B-006

UML MARTE Profile を用いた性能シミュレーション手法の提案
An Approach of Performance Analysis with UML MARTE Profile

磯田 誠† 田村 直樹†
Makoto Isoda Naoki Tamura

1. はじめに

近年，社会活動での組込みシステムの重要性が高くなっており，システムが大規模化・複雑化している．さらに，業務の拡大やユーザニーズの向上により，多機能化・高性能化の要求が高まっている．

しかし現時点では，機器単体のハードウェア性能は見積もれるが，ソフトウェアも組み合せた性能検証手法は確立していない．そのため，実システムを対象にした性能のモデル化が困難，高度化する要求に対応した性能検証のアルゴリズムの修正が困難といった課題がある．

本稿では，これらの課題に対して，システム設計結果に性能検証の数値情報を記入し，シミュレーションコードを作成して検証する手法を提案する．また，基本的な構成のサーバマシンを題材に，適用例と評価結果を示す．

2. 性能検証の関連技術

本章では，性能検証に関連する技術の概要を示す．

2.1 組込み向けリアルタイム性の UML プロファイル

MARTE Profile（UML Profile for Modeling and Analysis of Real-Time Embedded Systems）は，組込みシステム開発での性能分析に用いる標準記法である[1]．分析情報の記法を規定しているが，具体的な検証手法は特定していない．

2.2 LQN (Layered Queuing Network)

待ち行列ネットワークを拡張した LQN（Layered Queuing Network）は，システムの振舞いを階層的に記述するモデルであり，データ／制御フローが混在した動作を記述できる[2]．シミュレーションにより性能検証できるが，モデル記法が固有のため実開発で普及していない．

2.3 離散系イベントシミュレーション

離散系イベントシミュレーションは，因果関係に従った離散的なイベントの発生を，計算機上で模擬するための処理系である．既存の処理系を表 1に示す．

表 1 離散系イベントシミュレーションの処理系

項目	言語	ライセンス	ドキュメント
SimPy [3]	Python	LGPL	豊富
JSL [4]	Java	GPL	貧弱
PARSEC [5]	拡張 C 言語	有償	豊富

3. モデル駆動による性能検証のアプローチ

本章では，課題と解決方針，提案手法の特徴を示す．

3.1 性能検証の課題と解決方針

従来のアプローチでは，システム設計結果からハードウェアに着目した待ち行列モデルを作成し，システムのデータフローを解析することが多い．また，システム設計結果と待ち行列モデルの対応を分析して，設計変更が必要な箇所を特定する．従来の課題を以下に示す．

(1) 制御方式が複雑化しており，実システムの設計結果から誤りなく待ち行列のシミュレーションコードを作成することが困難．
(2) ソフトウェアでの要求実現の比重増加に対応した，検証アルゴリズムの修正が困難．
(3) システム構成が大規模化しており，シミュレーションコードの作成工数が増加．

上記の課題に対して，システム設計結果に数値情報を直接記入して性能検証モデルを作成し，これを基にシミュレーションコードを作成して検証する手法を提案する．また，性能検証モデル上で設計変更が必要な箇所を直接特定する．我々のアプローチを図 1に示す．

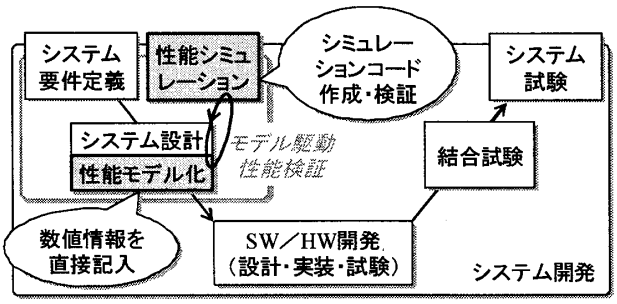

図1 モデル駆動による性能検証のアプローチ

本アプローチの狙いを以下に示す．

(1) 設計結果とシミュレーションコードの対応付けを容易にし，シミュレーションコードを誤りなく作成．
(2) シミュレーションコードの実行・検証環境の仕組みを詳細に定義し，検証アルゴリズムの修正を容易化．
(3) シミュレーションコードの共通部分をライブラリとして提供し，コードの作成工数を削減．

3.2 我々が提案するシミュレーション手法の特徴

本稿のシミュレーション手法の特徴を以下に示す．

(1) 実開発で普及している UML を拡張した MARTE Profile を用いて，性能検証の数値情報を記入．
(2) LQN を用いて実システムのデータ／制御フローを記述し，さらに検証環境の実装情報を明確化．
(3) オブジェクト指向言語で UML と相性がよい SimPy を用いて，シミュレーションライブラリを提供．

本手法で計算する検証指標を表 2に示す[6]．

表 2 検証指標

項目	説明
応答時間	要求入力に対する応答出力までの平均時間
リソース利用率	ソフトウェア／ハードウェアリソースでの処理時間とアイドル時間の比率

† 三菱電機株式会社 情報技術総合研究所
Mitsubishi Electric Corporation, Information Technology R & D Center

本手法の概要を，従来手法との対比を含めて図 2に示す．

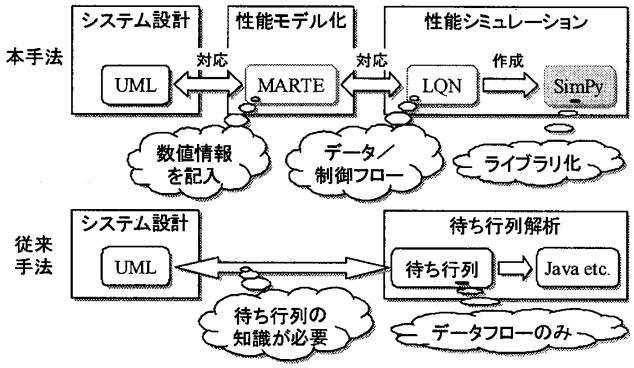

図2　シミュレーション手法の概要

4. MARTE Profile を用いたシミュレーション手法

本章では，モデルと検証環境の定義，計算精度を示す．

4.1 性能検証モデル

図 2に示した性能モデル化フェーズでは，MARTE Profile を用いて性能検証の数値情報をシステム設計結果に直接記入して，性能検証モデルを作成する．

MARTE Profile のサブプロファイルである PAM（Performance Analysis Modeling）を用いた性能検証モデルの概要を図 3に示す．

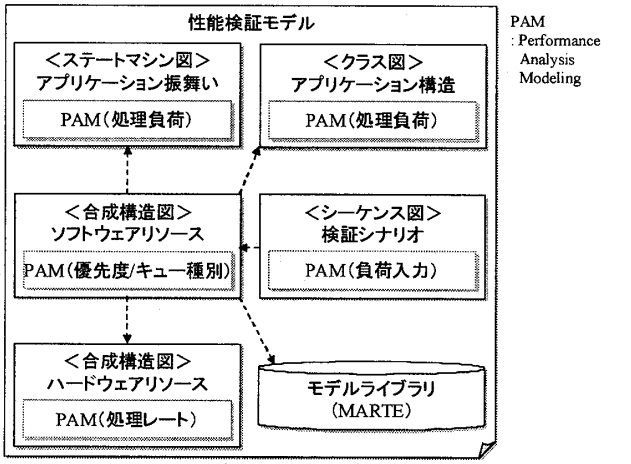

図3　性能検証モデルの概要

図 3に示した性能検証モデルの説明を以下に示す．

(a) アプリケーション構造／振舞いに対して，処理負荷（処理時間，処理回数）を記入．
(b) ソフトウェアリソースに対して，多重度，優先度，キュー種別，ハードウェアリソース割当を記入．
(c) ハードウェアリソースに対して，多重度，処理レートを記入．
(d) 検証シナリオとして，開放型／閉鎖型の種別，到着間隔（開放型），入力数（閉鎖型），同期／非同期メッセージを記述．

上記の性能検証モデルを構成する要素を表 3に示す．表 3では，MARTE Profile で定義されている 189 個のステレオタイプの中から，シミュレーション検証に必要十分な要素と属性を抽出している．

表 3　性能検証モデルの要素と属性

項目	MARTE 要素	要素の属性
アプリ構造	PaStep (提供サービス)	hostDemand
アプリ振舞い	PaStep (詳細ステップ)	hostDemand
SW リソース	SwSchedulable Resource	priorityElements
	SwMutualExclusion Resource	waitingQueue Policy
	PaRunTInstance	host
HW リソース	GaExecHost	throughput
検証シナリオ	GaAnalysisContext	context
	GaWorkloadEvent	pattern
	同期／非同期メッセージ	―

ここで，表 3の MARTE 要素欄の英語表記は MARTE Profile のステレオタイプ，タグ欄はステレオタイプの属性を表す．

4.2 シミュレーションコードと検証環境

図 2に示した性能シミュレーションフェーズでは，シミュレーションコードとこれを実行する検証環境を作成する．シミュレーションコードは LQN を用いて作成する．検証環境は SimPy を基盤に LQN 処理系として実装する．

シミュレーションコードと検証環境の概要を図 4に示す．

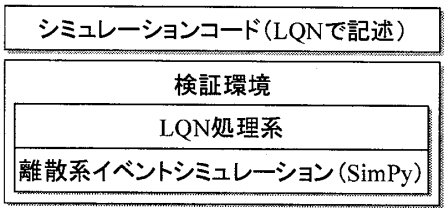

図4　シミュレーションコードと検証環境の概要

図 4に示したシミュレーションコードを構成する LQN の要素を表 4に示す．表 4で＊印を付けた要素は，文献[2]での定義を詳細化し，検証環境の実装に必要な要素を追加したものである．

表 4　シミュレーションコードの要素

LQN 要素	説明
Entry	提供サービス．処理時間を指定．
Activity	フォーク，ループなどの振舞いの詳細ステップ．処理時間を指定．
Task	ソフトウェアリソース．多重度，キュー種別，優先度を指定．
TaskWork (*)	Task が受信するメッセージ．
TaskThread (*)	Task が保持する個々のスレッド．
Processor	ハードウェアリソース．多重度を指定．
ReferenceTask	負荷入力．開放型／閉鎖型の種別，到着間隔（開放型），入力数（閉鎖型）を指定．
Reference Terminator (*)	システム出力の終端点．TaskWork を受信した時点で，検証指標を計算．
Request	メッセージ．同期／非同期を指定．

表 3に示した MARTE 要素と表 4に示した LQN 要素の変換規則を表 5に示す．

表 5　MARTE－LQN 変換規則

項目	MARTE 要素	LQN 要素
アプリ構造	PaStep (提供サービス)	Entry
アプリ振舞い	PaStep (詳細ステップ)	Activity
SW リソース	SwSchedulable Resource	Task (優先度)
	SwMutualExclusion Resource	Task (キュー種別)
	PaRunTInstance	Processor (割当)
HW リソース	GaExecHost	Processor (処理レート)
検証シナリオ	GaAnalysisContext	シミュレーション時間，パラメータ
	GaWorkloadEvent	ReferenceTask
	同期／非同期メッセージ	Request

4.3 検証アルゴリズムと計算精度

検証アルゴリズムの概要を図 5に示す．本アルゴリズムは図 4に示した検証環境に実装するものであり，統計学に基づくシミュレーションを実行する．

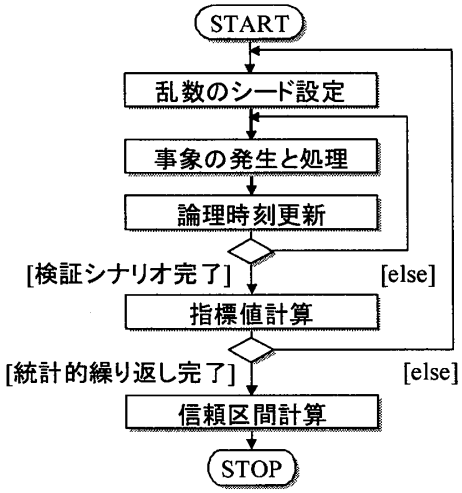

図5　検証アルゴリズムの概要

図 5では，確率計算のための乱数のシードを変えて検証シナリオを複数回実行する．1 回の実行で表 2に示す検証指標の値を計算し，統計学で定められた回数だけ実行を繰り返した後，表 6に示す検証指標の信頼区間を計算する．ここで，信頼区間は検証指標の許容誤差の範囲を表す．

表 6　検証指標の信頼区間の定義

項目	説明
95%信頼区間	検証指標の平均値±誤差の範囲が，真値を含む確率が 95%である区間．

検証アルゴリズムの計算精度の基本的な評価として，待ち行列理論の代表的な解析解[6][7]と比較する．計算精度の評価結果を表 7に示す．表 7のパラメータ欄の値の組は，{到着間隔，サービス時間，サーバ数，入力数} を表す．また，項目名は，待ち行列理論のモデルを表現するためのケンドールの記号を用いて記述している．

表 7　計算精度の評価結果－応答時間

項目	パラメータ	95%信頼区間	解析解
M/M/1	{3, 2, 1, -}	5.94±0.53 秒	6.00 秒
M/M/S	{1.2, 12, 17, -}	11.92±0.14 秒	12.05 秒
M(n)/M/S	{20, 1, 1, 30}	10.11±0.39 秒	10.26 秒

表 7では，すべての項目で応答時間の信頼区間が解析解を含んでおり，許容誤差の範囲内で一致した．これにより，検証アルゴリズムの基本的な正しさを確認できた．

5. シミュレーション手法の適用例

本章では，基本的な構成のサーバマシン[8]を題材に，4 章で定義したシミュレーション手法の適用例を示す．

検証対象のサーバマシンは，CPU とディスク装置で構成され，複数のバッチ処理を多重して実行する．シミュレーションのパラメータ値を表 8に示す．

表 8　シミュレーションのパラメータ値

CPU 個数	cpu：1
ディスク台数	disk1：1，disk2：1
バッチ入力数	batch1：2，batch2：2
バッチ処理 [秒] (*)	batch1：{3000, 1000, 0} batch2：{2000, 0, 2000}
優先度	batch1 と batch2 は同等

(*) 値の組は {cpu，disk1，disk2} を表す．

5.1 バッチ 2+2 多重の検証結果

2 種類のバッチ処理をそれぞれ 2 多重する場合の性能検証モデルを図 6～図 8に示す．

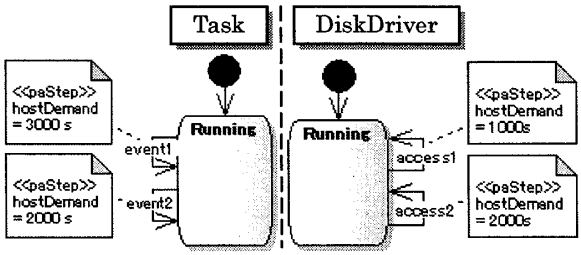

図6　性能検証モデル－アプリケーション振舞い

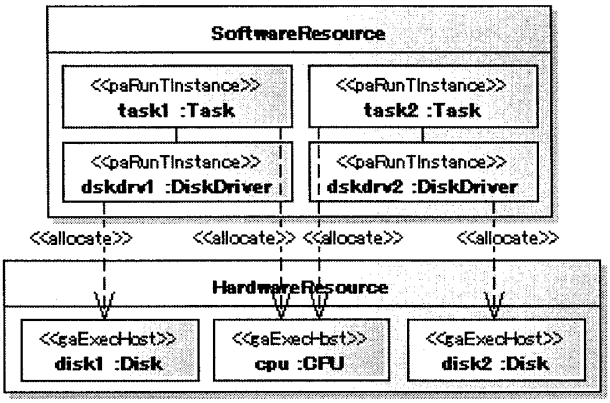

図7　性能検証モデル－リソース間の割り当て

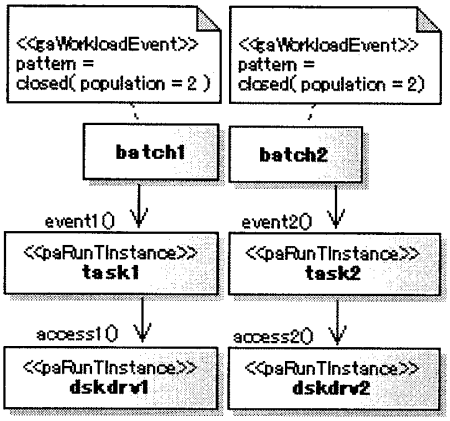

図8　性能検証モデルー検証シナリオ

検証結果と文献[8]の計算結果の比較を表 9に示す.

表 9　検証結果の比較

項目 1	項目 2	95%信頼区間	文献[8]	誤差
応答時間	batch1	9,152±402 秒	10,571 秒	10%以内
	batch2	10,267±373 秒	9,446 秒	5%以内
リソース利用率	cpu	99.2±0.3%	99.0%	なし
	disk1	21.6±1.2%	18.9%	8%以内
	disk2	37.3±2.5%	42.3.%	6%以内

表 9の検証結果の要約を以下に示す.

- CPU のリソース利用率は，許容誤差の範囲内で一致.
- バッチ処理の応答時間，ディスク装置のリソース利用率は誤差 5%～10%以内．許容誤差の 2～3 倍程度.

5.2 シミュレーションコードと LQN 処理系の実装

5.1で作成したシミュレーションコードと，4.2に示した LQN 処理系の実装コードの規模を以下に示す.

- シミュレーションコード　：0.07KLOC（10.0%）
- LQN 処理系　：0.63KLOC（90.0%）
- 合計　：0.70KLOC（100.0%）

6. シミュレーション手法の評価

本章では，提案手法の評価と今後の課題を示す.

6.1 性能検証モデルの評価

MARTE Profile を用いて，システム設計結果に性能検証の数値情報を直接記入する方法を規定した．また，性能検証モデルとシミュレーションコードの要素間の対応を 1 対 1に定義した.

これにより，システム設計結果とシミュレーションコードの対応付けが容易になり，シミュレーションコードを誤りなく作成するための実現性が確認できた.

ネットワーク特性を組み合せた性能検証モデルの拡張は今後の課題である．まず，時分割や Ethernet 型といった実運用されている代表的なネットワークをモデル化する.

6.2 シミュレーションコードと検証環境の評価

離散系イベントシミュレーションと LQN 処理系を合わせて検証環境とし，その上で実行するシミュレーションコードを定義した．また，LQN の定義を詳細化して，検証環境を実装する上で必要な要素を追加した.

これにより，シミュレーションコードを実行する検証環境の実装の仕組みが明確になり，検証アルゴリズムを容易に修正するための実現性が確認できた.

文献[2]で定義されている，各要素が応答を返した後に実行するバックグラウンド処理時間のサポートは今後の課題である.

6.3 検証アルゴリズムの評価

応答時間とリソース利用率の信頼区間計算のアルゴリズムを定義し，待ち行列理論の解析解と許容誤差の範囲内で一致することを確認した．また，サーバマシンでのバッチ処理を題材に検証し，文献[8]との誤差が 2%～10%以内（許容誤差の 2～3 倍以内）であることを確認した.

これにより，高度化する要求に対応して検証アルゴリズムを修正する際に，計算精度が維持されるか確認する方法が明確になった.

本手法と文献[8]の誤差要因の調査とアルゴリズムの改善は今後の課題である.

6.4 実装コードの評価

シミュレーションコードと LQN 処理系の実装コードの規模は全体で 0.70KLOC となった．また，LQN 処理系はさまざまなシステムの性能検証で再利用できるライブラリとして作成し，実装コードの比率は 90%となった.

これにより，手作業で作成するシミュレーションコードの作成工数の削減が見込める.

7. おわりに

本稿では，組込みシステム向けの性能シミュレーション手法を提案した．具体的には，システム設計結果に性能検証の数値情報を直接記入する方法を規定し，シミュレーションコードとの対応を 1 対 1 に定義した．また，検証環境の実装の仕組みを明確にした．また，サーバマシンでの複数バッチ多重処理を題材に本手法を適用し，既存手法との誤差が 2%～10%以内であることを確認した.

これにより，複雑な実システムを対象にした性能のモデル化，高度化する要求に対応した検証アルゴリズムの修正しやすさに関する実現性が確認できたと判断する.

この評価結果を踏まえ，今後は時分割や Ethernet 型といった代表的なネットワークのモデル化，各要素でのバックグラウンド処理時間のサポート，解析解との誤差低減のための検証アルゴリズムの改良といった課題に取り組む予定である.

参考文献

[1] *A UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems, Version 1.0*, OMG formal/2009-11-02, Object Management Group (2009).
[2] Greg Franks, et al., *Layered Queueing Network Solver and Simulation User Manual Revision 6840*, Department of Systems and Computer Engineering, Carleton University (2005).
[3] http://simpy.sourceforge.net/index.html
[4] http://www.uark.edu/~rossetti/research/research_interests/simulation/java_simulation_library_jsl/
[5] http://pcl.cs.ucla.edu/projects/parsec/
[6] 森戸 晋, 他, *システムシミュレーション*, 朝倉書店 (2000).
[7] 秋丸春夫, 他, *情報通信トラヒックー基礎と応用ー*, オーム社 (1990).
[8] 紀 一誠, *待ち行列ネットワーク*, 朝倉書店 (2002).