A Compact CPN representation for Embedded and Control Systems
Fault Diagnosis and Recovery

Anthony Spiteri Staines
Department of Computer Information Systems
University of Malta
Msida, MSD 15
MALTA
toni_staines@yahoo.com, tony.spiteri-staines@um.edu.mt

Abstract: - This paper describes how a reduced colored Petri net modeling approach can be used to represent faults and recovery action when designing embedded systems, control systems, real time hardware and other systems. The idea of the reduced CPN is to represent errors using complex tokens based on sets to store error information. A comprehensive example of a computerized fuel control system for a gas/combustion turbine is used to illustrate this methodology. Results and conclusions are given.

Key-Words: - Fault Diagnosis, Petri Nets, Colored Petri Nets, Embedded Systems, Control Systems

1 Introduction
Given the ever increasing complexity of hardware and computer systems, fault diagnosis and recovery have become increasingly important [5]. Embedded systems, real time controllers, control systems, real time computing etc. are just a few areas that require proper fault identification and handling. Many devices and systems exhibit behavior typical of embedded controllers or IC based control.

Different solutions have been suggested in literature. However including solutions after the design phase has been completed, implies that the system might require some radical changes. The principle of ‘correct by construction’ needs to be applied solving issues from arising later.

2 Fault and Recovery Problems
The number of ‘states’ in modern systems is on the increase. Increase in states implies an increase in complexity. The traditional models used to represent these systems have limited modeling capabilities. Most traditional models are based on block diagram notations or some specific language set. Models and modeling techniques based on diagrammatic notations are unsuitable for the design stage because of the fact that the requirements of the run-time need to be experimented with at the design stage. The model needs to be executed and validated before actual system construction.

Executable models that represent recovery or failsafe principles based on auto recovery and self healing properties are required at the design stage [4],[5],[11],[13],[14]. Intrinsic management mechanisms, where functionality and intelligence are represented, need to be integrated into the final application. Errors need to be accounted for as they occur in the real situation. A system should manage them without external intervention.

3 A Compact CPN Solution
A colored Petri net having a reduced number of places and transitions using compound color sets can be used to solve these issues. The proposed model still retains the main properties and structure of P/T nets which is useful for other forms of analysis.

3.1 Petri Nets, Colored Petri nets and Fault diagnosis
Petri nets offer extended modeling capabilities over automata when representing system behavior [3],[11]. Automata have limited capabilities. Place transition nets have a large state space making them unfeasible to represent complex systems. CPNs are proposed for fault diagnosis in [1]. A special type of place called a Latent Nestling place is used to store tokens.

PN are a well proven formalism suitable for studying the construction/design and behavior of discrete event systems [2]-[11]. Various classes of Petri nets exist ranging from elementary nets (EN) to higher order nets and Object Oriented nets.

Simple place transition nets offer easy validation but are not so useful for complex modeling. They are similar to automata. Higher order nets [10] offer detailed complex modeling at the disadvantage of validation and used. Petri nets are supported with a vast amount of literature and other formal methods.
3.2 Informal Description of CPNs
Colored Petri nets are based on extensions to normal Petri nets [12]-[13],[19]. Colored Petri nets extend the modeling capabilities of the traditional place transition PT net. Colored tokens can be defined from different types ranging from simple to complex. The token in a colored Petri net can encode a vast amount of information that determines transition firing. A colored token is a token that has an associated data value which may be of simple or complex type. This property of CPNs is often overlooked. Places are associated with color sets. This specifies the tokens that the place can have. A transition can be programmed using special constructs and functions. Additional constructs can be used to enable or disable transition firing. Input and output arcs can have expressions and functions related to them.

For a transition to be enabled the input arcs expressions need to bind successfully with the tokens present in the input places and the transition guard. Transition firing depends on the binding and the resultant output is derived again from the transition output arcs and their expressions. The tokens are placed in the respective output places. CPNs being a class of higher order nets [10]-[13] offer the advantage of having a memory state that is controlled via the tokens themselves. Parameters, complex data types, arc inscriptions, complex firing rules etc. programmable in functional languages like ML [12],[18] offer a substantial degree of control.

They can be used for fault diagnosis and investigation in control systems offering several advantages over traditional FSMs and Place transition nets.

3.3 Encoding Places with Information
A good process model should encode all possible information about the system it is describing. Events bring about state changes. States may be simple or compound. In the latter case a global state is composed of several sub-states. The states or errors that result from an event can be defined as $E = \{e_1, e_2, e_3, ..., e_n\}$. $E$ is a finite set of errors or states where $E \neq \phi$. An event can be considered to be atomic i.e. when a event is occurring another event cannot take place. But an event can have one or more transitions. $E$ is the global state composed of a set of elements $e$, where each element $e$ is a sub-state or sub-state value, i.e. the state of a device or some part of the system hence $e_i \in E$ and $e_i$ is derived from a fixed value range. A system event, transition or error can bring about a change in the one or more of the sub-states changing the global state.

High level places are used. A product color set is defined. The product color set is used to represent the sub-states of a device. E.g. In the CPN ML language specification [18] `colset device_status = product exexexe`. $e$ is an int type restricted to 100 values. The result is a compound color set created from pre-defined sets.

Given that the int type $e$ contains a value from 0 – 100, where 0 is no error and 1..100 are error values the resultant sum of all possible error combinations is 100.100.100.100 = $100^4$. Hence it is possible to use a single place to model errors having a set of 0..100 values. It is possible to increase the set size and number of values in the set. This approach drastically reduces the need for additional places in the Petri net.

3.4 Reduced Model Size
The Colored Petri net with the places used for error combinations allows all the information about the system or device states to be kept in a single token contrary to multiple tokens that have to be used in [1],[14] and other approaches. A single transition can model normal and all other abnormal conditions.

4 Comprehensive Example
4.1 Gas Turbine Fuel Control System
A computerized fuel control system of a gas turbine adapted from [16] is illustrates this approach. A gas turbine normally operates by using a burner or combustor. Fuel is used to heat compressed air extracting power form the hot air flow. The gas turbine is composed of several control systems, one of them being the fuel control. The gas/combustion turbine with the fuel control is illustrated in fig. 1.

Different diagrams and notations like UML activity diagrams, design structured diagrams (DSD), block notations, etc. can be used to show the operations and activities of the fuel control system. Fig. 2 shows the UML activity diagram constructed for this system.

For the fuel control system the main sequential steps are i) get engine parameters, ii) compute fuel requirements, iii)compute fuel schedule, iv)compare requirements with schedule and v) output estimated fuel command to actuator.

At each activity level an error discovery/handling routine is introduced. This implies that if a
step/action or activity fails the system will try to auto recover from the error and try to execute the step again. This is called a reset point and is typical of embedded system behavior.

For each activity it is possible to define a finite set of errors that can occur prior, during or after that activity. The errors will prevent the next activity from occurring until they are rectified.

At the first activity ‘get engine parameters’ there are at least four sensor values to read in once or more times. See fig. 1. E.g. i) Fuel flow feedback $F_{\text{Flow}}$, ii) Inlet pressure $P_{\text{in}}$, iii) Compressor speed $C_{\text{in}}$ and iv) demand signal $P_{\text{dem}}$. Each particular sensor $S_{1}..S_{4}$ can have a set of error values $e_{1}..e_{4}$. E.g. 1) no reading, 2) hardware error, 3) value too low, etc. and also unknown errors or different categories of hardware errors. E.g. the sensor can have a value from 0-99 where 0 is no error and all the other values record some error. If after execution of ‘Get Engine Parameters’ we have a token with (0,1,0,0) then an error has occurred in reading the Inlet pressure value $P_{\text{in}}$. Other errors can be identified for the other activities e.g. for ‘compute fuel schedule’ it is possible to define i) hardware failure having different values e.g. 1-50, ii) program errors having values 1-90, etc. These are again represented using the common set $\{e_{1},e_{2},e_{3},e_{4}\}$.

The UML activity diagram in fig. 2 depicts the basic activities that are taking place. Pin notations have been used in the activity diagrams to indicate the states of the fuel_control and enforce error handling control.

### 4.2 Reduced Colored Petri Net Model

The fuel control system main operations along with error handling and recovery are modeled in the Colored Petri net shown in fig. 3.
The CPN is built based on principles of sequential composition of places. It can be proven that sequential composition preserves soundness in certain classes of Petri nets like P/T nets, workflow nets, etc. The CPN was built using the CPN Tools and standard ML functions [18]. The CPN is an executable model that can be used for detailed system simulation and property investigation.

When executing the CPN the error data values can be entered manually or using specific functions. The second option was used to generate random errors that fall in the range from 1.99 or no error with a 0. Even the generation of no errors to a maximum of 4 errors was randomized. The code fragment in fig. 4 depicts the functions that were created to randomize error generation.

5 Results
5.1 Execution of the CPN
The CPN in fig. 3 was executed successfully using the CPN Tools [18]. All transitions were fired including all the reset ones. The tasks terminate successfully. The random generation of errors was successful. It is possible to program the function no_error() and error() to ignore trivial errors that would not halt the next step from being executed. During execution after a step it is possible to have errors or no errors at all. The errors are then reset by the reset transitions. Table 1 summarizes the token error or no error values generated randomly after the execution of each step for four successive runs.
If other runs are performed different values will be obtained randomly. The data in Table 1 demonstrates the correct functioning of the random error generation mechanism.

When a transition like ‘get engine parameters’ occurs the output edge of this transition invokes the random_error() function which generates an error or no error. This is placed in the connected place defined as type status. The next step is that one of the next two transitions are enabled but not both simultaneously.

### 5.2 Experimental Value of the Model

From execution of the CPN it is possible to play with different scenarios and conditions. More functions can be added for analyzing other scenarios.

### 5.3 Compactness and Patterns

The CPN model is more compact than most other Petri net models used. There are 11 transitions and 7 places initially. It is possible to reduce the model by combining transitions. The CPN presents a repeated sequential pattern that can be used for analysis and converted into other formalisms and notations [17].

### 5.4 Colored Petri Net Analysis

Different methods can be used to analyze the CPN [12]-[13],[18]-[19]. The CPN in fig. 3 is modified, the start and termination places are connected using another transition to create a repetitive cycle. Please note that this is not shown in fig. 3. The CPN can be reduced into a P/T net and incidence matrix analysis, invariants etc. can be used to determine basic properties like reachability, liveness, deadlock, boundedness, cyclic behavior, home states, etc. Here the CPN state space and strongly connected component graph SCC-graph inbuilt tools were used [13],[18].

State space analysis can be used to formally verify the model mathematically identifying all the reachable states. Normally the state space is a directed graph having a node for each reachable marking connecting it to the next reachable marking via an edge. This state space shows us some of the properties of the CPN. I.e. a large state space indicates that the model is more complex than one with a small state space. There are less chances of problems if the state space has a reduced size.

Table 2 indicates that the state space for the modified CPN is very small. There are just 10 nodes and arcs. The time to calculate this was minimal hence 0 seconds. The SCC-graph is a sub-graph of the state space telling us that we have a cycle in the state space. Similarly to the state space the reachability graph can be constructed.

### 6 Conclusion

The results obtained successfully indicate that it is possible to construct a reduced CPN model for control and embedded systems using a place type defined using sets. This approach still caters for complex error handling. It is also has been shown in the CPN analysis that the structure has a limited small state space which is highly desirable compared to other more complex models. The reduced CPN can be converted to other formal approaches. The results presented are a just a brief summary of the possibilities that can be investigated.

If the CPN model is compared to a place transition net that would model the errors, then we would
require a large numbers of places and transitions. In this work the model obtained is quite compact. It can be reduced into a standard place transition net for other forms of analysis. This done using PT net rules for transition and place fusion or augmentation. This model opens up the possibility for other forms of simulation. The time dimension can be included, the transitions in the model can be decomposed further by organizing them as a set of modules containing other levels of transitions [13]. It can be used for added complexity e.g. if an error occurs the next activity could still be allowed because the error is trivial. In this case the guard or a function must be added to cater for this.

If the CPN model is used for complex systems we could still end up having many places and transitions. It is recommended to use this approach for embedded systems and strict control systems.

References:
[18] CPNTools, CPN Group, Department of Computer Science, University of Aarhus, Denmark. http://www.daimi.au.dk/CPNets/