Gerenciamento de Projeto de Software
Métricas e estimativas

Profa. Flávia B. Blum Haddad
Email: flaviahaddad@utfpr.edu.br

Motivação

- Um dos objetivos básicos da Engenharia de Software é transformar o desenvolvimento *de sistemas de software, partindo de uma abordagem artística e "indisciplinada", para alcançar um processo controlado, quantificado e previsível.*

"Não se pode gerenciar o que não se pode medir".

Tom De Marco

Estimativas – questões

- Quanto esforço é necessário para completar cada atividade?
- Quanto tempo de calendário é necessário para completar cada atividade?
- Qual é o custo total de cada atividade?

O que são métricas de software?

- **Uma métrica é a medição de um atributo (propriedades ou características) de uma determinada entidade (produto, processo ou recursos). Exemplos:**
  - **Tamanho do produto de software (ex: Número de Linhas de Código – LOC)**
  - **Número de pessoas necessárias para preparar a especificação em UML de uma aplicação**
  - **Número de defeitos encontrados no documento de requisitos do software**

Por que medir um software?

- Entender e aperfeiçoar o processo de desenvolvimento
- Melhorar a gerência de projetos e o relacionamento com clientes
- Reduzir frustrações e pressões de cronograma
- Gerenciar contratos de software
- Indicar a qualidade de um produto de software

Propriedades desejáveis de uma métrica

- Facilmente calculada, entendida e testada
- Passível de estudos estatísticos
- Expressa em alguma unidade (tempo, pessoas, $)
- Obtida o mais cedo possível no ciclo de vida do software
- Passível de automação

Propriedades desejáveis de uma métrica

- **Repetível e independente do observador**
- **Comprometida com uma estratégia de melhoria**

Categorização das métricas

- **Métricas diretas (fundamentais ou básicas)**
  - **Medidas realizadas em termos de atributos observados (usualmente determinadas pela contagem)**
  - **Ex.: custo, esforço, número de linhas de código, número de páginas, número de diagramas, etc.**

Categorização das métricas

- **Métricas indiretas (derivadas)**
  - **Medidas obtidas a partir de outras métricas**
  - **Ex.: complexidade, eficiência, confiabilidade, facilidade de manutenção**

    → **uso de fórmulas**

Categorização das métricas

- **Métricas orientadas a tamanho**
  - **São medidas diretas do tamanho dos artefatos de software associados ao processo por meio do qual o software é desenvolvido.**
  - **Ex.: custo da implementação, número de linhas de código (LOC, KLOC) número de páginas de documentação, número de defeitos em uma especificação de requisitos, etc.**

Categorização das métricas

- **Métricas orientadas por função**
  - **Consiste em um método para medição de software que indica a complexidade do software.**
  - **Ex.: SOD (Speed of Delivery) - medida do número de funções desenvolvidas/entregues em um determinado período de tempo (mês), utilizando a equipe disponível.**

Categorização das métricas

- **Métricas de produtividade**
  - **Concentram-se na saída/resultado do processo de engenharia de software.**
  - **Ex.: PDR (Project Delivery Rate) - horas necessárias para desenvolver parte do sistema (função)**
- **Métricas de qualidade**
  - **Oferecem uma indicação de quanto o software se adapta às exigências do cliente**
  - **Ex.: defeitos por artefato**

Categorização das métricas

- **Métricas técnicas**
  - **Concentram-se nas características do produto e não no processo de desenvolvimento**
  - **Ex.: complexidade lógica e grau de manutenibilidade**

Estimativa de software – objetivos

1. Estimar o tamanho do produto
2. Estimar o esforço
3. Estimar o prazo
4. Fornecer estimativas dentro de uma faixa permitida e refinar essa faixa à medida que o projeto progride

Tipos de estimativas

- **Tamanho**
  - **Quantidade de software a ser produzida**
  - **Ex. número de linhas de código, número de requisitos funcionais, número de casos de uso**
- **Esforço**
  - **Derivado da estimativa de tamanho**
  - **Ex. estimativa de tamanho X produtividade**

    (qtde diagramas X tpo gasto para 1 analista preparar 1 diagrama = esforço (dias)

Tipos de estimativas

- **Prazo**
  - **Geralmente é baseado em datas requisitadas pelo cliente**
- **Qualidade**
  - **Medidas de resultados**
  - **Ex. defeitos por fase, esforço de mudanças**

Linhas de código (LOC)

- Fácil de contar quando os programas eram escritos em cartões perfurados.
- Em linguagens de alto nível, como Java, por possuírem declarações, comandos executáveis e comentários, não há uma relação simples entre as declarações de programa e as linhas de uma listagem.

Linhas de código (LOC)

Tabela 26.2 Tempos de desenvolvimento de sistema

	Análise	Projeto	Codificação	Testes	Documentação
Código Assembly	3 semanas	5 semanas	8 semanas	10 semanas	2 semanas
Linguagem de alto nível	3 semanas	5 semanas	4 semanas	6 semanas	2 semanas

	Tamanho	Esforço	Produtividade
Código Assembly	5.000 linhas	28 semanas	714 linhas/mês
Linguagem de alto nível	1.500 linhas	20 semanas	300 linhas/mês

Fonte: Engenharia de Software
Ian Sommerville

Linhas de código (LOC)

- Cada projeto pode obter as seguintes métricas a partir das linhas de código:
  - Erros por KLOC (K = milhares)
  - Defeitos por KLOC
  - $ por LOC
  - Páginas de documentação por KLOC
  - Erros por pessoa/mês
  - LOC por pessoa/mês
  - $ por pagina de documentação

Linhas de código (LOC)

Projeto	LOC	Esforço (mês)	$ (Mil)	Pág. Doc	Erros	Defeitos	Pessoas
Alfa	12.100	24	168	365	134	29	3
Beta	27.200	62	440	1.224	321	86	5
Gama	20.200	43	314	1.050	256	64	6
...	...	...	...	...	...	...	...

Linhas de código (LOC)

- Não é muito utilizada como parâmetro de estimativa, pois requer um nível de detalhes difícil de alcançar:
  - Padronização na forma de codificar
  - Manter apenas uma linguagem de programação
  - Não diferencia linhas de código geradas por ferramentas CASE daquelas criadas manualmente

Pontos de função

- A produtividade é expressa com o número de pontos de função implementados por pessoa/mês.
- Um ponto de função não é uma característica única, porém é calculado por meio da combinação de várias medições ou estimativas.

Pontos de função

- Baseado em uma combinação de características de programa:

- Entradas e saídas externas;
- Interações de usuários;
- Interfaces externas;
- Arquivos usados pelo sistema.

Pontos de função

- Um peso é associado com cada uma dessas características e a contagem de ponto de função é obtida pela multiplicação de cada contagem inicial pelo peso estimando e somando todos os valores.

UFC = Σ (número de elementos de dado tipo) x (peso)

*UFC (unadjusted function-point count)

Mais sobre Pontos por Função

Moodle (arquivo)

Produtividade de software

- É uma medida da taxa na qual os engenheiros individuais envolvidos no desenvolvimento de software produzem o software e sua documentação associada.
- Essencialmente, queremos medir a funcionalidade útil produzida por unidade de tempo.

Estimativa de produtividade

- Sistemas de tempo real embutidos, de 40 a 160 LOC/pm
- Programas de sistemas, de 150 a 400 LOC/pm
- Aplicações comerciais, de 200 a 900 LOC/pm

Fatores que afetam a produtividade

Fator	Descrição
Experiência no domínio da aplicação	O conhecimento do domínio da aplicação é essencial para o desenvolvimento eficiente de software. Os engenheiros que já compreendem um domínio serão, provavelmente, os mais produtivos.
Qualidade de processo	O processo de desenvolvimento usado pode ter efeito significativo sobre a produtividade. Esse tópico é abordado no Capítulo 28.
Tamanho de projeto	Quanto maior o projeto, mais tempo é necessário para a comunicação da equipe. Se menos tempo estiver disponível para o desenvolvimento, a produtividade individual será reduzida.
Apoio de tecnologia	Uma boa tecnologia de apoio, como ferramentas CASE e sistemas de gerenciamento de configurações, pode aumentar a produtividade.
Ambiente de trabalho	Conforme explicado no Capítulo 25, um ambiente de trabalho calmo com áreas de trabalho privativas contribui para o aumento da produtividade.

Fonte: Engenharia de Software
Ian Sommerville

O modelo COCOMO

- É um modelo empírico baseado na experiência de projeto.
- É um modelo bem documentado e independente, que não é ligado a um fornecedor específico de software.
- Uma longa história desde a versão inicial, publicada em 1981 (COCOMO 81) até o COCOMO II.

Modelo COCOMO II

- O COCOMO II incorpora uma gama de submodelos que produzem estimativas de software cada vez mais detalhadas.

- Os submodelos do COCOMO II são:
- Modelo de composição de aplicação: usado quando o software é composto de partes existentes;
- Modelo de projetos preliminar: usado quando os requisitos estão disponíveis , mas o projeto ainda não foi iniciado;

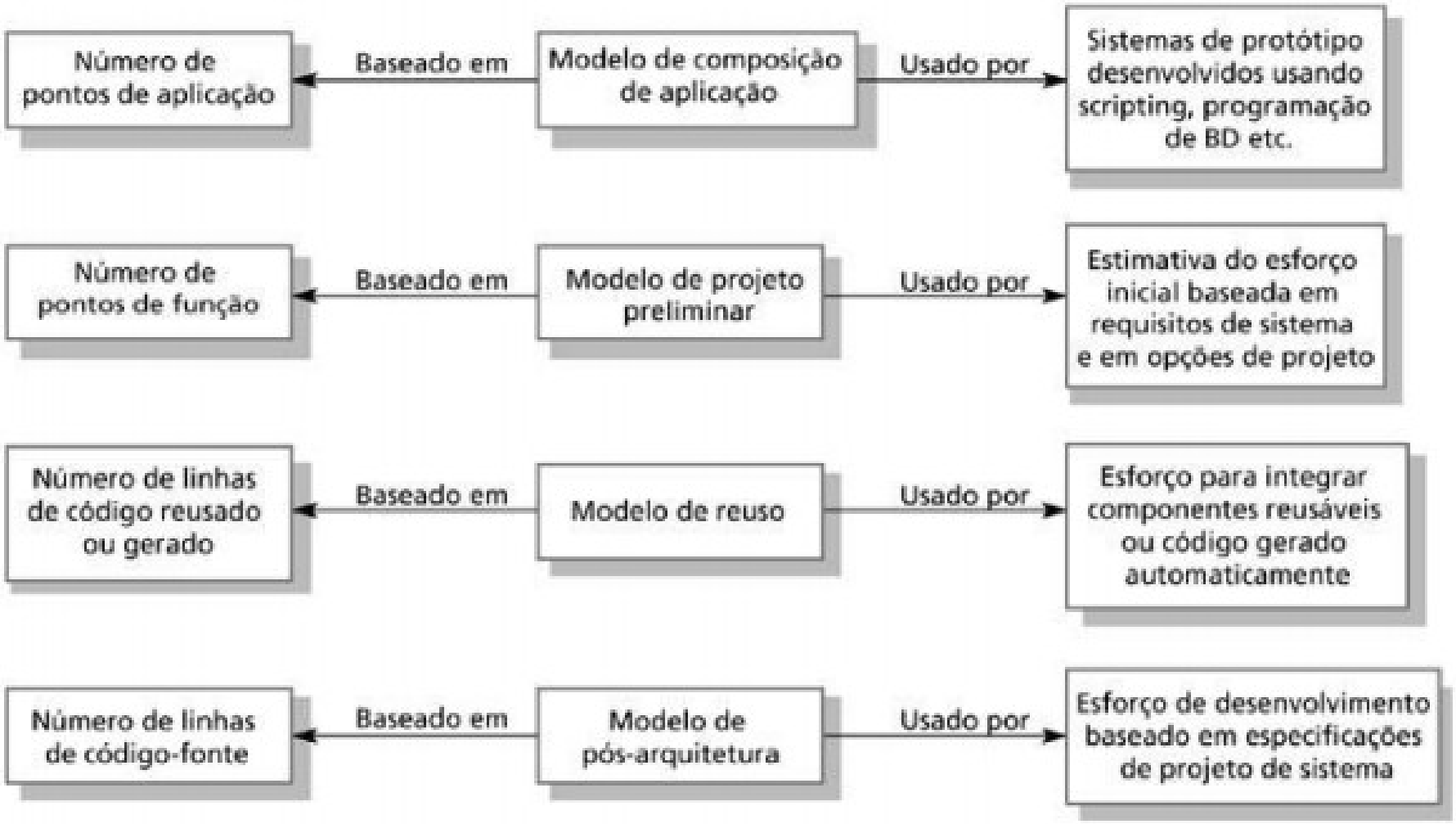
Modelo COCOMO II

- Os submodelos do COCOMO II são:

- Modelo de reuso: usado para calcular o esforço de integração de componentes reusáveis;

- Modelo de pós-arquitetura: usado quando a arquitetura de sistema foi projetada e mais informações sobre o sistema estão disponíveis.

Submodelos COCOMO II



Fonte: Engenharia de Software
Ian Sommerville