対話型計算機システムシミュレータにおけるモデル記述性の高度化

野瀬純郎† 小松俊雄††

情報通信システムの高度化，複雑化が進むにつれ，性能に関する問題の早期発見と的確な対処を可能とするため，システムの企画段階から運用段階までのライフサイクル全般にわたって，正確な評価の実施が重要かつ困難な課題となってきている．シミュレーションはそれに対する有効な手段のひとつであるが，高度のスキルと多大な工数を要することが障害になり，日常的に実施されるまでには至っていない．そこで著者らは，シミュレーション専用言語からの解放と評価工数の削減を目的として，計算機システムの性能評価用シミュレータ INSYDE を開発した．INSYDE は図表形式で入力されたモデル情報からシミュレーションプログラムを自動的に生成・実行し，結果を図表形式で出力するグラフィカルな対話型ツールである．このようなシミュレータを開発する際，実用性の観点から最も重要な課題は(1)モデル記述能力と，(2)シミュレーション実行速度であり，本論文ではこれらの2点を中心に，実現方式と効果について述べる．数百～数千個のリソースで構成され，1時間に数千～数十万件のトランザクションを処理する情報通信システムの評価に INSYDE を適用し，その実用性を確認するとともに，シミュレーション専用言語により直接プログラミングする従来の方法と比較して，性能評価のための所要工数を約1/5に削減できた．

Implementing Advanced Model Description Capability of an Interactive Simulator for the Computer Systems

Junro Nose† and Toshio Komatsu††

The growing complexity of today's highly sophisticated information network systems makes it important, but also difficult, to conduct accurate performance evaluations throughout the entire system life cycle, from the system design phase to the operational stage, in order to enable problems to be discovered early and dealt with properly. Simulation techniques are potentially effective means of evaluation, but are restricted by the need for high-level skills and the great amount of time required, so that they fall short of being an every day practical reality. The authors have developed a computer system performance evaluation simulator called INSYDE, with the aim of eliminating the need for use of a specialized simulation language and reducing the number of man hours required. INSYDE is a graphics-based interactive tool which automatically creates and executes a simulation program from model data input in graphical/tabular form, and outputs the simulation results as figures and tables. In developing this simulator, the most important design issues from the standpoint of practicality were (1) the model description capability and (2) the simulation execution speed. This paper focuses on these two issues in describing the realization method and results. INSYDE was applied to some information network systems consisting of several hundreds to thousands of resources, and processing several thousand to several hundred thousand transactions per hour. Besides cofirming the simulator's feasibility, we succeeded in reducing the number of required man hours to approximately one fifth the time needed with the conventional direct programming method using a dedicated simulation language.

1. はじめに

情報化社会の進展にあわせて，情報通信システムの分野でも次々に新しいサービスが開発・導入され，システムはますます高度化，複雑化してきている．これに伴い，性能に関する設計上，製造上，あるいは運用上の問題の早期発見と的確な対処を可能とするため，システムの企画段階から開発段階を経て運用段階に至るライフサイクル全般にわたって，正確かつタイムリな性能評価の実施が重要かつ困難な課題となってきて

† NTT ソフトウェア株式会社
NTT Software Corporation
†† NTT 情報通信網研究所
NTT Network Information Systems Laboratories

いる.

現実の情報通信システムを対象に，性能・負荷・リソース間のバランス分析を行い，目標性能達成度の確認，ボトルネックの抽出，適正なリソース量の算出，処理能力限界の推定，業務量増加に伴う性能劣化の予測，性能改善策の提案などを行うためには，待ち行列モデルによる解析では，扱える規模と得られる結果の精度に限界がある．これに対して，シミュレーションは非常に有効な手段のひとつである[1)~4)]が，GPSS[6)]，SLAM Ⅱ[7)]，SIMSCRIPT[8)] などの専用言語でシミュレーションプログラムを記述する従来の方法では，これらの言語やモデリング技法の習得が必要になるとともに，モデルが大規模かつ複雑になると，プログラミングおよび検証の作業量が大きくなる．さらに，問題の認識が深まるにつれモデルは進化・発展していくため，それに対応したプログラムの変更・修正が頻発に起こり，作業量も膨大になる．また，システム開発者／運用者などの評価者以外の第三者にとって，シミュレーションの正当性を確認することは困難なため，結果に対する不信がいつまでも残ることになるなどの問題がある．

これらの問題を解決するため，著者らは情報通信システムの分野を適用対象として，リソース，処理フロー，ワークロードなどのモデル条件を図表形式で記述し，それらの情報をもとにシミュレーションプログラムを自動的に生成・実行し，結果を図表形式で出力する，グラフィカルな対話型の性能評価用シミュレータ INSYDE (Integrated Support Tool for System Design and Evaluation) を開発した．

開発に当たっては実用性を重視し，(1)高い記述能力の実現と，(2)高速なシミュレーション実行を重点課題とした．前者については，①モデル条件は図および表で定義する，②計算機の内部動作に特有な制御に対応して 50 余種の処理フロー記述用ノード群をサポートし，③ノードのパラメータは固定的なもののほかに変数でも指定可能とする，④変数と条件の個々の対応付けは別に表形式で定義する，などにより実現した．また，後者については，①処理フロー記述用ノード群はすべて事象処理ルーチンとして用意し，②各ルーチンはシミュレーション専用言語ではなく，FORTRAN で記述する．③これらは INSYDE の立ち上げ時にプリコンパイルしておき，④モデリングの段階では接続関係の指定と，データの設定のみとする，などにより実現した．この結果，シミュレーションプログラムを直接 SLAM Ⅱ で作成した場合に比較して，実行時間は 0.5～1.5 倍となり，モデルが大規模，複雑になるほど高速化の効果が大きいことを確認した．また，モデル定義の際に入力した図表は，そのままシステム開発者／運用者などの評価依頼者とのレビューに使用し，モデル条件の確認ができた．

以下では，第2章で研究の背景として情報通信システムの特質と，関連する研究の状況を紹介し，第3章で INSYDE の概要として，システム構成，機能，およびモデリングについて説明する．第4章でモデル記述能力の向上のための方法について述べ，第5章でシミュレーション実行の高速化手法とその効果について示す．

2. 研究の背景

2.1 評価対象としての情報通信システムの特質

INSYDE の適用対象として想定している情報通信システムの，シミュレーションの観点からの特質は以下のようなものである．

①プロセッサ，ファイル記憶装置，回線，端末などのシステムを構成するハードウェア・リソースの数が膨大である（数百～数万台）．

②さらにこれらの内部に，数十～数百個のタスク，テーブル，ファイル（データ）などのソフトウェア・リソースが格納される．

③システム内を流れるトランザクションが，多種（数十～数百種），大量（数千～数十万件／時間）である．

④割り込み，排他制御，タスク間通信などの，計算機特有の処理がある．

⑤同種のトランザクション（処理の流れは同じ）でも，アクセスするリソースの番号，処理する量などがトランザクションごとに異なる．

⑥処理時間がリソース間で4桁以上（例えば ms～s）異なるため，精度に注意する必要がある．

以上のような，量の問題，複雑さの問題，煩雑さの問題を解決するモデル記述能力と，シミュレーション実行速度の実現が求められる．

2.2 関連する研究

グラフィカルな対話型シミュレータは種々の分野で発表されており，通信ネットワークの分野では PET[9)]，RESQME[10)]，TEDAS-S[11)]，TOPNET[12)]，NEST[13)] などがあるが，いずれも計算機の内部処理モデルを詳細に記述することは困難である．情報通信システムの

分野では AT & T Performance Analysis Workstation[14], CAB[15], SES/workbench[16] などがある．これらについても計算機特有の内部処理である割り込み処理，タスク間通信などが簡単に記述できず，複雑な処理モデルは記述不能かまたはユーザのオウンコーディングによるなど，記述能力に難点がある．著者らの調査した範囲では，大規模な情報通信システムを対象とした，モデル記述能力およびシミュレーション実行速度の点で実用性のあるツールは見当たらない．

3. INSYDE の概要

3.1 構成と機能概要

INSYDE の構成は**図1**に示すとおりである．

モデル定義情報取得部（MDEF）では，入力された図表形式のモデル条件の形式チェックを行い，シミュレーションプログラムの生成に適した形式に変換する．処理フローの入力画面の例を**図2**に示す．処理フロー中の各記号はノードと呼ばれ，すべてアイコンとして用意されている．操作者はノード一覧メニューから所定のノードを選択し，画面の所定の位置に配置し，ガイダンスを参照しながらパラメータを設定する．ノード間を結ぶ線分（アーク）はトランザクションの流れを示すもので，誤り（途中切断，ループなど）があればエラー表示される．

シミュレーションプログラム自動生成部（SGEN）では，MDEF で変換したデータにもとづき，シミュレーションプログラムを生成する．同時に，図表間の対応関係について論理チェックを行う．生成法の詳細については後述する．

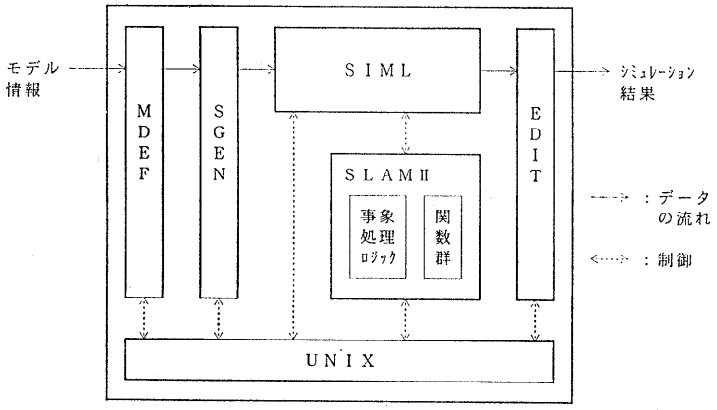
シミュレーション実行部（SIML）では，SGEN で生成したシミュレーションプログラムを実行し，全リソースおよびトランザクションの種類毎に時間統計情報，待ち統計情報を自動的に取得する．一回の実行で，統計情報を一定時間毎に複数回取得することが可能であり，シミュレーションの安定性を確認することができる．また，モデルの検証を容易化するため，事象毎にトランザクション番号，リソースの処理量，特定エリアの内容などの詳細なトレース情報を取得する．トランザクションは処理の途中で分裂することが多いので，その区別を可能とするため，最初の発生番号のほか，タスク内での分裂番号[5]およびタスク間での分裂番号を付与している．なお，ここでは事象中心（event oriented）の離散型シミュレーション[17]を採用している．

統計情報編集部（EDIT）では，SIML で取得した統計情報を蓄積し，①シミュレーションの実行毎に全統計情報を表示する，②指定された情報のみを表形式に編集・表示する，③編集された表の行項目と列項目の一覧メニューから選択された組合せをグラフ表示する．②，③については，複数回分のシミュレーションの結果をまとめて行うことも可能である．また，時系列に出力されたトレース情報の追跡を容易にするため，トランザクション番号，処理フロー名，ノード名，リソース名などを検索条件として，これらの論理式でトレース情報の検索・表示を行うことができる．

事象カレンダをもとに，シミュレーション全体の実行を管理する事象処理ロジック，および待ち行列へのトランザクションの設定や取り出し，トレースなどの関数については SLAM II の機能を活用した．

3.2 モデリング

INSYDE では，システムの動作を，トランザクションがいくつかのノードと，それらを結ぶアークで構成されるネットワーク上を動くモデルとして表現する．モデルの作りやすさ，柔軟性，拡張性などを考慮して，具体的にはリソース，処理フロー，トランザ



MDEF ：モデル定義情報取得部
SGEN ：シミュレーションプログラム 自動生成部
SIML ：シミュレーション実行部
EDIT ：統計情報編集部
SLAMII：汎用シミュレータ
UNIX ：OS

図1 INSYDE の構成
Fig. 1 INSYDE system structure.

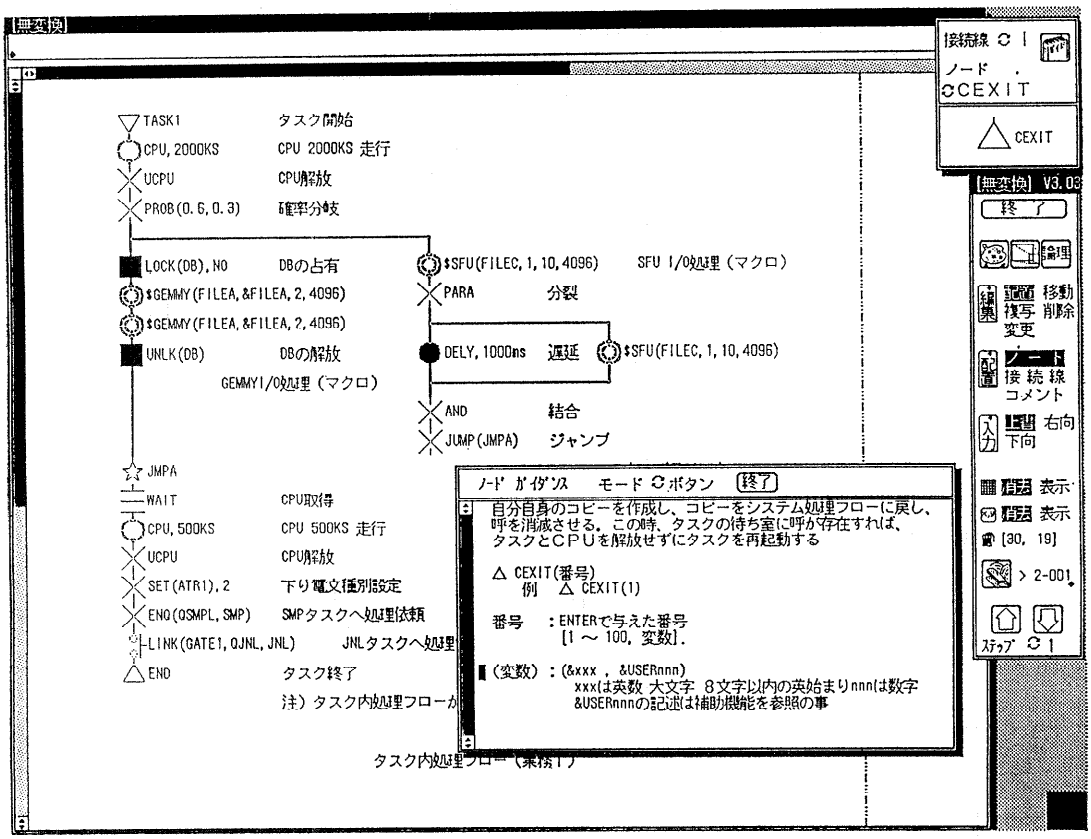


図 2 処理フローの入力画面例
Fig. 2 An input window for transaction flow.

クションの3種を定義することによりモデリングを行う．

（1）リソースの定義

システムを構成するハードウェア（CPU，ファイル装置，入出力装置，通信回線，端末など）およびソフトウェア（タスク，テーブル，データベース，ファイル，セマフォなど）のリソースについて，**表1**に示すような項目を定義する．これらは INSYDE で標準的に用意している枠組みであり，モデルの詳細度に応じて任意に省略可能である．

（2）処理フローの定義

処理フローは**表2**に示す 52 種類のノードを使用して記述する（図2参照）．図2中の各記号と表2のノードは1対1に対応する．標準のノードでは記述が困難あるいは煩雑な場合，ユーザが新たなノードを定義することができる．計算機システム内のトランザクションの流れは分岐，分裂，割り込みなどを繰り返すため非常に複雑なものとなり，すべてをひとつの平面で記述するには限界がある．そこで INSYDE では，いくつかの階層に分けて記述[5]することとした．通常は，トランザクションの発生から消滅までの処理の流れ全体を示す「システム処理フロー」，個々の業務処理を実現する「タスク処理フロー」，およびまとまった処理で何度も繰り返される「マクロ処理フロー」の3階層で記述する．

（3）トランザクションの定義

トランザクションの種類毎に発生分布，発生件数，発生開始時刻，優先順位，トランザクションの流れるシステム処理フロー名などを定義する．発生分布は，指数分布，一様分布，正規分布などの指定以外に，ユーザが任意の関数をプログラムで記述することができる．

4. 記述能力の向上方法

モデルの記述能力を考える場合，どこまで複雑な論理が書けるかという問題と，煩雑な条件がどれだけ簡

表 1 リソースの主な定義項目
Table 1 Main model definition items for resource in INSYDE.

種類		定義項目	内容
ハードウェアリソース		装置名称	
		性能	処理速度／所要時間
		属性値	条件により性能が異なる場合に属性値で区別
		装置台数	
		多重度	同時に処理可能な数
		起動契機	サービス開始条件
		待ち容量	有限の場合，溢れは呼損扱い
		待ち選択規則	待ち室のトランザクションを取り出す規則
ソフトウェアリソース	タスク	タスク名称	
		タスク枚数	タスク種別毎
		割り込み	CPU に対する強制割り込みの有無
		CPU 補足優先度	CPU を確保するためのタスク間の優先順位
		起動契機	サービス開始条件
		待ち室名称	
		待ち容量	
		待ち選択規則	待ち室のトランザクションを取り出す規則
		窓口数	同時に処理可能なタスク枚数
		タスク捕足優先度	タスクを確保するための優先順位
	テーブル	テーブル名称	ロック制御やゲート制御などの対象テーブル名
		多重度	
	ファイル	ファイル名称	

表 2 処理フローの記述ノード
Table 2 Function of description nodes for transaction flow.

分類		種類	機能
時間遅延ノード		8	リソースを確保した状態，あるいは確保しない状態でトランザクションを遅延させる．
制御ノード	開始，終端	8	処理フローの開始と終端を示す．終端においてはトランザクションの消滅以外に，処理の連続再開，他処理へのキューイングが可能である．
	分岐	13	無条件分岐と種々の条件分岐ができる．テーブル，ワークエリアに対し条件の設定ができる．
	処理依頼	6	システム処理とタスク処理間およびタスク処理相互間の処理依頼を行う．
	リソース管理	9	リソースの確保／解放を行う．複数のリソースを同時に確保／解放する事も可能である．
	分裂，統合	4	トランザクションを複数個に分裂させて，同一方向／複数方向に進める．分裂したトランザクションは全て統合／一部統合の指定が可能である．
ユーザノード		1	ユーザが独自にノードを定義する．
マクロノード		1	あらかじめ定義したマクロ処理フローを呼び出す．
測定ノード		2	任意の測定区間の統計情報を取得する．

略に設定できるかという問題の2面がある．これらの問題解決のために，INSYDE で実現した方法の主要部を以下に述べる．

4.1 複雑な条件の表現

(1)タスク構成の定義

ソフトウェアリソースのひとつであるタスク（またはプロセス）に関しては，性能の向上，プログラムの開発・維持管理の容易化などのため，様々な構成・制御方式が提案・実現されている．これらの特徴を損なうことなくモデリングを可能とするため，INSYDE では特にタスクの定義について詳細化をはかった（表1参照）．これにより，例えば**図3**に示すような種々

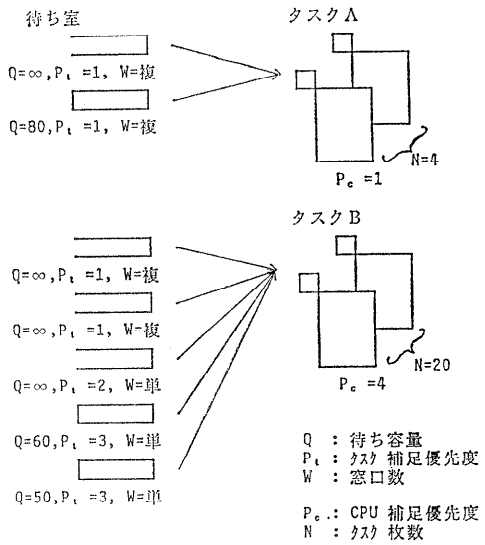


図 3 タスクの構成例
Fig. 3 An example of task formation.

の特性を持つ待ち室群とタスク群の対応付けが，表のみで定義できる．

(2)計算機特有の処理

すでに述べたように，処理フローは表2のノードを用いて記述するが，計算機特有の処理を表現する制御ノードの例を示す．

①排他制御：LOCK ノードと UNLK ノードにより，リソースのロックおよび解放を制御する．複数のリソースの制御を同時に行うことも可能である．複数のリソースの同時ロックに失敗した場合は，確保が可能であった一部のリソースもいったん解放し，改めて対象となるリソースのいずれかが解放される毎に，同時ロックの可否判定を自動的に行う．

②割り込み処理：PRE ノードにより，実行中のタスクの処理を中断し，CPU を強制的に確保して割り込み処理を行った後 CPU を切り離し，中断したタスクの処理を自動的に再開始する．

③タスク間通信：タスク間通信とは，タスクの間で情報のやりとりを行うことで，LINK ノードと LON ノードにより記述する．**図4**に示すように，複数のタスクの複数の箇所で行うタスク間通信に対しても，各通信毎に通信元を管理しており，LON ノードは応答先をパラメータとして記述する必要はない．

④タスクの連続起動：タスク処理が終了すると，通常は CPU を解放してタスクディスパッチャに制御を渡すが，あるトランザクションのタスク処理の終了時，次のトランザクションが該タスクの待ち室にあれば，処理効率の向上のため CPU を解放せずに，連続してタスクの処理を行う場合がある．このような指定を CEND ノードにより行う．

(3)システム情報の参照

シミュレーション実行中のシステム情報が，IF ノードにより任意の時点で参照可能であり，さまざまな条件判定に使用できる．参照可能な情報には，システム全体でひとつのグローバル情報と，トランザクション毎に持つローカル情報がある．前者はリソース毎の使用多重度，待ち数などである．また，後者はユーザ独自のワークエリアの内容，各ノードの変数の内容などである．

(4)ユーザルーチン

標準の図表形式では表現が困難な条件については，ユーザルーチンとして FORTRAN で記述することができる．ユーザルーチンが使用できる箇所は処理フロー中のユーザノードのみならず，各ノードのパラメータ，リソース定義表の待ち選択規則，トランザクション定義表の発生分布への組み込みが可能であ

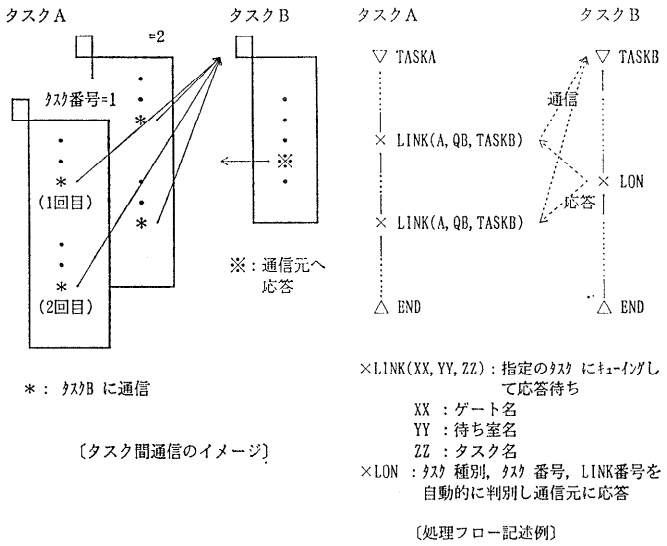


図 4 タスク間通信の記述例
Fig. 4 An example of description for communication between tasks.

表 3 プログラミング時に参照可能な INSYDE の情報
Table 3 INSYDE informations be able to refer in programming.

種　　類	主 な 内 容
定義情報	リソース トランザクション
システムの状況	リソース待ち数 リソース待ち時間 リソース使用多重度 リソース当たりの処理件数 処理フローの実行時間 処理フロー当たりの処理件数 シミュレーション時刻
トランザクションの情報	発生番号 確保中のタスク名 変数の内容 ワークエリアの内容

る．プログラミングに際しては，**表 3** に示すような INSYDE が管理・取得する情報を関数として参照できるほか，各種統計情報の取得関数，テーブルへのデータ設定関数が使用可能である．これらにより，例えば**図 5** に示すように，ある複数の装置に関して，トランザクションの滞留数のしきい値を超えるか否かを判定し，その結果を後続の処理の制御に利用するため，自分自身のワークエリアに結果を設定する，というような複雑な条件が記述できる．

〔内容〕 WS（装置番号 &N）での滞留数および CPU での滞留数をそれぞれしきい値と比較し，後続処理の流れを制御するノード

```
      SUBROUTINE     USREVT(A, FLG)

#INCLUDE   'USR.INC'                    :宣言文の展開
       CHARACTER*8   B, C

       D  =  A                          :自分自身のﾄﾗﾝｻﾞｸｼｮﾝを指定
       B  = "N"
       CALL  GETVAR(D, B, VAR)          :変数&Nの値を参照

       C  = "WS"
       E  =  VAR                        :変数&Nの値を指定
       IF(DWBSY(C, E).LT.50) GO TO 10   :WSでの滞留数のチェック

          C  = "CPU"
          IF(UWBSY(C).LT.10) GO TO 10   :CPU での滞留数のチェック

             WRK1 =0                    :比較結果をﾜｰｸｴﾘｱに設定
                GO TO 20

 10          WRK1 =1
 20    FLG=1                            : FLGはﾄﾗﾝｻﾞｸｼｮﾝの消滅ﾌﾗｸﾞ
       RETURN
       END
```

図 5 論理の深い条件のプログラミング例
Fig. 5 An example of programming for deep-logical condition.

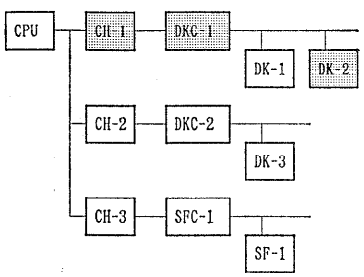
4.2 煩雑な条件の簡略化

（1）ファイルによるアクセスパスの間接指定

情報通信システムにおいては，ファイルへのアクセスが頻繁に生じ，その都度処理フローにファイルへのアクセスパス，すなわちチャネル装置，ファイル記憶制御装置およびファイル記憶装置を指定する必要があり，頻雑となる．そこで，アクセスパスを構成する装置群の接続関係を示すパス構成図と，ファイルの格納状況を示すファイル表を別に定義することにより，処理フローではファイル名を指定するだけで，間接的にアクセスパスが指定できるようにした．記述例を**図 6** に示す．線分は装置間の接続を表し，途中切断，二重定義などはチェックアウトされる．

（2）ノードのパラメータへの変数導入

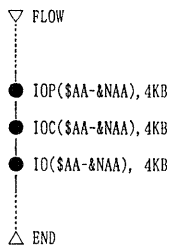
処理の流れが同じでも，トランザクションによってノードのパラメータの値（例えばアクセス装置番号，転送データ量など）が異なる場合，個々に処理フローを記述しなければならないとすると膨大な量になる．そこで，このようなパラメータを変数として定義可能とすることにより，ノードの種類が同じあればひとつの処理フローで記述できるように



〔パス構成図〕

〔ファイル表〕

	ﾌｧｲﾙ名	ﾌｧｲﾙ名	ﾌｧｲﾙ名
DK-1	AA-1	BB-1	CC-1
DK-2	AA-2	BB-2	CC-2
DK-3	AA-3	BB-3	CC-3
SF-1	AA-4	BB-4	—



例．ファイル名がAA-2(&NAA=2)ならばCH-1, DKC-1, DK-2 をアクセスパスとする

〔処理フロー記述例〕

図 6 ファイル名によるアクセスパスの間接指定
Fig. 6 Indirect appointment of access path by file name.

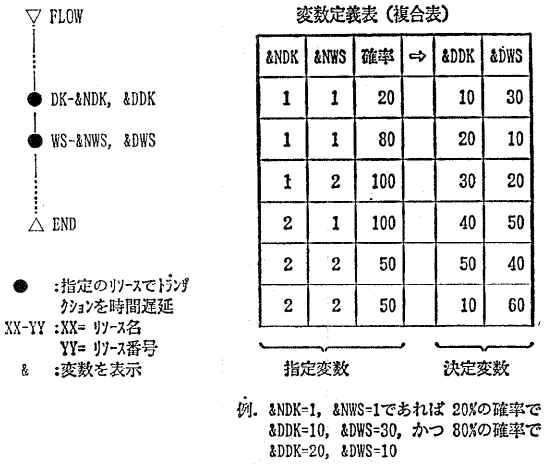


図 7 ノードパラメータの変数指定の例
Fig. 7 A definition example of node parameters by variable.

した．

（3）静的変数の表形式による定義

前（2）項で述べたノードパラメータの変数には，トランザクションが実際に処理フロー上を流れて初めて決定されるものと，モデルが定義された時点で一意に決定されるものがある．ここでは前者を動的変数とよび，実行条件との関係はユーザルーチンにより定義する．また，後者を静的変数とよび，実行条件との関係は表形式で定義する．表の種類には，条件との対応付けを確率で行う確率対応表，各変数の値の組合せで行う組合せ対応表，確率対応表および組合せ対応表が結合された複合表の3種がある．記述例を図7に示す．

5. シミュレーションプログラムの生成法および実行速度

5.1 シミュレーションプログラムの生成法

シミュレーション実行時間の短縮は，シミュレータの実用性の観点から最も重要な課題のひとつである．著者らは INSYDE の開発に当たり，グラフィカルな HMI を持たせながら，かつシミュレーションの実行速度は広く使用実績のある SLAMII と同程度を目標とした．このため，図表形式で入力されたモデル条件からシミュレーションプログラムを生成するに際して，SLAM ネットワークへの展開を経ないこととし，以下のような手法を採った．なお，シミュレーションプログラムはトランザクションの種類ごとに生成する．

（1）ノードの事象処理ルーチン化

シミュレーションは事象（システムの状態に何らかの変化をもたらす出来事）中心型とし，処理フロー記述用ノード群はすべて事象処理ルーチンとして，SLAM IIのベース言語である FORTRAN で作成する．ノード数は 52 種であるが，ルーチン内での条件判定回数の削減をはかるため，172 個の事象処理ルーチンに展開した．なお，事象をカレンダに従って早いもの順に実行し，シミュレーション全体を制御する事象処理ロジックについては，3.1 節で述べたように開発効率を考えて SLAM II の機能を利用することとした．SGEN で生成されたプログラムの展開イメージと，SIML でのシミュレーション実行の流れを図8に示す．

（2）静的変数定義表の結合

すでに述べたように，ノードパラメータの静的変数

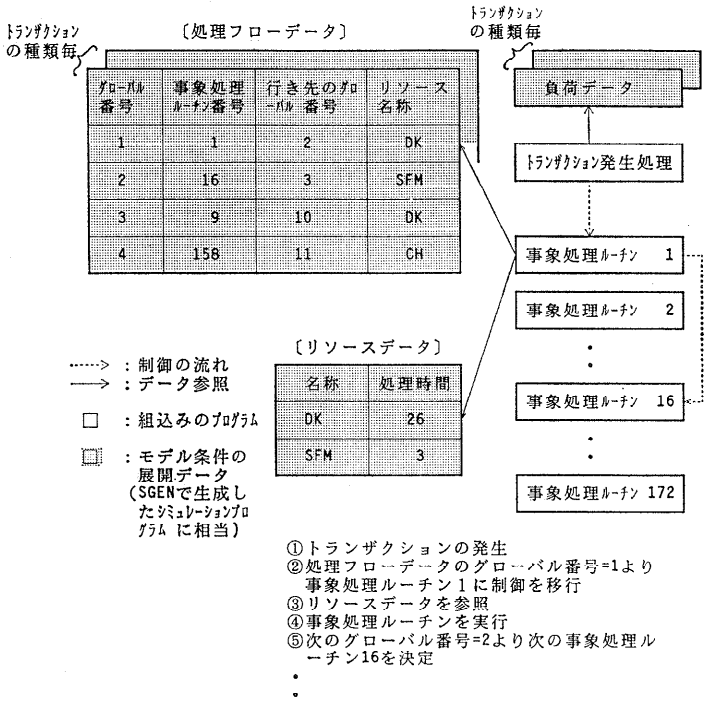


図 8 実行時のシミュレーションの流れ
Fig. 8 Simulation flow at simulation execution.

〔結合前〕

確率	⇨	&A
40		1
60		2

確率	⇨	&B
30		1
70		2

確率	⇨	&C
50		1
50		2

&A	&B	⇨	&D
1	1		1
1	2		2
2	1		3
2	2		4

&C	確率	⇨	&E
1	40		1
1	60		2
2	100		3

2つのグループに結合

〔結合後〕

確率	⇨	&A	&B	&D
12		1	1	1
28		1	2	2
18		2	1	3
42		2	2	4

確率	⇨	&C	&E
20		1	1
30		1	2
50		2	3

図 9 静的変数定義表の結合例
Fig. 9 A combination example of static variable definition tables.

は確率対応表，組合せ対応表および複合表で定義する．これらの定義表で，条件を指定する側（表の左側）を指定変数，値が決定される側（表の右側）を決定変数とよぶと，SGEN では決定変数が指定変数になるものについてはグルーピングし，ひとつの表に結合する．これにより，シミュレーション実行時の変数定義表の検索回数を削減することができる．結合の例を**図 9** に示す．なお，同じ指定変数を複数の定義表で何度も使用することが可能である．

（3）静的変数の事前決定

静的変数はトランザクションが実際に該当ノードに到達する前に一意に決定可能なため，SGEN ではすべての静的変数を計算し，その値を変数定義表に設定する．したがって，シミュレーションの実行にあたり，SIML はトランザクションを発生するごとに，該トランザクションに関する静的変数を事前に変数定義表から一組選択するのみでよい．

（4）マクロ処理の埋め込み

シミュレーションの流れは図 8 に示したとおり処理フローデータの内容に従うが，MDEF でマクロとして定義された処理は，SGEN で処理フローデータに展開し，マクロの呼び元に埋め込む．このため，SIML ではマクロであることを何ら意識することがなく，そのための判定を不要にしている．

5.2 シミュレーションの実行速度

（1）評価用モデル

シミュレーションの実行速度の評価のため，同一のモデルについて INSYDE と SLAM II で記述し，実行した．参考として，同一処理に関する両者の記述例を**図 10** に示す．評価用モデルの概要は**表 4** のとおりである．システム A，B，C とも NTT のオペレーションシステム（OLTP）であり，A は公衆ネットワークに対し，B は専用ネットワークに対し，それぞれ申込から保守までの業務を管理する大規模なシステムである．C は料金関連業務を集約する中規模なシステムである．D は比較的小規模な試験用 OLTP システムである．モデル作成者はシステムごとにそれぞれ異なる．

（2）評価結果

シミュレーションの実行時間比（P）を

$$\text{実行時間比 P} = \frac{\text{INSYDEの実行時間}}{\text{SLAM II の実行時間}}$$

とすると，**図 11** に示すように 0.5〜1.5 となった．

表 4 シミュレーション実行速度の評価用モデル
Table 4 Estimation models for simulation execution speed of INSYDE and SLAM II.

システム名	主なシステム条件				モデル作成規模	
	業務パターン数	装置数	タスク数	平均 IO 回数／業務	INSYDE のノード数（別掲：FORTRAN ステップ数）	SLAM II のステートメント数*1（別掲：FORTRAN ステップ数）
A	8	6種 20台	7種 25枚	37回	675 (50)	1,062 (342)
B	7	7種 26台	6種 14枚	18回	1,026 (181)	2,105 (455)
C	2	4種 9台	3種 32枚	22回	157 (0)	215 (125)
D	1	5種 32台	2種 101枚	12回	108 (58)	189 (194)

*1 リソース名称，多重度などを定義するリソース宣言文は除く

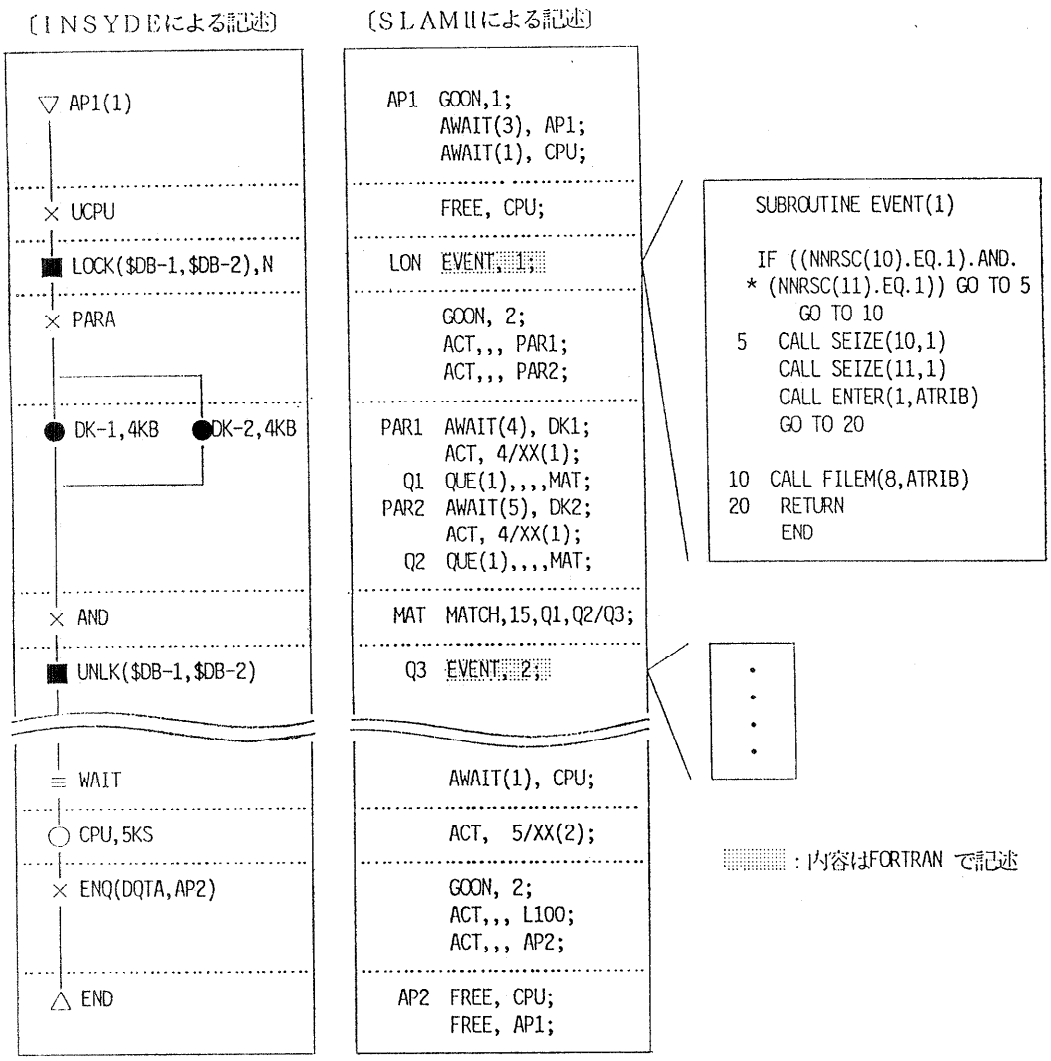


図 10 INSYDE と SLAM II の記述例
Fig. 10 An example of model description by INSYDE and SLAM II.

INSYDE が SLAM II よりも高速となる要因として想定できるのは，INSYDE では，①計算機特有の内部処理をシミュレートするためのノードを豊富に用意し，それらをすべて事象処理ルーチンとして FORTRAN で作成した，②静的変数定義表の結合と値の事前決定，マクロ処理の埋め込みなどにより，シミュレーション実行時の条件判定回数を削減した，③複数の静的変数定義表を可能な限り事前に結合し，シミュレーション実行時の検索回数を削減したことなどである．

一方，SLAM II より遅くなる要因としては，INSYDE では，リソースデータを処理フローデータとは別に定義しているため，シミュレーション実行時にその都度参照することによると想定している．

これらの要因が，それぞれどの程度作用しているのか定量的に明らかにすることは困難であるが，基本的には，SLAM II が汎用言語であり，計算機処理向けでないため，種々のルーチンをユーザが独自に作成することになり，その中で条件判定が多数繰り返されることによると考えられる．そこで，SLAM II で記述したシミュレーションプログラムの条件判定率(Q)を

$$\text{条件判定率}Q=\frac{\text{条件判定ステートメント数}}{\text{ダイナミックノード数}}$$

表 5 評価用モデルの条件判定率
Table 5 Condition decision rate in estimation models of SLAM Ⅱ.

システム名	SLAM Ⅱ		INSYDE	条件判定率
	業務当たりのルーチン呼出し回数（平均） ①	ルーチン当たりの条件判定ステートメント数（平均） ②	業務当りのダイナミックなノード数（平均） ③	①×②／③
A	6.6	46	377	0.81
B	7.8	33	470	0.55
C	23.3	9	428	0.49
D	8.4	10	211	0.40

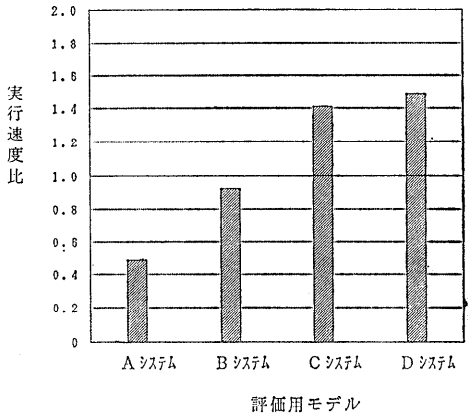
とすると，**表 5** のようになる．ここで，ダイナミックノード数とは，業務当たりの INSYDE の実行ノード数であり，正規化のために使用した．本来は SLAM Ⅱ のダイナミックステートメント数を使用すべきであるが，求めることが困難なため，代用したものである．各評価用モデルシステムの条件判定率 Q と，前述の実行時間比 P は非常に相関が強く，相関係数は -0.95 である．すなわち，Q が大きいモデルについては，INSYDE のシミュレーション実行速度の高速化効果が大きく，前記の想定が確認できた．評価の内容，モデルの精度などにより一概には言えないが，大規模で複雑なシステムほど，Q の値が大きくなる傾向にあるので，INSYDE の高性能化の効果が大きくなると言える．

6. おわりに

本論文では，シミュレーション専用言語からの解放と評価工数の削減を目的として，情報通信システムを適用対象に開発した性能評価用シミュレータ INSYDE について，モデル記述能力とシミュレーション実行速度を中心に述べた．

モデル記述能力については，①グラフィカルな HMI，②詳細なリソース特性定義表，③豊富な処理フロー記述用ノード群，④ノードパラメータへの変数の導入，⑤ユーザルーチンなどにより，複雑な条件の表現および煩雑な条件の簡略化を可能とした．

シミュレーション実行速度については，



（実行速度比＝INSYDE の実行時間／SLAM Ⅱ の実行時間）

図 11 INSYDE と SLAM Ⅱ の実行速度の比較
Fig. 11 Comparison of execution speed between INSYDE and SLAM Ⅱ.

①ノードの事象処理ルーチン化，②静的変数の事前決定と定義表の結合などにより高速化をはかり，大規模，複雑なモデルではシミュレーション専用言語 SLAM Ⅱ の 2 倍程度の実行速度を実現できた．

設計段階あるいは稼働中の様々な情報通信システムに INSYDE を実際に適用して性能評価を行い，モデ

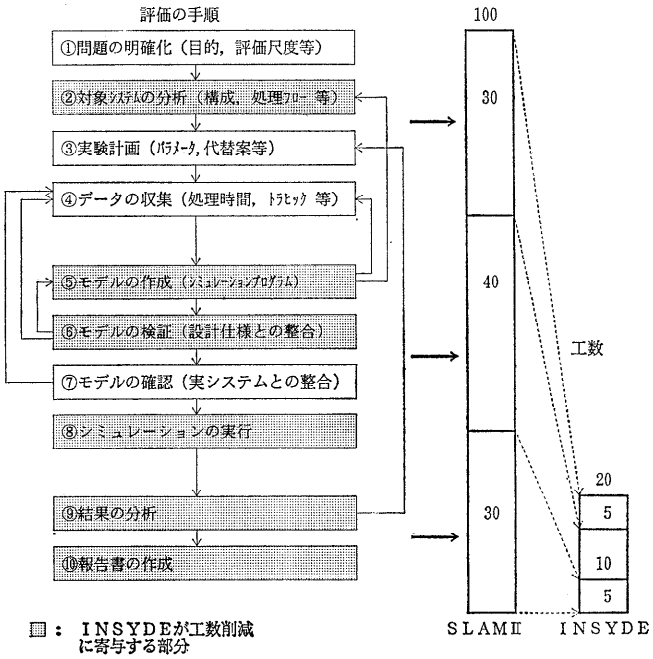


図 12 評価工数の削減効果
Fig. 12 Comparison of performance evaluation period between SLAM Ⅱ and INSYDE.

ル記述能力，操作性，シミュレーション実行速度，デバグ支援機能などに関してその実用性を確認するとともに，SLAM II を使用した従来の方法に比較して評価工数を約 1/5 に削減できた．これは，以下のような効果が相乗された結果による．①従来，漠然としていたモデリングに必要な情報が明確にされた．②モデル条件を図表形式で記述することにより，そのままの形で評価依頼者と評価者との情報交換が可能となり，改めて確認用の資料を作る必要がない．③評価が進むに従い問題の認識が深まり，評価の手順が**図 12** のように何度も繰り返されても，モデルの作成・変更が容易に行える．④シミュレーション結果の統計情報の編集・出力が容易なため，結果の分析が簡単に行える．⑤トレース機能の充実により，モデルの検証が容易である．

謝辞 本研究の機会を与えてくださった NTT 情報通信網研究所の石野福彌前所長（現 NTT ソフトウェア株式会社・事業開拓室長），和佐野哲男基本アーキテクチャ研究部長に感謝いたします．また，INSYDE の開発・評価に協力していただいた関係各位，ならびに実システムのモデル条件を提供し，有益な討論をしていただいた NTT 情報システム本部，通信ソフトウェア本部の多くの関係各位に深謝いたします．

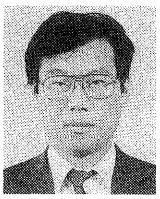
参 考 文 献

1) 野瀬純郎：シミュレーション技術の情報通信システムへの適用，計測と制御，Vol. 32, No. 2, pp. 138-141 (1991).
2) 野瀬純郎：情報通信システムのシミュレーション，精密工学会誌，Vol. 58, No. 7, pp. 1129-1123 (1992).
3) 小松俊雄，野瀬純郎：メッセージ交換型システムのスループット評価，情報処理学会研究報告，OS-46-2 (1990).
4) 小松俊雄，野瀬純郎：ネットワークサービスシステムのレスポンスタイム評価，情報処理学会研究報告，OS-50-9 (1991).
5) 小松俊雄，野瀬純郎：計算機システム性能評価用シミュレータ INSYDE，情報処理学会研究報告，OS-54-7 (1992).
6) Henriksen, J. O.: The GPSS/H User's Manual, Wolveriue Software, Falls Church, Va. (1979).
7) Pritsker, A.: Introduction Simulation and SLAM II, John Wiley and Sons (1986).
8) Mullarney, A.: SIMSCRIPT II. 5, *Programing Language*, CACI (1983).
9) Bharath-Kumar, K., Kermani, P.: Performance Evaluation Tool (PET): An Analysis Tool for Computer Communication Networks, *IEEE JSAC*, Vol. SAC-2, No. 1, pp. 220-225 (1984).
10) Gordon, R., Macnair, E., Welch, P.: Examples of Using The Research Queueing Package Modeling Environment (RESQME), *Proceedings of the* 1986 *Winter Simulation Conference*, pp. 494-501 (1986).
11) 住田修一，小沢利久，稲守久由：シミュレーション支援システム TEDAS-S, NTT R & D, Vol. 38, No. 12, pp. 1519-1528 (1989).
12) Marsan, M. A., Balbo, G., Bruno, G., Neri, F.: TOPNET: A Tool for the Visual Simulation of Communication Networks, *IEEE JSAC*, Vol. 8, No. 9, pp. 1735-1747 (1990).
13) Dupuy, A., Schwartz, J., Yemini, Y., Bacon, D.: NEST: A Network Simulation and Prototyping Testbed, *Communication of the ACM*, Vol. 33, No. 10, pp. 64-74 (1990).
14) Melamed, B., Morris, R. J. T.: Visual Simulation: The Performance Analysis Workstation, *IEEE Computer*, Vol. 18, No. 18, pp. 87-94 (1985).
15) 岡田義広，田中 譲：視覚的シミュレータの開発支援システム，情報処理学会論文誌，Vol. 32, No. 6, pp. 766-776 (1991).
16) SES/workbench 概説書，Scientific and Engineering Software, Inc. (1991).
17) 森戸 晋，相沢りえ子：SLAM II によるシステム・シミュレーション入門，構造計画研究所，東京 (1986).

(平成 5 年 1 月 11 日受付)
(平成 5 年 7 月 8 日採録)

野瀬 純郎（正会員）

1945 年生．1967 年神戸大学工学部電気工学科卒業．1969 年同大学院工学研究科電気工学専攻修士課程修了．同年，日本電信電話公社入社．以来，論理回路の故障診断，大型情報処理システム，通信処理システム，システム評価方式の研究実用化に従事．現在，NTT ソフトウェア株式会社開発合理化推進本部担当部長．



小松 俊雄（正会員）

昭和 25 年生．昭和 47 年九州工業大学電気工学科卒業．昭和 49 年同大学院電気工学科修士課程修了．同年日本電信電話公社横須賀通研入所．以来，主に DIPS 本体系装置の研究実用化，並列処理計算機，システム性能評価の研究に従事．現在，NTT 情報通信網研究所にて，システム品質に関する評価方式の研究を行っている．