オブジェクト指向ペトリネットに基づく ビジネスプロセスの時間制約に関するモデル検証

Verification of Temporal Constraints in Business Process Specified by Object Oriented Petri-Nets

秦 良平 †, 飯島 正 ‡
Ryohei HATA†, and Tadashi IIJIMA‡
† 慶應義塾大学大学院 理工学研究科
‡ 慶應義塾大学　理工学部
†Graduate School of Science and Technology, Keio Univ.
‡Faculty of Science and Technology, Keio Univ.

要旨

業務プロセス管理において、時間制約の概念は重要な要素の一つである．著者らは，オブジェクト指向ペトリネットというモデルにより業務プロセスを記述し，そのモデルに時間表現を導入し拡張することに試みている．ここでは，時間表現を導入したオブジェクト指向ペトリネットを時間オートマトンに変換し，UPPAAL というモデル検査ツールを利用して，業務プロセスにおける時間制約をモデル検査手法を用いて検証する試みについて報告する．

1. はじめに

企業において業務改善を行うために，情報システムを用いた業務プロセス管理 (BPM; Business Process Management) は重要である．業務プロセスの自動化においては，タイムアウト処理いわゆる時間制約のルールを考慮する場合が多い．そこで本研究では，業務のモデリングの際に定義した時間制約のルールに対象を絞り，著者らが提案する業務プロセス表記法に対して時間表現を導入することを示す．

近年，企業活動の範囲が広がり，複数の組織が連携して業務プロセスを実行することが多く，業務プロセスは巨大で複雑なものになる．この状況を踏まえ，これまで著者らは，組織間の協調と相互作用を扱うこと，ならびにアクター（実行者/サブシステム）を資源として扱うことを目的として,「オブジェクト指向ペトリネット」というモデルを業務プロセスの表記法として導入してきた．このモデルを用いることで，プロセスを構成要素の役割・機能ごとに分割してモジュール性を持たせ，それらの各要素同士の相互作用の記述のしやすさを示してきた [1][2].

本研究では，著者らが提案する業務プロセスの表記法であるオブジェクト指向ペトリネットに対して，時間制約の表現を導入し，モデル変換を行うことで，時間制約の業務ルールをモデル検査ツールで検証することを目標としている．

続く第 2 節では，対象とする時間制約のルールについて，第 3 節では，オブジェクト指向ペトリネットについて記述する．さらに第 4 節では，一般的なモデル検査手法とモデル検査ツール UPPAAL[10] について述べ，第 5 節でモデル検査のためのモデル変換の方法とその実装について述べる．第 6 節で実装したモデル変換の仕組みを適用したケーススタディを示し，第 7 節で現時点での評価を述べる．

2. 時間制約に関するルール

業務プロセスを効率よく管理するために，近年ではプロセスからルールを独立させて管理する業務ルール管理 (BRM; Business Rule Management) が重視されている．全てのルールは「定義付けのルール（構造ルール）」または「行動ルール（運用ルール）」のどちらかに分類される [5].

定義付けのルールには，たとえば特定区間の運賃を定義するものがある．著者らはこれまで定義付けのルールに対し，if-then ルール形式の DSL[4] により表現し，さらにそれに加え，決定表を導入することで，業務ルールというドメインの知識の共有を実現するとともに，業務プロセス管理の効率化を実現してきた [3].

一方，行動ルールとは，行為や行動に関する制約を表現することが多く，必要に応じて，判断や行動に関する評価をも含む．当事者の行為や行動を規制する制約条件の中で，時間制約のルールは重要である．業務プロセスの自動化には，タイムアウト処理はほぼ不可欠といえるからである．そこで本研究で

は，時間制約に関する業務ルールを対象を絞り，オブジェクト指向ペトリネットによる業務プロセス表現に時間表現を導入することを試みている．

3. オブジェクト指向ペトリネット

オブジェクト指向ペトリネットを業務プロセスを表現するモデルとして用いる．オブジェクト指向ペトリネットとは，オブジェクト指向の概念に基づいてモジュール性を取り入れたペトリネットの拡張モデルであり，著者らは，nets-within-nets 意味論に基づく参照ネット (Reference Net)[6] の考え方を採用している．ペトリネットで業務プロセスを記述する先行事例に YAWL[7] があるが，本研究は，この nets-within-nets 意味論に基づくモジュール性の導入に特徴がある．このモデルの例を示す (図 1)．

nets-within-nets 意味論 [6] とその拡張に基づくオブジェクト指向ペトリネットは次の 2 種類のトークンを持つ．

- 単純トークン (Black Token)
  通常の P/T ネット (Place/Transition ネット) におけるトークンのことである．プレースに存在する単純トークンは，トークン数で示す．

- 参照トークン (Reference Token)
  別のサブシステムを表現するサブネットへの参照 (reference) を持つトークンのことである．

図 1 の例では，クラス A のインスタンスへの参照を持っているトークンが参照トークンであり，クラス A のインスタンスを表現するペトリネットのプレース上に存在するトークンが単純トークンである．

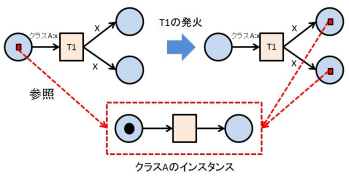

図 1: 参照トークンの例

この例で，参照トークンを持つ側のネットをシステムネットと呼び，参照トークンが参照している先のサブネットをオブジェクトネットと呼ぶ．このオブジェクトネットは，一つのオブジェクト (インスタンス) として識別される．

参照トークンも単純トークンと同様，ネットワーク中を遷移する．参照トークンがトランジションの発火によって複数個にコピーされるときは，オブジェクトネットの実体ではなく，それへの参照がコピーされる点に注意して欲しい．

また，オブジェクトネットおよびシステムネット同士で，トランジションを同期発火 (interaction) の関係を持たせることも可能となっている．この同期発火が可能となるのは，全ての同期トランジションが発火可能な状態であることが条件である．nets-whithin-nets 意味論 [6] とその拡張に基づくオブジェクト指向ペトリネットにおいては，(a)interaction(相互作用):システムネットとオブジェクトネットが同期して遷移する場合，(b)transport(移送):システムネットの発火に際し同期して発火するオブジェクトネットがない場合，(c)autonomous(自律):システムネットとは独立に，オブジェクトネット内のトランジションが発火する場合の 3 通りの発火則が考えられる．

4. モデル検査手法とモデル検査ツール UPPAAL

モデル検査とは，システムのモデルを設計した時点およびその範囲で，そのシステム上で起こり得る状態を網羅的に調べ,「システムが満たすべき動作を完了する」あるいは「満たすべきでない状態にならない」といった検証項目を確認する手法である [9]．

モデル検査手法は，組込みソフトウェアの検証において多く用いられているが，将来的に大規模化，複雑化していくと考えられる業務システムの開発においても，モデル検査手法を導入することで，システムの不具合の早期発見につながると期待できる．本研究では，そのような流れを汲み，著者らが提案する業務プロセス表記法でモデル検査を実施するためにモデル変換機能を導入する．

モデル検査を行うツールとして，SPIN[8] を代表に，数多くのものが存在する．しかし，それらの多くは動作ロジックに関する性質を検証するに留まり，時間制約に関する性質を検証する機能が含まれて

いない．

そこで，時間制約の性質についても検証可能なモデル検査手法の研究が行われてきた．その手法として，時間制約を表現することができる時間オートマトンに基づいたものがある．そのためのツールとして UPPAAL[10] が，デンマークの Aalborg 大学とスウェーデンの Uppsala 大学の共同で開発された．本論文では，この UPPAAL を採用している．

5. モデル検査のためのモデル変換

オブジェクト指向ペトリネットにより記述した業務プロセスのモデルに対して，時間制約に関するルールについてその性質の検証を実施するために，UPPAAL で検査可能なモデルである時間オートマトンに変換する．

本方式で前提とするのは，業務プロセス全体が一つの有界ペトリネットで記述できることである．有界性は，オートマトンへの変換の際に，状態数を有限に抑えるために不可欠である．この有界性は，単純トークンだけでなく，参照トークンにも求められる．すなわち，同じクラスのオブジェクトネット (インスタンス) は複数存在しても構わないが，その個数には上限がある．さらに，もう一つ強い条件として，実行途中に新たなオブジェクトネット (インスタンス) を生成させず，その上限個数のオブジェクトネット (インスタンス) を事前に全て生成しておき，全体で一つの有界ペトリネットに合成できるものとする．

時間オートマトンに変換する手順として，まず有界なペトリネットに時間表現を導入する．プロセスを構成する各オブジェクトの振る舞いを記述し，さらに時間表現が導入されたオブジェクトネット同士を合成し，一つのペトリネットを得る．その一つのペトリネットを一つの時間オートマトンに変換することにより，その一つの時間オートマトンを対象としたモデル検査を可能とさせる．

本来，オブジェクト指向ペトリネットのモデル検査のためには，ペトリネット (オブジェクトネット) を合成する代わりに，個々のオブジェクトネットを，それぞれ同期しあう並行プロセス (時間オートマトン) に変換した上で，モデル検査を行う方法もありうる．しかし，本論文の執筆時点では，オブジェクト指向ペトリネットに時間表現を付与するにあたり，まずは時間制約に関するビジネスルールとして，どのような表現力を与えるべきか模索しつつ，それをペトリネット上で表現し，さらに使用するモデル検査器 (今回は UPPAAL) で検証できるプロセス表現 (時間オートマトン) に変換できることを確認することを目的にとした．そのため，各オブジェクトネットを一つのペトリネットに合成して，時間オートマトンに変換する方法を採用し，まず第一段階として有用性の確認を試みた．

5.1. 有界ペトリネットから有限オートマトンへの変換

まず，時間制約のない有界なペトリネットからオートマトンへ変換することを考える．ペトリネットでは，全てのプレース上におけるトークン数をベクトルで表現したマーキングをシステムの一つの状態と見なせる．また，トランジションの発火によるマーキングの変化を「状態間の遷移」として対応付けられる．つまり，ペトリネットの全ての取り得るマーキングとその際の発火可能なトランジションを調べ上げることにより，有界ペトリネットから有限オートマトンを対応付け，変換が可能となる．図 2 は，この操作による有界ペトリネットから有限オートマトンへの変換の例である．

ここまで，ペトリネットのトークンが単純トークン (ブラックトークン) の場合を扱ってきたが，オブジェクト指向ペトリネットの場合，トークンが参照トークン (リファレンストークン) の場合がある．図 2 の 2 つのトークンが ON1，ON2 というオブジェクトネットへの参照を持つ場合，マーキングは単にトークン数で表現することができず，トークンがそれぞれ識別されるため，変換後のオートマトンの状態数は，図 3 のように組み合わせ的に増える場合がある．

5.2. ペトリネットへの時間表現の導入

オブジェクト指向ペトリネットを時間オートマトンに変換するために，時間オートマトンのモデル上で定義される時間表現を変換前のモデルに定義しておく．変換の方法として，最下層のオブジェクトネットに時間表現を導入し，それらを一つのペトリネットに合成して，時間オートマトンに変換する方針であるため，単純なペトリネットへの時間表現の導入を考える．時間オートマトンで定義される時間表現で，変換前のモデルにも定義したものは，以下の 3 つである．

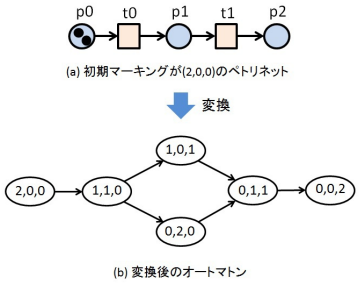

図 2: 初期マーキングが (2,0,0) の場合の変換

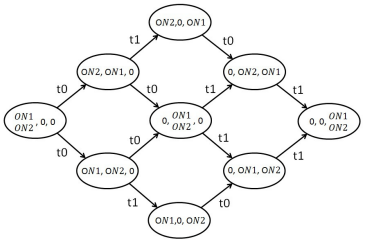

図 3: 参照トークンの場合の変換後のオートマトン

- **guard(ガード)**:プロセスが遷移するときに満たさなければならない条件である.
- **invariant(不変式)**:プロセスのある状態に滞在可能な条件である.
- **assignment(更新)**:変数の代入や初期化を行う式である.

ペトリネットに時間表現を導入する方法として，ペトリネットで状態遷移の要因となるトランジションの入力アークに先の 3 つの時間表現を導入した．これは，状態が遷移する際に条件を判定したり，変数の更新を行うためである．ペトリネットに時間表現を導入したものが，図 4 におけるトランジション t0 への入力アーク上の条件式で表現される．上から順に，guard(ガード)，invariant(不変式)，assignment(更新) を表現している．

5.3. 時間表現を導入したペトリネットの合成

続いて，それら各オブジェクトネットを一つのペトリネットに合成する方法を述べる．たとえば，図 5 のようなオブジェクト指向ペトリネットを考える．オブジェクトネット $ON1$ とオブジェクトネット $ON2$ のトランジション $t0$ が同期して発火するとする．$ON1$ の $t0$ の入力アークに時間表現を取り入れている．

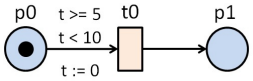

図 4: ペトリネットへの時間表現の導入

図 5 で記述される時間表現を導入したオブジェクト指向ペトリネットのモデルを，一つのペトリネットに合成する (図 6)．今回はトランジション間の同期制約の定義を簡単化するため，各オブジェクトネットの同期トランジションには同じ名前を与えており，合成の際には同名のトランジションを全て同一化する．名前を異なるものにしておくと，合成後はそのトランジションを保持したまま合成される．同様に，各オブジェクトネットのプレースにおいても，他のオブジェクトネット上のプレースと同じ名前を与えると，合成後に一つにまとめられ，名前が異なればそのまま保持される仕組みを取っている．

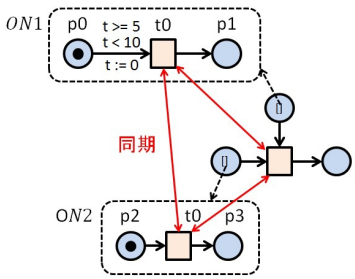

図 5: オブジェクト指向ペトリネットの例

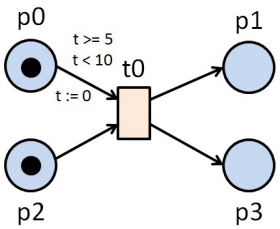

図 6: 合成後のペトリネット

5.4. モデリング・エディタによるモデル変換

前節までに述べた，オブジェクトネットに時間表現を導入し，オブジェクトネットを一つのペトリネットに合成して，時間オートマトンへモデル変換を自動で行える図形エディタを実装した．

6. ケーススタディ

時間表現を導入したオブジェクト指向ペトリネットで業務プロセスをモデル化し，そのモデルを時間オートマトンに変換し，時間制約に関する業務ルールを UPPAAL で検証する事例を示す．事例として，企業における社員の出張旅費申請のプロセスを挙げる．

6.1. 事例を用いたモデル変換

企業における社員の出張旅費申請のプロセスにおいて，プロセスを構成する実体となるオブジェクトについて，社員，申請書，課長，財務部を定義した．それぞれをオブジェクトネットとして定義し，オブジェクト指向ペトリネットでモデル化すると，図 7 のように定義できる．時間表現は申請書のモデルに記述している．今回申請書に時間表現を記述したのは，後に，ある時間において申請書が満たすべき状態を決める業務ルールの検証を行うことを目的としているためである．

定義した業務プロセスは次のような流れである．社員が出張旅費の申請書を作成し，申請書を課長に提出して，課長が受け付ける．課長は申請書の承認を行い，財務部へ申請書を提出して，財務部が受け付ける．財務部は申請書の内容を審査し，その結果を申請者である社員に通知し，社員が受信する．

前節の時間オートマトンへの変換方法に従い，図 7 の時間表現付きオブジェクト指向ペトリネットを一つの時間ペトリネットに合成すると，図 8 のモデルが得られる．

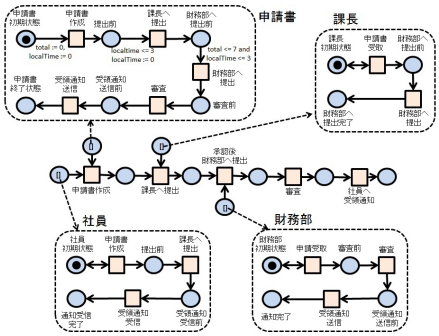

図 7: 出張旅費申請のプロセス

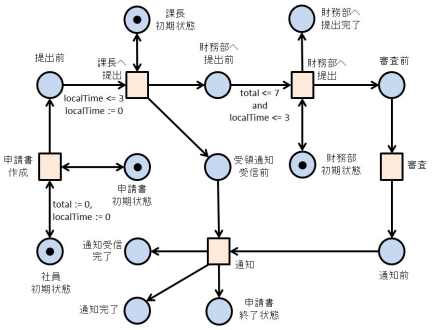

図 8: 出張旅費申請プロセスの合成ペトリネット

このモデル変換の一連の流れを実装し，モデリング・エディタの中で合成機能を実施した様子を図 9，図 10，図 11 に示す．エディタにより合成したペトリネットのノードの配置は，変換操作の際，自動的にレイアウトを行うため，図 8 とは見た目が異なる．

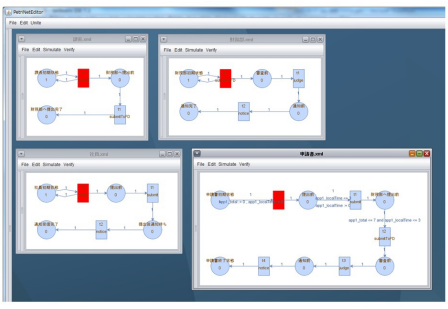

図 9: エディタ上でのオブジェクトネットの記述

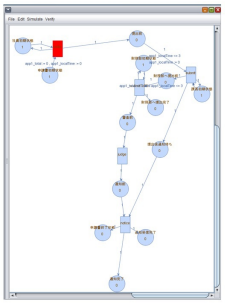

図 10: エディタ上での合成ペトリネット

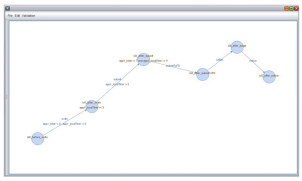

図 11: エディタ上での事例の時間オートマトン

6.2. 時間制約に関する業務ルールの検証

事例に対し，たとえば「出張旅費申請書に関して，社員がそれを作成した時点から，7 日以内に課長の承認を得て，財務部へ提出しなければならない.」という時間制約に関する以下のような業務ルールを定義し，この成立を検証する．これは検証にあたって次のように言い換えが可能である.「出張旅費申請

書が社員により作成された時点から 7 日以上経過しているとき，申請書は課長への提出前の状態または財務部へ提出前の状態にはあってはならない.」

UPPAAL を用いて上記の業務ルールを検証するには，それを時相論理式で記述しなければならない．それは次のように記述することができる．

- `A[] ( total >= 7 imply not (Process.after_write or Process.after_submit) )`

この検証式は，すべての実行系列で括弧内の性質，つまり業務ルールを言い換えた表現で示された性質が成立することを意味する．UPPAAL 上で事例の時間オートマトンモデルが展開されているの様子と，上記の検証式により，検証を行っている様子を示す (図 12 および図 13).

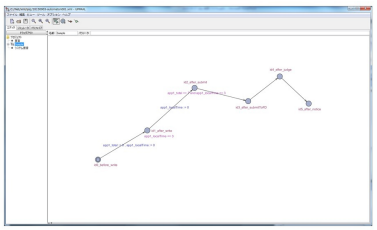

図 12: UPPAAL 上での時間オートマトンモデルの記述

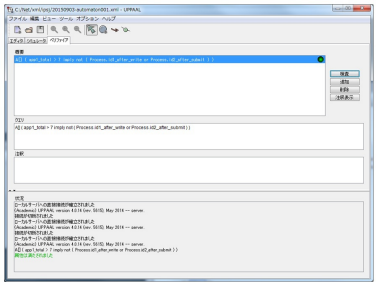

図 13: 検証式と検証

7. 評価

実装したエディタにより，時間表現付きペトリネットの合成，および，時間オートマトンへの自動変換，さらに UPPAAL での業務ルールの検証という，オブジェクト指向ペトリネットによる業務プロセスのモデルの設計から，モデル検査までの一連の流れを実行できることを確認できた．モデル記述の際に，初めから巨大なプロセスを記述する方法より，分割してモデルを記述し，それらを自動合成する方式で，モデル記述の効率性は向上したといえる．

しかし，今回は時間表現の記述はオブジェクトネットの中の一つに限定した．同じ時間変数が複数のオブジェクトネット上の時間表現の中で記述されたとき，それらをどのように合成するかには，現在対応していない状況である．

また，業務ルールの検証に関しては，プロセスの状態がどうあるべきか「義務」の様相に関して，検証可能であることが示された．しかし，他に「可能」や「許可」などの様相を用いて表現される業務ルールに対しての検証の可否は確認はできていない．

参考文献

[1] 飯島 正: “アスペクト指向ワークフロー変換 ～ オブジェクト指向ペトリネットによるワークフロー表現への適用 ～,” 技術報告 (知能ソフトウェア工学研究会), Vol.112, Vol.165, pp.1-6, 電子情報通信学会, 2012.

[2] 飯島 正, 片山 輝彦, 金子 良太, 高橋 貴大: “オブジェクト指向論理ペトリネットを使った業務プロセス/業務ルール管理,” 第 8 回 全国大会・研究発表大会, 情報システム学会, 2012

[3] 飯島 正, 秦 良平, 金子 良太: “オブジェクト指向ペトリネットとルールに基づく業務プロセスの理解支援,” 第 9 回 全国大会・研究発表大会, 情報システム学会, 2013

[4] Debasish Ghosh, (監訳) 佐藤 竜一: “実践プログラミング DSL ～ ドメイン特化言語の設計と実装のノウハウ,” 翔泳社, 2012.

[5] ロナルド・G・ロス: “アジャイル経営のためのビジネスルールマネジメント入門,” 日経 BP 社, 2013.

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

[7] W.M.P. van der Aalst and A.H.M. ter Hofstede: “YAWL: Yet Another Workflow Language,” QUT Technical Report, FIT-TR-2002-06, Queensland University of Technology, Brisbane, 2002.

[8] “SPIN”, `http://spinroot.com/spin/whatispin.html`

[9] (監修) 大須賀昭彦，(著者) 長谷川哲夫，田原康之，磯部祥尚: “UPPAAL による性能モデル検証,” 近代科学社, 2012.

[10] “UPPAAL”, `http://www.uppaal.org/`