リスポンシブシステム技術とその応用

角田 良明，楠本 真二，菊野 亨

1. まえがき

鉄道制御システム，銀行オンラインシステム等，フォールトの発生のいかんにかかわらずタイムリーな情報通信サービスの提供を保証する情報通信システムの必要性が認識されてきている．このようなシステムをリスポンシブシステム(Responsive System)と呼ぶ．リスポンシブシステムの実現に向けて，フォールトトレラントシステムとリアルタイムシステムの2つの分野の研究者は各々独立して研究を推進してきた．フォールトトレラントシステムの研究者は，空間を重視してシステムの構成を議論する場合が多い．一方，リアルタイムシステムの研究者は，時間に注意を払ってサービスを考察する傾向が見受けられる．そのため，システムに基づいたサービスの提供，あるいはサービスに基づいたシステムの開発が行われてきた．これでは，リスポンシブシステムとして不十分である．空間と時間の両方の観点から考察し，システムとサービスに関する統合的品質基準を満たすようにする必要がある．この品質基準の代表例としてリスポンシブネスがある．これは，システムにフォールトが発生しても要求された情報通信サービスを指定された時間内に終了できる確率と定義される．従って，リスポンシブネスを最適化するように，リスポンシブシステムを実現することが重要課題となる．

1990年テキサス大学オースチン校のMalek教授がリスポンシブシステムの概念を初めて提唱している．リスポンシブシステムは，並列／分散システム環境でフォールトトレラントシステムとリアルタイムシステムの機能を統合したものである[6][10][14][15]．本稿ではリスポンシブシステム技術とその応用について述べる．

本稿の構成は次の通りである．2節ではリスポンシブシステムの定義とその品質を表すリスポンシブネスについて述べる．3節ではリスポンシブシステムの一般的な設計法について述べる．4節ではリスポンシブマルチプロセッサシステムについて述べる．5節ではリスポンシブプロトコルについて述べる．6節ではリスポンシブソフトウェアプロセスについて述べる．7節では本稿のまとめと今後解決すべき研究課題について述べる．

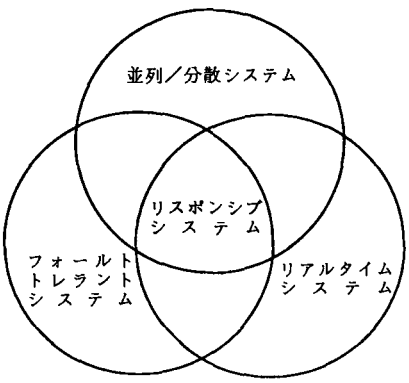

図1 リスポンシブシステム

2. リスポンシブシステム

2.1 リスポンシブシステムの定義

リスポンシブシステムは，並列／分散システム環境においてフォールトトレランス機能とリアルタイム機

かくだよしあき，くすもとしんじ，きくのとおる 大阪大学基礎工学部 〒560 豊中市待兼山町 1-3

能の両者を統合した機能を持つシステムと定義される(図1参照). 言い替えれば, フォールトが存在しても期待される情報通信サービスをタイムリーに実行するシステムである. リスポンシブシステムでは, フォールトからの回復処理時間を含めてシステム仕様を設計することにより, フォールトが存在した場合における情報通信サービスの実行状況を予測することが重要である.

2.2 リスポンシブシステムの品質

リスポンシブネス (Responsiveness) は, リスポンシブシステムの品質を評価する尺度である [6][14][15]. リスポンシブネス $r(t)$ はシステムにフォールトが起こっても要求された情報通信サービスを指定された時間内に終了できる確率であり, 次のように関数 f を使って形式的に定義される.

$$r(t) = f[p(t),\ a(t)]$$

ここで, $p(t)$ はシステムにフォールトが起こっても要求された情報通信サービスの実行を継続できる確率であり, $a(t)$ はシステムにフォールトが起こっても要求された情報通信サービスの実行を継続できるという仮定のもとに, 要求された情報通信サービスを指定された時間内に終了できる確率である.

今, 情報通信サービスの実行が $n(\geq 1)$ 個のタスクを独立に実行することにより実現されるとする. タスクにはそれぞれデッドラインが指定されており, それまでに実行の終了が要求されているとする. すると, 各タスクのリスポンシブネスの平均として, $r(t)$ の f は次のように具体的に定まる.

$$r(t) = [p_1(t) \cdot a_1(t) + \cdots + p_n(t) \cdot a_n(t)]/n$$

ここで, $p_i(t)(1 \leq i \leq n)$ はシステムにフォールトが起こってもタスク i の実行を継続できる確率であり, $a_i(t)(1 \leq i \leq n)$ はシステムにフォールトが起こってもタスク i の実行を継続できるときにタスク i の実行がその指定されたデッドラインまでに終了できる確率である. このように $r(t)$ はタスクに着目して定義することができる. しかし, $r(t)$ はタスクスケジューリング, システムアーキテクチャ, 通信機構などに依存するため, $r(t)$ を一般的に求めることは困難であり, 特定のシステムに対して求めていくことになると予想される.

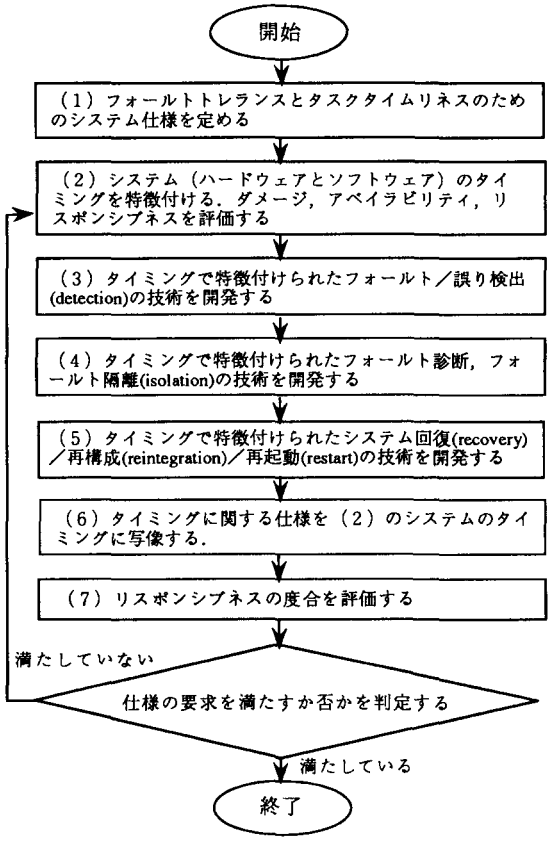

図2 リスポンシブシステムの設計法

3. リスポンシブシステムの設計法

システムの構成要素にフォールトが存在してもシステム障害に至らないという特性を備えたシステムをフォールトトレラントシステム [18] という. 一方, リアルタイムシステム [19][23][24] は, 決められた時間内に情報通信サービスの実行を終了することが要求されているシステムをいう. フォールトトレラントシステムの設計者は時間を低廉なリソースと考え, 空間の冗長度を最小化する傾向がある. 一方, リアルタイムシステムの設計ではしばしば空間を犠牲にして時間に関する項目に注意を払うことが要求される. リスポンシブシステムの設計では, 空間と時間の両者が考慮すべき要因となる.

リスポンシブシステムの設計法では，リスポンシブシステムをデッドラインを越えたものをフォールトと考えるフォールトレラントシステムとして設計する方法(方法FTS)と，フォールトへの対処も仕様の一部として考えるリアルタイムシステムとして設計する方法(方法RTS)が考えられる．フォールトレラントシステム技術は空間の冗長を活用することが多いので，方法FTSは実現が困難と思われる．これに対して，フォールトへの対処も仕様の一部として考えることによりリアルタイムシステム技術を活用することができるので，方法RTSは比較的実現可能と思われる．

しかし，個別のフォールトのクラスに対して，フォールトの診断，フォールトからの回復などの時間を求めることが方法RTSにとって不可欠となる．それができれば，フォールトへの対処時間を含めた仕様に対して正確なデッドラインの検証を行うことができる．リスポンシブシステムの一般的な設計手順を図2に示す．

4. リスポンシブマルチプロセッサシステム

4.1 再構成

本稿では，プロセッサは全て同じ種類であり，プロセッサにおけるフォールトの発生及びフォールトからの回復に伴う再構成によりプロセッサの数が増減するマルチプロセッサシステムを議論の対象とする．このようなシステムをリスポンシブシステムとしてモデリングする場合，フォールトトレラントシステムの特徴であるフォールトの発生及びフォールトからの回復の挙動だけでなく，リアルタイムシステムの特徴であるデッドライン違反についても表現しなければならない．

デッドライン違反を明確に考慮した研究は，Shinら[20]およびMuppalaら[17]によって行われている．これらの研究では，ジョブが処理されている間，システム構成は変化しないという仮定が置かれていた．システムの再構成には，プロセッサの切替やジョブの再割り当て等の作業の時間が必要であり，この仮定を置く限り，リスポンシブマルチプロセッサシステムの正確な性能評価はできない．文献[21][22]では，この仮定を除き，再構成のための時間を考慮した場合のリスポンシブマルチプロセッサシステムのモデリングと性能評価法を提案している．

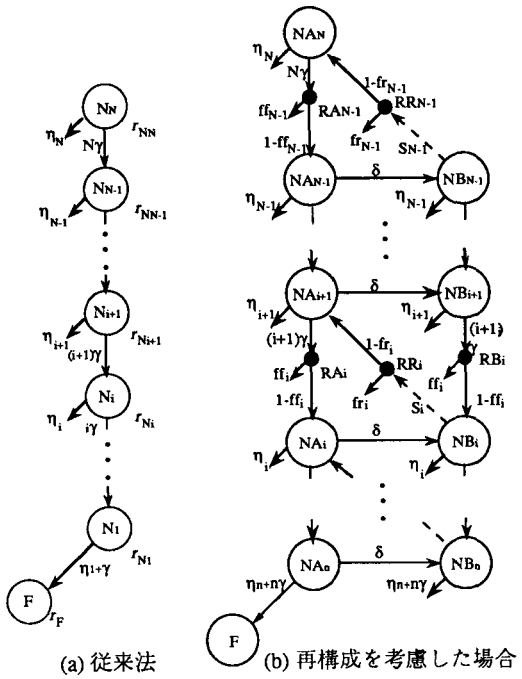

図3 リスポンシブマルチプロセッサシステムのモデル

4.2 モデリングと性能評価

典型的なフォールトトレラントシステムのモデリングでは，プロセッサにおけるフォールトの発生及びフォールトからの回復の過程をマルコフ連鎖で表し，その各状態にリウォードを付加したマルコフリウォードモデルが用いられていた．リスポンシブシステムのモデリングでは，フォールトの発生及びフォールトからの回復の過程に加えて，デッドライン違反を表現する必要がある．文献[17][20]では，ハードデッドラインを，マルコフリウォードモデルに，デッドライン違反によるシステム障害への遷移を付加することで表現している．

図3(a)に従来法のモデルを示す．同図において，状態N_iはi台のプロセッサでシステムが稼働している状態を，状態Fはシステム障害の状態を表している．プロセッサにおけるフォールトの発生及びフォールトからの回復はそれぞれレートγ，δに従っており，状態N_iにおけるデッドライン違反のレートはη_iである．しかし，図3(a)の従来法のモデルには再構成の作業が明

確に表現されていない．

そこで，文献[21][22]では，このモデルを一般化した図3(b)のモデルを提案している．同図において，状態N_iを，フォールトが検出され切り放されたプロセッサが回復中の状態NA_iと，回復が完了して再構成によりシステムに追加可能な状態NB_iに分けている．プロセッサの追加という再構成は点線の遷移で表している．この遷移は具体的な再構成法により詳細化される．黒丸の状態は再構成を表す．状態RA_i，状態RB_iはプロセッサのフォールトによる再構成を表し，状態RR_iはプロセッサの追加による再構成を表している．ff_i，fr_iはデッドライン違反を引き起こす確率である．

このようなモデルを用いることにより，リスポンシブマルチプロセッサシステムの再構成を考慮したモデリングが可能となる．このモデリングに基づく正確な性能評価によって，再構成が可能であればすぐに実行するだけでなく，システムの負荷が軽くなるまで待ってから再構成を実行した方が良い場合もあることを明らかにしている[21]．

5. リスポンシブプロトコル

5.1 リスポンシブプロトコルの定義

リスポンシブシステムの設計の一例として通信プロトコルの設計[13]について述べる．実際の通信プロトコルではフォールトが発生すると例外処理のルーチンで回復するように設計されている場合が多い。この方法では、フォールトの発生場所および種類に対して個別の例外処理のルーチンが用意されなければならず設計効率が極めて悪い。これに対して、例外処理のルーチンを用いずに、フォールトにより陥る異常状態から正常状態へのリアルタイムな回復機能を実現する通信プロトコルは，リスポンシブプロトコル[2][3][5][7][8][11]という．つまり，フォールトトレランス性とリアルタイム性の両方の性質を合わせ持つ通信プロトコルである．リスポンシブプロトコルの利点は，通信路の遅延，メッセージの転送エラー，プロセスの一時的なフォールトなどが起こっても例外処理ルーチンを用いずに，あらかじめ与えられた時間内にプロセス間の同期を回復すること，ネットワークのグローバルな状態を把握することなくプロセスの初期化を行なうことなどである．

5.2 リスポンシブプロトコルの設計法

リスポンシブプロトコルの一般的な設計法は次の通りである．

(1) プロトコルシステムの静的側面を設計する．具体的には，プロセス構成とプロセス間で送受されるメッセージを定める．

(2) プロトコルシステムの動的側面を設計する．具体的には，正常時ならびに準正常時のメッセージの送受信による遷移系列を定める．ここで，準正常時とは呼設定の途中で呼解放が開始される場合などを表す．

(3) 異常時のメッセージの送受信による遷移系列を定める．

(4)(2)および(3)の系列が，あらかじめ与えられた時間内に実行される否かを検証する(ハードデッドラインの検証)．

(5)(2)および(3)の系列に対して，プロトコルシステムのリスポンシブネスを評価し，プロトコル設計者の要求するリスポンシブネスを満たすか否かを検証する(ソフトデッドラインの検証)．

(6)(4)および(5)の条件が満たされていなければ，(1)～(5)を繰り返す．

5.3 リスポンシブプロトコルの検証

リスポンシブプロトコルの検証が，5.2で述べたリスポンシブプロトコルの設計法のステップ(4)で必要となる．

文献[7]では，プロトコルシステムの状態を述語で表したプロトコル検証法とマルチプロセッサシステムのためのタスクスケジューリング法を組み合わせたリスポンシブプロトコルの検証法を提案している．本検証法の特長は，変数の値域が有限ではない通信プロトコルに対する自動検証を可能としていることである．双方向ハンドシェイクプロトコルへの適用例を与え，その有効性を示している．

フォールトトレランス性については，プロトコルシステムの状態を述語で表し，述語で表された任意の異常状態から正常状態への遷移系列が存在するか否かを調べることにより判定する．

リアルタイム性については，全ての実行可能遷移系列が，あらかじめ与えられた時間内にその実行を終了

するか否かを，通信プロトコルのプロセスと遷移をそれぞれマルチプロセッサシステムのプロセッサとタスクに対応させたタスクスケジューリング問題に帰着させて解く．

5.4 リスポンシブネスの評価

5.2で述べたリスポンシブプロトコルの設計法のステップ(5)では，リスポンシブプロトコルのリスポンシブネス評価を行う．

文献[9]では，チェックポインティング／ロールバックリカバリ機能を追加することにより，通信プロトコルをリスポンシブプロトコルに変換する方法が提案されている．文献[8]では，このような通信プロトコルのリスポンシブネスの評価法を提案している．

通信プロトコルのリスポンシブネスは，一般に時間 t までに最終状態に到達する確率と定義することができる．通信プロトコルで起こるフォールトは，チェックポインティング／ロールバックリカバリ機能で回復可能なフォールトとそうでないフォールトに分類することができる．前者のフォールトが初期設定の誤り，メッセージの転送エラー，ソフトウェアのバグであり，後者のフォールトがハードウェアの故障，ロールバックリカバリ機能の誤りである．

通信プロトコルでは，これら2種類のフォールトは関連性が無いため，通信プロトコルのリスポンシブネス $RS(t)$ を次のように2つの部分に分けて独立に計算することができる．

$$RS(t) = R(t) \times P(t)$$

ここで，$R(t)$ は時間 t までに通信プロトコルに回復不可能なフォールトが起こらない確率，$P(t)$ は時間 t までに通信プロトコルに回復不可能なフォールトが起こらないという仮定のもとに通信プロトコルが時間 t までに最終状態に到達する確率である．文献[8]では，可到達グラフにループが存在しない通信プロトコルのリスポンシブネスの具体的な計算法を示している．

6. リスポンシブソフトウェアプロセス

1987年にOsterweilにおって提唱された“ソフトウェアプロセスもまたソフトウェアである”という考えから始まったソフトウェアプロセスの研究は，現在のソフトウェア工学における主要なテーマの一つとなっている．リスポンシブシステムの概念から考えると，リスポンシブソフトウェアプロセスは，“異常が発生しても期限までにユーザの要求を満足するソフトウェアを開発することができる機能を持つプロセス”と定義される．

通常このようなプロセスは，プロジェクト管理技術によって実現されている．プロジェクト管理とは，ソフトウェアに対する経済的，時間的，品質的要求を開発計画として具体化し，その計画にしたがってソフトウェア開発を最適にコントロールする技術である．しかし，不十分な開発計画，開発途中でのユーザの要求変更，開発者の技術不足等によって開発の遅れは頻繁に発生している[1]．

開発プロセスを効率良くコントロールするためには，(1)開発プロセスの現状把握と分析，(2)分析結果に基づく現状の改善策の作成とフィードバックによる開発プロセスの正常状態への復帰，が必要である．更に，こうしたプロセスの制御を効果的に実行するためには，(1)の現状把握と分析を定量的かつ客観的に行うことが望まれる．その前提となるのが，いわゆる計測である．しかし，ソフトウェア開発プロセス全体を系統的に計測するための形式的な手法はほとんど確立していない．

文献[16]では，ソフトウェア開発プロセスで計測されるすべての作業を数学モデルであるペトリネットで表現し，(1)プロセスのモデル化，(2)評価尺度の決定，(3)データ計測方法の実現，(4)データの収集と分析，を一貫して行う枠組を提案している．また，枠組の有効性を実験的に評価している．

一方，プロジェクト管理技術に加えて，フォールトトレラント技術をソフトウェア開発プロセスに導入することで，プロセスのリスポンシブ化を目指す研究も行われている．文献[12]では，ハードウェア向けに提案されているフォールトトレラント技術に対応して，ソフトウェア開発プロセスに対するフォールトトレラント技術として，(1)品質データの収集，(2)レビュー，(3)フォールト除去，(4)品質データに基づくプロセスの選択，(5)プロセスの制御，を新しく導入している．具体的には，(2)と(3)の技術の活用により障害の波及を最小限度の範囲にとどめることが可能となり，プロダクトの品質の直接的な改善が達成される．一方，

(1), (4), (5) の技術の活用によってソフトウェア開発プロセスの(従って，プロダクト品質の間接的な)改善が実現される．この中で，特に (4) の技術はいわゆるNバージョンプログラミングに対応するもので，最も有効なリスポンシブ化手法と考えられている．

導入した5つの技術の有効性を確認するために，ある企業における大規模ソフトウェア開発にこれらの技術を適用して実験的な評価を試みている．その結果，これら5種類のフォールトトレラント技術の適用によって，フォールト除去作業の効率の向上，開発作業の生産性の向上が可能であることが確認されている．

7. あとがき

本稿では，並列／分散システム環境においてフォールトトレラントシステムとリアルタイムシステムの機能を統合したリスポンシブシステムについて紹介した．まず，リスポンシブシステムとその品質を表すリスポンシブネスの定義を与えた．次に，リスポンシブシステムの一般的な設計法を示した．最後に，リスポンシブシステムの具体例として，リスポンシブマルチプロセッサシステム，リスポンシブプロトコル，リスポンシブソフトウェアプロセスについて述べた．

リスポンシブシステム技術は，信頼性と性能の両面で最適化された情報通信サービスを提供する情報通信システムの基盤となる技術である．1991年よりInternational Workshop on Responsive Computer Systemsが毎年開催され，リスポンシブシステムに関する研究の進展に貢献している [4]．リスポンシブシステムには仕様記述，設計法，言語，アルゴリズム，プロトコル，品質評価などの基本的事項ならびに様々な分野への応用など数多くの研究課題が残されており，その解決が望まれている．

謝辞 本研究の一部は平成 6 年度文部省科学研究費補助金一般研究 (C)(課題番号 06650414)，一般研究 (C)(課題番号 06650413)，奨励研究 (A)(課題番号 06780260) の補助を受けている．また，本研究の実施にあたりEAGL事業推進機構より助成を受けた．

参考文献

[1] M. V. Genuchten: "Why is software late? An empirical study of reason for delay in software development", Trans. on IEEE Softw. Eng., 17, 8, pp.582-590, 1991.

[2] H. Igarashi, Y. Kakuda and T. Kikuno:"Synthesis of protocol specifications for design of responsive protocols," IEICE Transactions on Information and Systems, E76-D, 11, pp.1375-1385, Nov. 1993.

[3] J.M. Jaffe and F.H. Moss:"A responsive distributed routing algorithm for computer networks", Proc. of International Conference on Distributed Computing Systems, pp.348-353(1981).

[4] 角田:"The Third International Workshop on Responsive Computer Systems の参加報告," 1994 年実時間処理に関するワークショップ (RTP'94), 信学技報 CPSY93-53, March 1994.

[5] Y. Kakuda and T. Kikuno:"Issues in responsive protocols design," Proc. 2nd International Workshop on Responsive Computer Systems (RCS'92), Oct. 1992, Dependable Computing and Fault-Tolerant Systems, 7, pp.17-26, Springer-Verlag, 1993.

[6] 角田, 菊野:"リスポンシブシステム", 計測と制御, 32, 9, pp.750-758, Sept. 1993.

[7] Y. Kakuda, T. Kikuno and K. Kawashima, :"Automated verification of responsive protocols modeled by extended finite state machines," Real-Time Systems Journal, 7, 3, pp.257-289, Nov. 1994.

[8] Y. Kakuda, T. Sugasawa and T. Kikuno:" Responsiveness evaluation of a class of communication protocols", Proc. the 1994 IEEE Workshop on Fault-Tolerant Parallel and Distributed Systems, to appear.

[9] Y.Kakuda, T.Kikuno, M. Malek and H. Saito:"A unified approach to design of responsive protocols," Proc. the 1992 IEEE Workshop on Fault-Tolerant Parallel and Distributed Systems, pp.8-15, July 1992.

[10] H. Kopetz and Y. Kakuda (eds.):"Responsive Computer Systems," Dependable Computing and Fault-Tolerant Systems, 7, Springer-Verlag, 1993.

[11] H. Kopetz and G. Grunsteidl:"TTP - A time-triggered protocol for fault-tolerant real-time systems", Proc. of 23rd International Symposium on Fault-Tolerant Computing, pp.524-533, 1993.

[12] S. Kusumoto, K. Matsumoto, T. Kikuno and K. Tanaka: "Improvement of software development process by using fault tolerant techniques", Journal of Computer Systems Science & Engineering, 9, 2, pp.83-88, 1994.

[13] M. T. Liu:"Protocol engineering", Advances in Computers, 29, pp.79-195, Academic, 1989.

[14] M. Malek:"Responsive systems (A challenge for the nineties)," Proc. EUROMICRO'90, 16th Symposium on Microprocessing and Microprogramming, Keynote Address, Amsterdam, pp.9-16, August 1990, North-Holland.

[15] M. Malek:"Responsive systems: A marriage between real time and fault tolerance," Proc. 5th International GI/ITG/GMA Conference on Fault-Tolerant Computing Systems, Keynote Address, Nurnberg, pp. 1-17, Sept. 1991, Springer-Verlag.

[16] K. Matsumoto, S. Kusumoto, T. Kikuno and K. Torii: "A new framework of measuring software development processes", IEEE-CS International Software Metrics Symposium, pp.108-118, 1993.

[17] J. K. Muppala, S. P. Woolet and K. S. Trivedi:" Real-time systems performance in the presence of failures," COMPUTER, 24, 5, pp.37-47, May 1991.

[18] 南谷崇:フォールトトレラントコンピュータ, オーム社, 1991.

[19] H. Shimakawa, H. Ohnishi, I. Mizunuma and M. Takegaki:"Acquisition and service of temporal data for real-time plant monitoring", Proc. of Real-Time System Symposium, pp.112-118, 1993.

[20] K. G. Shin, and C. M. Krishna:"New performance measures for design and evaluation of real-time multiprocessors," Computer Systems Science and Eng., 1, 4, pp.179-192, Oct. 1986.

[21] 土屋, 陳, 角田, 菊野:"再構成可能なリスポンシブシステムのモデリングと性能評価", 1994 年実時間処理に関するワークショップ (RTP'94), 信学技報, CPSY93-57, March 1994.

[22] T. Tsuchiya, C. Chen, Y. Kakuda and T. Kikuno:"Calculating performability measures of responsive systems", Proc. of Second ISSAT International Conference on Reliability and Quality in Design, to appear.

[23] 米田友洋 (編): リアルタイムシステム特集, 情報処理, 35, 1, 1994.

[24] T. Yoneda, A. Shibayama, H. Schlingloff and E.M. Clark:"Efficient verification of parallel real-time systems", LNCS697 Computer Aided Verification, pp.321-332, 1993.