Engenharia de Software

Introdução à Engenharia de Software

Profª Raquel Mini

raquelmini@ufmg.br

DEE / UFMG

Unidade I

- Introdução à Engenharia de Software
  - Motivação
  - Definição
  - Histórico
  - Conceitos fundamentais
  - Mitos relativos ao software
  - Responsabilidade profissional e ética do Engenheiro de Software

Motivação

- Em várias organizações mundo afora...
  - "... aquele programa não roda, só tem *bug*!"
  - "... o programa roda, mas não faz o que precisamos..."
  - "... o projeto que teima em nunca terminar!"
  - "... o cliente está bravo com isso, e disse que, desse jeito, não faz mais projeto com a gente ..."
  - "... testes? mas não deu tempo de fazê-los..."

Motivação

- Em várias organizações mundo afora...
  - "... lá vou eu de novo perder o meu fim de semana aqui, tentando arrumar isso... Nem sei se vai adiantar..."
  - "... com o que está escrito aqui, nem dá para entender o que é para fazer..."
  - "o que você quer que eu faça? O pessoal não me entrega os dados para que eu possa desenvolver..."

Motivação

- Será que temos que aceitar esta situação?
- Não existe um outro caminho, a não ser “chutar o balde"?

A Engenharia de Software objetiva minimizar e, por vezes, eliminar os problemas citados

O que é software?

- Programa de computador + documentação + dados de configurações necessários para fazer o programa operar corretamente

- Classificação fundamental
  - Produtos genéricos (ex. MS Office)
  - Produtos encomendados
    (ex. Software de Controle da Locadora do Zé)

Software

Software de sistemas

Software de aplicação

Software para linha de produtos

Software de inteligência artificial

Software científico e de engenharia

Aplicações da web

Software embutido (embarcado)

Software está em todo lugar

Perguntas frequentes sobre software

- O que é software?
  - Softwares são programas de computador e documentação associada. Produtos de software podem ser desenvolvidos para um cliente específico ou para o mercado em geral.
- Quais são os atributos de um bom software?
  - Um bom software deve prover a funcionalidade e o desempenho requeridos pelo usuário. Além disso, deve ser confiável e fácil de manter e usar.
- O que é engenharia de software?
  - É uma disciplina de engenharia que se preocupa com todos os aspectos de produção de software.

Perguntas frequentes sobre software

- Quais são as principais atividades da engenharia de software?
  - Especificação de software, desenvolvimento de software, validação de software e evolução de software.
- Qual a diferença entre engenharia de software e ciência da computação?
  - Ciência da computação foca a teoria e os fundamentos. Engenharia de software preocupa-se com o lado prático do desenvolvimento e entrega de softwares úteis.

Perguntas frequentes sobre software

- Qual a diferença entre engenharia de software e engenharia de sistemas?
  - Engenharia de sistemas se preocupa com todos os aspectos do desenvolvimento de sistemas computacionais, incluindo engenharia de hardware, software e processo. Engenharia de software é uma parte específica desse processo mais genérico.
- Quais são os principais desafios da engenharia de software?
  - Lidar com o aumento de diversidade, demandas pela diminuição do tempo para entrega e desenvolvimento de software confiável.

Perguntas frequentes sobre software

- Quais são os custos da engenharia de software?
  - Aproximadamente 60% dos custos de software são de desenvolvimento, 40% são custos de testes. Para software customizado, os custos de evolução frequentemente superam os custos de desenvolvimento.

A dependência mundial ao software

- A economia de todos os países desenvolvidos depende de complexos sistemas contendo software
- Produtos atuais incorporam sistemas controlados por computadores, com software de controle
  - Gastos com desenvolvimento de software representam uma fração significativa do PIB de muitos países
- Produzir software com boa relação custo-benefício é essencial na economia nacional e internacional

Erros de software mais famosos

1. **Problemas no Mariner (1962)**
   - **Custo:** 18,5 milhões dólares
   - **Desastre:** Mariner, um foguete com uma sonda espacial para Vênus, foi desviado de seu percurso de voo logo após o lançamento. O controle da missão destruiu o foguete 293 segundos após a decolagem.
   - **Causa:** Um programador, ao passar para o computador uma fórmula que haviam lhe entregado escrita manualmente, se esqueceu de uma barra. Sem ela, o software tratava variações normais de velocidade como se fossem sérios problemas, causando falhas por tentativas de correções que acabaram por enviar o foguete fora do curso.

Erros de software mais famosos

2. Hartford Coliseu Desmorona (1978)

- **Custo:** 70 milhões de dólares, além de outros danos de 20 milhões para a economia local
- **Desastre:** Poucas horas depois de 4746 fãs deixarem o Coliseu Hartford, o teto de treliça de aço desabou sob o peso da neve molhada.
- **Causa:** O programador do software CAD, utilizado para projetar o coliseu, incorretamente assumiu que o suporte do telhado de aço enfrentaria apenas compressão natural. Mas quando um dos suportes inesperadamente recebeu um bloco de neve, este desencadeou uma reação em cadeia que derrubou o telhado de outras seções como dominós.

Erros de software mais famosos

2. Hartford Coliseu Desmorona (1978)

Aerial view of the Hartford Civic Center roof, January 18, 1978 – Connecticut Historical Society

Erros de software mais famosos

3. Máquina medicinal mata (1985)

- **Custo:** Três mortos e três seriamente feridos
- **Desastre:** A máquina de radiação canadense Therac-25 irradiou doses letais em pacientes.
- **Causa:** Por causa de um *bug* sutil chamado de "*condição de corrida*", um técnico acidentalmente configurou o Therac-25 de modo que o feixe de elétrons seria como um fogo de alta potência.

Erros de software mais famosos

4. Crash na Wall Street (1987)

- **Custo:** U$500 bilhões em um dia
- **Desastre:** Em 19 de outubro de 1987, o índice Dow Jones caiu 508 pontos, perdendo 22,6% de seu valor total. Esta foi a maior perda que Wall Street já sofreu em um único dia.
- **Causa:** Um mercado em grande alta foi interrompido por uma série de investigações conduzidas pela SEC e por outras forças do mercado. Como os investidores fugiram de ações investigadas, um número muito grande de ordens de venda foram gerados pelos computadores, quebrando sistemas e deixando os investidores efetivamente cegos.

Erros de software mais famosos

5. Ariane Rocket Goes Boom (1996)

- **Custo:** $500 milhões
- **Desastre:** Ariane 5, o mais novo foguete da Europa não-tripulado, foi intencionalmente destruído segundos após seu lançamento em seu voo inaugural. Também foram destruídos quatro satélites científicos para estudar como o campo magnético da Terra interage com os ventos solares.
- **Causa:** O desligamento ocorreu quando o computador de orientação tentou converter a velocidade do foguete de 64 bits para um formato de 16 bits. O número era muito grande, o que resultou em erro de estouro. Quando o sistema de orientação desligou, o controle passou para uma unidade idêntica redundante, que também falhou porque nele estava sendo executado o mesmo algoritmo.

Desafios de produzir software

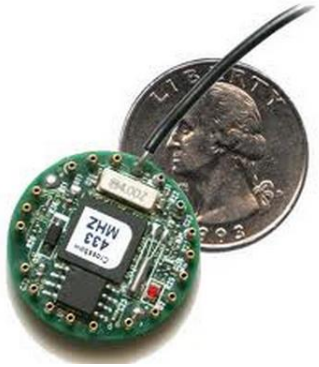
- Exemplo: Voo Air France Rio - Paris
  - 1. Dados conflitantes (falha nos sensores)
  - 2. Sistema assume o controle (piloto automático)
  - 3. Piloto tenta reiniciar o sistema (boot)
  - 4. Em 4 minutos o avião mergulha no oceano

Confiabilidade

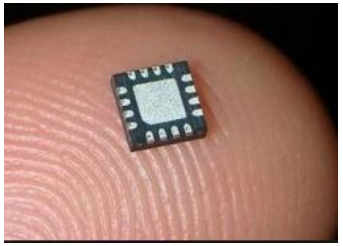
The Last Four Minutes of Air France Flight 447. *Fevereiro, 2010.*
http://www.spiegel.de/international/world/0,1518,679980,00.html

Desafios de produzir software

- Pouco espaço na memória
- Grande variação em características de aparelhos



Preço e desempenho



Desafios de produzir software

- Equipamentos médicos
  - Extremamente críticos
  - Lidam com vidas

- Caixas eletrônicos
  - Prejuízos financeiros

Sistemas Críticos

Em resumo...

- O desenvolvimento informal de software geralmente não é suficiente
  - Técnicas e métodos são necessários

- Algumas dificuldades
  - Heterogeneidade
  - Confiabilidade
  - Prazo de entrega
  - Mudança contínua

Software

- Crescente interesse nos recursos e funções dos softwares

Software

- Crescente interesse nos recursos e funções dos softwares

- Softwares estão ficando mais complexos

Projetar tornou-se fundamental

Software

- Crescente interesse nos recursos e funções dos softwares
- Softwares estão ficando mais complexos
- Softwares são utilizados na tomada de decisões importantes

Software deve apresentar elevada qualidade

Software

- Crescente interesse nos recursos e funções dos softwares
- Softwares estão ficando mais complexos
- Softwares são utilizados na tomada de decisões importantes
- Grande demanda por adaptação e aperfeiçoamento

Software deve ser passível de manutenção

Software

- Software, em todas as suas formas e em todos os seus campos de aplicação, deve passar pelos processos de engenharia

Engenharia de Software

Engenharia de Software

- Consiste no estabelecimento e o emprego de sólidos princípios de engenharia de modo a obter software economicamente viável que seja confiável e que funcione eficientemente em máquinas reais

Definição proposta por Fritz Bauer na conferência sobre o tema em 1968

- Aplicação de uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, na operação e na manutenção de software

Definição feita pela IEEE em 1993

Histórico

- Década de 50
  - Início do desenvolvimento de software

- Década de 60
  - Introdução de poderoso hardware de 3ª Geração viabilizou aplicações até então inimagináveis
  - Custos de hardware caíam e de software subiam
  - Aplicações maiores e mais complexas necessitavam uma abordagem mais formal, visando:
    - Evitar atrasos constantes (às vezes de anos)
    - Diminuir desvios (enormes) de custos
    - Melhorar a qualidade (produtos não confiáveis, de difícil manutenção e desempenho insatisfatório)

Histórico

- Década de 60
  - Fundamental: novas técnicas e métodos para controlar necessidades inerentes a grandes projetos de software
  - 1968: Conferência para discutir a "**Crise do Software**"
    - Surge o termo "**Engenharia de Software**"
      - Uso da engenharia para melhorar o desenvolvimento de sistemas de software com boa relação custo-benefício
      - Desenvolvimento eficaz de software de qualidade

Histórico

- Década de 60
  - Grande progresso desde então
    - Melhor compreensão dos processos envolvidos
    - Métodos eficazes de especificação, projeto e implementação
    - Novas notações e ferramentas de apoio
- Década de 70
  - Projetos mal-gerenciados
  - Orçamentos estourados
  - Prazos estourados
  - Produtos com baixa qualidade

Causa: Complexidade

Histórico

- Combate à crise do software
  - Inicia-se “o estabelecimento e uso de princípios abrangentes de engenharia de forma a obter software economicamente que é confiável e opera eficientemente em máquinas reais”

    - Fritz Bauer, 1ª Conferência de ES da OTAN
- Cientistas da computação:
  - Grupo que mais conhecia sobre desenvolvimento de Software
  - Diretamente envolvidos na criação Engenharia de Software

Engenharia de Software é engenharia?

- Aspectos conceituais
  1) Cabe na definição de Engenharia
  2) Possui modelos, padrões, notações e medidas
  3) Atende a problemas complexos e de escala

Engenharia de Software é engenharia?

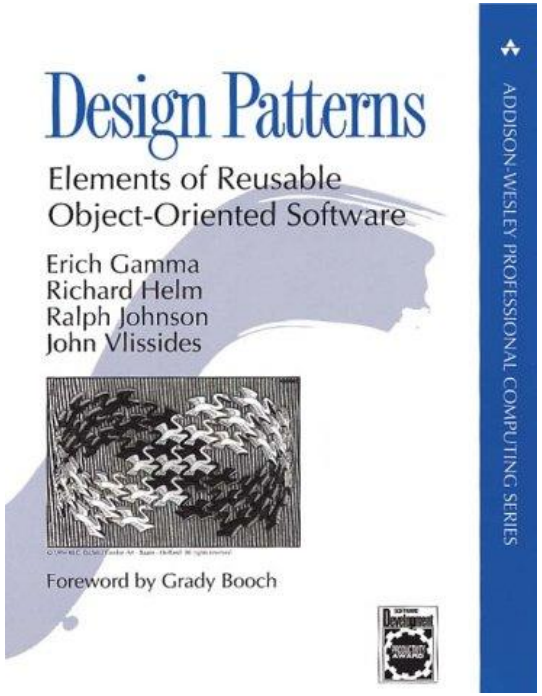
1) Cabe na definição de Engenharia:

- A Engenharia é a profissão em que o conhecimento das ciências matemáticas e naturais, obtido através do estudo, experiência e prática, é aplicado para se obter maneiras de se utilizar, de forma econômica, os materiais e recursos naturais e forças da natureza para o benefício da humanidade

Engenharia de Software é engenharia?

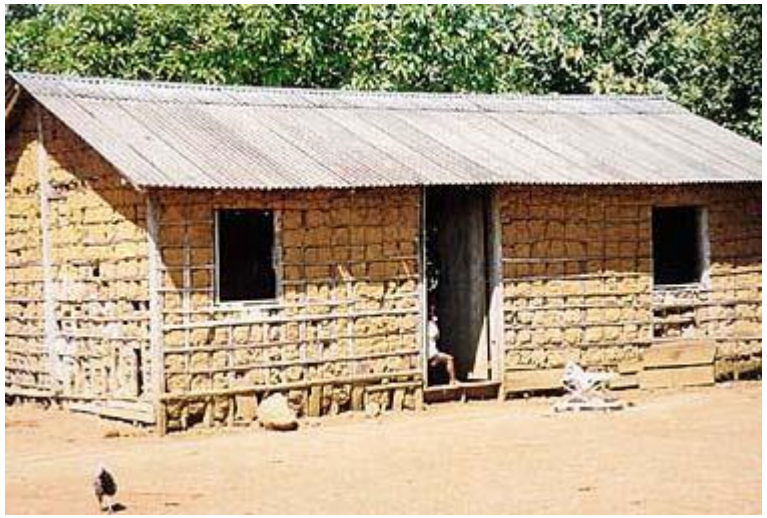
2) Possui modelos, padrões, notações e medidas:

- Ponte suspensa x padrões de projeto de software



Engenharia de Software é engenharia?

3) Atende a problemas complexos e “de escala”, assim como outras engenharias:
   - Levantar quatro paredes x levantar um edifício
   - Construir 1.000 x 1.000.000 de linhas de código



Características específicas da ES

1. Disciplina de descrições e abstrações
2. Software não se desgasta
3. Reposição
4. Especialização
5. Estratificação
6. Trata do desenvolvimento de software como produto

Características específicas da ES

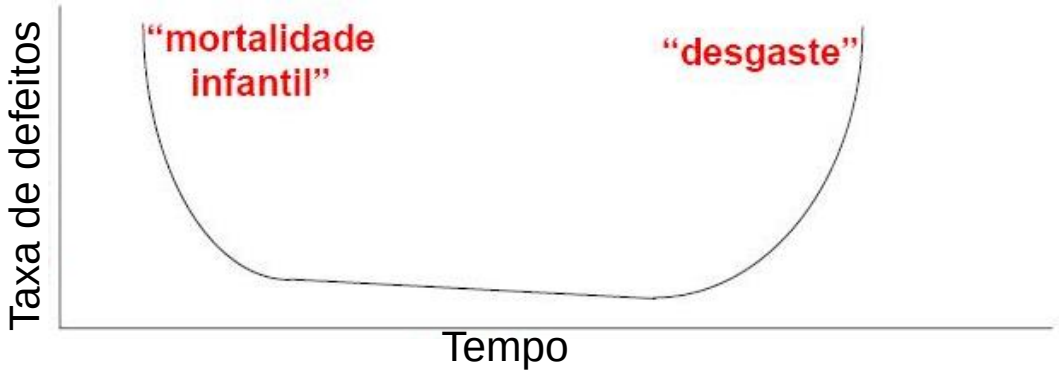
1. Disciplina de descrições e abstrações
   - Todas as engenharias usam modelos:
     - Representam a realidade de forma simples e barata
     - Cria-se o modelo e depois se constrói o produto do modelo
   - Na Engenharia de Software
     - Os modelos são as abstrações constantemente refinadas
       - requisitos → análise → desenho → código
       - No final, a abstração está tão detalhada que é o software
     - Custo do modelo é praticamente o custo total
       - Automóvel: gerar o protótipo tem um custo, assim como construir um automóvel
       - Software: o modelo é praticamente todo o custo, uma vez que a cópia é barata

Características específicas da ES

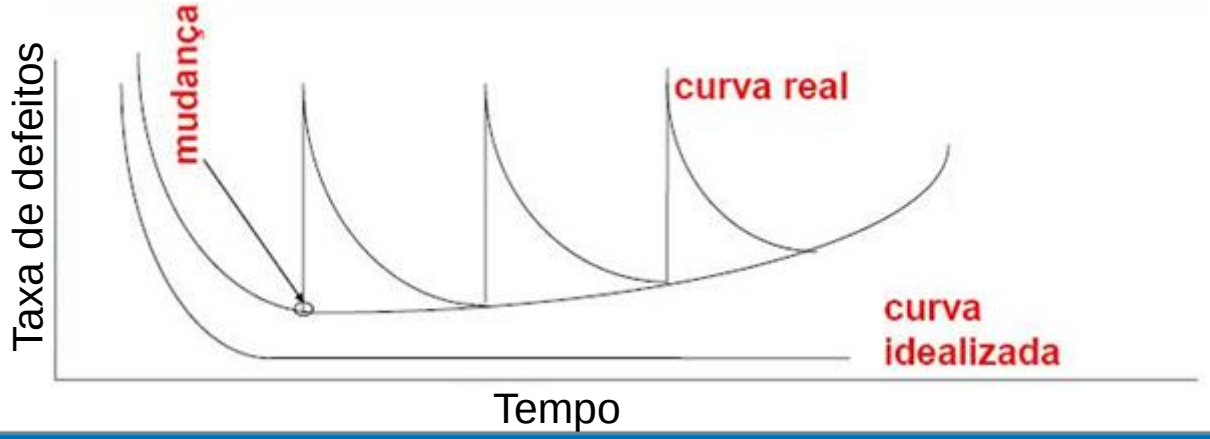
2. Software não se desgasta
   - Não sofre efeitos físicos
   - Entretanto, software de deteriora:
     - Mudanças no ambiente de execução
     - Novos requisitos, do melhor entendimento, de novas situações.

Características específicas da ES

Curva de defeitos para hardware



Curva de defeitos para software



Características específicas da ES

3. Reposição
   - Quando um componente de hardware se desgasta, ele é substituído por uma peça de reposição
   - Não existem peças de reposição para software
     - Cada defeito de software indica um erro no projeto ou no processo pelo qual o projeto foi traduzido em código de máquina executável
     - Manutenção de software implicam em complexidade consideravelmente maior que a manutenção de hardware

Características específicas da ES

4. Especialização
   - Cada engenharia lida com um domínio específico:
     - Civil: mecânica, estruturas, sustentação, fundação
     - Elétrica: potência, distribuição, conservação
     - Química: reações, refinamentos, processos
   - Dois domínios na ES
     - Do software: algoritmos, estruturas de dados
     - Do problema: Física, química, biologia, economia, diversão, entretenimento adulto, medicina, editoração, jornalismo

Características específicas da ES

5. Estratificação
   - Desenvolvedores de software perfazem todos os papéis no desenvolvimento de software
     - Requisitos, análise, desenho, implementação, testes
   - Outras engenharias possuem mais estratificação
     - Arquiteto, o engenheiro, o mestre de obras, o pedreiro

Características específicas da ES

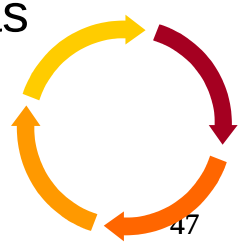
6. Trata do desenvolvimento de software como produto
   - A Engenharia de Software se preocupa no desenvolvimento de software como produto
   - Softwares desenvolvidos para satisfação própria ou para uso único não são alvo da Engenharia de Software

Exercícios

1. Qual é a diferença mais importante entre o desenvolvimento de um produto genérico de software e desenvolvimento de software sob demanda? O que isso pode significar na prática para usuários de produtos de software genérico?

Conceitos Fundamentais

- **Projeto**: unidade gerencial que cobre a execução de um processo de desenvolvimento de software
- **Processo**: maneiras pelas quais se realiza uma operação, segundo determinadas normas.
- **Processo de desenvolvimento de software**: conjunto de atividades cujo objetivo é o desenvolvimento ou a evolução de produtos de software tais como:
  - Especificação: o que o sistema deve fazer e suas restrições
  - Desenvolvimento: produção do sistema de software
  - Validação: verifica se o software é o que o cliente deseja
  - Evolução: mudanças no software em resposta a demandas

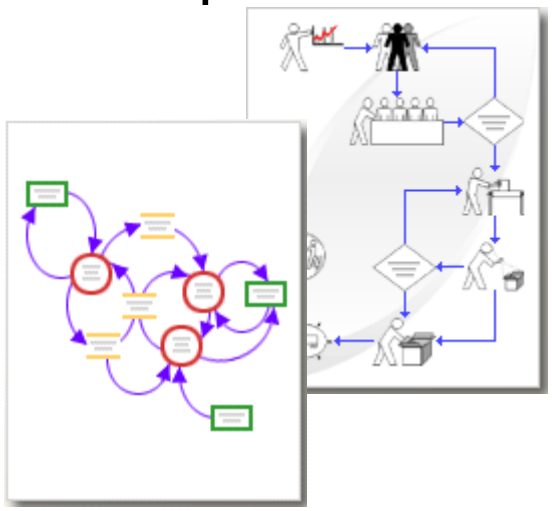


Conceitos Fundamentais

- Não existe um processo de software “ideal” e “padrão”
  - Diferentes processos organizam estas atividades de diferentes maneiras e são descritos em diferentes níveis de detalhe
    - Passos, prazos, custos, recursos e produtos variam muito
  - Um mesmo tipo de produto pode ser obtido por processos distintos
    - Alguns processos são mais adequados do que outros

Conceitos Fundamentais

- **Modelo de processo de software**: uma representação simplificada de um processo de software, apresentada de uma perspectiva específica
  - Exemplos de perspectiva de processo são:
    - Perspectiva de fluxo de trabalho (*workflow*)
      - Sequência de atividades
    - Perspectiva de fluxo de dados (*data-flow*)
      - Fluxo de informação
    - Perspectiva de papel/ação (*role* / *action*)
      - "Quem" faz "o que"



- Modelos gerais (paradigmas de desenvolvimento)
  - Modelo em cascata
  - Desenvolvimento evolucionário
  - Montagem a partir de componentes

Conceitos Fundamentais

- **Artefato**: coisa concreta produzida em um projeto
  - Código-fonte, executável, documentos,...
- **Produto**: conjunto de artefatos entregues aos usuários finais e ao cliente em um projeto
  - O conjunto não está limitado somente ao software ou versão executável, envolve quaisquer artefatos intermediários necessários
- **Cliente**: pessoa física ou jurídica que contrata a execução de um projeto ou seu representante autorizado, com poder de aceitação de propostas e produtos
- ***Stakeholders***: qualquer pessoa ou organização que tenha interesse, ou seja afetado pelo projeto

Conceitos Fundamentais

- **Usuário**: pessoa que efetivamente usa um produto de software
  - Pode ser o próprio cliente, funcionário de uma organização cliente ou mesmo não ser relacionado diretamente com o cliente
  - No caso de produtos de prateleira, o cliente é o setor da organização responsável pela definição dos requisitos do produto (o setor de *marketing*, por exemplo)
- **Marco**: ponto de tempo que representa um estado significativo de um projeto, geralmente associado à conclusão e aprovação de um ou mais resultados
- **Ciclo de vida**: conjunto da história de um software, desde seu início até sua retirada de operação

Conceitos Fundamentais

- **Refatoração**: melhoria da estrutura e organização de um programa

- **Método**: uma abordagem estruturada para o desenvolvimento
  - Conjunto razoavelmente completo de regras e critérios que estabelecem uma maneira precisa e repetível de executar uma tarefa ou prática para chegar a um resultado
  - Objetivo: facilitar a produção de software de alta qualidade, apresentando boa relação custo-benefício

Conceitos Fundamentais

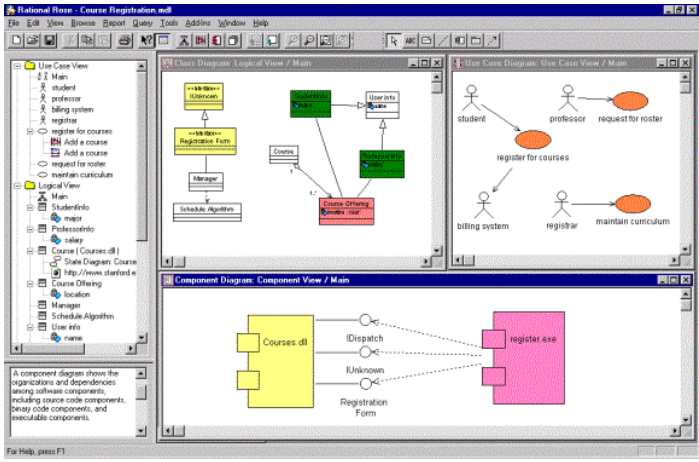
- **Método:**
  - Anos 70: surgem os métodos orientados a funções
    - Análise Estruturada (DeMarco - 1978), JSD (Jackson - 1983)
  - Anos 80 e 90: surgem os métodos orientados a objetos
    - Booch (raízes da orientação a objetos - 1990)
    - Rumbaugh (*Object Modeling Technique* -1991)
    - Jacobson (*Object-Oriented Software Engineering* -1992)
    - Posteriormente integradas em uma única abordagem
      - UML (*Unified Modeling Language*): versão 0.9 em 1996

Conceitos Fundamentais

- **Ferramentas:** sistemas de software projetados para fornecer suporte automatizado às atividades rotineiras do processo de software
  - Quando as ferramentas que apoiam os métodos se integram, forma-se a Engenharia de Software Auxiliada por Computador (CASE)
  - Ex: ambientes de desenvolvimento integrado (IDEs)
    - Contém editor, compilador, ferramenta de modelagem com gerador automático de código, depurador, testes automatizados, *refactoring* de código, ...

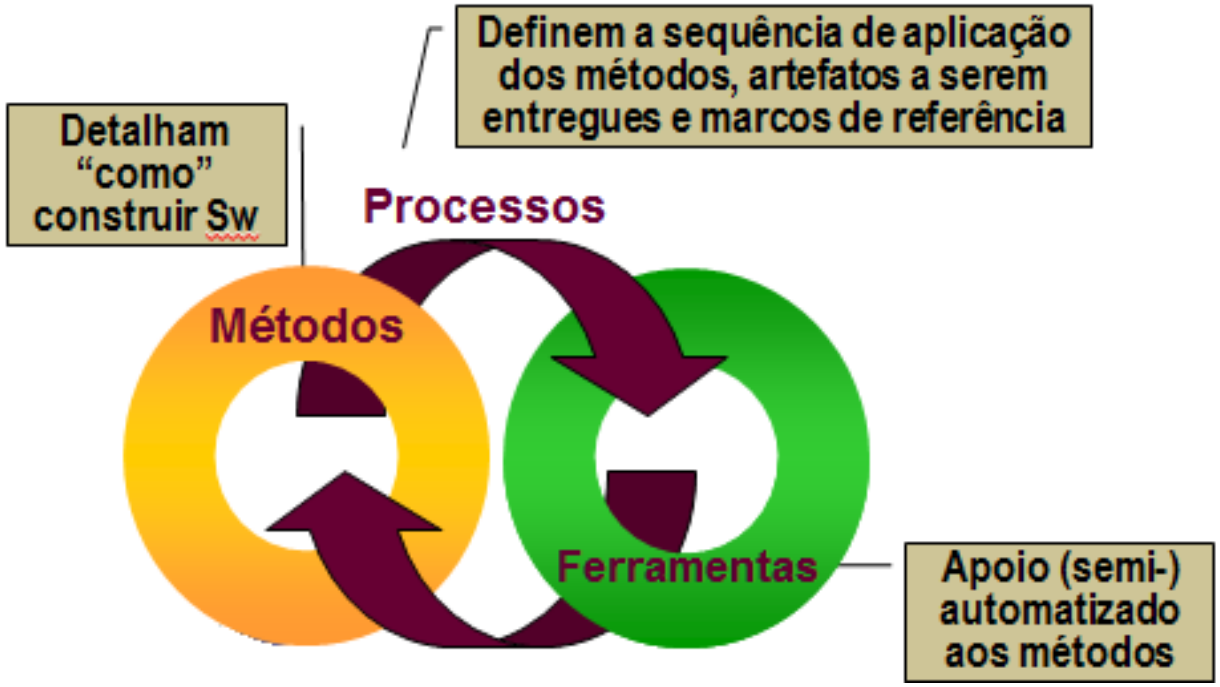
Conceitos Fundamentais

- **Ferramentas**:
  - Vantagens
    - Menos programação
    - Maior qualidade no produto final
    - Agilidade no retrabalho do software
    - Redução de custos na manutenção
    - Maior produtividade
  - Desvantagens
    - Incompatibilidade de ferramentas
    - Custo (alto) por licença
    - Treinamento para utilização



Conceitos Fundamentais

- **Método, processos e ferramentas:**



Conceitos Fundamentais

- **Software de qualidade**: é aquele que contém os requisitos solicitados pelo cliente e satisfaz outros atributos não associados ao que ele faz, como:
  - Comportamento quando em funcionamento (ex: tempo de resposta)
  - Estrutura e organização do programa fonte (ex: legibilidade)
  - Documentação associada (ex: manuais do usuário, de operação/instalação)

Conceitos Fundamentais

- **Software de qualidade:**
  - Exemplos de atributos de qualidade
    - Manutenibilidade: deve evoluir para atender a mudanças de requisitos
    - Nível de confiança: não deve causar prejuízo físico ou econômico se falhar
    - Eficiência: não deve desperdiçar os recursos do sistema
    - Usabilidade: deve ser fácil de usar pelos usuários para os quais foi projetado
  - Os atributos de qualidade variam em função da aplicação
    - Um sistema de controle de telefonia tem que ser confiável
    - Um jogo interativo tem que possuir resposta rápida
    - Um sistema bancário tem que ser seguro

Principais desafios para a ES

- Lidar com o código legado
  - Manter e atualizar código legado (com importantes funções corporativas) mantendo a prestação de serviços corporativos essenciais
- Lidar com a heterogeneidade
  - Desenvolver técnicas para construir softwares confiáveis e flexíveis o bastante para operar em ambientes distribuídos em redes, com diversos tipos de equipamento e sistemas de apoio
- Suprir demandas do mercado atual (dinâmico)
  - Reduzir o tempo para o fornecimento de sistemas grandes e complexos, sem comprometer a qualidade

Mitos relativos ao software

- Outras engenharias são mais maduras, pois geram produtos com maior qualidade e menos defeitos:
  - Exercício: olhar na estrutura em volta e procure qualquer tipo de defeitos: imperfeições, desnivelamentos, paredes tortas, ...

- Outras engenharias são mais previsíveis, seus custos e prazos podem ser estimados com precisão:
  - Exercício: contrate um engenheiro para uma reforma em sua casa e compare o prazo e o orçamento previstos com os realizados

Mitos relativos ao software – gerenciamento

Mito: Já temos um livro que está cheio de padrões e procedimentos para o desenvolver software. Ele não supre meu pessoal com tudo que eles precisam saber?

Realidade: Será que o livro é usado? Os profissionais sabem que ele existe? Ele reflete a prática moderna de desenvolvimento de software? Ele é completo? É adaptável? Está alinhado para melhorar o tempo de entrega, mantendo ainda o foco na qualidade?

Mitos relativos ao software – gerenciamento

Mito: Se o cronograma atrasar, poderemos acrescentar mais programadores e ficarmos em dia.

Realidade: O desenvolvimento de software não é um processo mecânico como o de uma fábrica. Acrescentar pessoas num projeto de software atrasado só o tornará mais atrasado ainda. Quando novas pessoas entram, as que já estavam terão de gastar tempo situando os recém-chegados, reduzindo o tempo destinado ao desenvolvimento produtivo. Novas pessoas somente devem ser adicionadas de forma planejada e bem coordenada.

Mitos relativos ao software – gerenciamento

Mito: Se eu decidir terceirizar o projeto de software, posso simplesmente relaxar e deixar essa empresa realizá-lo.

Realidade: Se uma organização não souber gerenciar e controlar projetos de software, ela irá, invariavelmente, enfrentar dificuldades ao terceirizá-los.

Mitos relativos ao software – clientes

Mito: Uma definição geral dos objetivos é suficiente para começar a escrever os programas, podemos preencher detalhes posteriormente.

Realidade: Embora nem sempre seja possível uma definição ampla e estável dos requisitos, uma definição de objetivos ambígua é receita para um desastre. Requisitos não ambíguos são obtidos somente pela comunicação contínua e eficaz entre cliente e desenvolvedor.

Mitos relativos ao software – clientes

Mito: Os requisitos de software mudam continuamente, mas as mudanças podem ser facilmente assimiladas, pois o software é flexível.

Realidade: O impacto das mudanças de requisitos varia dependendo do momento em que ela foi introduzida. Quando as mudanças são solicitadas cedo, o impacto sobre os custos é relativamente pequeno. Entretanto, conforme o tempo passa, ele aumenta rapidamente.

Mitos relativos ao software – profissionais da área

Mito: Uma vez feito um programa e o colocado em uso, nosso trabalho está terminado.

Realidade: Levantamentos indicam que entre 60 e 80% de todo o esforço será despendido após a entrega do software ao cliente pela primeira vez.

Mitos relativos ao software – profissionais da área

Mito: Até que o programa entre em funcionamento, não há maneira de avaliar sua qualidade.

Realidade: Um dos mecanismos de garantia de qualidade de software mais eficaz é a revisão técnica. Essa revisão pode ser aplicada desde a concepção de um projeto. Elas são um filtro de qualidade que mostram ser mais eficientes do que testes para encontrar certas classes de defeitos de software.

Mitos relativos ao software – profissionais da área

Mito: O único produto passível de entrega é o programa em funcionamento.

Realidade: Um programa funcionando é somente uma parte de uma configuração de software que inclui muitos elementos. Uma variedade de produtos derivados (modelos, documentos, planos) constitui uma base para uma engenharia bem-sucedida.

Mitos relativos ao software – profissionais da área

Mito: A engenharia de software nos fará criar documentação volumosa e desnecessária e, invariavelmente, irá nos retardar.

Realidade: A engenharia de software não trata de criação de documentos, trata da criação de um produto de qualidade. Melhor qualidade conduz à redução do retrabalho, e menos retrabalho resulta em maior rapidez na entrega.

Engenheiros de software e sociedade

- Papel dos engenheiros de software
  - Contribuem direta ou indiretamente com:
    - A análise, a especificação, o projeto, o desenvolvimento, a certificação, a manutenção e os testes de sistemas
  - Habilidades requeridas:
    - Capacidade de análise, negociação e gerência de projetos
    - Entendimento do domínio de aplicação e da atividade do usuário
    - Conhecimento técnico
    - Comunicação
  - Possuem oportunidades significativas para:
    - Praticar o bem ou causar o mal
    - Capacitar / influenciar outras pessoas para fazer o bem ou causar o mal

Engenheiros de Software e Sociedade

- Os engenheiros de software não devem se preocupar apenas com questões técnicas
  - Seu trabalho é realizado dentro de uma estrutura legal e social
  - A engenharia de software é delimitada por leis locais, nacionais e internacionais
  - Necessário se comportar de forma responsável
    - Tanto ética quanto moralmente
    - Para com a profissão e para com a sociedade
      - Não utilizar as capacitações e habilidades para se comportar de maneira desonesta ou que possa trazer descrédito à profissão
    - Manter padrões normais de honestidade e integridade

Responsabilidade Profissional

- Confidencialidade
  - Respeitar a confidencialidade de seus empregadores ou clientes, independente de ter formalizado um acordo
- Competência
  - Não aceitar conscientemente serviços que estejam fora de seu limite de competência
- Direitos de propriedade intelectual
  - Estar ciente das leis locais que regulam o uso da propriedade intelectual (patentes e direitos autorais)
  - Assegurar que a propriedade intelectual de empregadores e cliente sejam protegidas
- Má utilização dos computadores
  - Não empregar suas habilidades técnicas para o mau uso dos computadores de outras pessoas
    - Ex: disseminar vírus, uso indevido do computador da empresa

Conclusão

- Os engenheiros de software são também utilizadores das linguagens de modelagem, dos métodos e das ferramentas

- Assegurar que os engenheiros de software têm acesso a um conjunto eficiente de métodos, ferramentas e técnicas é o primeiro passo para a qualidade e facilidade de utilização de software

Exercícios

2. Além dos desafios de heterogeneidade, mudanças sociais e corporativas, confiança e proteção, identifique outros problemas e desafios que a engenharia de software provavelmente enfrentará no século XXI.

3. À medida que o software invade todos os setores, riscos ao público (devido a programas com imperfeições) passam a ser uma preocupação cada vez maior. Crie um cenário o mais catastrófico possível, porém realista, cuja falha de um programa de computador poderia causar um grande dano (em termos econômico ou humano).