Projeto Front End – IBM1741

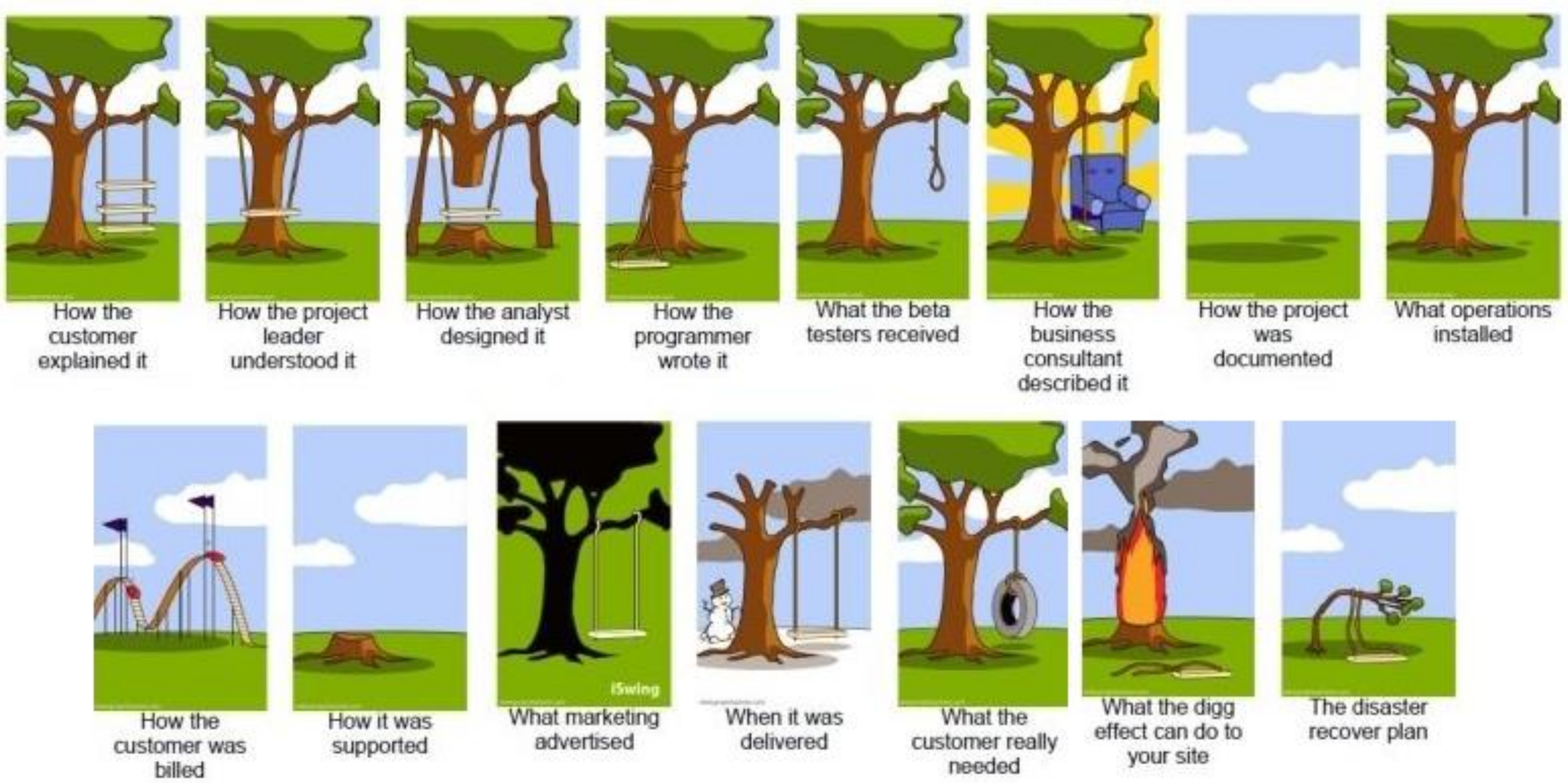
Msc. Jonh Edson Ribeiro de Carvalho

ibmec

Processo de desenvolvimento de software

- Divisão do desenvolvimento de software em uma série de fases e atividades, objetivando um melhor planejamento e uma melhor gestão.

Por que Projetos fracassam?

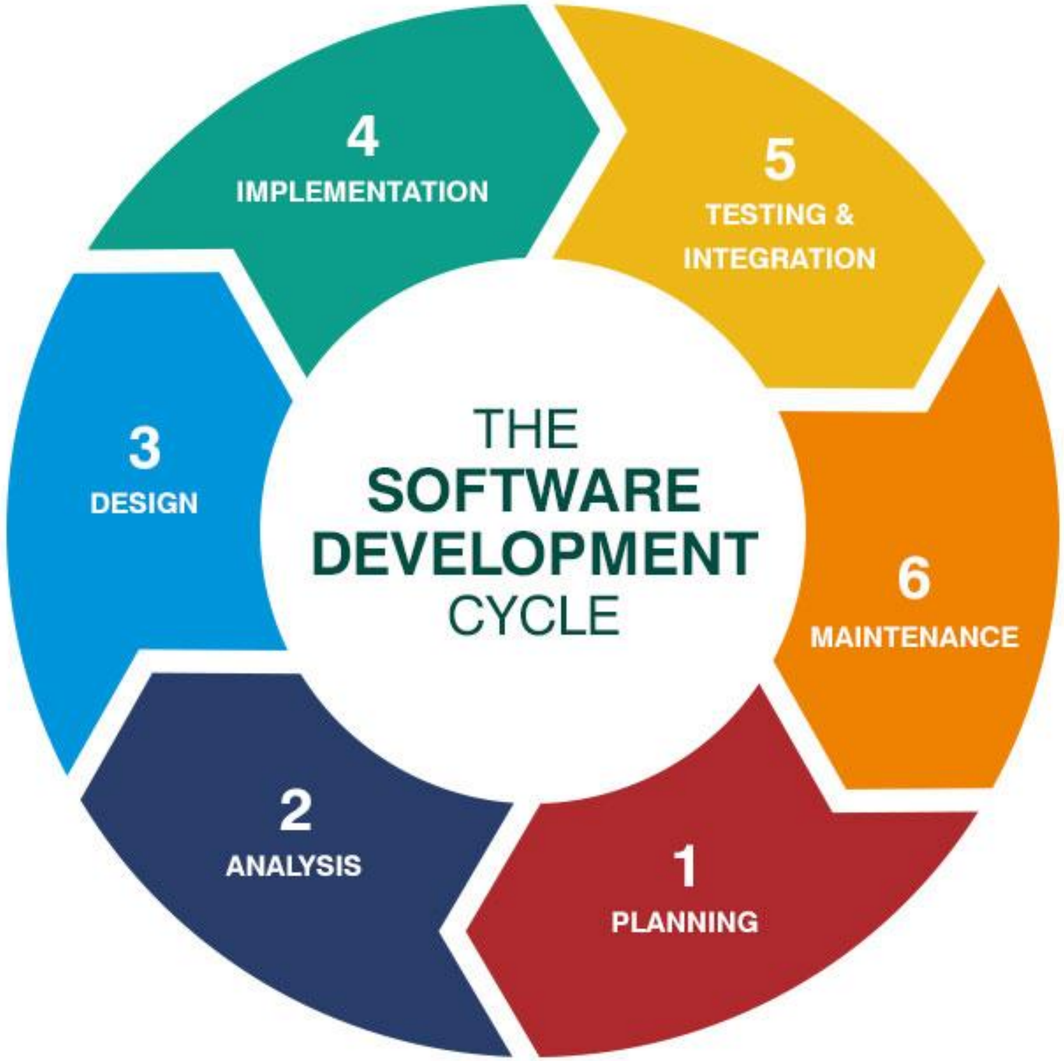


Por que Projetos fracassam?

- Onde estão as falhas e que tipo de falhas ocorrem?
  - Falhas de planejamento
  - Falhas de comunicação
    - Interna à equipe do projeto, com o cliente e do cliente
  - Falhas de execução
  - Falhas de verificação
  - Falhas de documentação
  - Falhas de (...)

Ciclo de Vida de Desenvolvimento de Software

- ISO/IEC 12207



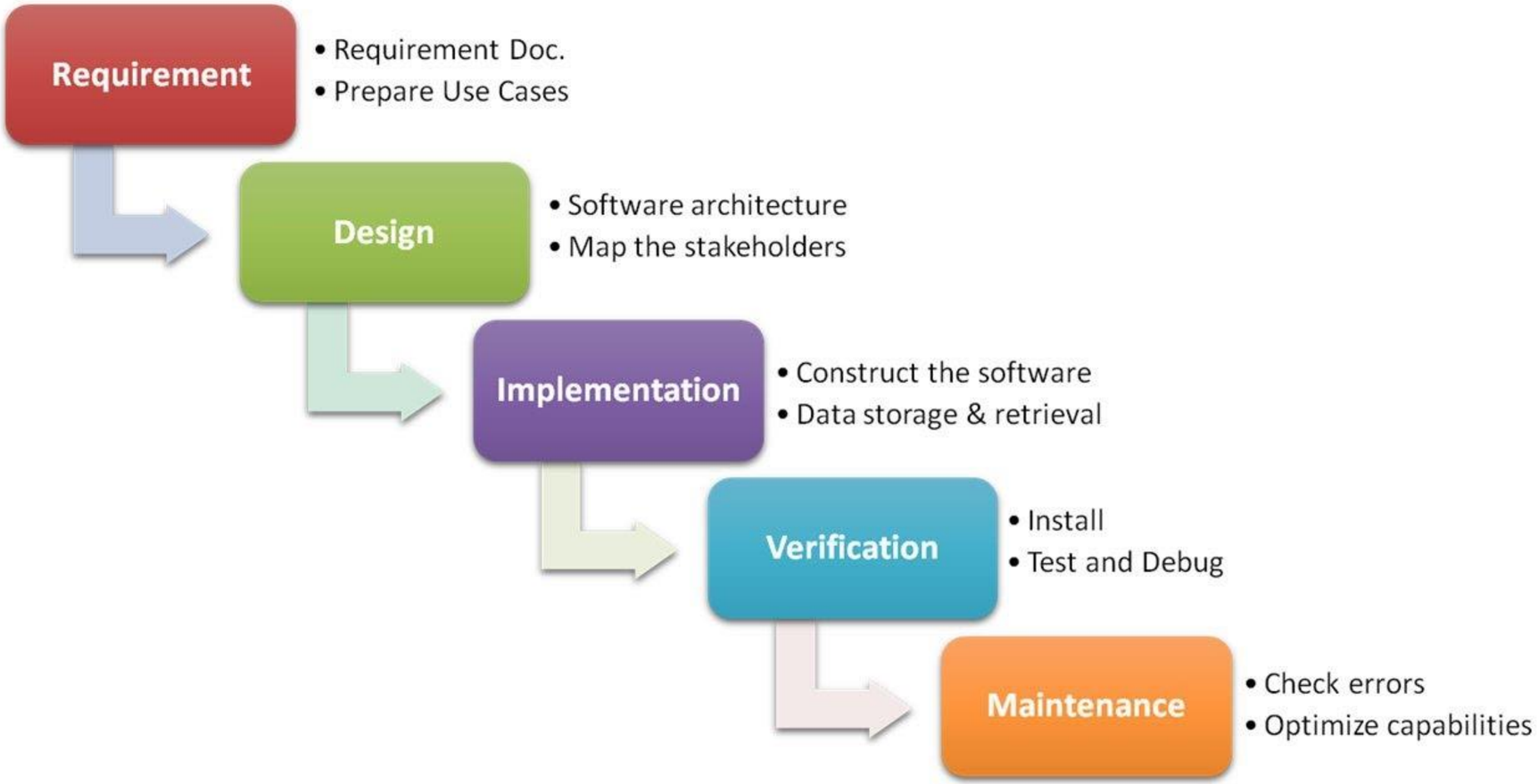
Tipos de Processos de Software

- Modelos lineares-sequenciais
  - Modelo Cascata
- Modelos incrementais e iterativos
  - RUP
  - XP
  - Metodologias Ágeis

Modelo Cascata

- Definido por Winston W. Royce em 1970
  - Ciclo de vida linear-sequencial de software
  - Cada estágio deve ser concluído antes do início do seguinte
  - Ao fim de cada estágio há uma revisão para verificar conformidade com os requisitos

Modelo Cascata



Modelo Cascata

- Vantagens:
  - Forte aspecto de documentação
  - Fácil compreensão
  - Fácil coordenação devido à rigidez do modelo
  - Estágios são executados um por vez

Modelo Cascata

- Desvantagens:
  - Surgimento de novos requisitos
  - Nem todos os problemas detectados durante uma fase podem ter sido corrigidos
  - Inflexibilidade das fases
  - Estimativa de custo e tempo para cada estágio é dificultosa
  - Se erros forem detectados nos testes, é difícil retornar à fase de projeto

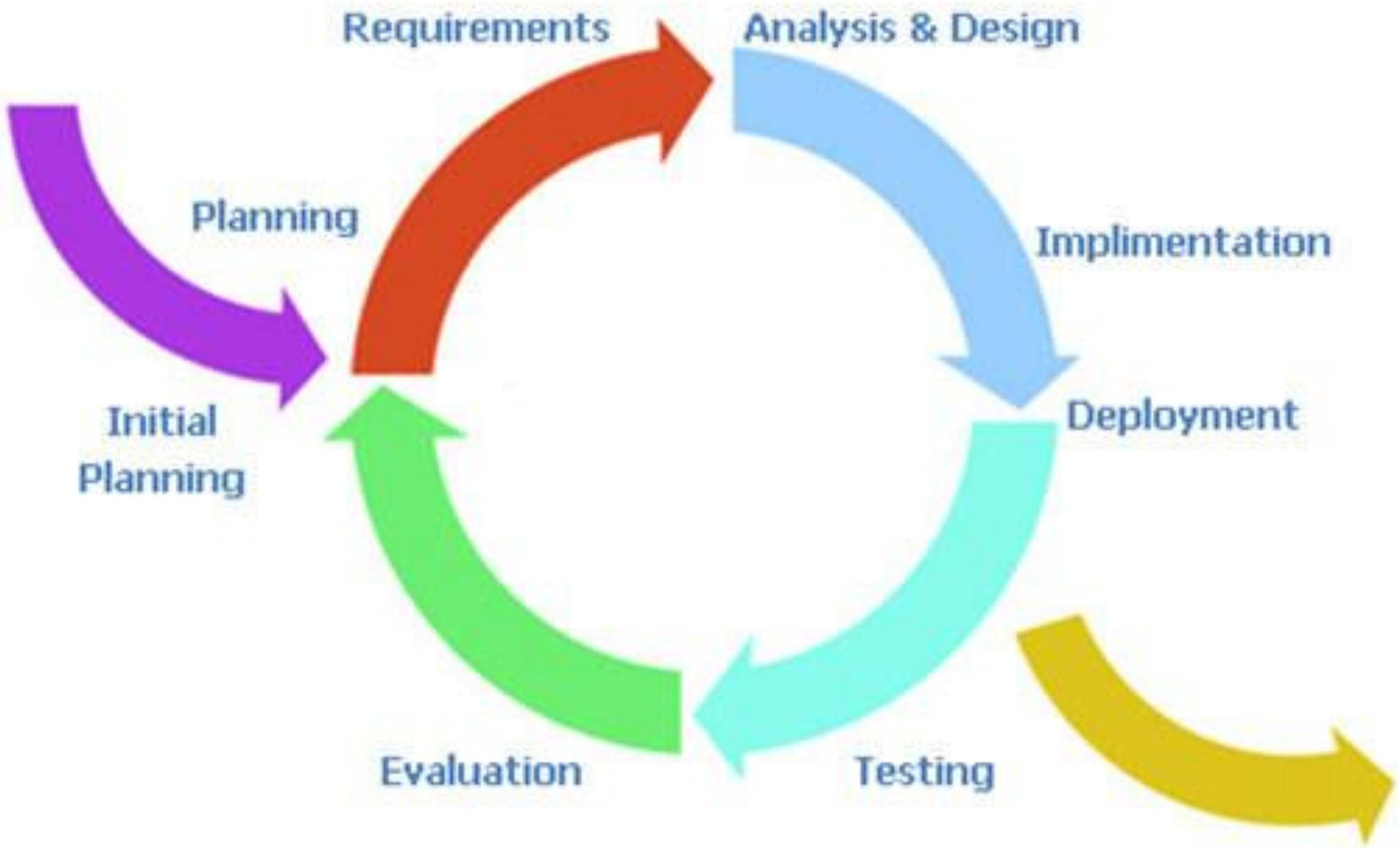
Modelo Cascata

- Recomendado para:
  - Requisitos plenamente claros e compreendidos
  - Definição do produto estável
  - Expertise na tecnologia
  - Projeto curto

Modelo Incremental

- Requisitos são divididos em subconjuntos
  - Vários ciclos de desenvolvimento
    - Várias e pequenas cascatas
  - Cada ciclo gera uma versão entregável do produto

Modelo Incremental



Modelo Incremental

- Vantagens:
  - Cada estágio gera um produto
  - Protótipos são entregues ao cliente
  - Resposta do cliente também é incremental
  - Flexibilidade
  - Facilidade de teste
  - Riscos são mais facilmente gerenciados

Modelo Incremental

- Desvantagens
  - Exige planejamento e projeto bons e rápidos
  - Exige clara e completa definição do sistema para sua subdivisão em ciclos
  - Custo total pode ser maior que o modelo cascata
  - Pode acabar se tornando um “conserta-e-entrega”
  - Pode despertar no cliente novos requisitos

Modelo Incremental

- Recomendado para:
  - Projetos com demanda de lançamento rápido
  - Novas tecnologias envolvidas
  - Escopo aberto

Metodologias Ágeis

Valores do manifesto ágil:

- Indivíduos e interações > processos e ferramentas
- Software funcional > documentação abrangente
- Colaboração do cliente > negociação de contratos
- Responder a mudanças > seguir um plano

Metodologias Ágeis

- Princípios:
  - Garantir a satisfação do cliente com entregas rápidas e contínuas de softwares funcionais
  - Mesmo mudanças tardias de escopo no projeto são bem-vindas para garantir a vantagem competitiva do cliente
  - Softwares funcionais são entregues frequentemente – em semanas, ao invés de meses
  - Projetos surgem através de indivíduos motivados, entre os quais existe relação de confiança

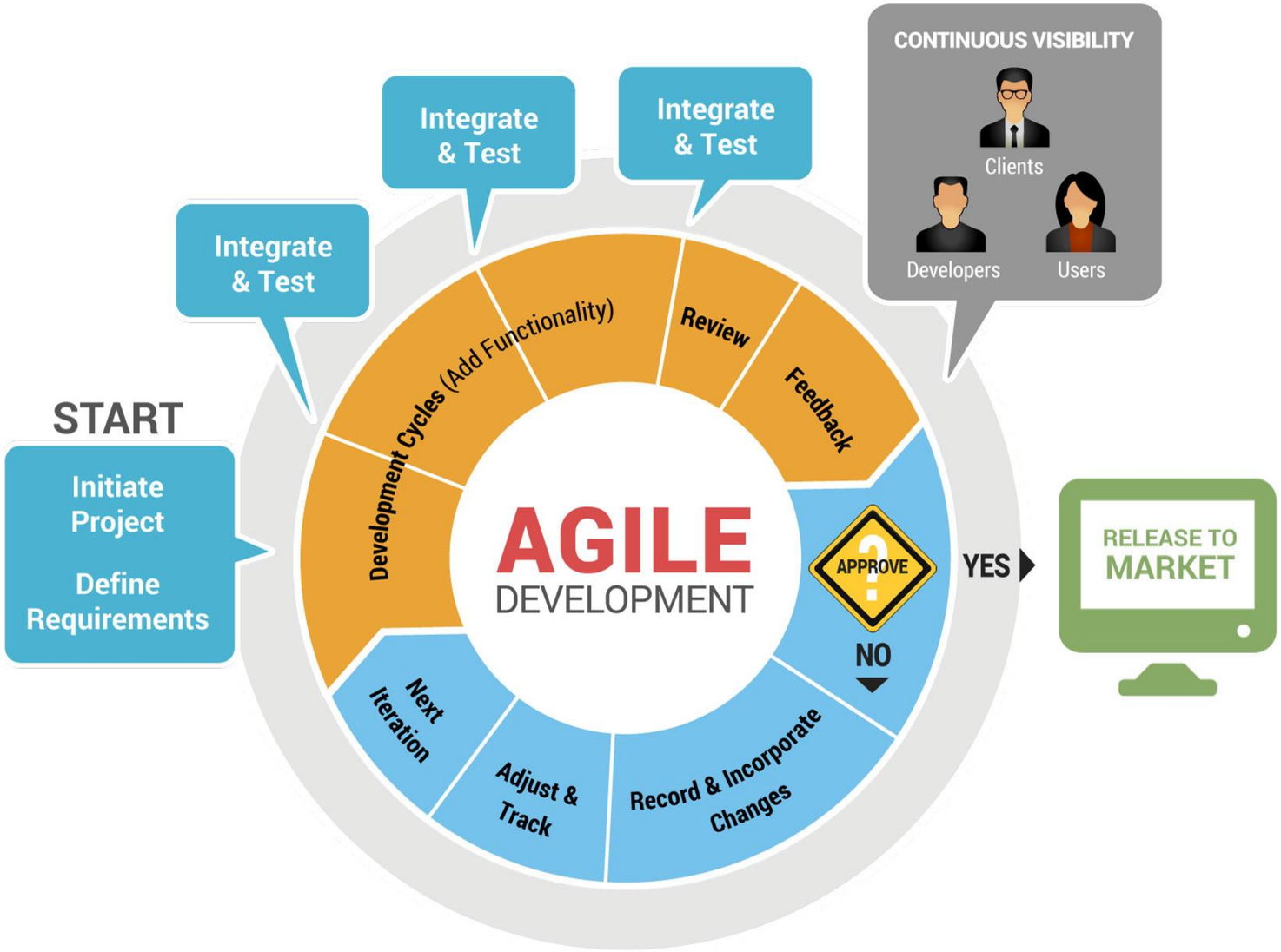
Metodologias Ágeis

- Princípios:
  - Transmitir informações através de conversas cara a cara
  - Softwares funcionais são a principal medida de progresso do projeto
  - Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
  - Projeto do software deve prezar pela excelência técnica

Metodologias Ágeis

- Princípios:
  - Simplicidade é essencial
  - As melhores arquiteturas, requisitos e projetos emergem de equipes automaticamente organizadas
  - A equipe reflete temporariamente como se tornar mais eficaz, sintonizando e ajusta seu comportamento apropriadamente
  - Cooperação diária entre pessoas que entendem do “negócio” e desenvolvedores

Metodologias Ágeis



Metodologias Tradicionais x Ágeis

Métrica	Tradicional	Ágil
Alicerces	Sistemas são totalmente especificáveis, previsíveis e podem ser construídos através de planejamento extensivo e meticuloso.	Softwares são naturalmente mutáveis e desenvolvimento de alto nível pode ser executado por equipes pequenas usando os princípios de melhorias contínuas de projeto através de retornos rápidos do cliente.
Controle	Centrado no processo	Centrado em pessoas
Estilo de gestão	Comando e controle	Liderança e colaboração

Metodologias Tradicionais x Ágeis

Métrica	Tradicional	Ágil
Gestão do conhecimento	Explícita	Tácita
Atribuição de papéis	Individual	Equipes auto-organizadas
Comunicação	Formal	Informal
Papel do cliente	Importante	Crítico
Ciclo do projeto	Guiado por tarefas e atividades	Guiado por recursos do produto
Modelo de desenvolvimento	Modelo de ciclo de vida	Modelo de evolução e entrega
Burocratização	Elevada	Flexível

O que levou aos métodos ágeis?

- Talvez não seja bem essa a pergunta correta.
- **Nos anos 90, entregar software conforme métodos ágeis sugerem não era sempre possível.**
  - Software era produto. Hoje, é serviço.
  - Software não era distribuído digitalmente.

Metodologias ágeis

- Software é um ativo mutável.

Metodologias Ágeis

- Prazos são cada vez mais curtos e, resultados, urgentes.

Metodologias ágeis

- Os resultados em TI (ainda) são vergonhosos.
  - Standish Group – CHAOS Report; Até o final dos anos 90, as falhas chegavam a 50%

MODERN RESOLUTION FOR ALL PROJECTS

	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011–2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.

Metodologias Ágeis

CHAOS RESOLUTION BY AGILE VERSUS WATERFALL

SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
All Size Projects	Agile	39%	52%	9%
	Waterfall	11%	60%	29%
Large Size Projects	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Medium Size Projects	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Small Size Projects	Agile	58%	38%	4%
	Waterfall	44%	45%	11%

The resolution of all software projects from FY2011–2015 within the new CHAOS database, segmented by the agile process and waterfall method. The total number of software projects is over 10,000.

Falácias da Engenharia de Software

Todos os requisitos são 100% conhecidos no início do projeto e foram minuciosa e corretamente levantados e detalhados.

O desenvolvedor sabe como construir todos eles.

Absolutamente nada irá mudar ao longo do caminho.

Verdades da Engenharia de Software

O cliente passa a compreender melhor sua necessidade conforme o projeto avança.

Assim, é desperdício de tempo e esforço detalhar minuciosamente os requisitos no início do projeto.

Muitas vezes (Sempre? Quase sempre?) os requisitos mudam, não necessariamente porque o cliente quer, mas porque a necessidade exige.

Software deve existir para agregar valor ao negócio.

Software deve agregar valor

Um processo rígido ou resistente a mudanças produz produtos medíocres. Os clientes podem até receber o que eles solicitaram primeiramente, mas é esse o produto que eles realmente querem logo quando eles o recebem? Coletando todos os requisitos no início e escrevendo-os sobre pedra, o produto é condenado a ser tão bom quanto a ideia inicial, ao invés de ser o melhor uma vez que as pessoas aprendem ou descobrem como fazer melhor.

- Jeff Sutherland

Então o certo é usar ágil. né?

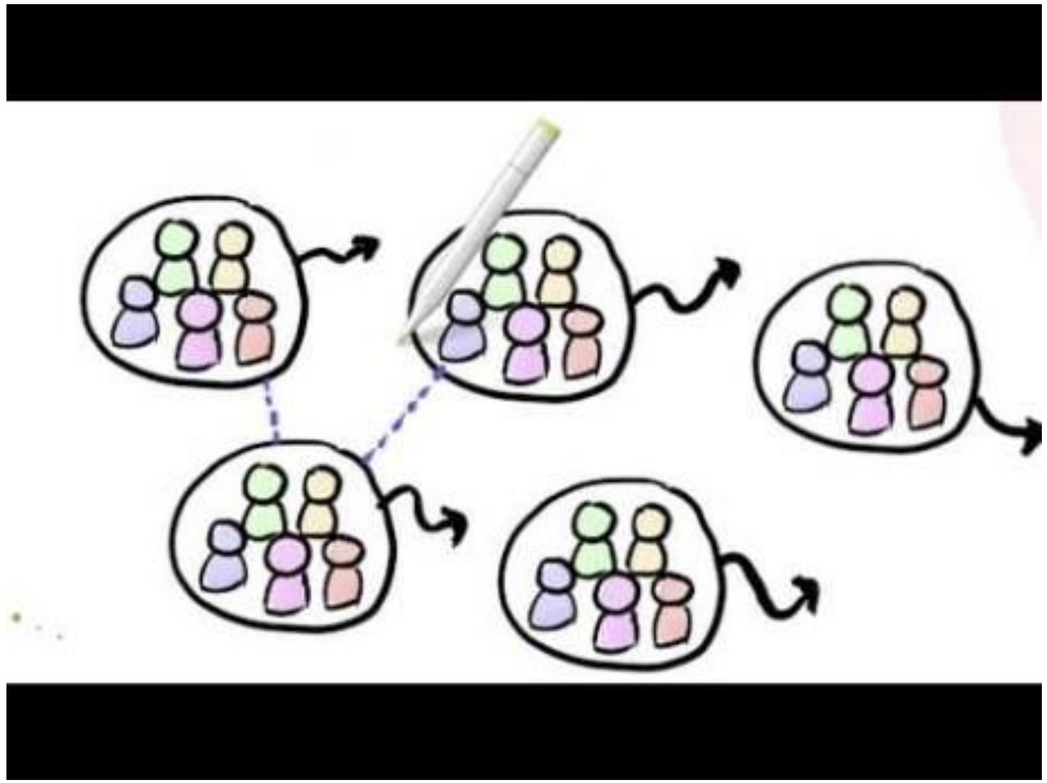
🤦‍♀️Não.

Fim.

Hibridização entre modelos

- Ser 100% ágil é como ser um unicórnio. 🦄
- **Na média, as empresas são quimeras.** 🐲🦁🐐
  - A agilidade ignora questões burocráticas hierárquicas sem as quais algumas empres não conseguem existir.
  - Agilidade não escala muito fácil.

Exemplo híbrido: o “modelo spotify”



IBMEC.BR

/IBMEC

IBMEC

@IBMEC_OFICIAL

@IBMEC

Ibmec