ソフトウエア工学 34−19
（１９８４. 2. １０）

ソフトウェアの開発・保守作業の効率改善のための モジュール分析の試み

後藤浩一　二階堂徳也　関栄四郎　大沢健二

（国鉄鉄道技術研究所）（国鉄中央情報システム管理センター）

1．まえがき

コンピュータシステムの開発・保守におけるソフトウェアに関わる経費は増大する一方であり，いわゆるソフトウェア危機が叫ばれているが，いまだ抜本的な解決策を見い出すには至っていない。これは国鉄においても同様であり，みどりの窓口で知られる座席予約システムをはじめとして，貨物情報システム，新幹線運転管理システム，全国的データ通信システム等の大規模なシステムを開発・維持していく上で，各種のソフトウェアに関する問題を抱えている。過去においてもソフトウェアに限らず，システム開発及び保守の効率化を図るために各種の手法やツールを使用してきたが，系統だてて導入が図られてきたとは言い難いのが実情である。また，過去のシステム開発や保守に関するデータの収集・解析が十分に成されていなかった。

これらの経験を踏まえて，財政的に厳しい状況に置かれているなかで，今後も各種の情報システムを開発・維持していくためにより一層の効率化を目指した活動を進めている。前述のように，ソフトウェアに関わる問題においては，今のところ特効薬的なものは期待できないため，自らを見つめなおし，着実に進めて行く方針である。ここではそのような活動の一つとして行っているソフトウェアのモジュール分析について紹介する。

2．背景

モジュール分析の目的や方法を述べる前に，その背景となっている国鉄におけるシステム開発及び保守の効率改善を目指した動きについて簡単に説明する。種々の項目があるが，以下のようにまとめることができる。

(1) 開発・保守作業支援環境の整備

システム開発・保守の効率化のためにいままで各種の手法が提案され，支援ツールも様々なものが開発されている。国鉄においても有効と思われるものは進んで導入してきたが，今後も外部の動向に注意し，有効と思われるものは積極的に導入したい。また，国鉄のシステムに適応した独自の手法やツールの開発も重要な課題であると考えている。

しかし，これらの手法やツールも個々バラバラに用いられているのみでは不十分であり，総合的な支援環境として整備していくことが必要である。時代の最先端をいくコンピュータシステムの開発が手作業中心であることは大きな矛盾である。他の工業製品の製造に使用されるＣＡＤ等と同様なコンピュータシステムの開発・保守自身のシステム化が今後の重要な課題と考える。

(2) ドキュメントに関する作業の改善

上で述べた支援環境としても様々な面から検討する必要があるが，その中でも重要な項目として，我々はドキュメントに注目している。コンピュータシステム開発には非常に多くのドキュメントが使用される。システムの仕様書，設計書，工程管理のためのドキュメント，障害記録，そしてプログラムリストもそうである。ソフトウェア開発は知的な作業と言われるが，単にドキュメントを清書したり，修正したりする手間も大きな割合を占めており，ドキュメントに関する作業を改善できればかなりの効率化が期待できる。その手段としては次の２つが考えられる。

①自動的に作成できるものは極力人手から離す

②どうしても人が作成せざるを得ないものに対しても，手書きのものをなくし，システムで情報を管理して，重複作業を防ぎ，修正を楽にする。

(3) 開発・保守におけるデータの分析

問題点を探りその方策を考えるためには，現状を知ることがまず第一歩である。過去の経験を将来に有用に生かすために，データの分析は必要である。また，このようなことは一過的に行っても効果は小さく，継続的に行う必要がある。そのためにどのような項目について，どう記録し，解析し，使用するのかをはっきりさせ，その具体的な手段・方法を明確にしておく必要がある。

(4) 管理・運営の問題

ソフトウェアの開発・保守においては技術的な問題以外に管理の問題が非常に重要である。特に国鉄の場合各種大規模システムを開発するなかで，各システム毎の独立性が高く，相互に開発方式や管理手法が異なっており，種々の問題を生じてきた。できる限りの統一へ向けて検討を進めているところである。また，最近は直轄作業が少なくなってきており，外注化の比率が高くなっている。これをどのように管理し，品質を維持し，期間・コストの低減を図るかは重要な問題である。開発費の積算基準の設定，納入されたソフトウェアの品質評価や総合的なシステム評価などのガイドラインの作成が求められている。

以上の各項目についての詳細は，まだ検討を進めている段階であり別の機会に譲るが，モジュール分析はこれらと関連した活動である。

3．モジュール分析の目的

プログラムのモジュール化はいままでの経験や各種の研究においても有効な手段とされており，国鉄においてもモジュール化を前提として開発が行われている。しかし，次のような点が不十分であると考えている。

(a) モジュール化を行うときの基準（どのように分割するか，人の担当はどうするか，内部構造はどうすべきか等）が明確とはいえない。現状はステップ数を目安とし，人間が機能や分担の状況により適宜判断しているが，これでよいという安心感は得られていない。

(b) (a)の結果として，出来上がったモジュールの評価基準が明確でない。良いモジュールとは何か，悪いモジュールとは何かがよく分からない。テストをどの程度やればよいか，障害の発生状況は妥当と考えてよいか等が不明である。

(c) 開発前のコストや期間の見積りが精度よくできない。

(d) モジュール化の結果や効果についての過去のデータの整理が十分ではない。

また保守側においても，その作業を支援するための管理資料を作成するツールの必要性があり，モジュールを主眼とした分析プログラムの開発を行っていたので，上記の問題の解決へ向けての一歩として，このプログラムで得た管理情報と障害情報とを利用して調査を行うこととした。

つまりモジュール分析の目的は以下の２点である。

(a) 保守作業を進めるための情報を与える。
- ・所有ソフトウェアの全体把握及び管理
- ・モジュールの機能・性質の解析
- ・モジュール，マクロ，コピー等を変更した時の関連モジュールの検索　など

(b) 開発を進めるための基礎データを得る。
- ・開発及び保守作業をしやすくするモジュール化の指針
- ・作成したモジュールの品質評価基準
- ・開発期間や経費の積算の基準　など

(b)については様々な要素がからんでおり，未解決の問題もあるため，容易に解答が得られるとは考えていないが，少なくともいままではこういった調査自体も行っておらず，現実のデータを分析することにより，何らかの前進は図れるものと期待している。

モジュール分析の２章で挙げた各検討項目における位置づけは次のとおりである。

(1) 開発・保守作業支援環境の整備

モジュール分析プログラムは支援環境のツールの一つとして位置づけられる。将来は統合的な支援環境の中に組み入れていきたい。

(2) ドキュメントに関する作業の改善

モジュール分析プログラムの出力資料の中には，従来手作業で作成していたものも含まれており，ドキュメント自動作成の一つの手段を与える。

(3) 開発・保守におけるデータの分析

これは上で述べたモジュール分析プログラムの目的の(b)そのものである。

(4) 管理・運営の問題

この分析によって，モジュール化の基準，作成したモジュールの品質評価基準，積算基準等が得られれば，管理・運営の有効な道具となる。

3．モジュール分析プログラム

モジュール分析プログラムはＰＬ／Ｉで作成され，いまのところ約10ｋステップであるが，今後とも改良を加えていきたいと考えている。

本プログラムは図１に示すように，プログラム・ライブラリからソース・プログラムを入力して分析する分析サブプログラムとその結果を入力して各種の分析表を印刷する印刷サブプログラムからなる。モジュールの定義に関しては各種のものがあるが，本プログラムが扱うのは，ＰＬ／Ｉではコンパイル単位とPROCEDURE BLOCK 単位，アセンブラ言語でアセンブル単位とセクション単位となっている。

なおバグとの関係をみる部分については，バグの整理が手作業であったこと，まだ試験的に行っていること等から，パソコン上で行っている。（BASIC 約1k steps）

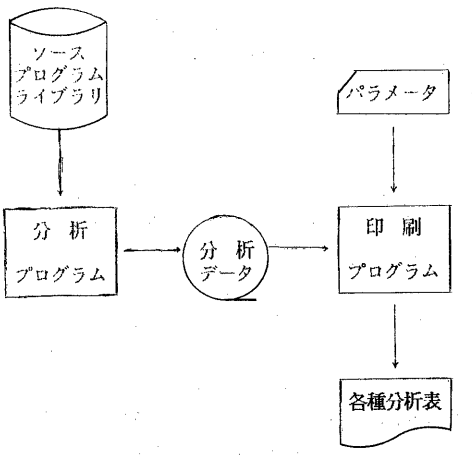

図１　モジュール分析プログラム

表1　モジュール分析プログラムの機能

機　能	概　要
①記述言語決定	ソースを分解し，PL／I，HIPLOT，アセンブラ等に分類する。
②手続き分解	ソースをPROC／END等により手続きに分解する。
③ソース行分類集計	ソースの中の各行を分類し集計する
④手続き名抽出	注釈文から手続きの名称を抽出する
⑤呼出ルーチン名抽出	CALL文等から呼び出しているルーチンを抽出する。
⑥使用コピー名抽出	手続き中で使用しているコピーを抽出する。
⑦使用マクロ名抽出	手続き中で使用しているマクロを抽出する。
⑧入力ファイル名抽出	GET／READ文等から入力ファイルを抽出する。
⑨出力ファイル名抽出	PUT／WRITE文等から出力ファイルを抽出する。
⑩パス解析	手続きの制御の流れをグラフ化し，パスを抽出する。
⑪複雑度解析	手続きの複雑度を計算する

モジュール分析プログラムの機能は表1のとおりであり，以下の分析表を出力する。

① モジュール構造図

各モジュールの関連をトリー状にして示す。

② モジュール別主従関係一覧表

各モジュールが呼び出しているモジュールを示す。

③ モジュール別従主関係一覧表

各モジュールを呼び出しているモジュールを示す。

④ モジュール別使用コピー一覧表

各モジュールが使用しているコピーを示す。

⑤ モジュール別使用マクロ一覧表

各モジュールが使用しているマクロを示す。

⑥ マクロ別使用モジュール一覧表

各マクロを使用しているモジュールを示す。

⑦ モジュール別入出力ファイル一覧表

各モジュールが入出力しているファイルを示す。

⑧ ファイル別入出力モジュール一覧表

各ファイルを入出力しているモジュールを示す。

⑨ モジュール別ステップ数一覧表

各モジュールのステップ数（単に総ステップ数だけではなく，実行文，データ定義文，制御文－コンパイラへの指示等，コメント文，その他の分類別も）を示す。また使用ファイル数，マクロ数等も記載する。

⑩ モジュール別パス解析表

各モジュールの入口から出口までのパスを示す。

⑪ モジュール別複雑度一覧表

各モジュールの複雑度を示す。尺度としては，いまのところMcCabeによるものを計算しているが，その他の尺度も採り入れて比較してみたいと考えている。

⑫ 各種分布表

上記の各項目をシステム全体として集計したものをグラフ化して示す。

分析表の例として，図2にモジュール構造図，図3にコピー別使用モジュール一覧表，図4にモジュール別ステップ一覧表をそれぞれ示す。これらの対象となったシステムはみどりの窓口で知られている座席予約システム（マルスと呼んでいる）の一部である。

4．モジュールの性質とバグとの関連

4．1　対象システムとバグ

対象としているシステムは，マルスシステムの現在稼働中のものであり，マルスS以外は大部分アセンブラ言語によって書かれている。マルスシステムの概要を表2に示す。

表2　マルスシステムの概要

サブシステム名	主たる機能	モジュール数	ステップ数－KS	使用言語
マルス105	指定券発売処理	2163	934	アセンブラ
〃 150	電話予約処理	492	219	〃
〃 202	団体予約処理	1409	678	〃
〃 S	新型端末装置対応処理	772	219	〃
		758	225	PL／I
		1036	311	HPL

4．2で紹介するバグのデータは，中央情報システム管理センター（システムの保守・運用を担当している）で発生したマルスS以外のサブシステムのここ8年間のデータ約560件であり，すべて運用段階に入ってからのデータである。本分析の目的は今後のシステム開発にあたって何らかの基準を得ることを目的としており，今後はほとんどのシステムを高級言語によって開発することが予想されるため，これらのデータは直接には役に立たない恐れもあるが，マルスSサブシステムのデータの整理がまだ十分進んでいないこともあって，試験的な意味で調査を行っているものである。マルスSのデータについては，開発段階でのモジュール結合テストにおけるバグのデータ約2000件について現在整理を進めているところであり，運用段階のデータのさらに詳細な分析とともに，また別の機会に報告したい。

上記いずれのデータもこのような分析を前提としては記録されておらず，特にコンピュータで扱える形になっていなかったため，整理はすべて手作業となり，非常な労力を要する結果となっている。障害発生時点でのデータの整理，特にコンピュータで扱える形でデータを整理しておくことの必要性を痛感した。しかし，そのためには，環境の整備と十分なツールが必要であり，何をどう分析すべきかの方法が明確になっていることが望ましい。

ROUTINE	NAME	MEMBER	LNG	VER	DATE	F	RTN	STEPS
RJ02T00		RJ02T00	PLI	002	83/02/21		15	11864

```
RJ02T00                                                                    3640 RJ02T00
  +----- RJ02TC1                  ｺﾝｿｰﾙ ｼｭﾂﾘｮｸ ｼｮﾘ ﾙｰﾁﾝ                    168 RJ02TC1
  |        +----- RJCCVH           16 ｼﾝ ｺｰﾄﾞ ﾍﾝｶﾝ ｼｮﾘ                      62 RJCCVH
  +----- SPLSOPN                   PLI SINIT ISSUE                           32 SPLSOPN
  +----- SPLTBGN                                                             41 SPLTBGN
  +----- SKFCP                                                               SKFCP
  +----- RJ02T40                   ﾚｺｰﾄﾞ ﾍﾝｼｭｳ ﾒｲﾝ                        7373 RJ02T40
  |        +----- SPLTBGN                                                    41 SPLTBGN
  |        +----- RJ02TC1 *        ｺﾝｿｰﾙ ｼｭﾂﾘｮｸ ｼｮﾘ ﾙｰﾁﾝ                    168 RJ02TC1
  |        +----- RJ02TC0          BS ﾎｳ AGT ﾃｰﾌﾞﾙ ｼｭｳ ﾍﾝｶﾝ ﾙｰﾁﾝ             127 RJ02TC0
  |        +----- RJ02TCZ                                                    31 RJ02TCZ
  |        +----- SKFCP                                                      SKFCP
  |        +----- RJ02TCX          BS ﾎｳ ﾚﾂｼﾔ ﾍﾝｶﾝ ﾃｰﾌﾞﾙ ｼｭｳ ﾙｰﾁﾝ             108 RJ02TCX
  |        +----- RJ02TCW          BS ﾎｳ ｴｷ ﾍﾝｶﾝ ﾃｰﾌﾞﾙ ｼｭｳ ﾙｰﾁﾝ               109 RJ02TCW
  |        +----- RJ02TCM          BS ﾎｳ ｾﾂｿﾞｸ ｺｰﾄﾞ ﾍﾝｶﾝ ﾃｰﾌﾞﾙ ｼｭｳ ﾙｰﾁﾝ        71 RJ02TCM
  |        +----- RJ02TCN          BS ﾎｳ ｹｲﾕ ｾﾝｸ ﾍﾝｶﾝ ﾃｰﾌﾞﾙ ｼｭｳ ﾙｰﾁﾝ            69 RJ02TCN
  +----- PLIRETC                                                             PLIRETC
  +----- RJ02TC0                   BS ﾎｳ AGT ﾃｰﾌﾞﾙ ｼｭｳ ﾍﾝｶﾝ ﾙｰﾁﾝ             127 RJ02TC0
  +----- RJCCVDT                                                             33 RJCCVDT
  +----- RJCCVH                    16 ｼﾝ ｺｰﾄﾞ ﾍﾝｶﾝ ｼｮﾘ                      62 RJCCVH
```

図2　モジュール構造図の例

NO	COPY	LNG	USED-MEMBERS									
0091	A¥EMTMUP	PLI	RF04W15	RF36W00	RF40R0	RF40W0	RF41R0	RF41WC4	RF42R00	RF42W00	RF44R20	RF44W00
			RF51W0	RF60W0	RH02R10	RJ03X20	RJ05W24	RJ90TR1	RJ90T00	RS20R00	RS20R21	RS22R20
			RS29R0	RS60R00	RT41600							
0092	A¥EMTP	PLI	RJ02T00	RJ04W20	RJ07W00	RJ07W02	RJ07W03	RJ07W20				
0093	A¥ER3H	MPL	KBUPM	KFPC010	KFPC020	KFPC030	KFPC040	KFPC050	KFPC060	KFPC070	KFPC100	KFPC200
			KFPF100	KFPF200	KFPF300	KFPF400	KFPF500	KFPF600	KFPF700	KFPM000	KFPM010	KFPS010
			KFPS020	KFPS100	KFPS200	KFPS210	KFPS220	KMCAL30	KMCAL40	KMOSC700	RALP210	RAMP000
			RF03C	RF05C	RF06C	RS02R0	RS02R3	RS02W0	RS02W1	RS02W11	RS02W12	RS03T0
			RS03T3	RS03T5	RS03T7	RS04T1	RS04T22	RS06T00	RS06T21	RS06T22	RS07T0	RS07T3
			RS07T5	RS07T7	RS08T1	RS08T21	RS11R0	RS11R2	RS11R3A	RS11R3F	RS11R31	RS11R32
			RS11R33	RS11R38	RS11R39	RS11TC1	RS11T0	RS11T1	RS11T2	RS11T3	RS11T4	RS11T5
			RS11T6	RS11T7	RS11T8	RS11T9	RS12R0	RS12R2	RS12R3	RS12T0	RS12T03	RS13TC1
			RS13TC2	RS13T0	RS13T1	RS14R0	RS14T0	RS21RC1	RS21RC4	RS21R0	RS21R2	RS21R3
			RS230C2	RS230C3	RS230C5	RS2300	RS23021	RS23022	RS2303	RS2304	RS2305	RS2306
			RS260C1	RS2600	RS2601	RS2602	RS2603	RS2605	RS30R0	RS30R2	RS30R3	RS30W0
			RS30W1	RS30W3	RS31T0	RS31T1	RS31T2	RS31T3	RS32T0	RS32T2	RS32T3	RS32T4
			RS34W11	RS34W12	RS34W2	RS41T2	RS41T5	RS420C2	RS4200	RS4201	RS4202	XTRTM00
			XTRTM10									

図3　コピー別モジュール一覧表の例

NO	MODULE	LNG	TOTAL	PROCS	DCLAR	CNTRL	NOTES	OTHER	VER	DATE	COMMENT
0836	RJCKYK	PLI	95	9	72	4	10	0	000	82/10/13	ｷｮｸｺｰﾄﾞ ｶﾅﾍﾝｶﾝ ｼｮﾘ
0837	RJCMAP11	ASM	19	0	16	2	0	1	004	83/02/03	
0838	RJCMPT	PLI	224	163	12	5	23	21	001	82/12/21	MPT ｼｭｳｾｲ ｼｮﾘ
0839	RJCPMCK	PLI	198	120	32	7	31	8	000	83/03/04	ﾆｭｳﾘｮｸ ﾊﾟﾗﾒｰﾀ ｶｲｾｷ
0840	RJCPM11	ASM	12	0	5	2	4	1	000	82/10/13	RJCPM11 INTERP
0841	RJCPM11	PLI	305	227	41	9	26	2	000	82/10/13	ﾊﾟﾗﾒｰﾀ ﾆｭｳﾘｮｸ&ﾁｪｯｸ ｼｮﾘ
01	RJCPM11		215	149	36	6	22	2			ﾊﾟﾗﾒｰﾀ ﾆｭｳﾘｮｸ&ﾁｪｯｸ ｼｮﾘ
02	RJCPM111		90	78	5	3	4	0			ﾍｲｷﾝｼｮﾘ(ﾆｭｳｲ､ｼｮﾂｹｲ(RJ72R) ﾃﾞﾊﾞｸﾞ(RJ43R
0842	RJCR0C	PLI	134	62	37	5	16	14	000	82/10/13	R0 ﾚｺｰﾄﾞ ｼｮﾘ
0843	RJCUSK	PLI	49	19	16	4	8	2	000	82/10/13	ｼｮｳｸﾌﾟﾝ ｶﾅ ﾍﾝｶﾝ ｼｮﾘ

図4　モジュール別ステップ一覧表の例

4．2　分析結果の一部

モジュール分析プログラムの分析表で得たモジュールの各種性質とバグの統計とを比較して，その間に何らかの関係があるものかどうかを調査している。もし，有意な関係が見つけられたすれば，その関係は一つの尺度として今後の開発・保守に利用したい。　まだ，すべてについて分析が終わっていないため，一部を紹介するにとどめる。

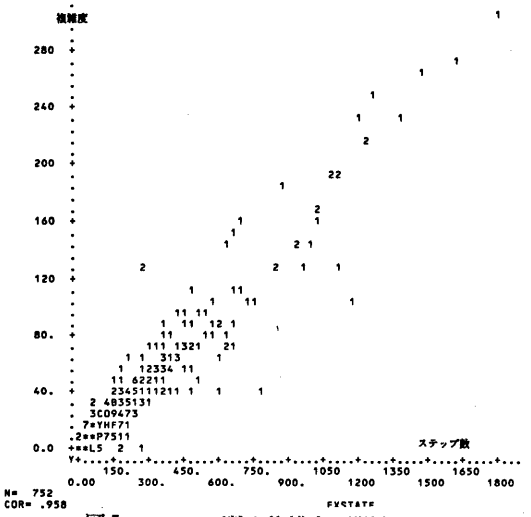

図5　ステップ数と複雑度の関係

```
========== BUG NO BUNPU (BY TOTAL STEPS) ==========
モジュールの大きさ  個数
  0              0
  0 - 50         9  *********
 50 - 100       18  ******************
 100 - 150      33  *********************************
 150 - 200      17  *****************
 200 - 250      31  *******************************
 250 - 300      20  ********************
 300 - 350      18  ******************
 350 - 400      16  ****************
 400 - 450      23  ************************
 450 - 500      19  *******************
 500 - 550      26  **************************
 550 - 600      22  ***********************
 600 - 650      15  ***************
 650 - 700      17  *****************
 700 - 750       9  *********
 750 - 800      11  ***********
 800 - 850      12  ************
 850 - 900       9  *********
 900 - 950       4  ****
 950 - 1000     11  ***********
1000 - 1050     10  **********
1050 - 1100      8  ********
1100 - 1150     12  ************
1150 - 1200     14  **************
1200 - 1250      6  ******
1250 - 1300      2  **
1300 - 1350      1  *
1350 - 1400     10  **********
1400 - 1450      7  *******
1450 - 1500      3  ***
1500 - 1550     10  **********
1550 - 1600      5  *****
1600 - 1650      5  *****
1650 - 1700      5  *****
1700 - 1750      4  ****
1750 - 1800      0
1800 - 1850      3  ***
1850 - 1900      4  ****
1900 - 1950      3  ***
1950 - 2000      4  ****
2000 -          58  **********************************
```

図6　モジュールの大きさ別のバグの分布

図5はマルスSサブシステムのうちPL／Iで作成されているモジュール約750個についてそのステップ数とMcCabeによる複雑度との相関関係を調べたものである。相関係数は0.958 と非常に高く，文献(1)で述べられているのと同様の結果が得られた。これよりMcCabeの複雑度だけでは十分そのモジュールの性質を表しているとはいえず，他の尺度を考える必要のあることが分る。今後他に提案されている複雑度についても調べてみたい。また，バグとの関係についても整理を早急に行って進めていきたい。

図6はアセンブラ言語で書かれているシステムのバグを発生したモジュールの大きさ別（50ステップ刻み）に並べたものである。小さなモジュールの方が件数が多いという結果になっている。これは具体的数字はここには掲載していないが，小さいモジュールの方が数が多いことからくる結果であると思われる。

```
========== BUG NO BUNPU (BY MODULE ATARI NO HASSEI RITSU)
モジュールの大きさ      発生率
  0 - 50         9   .0216  **
 50 - 100       18   .0429  ****
 100 - 150      33   .0938  *********
 150 - 200      17   .0489  *****
 200 - 250      31   .0963  **********
 250 - 300      20   .0847  ********
 300 - 350      18   .0914  *********
 350 - 400      16   .0894  *********
 400 - 450      23   .1533  ***************
 450 - 500      19   .145   **************
 500 - 550      26   .1884  ******************
 550 - 600      22   .2018  ********************
 600 - 650      15   .1485  **************
 650 - 700      17   .1954  ********************
 700 - 750       9   .1385  *************
 750 - 800      11   .1528  ***************
 800 - 850      12   .25    ************************
 850 - 900       9   .1731  *****************
 900 - 950       4   .1081  **********
 950 - 1000     11   .3438  ********************************
```

図7　モジュールあたりのバグの発生率

図7はモジュールの大きさ別のモジュール1個あたりのバグ発生率であり，これは大きなモジュールの方が発生率が高いという結果になっているが，ステップ数が多ければ，それだけバグの生じる機会も多くなるのは当然とも思われる。そのため複雑度のような大きさ以外の尺度も必要となるが，前述のように複雑度自身がステップ数と高い相関を持っている場合にはその尺度で十分であるとはいえない。

```
========== BUG NO BUNPU (BY STEP ATARI NO HASSEI RITSU)
モジュールの大きさ      発生率
  0 - 50         9   .00087  ****************************
 50 - 100       18   .00057  ******************************
 100 - 150      33   .00075  *********************************
 150 - 200      17   .00028  *************
 200 - 250      31   .00043  ********************
 250 - 300      20   .00031  ***************
 300 - 350      18   .00028  *************
 350 - 400      16   .00024  ************
 400 - 450      23   .00036  ******************
 450 - 500      19   .00031  ***************
 500 - 550      26   .00036  ******************
 550 - 600      22   .00035  *****************
 600 - 650      15   .00024  ************
 650 - 700      17   .00029  *************
 700 - 750       9   .00019  *********
 750 - 800      11    .0002  **********
 800 - 850      12    .0003  ***************
 850 - 900       9   .0002   **********
 900 - 950       4   .00012  ******
 950 - 1000     11   .00035  *****************
```

図8　ステップあたりのバグの発生率

図8はモジュールの大きさ別の1ステップあたりのバグ発生率である。小さいモジュールの方がバグの発生率が大きいという結果となっている。これは小さいモジュールでは1ステップあたりの寄与率が大きくなるためと思われるが，これをどのように正規化するかは難しい問題である。ＰＬ／Ｉにおけるバグのデータとも比較してさらに検討したい。また単に形態だけではなく，機能も考慮した分析が必要なことを示唆しているように思われる。

前述のようにモジュール分析プログラムは，各モジュール命令を実行文，データ定義文，コメント文，制御文，その他に分けた数字を出力する。これらの各命令のモジュールにおける割合とバグとの間に何か関係がないかどうかを調べてみたのが図9および図10である。図9の上のグラフはバグを生じたモジュールの実行文の割合の分布であり，下のグラフは全体のモジュールでの分布である。当然の結果であるともいえるが，実行文のないモジュール（データ定義のみなど）ではその個数に比べてバグは少なく，全体的にみてもバグの生じたモジュール群の方が実行文の割合の多い側へシフトしている。図10はコメント文について同様の比較をしたもので，バグの生じたモジュールの方がコメントの割合の少ない方へシフトしている。

```
==========  BUG MODULE NI OKERU PROC NO WARIAI  ===
割合 (%)        個数
0               11  ****
 0 - 5          12  *****
 5 - 10         1
 10 - 15        2   *
 15 - 20        5   **
 20 - 25        8   ***
 25 - 30        4   **
 30 - 35        9   ****
 35 - 40        9   ****
 40 - 45        19  *******
 45 - 50        35  **************
 50 - 55        47  ******************
 55 - 60        60  ************************
 60 - 65        91  ************************************
 65 - 70        84  *********************************
 70 - 75        63  *************************
 75 - 80        29  ************
 80 - 85        23  *********
 85 - 90        1
 90 - 95        0
 95 - 100       0

==========  TOTAL MODULE NI OKERU PROC NO WARIAI  ===
割合 (%)        個数
0               439 ***********************
 0 - 5          35  **
 5 - 10         22  *
 10 - 15        33  **
 15 - 20        43  **
 20 - 25        57  ***
 25 - 30        66  ***
 30 - 35        100 *****
 35 - 40        148 ********
 40 - 45        246 *************
 45 - 50        306 ****************
 50 - 55        424 **********************
 55 - 60        471 *************************
 60 - 65        452 ************************
 65 - 70        398 *********************
 70 - 75        316 ****************
 75 - 80        171 *********
 80 - 85        97  *****
 85 - 90        42  **
 90 - 95        5
 95 - 100       1
```

図9　実行文の割合とバグの関係

```
==========  BUG MODULE NI OKERU CMNT NO WARIAI  ====
割合 (%)        個数
0               45  ****
 0 - 5          247 ************************
 5 - 10         173 *****************
 10 - 15        56  *****
 15 - 20        14  **
 20 - 25        9   *
 25 - 30        3
 30 - 35        0
 35 - 40        0
 40 - 45        0
 45 - 50        0
 50 - 55        0
 55 - 60        0

==========  TOTAL MODULE NI OKERU CMNT NO WARIAI  ====
割合 (%)        個数
0               691 *********
 0 - 5          1251 ****************
 5 - 10         1007 *************
 10 - 15        491 *******
 15 - 20        251 ***
 20 - 25        113 **
 25 - 30        42  *
 30 - 35        14
 35 - 40        6
 40 - 45        4
 45 - 50        2
 50 - 55        0
 55 - 60        0
```

図10　コメント文の割合とバグの関係

これらの結果はまだ評価できる状態ではないが，いままで行われてきた各種の研究も参考にして，調査・検討を進めていきたい。また単にバグの件数ではなく，どのようなバグであるかのバグの性質についての検討も重要な課題であると考えている。

5．あとがき

本報告では，国鉄で進めているソフトウェアの開発・保守の効率化のための活動のうち，モジュール分析の試みについて報告した。まだデータの整理が十分でなく，有効な結果が得られたとは言えない状況であるが，より一層検討を進めたい。また，このようなデータの分析も継続して行なえるような環境を整備していくことが重要な課題であると考えている。最後に，本研究を進めるにあたって協力，助言をいただいている国鉄本社，鉄道技術研究所，東京システム開発工事局，中央情報システム管理センターの関係各位に感謝する。

〔参考文献〕

(1)花田他，：プログラム構造の複雑度の分析と評価，情報処理学会論文誌,Vol.23,No.6,pp.701-706,1982

(2)桑名他，：プログラムの複雑度の分析と評価，情報処理学会ソフトウェア工学研究会，32-3,1983

(3)ELSHOFF,J.L.：An Analysis of Some Commercial PL I Programs,IEEE,Trans.,Vol.SE-2,No.2,1976

(4)McCabe,T.J：A Complexity Measure,IEEE Trans.,Vol.SE-2,No.4,pp.308-320,1976

(5)FITZSIMMONS,A. et al：A Review and Evaluation of Software Science,ACM Computing Surveys,Vol.10,No.1