Medição e Qualidade de Software

A VALIDATION OF OBJECT-ORIENTED DESIGN METRICS AS QUALITY INDICATORS

1995

Victor R. Basili, Lionel Briand e Walcélio L. Melo

APRESENTAÇÃO: BRUNO AQUINO FILARDI FILHO
PROFESSOR: EDUARDO FIGUEIREDO

ROTEIRO

- Motivação.
- O problema.
- Características do experimento.
- Métricas propostas.
- Análise dos dados.
- Regressão Logística.
- Resultados Obtidos.
- Comparação com métricas de código.
- Conclusão.
- Trabalhos futuros.

MOTIVAÇÃO

- Projetos de grandes sistemas de software demandam muito tempo dentre outros recursos e necessita de ferramentas que auxiliem no gerenciamento das inúmeras atividades em todas as etapas do desenvolvimento.
- Testar grandes sistemas de software é um exemplo de uma atividade de grande consumo de tempo e recursos financeiros.
- Testar todas as partes do software pode ser tornar uma atividade proibitiva do ponto de vista financeiro.
- Muitas métricas têm sido propostas afim de auxiliar desenvolvedores nas diversas etapas, mais nem todas experimentalmente validadas.

O PROBLEMA

- As tecnologias OO começaram a ser inseridas na indústria (1985) e trouxe consigo um novo desafio para as companhias que utilizam métricas como ferramenta de monitoramento, controle, aprimoramento e manutenção de sistemas computacionais. A necessidade de definição e validação de métrica que suportam a especificidade do paradigma OO passaram a se tornar um grande problema pois vários estudos concluíram que métricas tradicionais até então utilizadas não eram suficientemente uteis para cobrir as características dessa nova tecnologia.

  Autores como [Abreu&Carapuça,1994; Bieman&Kang, 1995; Chidamber&Kemerer, 1994] propuseram métricas de OO no entanto com poucas exceções [Briand *et.al.*, 1994] e [Li&Henry, 1993], a maioria deles não apresenta um validação experimental.

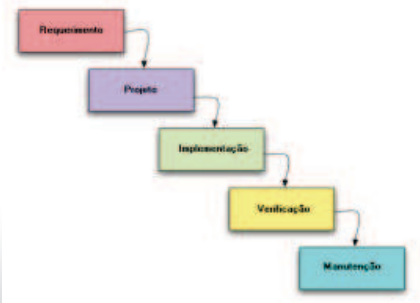
- Portanto esse artigo diz respeito a um trabalho complementar ao realizado por [Chidamber&Kemerer, 1994] cujo objetivo e apresentar os resultados de um estudo experimental realizado na universidade de Maryland a cerca de métricas OO e sua habilidade de identificar classes com potencial de falhas.

CARACTERÍSTICAS DO EXPERIMENTO

- Métricas foram aplicadas e dados foram recolhidos durante o desenvolvimento de oito sistemas de médio porte baseados em requisitos idênticos.
- Linguagem C++.
- Utilizados estudantes da graduação do departamento de ciência da computação da universidade de Maryland sem necessariamente experiência no domino da aplicação.
- Todos tinham alguma experiência em C/C++ e banco de dados relacional.

CARACTERÍSTICAS DO EXPERIMENTO

- Estudantes formaram oito grupos e realizaram o desenvolvimento segundo o modelo cascata com as seguintes etapas: Análise, projeto, implementação, teste e manutenção.



CARACTERÍSTICAS DO EXPERIMENTO

- As ferramentas utilizadas pelos grupos incluindo software para desenvolvimento (IDE), bibliotecas para acesso a banco de dados, interface gráfica entre outras, foram idênticas. Não foram oferecidos treinamentos para a utilização dessas ferramentas.
- A fase de testes foi realizada por um grupo independente composto por profissionais experientes de software o qual testou cada um dos sistemas de forma similar.
- Durante a fase de reparo, os alunos foram convidados a corrigir seu sistema baseado sobre os erros encontrados pelo grupo de teste independente.
- Dados Coletados:
  1. Código fonte ao fim da fase de implementação.
  2. Dados do programa.
  3. Dados sobre os erros encontrados na fase de teste e de reparo.
  4. Código fonte reparado no fim de todo o processo.

Métricas Propostas

Métricas de software não são independentes da linguagem utilizada e, portanto algumas dessas métricas tiveram de ser ligeiramente modificadas de modo a cobrir algumas características da linguagem C++.

Weighted Methods per Class (WMC): Mede a complexidade de uma classe. Segundo [Chidamber&Kemerer, 1994], considera-se todos os métodos de uma classe a ser igualmente complexo, em seguida, WMC é simplesmente o número de métodos e operadores definidos em cada classe. Uma classe com mais funções e operadores é mais complexa, e por consequência tende a ser mais propensos a falhas.

Depth of Inheritance Tree of a class (DIT): Mede o numero de ancestrais da classe já que C++ permite herança múltipla. Uma classe mais profundo localizada em uma estrutura de herança de classe é suposto ser mais propenso a falhas porque a classe herda um grande número de definições de seus ancestrais.

Number Of Children of a Class (NOC): Número de descendentes diretos da classe. Classes com grande número de filhos são difíceis de modificar e geralmente exigem mais testes porque a classe potencialmente afeta todos os seus filhos.

Métricas Propostas

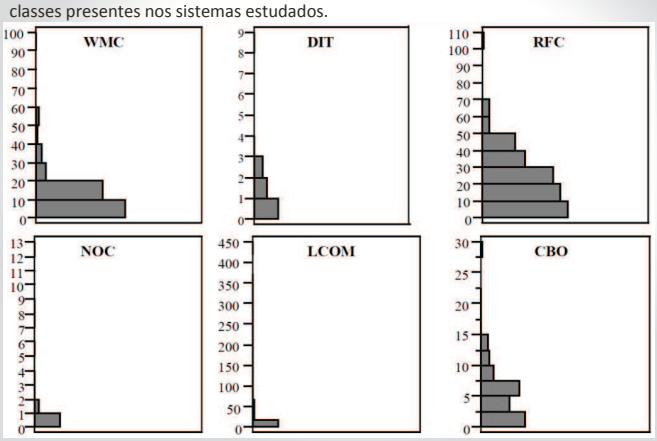
Coupling Between Object classes (CBO): Uma classe é acoplada a outra quando ela usa alguma função ou variável. Essa métrica mede o numero de classes que a classe esta acoplada. O pressuposto por trás dessa métrica é que as classes altamente acopladas são mais propenso a falhas do que as classes fracamente acoplados.

Response For a Class (RFC): Conta o numero de métodos + o numero de métodos chamados pelos métodos da classe. Maior a resposta de uma classe, a maior complexidade da classe, e o mais propenso a falhas e difícil de modificar.

Lack of Cohesion on Methods (LCOM): Mede o número de pares de funções da classe sem variáveis de instâncias compartilhadas, menos o número de pares de funções da classe com variáveis compartilhadas. Uma classe com baixa coesão ou alta falta de coesão entre os seus métodos sugere um projeto inadequado, (ou seja, o encapsulamento de objetos e funções de membro que não deveriam estar juntos) que é susceptível de ser propensas a falhas.

ANÁLISE DOS DADOS E RESULTADOS OBTIDOS

A Figura abaixo mostra as distribuições das métricas analisadas OO baseado em 180 classes presentes nos sistemas estudados.



ANÁLISE DOS DADOS

Alguns pontos observados pelo autor:

- Hierarquias de herança são um pouco plana (DIT) nenhum resultado prevalece entre Dit 1, 2, 3.
- Classe em geral tem poucos filhos (NOC) (este resultado é semelhante ao que foi encontrado em [Chidamber & Kemerer]).
- A maioria das classes mostram falta de coesão (LCOM) próximo de zero.(Resultado mascara os valores negativos presentes)

REGRESSÃO LOGÍSTICA

A **regressão logística** é uma técnica estatística que tem como objetivo produzir, a partir de um conjunto de observações, um modelo que permita a predição de valores tomados por uma variável categórica, frequentemente binária, a partir de uma série de variáveis explicativas contínuas e/ou binárias.

No modelo logístico, usamos os valores de uma série de variáveis independentes para predizer a ocorrência de uma variável dependente.

É útil para modelar a probabilidade de um evento ocorrer como função de outros fatores.

Utilizada com sucesso na medicina, indústria de seguros, instituições financeiras entre outros.

REGRESSÃO LOGÍSTICA

Seu modelo calcula a probabilidade do evento pela seguinte fórmula:

$$P(X) = \frac{1}{1 + e^{-(\alpha + \sum \beta_i X_i)}}$$

- Os termos α e βi neste modelo representam parâmetros desconhecidos que são estimados com base nos dados amostrais.
- Assim, sabendo os parâmetros α e βi e conhecendo os valores das variáveis independentes(valores das métricas) para uma classe, podemos aplicar a fórmula acima para calcular a probabilidade de que essa classe apresente falha ou não – P (X).

REGRESSÃO LOGÍSTICA

Seja $p(x)$ a probabilidade de êxito quando o valor da variável preditiva é x.
Então, seja

$$p(x) = \frac{1}{1 + e^{-(B_0 + B_1 x)}} = \frac{e^{B_0 + B_1 x}}{1 + e^{B_0 + B_1 x}}.$$

Depois de alguma álgebra prova-se que $\frac{p(x)}{1 - p(x)} = e^{B_0 + B_1 x}$;

onde $\frac{p(x)}{1 - p(x)}$ são os odds favoráveis (êxito).

Se tomarmos um valor de exemplo, digamos $p(50) = 2/3$, então

$$\frac{p(50)}{1 - p(50)} = \frac{\frac{2}{3}}{1 - \frac{2}{3}} = 2.$$

RESULTADOS OBTIDOS

Em particular, foi utilizada em um primeiro momento regressão logística univariada para avaliar a relação de cada um das métricas em isolamento com a probabilidade de falhas. Em seguida, foi realizada regressão logística multivariada, a avaliar a capacidade de previsão das métricas que foram satisfatórias na análise univariada.

Metrics	Coefficient	Δψ	p	R^2	Classes
WMC (1)	-0.022	98%	0.0607	0.007	ALL
WMC (2)	-0.086	92%	0.00035	0.024	New-Ext
WMC (3)	-0.027	103%	0.0656	0.0154	DB
WMC (4)	-0.0944	91%	0.0019	0.0467	UI
DIT (1)	-0.485	62%	0.0000	0.0648	ALL
DIT (2)	-0.868	42%	0.0000	0.1314	New-Ext
DIT (3)	-0.475	62%	0.043	0.0187	DB
DIT (4)	-0.29	75%	0.024	0.017	UI
RFC (1)	-0.085	92%	0.0000	0.0648	ALL
RFC (2)	-0.087	92%	0.0000	0.2477	New-Ext
RFC (3)	-0.077	93%	0.0000	0.188	DB
RFC (4)	-0.108	90%	0.0000	0.3624	UI
NOC (1)	3.3848	3000%	0.0000	0.1426	ALL
NOC (2)	3.62	3734%	0.0011	0.362	New-Ext
NOC (3)	2.05	777%	0.0000	0.0826	DB
CBO (1)	-0.142	87%	0.0000	0.068	ALL
CBO (2)	-0.079	92%	0.017	0.02	New-Ext
CBO (3)	-0.086	92%	0.006	0.034	DB
CBO (4)	-0.284	75%	0.0000	0.17	UI

RESULTADOS OBTIDOS

Variáveis da análise estatística.

- Coefficient (aparecendo nas Tabelas 2 e 3), o coeficiente de regressão estimado. Quanto maior for o coeficiente em valor absoluto, o maior impacto da variável explicativa sobre a p probabilidade de uma falha para ser detectado em uma classe.
- DELTA PSI Fornece uma avaliação do impacto da métrica no variável de resposta. Mais especificamente, representa a relação entre a probabilidade de acontecer e a probabilidade de não acontecer quando o valor da métrica é X. Por exemplo, se, por um determinado valor X, y (X) é 2, então é duas vezes mais provável que o classe contém um defeito de que não contém uma falha.
- A significância estatística (p, aparecendo nas Tabelas 2 e 3) fornece uma visão depreciação do coeficiente de estima, Além disso, quanto maior for o nível de significância, maior o desvio padrão dos coeficientes estimados, e menos o impacto crível calculada das variáveis explicativas.
- Quanto maior o R2, mais preciso o modelo. R2 pode ser descrita como uma medida da proporção de incerteza total que é atribuída ao ajuste do modelo.

RESULTADOS OBTIDOS

Análise das 6 métricas.

- (WMC) mostrou-se um pouco significativa (p = 0,06) de forma geral. Para as novas classes e para IU usuário (Interface gráfica e textual), os resultados são muito melhores: p = 0,0003 e p = 0,001, respectivamente. Estes resultados podem ser explicados pelo fato de que a complexidade interna não tem um forte impacto se a classe é reutilizada originalmente ou com modificações muito ligeiras, característica de DB. Como esperado, maior WMC, maior a probabilidade de detecção de falha.
- (DIT) mostrou-se muito significativa (p = 0,0000) de forma geral. Tal como esperado, o maior DIT, maior a probabilidade de detecção de defeitos. Mais uma vez, melhores resultados (R2 Logística passou de 0,06 para 0,13) apareceram quando apenas classes novas e extensivamente modificadas são considerados.
- (RFC) mostrou-se muito significativa de forma geral (p = 0,0000). Previsivelmente, a maior RFC, maior a probabilidade de detecção de defeitos. No entanto, o R2 logística melhorou significativamente para classes novas e extensivamente modificado e classes de interface do usuário (passou de .06 para 0,24 e 0,36, respectivamente). Novamente a mesma razão de WMC.

RESULTADOS OBTIDOS

- (NOC) mostrou-se bastante significativo(exceto em classes UI), mas a tendência observada é contrária ao que se esperava. Quanto maior for o NOC, menor a probabilidade de detecção de defeitos. Esta tendência surpreendente pode ser explicada pelo fato de que a maioria das classes não têm mais de um filho e que classes reutilizadas apresentam grande NOC. Desde que nós temos observado que a reutilização era um fator significativo na densidade de falha [Basili et al, 1995](reuso diminui chance de falha), isto explica porque grandes são as classes NOC menos propenso a falhas.
- (LCOM) mostrou-se insignificante em todos os casos (é por isso que os resultados não estão apresentados na Tabela 2), e isso deve ser esperado uma vez que a distribuição de LCOM mostra uma falta de variabilidade e alguns casos isolados com valores muito altos (ver tabela). Isso decorre, em parte, a partir da definição de LCOM onde a métrica é definida a 0 quando o número de pares atributos compartilhados da classe é maior do que a não compartilhados. Esta definição é definitivamente não é apropriado, no nosso caso, uma vez que define a coesão a 0 classes com o numero de coesão bem distintos estragando a amostra de dados utilizada.
- (CBO) é significativo e mais particularmente para as classes de interface do usuário (P = 0,0000 e R2 = 0,17). Nenhuma explicação satisfatória poderia ser encontrado para as diferenças de padrão entre UI e classes DB.

RESULTADOS OBTIDOS

Análise multivariada.

	Coefficient	p
Intercept	3.13	0.0000
DIT	-0.50	0.0004
RFC	-0.11	0.0000
NOC	2.01	0.0178
RFC	-0.13	0.0072
CBO	-0.238	0.0001
Origin	-1.84	0.0000

- A maior parte das métricas de projeto OO apresentados anteriormente podem ser usadas no início do desenvolvimento para obter modelo preditivo de classes propensas a falhas. A fim de obter um modelo ideal, o autor fez uma segunda análise incluindo essas métricas em um modelo de regressão logística multivariada.
- O atributo Origin foi incluído nessa modelo. Diz respeito ao nível de reutilização de classes. Essa informação foi obtida no final da fase de desenvolvimento quando o nível reutilização dos candidatos foram identificados por meio da analise das bibliotecas disponíveis e a quantidade necessária de mudança pode ser estimada.

RESULTADOS OBTIDOS

Resultados Gerais

- Utilizando o modelo de classificação, os resultados são obtidos usando um limiar de classificação pi (detecção de falha) = 0,5 para a probabilidade de detectar um único defeito de uma dada classe, isto é, quando pi > 0,5, a classe é classificado como defeituoso caso contrário como não defeituoso.
- Foi detectado 250 falhas em 48 classes previstas como propensas a falhas.
- Considerando o método 100% eficaz em achar classes com falhas, 80 classes de 180 teriam que ser avaliadas e 48 de 58 seriam identificadas com falhas antes da análise.
- Se levar em conta as falhas individuais, 250 faltas de 258 seriam detectadas durante a inspeção.

Comparação com métricas de código

Métricas de código propostas por Amadeus [Amadeus, 1994] e utilizando multivariate logistic regression model:

Métricas de Amadeus:
- **MaxStatNext** é o nível máximo de declaração aninhamento em uma classe.
- **FunctDef** é o número de declarações de função.
- **FunctCall** é o número de chamadas de função.

	Coefficient	p
Intercept	0.39	0.0384
MaxStatNest	-0.286	0.0252
FunctDef	0.166	0.0010
FunctCall	-0.0277	0.0000

Comparação com métricas de código

Resultados

- Métricas de código parecem ser um pouco mais pobres como preditores de classes com propensão a erro.
- 112 classes de 180 teriam de ser examinadas. 51 de 58 seriam detectadas
- Se agora levar em conta as falhas individuais, 231 faltas de 258 seriam detectadas durante a inspeção.
- Três classes defeituosas a mais seriam corrigidas (51 versus 48), mas 32 mais classes teria que ser inspecionadas (112 contra 80).
- Além disso, as métricas de projeto OO são melhores preditores de classes contendo um grande número de falhas já que foram encontradas 19 falhas a mais no caso avaliado (250 e 231).

CONCLUSÃO

- Neste experimento, foram coletados dados sobre defeitos encontrados em classes OO. Verificou-se experimentalmente o quanto a propensão de falhas é influenciada por atributos internos (por exemplo, coesão, tamanho) e externos (por exemplo, acoplamento) ao avaliar o projeto de classes OO.
- A partir dos resultados apresentados, cinco das seis métricas OO utilizadasa parecem ser úteis para prever classe com propensão de erro durante as fases iniciais do desenvolvimento.
- Esta validação empírica demonstra que a maioria destas métricas podem ser úteis indicadores de qualidade.
- A maioria destas métricas parecem ser indicadores complementares que são relativamente independentes uns dos outros.

TRABALHOS FUTUROS

- Reaplicar esse estudo na indústria em grandes projetos de desenvolvimento de software OO.
- Construir modelos e fornecer orientação para melhorar a alocação de recursos e esforços nas etapas de teste e verificação.
- Estudar as variações, em termos de definições métricas e resultados experimentais, entre diferentes linguagens de programação OO.

OBRIGADO