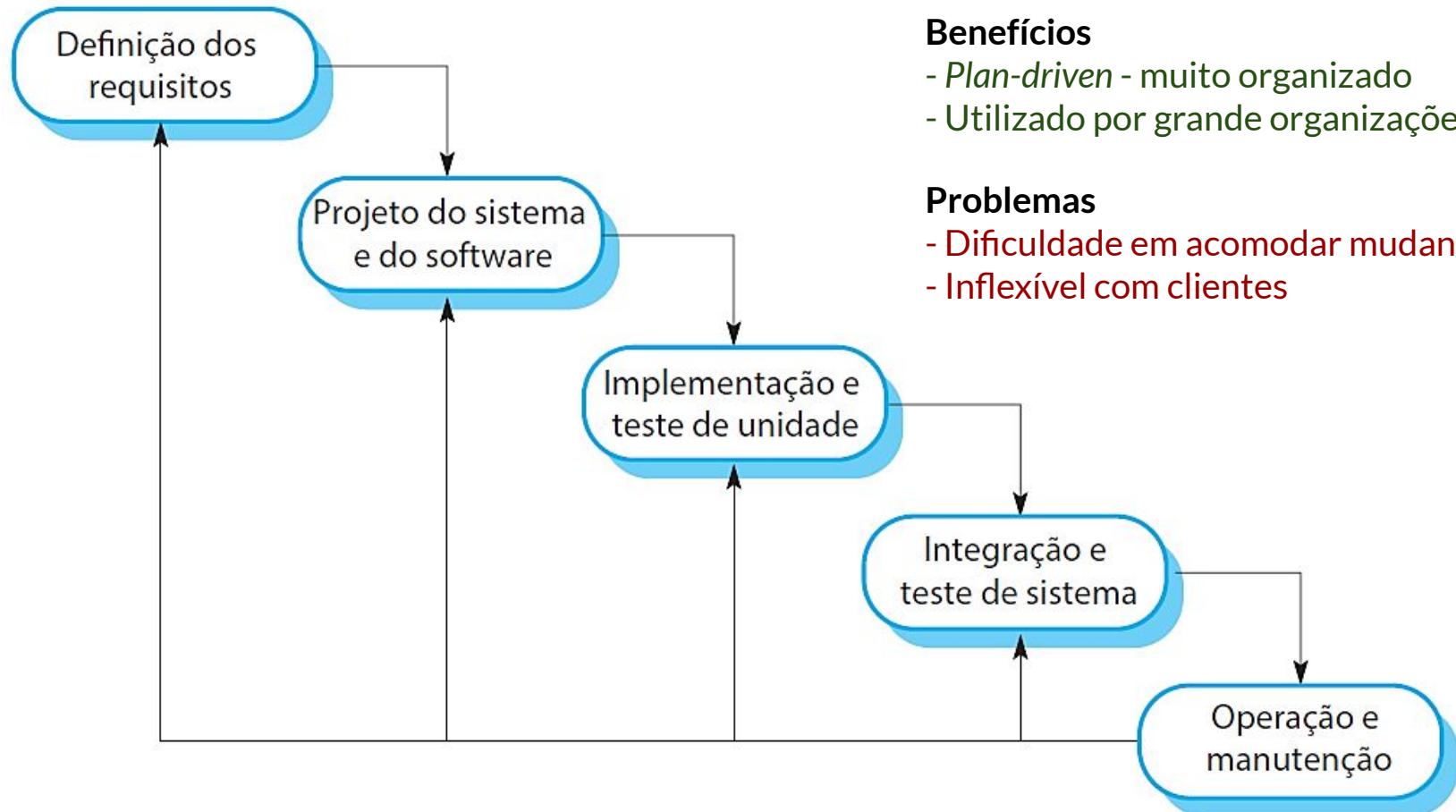
Natural que ES começasse com: Waterfall -> “Cascata” ou Sequencial



Modelos de Processo de Software: Waterfall



Modelos de Processo de Software: Waterfall



Benefícios

- *Plan-driven* - muito organizado
- Utilizado por grande organizações

Problemas

- Dificuldade em acomodar mudanças
- Inflexível com clientes



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
- Quando o software ficar pronto, ele estará obsoleto (legado)!

Dificuldade 2: Documentações Detalhadas

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



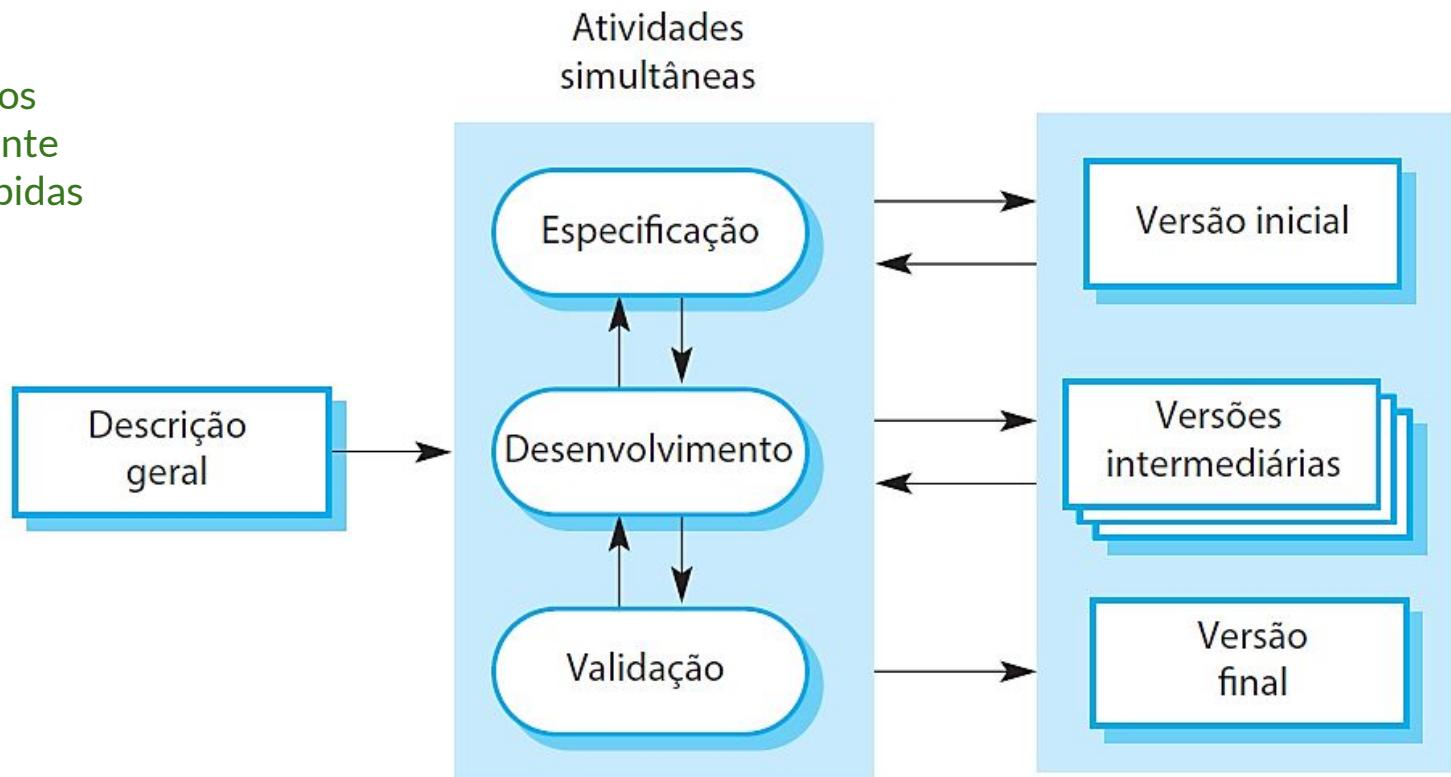
Dúvidas



Modelos de Processo de Software: Incremental

Benefícios

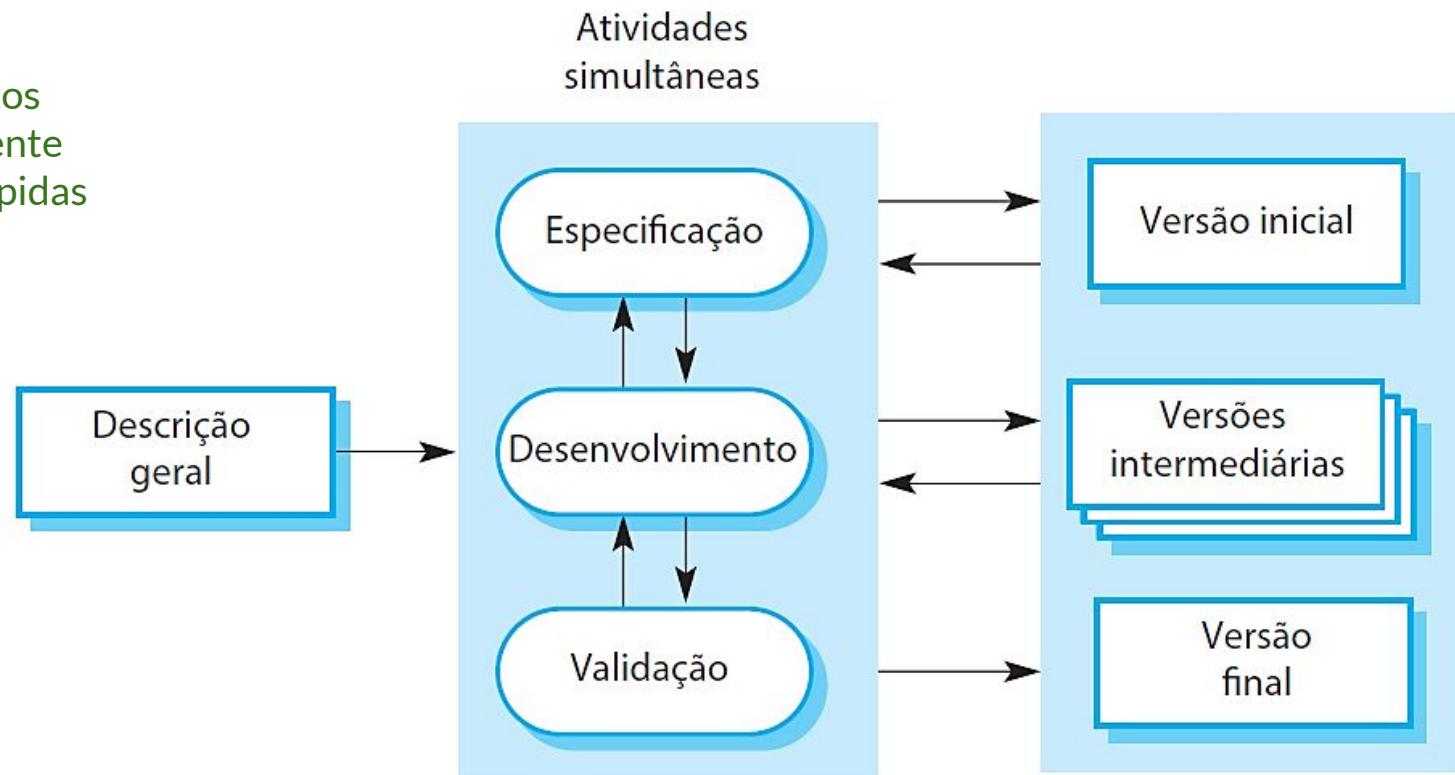
- Redução de Custos
- Feedback do Cliente
- Entregas mais rápidas



Modelos de Processo de Software: Incremental

Benefícios

- Redução de Custos
- Feedback do Cliente
- Entregas mais rápidas



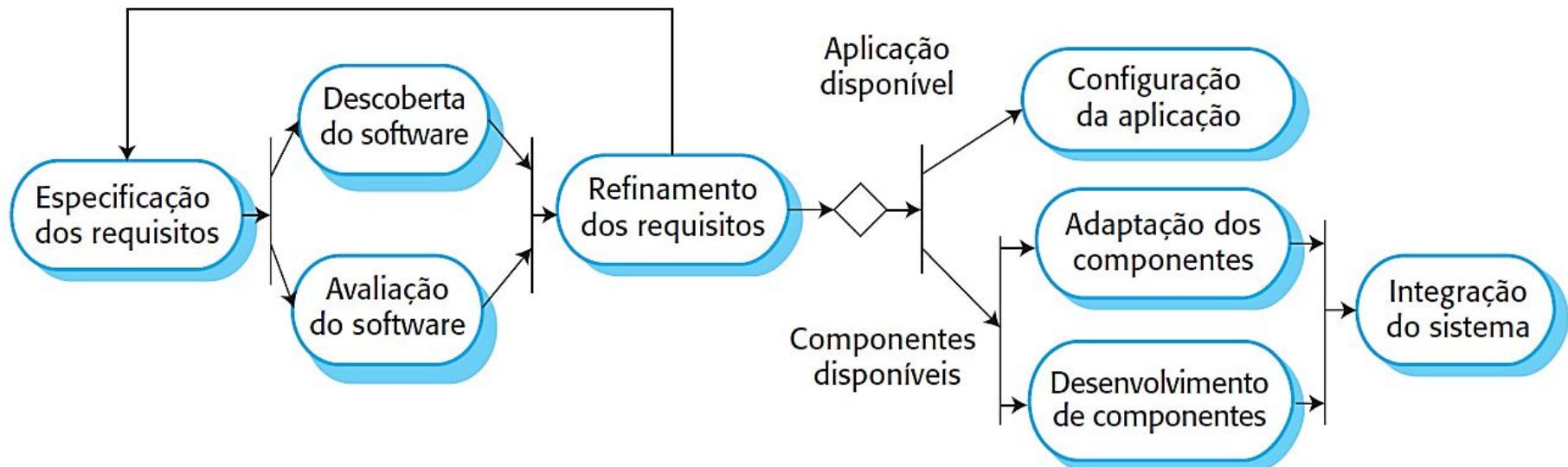
Problemas

- Processo não é visível (muitas versões)
- Degradação do sistema com novos incrementos

Modelos de Processo de Software: Reúso

Benefícios

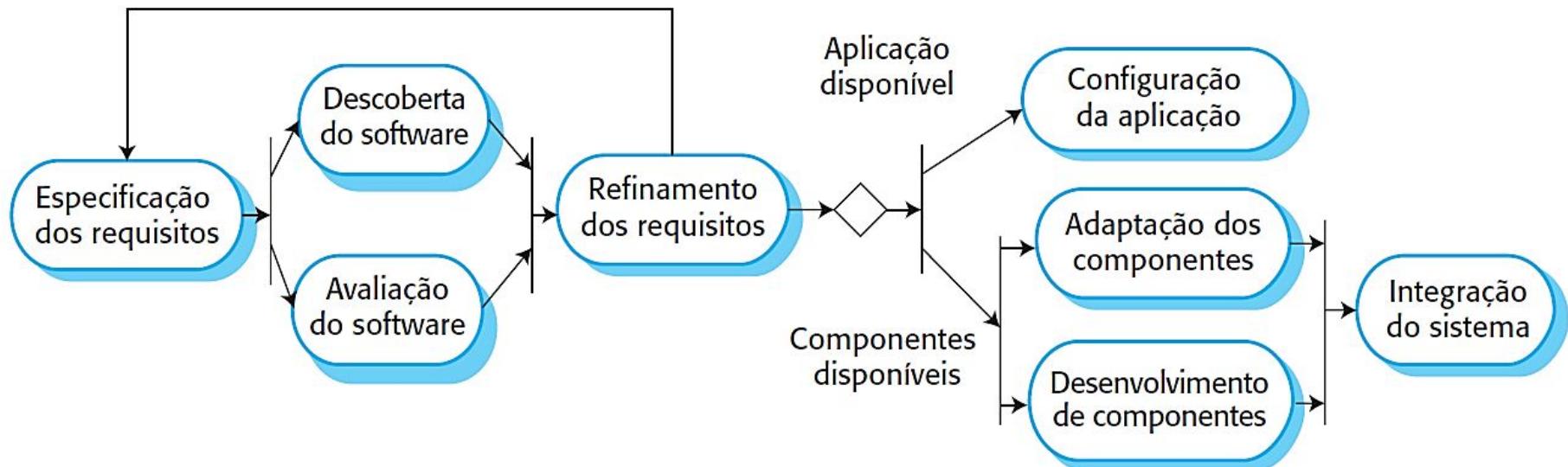
- Reduz os custos e riscos
- Entrega e implantação rápida do sistema



Modelos de Processo de Software: Reúso

Benefícios

- Reduz os custos e riscos
- Entrega e implantação rápida do sistema



Problemas

- Requisitos podem ser comprometidos
- Controle ruim da evolução dos elementos reutilizados

Transição -> 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) (**bem útil desenvolvimento web - em desuso**)
 - Rational Unified Process (RUP) (2003) (**LEGADO!**)

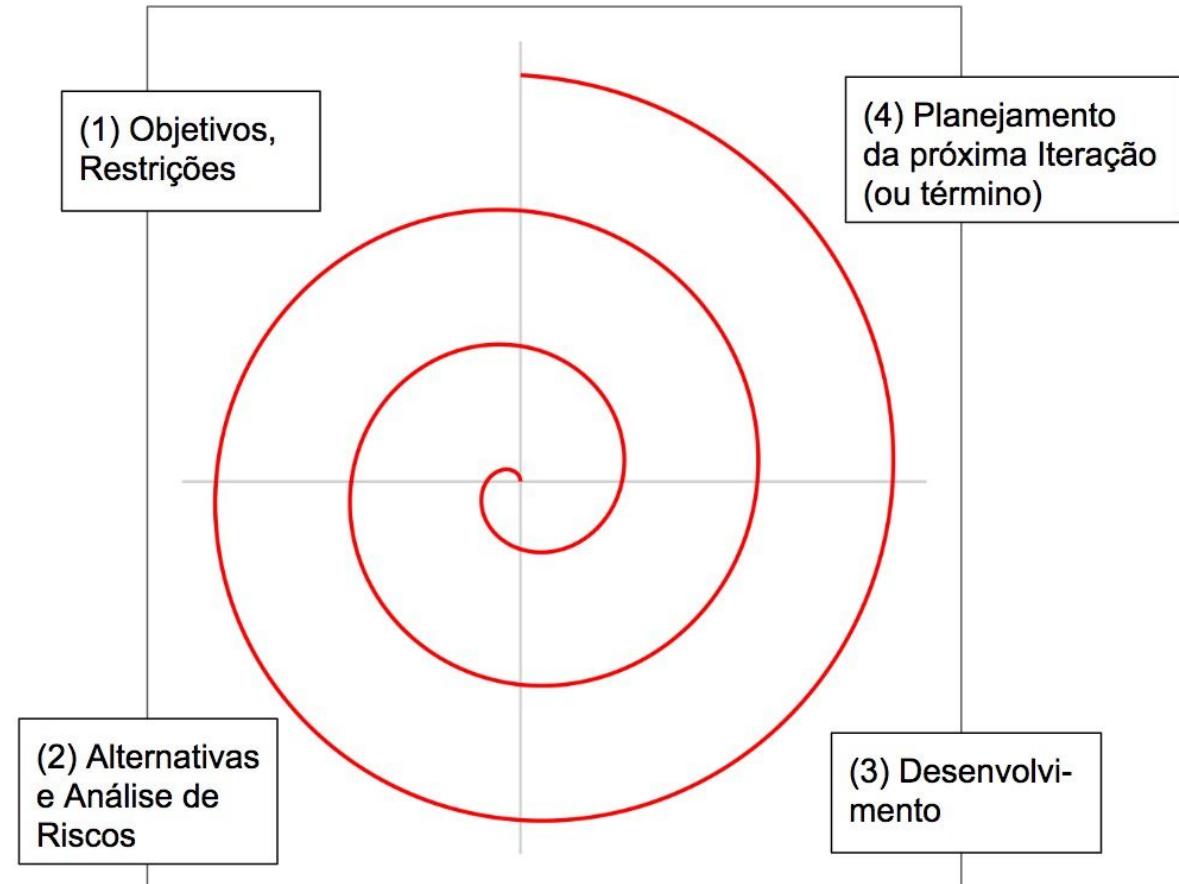
Modelo em Espiral

Proposto por Barry Boehm

Iterações:

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

LEGADO!



Surge o Manifesto Ágil (2001)

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.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler

James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick

Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

<https://agilemanifesto.org/iso/ptbr/manifesto.html>

© 2001, os autores acima.
esta declaração pode ser copiada livremente em qualquer formato,
mas somente integralmente através desta declaração.

Surge o Manifesto Ágil (2001)

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

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

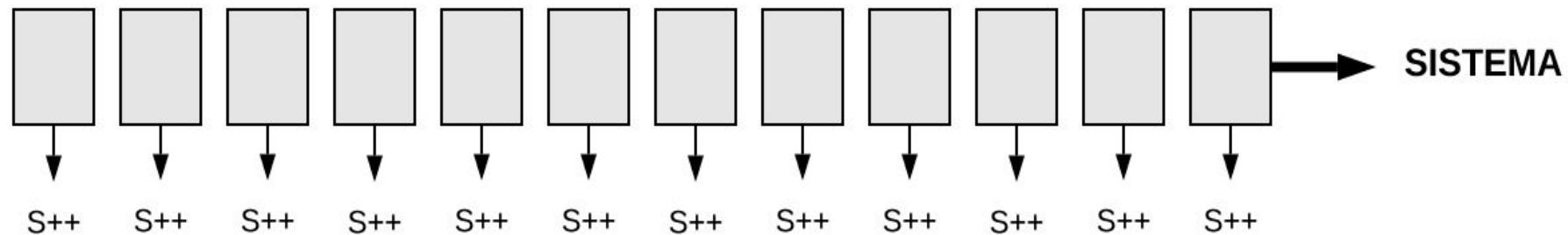
<https://agilemanifesto.org/iso/ptbr manifesto.html>

Ideia Central: Desenvolvimento Iterativo

Waterfall (Cascata ou Sequencial)



Ágil





Desenvolvimento Iterativo

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



Pontos importantes

- Menor ênfase em documentação
- Menor ênfase em *big upfront design* (projetar primeiro antes de executar)
- Envolvimento constante do cliente
- Novas práticas de implementação
 - Testes, *refactoring*, integração contínua, entre outras

Fizemos a seguinte pergunta para 415 devs brasileiros

Qual é o processo de desenvolvimento usado por sua empresa?

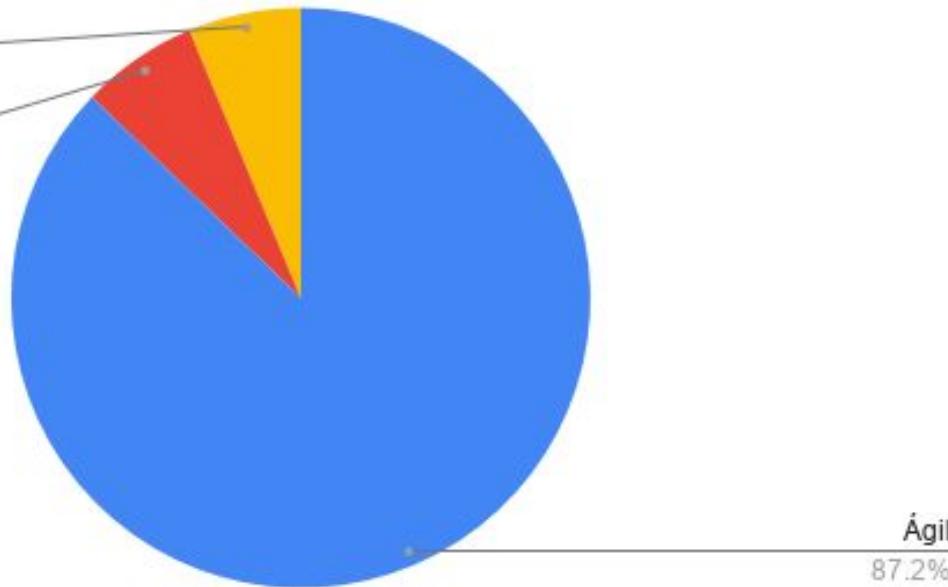
Resultados relativos a 415 respostas de desenvolvedores de software brasileiros

Outros

6.3%

Waterfall

6.5%



Fonte: Surveying the Impacts of COVID-19 on the Perceived Productivity of Brazilian Software Developers. SBES 2020

Antes de começar: Métodos Ágeis

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

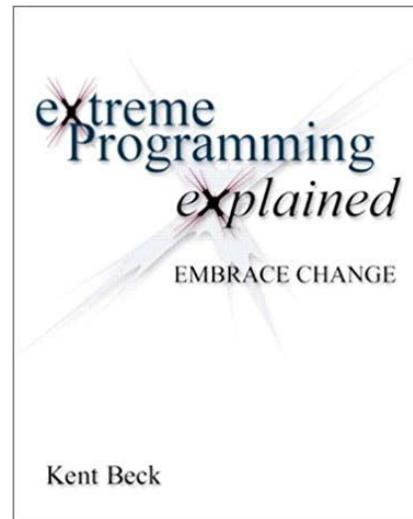
Principais Métodos Ágeis (Iterativos)

- Extreme Programming (XP)
- Scrum
- Kanban
- Lean
- Consistentes às ideias ágeis
 - Definem um processo, mesmo que leve
 - *Workflow*, eventos, papéis, práticas e princípios

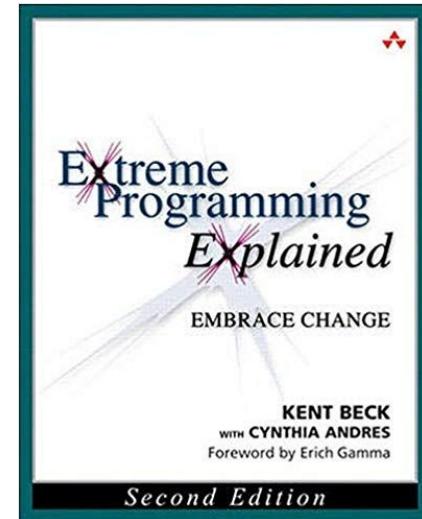
Dúvidas



Extreme Programming (XP)



1999

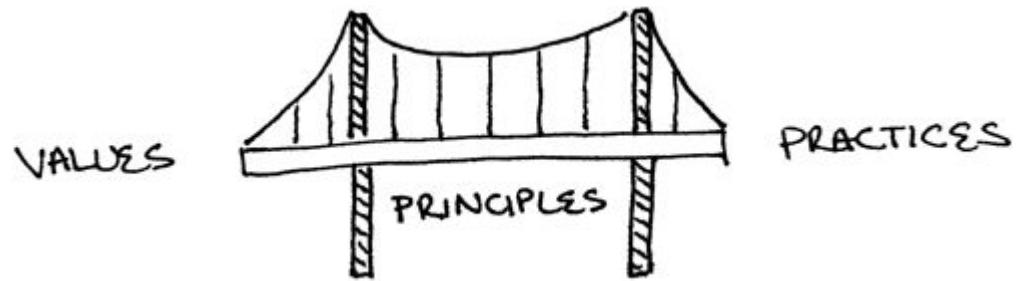


2004

Kent Beck

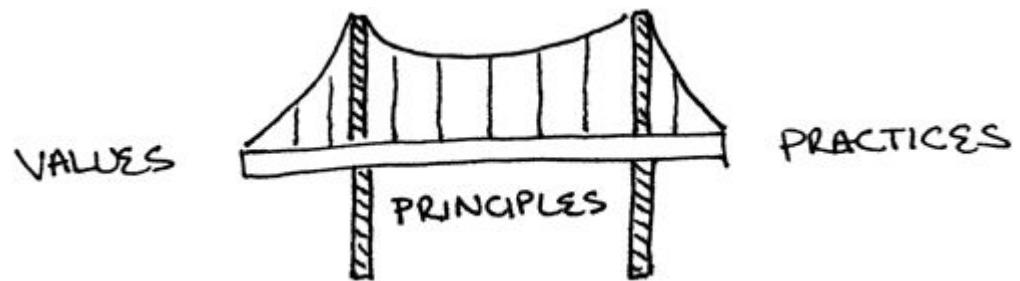
XP = Valores + Princípios + Práticas

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



XP = Valores + Princípios + Práticas

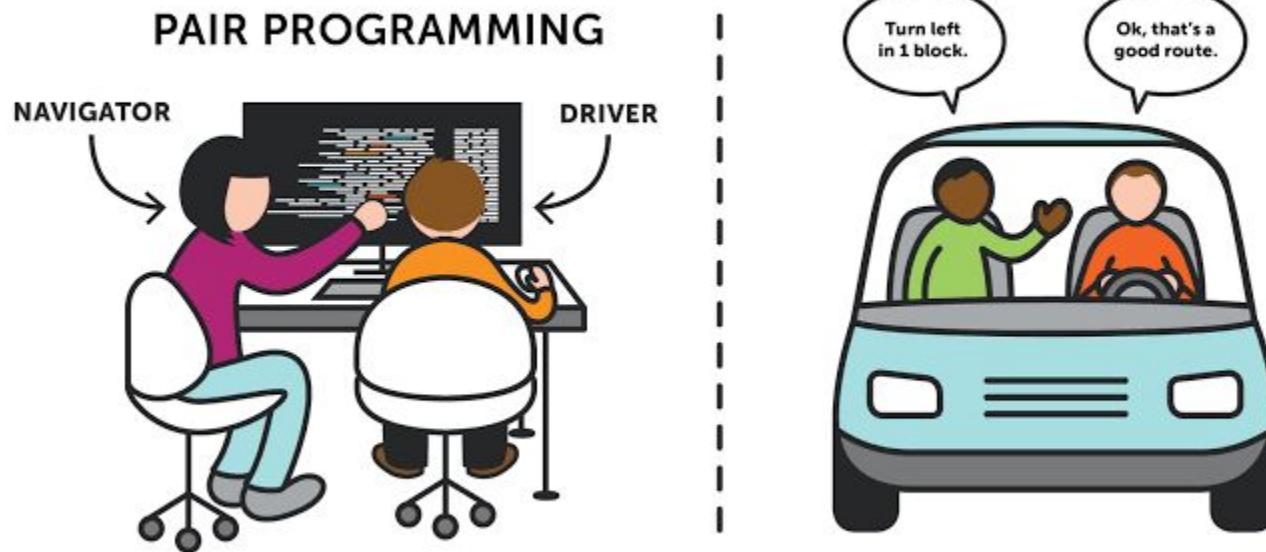
- Economicidade
- Melhorias Contínuas
- Falhas Acontecem
- Baby Steps
- Responsabilidade Pessoal



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

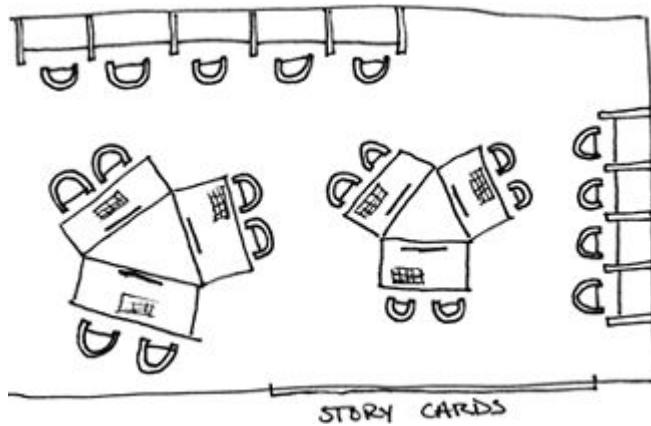
Pair Programming



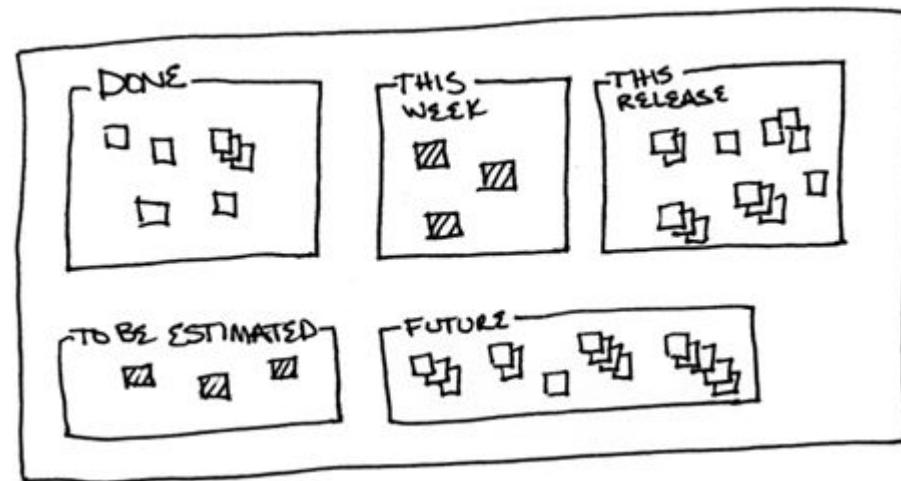
Estudo com Engenheiros da Microsoft (2008)

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

XP: Ambiente de Trabalho



Ambiente de trabalho



Cartazes para "visualizar trabalho" em andamento



Contratos de Software

- Dois tipos de contratos:
 - **Escopo Fechado**
 - Cliente define requisitos
 - Empresa desenvolvedora: preço + prazo
 - **Escopo Aberto (defendidos por XP)**
 - Escopo definido a cada iteração
 - Pagamento por homem/hora
 - Contrato renovado a cada iteração

XP: Contratos com Escopo Aberto

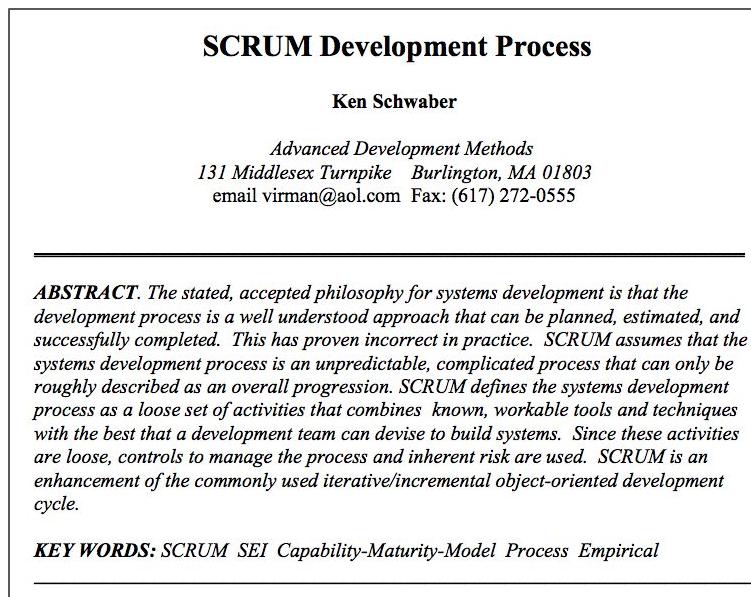
- Exige maturidade e acompanhamento do cliente
- Vantagens:
 - Privilegia qualidade
 - Não vai ser enganado ("entregar por entregar")
 - Pode mudar de fornecedor

Dúvidas

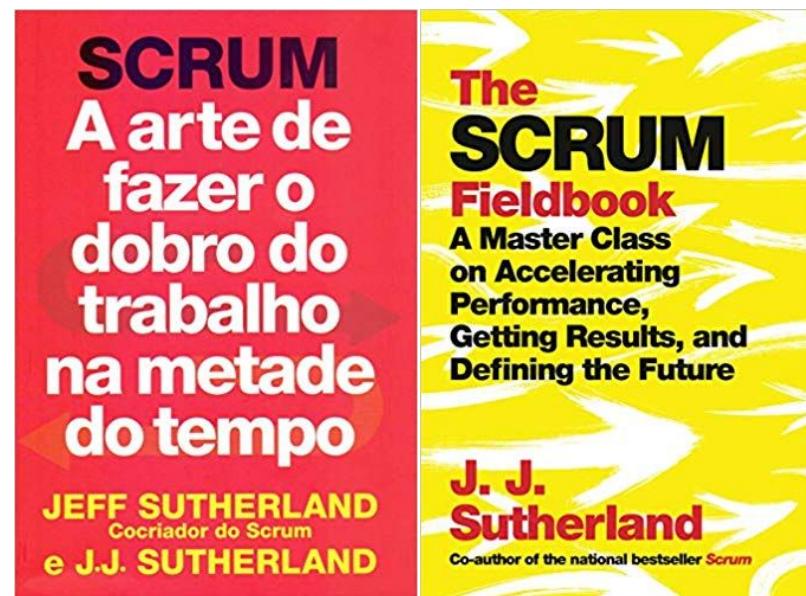


SCRUM: Origem

- Proposto como um processo de desenvolvimento...

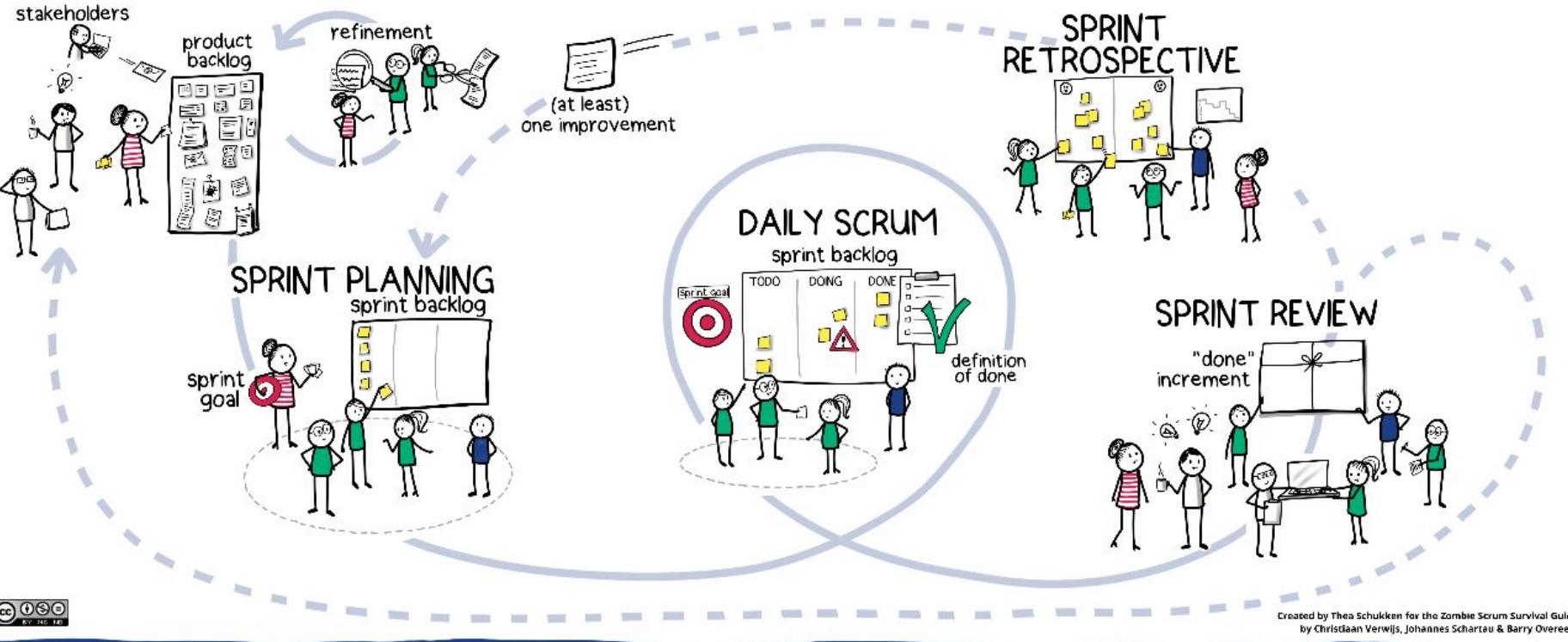


OOPSLA 1995



Livros atuais

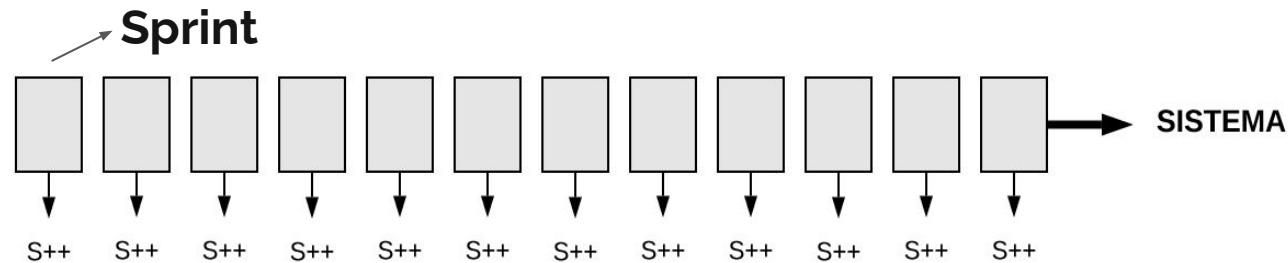
O que é SCRUM?



Fonte: <https://www.scrum.org/resources/scrum-framework-reduce-risk-and-deliver-value-sooner>

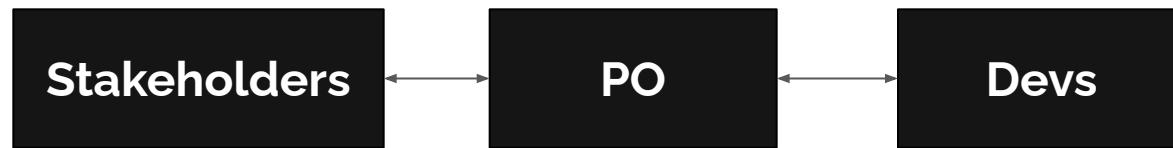
Evento Principal: Sprints

- O que é uma Sprint?
- O desenvolvimento é dividido por eventos com iterações, conhecidas como **Sprints**.
- Qual a duração de 1 sprint? depende?
 - 15 dias (normalmente); ou
 - até 1 mês.



Iterações Ágeis SCRUM

- Na sprint, o **Product Owner (PO)** explica as **histórias** para **Devs**.
- A documentação é expressa verbalmente e informal por meio de conversas entre o PO e os devs.



- **O que são as histórias de usuários?**
- **Quais as responsabilidades do PO e dos Devs?**

Histórias SCRUM

- As **histórias** são requisitos, normalmente, implementados como funcionalidades em sprints.
- As **histórias** são escritas pelo **PO** em um cartão “*post-it*”. Os **devs** devem implementar as histórias.
 - Exemplo:

Postar Pergunta

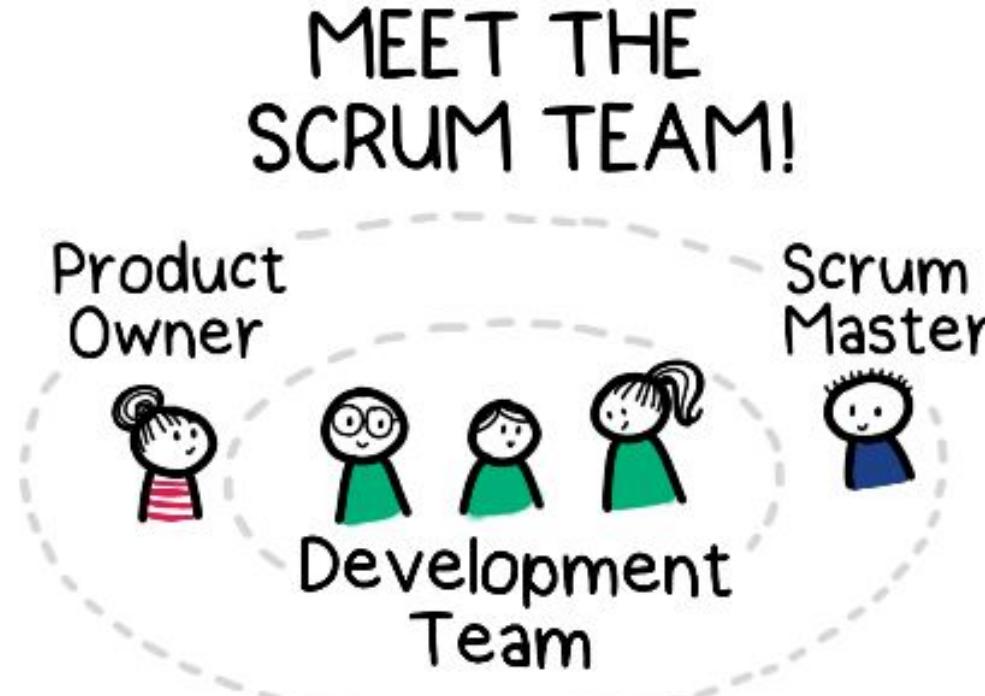
Um usuário, quando logado no sistema, deve ser capaz de postar perguntas. Como é um site sobre programação, as perguntas podem incluir blocos de código, os quais devem ser apresentados com um layout diferenciado.

Exemplo de História Detalhada

- História: Postar Perguntas
- Tarefas da História:
 - Projetar e testar a interface Web, incluindo layout, CSS templates.
 - 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.

Papéis SCRUM

- Time de 5 a 10 pessoas: 1 PO + 3 a 8 Devs + 1 Scrum Master



Fonte: <https://www.scrum.org/resources/scrum-framework-reduce-risk-and-deliver-value-sooner>



Created by Yael Schachar for the Scrum.org Scrum Framework
by Christian Vennin, Johnsen Goh, & Mary Czerwinski

Papéis SCRUM

Product Owner (PO)

- O **PO** deve escrever, explicar e priorizar histórias dos usuários entre os devs.
- Manter o **backlog do produto** (lista de histórias).
- O **PO** está próximo do problema.
- Exemplo de Sistema Bancário de Seguros:
 - Cliente: Departamento de Seguros do Banco
 - **#1: Dev. interno ⇒ PO: funcionário do banco**
 - **#2: Dev. interno ⇒ PO: analista do banco**

Papéis SCRUM

Development Team (Devs)

- São os **Desenvolvedores (Devs)** do produto.
- Os devs sendo responsáveis por entregar novos incrementos (iterações) do produto a cada sprint.
- O PO entrega as histórias para os devs implementarem
 - O PO não é o “chefe”. Todos tem o mesmo nível hierárquico.
- Os devs tem autonomia para comunicar o PO sobre o que é possível desenvolver em cada sprint.

Papéis SCRUM

Scrum Master

- Especialista em SCRUM que ajuda pelo menos um ou vários times em eventos (sprints).
- Atua como um professor ou coach/mentor do time.
- Não é o chefe, mas ajuda o time a resolver problemas por conta própria.

Dúvidas



Planejamento da Sprint



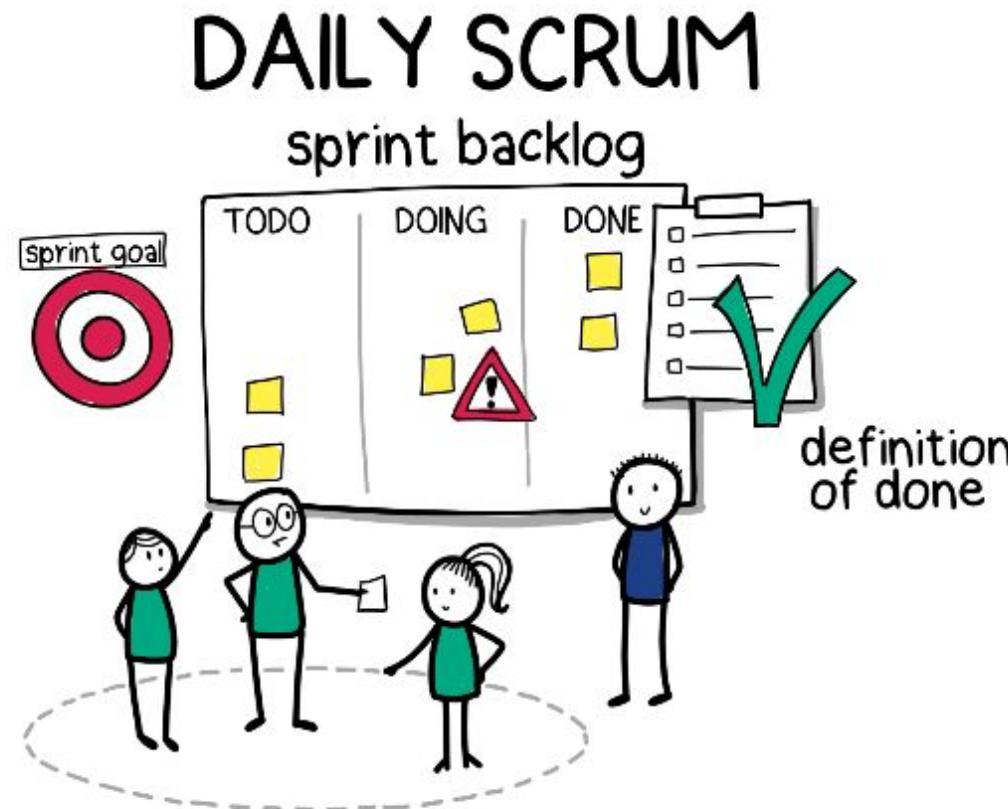
Created by Theis Schäffer for the Scrum Sprint Planning Game
by Christiane Valente, Christian Schäffer & Barry Ows夜

Fonte: <https://www.scrum.org/resources/scrum-framework-reduce-risk-and-deliver-value-sooner>

Planejamento da Sprint

- O PO propõe histórias para os devs implementarem.
- Os devs decidem se conseguem implementá-las em tempo hábil em uma ou várias sprints.
- A reunião é dividida em duas partes:
 - 1^a parte: Definir as histórias da sprint (**Backlog do Produto**).
 - 2^a parte: Dividir histórias em tarefas; e atribuir as tarefas aos devs (**Backlog da Sprint**).
- Os Backlogs (Produto e Sprint) são artefatos!!

Reuniões Diárias



Created by Thea Schüller for the Zombie Scrum Survival Guide
by Christian Seifert, Johannes Schäfer & Harry Künzli

Fonte: <https://www.scrum.org/resources/scrum-framework-reduce-risk-and-deliver-value-sooner>

Reuniões Diárias

- 15 minutos de duração com cada pessoa:
 - o que ele(a) fez ontem
 - o que pretende fazer hoje
 - dificuldades encontradas
- As reuniões diárias ajudam a:
 - Melhorar comunicação
 - Antecipar problemas

Revisão da Sprint

SPRINT REVIEW



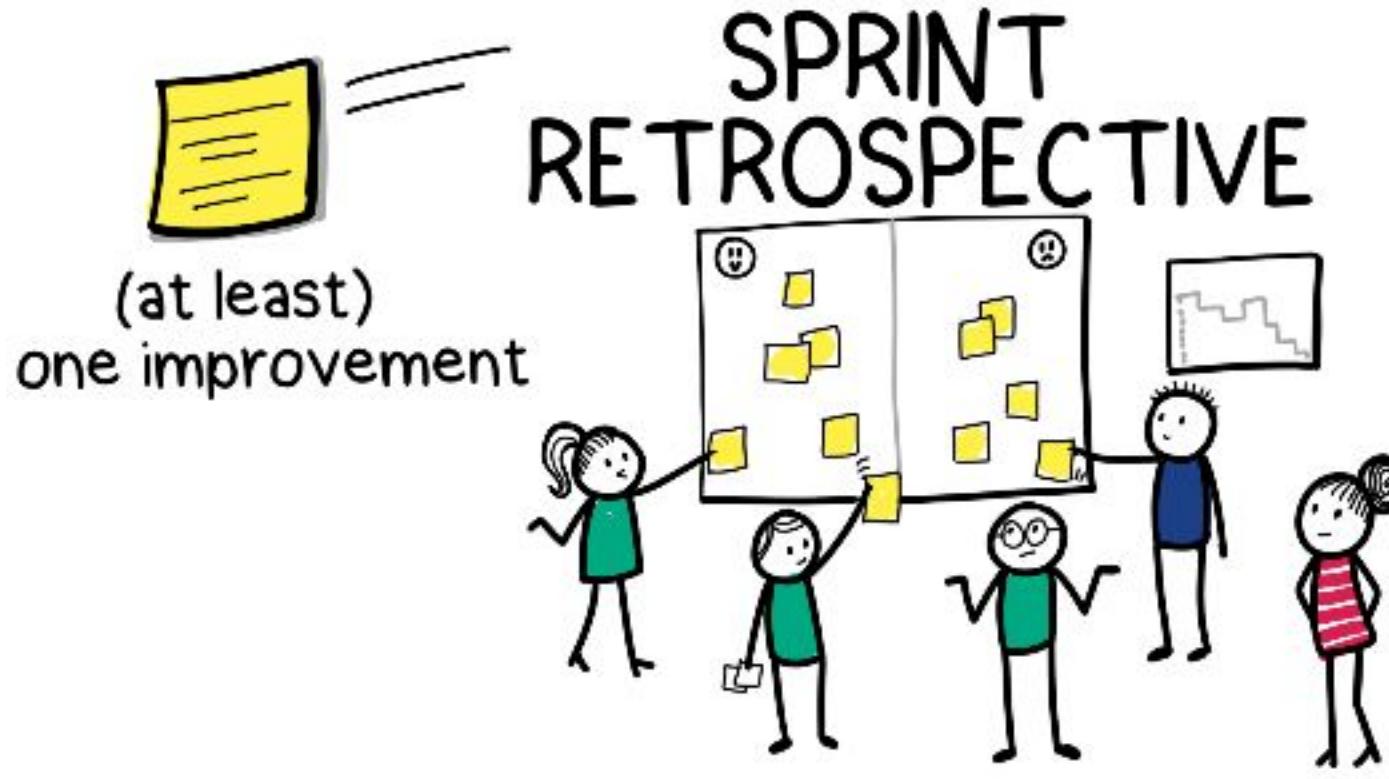
Created by Team Schellen for the Scrum Alliance Service Guide
by Christian Vennix, Julianne Schellen & Barry Overmars

Fonte: <https://www.scrum.org/resources/scrum-framework-reduce-risk-and-deliver-value-sooner>

Revisão da Sprint

- O time apresenta o resultado da sprint para PO e *stakeholders* (partes interessadas).
- A implementação das histórias pode ser:
 - **Aprovada; Aprovada Parcialmente;** ou **Reprovada.**
- Se a história for **Aprovada Parcialmente** ou **Reprovada** volta para o backlog do produto.
- **Done criteria** para concluir as histórias:
 - Qualidade externa: testes de aceitação e não-funcionais.
 - Qualidade interna: testes de unidade e revisão de código.

Retrospectiva da Sprint



Presented by The Scrum Alliance. Used with permission from the Scrum Alliance. © 2018 Scrum Alliance, Inc. All rights reserved. Scrum is a registered trademark of the Scrum Alliance, Inc.

Fonte: <https://www.scrum.org/resources/scrum-framework-reduce-risk-and-deliver-value-sooner>

Retrospectiva da Sprint

- Time se reúne para decidir o que melhorar
 - O que deu certo?
 - Em que momento precisamos melhorar?
- Discutir a melhoria contínua e constante do trabalho.
- **Importante:** *Não é para discutir fracassos ou gerar conflitos com o time (PO, Devs ou Scrum Master).*

Time-box SCRUM

stakeholders



refinement



(at least)



SPRINT RETROSPECTIVE

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

SPRINT PLAN



sprint goal

SPRINT REVIEW



"done"
increment

Created by Thea Schukken for the Zombie Scrum Survival Guide
by Christiaan Verwijs, Johannes Schartau & Barry Overeem

Fonte: <https://www.scrum.org/resources/scrum-framework-reduce-risk-and-deliver-value-sooner>

Story points

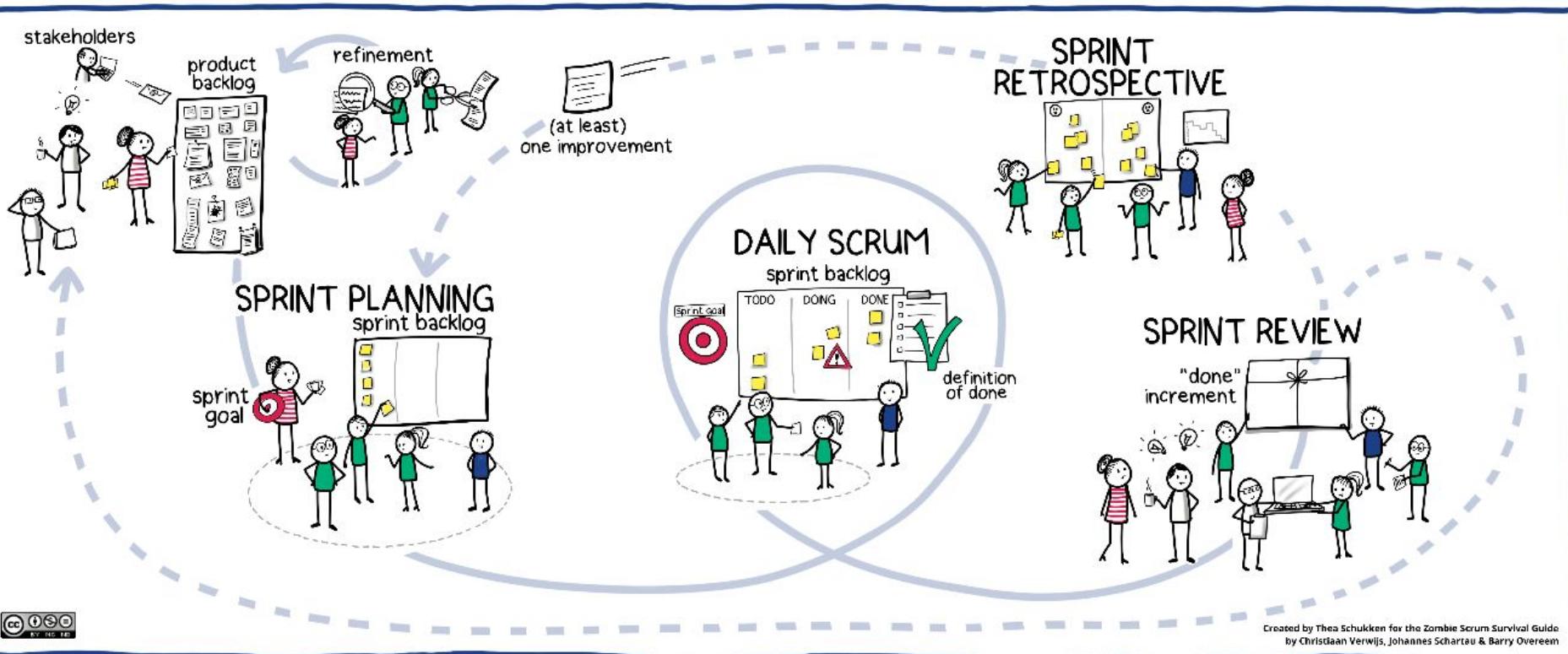
- Unidade (inteiro) para comparação do tamanho de histórias.
 - Exemplo de escala: 1, 2, 3, 5, 8, 13

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"

Resumindo SCRUM...



Fonte: <https://www.scrum.org/resources/scrum-framework-reduce-risk-and-deliver-value-sooner>

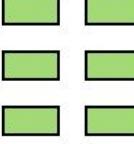
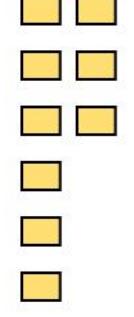
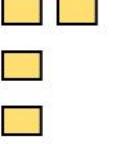
Dúvidas



Exemplo: SCRUM Board Físico



Exemplo: SCRUM Board Digital

Backlog	To Do	Doing	Testing	Done
				

SCRUM vs XP

- SCRUM não é apenas para projetos de software
 - Não define práticas de programação, como XP
- SCRUM define um "processo" mais rígido que XP
 - Eventos, papéis e artefatos bem claros

Atividade de Leitura

- Faça a leitura do documento. Disponível em:
 - <https://www.scrum.org/resources/scrum-framework-reduce-risk-and-deliver-value-sooner>



SCRUM: UM FRAMEWORK PARA REDUZIR RISCO E ENTREGAR VALOR MAIS CEDO

UMA VISÃO GERAL DO FRAMEWORK SCRUM, PARA PESSOAS NOVAS
AO SCRUM E AQUELES QUE GOSTARIAM DE ATUALIZAR OS SEUS
CONHECIMENTOS.

CHRISTIAAN VERWIJS, BARRY OVEREEM AND JOHANNES SCHARTAU (THE
LIBERATORS) | JUNHO 2020

Atividade Prática

- Em times de cinco alunos(as) realizem as tarefas:
 1. Acessar o site **trello.com** e fazer o cadastro.
 2. Criar um quadro digital (board).
 3. Definir os papéis de PO e Devs conforme estudamos.
 4. Compartilhar o quadro com o professor (Scrum Master) por email:
ricardogeraldi@professores.utfpr.edu.br
 5. Criar as listas (backlogs).
 6. Escrever pelo menos duas histórias (funcionalidades) no backlog do produto para um sistema de gerenciamento de carros elétricos.
 7. Definir tarefas em uma lista de backlog da sprint. Essas tarefas estarão na próxima Sprint.

Exemplo: Projeto Mozilla (Github projects)

The screenshot shows a GitHub Project board for the Mozilla project, specifically for the 'Smart Scheduling' board. The board is organized into four main columns: Backlog, Next up, In progress, and Done.

- Backlog:** Contains 100 items, updated 13 days ago. Items include:
 - Bug 1632870 - Store test configuration in the task definition
 - Bug 1667401 - Always schedule "new manifests" with manifest based bugbug optimizers
 - Investigate Treeherder UI/UX changes for test filtering
 - Handle pushes containing multiple backouts
 - When a task/group has inconsistent classifications, use the status of the task/group on the backout to figure it out
 - Bug 1635921 - Use |mach try fuzzy| as a means to select configurations with
- Next up:** Contains 1 item: Reinstate integration tests (mozci#390 opened by marco-c).
- In progress:** Contains 3 items:
 - 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, assigned: ahal, metrics)
 - Bug 1639164 - "mach try auto" should select the best platforms to run manifests on (assigned: marco, Added by marco-c)
- Done:** Contains 222 items, updated 13 days ago. Items include:
 - Make adr dependency optional (mozci#388 opened by marco-c)
 - Bug 1671422 - Add durations to group_result actions in errorsummary formatter (assigned: ahal, Added by ahal)
 - Cache results from data sources (mozci#368 opened by marco-c, enhancement)
 - OOMs while analyzing some pushes (mozci#366 opened by marco-c, bug)
 - Add support for getting groups and groups results in the Treeherder data source (mozci#312 opened by marco-c)

At the top right of the board, there is a search bar labeled 'Filter cards'.

Considerações: SCRUM

- Métodos ágeis não são uma “bala de prata”.
- SCRUM tem um processo “leve”, que exige disciplina e adaptação durante o desenvolvimento de software ágil.
- As pessoas (papéis) devem ter autonomia e estar sincronizadas como um time (não equipe).
- O SCRUM não simplifica todos os problemas durante o desenvolvimento de software.
- Não é aplicável apenas em projetos de software.

Dúvidas





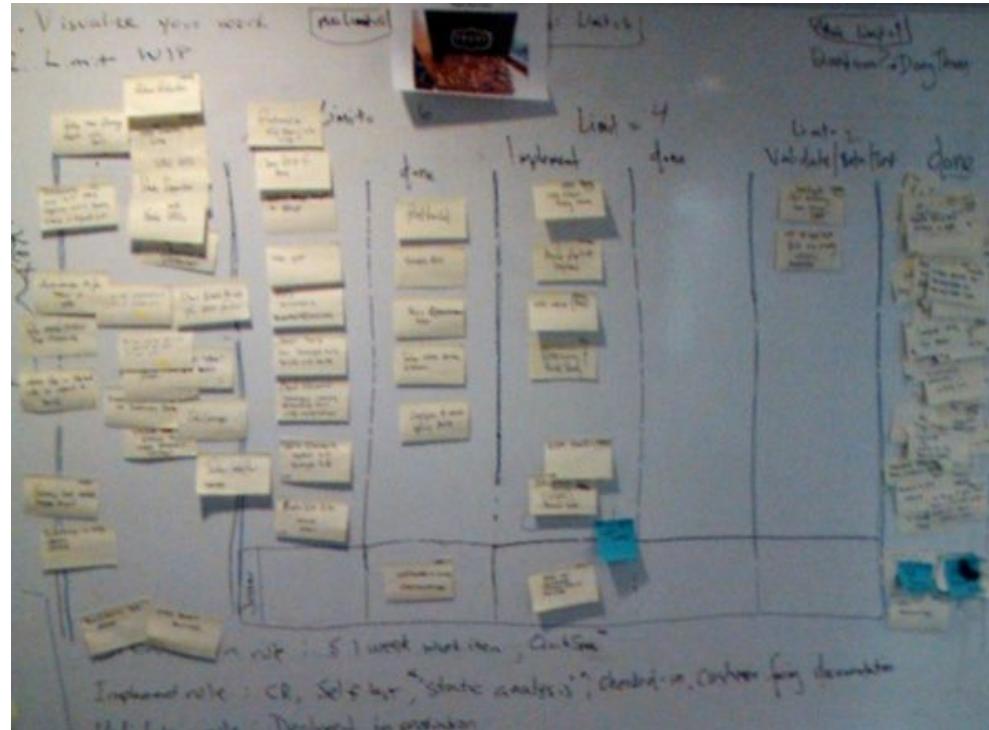
Kanban

- Origem na década de 50 no Japão
- Sistema de Produção da Toyota
- Manufatura Lean
- Produção *just-in time*

Kanban = "cartão visual"



Kanban em Desenvolvimento de Software



Exemplo Kanban Board -> trello.com

Template Kanban - Software

Personal Boards Free Visível à Área de trabalho Convidar

Quadro

A Fazer

Desenvolver Contas a Pagar

Fazer upload dos módulos do sistema no Github

2 de ago

+ Adicionar outro cartão

Em andamento

Desenvolver Cadastro de Produtos

+ Adicionar outro cartão

Concluídas

Desenvolver Contas a Receber

7 de jun

+ Adicionar outro cartão

+ Adicionar outra lista



Kanban

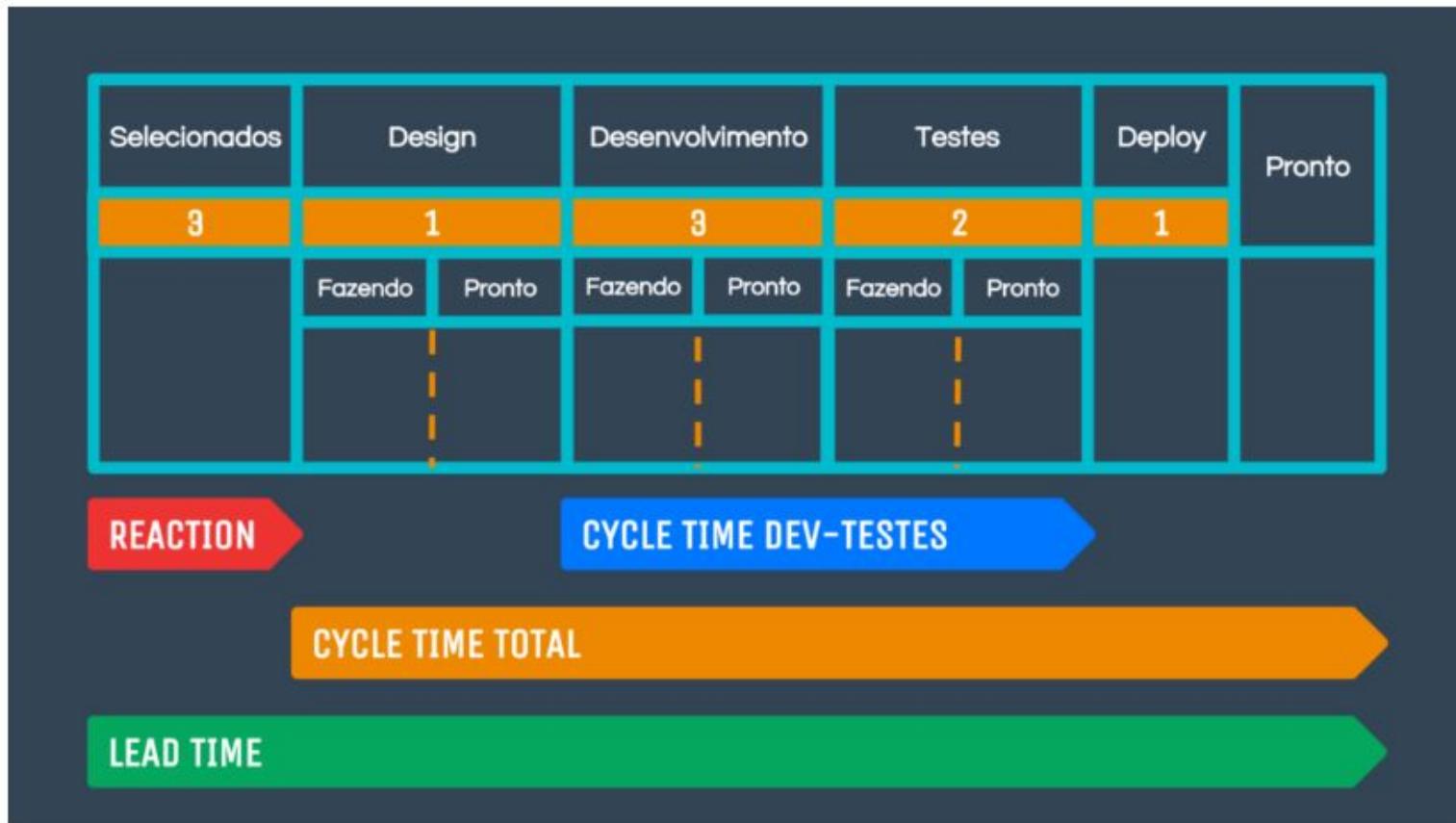
- Ideia central: sistema *pull*
- Membros "puxam" trabalho:
 - a. Escolhem uma tarefa para trabalhar
 - b. Concluem tarefa (movem ela para frente no quadro)
 - c. Voltam para o passo (a)

Exemplo Kanban em Empresa



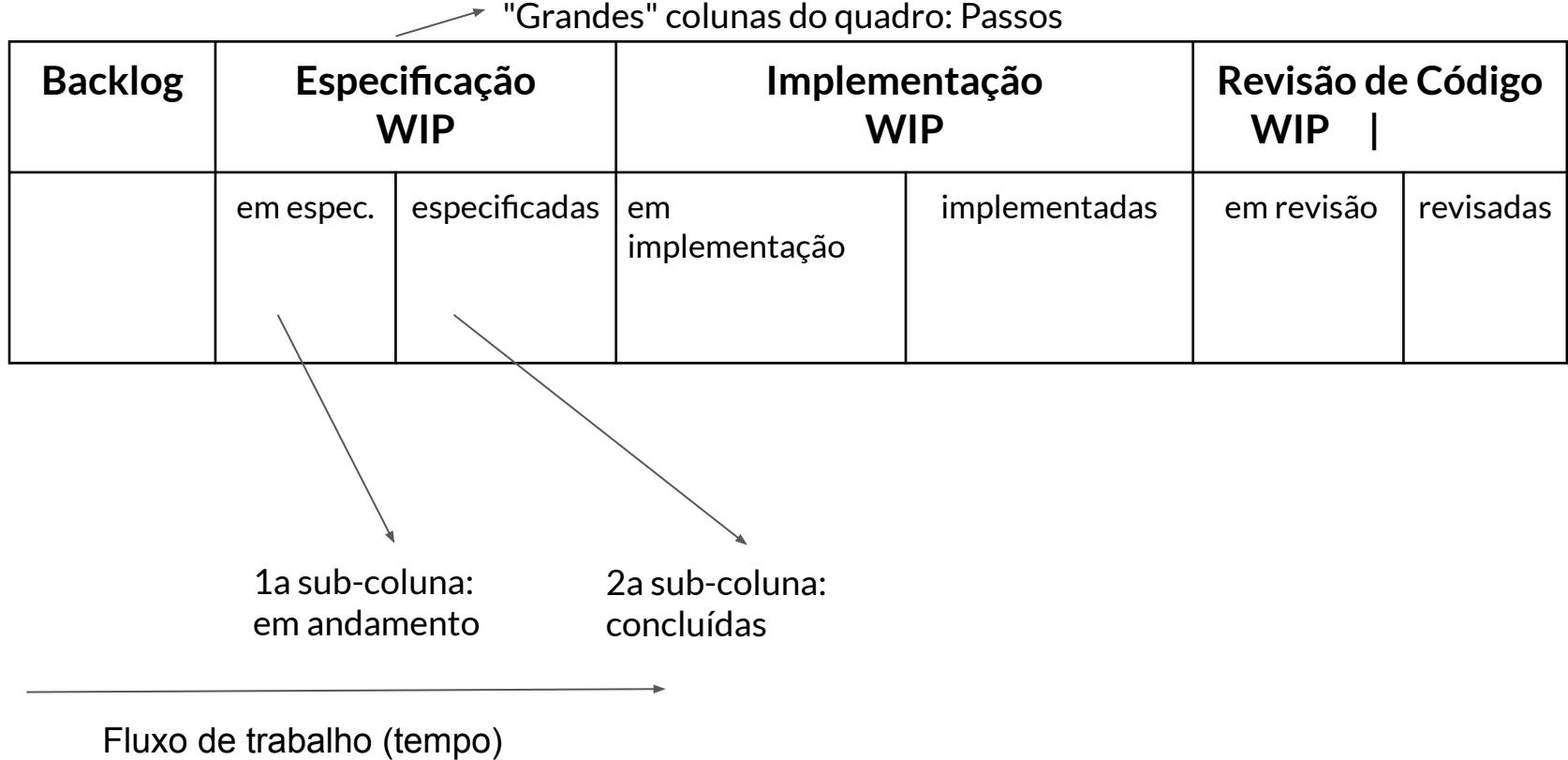
<https://targetteal.com/pt/blog/metodo-kanban/>

Exemplo Kanban em Empresa



<https://targetteal.com/pt/blog/metodo-kanban/>

Kanban: Work in Progress (WIP)





Objetivos dos Limites WIP

- WIP: Número de tarefas do time
- 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

Comentários Finais sobre Kanban

- Kanban é mais simples do que Scrum? Depende.
- Kanban é mais adequado para times mais maduros?
- Talvez: começar com Kanban e depois migrar para Scrum?

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

Qual é o melhor processo de desenvolvimento de software?

EXPLAINING SOFTWARE DEVELOPMENT METHODS BY FLYING TO MARS



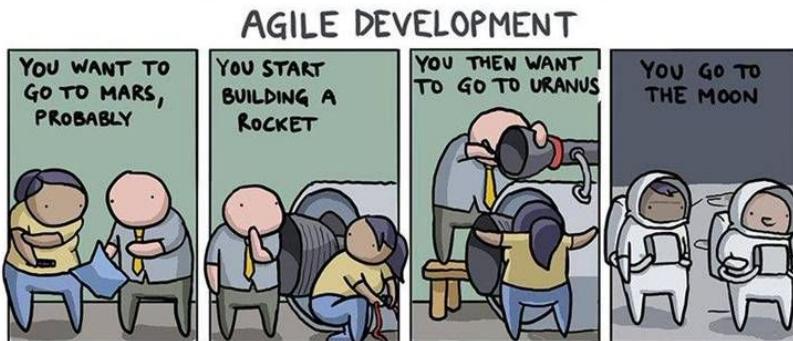
THE WATERFALL METHOD



THE KANBAN METHOD



SCRUM



AGILE DEVELOPMENT



LEAN DEVELOPMENT

Quando NÃO utilizar métodos ágeis: SCRUM ou Kanban

- Condições do mercado são previsíveis
- Requisitos são estáveis
- Clientes não estão disponíveis para colaboração
- Problemas podem ser resolvidos em silos funcionais
- Mudanças no final do projeto são caras ou impossíveis
- Impacto de mudanças provisórias pode ser catastrófico

Dúvidas





Considerações Finais

- Processos de software são atividades para desenvolver um software com qualidade
- Modelos de processo de software são representações abstratas de processos
- Atividades básicas em processo de software
 - **Especificação** - o que o sistema faz
 - **Projeto e Implementação** - organização e desenvolvimento
 - **Validação** - o que o cliente quer
 - **Evolução** - mudanças e responde as necessidades do cliente



Considerações Finais

- Processos atuais devem ser iterativos e flexíveis
 - Métodos ágeis (XP, Scrum, Kanban, Lean...)
- Mudanças devem ser incorporadas nos processos de software mais modernos (ágeis)
- Metodologias ágeis são úteis para a melhoria contínua de processos e produtos
- A maturidade de processos deve ser considerada uma prática essencial para a evolução do software

Considerações Finais

- Lembre-se do Manifesto Ágil...



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.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler

James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick

Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

Atividade de Leitura

- Ler Capítulo 2 (VALENTE, 2020)
 - <https://engsoftmoderna.info/cap2.html>



- Ler Capítulos 2 e 3 (SOMMERVILLE, 2015)
 - Existe uma edição traduzida do livro na Biblioteca



Atividade Prática

- Pesquisar sobre os modelos Lean e Spotify.
- A turma deve apresentar em aula a definição e curiosidades sobre os modelos.
 - **Times de até cinco pessoas!**
 - Se organizem entre si!
 - **Definam os papéis e tarefas de cada um no time.** Faz parte...
- **Quais as principais diferenças e aplicações dos modelos em comparação com Scrum e Kanban?**

Próxima Aula

- Introdução à Engenharia de Requisitos



Bibliografias Principais

Conteúdo em Português

1. VALENTE, M. T. Engenharia de Software Moderna (2020-2023):

<https://engsoftmoderna.info/> (Livro texto aberto online - Materiais em Português)

2. WAZLAWICK, R. S. Análise e Design Orientados a Objetos para Sistemas de Informação (2014-2023). Página pessoal:

<https://www.inf.ufsc.br/~raul.wazlawick/>

3. REINEHR, S. Engenharia de Requisitos (2020). Disponível em:

[https://integrada\[minhabiblioteca\].com.br/reader/books/9786556900674/](https://integrada[minhabiblioteca].com.br/reader/books/9786556900674/)



Bibliografias Principais

Conteúdo em Inglês e Alemão

1. SOMMERVILLE, I. Software Engineering (Inglês - English) (2015-2023) (existe uma edição traduzida do livro no Brasil e na Minha Biblioteca da UTFPR):
<https://software-engineering-book.com/> AND
<https://iansommerville.com/engineering-software-products/> (Materiais em Inglês)
2. (GAMMA et al., 2000) GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. Padrões de Projetos: Soluções Reutilizáveis de Software Orientados a Objetos. Bookman, 2000.
3. (FREEMAN et al., 2004) FREEMAN, E.; ROBSON, E.; BATES, B.; SIERRA, K. Head First Design Patterns. O'Reilly, 2004 (tem versão atualizada em 2020).
4. THÜM, T. Softwaretechnik (2021-2022):
<https://github.com/SoftVarE-Group/Software-Engineering-2021-2022> (Materiais em Inglês). Vídeos em Alemão (Deutsch)



Bibliografias Principais

GUIA ESSENCIAL PARA TODO ENGENHEIRO DE SOFTWARE:

SWEBOK v3 (2014). IEEE Computer Society Software Engineering Body of Knowledge Guide V3.
Disponível em: <https://ieeecs-media.computer.org/media/education/swebok/swebok-v3.pdf>

SWEBOK v4 (Beta) (2022 em atualização). Disponível em:
<https://waseda.app.box.com/s/elnhhnezdycn2q2zp4fe0f2t1fvse5rn>