


AN INTRODUCTION TO SOFTWARE ENGINEERING

Ian Sommerville, 8^o edição – Capítulo 1

Aula de Luiz Eduardo Guarino de Vasconcelos



"Se eu soubesse o que eu estava fazendo, não seria chamada pesquisa."
(Albert Einstein)

Objetivos

- Entender o que é engenharia de software e porque ela é importante
- Saber as respostas para questões-chave que fornecem uma introdução à Engenharia de Software
- Entender questões profissionais e éticas, relevantes para os engenheiros de software

Tópicos abordados



- Introdução (Histórico)
- FAQs sobre Engenharia de Software
- Responsabilidade profissional e ética

Introdução

- Onde os softwares são usados?
- Qual o impacto do software na sociedade?
- Quando desenvolver para web ou desktop?
- Como desenvolver?
- O que o software produz?
- O que levar em consideração ao desenvolver um software?
 - Layout, IHM, Padronização, LP, BD, Redes, Segurança, Qualidade, Público, Plataforma, SO, Multi-usuário
 - Pensar no hoje e no amanhã

Pesquisa – Capítulo 1

- Responda as questões e coloque **todas** as referências que você usou para cada uma.
- Entregar no fim do Capítulo 1, talvez seja na próxima aula (Individual)
 - ▣ Pesquise alguns, pelo menos 3, problemas causados por erros em software
 - ▣ Pesquise, pelo menos, 3 produtos de categorias diferentes onde o software faz a diferença
 - ▣ Qual a diferença de um software livre para open source?
 - ▣ O que é GNU GPL?
 - ▣ Que tipos de licenças de software existem?
 - ▣ O software livre impacta na sociedade de que forma?
 - ▣ Software livre resolve a pirataria? Explique.
 - ▣ Você já comprou algum software? Onde comprou? Como comprar? Você já pirateou software?
 - ▣ Como combater a pirataria?
 - ▣ Justifique se um usuário leigo está sujeito às penas da lei de proteção de software, anti-pirataria, etc.
 - ▣ Quem é responsabilizado quando se pratica pirataria de software em uma empresa?
 - ▣ Em quais circunstâncias é permitido ao usuário duplicar um programa de computador?
 - ▣ Posso instalar um programa legalmente adquirido em mais de um microcomputador?
 - ▣ Como ganhar dinheiro vendendo software?

Evolução do software

- Nos primeiros anos (1950)
 - ▣ Foco no hardware (maior custo)
 - ▣ Software desenvolvido sem administração e métodos
 - ▣ Software projetado sob medida para a aplicação
 - Quase não havia “produto de software”
 - Quem desenvolvia, usava e alterava o software
 - ▣ O projeto ficava na cabeça do desenvolvedor
 - Não havia documentação
 - O que acontecia se um desenvolvedor pedisse demissão?

Evolução do software

- Segunda fase (meados dos anos 60)
 - ▣ Interatividade, sistemas de tempo real
 - ▣ Uso de diferentes bibliotecas
 - Necessidade de adaptar (ao cliente ou ao hardware) e manter milhares de linhas de código
 - Surgem as atividades de manutenção de software
 - ▣ A manutenção passou a absorver muitos

Evolução do software

- Terceira fase (anos 70)
 - ▣ Redes, distribuição e concorrência
 - ▣ Generalização dos computadores pessoais
 - ▣ Crescimento das empresas de software
 - As empresas passaram a vender até 100 vezes mais produtos de software
 - ▣ Muitos passaram a gastar mais dinheiro com software do que com o computador

Evolução do software

- Quarta fase (atualmente)
 - ▣ Tecnologia orientada a objetos
 - ▣ Desktops poderosos
 - ▣ Redes neurais (IA)
 - ▣ A capacidade de construir software não acompanha o ritmo da demanda
 - ▣ Projetos ruins e o uso de recursos inadequados ameaçam a capacidade de manter os softwares existentes

Evolução do hardware

- Desenvolvimento de hardware
 - ▣ Era o maior custo
 - ▣ Uso de padrões técnicos (análise e projeto)
 - ▣ Emprego de controles, métodos e ferramentas (engenharia de software)

Evolução do software

- Desenvolvimento de software
 - ▣ Programação por tentativa e erro
 - ▣ Falta de métodos para o desenvolvimento
 - ▣ Hoje seu custo é bem maior que o do hardware
- Lamentações:
 - ▣ Por que demora tanto tempo para que os programas sejam concluídos?
 - ▣ Por que os custos são tão elevados?
 - ▣ Por que não descobrimos todos os erros antes de entregarmos o software ao nosso cliente?
 - ▣ Porque temos dificuldade em medir o progresso enquanto o software está sendo desenvolvido?

Conseqüências

- Aplicações escritas há 20 anos, modificadas ao longo do tempo, são impossíveis de manter.
- Pequenas alterações podem fazer o sistema falhar.
- Não há quem conheça estes sistemas
 - ▣ Falta de documentação do desenvolvimento
- Sistemas críticos (tráfego aéreo) de funcionamento “estranho” não são substituídos
 - ▣ Não há sistemas para substituí-los

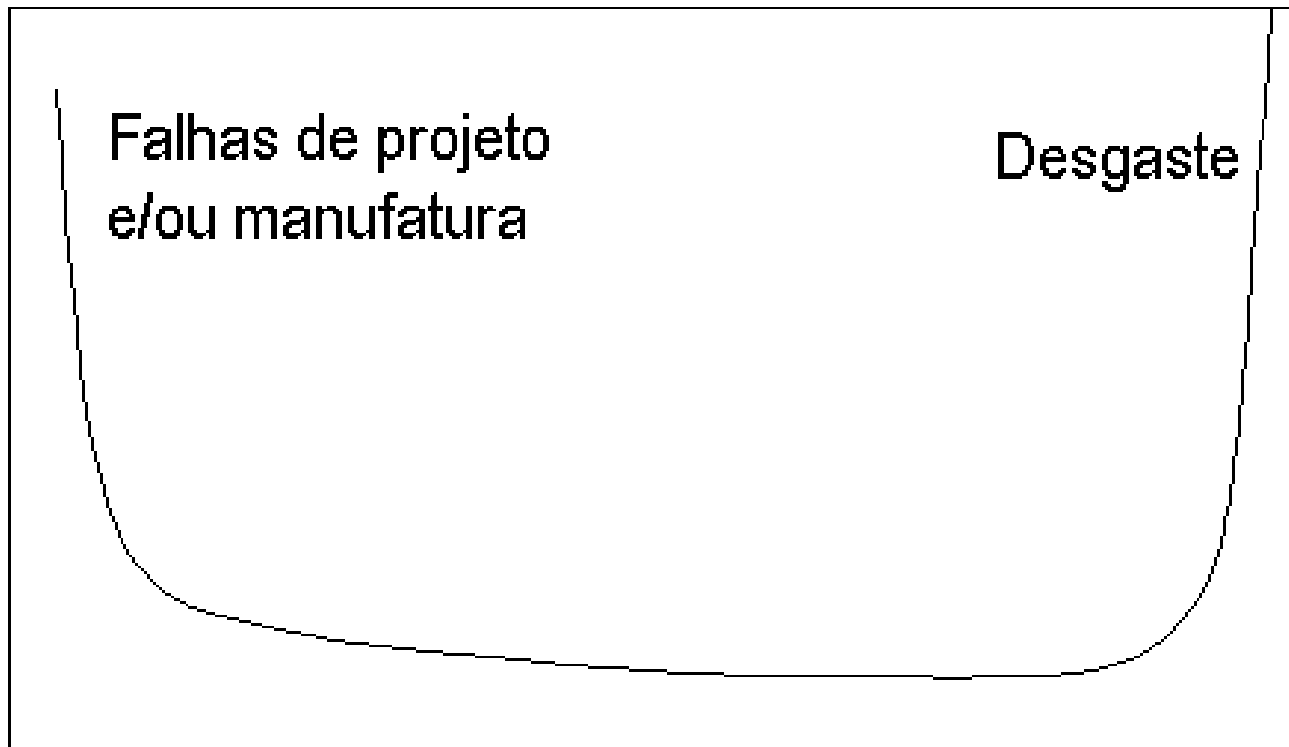
Curva de falhas para o hardware

Índice de falhas

Falhas de projeto e/ou manufatura

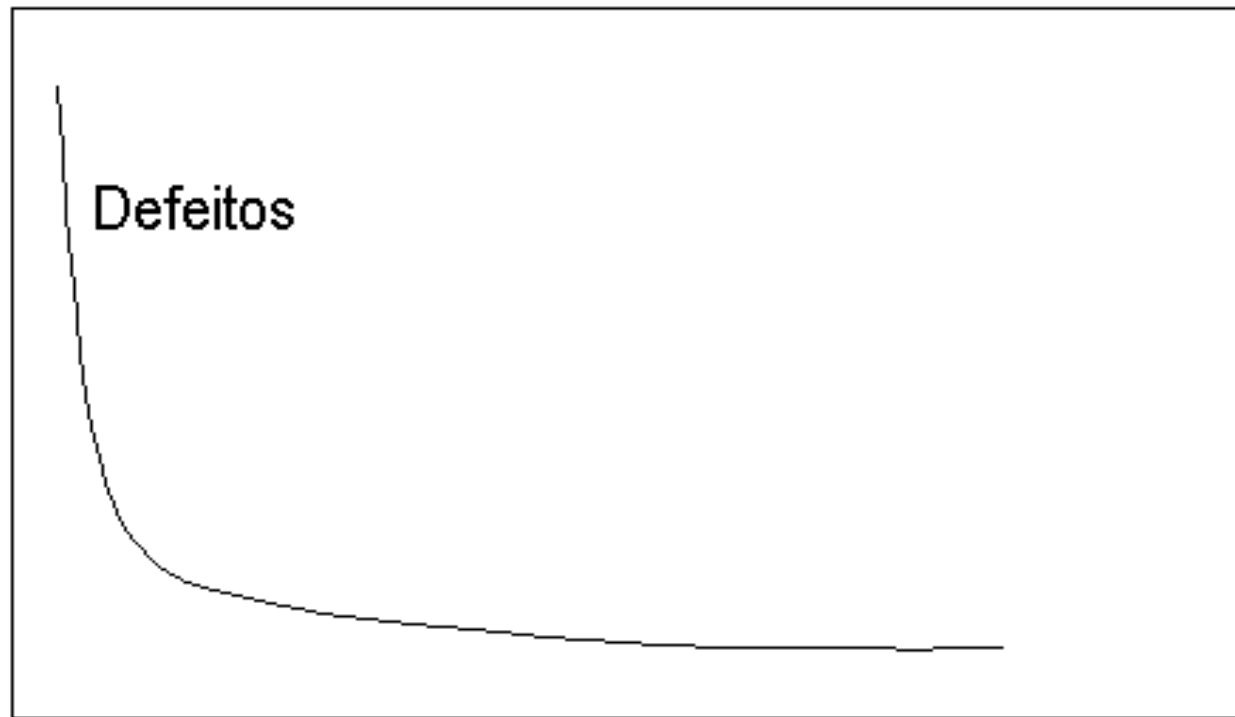
Desgaste

Tempo



Curva de falhas para o software

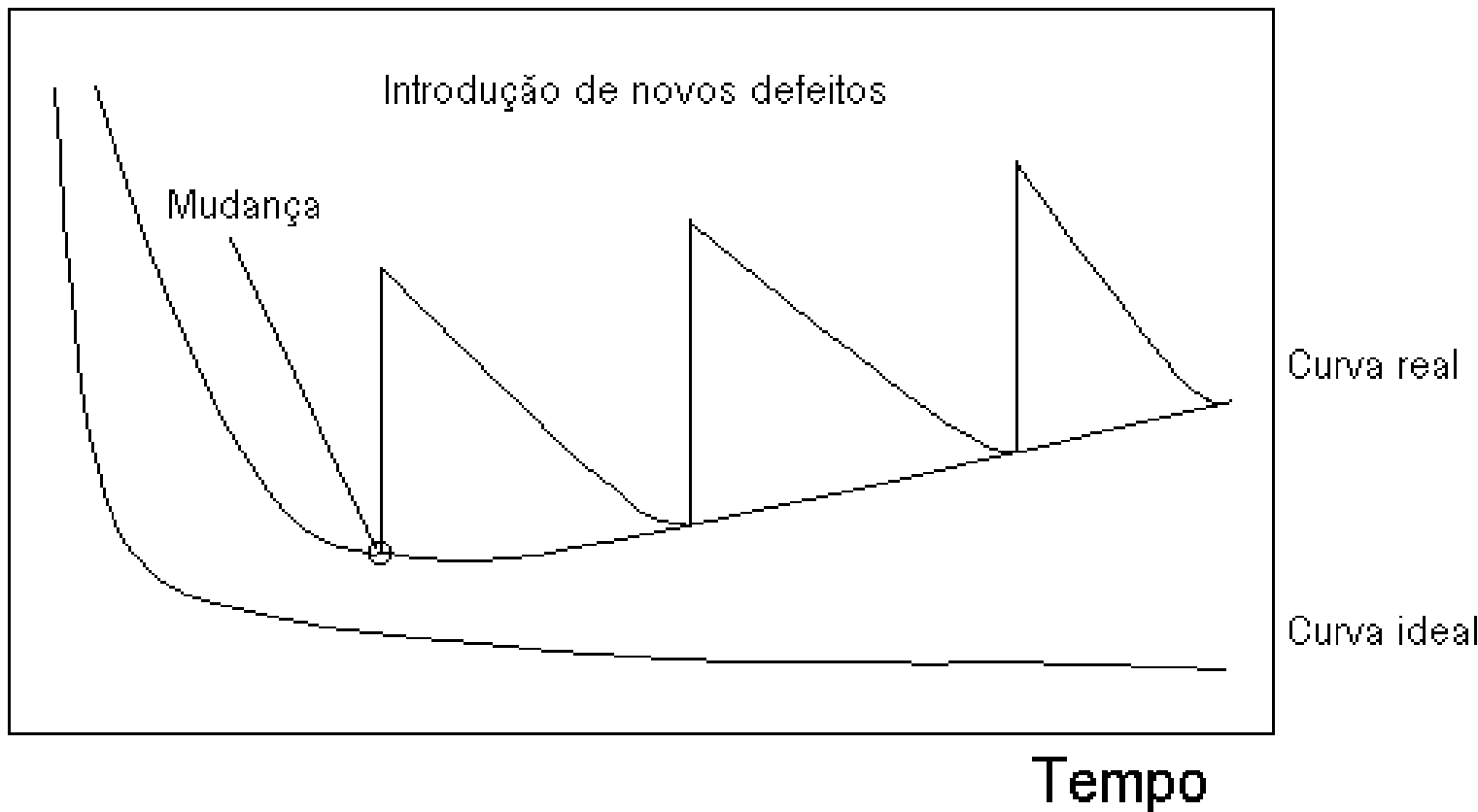
Índice de falhas



Tempo

Curva de falhas real para o SW

Índice de falhas



Mitos do software - Gerente

- Temos um manual repleto de padrões e procedimentos que oferecerá tudo o que os desenvolvedores precisam saber
 - ▣ Ele é usado?
 - ▣ As técnicas são adequadas ao software sendo desenvolvido?
 - ▣ Ele é completo?
- Temos os computadores mais modernos do mercado
 - ▣ Ferramentas (SW) que auxiliam o desenvolvimento são mais úteis
- Se o cronograma está atrasado adicionaremos mais programadores para tirar o atraso
 - ▣ Perda de produtividade na educação dos novos programadores

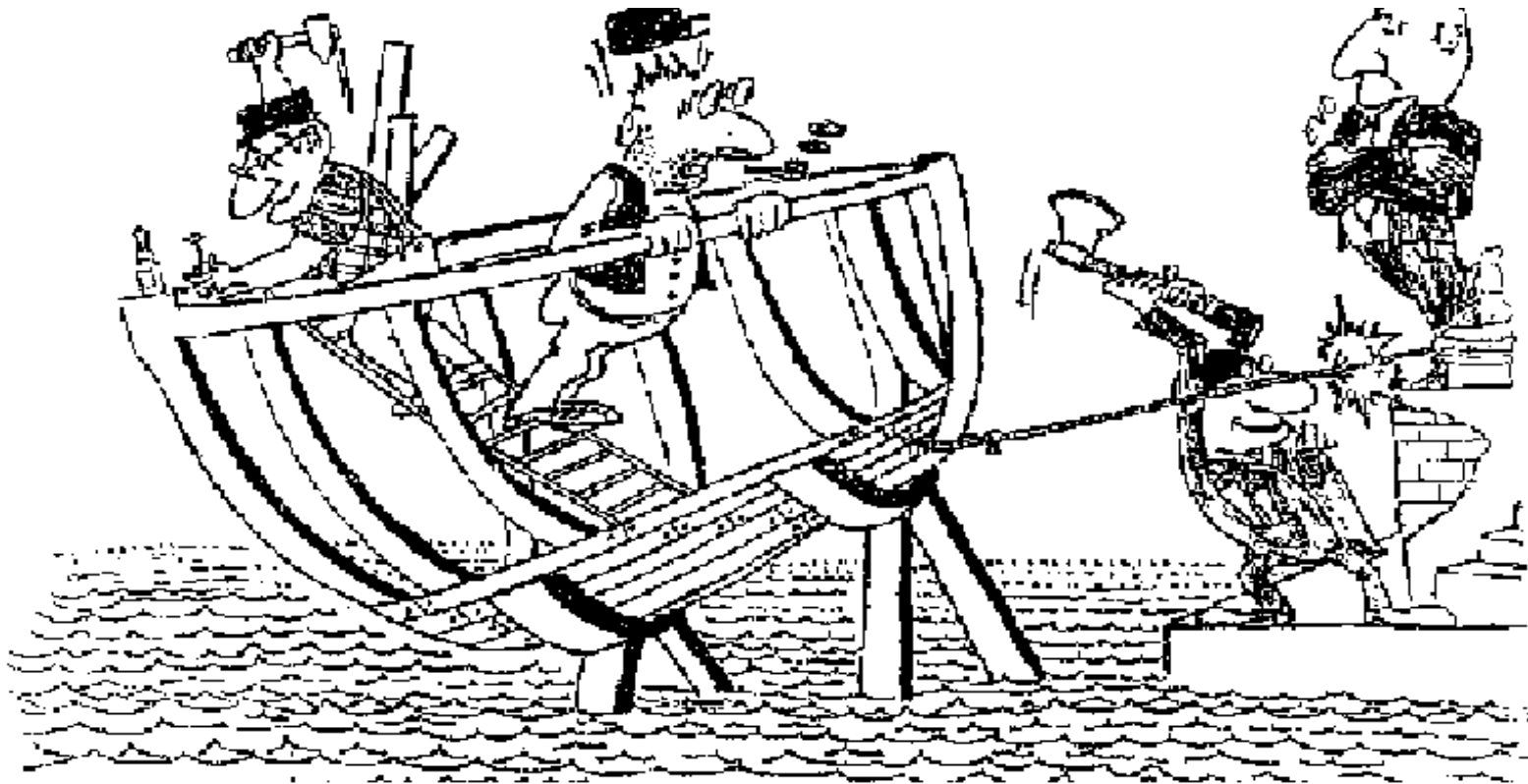
Mitos do software - Cliente

- Uma declaração geral dos objetivos é suficiente para começar a escrever o programa. Os detalhes podem ser preenchidos mais tarde.
 - ▣ Definição inicial ruim
 - ▣ Principal causa do fracasso de projetos de desenvolvimento de software
- Os requisitos modificam-se continuamente, mas as mudanças podem ser facilmente acomodadas.
 - ▣ Pouca atenção à definição inicial dos requisitos
 - ▣ O impacto negativo das mudanças é maior conforme o projeto vai se desenvolvendo
 - Definição (1x)
 - Implementação (1,5x a 6x)
 - Manutenção (60x a 100x) Pode ser mais barato desenvolver outro SW

Mitos do software - Desenvolvedor

- Depois de escrever e colocar o programa em funcionamento nosso trabalho estará terminado
 - ▣ De 50% a 70% do esforço é gasto depois de entregar a primeira versão do sistema ao cliente
- Enquanto não tiver o programa funcionando não posso medir sua qualidade
 - ▣ A qualidade não deve ser garantida por testes, mas por todas as atividades de desenvolvimento
- A única coisa a ser entregue em um projeto bem-sucedido é o programa funcionando
 - ▣ A documentação bem feita é crucial para as manutenções futuras

Não estabeleça prazos audaciosos demais



Prazo é prazo !

Sempre ouça o mercado



Usuários odeiam bugs



Experiência em simulação ajuda



Nem toda apresentação será um sucesso



O que serve para um cliente pode não servir para outro



Busque soluções eficientes



SUPORTE! Ferramentas diferentes para situações diferentes



Ajuda On-Line pode ser útil



Enfermeira, acesse a internet, vá até cirurgia.com e clique no ícone 'Você está totalmente perdido'.

Previsão e otimização podem ser complexas

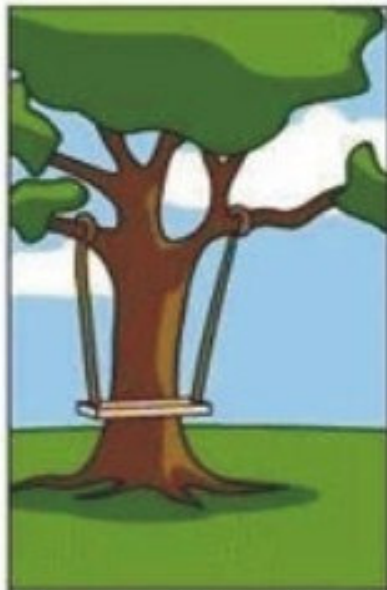


Observe os atributos significativos do seu cliente





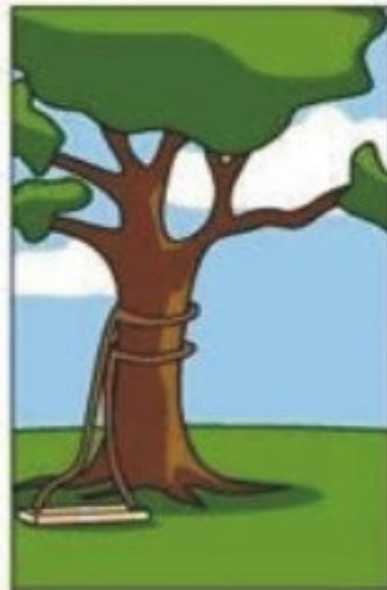
Como o cliente explicou...



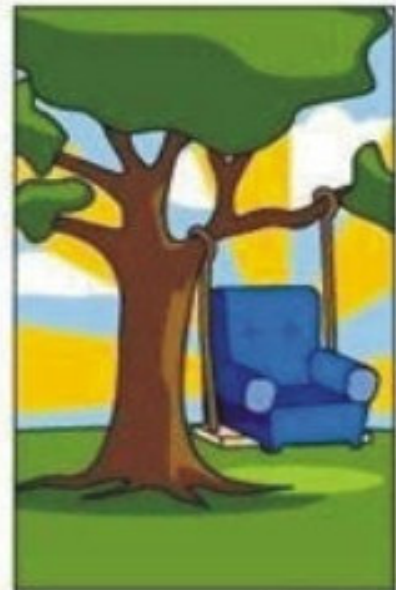
Como o líder de projeto entendeu...



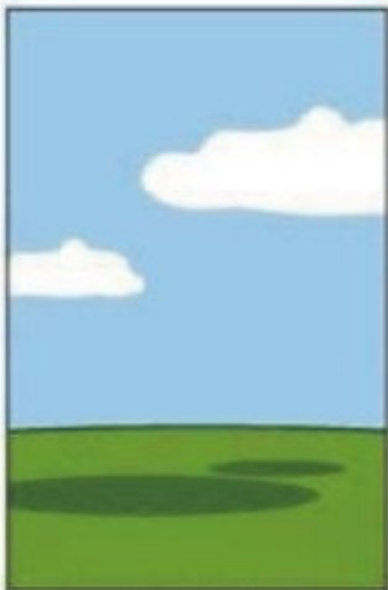
Como o analista projetou...



Como o programador construiu...



Como o consultor de negócios descreveu...



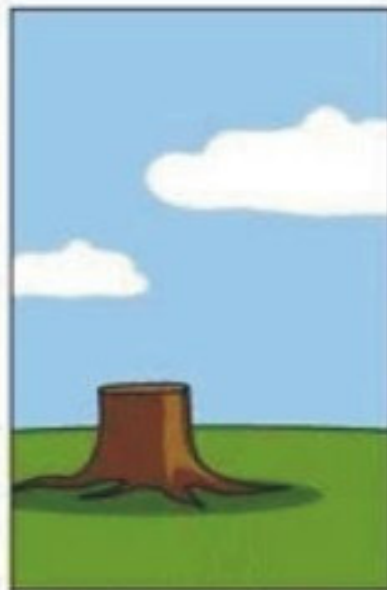
Como o projeto foi documentado...



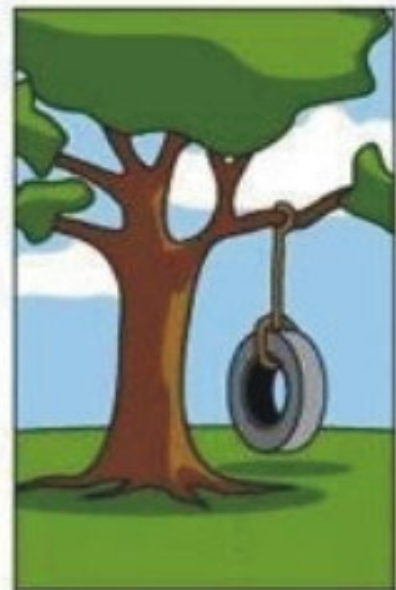
Que funcionalidades foram instaladas...



Como o cliente foi cobrado...



Como foi mantido...



O que o cliente realmente queria...

Dados

- Taxa de Sucesso → 16,2%
 - Atrasados ou Acima do Orçamento → 52,7 %
 - Cancelados → 31,1 %
-
- Alguns anos depois (**Pesquisa em 7500 projetos**)
 - Taxa de sucesso → 18,1%
 - Atrasados ou Acima do Orçamento → 51,4%
 - Cancelados → 30,5 %

Denver International Airport

- Custo do projeto: US\$ 4.9 bilhões
 - ▣ 100 mil passageiros por dia
 - ▣ 1,200 vôos
 - ▣ 53 milhas quadradas
 - ▣ 94 portões de embarque e desembarque
 - ▣ 6 pistas de pouso / decolagem
- Erros no sistema automático de transporte de bagagens (*misloaded, misrouted, jammed*):
 - ▣ Atraso na abertura do aeroporto com custo total estimado em US\$360 Milhões
- 86 milhões para consertar o sistema

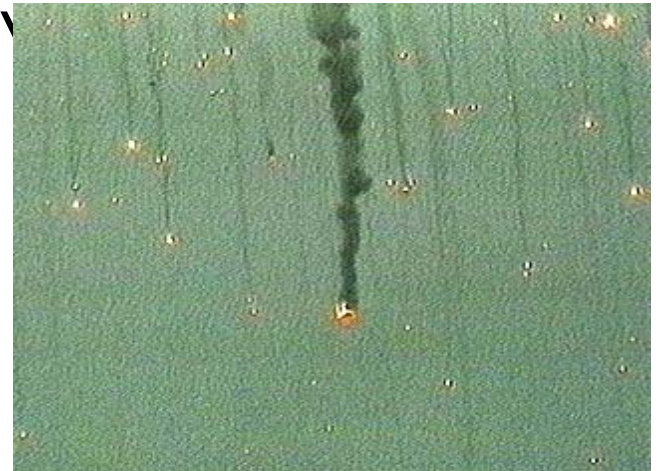
Ariane 5

- Veículo Lançador de Satélites
- Projeto da Agência Espacial Européia que custou:
 - ▣ 10 anos.
 - ▣ US\$ 8 Bilhões.
- Capacidade 6 toneladas.
- Garante supremacia européia no espaço.



Resultado = Fogos de Artifício

- ❑ Explode 37 segundos após seu lançamento.
- ❑ Falha no sistema (software) que calculava a trajetória e atitude, leva a uma pane nos sistemas adjacentes. Estes, enviam sinais de diagnóstico para os motores que os interpreta como dados comuns.....
- ❑ Destruição do foguete e carga avaliada em US\$ 500 milhões
- ❑ Falha de TesteBOOM.



Outros

- VLS (Alcântara)
- Família Windows
- Diversos softwares (com atualizações constantes ...)

Por quê?



□ Sintomas

- Compreensão incompleta ou imprecisa das necessidades do usuário
- Inabilidade de lidar com requisitos que evoluem
- Módulos incompatíveis
- Dificuldades de estender ou manter software
- Descoberta de defeitos graves no projeto em etapas avançadas de desenvolvimento ou mesmo em época de implantação ou uso
- Desempenho inaceitável do software
- Falta de coordenação na equipe

Por quê?

□ **Causas frequentes**

- Gerência de requisitos sem processo definido
- Comunicação ambígua e imprecisa entre partes envolvidas
- Complexidade crescente
- Inconsistências não detectadas em nível de análise, projeto e implementações
- Testes insuficientes
- Dificuldade em lidar e gerenciar riscos
- Falta de controle sobre propagação de mudanças
- Automação insuficiente
- Ubiquidade (disponível o tempo todo em qualquer lugar)
- Diversidade de plataformas
- Comunicação entre o cliente e o desenvolvedor é muito fraca.



- **Mas nem tudo está perdido**

- Sistemas complexos e grandes foram, e estão sendo, desenvolvidos.

- Simuladores de aeronaves, veículos

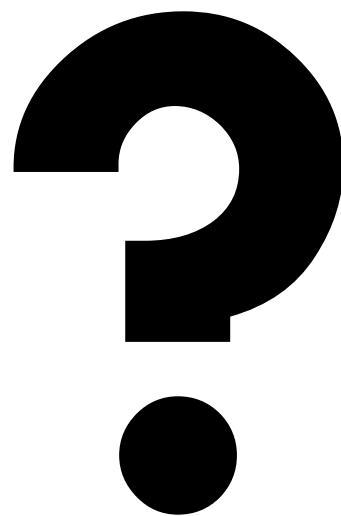
- Telemetria, processamento em tempo real

- Geoprocessamento

- Construção de plataformas, edificações

- etc, etc, etc

O que fazer?



Engenharia de Software - Resumo

- As economias de todos os países dependem de sistemas complexos baseado em computadores.
- Mais e mais sistemas são controlados por software
- Engenharia de Software se concentra nas teorias, métodos e ferramentas para desenvolvimento de software profissional.
- Despesas com software representam uma fração significativa dos gastos de todos os países desenvolvidos
- Crise de software – 1968. Apresentação do conceito de Eng. Software.
- Softwares complexos, altos custos, atrasos → necessidade de técnicas, processos, teorias → ES

Custos com Software

- ❑ Custos com software geralmente dominam os custos com sistemas computacionais. O custo de software em um PC é normalmente maior que o custo de hardware.
- ❑ Custos para manter softwares são maiores do que para desenvolver. Em sistemas com longo ciclo de vida os custos com manutenção podem ser muito maiores do que os custos de desenvolvimento.
- ❑ Engenharia de software foca no custo efetivo do desenvolvimento do software, através de técnicas ...

FAQs sobre Engenharia de Software

- ❑ O que é software?
- ❑ O que é engenharia de software?
- ❑ Qual a diferença entre engenharia de software e ciência da computação?
- ❑ Qual a diferença entre engenharia de software e engenharia de sistemas?
- ❑ O que é um processo de software?
- ❑ O que é um modelo de processo de software?

FAQs about software engineering

- ❑ Quais são os custos da engenharia de software?
- ❑ Quais são os métodos da engenharia de software?
- ❑ O que é CASE (Computer-Aided Software Engineering)?
- ❑ Quais são os atributos de um bom software?
- ❑ Quais são os desafios-chave da engenharia de software?

O que é software?

- Programas de computador E **documentação associada** com os requisitos, modelos de projeto e manuais de usuário, etc.
- Produtos de software podem ser desenvolvidos para cliente particular ou para mercado geral.
- Produtos de software podem ser
 - Genéricos – desenvolvidos para qualquer cliente. e.g. Software para PC como Excel ou Word. Especificação é da organização.
 - Personalizado (sob encomenda) – desenvolvido para um cliente específico de acordo com as especificações. Especificação é do cliente.
- Novo software pode ser criado configurando um software genérico ou reusando softwares existentes. e.g. SAP, web services, etc.

O que é engenharia de software?

- Eng. Software é uma disciplina da engenharia que esta concentrada em todos os aspectos da produção de software (Sommerville, 2007)
- “Disciplina que integra processo, métodos e ferramentas para o desenvolvimento de software”. (Pressman, 95)
- Engenheiros de Software devem adotar uma abordagem sistemática e organizada para trabalhar e usar ferramentas e técnicas apropriadas dependendo de cada problema a ser solucionado (considerar plataforma, SO, escopo, imediatismo, etc). (Sommerville, 2007)
- O estabelecimewnto e uso de sólidos princípios de engenharia para que se possa obter economicamente um software que seja confiável e que funcione eficientemente em máquinas reais.” (Fritz Bauer, 1969)

Qual a diferença entre engenharia de software e ciência da computação?

- ❑ Ciência da computação se concentra nas teorias e métodos fundamentais da computação;
- ❑ Eng. Software se concentra nos problemas práticos da produção de software.
- ❑ Ideal é que engenheiros de software se apoiassem nas teorias da computação, mas na realidade isso é complicado.

Qual a diferença entre engenharia de software e engenharia de sistemas?

- Engenharia de sistemas se concentra em todos os aspectos de sistemas baseados em computadores como hardware, software, políticas e processos de engenharia. Engenharia de Software é parte deste processo voltado ao desenvolvimento de software , das aplicações, dos bancos de dados, das UIs.
- Engenheiros de software são envolvidos na especificação de sistemas, projeto arquitetural, integração e distribuição.

O que é um processo de software?

- Conjunto de atividades com o objetivo de **desenvolver ou evoluir um software**.
- As **atividades genéricas** de todos os processos de software são:
 - ▣ Especificação – o que p sistema deverá fazer e quais as restrições de desenvolvimento
 - ▣ Desenvolvimento – produção do sistema de software, projetado e programado
 - ▣ Validação – verifica se o software é o que o cliente deseja
 - ▣ Evolução – mudanças do software em função da demanda, que podem ser requisitos dos clientes ou do mercado

O que é um modelo de processo de software?

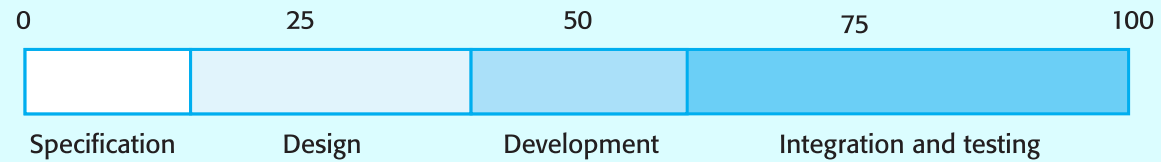
- Uma **representação simplificada** do processo de software que apresenta a visão do mesmo.
- Exemplos de perspectivas/modelos de processos são
 - ▣ Modelo de Workflow – sequência de atividades;
 - ▣ Modelo de fluxo de dados – fluxo da informação, como entrada é transformada numa saída;
 - ▣ Modelo de papel/ação – quem faz o que.
- **Modelos de processos** genéricos
 - ▣ Cascata;
 - ▣ Desenvolvimento iterativo;
 - ▣ Engenharia de software baseada em componente.

Quais são os custos da engenharia de software?

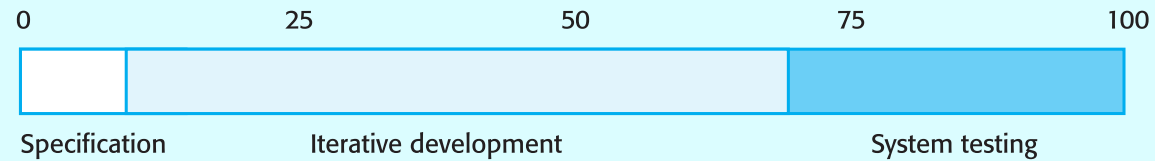
- ❑ Aproximadamente 60% do custo é para desenvolvimento, 40% para testes, mas isso pode variar.
- ❑ Para softwares personalizados os custos com evolução podem exceder os custos de desenvolvimento e são difíceis de planejar.
- ❑ Custos variam e dependem do processo e do tipo de sistema a ser desenvolvido, além dos atributos (imediatismo, performance, plataforma, segurança, disponibilidade, etc).
- ❑ Distribuição do custo depende do modelo que está sendo usado.

Distribuição de custo nas atividades de ES

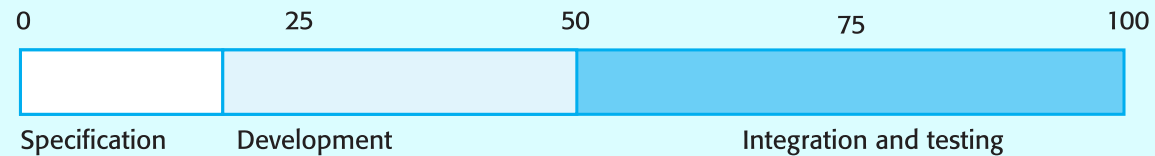
Waterfall model



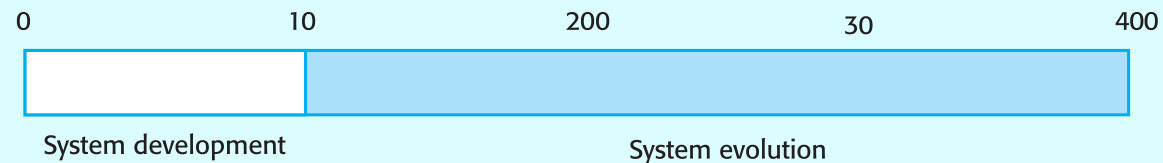
Iterative development



Component-based software engineering



Development and evolution costs for long-lifetime systems



○ que são métodos de engenharia de software?

- Abordagem estruturada para desenvolvimento do software que incluem modelos de sistemas, notações, regras, , guias de processos, recomendações, etc.
- Descrições de Modelos de sistemas
 - ▣ **Descrições de modelos gráficos** que devem ser produzidos. Ex.: Modelo de objetos, fluxo de dados, máquina de estado;
- **Regras**
 - ▣ Restrições aplicadas aos modelos. Ex.: toda entidade deve ter um único nome;
- **Recomendações**
 - ▣ Heurísticas que caracterizam boa prática de projeto nesse método. Ex: nenhum objeto deve ter mais que sete subobjetos;
- **Guia de processo**
 - ▣ Quais atividades devem ser seguidas. Ex.: atributos de objetos devem ser documentados antes de definir as operações do mesmo.

O que é CASE (Computer-Aided Software Engineering)

- Sistemas de Software que são usados para dar apoio às atividades do processo de software.
- Sistemas CASE são usados em métodos de ES, como editores, geradores de código, etc.
- Upper-CASE
 - ▣ Ferramentas que suportam as atividades iniciais do processo dos requisitos ao projeto;
- Lower-CASE
 - ▣ Ferramentas que suportam atividades finais do processo como programação, debugging e testes.

Quais são os atributos de um bom software?

- O software deve dispor das funcionalidades requeridas, além de performance para o usuário e fácil manutenção, usabilidade e confiança.
- Maintainability / Fácil manutenção
 - Software deve ser escrito de modo que possa evoluir de acordo com as mudanças necessárias;
- Dependability / Confiança
 - Software deve ter trustworthy (confiança). Não deve causar danos físicos ou econômicos em caso de falhas (COMPLICADO);
- Efficiency / Eficiência
 - Software não deve desperdiçar os recursos do sistema;
- Acceptability / Usabilidade
 - Software deve ser usável, sem esforço excessivo, pelo tipo de usuário para o qual foi projetado

Quais são os desafios-chave da engenharia de software?

- Heterogeneidade, entrega e confiança.
- Heterogeneity
 - ▣ Desenvolver técnicas para operar em sistemas distribuídos em plataformas e ambientes de execução diferentes;
- Delivery
 - ▣ Desenvolver técnicas para diminuir os tempos de entrega dos sistemas grandes e complexos sem comprometer a qualidade;
- Trust
 - ▣ Desenvolver técnicas que demonstrem que o software pode ter a confiança dos usuários.

Responsabilidade ética e profissional

- Engenharia de software limita a liberdade dos engenheiros que implicam em responsabilidades mais amplas do que a aplicação de habilidades técnicas.
- Engenheiros de software devem se comportar de forma honesta, responsável ética e moralmente para serem respeitados como profissionais
- Comportamento ético não está limitado por leis

Características da responsabilidade profissional

- Confidencialidade
 - ▣ Devem respeitar a confidencialidade dos clientes ou funcionários, independente de ter ou não assinado um acordo formal.
- Competência
 - ▣ Não deve aceitar trabalho que esteja fora do seu alcance.

Características da responsabilidade profissional

- Direitos sobre propriedade intelectual
 - ▣ Deve estar ciente das leis locais que regem o uso de propriedade intelectual tais como patentes e direitos autorais, para assegurar a devida proteção a funcionários e clientes.
- Mau uso de computadores
 - ▣ Não deve usar suas habilidades técnicas para mau uso (ex.: execução de jogos nas máquinas de funcionários e clientes até disseminação de vírus).

ACM/IEEE Código de Ética

- <http://www.acm.org/>
- <http://www.ieee.org.br/>
- As sociedades profissionais tem cooperado para produzir um código de ética e de prática profissional.
- O Código contém 8 Princípios que relacionam comportamento e decisões sobre o engenheiro de software profissional que inclui educadores, gerentes, supervisores, desenvolvedores de política, estagiários e estudantes

Código de ética - Preâmbulo

□ Preâmbulo

- A versão resumida do código apresenta as aspirações em um alto nível de abstração: as cláusulas que estão incluídas na versão completa fornecem exemplos e detalhes de como essas aspirações mudam a maneira como agimos como profissionais de engenharia de software. Sem essas aspirações, os detalhes podem se tornar muito específicos e tediosos; sem os detalhes, as aspirações podem se tornar aparentemente importantes, mas vazias; juntos, aspirações e detalhes formam um código coeso.
- Os engenheiros de software devem se comprometer a fazer da análise, especificação, desenvolvimento, testes e manutenção de software uma profissão benéfica e respeitada. De acordo com seu comprometimento com a saúde, segurança e bem-estar do público, os engenheiros de software devem aderir aos seguintes Oito Princípios:

Código de ética - Princípios

□ PÚBLICO

- Engenheiros de software devem agir consistentemente com o interesse público.

□ CLIENTE E EMPREGADOR

- Os Engenheiros de software devem agir dentro dos melhores interesses de seu cliente e empregador, de forma consistente com o interesse público.

□ PRODUTO

- Engenheiros de software devem assegurar que seus produtos e modificações a eles relacionadas atendam aos mais altos padrões profissionais possíveis.

Código de ética - Princípios

□ JULGAMENTO

- ▣ Engenheiros de software devem manter a integridade e a independência do seu julgamento profissional.

□ GERENCIAMENTO

- ▣ Gerentes e líderes de engenharia de software devem aceitar e promover uma abordagem ética no gerenciamento de desenvolvimento e manutenção de software.

□ PROFISSÃO

- ▣ Engenheiros de software devem promover a integridade e a reputação da profissão de forma consistente com o interesse público.

Código de ética - Princípios

□ COLEGAS

- ▣ Engenheiros de software devem ser honestos e colaborativos com seus colegas.

□ INDIVÍDUO

- ▣ Engenheiros de software devem participar, ao longo da vida, aprendendo, respeitando e promovendo uma abordagem ética na prática da profissão

Dilemas éticos

- ❑ Discordar, em princípio, das políticas da alta gerência.
- ❑ Seu empregador age sem ética e entrega versões de sistemas críticos sem finalizar os testes.
- ❑ Participação de desenvolvimento militar ou nuclear.
- ❑ Você deve ter seu entendimento. A posição ética depende inteiramente do ponto de vista dos indivíduos envolvidos

Pontos-chave

- ❑ Engenharia de software é uma disciplina relacionada a todos os aspectos de produção de software
- ❑ Os produtos de software consistem em programas desenvolvidos e documentação associada. Os atributos essenciais do produto são: fácil manutenção, confiança, eficiência e aceitação (usabilidade)
- ❑ O processo de software inclui todas as atividades envolvidas no desenvolvimento ou evolução de software. Especificação, Desenvolvimento, Validação e Evolução fazem parte de todos os processos de software
- ❑ Métodos são meios organizados de produção de software.

Pontos-chave



- ❑ Ferramentas CASE são sistemas de software desenvolvidos para apoiar o desenvolvimento de software
- ❑ Engenheiros de software tem responsabilidades com a profissão e com a sociedade. Não devem se preocupar apenas com as questões técnicas.
- ❑ Sociedades profissionais publicam códigos de conduta que definem os padrões de comportamento esperados de seus membros