



SÃO
PAULO
TECH
SCHOOL

Engenharia de Software

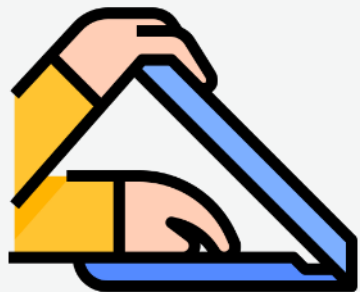
Processos de Software

Aulas 13 e 14

Gerson Santos

gerson.santos@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.
- **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

**Respeito**

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.**

Avaliações

Socioemocional: Binária (reprova ou aprova). *Feedbacks durante o semestre.*

Pesquisa e Inovação: Binária (reprova ou aprova). *Feedback no final de cada Sprint.*

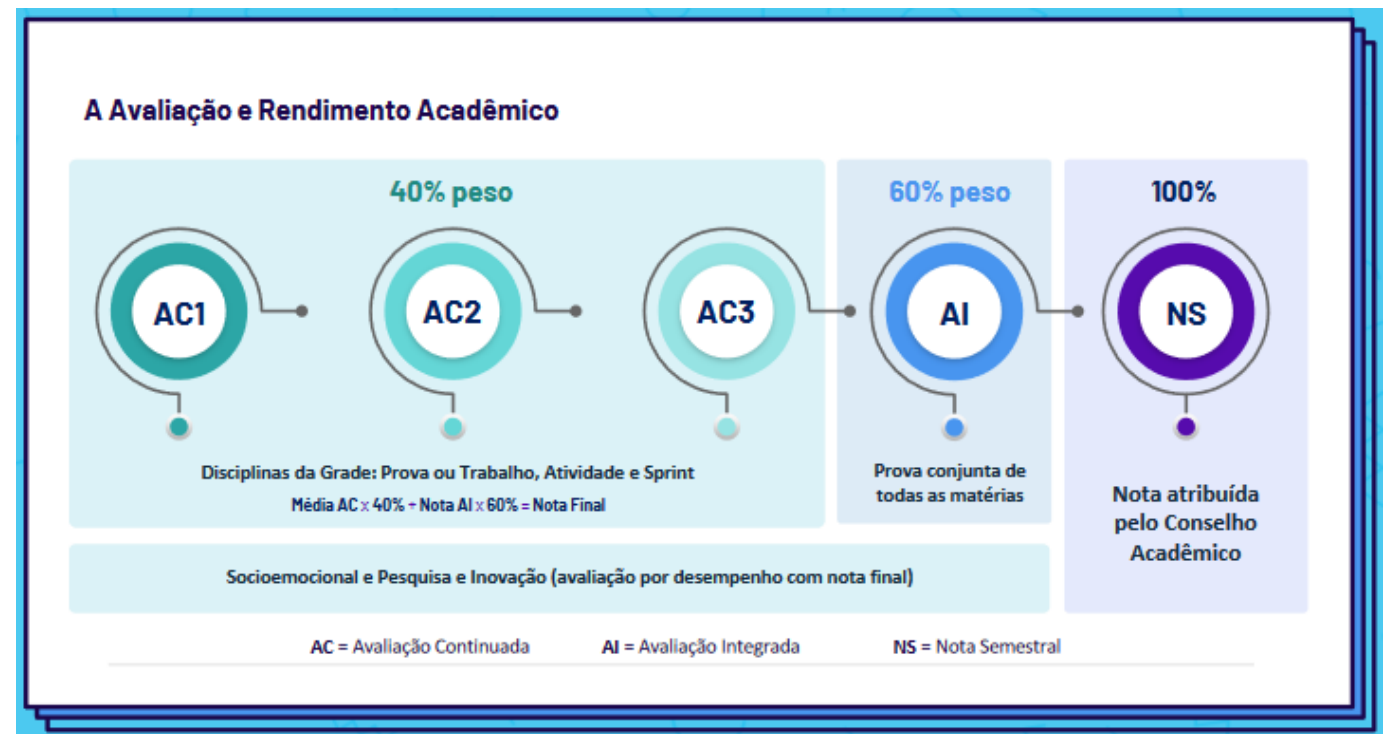
Média = 6

40% da Nota – Continuadas

1 Continuada por Sprint com possibilidade de inspeção Individual (São 3 continuadas)

60% da Nota

*1º Semestre = Projeto Individual ;
Demais semestres: Avaliação Integrada*



Manual do Aluno

Nosso Caminho

Legenda: Conteúdo / **Entregável PI** / Onde Estamos



S3

Final do Semestre

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

- **Planilha de UAT**
- **Testes Unitários – Junit**
- **Diagrama de Solução (Componente)**
- **Projeto Web aderente aos conceitos de UI e UX**

Continuada: 26/05/2022 / Sprint: 30/05/2022

S2

- ~~Design de Interfaces Web~~
- ~~Projeto de Software~~
- ~~Arquitetura de Software~~

- ~~**Desenho de Arquitetura**~~
- ~~**Protótipo em Alta Resolução**~~
- ~~**Diagrama de Solução de Software – Container**~~
- ~~**Planilha de Arquitetura**~~

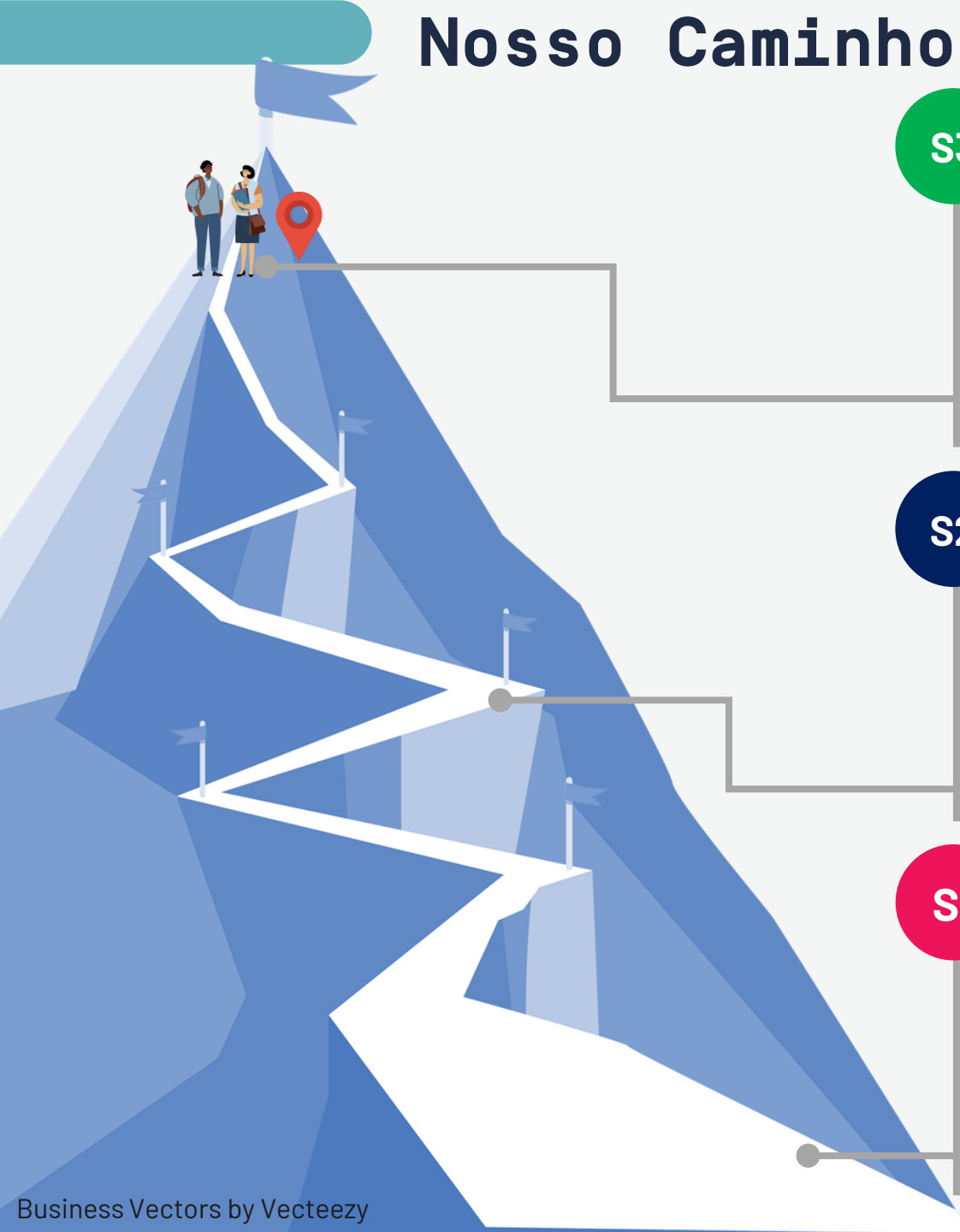
Entrega: 25/04/2022

S1

- ~~Apresentação~~
- ~~UI/UX~~
- ~~Fatores Humanos~~
- ~~Design de Interação~~
- ~~Design de Interfaces + Bootcamp~~

- ~~**Jornada do Usuário**~~
- ~~**Prototipação das Telas**~~

Entrega: 03/03/2022





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

- Modelos de Arquitetura de Software
- Qualidade



Nosso objetivo:

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



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.

The background is a dark, textured surface with a complex pattern of glowing, wavy lines and small, bright dots, resembling a digital or cosmic theme. The lines are thin and white, creating a sense of movement and depth. The dots are also white, some appearing as small stars or particles. The overall effect is a high-tech, futuristic aesthetic.

**ANTERIORMENTE EM ENGENHARIA
DE SOFTWARE...**

Qualidade

Bibliografia

Engenharia de Software 8ª Edição / Ian Sommerville

Engenharia de Software 6ª Edição / Roger Pressman



Adicional

Code Complete

SWEBOK

entre outros

Voltando ao desenho clássico



Como o cliente explicou...



Como o líder de projeto entendeu...



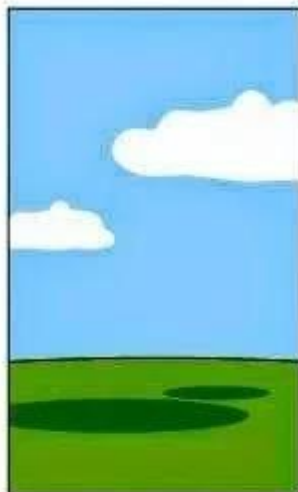
Como o analista projetou...



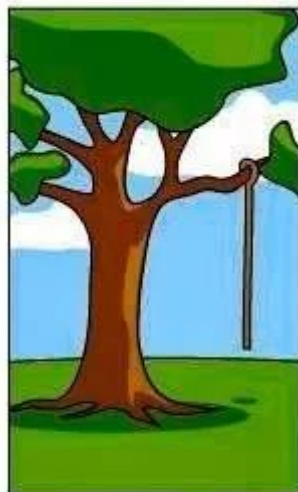
Como o programador construiu...



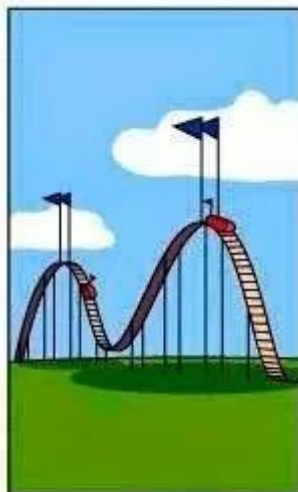
Como o Consultor de Negócios descreveu...



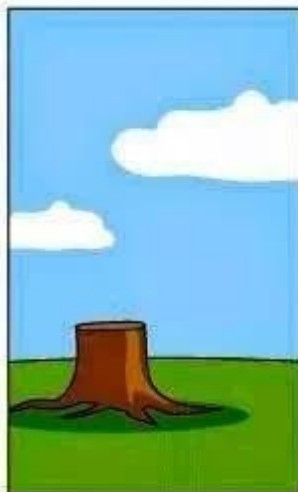
Como o projeto foi documentado...



Que funcionalidades foram instaladas...



Como o cliente foi cobrado...



Como foi mantido...



O que o cliente realmente queria...

Como medir?



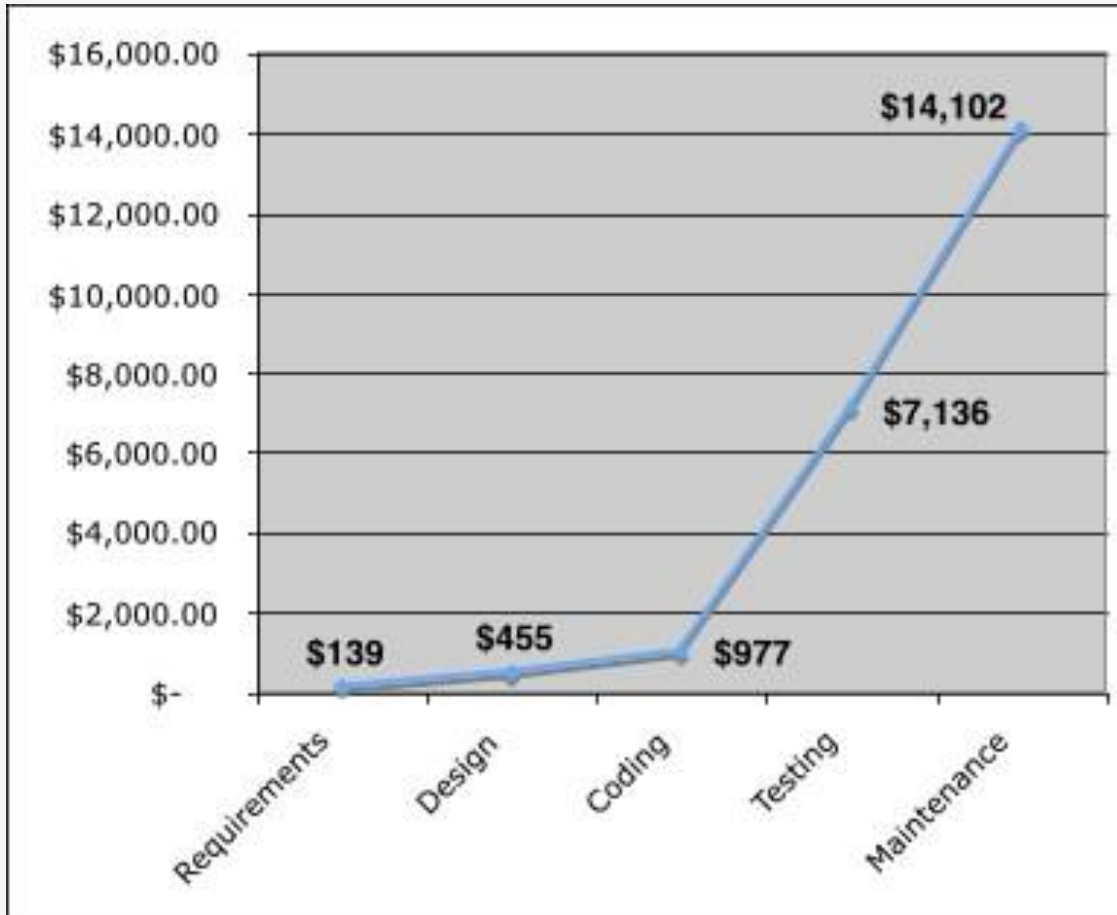
As equipes de projeto precisam desenvolver um conjunto de perguntas específicas para avaliar o grau em que cada fator de qualidade do aplicativo foi satisfeito

Medidas subjetivas de qualidade de software podem ser vistas como pouco mais que uma opinião pessoal

Métricas de software representam medidas indiretas de alguma manifestação de qualidade e tentam quantificar a avaliação da qualidade do software

Prevenir é melhor que remediar?

Quanto custa a qualidade? \$\$\$\$\$



Veja no exemplo que o custo de reparação fica extremamente alto logo depois que sai da mão do desenvolvedor.

Conseguem pensar o por quê?

Quanto custa a Qualidade?

Custo de Prevenção

- Planejamento de qualidade
- Equipamentos/Ambientes de teste
- Treinamento

Custos de Avaliação (Internos)

- Testes e Depuração
- Coleta de dados e métricas

Custo de Falha (Internos)

- Retrabalho e Correção
- Efeitos colaterais
- Coleta de Dados e Métricas
- Desgaste da Equipe

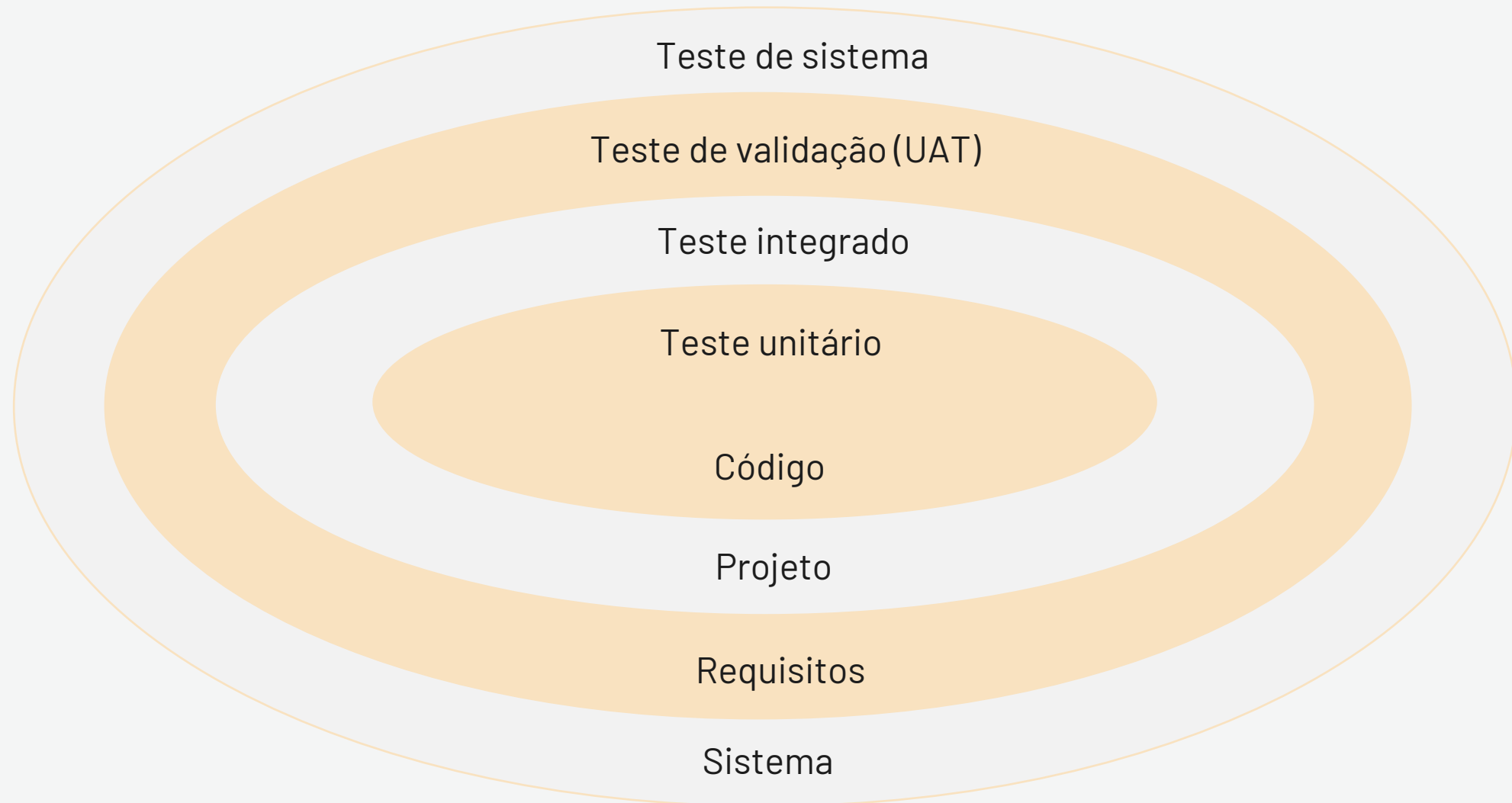
Custos Externos

- Resolução de reclamações
- Retorno e substituição do produto
- Suporte (SAC)
- Reputação
- Satisfação do Cliente
- Responsabilidade Civil

https://www.google.com/url?sa=i&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwi6gODV6fXhAhXsGLkGHabKD8oQjRx6BAgBEAU&url=http%3A%2F%2Fwww.folhavoria.com.br%2Feconomia%2Fblogs%2Fgestaoresultados%2F2012%2F02%2F15%2Fcustos-da-qualidade-1a-etapa%2F&psig=A0vVaw2Ha92fZgovRT_oLGGyRDPC&ust=1556645269822925

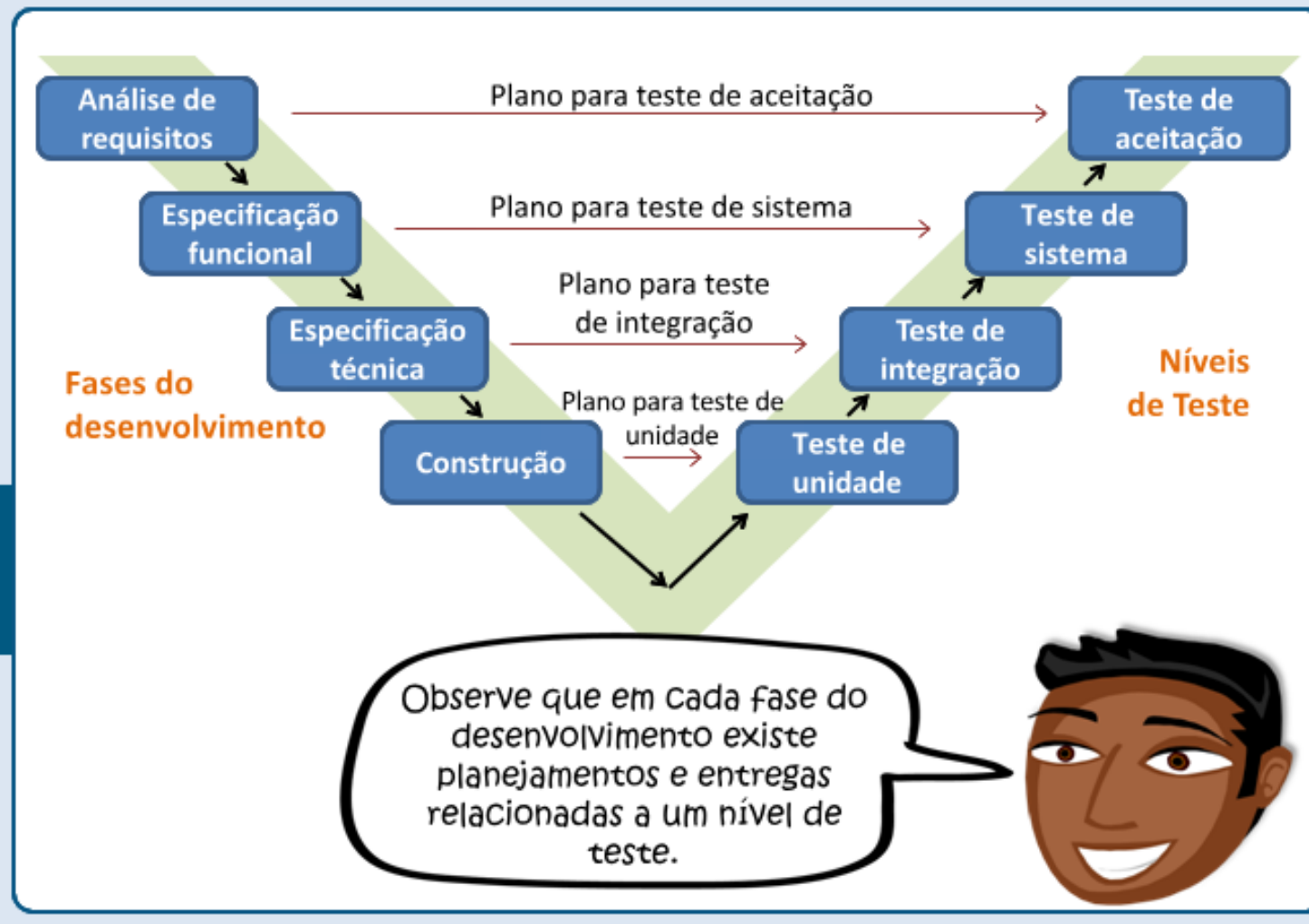


Níveis de Teste/Estratégia de Teste



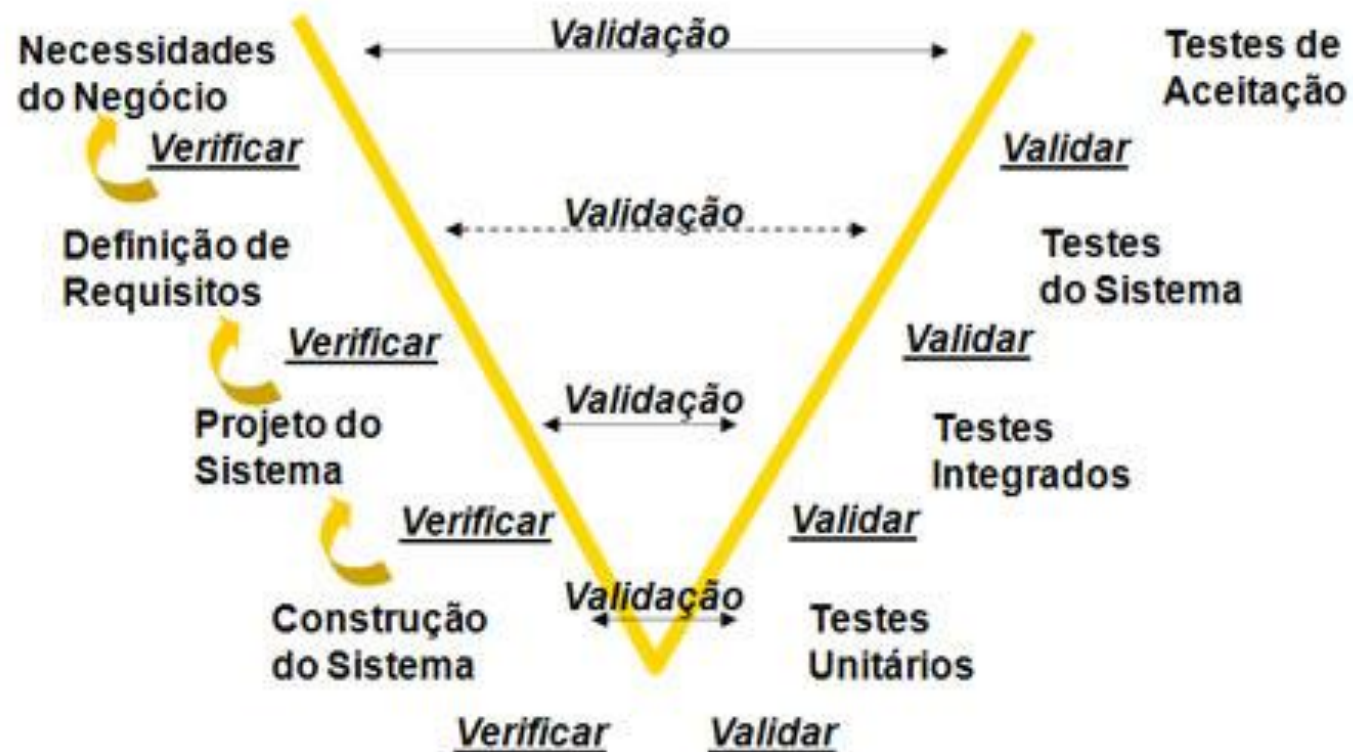
Níveis de Teste/Estratégia de Teste

Teste no ciclo de vida



Modelo IV&V

Cobertura Total



Tipos de Testes

Testes de Aplicações Web [Mobile é similar]

- **Conteúdo**
- **Banco de Dados**
- **Interface do usuário**
- **Usabilidade**
- **Compatibilidade**
- **Componentes**
- **Navegação**
- **Configuração**
- **Segurança**
- **Performance**
- **Carga**
- **Stress**



<https://paulomonteiro.wordpress.com/cansado/>

Teste integrado

A maioria dos testes que vimos, são testes integrados.

Entender se as partes somadas funcionam corretamente. Caso não seja feita o teste unitário, a complexidade/volume de erros nesta fase pode aumentar muito:

A forma de construção da Aplicação afeta a forma de realizar o teste integrado.

Formas:

- Big Bang
- Incremental

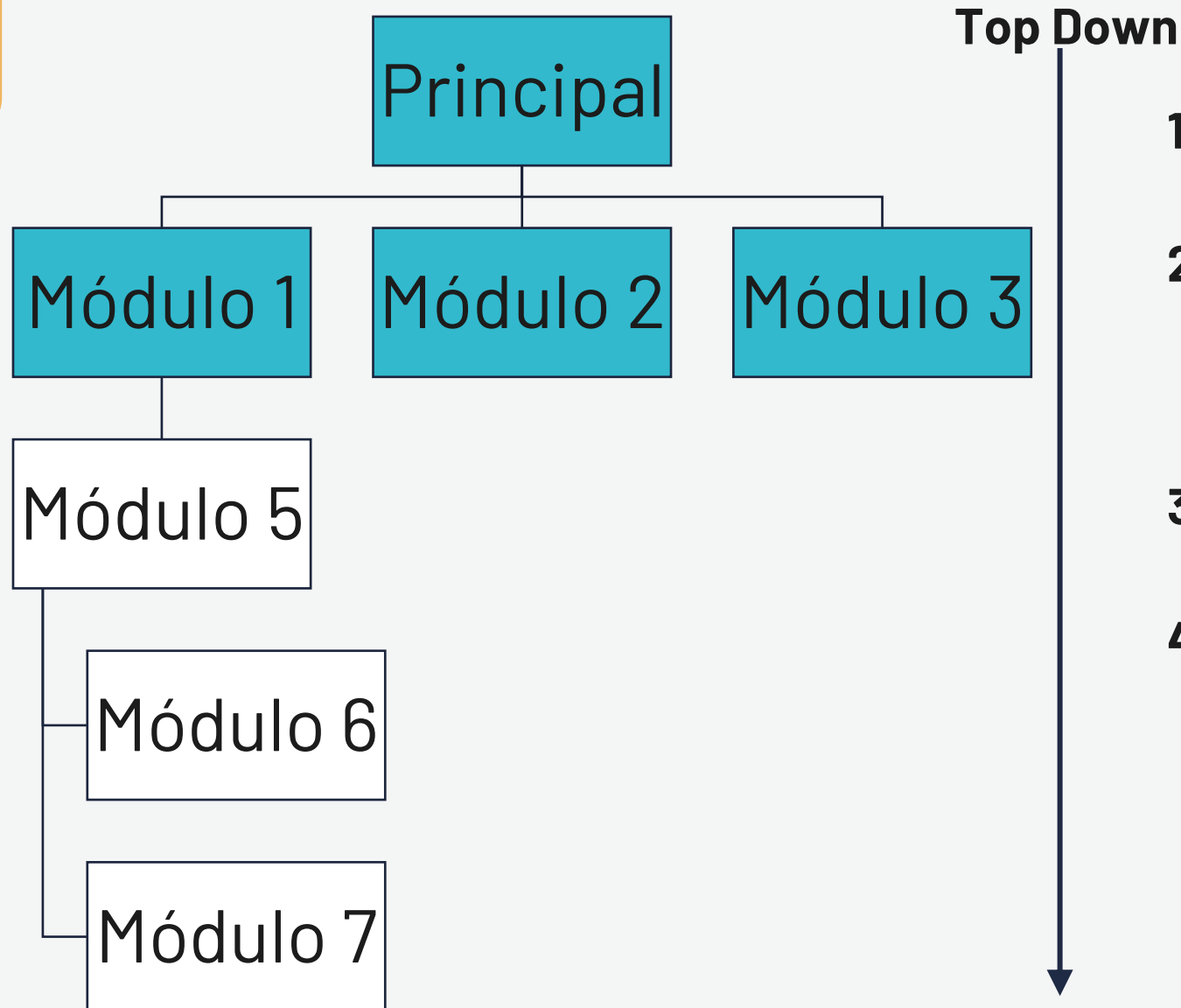
Teste integrado: Big Bang

- Desenvolve toda a aplicação e depois começa a testar!
- Normalmente se testam as partes isoladas
- Por mais estranho que pareça, é bem comum!



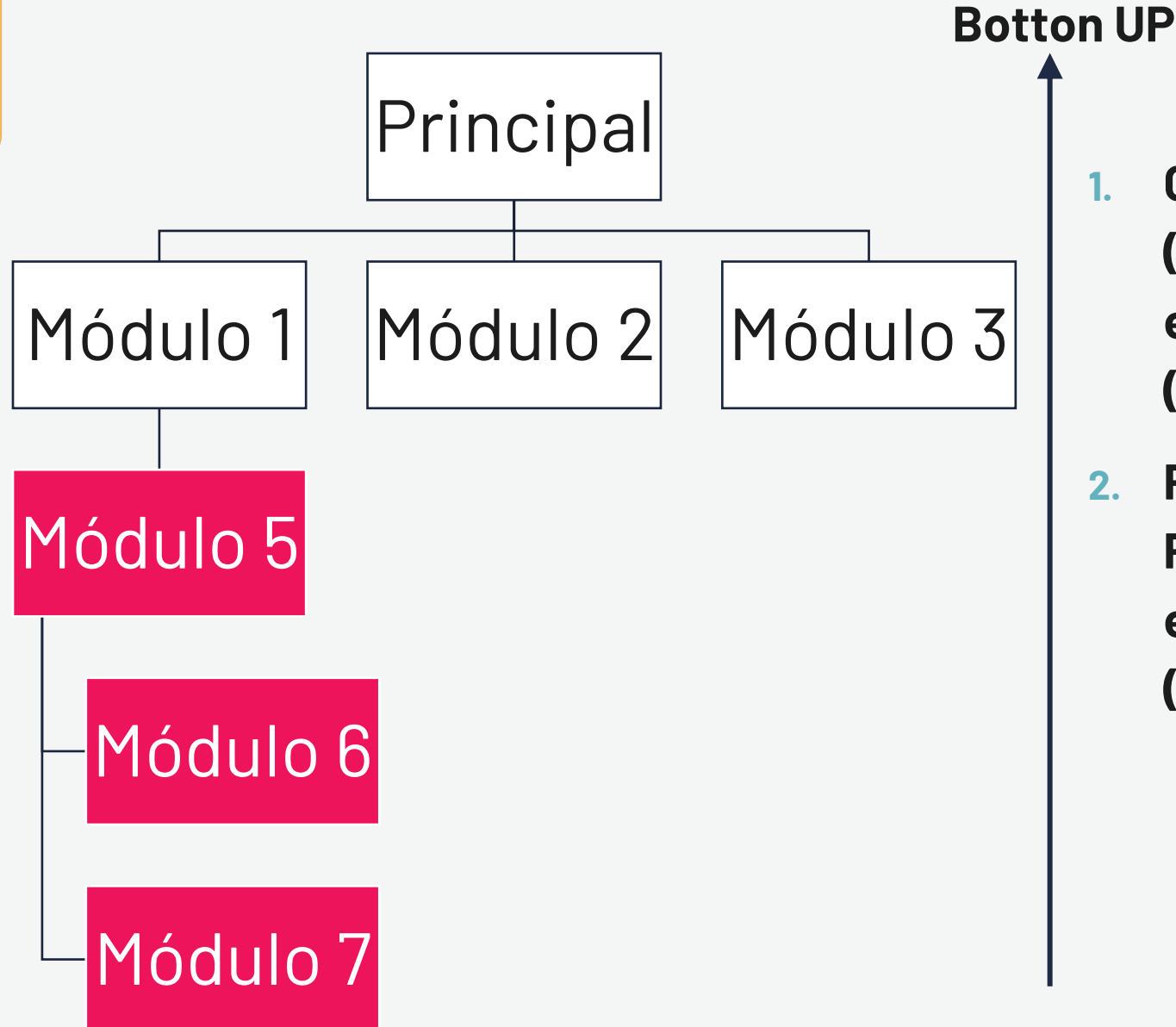
[Esta Foto](#) de Autor Desconhecido está licenciado em [CC BY-NC-ND](#)

Teste integrado: Integração descendente



1. Módulo de Controle Principal é utilizado como um testador
2. Os módulos podem ser substituídos por módulos pseudocontroladores dependendo da abordagem (stubs/mocks)
3. Os testes são realizados a medida que cada módulo é integrado.
4. Quando um novo módulo é integrado um conjunto de testes é realizado novamente (pode ser regressão)

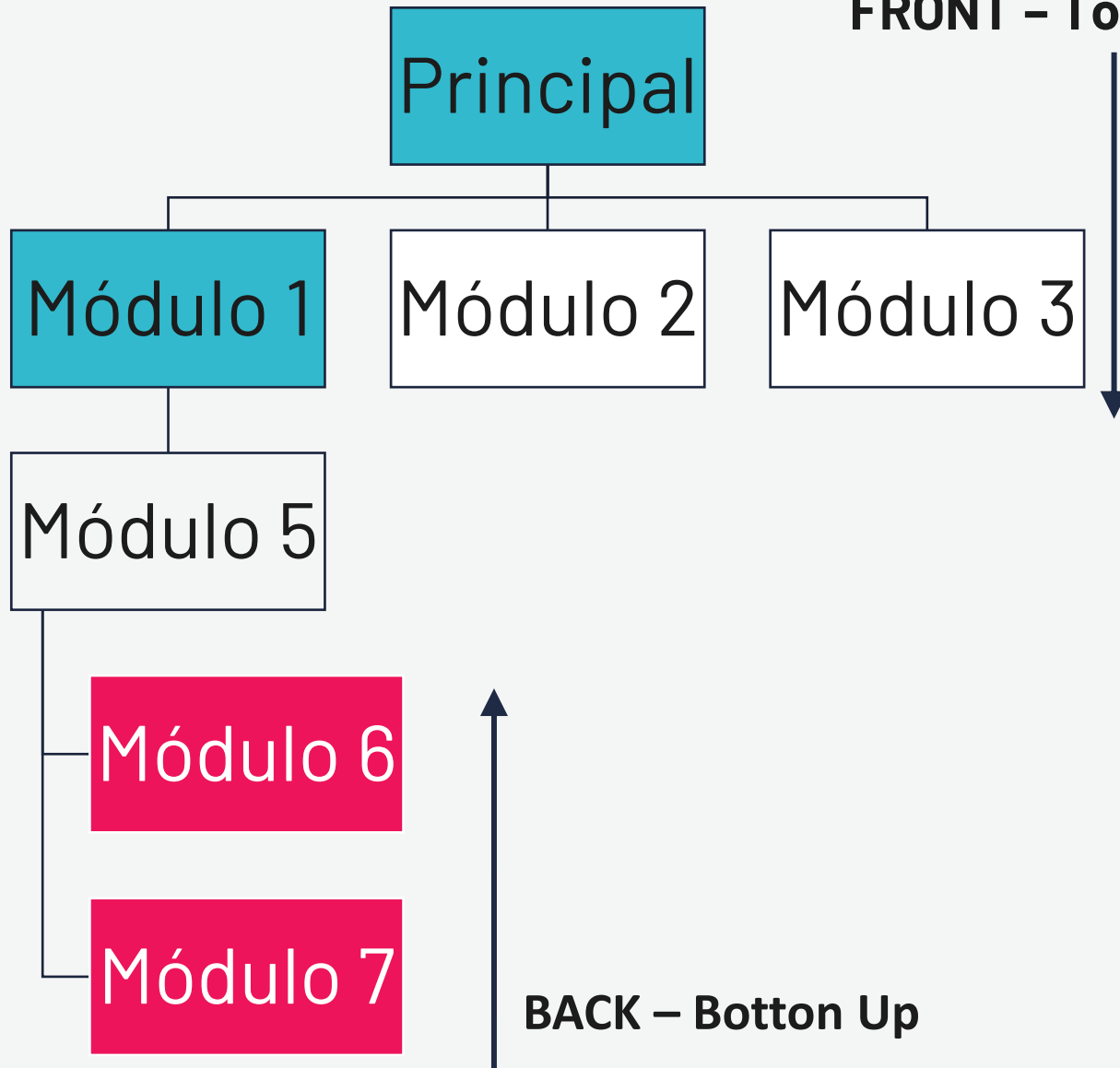
Teste integrado: Integração ascendente



1. Os Módulos de baixo nível são agrupados (builds) conforme a função que executam, ou seja, dependências. (Exemplo em azul)
2. Precisa ser escrito um PseudoControlador para coordenar a entrada e saída de teste. (drivers/mocks)

Teste Integrado: Sandwich

FRONT – Top Down



Comparação dos Testes

	Bottom-up	Top-down	Big Bang	Sandwich
Integração	Cedo	Cedo	Tarde	Cedo
Tempo para o funcionamento básico do programa	Tarde	Cedo	Tarde	Cedo
Drivers (mocks) – dublê do front	Sim	Não	Sim	Sim (Alguns)
Stubs (mocks) – dublê do back	Não	Sim	Sim	Sim (Alguns)
Trabalho em paralelo no início	Médio	Baixo	Alto	Médio
Capacidade de Testar caminhos particulares	Fácil	Difícil	Fácil	Médio

<https://qualidadebr.wordpress.com/2009/05/10/tecnicas-de-integracao-de-sistema-big-bang-e-sandwich/>

<https://pt.stackoverflow.com/questions/157330/qual-o-conceito-de-stubs-e-de-drivers-em-testes-de-integra%C3%A7%C3%A3o>



Processos de Software

Contextualização

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

Deixa eu te contar uma coisa...então você é...

RUIM!!!

Facilite a reutilização.

Se for fácil reutilizar, 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.

Não existe lugar para smeagol!



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.



The background is a dark, textured surface with a pattern of fine, glowing white lines that curve and swirl across the frame. Interspersed among these lines are numerous small, bright white dots, some of which have a soft, out-of-focus glow, creating a sense of depth and movement. The overall effect is reminiscent of a starry night sky or a complex, organic network.

Continuando . . . Turma 3ADSA

LOGO . . . PROCESSO DE SOFTWARE

NÃO SERVE PARA NADA!

CERTO?

Atividades Fundamentais

Especificação do Software

- Aula de Análise de Sistemas.

Projeto e Implementação de Software

- Arquitetura, especificações, interfaces, componentes, estrutura de dados, algoritmos.

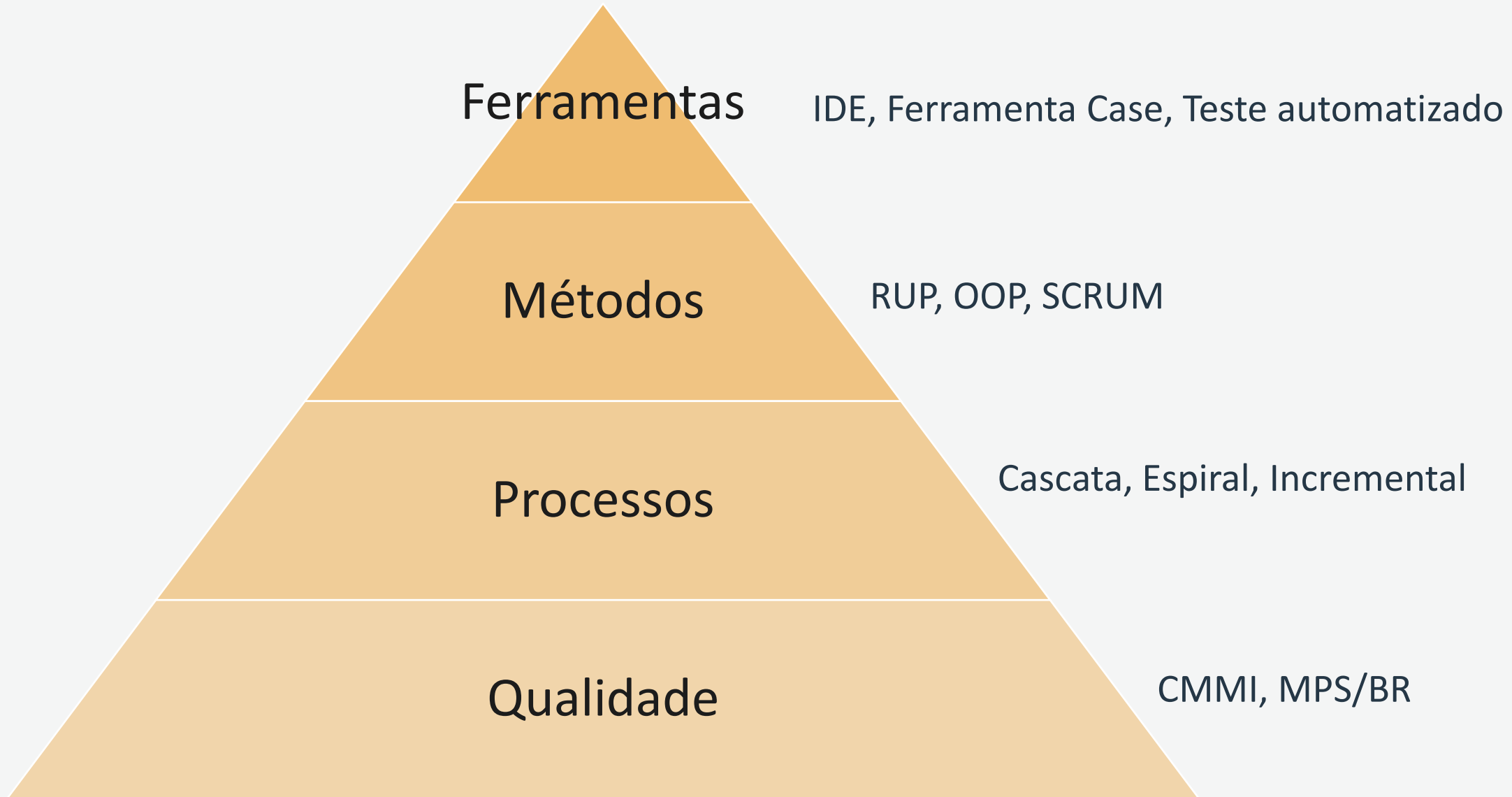
Validação de Software

- Testes de componentes, testes de sistema, testes de aceitação.

Evolução de Software

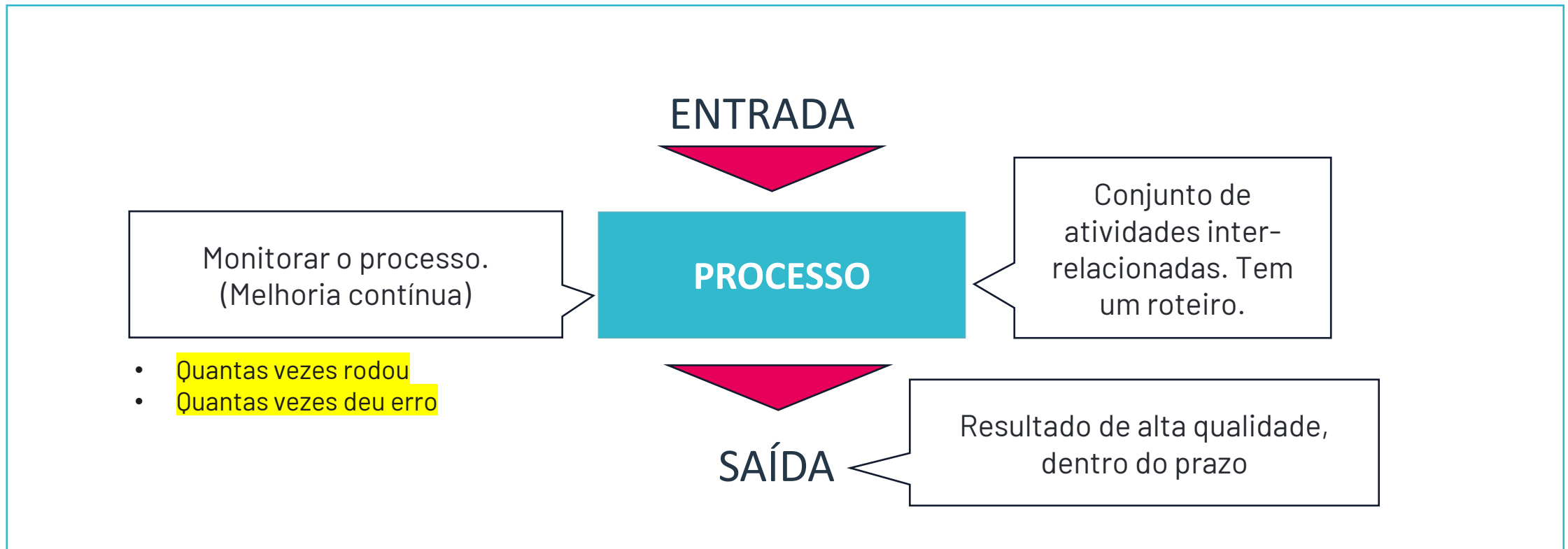
- Avaliar sistemas existentes, propor mudanças, modificar sistemas.

Processos então...tem relação com Qualidade?



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)



QUALIDADE

O que é **Mais importante** na Eng. de Software

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.

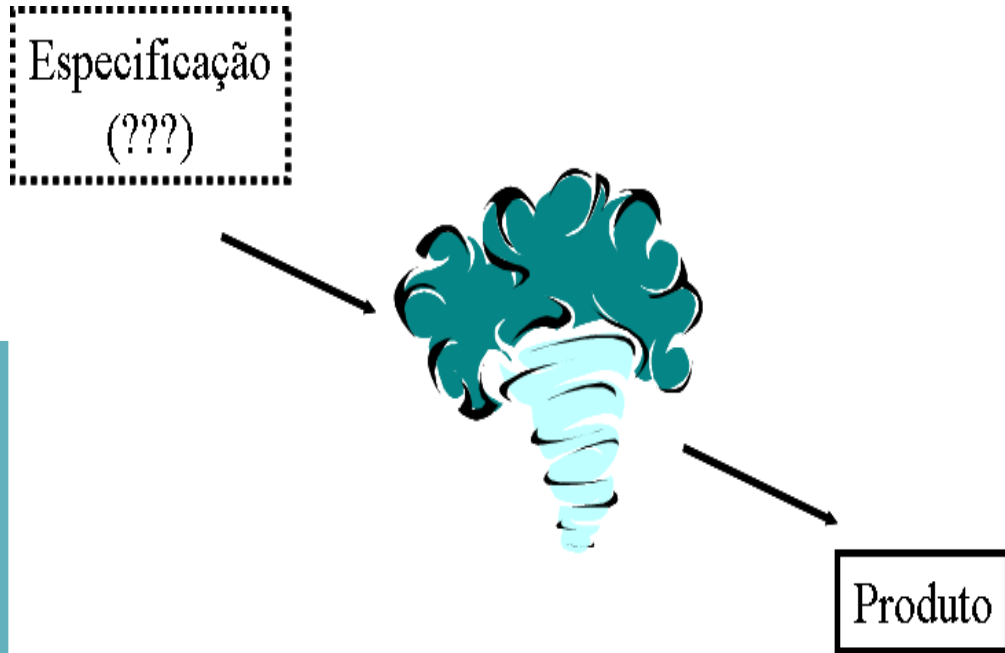
Evolução das Operações

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
Existe a chance de termos outros modelos até o final do século.				
1960-1970	1970-1980	1980-1990	1990-2000	Século XXI

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-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”.

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.

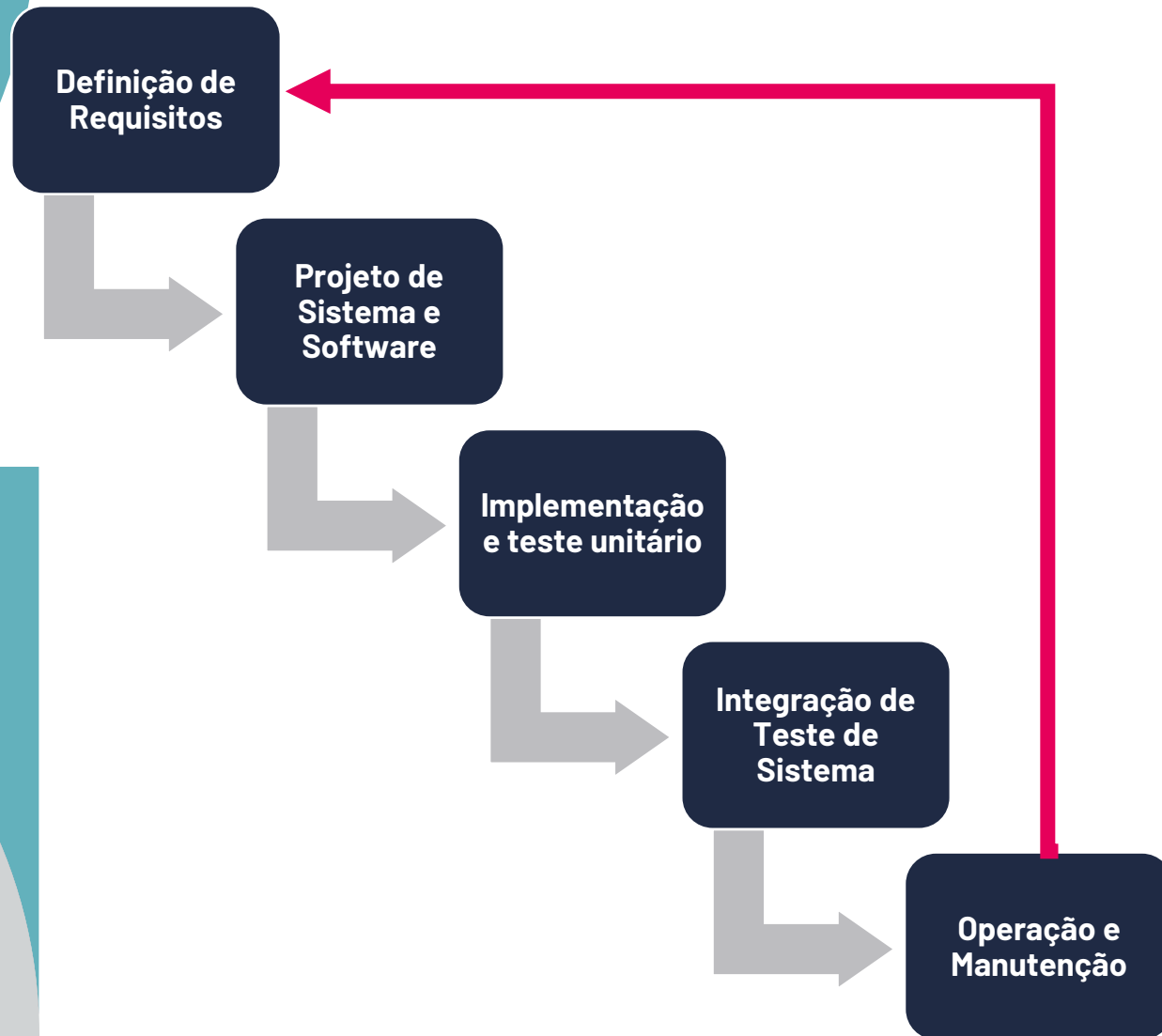
<https://gohorseprocess.com.br/extreme-go-horse-xgh/>

E aí?



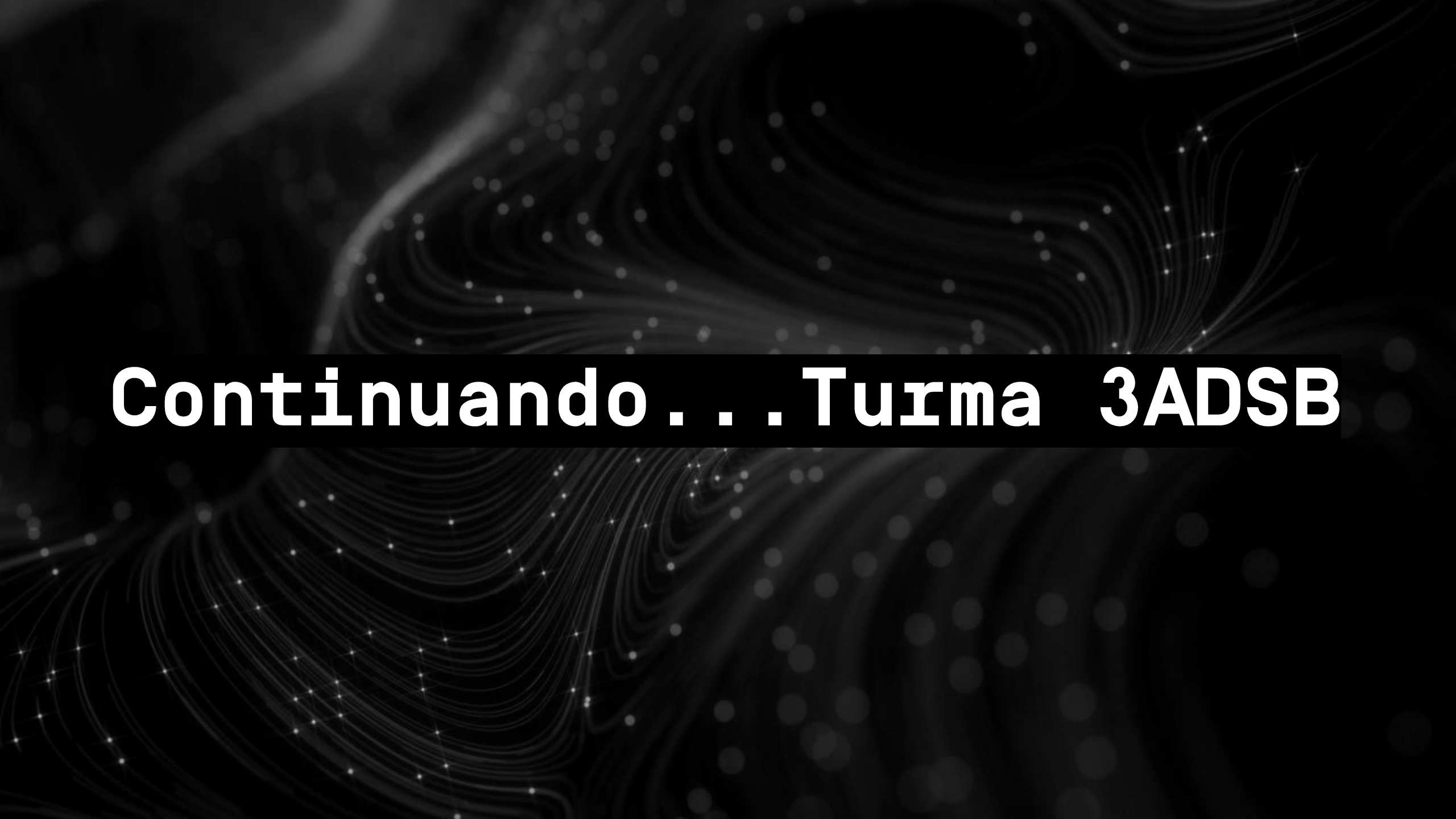
“FAZER DIFERENTE MOTIM VAI VIRAR. APRESENTAR FATOS E OPÇÕES AOS POUCOS VOCÊ DEVE. PACIÊNCIA FUNDAMENTAL É”

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;

- Você gosta que mudem o escopo do projeto que você está trabalhando?
- Você gosta de não saber o que vai fazer amanhã no trabalho? (previsibilidade)

The background is a dark, textured surface with intricate, flowing, and wavy lines in shades of gray and white. These lines create a sense of movement and depth. Scattered throughout the background are numerous small, bright, out-of-focus light spots, resembling bokeh or distant stars, which add to the ethereal and digital feel of the image.

Continuando . . . Turma 3ADSB

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 (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** 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.

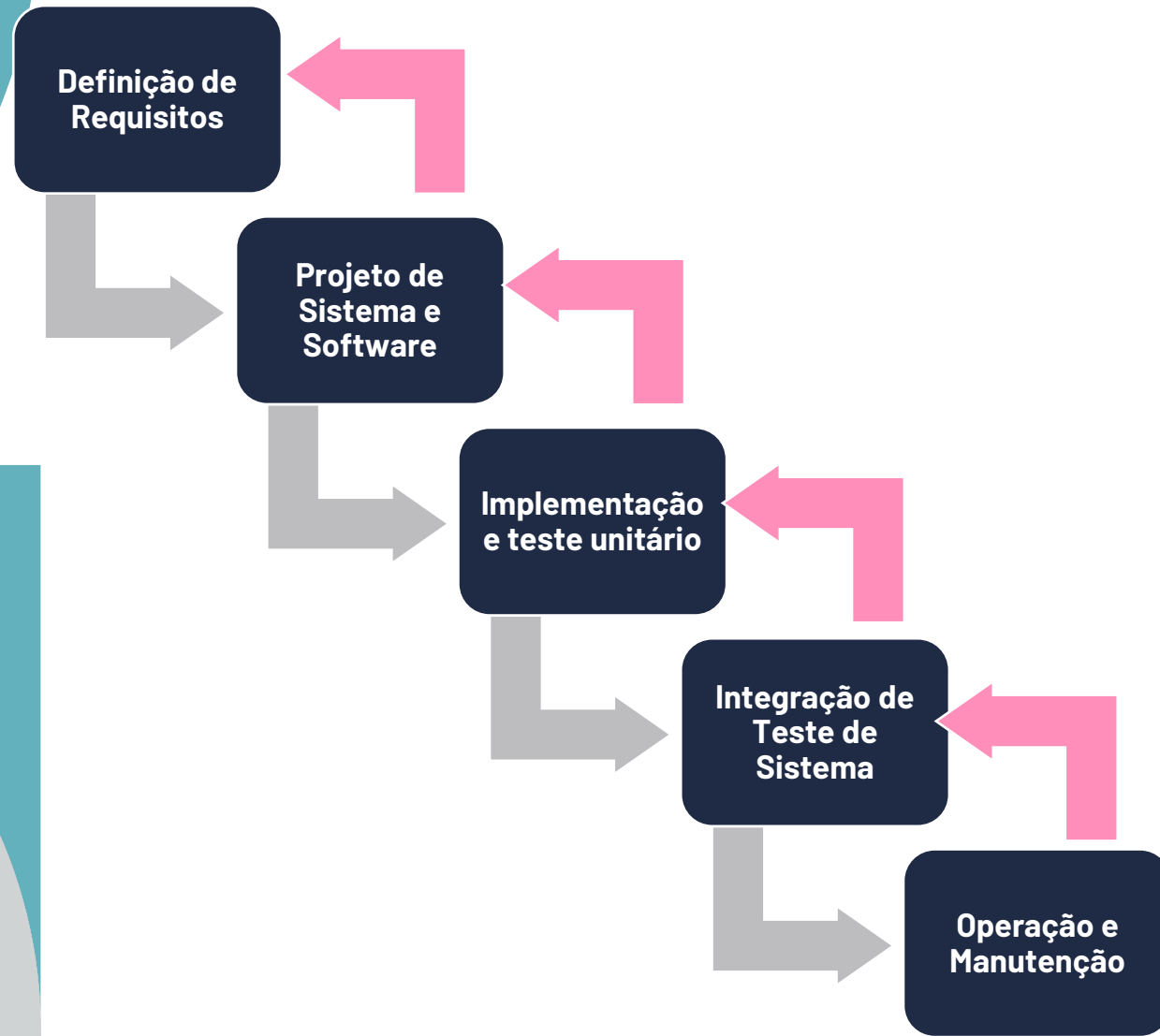
Desenvolvimento: Será dividido em fases para que possamos receber dinheiro do cliente a cada entrega, afinal o projeto é de 2 anos.

Cronograma: Utilizaremos cronograma para planejamento.

Exercício em Sala - Resultado

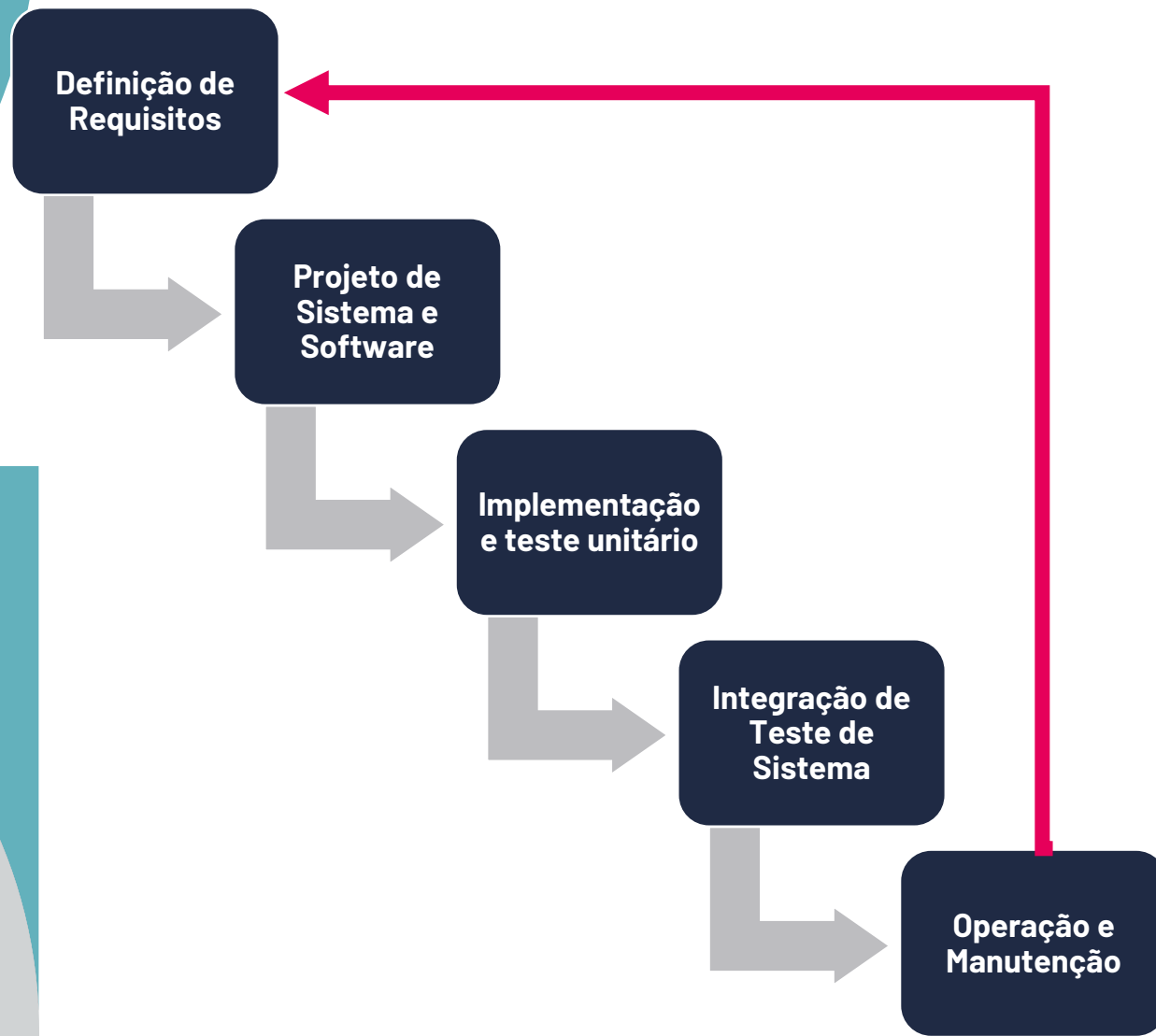
- GRUPO 1 - TRADICIONAL (CASCATA)
- GRUPO 2 - CASCATA SEM REALIMENTAÇÃO
- GRUPO 3 - CASCATA COM REALIMENTAÇÃO
- GRUPO 4 - CASTATA SEM REALIMENTAÇÃO
- GRUPO 5 - CASCATA SEM REALIMENTAÇÃO
- GRUPO 6 - CASCATA COM REALIMENTAÇÃO
- GRUPO 7 - CASCATA COM REALIMENTAÇÃO
- GRUPO 8 - CASCATA COM REALIMENTAÇÃO
- GRUPO 9 - SCRUM + XP

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.

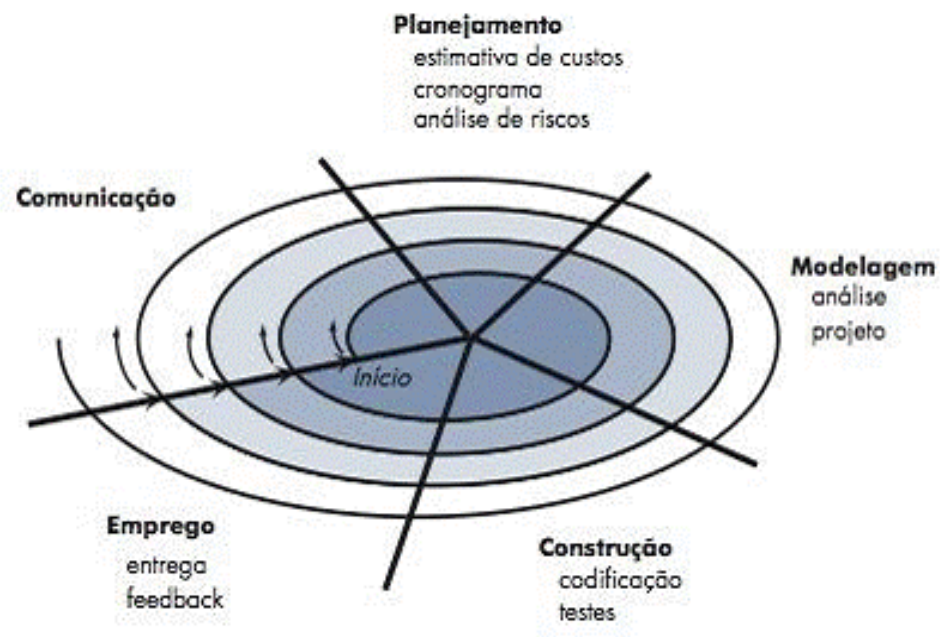
Modelos de processo Evolucionários



Satisfação do Cliente!



Modelo de Processo Evolucionário – Espiral

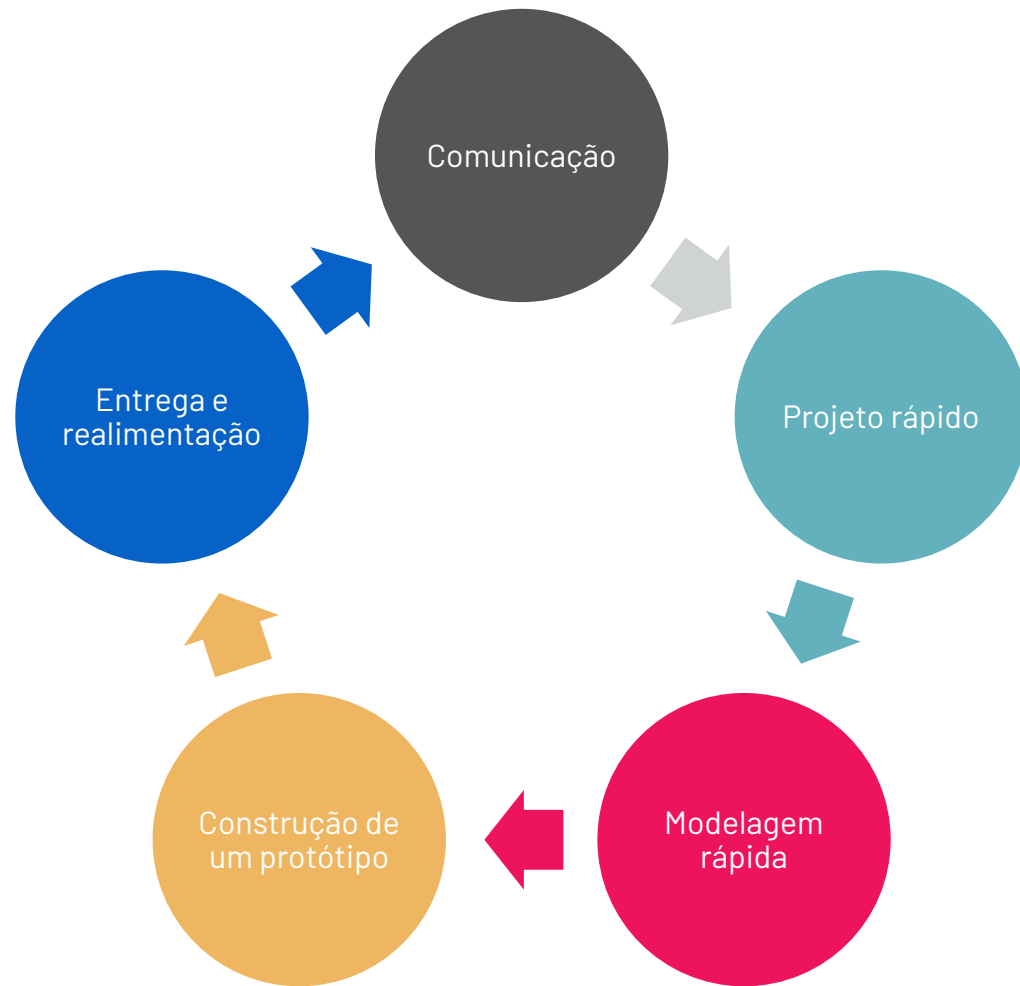


sommerville

- **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;
- **Os riscos são considerados a cada fase de evolução do projeto.**
- **Custo é alto;**
- **Difícil convencer o cliente que está tudo sob controle;**

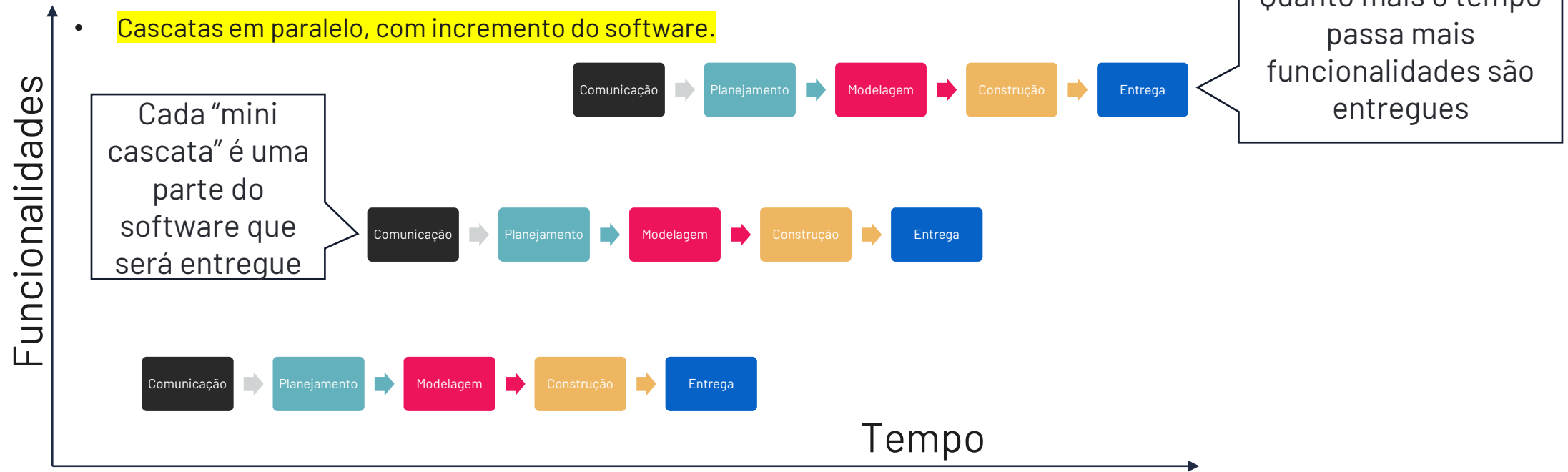
- Planeja, modela, constrói, entrega, planeja, modela, constrói, entrega,... – Para cada ciclo precisa de mais prazo e mais \$. Os clientes não gostam muito desse modelo de processo.

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



- 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;

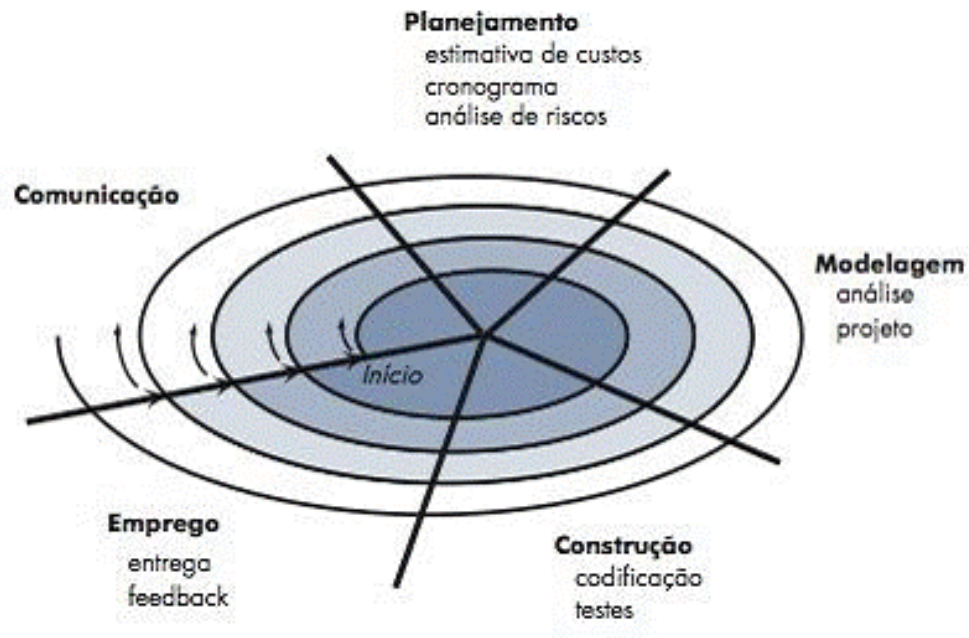
Envolve o desenvolvimento da aplicação a partir de componentes pré-existentes que são integrados a arquitetura;

Preocupação com a integração é fundamental;

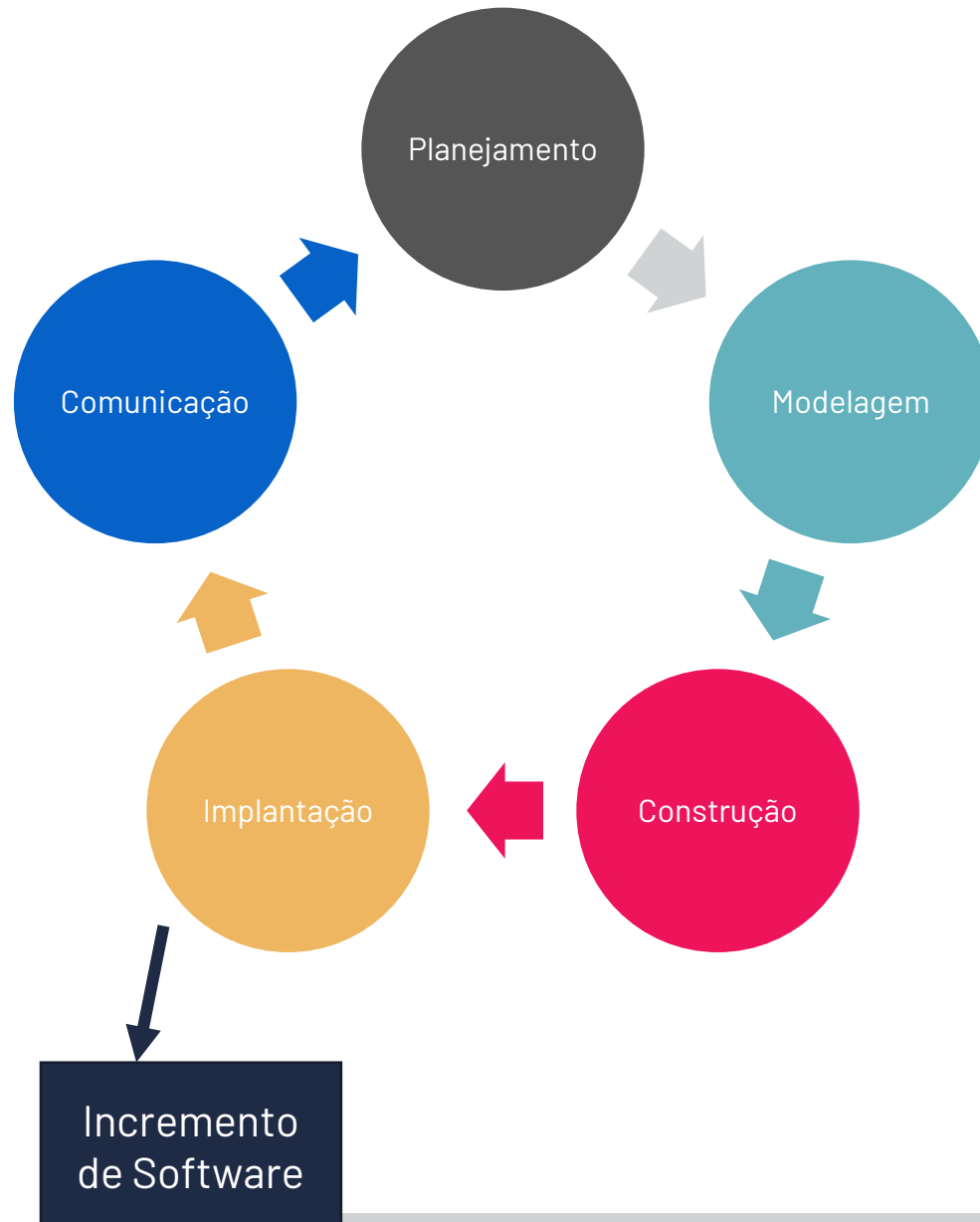
Testes integrados (completos) precisam ser realizados para garantir que o todo funcione corretamente;

O reuso é palavra chefe deste modelo;

- Normalmente utiliza tecnologias orientadas a objeto.
- Expiral + Componentização.

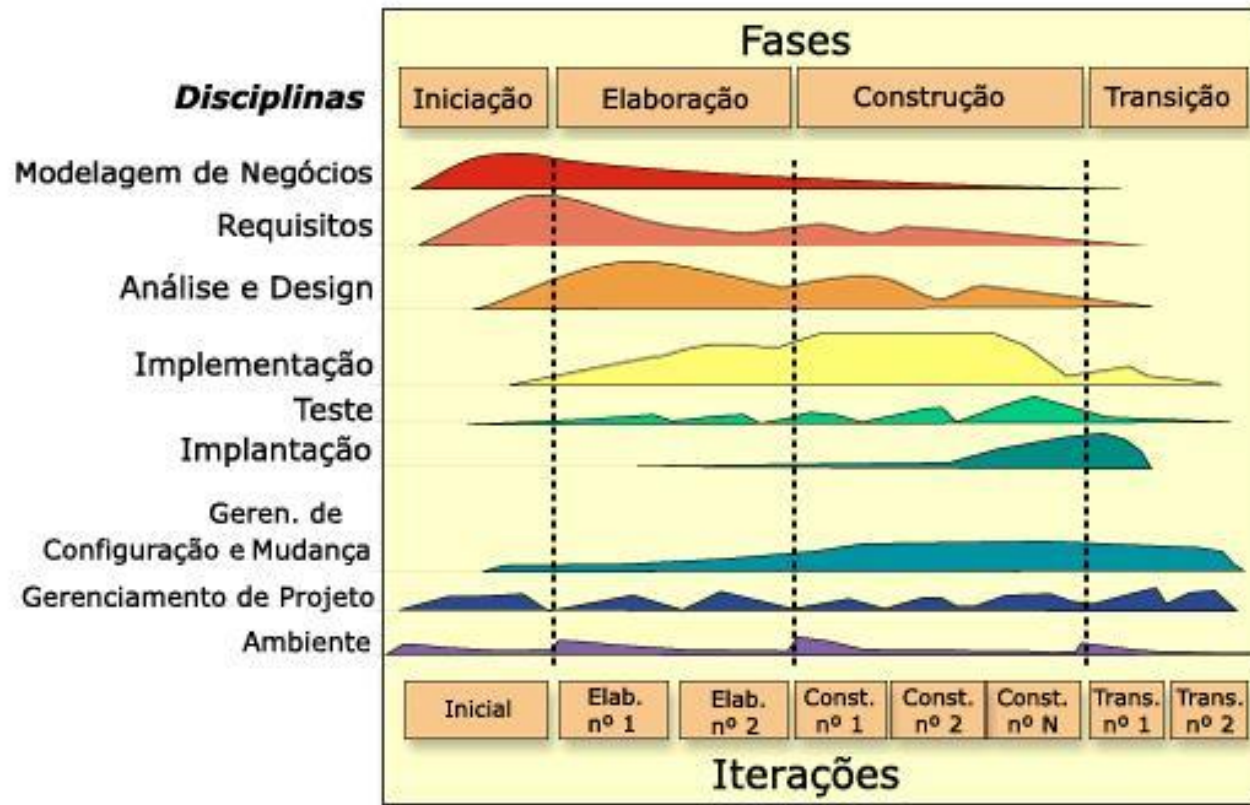


O Processo Unificado



- 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.
- **A cada rodada sai um incremento de software.**

RUP – IBM



- Mais relacionadas ao negócio que aspectos técnicos;
- Utiliza abordagem de orientação a objetos;
- 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?

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.

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 (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** 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.

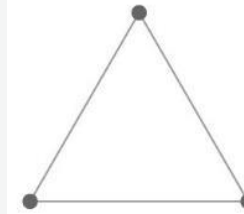
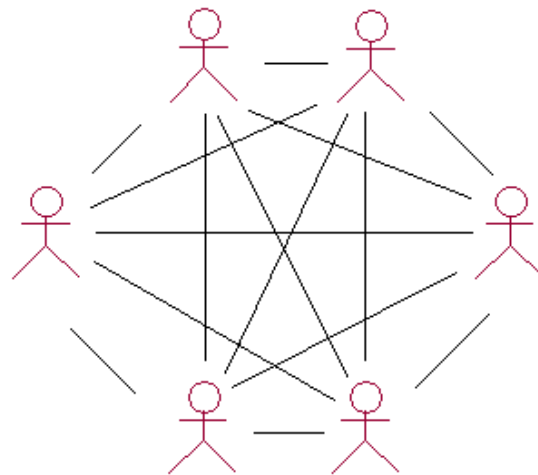
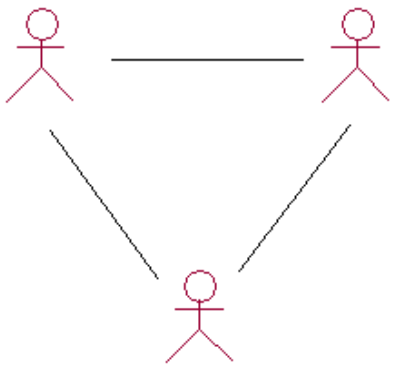
Desenvolvimento: Será dividido em fases para que possamos receber dinheiro do cliente a cada entrega, afinal o projeto é de 2 anos.

Cronograma: Utilizaremos cronograma para planejamento.

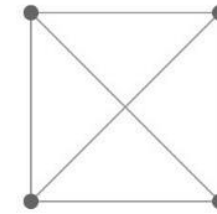
Exercício em Sala - Resultado

- GRUPO 1 - TRADICIONAL (CASCATA)
- GRUPO 2 - CASCATA SEM REALIMENTAÇÃO
- GRUPO 3 - CASCATA COM REALIMENTAÇÃO
- GRUPO 4 - CASTATA SEM REALIMENTAÇÃO
- GRUPO 5 - CASCATA SEM REALIMENTAÇÃO
- GRUPO 6 - CASCATA COM REALIMENTAÇÃO
- GRUPO 7 - CASCATA COM REALIMENTAÇÃO
- GRUPO 8 - CASCATA COM REALIMENTAÇÃO
- GRUPO 9 - SCRUM + XP

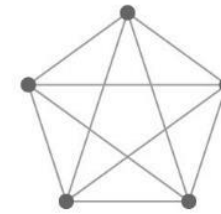
Motivações para o Agile



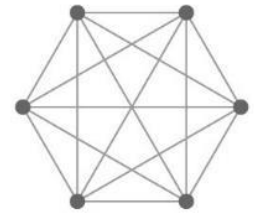
3 people, 3 lines



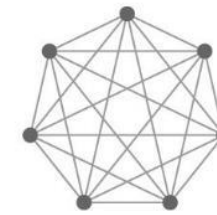
4 people, 6 lines



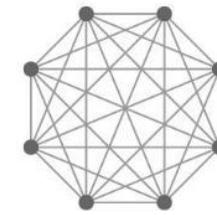
5 people, 10 lines



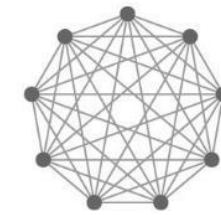
6 people, 15 lines



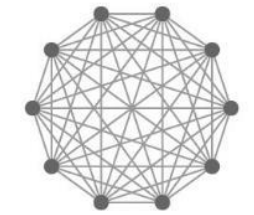
7 people, 21 lines



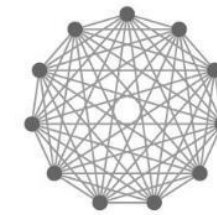
8 people, 28 lines



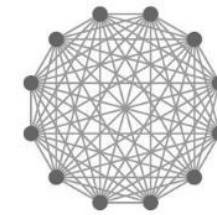
9 people, 36 lines



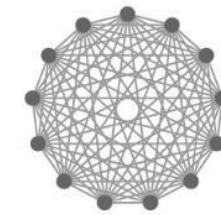
10 people, 45 lines



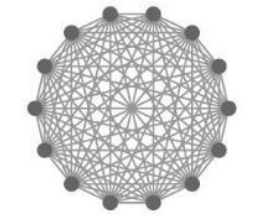
11 people, 55 lines



12 people, 66 lines



13 people, 78 lines

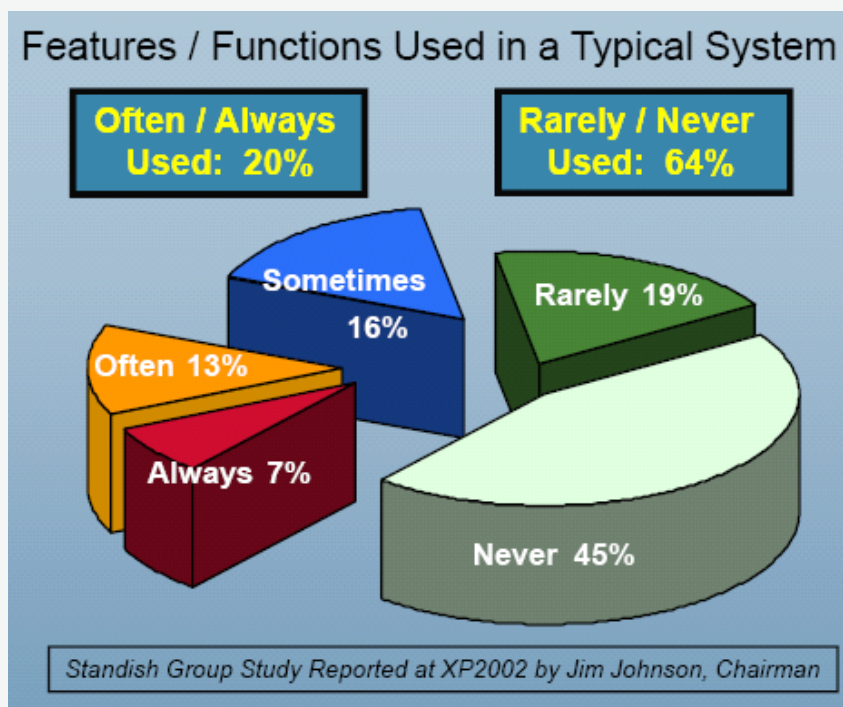


14 people, 91 lines

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

A comunicação fica muito complexa em equipes grandes.

Motivações para o Agile



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

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

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

Motivações para o Agile

MUNDO VUCA



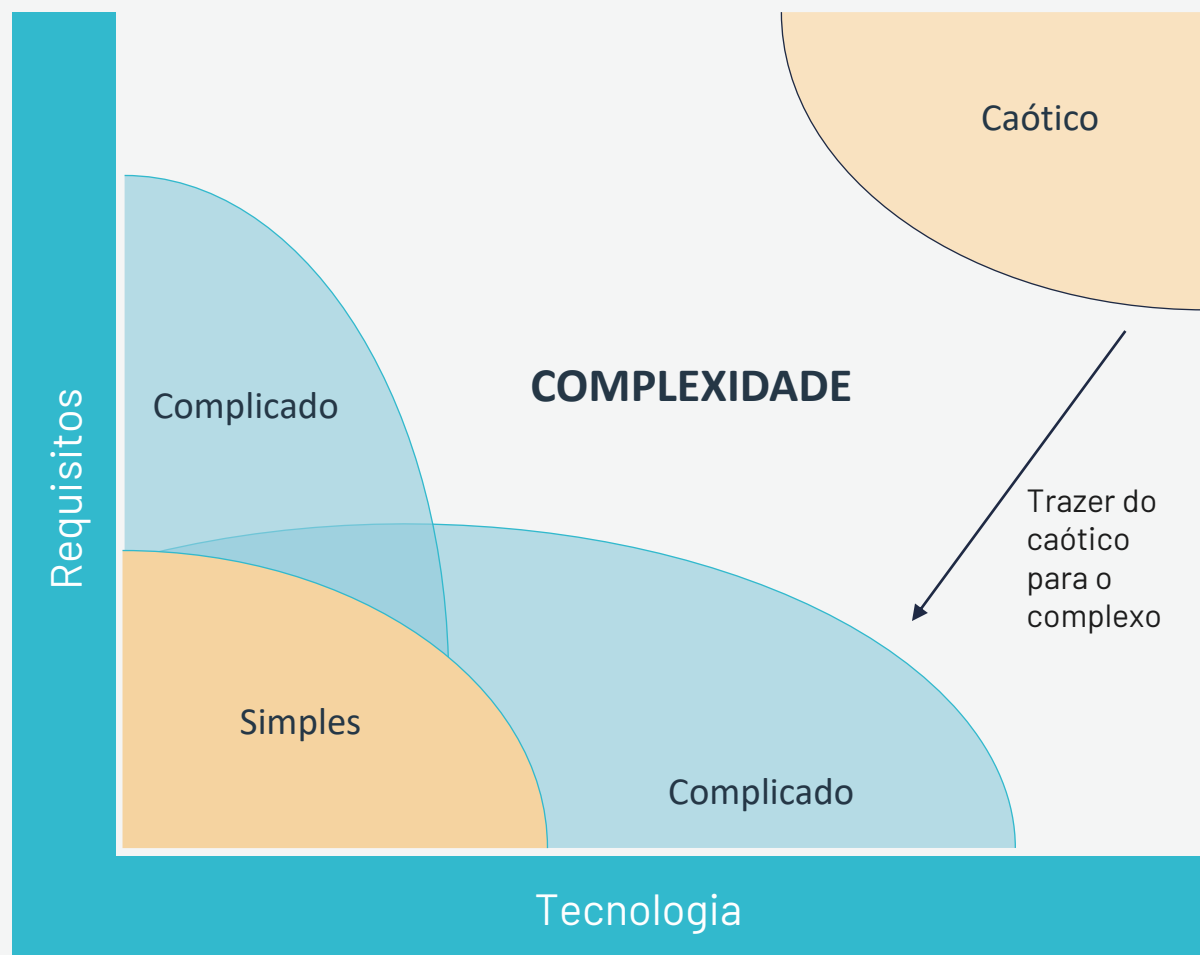
- Conceito vem das forças armadas americanas.
- O mundo atual é **caótico**.
- Se você sabe pouco sobre a situação e não consegue ter ações previsíveis, você não enfrenta nem o primeiro quadro, que é o da ambiguidade.
- Em português fica VICA. VUCA vem do Inglês.

Motivações para o Agile

“Teoria do Caos”

Matriz Stacey – Sistemas Complexos

Longe do Acordo
Ex: Requisitos Mutantes



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

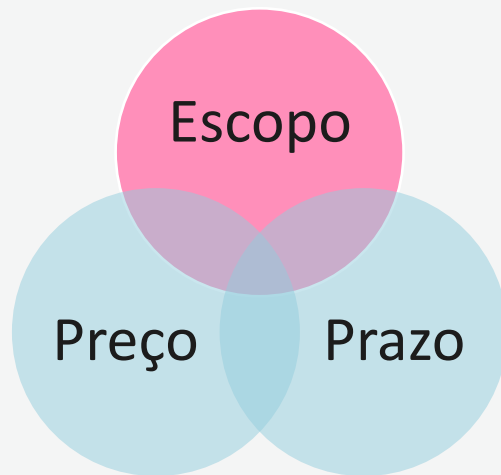
Trazer do caos para o complexo ajuda a controlar.

- 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.

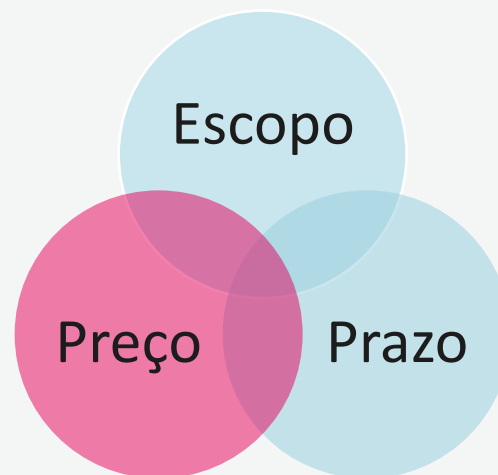
Teoria das Restrições - PMBOK



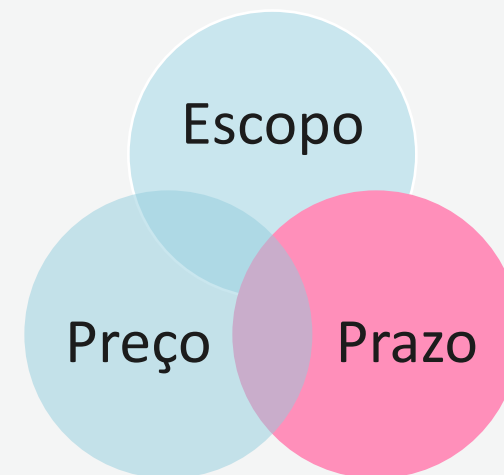
Nem em Conto de Fadas



Projeto Crítico
Escopo não pode alterar



Restrições de Orçamento



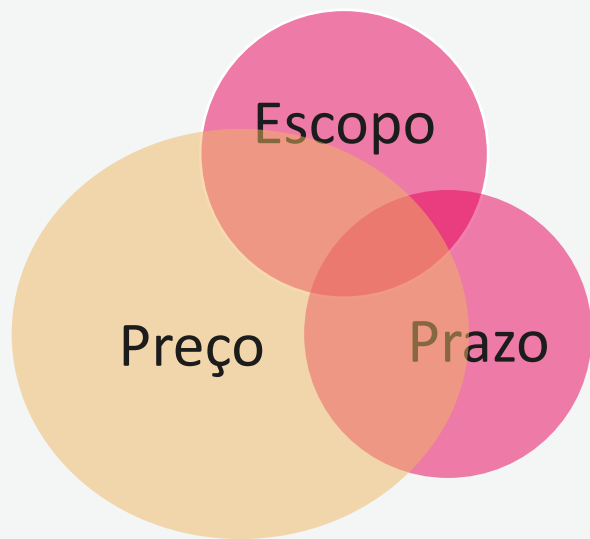
Projeto Urgente
(todos são 😊)

● Não pode mexer.

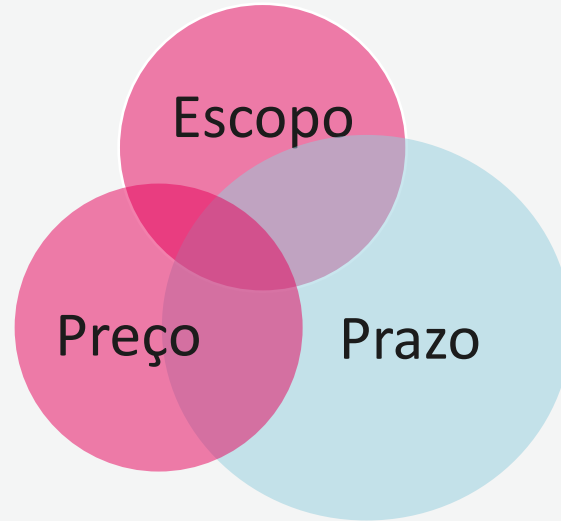
Estes são os cenários mais simples...

Teoria das Restrições - PMBOK

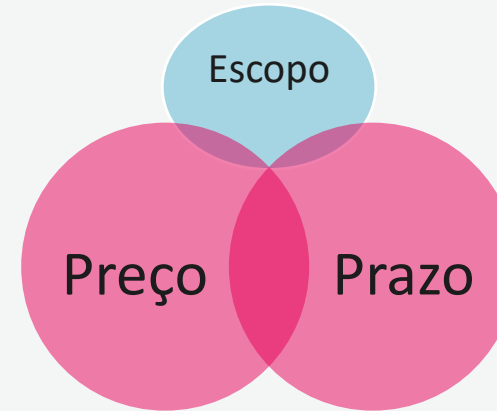
...Agora sim, a vida como ela é...



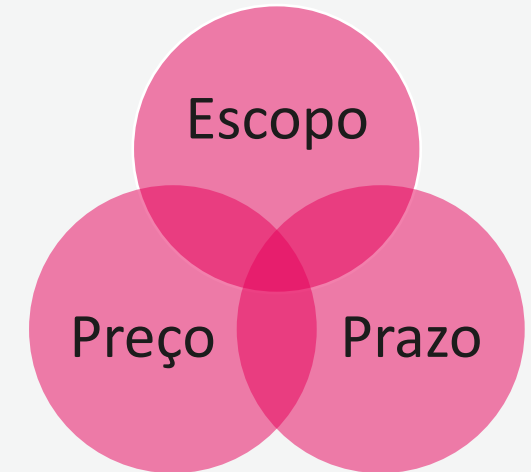
Projeto Crítico e Urgente
(Precisa de mais \$)



Projeto Crítico e com Pouco \$
(Aumenta o Prazo)



Pouco \$ e Urgente
(Reduz funcionalidade)



Falácia - Caos
(Coloque razão na situação)

Manifesto 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.

Os itens da direita também são importantes!

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

Iterativo (de iterar, repetir) + Incremental

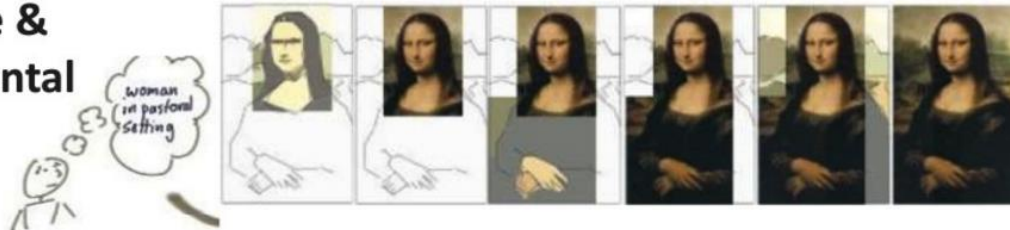
Iterative



Incremental



Iterative & Incremental

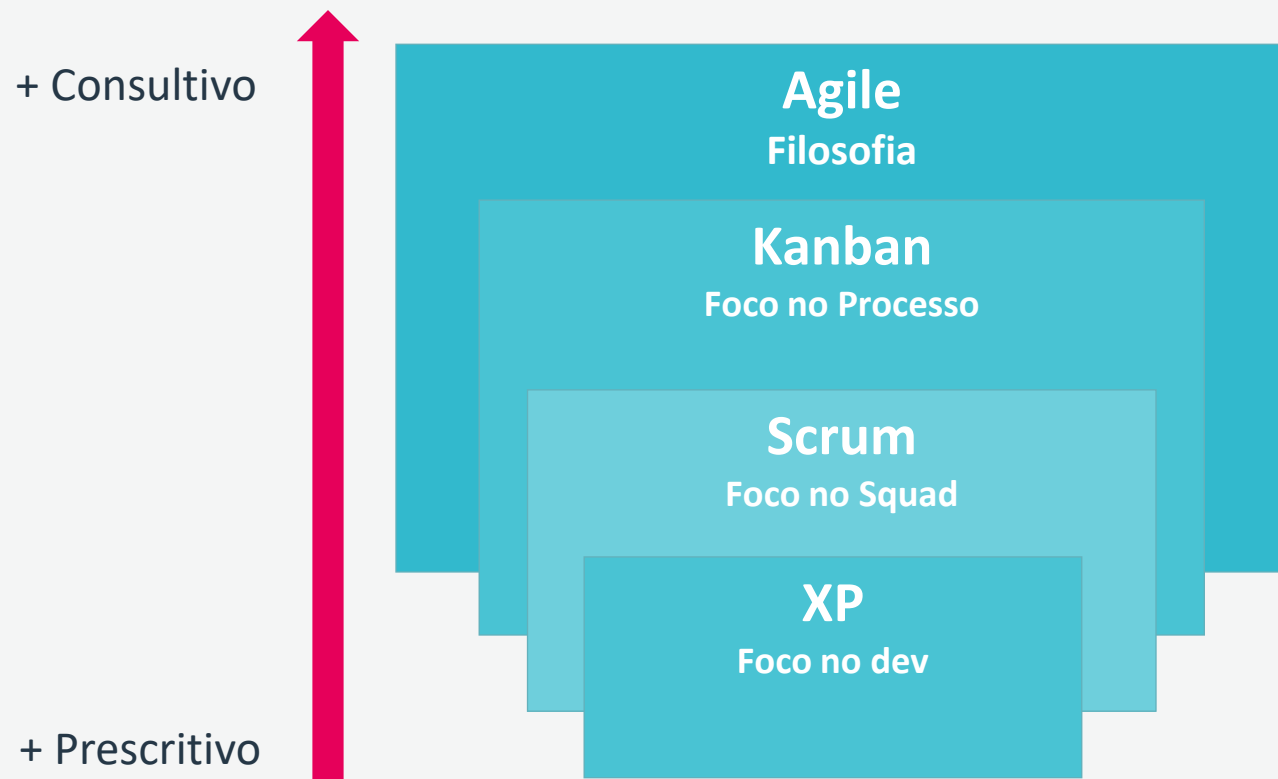


<https://medium.com/@SupriyaL/why-and-how-to-use-agile-methodology-for-an-online-food-delivery-service-503ad5cf1941>

- 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).

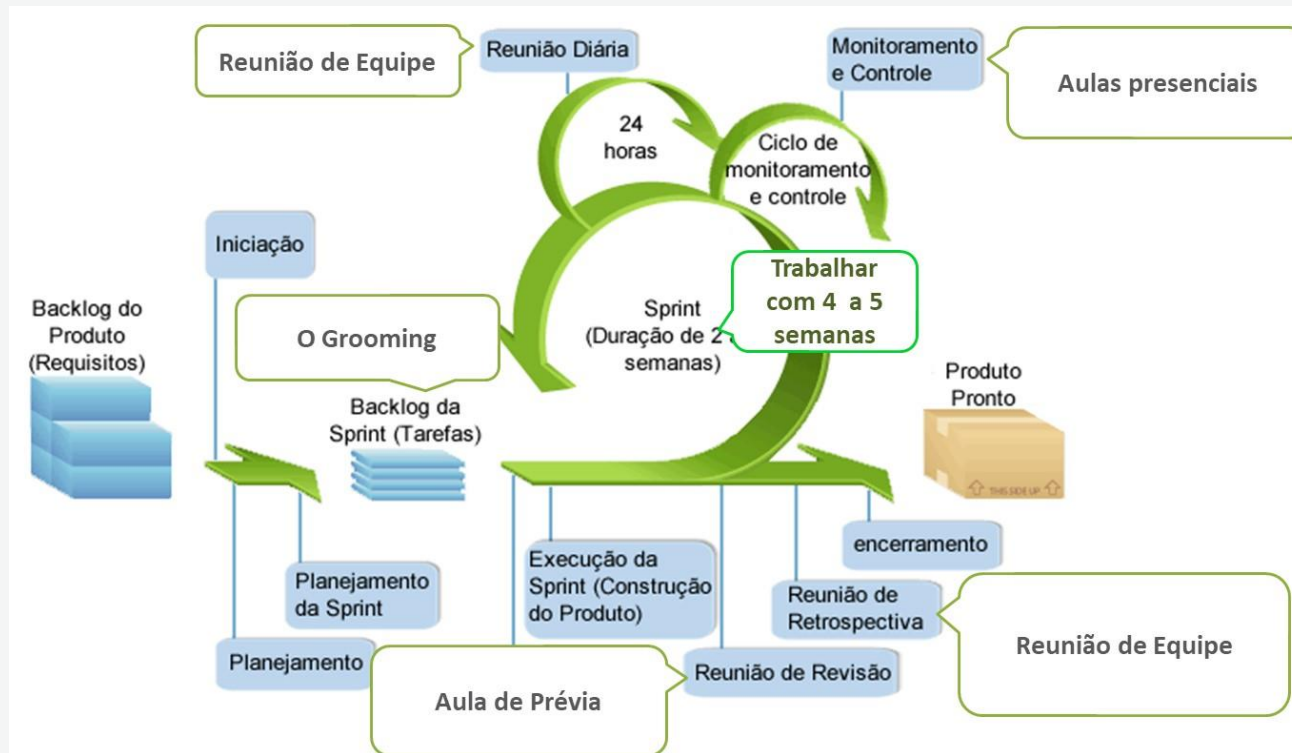
Agile

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.

Agile



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!

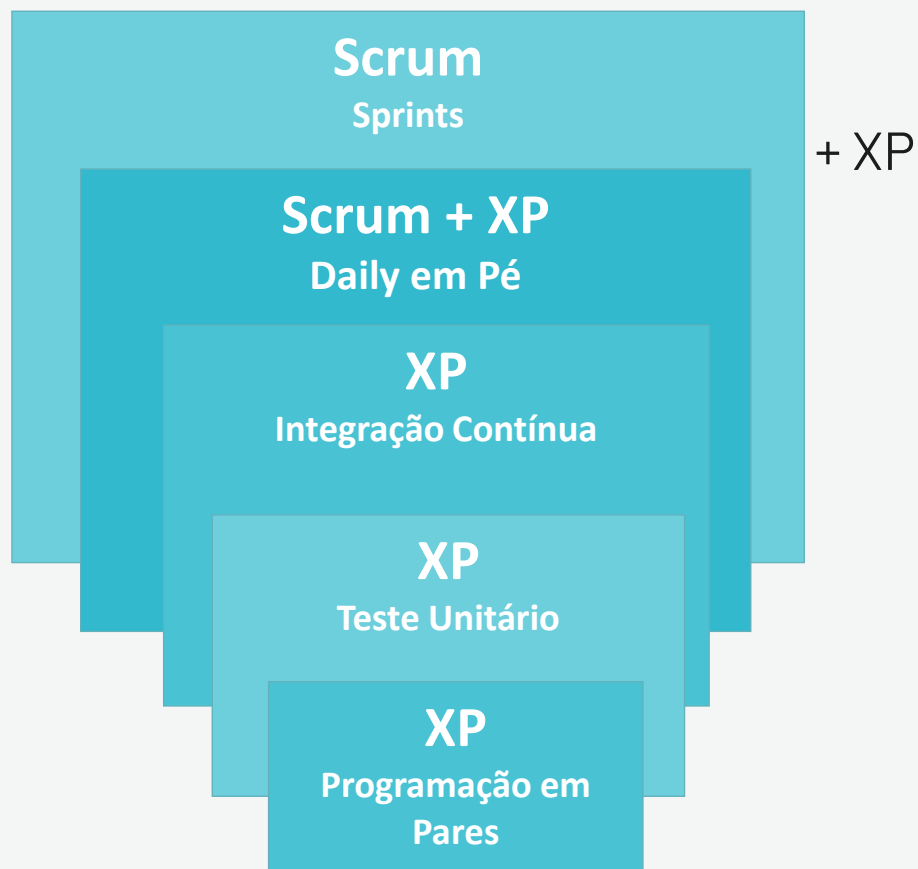
Refatoração
Processo de melhoria
contínua da programação.
Evitar a duplicação e
maximizar o reuso.

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

Time Coeso (harmonia)
Comunicação!
Usar o socioemocional

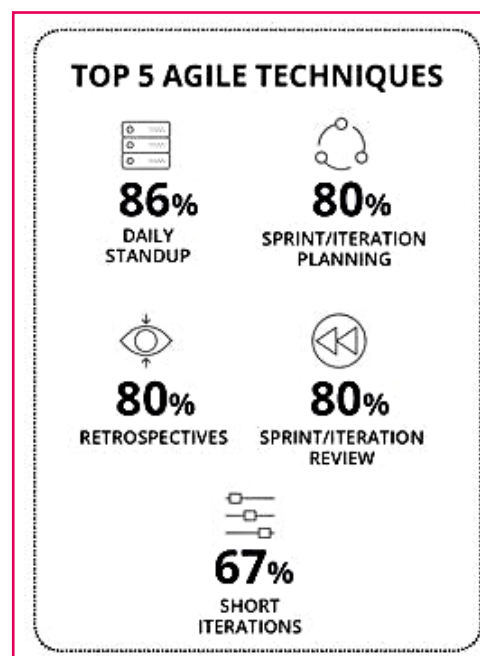
SCRUM + XP

As empresas combinam as práticas ágeis.

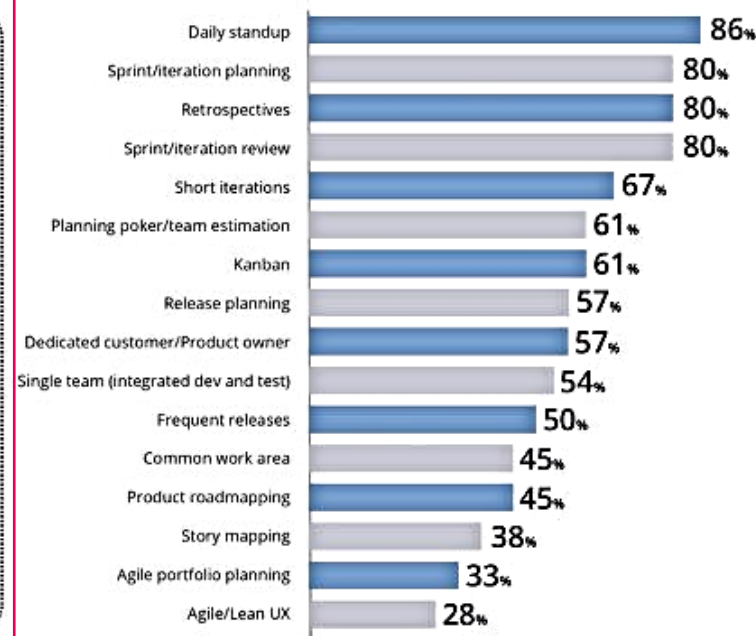


Agile Techniques Employed

Notable changes in agile techniques and practices that respondents said their organization uses were Release planning (57% this year compared to 67% last year) and Dedicated customer/product owner (57% this year compared to 63% last year).



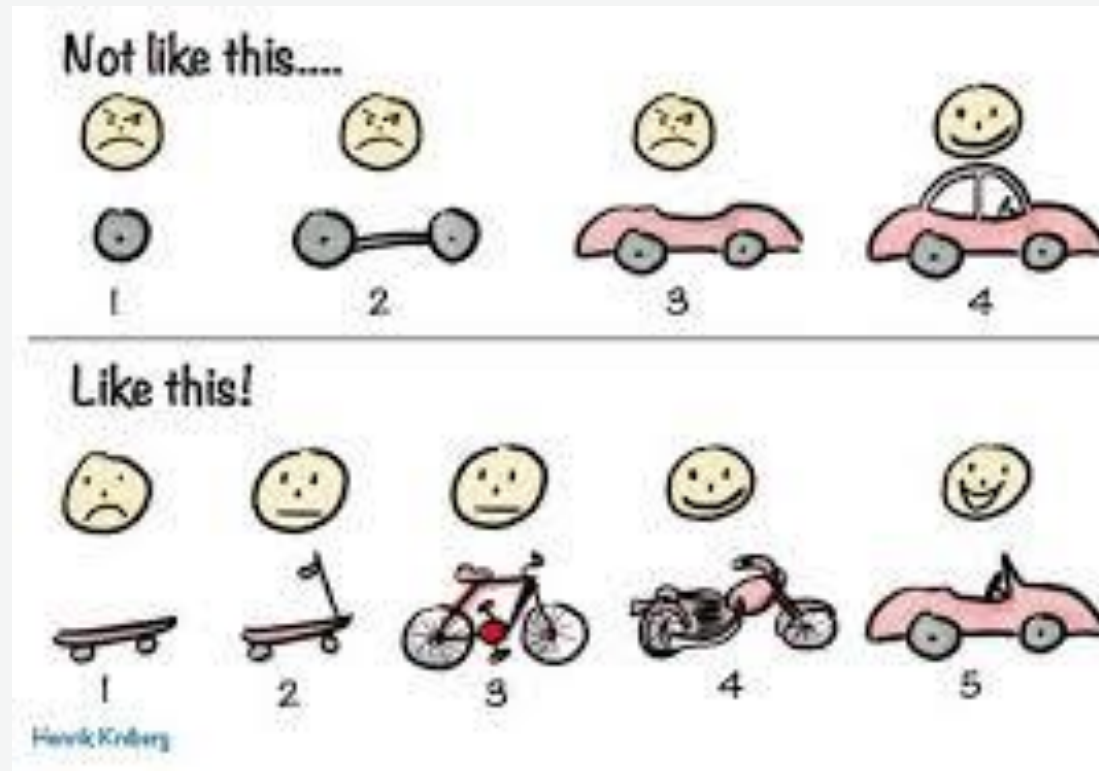
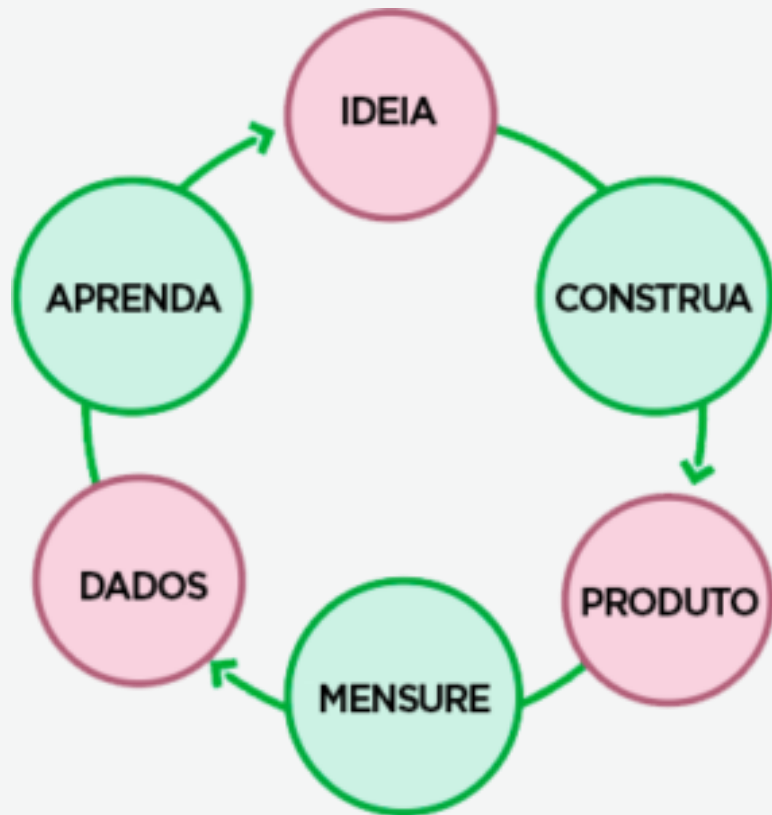
*Respondents were able to make multiple selections



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

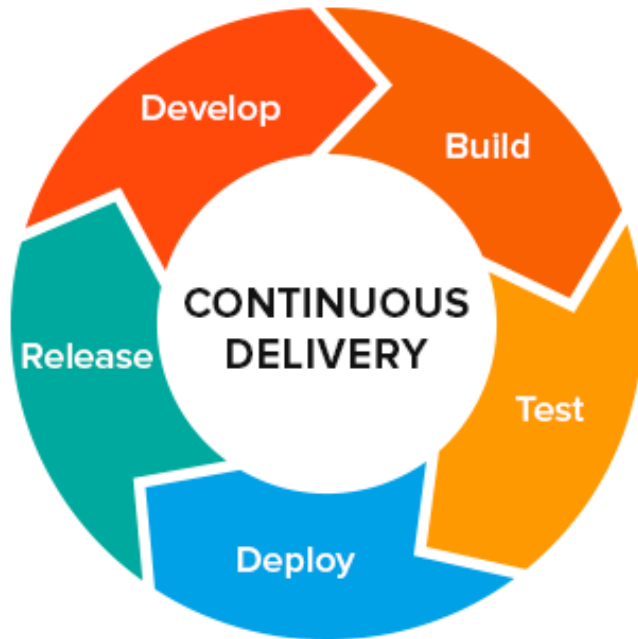
Abordagens Modernas - MVP

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

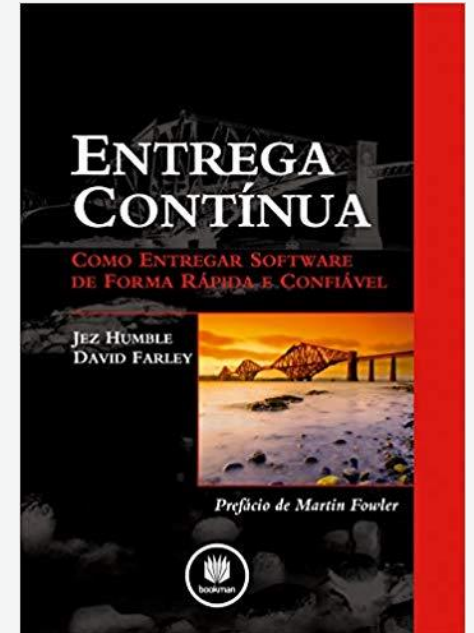


Se você já sabe muito bem o que vai fazer ou é uma nova demanda que é extensão do produto atual, por que razão utilizar esta abordagem?

Abordagens Modernas – Continuous Delivering



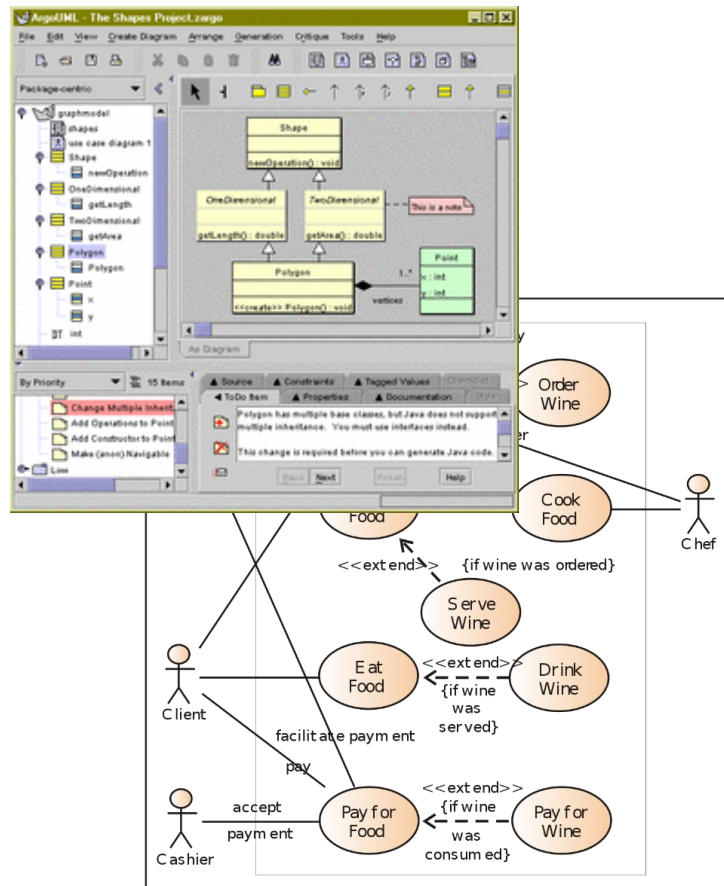
Abordagem em que o objetivo é entregar código em produção com qualidade e mais frequência, ou seja, pequenas partes mais rápido, assim em tese a chance de erros e impacto dos erros deve reduzir.



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

Ferramenta 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;

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

Classificação Funcional das Ferramentas	
Planejamento	PERT, Planilhas, Gerenciamento de Projeto
Edição	Editores de Texto, Editores de diagramas
Gerenciamento de Mudanças	Controle de requisitos, sistemas de controle de mudanças
Gerenciamento de Configuração	Gerenciamento de versões, construção de sistemas
Prototipação	Geradores de interface com o usuário
Apoio a métodos	Dicionário de dados, geradores de código
Processamento de linguagens	Compiladores, interpretadores
Análise de Programa	Analísadores estáticos e dinâmicos
Testes	Comparadores de arquivos, automação de testes
Depuração	Sistemas de depuração interativos
Documentação	Formatação de páginas, editores de imagens
Métrica	Estimar curso e esforço

EXERCÍCIO

Classificação Funcional das Ferramentas

Planejamento	PERT, Planilhas, Gerenciamento de Projeto
Edição	Editores de Texto, Editores de diagramas
Gerenciamento de Mudanças	Controle de requisitos, sistemas de controle de mudanças
Gerenciamento de Configuração	Gerenciamento de versões, construção de sistemas
Prototipação	Geradores de interface com o usuário
Apoio a métodos	Dicionário de dados, geradores de código
Processamento de linguagens	Compiladores, interpretadores
Análise de Programa	Analisadores estáticos e dinâmicos
Testes	Comparadores de arquivos, automação de testes
Depuração	Sistemas de depuração interativos
Documentação	Formatação de páginas, editores de imagens
Métrica	Estimar curso e esforço

FERRAMENTA 1

Nome:

Classificação:

Objetivo da Ferramenta:

Descrição simplificada do funcionamento (5 a 10 linhas):

Ganho que pode ser obtido:

O grupo utilizaria a ferramenta? () SIM () NÃO. Por que?

Links de Referência

Agradeço
a sua atenção!

Gerson Santos

gerson.santos@sptech.school

SÃO
PAULO
TECH
SCHOOL