Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems

Javier Rodriguez[1]

[1]Affiliation not available

February 3, 2018

1 Introduction

This report will explain the article written by *Kurt Jensen*, *Lars Michael Kristensen* and *Lisa Wells* about the concept of Coloured Petri Nets and CPN Tools program used to perform this diagrams. This document called *Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems* and published at the *International Journal on Software Tools for Technology Transfer (STTT)* in 2007, explains what are CPN and all the elements that can appear in a model, also shows one of the programs to perform this type of schemas and different possible tools used to test and reproduce various procedures.

Kurt Jensen, professor in Department of Computer Science at Aarhus University since 1999, head of the Steering Committee for the International Petri net Community from 2004 to 2011. Lars Michael Kristensen, professor in Department of Computing at the Bergen University College since 2009, actual head of the Steering Committee for the International Petri net Community. And Lisa Wells, research and innovation specialist health IT lab, administrator of the projects *Denmark as a telemedicine pioneer country* and *URB Grade.*

2 Context of the work

At the moment of representing the process followed by a system along its procedure, a graphical representation is one of the best choices to plot it. The Petri Net, defined in the 60's by Carl Adam Petri, confers a user to show in a different way the representation of all the steps followed in the performance of a system. Coloured Petri Net (CPN) is an improvement of the previous diagrams due to the fact that it is possible to distinguish between tokens and attach to them different values.

The problem of graphical representation has been present along the history, we have seen that some other standards have been created like Unified Modeling Language (UML), Business Process Model and Notation (BPMN) or Event-driven Process Chain (EPC). This standard offers a different way of showing the information and also has an advantage compared with the others shown previously: the fact that has a precise mathematical definition and a suitable process analysis theory.

In other words, at the moment of defining through a diagram the performance of a system introducing different parameters that can affect the behavior of each step as well as the response or the way to follow, CPN is one of the best ways to show all this characteristic in a very comprehensible and simplified manner.

3 Positioning and contributions

System engineering is the branch that embraces multiple activities as requirements engineering, design, implementation, testing and deployment. One way of fulfill this specifications, and specially to build an executable model to develop concurrent systems, is using CPN. CPN systems allow to the user to represent systems as a set of states and events, called transitions and that make the changes between states. CPN Tools is a program that makes possible to implement this type of diagrams and to make different analysis applying multiple variables and seeing each response of the system.

Test and visualize the different responses that a developed system performs is one of the crucial points to take into consideration at the moment of developing to see that satisfies all requirements and has the proper response depending on the different parameters inserted along the procedure. For testing purposes the user sets breackpoints to determine if the system follows the correct way and data collectors to check the values of all variables. Visualization is also a head problem to solve during the construction of a system, with that the user is able to have a global idea of the dimensions of the system and also of the relations between the different parts that are related along all the procedure.

This work developed by Kurt Jensen, Lars Michael Kristensen and Lisa Wells shows the way to represent a system through the CPN and also provides the basis to perform it using their own program CPN Tools to perform the visualization and the different tests in a proper manner.

3.1 Coloured Petri Nets

A CPN is a graphical drawing that contains different elements, distinguished by their shape and the info that contain. In the example shown in the following figure, we are going to explain the meaning of each element present in a basic CPN diagram.

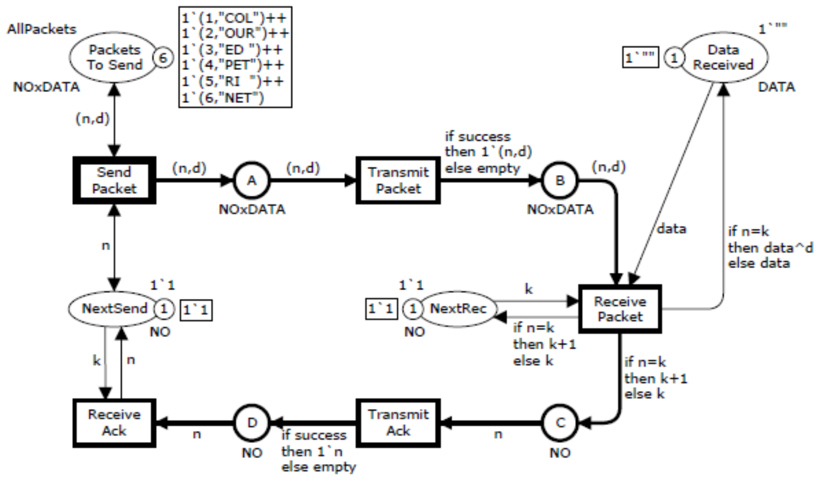

Figure 1: Figure 1. CPN diagram of a string packet delivery system.

Every system is formed by *places* and *transitions* represented by circles/ellipses and rectangles, respectively. They are connected using different *arcs* that are represented as arrows. As it is shown, there are any pair or nodes of the same type connected, that is one of the restrictions that is forbidden.

The inscriptions placed in all the elements presented let the user have a clear idea of the steps followed and the different variables shared for each. The small circle with a number inside called *current marking*, present on different places, shows the size of the data hold by those elements and the box next to the the content. At the upper left part, we can see the different tokens that node "Packets To Send" has and we also see that in node "Data Received" the bucket is empty as it is represented by a simple double quotes marks.

At the arcs, it can be shown that some variables ('n','k' and 'd') of different types ('n' and 'k' as integers and 'd' as string) are shared that can be compared with the ones represented on the places designated as A, B, C and D ('n' equals to NO and 'd' equals to DATA). Also it is possible to place different expressions to determine when a the system has to enable a change of transition. When a transition has place with any restriction all the parameters are removed from the input place (input socket) just to the output arc values (output socket).

In Figure 2, we can see the first segment of the system and how the values change when the system is executing.

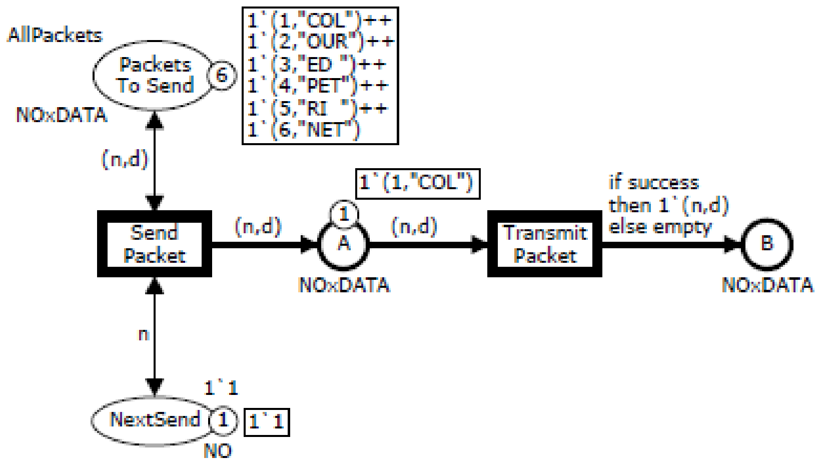

Figure 2: Figure 2. State of the system on the first step.

Once the main elements of the system are explained, it is necessary to know the rules that the system follows to execute each transition. Each turn is called *step* and it consist of a concurrently enable binding multi-set elements, the procedure is simple: if any of the transitions is possible to be made, the change is accomplished. So as we can see in Figure 3, the same packet is sending again during each step until the first loop is completed.

During an execution of a CPN model it is important to determine those markings that are reachable using just one path , called *deterministic*, and those that can happen with different combinations, called *non-deterministic*. Being conscious of this cases makes easier to avoid errors during the execution of the system.

A way to simplify the diagram but continue having the same performance is to create *modules*. A module consist on a box containing all the elements corresponding to a segment of the system, that internally has the same performance but graphically is represented with lower detail. It is commonly used when a segment is clearly understood and helps to have a cleaner view of the graph.

Previously we have talked about the problem of duplicated sent packages, there is a way to avoid that behavior just adding timing information to the system. Using this, it is possible to manage how frequently the system will active each of their nodes and the time that needs each place to perform their own task. Another difference with untimed CPN is that tokens can carry a second value called time stamp just to signal the current time value, the time is represented with a number representing the 'units of time'[1] that it takes and written after a '@'. Also it is possible to add an undetermined delay, describing a function that

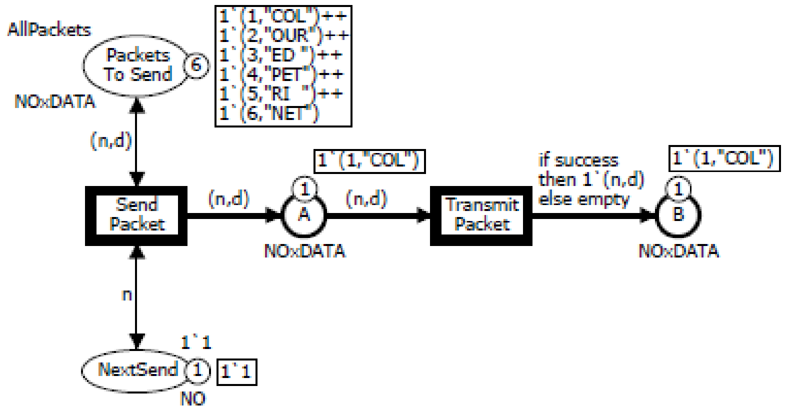

Figure 3: Figure 3. State of the system on the second step.

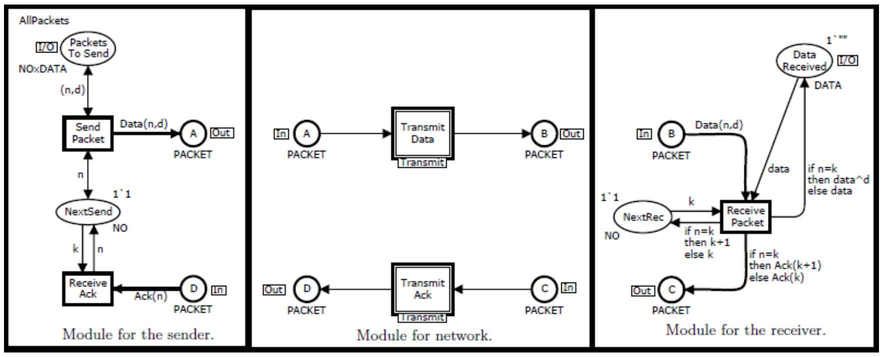

Figure 4: Figure 4. Segmentation of the system in different modules.

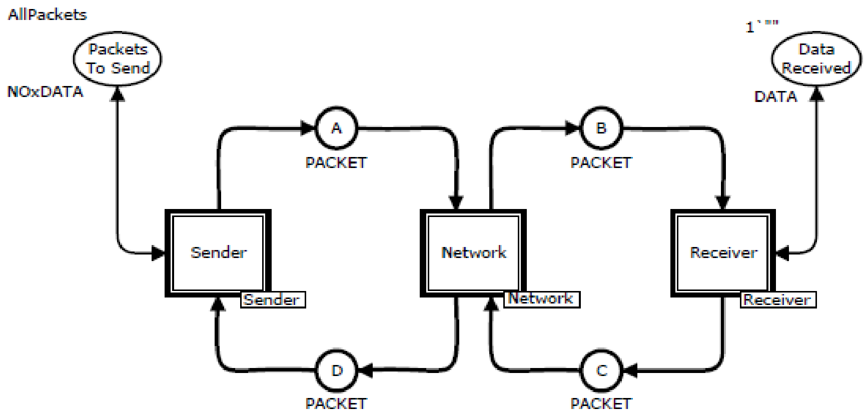

Figure 5: Figure 5. System represented using modules.

returns a random value between two predetermined, or waiting times, just using a constant, and adding it to the time stamp. The same diagram, but with time parameter added, is shown in Figure 6.

In case there exists any element that is declared without time stamp then it would be always prepared to make a transition on each step when its conditions are fulfilled.

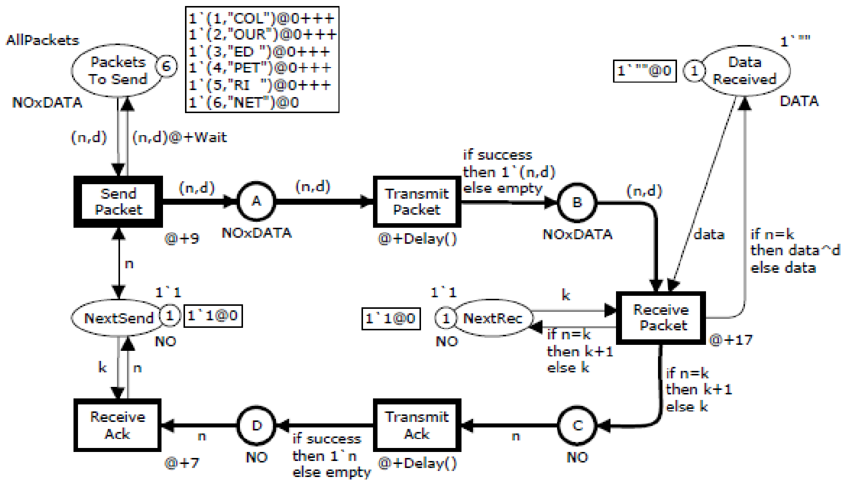

Figure 6: Figure 6. CPN diagram with time parameter added.

3.2 CPN Tools

In this section we are going to give a brief introduction to the GUI of CPN Tools, explaining the most relevant commands to create and operate with Coloured Petri Nets.

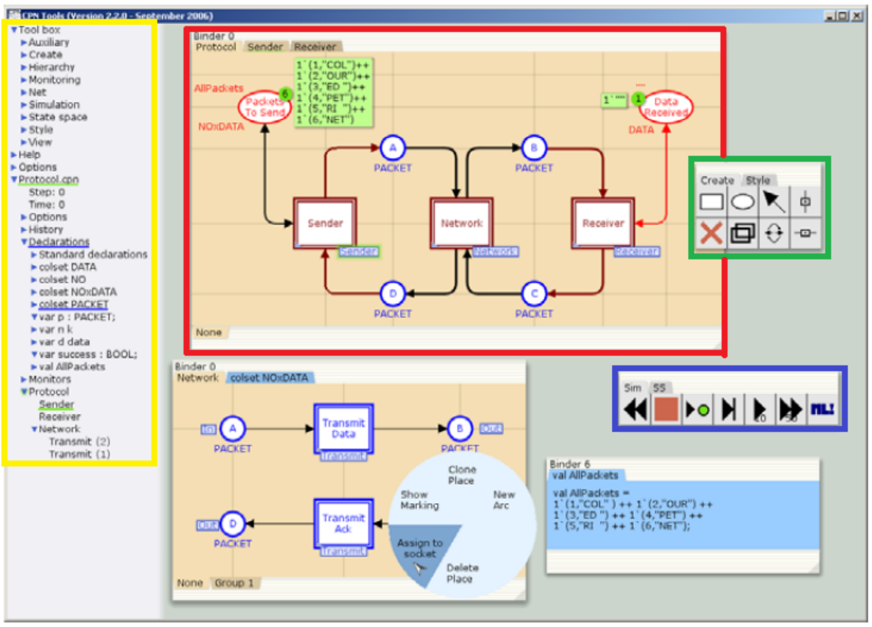

Figure 7: Figure 7. CPN Tools screenshot.

At the first sight, we can distinguish two main areas: the first one highlighted in yellow, called the *index* that contains all the tools necessaries to create the basic elements to create the diagrams and also allows to clone the ones present at the current diagrams, and the rest, called the *workspace*.

In this last part it is possible to appreciate five binders and a circular marking menu, placed at the bottom of the screen. There are two main different types of binders depending on the functions developed by them. The one marked with a red rectangle is used to developed a desired diagram, there the user can place different nodes and arcs that will define the complete system, in this case we can see that the diagram represented is equals to the one shown in Figure 5. The green one is a set of tools used for creating simple elements and that in this case is used to be more accessible because we can find the same functionality by pressing the tab with the same name placed on the index. The blue one, is very similar to the previous one, but in this case it is used for simulation purposes to see if the system runs on the correct way.

Although this programs makes easier to plot different diagrams, the main advantage at the moment of their use is the possibility to simulate the different procedures and see each one of the changes that suffers the diagram and its variables. Setting different breakpoints and seeing the values of the variables placed along the nodes and the ones that are being transmitted. Also, instead of placing a simple breakpoint that will stop each time the procedure is reached, a condition can be applied in a concrete node to stop the simulation just at the moment desired using a *monitor*. Each monitor can be personalized to take into account different parameters depending on the requirements.

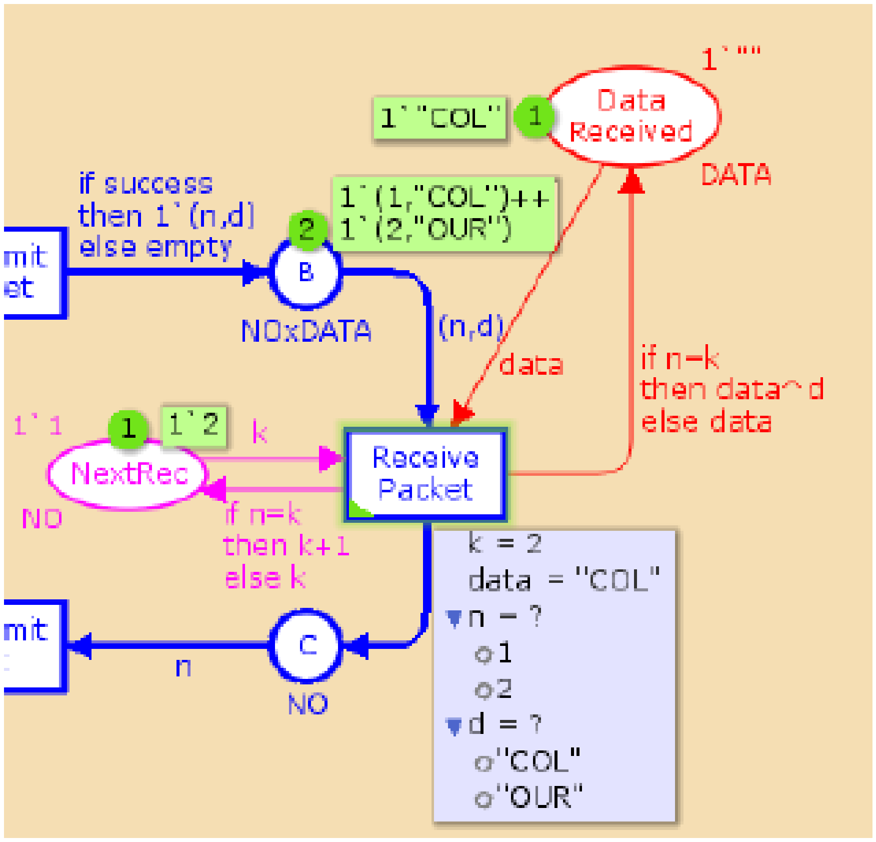

Figure 8: Figure 8. Data shown during simulation.

In the case shown in Figure 7, we can check that once the first package is received successfully the value of k change to let the system pass the next element to the receiver bucket. All this test can be performed before getting the *dead marking* that indicates the end of the simulation. Another way to see all the changes suffered by the variables placed along the diagram is to have a view over the *simulation report* that stores all the information of the diagram in each step of the simulation with the node involved and the values that have changed.

There exist another way to represent CPN diagrams called *full state space*. Where each reachable marking will be represented by a node and every binding element will be substitute by an arc. Taking the same example, but in its untimed form with a little modification to limit the total number of tokens that can be on the intermediate nodes and so eliminate the infinity of reachable markings, we will show in the next image the graphical representation.

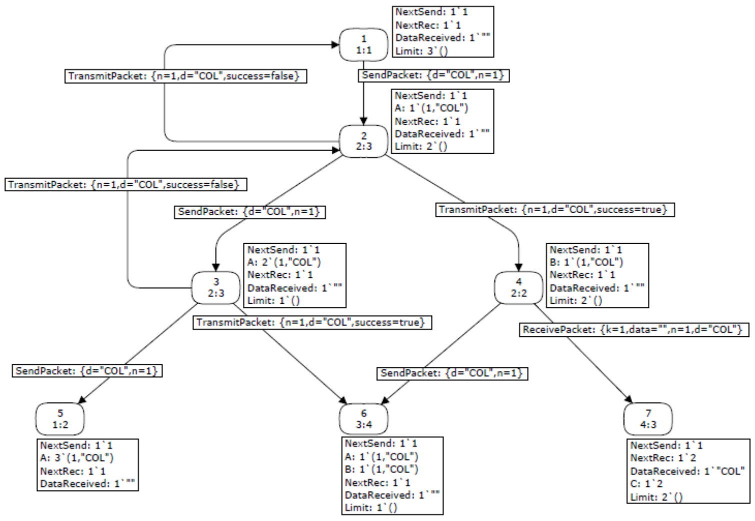

Figure 9: Figure 9. State space fragment of CPN model example.

Reachable markings and bindings elements are represented as we had said before, each node have their own *node descriptor* that provides all the information related with the corresponding node and also shows the interaction existing between all of them.

As seen with the simple simulation of the traditional diagram, there also exists a state space report that can be consulted to see in a brief way all the parameters, nodes, arcs,... and also the upper and lower bounds of the elements that can be held by each different place. It can also be consulted the *dead marking*, that points out the number of different steps been followed just to reach to the end, live transitions, that marks if there is a reachable marking that always have a possible next step to follow in every moment, and dead transitions, if there are no reachable markings in all the process.

Apart from this methods to simulate the system, it is possible to run different simulations and make multiple graphs showing different events or parameters selected by the user using all the data collected along those procedures. With this procedure it is possible to see if the execution of the system is efficient, for example checking in a timed CPN diagram the average time taken by the system to finish with all the transmissions. With that it is possible to obtain results as the one of Figure 10.

Apart from this plots representing the data collected and others related with the values reached during a simulation, it is helpful to use other ways to improve the visual part of the system at the moment of studying its behavior [3]. Related to this part it is possible to use a Message Sequence Chart (MSC), that in our example will follow the way followed by a packet from the sender to the receiver, and has an appearance similar to the UML sequence diagrams. An animation graphics lets also to simplify the schema and to see the performance of the entire system dynamically.

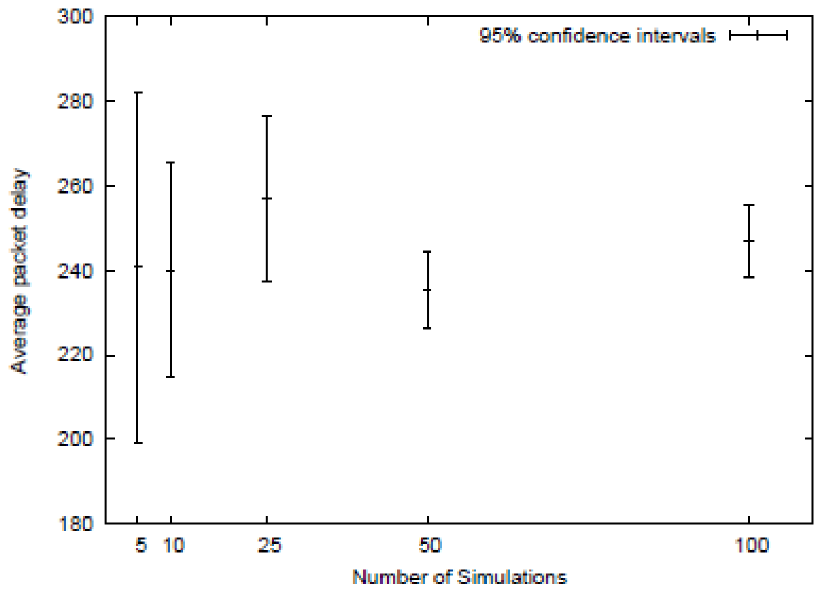

Figure 10: Figure 10. 95% confidence intervals for average packet delay.

4 Conclusion

Dealing with concurrent system, test and debugging procedures are necessary to make correct analysis. At the moment of representing a system and its procedure, a graph is particularly useful to show the tasks and all elements involved in the execution. Petri Nets allow the user to reproduce, through an intuitive plot, the system that has created and to cope with concurrent systems better than with other existing standards. The need of expressing a bigger amount of information making it possible to distinguish between different types of tokens, unleashed to improve this standard to Colored Petri Nets.

The automation and standardization of the procedure of creating those diagrams, the possibility of testing more efficiently and other multiple applications make the team to develop a program able to make all this task: CPN Tools. Though the simple representation of the diagram make it possible make all the test and obtain the necessary calculations, the use of other systems and platforms helps to find new ways to understand and improve the realization of an analysis [2].

5 Bibliography

References

Giovanni Chiola, Claude Dutheillet, Giuliana Franceschinis and Serge Haddad, *A Symbolic Reachability Graph for Coloured Petri Nets.* Theoretical Computer Science 176(1-2) , 1997.

Fei Liu, Mary Ann Blätke, Monika Heiner and Ming Yang, *Modelling and simulating reaction-diffusion systems using coloured Petri Nets.* Computers in Biology and Medicine, 2014.

Ousmane Diallo, Joel J.P.C. Rodrigues and Mbaye Sene, *Performances evaluation and Petri nets.* Modeling and Simulation of Computer Networks and Systems: Methodologies and Applications, 2015.