Pakistan Journal of Applied Sciences 2(1): 77-79, 2002

Metrics Evaluation for Industrial OO Petri Nets Models

Abdulruda A. Alfize
Faculty of Computer Science & Computer Engineering
Ajman University of Science & Technology, Abu Dhabi Campus
Abu Dhabi, P.O. Box 5102, U. A. E

Abstract: Petri nets are used for a graphical representation to model industrial controllers. The model parts are then mapped to OO classes whose software performances are evaluated using encapsulation, inheritance, coupling, and polymorphism metrics.

Keywords: Metrics Evaluation, Petri Nets, Object Oriented Programming

Introduction

Modeling industrial processes with Petri Net Object Oriented Data Structure (PNOODS) (Alfize, 2000), would require a method of performance evaluation. For such purpose, metrics evaluation is proposed to be used (Basili *et al.*, 1996; Hassison *et al.*, 1998). In this type of metrics evaluation, several parametrics factors are considered, that are as follow. Method Hiding Factor (MHF) and Attribute Hiding Factor (AHF) metrics are used jointly as measures of encapsulation. Method Inheritance Factor (MIF) and Attribute Inheritance Factor (AIF) metrics are used jointly as measures of inheritance. Polymorphism Factor (PF) metric is used as a measure of polymorphism potential, while Coupling Factor (CF) is used to measure coupling between classes.

To graphically model industrial processes, Petri net is often selected. Petri net structure (Zurawski and Meng Chu Zhou, 1994) is four tuples (P, T, I, O) where P = $\{p_1, p_2, ...p_n\}$ is a finite set of places (events) while T = $\{t_1, t_2, ... t_m\}$ is a finite set of transitions (actions). The set of places and the set of transitions are disjoint, $p \cap t = \emptyset$. I is an input function that defines directed arcs from places to transitions. O is an output function that defines directed arcs from transitions to places. Petri net is marked, i.e., contains tokens. Tokens (represented graphically by dots) reside in places, travel along arcs, and their flow through the net is regulated by transitions. A marked Petri net is defined by the quintuple PN = (P, T, I, O, m). The marking m is an n-dimensional vector whose ith component $m(p_i)$ represents token in the *ith* place p_i. The initial marking is denoted by m_0. The computation of all possible future markings starting from the initial marking can be drawn in the form of reachability tree. The marking m_r is said to be reachable form m_0 if there exists a firing sequence that will yield m_r. The set of all possible markings reachable from m_0 is called the reachability set, and denoted by $R(m_0)$.

Since their first arrival, Petri nets have been used for modeling concurrent computations. They captured the precedence relations and structural interactions of stochastic, concurrent synchronization, and asynchronous events. Their models are very compact representation of a Markov chain (Desrochers and Al-Jaar, 1995). Hence, they were widely applied in the design and simulation of operating systems, flexible (industrial - automated) manufacturing hardware communi- cation protocols, real time fault tolerant safety critical systems, and the multimedia (Guan *et al.*, 1998).

Object-Oriented Petri net can be considered as a special kind of high level Petri net. In this class of nets, tokens are considered as instances or tuples of instances (entities) of object classes that are defined as lists of attributes. The marking therefore, is not tokens but entities (Sibertin-Blanc, 1985).

Industrial Control Modeling Using Petri Net: In order to use PN for real time sequence control, timing and input/output sensory information have to be integrated into them. Therefore, association of real time with the transitions becomes a necessity. By modeling the I/O signals as places attributed tokens, the graphical complexity would be reduced. Still, sequence controllers are not necessarily strongly connected (existence of a directed path from every node to every other node in the net) when compared to Petri nets. To describe the use of Petri net modeling for industrial control, an illustrative example of Programmble Logic Controller (PLC) setup (Alfize, 2000) can be considered toward applying the targeted metrics analysis. Fig. 1 shows the case study example. The example is an automated ventilation system that is used to feed fresh air into a room and to remove used air from it. The room contains an exhaust air ventilator and a fresh air ventilator with monitors to control both ventilators. The fresh air ventilator can not be switched on unless the flow monitor indicates that the exhaust air ventilator is functioning properly. After a short time has elapsed and monitors detect no airflow while exhaust air ventilator is on, the system is switched off and a fault is signaled.

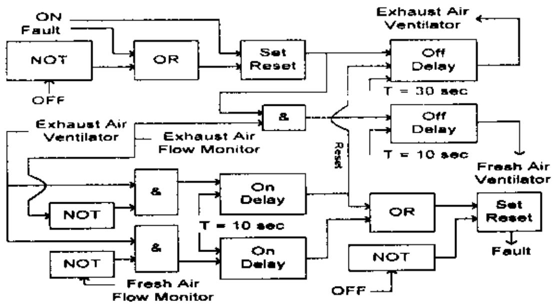

Fig. 1: Room Ventilation PLC Setup

ON (OFF) delay boxe in Fig. 1, indicate the delay of the assigned time of seconds that are needed to elapse before the ON (OFF) operation "output goes to 1 (0)" takes place. OFF (ON) operation take 0 seconds for ON (OFF) delay box. Reset line to exhaust air ventilator OFF delay has the priority to switch the ventilator off with zero delay. To draw the related Petri net, there is a need to realize two parameters that usually come with PLC's models. The two parameters are ON/OFF delays, and the monitor states. Fig. 2 represents the symbolic drawings within the net. Fig. 2, can be interpreted by looking at place p_1 where ON can not act on transition t_1 in case of OFF or Fault conditions is present (P_2 or P_8 has a token), due to the associated inhibitor arcs from t_2 and t_{23}. Inhibitor arcs are those which end with circles. The inhibitor arc connects an input to a transition that is inhibited when such arc is enabled (Guan *et al.*, 1998).

Upon pressing ON condition, Exhaust ventilator (EV)goes ON, i.e., P_3 gets a token, and continues to do so unless OFF is triggered or a Fault is happened. After the ON condition has been initiated, Fault and OFF signals should continue present for at least 30 sec, see transitions t_{18} and t_8, to enable the prioritized inhibitor arcs initiated from t_{10} and t_9. Triggering t_{10} or t_9 will cause the token to return to P_1, i.e., system ready to go ON provided no Fault or OFF signal presnet. Similar discussion goes for fresh air ventilator.

FV, where the place P_4 gets a token as long as there is no OFF or Fault signal and exhaust air monitor EM detects a flow (i.e., P_5 has a token), after the ON condition initiated. After then, if either OFF or Fault has been intiated, then there should at least 10 sec of continuity for such act to deactivate the ventilator FV. When exhaust air monitor EM and fresh air monitor FM go OFF, monitors states p_5 and p_6 lose their marking (allowing the firing of inverter transitions t_5 and t_6).

Note that timed transitions in the net are t_8, t_{13}, t_{14}, t_{18}, t_{19}, and t_{20}. For demonstration reason P_7 describes the complement state of P_5, i.e., P_7 contains a token only when P_5 has lost it. The net in Fig. 2 has the set of marking vectors $\{m_0, m_1, m_2, m_3, m_4, m_5\}$. Presence of *1* indicates a token in a place p_i, while *0* indicates absence of the token in place p_i.

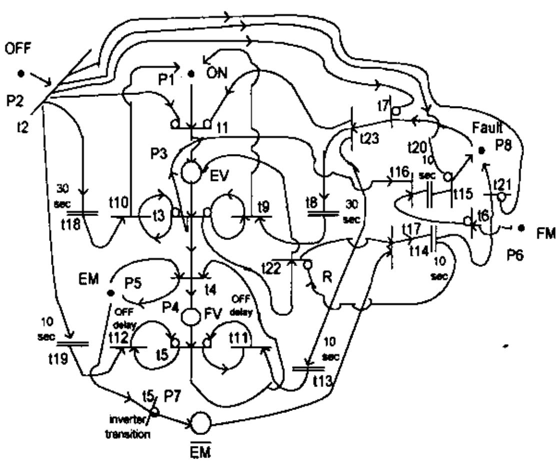

Fig. 2: Petri net model for the PLC setup

The marking vectors in table 1 have the reachability sets:

$R_1(m_0) = \{m_0, m_1, m_2, m_3\}$,
$R_2(m_0) = \{m_0, m_1, m_2, m_4\}$,
$R_3(m_0) = \{m_0, m_1, m_2, m_5\}$

Table 1: The Marketing Vectors

Marketing	m_0	m_1	m_2	m_3	m_4	m_5
P_1 ON	0	1	0	0	0	0
P_2 Off	1	0	0	0	0	0
P_3 EV	o	0	1	0	0	0
P_4 FV	0	0	1	0	0	0
P_5 EM	1	1	1	0	1	0
P_6 FM	1	1	1	1	0	0
P_7 EMoff	0	0	0	1	0	1
P_8 Fault	0	0	0	1	1	1
System condition	Idle	Ready On	Running	Fault; EM Off	Fault; FM Off	Fault; EM & FM off

Using these sets of reachability, the reachability tree can easily be obtained.

Coding the net model using Java code, eight classes were needed. The main class: Petri_net_mode*l*, and two threaded classes: Timed_delay_thirty and Time_delay_ten, plus five other classes that are: Single_transition, Place, Doubled_transition, Trippled_ transition, and Inverter_transition.

Metrics For Object Oriented Design:

Method Hiding Factor (MHF) and Attribute Hiding Factor (AHF) metrics are a measures for encapsulation. They are defined by (Brito Abreu and Melo, 1996):

$$\frac{\sum_{i=1}^{TC}\sum_{m=1}^{M_d(C_i)}(1-V(M_{mi}))}{\sum_{i=1}^{TC}M_d(C_i)}$$

where $M_d(C_i)$ is the number of methods (or attributes) declared in a class, and:

$$V(M_{mi}) = \frac{\sum_{j=1}^{TC} is\_visible(M_{mi}, C_j)}{TC-1}$$

TC is the total number of classes.

Inheritance in the software is measured

$$is\_visible(M_{mi}, C_j) = \begin{cases} 1..iff..j \neq i \wedge C_j. \\ .calls..M_{mi} \\ 0......otherwise \end{cases}$$

by Method Inheritance Factor (MIF) and Attribute Inhirtance Factor (AIF) metrics that can be defined as follows (Abreu and Melo, 1996):

$$\frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_a(C_i)}$$

where

$M_a(C_i) = M_d(C_i) + M_i(C_i)$

$M_d(C_i)$ = Number of methods/ attributes declared in a class,

$M_a(C_i)$ = Number of methods/ attributes that can be invoked in association with C_i,

$M_i(C_i)$ = Number of methods/ attributes inherited (and not overridden) in C_i.

Coupling Factor (CF) metric measures coupling between classes, excluding coupling due to inheritance. CF has been defined formally (Brito Abreu and Melo, 1996) as:

$$\frac{\sum_{i=1}^{TC}\left[\sum_{j=1}^{TC} is\_client(C_i, C_j)\right]}{TC^2 - TC}$$

where

$$is\_client(C_c, C_s) = \begin{cases} 1..iff..C_c \Rightarrow C_s \wedge C_c \\ \neq C_s \\ 0......otherwise \end{cases}$$

C_c □ C_s represent the relationship between a client class, C_c and a supplier class, C_s. CF is a direct measure of the size of relationship between two classes, for all pairwise relationships between classes in a system.

Polymorphism Factor (PF) metric measures polymorphism potential and is defined (Brito Abreu and Melo, 1996) as:

$$\frac{\sum_{i=1}^{TC} M_o(C_i)}{\sum_{i=1}^{TC}\left[M_n(C_i).DC(C_i)\right]}$$

where

$M_d(C_i) = M_n(C_i) + M_o(C_i)$

$M_n(C_i)$ = Number of new methods,

$M_o(C_i)$ = Number of overriding methods,

$DC(C_i)$ = Number of classes descending from C_i.

Thus PF metric is an indirect measure of the relative amount of dynamic binding in a system.

Analysis Summary: Table 2 shows obtained metrics values to nearest integer for the sake of performance measure. The AHF metric has a value of 100 indicating that all the attributes were declared as private, to adhere to the concept of information hiding. MHF has low value indicating lack in method hiding. Undefined PF reflects lack of inheritance. CF reflects no interclass coupling.

Total Classes	8
Total Methods	10
Total Attributes	30
AHF	100
MHF	2
AIF	43
MIF	18
CF	0
PF	Undefined

Conclusion

Use of Petri net to model industrial controllers, PLC in specific, has been demonstrated. The modeling required a representation to describe inverter transition, monitor states, and ON/OFF-delays that were presented. Java classes were written for a PLC case study illustrative example, whose Petri net model was constructed toward getting the Petri Net Object Oriented Data Structure (PNOODS) realization. Software performance measure was achieved through the investigation of object-oriented design metrics.

References

A. Alfize, 2000. "Petri Nets Object Oriented Data Structure for PLCs Models," International/IEEE Conference Proceedings on Artificial and Computational Intelligence for Decision, Control and Automation in Engineering and Industrial Applications, Monastir, Tunisia, 22-24: 223- 237.

A. A. Desrochers and R. Y. Al-Jaar, 1995. "Applications of Petri Nets in Manufacturing Systems," IEEE Press.

C. Sibertin-Blanc, 1985. "High Level Petri Nets with Data Structures," in Proc. 6th European Workshop on Application and Theory of Petri Nets, Helsinki, Finland,

F. Brito Abreu and W. Melo, 1996. "Evaluating the Impact of OO Design on Software Quality," Proc. 3rd. Int. Software Metics Symp., Berlin,

R. Hassison, S. J. Counsell, and R. V. Nithi, 1998. "An Evaluation of the MOOD Set of Object-Oriented Software Metrics," IEEE Trans. Software Eng.

R. Zurawski and MengChu Zhou, 1994. "Petri Nets and Industiral Applications: A Tutorial," IEEE Transcactions Industrial Electronics,41: 567 – 583.

S-U Guan, H-Y Yu, and J-S Yang, 1998. "A Prioritized Petri Net Model and Its Application in Distributed Multimedia Systems," IEEE Transactions on Computers.

V. R. Basili, L. C. Briand, and W. I. Melo, 1996. "A Validation of Object Oriented Design Metrics as Qulaity Inidators," IEEE Trnas. Software Eng. 22: 751 – 761.