# Diagnosis of Technological Systems based on their Coloured Petri Net Model

**Brigitta Márczi*, Adrien Leitold,****
**Miklós Gerzson***

**Dept. of Electrical Engineering and Information Systems, University of Pannonia, Veszprém, Hungary (e-mail: gerzson@almos.uni-pannon.hu)*
***Dept. of Mathematics, University of Pannonia, Veszprém, Hungary (e-mail: leitolda@almos.uni-pannon.hu)*

**Abstract:** A novel method for discrete event systems described by Petri nets are proposed in this paper for model-based diagnosis. The model of the investigated system was defined in hierarchical colored CP-net form. Both normal reference model describing the faultless operation and the extended model containing the different faults were developed. For the fault simulation we used the arc and transition inscriptions. In order to visualize the model in different faulty modes a converter program were developed. The proposed procedure was illustrated on a simple manufacturing process with different faulty modes.

*Keywords:* discrete event systems, structure identification, graph representation, Petri nets, fault isolation.

## 1. INTRODUCTION

Model-based techniques (Blanke, Kinnaert and Lunze, 2006) are widely used and are very popular in control and diagnostic applications because of their efficiency and good performance both for systems with continuous and discrete range spaces. The appropriate models in the discrete range space case are built using the tools and techniques of discrete event systems (Cassandras and Lafortune, 1999), and these are mainly in the form of Petri nets. When used for model-based fault detection and isolation, one not only needs a model for the normal operation of the system, but also other models are required that describe the considered faulty modes. This gives the possibility to isolate the actual faulty mode using measured data and the structurally different faulty models the comparison of which is the subject of the present paper.

The idea of using model structure identification and comparison of discrete event systems models (Meda, Ramirez and Malo, 1998) is not new, but the field has matured only recently by a review paper (Fanti and Seatzu, 2008) that focuses on Petri net models used for model-based diagnosis.

As a first step, the distance of the describing Petri net models should be defined that may characterize the severity of the fault if its model is compared to the normal model. As a next step, the actual model of the real operating system determined by process mining (van der Aalst and van Hee, 2002) from the measured data are compared to the models of the different faulty modes in order to achieve fault isolation.

Starting from these results the aim of our work is to develop diagnostic methods for the following purposes:

- investigation of the models describing both the normal, faultless operational mode and different faulty operational modes;
- diagnosis of the faults during a given course of the technological system;
- analysis of the work of the investigated system based on several courses using process mining tool.

To achieve these goals several problem has to be solved. The first is to develop a suitable modeling method for the simulation both faultless and faulty operational modes. At the recent stage of our research models are used both as a reference model for normal operation and for the simulation of the real process. As a further step we plan to use data coming from the investigated system instead of simulation.

The next step is to elaborate on the method for transforming data resulting either from the simulation or form the technological system into suitable form for process mining. For this a converter software is needed which transforms the output data of the simulation software and the structure of the model into the necessary form.

The analysis of the resulted data, i.e. the diagnosis is performed both with simulation and in theoretical way. The latter method is based on the comparison of graphs coming from process mining and our goal is to develop a suitable metrics to determine the distance between them.

The aim of this paper is to present our results and illustrate them on a simple manufacturing process with different faulty modes.

## 2. BASIC NOTIONS

### *2.1 Petri net model of a process*

Petri nets enable both the mathematical and the graph representation of a discrete event system to be modeled, where the signals of the system have discrete range space and time is also discrete (Fanti and Seatzu, 2008). Petri nets can

be used for describing a controlled or open-loop system, for modeling the events occurring in it and for analyzing the resulted model. During the modeling and analysis process we can get information about the structure and dynamic behavior of the modeled system.

For the different application purposes various modifications of the original Petri net were developed, and several types of nets were introduced by researchers from all over the world since the first application of Petri nets by C. A. Petri. The aim of these modifications is to improve the modeling capabilities of this method. One of them is the *colored Petri nets* (*CP-nets*). In our paper we use the CP-nets for diagnosis purposes, i.e. for the determination of faulty operational modes of the investigated system.

### *2.2 CP-nets*

The CP-nets combine the modeling advantages coming from Petri nets and compactness of the functional programming language Standard ML (Jensen, Kristensen and Wells, 2007).

A Petri net is bipartite graph having circles and rectangles as nodes. Circles refer to the places in the net and rectangles to the transitions. Places represents the state of elements of the modeled system while transitions the actions taken place in it. There are arcs between places and transitions referring to logical relations of the system. If an arc directs from a place to a transition then the place acts as a precondition of the given transition while the arcs in the opposite direction represent consequences of transitions. Each place can be marked with one or more tokens representing the state of the modeled element.

Here we emphasize two important novelties of CP-nets only. The tokens describing the state of the system have data value attached to them. This data value is called token color and it refers to the value of the measured signal or to the value of data sets. Places, transitions and arcs can have inscriptions. An inscription of a place determines the set of color that a token on the place can have. Another place inscription gives the actual number of the tokens on the place, i.e. the current marking of that place. The inscriptions of transitions can contain different types of functions. These functions determine the type of the color set of the incoming and outgoing tokens and the operation performed on them. The arc inscriptions can be used for evaluating of the result of the performed action at the previous transition. These conditional expressions define the color of the token on the following place.

One of the main advantages of modeling with Petri nets is the ability of describing sequences of discrete events. In a real system the events occur both in a serial and in a parallel way. In case of parallelism we can distinguish two different situations. In the first case the two or more series of events can take place independently from each other. In the other case only one of the event sequences can take place because that these events exclude mutually each other. Usually it means these events have the same precondition, and the occurrence of any of them makes this precondition invalid. This kind of parallelism is called *conflict situation*. In the Petri net the conflict can be recognized when a place is the precondition of two or more transitions. In this case it is randomly selected which transition takes place. While the real parallelism can occur in normal operational circumstances, the conflicts mean fault situations in general. Although faults also have their preconditions, but these are frequently invisible for the operator and it seems to be the effect of randomness which event takes place.

Hierarchical Petri nets form a special class of CP-nets. As one can use subroutines in a main program or submodels in a complex modeling process, so subnets can be used during the model construction of a technological system in CP-net form. There is a possibility to define subnets for separate technological subprocesses at the lower level, thereafter build them together defining their relation at the higher level. This way of modeling improves the readability of the model on the main level and allows the investigation of the subnets separately.

### *2.3 Logs and traces*

An *event log* is a set of finite event sequences, whereas each event sequence corresponds to a particular materialization of the process. We refer to an event sequence as a *trace* hereafter. We assume that it is possible to record events in a way that each event refers to an activity (i.e. a well defined step in the process), and each event belongs to a case (i.e. a process instance). In addition, each event can have a performer also referred to as originator (the person who executes or initiates the activity), and events have a time stamp, while they are totally ordered.

Having an event log, well established procedures of process mining (van der Aalst and van Hee, 2002) can be used for fitting the actions taken place in the modeled system to the transitions of a CP-net. It is important to emphasize that an event log contains a set of event sequences that each corresponds to a particular behavior, and the events recorded in a log may have ”measurement errors”, that is, some of them may be omitted or have a perturbed time stamp, for example.

## 3. FAULT DIAGNOSIS USING CP-NET MODELS

Petri net models of a manufacturing process can be constructed from the a priori engineering knowledge and from measured real data. Taking the data set of real process executions, i.e. the event logs, process mining techniques can be used for process discovery (van der Aalst and van Hee, 2002). This section deals with the latter way of construction and with the comparison of the constructed Petri net models.

During the process planning one can define the set of operation leading to the technological goal of the system. Also the relation of the steps to each other and their timing is determined. One of the most important questions of the plan is to define which actions can take place in parallel way and which are the mutually exclusive events.

Another important part of the planning procedure is the exploration of possible faults. One can prepare for some of the faults but the fault diagnosis during the operation of the system has also great significance. As a result of fault handling one can avoid them or their frequency can be reduced.

*3.1 Normal reference model*

As a first step the model of investigated manufacturing system has to be defined in CP-net form and be implemented in the CPNTools. CPNTools is a software tool which allows the modeling of discrete event system in the form of colored and hierarchical Petri nets. It was developed in Aarhus, Denmark (Jensen, Kristensen and Wells, 2007; CPN Group).

If we want to investigate the operation of the system with no fault, i.e. we simulate the work of the normal reference model, then a token can be interpreted as a piece of product, and the arc and transition inscriptions have no significance.

*3.2 Describing fault modes using Petri nets*

Let us assume that we have the Petri net of the normal operational course in the form of CP-net, the so called *normal reference model*. If we know the possible faulty cases then we can integrate them into the CP-net of the system either as faulty places or using the transition and arc inscriptions. The normal reference model completed with faults is called *extended model*.

Having an event log we can establish the fired transitions in the CP-net. If we know all the steps of normal operational mode and the faulty cases then this reconstructed net should be a subnet of the extended net of the system. Comparing the reconstructed net and the net of normal operational mode the difference between them refers to the deviation from the normal operational course. Comparing the reconstructed net and the extended net of the modeled system we can establish the most likely operational course and in case of fault the most feasible reason of it can be diagnosed.

Occurring of faults can be forced with arc and transition inscriptions. The transition inscriptions contain special check functions which return fault or no fault value with a user defined probability. The arc inscriptions are conditional statements. They interpret the result of the check function of the previous transition, i.e. the color of the token and deliver the token to the next place. In case of faults causing immediate shutdown of the process separate arcs ensure the break of this simulation course and the return back to the initial state.

If we use this way of fault modeling, then there is no need for separate fault places in the net. The occurrence of the fault happens automatically and the probability function in the arc inscription controls it with a user defined probability. This solution results in a smaller, compact net but it makes its readability more difficult. This method has another disadvantage. It makes more difficult the transformation of data between the software tools using for simulation and for process mining.

*3.3 Interpretation of log files*

During the simulation the CPNTools can record the important events in a trace file, where it is the modeler's responsibility to select the registered transitions belonging to those events. An entry of the registration contains the identifier of the transition and its originator, the type of the event (normal or fault), in case of fault its identifier and the time stamp. One course of the process generates one trace file. Performing the simulation several times, separate trace files are generated and traces can be collected in one log file. Our aim is to analyze these trace and log files in order to determine the diagnostic situation of the system, i.e. whether it works under normal conditions, and determine the type and number of occurring faults.

For this analysis a set of software tools was used. The CPNTools creates the trace files with *.cpnxml extension. These trace files can be collected into a log file with *.mxml extension with the software ProMImport. The final evaluation of the logs is done with ProM data mining software (Process Mining Group).

The CPNTools stores the CP-net of the modeled system in a file having *.cpn extension. For the evaluation we have to input both the structure of the CP-net and the log file to ProM in appropriate format. To achieve this we have to convert the original cpn file into standard pnml file. We developed an appropriate software tool to perform this conversion.

The main task of our converter is to handle the inscriptions present in the original CP-net model. As it was mentioned before, we modeled the occurrence of faults by special transition and arc inscriptions. These inscriptions can be interpreted in CP-net environment only but not in ProM. Our converter program interprets these inscriptions and generates distinct fault transitions to resolve this problem. Another task of this converter is to build in the subnets into the main net. As a result of the conversion the ProM tool is able to visualize and analyze the structure of the extended net.

There are different possibilities to process the log files and net structure describing files.

1. For the visualization of the structure of the *reference model of the normal operation* the following steps are needed:

a) Convert the cpn file describing the structure of a CP-net without any log file to pnml file using our converter software.

b) Input the resulted pnml file into the ProM. The ProM visualizes the structure of the reference net referring to normal operation.

2. For the visualization of the structure of *extended model* the following steps are needed:

a) Collect into one log file the trace files of different simulation courses generated by CPNTools using the ProMImport (Process Mining Group) software.

b) Convert the cpn file describing the structure of a CP-net with the log file to pnml file using our converter software.

c) Input the resulted pnml file into the ProM. The ProM visualizes the structure of the extended net containing all transitions referring to both normal operation steps and faults.

3. The following steps are necessary for the visualization of the *model and logs*.

a) Define the probability of possible errors and do the simulation predefined times. Convert the trace files into one log file using the ProMImport.

b) Convert the structure of a CP-net using our converter software but take into account the log file during the conversion. Before the conversion we have to declare what faults have to be considered from the faults found in the logs.

c) Input the resulted pnml file to ProM. In this case the visualized net contains the transitions referring to the selected faults together with the transitions referring to the normal operation steps.

4. There is a possibility to visualize *one trace file*, i.e. one course of simulation.

a) Define the probability of errors and do the simulation only once.

b) Convert the resulted trace file into a log file using the ProMImport.

c) Convert the net structure also into pnml format with our converter program.

d) Input these files to ProM. In this case we have to match the events found in the log file to the transitions of the extended net manually. If a transition has no pair in the log it should be declared as an invisible transition. As a result of this process the ProM visualize the extended net but the transitions referring to the events found in the log have their name tag while the invisible transitions appear as a black box.

5. The software ProM provides analysis of log files, too. For this create the log file from trace files using ProMImport and input it to ProM. We can get several data about the events and originators, such as classes and types of events, the absolute and relative frequency of occurrence of events and the number of operations performed by the different originators. Performance and structural analysis can be made, too.

### *3.4 Comparison of the reference model and the reconstructed model*

1. Comparison in the space of events

Here the comparison is performed by comparing the event log generated by the reconstructed model with the one generated by the reference model using some signal norm. As the event sequences (without the timing information but with their labels as symbols) can be seen as strings, the efficient algorithms of string comparison (see e.g. Cohen, Ravikumar and Fienberg, 2003) can be applied.

2. Comparison in the space of Petri net models

Here one compares the structure of the two models by using some general graph comparison methods and related graph distance (Bunke 1997, Bunke and Shearer, 1998) based thereon. Suppose we have a model of the process in the form of a CP-net *N1*. This model is based on our original concepts about the system and on the experiences resulted from the logs of many executions of the process. Note that this model may describe both normal (i.e. non-faulty) operation courses and operation courses belonging to the known faulty modes. Based on the workflow log of the actual operational courses and using some mining algorithm we construct another CP-net *N2*. The question is whether *N2* is a subgraph of *N1*. If it is true then we can determine whether the system works under normal operational conditions or we can isolate the fault. If it is not true then probably a new fault has been detected.

## 4. SIMPLE CASE STUDY

The aim of this section is to show the modeling process, the steps of log processing and the graph comparison using a simple illustrative case study. The investigated system is a manufacturing process containing serial and parallel actions and it has different faulty possibilities.

### *4.1 The manufacturing system and its operating procedure*

Our hypothetic manufacturing system contains four processes. Each process is set of coherent actions. Process 1 and Process 2 can take place simultaneously while Processes 3 and 4 are sequential, they can start if the previous process or processes are ready. The actions of a process take place in sequential way. We assume that one operational course of the system refer to manufacturing of the product.

According to the description of the system the events of the manufacturing operation are organized into processes. These processes were described in separate subnets. In this way we get a hierarchical Petri net. The upper level of CP-net model consists of these processes and some other auxiliary nodes which are necessary for the simulation and error handling. The lower levels describe the structure of processes. Each subnet contains set of actions and their perquisites and consequences. Transitions of the net refer to the actions of the process while places refer to their pre and post conditions.

Three types of faults are to be considered. The faults belonging to the first group are the most serious cases, if they occur the course of operation is immediately over. In this case the system gets back into its initial state and a new course can start. In this simple model we do not handle the pieces of the stopped course. In the second case the effect of the fault is smaller, all the process steps take place but the

product may have quality problem. In the third case the fault has minor effect only.

The main net of CP-net model of the system can be seen in Fig. 1 Rectangles with double line are referring to transitions containing subnets. The subnet of Process 1 is depicted in Fig. 2.

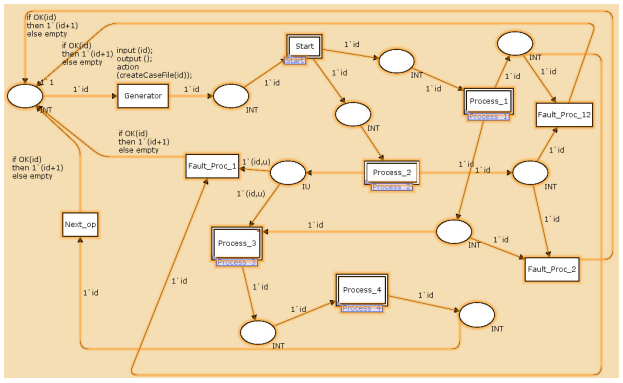


Fig. 1. The main net of the model

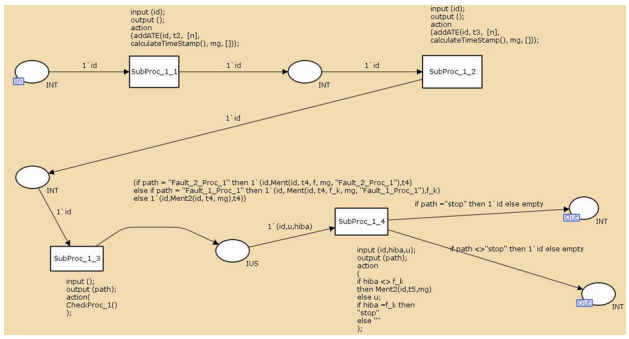


Fig. 2. The subnet of Process 1

As it was mentioned we use the transition inscriptions for generating the fault randomly. The check functions built into the transition inscriptions return with fault in predefined probability. In this way two different kinds of faults with different probability can be handled at the same transition. Fig. 3 shows an example for fault probability definition.

Applying our converter software the structure of the hierarchical colored Petri net depicted partly in Figs. 1 and 2 can be converted into low level net which is suitable for visualizing in ProM. This low level net of the reference model can be seen in Fig. 4.

```
Ellenorzesek
  fun OK(id) = if id < 1 then true else false;
  fun CheckProc_1() = let val p=discrete(0,99);
    in if p<20 then "Fault_1_Proc_1"
    else if p<40 then "Fault_2_Proc_1"
    else "ok" end;
  fun CheckProc_2_1() = let val p=discrete(0,99);
    in if p<30 then "Fault_1_Proc_2"
    else "ok" end;
  fun CheckProc_2_2() = let val p=discrete(0,99);
    in if p<0 then "Fault_2_Proc_2"
    else "ok" end;
```

Fig. 3. Fault probability definition

*4.2 Fault diagnosis investigations*

As it was mentioned earlier, different types of fault possibilities have been built into the model. Our aim was to investigate the diagnosability of these faults comparing the structure of the nets. For this we generated trace files containing one or two predefined faults. Three different cases have been investigated:

- operation containing stop error in earlier phase of the course;
- operation containing a minor fault;
- operation containing two faults, a minor and a stop error close to end of the course.

The certain occurring of faults was forced by giving the maximal probability value to each fault. The resulted Petri nets belonging to the different cases can be seen in Figs. 5-7.

In the first case the stop error faults occur in Process 2. In this situation - although Process 1 could terminate in normal way - but Process 2 and thus the complete operation ended immediately after fault F2 has occurred.

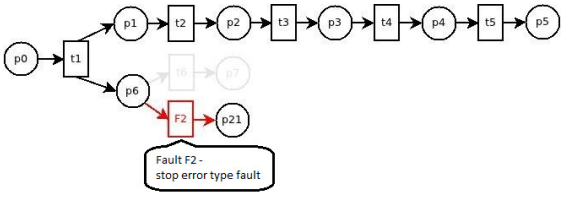


Fig. 5. A visualized trace when there is a stop error in the beginning of the process

In the second case a minor fault occurs during the operation only. Instead of transition *t*8 the fault F3 takes place. Because this fault modifies slightly the quality of the product the operation can go further, and it terminates in a normal way. Comparing Fig. 4 and Fig. 6, the difference between the two graphs is minimal. It is easy to show that we get the same result in case of any other minor faults.

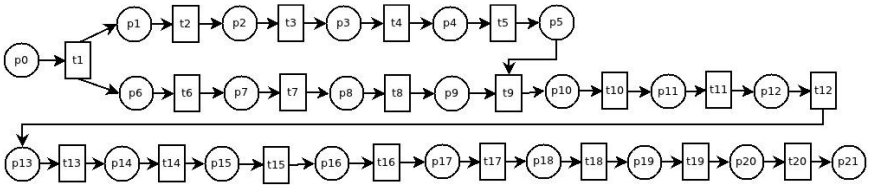


Fig. 4. The normal reference model visualized in ProM

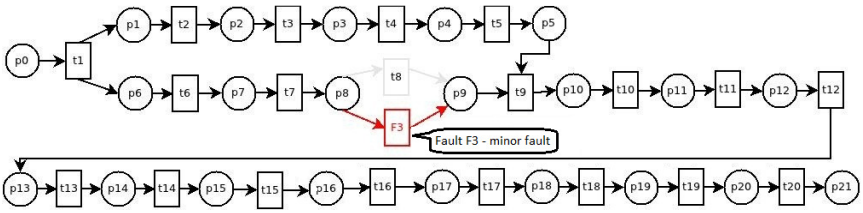


Fig. 6 A visualized trace when there is a small error in the process

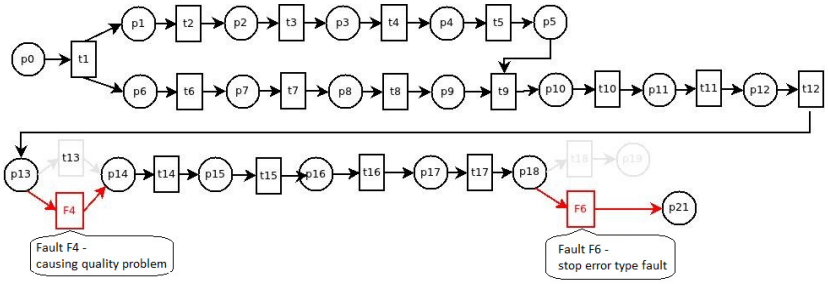


Fig. 7: A visualized trace when there is a stop error in the end of the process

In the third case a minor fault (F4) occurs during and operation and after that, as a consequence of a stop error (F6) the operation ends immediately. The graph of this situation can be seen in Fig. 7.

## 5. CONCLUSIONS AND FUTURE PLANS

A novel method for discrete event systems described by Petri nets are proposed in this paper for model-based diagnostic purposes.

The model of the investigated system was defined in hierarchical colored CP-net form. Both normal reference model describing the faultless operation and the extended model containing the different faults were developed. For the fault simulation we used the arc and transition inscriptions.

The visualization of net was performed by ProM. In order to get the files having the appropriate form and extension we developed a converter program.

The proposed procedure was illustrated on a simple manufacturing process with three faulty modes.

As a following step we plan to elaborate on a graph comparison method in order to determine the distance between the CP-net models of the normal and faulty operation.

## ACKNOWLEDGEMENT

Authors acknowledge the financial support of this work by the Hungarian State and the European Union under the TAMOP-4.2.1/B-09/1/ KONV-2010-0003 project “Mobility and environment: Researches in the fields of motor vehicle industry, energetic and environment in the Middle- and West-Transdanubian Region”. Part of the work is also supported by the Hungarian National Research Fund through project K83440.

## REFERENCES

Blanke, M., Kinnaert, M., Lunze, J., Staroswiecki, M. (2006). *Diagnosis and Fault-tolerant Control*, Springer-Verlag, Berlin.

Bunke H. (1997). On a relation between graphs edit distance and maximum common subgraph, *Pattern Recognition Letters* 18, pp. 689-694.

Bunke, H., Shearer, K. (1998). A graph distance metric based on the maximal common subgraph, *Pattern Recognition Letters* 19, pp. 255-259

Cassandras, C.G., Lafortune S. (1999). *Introduction to Discrete Event Systems*, Kluwer Academic Publishers,

Cohen, W., Ravikumar, P., Fienberg, S. (2003). A comparison of string distance metrics for name-matching tasks, *Proceeding of the IJCAI*, 2003.

CPN Group, University of Aarhus, Denmark: CPNTools 2.2.0 http://wiki.daimi.au.dk/cpntools/

Fanti M.P., Seatzu C. (2008). Fault diagnosis and identification of discrete event systems using Petri nets, *9th International Workshop on Discrete Event Systems, WODES 2008*, pp. 432–435.

Jensen, K., Kristensen, L.M., Wells L. (2007). Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems, *Int. J. of Software Tools for Technology Transfer*, 9. 3-4, pp. 213-254

Meda, M.E., Ramirez, A., Malo A. (1998). Identification in discrete event systems, *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 740–745.

Process Mining Group, Eindhoven Technical University, The Netherlands: ProM 5.2. and ProMimport, http://www.promtools.org/prom5/ and http://www.promtools.org/promimport/

van der Aalst W.M.P., van Hee K.M. (2002). *Workflow Management: Models, Methods and Systems*, MIT Press, Cambridge, MA.