

# Apresentação da Disciplina de Engenharia de Software II

Leonardo Gresta Paulino Murta  
leomurta@ic.uff.br

# Apresentações

- Quem sou eu?
  - Leonardo Murta
  - <http://www.ic.uff.br/~leomurta>
- Quem são vocês?
  - Nome?
  - Estágio? Projeto de Aplicação? Iniciação Científica?
  - O que achou de Engenharia de Software I?
  - Expectativas para Engenharia de Software II?

# Relembrando, o que é Engenharia de Software?

“Engenharia de Software é a aplicação de uma abordagem **sistemática, disciplinada e quantificável** ao desenvolvimento, operação e manutenção de software”

IEEE Std 610.12 (1990)

# Mas eu já sei modelar e programar!

- **Por que preciso de Engenharia de Software II?**
  - **Modelar e programar** são parte **importante** do processo de Engenharia de Software, **mas não são tudo!**
- **Precisamos também saber...**
  - como estimar um projeto (tamanho, custo, cronograma),
  - como monitorar o andamento de um projeto,
  - como testar o software,
  - como controlar a evolução do software,
  - etc.

# Programas de faculdade

- Requisitos estáveis e bem definidos
- Escopo pequeno (1 a 10 KLOCS)
- Prazos razoáveis
- Equipes pequenas
- Mão de obra gratuita
- Não entra em produção
- Não tem usuário
- Não precisa de manutenção

# Programas do “mundo real”

- Fazer software no “mundo real” deve considerar fatores como:

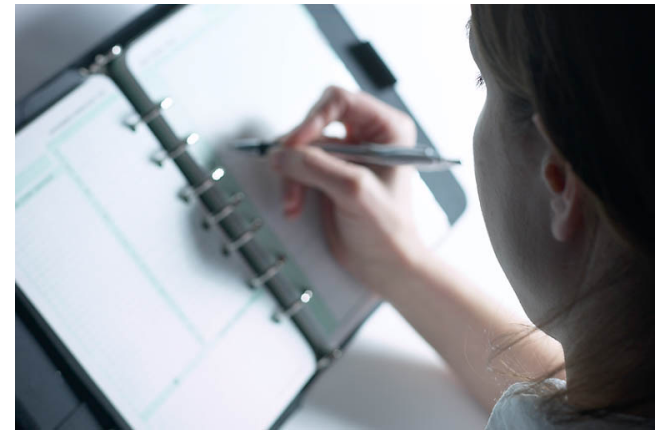
- Escopo
- Custo
- Prazo
- Qualidade



- Em função do tamanho do software, esses fatores se tornam difíceis de garantir!

# Cenário 1: Agenda Pessoal

- Objetivo
  - Guardar o nome e o aniversário de até 50 pessoas
- Quanto custa para fazer?
- Quanto tempo vai levar para ficar pronto?
- Qual a consequência no caso de defeito?



# Cenário 2: Boeing 777

- Objetivo
  - Controlar todo o hardware do Boeing 777
- Quanto custa para fazer?
- Quanto tempo vai levar para ficar pronto?
- Qual a consequência no caso de defeito?





# Cenário 2: Boeing 777

- Tamanho
  - Mais de 4 milhões de linhas de código
  - Linguagem dominante (>99%): Ada
- Documentação
  - De 100 a 10.000 páginas por sub-sistema
  - Total de 79 sub-sistemas integrados
- Duração
  - 4,5 anos de desenvolvimento
- Ampla utilização de Engenharia de Software
- Em operação desde 1995
  - Zero acidentes graves até 2006



<http://www.stsc.hill.af.mil/crosstalk/1996/01/Boein777.asp>

<http://www.boeing.com/news/techissues/pdf/statsum.pdf>

# Mas fazer software não é arte?

- Parte arte, parte engenharia...
  - Se o cantor/ator/pintor errar, a audiência fica chateada
  - Se o engenheiro civil errar o prédio pode cair
  - Se o médico errar o paciente pode morrer
- Se o desenvolvedor de software errar, o que pode acontecer?

# Caso real 1: Aeroporto de Denver

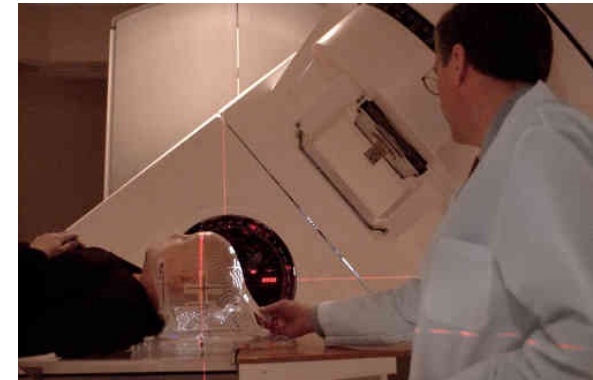
- Sistema de despacho de bagagem do aeroporto de Denver
- Problema:
  - O sistema nunca funcionou adequadamente
- Causa:
  - Arquitetura extremamente complexa
  - Mudança constante nos requisitos
  - Erros de estimativa de custo e prazo
  - Desprezo aos conselhos de especialistas
  - Intolerância a falhas
- Consequências
  - Entrega de algo muito menor do que o planejado, que foi desativado 10 anos depois
  - Atraso de 16 meses para entregar o aeroporto
  - Prejuízo de US\$ 560 milhões e custo mensal de manutenção de US\$ 1 milhão



[http://calleam.com/WTPF/?page\\_id=2086](http://calleam.com/WTPF/?page_id=2086)

# Caso real 2: Therac-25

- Máquina de radioterapia controlada por computador
- Problema:
  - Doses indevidas de radiação emitidas
- Causa:
  - Interface com usuário inapropriada
  - Documentação deficiente
  - Software reutilizado sem ser adaptado para o novo hardware
  - Software de sensores de falha com defeito
- Conseqüências
  - Ao menos 5 mortes entre 1985 e 1987



<http://sunnyday.mit.edu/papers/therac.pdf>

# Caso real 3: Ariane 5

- Foguete lançador de satélites
- Problema:
  - O foguete se auto-destruiu 40 segundos após o lançamento
- Causa:
  - Software reutilizado sem ser adaptado para o novo hardware
  - Ausência de testes deste software em solo
  - Defeito apresentado em voo
- Consequências
  - Prejuízo de mais de US\$ 370 milhões



Dowson, Mark. 1997. The Ariane 5 software failure.  
*SIGSOFT Softw. Eng. Notes* 22, no. 2.

# Motivação extra para estudar?

- Diversos concursos e oportunidades de emprego exigem conhecimento de Engenharia de Software
- Alguns exemplos:



# Como será o curso?

Só os Métodos  
Clássicos prestam!



Só os Métodos  
Ágeis prestam!



# Como será o curso?

- Veremos **sem preconceito** técnicas clássicas e ágeis de Engenharia de Software
- Utilizaremos o que considerarmos melhor para cada situação
- Mas... o processo base que utilizaremos será iterativo, incremental e ágil



# ES na UFF

Atividades  
Gerenciais



Planejamento  
de Projetos

Monitoramento  
e Controle

Melhoria de  
Processos

Gerência  
de Riscos

Atividades  
de Análise e  
Projeto



Levantamento  
de Requisitos

Modelagem



Arquitetura

Projeto

Reutilização

Atividades  
de Apoio



Garantia da  
Qualidade

Medição  
e Análise

Gerência de  
Configuração

Verificação,  
Validação e Testes

# Ementa da disciplina

- Gerência de projetos
- Processo de desenvolvimento de software
- Testes
- Gerenciamento da Qualidade de software
- Métricas
- Gerência de configuração de software
- Reengenharia

# Avaliação

$$Média = \frac{2 \times Prova_1 + 2 \times Prova_2 + Trabalho}{5}$$

# Avaliação

- APROVADO

*Presença  $\geq 75\%$*

**E**

*Média  $\geq 6$*

- VERIFICAÇÃO SUPLEMENTAR

*Presença  $\geq 75\%$*

**E**

*$4 \leq \text{Média} < 6$*

Será aprovado na VS se tirar nota maior ou igual a 6

- REPROVADO

*Presença  $< 75\%$*

**OU**

*Média  $< 4$*

# Trabalho

- Fazer um **jogo Rummikub** onde seja possível jogar contra o computador, usando as técnicas estudadas durante o curso.
- Se enxerguem como uma pequena *software house*
  - Grupo de 5 participantes
- Será avaliado tanto o produto quanto como esse produto foi desenvolvido (processos e técnicas aplicados)

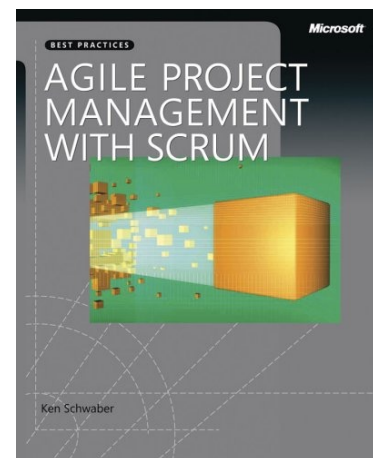
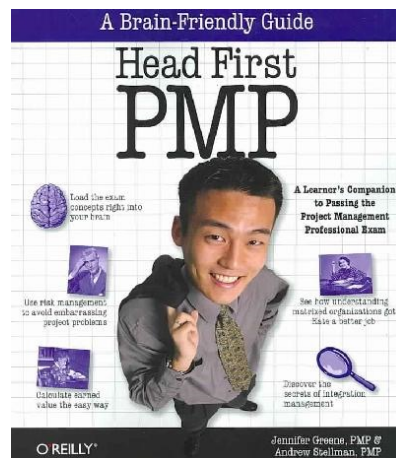
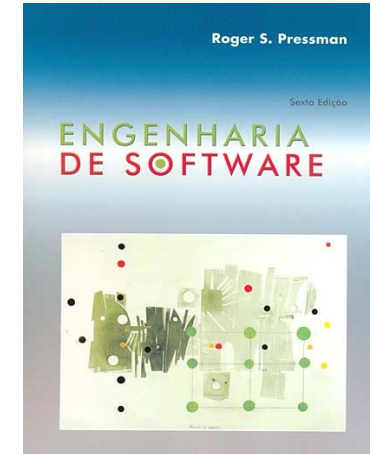
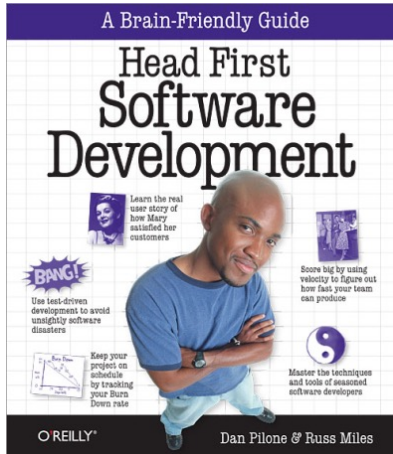
# Trabalho

- Três apresentações serão feitas durante o curso
- 1ª apresentação e entrega
  - Escopo do produto
  - Escopo do projeto
  - Estimativas de esforço e custo
  - Orçamento
  - Cronograma
  - Análise de riscos
  - Monitoramento e controle
  - Versão parcial do produto
- 2ª apresentação e entrega
  - Ferramenta de controle de versões e de controle de modificações
  - Estratégia de ramificação
  - Conteúdo do repositório
  - Monitoramento e controle
  - Versão parcial do produto
- 3ª apresentação e entrega
  - Testes de unidade, integração, sistema e aceitação
  - Casos de teste e resultados da sua execução
  - Monitoramento e controle
  - Versão final do produto

# Listas de Exercício


- Devem ser feitas individualmente
- Entregar no Google Classroom até a última aula antes da Prova 1 (listas 1 a 3) e da Prova 2 (listas 4 a 6)
- Valerão até 0,5 pontos na média para alunos com média entre 5,5 e 6,0, eventualmente arredondando a média para 6,0
- Não serão aceitas entregas fora do prazo

# Bibliografia do curso





# Página do curso



**Instituto de computação**

**Leonardo Gresta Paulino Murta**  
 Assistant Professor (Professor Adjunto IV), IC/UFF  
 D.Sc., COPPE/UF RJ, 2006  
 M.Sc., COPPE/UF RJ, 2002  
 B.Sc., IM/UF RJ, 1999

**Engenharia de Software II**

**Logística**

Disciplina: TCC00181 - Engenharia de Software II  
 Data: quartas e sextas, de 11:00 às 13:00  
 Sala: 321  
 Grupo: <https://www.facebook.com/groups/es2-2015.2> (todos os alunos devem se inscrever no grupo)

**Avaliação**

$Média = (2 \times Prova 1 + 2 \times Prova 2 + Trabalho) / 5$

**Trabalho**

O curso terá um trabalho feito em grupos de 5 participantes, que devem ser constituídos preferencialmente na primeira semana de aula e ter a mesma formação até o final do curso.

Cada grupo deve se envergar como uma pequena software house. O tema do trabalho é fazer um sistema de caixa bancário eletrônico. O sistema deve controlar usuários e contas, permitindo as operações de depósito, saque, pagamento, impressão e compensação de cheques, saldo, extrato, investimento e transferência.

O trabalho consiste em, durante o decorrer do curso, utilizar as técnicas aprendidas para gerenciar o projeto e desenvolver o produto. Será avaliado o produto final e como esse produto foi desenvolvido (processos e técnicas aplicados).

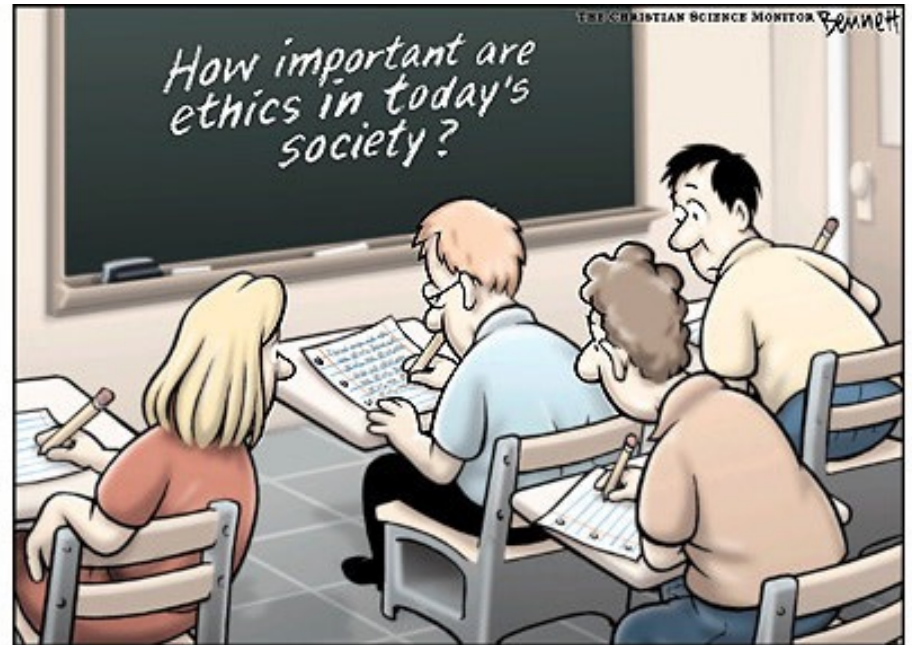
Foram reservados 3 momentos para apresentações de 15 a 20 minutos por grupo e entregas parciais do trabalho. O grupo poderá utilizar slides e quadro para as apresentações e as entregas devem ser feitas em papel, no momento das apresentações. A primeira apresentação e entrega parcial deve conter o escopo do projeto e do produto, as estimativas de esforço e custo, o orçamento, o cronograma de desenvolvimento, a análise de riscos e o monitoramento e controle do projeto. O grupo também deve mostrar a versão parcial do produto feita até então. A segunda apresentação e entrega parcial deve conter as ferramentas e técnicas de gerência de configuração adotadas no projeto, assim como o conteúdo do repositório e o monitoramento e controle do projeto, além de mostrar a versão parcial do produto feita até então. A terceira apresentação e entrega parcial deve conter as ferramentas e técnicas de testes usadas, assim como os casos de teste e resultados da sua execução, dados de monitoramento e controle do projeto e versão final do produto. Além disso, nessa última apresentação e entrega também é esperado um detalhamento das dificuldades encontradas e decisões tomadas, e informações de como

<http://www.ic.uff.br/~leomurta>  
 (no final da página tem o cronograma, com **datas** e **slides**)

**Importante:** cadastrem-se no Google Classroom (código informado na chamada)!

# Fair Play!

- Não colar ou dar cola em provas
- Não plagiar o trabalho
- Não trapacear nas leituras e listas de exercício
- Não sobrecarregar os colegas do grupo
- Não assinar presença por colegas
- Dar crédito apropriado quando usar trabalhos de terceiros



<http://www.claybennett.com/pages/ethics.html>

# Apresentação da Disciplina de Engenharia de Software II

Leonardo Gresta Paulino Murta  
leomurta@ic.uff.br