Modelling based on Synchronized Object-Oriented Petri Net

GEORGE CULEA
Department of Energetics, Mecatronics and Computer Science
„Vasile Alecsandri” University of Bacău
Calea Marasesti 157, Bacău
ROMANIA
gculea@ub.ro http://cadredidactice.ub.ro/culeageorge

Abstract: - This paper presents the Synchronized Object-Oriented Petri Net which benefits from the advantages of synchronized Petri nets and the advantages offered by using the concept of object orientation. After a brief description of the basic concepts, the paper focuses on definitions of a marked Synchronized Object-Oriented Petri Net, the functioning of these networks and how they can be implemented in C++. The facilities offered by these networks are highlighted by a case study. This study presents the modelling of transport and storage systems based on Synchronized Object-Oriented Petri Net in a flexible manufacturing system. The paper also presents the Petri net model and SOOPN, including the associated object chart. This example shows the advantages of SOOPN modelling.

Key-Words: - Synchronized Petri net, object-oriented, model, flexible manufacturing system

1 Introduction
One of the most commonly used instruments for modelling and analysis in current research is represented by Petri Nets (PN). After initial definitions of concepts by Carl Adam Petri, PN have developed continuously. Based on a graphic representation and an efficient mathematical system, they enable the modelling, visualisation and analysis of systems, highlighting certain concepts such as parallelism, synchronization, resource sharing and others. The underscoring of causality relationships when describing processes can be achieved by means of synchronized PN. In this case, the firing of a transition is dependent on the existence of a minimum number of tokens in input positions and on the occurrence of the event associated to it. An innovative solution approached by numerous authors lately refers to the employment of object-oriented concepts in PN-based modelling. Thus, it proposes: Object-Oriented PN (OOPN) [1], [3], Timed Hierarchical Object Oriented PN (TOPN), Object-oriented and Differential Predicate Transition Petri net (OO-DPT). The benefits of object-oriented modelling enabled the description of systems easily, intuitively and naturally. The characteristics of object orientation such as abstraction, encapsulation, inheritance, polymorphism and modularity are also specific to OOPN [3]. Based on the object-oriented conceptual framework, the model of the system is composed of a set of objects organised in classes. The objects can be defined by tokens, positions or sub-nets [1] which combine the first two approaches. In case of the last two methods, the messages transmitted between the objects are tokens. The object-oriented technique can also be applied in case of Coloured Petri Nets [2].

2 Synchronized Object-Oriented Petri Net
A Synchronized Object-Oriented Petri Net (SOOPN) model is a variation of OOPN which allows real time system modelling and analysis and especially discrete event system modelling and analysis. Based on the object-oriented concept, the SOOPN model is made up of a set of objects which are grouped in classes. Each object will model a certain sub-net with a set of specific characteristics. The connection between the objects will be realized by functions which establish the manner in which tokens are transferred. The evolution of the modelled system depends on the evolution of the objects’ states.

Definition. A marked Synchronized OOPN is 6-tuple \(\langle C_0, O_i, R, M_{os}, F_{ej}, E \rangle\) where:
- \(C_k\) – is the finite set of class \(C_k\)\{attributes: \(M_k\); methods: \(F_k\);\}
- \(O_i\) – is the finite set of instances of class \(C_k\) (object);
- \(R\) – a Petri Net defined by 4-tuple \(\langle P, T, Pre, Post \rangle\);
- \(M_{os}\) – initial marking of sub-nets;
- \(F_{ej}\) – is a synchronized function associated to each transition which defines firing conditions;
- \(E\) – is the finite set of external events.
In the definition, the following notations were used:
- \( M \) is the marking of sub-nets;
- \( F_j \) is a function associated to each transition which establishes the marking of the sub-net after firing of the \( T_j \);
- \( P = \{ p_1, p_2, \ldots, p_m \} \) is the finite set of places;
- \( T = \{ t_1, t_2, \ldots, t_n \} \) is the finite set of transitions;
- \( P \cap T = \emptyset \), i.e. the sets \( P \) and \( T \) are disjointed;
- \( \text{Pre} \) is a function \( \text{Pre}: P \times T \rightarrow N \) which defines the values of input arcs from transition \( T \);
- \( \text{Post} \) is a function \( \text{Post}: P \times T \rightarrow N \) which defines the values of output arcs from transition \( T \);
- \( o_{T_j} \) - set of input places of \( T_j \);
- \( T_j o \) – set of output places of \( T_j \).

3 The implementation of synchronized OOPN

The SOOPN model is an objectual one, identifying the modelled concepts and containing a synthesis of these objects, of the relations between them, as well as their properties, with reference to both descriptive and experimental characteristics. The construction of models based on the objects it operates with reflects the reality much better than in case of approaches starting from functional aspects. The links between the objects of the SOOPN model can be realized directly, by means of positions, which allows the employment of other types of objects, as well. For instance, we can identify objects belonging to classes which also define other types of Petri nets, such as Neural Petri Net or Hybrid Petri Net [4].

An implementation method could be in C++.

Definitions of classes

class sub-netx

{public:
    vector \( M_{XC} \), \( M_{X0} \), \( W_{X1} \), \( W_{X2} \), \( W_{X3} \);
    sub-netx(M_{XC}[m]=M_{X0}[m]);
    voidF_i() \{M_{XC}=M_{XC}+W_{X1}\};
    voidF_j() \{M_{XC}=M_{XC}+W_{X2}\};
    voidF_k() \{M_{XC}=M_{XC}+W_{X3}\};
    \}

    \}

obx1, obx2, \ldots, obxn – objects of the type sub-netx;

\( t_1 \) – Boolean variable which indicates the validity of transition 1;
\( npx \) – number of positions in sub-net \( x \);
\( mp_{xi} \) – marking of position \( i \) in sub-net \( x = MXC[i] \);
\( c_{Xi1} = Post(P_{Xi}, t_1)-Pre(P_{Xi}, t_1) \);
\( E1 \) – Event \( E1 \) associated to transition \( t_1 \).

4 Case study: modelling based on Synchronized Object-Oriented Petri Net of a flexible manufacturing system

Further on, we will present a modelling solution based on Synchronized Object-Oriented Petri Net for transport and storage systems in a flexible system for manufacturing prismatic parts. The analyzed flexible manufacturing system consists of tree machining centres (MC), preform loading system, buffer for intermediate storage of parts and part discharge system. Machining centres possess toolkits, thus ensuring the automatic processing of preform, the automatic replacement of the tool and manufactured part (figure 1).

Notations used in figure 1 refer to:

IS - input station;
R1 ... R8 - pallet control points in which its rotation occurs
A - activation main loading system
A1 ... A3 - activation loading system CP1 ... CP3;
PBC1 ... PBC3 - parts buffer;
T1 ... T3 - stop manufacturing in CP1 ... CP3;
E1 ... E3 - activation discharge system CP1 ... CP4;
E - activation main discharge system;
MC1 ... MC3 - machining centres;
Re - activation return station for new manufacturing;
DS - discharge station.
Fig 1. Flexible manufacturing system for prismatic parts

Parts enter the machining process through the input station, being set on pallets. The parts remain mounted on the same pallets for all operations until they are discharged through the discharge station. When entering the flux, the pallet stops in front of detector R1 which identifies the preform and if the loading centre is free, the pallet is taken and handled by the loading system. Parts are processed in two stages; first in the machining centre MC1 and then in machining centre MC2 or MC3.

Each machining centre has a buffer device which can contain a maximum number of three preforms which will be processed. The pallet will be activated if in at least one of the machining centres where the processing can take place there is available space for storage in device PB. In order to carry out the processing, the machining centre will read the identification code, then load the prefrom and then select the corresponding part programme. After the programmes ends, the machining centre will place the pallet carrying the part on the discharge station. If the discharge station is available, it will handle the pallet to point R8 and then it will return to point R1. At point R1, if the part’s machining is over, the part will be discharged through the discharge station, and if the part’s processing is not finalised it will return through the return station for further processing. The pallet’s transport will be carried out only if the return station is available.

4.1 The modelling of the flexible manufacturing system for prismatic parts

The Petri Nets model was employed for the analysis of the transport system in case of the flexible manufacturing system for prismatic parts, so that it could render its functioning as well as possible. The scheme realized gives the possibility of carrying out more operations simultaneously. For example, simultaneous transfer operations on the main loading station can be performed, loading operations for buffer areas, or discharge operations. The loading station can be activated or stopped without effects on the discharge station. When a part is taken for processing, if there are parts in the buffer section, they will be handled automatically until the first position is reached. The Petri Network model for this system is shown in figure 2.

Fig. 2 Petri Network model of flexible manufacturing system for prismatic parts

Transitions and places have the following significance:

T1- activation handling system A to R1
T2- activation handling system A1
T3- loading machining center 1
T4- part discharge and transport
T5- pallet/part return
T6- stop machining center 1
T7- start machining center 1
T8- activation handling system A to R2
T9- activation handling system A2
T10 - loading machining center 2
T11 - part discharge and transport
T12- pallet/part return
T13 - stop machining center 2
T14 - start machining center 2
T15 - activation handling system A to R3
T16 - activation handling system A3
T17 - loading machining center 3
T18 - part discharge and transport
T19 - pallet/part return
T20 - stop machining center 3
T21 - start machining center 3
P1 – pallet/part in point R1
P2 - pallet/part in point R2
P3- pallet/part buffer
P4- part processing
P5- pallet/part in point R8
P6 - pallet/part in point R1
P7 – buffer capacity
P8 – available machining center 1
P9 – maintenance machining center 1
P10 - pallet/part in point R3
P11 - pallet/part buffer
P12 - part processing
P13 - pallet/part in point R8
P14 - buffer capacity
P15 - available machining center 2
P16 - maintenance machining center 2
P17 - pallet/part in point R4
P18 - pallet/part buffer
P19 - part processing
P20 - pallet/part in point R8
P21 - buffer capacity
P22 - available machining center 3
P23 - maintenance machining center 3
P30 – available handling system A
P40 - available handling system B

The pallets carrying the preform will have access to the loading section if no other transfer is realized and if there is an available position in the buffer. This model highlights the transport and processing sequences for the three machining centers. Processing at one machining center is possible if it is available (so it does not process another part or is not under maintenance). At the same time, it reveals the common resources represented by handling system A and handling system B (places P30 and P40) which can be used for the transport of a single part. The activation of handling system A (firing the transition T1, T8 or T15) can occur only if it is available and there is at least one available/free position/place in the buffer (at least one marking/token in places P7, P14 or P21).

4.2 Separation into sub-nets for establishing object classes

From the analysis of the model of flexible manufacturing systems for prismatic parts we observed that the series of operations for processing performed by a certain machining centre is approximately the same. As a result, a class of objects can be defined which designates all the operations carried out for processing a part by one machining centre. This class, hereafter named SMA (stream manufacturing), represents a general form which groups the data and functions necessary for the transport and processing flux for a part by a machining centre. This processing flux can be represented by means of a Petri Net as shown in the figure 3.

![Fig. 3  Model of a part processing flux by a machining centre](image)

The SMA class of objects can be defined on the basis of this sub-net. In order to be able to use vectors, a vector class is defined which has the function of operator overloading + which will permit the direct addition of vectors.

```c++
int mp1=1, mp6=0, mp30=1, mp40=1;
class vector {
    public:
```
int v[11];
vector operator+(vector u)
{vector ad;
for(int i=1;i<=11;i++)
ad.v[i]=v[i]+u.v[i];
return ad;
}
void afisare();

class SMA
{public:
vector MXC, MX0;
vector WX1, WX2, WX3, WX4, WX5, WX6, WX7;

SMA(MX0){
MXC=MX0;
}
void F1() {MXC=MXC+WX1;}
void F2() {MXC=MXC+WX2;}
void F3() {MXC=MXC+WX3;}
void F4() {MXC=MXC+WX4;}
void F5() {MXC=MXC+WX5;}
void F6() {MXC=MXC+WX6;}
void F7() {MXC=MXC+WX7;}
};

int npx, i;
npx=11;

vector M1, M2, M3;
SMA Obsnet1(M1);
SMA Obsnet2(M2);
SMA Obsnet3(M3);

t1=TRUE;
for(i=1;i<=11;i++)
if(MXC[i]- Pre(i, 1)<0)
{t1=FALSE;
break;}
Obsnet1.F1();
....
t10=TRUE;
for(i=1;i<=11;i++)
if(MXC[i]- Pre(i, 3)<0)
{t10=FALSE;
break;}
Obsnet2.F3();
....
t17=TRUE;
for(i=1;i<=11;i++)
if(MXC[i]- Pre(i, 3)<0)
{t17=FALSE;
break;}
Obsnet3.F3();
....

voidF1() .. voidF7() represents the calculation functions of the new marking for the sub-net as a result of firing transitions T1...T7.
The Petri Network from figure 2 was decomposed in three sub-nets of the type shown in figure 3. The simulation of the network includes the simulation of sub-nets, and this will become possible by means if objects Obsnet1, Obsnet2 and Obsnet3 of the SMA class type. When they were created, the objects were initialized with the original marking of the sub-net. Vectors M_i have the following values:
M1[11]={mp1,0,0,0,0, mp6,3, 1,0, mp30,mp40};
M2[11]={mp6,0,0,0,0, mp1,3, 1,0, mp30,mp40};
M3[11]={mp6,0,0,0,0, mp1,3, 1,0, mp30,mp40};
Variables mp1, mp6, mp30 and mp40 are global variables, so they were initialized at the beginning of the programme.
We notice that the markings for the three sub-nets have markings mp30 and mp40 in common, and markings mp1 and mp6 are inter-conditioned. These links are visible in the object diagram (Figure 4).
If the decision to fire a transition of the type tj=TRUE is taken, we must analyse if it can be accomplished/fired, respectively \( \forall Pi \in 0Tj \) condition (MXC[i]- Pre(i, j)<0) is true. If the transition is executable, the calculation function for marking (Obsneti.Fj()) will be carried out for the object associated to the sub-net i in which the transition is found.
In figure 4 each object contains an interface with two columns with markings: the first column indicates the general marking of the sub-net and the second column shows the markings in object-oriented Petri Nets.
Calculation functions for the sub-net marking in relation to a certain object establish the new values for the current marking of the Petri Net (mp1..mp23, mp30, mp40). The links between objects are realized by means of global markings (mp1, mp6, mp30 and mp40), these links indicating, in fact, the connections which exist between sub-nets.

5 Conclusion
This paper investigated the definition, implementation and employment of Synchronized Object-Oriented Petri Net (SOON). This type of network integrates the facilities of object-oriented programming with Synchronized Petri Net, so it allows the elaboration of more compact models based on object classes. These models comprise objects which are specific to certain classes, and this enables a clearer exemplification of the interdependent relations between various modules (sub-nets.). However, SOON is a type of network which allows the development of more powerful simulators [5]. The study on the modelling of a flexible manufacturing system highlights the benefits of SOON modelling.

References: