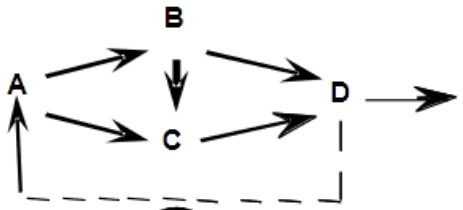
Processos de Software

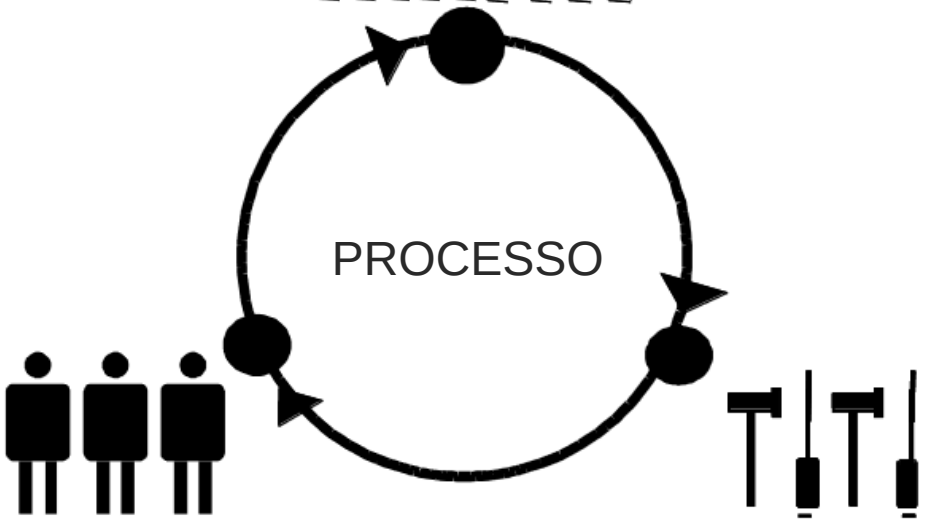
Eduardo Figueiredo

http://www.dcc.ufmg.br/~figueiredo

Processos

Procedimentos e métodos definindo relação entre tarefas



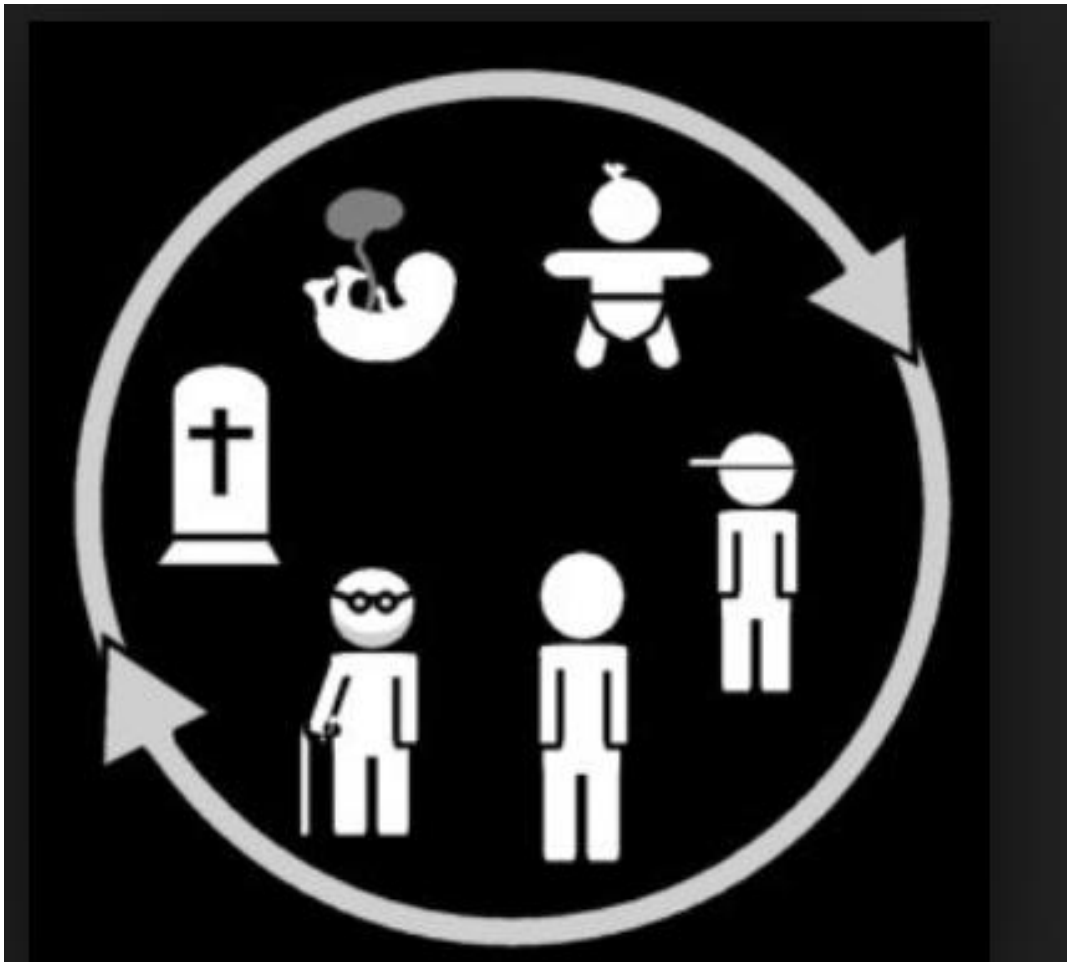


Pessoas com habilidades, treinadas e motivadas

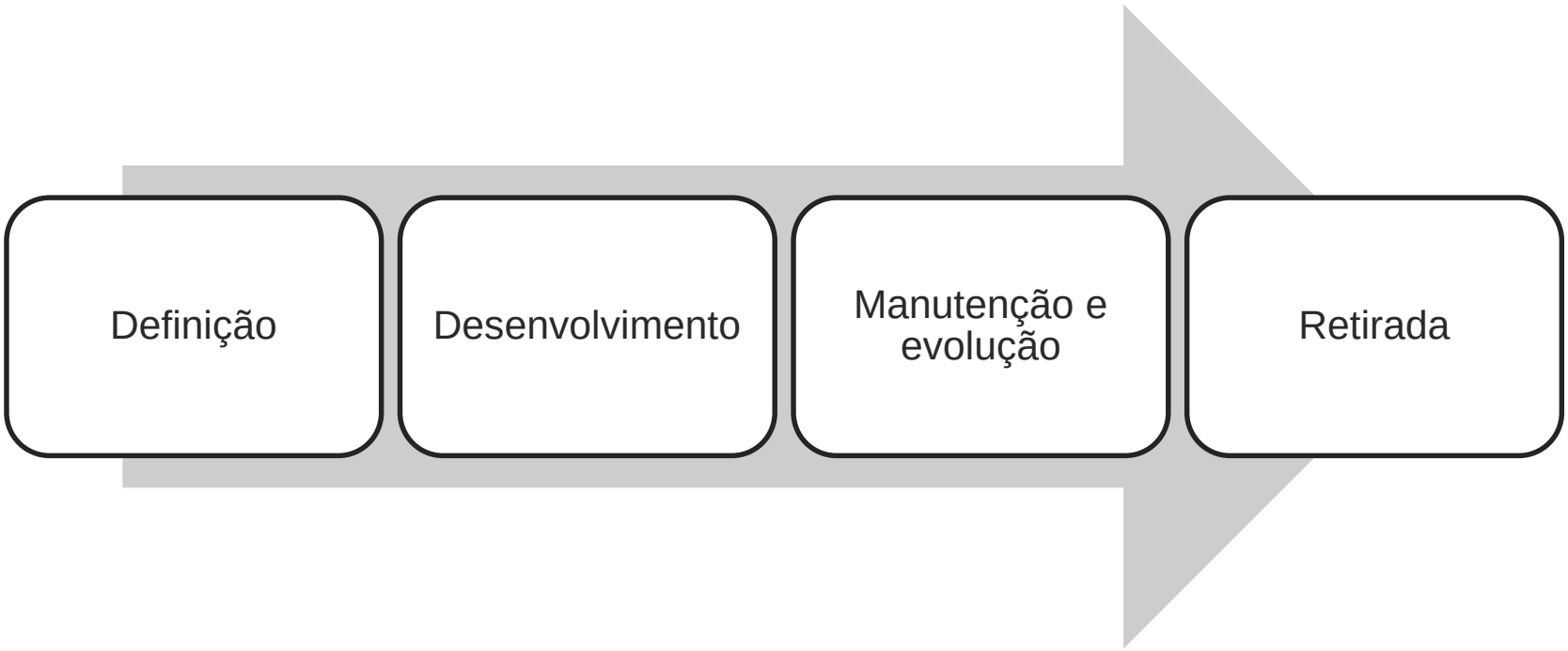
Ferramentas e equipamentos

Figura adaptada de “CMMI for Development 1.3 – CMMI Institute”

Ciclo de Vida



Ciclo de Vida do Software



Processos de Software

Um PROCESSO DE SOFTWARE é um conjunto de atividades relacionadas que leva a produção de um produto de software

Atividades e processos

- Atividades e processos são compostos por
  - Produtos/Artefatos/Produtos de trabalho – Resultados de uma atividade
    - Ex: Projeto arquitetural do software
  - Papéis – Responsabilidades das pessoas envolvidas no processo
    - Ex: Arquiteto de software
  - Pré/Pós-Condições – Condições que devem ser verdadeiras antes ou depois da execução ou conclusão de uma atividade
    - Ex: A atividade de “definição de arquitetura” só pode ser iniciada quando todos os requisitos tiverem sido definidos.

Principais atividades

- Especificação de Software
- Projeto e implementação de Software
- Validação de Software
- Evolução de Software

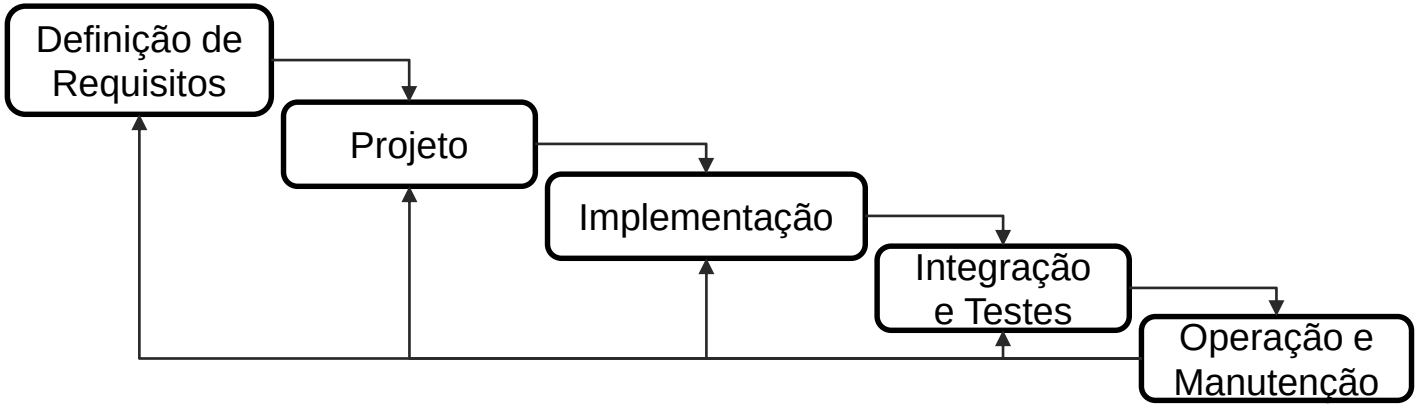
- Essas atividades são complexas
  - Contêm subatividades ou subprocessos
- Atividades de apoio também são importantes
  - Ex: Gerência de configuração, documentação

Modelos de Processo de Software

- Um modelo de processo de software é uma representação simplificada de um processo de software
- Frameworks ou modelos genéricos que podem ser estendidos ou adaptados para criar processos de software mais específicos
- Exemplos:
  - Cascata ou Waterfall
  - Desenvolvimento Incremental
  - Desenvolvimento orientado a reutilização

Cascata ou Waterfall

- Atividades sequenciais
- Uma fase deve ser concluída para a outra começar



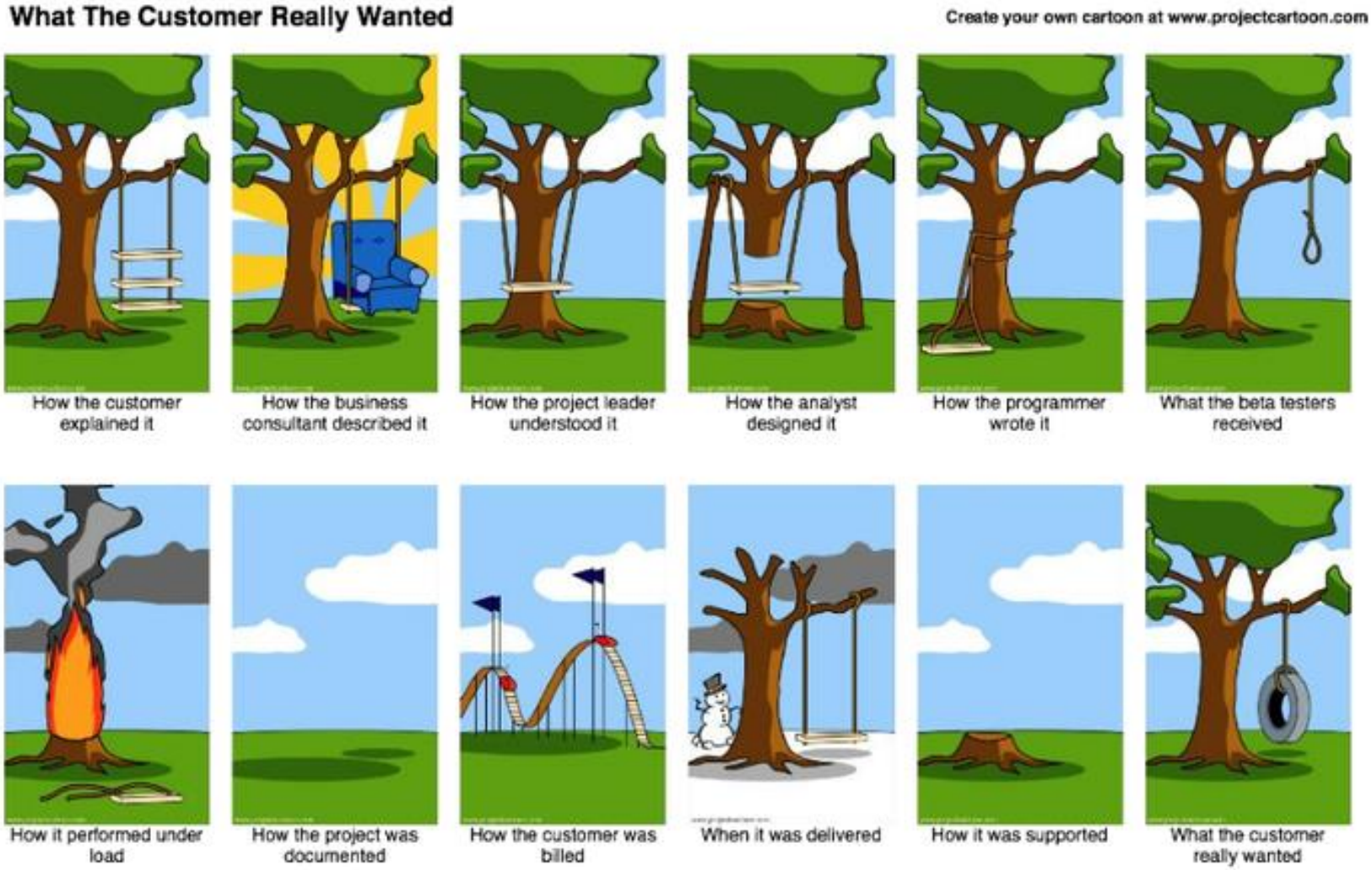
Características

- Modelo “orientado a planos”
  - Planejar e definir cronogramas para cada atividade antes de executá-las
  - Documentação rígida (idealmente completa) em cada atividade
- Reflete abordagens adotadas em outras engenharias
- Simplicidade do modelo torna a gestão do projeto mais simples
  - É mais fácil acompanhar o andamento do projeto baseado na conclusão de cada etapa.

Problemas

- Reação a mudanças difícil e lenta
- Alto custo de produzir e aprovar documentos
- Desenvolvimento de software raramente segue um fluxo linear
  - Erros de uma fase muitas vezes são percebidos em fases seguintes
- Requer que os requisitos sejam plenamente conhecidos no inicio do projeto
- Versão final só ficará pronta na fase final do projeto

Ops..

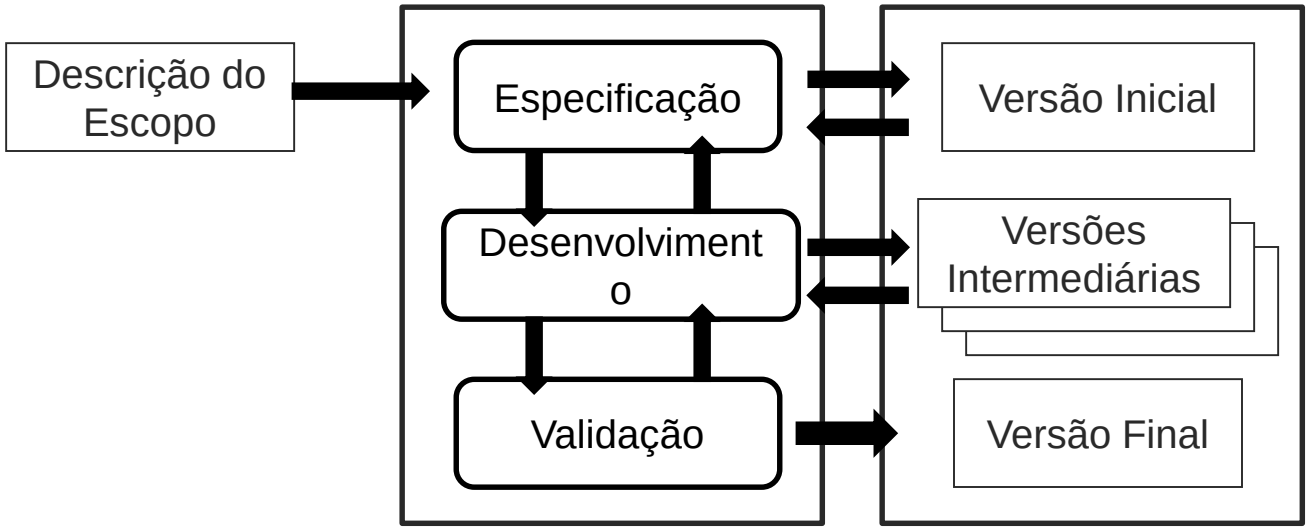

What The Customer Really Wanted
Create your own cartoon at www.projectcartoon.com
How the customer explained it
How the business consultant described it
How the project leader understood it
How the analyst designed it
How the programmer wrote it
What the beta testers received
How it performed under load
How the project was documented
How the customer was billed
When it was delivered
How it was supported
What the customer really wanted

Quando utilizar

- Quando os requisitos são bem conhecidos
- Quando há baixa probabilidade de mudança dos requisitos
- Sistemas críticos

Desenvolvimento Incremental

- Desenvolvimento de pequenos incrementos (versões parciais do software).
- Atividades intercaladas



Características

- Reflete a forma que resolvemos problemas
- Desenvolvimento incremental é uma parte fundamental de abordagens ágeis
  - No entanto, pode ser “orientado a planos” ou um meio termo.
- Cada incremento do Sistema incorpora funcionalidades desejadas pelo cliente
- Principais benefícios
  - Menor custo decorrente de mudanças de requisitos de cliente
  - Mais fácil obter feedback do cliente sobre o desenvolvimento já realizado
  - É possível a entrega e implantação de software ÚTIL ao cliente mais rapidamente

Problemas

- O processo pode não ser muito claro
  - Difícil acompanhar progresso
- A gerência do software é complicada
  - O sistema não é necessariamente especificado de forma completa à priori
- Risco de degradação da estrutura do produto com a adição de incrementos
  - O produto final pode se tornar mal estruturado
- Esses problemas são agravados no desenvolvimento de sistemas grandes, complexos e com longo tempo de vida.

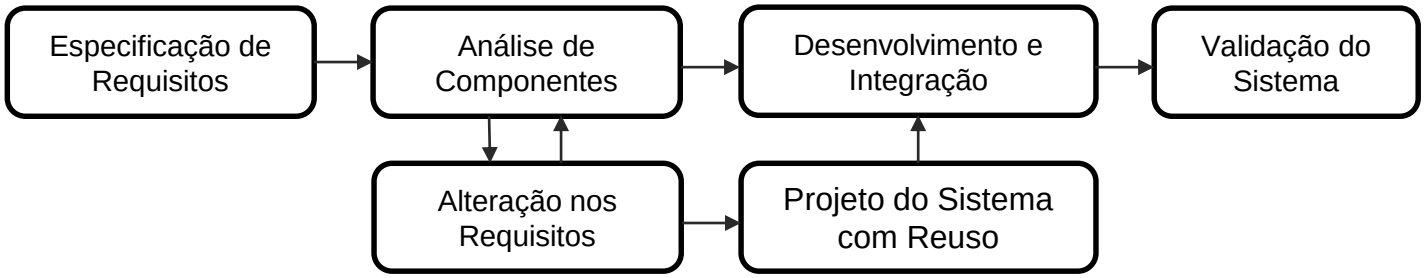
Degradação estrutural

Processo Orientado ao Reuso

- Baseia-se na existência de um número significativo de componentes reusáveis
- O processo se concentra na integração dos componentes reusáveis
- Inspirado na analogia com componentes de hardware
  - Exemplo: componentes elétricos / eletrônicos

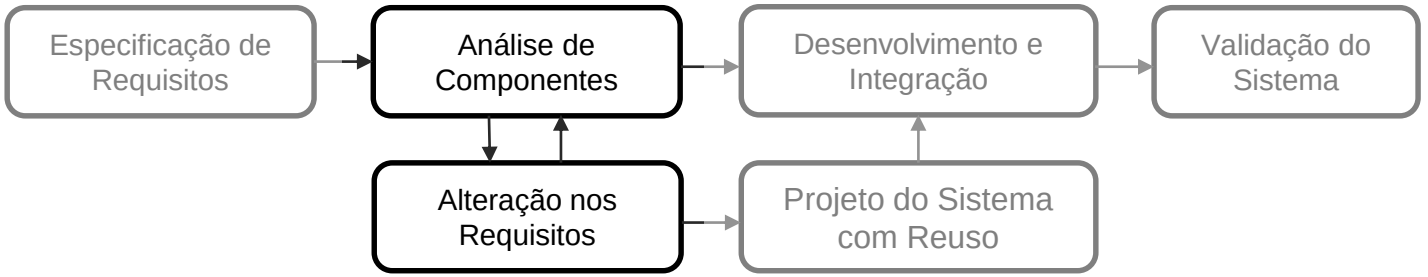
Representação

- Baseia-se na existência de um número significativo de componentes reusáveis
- O processo se concentra na integração dos componentes



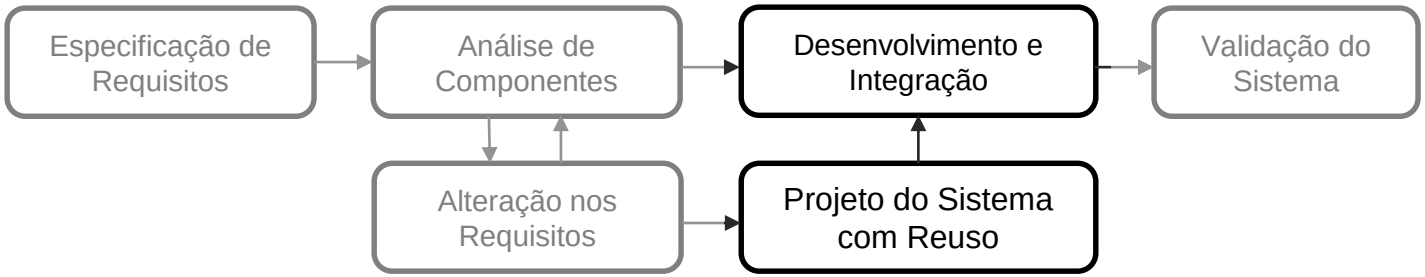
Alinhar componentes aos requisitos

- Análise de Componentes
  - Dada uma especificação, encontrar componentes que a atendam
- Alteração nos Requisitos
  - Se possível, os requisitos são adaptados aos componentes existentes



Integração dos Componentes

- Projeto do Sistema com Reuso
  - Se necessário, projeta-se novos componentes reusáveis
- Desenvolvimento e Integração
  - Desenvolvimento de novos componentes
  - Integração de todos os componentes



Vantagens

- Reduz a quantidade de software a ser desenvolvido
- Espera-se reduzir os custos e os riscos
- Espera-se uma entrega do produto mais rápida ao cliente

Desvantagens

- Pode-se desenvolver um produto que não atenda aos requisitos do cliente
- Pode ser mais difícil evoluir os sistemas
  - Componentes de terceiros
- A gerência de versões dos componentes pode ser complexa

Considerações sobre escolha de processos

- Não existe bala de prata
  - BROOKS, F.; KUGLER, H. J. **No Silver Bullet – Essence and Accident in Software Engineering**. April, 1987.

- Um bom processo é o processo que é adequado para o contexto

Processos que Lidam com Mudanças

Mudanças

- Mudanças são inevitáveis em projetos de software de grande escala
- Mudança é uma forte causa de “retrabalho”
- Abordagens para reduzir o custo de retrabalho
  - Evitar mudanças
  - Tolerância a mudanças

Lidando com Mudanças

- Prototipação
- Entrega Incremental
- Modelo Espiral

Prototipação

Prototipação

- É geralmente usada junto com outro modelo de processo
- Planeja e modela rapidamente um protótipo
  - Mais comum na definição de interfaces com os usuários (telas)
- Começa com os requisitos menos compreendidos
  - Objetivo: entender os requisitos

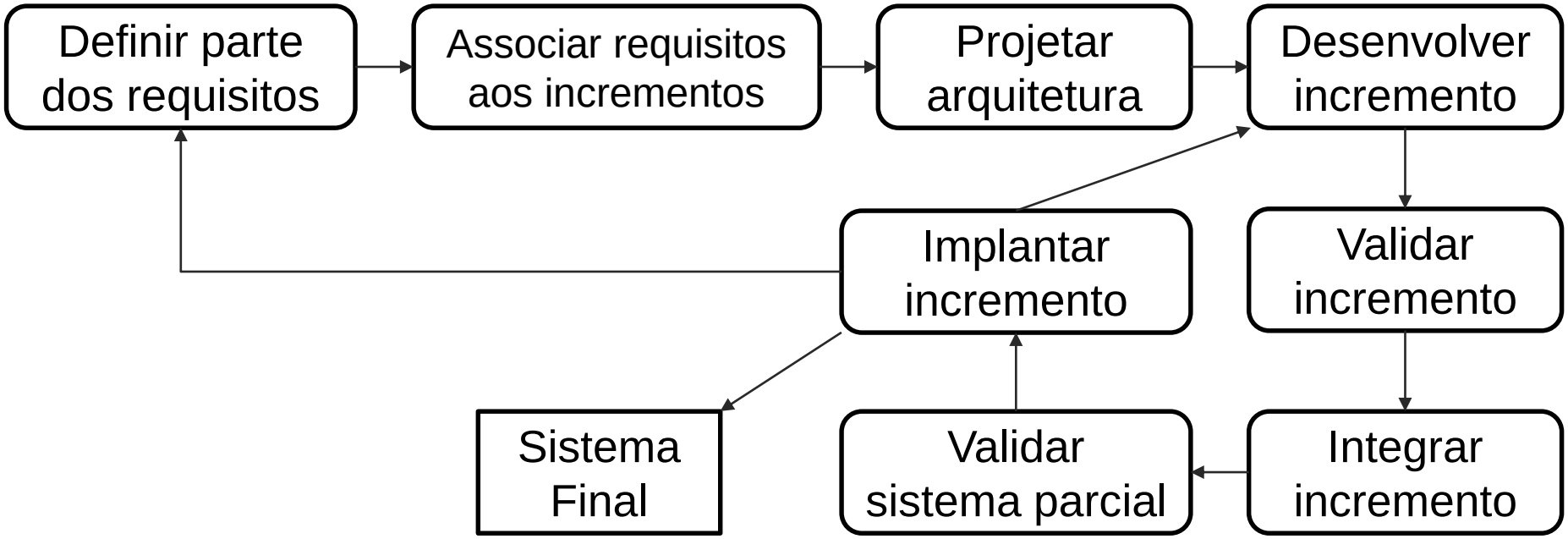
Prototipação

- O protótipo deveria ser descartado
  - E o sistema re-implementado usando outro processo (exemplo, Cascata)
- Principal vantagem
  - Auxilia o engenheiro de software e o cliente a entenderem melhor o que deve ser construído

Entrega Incremental

Entrega Incremental

- Combina elementos do modelo cascata aplicados de maneira iterativa



Vantagens

- Os clientes não precisam esperar a entrega final do sistema
  - Eles podem usar o sistema parcial
- Serviços de mais alta prioridade podem ser entregues primeiro
- O risco de falha global do projeto é menor que o Modelo Cascata

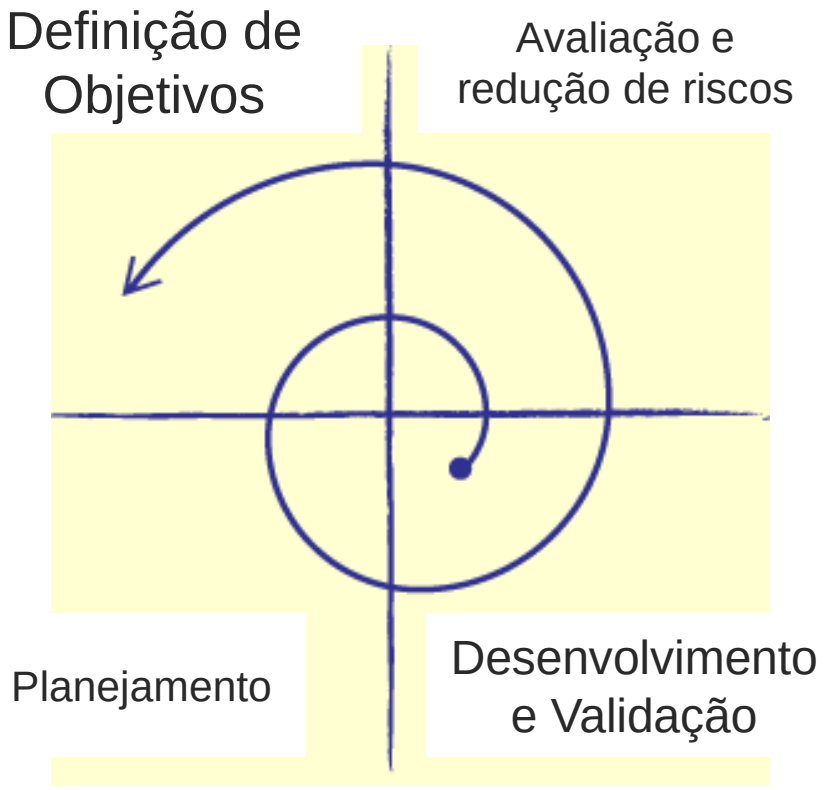
Desvantagens

- Pode ser difícil definir os recursos comuns e propriedades globais dos sistema
- Difícil adoção pelos usuários quando um novo sistema (parcial) irá substituir um sistema antigo (completo)
- Pode causar dificuldades no fechamento do contrato
  - Não há especificação completa a priori

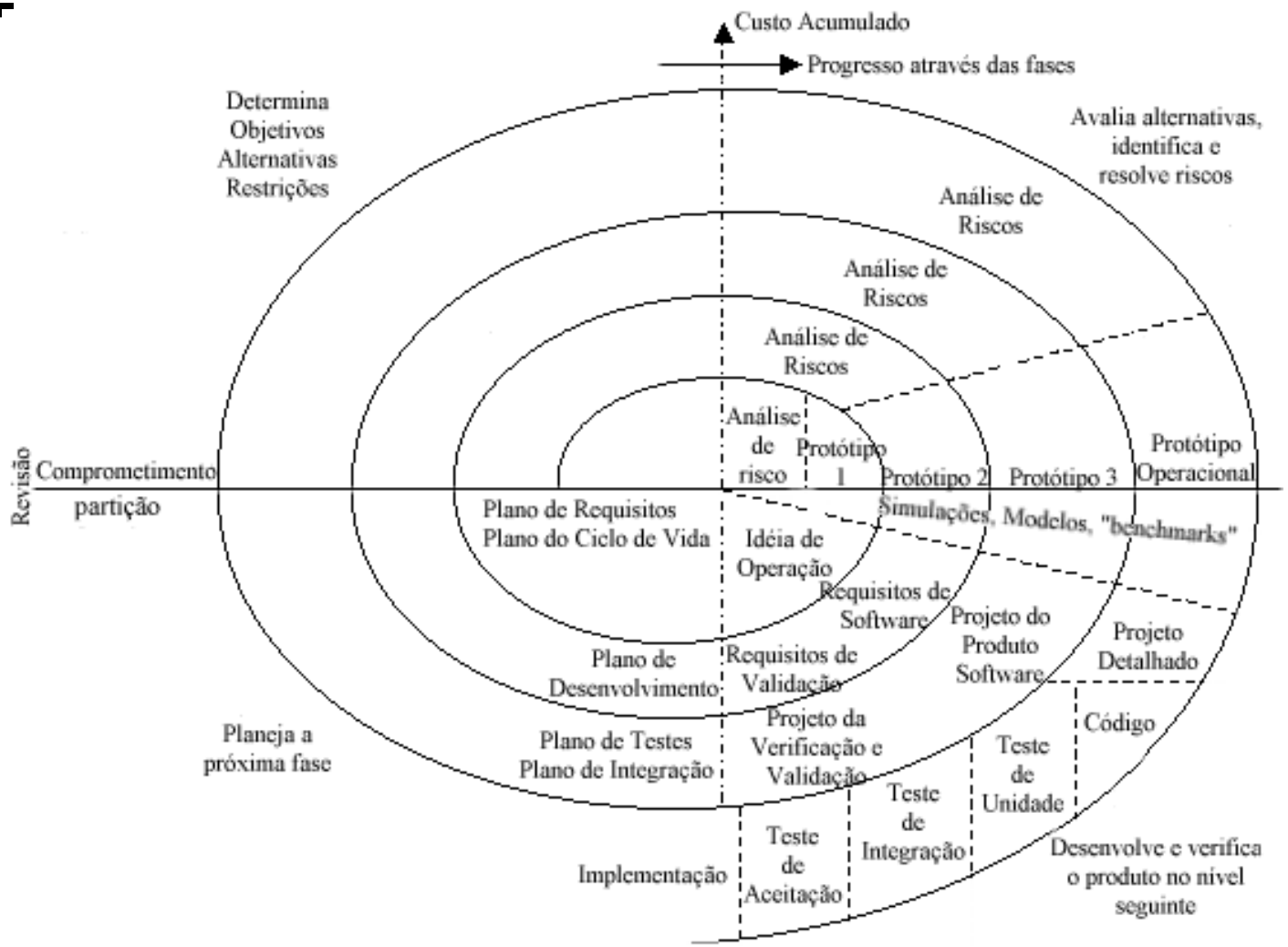
Modelo Espiral

Modelo Espiral

- Combina elementos dos modelos incrementais e de Prototipagem
  - E sequência adotada do Modelo Cascata
- Software é desenvolvido em versões
  - Prototipagem nas primeiras versões
  - Incremental nas últimas versões



Modelo Espiral



Bibliografia

- Ian Sommerville. **Engenharia de Software**, 9ª Edição. Pearson Education, 2011.
  - Capítulo 2 Processos de Software