計算機アーキテクチャ 77－5
（１９８９．７．１３）

対話型ビジュアル･シミュレータ

CAB(Computer Architect's Board)
岡田　義広　　田中　譲
北海道大学　工学部

CABは、グラフィカルな待行列網解析ツールであり、計算機のシステムレベルにおける性能評価を目的として開発した対話型シミュレータである。CABでは、対象となる計算機システムをグラフィックス画面上に視覚化し、個々の装置あるいはデバイスのグラフィックイメージをオペレータが画面上で組み合わせ作図することにより、対象システムを記述でき、種々のシステム構成の性能評価を対話的に行うことができる。このシミュレータ･システムの特徴と有効性について報告する。

INTERACTIVE AND VISUAL SIMULATOR

CAB(Computer Architect's Board)
Yoshihiro Okada　Yuzuru Tanaka
Electrical Engineering Department　Hokkaido University
Sapporo 060,Japan

We have developed the simulator CAB. CAB is a queueing-model-based graphical tool for performance analysis of computer systems. Computer system is described as a graphical image on a graphics display. Therefore a user can describe computer system by composing components' graphical images into a total image and execute simulation in an interactive manner. This paper reports the feature and the validity of the CAB simulator system.

1. はじめに

計算機の性能評価を行う場合、解析的方法とシミュレーションによる方法があるが、確率的要素が含まれたり必要なパラメータが多数ある場合には解析的評価は困難であり、シミュレーションによる評価が行われる。計算機のシステムレベルにおける性能評価では、ランダムな要素が多く含まれるため、シミュレーションによる評価が一般的である。従来のシミュレーションによる評価では、対象システムをモデル化しこれをシミュレーション用の記述言語あるいはプログラミング言語等を用いてテキスチュアルに記述し、実行形式へ変換あるいはコンパイルし、それを実行することで結果を得ていた[1]。しかし、これではモデルの動作と妥当性がつかみにくく、対象システムの構成の変更も容易に行うことができないなど、最適な結果を得るまでに多くの時間と労力を費やすことになる。そこで我々は、テキスチュアルな記述をせずに視覚的に対象システムを記述でき、実行形式への変換(コンパイル)をすることなしに対話的にシミュレーションを行うことができ、しかもその動作をグラフィカルに表現することのできるシミュレータを開発した。開発はSymbolics3620上でCommon Lisp及びFlavorを用いて行った。本論文では、このシミュレータの概要と実行機構について述べる。

CABの特徴

計算機のシステムレベルにおける評価解析では、ペトリネット網解析[2]、待行列網解析[6]がしばしば用いられる手法である。待行列網解析は、対象システムの各装置あるいはデバイスを1組の待行列(Queueing Model)で表し対象システムに合わせそれらの接続関係を与えることでシステムを記述できるためモデル化が容易であり、しかもグラフィカルな記述に適していると考えられる。従って、本シミュレータでは、待行列網解析によって対象システムの性能評価を行うこととした。計算機システムの各装置をQueueing Modelを用いて抽象化しそのモデルを視覚的に合成することによって対象システムを記述できるようにした。Queueing Modelをグラフィカルに表示し動作させシミュレーション中のシステムの動作を視覚的に理解できるようにした。さらに、Queueing Modelに内部状態を持たせその動作を状態遷移により記述することをも可能とし表現に柔軟性を持たせている。

1) モデル合成による計算機システムの記述

モデル合成とは、シミュレーションの対象を記述する場合に、対象全体を一度に定義するのではなく、まず対象を構成している部分要素へ分解し、その部分要素を個々にモデル化して定義し、この定義されたモデルを適当に組み合わせることによって対象全体を記述しようとするものである。従って、本シミュレータによる計算機システムの評価解析では、(1)部分要素であるモデルを定義することと、(2)定義されたモデルを用いて対象システムを記述しシミュレーションを行うことの二つの手続きを行う。部分要素の定義では、テキスチュアルな記述ができるためにモデルの動作を柔軟に記述することができ、しかも容易に部品化を計ることができる。シミュレーション時には、定義された部分要素のモデルを適当に組み合わせることで対象全体を記述することができ、対象システムが与えられたときにその構成から容易にシステムの記述をすることができるため、その変更も容易に行うことができ、対話的に作業をすることができる。

2) グラフィカルなモデルの表現

モデル合成を行う場合、個々のモデルがスクリーン上のグラフィックイメージとして視覚化されていると、オペレータは画面上でこのグラフィックイメージを適当に組み合わせ作図することによって対象システムを容易に記述することが可能となる。また、モデルの動作もグラフィカルに表現されていることによってシミュレーション中のシステムの動作を視覚的に理解することができ、シミュレーションよって得られる数値結果だけでは知ることのできない対象システムの動作やボトル・ネックなどを見つけることが容易にできる。本シミュレータでは、Queueing Modelをグラフィカルに視覚化し、このグラフィックイメージを対象システムの構成に合わせ適当に

組み合わ作図(エデット)することによって容易に対象システムを記述することができる。

3) 状態遷移による動作記述

待行列網解析では、個々の待行列における窓口(Server)のサービス時間と、ジョブ(Token)の到着時間間隔が重要な要素である。これらの要素を与えることで網全体の性能評価を行いボトル・ネックを見つけることができる。しかし、窓口でのサービスの内容については考慮されないため、サービスの内容を細かく記述することができない。そこで、本シミュレータではServerを確率遷移オートマトンとして記述することもできるようにし、Serverに内部状態を持たせQueueing Modelへの入力であるTokenにラベルを与え、このラベルの種類によってServerでの処理時間を変えたり、出力するTokenの種類を決めることができるようにした。

対話型グラフィックスを実現したシミュレータとしては、VLSI設計用のCADシステムとしていままで多数のシステムが開発されている。しかし、計算機のシステムレベルで対話型グラフィックスを実現したシミュレータ[11]は少ない。Queueing Modelをグラフィカルに視覚化したシミュレータとしては、B.Melamedらによって開発されたシステム[3]があるが、これはモデルの登録ができないため待行列網解析ツールとしての用途が強くシステム記述の柔軟性に欠ける。しかし、このシステムを使用した研究発表[4,5]もいくつか出され計算機のシステムレベルの評価において待行列網解析が有効であることが分かる。

以上が本シミュレータの特徴である。以下、第2節で本シミュレータの概要を説明する。第3節では、本シミュレータの開発環境とオブジェクト指向言語について述べる。

2. CABの概要

CABは、シミュレーションを行うためのウィンドウシステムと、計算機システムを記述するために予め定義されるモデル類から構成されている。また、動作方法としてイベント指向を採用している。以下、それぞれに関して詳しく説明する。

2.1 ウィンドウシステム

図2.1はCABの画面ハードコピーである。本シミュレータは、エディット、トレース、グラフ、リポートの各ウィンドウから構成されており、それぞれのウィンドウが統合された環境を実現している。

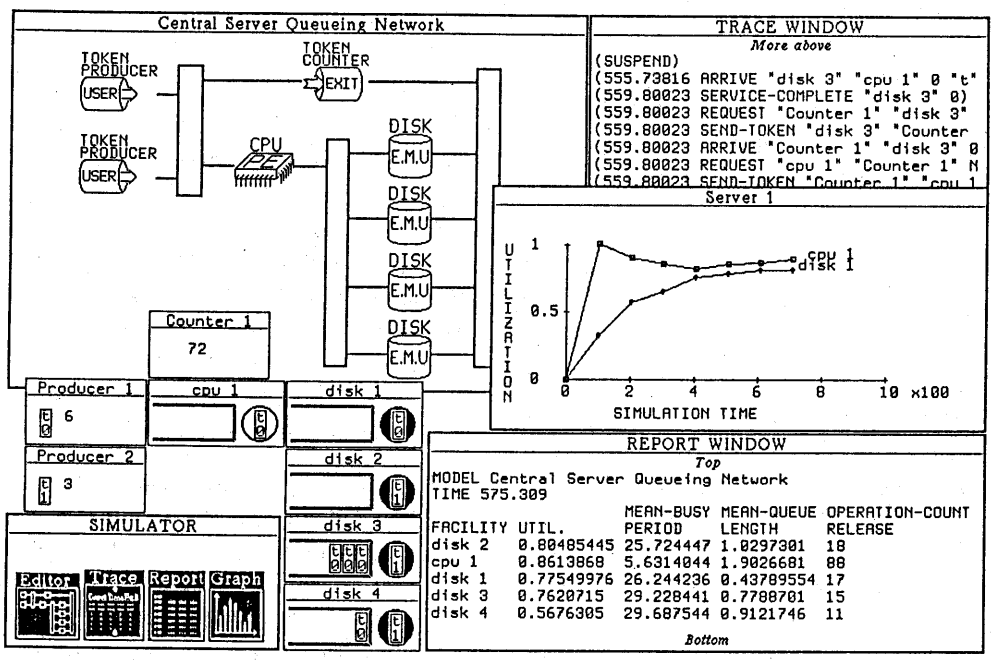

図2.1 CABの画面ハードコピー

オペレータがシミュレーションを行う場合、このウィンドウシステム内で行う。エディット･ウィンドウ内で対象システムをエディットしシミュレーションを行い、結果を見たい場合には、グラフ･ウィンドウでグラフ表示し、リポート･ウィンドウで数値表示する。シミュレーション中のイベントの流れを見たい場合には、トレース･ウィンドウでイベントをトレースする。これらの一連の操作によってシミュレーションを行うわけであるが、すべてマウスを用いて対話的に行うことができる。

･エディット･ウィンドウ

図2.1画面左上が計算機システムを記述するためのウィンドウで、予め定義されているモデル(計算機システムを構成する装置あるいはデバイス)がグラフィックスイメージとして表現され、オペレータはこれを画面上で対象システムに合わせ適当に配置することでシステムを記述することができる。このときの操作もすべてマウスにより行うことができ、あたかも紙の上で作図するかのごとく行うことができる。

･トレース･ウィンドウ

図2.1画面右上がトレース･ウィンドウである。本シミュレータの動作方法はイベント･ドリブンであり、シミュレーション中にこのイベントをトレースすることで、各装置の動作や処理の流れを詳しく観ることができどのようにシステムの状態が変化し動作しているのか理解することができる。

･グラフ･ウィンドウ

図2.1画面右中がグラフ･ウィンドウである。対象システムのある装置の特定の値に関して、シミュレート時間とともにその値の変化をグラフ表示するためのウィンドウである。これを観ることによってオペレータは容易に最適な結果を導きだすことができる。グラフ･ウィンドウは、決められたフォーマットのデータを与えられるだけで、その表示のための修正(領域、タイトルなど)を自動的に行う。

･リポート･ウィンドウ

図2.1画面右下がリポート･ウィンドウである。シミュレーションし得られた値を集計し、メニュー選択により様々な結果をフォーマットして数値表示するためのウィンドウである。これを観ることによって各装置の処理結果を詳しく理解することができ、比較検討を行うことができる。

2.2 計算機システムの記述用モデル

本シミュレータでは、計算機のシステムレベルでの性能評価を待行列網解析により行うものであり、予め定義されているモデルとしてQueueing Model,Token,Token-Producer, Token-Counter,Line,Hierarchy-Facilityがある。

･Queueing Model

本シミュレータでは、Queueing Modelをグラフィカルに表示し、各装置の処理を視覚的に理解できるようにした。どのようなジョブ(処理)がどのくらいの長さQueueに入っているのか、どのようなジョブがServerで処理されているのか、処理中であるのか、ないのかなど、シミュレーション中にグラフィックイメージとしてリアルタイムで観ることができる。各Queueing Modelはプロパティを持ち、シミュレーションの後では結果を導くための値を保持しそれぞれメニュー選択により対応するウィンドウを開いて表示、変更することができる。図2.2がQueueing Modelのグラフィックイメージでありプロパティ･シートと結果を表示したところである。プロパティには、Queueing Model特有の要素(サービス時間、Queueの容量、Serverの数)のほかに、Tokenの流れを制御するための様々なパラメータがある。結果として保持しているものには、待行列網解析によって得られる結果を導くための値すべてである。

本シミュレータでは、Queueing Modelの動作を状態遷移により記述することもできる。Queueing ModelのServerに内部状態を保持させ、入力されるTokenの種類により予め決められた確率で状態が遷移し、それに対応してサービス時間を変えたり、出力するTokenの

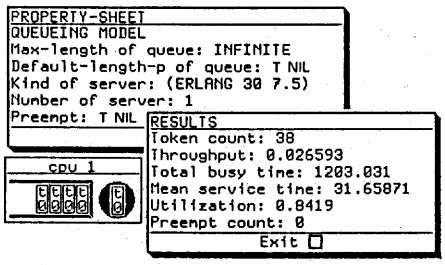

図2.2　Queueing Modelのグラフィックイメージ

種類を変えたりできる。これによって、計算機システムの各装置の動作を柔軟に記述できるようになった。

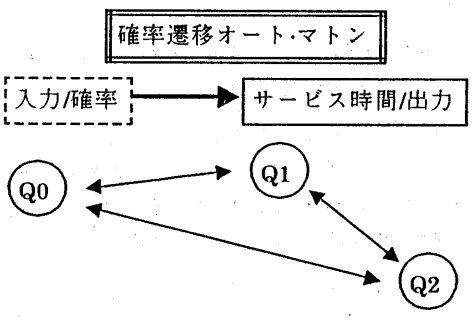

図2.3　状態遷移による動作

・Token

ジョブ(処理)を表すモデルである。プロパティとして、ジョブの種類を示すラベルと優先順位を示すプライオリティ・ナンバがある。Queueing Modelの要素であるサービス時間はこのラベルに対応して定義されており、同一の装置でジョブが異なる場合にもそれぞれのジョブに対応したサービス時間で処理が行われる。また、Queueing ModelのQueueではその待行列をこのプライオリティ・ナンバに従って構成する。

・Token-Producer

Tokenを予め決められた時間間隔(確率分布)に従って生成し、システムあるいはQueueing Modelに送るためのモデルである。プロパティとして、生成するTokenの種類と確率分布があり、メニュー選択により対応するウィンドウを開いて表示、変更することができる。従って、Token-Producerをいくつか用意することで対象システムに必要なジョブ(処理)をすべて表すことができる。これも、グラフィックスイメージとして表示されているので、シミュレーション中に生成したTokenの種類と数をリアルタイムで観ることができる。

・Token-Counter

システムあるいはQueueing Modelで処理されたTokenの回収や、カウントを行う。時に、シミュレートの実行終了の制御も行う。これも、グラフィックスイメージとして表示されており、シミュレーション中にカウントした数をリアルタイムで観ることができる。

上記の各モデル(Queueing Model、Token-Producer、Token-Counter)はそれぞれアイコン化され、このアイコンを用いて対象システムを記述する。

・Hierarchy-Facility

スクリーン上でシステムを平面的に記述するには限界がある。従って、大きなシステムを記述する場合には階層的に記述できる必要がある。そのための部分システムを記述するためのウィンドウである。下位のレベルとして部分システムをこのウィンドウ内で記述し、これをアイコン化して、その上位のレベルのシステムを記述するためのウィンドウ内でこのアイコンを用いる。

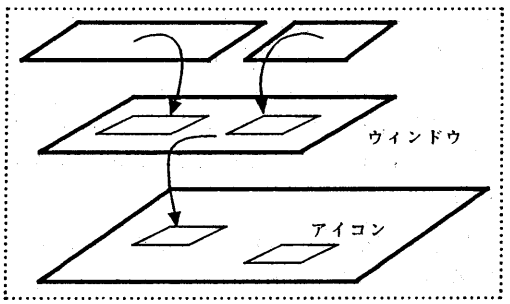

図2.4　アイコンによるウィンドウの階層構造

・Line

アイコン化され適当に配置された上記の各モデルを接続し、これらの接続関係を与えるためのモデルである。

以上のモデルは、本シミュレータが待行列網解析ツールとして使用するのに必要な基本要素を予め定義したものであり、必要に応じ

て新たにモデルを定義し組み込むことも可能である。

2.3 動作方法

本シミュレータでは、対話的にシミュレーションが行えることを目指している。そこでシミュレータの動作方法としてイベント指向を用いた。実現方法としては、モデルの動作をメッセージ(イベント)パッシングにより行い、イベントを管理するためEvent-List(イベントキューと呼ばれる)を要素に持つEvent-Managerを設けた。

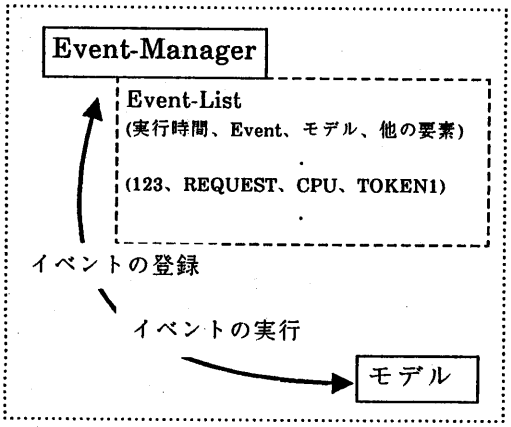

図2.5 シミュレーションの実行過程

Event-Listとは、イベントが起きるべき時間、イベントの種類とイベントが起きるモデル、そのほかに必要な要素を一つのEventとして記述し、それを時間順に保持しておくものである。各モデルは、メッセージを受けることで動作し状態を変化させ、この時新たに起きるEventがあればそれをEvent-Managerに送る。

Event-Managerでは、Event-Listの最初の要素(最初に起こるべきイベント)を引出し評価する。評価によってモデルに対してメッセージが送られる。このとき、モデルから新たなEventが送られたならそれをEvent-Listに登録する。このような一連のサイクルによって一イベントが実行される。従って、Event-Managerを適当に動作させることでシミュレーションを行うことができ、Event-Listの内容を変えなければシミュレーションの途中で中止、再開が自由にでき対話的な操作が可能となった。

各モデル間では、以下のEventによってTokenのフロー制御を行う。

(1) ARRIVE : Tokenの到着、サービス開始のEventを自分に対して送る。サービスが終了したときに、次に処理されるべき装置に対してアクセス可能であるかREQUESTを出す。

(2) REQUEST : アクセス可能ならば、要求のあった装置に対してSEND-TOKENを送る。アクセス不可能ならば、要求のあった装置に対してNOT-SEND-TOKENを送る。

(3) SEND-TOKEN : 次の装置にARRIVEを送る。

(4) NOT-SEND-TOKEN : ある一定の時間間隔をおき再びREQUESTを出す。

Eventは、各モデルのメソッドとして定義されているものであり、モデル固有のEventを新たに定義することで、そのモデルの細かな動作を表現することもできる。

2.4 評価結果について

待行列網解析では、予め個々の待行列の窓口での平均のサービス時間とシステムへ客が到着する平均の時間間隔が既知でなければならずこれをパラメータとして与え、このときの待行列網全体としての性能を評価する。

個々の待行列で得られる結果として以下のものがある。

T…シミュレート時間、C…サービスが終了した客の数、A…客が到着した数。
L=A/T…arrival rate(単位時間に到着した客の数)

X=C/T…throughput rate(単位時間にサービスが終了した客の数)

B…total busy time(全サービス時間)

M…mean busy time(一人の客に対する平均のサービス時間)

U=B/T…utilization(利用率)

これらの結果を評価して、待行列網全体の性能を評価するわけである。

計算機システムの評価では、計算機システムを構成する個々の装置の平均の処理時間

と、システムへ到着するジョブの平均の時間間隔が予め既知でなければならない。得られた結果からシステムのボトル・ネックを見つけ出しシステム全体としての性能を評価することになる。

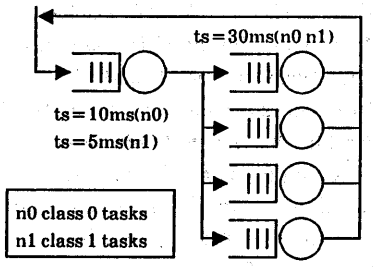

図2.6 Central Server Queueing Network

図2.6はCentral Server Queueing Networkシステムである。このシステムは、参考文献[1]にあるもので、参考例としてシミュレーションしたのでその結果を示す。図2.7がその結果をリポート・ウィンドウで表示したものである。

3 開発環境とオブジェクト指向言語

Symbolics上で開発を行った理由としては、まず第一にユーザ・インターフェイスが整っていることが挙げられる。本シミュレータでは、最初から使い易さに重点をおいていたため、マウスによる入出力をはじめウィンドウ操作などユーザ・インターフェイスが整っている必要があった。第二に、オブジェクト指向言語が使えることが挙げられる。シミュレータを開発するうえで、オブジェクト指向言語はプログラミングの点で有利である。第三に、処理速度が良くなければならなかった。シミュレーションをする場合、動作を何千何万回と繰り返すことがあり、実用に耐えられる処理速度が必要であった。以上の点から、Symbolics上で開発を行った。また、Lisp[8]環境自体がシステムの構築に有利であると思われる。

1) オブジェクト指向言語

Flavor[7]は、Lisp環境上で実現されたオブジェクト指向言語である。オブジェクト指向言語は、抽象データ型、メッセージ・パッシング、クラス継承が特徴であるが、なによりそのプログラミングが、オブジェクトの振る舞いを定義することである点に特徴があるといえる。一方、シミュレーションとは、ある対象に関してその動作や状態を模倣することであり、それはオブジェクト指向言語におけるオブジェクトを定義することと同じであると考えられる。従って、オブジェクト指向言語を用いることでシミュレーションの対象を容易に記述することができる。

実際、シミュレータの開発にオブジェクト指向言語を用いた例[10]は多数あり、ハードウェアのモデル化を目的としたオブジェクト指向言語の開発[9]も行われている。

2) Symbolicsのウィンドウ・システム

Flavorは、Symbolics上でウィンドウ・システムを構築するために実現されたものであり、Symbolicsのウィンドウ・システムはすべてFlavorによって書かれている。Flavorによって記述されているために、その特徴である継承によってユーザは容易に新たに目的のウィンドウをつくることができる。また、このウィンドウ・システムには、様々な種類のウィンドウとメニューが用意されており、ユーザはパラメータを設定するだけで簡単に

REPORT WINDOW
Top

TIME 96527.555

FACILITY	UTIL.	MEAN-BUSY PERIOD	MEAN-QUEUE LENGTH	OPERATION-COUNT RELEASE	PREEMPT	QUEUE
cpu 1	0.8151102	6.20167	1.2857765	12687	2682	7335
disk 1	0.7712469	29.970442	0.9329364	2484	0	1838
disk 2	0.75607973	29.703917	0.8465788	2457	0	1787
disk 3	0.7803563	29.843853	0.9715541	2524	0	1896
disk 4	0.7886158	30.028858	1.0513574	2535	0	1931

Bottom

図2.7 リポート・ウィンドウの画面ハード・コピー

使用することができる。しかし、このウィンドウ・システムには、アイコン化の機能がなかったため、アイコン化の機能を新たにウィンドウ・システム自体に組み込んだ。

3) 処理速度

本シミュレータでは各モデルがアイコン化されているため、アイコンを開いた状態でグラフィック表示を行いながらシミュレーションを行う場合と、アイコンをすべて閉じた状態でグラフィック表示をせずにシミュレーションを行う場合が考えられる。グラフィック表示は非常に時間を要する処理であり、当然前記の場合多くの処理時間を要することになる。そこで、前述したCentral Server-Queueing Networkシステムを例にシミュレーションし処理速度の比較をした。

グラフィック表示した場合‥‥11分05秒

グラフィック表示しない場合…6分20秒

以上のように数値結果だけを得たい場合には、グラフィック表示をせずにシミュレーションを行うことで処理速度の向上が図れる。この結果は、十分に実用に耐えられる速度であると思われる。

4 おわりに

以上、本シミュレータの仕様および開発環境について述べた。特に、計算機システムの記述にモデル合成を用い、各モデルをグラフィカルに表現したことによるメリット、およびQueueing Modelに内部状態を持たせ状態遷移により動作を記述することで、Queueing Modelでは表現できない計算機装置の細かな動作を記述することができた点について述べた。今後このシミュレータを用いて様々な計算機システムあるいはアーキテクチャの評価を実際に行い研究に生かしていくつもりである。また、本シミュレータシステムを待行列網解析のシステムとしてだけではなく、より広範囲のシミュレーションシステムとして拡張していくつもりである。

参考文献

1) M.H.MacDougall,"Simulating Computer Systems:Techniques and Tools",Computer Systems Series,The MIT Press,1987.

2) M.Ajimone Marsan,G.Balbo,and G.conte,"Performance Model of Multiproceessor Systems",Computer Systems Series,The MIT Press,1986.

3) B.Melamed,and R.J.T.Morris,"Visual Simulation:The Performance Analysis Workstation",IEEE COMPUTER,pp.87-94,August,1985.

4) A.S.Melambed,"Peformance Analysis of Unix-based Network File System",IEEE MICRO,pp.25-38,Febuary,1987.

5) A.A.Fredericks,"PERFORMANCE ANALYSIS MODELING FOR MANUFACTURING SYSTEMS",AT&T TECHNICAL JOURNAL,Vol 65,pp.25-34,1986.

6) E.Gelenbe,I.Mitrani共著、秋丸春夫、橋田温監訳、"計算機システムの解析と設計"、オーム社、1988.

7) Hank Bromley,"LISP LORE:A GUIDE TO PROGRAMING THE LISP MACHINE",Kluwer Academic Publishers,1986.

8) 湯浅太一、萩谷昌己著、"Common Lisp入門"、<岩波コンピュータサイエンス>、1987.

9) 杉本明、阿部茂、"オブジェクト指向言語VEGAMESによる構造レベル・ハードウェアのモデル化"、コンピュータソフトウェア、Vol3,No.3,pp.71-85,1986.

10) Pieter S.van der Meulen "INSIST:Interactive Simulation in SmallTalk",OOPSLA '87 Proceedings,pp.366-376,Octobe,1987.

11) Kathleen M.Nichols and John T.Edmark "Modeling Multicomputer Systems with PARET",IEEE COMPUTER pp.39-48, May,1988.