ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 11, November 2017

Automatic Conversion of Formal Specification into Code

[1] Manju Pandey
National Institute of Technology,Raipur

***Abstract*— This paper proposes an idea. The idea is to first express a requirement specification using a formal method. The second step is to work out an implementation of the specification in the form of running code in a standard contemporary programming language. The other step is to identify frequently occurring patterns in the formal specifications. And then direct the conversion from the formal specification to the running code. Intermediate steps shall include the development of a computer-based tool for expressing the requirement specifications. we require another tool that shall develop the conversion of the specifications expressed this way into running code. The steps and process are explained with a hypothetical example.**

***Keywords*- Requirement specification, conversion, formal specification, pertinent.**

I. INTRODUCTION

Petri nets is mathematical tool for modeling disseminated system and for specific notions of concurrency, non determinism, communication and synchronization.[1] Petri net facilitate simple model process organization, asynchronous procedures, contemporaneous operation, and variance or source allocation. Petri Nets have been used for simultaneous and comparable systems modeling and investigation, communication protocols, performance assessment and error-tolerant systems. In order to understand the performance and dependability problem in a given process or a system it is important to introduce the timing concept into the model. So in order to compute this many ways are given but simple way is to combine a firing delay with each transition[2][3] This delay relates the time that transition has to allow before it can fire. If this delay is arbitrary distribution function or random distribution function, then the resulting net class is stochastic Petri net. Many other transitions can be differentiated depending on the associated delay, exponential, deterministic transition.

Petri net is a multilateral directed graphs colonized by three different entity. These entities are transition, places, and directed arcs. Directed arcs are for connecting places to the transition or connecting transition to the places. Further Petri net can also be identified by connecting transition along with an input place and an output place. This kind of representation is very simple and can be use to model various system that are not so comple.[4]

As we can explain this with example, a transition and its input place and output place can be used model data processing activities, at least one input and at least one output is compulsory to be there in this process .Other than this the active behavior of the Petri net modeled in terms of state change can be modeled. Every place in this model may hold either none or positive number of token in it. This way of behavior is called as the dynamic behavior of pertinent.
Tokens are a primeval idea for Petri nets in count to places and transitions[1]

A Petri Net is a triple consisting of three finite sets:
PN=(P,T,F)

Where:
P is the set of places
T is the set of transitions,
F is the set of arcs

Places and Transitions cannot be the same node. Arcs connect places and transitions, and vise versa. Marking of a PN=(P,T,F) denoted by M is a mapping which assigns a positive integer number of tokens to each place of the net. A marking M (distribution of tokens over places) is often referred as the state of a given Petri Net.[1]

Arcs in the pertinent are used to explain the communication happening between the active and passive elements. Since it has only feasibility to connect various nodes (transition/place) of different type that is why it is called bipartite graphs. An arc has attached a natural number written near the corresponding arc, the arc weight, which specifies that a transition is enabled only when the input places has at least as many tokens as given by the arc weight. In this method arc facilitates transition fires so an equal number of tokens given by the arc weight in the input place is cracked[5][6][7]

Petri nets have been useful in understanding and developing different processes from the various protocols, system development, business process processes, resource development, automation process, production process and many more[8][9]
In the last two decades the classical Petri net has been extended with color, time and hierarchy [10].

II PROPOSED METHOD

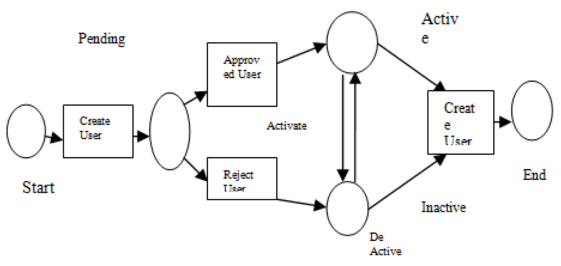
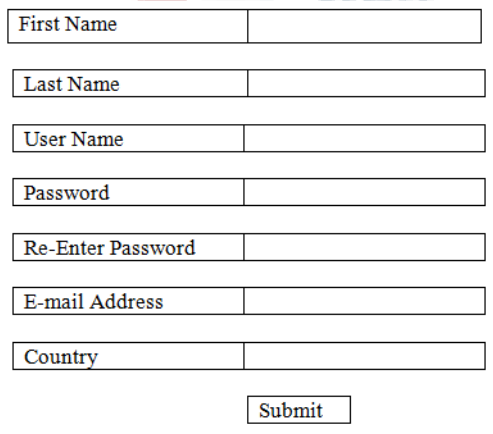


Fig. 1 Petri Net Model

Here Figure 1 shows a Petri net model of a simple user creation process, a pattern that occurs frequently in various contexts in a user's interaction with information systems
The process starts with creating the user and based on the process the user can be approved or it can be rejected.

The top-Down Design:

For automatic generation of code, the Petri net will have to be traversed from left to right. The create user screen will look something like the following



First Name	
Last Name	
User Name	
Password	
Re-Enter Password	
E-mail Address	
Country	

Submit

Fig. 2. Create New User

Once a new user enters his details, the request shall be either be approved or rejected depending on some pre determined criteria, like for example the password and the re-enter password fields being the same. The system shall prompt the user as to whether his request has been approved for rejected and ask him to the needful.
The basic idea is that this pattern is a frequently occurring pattern in different contexts. This can be utilized to generate a generic function that can generate Java code corresponding to the GUI from the corresponding node of the Petri net. This translation will follow the sequence established in the flowchart in the following section. The flow diagram is for converting Petri net specification into the code

Flow Diagram for Conversion of Petrinet Specification to Code

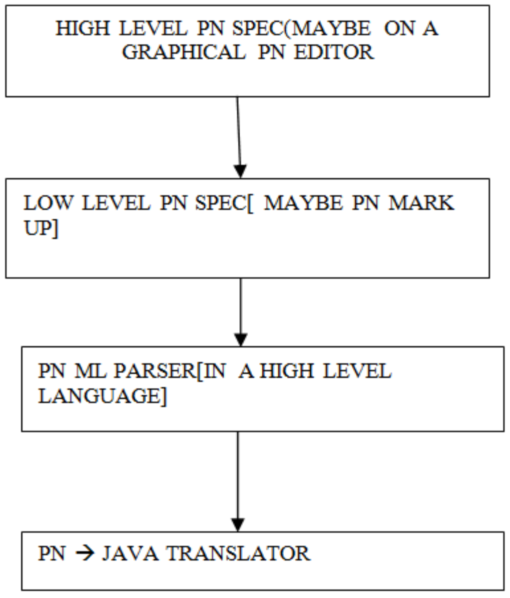

HIGH LEVEL PN SPEC(MAYBE ON A GRAPHICAL PN EDITOR
LOW LEVEL PN SPEC[MAYBE PN MARK UP]
PN ML PARSER[IN A HIGH LEVEL LANGUAGE]
PN → JAVA TRANSLATOR

Fig. 3. Conversion of Petri net into Code

In this flow diagram top down approach is considered. At the high level Petri net is graphical Petri net editor is designed that is considered to be the first stage in any such model development. Moving to the next level we require any low level Petri Net specification at this maybe the Petri net mark up used in the second level of designing. The next state in this flow is the Pertinent ML Parser that is incase we are going to deal with the High level language.
Finally we require the Petri Net we need the java translator in order to complete the flow.

The sequence of t he has to be maintained, if there is slight change in the deviation then we cannot think for rearranging them. The proposed method is applicable in any kind of domain and where we need to capture the flow process. The translator is simple mechanism where the modeling of multi threading concept is utilized as in Java.

III CONCLUSION

It will be an interesting exercise to actually build this translator. Concurrency can be modeled using multi-threading features of Java. It is also possible to support the confusion concept in Petri nets by using random choices instead of strictly deterministic ones. Advanced functionality to support database integration across modules and build entire information systems from their corresponding Petri net specifications can be included at a later stage.

REFERNCES

[1] Peterson, James L. (1977) Petri nets ACM computing surveys

[2] Peterson, james L. (1981), petri net theory and the modeling of systems, prentice Hall

[3] Matthias Jungel,Ekkart Kindler, and Michael Weber, “The petri net markup language September 2002

[4] I. Low, Y. Yang, and H. Lin, “Validation of Petri net apoptosis models using p-invariant analysis,” in Control and Automation, 2009. ICCA 2009. IEEE International Conference on, 2009, pp. 416 –421.

[5] A. Yakovlev, L. Gomes, and L. Lavagno, Eds., Hardware Design and Petri Nets. Kluwer Academic Publishers, 2000, ”ISBN0-7923-7791-5, 331 pgs”

[6] C.Giraud and R.Valk, Petri Nets for Systems Engineering. A Guide to Modeling, Verification, and Applications. Berlin - Heidelberg - New York: Springer, 2003

[7] Bernd Grahlmann and Eike Best. PEP — more than a Petri net tool. In Tiziana Margaria and Bernhard Steffen, editors, Tools and Algorithms for the Construction and Analysis of Systems, volume 1055 of Lecture Notes in Computer Science, pages 397–401. Springer Verlag, 1996

[8] C. Baral. Knowledge representation, reasoning and declarative problem solving. Cambridge University Press, 2003. ISBN 0521818028.

[9] C. Baral. Knowledge representation, reasoning and declarative problem solving. Cambridge UniversityPress, 2003. ISBN 0521818028.

[10] Willem Visser, Klaus Havelund, Guillaume P. Brat, and Seungjoon Park. Model checking programs. In ASE, pages 3–12, 2000.