コンピュータ・システム性能評価シミュレータ PACSS*

三上　徹**　久保　秀士**　高橋　勲**
有福　義範**　北浦　隆***

Abstract

This paper presents a model of computer systems, called CS model, and proposes a simulator based on this model for evaluating performance of computer systems, named PACSS (Program And Computer System Simulator).

It is shown that the action of a computer system can be modelled by mutual actions of message transactions, process transactions, activators and resources in the CS model. Also, user behaviours, user program characteristics and the operating system function of the system can respectively be represented by job generations, job processes and an activity block network of the CS model. It is presented that the job generations and job processes are represented in Markov chains of command sequences and those of SVC sequences respectively. The activity block network can be described with twentyfive standard activity blocks provided by PACSS.

It is also shown that PACSS can produce detailed and flexible statistics on cpu utilizations, other resource utilizations, response times, queues and other variables of the models. In addition, the structure and simulation mechanism of PACSS are briefly described. Finally, discussions on advantageous features of PACSS are given in comparison with functions of GPSS.

1. まえがき

複雑・大規模なコンピュータ利用システムの開発・設計時に遭遇する最も重要な問題の1つに，システムの性能解析・予測・評価の問題がある．このシステム評価の問題は，今後コンピュータの高度利用に対する要請が高まるにつれて，システムに要求される期待性能がさらに苛酷になるであろうことを考えると，ますます重要になってくるものと思われる．システム評価に対する的確な手法を欠いては，満足すべき性能を持ったシステムの開発が困難になるばかりか，近い将来においては，システムの設計そのものすら不可能になるのではないかということが予想される．このような考えに基づいて，著者らはシステムの開発・設計時点にそれに反映できるだけの充分な評価データを与えうる評価手法を提供することを目的として，コンピュータ・システム性能評価用の専用シミュレータを開発した．以下に，その概要を報告する．

コンピュータ利用システムの性能を表わす指標は，その利用者から見た場合

（a）システム応答の速さ

であり，システムを管理する側から見た場合には

（b）システムの生産性（各種資源の使用効率）のよさ

である．また，これらの性能指標に影響を及ぼすシステムのおもな要因は

（1）システムに印加される利用者ジョブの量，および性質（環境特性）

（2）システムの金物構成（システム構成）

（3）利用者ジョブに対する各種資源の配分の方式（システム制御方式）

である．したがって，コンピュータ・システムの性能評価とは，これら3個の要因変量の上記(a)，(b)の

* A Simulator for Fvaluating Performance of Computer Systems PACSS, by Toru Mikami, Hidehito Kubo, Isao Takahashi, Yoshinori Arifuku and Takashi Kitaura (Nippon Elecatric Co. Ltd.)

** 日本電気株式会社　中央研究所
*** 日本電気株式会社　データ通信システム事業部

性能指標に及ぼす影響を解析し，これに基づいてシステムの要求仕様に対する充足性を評価することである．実際には，これら最終的な性能指標のみでなく，これらに関連する多くの媒介変量，たとえば，各種資源における待行列の長さ，待時間などに関する解析をも必要とすることがしばしば生じる．

一般に，コンピュータ・システムの評価方法としては，大別して

（1） 理論解析（待行列理論など）による評価法
（2） 実物テストによる評価法
（3） シミュレーションによる評価法

の3方法が考えられる．理論解析による方法は，ある与えられたモデルに対しては，1点疑う余地のない正確な結果を与えてくれることに特長はあるが，それの取り扱える対象はきわめて単純なモデルに限定されるため，設計初期におけるごく大まかな性能計算か部分的モデルによる局所的解析ぐらいにしか有効でない．設計がある程度進んだ時点以後では，この方法は（現レベルの理論手法に頼る限り）ほとんど無力である．

実物テストにおける方法は，そのものズバリの結果を与えてくれるのであるから，これに越したことはないが，設計に先立ってパイロット・モデルにしろ実物に近いものを作製し，これをテストするための使用環境を整備することは，並大抵のことではないし，その上かなりのコスト高につく．したがって，この方法をシステム作製の事前評価に用いることは一般的ではない．これらに比べて，シミュレーションによる方法は，その結果の精度と信頼性の評価について，若干の問題が残されているにせよ，モデル化に幅広い柔軟性を持っていること，かなり詳細な解析が比較的低いコストで行なえることなどの利点により開発・設計時点の性能評価には幅広い領域にわたって有効であると思われる．とくに，システム総合処理能力の評価，動的変動の解析，OS の制御アルゴリズムの評価，システム隘路の発見などには，きわめて有力な方法である．

つぎに，シミュレーションによって性能評価を行なう場合，われわれはシミュレーション用具として

（1） 汎用言語（たとえば，PL/1）
（2） 汎用シミュレータ（たとえば，GPSS, SIMSCRIPT）
（3） 専用シミュレータ

のいずれを用いるのが得策かという問題がある．一般に，シミュレーション解析が効果的に行なえるためには，それに用いられる用具（シミュレータ）としてはつぎのような条件を備えていることが望ましい．

（1） 対象システムのモデル化が簡単で，モデル記述が容易であること
（2） モデル化に柔軟性があること
（3） モデルのデバッグが簡単に短時間で行なえること
（4） いわゆる走行時間比，すなわち

$$\frac{\text{モデルのシミュレーション実行に要した実際の時間}}{\text{シミュレーション実行中にモデル上で経過した時間}}$$

ができるだけ低いこと
（5） 性能評価のための統計諸量が必要に応じて柔軟に，かつ容易に得られること

モデル化の柔軟性と走行時間比の点においては，一般に汎用言語あるいは汎用言語に近いある種の汎用シミュレータ（たとえば，SIMSCRIPT のようなもの）の方が有利であるが，（1）と（5）に関しては専用シミュレータの方がはるかに有利となりうる．デバッグの容易性は主として対象とするシステムの複雑さと，デバッギング機能の大小に依存するものであるからこれらの言語間の優劣を一律に論ずるわけにはいかない．一方，シミュレーション解析中にはモデル変更，パラメータ再設定の要求がしばしば発生することを考えると，シミュレータが備えるべき条件としては，Huesmann らが指摘したように，（1）の条件が最も重要である[1]．また，コンピュータ・システムの性能評価においては，様々な特性を示すシステムを種々の角度から解析する必要があり，このためには目的にかなった種々の統計量が豊富に，かつ簡単に得られることが望ましい．これら2つの条件に対する充足度の低いシミュレーション用具を用いたために，シミュレーション実施中に発生する余分の労力と面倒は莫大なもので，これは専用シミュレータの開発工数を（もし，それがうまく設計されたときに）充分に上回るものである．

この論文で紹介するコンピュータ・システム性能評価のための専用シミュレータ PACSS (Program And Computer System Simulator) は，このような考えに基づいて開発されたものである．PACSS の用途としてのねらいは，設計中期から後期におけるシステムの総合処理能力の評価，OS 制御機能の評価，詳細なシステム動特性の解析にあり，また，その評価対象の領域としては，一応タイムシェアリング・システムに重点が置かれているが，その他のシステム，たとえばリモート・バッチ処理システム，リアルタイム・システム，

ページ方式，多重処理機構などについても充分考慮がはらわれており，適用可能である．

この種のシミュレータとしては，IBM の開発した CSS[2]，CEIR の開発した SSS[3] があるが，前者はその適用範囲が IBM/360 系統の機械に限定されている．

2. モデル化の基本概念

われわれは，つぎのような4個の働きを含む閉じたシステムのモデルをコンピュータ・システムのモデル (CS モデル) と呼ぶ．

(1) 端末にいてコンピュータと会話をしている利用者

(2) 端末にいる利用者と相互作用を行ないながら，コンピュータ内部で与えられた仕事を実行している利用者プロセス

(3) これらの仕事の遂行に必要なすべてのサービス資源を管理し，これらの動作を制御しているオペレーティング・システム

(4) 上記のすべての働きを物理的に実現しているハードウェア・システム

一般に，システム・モデルを実体的見地からながめるとき，これは実体要素のモデルのある集合であると考えられるし，また，これを機能的観点からながめるとき，これは機能要素のモデルのある関係として与えられる総合機能モデルとみなすことができる．そこで，われわれは CS モデルを定義するために，以下のような4個の実体概念と3個の機能概念を導入する．

(i) プロセス・トランザクション：プロセッサーによって処理される利用者ジョブ・プロセスを表わす抽象実体

(ii) メッセージ・トランザクション：処理の指令やデータなど各種の情報を伝達するメッセージを表わす実体

シミュレーションにおいて，メッセージ・トランザクションは，たとえば，端末からのコマンド，コンピューター端末間の入出力データのあるかたまり，利用者プロセスからの I/O 要求，その他のスーパーバイザー・コールなどを表わすことができる．

(iii) アクチベータ：中央処理装置の制御の所在点を示す実体

各プロセス・トランザクション，メッセージ・トランザクションは，いずれも2つの状態，活動状態，非活動状態を持ち，常にいずれか一方の状態にあり，これら状態間の遷移はアクチベータにより制御される．そして，これらは活動状態にあるときにのみ流れ，素子としての働きを示すことができる．アクチベータはこれらトランザクションの状態を制御することにより，各利用者プロセス-OS 間，あるいは OS 上における中央処理装置の制御点の推移を流れ素子のある流れとして表現している．

(iv) リソース：資源をシミュレートする実体で，プロセス・トランザクション，あるいはメッセージ・トランザクションにより使用される

リソースはさらにつぎの3種類のものに分類される．

(i) ファシリティー：同時にはただ1個のトランザクションのみによって使用されるリソース

(ii) ストーレジ：多重の容量を持ち，同時に1個以上のトランザクションによって使用できるリソース

(iii) 周辺リソース：アクチベータの動作とは並列にメッセージ・トランザクションを処理するリソース

たとえば，ファシリティーによっては，主記憶装置の各区画，I/O バッファーの1個1個，順次再使用可能ルーチンなどが，ストーレジによってはバッファ・プール，主記憶装置のページの集合などが，また，周辺リソースによっては磁気テープ，ディスク，ドラム，ライン・プリンタなどが，それぞれシミュレート可能である．

つぎに，機能概念としては，以下のようなものを導入する．

(i) ジョブ・プロセス：t' と e の対 (t', e) の有限列

$$(t', e)_1, (t', e)_2, \cdots, (t', e)_n$$

ただし，e はメッセージ・トランザクションの発生という事象を表わし，t' はある e が起きてから，つぎの e が起きるまでにプロセス上で経過する純所要時間（いわゆるプロセス・タイム）を表わしている．すなわち，この間に割り込まれた時間などは t' に算入されない．

(ii) ジョブ発生：t と e の対 (t, e) の有限列

$$(t, e)_1, (t, e)_2, \cdots, (t, e)_m$$

ただし，e はメッセージ・トランザクションの発生という事象を表わし，t はある e で発生したトランザクションがその仕事を終えて消滅した後，そのつぎのトランザクションが発生する（すなわち，つぎの e）までの時間を表わしている．

ジョブ・プロセス，ジョブ発生は，それぞれコンピュータ内での利用者プロセスの働き，端末での利用者の動作をある時間ごとに発生する処理要求呼の列で表現したものにほかならない．

(Ⅲ)アクティビティ：アクティビティ要素のある関係づけられた集合．ただし，アクティビティ要素とはメッセージ・トランザクション，プロセス・トランザクション，アクチベータが遂行する機能の分解された単位要素．

アクティビティ要素，アクティビティは，それぞれアクティビティ・ブロックおよびアクティビティ・ブロック網で表現される．

アクティビティ・ブロック：アクティビティ要素の表現．これによって表現された機能は，そこをトランザクション，アクチベータが通過し，そこに示された機能を遂行することによって実現される．

アクティビティ・ブロック網：種々のアクティビティ・ブロックをある関係に従って結びつけたネットワーク．トランザクション，アクチベータはこのアクティビティ・ブロック網内を流れ，それによって規定された機能を遂行する．

具体的には，アクティビティ・ブロックは，オペレーティング・システムの管理・制御機能を組み立てている種々の基本機能，たとえば，待行列からの要素の取り出し，待行列順序の再配列，待行列内容の探索，保留された割込原因の探索等々をシミュレートすることを主目的としている．したがって，これらのブロックを組み合わせて種々の機能を持ったブロック網を作ることにより，オペレーティング・システムの任意の制御機能をシミュレートすることができる．

さて，これまでに導入した基本概念を用いて，CS モデルを形式的につぎのように定義しよう．

(CS モデル)

CS モデルとは 7-tuple

$(A, T_m, T_p, R; P, G, N)$

である．ただし，A, T_m, T_p, R はそれぞれアクチベータ，メッセージ・トランザクション，プロセス・トランザクション，リソースの集合であり，また P, G, N はそれぞれジョブ・プロセスの集合，ジョブ発生の集合，アクティビティ・ブロック網を表わしている．

P, G, N は A, T_m, T_p, R のそれぞれの要素(実体要素）の実行すべき働きおよびそれらの相互関係を定義しており，実システムに対しては，それぞれつぎのような機能に対応している．

P：利用者プロセスの振舞

G：端末における利用者の振舞

N：オペレーティング・システムの管理・制御機能

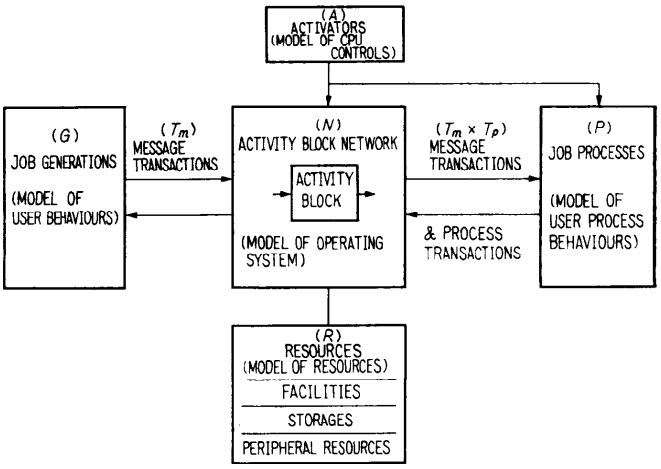

Fig. 1 The conceptual block diagram of CS model

モデル上，PとNとはメッセージ・トランザクション，プロセス・トランザクションの相互授受により，お互いに関係づけられており，またGとNはメッセージ・トランザクションの相互授受によりお互いに関係づけられている．メッセージ・トランザクションとプロセス・トランザクションは，N上をそれらが活動状態にあるときに流れ，N上の各所で規定された機能を遂行する．これらの機能動作の一部として，各種リソースの確保・解除という動作が行なわれる．

CS モデルのこのような動作過程を通じて，コンピュータ・システムの動作がシミュレートされる．CS モデルの各機能の相互関係の概念図を **Fig. 1** に示す．

3. ジョブ・プロセスとジョブ発生

前節で簡単に定義したジョブ・プロセスとジョブ発生は，CS モデルにおいて重要な役割を果たすので，これらの構造と機能について，さらに詳細な定義を以下に述べよう．

3.1 ジョブ・プロセス

ジョブ・プロセスはコンピュータ内における利用者プロセスの振舞をシミュレートするものであるが，この利用者のプロセスの振舞は，以下に示す 3 つのパラ

メータによって特長づけることができる (**Fig. 2** 参照).

(1) スーパーバイザ・コール (SVC) の列 (Fig. 2 の①)

(2) 相隣接する SVC 間の時間間隔の列 (利用者プロセスの初めと最初の SVC との間の時間間隔をも含む) (Fig. 2 の②)

(3) プロセス終了までに必要な全所要時間, あるいは総 SVC 個数 (Fig. 2 の③)

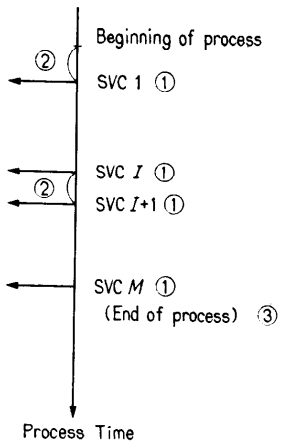

Fig. 2 The model of user job process

先に定義したジョブ・プロセスにおいて, e の列が SVC 発生の列に, t' の列が SVC 間隔の列に, また列の長さを示す n が全所要時間, あるいは総 SVC 個数にそれぞれ相当している.

つぎに, CS モデルにおいては, SVC の列はさらにつぎの 3 つの形の列の組合せによって構成されている (**Fig. 3** 参照).

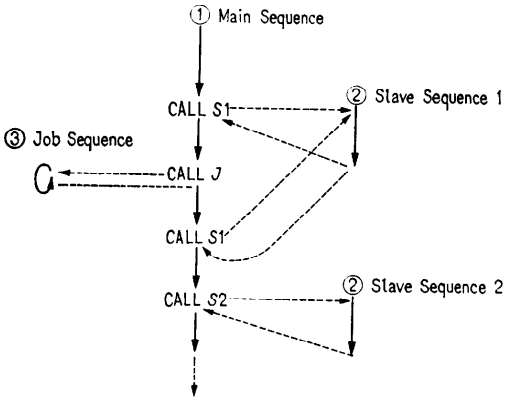

Fig. 3 The Structure of sequence of SVC's

(1) 主系列 (Fig. 3 の①)

(2) 従系列 (Fig. 3 の②)

(3) ジョブ系列 (Fig. 3 の③)

主系列が, 文字どおり, SVC 列の主列を成し, 従系列とジョブ系列は主系列内にそう入された CALL ステートメントにより SVC 列内に繰り込まれて SVC 列の一環を形造るようになっている. さらに, 主系列と従系列はどちらも, それぞれ定義された複数個の部分 SVC 列の 0 重あるいは 1 重マルコフ列によって表現される. また, ジョブ系列は SVC そのものの 0 重マルコフ列によって表わされる.

従系列は, たとえば, 繰り返して使われるサブルーチンのようなプログラムの特性を, またジョブ系列はその内部特性を詳細には規定し得ないようなプログラム (利用者ジョブのプログラムは大低これに当る) の大まかな特性を表わすために用いられる. 幾つかの定まった典型的 SVC 発生パターンの組合せによって構成されるようなプログラムの特性は, これらの典型的パターンを部分 SVC 列として記述することにより, 主系列もしくは従系列により簡単に表現することができる.

SVC 発生の時間間隔, およびプロセスの総所要時間, あるいは総 SVC 発生個数は, それぞれ定数としても, あるいは確率変数としても定義することができる.

3.2 ジョブ発生

ジョブ発生は端末における利用者の振舞をシミュレートするものであるが, この利用者の振舞は会話 (ある意味のあるまとまった会話, たとえば, 会話受付けから BYE at……まで) 動作の列として, さらに, 各会話動作はコマンドごとの会話動作の列として構成されると考えることができる. 各コマンドごとの会話動作は, さらに, その中に何回かのコンピュータとの相互やりとりを含んでいる場合もある. これらのことから, 端末における利用者の特性は, つぎに述べる 3 つのパラメータで特徴づけられると考えることができる (**Fig 4** 参照).

(1) 相隣接する会話間の時間間隔 (**Fig. 4** の①)

(2) 各会話におけるコマンドの列 (**Fig. 4** の②)

(3) あるコマンドの処理が終了してから, つぎのコマンドが発生するまでの時間間隔とコマンド内におけるある相互やりとり (interaction) の出力発生からつぎの相互やりとりの入力発生までの時間間隔との列 (Fig. 4 の③).

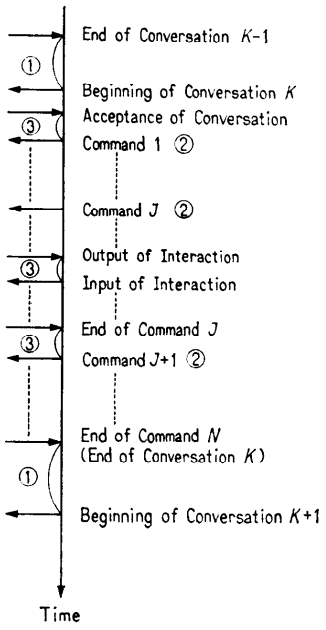

Fig. 4 The model of user job generation

先に定義したジョブ発生において，e の列が会話開始，コマンド発生および相互やりとりの入力発生の列に，t の列が上に述べた(1)および(3)の時間間隔の列にそれぞれ相当している．

コマンド列は，それぞれ定義された複数個の部分コマンド列，あるいは複数個の単独コマンドの0重あるいは1重マルコフ列によって表わされる．相互やりとりの入力の列は，ジョブ発生においては陽には定義されない．それは，ジョブ・プロセスでの相互やりとりを必要とする SVC の発生に応じて定まるものである．また，上記(1)，(3)で述べた時間間隔をさらに厳密に定義すると，これはコンピュータから端末への出力が開始した時点から，端末からコンピュータへの入力が終了する時点までの時間間隔である．したがって，この時間には，データあるいは指令の出力時間，思考時間およびデータあるいは指令の入力時間が含まれる．これらの時間はいずれも定数として，あるいは確率変数として与えられうる．

4. アクティビティ・ブロック

アクティビティ・ブロックは PACSS の最も重要な部分を占めており，CS モデルのアクティビティ要素はすべてアクティビティ・ブロックで表現され，シミュレートされる．アクティビティ・ブロックの主目的は，オペレーティング・システムの機能要素を表現することにあるが，PACSS では形式的に，ジョブ・プロセス，ジョブ発生もすべてアクティビティ要素の一種であるとみなされ，ある種のアクティビティ・ブロックで表現されるようになっている．もちろん，ジョブ・プロセスやジョブ発生を幾つかのアクティビティ・ブロックの組合せから成るネットワークで表現することも可能である．

PACSSには25種類の標準ブロックが用意されており，それらのすべてのブロックは（JOG ブロックだけを除いて）それぞれ個有のモード，通過時間，クラスを選択できるようになっている．ここでブロックのモードとは，そのブロックの機能実行の優先レベルを決めるものである．仮りに，あるブロックの機能実行中に割込みイベントが発生したとする．もし，割込みイベントのモードの方がそのブロックのモードより高ければ，そのブロックの実行は中断され，そうでなければ割込みイベントは保留されてブロックの実行が続行される．通過時間とは，トランザクションがブロックを通過するときに消費する時間のことである．また，クラスとはブロックの通過時間の累計を算出するときに用いられる類別化指標のことであって，ある測定期間中にトランザクションが通過したすべてのブロックの通過時間がクラスごとに分類されて集計される．シミュレーション・モデルの全ブロックを適当にクラス分けすることにより，たとえば，測定期間中の中央処理装置の純使用時間，オーバーヘッド時間，アイドル時間などを知ることができる．

標準アクティビティ・ブロックのおのおのについて以下簡単に説明しよう．

（I） 利用者および利用者ジョブの特性をシミュレートするブロック

（1） JOG ブロック

このブロックはジョブ発生機能のすべてを遂行する．このブロックに付随したテーブルには，つぎのようなパラメータが記入される．

- 会話間隔の確率分布
- コマンドおよび部分コマンド列の定義
- コマンド発生，あるいは部分コマンド列発生の確率，あるいは遷移確率
- 端末入出力装置ごとの入出力時間，および思考時間の固定値，あるいは確率分布

など．

（2） JOB ブロック

このブロックはジョブ・プロセス機能のすべてを遂行する．このブロックに付随したテーブルには，つぎのようなパラメータが記入される．

・主系列および従系列における部分 SVC 列の定義
・ジョブ系列における SVC の定義
・主系列および従系列における部分 SVC 列発生の確率，あるいは遷移確率
・ジョブ系列における SVC 発生の確率
・SVC 発生間隔の固定値，あるいは確率分布
・プロセス総所要時間，あるいは総 SVC 発生個数の固定値，あるいは確率分布

など．

(Ⅱ) トランザクションの取扱いに関するブロック

(3) TRG ブロック

このブロックはトランザクションを決められた時間間隔で発生する．

(4) TT ブロック

このブロックにトランザクションがはいると，そのトランザクションは消滅する．

(5) TD ブロック

このブロックにトランザクションがはいると，新しい別のトランザクションが，そのトランザクションから派生される．

(Ⅲ) リソースを管理するブロック

(6) HR ブロック

このブロックは周辺リソースの使用を開始させる．周辺リソースの使用が終わると，自動的に割込みイベントが発生する．

(7) SR ブロック

(8) RR ブロック

これらのブロックはファシリティを確保し，解除する．

(9) CRS ブロック

このブロックはストーレジの状態を変更する．

(Ⅳ) システム変数の取扱いに関するブロック

(10) ALG ブロック

このブロックは論理変数の値を計算し設定する．

(11) ANC ブロック

このブロックは算術変数の値を計算し設定する．

(Ⅴ) イベントの取扱いに関するブロック

(12) EVF ブロック

このブロックはイベント発生の原因になったトランザクションを取り出す．

(13) RFE ブロック

このブロックは，そのブロックで指定された時間後にイベントを発生させる．

(14) MC ブロック

このブロックにトランザクションがはいると，いわゆるモニタ・コールに相当するイベントが発生する．

(15) RTN ブロック

MC ブロックと RTN ブロックの対によりスーパーバイザ・コールとそれよりの帰還の動作がシミュレートされる．

(Ⅵ) ブロック網上におけるトランザクションの流れのルートを制御するブロック

(16) BL ブロック

(17) BC ブロック

(18) BP ブロック

BL ブロック，BC ブロック，BP ブロックは，トランザクションの流路をそれぞれ論理変数，算術変数，確率変数の値に応じて変更する．

(Ⅶ) トランザクションの待行列を管理するブロック

(19) ENQ ブロック

このブロックはトランザクションを待行列に投入する．

(20) DEQ ブロック

このブロックはトランザクションを待行列から取り出す．

(21) FTQ ブロック

このブロックはトランザクションを待行列から取り出すが，そのトランザクションの待行列内における順序位置は，そのまま保存される．

DEQ ブロック，FTQ ブロックでは，待行列管理方式パラメータの指定により，先入先出，後入先出，ある条件に合ったトランザクションの取り出しなどが自動的に実行される．

(22) RSQ ブロック

このブロックは待行列内の要素の順序を変更する．

(Ⅷ) その他のブロック

(23) CAL ブロック

(24) SMP ブロック

(25) UCD ブロック

このブロックは特殊なブロックで，特定の機能は何も持っていない．このブロックは，PACSS の利用者がなんらかの理由で，上記24種類の標準ブロック以外の機能を持ったブロックを用いたいときに，随意に固有のブロックを作成できるようにするために用意されたものである．利用者は汎用言語を用いて所望の機能を記述することにより，任意の機能をこのブロックに持たせることができる．

5. 測定と統計レポート作成機能

シミュレーション実行中に測定される変数には，あらかじめシステム自体 (PACSS) によって決められたものと利用者によってその都度指定されるものとがある．これらの変数に関する測定データは，シミュレーション中には，すべていったん磁気テープに格納され，シミュレーション終了後に一括統計処理されて所要の統計データが出力される．また，測定方法に関して，測定開始時刻，測定終了時刻，サンプリング測定のためのサンプリング周期などは任意に設定可能である．

ある種の統計量を得るためとシミュレーション・モデルのデバッギングを容易にするためとの目的で，PACSSには5種類の測定ブロックが用意されている．それらはつぎのようなものがある．

(1) MCL 1 ブロック

(2) MCL 2 ブロック

この2つのブロックは対にして用いられる．ブロック網の任意の2箇所にそれぞれのブロックをそう入すると，このブロック網上でトランザクションが MCL 1 ブロックを通過してから，MCL 2 ブロックを通過するまでに要した経過時間が測定される．

(3) STAT ブロック

このブロックは，そこをトランザクションが通過するごとに，そのブロックで指定された変数の値を測定し，これに関する統計量を作成する．

(4) BTR ブロック

(5) ETR ブロック

この2つのブロックも対にして用いられる．ブロック網上 BTR ブロックから ETR ブロックに至るトランザクションの流れに関する追跡情報が出力される．

さて，PACSS において，シミュレーション後出力される統計項目には，つぎのようなものが含まれている．

(1) トランザクションが通過した全ブロックの通過時間のクラス別累計

(2) ファシリティ，ストーレジ，周辺リソースの使用実績に関する統計量

(3) 会話時間に関する統計量

(4) 各コマンド別および全コマンドに対する応答時間の統計量

(5) 待行列の長さに関する統計量

(6) 待時間に関する統計量

(7) MCL 1 ブロックから MCL 2 ブロックに至るトランザクション経過時間の統計量

(8) STAT ブロックで測定された変数に関する統計量．たとえば，STAT ブロックでのトランザクション通過率

(9) 指定された変数のサンプル値に関する統計量

(10) トランザクション追跡データを指定により整理編集したレポート

など．

先に述べたように，シミュレーション実行中に測定されたデータのほとんどは，生データの形のままで磁気テープに保存されている．したがって，同一シミュレーションに関する統計諸量は，そのシミュレーションを繰り返し実行することなく，統計パラメータを変更して幾通りにも再生可能である(ただし，上記(1)，(6)，(8)は除く)．たとえば，レポート作成の対象期間，度数分布の階級幅を変更して応答時間に関する統計量を再生成することなどが容易に行なえる．

また，全測定期間を幾つかの部分区間に分割して，これらの部分区間ごとの統計量を作成することも可能である．さらに，シミュレーションをある適当な時点で中断し，それまでのシミュレーションに関する統計レポートを作成し，その結果に基づいて（もし必要とあれば）モデルのパラメータを変更して，シミュレーションを続行させることも可能である．

このような機能により，シミュレーション実行時間，統計レポート作成時間が実質的に大幅に短縮され，総体的にシミュレーション実施効率が著しく向上する．

6. 内部構造とシミュレーション機構

PACSS は3個のサブシステム，エディター，シミ

ュレーション実行部，レポート・ジェネレータから成り立っている．これらのプログラムのほとんどの部分は，システム記述用言語 BPL (Basic PL/1 : PL/1 のサブセット[4]) で書かれており，残りの部分はアセンブラ言語で書かれている．各プログラムのおよその規模はつぎのとおりである．

エディター：BPL で約 21,700 ステップ

シミュレーション実行部：BPL で約 8,400 ステップ＋アセンブラ言語で約 100 ステップ

レポート・ジェネレータ：BPL で約 6,900 ステップ＋アセンブラ言語で約 100 ステップ

また，これらのプログラムは，いずれも 256 K 字以上の主記憶容量を持った NEAC シリーズ 2200 コンピュータ上で実行可能である．

PACSS 言語（ブロック記述型言語）で記述された CS モデルは，まずエディターにより内部システム・イメージ（コンピュータ内部における CS モデルの表現）に変換される(**Fig. 5** 参照)．この内部システム・イメージに基づいて，シミュレーション実行部はシミュレーションを実行し，所定の変数につき所定の時期，方法で測定を行なう．測定されたデータの大部分は，即座に生データの形のままで，また一部分はシミュレーション実行終了後に累計化された形で，それぞれ磁気テープに書き込まれる．シミュレーション実行終了後，レポート・ジェネレータはこれらのデータを指定の方法に従って処理し，必要な統計量を出力する．

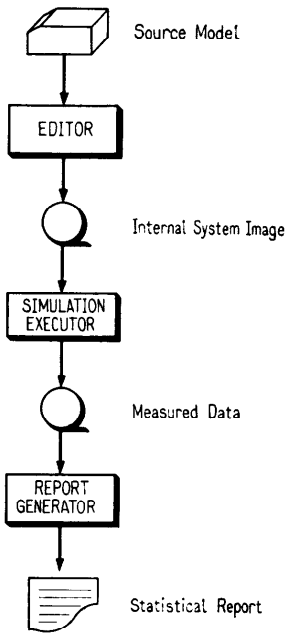

Fig. 5 steps of simulation in PACSS

CS モデルの内部システム・イメージは特性データと状態変数から成り，これらは，コンピュータの主記憶装置上に展開された CS モデルの特性と状態をそれぞれ表現している．特性データは，トランザクション・データ，リソース・データ，JOG ブロック・データ，JOB ブロック・データ，ブロック網データなどを含んでいる (**Fig. 6** 参照)．また，状態変数は，トランザクション，リソース，待行列などの状態を表わす PACSS 共通の変数と利用者により定義された特定のシステム変数とから成っている．

シミュレーション実行部は，さらに，シミュレーション・モニタ，タイミング・コントローラ，JOG ブロック・ルーチン，JOB ブロック・ルーチン，その他のブロック・ルーチンなどから構成されている．また，シミュレーション実行部には，モデル上の時間進捗を管理するための 1 個の時計と 4 個のイベント・リストが含まれている (**Fig. 6** 参照)．

各ブロック・ルーチンは，それぞれに対応したブロック・データを参照しながら，翻訳的にそれぞれの機能を実行し，それらの機能実行を通じて，トランザクション・リスト，リソース・テーブル，待行列テーブル，その他の変数テーブルなどの内容を逐次変更していく．同時に，所要の測定が行なわれ，測定データは外部記憶装置に書き込まれる．

PACSS では，シミュレーション・モデル上の経過時間はすべて整数で表わされ，これの最小単位をクロックという．すなわち，1 クロックとはシミュレーション実行部に含まれる時計の内容の単位変化によって示される時間のことである．PACSS で取りうる時間の範囲はクロック単位で 0 から $2^{24}-1$ である．シミュレーション実行部は，実際のシミュレーションに当っては，時間をクロック単位ごとにステップ バイ ステップには進めない．丁度 GPSS の場合と同じように，時間は，このつぎに最も早く起こるイベントの発生時刻まで一挙に進められる．

PACSS にはつぎのような 3 種類のイベントがある．

（1） CPU イベント：あるブロックにあるアクチベータ，もしくはトランザクションがそのブロックを離脱するという事象．それらが離脱する瞬間に，そのブロックの機能が実行される．CPU イベントが引き続いて発生しているときは，アクチベータ，あるいはトランザクションがブロック網上を正常に（割り込ま

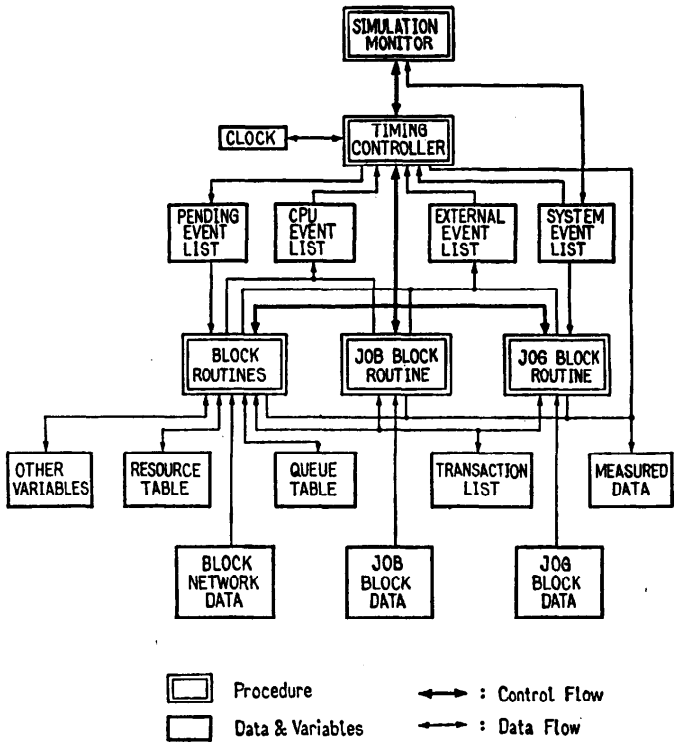

Fig. 6 The block diagram of Simulation Executor and the internal system image

れることなく）動いていることになる．

（2） 外部イベント：ブロック動作に対する外部割込みが発生するという事象．もし，外部イベントのモードの方が現在実行中のブロックのモードより高ければ，割込みが発生し，アクチベータはそのイベントにより指定されたブロックに移される．もし，そうでなければ，そのイベントはイベント保留リストに登録され，ブロック動作が続行される．

（3） システム・イベント：シミュレーション実行管理上の動作が発生するという事象．シミュレーション実行管理上の動作とは，測定開始，状態変数のサンプリング，シミュレーション中断，終了などのことである．

当然のことではあるが，モデル上の1台の中央処理装置に対して，1つの CPU イベントの（時系）列が発生する．PACSS では，同時に最高8個までの CPU イベントの列を発生させることができる．すなわち，最高8台までの中央処理装置（多重処理装置）が同時にシミュレート可能である．上に述べた3種類のイベントは，すべてタイミング・コントローラによって管理される．すなわち，タイミング・コントローラは，このつぎに最も早く発生すべきイベントを見つけ，時計をそのイベントの発生クロックまで進め，イベントの種類，モードなどを判別し，しかる後，そのイベントに対応した処理ルーチン（ブロック・ルーチンなど）にコントロールを渡す．このような過程を通じて，タイミング・コントローラはシミュレーションの進行を管理する．シミュレーション実行部のフロー・チャートの概要を **Fig. 7** に示す．

7. むすび

コンピュータ・システムの動作をその環境特性と関連づけて一体化して表現する CS モデル，およびそのモデルに基づきコンピュータ・システムの性能評価を目的として開発された専用シミュレータ PACSS について述べた．

CS モデルや PACSS の開発に当ってとくに意図した点は，まえがきにも述べたとおり，モデル記述の容易性と出力統計量の豊富・柔軟性の2点にあるが，その他にも，シミュレーション実行時間の短縮，レポー

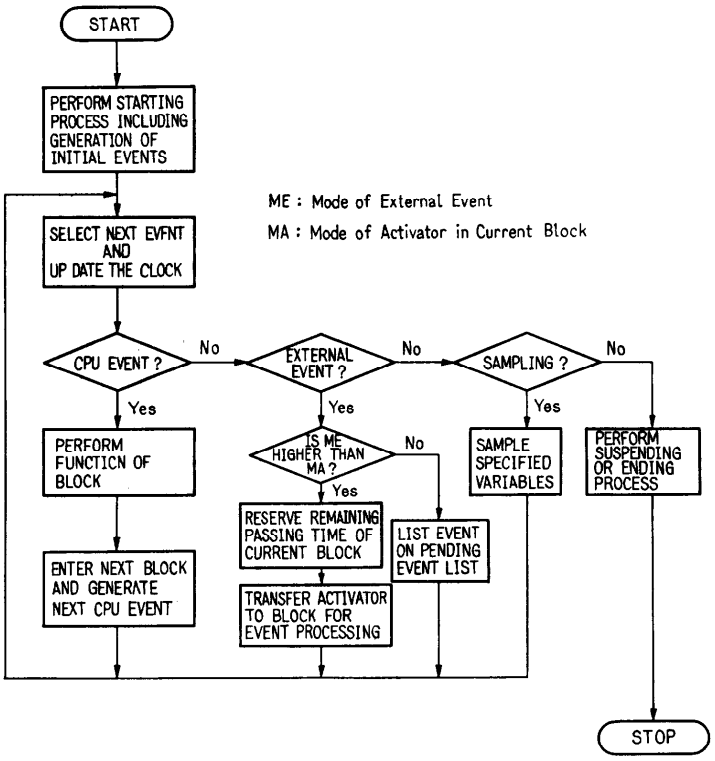

Fig. 7 General flow chart of simulation executor

ト作成時間の短縮など全体としてシミュレーション実施効率の向上に寄与すると思われる点については充分の考慮がはらわれた．これらの考慮された点が，実際にはこれまでに述べた PACSS のいろいろな機能として具現されているわけである．ここに，それらの特長ある機能を，GPSS の機能などと比較較しながらまとめてみると，以下のようになる．

（1） コンピュータ・システムの動作を，利用者，利用者プロセス，オペレーティング・システム，および金物相互間の関連動作としてとらえており，このような構成条件を満たす限り，どのようなシステムでもシミュレーションの対象となりうる．

（2） 利用者および利用者プログラムの特性が，JOG ブロックおよび JOB ブロッの使用により，容易かつ詳細に記述できる．GPSS や SIMSCRIPT でこのような記述をしようとすると，かなりの大作業になる．

（3） ENQ, DEQ, FTQ, EVF, RFE などのブロックにより，待行列やイベント・リストを陽に操作することができる．GPSS でこのような操作を行なうことは非常に困難であり，ある場合には，全く不可能である．

（4） アクチベータにより，中央処理装置の制御の流れが簡単に取り扱える．同じことを GPSS のブロックで記述すると冗長になり，ブロック網が不必要に大きくなる．

（5） 同一シミュレーションに関する統計レポートは，同じシミュレーションを再実行することなく，レポート・パラメータを変えて再生成することが可能である．

（6） 部分区間ごとの統計量作成が可能である．

（7） 応答時間に関する詳細な統計量が作成できる．

（8） トランザクションの追跡データを編集して出力することができる．GPSS には，この種の機能はないが，これはモデルのデバッギングに非常に便利である．

（9） シミュレーションの実行中断，レポート作成，モデル・パラメータの変更，シミュレーションの続行という一連の操作が可能である．

(10) 標準ブロックの1つとして，利用者がその機能を自由に定義できる UCD ブロックがある．

総評して PACSS は，GPSS や SIMSCRIPT に比較して（当然のことではあるが）コンピュータ・システムのモデル記述の容易性，総計レポート生成の柔軟性という2点においてすぐれている．シミュレーションの実行速度という点に関しては，同じレベルのモデルでは PACSS は SIMSCRIPT ほど速くはないが，GPSS よりかなり速いといえる．

上に掲げられたような種々の特長は，もちろん，多くの実施経験によって，初めてその効果が裏付けられるものである．著者らは PACSS を用いてすでに幾つかのシミュレーションを実施した．その結果，PACSS のシミュレーション実行速度，モデルのデバッギング支援機能に関して，若干の改善が必要であることが認められている．これらの問題点およびシミュレーション結果については，また別の機会に報告することとしたい．

謝 辞 終わりに，日本電信電話公社大前調査役以下多くの方々には，PACSS の基本機能に関する討議，PACSS による DEMOS タイムシェアリング・システムのシミュレーション実施などを通じて，多くの有益なご示唆をいただいた．また，日本電気・中央研究所渡部コンピュータ・サイエンス研究部長，同部緒方研究マネジャーには，シミュレーション評価技術について，日ごろ高い立場より適切なご指導をいただいている．PACSS の設計，製作に当っては，データ通信システム事業部杉崎第二システム技術部長，同部新田技術主任よりいろいろな面で多くの援助をいただいた．さらに，多くの方々が労を惜しまずプログラムの作成にあたられた．これらの方々に，ここに深く感謝の意を表したい．

参考文献

1) L. R. Huesmann and R. P. Goldberg: Evaluating Computer Systems through Simulation, The Computer Journal, August 1967, pp. 150～156.
2) P. H. Seaman and R. C. Soucy: Simulating Operating Spstems, IBM Systems Journal, Vol. 8, No. 4, pp. 264～279, 1969.
3) L. J. Cohen: System and Software Simulator S 3, C. E. I. R. Inc., 5772 River Road, Washington, D. C., 1968.
4) 小久保靖世，佐谷鉄夫：コンパイラ記述用言語 BPL，情報処理，Vol. 11, No. 6, pp. 342～349, 1970.
5) 三上 徹ほか：汎用TSSシミュレータPACSS，昭和45年電気四学会連合大会，No. 2585.
6) 久保秀士ほか：PACSS の機能と構成，昭和45年電気四学会連合大会，No. 2586.

（昭和45年9月4日受付）