Smalltalk-80 における拡張 MVC モデルとその応用†

増田英孝†† 笠原 宏††

本論文では，Smalltalk-80 を用いて対話型システムのユーザ・インタフェース・ツールを実現するために従来の MVC モデルを拡張した拡張 MVC モデルを提案する．グラフ表現を持つシステムのモデルの構造が，図式を用いて対話的に計算機内部に構築でき，さらに要素の移動や削除が直接画面上で図式を利用して実行できる．本研究ではスーパビュー内のサブビューが動的にオーバラップできるように新たにビュー・コントローラのクラスを作成し，ビューを開いた後でサブビューを開く，閉じるという機能も持たせた．以上の機能拡張により，グラフ表現を持つモデルの構成要素オブジェクトが MVC の組合せで実現できる．したがって，オブジェクトに図式表現を持たせることができ，オブジェクトが固有のメニューや操作を持つことができる．その結果として，ビジュアルオブジェクトが実現でき，さらに直接ユーザの種々の要求に答えることができる．また，これらのビジュアルオブジェクトを動的に生成・消滅できる．そして，異なったオブジェクトに対しても既存のビュー・コントローラの組を容易に再利用できる．応用として本方式を並行処理システムの解析等で最近注目されているペトリネットに適用した場合について述べる．

1. はじめに

最近の CAD 等の対話型システムでは，画面上でグラフィックを用いて対話的に対象となるシステムのモデルを計算機内部に構築することが要求される[1]．システムのモデルは，なんらかのグラフ表現を用いて表されることが多い[1]．グラフの構成要素としてはノードとアークが考えられ，これらの要素を画面上でグラフィックを用いて対話的に取り扱えることが必要となる．さらに，削除，変更等に対しても画面上で図式を利用して直接操作できなければならない[1]．

ウィンドウを用いた対話型アプリケーションを開発する一つの設計指針として，オブジェクト指向型言語である Smalltalk-80[2),3)] における MVC パラダイム[4)] がある．モデル・ビュー・コントローラの3種類のオブジェクトでアプリケーションを構築する MVC (Model-View-Controller) の概念は，アプリケーション本体とユーザ・インタフェース部分の明確な分離にある．

オブジェクト指向[5)]や MVC の概念[4)]により，対象となる任意のシステムのモデルを，グラフィックスを用いて対話的に編集・実行できるエディタ／シミュレータが開発可能[6)~10)]となり，短時間で容易に複数の任意のシステムの設計案を試すことができ，対象を直接操作して素早く複数の可能性を作り出すことができる[11)]．

このようなエディタでは，常にシステムの一つ一つの構成要素とユーザが対話できることが必要である．これは，システムのモデルにビュー・コントローラを持たせることで実現できるが，それだけでは対話の制御が複雑になってしまう．そこで，本研究では，この各々の構成要素にもビュー・コントローラを持たせることにする．これによって上記の対話性が向上でき，複雑さも解消できる．

また，これらの構成要素を動的に生成・移動・削除ができることも必要不可欠であるが，本来の Smalltalk-80 の階層構造を持つビューでは，サブビューはウィンドウの固定した構成要素として考えられており，このためサブビューの動的な取り扱いは考えられていない[12)]．

そこで，これらの目的を容易に達成するために，ユーザが自由に動的な構成を実現できるサブビューを導入し，それらのサブビューがそれぞれ異なったモデル（システムを構成する一つ一つの要素）を持つ新しい MVC フレームワークを提案する[13)]．

現在，ビュー（ウィンドウ）およびコントローラの改善を目的とした種々のツールが提案されたり，拡張がなされている．

モデルとビュー・コントローラ間の対話を標準化し，様々なモデルに手軽に利用できるようなインタフェースを実現する目的として，プラガブル (Pluggable) MVC[14)~17)] の概念が Smalltalk-80 に導入されている．モデルに依存しない標準化されたビュー・コントローラをプラグインすることによって MVC が簡単に

† Extended-MVC Model on Smalltalk-80 and Its Application by HIDETAKA MASUDA and HIROSHI KASAHARA (Department of Electrical Engineering, Faculty of Engineering, Tokyo Denki University).

†† 東京電機大学工学部電気工学科

構成できる．また，一つのモデルをいくつかのサブビューに表示するために，各サブビューに機能を表す名前（アスペクト）を付け，モデルの変化に従ってどのビューを書き換えるかを指示して不必要な再描画の遅延を避けている．しかしながら，一つのモデルを複数の視点（テキスト，ビットマップ，リスト等）から表示することを目的とするプラガブルMVCは，複数の同じ表示や機能を持ったサブビューを使用する本研究の目的には適さない．

対話形式でウィンドウを作成するためのツールとしてはGlazier[15]がある．Glazierではプラガブル MVCを利用し，さらに対話形式でサブビューの配置・再配置ができるが，アプリケーションに固有のウィンドウを作成し，そのクラスを自動生成することを目的としている．したがって，動的なサブビューの取り扱いができるが，本研究とは目的を異にしている．

本論文では，まず，Smalltalk-80 の MVC モデルの概要に触れ，次に拡張 MVC モデルについて述べる．最後に，応用例として本方式を並行処理システムの解析等で最近注目されているペトリネット (Petri Net)[18),19)] に適用した場合について述べる．

2. グラフ表現を持つモデル

通常，Smalltalk-80 を用いてグラフ表現を持つ対象のモデルを作成する場合は，次の2種類のオブジェクトについて考える[20)]．

a．対象を構成する要素である個々のオブジェクト

b．対象全体を管理するオブジェクト

以下，各々オブジェクト(a)，オブジェクト(b)と表す．

オブジェクト(a)にはノードとアークの2種類がある．オブジェクト(b)はノードとアークを含むグラフ全体である．このようなモデルの具体的な例としては，電気回路[20)]，ブロック線図[8)]，機械振動系[1)]といったモデルが考えられる．電気回路を例にとれば，構成要素であるオブジェクト(a)には抵抗，電源などがある．そして，電気回路そのものがこれらの要素を管理するオブジェクト(b)にあたる．これらの構成要素をクラスとして実現しておけば，必要なインスタンスを生成して様々な電気回路を構成できる．これらの要素を画面上で対話的にグラフィックを用いて接続情報を入力・編集できることが必要である．

3. 拡張 MVC モデル

3.1 MVC モデル

ウィンドウを用いた対話型アプリケーションを開発する設計指針として，モデル・ビュー・コントローラの3種類のオブジェクトでアプリケーションを構築する MVC (Model-View-Controller) パラダイム[4)]がある．MVC モデル（**図 1**）は，

モデル　問題対象としてのデータとそのデータに対する操作．

ビュー　ディスプレイを通じて，モデルからユーザへ情報を提供するもの．

コントローラ　ユーザからの入力を解釈して，モデルあるいはビューに適切な調整を施すもの．

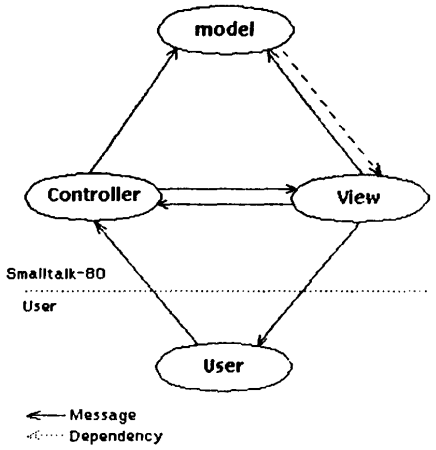


図 1 MVC モデルの概念図
Fig. 1 A Model-View-Controller (MVC) triad in Smalltalk-80.

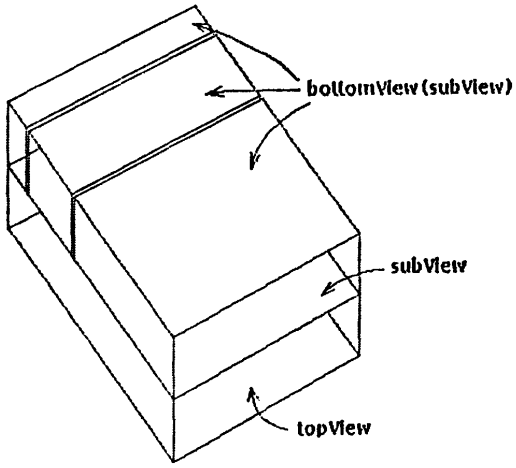


図 2 サブビューを持つビューの例
Fig. 2 SubViews in a view at Smalltalk-80.

の3種類のオブジェクトによって構成される[4]．モデルを様々な側面から見ることができるようにウィンドウ（ビュー）はさらに複数のビューを持つことができる[4]（**図 2**）．MVC の主な利点は，

1. アプリケーション本体は，自分のインタフェースであるウィンドウの状態を気にすることなく記述できる．
2. アプリケーション本体は，モデルとして各々が開発して，それに対するインタフェースは，既存のビューとコントローラを再利用してカスタマイズできる．

ことにある[21]．既にあるモデルおよび新しく作成したモデルに関わらず，既存のまたは自分で調整したビュー・コントローラを利用できるためユーザ・インタフェースの自由度が非常に大きくなる．また，MVC は，

1. 単純な MVC[4),22)]
2. 依存性を利用する MVC[4),23)]
3. プラガブルを利用する MVC[14)~17)]

の三つに大別され，開発過程やアプリケーションに適合した構成をとることができる．

Smalltalk-80 上でグラフィックを用いて対話的に対象の構造を入力・編集できるツール[7)~10)]には，CAD である INSIST[7)] や制御系の解析設計支援システム[8)] がある．システムのモデルをグラフィック表現を用いて入力することは比較的容易であるが，その後移動や削除といった操作を画面上でグラフィックを用いて対話的に行うためには対話の制御が複雑になる[1)]．これらのツールでは，モデル構造の入力・編集用のウィンドウを用意し，このウィンドウの上で編集作業が行える．システムのモデルに対してビュー・コントローラを用意し，このビュー・コントローラが入力・編集に対する対話のすべての制御を行う方法が一般的である．しかし，この方式では各要素や操作に対するコントローラの制御が複雑になる．

3.2 拡張 MVC モデルの構成

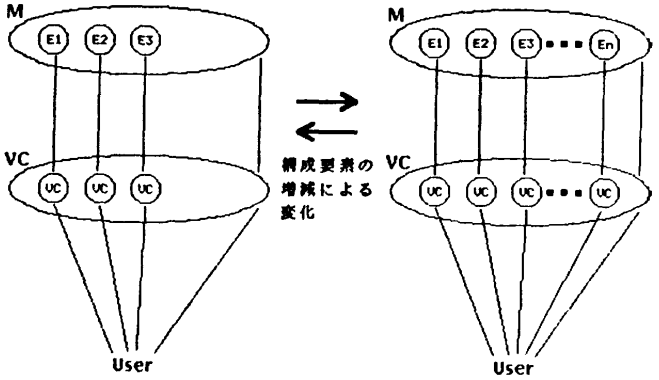
MVC の構成をとれば，任意のオブジェクトとユーザの対話が容易に実現できる．ユーザが必要なときに直ちにシステムの一つ一つの構成要素と対話できることは重要である．ウィンドウ上に見えるオブジェクトの中から一つをマウスで選択し，メニューを表示するといった処理が必要になる．

本研究では，対象となるモデルにビュー・コントローラを持たせるだけでなく，ユーザと構成要素との対話のために各々の構成要素にもビュー・コントローラを持たせることによってこれを実現する．これによって，MVC の利点が各構成要素に対しても利用でき，対話に必要な機能が各ビュー・コントローラに局所化できる．個々の構成要素に対するビュー・コントローラが，要素に対する表示と操作の責任を受け持つことになる．

しかし，これらの構成要素を動的に生成・移動・削除できることも必要不可欠であるから，これに伴うビュー・コントローラの対話的な生成・移動・消滅が必要になる．だが，本来の Smalltalk-80 のサブビューにはこれらの機能が実現されていないため，ユーザの操作によってアイコンや表示されたものの配置が変わる場合には簡単にサブビューを使用することができない．

そこで，この目的を容易に達成するために，ユーザが自由に動的な配置構成を実現できるサブビューを導入し，それらのサブビューがそれぞれ異なったモデル（システムを構成する一つ一つの要素）を持つ新しい MVC フレームワーク（**図 3, 図 4**）を提案し，これを拡張 MVC モデルと呼ぶ[13)]．

ユーザが自由に動的なサブビューの配置構成を実現できるツールに Glazier[15)] がある．Glazier では，プラガブル MVC を基にして，ユーザがアプリケーションに適合したウィンドウを対話的に作成することを目的とする．プラガブル MVC（**図 5**）[14)~17)] では，モデルに対して複数のビューを持たせることができ，それ



En: element
VC: View-Controller

図 3 拡張 MVC モデルの概念図
Fig. 3 A concept of Extended-MVC.

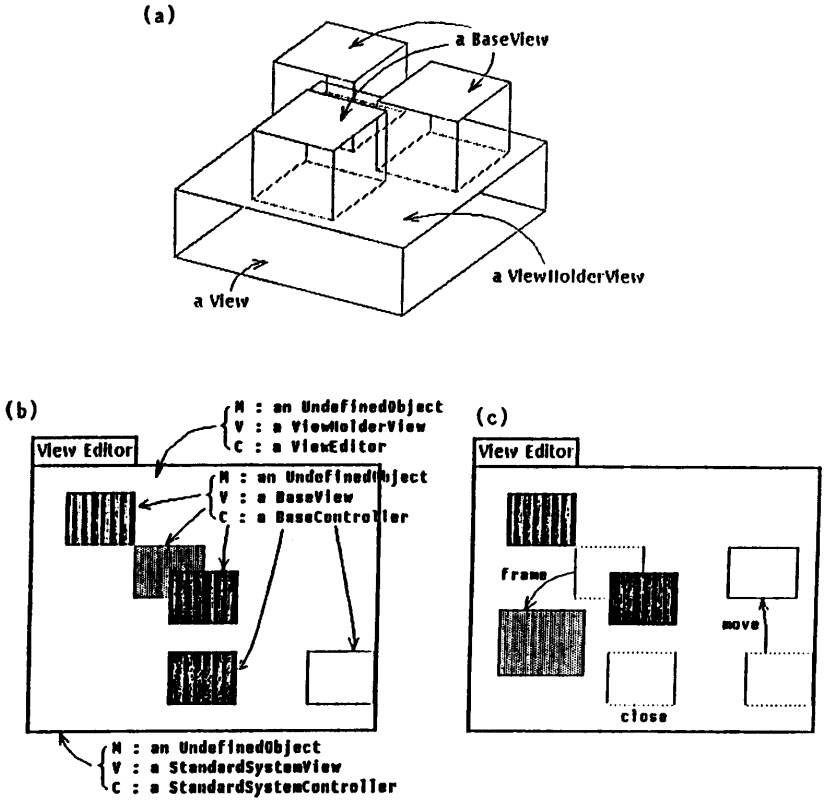


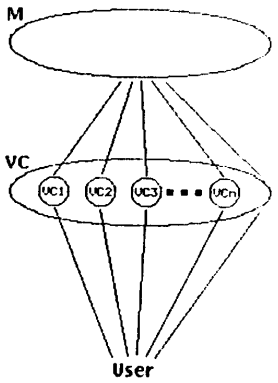
図 4 スーパビューの中の動的なサブビュー
(a) 拡張 MVC の構成例, (b) 各ビューの MVC 構造, (c) サブビューに対する操作例.
Fig. 4 Implementing an overlapping multipaneview in a window. (a) SubViews: BaseView that move freely in a superview: ViewHolderView. (b) View structure. (c) An example of operation.

ぞれのビューが機能を表す名前を持つ．モデルに対して，テキスト，ビットマップ，リスト等の異なった表現を容易に持たせることを目的としている．

本研究では，グラフ表現を持つオブジェクトの構成要素との対話を目的としている．そのため各構成要素に持たせるビュー・コントローラは機能が同じであり，それが構成要素の数だけできることになる（図3）．どの構成要素がどのビュー・コントローラと関係しているかという依存関係がわかればよく，名前管理をする必要がない．ビュー・コントローラが直接構成要素を持っているため要素に変化があった場合には依存を利用してそのビューを書き換えればよい．

プラガブル MVC では，機能が全く同じビューであっても異なった名前を付ける必要がある．例えば表計算[14]のように多数の同じ機能を持ったビューを使用するようなアプリケーションでは名前管理を考える必要がある．

本研究では，ユーザが自由に動的な配置構成を実現



VC*n*: View-Controller

図 5 プラガブル MVC の概念図
Fig. 5 A concept of Pluggable-views.

できるサブビューとスーパビューの2種類のビュー・コントローラのクラスをユーザ・インタフェース・フレームワークとして実現した．ユーザは拡張 MVC モデルを使用することによって，いつでもサブビューの

配置・再配置が動的に行える．2種類のビュー・コントローラの組としての拡張 MVC（図3，図4）の構成要素は，スーパビューとして，ViewHolderView (対になる ViewEditor) サブビューとして，BaseView (対になる BaseController) から構成される．クラス階層を図6に示す．拡張 MVC は通常の MVC と同等に使用できるため，ViewHolderView は，あるビューのサブビューになることができ，BaseView はさらにサブビューを持つことができる．

Smalltalk-80 にはモデルとビューの依存関係を管理する辞書である DependentsFields[8),14),15),17),23)] があり，本方式ではサブビューが各構成要素オブジェクト

```
Object
  Controller
    MouseMenuController
      BaseController
      ViewEditor
  View
    BaseView
    ViewHolderView
```

図 6 拡張 MVC のクラス階層
Fig. 6 Class hierachy of Extended-MVC.

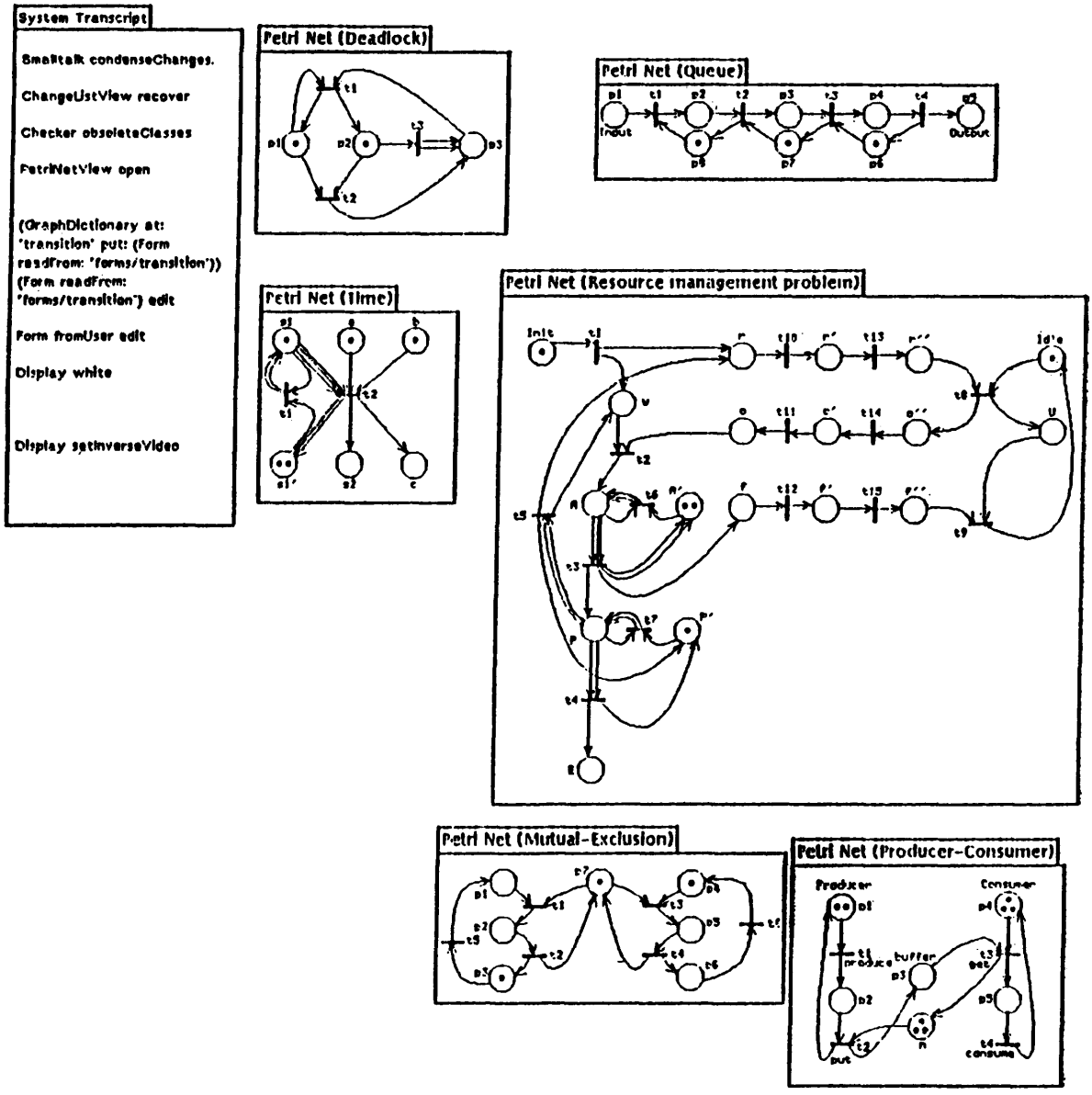


図 7 ペトリネット・エディタの例
Fig. 7 Several examples of designed Petri Nets.

ごとにDependentsFieldsに登録される．サブビューの生成時に，その構成要素モデルと依存関係にあるサブビューがDependentsFieldsに自動的に登録され，サブビューの消滅時に，その構成要素モデルと依存関係にあるサブビューがDependentsFieldsから削除される．構成要素はdependentsから自分のビューおよびコントローラを求めることができ，自分自身に変化があった場合は自分にchangedメッセージ[4),8),16),17),24)]を送ることによってビューが再描画される．

3.3 拡張MVCを用いたモデルの構造入力・編集

本方式では拡張MVCを用いて構成要素である各オブジェクト(a)にMVC構造を持たせる．そのため，オブジェクト(a)を表現しているビューにカーソルを持っていくだけで，そのオブジェクト(a)のコントローラがアクティブとなり，そのオブジェクト(a)に対して直接，移動や削除等の指示をすることができ，さらに各オブジェクト(a)に固有の操作をメニュー等で指示できる．各々のコントローラによってカーソルの位置からどの構成要素が指されているかをサーチする手間が省け，構成要素の表示に対しても各々のビューが責任を持つため制御が局所化され容易になる．移動や削除に伴う表示上の変更は，それぞれのビジュアルオブジェクトが個々に責任を持つ．

MVC構造で実現されたノードオブジェクト(a)をスーパビュー内に配置し，その後，アークオブジェクト(a)を生成し，構成要素の接続関係を画面上でマウスを用いて指定して接続情報を入力する．接続情報はメッセージによってオブジェクト(b)に伝えられ，計算機内部にその構造が構築される．

ビューおよびコントローラが独立しているため，モデルは新しく作成したもの，既存のものを問わずに組み合わせて利用でき，その表示方法もユーザが柔軟に変更できる[4)]．さらに，図式の接続関係だけでなく，グラフ表現を持つモデルが計算機内部に直接構築できるため，直ちにそのモデルが実行可能となる．本方式を用いれば，グラフ表現を持つオブジェクトの各々の構成要素に対する直接操作が容易に実現できる．

```
Object
  Controller
    MouseMenuController
      BaseController
        NetArcController
        NetPartsController
          PlaceController
          TransitionController
      ViewEditor
        PetriNetController
    NoController
  View
    BaseView
      NetArcView
      NetPartsView
        PlaceView
        TransitionView
    TokenView
    ViewHolderView
      PetriNetView
  NetArc
  PetriNet
  Place
  Token
  Transition
```

図8 ペトリネット・エディタのクラス階層
Fig. 8 Class hierachy of Petri Net Editor.

4. ペトリネット・エディタへの適用

4.1 ペトリネットのモデル構成

並行動作する対象のモデル化に用いられるペトリネット(Petri Net)[18),19)]は，近年様々なシステムの解析に利用されている．計算機を用いたペトリネットの解析[25)~27)]は盛んに研究されており，現在も発展中である．ペトリネットは数学的表記のほかにペトリネットグラフ[18),19)]と呼ばれる2部有向多重グラフで表され，これらの解析系では強力なグラフィック機能を持ったユーザ・インタフェースが求められている[26)]．ペトリネットは二つの型のノード（プレイス(place)とトランジション(transition)）およびアーク(arc)からなる．また，プレイスはマーキングのためにトークン(token)を持つことができる．オブジェクト指向では，

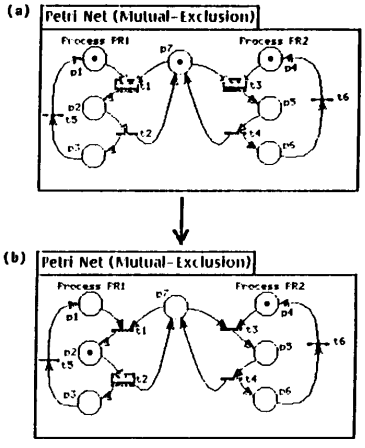


図9 ペトリネットの実行例
Fig. 9 An example of executed simulation of a Petri Net.
Transition: t1 in (a) is fired and changed to (b).

一般に目に見えるものをオブジェクトとすればよい[20]ので，ペトリネットのクラス構成は，PetriNet, Place, Transition, NetArc, Token の五つのクラスが考えられる．このうち，対象全体を管理するオブジェクト(b)は PetriNet のインスタンス，対象を構成する要素であるオブジェクト(a)は残りの4種類である．

4.2 ペトリネット・エディタの構成

ここではペトリネットを画面上で作成・編集し，さらにその実行をシミュレートするペトリネット・エディタに対して拡張 MVC を適用した例を示す．前節のペトリネットモデルは，メッセージ式の記述により単独で作成・実行ができる．本方式では，ペトリネットを構成する各クラスに対して対応するビュー・コントローラのクラスを用意し，画面上で動的にペトリネットが作成・編集できる．また，各構成要素が MVC 構造を持つので，例えばプレイスが持つトークンの数の設定などが直接画面上で行える．トークンはプレイスに存在するものであり，プレイスが MVC 構造を持つため PlaceView はさらにサブビューとして TokenView を持つことができる．ペトリネット・エディタの例を**図 7** に示す．TokenView は PlaceView の中を移動する必要がないので通常のサブビューである．これらのクラス構成を**図 8** に示す．ペトリネットの実行例を**図 9** に示す．なお，各オブジェクトに対する操作はメニューによって選択できる．ペトリネット・エディタの各ビューの構成を**図 10** に示す．PlaceView および TransitionView は，PetriNetView のサイズが変更されても大きさが変わらない固定サイズビュー[12]として実現している．NetArcView に関しては，現在はアーク上のある点の上に小さな領域を持た

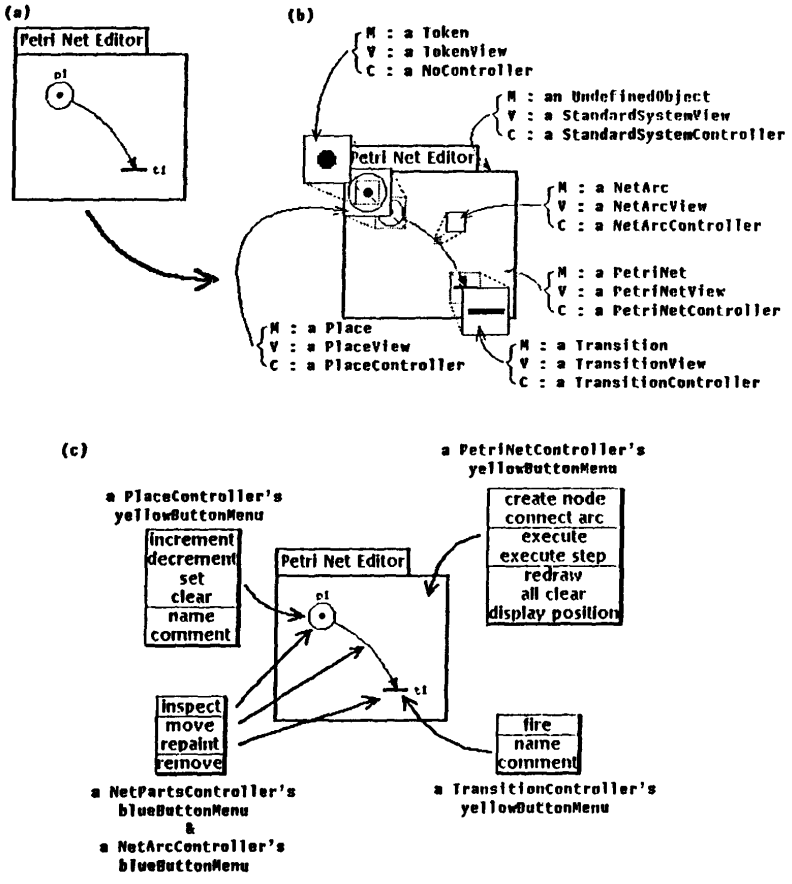


図 10 ペトリネット・エディタの MVC 構成
(a) ペトリネット・エディタの例，(b) (a) の MVC 構成，
(c) 各コントローラの持つメニュー．

Fig. 10 The view structure of Petri Net Editor.
(a) Place: p1 and Transition: t1 is connected by an arc. (b) View structure corresponding to (a). (c) Operation menus of each controller.

せている．アークは，制御点を対話的にマウスで移動させることによって任意の形状をとることができる．今後，NetArcViewに任意の形を持つことができるビューの機能を持たせる予定である．

5. おわりに

本論文で提案する拡張MVCを用いて，ウィンドウ内に直接操作可能なビジュアルオブジェクトを容易に実現できるようになった．そして，これらのビジュアルオブジェクトをウィンドウ内に配置し，マウスで接続関係を指示することによってグラフ表現を持つシステムのモデルが計算機内部に構成できる．ユーザは構成要素に対して直接触れることができ，さらに移動，削除等の操作も直接指示できる．

拡張MVCは直接操作可能なビジュアルオブジェクトを実現するためのフレームワークである．しかし，構成要素一つに対してビュー・コントローラという二つのオブジェクトが必要になり，オブジェクトの数の増加およびメモリの圧迫等がおこる．だが，近年のソフトウェア・ハードウェア技術の進歩により，使用できるオブジェクトポインタの数，メモリ容量等の点は克服されてきている．

本研究では実際の適用例として，近年並行処理システム等の解析で注目されているペトリネットについて応用した例をあげた．様々なペトリネットが画面上で対話的に作成でき，さらに実行できる．構成要素の移動・削除等の操作が各オブジェクトに対して統一的に行える．

今後，アークとして直線 (Line)，折れ線 (LinearFit)，曲線 (Curve)，スプライン曲線 (Spline)，ベジエ曲線 (BezierCurve) 等を統一的に扱えるビュー・コントローラのクラスを作成する予定である．さらにペトリネットを部分ネットの集合として階層的に取り扱うことができる機能を持たせる予定である．

最後に，本研究について有益なご討論とご協力をいただいたソニー・テクトロニクス株式会社情報機器部AI営業部，豊元詮氏に感謝いたします．

参考文献

1) 小林，原：オブジェクト指向によるCAEシステムのユーザ・インタフェース，*bit*, Vol. 20, No. 8, pp. 921-930, 共立出版 (1988).
2) Goldberg, A. and Robson, D.: *Smalltalk-80 The Language and Its Implementation*, Addison-Wesley (1983).
3) Goldberg, A.: *Smalltalk-80 The Interactive Programming Environment*, Addison-Wesley (1984).
4) Cunningham, W.: Smalltalk-80によるアプリケーション・プログラムの作り方，*bit*, Vol. 18, No. 4, pp. 379-396, 共立出版 (1986).
5) 山本：オブジェクト指向とシミュレーション，情報処理，Vol. 29, No. 4, pp. 374-381 (1988).
6) 仲田，北川：Smalltalk-80による商用アプリケーションの開発について，*TURING MACHINE*, Vol. 2, No. 2, pp. 67-72, オーム社 (1989).
7) van der Meulen, P. S.: INSIST: Interactive Simulation in Smalltalk, *Proc. of the ACM Conf. on Object-Oriented Programming Systems, Languages and Applications*, pp. 366-376 (1987).
8) 赤堀，原：オブジェクト指向に基づく制御系の解析・設計支援システム，計測自動制御学会論文集，Vol. 24, No. 5, pp. 506-513 (1988).
9) Grossman, M. and Raimund, K. E.: Logical Composition of Object-Oriented Interfaces, *Proc. of the ACM Conf. on Object-Oriented Programming Systems, Languages and Applications*, pp. 295-306 (1987).
10) Freburger, K.: RAPID: Prototyping Control Panel Interfaces, *Proc. of the ACM Conf. on Object-Oriented Programming Systems, Languages and Applications*, pp. 416-422 (1987).
11) Shneiderman, B.: *Designing the User Interface*, Addison-Wesley (1987).
12) Epstein, D. and LaLonde, W. R.: A Smalltalk Window System Based on Constraints, *Proc. of the ACM Conf. on Object-Oriented Programming Systems, Languages and Applications*, pp. 83-94 (1988).
13) 増田，笠原：図式表現を持つオブジェクトのための拡張MVCモデル，昭和63年度電気関係学会東北支部連合大会，1A-8 (1988).
14) 上谷：統合化プログラミング環境—Smalltalk-80とInterlisp-D—，丸善 (1987).
15) Alexander, J. H.: Painless Panes for Smalltalk Windows, *Proc. of the ACM Conf. on Object-Oriented Programming Systems, Languages and Applications*, pp. 287-294 (1987).
16) Smalltalk-80スクール・テキスト，ソニー・テクトロニクス株式会社情報機器テクニカル・サポート部，3版 (1988).
17) 使わないと損をするModel-View-Controller，富士ゼロックス情報システム株式会社，資料 (1988).
18) Peterson, J. L.: ペトリネット入門，共立出版 (1984).
19) 山崎：ペトリネットの理論と応用，情報処理，Vol. 25, No. 3, pp. 188-198 (1984).
20) 豊元，寺田，小方：Smalltalk-80. 昭晃堂 (1987).

21) 福永，沼尾：オブジェクト指向言語のプログラミング環境，情報処理，Vol. 29, No. 4, pp. 334-343 (1988).
22) 梅村：Smalltalk-80 入門，サイエンス社 (1986).
23) 春木：オブジェクト指向への招待，啓学出版 (1989).
24) Cunningham, W. and Beck, K.: A Diagram for Object-Oriented Programs, *Proc. of the ACM Conf. on Object-Oriented Programming Systems, Languages and Applications*, pp. 361-367 (1986).
25) Bruno, G. and Balsamo, A.: Petri Net-Based Object-Oriented Modelling of Distributed Systems, *Proc. of the ACM Conf. on Object-Oriented Programming Systems, Languages and Applications*, pp. 284-293 (1986).
26) 孫，落水：ペトリネットによる並行ソフトウェアシステムの設計時動作解析，情報処理学会論文誌，Vol. 29, No. 8, pp. 782-789 (1988).
27) 村田，鷹田，解良：色付きペトリネットに基づくリアルタイム情報処理用ソフトウェア，情報処理学会論文誌，Vol. 29, No. 12, pp. 1129-1140 (1988).

(平成元年 4 月 7 日受付)
(平成元年 11 月 11 日採録)

増田 英孝（学生会員）

昭和 40 年生．昭和 62 年東京電機大学工学部電気工学科卒業．現在同大学院工学研究科電気工学専攻修士課程在学中．オブジェクト指向とユーザ・インタフェースに関する研究に従事．ACM，日本ソフトウェア科学会各会員．

笠原 宏（正会員）

昭和 15 年生．昭和 39 年東京電機大学電気工学科卒業．昭和 45 年同大学博士課程満期退学．工学博士．同大学助手，講師，助教授を経て現在工学部電気工学科教授．パワーエレクトロニクス，計算機制御，制御用分散処理システム，オブジェクト指向言語によるシステム設計法の研究に従事．電気学会，ロボット学会，IEEE, ACM 各会員．