HUNGARIAN JOURNAL
OF INDUSTRY AND CHEMISTRY
VESZPRÉM
Vol. 41(1) pp. 17–26 (2013)

# MONITORING AND DIAGNOSIS OF MANUFACTURING SYSTEMS USING TIMED COLOURED PETRI NETS

ADRIEN LEITOLD[1], BRIGITTA MÁRCZI[2], ANNA IBOLYA PÓZNA[2], AND MIKLÓS GERZSON[2]✉

[1] Department of Mathematics, University of Pannonia, Egyetem str. 10, HUNGARY
[2] Department of Electrical Engineering and Information Systems, University of Pannonia, Egyetem str. 10, HUNGARY
✉E-mail: gerzson@almos.uni-pannon.hu

Novel fault modelling and integration method were applied in the case when the faultless operation of the system was modelled by a high-level, coloured Petri net. In order to achieve realistic investigations, a timed coloured Petri net model of the system was constructed, where faults can occur in the manufacturing lines. The faultless and fault containing models were implemented in CPNTools both for non-timed and timed cases. The resulted model was investigated both via simulation and using the occurrence graph. For efficient analysis of the occurrence graph a software module called OGAnalyser was developed.

**Keywords:** monitoring and diagnostics of manufacturing processes, coloured timed Petri nets, probabilistic models, occurrence graph

## Introduction

Models are often used for the description and investigation of complex systems even if they cannot perfectly describe the investigated system. The course of a manufacturing system can be split up to distinct steps of serial or parallel technological sub-processes. This enables the description of the system by a discrete event systems model [1] in the form of Petri nets. During the design process of a manufacturing system, often only a model of the faultless operation is created. The integration of possible faults into the model could give important information for more complex investigation of the system. In our previous work [2] we have integrated fault events with different occurring possibilities into low-level Petri net models of manufacturing systems in such a way that the size of the model remained almost the same.

In our recent work, we applied the above fault integration method for the case when the faultless operation of the system was modelled by a high-level, coloured Petri net (abbreviated as CP-nets) [3]. In low-level Petri nets the transitions fire instantaneously, but the events of a real system take place for a certain amount of time influencing the operation of the system. Therefore, a timed coloured Petri net model of the system is constructed in order to achieve realistic investigations.

CPNTools [4] offer tools for modelling and analysing of CP-nets. There are two possibilities for the investigation of a manufacturing system in CPNTools: the simulation and the analysis of the occurrence graph. In case of fault modelling using different occurring possibilities; however, the standard occurrence graph does not give information about the probability of the different occurring states of the system. Therefore, a special software module, called the OGAnalyzer has been developed for solving this problem. Weights can be assigned to the arcs of the occurrence graph and the software calculates the probability of each node in the occurrence graph, i.e. of each operational state of the system.

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

Petri nets enable both the mathematical and the graph representation of a discrete event system to be modelled, where the signals of the system have discrete range space and time is also discrete [5]. Petri nets can be used for describing a controlled or open loop system for modelling the events occurring in it, and for analysing the resulted model. For the different application purposes, various modifications of the original Petri net were developed with the aim of improving the modelling capabilities. One of the approaches is the *coloured Petri nets* (CP-nets). We use here the CP-nets for modelling technological systems and their diagnosis, i.e. for the determination of faulty operational modes of the investigated system.

### *Coloured Petri nets*

The CP-nets combine the modelling advantages of Petri nets and the compactness of the functional

programming language Standard ML [6]. 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 represent the state of the elements in the modelled system, while transitions correspond to 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 coloured tokens representing the state of the modelled element.

Here we emphasize two important novelties of CP-nets only. The tokens describing the state of the system have data value, the so-called *token colour* attached to them. In this paper these colours are used to identify workpieces and to describe the operation to be performed on them. Places, transitions and arcs can have 'inscriptions'. An inscription of a place determines the set of colours 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 colour set of the incoming and outgoing tokens and the operation performed on them. The arc inscriptions can be used for evaluating the result of the performed action at the previous transition. These conditional expressions define the colour of the token on the following place.

The original Petri net concept did not contain the time; however, the firing of transition takes place instantaneously. In case of real technological systems the time has a great role during the occurring of events. CP-nets also offer the possibility of adding time to the operation of transitions. The firing rule of a transition in *timed Petri nets* is as follows: a transition is enabled if all of its input places contain sufficient number coloured tokens defined by the arc functions and the allotted time has elapsed.

One of the main advantages of modelling with Petri nets is the ability of describing sequences of discrete events that 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 of each other. This situation occurs when workpieces can be elaborated in different manufacturing lines in parallel way. In the other case only one of the event sequences can take place because these events exclude mutually each other. 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 a 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. The conflict can have two different sources in technological systems. A conflict occurs in a technological sense when two or more processes want to use the same tool or resource e.g. a robot. Usually it is worth to assign priority to each conflicting transition in order to define their sequence.

When a fault occurs during the operation of the system, it also causes a conflict situation. This can be avoided by adding a special probability function to arc expression functions. By evaluating this function, the occurrence of the fault can be unambiguously determined during execution.

### *Analysis of Petri nets*

There are two basic directions for the analysis of a Petri net: (i) the structural method, which is independent of the initial state of the net and (ii) the investigations based on a given initial state (the behavioural analysis). In this paper the latter is used for the investigation of technological system behaviour.

Simulation is our primary tool for the checking the correctness of a model. Starting from a given initial state the user can check whether the operation of the system terminates in the appropriate state. It can also be investigated, which transitions become enabled in certain steps, whether there is a conflict among then. Simulation investigations do not give unambiguous answers to questions of formal analysis formulating in Petri net literature [7] but they complement them well.

Another Petri net analysis method uses the 'occurrence graph'. The basic idea of the occurrence graph is to construct a graph, which contains all of the reachable markings from a given initial state. These marking are the nodes of the occurrence graph and the arcs connecting the nodes refer to the logical relations realized by the firing transition between two markings. Unfortunately, the occurrence graph even of a small Petri net may become very large. Therefore, several reduction methods were proposed in order to get a relatively small occurrence graph [7]. Most of the simulation tools, as the CPNTools [4] used by us, are able to construct the occurrence graph.

## Modelling and analysis of technological systems

In the following, we wish to demonstrate the use of CP-nets and their occurrence graphs for modelling and analysis of technological processes. Both for normal faultless and faulty mode operation of the technological system are considered in the non-timed and timed cases. The analysis is performed via simulation and with different investigations of the occurrence graph.

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

A simple case study is presented here for a manufacturing system containing two manufacturing lines and a robot. The arrangement of the system can be seen in *Fig.1*.

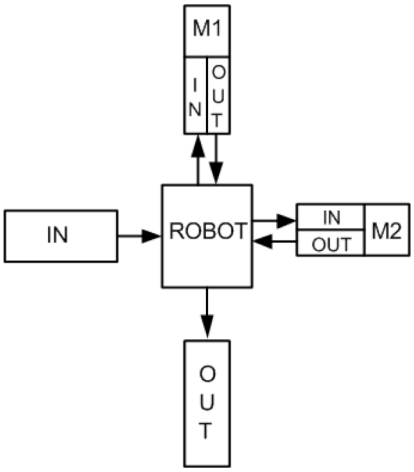


*Figure 1:* The manufacturing system

The workpieces to be processed appear on the input place IN. The task of the robot is to put them to the appropriate input place of a manufacturing line `M1_IN` or `M2_IN` according to operational instructions. Assume that the two manufacturing lines perform different actions on the workpiece of interest. When the manufacturing process is over, the finished workpiece appears at the output end of the line either on `M1_OUT` or `M2_OUT` depending on the performed action. If the workpiece has to be modified on the other manufacturing line then the robot puts it onto the other input place. If the manufacturing process is over then the robot puts it into the product container `OUT`. Assume that one workpiece at a time can be on the input and output places of manufacturing lines. It follows that the robot can only transfer a workpiece from place IN to the input place of a manufacturing line if this place is empty and the precondition of the start of a manufacturing process is that the output place of this line should be empty. As a general precondition of all transfer processes the robot has to be free.

The CP-net model of the normal (faultless) operation of the manufacturing system in the form of a screenshot from CPNTools can be seen in *Fig.2*. The process starts with a token at place `START`. The transition `generator` generates the prescribed number of tokens representing workpieces at place `IN`. The colour assigned to a token contains an identifier of the workpiece and a code referring to the manufacturing process or processes to be carried out. Four kinds of manufacturing mode are possible in this manufacturing system: the workpiece has to be processed on line 1 (denoted by `m1`) or on line 2 (`m2`) only, or it has to go through the line 1 and then through line 2 (`m12`) or in reverse order (`m21`). As an example, the token (`1,m12`) refers to the workpiece having identifier '`1`' and this piece has to be processed first on line 1 then on line 2. The state of input and output of manufacturing lines is modelled with two places. The places `Tin_empty`, `Tin_full` refer to state of inputs and they are mutually exclusive. The colour of tokens referring to the state of these places consists of the identifier of the line only. If the input place of the line 1 is empty then there is a token having colour `m1` on the place `Tin_empty`, and there has not to be a token having colour `m1` on the place `Tin_full`. The same applies for the places (`Tout_empty` and `Tout_full`) describing the state of the output places of manufacturing lines.

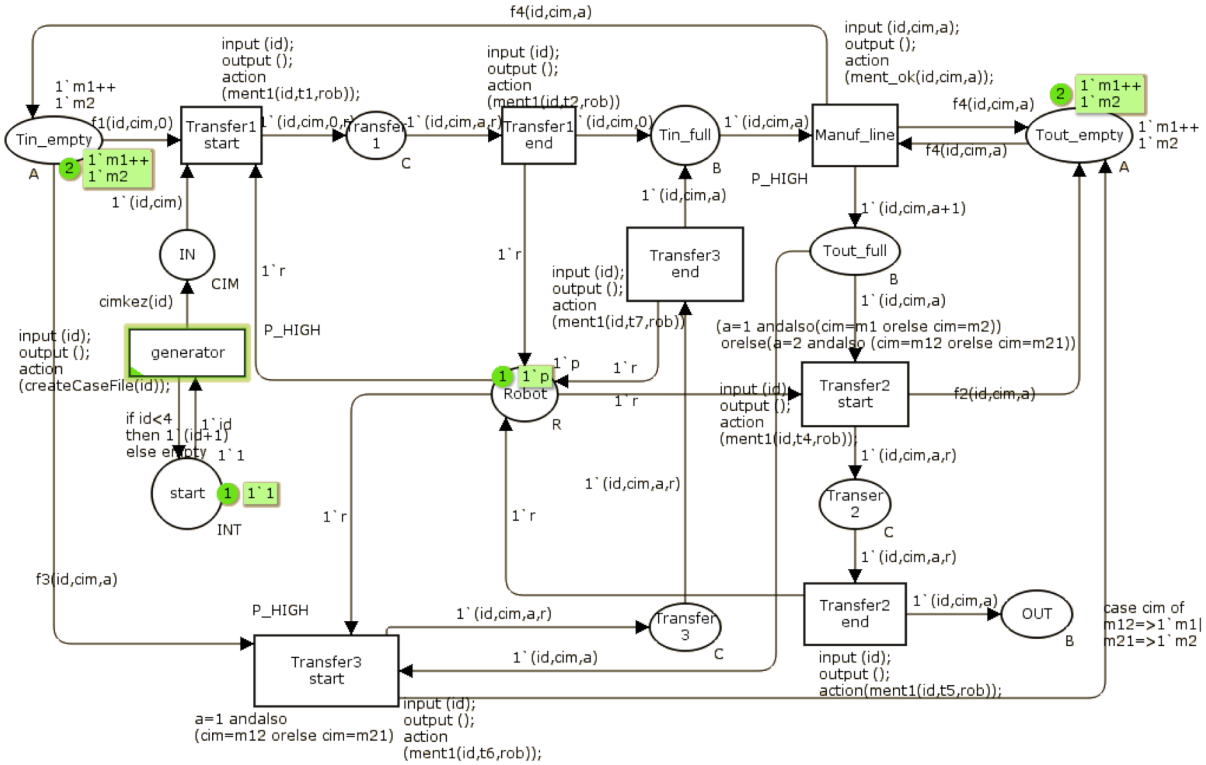


*Figure 2:* The CP-net model of the faultless operation of the manufacturing system

The transfer processes are disintegrated into three steps: to transitions referring to the (i) start and (ii) end of the transfer and to a place representing the (ii) transfer itself. Transitions `Transfer#_start` correspond to the start of transfer processes from `IN`, if `# = 1`, from `Tout_full` if `# = 2` or 3. The places `Transfer#` represent the transfer process 1, 2 or 3. The transitions `Transfer#_end` refer to completing of transfer process to the input place of a line (`# = 1`), to product container (`# = 2`) or to the input place to another manufacturing line (`# = 3`). The place `Manuf_line` refers to two manufacturing lines and the colour of token shows the line being processed.

Assume that only a single fault can occur in the system during manufacturing: the identification label of the piece can get damaged therefore it cannot be identified. Workpieces with damaged label get into a separate container represented by place `OUT_Fault`. The repairing of the label is not handled in this example. The modified part of the Petri net model can be seen in *Fig.3,* where the occurring of fault is taken into account. The occurrence of the fault is forced by a check function built into arc inscriptions in the Petri net model randomly. This check function returns with a fault in predefined probability. This probability of the can be set in the definition part of the net and different fault probability values can be assigned to the two manufacturing lines.

By comparing *Figs.2* and *3*, it can be stated that a new place `OUT_fault` appears as a consequence of fault modelling and integration and the arc expression functions of arcs starting from transition `Manuf_line` are extended with the fault checking.

For realistic investigation of a technological system the timed version of the CP-net model is used as a case study. A time point is assigned to the transitions. The transition `generator` does not belong to the technological system closely, so it fires under zero time, i.e. instantaneously. Different time units are assigned to the other transitions. These time values appear as transition inscriptions '`@+i`' (where `i` is an integer number defining the amount of time in seconds) on the net as it can be seen in the *Fig. 3*. During the simulation investigations different time values have been applied in order to check the possibility of a deadlock.

*Preliminary analysis by simulation*

As a first step, simulation is applied for the investigation of the developed Petri net model of the manufacturing system. These were carried out assuming both faultless system operation and when fault can occur during the manufacturing. Both non-timed and timed operational modes were considered.

The short description of the simulation is as follows. The simulation starts with the generation of tokens representing workpieces. The number of these tokens i.e. the number of workpieces to be processed can be modified in the arc expression function belonging to the transition `generator`. The type of processes to be

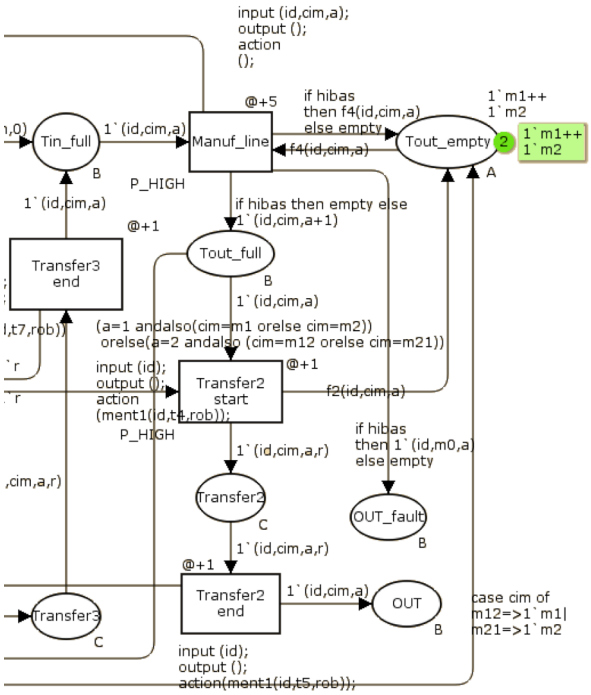


*Figure 3*: The Petri net model after integration of fault

performed i.e. the label referring to the manufacturing line(s) of the tokens can be set in the definition part of the CP-net. As a result, the colour of generated tokens refers to the identifier and to the process(es) to be carried out. Because of the highest priority of transition `generator` the transition `Transfer1_start` can only fire after the prescribed number of token appears at the place `IN`. As mentioned above, the transition `Transfer1_end` refers to the completion of the transfer of the workpiece to the input place of a manufacturing line. As a next step, the firing of transition `Manuf_line` corresponds to the completion of manufacturing process. The only fault in the system can occur during this process. If it happens then the token gets into the place `OUT_fault`, which represents the fault container. If the fault does not occur then the system checks whether the manufacturing process is over or the workpiece has to be also processed on the other line. In the first case the robot puts it to the place `OUT` representing the product container, while in other case it transfers the token representing the piece into the input place of another line. The modelling of the other two transfer processes (transfer of a workpiece from the output place of a manufacturing line either to the product container or to the input place of the other line) is similar to the transfer from place `IN` to place `Tin_full`. If the appropriate input place and the robot are free, another transfer process can start.

The primary goal of the simulation is to check the correct operation of the model. Another aim is the investigation of possible deadlock situations. In case of deadlock, the process stops without completing all the prescribed technological actions. It can happen in case of timed simulation of our investigated system and it refers to the wrong determination of the timing of actions.

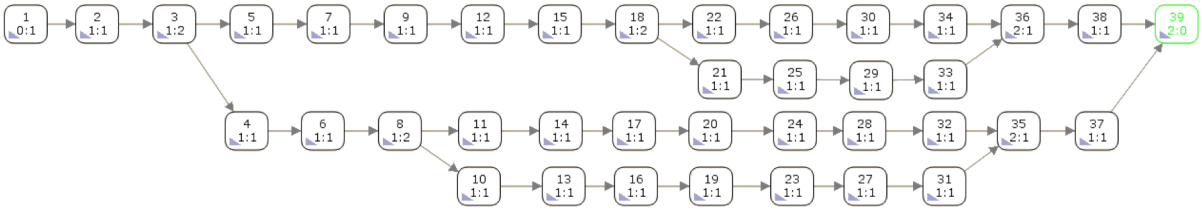

*Figure 4:* The occurrence graph - no fault, no timing

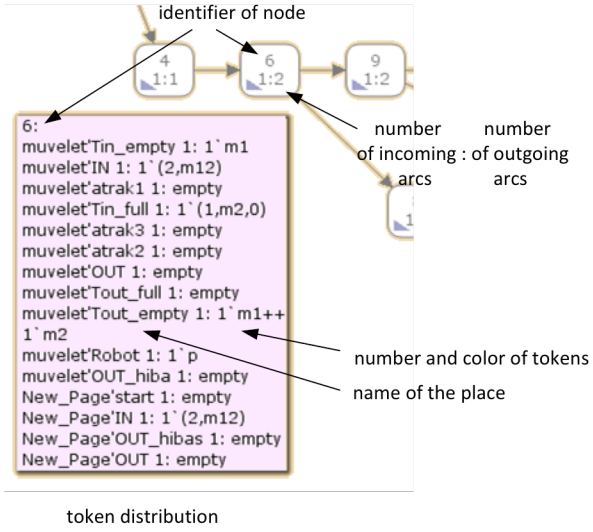


*Figure 5:* Key for the occurrence graph

Simulation cannot ensure the thorough investigation of the modelled system, but it complements well the further analysis. For the detailed investigations we applied the analysis of occurrence graph.

*Analysis based on the occurrence graph*

The thorough analysis of the behavioural properties of a CP-net can be performed using its occurrence graph. The concept of the occurrence graph was introduced above.

The CPNTools generates automatically the occurrence graph, but the check functions used for the fault generation have to be removed from the arc expressions otherwise the occurrence graph is generated only for the normal mode or for the faulty mode.

For the illustration of generation and analysis of the occurrence graph let us assume that there are two workpieces to be processed, one of them has to go through manufacturing line 2, while the other has to go first through manufacturing line 1 then through line 2. Let the operation of the system be faultless and let the firing of all transition be instantaneous, i.e. the net is non-timed. The resulted occurrence graph can be seen in *Fig.4*. The explanation of numbers in the occurrence graph can be seen in *Fig.5*. The frames in the upper part of boxes are the identifiers of system states. The expression *x:y* in lower part shows the number of preceding and succeeding states. The token distribution belonging to a node can be obtained by selecting the triangle in left low corner. The opening window contains the name of places and the number and colour of tokens.

All the branches on the graph are explained by technological reasons, as there is no built in rule for the robot to start the transfer with any particular workpiece. The graph has only one terminal node (highlighted by green) and it refers to the normal termination of the process i.e. all the prescribed manufacturing processes terminated properly.

Now we can repeat the simulation with the same initial condition relating to workpieces but assuming that faults can occur during manufacturing process. The resulted occurrence graph can be seen in *Fig.6*.

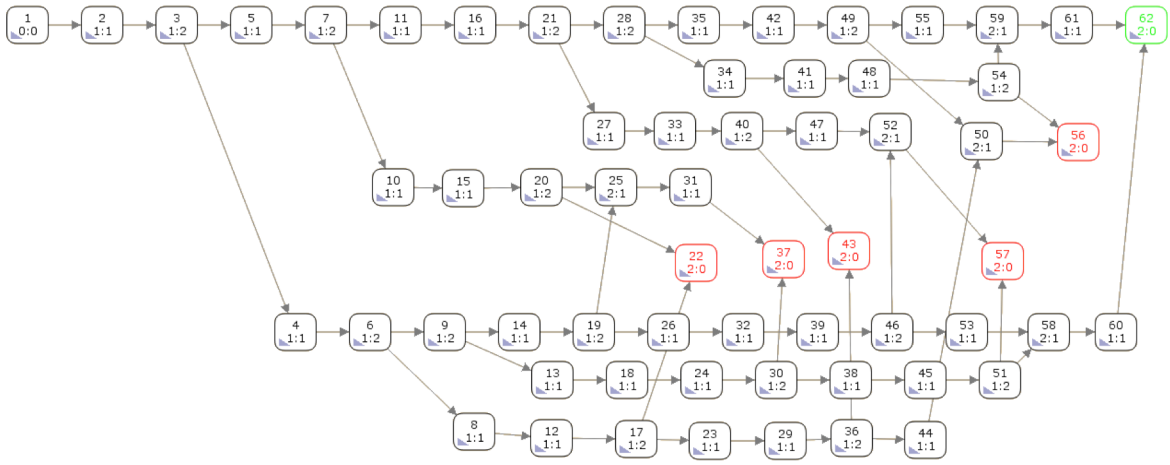

*Figure 6:* The occurrence graph - with fault, no timing

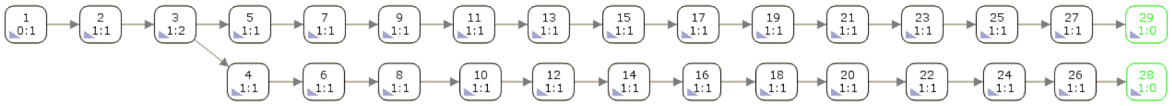

*Figure 7:* The occurrence graph - no fault, with timing

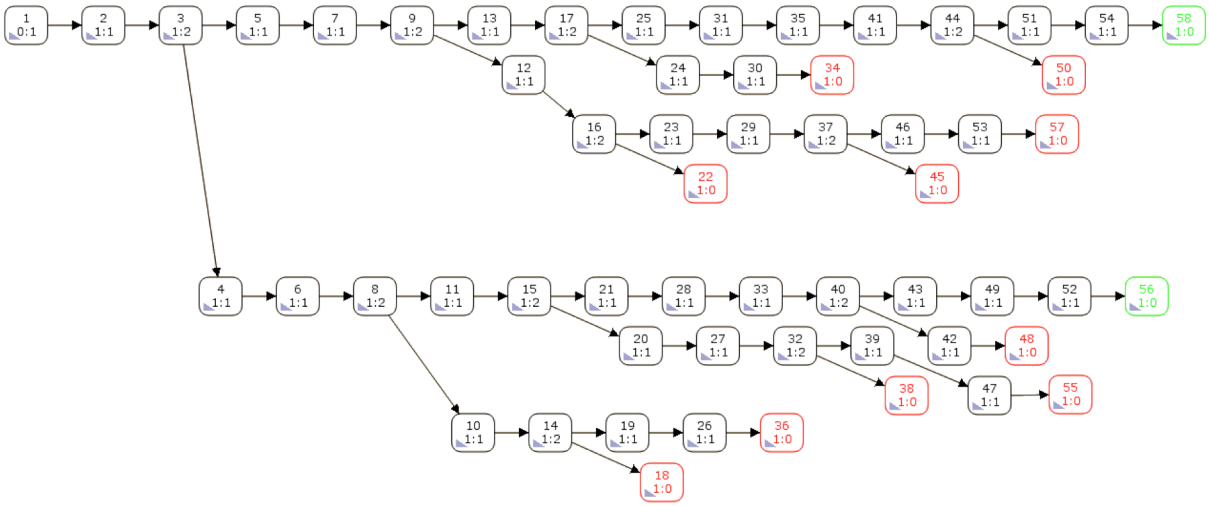

*Figure 8:* The occurrence graph - with fault and timing

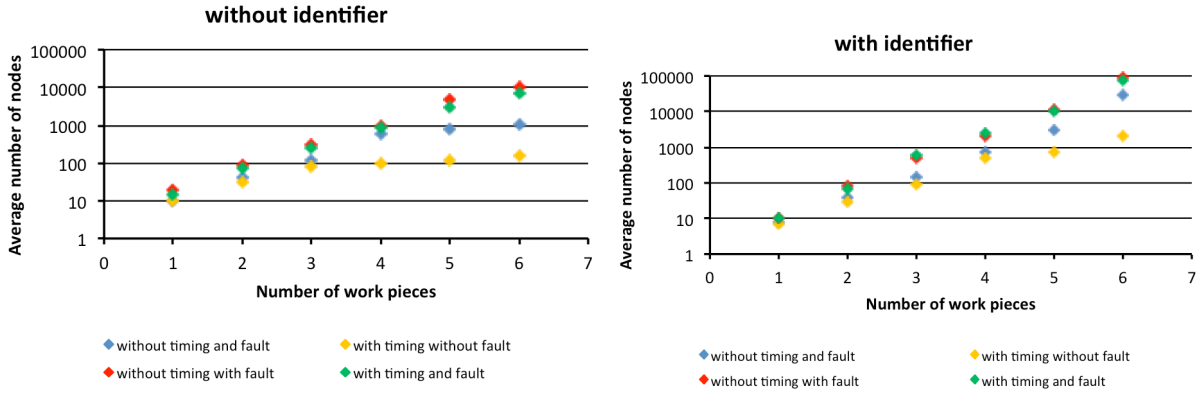


*Figure 9:* The node numbers in occurrence graph

It can be stated that the occurrence graph has become larger and the number of terminal nodes has increased due the effect of the possible faults. The reason of branches on the graph can be either technological (at the nodes 3, 9, 28, 51 and 54) or due to the fault. Only one of the terminal nodes refers to the normal termination of the process (highlighted by green, node 62), the others belong to the faulty cases (highlighted by red, nodes 22, 37, 43, 56 and 57). In case of faulty operation the identification tag of at least one workpiece get damaged during the manufacturing process.

Next we repeat the simulation again but adding the timing information to the net and assuming faultless operation. Let the time duration of transfer processes be equal to unit time, and the time of manufacturing processes is equal to 5 time units. The resulted occurrence graph can be seen in *Fig.7*.

The resulted occurrence graph is a tree and its two terminal nodes (nodes 28 and 29) differ only in time stamp of the tokens (the time stamp can be seen only in CPNTools with a detailed label for each node). Comparing this graph with the occurrence graph in *Fig.4*, it can be stated that the number of parallel branches is less and they have different terminal states. Adding timing information to the model reduces the number of the possible different technological variants.

As the fourth case, let the simulation be performed with both timing and faults. The resulted occurrence graph (see *Fig.8*) is a tree again, the number of nodes is almost the same but the number of terminal nodes has doubled comparing to the occurrence graph in *Fig.6*.

It can be stated based on *Figs.4* and *6–8* that the occurrence graph is relatively simple in case of small number of workpieces and the analysis of nodes can be done manually. Thus, it is easily to find the terminal node or nodes referring to normal, faultless termination of the process, and those terminal nodes where the manufacturing of one piece or of both pieces ends with fault. However, the size of the occurrence graph grows

*Table 1:* Comparing the structure of occurrence graphs – no timing

| No. of work-pieces | label (identifier, manufacturing information) | faultless operational mode | | | faulty operational mode | | |
|---|---|---|---|---|---|---|---|
| | | nodes | arcs | TN[a] | nodes | arcs | TN[a] |
| 1 | (1,m1) | 7 | 6 | 1 | 8 | 7 | 2 |
| 1 | (1,m12) | 10 | 9 | 1 | 12 | 11 | 3 |
| 2 | (1,m1)++(2,m2) | 29 | 32 | 1 | 42 | 50 | 4 |
| 2 | (1,m1)++(2,m12) | 37 | 40 | 1 | 60 | 70 | 6 |
| 2 | (1,m2)++(2,m12) | 39 | 41 | 1 | 62 | 72 | 6 |
| 2 | (1,m12)++(2,m21) | 42 | 44 | 1 | 82 | 95 | 9 |
| 3 | (1,m1)++(2,m2)++(3,m12) | 145 | 173 | 1 | 306 | 409 | 12 |
| 3 | (1,m2)++(2,m12)++(3,m21) | 165 | 189 | 1 | 414 | 536 | 18 |
| 3 | (1,m1)++(2,m1)++(3,m1) | 107 | 123 | 1 | 210 | 273 | 8 |
| 4 | (1,m1)++(2,m2)++(3,m12)++(4,m21) | 631 | 772 | 1 | 2063 | 2900 | 36 |
| 4 | (1,m1)++(2,m1)++(3,m1)++(4,m1) | 340 | 404 | 1 | 949 | 1300 | 16 |
| 5 | (1,m2)++(2,m12)++(3,m21)++(4,m1)++(5,m2) | 2208 | 2797 | 1 | 9813 | 14420 | 72 |
| 5 | (1,m21)++(2,m12)++(3,m21)++(4,m12)++(5,m21) | 3259 | 3839 | 1 | 23709 | 32939 | 243 |
| 5 | (1,m2)++(2,m1)++(3,m2)++(4,m1)++(5,m2) | 1621 | 2165 | 1 | 5252 | 8051 | 32 |
| 6 | (1,m2)++(2,m12)++(3,m21)++(4,m1)++(5,m2)++(6,m12) | 8698 | 10980 | 1 | 62751 | 93323 | 216 |
| 6 | (1,m21)++(2,m12)++(3,m21)++(4,m12)++(5,m21)++(6,m12) | 12184 | 14562 | 1 | 146992 | 208602 | 729 |
| 6 | (1,m2)++(2,m1)++(3,m2)++(4,m1)++(5,m2)++(6,m1) | 5638 | 7686 | 1 | 21630 | 38886 | 64 |

[a] terminal nodes

exponentially if the number of pieces becomes larger, as it can be seen in the left part of *Fig.9* assuming faultless or faulty operational mode, and non-timed or timed net.

In case of large number of simpler workpieces when the identification is not necessary for each item, the identification tag can be omitted from the colour of the token. Assuming this situation all the simulation investigations (with and without fault, with and without timing) is repeated. The size of the resulted occurrence graphs is less than an order of magnitude simpler compared to the equivalent case with identifier as it can be seen in the right part of *Fig.9*. Since the workpieces have to be processed on the same manufacturing line, they have the same colour, thus they are indistinguishable. Since the robot selects among workpieces having the same colour, there will be no technological branches on the occurrence graph. This results in a much simpler occurrence graph especially when there are large number of workpieces and the processes to be performed is one or two kind of sorts.

As an example, let us investigate the structure of occurrence graphs in case when there is no timing in the system, the token colour contains the identification tag and both faultless and faulty operational mode is assumed. According to data in *Table 1,* the size of the occurrence graph depends on the number of workpieces, the number of manufacturing procedures to be processed, and the presence of fault. If fault can occur during the manufacturing process both the number of nodes and the arcs increases dramatically. There are several terminal nodes, too, but only one refers to normal termination of manufacturing processes the others belong to different faulty situations. The labelling of workpieces has also a significant effect on the complexity of the occurrence graph. If there are one or more workpieces, which should be processed on both manufacturing lines then the number of nodes and arcs is doubled as it can be seen in the corresponding rows in *Table 1*.

The results of an investigation with timing information can be seen in *Table 2*. If the number of workpieces is less than four or the workpieces have to be processed on only one manufacturing line, then the size of the occurrence graph depends on the number of workpieces and the presence of fault. On the other hand if there are at least four workpieces and at least two of them have to be processed on both manufacturing lines, but in reverse order then a deadlock situation can occur. A further condition of a deadlock that the manufacturing time should be longer than the transfer time but it is true in general.

In case of a deadlock, the process stops because the precondition of transfer processes cannot be fulfilled. There are workpieces on input and output places of both manufacturing lines and therefore no further steps are enabled. If all the workpieces have to be processed on both manufacturing lines and there is no fault then all of terminal nodes refer to a deadlock as it can be seen in the rows marked by a footnote in the *Table 2*.

The identification tag can be omitted from the colour of token in certain cases. If the number of workpieces is small and they have to be processed in different ways then there is a significant change in the occurrence graph. On the other hand in case of large number of workpieces, the structure of occurrence graph becomes much simpler if the identification tag is removed from the colour.

*Analysis of the occurrence graph using the OGAnalyzer*

As mentioned above, CPNTools cannot use the information about the probability of faults at the generation of occurrence graph. However, assigning this value to the appropriate edges, the probability of each node of the occurrence graph, i.e. of each system state can be determined. For this purpose, software called

*Table 2:* Comparing the structure of occurrence graphs – with timing

| No. of work-pieces | label (identifier, manufacturing information) | faultless operational mode | | | faulty operational mode | | |
|---|---|---|---|---|---|---|---|
| | | nodes | arcs | TN/D[a] | nodes | arcs | TN/D[a] |
| 1 | (1,m1) | 7 | 6 | 1/0 | 8 | 7 | 2/0 |
| 1 | (1,m12) | 10 | 9 | 1/0 | 12 | 11 | 3/0 |
| 2 | (1,m1)++(2,m2) | 23 | 22 | 2/0 | 41 | 40 | 8/0 |
| 2 | (1,m1)++(2,m12) | 29 | 28 | 2/0 | 59 | 58 | 12/0 |
| 2 | (1,m2)++(2,m12) | 29 | 28 | 2/0 | 58 | 57 | 12/0 |
| 2 | (1,m12)++(2,m21) | 35 | 34 | 2/0 | 87 | 86 | 18/0 |
| 3 | (1,m1)++(2,m2)++(3,m12) | 73 | 74 | 4/0 | 264 | 263 | 48/0 |
| 3 | (1,m2)++(2,m12)++(3,m21) | 86 | 85 | 4/0 | 392 | 391 | 72/0 |
| 3 | (1,m1)++(2,m1)++(3,m1) | 85 | 84 | 6/0 | 262 | 261 | 48/0 |
| 4 | (1,m1)++(2,m2)++(3,m12)++(4,m21) | 173 | 172 | 8/2 | 1473 | 1472 | 258/2 |
| 4 | (1,m1)++(2,m1)++(3,m1)++(4,m1) | 365 | 364 | 24/0 | 2121 | 2120 | 384/0 |
| 5 | (1,m2)++(2,m12)++(3,m21)++(4,m1)++(5,m2) | 574 | 573 | 24/4 | 9320 | 9375 | 1582/8 |
| 5[b] | (1,m21)++(2,m12)++(3,m21)++(4,m12)++(5,m21) | 154 | 153 | 24/24 | 12076 | 12075 | 2196/60 |
| 5 | (1,m2)++(2,m1)++(3,m2)++(4,m1)++(5,m2) | 514 | 513 | 24/0 | 5152 | 5247 | 780/0 |
| 6 | (1,m2)++(2,m12)++(3,m21)++(4,m1)++(5,m2)++(6,m12) | 1725 | 1732 | 80/40 | 71817 | 72336 | 12300/116 |
| 6[b] | (1,m21)++(2,m12)++(3,m21)++(4,m12)++(5,m21)++(6,m12) | 325 | 324 | 72/72 | 45541 | 45540 | 8280/432 |
| 6 | (1,m2)++(2,m1)++(3,m2)++(4,m1)++(5,m2)++(6,m1) | 2341 | 2412 | 72/0 | 39349 | 41076 | 5760/0 |

[a] terminal nodes/deadlocks, [b] labels where all terminal nodes refer to deadlocks

*OGAnalyzer* has been developed. We assumed that the occurrence graph belonging to a given initial state of a CP-net model is finite and acyclic. The occurrence graphs of Petri nets modelling manufacturing systems fulfil this assumption in general.

Let the probability of faults be known from technological consideration and let the first step of the analysis be the assignment of the arc weights to the edges of occurrence graph as follows.

1. If a node on the occurrence graph has only one outgoing arc, then the next state follows unambiguously, thus the arc weight is equal to 1.
2. If there is more than one outgoing arcs from a given node then it means that different states can follow from it. These states come into existence with different probabilities depending on whether this branch has technological or fault related reason.
   a. If there is no fault in the system then every branch has a technological reason, because there is no built-in priority rule for the robot to the selection among the workpieces. It means that every selection that is every arc has the same probability so all of the arcs starting form this node have to get the same arc weight, which is equal to the reciprocal value of the number of outgoing arcs.
   b. The introduction of the fault into the model causes the appearance of another type of branching in the occurrence graph. Let us assume that only one type of fault can occur in a given system state. It results in 2 new system states: one for the normal operation and one for the faulty mode. Let the probability of fault be equal to $P_f$. Then the weight of arc leading to faulty mode is equal to the probability of the fault while the arc leading to the normal operational mode gets the value $1-P_f$.

Assigning these arc weights to the edges of the occurrence graph the probability of a given state on the graph can be determined in the following way if the faults occurring one after the other in the system are independent:

1. If the graph is a tree or only one route leads to the given node then the probability of the state representing by this node is equal to the product of arc weights along the route leading from the node representing the initial state to the given node.
2. If there are more than one route leading to the given node from the initial node then the probability value of each route has to be determined with the production of arc weights along the route as in the first step, then to sum these resulted values.
3. If the faults are not independent from each other, then the probability of nodes can be calculated in a similar way but using conditional probability values.

As described above, this determination method and the operation of OGAnalyzer has been illustrated using the example of two workpieces to be processed and one of them has to go through manufacturing line 2, while the other has to go first through line 1 then trough line 2. Faults can occur during the manufacturing process and let the fault probability be equal to 0.3 in case of line 1 and 0.1 in case of line 2. As before, we consider the timed case, when the transfer transitions have the same transition time of 1 time unit, while the manufacturing time is equal to 5 for both lines. After the simulation CPNTools generates the occurrence graph, the structure of, which is the input of OGAnalyzer.

As a first step OGAnalyzer reads the structure of the generated occurrence graph from the data file generated by CPNTools and visualizes it in its own window as it is shown in *Fig.10*. The user can get the token distribution belonging to nodes as it is shown at node 1 in *Fig.10*. The next step is the identification of branches. The software can distinguish between the two different types of branches on the occurrence graph (that are technological and fault caused branches). For this the user has to define the fault colour for the appropriate branches in a separate window.

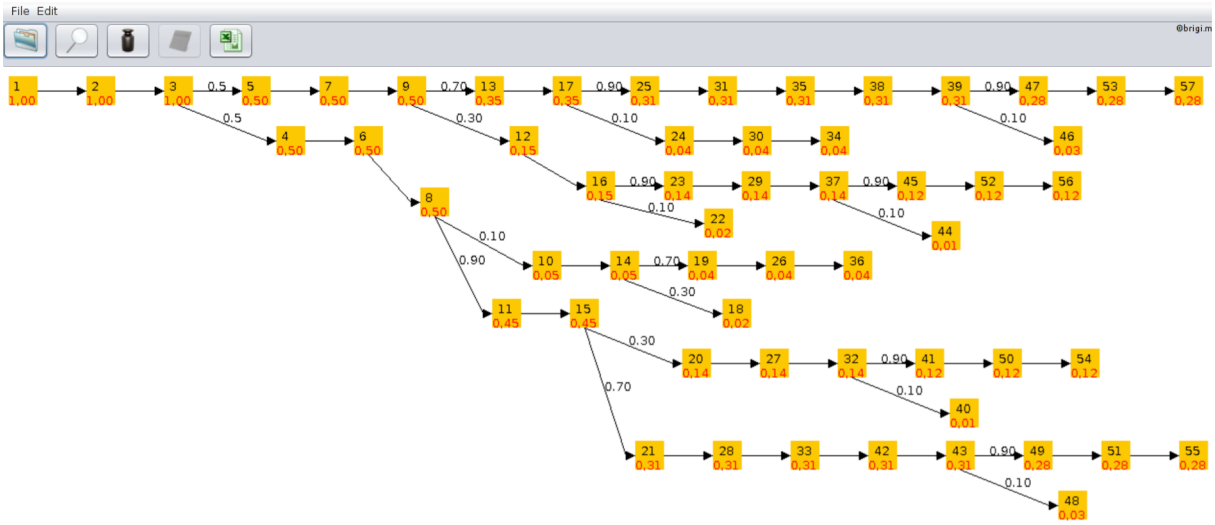


*Figure 10:* The occurrence graph of the example in the window of *OGAnalyzer*

It is assumed that the technological branches have the same probability, so OGAnalyzer assigns the reciprocal of the number of branches to these edges. The fault caused branches are collected into a table and the user has to define the probability of faulty and normal modes as it can be seen in *Fig.11*. The occurrence graph with arc weights depicted in OGAnalyzer can be seen in *Fig.12*. The black numbers attached to the arcs are the arc weights. Values equal to 1 were not depicted.

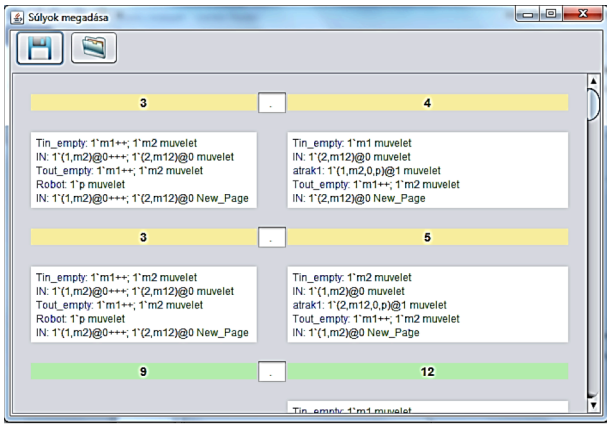


*Figure 11:* Defining the probabilities values of faults

Using these probability values the OGAnalyzer calculates automatically the probability of a node by multiplying arc weights along the path leading from the node representing the initial state to that node. If two or more paths lead to the node then the probability values belonging to these paths are summed. The calculated probability values can be also seen in *Fig.12* as red numbers assigned to nodes.

In this manner the probability of all system states in the investigated system can be calculated. For example, the probability of faultless completing of both workpieces is 0.28 + 0.28 = 0.56, which is equal to the sum of the probabilities belonging to the nodes 55 and 57. The probability of that case when the first workpiece is manufactured without fault, but the label of the other piece gets damaged during the second manufacturing process is 0.03 + 0.03 = 0.06 (sum of the probabilities belonging to the nodes 46 and 48).

In case of timed nets the occurrence graph is often a tree and the same token distribution belongs to different nodes because of the different time stamp. For the determination of probability of a given system state, the probability values belonging to different nodes have to be summed up.

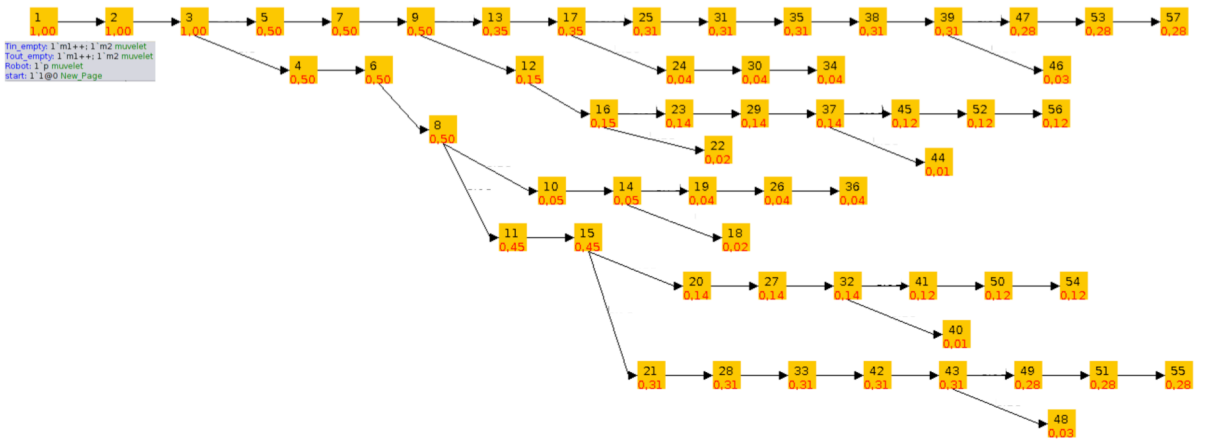

*Figure 12:* The occurrence graph with arc weights and probability values

## Conclusion

A novel occurrence graph investigation procedure for discrete event systems described by Petri nets was proposed in this paper for model-based diagnostic purposes that utilize the knowledge of the occurrence probability of faults. The model of the investigated system was defined in timed coloured Petri net form. The colours of tokens representing the workpieces were used to distinguish them and to assign a label of the processes to be carried out. The arc inscriptions and built-in probability functions were used for the fault modelling and integration.

The operation of the system was investigated via simulation both in timed and non-timed cases and in faultless and possible fault operational modes with given number pieces and prescribed manufacturing lines.

For the behavioural analysis of the model the occurrence graph method was used. A special software module, called OGAnalyzer has been developed for the handling of the probabilities on the occurrence graph and for calculating the occurrence probability of system states.

## Acknowledgements

This research is partially supported by the Hungarian Research Fund through grant No. K-83440. We also acknowledge the financial support of the Hungarian State and the European Union under the TAMOP-4.2.2.A- 11/1/ KONV-2012-0072.

## REFERENCES

[1] CASSANDRAS C.G., LAFORTUNE S.: Introduction to Discrete Event Systems, Kluwer Academic Publishers, *1999*

[2] GERZSON M., MÁRCZI B., LEITOLD A.: Diagnosis of Technological Systems based on their Coloured Petri Net Model, ARGESIM Report no. S38 (Eds. TROCH I., BREITENECKER F.) *2012*, p. 358/1–6

[3] JENSEN K.: Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use, Springer-Verlag, *1997*

[4] CPN GROUP, University of Aarhus, Denmark: CPNTools 2.2.0 http://wiki.daimi.au.dk/cpntools/ (last accessed: May 25, 2014)

[5] FANTI M.P., SEATZU C.: Fault diagnosis and identification of discrete event systems using Petri nets, Proc. 9th International Workshop on Discrete Event Systems, WODES, *2008*, 432–435

[6] JENSEN K., KRISTENSEN L.M., WELLS L.: Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems, Int. J. of Software Tools for Technology Transfer, *2007*, 9(3–4), 213–254

[7] MURATA T.: Petri Nets: Properties, Analysis and Applications, Proceedings of the IEEE, *1989*, 77(4), 541–580