Technical Paper

for

Feasibility Study and Design of Computer Assisted Object-oriented Petri-Net Tool

GUAT YEW TAN
School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM, Penang
email : gytan@cs.usm.my

1. Introduction

Petri nets and related graph models [1] have been used as graphical and descriptive mathematical modeling tools in a variety of applications, for example, performance modeling and evaluation, communication protocols, concurrent and parallel programs, etc. They are exceptionally well suited for designing and dynamic sequential dependencies.

In order to be usable in a convenient and effective way, the practical use of high level Petri nets [2] will only become significant with the aid of computer based tools. The main advantages in computer assisted Petri net modeling are possibilities to create better and faster results, to make interactive presentations of the results, to build certain more technical aspect of the Petri net theories into the computer system, to assist the user in structuring the process, etc. [4]

Owing to the vast advantages in the computerization of Petri net modeling and the fact that there is no one unique tool that can fulfill all the applications encountered, there were many computer based Petri net tools being implemented since 1978 [3] for various purposes and applications. However, the tools that were implemented during the early age contained only limited analysis functions and were text-oriented. The underlying net types used were high level Petri nets, stochastic Petri nets, timed Petri nets and other extended Petri net models[1]. In general, the tools were developed within small scale research activities and resulted in inconsistency in the sense that they are running on very different kinds of machinery and use different input/output format [4].

As raster graphics became faster and cheaper, Petri net tools designers had come to and idea of developing a user friendly tool by working directly with graphical representations of high level nets. The systems can be constructed, modified and analyzed via graphical editor. Consequently, the graphical and dynamic simulation has become an important part in the graphical implementation. Users can view clearly the "dynamic" instead of "static" token game. Each evolution of state can be seen explicitly and thus let the users have better control over the path in which each token should take. Basically, the tools in this stage emphasized on graphical representation.

Later, it became likely that the implementation of Petri net tools must be built on top of one of the existing high-quality graphical editors [4]. These tools will be much more compatible with each others, subsequently, more time and effort can be spent on the development in analysis functions and calculations in order to build certain more technical aspects into the tools. In general, the process of convergence was indicated instead of scattered pieces of tools developed un-orderly among small scale research activities.

However, as these tools becoming widely available, systems built by the aid of the tools tend to become more complex and usually end up with state explosion problem. Further, while the systems grow more complex and bigger, the generic Petri nets modeling becomes insufficient for all the applications encountered and thus there emerged *modified Petri nets* (termed *modi-nets* here), for example *SAN* in *UltraSAN*, *ER-net* in *Cabernet*, etc. [7], to fulfill various shortage of modeling presentation. As a result, users may be discouraged from using the tool in the sense that they need to have sufficient knowledge in understanding the mathematical theory of the modi-nets before they can learn to use the tool. In this case, one of the advantages of Petri net modeling which is to avoid the users from dealing with complicated mathematical formulae is no longer maintained. In addition, if users wish to switch from using one Petri net tool to another, they need to start learning all over again regarding another underlying net type found in the other tool which is totally different from the previous tool.

Lately, the introduction of object-oriented programming has drawn the attention of Petri net researchers to encode this idea into Petri net design. Recent work shows that there is a towering trend towards the research of hierarchical object-oriented Petri nets. State explosion problem which mentioned earlier can be solved now by structuring a large Petri net system in a hierarchical manner with different level of abstractions, or by dividing it into smaller modules/objects in which analysis can be carried out

more easily with simpler system. On top of that, the object-oriented concept also provides various invaluable schemes like information hiding, reusability, concurrency, etc. and thus has tremendously increased the modeling power of Petri nets, especially when producing a modi-net. Further, source code generation of the underlying system has also been taken into consideration to provide more user-friendly tools.

In the aspect of hardware and software used for implementation, there was a significant change when graphical editor was introduced in the tools. Another revolution in the programming environment occurred as modi-nets were used as underlying nets for the tools. More detail discussion will be made in the following section.

As there are overwhelmingly introduction of computerized Petri net tools, it will sometimes make the users confuse in choosing the most suitable tool. Some are too complex and complicated for beginners to learn, some do not provide GUIs (Graphical User Interfaces), some use *modi-nets* which are a slight deviation from the Petri net topic, etc. At this stage, the implementation of MASE is considered in order to include all the features of the existing tools and to design it as simple as possible that it can be used whether by beginners or advance users.

The organization of this paper is as follows: A brief introduction on Petri net concepts, notations and definitions of various components are described in section 2. Section 3 discusses different types of Petri nets available. The future direction and technologies are taken into consideration for developing future Petri net tools and is discussed in section 4. A Modeling Assisted Software Environment (MASE) tool is discussed in this section as well. The MASE tool is based on object-orientation and provides automatic generation of pseudo-code for the designed process.

2. Petri net

In this section, we discuss briefly notations, firing rules and properties' analysis by reachability tree approach of generic Petri nets. For detailed features, weaknesses, other analysis approaches and properties of Petri nets, please refer to [1, 9].

2.1. Notations and Definitions

Petri net (PN) is a 6-tuple, $PN=(P,T,I,O,M,W)$; where

P : a finite set of places represented by circles

T : a finite set of transitions represented by boxes or short straight lines

I : a finite set of input directed arcs connecting places to transitions, $P \times T$

O : a finite set of output directed arcs connecting transitions to places, $T \times P$

M : state of PN defined as tokens distributed within the places and represented by black dots

W : weight of the arcs

An example of the Petri net structure (Fig. 1(a)) is as follow,

$P = \{P_1, P_2\}$

$T = \{T_1, T_2\}$

$I = \{P_1 \cdot T_1, P_1 \cdot T_2, P_2 \cdot T_2\}$

$O = \{T_1 \cdot P_2, T_2 \cdot P_2\}$

$M_0 = \{4, 0\}$

$W(I) = \{2, 1, 1\}$

$W(O) = \{1, 1\}$

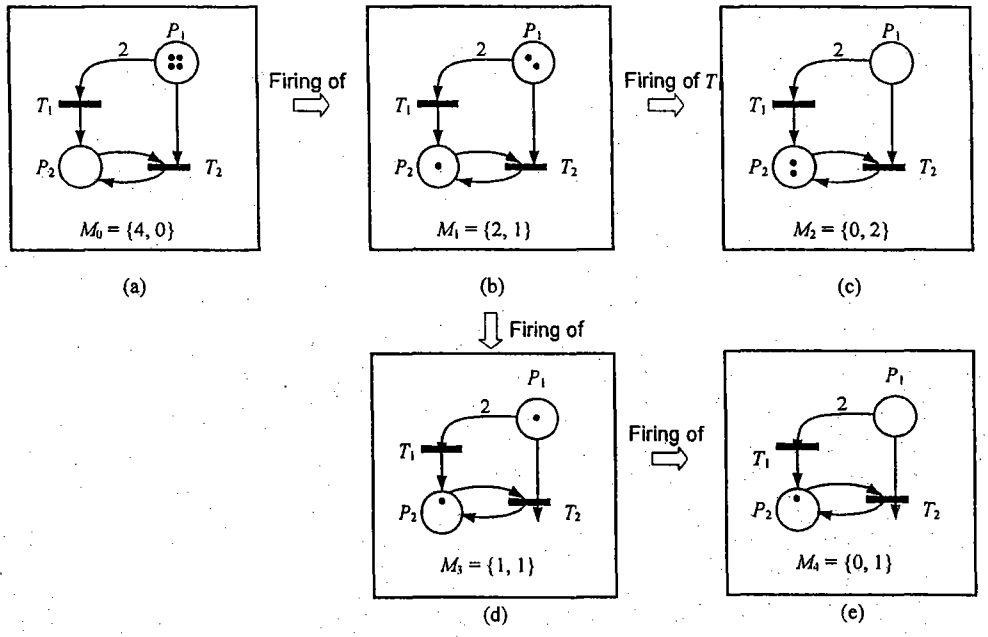


Figure 1. Example of a Petri net model (a) and its evolution of all possible firing (b) – (e).

2.2. Transition Firing

The dynamic behavior of Petri net is expressed by transition firing, the information flow in modeled systems. Petri net and its related graph models apply and introduce different rules in transition firing. For example, *place-transition net* says transition t is activated only if number of tokens for every preset p of t is greater than or equal to the weight of adjoining arc of p and t, and number of tokens in post-set of t will not exceed the capacity of t after the firing; *condition-event net* says that an event (transition) can occur if all of its pre-conditions are fulfilled and all of its post-conditions are unfulfilled; while *T-timed Petri net* says that transition t can only be activated if there is at lease one available token in each input places of t and that the associated time has elapsed [1, 10, 11]. We have seen in general, what all these transition firing of the models have in common is that the occurrence of transition firing will take place if there is at least one token in its input place and the inscribed condition is fulfilled.

As an example, in Fig. 1(a), from $M_0 = \{4, 0\}$, only T_1 is fireable as P_2 has no token for T_2 to become fireable. Marking $M_1 = \{2, 1\}$ is then obtained. In the case of M_1, both transitions are enabled. If T_1 is fired, $M_2 = \{0, 2\}$ is reached and meet deadlock. If T_2 is fired, $M_3 = \{1, 1\}$ is obtained and, one and the only one more marking can be obtaind if T_2 fires again before the system become stagnant is $M_4 = \{0, 1\}$ (Fig. 1 (b-e)).

2.3. Properties of Petri net

Of all the effort taken to design a system in a Petri net formalism, there is only one goal to achieve – analysis of Petri net properties [1, 2, 8]. Some of the important properties are as follows,

Reachability	- evolution of system's state from one to another.
Boundedness	- maximum tokens that can be contained in a place.
Safeness	- if the maximum token that can be found in a place is one.
Conservation	- the sum of tokens before and after a transition firing is equal in a Petri net.
Liveness	- a transition is defined as life if it can always fires without deadlock.
Persistent	- in any marking, an enabled transition can be disabled only from its own marking.

These properties can be analyzed in terms of mathematical notation and in terms of formalism. However, the beauty of Petri net is its concise structure and the information flow of modeled system carried by tokens' movements.

2.4. Analysis of Petri Net

The most common and much used analysis mechanism of Petri net properties is Reachability Tree Approach. It is a finite representation of reachability set which is the set of reachable marking from the initial marking. In our example, Fig. 2, which is a reachability tree of modeled system in Fig. 1(a) can be used to analyze properties like boundedness (=4), safeness (not safe), conservation (not conservative, since $\sum M_0 \neq \sum M_1$), liveness (deadlock occurs for markings M_2 and M_4), persistent (for M_1, either T_1 or T_2 can fire and meet deadlock soon; thus its not persistent), etc.

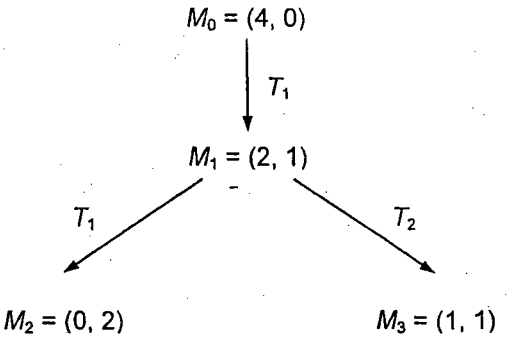


Figure 2. Reachability tree of Petri net model in Figure 1, showing all the states of the system.

3. Petri Net Tools Evolution

After investigating the tools in [3-7, 13], we conclude that the tools can be classified into four main categories, or four generations, which in turn depend on the facilities available for graphical representation, interfaces and windowing concepts.

3.1. First Generation

Consists of tools without graphical editor and the system is constructed via textual editor. In this case, incidence matrix of the system will be edited through matrix editor. If the systems seem to be a high level

or extended Petri net where the associated net attributes like timing, predicate, etc. cannot be described by incidence matrix, a description editor is created along with the tool for this purpose.

For some tools which are not constructed by incidence matrix, they have prepared a set of net language whereby users need to learn the net language which carries the functions similar to programming language. Underlying place-transition net and associated attributes mentioned earlier are described in terms of the net language. These type of tools act as Petri net compilers.

Languages used for implementation are mostly structural programming languages that can be run on IBM personal computers like Pascal and FORTRAN. In the aspect of operating system, it includes MS-DOS, etc.

Only limited analysis functions, for example, reachability tree construction and analysis, net reduction and S- / T-invariant construction, found in this generation. There is no on-screen dynamic simulation since graphical representation is not supported. Source code generation is still an idea far beyond this generation.

Output of the analysis, if available, will be in text from and consist of statistical data. Some tools provide line printer graphic output, for example, *PES* used line printer, etc. [3, 4], to compensate the absence of electronic graphical representation. In general, first generation covers the early versions of Petri net tools developed.

3.2. Second Generation

In this generation, tools have been improved in the aspect of graphical representation and the programming environments. In general, a substantial change from the first generation is the model construction. With the rapid growth of graphical user interface (GUI), each tool provides graphical editing and is becoming more and more user friendly. Mouse has taken over the role of keyboard as main input media wherever possible. Tools which used structural programming language as implementing language in the first generation, Pascal for example, developed new versions by using the graphics library of respective languages. On the other hand, some tools which were not extended in the above manner will be implemented or re-implemented by using C language and X window library in the UNIX environment. In fact, it can be seen that majority of them applied X window graphics library, Xlib, for the purpose of maximum portability

among various machines, due to the powerful GUI provided by X-window that is incomparable with other graphics systems.

In view of the implementation of GUI into the tool, state evolution can now be seen clearly on the screen immediately instead of scratches by using paper and pen previously which was tedious and time consuming. There are forward, backward and step evolutions, in order for the users to view the next or the previous possible states step-by-step with only button-press on that mouse. Further, in the aspect of decision-making during simulation, conflict for example, the tools provided functions where users can make the decision by him/herself or allow the tools to make the decision randomly. In other words, there is an improvement in performing the result to the first generation by using the interactive presentation.

3.3. Third Generation

There was a significant decrease in the number of tools from first and second generation to this generation. However, the decrement cannot be seen as the reduced interest in this area. Contrary, it shows that a process of convergence is taking place. Some small scale research groups have teamed up to produce a better tool. For example, DAIMI which was developed at Aarhus University under the supervision of Kurt Jensen [8] for the purpose of being of assistance in the teaching of Petri nets, is classified in the first generation tool with all the features mentioned above, has been combined with DESIGN, which supports graphical editors and has better GUI, developed as a commercial tool at Meta Software Corporation under the supervision of Rober Shapiro. Thus combination has resulted the emerge of Design/CPN which have more powerful analysis functions, better outlook of graphical representation and better control in performing the dynamic simulation. It is more user friendly and commercially available.

The desire to develop a multi-application Petri net modeling tool was stimulated in this generation with the availability of faster and cheaper micro-processor at the contemporary technology. Thus, besides fundamental analysis functions found in the first generation, more functions, for example, performance and reliability analysis, numerical analysis, structural and graph properties analysis, etc., have been added to cope with the need for upgrading the tool to a multi-application tool. However, as the modeled system becomes more complex, the chances for its Petri net prototype to face state explosion problem is higher owing to its unmanageable size.

3.4. Fourth Generation

Tool evolved in this generation have substantial changes in the modeling mechanism and the concept of result-generation, while maintaining all of the features described in the third generation.

The limitation of high level and extended Petri nets modeling in various applications [1] become a bottle-neck if the tools based on the existing net types are wished to be widely used. In order to be usable in different areas, new net types which is called modi-nets by the author have been introduced along with the tools to overcome those limitations. However, as mentioned earlier that the numerous modi-nets which use extensive mathematical theory might discourage the users from using them and make the situation tougher while users switching from using one tool to another.

As the object-oriented programming becoming more popular, by the diffusion of hierarchical object-oriented concept into Petri net theory (regardless of whether the Petri net is a classical, extended Petri net or other modi-net), Petri net can optimize its modeling power. In this case, large systems can be represented in different level of abstractions, whereby encapsulation, reusability and concurrency schemes can be easily applied on it. State explosion problem which was a critical drawback in second generation can be solved in this way. Another solution is to partition the large system into smaller modules/objects so that analysis can be carried out on simpler objects. Basically, the modi-nets in hierarchical object-oriented style have gained maximum attention of researchers from all over the world.

It is concluded that C++ is an ideal programming language to be used to implement the hierarchical object-oriented Petri net tool due to the object-oriented nature of the programming language itself. Thus, it is very easy for the third-generation-tool to be upgraded to fourth generation because most of them are implemented in C. Still, the hardware environment is UNIX with a slight changes in the graphics programming, i.e. from Xlib to Motif, though Motif and Xlib will be interpreted to the same source code in C.

While nearly all of the necessary analysis functions had been implemented in the previous generations, it seems nothing left to be accomplished at this stage. Thus, more attention has been put on the implementation of description tools which is used to interpret the inscribed attributes of Petri nets discussed in first generation. For example, if the tool is based on Timed Petri net, then besides timing conceived in

the transition/place, a supplementary system time is added as a clock of the system by the tool. Another example is the vareditor tool to edit performability variable in UltraSAN.

Besides the fancy interactive representation of the results, the idea of source code generation of the execution of the modeled system comes to the mind of Petri net designers at this stage. Hence, the user-friendliness of the fourth generation tools has been moved to one-step ahead those in the previous generations. By this generation, users need to know little about the programming languages in order to activate a system from its early stage of design to its end stage of execution.

4. Modeling Assisted Software Environment (MASE) Tool

MASE [12] is developed in view of the lack of GUI, weak expressive power in certain circumstances (zero testing for instance) due to the limited components provided by generic Petri net model, source-code generation for modeled system, etc.

4.1. Hierarchical-Object-Oriented-Petri-net

Basically, MASE adapts hierarchical-object-oriented-Petri-net as its underlying net. In terms of hierarchical object-orientation (HOO), MASE deals with the decomposition of larger objects into smaller components objects by providing subnet-creation feature. At its top level, any complete system is represented by an independent object. It also implies that any object on one layer needs to know nothing about its senior layer and as a result, information hiding resource is conformed. The reusability resource is conformed in the sense that functions that can be used in one layer, can be used in any other layers as well. The main advantage of HOO scheme is that the coupling between objects can be reduced in order to solve the primary problem encountered while building a large system – state explosion problem.

4.2. Graphical User Interface (GUI)

In the aspect of GUI, MASE is a mouse-driven software package; creation, modification and simulation of the modeled system are governed by mouse [12]. The unique GUI feature of MASE is the implementation of Execution Panel (Fig. 3). By monitoring the panel, the user can change the speed of firing, view the

frame-by-frame or step-by-step firing, halt the simulation as desired, rewind the system to its initial state, etc.

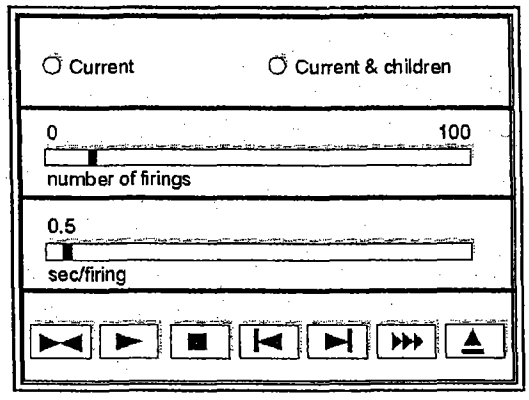
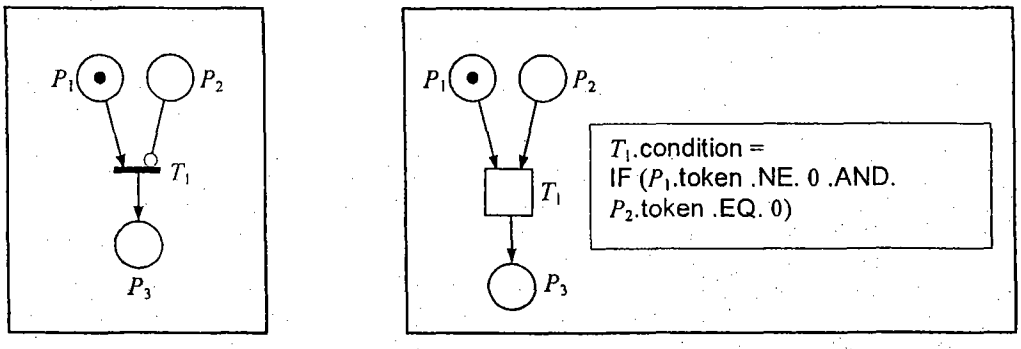


Figure 3. Execution Panel of MASE

4.3. MASE Language

A set of descriptive language named MASE Language (MASEL) is also implemented along with the package to strengthen the expressive power of Petri net model. In the complementary of MASEL, MASE hope to model as many systems which are unable to be model by generic Petri net and which need introduction of extra components as mentioned in modi-nets of section 1. In this way, various problems arose by modi-nets can be eliminated. For instance, in order to do zero-testing, in a common practice, inhibitor arc (which is a small circle attached to the end of the arc near related transition) will be introduced (Figure. 4(a)). However, in MASE (Fig. 4(b)), a novice user of Petri net can understand the structure without further reference.



(a) Symbol of inhibitor arc (b) In MASE, without the symbol of inhibitor arc

Figure 4. (a) Inhibitor arc. Transition T_1 can only fire if place P_1 contains token and P_2 does not contain token. (b) Some functionality of Petri net represented in MASE.

4.4. Code Generation

MASE provides a code-generator which generates pseudo-code of the modeled system so that the user can easily map the pseudo-code to desired programming source code. This feature will reduce the effort to recover human errors especially semantic errors and run-time errors, while prototyping the modeled system into executable program. As an example on how the modeled system in MASE can be mapped to pseudo-code, the system in Figure 5 is generated as follows,

```
INT y1, y2, y3;
BEGIN
    READ(y1); READ(y2); y3 = 1;
    WHILE (y1 > 0) BEGIN
        IF (ODD(y1)) BEGIN
            y3 = y3 *  y2; y1 = y1 – 1;
        ENDIF
        y2 = y2 * y2;
        y1 = y1 – 2;
    ENDWHILE
    OUTPUT (y3);
END
```

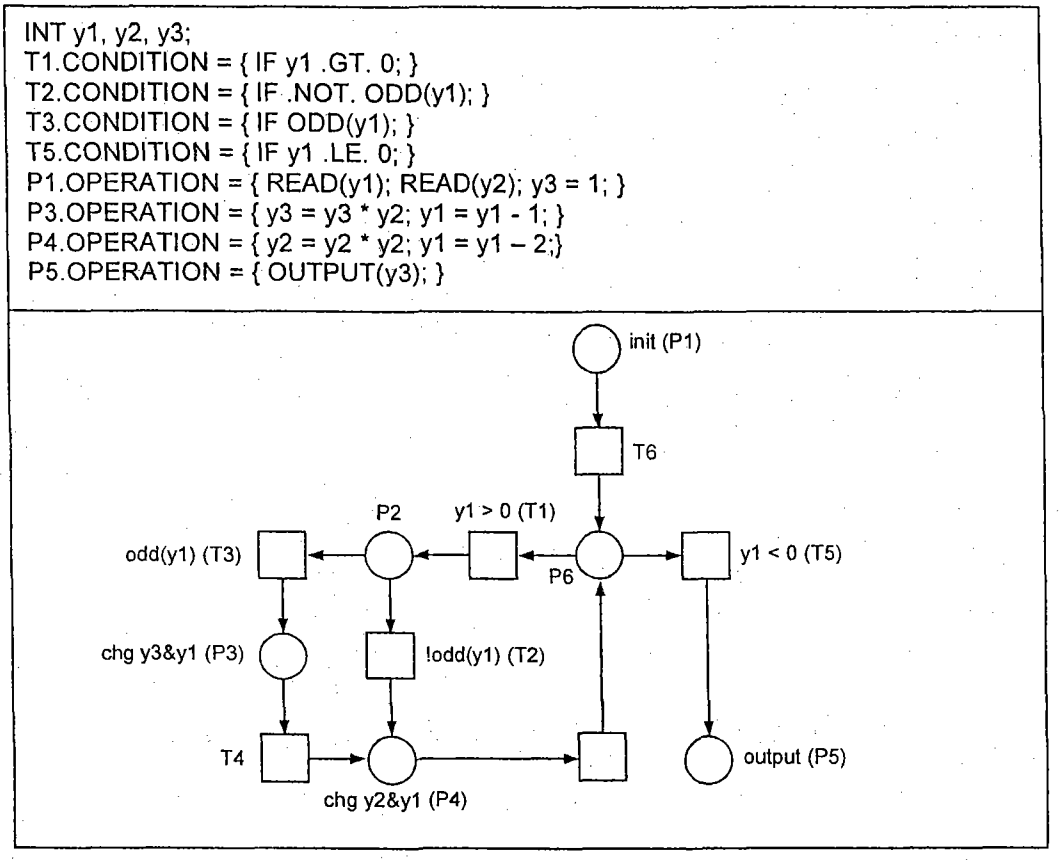


Figure 5. A sequential programming flow-chart which is modeled by Petri net.

In general, MASE is a fourth generation tool as it covers the features of fourth generation, i.e. it is hierarchical, object-oriented, written in C++ language, supports graphical user-interface which implemented in X/motif running on UNIX platform, includes a set of descriptive language to express the condition of the modeled system, provides course code generation, last and not least, it is easily extensible to various net models and possible to enhance the analysis functions.

5. Conclusion

In this paper, we have discussed various Petri net tools that can be classified in four generations. From first to second generation, we see an enormous improvement from text editing to graphical editing, as well as result presentation and analysis functions. In third generation, the collaborations of small scale research groups were a main factor for the tools to become more powerful and user friendly by the integration of

various tools from first and second generations. From third to fourth generation, modi-nets, object-oriented approach and source code generation had caused the revolution in designing the tools. Due to a growing interest in such tools and an increasing industrial usage, Petri net exhibition has become a standard part of the program of the yearly European Workshop on Applications and Theory of Petri nets.

We also discussed a modeling assisted software environment tool (MASE), a fourth generation user-friendly software package for the hierarchical object-oriented Petri net modeling. We emphasized on the user-friendliness and the omission in using modi-nets in this package.

Recently, the rapid growth in the modern technology in which microprocessors, memory chips and raster graphics become more powerful and cheaper, we hope that a more user friendly tool and a tool that can cover all aspects of the applications, fast result generated like reachability tree construction, unlimited construction of Petri net components, more favorable and impressive graphical representations will come to existence.

Further, the explosion of interest in artificial intelligent and multi-media may lead to the implementation of a science-fiction tool to reality, in the sense that instead of inefficient input media like mouse and keyboard, the users can command the tool verbally what application they are going to model and the computer will generate the diagram by itself. In addition, the intelligent tool may also rate the degree of feasibility of the model system and suggest another more feasible way.

References

1. G.S.Hura, Chapter 9: Use of Petri Nets For System Reliability Evaluation, *K.B.Misra(ed.): New Trends In System Reliability Evaluation. 1993 Elsevier Science Publishers B.V.*
2. K. Jensen, High-level Petri nets, *A. Pagnoni and G.Rozenberg (ed.): Application and Theory of Petri nets. Informatik-Fachberichte, vol 66. Springer Berlin 1983, 166-180.*
3. Frits Feldbrugge, Petri net tools, *G.Rozenberg (ed.): Advances in Petri Nets 1985. LNCS 222, Springer-Verlag 86, page 203-223.*
4. K. Jensen, Computer Tools For Construciton, Modification And Analysis Of Petri Nets, *LNCS, APN, 1986, page 1-19.*
5. Frits Feldbrugge, Petri net tool overview 1989, *G.Rozenberg (ed.): APN, LNCS, 1989, page 151-178.*

6. F.Feldbrugge and K.Jensen, Computer Tools for High-level Petri Nets, *G.Rozenberg (ed.): High Lovel Petri Nets – Theory and Application (1991).*

7. Petri Nets Tools, *posted by Groupe ECORP, Centre de Recherche Informatique de Montreal (CRIM), in the Petri nets mailing list.*

8. K.Jensen, The design of a program package for an introductory Petri Net course, *G.Rozenberg (ed.): APN, LNCS, 1984, page 259-266.*

9. T.Murata, Petri Nets: Properties, Analysis and Applications, *Proceedings of the IEEE, Vol. 77, No. 4, April 1989.*

10. Wolfgang Reisig, A primer in Petri net design, *Springer-Verlag Berlin Heidelberg 1992.*

11. J.L.Peterson, Petri Net Theory and Modelling of Systems, *Prentice-Hall Inc., Englewood Cliffs, N.J. 07632 1981.*

12. G.Y.Tan and G.S.Hura, A Petri-Net-Based Modeling Assisted Software Environment (MASE) Tool, *18th Int. Computer Software and Applications Conference (COMPSAC), Taipei, November 1994.*

13. Harald Storrle, An Evaluation of High-End Tools for Petri-Nets, *Ludwig-Maximilians-Universitat Munchen, Institut fur Informatik, Bericht 9802, Jun 1998.*