Quantificação de Reuso

Eduardo Figueiredo
http://www.dcc.ufmg.br/~figueiredo
reuso.software@gmail.com

28 Maio 2012

Disciplina 2012-2

- Medição e Qualidade de Software
- Terças e quintas: 9:25 as 11:05
- Ofertada para alunos de Pós-Graduação (PPGCC e Isolada)
  - Aberta para alunos de graduação em Ciência da Computação e Sistemas de Informação

Tópicos da Disciplina

- Medição e Qualidade de Software
  - Qualidade do produto e do processo
  - Refatoração e *bad smells*
  - Métricas de software
  - Métodos quantitativos para detecção de *bad smells*
  - Avaliação da qualidade de software
  - Planejamento de experimentos e análise dos resultados

Método de Avaliação

- Alunos de Graduação ou Pós
  - Prova 1 (P1): 20 pts
  - Prova 2 (P2): 20 pts
  - Seminário: 10 pts
  - Exercícios e participação: 10 pts.
  - Estudo experimental (TP): 40 pts
  - Prova substitutiva: 20 pts
    - Substitui a nota de P1 ou P2

Método de Avaliação

- Alternativa para Alunos de Graduação
  - Prova 1 (P1): 40 pts
  - Prova 2 (P2): 40 pts
  - Seminário: 10 pts
  - Exercícios e participação: 10 pts.
  - Prova substitutiva: 40 pts.
    - Substitui a nota de P1 ou P2

Questionário

- Favor preencher o questionário com informações sobre você
  - **Não** será avaliado
  - **Não** vou usar estas informações para aumentar ou abaixar sua nota
- Os dados serão usadas de forma anônima e geral
  - **Não** vou usar a informação de forma individualizada

Agenda da Aula

- Medições de Software
- Métricas de Produto
  - Métricas tradicionais
  - Métricas orientadas a objetos
- Métricas de Tamanho
- Métricas e Modelos para Reuso

Por que medir?

- Medição de atributos de software é essencial

“Somente podemos controlar o que podemos medir” DeMarco

"In God we trust; all others must bring data." W. Edwards Deming

Medições de Software

Métrica de software

- Medições se dedicam a obter um ou mais valores numéricos para um atributo de qualidade
  - Comparando os números, é possível tirar conclusões sobre a qualidade do produto
- Uma métrica de software é qualquer medição que se refere ao sistema
  - Medições de tamanho (exemplo, LOC)
  - Número de defeitos relatados, etc.

Uso de Medições

- Medições de software podem ser usadas de duas maneiras
  - Avaliar a qualidade do sistema e fazer previsões gerais sobre ele (exemplo, número de defeitos)
  - Para identificar partes (ou módulos) problemáticos

Adoção pela Indústria

- Muitas empresas ainda não usam medições sistemáticas para avaliar a qualidade
- Algumas razões
  - Os processos das empresas não são maduros o suficiente
  - A ausência de métricas padronizadas
  - Limitado apoio de ferramentas de medição

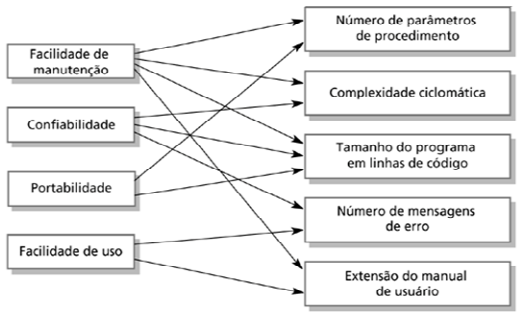
Problemas com Medições

- Geralmente é impossível medir um atributo de qualidade diretamente
  - Atributos de qualidade são fatores externos ao software
  - Métricas medem fatores internos
- Exemplos de atributos de qualidade
  - Facilidade de manutenção
  - Facilidade de uso
  - Confiabilidade

Solução: Modelos de Qualidade

- A solução é medir atributos que podem estar relacionados aos atributos de qualidade
- Deve haver um relacionamento claro e válido entre atributos de qualidade e atributos internos

Um Modelo de Qualidade



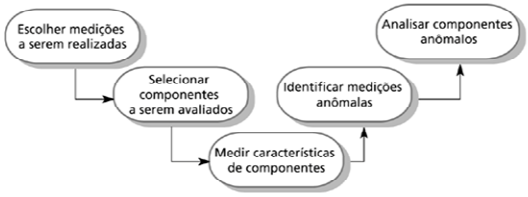
Validade dos Modelos

- Três condições devem ser verificadas em modelos de qualidade
  - O atributo interno deve ser precisamente medido
  - Deve haver relacionamentos entre o que podemos medir e o que queremos saber
  - Os relacionamentos são compreendidos e válidos

O Processo de Medição

- O processo de medição deve fazer parte do processo de controle da qualidade
  - Utilizam dados históricos de projetos anteriores
- As atividades do processo
  - Escolher medições a serem realizadas
  - Selecionar componentes a serem avaliados
  - Medir características dos componentes
  - Identificar medições anômalas
  - Analisar componentes anômalos

Modelo do Processo



Escolher Medições

- Uma abordagem para escolher as medições é o GQM
  - *Goal-Question-Metric*
- As questões são formuladas para atender um objetivo
- As métricas são escolhidas para responderem as questões

O Modelo de Medição GQM

- Objetivos
  - Definem o que a organização quer melhorar (exemplo: produtividade)
- Questões
  - Refinamento dos objetivos em áreas de incertezas (exemplo: linhas de código produzidas podem ser aumentadas?)
- Métricas
  - Medições necessárias para responder as questões (exemplo: LOC por desenvolvedor)

Selecionar Componentes

- Pode não ser necessário (ou desejável) medir todo o sistema
- Estratégias de escolha
  - Escolher um subconjunto representativo de todos os componentes
  - Escolher os componentes particularmente críticos na aplicação

Medir os Atributos de Qualidade

- Os componentes selecionados são medidos
- As medidas são associadas aos atributos de qualidade
  - Geralmente envolve uma representação dos componentes
- Ferramentas de medição podem estar incorporadas a outras ferramentas (ou ambientes) de desenvolvimento

Analisar Medições

- Uma vez feita as medições, é preciso compará-las a medições anteriores
  - Dados históricos são utilizados
- A análise deve procurar valores incomuns
  - Ou seja, valores muito altos ou muito baixos para cada métrica

Analisar Componentes

- Se um componente tem valores anômalos, este deve ser examinado
  - A inspeção é responsável por decidir se existe (ou não) problema no componente
- Um valor incomum para um componente não necessariamente significa que o componente tem baixa qualidade

Métricas de Produto

Métricas de Produto

- Quantificam atributos internos do software
- Exemplos de atributos
  - Tamanho
  - Acoplamento dentre componentes
  - Coesão de um componente, etc.

Tipos de Métricas

- Métricas Dinâmicas
  - São coletadas por medições realizadas durante a execução do programa
- Métricas Estáticas
  - São coletadas por medições realizadas na documentação de projeto ou código fonte do programa

Dinâmicas x Estática

- Métricas dinâmicas ajudam a avaliar atributos de qualidade como eficiência e confiabilidade
  - São medidas após o sistema ter sido implementado
- Métricas estáticas ajudam a avaliar atributos como complexidade e facilidade de manutenção
  - Podem ser medidas na fase de projeto

Algumas Métricas Estáticas

- Fan-in / Fan-out
- Tamanho do código
- Complexidade Ciclomática
- Tamanho do Vocabulário
- Profundidade de Aninhamento

Fan-in e Fan-out

- Fan-in
  - Conta o número de funções que chama uma determinada função
  - Valor alto significa grande impacto em mudanças (propagação)
- Fan-out
  - Conta o número de funções chamadas pela função
  - Valor algo significa grande complexidade da função

Tamanho e Complexidade

- Tamanho
  - Tamanho tem se mostrado como métricas mais confiáveis e úteis
  - Em geral, quanto maior o componente, mais complexo e difícil de reusar
- Complexidade Ciclomática
  - Mede a complexidade de controle do programa (*if*, *while*, *for*, etc.)
  - Está relacionada a facilidade de compreensão e de reuso

Vocabulário e Aninhamento

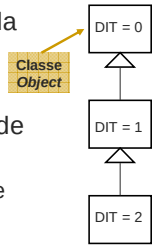
- Tamanho do Vocabulário
  - Conta a quantidade de identificadores (exemplo, nome de classes) do programa
  - Mais identificadores podem significar que eles são mais significativos
- Profundidade de Aninhamento
  - Conta estruturas internas como *if* e *while* aninhados
  - Estruturas aninhadas são mais difíceis de se compreender e de se reusar

Métricas de Programas OO

- Métricas de Chidamber-Kemerer (CK)
  - Métodos Ponderados por Classes (WMC)
  - Profundidade da Herança (DIT)
  - Número de Filhos (NOC)
  - Acoplamento entre Objetos (CBO)
  - Falta de Coesão em Métodos (LCOM)
- Número de Operações Sobreescritas

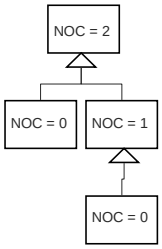
Profundidade de Herança (DIT)

- Representam o número de níveis que uma classe herda métodos e atributos
- Quanto maior a profundidade
  - Mais complexo o projeto
  - Mais difícil de se entender e reusar um módulo



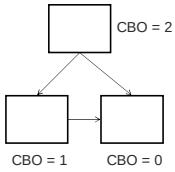
Número de Filhos (NOC)

- Conta o número de subclasses diretas
  - Mede a largura da hierarquia de uma classe
- Valor alto, pode indicar maior reuso



Acoplamento entre Objetos (CBO)

- Semelhante a Fan-out
  - Conta classes chamadas por uma classe
- Quanto mais acoplado uma classe
  - Mais difícil de entender, manter e reusar



Falta de Coesão (LCOM)

LCOM = PMN - PMC se PMN > PMC

LCOM = 0 caso contrário

- Tal que:
  - PMC = Pares de método que acessam algum atributo em comum
  - PMN = Pares de método que não acessam atributo em comum

Métricas para Métodos

- Métodos Ponderados por Classes (WMC)
  - Atribui pesos aos métodos de uma classe
  - Uma forma é "pesar" por linhas de código
  - Valores altos indicam complexidade
- Número de Operações Sobrescritas
  - Conta as operações de uma classe que são sobrescritas por subclasses
  - Valores altos indicam problema na hierarquia de herança

Análise de Medições

- Nem sempre é óbvio o que os dados significam
  - Entender uma grande quantidade de números é muito difícil
- Estatísticos devem ser consultados, se estiverem disponíveis
- A análise de dados deve levar em conta as circunstâncias locais

Exemplo (Arquivo XLS)

- Medição de várias versões
  - Métricas estáticas coletadas em várias versões do sistema Health Watcher
- Métricas
  - CBO: Coupling Between Objects
  - LCOM: Lack of Cohesion over Methods
  - DIT: Depth of Inheritance Tree
  - NOC: Number of Children
  - LOC: Lines of Code, etc.

Slides in English

Measuring Size for Reuse

N. E. Fenton, and S. L. Pfleeger, Software Metrics: A Rigorous and Practical Approach, 2nd ed. Thomson, 1996.

Measuring Size

- Size is the most obvious software attribute which can be measured statically
- Aspects of size
  - Length: physical size
  - Functionality: what users get
  - Complexity: problem solved

How hard is measuring size?

- Each software artifact is a physical entity
  - It can be described in terms of size
- Measuring size should be simple and straightforward
  - And consistent with measurement theory
- In practice, measuring size (software) presents great challenges

Example: LOC is not Simple

- Measuring Lines of Code (LOC) is not fully consistent
  - Some lines are more difficult to code than others
  - One solution could be give weight to lines that have more “stuff”
- However, this problem also occurs with most metrics

Analysing Metrics Together

- Let’s say human size is measured in terms of only height
  - Based only on height, we cannot determine whether a person is obese
- We need two size metrics, height and weight (+ empirical understanding)

Size and Reuse

- Reuse is the extent to which the software is new
  - Size of the reused product
- Extent of Reuse
  - Reused Verbatim (reused as is)
  - Slightly modified (< 25% LOC)
  - Extensively Modified (> 25% LOC)
  - New (nothing comes from a previous code)

Slides in English

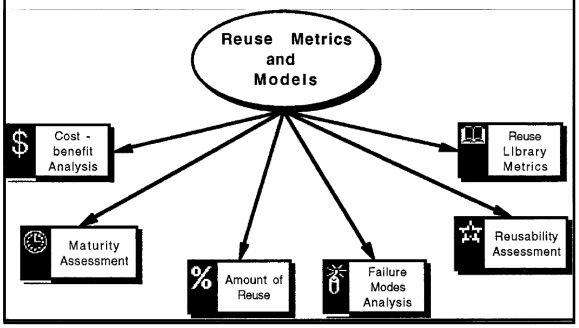
Software Reuse: Metrics and Models

W. Frakes and C. Terry. Software Reuse: Metrics and Models. ACM Computing Surveys, volume 28, issue 2, pp. 415-435, 1996.

Metric and Models

- Organizations implementing systematic reuse must be able to measure
  - Quantify their progress
  - Identify the most effective reuse strategies
- A metric is a quantitative indicator of na attribute of a thing (software, process,...)
- A model specifies relationships among metrics (and external attributes)

Reuse Metrics and Models



Reuse Metrics and Models

1. Reuse cost-benefits models
2. Maturity assessment
3. Amount of reuse
4. Failure model
5. Reusability
6. Reuse library metrics

Cost-benefits and Maturity

- Reuse cost-benefits models
  - Analysis of quality and productivity payoffs
  - Goal is justify the cost and time invested in systematic reuse
- Maturity assessment models
  - Categorize reuse process by maturity level
  - They are inspired by CMM / CMMI

Amount of Reuse and Failure

- Amount of reuse
  - Metrics are used to assess and monitor the reuse improvement
  - Track percentages of reuse for live cycle objects
- Failure model
  - Used to identify impediments to reuse
  - Help to understand why reuse is not taking place in the organization

Reusability and Reuse Library

- Reusability
  - Metrics that indicate the likelihood that a artifact is reusable
- Reuse library metrics
  - Metrics used to manage and track usage of a repository

Amount of Reuse

- In general

$$\frac{\text{amount of life cycle object reused}}{\text{total size of life cycle object}}$$

- LOC Example

$$\frac{\text{lines of reused code in a system}}{\text{total lines of code in a system}}$$

Reuse Level

- A system is composed of parts at different levels of abstraction
  - The metric level of abstraction must be defined to measure reuse
- Level of abstractions for a Java program
  - System, Package, Class, Method, Lines of Code, etc.

Reuse Level Definitions

- Given that a higher level item is composed of lower level items
  - **L** = the total number of lower level items in the higher level item
  - **E** = the number of lower level items from an external repository in the higher level item
  - **M** = the number of items not from an external repository that are used more than once

Reuse Level Metrics

- External Reuse Level (ERL)

  ERL = E / L
- Internal Reuse Level (IRL)

  IRL = M / L
- Total Reuse Level (TRL)

  TRL = ERL + IRL

Bibliografia da Aula

- Ian Sommerville. **Engenharia de Software**, 9ª Edição. Pearson Education, 2011.
  - Cap. 24 Gerenciamento de Qualidade
- N. Fenton, and S. Pfleeger, **Software Metrics: A Rigorous and Practical Approach**, 2nd ed. Thomson, 1996.
- W. Frakes and C. Terry. **Software Reuse: Metrics and Models**. ACM Computing Surveys, 1996.

Semana da Pós (PPGCC)

- Próxima quarta (30/05) não haverá aula
  - Os alunos de regulares de pós (PPGCC) devem assistir todas as palestras
- Para a disciplina (Reuso) será cobrada a frequência nas palestras de Engenharia de Software e LP
  - Quarta-feira (30/05) das 14:00 as 15:30
  - Procurem pela lista de chamada

Apresentação de Monografia

- Na próxima segunda (04/06), começam as apresentações de monografia
  - Cada apresentação: 20 a 25 minutos
  - Se necessário, continua na quarta (06/06)
- A ordem é a seguinte
  1. Alcemir e Denis
  2. Carlos e Juliana
  3. Douglas e Teo
  4. Leandro Perario
  5. Lucas Meirelles
  6. Sirlan Moraes
  7. Thieres Dias
  8. Lucas Tadeu