プログラム構造と信頼性に関する分析†

花田収悦†† 高橋宗雄††
永瀬淳夫†† 黒田幸明††

プログラム構造の研究に関しては，これまでプログラムの信頼性に影響をおよぼすいくつかの複雑さの要因が指摘されているが，制御構造やデータ参照特性に着目したものが多く，これら以外のプログラム構造特性に着目した複雑さの要因についてはほとんど明らかにされていない．

また，それらの複雑さの要因と信頼性との関係を定量的に分析した研究は少ない上に，その分析手法はプログラムの規模による影響を考慮していない等の点で実用上必ずしも十分ではない．

本論文では，データ構造，データ参照，処理，制御構造及びインタラクションの５つのプログラム構造特性を総合的に捉える観点から複雑さの要因とプログラムの信頼性（単位規模当りのバグ数）との関係を定量的に分析するための分析法を提案し，大規模 OS の開発過程で得られたデータを用いて両者の関係を分析する．

その結果，従来，論文等で提案されている複雑さの要因のすべてが OS のバグ数と必ずしも相関がないこと，さらにこれまで発表されていない UNSPEC 組込関数や PROCEDURE のオプション等のいくつかの要因がむしろ OS のバグ数との相関が強いことを明らかにする．

1. まえがき

プログラムの信頼性や保守性に影響を与える要因の一つであるプログラム構造の複雑さについては，これまで主としてプログラミング言語の要素に基づいた複雑さの要因に関する数多くの研究が行われている．たとえば，代表的なものに Dijkstra[1], Hoare[2], Wulf[3] 等の研究がある．Dijkstra は GOTO 文の有害性を指摘しており，Hoare はポインタ変数がプログラムの信頼性におよぼす影響を論じている．また，Wulf はグローバル変数についていくつかの問題点を指摘している．

しかし，これらの研究は制御構造やデータ参照などの単独の特性に着目したものが多く，これら以外のプログラム構造特性，たとえばデータ構造特性や処理特性等に着目した複雑さの要因についてはほとんど明らかにしていない．また，複雑さの要因とプログラムの信頼性との関係を定量的に分析した研究は少ない上にその分析手法がプログラムの規模による影響を考慮していない[4]，相互の関係を相関係数によるリニアリティのみで検定を行っている[5]，などがあり必ずしも実用上十分であるとはいえない．

本論文では，データ構造，データ参照，処理，制御構造及びインタラクションの５つのプログラム構造特性に着目した総合的な複雑さの要因を選択して，プログラム構造の複雑さとプログラムの信頼性（モジュール別の単位規模当りのバグ数）との関係について，プログラム規模による正規化や分散分析法の導入などによる実用性を加えて分析する方法を提案する．

さらに，大規模 OS を開発する過程で得られたデータを本提案により統計分析した分析結果について考察する．

2. プログラム構造の複雑さの測定法

2.1 複雑さの要因とその分類

プログラム構造の複雑さに関する指摘や OS の開発経験に基づく提案のなかには，具体的な影響要因を必ずしも明確に規定していないものがある．本論文ではプログラム構造の特徴的要因であるデータ構造，データ参照，処理，制御構造及びインタラクションの５つのプログラム構造特性を用いて，次の観点から複雑さの要因を選定した．

（1） プログラム構造特性対応にプログラム構造の複雑さの本質的な要因を考察する．たとえばデータ構造に関する複雑さは，① データの意味のわかりにくさ，② データの値の変化のわかりにくさ，③ データの関連のわかりにくさが本質的な要因であると考える．

† An Analysis on Program Structure and Reliability by SHUETSU HANATA, MUNEO TAKAHASHI, ATSUO NAGASE and KOMEI KURODA (Processing Programs Section, Yokosuka Electrical Communication Laboratory, N. T. T.).

†† 日本電信電話公社横須賀電気通信研究所処理プログラム研究室

表1 複雑さ要因
Table 1 Complexity factors.

プログラム構造特性	複雑さの要因	理由
データ構造	・レジスタ変数	ハードウェアの意識
	・構造体	データの多さ
	・定数[7),6)]	意味のわかりにくさ
	・ビット列変数[7)]	意味のわかりにくさ
	・ポインタ変数[3),7)]	アドレスの意識
	・BASED 変数	変数割付けの意識
データ参照	・変数参照	変数参照の多さ
	・レジスタ割付け／解放文	ハードウェアの意識
	・グローバル変数参照[3)]	変数参照の非局所性
処理	・アドレス処理 (ADDR 組込関数)	アドレスの意識
	・メモリビット操作 (UNSPEC 組込関数)	ハードウェアの意識
	・SUBSTR 組込関数	列操作の多さ
	・型変換	処理の複雑さ
	・代入文	処理の多さ
	・ポインタ演算[7)]	アドレスの意識
	・DO ループ制御変数の変更[7)]	処理の複雑さ
制御構造	・IF 文[8)]	パスの多さ
	・GOTO 文[1)]	制御の流れの複雑さ
	・外部モジュール呼出し[11)]	変数参照の非局所性
インタラクション	・複数入口，出口	制御の流れの複雑さ
	・外部モジュールの呼出し[10)]	インタラクションの多さ
	・PROCEDURE 文のオプション(レジスタ退避・回復指定等の入口・出口処理指定)	コンベンションの複雑さ
	・入出力パラメータ	コンベンションの複雑さ

(2) 本質的な要因に直接関連すると思われるプログラミング言語の要素を選定する．

以上の観点より導出した複雑さの要因をプログラム構造特性対応に分類したものを表1に示す．

なお，表1では手法を具体的に述べるために，複雑さの要因を表現するためのプログラミング言語としてPL/I に類似の言語 (SYSL[6)]) を想定しているが，他の高水準言語でも同様の要因を抽出できる．

2.2 複雑さの要因の測定

複雑さの要因の測定は，対応する言語要素の出現回数が多いほどプログラムの信頼性におよぼす影響が大きいと考えて出現回数で測定する．要因のうち，オプション類は質的データとしてその指定有無を測定する．

3. 複雑さ要因と信頼性との関係分析

3.1 分析の準備

(1) サンプル・プログラム

本実験に用いたプログラムは，OS のジョブ管理，障害復旧プログラムを中心に高水準言語で記述されたモジュールを対象とし，サンプル数は 258，全体規模は約 300 キロ・ステップ (ks) である．

(2) 基礎統計量の測定

(a) 測定方式

複雑さ要因の測定用として，コンパイラを改造したツールを製作し，そのツールによって測定した値をデータベースに格納する．データベースをアクセスす

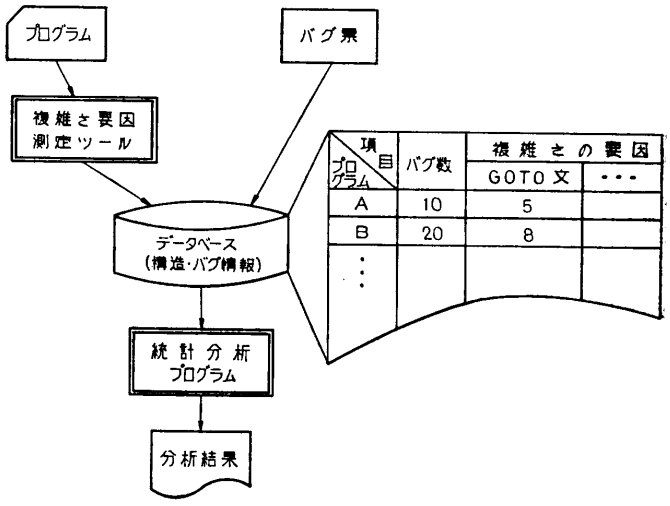


図1 複雑さとバグ数の測定方式
Fig. 1 Method to derive the relation between complexity factors and errors.

表 2 複雑さ要因の分析結果
Table 2 Result of the analysis on complexity factors.

プログラム構造特性	複雑さの要因	検定結果	寄与率(%)	従来の評価(注)
データ構造	・レジスタ変数	*	4.1	
	・構造体	*	3.7	
	・ビット列変数		0.0	○
	・ポインタ変数	*	3.7	○
	・BASED 変数		0.0	
データ参照	・変数参照		0.0	
処理	・ADDR 組込関数	*	5.3	
	・UNSPEC 組込関数	**	6.5	
	・SUBSTR 組込関数		0.2	
	・代入文		0.0	
制御構造	・IF 文		1.7	○
	・GOTO 文		6.0	○
	・内部モジュール呼出し	**	11.9	○
インタラクション	・外部モジュール呼出し		4.6	○
	・PROCEDURE 文のオプション	**	6.2	

(注) ○は従来指摘されていた複雑さ要因を示す.

*：95%有意
**：99%有意

る統計処理プログラムを用いて複雑さ要因間の関係を求めた（**図1**参照）.

（b） 測定結果

図2に示すように，モジュールごとの実行文数とモジュールごとの要因使用数とはきわめて相関が強いことが明らかであるが，実行文数とバグ数との間に強い相関のあることはすでに確かめられているので[10]，分析においては実行文数の影響を除く必要がある．従来の研究においては，要因が実行文数と相関が強いにもかかわらず実行文数の影響を考慮に入れていないものが多いが，本論文では，この影響を除くために要因使用数及びバグ数を実行文数によって正規化した.

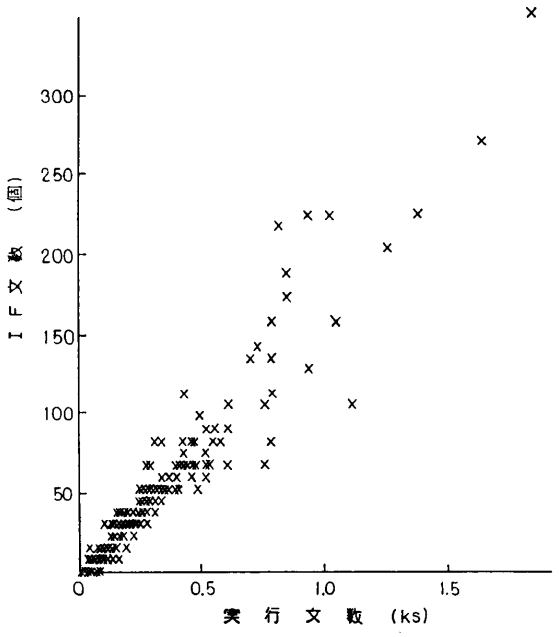


図 2 複雑さ要因と実行文数の関係例
Fig. 2 Example of a relation between complexity factor and steps.

（3） 分析対象プログラムと分析対象要因

正規化によって得られる各モジュールごとの要因使用率（そのモジュールの要因使用数／そのモジュールの実行文数）及び各モジュールごとのバグ率（そのモジュールのバグ数／そのモジュールの実行文数）を観測すると，次のような統計処理上の不適当な点が明らかになった.

（i） 使用率が極端に小さい要因がある.

（ii） 実行文数の小さいモジュールでは，バグ数の微少な変化に対してバグ率が大きく変動する.

これらの要因及びモジュールを除外して，分析対象要因は**表2**に示す15要因，分析対象モジュールは100ステップ以上の127個のモジュールとした.

3.2 分析手法

要因とバグとの関係の分析手法として，従来から相関分析が用いられている．しかし相関係数は二変数間にリニアな関係があるか否かを判定するものであり，両者の関係がリニアでない場合には相関がないと判定されることが多く，要因とバグ間の関係分析には不適当と判断した．本論文では，各要因がどの程度バグ率

に影響を与えているのかを求めることに重点を置くこととし，要因使用率を説明変数，バグ率を基準変数とする分散分析によって要因のバグ率に与える影響を観測する．

なお，各要因の使用率を説明変数の値として直接用いると水準数が多過ぎ，分散分析に適さない．使用率をいくつかの値にグループ化（水準分け）し，その値（水準値）を用いる．要因の水準分けは，次のように行う（**図3**参照）．

（i） 要因 i のモジュール j における使用率を f_{ij} とし，f_{ij} の低い順にモジュールを並べ，低い順に3グループに分ける．1グループ内のモジュール数（繰返し数）は同一となるようにする．

（ii） f_{ij} の低いグループから順に $l=1, 2, 3$ と水準値を割り当てる．あるグループ内のモジュールの f_{ij} をその水準値 l で置き換え，これを $\overline{f_{ij}}$ とする．

水準値 $\overline{f_{ij}}$ を説明変数，バグ率を基準変数として，分析対象要因すべてに対して一元配置の分散分析を行い，各要因のバグ率への影響度合を求める．複雑さの要因がオプション類のように質的データの場合には，指定無に0，有に1の水準値を割り当てる．

3.3 分析結果

分析結果は表2に示す（付録に各要因ごとの分散分析表を示す）．また，使用率がバグ率に影響を与えていることが有意である要因に関するバグ率の推定値のグラフを**図4**に示す．この結果から，主要な事項は以下のように要約できる．

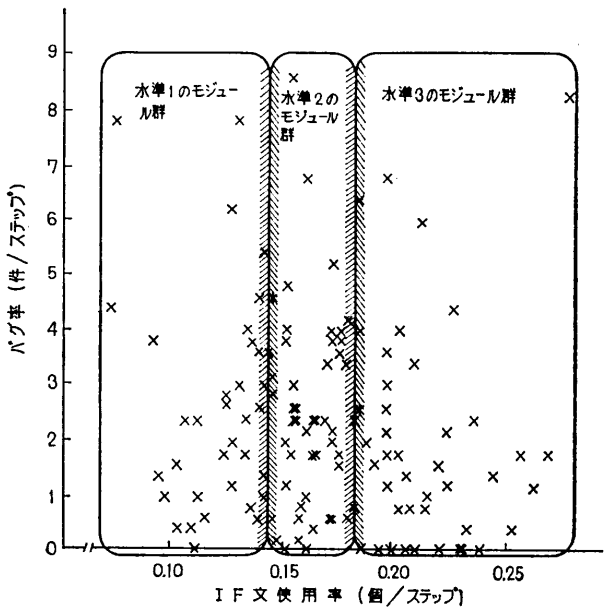


図3 水準化の方法
Fig. 3 Classification method

（1） バグに影響を与えている複雑さの要因は，プログラム構造のすべての構造特性に分布しているが，各々のバグ率への寄与率は小さい（5〜10%）．このことから，一つの複雑さ要因だけでバグ率の変動を説明しようとするのには無理がある．さらに，プログラム構造の複雑さに対する対策は，制御構造特性の良形化だけでなく，データ構造特性など，その他の特性についても考えていく必要がある．

（2） 構造体データの数のほか，UNSPEC 組込関数の数，ADDR 組込関数の数などのメモリ内容の直接参照機能や手続きの入口・出口処理用の PROCEDURE 文のオプションなどがバグ率に影響を与えている．

（3） GOTO 文の有害性については認められなかった．この理由として，GOTO 文使用法に，下方向分岐やループからの脱出用など，一定の制約を課していたことなどが主要な要因と考えられる．このことは，有害であるといわれている複雑さの要因でも，使用法について適切な規約を設けるなどの措置を施せば，悪影響を除去しうることが可能であることを示唆している．

（4） ポインタ変数参照数の多いプログラムほどバグ率は高くなる．システムプログラムは，ポインタによって複雑な内部テーブルを構築しており，ポインタ参照が多いものほど，複雑なテーブル操作が必要となる．本分析結果は，システムプログラムに対しても，Hoare の仮説が成立することを裏付けている．

（5） モジュールの呼出し数は，外部モジュールと内部モジュールでバグ率に与える影響が異なる．

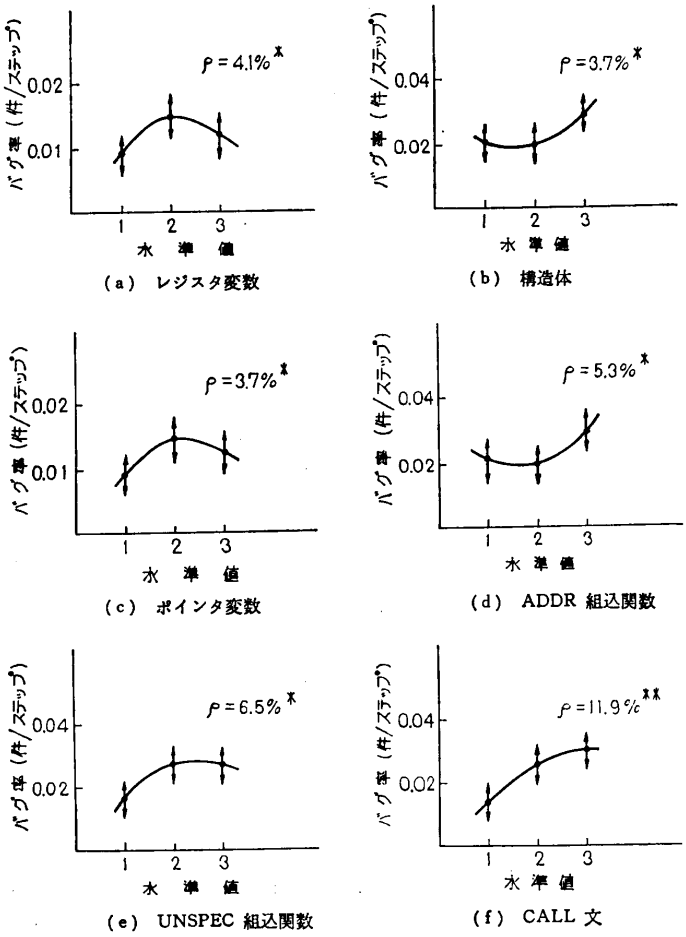
① 外部モジュール呼出し数

バグ率への影響は確認できなかった．むしろ，モジュール呼出し時の環境，協約の複雑さ（手続きの入口・出口処理の複雑さ）の方が有害であることを示している．

② 内部モジュール呼出し数

内部手続きを呼び出す回数の多いプログラムほどバグ率は高くなる．複合設計[11]における内部サブルーチン有害説を裏付けた形となっているが，外部モジュールほど内部モジュールは，モジュール化の基準や管理が明確でなく，担当プログラマにすべてが任されることに大きな原因があると思われる．

（6） IF 文数のバグ率に与える影響は確認できな



(a) レジスタ変数　(b) 構造体
(c) ポインタ変数　(d) ADDR 組込関数
(e) UNSPEC 組込関数　(f) CALL 文
(g) オプション指定

ρ: 寄与率, *: 95%有意, **: 99%有意

図 4 推定値のグラフ

Fig. 4 Estimate graphs.

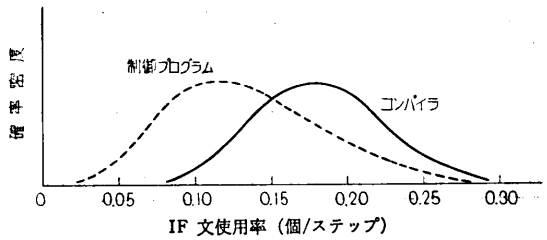


図 5 OS における IF 文使用率分布

Fig. 5 Distribution of IF statements/step in OS.

かった．制御プログラムは，処理プログラムと比較して，IF 文の使用率が 2/3 程度と低く，条件分け型というよりは処理型のプログラムといえる（図5参照）．このことは，条件分けが単純でバグに影響を与えていないとも考えられるが，詳細は今後の分析が必要である．

3.4 分析手法の適用

ソフトウェアの生産性，信頼性分析は，要因数が膨大でしかも定量化困難なものが多いこと，実験が困難であることなどの理由から，本論文のように 300 ks の大規模なプログラムを対象として，定量的かつ組織的に行われたことは少ない．また，要因が有意であることを確認せずに，回帰分析の説明変数として使用するなど，統計処理上不適当な分析を行っているものも多い．

本論文に示した分析手法は，プログラム設計法や複雑さ要因についての仮説を検証するためだけでなく，直観的にリスト・アップした要因が影響を与えているか否かを，既存の蓄積されているデータに基づいて判定する方法であり，複雑さ要因とバグに関するデータが完備されているシステムであれば適用可能である．この分析法によって，種々の問題領域固有の複雑さ要因を抽出することができる．

4. むすび

本論文では，プログラム構造の複雑さの要因と信頼性（バグ数）との関係を定量的に分析するための分析法を提案し，大規模OS（規模約 300 ks）の開発過程で得られたデータを用いて両者の関係を分析した．

提案した分析法は従来の手法に比べると，

（1） 複雑さの要因と強い相関のあるプログラム規模による影響を除くことにより，複雑さの要因の真の影響度合を求めることができる．

（2） 複雑さの要因とバグ数との関係がリニアでない場合でも相互の関係を定量的に求めることができる．

（3） 複雑さの要因が定量的に測定できない質的データの場合でも適用可能である．

点に特徴があり，実用的分析法である．

本分析法による結果を要約すると次のとおりである．

（1） プログラムの信頼性に影響をおよぼす複雑さの要因はプログラム構造特性全体に分布している．しかし，各々のバグ率への影響度合は小さく5～10%である．

（2） 従来，提案されている複雑さの要因のうち，GOTO 文，IF 文，外部手続き呼出しについては有害性は認められなかった．この理由として，GOTO 文については使用法が制限されていたこと，IF 文及び外部手続き呼出しについては使用率が低く，制御プログラムの特性による影響があったことが考えられる．

一方，ポインタ変数及び内部手続き呼出しについてはプログラムの信頼性に影響を与えていることが確認され，Hoare の仮説及び複合設計における有害説が妥当であることを裏付ける一つの分析例である．

（3） 新たに明確にした複雑さの要因としては，UNSPEC 組込関数や PROCEDURE 文のオプションなどがある．

今後は，分析結果を信頼性向上をねらいとした新しいシステム記述言語やその支援系の開発等に発展させる予定である．

おわりに，ソフトウェアのメトリックス (Metrics) について教示していただいた当研究所データ通信研究部戸田巌部長，ソフトウェアの開発管理に科学的手法を導入するとの観点から本研究の機会を与えていただいたデータ処理研究部伊吹公夫部長，筑後道夫元調査役(現東京芝浦電気総合研究所情報システム研究所長)及び有益な御助言をいただいた関係各位に深く感謝する．

参考文献

1) Dijkstra, E. W.: GO TO Statement Considered Harmful, CACM, Vol. 11, No. 3, pp. 147-148 (1968).
2) Hoare, C. A. R.: Data Reliability, Proceedings of International Conference on Reliable Software, pp. 528-533 (1975).
3) Wulf, W. A. and Shaw, M.: Global Variable Considered Harmful, ACM SIGPLAN Notices, Vol. 8, No. 2, pp. 28-34 (1973).
4) Baker, A. L. and Zweben, S. H.: The Use of Software Science in Evaluating Modularity Concepts, IEEE Trans. Softw. Eng., Vol. SE-5, No. 2, pp. 110-120 (1979).
5) Curtis, B. et al.: Measuring the Psychological Complexity of Software Maintenance Tasks with the Halstead and McCabe Metrics, IEEE Trans. Softw. Eng., Vol. SE-5, No. 2, pp. 96-104 (1979).
6) 寺島，高橋：システム製造用言語 SYSL-2 の設計，情報処理，Vol. 16, No. 8, pp. 692-697 (1975).
7) Weissman, L.: Psychological Complexity of Computer Programs: An Experimental Methodology, SIGPLAN Notices, Vol. 9, No. 6, pp. 25-36 (1974).
8) 高橋，田野実：プログラミングにおける定数管理の一手法，昭和55年度情処全大，pp. 177-178 (1980).
9) Weinberg, G. et al.: IF-THEN-ELSE Considered Harmuful, SIGPLAN Notices, Vol. 10, No. 8, pp. 34-44 (1975).
10) Akiyama, F.: An Example of Software System Debugging, Proc. IFIP Congress, pp. 353-359 (1971).
11) 久保，国友：高信頼性ソフトウェアー複合設計，近代科学社 (1976).

(昭和56年5月15日受付)
(昭和56年6月16日採録)

付 録 複雑さ要因ごとの分散分析表 (1/2)

Appendix Anova lists of complexity factors. (1/2)

要因	平方和	自由度	不偏分散	F 値	検定結果	純平方和	寄与率
レジスタ変数	0.0008	2	0.00040			0.0006	4.1
誤差	0.0135	124	0.00011	3.68	*	0.0137	95.9
合計	0.0143	126				0.0143	100.0
構造体	0.0024	2	0.00120			0.0017	3.7
誤差	0.0439	124	0.00035	3.39	*	0.0446	96.3
合計	0.0463	126				0.0463	100.0
ビット列変数	0.0002	2	0.00010			0.0000	0.0
誤差	0.0141	124	0.00011	0.91		0.0143	100.0
合計	0.0143	126				0.0143	100.0

ポインタ変数	0.0008	2	0.00038			0.0005	3.7
誤差	0.0135	124	0.00011	3.44	*	0.0138	96.3
合計	0.0143	126				0.0143	100.0
BASED 変数	0.0000	2	0.00002			0.0000	0.0
誤差	0.0143	124	0.00002	0.18		0.0143	100.0
合計	0.0143	126				0.0143	100.0
変数参照	0.0000	2	0.00001			0.0000	0.0
誤差	0.0143	124	0.00012	0.12		0.0143	100.0
合計	0.0143	126				0.0143	100.0
ADDR 組込関数	0.0031	2	0.00157			0.0024	5.3
誤差	0.0431	124	0.00015	4.52	*	0.0438	94.7
合計	0.0462	126				0.0462	100.0
UNSPEC 組込関数	0.0037	2	0.00185			0.0030	6.5
誤差	0.0425	124	0.00034	5.39	**	0.0432	93.5
合計	0.0462	126				0.0462	100.0

*：95%有意
**：99%有意

複雑さ要因ごとの分散分析表 (2/2)

Anova lists of complexity factors. (2/2)

要因	平方和	自由度	不偏分散	F 値	検定結果	純平方和	寄与率
SUBSTR 組込関数	0.0008	2	0.00041			0.0000	0.2
誤差	0.0454	124	0.00037	1.13		0.0462	99.8
合計	0.0462	126				0.0462	100.0
代入文	0.0003	2	0.00015			0.0000	0.0
誤差	0.0460	124	0.00037	0.39		0.0463	100.0
合計	0.0463	126				0.0463	100.0
IF 文	0.0015	2	0.00075			0.0008	1.7
誤差	0.0448	124	0.00036	2.06		0.0455	98.3
合計	0.0463	126				0.0463	100.0
GOTO 文	0.0035	2	0.00174			0.0028	6.0
誤差	0.0428	124	0.00035	5.04	**	0.0435	94.0
合計	0.0463	126				0.0463	100.0
内部モジュール呼出し	0.0061	2	0.00308			0.0055	11.9
誤差	0.0401	124	0.00032	9.52	**	0.0407	88.1
合計	0.0462	126				0.0462	100.0
外部モジュール呼出し	0.0028	2	0.00138			0.0021	4.5
誤差	0.0435	124	0.00035	3.94	*	0.0442	95.5
合計	0.0463	126				0.0463	100.0
PROCEDURE 文のオプション	0.0032	2	0.00323			0.0029	6.2
誤差	0.0430	124	0.00035	9.38	**	0.0434	93.8
合計	0.0462	126				0.0463	100.0

*：95%有意
**：99%有意

訂　　正

第23巻第1号 pp. 9-15 (1982) に掲載しました花田他の論文「プログラム構造と信頼性に関する分析」は，以下のように訂正いたします．

訂正箇所

1) p. 10，表1　制御構造欄の「外部モジュール呼出し」を「内部モジュール呼出し」とする．

2) p. 12，図3　縦軸のバグ率の単位「件/ステップ」を「件/KS」とし，かつ，目盛の値を10倍する．