A-020

MARTE Profile によるリアルタイムシステム向け性能検証手法の提案
An Approach of Performance Analysis for Real-Time Systems with MARTE Profile

磯田 誠† 徳永 雄一†
Makoto Isoda Yuichi Tokunaga

1. はじめに

組込みリアルタイムシステムは，制御装置を中心にセンサ装置と駆動装置を備え，外部環境に対して適切な制御をかけることを目的とする．近年の多機能・高性能の要求を実現するため，複数計算機をネットワーク接続して運用することで，システム単体での機能・性能の限界を克服する取り組みが盛んである．全体としてはあたかも一つのシステムとして機能・性能を提供する必要があるため，ネットワークを介した場合でもリアルタイム動作することを特徴とする．

複数計算機をネットワーク接続する構成をとる場合，接続先の計算機をすべて揃えたり代替用の評価環境を構築したりするのは困難である．特に性能面では，従来はシステムの動作を待ち行列モデルで記述・シミュレーションする方法が代表的だが，設計担当者に専門知識が必要，呼び出しの動作を表せない，周期タスクを表せないという課題がある．

本稿では，これらの課題に対して，設計工程で作成した UML モデルに数値情報を直接記入して性能検証モデルを作成し，これをシミュレーションモデルに自動変換して検証する手法を提案する．また，制御系システムの題材を用いて，本手法の適用例と評価結果を示す．

2. 性能検証の関連技術

2.1 LQN

待ち行列ネットワークを拡張した LQN（Layered Queuing Network）は，システムのデータフローおよび制御フローの階層化が可能であり，メッセージと呼び出しが混在する場合の平均応答時間をシミュレーションできる[1]．しかし，モデル記法が固有のため実開発では普及していない．

2.2 RMA

RMA（Rate Monotonic Analysis）は，1 つの CPU 上で複数のタスクが周期的に実行される場合の最悪応答時間を解析的に計算できる[2]．しかし，タスク間のメッセージ送受信を扱えないため，実システムの仕様を抽象化または限定する必要がある．

2.3 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[3]．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．

† 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

3. モデル駆動による設計工程での性能検証

3.1 性能検証の課題

従来の取り組みでは，計算機やネットワークで構成されるシステムの動作をイベントタスク（メッセージ受信時に起動されるタスク）とその間のメッセージ送受信を表す待ち行列モデルで記述し，性能を実現できるかシミュレーションする方法が代表的である．従来の待ち行列シミュレーションの課題を以下に示す．

(1) 設計担当者が直接待ち行列モデルを作成するため，専門知識・スキルが必要．
(2) システム内の呼び出しの動作を表せない．設計情報として呼び出しの記述は必須．
(3) システム内の周期タスク（指定周期で起動されるタスク）を表せない．一般に，リアルタイムシステムではイベントタスクと周期タスクが混在する．

3.2 我々が提案する性能検証手法の特徴

3.1に示した課題に対して我々は，設計工程で作成した UML モデルに数値情報を直接記入して性能検証モデルを作成し，これをシミュレーションモデルに自動変換して検証する手法を提案する．本手法での性能検証プロセスを図1に示す．

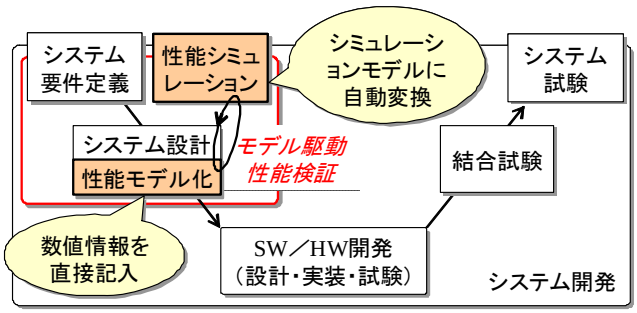

図1 設計工程での性能検証プロセス

本手法の特徴を以下に示す．また，従来手法との対比を図 2に示す．我々は論文[4]で(1)(2)の検討・評価結果を報告した．本稿では(3)に関して，論文[4]での提案手法の拡張を検討・評価した結果を報告する．

(1) 実開発で普及している UML を拡張した MARTE Profile[3]を用いて，性能検証のための数値情報の記法を定義．待ち行列モデルの専門知識・スキルが不要．
(2) シミュレーションモデルとして LQN のアルゴリズムを実装．呼び出しの動作を表現可能．
(3) イベントタスク－周期タスク間でメッセージを送受信できるようにモデルを拡張．イベントタスクと周期タスクの混在が可能．

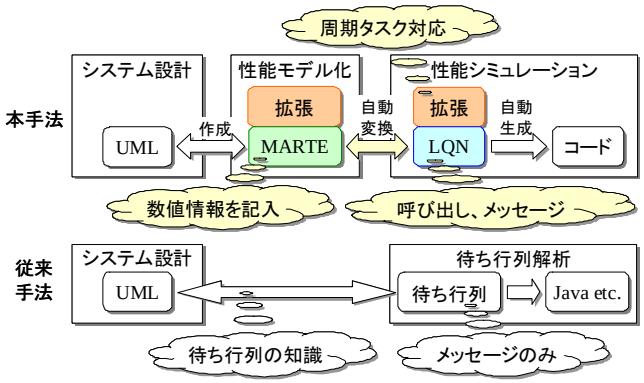

図2　本手法の特徴および従来手法との対比

本手法で計算する検証指標を表 1に示す.

表 1　本手法の検証指標

項目	説明
応答時間	システムに入力された要求メッセージに対する応答が出力されるまでの平均時間.
サービス時間	システム内で要求メッセージを実際に処理している時間の累積値.
待ち時間	応答時間からサービス時間を除いた時間．キューイング時間，プロセッサ取得時間，周期満了時間の和に細分化される.
利用率	ハードウェアが，処理を実行している時間と実行していない時間の比率の平均値.
キュー長	システム内の各要素が備えるメッセージキューでの滞留数の平均値.

4. 周期タスクに対応した性能モデリング手法

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 UML モデルに直接記入して，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.

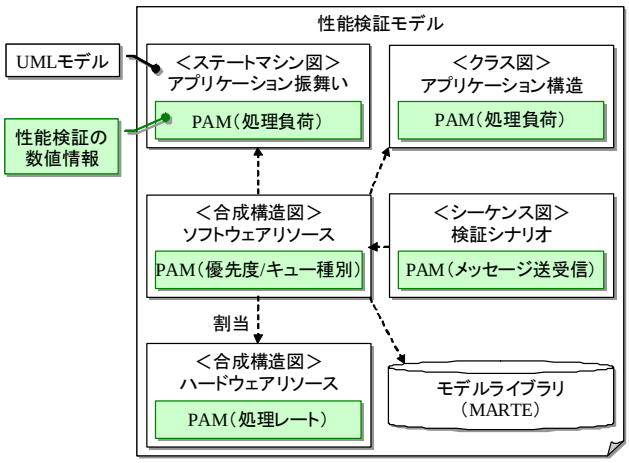

図3　性能検証モデルの概要

図 3に示した性能検証モデルの構成に従って，システムを構成する計算機とネットワークのモデルの記述内容をそれぞれ表 2，表 3のとおり定義する.

表 2　計算機の性能検証モデル

項目	説明
アプリ構造／振舞い	クラス／ステートマシンの処理時間または処理回数
SW リソース	タスクの多重度，優先度，キュー種別，起動周期，ロック数，プロセッサ割当
HW リソース	プロセッサの多重度，処理レート，プリエンプト可否
検証シナリオ	負荷入力の開放型／閉鎖型種別，到着間隔，入力数，メッセージと呼び出し

表 3　ネットワークの性能検証モデル

項目	説明
アプリ構造	転送処理の通信時間または通信量
SW リソース	通信チャネルの多重度，優先度，キュー種別，起動周期，ロック数，通信メディア割当
HW リソース	通信メディアの多重度，処理レート，プリエンプト可否
検証シナリオ	メッセージと呼び出し（開放型／閉鎖型種別などは計算機のモデルで記述）

計算機のモデルでは，ハードウェア上で動作するソフトウェア，ミドルウェア，OS，I/O 割り込みなどを記述する．ネットワークのモデルでは，時分割，Ethernet 型，トークンリングなどのネットワークを記述する.

計算機とネットワークのモデルを表現するため，表 4に示す両者に共通な記法を定義する．表 4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，性能モデリングに必要十分な要素を抽出している.

表 4　性能検証モデルの要素

項目	MARTE 要素	要素の属性
アプリ構造／振舞い	PaStep	hostDemand
SW リソース	SwSchedulable Resource	resMult
		priorityElements
	SwMutualExclusion Resource	waitingQueue Policy
	SwTimerResource	durationElements
	PaLogicalResource	poolSize
	PaRunTInstance	host
HW リソース	GaExecHost	resMult
		throughput
		isPreemptible
検証シナリオ	GaWorkloadEvent	pattern
	メッセージ／呼び出し	―

4.2 シミュレーションモデル

図 2に示した性能シミュレーションフェーズでは，表 1に示した検証指標を計算するため，性能検証モデルの記述内容からシミュレーションモデルを作成する．シミュレーションモデルを表現するため，LQN を用いて表 5に示す記法を定義する．本手法で追加した要素を*印で示す.

表 5　シミュレーションモデルの要素

LQN 要素	説明
Entry	提供サービス．処理時間を指定．
Task	ソフトウェアリソース．多重度，優先度，キュー種別，ロック数を指定．
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．
TaskLock (*)	制御フローを同期するためのロック．
TimerTask (*)	周期的に動作する Task．周期を指定．
Processor	ハードウェアリソース．多重度，プリエンプト可否を指定．
ReferenceTask	負荷入力．開放型／閉鎖型種別，到着間隔，入力数を指定．
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	要素間でやり取りするメッセージ．メッセージ／呼び出し種別を指定．

4.3 モデル間の変換規則

4.1に示した計算機とネットワークのモデルから4.2に示したシミュレーションモデルへの変換規則を表 6に示す．モデル間の変換およびシミュレーション処理系をツール実装することで，設計担当者の作業を自動化する．

表 6　モデル間の変換規則

項目	MARTE 要素	LQN 要素
アプリ構造／振舞い	PaStep	Entry
SW リソース	SwSchedulable Resource	Task
	SwMutualExclusion Resource	
	PaRunTInstance	
	SwTimerResource	TimerTask
	PaLogicalResource	TaskLock
HW リソース	GaExecHost	Processor
検証シナリオ	GaWorkloadEvent	ReferenceTask
	メッセージ／呼び出し	Request

5. 制御系システムへの適用例

5.1 適用対象の特徴と仕様諸元

制御系システムの題材である Open Robotics Controller（以降，ORC）[5]に対する本手法の適用例を示す．

ORC はロボットアームとその制御装置で構成される．制御装置には主タスク A1，B1，C が実装され，これらのタスクがそれぞれ制御指令を処理してロボットアームを駆動する．また，ORC の付加価値向上のため，サードパーティによる拡張タスク M の追加実装が可能である．ORC が性能面で満たすべき事項を以下に示す．

(1) 拡張タスクが主タスクのリアルタイム処理に悪影響を与えないこと．

(2) 拡張タスクの平均的な応答時間を予想可能なこと．

ORC の構成を図 4に，実装するタスクの主要な仕様諸元を表 7に示す．主タスクと拡張タスクは 1 つの CPU 上で動作する．

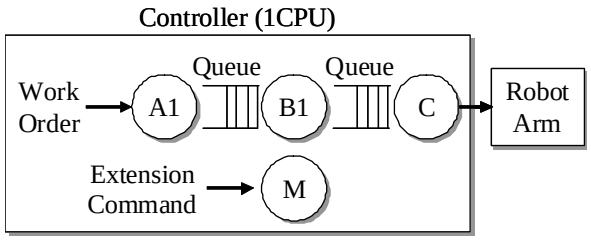

図4　Open Robotics Controller（ORC）の構成

表 7　シミュレーションモデルの仕様諸元

タスク	種別	イベント到着／起動周期	処理時間	優先度
A1	イベント	平均 75ms の指数分布	平均 9ms の指数分布	Low
B1	周期	24ms	1～2ms の一様分布	High
C	周期	4ms	0.5～1ms の一様分布	Very High
M	イベント	平均 100ms の指数分布	15 ～ 25ms の一様分布	Medium

5.2 性能検証モデルの作成

タスク構成を表す合成構造図を図 5に示す．すべてのタスクを 1 つの CPU クラスに割り当てる（図では省略）．

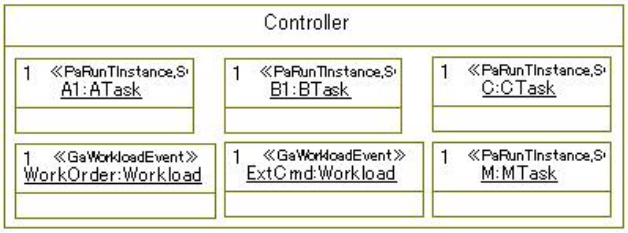

図5　タスク構成

検証シナリオを表すシーケンス図を図 6，図 7に示す．

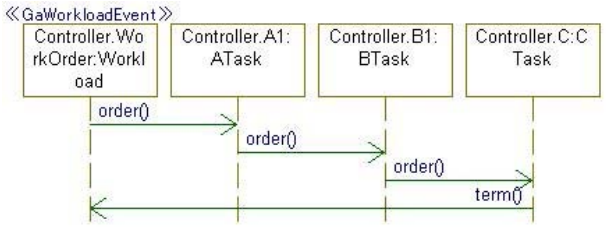

図6　検証シナリオ－主タスク A1，B1，C

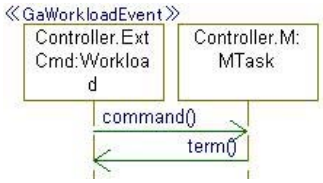

図7　検証シナリオ－拡張タスク M

5.3 シミュレーションモデルへの変換

5.2で作成した性能検証モデルを変換規則に従ってシミュレーションモデルに変換して検証指標を計算する．本作業はツール実装して自動化したので，変換前後のモデルを目

視で比較して，意図どおりに変換されたことを確認した．検証指標の計算結果は5.4に示す．

5.4 適用結果

拡張タスク M の到着率（イベント到着間隔の逆数）をパラメータとして仕様諸元から増加させたときの，応答時間の変動傾向のシミュレーション結果を図 8に示す．

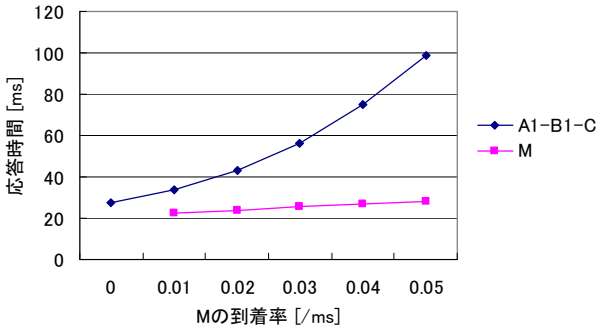

図8　応答時間の変動傾向

図 8では，拡張タスク M の処理量が増加すると，主タスク A1-B1-C の応答時間が大幅に増加している．主タスク A1 のサービス時間とキュー長も同じ傾向で増加しており，拡張タスク M より優先度が低いことが原因で，制御指令が滞留していることが判明した．

主タスク A1 と拡張タスク M の優先度をそれぞれ Medium，Low に変更したときの，応答時間の変動傾向のシミュレーション結果を図 9に示す．

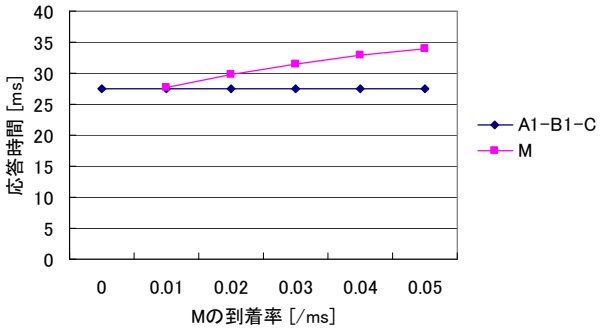

図9　優先度変更後の応答時間の変動傾向

図 9では，主タスク A1-B1-C の応答時間が一定している．また，拡張タスク M の応答時間は線形に増加しており，予想可能である．また，CPU 利用率は到着率 0.5 で約 80%であり，CPU の増強はまだ必要ない．以上より，ORC が性能面で満たすべき事項を確認できた．

6. 性能検証手法の評価

6.1 性能検証モデルの評価

論文[4]で検討した性能検証モデルをベースに，計算機とネットワークのモデルに共通な記法を定義し，周期タスクの記法を追加した．また，ベースとした記法では 7 要素 7 属性，拡張した記法では 8 要素 11 属性であった．

これにより，イベントタスクと周期タスクの混在というリアルタイムシステムの特徴のモデル化が可能となった．また，設計担当者が複数計算機をネットワーク接続した構成をモデル化する際の作業負担の軽減が見込める．

一方，ステートマシン図からシミュレーションモデルへの変換は未対応である．

6.2 シミュレーションモデルの評価

論文[4]で検討したシミュレーションモデルをベースに，6.1と同様に周期タスクに対応し，モデル間の変換と検証指標の計算を自動化した．また，制御系システムの題材に対して応答時間の変動傾向を計算し，タスクの優先度変更により性能面で満たすべき事項を確認した．

これにより，システム設計段階で決められる情報を用いてシミュレーションし，以降の工程で性能問題を起こしそうな部分を事前に特定可能となった．また，シミュレーションモデルへの変換と検証指標の計算に関する作業負担の軽減が見込める．

一方，シミュレーションの計算結果の正しさとしては，変動傾向による定性的評価に留まっている．

6.3 今後の課題

6.1，6.2の評価結果から以下の改善点を抽出した．

(a) 性能検証モデルについては，ステートマシン図の変換に対応し，実システムの複雑な振舞いの性能検証を可能とする．

(b) シミュレーションモデルについては，シミュレーション結果の定量的評価を進め，計算精度を向上させていく．

7. おわりに

本稿では，リアルタイムシステム向けの性能検証手法を提案した．具体的には，システムを構成する計算機とネットワークに共通なモデルの記法を定義した．また，シミュレーションモデルへの変換と検証指標の計算をツール実装して自動化した．

これにより，イベントタスクと周期タスクの混在というリアルタイムシステムの特徴のモデル化，システム設計段階で決められる情報を用いたシミュレーションによる性能問題の事前特定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，今後はステートマシン図への対応による実システムの複雑な振舞いの性能検証，シミュレーション結果の定量的評価と計算精度の向上という課題に取り組む予定である．

参考文献

[1] Greg Franks, et al., *Layered Queueing Network Solver and Simulation User Manual Revision 6840*, Department of Systems and Computer Engineering, Carleton University (2005).

[2] C. L. Liu, James W. Layland, “Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment,” *Journal of ACM*, Vol.20, No.1, pp.46-61 (1973).

[3] *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).

[4] 磯田 誠, 田村直樹, “UML MARTE Profile を用いた性能シミュレーション手法の提案,” *FIT2010*, B-006, pp.259-262 (2010).

[5] Scott A. Hissam, Mark Klein, “A Model Problem for an Open Robotics Controller,” *CMU/SEI-2004-TN-030*, Carnegie Mellon University (2004).