CODING TCPN MODELS INTO THE SIMIO SIMULATION ENVIRONMENT

Miguel Mujica[(a)], Miquel Angel Piera[(b)]

[(a,b)]Autonomous University of Barcelona, Faculty of Telecommunications and Systems Engineering, 08193, Bellaterra, Barcelona

[(a)]miguelantonio.mujica@uab.es, [(b)]miquelangel.piera@uab.es

ABSTRACT

The coloured Petri net formalism has been widely used by scientific community to perform not only research and behavioural analysis of models but also as a simulation tool to carry out systems' analysis. Its characteristics allow a better understanding of the causal relationships present in systems. On the other hand discrete event system simulation software has evolved in order to reduce the efforts needed in simulation projects; the developers have improved the developing paradigm, graphical interface and analysis tools among other aspects. Unfortunately when dealing with big projects the simulation paradigms hinder the understanding of the causal relationships of systems. In this paper a way to integrate the timed coloured Petri net modelling formalism with the SIMIO simulation software is presented in order to overcome this problem.

Keywords: Discrete event systems, scheduling, decision support systems, timed coloured Petri nets, Simulation, Simio.

1. INTRODUCTION

Simulation is a very well recognized methodology which possesses a high descriptive level. Most commercial simulators have several modules to analyze data, implement the relationship between elements and to perform simulation experiments. In recent years discrete-event system simulation developers have put focus on improving the characteristics of the simulation software in order to reduce the efforts needed to develop a simulation project. Some simulators such as PROMODEL(www.promodel.com) or Witness (www.lanner.com) have developed graphical modules in order to improve the graphical representation of systems, but unfortunately its original source code remains the same. SIMIO simulation software (www.simio.com) was developed by the creators of the very well known ARENA (www.arenasimulation.com) software. They have developed a novel simulation program that improves several aspects of the simulation software; they combine the processes approach with the object oriented paradigm (OOP), 2D-3D visualisation among other features that makes it a very powerful tool. With the use of the OOP the developing phase of the simulation project is improved taking advantage of the characteristics of this approach (encapsulation, inheritance, polymorphism). The developing and analysis characteristics present in SIMIO can be further improved if some underlying logic is added in order to achieve a better understanding of the modelled system.

The timed coloured Petri net formalism (TCPN) has characteristics that allow modelling true concurrency, parallelism or conflicting situations present in industrial systems (Jensen 1997, Moore & Gupta 1996, Mušič, 2009). Unfortunately when the TCPN formalism is used with the purpose of systems analysis it lacks of tools that can be used to perform statistical analysis by a user who is not expert in the TCPN field (industrial engineers, process engineers, managers, etc). Furthermore if the model or the results are going to be presented to decision makers within a firm the graphical representation results difficult to understand when the people is not familiar with the formalism. In the best cases the token game is the one that can be obtained which is the case of CPNtools (www.daimi.au.dk/~cpntools/) or Petrisimm (http://seth.asc.tuwien.ac.at/petrisimm/).

SIMIO simulation software has the processes paradigm that allow to code the TCPN firing rules using simple steps thus the casual relationships between the generated events can be governed by the TCPN semantic rules. The advantage of integrating TCPN models to govern some activities of the SIMIO model is that it is possible to develop models using the formalism and at the same time take advantage of the characteristics and the graphical potential that SIMIO possesses. The article is organized as follows. Section 2 presents the TCPN modelling formalism, section 3 discusses some characteristics of SIMIO; section 4 describes a way to code TCPN models using some elements of SIMIO for governing some events of the modelled system. Section 5 discusses briefly the graphical and analytical capabilities of SIMIO and section 6 gives the conclusions of the article.

2. TIMED COLOURED PETRI NETS

Coloured Petri Nets (CPN) is a simple yet powerful modelling formalism, which allows the modelling of complex systems which present an asynchronous, parallel or concurrent behaviour and can be considered discrete event dynamic systems (Jensen & Kristensen 2009). The formalism allows developing models without ambiguity and in a formal way. It is possible to model not only the dynamic behaviour of systems but also the information flow, which is an important

characteristic and very useful in industrial systems modelling and decision making.

In order to investigate the KPI's (Key Performance Indicators) at which the industrial systems operate under different policies, such as scheduling, resource occupancy, costs and inventory among others it is convenient to extend CPN with a time concept. This extension is made by introducing a global clock for the model, time stamps for the entities and a time delay for the model transitions; the nets that use this extension are known as *timed coloured Petri nets*(TCPN). When using TCPN the global clock represents the model time, and the time stamps describe the earliest model time at which the entities of the model, graphically represented by dots (tokens), can be used for the transition evaluation process (Jensen 1997). A token is *ready* if the correspondent time stamp is less than or equal to the current model time. If the token is not ready, it can not be used in the transition enabling procedure.

The transitions of the TCPN are used to model the activities of the real system and a time delay is attached in order to simulate time consumption of a certain activity.

It is a common convention to use the sign @ to denote time in the elements of the model. When it is attached to transitions, it specifies the time consumption.

The formal definition of TCPN is the following one.

Definition 1. The non-hierarchical TCPN is the tuple:

TCPN = (P, T, A,$\sum$, V, C, G, E,D, I) where

1. P is a finite set of places.

2. T is a finite set of transitions T such that $P \cap T = \varnothing$

3. $A \subseteq P \times T \cup T \times P$ is a set of directed arcs

4. $\sum$ is a finite set of non-empty colour sets.

5. V is a finite set of typed variables such that Type [υ] $\in \sum$ for all variables $\upsilon \in V$.

6. C: $P \rightarrow \sum$ is a colour set function assigning a colour set to each place.

7. G: $T \rightarrow$ EXPR is a guard function assigning a guard to each transition T such that Type [G(*T*)] = Boolean.

8. E: $A \rightarrow$ EXPR is an arc expression function assigning an arc expression to each arc *a,* such that:

Type [E(*a*)] = C(p)

Where p is the place connected to the arc *a*

9. D: $T \rightarrow$ EXPR is a transition expression which assigns a delay to each transition.

9. I is an initialization function assigning an initial timed marking to each place p such that:

Type [I(*p*)] = C(p)

EXPR denotes the mathematical expressions associated to the elements of the formalism (variables, colours, logic conditions) where the syntax can vary when coding the formalism in a programming language. The TYPE[*e*] denotes the type of an expression $e \in$ EXPR, i.e. the type of values obtained when evaluating *e.* The set of free variables in an expression *e* is denoted VAR[*e*] and the type of a variable *v* is denoted TYPE[*v*].

The formalism can be graphically represented by circles which represent the place nodes and rectangles or solid lines that represent the transition nodes. The place nodes are used to model resource availability or logic conditions that need to be satisfied. The transition nodes can be associated to activities of the real system. The developed models can be analyzed with the help of available academic software such as CPNtools or PetriSimm. These types of software programs are commonly used by the scientific community in order to carry out analysis of the model to verify behaviour or systems' performance. This analysis can be performed through simulation of the system making use of the token game or performing state space analysis (Christensen et al. 2001, Mujica & Piera 2011, Mujica et al. 2010).

3. SIMIO SIMULATION SOFTWARE

The most common discrete event system (DES) simulators have been coded using the *activity scanning* or the *event scheduling* approach (Shannon 1997). These approaches have been developed to reproduce the behaviour of systems whose states change in discrete instants of time. Most of the commercial tools are very efficient in modelling DES using one of these approaches or even combinations of them.

With the development of fast CPU processors, powerful graphic cards, efficient statistical analysis tools etc., simulation has become more important than before. Some years ago most of the effort during a simulation project was put on the development phase of the model but with the development of more efficient software programs this effort has been considerably reduced. The available tools allow analysts to spend more time in the analysis phase of the simulation project. In order to reduce the lead time of the simulation project some developers have invested time in implementing new paradigms in their simulation products; this is the case of SIMIO. This software has been coded merging the OOP together with the processes paradigm in order to reduce the number of blocks needed to develop complex models. Based on these programming paradigms the developers coded SIMIO as a collection of objects that are instantiated from classes. These classes were designed using the principles of abstraction, encapsulation, polymorphism, inheritance and composition (Pegden 2007).

Making use of the few available objects it is only necessary to add new functionalities (processes) to the original ones in order to have additional behaviour or logic (inheritance) or even overriding the original one.

Since the objects in SIMIO follow the encapsulation principle their implementation is sealed from the outside world. The composition principle allows building new classes combining the existing ones. These characteristics allow great flexibility when developing a model.

One important aspect of the simulation project is the implementation phase which depends strongly on how the results are presented to decision makers.

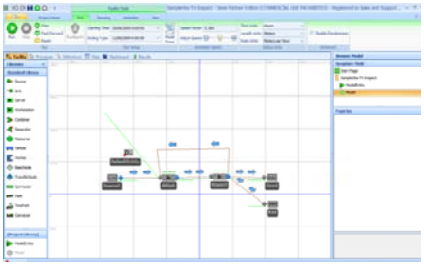

Figure 1. Example of a SIMIO model in 2D

In this sense, the graphical interface of SIMIO has been developed in such a way that it is very easy to have very good-looking results in short time. It can switch between 2D and 3D depending on which kind of task is being performed (development or validation). Figure 1 presents a typical 2D view of the SIMIO model and Figure 2 the 3D view of the same model.

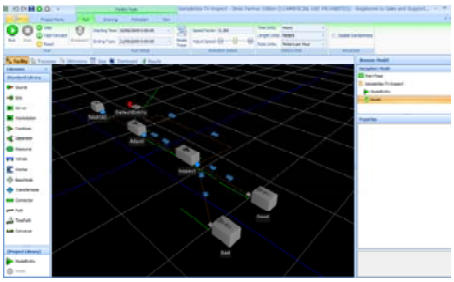

Figure 2. SIMIO model in 3D view.

The switch between both views is as easy as clicking the mouse. It is important to mention that SIMIO comes with a basic graphical library but it can be extended with graphical models from the GOOGLE 3D warehouse (http://sketchup.google.com/3dwarehouse/?hl=en).

4. INTEGRATING THE TCPN MODELS IN THE SIMIO ENVIRONMENT

Due to the dual developing paradigm used in SIMIO (process/object) it is possible to extend the functionality of the SIMIO objects with a sequence of steps which fit the purpose of coding the TCPN rules. These rules can be easily implemented using some elements of SIMIO such as the following ones.

Objects:

- SOURCE
- SINK
- SEPARATOR
- TRANSFER NODE

Steps:

- Decide
- Search
- Destroy
- Transfer
- Assign

Elements:

- Station

Other objects can be also added after the development of the model in order to improve the visual aspect of the model or to make the simulation more detailed.

The model of Figure 3 will be used to illustrate the implementation of the TCPN rules into SIMIO.

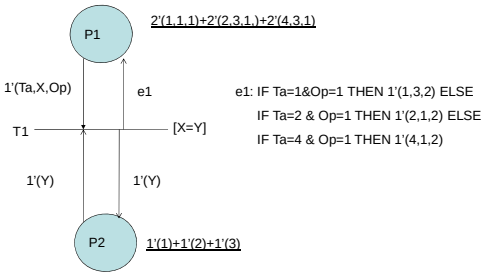

Figure 3. TCPN model

4.1. Modelling the Place nodes

The place nodes are modelled using the *Station* element which has been developed with the objective of storing entities.

The stations are *Elements* that need to be defined in the *Definitions* area of the software. Figure 4 illustrates the station definition.

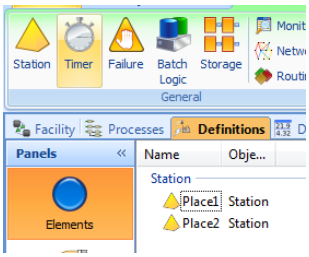

Figure 4. Station definition

In this figure Place1 and Place2 represent the place nodes of Figure 3.

Graphically a *Station* is not related to any predefined object of the *Standard Library* in SIMIO, but its graphical visualization can be performed associating a *detached QUEUE* to the *Station* element. The entities of SIMIO can have any 3D appearance which allows exploiting the graphical potential of SIMIO. Figure 5 gives an example of a detached queue associated to the place (Place1) in the SIMIO model.

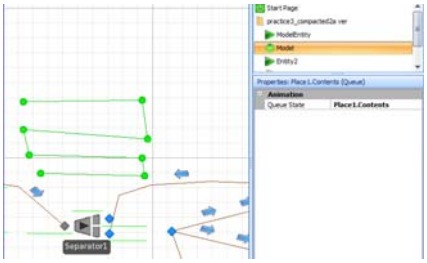

Figure 5. A detached queue for the Place1

The QUEUE is just added into the *facility area* and the association is performed in the *Properties* window:

Queue State| Place1.Contents.

Depending on the type of model the resource availability can also be modelled using another predefined object which can be used as a resource such as the SERVER or WORKSTATION; moreover with the use of a predefined object the graphical representation can be enhanced by taking advantage of the object's behaviour. In the example presented here the two place nodes will be modelled making use of two *Station* elements.

4.2. Defining the Token Colours

The entities that represent the tokens are generated using the SOURCE object and they are destroyed using the SINK object or the **Destroy** step in the *Processes* area of SIMIO.

The entities in SIMIO can also be extended with attributes. These attributes work as the fields of a record in any simulation language and they are associated to the entities in the *States Definitions* area. The states in SIMIO have been conceived as variables whose values change during the simulation run. They can be modified to include as many attributes as colours present in the token. The states can be of different types: Real, Integer, Boolean, Date, or Strings.

4.3. Modelling the TCPN Restrictions

The dynamics of the TCPN models are governed by the input/output flux of tokens that takes place when a transition is fired. The *Processes* area of SIMIO is used to code the TCPN logic which evaluates the constraints to unchain the processes modelled by the transitions.

In order to enable a transition the following conditions must be fulfilled:

- the number of tokens in the input place nodes are greater than or equal to the arc weight
- the colours of the correspondent tokens must have the particular value that is stated in the arc inscription
- The colour binding must satisfy the boolean expression stated by the *Guards*

The evaluation of the restrictions imposed by the arcs and guards can be performed using a combination of **Decide, Search, Assign**, and **Destroy** steps such as the one depicted in Figure 6:

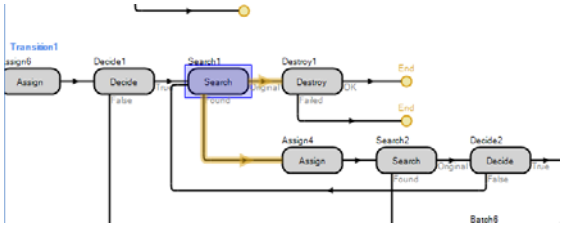

Figure 6: Evaluating the restrictions

- Decide1: the **Decide** step works like an IF..THEN..ELSE instruction in any programming language. The *Decide Type* attribute (Figure 7) must be put in *condition based* in order to state the expressions that need to be satisfied. The attribute *Expression* is used to specify the logic conditions that must be fulfilled by the entities of the TCPN model. The expression for the example model can be written in the following way.

Place1.Contents>0&Place2.contents>0

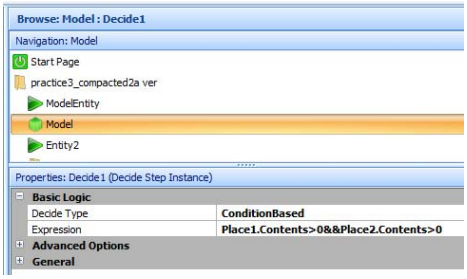

Figure 7: Decide condition

The Decide1 is used to verify that both stations (place nodes) have at least the number of entities (tokens) imposed by the arc weights.

A similar expression can be used if one place is modelled using a SERVER or any other object of SIMIO (Resource is the object's name).

PLACE1.Contents>0&Resource.ResourceState==0

- Search1: The **Search** is used to perform searches within the list of elements of the stations or queues under particular restrictions. If the arc expressions have known information such as constant values, this condition is stated using the *match condition* attribute of the Search step. When the Search step has found one entity that satisfies the restrictions imposed by the arc, this entity goes out from the *found* side of the step and continues through the rest of the flowchart to check the conditions of the remaining place node. The latter is performed in the *Search1* of the flowchart of Figure 6. If the *Search1* founds an entity that satisfies the restriction, it leaves the Search step through the *found* side of the step. If it does not found any entity that fulfills the

restrictions then it leaves the step and goes to the *Destroy1*step. If it finds an entity that satisfies the restriction, it flows through the *Found* side of the search step and continues to the Assign4.

- *Assign4*: is used to bind the value of the entity to the variable which will be evaluated by the Guard (X variable)
- *Search2*: This step is used to perform searches in the Station2 (Place2) under the restrictions imposed by the value of the variable (X=Y). If the search obtains one entity that satisfies the restrictions it flows through the *found* side of the step and continues with the animation activities. If no entity is found then the original one continues to the *Decide2* step.
- *Decide2*: This step checks whether the last search step found or not an entity that satisfied the restrictions, if not it sends the entity to perform the cycle again testing another combination of entities from Station1 and Station2.

4.4. Sending the entities to the facility

In order to animate the capturing of resources, the entities can be sent to the *facility* window where the animation can be performed depending of the resource that is used (colours of tokens from place P2). Figure 8 illustrates the flowchart section where the successful entity is sent via *transfer step* to the correspondent node within the *facility* window based upon the attribute value of the correspondent token. In this example the attribute value can have three different values depending on the resource used (1,2,3) therefore three possible outcomes are included in the flowchart (two different *Decide* steps).

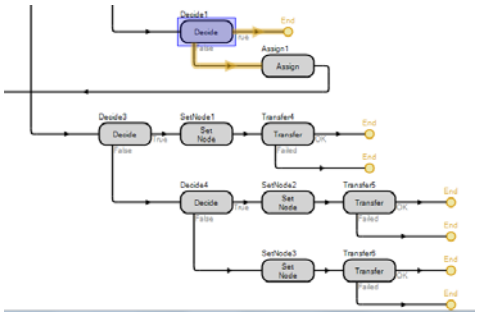

Figure 8. Sending the entities to the correspondent nodes

The **Decide** steps are used only to evaluate the colour attribute that specifies the kind of resource being used and based on the result of the evaluation the entities are sent to particular locations within the *Facility window*.

The continuous evaluation of Transition T1 is performed in the facility window making use of a SEPARATOR object after each *Server* has been released. It makes a copy of the entity that comes from the Server and it is sent to the node where the transition T1 is evaluated.

4.5. Time Consumption

The time consumption is modelled in a straightforward way using the time attributes available in every SIMIO object (these attributes can be found in the properties window). In most of the objects it is possible to associate time to the activities performed by the objects (entering the object, exiting the object, seize, delay etc). In the case of a predefined object such as the SERVER the time consumption is modelled by the *delay* or any other time-consuming activities available in the object (Figure 9). The advantage of using these properties is that it is possible to model activities which consume time in a deterministic or stochastic way.

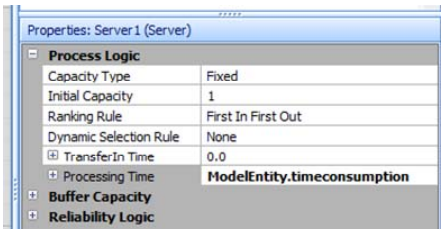

Figure 9: Defining the time consumed by an activity

4.6. Attaching a Transition to the SIMIO model

Finally, in order to govern the flow of entities in the SIMIO model, it is necessary to attach the logic (coded in the process section of SIMIO) of the TCPN to an event of the SIMIO model. All the objects in SIMIO have the *Add-On Process Triggers* which enumerate all the possible events associated to the object. These *triggers* are used to call the user-defined processes when a particular event occurs. The processes (transition evaluations) can be called at almost any point of the SIMIO model within the *facility* area.

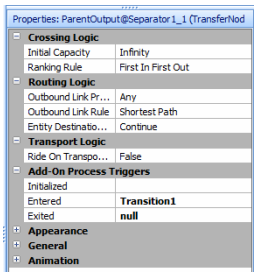

Figure 10. Adding the transition

Figure 10 illustrates the *Properties* window associated to an object. In this case *Transition1* is called every time the entity *enters* the object in run-time mode. When this event happens, the entity behaviour is governed by the logic defined by the flow diagram of the process used to model *Transition1*.

4.7. Modelling the output arc

The last element to be modelled in the example of Figure **3** is the output arc whose output function assigns the values of the colours for the output tokens.

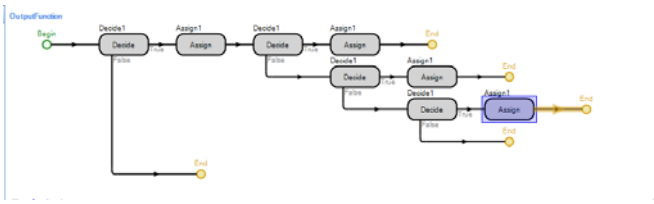

Figure 11. Coding the output function

This function is coded in SIMIO in a simple way (Figure 11) making use of nested combinations of **Decide-Assign** steps in order to evaluate the values of the attributes of the entities and afterwards based upon those evaluations the new ones are updated with the **Assign** steps before sending the entities (via a **Transfer** step) to the facility window.

5. ANALYSING THE SYSTEM

Once the TCPN rules have been defined within the *Process* area and attached to an event of the SIMIO model the simulation can be performed in the typical way.

5.1. Experimenting with the TCPN/SIMIO model

As it has been mentioned, one advantage of integrating CPN models in SIMIO is that it is possible to use the analytical tools that are integrated within SIMIO. Using those tools it is possible to obtain KPI's that allow the decision makers to evaluate the best engineering decisions. This analysis is performed making use of the *Experiment* tool (Kelton et al. 2010) which allows performing experiments (replications) of the model in order to gather information from the model. After performing the experiments a report or a matrix called *Pivot Grid* are generated and they can be used to analyze and filter the information obtained from the experiment.

Performance indicators such as resource utilization, average number of entities in the stations, average processing time, etc. are the kind of information that can be obtained from the analysis performed.

5.2. Improving the graphical view of the model

One great attribute of SIMIO is the graphical interface. Since it can switch easily from 2D to 3D and the graphical models can be downloaded directly from GOOGLE 3D Warehouse the resulting models can be graphically improved without effort.

Figure 12. Final view of a model

The model can be constructed starting with the available objects in SIMIO and afterwards download the figures that will represent the actual objects in the system. Figure 12 shows a view of a manufacture model that has been graphically enhanced using the Google 3D Warehouse models.

6. CONCLUSIONS

A way of implementing TCPN models making use of the analysis and graphical potential of SIMIO software has been presented. This approach can be used for developing simulation projects taking advantage of the characteristics of the TCPN formalism and the graphical and analytical tools available in SIMIO. In addition the GUI allows the user a better understanding of the real system and it allows giving a better aesthetic appearance if the simulation model is used as a tool for decision making.

REFERENCES

Christensen, S; Jensen, K; Mailund, T; Kristensen, L.M., 2001."State Space Methods for Timed Coloured Petri Nets", in Proc. of 2nd International Colloquium on Petri Net Technologies for Modelling Communication Based Systems, pp. 33-42, Berlin, 2001.

Jensen K; Kristensen L.M.; 2009."Coloured Petri Nets: Modelling and Validation of Concurrent Systems", Springer,2009.

Jensen K.; 1997."Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use", Vol.1 Springer-Verlag. Berlin, 1997.

Kelton, W.D.; Smith, J.S.; Sturrock, D.T.; Verbraeck, A.; 2010."Simio & Simulation: Modeling, Analysis, Applications", McGraw-Hill, Boston, 2010.

Moore, K.E.; Gupta, S.M.;1996." Petri Net Models of Flexible and Automated Manufacturing Systems: A Survey", International Journal of Production Research, Vol. 34(11), pp. 3001-3035, 1996.

Mujica, M.A.; Piera M.A.; 2011. "A Compact Timed State Approach for the Analysis of Manufacturing Systems: Key Algorithmic Improvements", International Journal of Computer Integrated Manufacturing, Vol.24 (2), February 2011.

Mujica, M.A.; Piera, M.A.; Narciso M.; 2010. "Revisiting state space exploration of timed coloured petri net models to optimize manufacturing system's performance", Simulation Modelling Practice and Theory, vol.8(9), pp. 1225-1241, Oct. 2010.

Mušič G., 2009."Petri Net Based Scheduling Approach Combining Dispatching Rules and Local Search", in Proceedings of the I3M2009 Multiconference, Tenerife, Spain, 2009.

Pegden, D., 2007."SIMIO: A new simulation system based on intelligent objects", in Proc. of the 39th winter simulation conference, 2007.

Shannon, R. E. , 1997."Systems Simulation, the Art and Science", Englewood Cliffs, N. J., Prentice Hall, 1975.