USP - Universidade de São Paulo

IME - Instituto de Matemática e Estatística

Introdução

Prof. Marco Aurélio Gerosa
gerosa@ime.usp.br

Sobre mim

- Graduação (1999)
  - Engenharia de Computação – Universidade Federal do Espírito Santo (UFES)
- Mestrado (2002)
  - Informática, na área de Sistemas Colaborativos – PUC-Rio
- Doutorado (2006)
  - Informática, na área de Engenharia de Software – PUC-Rio
- Docente do IME (desde 2008)
- Desenvolve sistemas desde 1988, profissionalmente desde 1995
- Coordenador do projeto de software livre Groupware Workbench
- Pesquisas em
  - desenvolvimento baseado em componentes, sistemas Web 2.0, sistemas colaborativos, evolução de software, mineração de repositórios, coreografia de serviços web, internet do futuro

- Espero contribuir para que vocês aprimorem seus conhecimentos sobre Engenharia de Software.

Sobre a disciplina

- **Objetivo:**
  - Compreensão das atividades que compõem o processo de desenvolvimento de software e seus propósitos. Estudo de aplicação de princípios de gerenciamento das atividades e seu impacto no andamento do projeto e no produto final.
  - Apresentar os conceitos básicos da Engenharia de Software e levar os alunos a vivenciarem algumas de suas práticas.

- **Perfil do aluno:**
  - São desejáveis conhecimentos básicos de orientação a objetos, Java e programação para Web.
  - É imprescindível tempo extraclasse para se dedicar às atividades do curso.

- **Ementa:**
  - Gerenciamento de projeto. Estimação de custos. Análise e especificação de requisitos. Especificações formais. Interface com o usuário. Modelagem de dados. Técnicas e modelagens para projeto e implementação: arquitetura de projeto, projeto estruturado, projeto orientado a objetos. Gerenciamento de versões e configurações. Verificação: testes, revisões e inspeções. Validação e certificação de qualidade. Manutenção. Documentação

Abordagem pedagógica

- Enfoque da disciplina
  - Entregar informação aos alunos (pizza delivery) x fazer os alunos aprenderem
- *"Escutei e esqueci,*
  li e entendi,
  fiz e aprendi."
  - Implementação
  - Várias atividades
- Atividades em grupo realizadas de forma colaborativa e individual

- *O mérito do sucesso é 10% do professor, 10% dos colegas e 80% de si próprio.*

Avaliação

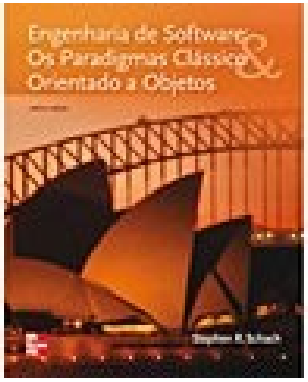
- Provas
- Projeto
- Listas de exercícios
- Trabalhos

Questionário

1. Qual o seu nome e período?
2. Você está trabalhando no momento? Onde e com que?
3. Qual a sua experiência com desenvolvimento de software?
4. O que você espera aprender nesta disciplina?

Introdução

Referências desta aula

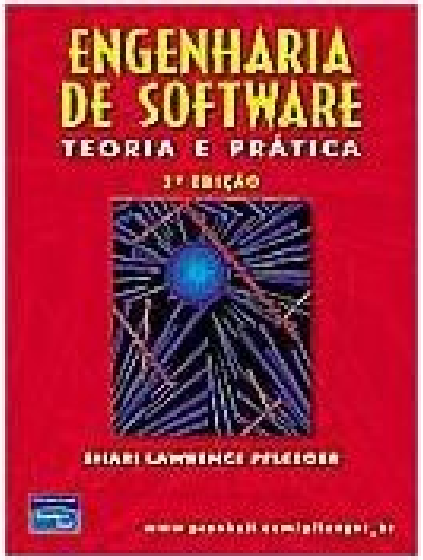


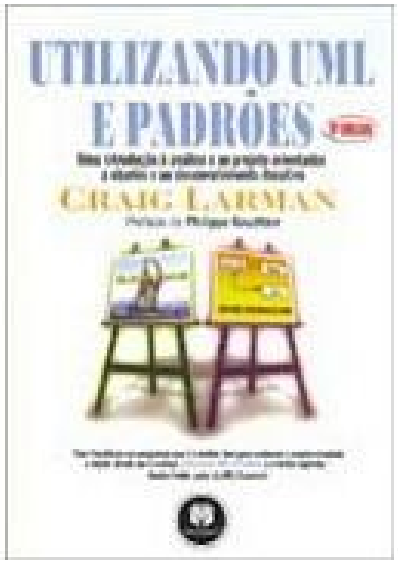
Schach, Cap 01

Pressman, Cap 01

- Experiência pessoal e outras leituras

Outras referências



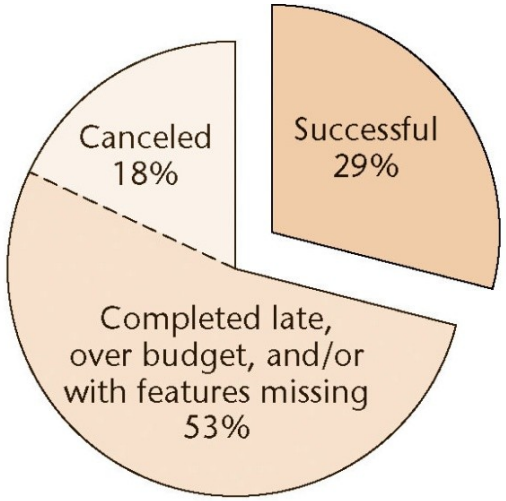


Questões a serem discutidas

- Por que é necessário uma disciplina de “engenharia de software”, não basta “desenvolvimento de algoritmos”?
- O que deve tratar a Engenharia de Software?
- Quais os problemas do desenvolvimento de software atualmente?
- Qual a importância do software na sociedade moderna?
- Todo o conhecimento da área de Engenharia de Software está dentro da grande área Computação e Informática?
- Há quantos anos existe o termo Engenharia de Software?
- Quais as particularidades da Engenharia de Software com relação às demais engenharias? Ela poderia ser uma das engenharias da Poli?
- A Engenharia de Software é determinística? Se dermos o mesmo problema para dois engenheiros, eles chegam na mesma solução?

1.1 Por que uma Engenharia de Software?

- *"Programar é divertido, porém desenvolver software de qualidade é difícil." [Craig Larman*]
- Software é entregue:
  - depois do prazo
  - acima do orçamento,
  - com falhas
  - não atende a necessidade do cliente
- Pesquisa Stanish Group (2004) com 9.236 projetos



Falhas

- Algumas falhas notáveis
  - 1979 – O sistema de defesa americano disparou um alarme sobre um ataque de mísseis da União Soviética
  - 1985 – Devido a uma falha no software de um equipamento médico, pacientes morreram devido a overdose de radiação
  - 1991 – Na Guerra do Golfo, a bateria anti-míssel Patriot ficou operando por mais de 100 horas, causando uma falha em um acumulador. Um míssel Scud atingiu um acampamento millitar
  - 2003 – Uma falha no sistema de pagamento de aponsentadoria nos EUA causou o envio de milhares de cheques inválidos

- O que falha mais: um software ou uma ponte?
  - 1940 – Cai ponte em Washington (EUA) http://www.youtube.com/watch?v=P0Fi1VcbpAI
  - 2004 – Ponte entre Alemanha e Suiça – ao conectar as duas partes havia uma diferença de altura devido a diferença de interpretação entre “nível do mar”

Outros dados

- Cutter Consortium (2002):
  - 78% das empresas de TI se envolveram em disputas judiciais por conta de software entregue
  - 67% dos casos o software não entregava o pedido
  - 56% as datas prometidas não foram cumpridas
  - 45% apresentavam falhas graves
- Evolução do hardware x evolução do software
- Crise do software (ou Depressão do Software?)

História

- A indústria de software começou no final dos anos 50
- Computer Usage Corporation (CUC) fundada em 1955 por dois ex-funcionários da IBM – primeira empresa de desenvolvimento de software
  - Em 1967 tinha 700 funcionários em 12 escritórios, com uma receita de mais de US$ 13 milhões
- Estima-se que em 1967 havia 2.800 empresas de software nos EUA
- Um grupo de estudos da OTAN cunhou o termo Engenharia de Software em 1967
- NATO Conference 1968
- Em 1976, 52 produtos ultrapassavam receitas de US$ 5 milhões

Software History Museum
http://www.softwarehistory.org

Desenvolvimento de sw x hardware

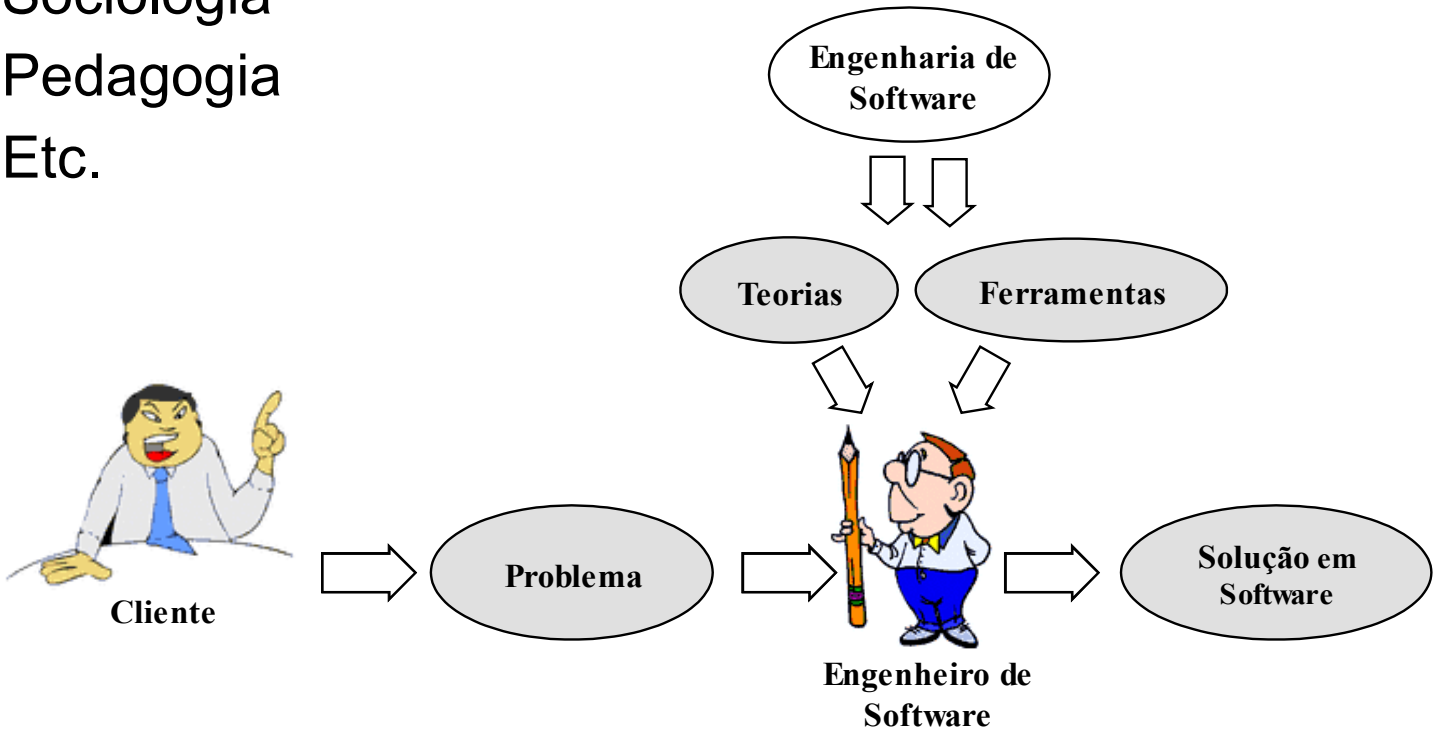
- Custos do software são concentrados na engenharia
- Software não desgasta, mas se deteriora
- Fase de fabricação de um hardware pode produzir problemas de qualidade
- Quando um hardware falha, pode substituir por um sobressalente

1.2 O que é Engenharia de Software?

- A Engenharia de Software é uma disciplina cujo objetivo é produzir software isento de falhas, entregue dentro do prazo e orçamento previstos, e que atenda a necessidade do cliente. [Schach, 2009]
- O estabelecimento 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]
- "**software engineering.** (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1).“ [IEEE, 1991]
- (1). Engenharia que aplica: uma abordagem sistemática, disciplinada e quantificável; os princípios da ciência da computação, design, engenharia, administração, matemática, psicologia, sociologia e outras disciplinas se necessário for; e às vezes pura invenção, para criar, desenvolver, operar e manter de forma econômica, confiável e correta, soluções de alta qualidade para problemas que envolvam software. (2). Engenharia de Software também é o estudo e a busca por abordagens para a realização das atividades (1). [Berry, 1992]

Competências envolvidas na ES

- Matemática e ciência da computação
- Economia
- Administração
- Psicologia
- Sociologia
- Pedagogia
- Etc.



Desenvolvimento de software

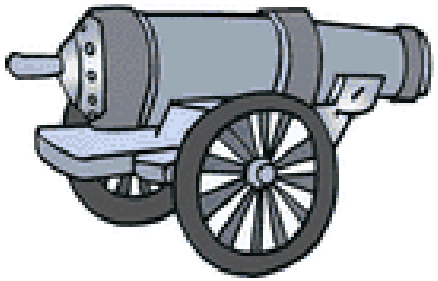
- Ciência da computação x sistemas de informação
- Atividade meio x como atividade fim (químicos x engenheiros químicos)
- O engenheiro de software constrói soluções computacionais para problemas dos usuários – quase sempre não são problemas da área de informática.
- Necessidade de criação e invenção contínua
- A **experiência**, a **criatividade** e a **perspicácia** são fundamentais
- Um programador bom é melhor do que um programador mediano mais equipado?

Passos do desenvolvimento

- Entendimento do problema
- Modelagem
- Estudo de possíveis soluções
- Seleção de acordo com critérios específicos (performance, segurança, eficiência, precisão, integração, escalabilidade, modificabilidade, usabilidade, legibilidade, etc.)
- Implementação
- Implatantação

O que é uma solução boa?

- Atende aos requisitos
- Relação custo x benefício
- Evitar o overkill

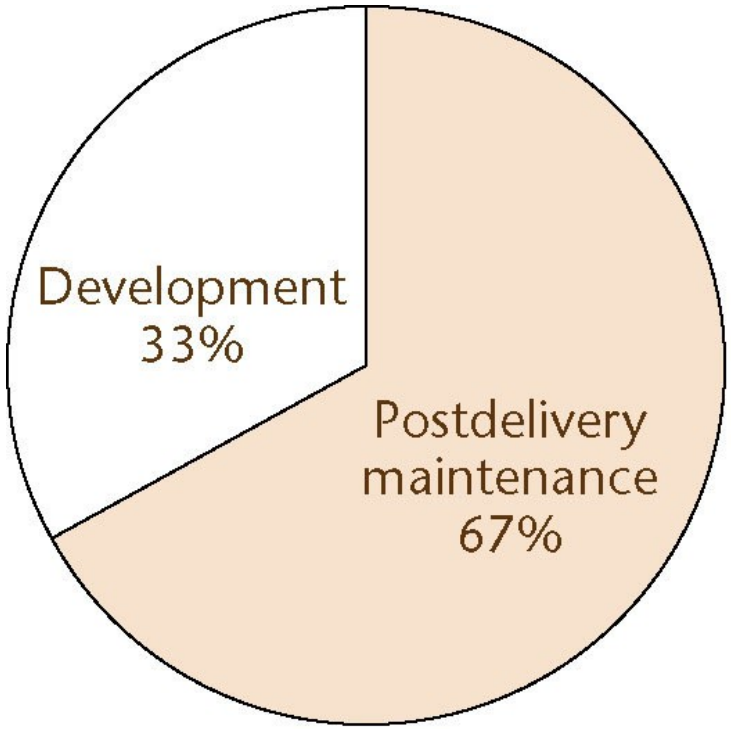


1.3 Ciclo de vida do desenvolvimento

- Modelo clássico (1970)

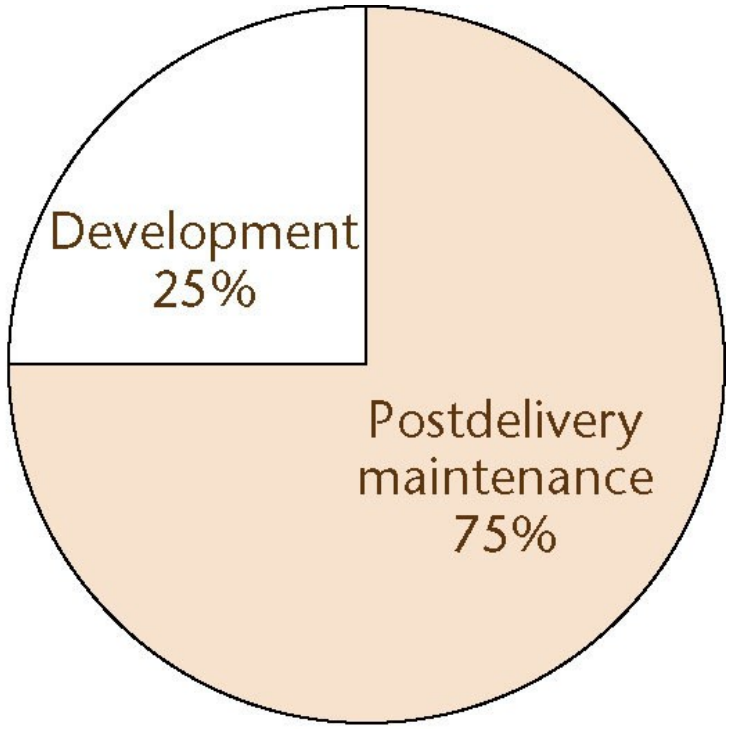
1. Requirements phase
2. Analysis (specification) phase
3. Design phase
4. Implementation phase
5. Postdelivery maintenance
6. Retirement

Desenvolvimento x Manutenção



(a)

1976-1981



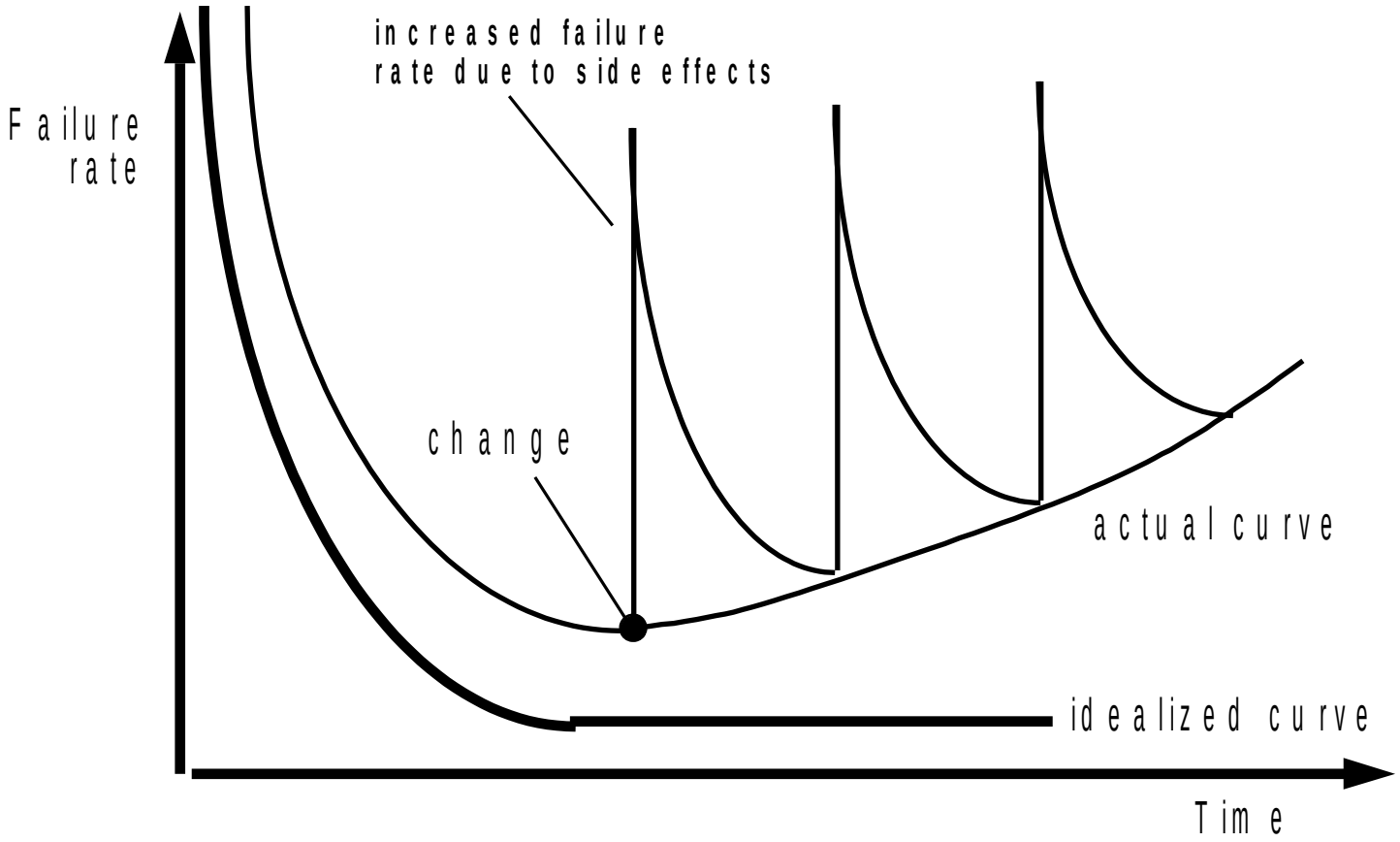
(b)

1992-1998

[Schach, 2009]

Para onde direcionar os esforços?

Deterioração de software



[Pressman, 2006]

Economia da Engenharia de Software

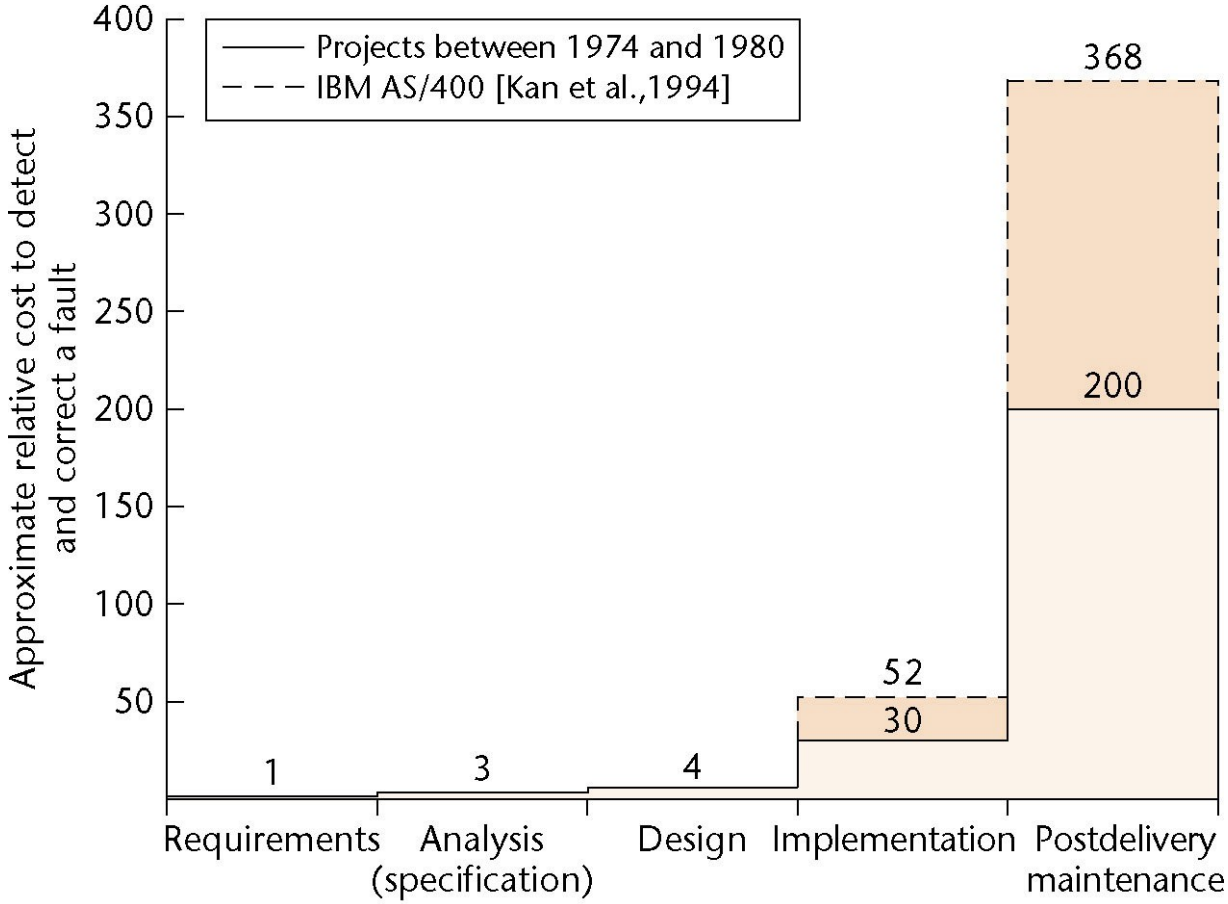
- O método de codificação M2 é 10% mais rápido que M1 atualmente em uso. Deve ser usado?

Economia da Engenharia de Software

- O método de codificação M2 é 10% mais rápido que M1 atualmente em uso. Deve ser usado?
- Depende!
- Tem que considerar:
  - Treinamento
  - Tempo para atingir proficiência (curva de aprendizagem)
  - Impacto do novo método na manutenção
  - Código a ser transformado em legado

Custo de uma falha

Projetos na IBM [Boehm, 1981]



Para onde direcionar esforços?

Documentação

- Não deve ser feita somente após o final do desenvolvimento
  - Rotatividade de membros da equipe
  - Para iniciar uma fase é necessário que a anterior esteja documentada
  - Documentação facilita o teste
  - Manutenção
- Problema: atualização da documentação

Padronizações

- ISO – International Organization for Standardization
  - Além de ser um acrônimo, é derivado da palavra grega que significa “igual”
  - Rede de institutos nacionais de padronização de 147, com sede em Genebra, Suiça

Ética

- IEEE-CS ACM Software Engineering Code of Ethics and Professional Practice www.acm.org/serving/se/code.htm
  - Atuar de acordo com o interesse do **público**
  - Benéfica para o **cliente** e **empregador**
  - Os **produtos** atendam aos padrões profissionais mais elevados
  - Manter integridade e independência nas **avaliações** profissionais
  - Enfoque ético no **gerenciamento** do desenvolvimento
  - Integridade e reputação da **profissão**
  - Ser justo com seus **colegas** de trabalho
  - Postura de **aprendizagem** por toda vida

Mitos ou verdades

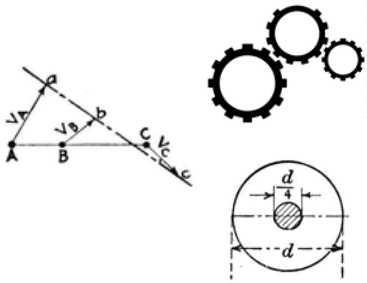
- Software bem feito não sofre manutenção
  - Software ruim é descartado, para software bom há trabalho de manutenção por anos
- Se nos atrasarmos no cronograma, podemos adicionar mais programadores
  - Adicionar pessoas a um projeto de software atrasado, atrasa-o ainda mais. [Brooks, 1975]
- Quando escrevemos um programa e o fazemos funcionar, nosso trabalho está completo

Diferentes tipos de desenvolvimento

- Desenvolvimento interno por demanda
- Desenvolvimento por contrato
- Desenvolvimento de COTS
- Desenvolvimento de linha de produto
- Desenvolvimento de F/OSS (Free and Open Source Software)
  - "Com um bom número de olhos, todos os bugs são superficiais" (Raymond, 2000) => Lance o produto logo e frequentemente.
- Desenvolvimento Web
- Desenvolvimento sistemas críticos
- Desenvolvimento de sistemas de tempo real
- Desenvolvimento de sistema embarcado
- Desenvolvimento de sistemas científicos
- Etc.

ES é mesmo uma Engenharia?

- Atualmente alguns pesquisadores começam a contestar esta apropriação.



ES é mesmo uma Engenharia?

Engenharia tradicional	Engenharia de software
Escopo mais restrito	Incontáveis domínios de aplicação
Soluções restritas por leis físicas	Poucas limitações tecnológicas
Soluções similares	Pluralidade de soluções
Aplicação de técnicas de forma determinística	Criação e invenção contínua
Soluções para problemas específicos	Software modela processos abstratos do mundo real
Uso intenso da matemática	Uso restrito da matemática em algumas etapas

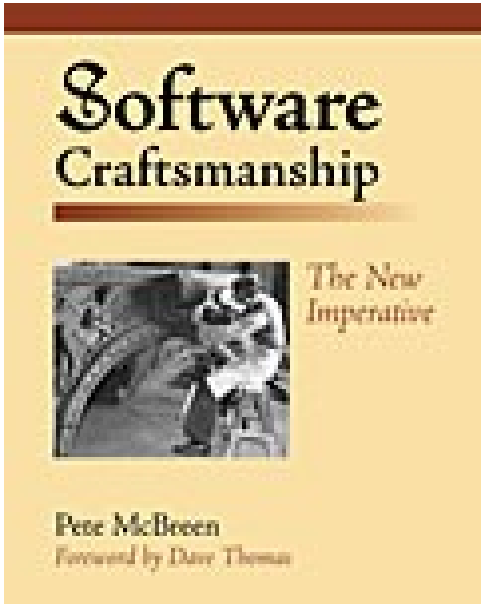
ES é mesmo uma engenharia?

- *"Desenvolvimento de software é um jogo cooperativo de invenção e de comunicação. Nunca foi engenharia, apesar de toda a propaganda neste sentido. Desenvolvimento de software consiste em nada mais do que idéias, concretizadas. Consiste em pessoas inventando e se comunicando, trabalhando em um problema que ainda não entendem, e que não pára de mudar, criando uma solução que ainda não entendem, e que não pára de mudar, expressando suas idéias usando linguagens restritas, que quase não entendem, para um interpretador que não perdoa erros."*

[Alistair Cockburn, 2001]

Como se compara com outras analogias?

- Arquitetura
- Arte
- Carpintaria
- Literatura



Treinamento do desenvolvedor

- Lidar com incerteza e indefinição
- Criatividade e inovação
- Capacidade de avaliar processos, métodos e ferramentas
- Capacidade de adaptar processos, métodos e ferramentas para cenários específicos
- Trabalho em grupo
- Comunicação

- Volta ao passado? [Teles, 2004]

Sociedade Agrícola	Revolução industrial
Trabalho em casa	Local de trabalho
Ofício	Divisão de trabalho
Aprendizagem assíncrono	Aprendizagem síncrono

- Aprendizagem de uma manufatura?

Atividades da disciplina

Calendário

- Não haverá aula
  - 05 e 07/09 Semana da pátria
  - 26 e 28/09 CBSoft?
  - 10 e 12/10 Break
  - 02/11 Finados
  - 14 e 16/11 Break
  - Em outras datas haverão substitutos por conta de viagens para participação em eventos (a ser avisado oportunamente)
- Provas
  - Prova 1: 14, 19 ou 21 de setembro (?)
  - Prova 2: 28/11
- Projeto
  - Entrega final: 30/11
- Outras atividades
  - Datas de entregas a ser divulgadas oportunamente

Desenvolvimento de um projeto real

- Iterações de 3 semanas
  - 22/08 a 11/09
  - 12/09 a 02/10
  - 03/10 a 23/10
  - 24/10 a 08/11 (essa iteração é um pouco mais curta por conta do break)
  - 09/11 a 30/11
- Equipes de 6 desenvolvedores (serão montadas aleatoriamente)
  - Nessa disciplina espera-se Cooperação e não Colaboração
- Mais detalhes serão divulgados no dia 22/08
- Não faltem no primeiro dia de cada iteração – será feita a reunião de retrospectiva, planejamento e início das atividades em sala.

- 1º período de 3 semanas – nivelamento tecnológico

Nivelamento tecnológico

- As tecnologias a serem usadas no projeto serão:
  - HTML, CSS, Java, JSP, VRaptor, JPA+Hibernate, JUnit, Tomcat
- Objetivo do nivelamento
  - Entregar um CRUD usando as tecnologias acima no dia 21/08
- Cronograma do nivelamento
  - 03/08 – Trabalho em sala usando VRaptor (estudar até lá)
  - 07/08 – Entrega do sistema com VRaptor
  - 10/08 – Trabalho em sala usando JPA
  - 14/08 – Entrega do sistema usando VRaptor+JPA
  - 15/08 – Trabalho em sala usando JUnit
  - 21/08 – Entrega do sistema

Tarefas

- Cadastrar-se no Moodle (Paca)
- Estudar framework VRaptor – http://vraptor.caelum.com.br
- Estudar programação de sistemas web com Java – ver apostilas da Caelum http://www.caelum.com.br/apostilas/
  - Java e Orientação a Objetos
  - Desenvolvimento ágil para Web com VRaptor, Hibernate e Ajax
  - Java para desenvolvimento Web (partes)
- Acessar o site do CBSoft 2011 e considerar seriamente a possibilidade de participar do evento
  - Quem apresentar um relatório sucinto da participação no evento receberá uma bonificação na nota

CBSoft 2011

- Simpósio Brasileiro de Engenharia de Software (SBES)
- Simpósio Brasileiro de Componentes, Arquiteturas e Reúso de software (SBCARS)
- Simpósio Brasileiro de Métodos Formais (SBMF)
- Simpósio Brasileiro de Linguagens de Programação (SBLP)
- MiniPlop – Conferência sobre padrões de projeto
- Workshops
- Trilha da indústria
- Palestras internacionais
- Etc.
- Inscrições com desconto até dia 12/08