情報処理学会論文誌 Vol.53 No.12 2650–2659 (Dec. 2012)

組込みソフトウェアの動的振舞いによるシステム性能の評価手法

坂本 佳史[1,a)] 小野 康一[2] 中田 武男[2] 安浦 寛人[3]

受付日 2012年3月5日, 採録日 2012年9月10日

概要：本稿では，リバースモデリング手法とモデル・シミュレーションを組み合わせることで，ソフトウェアの動的な振舞いに基づいて組込みシステムの性能を評価する手法を提案する．組込みシステムに搭載するソフトウェアはシステム全体の性能に大きな影響を及ぼす．そのためソフトウェア開発の上流工程であるアーキテクチャ設計において，組込みシステムの性能を評価することは重要である．しかしソフトウェア設計が詳細化されていない上流工程において組込みシステムの性能を評価する有効な手段が確立していないことが問題である．そこで組込みシステムの性能に強い影響を及ぼすソフトウェアの動的な振舞いに着目する．リバースモデリング手法によりソフトウェアの動的な振舞いを表すモデルを作成する．上流工程における組込みシステムのハードウェア要素と，ソフトウェア要素の構成やソフトウェアのアーキテクチャは，UML を用いて高い抽象度で表現する．これらのモデルを用いたモデル・シミュレーションにより組込みシステムの性能を評価する．本手法を実際の組込みシステムであるマルチ・ファンクション・プリンタ製品に適用した．シミュレーションによる性能評価の結果と，実システムの測定値が同様の傾向を示すことから，本手法が組込みシステム開発の上流工程における性能評価に有効であることを確認した．

キーワード：組込みソフトウェア，動的振舞い，性能評価，リバースモデリング，UML モデル

System-level Performance Estimation for Embedded System Based on the Dynamic Behavior of Software

Yoshifumi Sakamoto[1,a)] Kouichi Ono[2] Takeo Nakada[2] Hiroto Yasuura[3]

Received: March 5, 2012, Accepted: September 10, 2012

Abstract: In this paper, we propose a combination of reverse modeling techniques and model based simulation, to estimate the performance of embedded systems based on the dynamic behavior of embedded software. Software that is installed in an embedded system, greatly influences on overall system performance. Therefore, in the architectural design is the early stage of software development, to estimate the performance of embedded systems is important. A major problem of software development is unreliable system-level performance estimation at the early stages. We focus on the dynamic behavior of software a strong influence on the performance of embedded systems. To create a model that represents the dynamic behavior of the software by reverse modeling method. The dynamic behavior of the software, the hardware configuration and software architecture is described by a high level of abstraction using the UML. The model-based simulation using these models to estimate the performance of embedded systems. By comparing the results of performance evaluation by simulation, the results of the measurement of the actual system, it was confirmed that they showed a similar trend. We confirmed that the proposed method is effective to estimate the performance in the early stage process.

Keywords: embedded software, dynamic behavior, performance estimation, reverse modeling, UML model

[1] 九州大学大学院システム情報科学府
Graduate School of Information Science and Electrical Engineering, Kyushu University, Fukuoka 819–0395, Japan
[2] 日本アイ・ビー・エム株式会社東京基礎研究所
IBM Research - Tokyo, Kouto, Tokyo 135–8511, Japan
[3] 九州大学
Kyushu University, Fukuoka 819–0395, Japan
[a)] sakay@jp.ibm.com

1. はじめに

組込みシステムの高性能化に対する市場の要求は年々高くなっている．その要求に対応してシステム性能は向上しているが，性能要件の充足可能性は開発プロセスの下流工程である実装や検証の段階まで確定できないことが多くある．その結果として性能改善のために開発の手戻りが発生することが，組込みシステム開発における大きな課題の1つとなっている [1]．組込みシステムの性能には，上流工程のシステム・アーキテクチャ設計が下流工程の詳細設計よりも大きく関与する．したがって性能改善を目的とする開発の手戻りは上流工程まで遡る場合が多く，開発スケジュールの遅延や開発費の大幅な増大を招く．そのため，組込みシステムの性能要件の充足可能性を開発の上流工程において高い精度で見積もることは，手戻りのリスクを大幅に軽減する．組込みシステムの性能は組込みシステムに搭載するソフトウェアの振舞いから強い影響を受ける．ハードウェア資源であるアクセラレータや，アプリケーション固有の処理に特化した回路ブロックは，その動作のタイミングや手順がソフトウェアによって制御されることで有効に機能し，所定の性能を発揮することができる．そのため組込みシステムの性能を見積もるためにはソフトウェアの振舞いを反映する必要がある．

一方，近年の組込みシステム開発では，既存のシステムに機能の追加や更新を適用して，次期のシステムを開発する形態が多く用いられる [2]．このような開発形態は差分開発と呼ばれる．差分開発を適用する利点は再利用である．次期のシステムに対する要求分析に基づいて既存のシステムで変更を必要としない構成要素は，そのまま用いることができる．差分開発は既存のシステムを開発の出発点とすることから，開発プロセスの上流工程から下流工程へ，段階的な詳細化を完全な形では実施せずに開発される場合がある．組込みソフトウェアの差分開発を例にあげると我々の経験において，差分開発における再利用の対象は下流工程の主な成果物であるソースコードに限定される場合が多い．また要件定義の終了直後に実装を開始することから，システム・アーキテクチャ設計やソフトウェア・アーキテクチャ設計に関連する技術検討が実施されず，結果として技術検討の成果物である仕様書などの技術資料が更新されないことも多く見受けられる．反面，1回の差分開発の前後では，組込みシステムに求められる要件や制約の多くが共通するという特徴がある．この点に着目すると，既存のシステムを解析することによって，次期のシステムに必要な要件や制約の多くを得ることができるという利点がある [3]．したがって，差分開発を適用している組込み製品開発では，既存のシステムを解析するリバースエンジニアリング技術の利用価値と重要性は高い．従来のリバースエンジニアリング技術は主にソースコードをその対象とするため，下流工程においてリファクタリングなどを目的とした解析の手法として活用されている．しかしながら，上流工程においてシステムの性能を見積もることを目的として，ソフトウェアの振舞いを解析する手法としては十分な効果が得られない．その理由は，解析の対象であるソースコードは抽象度が低く，機能やデータの粒度も詳細であるため，上流工程で考慮される抽象度の解析には適さないからである．つまり上流工程の抽象度に見合った組込みシステムの表現が必要であり，詳細設計に比べて高い抽象度が求められる．加えて性能を評価するためには，時間情報を取り扱うことが必要となる．

これらを解決するために本稿では，リバースモデリング手法とモデル・シミュレーションを組み合わせることで，ソフトウェアの動的な振舞いに基づいて組込みシステムの性能を評価する手法を提案する．我々はこれまでにもリバースモデリング手法と，それらリバースモデリング手法によって作成したモデルを，次期のシステム開発に再利用するモデル駆動開発手法を提案している [4], [5], [6]．従来のリバースエンジニアリング技術の適用では，ソフトウェアの振舞いを得ることが困難 [7] であるため，我々は動的振舞いに着目する．ここで動的振舞いとは，特定のシナリオで実際のシステムを動作させたときのソフトウェアの挙動であり，ハードウェア要素との相互作用も含めた処理フローと定義する．リバースモデリング手法とは静的解析，設計文書解析，動的振舞解析 [4], [5], [6] の3種類のリバースエンジニアリング技術を用いて既存のシステムを解析することでモデルを作成する手法である．本手法で作成するモデルとは，ソフトウェアの動的振舞いを表す動的振舞いモデル，対象システムのハードウェア資源であるリソースの構成とリソース使用量ならびにシステムの振舞いを高い抽象度で表現するシステム・リソースモデル，それら2種類のモデルを組み合わせた性能評価モデルの3種類である．ここで高い抽象度によるモデルの表現とは，性能情報を処理単位の粒度でモデルに表現することを意味する．処理単位とはシステムにおいて固有の機能を実現するための，まとまりのある一連の処理を意味する．本手法におけるモデリングは性能評価を前提とすることから，その文脈においては，システムを構成する各要素について全体の性能に対する影響や寄与の低いものを省略し，システム性能に対して支配的となる要素に限定して記述する．具体的には，対象システムのリソースと，各ソフトウェア要素の振舞いについて，性能の観点で取捨選択してモデリングする．ここでリソースとはプロセッサ，バス，メモリなどのハードウェア資源に加えて，システム全体ならびに処理単位における所要時間を意味する．本手法における性能評価の対象はこれらのリソース使用量である．本稿でメモリ使用量を性能評価の対象とした理由は，組込みシステム製品においてメモリ要素が製品コストや消費電力の観点から搭載量に

厳しい制限を受けることが多くあるからである．モデル・シミュレーションにおいては動的振舞いモデルの個々の処理単位に対応するリソース使用量を，システム・リソースモデルの内部コンポーネントであるリソース管理モジュールが算出することで性能を見積もることを実現している．なお，性能評価のためのモデル・シミュレーションはトレース駆動シミュレーション [8], [9] を適用，モデリング言語には UML（Unified Modeling Language：統一モデリング言語）を用いる．UML は OMG（Object Management Group）[10] の提唱するモデル記述言語である．

本稿では，組込みシステムの代表的な製品の 1 つであるマルチ・ファンクション・プリンタ（MFP：Multi Function Peripheral/Printer，複合機）に対して本手法を適用することでその有効性を考察した．シングルプロセッサのシステム・アーキテクチャを採用する現行の MFP にリバースモデリングを適用して，性能評価モデルである現行システムモデルを作成する．要求分析の結果として得られた次期のシステムに対する機能要求と，性能要求に基づくシステム・アーキテクチャの変更を現行システムモデルに施すことで，次期システムの性能を評価するために用いるモデルである次期システムモデルを作成する．本稿において新しく次期システムモデルに採用したシステム・アーキテクチャはマルチコア・アーキテクチャである．モデル・シミュレーション [11] による性能評価は，次期システムモデルを用いることによって MFP の性能である印刷処理時間を評価する．本手法による性能評価のシミュレーション結果と，実機の実測値との比較において，印刷処理時間の誤差は 6.0%未満であることを確認した．

本稿の構成は次のとおりである．2 章ではリバースエンジニアリングとモデルベースのシミュレーション手法についての関連研究を述べる．3 章ではリバースモデリング手法，4 章ではモデル・シミュレーション，5 章では提案手法の有効性を評価する．最後に，6 章でまとめと今後の課題を述べる．

2. 関連研究

既存のリバースモデリング手法は 2 つに大別される．1 つはソースコードを対象とする静的解析であり，もう 1 つは実際のシステムを動作させることによって得られる実行トレースデータを対象とする動的解析である．静的解析によるリバースモデリング手法としては，ソースコードの構造に重点をおいてモデルを作成する手法が提案されている [12]．また C++ ソースコードのフロー解析結果から，UML の相互作用図を自動抽出する手法が提案されている [13]．この手法により抽出されるモデルの抽象度はソースコードと同等の低い抽象度である．上流工程では現行のシステム・アーキテクチャを変更する，あるいは，構成可能な複数のシステム・アーキテクチャ候補を作成・検討する必要がある．このため，低い抽象度でシステム・アーキテクチャを表現すると，機能やデータの粒度も詳細であるため，変更や作成が困難であり上流工程には適さない．動的解析によるリバースモデリング手法も提案されている [14]．この手法はメタモデルと一貫性規則を用いて，動的振舞いの情報からモデルを作成する．モデル作成のためには，制御構造を把握する必要があることから動的振舞いの情報に条件分岐や反復に関する制御情報を含めなくてはならない．またこの手法は，システム測定による侵襲性により，システムの動的振舞いを変化させる影響があるため，システムの動的振舞いとシステム性能を測定するための正確な時間情報が得られない可能性がある．システム・アーキテクチャをモデルによって表現し，性能評価のシミュレーションに用いるモデルベースのシミュレーション手法が提案されている [15], [16], [17]．これらの手法はモデル記述言語に SystemC に代表されるハードウェア記述言語（HDL：Hardware Description Language）を用いる手法である．また独自のモデル記述言語で作成されたモデルを用いるシミュレーション手法などがある [18], [19]．いずれの手法においても UML を用いた高い抽象度のモデル表現はみられない．文献 [20] は，第 41 回 Design Automation Conference（DAC 2004）併設の Workshop on UML for SoC Design における主要な参加者による論文集であり主な論点は，ハードウェアとソフトウェアの協調設計，コード作成，モデル変換，モデル検証などである．システムレベルの設計モデルを下流工程で活用することを主目的としており，高い抽象度のモデルによる性能評価についての言及はみられない．

3. リバースモデリング手法

本稿におけるリバースモデリング手法とは，既存の組込みシステムに対してリバースエンジニアリングの解析技術を適用することで得られた結果を用いて，高い抽象度のモデルを作成することである（図 **1**）．

リバースエンジニアリングに用いる解析技術は，ソースコードを対象とする静的解析，ソフトウェア開発における技術文書である仕様書や要求文書を対象とする設計文書解析，実際のシステムを解析対象とする動的振舞解析の 3 種類の技術である．静的解析は，ソフトウェア・アーキテクチャに関連するモジュール・バウンダリの把握，レイヤ構

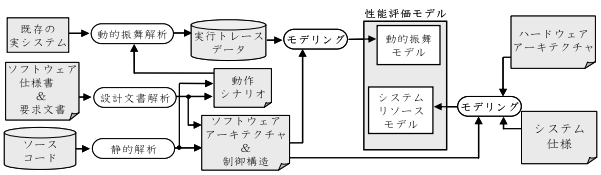

図 1 リバースモデリング手法によるモデル作成フロー
Fig. 1 Modeling flow based on the reverse modeling.

造の把握，モジュール間依存関係の把握による制御構造の取得や，関連する属性の識別を目的とする解析であり，その解析対象は主にソースコードである．設計文書解析は，ソフトウェア仕様書やテスト仕様書，要求文書をその対象とする．解析の目的は，システムの構成や機能割当て，オブジェクトやユースケースに関する情報の獲得と，システムが満たすべき性能要件の把握である．これらの情報は静的解析と動的振舞解析の基礎情報となる．動的振舞解析は実システムである組込みシステムの実機を動作させて行う解析手法である．実システムを動作させることから，動作シナリオを必要とする．動作シナリオは要求文書から得た性能要件の充足を確認するために必要な実行パスをシステムに与えるものである．動作シナリオの作成は，静的解析と設計文書解析によって得られたアーキテクチャ情報や制御構造を参照することで行う．動的振舞解析によって得られる実行トレースデータは，ソフトウェアの動的振舞いのデータであり，解析対象となる関数の呼び出しや復帰とその実行時間の情報，関数を実行するスレッドの識別子，および指定された関数引数の値から構成される．実行トレースデータの取得には，測定による侵襲性が低いシステム測定技術 [21] を使用する．この技術ではハードウェアの出力ポートに対して関連データを直接出力するためのコード断片をソフトウェアに埋め込む．次に，このソフトウェアを実行シナリオによって実システムで動作させる．ハードウェア出力ポートから出力される関連データは，外部の専用ハードウェアによって高速に取得する．これによりシステムの動的振舞いを変化させる影響を少なくできることから，正確な振舞いデータを得ることができる．動的振舞いモデルは，アーキテクチャ情報や制御構造を参照して作成するモデルであり，加えて実行トレースデータから抽出した動的振舞いのデータを含む．システム・リソースモデルは，ハードウェアリソースの構成，リソース使用量，システムの振舞いを高い抽象度で表現する．システム・リソースモデルはシステム仕様とハードウェア・アーキテクチャを元に作成する．その際に補助的な情報として，ソフトウェア・アーキテクチャと制御構造も使用する．この 2 つの情報はソフトウェア仕様書などの技術文書やソースコードの解析結果から獲得する．

3.1 性能評価モデル

性能評価モデルは，動的振舞いモデルとシステム・リソースモデルを統合したモデルであり，シミュレータ [22] 上において，性能評価のシミュレーションに用いる（図 **2**）．

上流工程において，いくつかのアーキテクチャ候補に対して性能を分析するためにはモデルの変更容易性が保証される必要がある．そこで動的振舞いモデルとシステム・リソースモデルが分離された構造を採用する．動的振舞いモデルは，タスクマネージャ・モジュールとパラメータファイルの 2 つの内部モジュールで構成する．パラメータファイルは，実行トレースデータに性能評価の指標を視点とした捨象技術 [23] や情報抽出を適用して得られた動的振舞いのデータであり，関数の呼び出しや復帰と実行時間の情報，リソースの獲得・解放要求情報で構成される外部ファイルである．多様な条件におけるシミュレーションを実施するために，パラメータは外部ファイルとして独立した構造を採用している．またパラメータファイルはソフトウェアの振舞いの変化と実行時間の情報を含むことから，性能評価のシミュレーションにおいて，実行シナリオとしての側面をあわせ持つ．タスクマネージャ・モジュールはソフトウェアの振舞いを表現したモデルであり，パラメータファイルを順次読み込むことで，振舞いの処理ステップを実行する．また，このモジュールはシステム・リソースモデルに対して，リソース獲得の要求通知や振舞い単位の処理時間を送信，リソース獲得の成功・失敗通知や振舞い単位の処理完了の通知を受信する．これらの通知を受信後，次の処理ステップに移行する．システム・リソースモデルは，インタフェース・モジュール，リソース・モジュール，リソース管理モジュールの 3 種類の内部モジュールから構成する．インタフェース・モジュールは，動的振舞いモデルとの間で通知を送受信する．リソース・モジュールは，対象とするハードウェアリソースに対応したオブジェクトであり，リソースの振舞いとリソースの使用量を表現する．リソース管理モジュールは，複数のソフトウェア要素が並行してリソースを使用している状況において，全体としてのリソース使用量を計算する．

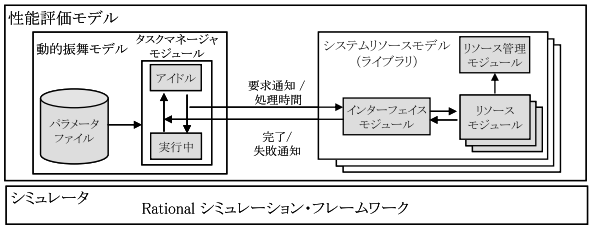

図 **2** 性能評価モデルの構成

Fig. 2 Model configuration to performance estimation.

性能評価モデルを用いた性能評価の対象は，システム全体ならびに処理単位における所要時間と，振舞いに対応したメモリ使用量の推移と最大値である．そのため，所要時間を表現するタイム・リソースモデルと，メモリ使用量を表現するメモリ・リソースモデルの 2 種類のシステム・リソースモデルを用いてそれらを表現する．なお，システム・リソースモデルは外部モデルとして独立させ，さらに性能評価モデルの作成用にライブラリ化する．これにより性能評価モデルの作成において，システム・リソースモデルを個別に作成する必要はない．

3.2 タイム・リソースモデル

タイム・リソースモデルは，システムの時間経過を表現するモデルである（図 3）．このモデルは動的振舞いモデルから受け取った処理時間のデータであるジョブを登録してハードウェアリソース・モジュールに振り分けるジョブキュー，MPU などの各ハードウェアリソースによる時間経過を処理するハードウェアリソース・モジュール，動的振舞いモデルとのインタフェース・モジュール，タイム・リソースモデル全体の処理時間を計算するタイム管理モジュールの 4 種類の内部モジュールで構成する．動的振舞いモデルは，タイム・リソースモデルに対してジョブ登録要求を，シグナルとして送信する．インタフェース・モジュールは，動的振舞いモデルからジョブを受け取り，ジョブキュー・モジュールに登録する．次に登録されたジョブをモジュール内の 1 個，もしくは複数個のハードウェアリソース・モジュールに対してジョブキューがジョブの割当てを行うことで，処理時間の経過をシミュレートする．ジョブの割当てはプロセッサの構成が AMP（Asymmetric Multiprocessing：非対称型マルチプロセッシング）構成の場合，プロセッサを表すハードウェアリソース・モジュールに対して静的に固有のジョブをジョブキューがプロセッサに割り当てる．SMP（Symmetric Multiprocessing：対称型マルチプロセッシング）構成の場合には，プロセッサを限定せず動的にジョブキューがジョブを各プロセッサに割り当てる．タイム・リソースモデルからのジョブ完了通知を動的振舞いモデルに通知することで，動的振舞いモデルは次の処理ステップに移行する．動的振舞いモデルと複数のタイム・リソースモデルを組み合わせることで，プロセッサごとに静的に固有のプログラムを割り当てる AMP 構成におけるシステムの時間経過を，モデルによって表現することを可能とした．さらにタイム・リソースモデル内のプロセッサ並列数と処理時間割当ての制御方法を変更することにより SMP 構成の並列性を表現することが可能である．

タイム・リソースモデルのクラス図においてモデル本体を表す Scheduler クラスは，内部クラスとしてジョブキューを表す JobQueue クラス，ならびにハードウェアリソース・モジュールを表す MPU クラスを持つ．動的振舞いモデルとのインタフェース・モジュールは動的振舞いモデルの基底クラスとして SysModule を定義し動的振舞いモデル・クラスからの継承により，タイム・リソースモデルと動的振舞いモデルの間でシグナルの送受信を実現する．

3.3 メモリ・リソースモデル

メモリ・リソースモデルは，システムの処理単位に割り当てられるメモリリソースの使用量を表現するモデルである（図 4）．このモデルは処理単位に割り当てる個別のメモリ・モジュール，動的振舞いモデルとのインタフェース・モジュール，メモリ管理モジュールの 3 つの内部モジュールで構成する．メモリ・リソースモデルは，動的振舞いモデルからメモリ獲得，もしくはメモリ解放要求を受信する．メモリ・モジュールは，ハードウェリソースであるメモリの消費量を表現するモデルであり，複数のメモリ領域を表現することが可能である．インタフェース・モジュールは，動的振舞いモデルから受信したメモリ獲得もしくはメモリ開放要求に基づいて，メモリ・モジュールに割り当てたメモリ領域の残量からその要求の可否を判断し，要求の成功もしくは失敗を動的振舞いモデルに通知する．動的振舞いモデルは，受信した成功または失敗の通知に応じて次の処理ステップに移行する．

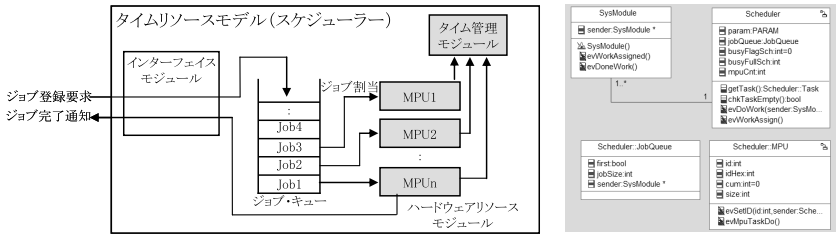

図 3 タイム・リソースモデルの構成とクラス図

Fig. 3 Internal configuration of the time resource model and class diagram.

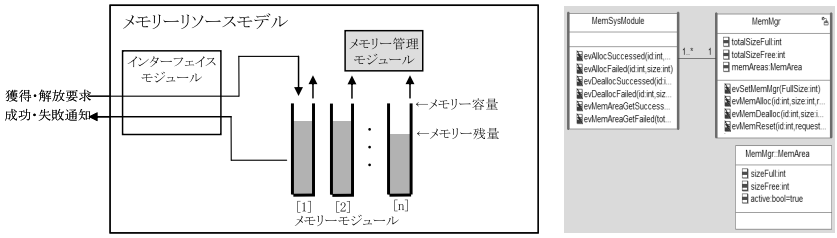

図 4 メモリ・リソースモデルの構成とクラス図

Fig. 4 Internal configuration of the memory resource model and class diagram.

メモリ・リソースモデルのクラス図において，メモリ・リソースモデルの本体を表す MemMgr クラスは，内部クラスとしてメモリ・モジュールを表す MemArea クラスを持つ．動的振舞いモデルに対するインタフェース・モジュールは，タイム・リソースモデルと同様に基底クラス MemSysModule を定義し，動的振舞いモデル・クラスからの継承により，メモリ・リソースモデルとの間でシグナル送受信を可能にする．

3.4 MFP の性能評価モデル

本稿において，モデル・シミュレーションによる性能評価を実施する組込みシステムは MFP である．性能評価の指標は印刷処理の時間とメモリ使用量であり，対象とした機能は印刷機能である．同じく対象とした動作は複数ページの連続印刷である．既存のシステムである現行 MFP に対して静的解析と文書解析を実施した結果，印刷処理は次の 5 つの処理単位で構成され，それらが順次，実行されることを確認した．

i．ホスト PC などから PostScript などのデータを受信する印刷データ受信

ii．データ処理に適した表現に印刷記述言語データを変換する内部表現生成

iii．内部表現をラスタライズ処理により画像データに変換する内部表現処理

iv．画像データを印刷エンジンに送信する印刷準備

v．用紙の処理（印刷エンジンでの印刷，ドラム間移動，排出）

現行 MFP の MPU 構成はシングルプロセッサであり．内部表現処理には 4 つの専用ハードウェア・アクセラレータを用いる（**図 5**）．ここでモデリングの対象となるハードウェアリソースと，ソフトウェアの動的振舞いは印刷処理に関連するものに限定する．

次期 MFP に新しく採用するシステム・アーキテクチャは，AMP 構成の関数呼び出し型マルチコア・アーキテクチャである．また 2 つのプロセッサのハードウェアは同一のホモジニアス構成である．そこで次期 MFP システムに，現行 MFP と同一のプロセッサをサブ MPU として追加する．次に 2 つの MPU 間の通信を目的とするメッセージキューを追加する．オペレーティングシステムは Linux をそれぞれのプロセッサで動作させ，印刷処理のプログラムを実行する．メモリ領域は，メイン MPU のメモリ領域，サブ MPU のメモリ領域，それぞれの MPU からアクセスが可能な共有メモリ領域の 3 つの領域によって構成する（**図 6**）．

次に現行 MFP に対して動的振舞解析を適用し，印刷処理時間のボトルネックの解析を実施した．この解析の結果から，印刷処理における内部表現生成が，印刷処理時間のボトルネックの 1 つであることが判明した．また内部表現生成は，ページ間での相互依存性がないことから複数ページ印刷において，並列処理を適用することにより性能向上が見込める処理である．そこで，内部表現生成の処理を印刷ページ単位で分割，それぞれの MPU に割り当てることで処理を並列化する．ジョブ管理と奇数ページの内部表現生成ならびに内部表現処理はメイン MPU に，偶数ページの内部表現生成をサブ MPU にそれぞれ割り当てる．メイン MPU で動作する内部表現生成のソフトウェアは，その処理対象が奇数ページに限定されることを除いて，現行 MFP と共通である．サブ MPU では内部表現生成のソフトウェアを動作させるために必要となる最小構成のオペレーティングシステムと，偶数ページの内部表現生成のソフトウェアが動作する（図 6）．動的振舞解析に用いた動作シナリオは，社団法人電子情報技術産業協会（JEITA：Japan Electronics and Information Technology）が定める印刷性能の評価用画像データ [24] を用いた 2 ページ連続印刷である．画像データに依存して印刷処理時間が偏よること避けるために，評価用画像データから内部表現生成の処理時間と，印刷処理全体における内部表現生成の処理時間の比率の異なる 5 種類の画像データを組み合わせて用いる．これらの画像データを用いて合計 25 組みの連続印刷を実システムで動作させることで実行トレースデータを取得し動的振舞いモデルを作成した．

現行 MFP の性能評価モデルである現行 MFP モデルは，MPU を表す 1 個のタイム・リソースモデル，内部表現処理に用いる 4 個のハードウェア・アクセラレータを表す 6 個のタイム・リソースモデル，メモリを表す 1 個のメモリ・リソースモデルで構成する（**図 7**）．ハードウェア・アクセラレータには 1 つのタイム・リソースモデルで表す 2 種類のハードウェア・アクセラレータと，2 つのタイム・リソー

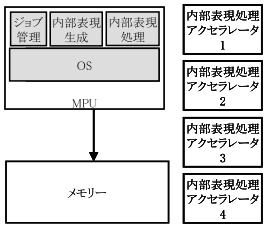

図 5 現行 MFP のシステム構成
Fig. 5 System configuration of baseline MFP.

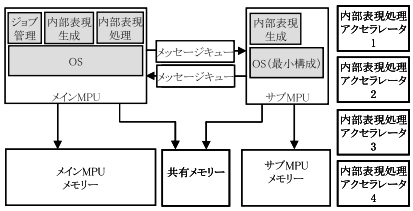

図 6 次期 MFP のシステム構成
Fig. 6 System configuration of new MFP.

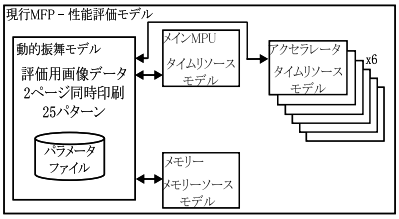

図 7 現行 MFP の性能評価モデル

Fig. 7 Performance estimation model of baseline MFP.

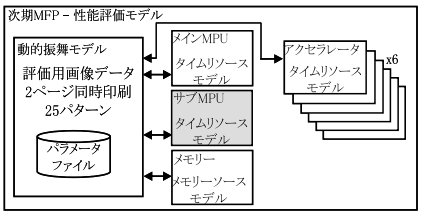

図 8 次期 MFP の性能評価モデル

Fig. 8 Performance estimation model of new MFP.

スモデルで表す 2 種類のハードウェア・アクセラレータがある．次期 MFP の性能評価モデルである次期 MFP モデルは，現行 MFP モデルを再利用することで作成する．次期 MFP モデルに採用するアーキテクチャが AMP 構成のマルチコア・アーキテクチャであることから，現行 MFP モデルに対して，2 つ目のプロセッサとしてタイム・リソースモデルを追加する．メモリ・リソースモデルは複数のメモリ領域を表現することが可能であることから追加を必要とせず，内部モジュールであるメモリ・モジュールの割当てを変更することで次期 MFP モデルへの対応を実現する（図 8）．

4. モデル・シミュレーション

シングルプロセッサ構成である現行 MFP の性能評価モデルと，マルチコア・アーキテクチャを採用した次期 MFP の性能評価モデルを用いたモデル・シミュレーションによる性能評価の結果を報告する．性能評価の目的はマルチコア・アーキテクチャの採用による印刷性能の向上を予測して評価することであり，性能評価の指標は印刷処理時間と印刷処理の時間経過にともなうメモリ使用量の推移，ならびにメモリ使用量の最大値である．

4.1 シミュレーション結果

現行 MFP モデルでは，1 ページ目の内部表現処理が完了した後に，2 ページ目の内部表現生成を開始する（図 9）．一方，マルチコア・アーキテクチャを採用することで内部表現生成を並列化した次期 MFP モデルでは，内部表現生成の開始タイミングを変更して，1 ページ目の内部表現生成と同時に 2 ページ目の内部表現生成を開始する．この並列化の効果により，2 ページ目の排出完了までの印刷処理時間が短縮される．印刷準備，印刷，ドラム間移行，排出の

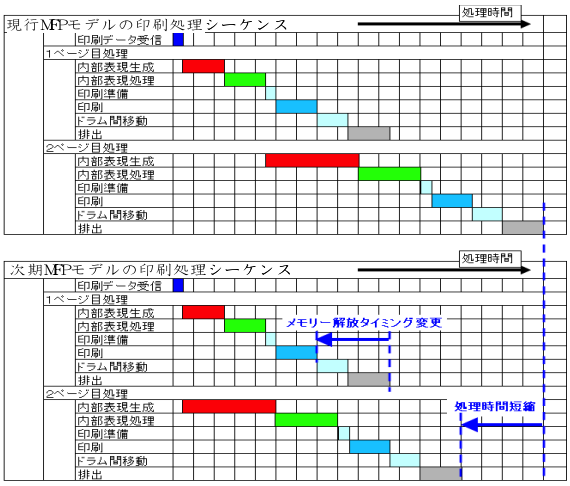

図 9 現行 MFP モデルと次期 MFP モデルのシミュレーション結果比較

Fig. 9 Compare simulation results of baseline MFP and new MFP.

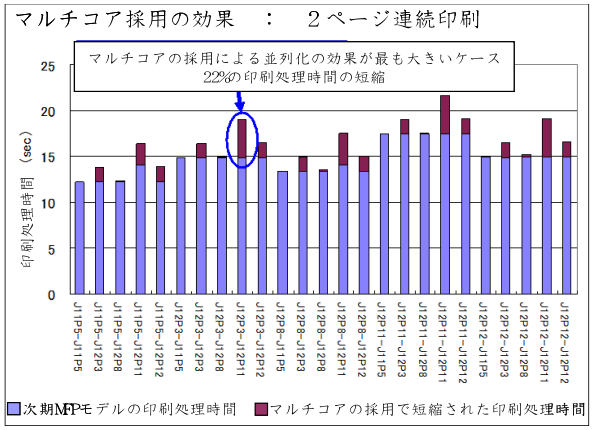

図 10 マルチコア・アーキテクチャの採用による印刷処理時間の短縮

Fig. 10 Reduce the processing time to adoption of multi-core architecture.

処理は MFP に内蔵される印刷エンジンとのインタフェースならびに印刷エンジンの機械機構の動作による制約を受けるため，マルチコア・アーキテクチャの採用による印刷処理時間の短縮は望めない．このシミュレーションにおける印刷処理時間の評価結果により，マルチコア・アーキテクチャの採用による印刷性能の向上は，印刷処理時間の短縮率が最大 22.0%，平均 8.3%であることを確認した（図 10）．メモリ消費量においては次期 MFP モデルにおいて A3 用紙サイズの連続印刷時に，ラスタライズ処理を実行するために必要となるデータ格納用メモリ領域の不足が発生する．すなわちメモリの空き領域の不足により，メモリの確保に失敗して待機状態となるメモリ・ストールが発生することをシミュレーションによって確認した（図 11）．メモリ・ストールの原因は内部表現生成を並列化したことにより，ラスタライズ処理に必要なデータ格納用メモリ領域のサイズが増加したことである．ラスタライズ処理にお

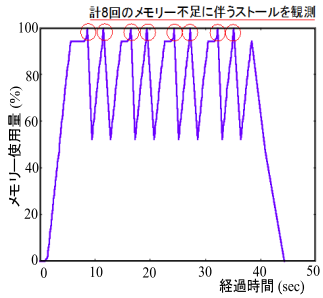

図 11 メモリ使用量の遷移
Fig. 11 Transition of memory usage.

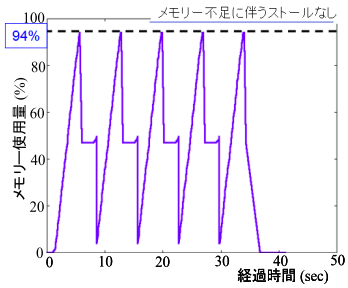

図 12 タイミング変更後のメモリ使用量の遷移
Fig. 12 Transition of the memory usage after changing the timing.

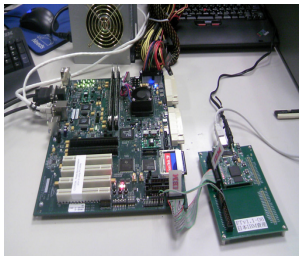

図 13 FPGA を搭載した評価プラットフォーム
Fig. 13 Evaluation platform with FPGA.

けるメモリ消費量は，印刷用紙の大きさと連続印刷の枚数に比例して増加する．図 11 に示すように連続して 10 ページの印刷を行った場合，最初と最後のページを除いて計 8 回のメモリ・ストールが発生する．この問題に対してラスタライズ処理のデータ格納用メモリ領域の開放タイミングを用紙の排出処理の完了後から，印刷の完了後へ変更することによって（図 9），メモリ消費量の最大値はメモリ容量全体の 94%となりメモリ・ストールが解消されることをモデル・シミュレーションによって確認した（**図 12**）．

4.2 有効性の評価

本手法の妥当性検証を，FPGA を搭載した評価プラットフォーム（**図 13**）とモデル・シミュレーションの結果を比較することで実施した．評価プラットフォームは Xilinx 社の FPGA（Field-Programmable Gate Array）である Virtex4-FX を搭載する [25]．評価プラットフォームのハードウェアは 450 MHz で動作する PowerPC® 440 プロセッサ，DDR2 メモリ，プロセッサ・ローカルバスがそれぞれ 2 組で構成されている．2 つの DDR2 メモリはそれぞれのプロセッサに独立して割り当てる．評価プラットフォームをシングルプロセッサ構成に設定して印刷処理を実行することで，現行 MFP モデルのモデル・シミュレーションの結果と比較した．同様に評価プラットフォームをマルチコア構成に設定して次期 MFP モデルのモデル・シミュレーションの結果と比較した．マルチコア構成に設定した場合には，シングルプロセッサ用に設計した印刷処理のプログラムをマルチコア構成で動作可能なように，内部表現生成におけるデータ参照の一部分をプロセッサ間通信に変更した．テストシナリオは JEITA の印刷性能の評価用画像データの中から 6 種類を用いた．比較の対象とした MFP の動作は，評価用画像データの 4 ページ連続印刷である．

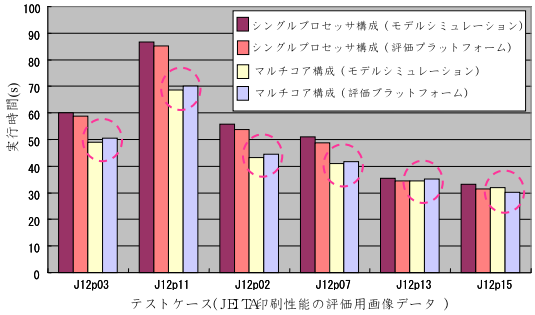

図 14 モデル・シミュレーションと評価プラットフォームとの印刷処理時間の比較
Fig. 14 Comparing the processing time of the model simulation and evaluation platform.

モデル・シミュレーションの結果と，評価プラットフォームで実行した印刷処理時間の実測値との対応を示す（**図 14**）．シングルプロセッサ構成の現行 MFP モデルによるモデル・シミュレーションと，評価プラットフォームの印刷処理時間の誤差は 1.0%から 5.1%である．マルチコア構成の次期 MFP モデルによるモデル・シミュレーションとマルチコア構成に設定した評価プラットフォームの印刷処理時間の誤差は 1.1%から 6.0%であることを確認した．

5. おわりに

本稿では，リバースエンジニアリングの解析結果を用いて組込みシステムを高い抽象度のモデルで表現するリバースモデリング手法と，それらのモデルを用いたモデル・シミュレーションによる性能評価手法を提案，本手法を実際の組込みシステムであるマルチ・ファンクション・プリンタに適用した．リバースモデリング手法を適用することにより，開発の上流工程であるアーキテクチャ設計の段階と同様の高い抽象度でシステムを表現することを可能とした．

次に，それらのモデルを用いたモデル・シミュレーションにより，性能評価を実現した．リバースモデリング手法

で作成したモデルを再利用することで次期システムの性能評価を実施することで，マルチコア・アーキテクチャの採用による性能向上の有効性を示した．さらにマルチコア・アーキテクチャの採用にともなうメモリ消費の変化を評価して，システムに搭載するメモリ量を現行システムから増加させることなく性能向上が実現できることを示した．性能評価のモデル・シミュレーションと評価プラットフォームとの印刷性能の比較においては，印刷処理時間の誤差が6.0%未満という結果が得られた．

これより，再利用を目的に現行システムに変更を必要としない構成要素をそのまま用いる差分開発のアーキテクチャ設計において，システムを構成する各処理単位の動作に変更のない条件で，それらの処理単位の実行を制御する方法を現行システムから変更した次期システムの性能を予測する手法として効果的であると考える．差分開発は近年の組込みシステムに用いられることが多いことから，本稿で検証した範囲での性能評価用途で十分な実用性があると考える．ただし，処理単位の動作に変更がある場合や並列に動作する処理単位に相互作用がある場合の性能評価への適用には別の考え方を取り入れる必要があると考える．

今後の課題として，MFP におけるファックス機能やスキャナ機能を含めた統合動作のモデルによる再現と性能評価，さらにシステムの性能を大きく左右するバス接続や通信の帯域に加えて，消費電力量をモデルで表現することでモデル・シミュレーションによる組込みシステムの総合的な評価手法の確立が重要であると考える．

謝辞 本稿の執筆に際し，多くのご支援とご助言をいただいた京セラドキュメントソリューションズの福岡直明氏に深謝いたします．

参考文献

[1] 経済産業省：2004 年版組込みソフトウェア産業実態調査報告書，入手先 ⟨http://www.meti.go.jp/policy/it_policy/technology/softjittaityousa-gaiyou.pdf⟩.

[2] 上野秀剛，亀井靖高，門田暁人，松本健一：原価率とプロジェクトメトリクスに着目したソフトウェア開発プロジェクトの特徴分析，プロジェクトマネジメント学会誌，Vol.12, No.5, pp.25–30 (2010).

[3] Khusidman, V.: Architecture-Driven Modernization: Transforming the Enterprise DRAFT V.5, available from ⟨http://adm.omg.org/⟩.

[4] Ono, K., Toyota, M., Kawahara, R., Sakamoto, Y., Nakada, T. and Fukuoka, N.: A modeling method for performance analysis in model-driven development, *Proc. 13th Design, Automation and Test in Europe* (*DATE 2010*) (2010).

[5] Ono, K., Toyota, M., Kawahara, R., Sakamoto, Y., Nakada, T. and Fukuoka, N.: A Model-based Method for Evaluating Embedded System Performance by Abstraction of Execution Traces, *Proc. 6th European Conference on Modeling Foundations and Applications* (*ECMFA 2010*), pp.233–244, Springer (2010).

[6] Sakamoto, Y., Nagano, T., Nakada, T., Ono, K., Hisazumi, K. and Fukuda, A.: Development of Embedded Systems Using Reverse Engineering and Model-based Performance Evaluation, *Proc. 5th International Conference on Project Management* (*ProMAC2010*), pp.160–168 (2010).

[7] 藤井将人，横森励士，山本哲男，井上克郎：動的情報を利用したソフトウェア部品評価手法の提案と評価，電子情報通信学会技術研究報告，SS2002-42, pp.31–36 (2003).

[8] Hsu, J.M. and Banerjee, P.: Performance measurement and trace driven simulation of parallel CAD and numeric applications on a hypercube multicomputer, *IEEE Trans. Parallel and Distributed Systems*, Vol.3, No.4, pp.451–464 (1992).

[9] Prete, A.M., Prina, G. and Ricciardi, L.: A trace-driven simulator for performance evaluation of cache-based multiprocessor systems, *IEEE Trans. Parallel and Distributed Systems*, Vol.6, No.9, pp.915–929 (1995).

[10] Object Management Group, available from ⟨http://www.omg.org/⟩.

[11] Kawahara, R., Nakamura, K., Ono, K., Nakada, T. and Sakamoto, Y.: Coarse-Grained Simulation Method for Performance Evaluation a of Shared Memory System, *Proc. 16th Asia and South Pacific Design Automation Conference* (*ASP-DAC 2011*), pp.413–418 (2011).

[12] SESSAME WG2：組込みソフトウェア開発のためのリバースモデリング手法，翔泳社 (2007).

[13] Tonella, P. and Potrich, A.: Reverse engineering of the interaction diagrams from C++ code, *Proc. 19th International Conference on Software Maintenance* (*ICSM 2003*), pp.159–168, IEEE Computer Society (2003)

[14] Briand, L.C., Labiche, Y. and Leduc, J.: Toward the reverse engineering of UML sequence diagrams for distributed Java software, *IEEE Trans. Softw. Eng.*, Vol.32, No.9, pp.642–663 (2006).

[15] Kempf, T., Karuri, K., Wallentowitz, S., Ascheid, G., Leupers, R. and Meyr, H.: A SW performance estimation framework for early system-level-design using fine-grained instrumentation, *Proc. 9th Design, Automation and Test in Europe* (*DATE 2006*), pp.468–473 (2006).

[16] Posadas, H., Herrera, F., Sanchez, P., Villar, E. and Blasco, F.: System-level performance analysis in SystemC, *Proc. 7th Design, Automation, and Test in Europe* (*DATE 2004*), pp.378–383 (2004).

[17] Reyes, V., Kruijtzer, W., Bautista, T., Alkadi, G. and Nunez, A.: A unified system-level modeling and simulation environment for MPSoC design: MPEG-4 decoder case study, *Proc. 7th Design, Automation and Test in Europe* (*DATE2006*), pp.474–479 (2006).

[18] Madl, G., Dutt, N. and Abdelwahed, S.: Performance estimation of distributed real-time embedded systems by discrete event simulations, *Proc. 7th ACM Conference on Embedded Systems Software* (*EMSOFT2007*), pp.183–192 (2007).

[19] Paul, J.M., Thomas, D.E. and Cassidy, A.S.: High-level modeling and simulation of single-chip programmable heterogeneous multiprocessors, *ACM Trans. Design Automation of Electronic Systems* (*TODAES*), pp.431–461 (2005).

[20] Martin, G. and Müller, W.: *UML for SOC Design*,

Springer-Verlag (2005).
[21] Ohba, N. and Takano, K.: Hardware debugging method based on signal transitions and transactions, *Proc. 11th Asia and South Pacific Design Automation Conference* (*ASP-DAC 2006*), pp.454–459 (2006).
[22] IBM Rational Rhapsody, available from ⟨http://www-06.ibm.com/software/jp/rational/products/rhapsody/productline/⟩.
[23] 高鋪 守, 久住憲嗣, 小野康一, 河原 亮, 坂本佳史, 中田武男, 長野 正, 福田 晃：組込みソフトウェアにおける実行トレースに基づく捨象モデリングツールの実装と評価, 情報処理学会研究報告, ユビキタスコンピューティングシステム, pp.255–262 (2010).
[24] 社団法人電子情報技術産業協会：プリンタ用標準テストパターン, IT-3011, available from ⟨http://www.jeita.or.jp/cgi-bin/standard/pdf.cgi?jk_n=518&jk_pdf_file=IT-3011_j.pdf⟩.
[25] Xilinx, Virtex-4 Family Overview, available from ⟨http://www.xilinx.com/support/documentation/data_sheets/ds112.pdf⟩.

坂本 佳史 （正会員）

ICP-エグゼクティブ・プロジェクトマネージャー．1985 年日本アイ・ビー・エム株式会社に入社．組込みシステム開発プロジェクトに従事．2012 年九州大学大学院システム情報科学府後期博士課程単位取得退学．プロジェクトマネジメント学会会員．

小野 康一 （正会員）

1990 年から 1992 年まで早稲田大学情報科学研究教育センター助手．1994 年早稲田大学大学院理工学研究科後期博士課程単位取得退学．同年日本アイ・ビー・エム株式会社東京基礎研究所．2003 年から 2006 年まで日本ソフトウェア科学会理事．ソフトウェア検証，移動エージェント，Web アプリケーション開発支援，組込みシステムのモデル駆動開発，等の研究に従事．日本ソフトウェア科学会，IEEE-CS，ACM 各会員．

中田 武男 （正会員）

1984 年東北大学工学部通信工学科卒業．1986 年東北大学大学院情報工学専攻修士課程修了．同年日本アイ・ビー・エム株式会社入社．東京基礎研究所にて並列処理システム・アーキテクチャ，並列処理アルゴリズム，マルチプロセッサ用キャッシュの研究のほか，Notebook 型 PC 省電力設計，ASIC/SoC 設計・検証の技術開発に従事．IEEE，電子情報通信学会各会員．

安浦 寛人 （正会員）

京都大学大学院工学研究科修了．1986 年京都大学工学部助手．九州大学大学院システム情報科学研究院長，システム LSI 研究センター長等を経て，2008 年理事・副学長に就任．