変換項目を考慮したソフトウェア移植工数モデル

金井 敦[†] 古山恒夫[††] 高橋宗雄[†††]

ソフトウェアの移植作業は，新規開発とは異なる作業を行う．したがって，移植コストの予測に当っては，新規開発のコストモデルではなく移植のためのコストモデルが必要となる．移植コストモデルは従来からいくつか提案されているが，従来のモデルは新規開発と同様に移植プログラムの規模のみを最も重要な影響要因としているため，精度に限界があった．本論文では，プログラム規模だけでなく，プログラムの変換項目数にも着目した新しい移植コストモデルを提案する．このモデルは，移植コストを移植プログラム規模に比例する部分とプログラム手作業変換項目数に比例する部分の2つから構成される．プログラム手作業変換項目はコンバータの機能やプログラム内に含まれる変換項目から導かれる．モデルの妥当性は，実際の移植プロジェクトのデータを用いて検証した．移植プロジェクトのデータを元にモデル式のパラメータを推定し，移植コストを見積もった．その結果，規模のみで移植コストを推定する場合には，平均相対誤差が41%であったのに対し，本モデルでの平均相対誤差は24%に減少した．

Software Conversion Cost Model Based on Conversion Items

Atsushi Kanai,[†] Tsuneo Furuyama[††] and Muneo Takahashi[†††]

A model for estimating the cost to convert a program from one eivnronment to another, without changing the language in which it is written, has been developed. It was developed by analyzing actual data from a FORTRAN program conversion project. A new feature in this model is that manual conversion items are treated as one parameter. The model is divided into two distinct parts: the first depends on the program size and the other depends on the number of manual conversion items, which are derived from converter functions and program conversion items. This model can more precisely estimate the cost to convert a program than conventional models that only consider program size. Since the parameters of this model were inferred and estimated using actual conversion project data, the average relative error of estimated values is 24%, whereas the error of values estimated using only program size is 41%.

1. はじめに

あるソフトウェアを開発するときに，新規開発をするかあるいはすでにあるソフトウェアを移植するかは重要な判断ポイントである．この判断を行う場合に，新規開発コストと移植コストを比較することが必要である．新規開発コストについては古くから多くのコストモデルが提案されており[1)~3)]，コスト予測の最も重要なファクタである規模の予測についても，ファンクションポイント法[4)]やその改善手法など多くの方法が提案されている．一方，移植コストについては，プログラムの規模のみに着目した aL^b (a, b：コンスタント，L：プログラム規模) 型のモデル[5)]とソースプログラムコンバータの効果を考慮したモデル[6)]が提案されている．しかし，後者のモデルは定性的なモデルであり，前者の aL^b 型のモデルはその正確さに議論の余地がある．このように，移植コストモデルについては有効なコストモデルがほとんど提案されていないのが現状である．

同一言語を異なる環境に移す場合のような移植では，ソースプログラムコンバータを使用すると非常に簡単に変換ができる場合があるため，中大規模な移植ではソースプログラムコンバータ（以後，コンバータと呼ぶ）が使われるのが一般的である．このため，プログラムの規模に関係なく完全自動でコンバージョンを行えるような極端な場合は，大規模なプログラムでも簡単に移植が可能である．このように，移植の場合

† 日本電信電話株式会社 NTT ソフトウェア研究所第一プロジェクト
Project Team-1, NTT Software Laboratories, Nippon Telegraph and Telephone Corporation
†† 日本電信電話株式会社 NTT ソフトウェア研究所ソフトウェア技術研究部
Software Engineering Laboratory, NTT Software Laboratories, Nippon Telegraph and Telephone Corporation
††† 桐蔭学園横浜大学工学部制御システム工学科
Department of Control & System Engineering, Faculty of Engineering, Toin University of Yokohama

は規模との相関がそれほど大きくないと考えられる．また，コンバータの効果は移植対象のプログラムの内容により大きく異なるため，プログラムへのコンバータの効果を考慮する必要がある．筆者らは，先にこの考えに基づいた移植コストモデルを提案した[7)~9),11)]．しかし，このモデルは非常に複雑で実用上問題があった．

本論文では，移植コストモデルを移植プログラム規模に依存する部分とコンバータの効果が大きく影響する部分の大きく2つに分割して考え，簡単に移植コストを計算できるモデルを提案する．第2章では，モデルの基本的な考え方と基本モデルを示し，第3章で，実際のプロジェクトデータを使用して第2章で示した基本モデルを検証する．第4章では基本モデルをベースに詳細なモデルを構築する．第5章ではプロジェクトデータを元にパラメタを推定し，モデルの評価を行う．

2. 基本モデル

コンバータを使用した移植は**図1**に示すような手順で行われるのが一般的である．この手順から分かるように，新規開発コストに比較して，移植コストは非常に異なる傾向を持つと想定される．コンバータを使用することにより手作業によるソースプログラムの変換作業が不要である場合には，移植作業工数は移植対象となるプログラムの規模とほとんど関係しない．もし，同一プログラムであってもコンバータを使用した環境では移植工数は少なく，コンバータの無い環境で移植された場合には手作業による変換作業量が増加し，両者の場合で移植工数が大幅に異なる場合が想定される．これは，たとえ同一なプログラムであってもコンバータを使用する場合としない場合，あるいはコンバータの変換能力が高い場合と低い場合などの条件により移植コストが非常に大きく影響を受けることを示している．また，同一プログラムを異なるターゲット環境に移植するためにコンバータを用いた場合では，ターゲット環境により変換する部分が異なるため，たとえ同一のプログラムであっても，ターゲット環境により移植工数は大きく異なる．

以上の考察から，移植コストに影響を与える支配的な変数はプログラム規模（L）とプログラムが持つ変換項目の種類と数（*prog*）とコンバータの機能（*conv*）になると想定できる．従って，概念的に移植コスト（E）は以下のように表現できる．

$$E=e(L, prog, conv) \qquad (1)$$

ここで，e は任意な写像である．

図1に示す作業のうちで，予備調査，ファイル変換，言語変換，ドキュメント作数，確認（コンパイル／リンク，TP 走行／確認）は，明らかに *prog* には依存しないが依植されるプログラムの規模に比例して作業量が増加すると思われる．それに対して手作業変換およびデバグの工数はプログラムが持つ変換項目の種類と数（*prog*）とコンバータの機能（*conv*）に依存すると思われる．そこで，プログラム規模に比例す

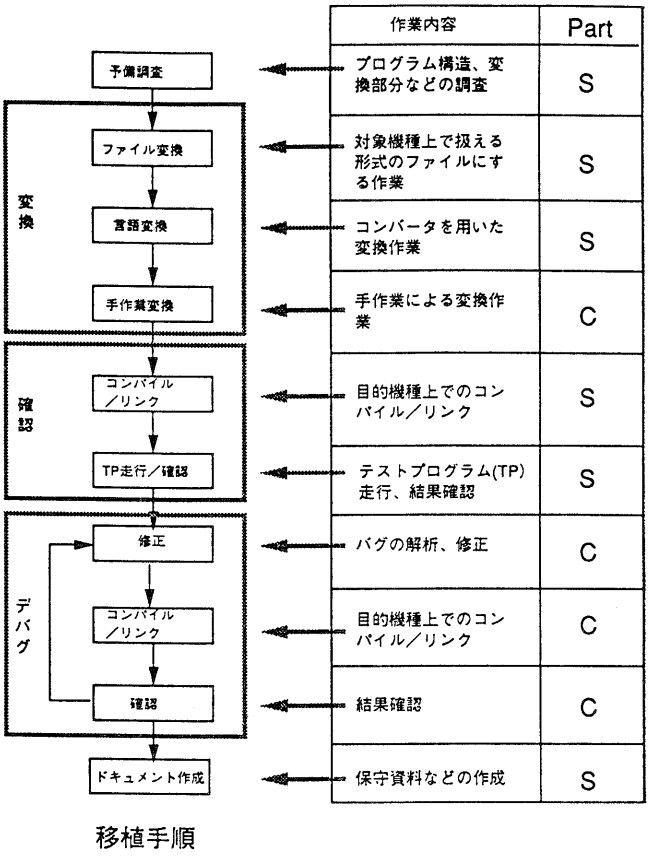


図 1 移植手順とパート
Fig. 1 General conversion process.

ると思われる作業に対応するコストを E_s, *prog* と *conv* に依存すると思われる作業に対応するコストを E_c とすると，式(1)はさらに以下のように表される．

$$E=E_s(L)+E_c(prog, conv) \quad (2)$$

それぞれの作業が E_s (以後，S-part と呼ぶ) と E_c (以後，C-part と呼ぶ) のどちらに分類されるかを図1の表に示す．

3. 基本モデルの検証

3.1 測定対象プロジェクトの概要

実際の移植データを収集したプロジェクトの概要を以下に示す．このプロジェクトは，大型計算機上の FORTRAN で記述された LSI-DA (LSI-Design Automation) プログラムを他の大型計算機に移植するプロジェクトである．実際に工数が測定されたプログラムは6本であり，全体の規模は 131 KLOC である．さらに，変換項目がすべて測定されたプログラム（工数が測定されていないものもある）は9本であり，全体で 226 KLOC である（前述の6本を含む）．このプロジェクトで行われた移植作業は，図1に示した手順と同一の手順で行われた．本プロジェクトでは，高機能なコンバータ FSCONV[10] を使用してソースプログラム変換を行った．工数データは図1に示した作業項目ごとに収集した．

3.2 妥当性評価

E_s に寄与する作業部分 (S-part) と E_c に寄与する作業部分 (C-part) の全工数に占める比率を示したのが**表1**である．この表に示すように，C-part, S-part ともに小さい比率ではなく，それぞれの比率はプログラムにより大きく異なることがわかる．さらに，それぞれの比率は移植プログラム規模に依存していない．このことは，C-part, S-part ともに無視するわけにはいかず，コストモデルとしてどちらも考慮する必要があることを示している．

各プログラムごとの変換項目の出現分布を示したのが**図2**である．図から分かるように同じ LSI-DA 関連のプログラムであっても個々のプログラムごとに変換項目の出現数の傾向が大きく異なる．これより，プログラムの移植では，移植プログラム個々の変換項目分布を考慮しなければ，正確な移植工数の予測はなし得ないことが分かる．

4. モデルの詳細化

4.1 S-part

S-part と C-part の移植プログラム規模との分散分析の結果を**表2**と**表3**に示す．この表に示されるように，S-part はプログラム規模と有意な関係にあるが，C-part はプログラム規模とは有意な関係にないことが分かる．このことから，S-part は次式に示すようにプログラム規模の一次式で表現できる．

$$E_s(L)=C_{s1}L+C_{s2} \quad (3)$$

ここで，C_{s1} と C_{s2} は定数である．

4.2 C-part

4.2.1 定 式 化

本節では，C-part をモデル化することを試みる．同一言語間の移植の場合は，プログラム構造まで変化す

表1 C-part と S-part の比率
Table 1 Ratio of C-part to S-part.

プログラム	C-part (%)	S-part (%)
A	25.8	74.2
B	7.6	92.4
C	56.6	43.4
D	30.0	70.0
E	13.1	86.9
F	37.8	62.2

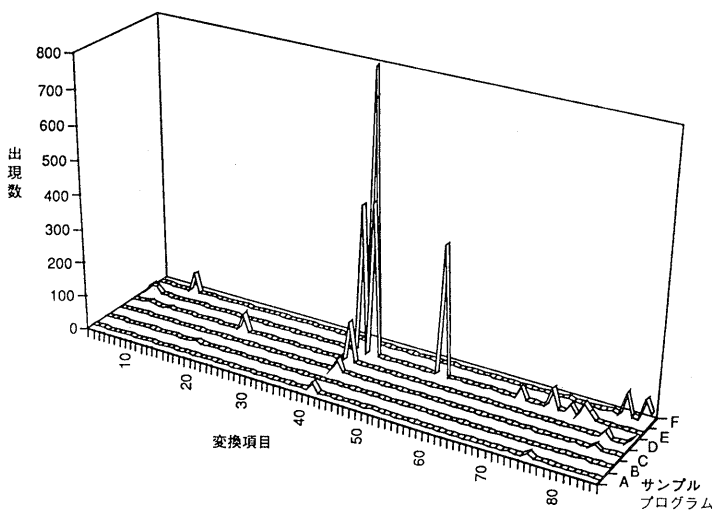


図2 変換項目の出現傾向
Fig. 2 Appearance of conversion items.

ることはまれである．実際，**表4**に一例を示すように測定したプロジェクトで発生した変換項目は構造変化まで伴うようなものではなかった．C-part は基本的にこの変換項目を手作業で変換することにより発生する工数である．なお，以下の分析では，各変換項目を手作業で変換する工数は変換項目ごとに独立であると仮定する．

実際のコンバータが変換できる変換項目はコンバータごとにそれぞれ異なる．原理的に存在し得る1から N (N は自然数) の変換項目を要素とする変換項目集合を S_N とする．ただし，$S_N=\{j|1\leq j\leq N\}$．コンバータの機能 f を変換項目集合 S_N から S_N の部分集合である自動変換できない変換項目集合 S_n に変換する写像と考える (式(4))．S_n ($S_n=\{j|j\in N\}$) は自動変換できない変換項目つまり手作業変換が必要な変換項目の集合である．

$$S_n=f(S_N) \qquad (4)$$

n はコンバータが与えられた環境においてコンバータにより自動変換できない変換項目の集合となり，結果的に手作業による変換が必要な変換項目の集合である．

手作業変換項目は以下の2種類に分類できる．

- クラスa：ソースプログラムの変換作業で手作業変換する変換項目

  この場合は，すでにどの項目について変換する必要があるかあらかじめ分かっているため，変換工数としては，ソースプログラムの変換作業と確認試験のための工数だけである．

- クラスb：確認作業でバグとして発見する変換項目

  この場合は，移植先のコンパイラの内部処理に依存するため事前には修正すべき箇所が分からない場合などがある．バグとして顕在化するため，その原因の追求，その変換方法の検討のための工数およびソースプログラムの変換作業の工数が変換工数となる．

従って，そのコンバータを使用した移植で発生する手作業変換項目は，コンバータの機能 f を使って以下に示す a, b の二種類に分類される．

$$a=f(A) \qquad (5)$$

$$b=f(B) \qquad (6)$$

$$n=a+b \qquad (7)$$

ここで，クラスaに所属する変換項目の集合を A，クラスbに所属する変換項目の集合を B とする．

さらに特定の移植プログラムについて考えると，aおよびbの中からそのプログラムに存在する変換項目（それぞれ α, β とする）のみが手作業の対象となる．特定なプログラムに存在する手作業変換項目集合を求める変換項目集合から変換項目集合への写像を p とする．

$$\alpha=p(a) \qquad (8)$$

$$\beta=p(b) \qquad (9)$$

$$\eta=\alpha+\beta \qquad (10)$$

$$=p(n) \qquad (11)$$

式(4)～(11)から以下の式が導ける．

$$\alpha=p(f(A)) \qquad (12)$$

$$\beta=p(f(B)) \qquad (13)$$

$$\eta=p(f(N)) \qquad (14)$$

上記の各種変換項目の関係を**図3**に示す．

式(12)，(13)，(14)から，移植プログラムごとに固有な変換項目を求めるには，コンバータの機能と移植プログラムが持つ変換項目の情報が必要であることが

表2 S-part とプログラム規模の分散分析
Table 2 Variance analysis of S-part for program size.

	平方和	自由度	不偏分散	F値
回帰	$1.06*10^6$	1	$1.06*10^6$	44.9**
誤差	$9.45*10^4$	4	$2.36*10^4$	
合計	$1.16*10^5$	5		

** 有意水準1%で有意

表3 C-part とプログラム規模の分散分析
Table 3 Variance analysis of C-part for program size.

	平方和	自由度	不偏分散	F値
回帰	$4.48*10^4$	1	$4.48*10^4$	0.93
誤差	$1.92*10^5$	4	$4.80*10^4$	
合計	$2.37*10^5$	5		

表4 変換項目の例
Table 4 Examples of conversion item.

項番	変換項目	
	変換前	変換後
1	*I2/'ABC'/*	*I2/Z'4142'/*
2	*GOTO(K, K, ...)i* in the case of $i>100$	*GOTO(K, K, ...)i* *GOTO(K, K, ...)i−100*
3	DISPLAY 文がある．	コメントアウトする．
4	論理型データ初期値 T/F	TRUE/FALSE に変換

分かる．

ここで，変換項目 i が顕在化した原因の追求とその変換方法の検討のための工数を b_{1i} とし，ソースプログラムの変換作業の工数を b_{2i} とする．クラスaの変換項目の場合は，b_{2i} のみであるが，クラスbの変換項目の場合は，b_{1i} と b_{2i} の両方の工数が発生する．原理的には，上記の仮定からすべての変換項目について b_{1i} と b_{2i} が定まるはずである．また，式(12)，(13)，(14)から分かるように，変換しようとするプログラムに特有な変換項目に依存した形で変換項目が発生し，それぞれの変換項目の存在数も変換項目ごとにより異なる．それぞれの変換項目の存在数を n_j $(1 \leq j \leq N)$ とすると，全手作業変換工数 E_c は以下のようになる．

$$E_c = \sum_j^{\beta} b_{1j} + \sum_j^{\eta} b_{2j} n_j \qquad (15)$$

ここで，$\sum_j^{\alpha}$ は α のすべての要素 j についての総和を求めることを意味するものとする．

4.2.2 C-part の性質

式(15)に着目して，C-part の性質を分析する．クラスbにどの変換項目が分類されるかは，移植技術者のスキルレベルやプロジェクトの習熟度などの環境に依存する．従って，クラスbに分類される変換項目集合 β は環境が決まれば決定する．

また，式(15)の第一項目は以下の関係がある．

$$0 \leq \sum_j^{\beta} b_{1j} \leq \sum_j^{B} b_{1j} \qquad (16)$$

上記の分析から，ある範囲の類似プログラム群の移植では，環境定数として式(16)の範囲内の定数になると考えられる．そこで，第一項を定数と仮定し，C_{c1} と置く．

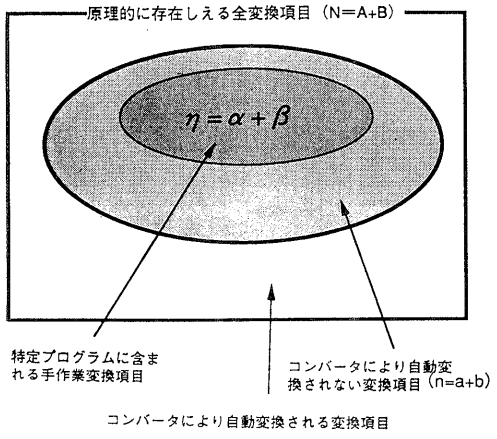


図 3 変換項目の関係
Fig. 3 Relationship among conversion items.

また，式(15)の第二項は，手作業で変換する工数であり，変換作業自身は単純作業が多いため b_{2j} のばらつきは比較的少ないと考えられる．そこで，b_{2j} を定数 C_{c2} と置いて，第二項は以下のように変形できる．

$$\sum_j^{\eta} b_{2j} n_j = C_{c2} \sum_j^{\eta} n_j = C_{c2} n_p \qquad (17)$$

ここで，$\sum_j^{\eta} n_j$ $(=n_p)$ は移植プログラム中に現れる手作業変換項目の総数になる．

以下の考察から，E_c は以下に示すような簡単な式になる．

$$E_c = C_{c1} + C_{c2} n_p \qquad (18)$$

ここで，C_{c1} と C_{c2} は定数である．

これは，移植プログラム中に存在する手作業変換項目の総数が分かれば E_c が計算できることを示している．

4.2.3 E_c の検証

ここでは，式(18)が実際に成り立っていることを検証する．移植プロジェクトデータから C-part のコスト（E_c）と総手作業変換項目数（n_p）との分散分析の結果を**表5**に示す．この結果から，C-part は総手作業変換項目数と強い相関があることが分かる．従って，E_c は式(18)に示す n_p の一次式で表現して問題がないことが分かる．

4.3 コストモデル

3章の検討結果から，式(3)と式(18)を加えた全移植工数すなわち移植コストモデルは次式となる．

$$\begin{aligned} C &= E_s + E_c \\ &= (C_{s1}L + C_{s2}) + (C_{c1} + C_{c2} n_p) \\ &= C_1 L + C_2 n_p + C_3 \end{aligned} \qquad (19)$$

ここで，C_1，C_2，C_3 は定数である．具体的な定数は移植環境により異なる．前章の解析より，コンバータの機能が影響する部分は式(19)には陽には含まれてい

表 5 C-part と手作業変換項目総数（n_p）の分散分析
Table 5 Variance analysis of C-part for the total number of manual conversion items (n_p).

	平方和	自由度	不偏分散	F 値
回帰	$2.21 * 10^5$	1	$2.21 * 10^5$	55.5**
誤差	$1.59 * 10^4$	4	$3.98 * 10^3$	
合計	$2.37 * 10^5$	5		

** 有意水準1%で有意

ない．しかし，総手作業変換項目数 n_p を求める過程でコンバータの機能が必要となる．つまり，コンバータの機能は n_p の中に陰に含まれている．

5. コストモデル評価

5.1 パラメータの推定

表2と表5の分散分析を行ったデータを元に回帰分析を行い回帰係数を求め，式(3)と式(18)のパラメータを決定した結果を次式に示す．

$$E_s = 16.9L + 112.2 \text{ (人時)} \quad (20)$$

$$E_c = 0.95n_p + 56.2. \text{ (人時)} \quad (21)$$

従って，全移植工数は次式となる．

$$C = 16.9L + 0.95n_p + 168.4 \text{ (人時)} \quad (22)$$

5.2 評　　価

式(22)の妥当性を評価するために，総移植工数を縦軸に移植プログラム規模を横軸にし，本モデルによる計算値と実測値を**図4**に示す．この図に示されたプログラムは，工数の測定対象となった6本のプログラム(総規模 131 KLOC) である．また，実測値と計算値を**表6**に示す．この表から分かるように，本モデルの場合の平均相対誤差は約 23.8% である．一方，規模のみを変数とした一次式モデルの場合，最小2乗法を用いた評価式は以下のようになる．

$$C = 20.2L + 228.4 \text{ (人時)} \quad (23)$$

この場合の平均相対誤差は約 41% である．本モデルを使用した方がより正確にコスト評価を行うことができることが分かる．

表6に示すように，本モデルによる移植工数評価では移植プログラムの規模が大きくなるほど相対誤差が小さくなる傾向があることが分かる．これは，本モデルが規模と総手作業項目数の関数となっているため，これらの値が小さい時は個々のばらつきが大きく影響するが，大きくなるほど平均化されてくるためと思われる．

5.3 n_p の予測法に関する考察

式(19)で表現されるモデルは，総手作業変換項目数 n_p が変数として含まれるため，このモデルを適用する場合にはあらかじめ移植プログラムに含まれる総手作業変換項目数を予測する必要がある（移植の場合は新規開発と異なりプログラム規模はすでに判明しているので予測する必要はない）．

総手作業変換項目数を知るためには，原理的に存在し得るすべての変換項目集合 (S_N) とコンバータの機能 (f) と特定のプログラムに存在する変換項目を求める写像 (p) から，プログラムに存在する変換項目集合 (α, β) を求め，α, β に対応した変換項目の存在数 (n_i) の情報を求める必要がある．この中で，N と f は移植元と移植先の環境および使用するコンバータが決まれば値が定まる性質があるため，個別の移植プログラム対応に求める必要はない．このような性質のパラメータは大規模な移植や定常的に行われるプロジェクトにおいてはあらかじめ求めておけるので，コスト予

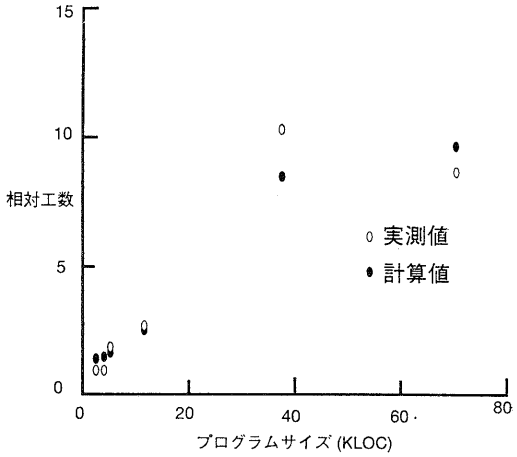


図4 実測値と評価値の比較
Fig. 4 Practical data and estimated values.

表6 実測値と計算値
Table 6 Variables, practical data and estimated values.

プログラム	規模 (KL)	n_p	実測値 (人時)	本モデル		規模の一次式モデル	
				計算値(人時)	相対誤差(%)	計算値(人時)	相対誤差(%)
A	2.6	18	155	229	47.7	281	81.3
B	3.5	10	160	237	48.1	299	87.0
C	4.7	25	310	272	12.3	323	4.2
D	11.6	44	420	406	3.3	463	10.2
E	37.2	573	1620	1341	17.2	980	39.5
F	70.6	174	1340	1527	14.0	1655	23.5

測上大きな問題にはならない．

一方 α，β と n_j は移植プログラムに依存する言わば移植プログラム特性と考えられるような性質を持っている．従って，個別の移植プログラムごとにそれぞれの値を求める必要がある．この場合，n_p は以下に示す式で表現される．

$$n_p = \sum_{j}^{\alpha} n_j + \sum_{j}^{\beta} n_j \qquad (24)$$

上式の右辺第1項はクラスaに属する変換項目に関する値であるためあらかじめプログラムを丹念に調べれば値を求めることができるが，第2項はクラスbに属する変換項目に関する値であるため，移植した結果からしか値が求まらない．また，第1項に関しても，原理的にはすべて数えあげることは可能であるが，実際には，時間的制約などから正確にカウントできるとは限らない．従って，n_p については，ソフトウェアの新規開発時のプログラム規模予測と同様な予測手法が必要になる．

以下では，ツールを使用して変換項目を予測する方法について考察する．十分に経験した環境では，たとえコンバータで自動変換できなくても，変換位置を検出するツールを構築することができる場合がある．測定プロジェクトで使用したコンバータ FSCONV はあらかじめ移植環境を調査し考えられる変換項目についてその存在を知らせる機能を持っている．測定したプログラム9本，総規模 226 KLOC で，FSCONV が検出できた手作業変換項目数は 1334 であり，一方総手作業変換項目数 n_p は 1348 であった．この場合に，自動検出率は 99% となる．従って，十分経験した移植環境かあるいは十分に調査した移植環境であるならば検出ツールなどを用いることにより n_p を非常に正確に予測できることが分かる．ただし，クラスbの手作業変換項目は，単純な予測モデルに従わない場合があり，十分に予測できないことがある．この場合の手作業変換工数は予想以上に高くなる場合が多いため，手作業変換項目の自動検出確率が高いからといって必ずしも正確に工数を予測できるとはかぎらない場合があるので注意を要する．

6. おわりに

同一言語間のソフトウェア移植時のコストモデルを提案した．本モデルはシンプルな構造であるが，正確な工数の評価を可能にする．本モデルは同一言語での移植に限定しているが，言語が変わる移植においても大きな構造変化が伴わないレベルの移植であるならば，C-part と S-part に分割して考える基本コンセプトは有効であると思われる．

新規開発におけるプログラム規模に対応する重要な変数は，本モデルでは総手作業変換項目数 n_p である．従って，新規開発時と同様に，工数をあらかじめ評価したい場合は，この n_p を移植前に正確に予測することが重要となる．予測手法については，本論で検出ツールを使用する方法について提案したが，あらかじめ変換が予想される箇所をソースプログラム中で集中させる方法あるいはソースプログラム中に移植を考慮した記述をする[12]などの方法も有効であると思われる．

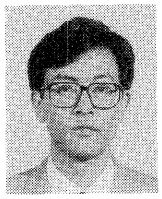
本モデルのパラメータは移植環境により変化する．従って，今後は個々の移植プロジェクトについてパラメータを求め，移植環境と定数値との関係を蓄積して行くことが重要である．

参考文献

1) 山田 茂：ソフトウェアマネジメントモデル入門1，2，3，完，bit, Vol. 22, No. 12 (1990), Vol. 23, No. 1, 2, 3 (1991).
2) Boehm, B. W.: *Software Engineering Economics*, Prentice-Hall (1981).
3) Putnam, L. H.: A General Empirical Solution to the Macro Software Sizing and Estimating Problem, *IEEE Trans. Softw. Eng.*, Vol. SE-4, No. 4, pp. 345-361 (1978).
4) Albrecht, A. J. and Caffney, J. E., Jr.: Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation, *IEEE Trans. Softw. Eng.*, Vol. 9, No. 6, pp. 639-648 (1983).
5) Wolberg, J. R.: Comparing the Cost of Software Conversion to the Cost of Reprograming, *SIGPLAN Notices*, Vol. 16, No. 4, pp. 104-110 (1981).
6) Wolberg, J. R.: A Costing Model for Software Conversion, *Software-practice and Experience*, Vol. 12. pp. 1043-1049 (1982).
7) 金井 敦，高橋宗雄，古山恒夫：プログラム移植コスト予測の一方法，信学会全国大会予稿集，1723，p. 7-45 (1984).
8) 金井 敦，高橋宗雄，古山恒夫：プログラム移植工数モデルの応用例，第29回情報処理学会全国大会論文集，pp. 463-464 (1984).
9) 金井 敦，高橋宗雄，古山恒夫：コンバータの効果とプログラム特性を考慮したプログラム移植工数モデルとその評価，情報処理学会ソフトウェア工学研究会，SE 40-6，pp. 31-36 (1985).

10) 藤田晨二，葛山善基，川原洋人：プログラムの移植について，情報処理，Vol. 21, No. 11, pp. 1128-1135 (1980).
11) Kanai, A., Takahashi, M. and Furuyama, T.: A Cost Model for Software Conversion Based on Program Characteristics and a Converter Effect, *Proceeding of COMPSAC '92*, pp. 63-68 (1992).
12) Hague, S. J. and Ford, J. B.: Portability—Prediction and Correction, *Software-practice and Experience*, Vol. 6, pp. 61-69 (1976).

(平成6年2月28日受付)
(平成6年7月14日採録)



金井　敦（正会員）

昭和55年東北大学工学部通信工学科卒業．昭和57年同大学院工学研究科情報工学教室修士課程修了．同年日本電信電話公社横須賀電気通信研究所入所．以来，ソフトウェア移植工数モデル，マルチターゲットクロスコンパイラ，分散開発環境，通信ソフトウェア開発法の研究に従事．現在，NTT ソフトウェア研究所にて，ソフトウェアの開発方法論，流通技術などの研究に従事．電子情報通信学会，IEEE 各会員．

古山　恒夫（正会員）

1945年生．1968年東京大学工学部計数工学科卒業．1973年同大学院博士課程修了．同年日本電信電話公社入社．横須賀電気通信研究所で，拡張型言語，Ada, Common LISP などの言語処理プログラムの研究実用化に従事．現在，日本電信電話(株)ソフトウェア研究所でソフトウェアプロジェクト管理法，ソフトウェア品質保証法，ソフトウェア見積り法などの研究実用化に従事．電子情報通信学会．日本ロボット学会各会員．工学博士．

高橋　宗雄（正会員）

昭和19年生．昭和42年千葉大学工学部電気工学科卒業．同年日本電信電話公社電気通信研究所入社．平成3年桐蔭学園横浜大学工学部制御システム工学科助教授，現在に至る．工学博士(九州大学)．システム製造用言語，ソフトウェアメトリックス，ソフトウェア品質管理技術などの研究に従事．著書（共著）「ソフトウェアマネッジメントモデル入門」(共立出版）など．昭和59年情報処理学会論文賞受賞．電子情報通信学会，日本ソフトウェア科学会，IEEE 各会員．