社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report

飛行船自動航行ソフトウェアの事例による設計段階でのスループット性能検証手法の評価

嘉戸 彰　　中田 明夫

広島市立大学　大学院情報科学研究科 システム工学専攻
E-mail:　{akira@sos.info. , nakata@}hiroshima-cu.ac.jp

あらまし　組込みシステム開発において，設計段階で厳しいリソース制約の下で性能要求を満たすか否かを判別するのは一般に困難である．このため我々は，タスク間の制御構造を記述したマルチタスクソフトウェアの設計モデル，および，プロセッサやバスなどのリソース情報から，指定したスループット要求を満たすか否かの検証を行う手法（スループット性能検証手法）を従来提案している．しかし，当該手法では検証にかかる計算量を抑えるため，タスク内部の詳細な振る舞いを捨象しており，検証結果と実装での性能との間に差異が生じる可能性がある．本研究では，ある程度の規模で構成され具体的な機能を実現しているソフトウェア設計事例として，飛行船自動航行ソフトウェアの設計事例を取り上げ，性能検証による性能見積もり結果と性能の実測値との間にどの程度差が生じるかを評価し，当該手法の有用性について評価検討する．

キーワード　組込みシステム，マルチタスクソフトウェア，設計モデル，性能検証，スループット

Throughput Performance Verification of Airship Autopilot Software: A Case Study for Evaluating Performance Verification of Software Design Model

Akira KADO　　and　Akio NAKATA

Department of Systems Engneering, Graduate School of Information Sciences, Hiroshima City University
E-mail:　{akira@sos.info. , nakata@}hiroshima-cu.ac.jp

Abstract　In the development of embedded systems, it is generally difficult to check, in the design phase, whether or not a designed system meets its performance requirement under its severe resource constraints. So far, we have proposed a method (throughput performance verification) to check whether a multitask software specification with resource information (processor, bus, etc.) meets a given throughput performance requirement. However, in the method, in order to reduce verification complexity, the detailed behaviors inside the tasks are abstracted. Thus, there may be some difference between the verification result and the implementation.　In this case study, we take an airship autopilot software as an example of a practical scale embedded system design along with a working implementation. With this example, we evaluate the difference between the throughput performance verification result and measured throughput performance of its implementation to examine usefulness of the throughput performance verification method.

Keyword　embedded systems, multitask software , design model, performance verification, throughput,

1. はじめに

家電製品などに組み込まれるソフトウェアである組込みシステムの開発においては，厳しいリソース(ＣＰＵ，メモリ，ネットワークなど)制約の下での実時間性の実現などの非機能的制約の達成が重要となる[1]．しかし，伝統的なソフトウェア開発手法においてはソフトウェア機能の正しさに焦点が置かれており，設計の途中で性能が大きく変化することを考慮していない．そのような開発手法においては性能問題が開発プロセスの後半で見つかることが多く，設計の見直しなどの手戻りが生じる．このため，設計段階で性能検証を行うことができれば有用である．

我々は従来，複数タスク動作仕様が性能要求を満たすか否かを検証する手法を提案している[2]. [2]の手法は，与えられた複数タスク動作仕様が指定した性能要求を満たさなければデッドロックになるような検証モデルへ変換し，モデルがデッドロックになるか否かを既存のモデル検査ツールで検証するというものである．[2]の手法は確定的な時間制約を記述可能な振る舞い検証モデルである時間ペトリネットの拡張モデルを用いて，システムが最悪時に性能を満たせるか否かを検証する．しかし，この[2]の性能検証手法は，具体的な開発事例における有用性の評価が行われていないという問題点がある．

一方，ESS ロボットチャレンジ[3]というコンテストが開催されており，そこでは飛行船を対象とし，自動で離陸，着陸，コースに沿って飛行などの動作を行わせる飛行船自動航行ソフトウェアの開発を競っている．飛行船自動航行ソフトウェアの制御が精度よく動作するには，単位時間に処理できる制御入力数に関する要求，一般にはスループット要求を満たすことが重要である．この事例（飛行船自動航行ソフトウェア）は，意味のある機能及び規模を持っており，我々の研究グループでは UML[4]で記述された設計文書と実装を保有しているので[2]の性能検証手法の有用性を確認するのに適当な事例と考えられる．

本研究の目的は[2]の性能検証手法に UML シーケンス図で記述された飛行船自動航行ソフトウェア仕様の事例[3]を適用し，[2]の手法による性能検証結果と実測値による性能検証結果を比較しどれぐらいの差異が生じるか調査を行い，[2]の手法の有用性の評価を行うことである．本研究では，まず UML シーケンス図で書かれている飛行船自動航行ソフトウェア仕様を[2]の性能検証手法に適用するために複数タスク動作仕様へ変換する．変換は，UML シーケンス図のメッセージと実行仕様のセットを一つのタスクとみなし，シーケンス図から動作の順序関係を抽出し，対応する複数タスク動作仕様に変換するという方針により行う.そして，事例（飛行船自動航行ソフトウェア仕様）を[2]の性能検証手法に適用し，実機を実測した性能との違いを実験により評価する．実験の方針として，仕様における各タスクの実行時間と実装の各タスクの実行時間を等しく設定し，性能検証手法による性能評価結果と，実機による実測値を比較検証する．

2. 複数タスク動作仕様

複数タスク動作仕様[5] は，データ駆動型のマルチタスクシステムにおいて複数タスクの起動周期，データの受け渡しによる順序関係や並列・選択などの制御構造を記述したもの（タスクグラフ）と，各タスクがそれぞれ実行のために必要とするリソースとの対応関係を記述したもの(リソース割り当て図)の組である．各タスクにはタイムバジェット（設計時に当該タスクの実行に必要な時間を配分したものであり，実装時の最悪実行時間に相当）と相対デッドライン（当該タスクが起動されてから実行を終了するまでの時間に対する制約）が与えられている．

図 1 の例では，並列・並列合流・選択・選択合流を組み合わせた動作が記述されている．図 1 の動作は次のとおりである．まずタスクグラフにおいて，100 単位時間ごとに到着する入力 Input によってτ1 が起動する．τ1 が終了するとτ2 とτ3 へそれぞれ並列に起動要求が出される．τ3 が終了するとτ4 に起動要求が出される．τ2 とτ4 がともに終了すると，τ5 かτ6 のいずれか一方に起動要求が出される．τ5 とτ6 のいずれか一方が終了するとτ7 に起動要求を出される．τ7 が終了すると外部出力(output) を行う．次にリソース割り当てにおいて，タスクτ1，τ2，τ6 は，固定優先度スケジューリング（FP）でリソース resource1 を共有し，タスクτ3，τ4，τ5，τ7 は，横取り不可能な早い者勝ちスケジューリング（FIFO）でリソース resource2 を共有している．

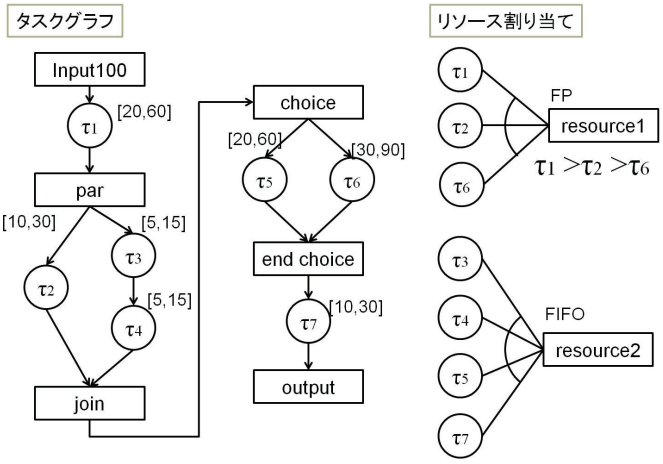


図 1：複数タスク動作仕様の例

3. 優先権付きストップウォッチペトリネットを用いた複数タスク動作仕様の性能検証手法

[2] の性能検証手法は，複数タスク動作仕様から優先権付きストップウォッチペトリネットへ変換し，性能検証モデルに対してスループット要求を満たすか否かの検証を行う．時間ペトリネット[6]とは，プレースと呼ばれるノード，トランジション，及びプレースとトランジション間を結ぶアークと呼ばれる有向枝を持つ有向 2 部グラフであり，各トランジションに 2 つの属性を付加したものである．優先権付きペトリネット[7]とは，時間ペトリネットにトランジションと呼ばれるノード間の優先権を付与したものである．優先権付きストップウォッチペトリネット(PrSwPN)[8]とは，優先権付きペトリネットにストップウォッチ機能を付与したものである．

変換は，与えられたすべてのタスクから，タスクの各動作(任意のタスクのリリース，実行開始，実行終了など) の実行系列の集合と，対応するトランジションの実行時刻も含めた発火系列の集合が等しいような PrSwPN に変換することにより行う.この変換を，まず，タスク，スケジューラなどのシステムの構成要素それぞれに対して，対応する PrSwPN による部分モデルを構成し，それらを結合することにより全体の動作を表現するモデルに変換する方針で行う[2].

なお，プリエンプション（必要なリソースを他のタスクが獲得中の場合に，そのタスクから該当リソースを横取りして実行する動作）を PrSwPN のストップウォッチ機能を用いて表現している.[2] ではスループット要求の検証のため、スループット要求が満たされない状態を検出するアサーションを性能検証モデルに追加している.

PrSwPN の到達可能性検証問題は一般的に決定不能[8]であるが，近似的に到達可能性解析を行うアルゴリズムが提案されており[8]，そのアルゴリズムを実装した TINA[9] という検証ツールがある.

4. 飛行船自動航行ソフトウェアの動作仕様

本研究では，飛行船自動航行ソフトウェアの動作仕様を具体的な事例として取り上げる．飛行船自動航行は，上昇下降用のプロペラ 1 つに旋回や前進・後進用のプロペラ 2 つがついた飛行船本体，飛行船と通信を行う無線通信マイコン，飛行船の現在位置を受信するマイコン，その情報を処理し飛行船に指示を出すグランドコントロール（PC），そして飛行船の現情報を表示するモニタで構成されている．リソースは，2 コアの CPU，カメラとの通信リソース，すべてのグローバル変数を扱っているデータリソース，自動航行を制御する変数リソース，カメラによって得られる画像の通信リソース，飛行船を制御する通信リソースの 6 種類で構成されている．これらを用いて飛行船は，目標の高さまで離陸，数秒間のホバーリング，目標地点まで移動，その場で旋回，離着陸エリアまで帰還，そして着陸の動作を行うことができる．また，動作仕様はシーケンス図で書かれている．制御処理を行うモジュール（メインモジュール（図 2)），カメラからの情報を処理するモジュール（カメラモジュール（図 3)），現情報を画面に出力するモジュール（モニタモジュール（図 4)），飛行船と通信するモジュール（MagicalBox モジュール（図 5)）の 4 つのシーケンス図で書かれている．メインモジュール（図 2）は，グローバル変数より情報を取得・演算し，モニタモジュール，MagicalBox モジュールに演算結果を渡す動作が書かれている．ここで，グローバル変数には，すべてのモジュールで共有するデータ（現在位置，高度，角度などの情報）を格納している．カメラモジュール（図 3）は，カメラより得られた画像より現在位置・角度・高度を算出し，グローバル変数へ算出結果を渡す動作が書かれている．モニタモジュール（図 4）は，メインモジュールから得られた情報を画面上に出力する動作が書かれている．MagicalBox モジュール（図 5）は，メインモジュールから得られた結果をもとに，飛行船のプロペラの制御を行っている動作が書かれている．

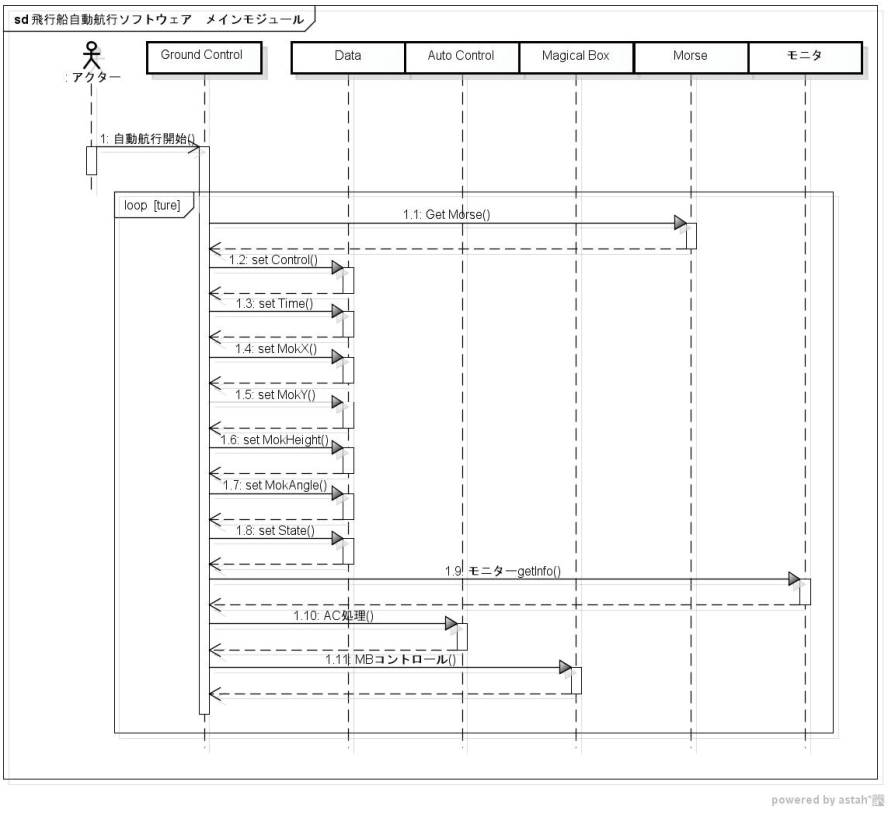


図 2：飛行船のメインモジュール

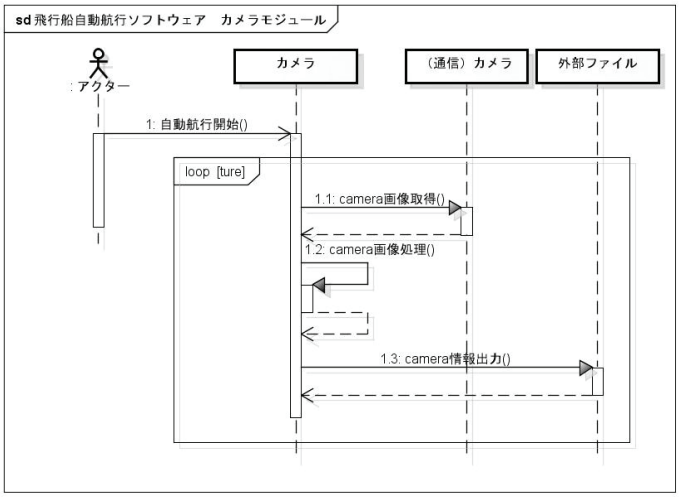


図 3：飛行船のカメラモジュール

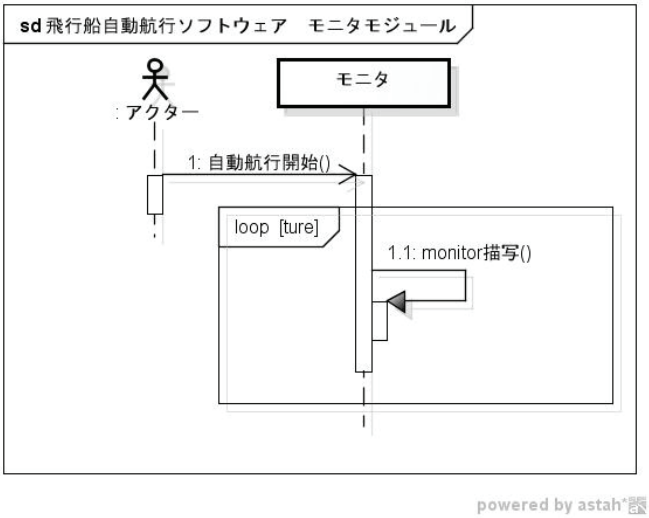


図 4：飛行船のモニタモジュール

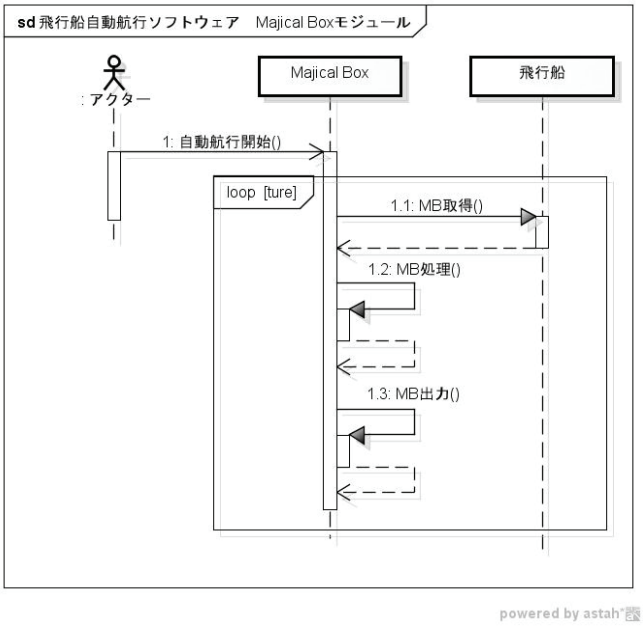
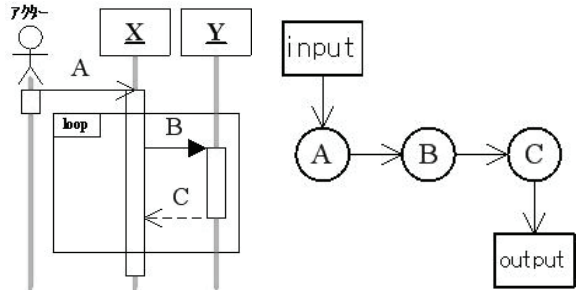


図 5：飛行船の MagicalBox モジュール



(a)　　(b)

図 6：シーケンス図と複数タスク動作仕様の同期の例

5. シーケンス図からタスクグラフへの変換

本研究では，次の方針で UML シーケンス図からタスクグラフへの変換を行った．まず UML シーケンス図のメッセージ（矢印）と実行仕様のセットを、複数タスク動作仕様では一つのタスクとして扱う．また，変換した時，シーケンス図の実行仕様が終了してもほかのオブジェクトへメッセージを渡すものがない場合，複数タスク動作仕様の出力を意味する Output に行くものとする．この下で，シーケンス図に記述されたタスク間の順序関係を抽出し，対応するタスクグラフに変換する．具体的な変換方法の一例として同期メッセージを含む変換方法を示す．

図 6(a)のシーケンス図の動作は次のとおりである．まずアクター（外部からのイベントの生成元）からの入力メッセージ A がオブジェクト X へ送信され，処理の一部が実行され，次にメッセージ B がオブジェクト Y へ送信され，処理が実行される．メッセージ B は同期通信であり，応答メッセージ C を受信するまでオブジェクト X の処理は中断する．最後にメッセージ C が送信され，オブジェクト Y の処理が終了すると，オブジェクト X の処理が再開される．この動作を図 6(b)のようにタスクグラフに変換する．

ここで，図 6(b)のタスク A は図 6(a)の A の送信から B の送信の直前までの処理，タスク B は B の送信から C の送信直前までの処理，タスク C は C の送信から終了までの処理に対応する．

6. 実験

4 章の飛行船自動航行ソフトウェアのシーケンス図とその実装を対象に，[2]の検証手法によって得られる結果と実機の性能との比較評価を行う．

6.1. 実験方法

4 章の飛行船自動航行ソフトウェアのシーケンス図とその実装を対象に，[2]の検証手法によって得られる結果と実機の性能との比較評価を行う．

実験は次の手順で行う．まず，飛行船自動航行ソフトウェアを搭載した実機を競技の時を再現した上で，実機のプログラムを実行し，各タスクの最悪実行時間および全体のスループット性能を測定する．次に，飛行船自動航行ソフトウェア仕様から複数タスク動作仕様への変換を行う．シーケンス図からタスクグラフへの変換は 5 章に示した方針で行う．なお，各タスクのタイムバジェットは，先の測定で得られた各タスクの最悪実行時間の実測値を一定の割合だけ長くしたものをタイムバジェットとする．本来，タイムバジェットは設計時に見積もって設定するものであるが，本実験においては，タイムバジェットをほぼ実装と同じ値に正確に見積もれた前提でスループット性能の見積もり値を評価するためにこのようにしている．また，実測値より長くした理由は，測定においてはすべての動作を網羅できたわけではなく，測定値が最悪実行時間とは限らないためである．最後に以下に述べる手続きにより複数タスク動作仕様からスループット性能を見積もる．まず，スループット性能の見積もり値の初期値を設定し，その値をスループット要求として性能検証手法を適用する．その結果，もし「満たさない」と判定されたならば,「満たす」と判定されるまでスループット要求の値を一定量ずつ増やしながら性能検証を繰り返す．逆に，性能検証結果が「満たす」と判定されたならば,「満たさない」と判定されるまでスループット要求の値を一定量ずつ減らしながら性能検証を繰り返す．このようにして最後に「満たす」と判定されたスループット要求の値を見積もり値として得る．

各タスクの最悪実行時間を測定するために，1 つのタスクに対し一定回（本実験では 100 回）そのタスクが実行に要した時間を測定した．スループット性能を測定するために，4 つのモジュール（メインモジュール，カメラモジュール，モニタモジュール，MagicalBox モジュール）それぞれのメインループ（入力イベントに対して処理される一連の処理全体）が 1 回実行される時間を一定回（本実験では計 100 回）測定し，最悪値を測定値とした．

6.2. 実験結果

飛行船自動航行ソフトウェアのシーケンス図とリソース割り当て情報(CPU や通信リソースなどのタスクへの割り当て情報)から複数タスク動作仕様へ変換すると図 7 の通りになった．図 7 の左から 1 列目はメインモジュール，左から 2 列目の上段はカメラモジュール，左から 2 列目の中段がモニタモジュール，左から 2 列目の下段が MagicalBox モジュールのシーケンス図からタスクグラフへ変換したものがそれぞれ書かれている．図 7 の右側は，それぞれのタスクが使用するリソース（CPU など）を示すリソース割り当て図が書かれている．また，6.1 章の実験結果を表 1 に示す．

表 1 : 実機のスループット性能測定値と性能検証手法によるスループット評価値

	実機（測定値）[ミリ秒]	性能検証手法（評価値）[ミリ秒]	相対誤差 [%]
メイン	2051	2100	2.4
カメラ	5840	6000	2.7
モニタ	1491	1500	0.6
MagicalBox	2938	3000	2.1

※評価値：性能検証により「満たす」と判定されたスループット要求値

6.3. 考察

実機と性能検証手法のスループットの相対誤差が 3%を切るほどの差であることが分かった．このことより実機の性能測定値と性能検証手法による性能の評価値の差は比較的小さいと判断できる．この結果より,[2]の性能検証モデルは,本研究で用いた事例においては,性能の見積もりに利用可能であると考えられる．

7. まとめ

本研究では[2]の性能検証手法を具体的な開発事例に適用し，両者の性能見積もりの差を比較した．その結果，実機の性能測定値と性能検証手法による性能の評価値の差が比較的小さいという結果が得られ，[2]の手法は性能見積もりに利用可能であることを示した．しかし一方で，性能検証手法のスループットは，さらに性能要求を厳しくしても動く可能性があり，最適なスループットを決定する方法を考慮する余地がある．今後の課題として，性能見積もりの精度の比較評価，および，性能見積もりの結果を考慮した開発方法の確立などがあげられる．

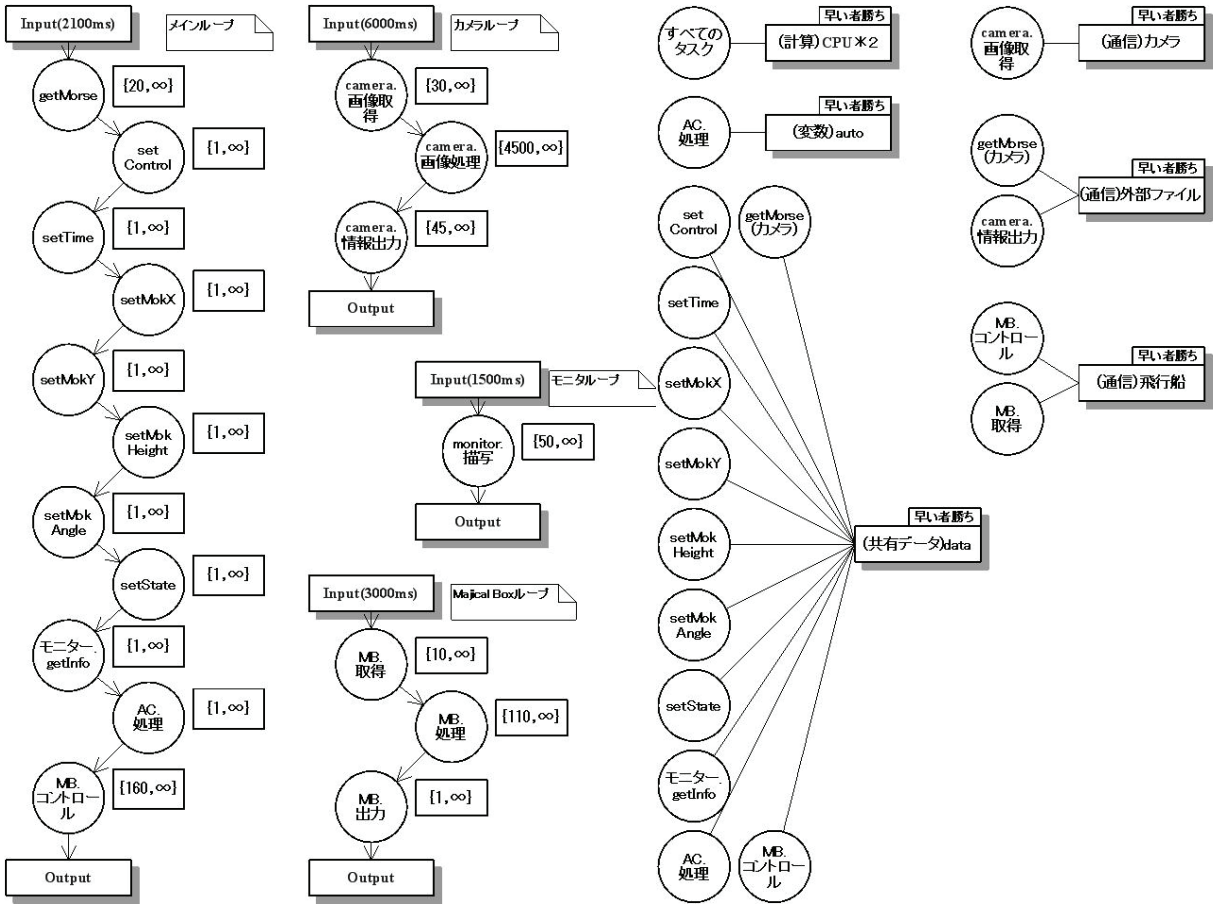


図 7：飛行船自動航行ソフトウェアの複数タスク動作仕様

文　献

[1] Henzinger, T.A. and Sifakis, J.: Embedded Systems Design Challenge, in Proc. of 14th Int. Symp. on Formal Methods (FM 2006), Lecture Notes in Computer Science, Vol.4085, Springer Verlag, pp.1-15, 2006.

[2] 百々太市, 中田明夫. “プリエンティブスケジューリングによるリソースを共有する複数タスク動作仕様の性能検証,” 組込みシステムシンポジウム（ESS2010）論文集, pp.107-112, 情報処理学会, 2010.

[3] ESS ロボットチャレンジ実行委員会. “ESS ロボットチャレンジ競技概要,” 2011. http://sigemb.jp/rc2011/sub/ESSroboto2011.pdf.

[4] 株式会社オージス総研　オブジェクトの広場編集部. “その場で使えるしっかり学べる UML2.0,” 株式会社　秀和システム, 2006.

[5] 倉田和哉, 百々太市, 中田明夫, “リソース制約を持つ複数タスク動作仕様におけるタイムバジェット最適化の一手法,” 信学技報 SS2011-34, 電子情報通信学会, 2011.

[6] Berthomieu, B. and Diaz, M.: Modeling and Verification of Time Dependent Systems Using Time Petri Nets, IEEE Trans. Softw. Eng., Vol. 17, No. 3, pp. 259-273, 1991.

[7] Berthomieu, B., Peres, F. and Vernadat, F.: Model Checking Bounded Prioritized Time Petri Nets, in Proc. of 5th Int. Symp. on Automated Technology for Verification and Analysis (ATVA 2007), Vol. 4762 of Lecture Notes in Computer Science, pp. 523-532, Springer-Verlag, 2007.

[8] Berthomieu, B., Lime, D., Roux, O. H. and Vernadat, F.: Reachability Problems and Abstract State Spaces for Time Petri Nets with Stopwatches, Journal of Discrete Event Dynamic Systems, Vol. 17, pp. 133-158, 2007.

[9] Berthomieu, B. and Vernadat, F.: Time Petri Nets Analysis with TINA, Proc. of 3rd Int. Conf. on the Quantitative Evaluation of Systems (QEST 2006), IEEE Computer Society Press 2006.