Abstract

ペトリネットは非同期的でかつ並列的に振る舞うシステムに対して、その中の情報の流れや制御を記述し解析するために考え出されたもので、いくつかの事象が並列的に発生する中で、それらの発生の順序、頻度などにある制約が与えられているようなシステムをモデル化するために主として用いられてきた。「システム」とは、ハードウェアシステムまたは、ソフトウェアシステムのことを指す。

また仕様書とは、システムの論理的な機能を説明する文章である。通常システム開発を行う際に、人間がその仕様書の意味を解釈して、システムを作り上げる。また、出来上がったシステムと仕様書の内容が同一であるかどうかについても、人間がテストして判断をしている。簡単なシステムではそれも可能であるが、複雑なシステムになると人間の判断ではチェックしきれなくなってくる。さらに、システムの開発途中に仕様が変わることもよくあることであるが、あとから仕様書を実際のシステムに合わせるのも大変な作業となる。これらの作業を自動化するために、ペトリネットからソフトウェアシステムを開発し、その仕様書もペトリネットから自動で書き出し、自然言語の文章に出来ないかということを考えてみる。本論は、ペトリネットを応用したシステム仕様書の作成方法を考察し、またそれを実現するためのツールの作成を目的としている。

ペトリネットをデザインするツールの作成は、ペトリネットをユーザーが視覚的にデザインし、なおかつ実行後の状態推移の検証やデザインしたペトリネットからのソフトウェアの自動作成ならびに自然言語の文章での仕様書の自動作成を目的としたものである。
これまでにも MS-DOS などの CUI(Character-baced User Interface)環境下で動作するものは考えられてきたが、今後は Windows などの GUI(Graphical User Interface)環境下で、視覚的操作性などの特長を生かしたものを考えていかなくてはならない。また昨今コンピュータのネットワーク化は当然となってきており、個々のコンピュータに依存することなく、同一の操作環境を提供出来ることなども考慮しなくてはならない。これらのことから使用する言語は java 言語を選択した。GUI 環境下でなおかつ、ネットワーク内のコンピュータ上において、同じ操作性で使えるものを考えていく。

目次

1　序論

ペトリネットとは、1960 年代にドイツ人の Carl Adam Petri によって考案された理論的計算モデルである。グラフ・ノードにトークンを用いてマークを付けることにより、システムの状態をモデル化することができる。ペトリネットのグラフには二種類のノードが存在する。円で表現されたプレース（place）と四角で表現されたトランジション（transition）である。ペトリネットの状態は、各プレースの中に小丸を書き入れることで表現できる。この小丸をマーク（mark）またはトークン（token）という。また状態は時々刻々と変化する。変化のしかたは、各トランジションの入力プレース全てにマークがあるときそれらのマークを一つずつ取り去り、反面各出力プレースにマークを一つずつ増やす。これをトランジションが発火するという。ペトリネットの基本的な動きは例えば、プレース要素が 3 つ（s1,s2,s3）、トランジション要素が 2 つ（t1,t2）、初期マーキングは s1 にマークが 1 つ、発火可能なトランジションは 1 つ（t1）というペトリネットを考える。（図 1）

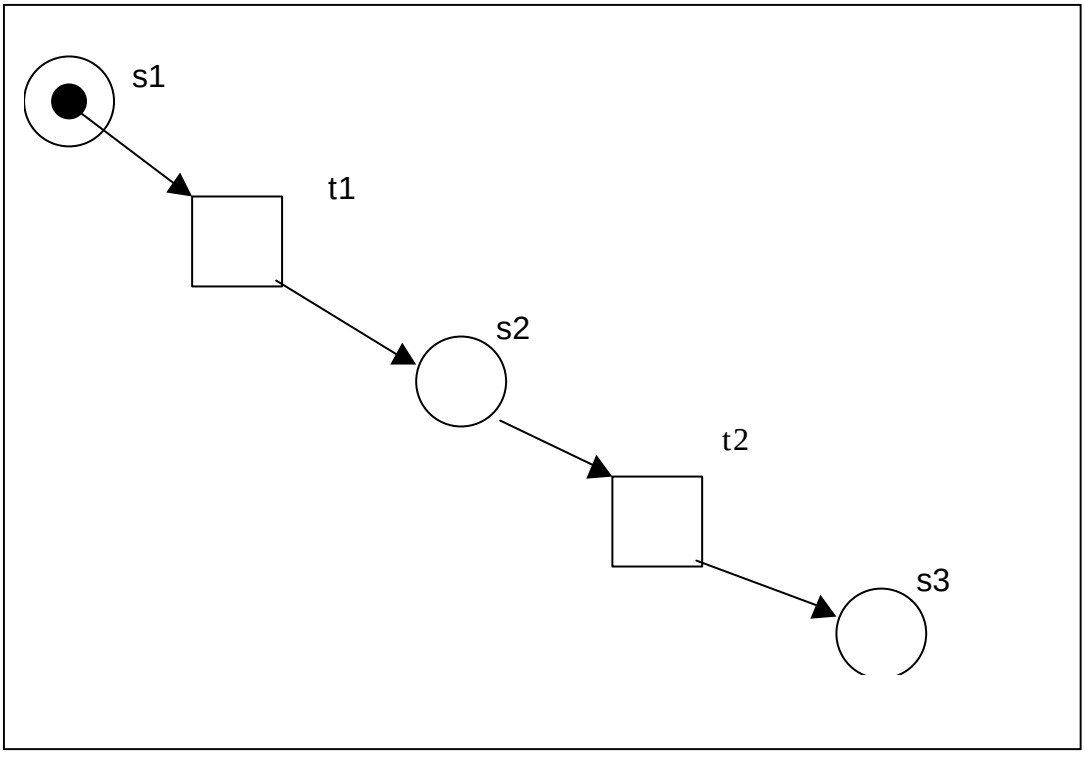


図 1：簡単なペトリネット

これを「実行」すると、発火可能なトランジション t1 を発火させ、マーキングは s1→0,s2→1,s3→0 になる。次は、s1→0,s2→0,s3→1 になり、ペトリネットはこれ以上変化しない。

これを発展させた考え方に、カラーペトリネットがある。前述したペトリネットとの違いは、マークがデータを保持している点である。トランジションは、その全ての入力プレースにマークがそろったとき発火し、その下流にマークを送るとする。そのとき計算式を

下流プレースごとに用意しておき、マークの中のデータを計算してその値を下流に送るマークにデータとして持たせる。以下に例を示す。

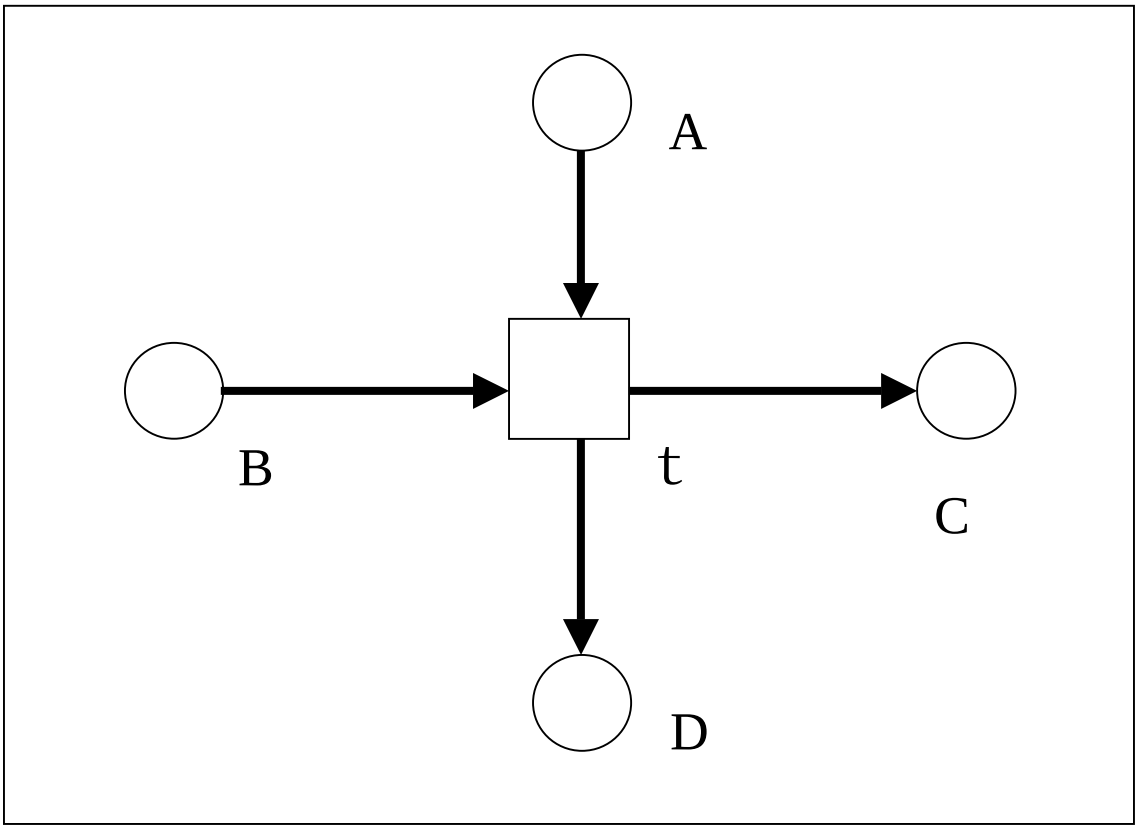


図 2：カラーペトリネットの例

例えば図 2 において、トランジション t の入力プレースが A、B とする。出力プレースは C、D である。プレース A、B に一桁の整数をデータとして持つマークが入るとし、それらの値を a、b とする。トランジション t では、下流プレース C のために

$$c = (a+b) \bmod 10$$

の計算をする。mod は余りの意味である。また下流プレース D のためには

$$d = (a+b) \div 10$$（小数以下切り捨て）

という計算をする。c、d はプレース C、D に入るマークの値を表す。これは、10 進数の計算回路の１ユニットを表す。

２．ペトリネットの考え方

ペトリネットの概念を一言でいうと、「協調して仕事をするべき作業者が複数いたとき、それらの仕事の前後関係と、各作業者が仕事をいつ開始できるかを表した図」となる。作業者とは、仕事をしうるものであれば、コンピュータであってもロボットであってもソフ

トであっても良い。全体は抽象的に記述されるので、その対象範囲は広いと言える。考え方の例を示す。みかん農家がみかんを出荷するときに梱包するための作業所を考える。（図 3）

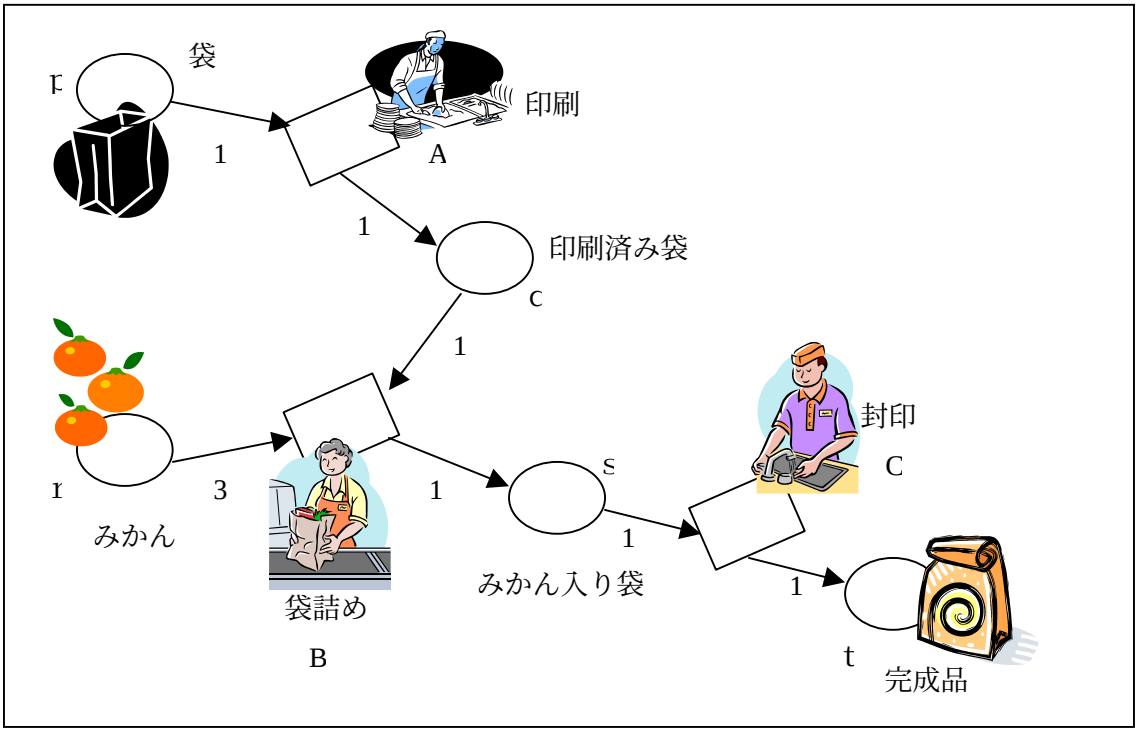


図 3：みかん農家の作業の流れ

各作業者の前後には、材料あるいは半完成品（または完成品）を置く場所がある。それらを p、q、r、s、t とする。矢印の下の数字は 1 度に動く物の数を表す。例えば、置き場 r から作業者 B へは、一度に 3 つのみかんが動く。作業者達は作業のスピードが異なっている。そのため中間の置き場には、半完成品がたまってしまったり、あるいは無くなってしまったりする。もし自分の上流にある置き場に物が無くなってしまえば、その作業者は手を休めなくてはいけない。例えば作業者 B は、置き場 q に袋が無くなってしまえば、手を休めなくてはいけない。あるいは、置き場 r にあるみかんが 3 個未満になったら、やはり手を休めることになる。そして次に作業を開始出来るのは、置き場 q に少なくとも 1 つの袋が置かれ、置き場 r には少なくとも 3 つのみかんが置かれたときである。このような事象を抽象化して表したのがペトリネットである。（図 4）

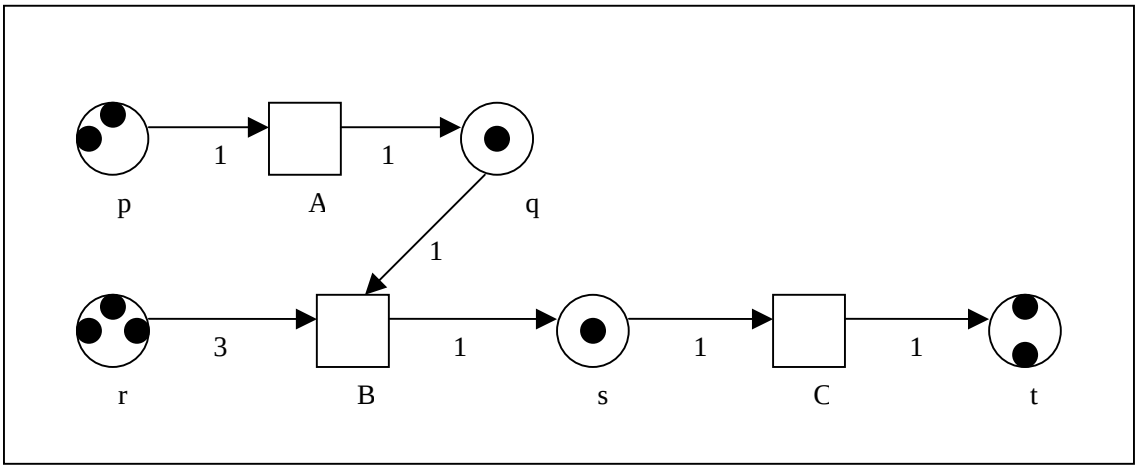


図 4：状態の流れを抽象化する。

丸が置き場を表し、四角が作業者を表す。矢印は物の流れの方向を表す。また黒い小丸は材料や半完成品または完成品などの物を表す。これをペトリネットの用語で表すと

置き場を表す丸は	プレース(place)
作業者を表す四角は	トランジション(transition)
物の流れを表す矢印は	アーク(arc)
物を表す黒小丸は	マークまたはトークン(mark, token)
一度に流れる量を表す	
矢印の下の数字は	ウエイト(weight)

となる。なおウエイトが 1 のとき記入を省略されることもある。またトランジションが作業を開始し、1 単位の仕事をすることをトランジションが発火する(fire)という。

ペトリネットのプレースの中に書かれたマークは、トランジションの発火後にその数を変化させる。例えば上のペトリネットでは、プレース s に 1 個、プレース t に 2 個のマークが入っているが、トランジション C が発火するとその後はそれぞれのマークの数がプレース s は 1 個減って 0 個、プレース t は 1 つ増えて 3 個になる。このようにペトリネットの状態は時々刻々変わっていく。

３．カラーペトリネット

ペトリネットの中の 1 個のマークは､ただあるかないかの情報をもたらすに過ぎないが、もしこのマークの中に情報が書き込めるとすると、さらにペトリネットの適用範囲は広くなる。例えば、a×b＋c の計算が出来る回路は次のようなペトリネットで表せる。（図 5）

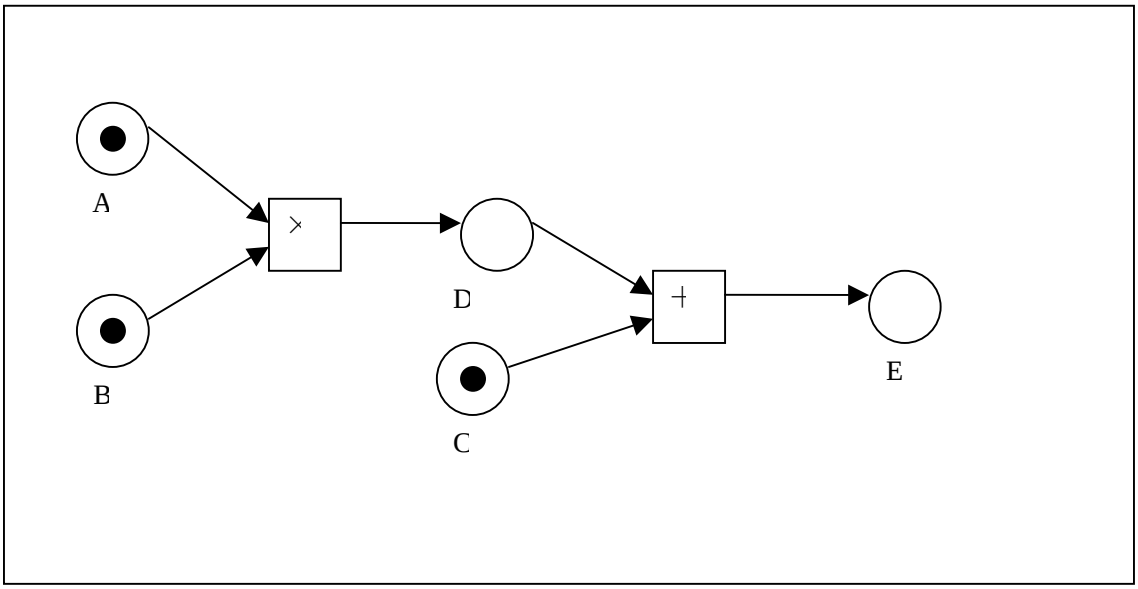


図 5：ペトリネットで回路を表す。

プレース A には数 a が書き込まれたマークが置かれる。プレース B には数 b が書き込まれたマークが、そしてプレース C には数 c が書き込まれたマークが置かれる。記号×のトランジションでは掛け算が行われ、結果をマークに書き込みプレース D へ送る。同様に記号＋のトランジションでは、プレース D のマークとプレース C のマークを一つづつとって、それらに書かれた数をもとに加算が行われ、結果を新しいマークに書き込みプレース E へ送る。3 つのプレース A、B、C にまた異なる数が書き込まれた新しいマークがそれぞれに入れられると、また次の計算が行われる。このようにマークに数が書き込めるという仮定をすることで、計算のような仕事もペトリネットで記述でき、このことはプログラムや演算回路までペトリネットで記述出来る可能性を示している。

マークに書き込む数を色の種類で表すと解釈すると、マーク一つ一つはある特定の色に塗られていると考えることも出来る。このことから、マークに数が書き込めるペトリネットをカラーペトリネットという。下流のプレースに送るマークの色は、あらかじめある規則で決められるとしておく。例えば、数としての掛け算や足し算で決まるとしておいてもよいし、あるいは上流プレース中のマークの色や数の組み合わせに対し、下流のプレースに送るマークの色を与える表を用意してもよい。またトランジションの発火条件も、上流のプレースの中のマークの色の組み合わせで指定することが出来る。例えば「上流のプレース全てに赤のマークがあったら発火せよ」といった複雑なものでもよい。これは発火規則と呼ばれ論理式などで表せる。更にペトリネットの適用範囲を広げるために、あるトランジションが発火した後、上流のプレース中のマークを取り去らなくてもよいとする。これは回路などでデータが記憶回路に入っているようなとき、そのデータを使って演算が行われても使われたデータが消えるわけではないようなことがあるためである。物と違って情報は使われても無くならないという性質がある。情報処理にペトリネットを適用するためには、場合によっては上流のマークは無くならないと考えたほうがよい。またマークを

除くか除かないかは、上流のマークの色の組み合わせにも依存するとする。例えば上流のプレースが 1、2、3 と番号をつけられていたとすると

	色の組み合わせ			発火後マークを		
プレース	1	2	3	1	2	3
	赤	赤	赤	取る	取る	取る
	白	赤	赤	取らない	取る	取る
	赤	白	赤	取る	取らない	取る
	赤	赤	白	取る	取る	取らない

図 6：トランジション発火後に上流プレースのマークを取り除くための組み合わせ(icp)

のような表が考えられる。（図 6）この表をみると、あるトランジションの上流に 3 つのプレースがあり、それらの中のマークが全て赤か、一つだけ白で後は赤という組み合わせになったときこのトランジションは発火し、そのとき上流の赤のマークだけを取り除くという規則がわかる。この表は、上流にあるマークの色（＝数）の各組み合わせに対し、発火後上流の発火に使ったマークを取り除くか否かを書いたものである。このような表を icp(input clear place)という。実際にはこの表全体を記述しなくても、「赤のときだけ取り除く」のようなルールあるいはこのルールを表す論理式を記述すればよい。同様にして下流のプレースにマークを送るとき、選択的に下流のあるプレースだけに限ることもできる。これは上流のマークの処理結果に応じ、マークの行き先を変えるようなときに使われる。具体的には、小包の仕分け作業（宛て先によって送り先を変える）をペトリネットで記述するようなときである。これも icp と同様に表や規則や論理式で記述できる。それは ocp(output set place)といい、下流のマークの置かれるプレースを指定するものである。

カラーペトリネットの例

コンピュータソフトはカラーペトリネットで記述することが出来る。例えば下図のようなフローチャートを考える。1 から 100 までの数を順に加えて 1＋2＋3＋・・・＋100 を求めるものである。

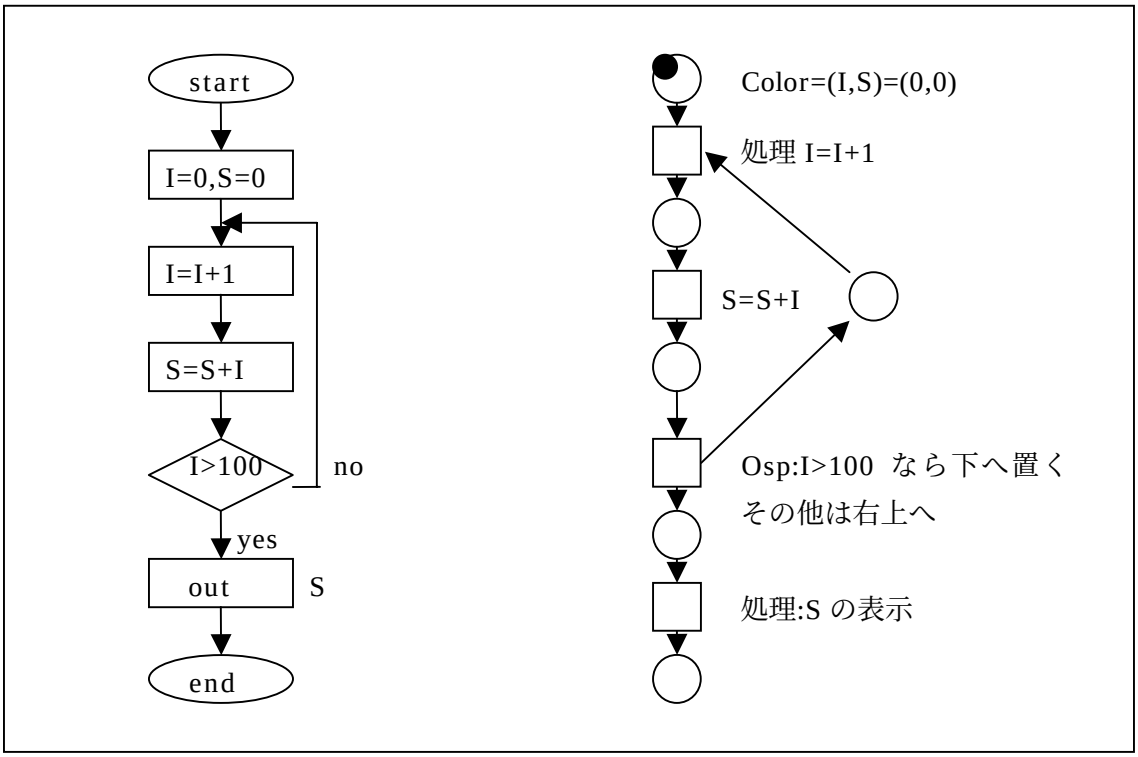


図 7：プログラムの流れをカラーペトリネットで表す。

これをカラーペトリネットで表すと図 7 の右の図のようになる。条件分岐のところでは osp の式に従ってマークは行き先を変えることになる。マークはこの図ではスタートを表すプレースのところにあるが、時間とともに下に降りてきてループのところを何回か回った後に終了を表す一番下のプレースに入る。これで計算終了である。

各トランジションでの

発火条件は「上流のプレースの一つにマークがあったとき」であり、

処理は「図中に示したものでマークの色を変えるのみ」であり、

icp は「マークのあるプレースからのみ除く」であり、

ocp は「分岐のトランジションでのものは図中に示されており、その他はマークを下流のプレースに置く」である。

フローチャートはただ一箇所でしか計算をしないため、それを表すペトリネット上を動くマークはただ一つである。

４．システム開発とペトリネット

ここで本論の目的である、システム仕様書の作成に話を戻したい。ここでいう「システム」とは、ハードウェアシステムまたは、ソフトウェアシステムのことを指す。仕様書とは、システムの論理的な機能を説明する文章である。通常システム開発を行う際に、人間がその仕様書の意味を解釈して、システムを作り上げる。また、出来上がったシステムと仕様書の内容が同一であるかどうかについても、人間がテストして判断をしている。簡単

なシステムではそれも可能であるが、複雑なシステムになると人間の判断ではチェックしきれなくなってくる。さらに、システムの開発中に仕様が変わることもよくあることであるが、あとから仕様書を実際のシステムに合わせるのも大変な作業となる。これらの作業を自動化するために、ペトリネットからソフトウェアシステムを開発し、その仕様書もペトリネットから自動で書き出し、自然言語の文章に出来ないかということを考えてみる。

５．ペトリネットからソフトウェアシステムを開発するための考察

まずペトリネットからソフトウェアシステムを作る方法を考える。例として、クイズゲームを考える。内容はゲームを開始すると問題が出題され、ユーザはそれに答えるというものである。答えをチェックし、一題終了すると次の問題を行うかを判断する。これに必要な材料として、次のものが挙げられる。

① Quiz data: questions and answers:問題に使う質問と答えが入っているファイル
② Petri net engine:ペトリネットを「実行する」プログラム
③ Petri net data:ペトリネットの仕様が入っているデータファイル
④ Screen:ユーザに出力する画面
⑤ Keyboard:ユーザからの入力を読むキーボード

これを図 8 で表すと次のようになる。

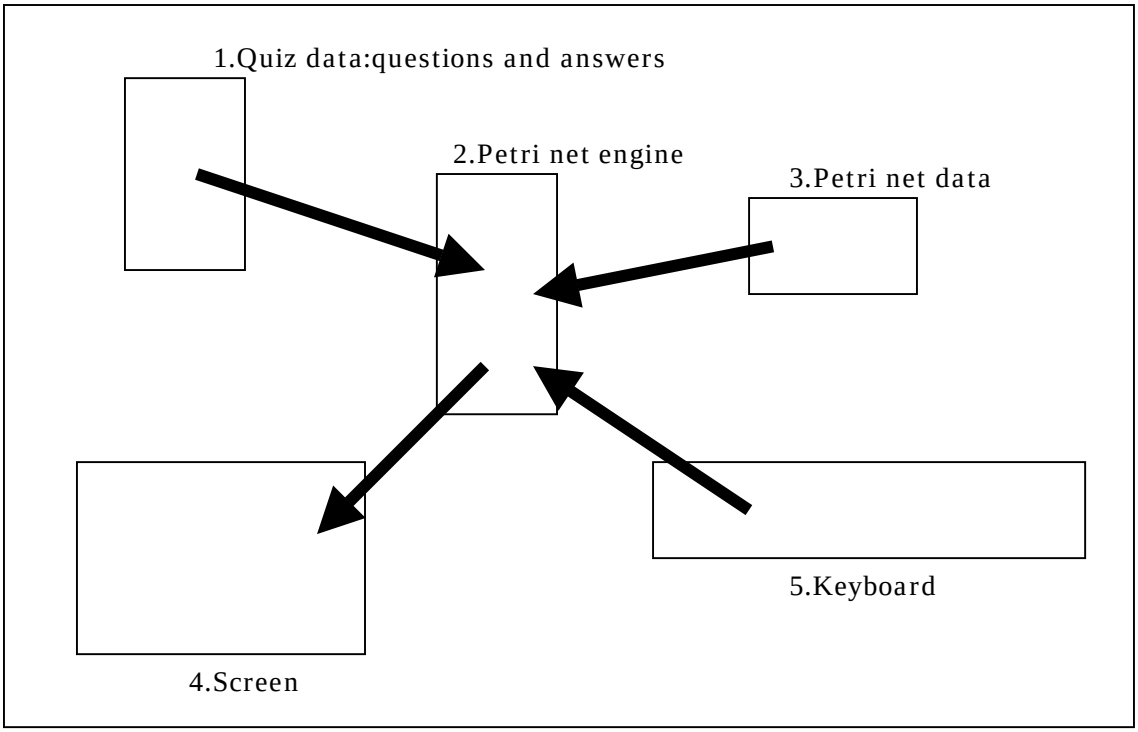


図 8：ペトリネットからソフトウェアの作るためのスケッチ

クイズゲームの流れを状態で考える

0:スタート
1:クイズ実行中
2:終了判断
3:終了

状態の流れをペトリネットで書いてみると、次のようになる。（図 9）

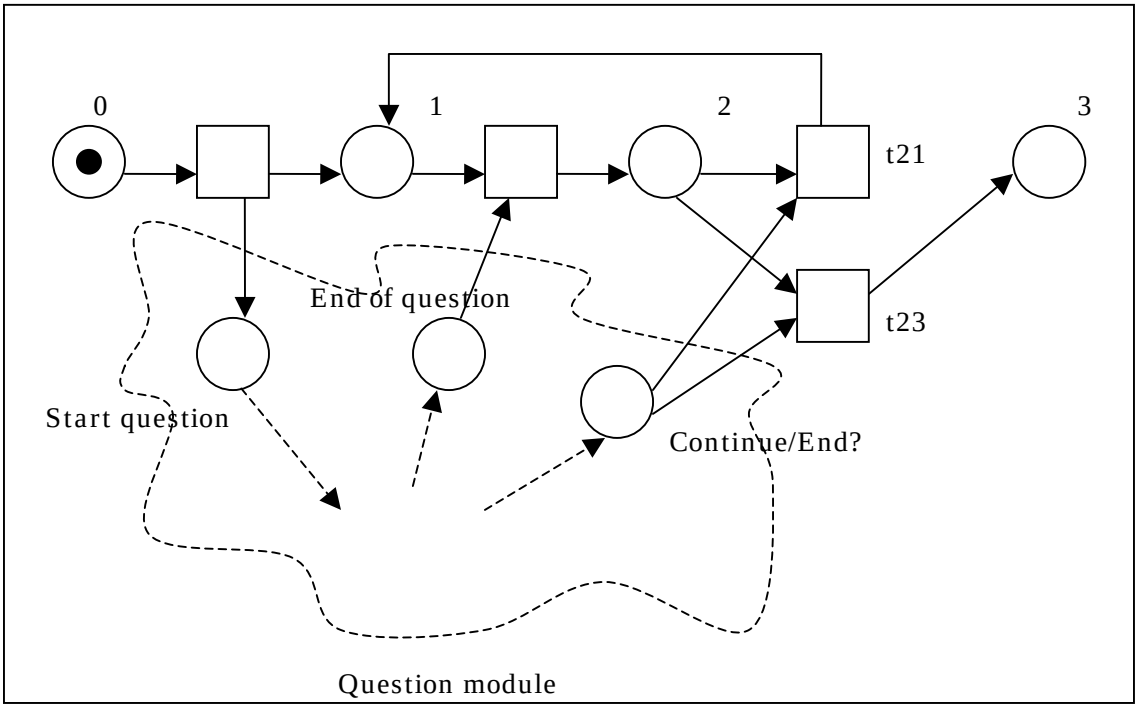


図 9：クイズゲームの流れをペトリネットで表す。

Question module の部分はクイズの問題を表示したり、ユーザの答えを処理したりするペトリネットである。Start question プレースにマークが入ると処理を始める。一つの問題が終われば、End of question プレースにマークが入る。Continue/End?プレースは、ユーザが次の問題に進みたいか、クイズを終了したいかを表す。ここで問題として、普通のペトリネットであれば、t21 と t23 の発火評価は難しくなることを考えなくてはいけない。一つのクイズ問題が終了して、状態 2 に移った時、Continue/End?プレースにマークが有る＝次の問題に進む、無い＝終了と考えると t23 は終了したい時に発火しない。t21 と t23 の発火評価順序によっては、次の問題に進みたい時に終了してしまう事になる。そこでペトリネットに拡張した考え方を取り入れる。

拡張 1：カラー（色）付きマーク
Continue/End?プレースを 2 つに分けることも考えられるが、ペトリネットの要素をあまり

増やさずに解決する方法としてマークにデータを持たせる。

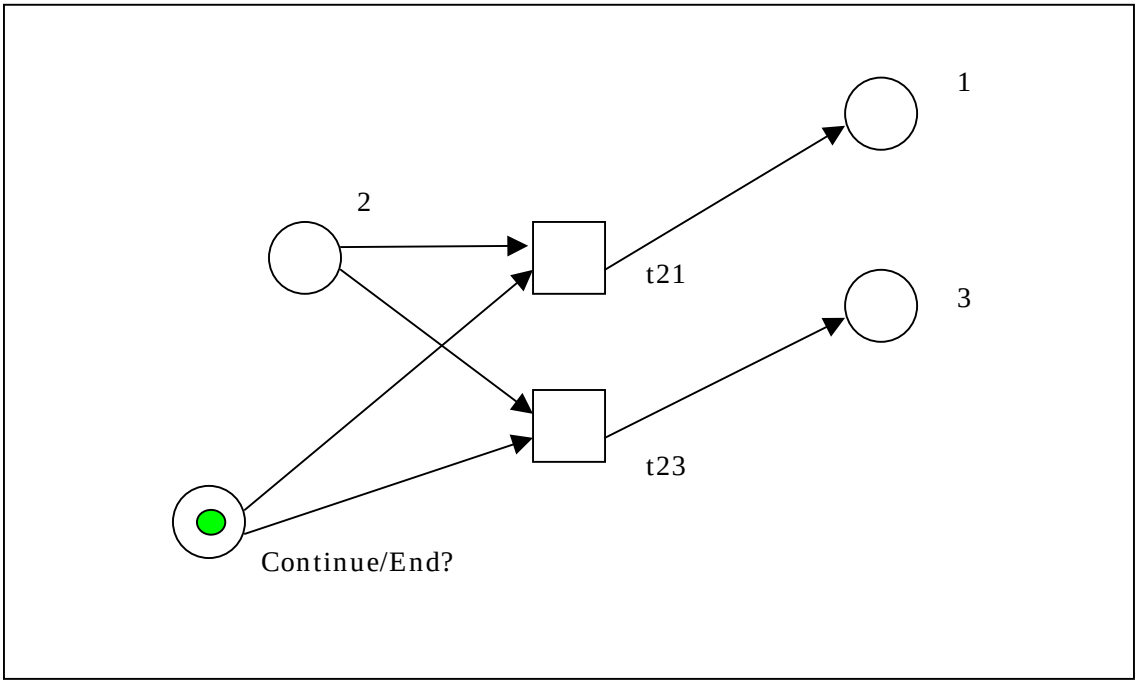


図 10：緑マーク＝次へ進む、赤マーク＝終了とすれば良い。

拡張 2：トランジションの発火評価と発火処理の記述
カラー付きマークを導入すれば、トランジションの発火評価と発火処理も拡張しなければならない。例えば、t21 は Continue/End?プレースに緑マーク、プレース 2 に任意のマークがある時に発火する、発火した時にプレース 2 からマークを一つ、Continue/End?プレースから緑マークを一つ消すという定義をしなければならない。

ここで Question module の中身を考えてみる。

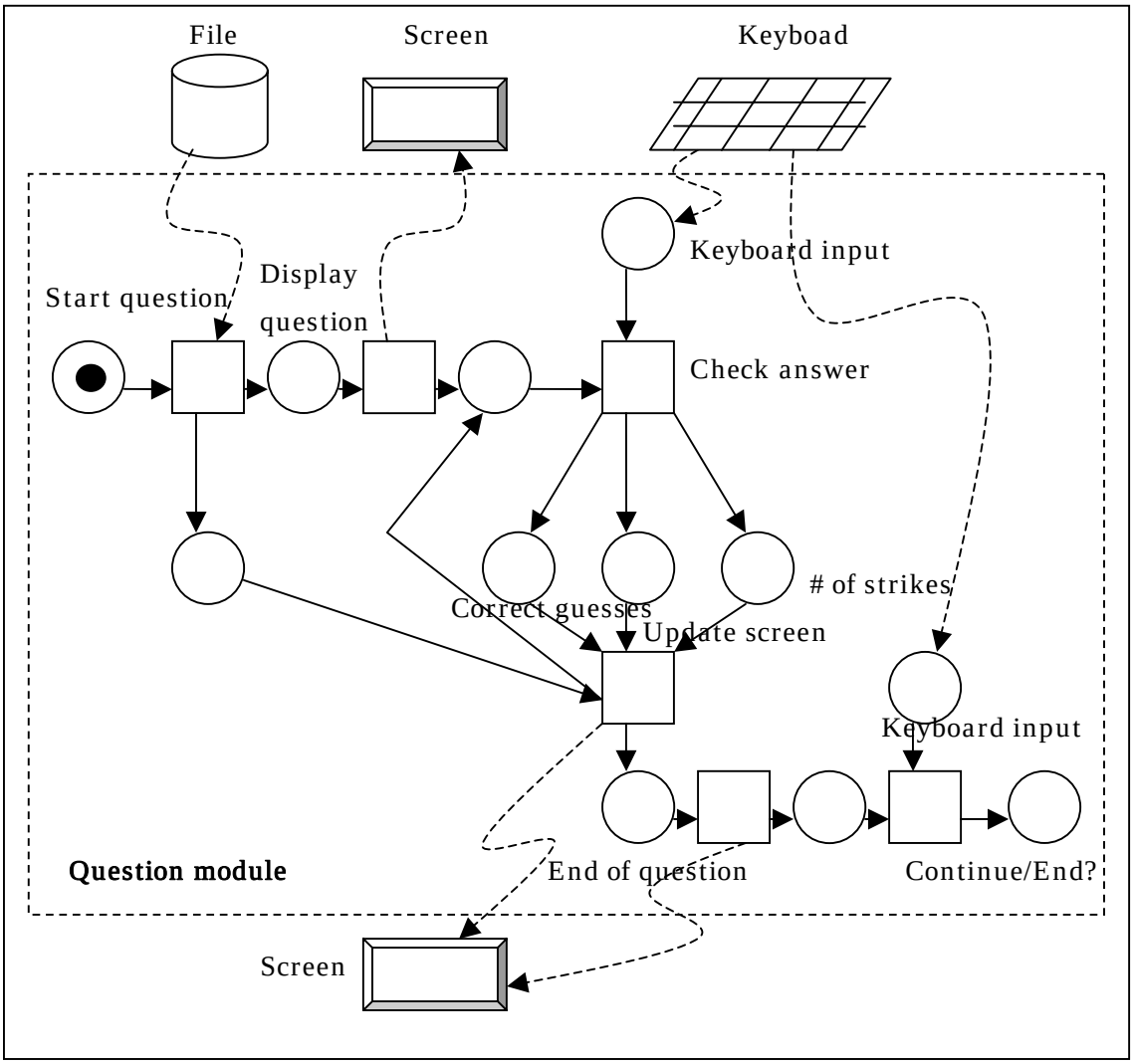


図 11：Question module の一つの考え方

すると解決しなければならない問題がいくつかある。まず、ペトリネットの「外」にある要素（ファイル、画面、キーボード）とのやり取りをどうするかである。

拡張 3：外部インターフェース

ペトリネットでプログラムを作成すると、ユーザ入力や画面出力のようなペトリネット外部とのインターフェースを定義しなければいけない。ここで次のものを考える：

トランジションワーク

発火可能なトランジションが実行できる「仕事」を次のタイミングで行う：

１）発火評価

２）発火可能な場合はワークを実行

３）入力・出力プレースの書き換えを行う

ワークの種類：ファイルの読み込み、書き込み

記述：　read(filename{,mark to house data});

write(filename, data);

ワークの種類：画面出力

記述：　print(data, x, y);

特殊プレース

キーボードバッファーに入力データがある場合、特殊な「キーボードプレース」に書き込むようにする。トランジションの発火評価処理をする度に、まずこのキーボードプレースへの書き込みを済ませてから行う。

さらにペトリネットをわかりやすくするための工夫を行う。

1）上記のようなペトリネット図を分かりやすくするために、外部インターフェース用の特殊プレースを少し違うスタイルで示すことにする。（例えば、2 重○や外○が点線）

2）矢印の線をあちこち書かなくてすむように、プレースのクローン（コピー）を作成出来るようにする。元のプレースとそのクローンは実は同じプレースであるのを忘れないようにクローンになっているプレースを別の色にする。

3）モジュールの入口（START プレース）と出口（END プレース）を一つずつにして 2 重○で印をつけ、モジュールをアイコン化する。モジュールに入るアークはモジュールの START プレースに入るアークと解釈する。同じように、モジュールから出るアークはモジュールの END プレースからでるアークの意味である。

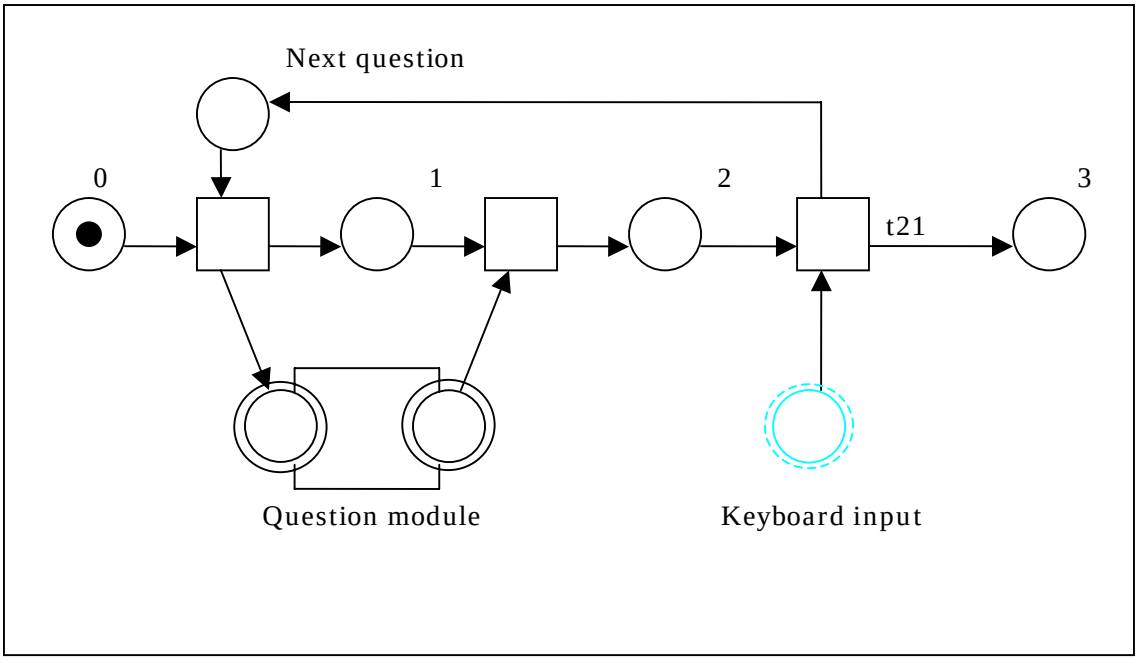


図 12：ペトリネットをわかりやすくするための工夫が必要。

これらの工夫を取り入れて Question module の中身を次のように作り直す。

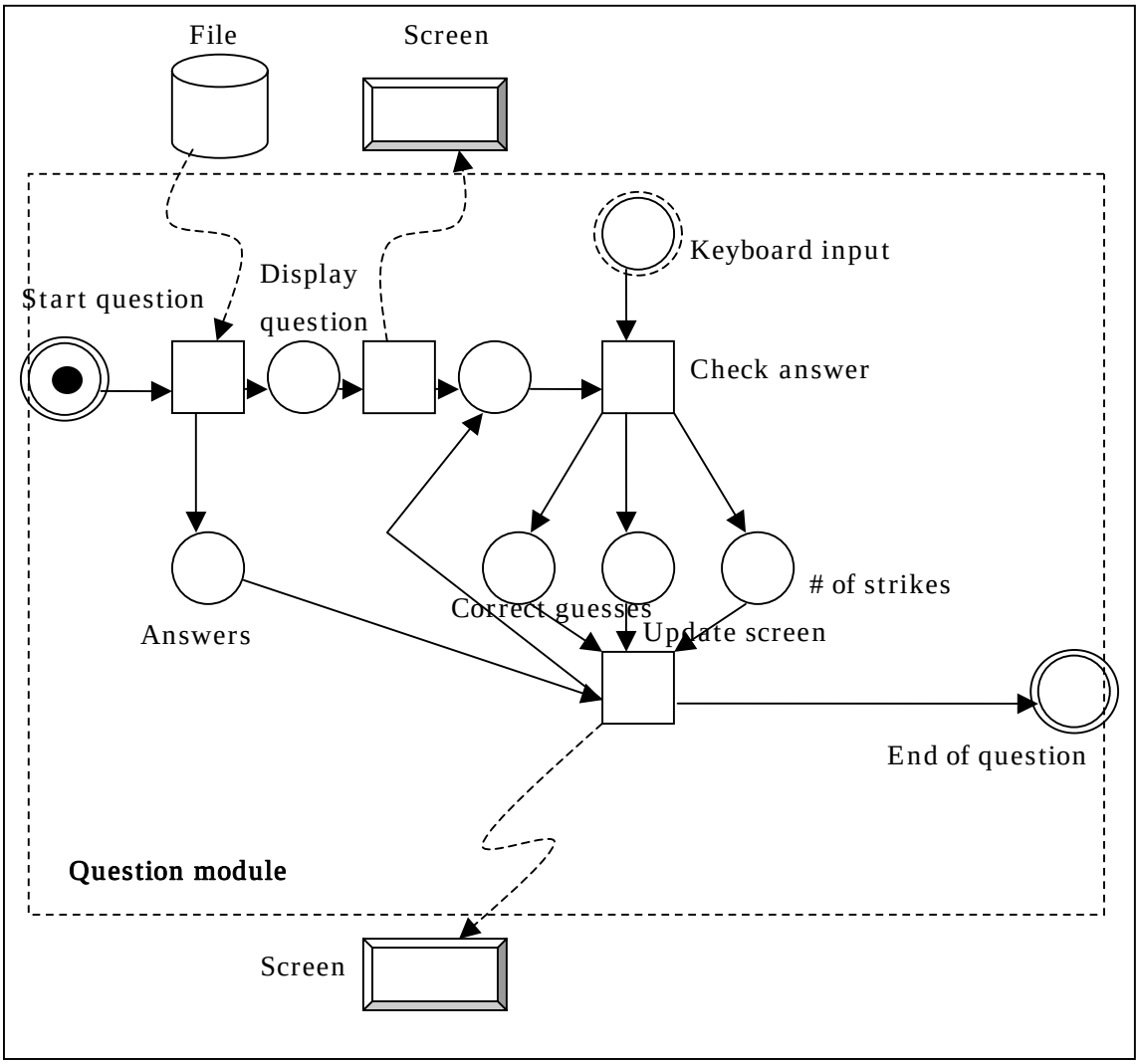


図 13：Question module をわかりやすく作り直す。

拡張 2（続き）

トランジション t1 を例にして、発火評価ルールと入力・出力プレースの書き換えルールに

ついて考えてみる。

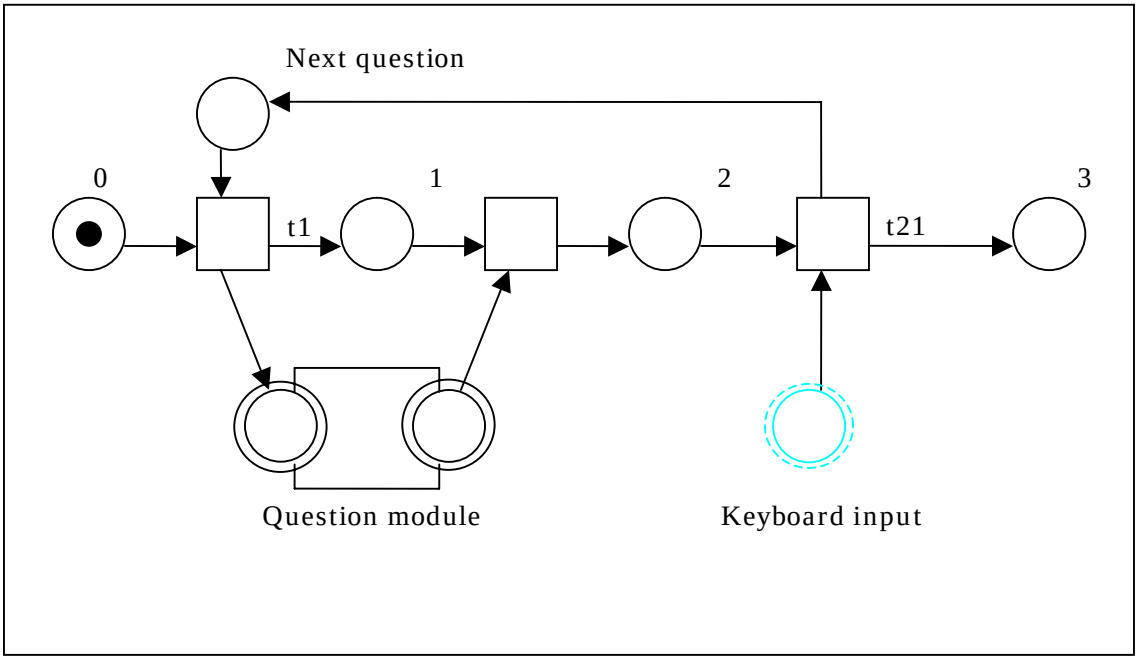


図 14：t1 の発火評価ルールについて考える。

入力プレース 0 にマークがある時は発火してほしいが、Next question プレースのことを考えなくてはいけない。最初の問題の時はそこにマークはない。マークがある時は、次に処理してほしい問題番号がデータ（色）として書き込まれていると仮定し、Question module（の START プレース）にそれと同じ色のマークを送り込みたい。発火処理の最後に入力プレースのマークを削除したい。

発火評価ルール

トランジションの発火評価ルール（発火条件）を記述する時に論理式が使えると便利。

t1 の場合：

発火評価ルール：　m(“0”)>0

のように書く。m( )関数は引数で指定する入力プレース（ここでは、名前指定）のマーク数を返すものとする。

ワーク

t1 は外部に対して仕事を行わないので、

ワーク：　なし

入力プレースのクリア(ICP)

上記の発火評価ルールが TRUE になり、t1 が発火した場合、入力プレースからどのように

マークを削除するかを記述する。このマーク削除処理を Input Clear Place(ICP)と呼ぶ。
t1 の場合：

ICP：　TRUE：ICP(“0”),ICP(“Next question”);

出力プレースのセット(OSP)
上記の発火評価ルールが TRUE になり、t1 が発火した場合、出力プレースへどのようにマークを入れていきたいかを記述する。このマーク生成処理を Output Set Place(OSP)と呼ぶ。t1 を考えるとき、Next question プレースにマークがあった場合、そのマークと同じ色のマークを生成し、Question module の START プレースに入れたい。Next question プレースにマークが無い場合、「1」のデータを持つマークを出力する。

OSP：　m(“Next question”)!=0：OSP(“Start question”,m(“Next question”).1);
　　TRUE：OSP(“Start Question”,1);

m(“Next question”).1 は Next question プレース内の 1 番目のマークのデータである。この例のように、複数の[条件]：[OSP 処理]ルールが記述された場合、上から順に見て、最初に TRUE になる条件の OSP 処理のみを行う。（ICP も同じように、複数のルールを記述しても良い）

ペトリネットエンジンを実現するには拡張があといくつか必要となるが、その前にペトリネットで作ったシステムの仕様化をどうするか考えてみる。

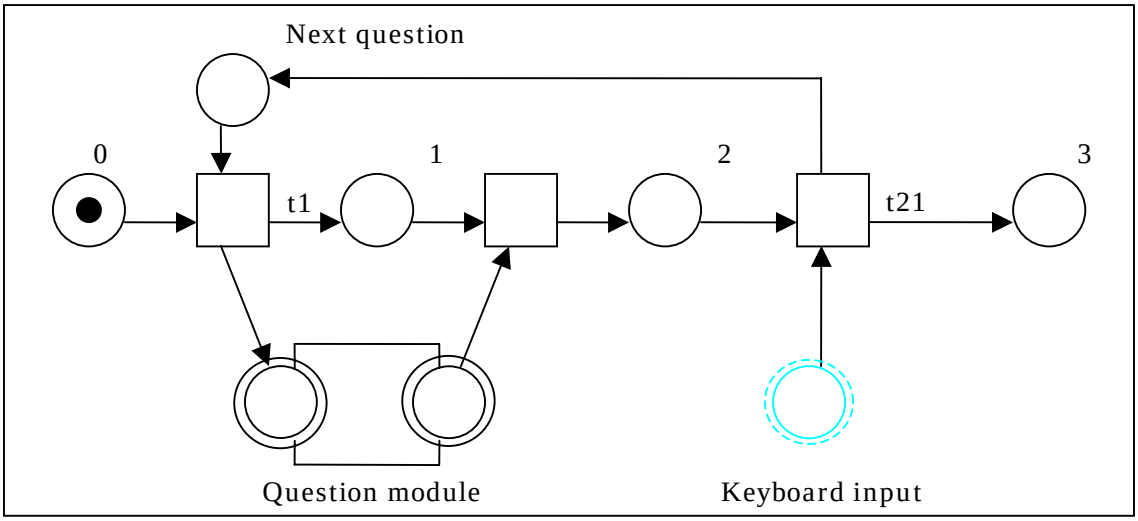


図 15：プログラムの流れ
クイズゲームの例は状態の流れで始めたが、外部インターフェースのプレースやプログラム変数の働きをする要素も加わり、全体の整理に気を付けないと、ペトリネットが大きく

なり何の動きをしているかがわからなくなる。そこで先に紹介したモジュールとクローンの概念を使って、プログラムの重要な「スケッチ」を作ることにする。
上の例のように、状態の流れでプログラムの主となる動作を設計する場合、状態プレースのクローンを一つのモジュールにまとめると、ペトリネットの実行中に現在の状態や状態の遷移が分かりやすくなる。
同じように、ペトリネットの外界とのインターフェースにあたる要素を「I/O モジュール」というモジュールにまとめることも動作のチェックで便利となる。
モジュールとクローンを使って、プログラムの様々な断面図を作りながら、システム仕様に出来るものを検討していきたい。

6．ペトリネットからシステム仕様書を作成するための考察

システムをペトリネットを用いて仕様化することを考える。これについてはアプローチの方法の一つとして、出来上がったシステムを他人に理解してもらうためには、何を伝えたいかということを考えてみる。ペトリネットをツールなどで実行するとそれがそのままシステムとなる。多くの場合、実行しているシステムを理解したい時、システムの状態を把握することが必要である。しかし大きなペトリネットが動いている場合、どこが状態の部分か容易には判断出来ない。そこでモジュールとクローンの概念を用いて状態を表すプレースを集めてみる。

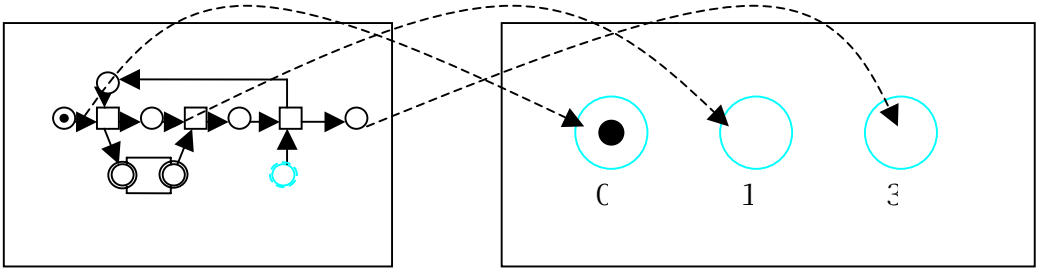


図 16：状態を表すプレースのクローンを作り、一つのモジュールに集める

クローンは元のプレースのコピーを別に作ったのではなく、元のプレースと同じ要素と考える。ペトリネットの断面図では 2 つのプレースのように見えるが、実際は 1 つである。元のプレースにマークが入ると、クローンにもマークが入っているようになる。このように、状態を表すプレースにマークが入っていることを確認し、システムの状態を把握出来るようになる。元のペトリネットを Logic module、状態を抜き出したものを Model module と呼ぶ。また Model module を元にしてシステムの状態を視覚的な直感で把握出来るようにしたものを Image module、ペトリネットと外部との入力や出力を行うプレースを集めたものを Input/Output(I/O) module と呼ぶ。ペトリネットで考えるシステムの仕様はこれら

4 つの部分集合から構成されると考えられる。今後はこれらの module を使ってペトリネットを用いたシステムの仕様化をより具体的に進めていきたい。

７．ペトリネットデザインツール(Petri Net Design Tool)の作成

ここからは、これまでの議論を具体化していくため、ペトリネットをデザインするツールの作成について考えていく。これはペトリネットをユーザーが視覚的にデザインし、なおかつ実行後の状態推移の検証やデザインしたペトリネットからのソフトウェアの自動作成ならびに自然言語の文章での仕様書の自動作成を目的としたものである。
これまでにも MS-DOS などの CUI(Character-baced User Interface)環境下で動作するものは考えられてきたが､今後は Windows などの GUI(Graphical User Interface)環境下で、GUI の操作性などより特長を生かしたものを考えていかなくてはならない。また昨今コンピュータのネットワーク化は当然となってきており、こうしたネットワーク化は資源の共有化と共に操作環境の同一化ももたらしてくれる。つまり、ネットワーク化された個々のコンピュータに依存することなく、同一の操作環境を提供出来ることを考慮しなくてはならない。これらのことから使用する言語は java 言語を選択した。

７．１．ツール作成において主に考慮すべき点

ツールの作成において主に考慮すべき点は、以下の 2 つである。

１）クラスの作り方（全てのオブジェクトを別々のクラスにするのが良いか、いくつかは同じクラスで良いのかの判断）

２）グラフィックスの制御方法

７．２．クラスの作り方

今後のソフトウェアの拡張性を左右する重要な項目である。結局は作成しながら、決めていく部分が多いが基本的には、グラフィックスとデータの部分を分けておきたい。これは今後他の側面からデータのみの再利用をうながすためである。
現状は以下のようになっている。

arc_obj_component.class（アークを描く）
const_h.class（変数の初期化）
elem_location_obj.class（デザインエリアの位置情報）
InputWindow.class（プレースやトランジションへのデータ入力用画面）
main_banner_component.class（バナーの部分）
main_mode_menu_marker_component.class（現状モードのマーカー）
main_wdw_obj.class（画面を制御するクラス）
mark_obj.class（マーク情報）
module_obj.class（位置情報）

net_elem_id_obj.class（状態変数）
petrinet_obj.class（デザインエリアに描画するときに使用）
place_obj.class（プレース情報）
place_obj_component.class（プレースを描く）
placeref_obj.class（プレース情報の初期化）
rel_location_obj.class（位置変数）
trans_fire_obj.class（トランジションを発火させる）
trans_obj.class（トランジション情報）
trans_obj_component.class（トランジションを描く）
transref_obj.class（トランジション情報の初期化）
web_pn_sys.class（全体を制御する最上位クラス）

７．３．グラフィックスの制御について

画面の構成は、「デザインエリア」、「メニュー」、「バナー」と大きく 3 つに分かれている。表示のスピードアップを図るため、当初 canvas クラスを使用していたが、component クラスへの変更や、プレースやトランジション部品配置後の画面の再描画をデザインエリアのみに限定した。

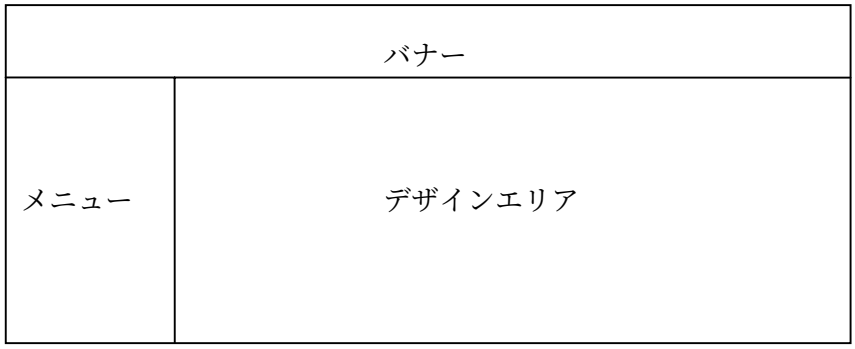
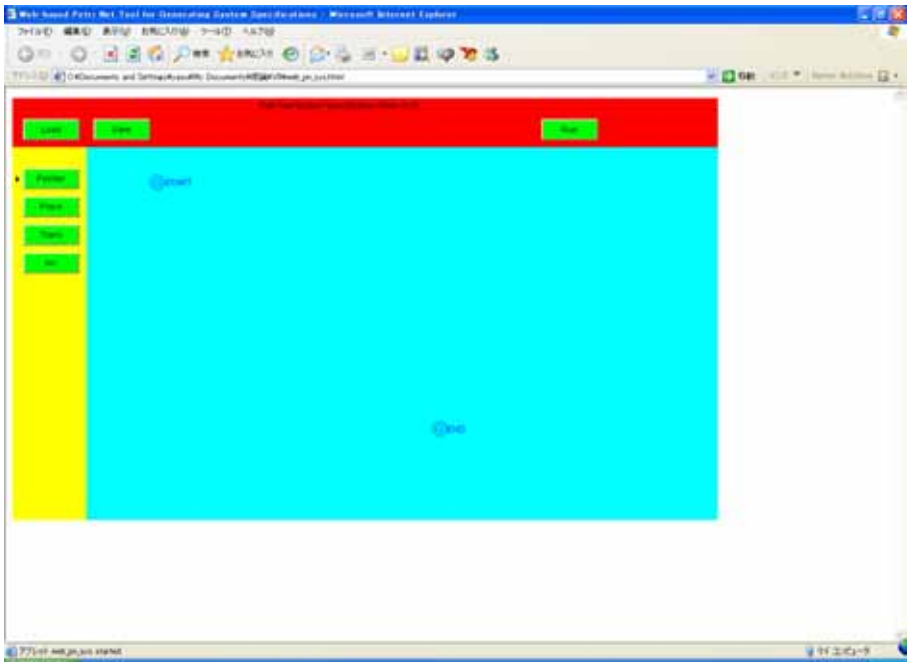


図 17：ツールの画面構成図

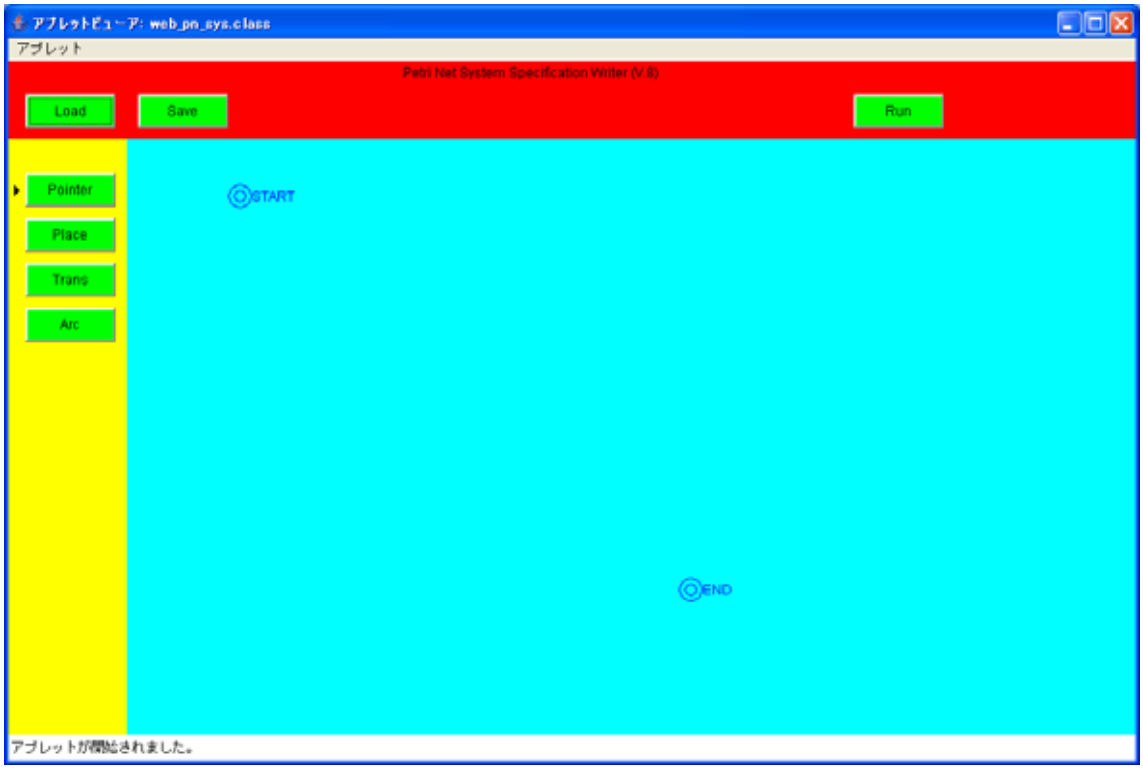
８．ペトリネットデザインツール(Petri Net Design Tool)の使用方法

ツールについてはいまだ開発途中となっている。現状ある機能を説明したい。
起動方法：web_pn_sys.html をダブルクリックで起動。



起動方法：コマンドプロンプトを表示させ、

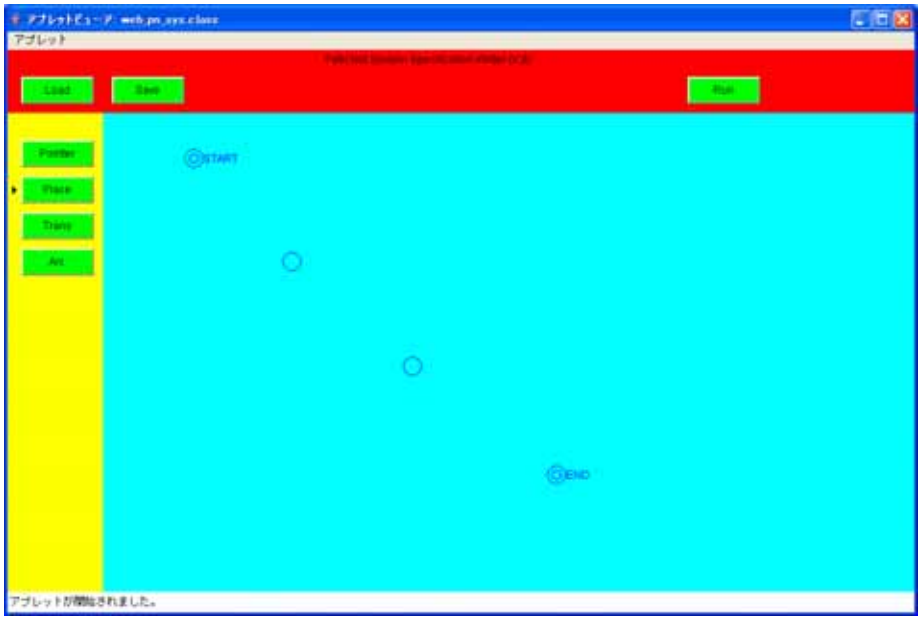
c:¥<path>¥appletviewer web_pn_sys.java（デバッグモード）



（初期画面）

Place mode：プレース(Place)を配置する。

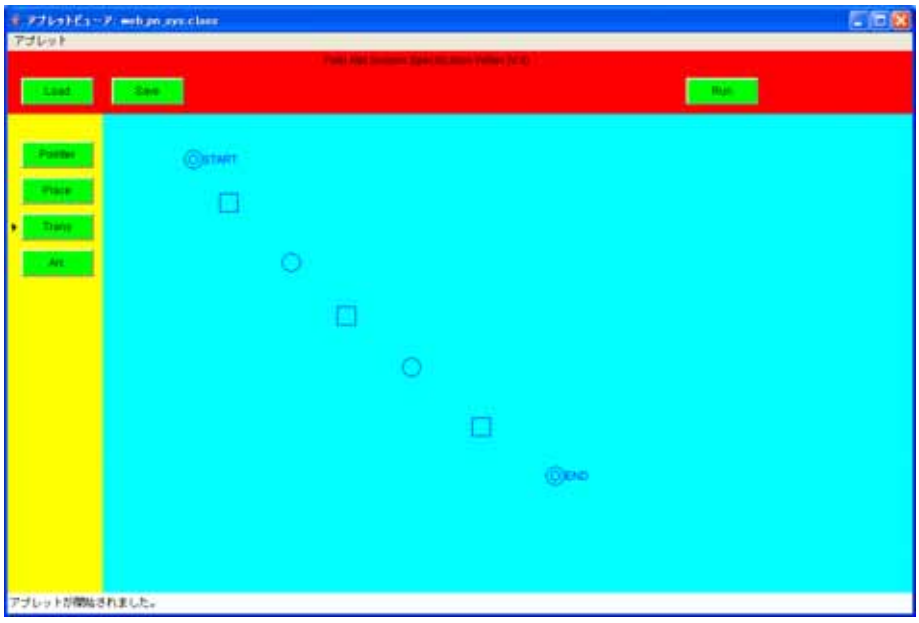
　左側メニューの Place ボタンを押下すると marker が移動するので、その状態で水色のデザインエリア内にクリックすると、Place が配置される。



（プレースを配置する）

Trans mode：トランジション(Transition)を配置する。

左側メニューの Transition ボタンを押下すると marker が移動するので、その状態で水色のデザインエリア内にクリックすると、Transition が配置される。

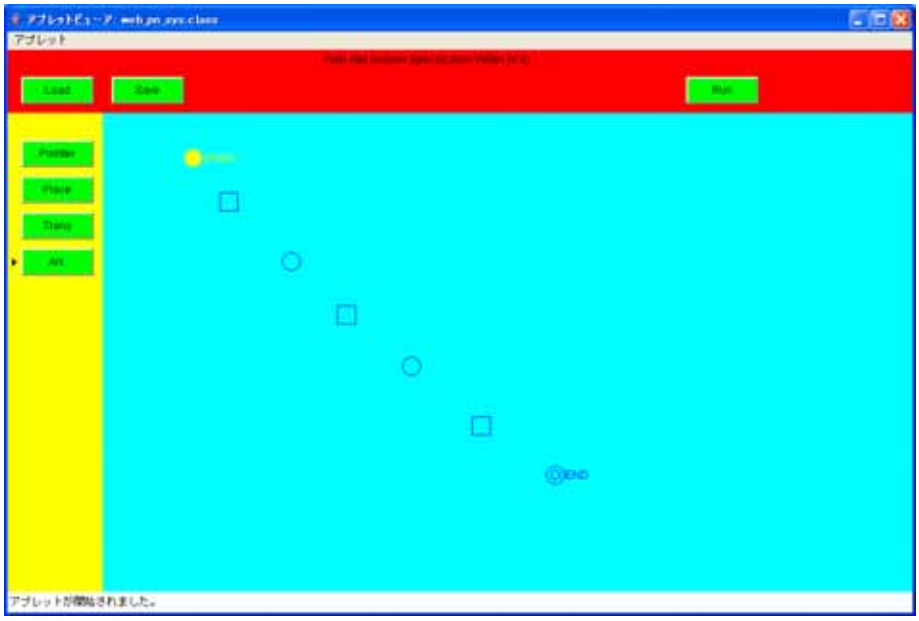


（トランジションを配置する）

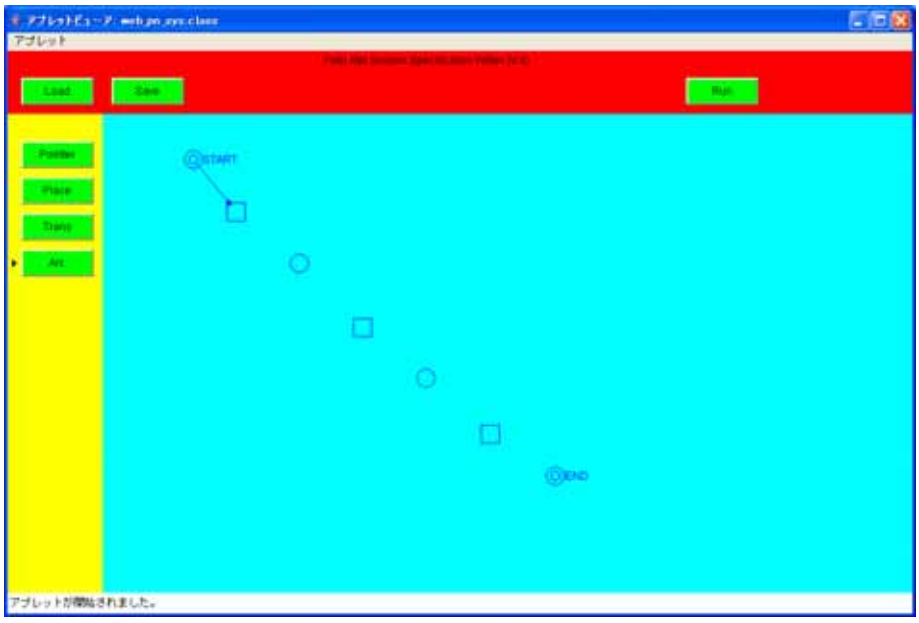
Arc mode：配置された Place→Trans、Trans→Place 間にアークを配置する。

左側メニューの Arc ボタンを押下すると marker が移動するので、その状態で水色のデザインエリア内に配置されたプレースをクリックすると、プレースが黄色に変化する。そ

の後つなげたいトランジションをクリックするとアークが表示される（プレース→トランジション）。トランジション→プレースは逆の操作。

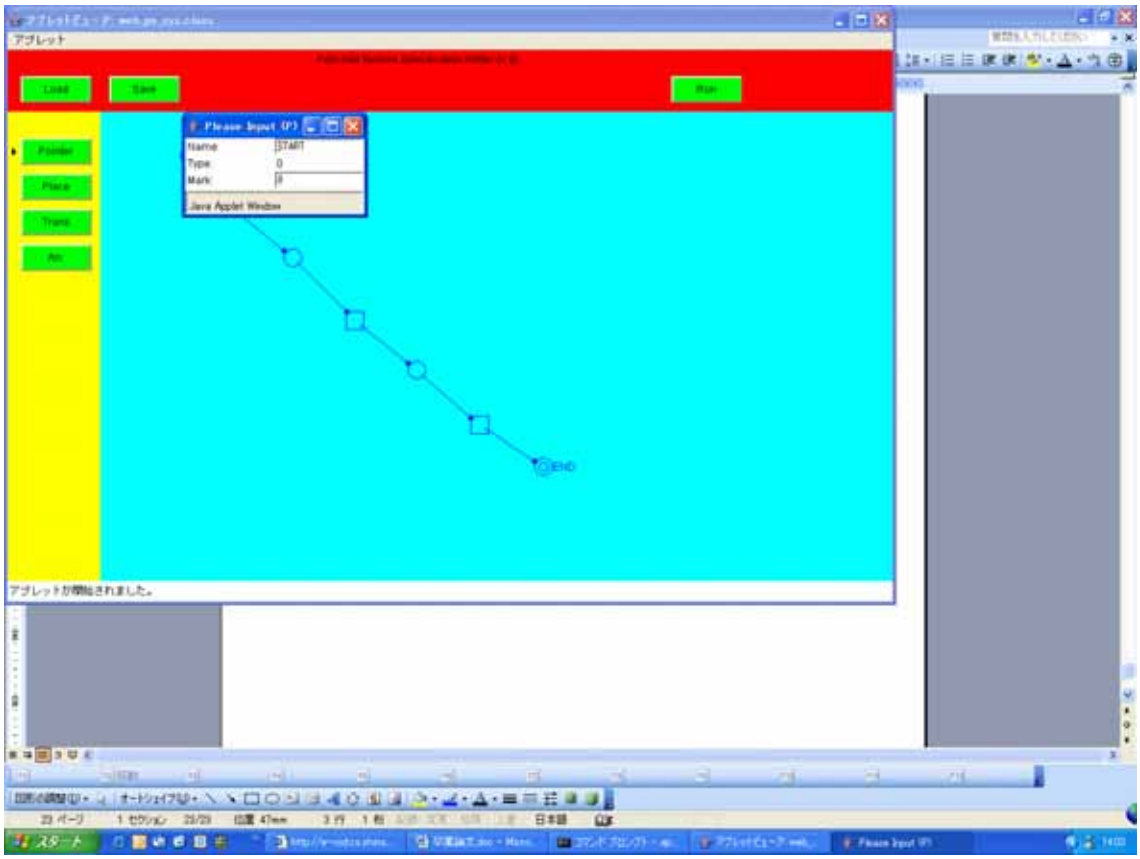


（プレースをクリックする）



（その後トランジションをクリックするとアークが表示される）

Pointer mode（初期値）：マウスポインターで配置されたプレース(Place)やトランジション(Transition)をダブルクリックすると情報入力用の Window が表示される。



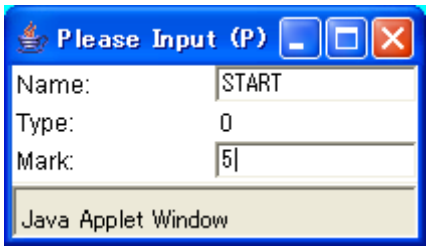
（情報入力用の Window が表示される）

情報入力用の Window には、以下を入力する。

Name：プレースやトランジションの名前

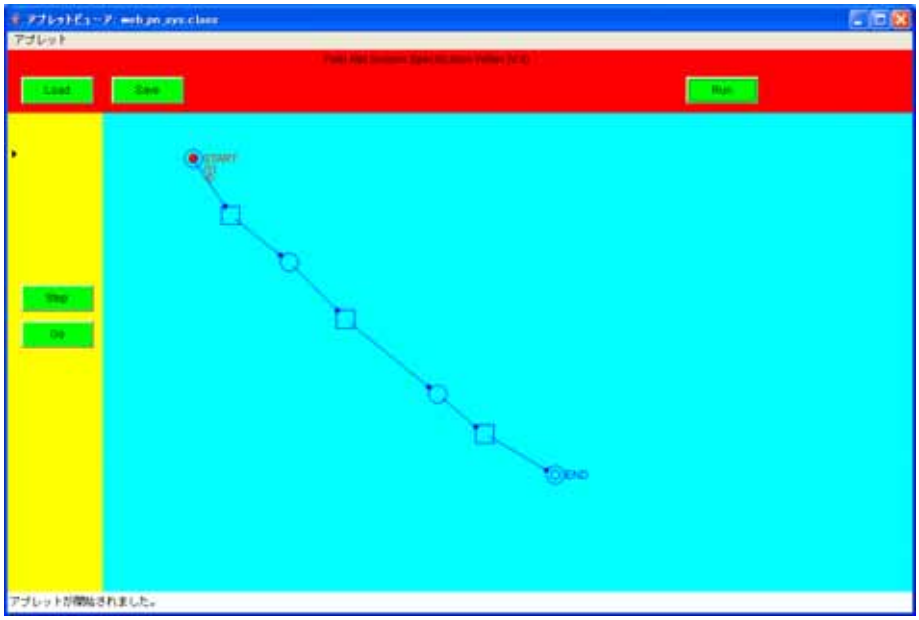
Mark：カラーペトリネットではマークは情報を持つので数値を入力する。

入力が完了したら、×ボタンで Window を閉じる。



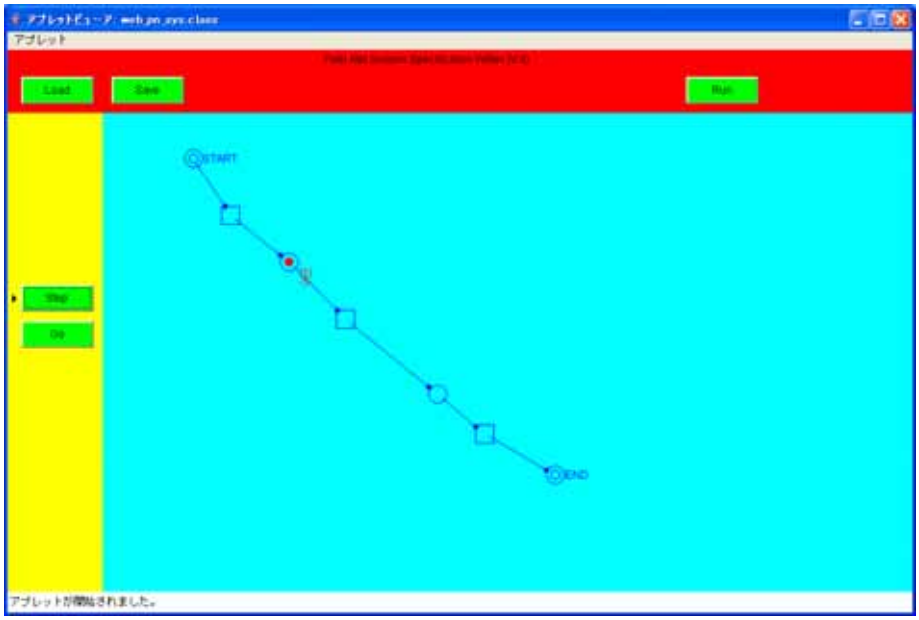
（情報入力用 Window の拡大）

Run mode：実行モードに切り替わり、Step や Go ボタンが表示される。



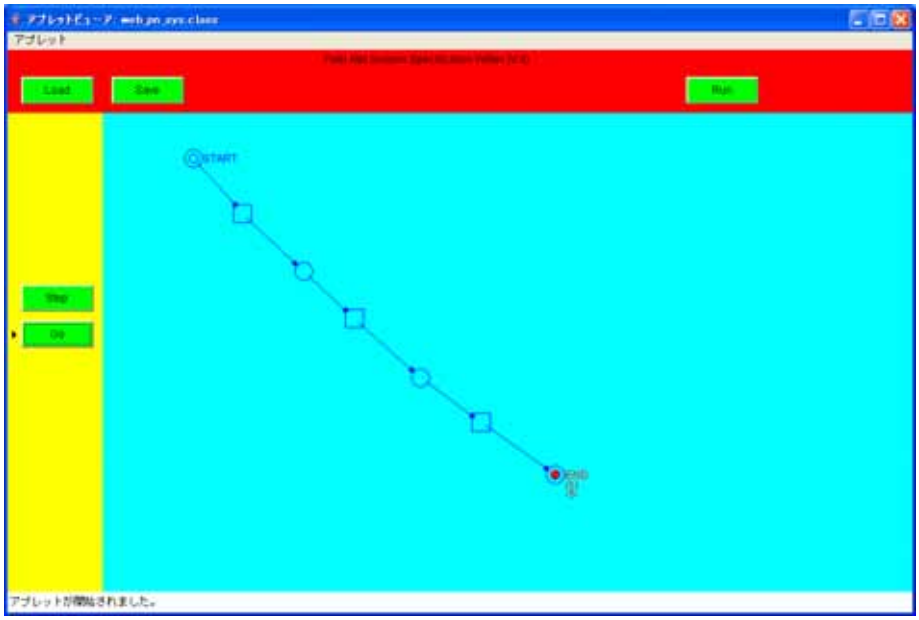
（実行モード）

Step（実行）：トランジションの index 毎に発火可能か確認していく。



（Step が一度行われた）

Go（実行）：トランジションが連続発火し、END プレースで終了となる。



（Go 実行後、END で終了となる）

９．結び

ツールの作成で目標としていることは、デザインしたペトリネットからソフトウェアや、その仕様書を自然言語の文章で自動生成するといった仕様化を行うことである。このためには、デザインしたペトリネットを更に詳しく解析していくために、先に説明したようなモジュールに分ける必要があるが、本論ではそこまで到らなかった。今後も、本研究を進めていき実際の仕様化にまで、ツールの開発を進めていきたい。また現状のツールについては、すぐに考えられる改良点は以下のものである。

Ⅰ. icp(input clear place)や osp(output set place)などトランジション発火ルールの設定。

Ⅱ．ファイルの出力や読み込みの機能追加。

Ⅲ．モジュールやクローン機能の追加。

Ⅳ．インターフェースにあたる要素を「I/O モジュール」にまとめる。

また、本ツールは web 上で使用することを前提としているので、実用化すればネットワーク上で共有化出来、より強力なツールとなる。

ペトリネットは、システムをモデル化するために主として用いられてきた。例えば、病院情報システム開発などへの利用も考えてみる。システム開発にあたっての問題点として、病院の組織化と情報提供の手続きの複雑さ、また大量の医療に関するデータや管理に関するデータが存在することがある。病院におけるこのような処理は、多種多様な方法で、相互連結したり相互依存したりしているから、その複雑さの要求に対し、活動機能とデータの両方の全体的な概観を与える形式的記述を作るのは難しい。複雑なソフトウェアシステムの設計や開発に用いられてきた多くの形式的な仕様の方法があるが、それらのほとんどはソフトウェア工学に基礎をおくソフトウェアツールである。しかし、それらにはいくつ

かの問題がある。例えば、ある特別なシステムに制限されているとか、情報の条件的な同期が考慮できないとか、大規模なものが扱えないといったものである。ペトリネットを用いると、このようなソフトウェアの欠点を補ったシステムの構築が可能になることが期待される。

現代社会において、社会システムはより複雑化してきており、コンピュータのハードウェア・ソフトウェアに限らず、論理的思考によりシミュレーションを行いより正確にその動きを予測し、問題解決の糸口にしようという要求が高まってきている（人口増加・減少の動向や、地球温暖化の動向→オゾン層の破壊・二酸化炭素の排出等)。こうした複雑な問題を解決するための論理的思考方法（ツール）の実用化はより注目されることと思われる。

References

[1]Pauline N.Kawamoto 講義ノート 2003/10－2005/03

[2]Pauline N.Kawamoto(1996)

Applications of Petri Net Theory in the Design of Hardware/Software Systems

[3]Pauline N.Kawamoto,Munehisa Sakamoto,Yasushi Fuwa,Yatsuka Nakamura(1996)

A PETRI NET ENVIRONMENT FOR DEVELOPING ROBOT CONTROLLERS

[4]信州大学工学部情報工学科基礎研究室ホームページ

「ペトリネットおよび LC ペトリネットについて」

http://markun.cs.shinshu-u.ac.jp/kiso/projects/petrinet/pds/LCPN/index-j.html

アクセス日時：2005.01.02 19:00

[5]椎塚久雄(1992)「実例ペトリネット」、コロナ社

[6]Java World for Beginners （㈱IDG ジャパン）2004 年

[7]Java トレーニングブック（㈱ソーテック）2003 年

[8]Java アルゴリズム＋データ構造完全制覇（㈱技術評論者）2002 年

Appendix

ペトリネットデザインツールは、信州大学工学部のカワモト・ポーリン・ナオミ助教授と共同開発を行った。以下はツールを構成しているソースである。

<web_pn_sys.java>

最上位のクラス

```
/* Web-based Petri Net Tool for Generating System Specifications */

<APPLET CODE="web_pn_sys.class" WIDTH=1000 HEIGHT=600>
</APPLET>

*/
/*-------------------------------------------*/

import java.applet.Applet;

import java.awt.*;
import java.awt.event.*;
import java.util.*;

/*---------------*/
/* WebPDS Applet */
/*---------------*/
public class web_pn_sys extends Applet
{
/* Version */
   public static String  version = "Petri Net System Specification Writer (V.8)";

/* Local Petri net object */
   public static petrinet_obj        local_petrinet = new petrinet_obj();
   public static place_obj           local_place = new place_obj();
   public static trans_obj           local_trans = new trans_obj();

/* Windows メイン画面*/
   public static main_wdw_obj      main_wdw_ptr;
```

```
/*-------------------------------------------------*/
/* Method: init_sys( )                             */
/*                                                 */
/* Function:   To initialize the web_pn_sys objects.   */
/*                                                 */
/* Expects:    Nothing                             */
/*                                                 */
/* Returns:    const_h.SUCCESS, operation successful   */
/*             const_h.ERROR, error encountered        */
/*-------------------------------------------------*/
   public int init_sys( )
   {
      int      status;

/* Build trivial Petri net. */
      if( (status = local_petrinet.build_trivial_petrinet( ))
         != const_h.SUCCESS )
      {
         return const_h.ERROR;
      }

      return const_h.SUCCESS;
   }

/*---------------------------------------------------*/
/* Method: update( Graphics g )                      */
/*     (override Applet update( ))                   */
/*                                                   */
/* Function:   To suppress repainting background.    */
/*---------------------------------------------------*/
   public void update( Graphics g )
   {
      paint( g );
   }

/*-----------------------------------------*/
```

```
/* Method: init( )                                  */
/*    (override Applet init( ))                     */
/*                                                     */
/* Function:    To intialize the applet         */
/*     display and local Petri net object.    */
/*-----------------------------------------*/
   public void init( )
   {
      int       status;

/*----------------------------*/
/* Step 1: Initialize applet. アプレットの初期化*/
/*----------------------------*/
/* Initialize applet container. */
       setLayout( null );
       setBackground( new Color(0,0,255) );
       Dimension applet_dimensions = getSize( );

/* Initialize web_pn_sys. */
       if( (status = init_sys( )) != const_h.SUCCESS )
       {
          System.out.println( "Error:    init_sys" );
       }

/* Create main window. */
       main_wdw_ptr = new main_wdw_obj( applet_dimensions );

/* Add main window to applet. */
       add( main_wdw_ptr );

/* Draw Petri net objects. */
       main_wdw_ptr.disp_net_obj( );
   } /* end of init */
}
```

<main_wdw_obj.java>

画面を制御するクラスの集まり。

```
/*************************************************************/
/*                          MAIN_WDW_OBJ.JAVA                */
/*   This file contains the class definition for the         */
/*   main window elements of the web_pn_sys applet.          */
/*   Written by:   Pauline N. Kawamoto (Shinshu University) */
/*************************************************************/

/*-----------------------------------------------------------*/
/* Edit History                                              */
/*-----------------------------------------------------------*/
/* [5/18/2004] - Quit packages.                              */
/* [1/21/2004] - File created.                               */
/*-----------------------------------------------------------*/

import      java.awt.*;
import      java.awt.event.*;
import java.util.*;

/*-------------------*/
/* Main Window Class     バナー*/
/*-------------------*/
class main_banner_component extends Component
{
   int         banner_width, banner_height;

   public main_banner_component( int width, int height )
   {
      banner_width = width;
      banner_height = height;
      setSize( banner_width, banner_height );
   }

   public void paint( Graphics g )
   {
      g.setColor( new Color( 255, 0, 0 ) );
```

```
        g.fillRect( 0, 0, banner_width, banner_height );
        g.setColor( new Color( 0,0,0 ) );
        g.drawString( web_pn_sys.version, 350, 15 );
    }
}
/*メニュー*/
class main_mode_menu_marker_component extends Component
{
    public main_mode_menu_marker_component( )
    {
        setSize( 5, 10 );
    }

    public void paint( Graphics g )
    {
        int x_coords[] = { 0, 5, 0 };
        int y_coords[] = { 0, 5, 10 };

        g.fillPolygon( x_coords, y_coords, 3 );
    }
}

class place_obj_component extends Component
{
    int k;
    place_objplace_obj_ptr;
    mark_obj                mark_obj_ptr = new mark_obj();
    boolean   highlight;

    public place_obj_component( place_obj place_obj_ref )
    {
        /*setSize( 21, 21 );*/
        setSize( 60, 60 );

/* Record place object being drawn. */
        place_obj_ptr = place_obj_ref;
```

```
    } /* end of place_obj_component constructor */

    public void paint( Graphics g )
    {
        place_obj             temp_place_obj_ptr;

/* Check place attributes. */
        if( highlight == const_h.ON )
        {
            g.setColor( new Color(255,255,0) );
            g.fillOval( 0, 0, 20, 20 );
            g.drawString(place_obj_ptr.name,21,15);
        }
        else if( ((place_obj_ptr.type & const_h.START_PLACE) != 0)
            | | ((place_obj_ptr.type & const_h.END_PLACE) != 0) )
        {
            g.setColor( new Color(0,0,255) );
            g.drawOval( 0, 0, 20, 20 );
            g.drawOval( 5, 5, 10, 10 );
            g.drawString(place_obj_ptr.name,21,15);
/*start [12/23/2004] by kaneko*/
            if(place_obj_ptr.mark != 0)
            {
               g.setColor( new Color(0,0,255) );
               g.drawOval( 0, 0, 20, 20 );
               g.setColor( new Color(255,0,0) );
               g.fillOval( 5, 5, 10, 10 );
               g.drawString(place_obj_ptr.name,21,15);
               g.drawString("("+String.valueOf(place_obj_ptr.mark)+")",21,26);
                  for(k=0;k<place_obj_ptr.mark_lnklst.size();k++)
                  {
                     mark_obj_ptr=(mark_obj)place_obj_ptr.mark_lnklst.get(k);
                  g.drawString("-"+String.valueOf(mark_obj_ptr.number)+"-",21+10*k,36);
                  }
            }
        }
```

```
/*end [12/23/2004] by kaneko*/
/*start [11/4/2004] by kaneko*/
      else if(place_obj_ptr.mark != 0)
      {
         g.setColor( new Color(0,0,255) );
         g.drawOval( 0, 0, 20, 20 );
         g.setColor( new Color(255,0,0) );
         g.fillOval( 5, 5, 10, 10 );
         g.drawString(place_obj_ptr.name,21,15);
         g.drawString("("+String.valueOf(place_obj_ptr.mark)+")",21,26);
         for(k=0;k<place_obj_ptr.mark_lnklst.size();k++)
         {
               mark_obj_ptr=(mark_obj)place_obj_ptr.mark_lnklst.get(k);
               g.drawString("-"+String.valueOf(mark_obj_ptr.number)+"-",21+10*k,36);
         }
      }
/*end [11/4/2004] by kaneko*/
      else
      {
         g.setColor( new Color(0,0,255) );
         g.drawOval( 0, 0, 20, 20 );
         g.drawString(place_obj_ptr.name,21,15);
      }
   } /* end of place_obj_component paint */
} /* end of place_obj_component class */

/*-----------------------*/
/* [1/12/2004] by kaneko */
/*-----------------------*/
class trans_obj_component extends Component
{
   trans_obj               trans_obj_ptr;
   boolean   highlight;

   trans_obj_component( trans_obj trans_obj_ref )
   {
```

```
        setSize( 60, 21 );

/* Record transition object being drawn. */
        trans_obj_ptr = trans_obj_ref;
    }

    public void paint( Graphics g )
    {
/* Check transition attributes. */
        if( highlight == const_h.ON )
        {
            g.setColor( new Color(255,255,0) );
            g.fillRect( 0, 0, 20, 20 );
            g.drawString(trans_obj_ptr.name,21,15);
        }
        else
        {
            g.setColor( new Color(0,0,255) );
            g.drawRect( 0, 0, 20, 20 );
            g.drawString(trans_obj_ptr.name,21,15);
        }
    } /* end of paint for trans_obj_component */
} /* end of trans_obj_component class */

class arc_obj_component extends Component
{
    int         width, height, arc_length;
    rel_location_obj        start_coord = new rel_location_obj( ),
                                        end_coord = new rel_location_obj( ),
                                        origin_coord = new rel_location_obj( );

    arc_obj_component( place_obj src_place_ptr, trans_obj dest_trans_ptr )
    {
        int       dx, dy, qx_for_src_place, qy_for_src_place,
                            qx_for_dest_trans, qy_for_dest_trans;
```

```
/* Set start/end coordinates. */
      start_coord.rel_x = src_place_ptr.location.coord.rel_x;
      start_coord.rel_y = src_place_ptr.location.coord.rel_y;

      end_coord.rel_x = dest_trans_ptr.location.coord.rel_x;
      end_coord.rel_y = dest_trans_ptr.location.coord.rel_y;

/* PNK - Debug */
      System.out.println( "Constructed a pt arc, Start coord: (" + start_coord.rel_x + ", " + start_coord.rel_y
+ "), End coord: (" + end_coord.rel_x + ", " + end_coord.rel_y + ")"    );

/* Set width, height, and origin. */
      init_arc_obj_component( );

/* Trim arc. */
      dx = width - 20;
      dy = height - 20;
      arc_length = (int)(Math.sqrt(Math.pow(width,2)+Math.pow(height,2)));

      qx_for_src_place = 0;
      qy_for_src_place = 0;
      qx_for_dest_trans = 0;
      qy_for_dest_trans = 0;

      if( arc_length != 0 )
      {
         qx_for_src_place = (10 * dx)/arc_length;
         qy_for_src_place = (10 * dy)/arc_length;

         if( dx == 0 )
         {
            qx_for_dest_trans = 0;
            qy_for_dest_trans = 10;
         }
         else if( dy == 0 )
         {
```

```
            qx_for_dest_trans = 10;
            qy_for_dest_trans = 0;
        }
        else
        {
            if( dy/dx < 1 )
            {
                qx_for_dest_trans = 10;
                qy_for_dest_trans = (10 * dy)/dx;
            }
            else
            {
                qx_for_dest_trans = (10 * dx)/dy;
                qy_for_dest_trans = 10;
            }
        }
    }

System.out.println( "qx_for_src: " + qx_for_src_place );
System.out.println( "qy_for_src: " + qy_for_src_place );
System.out.println( "qx_for_dest: " + qx_for_dest_trans );
System.out.println( "qy_for_dest: " + qy_for_dest_trans );

    if( start_coord.rel_x > end_coord.rel_x )
    {
        start_coord.rel_x = start_coord.rel_x - qx_for_src_place;
        end_coord.rel_x = end_coord.rel_x + qx_for_dest_trans;
    }
    else
    {
        start_coord.rel_x = start_coord.rel_x + qx_for_src_place;
        end_coord.rel_x = end_coord.rel_x - qx_for_dest_trans;
    }

    if( start_coord.rel_y > end_coord.rel_y )
    {
```

```
        start_coord.rel_y = start_coord.rel_y - qy_for_src_place;
        end_coord.rel_y = end_coord.rel_y + qy_for_dest_trans;
      }
      else
      {
        start_coord.rel_y = start_coord.rel_y + qy_for_src_place;
        end_coord.rel_y = end_coord.rel_y - qy_for_dest_trans;
      }


/* PNK - Debug */
      System.out.println( "Trimmed start coord: (" + start_coord.rel_x + ", " + start_coord.rel_y
+ ")" );
      System.out.println( "Trimmed end coord: (" + end_coord.rel_x + ", " + end_coord.rel_y
+ ")" );
      System.out.println( "Origin: (" + origin_coord.rel_x + ", " + origin_coord.rel_y + ")"    );
   } /* end of arc_obj_component constructor for pt-arcs */

   arc_obj_component( trans_obj src_trans_ptr, place_obj dest_place_ptr )
   {
/*start kaneko 20040617*/
      int      dx, dy, qx_for_dest_place, qy_for_dest_place,
                        qx_for_src_trans, qy_for_src_trans;

/* Set start/end coordinates. */
      start_coord.rel_x = src_trans_ptr.location.coord.rel_x;
      start_coord.rel_y = src_trans_ptr.location.coord.rel_y;

      end_coord.rel_x = dest_place_ptr.location.coord.rel_x;
      end_coord.rel_y = dest_place_ptr.location.coord.rel_y;

/* PNK - Debug */
      System.out.println( "Constructed a tp arc, Start coord: (" + start_coord.rel_x + ", " + start_coord.rel_y
+ "), End coord: (" + end_coord.rel_x + ", " + end_coord.rel_y + ")"    );

/* Set width, height, and origin. */
```

```
        init_arc_obj_component( );

    /* Trim arc. */
        dx = width - 20;
        dy = height - 20;
        arc_length = (int)(Math.sqrt(Math.pow(width,2)+Math.pow(height,2)));

        qx_for_src_trans = 0;
        qy_for_src_trans = 0;
        qx_for_dest_place = 0;
        qy_for_dest_place = 0;

        if( arc_length != 0 )
        {
           qx_for_src_trans = (10 * dx)/arc_length;
           qy_for_src_trans = (10 * dy)/arc_length;

           if( dx == 0 )
           {
              qx_for_dest_place = 0;
              qy_for_dest_place = 10;
           }
           else if( dy == 0 )
           {
              qx_for_dest_place = 10;
              qy_for_dest_place = 0;
           }
           else
           {
              if( dy/dx < 1 )
              {
                 qx_for_dest_place = 10;
                 qy_for_dest_place = (10 * dy)/dx;
              }
              else
              {
```

```
            qx_for_dest_place = (10 * dx)/dy;
            qy_for_dest_place = 10;
          }
        }
      }

System.out.println( "qx_for_src: " + qx_for_src_trans );
System.out.println( "qy_for_src: " + qy_for_src_trans );
System.out.println( "qx_for_dest: " + qx_for_dest_place );
System.out.println( "qy_for_dest: " + qy_for_dest_place );

      if( start_coord.rel_x > end_coord.rel_x )
      {
        start_coord.rel_x = start_coord.rel_x - qx_for_src_trans;
        end_coord.rel_x = end_coord.rel_x + qx_for_dest_place;
      }
      else
      {
        start_coord.rel_x = start_coord.rel_x + qx_for_src_trans;
        end_coord.rel_x = end_coord.rel_x - qx_for_dest_place;
      }

      if( start_coord.rel_y > end_coord.rel_y )
      {
        start_coord.rel_y = start_coord.rel_y - qy_for_src_trans;
        end_coord.rel_y = end_coord.rel_y + qy_for_dest_place;
      }
      else
      {
        start_coord.rel_y = start_coord.rel_y + qy_for_src_trans;
        end_coord.rel_y = end_coord.rel_y - qy_for_dest_place;
      }


/* PNK - Debug */
      System.out.println( "Trimmed start coord: (" + start_coord.rel_x + ", " + start_coord.rel_y
```

```
+ ")" );
      System.out.println( "Trimmed end coord: (" + end_coord.rel_x + ", " + end_coord.rel_y
+ ")" );
      System.out.println( "Origin: (" + origin_coord.rel_x + ", " + origin_coord.rel_y + ")"    );
/*end kaneko 20040617*/
   } /* end of arc_obj_component constructor for tp-arcs */

   void init_arc_obj_component( )
   {
/* Set size. */
      if( start_coord.rel_x > end_coord.rel_x )
      {
         origin_coord.rel_x = end_coord.rel_x - 10;
         width = start_coord.rel_x - end_coord.rel_x + 20;
      }
      else
      {
         origin_coord.rel_x = start_coord.rel_x - 10;
         width = end_coord.rel_x - start_coord.rel_x + 20;
      }

      if( start_coord.rel_y > end_coord.rel_y )
      {
         origin_coord.rel_y = end_coord.rel_y - 10;
         height = start_coord.rel_y - end_coord.rel_y + 20;
      }
      else
      {
         origin_coord.rel_y = start_coord.rel_y - 10;
         height = end_coord.rel_y - start_coord.rel_y + 20;
      }

      setSize( width, height );


/* PNK - Debug */
```

```
      System.out.println( "Size: " + '[' + width + ", " + height + ']'
+ "), Origin: (" + origin_coord.rel_x + ", " + origin_coord.rel_y + ")"    );
   }

   public void paint( Graphics g )
   {
      g.setColor( new Color(0,0,255) );
      g.drawLine( start_coord.rel_x - origin_coord.rel_x,
         start_coord.rel_y - origin_coord.rel_y,
         end_coord.rel_x - origin_coord.rel_x,
         end_coord.rel_y - origin_coord.rel_y);

      g.fillOval( end_coord.rel_x - origin_coord.rel_x - 3,
         end_coord.rel_y - origin_coord.rel_y - 3, 7, 7 );
   } /* end of paint */
} /* end of pt_arc_obj_component class */

class net_elem_id_obj extends Object
{
   int         type;
   Component            component_ptr;

   public net_elem_id_obj( )
   {
      type = 0;
      component_ptr = null;
   } /* end of net_elem_id_obj constructor */

   void set( place_obj_component place_obj_component_ptr )
   {
      type = const_h.PLACE;
      component_ptr = place_obj_component_ptr;
   } /* end of set method for place_obj_component */

   void set( trans_obj_component trans_obj_component_ptr )
   {
```

```
    type = const_h.TRANS;
    component_ptr = trans_obj_component_ptr;
  } /* end of set method for trans_obj_component */

  void clear( )
  {
    type = 0;
    component_ptr = null;
  } /* end of clear method */
}

/*start kaneko 20040927*/
class InputWindow extends Frame /*implements ActionListener*/
{
        Label l1 = new Label("Name:");
        TextField t1 = new TextField("",10);
        Label l2 = new Label("Type:");
        Label l4 = new Label("");
        Label l3 = new Label("Mark:");
        TextField t2 = new TextField("",10);

        InputWindow(String title, final place_obj place_obj_ptr)
        {
                super(title);
                setLayout(new BorderLayout());
                this.setLayout(new GridLayout(3,2));

                add(l1);
                add(t1);
                add(l2);
                add(l4);
                add(l3);
                add(t2);

                pack();
```

```
this.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent ev)
            {
                if(place_obj_ptr != null)
                {
                    place_obj temp_place_obj_ptr;
                    mark_obj temp_mark_obj_ptr;

                    temp_place_obj_ptr =
(place_obj)web_pn_sys.local_petrinet.place_lnklst.get(web_pn_sys.local_petrinet.place_lnklst.indexOf(place_obj_pt
r));
                    temp_place_obj_ptr.name=t1.getText();
                    temp_place_obj_ptr.mark++;
                    /*for(int i=0;i<temp_place_obj_ptr.mark;i++)
                    {*/
                    temp_mark_obj_ptr = new mark_obj();

temp_mark_obj_ptr.place_index=web_pn_sys.local_petrinet.place_lnklst.indexOf(place_obj_ptr);

temp_mark_obj_ptr.number=Integer.parseInt(t2.getText());

/*web_pn_sys.local_petrinet.add_mark( temp_mark_obj_ptr, const_h.TAIL );*/

temp_place_obj_ptr.mark_lnklst.add( temp_mark_obj_ptr );

System.out.println("place_index:"+temp_mark_obj_ptr.place_index);

System.out.println("number:"+temp_mark_obj_ptr.number);
                    /*}*/

System.out.println("name:"+temp_place_obj_ptr.name);

System.out.println("place_lnklst:"+web_pn_sys.local_petrinet.place_lnklst);

System.out.println("mark_lnklst1:"+web_pn_sys.local_place.mark_lnklst);
```

```
System.out.println("mark_lnklst2:"+temp_place_obj_ptr.mark_lnklst);
                                                }
                                                setVisible(false);
                                        }
                                }
                        );

                }
/*情報入力用 window*/
                InputWindow(String title, final trans_obj trans_obj_ptr)
                {
                        super(title);
                        setLayout(new BorderLayout());
                        this.setLayout(new GridLayout(2,2));

                        add(l1);
                        add(t1);
                        add(l2);
                        add(l4);
                        /*add(l3);
                        add(t2);*/

                        pack();

                        this.addWindowListener(new WindowAdapter()
                                {
                                        public void windowClosing(WindowEvent ev)
                                        {
                                                if(trans_obj_ptr != null)
                                                {
                                                        trans_obj temp_trans_obj_ptr;
                                                        temp_trans_obj_ptr                              =
(trans_obj)web_pn_sys.local_petrinet.trans_lnklst.get(web_pn_sys.local_petrinet.trans_lnklst.indexOf(trans_obj_pt
r));
                                                        temp_trans_obj_ptr.name=t1.getText();
```

```
            System.out.println("name:"+temp_trans_obj_ptr.name);

            System.out.println("lnklst:"+web_pn_sys.local_petrinet.trans_lnklst);
                                                    }
                                                    setVisible(false);
                                            }
                                    }
                            );
            }
}
/*end kaneko 20040927*/
/*トランジションを発火させる。*/
/*start kaneko20041208*/
class trans_fire_obj /*extends Thread*/
{
  int     num_places, num_trans, num_marks, num_in_places, num_out_places, i, j, k, l, mark_flg,
out_trans_flg1, out_trans_flg2;
  place_obj                 temp_place_ptr;
  trans_obj                         temp_trans_ptr;
  mark_obj                          temp_mark_ptr;
  mark_obj                          temp_mark_obj_ptr;
  place_obj_component       temp_place_obj_component;
  placeref_obj              temp_in_place_ptr, temp_out_place_ptr;

/* Check transition firabie. */
  public void trans_fire_obj( int main_design_mode )
  {
    num_trans = web_pn_sys.local_petrinet.trans_lnklst.size( );
    out_trans_flg1=1;
    out_trans_flg2=0;
System.out.println("num_trans:"+num_trans);

    for( i=0; i < num_trans; i++ )
    {
      mark_flg=0;
```

```
        temp_trans_ptr = (trans_obj)web_pn_sys.local_petrinet.trans_lnklst.get(i);

/* input mark. */
        num_in_places = temp_trans_ptr.in_place_lnklst.size( );

        for( j=0; j < num_in_places; j++ )
        {
           temp_in_place_ptr = (placeref_obj)temp_trans_ptr.in_place_lnklst.get(j);

           if(temp_in_place_ptr.place_obj_ptr.mark != 0)
              {
              mark_flg++;
              }
           else break;
        }
        if( main_design_mode == const_h.STEP_MODE )
        {
           if(temp_in_place_ptr.place_obj_ptr.mark == 0) continue;
        }
System.out.println("temp_in_place_ptr.place_obj_ptr.mark:"+temp_in_place_ptr.place_obj_ptr.mark);
System.out.println("num_in_places:"+num_in_places);
System.out.println("mark_flg:"+mark_flg);
/* output mark. */
        if(mark_flg == num_in_places)
        {
           for( j=0; j < num_in_places; j++ )
           {
              temp_in_place_ptr = (placeref_obj)temp_trans_ptr.in_place_lnklst.get(j);
                 if(out_trans_flg2==0) out_trans_flg1=temp_in_place_ptr.place_obj_ptr.out_trans_lnklst.size( );
           }

              num_out_places = temp_trans_ptr.out_place_lnklst.size( );
              /*num_marks = web_pn_sys.local_place.mark_lnklst.size( );*/

System.out.println("num_out_places:"+num_out_places);
```

```
System.out.println("num_marks:"+num_marks);

        for( j=0; j < num_out_places; j++ )
        {
           temp_out_place_ptr = (placeref_obj)temp_trans_ptr.out_place_lnklst.get(j);
           /*temp_out_place_ptr.place_obj_ptr.mark++;*/
               num_marks = temp_in_place_ptr.place_obj_ptr.mark_lnklst.size( );
           for( k=0; k < num_marks; k++ )
           {
              /*temp_mark_ptr = (mark_obj)web_pn_sys.local_place.mark_lnklst.get(k);*/
                  temp_mark_ptr = (mark_obj)temp_in_place_ptr.place_obj_ptr.mark_lnklst.get(k);
              for( l=0; l < num_in_places; l++ )
              {
                 temp_in_place_ptr = (placeref_obj)temp_trans_ptr.in_place_lnklst.get(l);
                 if(temp_mark_ptr.place_index                                                                    ==
web_pn_sys.local_petrinet.place_lnklst.indexOf(temp_in_place_ptr.place_obj_ptr))
                   {
System.out.println("mark_place_index1:"+temp_mark_ptr.place_index);
                       temp_mark_obj_ptr = new mark_obj();

            temp_mark_obj_ptr.place_index=web_pn_sys.local_petrinet.place_lnklst.indexOf(temp_out_place_ptr.pl
ace_obj_ptr);
                          temp_mark_obj_ptr.number=temp_mark_ptr.number;

/*temp_mark_ptr.place_index=web_pn_sys.local_petrinet.place_lnklst.indexOf(temp_out_place_ptr.place_obj_ptr);*
/
                       temp_out_place_ptr.place_obj_ptr.mark_lnklst.add( temp_mark_obj_ptr );
System.out.println("mark_place_index2:"+temp_mark_obj_ptr.place_index);
System.out.println("mark_place_number2:"+temp_mark_obj_ptr.number);
                   }
                }
              }
                 temp_out_place_ptr.place_obj_ptr.mark=temp_out_place_ptr.place_obj_ptr.mark_lnklst.size( );
        }
            if(out_trans_flg1!=1)
            {
```

```
                out_trans_flg1--;
                out_trans_flg2=1;
                continue;
              }
            else
              {
              for( j=0; j < num_in_places; j++ )
              {
                temp_in_place_ptr = (placeref_obj)temp_trans_ptr.in_place_lnklst.get(j);
                  temp_in_place_ptr.place_obj_ptr.mark = 0;
                  for( k=0; k < temp_in_place_ptr.place_obj_ptr.mark_lnklst.size(); k++)
                  {
                    temp_in_place_ptr.place_obj_ptr.mark_lnklst.removeFirst();
                  }
              }
              }
          }
          if( main_design_mode == const_h.STEP_MODE )
          {
            break;
          }
        }
    }
}

class save_state_obj
{
    int         num_places, num_trans, num_marks, num_out_trans, num_in_places, num_out_places, i;
    place_obj                       temp_place_ptr;
    trans_obj                               temp_trans_ptr;
    mark_obj                                temp_mark_ptr;
    placeref_obj                            temp_in_place_ptr, temp_out_place_ptr;
    transref_obj                            temp_out_trans_ptr;

    public void save_state_obj()
    {
```

```
    num_places = web_pn_sys.local_petrinet.place_lnklst.size( );
    for( i=0; i < num_places; i++ )
    {
       temp_place_ptr = (place_obj)web_pn_sys.local_petrinet.place_lnklst.get(i);
       System.out.println("temp_place_ptr:"+temp_place_ptr);
       /*System.out.println("temp_place_ptr.location.coord.rel_x:"+temp_place_ptr.location.coord.rel_x);
       System.out.println("temp_place_ptr.location.coord.rel_y:"+temp_place_ptr.location.coord.rel_y);*/
    }

    num_trans = web_pn_sys.local_petrinet.trans_lnklst.size( );
    for( i=0; i < num_trans; i++ )
    {
       temp_trans_ptr = (trans_obj)web_pn_sys.local_petrinet.trans_lnklst.get(i);
       System.out.println("temp_trans_ptr:"+temp_trans_ptr);
    }

    num_marks = web_pn_sys.local_place.mark_lnklst.size( );
    for( i=0; i < num_marks; i++ )
    {
       temp_mark_ptr = (mark_obj)web_pn_sys.local_place.mark_lnklst.get(i);
       System.out.println("temp_mark_ptr:"+temp_mark_ptr);
    }

    num_out_trans = web_pn_sys.local_place.out_trans_lnklst.size( );
System.out.println("num_out_trans:"+num_out_trans);
    for( i=0; i < num_out_trans; i++ )
    {
       temp_out_trans_ptr = (transref_obj)web_pn_sys.local_place.out_trans_lnklst.get(i);
       System.out.println("temp_out_trans_ptr:"+temp_out_trans_ptr);
    }

    num_in_places = web_pn_sys.local_trans.in_place_lnklst.size( );
    for( i=0; i < num_in_places; i++ )
    {
       temp_in_place_ptr = (placeref_obj)web_pn_sys.local_trans.in_place_lnklst.get(i);
       System.out.println("temp_in_place_ptr:"+temp_in_place_ptr);
```

```
      }

      num_out_places = web_pn_sys.local_trans.out_place_lnklst.size( );
      for( i=0; i < num_out_places; i++ )
      {
         temp_out_place_ptr = (placeref_obj)web_pn_sys.local_trans.out_place_lnklst.get(i);
         System.out.println("temp_out_place_ptr:"+temp_out_place_ptr);
      }
   }
}
/*end kaneko20041208*/

public class main_wdw_obj extends Panel implements ActionListener, MouseListener
{
/* Main window components */
   main_banner_component         main_banner;

   main_mode_menu_marker_component main_mode_menu_marker;

/*start kaneko20041116*/
   public Button main_load_button = new Button( "Load" );
   public Button main_save_button = new Button( "Save" );
   public Button main_run_button = new Button( "Run" );
/*end kaneko20041116*/

   public Button main_pointer_mode_button = new Button( "Pointer" );
   public Button main_place_mode_button = new Button( "Place" );
   public Button main_trans_mode_button = new Button( "Trans" );
   public Button main_arc_mode_button = new Button( "Arc" );

/*start kaneko20041117*/
   public Button main_step_button = new Button( "Step" );
   public Button main_go_button = new Button( "Go" );
/*end kaneko20041117*/

   public Panel main_design_area_panel = new Panel( null );
```

```
/* Main window state */
   public byte                          main_design_mode;
   net_elem_id_obj      main_curr_net_elem_id = new net_elem_id_obj( );

   public main_wdw_obj( Dimension applet_dimensions )
   {
/* Initialize main window panel. */
      setSize( applet_dimensions.width, applet_dimensions.height );
      setLocation( 0, 0 );
      setBackground( new Color(255,255,0) );
      setLayout( null );

/* Create and initialize banner component. */
      main_banner = new main_banner_component( applet_dimensions.width,
         70 );
      main_banner.setLocation( 0, 0 );

/* Initialize buttons. */
/*start kaneko20041116*/
      main_load_button.setSize( 80, 30 );
      main_load_button.setLocation( 15, main_banner.getHeight( ) - 40 );
      main_load_button.setBackground( new Color( 0, 255, 0 ) );
      main_load_button.addActionListener( this );

      main_save_button.setSize( 80, 30 );
      main_save_button.setLocation( 115, main_banner.getHeight( ) - 40 );
      main_save_button.setBackground( new Color( 0, 255, 0 ) );
      main_save_button.addActionListener( this );

      main_run_button.setSize( 80, 30 );
      main_run_button.setLocation( 750, main_banner.getHeight( ) - 40 );
      main_run_button.setBackground( new Color( 0, 255, 0 ) );
      main_run_button.addActionListener( this );
/*end kaneko20041116*/
      main_pointer_mode_button.setSize( 80, 30 );
```

```
        main_pointer_mode_button.setLocation( 15, main_banner.getHeight( ) + 30 );
        main_pointer_mode_button.setBackground( new Color( 0, 255, 0 ) );
        main_pointer_mode_button.addActionListener( this );

        main_place_mode_button.setSize( 80, 30 );
        main_place_mode_button.setLocation( 15, main_banner.getHeight( ) + 70 );
        main_place_mode_button.setBackground( new Color( 0, 255, 0 ) );
        main_place_mode_button.addActionListener( this );

        main_trans_mode_button.setSize( 80, 30 );
        main_trans_mode_button.setLocation( 15, main_banner.getHeight( ) + 110 );
        main_trans_mode_button.setBackground( new Color( 0, 255, 0 ) );
        main_trans_mode_button.addActionListener( this );

        main_arc_mode_button.setSize( 80, 30 );
        main_arc_mode_button.setLocation( 15, main_banner.getHeight( ) + 150 );
        main_arc_mode_button.setBackground( new Color( 0, 255, 0 ) );
        main_arc_mode_button.addActionListener( this );

/*start kaneko20041117*/
        main_step_button.setSize( 80, 30 );
        main_step_button.setLocation( 15, main_banner.getHeight( ) + 190 );
        main_step_button.setBackground( new Color( 0, 255, 0 ) );
        main_step_button.addActionListener( this );

        main_go_button.setSize( 80, 30 );
        main_go_button.setLocation( 15, main_banner.getHeight( ) + 230 );
        main_go_button.setBackground( new Color( 0, 255, 0 ) );
        main_go_button.addActionListener( this );
/*end kaneko20041117*/

/* Create and initialize mode menu marker component. */
        main_mode_menu_marker = new main_mode_menu_marker_component( );
        main_mode_menu_marker.setLocation( 5, main_banner.getHeight( ) + 40 );

/* Add components to main window panel. */
```

```
/*start kaneko20041116*/
        add( main_load_button );
        add( main_save_button );
        add( main_run_button );
/*end kaneko20041116*/
        add( main_banner );
        add( main_pointer_mode_button );
        add( main_place_mode_button );
        add( main_trans_mode_button );
        add( main_arc_mode_button );
        add( main_mode_menu_marker );

/* Initialize design panel. */
        main_design_area_panel.setBackground( new Color( 0, 255, 255) );

        main_design_area_panel.setSize( applet_dimensions.width - 105,
           applet_dimensions.height - 70 );
        main_design_area_panel.setLocation( 105, main_banner.getHeight( ) );
        main_design_area_panel.addMouseListener( this );

/* Add design area to main window panel. */
        add( main_design_area_panel );

/* Initialize window state. */
        main_design_mode = const_h.POINTER_MODE;
        main_curr_net_elem_id.clear( );
    } /* end of main_wdw_obj constructor */

/*-------------------------------------------------------*/
/* Method: disp_net_obj( )                               */
/*                                                       */
/* Function:   To display the objects of the local       */
/*     Petri net.                                        */
/*                                                       */
/* Expects:    Nothing                                   */
/*-------------------------------------------------------*/
```

```
   public void disp_net_obj( )
   {
      int      num_places, num_trans, num_in_places, num_out_places, i, j;
      place_obj                              temp_place_ptr;
      trans_obj                              temp_trans_ptr;
      placeref_obj                 temp_in_place_ptr, temp_out_place_ptr;
      place_obj_component          temp_place_obj_component;
      trans_obj_component          temp_trans_obj_component;
      arc_obj_component            temp_arc_obj_component;

/* Draw place objects. */
      num_places = web_pn_sys.local_petrinet.place_lnklst.size( );

      for( i=0; i < num_places; i++ )
      {
         temp_place_ptr = (place_obj)web_pn_sys.local_petrinet.place_lnklst.get(i);

         temp_place_obj_component = new place_obj_component( temp_place_ptr );
/* PNK - what happens if you can't? */
         temp_place_obj_component.setLocation( temp_place_ptr.location.coord.rel_x - 10,
            temp_place_ptr.location.coord.rel_y - 10 );

/* Add to design area panel. */
         main_design_area_panel.add( temp_place_obj_component );
         temp_place_obj_component.repaint( );
      }

/* Draw transition and arc objects. */
      num_trans = web_pn_sys.local_petrinet.trans_lnklst.size( );

      for( i=0; i < num_trans; i++ )
      {
         temp_trans_ptr = (trans_obj)web_pn_sys.local_petrinet.trans_lnklst.get(i);

         temp_trans_obj_component = new trans_obj_component( temp_trans_ptr );
/* PNK - what happens if you can't? */
```

```
        temp_trans_obj_component.setLocation( temp_trans_ptr.location.coord.rel_x - 10,
          temp_trans_ptr.location.coord.rel_y - 10 );

/* Add to design area panel. */
        main_design_area_panel.add( temp_trans_obj_component );
        temp_trans_obj_component.repaint( );

/* Draw input arcs. */
        num_in_places = temp_trans_ptr.in_place_lnklst.size( );

        for( j=0; j < num_in_places; j++ )
        {
          temp_in_place_ptr = (placeref_obj)temp_trans_ptr.in_place_lnklst.get(j);

          temp_arc_obj_component = new arc_obj_component( temp_in_place_ptr.place_obj_ptr,
            temp_trans_ptr );

/*start kaneko*/
/* PNK - what happens if you can't? */
        temp_arc_obj_component.setLocation( temp_in_place_ptr.place_obj_ptr.location.coord.rel_x - 10,
          temp_in_place_ptr.place_obj_ptr.location.coord.rel_y - 10 );
/*end kaneko*/

/* Add to design area panel. */
          /*main_design_area_panel.add( temp_arc_obj_component );*/
          temp_arc_obj_component.repaint( );
        }

/* Draw output arcs. */
        num_out_places = temp_trans_ptr.out_place_lnklst.size( );

        for( j=0; j < num_out_places; j++ )
        {
          temp_out_place_ptr = (placeref_obj)temp_trans_ptr.out_place_lnklst.get(j);

          temp_arc_obj_component = new arc_obj_component( temp_trans_ptr,
```

```
                temp_out_place_ptr.place_obj_ptr );

/*start kaneko*/
/* PNK - what happens if you can't? */
          temp_arc_obj_component.setLocation( temp_trans_ptr.location.coord.rel_x - 10,
             temp_trans_ptr.location.coord.rel_y - 10 );
/*end kaneko*/

/* Add to design area panel. */
             /*main_design_area_panel.add( temp_arc_obj_component );*/
             temp_arc_obj_component.repaint( );
          }
       }
    } /* end of disp_net_obj */

/*-------------------------------------------------*/
/* Method: actionPerformed( ActionEvent e )        */
/*     (override ActionListener actionPerformed)   */
/*                                                 */
/* Function:    To process actions (e.g., mouse    */
/*     clicks) on components.                      */
/*                                                 */
/* Expects:    e, event information to be provided */
/*                    by the system                */
/*-------------------------------------------------*/
    public void actionPerformed( ActionEvent e )
    {
       Object curr_obj;
       place_obj_component          temp_place_obj_component;
       trans_obj_component          temp_trans_obj_component;
       trans_fire_obj       temp_trans_fire_obj = new trans_fire_obj( );
       save_state_obj         temp_save_state_obj = new save_state_obj( );

       curr_obj = e.getSource( );

       if( curr_obj instanceof Button )
```

```
    {
/* De-highlight and clear current net element, as necessary, at each click. */
    switch( main_curr_net_elem_id.type )
    {
      case const_h.PLACE:
        temp_place_obj_component = (place_obj_component)main_curr_net_elem_id.component_ptr;
        temp_place_obj_component.highlight = const_h.OFF;
        temp_place_obj_component.repaint( );
        break;

      case const_h.TRANS:
        temp_trans_obj_component = (trans_obj_component)main_curr_net_elem_id.component_ptr;
        temp_trans_obj_component.highlight = const_h.OFF;
        temp_trans_obj_component.repaint( );
        break;

      default:
        break;
    }

/* Clear current net element. */
      main_curr_net_elem_id.clear( );

/* Check which button. */
/*start kaneko20041117*/
      if( curr_obj == main_load_button )
      {
        add( main_pointer_mode_button );
        add( main_place_mode_button );
        add( main_trans_mode_button );
        add( main_arc_mode_button );
        remove( main_step_button );
        remove( main_go_button );
      }
      else if( curr_obj == main_save_button )
      {
```

```
            temp_save_state_obj.save_state_obj();
        }
        else if( curr_obj == main_run_button )
        {
            remove( main_pointer_mode_button );
            remove( main_place_mode_button );
            remove( main_trans_mode_button );
            remove( main_arc_mode_button );
            add( main_step_button );
            add( main_go_button );
        }
/*end kaneko20041117*/
        else if( curr_obj == main_pointer_mode_button )
        {
            main_design_mode = const_h.POINTER_MODE;

/* Move mode menu marker. */
            main_mode_menu_marker.setLocation( 5, main_banner.getHeight( ) + 40 );
        }
        else if( curr_obj == main_place_mode_button )
        {
            main_design_mode = const_h.PLACE_MODE;

/* Move mode menu marker. */
            main_mode_menu_marker.setLocation( 5, main_banner.getHeight( ) + 80 );
        }
        else if( curr_obj == main_trans_mode_button )
        {
            main_design_mode = const_h.TRANS_MODE;

/* Move mode menu marker. */
            main_mode_menu_marker.setLocation( 5, main_banner.getHeight( ) + 120 );
        }
        else if( curr_obj == main_arc_mode_button )
        {
            main_design_mode = const_h.ARC_MODE;
```

```
/* Move mode menu marker. */
            main_mode_menu_marker.setLocation( 5, main_banner.getHeight( ) + 160 );
        }
/*start kaneko20041117*/
        else if( curr_obj == main_step_button )
        {
            main_design_mode = const_h.STEP_MODE;
                temp_trans_fire_obj.trans_fire_obj(main_design_mode);

/* Move mode menu marker. */
            main_mode_menu_marker.setLocation( 5, main_banner.getHeight( ) + 200 );

            disp_net_obj( );
        }
        else if( curr_obj == main_go_button )
        {
            main_design_mode = const_h.GO_MODE;
                temp_trans_fire_obj.trans_fire_obj(main_design_mode);

/* Move mode menu marker. */
            main_mode_menu_marker.setLocation( 5, main_banner.getHeight( ) + 240 );

                disp_net_obj( );
        }
/*end kaneko20041117*/
      }
   } /* end of actionPerformed */

/*-------------------------------------------------------------*/
/* Method: mouseClicked( MouseEvent mouse_event    )          */
/*     (override MouseListener mouseClicked)                   */
/*                                                             */
/* Function:    To process mouse clicks.                       */
/*                                                             */
/* Expects:    mouse_event, mouse event information to be    */
```

```
/*                    provided by the system                          */
/*------------------------------------------------------------*/
   public void mouseClicked( MouseEvent mouse_event )
   {
      int                  curr_x, curr_y, abs_x, abs_y;
      place_obj                        temp_place_ptr;
      trans_obj            temp_trans_ptr;
      placeref_obj         temp_placeref_ptr;
      transref_obj         temp_transref_ptr;
      place_obj_component            temp_place_obj_component;
      trans_obj_component            temp_trans_obj_component;
      arc_obj_component              temp_arc_obj_component;
      Object               src_obj;
      Component            temp_component;

/* De-highlight current net element, as necessary, at each click. */
      switch( main_curr_net_elem_id.type )
      {
        case const_h.PLACE:
           temp_place_obj_component = (place_obj_component)main_curr_net_elem_id.component_ptr;
           temp_place_obj_component.highlight = const_h.OFF;
           temp_place_obj_component.repaint( );
           break;

        case const_h.TRANS:
           temp_trans_obj_component = (trans_obj_component)main_curr_net_elem_id.component_ptr;
           temp_trans_obj_component.highlight = const_h.OFF;
           temp_trans_obj_component.repaint( );
           break;

        default:
           break;
      }

      curr_x = mouse_event.getX( );
      curr_y = mouse_event.getY( );
```

```
      src_obj = mouse_event.getSource( );

      if( src_obj instanceof Panel )
      {
        if( src_obj == main_design_area_panel )
        {
/* Check design mode. */
          switch( main_design_mode )
          {
            case const_h.POINTER_MODE:
              if( mouse_event.getClickCount() == 2 )
              {
                System.out.println( "Double clicked..." );
              }
              else
              {
                System.out.println( "Single clicked..." );
              }

              abs_x = curr_x + main_design_area_panel.getX();
              abs_y = curr_y + main_design_area_panel.getY();

              temp_component = findComponentAt( abs_x, abs_y );
              if( temp_component == main_design_area_panel )
              {
/* Clear current net element. */
                main_curr_net_elem_id.clear( );

                System.out.println( "Clicked nothing..." );
              }
              else if( temp_component instanceof place_obj_component )
              {
                temp_place_obj_component = (place_obj_component)temp_component;
                temp_place_ptr = temp_place_obj_component.place_obj_ptr;

                System.out.println( "Clicked place_obj_component...number:"
```

```
                + web_pn_sys.local_petrinet.place_lnklst.indexOf(temp_place_obj_component.place_obj_ptr) );

            if( mouse_event.getClickCount() == 2 )
            {
/*start kaneko 20040927*/
                InputWindow iw = new InputWindow("Please Input (P)",temp_place_obj_component.place_obj_ptr);
                iw.setSize(210,120);
                iw.setLocation(abs_x,abs_y);
                iw.setVisible(true);
                iw.t1.setText(temp_place_obj_component.place_obj_ptr.name);

        iw.l4.setText(String.valueOf(web_pn_sys.local_petrinet.place_lnklst.indexOf(temp_place_obj_component.place_obj_ptr)));
                iw.t2.setText(String.valueOf(temp_place_obj_component.place_obj_ptr.mark));
              System.out.println( "Clicked place_obj_component...name:"
                + temp_place_obj_component.place_obj_ptr.name);

/*end kaneko 20040927*/

                System.out.println( "Double clicked..." );

            }

          }
          else if( temp_component instanceof trans_obj_component )
          {
            temp_trans_obj_component = (trans_obj_component)temp_component;
            temp_trans_ptr = temp_trans_obj_component.trans_obj_ptr;
/*start kaneko 20041102*/
            System.out.println( "Clicked trans_obj_component...number:"
                + web_pn_sys.local_petrinet.trans_lnklst.indexOf(temp_trans_obj_component.trans_obj_ptr) );

            if( mouse_event.getClickCount() == 2 )
            {
                InputWindow iw = new InputWindow("Please Input
```

```
(t)",temp_trans_obj_component.trans_obj_ptr);
                    iw.setSize(210,100);
                    iw.setLocation(abs_x,abs_y);
                        iw.setVisible(true);
                        iw.t1.setText(temp_trans_obj_component.trans_obj_ptr.name);

            iw.l4.setText(String.valueOf(web_pn_sys.local_petrinet.trans_lnklst.indexOf(temp_trans_obj_componen
t.trans_obj_ptr)));
                System.out.println( "Clicked trans_obj_component...name:"
                  + temp_trans_obj_component.trans_obj_ptr.name);
                  System.out.println( "Double clicked..." );
                }
/*end kaneko 20041102*/

/* PNK - Finish this. */

              }
              else
              {
/* Clear current net element. */
                main_curr_net_elem_id.clear( );

                System.out.println( "Something else..." );
              }

              break;

            case const_h.PLACE_MODE:
/* Draw a new place. */
              temp_place_ptr = new place_obj( );
              temp_place_ptr.location.coord.rel_x = curr_x;
              temp_place_ptr.location.coord.rel_y = curr_y;

/* Add to local petrinet. */
              web_pn_sys.local_petrinet.add_place( temp_place_ptr, const_h.TAIL );
```

```
/* Create place graphics object. */
                temp_place_obj_component = new place_obj_component( temp_place_ptr );
                temp_place_obj_component.setLocation( curr_x-10, curr_y-10 );

/* Add to design area panel. */
                main_design_area_panel.add( temp_place_obj_component );
                temp_place_obj_component.repaint( );
                break;

              case const_h.TRANS_MODE:
/* Draw a new transition. by kaneko*/
                temp_trans_ptr = new trans_obj( );
                temp_trans_ptr.location.coord.rel_x = curr_x;
                temp_trans_ptr.location.coord.rel_y = curr_y;

/* Add to local petrinet. */
                web_pn_sys.local_petrinet.add_trans( temp_trans_ptr, const_h.TAIL );

/* Create transition graphics object. */
                temp_trans_obj_component = new trans_obj_component( temp_trans_ptr );
                temp_trans_obj_component.setLocation( curr_x-10, curr_y-10 );

/* Add to design area panel. */
                main_design_area_panel.add( temp_trans_obj_component );
                temp_trans_obj_component.repaint( );
                break;

              case const_h.ARC_MODE:
                abs_x = curr_x + main_design_area_panel.getX();
                abs_y = curr_y + main_design_area_panel.getY();

                temp_component = findComponentAt( abs_x, abs_y );
                if( temp_component == main_design_area_panel )
                {
/* Clear current net element. */
                  main_curr_net_elem_id.clear( );
```

```
                System.out.println( "Clicked nothing..." );
            }
            else if( temp_component instanceof place_obj_component )
            {
                temp_place_obj_component = (place_obj_component)temp_component;
                temp_place_ptr = temp_place_obj_component.place_obj_ptr;

                switch( main_curr_net_elem_id.type )
                {
                    case const_h.TRANS:
                        temp_trans_obj_component = (trans_obj_component)main_curr_net_elem_id.component_ptr;
                        temp_trans_ptr = (trans_obj)temp_trans_obj_component.trans_obj_ptr;

/* Add to output places list. */
                        temp_placeref_ptr = new placeref_obj( );
                        temp_placeref_ptr.place_obj_ptr = temp_place_ptr;
                        temp_trans_ptr.out_place_lnklst.add( temp_placeref_ptr );

/* Create arc graphics object. */
                        temp_arc_obj_component = new arc_obj_component( temp_trans_ptr,
                            temp_place_ptr );

/* Add to design area panel. */
                        main_design_area_panel.add( temp_arc_obj_component );
                        temp_arc_obj_component.setLocation( temp_arc_obj_component.origin_coord.rel_x,
                            temp_arc_obj_component.origin_coord.rel_y );
                        temp_arc_obj_component.repaint( );

/* Clear current net element. */
                        main_curr_net_elem_id.clear( );
                        break;

                    default:
                    case const_h.PLACE:
/* Set current net element. */
```

```
                main_curr_net_elem_id.set( temp_place_obj_component );
                temp_place_obj_component.highlight = const_h.ON;
                temp_place_obj_component.repaint( );
                break;
            }
            System.out.println( "Clicked place_obj_component...number:"
              + web_pn_sys.local_petrinet.place_lnklst.indexOf(temp_place_obj_component.place_obj_ptr) );

        }
        else if( temp_component instanceof trans_obj_component )
        {
            temp_trans_obj_component = (trans_obj_component)temp_component;
            temp_trans_ptr = temp_trans_obj_component.trans_obj_ptr;

            switch( main_curr_net_elem_id.type )
            {
              case const_h.PLACE:
                temp_place_obj_component = (place_obj_component)main_curr_net_elem_id.component_ptr;
                temp_place_ptr = (place_obj)temp_place_obj_component.place_obj_ptr;

/* Add to output transitions list and input places list. */
                temp_transref_ptr = new transref_obj( );
                temp_transref_ptr.trans_obj_ptr = temp_trans_ptr;
                temp_place_ptr.out_trans_lnklst.add( temp_transref_ptr );

                temp_placeref_ptr = new placeref_obj( );
                temp_placeref_ptr.place_obj_ptr = temp_place_ptr;
                temp_trans_ptr.in_place_lnklst.add( temp_placeref_ptr );

/* Create arc graphics object. */
                temp_arc_obj_component = new arc_obj_component( temp_place_ptr,
                  temp_trans_ptr );

/* Add to design area panel. */
                main_design_area_panel.add( temp_arc_obj_component );
                temp_arc_obj_component.setLocation( temp_arc_obj_component.origin_coord.rel_x,
```

```
                        temp_arc_obj_component.origin_coord.rel_y );
                    temp_arc_obj_component.repaint( );

/* Clear current net element. */
                    main_curr_net_elem_id.clear( );
                    break;

                  default:
                  case const_h.TRANS:
/* Switch current net element. */
                    main_curr_net_elem_id.set( temp_trans_obj_component );
                    temp_trans_obj_component.highlight = const_h.ON;
                    temp_trans_obj_component.repaint( );
                    break;
                }
                System.out.println( "Clicked trans_obj_component...number:"
                  + web_pn_sys.local_petrinet.trans_lnklst.indexOf(temp_trans_obj_component.trans_obj_ptr) );
              }
              else
              {
/* Clear current net element. */
                main_curr_net_elem_id.clear( );

                System.out.println( "Something else..." );
              }




              break;

            default:
              break;
          }
        }
```

```
    }
  } /* end of mouseClicked */
  public void mouseEntered( MouseEvent mouse_event )
  {
    ;
  } /* end of mouseEntered */

  public void mouseExited( MouseEvent mouse_event )
  {
    ;
  } /* end of mouseExited */

  public void mousePressed( MouseEvent mouse_event )
  {
    ;
  } /* end of mousePressed */

  public void mouseReleased( MouseEvent mouse_event )
  {
    ;
  } /* end of mouseReleased */
}
```

<const_h.java>

変数の初期化を行う。

```
/*************************************************************/
/*                          CONST_H.JAVA                     */
/*   This file contains the class definitions for the       */
/*   web_pn_sys system constants.                            */
/*   Written by:   Pauline N. Kawamoto (Shinshu University) */
/*************************************************************/

/*-----------------------------------------------------*/
/* Edit History                                        */
/*-----------------------------------------------------*/
/* [5/18/2004] - Quit packages.                        */
```

```
/* [1/7/2004] - File created.                                    */
/*--------------------------------------------------------------*/

public class const_h extends Object
{
/* Linked List Constants */
   static final int         HEAD = 0;
   public static final int              TAIL = -1;

/* Status Codes */
   static final int         SUCCESS = 1;
   static final int         ERROR = -1;

   static final boolean  ON = true;
   static final boolean  OFF = false;

/*--------------------------------------------------------------*/
/* Main Window Constants                                        */
/*--------------------------------------------------------------*/
/* Window modes */
   public static final int POINTER_MODE = 1;
   public static final int PLACE_MODE = 2;
   public static final int TRANS_MODE = 3;
   public static final int ARC_MODE = 4;
/*start kaneko20041208*/
   public static final int STEP_MODE = 5;
   public static final int GO_MODE = 6;
/*end kaneko20041208*/

/*--------------------------------------------------------------*/
/* Place/Transition/Module Constants                            */
/*--------------------------------------------------------------*/
/* Net element types for net_elem_id_obj */
   static final int         PLACE = 0x0001;
   static final int         TRANS = 0x0002;
```

```
/* Constants for place_obj */
   static final int        MAX_PLACE_NAME_LEN = 8;

   static final int        START_PLACE = 0x0001;
   static final int        END_PLACE = 0x0002;
}
```

<elem_location_obj.java>

デザインエリアの位置情報

```
/*************************************************************/
/*                                                           */
/*                    ELEM_LOCATION_OBJ.JAVA                 */
/*                                                           */
/*   This file contains the class definition for the         */
/*   web_pn_sys element location object.                     */
/*                                                           */
/*   Written by:   Pauline N. Kawamoto (Shinshu University) */
/*                                                           */
/*************************************************************/

/*-----------------------------------------------------------*/
/* Edit History                                              */
/*-----------------------------------------------------------*/
/* [5/18/2004] - Quit packages.                              */
/* [1/7/2004] - File created.                                */
/*-----------------------------------------------------------*/

import java.util.*;

/*-----------------------------*/
/* Window Information Objects */
/*-----------------------------*/
class rel_location_obj extends Object
{
   public int              rel_x;
   public int              rel_y;
```

```
};

public class elem_location_obj extends Object
{
   public module_obj                cover_module_ptr;
   public rel_location_obj          coord = new rel_location_obj( );

   public elem_location_obj( )
   {
      cover_module_ptr = null;
      coord.rel_x = 0;
      coord.rel_y = 0;
   } /* end of elem_location_obj constructor */
};
```

<mark_obj.java>

マーク情報

```
/*************************************************************/
/*                          MARK_OBJ.JAVA                    */
/*   This file contains the class definition for the         */
/*   web_pn_sys mark object.                                 */
/*   Written by:   Yasuhiro Kaneko (Shinshu University)      */
/*************************************************************/

/*-----------------------------------------------------------*/
/* Edit History                                              */
/*-----------------------------------------------------------*/
/* [11/7/2004] - File created.                               */
/*-----------------------------------------------------------*/

import java.awt.*;
import java.awt.event.*;
import java.util.*;

/*--------------*/
/* Mark Object   */
```

```
/*--------------*/
public class mark_obj extends Object
{
   int                place_index;
   int                        number;

   public mark_obj( )
   {
      place_index = 0x00;
      number = 0x00;
   } /* end of mark_obj constructor */
}; /* end of mark_obj class */
```

<module_obj.java>

位置情報

```
/*************************************************************/
/*                          MODULE_OBJ.JAVA                  */
/*   This file contains the class definition for the         */
/*   web_pn_sys system module object.                        */
/*   Written by:   Pauline N. Kawamoto (Shinshu University) */
/*************************************************************/

/*-----------------------------------------------------------*/
/* Edit History                                              */
/*-----------------------------------------------------------*/
/* [5/18/2004] - Quit packages.                              */
/* [1/7/2004] - File created.                                */
/*-----------------------------------------------------------*/

import java.util.*;

/*---------------*/
/* Module Object */
/*---------------*/
public class module_obj extends Object
{
```

```
   elem_location_obj     location = new elem_location_obj();
   LinkedList child_lnklst_ptr = new LinkedList( );

/* Constructor: module object */
   public module_obj( module_obj parent_module )
   {
      location.coord.rel_x = 0;
      location.coord.rel_y = 0;
      location.cover_module_ptr = parent_module;
   }

/*---------------------------------------*/
/* Method:    purge_child_lnklst( )          */
/*                                           */
/* Function:    To clear the child modules */
/*     linked list.                          */
/*---------------------------------------*/
   public void purge_child_lnklst( )
   {
      child_lnklst_ptr.clear( );
   }
};
```

<petrinet_obj.java>

デザインエリアに描画するときに使用する。

```
/*********************************************************/
/*                          PETRINET_OBJ.JAVA                          */
/*   This file contains the class definitions for the        */
/*   web_pn_sys Petri net object.                              */
/*   Written by:    Pauline N. Kawamoto (Shinshu University) */
/*********************************************************/

/*-------------------------------------------------------*/
/* Edit History                                           */
/*-------------------------------------------------------*/
/* [5/18/2004] - Quit packages.                          */
```

```
/* [1/7/2004] - File created.                                    */
/*--------------------------------------------------------------*/

import java.util.*;
import      java.awt.*;
import      java.awt.event.*;

/*------------------*/
/* Petri Net Object */
/*------------------*/
public class petrinet_obj extends Object
{
   public LinkedList    place_lnklst = new LinkedList();
   public LinkedList    trans_lnklst = new LinkedList();
   public LinkedList    module_lnklst = new LinkedList();
   public module_obj    module_tree_root_module = new module_obj(null);
   public module_obj    curr_cover_module_ptr;

/*-----------------------------------------------*/
/* Method: purge( )                              */
/*                                               */
/* Function: To purge all elements from          */
/*     the linked lists of the Petri net object. */
/*-----------------------------------------------*/
   public void purge( )
   {
      place_lnklst.clear( );
      trans_lnklst.clear( );
      module_lnklst.clear( );
      module_tree_root_module.purge_child_lnklst( );
      curr_cover_module_ptr = module_tree_root_module;
   } /* end of purge */

/*-------------------------------------------------*/
/* Method: build_trivial_petrinet( )               */
/*                                                 */
```

```
/* Function:    To build the trivial Petri                  */
/*      net object - START/END places in a                  */
/*      cover module.                                       */
/*                                                          */
/* Returns:    const_h.SUCCESS, operation successful */
/*                  const_h.ERROR, error encountered        */
/*----------------------------------------------------*/
   public int build_trivial_petrinet( )
   {
      place_obj             temp_place_ptr;
      int       tmp_place_lnklst_index;

/* Purge the Petri net object. */
      purge( );

/* Build START/END places. */
      if( (temp_place_ptr = new place_obj( )) != null )
      {
         temp_place_ptr.name = "START";
         temp_place_ptr.type = const_h.START_PLACE;
         temp_place_ptr.location.coord.rel_x = 100;
         temp_place_ptr.location.coord.rel_y = 50;
         temp_place_ptr.location.cover_module_ptr = module_tree_root_module;

         add_place( temp_place_ptr, const_h.TAIL );
      }
      else
      {
         return const_h.ERROR;
      }

      if( (temp_place_ptr = new place_obj( )) != null )
      {
         temp_place_ptr.name = "END";
         temp_place_ptr.type = const_h.END_PLACE;
         temp_place_ptr.location.coord.rel_x = 500;
```

```
      temp_place_ptr.location.coord.rel_y = 400;
      temp_place_ptr.location.cover_module_ptr = module_tree_root_module;

      add_place( temp_place_ptr, const_h.TAIL );
    }
    else
    {
      return const_h.ERROR;
    }

    return const_h.SUCCESS;
  } /* end of build_trivial_petrinet */

/*---------------------------------------------------------------*/
/* Method:   add_place( place_obj place_obj_ptr, int position ) */
/*                                                               */
/* Function:   To add a place object to the Petri net place      */
/*    linked list at the designated position.                    */
/*                                                               */
/* Expects:   place_obj_ptr, pointer to place object to          */
/*                be added                                       */
/*              position, position in list to add                */
/*                (0(HEAD),1,...,-1(TAIL)                        */
/*                                                               */
/* Returns:   const_h.SUCCESS, operation successful              */
/*              const_h.ERROR, error encountered                 */
/*---------------------------------------------------------------*/
  public int add_place( place_obj place_obj_ptr, int position )
  {
    if( position == const_h.TAIL )
    {
      place_lnklst.add( place_obj_ptr );
    }
    else
    {
      place_lnklst.add( position, place_obj_ptr );
```

```
      }

      return const_h.SUCCESS;
   } /* end of add_place */

/*-----------------------------------------------------------------*/
/* Method:    add_trans( trans_obj trans_obj_ptr, int position ) */
/*                                                                 */
/* Function:    To add a transition object to the Petri net        */
/*     transitions linked list at the designated position.        */
/*                                                                 */
/* Expects:    trans_obj_ptr, pointer to transition object to     */
/*                    be added                                     */
/*                 position, position in list to add               */
/*                    (0(HEAD),1,...,-1(TAIL)                      */
/*                                                                 */
/* Returns:    const_h.SUCCESS, operation successful              */
/*                 const_h.ERROR, error encountered                */
/*-----------------------------------------------------------------*/
   public int add_trans( trans_obj trans_obj_ptr, int position )
   {
      if( position == const_h.TAIL )
      {
         trans_lnklst.add( trans_obj_ptr );
      }
      else
      {
         trans_lnklst.add( position, trans_obj_ptr );
      }

      return const_h.SUCCESS;
   } /* end of add_trans */

/*-----------------------------------------------------------------*/
/* Method:    add_mark( mark_obj mark_obj_ptr, int position )      */
/*                                                                 */
```

```
/* Function:    To add a mark object to the Petri net                */
/*     mark linked list at the designated position.                  */
/*                                                                   */
/* Expects:    mark_obj_ptr, pointer to mark object to               */
/*                  be added                                         */
/*                position, position in list to add                  */
/*                  (0(HEAD),1,...,-1(TAIL)                          */
/*                                                                   */
/* Returns:    const_h.SUCCESS, operation successful                 */
/*                const_h.ERROR, error encountered                   */
/*-------------------------------------------------------------------*/
   public int add_mark( mark_obj mark_obj_ptr, int position )
   {
      if( position == const_h.TAIL )
      {
         web_pn_sys.local_place.mark_lnklst.add( mark_obj_ptr );
      }
      else
      {
         web_pn_sys.local_place.mark_lnklst.add( position, mark_obj_ptr );
      }

      return const_h.SUCCESS;
   } /* end of add_mark */
}
```

<place_obj.java>
プレースの情報

```
/*************************************************************/
/*                          PLACE_OBJ.JAVA                   */
/*   This file contains the class definition for the         */
/*   web_pn_sys place object.                                */
/*   Written by:    Pauline N. Kawamoto (Shinshu University) */
/*************************************************************/

/*-----------------------------------------------------------*/
```

```
/* Edit History                                                    */
/*-------------------------------------------------------------*/
/* [5/18/2004] - Quit packages.                                   */
/* [1/7/2004] - File created.                                     */
/*-------------------------------------------------------------*/

import java.awt.*;
import java.awt.event.*;
import java.util.*;

/*--------------*/
/* Place Object */
/*--------------*/
public class place_obj extends Object
{
   String     name;
   int                    type;
   int              mark;
   public elem_location_obj          location = new elem_location_obj( );

   LinkedList             out_trans_lnklst = new LinkedList( );
   LinkedList      mark_lnklst = new LinkedList();

   public place_obj( )
   {
      name = "";
      type = 0x00;
      mark = 0x00;
   } /* end of place_obj constructor */
}; /* end of place_obj class */

/*************************************************************/
/*                      PLACEREF_OBJ.JAVA                    */
/*   This file contains the class definition for the         */
/*   web_pn_sys place reference object.                       */
/*   Written by:   Pauline N. Kawamoto (Shinshu University) */
```

```
/************************************************************/

/*----------------------------------------------------------*/
/* Edit History                                             */
/*----------------------------------------------------------*/
/* [5/21/2004] - File created.                              */
/*----------------------------------------------------------*/

import java.awt.*;
import java.awt.event.*;
import java.util.*;

/*------------------------*/
/* Place Reference Object */
/*------------------------*/
public class placeref_obj extends Object
{
   public place_obj       place_obj_ptr;

   public placeref_obj( )
   {
      place_obj_ptr = null;
   } /* end of trans_ref_obj constructor */
};
```

<trans_obj.java>

トランジションの情報

```
/************************************************************/
/*                          TRANS_OBJ.JAVA                  */
/*   This file contains the class definition for the        */
/*   web_pn_sys transition object.                          */
/*   Written by:   YASUHIRO KANEKO (Shinshu University)     */
/************************************************************/

/*----------------------------------------------------------*/
/* Edit History                                             */
```

```
/*-----------------------------------------------------------*/
/* [5/28/2004] - Add input/output place linked lists.        */
/* [5/18/2004] - Quit packages.                              */
/* [1/21/2004] - Changed from transition_obj to              */
/*     trans_obj.                                            */
/* [1/12/2004] - File created.                               */
/*-----------------------------------------------------------*/

import java.awt.*;
import java.awt.event.*;
import java.util.*;

/*-------------------*/
/* Transition Object */
/*-------------------*/
public class trans_obj extends Object
{
   String name;
   public elem_location_obj          location = new elem_location_obj( );

   LinkedList              in_place_lnklst = new LinkedList( );
   LinkedList              out_place_lnklst = new LinkedList( );

   public trans_obj( )
   {
      name = "";
   } /* end of trans_obj constructor */
};

/*************************************************************/
/*                        TRANSREF_OBJ.JAVA                  */
/*   This file contains the class definition for the         */
/*   web_pn_sys transition reference object.                 */
/*   Written by:   Pauline N. Kawamoto (Shinshu University) */
/*************************************************************/
```

```
/*-------------------------------------------------------*/
/* Edit History                                          */
/*-------------------------------------------------------*/
/* [5/21/2004] - File created.                           */
/*-------------------------------------------------------*/

import java.awt.*;
import java.awt.event.*;
import java.util.*;

/*------------------------------*/
/* Transition Reference Object */
/*------------------------------*/
public class transref_obj extends Object
{
   public trans_obj       trans_obj_ptr;

   public transref_obj( )
   {
      trans_obj_ptr = null;
   } /* end of trans_ref_obj constructor */
};
```