オブジェクト指向ペトリネットによる ワークフローのモデル化と分析

Modeling and Analysis of Business Process by Object-Oriented Petri-Nets

孫騰涛† 塚原浩太‡ 飯島正‡
Tengtao Sun† Kota Tsukahara, and Tadashi Iijima‡
†慶応義塾大学大学院　理工学研究科
‡慶応義塾大学　理工学部
† Graduate School of Science and Technology, Keio Univ.
‡ Department of Science and Technology, Keio Univ.

要旨

近年の情報技術の進歩に合わせ，ワークフロー管理システムが注目を集めている．ワークフローとは文字通り仕事の流れであり，複数の人間や機械（リソース）によって実行されるプロセスである．また，モデル検査とは形式的検証手法の１つである．システムの有限状態モデルがある性質を満たすかどうかの検証を自動的に，状態の網羅的な探索により行ことができる. 本研究では, オブジェクト指向ペトリネットを用いてワークフローをモデル化し，それをモデル検証技術で解析する手法を提案する.

1. はじめに

ワークフローとは業務の流れであり，複数の人間や機械（リソース）によって実行されるプロセスである．一般的に，業務の流れには人間や機械，書類，生産物など様々なリソースが複雑に絡み合っている．そのためワークフローは，それらリソースや作業間の関係，必要な制約条件などがわかりやすく整理され表わされなければならない.

これまで，ワークフローをモデル化するための研究は多々行われてきた．その中の一つとしてペトリネットによるモデル化がある．しかし，これらの多くは，ワークフローを単独のペトリネットでモデル化したものであり，その表現力や活用方法に限界がある．そこで本研究ではまず，ワークフローのモデル化手法として，オブジェクト指向ペトリネットを用いる.

さらに本研究では，モデル化したワークフローの分析ための検証の手法を提案する．システムの検証手法としてこれまで用いられてきた主な方法はシミュレータを用いたテストによるものであった．しかし，テストでは全数検査を行えない為に，検出できない重要な不具合が残りがちであるといった問題がある．これに対し，モデル検査は，時相論理で表現された性質をシステムが満たすかどうかを検査する手法であり網羅的ににシステムを検証できる.

本論文では，オブジェクト指向ペトリネットの概念を用いてワークフローを表現し，更に，そのペトリネットモデルをモデル記述言語 Promela で表現されたモデルに変換して，ワークフローの性質の検証を行う方法を提案する．まず第 2 章では，ワークフローとオブジェクト指向ペトリネットの概念を述べる．第 3 章では，モデル検査の基礎を説明する．第 4 章では，ペトリネットモデルを Promela モデルの変換ルールを提案する.

2. ワークフローとオブジェクト指向ペトリネット

2.1. ワークフロー

ワークフローとは業務に関わる人間や機械などの流れを表したものである．一般的に，業務の流れには人間や機械，書類，生産物など様々なリソースが複雑に絡み合っている．そのためワークフローは，それらリソースや作業間の関係，必要な制約条件などがわかりやすく整理され表わされなければならない．また，書き上げたワークフローシステムに問題がないかの検証も必要である.

2.2. オブジェクト指向ペトリネット

本研究では，ワークフローの記述のためにオブジェクト指向ペトリネットを用いる．オブジェクト指

向ペトリネットとは，その名の通り，オブジェクト指向の考え方を取り入れたペトリネットの拡張モデルである．オブジェクト指向ペトリネットにおいて，トークンはそのトークンごとに対応するリファレンスを持つことができる．つまりトークンは，そのトークンごとに対応するペトリネット（オブジェクトネット）を持つことができる．

オブジェクト指向ペトリネットをワークフロー記述に利用することのメリットとして挙げられるのは，各人単位，機能単位でワークフローの記述ができることである．

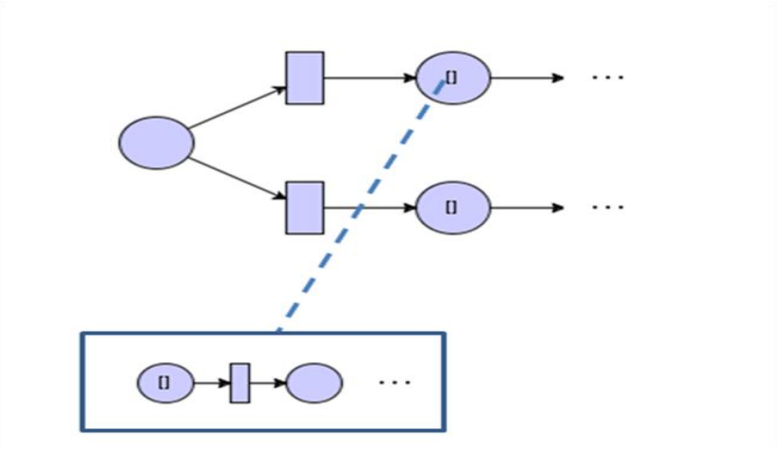


図 1　オブジェクト指向ペトリネット

本研究で用いるオブジェクトネットの概念は，EOS モデル[1]の概念に基づき以下の定義となる．

定義 1 オブジェクトネットの定義

オブジェクトネットは，$OS = (SN, ON_{m^0}, \rho, R_0)$ の集合である．SN は，システムネットと呼ぶ．ON_{m^0} は，P/T ネットの有限集合で，オブジェクトネットと呼ぶ．ρ は、オブジェクトのトランジションとシステムのトランジションの間の動作関係を表す．R_0 は，初期化マーキングである．

3. モデル検査手法

本研究では，モデル検査ツールとして SPIN[2]を利用する．以下，SPIN におけるモデルと性質の記述について説明する．モデル検査[1] とは，時相論理 LTL で表現された性質をシステムが満たすかどうかを検証する手法である．モデル検査ツール SPIN では主に，システムの実行経過が満たすべき順序に関する性質などを検査する．SPIN を用いた一般的な検証作業の流れを図 2 に示す．まず，設計仕様の従って，Promela 言語でモデルを作成する．次にシステムに要求する性質を論理式 LTL で記述する．最後に検査機によって，モデルが論理式を満たすか否かを検証する．

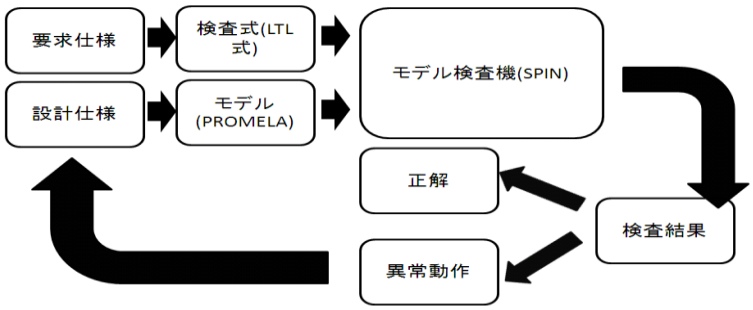


図 2　モデル検査における検証作業の流れ

3.1. 時相論理 LTL

SPIN では LTL(Linear Temporal Logic)という時相論理を使う．LTL では，命題論理の結合子，すなわち否定“¬”，論理和“V”，論理積“∧”，含意すなわち，ならば“→”に加えて，「ずっと成り立つ」を表す “G”や「いずれ必ず成り立つ」を表す“F”といった様相演算子を使うことができる．例えば「今も将来もずっと p が成り立つ」を“Gp”と，また「将来いずれ必ず p が成り立つ」を“Fp”とそれぞれ書くことができる．

図 2 示すように，まず，利用者は，検査対象のシステム(モデル)を Promera 言語を用いて状態遷移システムとして記述し，一方で LTL を用いて検査項目を記述する．これら 2 つをモデル検査器に入力することで，システムが検査項目を満たすか否かをシステムの状態空間を全探索することにより検査する．システムが検査項目を満たさない場合は，そのような実行の一例を反例として出力する．

3.2. 仕様記述言語 Promela 言語

SPIN では入力言語として Promela(Process Meta Language) を用いるため，検査対象であるシステムを Promela を用いて記述する．プロセスは proctype 宣言文で定義する．複数のプロセスを記述することが可能である．以下のように簡単に定義できる．

```
Proctype hello () {
   プロセスの実行内容
}
```

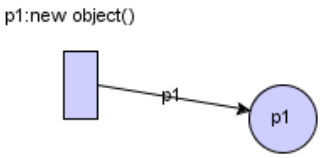
4. オブジェクト指向ペトリネットから Promera 言語への変換

本研究では，オブジェクト指向ペトリネットによりワークフローをモデル化し，モデル検査を用いて検証する手法を提案する. 本節では, オブジェクト指向ペトリネットからモデル検査機のモデル(Promela)モデルの変換ルールを述べる. その概略は以下のとおりである. ペトリネットのトランジション(transiton)は proctype のプロセス文でマッピングし，プレース(plcae)はチャンネルの配列で記述する．トークンはチャンネルの中のメッセージとして表す．マーキング(marking)と初期化は，チャンネルに対するメッセージの入力と出力にマッピングする．

4.1. プレースとトークン

プレースはトークンの情報を扱い，システムの状態を表す．そこで，プレースは PROMELA 言語の chan データ型に変換する．また，トークンは mtype 型としてチャンネルのメッセージに変換する．以下の表に具体的に示す．トークンには通常のペトリネットでのトークンに相当するブラックトークンと，オブジェクト指向ペトリネットに特有のリファレンスを表すオブジェクトトークンに分けられる．

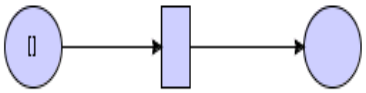
表 1　プレースとトークン

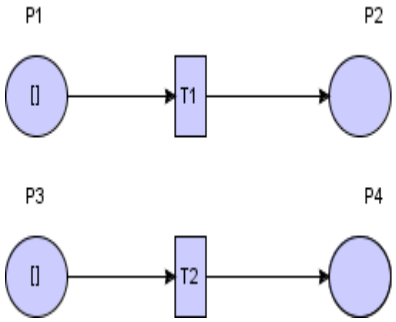
プレースとブッラクトークン		mtype={bk}; chan p =[capacity] of { mytpe } • capacity は，プレイスの容量. • {mtype}はチャンネル p の中で扱えるトークンの型. • bk は，以上のように定義によりチャンネル p の中で保存できる.
ブレースとオブジェクトトークン		mtype={p1}; chan p =[capacity] of { mytpe }; • p1 は，以上のように定義によりチャンネル p の中で保存できる.

4.2. トランジション

トランジションはシステムの状態を変換させる．オブジェクト指向ペトリネット[1]では，同期トランジションと非同期トランジションに分けられる．同期トランジションが発火するためには，それと同期している全てのトランジションの発火条件を満たすことが必要になる．一方，非同期のトランジションは単独発火することができる．そこで 同期化されたトランジション(T1 と T2)に対して，同時発火するという意味で以下のようにトランジション（T1_T2）を定義する．以下の表に具体的に示す.

表 2　トランジションの変換ルール

非同期トランジション T		proctype　Transition T1() {
		BeforFire:
		do
		::(T1 の発火条件の判断)->;goto T0Fire
		od;
		AfterFire:
		do
		::true->発火後，T1 の出力のプレース上のトークンの計算;goto T0BeforFire
		od
		}

同期トランジション T1 と T2		proctype　Transition T1_T2() {
		BeforFire:
		do
		::(T1 と T2 の発火条件の判断)->goto T0Fire
		od;
		AfterFire:
		do
		::true->発火後，T1 と T2 の出力のプレース上のトークンの計算;goto T0BeforFire
		od
		}

5. ツールの設計と実装

5.1. 概要

ツールのシステム構成の概略は図 3 のようになる. ここで GUI はペトリネットエディタを兼ねていて，オブジェクト指向ペトリネットの作成や保存およびシミュレーション実行ができるだけでなく，モデル検査部(MC)の入力形式(Promela)への変換を起動する. PNMLX(Petri Net Markup Language x)は PNML(Petri Net Markup Language)をオブジェクト指向ペトリネット用に，独自に拡張した保存形式である.

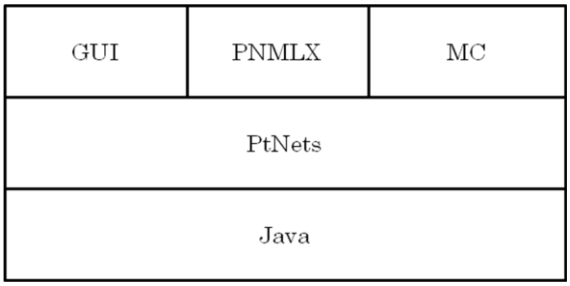


図 3　システム構成

Java で作成した PtNets クラスのオブジェクトをオブジェクト指向ペトリネットの内部表現とし，

PNMLX 形式との間で相互に変換し入出力を行う．モデル検査ツール(MC)として外部ツール Jspin[3]を利用するため，ペトリネットから Promela モデルに変換してモデル検査を行う．

5.2. 実装

5.2.1. エディタ

Editor の実装には Java のライブラリである JGraph X(http://www.jgraph.com/jgraphx.html)を使用した．Editor 上に描かれたペトリネットは，内部で place オブジェクト，transition オブジェクト，arc オブジェクトといったオブジェクトにマッピングし管理する．図 4 にクラス構成と実行画面を示す．

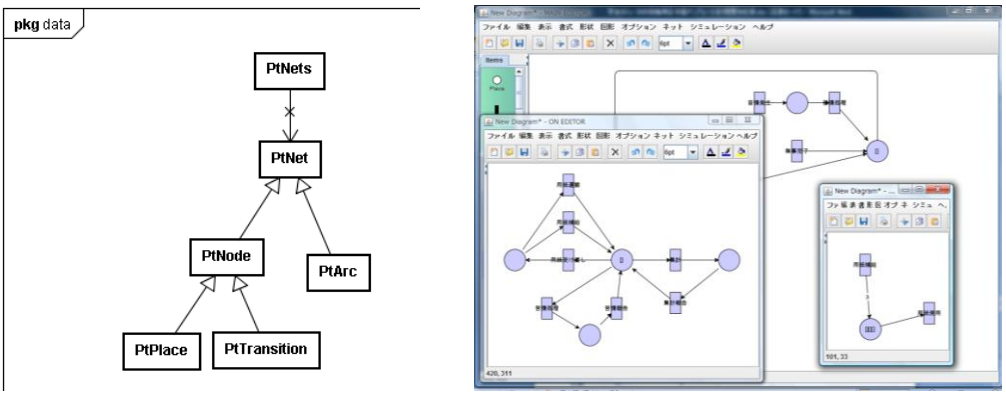


図 4　PtNets クラス群のクラス図とエディタ実行画面

5.2.2. モデル検査部

モデル検査の実装については Java で実装された外部ツール jspin を利用した．図 5 のように，画面左側は Promela モデルの記述であり右側には検査結果で表示される．右上は，検査項目である LTL 式の入力フィールドである．

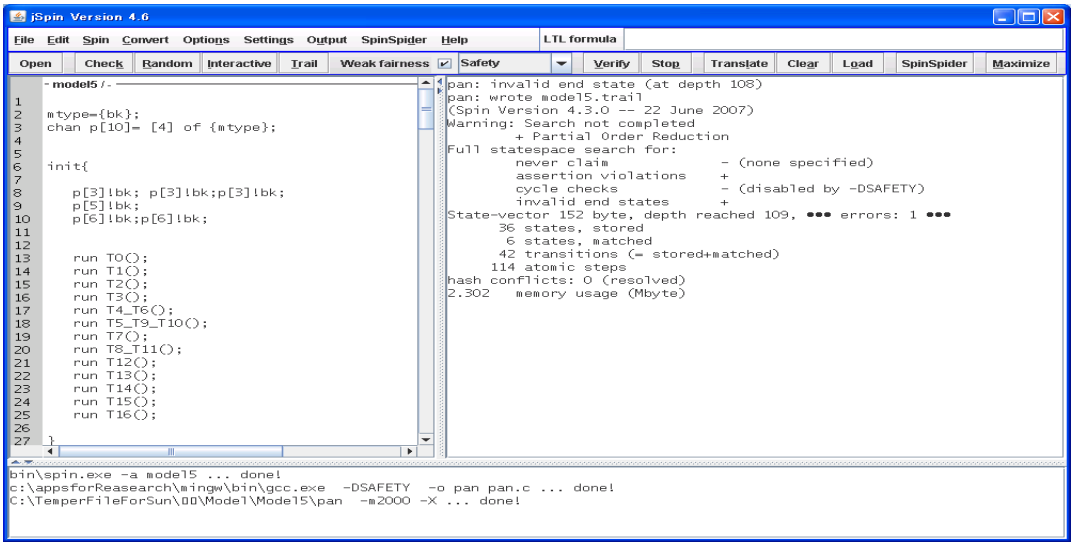


図 5　jspin によるモデル検査の実行画面

6. 実験と評価

6.1. 電話調査会社の業務

ここでは，例として，ある電話調査会社におけるアンケート調査業務を考える．図 6 に示すように，スタッフは調査対象の家に電話をかけ，アンケート協力の依頼をし，アンケート結果を用紙に記入する．また，アンケート中に苦情が発生した場合はマネージャとともに応対する．電話終了後は，新たなアンケート用紙をマネージャから受け取り，次の調査対象へ電話をかける．マネージャはフロアを巡回し，デスク上の用紙ボックスにある新たなアンケート用紙をスタッフに渡す．デスク上の用紙ボックスからアンケート用紙がなくなったら，スタッフとともに，保管庫にある新たな用紙ボックスを取りにいく．

また，集まったアンケートを集計し，本部へ報告するなどの仕事も行う．

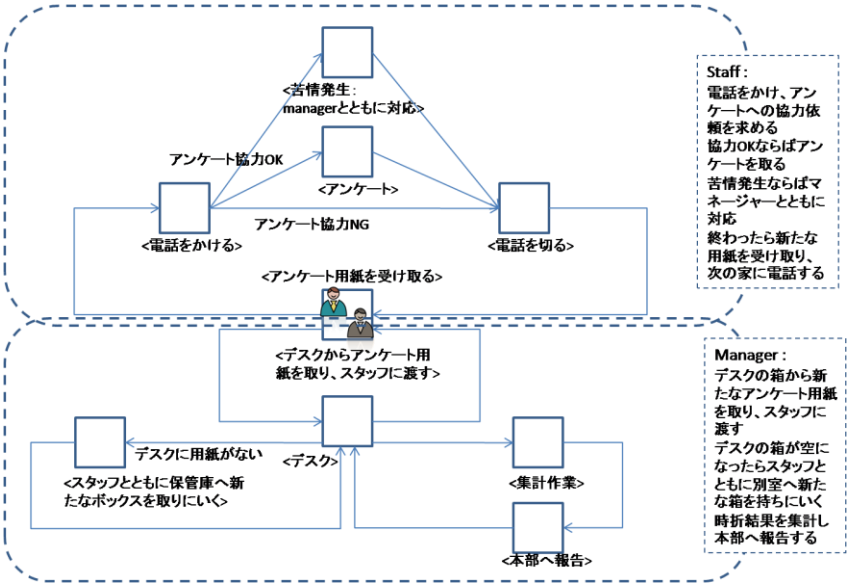


図 6　電話調査会社のワークフロー

6.2. 電話調査会社の業務のオブジェクト指向ペトリネット

上記のワークフローをオブジェクト指向ペトリネットで表わしたものが次の図 7 である．

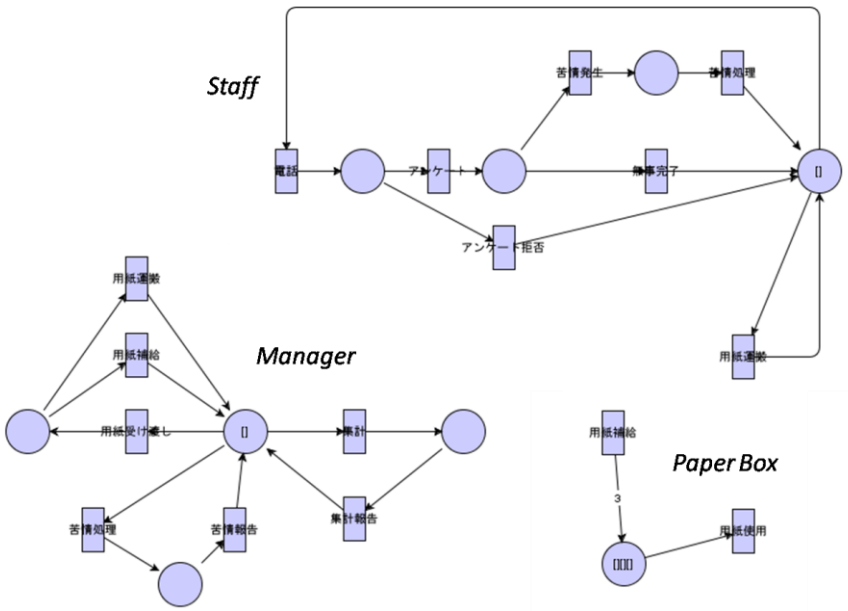


図 7　オブジェクト指向ペトリネットでの表現(staff，manager，paper box)

前にも述べたように，オブジェクト指向ペトリネットをワークフロー記述に利用する利点は，各人単位，機能単位でワークフローの記述ができることである．これによりワークフロー検討時の編集や再利用が容易になる．例えば，フロアマネージャのワークフローに修正の必要が出たときは，フロアマネージャのワークフローのみを取り出し，修正すれば良い．

6.3. 電話調査会社の業務の Promela 言語モデル

上記オブジェクト指向ペトリネットをモデル検査するためには Promela 言語モデルに変換する必要がある．まず，プレースとトークンの表示するために，以下のようにチャンネルと mtype 型を定義する．

```
mtype={bk,Sf,Mg};
chan p[10]= [4] of {mtype};
```

また，各ペトリネットのトランジションは，プロセスとして記述する．非同期で単独発火できる「苦情発生」というトランジションは以下のように記述する．

```
/* 「苦情発生」トランジション　*/
proctype T2()　{
   atomic{
   BeforeFire:
     do
     ::(p[1]??[bk])->goto AfterFire          /* 「苦情発生」トランジションの発火条件の判断　*/
     od;
   AfterFire:
     do
     ::true->p[1]?bk;p[2]!bk;goto BeforeFire          /* 「苦情発生」トランジションの発火後の計算　*/
     od;
  }
}
```

6.4. 検査結果

さらに spin で検査を行って検出できたエラーの結果は，以下の表 3 のようである．

表 3　トランジションの変換ルール

検査項目	LTL 式	システムエラーへのルート
デットロックの存在 (マネッジャとスタッフがお互いに待つ状態になって, 両方の仕事は進めない状態)	無	T13->T14->T13->T3,T3,T3->T14->T7->T8_T11->T13->T0-,T0,T0->T14->T7->T8_T11->T13->T2-,T2,T2>T14->T7 スタッフ 3 人が「苦情処理」でマネージャを待つ．一方，マネージャは保管庫で「用紙運搬」でスタッフを待つ． お互いに待ち状態になり両方の仕事が進まない．
苦情の発生後, マネジャは必ず対応処理を行う	[](p[2]??[bk]-><>p[7]??[bk])	T13->T3,T3,T3->T14->T7->T8_T11->T13->T0,T0,TO->T14->T7->T8_T11->T13->T2,T2,T2,-> <<<<<STARTOF YCLE>>>>>T13->T14 マネジャは「集計」と「集計報告」のワークルートで回っており「苦情処理」ができなくなった．

7. おわりに

第 6 節の電話調査会社の例で見たように，オブジェクト指向ペトリネットを用いてワークフローをモデル化することが可能であった．また，モデル化過程において，オブジェクト指向ペトリネットを用いたことによる利点も確認できた．さらに，そのモデルを Promela モデルに変換することでワークフローの分析を試みた．実験では LTL 式を利用して，ワークフローのエラーなどを自動探索することができ，そのエラー状態に至るまでのルートを出力することができる．しかし，その検証結果は決して読み取りやすいものではない．ワークフローがより複雑になれば尚更である．今後，より複雑なワークフローのために，検証結果を画面上でペトリネットに反映させる機能の開発を検討している．

参考文献

[1] Rüdiger Valk: “Object Petri nets – Using the nets-within-nets paradigm,” LNCS 3098, pp.819-848, Spriner, 2004.

[2] E.M.Clatle,Jr., O.Grumberg, and D.A. Pe(eds.): “Model Checking,” The MIT Press, 1999.

[3] Petri Net Markup Language (PNML) http://www2.informatik.hu-berlin.de/top/pnml/about.html

[4] M. Ben-Ari: “Principles of the Spin Model Checker, ” Springer, 2008.