ソフトウェア工学 37－2
（1984. 7. 25）

# ファンクション・ポイントによる ソフトウエアの機能的生産性評価

建石　雄三（日本アイ・ビー・エム（株））

## 1. はじめに

ソフトウェア開発の生産向上が叫ばれ、また それを目指す諸活動が展開されてきて久しく、その途上でもある。「ソフトウェア開発の生産性が何パーセント向上」という話をよく耳にもし、且つ、自分自身でも何らかの数字を算出してしまう。しかし、その論拠は確立した方法に基づいているとはいえず、どうしても感覚の域を脱していない。

つまり、ソフトウェアの生産性を評価しさらに見積の方法まで拡張していこうとすると、様々な疑問に遭遇する。それらは、多くの本質的問題を含んでいると思われ、その解消には長時間かかるだろう。このため、開発環境に即した現実的な前提の下に簡便な方法を用いることにより、正確性の点では若干の難点はあるが、大筋を把握できる程度の評価を実施しているのが実情である。

ソフトウェアの生産性評価の手法は、ソフトウェアが大規模化し、開発技法の発展とともに、多くの方法が試みられてきた。生産性とは、尺度によって生産物を測定し、それを多くの場合、投入された工数との比較で表そうとするものである。尺度に対するアプローチとして二つに大別することができよう。それらを仮に

・　トップダウン
・　ボトムアップ

とする。

トップダウンは、過去のプロジェクトの経験を利用する方法で、システムを機能的にはっきりした単位まで分割し、そこに尺度をあてはめ数値化していこうとするものである。経験値に含まれていない、特殊な、あるいは困難な問題は、数値化されにくいという危険性がある。

ボトムアップは、全システムの生産物としての構成要素に分割し、1つ1つの要素が内容的に明確になるまで追求した上で数値化するものである。欠点としてシステムの規模が見通せないと使いにくく、また分割した要素の内容の難易度までには、立ち入っていない。

開発生産物を尺度によって測定しようとすることと全く異なったアプローチで生産性を見ようとする動きもある。それは、開発工程に着目することであり、

・　開発期間は最適か、
・　投入人数は最適か、

等の問題である。これらも生産性、とりわけDP部門全体の生産性を左右する事象であり、生産物による生産性測定と同様に重要な問題であろうと考える。

## 2. 従来の生産性評価－LOC

日本アイ・ビー・エムの情報開発は、社内事務処理の機械化部門であり、システムズエンジニア、プログラマ合わせて約140人で構成され、適宜ソフトウェアの外注も含めて年間約250個のプロジェクトが推進されている。内容的には、200

人月以上費やし、開発期間が数年にわたるプロジェクトから、1人月にも満たない全保守タイプのプロジェクトまで千差万別である。生産性評価方法としては、過去久しく、プログラム行数による評価(LINE OF CODE,以下LOCと略す)が用いられてきた。LOCは一番表面的でわかりやすい尺度であり、実際のコスト見積りの場合においては、現実に用いられ、特にソフトウェアの外注の際、外注先と共通の土俵で話しあえる尺度として用いられて来た。

LOCの考え方は、ボトムアップに属し、ソフトウェアの量的な生産性評価の一つである。過去2年間に開発が終了したプロジェクトの中から、26個のプロジェクトを選定し、それをもとにLOCの分析及び問題点を指摘したい。なお、26個のプロジェクトのデータについては、付録として添付する。

プロジェクト・データの中で、プログラム行数が把握できたものは、8プロジェクトと少ないが、それをもとに全工数の分析を行うと、

LOC : プログラム行数( K 行)

W A : 全工数

とすると、

W A = 121.0LOC + 5.7 ( 単相関係数 ; 0.817)

が得られる。

これから、1000行のコードを生成するために約 127人時間およそ1人月必要ということになる。平均1プログラム当りの行数が350行程度であるから、プログラム1本当り作成するのに48人時間、プログラミング／テスト段階では約20人時間必要であるということになる。現在設定されている標準(26人時間)よりかなりよい結果となっている。これは、サンプル中に含まれるプロジェクトの中には、プログラム・コードの再利用したものが多いためで、これをもって、全体像と見ることは危険である。

LOCをソフトウェア生産物の測定尺度として用いる場合、上述のような問題が発生する。それらを整理すると次の用になる。

a．処理方式による LOCの相違

ソフトウェアにおいて同一機能を実現するために、単一でないコーディングが存在し、それによってLOCが結果的に異なる。コーディング・ガイド等の標準でこれを補うには限界がある。

b．プログラム言語の相違

PL/I,アセンブラ，APL等のプログラム言語の相違をどのように扱うか。

c．プログラム行数の範囲

プログラム行数の算出の際、コメント行、マクロ展開を含めるか否かである。サンプル・データでは、コメント行は含み、マクロ展開は含んでいない。

d．ツール，テストデータ等の処置

特に大規模なソフトウェアの開発においては、開発を支援するためのツール類や、テスト・データを作成するためのプログラムを作成する。これらを生産物として認知するかという問題である。

e．コードの再利用

既存のソフトウェアを流用して、開発する場合が増加しつつある。ひとまとまりの機能単位の置換または追加の場合には、LOCを計算することはたやすいが、部分的な機能の削除と追加が混在する場合、もしくは部分的な削除/追加

のみの場合はLOCの算出が煩雑になる。

3. ソフトウェアの機能的評価

従来のLOCによる生産性の評価方法はいくつかの問題を含んでいるためLOCから脱却した評価方法の摸索が行なわれている。一つのプロジェクトに対する評価という観点ばかりではなく、ユーザーとの係わり、DP部門の生産性をも含めた考え方が必要である。

ユーザーと合意した事項を守るということは、一口にいって次の三つを厳守することに他ならない。すなわち

- ユーザーが要求した機能を提供したかどうか。
- 合意したコスト内か。
- 納期

もし、これらが守られたならば、そのプロジェクトは合意事項を守ったこととなり、この厳守の度合が合意事項の測定になる。この尺度で高得点を得るには、事前の見積りが合理的でなければならない。つまり過去に終了したプロジェクトの経験に基づいている。

ユーザーとDP部門で合意した事項が適切か否かという問題が次に現れてくる。合意内容が挑戦的ではなかった、換言すれば、内容があいまいで甘さがあることがおこる可能性が十分にある。そうしたDP部門ではソフトウェア開発の仕事がどれほど出来るかということが理解されていないといえる。結果として「競争力」「生産性向上」に対して疑問が出てくる。生産性を系統的に測定すると、平均よりも生産性が高いプロジェクトと低いプロジェクトの二つに分類できる。次に、生産性が平均以上のプロジェクトの要因を調べ、その要因を他のプロジェクトに広め、同様に平均以下のプロジェクトの要因を取り除くことが可能である。

生産性評価を効果的に行うためには、プロジェクトから出力されたソフトウェア製品に何らかの測定というものを行う必要があり、また、要因つまり生産性に影響のあるプロジェクト特性も把握しなければならない。

ソフトウェア製品の測定とプロジェクト特性とを完全に分離した尺度を持ち、ユーザーにとって理解しやすい言葉で構成され、更に見積りの検証に使うために開発過程の早い段階で利用可能な評価方法が、1978年に米IBM のAlbrechtが提案したファンクション・ポイントである。

ファンクション・ポイントは、アプリケーションの開発生産性が、そのアプリケーションが扱うデータタイプの数に基づいていると仮定している。そのデータタイプとは、

a. インプット

スクリーンやテープによって入ってくるトランザクション、あるいは他のシステムから自動的に送られてくるトランザクション

b. アウトプット

紙によるレポート、スクリーン、あるいは他のアプリケーションに対するトランザクション

c. インクウァイアリー

インプットやスクリーンが対になった特殊ケースで、アプリケーションに対してデータの加工は行わない。

d. 内部ファイル

e. インターフェイス

他のアプリケーションと共用しているファイル

の五つである。これらのデータタイプを洗出し、各々に対して単純、平均、複雑の三つに分類し、それぞれに対して加重を掛け、調整前のファンクションポイントを算出する。次に、この結果に対し一般的な内部処理の複雑性を示す14個のプロジェクト特性によって調整を行う。それらは、

- 高可用性
- 信頼性
- 対話型のオペレーション
- 高いレスポンスの要求

等である。これらによって調整値を得て、調整前のファンクション・ポイントに乗じることによって、最終的なファンクション・ポイントが算出される。

*FUNCTION COUNT

| I.D | ITEM \\ COMPLEXITY \\ | SIMPLE | AVERAGE | COMPLEX | TOTAL |
|---|---|---|---|---|---|
| IO | User External Inputs | ..... X 3 = ..... | ..... X 4 = ..... | ..... X 6 = ..... | |
| OO | User External Outputs | ..... X 4 = ..... | ..... X 5 = ..... | ..... X 7 = ..... | |
| MO | User Logical Master Files | ..... X 7 = ..... | ..... X 10 = ..... | ..... X 15 = ..... | |
| FI | Interfaces to Other Systems | ..... X 5 = ..... | ..... X 7 = ..... | ..... X 10 = ..... | |
| QO | User External Inquiries | ..... X 3 = ..... | ..... X 4 = ..... | ..... X 6 = ..... | |
| AA | TOTAL UN-ADJUSTED FUNCTION POINTS = | | | | |

*PROCESSING COMPLEXITY FACTORS

| I.D. | FACTOR DESCRIPTION | VALUE | I.D. | FACTOR DESCRIPTION | VALUE |
|---|---|---|---|---|---|
| B.1 | Communication Facility is Used | | B.8 | On-Line Update | |
| B.2 | Distributed Processing or Data | | B.9 | Complex Processing Logic | |
| B.3 | Application Performance Consideration | | B10 | Designed for Code Re-Use | |
| B.4 | Resource Conservation Consideration | | B11 | Conversion/Installation Ease | |
| B.5 | High Transaction Rate | | B12 | Ease of Operation | |
| B.6 | On-Line Data Entry | | B13 | Multi-Site Use (Common System) | |
| B.7 | Conversational Data Entry | | B14 | Ease of Change/Use by User | |
| | | | BB. | TOTAL VALUE ADJUSTMENT = : | |

) Not Present, or No Influence = 0 ) Moderate Influence = 2 ) Significant Influence = 4
) Incidental Influence = 1 ) Average Influence = 3 ) Strong Influence Throughout = 5

CC : VALUE OF ADJUSTMENTS = 0.65 + ( 0.01 X BB ) = :_____:

DD : FUNCTION POINT INDEX ( AA X CC ) = :_____:

図1 ファンクション・ポイントの計算

ファンクション・ポイントは、トップダウンのアプローチの一つで、その場合の機能分割単位はファンクションそのものである。

サンプル・データからファンクション・ポイントと工数との関係をみると、まず、全工数とファンクション・ポイントとの関係では、

WA = 全工数

FP = ファンクション・ポイント

とすると

$WA=23.32(FP)^{0.784}$ (相関係数 0.740)

が得られる。また 概要設計段階から、導入段階までの工数 (W 24)との関係では

$W24=19.76(FP)^{0.783}$ (相関係数 0.772)

を得ることができる。さらに、詳細設計段階から、導入段階までの工数 (W 34) との関係では、

$W34=44.48(FP)^{0.577}$ (相関係数 0.540)

が得られる。

ここで、この３ケースのうち、概要設計段階から導入段階までの工数とファンクション・ポイントとの関係が一番相関係数が高いことに着目したい。これは、ファンクション・ポイントの定義としてのファンクションを規定することは要求定義段階の主要な作業であって、要求定義段階を含む全工数との関係ではその分だけ振れることを物語っている。最近のアプリケーションは、サービス形態が多様化し、新しいハードウェア、システム・プログラムの選定により慎重になり、機能の決定までスムーズに至っていないことが多いことを現わしている。又、相関係数が低い詳細設計段階以降の工数との関係から、ファンクション・ポイントとプログラムの具現化がさほど関係をもたないといえる。

概要設計段階から導入段階までの工数とファンクション・ポイントの関係をオンライン・システム／バッチ・システムの別に見ると、前者の相関係数が0.746に対して後者が0.847と高い。現在のファンクション・ポイントの考え方、あるいは、各機能の重みづけがよりバッチ・システムに適合していると解することができる。

次に、ファンクション・ポイントを開発の生産性の評価に用いている例をあげてみる。右図の例は、ある米IBM の開発部門で、1978年までさかのぼり、ファンクション・ポイントを算出し、1983年までに終了したプロジェクトの工数とともに集計した結果である。この例では、年率22％で生産性を向上させており、また現在、平均1人月あたり17ファンクション・ポイントの割合でアプリケーションを生産している。この傾向は、この部門の改善の結果を表わすものとして使用できるが、注意しなければならないことは 、全体的なプロジェクトの計画には利用できるが、個

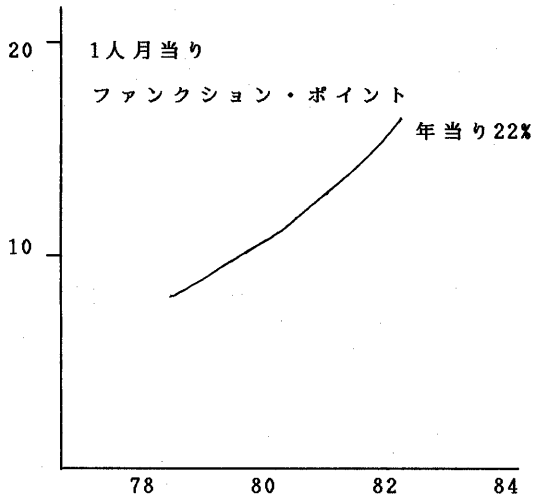


表1　部門の生産性

個のプロジェクトに際しては使用できるものではないということである。なぜなら、プロジェクト間には差があり、それをプロジェクト特性として加味しなければならないからである。この部門がこれだけの生産性向上を果した主因は、開発管理の手法の改訂とされている。

ファンクション・ポイントによる生産性評価は、LOCの考え方に内在している問題点からの脱却を意図して考えられたが、ファンクション・ポイント自体にも問題がないわけではない。それらは、次である。

- ファンクション・ポイントでのファンクションは、要求定義段階での生産物であり、そのためにこの段階の内容をファンクション・ポイントで説明するには無理がある。
- ファンクション・ポイントの算出にあたり、単純、平均、複雑の区別等、どうしても主観がはいる余地がある。

・　プロジェクト特性の検証がむずかしい。
・　ファンクション・ポイントを表せない分野が残る。

## 4　LOC とファンクション・ポイントの関係

LOC とファンクション・ポイントの関係を先のサンプル・データから見てみよう。

LOC ： プログラム行数(k行)
FP　： ファンクション・ポイント
とすると、

FP = 20.26LOC + 47.04　　(相関係数　0.91)

が得られる。この式からファンクション・ポイント当たりのPL/Iプログラム行数を求めると、約47行となる。LOC とファンクション・ポイントの相関は、一応高いといってよい。

同様の調査が他でもなされている。まず日本ガイド・シェア( 日本IBM のユーザー研究団体）では、８プロジェクトをサンプリングし、相関91％で１ファンクション当たり約129行としている。ALBRECTが最初にファンクション・ポイントを提案した論文によれば、相関は不明であるが、1ファンクション・ポイント当たり65行であったと述べている。これらの結果から直ちに生産性の比較をすることはできない。三者ともサンプル数が少ないことと、 LOC、ファンクション・ポイントとも問題点を内包しているからである。

## 5　機能的評価の拡張

LOCによる生産性評価からの脱却をねらって、ファンクション・ポイントの導入を行ってきたが、ファンクション・ポイントにおいても、前述したような問題点を含んでいる。それらを個別に解決する手段をとることが最良である。しかし、そうした解決策を見出すことは非常にむずかしい。今までいろいろ提起された解決策を見ると、問題を除去するために、多くの要素をとり入れたモデルを構築している例が多い。例えば、WALSTON-FELIX のモデルは入力項目が24個にも及んでいる。こうすると確かに合てはまり具合はよくなるが、実際の問題として実用性は貧しい。

生産性の評価のためには、より単純なモデル（計算式）を設定することが必須である。理由としては、

・　実用性
・　得られた結果の信頼性が数パーセント異っていたところで、プロジェクト推進上の変動要因の方が大きいと考えられるので、その要因を除去することを行なえば、信頼性の差がほとんど問題にならない。

が上げられる。

ソフトウェア製品の構成要素であるLOC と機能要素を表わす尺度であるファンクション・ポイントとによって工数との関係を調べると、次のようになる。ここでLOC とファンクション・ポイントは相関が高いので多重共線性がおこり得る可能性があるが、あえて意味的に興味深いのでとりあげる。

N24 ： 概要設計段階から導入段階までの工数
とすると、

N24 = 2.07FP + 0.097LOC- 541.7　　(重相関係数　0.897)
が得られる。

意味としては、プログラム行数の約一割とファンクション・ポイントの二倍が工数となっている。ファンクション・ポイントとLOC の影響度合はどうであろうか。プログラム行数を一万行、ファンクション・ポイントが500 の場合、工数は約1450人時間で、双方の影響した部分は45%ずつとなる。ファンクション・ポイントが1000となった場合、その影響度合は67%となる。しかし、要求定義段階が一応終了しているならば、ファンクション・ポイント自体は、さほどの変動要因とはならない。逆にLOC の方が詳細設計の内容によって変動する可能性が大きいといえる。つまり、ファンクション・ポイントだけでなく、LOCを加味することによって、開発過程の後半の生産性の測定に織り込むことができる。

LOC の代わりにプログラム本数を用いることによって、LOC での問題点のいくつかが除去できる。

NP : プログラム本数

とすると、

W24 = 0.082FP + 42.9NP - 314.0　　(重相関係数　0.877)
が得られる。この場合、プログラム１本あたり約43人時間に対してファンクション・ポイントが全体的に加算され、プロジェクトのサイズを反映させる要因となっている。

LOC あるいはプログラム本数を事前に予測するのは、ある程度むずかしい面もある。プロジェクト・サイズが大枠として見えてくると、ごく自然に投入人数が決定してくる。ごく自然にいった所で実際は、過去の経験に基づいて決められていることに他ならない。開発期間についても、事前にあたえられていて、しかも非常にきびしい場合を除いて、プロジェクトの規模から経験的に求められる。そうした経験値をLOC もしくはプログラム本数の代替として使うことはできないだろうか。そこで、まず、

N2 : 概要設計段階の投入人数

NA : 詳細設計段階の投入人数

NB : プログラミング／テスト段階の投入人数

とすると

W24 = 2.46FP + 179.0N2 + 945.5NA + 37.1NB - 2180.8(重相関係数　0.901)
となる。投入人数の係数を見た場合、詳細設計段階の投入人数の係数が大きいことに注目したい。概要設計段階では、ファンクション・ポイントで規定している機能を仕様として変換することが主目的なため、振れが大きくないのに対し、詳細設計段階でのプログラム仕様に具現化することは、ファンクション・ポイントでは説明しきれない部分があることを物語っている。これはLOC とファンクション・ポイントを合わせたモデルと同じ結果となっている。

次に開発期間と投入人数、ファンクション・ポイントの関係、

NM : 全開発過程での投入人数

TM : 全開発期間(月数)

とすると、

W24 = 1.83FP + 214.1NM + 251.3TM - 2156.7　　(重相関係数　0.910)
が得られる。つまり、工数は、開発期間と投入人数に依存する度合が大きく、フ

ファンクション・ポイントは、プロジェクト・サイズの大きさを示す調整要素として考えることができる。逆に、プロジェクトの大きさが定ったならば、不用意に開発期間、あるいは投入人数を増大させることを避けねばならない。

## 6　今後の展開

ソフトウェア開発におけるプロジェクトの生産性評価の改善のために、従来のLOC による方法とは、観点のちがったファンクション・ポイントによる機能的生産性評価とその拡張の可能性について述べてきた。生産性評価をめぐる古くて新しい問題がこれによって一気に解決できるとは思わないが、そのための一助になればと思っている。

今後の展開としては、ここで述べたモデルをより改善していくために、収集データをもっと多くしていかなければならない。それとともにプロジェクト特性の見直しも必要であろう。

さらに、生産性評価の眼目の一つである開発支援ツールが生産性にどのような効果を及ぼしたかを明確にしていく方法を考えたい。これについては、今のところデータ収集以前にどのようなデータをとるべきを考えなくてはならない。

## 付録　サンプル・データ

| | 工数（人時間） | | | | | | 投入人数 | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| プロジェクト | 要求定義 | 概要設計 | 詳細設計 | プログラム作成 | 導入 | 保守 | 要求定義 | 概要設計 | 詳細設計 | プログラム作成 | 計 | 開発言語数 | レベル | 画面数 | 帳票数 | プログラム修正箇所数 | COMMON 入力数 | COMMON 出力数 |
| A | 2284 | 1778 | 223 | 283 | 134 | 288 | 5 | 10 | 3 | 1 | 3491 | 0 | 0 | 24 | 0 | 0 | 0 | 1 |
| B | 176 | 37 | 37 | 701 | 47 | 0 | 2 | 2 | 2 | 3 | 154 | 8 | 947 | 7 | 4 | 126 | 1 | 1 |
| C | 371 | 191 | 543 | 481 | 97 | 0 | 4 | 2 | 3 | 6 | 246 | 49 | 15772 | 25 | 8 | 0 | 1 | 0 |
| D | 70 | 11 | 39 | 272 | 0 | 0 | 3 | 2 | 2 | 2 | 109 | 0 | 0 | 5 | 4 | 53 | 0 | 1 |
| E | 72 | 91 | 72 | 242 | 71 | 0 | 2 | 2 | 2 | 3 | 172 | 0 | 0 | 6 | 3 | 70 | 0 | 1 |
| F | 3210 | 4723 | 795 | 3614 | 1835 | 0 | 7 | 12 | 9 | 13 | 1647 | 0 | 0 | 36 | 5 | 478 | 0 | 1 |
| G | 1495 | 2708 | 1704 | 5832 | 761 | 1063 | 6 | 7 | 6 | 9 | 4048 | 0 | 0 | 34 | 14 | 944 | 0 | 1 |
| H | 2537 | 2055 | 1874 | 6184 | 227 | 0 | 6 | 6 | 7 | 18 | 910 | 0 | 0 | 24 | 7 | 644 | 0 | 0 |
| I | 991 | 340 | 71 | 438 | 178 | 0 | 2 | 2 | 2 | 3 | 103 | 10 | 3430 | 10 | 3 | 40 | 0 | 0 |
| J | 783 | 1013 | 1227 | 1437 | 130 | 1700 | 2 | 5 | 3 | 2 | 666 | 120 | 47061 | 20 | 9 | 0 | 1 | 0 |
| K | 1551 | 1418 | 663 | 930 | 330 | 1315 | 4 | 4 | 3 | 3 | 481 | 76 | 34823 | 16 | 4 | 287 | 1 | 0 |
| L | 95 | 230 | 71 | 383 | 13 | 513 | 1 | 2 | 1 | 2 | 216 | 0 | 0 | 11 | 4 | 0 | 1 | 0 |
| M | 83 | 114 | 508 | 717 | 69 | 0 | 3 | 4 | 4 | 5 | 1055 | 84 | 22503 | 6 | 2 | 204 | 1 | 0 |
| N | 119 | 83 | 15 | 528 | 23 | 0 | 3 | 3 | 1 | 6 | 318 | 0 | 0 | 5 | 3 | 0 | 1 | 0 |
| O | 0 | 272 | 111 | 740 | 0 | 0 | 0 | 3 | 3 | 9 | 143 | 0 | 0 | 3 | 2 | 0 | 0 | 0 |
| P | 0 | 204 | 102 | 1605 | 0 | 0 | 0 | 2 | 2 | 4 | 531 | 0 | 0 | 7 | 0 | 0 | 1 | 0 |
| Q | 293 | 301 | 327 | 2112 | 396 | 0 | 4 | 5 | 6 | 8 | 371 | 0 | 0 | 6 | 3 | 0 | 0 | 1 |
| R | 637 | 766 | 436 | 984 | 0 | 0 | 2 | 3 | 3 | 4 | 938 | 0 | 0 | 16 | 6 | 0 | 0 | 0 |
| S | 325 | 873 | 233 | 1727 | 333 | 0 | 1 | 2 | 2 | 7 | 355 | 41 | 24309 | 18 | 6 | 427 | 1 | 1 |
| T | 1191 | 3647 | 6834 | 491 | 0 | 3950 | 3 | 7 | 7 | 5 | 1643 | 183 | 62345 | 23 | 1 | 0 | 1 | 1 |
| U | 149 | 287 | 133 | 837 | 84 | 0 | 7 | 5 | 6 | 7 | 46 | 0 | 0 | 3 | 2 | 194 | 0 | 0 |
| V | 503 | 740 | 744 | 1448 | 84 | 0 | 3 | 3 | 5 | 14 | 282 | 0 | 0 | 12 | 4 | 112 | 0 | 1 |
| W | 36 | 24 | 18 | 324 | 110 | 0 | 2 | 1 | 1 | 4 | 29 | 0 | 0 | 4 | 2 | 36 | 1 | 1 |
| X | 86 | 284 | 87 | 194 | 14 | 0 | 2 | 2 | 2 | 2 | 78 | 0 | 0 | 4 | 2 | 0 | 1 | 1 |
| Y | 18 | 226 | 143 | 203 | 38 | 0 | 1 | 1 | 1 | 3 | 298 | 0 | 0 | 6 | 2 | 0 | 1 | 1 |
| Z | 6245 | 7853 | 4572 | 6626 | 2316 | 4750 | 7 | 8 | 9 | 11 | 2215 | 0 | 0 | 42 | 12 | 924 | 0 | 1 |

## 参考文献


1　花田収悦「ソフトウェア生産性の評価と管理」情報処理,Vol.21,No.10 80年,PP1057-1064

2　「システム開発の生産性測定－Ａ」83年度日本ガイド/シェア報告書,84年,PP207-223

3　ALBRECHT, J. A, "MEASURING APPLICATION DEVELOPMENT PRODUCTIVITY" PROCEEDIMG APPLICATION DEVELOPMENT SYMPOSIUM 79年 PP83-92