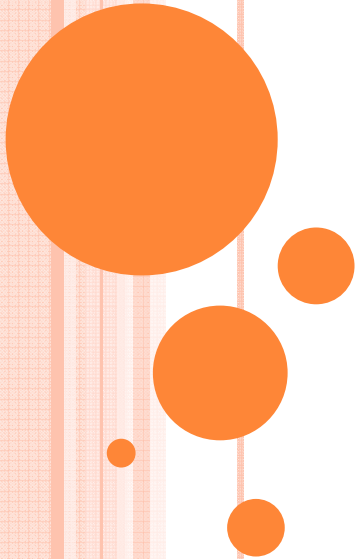


Plano de aulas e Introdução

INE 5419 – Engenharia de Software II

Prof. Raul Sidnei Wazlawick

UFSC-CTC-INE

2012.1

Carga horária

- 72 horas
- 3ª 13h30-15h10
- 5ª 13h30-15h10

EMENTA

- Evolução da prática de desenvolvimento de software;
- qualidade de artefatos de software;
- modularidade e reusabilidade;
- modelagem estrutural e dinâmica em orientação a objetos, diferentes visões de um sistema;
- metodologias de análise e projeto orientadas a objetos;
- teste de software;
- manutenção de software;
- modelos de ciclo de vida;
- engenharia reversa;
- modelagem formal de sistemas;
- abordagens voltadas ao reuso de software;
- gerenciamento do processo de produção de software e técnicas de apoio ao gerenciamento do processo de produção de software;
- apoio automatizado ao desenvolvimento de software.

Objetivo geral

- Dar ao aluno condições de perceber o desenvolvimento de software como um processo de engenharia, baseado em planejamento, medição e melhoria contínua.

Objetivos específicos

- Apresentar os conceitos de **qualidade de processo e de artefato** de software.
- Apresentar a engenharia de software como um **processo** com seus **atributos de qualidade**.
- Apresentar os diferentes **ciclos de vida** do software.
- Dar ao aluno condições de realizar o **planejamento** do desenvolvimento de software.
- Mostrar como gerenciar **riscos** no processo de desenvolvimento de software.
- Discutir as diferentes formas de **organização do processo** de desenvolvimento de software e modelos de **reusabilidade**.
- Apresentar técnicas de **especificação formal** de software.
- Identificar as **etapas de implementação, teste e manutenção** de sistemas de computação e ser capaz de realizá-los e/ou coordená-los.
- Conhecer e saber aplicar métodos de **controle da qualidade** do processo de software.

O Processo de Engenharia de Software [20 h/a]

- 06/03 – Introdução [AEX]
- 08/03 – Processo [AEX]
- 13/03 – Modelos de Processo Prescritivos [AEX]
- 15/03 – Modelos de Processo Prescritivos [AEX]
- 20/03 – Modelos Ágeis [AEX]
- 22/03 – Modelos Ágeis [AEX]
- 27/03 – Modelos Ágeis [AEX]
- 03/04 – Processo Unificado [AEX]
- 05/04 – Processo Unificado [AEX]
- 10/04 – PROVA 1

Planejamento e gerência de projetos

- 12/04 – Planejamento [AEX]
- 17/04 – Estimação de Esforço [AEX]
- 19/04 – Estimação de Esforço [AEX]
- 24/04 – CIbSE
- 26/04 – CIbSE
- 03/05 – Gerenciamento de Riscos [AEX]
- 08/05 – Gerenciamento de Projeto [AEX]
- 10/05 – Gerenciamento de Configuração e Mudança [AEX]
- 15/05 – PROVA 2

QUALIDADE

- 17/05 – SBSI
- 22/05 – Qualidade de Produto [AEX]
- 24/05 – Qualidade de Processo [AEX]
- 29/05 – Teste [AEX]
- 31/05 – Teste [AEX]
- 05/06 – Manutenção [AEX]
- 12/06 – PROVA 3

Especificação Formal de Software

- 14/06 - Invariantes [AEX]
- 19/06 - Invariantes [AEX]
- 21/06 - Invariantes [AEX]
- 26/06 - Contratos [AEX]
- 28/06 - Contratos [AEX]
- 03/07 - PROVA 4
- 05/07 - Prova de recuperação

AVALIAÇÃO

- MF = (3*prova1 + 4*prova2 + 5*prova3 + 6*prova4)/18
- Provas individuais SEM consulta.

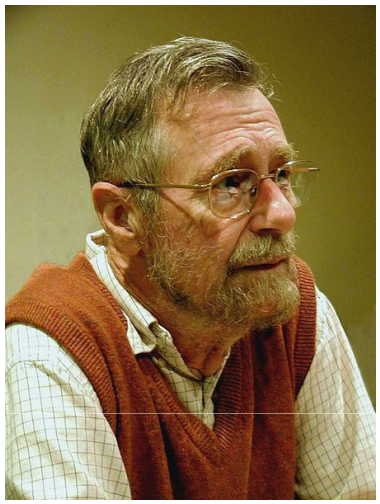
- Recuperação = (MF + ProvaRec) / 2
  - Para alunos com média arredondada entre 3,0 e 5,5.
- Arredondamento sempre para a fração mais próxima.

- Freqüência obrigatória de 75%.
  - É permitido ter no máximo 9 faltas

Introdução

- A crise do software
- Os mitos do software
- Definição de engenharia de software

CRISE DO SOFTWARE



- O termo “crise do software” foi usado pela primeira vez com impacto por Dijkstra (1972).
- Ele avaliava que considerando o rápido progresso do hardware e das demandas por sistemas cada vez mais complexos, os desenvolvedores simplesmente estavam se perdendo, porque a Engenharia de Software, na época era uma disciplina incipiente.

Problemas relatados por Dijkstra em 1972

- Projetos que estouram o cronograma.
- Projetos que estouram o orçamento.
- Produto final de baixa qualidade ou não atendendo aos requisitos.
- Produtos não gerenciáveis e difíceis de manter e evoluir.

Embora a Engenharia de Software tenha evoluído como ciência, sua aplicação na prática ainda é muito limitada.

Crise do Software?

- Não! É a crise dos desenvolvedores de software menos preparados

GARTNER GROUP, 2000

- Questionário aplicado à alta direção de 200 empresas de porte médio/grande, sobre as principais falhas/dificuldades com a Informática:
  - Cumprimento dos prazos 26,3%
  - Custos elevados 25,4%
  - Prioridade desenvolvimento x manutenção 25,4%
  - Manutenção dos sistemas em uso 21,1%
  - Recrutar profissionais qualificados 18,4%

Frustrações dos usuários

- Esperança que pelo menos parte da promessa da informática se cumpra;
- Frustração pela pequena parte da promessa já cumprida, em razão de:
  - Erros, falhas e inadequação dos produtos de software
  - Insegurança na utilização
  - Prazos excessivamente longos
  - Custos altos e constantes
  - Constante necessidade de manutenções

Frustrações dos desenvolvedores

- Baixa produtividade no desenvolvimento;
- Baixa qualidade do produto gerado (erros e adequação às necessidades do usuário);
- Impossibilidade de cumprir prazos e custos;
- Dificuldade em treinar os profissionais nas novas tecnologias;
- Mudanças constantes em TI/SI (insegurança e necessidade constante de atualização)

Grady Booch

- “uma doença que dure tanto tempo quanto esta, francamente, deveria ser chamada de normalidade”

- Está mais para uma “aflição crônica”.

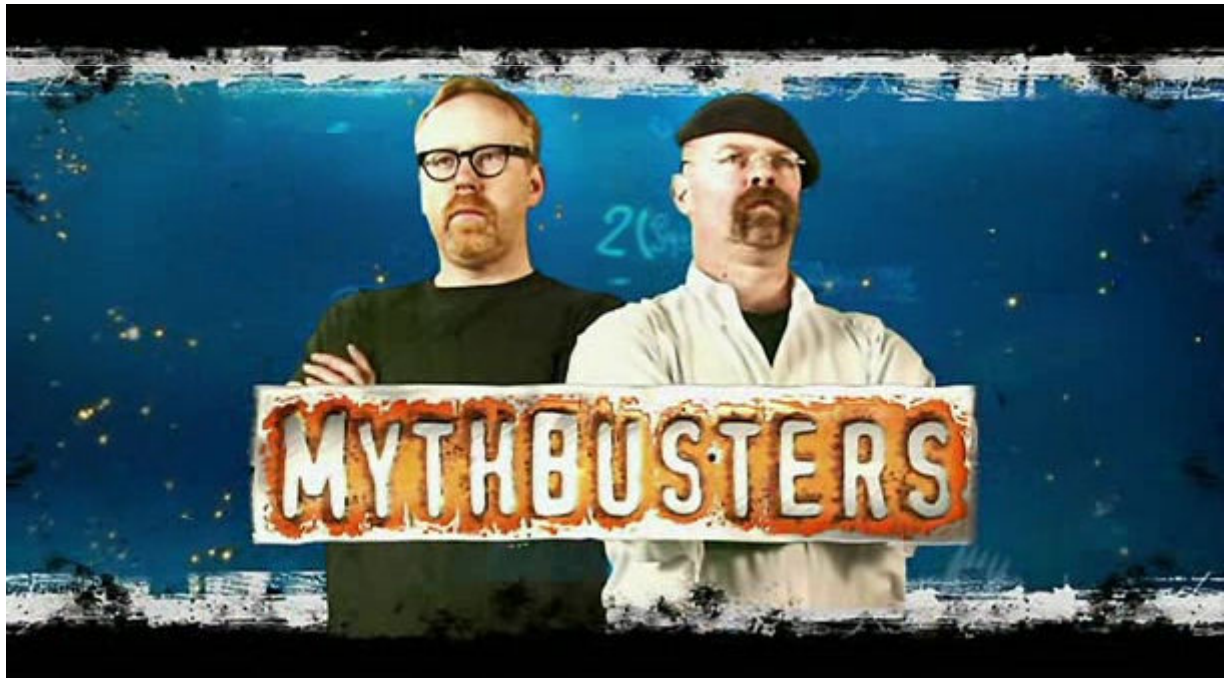
Crise do Software e Idade Média

- Antes da revolução industrial, os sapatos eram feitos de forma muito individual. Nesses tempos mais remotos, os sapateiros faziam cada par de sapatos de forma única para cada cliente, desde a obtenção da matéria prima até o produto final.
- As semelhanças com a indústria de software começam logo aí. Primeiro, porque as técnicas de desenvolvimento de software ainda não estão totalmente maduras e consolidadas. Afinal, a variedade de técnicas que surgiram nas últimas décadas é enorme.
- Em segundo lugar, existe uma tendência muito forte em desenvolver software sem aproveitar o material produzido no passado. E para piorar, além de entregá-lo quase sempre mal documentado, a maior parte do conhecimento envolvido na sua construção permanece apenas na cabeça dos desenvolvedores, o que deixa a situação muito parecida com a do sapateiro do exemplo.

http://www.ebah.com.br/content/ABAAAAeI4AH/crise-software

Mitos do Software (Pressman)

- Administrativos
- Do cliente
- Do profissional



Mitos Administrativos

- *A existência de um manual de procedimentos e padrões é suficiente para a equipe produzir com qualidade.*
  - Na verdade deve-se questionar se o manual é realmente usado, se ele é completo e atualizado.
  - Deve-se trabalhar com processos que possam ser gerenciáveis e otimizados, ou seja, sempre que a equipe identificar falhas no processo deve haver um processo para modificar o processo
- *A empresa deve produzir com qualidade, pois tem ferramentas e computadores de última geração.*
  - Na verdade ferramentas e computadores de boa qualidade são condições necessárias, mas não suficientes.
- *Se o projeto estiver atrasado sempre é possível adicionar mais programadores para cumprir o cronograma.*
  - O desenvolvimento de software é uma tarefa altamente complexa.
  - Adicionar mais pessoas sem que houvesse antes um planejamento para isso pode causar mais atrasos ainda

Mitos do Cliente

- *Uma declaração geral de objetivos é suficiente para iniciar a fase de programação. Os detalhes podem ser adicionados depois.*
  - É verdade que não se pode esperar que a especificação inicial do sistema esteja correta e completa antes de iniciar a programação.
  - Mas ter isso como meta é péssimo.
  - Deve-se procurar obter o máximo de detalhes que for *possível* antes de iniciar a construção do sistema.
  - Técnicas mais sofisticadas de análise de requisitos e uma equipe bem treinada poderão ajudar a construir as melhores especificações possíveis sem perda de tempo.
- *Os requisitos mudam com freqüência, mas sempre é possível acomodá-los, pois o software é flexível.*
  - O software só será efetivamente flexível se for construído com esse fim.
  - É necessário entre outras coisas identificar os requisitos permanentes e transitórios, e, no caso dos transitórios, preparar o sistema para sua mudança, pela utilização de padrões de projeto adequados.

MITOS DO PROFISSIONAL

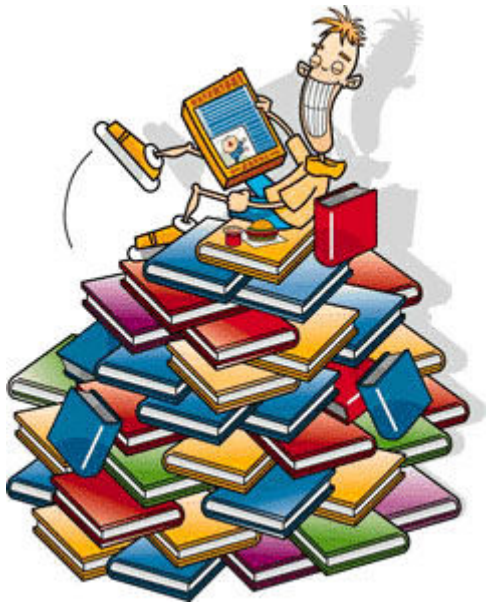
- *Assim que o programa for colocado em operação nosso trabalho terminou.*
  - Na verdade, ainda haverá *muito* esforço a ser dispendido depois da instalação do sistema devido a erros dos mais diversos tipos.
- *Enquanto o programa não estiver funcionando, não será possível avaliar sua qualidade.*
  - Na verdade o programa é apenas um dos artefatos produzidos no processo de construção do software (possivelmente o mais importante, mas não o único).
  - Existem formas de avaliar a qualidade de artefatos intermediários como casos de uso e modelos conceituais para verificar se estão adequados mesmo antes da implementação do sistema.

Conclusão

- De nada adianta apenas estar consciente dos mitos.
- Para produzir software com mais qualidade e confiabilidade é necessário utilizar um série de práticas, algumas das quais serão apresentadas nesta disciplina.

Definição da Engenharia de Software

- Segundo a Desciclopédia (2010), Engenharia de Software pode ser definida assim:
  - "A Engenharia de Software forma um aglomerado de conceitos que dizem absolutamente nada e que geram no estudante desta área um sentimento de Nossa, li 15 kg de livros desta matéria e não aprendi nada. É tudo bom senso.".

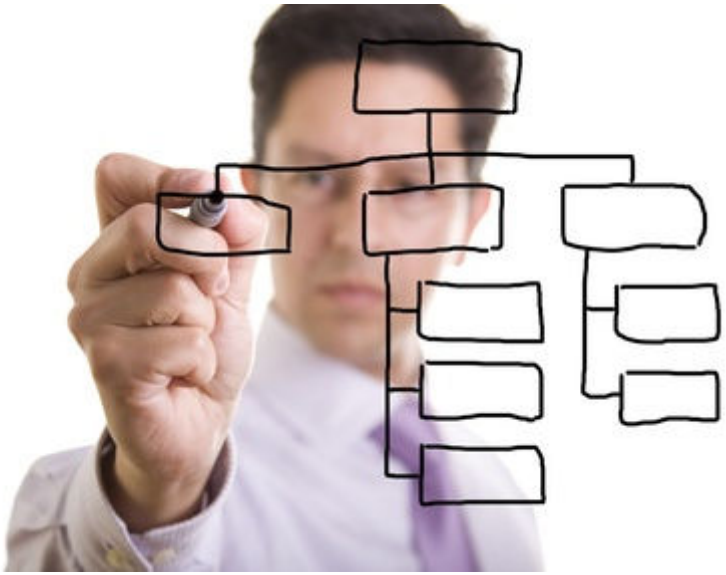


NÃO É SIMPLES CONCEITUAR E PRATICAR A ENGENHARIA DE SOFTWARE. MAS É *NECESSÁRIO*.

- Nesta disciplina vamos aprender:
  - Processos de engenharia de software são diferentes dependendo do tipo de software que se vai desenvolver.
  - Dependendo do nível de conhecimento ou estabilidade dos requisitos deve-se optar por um ou outro ciclo de vida.
  - Uma área aparentemente tão subjetiva como “riscos” pode ser sistematizada e tratada efetivamente como um processo de engenharia, sem as características de não determinismos e relatividade absoluta que usualmente se observa.
  - Existem formas objetivas e padronizadas para mensurar o esforço do desenvolvimento de software de forma a gerar números que sejam efetivamente realistas, o que já vem sendo comprovado em diversas empresas.

O Engenheiro de Software

- Uma das primeiras confusões que se faz nesta área é confundir o *desenvolvedor* com o *engenheiro* de software.
- Isso equivale a confundir o engenheiro civil com o pedreiro ou com o mestre de obra.



- O desenvolvedor, seja ele analista, projetista, programador ou gerente de projeto, é um *executor* do processo de construção de software.
- Os desenvolvedores, de acordo com seus papeis, têm a responsabilidade de descobrir os requisitos e transformá-los em um produto executável.
- Mas o engenheiro de software tem um meta-papel em relação a isso.
- Pode-se dizer que o engenheiro de software não coloca a mão na massa, assim como o engenheiro civil não vai à obra assentar tijolos ou concretar uma laje.

- Então, o engenheiro de software não é um desenvolvedor que trabalhe na produção de código.
- Porém, a comparação com a engenharia civil termina por aqui, já que o engenheiro civil será o responsável pela especificação do projeto.
- Na área de computação, a especificação do projeto fica a cargo do analista e projetista, o primeiro com a responsabilidade de identificar os requisitos e o segundo com a responsabilidade de desenhar uma solução que utilize a tecnologia para transformar estes requisitos em um sistema executável.

- O engenheiro de software assemelha-se assim mais ao engenheiro de produção.
- Ele deve fornecer aos desenvolvedores (inclusive analistas e projetistas), as ferramentas e processos que estes deverão usar e ele será o responsável por verificar que tais ferramentas e processos sejam efetivamente usados, que sejam usados de forma otimizada e que caso apresentem qualquer problema ele será responsável por realizar as modificações necessárias no próprio processo, garantindo assim sua contínua melhoria.

Engenheiro de Software X Gerente de Projeto

- O gerente de projeto deve planejar e garantir que o projeto seja executado de forma adequada dentro dos prazos e orçamento especificados.
- Mas o gerente de projeto tem uma responsabilidade mais restrita ao projeto em si e não ao processo de produção.
- Neste sentido, o gerente de projeto também é um executor, ele utiliza as disciplinas definidas no processo de engenharia de software para gerenciar seu projeto específico, mas ele não é necessariamente o responsável pela evolução destes processos nem necessariamente pela sua escolha.
  - Este papel cabe ao engenheiro de software.

Resumo dos Papeis

- Engenheiro de Software
- Gerente de Projetos
- Analista
- Projetista
- Programador

Evolução da Engenharia de Software

- 1940 – Não havia software.
- 1960 a 1980 – Crise do software (mitos).
- 1980 a 1990 – Balas de prata.
- 2000 – Métodos ágeis.
- 2004 – Swebok.

Tipos de software do ponto de vista da engenharia

- Software básico
- Software de tempo real
- Software comercial
- Software científico e de engenharia
- Software embutido e embarcado
- Software pessoal (aplicativos)
- Jogos
- Inteligência artificial
- ...

Princípios da engenharia de software

- Decomposição
- Abstração
- Generalização
- Padronização
- Flexibilização
- Formalidade
- Rastreabilidade
- Desenvolvimento iterativo
- Gerenciamento de requisitos
- Arquiteturas baseadas em componentes
- Modelagem visual
- Verificação contínua da qualidade
- Controle de mudanças
- Gerenciamento de riscos