第5回ソフトウェア工学国際会議報告（II）

木村　泉
東京工業大学・理学部

0．はじめに

San Diego の 5th International Conference on Software Engineering（ICSE 5）の概要については別に片山氏から報告があるはずになっている。ここでは片山氏とともに会議に参加した者として、同氏の話とはなるべくちがう視点から有益と思われる情報をご提供したい。ICSE 5 はパラレルセッションが三つあり、どうがんばっても講演は全体の1/3しか聞けない仕掛けである。その上、今回は会議出席は添え物で、ほかにたくさん仕事があったため、網羅的な報告はできない。ここではわが国でまだあまり本格的に取り挙げられていない領域の中に重要ないし有益であるようなものはないか、という目で話題を拾ってみることにする。

なお、同じ会議の第6回（ICSE 6）は1982年9月13-16日の4日間東京都文京区の学習院大学で開かれる予定である。そこで本文の終りの部分では、このチャンスを生かすための若干の提案をしたい。

1．問題領域の例、その1——人間工学領域

"Human Engineering" のセッションは University of Texas の R. Brooks が座長であった。三つの発表があり、必ずしもびっくりするような結果は出ていないものの、新しい方向を示すものとして興味深かった。

1．1　プログラム設計の記述方法に関する実験的研究

Sylvia B. Sheppard ほか（General Electric 社）の The Effects of Symbology and Spatial Arrangements on the Comprehension of Software Specification は、ごく簡単にいうとプログラムの設計を記述する方法として日常語、擬似コード、流れ図のいずれがよいか実験的に調べてみたものだといえる。心理学的研究の結果がつねにそうであるように早合み込みは危険であるが、一応結論は擬似コードがベストで日常語（この場合は英語）による記述はだめ、という風に読める。

プログラミングにおける流れ図の役割については、使ってみても大したプラスにならないことを暗示するような結果が2，3出ている。一方プログラミング以外の領域での心理学的研究では、文章で書くより流れ図の方がわかりやすい、ととれるような実験結果がある。そこでこの問題を再検討してみたいというのが論文の趣旨である。

具体的実験では，記述の構成単位として

（1）英文による記述
（2）プログラム風の記述
（3）図形的記述

の3種を考え、それらをそれぞれ

（a）（プログラムのリスティングにあるように）たてにぞろぞろと並べること（Sequential）、
（b）（流れ図にあるように）実行の流れに沿って枝わかれしたグラフの形に

並べること（branching）、および
（c）下部構造を下にぶらさげた階層図の形式に並べること（hierachical）によって、プログラムのはたらきを表現する合計9種の文書を作り、それらを被検者に見せ、プログラムについての質問に答えさせる、という方法でおこなわれた。

質問としては
（i）プログラムの流れをたどってみればわかるという種類のもの
（ii）プログラムの途中のどこかを指定して、そこへやってくるための条件を問うという形のもの
（iii）入力を与えて出力を問う形のもの
の3種がもちいられた。

とりあげた問題は
（イ）弾道計算に関するもの
（ロ）在庫管理に関するもの
（ハ）空港におけるトラヒックのシミュレーションに関するもの
の3題であり、標準教科書から取られた。Fortranでコーディングした結果それらはそれぞれ50行程度になった。それらのFortranプログラムから出発して上記9種の設計記述を作った。被検者72人のうち54人はGeneral Electric社の社員、他は米軍関係者であって、経験年数は平均6.8年であった。

問題に対する答は計算機からのプロンプトに答えて端末から「急いで、だが正確に」打ち込まれた。

統計学的分析の結果は次の通りである。第1に誤答の割合は記法のいかんにあまり関係なかった。第2に応答の速さをたとえば問題の型（i）についてみれば、プログラム風の記述（2）は他の記述よりすばやく理解され、また空間的配列法についていえば枝わかれ形式（b）がもっとも速かった。問題の型（ii）についても結果は似たようなものであったが、型（iii）については、形式間の有意差は認められなかった。問題（イ）－（ハ）の間で応答時間にかなりの差があったが、これについては、同種の問題を解いたことがあるかどうかが大きく効いているとも解釈できる結果が得られた。経験年数および使ったことのある言語の数は多いほど応答時間が早まる傾向があった。

この発表は聴衆から大きな拍手を引き出した。ともかくはっきりした結果であり、またプレゼンテーションの手ぎわがよかったことも手伝っていたのであろう。

実は筆者としては、少なくともその場では「そりゃそうなるだろうけれど、だからといって、どうということもないなあ」と感じたが、もちろん筆者の感覚が正しい保証はない。そういう風に思った理由を反省してみると、次のようなことのようである。（イ）－（ハ）の問はいわば被検者を計算機の身にならせてみたらどうか、という話である。自然言語風、計算機言語風、記号風の中で一番計算機くさいのが計算機風の表現なのだから、それが一番成績がよかったとしても当り前なのではないかという気がするのである。それらの形式の文書を、最終的なプログラムを経由しないで作り出す手間を何らかの方法できちんと評価し、それとこの結果を関連づけてみれば、またちがった展望が開けるかも知れない。

そのほか、あえてこの研究に問題点を探すとすれば取り挙げられたプログラムが小さい、という点がある。これは心理学的研究で、つねに問題になることであ

る。Fortranで50行というのは決して大きいプログラムではない。一つのモジュールを見れば適正な大きさはこの程度だというのが著者たちのいい分であるように思われるが、そんなことでほんとうによいものかどうか、多少の疑問は残るようである。

とはいうものの、この種の研究は費用、手間とも大変にかかるものである。たとえば同じプログラムについて9種の設計文書が作られたわけであるが、それがたしかに完全に同じことを表現しているかどうか確認するために長い長い期間を要したという。また人件費も専門プログラマを何十人も使うわけであるから厖大な額にのぼる。（この研究の場合、被験者が学生でなく、一応プログラム書きでメシを食っている人々であるという点、大きなプラスといえる。）会場で座長Brooks氏が試算してみせたが、この研究の場合，本格的なコンパイラを一つ書くぐらいのマンパワーを要したことになっている。あだやおろそかではできないわけであり、得られた結果もぞんざいに扱うべきでないといえよう。

1．2　プログラムの単位分解法および注釈づけのわかりやすさへの影響

同じセッションの次の講演はS. N. Woodfield（アリゾナ州立大学）ほかのThe Effect of Modularization and Comments on Program Comprehensionと題するものであった。これはもとパデュー大学でおこなわれた実験の結果を述べたものでそのあらすじは次のとおりである。

同じ問題を解くFortranプログラムを

（1）サブルーチンわけを使わず、一枚岩風に書く（monolithic）

（2）機能ごとに独立のルーチンを作る（functional modularization）

（3）ルーチンあたり5－10行程度にこまかく分解してしまう（super modularization）

（4）原則的には（2）と同じで機能ごとに独立のルーチンを立てるが、プログラムのあちこちから共通的に操作されるデータ型については、それを処理するルーチンをENTRY文を使ってひとつにまとめる（abstract data type modularization）

の以上4種の方針にしたがって書く。そのそれぞれについて

（a）注釈のついたもの

（b）注釈のつかないもの

の2種、合計8個のプログラムを作る。注釈はそれぞれの単位分解形式に適合するように配置する。注釈のあるなしによるちがいをはっきりさせるため変数名などは無意味なものを使い、またプログラムの字下げはいっさいしないことにする。

被験者はプログラムのリスティングと問題集（20問）を与えられ、まず15分の間説明をきき、質問をすることを許された。次に60分間を与えられて問題を解いた。（論文のこのあたりの記述は必らずしも明瞭でない。プログラムの機能に関する説明がどういう形で与えられたか、説明が簡単すぎてあまりはっきりしないなど、筆者にはよくわからない点がある。）与えられた問の例として、

（i）ヒストグラムの印刷に際して、もしH(i)の値が0であったら第i行には何が印刷されるか？

（ii）プログラムが正常に終了するのはどんなときか？

などがあがっている。問題は「上から順に」やるものときめられており、プログ

ラム中の見なければならない場所があちこちピョンピョンとびまわることになるように配列された。答の評価は完全な正解と完全な誤答のほか、中間的な得点を許す形でなされた。

プログラムの内容は1種で、プログラムの単位分解に関するもの、とのよしであり、そういうものを選んだのは被験者がその問題についてよく知らないことがねらいであったという。(1)の一枚岩方式のプログラムの場合、文の数は11ノ個であった。　被験者は48人で、パデュー大学の大学院学生および学部上級生であり、1回75分のテストは2週間にわたり、計7回おこなわれた。被験者への問題の割り振りについては実験計画の手法が使われた。

結果はおよそ次のようなものであった。まず当然ながら、注釈のあるとないとでは、単位分解法(1)-(4)のいかんにかかわらず、注釈つき(a)の方が点数がよかった。驚いたことに、注釈がない場合、一枚岩方式(1)の方が機能別方式(2)および細分方式(3)より成績がよかった。もっとも成績がよかったのは注釈つき(a)、注釈なし(b)とも、データ抽象方式(4)であった。

統計学的取り扱いの結果、注釈の有無については危険率0.001、単位分解法については危険率0.22で、それぞれ有意差あり、との結果が出た。注釈と単位分解法の間の相関ははっきり出なかった。単位分解法による相違はおもに抽象データ型方式(4)がとびぬけてよいことに起因するものと判断された。

1.1の論文と同様ここでも、得られた結果はプログラミング経験者には容易に予測のつくものであるが、それを数量的実験的にきっちりさせるというのは大事なことである。

ちょっと気になるのは、例題として使われたプログラムが、書法という点から見て見るも無残な姿をしているという点である。これはさきに述べたように注釈の効果をはっきりさせるために意味のない名前を使い、字下げもやめてしまった、というのがおもな原因である。たしかに内容をよくあらわした名前を使えばそれだけでかなりの注釈的効果がある。

だが、データ抽象を使って書いたよいプログラムが読みやすいのは、プログラムが解くべき問題に内在する自然な構造をよく反映するからであり、そしてそれらの構造を人が把握するのは、主としてそれらに心の中で適切な名前をつけることによるのではないかと筆者は思う。(これはもとより実験的研究の結果によるのではなく、筆者のプログラミング経験に基づく直観である。)もしそうとすれば、この研究でおもに意味があるのは注釈つきの部分であって、注釈なしの部分は不自然なことをやっているのではないか、という気がする。

被験者が学生であって職業的プログラマでないこと、プログラム例が小さいことも、この種の研究においては避けがたいことをながら、気になる点ではある。

なおこの講演の講演者は、まじめに心理学をやらなくてはいかん、という演説を延々とやり、そのため肝心の技術的内容について述べる時間が食われてしまった。そういう話しかたは聴衆にきらわれ、また内容を低く見られて損と思う。

1.3　ソフトウェア工学における対照実験の改善方法

セッション3番目はミネソタ大学のThomas Moherらの*Methods for Improving Controlled Experimentation in Software Engineering*と題する発表であった。趣旨は、プログラマの作業状況に関するきっちりした実験をしようとすると

き被験者の質をどう評価したらよいかを考えよう、というものであった。講演者は「これはexperimentではない、studyである。」と強調していた。

具体的には

(1) 比較的短いプログラム(Fortranで51行)を読んで理解する問題(SR)、
(2) 比較的長いプログラム(Fortranで221行)を読んで理解する問題(LR)、
(3) 比較的簡単なプログラムを紙の上に書かせる——計算機は使わない——問題(SW)

の三つを100人の学生有志、および60人のプロ(職業的プログラマ)有志にやってもらい、一方彼らに経歴その他を書いてもらって、経験とテスト成績を統計的手法により比較対照したのである。

ちなみにSRで使われた問題は整数の列を与えられて最頻値を求めるもの、LRの問題は文書ファイルの中の語および文字の出現頻度を要約した表を作るものであった。SWでは整列(ソーティング)を含む問題が使われた模様である。

成績は、問題SR, LRについては

(i) このプログラムは何をするかというタイプの質問、
(ii) この入力を与えると何が出るかというタイプの質問、
(iii) 配列Xの大きさはなぜ50かというタイプの質問、
(iv) 文10はどういう条件下で実行されるかというタイプの質問、および
(v) もし文10を----と変えたら出力はどう変るかというタイプの質問

に答えてもらう、という形式で評価された。SR用の問題とLR用の問題の間には対応関係がつくように注意した。

SWについてはまず文の種類、頻度、構文誤り、使ったアルゴリズム、作成の所要時間などについて採点した。ほか、プログラムの入出力関係という点から見た正しさを一定の基準を設け、2人の採点者によって独立に評価した。

結果ははなはだ多岐にわたるが、おおそ次のように要約できよう。まずテスト結果のちがいを説明するに役立ちそうな要因が、経歴データの中から統計学的手法によって順に拾い出された。学生の場合、SRおよびLRの成績には計算機科学関係科目の成績および計算機科学関係科目修得数が主として効いているが、SWについてはこれらのほか、プログラミング経験年数も効く、という結果が出た。また職業的プログラマの場合はSR, LRに効くのは要するにプログラミング経験年数だけ、SWについてはそのほかに計算機科学関係修得科目数も効く、という結果が出た。

そこでこれらの要因と得点の関係を種々プロットしてみたところ、学生ではSR, SWとも修得科目数、科目平均点が増すにつれて成績がよくなることがわかったが、学生の場合、LRについてのよくなりかたは(たしかによくはなるが)SR, SWの場合ほど顕著でなく、最高ランクの者についてなお、正答率でSR, SWとかなりの差があった。これに対し職業的プログラマでは、SR, SW, LRとも経験年数とともに正答率が着実に増加し、経験6年以上についてはSRとLRの差がまったくなかった。標語的にいえば、プロは「長いプログラムでも平気」ということになろう。

なお著者たちは、SRとLRで成績に平行関係があったので小さいプログラムでもけっこうものがいえる、と安心したといっているが、この辺はいろいろ議論の余地もあろう。

これは大変おもしろい研究である。発表者たちのおもなねらいは、今後心理学実験をやって行く上で、被検者選びの基準として役立てたい、というところにあったようであるが、それだけでなく、プロ予備軍の養成方法などにも関連して考えさせられるところが大きい。

きっちりした実験ではっきりした結果を出すというのも、もちろん必要な、かついわば「かっこいい」方向であるが、このような地味な調査も、またたいせつと思う。むしろ、ソフトウェア工学の人間工学的方面においては問題があまりにも複雑であるだけに、かっこいい結果が出るとすれば僥倖であり、そういうことを1する間に「かっこわるい」地味なことを10やって行く必要があるのではないか、というのが筆者の感想である。

2. 問題領域の例、その2——会話型システム

あともう一つ、会話型システム（Interactive Systems）のセッションからNathan Relles（スペリー・ユニバック社）ほかの、A User Interface for OnLine Assistance を紹介しておきたい。この講演はプレゼンテーションがおそろしく凝っていて、わるい会話型システムの利用者がどんなに途方に暮れるか、という話をマンガつき、バックグラウンドミュージック入りでやり、聴衆をげらげら笑わせていた。

この話のUser Interface（利用者界面）はウィスコンシン大学に設置されたUnivac 1100、およびNASAのAmes研究センターのPDP-11/70（UNIX使用）で動いており、たとえばUnivac版はFortran 1800行（注釈含む）とアセンブラ1000行から成り、これを既成の会話型システムに組み込んで使う。

利用者が“?ERROR”というと、たった今出たエラーメッセージについての説明が出る。もう一度“?ERROR”というと、もっとくわしい説明が出る。さらにもう一度“?ERROR”というと、（システムに用意がある限りは）いっそうくわしい説明が出る、という調子である。“?QUESTION”というと、システム側からその直前に出された問い掛けについての説明が出る。もっとくわしい説明がほしければ、また“?QUESTION”といえばよい。このほか、正しい入力形式の例を求めること、術語の定義をきくこと、指令の打ち込み形式をきくこと、使える指令の一覧表を求めること、マニュアルの指定の節を打ち出すよう要求することなど、いろいろのことができる。“?ERROR”、“?QUESTION”などは、短縮形も定義できる。

出力される説明文は、システム側の人間（プログラマ）が書かなければならないことはもちろんである。そしてそれを助ける道具が、いろいろ用意されている。これで間に合わなかったらこれ、というように段々くわしい説明を出すには、それらの説明をセミコロンではじまる行で切って並べておけばよい。それを順々に出す仕事は「利用者界面」の側でする。また説明文に種々のパラメータ（数、ファイル名、マニュアルの節番号など）を埋め込むことができる。UNIX版の場合、説明文は標準の文書フォーマッタNROFFをもちいてきれいに整形してから出力される。

こういう手助けがあってもなお、よい説明文を書くのは紙に印刷したマニュアルを作るのと同じくらい大変だ、とあるが、それはそういうものであろう。一方、

そういう道具がなかったら、もっと大変になって、細ろくな説明文はできないとは経験の示すところである。

そのほか、利用者との間でどのようなやりとりがあったかを統計ファイルに記録することができる。その結果を参考にして、説明文を改良して行くことができる。

なお、この「利用者界面」は、もとの会話型システムと利用者の間に立ちふさがって仲を取りもつ形のものであるので、システムの原プログラムがいじれない場合には組み込めない、という泣き所をもっている。

この論文の考えかたはよくバランスの取れた、妥当なものであり、進むべき方向を示しているといえる。特に論文の中で、よい説明文書きのためのガイドラインとして掲げられているところがおもしろい。

a. 利用者の語彙を使って書け。
b. 説明文の最後にはできれば次に何をすべきかを書け。
c. 説明文を無理に短くするな。
d. 種々の要求が同じ説明文で満たされることもあることに注意。
e. 礼儀正しいことばづかいをせよ。擬人化した表現は不適当。利用者がまちがったといわず、システムにこういう制限があるといえ。
f. どんな説明サービスが用意されているかを説明するサービスを用意せよ。
g. エラーメッセージが出てないのに「いまのエラーメッセージはどういう意味?」ときかれたようなとき(空のメッセージの場合)は特に注意し利用者の身になって考えよ。
h. (利用者界面内部の問題として)説明文の入れ場所には使いかたに即した名前をつけよ。

の計8項目があがっているが、ことにgなどは実際にやってみなければなかなか思いつかないことを述べているものとして評価できる。

わが国では、会話型システムが本格的に普及しはじめたのは比較的最近であるので、こういうことがまだ十分理解されていないうらみがある。だが、こういうところがよくならないと利用者側のマンパワーまで含めた意味でのシステムの総合的な生産性を高めることはできない。

3. 付言——われわれ自身のプレゼンテーションについて

以上、こんな方向の研究はおもしろいのではないか、という意味あいでICSE5に出た話をいくつか紹介、論評してきた。本文を終るにあたり、ここで1982年9月に東京で開催されることになった第6回会議(ICSE6)に向けて、ちょっとした心おぼえを記したい。話題はわれわれ自身のプレゼンテーション技術に関することである。

近来、学会などでの日本語によるプレゼンテーションの質が目立ってよくなってきたことはよろこばしい。情報処理学会の大会などでも、耳できいてわかる話が大部分を占めている。当然のことであるが十数年前には必らずしもそうは行っていなかったように思う。

ところで多くの国際会議(たとえばICSE6)では、英語が中心である。そして残念ながら、英語による日本人のプレゼンテーションは、これも改善されつつあるとはいうものの、日本語による場合に比べれば未だしの感があり、なかなか

耳できいただけではわからないことが多い。プレゼンテーションの平均的水準の高い国際学会では聴衆がなかなかせいたくであり、ぶつぶつ文句をいわれるだけならまだしも、間々はじめから理解のための努力を放棄されてしまう。

ICSE6までにはこの状況を少しでも改善したいものである。それにはどうしたらよいか？　ICSE5の発表をきいていて、これまで気づかれることの比較的少なかった問題点として次のようなことがあるのではないかと考えた。

3.1. 文化的相違

日本人は武士道時代の名残りか、とかく大上段に構える。それがしばしばハンディキャップとなる。文化的相違に注意を払う必要がある。具体的には、

a. 一般論から説き起すことは一般にはきらわれる。アメリカ育ちの講演者の中にも、それをやって聴衆にそっぽを出て行かれてしまったのがいた。ずばりと本題に入った方がよい。ひねりのよい冗談で聴衆をげらげら笑わせる能力の持ち主の場合は一般論からはじめることも不可能ではないが、それ以外の場合は得策でない。

b. 同じ前の話だが、日本人がよく使う「まくら」はない方がよい。「議長、紳士淑女各位、本日ここでお話をいたす機会を得ましたことはまことに光栄であり」とか、「私は英語がまずくてすみませんが」とかは、むしろいわない方が効果的である。英語が下手はひとことさえばわかることだし、はじめに下手下手というといっそう身構えられてしまって損である。

c. 技術的内容についても、いらないことはいわない方がよい。自分の話の中に、いわないでもそうひどいマイナスにならないことはないか、とよく探すべきである。そういうことはさぼった方が結局得である。こまかいニュアンスを出そうという努力は、われわれが思うほど重要視されない。下手にやれば裏目に出る。われわれは外国人なのだから、表現よりは内容で勝負すべきである。

3.2. 話のめりはり

われわれは英語で話すとき、とかく話を英単語の発声の時系列としてとらえ勝ちである。それは主として話し手としての心の余裕のなさからくることだが、ノッペラボーな話はわかりにくいに決っている。そこで

a. 強立て、節立てをはじめにびしっというとよい。そうしておけば、あとで多少わかりにくいところが出てきても、聴き手の方で内容についての推測ができるので、状況はずっとましになる。

b. また、章、節の切れ目で休んで（または声色を変えて）、切れ目をはっきりさせるとよい。うまいニュースアナは、のべつまくなしにまくし立てて、しかもちゃんとわかるように話すものだが、そういう話しかたは母国語でないと無理である。少しでもつかえ、いいちがいなどがあると、それが話の区切りのようにきこえて、聴き手の理解がはなはだしく妨げられる。われわれの場合、つかえなどは必ずある。そういうノイズの影響を減らすための配慮が必要である。

3.3. 視聴覚機器の利用

話しことばが不十分なわれわれにとって、視聴覚機器は大きな助けとなる。彼らよりもっと注意が必要である。具体的には、

a. ICSEは大きな会議である。ICSE5でも、700人近く入る部屋が三つと、その倍の部屋が一つ使われていた。たいていのスライドの文字が、うしろからはよく読めなかった。大きな字を書くよう、またなるべく絵を使うよう、配慮が必要である。

b. そのためには文字数を減らす必要がある。スライド上では英語でも、主語の省略がおこなわれる（WILL COME BACK LATER と書いて WE をさぼるなど、日記風の文体が使われる）くらいである。言語上の操作よりもむしろ危険かも知れないが、なるべく余計な情報は省き、ことばよりは図を重視すべきである。

c. 絵に魅力をもたせると効果的である。1枚か2枚、プロまたはセミプロに描いてもらったマンガを含めると有効であろう。不謹慎なのでは、などと心配する必要はない。実際、ICSE5でも、パネルですばらしいマンガを見せた人があったほか、2部でちょっといったようにマンガのほかバックグラウンドミュージックまで動員した一般講演があった。やはりそういうのは確実にウケる。もちろん費用は掛かる。だがそういう費用は有効な費用である。特に企業からの発表者の場合、派遣企業のイメージアップになる度合は大変なものである。ぜひ関係者の一考をわずらわしたい。

3.4. 練習

会議に出掛ける前には日常業務にかたをつけておかなければならないので、講演者はとかく練習不足になる。外国語で話すのに練習不足では話にならない。練習をしすぎるとよくないことは周知の通りだが、それが問題になるほどたっぷり練習ができるなら大したものである。もっと練習した方がよい。長期にわたって少しずつやるとよかろう。たとえば次のような手はどうか？

a. 同じパターンの練習を繰り返すと話に勢いが失われるので、ペースを変えるとよい。20分の話の内容を5分で要約してみようとする、などが有効であろう。

b. 一度話の内容を録音して同じ内容を母国語民にしゃべってもらい、それをきいてまねするようにするとよいかも知れない。お金さえ出せば、そういうことをやってくれる人はたくさんいそうである。

c. いっそ会議の席上、母国語民をたのんで代読してもらうのはどうか？ 主催者側と話をつけておかなくてはならないが、理解を得ることはそんなにむずかしくないような気がする。

3.5. ことばそのものについて

そのほか、次のような問題点もある。

a. 日本人はうっかり英文訳風の表現を使ってきょとがえられることが多い。しゃれた表現は避けた方が無難である。

b. 同じことならイギリス風の英語は避け、アメリカ風で統一した方がよい。いずれにせよ、その二つがちがう、ということぐらいは知っておいた方が

よいと思う。

C. 日本人がLとRを混同することは有るが、母音のひびきにも注意する。同じア、アーに何種類もあることを認識する必要がある。また一般に、子音の発声が弱くならないよう、注意する。そんなことをいったって、なかなか一朝一夕には行くものではない、と思い勝ちだが、たとえば3.4節のようなことを一度やると、すっとよくなるものである。