カラーペトリネットを用いた 分散ワークフロープロセスのモデリングとその解析

小林　昌徳† 　横田　治夫††

† 東京工業大学 大学院 情報理工学研究科 計算工学専攻
†† 東京工業大学 学術国際情報センター
〒 152–8552 東京都目黒区大岡山 2–12–1
E-mail: †masanori@de.cs.titech.ac.jp, ††yokota@cs.titech.ac.jp

あらまし　近年，分散トランザクショナルワークフローの必要性が増大しているが，アクティブデータベースを分散かつ独立に配置し，それらノード群が協調動作しながら分散トランザクショナルワークフローを実現するメカニズムとして，DIADEM が提案されている．我々は，DIADEM におけるワークフローをカラーペトリネットを用いてモデル化，それを解析する手法を提案してきた．本稿ではまず，DIADEM におけるノード間連携関係を明確に記述するため，特殊なトランジションとプレースを提案する．また，DIADEM におけるプロセス定義設計フェーズの手順を定め，拡張したカラーペトリネットによるノード間での障害の扱いについて述べる．その後，拡張したカラーペトリネットはモニタリングフェーズにも適用可能であることを示す．そして実験システムを実装し，提案した手法の有効性を確認する．

キーワード　DIADEM，分散トランザクショナルワークフロー，カラーペトリネット，アクティブデータベース，プロセス定義

The Modelling and Analysis for Distributed Workflow Process using Coloured Petri Nets

Masanori KOBAYASHI† and Haruo YOKOTA††

† Department of Computer Science, Tokyo Institute of Technology
†† Global Scientific Information and Computing Center, Tokyo Institute of Technology
E-mail: †masanori@de.cs.titech.ac.jp, ††yokota@cs.titech.ac.jp

Abstract　Importance of distributed transactional workflows increases recently. As a mechanism to handle them, the DIADEM has been proposed, which constructs nested transactions spread over distributed independent active databases. To model and analyze workflows in the DIADEM, we have already proposed a method using Coloured Petri Nets (CPN). In this paper, we propose special transitions and places for CPN to describe relationships between nodes in the DIADEM clearly. We define a procedure in a design phase and a policy of failure treatments between nodes with the extended CPN. We show that the extended CPN is also applicable for a monitoring phase. We then implement an experimental system to demonstrate usefulness of the proposed method.

Key words　DIADEM, distributed transactional workflow, Coloured Petri Nets, Active Database, workflow-process definition

1. はじめに

日本経済の低迷に苦戦する各企業が生き残りをかけたビジネス再編の必要性に迫られている昨今，ビジネスプロセスにおけるコスト削減，時間短縮，情報共有の効率化，そして BPR(*B*usiness *P*rocess *R*eengineering) の実現手段としてワークフロー管理システムの導入が有効と期待されている．

ワークフロー管理団体 WfMC [1] によるとワークフローとはビジネスプロセスの全体または一部を自動化することと定義されている．インスタンスをトランザクションの保護の下で実行するトランザクショナルワークフロー [2] が，データベースの一貫性を達成する上で信頼性が高いと認識されている．また近年では，単一の組織内での閉じたワークフローだけでなく，他組織間にまたがる分散ワークフローのニーズがある．

そこで，分散かつ独立配置したアクティブデータベース [3] 間のメッセージ通信によって分散トランザクショナルワークフローを実現する方式として DIADEM(Distributed Independent Active Database with subtransaction Exporting Mechanism) [4] [5] [6] が提案されている．DIADEM は，アクティブルールにより発生するトランザクションを，その階層構造を保ちながら独立した異なるアクティブデータベース上で実行することを特徴とする．

我々は DIADEM において，他ノードにおける見えない処理をカラーペトリネット [7] の階層化の概念を用いて処理委譲を表現することで，プロセス定義の解析を行う手法を提案してきた [8]．本稿では更に，DIADEM のモデリングに適するよう特殊なトランジションとプレースを導入，そのカラーペトリネットを用いてプロセス定義設計フェーズ及びモニタリングフェーズを扱う仕組みを提案する．また実験システムを実装，ワークフローの例を用いて提案した仕組みの有効性を確認する．まず第 2. 章では，分散トランザクショナルワークフローのメカニズム DIADEM の特徴を述べる．次に第 3. 章では，DIADEM におけるワークフロープロセスモデリングを行うためのカラーペトリネットを提案する．提案したカラーペトリネットをプロセス定義設計フェーズとモニタリングフェーズにおいてどのように活用するかそれぞれ第 4. 章，第 5. 章で解説する．第 6. 章では実装した実験システムを概観し，第 7. 章でそのシステムを用いた実験を述べる．第 8. 章で関連研究について述べ，第 9. 章では，まとめと今後の課題について述べる．

2. DIADEM

2.1 DIADEM の特徴

DIADEM は，ワークフロー処理のためのアクティブルールを ECA(Event-Condition-Action) ルール [9] としてアクティブデータベースに積み，分散かつ独立に配置されたそれらノード群で協調動作しながらワークフローを実現するメカニズムである．

アクティブデータベースとは，ルールベースとデータベースを統合してデータベースが能動的に動作することを可能にしたものであり，既存のデータベースを大きく変更することなく利用することができる．

DIADEM における各ノードは，自ノードにおいて処理されるトランザクションに対してのみ責任を持つ．DIADEM のノードには，自分の分担する処理を記述した ECA ルールを保持し，DIADEM 全体としてワークフローを構成する．

DIADEM の主な特徴は以下の通り．

入れ子トランザクション　DIADEM におけるワークフローは入れ子トランザクション [4] を基礎にして処理される．

カップリングモード　アクティブデータベースにおけるアクティブルールは，ECA(Event-Condition-Action) ルールにより記述される．その際，E-C 間，C-A 間で生成されるトランザクションとの関係には以下の 4 種類がある．

- **Immediate モード**　子トランザクションとして生成，即座に実行される．
- **Deferred モード**　子トランザクションとして生成されるが，親トランザクションが pre-commit 状態になるまで待機する．
- **Independent Decoupled モード**　別トランザクションとして生成，即座に実行される．
- **Dependent Decoupled モード**　別トランザクションとして生成されるが，生成した親トランザクションが pre-commit 状態になるまで待機する．

カップリングモード 別のロールバック/補償範囲　Immediate, Deferred モードによって生成された子トランザクションがロールバック/補償される際，親トランザクションまで及ぶ．対照的に，Decoupled モードによって生成された子トランザクションがロールバック/補償される際，親トランザクションまで及ばない．親トランザクションがロールバック/補償される際，いかなるモードにおいても全ての子トランザクションはロールバック/補償される．

3. カラーペトリネットを用いたワークフロープロセスモデリング

3.1 DIADEM の問題点

各アクティブデータベースにおける ECA ルールは入れ子構造を持たないが，DIADEM では ECA ルール間が連鎖し合いながら分散したワークフローの処理方法を記述しなければならない．従って，ユーザにとって全体の挙動の把握が難しく，プロセス定義設計フェーズにおいては設計ミスをしたり，モニタリングフェーズにおいては障害の原因特定が難しかったりした．これらの事由から，DIADEM におけるワークフロープロセスモデリングのフレームワークを示す必要があった．

3.2 解決へのアプローチ

通常のペトリネット (プレース/トランジションネット) を拡張して論理的記述を可能にした拡張ペトリネットをカラーペトリネットという．これは，一つのトランジションに別のカラーペトリネット（ページ）を対応させることで，階層構造表現が可能になる．

カラーペトリネットを DIADEM のワークフロープロセスモデリングに適用することで以下の利点が得られる．

離散事象システムの記述　離散事象システムであるワークフローの記述に適している．

入れ子構造の記述　DIADEM の入れ子構造を，カラーペトリネットの階層化の概念を用いて記述できる．

グラフィカル表現　ワークフロープロセスの挙動を視認しやすい．

形式的意味論　明確なワークフローの意味論を定義できる．

解析力　単なるユーザインターフェースのためのモデリング手法でなく，構造的，性能的解析が行える．

汎用性　プロセス定義設計フェーズとモニタリングフェーズ両方に適用可能である．

3.3 DIADEM 記述用ペトリネットの導入

従来の DIADEM は，自ノードや他ノードでの処理の識別が明確でなかったが，自ノード内でのイベントや処理と，他ノードからのイベントや処理を識別して設計することで，障害検出場所の特定や問題発見が容易になる．

そこで本稿では，図 1 のように表記される特殊なトランジションとプレースを導入することで，自ノードと他ノードでのイベントや処理を識別している．

DIADEM 内で使用するトランジションは次の 2 種類.

内部トランジション　自分のノード内での処理を表現.

外部トランジション　他のノード内での処理を表現.

DIADEM 内で使用するプレースは次の 5 種類.

内部イベント入力プレース　内部イベントの受信を示すプレース．入力トランジションを持たないが，対となる内部イベント出力プレースを持つ.

内部イベント出力プレース　内部イベントの終了を示すプレース．出力トランジションを持たないが，対となる内部イベント入力プレースを持つ.

外部イベント入力プレース　外部からの処理委譲を示す外部イベントを受け取るプレース．入力トランジションを持たないが，対となる外部イベント出力プレースを持つ.

外部イベント出力プレース　外部から委譲された処理を終了し，トークンを委譲元へ返すプレース．出力トランジションを持たないが，対となる外部イベント入力プレースを持つ.

ルール間連結プレース　自ノード内のルールとルールを連結するプレース．入力トランジション，出力トランジションを 1 つ以上持つ.

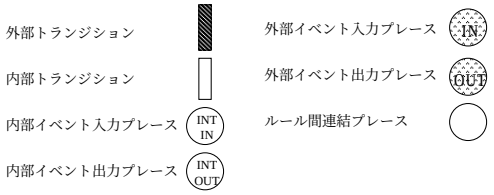

図 1　表　記　法

カラーペトリネットと ECA ルールのマッピングは，表 1 の通りに行う.

表 1　カラーペトリネットと ECA ルールのマッピング

ペトリネット	ECA ルール
トークン	インスタンス
入力プレース	イベント
ガード関数	コンディション
トランジション	アクション

3.4 ノード間連携関係の方針

従来の DIADEM は，ノード間連携関係を明示してなかったため，設計ミス情報や障害情報をどのノードに，そしてどのように伝達するべきか曖昧だった.

ここでは，ノード間連携関係の方針を以下のように定める.

設計フェーズ，モニタリングフェーズ共に以下の方針に従うノード間連携関係を持つとする．ここで，ある親トランザクションを持つノードを N_0 とし，$N_0, ..., N_{i-1}, N_i, N_{i+1}, ..., N_n$ という親子関係を持つとする.

（1） N_0 内の処理に関する障害は N_0 内で対策し，解決する.

（2） N_i において，

- 外部トランジションに関する障害は，関係するノード名とその障害情報を N_{i-1} へ伝達する.
- 内部トランジションに関する障害は，N_i で障害が発生したことのみを N_{i-1} へ伝達する.

3.5 DIADEM の処理フェーズ

以下のフェーズを辿る.

（1） プロセス定義設計フェーズ：ワークフロープロセス定義を設計し，設計ミスの解析を行い，ECA ルールへ変換してルールテーブルへ格納する.

（2） モニタリングフェーズ：DIADEM の挙動をモニタリングし，障害発生時には，ユーザインターフェースを介してユーザへ伝達，必要ならば他ノードへ障害情報を伝達する.

（3） 性能解析フェーズ：モニタリングフェーズにより得られたログからプロセス定義の性能を評価し，プロセス定義設計フェーズへ移り再定義を行う.

性能解析フェーズにおける解析すべき要素は，実行環境やワークフローの種類に強く依存するため，一般的な指標を示すことは難しい．従って，本稿では扱わない．

4. プロセス定義設計フェーズ

4.1 プロセス定義設計手順

以下の設計手順に従い，プロセス定義を決定し，DIADEM へ実装される．

（1） 各ノード内のワークフロープロセス定義をユーザインターフェースを通じ，カラーペトリネットを用いて設計する．
（2） 対応するイベント，コンディション，アクションを記述する．
（3） 自ノードのペトリネットシステムを通じて他のペトリネットシステムと連携しながら，プロセス定義の可達性，活性，有界性判定を行う．
（4） ペトリネットシステムの変換器により，定義したペトリネットから ECA ルールへ変換されて DIADEM へ実装，稼動する．

4.2 プロセス定義が満たすべき性質

実際のワークフロープロセスに要求される性質は環境やプロセス定義により様々であるが，DIADEM ワークフローの設計における共通して満たすべき性質として，以下の 3 つの指標を取り上げる．

可達性 指定した終端状態に達することなくワークフローが完遂することはない．従って終端状態に達しないプロセス定義は設計ミスである．これは，カラーペトリネットの可達性解析により判定することができる．

本稿における可達性解析とは，イベント入力プレース i を含む状態 M から生起されるパス全てにおいて，i と対となるイベント出力プレース o を含む状態 M' が存在することを示すことである．

活性 意図したワークフローとは関係のない実行されないルールを記述するかもしれない．実行されないルールの存在は設計ミスとする．これは，カラーペトリネットの活性解析により判定することができる．

本稿における活性解析とは，自ノードにおける全てのトランジションは，それぞれ少なくとも 1 回は発火することを示すことである．

有界性 1 つのインスタンスが無限のインスタンスを生成することはない．これは，カラーペトリネットの有界性解析により判定することができる．

本稿における有界性解析とは，全てのプレースにおいてどのような状態でもトークンの数は有限であることを示すことである．

4.3 解析手法

プレース/トランジションネットの可達グラフは，カラーペトリネットでは Occurrence Graph という．トランジション t とそのガード関数を真にする束縛 b との対 (t,b) を束縛要素というが，Occurrence Graph とは，初期状態から到達可能な状態間を束縛要素により連結した有向グラフのことである．このグラフを生成することで動的な挙動の全探索が可能になる．有限なトークン数にもかかわらず状態空間爆発を起こす可能性があるが，本稿では簡単のため，扱うカラーペトリネットの状態空間は有限と仮定する．

可達性，活性，有界性の判定は，入力プレース全てに解析用トークンを配置した状態を初期状態として Occurrence Graph を生成し，それを解析することにより行う．

5. モニタリングフェーズ

5.1 モニタリングの手順

以下の手順に従うことでモニタリングを行う．

（1） ルールモニタを通じて自ノードの挙動をモニタリングする．
（2） ルールモニタが障害を検出した場合，ペトリネットシステムを通じてユーザインターフェースへ報告する．

この手順を実現するために，次節以降，トークンとトランジションの挙動を明らかにする．

5.2 トークンの挙動

トークンは，以下のコントロール情報を持つ．

InstanceID ワークフローインスタンス ID
BeginID ワークフローを開始したノードの ID
ExtID 外部イベントを発生させたノードの ID
TransID 直前に発火した入力トランジション ID

これにより，以下のケースに対応可能となる．

外部イベント出力プレースにトークンが入った場合 ExtID を元に，外部トランジション処理を委譲したノードへの完了報告行う．
障害によりトークンが遷移不能になった場合 アクションが完了できない，あるいはコンディションが真にならないなどの理由によりトークンがこれ以上遷移できなくなった場合，障害情報 (Ex. 外部トランジションが発火できない，コンディションが真にならなかった等) をトークンに与え，TransID を元に発火系列を逆順に辿ることで障害情報を収集する．

5.3 トランジションの挙動

トランジションはアクションを行うだけでなく，障害検出などのためのロギングやトークンの ID を変換を行う．

トークンの InstanceID の変換 トランジションが複数の入力プレースを持つ時，それら複数のトークンの InstanceID が異なっていれば 1 つの InstanceID に変換しなければならない．トランジションの発火により，トークンの (旧

InstanceID ⇒ 新 InstanceID) への変換を行い，その変換履歴をトランジションが保持する．

トークンの TransID の変換　トークンは直前に通過したトランジションの ID，即ち TransID を保持する．トランジションの発火後，その ID を発火した TransID に変換し，出力プレースへそのトークンを渡す．その変換履歴をそのトランジションは保持する．

外部トランジション　外部トランジションは他ノードに処理を委譲し，その完了報告を待つ．従って，外部トランジションの発火により，InstanceID を新しい ID に変換し，また ExtID と TransID に自分の TransID を保持して新しく生成したトークンを送信する．委譲先のノードにおける処理の完了報告を受けたら，ExtID を以前の ID に変換して処理を継続する．

5.4　障害検出時の動作例

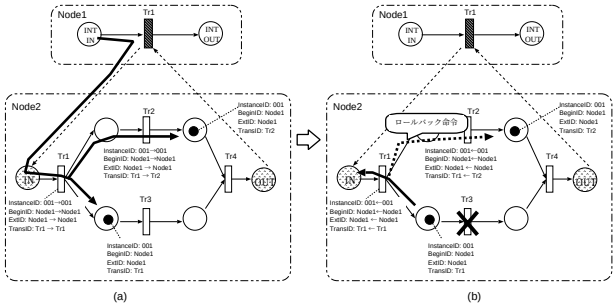

図 2　モニタリングにおけるトークンとトランジションの挙動

図 2 (a) は，Node1 から受けた処理委譲を Node2 が処理している途中を表現している．Node2 の Tr1 や Tr2 において通過したトークンが持つコントロール情報のロギングが行われていることが分かる．

図 2 (b) は，Node2 の Tr3 が何らかの障害により発火できなかった直後を表現している．2.1 節にて，DIADEM はカップリングモード別のロールバック/補償の範囲を持つことを述べたが，(b) はトークンがコントロール情報とトランジションにおけるログを元にして発火系列を逆順に辿ることで，ロールバック/補償が行われる様子を表現している．Node2 の Tr1 を親トランザクションとする Tr3 のカップリングモードは immediate/deferred であり，Tr2 に対してロールバック命令を発行している．

6.　実験システムの実装

6.1　システム構成

これまで紹介した機能の有効性を明らかにするため，実験システムを構築する．

実装には，PC/AT，Java JDK1.3.1，PostgreSQL といった入手しやすいコンポーネントを用いた．

図 3 は，従来の DIADEM マネージャにペトリネットシステム，ルールモニタを連携させ，カラーペトリネットを GUI

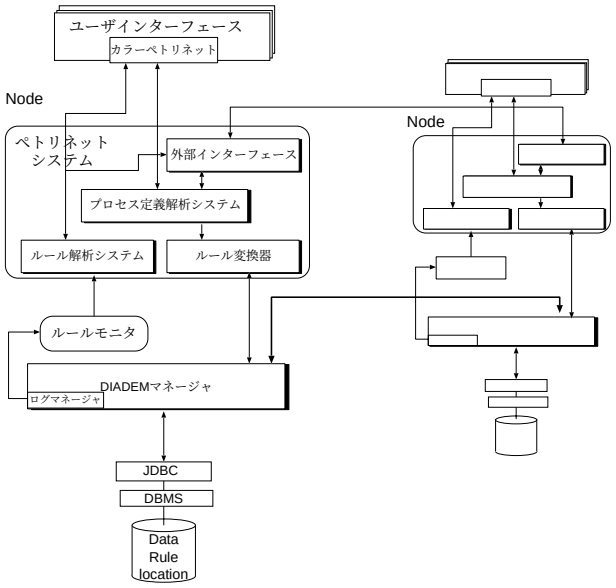

図 3　実験システムの構成

により設計することができるユーザインターフェースを持つシステム構成を表している．処理の委譲は RMI を用い，データベースとの接続は JDBC を用いる．

ユーザインターフェース　ワークフロープロセス定義を行い，可達性，活性，有界性の解析，ECA ルールへの変換，その格納の操作を行う．また，ルールモニタから伝達される障害情報をペトリネットを介してユーザは知ることができる．図 4 はプロセス定義を行う様子のスクリーンショットである．

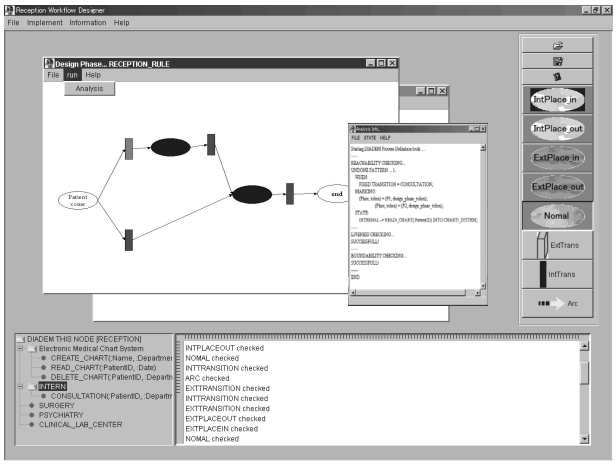

図 4　ユーザインターフェースのスクリーンショット

外部インターフェース　設計フェーズにおけるプロセス定義の解析時の情報の受け渡しやモニタリング時の情報の受け渡しを他ノードと行う．

プロセス定義解析システム　設計フェーズにおいて，ユーザの定義したカラーペトリネットの Occurrence Graph を作成し，可達性，活性，有界性を調べる．設計フェーズ

における解析は稼動中の DIADEM とは独立に動いており，可達性，活性，有界性を全て満たすプロセス定義のみルール変換器を通して DIADEM のルールとして実装し，稼働させることができる．

ルール解析システム　ルールモニタからの情報とペトリネットとを照らし合わせて，ユーザへ通知する障害情報を生成し，ユーザインターフェースへ伝える．また，他ノードへの障害情報伝達の必要があるならば外部インターフェースを通じて伝達する．

DIADEM マネージャ　DIADEM のトランザクションの管理を行う．

ログマネージャ　ノードの動作をロギングし，障害発生時にはログを用いて DIADEM の復旧を行う．常にルールモニタに監視され，復旧不可能な障害が起こった場合，ルールモニタを通じてユーザへ伝達される．ログは JDBC を通じて DBMS に格納される．

ルールモニタ　ログマネージャの監視を行い，障害情報を得たら，それをユーザへ伝達する．

DBMS　SQL インターフェースを前提とする他はストアドプロシージャなどの機能は必要としない．データベースには，各部局のデータやログ，外部データベースの情報等が格納される．

7. 実験システムによるプロセス定義の設計とその解析

7.1 想定するワークフロー

実装した実験システムを用いて，病院における外来患者のワークフローを例に挙げ，可達性，活性，有界性判定の解析を行う．

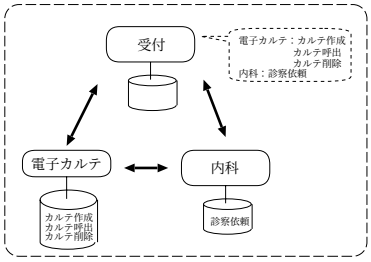

図 5　想定する病院内情報システム

図 5 は，想定する病院内情報システムの例である．内科，電子カルテシステムにおいて公開されている外部トランジションは以下の通りである．

内科

CONSULTATION(:PatientID, :Department, :Date)　患者の診察を依頼する．

電子カルテシステム

CREATE_chart(:Name, :Number, :Date)　カルテ作成．

READ_chart(:PatientID)　カルテ呼出．

DELETE_chart(:PatientID)　カルテ削除．

この条件下において，受付における外来患者のワークフローを設計する．

7.2 可達性を満たさない例

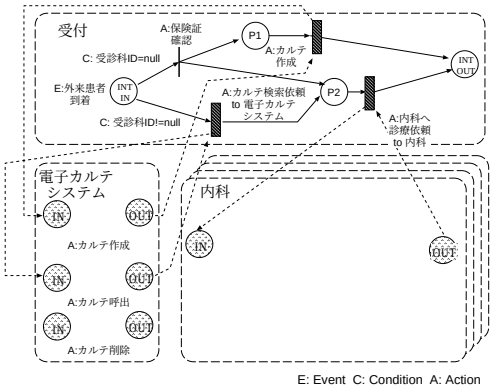

図 6　可達性を満たさない受付におけるプロセス定義

図 6 は，電子カルテシステムと内科において公開されている外部トランジションを用いて，受付におけるワークフローを記述したものである．受付から見えるのは，電子カルテにおける「カルテ作成」「カルテ呼出」「カルテ削除」の 3 つの外部イベントの仕様と，内科における「診察依頼」という外部イベントの仕様である．しかし，内部の挙動については公開されないので，例えば内科における電子カルテの呼出がどのタイミングか (診察依頼した直後か，あるいは診察する直前かなど) を受付において知ることはできない．

受付におけるワークフローの流れとしては，外来患者の受付によりワークフローが生起され，受診科 ID を持つ，即ち再診患者ならば電子カルテシステムからカルテを呼び出し，特記項を確認，問題がなければ，内科へ診察依頼の処理委譲を行う．受診科 ID を持たない，即ち初診患者ならば保険証の確認を行い，そのデータを元に，電子カルテシステムにおいてカルテ作成の処理委譲を行う．時間短縮のため，カルテ作成の処理委譲と並行して内科への診察依頼の処理委譲を行うというワークフローの記述になっている．

プロセス定義解析システムは，可達性，活性，有界性判定を行い，図 7 のような解析結果を出力する．

これは，プレース P1 および P2 に解析用トークンが入っている状態から，トランジション「診察依頼 (CONSULTATION)」が発火しようとした際にアクションが完遂できずに遷移が停止するパスがあることを表している．原因を特定する情報として STATE:節がある．STATE:節には，内科においてカルテシステムの外部トランジション「カルテ呼出 (READ_chart)」が発火できなかったという記述がある．この結果と設計したプロセス定義を鑑みて，「カルテ作成 (CREATE_chart)」よりも「カルテ呼出 (READ_chart)」を発火しようとしたら発火できなかった，即ちカルテを作る前

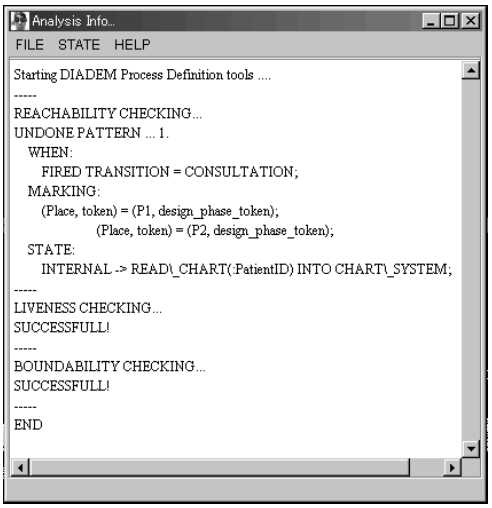

図 7　図 6 の解析結果

にそのカルテを呼び出そうとしたという実行順序関係の設計ミスを推測できる．

ワークフローは BPR を実現する手段であるため，ユーザはより効率の良いプロセス定義を設計しようとする．しかしそうするあまり，アクションの実行順序関係を誤るというミスを犯しやすい．図 6 の受付内のみの可達性判定は成功するが，階層化された部分を視野に入れると図 8 のような構造のペトリネットを解析することになり，結果的に可達性を満たさないことになる．

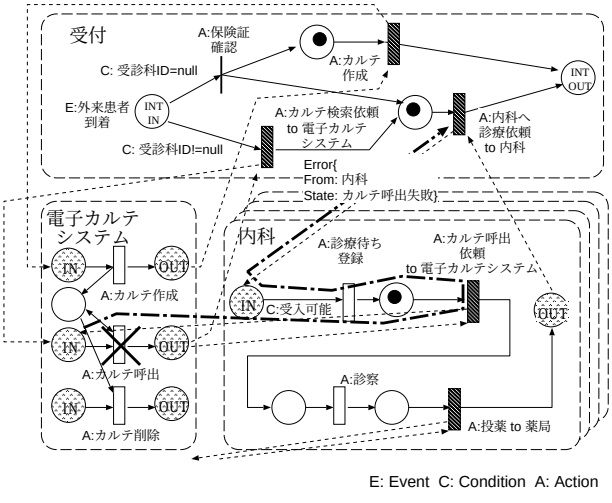

図 8　図 6 におけるワークフローの全体像

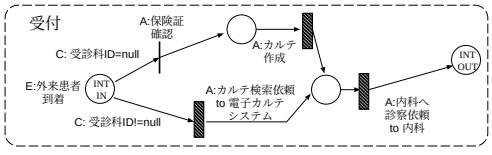

図 9　可達性を満たす受付におけるプロセス定義

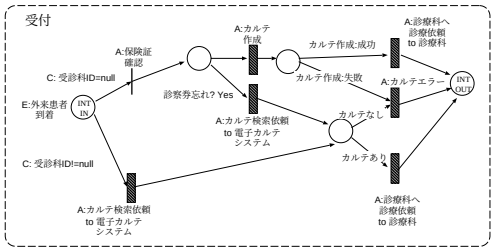

図 10　活性を満たさない受付におけるプロセス定義

7.3　活性を満たさない例

不活性なトランジションの存在は可達性判定により検出できるケースもあるが，可達性を満たしていても活性を満たさないケースもあるため，活性判定も重要である．

図 10 は可達性は満たすが，活性は満たさない例である．図 10 の右の真中にあるトランジション「A:カルテエラー」が不活性である．ユーザは,「カルテ作成に失敗したかあるいは電子カルテシステム内にカルテがない」などの理由から「患者のカルテを用意することができなかった」という旨を伝えるアクションを記述したつもりだが，カラーペトリネットの構造上，カルテエラーのトランジションが発火可能になることはない．これはユーザが「「A:カルテエラー」のガード条件として,(「カルテ作成失敗」or「カルテなし」)とするべきの所を（「カルテ作成失敗」and「カルテなし」）という記述をしてしまったことに起因する．

解決策としては,「カルテ作成失敗」と「カルテなし」の場合のそれぞれのエラーアクションを行うトランジションを用意すればよい．

7.4　有界性を満たさない例

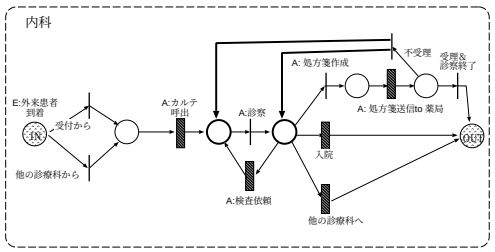

図 11　有界性を満たさない内科におけるプロセス定義

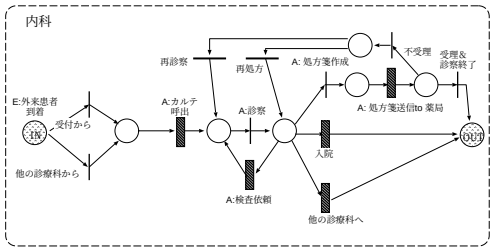

図 12　有界性を満たす内科におけるプロセス定義

最後に有界性を満たさないプロセス定義の例を挙げる．

図 11 は内科における診察のプロセス定義である．これは

有界性を満たさず，更に無限ループを構成する可能性があることから，可達性も満たさない．これは，1 つのトークンが無限のトークンを生成してしまう可能性を持つ構造になっている．ユーザは，薬局に対する処方箋送信の後，薬局から受け取ったイベントが不受理であった場合，処方箋を作成し直したり入院を勧めたりするか，あるいは医者によって再び診察することもあると考え，図中の太字の部分のような記述をした．しかし，これでは同じ患者が 2 倍に増えてしまうことになる．

解決策として，処方箋を薬局側によって受理されなかったという事象の後であることから，図 12 のように再診察と再処方というルールを設けて分岐させる方法がある．

8. 関 連 研 究

ワークフローのモデリングにペトリネットを適用した研究事例としては，ワークフローネット [10] [11] [12] がある．これは，1 つのトランジションを 1 つのアクティビティに対応させ，以下の 2 つの条件を満たすペトリネットを用いてモデリングする手法を提案している．

（1） 特殊な 2 つのプレース (入力トランジションを持たない入力プレース，出力トランジションを持たない出力プレース) を持つ．
（2） 全てのトランジションとプレースは，入力プレースと出力プレースへのパス上に存在する．

しかしこの制約は，分岐した処理は必ず合流しなければならないプロセス定義を強要するため，分岐した処理がそれぞれで単独終了するような設計には対応できない．また分散ワークフローやトランザクショナルワークフローは考慮されていない．

ワークフローネットをトランザクショナルワークフローの仕様を満たすよう変更する方式を提案している研究事例もある [13] が，分散ワークフローについては深く言及されていない．

本稿では，DIADEM というアクティブデータベースを用いた分散トランザクショナルワークフローのメカニズムに対して，そのプロセスモデリングにカラーペトリネットを適用することを提案している．他ノードとの分散処理にカラーペトリネットの階層化の概念を用い，設計したプロセス定義が満たすべき性質を定めそれを解析することを提案しているが，このような研究事例は他に見当たらない．

9. まとめと今後の課題

本稿では，DIADEM におけるワークフロープロセスモデリングにカラーペトリネットの適用を提案した．提案した手法では，特殊なトランジションやプレースを導入し，他ノードでの処理は外部トランジションとして階層構造にしてモデリングを行う．設計したプロセス定義が，可達性，活性，有界性を満たすか解析することで設計ミスを検出する手順を紹介した．また提案したカラーペトリネットは，モニタリングフェーズにおいても適用可能であることを述べた．実験システムを JAVA を用いて構築，設計フェーズについて実際のワークフローを設計し解析，出力結果が設計ミスの原因の絞込みに有用であることを述べた．このような手法は，SOAP を用いた BTP($BusinessTransactionProtocol$) などの Web サービス上でのワークフローにも適用可能である．

今後は，ルールモニタの実装を行い，モニタリングフェーズにおける障害検出時の情報伝達の実験を行う必要がある．また性能解析フェーズについても検討する必要がある．解析すべき要素はワークフローの種類や環境に強く依存するため，全てのワークフローに適する評価指標を示すことは難しい．しかし，時間やコスト面を重視するワークフロー事例は多い．提案したカラーペトリネットにより時間やコスト面の評価を行う検討も必要になる．

謝　　辞

本研究の一部は，文部科学省科学研究費補助金基盤研究 (14019035) の助成により行われた．

文　　献

[1] The Workflow Management Coalition. http://www.wfmc.org.
[2] M. Rusinkiewicz and A. Sheth. Specification and Execution of Transactional Workflows. In *W. Kim, editor, Modern Database Systems: The Object Model, Interoperability and Beyond*. ACM Press, 1994.
[3] J. Widom and S. Ceri (ed.). *Active Database Systems: Triggers and Rules for Advanced Database Processing*. Morgan Kaufmann Pub, 1996.
[4] 井川 智崇, 宮崎 純, and 横田 治夫. 独立アクティブデータベース間の入れ子トランザクションの実現. In **情報処理学会研究会報告**. 第 10 回データ工学ワークショップ, March 1999.
[5] 井川 智崇 and 横田 治夫. 分散独立アクティブデータベース上でのワークフロー用トランザクションモデル. In **信学技法** *DE2000–86*, pages 119–126. 電子情報通信学会, 2000.
[6] 井川 智崇 and 横田 治夫. 分散独立アクティブデータベースによる障害ハンドリング. In **第** *11* **回データ工学ワークショップ論文集**, *DEWS2001 3A–5*. 電子情報通信学会データ工学研究専門委員会, 2001.
[7] K. Jensen. *Formal definition of coloured Petri nets*. PhD thesis, 1992.
[8] 小林 昌徳 and 横田 治夫. 分散独立アクティブＤＢによるワークフロー管理のためのフロー解析手法. In **信学技法** *DE2002–13*. 電子情報通信学会, 2002.
[9] Dennis R. McCarthy and Umeshwar Dayal. The Architecture of an Active Data Base Management System. In *Proc. of SIGMOD Conf. '89*, pages 215–224, 1989.
[10] W.M.P. van der Aalst, K.M. van Hee, and G.J. Houben. Modelling and analysing workflow using a petri-net based approach. 1998.
[11] W.M.P. van der Aalst and A.H.M. ter Hofstede. Verification of workflow task structures: A petri-net-based approach. 1998.
[12] W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. pages 8(1):21–66. Systems and Computers, 1998.
[13] Wijnand Derks, Juliane Dehnert, Paul Grefen, and Willen Jonker. Customized atomicity specification for transactional workflows. In *CODAS*, pages 155–164, 2001.