BandTec
DIGITAL SCHOOL

BandTec
DIGITAL SCHOOL

Engenharia de Software

Processos de Software

Professor Esp. Gerson Santos

Objetivo da Aula

- Processos de Software
- Ferramentas CASE

Bibliografia

Bibliografia:

Engenharia de Software 8º Edição / Ian Sommerville

Engenharia de Software 6º Edição / Roger Pressman

Adicional

- Code Complete
- SWEBOK
- entre outros

Engenharia de Software– Nosso caminho

introdução

- ✓ Conceitos de UX + UI
- ✓ Fatores Humanos
- ✓ Design de Interação
- ✓ Design de Interfaces + BootCamp
- ✓ Jornada do Usuário
- ✓ Prototipação das Telas

06/09

- UI/UX para WEB
- Projeto de Software
- Interface WEB com regras de usabilidade
- Diagrama de Solução de Software
- Planilha de Arquitetura

18/10

- Qualidade e Testes
- Processo de Software
- Aula Especial

29/11

Final de Semestre

- Apresentação PI
- Avaliação Integrada

LEGENDA

- Conteúdo
- Entregável PI
- ✓ Conteúdo Finalizado
- ✓ Entregável Finalizado

Onde Estamos

Semana final das Sprints

Semana das Entregas de PI

O que é Engenharia de Software?

- ISO/IEC/IEEE Systems and Software Engineering

  "A aplicação sistemática, disciplinado e quantificáveis abordagens para o desenvolvimento, operação e manutenção de software"

- É uma disciplina de engenharia relacionada a todos os aspectos de produção de software (Sommerville)

- Engenharia de software engloba processos, métodos e ferramentas que possibilitam a construção de sistemas complexos baseados em computador dentro do prazo e com qualidade. (Pressman)

- en·ge·nha·ri·a (engenho + -aria) substantivo feminino

  1. Conjunto de técnicas e métodos para aplicar o conhecimento técnico e científico na planificação, criação e manutenção de estruturas, máquinas e sistemas para benefício do ser humano.

  2. Ciência ou arte da construção (ex.: engenharia mecânica, engenharia militar, engenharia naval)

Nosso Objetivo

Aprender/Ensinar processos, métodos e ferramentas para construção e manutenção de softwares profissionais.

PALAVRA CHAVE

PRAGMÁTICO

Mais definições que vamos repetir muito....

prag·má·ti·co

adjetivo

1. Relativo à pragmática ou ao pragmatismo.

2. Que tem motivações relacionadas com a .ação ou com a eficiência. = PRÁTICO

adjetivo e substantivo masculino

3. Que ou quem revela um sentido prático e sabe ou quer agir com eficácia.

prag·ma·tis·mo

(inglês pragmatism)

substantivo masculino

Doutrina que toma por critério da verdade o valor prático e se opõe ao intelectualismo.

MANTRA

PRAGMATISMO

Princípios Básicos do Desenv. de Software

(David Hooker)

- RAZÃO DE EXISTIR
- KISS (KEEP IT SIMPLE, STUPID). Faça de forma simples, tapado
- MANTENHA A VISÃO
- ESTEJA ABERTO PARA O FUTURO
- PLANEJE COM ANTECEDÊNCIA, VISANDO A REUTILIZAÇÃO
- PENSE!
- O QUE UM PRODUZ, OUTROS CONSOMEM

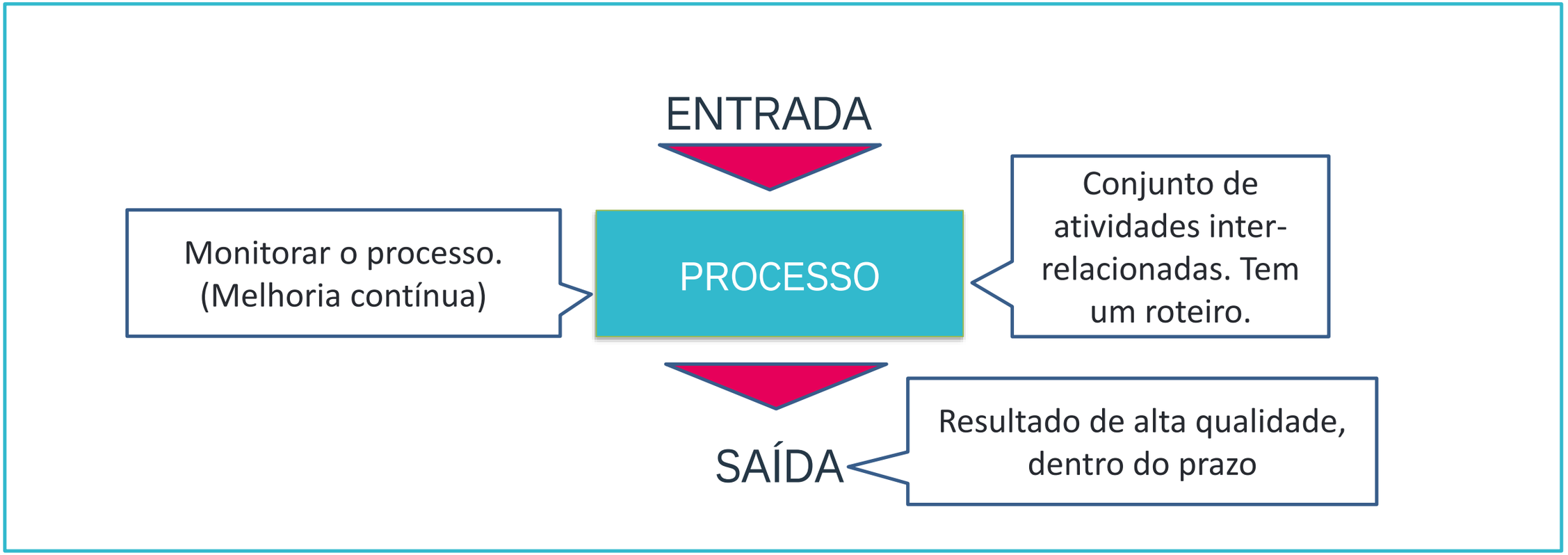
Mitos

- Se decorar os padrões e procedimentos vou ter sucesso!
- Se o cronograma atrasar, põe mais gente!
- Se terceirizar, relaxa que vão entregar
- Defina o objetivo e comece a escrever o código, os detalhes vem depois
- Uma vez que o programa está em uso, o trabalho acabou
- Enquanto não colocar para funcionar, não dá para avaliar a qualidade
- Engenharia de software é para atrapalhar a gente escrever código.

O que é Processo de Software?

- É um conjunto de atividades que leva à produção de um produto de software (Sommerville)

- ...um roteiro que ajude a criar um resultado de alta qualidade e dentro do prazo estabelecido. (Pressman)



Objetivo do Processo de Software

QUALIDADE

Mais importante na Eng. de Software

PESSOAS

Evolução das Operações

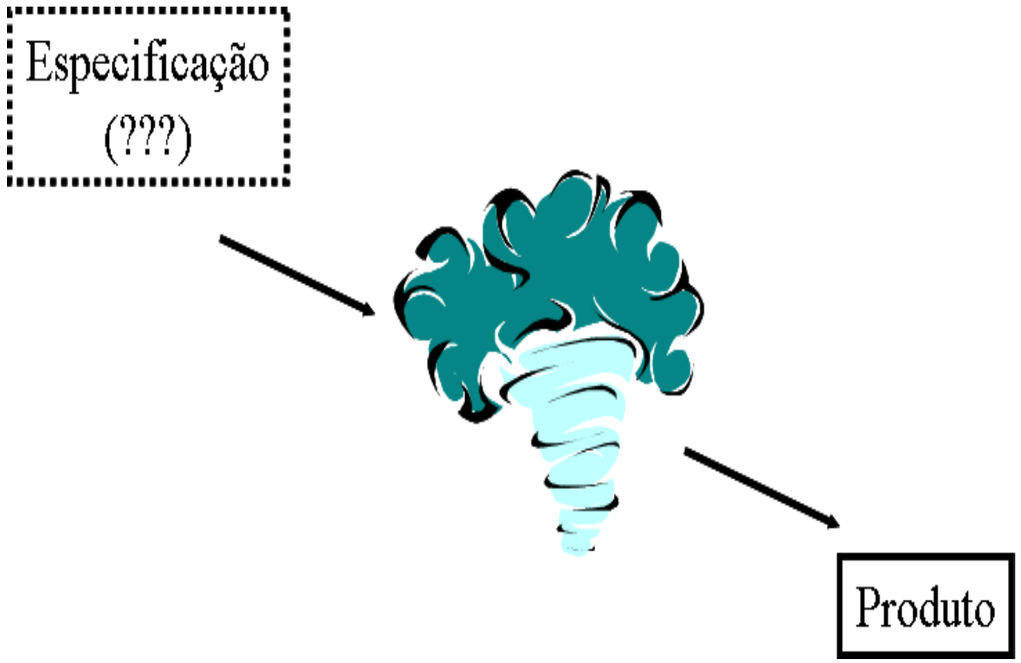
Operações				
Artesanal	Artesanal	Fábrica	Fábrica + Outsourcing Integrado	Linha de produção de Software

Processos				
	Processos Proprietários	CMM	PMI, RUP, ISOs	XP, ASD (Adaptative Software Developmen)t
Plataformas				
Fortran, Assembler	Cobol, PL1 (IBM)	Natural, C, C++, Clipper	VB, Delphi	Java, .NET

Modelos de Processos				
Waterfall		Evolucionários	Especializados: Componentes, OO, UML	?Agile
1960-1970	**1970-1980**	**1980-1990**	**1990-2000**	**Século XXI**

Fernandes, Aguinaldo Aragon. Fábrica de Software. 1.ed. São Paulo: Atlas; 2007. (Cap. 1)

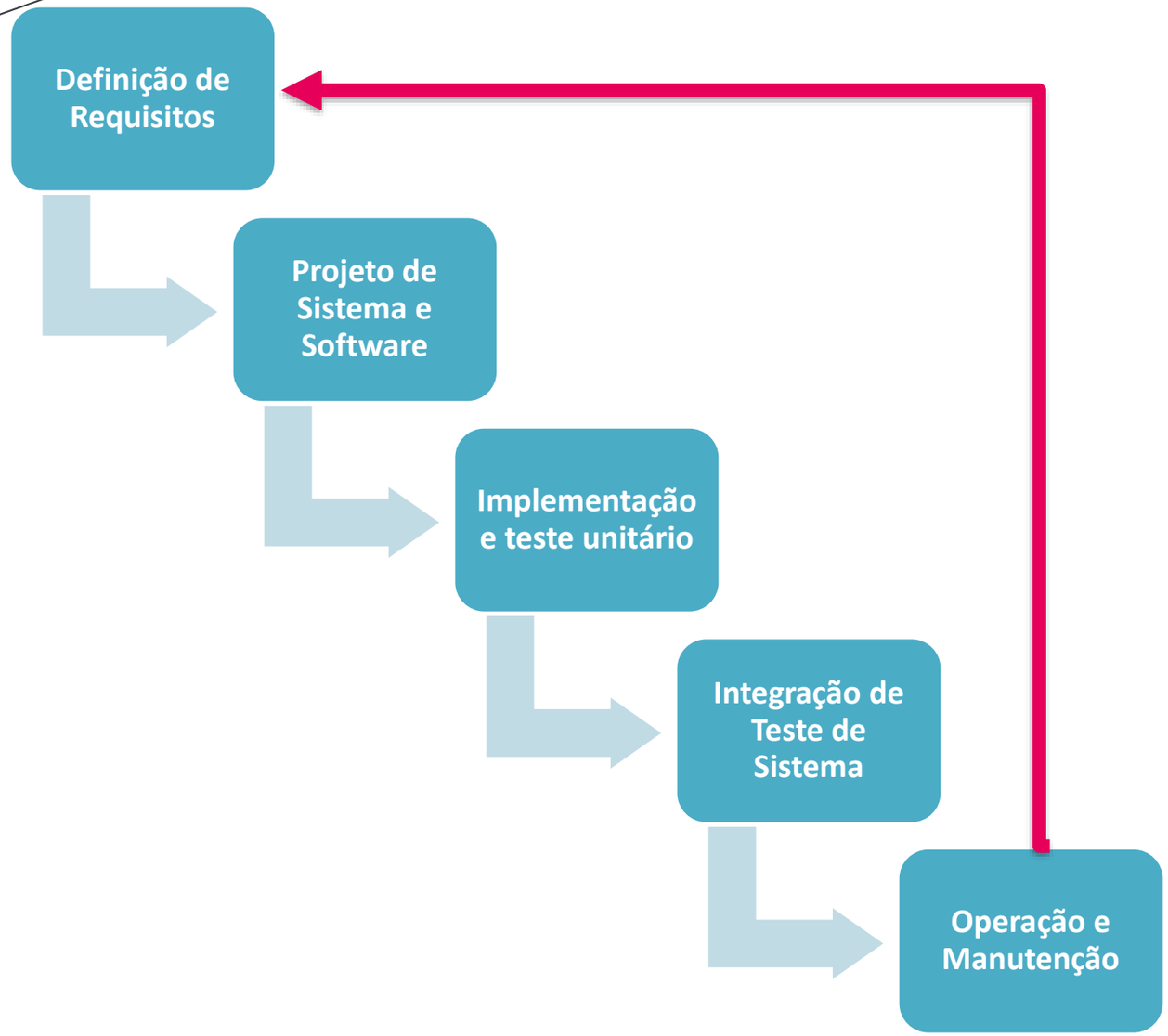
Modelo – Codifica-remenda



(Wilson de Pádua. 4ª edição)

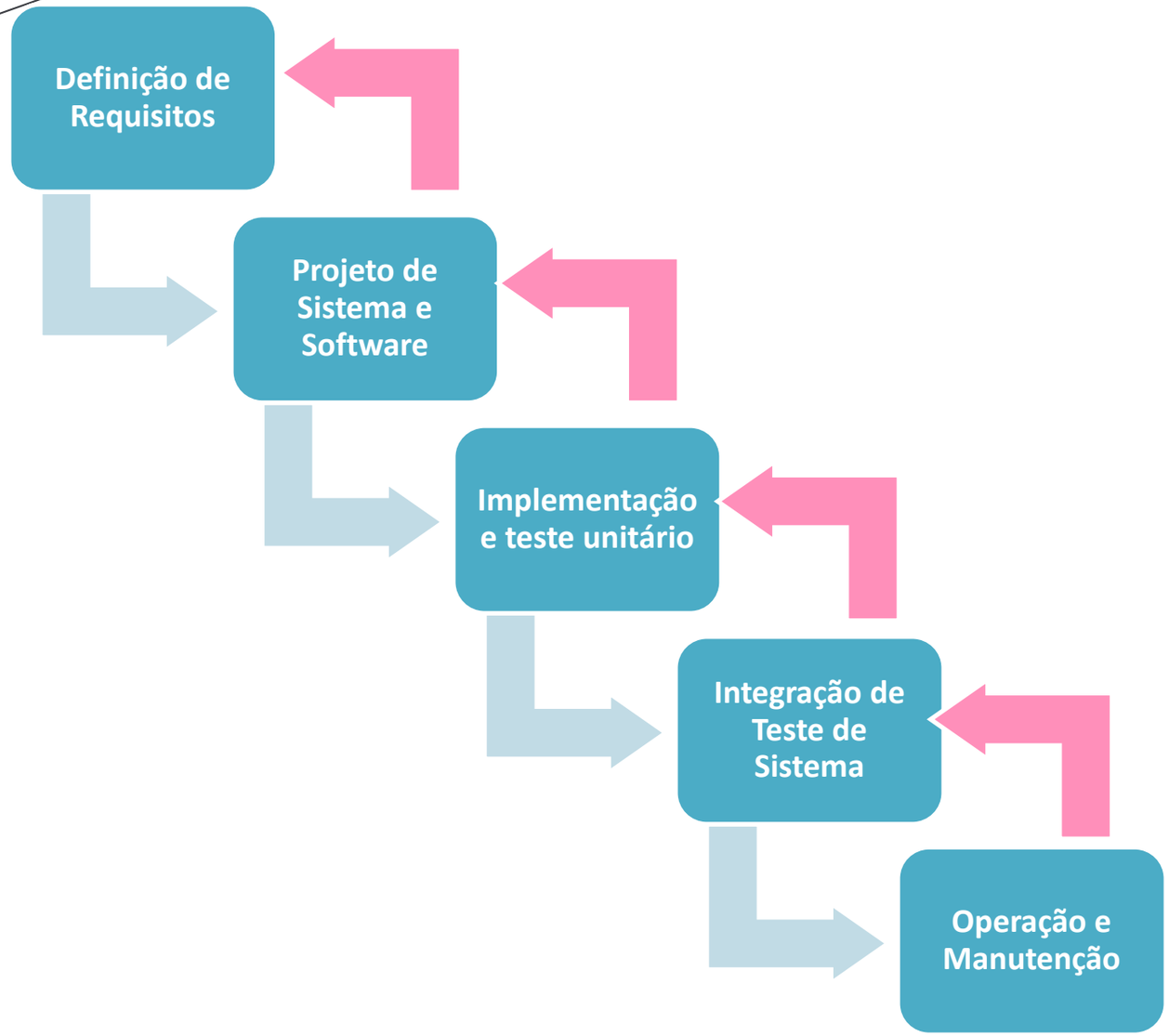
- Não é modelo espiral (apesar do redemoinho);
- Mais utilizado no mundo;
- Impossível de fazer gestão;
- Não permite assumir compromissos;
- A qualidade é baseada no acaso;
- Existem diversas variações. “Dá seu jeito”, “Pizza de baixo da porta”.

Modelo em Cascata



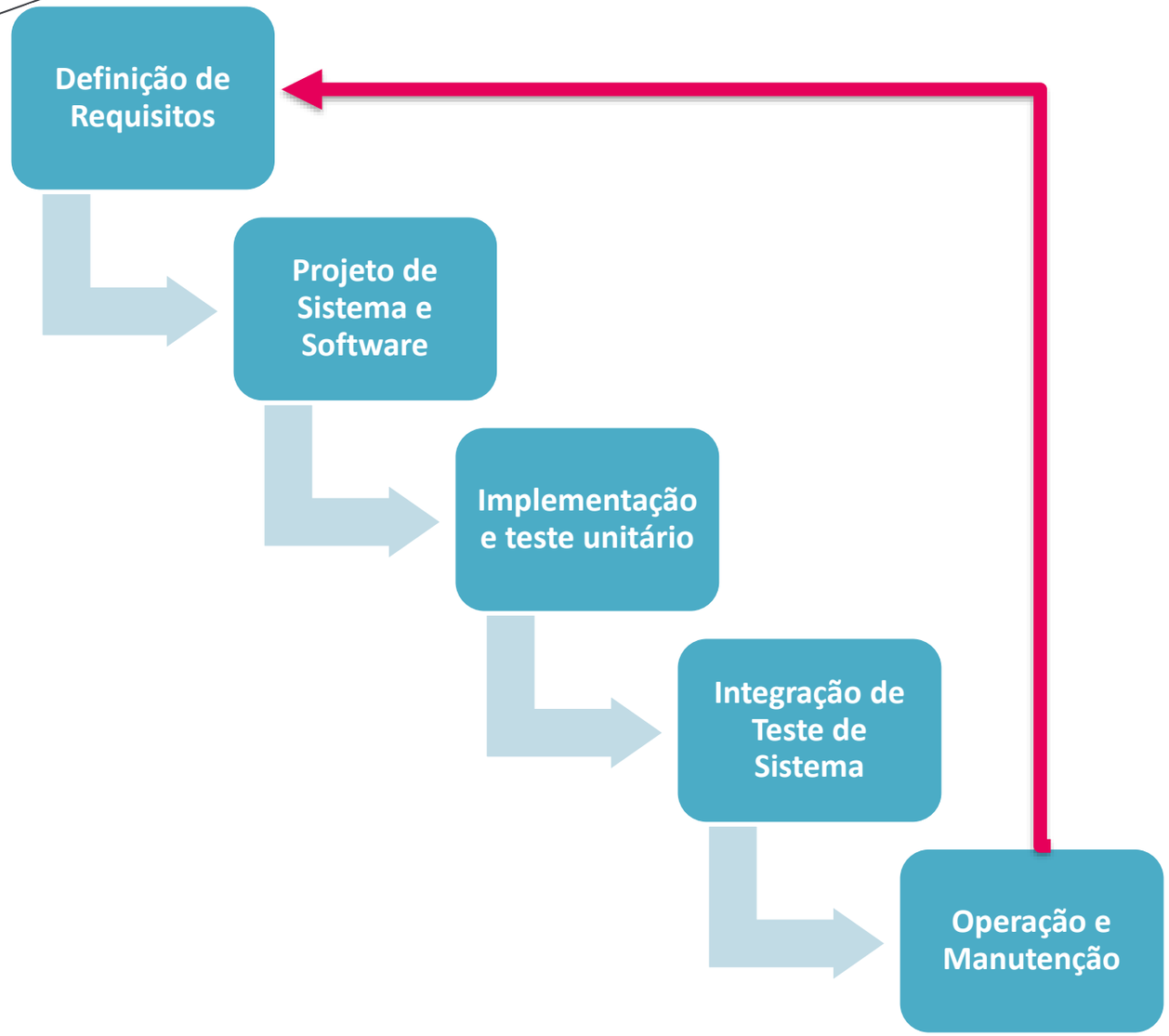
- Segue os processos gerais de Engenharia; É o mais antigo.
- É sequencial; costuma levar mais tempo.
- O resultado de cada fase consiste em uma validação formal de qualidade;
- Falhas e melhorias são realizadas posteriormente após todo o processo terminar;
- Gera muita documentação;
- É previsível; normalmente os requisitos estão bem definidos e mensuráveis;
- Utilizado quando há muito risco envolvido;

Modelo em Cascata com Realimentação



- Cada etapa é realizada e pode revisar uma etapa anterior;
- Mais difícil de gerenciar.

Modelo em Cascata - Desvantagens

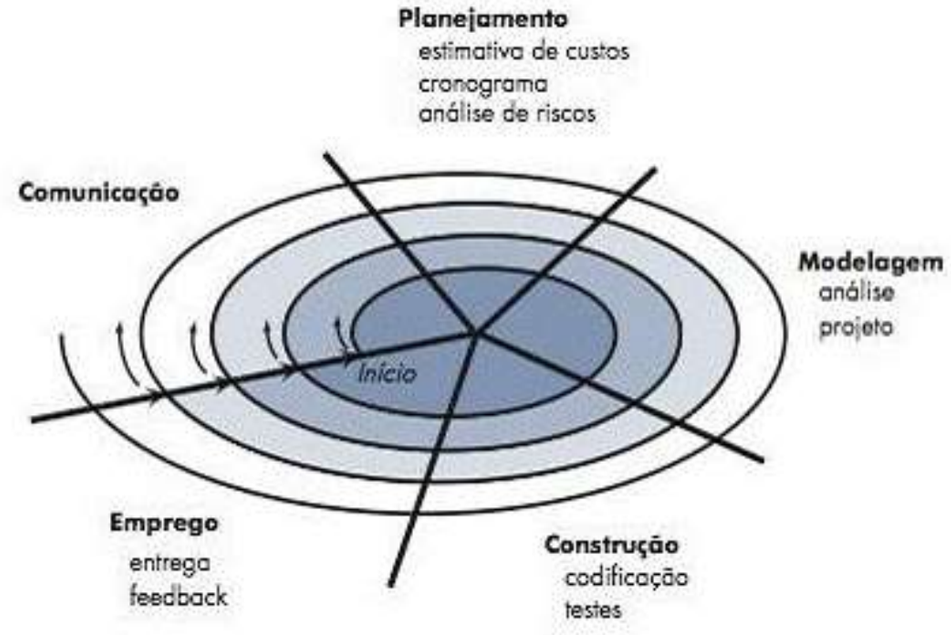


- Você precisa terminar um estágio para começar o próximo. Projetos reais raramente respeitam essa regra;
- Dificuldade de conseguir todos os requisitos de uma única vez;
- Falta de paciência do cliente em ver o resultado só no final de todo o ciclo;
- As entregas são normalmente enormes;
- Problemas de desenho, requisitos são detectados muito tarde;
- Os requisitos, após o término da fase de levantamento, são congelados.

Modelos de processo Evolucionários

Satisfação do Cliente!

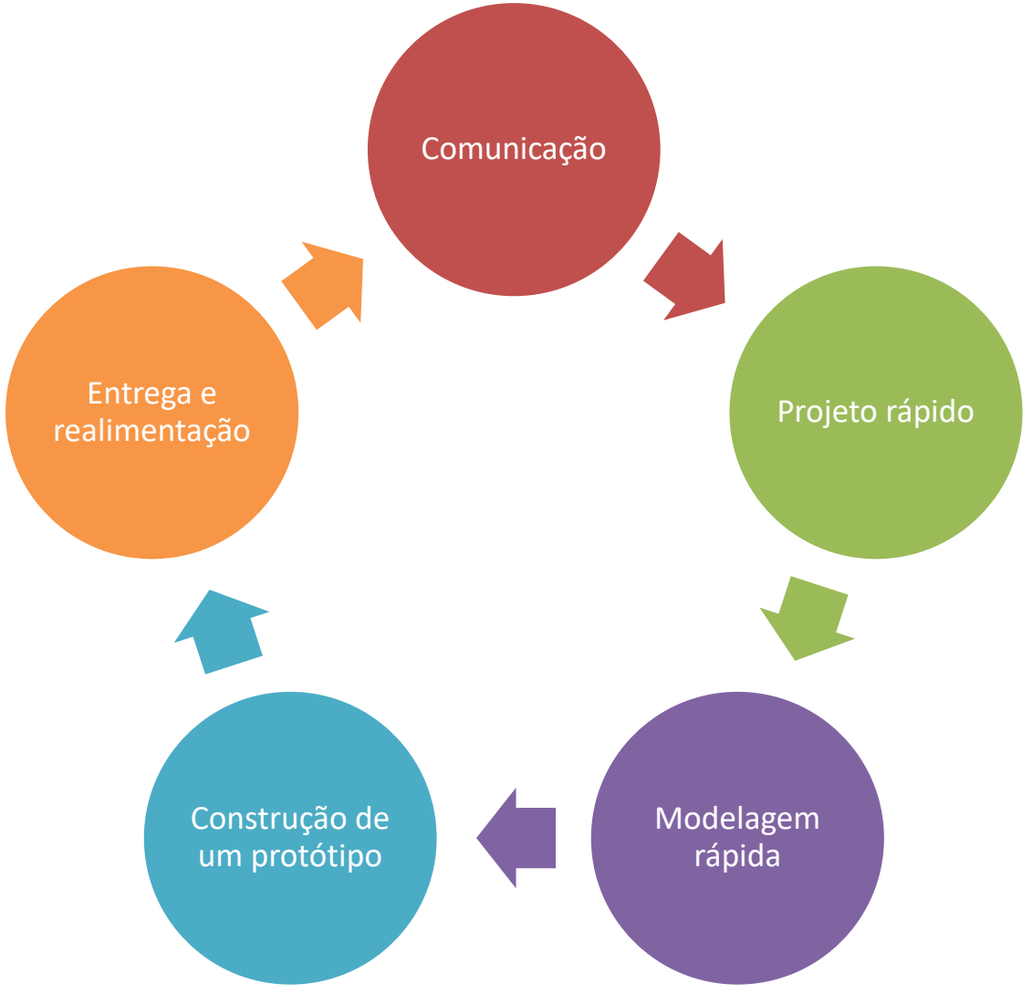
Modelo de Processo Evolucionário – Espiral



sommerville

- O software é desenvolvido em uma série de versões evolucionárias;
- Começa como um protótipo, mas à medida que o tempo passa são produzidas versões cada vez mais completas;
- Os custos e cronograma são ajustados de acordo com a evolução e feedback;
- O modelo pode continuar mesmo depois do software entregue;
- Os riscos são considerados a cada fase de evolução do projeto.
- Custo é alto;
- Difícil convencer o cliente que está tudo sob controle;

Processo Evolucionário – Prototipação



- Utilizado quando há objetivos gerais mas o cliente não identifica claramente os requisitos;
- Quando há insegurança do desenvolvedor;
- Como é protótipo normalmente precisa ser reconstruído para uma versão com qualidade;
- Requisitos de Engenharia de Software são ignorados ou não analisados. (Ex: Linguagem, versão de SO).
- Modelo MVP é baseado neste processo.

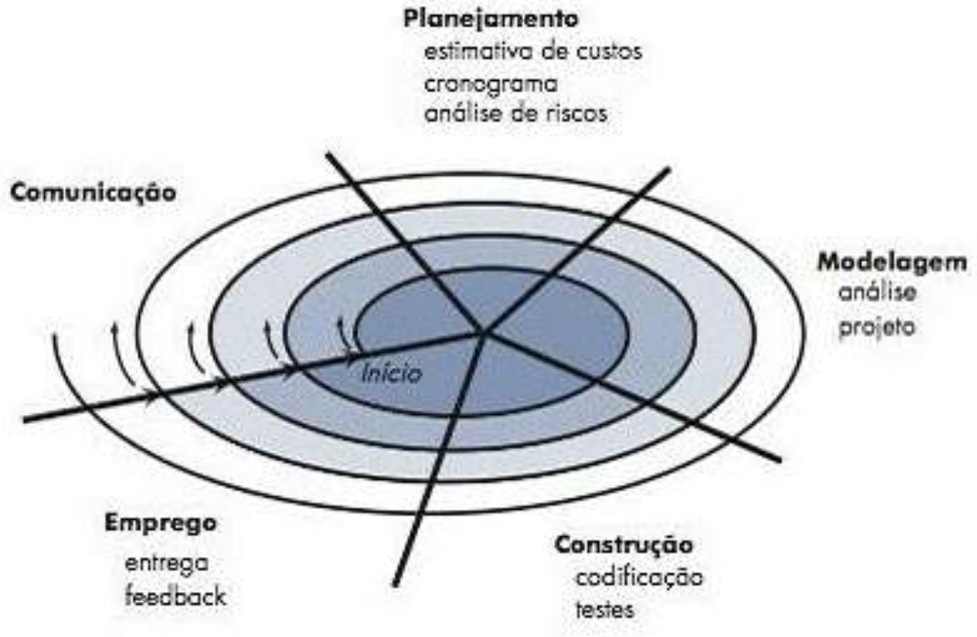
Modelo de Processo Incremental

- Combina o uso do modelo cascata aplicado a forma interativa;
- Tem fluxos paralelos, o desenvolvimento ocorre com entregas sucessivas; podem existir equipes diferentes em cada incremento;
- Cada sequência gera uma entrega; O cliente devolve feedback a cada entrega;
- Os requisitos mais importantes são priorizados nas primeiras entregas; Os incrementos iniciais podem ser usados como protótipos;
- Métodos ágeis como o XP são baseados neste processo;
- São fases de cada etapa do modelo incremental. ESPECIFICAÇÃO, DESENVOLVIMENTO e VALIDAÇÃO.

Processo Incremental - Desvantagens

- Precisa de um bom planejamento e desenho para que os incrementos possam ser “combinados”; Você precisa desenhar o sistema todo antes de “quebrar em partes”;
- O custo é mais alto que o processo cascata;
- Os requisitos comuns são mais difíceis de serem identificados;

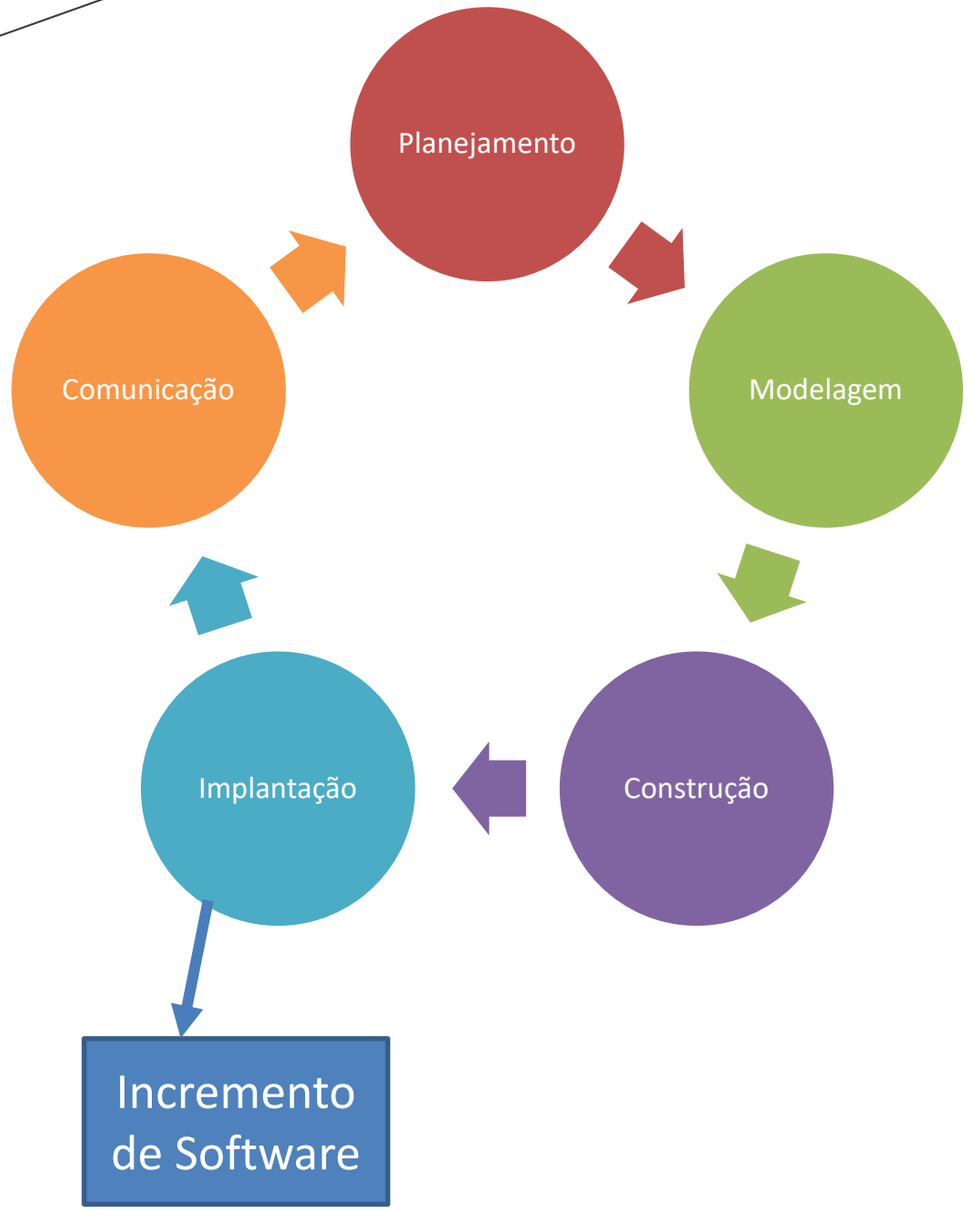
Processo Especializado- Baseado em componentes



- Baseado no modelo espiral;
- Envolve o desenvolvimento da aplicação a partir de componentes pré-existentes que são integrados a arquitetura;
- Preocupação com a integração é fundamental;
- Testes integrados (completos) precisam ser realizados para garantir que o todo funcione corretamente;
- O reuso é palavra chefe deste modelo;
- Normalmente utiliza tecnologias orientadas a objeto.

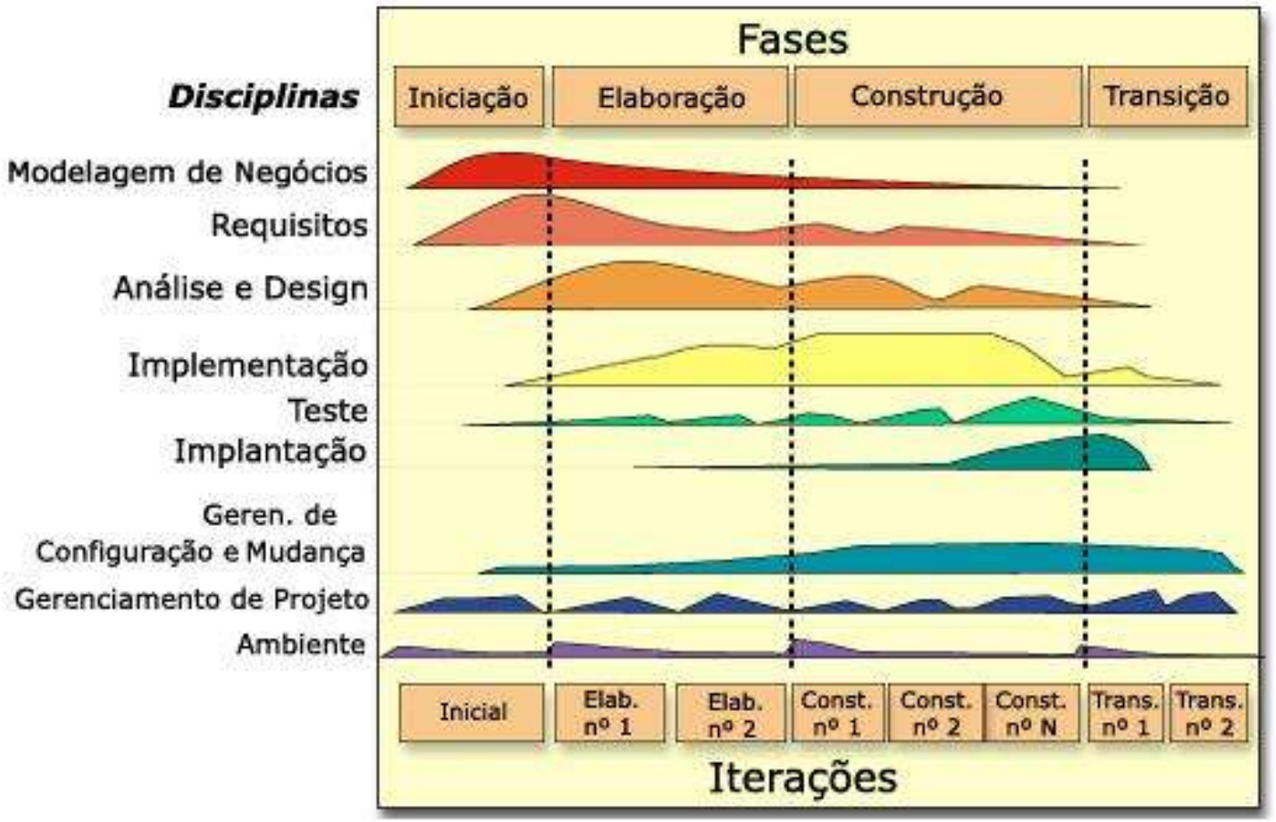
Identifique os componentes adequados → Procure componentes na biblioteca → Use os componentes disponíveis → Construa os que não existem → Adicione a biblioteca → Construa a interação

O Processo Unificado



- Surge com a necessidade de um processo de software dirigido a casos de uso, centrado na arquitetura, iterativo e incremental
- Combinação das melhores características dos diversos processos;
- Ênfase na comunicação com o cliente e métodos simplificados
- UML nasce desce processo
- Fases: Concepção, Elaboração, Construção, Transição, Produção.

RUP – IBM



- Mais relacionadas ao negócio que aspectos técnicos;
- Cada fase deve ser desenvolvida de forma iterativa (varias vezes) com o resultados incrementais;
- O RUP foi projetado em conjunto com UML;
- Práticas Fundamentais do RUP:
  1. Desenvolver o software iterativamente (foco no cliente)
  2. Gerenciar requisitos (análise de mudança/impacto)
  3. Usar arquitetura baseada em componentes
  4. Modelar o software visualmente (UML)
  5. Verificar a qualidade do software
  6. Controlar as mudanças do software

Tendência de entrega acelerada de software

MAIS RÁPIDO, MAIS BARATO E COM BAIXO RISCO

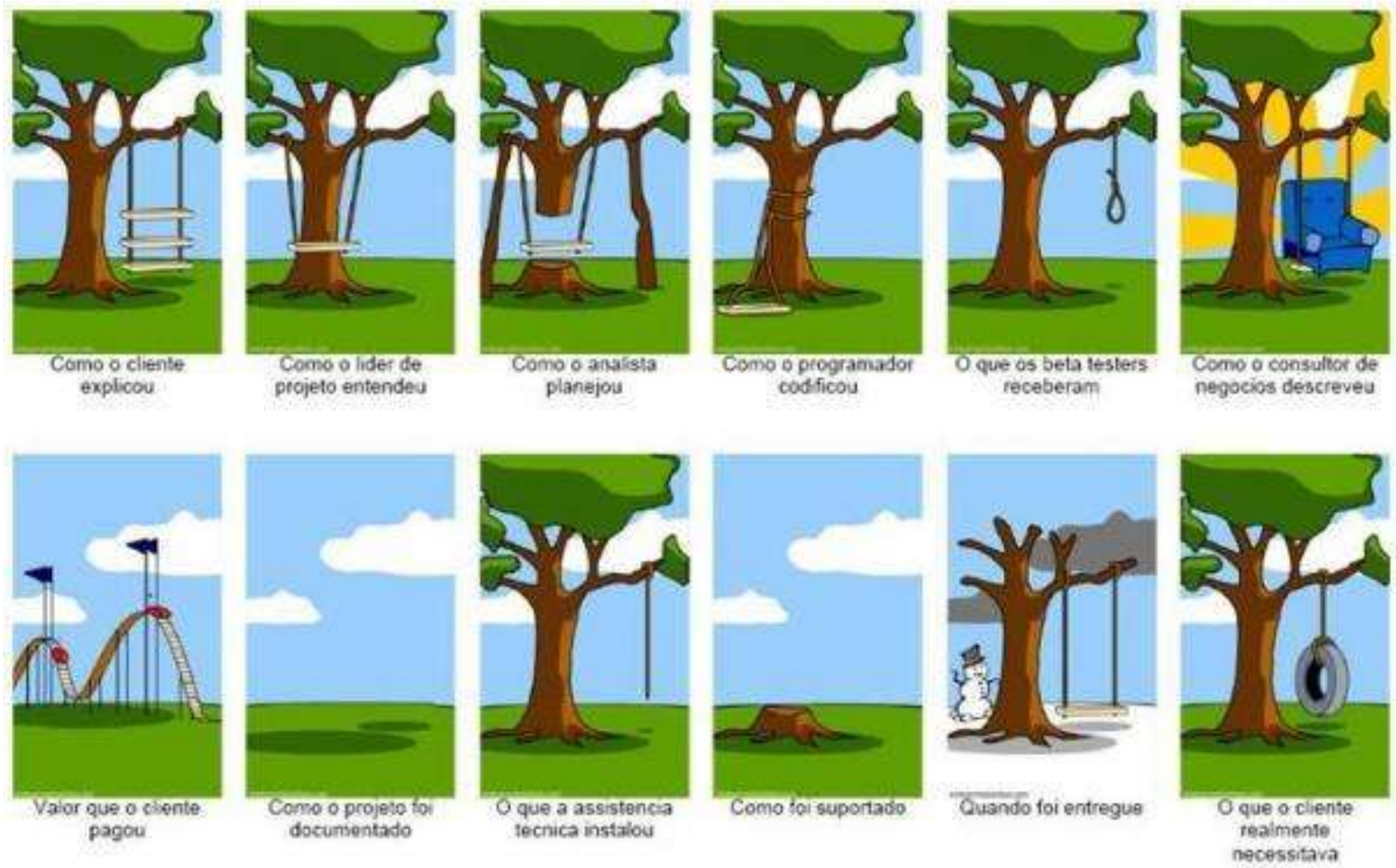
	1970 – 1980	1990	2000 - Hoje
Era	Mainframes	Cliente/Servidor	Comoditização e Nuvem
Tecnologia de Exemplo	Cobol, DB2	C++, Oracle, Solaris	Java, MySQL, PHP
Tempo do ciclo de Software	1 a 5 anos	3 a 12 meses	2 a 12 semanas
Risco	A empresa inteira	Uma linha de produto ou divisão	Um recurso do produto
Custo da falha	Falência, venda da empresa, demissões em massa	Perda de lucro, emprego do CIO	Insignificante

Será mesmo?

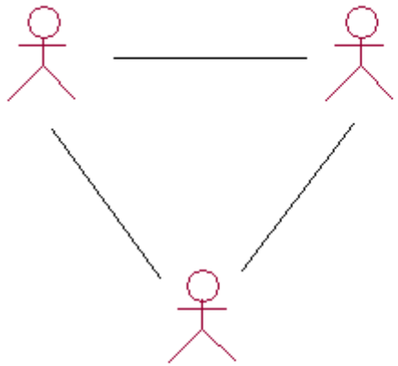
Fonte: Adrian Cockcroft, “Velocity and Volume” (or Speed Wins)

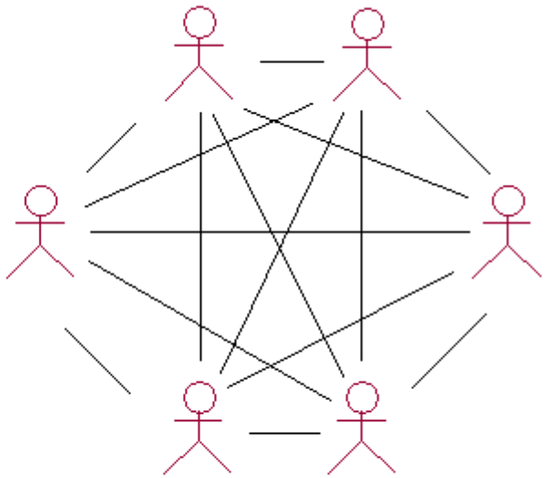
Motivações para mudar os modelos

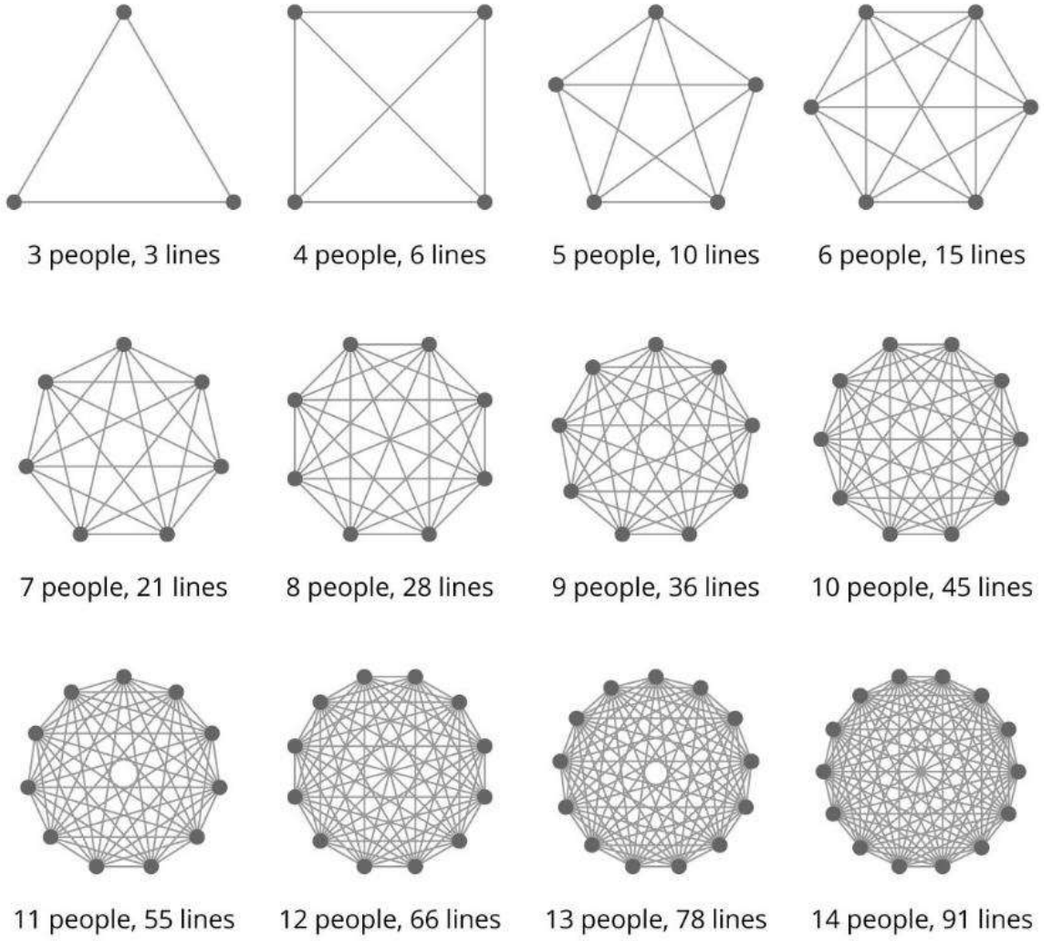
Desenho clássico



Motivações para o Agile



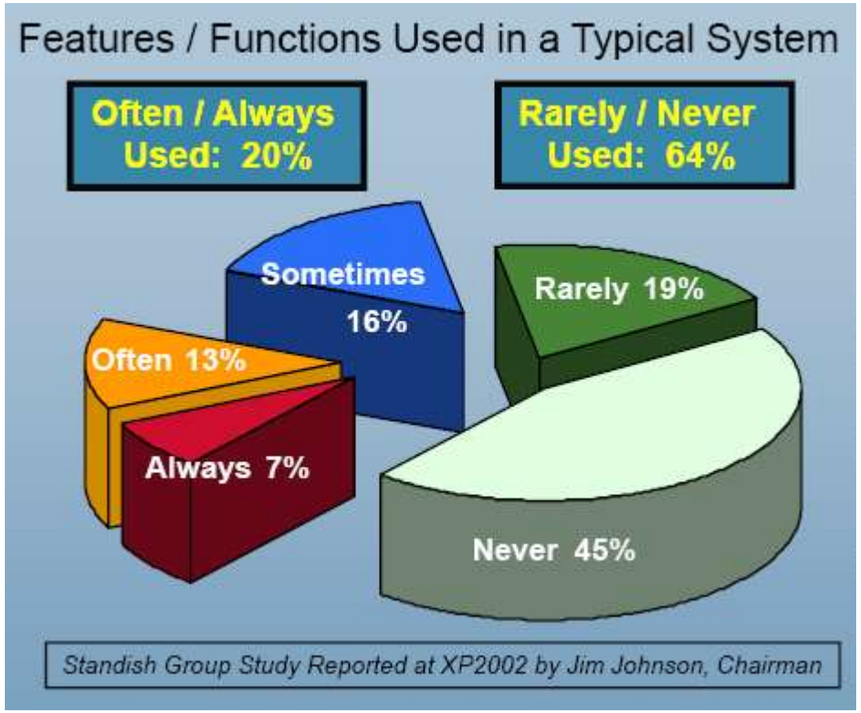




https://www.leadingagile.com/2018/02/applying-brooks-law/

A comunicação fica muito complexa em equipes grandes.

Motivações para o Agile



https://theagileexecutive.com/2010/01/11/standish-group-chaos-reports-revisited/

https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

- Aproximadamente metade das funcionalidades desenvolvidas não são utilizadas.
- 20% das funcionalidades são realmente utilizadas.

Motivações para o Agile

MUNDO VUCA

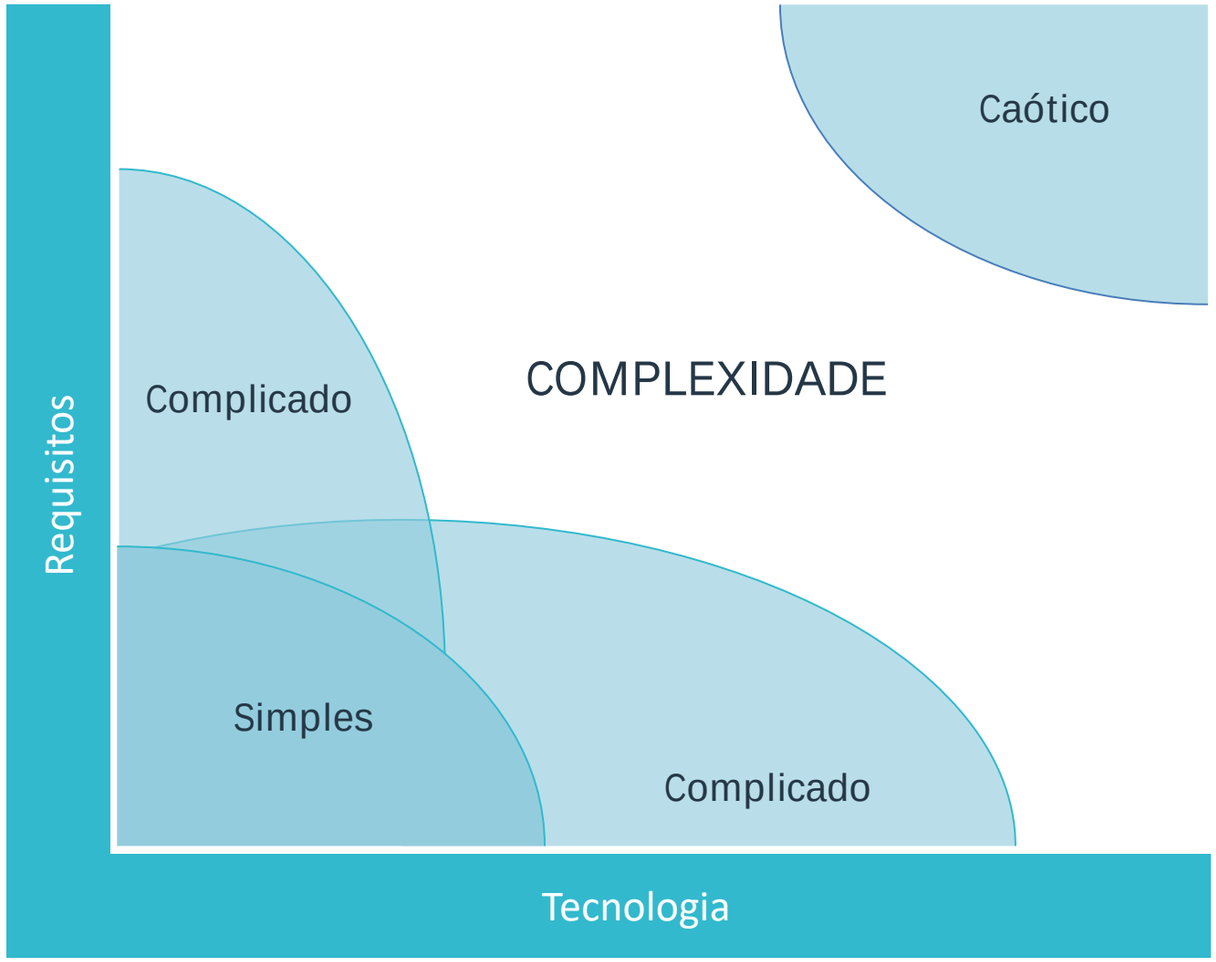
- Conceito vem das forças armadas americanas.
- O mundo atual é caótico.
- Se você sabe pouco sobre a situação e não consegue ter ações previsíveis, você não enfrenta nem o primeiro quadro, que é o da ambiguidade.
- Em português fica VICA. VUCA vem do Inglês.

Motivações para o Agile

"Teoria do Caos"

Matriz Stacey – Sistemas Complexos

Longe do Acordo
Ex: Requisitos Mutantes

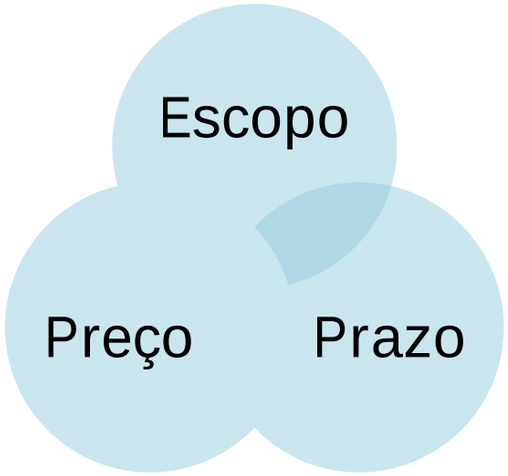


Longe da Certeza/Segurança
Ex: Tecnologia Nova

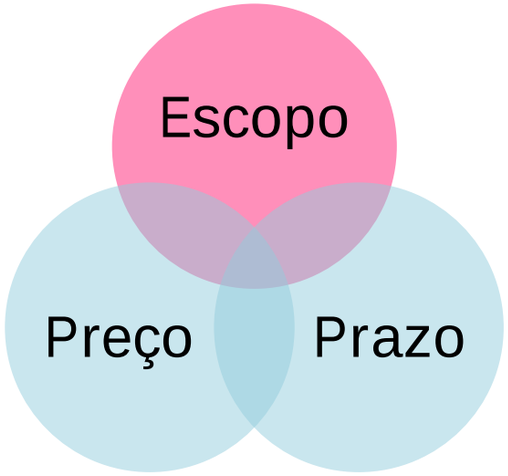
- Desenhado para ajudar a entender os fatores que contribuem para a complexidade dos sistemas.
- A relação de Incerteza versus falta de um acordo aumenta a complexidade.
- Em ambientes caóticos as abordagens são pouco eficientes, mas ainda sim, melhor com elas. Ex: Kanban.

Adaptado: https://www.praxisframework.org/en/library/stacey-matrix/

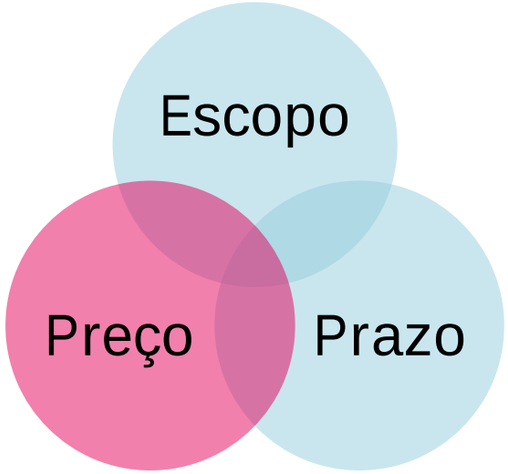
Teoria das Restrições – PMBOK



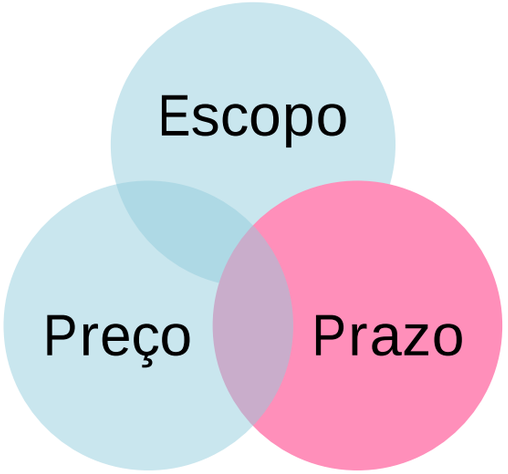
Nem em Conto de Fadas



Projeto Crítico
Escopo não pode alterar



Restrições de Orçamento

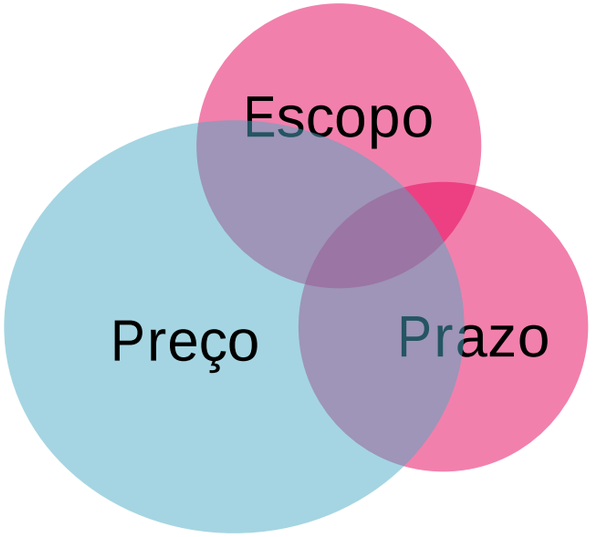


Projeto Urgente
(todos são ☺)

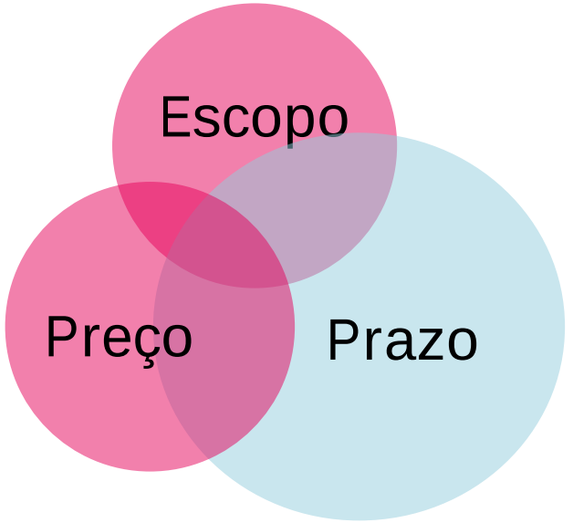
Estes são os cenários mais simples...

Teoria das Restrições – PMBOK

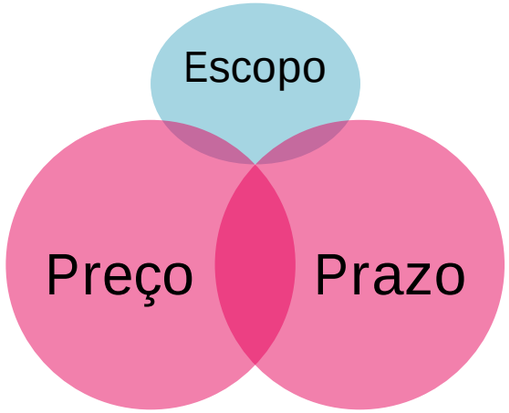
...Agora sim, a vida como ela é...



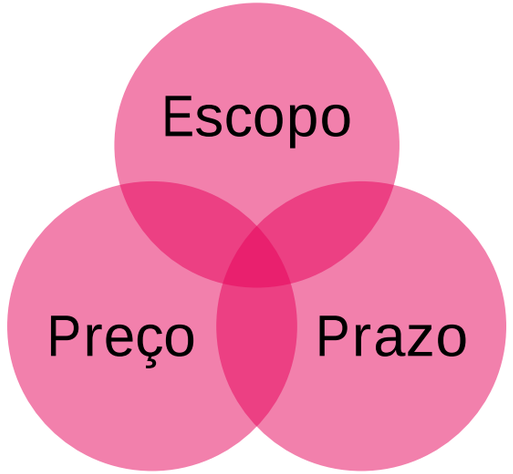
Projeto Crítico e Urgente
(Precisa de mais $)



Projeto Crítico e com Pouco $
(Aumenta o Prazo)



Pouco $ e Urgente
(Reduz funcionalidade)



Falácia - Caos
(Coloque razão na situação)

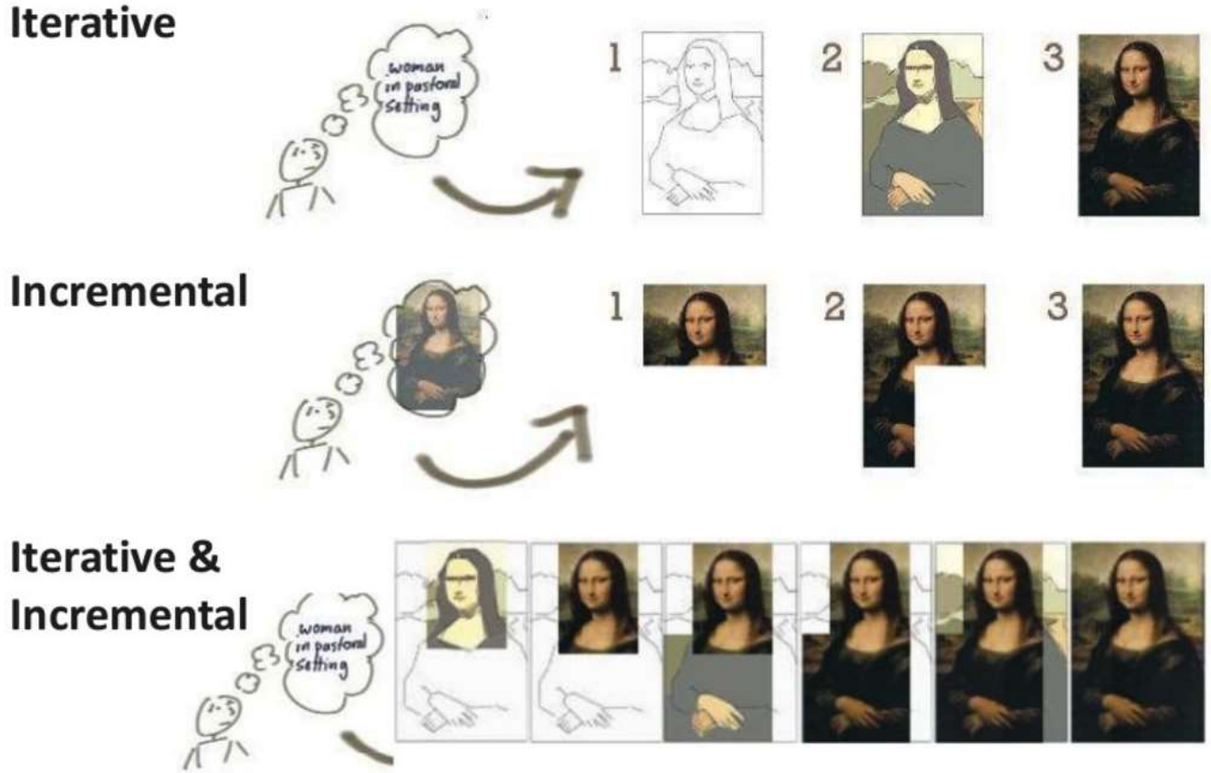
Manifesto Ágil - 2001

Os itens da direita também são importantes!

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

Agile

Iterativo (de iterar, repetir) + Incremental

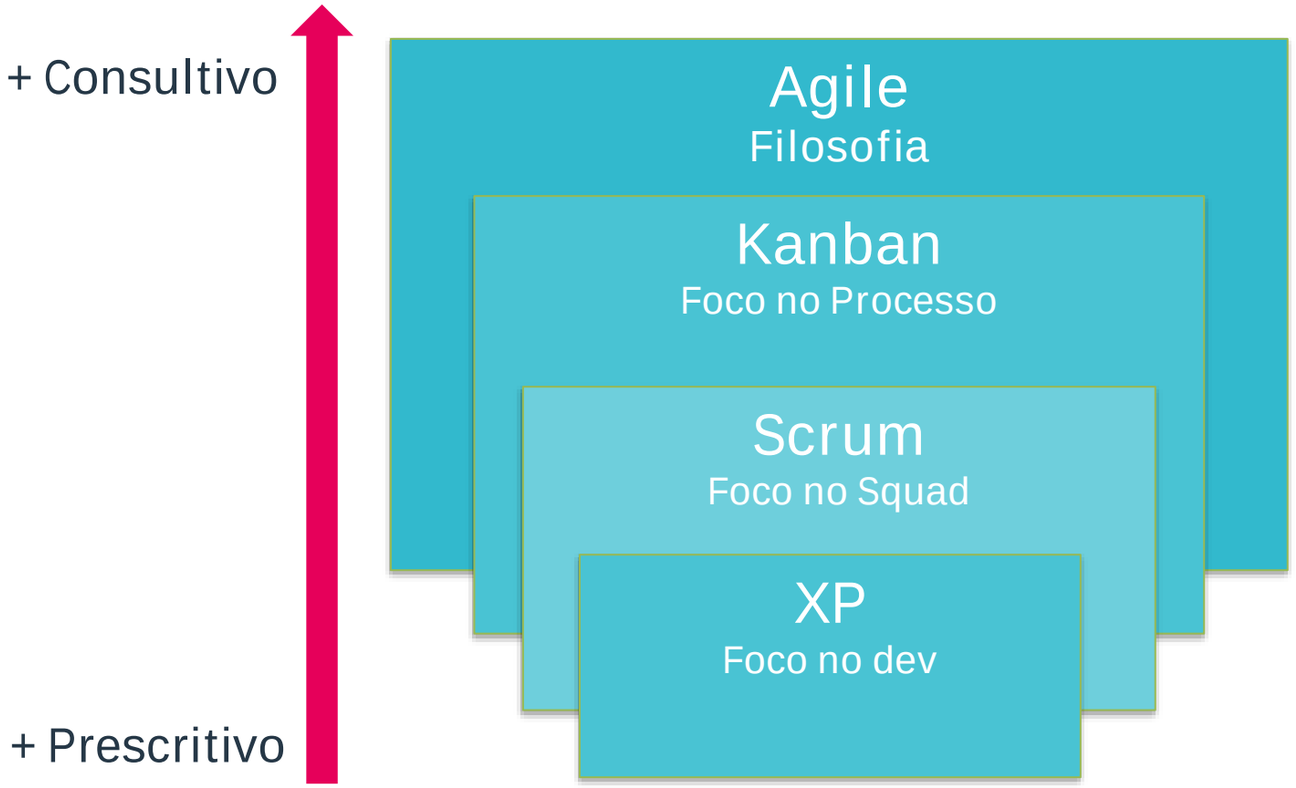


https://medium.com/@SupriyaL/why-and-how-to-use-agile-methodology-for-an-online-food-delivery-service-503ad5cf1941

- O que entregamos no Agile deveria ser utilizável;
- Há equipes ou empresas que trabalham apenas com o incremental; (Faseamento).
- Também existem projetos que trabalham apenas de forma iterativa. (Protótipo).

Agile

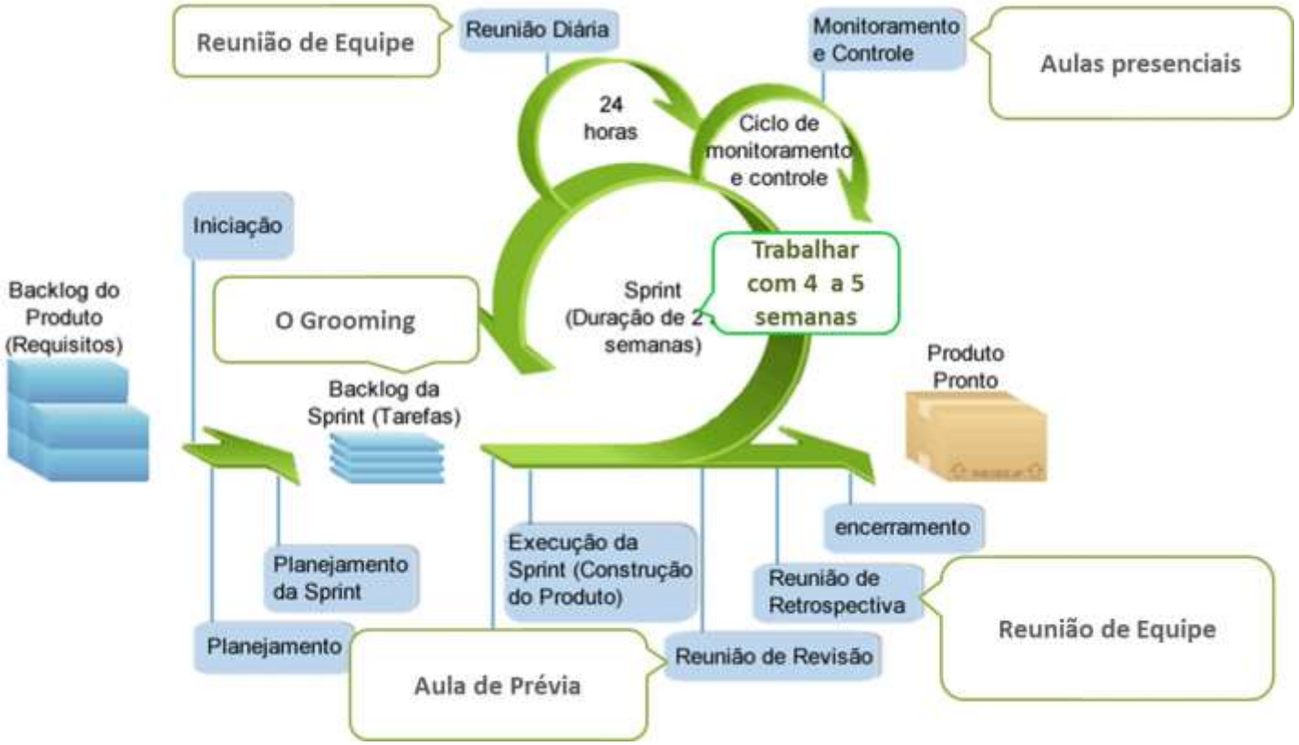
Iterativo (de iterar, repetir) + Incremental



- Prescritivo vem de prescrição, ou seja, mais diretivo, com mais regras.
- Podemos notar que Agile é uma filosofia, ou seja, não determina as regras.
- As empresas normalmente combinam as práticas e métodos.

Adaptado de: https://www.slideshare.net/kurtsolarte/sdc-sydney-2011solarte

SCRUM



Já vimos ao longo das aulas de Análise e Pesquisa e Inovação.

Apenas algumas dicas para quem quer se aprofundar.

https://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf

https://www.scrumstudy.com/portuguese

XP – Programação Extrema

Valores	Exemplos de Práticas
Comunicação	Programação em Pares, Reuniões em Pé, Envolver usuários em testes de aceitação
Feedback	Estimativas de tempo, TDD, ciclo rápido de desenvolvimento, integração contínua.
Coragem	Fazer correções, ser transparente, simplificar e refatorar códigos, jogar código fora, receber crédito por código completo.
Simplicidade	Cartões de papel para escrever as funcionalidades, não criar nada que não seja justificável pelo requisito.
Respeito	Saber ouvir, compreender e respeitar o ponto de vista do outro (empatia).

XP – Programação Extrema

Programação em Pares
(novato + experiente).
Novato no computador.
O programa é revisto por 2 pessoas, reduzindo erros.

Ritmo Sustentável
Trabalhar com qualidade, sem horas extras. Para isso, ambiente de trabalho e equipe precisam estar em harmonia.

Teste de Aceitação
Os testes são construídos pelo cliente.

Padrão de Codificação
A equipe de devs precisa estabelecer regras para programar e todos devem seguir estas regras.

Desenvolvimento Orientado a Testes (TDD)

Testes automatizados!
Testes sempre!

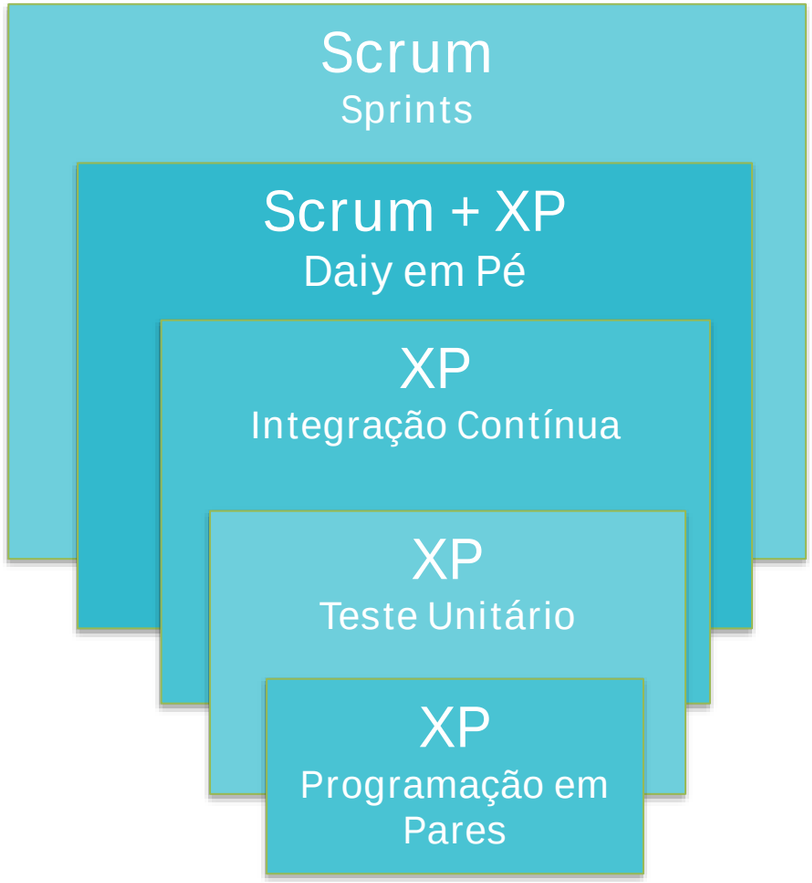
Refatoração
Processo de melhoria contínua da programação.
Evitar a duplicação e maximizar o reuso.

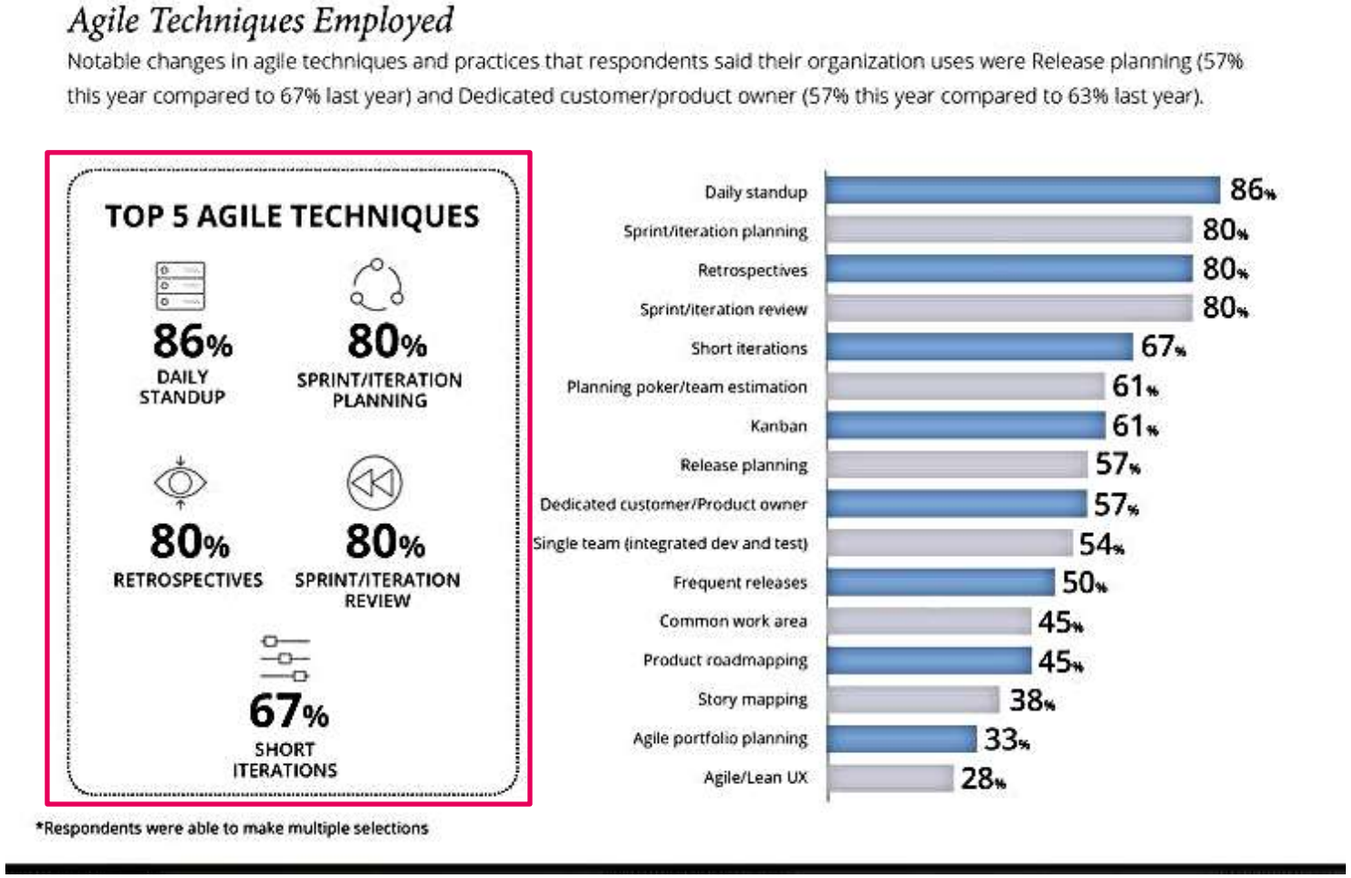
Integração Contínua
Saber o status real da programação.
Tem merge? Quebrou? Saiba antes!

Time Coeso
Comunicação!
Usar o socioemocional

Exemplo: Scrum + XP

As empresas combinam as práticas ágeis.

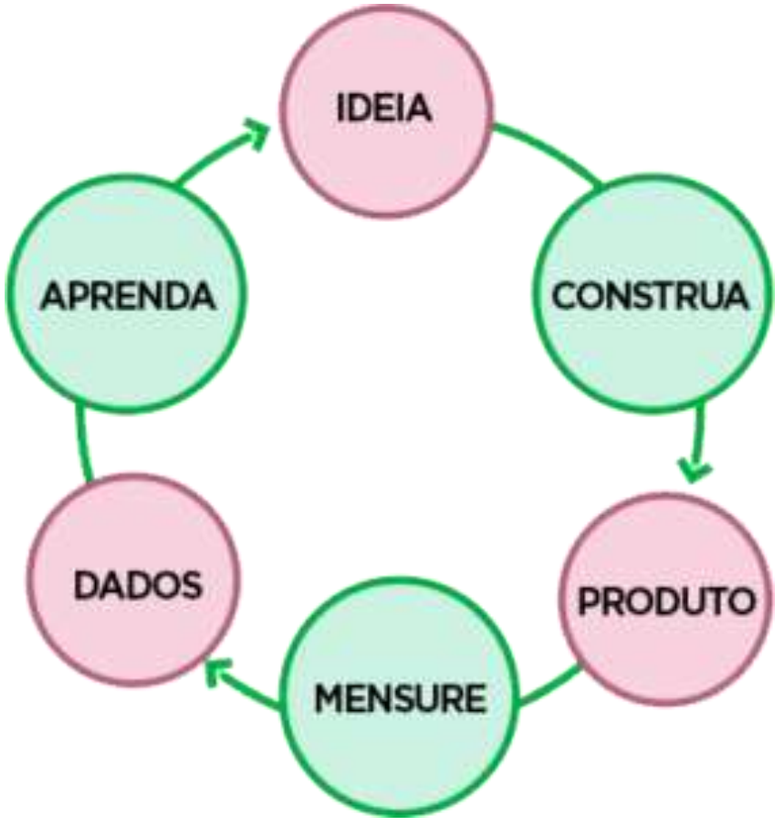


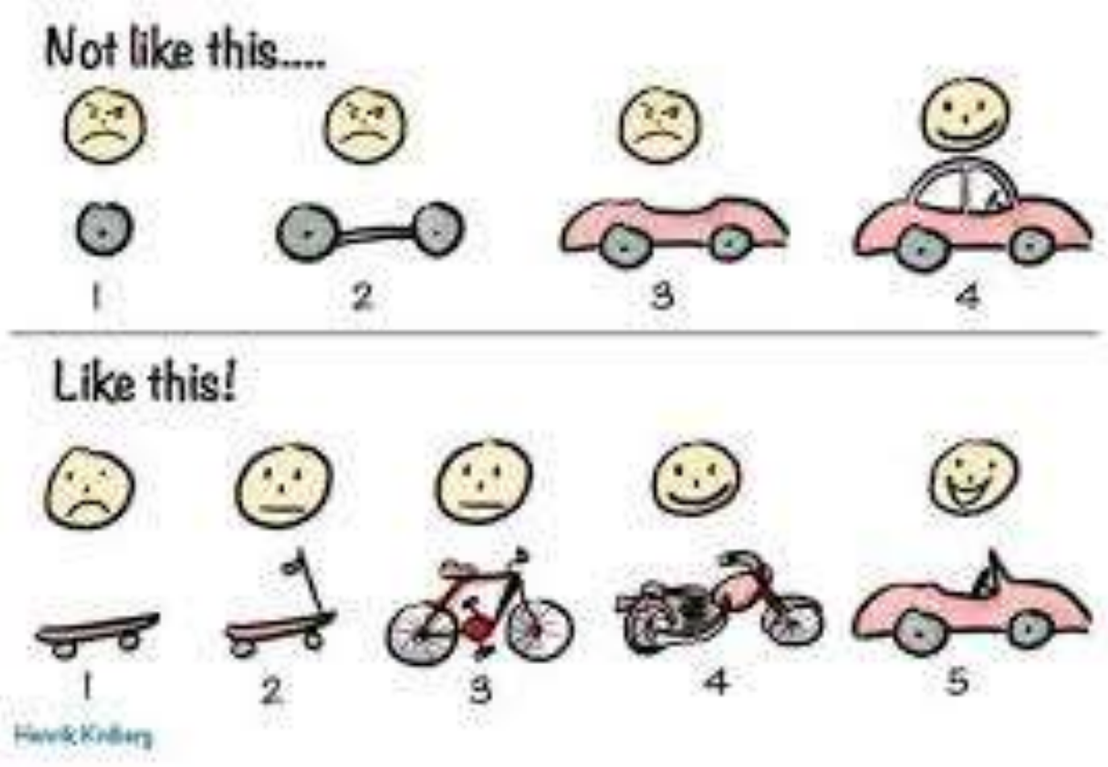


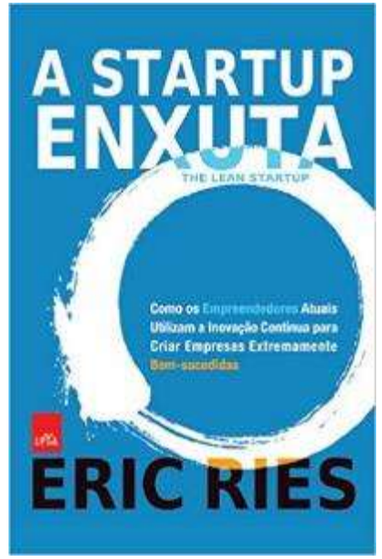
https://www.stateofagile.com/#ufh-i-521251909-13th-annual-state-of-agile-report/473508

Abordagens Modernas - MVP (Talvez nem tanto)

Mínimo Produto Viável. Criação de novos negócios de forma ágil.

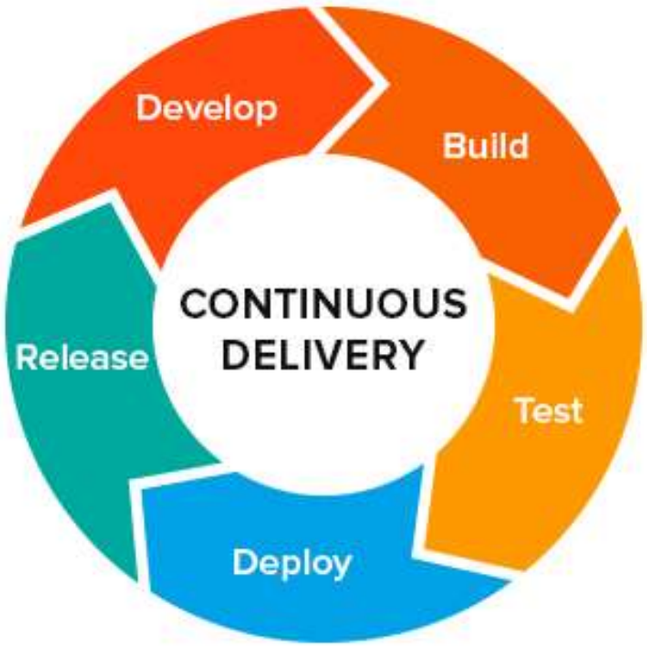




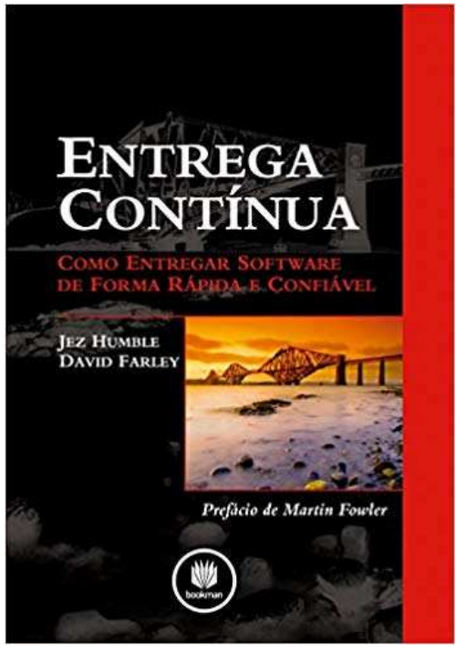


Se você já sabe muito bem o que vai fazer ou é uma nova demanda que é extensão do produto atual, por que deve utilizar esta abordagem?

Abordagens Modernas – Continuos Delivering



Abordagem em que o objetivo é entregar código em produção com qualidade e mais frequência, ou seja, pequenas partes mais rápido, assim em tese a chance de erros e impacto dos erros deve reduzir.



Parece com alguma abordagem até aqui?
Novamente...nada se cria do zero....

SCRUM x PMBOK. Qual o melhor?

Cuidado!

A Tartaruga pode ser Ninja!

O coelho pode ser lento!

Tradicional vs Ágil

Tradicional	Ágil
Orientado por atividade. Sucesso é entregar o Planejado.	Orientado por produto. Sucesso é entregar o desejado.
Foco no processo. Seguir o processo garante a qualidade.	Foco em pessoas. Pessoas comprometidas e motivadas garantem a qualidade
Rígido. A especificação tem que ser seguida. Detalhar bastante o que não é conhecido.	Flexível. Os requisitos podem mudar. Conhecer o problema e resolver o crítico primeiro.
Para projetos estáveis (Poucas mudanças de escopo)	Projetos que mudam constantemente
Projetos Grandes	Projetos Pequenos (5 a 10 pessoas) Mas pode ser usado para projetos grandes
Gerente de Projetos tem controle total	Gerente de Projetos é um facilitador (SCRUM Master)
Equipe tem papel claro, definido e controlado	Equipe tem autogerenciamento, é colaborativa e tem mais de um papel
Cliente tem papel definido. Lista requisitos e Valida.	Cliente precisa fazer parte da equipe do Projeto.
Planejamento Extenso e Detalhado. Equipe nem sempre participa.	Planejamento Curto e TODOS são envolvidos.
Muitos artefatos, mais formal Documentação é garantia de confiança	Poucos artefatos, menos formal Comunicação é garantia de confiança.

Adaptado de Chin, 2004

Exercício – Escolha de Processos

Caso Z – Sistema para uma empresa da indústria de celulose
Seu grupo foi contratado para desenvolver um software que deve monitorar o crescimento das árvores e o aparecimento de pragas da “plantação” de eucaliptos que é utilizada posteriormente para a fabricação de papel. A indústria e a plantação ficam sediadas no norte do Paraná em local com acesso restrito.
Seu software deverá gerenciar dispositivos IoT desenvolvidos por uma fábrica de Singapura. A especificação já existe mas o hardware está em fase final de homologação. A integração deverá ser utilizada utilizando C++.
Os dispositivos ficarão no meio da plantação, ou seja, uma vez disponibilizados no ambiente não está orçado o custo de recolocação. O software dos dispositivos só se atualiza via WIFI o que significa que as mudanças de firmware são bem difíceis de fazer. O software envia os dados usando a rede LoRa.
Seu time tem 2 anos para desenvolver, homologar e implantar o software e a primeira coisa a fazer é definir como irão trabalhar.
O escopo é bem amplo e inclui além da integração: Dashboards, Aplicações para controle operacional, Apps, Intranet (WEB) como Portal de Gestão e Configuração

Qual modelo/abordagem será utilizada? Justifique o motivo da escolha e os benefícios.

Quais serão as etapas e atividades que serão utilizadas na abordagem escolhida, importante listar e explicar o que ocorre na etapa.

Exercício – Escolha de Processos

Modelo de resposta meramente ilustrativo, as respostas não tem relação com o caso

Abordagem: Codifica-remenda + Cascata
Justificativa: O sistema não é crítico e os modelos acima citados são complementares. Com este modelo daremos liberdade para o desenvolvedor agir conforme seu bom senso e teremos uma equipe integrada e feliz.

Etapas, atividades relevantes ou práticas:
Especificação: Realizaremos reuniões com o cliente e faremos a especificação completa antes de iniciar o projeto.
Protótipo: Será gerado um protótipo para validar se o sistema consegue integrar-se com o equipamento.
Desenvolvimento: Será dividido em fases para que possamos receber dinheiro do cliente a cada entrega, afinal o projeto é de 2 anos.
Cronograma: Utilizaremos cronograma para planejamento.