解　説

ソフトウェアのコストについて†

—見積りと実績—

上　條　史　彦††

ソフトウェアの開発コストに関係する要因には実にいろいろなものが指摘されている．現実の開発プロジェクトについての分析例も多い．たとえばWalstonとFelixは「客先との関係」,「要求定義時における客先の協力度」などに始まる29項目を，生産性に関係する因子だとして，選びだしている[1]．

一方 Aron[2] は，NATO の科学委員会の会議の席上で，システム開発計画の数量的側面にふれ，そのシステムに必要となるソフトウェアについて，規模の予測から始まる生産管理手法を提唱した．

プログラマの生産性と，システムの規模の見積りは，ソフトウェアコストを推定する上での2大要素であることは自明である．そして両者共に，技術的な議論はさておいて，現実の場では経験と勘に頼る姿が続いている．本稿は筆者が関係した26件の開発プロジェクトについて，立案時の予測と，プロジェクト終了後のデータを対比し，上記，生産性指標と規模指標のほかに，開発契約の形態，管理者の工程管理体勢が大きな影響を持つことを指摘しようとするものである．

1. 分析の対象としたプロジェクトのプロフィール

とりあげた26件のプロジェクトは，特定の利用者があらかじめ想定されている形のものではない．すなわち現今のソフトウェア開発の大勢を占める「自家用」とは異なり，利用者としては関係のない第三者が想定されている．いずれもここ数年の間に開発され，なかには数十件におよぶ利用者の販売済みのものも含まれている．製品としてのプログラムは Brooks のいうプログラミング製品（一般的，検査済，保守可能，文書化済）から，多種 OS，特殊周辺装置，ユーザの既存システムなどのインタフェースを含むプログラミングシステム製品を目指したものである[3]．

表-1 として個別プロジェクトごとに開発したシス

表-1　プロジェクト

プロジェクト		A	B	C	D	E	F	G	H	I	J
アプリケーション	事務処理	○	○		○	○		○	○		
	技術計算			○			○				○
システムプログラム	制御系										
	言語系									○	
	支援系										
プログラミング言語	COBOL	○	○		○	○		○		○	
	FORTRAN				○		○				○
	PL/I			○							
	アセンブリ言語								○	○	
	その他							○			○
オンライン機能	TSS		○								
	その他					○		○			
データベース	DBMS利用										
	データベース作成										
	その他										
図形処理	グラフィックス										○
	プロッタ			○	○		○				
ミニコンピュータ	専用系	○							○		
	分散処理系										
	インテリジェント端末					○		○			○

† Software Cost Estimates by Fumihiko KAMIJO (Information-technology Promotion Agency, Japan).

†† 情報処理振興事業協会

テムの特性をかかげる．

26件中アプリケーションプログラムが18件，システムプログラム系が7件，両者の機能をあわせもつと考えられるものが1件である．アプリケーションとしては技術計算，とくに有限要素法関連のものが目立つ．システムプログラムとしては制御プログラム（ミニコン用を含む），言語プロセッサ，プログラミング支援系をまとめてあるが，数も少なく，はっきりとした特徴は見い出せない．

最近話題となっているオンライン化（TSSを含む）とデータベース化についてみると，9件と半数近くがなんらかの意味でデータ通信機能を含んでいるが，DBMS の影響は未だしである．自家用システムでは1つの流行となっているが，プログラム製品にまでその波が及ぶか否かは今後の動向として興味ある点だ．

図形処理の9件，ミニコンの11件は時流にそっていることは説明するまでもない．表-1 に示されたうち図形処理ではプロッタ中心のゆき方が依然として主流をしめており，自家用システムとは異なるようだ．ミニコンでは専用形からインテリジェントターミナルや分散処理系（ここでは1つの処理系が，中央システムとサテライトシステムの間で分担処理されているものを指す）への傾向が件数増の形であらわれている．

2. プロジェクトの規模

ソフトウェアコストの議論をするにあたって，どのような規模のサンプルをとるかは重要な問題である．

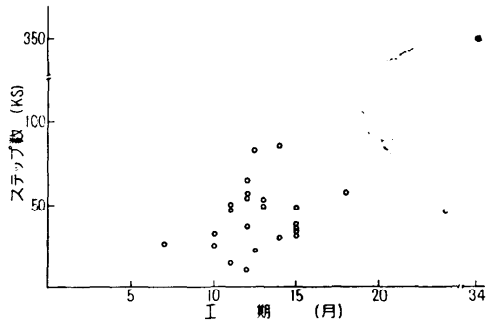

図-1　プログラムの規模と工期

ソフトウェアコストモデルの例としてよく引用されるPutnam のモデル[4),5)]は米陸軍の諸システム（30人年から1,000人年，またはそれ以上）を基礎としており，大型システム開発プロジェクトを代表するものということができよう．

今回は10人年±5人年ぐらいの中小規模のプロジェクトが中心である．日常よくある程度のシステムである．もっとも，概要を与えられて，所要人月数あるいはソース行数を予測するには，中小規模といえどもかなりの困難をともなう．カーネルだけを第1期工事とし，第2期工事以降で完成に近づける自家用システムの場合と異なる．第三者向けのパッケージ化されたプログラムの場合には，例外処理，機能別モジュール化，異 OS への配慮，サポートすべき周辺装置の種類，文書化など，明確な境界線がひけない部分が多く，単純な経験的見積りでは実情にそぐわないからで

プロフィール

K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
○				○	○		○	○	○			○	○	○	○
	○					○				○	○		○		
	○	○	○												
○		○	○	○			○	○	○	○		○			○
	○				○	○	○			○	○		○	○	
		○	○												
						○									
				○	○								○		○
○							○								
						○		○							
							○	○	○					○	
					○		○						○		
							○								

表-2　使用したプログラミング言語による分類

言　語	
COBOL	6
FORTRAN	6
PL/I	1
アセンブリ言語	5
COBOL と FORTRAN	1
COBOL とほかの高水準言語	3
COBOL とアセンブリ言語	3
FORTRAN と COBOL	1

ある．

図-1 から明らかなように，対象としたサンプルシステムはソース行数で 10K—350K，開発期間は7カ月—34カ月のものである．ソース行数の平均は56.6K となる．傾向としては工期が長ければ規模も大きくなっているが，工程の管理指標となるほどはっきりとしたものではない．これはプログラム開発の特徴でもあり，他面，現在の開発専門業者がリソースとしての技術者の稼動率を上げるため，受注残を常時かかえたり，並行開発を行わせたりすることにも起因している．

プログラミング言語の種類もプロジェクトの性格に影響を与える．高水準言語によるものが21件をしめ，しかも制御系のプログラムが少ないので，詳細な分析に適したサンプルとはいい難い．言語による差については後日，別のサンプルを選んで調べることとしたい．

表-2 に記述言語別の件数を示しておく．

3. 見積りと実績

ソフトウェアについては，個々の開発・運用段階におけるコストでなく，ライフサイクル全体を見渡しての最適化を図るべき，とするライフサイクル論が盛んである．一応もっともであるが，後述の Putnam らの数量的議論は米軍向けの長寿命システムを前提としての議論であることを忘れてはならない．管制制御系そのほかの巨大システムは，それ自身20年にも及ぶ長期の運用がなされており，民需，特にわが国における企業向けシステムの短命，激動の様相とは事情を異にする．わが国で類似の分野を求めるとすればプロセス制御系であって，ここにあげたような汎用計算機中心のアプリケーションには長命型のものはまことに少ない．

ソフトウェアのライフサイクルについては Putnam[4),5)]（前出）や Wolverton[6)] に詳細な記述がある．

図-2 はライフサイクルの概念図を示したものである．一般にはプログラムの設計・制作・検査・補修・改訂のサイクルをとるに要するマンパワーをモデル化したもので，システムの定義（要求定義）や機能設計の段階は外して考えている．Rayleigh の曲線，すなわち

$$\frac{dY}{dt}=\frac{K}{t_d{}^2}te^{-\frac{t^2}{2t_d{}^2}}$$

がよい近似を与えることが知られている．ここで

Y'：所要のマンパワー（工数/時間）

t_d：開発開始から，ピーク時までの時間

である．

今回議論の対象としているのは，ライフサイクル全体ではない．設計・制作・検査の段階だけが問題としている分野である．図-2 の斜線で示した区間がそれに対応する．それ以外の段階については，完成後，時日を経ていないこともあって，後日に譲りたい．

3.1　規模の見積り・実績対比

図-3 に規模（ソースコード行数）についての見積り・実績対比を示す．図から明らかなように，規模の見積りは 20～30% の誤差を容認すれば，実用上さし

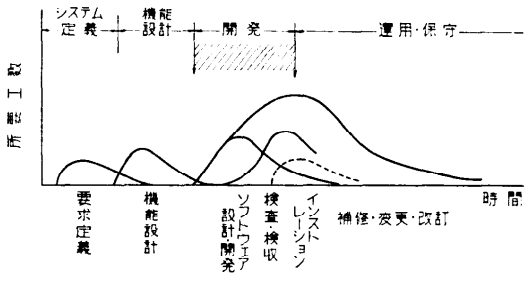

図-2　ソフトウェアのライフサイクル

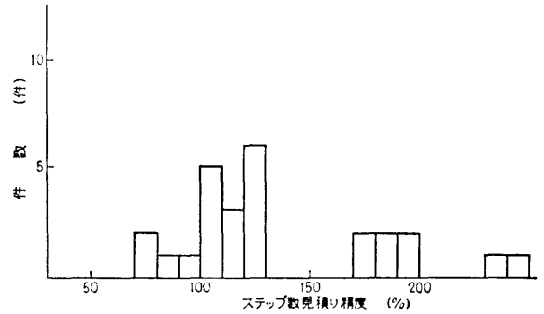

図-3　ソフトウェア開発の見積り精度

つかえのない範囲で的中している．今回とり上げたサンプルのうち 70% ぐらいがそのわくに納っている．問題は残りの 30% にあり，次の2点に集約される．

（1） 当初の見積りの2倍以上に膨張してしまったプロジェクトが 30% 近くあること．

（2） 平均値が当初見積りの 35% 増というオーバランを示していること．

26件のプロジェクトはすべて「確定上限値つき出来高払い方式」の契約のもとに遂行されたものである．「確定上限値つき契約」とは契約額を超えた費用を受託者側が無条件で負担することを条件とする委託契約の一種である．大幅に超過する確率が 30% もあり，かつ平均で 35% ものオーバランがあるという事実は，形式的には非常にリスクの高い事業といわざるをえない．

異常値を示した8件を個別に調べれば，それなりの事情があろうが，ここでは全体像として考えてみたい．

（1） 記述言語の影響

使用された言語はつぎのとおりである．(主言語で分類，副言語との組合せを含めている)

COBOL	5件
アセンブリ言語	2件
FORTRAN	1件

比較的やさしいといわれる COBOL が多いことから，言語の難しさに原因を求めるわけにはいかない．

（2） プログラムの規模

使用言語別にみた規模の平均を次に記す．

COBOL	46Kステップ
FORTRAN	48Kステップ

アセンブリ言語については 350K 近い大規模プロジェクトが入っているため，比較対象には加えられない．全体の総平均は 57K であり，大規模化が見積り誤差の原因と断じるには，平均規模が小さすぎる．規模の問題もまた，問題の解答ではない．

3.2 プロジェクト固有の要因

対象となるプロジェクトにはさまざまな特徴がある．アプリケーションの型別に整理してみよう．

問題となった8件のなかにはオンラインシステム（1件），多種 OS との対応を行ったもの（1件），プロッタを含むもの（1件），専用ミニコン（2件）など，単純な事務処理プログラムとかけ離れた特性を有するプログラムが5件ほど区別できる．開発経験の不足は見積り違いをおこす要因ではある．しかし設計仕様などの成果物の内容から推察するに，未経験とは断じがたい．経験者ならば，むしろ当初の積算時点で，難易度などを算入ずみであるべきだ．

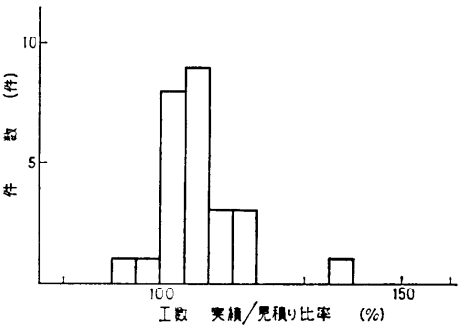

図-4 開発工数 見積り精度

3.3 契約形態からくる影響

工数の見積り・実績対比（図-4）を考察の材料に加えてみよう．

工数の実績値は，ステップ数のそれとは異なり，見積りの 110% 以内に集中している．すなわち，規模見積りの誤差は，生産性の向上によって補完されたことになっている．

契約方式の影響をここに見ることができる．「確定上限値つきの出来高払い契約」の下では，プロジェクトリーダに課せられるのは，目論見通りのプログラムを作ることにより，定められた工数内で，機能仕様を充足する製品を産みだすという責任なのである．

総ステップ数を生産性（ステップ/単位工数）で除して，所要工数を求めるのが，普通一般の方法である．しかしプロジェクト管理の場では，その逆の計算は行われていない．ベテランを投入したり，補助員を増したり，マシンタイムの割当上便宜を図ったり，いろいろな手段がある．単純に“良い”製品が目論見通りの形ででき上ることは至上目的ではなくなっているのである．

3.4 その他の話題

a） 納 入 期 限

プログラム開発の契約では，期限はあっても，やむを得なければ延長できること，が不文律である．ほとんどの場合に成果物の代替は得られないから，発注者にとっても，完成を待つしかしかたがない．欧米では事態を予期してペナルティクローズが付くことが多い．

今回とりあげた例はやや異なった条件で契約されたものである．期限に関する違約については，契約の破棄をもって臨むことになっている．予算制度上やむを得ないのである．異常にふくれ上ったいくつかのプロ

ジェクトについては，ここに原因を求めることができよう．

b） 規模の考え方

発注者と受注者では規模についての見方はなかなか一致しない．発注者は機能仕様（もしくはプログラム仕様）から「製品」の規模を推算する．受注者側では工数の節約，品質の確保，それにも増して自社の開発慣行に合致させること，などを目的として種々のプログラミングツールを作成・利用しようとする．コメントの行数についても，製品の姿としてみて必要なものと，開発チーム内のコミュニケーション用のものとでは，当然異なってくる．

c） 手戻り作業

人間関係の投影ともいえるプログラム製品の場合，開発チーム内のコミュニケーションの不足，利用環境(特に OS) の変動，などの影響により，手戻り作業が生ずる確率が高い．

仕様確定の時期，設計技法，組織の管理，プログラムツールの利用など種々な対応策はあるものの，なおプロジェクトリーダのスキルに依存する部分が大きい．

ただし，今回の実験値のデータとしては，手戻りは単純加算としてあつかった．積算にあたって当然考えなければならない因子だからである．

4. マンパワーの投入率（工数消化率）

マンパワーをどの時点でどれだけ投入するかは，工期を守り，製品の品質を確保する上で大変重要なことである．ライフサイクル論にたてば，投入計画はまた，見積りの基礎である．

26件のプロジェクトについては，およそ3カ月ごとに，工数，マシンタイム等の指標のデータを収集した．実績のデータを後追いで分析する形なので，ここでは工数消化率と呼ぶことにする．

工数消化率はそのプロジェクトが全体として必要とした工数の何パーセントを，当該工期に消費したかを示す数字である．プロジェクトごとに工期が異なるので，t_s すなわちプロジェクトの開始時点と，t_d すなわち完成品の引渡し時点の間の期間に対してノーマライズしてある．

工数消化率は図-2 の斜線部分を抜き出したものである．したがってソフトウェア(プログラム)の設計，開発，検査，検収に対応する開発者側の所要工数のすべてを含む．もともと t_d はライフサイクル曲線の頂点に対応する時点として，Putnam らによって定義されているが，今回使ったデータが3カ月ごとの集計値であることから，上述の解釈をとったことをお断りしておく．

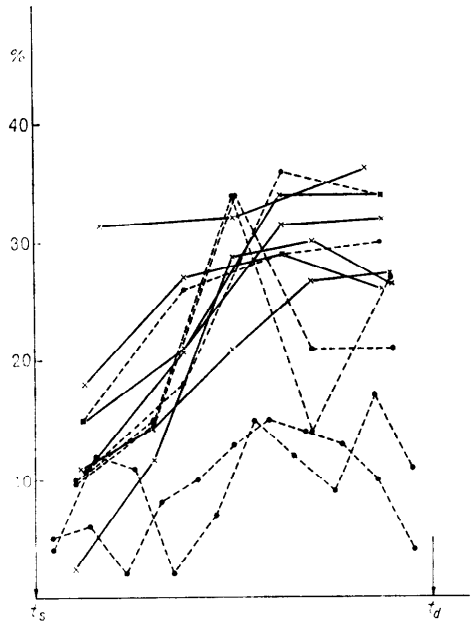

図-5　工数消化率（その1）

工数消化率の経時曲線の特徴から，全体を3個のグループにわけることができる．**図-7** はその概要を示したものである．

図-5 はプロジェクトのうち，Putnam らの主張に比較的従っているとみられるものの一群である．26件中12件が記録されている．（グループ1）

図中，下位2件はほかとやや異なる動きをしているので説明の要がある．両者とも，かなり長期にわたったプロジェクトであって，管理の都合で委託者が中途で契約期間を終了させ，途中経過を検査した後，作業の継続を行った．波を打っているのは契約が中断した点に相当していると考えてよい．弧状の動きを示しているものは，事情は同じであるが，多種 OS に相当製品を作製するために，まず一個の OS で完成させ，その後，プログラムの変換を行っている．分類上は第2のグループに入れてもよかったかも知れぬ．

図-6 はグループ1に属さないほかの14件をまとめてある．ちょっと見ただけでは判りにくいが，2つのグループがある．グループ2は工期の半ば以後にピークのあるもの ($t_{peak} \geqq (1/2)t_d$)，グループ3は前半に

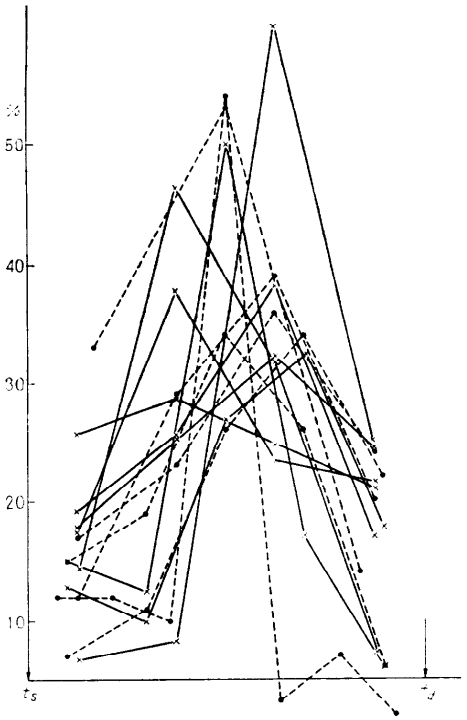

図-6 工数消化率（その2）

ピークのあるもの（$t_{peak}<(1/2)t_d$）である．

グループ2については，図-2のインストレーションが含まれていないことから説明がつく．既製品としてのプログラムでは，受注品のように完成，即ユーザへの引渡しという手順にならない．ファーストインストレーションは営業活動の後になる．しかも最初のユーザを探すことが，プログラム販売の最も困難な作業といわれている．事実上インストレーションがあったのと同じ程度の検査（出荷検査）を行えば図-2の姿に近づくわけであるが，それ程の余裕がなかった証拠だ．

グループ3についてはライフサイクル論では説明できない．$(1/3)t_d$ 付近が3件，$(1/2)t_d$ よりやや手前に4件と別れているが，集計周期を考慮すると同じグループに入れてもよかろう．おそらく技術者のやりくりの都合であろうという推定はできるが，それでは，なぜ納期を早めなかったかという点に疑問が残る．開発者の経営方針とかかわりあう現象であることは確かだ．

5. まとめ

ソフトウェア開発はまことに人間臭い仕事である．

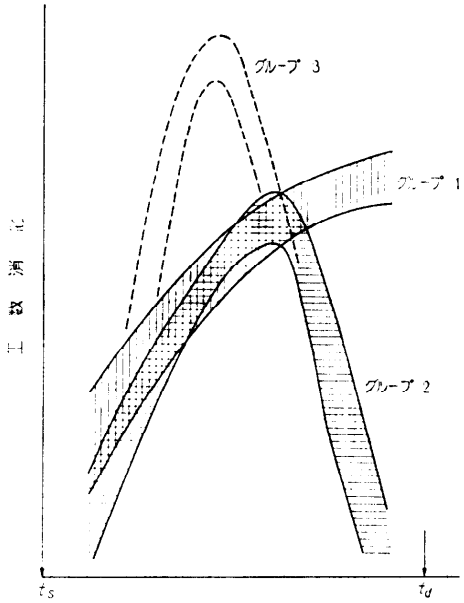

図-7 工数消化率（その3）

高級なソフトウェアは水準の高いスキルから産れるといわれてきた．しかしながらコンピュータ化の拡大が急ピッチに過ぎたため，ソフトウェアへの需要の伸びが著しく，少数の練達の士の供給力では追いつけない事態となってから久しい．多くの開発プロジェクトは初心者や未経験者を投入したり，ニーズの内容を知らない外部技術者の力に頼らない限り，所期の期日をまっとうできない．

生産技術や設計技術のないところで，工場生産をおしすすめるような状況である．コスト積算は諸般のむじゅんを背負いこんだ形で，経験論的な判断を少しでも計数処理に近づけようと努力している分野の好例といえよう．

前出の実例をまとめた結果，ソフトウェア開発の分野では，規模の推定，生産性の推定のほかに，プロジェクトの外から与えられる重要な因子がいくつかあることが明らかになった．

契約の方式はプロジェクトリーダの考え方に与える影響が大きく，なかでも重要な因子である．上限金額の有無，ペナルティの種類，インセンティブの内容，保守責任，そのほか多くの項目が例示できる．

プロジェクトチームのメンバの稼動率も，日程計画に反映される条件の1つだ．

経営方針，具体的には受注残の水準の定め方，並行割当ての有無も，生産性に大きな影響を与える．プロ

ジェクトチームが自社のコンピュータを使うか，他社のものを借りるか，といったことも分析に価する．

プログラミング言語の選択，開発しようとするシステムの特性など，純技術的な因子は以外に影響力が小さい．おそらくは優秀なリーダ諸氏が，このような問題については十分に配慮ずみといったところか．

参 考 文 献

1) Walston, C. E. and Felix, C. P.: A Method of Programming Measurement and Estimation, IBM Systems Journal, Vol. 16, No. 1, pp. 54-73 (1977).
2) Aron, J. D.: Estimating Resources for Large Programming Systems, Report on a Conference Sponsored by the NATO Science Committee, Rome, Italy, October 1969, Software Engineering Techniques, Mason Charter.
3) Brooks, Jr., F. P.: The Mythical Man-month: Essays on Software Engineering, Addison-Wesley Publishing Company, Inc. (1975).
4) Putnam, L. H.: The Software Life Cycle: Evidence and Foundation, COMPSAC 77 Tutorial Quantitative Management: Software Cost Estimating, p. 326, IEEE (1977).
5) Putnam, L. H. and Fitzsimmons, A.: Estimating Software Costs, Datamation, Sep. pp. 189-198, Oct., pp. 171-178 and Nov., pp. 137-140 (1979).
6) Wolverton, R. W.: The Cost of Developing Large Scale Software, IEEE Transactions on Computers (June 1974).

(昭和55年6月25日受付)