Software e Engenharia de Software

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.

O que é software?

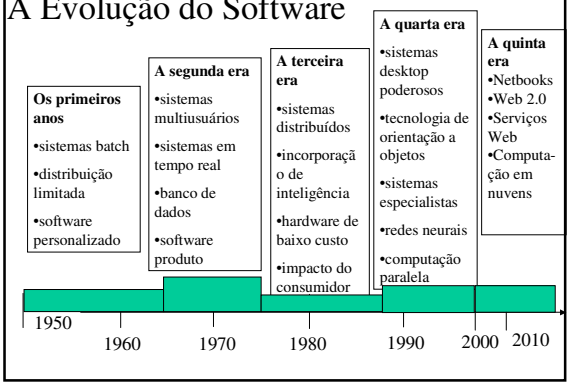
Conceito mais amplo que inclui também:

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

Tipos de Sistemas de Software

- Software básico
- Software para sistema em tempo real
- Software comercial
- Software para engenharia e aplicações científicas
- Software embarcado (ex. microwave)
- Software para computadores pessoais (shrink-wrap)
- Software baseado em inteligência artificial
- Software de entretenimento

A Evolução do Software



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.

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;

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.

Crise de Software

Alguns autores associam a palavra "crise" aos problemas para desenvolver software

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.
- Mitos da ES
- Comunicação deficiente - insatisfação de usuários.
- Carência de conceitos quantitativos sobre confiabilidade, qualidade, reusabilidade.
- Software existente é de difícil manutenção.

Crise de Software

Solução:

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

=> Disciplina: Engenharia de Software.

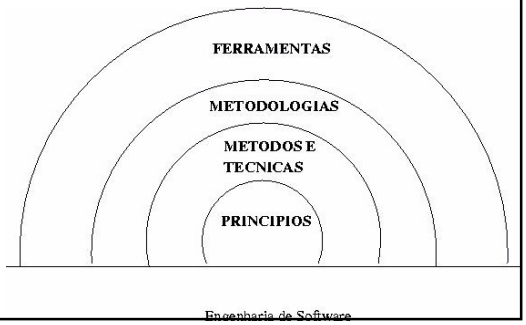
Engenharia de Software

- Abordagem sistemática para o desenvolvimento, operação, manutenção 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.
- ".. construção por muitas pessoas de um software com múltiplas visões". (Parnas 1987)

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.

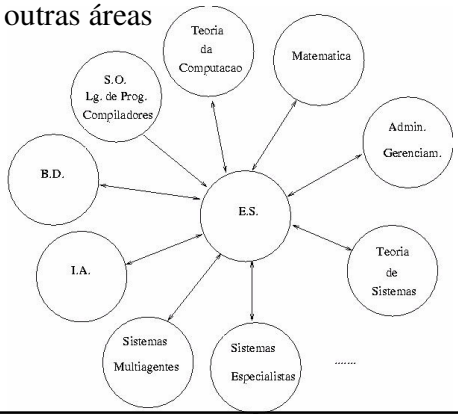
Engenharia de Software



Engenharia de Software

- Métodos e Técnicas: detalhes de como fazer
- Metodologias: como aplicar
- Ferramentas: automatizam os métodos, dão apoio a utilização.
  CASE => (Computer-Aided Engineering): Ferramentas integradas para desenvolver software.

ES e outras áreas



Princípios da Engenharia de Software

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

Processo de Software

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

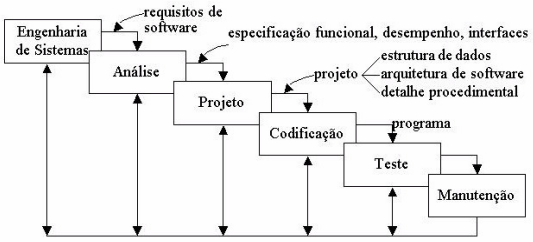
Processo de Software

- Um conjunto estruturado de atividades requeridas para desenvolver um sistema de software
- Um modelo de processo de software é uma representação abstrata de um processo. Apresenta uma descrição de um processo de alguma perspectiva particular

Ciclo de Vida Clássico (cascata)

- modelo mais antigo e o mais amplamenteconhecido da engenharia de software
- modelado em função do ciclo da engenharia convencional
- requer uma abordagem sistemática, seqüencial ao desenvolvimento de software
- O resultado de uma fase é entrada para outra fase.

Ciclo de Vida Clássico (cascata)



Engenharia de sistemas

- envolve a coleta de requisitos em nível dosistema, com uma pequena quantidade deprojeto e análise de alto nível
- esta visão é essencial quando o software devefazer interface com outros elementos(hardware, pessoas e banco de dados)

Análise de Requistos de Software

- o processo de coleta dos requisitos é intensificado e concentrado especificamente no software
- deve-se compreender o domínio da informação, a função, desempenho e interfaces exigidos

Projeto

- tradução dos requisitos do software para umconjunto de representações que podem seravaliadas quanto à qualidade, antes que acodificação se inicie

Codificação

- tradução das representações do projeto parauma linguagem “artificial” resultando eminstruções executáveis pelo computador

Teste

Concentra-se:

- nos aspectos lógicos internos do software,garantindo que todas as instruções tenham sido testadas
- nos aspectos funcionais externos, paradescobrir erros e garantir que a entrada definida produza resultados que concordem com os esperados.

Manutenção

- provavelmente o software deverá sofrer mudanças depois que for entregue ao cliente
- causas das mudanças: e*rros*, *adaptação do software para acomodar mudanças em seu ambiente externo* e *exigência do cliente para acréscimos funcionais e de desempenho*

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

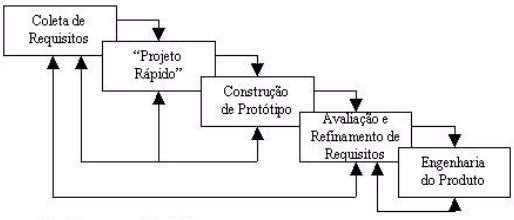
Ciclo de Vida Clássico

- O modelo Cascata trouxe contribuições importantes para o processo de desenvolvimento de software:
  - imposição de disciplina, planejamento e gerenciamento;
  - a implementação do produto deve ser postergada até que os objetivos tenham sido completamente entendidos

Prototipação

- Possibilita que o desenvolvedor crie um modelo (protótipo) do software a ser construído para auxiliar a entender os requisitos do usuário
- Apropriado para quando os requisitos não estão claramente definidos
- Derivado da técnica de extração de requisitos

Prototipação



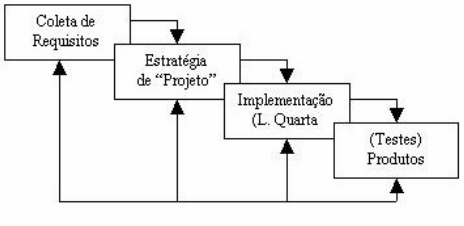
Abordagem Prototipação

- Validar a precisão dos requisitos ou aceitabilidade das decisões
  - O usuário irá colaborar?
- Validar a viabilidade de uma estratégia proposta.
  - O software é particionável?
- Observações:
  - protótipos só são válidos se construídos rapidamente
  - protótipos devem ser desprezados.

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.

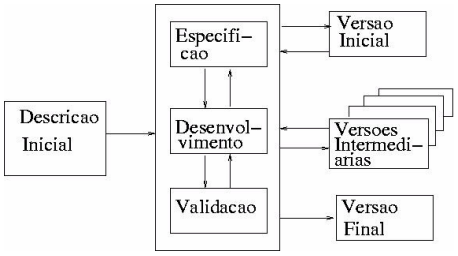
Linguagens de Quarta Geração

- 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

Evolutivo

- Tudo merece uma nova chance – situações para acomodar um processo que evolui com o tempo
- Incorporação de diferente partes e criação de diferentes versões - são iterativos
- Inclui prototipação
- Permite o desenvolvimento exploratório

Evolutivo



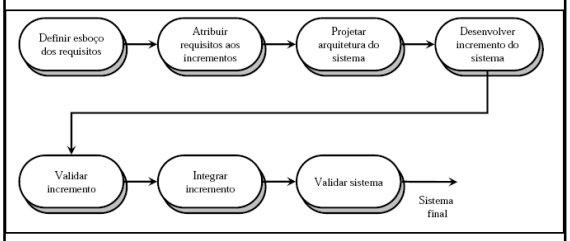
Incremental

- Abordagem intermediária
- Combina vantagens dos paradigmas ciclo de vida clássico e evolutivo
- 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

- O valor agregado ao Cliente está na entrega em cada incremento de modo que a funcionalidade do sistema estará disponível mais cedo
- Incrementos iniciais funcionam como protótipos para ajudar a evocar requisitos para incrementos posteriores
- Menores riscos de falha no projeto em geral
- Os serviços do sistema de alta prioridade tendem a receber a maioria dos testes

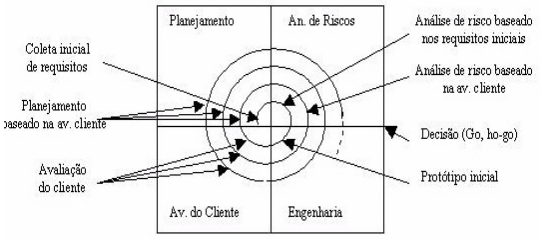
Incremental



Espiral

- Modelo que acopla a natureza iterativa da prototipação co os aspectos controlados e sitemáticos do modelo cascata
- Dividio em regiões ou atividades de trabalho.
- Cada volta do espiral representa um desenvolvimento completo.
- O loop mais interno se concetra mais na definição dos requisitos
- A gerência pode adaptar o modelo e suas fases

Espiral



Espiral

- Paradigma mais realístico - sistemas grandes
- É um metamodelo
- Incorpora análise de riscos.
- Permite prototipação em mais de um estágio

- Problemas: O modelo é relativamente novo. Requer esperteza. Pode nunca terminar, precisa ser evolutiva e controlável.

Considerações sobre processos

- Que processo usar ?

  Depende da natureza da aplicação.

  Métodos e ferramentas disponíveis, etc.

Outros processos

- RAD (Rapid Application Development)
- Processo formal
- Modelo de componentes
- Open
- Processos ágeis
- Scrum
- XP
- OpenUP

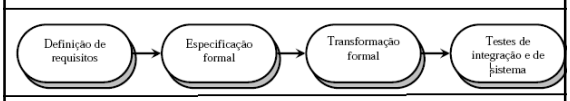
Programação Extrema – XP (eXtreme Programming)

- Nova abordagem para o desenvolvimento de software baseado no desenvolvimento e entrega de incrementos de funcionalidade bem pequenos
- Conta com melhoramento constante do código, envolvimento do usuário no time de desenvolvimento e programação em pares

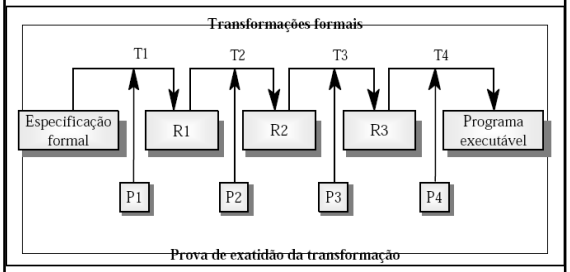
Desenvolvimento de sistemas formais

- Baseado na transformação de uma especificação matemática através de diferentes representações para um programa executável
- Transformações são 'preservadoras de exatidão', portanto, são diretas para mostrar que o programa está de acordo com sua especificação
- Contido na abordagem 'Cleanroom' para desenvolvimento de software

Desenvolvimento de sistemas formais



Transformações Formais



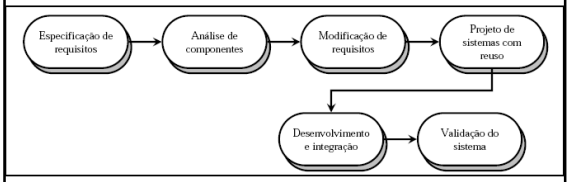
Desenvolvimento de sistemas formais

- Problemas
  - Necessidade de habilidades especializadas e treinamento para aplicar a técnica
  - Difícil de especificar formalmente alguns aspectos do sistema como a interface de usuário
- Aplicabilidade
  - Sistemas críticos, especialmente aqueles no qual um case de segurança deve ser feito antes do sistema ser posto em operação

Desenvolvimento Baseado em Componentes

- Baseado no reuso sistemático, onde os sistemas são integrados de componentes existentes ou sistemas padronizados
- Estágios do Processo
  - Análise do componente
  - Modificação dos requisitos
  - Projeto do sistema com reuso
  - Desenvolvimento e integração
- Esta abordagem está se tornando mais importante, mas a experiência ainda é limitada com ela

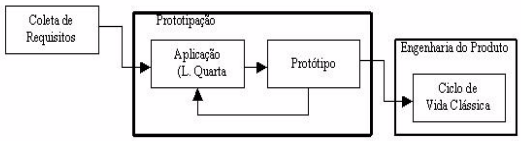
Desenvolvimento orientado a componentes



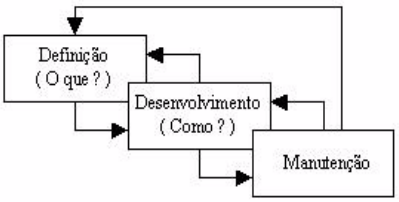
Que processo usar

- Sempre deve existir um processo de software definido - padrões de qualidade.
- Criar um processo baseado em fases específico para cada projeto.
- O profissional deve estar apto a avaliar a aplicação a ser desenvolvida e a situação do ambiente de desenvolvimento para decidir qual o melhor processo de software a ser definido.

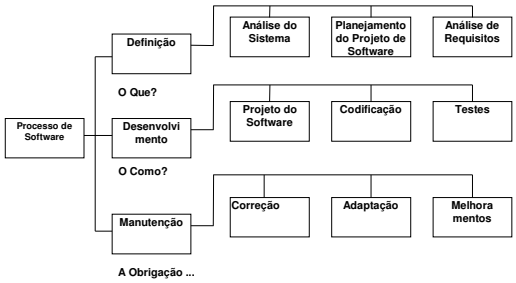
Combinação



Uma Visão Genérica da E.S.



Uma Visão Genérica da E.S.



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.

Uma Visão Genérica da E.S.

2) Desenvolvimento:

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

Projeto de software.

Codificação.

Testes

Uma Visão Genérica da E.S.

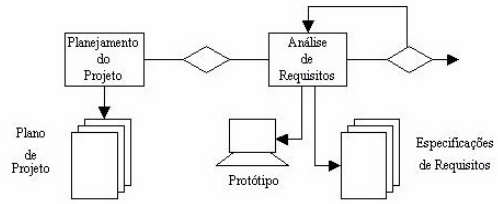
3) Manutenção

Corretiva: para corrigir defeitos;

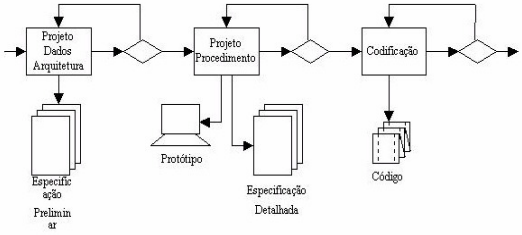
Adaptativa: para acomodar mudanças no ambiente externo do software (S.O., periféricos, etc)

Perfectiva: para inclusão de novas funcionalidades

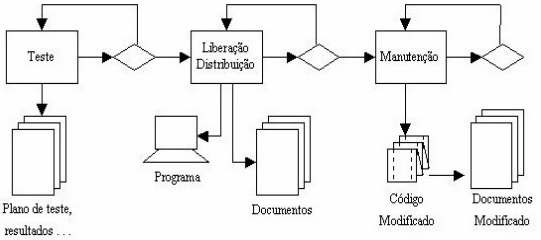
Uma Visão Genérica da E.S.



Uma Visão Genérica da E.S.



Uma Visão Genérica da E.S.



O Ciclo de Vida Canônico

- Estudo de Viabilidade
- Iniciação do projeto
- Especificação de requisitos
- Projeto da arquitetura
- Projeto detalhado
- Codificação
- Teste de unidade
- Teste de aceitação
- Teste operacional
- Encerramento do projeto
- Operação
- Desativação do produto

O modelo canônico deve ser tratado como uma referência que deve ser adaptada para cada situação.

Referências

• Pressman, R. Software Engineering: A Practitioner's Approach. McGraw-Hill, 6th edition, 2006.

• Sommerville, I. Software Engineering:) International Computer Science Series) 8th edition, 2006.