Design and implementation Petri net editor

Assist Lecture . Mazin Haithem AL- Shaikhly
Assist Lecture . Zaid Abass Fadahi
Baghdad College of Economic Sciences University

Abstract

Many modeling editor(modeling and simulation “systemwiki.org”, online diagram ,…etc) but without operate your design In this research design and implementation dynamic modeling (Petri net) tools(A **dynamic model** represents the behavior of an object over time " **dynamic model** are: States") using one of popular dynamic modeling tool (The Petri modeling formalism allows for the convenient graphical visualization of system models, as well as the analysis of correctness and performance properties.) rules as main attribute for editor with accept graph only by automatic correction (active editor In this editor provides a flexible create of model without any error in main rule),test graph result and firing step by step , good agreement achieved with implement package

Key word :Petri net ,active editor ,fire ,save step

المستخلص

العديد من محررات النمذجة والمحاكاة "systemwiki.org" ، والرسم على الانترنت ... الخ)تعتمد فقط لرسم المخططات ولكن من دون تشغيل التصميم المقترح للنظام في هذاالبحث نقوم بتصميم و تنفيذ المخططات الخاصة بنا وفق تصميم نموذج ديناميكي يمثل سلوك النظام خلل واحدة الزمن. النموذج الديناميكي هو تحول القواعد و الاوامر الى مخططات رسومية للنظام المقترح . يوفر النظام المقترح تصحيح تلقائي (المحرر يوفر مرونة في بناء نموذج من دون أي خطأ في القاعدة الرئيسية)، من خلل اختبار نتيجة الرسم و تنفيذه خطوة بخطوة،و اعطى نتائج جيدة عند تنفيذ الحزمة .

1. Introduction

Modeling and simulation (M&S) is getting information about how something will behave without actually testing it in real life. For instance, if we wanted to design a race car, but weren't sure what type of spoiler would improve traction the most, we would be able to use a computer simulation of the car to estimate the effect of different spoiler shapes on the coefficient of friction in a turn. We're getting useful insights about different decisions we could make for the car without actually building the car.[1,2].
(Blei et al., 2003; McCallum et al., 2004; Rosen-Zvi et al., 2004; Grif- fiths and Steyvers, 2004; Buntine and Jakulin, 2004; Blei and Lafferty, 2006). These models are called “topic models” because the discovered patterns often reflect the underlying topics which combined to form the documents. Such hierarchical probabilistic models are easily generalized to other kinds of data; for example, topic models have been used to analyze images (Fei-Fei and Perona, 2005; Sivic et al., 2005), biological data (Pritchard et al., 2000), and survey data (Erosheva, 2002). Section 2 Perti net Dynamic modeling tools , Section 3 editor definition and main activity , Section 4 petri net active editor algorithm and implementation , Section 5 experiment result ,Section 6 conclusion and feature work .

2. Perti net Dynamic modeling tools

A Petri net consists of *places*, *transitions*, and *arcs*. Arcs run from a place to a transition or vice versa, never between places or between transitions. The places from which an arc runs to a transition are called the *input places* of the transition; the places to which arcs run from a transition are called the *output places* of the transition.

Graphically, places in a Petri net may contain a discrete number of marks called *tokens*. Any distribution of tokens over the places will represent a configuration of the net called a *marking*. In an abstract sense relating to a Petri net diagram, a transition of a Petri net may *fire* if it is *enabled*, *i.e.* there are sufficient tokens in all of its input places; when the transition fires, it consumes the required input tokens, and creates tokens in its output places. A firing is atomic, i.e., a single non interruptible step.

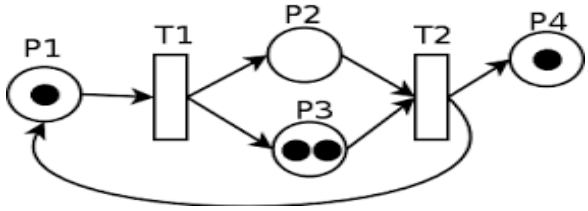


Figure 1:petrinet model

Unless an *execution policy* is defined, the execution of Petri nets is nondeterministic: when multiple transitions are enabled at the same time, any one of them may fire.

Since firing is nondeterministic, and multiple tokens may be present anywhere in the net (even in the same place), Petri nets are well suited for modeling the concurrent behavior of distributed systems.
With another word A **Petri net** (also known as a place/transition **net** or P/T **net**) is one of several mathematical modeling languages for the description of distributed systems.
A Petri net is formally defined as a 5-tuple N = (P, T, I, O, M0),
where
(1) P = {p1, p2, …, pm} is a finite set of places;
(2) T = {t1, t2, …, tn} is a finite set of transitions, $P \cup T \neq \emptyset$, and $P \cap T = \emptyset$;
(3) I: $P \times T \rightarrow N$ is an input function that defines directed arcs from places to transitions, where N is a set of nonnegative integers;
(4) O: $T \times P \rightarrow N$ is an output function that defines directed arcs from transitions to places; and (5) M0: $P \rightarrow N$ is the initial marking.

A marking in a Petri net is an assignment of tokens to the places of a Petri net. Tokens reside in the places of a Petri net. The number and position of tokens may change during the execution of a Petri net. The tokens are used to define the execution of a Petri net.
Transition Firing The execution of a Petri net is controlled by the number and distribution of tokens in the Petri net. By changing distribution of tokens in places, which may reflect the occurrence of events or execution of operations, for instance, one can study the dynamic behavior of the modeled system. A Petri net executes by firing transitions. We now introduce the enabling rule and firing rule of a transition, which

govern the flow of tokens: (1) Enabling Rule: A transition t is said to be enabled if each input place p of t contains at least the number of tokens equal to the weight of the directed arc connecting p to t, i.e., $M(p) \geq I(t, p)$ for any p in P. (2) Firing Rule: Only enabled transition can fire. The firing of an enabled transition t removes from each input place p the number of tokens equal to the weight of the directed arc connecting p to t. It 3 also deposits in each output place p the number of tokens equal to the weight of the directed arc connecting t to p. Mathematically,

firing t at M yields a new marking
$M'(p) = M(p) - I(t, p) + O(t, p)$ for any p in P.

Modeling Power The typical characteristics exhibited by the activities in a dynamic event-driven system, such as concurrency, decision making, synchronization and priorities, can be modeled effectively by Petri nets. 1. Sequential Execution. In Figure 2(a), transition t2 can fire only after the firing of t1. This imposes the precedence constraint "t2 after t1." Such precedence constraints are typical of the execution of the parts in a dynamic system. Also, this Petri net construct models the causal relationship among activities. 2. Conflict. Transitions t1 and t2 are in conflict in Figure 2(b). Both are enabled but the firing of any transition leads to the disabling of the other transition. Such a situation will arise, for example, when a machine has to choose among part types or a part has to choose among several machines. The resulting conflict may be resolved in a purely non-deterministic way or in a probabilistic way, by assigning appropriate probabilities to the conflicting transitions.

3. Concurrency. In Figure 2(c), the transitions t1, and t2 are concurrent. Concurrency is an important attribute of system interactions. Note that a necessary condition for transitions to be concurrent is the existence of a forking transition that deposits a token in two or more output places.

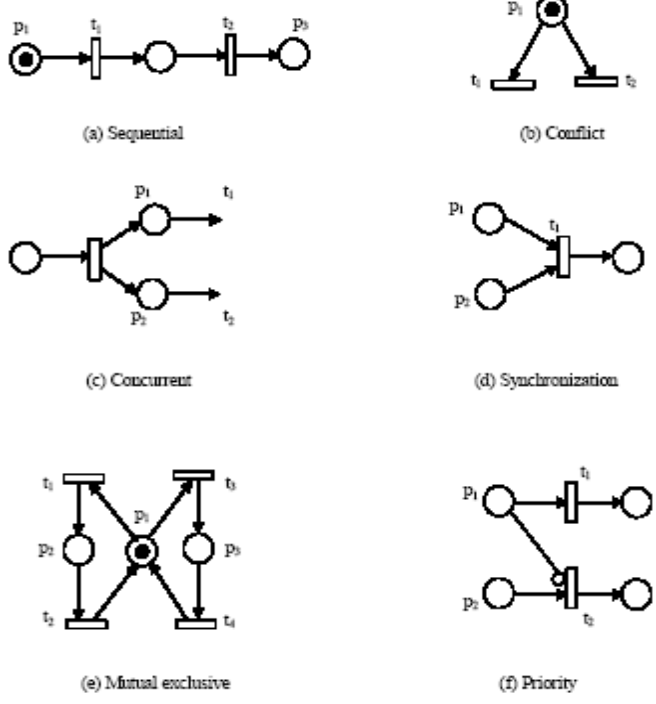


Figure 2,petrinet case

4. Synchronization. It is quite normal in a dynamic system that an event requires multiple resources. The resulting synchronization of resources can be captured by transitions of the type shown in Figure 2(d). Here, t1 is enabled only when each of p1 and p2 receives a token. The arrival of a token into each of the two places could be the result a possibly complex sequence of operations elsewhere in the rest of the Petri net model. Essentially, transition t1 models the joining operation.
5. Mutually exclusive. Two processes are mutually exclusive if they cannot be performed at the same time due to constraints on the usage of shared resources. Figure 2(e) shows this structure. For example, a robot may be shared by two machines for loading and unloading. Two such structures are parallel mutual exclusion and sequential mutual exclusion. t1 p t2 1 p1 p3 t1 t2 (a) Sequential (b) Conflict t1 t2 p1 p2 t1 (c) Concurrent (d) Synchronization p2 p1 t1 t2 p2 t1 (e) Mutual exclusive (f) Priority p2 p1 t2 p3 t3 t4 p1 Figure 2 Petri net primitives to represent system features.
6. Priorities. The classical Petri nets discussed so far have no mechanism to represent priorities. Such a modeling power can be achieved by introducing an inhibitor arc. The inhibitor arc connects an input place to a transition, and is pictorially represented by an arc terminated with a small circle. The presence of an inhibitor arc connecting an input place to a transition changes the transition enabling conditions. In the presence of the inhibitor arc, a transition is regarded as enabled if each input place, connected to the 5 transition by a normal arc (an arc terminated with an arrow), contains at least the number of tokens equal to the weight of the arc, and no tokens are present on each input place connected to the transition by the inhibitor arc. The transition firing rule is the same for normally connected places. The firing, however, does not change the marking in the inhibitor arc connected places. A Petri net with an inhibitor arc is shown in Figure 2(f). t1 is enabled if p1 contains a token, while t2 is enabled if p2 contains a token and p1 has no token. This gives priority to t1 over t2.

3- editor definition and main activity

Sometimes called *text editor*, a program that enables you to create and edit text files. There are many different types of editors, but they all fall into two general categories:

 line editors: A primitive form of editor that requires you to specify a specific line of text before you can make changes to it.

 screen -oriented editors: Also called *full-screen editors* ,these editors enable you to modify any text that appears on the display screen by moving the cursor to the desired location.

The distinction between editors and word processors is not clear-cut, but in general, word processors provide many more formatting features. Nowadays, the term editor usually refers to source code editors that include many special features for writing and editing source code [7].

Editor is a computer program that permits the user to create or modify data as text or graphics especially on a display screen. Also there is a text editor to define as any word processing program that can use to type and edit text. One can't call it a text editor for

nothing. WordPad and notepad for windows and simple text and text editor for Macintosh are common text editors. Larger programs such as Microsoft word and word perfect are also text editors, but they have many more features. We can actually write HTML code and create HTML pages with simple text editor; one can know the correct HTML syntax [3].

4- petri net active editor algorithm and implementation

Many editor using to design Petri net but without show transition table or no fire control with following algorithm support (fire "run diagram", transition table, and auto correct)

Algorithm 1

Input (mouse click, keyboard)
Output(diagram "transition ,place ,arc and token", fire , save)
1- Start
2- Chick input source and analyses
a-mouse
-right click show sub menu (transition , place ,arc , token and information about current board).
- function bar icon clear, save, open, Fire, info. about editor
b-function key
F1 info F2 fire Esc cancel action .
d- quit goto step 4
3- Goto 2
4- end

Note attempt to see screen appendix

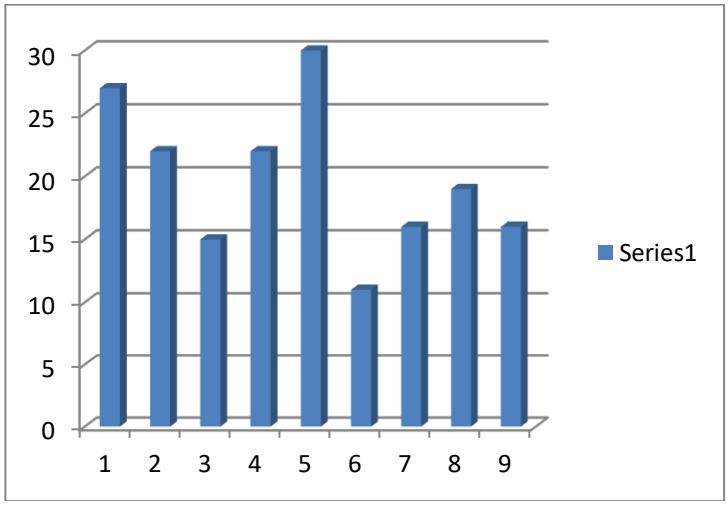
algorithm 2
input (function key F2)
output (can fire)
1- start
2- if graph available then test rule
3- if test then can fire else error "No fire"
4- end

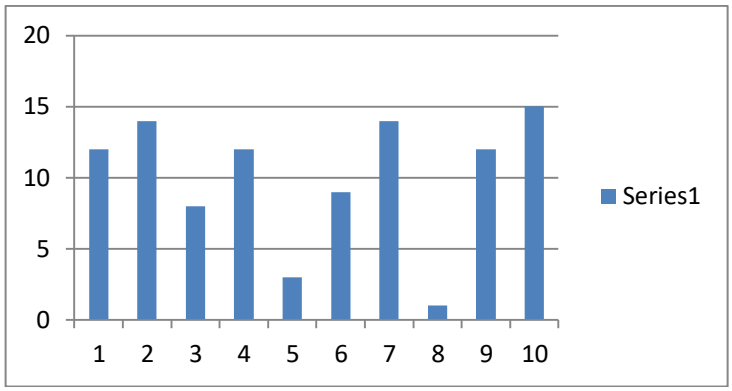
```
algorithm 3
input (can fire)
output (fire)
    1- start
    2- select  Petri net case
            implement petri net case available
       other wise
            no action
       end select
    3- end
```

```
algorithm 4
input (Icon click)
output ( implement action )
    1- start
    2- check which icon click
    3- select icon depend on click
    4- implement icon function goto main
    5- end
```

5- experiment result

For 10 design with editor compare with 10 design without active editor (another editor type) the differ is clear value of series 1 , 2 have new graph without active no fire no accept design and graph series 2 is better then 1 in time ,fire and flexibility(save ,retrieve).





6- conclusion and feature work

1- all result give good indicate and high performance in run.
2- in active editor all graph must correct thus win time .
3- active editor with save function restore graph and update in any time need .
4- with fire function test graph and implementation it.
5- Print function document design as hard copy.
6- Arc place table and fire state table give view for design behavior.

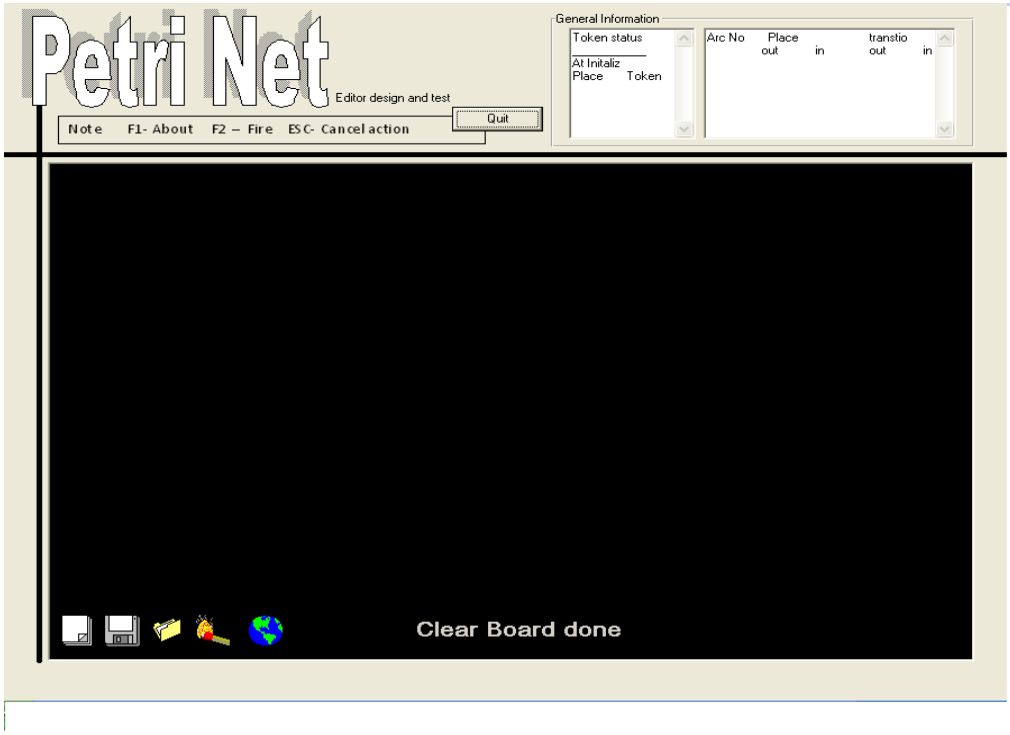
Feature work

1- implement all petri net case
2- add reach ability tree for design after set token .

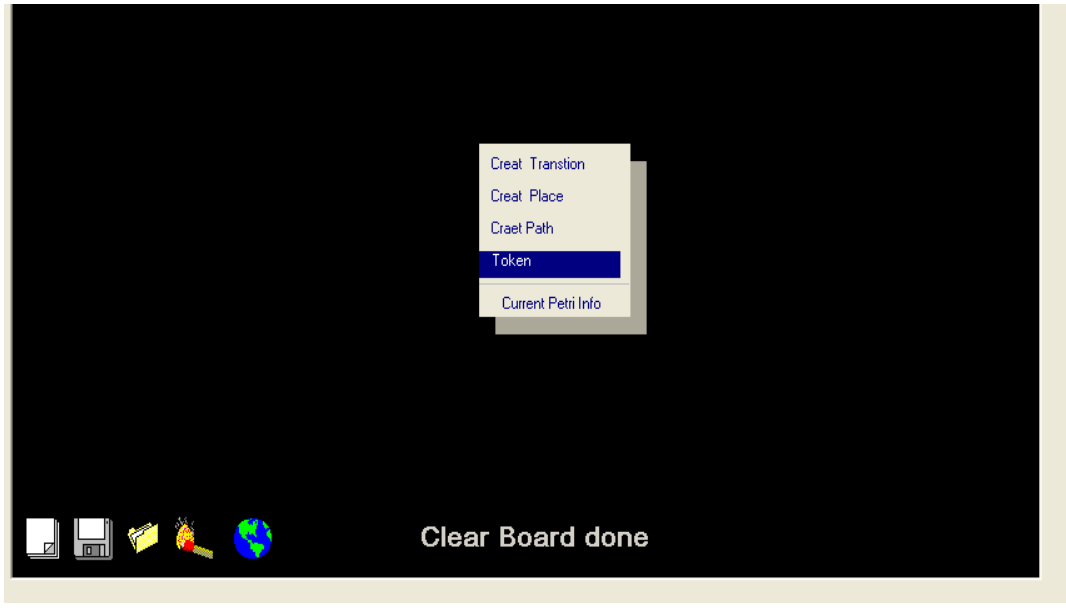
References

1. "Department of Defense Modeling and Simulation the term as be defined(M&S) Glossary", DoD 5000.59-M, Department of Defense, 1998
2. **Jump up^** National Science Foundation (NSF) Blue Ribbon Panel (2007). Report on Simulation-Based Engineering Science: Revolutionizing Engineering Science through Simulation. NSF Press, May
3. Available at http://www.wisegeek.com.
4. *Petri, Carl Adam; Reisig, Wolfgang (2008). "Petri net".Scholarpedia **3** (4): 6477. doi:10.4249/scholarpedia.6477.*
5. **Jump up^** *Rozenburg, G.; Engelfriet, J. (1998). "Elementary Net Systems". In Reisig, W.; Rozenberg, G. Lectures on Petri Nets I: Basic Models - Advances in Petri Nets. Lecture Notes in Computer Science **1491**. Springer. pp. 12–121.*
6. **Jump up^** *Reisig, Wolfgang (1991). "Petri Nets and Algebraic Specifications". Theoretical Computer Science **80** (1): 1–34. doi:10.1016/0304-3975(91)90203-e.*
7. **Jump up^** *Desel, Jörg; Juhás, Gabriel (2001). "What Is a Petri Net? Informal Answers for the Informed Reader". In Ehrig, Hartmut; et al. Unifying Petri Nets. LNCS **2128**. Springerlink.com. pp. 1–25. Retrieved 2014-05-14.*
8. https://www..wikipedia.org%2Fwiki%2FPetri_net&usg=AFQjCNGjKRRSnCOxlYfIDS60FzvnmpZ7ZA
9. Bause, F., and P. Kritzinger. 2002. Stochastic Petri Nets -- An Introduction to the Theory. Germany: Vieweg Verlag.
10. Desrochers, A., and R. Ai-Jaar. 1995. Applications of Petri Nets in Manufacturing Systems: Modeling, Control and Performance Analysis. IEEE Press.

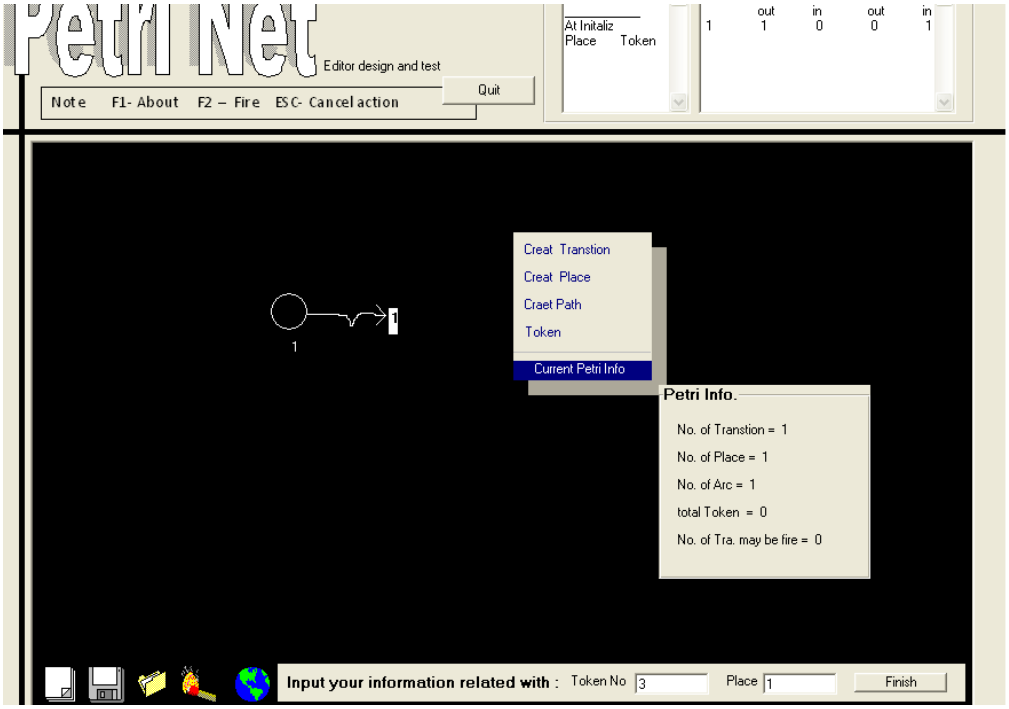
Appendix



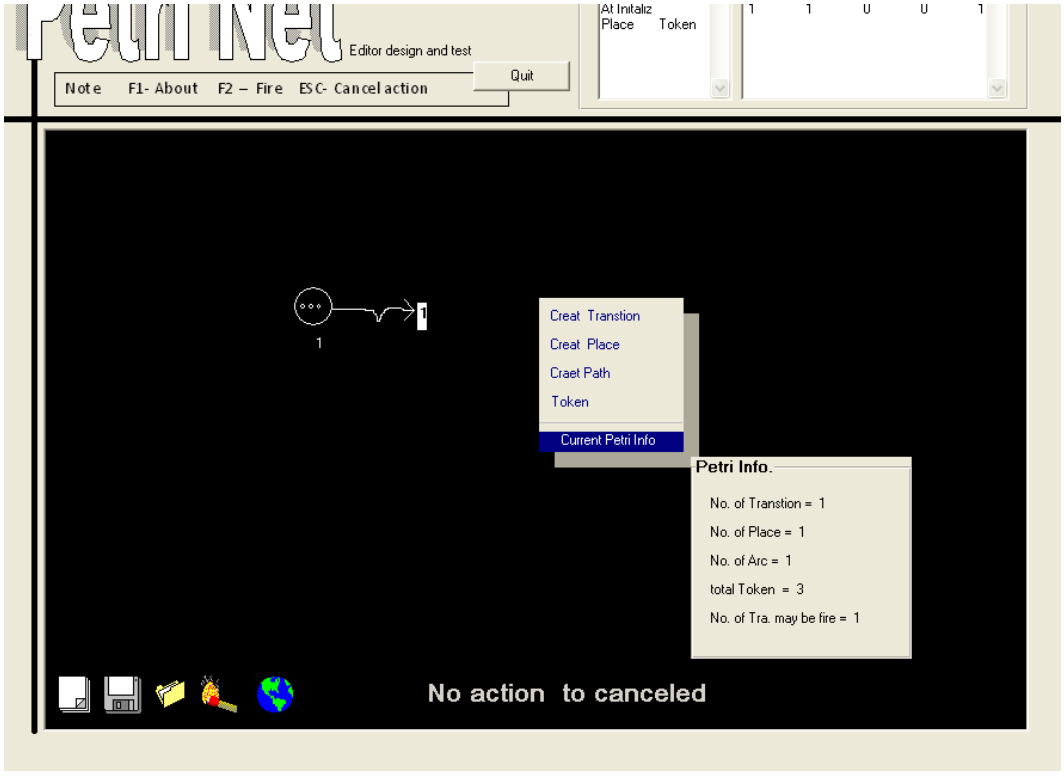
Screen 1



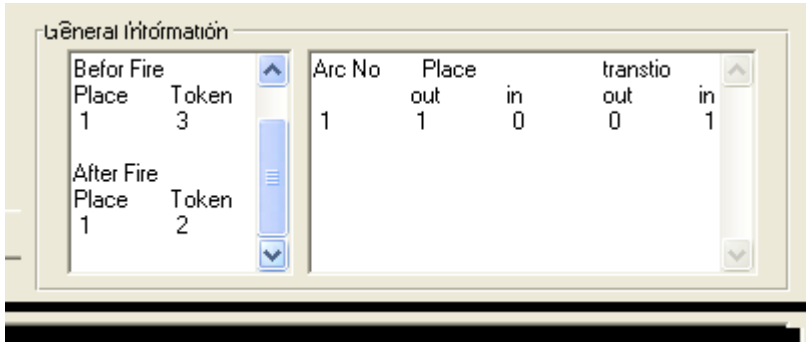
Screen 2

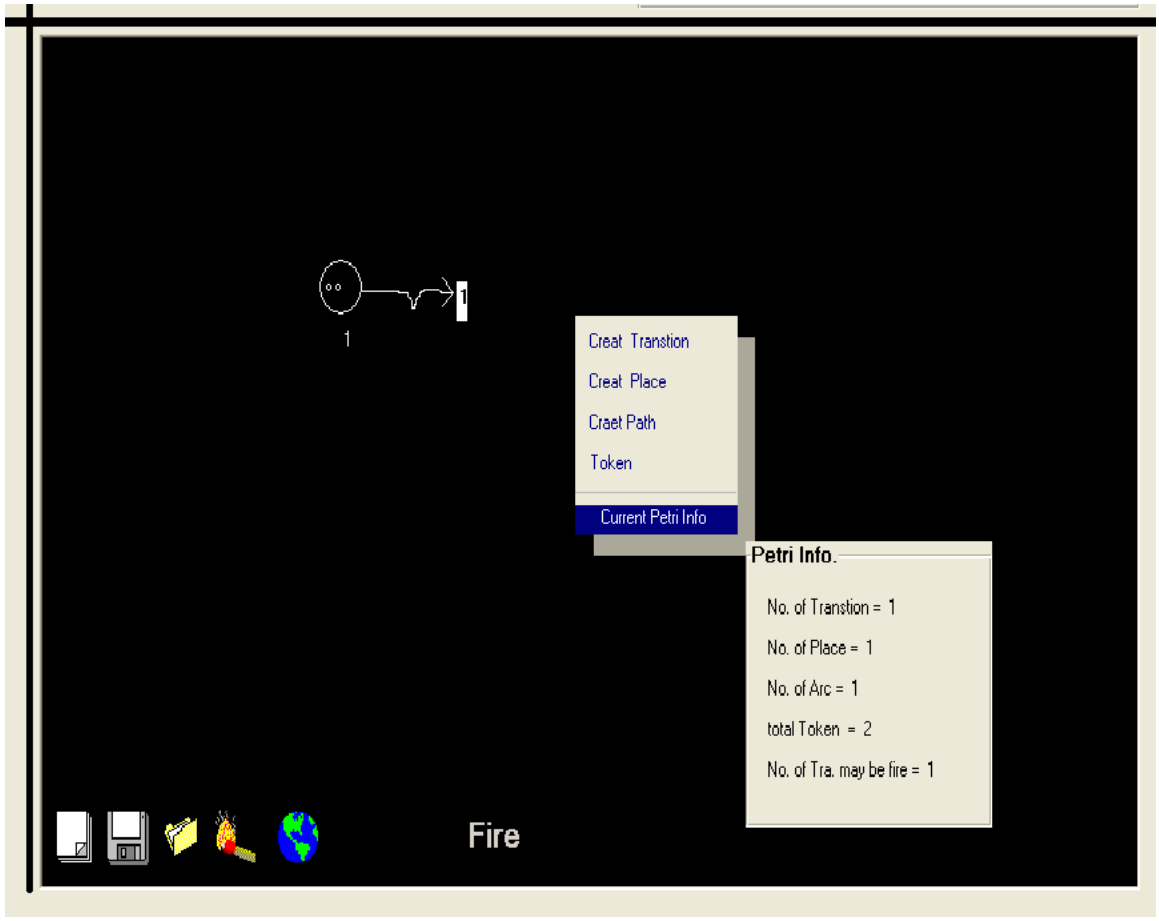


Screen 3



Screen 4


General Information
Befor Fire
Place Token
1 3
After Fire
Place Token
1 2
Arc No Place transtio
out in out in
1 1 0 0 1


1
1
Creat Transtion
Creat Place
Craet Path
Token
Current Petri Info
Petri Info.
No. of Transtion = 1
No. of Place = 1
No. of Arc = 1
total Token = 2
No. of Tra. may be fire = 1
Fire

Screen 5