ソフトウェア工学 32－3
（1983. 9. 28）

プログラムの複雑度の分析と評価

桑名栄二・亀田壽夫（電気通信大学）

abstract

This study aims at examining the dependency of program reliability on the complexity of program structure. For this purpose, we have developed program analyzing systems for FORTRAN and Pascal programs. These systems deal with typical complexity measures such as cyclomatic number, and with several factors of program complexity such as the number of global variables and the number of called modules.

This report presents the correlation coefficients among complexity measures, and the correlation coefficients between complexity measures and program errors. These data show that there is strong correlation among complexity measures, and that program reliability depends on program control structure and global data usage.

1. はじめに

プログラムの誤りの発生、発見に対して、そのプログラムの構造、制御の流れ、データの流れは、大きな影響をおよぼす。現在までに、プログラムの信頼性を、

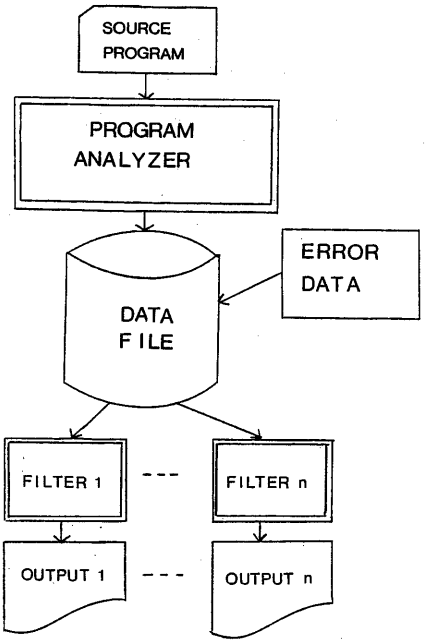
（1）プログラムの構造の複雑さから評価する。
（2）プログラム誤りの発生率から信頼度成長モデルを開発し、評価する。

という種々の研究がなされてきた。
（2）の方法は、発見誤り数を用いた確率的モデルから、信頼性評価尺度として、プログラムの残存誤り数、プログラム信頼度を推測しようとするものである。これに対して（1）の方法は、プログラムのソーステキストから得られる情報により、静的にプログラム複雑度を求め、信頼性を評価するものである。プログラム複雑度として、現在までに、グラフ理論から得られたMcCabeのCyclomatic number[1]、プログラム開発におけるeffortをプログラム内の、オペランド数、オペレータ数から、経験に基づいて求めた、Halsteadの Software effort[2]、プログラムの制御構造のネスティング・レベルに着目したHarrisonらのScope metric[3]、ChenのMIN[4]など、種々の複雑度が提案された。さらに、発見誤り数と種々の複雑度との間の関係を分析することにより、複雑度の評価も行なわれてきた。[5),6)]

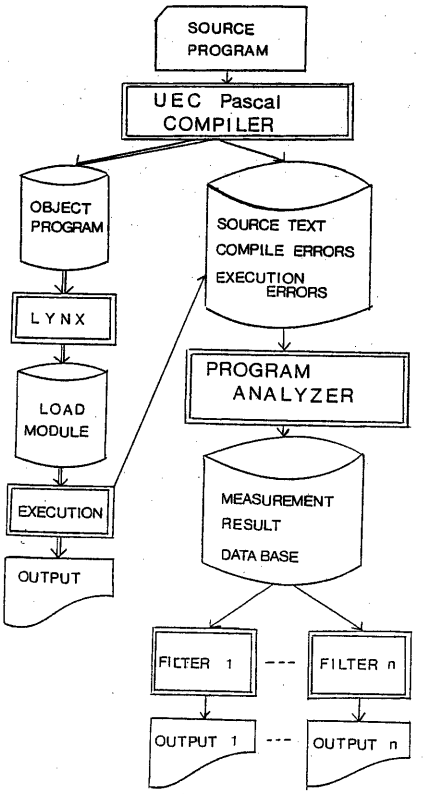
本研究では、プログラムの構造の複雑さの面から、FORTRANプログラム、Pascalプログラムから得られるいくつかのプログラム複雑度をとり上げ、自動的に測定するシステムを作成した。FORTRANプログラム用システム（以下、PRANFOR* と記す）、Pascalプログラム用システム（以下、PRANPAS* と記す）は、各々プログラムの複雑さの要因、尺度として、3節に示すものをとり上げた。そしてPRANFOR, PRANPASを用いて得られた複雑度間の相関から、複雑度間の関係の考察を行なった。さらに、PRANFORから得られたデータはプログラム開発過程を3つのフェイズに分け、各フェイズにおいて得られた誤りとの相関を求め、PRANPASは、コンパイル過程において発生した誤りとの相関を求めることにより、誤り発生の要因について評価した。

2. 測定システム

PRANFOR, PRANPASは、(Fig.-1), (Fig.-2)に示す方式により多尺度の評価を行なうものである。PRANFORの誤り情報は、システムの外から入力するのに対し、PRANPASは、コンパイラが自動的に、誤り情報と、その時のソーステキストをデータファイルに書き出し、それを分析するという違いがある。



[Fig.1] PRANFOR



[Fig.2] PRANPAS

* PRANFOR (Program Analyzer for FORTRAN program)
** PRANPAS (Program Analyzer for Pascal program)

3. プログラム複雑度

PRANFOR, PRANPASでとり上げたプログラム複雑度を(Table-1)(Table-2)に示す。ここでとり上げた尺度の中で、cyclomatic number ($v(G)$)[1], software effort[2]は、提案されている尺度の中で、代表的な尺度である。cyclomatic numberは、プログラム内の制御構造に着目し、フローグラフ(graph G)化したときのnodeとedgeの数を各々、n, eとしたとき、次のように定義される。

$$v(G) = e - n + 2 \qquad (3.1)$$

さらに$v(G)$は、プログラム内の分岐構造を作る文の数(ds)により、次のように簡単に計算できることもわかっている[1]。

$$v(G) = ds + 1 \qquad (3.2)$$

software effortは、プログラム内のユニークなオペレータ数、オペランド数をn_1, n_2, オペレータ総数、オペランド総数をN_1, N_2とするとき、次の、program volume, program difficultyを定義し、(3.5)式で得られる

尺度である。

$$\text{program volume}(V) = (N_1 + N_2)\log_2(n_1 + n_2) \quad (3.3)$$

$$\text{program difficulty}(D) = n_1 N_2 / 2n_2 \quad (3.4)$$

$$\text{software effort}(E) = V \cdot D = n_1 N_2 (N_1 + N_2)\log_2(n_1 + n_2)/2n_2 \quad (3.5)$$

この他にPRANFORでは、分岐構造を作る文毎の出現頻度を要因としてとり上げ、PRANPASではプログラムの内部変数の使用に関する複雑度（または、モジュール毎のlocalityを示す尺度）として、global dataの使用数や、モジュール呼び出しに関する複雑度としてcallしたモジュール数、さらに制御構造の複雑さとして、ネステング・レベルの深さをとり上げた。

[Table 1] Complexity measures (PRANFOR)

F0 -- no. of executable statements
F1 -- cyclomatic number
F2 -- no. of sequences
F3 -- no. of 2-way branches
F4 -- no. of 3-way branches
F5 -- no. of more than 4-way branches
F6 -- no. of IF & GOTO statements
F7 -- no. of type declaration statements
F8 -- no. of COMMON or EQUIVALENCE statements

[Table 2] Complexity measures (PRANPAS)

P0 -- no. of delared labels
P1 -- max depth of nesting level(control flow)
P2 -- cyclomatic number
P3 -- cyclomatic number + no.of operators in decision st.
P4 -- software effort
P5 -- program volume
P6 -- program difficulty
P7 -- no. of executable statements
P8 -- no. of global variables
P9 -- no. of global variable references
P10-- no. of called modules
P11-- total no. of called modules

4. プログラムの誤り

PRANFORで集めたプログラム誤りは、ある小規模なプログラムの開発過程で発生したものである。誤りとプログラム複雑度を比較するにあたり、従来の研究では、単にモジュール毎の発生誤りの絶数（または、誤り発生率）のみにより、比較判断が行なわれてきたようである。しかし、誤りの絶数のみで比較するばかりでなく、その発生した時期・性質により分類し、比較するのが適当であると考える。[8] そこで今回、プログラムの開発において発生した誤りの時期から、compile phase, testing phase, maintenance phaseの3つのフェイズ分けを行なった。さらに、比較的多くの誤りの見つかったtesting phaseをその性質から7つに分類した（Table-3, Table-7）。

5. 分析結果

（Table-4）に示すテスト・プログラムを用いて、PRANFOR, PRANPASで分析した結果について示す。

[Table 3] Program errors

1. testing phase errors
   1. logical errors
   2. data structure errors
   3. data definition and reference errors
   5. module interface errors
2. compile phase errors
3. maintenance phase errors

[Table 4] Test programs

1. test1*	(program analyzer)	1500 steps
2. test2	(BENCHMARK)	3500 steps
3. test3	(FORDAP system)	1500 steps
4. test4	(SIMTRAN)	1100 steps
5. test5*	(utility)	600 steps
6. test6**	(program analyzer)	6000 steps

1~5:FORTRAN program
6:Pascal program
*:analyze software errors
**:analyze compile errors

5.1 複雑度間の相関

[Table 5] Correlation coefficients among complexity measures (PRANFOR)

F0 -- no. of executable statements
F1 -- cyclomatic number
F2 -- no. of sequences
F3 -- no. of 2-way branches
F4 -- no. of 3-way branches
F5 -- no. of more than 4-way branches
F6 -- no. of IF & GOTO statements
F7 -- no. of type declaration st

	F0	F1	F2	F3	F4	F5	F6	F7
F1	.907							
F2	.991	.873						
F3	.923	.812	.882					
F4	.408	.692	.385	.164				
F5	.301	.310	.241	.379	-.024			
F6	.727	.569	.676	.873	-.059	.359		
F7	.319	.277	.281	.418	-.043	.348	.353	
F8	.131	.081	.087	.289	-.213	.353	.410	.684

[Table 6] Correlation coefficients among complexity measures (PRANPAS)

	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
P1	.188										
P2	.364	.809									
P3	.406	.775	.993								
P4	.329	.557	.874	.870							
P5	.247	.525	.757	.751	.869						
P6	.342	.769	.855	.845	.781	.733					
P7	.284	.650	.888	.875	.908	.878	.830				
P8	.237	.508	.656	.649	.683	.854	.591	.687			
P9	.254	.470	.728	.730	.811	.927	.645	.863	.859		
P10	.285	.581	.654	.634	.508	.463	.754	.581	.418	.389	
P11	.282	.739	.916	.898	.860	.756	.888	.890	.606	.674	.757

5.2 誤りの分布

[Table 7] Distribution of software errors (PRANFOR)

1. testing phase errors		48.0 %
1. logical errors	14.5%	
2. data structure errors	13.2%	
3. data definition and reference errors	6.9%	
5. module interface errors	5.2%	
2. compile phase errors		27.7%
3. maintenance phase errors		24.3%

5.3 誤りとプログラム複雑度間の相関

[Table 8] Correlation coefficients between software errors and complexity measures (PRANFOR)

		no.of executable statements	cyclomatic number	no.of IF&GOTO statements	no.of COMMON or EQIVALENCE statements
compile phase errors		.818	.828	.814	.486
testing phase errors	logical	.843	.827	.877	.452
	total	.743	.728	.710	.584
maintenance phase		.202	.200	.123	.720
total errors		.741	.737	.695	.723

(Table-5)の結果から、実行文数とcyclomatic numberは高い相関にあることがわかる。またPRANPASのデータからも、実行文数とcyclomatic numberは、高い相関関係にある。さらにSoftware effort, program volume, program difficulty, 参照したglobal変数の総和, callしたモジュールの総数なども同様の傾向が見られる(Table-6)。

次に誤りとプログラム複雑度の間の相関を見ると、cyclomatic number, 実行文数, IF&GOTO文数は、compile phase, testing phaseに発生した誤り、および誤り総数と高い相関にある。しかし、尺度の評価の立場からは、実行文数と誤りの相関が高い場合には、実行文数と高い相関関係にある尺度は、実行文数の影響をとり除いて評価しなければならない。[5),6),7),8)]

[Table 9] Correlation coefficients between software error rate and normalized complexity measures (PRANFOR)

		normalized cyclomatic number	ratio of IF&GOTO statements	ratio of COMMON or EQUIV.st.
compile phase error rate		.26	-.29	.57
testing phase error rate	logical	.60	-.11	.43
	total	.51	-.21	.48
maintenance phase error rate		-.19	-.23	.62
rate of total errors		.36	-.32	.78

(Table-9)より、実行文数で正規化して評価すると尺度と誤りの間の相関は低くなる。しかしながら、論理的な誤りの発生率に対しては、normalized cyclomatic numberは、他の尺度よりも相関が高い。また全誤りの発生率に対して、内部変数の使用に対する複雑さの尺度であるCOMMON文, EQUIVALENCE文の出現頻度が高い相関を示し、プログラム内のglobalなデータのとり扱い、モジュールのLocalityが、信頼性に影響していることを示している。

PRANPASのデータに対しての同様な解析結果の一例を(Table-10)に示す。

PRANPASの誤り情報は、現在分析中であり、今回はコンパイル誤りのみを対象とした。

[Table 10] Correlation coefficients between compile phase errors and complexity measures (PRANPAS)

	P0	P1	P2	P3	P4	P5	P6	P7
no.of compile phase errors	.18	.51	.59	.56	.54	.47	.58	.59
	-.03	-.01	.23	.22	-.11	.01	.27	--

	P8	P9	P10	P11
no.of compile phase errors	.45	.47	.50	.59
	.13	.05	.10	-.04

上段：compile phase の誤りと複雑度の相関
下段：実行文数で正規化した誤りと複雑度の相関

6. おわりに

本稿では、種々の複雑度間の相関を示すことにより、従来考えられてきた複雑度の間には、高い相関関係が存在することを示した。さらにプログラムのサイズで複雑度を正規化し、誤りの発生率との相関で評価した場合、強い相関ではないが、プログラムの論理的な誤りに対して、プログラムの制御構造が、影響していることが認められた。また、プログラムの内部変数の使用の仕方も、誤りの発生に対して影響していることが認められた。しかしながら、これらの尺度は、プログラムの複雑さすべてを表わしておらず、新しい尺度の開発が、課題である。だが、尺度の計算に比較的時間のかかるもの、例えばHarrisonらのscope metricなどは適当でないと考える。簡単に計算でき、プログラムの複雑さすべてを表わす尺度の開発が課題である。

参考文献

[1] T.J.McCabe: "A Complexity Measure," IEEE Trans. Software Eng., SE-2,4(1976),308-320.
[2] M.H.Halstead: Elements of Software Science, North-Holland, NEW YORK(1977)
[3] W.A.Harrison et al.: "A Complexity Measure Based on Nesting Level," ACM SIGPLAN Notices,16,3(1981),63-74.
[4] E.T.Chen: "Program Complexity and Programmer Productivity," IEEE Trans. Software Eng.,SE-4,3(1978),187-194.
[5] D.Potier et al.: "Experiments with computer software complexity and reliabilty," Proc. of 6th ICSE(1982),94-103.
[6] 花田,他:「プログラム構造の複雑さ尺度の評価と導出法の提案」情報処理学会論文誌,23, 6(1982), 701-706.
[7] P.G.Hamer et al.: "M.H.Halstead's Software Science - A Critical Examination," Proc. of 6th ICSE(1982),197-206.
[8] 粟名,鷹田:「ソフトウェアの複雑度と分析法」情報処理学会,第26回全国大会(1983),527-528.