Mem. Fac. Sci. Eng. Shimane Univ.
Series A
31, pp. 243-258 (1997)

プログラム変数の命名
～変数はどの様に命名されているか～

佐藤　匡正
数理・システム情報学科
総合理工学部　島根大学

An Analysis on the Methods of Naming Program Variables
～How are Program Variables Named?～

SATOU Tadamasa
Department of Mathematics and Computer Science,
Interdisciplinary Faculty of Science and Engineering Shimane University
(Received September 19, 1997)

ABSTRACT

Variable names in program source codes provide for important hints in reading programs. Reasonable names make maintenance works promote. Naming conventions are required to be established. This paper presents a quantitative analysis of program variable names in commercial use programs. Through the analysis the construction and abbreviation of words that are included in a variable name are focused.

As the result the following items are clarified;

i) major factors for naming conventions

ii) variation range within naming

1. 序　　論

変数の名前をどのようにつけるかはプログラミングにおいて重要な事項のひとつである．変数の名前は，プログラム・コードの主役ともいえる存在である．分かりやすい名前がつけられたプログラム・コードは妥当性の確認や保守がしやすくなる．

一般に，変数は値の型と名前で定義される．名前には，管理面と生産技術面の二通りの立場がある．管理面の立場では，システムで使っている名前の衝突を回避し混乱を防ぐことにその目的がある．一方，生産技術面の立場では，処理記述に意味を与え難解なコードを分かり易くすることを目的とする．名前に変数のもつ役割や値の意味を盛り込めば，変数名がソース・コードの解読の有力な手がかりとなるので読み易くなる．

変数に名前を与えるのはプログラマの役目で，自分のためにも効果的な名前を付けようと苦労する．しかし，苦労してつけた名前も必ずしも適切とはいえない．名前を付けるための具体的な基準があれば効果的である．しかし，現状ではこのような基準は十分でなく単なる

基本方針にとどまっている．つまり，一読して分かる名前をつけねばならないとか，語の綴りの略し方を一定し，曖昧にならないようにすべきだとか，長い変数には句切りを示す，接頭語としてデータ型を使えといった留意事項や提案[1),2),3),4)]の類いはあるが，具体的な基準への展開は十分とはいえない．

こうした提案は，提案者の長い経験からの賜であるから尊重すべきなのであるが，基準として具体化するとなる内容が問題となる．つまり，内容が指針の域であるから実施するためには，更に具体化せねばならないこと，現状をどの程度踏まえているのかといった問題である．この具体的な基準とは，例えば，変数の長さはどのぐらいにすべきであるかとか，「カウンタ」はどの様な綴りにしたらよいか，複数の語を複合させたものはどの様に名付ければよいかといったことが規定できるものである．このような具体的な基準がないためにプログラマ個人の基準に任されている．この結果，不要な個人差を生じさせ，ソース・コードを難解なものにしている一つの要因ともいえる．

先に述べたように変数は，ソース・コードの解読の重要な手がかりとなる．したがって変数の役割やその値の意味を的確に表すような，個人差のない名前の付けられる指針や基準があれば，コーディングにおける苦労は軽減され，保守性は改善される．このような基準を設けるための要因を明らかにするために，実際のプログラムを調べてみた．ここではこの調査における分析結果を報告する．

2. 命名の背景

変数名は様々な事項を背景にして名付けられる．この要因を体系化するために，命名の過程をモデル化して考えることにする．このモデルを述べるためには変数名の種類を明らかにしておくことが必要で，これについてまず最初に述べる．次に，モデルについて述べ，そして，命名するにあたっての主要な要因である，命名基準，表現方法，及び省略方法について，その概説を述べる．

(1) 名前の種類

変数名についての明確な分類は確立していないが，論を進める都合から三種類に分ける．これらを，名前の特徴から便宜的に，①識別名，②連想名，③序数付き，と名付ける．識別名は，名前の桁文字個々がコード体系をなすもので，桁毎に特定の意味をもつ．文字のつながり自体には意味をもたない．連想名は，文字のつながりによって特定の意味を連想させる名前である．序数名は，用途を同じにする変数に対する名前で，識別のために序数をもつ．用途を表す部分は共通でこれに識別のために，01, 02 とか A, AA とかいった序数を加える．以上の名前の分類を表 1 に整理する．

プログラミングではこれらの名が組み合わされて用いられるが，この基本は連想名である．ここでの分析では，連想名を対象とする．

(2) 命名過程モデル

連想名を命名する過程をモデル化して，命名にあたっての要因を明らかにする．

命名の過程は，6 段階に分けることができる．まず，変数の役割を考え，この役割を表す

表1 命名法の分類

名前の種類	概　　要	例
識別名	桁構成にコード体系を重ねる。 桁毎に特定の意味をもつ。	AMcalicul └モジュール識別 └プログラム種別
連想名	文字のつながりで特定の意味を連想させる。	stk …スタック
序数名	同じ用途を番号で意識する。	wk1, wk2 …作業用

いくつかの鍵語を頭に思い浮かべる．この変数の役割とは，変数の値の用途や特性である．次に，この役割を日本語の文句で表す．この文句は，一般に長くなるので縮める．通常は名詞の連語，つまり名詞句にする．そして，更に短くするために語を選ぶ．語の選び方には，単一語とする場合と，いくつかの語を複合させる場合がある．

こうして選んだ語を英字で表現するために，英訳するか，ローマ字で表す．さらに，この綴りを縮めて変数名とする．もちろん，最初から英語の文句で考えることもあるし，日本語のままで仮名混じりの漢字とすることもある．しかし，命名の過程は同様である．図1にこの命名過程のモデルを示す．

このモデルによって，次の三項目が名前を決める要因であることが分かる．

① 鍵となる代表語の選択
② 名前の構成
③ 綴りの短縮

例えば，「部品の使用頻度を数える」変数に対して名前を付ける場合を想定する．この変数の役割，つまり，値は「頻度」である．従って，部品，カウンタ，使用頻度という用語が頭に浮かぶ．これを整理して，「部品の使用頻度を数える」という日本語の文ができる．これを短くするために連語にする．名詞句「部品使用頻度の計数」となる．この代表語として例えば，部品とカウンタを選んで英訳すると，parts counter という英語の合成語ができる．この綴りを縮めれば，partsCtr や PCTR という変数名ができあがる．この名前に，型や序数を添えると，intPartsCtr や，partsCtr2 といった名前ができる．ここで，int は，整数型（integer type）に因む．

この例の intPartsCtr という名前において，上であげた三項目の基準は次のようである．まず，代表語について，ここでは，部品とカウンタとした．もちろん，頻度を選べば別名となる．名前の構成を型名と連想名としたので，整数型を表す int で始めている．また，語の区別は大文字と小文字の文字種別によっている．最後に，綴りについては，parts はそのままであるが，counter は縮めて ctr としてある．

(3) 命名の基準

変数名の命名規則について現状を横観し，整理を試みる．

複数の要員の参加による開発プロジェクトでは，作業標準の一環として命名の付与基準を

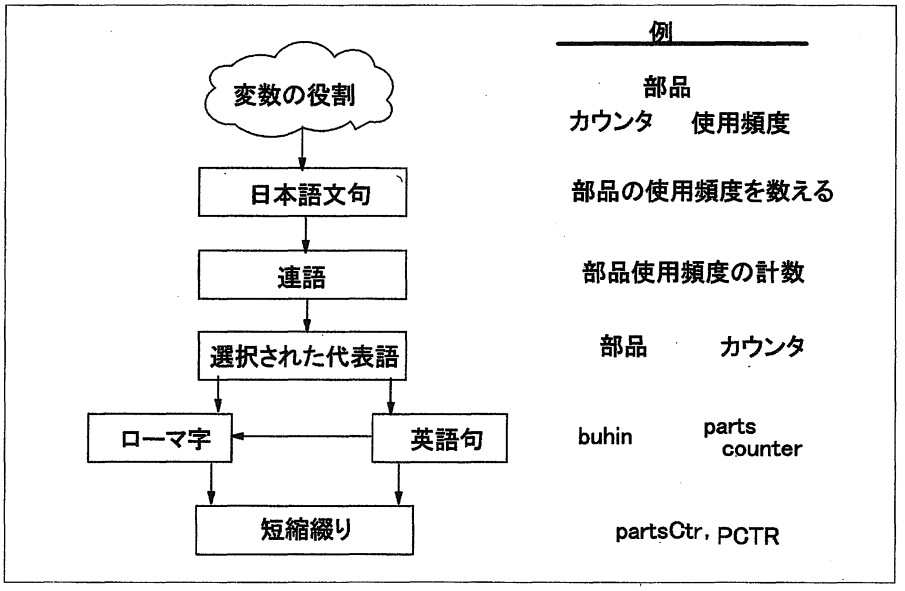

図1 命名過程のモデル

設けるのが普通である．この基準は管理を目的とし，名前の衝突を避ける目的で，プログラム名，システムで共通に使う制御テーブル名やそのエントリ名，データベースのデータ名などが対象となる．こうした名前は主として識別名が使われる．プログラムの変数名については，必ずしも明確な基準はない．また，こうした基準はプロジェクト内部の資料であり，公表されないのが普通である．

公表されたものとして，ハンガリアン命名規則[4]と文献3の規則があげられる．前者は，接頭語による命名規則である．接頭語は値の型で，ポインタとか，ハンドルといった予め決めてある2〜3文字長である．これを付けて名前にするものである．後者は，変数の命名規定に関連する著者の経験を随筆的な読み物として述べたものである．この内容は，次のように整理できる．

① 定数名を使う
② データ型名を付ける
③ データの型に応じて品詞を選ぶ
→手続き名‥動詞，変数名…名詞，論理型‥形容詞/過去分詞
④ 長さは八文字程度
⑤ 省略方法を決める
→先頭から四文字程度/子音を主体として選ぶ

また，関連してデータベースのデータ項目名については変数名と似ている．ただし，名前は処理寄りでなく，利用者向けの点が異なる．この命名には，ドゥレルの命名規則[5]に基づく規則が採用されている．この規則では，処理の内容が理解できない利用者にも分かるよう

に名前を区分し，三要素，主要語，修飾語，およびコードで構成する．構成の順序は，修飾語→主要語→コードとし，主要語の詳細を修飾語及び値の型であるコードで補足する体系である．この規則に基づいて名前の標準化を自動化する試みが提案されている[6]．

(4) 言語の選択

プログラムは機械に対する指示書であるから文書といえる．プログラムを記述している言語は英語が基本であるから，変数には英語を用いるのが自然である．また，国際的な標準化という観点からも英文にする方が好ましい．変数名だけが漢字混じりの日本語では文書としてバランスが悪い．

英字で日本語を表現する方法としてローマ字表現がある．この表現方法は，英文の文書という観点からは問題であるが，現実には広く用いられている．問題点として，①カナ語の表記方法，②表記方法の違い（ヘボン式，訓令式，日本式），があげられる．カナ語は，ビルとかパソコンといった語の表記方法である．本来語の綴りとするか，そのままローマ字で綴るか判断を要する．表記方法に関しては，方式の違いに加えて，母音と重なる場合の表記に多様性が生ずる．例えば，佐藤をローマ字で書くと，次となる．

「satō, sato, satou, satoh」

因みに日本国の外務省は，ヘボン式で「sato」とせよとしているが，この指示は適切とは言えない．本来は「さとう」であり，「さと」ではないから「satou」とすべきである。また，ローマ字綴りの省略方法は英語と同様に母音を略す．しかし，ローマ字には同音異義語が多いから英語より混乱する．例えば，TRK は，登録にも取引にも相当する．

(5) 語の種類

変数名に用いられる語は，そのプログラムに関係する分野の用語に因んでいる．関連する分野とは，問題分野とシステム分野である．問題分野は，そのプログラムの適用される問題に関連する領域であり，システム分野は，プログラムの動作を支えている環境に関する領域

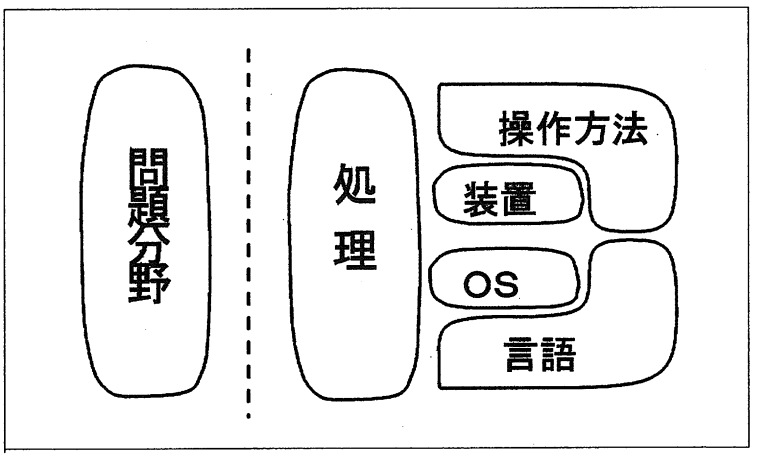

図2 用語と分野

である．具体的には，処理，装置，その操作法，0S，パッケージである．この様子を図2に示す．この分類における用語の例をあげる．問題分野の用語は適用分野によって変わるが，大学における教務処理ならば，学籍番号，成績，単位履修といった用語である．これに対して，システム分野は共通である．代表的な例をあげる．処理では，初期化や，チェック，フラグ，ポインタが，装置では，カーソルや色が，操作法では，メニューやコマンドが，0Sではファイルやディレクトリが，パッケージではその製品固有の用語が使われる．

(6) 語の句切り

変数名の構成要素は，語，序数（1, 2, 3..; A, B, C,..），句切り記号（_, −, ., ·, など）である．語の句切りは，構造体のように言語仕様で句切り記号の使用を強制するものと，書き手が意図して句切り記号を使うものとがある．また，この句切り記号を避けるための工夫として，大文字と小文字の使い分けがある．既に述べたpartsCtrがこの例である．

(7) 綴りの縮め

名前の綴りが一定の長さ以上になると扱い難くなる．プログラミング言語における長さの制約はもちろんだが，制限がなくとも問題は生ずる．条件付けや繰り返しの制御構造の記述では字下げが行われる．こうすると，あまり長い変数名は一行で収まらなくなる．特に，入れ子が深くなると，こうした可能性は高くなる．更に，記憶に止めるにはあまり長くない方が好ましいとも言われている．こうした理由で綴りの短縮化が行われる．短かくするにあたっての要件を整理すると次のようになる．

① 曖昧でない
② 意味が分かりやすい
③ 簡潔である

この短縮の方法として自然語（英語）における綴りの短縮方法が基本である．語レベルでは，先頭から前部の音節を省略する前部省略と，最終からの後部音節を省略する後部省略がある．前者の例としては，photoやplaneが，後者はexamやmathがある．また，fridge (refrigerator) のように中間の音節を拾うやり方もある[7]．連語の場合は，頭文字を組み合わせるやり方は頭文字語と呼び，一語として読むものはアクロニムacronymという．これらはU.S.A.とNATOが代表的な例である．変数名についてもこうした縮め方は行われるが，この他，特に母音が省略される．語の綴りの縮め方を整理すると，表6の様になる．

綴りを縮めるにあたっての条件を満足させるために留意すべき点は，次に要約できる．

① 語の選択，

表2 短縮の目的と要因

目的	要因
曖昧でない	単語の種類
意味が分かりやすい	語の長さ
簡潔である	略し方
	日本語，英語の選択

② 語の長さ,
③ 略し方（語の綴り字のどの字を残すか),
④ 日本語か英語か（ローマ字綴りか英語綴りか）の選択,

これらの要件と要因の間には関係がある．例えば，曖昧さは語の長さと略し方に関係する．この関係を表2に整理する．

3. 現状の調査

第2章で述べた変数の命名が，現状においてどの様であるかを調査する．この調査方法と調査の結果，および結果の考察について述べる．

3.1 調査方法

(1) 調査の目的

変数命名の現状を把握し，命名に関してその目安や問題点を明らかにする．調査の普遍性に配慮する．

(2) 調査項目

基本的には，第2章の述べた命名の背景に添って調査を試みる．つまり，変数名の構成，語の種別，略し方について調査する．

(3) 調査の対象

分析結果の普遍性に配慮して，調査の対象は，ツールや個人的に使用するプログラムでなく，実際に企業で用いられているプログラムとし，この中で用いられている変数名を分析する．また，分野に偏らず，また結果の違いが比較できるように，システム寄りと業務寄りのプログラムを選ぶ．この考えに添ってシステムAとシステムBの二システムのプログラムを対象とする．システムAはシステム寄りのプログラムで，ソフトウェア開発支援機能を実現したものである．システムBは業務寄りで，企業における営業業務システムの一部である．プログラムの規模は，両者を合わせ59のモジュールで，約19千行である．この諸元を表3に示す．

表3 ソース・プログラムの諸元

特性		システムA	システムB	備考
システムの用途		ソフトウェア開発支援システム	営業システム	
規模［千行］	有効行数	11.4	7.86	コメント行や空白行は除く
	モジュール数	39	20	コンパイル単位
記述言語		PL/Iサブセット	C	
要員数［人］		5	3	直接作成者

(4) 調査の進め方

基本的にはソース・プログラムから変数名を取り出して，これをデータベースに取り込み分析する．次の順序による．

step. 1. ソース・プログラムを構文解析して変数名を取り出す．

step. 2. 取り出した変数命名の対象を絞る．
…モジュール名，同一の名前を除く．

step. 3. 変数を部分に分け部分変数を取り出す．
…句切り記号で部分化し，同一の名前を除く．

step. 4. 部分化した変数名の語構成を調べる．

step. 5. 分類し，集計する．

3.2 変数の全般的な性質

本節から第3.5節までに亙って分析の結果を述べる．最初に，全般的な事項として，変数の数と名前の長さの出現頻度について述べる．次に，変数の構成に関する事項，続いて語の種別，最後に綴りの縮め方について述べて行く．

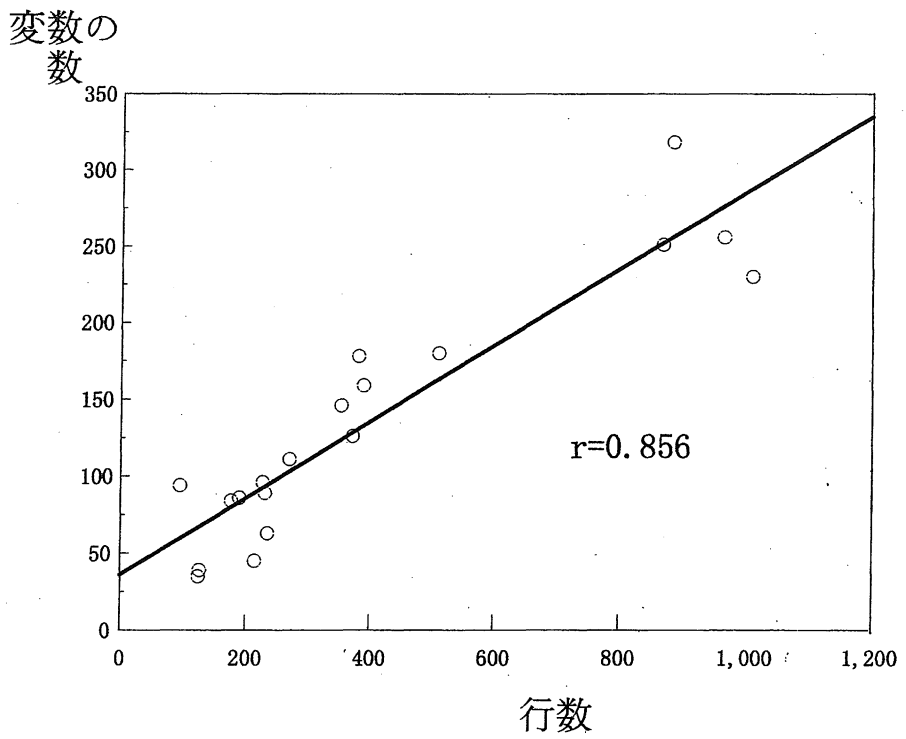

図3 モジュールの行数と変数の数

(1) 変数の数

表5に示す様に，システムAでは，全体で4,547個の変数が使われている．システムBでは，2,675個である．これらにおける同一名を削除すると，この数は減って，それぞれ1,646 (36.2%), 935 (35.0%) となる．この数は，ほぼ変数の定義（宣言）数である．

モジュールの行数とここで使用されている変数の数を，システムBに関してプロットしたものが，図3である．両者には相関が認められる．図から特にモジュール行数の比較的少ない方に相関が認められる．ここに着目すると，10行あたり3～4個の変数というのが目

表4 句切りパターンの出現状況

長さ	区切り方	分割数					
		1	2	3	4	5	6
1	1	13					
2	2	39					
	1-1		16				
3	3	162					
	2-1		19				
	1-2		66				
	1-1-1			9			
4	4	54					
	1-3		55				
	3-1		33				
	2-2		12				
	1-1-2			14			
	1-2-1			4			
	2-1-1			2			
	1-1-1-1				5		
5	5	3					
	3-2		3				
	2-3		2				
	4-1		3				
	1-4		2				
6	3-3		2				
	4-2		1				
	1-2-3			3			
7	2-5		1				
	4-3		1				
	3-1-3			1			
8	2-1-1-1-1-2						6
合計		271	216	33	5		6

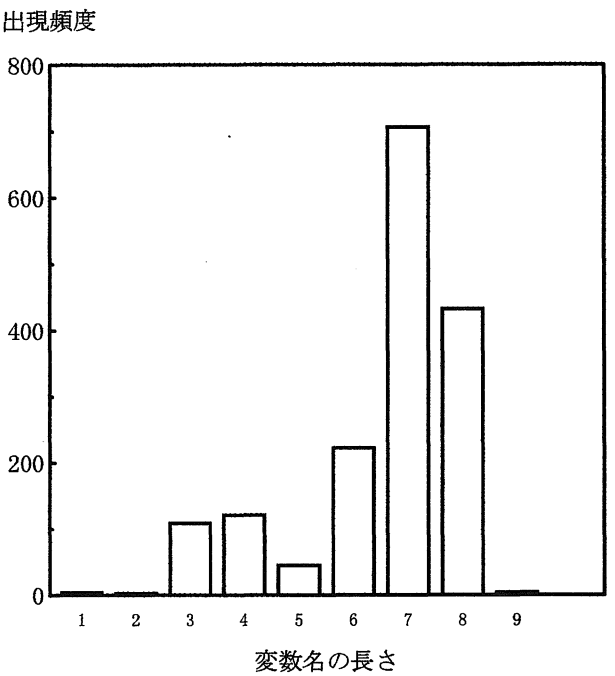

図4(1) 変数名の長さの出現頻度
(システムA)

安である.

(2) 名前の長さ

長さは文字数である.この出現頻度を,図4(1),(2)に示す.(1)はシステムAで,(2)はシステムBである.システムAでは,長さ7が最も出現頻度が高い.長さ6~8に集中して分析している.これらで全体の83%を占める.なお,9以上の長さのものがない理由は言語仕様上の制限からである.一方,システムBでは,長さ14のときの出現頻度が高いが,これを除いて考えると,長さ6~10に集中している.両者から変数名の長さとしては,6~10が標準的な値と言えよう.

3.3 変数の構成

変数名を構成している要素変数の数,長さ,句切りパターンについて述べる.

(1) 要素変数の数

変数の構成について,いくつの要素変数から成るかを表7の合計欄に示す.単一語の変数名は半数271件 (51%)で,残りの半数は要素変数をもつ.要素変数の数については,2個のものが41%を占める.要素変数名で長さ1の語を含むものが92%ある.つまり,長さ1の名前は要素変数用の縮め方が大部分であるといえる.

(2) 要素変数の長さ

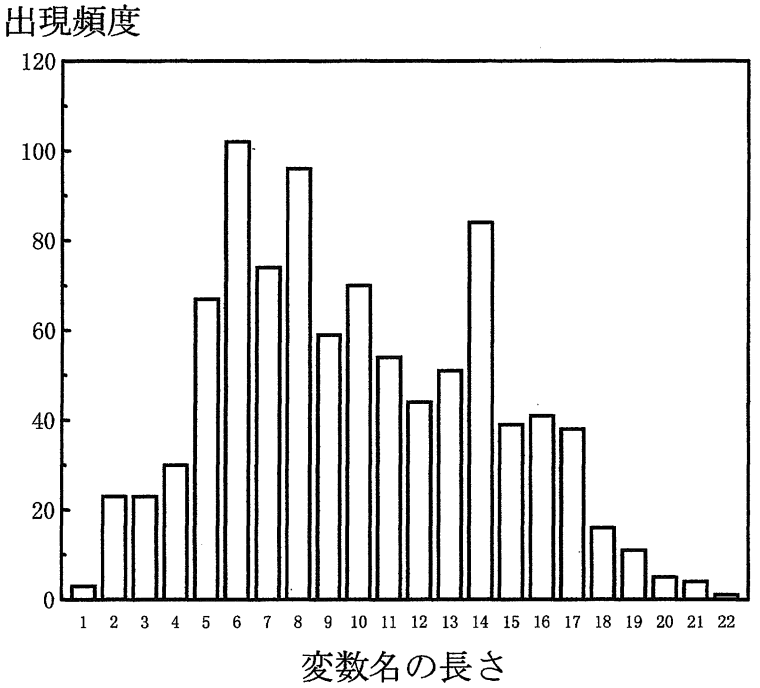

図4(2) 変数名の長さの出現頻度
(システムB)

要素変数名の，システムBにおける出現頻度を図5に示す．これによれば，長さ3が最大の出現頻度を示している．一方，システムAについても，やはり長さ3に集中している．この様子は表5の合計欄に示す．

(3) 句切りパターン

変数名が，その構成する要素変数名に容易に分解できると理解するのに役立つ．この分解は，句切り（記号）のあるときは問題はないが，ない場合はどこで句切るかは一通りではなく，様々な可能性が考えられる．句切りが成立するか否かは，句切った部分が要素変数として意味をなすかどうかによる．句切りの可能性は1が望ましい．複数あれば分かりにくい要因になる．

例えば，CHNGはCHaracter NGとも，CHaNGeとも，Code HaNGupとも解釈できる．システムAにおいて，変数名の長さと句切りの可能性の数の関係を表5に示す．この表に示した以外の句切りパターンは出現していない．要素変数の数は2が多いが，全般にみて前半よりも後半の方を長くする傾向が想像される．接頭語に型の名前を付す考えがあるためと想像される．

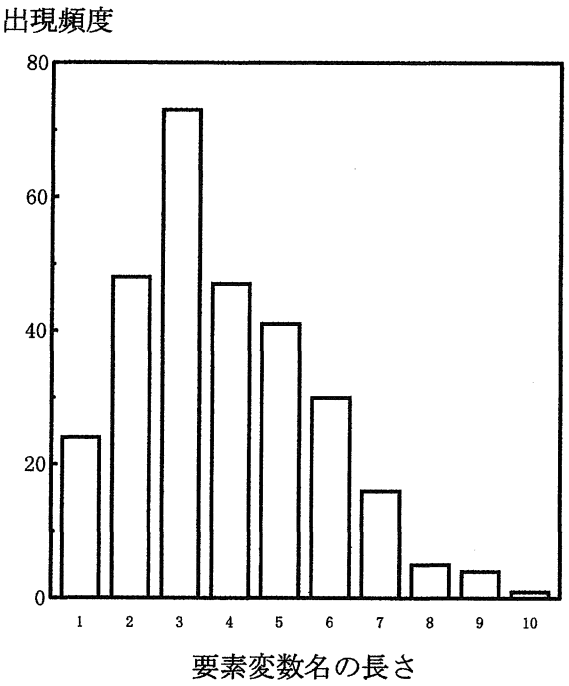

図5 要素変数名の長さの出現頻度

表5 変数名とその出現数の変化

名 前	内 容	出現数	
		システムA	システムB
検出した変数名	ソースコードから取り出した変数名	4547	2675
対象の変数名	同一の名前，モジュールの名前を除く	1646	935
部分化変数名	変数名を部分化して，同一名を除く	531	478
要素変数名	部分化変数名から序数などを除く	437	289
用 語	略し化の異なるものを一つにする。	237	239

3.4 変数と言語

変数名の表現として，英語が自然であるが，この実態を次に示す．また，変数名に用いられている用語についての状況についても示す．

(1) ローマ字表現

システムAについて，ローマ字は8.1%で残りの9割強は英語の綴りである．一方，システムBでは，30.0%である．これは，業務用のデータベースのデータ項目名に，伝統的にローマ字が使われてきた事情による．

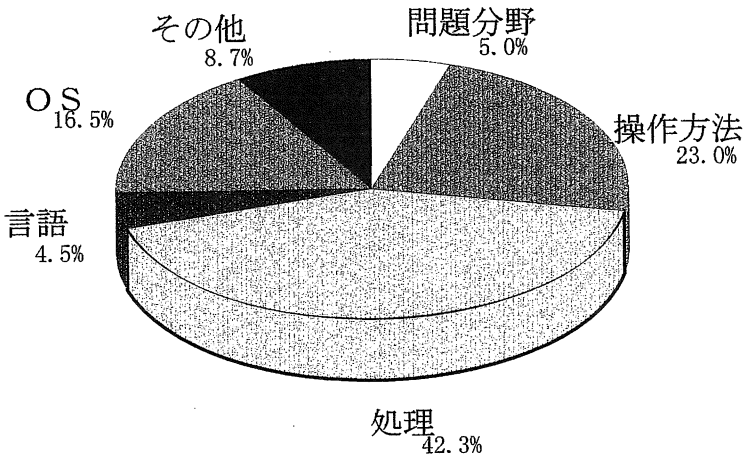

図6 分野ごとの用語の出現状況

(2) 用語の種類

表4に変数名と，この名前の背景となった語の種類を示す．表中の用語とは，綴りの縮め方の異なるものを一つにしたものである．つまり，BUF, BFR, BF を BUFFER に統一して数えたものである．用語の種類は，全変数の数に対して，システムAでは約1/7で，システムBでは約1/2である．これは，システムAは，システム向けなので用語の種類が少ないことによる．この詳しい考察は次で行う．

(3) 分野ごとの語の出現割合

上の用語が，図2においてどの分野のものであるかを図6に示す．図はシステムAに関するものである．これによれば，処理の語が42.3%を占め，問題分野は5.0%である．しかし，システムAは操作法に関するプログラムを含んでいるから，操作法の用語は問題分野と位置づけられる．これを合わせると，28%となる．

用語を，問題分野とシステム分野に大別して考える．問題分野は66で，システム分野は171である．システムBについては，それぞれ38.1%, 61.9%であるから，91と148となる．これから，問題用語はプログラムによって，その数に差があるが，システム用語については，目安として150～180程度と想像される．

3.5 変数名における綴りの縮め

語の綴りを簡潔にするために縮めるが，縮め方によっては分かりにくくなったり曖昧になったりする．現状において行われている方法について，縮める長さ，縮めの多様さ，曖昧さとして衝突状況を調べる．

(1) 変数名の縮め方と長さ

変数名の縮め方は様々に行われているが，代表的と考えられる6通りの縮め方を，その長さごとに出現頻度を数えて表6に示す．表中の「例」欄に ONLINE をオリジナルとして

表6 縮め方とその出現頻度

縮め方	例（ONLINE）	長さ					計
		1	2	3	4	5	
頭からn字	ONL	100	24	78	16		218
頭部と尾部のみ	ONE		27	75	4		106
頭部と子音	OLN	8	11	6	2		27
慣用	ON		5	12	16		33
思い付き	NLN	3(1)*	3(2)	6(4)	2(1)	1(1)	15(9)
省略しない	ONLINE		2	13	20	3	38
合計		111	72	190	60	4	437

*：綴り誤り分の再掲

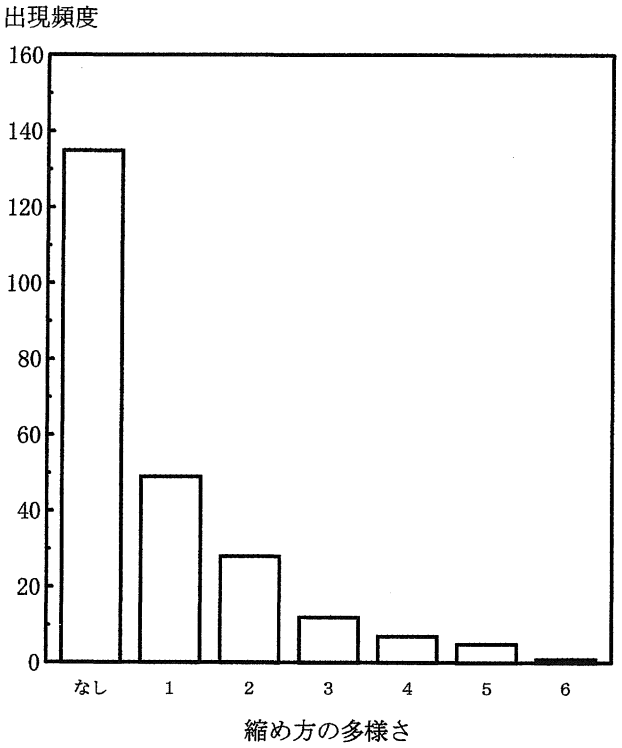

図7 縮め方の多様さ

どのように縮められるかの例を示してある．本表はこれらの縮め方についてまとめたものである．この結果，先頭からn文字取り出す方法，および先頭と尾部を取り出す方法を合わせると全体の約3/4を占める．

表7 縮め方と衝突数

長さ	衝突数										合計
	2	3	4	5	6	7	8	9	10	11	
1	6	3	1	4	1	1	1		1	3	21
2	7										7
3	8	1									9

なお，loop を roop とするといったような英語の綴り誤りは2%であった．

(2) 縮め方の多様さ

綴りの縮め方は一意ではなく，人や状況によって様々に縮められる．この様子を知るために，縮め方のバリエーション数の出現頻度を図7に示す．ここで，バリエーション数は，用語の縮め方としての事例数を言う．第3.4節(2)項で述べた BUFFER の例では3である．0は他の縮め方のないことを意味する．図は，このバリエーション数に基づいて出現頻度を数えたものである．バリエーションのあるものは43%で，バリエーション数の最大は6である．

(3) 衝突

縮めた名前が別の名前と衝突すると，解釈に曖昧さを生ずる．この状況を知るために，表7に変数名の長さと，衝突数の関係を示す．長さが短いと衝突の可能性も高い．長さが1では11個の語と衝突する場合が3件ある．長さが4以上では衝突する場合はない．つまり，長さを4以上にすれば衝突は避けられると言える．

(4) 本来の綴りのままの名前の長さ

綴りを縮めないでそのまま変数名とすればコードは見やすくなる．特に他人のコードを読む場合には効果的である．この反面，煩わしくなるといった長いことの問題点（第2章(7)項）もある．綴りをそのままで変数名とするものは全体の8.7%(表6) あるが，その長さが4以下のものが92%である．つまり，現状では綴りを省略しないでそのまま変数名としているのは長さが4以下の語であるといえる．

4. 結　　論

今回の調査分析をした変数名の命名法は次のように整理することができる．

(1) 全体の名前の長さは，6～10文字である．2から3の要素変数をもつ．

(2) 要素変数の長さは3～4文字である．

(3) 要素変数をもつ複合した名前では句切りが必要である．つけないと句切りに多義性が生じ曖昧になる可能性が高い．

(4) 変数名の背景となる用語には，問題分野に関するものとシステムに関するものとがあるが，前者における用語の数は問題領域によって異なる．後者は目安として150程度と推定される．

(5) ローマ字が使われている．また，綴りの誤りも存在する．
(6) 綴りの縮め方は，先頭から何文字かを取り出す方法が多く使われている．
(7) 縮めると衝突の可能性が生ずるが，4文字以上だと避けられる．

参 考 文 献

1) Christine N. Ausnit, et al.: Ada in Practice, Springer-Verlag (1984) ISBN 0-387-96051-1.
2) 佐藤：プログラム変数の命名法，ソフトウェア工学研究会58-15, (1988).
3) TSINKY：プログラミング・セミナー△名前のつけ方，bit, pp 1206-1210, Vol. 11, No. 12.
4) 富士ソフトウェア教育出版部：OS/2 プレゼンテーション・マネージャ (1989), ISBN 4-938455-09-9 C30SC.
5) William R. Durell: Data Administration, データ資源管理（味村重臣監修）日経マグロウヒル (1987).
6) 関根他：体系的なDB構築のための用語辞書を用いたデータ標準化手法，情報処理学会論文誌 Vol. 34 No. 3 (1993).
7) 森戸由久：ヒアリング上達法，現代新書講談社 (1986).