COMPARANDO METRICAS DE COMPLEXIDADE DE PROGRAMAS

Ana Price
Universidade Federal do Rio Grande do Sul
Porto Alegre - *Brasil*

Introdução

Na avaliação de qualidade de um produto de software, um dos atributos mais freqüentes mencionados é a complexidade do fluxo de controle do sistema. Nos últimos anos várias tentativas tem sido feitas para derivar medidas a partir de programas-fonte que quantifiquem a noção de complexidade de programa.

O que vem a ser complexidade de programa? Encontramos na literatura várias definições para o termo, sendo que a maioria poderia ser incluída na seguinte:

"Complexidade de um programa é a medida da dificuldade de entender e trabalhar (modificar, depurar, testar) com o programa.

A medida de complexidade de um programa é importante por várias razões: (1) permite-nos estimar o esforço, tempo e custo de manutenção do programa; (2) permite-nos identificar pontos em que o programa poderá apresentar problemas e portanto re-estruturá-lo; (3) fazer comparações entre programas distintos que implementem a mesma função; e também, (4) fazer previsões do número de erros do programa.

Existem vários fatores que tem influência direta na manutenção e teste de programas, como por exemplo, o próprio tamanho do programa, a complexidade das estruturas de dados utilizadas, o fluxo de dados, o nível de embutimento das estruturas de controle, etc. Dentre as várias métricas desenvolvidas para avaliar a complexidade de programas, a complexidade de fluxo de controle e o aspecto de software mais freqüentemente medido.

Entre as métricas mais difundidas encontram-se "Software Science"" /Halstead 77/ e a Complexidade Ciclomática /McCabe 76/. A métrica de Halstead constitui-se num refinamento da medida de tamanho de programa pela contagem de linhas de código. A complexidade medida por Software Science baseia-se no número de operandos (variáveis e constantes) e operadores (aritméticos, lógicos, palavras-chave e delimitadores) que aparecem no programa. Software Science inclui também medidas quantitativas de nível de programa e de linguagem, e efeito de modularização. Prediz o tamanho de progra-

mas e estima o tempo que um programador de nível médio leva para implementar um dado algoritmo. Estudos estatísticos mostraram fortes correlações entre as predicações da teoria e medidas reais de tempo de programação e número médio de programas /Fitzsimmons 78/. Entretanto, existem críticas de que os programas usados para validar a teoria eram muito pequenos e que a base (contagem de operandos e operadores) em que Software Science se fundamenta é fraca devido a ambigüidades relativas a "o que" deve ser contado e "como" contá-lo |Shen 83|.

A métrica de McCabe define como medida de complexidade de programa o número ciclomático de um grafo que representa o fluxo de execução do programa. Segundo McCabe, a complexidade de um programa é independente de seu tamanho (número de comandos), mas depende unicamente de seu fluxo de controle. Quanto maior o número de ciclos no programa, maior a sua complexidade.

O objetivo deste trabalho é apresentar algumas métricas de complexidade de fluxo e compará-las quanto a dois aspectos relevantes da complexidade de programas: a estrutura do programa e o nível de embutimento de predicados de controle. As métricas a serem estudadas compreendem: Complexidade Ciclomática |McCabe 76|, Scope Ratio |Harrison 81|, Complexidade de Expressões Regulares |Magel 81| e Nesting Level |Piwowarski 82|.

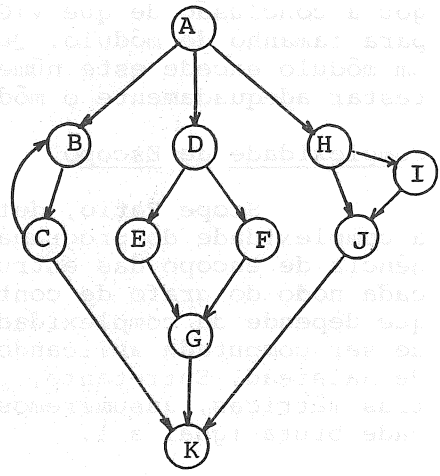
Medindo a Complexidade de Fluxo de Controle

A maioria dos trabalhos desenvolvidos sobre complexidade de software se relaciona com os efeitos do fluxo de controle sobre a complexidade de programas. Para explicar a importância do fluxo de controle com relação a complexidade, consideremos, por exemplo, um programa de 30 linhas com 15 comandos IF-THEN-ELSE. Este programa poderá conter acima de 30 mil caminhos de execução, e é obvio que tal configuração certamente será difícil de ser completamente compreendida.

As estratégias de medida de complexidade do fluxo de controle, usualmente, representam o programa como um grafo dirigido a fim de mostrar a topologia do fluxo de execução do programa. Um grafo dirigido G = (V,E) consiste de um conjunto de nodos V e um conjunto de arcos dirigidos E conectando os nodos. Num grafo de fluxo cada nodo representa um bloco de código seqüencial e os arcos correspondem ao fluxo de controle entre os vários nodos. Assumiremos que um grafo de fluxo de programa contém apenas um nodo inicial e um único nodo final, e as propriedades de que todo nodo é acessível a partir do nodo inicial, e por sua vez, o nodo final e acessível a partir de qualquer outro nodo do grafo. O grafo dirigido G1 ilustrado na Figura 2 representa o fluxo de controle do programa mostrado na Figura 1.

```
BEGIN
   CASE   exp  OF
          C1 : REPEAT  S1
               UNTIL  P1;
          C2 : BEGIN
                 IF  P2
                 THEN  S2
                 ELSE  S3;
                 S4;
               END;
          C3 : BEGIN
                 IF  P3
                 THEN S5
                 S6
               END;
    END;
  S7;
END;
```

Programa Exemplo - Figura 1



Grafo G1 - Figura 2

Complexidade Ciclomática de McCabe

McCabe afirma que a complexidade ciclomática de um grafo de programa é aplicável na determinação da dificuldade de depurar e testar o programa. O número ciclomático V(G) de um grafo dirigido G fortemente conexo é igual ao número de circuitos linearmente independentes. Num programa estruturado a complexidade ciclomática é igual ao número de condições acrescido de 1. V(G) também pode ser obtido pela fórmula:

$$V(G) = m - n + p$$

onde m = número de arcos de G, n = número de nodos de G, e

P = número de componentes individuais.

Calculando-se a complexidade ciclomática do programa exemplo (Figura 1) representado pelo grafo G1, obtemos:

$$V(G)=16 - 11 + 1 = 6$$

(Foi considerado um arco do nodo K para o nodo A a fim de tornar o grafo fortemente conexo).

Como o número ciclomático cresce com o número de caminhos de decisão e ciclos, a métrica de McCabe fornece uma medida quantitativa da dificuldade de testar um programa. Estudos experimentais indicaram relações entre a métrica de McCabe e o número de erros existentes no código fonte do programa, bem como o tempo requerido para encontrar e corrigir tais erros |Pressman 82|.

McCabe assegura que V(G) pode ser usado para fornecer uma indicação quantitativa do tamanho máximo de um módulo. A partir de dados de vários projetos reais, McCabe che

gou a conclusão de que V(G)=1Ø parece ser um limite superior para tamanho de módulo. Quando a complexidade ciclomática de um módulo excede este número, torna-se extremamente difícil testar adequadamente o módulo.

Complexidade de Escopo

Scope Ratio, definida por Harrison e Magel, deriva a complexidade do programa a partir de uma análise da abrangência de escopo das estruturas de controle. Nesta métrica, cada nodo do grafo de controle tem uma complexidade "bruta" que depende da complexidade dos comandos do nodo, e que pode ser computada aplicando-se, por exemplo, Software Science de Halstead. Entretanto, para efeitos de comparação com outras métricas, assumiremos que todos os nodos tem complexidade bruta igual a 1.

Scope Ratio considera que o grafo de um programa tem dois tipos de nodos: nodo seletor, que tem dois ou mais arcos partindo de si próprio, e nodo receptor, com apenas um arco de partida. Para obter a medida de escopo, criamos um subgrafo formado por todos os nodos que sucedem a um nodo seletor. Este subgrafo tem no mínimo um nodo, chamado "limite inferior", o qual está sobre o caminho de todos os nodos. Aquele limite inferior que precede a todos os outros limites inferiores do subgrafo é chamado "limite inferior maior" (LIM). As complexidades brutas de todos os nodos sobre os caminhos que levam ao LIM (excluindo-se a deste último) são somadas e adicionadas a complexidade bruta do nodo seletor obtendo-se a complexidade ajustada do nodo.

Este processo é repetido para cada nodo seletor do grafo de controle. A complexidade ajustada de um nodo receptor é simplesmente a sua complexidade bruta. As complexidades ajustadas de todos os nodos são então somadas, e esta soma, chamada Número de Escopo, é a complexidade do programa como um todo.

Como foi atribuída a todo nodo uma complexidade bruta de valor igual a 1, a complexidade real do programa pode ficar deturpada. Por esta razão, os autores introduziram uma medida chamada "Razão de Escopo", que é a razão entre o número de nodos do programa e o Número de Escopo. A medida que a magnitude da Razão de Escopo decresce, a complexidade do programa cresce. Por exemplo, um programa com uma razão de escopo menor do que Ø.4 é considerado razoavelmente complexo.

A tabela a seguir é o resultado da aplicação de Número e Razão de Escopo sobre o grafo do programa exemplo (Figura 2).

Seletores	Escopo	LIM	Complexidade
A	B C D E F G H I J	K	1Ø
C	B C	K	3
D	E F	G	3
H	I	J	2
			18

Receptores	Complexidade
B E F G I J K	7

Número de Escopo (NE) = 18 + 7 = 25
Razão de Escopo (RE) = 11/25 = Ø.44

Métrica de Piwowarski

Piwowarki alega que existem três aspectos segundo os quais as métricas existentes, em geral, não abragem as noções intuitivas de complexidade: (1) um programa estruturado é menos complexo que uma versão não estruturada do programa; (2) estruturas de controle embutidas são mais complexas que estruturas seqüencias de controle; e (3) um comando CASE com N alternativas é menos complexo que N-1 comandos IF embutidos |Piwowarski 82|.

"Nesting Level Complexity Measure" propõe a seguinte medida:

$$NL = V^*(G) + \sum_i P(i)$$

onde V*(G) é a complexidade ciclomática ajustada com estruturas CASE tratadas como predicados únicos. P(i) é a profundidade de embutimento do i-ésimo predicado, e tem como base o conceito de escopo de predicado definido em |Harrison 81|. A profundidade (nível) de embutimento do predicado i é o número de escopos de predicado sobrepostos ou contidos pelo i-ésimo nodo-predicado. Qualquer nodo acessível a partir do nodo-predicado, excluindo-se o "limite inferior maior" e os nodos seguintes, está dentro do escopo do nodo-predicado. A métrica de Piwowarski aplicada ao grafo da Figura 2 resultou em:

Predicados	Escopo	Complexidade	
A	C D H	3	O nodo A é o único nodo-predicado que contém outros nodos de controle.
C	-	Ø	
D	-	Ø	
H	-	Ø	
V(G) = 6	NL = 6 + 3 = 9		

Métrica de Magel

Segundo Magel, expressões regulares podem ser usadas para observar a complexidade de seqüencias possíveis de execução de um programa |Magel 81|.

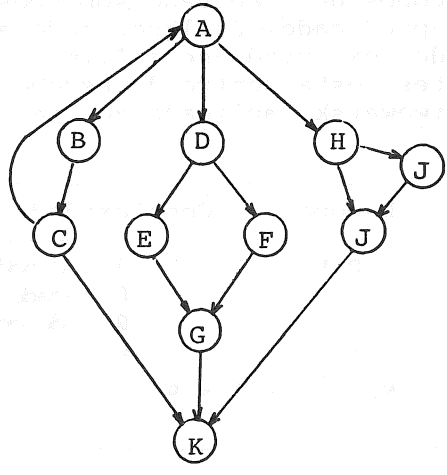
A métrica de Magel utiliza o operador de fechamento de Kleene "*" para indicar iteração e o operador de união "+" para indicar seleção. Por exemplo, a expressão regular que representa todas as seqüencias possíveis de execução para o programa exemplo da Figura 1 é a seguinte:

A (BC (BC) * + D (E+F) G + H (IJ+J)) K

A medida de complexidade de uma expressão regular é computada pela contagem do número de símbolos (operandos, operadores e parenteses) na expressão regular equivalente minimamente parentizada. Por exemplo, A(BC(BC)*+D(E+F)G+H(IJ+J))K tem complexidade 27 (14 operandos, 1 operador "*" e 4 operadores "+", e oito parenteses).

Programa Estruturado vs Não-Estruturado

Usualmente, programas estruturados (formados a partir das construções básicas Seqüencia, Seleção e Iteração) são considerados menos complexos do que programas equivalentes contendo GO-TO's. Como um dos objetivos deste trabalho é comparar as métricas apresentadas quanto a avaliação da estrutura do programa, vamos fazer uma modificação no grafo G1 (Figura 2) a fim de alterar a sua estrutura. Vamos substituir o arco CB pelo arco CA (Figura 3). Assim, ao invés do comando REPEAT-UNTIL, representado pelo ciclo entre os nodos B e C, temos agora um bloco de comandos (nodo B) seguido por um comando seletivo (C) no qual um dos arcos é um desvio incondicional (arco CA). A modificação realizada supõe a adição de um comando GO-TO ao grafo G1.



Grafo G2 - Figura 3

Aplicando-se as métricas descritas acima ao grafo modificado obtemos a seguinte tabela comparativa.

	V(G)	NE	RE	NL	ER
G1 estruturado	6	25	Ø.44	9	27
G2 contendo GO-TO	6	34	Ø.32	12	28

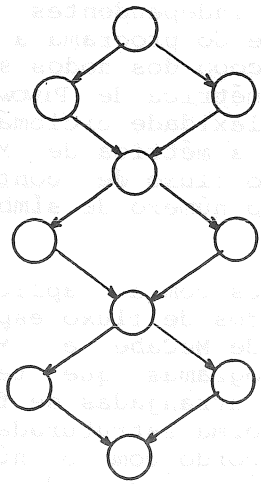
Observando-se a tabela, vemos que a complexidade ciclomática não levou em conta a modificação na estrutura do programa. Por outro lado, as métricas de escopo e de nível de embutimento acusaram um aumento considerável de complexidade com a alteração do programa. O nodo C teve sua complexidade aumentada de 3 para 11 no cálculo da Razão de Escopo pois seu escopo passou a englobar todos os nodos do grafo exceto o nodo K. No caso da métrica de Nível de Embutimento, a complexidade do nodo C cresce de Ø para 3. Por último a Métrica de Magel resultou na expressão regular

A (BCA) * (BC+D (E+F) G+H (IJ+J)) K

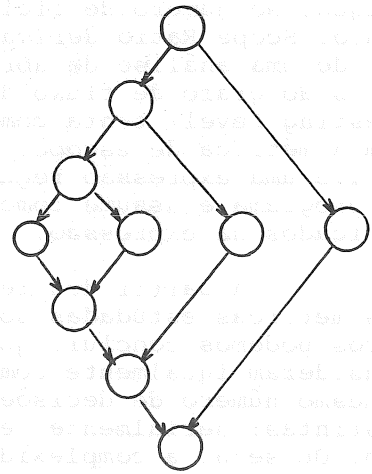
ligeiramente mais complexa que a anterior.

Embutimento de Estruturas de Controle

O embutimento de estruturas de controle é considerado por vários autores como fator incremental na complexidade de programas: quanto maior o embutimento de precidados, maior é o grau de complexidade do programa |Harrison 81, Piwowarski 82|.



Grafo G3



Grafo G4

Figura 4

Os grafos de fluxo G3 e G4 poderiam representar dois programas diferentes com blocos de comandos sequenciais identicos porém com as estruturas de controle arranjadas diferentemente. Estes arranjos podem afetar consideravelmente a complexidade dos programas. No programa representado pelo grafo G3 as decisões são seriais enquanto que no outro temos dois níveis de embutimento de predicados. A tabela abaixo nos mostra as complexidades calculadas para os dois grafos:

	V(G)	NE	RE	NL	RE
G3	4	16	Ø.62	4	19
G4	4	25	Ø.40	7	19

Da tabela podemos inferir que ambas as métricas de Complexidade Ciclomática e Expressões Regulares consideram igualmente complexos os dois programas. Isto é, consideram que o embutimento de estruturas de controle não altera a complexidade do programa. As métricas de escopo e nível de embutimento acusaram um aumento de complexidade considerável.

Conclusões

As métricas de complexidade de fluxo de controle de programas trabalham, geralmente, sobre grafos dirigidos os quais representam fluxos de controle de programas.

Foram descritas quatro métricas de complexidade: Complexidade Ciclomática de McCabe, Scope Ratio de Harrison e Magel, Nesting Level de Piwowarski e a Métrica de Magel sobre Expressões Regulares. McCabe define como medida de complexidade, o número ciclomático do grafo do programa, o qual é igual ao número de ciclos linearmente independentes no grafo. Scope Ratio deriva a complexidade do programa a partir de uma análise de abrangência de escopo dos nodos seletores do grafo de fluxo de controle. A métrica de Piwowarski (Nesting Level) tenta combinar a complexidade ciclomática com a métrica de escopo. E finalmente, a métrica de Magel deriva uma expressão regular a partir do fluxo de controle do programa e assume como complexidade o número de símbolos indicados na expressão.

A partir dos resultados obtidos com a aplicação das métricas estudadas sobre quatro grafos de fluxo específicos podemos concluir que as métricas de McCabe e Magel consideram igualmente complexos dois programas que tenham o mesmo número de decisões. porém estas arranjadas de formas distintas: serialmente, embutidas, de forma estruturada ou não. Ou seja, a complexidade varia de acordo com o número de decisões não importanto como estas são posicionadas no fluxo de controle do programa. Por outro lado, as métricas de escopo e nível de embutimento acusarem um acréscimo considerável na complexidade dos programas quando da inserção

de GO-TO e embutimento de estruturas de controle.

As métricas de fluxo de controle, tais como quaisquer outras classes de medidas de complexidade, são falhas porque não levam em conta qualquer outro fator que não seja a complexidade do fluxo de execução do programa. Entretanto, elas fazem um bom trabalho ao diferenciar dois programas que do contrário seriam equivalentes em outras características, tal como o tamanho.

Agradecimentos

A autora agradece os incentivos recebidos da FINEP e CNPq.

Referências Bibliográficas

|Fitzsimmons 78|
A. Fitzsimmons e T. Love, "A Review and Evaluation of Software Science", ACM Computing Surveys, Março 1978, pag. 3-18.

|Halstead 77|
M. Halstead, "Elements of Software Science", Elsevier North-Holland, Inc., New York, 1977.

|Harrison 81|
W. Harrison e K. Magel, "A Complexity Measure based on Nesting Level", ACM SIGPLAN Notices, Vol. 6, Nº.3 , Março 1981, pag. 63-74.

|Harrison 82|
W. Harrison, K. Magel, R. Kluczny e A. DeKock, "Applying Software Complexity Metrics to Program Maintenance, Computer, Setembro 1982, pag. 65-79.

|Magel 81|
K. Magel, "Regular Expressions in a Program Complexity Metric", ACM SIGPLAN Notices, Julho 1981, pag. 61-65.

|McCabe 76|
T. McCabe, "A Complexity Measure", IEEE Transactions on Software Engineering, Dezembro 1976, pag. 308-320.

|Piwowarski 82|
P. Piwowarski, "A Nesting Level Complexity Measure", ACM SIGPLAN Notices, Setembro 1982, pag 44-50.

|Pressman 82|
R. Pressman, "Software Engineering: A Practitioner's Approach", McGraw-Hill, Inc., 1982.

|Shen 83|
V. Shen, D. Samuel e H. Dunsmore, "Software Science Revisited: A Critical Analysis of the Theory and Its Empirical Support", Março 1983, pag. 155-165.