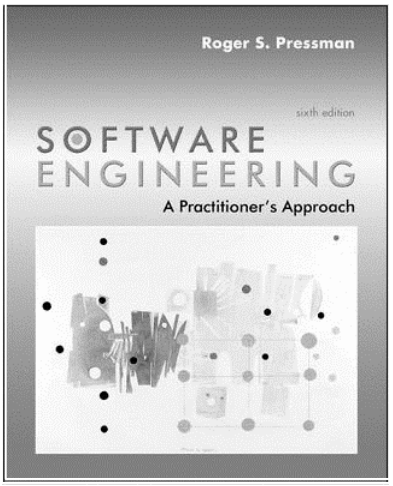
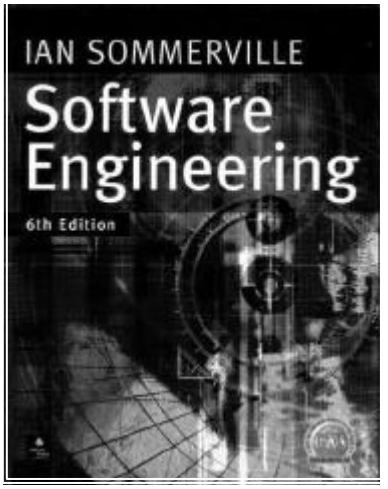
Bibliografia

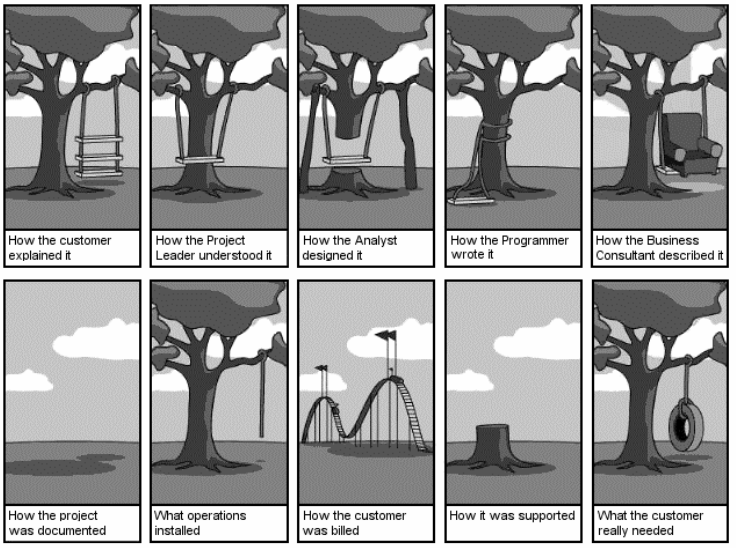




Quais são os problemas?

- A sofisticação do software ultrapassou nossa capacidade de construção.
- Nossa capacidade de construir programas não acompanha a demanda por novos programas.
- Nossa capacidade de manter programas é ameaçada por projetos ruins.
- Custo, tempo e resultado estão sempre fora das estimativas iniciais.

Visões de Projeto



Causas:

Causas Óbvias:

•Não dedicamos tempo para coletar dados sobre o desenvolvimento do software - resulta em estimativas “a olho”;

•Comunicação entre o cliente e o desenvolvedor é muito fraca;

•Falta de testes sistemáticos e completos.

Causas menos óbvias:

•O Software é desenvolvido ou projetado por engenharia, não manufaturado no sentido clássico (característica 1);

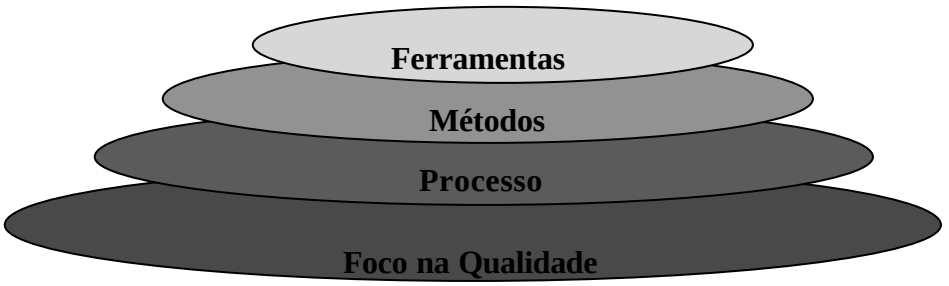
•Gerentes sem formação específica ou treinamento em desenvolvimento de SW;

•Profissionais recebem pouco treinamento formal;

•Faltam investimentos;

•Faltam métodos e automação.

Engenharia de Software em Camadas



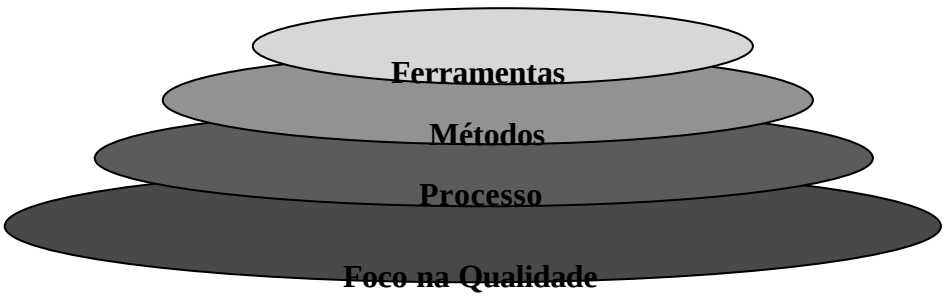
Processos de Software

- Formam a base para o controle gerencial de projetos de software e estabelecem o contexto no qual os métodos técnicos são aplicados, os produtos de trabalho (modelos, documentos, dados, relatórios, formulários etc.) são produzidos, os marcos são estabelecidos, a qualidade é assegurada e as modificações são adequadamente geridas.

Processo de Software Genérico

- Comunicação: levantamento de requisitos e atividade relacionadas.
- Planejamento: business plan (riscos prováveis, recursos disponíveis, produtos gerados, cronograma de trabalho).
- Modelagem: criação de modelo que permitem ao desenvolvedor e cliente entender melhor os requisitos do software
- Construção: geração de código (automática ou não) e testes de software
- Implantação: entrega do software e feedback do cliente

Engenharia de Software em Camadas



Métodos de ESOF

- Fornecem a técnica de “como fazer” para construir softwares. Abrangem um amplo conjunto de tarefas para cada uma das fases propostas por um processo adotado. Por exemplo, dentro da fase de Modelagem do processo, diferentes tipos de modelos podem ser adotados (UML, DFD, DHF).

Levantamento de Requisitos - Tarefas

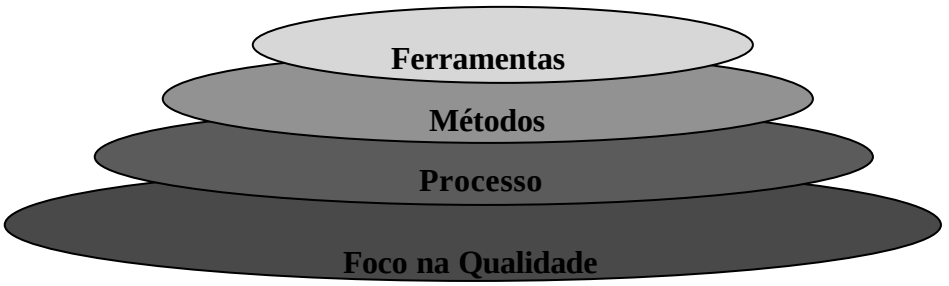
Projeto de Pequeno Porte

1. Lista de interessados (*stakeholder*).
2. Reunião informal com *stakeholders*.
3. Cada interessado fazer uma lista das características e funções desejadas.
4. Discuta requisitos e construa uma lista formal.
5. Priorize os requisitos
6. Observe áreas de incerteza.

Projeto de Grande Porte

1. Lista de interessados (*stakeholders*).
2. Entrevista de cada interessado separadamente.
3. Construa lista preliminar de funções.
4. Reuniões em grupos comuns
5. Produza cenários informais por área de interesse comum.
6. Refina os cenários de acordo com feedback.
7. Construa lista revisada de requisitos.
8. Crie pacotes seqüenciais de entrega dos requisitos.
9. Anote as restrições e limitações que serão colocadas no sistema.
10. Discuta métodos para validação do sistema.

Engenharia de Software em Camadas



Ferramentas CASE

- Fornecem um apoio automatizado ou semi-automatizado para o processo e para os métodos.
- CASE: *Computer Aided Software Engineering*.

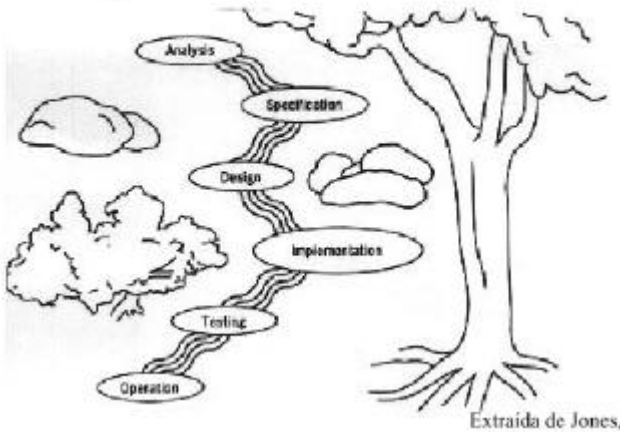
Paradigma: Conjunto de etapas que envolve métodos, ferramentas e procedimentos. Destacam-se:

- Clico de vida Clássico - modelo cascata
  - Abordagem Cascata Pura
  - Abordagem Incremental
  - Abordagem Evolucionária
- Prototipação*
- Modelo Espiral
- Técnicas de Quarta Geração

* ***Não considerado como paradigma, e sim como abordagem, por alguns autores***

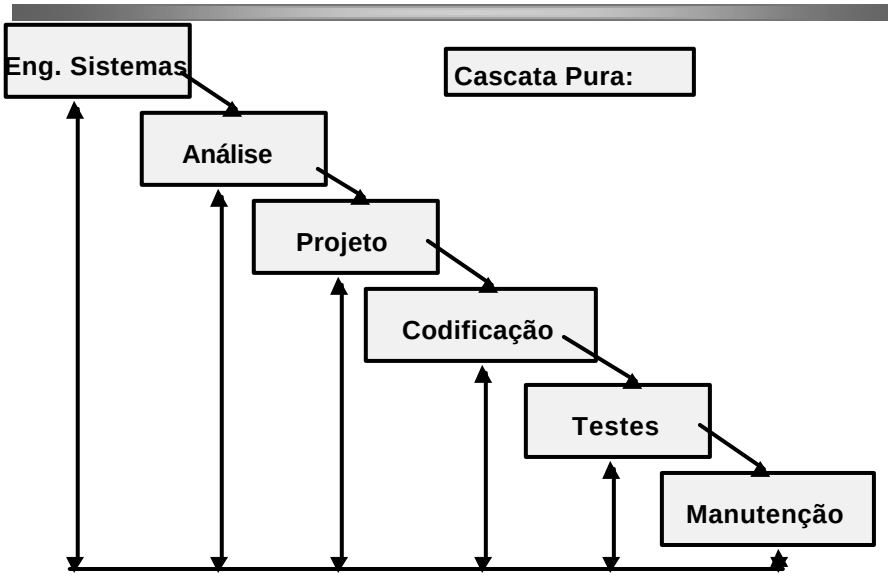
Ciclo de Vida Clássico

Ciclo de vida clássico ("Waterfall" ou Cascata)



Extraída de Jones, 1990

Ciclo de Vida Clássico



Ciclo de Vida Clássico

- Eng. Sistemas:
  - estabelecimento de requisitos;
  - envolvimento de diversos profissionais;
- Análise:
  - analista compreende o domínio da informação para o SW;
  - analista compreende a função/desempenho e interface exigidos;
  - documentação de requisitos (do usuário e do sistema);
- Projeto:
  - concentração em estrutura de dados, arquitetura do SW, detalhes procedimentais e caracterização da interface

Ciclo de Vida Clássico

- Codificação:
  - tradução do projeto em uma forma legível para o computador;
- Testes:
  - concentração em aspectos lógicos do SW;
  - garantia da execução de todas as instruções;
  - análise de aspectos funcionais externos;
  - detecção de erros.
- Manutenção:
  - mudanças para adequação/ correção de erros/ melhorias/acréscimos funcionais.

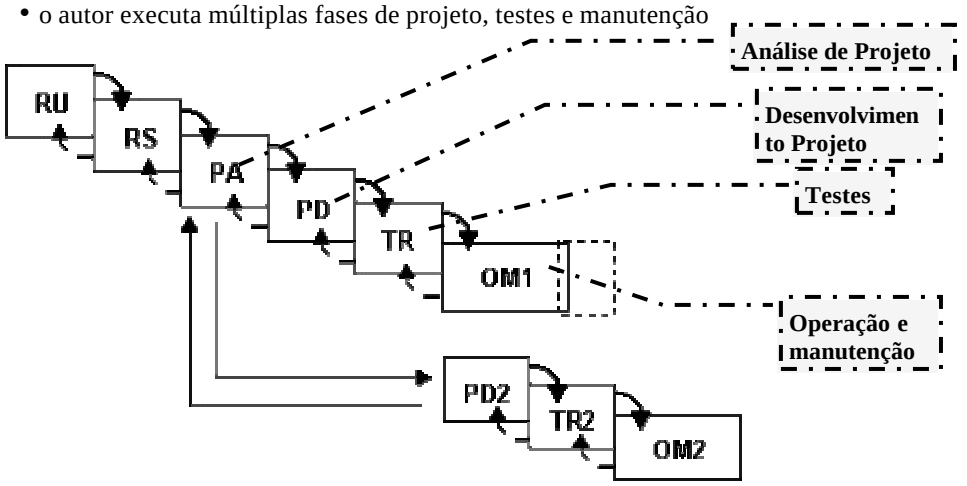
Ciclo de Vida Clássico

- Problemas do modelo Cascata:
  - Divisão (inflexível) do projeto em fases
  - Dificuldade de acomodar mudanças de requisitos dos usuários
  - Modelo apropriado à situações na qual os requisitos não sofrerão alterações e não serão modificados (??)

Ciclo de Vida Clássico

2. Abordagem Incremental:

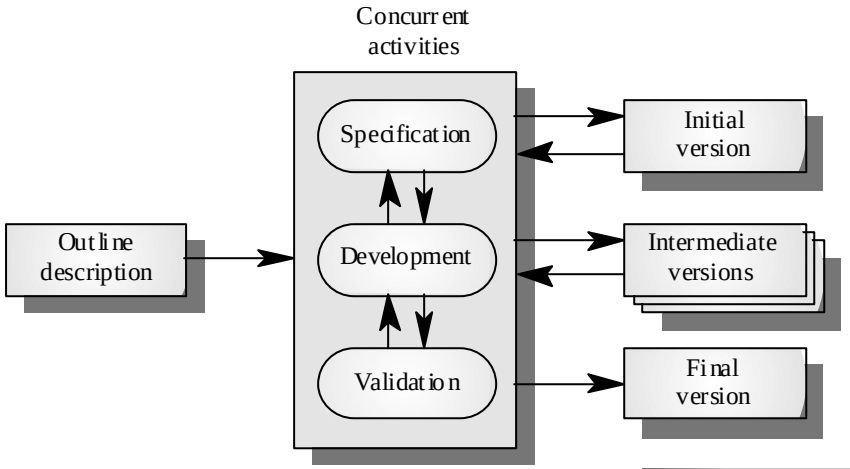
• o autor executa múltiplas fases de projeto, testes e manutenção



Ciclo de Vida Clássico

3. Abordagem Evolucionária

•o autor executa múltiplas cascatas em seqüência



Ciclo de Vida Clássico

Característica básica:

• há múltiplos ciclos de abordagem cascata pura, com sobreposição de fases de operação e manutenção;

É adequada nas situações:

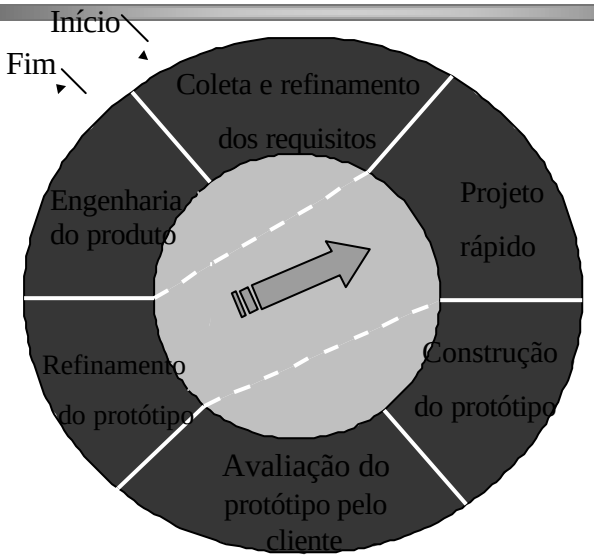
•é necessário alguma experiência do usuário para refinar e completar requisitos;

•algumas partes da implementação podem depender da existência de tecnologia ainda não disponível;

•nem todos os requisitos do usuário são conhecidos;

•alguns requisitos são mais difíceis de serem implementados que outros e não serão implementados para não atrasarem o projeto.

Prototipação



Prototipação

Características Gerais:

• desenvolvimento de protótipos de forma a refinar os requisitos do sistema;

•há refinamento de protótipos;

•o protótipo é, via de regra, reutilização de outras soluções anteriores e já existe;

•o protótipo não é versão do produto;

•é necessária uma boa definição das regras entre cliente e projetista;

•o custo pode ser maior que o esperado, uma vez que podemos dar muitas voltas no ciclo até que o cliente esteja satisfeito.

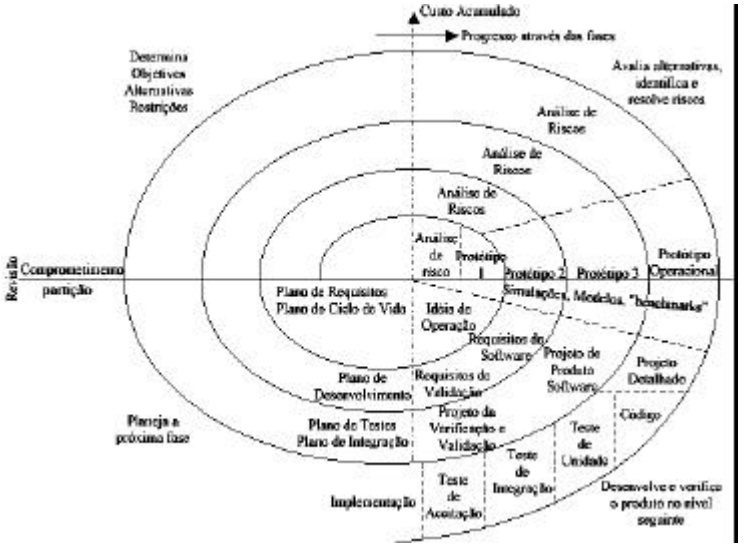
Problemas Gerais:

• Cliente vê versões do produto;

• Concessões de implementação;

• Necessidade de rígida definição de regras.

Modelo Espiral



Modelo Espiral

FASES do Modelo Espiral:

I. Definição dos objetivos

Especificação dos objetivo/riscos/alternativas e restrições.

II. Análise dos riscos

Identificação e solução dos principais riscos

III. Desenvolvimento e validação

IV. Avaliação pelo Cliente

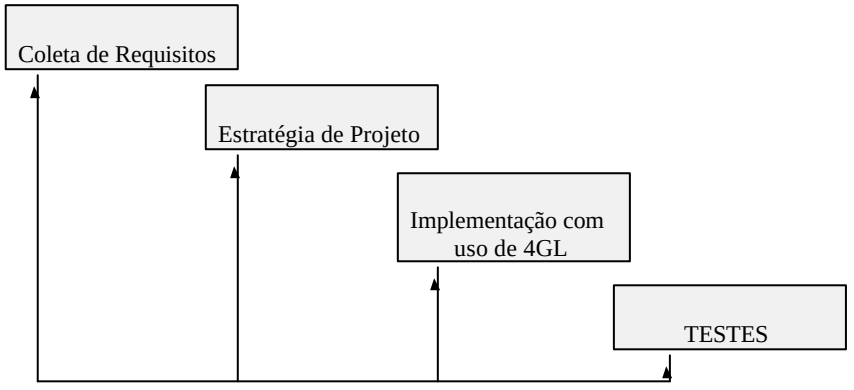
O projeto é revisto e se define planos para a próxima "volta da espiral"

A cada interação, ao redor da espiral, versões mais completas do Software são construídas. Ao final de cada arco da espiral, a conclusão da análise de riscos resulta em uma decisão de prosseguir ou não - Forte Análise de Riscos.

Técnicas de Quarta Geração - 4GT

Trata-se de um amplo conjunto de Ferramentas, visando que o desenvolvedor especifique características do SW em alto nível.

Visa geração automática de SW.



Técnicas de Quarta Geração - 4GT

Observações e Problemas Gerais:

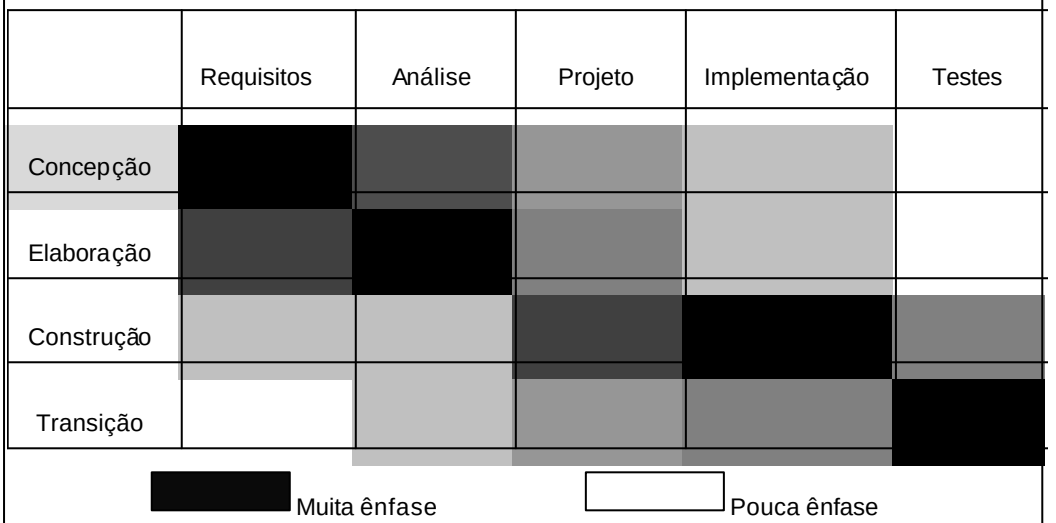
• desenvolvedor deve realizar testes cuidadosos - desenvolver uma documentação significativa e executar as demais atividades de transição;

• há divergências entre os diversos autores sobre resultados;

• muito aplicadas em sistemas de informação comerciais;

• a aplicação de técnicas de 4GT para grandes aplicações exigem tanto ou mais análise, planejamento e testes para conseguir substancia is reduções de tempo.

O Processo Unificado



- **Executar o processo de Engenharia de Requisitos (levantamento e análise).**
- **Determinar o que deve ou não estar na nova versão do produto.**

CONCEPÇÃO - Artefatos

- **Plano do Projeto (global e detalhado para a fase de elaboração).**
- **Especificação de Requisitos.**
- **Diagrama de Pacotes e Diagrama de Domínio.**

O Processo Unificado

	Requisitos	Análise	Projeto	Implementação	Testes
Concepção					
Elaboração					
Construção					
Transição					

Muita ênfase Pouca ênfase

O Processo Unificado

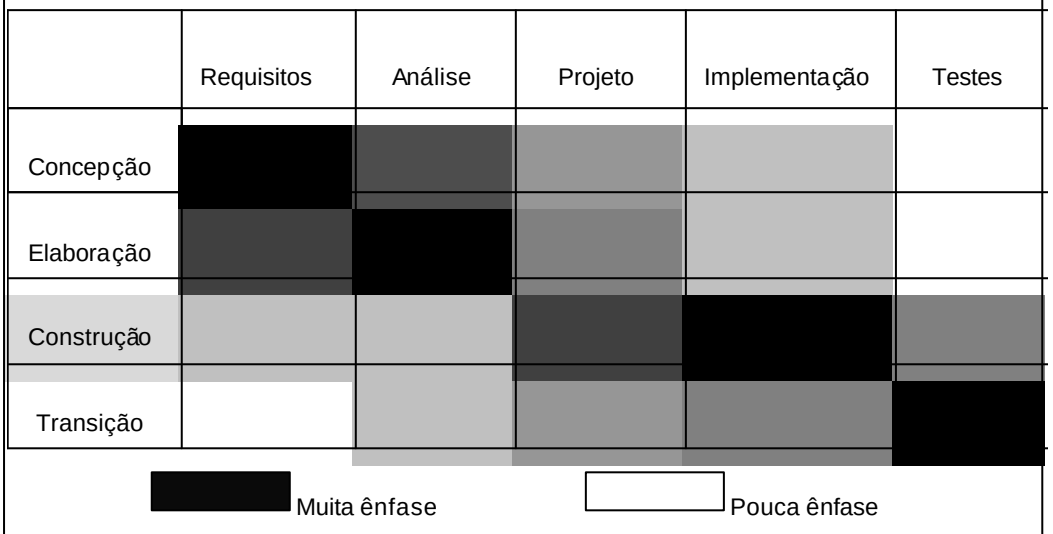
- **Analisar mais profundamente o domínio do problema, identificando os componentes de cada subsistema do software.**
- **Assegurar que os requisitos, a nova arquitetura do sistema e os planos estão suficientemente estáveis e os riscos suficientemente diminuídos.**
- **Prever com precisão e segurança a conclusão do desenvolvimento.**

O Processo Unificado

ELABORAÇÃO - Artefatos

- **Plano do Projeto revisado.**
- **Especificação de Requisitos e Declaração de Escopo.**
- **Documento de Arquitetura do Sistema**
  - **Diagramas de Pacotes.**
  - **Diagramas de Domínio.**
  - **Diagramas de Classes.**
  - **Diagramas de Interação.**
  - **Diagramas de Atividades.**

O Processo Unificado



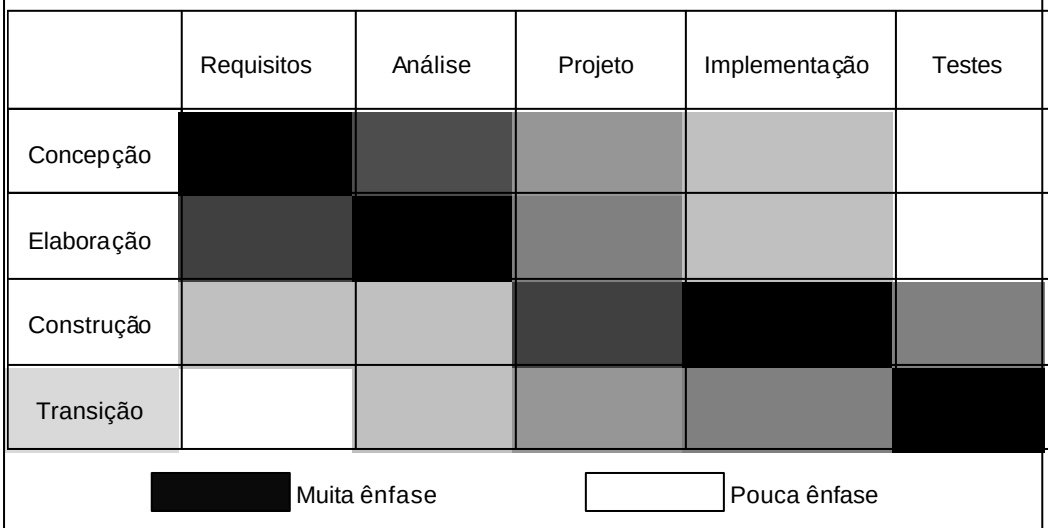
O Processo Unificado

- **Escrever o código-fonte de cada parte do sistema (subsistema), considerando os requisitos validados e a arquitetura de software definida.**
- **Produzir a documentação, de forma iterativa e incremental, dos módulos implementados de cada subsistema.**
- **Desenvolver atividades e programas para implantação, empacotamento e distribuição do software (produto final).**

O Processo Unificado

- **Plano do Projeto revisado.**
- **Especificação de Requisitos e Declaração de Escopo revisados.**
- **Documento de Arquitetura do Sistema atualizado.**
- **Documentos gerados pela implementação.**
- **Testes e integrações.**

O Processo Unificado



O Processo Unificado

- **Disponibilizar o sistema para uso.**
- **Treinar usuários e equipe de manutenção do sistema através de testes beta.**
- **Coletar observações e novos requisitos de usuários.**
- **Corrigir erros para melhorar o desempenho e a usabilidade do sistema.**
- **Executar as atividades e programas para implantação, empacotamento e distribuição do sistema.**

O Processo Unificado

TRANSIÇÃO - Artefatos

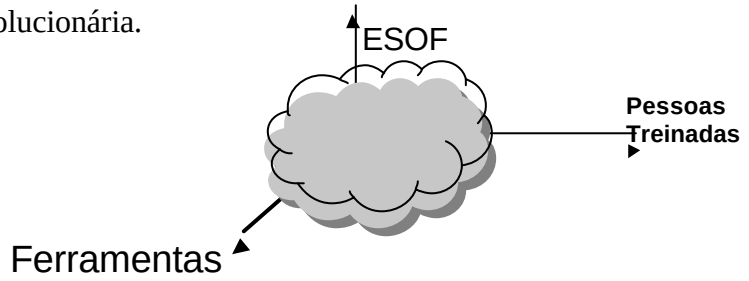
- **Plano de Projeto realizado.**
- **Especificação de Requisitos e Declaração de Escopo implementados.**
- **Documento de Arquitetura atualizado.**
- **Documentos gerados pela implementação revisados.**
- **Código-objeto para distribuição.**
- **Sistema testado e implantado.**

Combinando Paradigmas

✓Há viabilidade na combinação de paradigmas - PRESSMAN (2005).

✓Não há necessidade de ser dogmático em relação à escolha de paradigmas para a Engenharia de Software;

✓O paradigma do modelo espiral realiza isso diretamente, combinando a prototipação e elementos do ciclo de vida clássico em uma abordagem evolucionária.



Exercícios

1. Descreva 2 casos de problemas causados por erros em software que você conhece (OBS: não adianta falar do Windows)
2. Descreva 2 produtos (que não seja um computador) onde o software faz a diferença.