Gerenciamento de Projetos pelo PMBoK 6ª.Edição

Projetos de Software

SOFTWARE

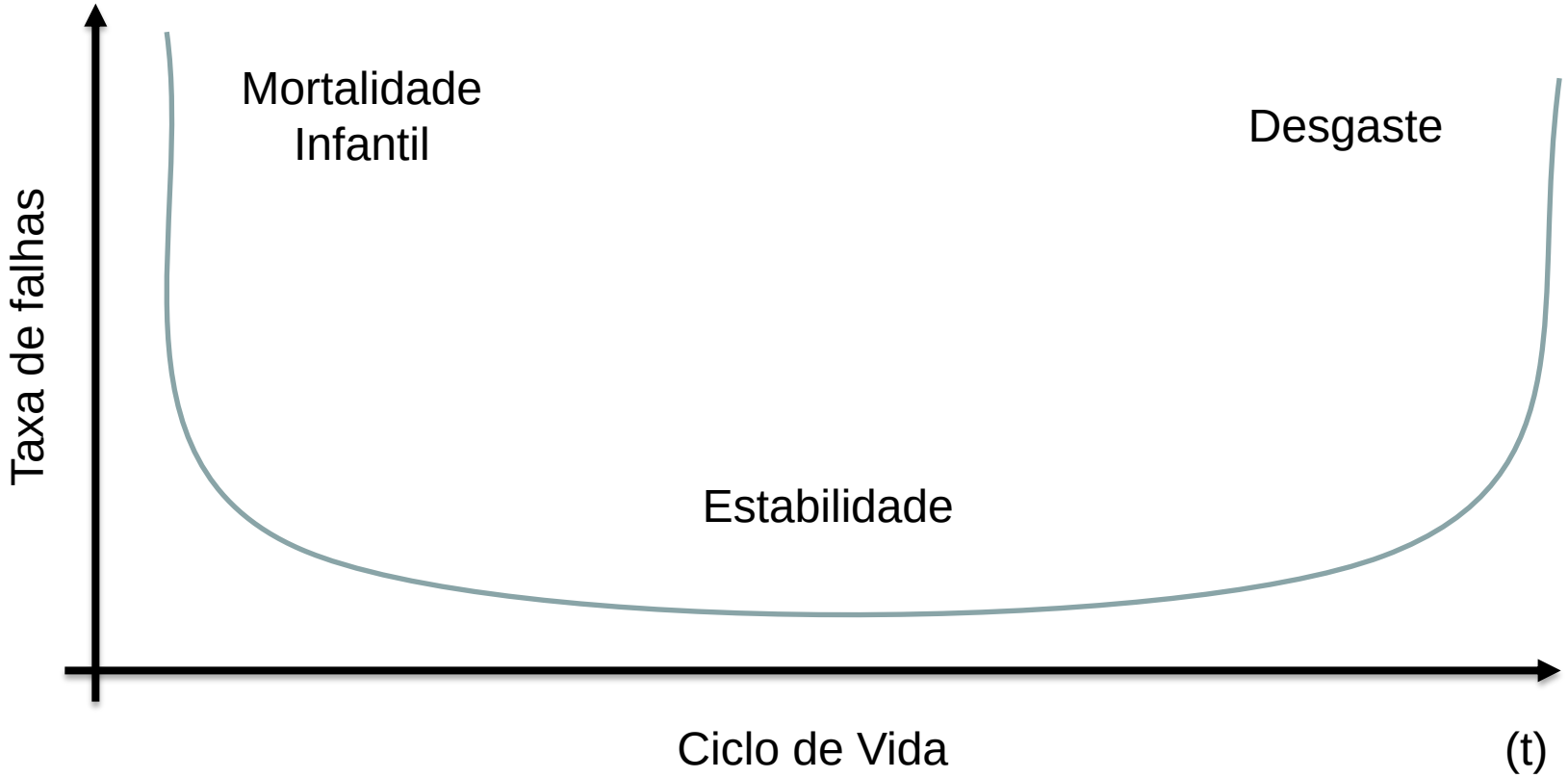
Estrutura

- INSTRUÇÕES que quando executadas produzem a função e o desempenho desejados
- ESTRUTURAS DE DADOS que possibilitam que os programas manipulem adequadamente a informação
- DOCUMENTOS que descrevem a operação e o uso dos programas

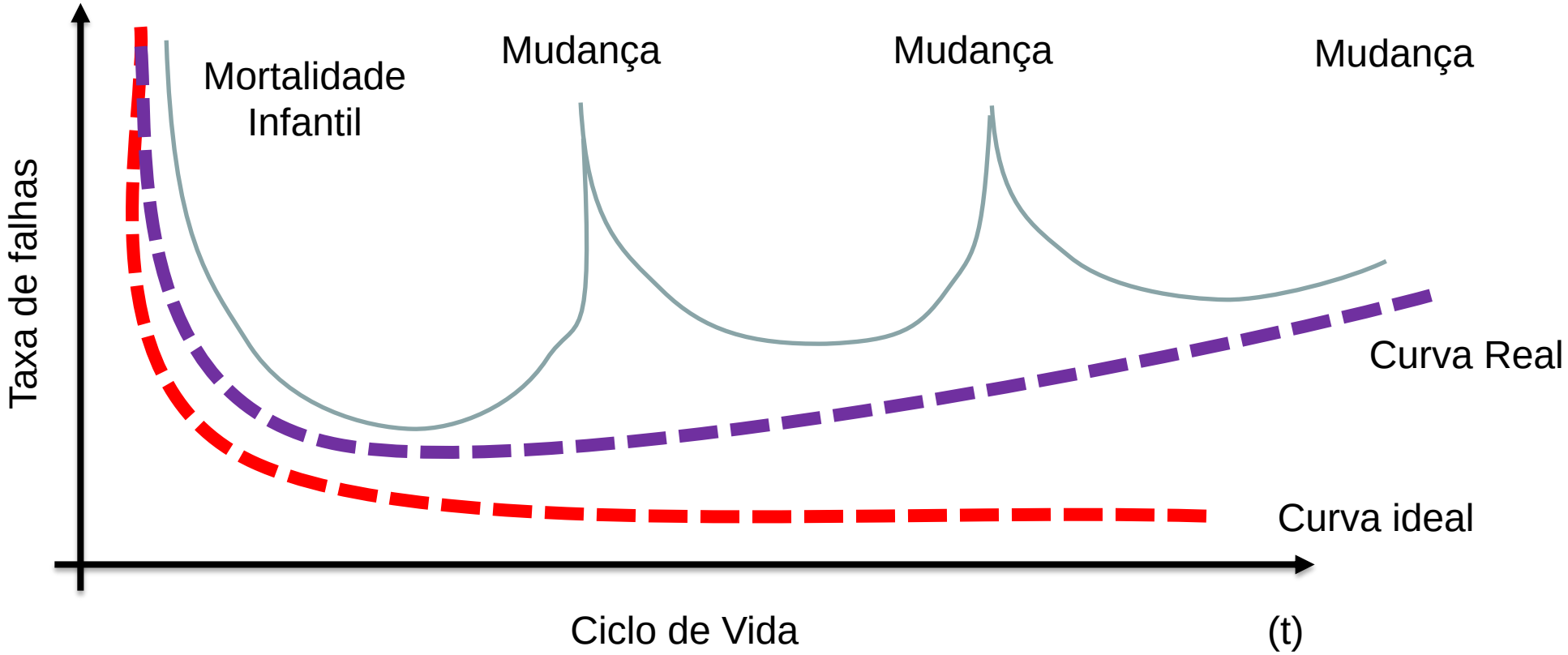
Características

- Desenvolvido ou projetado por engenharia
- Não manufaturado no sentido clássico
- Não se desgasta, mas se deteriora

Curva de confiabilidade - Hardware



Curva de confiabilidade - Software



O que são Projetos ?

"Um projeto é **um esforço temporário** empreendido **para criar um produto, serviço ou resultado exclusivo**." (PMI, 2017)

São realizados para cumprir um objetivo por meio da produção de entregas." (PMI, 2017)

Um objetivo é um resultado para o qual o trabalho é orientado como uma posição estratégica, um propósito a ser atingido, um produto a ser produzido ou um serviço a ser realizado. (PMI, 2017)

Uma entrega é definida como qualquer produto resultado ou capacidade únicos e verificáveis que pode ser produzido para entregar um produto, fase ou projeto. (PMI, 2017)

Porque realizar Projetos ?

REQUISITOS

Cumprir requisitos regulatórios, legais ou sociais

NECESSIDADES

Atender a pedidos ou necessidades de partes interessadas

Forças concorrenciais

Considerações ambientais

Problemas de materiais

Mudanças econômicas **Novas** *Melhorias em processos de negócios*

Solicitação do cliente

Requisitos legais **tecnologias** *OPORTUNIDADE ESTRATÉGICA*

Demandas de mercado Mudanças políticas *Necessidade social*

Demandas de partes interessadas

NECESSIDADE DE NEGÓCIO

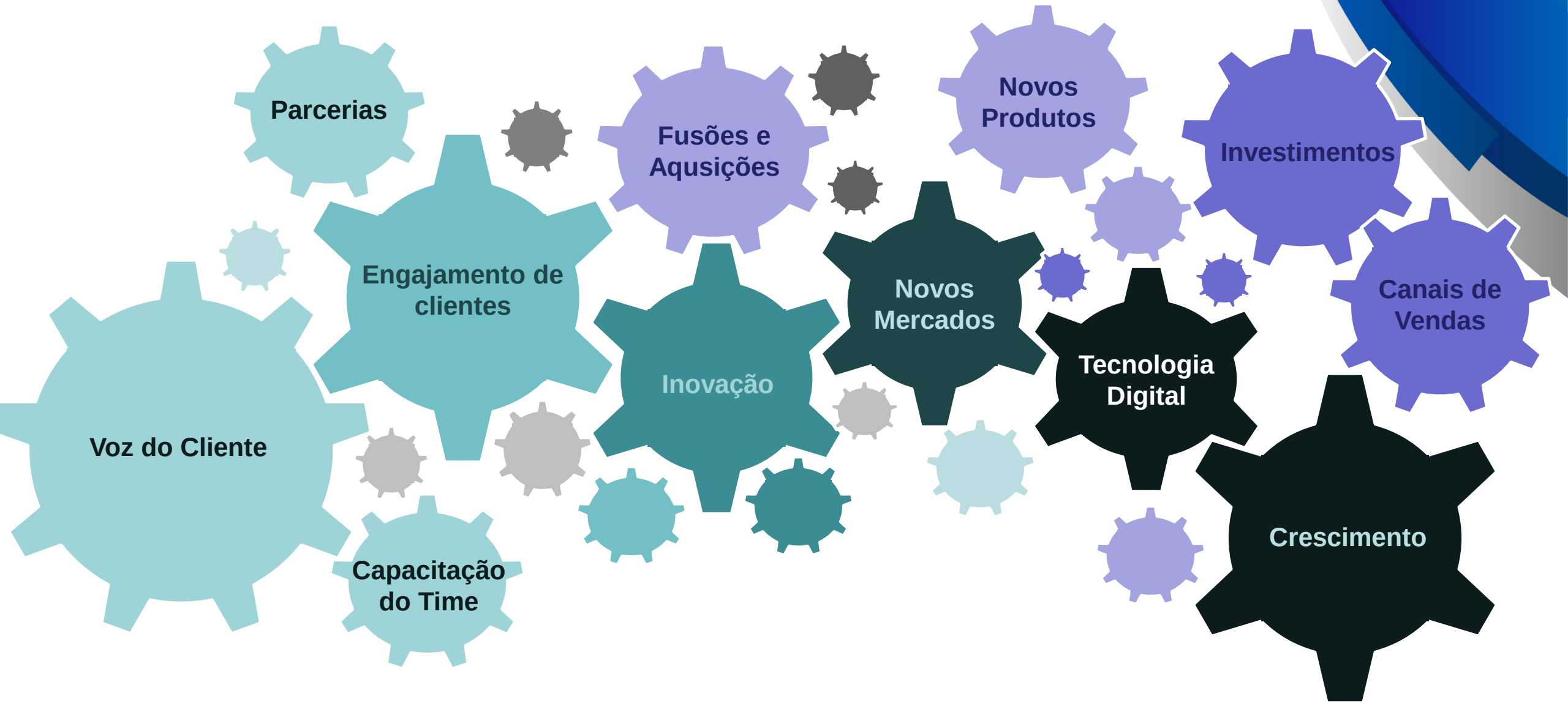
MELHORIA

Criar melhorar corrigir produtos, processos ou serviços

ESTRATÉGIA

Implementar ou alterar estratégias de negócio ou tecnológicas

Estratégia da organização

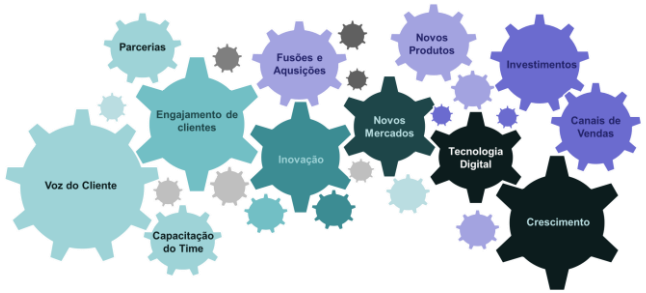


Gerenciamento de Projetos Organizacional (GPO)

- Portfólios, programas e projetos são alinhados por estratégias organizacionais e diferem da maneira como contribuem para a realização dos objetivos estratégicos.
- O gerenciamentos de portfólios alinha portfólios com estratégias organizacionais através da seleção de programas ou projetos corretos, priorização do trabalho e fornecimento dos recursos necessários.
- O gerenciamento de programas harmoniza seus componentes de programas e controla interdependências a fim de ralizar os benefícios especificados
- O gerenciamento de projetos permite os atingimento dos objetivos e metas organizacionais

Portfolio, Programas e Projetos

Estratégia da Organização



Portfólio da Organização

Programa A | **Programa B** | **Programa C**

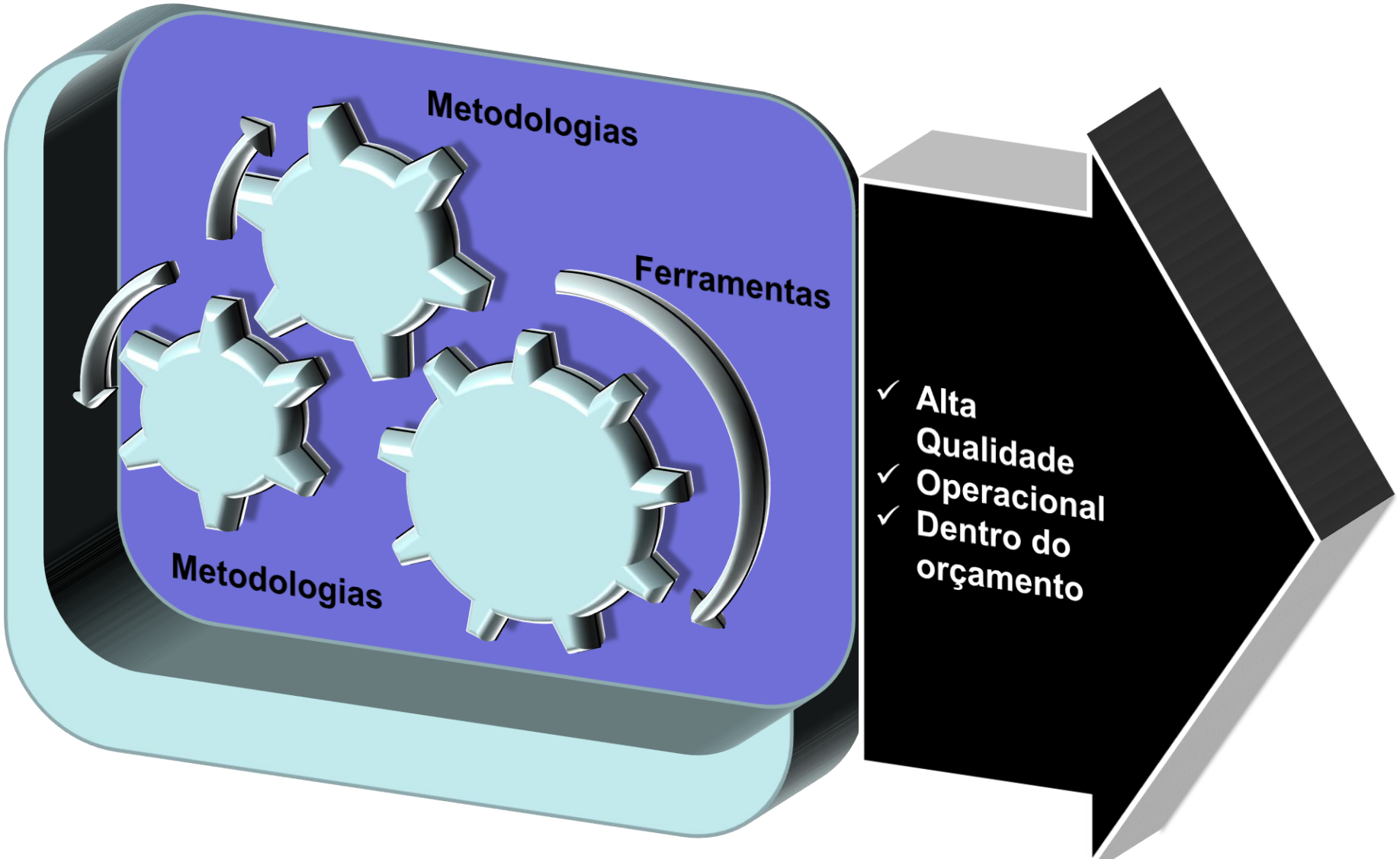
Projeto 1	Projeto 2	Projeto 3	Projeto 4	Projeto 5	Operações
Atividades	Atividades	Atividades	Atividades	Atividades	Atividades
Atividades		Atividades	Atividades	Atividades	Atividades
Atividades	Atividades	Atividades	Atividades	Atividades	

Recursos compartilhados pelas Partes Interessadas

Por que é tão difícil desenvolver um bom software?

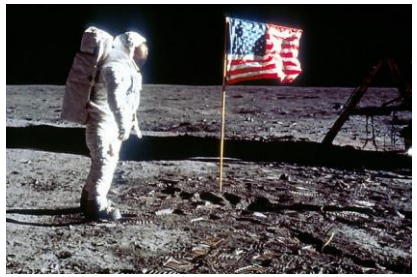
Porque Engenharia de Software ?

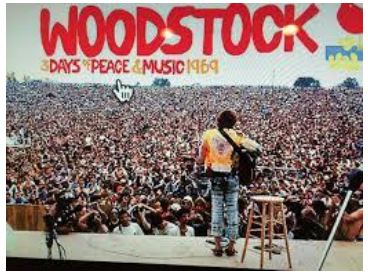
Estrutura da Engenharia de Software

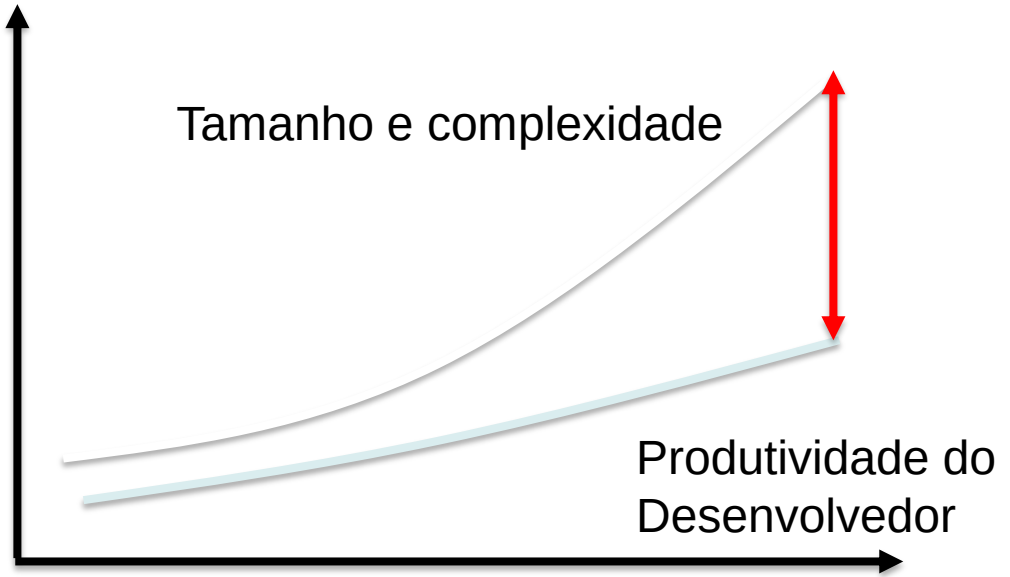


Uma perspectiva histórica

- Anos 60 e 70:
  - Homem na Lua
  - Festival de Wodstock
  - Máquina Polaroid
- A crise do software:
  - De HW para SW
  - Demanda crescente
  - Tamanho e complexidade crescente
  - Esforço de desenvolvimento
  - Produtividade







A Crise do Software (Nakagawa, 2016)

Efeitos

- Estimativas de prazo e de custo frequentemente são imprecisas
- Insatisfação do cliente com o sistema concluído
- Qualidade de software às vezes é menos que adequada
- Software existente é muito difícil de manter

Causas

- PRÓPRIO CARÁTER DO SOFTWARE
- FALHAS DAS PESSOAS RESPONSÁVEIS PELO DESENVOLVIMENTO DE SOFTWARE
- MITOS DO SOFTWARE

Mitos administrativos

MITO

Manual

- Já temos um manual repleto de padrões e procedimentos para a construção de software.
- Isso não oferecerá ao meu pessoal tudo o que eles precisam saber?

Ferramentas

- Meu pessoal tem ferramentas de desenvolvimento de software de última geração.

Realidade

Manual

- Será que o manual é usado?
- Os profissionais sabem que ele existe?
- Ele reflete a prática moderna de desenvolvimento de software?
- Ele é completo?

Ferramentas

- *É preciso muito mais do que os mais recentes computadores e ferramentas para se fazer um desenvolvimento de software de alta qualidade.*

Mitos administrativos

MITO

- Se nós estamos atrasados nos prazos, podemos adicionar mais programadores e tirar o atraso

Realidade

- *O desenvolvimento de software não é um processo mecânico igual à manufatura. Acrescentar pessoas em um projeto torna-o ainda mais atrasado.*
- *Pessoas podem ser acrescentadas, mas somente de uma forma planejada.*

Mitos Cliente

MITO

- Uma declaração geral dos objetivos é suficiente para se começar a escrever programas. Podemos preencher os detalhes mais tarde.

Realidade

- *Uma definição inicial ruim é a principal causa de fracassos dos esforços de desenvolvimento de software.*
- *É fundamental uma descrição formal e detalhada do domínio da informação, função, desempenho, interfaces, restrições de projeto e critérios de validação.*

Mitos Cliente

MITO

- Os requisitos de projeto modificam-se continuamente, mas as mudanças podem ser facilmente acomodadas, porque o software é flexível.

Realidade

- Uma mudança, quando solicitada tardiamente num projeto, pode ser maior do que a ordem de magnitude mais dispendiosa da mesma mudança solicitada nas fases iniciais.

FASES	CUSTO DE MANUTENÇÃO
DEFINIÇÃO	1 x
DESENVOLVIMENTO	1.5 - 6x
MANUTENÇÃO	60 - 100x

Mitos Profissional

MITO

- Assim que escrevermos o programa e o colocarmos em funcionamento, nosso trabalho estará completo.
- Enquanto não tiver o programa "funcionando", eu não terei realmente nenhuma maneira de avaliar sua qualidade.

Realidade

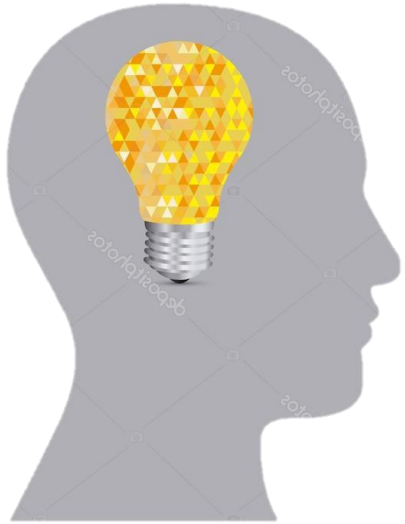
- *Os dados da indústria indicam que entre 50 e 70% de todo esforço gasto num programa serão despendidos depois que ele for entregue pela primeira vez ao cliente.*
- Um programa funcionando é somente uma parte de uma Configuração de Software que inclui todos os itens de informação produzidos durante a construção e manutenção do software.

Evidências da Crise

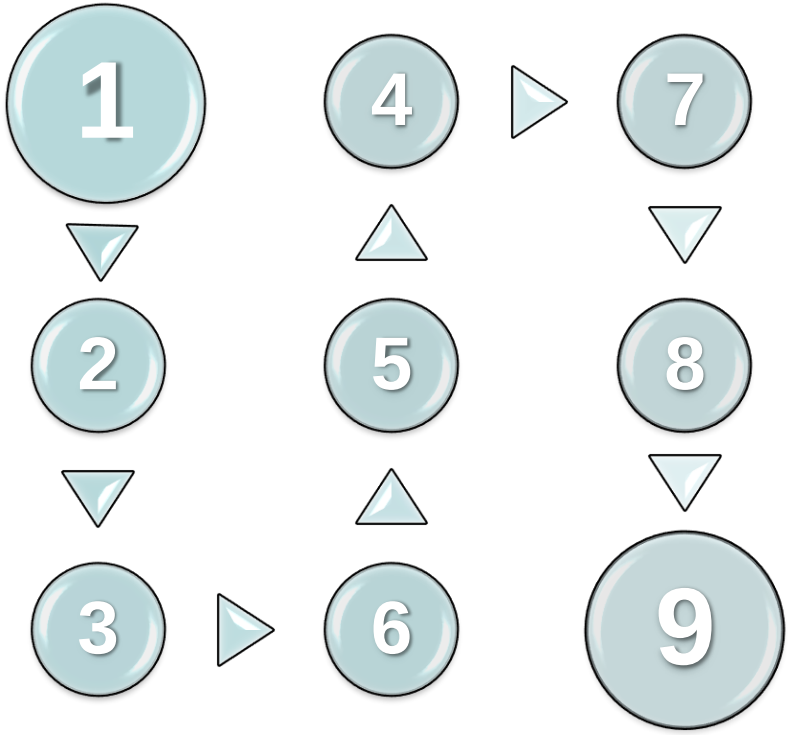
- **Estudo de Davis (1990)**
  - 9 contratos para desenvolvimento de software
  - Valor total US$ 7.0 mi
  - Resultado
    1. Software usado como entregue
    2. Software usado após mudanças
    3. **Software usado após MUITAS mudanças**
    4. **Software não usado mas entregue**
    5. **Software não entregue**

  - **Dos 9 contratos um valor de US$ 5.0 mi estavam nas condições 3, 4 ou 5.**
- **Conferências da NATO (1969) concluem que identificar e difundir as melhores prática no desenvolvimento de software seriam um componente fundamental para a resolução da crise do software**

Da ideia até o produto



Ideia abstrata

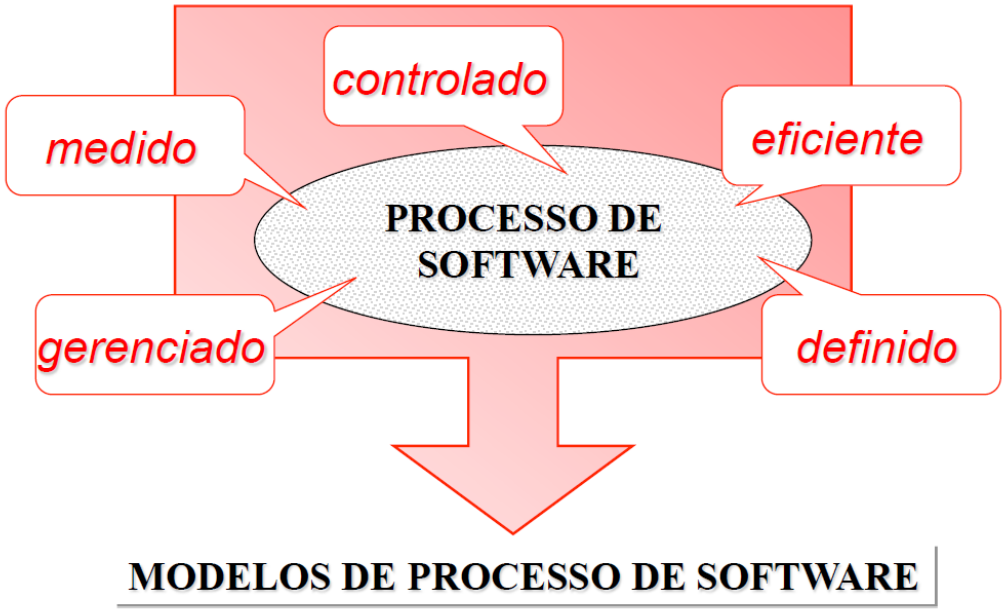


Processo Sistemático
e Formal

Produto Final

Definição Engenharia de Software

- A aplicação de uma abordagem sistemática, disciplinada e possível de ser medida para o desenvolvimento, operação e manutenção do software (IEEE)



Fases genéricas

Definição – O que será desenvolvido

- que informação vai ser processada
- que função e desempenho são desejados
- que comportamento pode ser esperado do sistema
- que restrições de projeto existem
- que critérios de validação são exigidos para definir um sistema bem sucedido

Desenvolvimento – Como será desenvolvido

- como os dados vão ser estruturados
- como a função vai ser implementada como uma arquitetura de software
- como o projeto será traduzido em uma linguagem de programação
- como os testes serão efetuados

Manutenção – Mudanças que irão ocorrer após o desenvolvimento

- A fase de manutenção reaplica os passos das fases de definição e desenvolvimento, mas faz isso no contexto de um software existente.

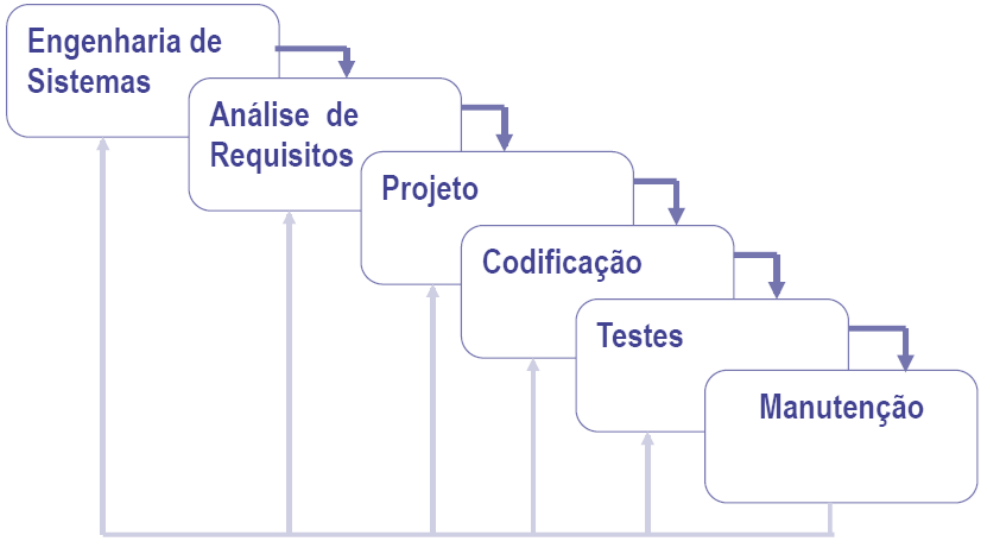
Apoio

- Garantia de Qualidade de Software
- Gerenciamento de Configuração de Software
- Preparação e Produção de Documentos
- Gerenciamento de Reusabilidade
- Medidas

Modelos de Processo de Software

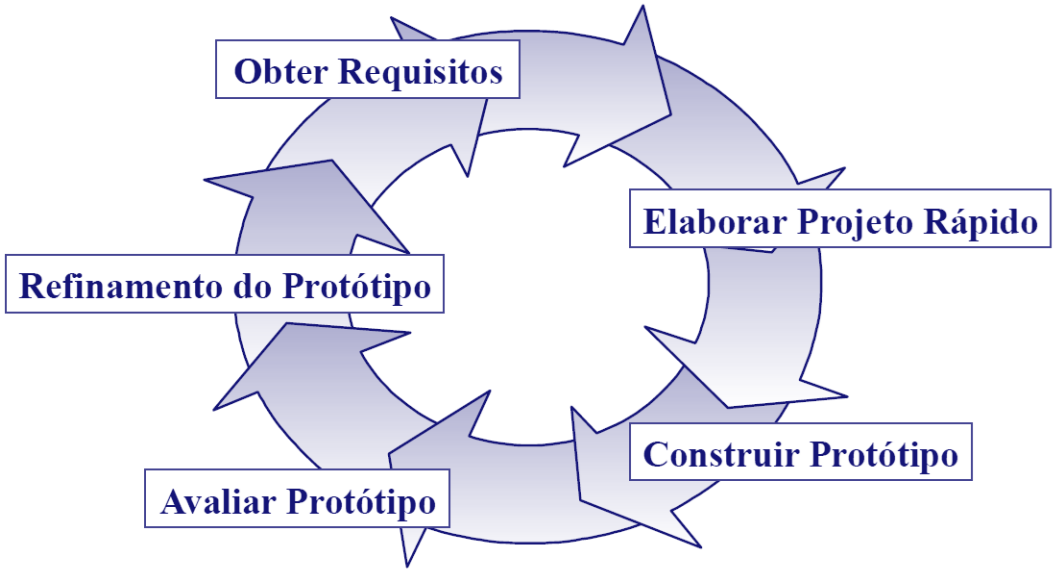
- Modelo Sequencial Linear (também chamado Ciclo de Vida Clássico ou Modelo Cascata)
- Paradigma de Prototipação
- O Modelo RAD (Rapid Application Development)
- Modelos Evolutivos de Processo de Software
  - O Modelo Incremental
  - O Modelo Espiral
  - O Modelo de Montagem de Componentes

Modelo cascata



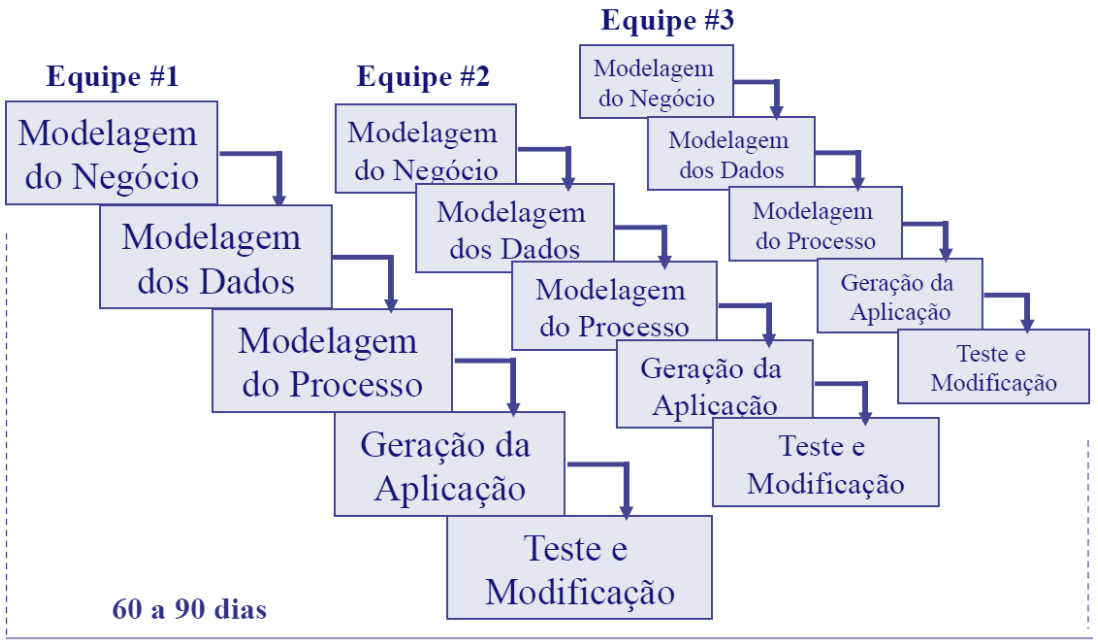
- Projetos reais raramente seguem o fluxo seqüencial que o modelo propõe
- Logo no início é difícil estabelecer explicitamente todos os requisitos. No começo dos projetos sempre existe uma incerteza natural
- O cliente deve ter paciência. Uma versão executável do software só fica disponível numa etapa avançada do desenvolvimento

Paradigma de Prototipação



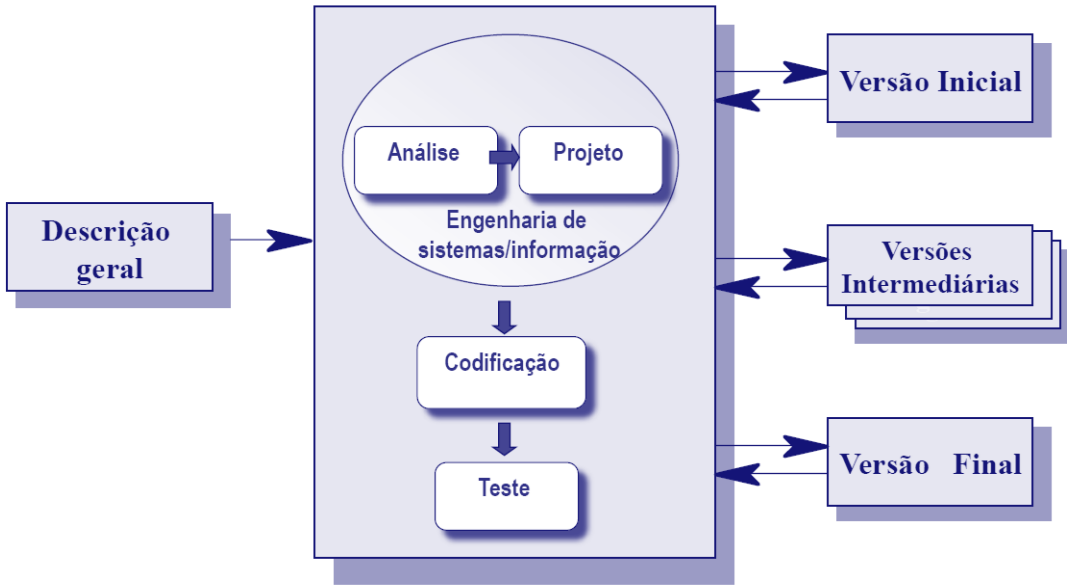
- o objetivo é entender os requisitos do usuário e, assim, obter uma melhor definição dos requisitos do sistema.
- possibilita que o desenvolvedor crie um modelo (protótipo)do software que deve ser construído
- apropriado para quando o cliente não definiu detalhadamente os requisitos.
- ainda que possam ocorrer problemas, a prototipação é um ciclo de vida eficiente.
- a chave é definir-se as regras do jogo logo no começo.
- o cliente e o desenvolvedor devem ambos concordar que o protótipo seja construído para servir como um mecanismo a fim de definir os requisitos

Modelo RAD (*Rapid Application Development*)



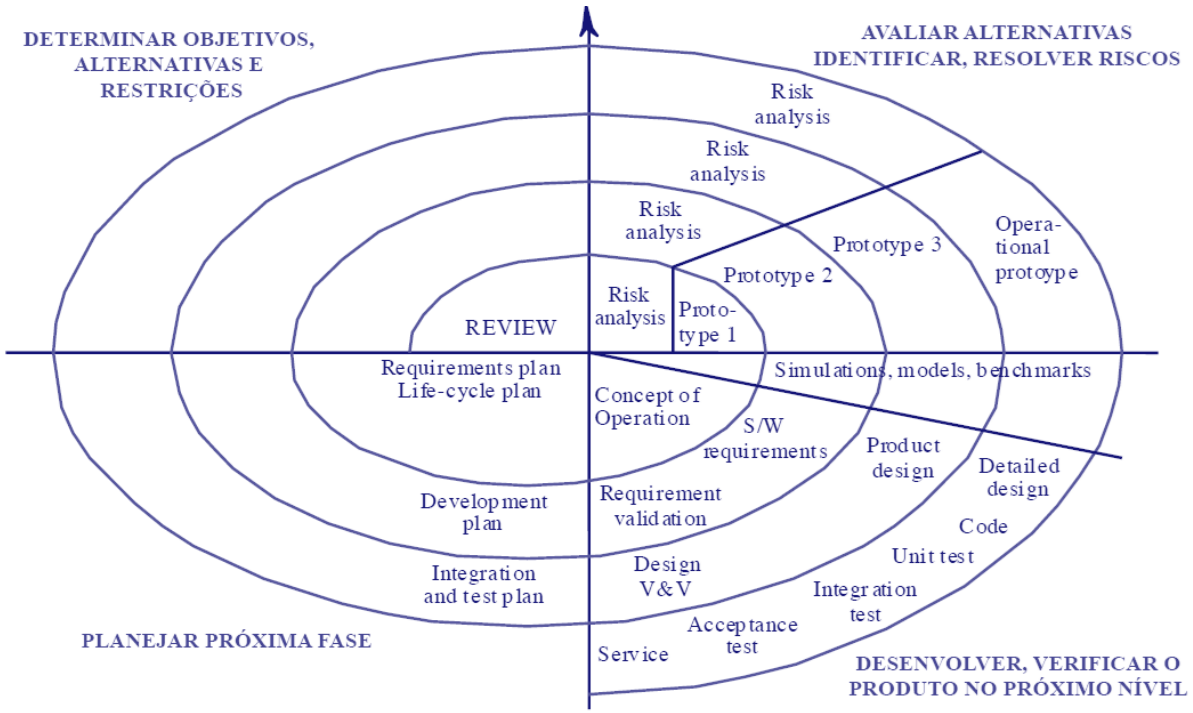
- RAD (Rapid Application Development) é um modelo sequencial linear que enfatiza um ciclo de desenvolvimento extremamente curto
- O desenvolvimento rápido é obtido usando uma abordagem de construção baseada em componentes.
- Exige recursos humanos suficientes para todas as equipes
- Exige que desenvolvedores e clientes estejam comprometidos com as atividades de “fogo-rápido” a fim de terminar o projeto num prazo curto

Modelos Evolutivos - Incremental



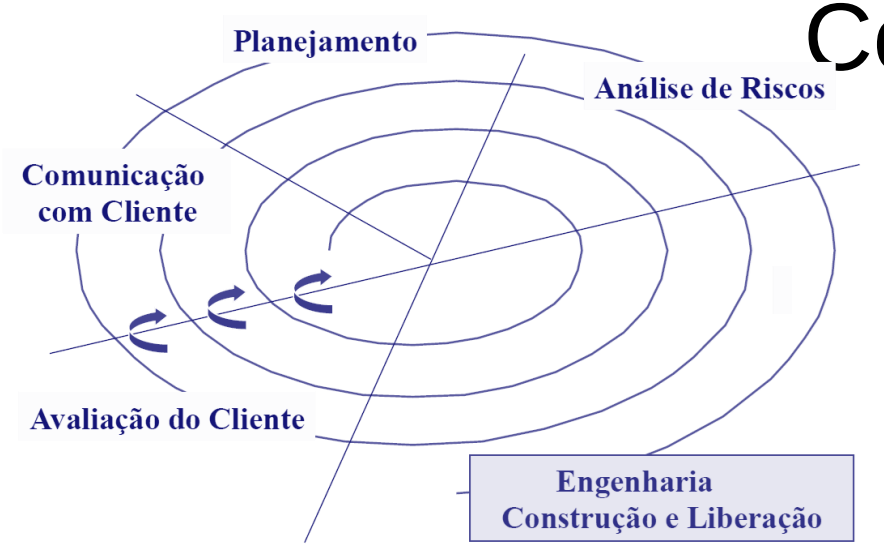
- o modelo incremental combina elementos do modelo cascata (aplicado repetidamente) com a filosofia iterativa da prototipação
- o objetivo é trabalhar junto do usuário para descobrir seus requisitos, de maneira incremental, até que o produto final seja obtido.
- o modelo incremental é mais apropriado para sistemas pequenos
- As novas versões podem ser planejadas de modo que os riscos técnicos possam ser administrados (Ex. disponibilidade de determinado hardware)

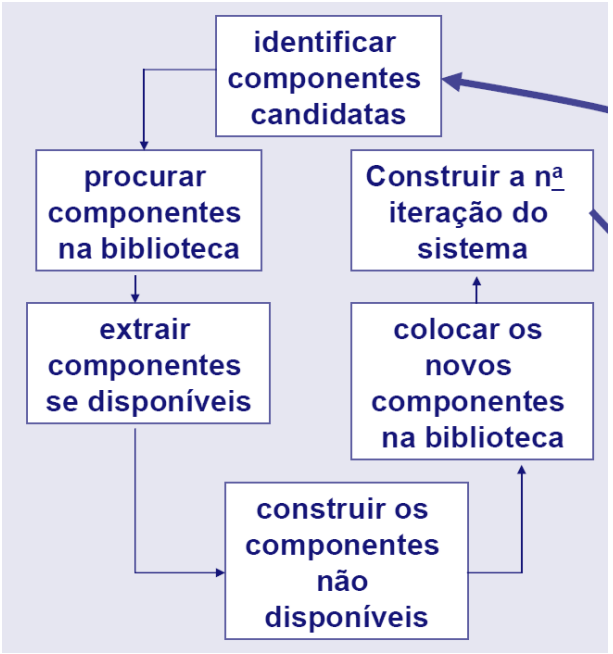
Modelo Evolutivo - Espiral



- O modelo espiral acopla a natureza iterativa da prototipação com os aspectos controlados e sistemáticos do modelo cascata.
- O modelo espiral é dividido em uma série de *atividades de trabalho* ou *regiões de tarefa*.
- Existem tipicamente de 3 a 6 regiões de tarefa
- engloba as melhores características do ciclo de vida Clássico e da Prototipação, adicionando um novo elemento: a *Análise de Risco*
- é, atualmente, a abordagem mais realística para o desenvolvimento de software em grande escala.
- usa uma abordagem que capacita o desenvolvedor e o cliente a entender e reagir aos riscos em cada etapa evolutiva
- pode ser difícil convencer os clientes que uma abordagem "evolutiva" é controlável

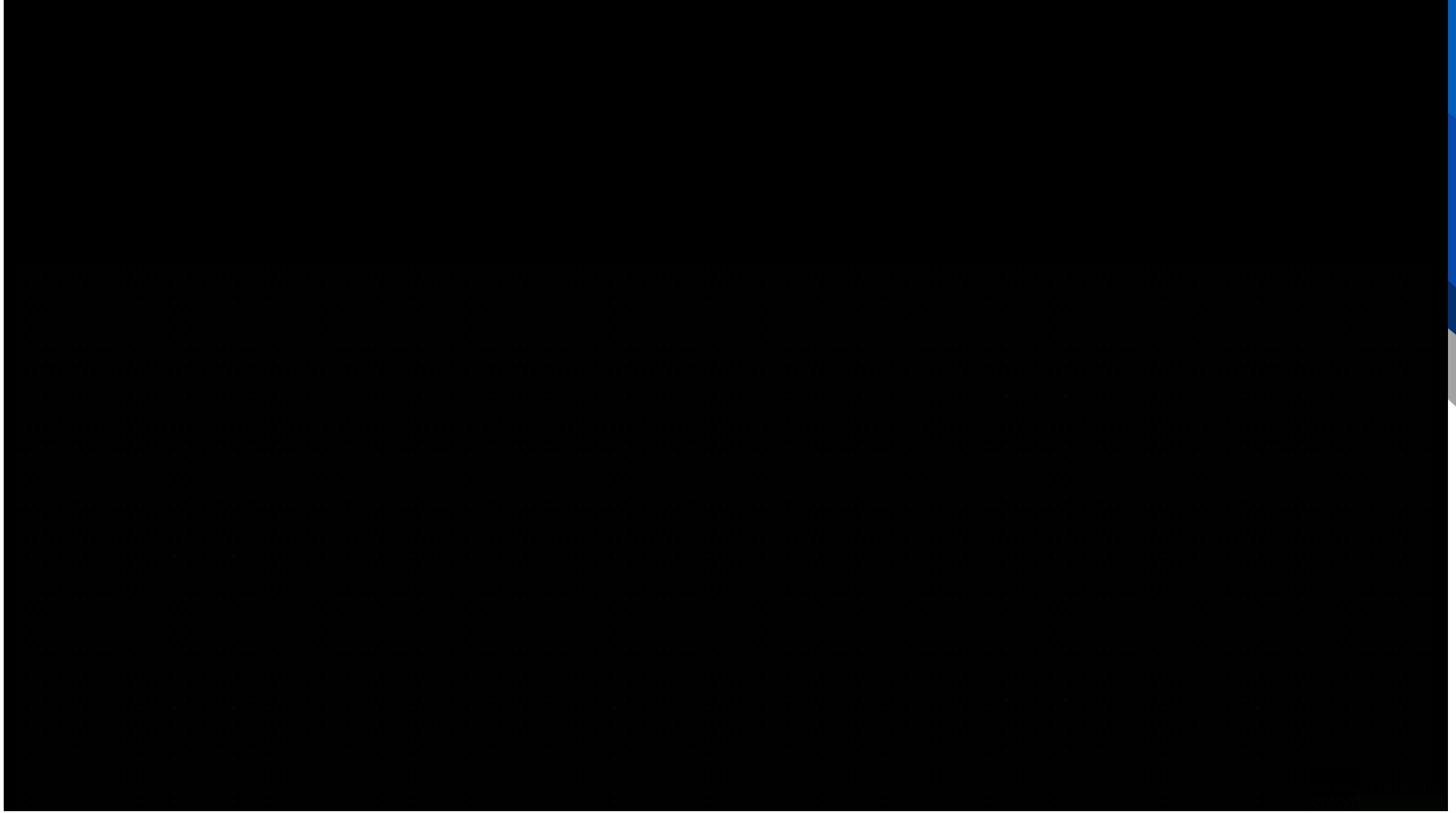
O Modelo de Montagem de Componentes





- Utiliza tecnologias orientadas a objeto
- Quando projetadas e implementadas apropriadamente as classes orientadas a objeto são reutilizáveis em diferentes aplicações e arquiteturas de sistema
- O modelo de montagem de componentes incorpora muitas das características do modelo espiral.
- O modelo de montagem de componentes conduz ao reuso do software
- a reusabilidade fornece uma série de benefícios:
- redução de 70% no tempo de desenvolvimento
- redução de 84% no custo do projeto
- índice de produtividade de 50% mais que o normal da indústria
- esses resultados dependem da robustez da biblioteca de componentes


TED Ideas worth spreading
WATCH
DISCOVE
Share
Add to list
Like
Recommend
0
Orient
Plan
18
Minutes
Ta-Da!
TED
Tom Wujec | TED2010
Tom Wujec: Construa uma torre, construa uma equipe
6:37



O que podemos aprender com esse vídeo

Link para o Video

https://www.ted.com/talks/tom_wujec_build_a_tower_build_a_team?language=pt-br