ソフトウェア工学 40－16
（1985. 2. 8）

プログラムの複雑度と信頼性について

桑名栄二（電電公社　横須賀電気通信研究所）
亀田壽夫（電気通信大学　計算機科学科）

1．　はじめに

近年、プログラムの設計、開発、テスト、保守等に要するコストの割合が大きくなり、プログラムの信頼性がソフトウェア工学の分野で大きな問題となっている。プログラムの品質、信頼性を定量的に分析・評価するために、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(error) の発生率から信頼度成長モデルを開発し評価する。

などという種々の研究がなされてきた。

ソフトウェアの誤りの発生、発見に対して、そのプログラムの構造・制御の流れ、および、データの流れは大きな影響をおよぼす。この考えをもとに、（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、静的なプログラム複雑度を求め、プログラムの品質・信頼性を評価するものである。

（2）の方法は、発見誤り数を用いた確率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り、種々の確率モデルが提案されている[1,2,3]。

プログラム複雑度は、プログラムの原文の(textual) 複雑度とプログラムの構造的な(structual) 複雑度とに大別できる。前者の代表的な尺度として、プログラムサイズ(program size)、プログラム内のオペランド数・オペレータ数からプログラム開発に於ける労力(effort)を経験に基づいて求めたHalsteadのSoftware effort プログラム内の命令数(No. of instructions) などがある[4]。後者の代表的な尺度として、現在までに、グラフ理論から得られたMaCabeのCyclomatic number [5]、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[6] 及びChenのMIN [7] など数多くのプログラム複雑度が提案されている。しかしながら、Cyclomatic number, Software effort などの一部の尺度については、尺度の性質が調べられているが、尺度間の関係、及び、各尺度とプログラムの信頼性との関係はほとんど明らかになっていない。

近年の研究により代表的な尺度であるCyclomatic number やSoftware effort と、最も単純な尺度である実行文数の間に強い相関関係が存在することが明らかになっている。また、プログラム誤り数との間にも強い相関が存在することが明らかになって来た[8,9]。

本研究は、プログラムの信頼性をプログラム複雑度から推測することを主眼としている。本研究では従来の代表的な尺度の他に、その性質が明らかになっていない種々の尺度について分析するために、プログラム複雑度評価システム(PRANFOR:FORTRANプログラム用,PRANPAS:Pascal プログラム用）を作成した。これらのシステムの主機能は、各プログラム複雑度の測定、プログラム複雑度間の関係分析、プログラム誤りとプログラム複雑度間の関係分析の3つである。

本研究に於いても従来の結果と同様にCyclomatic number やSoftware effort と最も単純な尺度である実行文数の間に強い相関関係が存在することが確認された。また、今まで明らかになっていなかった他の尺度についても実行文数との間に強い相関関係が存在し、従来の尺度のほとんどが実行文数の影響を受けていることがわかった。そこで、本研究では、実行文数以外の尺度を実行文数で正規化し分析することにした。また、従来の研究がプログラム誤り総数のみを対象としている。これに対し、本研究では、総数のみでなく、その性質・発生時期からプログラム誤りの分類を行ない、尺度との関係分析を行なった。分析方法は、

（1）正規化した尺度とプログラム誤り発生率との相関分析

（2）正規化した尺度を因子として選んだときの、プログラム誤り発生数、誤り発生率に対する分散分析法（1元配置法）

の2つの方法で分析・評価した。

本稿では、取り上げたプログラム複雑度を2節で、3節でプログラム誤りの詳細について、4節で評価手法について、5、6節で分析システムと実験結果について述べる。

2．プログラム複雑度

プログラム複雑度は、

（1）プログラムテキストの複雑度

（2）プログラムの構造的複雑度

（3）プログラム論理（アルゴリズム）の複雑度、の3つに大別することができる。(1) のプログラムのテキストの複雑度の代表例としては、プログラムサイズ、HalsteadのSoftware effort などがある。(2) のプログラムの構造的複雑度は、プログラム内の制御の流れに着目し、プログラムをフローグラフ化したときグラフ理論などを用いて得られる複雑度である。(3) のプログラム論理の複雑度においてはアルゴリズムの複雑さを、TIME COMPLEXITY やSPACE COMPLEXITYで表現するものである。

本研究では、プログラムの信頼性と(1),(2) の複雑度の関係について調べた。プログラム誤りの発生については、

（A）プログラムの大きさ

（B）プログラム内のモジュールの関係

（C）モジュール間の変数の取り扱い

（D）プログラム内部の制御構造
　　(if then else,while do,etc.)

（E）プログラミング言語の選択

（F）プログラムの設計、開発方法

等種々の要因が考えられる。プログラム複雑度は、これらの要因から作り出されるものや、その要因自身が尺度として取り扱われている。本稿では、これらの複雑度・要因をまとめてプログラム複雑度と呼んでいる。本研究で採用したプログラム複雑度をTable-1,Table-2 に示す。

[Table-1] Complexity measures(PRANFOR)

F0: no. of executable statements
F1: cyclomatic number
F2: no. of sequences
F3: no. of 2-way branches (logical if st.)
F4: no. of 3-way branches
(arithmetic if st., computed GOTO st.)
F5: no. of more than 4-way branches
(computed GOTO statements)
F6: no. of IF&GOTO statements
F7: no. of type declaration statements
F8: no. of COMMON or EQUIVALENCE statements

[Table-2] Complexity measures(PRANPAS)

F41: no. of declared labels
F42: max depth of nesting level
(control structures)
F43: cyclomatic number
F44: no. of conditions in predicate + 1
F45: software effort *
F46: program volume *
F47: program difficulty *
F48: no. of executable statements
F49: no. of global variables
F50: no. of global variable references
F51: no. of called modules
F52: total no. of called modules
F53: interface complexity(=F49*F51)
F54: program length *
F55: program vocabulary *
F56: nesting level complexity measure
*：Halstead's definition

3．プログラム誤り

プログラムの信頼性を推測する上で、プログラム（ソフトウェア）の誤りはハードウェアの誤りとはその性質が異なり、プログラム複雑度と誤りの関係を分析する上で、誤りの意味・性質について定義することが必要である。プログラムの開発、運用過程は通常、

(1) 要求定義 (2) 設計 (3) コーディング
(4) デバッグ・テスト (5) 保守・管理

のように分類される。このソフトウェア・ライフサイクルの各フェイズにおいて、各種のプログラム誤りが混入する。誤りの混入時期、誤りの性質により、その誤りが誤りの発見・訂正におよぼす影響は全く異なる。

プログラム誤りはその性質から大きく、

(1) 不注意による誤り（例えば、プログラム中のオペランド、オペレータの綴りミス）
(2) プログラム論理の誤り

に大別できる。(1) の綴りミスなどは主にコーディングの過程で混入し、コンパイルなどを行なうことにより発見され訂正される。しかしながらその発見のされ方は、採用されたプログラミング言語、及び処理系によって異なる。従って、使用する言語よって綴りミスもその重みは異なる。これに対して、(2) のプログラム論理の誤りがプログラムの信頼性におよぼす影響は最も大きい。この種の誤りは、設計、コーディングフェイズで混入し、それが発見されるのは、主にテストフェイズ以降である。(1) の不注意による誤りの訂正に対して、プログラム論理の誤りの訂正はプログラムの中の他の部分の変更にまで及ぶことがある。従って、プログラム論理の誤りはその性質で分類し分析することが望ましい。

現在までに、プログラム誤りについて種々の分類がなされ、その発生頻度、発生の防止などについて研究がなされている[10]。ところが、プログラム複雑度を用いて信頼性を推測することを狙いとする諸研究では、プログラム誤りの総数のみが取り扱われている。本研究では、プログラム誤りはその性質、発生時期で信頼性に対する影響の度合いは異なるという理由から、プログラムのライフサイクルを大きく誤り測定可能な３つのフェイズに分けた(Table-3)。また(2) のプログラム論理の誤りをその性質で６つに分類し比較することにした(Table-4)。

[Table-3] Program life cycle

1: Compile phase(debug phase)
2: Testing phase
3: Maintenance phase

[Table-4] Program error categories

1: logical error
2: data definition and reference error
(data handling error)
3: data structure error
4: module interface error
5: I/O error
6: computational error

4．プログラム複雑度の評価方法

現在までにプログラム複雑度の評価方法として、主にプログラム誤りとプログラム複雑度間の相関関係を調べる方法がとられてきた。しかしながら、プログラム複雑度間の関係を考慮し誤りとの関係が分析されていないという問題点がある[8,11,12]。

本研究では、

(1) プログラム複雑度間の分析を行なう。
(2) プログラム複雑度間の関係を考慮し、プログラム誤りと複雑度の関係分析で各プログラム複雑度の評価を行なう。

という順序で実験を行なった。分析方法は、

(1) プログラム複雑度間の相関分析
(2) プログラム複雑度とプログラム誤りの相関分析
((1) の結果を考慮した上での相関分析も含む)
(3) プログラム複雑度を因子として選んだときの、プログラム誤り発生数・発生率に対する分散分析

の３つの分析方法を用いてプログラム複雑度の分析評価を行なった。

4．1相関分析

相関係数(r) = $Cov(x,y)/(\sqrt{Var(x)}\cdot\sqrt{Var(Y)})$

4．2分散分析法（一元配置法）

分散分析法は、種々の条件のもとで得られたデータをもとにして、その条件が観測値に対し相対的な差異を生じさせているかどうか調べる方法である。観測（特性）に対して関係すると考えられる因子(factor)を１つ選び、その因子を量（または質）の点でいくつかの水準に分け、各水準で観測値が異なるかどうか調べる。観測値が異なるかどうかは観測値のばらつきが用いられ、ばらつきの原因を調べるために分散を２つの成分に分解し、その２つの成分の大きさを比較する

方法がとられる(Fig.-1, Fig.-2)。

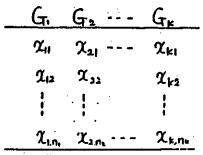


Fig-1

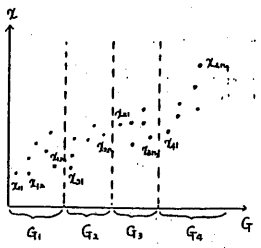


Fig-2

いまFig.-2のように１つの因子(G) を選びその因子を量（または質）の点でｋ水準に分ける。このとき全観測値の変動（Ｓt)を、各水準毎の平均値のばらつきが全体の変動に寄与している部分の級間変動（Ｓg)と因子以外の他の全ての要因が観測値におよぼす影響の度合いを示す級内変動（誤差変動(Ｓe)）に分解する。各水準間に差があるかどうかは、諸々の誤差に対して各水準間の差異が相対的に際立っているかどうか調べればよい。そこで級間、誤差の変動を各々自由度k-1, N-k で割り２つの不偏分散を比較する。このとき分散比（f)は、各水準の主効果が等しいという仮設のもとで、自由度(k-1,N-k）のＦ分布をすることがわかっているので、これに基づいて因子の観測値に対する有意性について検定することができる。さらに、ある因子のもとでの分散分析に於いてその因子の有意性が認められたとき、その因子が全変動の原因の何%を占めているのか（寄与率）見積ることができる。

４．３分散分析法の適用

各プログラム複雑度間の実験結果（参照６節）から各プログラム複雑度が最も単純な尺度である実行文数と強い相関関係にあり、プログラム誤り数と実行文数との間にも強い相関関係が存在することが認められた。そこで実行文数の影響を取り除くために実行文数以外のすべてのプログラム複雑度とプログラム誤り数を実行文数で正規化した。次に実行文数、及び、正規化したプログラム複雑度を因子として選び、プログラム誤り数とプログラム誤り発生率に対して分散分析を行なった。

実行文数、及び、正規化したプログラム複雑度の測定値の小さい順に各モジュールをｋグループに分ける。次に各グループに水準値i=1,2...,kを割り当て、水準値を説明変数、プログラム誤り数、プログラム誤り発生率を目的変数としている。水準数ｋは各グループ内の繰り返し数が大きくなるように、また水準値をできるだけ細かくとるために、PRANPAS ではｋ＝４、PRANFOR ではｋ＝３とした。分散分析の例をTable-5に示す。

４．４適用上の問題点

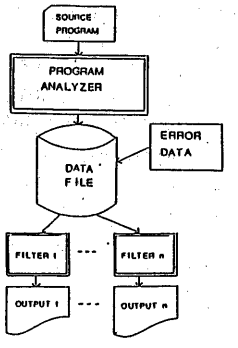
本来の分散分析法の条件は、因子のパラメータが各水準毎に決定され、各パラメータ値均等の条件のもとでｎ回づつ繰り返し実験を行なう。本実験では、得られたデータを尺度の大きさでグループ分けし各グループに水準値(i=1,2,...,k) を割り付けているが、実験試料をグループ分けしたときの各グループの真のパラメータ値は一様均質でない。つまり、実験試料のパラメータ値自身がばらつきを持っている。

従って、寄与率などを計算したときその値が正確な推定値でなく、誤差を含んでいると考えなければならない。そこで、実験試料のパラメータ値がばらつきを持っている場合の分散分析結果に与える影響について水準数を変えることで考察を行なった（参照６節）。

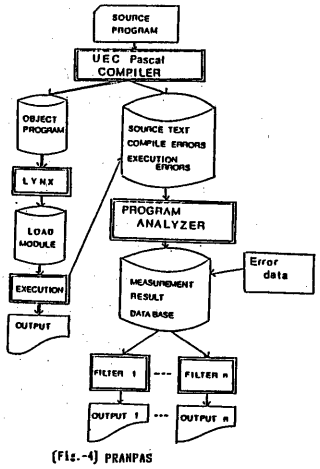
５．プログラム複雑度分析システム

Fortran, Pascal で書かれたプログラムの複雑度を測定し分析する手段として２つのシステムを作成した。Fortran 用システムをPRANFOR(Program Analyzer for FORTRAN)、Pascal用システムをPRANPAS (Program Analyzer for PASCAL)と呼ぶ。２つのプログラム複雑度分析システムの構成図をFig.-3、Fig.-4に示す。これらのシステムは、プログラム複雑度の測定と、プログラム複雑度とプログラム誤りの関係分析を目的としており、プログラム複雑度測定部とプログラム複雑度分析部の２つに分けることができる。PRANFOR の入力データはFortran ソースプログラムとそのプログラム作成過程・保守過程で発生した誤りデータの２である。一方、PRANPAS はPRANFOR とは異なり、自動的にプログラムソーステキスト、コンパイル誤り、実行時誤りが記録されるようになっている。自動的に記録されたデータやソースプログラムを各々測定部に入力することで各プログラム複雑度の測定を行なう。次にこの測定結果とプログラム誤りデータを分析部に入力しプログラム複雑度、プログラム誤りの分析を行なう。

PRANPAS 、PRANFOR の複雑度測定部は、Table-1 、Table-2 に示した複雑度の他に各言語の命令の出現頻度なども測定することができる。複雑度測定はコンパイラの字句解析部と構文解析部を改変することで容易に実現できることがPRANFOR の作成過程に於いて明らかになった。そこでPRANPAS 測定部は既製のPascal Compiler を改変しそれに複雑度測定ルーチンを付け加えることで実現した。



[Fig.-3] PRANFOR



(Fig.-4) PRANPAS

6．実験結果

6．1サンプルプログラム

本実験で用いたサンプルプログラムの特性をTable-6、Table-7 に示す。3つのフェイズ(Compile,testing,maintenance phase) 全ての誤りデータの得られているものは、プログラム複雑度分析のために本実験で作成したProgram Analyzer及びUtility Program であるPRANFOR で用いたサンプルプログラムにはBENCHMARK TEST,FORDAP,SIMTRAN が含まれている。 Table-6,Table-7 からわかるように本実験で用いたサンプルプログラムは、比較的小規模なプログラムを対象としている。

6．2プログラム複雑度間の相関

全サンプルプログラムを対象としたときのプログラム複雑度間の相関関係をTable-8,Table-9 に示す。この結果から、従来の研究の結果と同様にCyclomatic number, Software effortの尺度とも非常に実行文数と強い相関関係にあることがわかる。さらに現在までわかっていなかった他の殆どの尺度もCyclomatic number などと同様に実行文数と強い相関関係にあった。しかし、大域データ領域数や多分岐ステートメント数などは実行文数との間に無相関ではないが強い相関は認められなかった。

6．3プログラム誤りの分布

Fortran,Pascalの各フェイズ毎のプログラム誤りの分布、及び比較的多くの誤りの見つかったTesting Phase の誤りの性質での分類結果をTable-10, Table-11に示す。

6．4プログラム複雑度とプログラム誤りの相関

プログラム複雑度とプログラム誤りの相関関係をTable-12,Table-13 に示す。実行文数とプログラム誤りの相関に着目すると、Fortran, Pascal 共に全誤りに対して強い相関にある。従って尺度の評価を行なう上で、実行文数の影響を取り除いて均質条件で評価しなければならない。そこで、全ての尺度を実行文数で正規化しプログラム誤りも発生率として比較を行なった。PRANFOR のデータから、Ratio of COMMON or EQUIVALENCE st. を除き殆ど相関はなくなっている。PRANPAS も同様に殆どの尺度が、誤り発生率と相関が見られなくなっている。PRANFOR の1部の尺度で相関が見られたが、どの程度までその尺度が関与しているのかは相関関係だけからは判断できない。相関関係を調べるのは2つの変数の間に1次関係が存在するかどうかを調べるのみである。このようなことから今回得られたデータを用いての相関分析からだけでは、尺度の評価は困難であること考える。

6．5分散分析結果

PRANFOR,PRANPAS で得られたデータに対して、因子としてプログラム複雑度を選び（実行文数以外は全て実行文数で正規化）、プログラム誤り・発生率に対し分散分析を行なった(Tbale-16,Table-17)。＊印は有意水準5%で、＊＊印は有意水準1%で有意であったことを示している。

これらの結果から、実行文数やProgram Vocabularyがどのサンプルプログラムに対しても有意差の見られた尺度であった。各尺度がプログラム誤りの発生率、発生数の全変動の何%を占めているのか寄与率の推定値を計算したものをTable-18,Table-19 に示す。さらに、各水準の実験試料のパラメータ値のばらつきの影響を調べるために、水準数を大きく取った場合の寄与率の変化をFig.-6に示す。水準数を大きくすると級間のばらつきが大きくなり寄与率が増大する傾向にある。しかし、各グループの繰り返し数が6程度で非常に少ないときでも、実行文数、Program Vocabularyの寄与率は各々40%,10%程度であった。

しかし、これらの寄与率はあまり大きくなく、プログラム誤りに対して、プログラムの構造等の従来から述べられて来た原因以外の他の複数の要因が大きく関与していることが示唆された。

7．おわりに

本論文では、比較的小規模なSEQUENTIALに動作するプログラムについて、その複雑と誤りの関係の分析結果について述べた。今後このようなデータの収集、分析を種々のプログラム（例えばＯＳ）について行ない、プログラム複雑度からどの程度信頼性を推測できるかさらに調べていかねばならない。

［参考文献］

[1] B.Littlewood, "Theories of Software Reliability," IEEE Trans. Software Eng.,Vol.SE-6 No.5(1980)

[2] B.Littlewood, "How To Measure Software Reliability and How Not To," IEEE Trans. Reliability, Vol.R-28, No.3(1979)

[3] I.Miyamoto, "Software Reliability in Online Real Time Environment," Proc. of Int. Conf. Reliable Software, LosAngeles Calif.(1975)

[4] M.H.Halstead, Elements of Software Science, Elsevier North-Holland,Inc.(1977)

[5] T.J.MaCabe, "A Complexity Measure," IEEE Trans. Software Eng.,Vol.SE-2,No.4(1976)

[6] W.Harrison et al."A Complexity Measure Based on Nesting Level," ACM SIGPLAN Notices(1981)

[7] E.Chen, "Program Complexity and Programmer Productivity," IEEE Trans. Software Eng., Vol.SE-4,May(1978)

[8] D.Potier et al.,"Experiments with Computer Complexity and Reliability," Proc. 6th ICSE (1982)

[9] B.Curtis et al.,"Measuring the Psychological Complexity of Software Maintenance Tasks with the Halstead and MaCabe Metrics," IEEE Trans. Software Eng., Vol.SE-5,No.2(1979)

[10]A.T.Thayar et al.,Software Reliability, North-Holland,New York(1978)

[11]花田、高橋他「プログラム構造の複雑さ尺度の評価と導出法の提案」情処学会論文誌 Vol.23,No.6 (1982)

サンプルプログラム

(Table-6) Sample program(PRANFOR)

name	no. of source lines	no. of modules	C	T	M	analysis
TEST-101	1500	27	o	o	o	RM, RME
TEST-102	3500	71	x	x	x	RM
TEST-103	1500	30	x	x	x	RM
TEST-104	1000	24	x	x	x	RM
TEST-105	600	13	o	o	o	RM, RME

(Table-7) Sample program(PRANPAS)

name	no. of source lines	no. of modules	C	T	M	analysis
TEST- 1	5900	153	o	o	o	RM, RME
TEST- 2	1100	31	o	o	o	RM, RME
TEST- 3	800	24	o	x	x	RM, RME
TEST- 4	700	22	o	x	x	RM, RME
TEST- 5	1100	30	o	x	x	RM, RME
TEST- 6	1800	169	o	x	x	RM, RME
TEST- 7	2500	199	o	x	x	RM, RME
TEST- 8	4900	167	x	x	x	RM
TEST- 9	2900	122	x	x	x	RM
TEST-10	320	7	o	o	o	RM, RME
TEST-11	420	9	o	o	o	RM, RME
TEST-12	330	12	o	o	o	RM, RME
TEST-13	400	13	o	o	o	RM, RME
TEST-14	270	13	o	o	o	RM, RME
TEST-15	820	22	o	o	o	RM, RME
TEST-201	Collection of TEST-1,TEST-2, TEST-10 ~ TEST-15		o	o	o	RM, RME

C : Compile phase errors, T : Testing phase errors,
M : Maintenance phase errors.
o --- error data collected.
x --- error data not collected.
RM : analysis of relationships among complexity measures.
RME : analysis of relationships between complexity measures and program errors.

プログラム複雑度間の相関

(Table-8) Correlation coefficients among complexity measures(PRANFOR). sample program : TEST-101 ~ TEST-105.

		F0	F1	F2	F3	F4	F5	F6	F7
no. of executable statements	F0								
cyclomatic number	F1	.907							
no. of sequences	F2	.991	.873						
no. of 2-way branches	F3	.923	.812	.882					
of 3-way branches	F4	.408	.692	.385	.164				
no. of more than 4-way branches	F5	-.301	.310	.241	.339	-.024			
no. of IF & GOTO statements	F6	.727	.569	.676	.873	-.059	.359		
no. of type declaration st.	F7	.319	.277	.201	.418	-.043	.348	.353	
no. of COMMON or EQUIVALENCE st.	F8	.131	.081	.087	.289	-.213	.353	.410	.684

(Table-9) correlation coefficients among complexity measures(PRANPAS) sample program: all sample prog.

	F41	F42	F43	F44	F45	F46	F47	F48	F49	F50	F51	F52	F53	F54
no.of declared labels:F41														
max depth of nesting level:F42	0.33													
cyclomatic number:F43	0.37	0.69												
max cyclomatic number:F44	0.37	0.69	0.99											
software effort:F45	0.16	0.47	0.63	0.65										
program volume:F46	0.26	0.44	0.66	0.66	0.76									
program difficulty:F47	0.28	0.69	0.64	0.65	0.76	0.60								
no.of executable st.:F48	0.30	0.61	0.83	0.84	0.77	0.73	0.70							
no.of global variables:F49	0.26	0.28	0.52	0.51	0.43	0.83	0.32	0.48						
no.of global variable refer.:F50	0.33	0.39	0.63	0.64	0.66	0.91	0.51	0.70	0.88					
no.of called modules:F51	0.17	0.45	0.46	0.46	0.31	0.28	0.50	0.51	0.26	0.30				
total no.of called modules:F52	0.19	0.44	0.75	0.74	0.48	0.55	0.44	0.79	0.47	0.46	0.53			
interface complexity:F53	0.18	0.50	0.70	0.69	0.58	0.60	0.47	0.76	0.55	0.61	0.70	0.74		
program length:F54	0.26	0.47	0.65	0.66	0.77	1.00	0.65	0.74	0.81	0.89	0.29	0.55	0.58	
program vocabulary:F55	0.45	0.54	0.70	0.69	0.59	0.81	0.59	0.68	0.76	0.88	0.45	0.53	0.55	0.82

プログラム誤り

(Table-10) Distribution of program errors(PRANFOR). sample progcam TEST-101,TEST-105. (40 moddules)

phase	no. of errors		%
Compile phase error	48		27.7%
Testing phase error	83		
1. logical error	25	30.1%	
2. data handling error	12	14.5%	
3. data structure error	23	27.7%	48.0%
4. module interface error	10	12.0%	
5. I/O error	8	9.6%	
6. computational error	3	3.6%	
7. others	2	2.5%	
Maintenance phase error	42		24.3%
Total	173		100.0%

(Table-11) Distribution of program errors(PRANPAS). sample program TEST-1,TEST-2,TEST-10~15. (260 modules)

phase	no. of errors		%
Compile phase error	664		64.4%
Testing phase error	327		
1. logical error	121	37.0%	
2. data handling error	58	17.7%	
3. data structure error	30	9.2%	31.7%
4. module interface error	46	14.1%	
5. I/O error	46	14.1%	
6. computational error	16	4.9%	
7. others	10	3.0%	
Maintenance phase error	40		3.9%
Total	1031		100.0%

プログラム複雑度とプログラム誤りの相関

(Table-12) Correlation coefficients between progcam errors and complexity measures(PRANFOR). sample program: TEST-101,TEST-105.

		no.of executable statements	cyclomatic number	no.of IF&GOTO statements	no.of COMMON or EQIVALENC statements
compile phase errors		.818	.828	.814	.486
testing phase errors	logical	.843	.827	.877	.452
	total	.743	.728	.718	.584
maintenance phase		.282	.288	.123	.720
total errors		.741	.737	.695	.723

(Table-13) Correlation coefficients between program errors and complexity measures (PRANPAS).

name	TEST-201	TEST-201	TEST-201	TEST-201	all prog.
phase	compile phase	testing phase	maintenance phase	total phase	compile phase
F41	0.13	0.43	0.20	0.31	0.25
F42	0.50	0.16	0.07	0.48	0.39
F43	0.56	0.30	0.17	0.59	0.37
F44	0.53	0.33	0.17	0.58	0.36
F45	0.44	0.36	0.13	0.51	0.16
F46	0.45	0.34	0.10	0.51	0.23
F47	0.45	0.45	0.21	0.56	0.40
F48	0.58	0.35	0.12	0.62	0.29
F49	0.45	0.22	0.11	0.46	0.26
F50	0.46	0.27	0.10	0.49	0.25
F51	0.50	0.21	0.15	0.54	0.32
F52	0.56	0.32	0.12	0.59	0.24
F53	0.63	0.26	0.07	0.62	0.24
F54	0.45	0.35	0.19	0.52	0.25
F55	0.49	0.33	0.18	0.54	0.54

(Table-14) Correlation coefficients between program error rate and normalized complexity measures(PRANFOR) sample prgcam: TEST-101,TEST-105.

		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	-.31	.48
maintenance phase error rate		-.19	-.23	.62
rate of total errors		.36	-.32	.78

(Table-15) Correlation coefficients between program error rate and normalized complexity measures (PRANPAS).

name	TEST-201	TEST-201	TEST-201	TEST-201	all prog.
phase	compile phase	testing phase	maintenance phase	total phase	compile phase
F41	-0.01	-0.03	0.02	-0.02	0.38
F42	0.03	0.05	-0.14	0.04	0.07
F43	0.19	0.19	-0.11	0.22	0.19
F44	0.17	0.18	-0.10	0.21	0.18
F45	-0.07	-0.05	0.03	-0.08	0.07
F46	-0.02	-0.01	0.05	-0.01	0.14
F47	0.22	0.21	0.05	0.26	0.14
F48	----	----	----	----	----
F49	0.17	0.13	-0.02	0.19	0.19
F50	0.09	0.11	0.02	0.12	0.14
F51	0.06	0.05	0.21	0.09	0.14
F52	-0.03	-0.05	0.13	-0.04	0.15
F53	0.03	-0.09	0.16	-0.01	0.05
F54	0.04	0.05	0.04	0.05	0.15
F55	0.25	0.20	0.04	0.29	0.41

分散分析結果

[Table-16] Test of the ratio $f=V_G/V_e$.
sample program: TEST-101,TEST-105.
(PRANFOR)
degree of freedom among groups=2,
degree of freedom within group=33.

	compile phase	testing phase	total phase
E0	5.849**	17.506**	12.670**
E1	5.749**	0.246	1.272
E2	1.047	2.099	1.811
E3	2.242	0.867	2.410
E4	0.898	0.299	0.181
E5	1.404	0.537	0.319
E6	0.849	0.515	0.906
E7	5.854**	0.799	2.047
E8	3.636*	1.345	3.224

[Table-17] Test of the ratio $f=V_A/V_e$.
sample program: TEST-201.
degree of freedom among groups=3,
degree of freedom within group=248.
error: compile, testing, all phase errors.

	compile phase	testing phase	all phase
E41	0.816	0.934	0.206
E42	0.555	2.515	1.347
E43	1.962	5.130**	3.196*
E44	0.861	2.557*	1.668
E45	1.095	1.326	1.116
E46	0.483	0.877	0.332
E47	4.392**	5.768**	6.830**
E48	18.815**	19.518**	26.281**
E49	1.291	1.641	1.917
E50	0.947	2.283	1.698
E51	0.637	2.729*	0.846
E52	0.667	1.304	0.685
E53	0.542	1.516	0.367
E54	3.719*	2.289	4.117**
E55	4.674**	5.884**	7.884**
E1	0.950	0.741	0.708
E40	0.528	1.320	1.130

[Table-18] Estimation of ρ (%).
$\rho=(S_G-(k-1)*V_e)/S_T$.
sample program: TEST-101,TEST-105.
(PRANFOR)
degree of freedom among groups=2,
degree of freedom within group=33.

	compile phase	testing phase	total phase
E0	21.70%	48.54%	40.01%
E1	21.34	0	1.53
E2	0.27	5.91	4.43
E3	6.33	0	7.46
E4	0	0	0
E5	2.26	0	0
E6	0	0	0
E7	21.74	0	7.44
E8	13.09	1.93	11.27

[Table-19] Estimation of ρ (%).
$\rho=(S_A-(k-1)*V_e)/S_T$.
sample program: TEST-201.
degree of freedom among groups=3,
degree of freedom within group=248.
error: compile, testing, all phase errors.

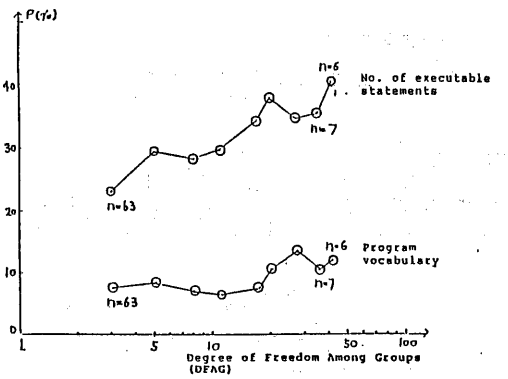
	compile phase	testing phase	all phase
E41	0	0	0
E42	0	1.78	0.41
E43	1.14	4.78	2.56
E44	0	1.83	0.79
E45	0.11	0.39	0.14
E46	0	0	0
E47	3.90	5.39	6.51
E48	17.55	19.21	23.20
E49	0.35	0.76	1.00
E50	0	1.51	0.83
E51	0	2.03	0
E52	0	0.36	0
E53	0	0.61	0
E54	3.15	1.52	3.66
E55	4.21	5.52	7.60
E1	0	0	0
E40	0	0.38	0.16

[Table-20] Test of the ratio $f=V_A/V_e$.
error: compile phase errors.

name	All program	TEST-7	TEST-6	TEST-1	TEST-201
DFWG	668	188	160	144	248
E41	2.662*	4.610**	2.487	1.375	0.816
E42	2.251	6.150**	1.359	0.836	0.555
E43	5.541**	8.664**	2.787*	1.805	1.962
E44	5.661**	8.966**	2.743*	1.093	0.861
E45	3.368*	3.799**	2.063	1.443	1.095
E46	7.447**	7.989**	0.712	0.721	0.483
E47	3.305*	1.764	1.068	6.201**	4.392**
E48	9.682**	7.520**	3.156*	16.412**	18.815**
E49	6.684**	8.347**	1.332	1.843	1.291
E50	1.855	1.838	0.938	0.877	0.947
E51	1.700	0.701	0.435	0.901	0.637
E52	3.600*	4.929**	0.308	0.684	0.667
E53	2.371	2.376	1.783	3.106*	0.542
E54	8.534**	6.583**	1.200	6.471**	3.719**
E55	8.899**	4.610**	4.988**	6.626**	4.674**
E1	3.219*	2.912*	0.914	1.065	0.950
E40	1.126	2.411	0.933	0.567	0.528

DFAG: Degree of Freedom Among Groups(=3)
DFWG: Degree of Freedom Within Group

[Fig.- 6] P vs Degree of Freedom Among Groups(DFAG)
ni: no. of data within group i
($1 \leq i \leq$ (DFAG+1))



[Table- 5] DATA ANALYSIS SYSTEM

Complexity measure ----- Program vocabulary
Factor ---------- Total phase errors
Number of data --------- 252 modules
Data normalization ----- Yes

<<< 1-way classification >>>

	group1	group2	group3	group4
Sum	8.079	7.921	12.153	27.461
Average	0.128	0.126	0.193	0.436
Measure	1.100	1.857	2.826	5.383

<<< Analysis of variance table >>>

Source	sum of square	degree of freedom	mean square	f	jun kouka	kiyoritsu (%)
G	4.073	3	1.358	7.884	3.556	7.60
e	42.703	248	0.172		43.220	92.40
Total	46.776	251			46.776	100.00