SÃO
PAULO
TECH
SCHOOL

Engenharia de Software

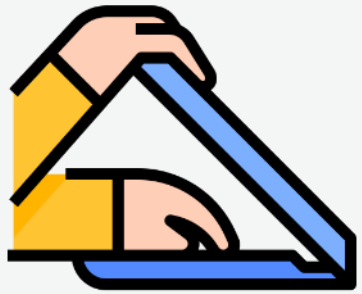
Processos de Software

Aula 12

Fábio Figueredo

Fabio.figueredo@sptech.school

Regras básicas da sala de aula



1. Notebooks Fechados: Aguarde a liberação do professor;

2. Celulares em modo **silencioso e guardado**, para não tirar sua atenção

- Se, caso haja uma situação urgente e você precisar **atender ao celular, peça licença para sair da sala** e atenda fora da aula.

3. Proibido usar Fones de ouvido: São liberados apenas com autorização do professor.

4. Foco total no aprendizado, pois nosso tempo em sala de aula é precioso.

- Venham sempre com o **conteúdo da aula passada em mente** e as atividades realizadas.
- Tenham caderno e caneta;
- **Evitem faltas** e **procure ir além** daquilo que lhe foi proposto.
- **Capricho, apresentação e profundidade** no assunto serão observados.
  - "**frequentar as aulas** e demais atividades curriculares aplicando a **máxima diligência no seu aproveitamento**" (Direitos e deveres dos membros do corpo discente - Manual do aluno, p. 31)

Regras básicas da sala de aula

As aulas podem e devem ser divertidas! Mas:

- **Devemos respeitar uns aos outros** – cuidado com as brincadeiras.
  - "observar e cumprir o regime escolar e disciplinar e comportar-se, dentro e fora da Faculdade, **de acordo com princípios éticos condizentes**" (Direitos e deveres dos membros do corpo discente - Manual do aluno, p. 31)

Boas práticas no Projeto

COMPROMISSO

COM VOCÊ: **ARRISQUE**, NÃO TENHA MEDO DE ERRAR

COM OS PROFESSORES: ORGANIZE A **ROTINA** PARA OS ESTUDOS

COM OS COLEGAS: **PARTICIPAÇÃO ATIVA** E PRESENTE

COM O PROJETO: **RESPEITO** E FLEXIBILIDADE

Boas práticas no Projeto

Reações **defensivas** não levam ao envolvimento verdadeiro!

Transforme cada problema e cada dificuldade em uma **OPORTUNIDADE** de aprendizado e crescimento.

EVITE:

- Justificativas e Desculpas
- Transferir a culpa
- Se conformar com o que sabe
- Se comparar com o outro

Dica: **Como ter sucesso** (*Maiores índices de aprovações*)

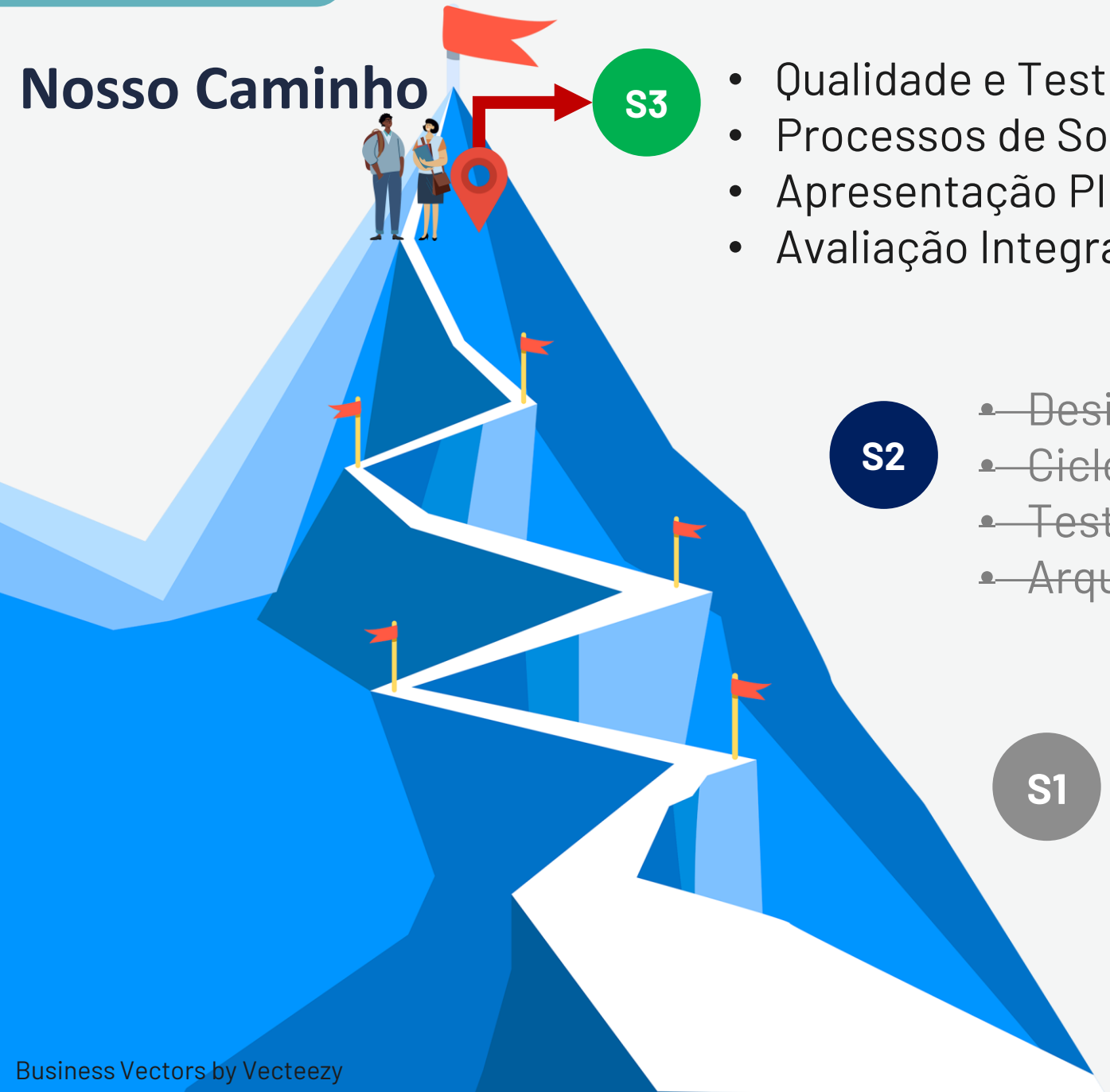
Comprometimento

- Não ter faltas e atrasos. Estar presente *(Não fazer 2 coisas ao mesmo tempo)*
- Fazer o combinado cumprindo os prazos

Atitudes Esperadas:

- **Profissionalismo**: Entender que não é mais ensino médio *(Atitude, comportamento, etc.)*
- **Não estar aqui só pelo** estágio ou pelo diploma
- Não ficar escondido: precisa **experimentar**
- **Trabalhar** em grupo e **participar** na aula
- **Não ser superficial** ou "achar que sabe"
- **Não se enganar** utilizando de "cola"
- Assumir a responsabilidade: Não colocar a culpa em outra coisa. **Não se vitimizar.**

Nosso Caminho



S3

- Qualidade e Testes
- Processos de Software
- Apresentação PI
- Avaliação Integrada

S2

- ~~Design Inclusivo~~
- ~~Ciclo de Vida de Produto~~
- ~~Teste de Usabilidade~~
- ~~Arquitetura de Software~~

S1

- ~~Introdução a Engenharia de Software~~
- ~~Conceitos de UI e UX~~
- ~~Fatores Humanos~~
- ~~Personas~~
- ~~Design de Interface Básico~~

Palavra-chave dessa Sprint:

PRAGMATISMO

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.

Frase dessa sprint:

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

Break

> 10 minutos, definidos pelo professor.

Obs: Permanecer no andar, casos específicos me procurar.

Atenção: Atrasados deverão aguardar autorização para entrar na sala.

Tópicos da Aula

- Processos de Software
- Atividade

KAROOTS!

KAROOTS

- Hardening é um tipo de Teste de Segurança?
- O que mede o Teste de Carga?
- O que o Teste de Stress identifica?
- Para que serve o Teste de Conteúdo?
- Fale sobre o propósito do Teste de Navegação/Usabilidade?
- Segundo Presman, qual o conceito de Bug e Defeito?
- Para que serve o processo de Validação?
- E o processo de Verificação?

KAROOTS

- De quem é a responsabilidade do Teste Unitário?
- O que o Tester de Software (Equipe de QA) deve garantir?
- Para que serve o Teste de Regressão?
- Para que serve o Teste de Fumaça?
- O que são Mocks e Stubs?
- Quem não é obrigado a testar?

Aula de hoje!

PROCESSOS DE SOFTWARE

Vamos voltar no tempo...

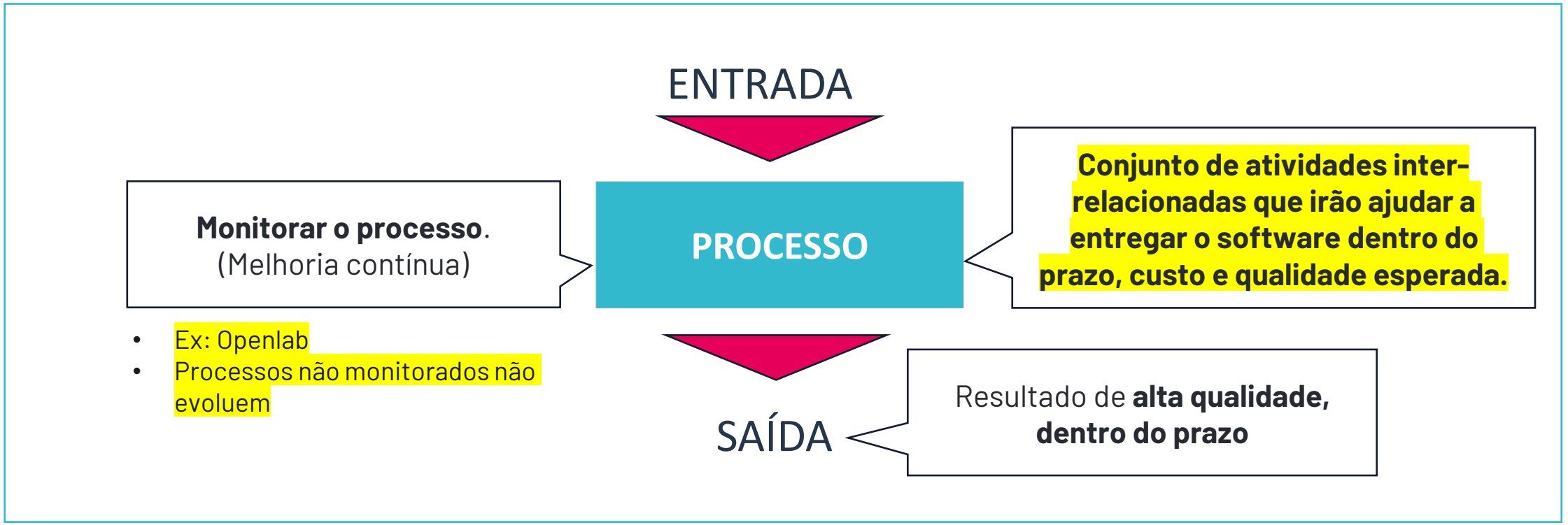
O que é Engenharia de Software

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**)

O que é Processo de Software?

É um **conjunto de atividades, métodos, práticas** e **técnicas utilizadas** na **criação, manutenção** e **evolução,** garantindo a **construção** de um **software** com **qualidade** e **dentro do prazo** e **orçamento estabelecidos**.



Engenharia de Software x Processo de Software

Engenharia de Software é uma **disciplina da Engenharia** que se dedica a **aplicação de princípios**, **processos**, **métodos** e **técnicas** para o **desenvolvimento de softwares de alta qualidade**.

Processo de Software é um **conjunto de atividades**, métodos e **técnicas** para **produzir softwares de qualidade**.

A **Engenharia de Software** usa os **Processos de Software** para atingir seu objetivo.

Contextualização

No trabalho tem um item que **SÓ EU** sei mexer...

Está errado!!!

Falta de compartilhamento de conhecimento é um problema grave dentro de uma equipe de desenvolvimento e **atrapalha** significamente **os processos de software** e consequentemente a **qualidade**.

Momento Pragmático – Socioemocional

NÃO EXISTE LUGAR PARA SMEAGOL!

Se for **possível**, **deverá ser reutilizado**, não **deixe o medo** e a **preguiça** de ler e entender o que outro fez, **fazer você trabalhar mais**. **Seja ainda mais preguiçoso e reuse**. Crie um ambiente que apoie a reutilização. **Entendendo** o que **foi feito você aprende mais** e com o tempo **conseguirá EVOLUIR o que foi feito**.

FACILITE O REUSO E O COMPARTILHAMENTO DE INFORMAÇÃO.

Diga NÃO ao heroísmo!

Se o **herói ficar doente**, **tirar férias**, ou fique **indisponível por qualquer motivo**, o andamento do **projeto será prejudicado**.

Portanto, o **conhecimento deve ser compartilhado entre os membros da equipe** para que mais pessoas possam atuar com a mesma demanda/ tecnologia.

Princípios Básicos do Desenvolvimento de Software

(David Hooker)

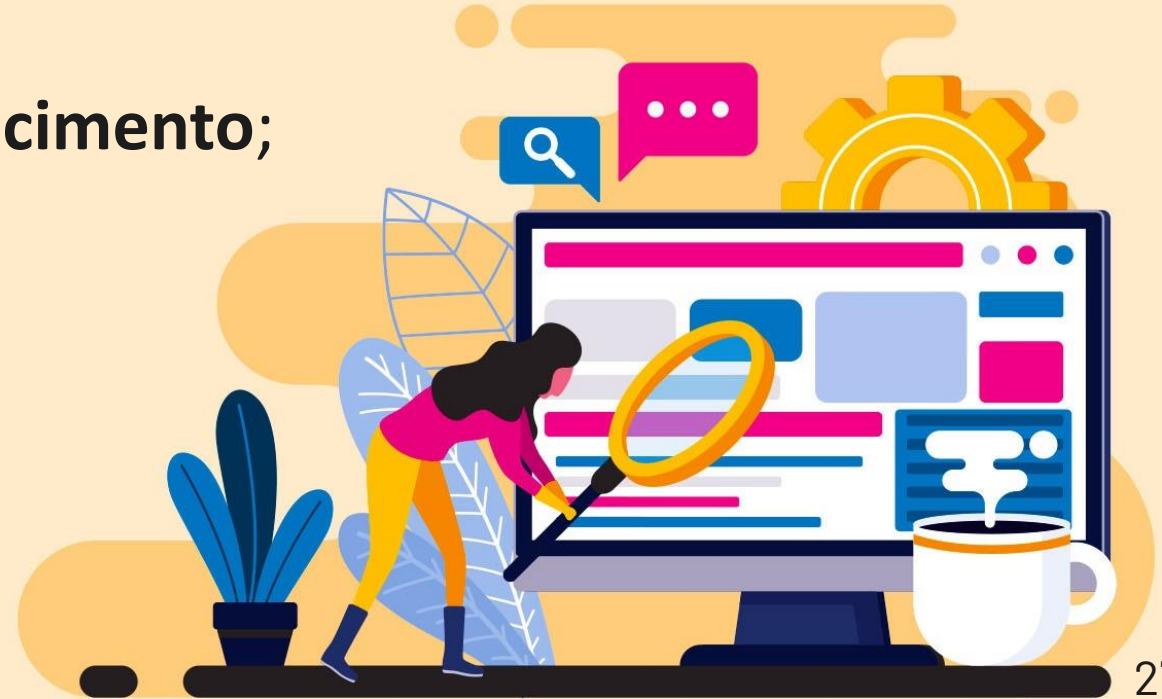
- RAZÃO DE EXISTIR
- KISS (KEEP IT SIMPLE, STUPID). Faça de forma simples, tapado.
- MANTENHA A VISÃO
- ESTEJA ABERTO PARA O FUTURO
- PLANEJE COM ANTECEDÊNCIA, VISANDO A REUTILIZAÇÃO
- PENSE!
- O QUE UM PRODUZ, OUTROS CONSOMEM

MITOS

- Se **decorar** os **padrões** e **procedimentos vou ter sucesso**!
- Se o **cronograma atrasar**, **põe mais gente que resolve**!
- 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**.

OBJETIVO DO PROCESSO DE SOFTWARE

- **Além** de produzir **softwares de alta qualidade**, **dentro do prazo** e do **orçamento estabelecidos**...
- Visa **promover a colaboração** e a **comunicação entre as equipes** envolvidas no projeto;
- **Facilita** o **compartilhamento de conhecimento**;
- **Identificação precoce de problemas**;
- **Reduz riscos e incertezas** do projeto;



O que é Mais importante na Eng. de Software?

O mais importante são as PESSOAS

- Software são feitos de **pessoas** para **pessoas**!
- **Conforme a tecnologia evolui, mais tarefas são automatizadas**. Hoje temos iniciativas como **lowcode**, **nocode**, onde os processos são cada vez mais automatizados e o objetivo é substituir parte do trabalho dos desenvolvedores.

Modelos de Processo de Software

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 Development)
Plataformas				
Fortran, Assembler	Cobol, PL1 (IBM)	Natural, C, C++, Clipper	VB, Delphi	Java, .NET

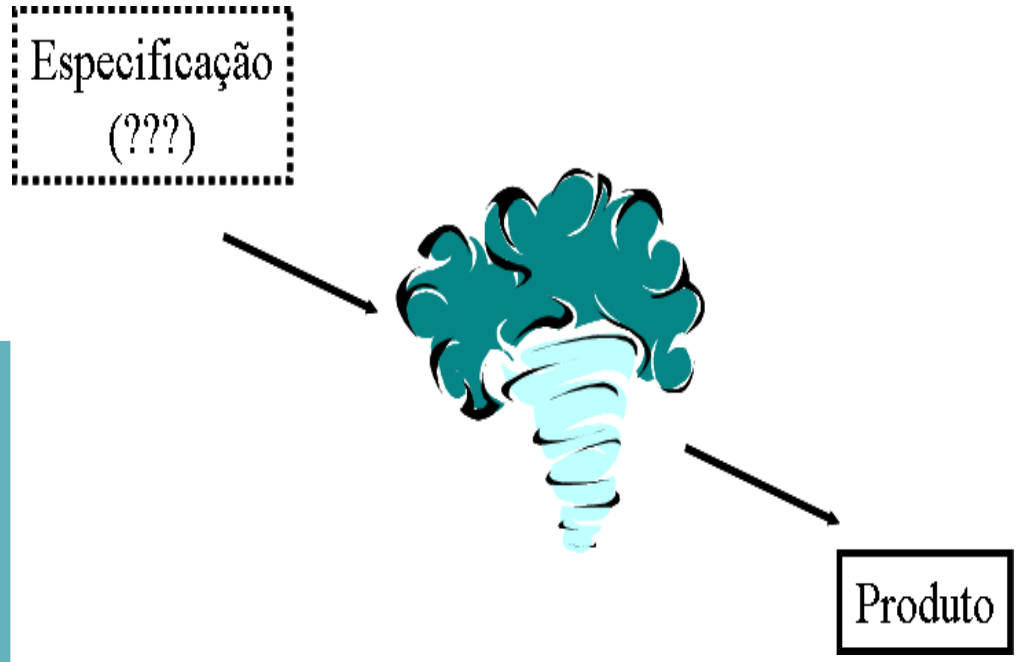
Modelos de Processos				
Waterfall		Evolucionários	Especializados: Componentes, OO, UML	?Agile
1960-1970	**1970-1980**	**1980-1990**	**1990-2000**	**Século XXI**

Existe a chance de termos outros modelos até o final do século.

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

- Antigamente, cada compilação levava muito tempo. Se desse erro, a nova compilação precisaria se agendada.

Modelo Codifica e Remenda



(Wilson de Pádua. 4ª edição)

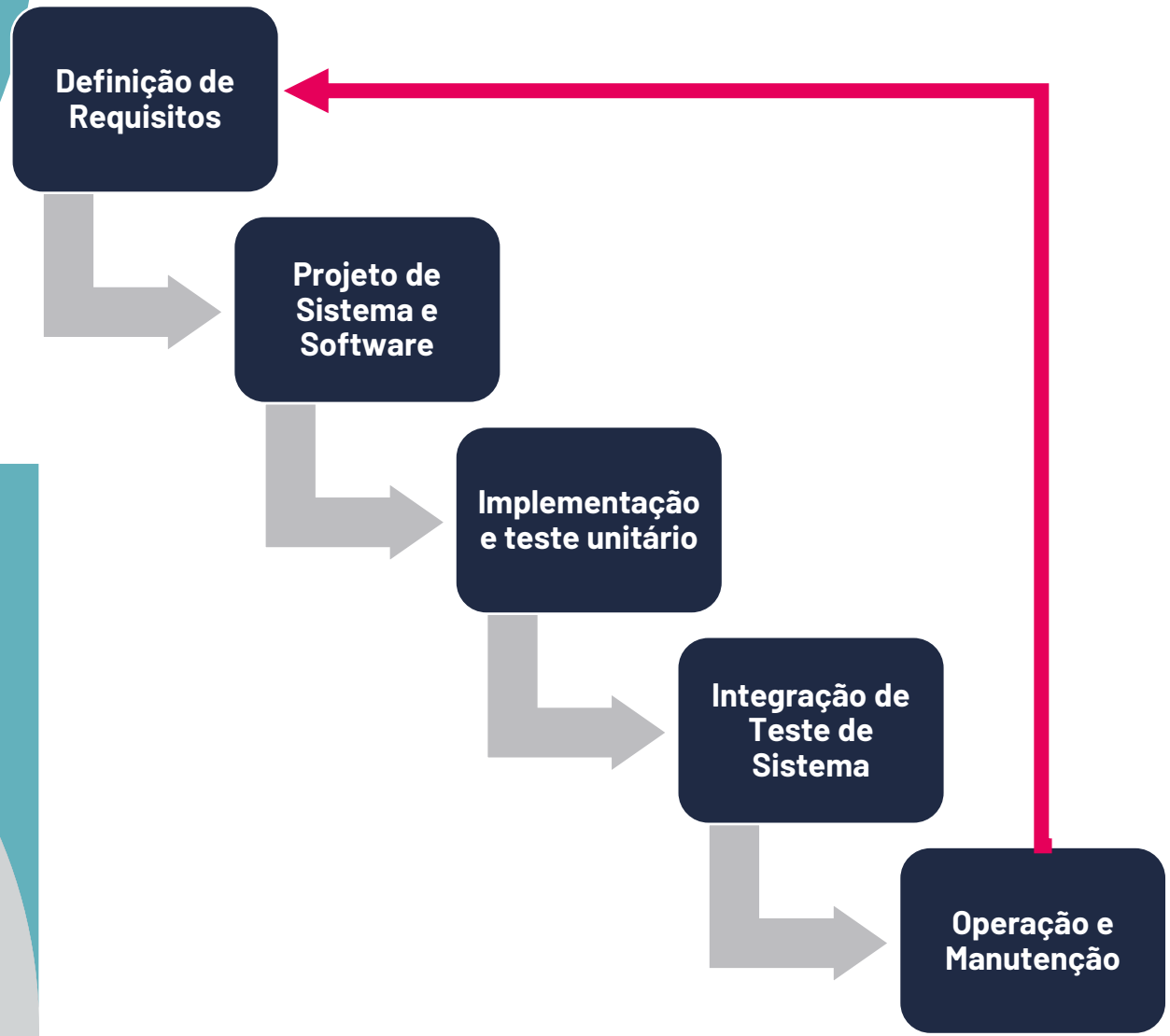
- Não é modelo espiral (apesar do redemoinho);
- **Sem especificação ou com especificação incompleta**;
- **Mais utilizado no mundo**;
- **Impossível de fazer gestão**;
- **Não permite assumir compromissos**;
- **A qualidade é baseada no acaso**;
- Existem diversas variações. “Dá seu jeito”, “Pizza de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

A VIDA COMO ELA É!

Você aprende diversos processos na faculdade, se empolga com o RUP e entende que esse é o caminho da felicidade, tanto que tira a certificação IBM.
Então você é contratado por um órgão governamental ligado a Educação, você fica feliz e pensa que agora você achou o lugar certo.
Então no seu primeiro dia, te apresentam o processo da empresa.
https://gohorseprocess.com.br/extreme-go-horse-xgh/
E aí?

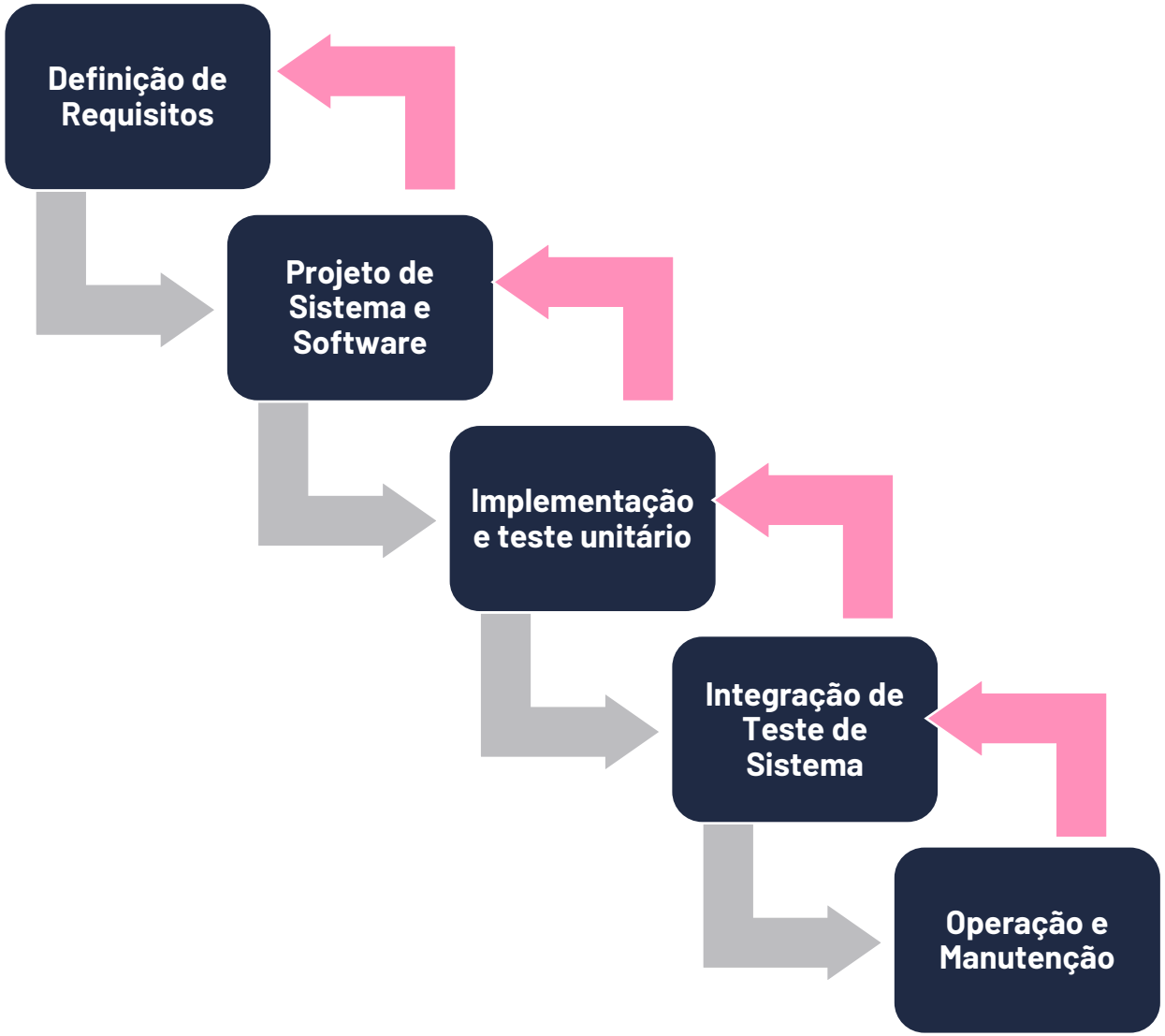
"FAZER DIFERENTE MOTIM VAI VIRAR. APRESENTAR FATOS E OPÇÕES AOS POUCOS VOCÊ DEVE. PACIÊNCIA FUNDALMENTAL É"

Modelo em Cascata



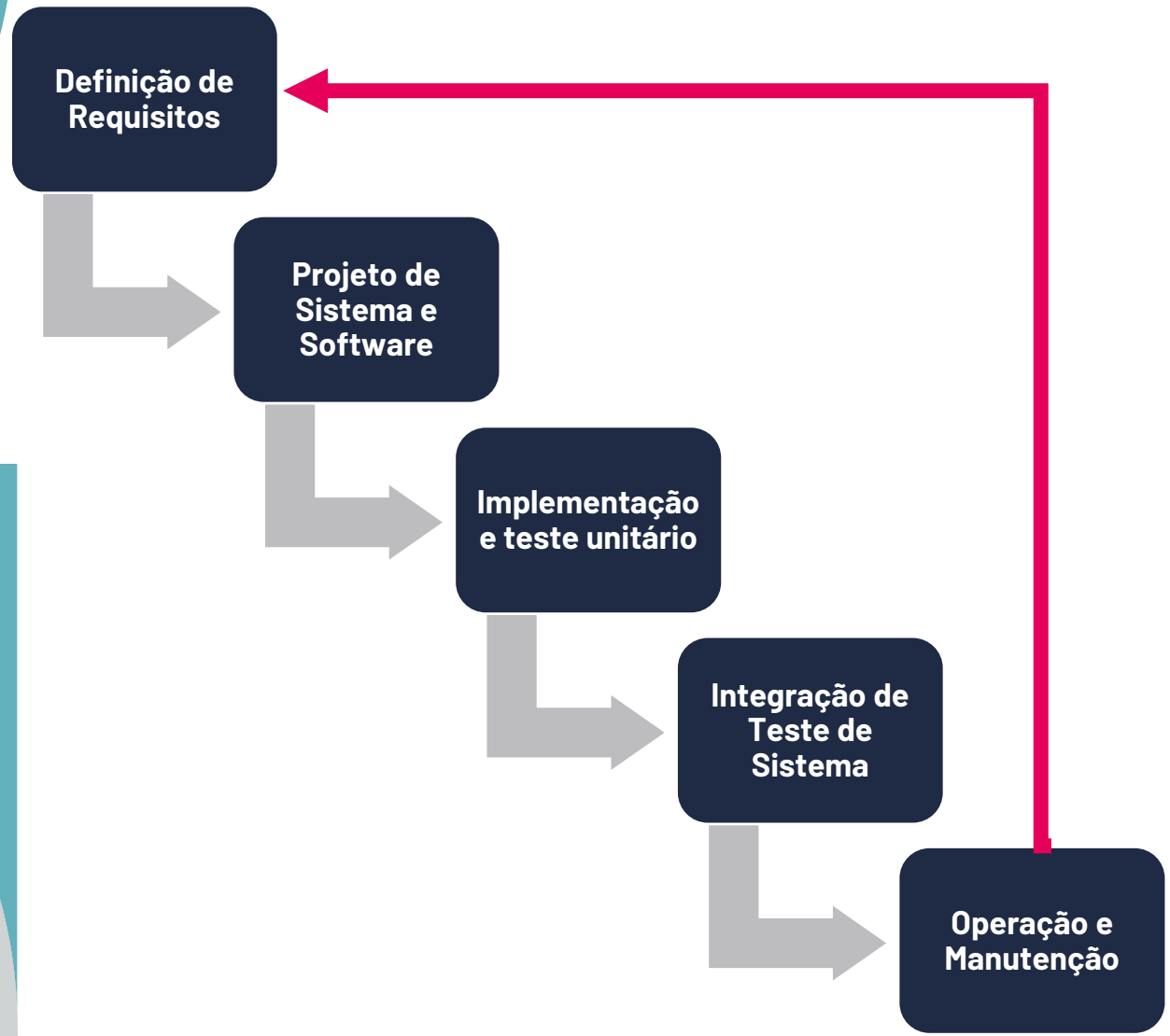
- Segue os processos gerais de Engenharia;
- **É o mais antigo**;
- É **sequencial**, **costuma levar mais tempo**;
- O resultado de cada fase consiste em uma validação formal de qualidade;
- **Falhas** e **melhorias são realizadas posteriormente após todo o processo terminar**;
- **Gera muita documentação**;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

Modelo em Cascata com Realimentação



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

Modelo em Cascata - Desvantagens



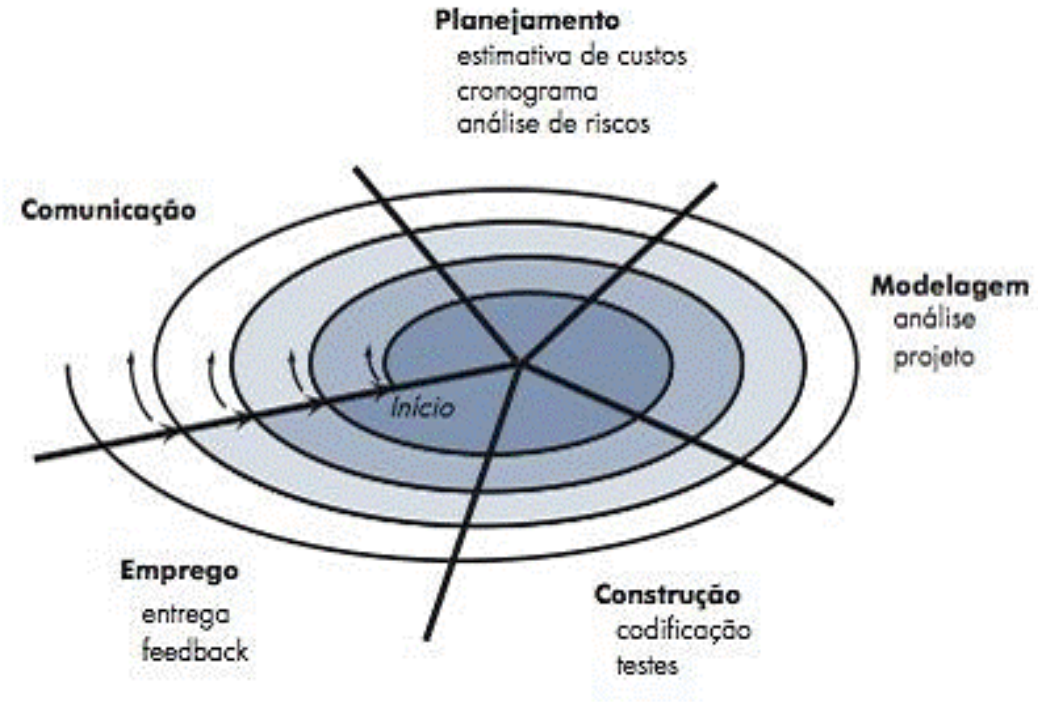
- **Você precisa terminar um estágio para começar o próximo**. Projetos reais raramente respeitam essa regra;
- **Dificuldade de conseguir todos os requisitos de uma única vez**;
- Falta de paciência do cliente em ver o resultado só no final de todo o ciclo;
- **As entregas são normalmente enormes**;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

> MODELOS DE PROCESSO EVOLUCIONÁRIOS

Estão diretamente relacionados à satisfação do cliente

- **Se adaptam** melhor às **mudanças de requisitos**;
- Permite um **feedback rápido** dos **usuários** e dos **clientes**, o que melhora a **qualidade** e **satisfação com o software**;
- **Reduzem** os **riscos de desenvolvimento**, pois as **falhas** podem ser **detectadas** e **corrigidas mais cedo**;
- **Facilitam** a **entrega antecipada** de **funcionalidades**;

Modelo de Processo Evolucionário – Espiral

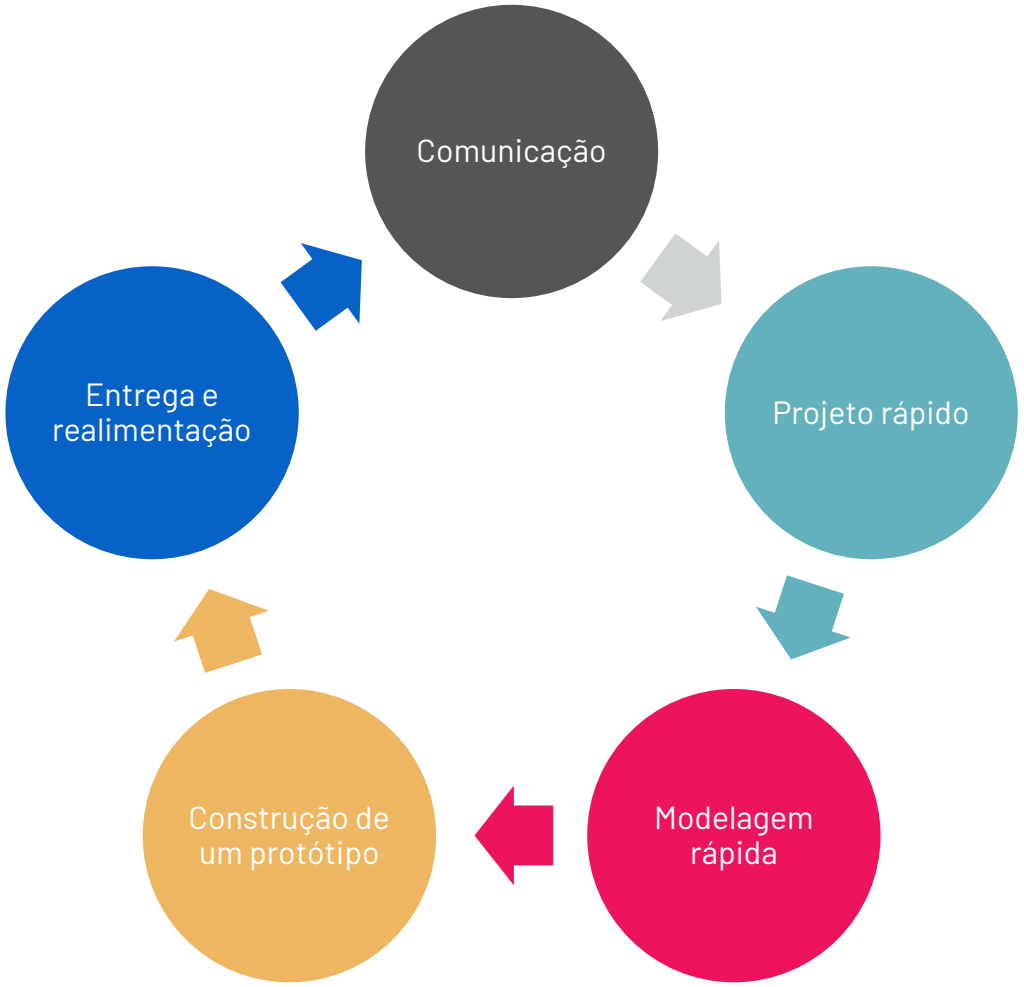


sommerville

- **Fala com o cliente, modela, constrói, entrega e repete**;
- **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;
- Enfatiza a análise e gestão dos riscos;
- **Os riscos são considerados a cada fase de evolução do projeto.**
- **Custo é alto**;
- **Difícil convencer o cliente que está tudo sob controle**;

- Planeja, modela, constrói, entrega, planeja, modela, constrói, entrega,... – Para cada ciclo precisa de mais prazo e mais $. Os clientes não gostam muito desse modelo de processo.

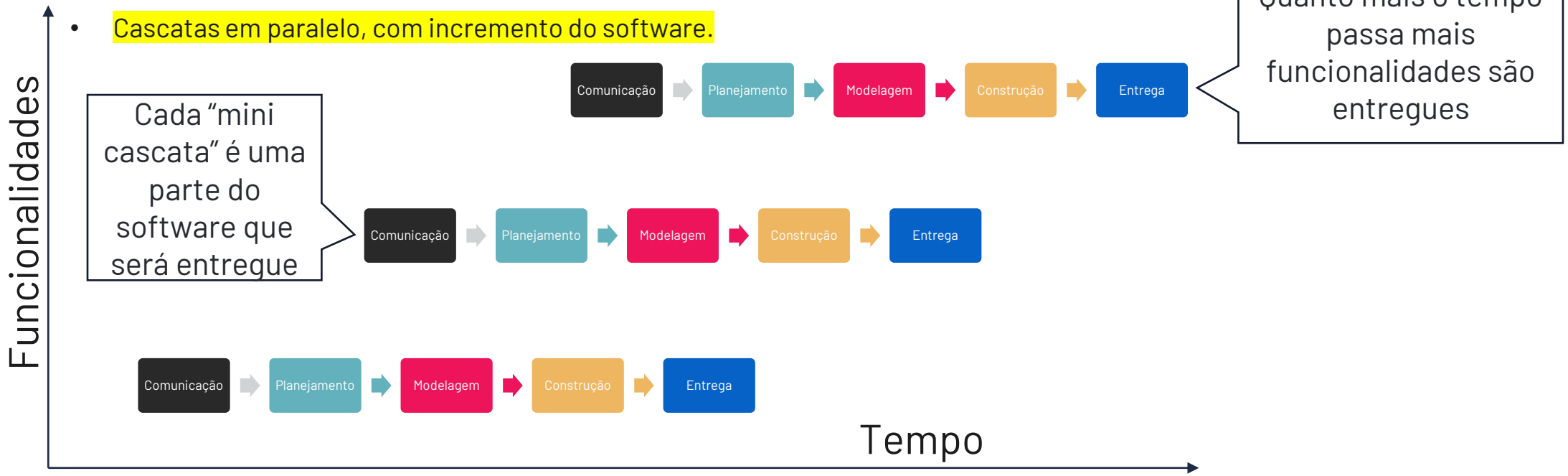
Processo Evolucionário – Prototipação



- **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

- Cascatas em paralelo, com incremento do software.

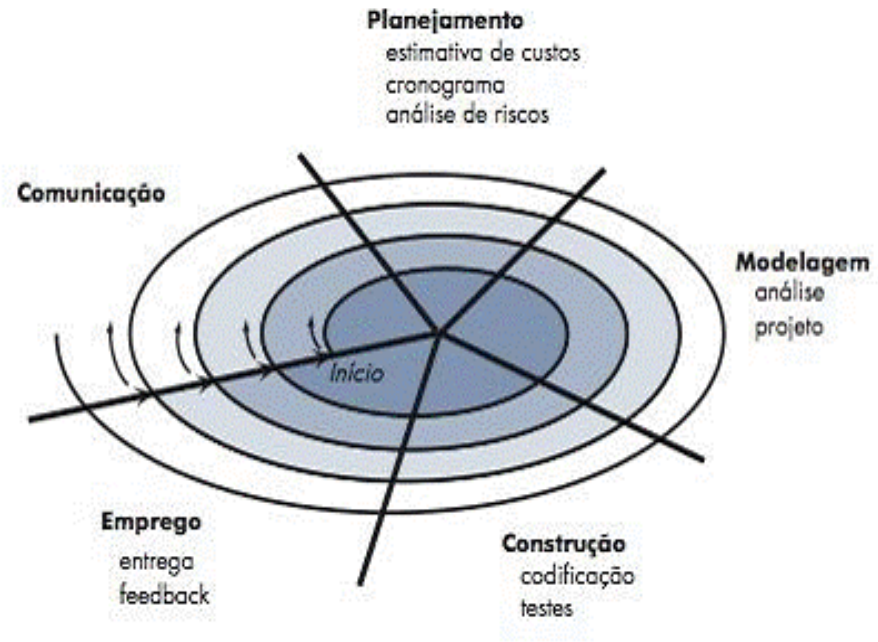


- **Combina o uso do modelo cascata aplicado a forma iterativa (repetitiva)**;
- **Tem fluxos paralelos**, o desenvolvimento ocorre com entregas sucessivas; podem existir equipes diferentes em cada incremento;
- **Cada sequência gera uma entrega**; O cliente devolve feedback a cada entrega;
- **Os requisitos 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, pois você foca em uma cascata de cada vez;

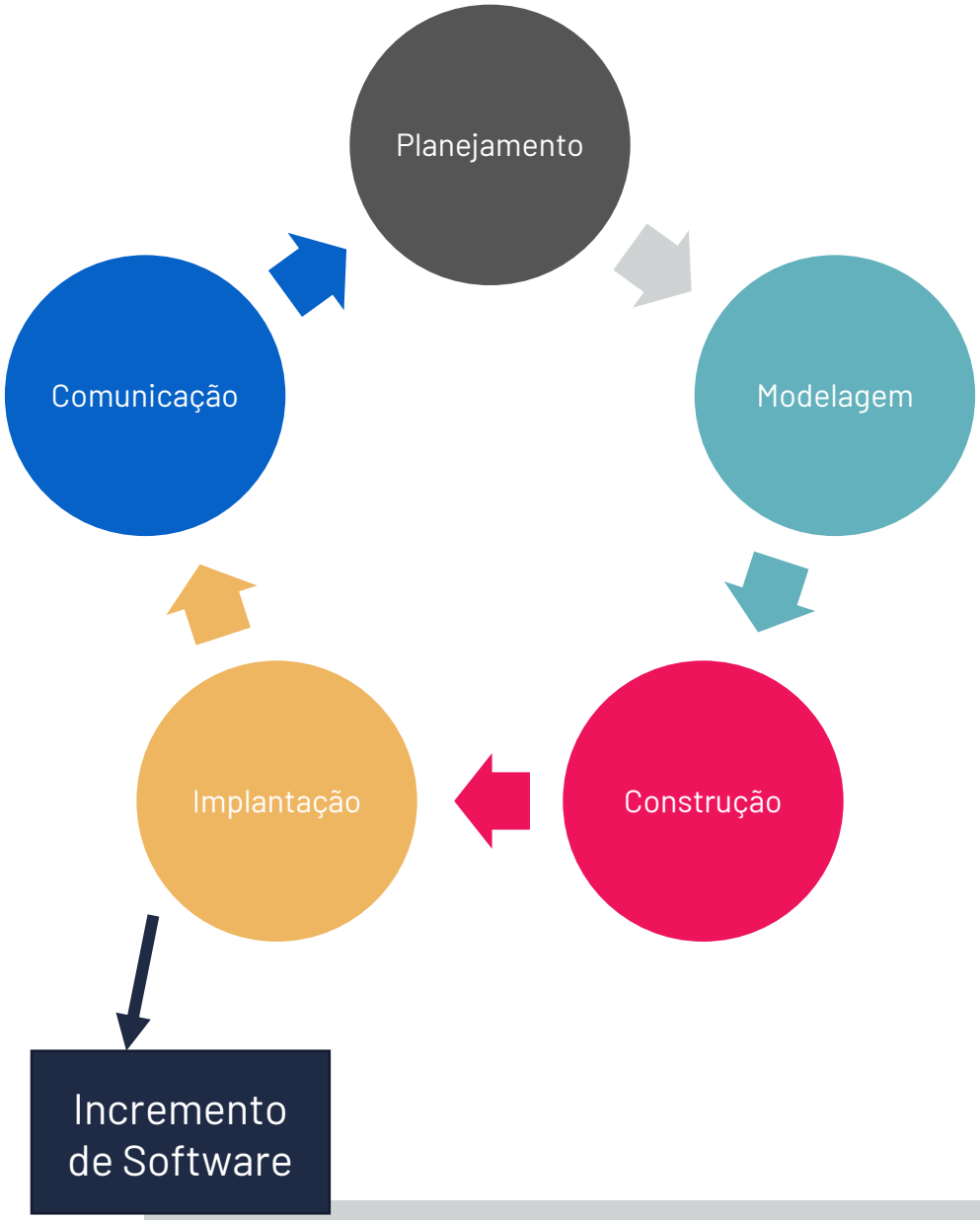
Processo Especializado- Baseado em componentes



- Baseado no modelo espiral;
- Envolve o desenvolvimento da **aplicação a partir de componentes pré-existentes** que são integrados a arquitetura – **Componentes já existentes na empresa**;
- **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.
- Expiral + Componentização.

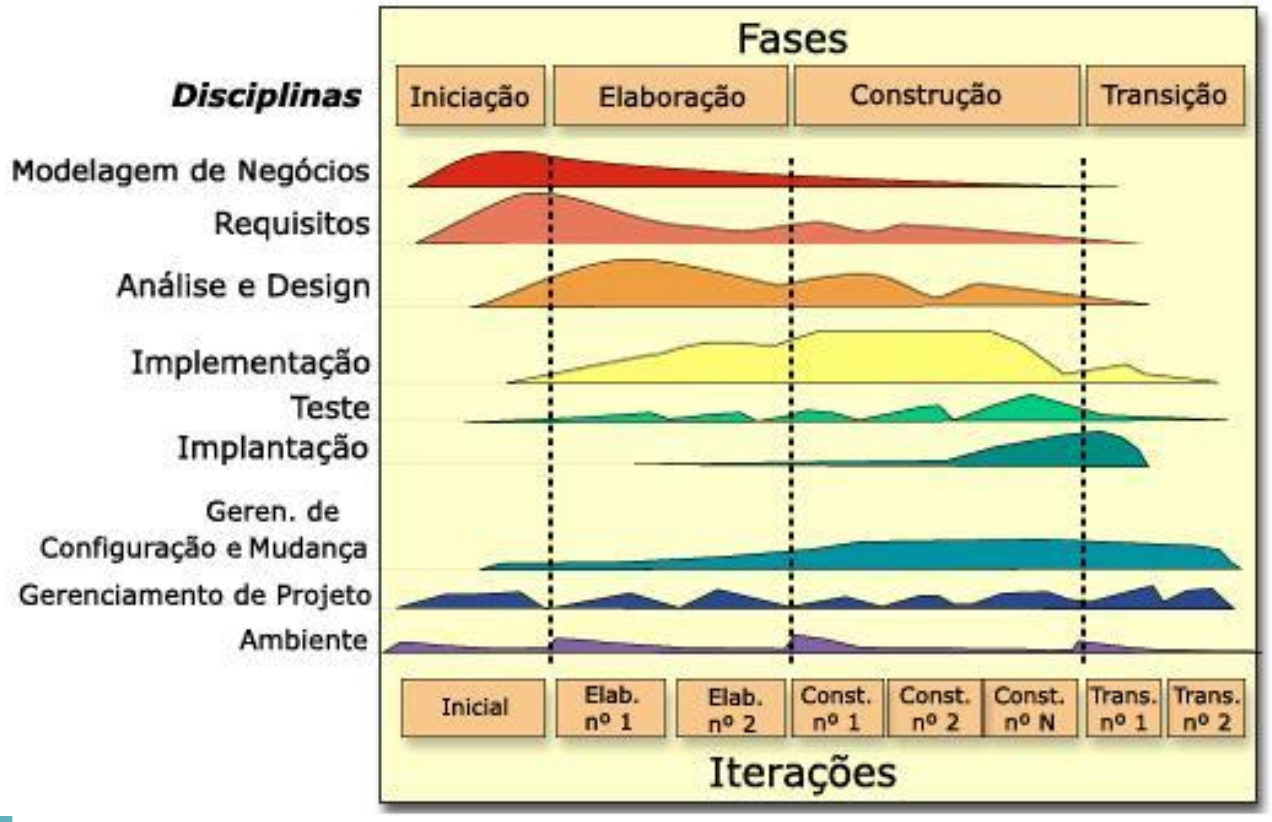
Identifique os componentes adequados → Procure componentes na biblioteca → Use os componentes disponíveis → Construa os que não existem → Adicione a biblioteca → Construa a interação

O Processo Unificado



- **Apoia o desenvolvimento de Software Orientado a Objetos.**
- **É iterativo e incremental, divido em ciclos de desenvolvimento que se repetem e cada ciclo produz uma versão funcional do software;**
- **Combinação das melhores características dos diversos processos**;
- **UML nasce desce processo**
- **Fases**: Concepção, Elaboração, Construção, Transição, Produção.
- **Permite que o desenvolvedor e o cliente interajam e revisem o software a cada etapa**.

RUP – IBM



- Mais relacionadas ao negócio que aspectos técnicos;
- Cada fase deve ser desenvolvida de forma iterativa (várias 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

COM O PASSAR DOS ANOS O DESENVOLVIMENTO DE SOFTWARE ESTÁ MAIS RÁPIDO, MAIS BARATO E COM MENOR 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)

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 feita 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 (tecnologia de rádio frequência).
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 mas está bem definido** 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

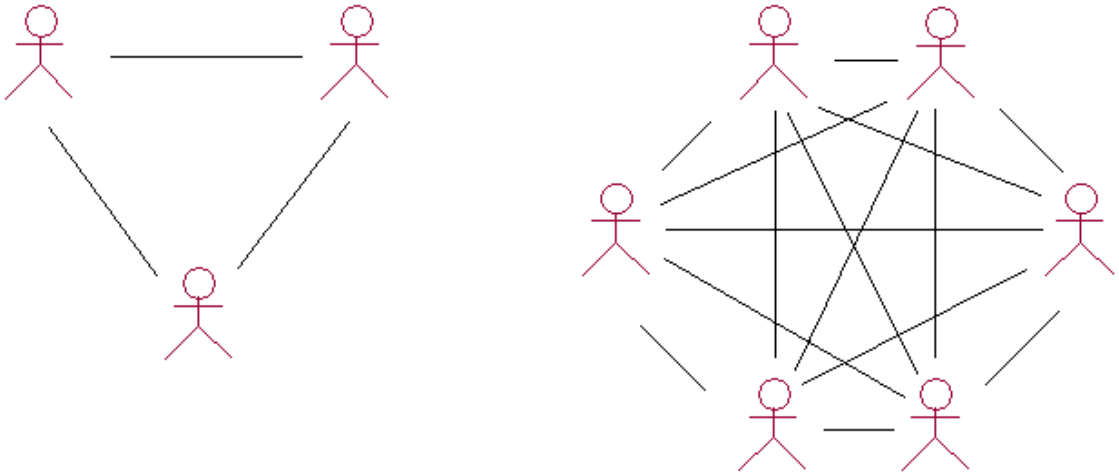
Modelo de resposta meramente ilustrativo, as respostas não tem relação com o caso
ISSO É UM CONTRA-EXEMPLO, OU SEJA, O QUE NÃO SE DEVE FAZER!

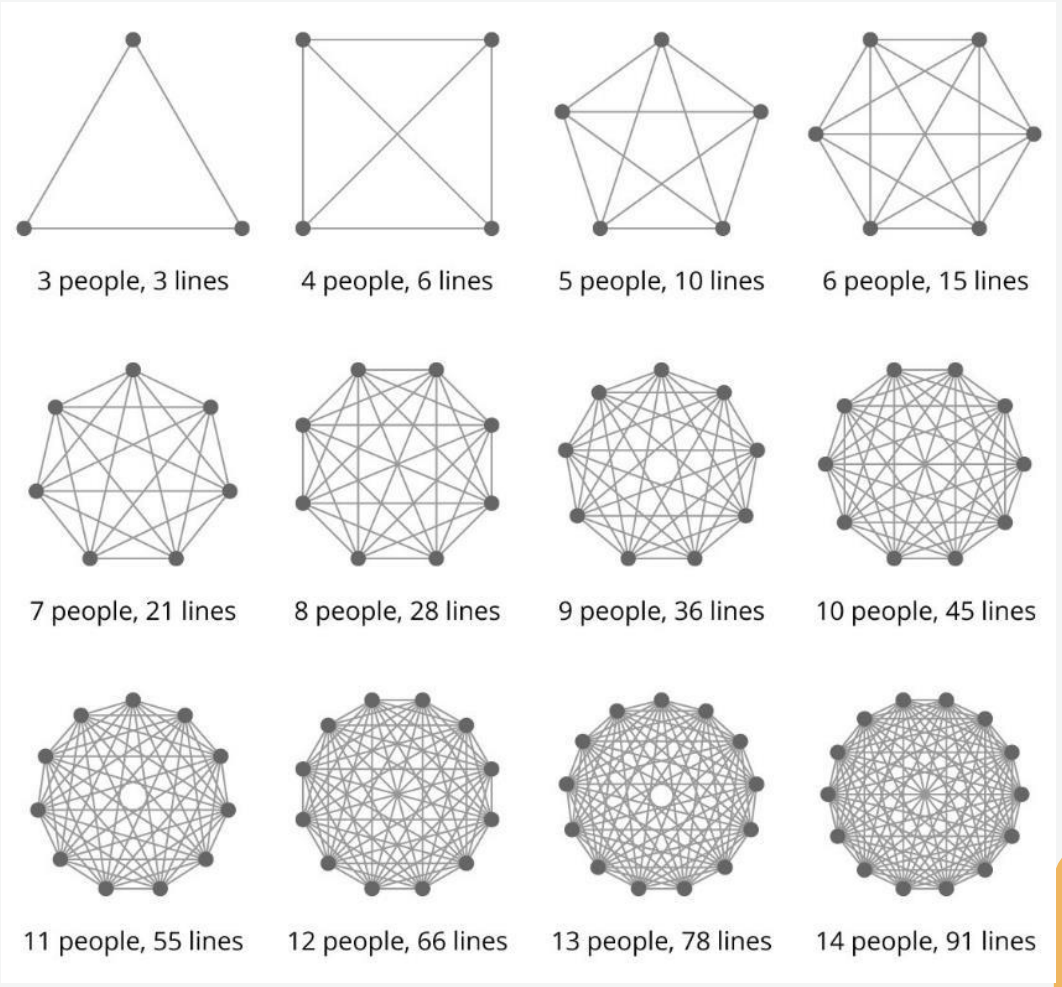
Abordagem: Codifica-remenda + Cascata
Justificativa: O sistema não é crítico e os modelos acima citados são complementares. Com este modelo daremos liberdade para o desenvolvedor agir conforme seu bom senso e teremos uma equipe integrada e feliz.

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.

Motivações para o Agile

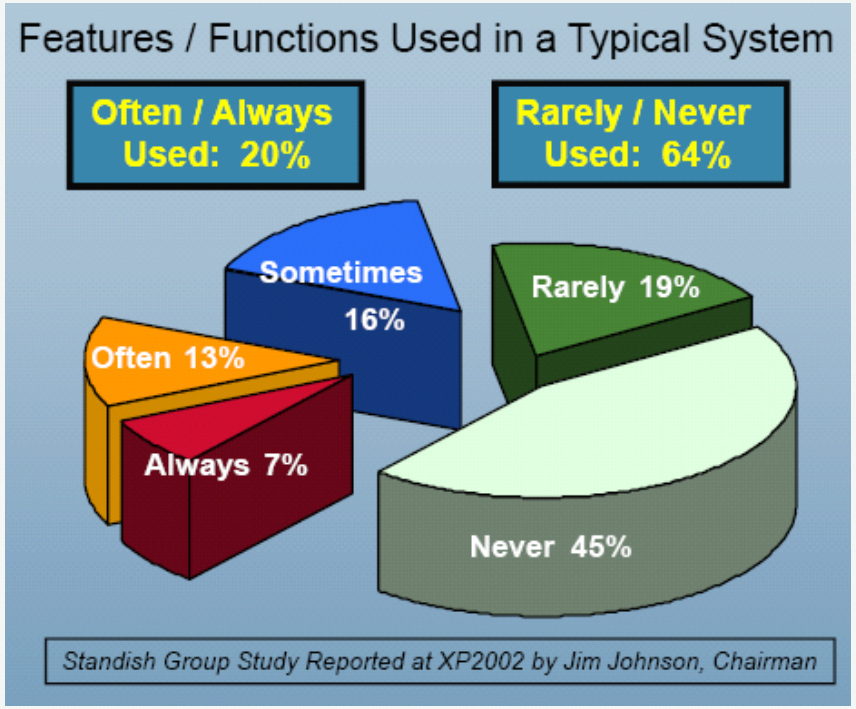




https://www.leadingagile.com/2018/02/applying-brooks-law/

A comunicação fica muito complexa em equipes grandes.

Motivações para o Agile



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

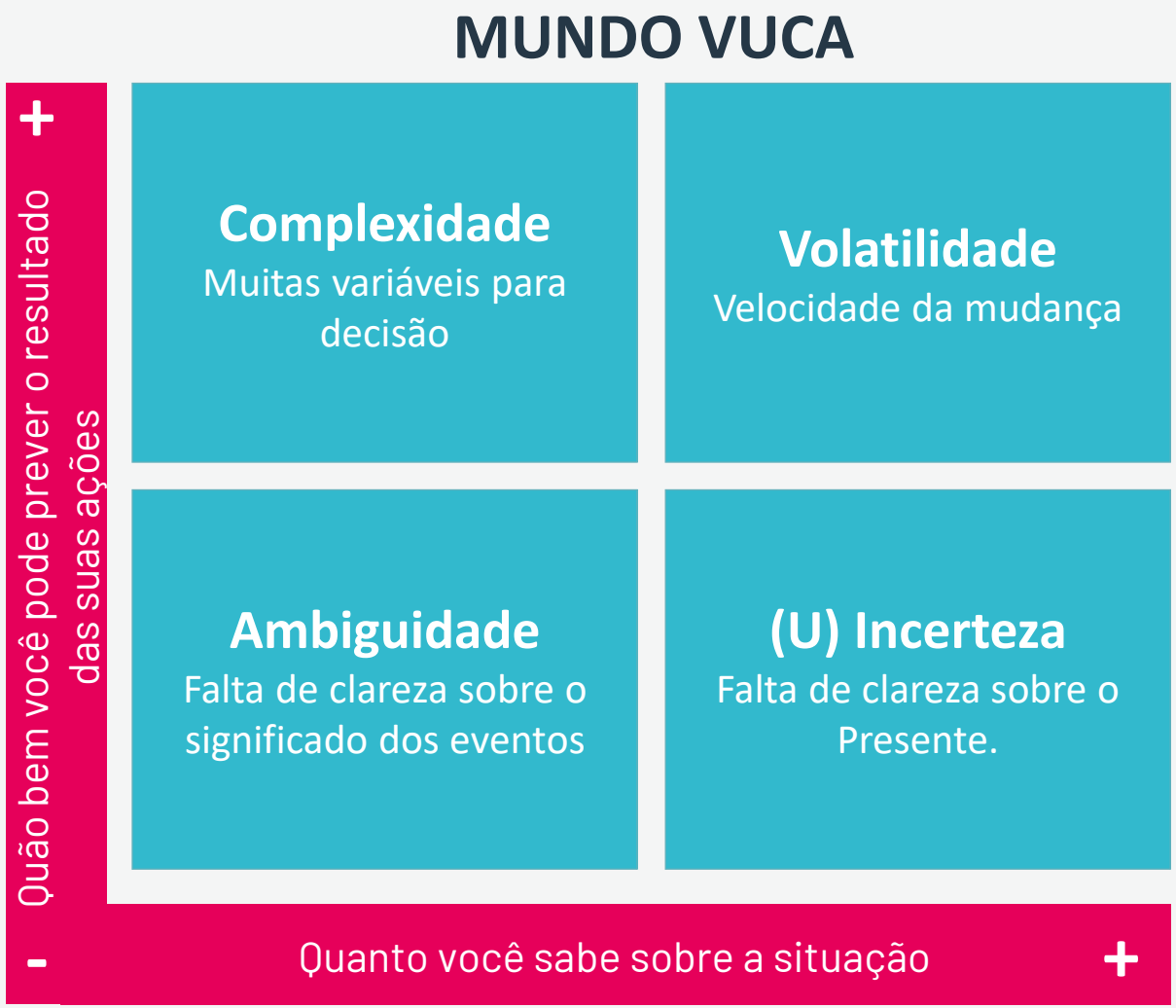
https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

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

> MUNDO VUCA

MUNDO INCERTO

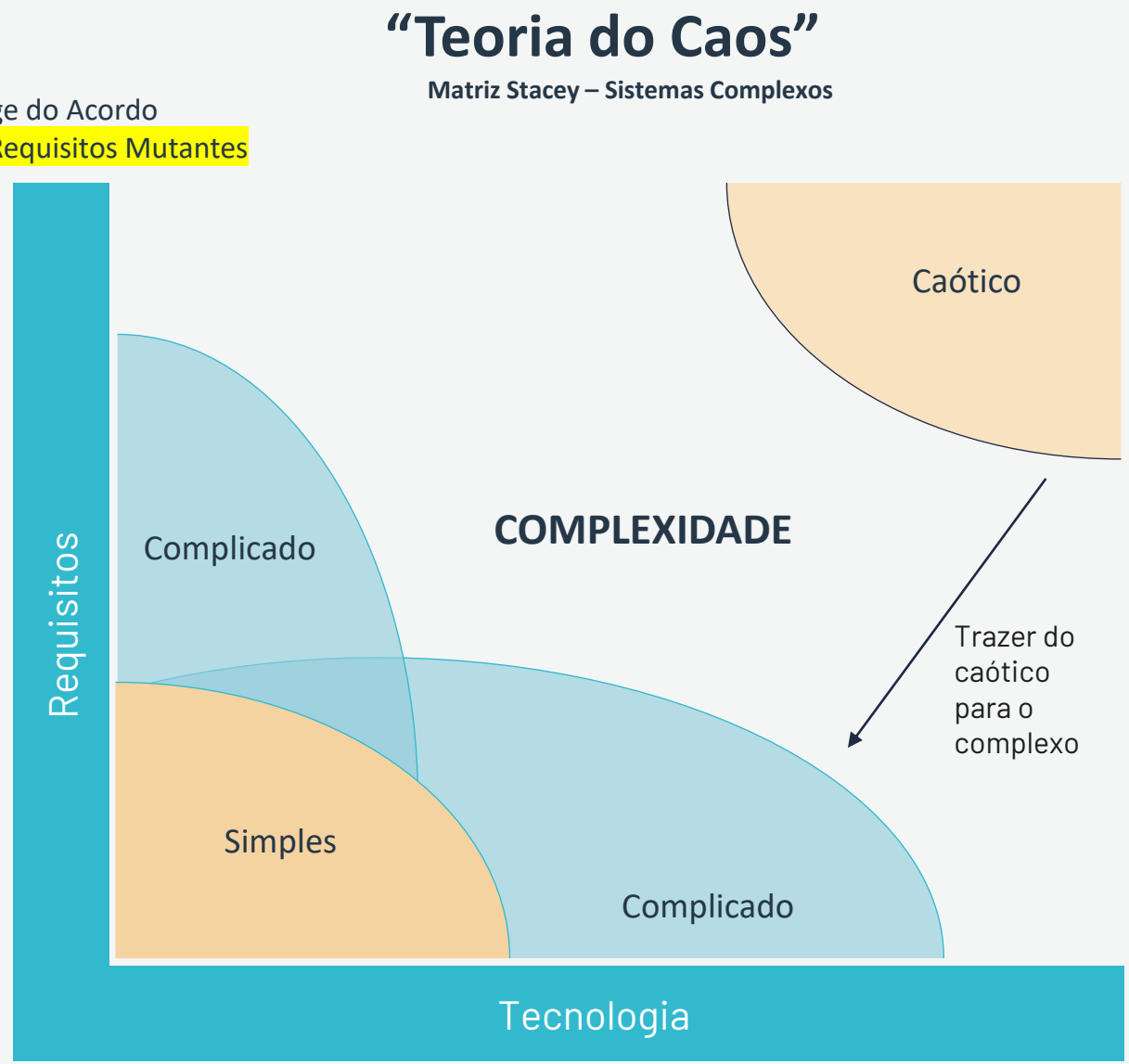
Motivações para o Agile



- 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.

Quanto menos você consegue prever, e menos conhece sobre a situação, mais difícil será lidar com situações adversas.

Motivações para o Agile



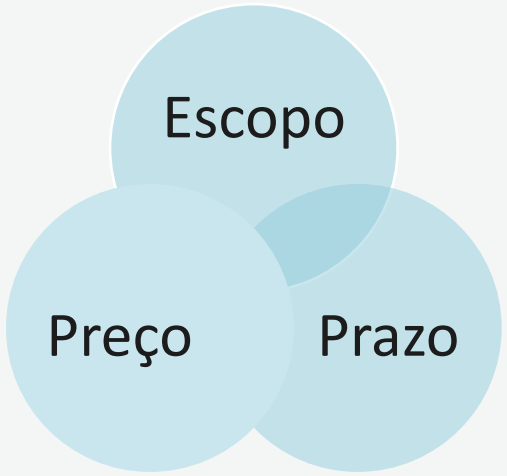
Tudo muda a toda hora, e eu não conheço a tecnologia = CAOS

Trazer do caos para o complexo ajuda a controlar.

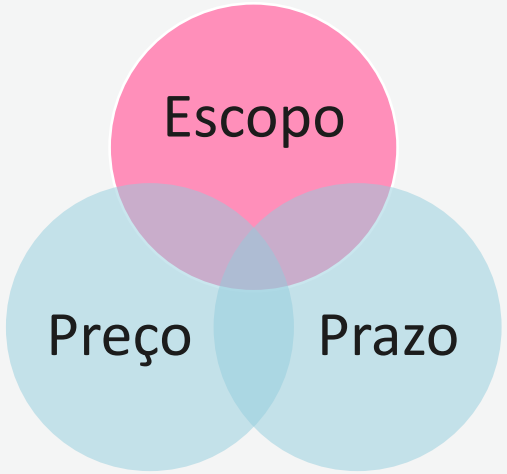
- Desenhado para ajudar a entender os fatores que contribuem para a complexidade dos sistemas.
- A relação de Incerteza versus falta de um acordo aumenta a complexidade.
- Em ambientes caóticos as abordagens são pouco eficientes, mas ainda sim, melhor com elas. Ex: Kanban.

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

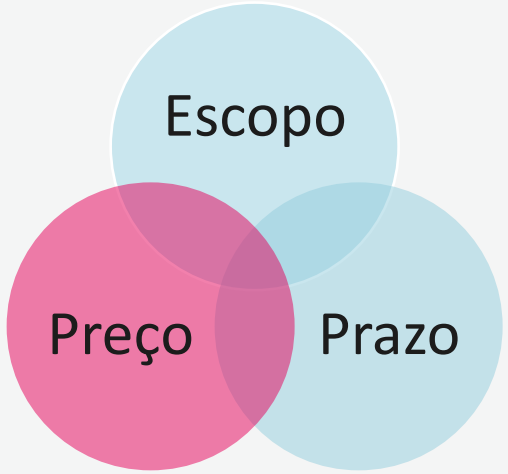
Teoria das Restrições - PMBOK



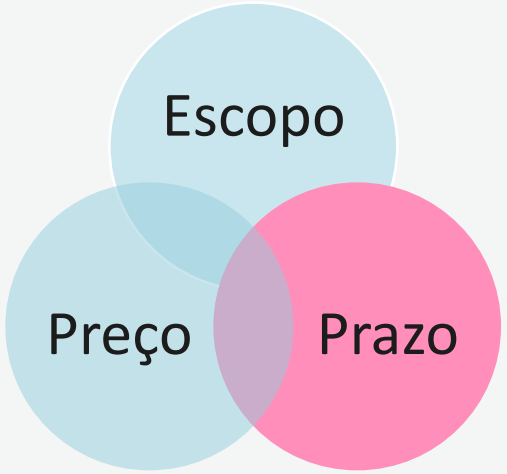
Projeto onde tudo pode!
Nem em Conto de Fadas



Projeto Crítico
Escopo não pode alterar



Restrições de Orçamento



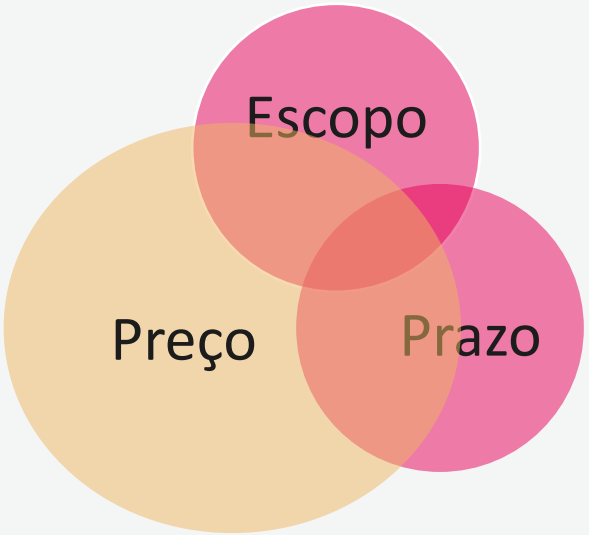
Projeto Urgente
(todos são ☺)

Não pode mexer.

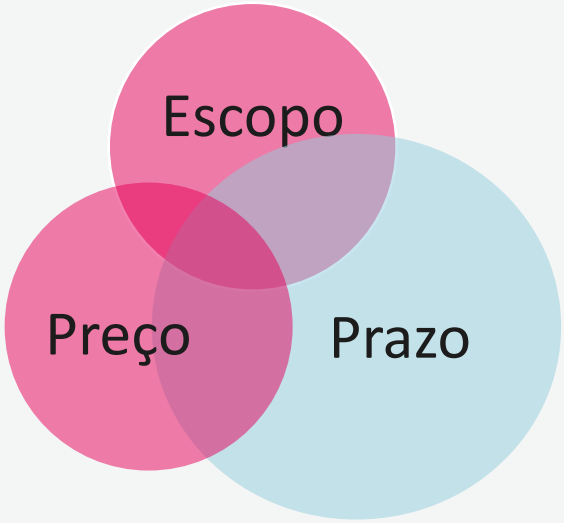
Estes são os cenários mais simples...

Teoria das Restrições - PMBOK

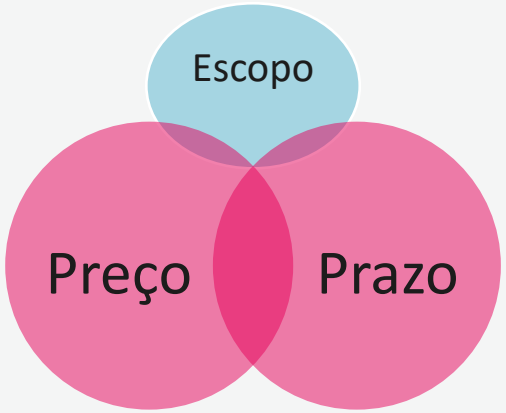
...Agora sim, a vida como ela é...



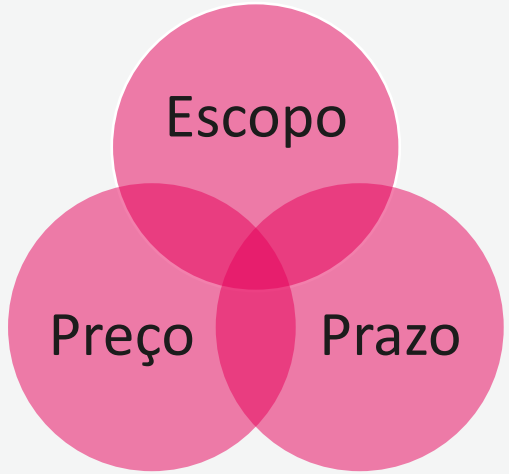
Projeto Crítico e Urgente
(Precisa de mais $)



Projeto Crítico e com Pouco $
(Aumenta o Prazo)



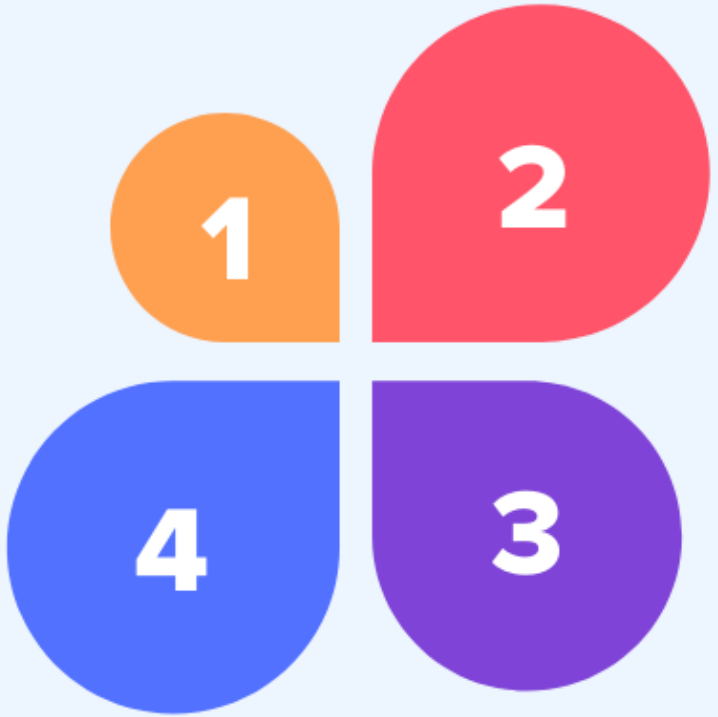
Pouco $ e Urgente
(Reduz funcionalidade)



Falácia - Caos
(Coloque razão na situação)

Manifesto Ágil

Indivíduos e interações
mais que processos e ferramentas



Software em funcionamento
mais que documentação abrangente

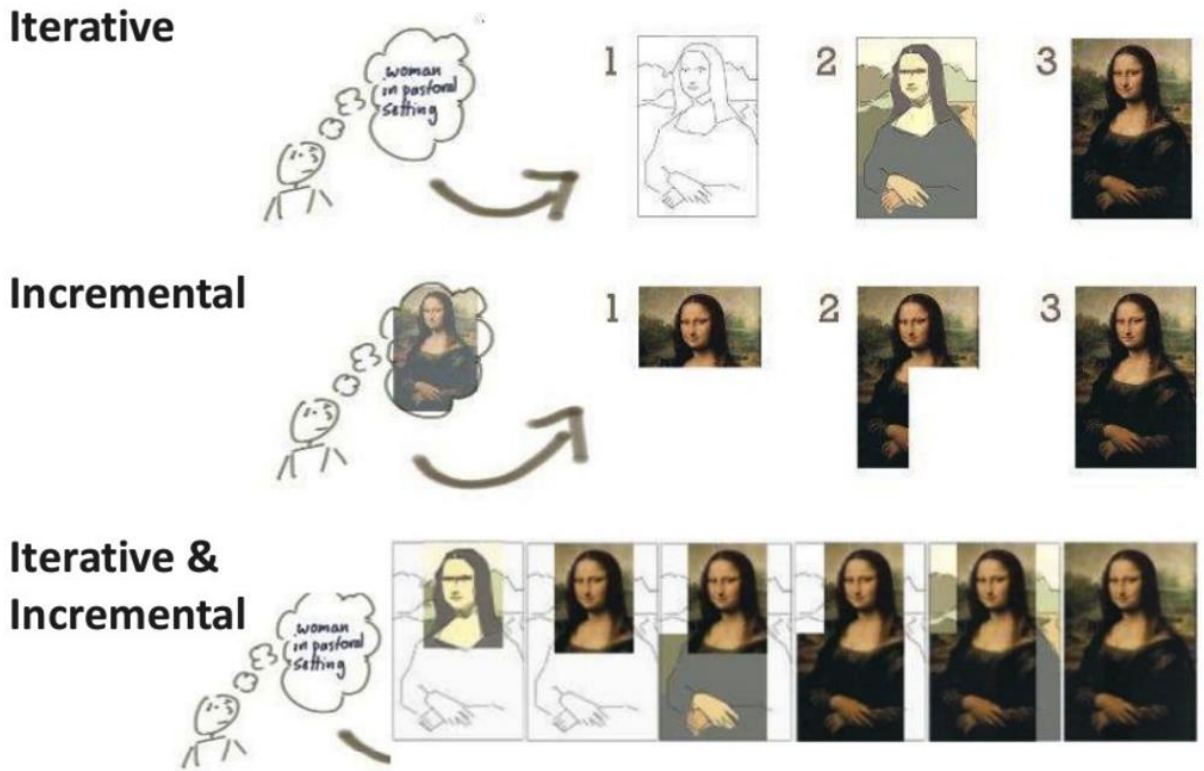
Responder a mudanças
mais que seguir um plano

Colaboração com o cliente
mais que negociação de contratos

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

Agile

Combinação do Iterativo (de iterar, repetir) + Incremental



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

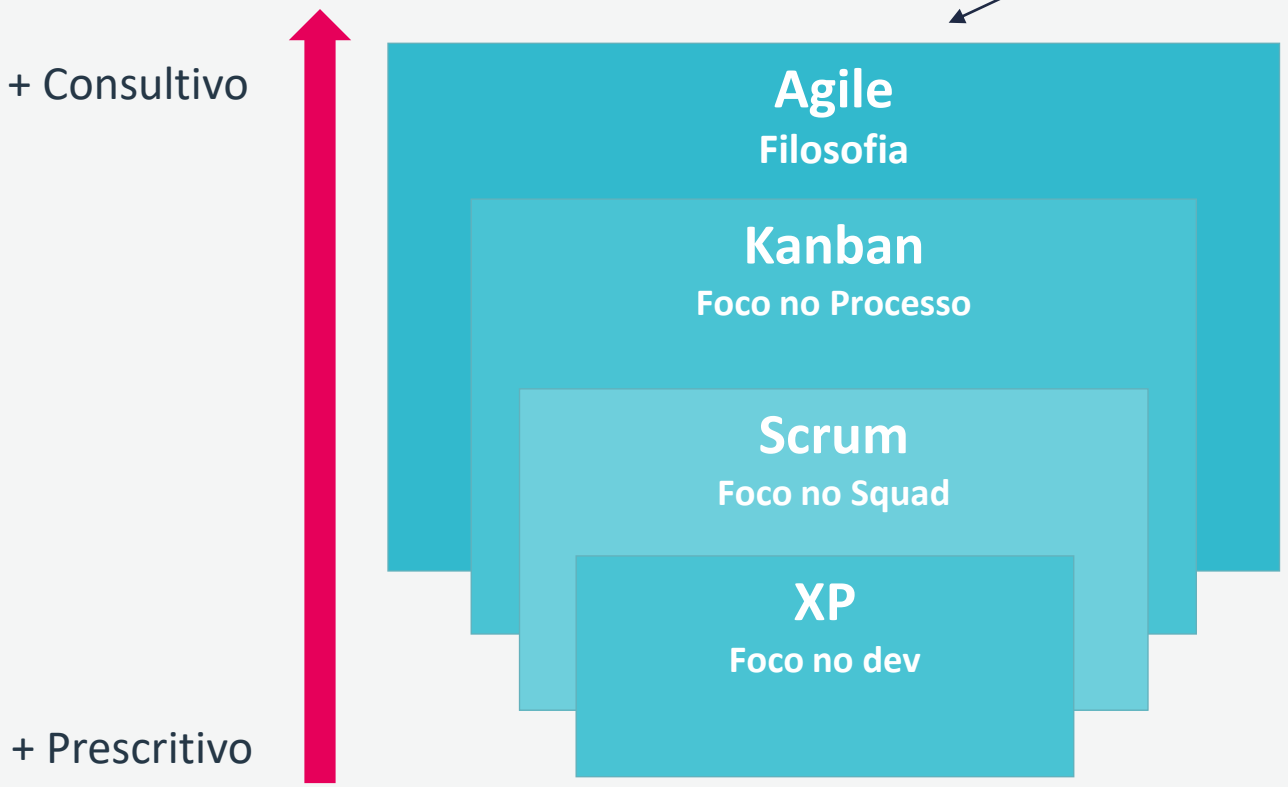
- O que entregamos no Agile deveria ser utilizável;
- Há equipes ou empresas que trabalham apenas com o incremental; (Faseamento).
- Também existem projetos que trabalham apenas de forma iterativa. (Protótipo).

Tem que conseguir utilizar!

Agile - Filosofia

Iterativo (de iterar, repetir) + Incremental

Agile é consultivo: É bom que as pessoas interajam...



- 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.

XP é prescritivo: Faça isso..., faça aquilo...

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

XP – Programação Extrema (Foco no DEV)

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·

Teste de Aceitação
Os testes são construídos pelo cliente·

Desenvolvimento Orientado a Testes (TDD)

Testes automatizados!
Testes sempre!

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

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

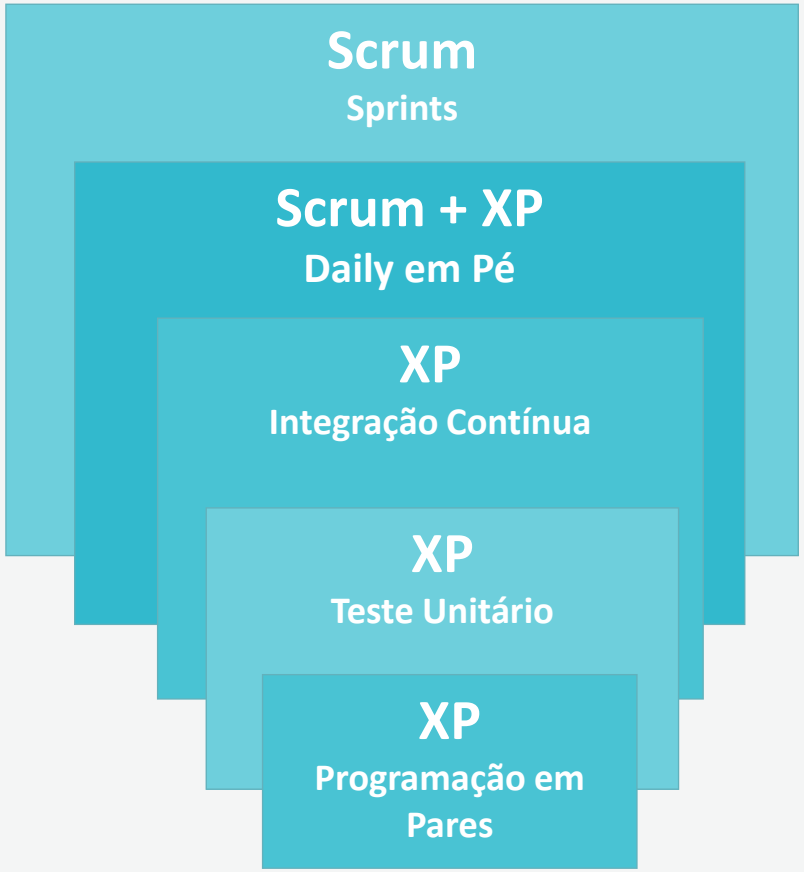
Padrão de Codificação
A equipe de devs precisa estabelecer regras para programar e todos devem seguir estas regras·

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

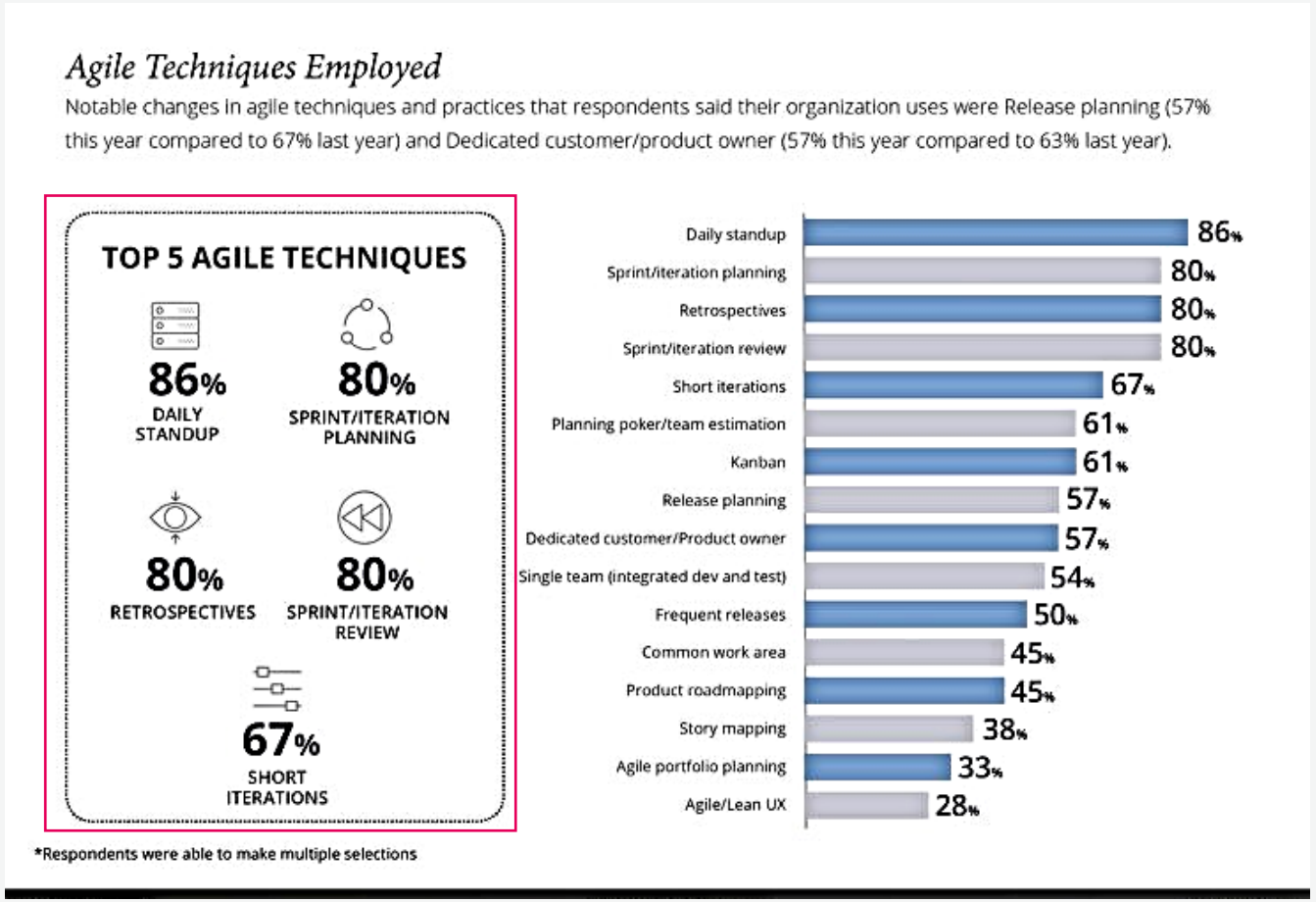
Time Coeso (harmonia)
Comunicação!
Usar o socioemocional

SCRUM + XP

As empresas combinam as práticas ágeis.



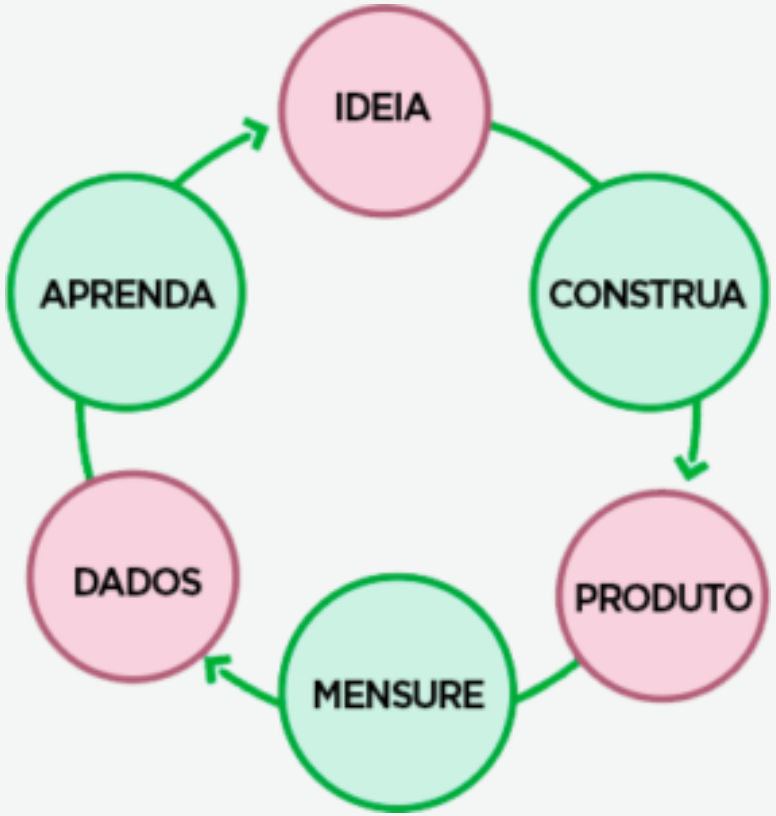
+ XP

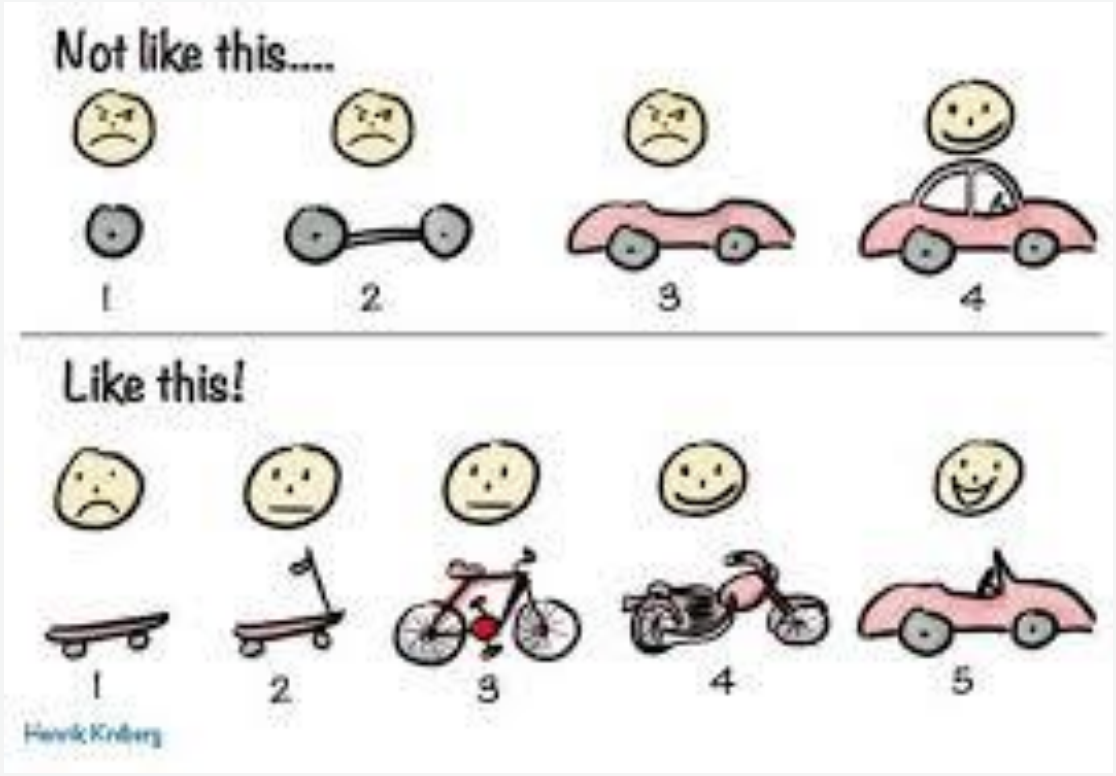


https://www.stateofagile.com/#ufh-i-521251909-13th-annual-state-of-agile-report/473508

Abordagens Modernas - MVP

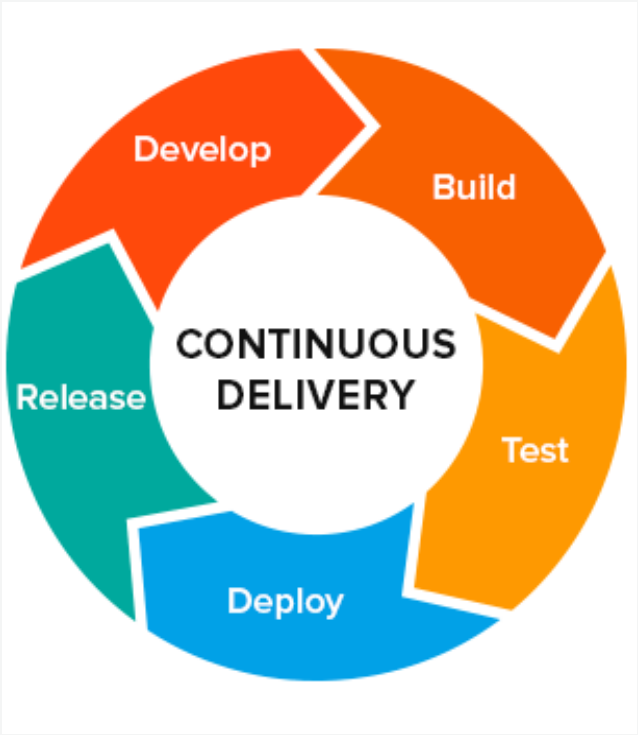
Mínimo Produto Viável. Criação de novos negócios de forma ágil.





Se você já sabe muito bem o que vai fazer ou é uma nova demanda que é extensão do produto atual, por que razão 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**.

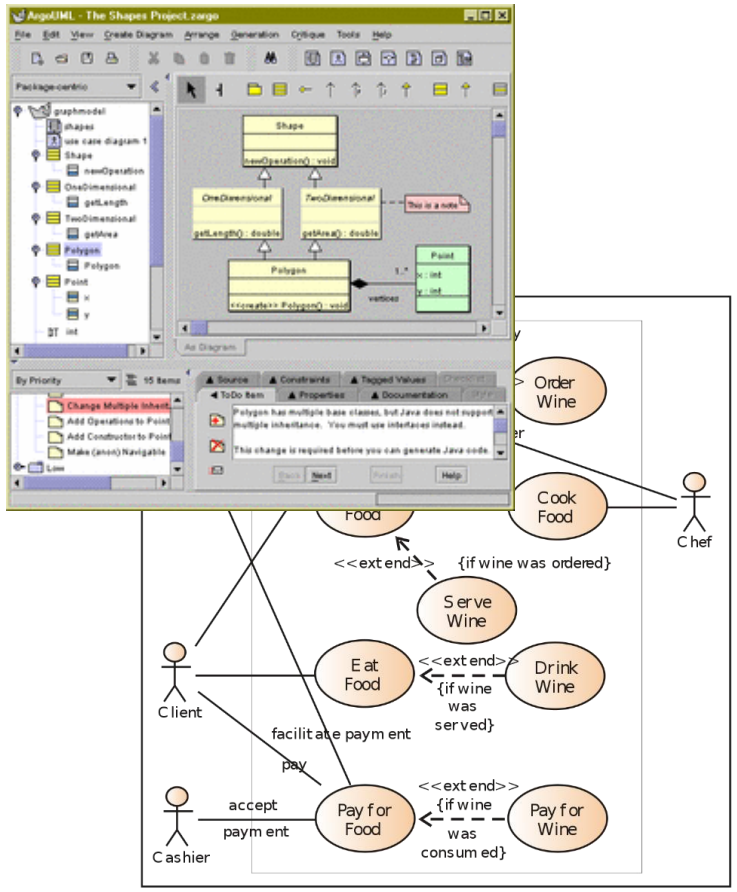
Parece com alguma abordagem até aqui?
Novamente...nada se cria do zero....

Tradicional x Ágil

https://www.agile-minds.com/when-to-use-waterfall-when-agile/

Ferramentas CASE

- Engenharia de Software Auxiliada por Computador (CASE – *Computer-Aided Software Engineering*)



- É o nome dado ao **software utilizado para apoiar as atividades de processo de software: engenharia de requisitos, processo, projeto, desenvolvimento, teste, etc**.
- As **ferramentas CASE automatizam as atividades rotineiras**, reservando mais tempo para as atividades criativas;
- **Ajuda MUITO na interação das equipes**;
- **HOJE não existe time de ALTA PERFORMANCE que não use estas ferramentas**;

Exemplos de Ferramentas Case

Classificação Funcional das Ferramentas		Exemplos
Planejamento	PERT, Planilhas, Gerenciamento de Projeto	Project, Primavera, dotProject, Excel
Edição	Editores de Texto, Editores de diagramas	Word, Argo UML, Visio
Gerenciamento de Mudanças	Controle de requisitos, sistemas de controle de mudanças	Jira, Trello
Gerenciamento de Configuração	Gerenciamento de versões, construção de sistemas	GIT, SVN,
Prototipação	Geradores de interface com o usuário	AppInventor, Balsamic Mockup, Powerpoint, Figma
Apoio a métodos	Dicionário de dados, geradores de código	Erwin, Eclipse, Visio
Processamento de linguagens	Compiladores, interpretadores	JDK, .NET Framework, DBs em Geral, Apache Ant
Análise de Programa	Analisadores estáticos e dinâmicos	
Testes	Comparadores de arquivos, automação de testes	Junit, jmeter, soapui, scripts em shell
Depuração	Sistemas de depuração interativos	Debuggers em geral
Documentação	Formatação de páginas, editores de imagens	Javadoc, wiki
Métrica	Estimar curso e esforço	USC-COCOMO, CAST Software

Agradeço

a sua atenção!

Fábio Figueredo

fabio.figueredo@sptech.school

SÃO PAULO TECH SCHOOL