



## 15º Congresso de Iniciação Científica

### **AVALIAÇÃO DA RELAÇÃO ENTRE EFICÁCIA E CUSTO NA ATIVIDADE DE TESTE DE SOFTWARE**

#### **Autor(es)**

CAROLINA FONTANA

#### **Orientador(es)**

Waldo Luís de Lucca

#### **Apoio Financeiro**

FAPIC

#### **1. Introdução**

A grande demanda atual por software, requer o desenvolvimento de técnicas para uma melhor qualidade, tanto do produto como do processo. Visando isso, metodologias e técnicas têm sido desenvolvidas, entre essas, técnicas de teste estrutural e funcional. Com o objetivo de revelar a presença de defeitos (MYERS, 1979), o teste de software contribui para se atingir um nível superior de qualidade. Entretanto esse, muitas vezes, é negligenciado por consumir muitos recursos e tempo. Conseqüência disso, é um software que apresenta falhas e sem qualidade necessária para sua utilização. Estudar a relação entre eficácia e custo do teste pode ser positivo para a formulação de técnicas de teste que tenham a máxima eficácia para o reduzido custo. Para isso, este trabalho levou em consideração três hipóteses, a saber: - Hipótese 1 (estrutural): Um segmento de código de programa mais complexo tem maior probabilidade de ter defeitos do que um segmento mais simples. - Hipótese 2 (funcional): Uma funcionalidade mais complexa de um software tem maior probabilidade de ter defeitos do que uma funcionalidade mais simples. - Hipótese 3 (relacional): As funcionalidades mais complexas do software são implementadas pelos segmentos de código mais complexos. Partindo-se das hipóteses citadas anteriormente, foram realizados experimentos buscando estabelecer uma relação entre o custo e a eficácia da atividade de teste e uma relação entre a complexidade de uma funcionalidade de software e o código de programa que a implementa.

#### **2. Objetivos**

O objetivo deste trabalho é estabelecer conceitos e propriedades de software que tornam o teste de software mais eficaz e menos oneroso. Para isso, inicialmente foi estudado as técnicas de teste. Resumidamente,

serão citadas duas técnicas principais: o teste funcional (LUCCA, 2007; PLEEGER, 2004; PRESSMAN, 1995) ou teste de caixa preta (baseia-se na especificação dos requisitos funcionais do software e é realizado nas interfaces do software) e o teste estrutural (RAPPS e WEYUKER, 1985) ou teste de caixa branca (teste realizado nas estruturas do programa, se preocupando com o funcionamento interno de um software). Além disso, para se aplicar o teste, é importante se conhecer as métricas, que tem sido um meio de identificação para estimar o prazo e o custo para aplicar o teste e obter qualidade do software. Uma técnica que permite a medição da complexidade funcional de um software é a “Análise de Pontos de Função” que usa pontos de função (PF) como métrica e foi utilizada para um experimento realizado (FERNANDES, 1995; HAZAN, 1999).

Para confirmação das hipóteses citadas na introdução, foram realizados experimentos que serão especificados no desenvolvimento desse artigo.

### 3. Desenvolvimento

---

Como metodologia para o desenvolvimento dos experimentos, primeiro foi realizada uma revisão bibliográfica sobre engenharia de software, com ênfase no teste de software e suas técnicas de teste estrutural e funcional, além de revisar sobre métricas de software.

Posteriormente, foram realizados dois experimentos, utilizando programas com defeitos reais, seguindo as seguintes etapas (BERTOLINO, 2001; KITCHENHAM, 2002):

1. Definição das hipóteses: foram definidas três hipóteses a serem testadas. 2. Seleção de programas: foram escolhidos os programas a serem utilizados para a realização dos experimentos. 3. Escolha das métricas: definição das métricas utilizadas nos experimentos. 4. Registro das métricas coletadas: foram apresentados os dados coletados a partir das métricas observadas. 5. Análise dos resultados: após a interpretação dos resultados obtidos com os dados apresentados, foi gerada uma conclusão sobre o experimento que permitiu confirmar a hipótese levantada anteriormente. Para o primeiro experimento foram selecionados 20 programas desenvolvidos por Deitel & Deitel, disponíveis no CD do livro Java: Como Programar – 3.<sup>a</sup> Edição. As métricas escolhidas foram Comprimento, Volume, Nível de Linguagem, Tempo Menor, Erros, Vocabulário, Dificuldade, Nível de Esforço e Tempo Maior da Ciência de Software de Halstead (HALSTEAD, 1977); LOC (Linhas de Código) e Complexidade Ciclômica de McCabe (V(G)). O segundo experimento constituiu na escolha do programa Merci, desenvolvido para demonstrar a aplicação do modelo de processos Praxis (PAULA FILHO, 2007), de autoria do Prof. Wilson de Pádua Paula Filho (Universidade Federal de Minas Gerais) e disponível em <http://www.wppf.uaivip.com.br/praxis/>. Como métricas para esse experimento, foram selecionadas: Dificuldade, Esforço e Erros de Halstead; LOC (Linhas de Código) e PF (Pontos de Função).

### 4. Resultados

---

Considerando o primeiro experimento, foi feita uma planilha do Microsoft Excel para registrar as métricas relacionando-as com cada programa, apresentada na Figura 1 desse artigo. Analisando estatisticamente os resultados do primeiro experimento, calcula-se a média de defeitos/programa e relaciona-se com as métricas que estão acima da média. Teve-se como resultado uma média de 0,66 defeitos/programa e todas as outras métricas tiveram valores superiores a média, concluindo assim que quanto maior defeitos/programa, maior a probabilidade de defeitos por programa, que acaba confirmando a primeira hipótese. Já para o segundo experimento, foi gerada a Figura 2, que relaciona o caso de uso com as métricas. Fazendo a análise da correlação entre as métricas, verificou-se que na maioria dos casos de uso, funcionalidades mais complexas (com maior número de PF), tem maior número de defeitos do que funcionalidades mais simples. Porém, para aproximadamente 33,33% das funcionalidades, o número de defeitos diminuiu e, para 22,22% das funcionalidades, o número de defeitos aumentou. Isso provocou o desvio no coeficiente de correlação. Apesar do desvio na correlação, o experimento 2 serviu para firmar a segunda hipótese. Utilizando agora a mesma tabela do experimento 2, pode-se comparar os pontos de função com as métricas dificuldade, esforço e LOC, isso para tentar provar a terceira hipótese. Para isso, foi calculado os coeficientes de correlação entre as métricas e foi gerado a Figura 3. Como resultado, observa-se que há uma forte

correlação entre PF e dificuldade. Já entre PF X esforço e PF X LOC, a correlação não é tão forte, isso porque algumas funcionalidades têm segmentos de códigos que provocam desvio no resultado, mas é uma boa correlação. Em geral, as funcionalidades mais complexas do software são implementadas pelos segmentos de código mais complexos, confirmando assim a terceira hipótese.

## 5. Considerações Finais

---

Este trabalho teve como objetivo mostrar que é possível tornar o teste de software mais eficaz e menos oneroso. Pensando nisso, foram realizados experimentos com softwares documentados. Os resultados obtidos permitiram comprovar as três hipóteses levantadas na introdução desse relatório. Além dos dois experimentos feitos e descritos nos resultados, foi desenvolvido mais um experimento que não foi concluído por falta de informações na documentação do software. Por fim, para confirmar os resultados obtidos, novos experimentos podem ser realizados para confirmar, de forma mais segura, as hipóteses investigadas neste trabalho.

## Referências Bibliográficas

---

- BERTOLINO, A. Software Testing. In: A Abran, J.W. Moore, P. Bourque, R. Dupuis e L.L.Tripp. Ed. Guide to the Software Engineering Body of Knowledge: Trial Version. IEEE Press, 2001.
- FERNANDES, A.A. Gerência de Software Através de Métricas. Ed. Atlas, 1995.
- HALSTEAD, M.H. Elements of Software Science. Operating and Programming System Series, Elsevier Science, 1977.
- HAZAN, C. Análise de Pontos por Função: Uma Abordagem Gerencial. Anais da XIX Jornada de Atualização em Informática da Sociedade Brasileira de Computação, vol. 2, 1999, p. 287-326.
- KITCHENHAM, B.A. , PFLEEGER, S.L., PICKARD, L.M., JONES, P.W., HOAGLIN, D.C., EMAM, E. e ROSENBERG, J. Preliminary Guidelines for Empirical Research in Software Engineering. IEEE Transactions on Software Engineering, V.38, N.8, agosto de 2002, PP. 721-734.
- LUCCA, W.L. Teste de Software. Relatório Técnico – Universidade Metodista de Piracicaba, Piracicaba, 2007.
- MEYERS, G.J. The Art of Software Testing. John Willey & Sons, 1979.
- PAULA FILHO, W.P. Processo Práxis – Disponível em <[www.wppf.uaivip.com.br/praxis](http://www.wppf.uaivip.com.br/praxis)>. Acesso em: 01/02/2007.
- PFLEEGER, S.L. Engenharia de Software – Teoria e Prática. 2.ª Edição. Pearson: Prentice Hall, 2004.
- PRESSMAN, R.S. Engenharia de Software. Makron Books, 1995.
- RAPPS, S. e WEYUKER, E.J. Selecting Software Test Data Using Data Flow Information. IEEE Transactions on Software Engineering, v.11, n.4, abril de 1985, p.367-375.

## Anexos

---



	Compr.	Volume	Niv. Ling.	T. Menor	Erros	Vocab.	Dificuld.	Niv. L.
Addition	88	471,46	0,08	308,91	0,16	41	13,10	61
Analysis	130	741,06	0,05	808,14	0,25	52	21,81	16
Average1	107	597,59	0,06	501,55	0,20	48	16,79	10
Average2	161	939,10	0,04	1126,91	0,31	57	24,00	22
BinarySearch	435	2972,31	0,03	5366,14	0,99	114	36,11	107
BreakLabelTest	106	604,25	0,06	530,87	0,20	52	17,57	10
Comparison	188	1091,78	0,03	1637,67	0,36	56	30,00	32
ContinueLabelTest	100	567,24	0,06	491,61	0,19	51	17,33	98
Craps	484	3236,27	0,03	5752,13	1,08	103	35,55	115
FactorialTest	103	565,66	0,05	547,02	0,19	45	19,34	10
FibonacciTest	202	1233,92	0,05	1233,92	0,41	69	20,00	24
Interest	125	741,34	0,07	560,88	0,25	61	15,13	11
LogicalOperators	269	1690,77	0,04	1944,39	0,56	78	23,00	38
Maximum	154	894,33	0,07	613,99	0,30	56	13,73	12
RandomInt	99	561,57	0,06	499,47	0,19	51	17,79	99
RollDie	197	1215,48	0,05	1330,95	0,41	72	21,90	26
Scoping	156	880,44	0,07	623,65	0,29	50	14,17	12
SquareInt	112	638,45	0,06	556,99	0,21	52	17,45	11
Sum	71	385,26	0,07	266,62	0,13	43	13,84	53
SwitchTest	170	987,25	0,04	1221,72	0,33	56	24,75	24
Média	172,85	1050,777	0,0535	1296,177	0,3505	60,35	20,668	259

Figura 1. Relação entre programas e métricas de software.

Caso de Uso	PF	LOC	Dificuldade	Halstead Esforço	Err
Gestão de Usuários	23	670	579,27	123.667,82	4
Gestão de Mercadorias	26	839	595,91	194.892,35	6
Operação de Venda	8	912	563,23	172.771,88	5
Abertura e Fechamento do Caixa	13	323	217,88	54.175,84	2
Emissão de Nota Fiscal	8	304	166,16	94.984,68	2
Gestão de Fornecedores	25	472	361,05	114.061,69	3
Gestão Manual de Estoque	6	267	169,58	57.499,07	1
Gestão de Pedidos de Compra	36	1.234	922,84	270.272,89	8
Emissão de Relatórios	20	461	396,69	88.650,54	2
	165	5.482	3.972,61	1.170.976,76	37
Médias:	18	609	441,40	130.108,53	4

Figura 2. Relação entre caso de uso e métricas.

Métricas	coeficiente de correlação
PF X dificuldade	0,77
PF X esforço	0,69
PF X LOC	0,65

Figura 3. Coeficientes de correlação entre PF, dificuldade, esforço e LOC.