

# Análise e Projeto de Sistemas

Introdução à Engenharia de Software

# Objetivo

- Definir a engenharia de software e explicar sua importância
- Discutir o conceito de produto de software e processo de software
- Explicar a importância da visibilidade do processo
- Introduzir a noção de responsabilidade profissional

# Contexto

- A economia de todos os países do mundo depende cada vez mais de software (ex. indústria de turismo).
- Existem cada vez mais sistemas controlados por software
- Os gastos com desenvolvimento de software representam uma fração significativa do PIB de muitos países.

# Contexto - primórdios

- O desenvolvimento de software era puramente artesanal;
- As pessoas desenvolvendo sistemas erravam constantemente nas suas estimativas de custo e tempo;
- Os sistemas continham muitos erros;
- Consertar erros geralmente produzia mais erros;
- Tamanho dos sistemas crescendo consistentemente.

# Contexto - exemplos interessantes

- Estudo feito em 1979 pelo governo dos Estados Unidos em relação ao software produzido:
  - 2% Funcionava;
  - 3% Funcionaria com poucas correções;
  - 45% Entregues mas nunca foram usados com sucesso;
  - 20% Usados, mas tremendamente modificados ou abandonados;
  - 30% Pagos, mas nunca foram terminados e/ou entregues.

# Contexto - exemplos interessantes

- A proporção dos custos com software e hardware mudou bastante ao longo do tempo:
  - Década de 60: 20% - 80%
  - Década de 70: 50% - 50%
  - Década de 80: 80% - 20%

# Custo do Software

- Custo do software geralmente domina o custo do desenvolvimento. Os custos do software de um PC são geralmente maiores do que o hardware.
- Software custa mais para manter do que para desenvolver! Para sistemas com uma longa vida, os custos de manutenção são várias vezes maior do que o de desenvolvimento.
- Engenharia de Software trata do desenvolvimento de software de **forma eficaz**.

# Definições de Engenharia de Software

- É uma disciplina que se preocupa com os problemas práticos inerentes ao desenvolvimento de sistemas de grande porte.
  - Não é simplesmente programação;
  - Também não é só ciência da computação;
  - Uso de teorias (resultados), métodos, e ferramentas na resolução de problemas

# Definições de Engenharia de Software

- É o estabelecimento e uso de sólidos princípios de engenharia visando obter economicamente um software que seja confiável e que funcione eficientemente em máquinas
- Engenharia de Software trata do desenvolvimento de software de forma eficaz
- É a abordagem sistemática ao desenvolvimento, operação, manutenção e aposentadoria de Software

# Características da Engenharia de Software

- Engenharia de Software se refere a software (sistemas) desenvolvidos por grupos ao invés de indivíduos;
- Engenharia de Software usa princípios de engenharia ao invés de arte, e
- Engenharia de Software inclui tanto aspectos técnicos quanto não técnicos.

# Características da Engenharia de Software

- O principal objetivo da Engenharia de Software é produzir a um custo baixo, software de qualidade
  - Custo é fácil de ser medido
  - Qualidade não é
- O processo de planejamento é crucial na engenharia de software. A implementação é só uma parte do processo
- A Engenharia de Software engloba todo o ciclo de vida do Software (concepção, implementação, uso e manutenção)

# O que é Software?

- Há 30 anos, pouquíssima gente sabia explicar o que é software
- Hoje, praticamente todo mundo acha que sabe ...
- Software não é só os programas!
- A documentação necessária para instalar, usar e manter os programas são uma parte integrante do software

# Algumas características do Software

- O software é desenvolvido/projetos, ao invés de manufatura/fabricado;
- Software não se desgasta com o uso: não existem peças de reposição
- Software não pode ser visto ou tocado: para analisar o progresso de um projeto de software é preciso recorrer à sua documentação
- Grandes sistemas de software são normalmente desenvolvidos uma única vez. Assim, a experiência adquirida com outros projetos tem um valor limitado

# Algumas características do Software

- A maioria dos software é feita sob medida (por encomenda), ao invés de ser montada a partir de componentes
  - Produtos Genéricos: sistemas produzidos por uma organização e vendidos a todos os clientes que quiserem comprá-los
  - Produtos Customizados: sistemas que são encomendados e desenvolvidos para um determinado cliente
  - O maior gatilho de software é em produtos genéricos, mas o maior esforço de desenvolvimento está nos produtos customizados

# Atributos dos produtos de Software

- Manutenibilidade
  - Deve ser possível para o software evoluir de forma a atender a requisitos que mudam
- Depenbilidade
  - Software não deve causar prejuízo físico ou econômico no caso de uma falha
- Eficiência (Espaço *versus* Tempo)
  - Software não deve desperdiçar recursos do sistema
- Usabilidade
  - O software deve ter uma interface de usuário adequada e ser documentado

# Importância das características do produto

- A importância relativa destas características depende do produto e do ambiente em que ele será usado
- Em alguns casos, alguns dos atributos podem ser mais importantes
  - Em sistemas de segurança-críticos de tempo-real, os atributos chave podem ser dependabilidade e eficiência
- Os custos tendem a aumentar exponencialmente se altos níveis de um atributo são necessários

# O que é Software de Qualidade?

- É software que “funciona” (é confiável):
  - ele não deve falhar mais do que os especificado na documentação
- É software que funciona de acordo com a sua especificação
  - Mesmo software que *aparentemente* funciona pode não estar satisfazendo a sua especificação
- É software que é fácil de manter:
  - Código bem escrito
  - Documentação apropriada

# O que é Software de qualidade?

- É software que funciona de maneira eficiente
  - Software mais eficiente não é necessariamente software que roda mais rápido ou que gasta menos memória/disco
  - A complexidade do código e o custo também são fatores importantes
- É software que possui uma boa interface com o usuário
  - Muitos software não funcionam direito porque são difíceis de usar

# A Crise de Software

- O que é esta crise?
- Métodos de desenvolvimento de software existentes não são bons o bastante para o desenvolvimento de software de grande porte.

# A Crise de Software: crise?

- CRISE?
  - Momento perigoso ou decisivo
  - Decadência, carência, queda
  - Mas a chamada crise de software já dura quase 30 anos...
- DOENÇA?
  - No momento não se conhece uma *cura*, apenas paliativos para reduzir a “dor”
  - ....

# A Crise de Software - principais problemas

- Os grande software não funcionam adequadamente
  - Insatisfação dos usuários
- Os projetos de software estão sempre atrasados
- Os custos dos projetos de desenvolvimento de software são sempre maiores do que o previsto.

# A Crise de Software - principais problemas

- Os computadores estão cada vez mais rápidos, sofisticados e baratos;
- Os softwares estão cada vez maiores e mais sofisticados e a produtividade não acompanha a demanda
- Os custos com manutenção são muito altos: para sistemas com uma longa vida, eles são várias vezes maiores do que os custos de desenvolvimento

# A Crise de Software - outras dificuldades

- Dedicar-se pouco tempo à coleta de dados (requisitos dos clientes):
  - Normalmente apenas um subconjunto das necessidades do cliente são levadas em conta ...
  - Os profissionais estão sempre com muita pressa para começar a programar ...
- A qualidade geralmente é muito suspeita
  - Testes sistemáticos e tecnicamente completos raramente são feitos
  - A flexibilidade da maioria dos software também é bastante limitada

# A crise de Software - outras dificuldades

- Não muito interesse em se gastar tempo para se entender mais a respeito de estimativas, produtividade, precisão e eficácia de novos métodos e novas ferramentas, etc.
- A resistência a mudanças é grande ...

# A crise de Software - existe solução?

- O uso de melhores técnicas, métodos e ferramentas
- Mis treinamento e educação:
  - atualmente se investe muito pouco!

# Mitos do Software

- Propagam desinformação e confusão
- No passado eram tomados como verdades absolutas
- Ainda há resquícios: é difícil mudar hábitos antigos.
- Existem 3 tipos de mitos:
  - do Cliente
  - Administrativos
  - do Profissional

# Mitos do Software - Mitos do Cliente

- Uma declaração geral dos objetivos é suficiente para começar a escrever os programas: podemos preencher os detalhes mais tarde
- As necessidades do projeto mudam continuamente mas isto não é problema pois o software é flexível (custo não muda)

# Mitos do Software - mitos Administrativos

- Temos um manual de padrões e procedimentos para a construção de software: isto basta!
- Temos ferramentas de desenvolvimento de última geração pois compramos os computadores mais novos!
- Estamos atrasados no prazo: podemos tirar o atraso colocando mais programadores no projeto (não há necessidade de integração)

# Mitos do Software - mitos do Profissional

- Assim que escrevermos o programa e ele funcionar o nosso trabalho está terminado (não há necessidade manutenção)
- Enquanto o programa não estiver funcionando não como avaliar a sua qualidade (revisões técnicas não são necessárias)
- A única coisa a ser entregue em um projeto bem sucedido é o programa funcionando (documentação é descartada)

# Modelos de Ciclo de Vida de Software

- Se referem à progressão dos projetos de software, ao desenvolvimento e manutenção, e eventualmente à sua substituição
- Descrições abstratas do processo de desenvolvimento de software, mostrando as atividades e dados usados no ciclo de vida do software
- São consequência direta da crise de software e da necessidade de se ver o processo de desenvolvimento de software como uma engenharia

# O Processo de Software

- Um conjunto estruturado de atividades necessárias para o desenvolvimento de um sistema de software
  - Especificação
  - Projeto
  - Validação
  - Evolução
- Atividades variam com a organização e o tipo de sistema sendo desenvolvido
- Deve ser modelado explicitamente se o processo precisa ser gerenciado

# Modelos de Ciclo de Vida de Software

- Principais modelos ou processos
  - Modelo clássico (ou em cascata);
  - Programação exploratória;
  - Prototipagem (prototipação);
  - Transformação formal (ou refinamento);
  - Modelo espiral (ou baseado em riscos);
  - Técnicas de quarta geração.

# Características do Processo

- Entendível
  - O processo está definido e bem entendido?
- Visibilidade
  - O progresso do processo está externamente visível?
- Suportabilidade
  - O processo pode ser apoiado por ferramentas CASE?
- Aceitabilidade
  - O processo é aceito pelas pessoas envolvidas nele?

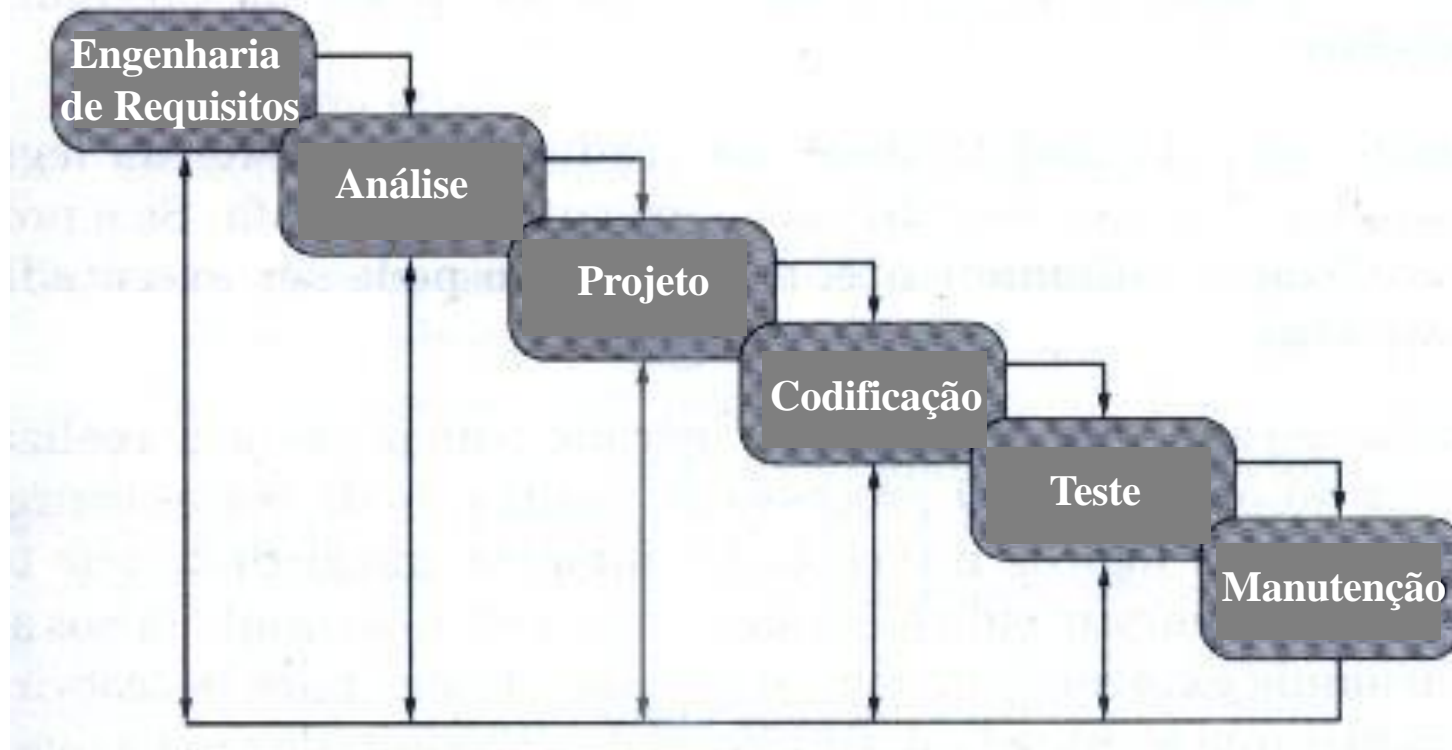
# Características do Processo

- Confiabilidade
  - Os erros no processo são descobertos antes deles resultarem em erros no produto
- Robustez
  - O processo pode continuar mesmo com problemas não esperados
- Manutenibilidade
  - O processo pode evoluir de forma a exigir necessidades organizacionais que evoluem
- Rapidez
  - Com que rapidez pode o sistema ser produzido

# Modelo Clássico (ou em Cascata)

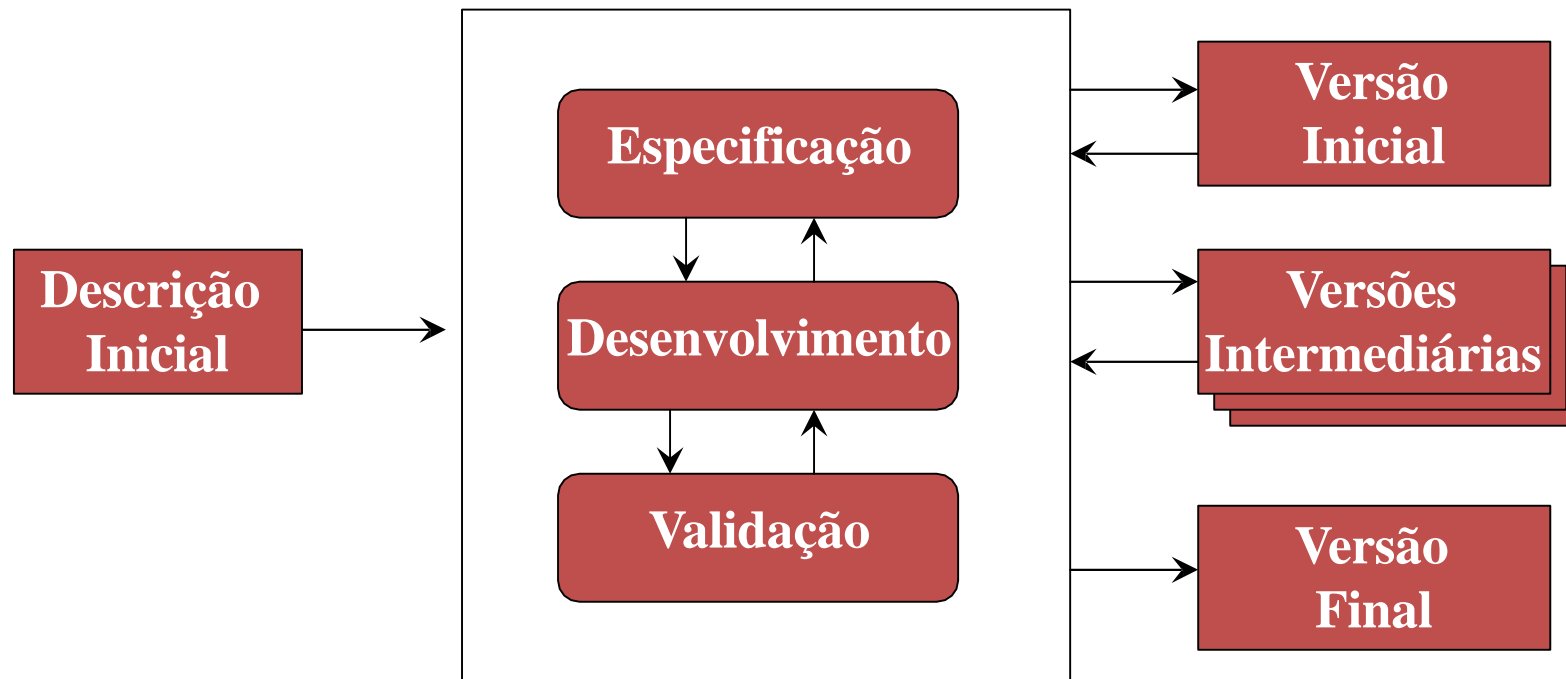
- Sua estrutura é composta de várias fases que são executadas de forma sistemática e sequencial
- Na prática, existe uma interação entre as fases e cada fase pode levar a modificações nas fases anteriores
- Este é o modelo mais antigo mas ainda o mais usado

# Modelo Clássico - fases



- Verificação: desenvolvimento correto?
- Validação: produto correto?

# Programação Exploratória (Desenvolvimento evolucionário)



OBRIGADO!

