ソフトウェア工学 100−12
（1994．9．8）

メトリクスによる
組み込み型ソフトウエア開発の評価

大西 荘一　　呉 旻炳　　冨岡 学　　原田 等
岡山理科大学理学部 応用数学科　　同志社大学工学部 知識工学科

組み込み型ソフトウエアを開発している一組織に対する、メトリクスによる開発評価を報告する。ソフトウエア・サイズと開発工数の関係から、改造サイズを考慮した工数推定モデルを提案する。このモデルから改造生産性を算出する。工程別工数比率を求めテスト工程比率が意外に大きいことを示す。設計担当者に対するアンケート調査とメトリクスの結果を比較検討し、直感と経験による設計者の意識が現実とかなり差があることを示す。
メトリクスにより組み込み型ソフトウエア開発を種々評価しデータを提供する。

An Evaluation of Development of Embedded-Mode Software by using Software Engineering Techniques

Soichi ONISHI
Dept. Appl. Math., Okayama University of Science
Min Byung OH,　　Manabu TOMIOKA,　　Hitoshi HARADA
Dept. KE and CS, Doshisha University

This paper reports the result of an analysis by using software engineering techniques of one organization developing embedded-mode software. A linear cost estimation model is presented in consideration of the program size of modified part, which is derived from evaluating actual data of development effort to program size. Software productivities of modified part are computed using the cost model presented. By viewing the relative effort ratio of each software development phase, that of test is found to take an unexpectedly large part in all phase. Comparing estimated results with data by questionnaire to programmers, it is shown that conciousness of programmers with inspirations and experiences differs considerably from actual development activities.

【１】はじめに

１９７１年に電卓用として開発されたマイクロプロセッサー[1]はその利便性と柔軟性から情報機器、医療機器、計測器、家電、娯楽機器、等あらゆる機器に急速に応用されて行ったことは周知である。これらの機器のソフトウエアは普通ROMに格納され機器の中に組み込まれる。従って、実行時に補助記憶装置からRAMにロードされる汎用コンピュータやパソコン用ソフトウエア（以後、ロード型ソフトウエアと記す）とは種々の面で異なる世界を持っている。

組み込み型ソフトウエアはROMに格納されるためそのプログラムサイズはROMの容量に依存してきた。初期においてはプログラムサイズは数KByte程度であったが、半導体技術の進歩によりROMの１チップ当たりの容量は飛躍的に増大し、又コストも著しく低下したため、それに伴いプログラムサイズも今や１００KBを越える物も珍しくはない。ここに至って組み込み型ソフトウエアにおいてもソフトウエア工学の手法が必要となってきた。

ソフトウエア工学で生産性や品質を問題にするとき経験による抽象的評価ではなく数値による具体的評価が必要である。メトリクスによる生産性評価は大規模ロード型ソフトウエアに対して１９７０年代より盛んに研究され[2][3]コストモデルとしてCOCOMOモデル[4]等が提案されている。組み込み型ソフトウエアに対してはHP社の報告[5]や、藤村の横断的な調査研究報告[6]等があるが、数は少ない。

本研究はほぼ３年にわたり同一の組織で開発された組み込み型ソフトウエアの４２のプロジェクトについて計測分析し、数値による評価を行い、従来担当エンジニアが経験的に意識下にあった内容と比較検討した。現状の認識と生産性向上のための指針を得ることを目的とした。

【２】計測条件

同一の組織において、１９８８年後半から１９９１年前半にかけて開発された４２のプロジェクトで計測した。これらのソフトウエアはキーボード、ディスプレイ、プリンターを有するマイクロプロセッサー応用の流通業界向け機器に組み込まれている。開発環境は各プロジェクトとも大差はなく次のとうりである。

①プログラム言語：アセンブラ
②開発ツール　　：専用デバッガー
③応用分野　　：流通業界向け機器

計測は下記の開発工程別に行った。計測のために特別なツールは使用していない。工数は各工程の担当者が統一された基準の元に自分自身の作業時間を計測し記録し、開発完了時に報告された。

〈開発工程〉
①仕様作成工程
②設計・コーディング工程
③デバッグ工程
④テスト工程（第三者による詳細なテスト）

【３】計測データ

下記データを計測した。
①プロジェクト名
②工数（工程別）：人月
③全サイズ　　：KByte
④新規作成サイズ：KByte　（新規サイズ）
⑤改造作成サイズ：KByte　（改造サイズ）
⑥工程別担当者名：名前
⑦工程別カレンダ：年月日
⑧BUG数　　：個（テスト工程で発見されたBUG）
⑨新作サイズ　：KByte（新規＋改造）
⑩新規率　　：％（新作に対する新規の割合）
（注）
全サイズ＝新規サイズ＋改造サイズ＋流用サイズ
改造サイズ：改造したルーチン全体のサイズ

上記のサイズはオブジェクトコード量である。生産性のベースとなる量としてはソースプログラムの行数が一般的[4]である。しかし、組み込み型

ソフトウエアにおいては前もって決められているROMサイズにプログラムが入りきるかどうかが重要な問題である。開発担当者は絶えずオブジェクトサイズを気にしながら開発を進めている。従って、生産性のベースデータとしてオブジェクト・コードのサイズを採用した。

【４】 データ整理・分析

データは表計算ソフトを用いて整理しグラフを作成した。この方法は文献（５）でも推奨している。４２プロジェクトの内、明らかに計測あるいは記載ミスをしていると思われる５プロジェクトのデータを削除した。３７プロジェクトのデータを分析した。

【４－１】工数と全サイズの関係

図１は新規サイズが２KB～３KB、かつ改造サイズが３KB～５KBの範囲にある、すなわち新作サイズがほぼ同程度のプロジェクトの総工数と全サイズの関係である。これより総工数は全サイズから影響を受けていないと言える。この傾向は各工程別工数も同様である。

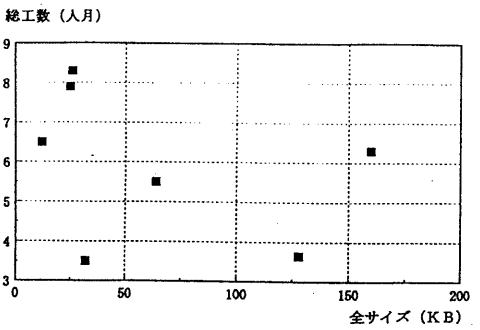

図１．全サイズと総工数の関係

評価１：総工数、及び工程別工数は全サイズの影響を受けない。

評価１より

（１）本ソフトウエアは**良く構造化ができている**ものと考えられる。ソフトウエアの構造が不明確な場合、全サイズが大きいほど設計は混乱し工数が増大する。

（２）全サイズは工数推定モデルの要因から除外する。

【４－２】工数と新規サイズ、改造サイズの関係

図２は改造サイズが０KBのプロジェクトの総工数と新規サイズの関係である。図３は新規率が２０％以下、すなわち新規サイズの影響の少ないプロジェクトの総工数と改造サイズの関係である。両者より総工数は新規サイズ、及び改造サイズに比例している。仕様作成工程以外の工程別工数も同様である。仕様作成工程の工数は比例関係にないが、この工程は後工程に影響を与えているはずである。その結果として総工数が比例関係にあるので仕様作成工程も他工程同様に新規サイズ、改造サイズの影響を受けていると考えられる。

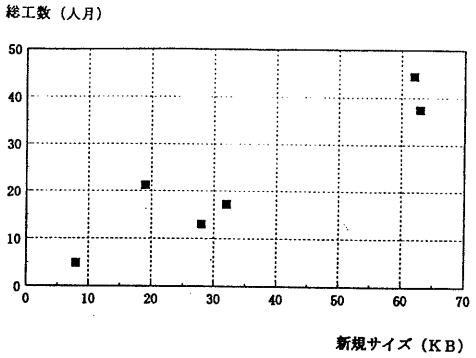

図２．新規サイズと総工数の関係

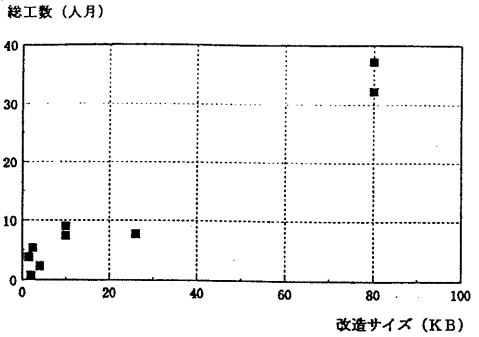

図３．改造サイズと総工数の関係

評価２：総工数は新規サイズ、及び改造サイズに依存する。

【４－３】各工程の工数比率

（１）全プロジェクトの工程別工数比率

図４は全プロジェクトの算術平均の工程別工数比率である。テスト工程が４７．３%で全体のほぼ半分を占めており、生産性に大きく影響している。HP社の報告[5]では２７．７%であり、これに比べてもかなり大きい。

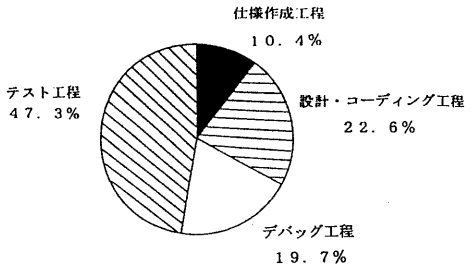

図４．全プロジェクトの工程別工数比率

評価３：テスト工程の平均工程比率は約５０%あり、生産性に大きく影響している。

テストの効率化が生産性向上の大きなポイントであることを示している。

全プロジェクトの平均工程比率の概略値

仕様作成工程	**：１０%**
設計・コーディング工程	**：２３%**
デバッグ工程	**：２０%**
テスト工程	**：４７%**

（２）工程比率と全サイズの関係

図５、図６は全サイズの大小別工程比率である。図より全サイズ２０KB未満と１００KBより大の工程比率に大きな差は見られない。

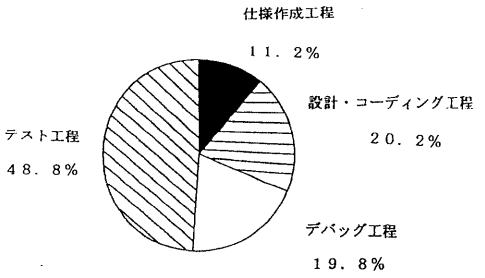

図５．全サイズ＜２０ＫＢの工程比率

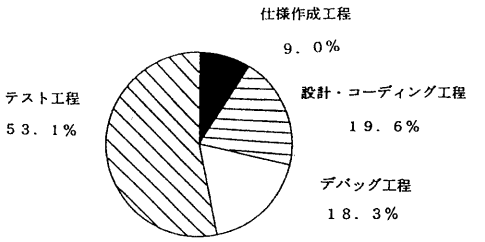

図６．全サイズ＞１００ＫＢの工程比率

評価４：工程別工数比率は全サイズの大小によりさほど影響を受けない。

（３）工程比率と新規率の関係

図７、図８は新規率の大小別工程比率である。設計・コーディング工程とテスト工程の比率に大きな違いが見られる。新規率３０%未満のテスト工程比率は新規率７０%より大のテスト工程比率より１３%大きい。この違いは改造サイズの差によるものと思われる。改造サイズには設計・コーディングされていない部分、すなわち流用部分を含んでいる。そのため改造サイズの大きい新規率小のプロジェクトでは設計・コーディングの工数が少なくなり、テストは流用部分も含めて行われるためテスト工数はそれほど少なくならないと考えられる。この傾向は新規率３０%未満と７０%より大のプロジェクトで顕著である。

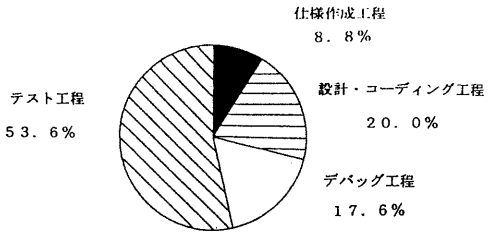

図７．新規率＜３０%の工程比率

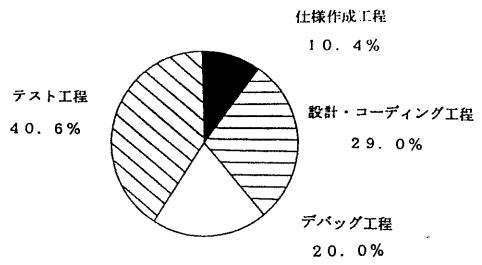

図８．新規率＞７０%の工程比率

評価５：テスト工程比率は新規率が小さいと大きい。新規率３０%未満と７０%より大では１０%以上の差がある。

改造部分のみに限定したテストが出来れば、生産性の向上が期待出来ることを意味する。

（４）工程比率とBUG発見個数の関係

図９、図１０はテスト工程で発見されたBUG数別工程比率である。図より仕様作成工程比率と設計・コーディング工程比率の合計は

BUG数　５個未満では　３５．１%

BUG数３０個より大では３１．９%

差は少ないが仕様作成工程比率は

１４．３%と４．３%で大きな差がある。これは仕様作成をしっかりやることが後工程に良い影響を及ぼしソフトウエアの品質を向上させる、と言う経験的に理解していたことを数値で確認出来たことになる。

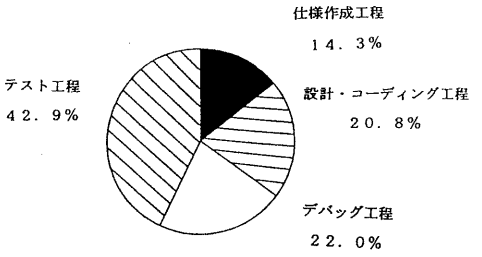

図９．BUG数＜５個の工程比率

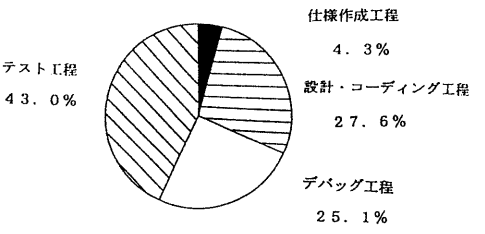

図１０．BUG数＞３０個の工程比率

評価６：仕様作成工程の充実はソフトウエアの品質に大きく影響する。

【５】工数推定モデル

評価１、評価２より工数は全サイズに依存せず、新規サイズと改造サイズに依存する。このことより、工数推定モデルとして次の式を提案する。

評価７：工数推定モデル

$$E = a \cdot (N + kM)$$

E　：総工数（人月）

N　：新規作成サイズ（KByte）

M　：改造サイズ（KByte）

a，k：定数（開発形態により決まる種々の因子を含む）

kは改造サイズを新規作成サイズに換算する係数を意味する。

1／a＝（N＋kM）／E　（KB／人月）

となり1／aは生産性を意味する。

モデル式において

N＝0　とすると

1／（ak）＝M／E　（KB／人月）

となり

1／（ak）は改造の生産性を意味する。

以上まとめると

k　　：新規サイズ換算係数

1／a　　：生産性　（KB／人月）

1／（ak）：改造生産性　（KB／人月）

全プロジェクトのデータより最小2乗法でa，kを算出すると、

a＝0．644

k＝0．411

となった。モデル式は

E＝0．644（N＋0．411M）

となる。

評価8：全プロジェクトの平均生産性

生産性　＝1．6　（KB／人月）

改造生産性＝3．8　（KB／人月）

現実のソフトウエア開発では、新規作成のみと言うのは少なく改造と流用をあわせもつ場合が多い。改造サイズを考慮した本モデルは実際的であると考える。又、生産性と共に改造生産性も算出でき新規作成と改造との比較検討が可能である。

a，kはソフトウエア開発形態で決まり、非常に多くの因子を含んでいる。従って組織体によって、あるいは同一組織体であってもソフトウエアのカテゴリーや開発環境が異なれば数値は異なる。

【6】工数推定モデルの検証

図11は（N＋M）と総工数、

図12は（N＋0．411M）と総工数の関係である。両者より明らかに図12の方が直線性が良い。k値の有効性が分かる。

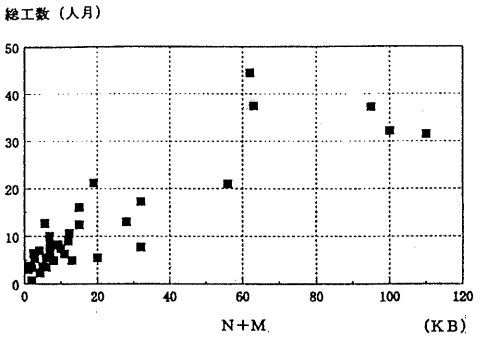

図11．N＋Mと総工数の関係

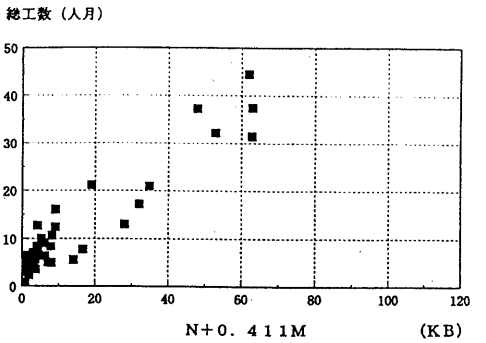

図12．N＋0．411Mと総工数の関係

【7】工数推定モデルの利用

プロジェクト・リーダ別生産性

プロジェクト・リーダ別にa，kを算出し生産性、及び改造生産性を求めた。図13はそのグラフである。

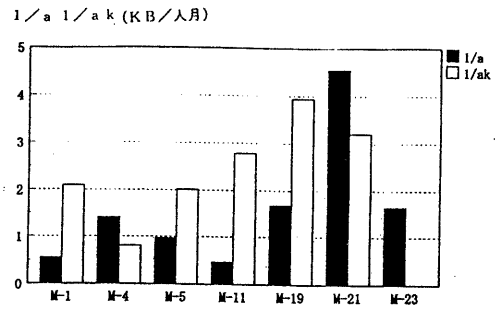

図13．プロジェクトリーダー別生産性・改造生産性

評価９：プロジェクト・リーダ別生産性

最大値　４．　５（ＫＢ／人月）

最小値　０．　５（ＫＢ／人月）

最大値は最小値の９倍である。

評価10:プロジェクト・リーダ別改造生産性

最大値　３．　９（ＫＢ／人月）

最小値　０．　８（ＫＢ／人月）

最大値は最小値の約４．９倍である。

【８】開発担当者の意識と現実

開発設計担当者（実際に計測した担当者）１３名にメトリクスの結果を開示する前にアンケート調査を行った。アンケートの結果を**《担当者の意識》**、メトリクスの結果を**《開発の現実》**として両者を比較検討した。

（１）生産性の意識と現実

図１４は生産性に対するアンケートの回答である。図中ａ，ｃのラインはそれぞれプロジェクト・リーダ別に計算した生産性の最大値、最小値である。ｂのラインは全プロジェクトの平均値である。　現実の平均値は１．６（ＫＢ／人月）、意識の平均値は３．２（ＫＢ／人月）であり、意識の生産性は現実の２倍であった。又、１～２（KB/人月）と回答したのは、則ち意識と現実がほぼ一致したのは３／１３名(２３％）であった。

９／１３名（６９％）は現実より大きい生産性を回答した。

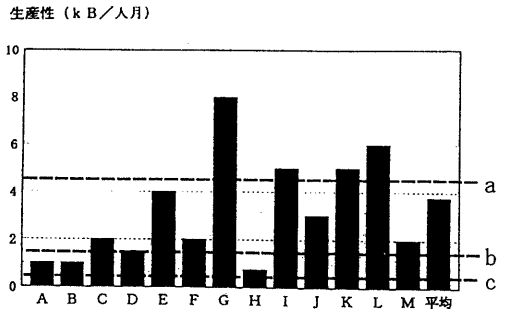

図１４．生産性に対するアンケート結果

評価11：開発担当者の生産性に対する意識は現実より大きい。

（２）工程別工数比率の意識と現実

図１５は工程別工数比率の意識の平均値である。下記に意識と現実を比較する。（単位：％）

	現実	意識	現実と一致
仕様作成	１０	１６	４６
設計	２３	３６	８
デバッグ	２０	２０	８５
テスト	４７	２８	２３

現実と一致のデータは意識が現実とほぼ一致していた人数のアンケート者１３名に対する率である。設計（コーディング含む）とテストに大きな差がある。テストは第３者テストであり設計者はテスト者を指導する程度で直接テスト・オペレーションはしない。その結果、設計者のテスト工程に対する意識が低くなったようである。又、その逆に設計者の主業務である設計・コーディングの比率が高くなっている。

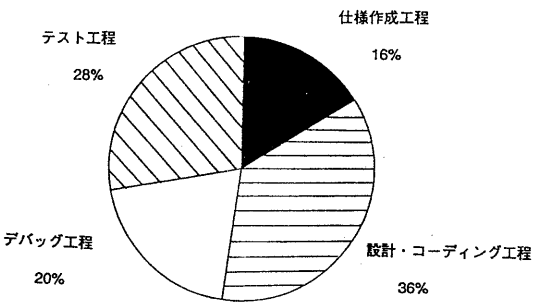

図15.工程別工数比率の意識

評価12：開発設計者はテスト工程の工数を過少評価している。

このことが設計者が生産性を高めに意識している原因の一つと思われる。

（３）その他の意識と現実

全サイズと生産性の関係

現実：全サイズは生産性に影響しない。

意識：全サイズが大きいほど生産性は悪くなる

新作サイズと生産性の関係

新作サイズが大きくなると生産性は

現実：低下しない。

意識：低下する。

評価13：設計者はソフトウエア・サイズと生産性の関係を正確に理解していない。

以上より設計者の直感と経験による意識はあまりあてにならないと言うことになる。

【９】おわりに

組み込み型ソフトウエアを開発している一組織に対し、メトリクスにより集中的にその開発を評価することが出来た。評価内容は開発設計担当者が経験的に普段から理解している内容と異なることが多く、意外であった。改めてメトリクスの必要性を認識した。

改造サイズの考慮を特徴とする工数推定モデルを提案した。このモデルから平均的に改造は新規作成の約２倍の生産性があることが分かった。しかし一方、少数のプロジェクトで逆に改造が新規作成より生産性が悪い場合もあり、ソフトウエア開発の難しさを表した。

工程別工数比率ではテスト工程が約５０％を占め生産性に大きく影響していることが判明し、テストの効率化が生産性向上の大きなポイントであることを示せた。BUG数と工程別工数比率の関係より従来から経験的に理解していた、仕様作成の充実が品質に良い結果を与える、ことを数値で説明出来た。

意識と現実の比較により、設計担当者の開発プロセスに対する理解が曖昧であることを示した。特にテスト工数の差が大きいことを示し、設計者の認識を新たにすることが出来た。

本研究の結果は一組織に対してのものであり、これが全てとは当然考えていない。ソフトウエア開発はきわめて多様であり、他の多くの組織で同様な研究がなされ、それらの結果と比較検討したいと考えている。

今後、プログラム構造、開発期間、担当者人数、品質、等さらに詳細に評価する。より精度の高い工数推定モデルに改善をしていく。計測を継続しデータの変化を追跡する。

参考文献

（１）嶋正利　著：“マイクロコンピュータの誕生”　岩波書店（１９８７）

（２）上篠史彦　監訳：“ソフトウエアのコスト積算”　産業図書（１９８１）
原著“Software Cost Estimation Study” Doty Associates, Inc.（１９７７）

（３）B.W.Boehm：“Software Engineering Economics”　IEEE Trans. on Soft. Eng. SE-10, No.1, pp4-21（Jan. １９８４）

（４）B.W.Boehm：“Software Engineering Economics”, Prentice-Hall （１９８１）

（５）清水、水野　訳：“ソフトウエア・メトリクス”、日経BP社　（１９９０）

（６）藤村直美：“組み込み型マイクロプロセッサ用ソフトウエアにおけるコスト・モデル”情処学会研究報告　９４－SE－９８（１９９４）