

6 T-5

プログラムのわかりやすさを表すメトリクスの一提案

飯尾和彦 三宅武司 古山恒夫

NTTソフトウェア研究所

1 はじめに

ソフトウェアシステムの保守にかかるコストは、開発のライフサイクル全体の50%から70%に相当するといわれている。したがって、開発のコストを削減するためには、保守性(Maintainability)を考慮してソフトウェアを設計する必要がある。

一方、実際の保守作業においては、保守担当者はドキュメントとソースプログラムの解説に最も時間をかけており、特にソースプログラムの内容を理解するのに最も苦勞している。その時、プログラム全体とというよりは、着目するモジュール単位で、保守箇所を理解することが多いため、モジュールがそれぞれ独立してわかりやすく設計されている必要がある。

モジュールのわかりやすさを表すメトリクスには、従来から多くのものが提案されているが、グローバル変数に着目したものはない。本論文では、グローバル変数を用いたモジュールのわかりやすさを表すメトリクスを提案し、実験によりその妥当性を確認したので報告する。

2 既存のメトリクス

これまでに提案されたモジュールのわかりやすさに関連する代表的なメトリクスは次の四つである。

(1) サイクロマチック数^[1]

プログラム内の制御パスの点、辺の数に着目し、グラフ理論に基づき複雑さのメトリクスを導出している。

(2) 内部データ量

モジュールの内部で使われているデータの数。データ量が多いほど複雑でわかりにくいと考えられる。

(3) Fanout数^{[2][3]}

モジュールの呼び出しの数。呼び出し数が多いほど、モジュールの処理は複雑であると考えられる。

(4) モジュール間データ量^[3]

パラメータの受渡しでやりとりされるデータの量。

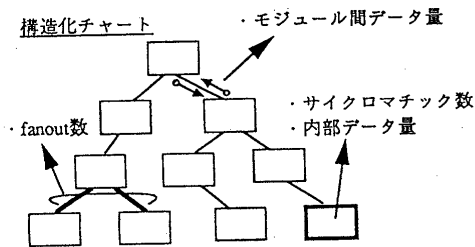


図1. わかりやすさを表すメトリクス

*Program Understandability Metrics based on Global Variables
Kazuhiko IIO, Takeshi MIYAKE, Tsuneo FURUYAMA
NTT Software Laboratories

3 グローバル変数のメトリクス

今までに、モジュールのわかりやすさに関して、2章のようなメトリクスが提案されているが、グローバル変数に着目したものはない。しかし、グローバル変数は、あるモジュールを理解する時に、調べにいかなくてはならない周囲のモジュールの量に影響する主要な指標になると思われる。そこで、グローバル変数に関する次の5つのメトリクスを考える。

3.1 5種類のメトリクスの提案

1つのモジュールを理解する上でのグローバル変数に基づいたわかりやすさを表すメトリクスとして、次の(1)~(5)を提案する。それぞれのメトリクスは、わかりやすさが次のことに依存していることを意味している。

(A) 関連するグローバル変数の種類数または個数に依存

(1)UG:the number of Unique Global variables

グローバル変数の種類数

(2)NG:the Number of modules which includes the Global variables

グローバル変数を含むモジュールの個数

(B) 関連するグローバル変数の種類または個数と距離に依存

(3)DG:the sum of the Distances between modules

which includes the Global variables

グローバル変数を含むモジュールまでの距離の総和

(4)ADG:the sum of the Average Distances between modules which includes the Global variables

グローバル変数ごとの平均距離の総和

(あるモジュールを理解するのに、全てのグローバル変数についてそれぞれ一回だけ調べに行くが、各グローバル変数についてどれを調べに行くかは必ずしも定まっていないことを意味する。)

(5)MDG:the sum of the Modified average Distances between modules which includes the Global variables

グローバル変数ごとの重みづけ平均距離の総和

(最も近いモジュールに含まれる変数は必ず調べに行き、他の変数については3個に1個の割合で調べに行くことを意味する。)

3.2 メトリクスの定義

3.1節で示したメトリクスについての厳密な定義を行なう。構造化チャートにおいて、あるモジュールに着目した時に、そのモジュールに含まれるグローバル変数の量と距離に関連したメトリクスを提案する。

構造化チャートをグラフGとし、各モジュールを点、呼び出し関係を距離1の枝と考える。なお、呼び出しは一方だけのデータのやり取りではないので、枝の方向はないものと仮定する。

あるモジュール $M_i (i = 1 \dots n; n$ はモジュール数) に着目したときに、 M_i に含まれるグローバル変数を $g_1, g_2 \dots g_j \dots g_N$ とする。そのとき、

$$UG_i := N_i \quad (1)$$

と定義する。また、プログラム中の全モジュール M のうち、 g_j を含むモジュールの個数を P_j として、 g_j を含むモジュールの集合を

$$\{M_{g_j,k} | M_{g_j,k} \in M\} \quad k = 1, \dots, P_j$$

と定義する。ここで、 M_i, M_j 間の距離を

$$d(M_i, M_j) (i = 1 \dots n, j = 1 \dots n)$$

とすると、

$$MD_i := \sum_j \min\{d(M_i, M_{g_j,k}) | 1 \leq k \leq P_j, i: fix, j: fix\}$$

$$NG_i := \sum_j P_j \quad (2)$$

$$DG_i := \sum_{j,k} d(M_i, M_{g_j,k}) \quad (3)$$

$$ADG_i := \sum_{j,k} \frac{d(M_i, M_{g_j,k})}{P_j} \quad (4)$$

$$MDG_i := MD_i + \frac{\sum_{j,k} d(M_i, M_{g_j,k}) - MD_i}{.3} = \frac{\sum_{j,k} d(M_i, M_{g_j,k}) + 2MD_i}{3} \quad (5)$$

と定義することができる。

この5種類のメトリクスの値はいずれも、モジュールごとに算出可能である。

構造化チャート

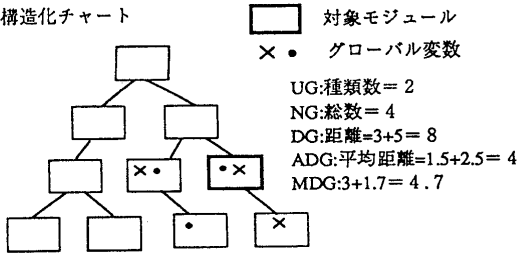


図2. 尺度の計測例

4 評価実験

従来のメトリクスおよび前章で提示した5つのメトリクスと、モジュールのわかりやすさの関係を実験により確認する。

4.1 実験方法

(1) 対象プログラム

550step から 700step 程度の C 言語で書かれた 4 ケースのアプリケーションプログラムを実験に用いた。それぞれのプログラムの中から 40step 程度、サイクロマチック 5 程度のモジュールを、理解度を測るための対象モジュールとして選んだ。

(2) 理解度の測定方法

それぞれのモジュールの理解度を測るために、対象モジュールについて客観的に採点できる設問を合計 9 問それ

ぞれのケースごとに作成し、何分間で 9 問を正答したかを計測した。

(3) 被験者と実験回数

C 言語の経験が豊富な 3 人の被験者に対して 4 回 (4 ケース) の実験を行なった。なお、被験者には実験の主目的は知らされていない。

4.2 結果

(1) 測定結果

表1. 実験結果

		ケース1	ケース2	ケース3	ケース4	時間との相関係数
理解に要した時間(分) (注)		155.8	163.9	225	289.3	
グローバル変数	UG	4	2	5	2	0.49
	NG	6	15	11	8	0.05
	DG	11	29	16	27	0.36
	ADG	6.5	3.8	4	11.3	<u>0.72</u>
	MDG	6.3	7.7	6.7	14.3	<u>0.85</u>
内部データ量		8	11	22	4	-0.16
モジュール間データ量		2	0	4	0	-0.11
fanout		4	5	2	3	-0.66

(注) 実験時間は50分で、50分以上かかった場合は50分の時点で何問正解したかを計測し、50×正答数÷9の式を用いて時間に換算した。この換算時間をモジュールのわかりやすさを測るための代用尺度とした。

実験の結果、わかりやすさと各メトリクスの相関は表1のようになった。

(2) 評価と考察

表1から、モジュールのわかりやすさは平均距離を考慮したグローバル変数のメトリクスと最も高い相関があることがわかる。MDGの方がADGより相関が高いが、これは被験者の行動を観察した結果と合致している。

今回の実験では、グローバル変数以外のメトリクス(データ量と fanout 数)はモジュールのわかりやすさとはあまり関係がないという結果が得られた。

5 まとめ

今回、プログラムのわかりやすさを表すメトリクスとして、新たにグローバル変数に関するメトリクスを提案した。次に、プログラムをモジュール単位に理解する時には、重みをつけた平均距離である MDG が最もプログラムのわかりやすさと相関が高いことを確認した。

参考文献

- [1] T.J.McCabe, "A complexity measure", IEEE Trans. Software Eng., vol.SE-2, pp.308-320, Dec.1976.
- [2] S.Henry, "Information flow metrics for the evaluation of operating systems' structure", Ph.D., dissertation, Iowa State Univ., 1979.
- [3] David N. Card, "Measuring Software Design Quality", Prentice Hall, Englewood Cliffs, NJ, 1990.