Critérios Potenciais Usos: Análise da Aplicação de um Benchmark

José Carlos Maldonado
Silvia Regina Vergílio
Marcos Lordello Chaim
Mario Jino

Departamento de Ciências de Computação e Estatística
ICMSC
Universidade de São Paulo -USP
C.P.668 CEP 13560 - São Carlos - SP
Telefone 0162 - 726222 -r 3847 (e-mail :jcmaldon@icmsc.usp.ansp.br)

Resumo

Os resultados da aplicação de um benchmark para avaliação empírica dos critérios Potenciais Usos são apresentados. O experimento foi conduzido utilizando-se uma ferramenta de teste, denominada POKE-TOOL. A análise dos resultados obtidos indica que, do ponto de vista prático, esses critérios demandam um baixo número de casos de teste e contribuem para demonstrar que os critérios de teste estrutural baseados em análise de fluxo de dados são factíveis. Vários modelos de estimativas para prever o número de casos de teste requeridos são analisados. São também discutidos alguns aspectos de medidas de complexidade de software relacionados com as atividades de teste de software.

Abstract

The results of benchmarking Potential Uses Criteria for empirically evaluating these criteria are presented. A testing tool named POKE-TOOL, that supports Potential Uses Criteria application, has been used. From the practical point of view, this benchmark has contributed to demonstrate that data-flow based criteria are a real goal, i.e., the results obtained are an indication that satisfaction of these criteria can be adopted as a practical goal of testing; Potential Uses criteria require a small amount of test case to be satisfied. Many estimation models of test case number to satisfy these criteria are analyzed. Also, some complexity aspects related to software testing activities are discussed.

1 Introdução

O estabelecimento de estratégias, métodos e critérios de teste visa fornecer uma maneira sistemática para selecionar um subconjunto relativamente pequeno do domínio de entrada — um conjunto de casos de teste T — e ainda assim ser efetivo quanto à meta principal das atividades de testes: revelar a presença de defeitos no programa. Idealmente, o programa deveria ser exercitado para todos os valores de entrada possíveis, no entanto é conhecido que, na prática, o teste exaustivo é impossível devido à restrições de tempo e custo. Quanto ao item custo, as atividades de testes de software constituem aproximadamente metade do custo total do desenvolvimento de software.

Um dos critérios de teste de programas mais difundido e mais utilizado é o critério *teste de ramos* que requer a execução de todos os ramos (arcos) de um grafo de programa; os arcos do grafo de programa representam possíveis transferências de controle entre os comandos do programa. Esse critério pertence à classe de critérios de teste estrutural que usam informações de fluxo de controle da estrutura do programa para derivar os requisitos de teste. A utilização dessa classe de critérios tem mostrado que eles não são efetivos para indicar a presença de erros nos programas, principalmente erros computacionais. Este fator foi uma das motivações para que mais recentemente fosse introduzida uma grande variedade de critérios que utilizam a análise de fluxo de dados como fonte de informação para derivar os requisitos de teste; esses critérios são conhecidos como *critérios baseados em análise de fluxo de dados*[HER76, RAP85, FRA87, NTA88, MAL88, MAL91].

Os critérios baseados em análise de fluxo de dados requerem que as interações que envolvem definições de variáveis de programa e subseqüentes referências a essas definições sejam testadas; portanto, esses critérios baseiam-se, para a derivação de requisitos de teste, nas associações entre uma definição de uma variável e os seus possíveis usos subseqüentes.

Maldonado, Chaim e Jino [MAL88] definiram os *Critérios Potenciais Usos* baseados no conceito de *potencial uso* ; são denominados: critérios *todos-potenciais-du-caminhos*, *todos-potenciais-usos* e *todos-potenciais-usos/du*. Estes requerem associações independentemente da ocorrência explícita de uma referência a uma determinada definição; se um uso dessa definição pode existir — *um potencial uso* — a potencial associação é requerida.

Estudos comparativos entre esses critérios têm sido conduzidos suportados principalmente por uma relação de inclusão, que estabelece uma ordem parcial entre esses critérios, e pelo estudo da complexidade dos critérios [CLA85, RAP85, NAT88]. A complexidade de um critério é definida como o número máximo de casos de teste requerido, no pior caso. Estes dois pontos refletem em geral os aspectos básicos que devem ser considerados no processo de definição e ou de seleção de um critério C: i) o critério deve incluir o critério todos os ramos, ou seja, um conjunto de casos de teste que exercita os elementos requeridos pelo critério C deve exercitar todos os ramos do programa; ii) do ponto de vista de fluxo de dados, ao menos um uso de todo resultado computacional deve ser requerido pelo critério C; e iii) o conjunto de casos de teste requeridos deve ser finito. Em particular os critérios Potenciais Usos satisfazem estes três requisitos básicos [MAL91].

Segundo Ntafos [NTA88], um fator comum no custo das atividades de teste — geração e execução dos casos de teste, e análise dos resultados — é o número de casos de teste necessários para satisfazer cada critério. Todos os critérios baseados em análise de fluxo de dados têm complexidade de ordem exponencial, mesmo o mais fraco dos critérios dessa

classe [MAL91]. A complexidade do critério fornece, portanto, um limitante superior para a estimativa do número de casos de testes necessários para satisfazer um dado critério; do ponto de vista prático, no entanto, é de fundamental importância a realização de experimentos que forneçam indicações da média do número de casos de testes requeridos para classes de programas mais usuais.

Estudos empíricos têm sido conduzidos com o objetivo principal de investigar o custo do uso de critérios de testes baseados em fluxo de dados nas atividades de teste (por exemplo, vide [WEY90, BIE89]). O conjunto de programas utilizado por Weyuker [WEY90] — extraído do livro de Kernighan & Plauger [KER81] — tem sido encarado como um benchmark [BEI90]. Um outro objetivo desses estudos é determinar uma maneira de estimar o número de casos de testes necessários para satisfazer um dado critério para um dado programa P a ser testado. Weyuker [WEY90] observou que para a Família de Critérios de Fluxos de Dados [RAP85], o estudo empírico indicou que o número de casos de teste requeridos para satisfazer os critérios dessa família pode ser visto como linear em função do número de comandos de decisão do programa. Esta conclusão é suportada principalmente pelo resultado da regressão linear entre o número de casos de teste — *variável resposta* — e o número de comandos de decisão — *variável independente* . Outro ponto importante é a caracterização do pior caso empírico, denominado *complexidade empírica*: o máximo valor obtido dentre os quocientes do número de casos de teste pelo número de comandos de decisão.

A falta de planejamento dos recursos necessários para o desenvolvimento de software tem sido apontada como um dos motivos da crise de software. O planejamento das atividades de teste deve fazer parte do planejamento global do sistema; a falta de tempo e de recursos humanos capacitados, e a indisponibilidade de ferramentas adequadas são os principais problemas enfrentados pelas equipes de teste, decorrentes da falta de planejamento e da consequente elaboração de um plano de teste.

Modelos para estimar o esforço das atividades de teste são iniciativas que contribuem e motivam uma prática mais sistemática e disciplinada no desenvolvimento de um produto de software. A quantificação do esforço de teste pode então ser incorporada em planos de teste e auxiliar nas atividades de garantia de qualidade. Esta prática, em geral, leva a um aprimoramento da qualidade global do produto e o custo de atividades de outras fases do desenvolvimento pode ser sensivelmente reduzido; por exemplo, o custo do teste de regressão na fase de manutenção.

Neste trabalho são descritas a realização e análise da aplicação dos Critérios Potenciais Usos, utilizando-se a ferramenta de teste POKE-TOOL[1] [MAL89, CHA91], no mesmo benchmark extraído de [KER81]. São três os objetivos deste trabalho:

i) Contribuir para demonstrar a factibilidade dos critérios baseados em análise de fluxo de dados, em particular, dos Critérios Potenciais Usos;

ii) Determinar modelos para estimar, nas diversas etapas de desenvolvimento do software, o número de casos de teste necessários para testar um dado programa com os critérios Potenciais Usos.

[1] O nome da ferramenta foi mudado para POKE-TOOL.

iii) Explorar a influência de outras características do produto em teste na determinação dos modelos de estimativas, além do número de comandos de decisão. Por exemplo, seria de se esperar que o número de variáveis utilizadas no programa fosse significativo na estimativa do número de casos de teste necessários para satisfazer os critérios baseados em análise de fluxo de dados, em particular para os critérios Potenciais Usos.

Na Seção 2 são descritas a aplicação e a coleta de medidas, base para a análise apresentada na Seção 3; nesta são discutidos e analisados diversos modelos para a estimativa do número de casos de testes requeridos pelos critérios Potenciais Usos e comparados os resultados com os obtidos para a Família de Critérios de Fluxo de Dados. Ainda, na Seção 3 são fornecidos resultados indicativos da influência das variáveis de controle consideradas nos modelos de estimativa e levantadas algumas conjecturas sobre o estabelecimento de métricas de complexidade de software. As conclusões e direções de pesquisa são apresentadas na Seção 4. Para uma melhor compreensão do trabalho, no Apêndice A são fornecidos alguns conceitos e terminologia básicos e a definição de um dos critérios Potenciais Usos.

2 Aplicação do Benchmark e Coleta de Medidas

Neste experimento procura-se explorar, além da influência do número de comandos de decisão na estimativa do número de casos de teste, a relação entre a ocorrência de variáveis no programa e o número de casos de teste requeridos para satisfazer os critérios baseados em análise de fluxo de dados, em particular os critérios Potenciais Usos. Um outro aspecto explorado foi a a influência dessas variáveis de controle no número de caminhos e associações não executáveis[2]; resultados relativos a esse aspecto são apresentados em [VER92].

Deve-se salientar que são vários os pontos que influenciam o número de casos de teste requeridos por um critério baseado em fluxo de dados; por exemplo, o número de comandos de decisão tem uma forte influência, assim como as definições e (potenciais) usos ao longo do programa. Outros pontos são de relevância, tais como o tipo de comando (if, while, ...), a ordem de ocorrência dos comandos, o número de definições e a seqüência de suas ocorrências, entre outros. Por exemplo, pode-se elaborar programas com distribuição adequada de ocorrências de variáveis com o objetivo de maximizar o número de elementos requeridos, assim como, pode-se distribuir estas ocorrências de forma a minimizar esse número. Esses fatores foram considerados na análise de complexidade dos Critérios Potenciais Usos [MAL88, MAL91] e têm forte influência nas estimativas realizadas; sem dúvida, são explicações para a variabilidade dos modelos apresentados, pois as variáveis de controle utilizadas na determinação dos modelos não captam todos esses aspectos.

Intuitivamente, espera-se que o uso de variáveis tenha uma forte influência no número de casos de teste uma vez que os caminhos e associacões requeridos são extremamente dependentes do fluxo de dados; mais ainda, na determinação da complexidade dos critérios,

[2] Um *caminho é executável ou factível* se existe um conjunto de valores pertencentes ao domínio de entrada do programa que causam a sua execução; caso contrário, diz-se que ele é *não executável*.

a ocorrência e a distribuição das definições das variáveis do programa foi um fator relevante. Neste sentido, serão observados o número de variáveis utilizadas no programa, o número de definições de variáveis, assim como o número de nós com definição de variáveis. A seguir, são descritos a realização do experimento e a organização e coleta dos resultados obtidos.

2.1 Estratégia adotada para a condução do Benchmark

Inicialmente, para cada um dos programas que compõem o benchmark, determinaram-se os seguintes dados (*variáveis de controle*), sintetizados na Tabela 1:

- número de comandos de decisão (C1).
- número de variáveis utilizadas (C2).
- número de definições de variáveis (C3).
- número de nós com definição de variáveis (C4).
- número de nós do grafo de programa (C5).

Para a realização do benchmark propriamente dito, ou seja, elaboração dos conjuntos de casos de testes (CCT) e análise de adequação dos conjuntos de casos de teste em relação a alguns dos critérios Potenciais Usos — *todos-potenciais-usos, todos-potenciais-usos/du* e *todos-potenciais-du-caminhos* — adotaram-se, a exemplo de [WEY90], algumas diretrizes, concretizadas nas etapas descritas a seguir, que deverão constituir uma conduta padrão em experimentos semelhantes posteriores que venham a ser realizados.

1) Leitura da descrição funcional da unidade, de detalhes e alternativas que nortearam a implementação das unidades a serem testadas.

2) Elaboração do conjunto inicial de casos de teste (CICT). A minimização dos conjuntos de casos de teste não é almejada. São selecionados casos de teste típicos da aplicação, atômicos, i.e., com propósito único, ao invés de selecionarem-se casos de teste que cobrissem várias características simultâneamente; no entanto, não ficou caracterizada a aplicação explícita de nenhum critério de teste funcional. Esses conjuntos foram derivados essencialmente a partir de exemplos, sugestões e considerações de Kernighan & Plauger [KER81], de maneira informal e não sistemática.

3) Implementação da unidade a ser testada utilizando a linguagem de programação *C*, uma vez que a versão atual da POKE-TOOL suporta apenas esta linguagem.

4) Submissão do conjunto CICT à unidade a ser testada, sob o controle da POKE-TOOL.

5) Análise de adequação do CICT em relação aos critérios Potenciais Usos (PU).

6) Geração e submissão de novos casos de testes a partir de informações oriundas da análise de cobertura feita pela POKE-TOOL, com o objetivo de cobrir as associações requeridas pelo critério todos-potenciais-usos. Casos de testes são gerados até que todas as associações executáveis requeridas sejam exercitadas. Se o conjunto de caminhos e associações ainda a serem exercitados for composto somente de caminhos não executáveis, dá-se por encerrado o teste da unidade de acordo com o critério em questão; neste caso, diz-se que o Conjunto de Casos de Teste "quase satisfaz" o critério [WEY90]. A determinação da executabilidade de um caminho ou associação é indecidível, e é de responsabilidade única do testador, uma vez que a POKE-TOOL, na versão atual, não fornece suporte nesta direção. O Conjunto de Casos de Testes nesta etapa é denominado CCT/PU.

7) O mesmo procedimento foi adotado para associações requeridas pelo critério todos-potenciais-usos/du ainda não exercitadas. O Conjunto de Casos de Testes nesta etapa é denominada CCT/PUDU.

8) O mesmo procedimento foi adotado para os caminhos requeridos pelo critério todos-potenciais-du-caminhos ainda não exercitados. O Conjunto de Casos de Testes nesta etapa é denominado CCT/PDU.

9) Coleta e Organização de Informações para a Análise dos resultados do benchmark.

Como resultado imediato da realização do benchmark, os seguintes resultados, sintetizados na Tabela 2, foram obtidos:

- Conjunto inicial dos Casos de Testes (CICT) e a análise da adequação do CICT para cada um dos critérios, ou seja, a análise de cobertura em relação a cada um dos critérios Potenciais Usos.
- Conjunto de Casos de Testes para o Critério CR1 (CFCT/PU)
- Conjunto de Casos de Testes para o Critério CR2 (CFCT/PUDU)
- Conjunto de Casos de Testes para o Critério CR3 (CFCT/PDU)
- Conjunto de associações e caminhos não executáveis para os critérios Potenciais Usos.

Tabela 1: Variáveis de Controle relativas ao Benchmark

UNIDADE	# Comandos de Decisão / # nós (C1/C5)	# Variáveis Utilizadas (C2)	# Definições de variáveis (C3)	# Nós com Definição de Variáveis (C4)
DODASH	6/19	7	8	3
ARCHIVE	6/18	4	9	6
RQUICK	6/14	11	14	6
GETFNS	6/17	6	13	9
COMPARE	6/14	9	16	5
ENTAB	6/15	4	10	8
EXPAND	6/18	2	6	5
CMP	5/16	3	5	2
COMPRESS	5/16	3	6	5
UNROTATE	7/20	5	21	13
TRANSLIT	12/30	8	22	12
COMMAND	15/19	29	44	15
GETCMD	14/43	2	16	15
AMATCH	7/22	6	19	12
OMATCH	15/29	5	12	8
GTEXT	3/9	9	13	5
GETDEF	9/26	7	12	7
GETONE	9/22	7	8	3
GETNUM	6/19	4	4	1
MAKEPAT	10/30	8	22	13
SPREAD	5/14	9	18	8
CHANGE	5/11	4	12	7
GETFN	5/13	5	7	4
SUBST	9/27	17	36	16
EDIT	9/22	7	16	10
GETLIST	6/16	9	19	8
APPEND	5/16	6	12	8
CKGLOB	6/19	6	11	8
OPTPAT	5/15	3	5	3

Tabela 2: Variáveis Respostas e Associações Requeridas relativas ao Benchmark

UNIDADE	Card.	Cardinalidade CFCT			# Assoc. não exec./requeridas		
	CICT	CR1	CR2	CR3	CR1	CR2	CR3
DODASH	7	15	15	15	0/33	0/33	0/21
ARCHIVE	7	7	7	7	0/17	0/17	0/17
RQUICK	1	2	2	2	6/55	19/55	33/51
GETFNS	5	7	7	7	8/44	14/44	9/34
COMPARE	10	11	12	12	0/38	6/38	30/64
ENTAB	8	14	14	14	23/80	31/80	41/70
EXPAND	12	12	13	13	14/39	15/39	10/25
CMP	7	7	7	7	3/13	3/13	3/12
COMPRESS	12	14	14	14	3/39	6/39	10/34
UNROTATE	3	6	6	6	11/91	39/91	29/70
TRANSLIT	33	37	38	48	6/138	6/138	203/333
COMMAND	15	15	15	15	3/47	3/47	19/61
GETCMD	15	15	15	15	0/119	0/119	0/119
AMATCH	9	14	14	14	70/112	70/112	61/87
OMATCH	13	13	13	13	12/33	12/33	25/43
GTEXT	5	5	5	5	6/33	11/33	10/22
GETDEF	13	20	23	23	3/59	9/59	9/49
GETONE	8	12	14	20	3/62	6/62	147/177
GETNUM	7	8	8	8	0/8	0/8	5/12
MAKEPAT	36	39	39	39	73/207	87/207	169/253
SPREAD	7	12	17	17	7/61	10/61	14/47
CHANGE	8	8	8	8	1/38	1/38	2/27
GETFN	7	7	7	7	2/18	2/18	7/20
SUBST	10	29	26	28	43/219	74/219	235/322
EDIT	17	32	40	40	20/130	21/130	261/357
GETLIST	6	13	14	15	20/82	22/82	64/102
APPEND	7	7	7	7	11/49	12/49	22/43
CKGLOB	7	7	7	7	1/35	11/35	7/27
OPTPAT	5	5	5	5	1/13	1/13	13/21

3 Análise dos Resultados do Benchmark

Alguns pontos devem ser ressaltados quanto aos programas testados e quanto aos resultados obtidos antes de procedermos à análise propriamente dita desses resultados. Uma consideração importante é que dois dos programas testados — RQUICK e AMATCH — são recursivos e a atual versão da POKE-TOOL não possibilita um tratamento uniforme de procedimentos recursivos e procedimentos não recursivos [MAL91]. A análise foi então conduzida para dois conjuntos de pontos distintos — um conjunto incluindo procedimentos recursivos, totalizando 29 pontos observados, e o outro excluindo os procedimentos recursivos, com 27 pontos.

Alguns procedimentos apresentaram maior dificuldade de serem testados do que outros; por exemplo, os programas COMMAND, GETCMD e OMATCH demandaram cerca de t casos de teste, onde t é o número de comandos de decisão. Estes programas apresentam complexidade ciclomática 16, 15, 16, respectivamente, e contêm estruturas de controle CASE ou ELSEIF. Po outro lado, outros programas, de complexidade ciclomática da mesma ordem, apresentaram dificuldades para a elaboração de conjuntos de casos de teste que satisfizessem os critérios Potenciais Usos; por exemplo, os programas MAKEPAT, SUBST e TRANSLIT demandaram cerca de $4 * t$ casos de teste. Estes resultados são coerentes com os resultados discutidos por MacCabe [McC76].

Dois outros programas — EDIT e SPREAD, de complexidade ciclomática menor que dez, apresentaram dificuldades de serem testados, contrariando a observação de McCabe de que programas com complexidade ciclomática menor que 10 (dez) são mais fáceis de serem testados do que programas com complexidade ciclomática maior do que dez.

Uma possível explicação é que os programas que apresentaram maior dificuldade, citados acima, contêm estruturas de controle similares à estrutura de controle que maximiza o número de potenciais-du-caminhos [MAL91].

Outro programa que merece destaque é o UNROTATE, de complexidade ciclomática oito, pois exigiu menos do que t casos de teste; este programa utiliza 5 variáveis e 21 definições de variáveis e contém 4 laços, o que não favorece a geração de potenciais-du-caminhos.

Para todos os programas citados acima, o número de potenciais-du-caminhos presentes em cada um deles parece constituir uma explicação razoável para as dificuldades identificadas. Todos os demais programas do benchmark demandaram entre t e $3 * t$ casos de teste e, da mesma forma, o número de potenciais-du-caminhos reflete as dificuldades encontradas para o teste dessas unidades.

Para cada um dos critérios Potenciais Usos, os seguintes resultados, sintetizados na Tabela 3, foram computados em função dos dados coletados:

1) A média ponderada M da razão entre o número de comandos de decisão t de um dado programa pelo número de casos de teste, nct, suficiente para satisfazer o critério selecionado (*análise do caso médio empírico*); expressou-se então nct em função de M, ou seja, $nct = (1/M)t$, onde M é definido por

$$M = \frac{1}{n} \sum_{i=1}^{n} \frac{t_i}{nct_i}$$

e n é o número de programas considerados.

Tabela 3: Valores Médios

	CR1	CR2	CR3
media ponderada	nct = 1.38t	nct = 1.41t	nct = 1.44t
maximo nct/t	3.9	4.45	4.45
minimo nct/t	0.34	0.34	0.34

2) O máximo valor da razão entre o número de comandos de decisão, t, de um dado programa pelo número de casos de teste, nct, suficiente para satisfazer o critério selecionado (*análise do pior caso empírico*).

3) O mínimo valor da razão entre o número de comandos de decisão t de um dado programa pelo número de casos de teste, nct, suficiente para satisfazer o critério selecionado.

A título de comparação, para o critério *todos-du-caminhos* , o critério mais exigente da Família de Critérios de Fluxo de Dados, a complexidade empírica no pior caso, para o benchmark em questão, é *3.67*, ou seja, o número máximo de casos de teste (nct) é igual a *3.67* vezes o número de comandos de decisões (t), expresso na equação $nct = 3.67 * t$; considerando a média ponderada M, temos que $nct = 0.81t$. Para o *critério todos-potenciais-usos, no pior caso empírico*, $nct = 4.45 * t$; considerando a média ponderada M, tem-se $nct = 1.44t$. Deve-se ressaltar que o mesmo programa contribuiu para o pior caso empírico dos três critérios Potenciais Usos; uma das explicações é que este programa contém estruturas de controle similares à da estrutura que maximiza o número de potenciais-du-caminhos.

Esses dados indicam que, do ponto de vista prático, os critérios Potenciais Usos requerem, em geral, um número bastante modesto de casos de teste. Considerando que a complexidade desses critérios é de ordem exponencial e que a satisfação de um critério de teste usualmente é um dos itens dos critérios de encerramento das atividades de teste, esses resultados motivam a realização de outros trabalhos em torno dos critérios Potenciais Usos; em particular em torno do conceito *potencial uso*, fazendo-se por exemplo, a combinação e o estudo comparativo entre critérios baseados em fluxo de dados.

Objetivando fornecer uma maneira de estimar o esforço necessário para conduzir as atividades de teste, vários modelos de estimativas foram explorados para os critérios Potenciais Usos, com o apoio do Sistema MINITAB [RYA76] – um Sistema de Computação Estatístico de propósito geral – , utilizando-se as *variáveis de entrada (variáveis independentes ou preditores)* descritas na Tabela 1. Duas variáveis respostas ou dependentes foram alvo da análise apresentada: i) o número de casos de teste e ii) o número de caminhos não executáveis. A Tabela 4 apresenta uma síntese dos melhores modelos obtidos para estimar o número de casos de teste para o critério todos-potenciais-du-caminhos, considerando-se procedimentos não recursivos; o intervalo de validade desses modelos é

Tabela 4: Modelos de Estimativas
Critério todos potenciais-du -caminhos (CR3)

Expr.	R2	S
nct = - 16.3 + 4.86 t	82.2	4.837
nct = - 17.1 + 4.58 t + 0.454 variav	83.7	4.750
nct = - 16.4 + 4.23 t + 0.333 def	85.8	4.433
nct = - 16.5 + 4.24 t + 0.624 nodef	85.0	4.545
nct = 7.04 + 0.0976 ducam	82.7	5.155

$3 \leq t \leq 12$. Para uma melhor ilustração é também apresentada uma síntese de alguns parâmetros estatísticos desses modelos. A coluna ($R2$) representa o quadrado do coeficiente de correlação amostral r e indica a proporção de variabilidade explicada pelo modelo; a coluna (S), por sua vez, indica o desvio padrão da estimativa e fornece uma indicação do desvio padrão σ.

Na exploração dos modelos para estimativa, os programas (pontos) discutidos no início desta seção, caracterizaram-se como os mais influentes na determinação do modelo mais adequado. Observou-se que alguns desses pontos chegavam até mesmo a mascarar a relevância das variáveis de entrada no estabelecimento do modelo propriamente dito. Os três pontos com $t \geq 12$ observados, indicam uma tendência de diminuição da relação nct/t à medida que o número de comandos de decisão t aumenta; esperava-se que, em geral, à medida que o tamanho do programa crescesse, o número de casos de teste requeridos crescesse até, eventualmente, numa taxa maior. Esta mesma tendência foi observada nos dados do experimento conduzido por Shimeall e Leveson apresentados por Weyuker [WEY90]. Uma conjectura na tentativa de explicar este comportamento é que a relação $varia/t$ diminui sensivelmente à medida que t cresce, ou seja, o número de variáveis utilizadas não cresceria na mesma proporção que o número de comandos de decisão; adicionalmente, o número de definições de variáveis poderia crescer e suas distribuições poderiam contribuir no sentido de minimizar a inter-relação entre a estrutura de fluxo de controle e o fluxo de dados no programa. Isto de certa forma contribuiria para minimizar o esforço das atividades de teste. Estes fatos apontam a necessidade de que novos dados sejam obtidos tanto para a classe de programas que caracterizou este benchmark, como para outras classes de programas e áreas de aplicação. Ainda, esses resultados ressaltam a necessidade de uma análise estatística mais minuciosa dos dados obtidos na condução do benchmark, principalmente a realização de uma análise de inferência e o uso de métodos robustos de regressão [ACH90, DENS2] para a escolha de um modelo definitivo de estimativa do número de casos de teste necessários para satisfazer os critérios Potenciais Usos.

Apesar das limitações abordadas acima, dois pontos devem ser ressaltados como consequência dessa análise exploratória. O primeiro refere-se à relevância das variáveis de

controle *variav*, *def* e *nodef* na determinação dos modelos de estimativa do número de casos de testes e do número de caminhos não executáveis. Considerando-se os níveis de significância usuais utilizados na realização de análises estatísticas utilizando-se a distribuição t de Student e através de uma análise breve dos resíduos, conclui-se que todas as três introduzem melhoras significativas no modelo que leva em conta somente o número de comandos de decisão e são importantes na determinação do modelo. O segundo ponto, refere-se à forte correlação do número de potenciais-du-caminhos com o número de casos de testes e com o número de caminhos e associações não executáveis para o critério todos-potenciais-du-caminhos. Isto significa que dado o número de potenciais-du-caminhos a previsão do número de casos de teste e do número de caminhos não executáveis presentes no programa é realizada com uma confiança bastante grande.

Obviamente, alguns modelos são mais adequados para as fases iniciais do desenvolvimento do que outros; por exemplo modelos que consideram o número de potenciais-du-caminhos, o número de nós com definição de variáveis ou mesmo o número de definições de variáveis não seriam tão úteis na fase de análise e especificação quanto os modelos que consideram somente o número de comandos de decisão e o número de variáveis como variáveis de controle. Os modelos apropriados para as fases iniciais de desenvolvimento de software são muito importantes para as atividades de planejamento do desenvolvimento.

O conceito de *potencial uso* e a consequente definição de *potencial-du-caminho* fornecem, na realidade, uma medida do grau de inter-relacionamento entre a estrutura de controle do programa e o fluxo de dados; isto pode refletir outras propriedades associadas ao programa: dificuldades de testar, número de caminhos não executáveis, dificuldade de depuração e de manutenção, entre outras. Na análise conduzida, observou-se que o número de potenciais-du-caminhos tem uma correlação bastante alta com o número de casos de teste necessários para satisfazer os critérios Potenciais Usos e com o número de caminhos não executáveis, constituindo um indicativo bastante forte da dificuldade de condução das atividades de teste. Esta medida parece ser uma medida robusta que deveria ser utilizada no estabelecimento de métricas e de modelos de previsão. Estes fatos ressaltam que os conceitos básicos que norteiam a definição de um critério de teste são extremamente importantes e que, na realidade, um critério de teste pode ser visto como uma medida de complexidade. Deve-se ressaltar que a determinação do número de potenciais-du-caminhos em um dado programa é uma função disponível na POKE-TOOL.

4 Conclusões

Apresentaram-se os resultados da análise da aplicação de um benchmark para os critérios Potenciais Usos, utilizando-se a ferramenta de teste POKE-TOOL. Com a aplicação do benchmark obtiveram-se resultados bastante interessantes; em geral, pode-se dizer que esses critérios, do ponto de vista prático, são factíveis e demandam um número de casos de teste relativamente pequeno. Foram obtidos vários modelos, bastante razoáveis, para estimar as variáveis resposta: número de casos de teste requeridos (*nct*) e número de caminhos não executáveis (*nex*); identificaram-se modelos adequados para diversas fases de desenvolvimento de software. Outra contribuição nesse sentido foi evidênciar que o uso de informações sobre as variáveis do programa são relevantes na determinação de modelos

para estimar estas variáveis respostas; essas informações devem, portanto, ser consideradas em estudos posteriores semelhantes ao apresentado neste trabalho. Um outro resultado foi ressaltar que o *número de potenciais-du-caminhos* é uma medida de complexidade robusta e que reflete características importantes do software para as atividades de teste, depuração e manutenção. Com base nessa medida, duas métricas disponíveis na POKE-TOOL foram discutidas.

Os resultados obtidos com a realização do experimento podem ser considerados bastante promissores e motivadores de novas pesquisas e estudos empíricos em torno dos Critérios Potenciais Usos, principalmente em torno da combinação de conceitos na definição de novos critérios e do estudo comparativo do custo da aplicação desses critérios e de suas adequações a classes de erros. Também motivam a condução de manutenção perfectiva na ferramenta POKE-TOOL, visando principalmente incorporar novas facilidades de suporte às atividades de teste, tais como, visualização gráfica, identificação e manipulação de caminhos não executáveis, recursos para manutenção de conjuntos de casos de teste, entre outras.

Uma vez que as hipóteses consideradas no início do experimento foram comprovadas, ou pelo menos existem fortes evidências nesta direção, este experimento está sendo repetido utilizando-se critérios funcionais para a elaboração dos conjuntos iniciais de casos de teste; um dos objetivos é analisar o grau de cobertura desses critérios em relação aos critérios Potenciais Usos. Outras áreas de aplicação serão alvo de estudos semelhantes; por exemplo, análise numérica, estatística, otimização, etc. Com um volume maior de dados caracterizar-se-ão modelos de estimativas gerais, para áreas específicas e para classes de estrutura de controle, por exemplo. Uma classe de estrutura de interesse seria a composta por estruturas similares à estrutura que maximiza o número de potenciais-du-caminhos; essa classe determinaria o que poderíamos definir de *Modelo de Estimativa Conservadora*. Outra classe seria a definida por estruturas de controle que minimizam a geração de potenciais du-caminhos, como por exemplo, estruturas de controle que contêm predominantemente laços, e definiria o que poderíamos chamar de *Modelo de Estimativa Mínima*.

Finalizando, pode se dizer que os resultados obtidos neste experimento constituem uma contribuição relevante para as atividades de desenvolvimento de software: planejamento, controle de qualidade, teste, depuração e manutenção, por exemplo. Deve-se ressaltar que o estabelecimento de critérios de teste, o desenvolvimento de ferramentas de suporte e a avaliação empírica desses critérios fornecem subsídios suficientes para que o uso deles seja rotineiro nas atividades de desenvolvimento. Como efeito secundário tem-se a introdução de conceitos, princípios e métodos de desenvolvimento de software com o consequente aumento da produtividade e qualidade dos produtos de software, a um custo reduzido.

AGRADECIMENTOS[3]

Ao Professor Paulo Cesar Masiero pelos comentários e sugestões que levaram ao aprimoramento do trabalho. Aos Professores Jorge Alberto Achcar, Josemar Rodrigues e Mariano Martinez Espinosa pelo auxílio e sugestões na análise dos resultados. À Luisa A. Spadacini Lacra pelo auxílio na elaboração das tabelas.

[3]Os possíveis erros contidos neste trabalho devem ser atribuídos exclusivamente aos autores.

A Conceitos Básicos

Um programa é representado por um grafo de fluxo de controle (grafo de programa) $G(N,A,s)$, onde N é o conjunto de nós, A o conjunto de arcos e s o nó inicial. Aos nós são associados blocos de comandos e aos arcos (ramos) possíveis transferências de controle entre os blocos.

Define-se: *caminho* como sendo uma sequência finita de nós $(n_i,\ldots,n_k)$, $k \geq 2$, tal que existe um arco de n_i para n_{i+1}, $i = 1,2,\ldots,k-1$; *caminho simples* como um caminho onde todos nós, exceto possivelmente o primeiro e o último, sejam distintos; *caminho livre de laço* como um caminho com todos os nós distintos; e *caminho completo* onde o nó inicial é um nó de entrada e o nó final é um nó de saída do grafo G.

Conforme o modelo de dados utilizado [MAL91b, CHA91], em geral, uma *definição de variável* ocorre quando um valor é armazenado em uma posição de memória. A ocorrência de uma variável é um *uso* quando a referência a essa variável não estiver sendo definida.

O grafo de fluxo de controle estendido pela associação a cada nó i do conjunto de variáveis definidas em i ($defg(i)$) é denominado *grafo def*.

Um caminho $(i,n_1,\ldots,n_m,j)$, $m \geq 0$ que não contenha definição de uma variável nos nós $n_1,\ldots,n_m$ é chamado de *caminho livre de definição* com respeito a (c.r.a) x do nó i ao nó j e do nó i ao arco (n_m,j). Um caminho livre de definição $(n_1,n_2,\ldots,n_j,n_k)$ onde o caminho $(n_1,n_2,\ldots,n_j)$ é um caminho livre de laço e n_1 tem uma definição de x, é denominado *potencial-du-caminho c.r.a x*.

Um caminho $\pi_1 = (i_1,\ldots,i_k)$ é dito estar incluído em um conjunto Π de caminhos se Π contém um caminho $\pi_2 = (n_1,n_2,\ldots,n_m)$ tal que $i_1 = n_j, i_2 = n_{j+1},\ldots,i_k = n_{j+k-1}$, para algum j, $1 \leq j \leq m-k+1$. Então, π_1 está incluído em π_2 ou que π_1 é um sub-caminho de π_2.

Os critérios potenciais usos – todos-potenciais-usos, todos-potenciais-usos/du e todos-potenciais-du-caminhos – requerem basicamente que caminhos livres de definição, em relação a qualquer nó i que possua definição de variável e a qualquer variável x definida em i, sejam executados, independentemente de ocorrer uso dessa variável nesses caminhos. Por exemplo, seja Π um conjunto de caminhos completos:

Critério Todos-potenciais-du-caminhos – Π satisfaz o critério todos-potenciais-du-caminhos se para todo nó i e para toda variável x para a qual existe uma definição em i, Π inclui todos *potenciais du-caminhos* c.r.a x em relação ao nó i.

Referências:

[ACH90] J.A. Achcar e J. Rodrigues, " Introdução de Estatística para Ciências e Tecnologia", ICMSC/USP – São Carlos, SP, Brasil, 1990.

[BEI90] B. Beizer, Correspondência Pessoal, 1990.

[BIE89] J. Bieman e J. Schultz, "Estimating the Number of Test Cases Required to Satisfy The All-du-paths Testing Criterion", in *Proc. of the ACM SIGSOFT'89 - Third Symposium on Software Testing, Analysis and Verification (TAV3)*, Flórida, USA, Dec, 1990.

[CLA85] L. Clarke, A. Podgurski, D. J. Richardson e S. J. Zeil, "A Comparison of Data Flow Path Selection Criteria," in *Proc. of the 8th Int'l Conf. on Software Engineering*, pp. 244-251, Aug. 1985.

[CHA91] M.L. Chaim, "POKE-TOOL – Uma Ferramenta para Suporte ao Teste Estrutural de Programas Baseado em Análise de Fluxo de Dados," Tese de Mestrado, DCA/FEE/UNICAMP - Campinas, SP, Brasil, Abril 1991.

[DEN82] C.R. Dennis e W. Sanford, *Residuals and Influence in Regression*, New York: Chapman and Hall, 1982.

[KER81] B. W. Kernighan e P. J. Plauger, *Software Tools in Pascal*, Massachusetts: Addison-Wesley Publishing Company, Reading, 1981.

[MAL88] J. C. Maldonado, M. L. Chaim, M. Jino, "Seleção de Casos de Testes Baseada nos Critérios Potenciais Usos", in *Proc. II Simpósio Brasileiro de Engenharia de Software*, Canela, RS, Brasil, pp. 24-35, Oct. 1988.

[MAL89] J. C. Maldonado, M. L. Chaim, M. Jino, "Arquitetura de uma Ferramenta de Teste de Apoio aos Critérios Potenciais Usos ", *Proc. XXII Congresso Nacional de Informática* São Paulo, SP, Brasil, Sept. 1989.

[MAL91b] J. C. Maldonado, Tese de Doutorado, "Critérios Potenciais Usos: Uma contribuição ao Teste Estrutural de Software", DCA/FEE/UNICAMP - Campinas, SP, Brasil, Junho, 1991.

[McC76] T. J. McCabe, "A Complexity Measure," *IEEE Trans. Software Eng.*, Vol. SE - 2, DEC. 1976.

[NTA88] S. C. Ntafos, "A Comparison of Some Structural Testing Strategies," *IEEE Trans. Software Eng.*, Vol. 14, No. 6, pp. 868-873, Jun. 1988.

[RAP85] S. Rapps e E. J. Weyuker, "Selecting Software Test Data Using Data Flow Information," *IEEE Trans. Software Eng.*, Vol. SE - 11, pp. 367-375, Apr. 1985.

[RYA76 T. A. Ryan,Jr. e B.L. Joiner e B. F.Ryan, *MINITAB – Student Handbook*, California: Wadsworth Publishing Company, 1976.

[VER92] S. R. Vergílio, Tese de Mestrado, DCA/FEE/UNICAMP - Campinas, SP, Brasil, Jan 1992.

[WEY90] E. J. Weyuker, "The Cost of Data Flow Testing: An Empirical Study," *IEEE Trans. on Software Eng.*, Vol. SE - 16, No. 2, pp. 121-128, Feb. 1990.