Introdução à Engenharia de Software

Slides do Professor Humberto Rezente (UFU) e Tiago Silva (Unifesp)

Contém *slides* de Ian Sommerville, Sergio Soares (Cin/UFPE) e Luzia Nomura (UFABC)

Como o cliente explicou

Como o lider de projeto entendeu

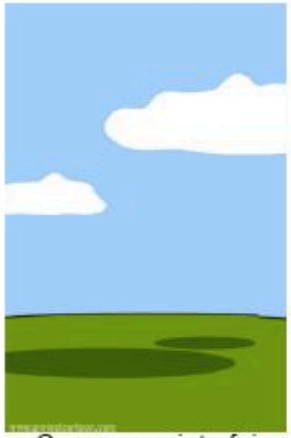
Como o analista planejou

Como o programador codificou

O que os beta testers receberam

Como o consultor de negocios descreveu

Valor que o cliente pagou



Como o projeto foi documentado

O que a assistencia tecnica instalou

Como foi suportado

Quando foi entregue

O que o cliente realmente necessitava

INTRODUÇÃO

Software

- Anos 1970: poucos saberiam definir software; hoje: muitos pensam que entendem
- Será?

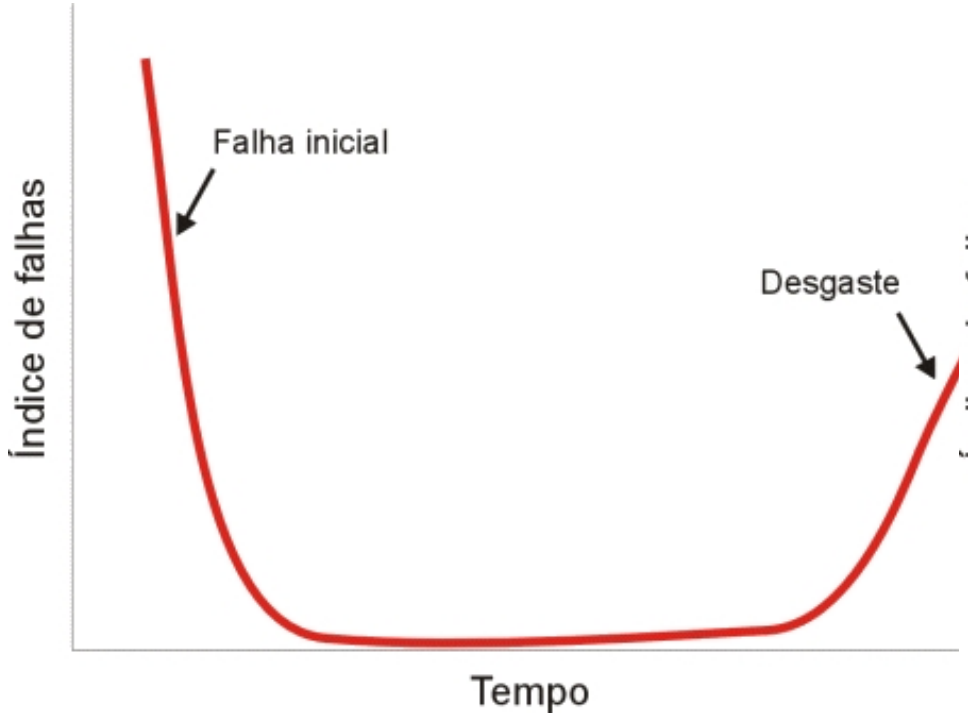
(1) Instruções (programas de computador) que quando executadas fornecem funcionalidades desejadas com desempenho esperado; (2) estruturas de dados que permitem que programas manipulem informações adequadamente; (3) documentação que descreve a operação e uso dos programas.

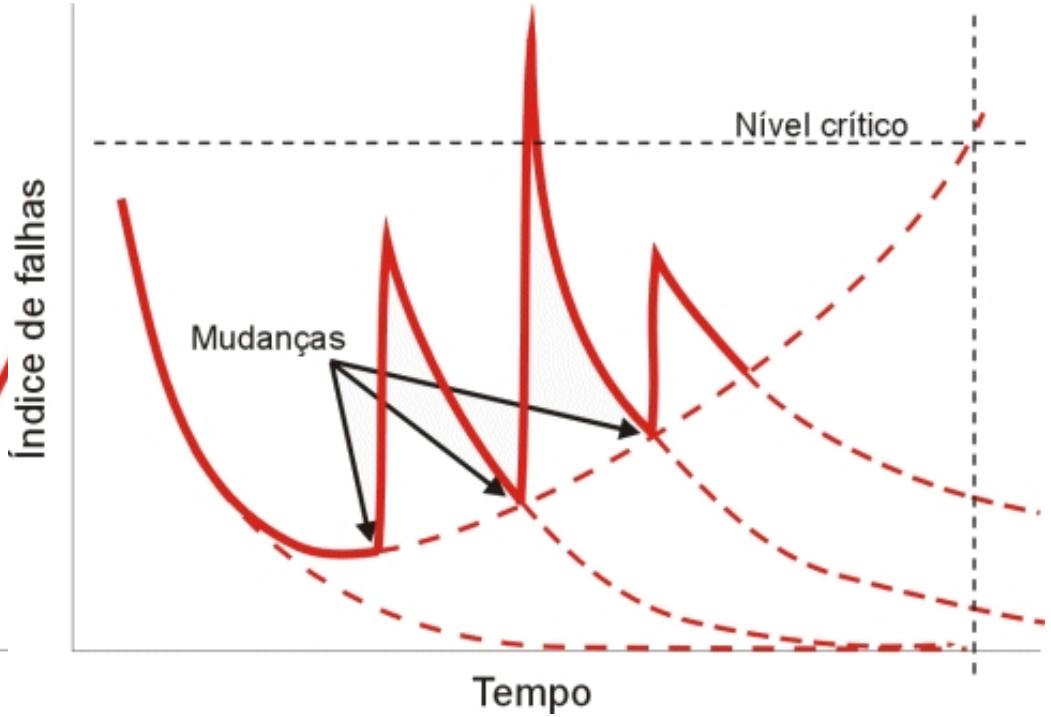
INTRODUÇÃO

Software

- Distinguir software de outros tipos de produtos:
  1. Desenvolvido, não manufaturado: criativa do começo ao fim – não basta projeto e máquinas para fabricar.
  2. Não 'gasta' – poeira, vibração, temperatura etc. não afetam o software. Entretanto, deteriora.
  3. Indústria – movendo para componentes; entretanto, maioria ainda feito por encomenda.

Taxa de Falhas do Software





INTRODUÇÃO

Engenharia de Software

- Progrediu rápido em comparação com outras engenharias
- Ideia: - de 50 anos atrás – sistemas pequenos e feitos por cientistas
  - Problemas matemáticos específicos e pequenos
  - Defeitos nesses sistemas – consequências pequenas
- Hoje – sistemas enormes, em complexidade e tamanho

INTRODUÇÃO

Papel Evolutivo do Software

- 1950 – 1965: Os primeiros anos
  - O hardware sofreu contínua mudança
  - O hardware era de propósito geral
  - O software era de propósito específico
  - Havia poucos profissionais e métodos sistemáticos para o desenvolvimento de software
  - A documentação era praticamente inexistente
  - A programação predominante era de baixo nível

INTRODUÇÃO

Papel Evolutivo do Software

- 1965 – 1975: A segunda era
  - Multiprogramação e sistemas com múltiplos usuários
  - Evolução das linguagens e paradigmas de programação
  - Primeira geração dos SGBDs
  - Aumento do número de profissionais de software
    Bibliotecas de software
  - Aumento do uso de computadores e sistemas de informação
  - Aumento da complexidade do software
  - Crise do Software

INTRODUÇÃO

Papel Evolutivo do Software

- 1975 – 1985: A terceira era
  - Sistemas distribuídos
  - “Inteligência” embutida
  - HW de baixo custo
  - Impacto de consumo

INTRODUÇÃO

Papel Evolutivo do Software

- 1985 - 2000: A quarta era
  - Sistemas desktop poderosos
  - Tecnologias orientadas a objeto
  - Sistemas especialistas
  - Redes neurais artificiais
  - Computação paralela

INTRODUÇÃO

Papel Evolutivo do Software

- 2000 - 2010: A quinta era ?
  - Internet
  - Web-based Systems
  - ...
  - ...
  - ...

INTRODUÇÃO

Papel Evolutivo do Software

- 2010 - ...: A sexta era ?
  - Cloud
  - Mobile
  - Computação Ubíqua
  - …

INTRODUÇÃO

Crise do Software

- 1968
  - Conferência organizada para discutir a Crise do SW
  - Introdução de novo HW baseado em circuitos integrados
  - Viabilizando o desenvolvimento de SWs maiores e mais complexos
  - Desenvolvimento informal não era suficiente
  - Anos de atraso
  - Custo superava as previsões
  - Desempenho insatisfatório
  - Necessidade de novas técnicas e métodos

INTRODUÇÃO

Crise do Software

- Características do software
- Características do processo de desenvolvimento
- Falha das pessoas responsáveis:
  - Gerência sem experiência
  - Profissionais sem boa formação e treinamento
- Falta de métodos adequados para a realização de estimativas e acompanhamento de projeto

INTRODUÇÃO

Crise do Software

- Melhor entendimento das atividades
- Métodos eficazes de especificação, projeto e implementação
- Novas notações e ferramentas
- Não existe uma abordagem “ideal”
- Em geral, abordagem sistêmica e organizada
- Mitos do software:
  - Administrativos
  - Clientes
  - Profissionais

INTRODUÇÃO

Mitos Administrativos

- Já temos um manual repleto de padrões e procedimentos para a construção de software. Isso não oferecerá ao meu pessoal tudo o que eles precisam saber?

INTRODUÇÃO

Mitos Administrativos

- Já temos um manual repleto de padrões e procedimentos para a construção de software. Isso não oferecerá ao meu pessoal tudo o que eles precisam saber?
- Realidade:
  - Será que o manual é usado?
  - Os profissionais sabem que ele existe?
  - Ele reflete a prática moderna de desenvolvimento de software?
  - Ele é completo?

INTRODUÇÃO

Mitos Administrativos

- Meu pessoal tem ferramentas de desenvolvimento de software de última geração.

INTRODUÇÃO

Mitos Administrativos

- Meu pessoal tem ferramentas de desenvolvimento de software de última geração.
- Realidade:
  - É preciso muito mais do que os mais recentes computadores e ferramentas para se fazer um desenvolvimento de software de alta qualidade.

INTRODUÇÃO

Mitos Administrativos

- Se nós estamos atrasados nos prazos, podemos adicionar mais programadores e recuperar o atraso.

INTRODUÇÃO

Mitos Administrativos

- Se nós estamos atrasados nos prazos, podemos adicionar mais programadores e recuperar o atraso.
- Realidade:
  - O desenvolvimento de software não é um processo mecânico igual à manufatura.
  - Acrescentar pessoas em um projeto torna-o ainda mais atrasado.
  - Pessoas podem ser acrescentadas, mas somente de uma forma planejada.

INTRODUÇÃO

Mitos dos Clientes

- Uma declaração geral dos objetivos é suficiente para começar a escrever programas - podemos preencher os detalhes mais tarde.

INTRODUÇÃO

Mitos dos Clientes

- Uma declaração geral dos objetivos é suficiente para começar a escrever programas - podemos preencher os detalhes mais tarde.
- Realidade:
  - Uma definição inicial ruim é a principal causa de fracassos dos esforços de desenvolvimento de software.
  - É importante uma descrição formal e detalhada do domínio da informação, função, desempenho, interfaces, restrições de projeto e critérios de validação.

INTRODUÇÃO

Mitos dos Clientes

- Os requisitos de projeto modificam-se continuamente, mas as mudanças podem ser facilmente acomodadas, porque o software é flexível.

INTRODUÇÃO

Mitos dos Clientes

- Os requisitos de projeto modificam-se continuamente, mas as mudanças podem ser facilmente acomodadas, porque o software é flexível.
- Realidade:
  - Uma mudança, quando solicitada tardiamente num projeto, pode ser maior do que a ordem de magnitude mais dispendiosa da mesma mudança solicitada nas fases iniciais.
  - Obs: Atualmente existem diversos modelos de processo e abordagens que facilitam a gestão de mudanças. Os requisitos de sistemas de diversos domínios são muito voláteis.

INTRODUÇÃO

Mitos dos Profissionais

- Assim que escrevermos o programa e o colocarmos em funcionamento nosso trabalho estará completo.

INTRODUÇÃO

Mitos dos Profissionais

- Assim que escrevermos o programa e o colocarmos em funcionamento nosso trabalho estará completo.
- Realidade:
  - Os dados da indústria indicam que entre 50% e 70% de todo esforço gasto num programa serão despendidos depois que ele for entregue pela primeira vez ao cliente.

INTRODUÇÃO

Mitos dos Profissionais

- Enquanto não tiver o programa "funcionando", eu não terei realmente nenhuma maneira de avaliar sua qualidade.

INTRODUÇÃO

Mitos dos Profissionais

- Enquanto não tiver o programa "funcionando", eu não terei realmente nenhuma maneira de avaliar sua qualidade.
- Realidade:
  - Um programa funcionando é somente uma parte de uma Configuração de Software que inclui todos os itens de informação produzidos durante a construção e manutenção do software.

ENGENHARIA DE SOFTWARE

Engenharia

- Engenharia
  - (...) a aplicação **sistemática** e **regular de conhecimento científico e matemático** ao projeto, construção e operação de máquinas, sistemas e outros, para o **uso prático** e, assim, com **valor econômico**. Uma característica particular dos engenheiros é que eles tomam a sério sua **responsabilidade pela correção** (ou **corretude**), **adequação** e **segurança** dos resultados de seus esforços. (Baber)

ENGENHARIA DE SOFTWARE

- A Engenharia de Software surgiu como resposta aos problemas relacionados ao desenvolvimento de software.
- Ela é definida como “a aplicação de uma abordagem sistemática, disciplinada e possível de ser medida para o desenvolvimento, operação e manutenção do software” (IEEE).
- Além dos aspectos técnicos, a Engenharia de Software também define atividades gerenciais que são importantes durante o desenvolvimento.

ENGENHARIA DE SOFTWARE

- 3 problemas importantes:

ENGENHARIA DE SOFTWARE

- 3 problemas importantes:
  - **Uso prático, valor econômico**: cliente – muito importante; custo.
    - ES = + qualidade, - custo

ENGENHARIA DE SOFTWARE

- 3 problemas importantes:
  - **Uso prático, valor econômico**: cliente – muito importante; custo.
    - ES = + qualidade, - custo
  - **Responsabilidade pela corretude, adequação e segurança**: em alguns casos, programas têm problemas críticos que podem causar morte.

ENGENHARIA DE SOFTWARE

- 3 problemas importantes:
  - **Uso prático, valor econômico**: cliente – muito importante; custo.
    - ES = + qualidade, - custo
  - **Responsabilidade pela corretude, adequação e segurança**: em alguns casos, programas têm problemas críticos que podem causar morte.
  - **Aplicação regular de conhecimento científico e matemático**: corpo de conhecimento para a ES: início, importância da pesquisa…

ENGENHARIA DE SOFTWARE

Exemplo da importância

- Ariane 5

ENGENHARIA DE SOFTWARE

Exemplo da importância

- Ariane 5
  - Em 4 de Junho de 1996 um foguete Ariane 5 não tripulado lançado pela Agência Espacial Europeia explodiu 40 segundos após seu lançamento
  - Primeira viagem do foguete depois de uma década de desenvolvimento – US$7 bilhões
  - Carga avaliada em US$0.5 milhão

ENGENHARIA DE SOFTWARE

Exemplo da importância

- Ariane 5
  - Causa da falha – defeito no software:
    - Um número ponto flutuante de 64 bits – velocidade horizontal do foguete em relação a plataforma – foi convertido para um inteiro de 16 bits
    - O número era maior do que 32.768, maior inteiro 16 bits – a conversão falhou
    - Falta de Tratamento de Exceção

ENGENHARIA DE SOFTWARE

Exemplo da importância

- Míssil Patriot

ENGENHARIA DE SOFTWARE

Exemplo da importância

- Míssil Patriot
  - Em 25 de Fevereiro de 1991, durante a guerra do Golfo, um míssil Patriot americano falhou ao tentar interceptar um Scud iraquiano
  - O Scud atingiu barracas do exército americano, matando 28 soldados

ENGENHARIA DE SOFTWARE

Exemplo da importância

- Míssil Patriot
  - Causa da falha na interceptação – defeito no software:
    - O erro aconteceu no cálculo do tempo transcorrido desde o boot do sistema – defeito de aritmética
    - Ao arredondar um número binário, um erro pequeno foi multiplicado por um número maior, levando a um erro significativo
    - Um Scud viaja a 1.676 metros por segundo, dessa forma o erro fez com que o Patriot não conseguisse alcançá-lo em tempo...

Engenharia de software

- As economias de TODAS as nações desenvolvidas são dependentes de software.
- Cada vez mais sistemas são controlados por software.
- A engenharia de software se dedica às teorias, métodos e ferramentas para desenvolvimento de software profissional
  - Sistemas não-triviais
  - Com base em um conjunto de requisitos

Custos de software

- Os custos de software dominam os custos de sistemas computacionais.
  - Em geral, software custa mais que hardware
- Manter um software custa mais que desenvolvê-lo
- A engenharia de software dedica-se ao desenvolvimento de software com custos adequados
  - Respeitando o cronograma acordado
  - Satisfazendo as necessidades dos clientes
  - Minimizando o custo de manutenção

FAQs sobre engenharia de software

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

FAQs sobre engenharia de software

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

O que é software?

- Programas de computador e artefatos associados
- Produtos de software podem ser
  - **Genéricos** – desenvolvidos para serem vendidos para uma grande variedade de clientes (e.g., Excel e Word)
  - **Personalizados** – desenvolvidos para um único cliente de acordo com as suas especificações
- Um software novo pode ser criado através de
  - desenvolvimento de novos programas
  - configuração de sistemas de software genéricos; ou
  - reutilização de um software existente

O que é engenharia de software?

- Engenharia de software é uma disciplina relacionada com todos os aspectos da produção de software
- ... e propõe ferramentas, técnicas e processos para
  - Entender com precisão qual é o problema
    - as necessidades associadas ao sistema que deve ser construído/modificado
  - Produzir uma solução adequada para esse problema
    - um sistema pronto para usar, levando-se em consideração as necessidades das partes interessadas
  - Levando-se em conta restrições de desenvolvimento e recursos disponíveis

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

- A ciência da computação dedica-se à teoria e aos fundamentos
  - Engenharia de software dedica-se aos aspectos práticos de desenvolvimento e de entrega de software

- Teorias de ciência da computação são ainda insuficientes para atuar como uma base completa para a engenharia de software (diferente de, por exemplo, física e engenharia elétrica)
  - Em outras palavras: não há receitas prontas!

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

- Engenharia de sistemas:
  - Mais ampla
  - Muita ênfase em aspectos de hardware e infra-estrutura
    - Abstração do hardware
    - Organização física das partes do sistema
    - Aspectos de comunicação
  - Engloba a engenharia do software
- Os engenheiros de sistema estão envolvidos em diversas atividades da engenharia de software
  - Projeto da arquitetura
  - Elicitação e especificação de requisitos

O que é processo de software?

- Um conjunto estruturado de atividades, práticas, artefatos e ferramentas necessários para o desenvolvimento de um sistema de software
  - Especificação
  - Projeto
  - Validação
  - Evolução
- Exemplos: Processo Unificado (RUP), Programação Extrema, UML Components

O que é processo de software?

- Alguns elementos de um processo:
  - Modelos de sistema
    - Modelos gráficos que podem/devem ser produzidos e as notações que devem ser empregadas
    - Restrições aplicadas aos modelos de sistema
  - Recomendações de boas práticas de projeto
  - Atividades que devem ser seguidas em determinada ordem
  - Às vezes também prescrevem ferramentas
- Um processo adere a um ou mais modelos de processo

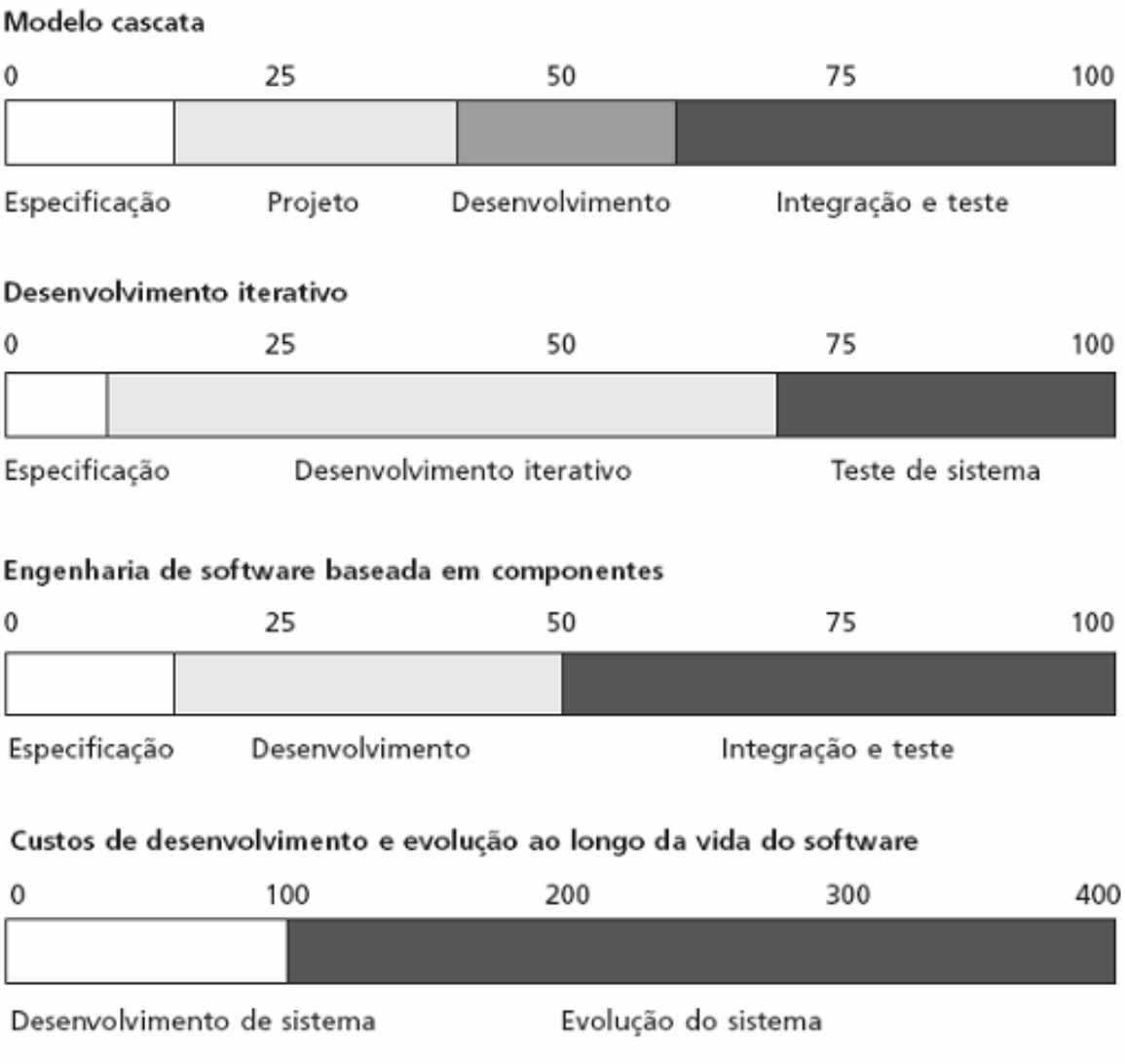
O que é um modelo de processo de software?

- Uma representação simplificada de um processo de software, apresentado sob uma perspectiva específica
  - Incluindo algumas atividades e sua organização de alto nível
- Modelos gerais de processo
  - Cascata
  - Desenvolvimento iterativo
  - Engenharia de software baseada em componentes
- Os modelos não são necessariamente mutuamente excludentes!
- Representações de modelos de processo
  - Modelo de *workflow* – sequência de atividades
  - Modelo de fluxo de dados – fluxo de informações
  - Modelo de papel/ação – quem faz o quê

Quais são os custos da engenharia de software?

- 60% dos custos são custos de desenvolvimento
- 40% são custos de testes
- Para software sob encomenda, os custos de evolução normalmente excedem os de desenvolvimento
- Os custos variam dependendo do tipo de sistema que está sendo desenvolvido e dos requisitos do sistema, tais como desempenho e confiabilidade
- A distribuição de custos depende do modelo de desenvolvimento que é usado

Distribuição de custos nas atividades



O que é CASE (*Computer-Aided Software Engineering*)

- Sistemas de software que se destinam a fornecer apoio automatizado para as atividades de desenvolvimento de software
- Sistemas CASE são usados frequentemente para apoiar um método específico
- *Upper-CASE*
  - Ferramentas para apoiar as atividades iniciais de processo de requisitos e de projeto
- *Lower-CASE*
  - Ferramentas para apoiar as atividades finais tais como programação, *debugging* e teste

Quais são os atributos de um bom software?

- O software deve fornecer a funcionalidade e o desempenho requeridos para o usuário e deve apresentar certas características
- Facilidade de manutenção
  - Deve ser fácil e barato fazer com que o sistema, depois de implantado, evolua para atender às necessidades dos clientes
- Confiabilidade
  - O software deve funcionar conforme sua especificação
- Eficiência
  - O software deve usar o mínimo de recursos e prover máxima funcionalidade
- Usabilidade
  - O software deve ser compreensível e fácil de usar

Quais são os desafios-chave enfrentados pela engenharia de software?

- Heterogeneidade
  - Sistemas de software devem suportar diferentes plataformas de hardware e ambientes de execução
- Entrega
  - O sistema deve ser entregue ao cliente no menor tempo possível, com o menor custo possível
- Confiança
  - O usuário deve poder justificadamente depositar sua confiança no sistema
- Escala
  - O sistema deve funcionar adequadamente mesmo quando um grande número de usuários o está usando

Responsabilidade profissional e ética

- A engenharia de software envolve responsabilidades mais amplas do que simplesmente a aplicação de habilidades técnicas
- Os engenheiros de software devem se comportar de modo honesto e eticamente responsável para serem respeitados como profissionais
- O comportamento ético é mais do que simplesmente a sustentação de leis

Questões de responsabilidade profissional

- Confidencialidade
  - Os engenheiros de software devem normalmente respeitar a confidencialidade de seus funcionários ou clientes, independentemente de ter ou não assinado um acordo formal
    - Caso não aceitem essas condições, devem deixar isso explícito para seus contratantes
- Competência
  - Os engenheiros não devem conscientemente aceitar um trabalho que esteja fora de sua competência

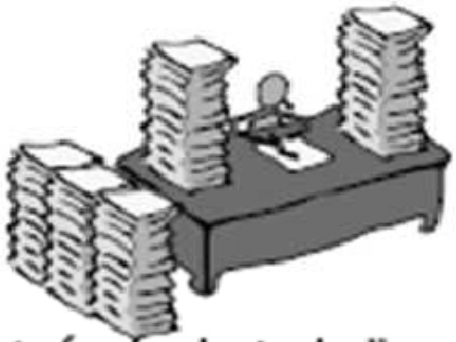
Questões de responsabilidade profissional

- Direitos sobre propriedade intelectual
  - Desenvolvedores devem estar cientes das leis locais que regem o uso de propriedade intelectual, tais como patentes, direitos autorais, etc.
  - Eles devem tomar cuidado para assegurar que a propriedade intelectual dos funcionários e clientes seja protegida

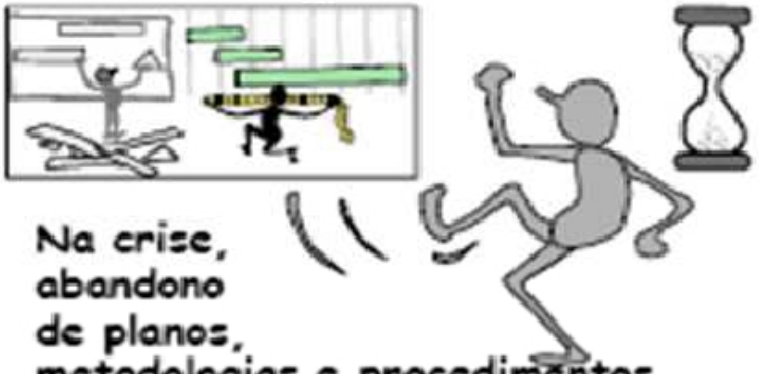
Dilemas éticos

- Discordância, em princípio, das políticas da gerência sênior
- Um funcionário age de uma forma não ética e libera um sistema de segurança crítico sem finalizar o teste do sistema
- Participação no desenvolvimento de sistemas de armamentos militares ou de sistemas nucleares

Cenas de uma empresa de software ...



Acúmulo de trabalho
e crises constantes



Na crise,
abandono
de planos,
metodologias e procedimentos

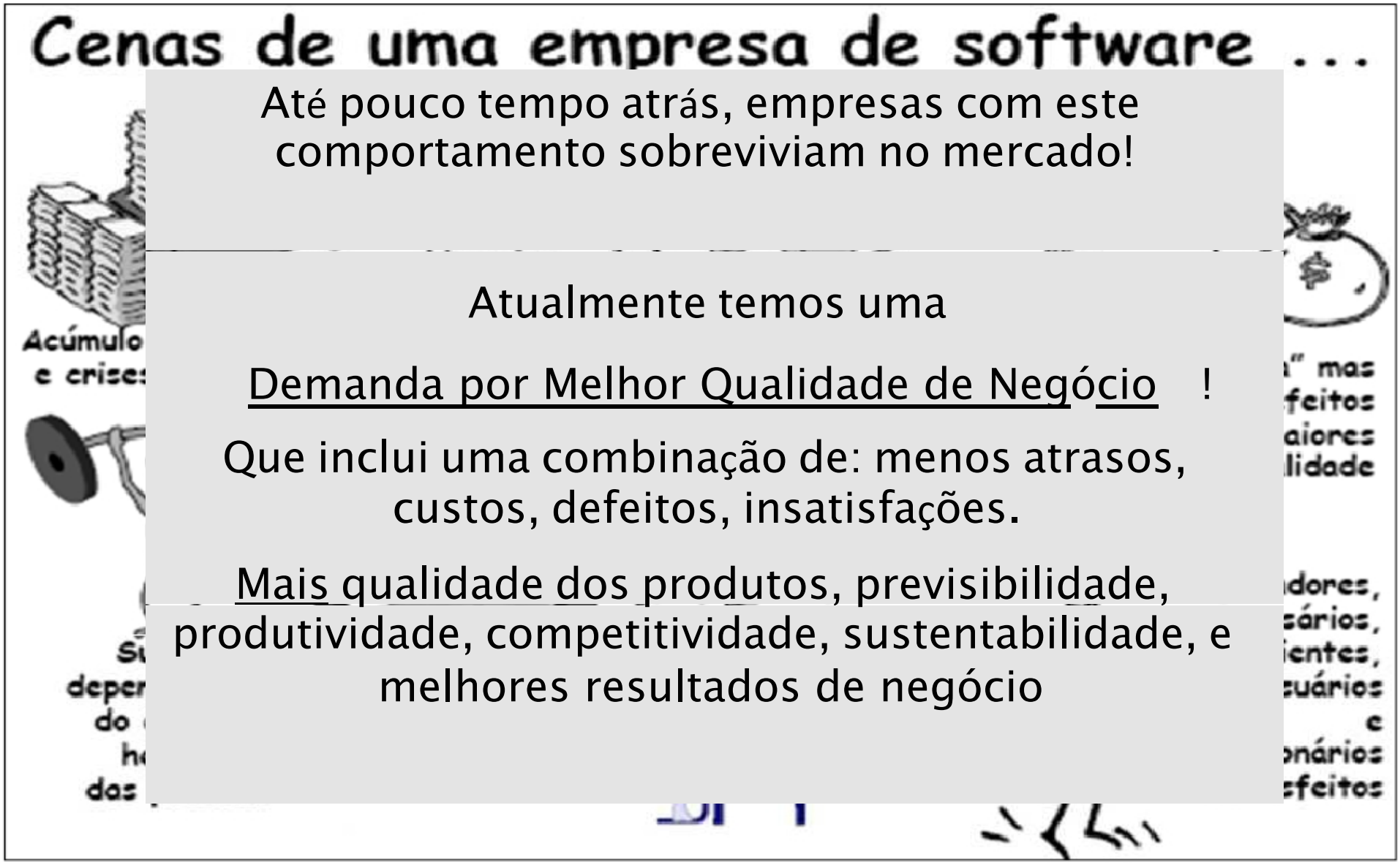
Produto até "funciona" mas
com mais defeitos
prazos e custos maiores
e menos funcionalidade

Sucesso
depende muito
do esforço
heróico
das pessoas

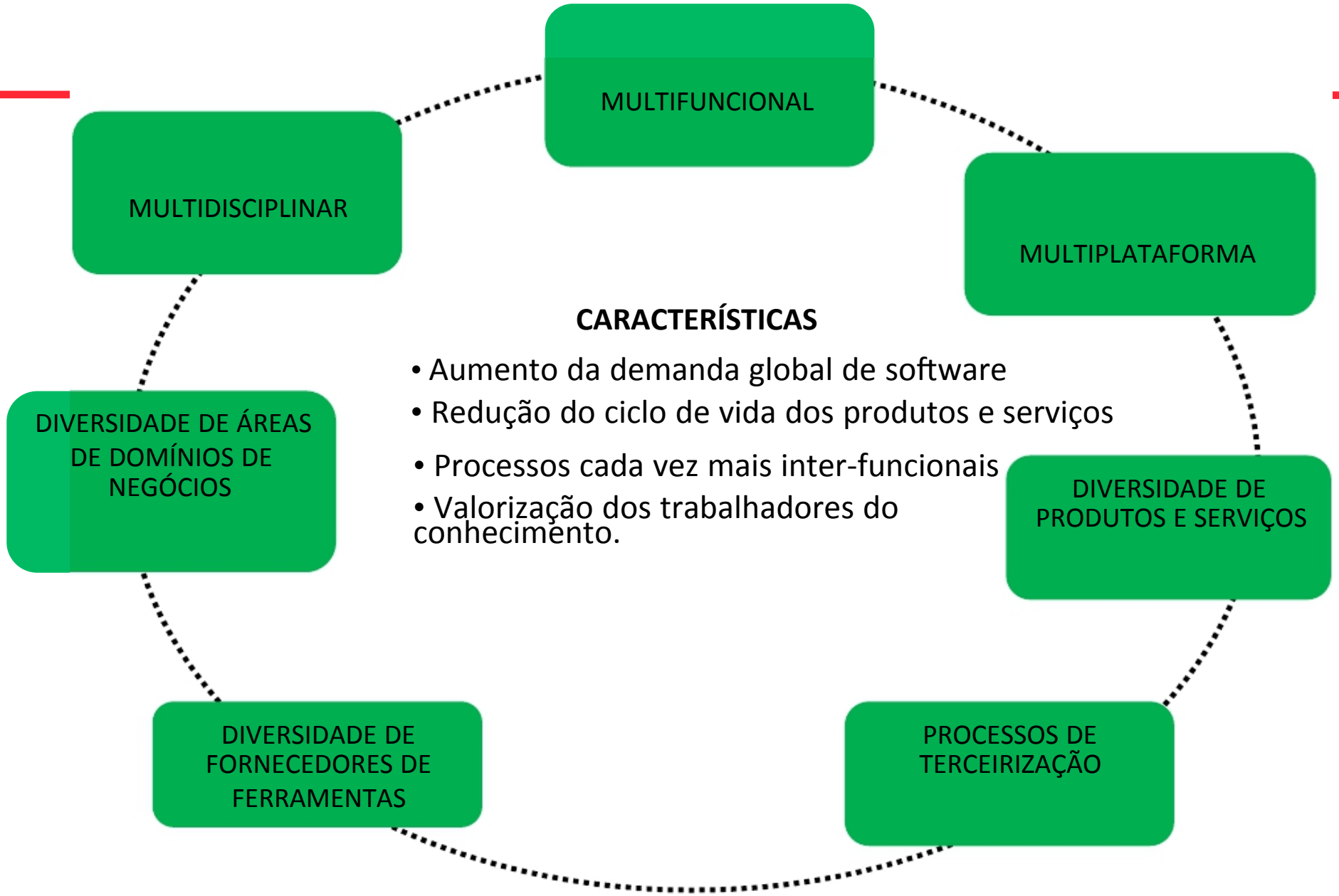
Pouca
repetibilidade

Grande distância
entre
teoria
e
prática

Pesquisadores,
empresários,
clientes,
usuários
e
funcionários
insatisfeitos


Cenas de uma empresa de software ...
Até pouco tempo atrás, empresas com este comportamento sobreviviam no mercado!
Atualmente temos uma
Demanda por Melhor Qualidade de Negócio !
Que inclui uma combinação de: menos atrasos, custos, defeitos, insatisfações.
Mais qualidade dos produtos, previsibilidade, produtividade, competitividade, sustentabilidade, e melhores resultados de negócio

Características dos Ambientes de Empresas de TI



Características do contexto de um Departamento de Desenvolvimento de SI

Multifuncional: desenvolve processos inter-funcionais
Exemplos de áreas funcionais: Atendimento, Comercialização, Negócios, Análise e Design, Desenvolvimento (Codificação), Testes e Homologação, Suporte, Infraestrutura, Qualidade, Planejamento e Controle de Produção...

Multidisciplinar: envolve inúmeras áreas do conhecimento.
Exemplos: Engenharia de Software, Engenharia de Sistemas, Gestão de Processos, Gestão de Projetos, Gestão de Operações, Gestão de Serviços, Gestão de Competências, Qualidade, Administração de Banco de Dados, Arquitetura, Redes...

Características do contexto de um Departamento de Desenvolvimento de SI

Multiplataforma operacional e de aplicações. Exemplos: mainframe, web, sistemas integrados, sistemas distribuídos, cliente-servidor, sistemas de processamento em lote, sistemas on-line...

Diversidade de demandas de produtos e serviços. Exemplos: Propostas, projetos novos, rotinas, componentes, diferentes categorias de manutenções de sistemas, terceirização, serviços ...

Diversidade de aplicações de negócios. Exemplos: Educação, Finanças, Saúde, Gestão, Geoprocessamento ...

Características do contexto de um Departamento de Desenvolvimento de SI

Diversidade de Fornecedores: Embora a diversidade de fornecedores no portfólio de ofertas da organização seja saudável para a dinâmica do mercado, por outro lado, causa um enorme problema para os gestores em administrar não só os múltiplos fornecedores como os vários modelos de contrato.

Características do contexto de um Departamento de Desenvolvimento de SI

Processos de Terceirização: Embora a terceirização tenha várias vantagens, ela deve ser praticada com cautela. O processo de terceirização em uma organização deve levar em conta diversos fatores de interesse, tais como a redução de custos e principalmente o foco na sua atividade-fim. A terceirização precisa estar em conformidade com os objetivos estratégicos da organização, os quais irão revelar em que pontos ela poderá alcançar resultados satisfatórios .

Conceitos de Engenharia de SW

- Engenheiros de software devem:

  dependendo do problema a ser resolvido,
  das restrições de desenvolvimento, e
  dos recursos disponíveis

adotar uma abordagem sistemática e organizada para seu trabalho, além de usar ferramentas e técnicas apropriadas

Engenharia de Software
Tecnologia em Camadas

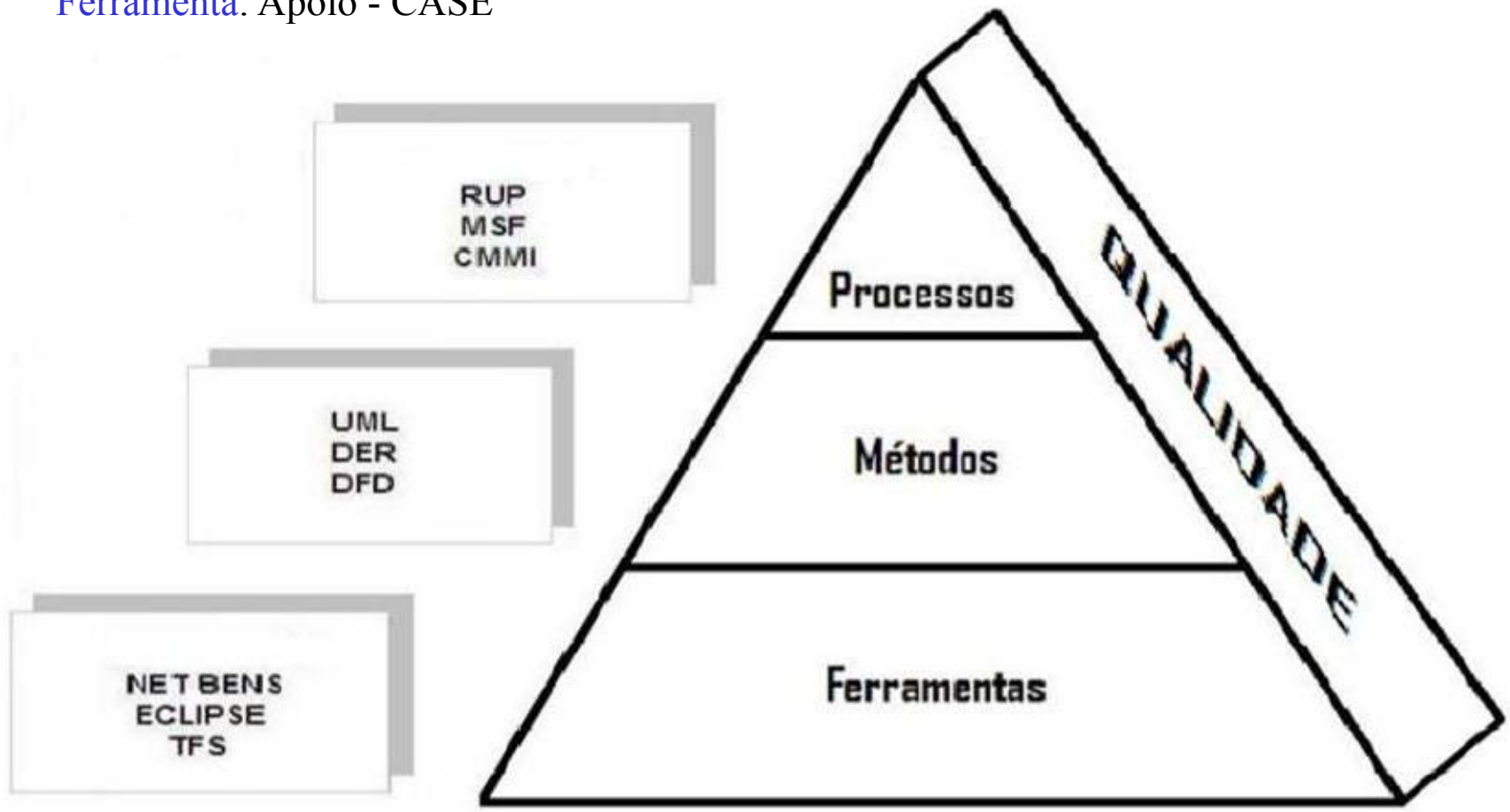
Pressman 2006

Processo x Método x Ferramenta

Processo (procedimento/metodologia/ciclo de vida): um conjunto de *atividades* aplicáveis, seguindo um modelo prescritivo .

Método: como fazer? Uso de técnicas

Ferramenta: Apoio - CASE



O que é processo de software?

- É um conjunto de atividades cuja meta é o desenvolvimento ou evolução de software.
- As atividades genéricas em todos os processos de software são:
  - **Especificação** – o que o sistema deve fazer e suas restrições de desenvolvimento.
  - **Desenvolvimento** – produção do sistema de software.
  - **Validação** – verificação de que o software é o que o cliente deseja.
  - **Evolução** – mudança do software em resposta às demandas de mudança.

O que é um modelo de processo de software?

- Uma representação simplificada de um processo de software, apresentado sob uma perspectiva específica.

- Exemplos de modelos de processo são:
  - Modelo de *workflow* – seqüência de atividades;
  - Modelo de fluxo de dados – fluxo de informações;
  - Modelo de papel/ação – quem faz o quê.

O que é um modelo de processo de software?

- Modelos gerais de processo
  - Cascata;
  - Desenvolvimento iterativo;
  - Engenharia de software baseada em componentes.

Quais são os métodos de engenharia de software?

- Abordagens estruturadas para desenvolvimento de software que incluem modelos de sistema, técnicas, notações, regras, recomendações de projeto, guias de processos,etc.

- Método ou técnica é um procedimento formal para produzir algum resultado

Quais são os métodos de engenharia de software?

- Regras
  - Restrições aplicadas aos modelos de sistema;
- Recomendações:
  - Recomendações de boas práticas de projeto;
- Guia de processo:
  - Quais atividades devem ser seguidas.

Ferramentas

CASE – *Computer Aided Software Engineering*

Sistemas de software destinados ao apoio automatizado às atividades dos processos e métodos de software

Foco na Qualidade

- Processo: CMMI, MPSBR, ITIL, ISO 12207
- Produto: ISO 9126, ISO 25051
- Projeto: PMBOK, SCRUM
- Pessoas: capacitação, experiência, certificações

Objetivos da Engenharia de Software

- Controle sobre o desenvolvimento de software dentro de **custos**, **prazos** e níveis de **qualidade** desejados
- Produtividade no desenvolvimento, operação e manutenção de software
- Qualidade *versus* Produtividade
- Permitir que profissionais tenham controle sobre o desenvolvimento de software dentro de custos, prazos e níveis de qualidade desejados

Características da Engenharia de Software

- A Engenharia de Software se refere a software (sistemas) desenvolvidos por grupos ao invés de indivíduos
- Usa princípios de engenharia ao invés de arte, e
- Inclui tanto aspectos técnicos quanto não técnicos

O que é um software de qualidade?

- O software que satisfaz os requisitos solicitados pelo usuário. Deve ser fácil de manter, ter boa performance, ser confiável e fácil de usar
- Alguns atributos de qualidade
  - **Manutenibilidade**
    - O software deve evoluir para atender os requisitos que mudam
  - **Eficiência**
    - O software não deve desperdiçar os recursos do sistema
  - **Usabilidade**
    - O software deve ser fácil de usar pelos usuários para os quais ele foi projetado

Importância da Engenharia de Software

- Qualidade de software e produtividade garantem:
  - Disponibilidade de serviços essenciais
  - Segurança de pessoas
  - Competitividade das empresas
    - Produtores
    - Consumidores

Quais são os desafios-chave enfrentados pela engenharia de software?

- Heterogeneidade
  - Técnicas de desenvolvimento para construção de software que podem lidar com plataformas heterogêneas e ambientes de execução;
- Entrega
  - Técnicas de desenvolvimento para conduzir a entrega mais rápida de software;
- Confiança
  - Técnicas de desenvolvimento que mostram que o software pode ter a confiança dos seus usuários.

Conclusões

- A engenharia de software é uma disciplina de engenharia relacionada com todos os aspectos de produção de software.

- Os produtos de software consistem em programas desenvolvidos e documentação associada. Os atributos essenciais do produto são: manutenibilidade, confiança, eficiência e aceitabilidade.

Conclusões

- O processo de software compreende todas as atividades envolvidas no desenvolvimento de produtos de software.

- Métodos são meios organizados de produção de software. Eles incluem sugestões para o processo a ser seguido, as notações a serem usadas, modelos de sistemas a serem desenvolvidos, regras que regem estes modelos e diretrizes e técnicas para o projeto.

Conclusões

- Ferramentas CASE são sistemas de software projetados para apoiar as atividades rotineiras no processo de software, tais como edição de diagramas de projeto, verificação da consistência de diagramas e rastreabilidade de testes de programa realizados.

- Engenheiros de software têm responsabilidades com a profissão de engenharia e a sociedade. Eles não devem se preocupar apenas com assuntos técnicos.

Bibliografia

- *Blaha, Rumbaugh. Modelagem e Projetos Baseados em Objetos com UML2*
- *Pressman, Roger S. Engenharia de Software. 6.ed. - Rio de Janeiro: McGraw-Hill, 2006.*
- *Sommerville, I. Engenharia de Software. 8.ed. – São Paulo : Addison-Wesley, 2007.*
- *Pfleenger, Shari L. Engenharia de Software. 2 ed. – São Paulo: Prentice Hall,2007.*
- *O'Brien, James A. Sistemas de Informação e as decisões gerenciais na era da Internet. 3ª Ed. São Paulo, Saraiva, 2009.*