待ち行列システムのモデリング言語RESQ
—RESearch Queueing package—

村田　正幸

1. はじめに

RESQ (RESearch Queueing package) [6] は，1970年代中頃から，IBMのワトソン研究所において，待ち行列網(Queueing Network；以下，QNと略す.)システムのモデルを構築し，その評価を行なうためのツールとして，研究，開発されてきたソフトウェア・パッケージの名称である．その対象とするシステムは，単に，コンピュータ／ネットワーク・システムのように，従来から，QNモデルとなじみの深いシステムにとどまらず，社内では，製造ラインのシステム評価等，さまざまな分野で活用されるようになっている．

RESQは，後に詳しく述べるように

1. ネットワーク・モデルの構成要素の記述が，従来のシミュレーション言語（たとえば，GPSS）に比較して，より抽象化された形で記述でき，そのためのテンプレートも豊富に用意されている．
2. シミュレーションだけでなく，あるクラスのモデルに対しては，解析解［5］も得られる．

という特徴をもち，単なるシミュレーション言語というよりも，システム・モデリングのための支援ツールであるといえよう．

むらた　まさゆき　日本アイ・ビー・エム㈱
サイエンス・インスティチュート

特に第2版であるRESQ 2［8］［9］になって，対象を後述する拡張QN (Extended QN) モデル［7］［11］に拡げ，ネットワーク・モデルの記述能力が高まり，解析解についても，mean value analysis［12］，tree convolution algorithm［3］を実装するに至り，より強力なツールになった．

本稿は，IBM社外にすでに発表された資料をもとにRESQの概念・構成・特徴，RESQによるシステム・モデリングの方法などをまとめたものである．以下，第2節では，QNにおける各種資源やジョブの流れをRESQでは，どのように表記するのかについて簡単にまとめ，RESQによる，QNモデルの構築の方法について述べる．第3節でRESQによるQNモデルの性能評価の手順を説明する．第4節では，簡単なモデルを用いて，RESQの適用例を紹介し，最後に第5節で，RESQの最近の動向と，残された問題点をまとめる．

2. RESQネットワーク・モデルの構築

2.1 QNモデル

QNモデルは，コンピュータ／通信システムにおけるCPUやI／Oデバイス，バッファ等各種資源の集合と，コンピュータ・システムにおけるプロセスや通信システムにおけるメッセージ等，資源からサービスを受けるジョブの集合から構成されると考えられる．ジョブは，サービスを受けるためにシステム内を資源から資源へと移動し，

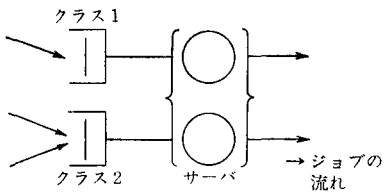


図 1 アクティブ・キュー（2クラス，2サーバ）

サービスを要求する際に，他のジョブと競合をおこす．結局，モデル化の目的とは，システムにおけるジョブの流れと，そのとき生じる，これらの競合の度合いを解析することであるといえよう．

資源は，さらに，その種類と数，サービス規律，および，到着したジョブに与えるサービス時間の確率分布によって特徴づけることができ，一方，ジョブを記述するためには，資源から資源へ遷移するその道筋（ルーティング）と到着過程を与えてやらねばならない．RESQ では，このような観点から，次の5つの要素の集合からなるQNモデルを定義し，効率的なQNモデル記述言語として設計されている．

(1) アクティブ・キュー(active queue)

アクティブ・キューは，資源を表現し，3つの要素(待行列(waiting area)，サーバ，サービス規律）からなる．待行列の記述のために，「クラス」の概念が導入され，キューにおける複数の待行列はそれぞれ異なったクラスに属すると考える．他のキューから遷移してきたジョブはいずれかのクラスに割当てる(図1)ことにより，それぞれの待行列が形成される.(一般の待行列理論でいうクラスとは異なり，ここでいうクラスは，各キューごとに局所的に定義される.）サービス時間分布は，それぞれのクラスごとに指数分布，ブランチング・アーラン分布，一様分布，PL/I 言語によって書かれたユーザー定義の分布などが，利用できる．また，指定可能なサービス規律を表1に示す．

表 1 アクティブ・キューにおけるサービス規律

FCFS	first come first served
PRTY	prioriry
PRTYPR	priority with preemption
PS	processor sharing
LCFS	last come first served

(2) ジョブ (job)

RESQ モデルにおけるジョブは「資源に対してサービスを要求するもの」と定義され，同等の性質をもつジョブごとにクラスを形成する．

(3) ノード (node)

ノードはそれ自体サービスを提供するものではないが，モデルをより精密に表現するために用いられる．表2に，そのシンボルと役割を示す．

(4) ルーティング規則

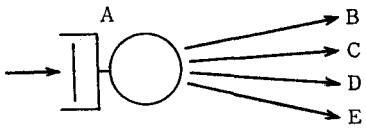
キュー（クラス）やノードからの分岐表現として，(無)条件分岐，確率的分岐が指定でき，その組合せも可能である．図2にあげた例の場合，変数 delay 1, delay 2 の大小関係によりB，Cへ分岐し，等しい場合は，それぞれ 1/2 の確率でD，Eへ分岐する．

(5) チェーン (chain)

ジョブのシステム内での流れを記述するのにチ

表 2 ノード記号とその機能（( ) 内に適用例を示す）

ノード名	記号	機　　能
source sink		システム外からのジョブの到着とシステム外へのジョブの離脱（オープン・ネットワークに適用）
set		変数に対する値の割当て
fission fusion		あるジョブに対してもう1つのジョブを派生させたりシステムから消し去る．（メッセージのパケット分割やプロセス同期に有効である）
split		ジョブに対してもう1つの独立したジョブを派生させる．(ACK メッセージの発生など)
dummy		ルーティングを決定する場合に，キューや上のノードだけでは、表現できない場合に用いる．



A → B C D E;
if(delay1>delay2) if(delay1<delay2) 0.5 0.5

図 2 キュー，ノードからの分岐表現の例

ェーンの概念を導入する．すなわち，ジョブ，アクティブ・キュー（正確にはキューにおけるクラス），ノード，および，ルーティング規則からなる1つの集合をチェーンと考える．たとえば，ポーリング・システムを考えた場合，同一システムに，データ・メッセージとコントロール・メッセージが混在するが，両者の性格は明らかに異なっており（前者はシステム外から到着してオープン(Open)チェーンを形成するのに対し，後者はネットワーク内で一定の数を保つクローズド（Closed）チェーンを形成すると考えられる），2つのチェーンが形成される．

2.2 拡張QNモデル

RESQは，QNのモデリング・ツールを提供するという意味で一応の成功をおさめたが，さらに柔軟性のあるQNモデルの表現のために，第2版のRESQ2では，次のような機能拡張がなされた．

1) ジョブ変数（job variable）の導入

それぞれのジョブに対して，ジョブ固有の情報（たとえば，ジョブをネットワーク・システムにおけるメッセージとした場合，メッセージ長など）を与える．

2) システムの状態に依存したサービス時間，ルーティング規則表現の実現

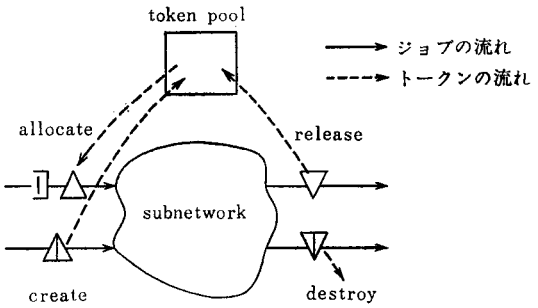
キューやノードから分岐表現として，ジョブ変数やグローバル変数，キューの状態（たとえば，待行列長など）に依存したルーティングの記述を可能にした．また，各キューにおけるサービス時間についても，同様に，たとえば，ジョブ変数を用いれば，メッセージ長に応じたサービス時間分布が設定できるようになった．

3) パッシブ・キューの概念の導入

各ジョブが共有する資源（たとえば，コンピュータ・システムにおけるメモリや，ウインドー・フロー・コントロール（window flow control）におけるウインドーなど）を表現するために，新しくパッシブ・キューの概念が導入された．共有資源はトークンの集まり（token pool）として表現され，トークンの数によってジョブが要求する資源の量を示す．トークンに対する操作は，図3に示す各ノードで行なわれ，これらのノードにより，資源の割当て待ちや，ブロッキングが表現される．

4) キュー・タイプ（queue type），サブモデル（sub model）の概念の導入

モデルを階層的に表現するために，RESQ2では，2つのテンプレート（キュー・タイプとサブモデル）が新しく導入され，これらによって，QNモデルをトップダウンの手法で構築していくことが可能になった．キュー・タイプは先にあげたキューをテンプレートとして，パラメータに変数を与えて定義する．以後は，キュータイプで定義した名前に実際のパラメータを与えることにより，くりかえし引用できる．一方，サブモデルは，サ



ノード名	機　能
allocate	トークンの割当てを待つ(クラスを指定し，それぞれに優先権を割当てることも可能)
release	不要になったトークンを放す
create	新たなトークンを発生させる
destroy	不要になったトークンを消す

図 3 パッシブ・キュー

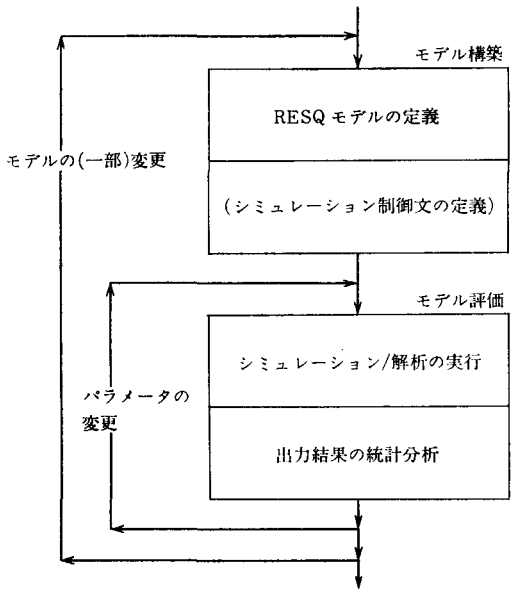


図 4 RESQによるモデル評価の手順

ブネットワークの定義を行ない，キュー・タイプと同様，他のモデルからの引用が可能になる.(詳しくは4節の例参照)

3. RESQによるQNモデルの性能評価

RESQによるQNのモデル化および，その評価の手順を図4に示す. 図に示すようにRESQによりモデル表現されたNQを，シミュレーションによる評価を行なう場合には，シミュレーション制御文を定義する必要がある. すなわち信頼区間の評価のために

(1) Independent Replication法，(2) Regenerative法，(3) Spectral法[2]のいずれを選択し，(2)か(3)の場合には sequential stopping rule(あらかじめ，指定した範囲に信頼区間が狭まると自動的にシミュレーションを停止する)[4]を適用するかどうか決め，必要なパラメータを与える. またRESQより提供される性能尺度(performance measure)は，表3に示すものの他に，明示的に宣言すればその分布が得られる. このように，シミュレーション結果に対する統計分析のための機能が豊富に用意されていることもRESQの特徴としてあげられよう.

表 3 RESQ提供のパフォーマンス・メジャー一覧

記号	意味
ut	アクティブ・サーバ，パッシブ・キューにおけるトークンの利用率
tp	スループット
ql	平均待行列長*
sdql	待行列長の標準偏差
qt	平均待時間*
sdqt	待時間の標準偏差
tu	平均使用トークン数*
tt	平均未使用トークン数*
mxql	最大待行列長
mxqt	最大待時間
po	オープン・チェーンにおける平均ジョブ数
rtm	オープン・チェーンにおける平均応答時間

(表中，*のものは，オプション指定によりその分布が得られる)

RESQ モデル・ファイルの作成は，(1) 会話モード，(2) バッチ・モードのどちらでも実行できるように設計されており，たとえば一番最初のモデル構築は会話モードで行ない(入力誤りは，直ちに検出される)，以後，モデルの一部変更は，エディタにより，直接，ファイルを書き変えてもよい. 一方，モデル評価段階においても，会話型処理が可能で，たとえば，パラメータを変更しながら，実行をくりかえし行ない，システムに与える影響を調べることも容易に行なえる. また，入力パラメータ・ファイルをあらかじめ準備しておけば，バッチ処理がシステムによって自動的に選択され，ユーザーにとっては，非常に使い勝手のよいシステムであるといえるだろう.

4. 適用例

本節では，何台かの TSS 端末と CPU，および，2台の2次記憶装置よりなる簡単な会話型コンピュータ・システムを用いて，RESQ を用いたモデリング例を示す. 図5にそのダイアグラムを示し，図6に，このモデルの記述とそのシミュレーション方法を示す.

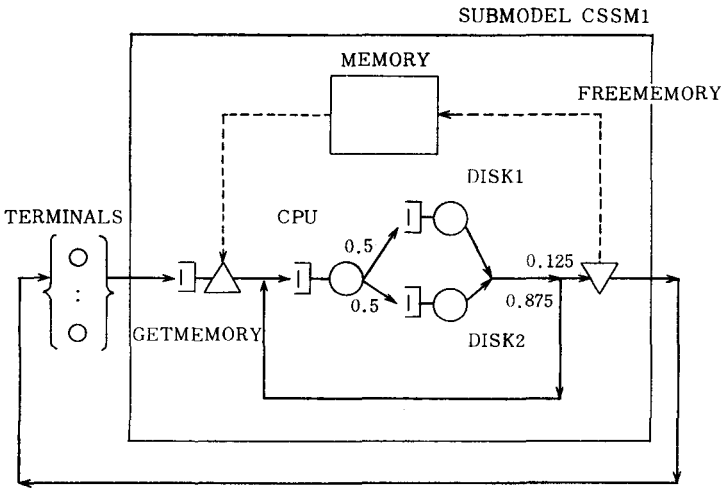


図 5 会話型コンピュータ・システム（モデル名CSM）のダイアグラム

まず，モデル名等の宣言①の後，アクティブ・キュー terminalsq ②の定義を行なう．CLASS LIST は後にルーティングの定義に用いる．

次に，サブ・モデルCSSMの定義③を行なう．2台のディスクは，同じような性質をもつとし，キュー・タイプを用いて定義する．そのために，まず，キュー・タイプ diskdef ④の定義をする．ノード・パラメータ⑤により，disk 1, 2，それぞれのクラス名が後に定義される．サブ・モデルは，メモリをあらわすパッシブ・キュー MEMORY ⑥と，CPU，ディスクをあらわすアクティブ・キュー CPUQ ⑨，DISKQ 1 ⑩，DISKQ 2 ⑪よりなる．MEMORY におけるトークンはページ・フレーム数を示し，それぞれのジョブに対して，確率0.25，0.5，0.25で，16，32，48のページ・フレームを，アロケート・ノードGETMEMORYで与え⑦，リリース・ノード FREEMEMORY で放す⑧，CPUはラウンド・ロビン(round robin) でスケジューリングされるとし，PS で近似する．最後に，サブ・モデル内におけるチェーンを定義する⑫．

モデルCSMでは，このサブモデルCSSMを，CSSM 1 として引用する⑬．また，チェーン⑭は，図より明らかにクローズド・チェーンを形成する．これで，モデルの定義が終了した．

次に，シミュレーション制御部を定義する(図6-(b))．まず，CPUにおける平均待時間の分布を得ることを宣言する⑮．また，CPUの平均待時間の信頼区間を得るのに spectral 法を用い⑯，信頼度90%とする⑱．さらに，sequential stopping rule を適用し⑲，CPUにおける平均待時間の信頼区間の幅が10%に収まったとき，シミュレーションを停止する⑳．また，信頼区間を導出するのに必要な標本値を取る時間を与える㉒．

シミュレーションを開始するためには，まず，パラメータ THINKTIME と USER に値を与える必要がある㉕(図 6-(c))．実行終了後，ユーザーは，会話形式で，必要な統計量を得る．図6-(c)では，すべてのキューの利用率㉖，CPUQ の平均待時間とその信頼区間㉗，MEMORYQ の待行列長分布㉘を出力している．シミュレーションを継続する場合には㉙で yes を入力する．また，入力パラメータの変更は，㉚で数値を実際に入力すればよい．

5. まとめ

以上のように，RESQ は，システムを資源とジョブ中心にとらえているため，自然なモデル化が可能で，初心者にとってもモデル評価のための強力な支援ツールとなり，一方，複雑なシステムに対しても，ネットワーク・モデル表現の柔軟性，多様性により，十分対処しうるものであると考えられる．

RESQ は，現在もさまざまな面でその機能向上が図られており，たとえば PET[1] のように，RESQ のプリ・プロセッサとして IBM の通信ネットワーク・アーキテクチャ SNA 専用のモ

```
setup CSM
MODEL:CSM                                   /* モデル名 Computer System Model */
① METHOD: simulation                        /* シミュレーションによる評価 */
  NUMERIC PARAMETERS: thinktime users        /* 入力変数の宣言 */
  QUEUE: terminalsq                         /* 端末の定義 */
② TYPE: is                                  /* infinite server */
  CLASS LIST: terminals
    SERVICE TIMES: thinktime    /* サービス時間は、平均thinktimeの指数分布 */
③ SUBMODEL:cssm                             /* サブモデル名(Compuetr System Submodel) */
  CHAIN PARAMETERS: interactive
  QUEUE TYPE: diskdef                       /* キュー・タイプdeskdefの定義 */
⑤   NODE PARAMETERS: servicecls             /* クラス名をパラメータにする */
④ TYPE: fcfs                                /* first come first service */
  CLASS LIST: servicecls
    SERVICE TIMES: .06
  END OF QUEUE TYPE DESKDEF
  QUEUE: memory                             /* パッシブ・キューmemoryの定義 */
    TYPE: passive                           /* キュー・タイプはパッシブ */
⑥   TOKENS: 128                             /* ページフレーム数は１２８ */
    DSPL: fcfs
⑦   ALLOCATE NODE LIST: getmemory
      NUMBERS OF TOKENS TO ALLOCATE:discrete(16.,25; 32,.5; 48,.25)
⑧   RELEASE NODE LIST: freememory
  QUEUE: cpuq                               /* アクティブ・キューCPUQの定義 */
⑨   TYPE: ps                                /* processor sharing */
    CLASS LIST: cpu
      SERVICE TIMES: .75
  QUEUE: diskq1                 /* キュー・タイプdiskdefを用いたdiskq1の定義 */
⑩   TYPE: diskdef
    SERVICECLS: disk1           /* クラス名をdisk1とする */
  QUEUE: diskq2                 /* キュー・タイプdiskdefを用いたdiskq2の定義 */
⑪   TYPE: diskdef
    SERVICECLS: disk2           /* クラス名をdisk2とする */
  CHAIN: interactive            /* サブモデル内チェーンの定義 */
    TYPE: external              /* 外部から引用することを宣言 */
    INPUT: getmemory            /* サブモデルCSSM引用時の入口名の定義 */
⑫   OUPTUT: freememory          /* サブモデルCSSM引用時の出口名の定義 */
    :getmemory→cpu→disk1 disk2; .5 .5 /* CPUからdiskへは同確率で分岐する */
    :disk1→cpu freememory; .875 .125   /* diskからは、.875,.125の割合で分岐 */
    :disk2→cpu freememory; .875 .125
END OF SUBMODEL CSSM
⑬ INVOCATION: cssm1  /* サブモデルcssmの起動 モデルCSMでは、CSSMの名で引用する */
  TYPE: cssm
  INTERACTIVE: interactive
  CHAIN:interactive                         /* チェーンの定義 */
⑭   TYPE:closed
    POPULATION:users                        /* クローズド・チェーンにおけるジョブ数 */
    :terminals→cssm1→terminals
```

図 6-(a)　RESQ による会話型コンピュータシステム（図５）のモデル記述

```
⑮ QUEUES FOR QUEUEING TIME DIST: cssm1.memory
    VALUES: 1 2 3 4 5 6 7 8          /* cssm1.memoryの平均待時間分布をとる */
⑯ CONFIDENCE INTERVAL METHOD: spectral
⑰ INITIAL STATE DIFINITION-          /* シミュレーションの初期状態の定義 */
    CHAIN: interactive
    NODE LIST: terminals       /*
      INIT POP: users
⑱ CONFIDENCE LEVEL: 90 /* percent */
⑲ SEQUENTIAL STOPPING RULE: yes
⑳ CONFIDENCE INTERVAL QUEUES: cssm1.cpuq
    MEASURES: qt /* queueing time */
    ALLOWED WIDTHS: 10 /* percent */
  INITIAL PORTION DISCARDED: 10 /* percent */
㉒ INITIAL PERIOD LIMITS-
    QUEUES FOR DEPARTURE COUNTS: cssm1.cpuq
      DEPARTURES: 10000
㉓ LIMIT - CP SECONDS: 30           /* CPU run timeの上限は30秒 */
```

図 6-(b) RESQ によるシミュレーション制御部の記述

デリング・ツールを構築しようという試みなども盛んに行なわれている.

現在，とりくまれている課題として

(1) 出力結果のグラフ処理機能の充実

現在，サポートされているのは，バッチ・モードで3270端末上へグラフ出力させる機能のみで，シミュレーション終了後，会話形態によってユーザーに必要なグラフを提供する機能が望まれる.

(2) RESQ のパーソナル・ユースとしての機能を高めるため，IBM・PC へ移植する.

(3) シミュレーション実行時のデバグ(debug)機能の充実

現在，図6の trace 指定により，ある時間帯に発生した事象がレポートされるのみで，たとえば任意の時刻，あるいはある事象が発生した時点で，ユーザーが必要とする情報を会話的に提供する機能が必要と思われる.

などがあげられる

参考文献

[1] Bharath-Kumer, K. and Kermani, P., "Performance Evaluation Tool(PET) : An Analysis Tool for Computer Communications Networks", IEEE Journal of Selected Areas in Communications, vol.1, no.1, Jan. 1984

[2] Heidelberger, P. and Welch, P. D., "A Spectral Method for Confidence Interval Generation and Run Length Control in Simulations", CACM, vol.24, no.4, pp.233-245, April 1981

[3] Lam, S. S. and Lien, Y.L., "A Tree Convolution Algorithm for the Solution of Queueing Networks", CACM, vol.26, no.3, pp.203-215, March 1983

[4] Lavenberg, S. S. and Sauer, C. H., "Sequential Stopping Rules for the Regenerative Method of Simulation", IBM J. of Res. Dev., vol.21, pp.545-556, Jun. 1977

[5] Reiser, M., "QNET4 user's guide", IBM Res. Rep., RA-71, May 1975

```
㉔  eval csm
    RESQ2 VERSION DATE: JANUARY 24, 1983 - TIME: 15:00:10 DATE 04/15/85
    MODEL:CSM
㉕  THINKTIME:16        /* パラメータの入力 */
    USERS:25
    SAMPLING PERIOD END: CSSM1.CPUQ DEPARTURE LIMIT
    SAMPLING PERIOD END: CSSM1.CPUQ DEPARTURE LIMIT
    SAMPLING PERIOD END: CSSM1.CPUQ DEPARTURE LIMIT
    NO ERRORS DETECTED DURING SIMULATION. 2135 DISCARDED EVENTS
           SIMULATED TIME: 1964.54370
                CPU TIME:      13.76
         NUMBER OF EVENTS:      43077
    WHAT:ut                 /* 利用率(utilization) */
         INVOCATION  ELEMENT     UTILIZATION
                     TERMINALSQ  0.00000
㉖       CSSM1       MEMORY      0.65356
         CSSM1       CPUQ        0.77910
         CSSM1       DISKQ1      0.31053
         CSSM1       DISKQ2      0.30919
    WHAT:qtbo(CSSM1.CPUQ)       /* CPUQの平均待時間と信頼区間 */
㉗       INVOCATION  ELEMENT     MEAN QUEUEING TIME
         CSSM1       CPUQ        0.17874(0.17133,0.18616) 8.3%
    WHAT:qld(cssm1.memory)      /* MEMORYの待行列長分布 */
         INVOCATION  ELEMENT     QUEUE LENGTH DISTRIBUTION
㉘       CSSM1       MEMORY      0:0.08349
                                 1:0.1336916
                                      :
                                 16:7.1171E-04
    WHAT:↵                  /* null reply */
㉙  CONTINUE RUN:no          /* yesと答えた場合、シミュレーションを継続する */
㉚  THINKTIME:↵             /* あるいは、入力パラメータを変更してシミュレーションの実行 */
```

図 6-(c) RESQによるシミュレーション結果の出力

[6] Sauer, C.H. and MacNair, E.A., "Queueing Network Software for System Modeling", Software-Prac. Exp., 9, 5

[7] ——, —— and Salza, S., "A Language for Extended Queueing Network Models", IBM J. Res. Dev., vol.24, pp.745-755, Nov. 1980

[8] ——, —— and Kurose, J.F., "The Research Queueing Package Version 2 : Introduction and Examples", IBM Res. Rep. RA-138, Apr. 1982

[9] ——, —— and ——, "The Research Queueing Package Version 2 : CMS User's Guide", IBM Res. Rep. RA-139, Apr. 1982

[10] ——, —— and ——, "Queueing Network Simulations of Computer Communication", IEEE Journal of Selected Areas in Communications, vol.1, no.1, Jan. 1984

[11] —— and ——, "The Evolution of the Research Queueing Package", IBM Res. Rep. RC-10376, Feb. 1984

[12] Tucci, S. and MacNair, E.A., "Implementation of Mean Value Analysis for Open, Closed Mixed Queueing Networks, "Computer Performance, vol.2, no.4, pp.233-239, Nov. 1982