Engenharia de Software II

Aula 21

http://www.ic.uff.br/~bianca/engsoft2/

Ementa

- Processos de desenvolvimento de software
- Estratégias e técnicas de teste de software
- Métricas para software
- **Gestão de projetos de software**
  - **Conceitos (Cap. 21)**
  - **Métricas (Cap. 22 – Seções 22.1 e 22.2)**
  - **Estimativas (Cap. 23 – Seções 23.1 a 23.7)**
  - **Cronogramação (Cap. 24)**
  - **Gestão de risco**
  - **Gestão de qualidade**
  - **Gestão de modificações**
- Reengenharia e engenharia reversa

Modelos de Estimativa Empíricos

- Dados certos valores estimados para o tamanho do software (LOC ou FP), os modelos estimam o esforço.
- Os parâmetros dos modelos são estimados com base em projetos anteriores.
  - A amostra de projetos é limitada.
  - O modelo só é válido para aquela classe de projeto.
- Estrutura geral:

  $E = a + b \times t^c$

  onde E é o esforço em pessoa-mês e t é a variável de de estimativa (LOC ou FP).

Exemplos de Modelos

- Modelo de Walston-Felix

  $E = 5{,}2 \times (KLOC)^{0{,}91}$

- Modelo de Bailey-Basili

  $E = 5{,}5 + 0{,}73 \times (KLOC)^{1{,}16}$

- Modelo de Albrecht e Gaffney

  $E = -91{,}4 + 0{,}355 \times FP$

- Modelo de Kemerer

  $E = -37 + 0{,}96 \times FP$

Modelo COCOMO

- Nos anos 1980, Barry Boehm propôs uma hierarquia de modelos de estimativa de software denominada COCOMO (Constructive Cost Model).
  - O modelo COCOMO tornou-se um dos modelos de estimativa de custo de software mais amplamente usados.
- Mais tarde, o modelo COCOMO evoluiu para um modelo mais abrangente, chamado COCOMO II.

Modelo COCOMO II

- É uma hierarquia de modelos que tratam das seguintes áreas:
  - Modelo de composição da aplicação
    - Usado nos primeiros estágios da engenharia de software
  - Modelo do primeiro estágio de projeto
    - Usado depois que os requisitos foram estabilizados e a arquitetura foi estabelecida.
  - Modelo para o estágio após a arquitetura
    - Usado durante a construção do software.

Modelo COCOMO II

- Como todos os modelos de estimativa de software, requer informação de tamanho.
  - Três opções:
    - Pontos por objeto
    - Pontos por função
    - Linhas de código-fonte
- Métrica pontos por objeto
  - É calculada usando-se a contagem da quantidade de:
    - Telas (na interface com o usuário)
    - Relatórios
    - Componentes que serão necessários pra construir a aplicação.
  - Cada objeto é classificado em três níveis de complexidade.

Modelo COCOMO II

- A porcentagem de re-uso é estimada e a contagem de pontos por objeto é ajustada.
- A produtividade é estimada usando-se:
  - A experiência/capacidade do desenvolvedor.
  - A maturidade/capacidade do ambiente de desenvolvimento.
- O esforço é estimado usando-se os pontos por objeto e a produtividade.

  E = NOP/PROD

Modelo Equação de Software

- Foi derivado de dados de produtividade coletados em mais de 4 mil projetos de software contemporâneo.
- O esforço é dado por:

  $E = [LOC \times B^{0,333}/P]^3 \times (1/t^4)$

  onde:

  E = esforço em pessoas-mês ou pessoa-ano

  t = duração do projeto em meses ou anos

  B = fator de aptidões especiais

  P = parâmetro de produtividade
- P pode ser derivado usando dados históricos.
- B aumenta lentamente à medida que cresce a necessidade de integração, teste, garantia de qualidade, etc.

Ementa

- Processos de desenvolvimento de software
- Estratégias e técnicas de teste de software
- Métricas para software
- **Gestão de projetos de software**
  - **Conceitos (Cap. 21)**
  - **Métricas (Cap. 22 – Seções 22.1 e 22.2)**
  - **Estimativas (Cap. 23 – Seções 23.1 a 23.7)**
  - **Cronogramação (Cap. 24)**
  - **Gestão de risco**
  - **Gestão de qualidade**
  - **Gestão de modificações**
- Reengenharia e engenharia reversa

Cronogramação de Projeto de Software

- Datas de entrega impraticáveis são o cotidiano nos projetos de software.
  - Atrasos são muito freqüentes.
- Muitas vezes as datas são estabelecidas por alguém de fora do grupo de engenharia de software.
  - Se as estimativas indicam que a data de entrega é impraticável, um gerente competente deve proteger sua equipe da pressão do cronograma.
  - Uma solução é usar um modelo incremental e só garantir alguma funcionalidade no primeiro incremento.

Cronogramação de Projeto de Software

- Um projeto técnico envolve centenas de pequenas tarefas que devem ser executadas para conseguir uma meta maior.
  - Algumas situam-se fora do fluxo principal e não tem grande impacto na data de finalização do projeto.
  - Outras situam-se no caminho “crítico”.

Cronogramação de Projeto de Software

- O objetivo do gerente é:
  - Definir todas as tarefas do projeto.
  - Construir uma rede que mostre suas interdependências.
  - Identificar as tarefas que são críticas nessa rede.
  - Acompanhar o progresso das tarefas para certificar-se de que atrasos sejam reconhecidos logo.
    - Para conseguir isso, o gerente precisa elaborar um cronograma.

Cronogramação de Projeto de Software

- A cronogramação do projeto de software é uma atividade que distribui o esforço estimado pela duração planejada do projeto.
- O cronograma evolui com o tempo.
  - Durante os primeiros estágios de planejamento, um cronograma macroscópico é desenvolvido.
    - Identifica as principais atividades de arcabouço e as funções do produto a que se aplicam.
  - À medida que o projeto evolui, cada entrada no cronograma macroscópico é refinada em um cronograma detalhado.
    - Tarefas de software específicas são cronogramadas.

Princípios Básicos da Cronogramação

- Compartimentalização
  - O projeto deve ser decomposto em um certo número de tarefas e atividades gerenciáveis.
- Interdependência
  - A interdependência entre cada atividade, ação ou tarefa deve ser determinada.
  - Algumas atividades devem ocorrer em seqüência, outras podem ocorrer em paralelo.
- Atribuição de tempo
  - A cada tarefa deve ser atribuído um certo número de unidades de trabalho (ex: pessoa-dia).
  - A cada tarefa devem ser atribuídas datas de início e término, que são funções das interdependências e do trabalho.

Princípios Básicos da Cronogramação

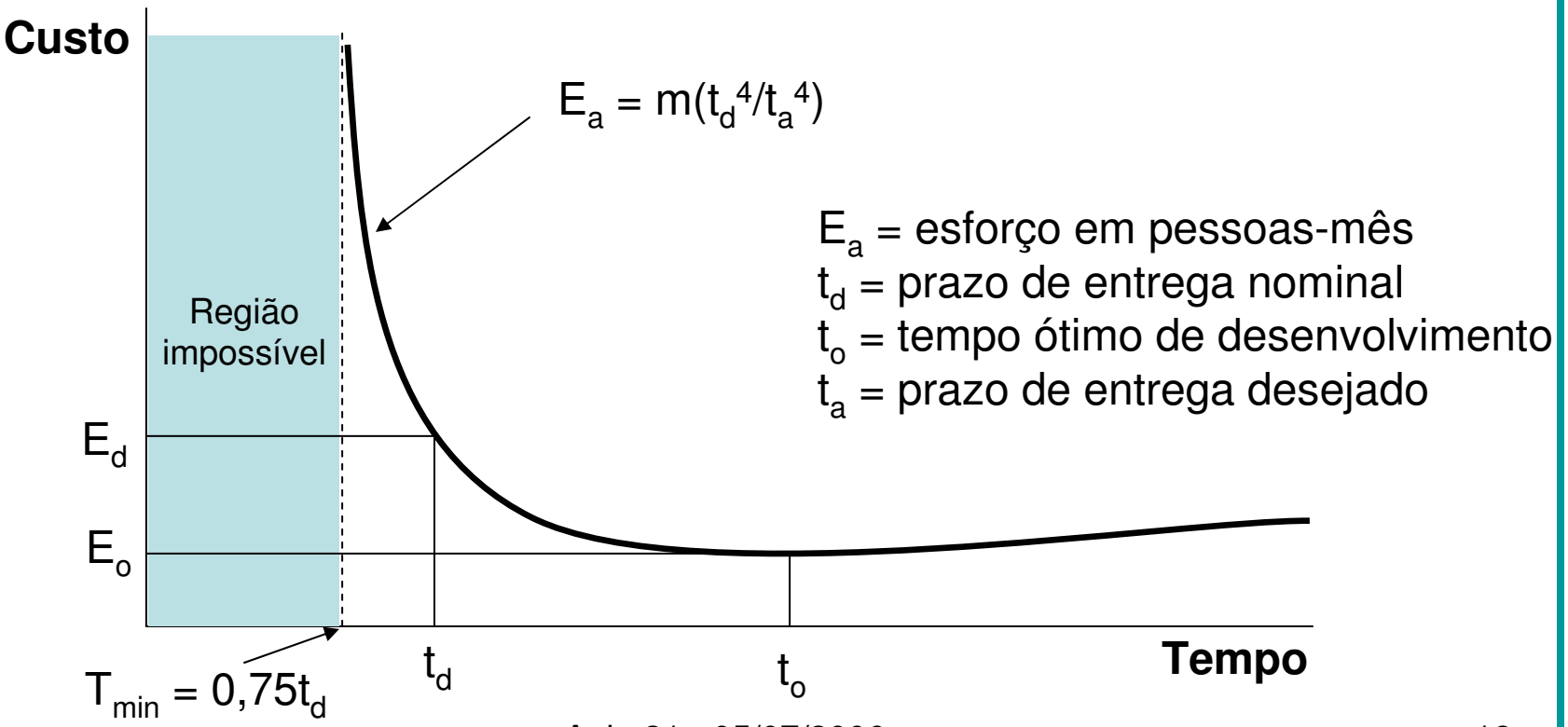
- Validação do esforço
  - O gerente de projeto deve garantir que não mais do que o número disponível de pessoas seja cronogramado em um determinado momento.
- Responsabilidades definidas
  - Cada tarefa deve ser atribuída a um membro específico da equipe.
- Resultados definidos
  - Cada tarefa deve ter um resultado definido, normalmente um produto de trabalho.
- Marcos de referência definidos
  - Um marco de referência é atingido quando um ou mais produtos do trabalho tiverem sido revisados quanto à qualidade.

Relacionamento entre Pessoal e Esforço

- Mito comum: “se ficarmos atrasados podemos sempre agregar mais programadores e recuperar o atraso”.
  - Normalmente isso causa atrasos ainda maiores.
    - Quem treina os novos programadores são os que estavam fazendo o trabalho.
    - Aumenta a complexidade de comunicação.

Relacionamento entre Pessoal e Esforço

- Curva Putnam-Norden-Rayleigh



Distribuição de Esforço

- Regra 40-20-40
  - 40% do esforço é reservado para a análise e projeto iniciais.
  - 20% do esforço é reservado para a codificação.
  - 40% do esforço é reservado para o teste final.
- Essa distribuição deve ser usada apenas como diretriz.
  - Alguns acreditam que mais de 40% do esforço deve ser gasto na análise e projeto.
  - Proponentes do desenvolvimento ágil alegam que a maior parte do esforço deve ser usada na codificação.