ソフトウェア開発工数論

浅　井　邦　彦*

The Theory of Man-Month on the Development of Software

Kunihiko Asai

There are two kinds of way to utilize software for computer in business: one is to buy the package of software; the other, to develop the new software adapted to each works. Compared to clothes, the former is a ready-made, whereas the latter is an order-made.

In these days, the diverse and cheap packages of software are available, and it is effective for a company to appropriately utilize them.

However, the expansion of the sphere to utilize computer does not always indicate the presence of the packages suitable for a work, but often does the necessity to develop a new one.

In that case, it is difficult to foresee a prospect of the man-month and costs to develop software. For we cannot use the past achievement of the development of software, as it is, which is always "novelty" such as the creation of novel and music.

Each companies usually depend on an expert in software such as a warehouse, rather than solve by itself. Even if so, it is important for the company as a customer to foresee the man-month and costs to develop software as well as to expect the term to finish ordered software.

In this paper, we will keep our eyes on man-month under the preposition that costs could be counted by man-month.

はじめに

企業におけるコンピュータソフトウエアの利用方法は二種類に大別される。一つは既成のパッケージソフトの購入であり、もう一方は、その業務に適したソフトウエアを開発することである。服にたとえれば、前者は既製服であり、後者は注文服である。

＊経営工学科

最近、パッケージソフトも豊富に、安価に出回っており、それらを適宜、利用することは、企業にとっても有効なことである。

しかし、コンピュータの利用分野が広がるにつれて、対象業務に適したパッケージソフトがあるとは限らず、新たに開発しなくてはならないことも多い。

その場合、ソフトウエア開発の工数やコストについて見通しがなかなか立てにくいものである。それは、小説や音楽の創作のように、一般に、常に「新しい」ものであるので、過去の実績は、そのままは利用できないためである。

ふつう、実際のソフトウエア開発自身を各企業が自社内で担当することよりも、ソフトウエアハウス等、「専門」のところを利用することが多いが、その場合でも依頼者である一般企業が、ソフトウエア開発工数とコストとを見通しておくことは、完成期間を予定しておくのと同様に重要である。

当稿では、主に工数を考えることにし、コストは工数から計算されるという前提を置くことにした。

１．工数を決定する要因

工数を決定する要因として、（１）作業範囲、（２）創造性、（３）記述言語、（４）難易性の４種に分けられる。この他に、事務計算か技術計算か「対象分野」や、バッチ処理かオンライン処理かなどの「使用種別」も、要因として考えられるが、これらは（１）～（４）に間接的に含まれる。

１．１　作業範囲（R）

ソフトウエア開発工数は、要求定義の分析から最後の資料作成までのいろいろなステップがあるが、実際の開発にあたって、既に開発ずみのものを流用したりすることがあるので、ここではステップ毎にその作業があるかないかを調べる。

それぞれのステップで、どのくらいの工数がかかるのかとか、１つのステップ内での部分流用については、次項以下に述べる。

＜ステップ＞

要求定義の分析　：　R_r（あれば１、なければ０）
設計　：　R_d（あれば１、なければ０）
プログラミング　：　R_p（あれば１、なければ０）
総合テスト　：　R_t（あれば１、なければ０）
まとめ、資料作成：　R_a（あれば１、なければ０）

１．２　創造性（Ｇ）

これは記述言語に対応しながら、新規か既成のソフトウエアの部分的再利用かの度合いである。

新規作成見積り行数　　：G_s（コメントを除く）
修正作成見積り行数　　：G_m（コメントを除く）
修正のための見直し見積り行数：G_t（コメントを含む）
　↖これは修正の影響がどのくらいあるかを調べるための調査用行数である。

なお、プログラム行数自身の見積りは、いろいろと報告されているので、ここでは深く立ち入らない。一応の目安として、プログラミング（R_p）だけだったらＦＯＲＴＲＡＮで２０００行／人月と仮定している。

１．３　記述言語（Ｌ）

ここでは使用している記述言語が何んであるかを問うている。その行数については、１．２創造性の項で述べた。

なお、下記に該当しない言語は、類似性から代用する。たとえば、ＡＬＧＯＬ（現在ほとんど使われていないが）ならば、ＦＯＲＴＲＡＮで置き換えてみる。

ただし、ＡＩ言語であるＰＲＯＬＯＧやＬＩＳＰ、オブジェクト指向言語のＳＭＡＬＬ－ＴＡＬＫなどは対象としない。これらについては別に検討することとする。

ＦＯＲＴＲＡＮ：L_{FO}（あれば１、なければ０）
ＣＯＢＯＬ　　：L_{CO}（あれば１、なければ０）
アセンブル言語：L_{AS}（あれば１、なければ０）
Ｃ言語　　：L_{CL}（あれば１、なければ０）

１．４　難易性（D_J）

ソフトウエア開発の難易性の目安を、階層性によることとする。

なお、ここで言う難易性は、ソフトウエア開発自身のものであって、それ以前の問題は含まない。たとえば、原子力の反応に関するソフトウエアを開発する必要が生じた場合、その工数は算定の対象になっても、原子力の物理的解明は範囲外である。もっとも、これらの問題も間接的には難易性に含まれることが多い。

第１階層のモジュール数：D_1（主プログラム相当）
第２階層のモジュール数：D_2（子のサブプログラム相当）
第３階層のモジュール数：D_3（孫のサブプログラム相当）
第４階層のモジュール数：D_4（曽孫のサブプログラム相当）

２．工数の見積り式

１項で述べた要因を使って、工数の見積り式を設定する。工数は人月でMMとする。

$$MM = \left(\frac{D_1 + 1.5D_2 + 2D_3 + 3D_4}{D_1 + D_2 + D_3 + D_4}\right) \times \{f(G_s, G_m, G_f)(L_{FO}, L_{CO}, L_{AS}, L_{CL})\}^{*注} \times \{(\alpha R_r + 4\beta R_d + 2R_p + 2R_t + R_a)/10\}/400 \quad \cdots\cdots (1)$$

*注（１）式の$f(G_s, G_m, G_f)(L_{FO}, L_{CO}, L_{AS}, L_{CL})$の意味は次の通りである。

$$f(G_s, G_m, G_f)(L_{FO}, L_{CO}, L_{AS}, L_{CL}) =$$
$$\text{FORTRAN}: (G_s + 2G_m + 0.05G_f) \times L_{FO}$$
$$+\text{COBOL} : (G_s + 2G_m + 0.05G_f) \times 1.5L_{CO}$$
$$+\text{アセンブル語} : (G_s + 2G_m + 0.05G_f) \times 0.5L_{AS}$$
$$+\text{C言語} : (G_s + 2G_m + 0.05G_f) \times 2.5L_{CL}$$

＜上記式の記号の説明＞

- D_iについた係数は難易の度合いを示している。
- α、βは修正の度合いを表わす係数で、全部新規ならば１、半分修正ならば０．５となる。
- α、β以外のR_jに関する係数は、各ステップでの工数の相対的重みを表わしている。たとえば、$2R_p$、R_aとなっているので、R_p(プログラミング)はR_a(まとめ)に対して、２倍の重みをかけている。
- L_Kに関する係数は、プログラム言語間の行数の違いで、たとえば、L_{FO}、$0.5L_{AS}$ なので、同じ問題ならば、アセンブル語は、ＦＯＲＴＲＡＮに比べて、行数が２倍かかることを表している。
- G_i関する係数は、新規か修正かの相対的違いを表している。
- ／１０と／４００は、人工（MM）に直すための調整値である。

3．具体例

技術計算を主体としたソフトウエア開発の工数を例にとる。ただし、プログラム自身は過去の実績等から推定して見積もる。

（1）作業範囲

要求定義から最後のまとめまで、すべてを対象とする。

$R_r = 1$，$R_d = 1$，$R_p = 1$，$R_t = 1$，$R_a = 1$

（2）創造性と記述言語

流用はなく、すべて新規作成とする。（$G_m = 0$、$G_t = 0$）

$\alpha = 1$、　$\beta = 1$

FORTRAN　12000行：$G_s = 12000$　$L_{FO} = 1$

アセンブル語　　2000行：$G_s = 2000$　$L_{AS} = 1$

（3）難易性

$D_1 = 10$、$D_2 = 5$、$D_3 = 3$、$D_4 = 0$

次に上の（1）、（2）、（3）の各数値を前の頁のMM式に代入

$$MM = \left(\frac{10 + 1.5 \times 5 + 2 \times 3 + 3 \times 0}{10 + 5 + 3 + 0}\right) \times$$

$\{(12000 + 2 \times 0 + 0.05 \times 0) \times 1$　←FORTRAN

$+ (2000 + 2 \times 0 + 0.05 \times 0) \times 0.5 \times 1\} \times$　←アセンブル語

$\{(1 \times 1 + 4 \times 1 \times 1 + 2 \times 1 + 2 \times 1 + 1) / 10\} / 400$

$\fallingdotseq 42.43$（MM）

42.43人月（MM）ということは、担当者の平均人数を4人とすれば、この作業に約10.5ケ月かかることを示している。

４．展望と課題

前章の具体例において、４２．４３人工で４人を当てはめれば、割り算して、１０．５ケ月かかるとした。これは経験的にも妥当である。しかし、もしも４２人で取りかかれば、１ケ月で完成できるかというと、そうはいかない。

下流部分になるほど、いわゆる「人海戦術」が効いてくるが、上流部分は「少数精鋭」が望ましい。また、ソフトウエアの構成が独立性の強いものの集合であるほど、人手をかけて短期間に仕上げることができる。

一定期間に同時にかける人数について、きわめてマクロ的に判断すると、要求定義の分析と設計を１としたとき、それ以降は３ぐらいが望ましい。

その他、担当者の質的レベルを考慮することも重要であるが、当論文では平均的レベルという前提とした。（筆者の経験では上流になるほど担当者の質的レベルの差異によって開発の進捗に大きな違いがでてくることがわかったが、この問題については別の機会に考察することにし、これ以上は深堀しないこととする。）

なを、ＴＲＷのＢöｅｍ氏が工数について、ソフトウエア工学の観点から論述しているが、どちらかと言うと、下流部分に力点をおいている。

（平成12年10月18日受理）