ソフトウェア工学 109－5
（1996．5．17）

作業の並行化による影響を考慮した開発プロセスシミュレーションモデルの提案

永島 淳　飯田 元*　松本 健一　鳥居 宏次

奈良先端科学技術大学院大学 情報科学研究科
*奈良先端科学技術大学院大学 情報科学センター

〒630-01 奈良県生駒市高山町 8916-5

E-mail:{jyun-e, iida, matumoto, torii}@is.aist-nara.ac.jp

現状のソフトウェア開発においては，上流工程の作業の完了を待たずに下流工程の作業を開始することによって，開発期間の短縮化が図られることが多くなってきている．しかし，上流工程・下流工程の作業を並行して行うことにより，作業間のコミュニケーションによるオーバーヘッドが増大し，開発総工数は増加すると考えられる．そこで本研究では，作業を並行して行うことが，開発期間と開発総工数に与える影響を予測するための方法として，並行作業間の依存関係を考慮した進捗モデルを提案し，それに基づく開発プロセスのシミュレータを開発した．開発したシミュレータでは，個々の作業に対して要員の割当や作業の開始時刻・条件を変更することにより，開発期間と開発総工数への影響を観察することで，プロジェクト立案をサポートできる．

Simulation Model of Development Process Based on Progress of Activities

Jun EIJIMA, Hajimu IIDA*, Ken-ichi MATSUMOTO and Koji TORII

Graduate School of Information Science, Nara Institute of Science and Technology
* Information Technology Center, Nara Institute of Science and Technology

8916-5 Takayama, Ikoma, Nara 630-01, Japan

In many cases of software development, development time is reduced by starting downstream activities before upstream activities have completely finished. However, by carrying out upstream and downstream activities in parallel, the dependencies between activities have become complex. This results the communication between activities to increase, which will also result in the total work effort to increase. This paper proposes a model to describe the parallel development process and the effects that each activity executed in parallel have with each other. Based on this model, a software process simulator was developed. By using this simulator, it is possible to support project planning (scheduling and staffing) with constraints on cycle time and total work effort.

1 はじめに

ソフトウェアの需要量が増大するにつれ，製品寿命の短期化が進み，開発期間の短縮化の傾向が強まってきている [8][11][12].

開発期間の短縮化のため，ソフトウェア開発の現場においては，新たな開発法の導入など，様々な手段を試みている．とりわけ，プロジェクト管理においては次の 2 種類の方法を採ることにより，開発期間の短縮化を図っている (図 1参照).

1 つ目の方法は，作業への要員の投入数を増加することによって，各作業にかかる時間を短縮する方法である (図 1(b)). 各作業にかかる時間を短縮することができれば，結果的に開発期間を短縮することができる．しかし，作業に割り当てる要員数を増加すると，作業者間のコミュニケーションによるオーバーヘッドが増加してしまい開発総工数は増加する [3].

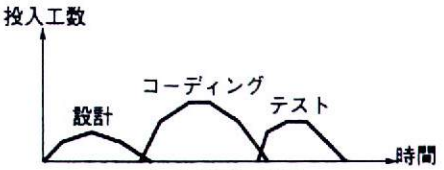
二つ目の方法として，上流工程の作業の完了を待たずに，下流工程の作業を開始することによって開発期間の短縮化を図ることも多くなってきている [8](図 1(c)). このような開発方法を並行作業による開発と呼ぶことにする．しかし，作業を並行に行うことにより，上流工程の作業の成果が，不完全なまま下流工程の作業に与えられることになる．つまり下流工程の作業に与えられる情報量が少なくなるため，問い合わせが必要になり，コミュニケーションにかかるオーバーヘッドが増加する [4]. このため，各作業の完了までにかかる時間が長くなり，その結果，全体の開発期間としては見かけ上短縮されるが，開発総工数 (各作業に要する時間の総和) は増加すると考えられる.

以上のように，要員の投入数を増加する方法，作業を並行化する方法のどちらにおいても，開発期間を短縮すると開発総工数は増加すると考えられる．実プロジェクトにおいては開発期間だけではなく，開発総工数 (コスト) にも制約があるため，開発期間と開発総工数のトレードオフを考慮した上で開発計画を決定することがプロジェクトの管理において重要である [5][9][10].

従来より，このような問題に対し，開発プロセスをモデル化，シミュレートすることによって，開発期間と開発総工数の関係を予測し，開発計画のサポートすることが試みられている [6][7].

しかし，従来提案されてきたモデルでは，要員投入による開発期間の短縮を対象としており，作業を並行化することによる開発期間の短縮効果に関しては，考慮されていないものがほとんどである.

そこで本稿では，多人数での並行作業による開発を記述するためのプロセスモデルと，並行作業間の影響を記述するための進捗モデルの 2 つのモデルを提案する．また，本モデルを小規模な例題プロセスに適用し，並行作業による開発時の開発期間と開発総工数の関係を予測することを試みる.



(a) 通常の開発

(b) 各作業への要員投入数の増加による開発期間の短縮化

(c) 作業の並行化による開発期間の短縮化

図 1: 開発期間の短縮化の方法

2 関連する研究

本章では，関連研究として，まず開発期間と開発総工数の一般的な関係を示したソフトウェア方程式について説明する．また，開発期間と開発総工数の関係を予測し，開発計画のサポートを試みた従来のシミュレーションモデルについて，その概要を説明し問題点について述べる.

2.1 ソフトウェア方程式

COCOMO モデルの embedded モードにおいて，開発総工数 MM_{DEV}(人月) がわかった時，期待される開発期間 T_{DEV} は次式で求まる [2].

$$T_{DEV} = 2.5 \times (MM_{DEV})^{0.32}$$

Putnam は，この期待される開発期間 T_{DEV} より短期間で完了した過去のプロジェクトの実績データから，次のソフトウェア方程式を導き出した [9].

$$S_s = C_k K^{1/3} t_d^{4/3}$$

ここで,

S_s : システムのサイズ
K : 開発総工数
t_d : 開発期間
C_k : 生産性パラメータ (組織の過去の実績データから求める値)

上式より開発期間と開発総工数の関係は次式で表される.

$$K = \frac{constant}{t_d^4}$$

つまり, 期待される開発期間より短期間でプロジェクトを完了させようとすると, 開発総工数は急激に増加することを表している.

実際の開発プロセスにおいては, 要員投入による方法, および作業の並行化による方法の両方法を用いることで開発期間の短縮化が図られている. Putnam の導いた開発期間と開発総工数の関係は, 過去のプロジェクトの実績データから得られたものであるため, 要員投入による方法, 作業の並行化による開発期間の短縮の両方の効果を集約したものであると考えられる.

2.2 従来のシミュレーションモデル

STATEMATE[7]

Kellner は, STATEMATE を用いて機能, 動作, 構成の 3 つの側面から開発プロセスを詳細に記述し, シミュレートすることによって, スケジューリングおよび開発総工数の見積りを試みている.

STATEMATE では, プロセスを詳細に記述できるため, 作業を並行して行うような開発プロセスをはじめとする, 様々なプロセス形態を記述することができる. また, 要員の投入数を変更しシミュレートすることができるため, 要員の投入による開発期間の短縮効果を見ることができる. その反面, 作業の開始条件を変更することができないため, 作業の並行化による開発期間の短縮効果を見ることはできない.

平山らのモデル [6]

平山らは, プロセスを一般化確率ペトリネットに基づきモデル化, シミュレートすることで, 品質, 開発期間, 開発総工数の観点からを総合的に分析することを試みている.

このモデルでは, 各作業の開始条件を記述することができ, 上流の作業からの影響を考慮しているため, 作業の並行化による開発期間の短縮効果を見ることができる.

しかし, 作業の並行化による影響として, 下流の作業からの影響をモデル化することはできない.

3 提案モデル

図 2に提案モデルの概略を示す. プロセスモデルでは, 開発プロセスを作業の集合, スタッフの集合, および, 作業の前後関係, スタッフの割当で定義し, 作業, スタッフに対してさまざまな属性を付加している. 作業の属性としてその開始時刻を与えることにより, 並行作業による開発プロセスを記述することができる.

また, 各作業はそれぞれ進捗モデルを持つ. 進捗モデルは, プロセスモデルで表される情報をもとに, 作業の進捗の時間変化を計算するための数式モデルである. 作業の進捗を計算する際に, 前後関係のある作業の属性をパラメータとして含むため, 並行作業間の影響をモデル化することができる.

本モデルでは, プロセスモデルに基づいて記述された情報をもとに, 各作業の進捗モデルに従って進捗の変化を計算することで, プロセスの進行をモデル化することができる.

このように本モデルでは作業の進捗という概念をもとにプロセスの進行を考える. そこでまず, 作業の進捗を定義し, 次に, プロセスモデル, 進捗モデルの 2 つのモデルについて説明する.

3.1 進捗の定義

ここでは, 作業の進捗を次のように定義する.

まずプロジェクトは, 解決を見つける必要のある, 多数の問題の集合であると考える. この問題の集合を Q, 解決の集合を Q' とする. 開発プロセスは問題の集合 Q を, 問題の変換を繰り返すことにより解決 Q' を導く過程であると考える.

$$Q \rightarrow Q_1 \rightarrow Q_2 \rightarrow \cdots \rightarrow Q_n = Q'$$

開発プロセスにおけるそれぞれの変換を作業と考える. つまり, 作業は与えられた問題 Q_{k-1} の解決 Q_k を発

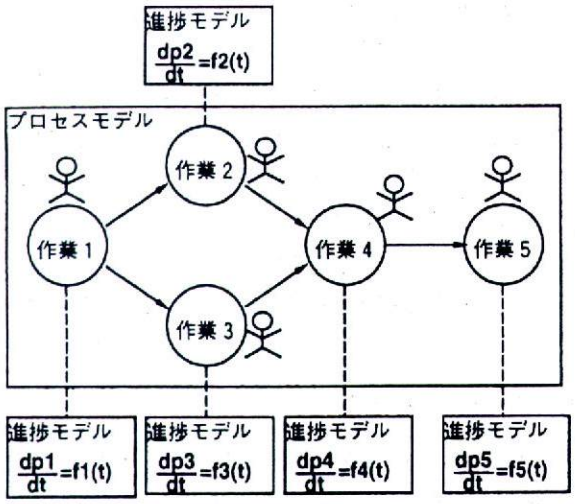


図 2: モデルの概略

見する過程であり，解決 Q_k は次作業に与えられる問題となる．

時刻 t における作業の進捗 $p(t)$ を次のように表す．

$$p(t) = \frac{W(t)}{|Q_k|}$$

ここで $W(t)$ は，時刻 t においてすでに解決された問題の数を表す．

つまり，解決された問題の割合を作業の進捗と定義する．

$$0 \leq p(t) \leq 1$$

作業への入力となるプロダクトと，出力となるプロダクトは一般に異なるため，問題 Q_k の規模をプロダクトの行数で計測すると，問題の規模とその解決の規模が異なる値になる．

また，コーディング言語の違いによっても問題の規模が異なる値になる．従って，ファンクションポイント法 [1] や DeMarco のモデル [5] 等を用いて，プロダクトの種類の違いやコーディング言語の違いに関係なく問題の規模を計測することが必要である．そこで以降は，ファンクションポイント法を用いて問題の規模 (FP) を計測するものとする．

3.2 プロセスモデル

プロセスモデルでは，開発プロセスを作業の集合，スタッフの集合，および，作業間の前後関係，スタッフの割当としてモデル化する．

表 1: 作業・スタッフの属性

作業 $a \in A$ の属性	
進捗	$0 \leq p(t) \leq 1$
開始条件	$t = T_s(day)$ または $q(t) = P_s$
	$q(t)$ は上流作業の進捗
終了条件	$t = T_e(day)$ または $p(t) = P_e$
規模	$L(FP)$
難易度	$D > 0$

スタッフ $s \in S$ の属性	
能力	$C(FP/day)$

開発プロセス P を次の 4 つ組で定義する．

$$P = (A, R, S, G)$$

ここで，

A : 作業の集合
R : 作業の前後関係．$R : A \rightarrow A$
S : スタッフの集合
G : スタッフの作業への割当．$G : S \rightarrow A$

開発プロセスの構成要素である作業，スタッフには，それぞれの概念を特徴付ける属性を付加する (表 1)．

作業には 5 種類の属性 (進捗，開始条件，終了条件，規模，難易度) を与える．進捗は 3.1節での定義に基づき，時刻 t までに解決した問題の割合を示す．開始条件は次の 2 種類の条件を考える．

1. 時刻 $t = T_s$ に作業を開始する．
2. 上流の作業の進捗 $q(t) = P_s$ になったら作業を開始する．

同様に終了条件についても 次の 2 種類を考える．

1. 時刻 $t = T_e$ に作業を終了する．
2. 自作業の進捗 $p(t) = P_e$ になったら作業を終了する．

規模 L は，作業で解決すべき問題の規模をファンクションポイント法で計測した値である．難易度は作業の難しさを表す定数である．この値は，プロジェクトの実績データから求まる値である．スタッフには属性として能力値（単位は FP/day） を与える．

3.3 進捗モデル

以降の説明のために，作業 a の上流作業，下流作業を次のように定義する．

$$\begin{aligned} \text{上流作業 } b_i &: b_i \in A, b_i R a \\ \text{下流作業 } c_j &: c_j \in A, a R c_j \end{aligned}$$

各作業は，上流作業，下流作業の影響を受け，進捗がどのよう変化するのかを計算するための進捗モデルを持つ．ここでは，進捗モデルは 1 階常微分方程式で表し，進捗の変化に次のような仮定をおく．

- 進捗の増加速度は，割り当てられたスタッフの能力，作業の規模，難易度によって変化する．
- 進捗の増加速度は，上流作業の進捗と自作業の進捗の差に比例する．
- 下流作業の進捗の影響を受けない．

これらの仮定を満たす進捗モデルを形式的に記述すると次のようになる．

$$\frac{dp}{dt} = \begin{cases} f(t) = K(q(t) - p(t)) & \text{(作業実行中)} \\ 0 & \text{(作業非実行中)} \end{cases}$$

$$p(0) = 0$$

ここで，

$p(t)$: 自作業の進捗
$q(t)$: 上流作業の進捗
K : 進捗速度係数 $(K > 0)$

作業中かどうかは開始条件，終了条件より判断できる．

上流作業が複数存在する場合，$q(t)$ として，各上流作業の進捗 $q_i(t)$ の加重平均した値を用いることにする．重みは，各上流作業の規模，難易度に応じて配分する．

K は自作業に割り当てられたスタッフの能力 C，自作業の規模 L，難易度 D によって決まる値であり，作業の並行化による影響に関係なく次のように求まる．

1. まず，他の作業の影響を受けない場合の進捗モデルの式は，

   $$\frac{dp(t)}{dt} = K(1 - p(t))$$

   この微分方程式を解くと次式になる．

   $$p(t) = 1 - \exp(-Kt)$$

   作業終了時には次式が成り立つ．

   $$P_e = 1 - \exp(-Kd) \qquad (1)$$

   ここで，

   P_e : 作業の終了時の進捗 (終了条件)
   d : 作業にかかる日数の見積り値

2. 作業の規模 L を見積もる．
3. 作業の難易度 D を与える．
4. 作業を行うスタッフの能力 C を与える．
5. ここで，作業にかかる日数の見積り値 d は次の式で算出される．

   $$d = DL/C \qquad (2)$$

6. よって (1) 式，(2) 式より，進捗速度係数 K が求まる．

   $$P_e = 1 - \exp(-KDL/C)$$

   $$K = -\frac{C \log(1 - P_e)}{DL}$$

このように，進捗の変化が，自作業の属性と自作業に割り当てられたスタッフの属性のみに影響されるのではなく，上流作業の属性にも影響されるように考えることにより並行作業間の影響を記述している．

3.4 モデルの特長

本モデルは次のような特長を備えている．

1. 作業の属性として開始条件を含むため，並行作業による開発プロセスを記述することができる．
2. 作業の属性として終了条件を含むため，作業の完了度 (プロダクトを作成する作業ならプロダクトの完成度) を変化させることができる．
3. スタッフの属性として能力を含むため，スタッフの能力の違いによる処理速度の違いの影響を表すことができる．
4. 進捗モデルにおいて作業の進捗を計算する際に，その並行作業の属性をパラメータとして含むことにより，並行作業間の影響をモデル化することができる．

また，各作業の開始条件，終了条件を様々に変化させることにより，作業の並行化による開発期間，開発総工数，プロダクトの完成度に対する影響を予測することが可能となる．

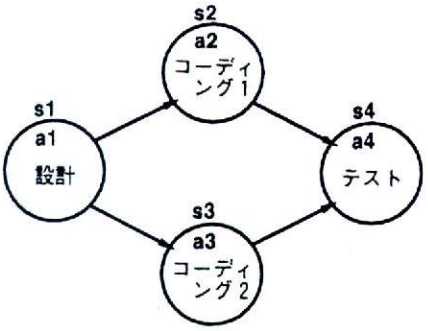


図 3: 例題プロセス

4 シミュレーション例

4.1 プロセスモデルによる記述

例として，図 3のようなプロセスモデルで記述される小規模な開発プロセスを考える．この図で，円は作業 (a_i) を，矢印は作業間の前後関係を表し，矢印の元の作業の次に，矢印の先の作業が行われることを示す．また，s_j は各作業に割り当てられたスタッフを表す．

要求仕様はすでに決定されており，開発プロセスは設計作業から始まり，ソースコードを作成しテストを行うまでとする．

例題プロセスは 4 つの作業 ($a_1 \sim a_4$) からなる．また，スタッフは 4 名おり ($s_1 \sim s_4$)，各作業に 1 人ずつ割り当てている．

$$
\begin{aligned}
A &= \{a_1, a_2, a_3, a_4\} \\
R &= \{(a_1, a_2), (a_1, a_3), (a_2, a_4), (a_3, a_4)\} \\
S &= \{s_1, s_2, s_3, s_4\} \\
G &= \{(s_1, a_1), (s_2, a_2), (s_3, a_3), (s_4, a_4)\}
\end{aligned}
$$

4.2 シミュレーション結果と考察

図 4は，上流作業の進捗が (a)100%，(b)70%，(c)20% になった時点で作業を開始するという条件で，図 9 の例題プロセスのシミュレーションを行った結果である．ここで，a_1, a_2, a_3, a_4 はそれぞれ設計，コーディング 1，コーディング 2，テスト作業の進捗の時間変化を示している．また，effort は累積工数の変化を示している．

図 5は各作業の開始条件を様々に変化させシミュレートした時の，開発期間と開発工数の関係を示すグラフである．このグラフより，各作業を上流作業の進捗が 70% になった時点で開始するという条件より早める，例えば，上流作業の進捗が 60% になった時点で，下流作業を開始するとしても，開発期間はほとんど短縮されないが，開発総工数は急激に増加することがわかる．

Putnam は，期待できる値よりも開発期間を短縮しようとすると開発工数が急激に上昇することを，過去のプロジェクトのデータから示している [9]．Putnam のこの観測は，各作業への要員投入数の増加による開発期間の短縮化だけでなく，作業の並行化による開発期間の短縮化の影響も含んでいると考えられる．

シミュレーション結果もこの Putnam の観測に一致するので，進捗モデルは，並行作業間の影響の，ある一面を捉えていると考えられる．

図 6は 12 作業からなるより大規模なプロセスを想定し，各作業の開始条件を，上流作業の進捗の 100%, 70%, 20% の 3 条件とし，シミュレートしたときの累積工数の変化である．この場合においても，作業の開始条件を 70% から 20% にしても，期間はさほど短縮されず，工数は急激に増加する結果となっている．

5 おわりに

作業の並行化による開発期間と開発総工数に対する影響を予測するために，プロセスモデル，進捗モデルの 2 つのモデルを提案した．

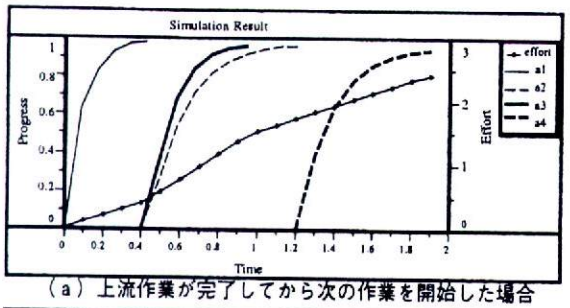
提案モデルでは，

- プロセスモデルの要素である作業の属性として，作業の開始時刻を与えることにより，並行作業による開発プロセスを記述することができる，
- 進捗モデルにおいて作業の進捗を計算する際に，その並行作業の属性をパラメータとすることで，並行作業間の影響をモデル化することができる．

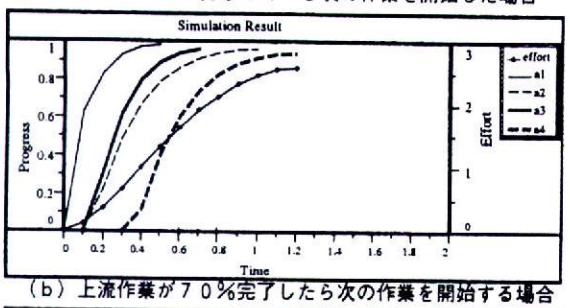
このように，並行作業による開発プロセスを記述することができ，並行作業間の影響をモデル化することができるため，各作業の開始時刻を様々に変化させることにより，作業の並行化による開発期間と開発総工数に対する影響を予測することが可能となった．

また，開発プロセスを想定しに適用した結果，開発期間をあるところより短縮しようとすると，開発総工数は急激に増加する，という Putnam の観測に一致することを確認した．

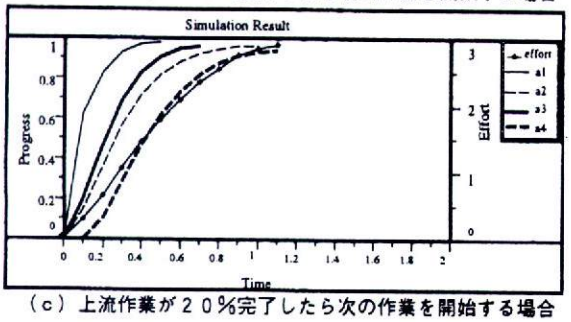
今後の課題としては，まず，進捗モデルを，実データに基づき検証，修正することが挙げられる．また，本モデルでは作業の動的な属性として進捗のみを扱って



（a）上流作業が完了してから次の作業を開始した場合



（b）上流作業が７０%完了したら次の作業を開始する場合



（c）上流作業が２０%完了したら次の作業を開始する場合

図 4: 例題プロセスのシミュレーション結果

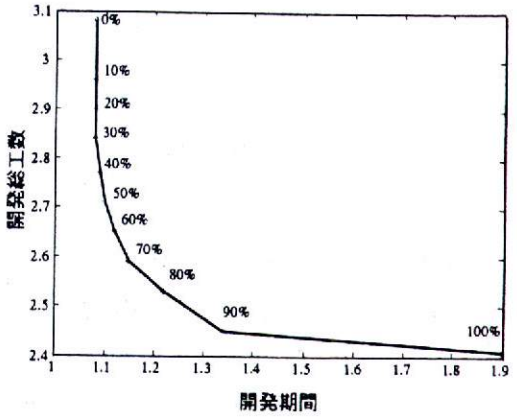


図 5: 開発期間と開発総工数の関係

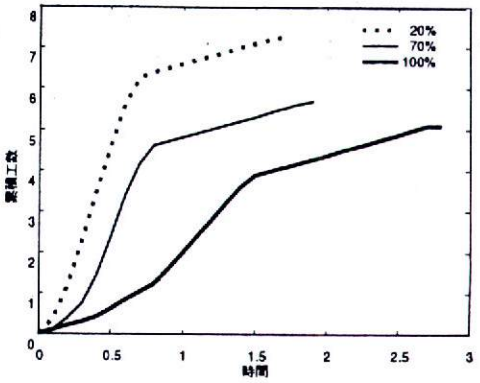


図 6: より大規模なプロセスのシミュレーション結果

いるが，より詳細なシミュレーションを行うために，品質なども属性とし，モデルを拡張することなどが挙げられる．

参考文献

[1] Albrecht A.J. and Gaffney J.E., Jr.: "Software function, source lines of code, and development effort prediction: A software science validation," IEEE Transactions on Software Engineering, Vol.SE-9, No.6, pp.639–648 (1983)

[2] Boehm B.W.: "Software engineering economics," IEEE Transactions on Software Engineering, Vol.SE-10, No.1, pp.4–21 (1984).

[3] Brooks F.P., Jr.: "The mythical man-month," Reading, MA, Addison-Wesley (1975).

[4] Curtis B., Krasner H. and Iscoe N.: "A field study of the software design process for large systems, " Communications of the ACM, Vol.31, No.11, pp.1268–1287 (1988).

[5] DeMarco T.: "Controlling software projects," Yourdon Inc. (1982), " ソフトウェア開発プロジェクト技法", 渡辺 純一訳, 近代科学社 (1987).

[6] 平山, 水野, 楠本, 菊野: "ソフトウェアプロジェクトの定量的管理のための階層的モデルの提案", 電子情報通信学会技術報告, FTS95-74, pp.89–96 (1995).

[7] Kellner M.I.: "Software process modeling support for management planning and control," Proceedings of 1st International Conference on Software Process, pp.8–28 (1991).

[8] 内藤，飯田，松本，鳥居: "役割別工数投入計画のための見積りモデルの提案", 情報処理学会研究報告, 96-SE-107, pp.1–8 (1996).

[9] Putnam L.H.: "A general empirical solution to the macro software sizing and estimating problem," IEEE Transactions on Software Engineering, Vol.4, No.4, pp.345–361 (1978).

[10] 高根 宏士: "ソフトウェア工程管理技法", ソフト・リサーチ・センター (1991).

[11] Tvedt J.D. and Collofello J.S.: "Evaluating the effectiveness of process improvement cycle time via system dynamics modeling," Proceedings of COMPSAC95, pp.318–325 (1995).

[12] 山田 茂, 高橋 宗雄: "ソフトウェアマネジメントモデル入門", 共立出版 (1993).