ソフトウェア工学 130−13
（２００１．３．２３）

基本ブロックを用いたオブジェクト指向リアルタイムシステムの開発

渡辺 晴美 立命館大学 理工学部
E-mail: harumi@selab.cs.ritsumei.ac.jp

リアルタイムシステムをオブジェクト指向開発する際、アーキテクチャ設計段階で、分析モデルで抽出したオブジェクトモデルからスレッドモデルを構築する。構築したスレッドモデルは、実装と直接結びつくため、システムの性能を決定することになる。従ってスレッドモデルの構築は極めて重要なプロセスである。本稿では、オブジェクモデルから性能の良いスレッドモデルを効率的に設計する方法を提案する。本方法では、ステートチャートとクラス図で表したオブジェクトモデルを基本ブロック (合流・分岐のない部分ステートチャート) に分割し、性能解析、ブロック間のデータ依存関係、通信関係を解析する。その結果を基に基本ブロックを組合わせることで、スレッドモデルを構築する。

A Developing Method of Object-Oriented Real-Time Systems Using Basic Blocks

Harumi Watanabe
Department of Computer Science, Ritsumeikan University

In the architecuture designing process for Object-Oriented Real-Time Systems, a objects model maps to a threads model. The process is significant since the threads model is directly related to an implimantation, and impacts on its performance. In this paper, we present an efficient designing method for high performance threads model an objects model. In this paper, we present a method to design a high performance threads model from an objects model. In this paper, we present an efficient method to achieve high performance in designing a threads model from an objects model. In the method, an object model, which consists of statecharts and class diagrams, is divided into basic blocks which is partial of a statechart without branchs and joints. By acquireing the result of performance, dependency and message passing analysis a thread model is constructed on basic blocks.

1 はじめに

昨今、半導体技術の進歩に伴いリアルタイムシステムの大規模化、複雑化が進みつつあり、リアルタイムシステムに対してもオブジェクト指向開発法の適用が望まれているが、その恩恵を十分に受けるに至ってない。リアルタイムシステムでは、その時間的性質により、簡単に機能変更ができず、オブジェクト指向開発を行っても、再利用性が向上できない。オブジェクト指向開発の要求分析で得たオブジェクトは性能を考慮していないため、実装と結びつかないといった問題がある。これらの問題を解決するために、最近、アーキテクチャ設計段階にオブジェクトをスレッドに割当てる方法 [5][1] が取られているが、具体的な方法は確立されていない。

本稿では、オブジェクモデルからスレッドモデル構築する方法を提案する。オブジェクトモデルは UML のステートチャートとクラス図で表され、要求分析から得られる。スレッドモデルも、ステートチャートとクラス図で表されるが、実装と直接対応付くモデルである。従って、構築したスレッドモデルは実装と直接結びつくため、システムの性能を決定することになる。本稿で提案する方法は、良いスレッドモデルを構築するために、オブジェクトモデルを基本ブロック (合流・分岐のない部分ステートチャート) に分割し、性能解析、ブロック間のデータ依存関係、通信関係を解析する。その結果を基に基本ブロックを組合わせることで、スレッドモデルを構築する。本稿では、オブジェクトモデルからの基本ブロックの生成、シーケンス図とデータ依存グラフを用いたスレッドの識別、基本ブロックからのスレッドの構築について述べる。

以下、2で、提案する基本ブロックを用いたリアルタイムシステム開発法の概要についてまとめ、3でその詳細について説明する。4ではエレベータシステムを例に提案した方法に適用し、その結果を 5で関連研究とともにに考察する。最後に 6で、本稿のまとめと、今後の展望について述べる。

2 基本ブロックを利用したシステムの解析とスレッドの構築

本節では、本論文で提案する分析段階で抽出したオブジェクトモデルから、実装段階と直接関連付くモデルであるスレッドモデルへ変換する方法の概要について述べる。

図 1に基本ブロックをオブジェクトモデルのクラス図とステートチャートから抽出し、スレッドモデルを構築する様子を示す。オブジェクトモデルは UML のクラス図とステートチャートらなる。本方法では、設計者はまず UML のクラス図とステートチャートを記述する。次にステートチャートとクラス図を基本ブロックに分割する (図 1の上部)。基本ブロックに関しては次節で述べる。

カラーペトリネット (CPN)[4] を用いてオブジェクトモデルのデッドロックなどの誤りの検出と性能解析を基本ブロックを考慮して行う。解析結果からシーケンス図を生成する (図 1の中央左)。一方、ステートチャートからデータ依存関係を求め、データ依存グラフ (図 1の中央右) を生成する。シーケンス図、データ依存グラフについては次節で述べる。

開発者は、シーケンス図から時間がかかる基本ブロックや、頻繁にアクセスのある基本ブロックを見つける。そして、基本ブロック間のデータ依存関係をデータ依存グラフを参照し、開発者は性能が向上するように基本ブロックを組替えることで、スレッドモデルを構築する (図 1の下部)。スレッドモデルは、再び CPN へ変換し、誤り検出と性能解析を行い、変更時に混入される誤りの検出と、性能の向上を確認する。

CPN を用いた性能解析と誤り検出は、クラス図と状態図を CPN へ変換することで行う。CPN への変換は、我々が [3] で提案してきた方法を用いる。[3] を用いれば、並行性、オブジェクトの生成、ダイナミックバインディングなど実行時に挙動が決定する性質をモデルが持つ場合においても解析可能である。CPN を用いれば性能解析も可能である。

図 1の上部は、クラス ObjA のインスタンスとクラス ObjB のインスタンスは並行に動作

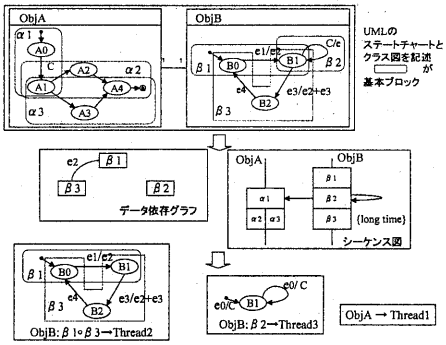


図 1: 基本ブロックによるスレッドの構築概要

することを表している。図1の中部右は、シーケンス図で表したところ、基本ブロック β_2 の処理時間が長いことが判明した様子を表している。データ依存グラフから、基本ブロック β_2 がObjB の他の基本ブロックとデータ依存関係がないことが分かる。そこで、基本ブロック β_2 の処理を独立させ、スレッドを3つ作成する。一つ目のスレッドは、ObjA をそのまま利用し、二つ目のスレッドは、基本ブロック β_1 と基本ブロック β_3 を接続することで得る。三つ目のスレッドは、処理時間が長い基本ブロック β_2 となることを図1の下部は示している。

本稿では、オブジェクトモデルを基本ブロックへ分割し、スレッドモデルを構築する方法にのみ着目し、次節以降で述べる。

3 オブジェクトモデルからスレッドモデルの構築法

本節では、オブジェクトモデルからスレッドモデルを構築する方法について述べる。まず、提案する方法の基盤となる基本ブロックの定義と生成規則を与える。次に、シーケンス図とデータ依存グラフ、そしてそれらを用いたスレッドを識別する方法について述べる。最後に、基本ブロックからスレッドを構築する規則を示す。

3.1 基本ブロック

基本ブロックは、本来コンパイラの最適化処理に利用される単位で、プログラム言語での基本ブロックとは、分岐・合流のない文の列のことを言う [2]。本稿では、基本ブロックは分岐・合流のない部分ステートチャートとする。図2に、基本ブロックの生成規則をステートチャートの例で示す。基本ブロックは、スレッドモデルが時間的性質を満たすことを目的としているため、アトミック処理と時間を費やすイベントを考慮している。アトミック処理は不分割な処理であるため、複雑な図であっても一つの基本ブロックとして扱う。一方、時間を費やすイベントは、分岐合流に関係なく、その事象を一つの基本ブロックとする。

3.2 シーケンス図

ステートチャートとクラス図を実行し、シーケンス図を生成する。本方法では、実行には CPN を用いる。UML のステートチャートにはフォーク、ジョイン、階層の概念を持つ。CPN がこれらの概念を持つモデルを解析に適していることが判明している [3]。UML のシーケンス図と異なり、本方法で用いるシーケンス図には基本ブロックを明示する。シーケンス図の例を図7に示す。

3.3 データ依存グラフ

基本ブロック間のデータ依存を調べ、データ依存グラフを生成する。データ依存グラフは、一つのステートチャートに対し、一つのグラフを生成する。二つの基本ブロック間にデータ依存があるとは、二つの基本ブロック内に同じ変数を持つ場合をいう。変数は、ステートチャート内のイベントとガードを表す式に用いられる。データ依存グラフは、スレッドを構築する際、一つのスレッドを形成する予定の基本ブロック間でも必用に応じて作成する。データ依存グラフの例を図8に示す。

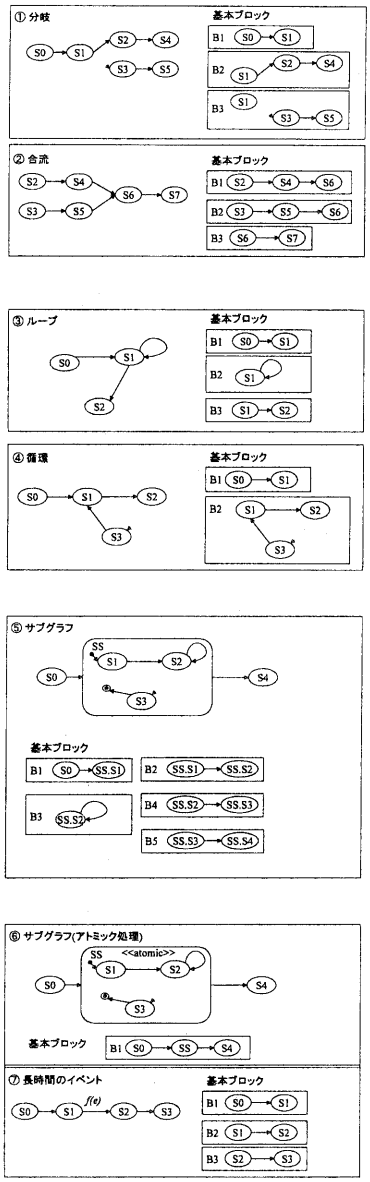


図 2: 基本ブロックの生成規則

3.4 スレッドの識別

データ依存グラフとシーケンス図をもとにスレッドを識別する。本節では、リアクティブ性に着目した場合のスレッドの識別例について述べる。処理時間を減少させるために、処理時間が長いスレッドの負荷分散、通信回数、コンテキストスイッチの回数の削減について考慮する。処理時間が長いスレッドに関しては、処理時間が長い部分を抽出し、以下の処理を行う。

1. シーケンス図から処理時間の長い基本ブロックを抽出する。
2. その基本ブロックが同じオブジェクトの他の基本ブロックとデータ依存があるかを調べる。
3. データ依存がなければ、この基本ブロックを別なスレッドにする。データ依存関係がある場合、基本ブロックを変更し、変更した基本ブロックを接続し、スレッドにする。

通信回数やコンテキストスイッチの回数が多い場合、スレッド数を削減する。スレッド数を削減するために以下の処理を行う。

1. シーケンス図から通信が頻繁にある基本ブロックを抽出する。
2. それらの基本ブロックを一つのスレッドにまとめる。

3.5 スレッドの構築

本節では、基本ブロックからスレッドを構築する規則を例を用いて示す。以下の処理を行いスレッドを構築する。

1. 必用に応じて基本ブロックを変更する。変更には以下がある。

   基本ブロック B の内部を変更し B' にする場合。: $B' = modf(B)$
   基本ブロック B1 と基本ブロック B2 を融合する場合。: $B' = modf(B1 + B2)$
   基本ブロック B を二つに分割する場合。: $B'1 = modf_1(B), B'2 = modf_2(B)$

2. 基本ブロックを接続する。接続規則を図 3 に示し、接続例を図 4に示す。

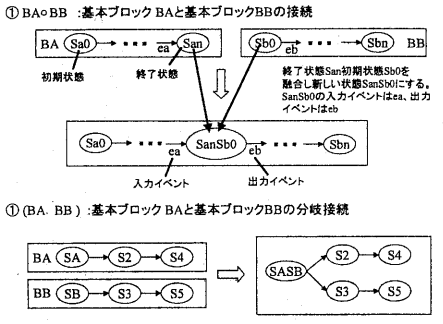


図 3: 基本ブロックの接続規則

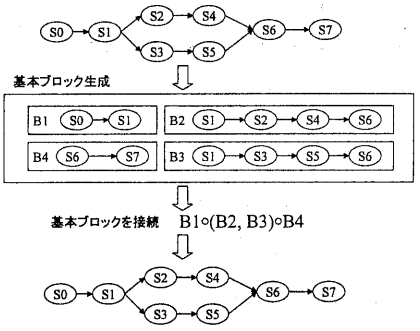
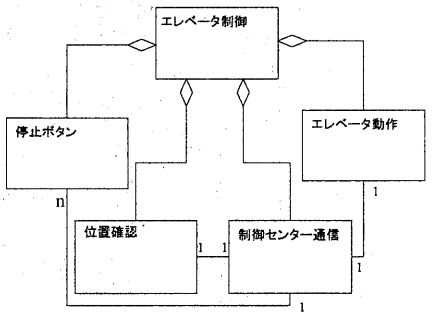


図 4: 基本ブロックの接続例



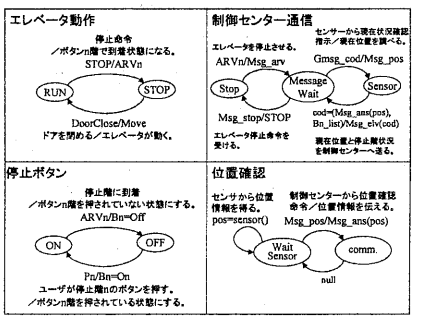


図 5: エレベータシステムのオブジェクトモデル

4 適用例

本節では、提案した方法をエレベータシステムに適用した例を示す。エレベータシステムは、エレベータ制御部、制御センター部、フロア部からなる。エレベータ制御部は各エレベータに備えられ、制御センター部は、各フロアに備えられたフロアの状況を管理するフロア部と、各エレベータの状況とを併せて全てのエレベータの運行を管理する。図 5は、エレベータ制御部の一部を表している。例は、提案した方法の特徴のみを示せるように、処理や状態などの多くを省略している。

例では、「制御センタ通信」で各エレベータは制御センターとのやりとりを行う。エレベータの位置情報、停止階情報を送信し、停止命令を受ける。「エレベータ動作」は、「制御センター通信」が停止命令を受けると、停止作業を行う。「位置確認」は、エレベータの位置情報を「制御センター通信」へ通知する。「停止ボタン」は停止階情報を「制御センター通信」へ通知する。

図 6に図 5から生成した基本ブロックを表す。シーケンス図、データ依存グラフ、スレッドモデルを各々図 7、図 8、図 10にを示す。

図 7のシーケンス図で、「制御センター通信」と「位置確認」間、特に基本ブロック BBC1 と BBP2 で、メッセージのやりとりが多いことが分かる。データ依存グラフで BBC1 と BBC2 にデータ依存関係がないことが分かる。BBC1 を「エレベータ動作」と、BBC2 を「位置確認」と併せることで余計な通信を減らすことができる。基本ブロックを変更し接続することで、スレッドを得る様子を図 9に示す。

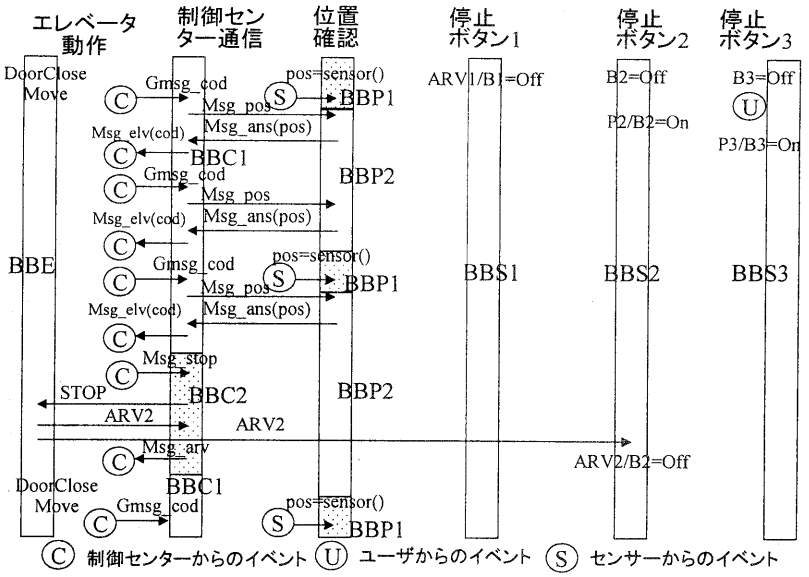


図 7: シーケンス図

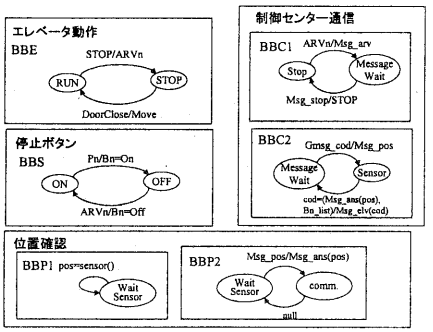


図 6: エレベータシステムの基本ブロック

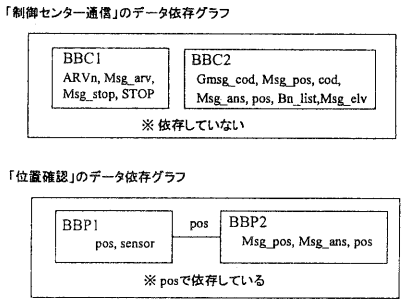


図 8: データ依存グラフ

5 議論

本節では、前節での適用例をもとに評価考察を行い、関連研究との比較を行う。適用例では、エレベータシステムに提案した方法を適用し、基本ブロックを生成し、シーケンス図からブロック単位で通信回数が多い個所を発見し、そのブロックがデータ依存グラフからデータ依存がないことを確認した。そして、4 つのオブジェクトから一つを削除し、3 つのスレッドへすることで、余分な通信を減らすことができた。基本ブロックの生成、スレッドの構築では、提案した

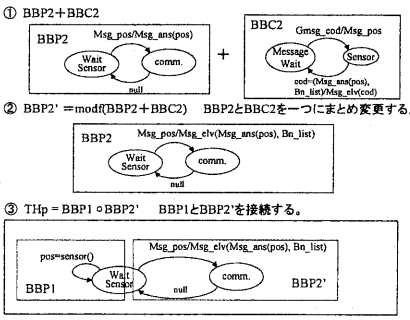


図 9: 基本ブロックの変更とスレッドの構築

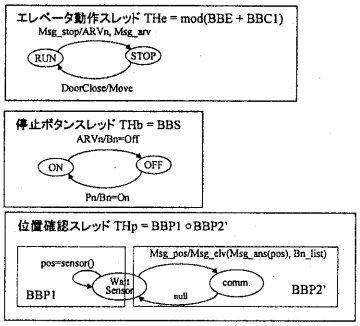


図 10: エレベータのスレッドモデル

規則により作業を効率的に行うことができた。

まず、適用例から基本ブロックの生成について考察する。UML のステートチャートは様々な疑似状態を持つが、これらの状態はフォークとジョインを除けば、一般的な FSM へ変換することができる。適用例では、本稿で示さなかった部分も含め、提案した規則で十分対応できた。今後、これらの規則と疑似状態との対応付けを明らかにすれば UML のステートチャートに十分対応できる。

シーケンス図は、適用例では手動で生成した。解析しやすいシーケンス図を自動的的に得るためには、ステートチャートとクラス図から CPN へ変換する方法と CPN の実行結果を工夫する必要がある。

スレッドの構築に関しては、基本的な規則のみ用意した。本プロセスが最も難しく、性能と密接に関連付く。効率よく構築するために、またリアルタイム性を満たすためには、シーケンス図、データ依存図から、性能を阻害する要因のパターン化が必要となる。また、適用例で示したように、基本ブロック内の変更には規則はない。データ依存が複雑な基本ブロック間に関しての規則も必要である。しかし、提案した規則は、基本ブロック内を変更してから、接続を行うため、今後は、ブロック内の変更規則とデータ依存が複雑な場合の規則のみを追加すれば良い。

提案した方法と関連研究との比較を行う。[5] では、スレッドを識別するために、イベントを分類する手法を用いているが、具体的にオブジェクトモデルとスレッド対応付ける方法については触れていない。[1] では、SES モデルと呼ばれるスレッドモデルを提案し、オブジェクト間の協調動作によって発生する処理列に着目することで、オブジェクトモデルとの対応付けを行っている。B. Selic らは、ROOM[5] と UML を統合させ、アーキテクチャ記述に適した ROOM を UML で利用できるようにしている [8]。

オブジェクトモデルとスレッドモデル間を基本ブロックを用いて関連付ける方法は提案されていない。基本ブロックの基本的な考えは、コンパイラの最適化にある。従って、基本ブロックを単位にモデルを解析する手法は、性能の良いスレッドを構築する方法に適している。

6　おわりに

本稿では、基本ブロックを利用したオブジェクトモデルからスレッドモデルを構築する方法を提案した。オブジェクトモデルとスレッドモデルの関連付けの難しさにより、リアルタイムシステムの開発は、オブジェクト指向技術の恩恵を十分に受けることができなかった。本方法によりこの問題を一歩解決へ近づけることができた。また、本方法により、性能の良いスレッドモデルを従来より効率よく構築できることを適用例で示した。

さらに、効率よく構築するために、またリアルタイム性を満たすために、スレッドの構築規則を増やし、シーケンス図、データ依存図そして基本ブロック間から性能の良いスレッドを構

築できるパターンを抽出する必要がある。

参考文献

[1] 青木 利晃, 内藤 壮司, 片山卓也, “オブジェクト指向組込みシステム開発のためのSES-Basedアプローチ”, 日本ソフトウエア科学学会誌コンピュータソフトウエア Vol.16 No.2, pp.67-71, 1999

[2] 中田育男, “コンパイラの構成と最適化”, 朝倉書店, 1999

[3] 渡辺 晴美, 徳岡 宏樹, Wu Wenxin, 佐伯 元司, “カラーペトリネットによるオブジェクト指向ソフトウエアのテストと解析方法”, 電子情報通信学会和文誌 D-I, Vol.J82-D-I, No.3, 1999, 3

[4] K. Jensen, “COLOURED PETRI NETS – Basic Concepts, Analysis Methods and Practical Use – Volume 1-3”, Springer-Verlag, 1992, 1994, 1997

[5] B.P.Douglass, “Real-Time UML 2nd Edition Developing Efficient Objects for Embedded Systems”, Addison Wesley, 1999

[6] G.Booch, J.Rumbaugh, I.Jacobson, “The Unified Modeling Language User Guide”,Addison Wesley, 1999

[7] B. Selic, G. Gullekson, P.Ward, “Real-Time Object-Oriented Modeling”, John Wiley, New York, 1994

[8] B. Selic, J. Rumbaugh, Using UML for Modeling Complex Real-Time Systems. ObjectTime Limited, 340 March Rd., Kanata, Ontario, Canada 1998. http://www.objectime.com/otl/technical/umlrt.html