Software e Engenharia de Software

Silvia Regina Vergilio - UFPR

1) O que é software?

- Programas de computador
- Entidade abstrata.
- Ferramentas (mecanismos) pelas quais:
  - exploramos os recursos do hardware.
  - executamos determinadas tarefas
  - resolvemos problemas.
  - interagimos com a máquina.
  - tornamos o computador operacional.

1) O que é software?

Conceito mais amplo que inclui também:

- Instruções que executam uma função desejada.
- Estrutura de dados para manipular informação.
- Documentos para desenvolver, operar e manter os programas.

2) Dificuldades para desenvolver software

- Saber o que o software deve fazer : quais os requisitos (abstração);
- Ferramentas; linguagem; so
- Tempo e custos elevados de desenvolvimento.
- Prever falhas (antes de entregar).
- Tratar manutenção e versões.
- Produtividade não cresce com a demanda de serviços.

3) Características do Software

- software não é um elemento físico; é um elemento lógico (não tem propriedades físicas, como visualizar, medir ...)
- abstração maior; o produto final é diferente
- o software não pode ser manufaturado; custos estão concentrados no desenvolvimento e não na manufatura.
- o processo de gerenciamento é diferente; o relacionamento entre as pessoas é diferente;

3) Características do Software

- existem diferentes abordagens para se chegar no produto final
- o software não se desgasta com o uso; mas deteriora-se
- não há peças de reserva. => manutenção, correção, aperfeiçoamento.
- não é construído aproveitando-se componentes prontos.
- um erro durante um teste => erro de projeto; mais difícil de testar.

4) Crise de Software

Alguns autores associam a palavra “crise” aos problemas para desenvolver software

4. Crise do Software – Eras da Computação

50 –65 – Primeira Era

- Software Customizado
- Personalizados sem nenhuma documentação
- Batch

4. Crise do Software – Eras da Computação

65-75 – Segunda Era

- Multiusuário
- Tempo Real
- BD, produto de software
- software houses ➔ crise de software

4. Crise do Software – Eras da Computação

75-85 – Terceira Era

- Sistemas Distribuídos
- IA
- Hardware de baixo custo

4. Crise do Software – Eras da Computação

85- …. Quarta Era

- Sistemas Especialistas
- Redes Neurais
- Computação Parela

4. Crise do Software – Eras da Computação

- Primeira Geração - assembley
- Segunda Geração – Fortran, Cobol, Algol
- Terceira Geração – Pascal, Eifel, …
- Quarta Geração – DBASE, lg de consultas SQL, geradores de programa, lg de especificação formal, ferramentas …

4) Crise de Software

Problemas:

- Software inadequado.
- Cronogramas e custos imprecisos - dificuldades em prever o progresso durante o desenvolvimento.
- Inexistência de dados históricos sobre o processo de desenvolvimento.
- Comunicação deficiente - insatisfação de usuários.
- Carência de conceitos quantitativos sobre confiabilidade, qualidade, reusabilidade.
- Software existente é de difícil manutenção.

4) Crise de Software

Solução:

- Combinar métodos para as fases de desenvolvimento.
- Ferramentas para automatizar esses métodos.
- Técnicas para assegurar qualidade.

=> Disciplina: Engenharia de Software.

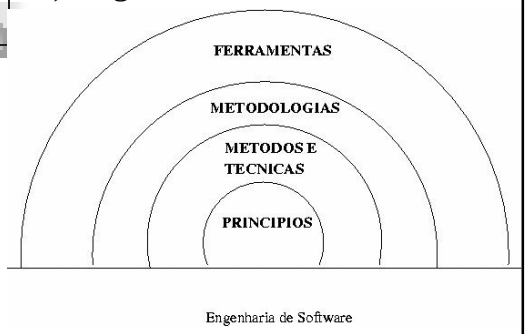
5) Engenharia de Software

- Abordagem sistemática para o desenvolvimento, operação e descarte de software.
- Aplicação prática de conhecimento científico ao projeto e construção de software.
- Disciplina que utiliza princípios de engenharia para produzir e manter softwares dentro de prazos e custos estimados.

5) Engenharia de Software

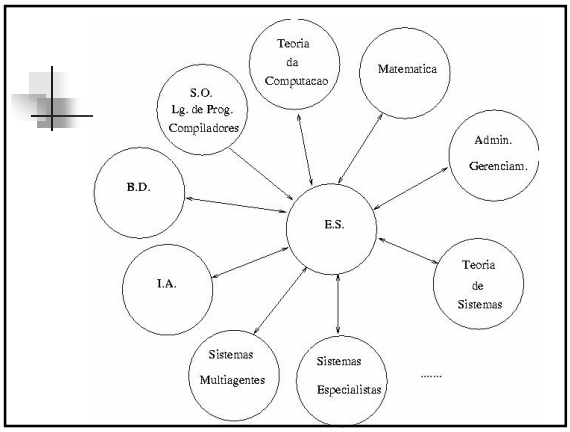
- Objetivos: Melhorar a qualidade do software e aumentar a produtividade e satisfação profissional de engenheiros de software.
- Definição: Disciplina que utiliza um conjunto de métodos, técnicas e ferramentas para analisar, projetar e gerenciar desenvolvimento e manutenção de software.

5) Engenharia de Software



5) Engenharia de Software

- Métodos e Técnicas: como fazer
- Metodologias: como aplicar
- Ferramentas: Automatizam os métodos, dão apoio à utilização dos mesmos. **CASE** => (Computer-Aided Software Engineering): Ferramentas integradas para desenvolver software.



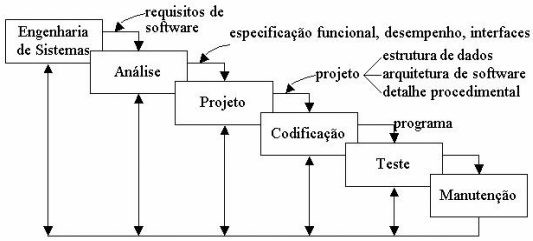
6) Princípios da Engenharia de Software

- Formalidade: reduz inconsistências
- Abstração: aspectos importantes, ignorar detalhes
- Decomposição: lidar com complexidade
- Generalização: reutilização, custo
- Flexibilização: mudanças, processo incremental

7) Paradigmas da Engenharia de Software

À E. S. está associado um conjunto de passos
(que englobam métodos, ferramentas, etc)
denominado **paradigma**

Ciclo de Vida Clássico

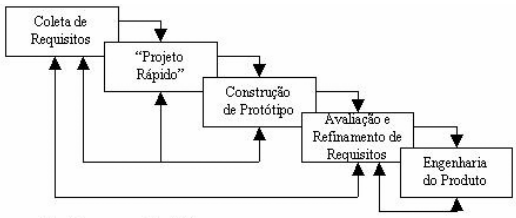


Ciclo de Vida Clássico

Problemas para aplicação:

- Na prática, projetos não seguem o fluxo seqüencial.
- Acomodações de incertezas no início do projeto é difícil.
- Versão funcional dos programas disponível após os últimos estágios do projeto

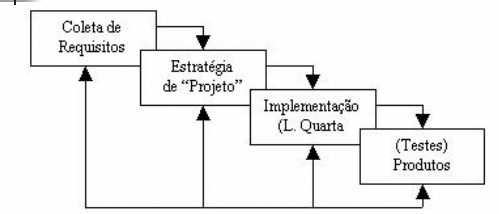
Prototipação



Prototipação

- Localiza "aspectos visíveis" para o usuário (E/S).
- A iteração pode adequar o protótipo às necessidades do usuário.
- O protótipo pode ser descartado ou fazer parte do produto final.
- Problemas: Cliente insiste que o protótipo seja com ligeiras modificações, a versão final do produto. Decisões e soluções improvisados tornam-se parte do produto final.

Linguagens de Quarta Geração



Linguagens de Quarta Geração

- Ferramentas para especificação de alto nível (L4G):

  Consulta a base de dados.

  Geração de relatórios.

  Manipulação de dados.

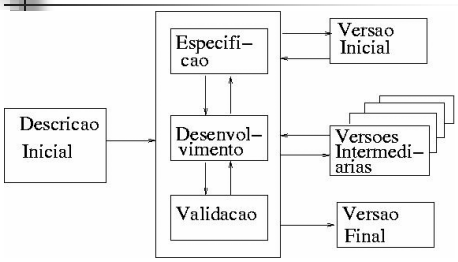
  Definição e interação com Telas.

  Geração de código.

Linguagens de Quarta Geração

- Domínio predominante : Sistemas comerciais de informação.
- Boa produtividade para sistemas pequenos e médios e aplicação específicas.
- Problemas: Para sistemas grandes, demanda muito tempo; e ainda permanece a necessidade de projeto

Evolucionário



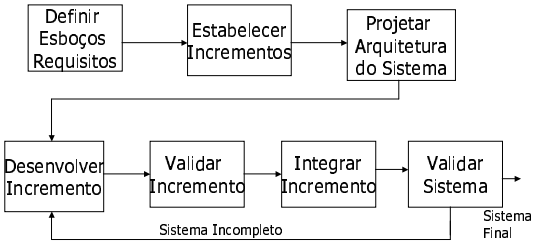
Evolucionário

- Tudo merece uma nova chance
- Incorporação de diferente partes e criação de diferentes versões
- Inclui prototipação
- Permite o desenvolvimento exploratório

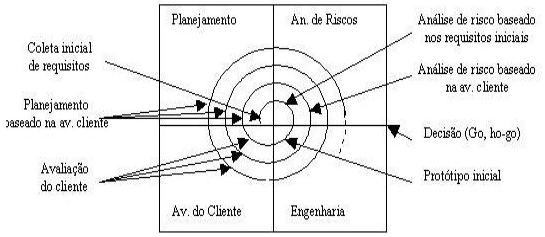
Incremental

- Abordagem intermediária
- Combina vantagens dos paradigmas ciclo de vida clássico e evolucionário
- Identificação das funções do sistema, estabelecimento de incrementos e prioridades
- Cada incremento pode utilizar um paradigma de desenvolvimento diferente
- Dificuldade para dividir e gerenciar versões

Incremental



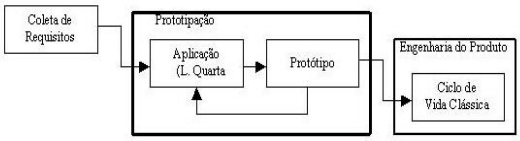
Espiral



Espiral

- Paradigma mais realístico - sistemas grandes
- É um metamodel o
- Incorpora análise de riscos.
- Permite prototipação em mais de um estágio
- Problemas: O modelo é relativamente novo. Requer esperteza. Pode nunca terminar.

Combinação de Paradigmas



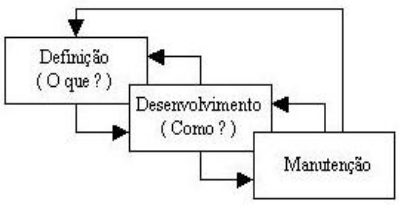
7) Paradigmas da Engenharia de Software

- Que paradigma usar ?

  Depende da natureza da aplicação.

  Métodos e ferramentas disponíveis, etc.

8) Uma Visão Genérica da E.S.



8) Uma Visão Genérica da E.S.

1) Definição

Função, desempenho, interface, restrições de projeto, critérios de validação.

Análise de sistemas

Planejamento de projeto de software.

Análise de requisitos.

8) Uma Visão Genérica da E.S.

2) Desenvolvimento :

Estrutura de dados, Arquitetura de software, detalhes procedimentais, programas, testes.

Projeto de software.

Codificação.

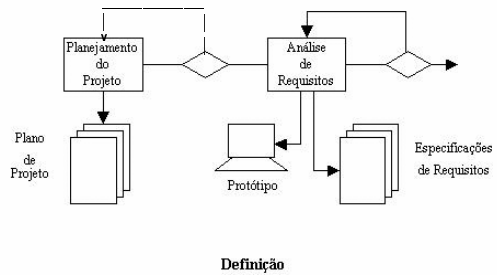
Testes

8) Uma Visão Genérica da E.S.

3) Manutenção

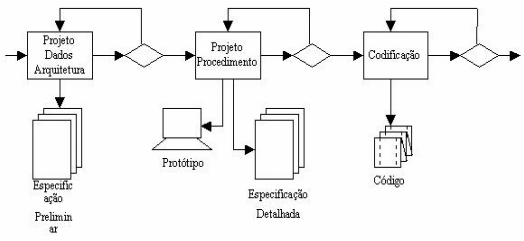
Corretiva: para corrigir defeitos;
Adaptativa: para acomodar mudanças no ambiente externo do software (S. O., periféricos, etc)
Perfectiva: para inclusão de novas funcionalidades

8) Uma Visão Genérica da E.S.



Definição

8) Uma Visão Genérica da E.S.



Desenvolvimento

8) Uma Visão Genérica da E.S.