ペトリネットを用いたシステム仕様書作成に関する考察とツールの作成

信州大学大学院　情報工学専攻
02TA525D　金子康浩

概要

- 「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。

概要

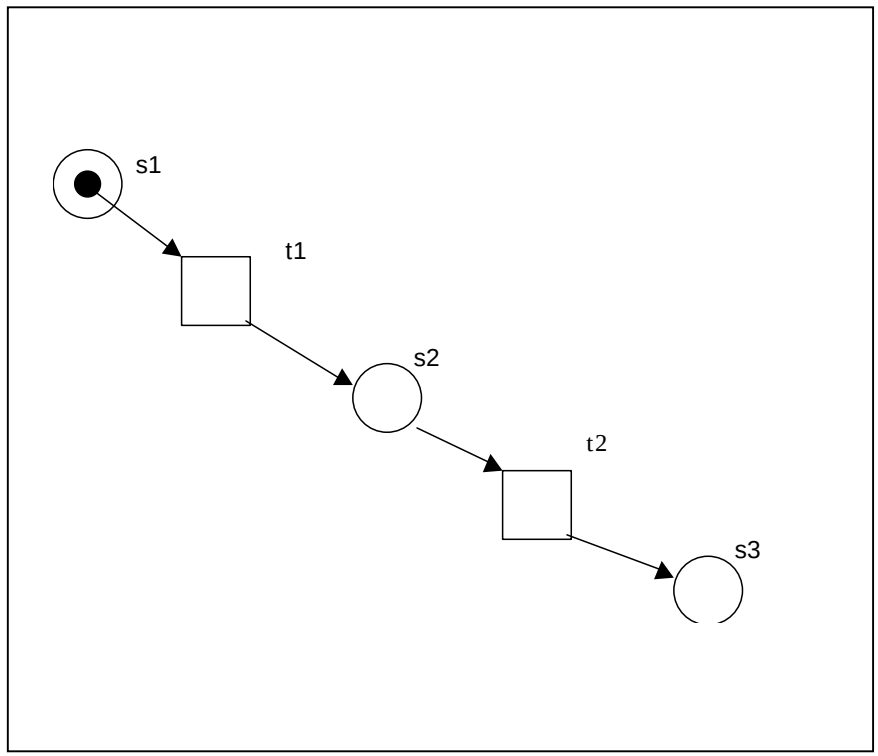
- これらの作業を自動化するために、ペトリネットからソフトウェアシステムを開発し、その仕様書もペトリネットから自動で書き出し、自然言語の文章に出来ないかということを考えてみる。本論は、ペトリネットを応用したシステム仕様書の作成方法を考察し、またそれを実現するためのツールの作成を目的としている。

はじめに

- 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。

はじめに

- 円で表現されたプレース (place)
- 四角で表現されたトランジション (transition)
- 小丸をマーク (mark) またはトークン (token) という。



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

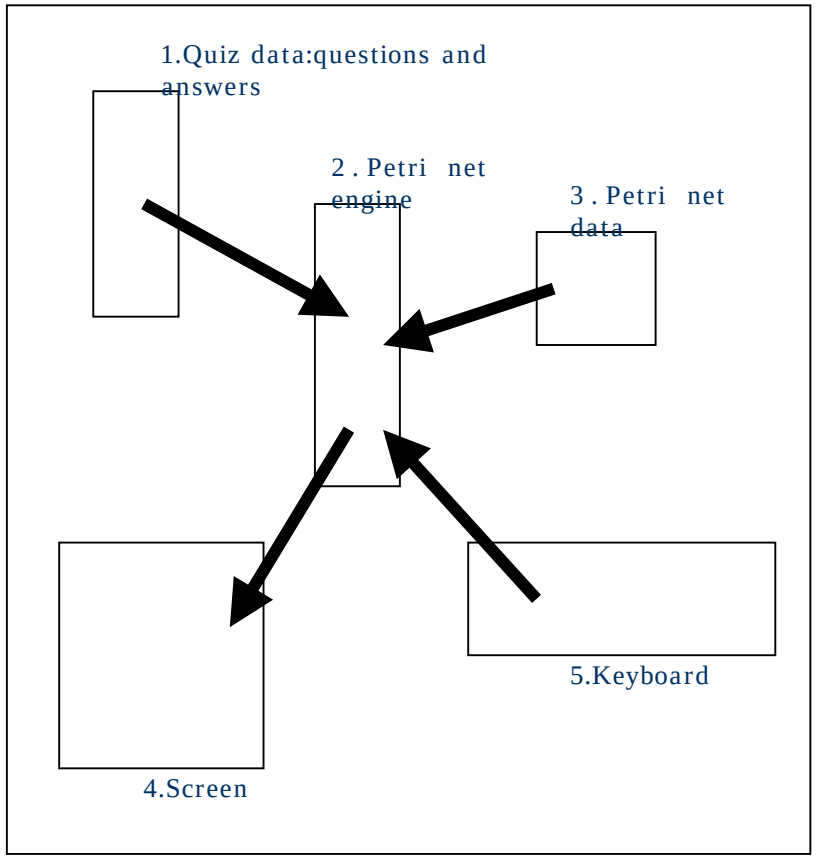
①問題に使う質問と答えが入っているファイル

②ペトリネットを「実行する」プログラム

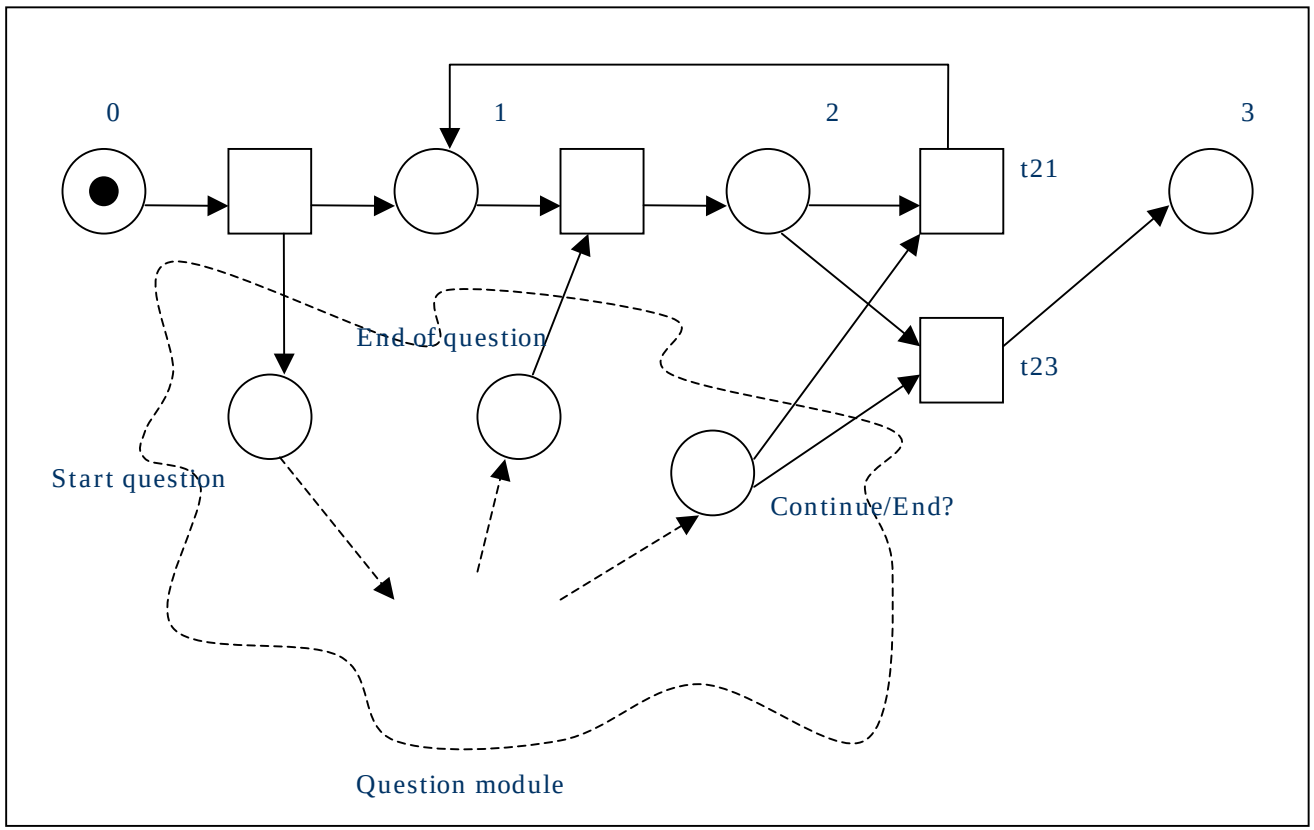
③ペトリネットの仕様が入っているデータファイル

④ユーザに出力する画面

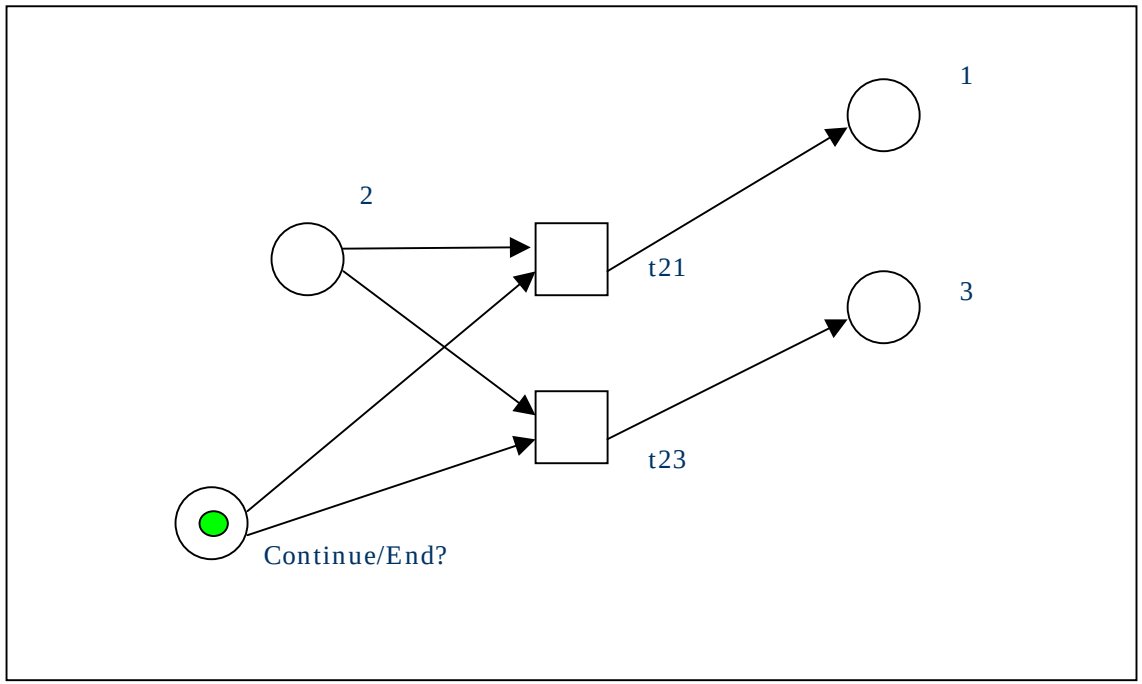
⑤ユーザからの入力を読むキーボード



クイズゲームの流れを状態で考える



拡張1：カラー（色）付きマーク



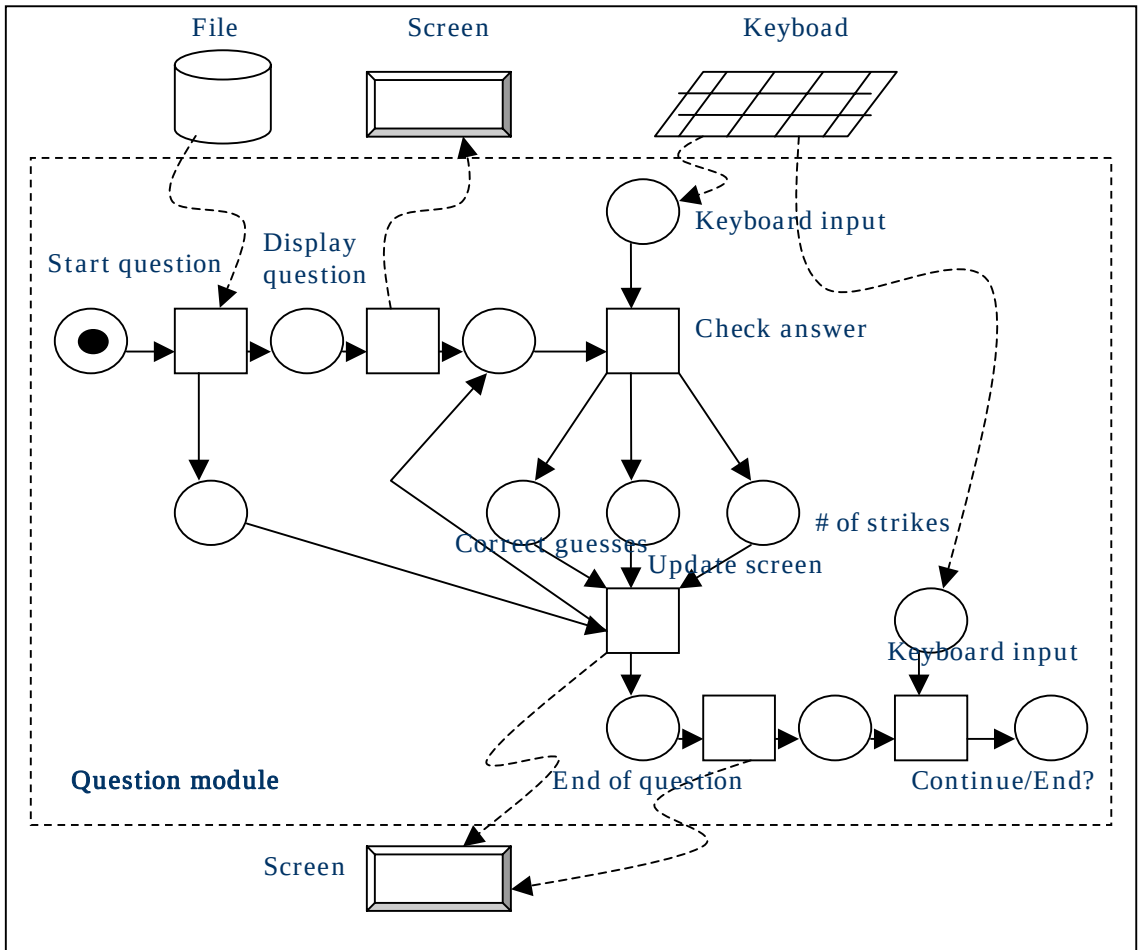
緑マーク＝次へ進む、赤マーク＝終了とすれば良い。

拡張2：トランジションの発火評価と発火処理の記述

- 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。

という定義が必要。

Question moduleの一つの考え方



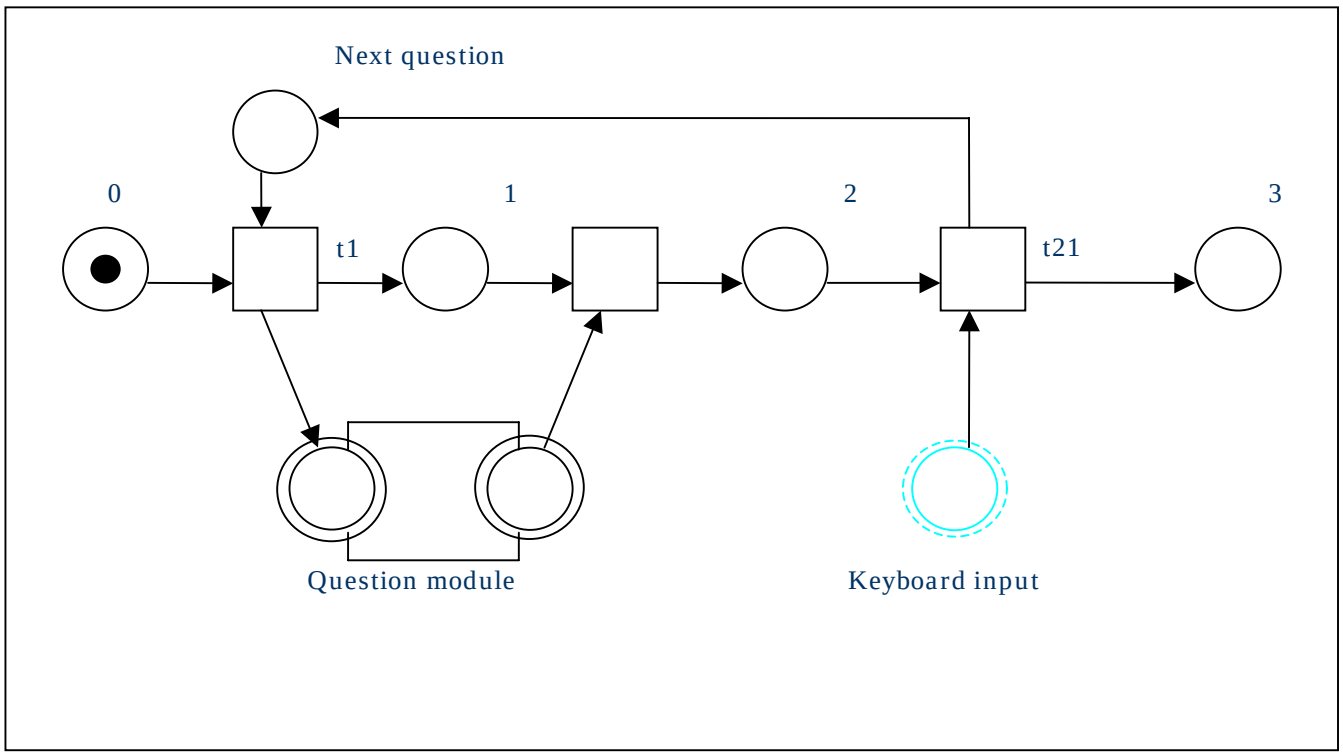
拡張3：外部インターフェース

- トランジションワーク
  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
  ワークの種類：画面出力
- 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。

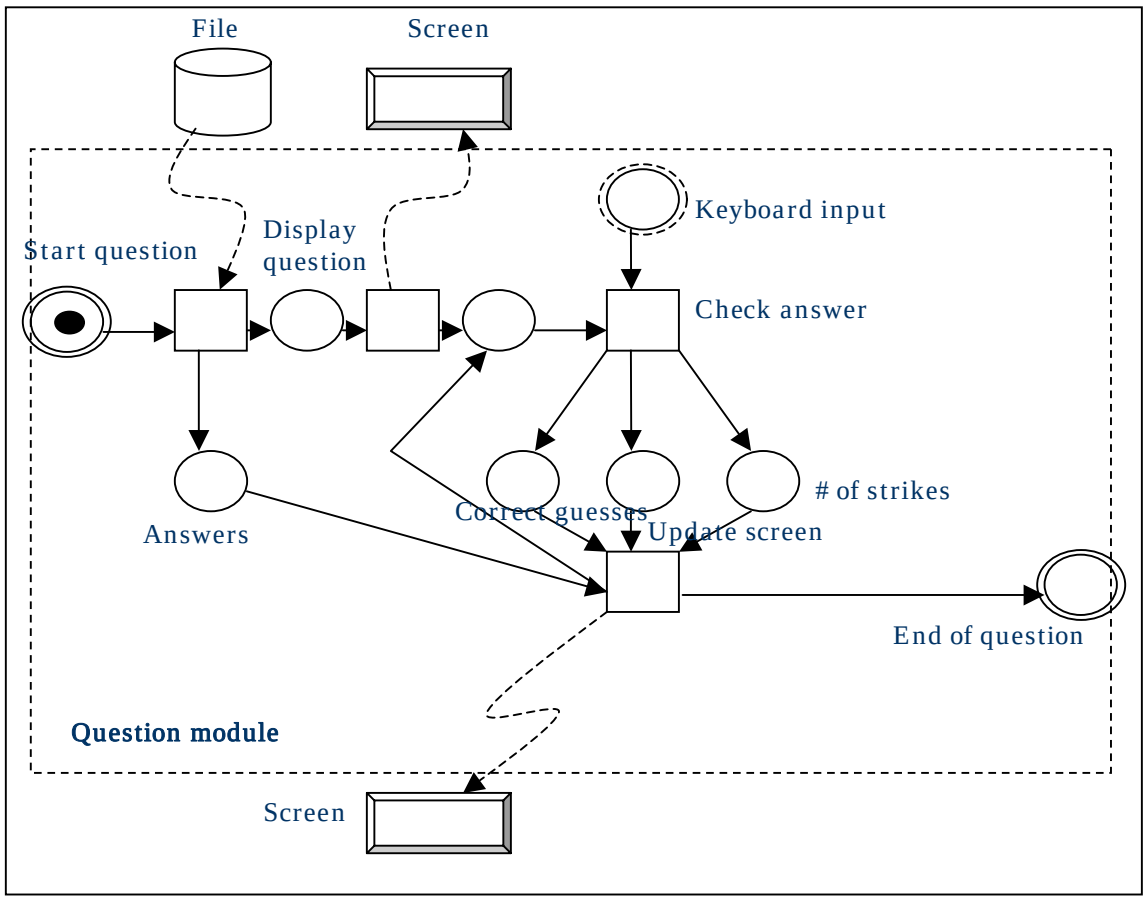
ペトリネットをわかりやすくするための工夫

- 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。
- プレースのクローン（コピー）を作成出来るようにする。
- 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。

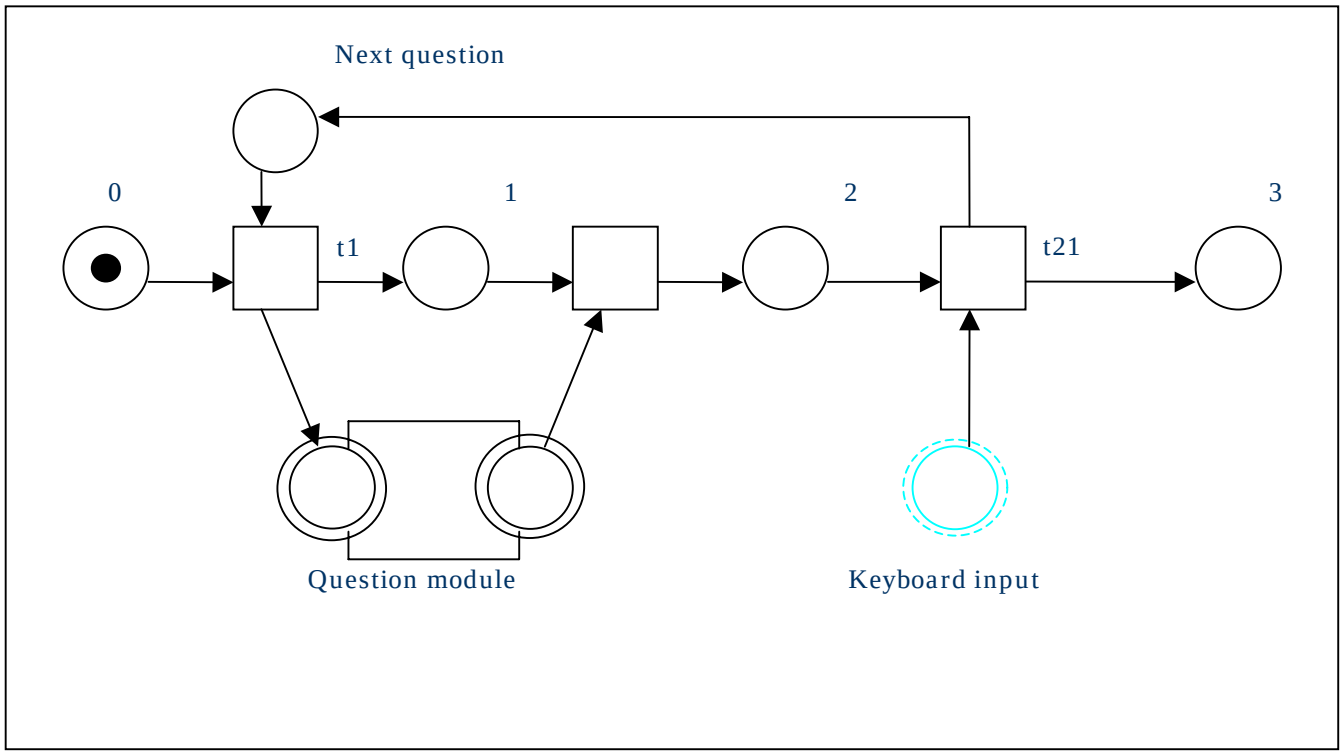
ペトリネットをわかりやすくするための工夫



Question moduleの中身を作り直す



拡張2の続き



t1の発火評価ルールについて考える。

トランジションt1の発火評価ルールを考える

- 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：m(“0”)>0

  （m( )関数は引数で指定する入力プレース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「なし」。

トランジションt1の発火評価ルールを考える

- **入力プレースのクリア**(icp)

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

- **出力プレースのセット**(OSP)

  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。**

ペトリネットで作ったシステムの仕様化

- 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。
- 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。
- 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

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

ツール作成において主に考慮すべき点

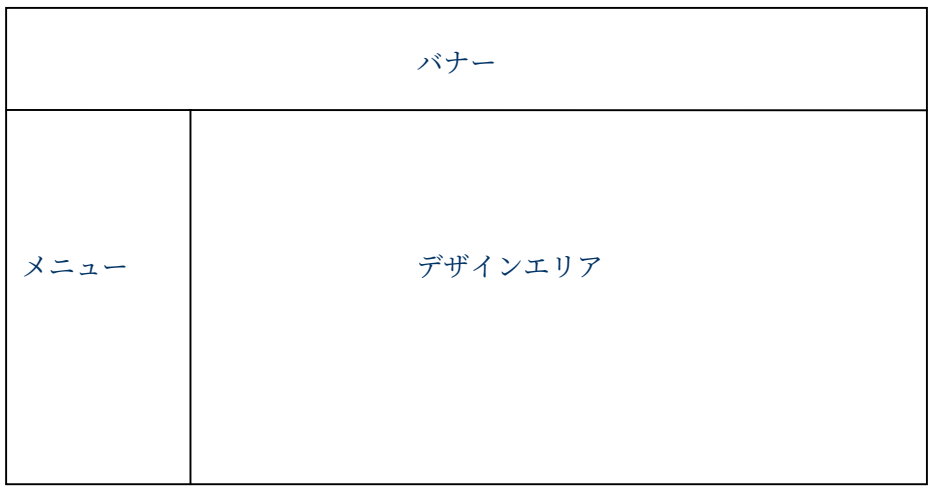
1) クラスの作り方 (全てのオブジェクトを別々のクラスにするのが良いか、いくつかは同じクラスで良いのかの判断)

2) グラフィックスの制御方法

ツールで使用しているクラスの種類

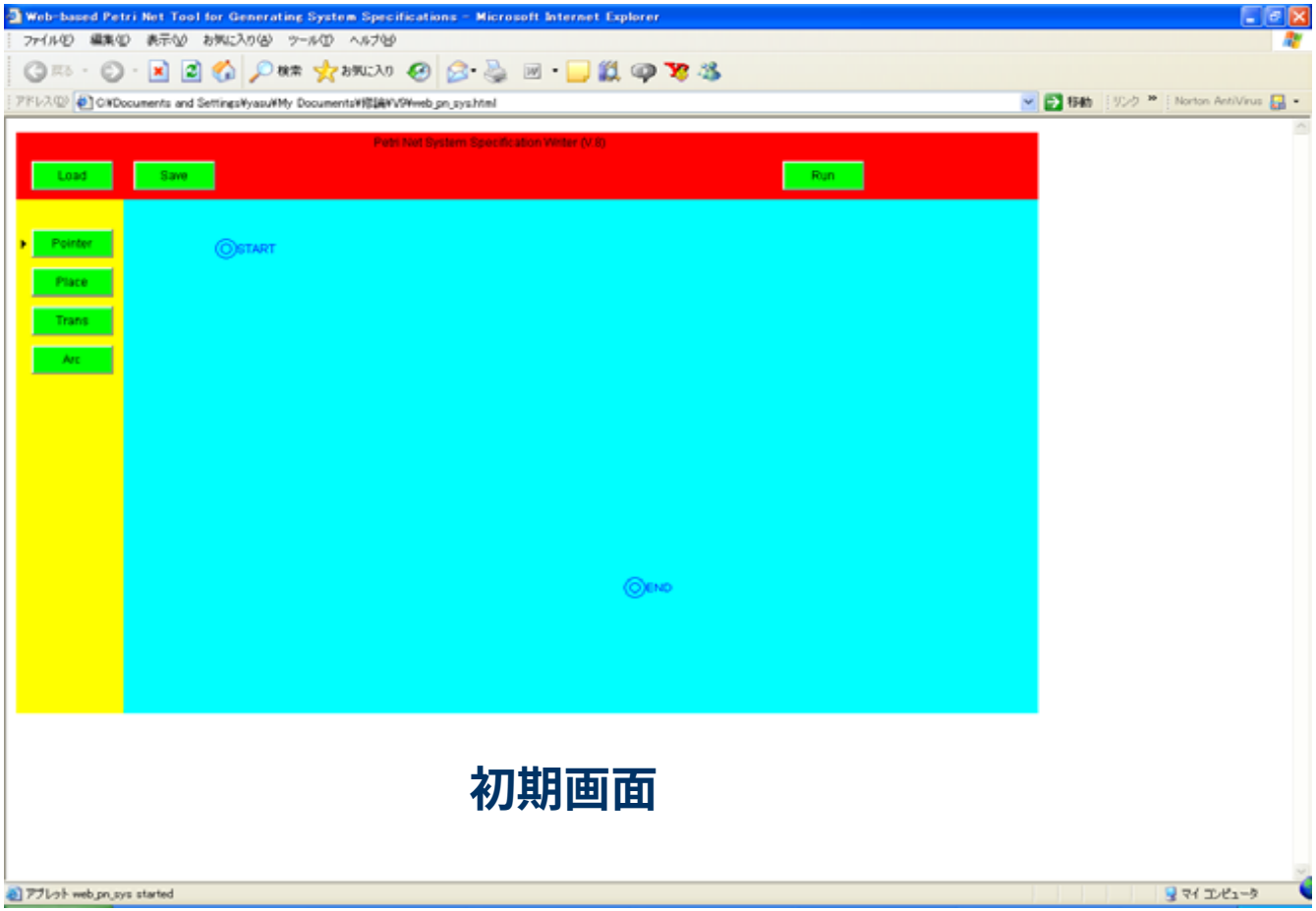
- arc_obj_component.class (アークを描く)
- const_h.class (変数の初期化)
- elem_location_obj.class (デザインエリアの位置情報)
- InputWindow.class (プレースやトランジションへのデータ入力用画面)
- main_banner_component.class (バナーの部分)
- main_mode_menu_marker_component.class (現状モードのマーカー)
- main_wdw_obj.class (画面を制御するクラス)
- mark_obj.class (マーク情報)
- module_obj.class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

グラフィックスの制御について



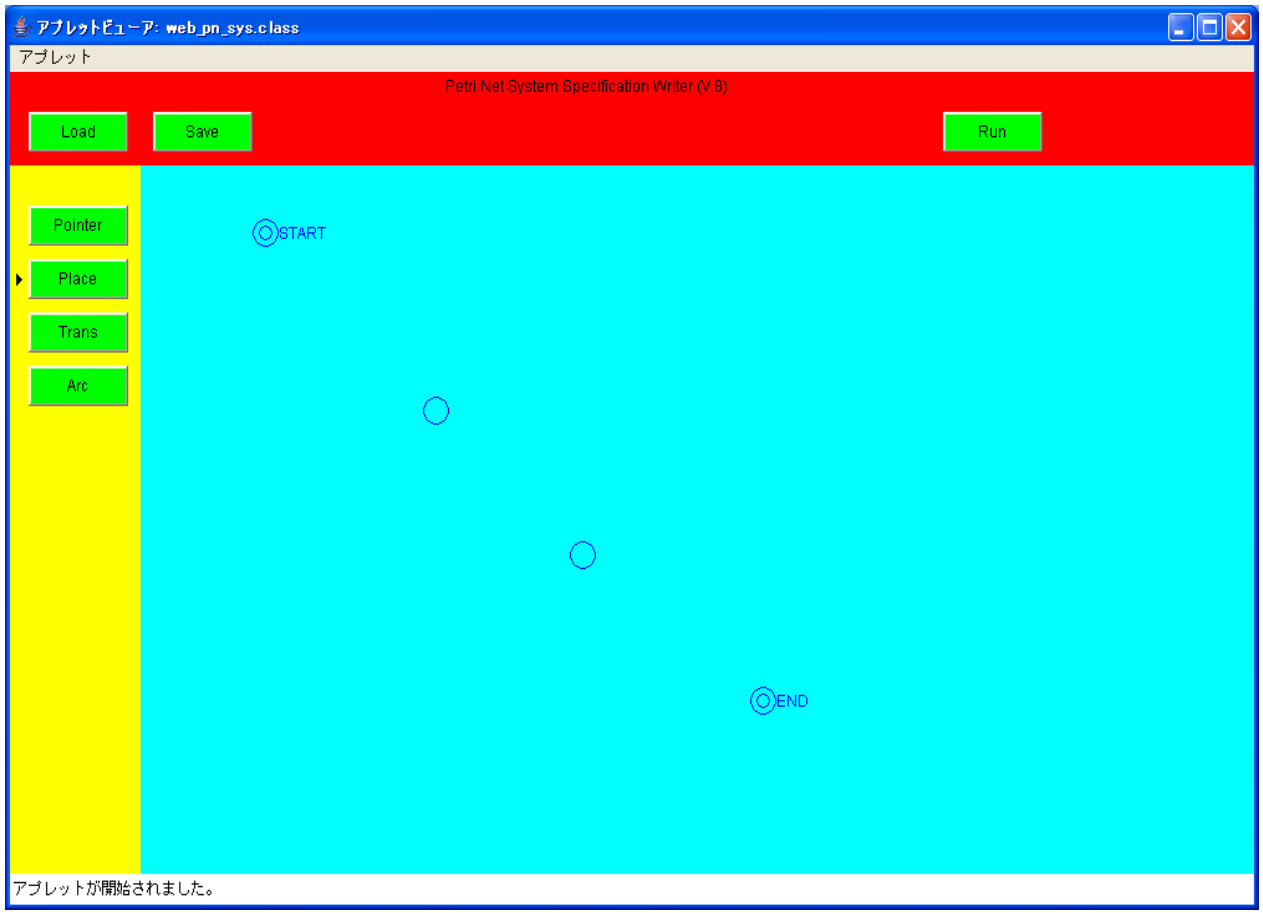
画面の構成は、「デザインエリア」、「メニュー」、「バナー」と大きく3つに分かれている

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

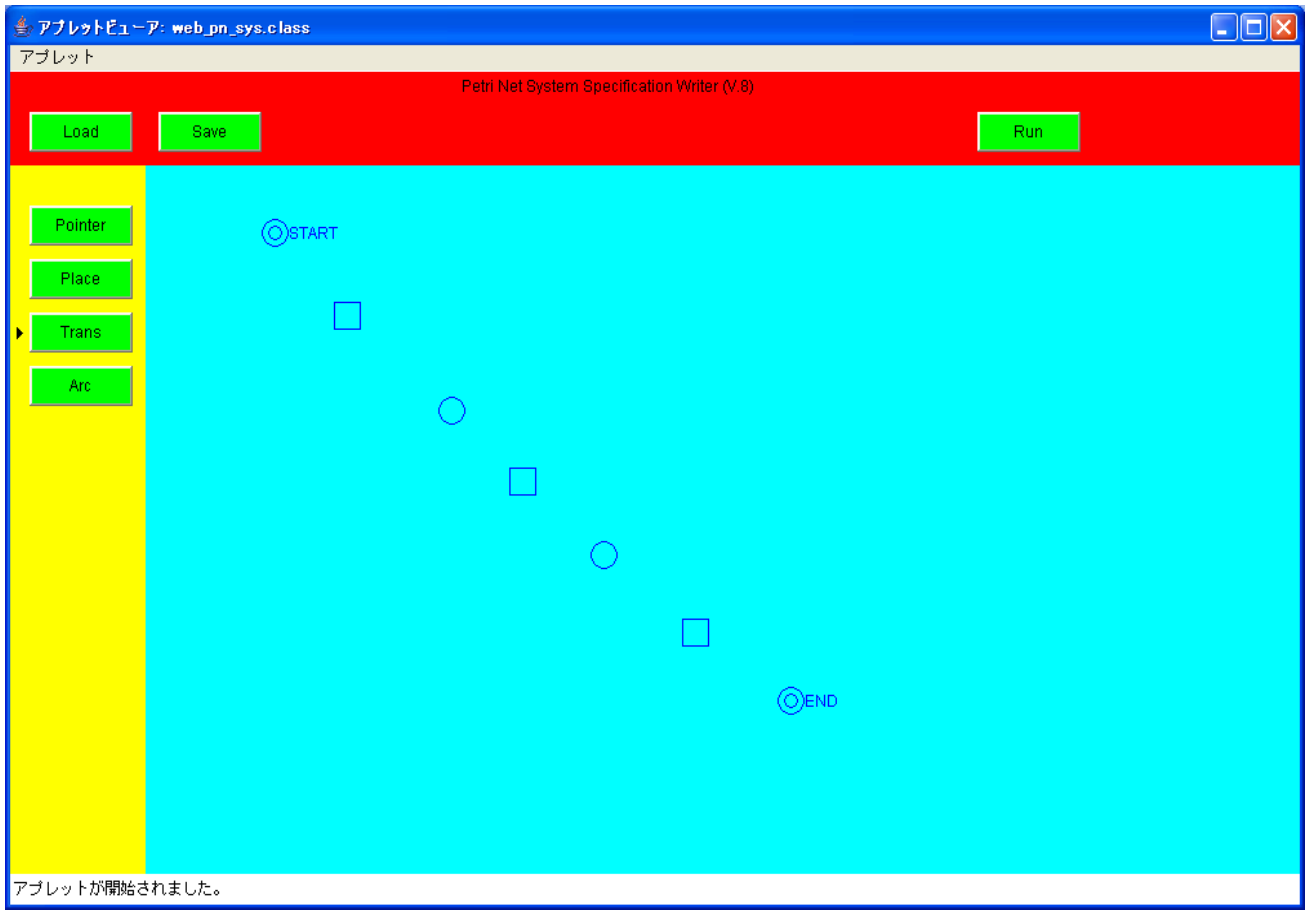


初期画面

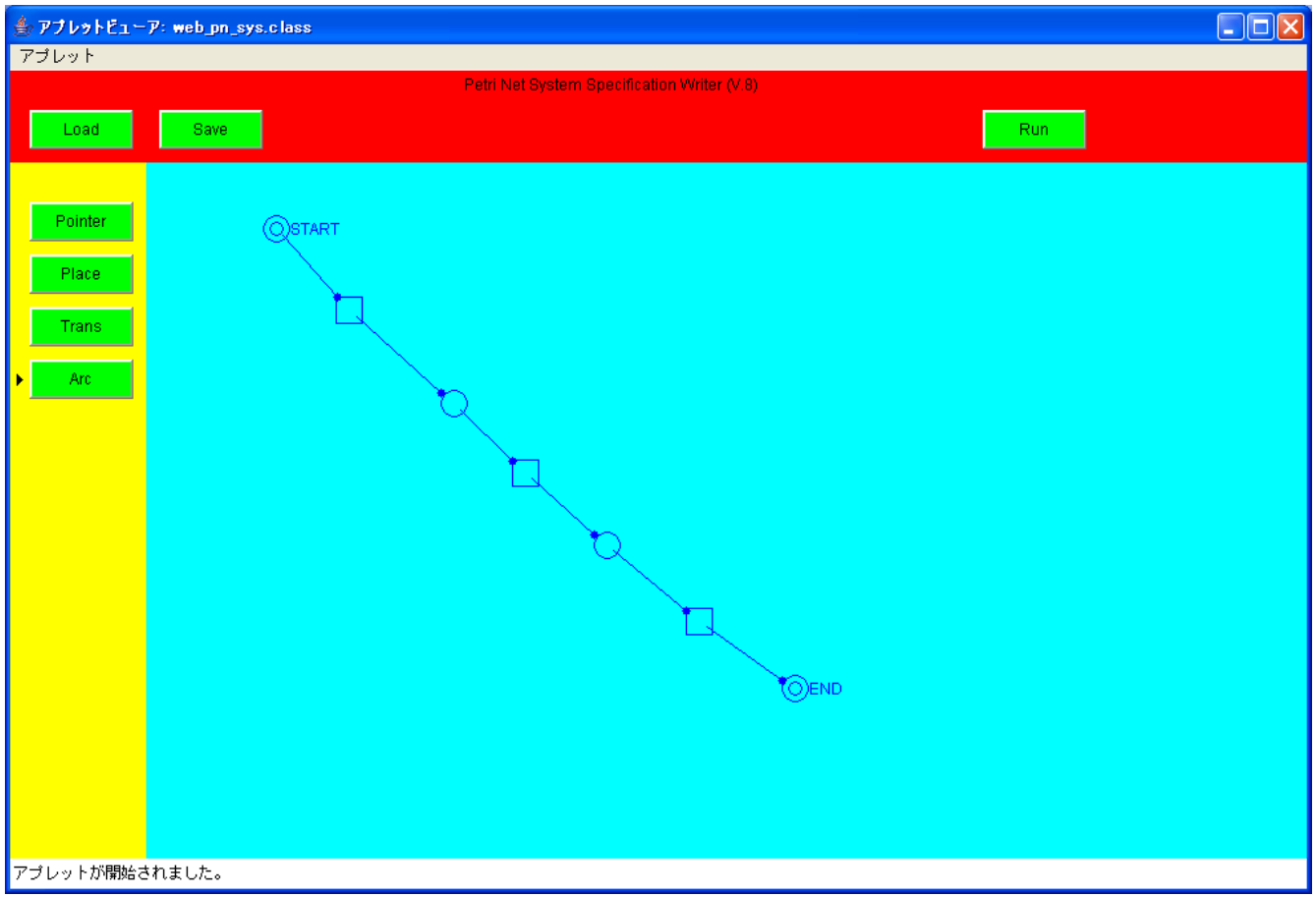
Place mode：プレース(Place)を配置する



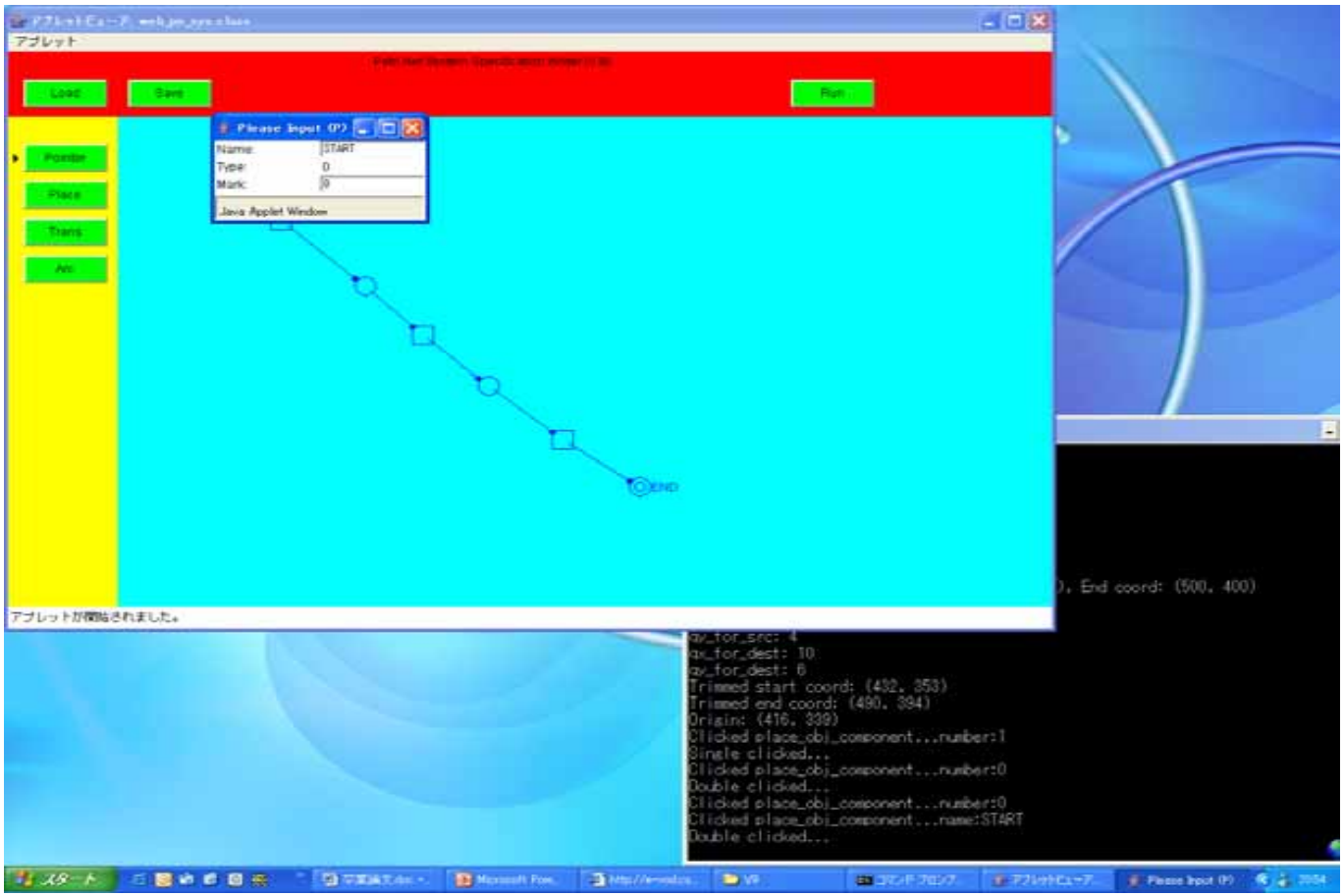
Trans mode：
トランジション(Transition)を配置する



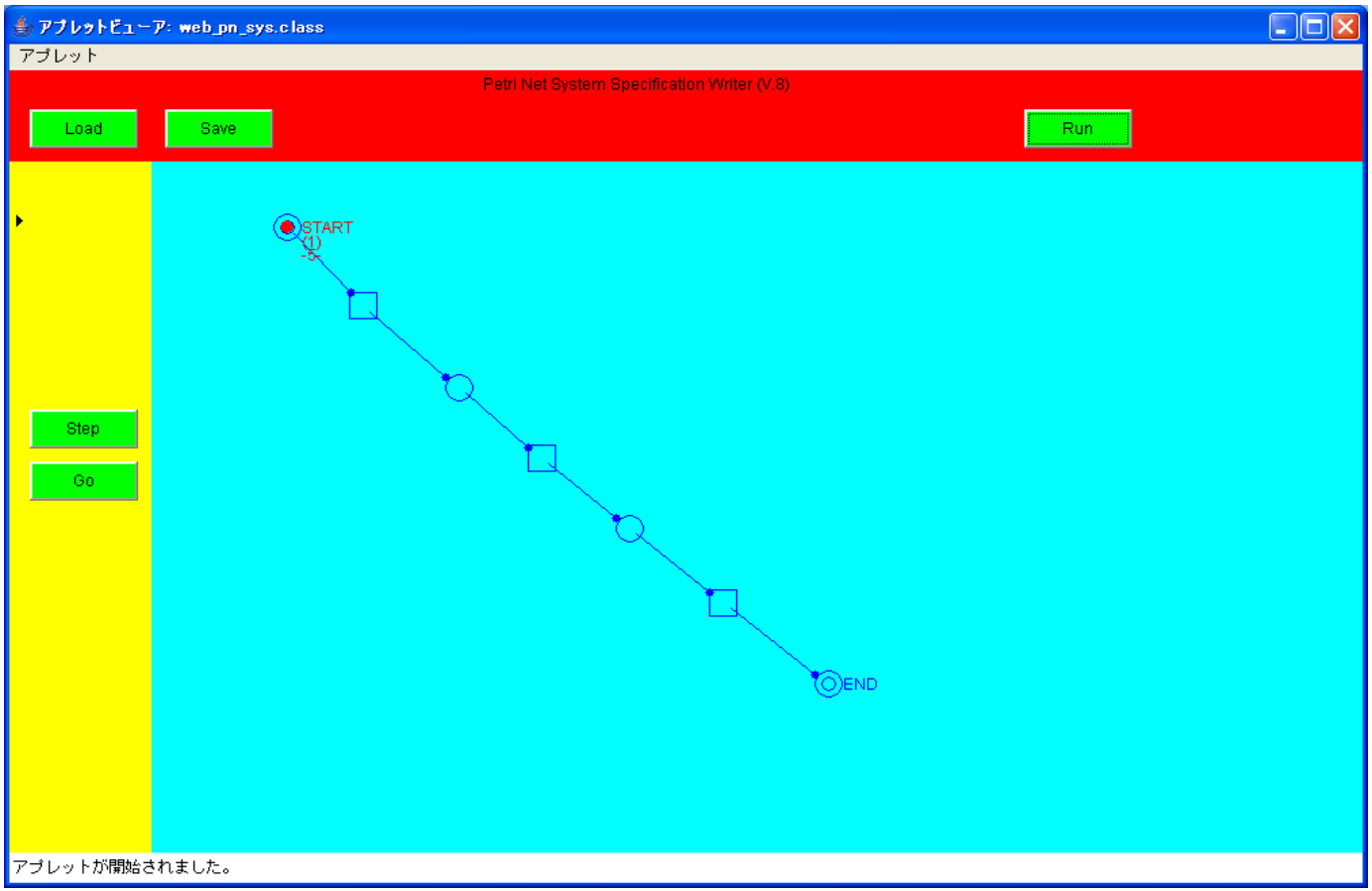
Arc mode：配置されたPlace→Trans、Trans→Place間にアークを配置する



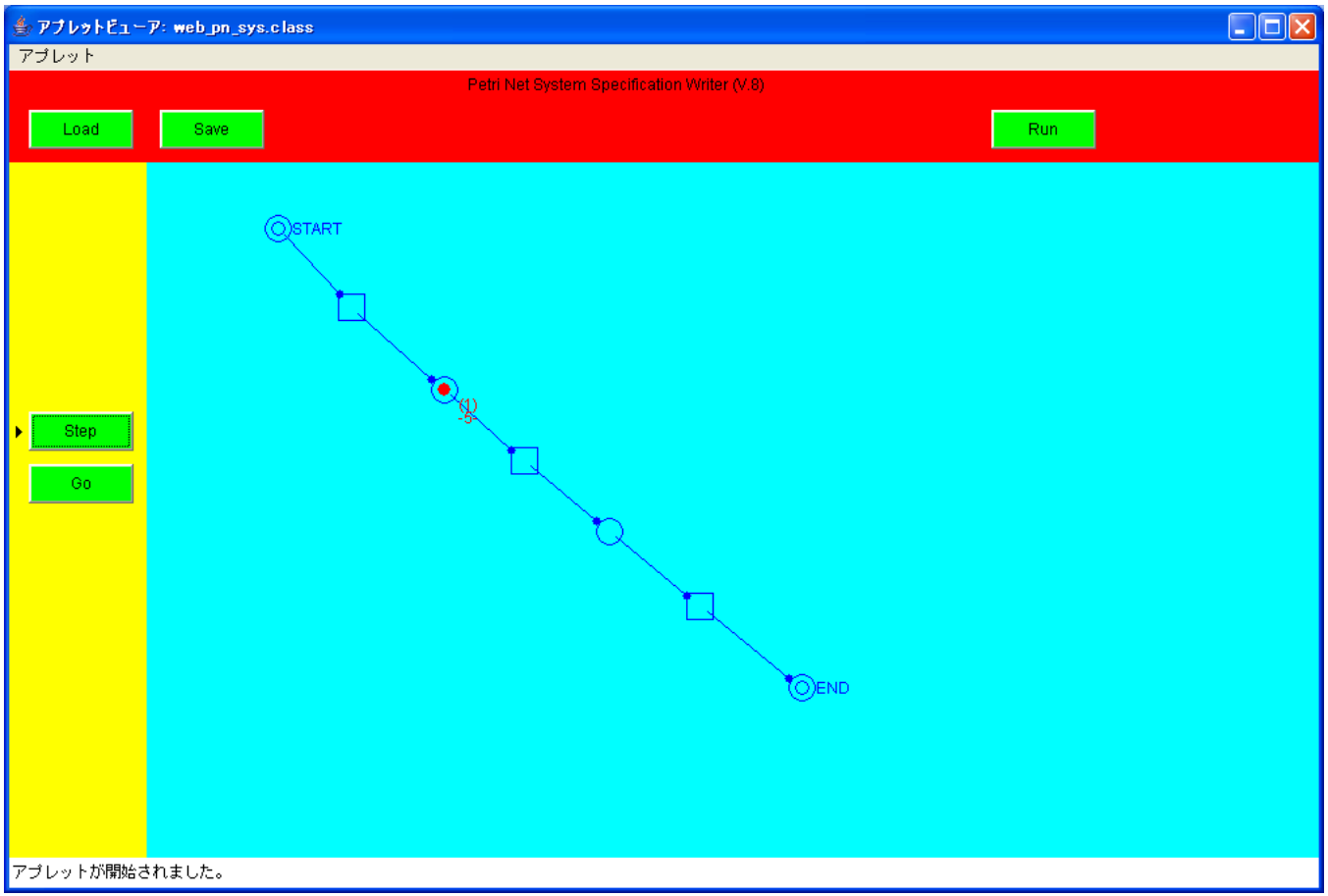
Pointer mode (初期値)：



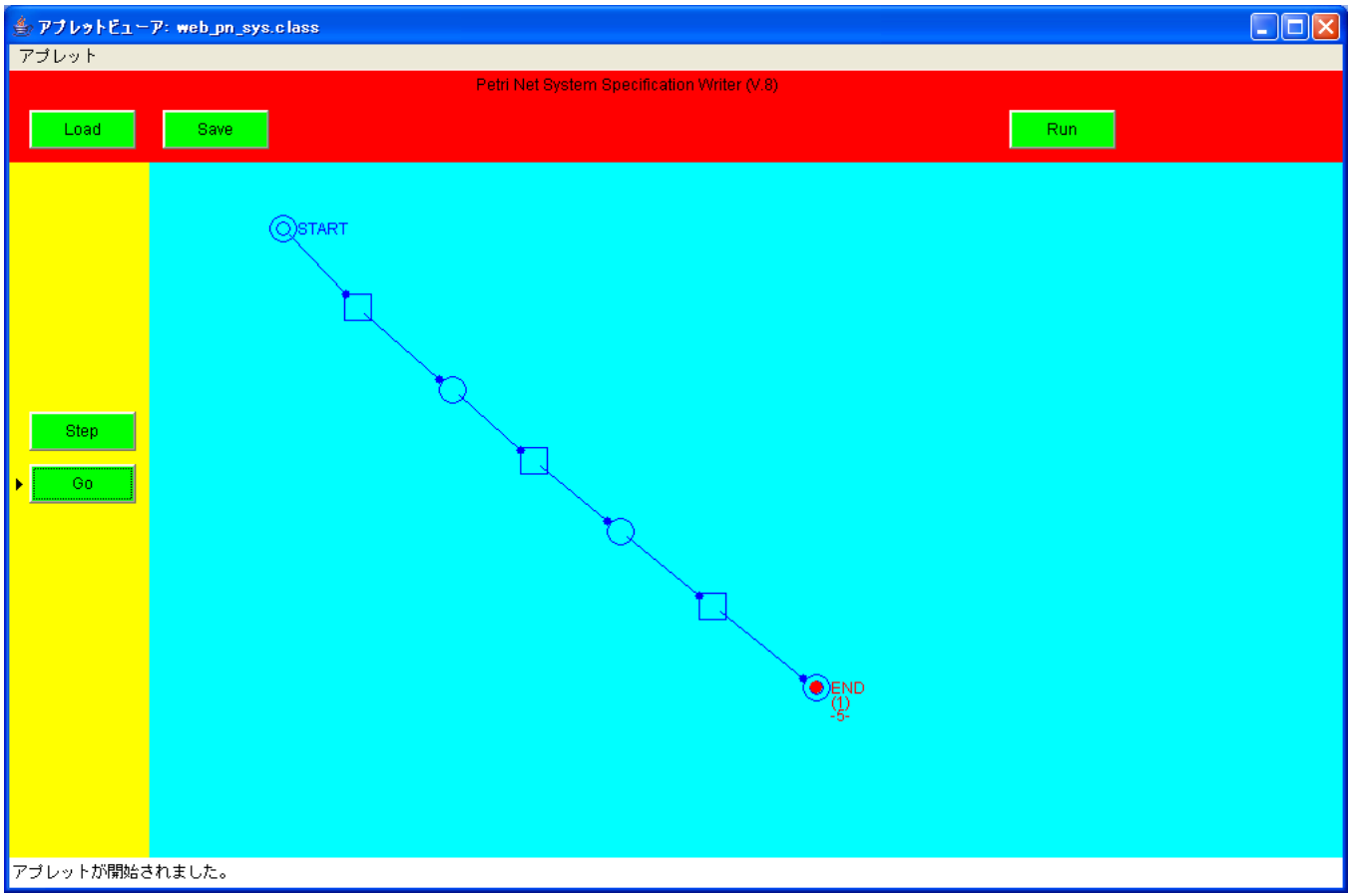
Run mode：実行モードに切り替わる



Step (実行)


アプレットビューア: web_pn_sys.class
アプレット
Petri Net System Specification Writer (V.8)
Load
Save
Run
START
Step
Go
END
アプレットが開始されました。

Go (実行)


アプレットビューア: web_pn_sys.class
アプレット
Petri Net System Specification Writer (V.8)
Load
Save
Run
START
Step
Go
END
(1)
-5-
アプレットが開始されました。

結び

ツールの今後の改良点

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の表示。

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。

結び

- 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 。
- 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。

ご清聴ありがとうございました。