解　説

IPS

ネット指向パラダイムを求めて

3. ネット指向ソフトウェア構築ツールについて†

寺　内　睦　博††

1. はじめに

ソフトウェア工学では，その生産効率性及び理論的な立場から開発工程をサポートする環境に関する研究が行われ，数学的な基盤をもつ概念が展開されつつある．一方，ネット理論の研究においては，理論的探究の成果を応用面に活かしたいという強い欲求があるようである．近年，両者の研究領域は相互の分野へ活路を見いだし，それぞれの成果を参考に積極的に融合しようとする動きがある[1]．

本稿では，ペトリネットユーザという立場から，最近提案されたいくつかのネットワークツールの概要を紹介するとともに，そのしくみを示すために Design/CPN という一つのツールを取り上げ，その構成とソフトウェア生産の適用可能性について述べる．

2. CASE について

ソフトウェア生産の現場では，近年の対象とするシステムの大規模化，および複雑化の要求に対してさまざまな努力がなされてきている．このうち，ソフトウェアの生産効率や品質向上を目的として，計算機上の各種の支援ツールを用いてこれらの改善を行うアプローチがある．これは CASE と呼ばれ，コーディングの支援からシステム全体の仕様記述のサポートまで各レベルに応じた環境が提案されている．本章では，これらの概要についてツール群に求められる望ましい条件という観点から簡単にまとめる．

構造化プログラミングやソフトウェア工学技法が，主にプログラム構造，開発工程，開発支援ツールに強い影響を与えているが，これらコードレベル，モジュールレベル，システムレベルの三つに共通するのは構造分析概念である[2]．システム設計はまず仕様設計を行い，それらをしだいに詳細化して実際のコーディングが行われる．その方法論として，機能分解法，データフロー設計法，データ構造設計法がよく利用される．これらの技法は対象とするシステムを階層的に表現するとともに，モジュール間の関係を因果的に，かつ仕様との整合性を保ちながら記述し，構築しようとするものである．プログラム構造を取り扱っている技法は，ツールやプロセスに関するものの基盤となっている．もちろん支援ツールや開発工程はプログラム構造に強く影響を及ぼすが，必要なプログラム構造に適応する技法を選択する必要がある．

JSD 法はマイケルジャクソンの提案したシステム開発手法であり，従来の構造的手法におけるウォーターフォールモデルによるソフトウェア開発の問題点を考慮した上で，異なる視点からそれを解決した開発手法である[3]．この手法では，モデルを記述するためにデータフロー図に類似したシステム仕様図と，逐次型プロセスの非決定的事象遷移図であるジャクソン構造図を用いる．このように JSD 法では状態遷移を考える代わりに，それの因子である事象に着目し，状態を変化させる事象（イベント）の遷移関係をグラフ表現し，対象の構造としている．

以上のような観点から，対象をグラフ表現し，システムをネットワークとして捉えることにより，オブジェクトを視覚化した環境であるネットワーク型のシステム構築手法が有効であると推察できる．適度な階層化（抽象化）により，システム全体の見通しが良くなるとともに，階層間での仕様記述についてもオブジェクト指向の思考法が自然に表現できると考えられる．

† On Net-Oriented Software Tools by Mutsuhiro TERAUCHI (Department of Circuits and Systems, Faculty of Engineering, Hiroshima University).
†† 広島大学工学部第二類

3. ネット指向ソフトウェア構築ツール

前章までの議論から考えると，今後の CASE ツールではトップダウン設計技法をサポートし，データの流れやモジュールの関連を図式的に表現でき，さらに数学的な検証方法が確立されているモデルをもつものが必要とされていることが分かる．本章では，最近発表されたネットベースツールの概要について紹介する．

1) **HOOD Nets:** Hierarchical Object Oriented Design Nets[4]

HOOD（階層オブジェクト指向設計）とはソフトウェアライフサイクルのアーキテクチャ設計フェーズのための手法である．これはAdaプログラム開発のためのツールで，抽象マシンとオブジェクト指向設計概念を結合させたものである．オブジェクトに組み込まれる情報の多くは Ada あるいは疑似 Ada コードで記述されている．HOOD ではコードの再利用を薦めているが，疑似コードの場合，オブジェクトの特性の形式的検証はできないので，形式的仕様技法を導入している．この技法としてペトリネットのサブクラスであるステートマシンが採用されている．このように制御構造としてステートマシンを導入していることから，システムが複雑になるとネットが巨大になるという欠点があるが，ステートマシンを一般あるいはハイレベルペトリネットへ変換することによりコンパクト化し，見通しをよくすることができる．

2) **Great SPN:** GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets[5]

Great SPN はファイル共有によるペトリネットモデルのエディタ，及びいくつかの解析ツールから構成されている．このツールの代表的な機能を以下に示す．

ウィンドウ上でネットの編集，階層化，図形的操作を行う図形エディタでは，構造の表示，トークンゲーム，サブモデルの付加，モデルデータベースへのアクセス，タイミングや確率変数の仕様定義，メニュを用いたネット解析，実行結果の図式表示，時間及び確率的ペトリネットのシミュレーションなどのさまざまな処理が行える．

また，解析ツールを用いて，Pインバリアント，Tインバリアント，デッドロック，トラップ，相互排他，構造的競合，構造的バウンドなど多くのペトリネットの構造的な特徴を解析できるとともに，可到達グラフの生成及び解析を行うプログラムが提供される．これにより，ホームスペース，デッドロック，ライブロック，トランジッションライブネスの解析を行うことができる．さらに，シミュレーションモジュールによりインタラクティブなイベント駆動シミュレーションが行える．また任意のイベントの再スケジューリングや周期的サンプリングをともなうバッチ的シミュレーション機能をも提供する．

3) **Topnet** a TOol based on Petri NETs for the Simulation of Communication Networks[6]

Topnet は通信ネットワークとそのプロトコルのシミュレーションモデルの開発及び実行を支援するネットツールである．このツールの意図するものは，最大限可能なシミュレーション実験の実現とモデル記述の再利用にある．これによりライブラリから持ち出した多くのサブモデルを結合することにより大規模なシステムを構築することが可能となる．Topnet ではペトリネットに時間とデータ操作機能を付加した PROT nets (Process Translatable Petri Nets) を内部記述として採用している．ネットのグラフィック表現とともに，スクリプトと呼ばれるAdaコードを内包し，トランジション発火時にそのアクションが実行される．Topnet のグラフィックインタフェースの特徴として，オブジェクト指向アプローチに準じたトップダウン開発に適した階層的グラフィックエディタがあげられる．Topnet は Ada コードに変換可能であり，これを図式的に実行させることもできる．実行時のアニメーション機能と同時に，測定された性能を表示可能である．

4) **Voltaire** a Discrete Event Simulator[7]

Voltaire はシミュレータと階層ペトリネット記述言語からなる時間付きペトリネットをベースとした離散事象シミュレータである．Voltaire は巨大で複雑なシステムを比較的速く効率的にモデル化し，シミュレートする機能を提供する．このシステムの特徴は複雑なペトリネットのネット要素をブロックとして構造化し，それ自身を階層化し，高レベルブロックとして組織化できる点である．これによりトップダウンモデル化技法を設計者に提供できる．ペトリネットを導入することにより広範なシステムクラスをモデル化でき，さら

に同期・並行事象を簡単にモデル化できるのみならず，それらの依存関係を求めることもできる．トランジションのアクションの記述はC++言語で書かれ，トークンには任意の属性を設定でき，大規模システムの特徴を集中管理する大域メモリとリソース制御を行うセマフォを有する．このツールの特徴であるブロック化はオブジェクト指向でのカプセル化原理に準じ，アクション記述言語のC++とも相まってトップダウン思考の方法論を支援する．もちろんボトムアップ手法による開発も可能である．

5) A Software Environment for the Generator, Representation and Analysis of PN based Models of Manufacturing Systems[8]

このソフトウェア環境はペトリネット表現を通してマニュファクチャリングシステム（以下 MS と略記）をモデル化し，解析することを目的としている．

ペトリネットモデルの自動生成と表示：要素集合とそれらの接続関係が与えられると，対応するネットモデルが生成される．さらに，その表示のために，平面性の判定を用いて枝の交差を最小にする描画が与えられる．**調整エディタ**：このエディタは，プレースの名前やマーキング属性，トランジションの名前，発火時間などの属性を設定する．さらに競合解消戦略，データ依存型の発火時間をもつトランジションも設定できる．**解析・シミュレーションツール**：このモジュールはネットのシミュレーションを行い，次の三つのイベントにより動作を停止することができる．1)サイクルの長さ，2)トランジションの発火回数，3)プレースのトークン数．またシミュレーション中の各プレースのトークンの分布とトークンの平均数が結果として出力される．このソフトウェア環境はCとC++でインプリメントされている．

6) **FORSEE** a collection of tools for FORmal System Engineering Environment[9]

FORSEE は現在，TORAS：可到達性解析ツール，PROMPT：自動インプリメンテーションツール，Design/CPN：グラフィカルエディタ・シミュレータの三つの要素から構成されている．現在，数学的技法としてペトリネットを採用している．これはペトリネットが並行性を含んだ形で定式化されており，実行可能であり，多くの解析手法を備えた数学的な基盤をもっているからである．FORSEE において，TORAS は検証ツールとして，PROMPT は形式的仕様をC言語へ変換するツールとして，グラフィカルエディタ・シミュレータである Design/CPN を FORSEE のフロントエンドとして位置付けている．以下にこれらのツールの概要を述べる．

TORAS：TORAS はペトリネットの可到達性の解析を行う．これは初期状態から到達可能なすべての状態と状態遷移を生成する．この際，状態数の爆発が起きるが，数種のアルゴリズムを利用して，デッドロックやライブロックなどの特性を失うことなく状態空間を小さく抑えている．100のプロセスをもつ簡単な事例ではすべての状態空間は 10^{47} であるが，デッドロックを保存しながら，30,000 に簡約化した例もある．**PROMPT**：PROMPT は通信ソフトウェアの開発環境を提供する．これにはコンパイラ，デバッガ，制御インタフェース，ログファイルアナライザという4つの要素がある．コンパイラは拡張ネット言語 (XNL) を入力とし，C言語コードへ変換する．デバッガは XNL レベルで動作する．制御インタフェースはいくつかの XNL 仕様を同時実行させ，ユーザの事象制御のログを取る．**Design/CPN**：Design/CPN はカラーペトリネットの編集とシミュレーションを行うパッケージソフトウェアである（詳しくは次章を参照）．

7) **UltraSAN** Stochastic Activity Networks[10]

UltraSAN は各種エディタ，シミュレータ，ソルバから構成される．

エディタには三種類のものがあり，ペトリネットのグラフィックエディタと階層性あるいは subnet 間の関係などの構成モデル (subnet) の構造を定義するエディタ，そして，ネット上の変数を定義するエディタである．

Reduced Base Model Constructor：ネットの理論的解析のために，そのシステムの内部表現からモデルの定常表現が構築される．この表現には状態集合，状態間の推移率，及び各状態の変数が含まれる．これを初期状態として，subnet 及びそれらの関係を用いて状態を遷移させ，これにより各パラメータが求まる．その後，LU 分解，繰り返し緩和法，マルコフ過程近似などの手法を用いてネットを解析する．　**Simulation Solver**

(Simulator)：ネットの実験的解析には，ネットは実際に Simulator により実行される．

8) **PROOFS** PROmotion Of Formal Methods in European Software Industry[11]

PROOFS プロジェクトの目的は形式的手法が分散した産業用アプリケーション開発に適していることを示すことである．

形式的手法：ネットワークの形式論には時間付き色付きペトリネットが採用されている．状態空間のモデリングには NIAM の中のモデルと同様の2値モデルと機能データモデルが，操作の記述には宣言的言語と集合論的仕様記述言語 (Z, VDM) が採用されている．形式論が組み合わされると，これらの形式論に基づく手法をリンクすることができる．

9) **AMI**[22),23)]

AMI は属性をもつ型付きグラフに基づくインタラクティブなソフトウェア環境である．新しい種類のグラフやペトリネットにも容易に拡張できる．AMI ソフトウェア環境はグラフの生成，操作，変換，解析及びチェックのためのインタラクティブなツールキットからなる．グラフィックエディタ MACAO は多重ネット入力を許容する．推論エンジン MIAMI は制約に基づき，エキスパートシステムルールの中に論理プログラミングを導入するために統合されている．AMI は新しいアプリケーションの簡単な統合を実現する開放型環境である．現在，AMI は多くのペトリネットツールとともに動作する多くのアプリケーションを統合している．

その他のツール

上記のネットツール以外にも多くのツールが提案され，または開発途上にある．これらについては文献 22)，23)，および 24)，25) を参照されたい．また，これらのうちのいくつかはインタネットの ftp サーバで公開されている．その代表的なものを以下に示す．

GSPN ftp from bikini.cis.ufl.edu in pub/sim-digest directory

LOOPN ftp from ftp.utas.edu.au as departments/computer_science/loopn.tar.Z

また，Great SPN, PROD については以下に連絡先を記す．

Great SPN : chiola@di.unito.it

PROD : Kimmo.Varpaaniemi@hut.fi

4. ネットツールのしくみ—Design/CPN を例にして—

本章では，Meta Software 社の Design/CPN という開発ツールの概要について述べる．以下にこのソフトウェア・ツールの簡単な紹介をする．

前章の各ツールの特徴からも分かるように，ソフトウェア・ツールに必要な枠組みは 1)図式的なエディタを備えること，2)ソフトウェア構造をモデル化し，数学的に解析あるいはシミュレーションを実行できること，3)モジュールのオブジェクト指向化や再利用を促進する階層性を有することがあげられる．

このような観点からペトリネットをモデル化言語として採用することは有効であると考えられる．しかし，ペトリネットを実システムに応用する際に大きな障害となるものの一つにネットの大きさの問題があげられる．これは対象とするシステムが小さなものであってもこれに相当するペトリネットを記述すると途端にネットの規模が大きくなるという問題である．このモデルサイズの大きさの問題はペトリネットに限らず，すべての種類のシステムモデルで遭遇するものである．

さて，システムのモデル化と設計の分野では単一レベルのシステムモデルでは，見通しの悪さ，詳細記述の繁雑さなど，多くの問題点があることが知られている．これらの問題を克服するために階層的モデル化言語が導入され，実際の現場でよく使われるようになった．そのほかにも SADT や IDEF があり，このような階層的モデル化言語は，(1)詳細な記述の隠蔽，(2)モジュール分割，(3)ソフトウェアの再利用可能性，(4)トップダウン及びボトムアップの両開発戦略，(5)強力なグラフィック表現能力などの長所をもつ．さらに，モデル化言語は次のようなことがらを満足しなければならない．

- 正確で一貫性のある要素に対する行動概念のサポート
- 異なったレベルにおける巨大で複雑なシステムモデルの実行を観測可能なこと

上述のグループのモデル化言語の多くはこれらの実行可能性という能力をもっていない．これに対して，最近のハイレベル・ペトリネットは並行動

作を表現する優れたモデル化言語として受け入れられつつある．その理由の一つはハイレベル・ペトリネットの数学的な基盤にある[14)~17)]．

4.1 ネットツールの構成要素

4.1.1 基本モデル化言語—色付きペトリネット—

今日，ペトリネットの最も現実的な応用はハイレベル・ペトリネットを用いたものである．ハイレベル・ペトリネットとして，従来のモノクロのトークンを色付きに拡張したもの（命題論理から述語論理への拡張）やトランジションの発火に確率的な因子を導入したものなどが提案されている．ハイレベル・ペトリネットと普通のペトリネットとの関係は高級プログラミング言語とアセンブリ言語との関係に例えることができる．これらの中でも色付きペトリネットと述語トランジションネットは有力なハイレベル・ペトリネットである．

色付きペトリネットは複雑なシステムを簡潔に表現するために，数学的表現能力とグラフィックモデルの明快さを合わせもつ．ある任意の色付きペトリネットの挙動は普通のペトリネットと等価であるが，このとき後者は述語はもたないものの，非常に多くのグラフィックオブジェクトをもつことになる．色付きペトリネットの定型的述語はカラーセット，マーキング，アーク表現，ガードからなる．色付きペトリネットの簡単な紹介は文献 15) に，詳細な定義については文献 14) を参照されたい．

Design/CPN には階層色付きペトリネットのインタラクティブな編集をするためのエディタが付属する．つまり色付きペトリネットは一つのサブネット（もちろんこれがさらに下位のサブネットを含んでもよい）を含んでもよい．またこれらは形式的に相互作用する．このサブネットはページと呼ばれ，代入，呼出，融合という三種類の異なる関係をもつ．これらの関係は有効な構造化手法を構成し，モジュール的で管理の容易な大規模な色付きペトリネットを構築することができる．シミュレーション，可達グラフ，プレース・インバリアントの三種類の CPN 解析手法を拡張することにより，それらは階層的な CPN へも適用可能である[18)]．

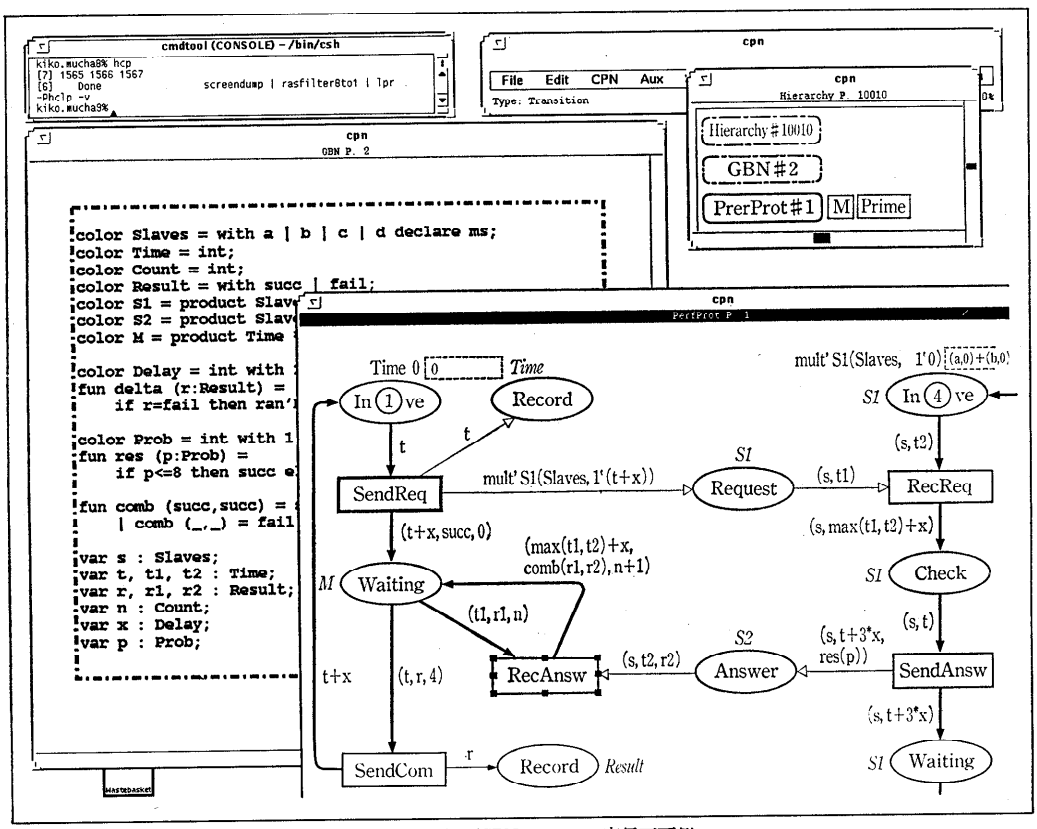

図-1　Design/CPN ツールの実行画面例

4.1.2 Standard ML[19),20)]

Design/CPN の評価エンジン (CPN/ML) はエジンバラ大学で開発された関数型プログラミング言語である Standard ML (SML) コンパイラ (ESML) を基盤としている．これは，それがデータ型をもつ関数言語で，かつインタプリタであることに基づいている．

4.1.3 IDEF

Design/CPN には IDEF の図を作成し，それらの図を同等の CP-net に変換するプリプロセッサ機能がある．IDEF は階層的システム記述手法として広く使われており，SADT という名前でも知られている[12)]．

IDEF/CPN を用いてユーザは，実行可能な挙動モデルを定義することができる．これと IDEF との違いは枝表現に対するさらに精密な文法を設定していることである．これにより，IDEF のアクティビティと CP-net のトランジション，IDEF の枝と CP-net のプレース，IDEF の枝表現と CP-net のアーク表現，および IDEF の階層と CP-net の階層の間には，ほぼ 1 対 1 の対応関係がある．

4.2 Design/CPN の実行

Design/CPN には二つのモードが存在し，それを使い分けることができる．しかし，これらは同一プログラムであるため，データの受渡しなどは不要である．

4.2.1 編集モード

Design/CPN を使うことによりユーザは複雑な階層的色付きペトリネットを構築し，編集することができる（図-1）．

Design/CPN のエディタは CP-net に含まれる異なる種類のオブジェクト（プレース，トランジション，アーク，初期マーキング，ガード，アーク表現など）のおのおのについてデフォルト属性（サイズ，形，濃淡，フォント，色など）をもち，ユーザはこれらを変更することができる．また Design/CPN のエディタは非常に多くの文法チェック機能をもっている．これらの機能はプログラミング言語での型チェックに相当する．つまりシステムにより潜在的に安全でない構造が検出された場合，それをユーザに教えてくれるのである．

ユーザは新しいページを生成することによりトップダウン階層を構築でき，現存するサブネットを新しいページに移動することによりボトムアップ階層を構築できる．

4.2.2 シミュレーションモード

Design/CPN のシミュレーションでは，階層的 CP-net のおのおののページはそれ自身のウィンドウに表示される．発火可能なトランジションはシステムによりハイライト表示され，シミュレーションステップの発火がアニメートされる．これによりユーザはトランジションへ向けて入力枝をトークンが移動したり，トークンが出力されるのを見ることができる．

シミュレーションには自動とマニュアルの二つのモードがあり，マニュアルモードではユーザがシミュレーションステップ全体にわたって詳細に制御することができる．しかし，この場合でもシステムから多くの手助けを得ることができる．たとえば，ユーザは Design/CPN に対して，あるトランジションについてバインドし得るすべての値を計算させることができる．このようなシステムの機能により，ユーザはステップ計算に労力を費やすことなしに，シミュレーションの理解に集中することができる．

4.3 プログラム構造解析—Design/CPN Palette によるネット解析—

シミュレーションは，システム設計の挙動特性を解析したり，プロトタイプ作成やインプリメンテーション，及び性能評価の支援に必要とされる有効なツールである．しかし，あるシステムが論理的に正常に動作するかどうか，すなわち，仕様に適合しているかを検証するためには，さらに形式的な解析方法を必要とする場合が多い．CP-net を採用する重要なポイントは，モデル化されたシステムの特性を形式的に証明する形式的解析方法を提供できる点にある．すなわち，モデル化されたシステムの特性を数学的に検証できるような開発環境となりうるか否かということが重要となる．

形式的な解析方法には，次の 4 つの手法が提供される．1)可達グラフの構築（到達可能なすべてのマーキングを表現するグラフ），2)線形システムインバリアントの計算と解釈（P（プレース）インバリアントおよびT（トランジション）インバリアント），3)構造的特性の検証（挙動特性），4)リダクション（ある選択された特性集合を変化させずにネットを縮約する），ここでは可到達グラフ解

析とPインバリアント解析について説明する.

4.3.1 可到達グラフ

Design/CPN Palette では，ユーザが CP-net に対する可到達グラフを構築したり，解析したりすることができる．この構築と解析はともにほぼ自動的に行われる.

この方法では，ユーザはおのおののカラーセットに対して可能な順列の集合を定義することができる．順列集合には，その順列が代数群を形成し，この制約が CP-net のマーキングにおいてその順列が同等な関係を導くという十分な保証が得られなくてはならない．通常の可達グラフは CP-net の到達可能なマーキングのおのおのに一つのノードを割り当ててある．同クラスの可達グラフ (OE-graphs) はおのおのの到達可能な同等のクラスに対して一つのノードしか割り当てていない．可到達グラフを再構成する場合には OE-graph のもつノードとアークの名前集合を使って対応を取ることができる.

おのおのの OE-graph には，対応する可到達グラフと同じ情報が正確に保持されているので，可到達グラフと同様な方法で，デッドロック，マーキングの最大バウンド，基本状態，活性，到達可能性などのシステム特性を検証することができる．これらのシステムの諸特性は OE-graph を用いた解析によりすべてのバウンドされた CP-net に対して決定可能である．さらに，その解析は自動的に実行可能である.

Design/CPN Palette における OE-graph の自動解析は標準的なグラフアルゴリズムを提供する．上述のすべての特徴は，強連結成分 (SCC graph) の計算と単純な探索 (たとえば，最大値探索など) で実現される[21].

4.3.2 Pインバリアント

Design/CPN Palette では，ユーザは構築された CP-nets に対するPインバリアントを見つけることができ，それらのPインバリアントにより，プログラム検証に用いるインバリアントの利用と同様に，CP-nets の特性解析ができる．CP-net のPインバリアントは個々のプレースに付加された重みの集合である.

CP-nets およびほかの種類のハイレベル・ペトリネットに対しては，すべてのPインバリアントを自動的に検出することは可能である．(これはいろいろな方法で実現可能である．たとえば，ハイレベル・ネットをプレースとトランジションのネットに展開して，ガウス消去法を適用する，など) しかしながら，そのような基底を見つけることは，あまり有用でないことがある．その理由は，基底から所望するPインバリアントをいかに得るかを示す経験的，理論的なガイドラインが存在しないからである．こういう理由ですべてのPインバリアントの基底の計算は行わないという仕様となっている[21].

4.4 トランジションから実行コードへの変換

階層的 CP-net の中のおのおののトランジションには CPN ML(将来的にはほかの言語 C, Pascal などでも可能となる) で書かれたコードセグメントを付加することができる．これにより，システムのシーケンシャルな部分は Standard ML で記述し，制御構造は CP-net でプログラムするソフトウェアを生成することができる．そのプログラムはコードセグメント内の入出力コマンドにより，ほかのプロセスと通信することもできる.

プログラムを解析し，シミュレータ及びそのほかの解析ツールによりデバッグした後に，それらは実行可能な ML コードに変換できるようになる．ML コードには，コードセグメント及び図式的な情報をもたない CP-net の概略表現が含まれている．さらに ML コードは簡単なシミュレータをもち，その CP-net を実行し，発火トランジションに当たるコードセグメントを起動する.

5. ま と め

以上，いくつかのネットワークツールの概要とそのうちの一つである Design/CPN の基本仕様について述べた．これらのソフトウェアはすべてペトリネットをモデルとして採用していることから，並列分散システムのシミュレーション，プロトコル記述に適したものであり，ネットワークの構造解析が可能なシステムである．つまり，これらの各種のツールは対象モデルの解析能力を十分に備えているといえる．しかし，これらはペトリネットという特別な言語の知識を必要とし，ソフトウェアの生産現場で素直に受け入れられるとは言い難い．けれども，その応用分野はまだ未開発であり，図式的なもの以上に言語的なインタフェースを整備すれば，ソフトウェアのみならず，

さまざまな可能性を秘めていると推測される．今後さらに，これらの可能性を拡大する努力が必要であると思われる．

謝辞 本稿の執筆を勧めていただいた，大蒔和仁(電総研)，翁長健治(琉球大)，両氏に感謝いたします．また，数々の討論および情報を提供していただいた，渡辺敏正(広島大)，辻孝吉(福井大)，本位田真一(東芝)，Hans Fuss (GED), Alex Blakemore (UMD), Dick de Reus, Paul Rambags (Eindhoven Univ. of Tech.), Philippe Palanque (Univ. Toulouse), Takashi Kobayashi (Hitachi, Ltd.), Christoph Lindemann (TU-Berlin), Wilfried Pfister (Uni. de Grenoble), Mike Koopman (Concurrent Tech. Corp.), Charles Lakos (Univ. of Tasmania), Pierre Azema (LAAS-CNRS), Fabrice Kordon (Univ. P. et M. Curie-CNRS), Peter Starke (Humboldt Univ. zu Berlin), Wolfgang Reisig (Technischen Univ.) の各氏に感謝の意を表します．また本稿の内容の一部は文部省科学研究費重点領域研究「自律分散システム」No. 04218110の補助を受けた．併せて感謝いたします．

参考文献

1) 翁長健治：情報システム構築の新しいパラダイムを求めて，ネット指向ソフトウェア設計技術に関するチュートリアル講演論文集，pp. 1-11 (1992).
2) Bergland, G. D.〔妹尾 稔(訳)〕：構造化設計法，ソフトウェア・ツール (bit 増刊)，Vol. 14, No. 3, pp. 221-246, 共立出版，東京 (1982).
3) 有澤 誠他：マイケルジャクソンのシステム開発法，bit, Vol. 22, No. 1, 共立出版，東京 (1990).
4) Di Giovanni, R.: Petri Nets and Software Engineering: HOOD Nets, Proc. of 11 th ICATPN, pp. 123-138 (1990).
5) Chiola, G.: GreatSPN 1.5: GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets, Tool Description, PNPM '91 (1991).
6) Ajmone Marsan, M. et al.: TOPNET: A Tool Based on Petri Nets for the Simulation of Communication Networks, Tool Description, PNPM '91 (1991), also in IEEE Journal on SAC, Vol. 8, No. 9, pp. 1735-1747 (1990).
7) Parent, P., Tamir, O.: Voltaire: A Discrete Event Simulator, Tool Description, PNPM '91 (1991).
8) Archetti, F. et al.: A Software Environment for the Generator, Representation and Analysis of PN based Models of Manufacturing Systems, Tool Description, PNPM '91 (1991).
9) Billington, J. et al.: FORSEE, Tool Description, PNPM '91 (1991).
10) Couvillion, J. et al.: Performability Modeling with UltraSAN, Proc. of PNPM '91, pp. 290-299 (1991).
11) ESPRIT Project PROOFS Leaflet, in ICATPN '92 (1992).
12) Marca, D. A. and McGowan, C. L.: SADT, McGraw-Hill, New York (1988).
13) Harel, D.: Statecharts: A Visual Formalism for Complex Systems, in Science of Computer Programming, Vol. 8, North Holland, pp. 231-274 (1987).
14) Jensen, K.: Coloured Petri Nets, in Brauer, W. et al. (eds.) Lecture Notes in Computer Science, Vol. 118, Springer-Verlag, pp. 327-338 (1981).
15) Jensen, K.: Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use, Vol. 1, Basic Concepts, Springer-Verlag (1992).
16) Grenrich, H. J. and Lautenbach, K.: System Modelling with High-Level Petri Nets, Theoretical Computer Science, 13, pp. 109-136 (1981).
17) Grenrich, H. J.: Predicate/Transition Nets, in Brauer, W. et al. (eds.) Lecture Notes in Computer Science, Vol. 254, Springer-Verlag, pp. 207-247 (1987).
18) Huber, P. et al.: Hierarchies in Coloured Petri Nets, 10th Int. Conf. on Application and Theory of Petri Nets, Bonn (1989).
19) Wikstrom, A.: Functional Programming Using Standard ML, Prentice-Hall (1987).
20) Harper, R. et al.: Standard ML, Tech. Report ECS-LFCS-86-14, University of Edinburgh, Edinburgh (1986).
21) Albrecht, K. et al.: Design/CPN: A Tool Package Supporting the Use of Colored Petri Nets, Meta Software, Cambridge (1989).
22) Jensen, K. and Rosenberg, G. (Eds.): High-Level Petri Nets, Theory and Application, Springer-Verlag, Berlin (1991).
23) Feldbragge, F.: Petri Net Tools Overview 92, Petri Net Newsletter, 41, Gesellschaft fur Informatik, Bonn (1992).
24) Lindermann, C.: DSPNexpress: A Software Package for the Efficient Solution of Deterministic and Stochastic Petri Nets, Proc. of 6th Int. Conf. on Modeling Techniques and Tools for Computer Performance Evaluation (1992).
25) van Hee, K. M., Somers, L. J. and Voorhoeve, M.: Executable Specifications for Distributed Information Systems, In Falkenberg, E. D. and Lindgreen, P. (Eds.) Information System Concepts: An In-depth Analysis, North-Holland, pp. 139-156 (1989).

(平成5年3月15日受付)

寺内 睦博(正会員)

1960年生．1984年関西大学工学部機械工学科卒業．1986年同大学院博士前期課程修了．1989年広島大学大学院博士後期課程(情報工学)単位取得退学．同年広島大学工学部第二類(電気系)助手．現在に至る．コンピュータビジョン，画像理解，並列分散処理，マン・マシンインタフェースに関する研究に従事．電子情報通信学会，計測自動制御学会，人工知能学会，視聴覚情報研究会，コンピュータ支援画像診断学会，日本機械学会，日本人間工学会，IEEE各会員．