A Compact Colored Petri Net Model for Fault Diagnosis and Recovery in Embedded and Control Systems

A. Spiteri Staines

***Abstract*—** This paper describes the modeling and use of a reduced Colored Petri net for fault diagnosis and recovery in embedded and control systems. The reduced or compact Colored Petri net modeling approach can be extended to other classes of real time systems, real time hardware, etc. A reduced colored Petri net is a compact form of a Colored Petri net having complex token types based on sets or complex sets containing the structured information for error handling. The approach presented here will reduce the size of the Colored Petri net because information is put in the token instead of having many additional places and transitions as is typically done. This approach is illustrated with a comprehensive example of a computerized fuel control system for a combustion turbine. The Colored Petri net is an executable model. It is analyzed structurally and results are shown and interpreted.

***Keywords*—** Fault Diagnosis, Petri nets, Colored Petri nets, Embedded Systems, Control Systems

I. Introduction

Fault diagnosis and recovery have become increasingly important over the past decade [5]. As hardware and computer systems evolve in complexity, embedded systems, real time controllers, control systems and real time systems have more reliability requirements than before. Many modern devices and systems exhibit behavior typical of embedded controllers or IC based control. These systems are just a few areas which require proper fault identification and handling. Including solutions after the design phase is not an option. The principle of 'correct by construction' should be applied. This would solve any issues that might arise.

In literature different methods and approaches are suggested. Different models ranging from static block diagram notations to discrete event models are used as required. Other approaches like formal languages have been developed.

II. Fault and Recovery Issues

As software functionality in modern embedded devices and control systems increase the system's state space will also increase accordingly. Increase in states also 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, must be integrated into the final application. Errors should be accounted for as they occur in the real situation. A good system should manage them with the least amount of external intervention.

III. A Colored Petri Net 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.

A. Petri Nets, Colored Petri nets and Fault diagnosis

Discrete event models are used to model the error handling and fault diagnosis for certain classes of embedded and control systems. Petri nets offer extended modeling capabilities over automata when representing system behavior [3],[11]. Automata have limited capabilities. If unreduced place transition nets are used, it is possible to end up with a large state space making them unfeasible to represent complex systems error handling. Colored Petri nets are proposed for fault diagnosis in [1]. A special type of place called a latent nestling place is used to store tokens.

Petri nets are a well proven formalism suitable for studying the construction/design and behavior of discrete event systems [2]-[11]. Different classes of Petri nets exist, ranging from elementary nets (EN) to higher order nets and Object Oriented nets. In literature Petri nets have been used to model different

Manuscript sent Dec 31, 2008: revised Feb 2009

A. Spiteri Staines is with the Department of Computer Information Systems, Faculty of ICT, University of Malta, Msida, MSD 2080, Malta, Europe. phone: 00356-21373402; fax: 21312110; (e-mail: toni_staines@yahoo.com, tony.spiteri-staines@um.edu.mt)

forms of discrete processing in computing ranging from communication networks to hardware components and real time systems. Petri net models can be decomposed using rules for fusion and augmentation of places and transition. Higher order nets can contain a vast amount of information which is encoded in the tokens, places and transitions.

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 ease of use. Petri nets are well supported with a vast amount of literature and other formal methods.

B. Informal Description of Colored Petri Nets

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 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 Colored Petri nets is often overlooked. Places are associated with color sets. The token types of places are specified using a special language or functions. Transitions also 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. In short there is the possibility to include different rules and conditions for different components of the Petri net.

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 the arc expressions. The tokens are placed in the respective output places.

Colored Petri nets being a class of higher order nets [10]-[13] offer the advantage of having a complex memory state that can be controlled via the tokens themselves. Parameters, complex data types, arc inscriptions, complex firing rules etc. are programmable in functional languages like ML [12],[18]. These features offer a substantial degree of control.

Colored Petri nets can be used for fault diagnosis and investigation in control systems offering many advantages like reduced model size, reduced marking graph, more realistic execution, etc. over traditional FSMs and place transition nets.

C. Compact Colored Petri Net Model Approach

The idea for creating reduced or simplified Petri net models is already known [20]-[22]. E.g. in [20] a task interaction graph based Petri net (TBPN) is used to create a reduced size Petri net for Ada task programs. As a general idea, a smaller Petri net has a simpler reachability graph which is simpler to construct. In Petri net theory various rules have been established for the reduction of models by combing places or transitions. Normally the approach is to create the complex model and simplify later. A Colored Petri net still retains structural similarity with other types of Petri nets.

A Colored Petri net can be structurally reduced more than a place transition net and yet have more information. In this work the idea presented is that of using a compact Colored Petri net model from the start to keep the number of places, transitions and arcs used to a minimum. The information about the errors resides with the tokens. The tokens are based on sets and can contain all error values or none. Here the same token type is used thought the net hence simplifying it even more than the Colored Petri net latent nestling method presented in [1], [14]-[15] and other approaches where multi token types are normally used. The result is that the final Colored Petri net structure is kept simple and it is more suitable for analysis and investigation.

D. 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 $E= \{e_1,e_2,e_3,\ldots,e_n\}$. E is a finite set of errors or states where $E \neq \phi$. An event can be considered to be atomic i.e. when an 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 , $e_i \in$ 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 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.

E. 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. The single transition can model normal and all other abnormal conditions.

IV. COMPREHENSIVE EXAMPLE

A. Gas Turbine Fuel Control System

The error handling of a computerized fuel control system for a gas turbine is used to illustrate the compact or reduced Colored Petri net approach.

A gas turbine normally operates by using a burner or combustor. Fuel is used to heat compressed air extracting

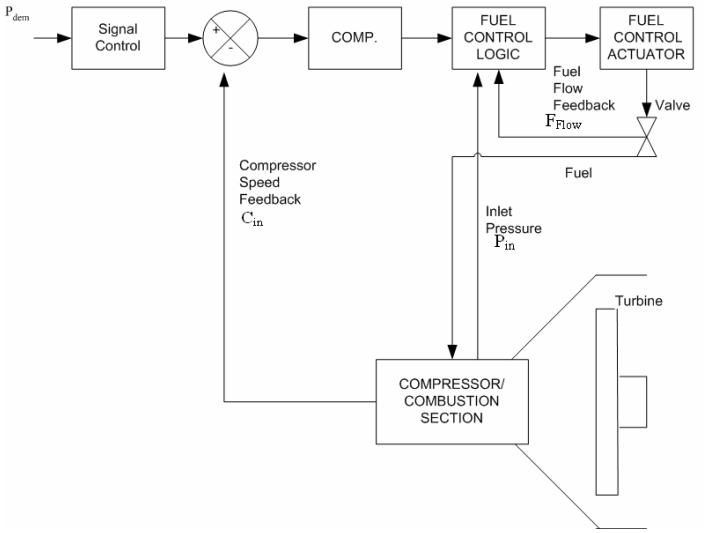

Fig. 1 Gas Turbine/ Compressor Fuel Control System
adapted and modified from [16]

power form the hot air flow. The gas turbine is composed of several control systems, one of them being the fuel control. The gas turbine fuel control system is a example of an embedded real time computerized control which has special requirements. The gas/combustion turbine along with the fuel control section which has been adapted from [18] and modified is shown in figure 1.

Different diagrams and notations like UML activity diagrams, data 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

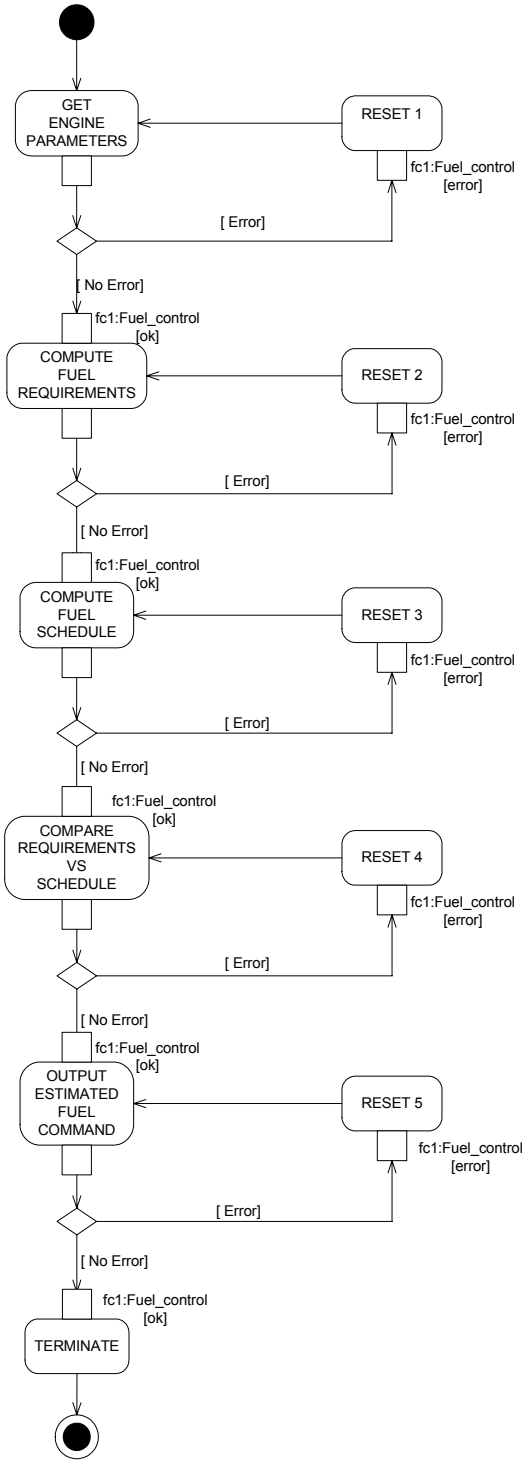

Fig. 2 Gas Turbine Fuel Control System Activity Diagram

are rectified i.e. removed automatically or manually.

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_{Flow}, ii) Inlet pressure P_{in}, iii) Compressor speed C_{in} and iv) demand signal P_{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_{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.

B. Reduced Colored Petri Net

The fuel control system main operations along with error handling and recovery are modeled in the Colored Petri net shown in fig. 4.

The Colored Petri net was built using the 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 Colored Petri net was constructed using the CPN Tools and standard ML functions [18]. The Colored Petri net is an executable model that can be used for detailed system simulation and property investigation.

When executing the Colored Petri net the error data values can be entered manually or using specific functions. It was opted to go for the second option where random error generation was done using functions. For each error, error values are randomly generated in the range from 1..99 or a 0 value, implying that there was no error. The error combinations generated range from 0 combinations to a maximum of 4 error combinations.

```
fun error_gen()=discrete(0,1);
fun error_value(c:int): int =( if c > 0
  then discrete (1,99) else 0);
fun err_no()= discrete (0,4);
fun error_gen1():int=(error_value(error_gen
  ()));
fun ran4(s:status): status=
  ( error_gen1() , error_gen1(), error_gen1(),
  error_gen1());
fun ran3(s:status): status= ( #1 s,
  error_gen1(), error_gen1(), error_gen1());
fun ran2(s:status): status= (# 1 s, # 2 s,
  error_gen1(), error_gen1());
fun ran1(s:status): status= (#1 s, # 2 s,
  # 3 s, error_gen1());
fun ran0(s:status): status=(#1 s, # 2 s,
  # 3 s, # 4s);
fun random_error(s: status, c: int) : status=(
  if c = 4 then ran4(s) else (if c =3 then ran3(s)
  else (if c = 2 then ran2(s) else ( if c = 1 then
  ran1(s) else (ran0(s))))));
```

Fig. 3 CPN Functions for Random Error Generation

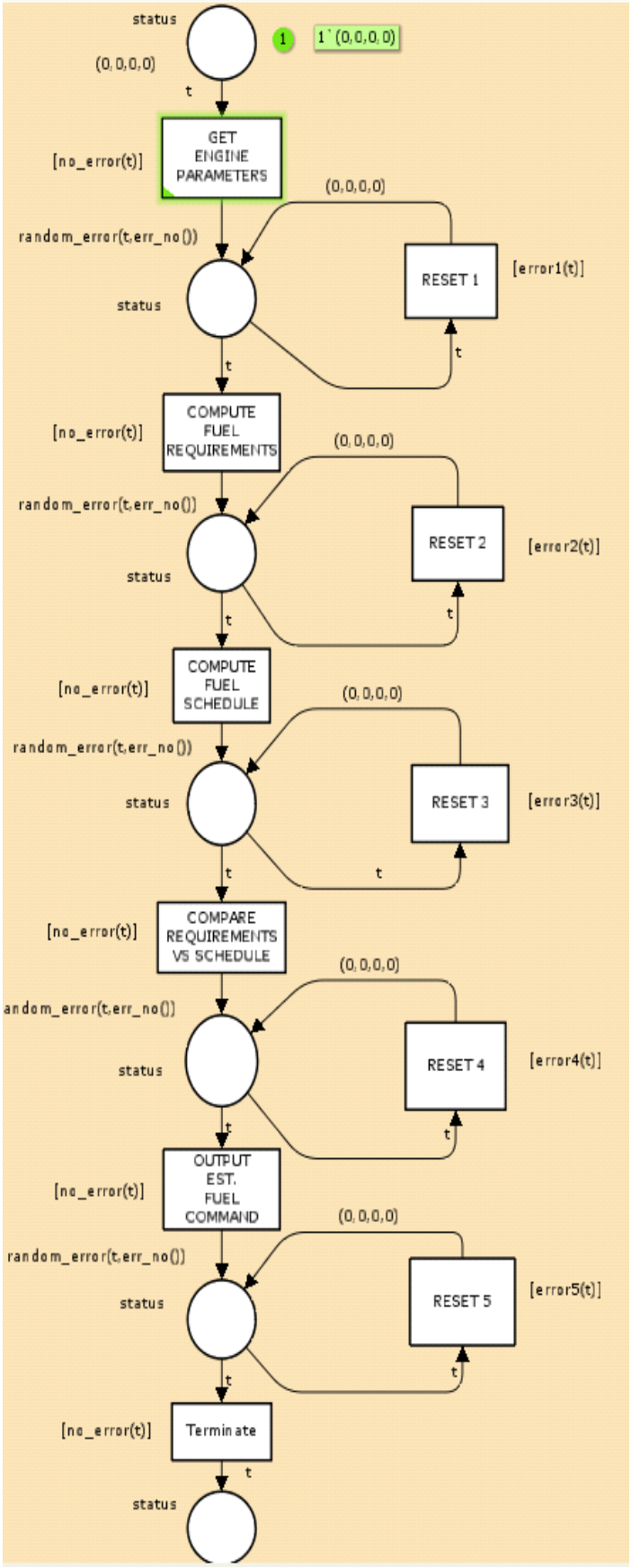

Fig. 4 Reduced CPN for Gas Turbine Fuel Control Faults

Error generation was randomized. The code fragment in fig. 3 depicts some of the functions that were created to randomize error generation.

V. RESULTS

A. Execution of the Colored Petri Net

The Colored Petri net in fig. 4 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 I 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 I and II demonstrate 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 it is not possible to enable both simultaneously. More detailed analysis can be performed by including the time for the transition firings.

Table I Random Token Data Run 1 and 2

TRANSITION EXECUTION	RUN 1	RUN 2
1 GET ENGINE PARAMETERS	(12,27,0,0)	(0,0,0,0)
2 COMPUTE FUEL REQUIREMENTS	(0,97,12,75)	(0,0,0,40)
3 COMPUTE FUEL SCHEDULE	(0,0,4,63)	(0,0,77,30)
4 COMPARE REQUIREMENTS VS		
SCHEDULE	(0,0,0,72)	(0,0,0,0)
5 OUT. ESTIMATED FUEL COMMAND	(0,0,0,0)	(0,0,96,0)

Table II Random Token Data Run 1 and 2

TRANSITION EXECUTION	RUN 3	RUN 4
1 GET ENGINE PARAMETERS	(0,0,85,9)	(0,0,0,0)
2 COMPUTE FUEL REQUIREMENTS	(0,0,74,29)	(0,8,85,0)
3 COMPUTE FUEL SCHEDULE	(32,0,0,0)	(0,23,59,67)
4 COMPARE REQUIREMENTS VS		
SCHEDULE	(0,0,36,0)	(0,0,27,28)
5 OUT. ESTIMATED FUEL COMMAND	(0,6,67,0)	(0,0,0,0)

B. Experimental Value of the Model

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

C. Compactness and Patterns

The Colored Petri net model is more compact than most other Petri net models used.

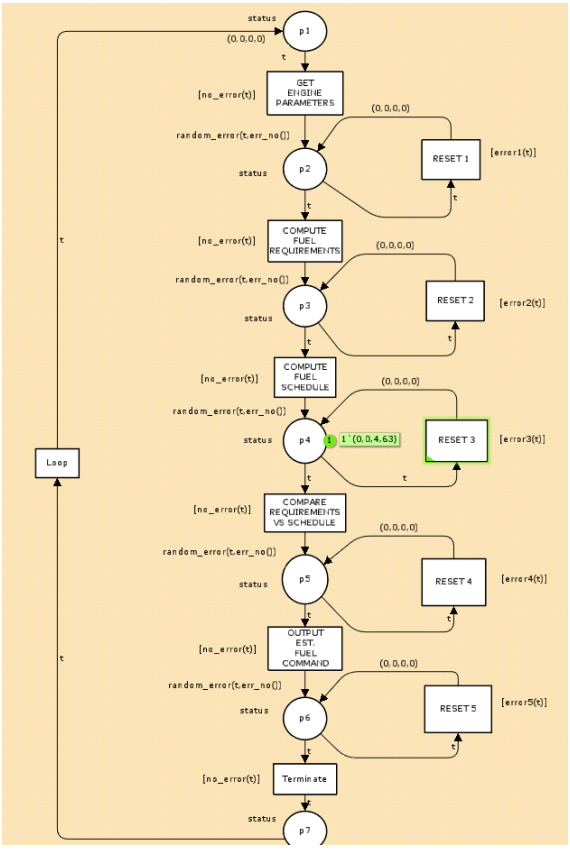

Fig. 5 Reduced CPN with Added Loop Showing Error

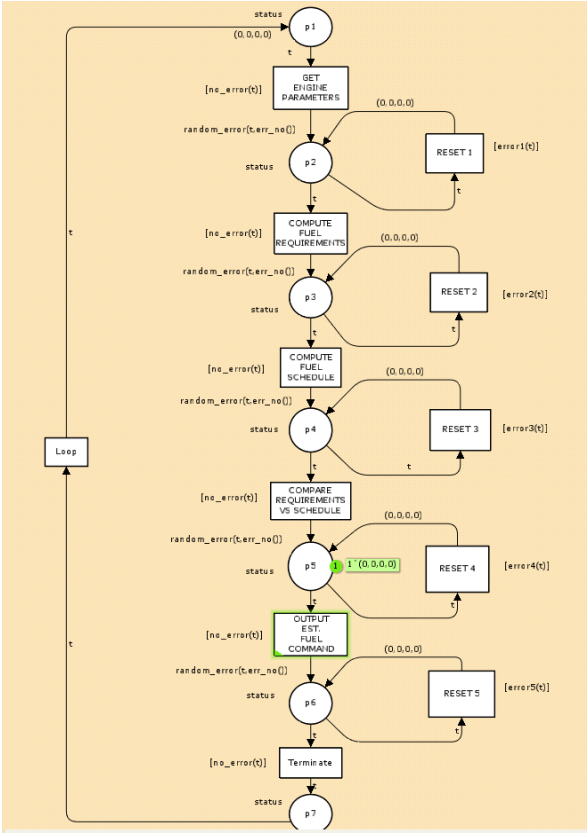

Fig. 6 Reduced CPN with Added Loop No Error

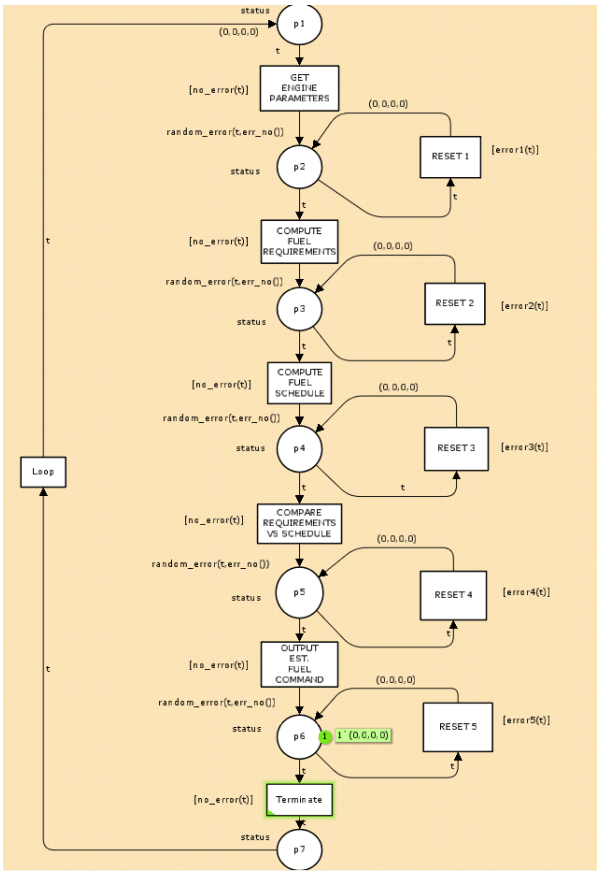

Fig. 7 Reduced CPN Successful Termination

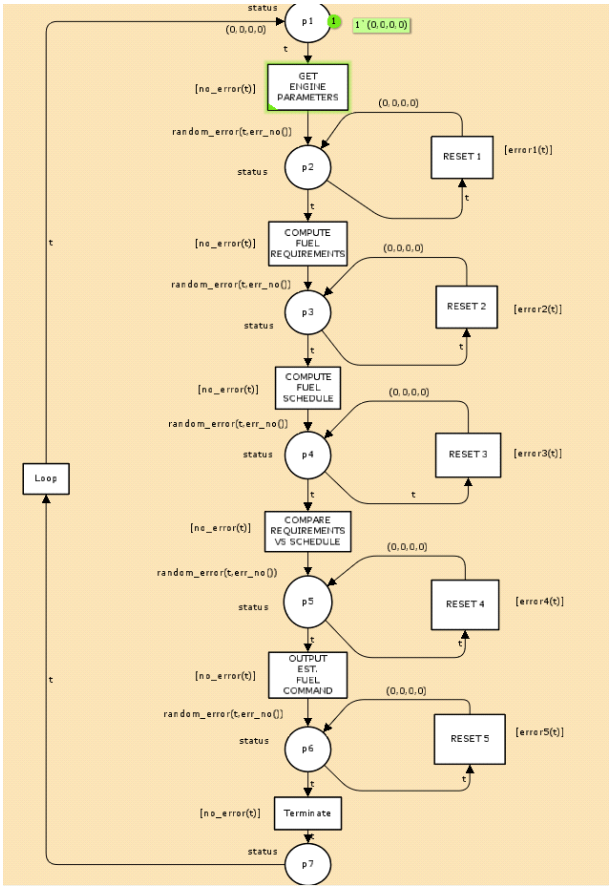

Fig. 8 Reduced CPN Initialization

There are just 11 transitions and 7 places initially. It is possible to reduce or simplify the model by combining transitions or places. The Colored Petri net represents a repeated sequential pattern that can be used for analysis and converted into other formalisms and notations [17]. The repeated pattern also indicates the equivalent UML 2 activity diagram.

D. Colored Petri Net Analysis

Different methods can be used to analyze the Colored Petri net [12]-[13],[18]-[19]. The Colored Petri net in fig. 3 is modified, the start and termination places are connected using an additional transition which creates a repetitive cycle. This is shown in fig. 5 and 6 but not in fig. 4. In fig. 6 the transition "output estimated fuel command" is shown enabled for firing. This is because the token in fig. 6 does not contain any error values i.e. it is (0,0,0,0) contrary to that in fig. 5 which reads (0,0,4,63), implying that there is error value 4 on S_3 and error value 64 on S_4. The errors in fig. 5 disable the next step from firing and instead the reset point is enabled. Once the error is reset it is possible to continue normally with the next transition. Fig. 7 shows successful termination of the model. After termination the model is re-started again. Fig. 8 shows the initialization of the model and the token contains no error.

The Colored Petri net can be reduced into a place transition net and incidence matrix analysis, invariants etc. can be used to determine basic properties like reachability, liveness, deadlock, boundedness, cyclic behavior, home states, etc. These properties represent the structural properties of the Petri net. Here the Colored Petri net 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 by 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 Colored Petri net. 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. For the model in fig.5 the state space graph just has 10 nodes and 10 arcs which is very small.

Table III State Space and SCC Graph for modified CPN

	STATE SPACE	STRONGLY CONNECTED COMPONENT GRAPH
Nodes	10	1
Arcs	10	0
Time in seconds	0	0
Status	FULL	

Table III indicates that the state space for the modified Colored Petri net is very small. The time to calculate this is negligible hence the 0 seconds. This is indicates that the

model is reduced or simplified.

The strongly connected component (SCC-graph) is a sub-graph of the state space telling us that we have just one cycle in the state space. Similarly to the state space the reachability graph or occurrence graph can be constructed.

Table IV Home and Dead Markings for modified CPN

Home Markings	ALL
Dead Markings	NONE

Table IV indicates that the modified Colored Petri net has cyclical behavior. It has home markings and no dead markings. This means that the behavior is repeatable and the net has sound formal properties and behavior, i.e. it is deadlock free. If the model is transformed or reduced into a place transition net, place and transition invariants analysis can be performed and similar results should be obtained.

These results would give further evidence of the compactness of the model's structural properties. These properties are very important for embedded and control system behavior.

If we consider the fairness properties of the net it has fair and impartial results because there is the option to select one transition for firing from two transitions, it is similar to a choice.

VI. CONCLUSION

The idea of reducing or simplifying Petri nets is well known in conventional and fundamental Petri net theory. Having a reduced or simplified Petri net offers several advantages from the analysis point of view. Ideally reducing or simplifying a complex Petri net should be possible and the most important properties need to be preserved. In this work the idea is to try to construct a reduced Colored Petri net from the start. This is possible for error handling because much of the error information can refer to the same step or process.

It has been shown how a reduced Colored Petri net model can be constructed for modeling the error handling logic for the fuel control system of a gas turbine or compressor. The results obtained successfully indicate that it is possible to construct a reduced Colored Petri net 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 Colored Petri net analysis that the structure has a limited small state space and small marking graph. Models having a small state space are preferable to those with a large state space. Complex models can have a large state space making their analysis and control more difficult and prone to problems.

The reduced Colored Petri net can be converted to other formalisms and formal approaches such as Z or VDM. Schemas can be created to represent various aspects of the system's behavior formally.

The approach presented here opens up the possibility for further experimentation. The results and analysis obtained are just a brief summary of the possibilities that can be opened up for investigation.

If the Colored Petri net 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 is done using place transition net rules for transition and place fusion or augmentation.

This model can be used for other forms of simulation. The time dimension can be included. This would open up the possibility for more detailed simulation and modeling. It is possible to include time transitions that could act as reset points. E.g. if an error occurs it can be corrected in a given period of time.

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 to take place if the error is trivial. In this case the guard or a function must be added to cater for this.

Although the reduced Colored Petri net approach presented can solve problems for certain classes of systems it is still possible to encounter problems. If a complex system needs to be modeled and the errors are not compacted we could still end up having many places and transitions. Another problem is that constructing the Colored Petri net is a time consuming task. The idea of using compacted Petri net models is definitely desirable because it simplifies the analysis issues involved.

To conclude it is recommended to use this approach for embedded systems and strict control systems, owing to the advantages over other approaches.

REFERENCES

[1] E. Garcia, L. Rodríguez, F. Morant, A.Correcher, E.Quiles, R. Blasco, "Fault Diagnosis with Colored Petri Nets using Latent Nestling Method", *Proc. of ISIE08*, Cambidge, UK, Jun 2008.

[2] B. Yang, S.K. Jeong, Y.M. Oh, A.C. Chiow Tan, "Case Based-Reasoning systems for Induction Motor Fault Diagnosis", *Expert Systems with applications*, Elservier, Vol 27 issue 2, Aug 2004, pp. 301-311.

[3] J. Brusey, D. McFarlane, "Designing Communication Protocols for Holonic Control Devices using Elementary Nets*", Holonic and Multi-Agent Systems for Manufacturing, 2nd Int. Conf. on Industrial Applications of Holonic and Multi-Agent Systems*, Aug 2005, pp. 76-78.

[4] L. A. Cortes, P. Eles, Z. Peng , "A Petri Net Based Model for Heterogenous Embedded Systems", *Proc. Norchip Conf.,* Oslo, Norway, Nov 1999, pp. 248-255.

[5] V. Baggiolini, J. Harms, *Generic Fault Management Techniques*, Hpovua Publications, Technical Report,Univ. of Geneva, Switzerland www.hpovua.org/publications/proceedings/5_hpovuaws/62.ps.gz, 1998.

[6] K. Grigorova, "Process Modelling Using Petri Nets", *Int. Conf. on Computer Systems and Technologies*, CompSysTech'2003, Bulgarian Computer Science Conference, Sofia, Bulgaria, Jun 2001.

[7] D.I. Kharitonov, G.V. Tarasov, "Towards Petri Nets application in Parallel Programming Debugging",*6th Asian Computational Fluid Dynamics Conf.*, Taiwan, Oct 2005.

[8] M. Rautenberg, S.Chluep, M. Fjeld, "Modeling of Cognitive Complexity with Petri Nets an Action Theoretical Approach", R. Trappl(ed..) *Cybernetics and Systems'98*, Vol2 , Wein, Austria, pp.842-847.

[9] J. E. Hopcroft, R. Motwani, J.D. Ullman,*Introduction to Automata Theory, Languages and Computation*,Addison-Wesley, 3rd ed., Jul 2006, ISBN-13 9780321462251.

[10] K. Hoffmann, T. Mossakowski, "Algebraic Higher-Order Nets: Graphs and Petri Nets as Tokens", *16th Int. wsop WADT*, Lecture notes in CS Springer-Verlag, Vol. 2755, Nov 2003, pp.253-267.

[11] J.W. Janneck, R. Esser, "Higher-Order Petri Net Modeling – Techniques and Applications", *ACM Int. Conf. on Application Theory of Petri Nets : Formal Methhods in Software Engineering & Defence Systems*, Vol. 12, Adelaide, Australia, 2002, pp. 17-25.

[12] L.M. Kristensen, S. Christensen, K. Jensen, "The Practioner's Guide to Coloured Petri Nets", *International Journal On Software Tools for Tech. Transfer (STTT)*, Vol. 2, Springer-Verlag,1998, pp. 98-132.

[13] K. Jensen, L. M. Kristensen, L. Wells, "Colored Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems", *International Journal On Software Tools for Tech. Transfer (STTT)*, Springer- Verlag, Vol. 9, Springer-Verlag, 2007, pp. 213-254.

[14] E. Garcia, F. Morant, A. Correcher, E. Quiles, "Application of Latent Nestling Method using Colored Petri Nets for Fault Diagnosis in the Wind Turbine Subsets", *Proc. of EFTA08*, Hamburg, Germany, Sep 2008.

[15] E. Garcia, L. Rodríguez, F. Morant, A.Correcher, E.Quiles, R. Blasco, " Latent Nestling Method: A New Fault Diagnosis Methodology for Complex Systems", *Proc. of IECON08*, Orlando, Florida, USA, Nov 2008.

[16] J.E. Cooling, *Software Design for Real-Time Systems*, Chapman & Hall, London, 1995.

[17] T. Gehrke, U. Goltz, H. Wehrheim, "The Dynamic Models of UML: Towards a Semantics and its Application in the Development Process", *Technical Report Informatik-Bericht 11/98*, University of Hildesheim, Germany, 1998.

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

[19] K. Jensen, G. Rosenberg, *High-Level Petri Nets: Theory and Application* , Springer – Verlag, Berlin, 1991.

[20] M.B. Dwyer, L.A.Clarke, "A Compact Petri Net Representation and Its Implications for Analysis", IEEE Transactions on Software Engineering, Vol 22 no 11, Nov 1996, pp. 794-811.

[21]] M.B. Dwyer, L.A.Clarke, "A Compact Petri Net Representation for Concurrent Programs", Proc. 17th Int. Conf. Software Engineering,Apr. 1995, pp. 147-148.

[22] S.M. Shatz, S.Tu, T. Murata, S. Duri, " Theory and Application of Petri Net Reduction for Ada Tasking Deadlock Analysis", Technical Report, Dept. Of Electrical Enginnering and Computer Science, Univ. Of Illinois, Chichago, USA, 1994.