

性能予測機能を持つ自動 プログラミングシステム

古宮誠一

情報処理振興事業協会 技術センター

transformational softwareでは、要求仕様を満足するプログラムを完全自動で生成するだけでなく、自動生成されるプログラムの性能を事前に予測することが可能である。本論文では、transformational softwareにおいて、性能の事前予測が可能な理由と、自動生成されるプログラムの性能をどのように予測するかについて明らかにしている。

An Automatic Programming System with Facilities Estimating Performance of Target Program

Seiichi KOMIYA

Software Technology Center
Information-technology Promotion Agency, Japan (I.P.A.)

Shuwa Shibakoen 3-chome BLDG.
3-1-38, Shibakoen, Minatoku, Tokyo, 105 Japan.

In transformational software, it is possible that we can develop the tool which do not only full-automatically create the programs to satisfy users' requirements, but also estimate the performane of programs to be created. This paper describes why this tool can estimate the perfomane of programs to be created, and how this tool estimate it.

1. はじめに

transformational softwareとは、入力データを入力時のデータ形式から出力時のデータ形式に変換するようなソフトウェアである。従って、transformational softwareは、入出力データの構造が決まれば自ずからその処理アルゴリズムが決定する(= data-drivenな振舞いをする)という性質がある。transformational softwareに対する開発方法の理想は、ソフトウェア開発が下記のステップ①～④で行われ、条件⑤⑥を満足することである^{11), 14)}。

①ライフサイクルの早い段階に、プロトタイプングすることにより、与えられた要求仕様がユーザの意図どおりであること(= 仕様の正当性)を確認する。

ここで言うべきことは、データ構造を決定し、システムの振舞い(データの変換仕様)を決定することである。そして、その決定にユーザの意図を反映することである。このため、この過程はユーザの参加により試行錯誤で行われる。ここで特筆すべきことは、システムの振舞いの確認に先立ってデータ構造を決定するという点である。これはtransformational softwareの性質によるものであり、reactive software(= event-drivenな振舞いをするソフトウェア)では見られないことである。システムの振舞いの確認には、アニメーションによるプロトタイプング手法が用いられる¹⁴⁾。ここでは、データ項目間やプロセス間での制約が利用される。

また、システムが完全性(= 仕様に抜けがないこと)や一義性(= 曖昧さがなく一意)を満足することも①の過程で確認される。

②与えられた要求仕様を解析して、仕様の一貫性(= 要求仕様そのものに矛盾がないこと)や到達可能性などを満足することを確認する。

①によって、システムの振舞いに関するユーザの要求が確認された。しかし、プロトタイプングだけでは、要求仕様そのものに内在する矛盾や到達可能性などはチェックできない。これらをチェックするのがステップ②である。

③①と②で確認された要求仕様を満足するプログラムが得られる前に、そのプログラムの実行時における性能(レスポンスタイムやスループット)を計算し評価して、性能面でもユーザの了解を得る。

機能面でもユーザの意図どおりであることを確認し、要求仕様そのものにも問題がないことが判明したこの段階になると、要求仕様を満足するプログラムを完全自動で生成することが可能となる。しかし、プログラムを自動生成する前に、自動生成するプログラムの性能を計算し評価して、性能面でもユーザの了解を得る必要がある。このためのステップが③である。

なお、自動生成されるプログラムの性能予測が可能であることは、要求仕様を満足するプログラムを完全自動で生成できる^{4), 12)}ことに起因している。

また、PAPSで自動生成されるプログラムは、性能が最高となるように工夫されているので、プログラムの改良だけではこれ以上性能が上がらない。このため、試算された値よりも上の性能を要求された場合には、データベースのチューニングを勧め、それでも性能が上がらないときは、より高性能なハードウェアを推奨する形でコンサルテーションを行う。システム価格が折り合わなかった時には、試算した性能と現行のハードウェア構成でユーザに我慢して戴くことになる。

④機能と性能の両面でユーザの要求仕様を満足するプログラムを完全自動で生成する。

機能と性能の両面でユーザの意図どおりであることを確認したこの段階になって、初めてユーザの要求する仕様を満足するプログラムを完全自動で生成する。

⑤自動生成されるプログラムは品質が保証されており、ユー

ザはデバッグする必要がない。

ソフトウェアの生産性が最も高いのは自動プログラミングである。自動プログラミングで得られたプログラムを目の前にうざ高く積まれ、バグがあるからデバッグしてくれと言われたら、いくらプログラム生成率が高くても誰でも逃げ出すであろう。それ故、自動プログラミングによって得られるプログラムは、その品質を保証することにより、ユーザをデバッグ作業から解放しなければならない。

なお、自動生成されるプログラムの品質を保証できる¹²⁾のは、要求仕様を満足するプログラムを完全自動で生成できる^{4), 12)}ことに起因している。

⑥上記の過程①～④で得られた仕様は、上記の過程①～⑤で作成されたプログラムの保守やこれに類似したプログラムを作成する際に再利用できる。

本稿では、上記①～⑥の機能を実現する自動プログラミングシステムPAPSにおける、③の機能の実現方式について論述する。

コンピュータシステムの性能は、待ち行列理論(queueing theory)に基づいて計算するのが普通である。従って、自動プログラミングシステムPAPSで自動生成されるプログラムの、実行時における性能の予測も待ち行列理論に基づいて行われる。そこでは、性能予測のために対象システムを待ち行列のモデルとしてどのように捉えるか、そして、それはどのような理由によるものかを明らかにする。しかる後に、性能予測の計算をどのような手順で行うかを具体例を挙げて記述する。

2. 性能予測に用いる待ち行列のモデル

本章では、性能予測のために対象システムを待ち行列のモデルとしてどのように捉えるか、そして、それはどのような理由によるものかを明らかにする。

先ず、性能予測の対象としてオンライン・システムを仮定する。そして、議論を明確にするために、以下の議論では待ち行列のモデルをD. G. Kendallの記号

到着パターン/サービスパターン/窓口数

または、Kendallの記号の後ろに入力源の大きさを追加した到着パターン/サービスパターン/窓口数/入力源の大きさという記号を用いて表す。

(1) CPUでの待ち行列モデル

CPUでの待ち行列への入力源には、回線、周辺装置、プログラム、タスク、タイマー、コンソールなど多くのものがあり、実用では入力源を無限と仮定してよい。

窓口数は、CPUが1台であるから1となる。

CPUでの待ち行列における到着パターンおよびサービス・パターンは、共にランダム(ランダム到着のランダム・サービス)と見なしてよい。ここで、到着パターン(またはサービス・パターン)がランダムであるとは、次のことを仮定している。

①定常性

客の到着の(サービスを受ける)度合は、どの時間をとっても一定であること。

②残留効果なし(Marcov性)

客の到着の(サービスを受ける)度合は、過去の事象には左右されない。

③同時到着の希少性

同時到着(サービス)は希れにしか起こらないので無視できるということ。

これらの仮定は自然であろう。そして、これらの仮定の①と②から到着(サービス)間隔分布として指数分布が容易に導かれる。そして、仮定の③の下で指数分布を適用すれば、

到着（サービス）間隔分布としてポアソン分布が容易に導かれる。通常、指数分布またはポアソン分布をMで表わす。

従って、CPUでの待ち行列モデルは $M/M/1/\infty$ となる。このとき、CPUの利用率を ρ とすれば、CPUでの待ち時間は $\rho/(1-\rho)$ で求められる。

(2) チャンネルでの待ち行列モデル

DISCチャンネルでの待ちには、次の3つの要因がある。

- ① コマンド・チェーン開始のための待ち（SEEK以下のコマンドを発行するための待ち）
- ② 入出力開始のための待ち（SEEKが終わり、SEARCH動作に移るための待ち）
- ③ チャンネル再結合待ち（回転位置検出機構 RPSを用いる場合、所定の回転位置に達したとき、チャンネルが使用中ならばもう一回転待たされる。）

チャンネル再結合の待ちは1回転時間の単位で待ちが発生するから、チャンネル利用率を ρ とすれば、1回転待ち確率は ρ 、2回転待ち確率は ρ^2 、3回転待ち確率は ρ^3 、…、n回転待ち確率は ρ^n である。従って、チャンネル再結合待ち時間は、 $\rho + \rho^2 + \rho^3 + \dots + \rho^n + \dots = \rho/(1-\rho)$ 回転時間となる。

一方、チャンネル・サービス時間は次のとおりである。

・ DISCチャンネル

SEEK発行時	0
読み書き時（RPSなし）	回転待ち時間+転送時間
読み書き時（RPSあり）	ほぼ転送時間

・ MTチャンネル 起動時間+転送時間

入力源の数については次のように考えればよい。入出力要求は、（チャンネルから見て）当然のことながらデバイスの数以上には発行されない。また、マルチタスク・システムでは、各サブタスク内での入出力は通常シリアルに行われるので、サブタスクの個数以上には入出力要求は発行されない。従って、入力源の数は、 $\text{Min}\{\text{デバイス数}, \text{サブタスク数}\}$ となる。

DISCやMTは1台のチャンネルに複数のデバイスが接続されるから、窓口数は1である。

チャンネルでの待ち行列における到着パターンとサービス・パターンは共にランダムと見なしてよい。

従って、チャンネルでの待ち行列のモデルは、

$M/M/1/\text{Min}\{\text{デバイス数}, \text{サブタスク数}\}$ となる。

(3) DISCデバイスでの待ち行列

DISCの入出力動作のうち、SEEKとRPS使用時の回転待ちは、大半がチャンネル・フリーで行われる。そして、各ファイルのレコード/トラック/シリンダーを一様にアクセスするものと仮定する。

SEEK時間は、アームがシリンダー間とトラック間を移動する距離の平均だとすれば、アームの平均移動距離dは次のいずれかになる。即ち、

- ① シリンダー数nのファイル内を移動するだけのときには $n/3$ 、
- ② シリンダー数がそれぞれ n_1, n_2 の相隣接するファイル内を移動するときには $n_1/2 + n_2/2$ 、
- ③ シリンダー数がそれぞれ n_1, n_2 のファイル内の間にシリンダー数 n_3 のスペースがあり、このスペースを飛び超えて移動するときには $n_1/2 + n_2/2$ 、の3つのいずれかになる。

従って、各ファイルでのアクセスが各々どれ位の頻度で起こるのか、その確率を加味して（重みを付けて）加えたものが平均移動距離である。

回転待ち時間は、RPS使用の有無に拘らず1/2回転と見なしてよい。

マルチタスク・システムでは、各サブタスク内での入出力は通常シリアルに行われるので、サブタスクの個数以上には

入出力要求は発行されない。従って、入力源の数はサブタスク数と考えてよい。

DISCデバイスでの待ち行列における到着パターンとサービス・パターンは共にランダムと見なしてよい。

従って、DISCデバイスでの待ち行列モデルは

$M/M/1/\text{サブタスク数}$ となる。

なお、待ち行列のモデルとして、DISCデバイスと同じモデルになるものにはジャーナル・ルーチンの待ち行列、DISCホールド（排他制御機構）待ち行列、リユーザブル・プログラム待ち行列、オーバーレイ・リージョン待ち行列などがある。

(4) サブタスク割当て待ち行列

サブタスク割当て待ち行列への入力源は、回線、コンソール、タイマーなどであるが、オンライン・システムでは通常は入力回線のみを考えればよい。

窓口数はサブタスク数nであり、複数窓口となる。

サブタスク割当て待ち行列における到着パターンとサービス・パターンは共にランダムと見なしてよい。

従って、サブタスク割当て待ち行列のモデルは

$M/M/n/\text{入力回線数}$ となる。

複数窓口のシステムにおける待ち時間を求めるのは複雑な計算になるので、グラフや表（delay table）を利用するのが普通である。

DISCのホールド/フリー機構（排他制御）の制御単位には、デバイス単位、シリンダ単位、トラック単位、ブロック単位、レコード単位などの様々なレベルがあるが、どのレベルの制御になっているかがサブタスク割当て待ち行列のモデルを考える上で重要なファクターになる。例えば、デバイス単位の制御のときには、DISCホールドによる待ち合わせ時間をサブタスク割当て待ち時間に加算するが、DISCデバイス待ち合わせ時間は、平均1件処理待ち合わせ時間に加算しなくてよい。ところが、その他の単位で制御されているときには、DISCホールドによる待ち合わせ時間をサブタスク割当て待ち合わせ時間に加算しなくてもよいが、DISCデバイス待ち合わせ時間は、平均1件処理待ち合わせ時間に加算する必要がある、などである。

3. 性能予測のための例題

自動プログラミングシステムPAPSでは、要求仕様を与え終わった時点で、要求仕様を満足するプログラムを自動生成するのか、自動生成によって得られるプログラムの実行時における性能を予測するののどちらか一方を選択できる。ここで性能予測機能を選択すると、PAPSは、自動生成するプログラムに関する性能予測のための前提条件を入力するようにメニュー方式で要求してくる。これに対してユーザは、メニューに応える形で性能予測のための前提条件を入力しなければならない。ここでは、性能予測の処理手順を明らかにするために、このときユーザが入力する性能予測のための前提条件を要約したものを以下に示す。例えば、下記のような前提条件を持つ銀行オンライン・システムを考える。但し、ここに掲げた前提条件は、説明を簡単にするために実際のシステムよりもずうっと簡単になっている。

(1) オンライン処理の対象科目

普通預金、当座預金、定期預金、通知預金、納税準備預金、従業員預金、別段預金、定期積金、日計業務

(2) 使用するハードウェアと各々の性能単価

① CPU

平均命令実行時間	管理プログラム	2.3 μ S
	CUP	6.0 μ S
データ転送サービス	SELチャンネル	0.84 μ S/4B
	BYMPXチャンネル	(R) 5.25 μ S/B
		(W) 5.04 μ S/B

②DISC

SEEK時間	75.0mS
1回転の時間	25.0mS

③MT

起動時間	6.5mS
転送時間	60.0KB/S

④端末と回線

窓口装置	ハーフ・サイズ	96Byte
回線数	36回線 (1.5TCE/回線)	
通信速度	1200BPS	
信号方式	半リットル/レバション方式	

(3)使用するソフトウェアと各々の実行速度

①管理プログラムの

マクロ命令	CMGET	0.4mS
	CMPUT	0.4mS
	EXTSK	0.5mS
	EXCPW(DISC)	2.6mS
	EXCPW(MT)	1.3mS
	TTSK	0.3mS
	XITSK	0.5mS
	UPRB	0.4mS
	LPOB	8.9mS

②CUP

メインメモリ	1本
サブメモリ	3本
DISCホールド/フリー	デハイス単位
ホールド/リリース	1面/77タス

(4)トラフィック

平均月のピーク日、ピーク時のトラフィックを

12700件/時とする。(問題を簡単にするために、業務によらずトラフィックは同一とした。)

(5)対象業務の処理を図1のように入出力処理を主体にしたフローチャートで示す。

なお、PAPSでは、与えられた要求仕様のみから図1相当のフローチャートを自動的に出力する。但し、欄外の説明文などは出力されない。また、業務比率、サブタスクの割当て/解放のタイミング、ホールド/フリーの開始/終了のタイミング、ジャーナル・ルーチンの割当て/解放のタイミング、オーバーレイ・リージョンの割当て/解放のタイミングなどはフローチャートが出力された後にユーザが与える。

4. 性能予測の処理手順

自動プログラミングシステムPAPSが自動的に行う性能予測の処理手順を以下に順を追って記述する。

(1)図1のフローチャートから、表1のようなマクロフロー表#1を作成する。

(2)表1のマクロフロー表#1から、表2のようなマクロフロー表#2を作成する。

(3)表2のマクロフロー表#2から、表3のようなCPUと入出力の処理時間まとめ表を作成する。

ここで、無効ポーリングのためのCPU時間を求める(算出方法の説明省略)。

(4)図4のCPUと入出力の処理時間まとめ表から、各リソースの使用率、待ち時間、1件当たりの平均処理時間を求める。

①CPUの使用率、待ち率、待ち時間

$$\text{CPU使用率 } \rho_c = (108.86 \times 12700) \div (3600 \times 103) = 0.38$$

$$\text{CPU待ち率 } \omega_c = \rho_c \div (1 - \rho_c) = 0.62$$

$$\text{TotalのCPU待ち時間} = (\text{TotalのCPU占有時間}) \times \omega_c = 108.86 \times 0.62 = 67.5\text{mS}$$

②DISCデハイスの占有時間、MTデハイスの占有時間

$$\text{DISCデハイスの占有時間} = 462.3\text{mS}$$

$$\text{MTデハイスの占有時間} = 61.3\text{mS}$$

③DISCチャンネルの使用率、待ち率、待ち時間

$$\text{使用率 } \rho_D = (154.7 \times 12700) \div (3600 \times 103) = 0.55$$

M/M/3/1で $\rho_D = 0.55$ だからDelay Tableより、

$$\text{DISCデハイス待ち率 } \omega_D = 0.46$$

TotalでのDISCデハイス待ち時間

$$= (\text{DISCデハイス占有時間}) \times \omega_D = 71.2\text{mS}$$

④ジャーナル・ルーチン(JNL)の使用率、待ち率、待ち時間

$$\text{JNL内CPU使用時間} = 8.51\text{mS}$$

$$\text{待ちなしJNL占有時間} = (\text{JNL内CPU}) + (\text{JNL内MT})$$

$$= 8.51 + 61.3 = 69.8\text{mS}$$

$$\text{JNL内CPU待ち時間} = (\text{JNL内CPU使用時間}) \times \omega_c$$

$$= 8.51 \times 0.62 = 5.28\text{mS}$$

JNL占有時間

$$= (\text{待ちなしJNL占有時間}) + (\text{JNL内CPU待ち時間}) = 75.1\text{mS}$$

$$\text{JNL使用率 } \rho_J = (75.1 \times 1.03 \times 12700) \div (3600 \times 103)$$

$$= 0.27$$

M/M/3/1で $\rho_J = 0.27$ だからDelay Tableより、

$$\text{JNL待ち率 } \omega_J = 0.20$$

$$\text{JNL待ち時間} = (\text{JNL占有時間}) \times \omega_J = 57.1 \times 0.20$$

$$= 15.0\text{mS}$$

(注)MTチャンネルはJNLでしか使用していないので、

MTチャンネル待ちは発生しない。

⑤DISCホールド機構の占有時間、使用率、待ち時間

$$\text{ホールド内CPU使用時間} = 28.23\text{mS}$$

$$\text{ホールド内CPU待ち時間} = (\text{ホールド内CPU使用時間}) \times \omega_c$$

$$= 28.23 \times 0.62 = 17.5\text{mS}$$

$$\text{ホールド内DISCデハイス使用時間} = 80.1\text{mS}$$

$$\text{ホールド内DISCデハイス待ち時間}$$

$$= (\text{ホールド内DISCデハイス使用時間}) \times \omega_D = 36.8\text{mS}$$

$$\text{ホールド内IO時間} = (\text{DISCのIO時間}) + (\text{MTのIO時間})$$

$$= 183.7 + 46.3 = 230.0\text{mS}$$

(注)DISCのデハイス時間は、DISCのデハイス時間に吸収されるから不要

$$\text{ホールド内JNL待ち時間}$$

$$= (\text{JNL待ち時間}) \times (\text{ホールド内JNL回数/全体}) = 7.5\text{mS}$$

$$\text{DISCデハイス・ホールド占有時間}$$

$$= 28.2 + 17.5 + 36.8 + 230.0 + 7.5 = 320.0\text{mS}$$

$$\text{ホールド機構使用率 } \rho_H$$

$$= (320.0 \times 1.03 \times 12700) \div (3600 \times 103 \times 6) = 0.19$$

M/M/3/1で $\rho_H = 0.19$ だからDelay Tableより、

$$\text{ホールド待ち率 } \omega_H = 0.11$$

$$\text{ホールド待ち時間}$$

$$= (\text{DISCデハイス・ホールド占有時間}) \times \omega_H = 35.2\text{mS}$$

⑥サブタスクの占有時間、使用率、待ち時間

$$\text{77タス内CPU使用時間} = 4.31\text{mS}$$

$$\text{77タス内CPU待ち時間}$$

$$= (\text{77タス内CPU使用時間}) \times \omega_c = 2.5.6$$

$$\text{77タス内DISCデハイス使用時間} = 141.4\text{mS}$$

$$\text{77タス内DISCデハイス待ち時間}$$

$$= (\text{77タス内DISCデハイス使用時間}) \times \omega_D = 65.0\text{mS}$$

$$\text{77タス内IO時間} = (\text{DISCのIO時間}) + (\text{MTのIO時間})$$

$$= 374.0 + 46.3 = 420.3\text{mS}$$

(注)DISCのデハイス時間は、DISCのデハイス時間に吸収されるから不要

$$\text{77タス内JNL待ち時間}$$

$$= (\text{JNL待ち時間}) \times (\text{77タス内JNL回数/全体}) = 7.5\text{mS}$$

$$\text{77タス内ホールド待ち時間} = \text{ホールド待ち時間} = 35.2\text{mS}$$

$$\text{77タス占有時間} = 41.3 + 25.6 + 65.0 + 420.3 + 7.5 + 35.2$$

= 594.9ms

プログラムの利用率 ρ_s

= $(594.9 \times 1.03 \times 12700) \div (3600 \times 103 \times 3) = 0.72$

M/M/36/3で $\rho_s = 0.72$ だから Delay Table より、

プログラムの待ち率 $\omega_s = 0.44$

プログラムの待ち時間 = (プログラムの占有時間) $\times \omega_s = 261.7\text{ms}$

⑦ 1件処理平均時間

待ち合わせを含まない 1件処理平均時間

= (TotalのCPU時間) + (DISCの占有時間)

+ (MTの占有時間)

(注) DISCの待ち時間は、DISCの占有時間に吸収されるから不要

= $108.86 + 462.3 + 61.3 = 632.5\text{ms}$

待ち合わせを含む 1件処理平均時間

= (待ちを含まない 1件処理平均時間)

+ (TotalでのCPU待ち時間) + (TotalでのDISC待ち時間)

+ (JNL待ち時間) + (プログラムの待ち時間)

= $632.5 + 67.5 + 71.2 + 15.0 + 261.7 = 1047.9\text{ms}$

5. おわりに

上記のような作表や計算が自動的に行える理由は、

① 与えられた要求仕様のみから、求めるプログラムを完全自動で生成できること^{(4), (12)}。

② 自動プログラミングのために与える要求仕様 (特に、プログラムの制御構造を決定するために与える部分) が主としてメニュー・ドリブンである^{(4), (12)}こと。

の2つに起因している。即ち、要求仕様がメニューで与えられる部分については、この2つの事実によって、要求定義におけるメニューの選択肢に対応して生成されるプログラムが一意的に決定する。言い換えれば、生成されるプログラムの実行ステップ数 (ダイナミック・ステップ数) などがメニューの選択肢に対応して、データベースの形で予め用意できるからである。

要求仕様がメニュー選択だけで与えられたのでは、メニューにないプログラムは生成できないことになり、ユーザのあらゆる要求を受け付けられなくなってしまう。ところが、よく調査してみると、オンライン処理用とオフライン処理用の各々に13種類ずつ程度の骨組み部品を用意すれば、ユーザの要求するすべてのプログラムの制御構造を実現できることが判明している⁽¹³⁾。従って、ユーザの要求するすべてのプログラムを実現可能にするには、メニュー選択だけで得られる筈のプログラムにユーザ個別の機能を追加できるように機構を用意すればよい。具体的には、メニュー選択によりプログラムの制御構造が決定した後に、業務プログラム固有の機能をユーザ主導の方式 (仕様記述言語による記述) で仕様を与え、これに対応するプログラム・コードを生成する機構を用意すればよい。

業務プログラム固有の機能は、入力データのチェックをどのように行うかということと、入力レコードから出力レコードをどのように作成するか、の2つに集約される。前者の処理仕様はメニュー方式がフィットする。従って、この場合には、生成されるプログラムの実行ステップ数は予測可能である。

一方、後者は次のように与えられる。それは、入力レコードと出力レコードとの対応において、出力レコードの或る項目の内容の求め方が分からないときに、PAPSはその項目の内容の求め方を尋ねてくる。このとき、ユーザの応え方に次の2つの場合がある。

・ その項目の内容が、現在メモリ中にあるレコードの項目間の関係のみから求める (デシジョン・テーブルを使用しない)。

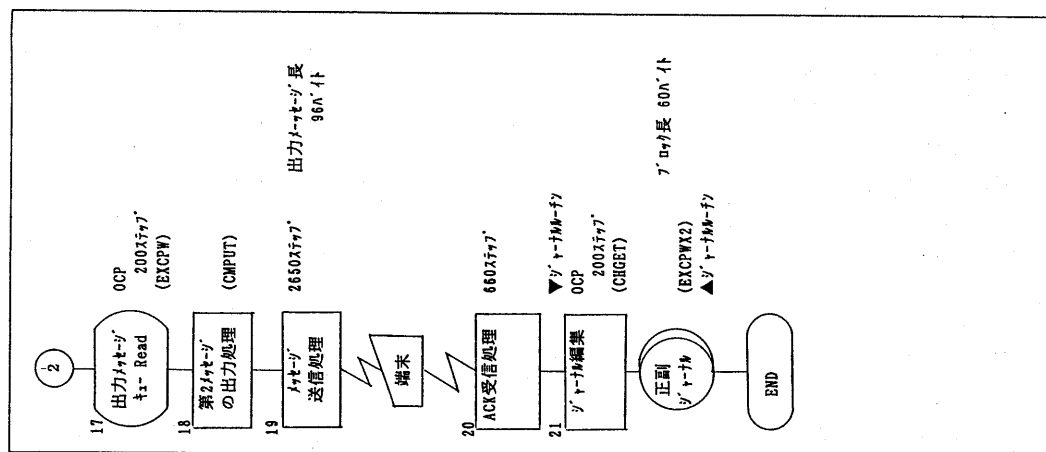
・ 既に入力されているレコードの項目の内容と、これから入力されるレコードの項目の内容との関係から求める (デシジョン・テーブルを使用する)。

その求め方は、いずれの場合も、PAPSの仕様記述言語を使って、項目間の演算を指定することになるが、前者は演算にレコード入力に伴わないのに対して、後者は演算にレコード入力に伴うという違いがある。従って、実用的には後者におけるレコード入力の実行ステップ数のみを算出できれば充分であるが、この部分の仕様記述が、生成されるプログラム・コードの行数が固定的なものの組合せに分解できるので、両者とも実行ステップ数の予測は可能である。

以上で、PAPSでは、自動生成されるプログラムの性能予測が可能であることが示された。

【参考文献】

- [1] 古宮誠一: 「プログラム・シンセシス法の分析と実用化への方向付け」, 第4回技術発表会論文集, 情報処理振興事業協会, pp. 71-85 (1985).
- [2] 古宮誠一: 「部品合成によるプログラム自動合成システム ~ 部品表現のモデルとその意味記述について」, 第5回技術発表会論文集, 情報処理振興事業協会, pp. 151-154 (1986).
- [3] 古宮誠一: 「知識ベースを利用したプログラム自動合成システム」 第5回技術発表会論文集, 情報処理振興事業協会, pp. 17-25 (1986).
- [4] 古宮誠一: 「部品合成によるプログラム自動合成システム PAPS ~ 知識工学的アプローチを用いたその実現方式」, 情報処理学会研究報告, 87-SF-21, Vol. 87, No. 40, pp. 5-12 (1987).
- [5] 古宮誠一, 原田 実: 解説「部品合成による自動プログラミング」, 情報処理, Vol. 28, No. 10, pp. 1329-1345 (1987).
- [6] 古宮誠一: 「部品合成によるプログラム自動合成システム PAPS ~ 部品追加に強い部品の管理・検索方式」, 第35回全国大会講演論文集, pp. 1062-1062 (1987).
- [7] 古宮誠一: 「ソフトウェアの再利用における類似部品の扱い方について」, 電子情報通信学会春季全国大会講演論文集, D-374 (374).
- [8] 古宮誠一: 「類似性の概念と類推の正しい適用方法について ~ 部品合成による自動プログラミングへの応用を考える」, ソフトウェア・シンポジウム'88, pp. 69-78 (1988).
- [9] 古宮誠一: 「部品合成による自動プログラミング・システムの実現方式について」, 情報処理学会研究報告, 88-SF-24 (1988).
- [10] 部品合成による自動プログラミング・システム PAPS ~ プログラムの知識とその使い方について」, 電子情報通信学会, 技術研究報告, A188-34, pp. 19-26 (Sept., 1988).
- [11] 古宮誠一: 「プロトタイピング技術の分析と実用化への方向付け」, 第7回技術発表会論文集, 情報処理振興事業協会, pp. 37-46 (1988).
- [12] 古宮誠一: 「部品合成による自動プログラミングシステムで生成されるプログラムの品質保証方法」 電子情報通信学会, 技術研究報告 SS89-7 (July, 1989).
- [13] 古宮誠一: 「ソフトウェア再利用支援ツールの構築に必要な部品とその完備性の保証方法」, 日本ソフトウェア科学会全国大会論文集, pp. 377-380 (1989).
- [14] 古宮誠一: 「アニメーションによるプロトタイピング ~ 制約を用いたその方式 ~」, 情報処理学会研究報告, SE-71-16, pp. 121-130 (1990).



— 80 —

表1 マクロフロー表 #1

表 1 マクロフロー表 #1									
項 番	処 理 項 目	比 率	管理プログラム		EOスチールタイム		CUP		CPU 時間合計
			ステップ	時 間	WPA' (分)	SELV' (分)	時 間	ステップ	
1	メモリー受信	1.00	2420	5.37	70		0.37		5.34
2	入力処理	1.00	720	1.66				600	3.40
3	入出力処理	1.00						500	3.00
4	入出力処理	0.24	12760	29.35		2400	0.50		7.16
5	マススタRead	1.00	2220	5.11		3600	0.76		5.87
6	サブRead	0.23	1110	2.55		180	0.04		0.75
7	ファイル更新	1.00						2300	12.40
8	ジャーナル処理	1.00	1140	2.62		2000	0.24	200	4.24
9	サブWrite	0.23	1110	2.55		180	0.04		0.75
10	マススタWrite	1.00	1110	2.55		1300	0.27		2.82
11	出力要求	1.00							
12	キューWrite	1.00	1240	2.80		96	0.02		2.92
13	タスク終了処理	1.00	400	0.92				500	3.92
14	メモリー送信	1.00	2640	6.10	96		0.50		6.60
15	ACB受信	1.00	660	1.52					1.52
16	入力処理	1.00	180	0.41				50	0.71
17	キューRead	1.00	1110	2.55				200	3.15
18	出力処理	1.00	180	0.41					0.41
19	メモリー送信	1.00	2430	6.10	96		0.50		6.60
20	ACB受信	1.00	660	1.52					1.52
21	ジャーナル処理	1.00	1230	3.04		120	0.03	200	4.27
22									
入出力処理時間									
No. ()									
入出力処理時間									
No. ()									
入出力処理時間									
No. ()									
ジャーナル処理時間									
No. ()									
内合計									
No. ()			88 + 821 4.51						
ディスクサブシステム									
No. ()			85~210 28.33						
内合計									
No. ()			22~212 41.31						
サブシステム内合計									
No. ()			81.81						
総 計									

表3 CPUと入出力の処理時間まとめ表

項 番	項 目	処理 件数 比率 (%)	C P U								D I S C チ ャ ネ ル						DISCデバイス		MTチャネル	
			TOTAL		サブタスク		ホールド		リターン		TOTAL		サブタスク		ホールド		TOTAL		TOTAL	
			ms/件	比率 × ms/件	ms/件	比率 × ms/件	ms/件	比率 × ms/件	ms/件	比率 × ms/件	ms/件	比率 × ms/件	ms/件	比率 × ms/件	ms/件	比率 × ms/件	ms/件	比率 × ms/件	ms/件	比率 × ms/件
1	預 金	103.00	81.81	84.26	41.31	42.55	28.23	29.08	8.51	8.77	154.7	159.4	141.4	145.7	80.1	82.6	482.3	476.2	81.3	83.1
2	無効レター	100.00	24.60	24.60																
3																				
4																				
5																				
6																				
7																				
8																				
9																				
10																				
	センター処理 平均占有時間	100.00		108.86		42.55		29.08		8.77		159.4		145.7		82.6		476.2		83.1