IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 22, 2018, accepted March 23, 2018, date of publication April 3, 2018, date of current version May 9, 2018.

*Digital Object Identifier 10.1109/ACCESS.2018.2822763*

# A Novel Model-Based Dynamic Analysis Method for State Correlation With IMA Fault Recovery

**RONGBIN HAN[1], SHIHAI WANG[2], BIN LIU[2], TINGDI ZHAO[2], AND ZHIAO YE[1]**

[1]School of Reliability and Systems Engineering, Beihang University, Beijing 100083, China
[2]Science and Technology on Reliability and Environmental Engineering Laboratory, School of Reliability and Systems Engineering, Beihang University, Beijing 100083, China

Corresponding author: Bin Liu (liubin@buaa.edu.cn)

This work was supported by a grant from the Major State Basic Research Development Program of China (973 Program) under Grant 2014CB744904.

**ABSTRACT** Integrated modular avionics (IMA) systems present many advantages. However, the resource sharing mechanism also brings a series of system problems, including the frequency of fault propagation and the difficulties of system design verification. The traditional analysis approaches for system designers have limits to analyze dynamic faults which are caused by unreasonable designs. These dynamic faults come up with component fault states, component state correlation, and system dynamic behaviors. In this paper, a new model-based dynamic analysis method for state correlation with IMA fault recovery is proposed, which helps to check system states and verify system designs by means of analyzing the dynamic behaviors of systems in a new view of systems' correlated states. A colored generalized stochastic Petri net (CGSPN) provides advantages to system modeling and simulation, but there are some difficulties for modeling component state correlations and system dynamic behaviors in detail on the IMA system. We make an improvement on CGSPN for modeling IMA by adding an element and changing fairing rules. In addition, multiconstraint specified to solve the configuration satisfying problem for IMA is built into the model. Afterward, according to results of model simulation, system dynamic faults are analyzed and system designs are checked, which will help to guide the system designers to adjust system architecture at the early stage of system development. Finally, a case study is given for demonstrating how to apply this new method.

**INDEX TERMS** IMA, state correlation, fault recovery, petri net.

## I. INTRODUCTION

The Integrated Modular Avionics (IMA) is a highly integrated system with a set of sharing hardware and software resources. It is dedicated to improving the performance of avionics systems, enhancing safety as well as reducing the life cycle cost [1]. Nowadays, it is an emerging trend [2] for on-board avionics systems in both military and civil areas to be equipped with IMA systems.

Features of IMA system are the resource sharing strategy [3], [4], fault recovering strategy [5] and configuration [6], [7]. Compared with traditional federated avionics architectures, IMA architectures provides a shared platform with the flexible and complex hardware and software components. On the IMA platform, health monitoring (HM) and fault management (FM) are introduced to ensure expected behaviors of systems in the presence of system faults. The health monitoring module is responsible for identifying, masking and reporting the failure of hardware, operation systems and applications. After the fault management module receives reports from the HM module, it will locate and handle the specific fault to conduct trouble-shooting work with the assistance of the runtime blueprint. This runtime blueprint is the configuration of IMA, which is much more flexible with the concept of reconfiguration [8] being raised.

Although IMA presents the improvement of functionality and performance of system with those features, a series of system problems comes up along with the advantage. In the form of state correlation, the problem is inherent on the IMA platform. State correlation is the influence between system states. For example, when a shared resource fails to work in a fault state, objects using this resource may fall into fault states. Besides, dynamic properties of running systems come with correlation of system states, which brings challenges for the analysis.

Analyses on state correlation are able to help system integrators to find out potential hazards without ignoring system dynamic behaviors. However, there are few research work considering components' states of IMA systems.

State correlation problems still exist, and there are few methods to solve it because traditional methods ignore strategies of IMA mentioned (resource sharing/fault recovery) and constraint satisfaction problems. Besides, unreasonable configuration [9] could cause some components faults, which can influence components' states. Thus, constraint satisfaction problems for configuration should be considered with multi-aspect for state correlation. Giuseppe Montano [10] proposed a constraints-based ontology in his thesis. But his ontology only mentioned the concept of some constraints without a specific description. One of motivations of this paper is to solve the problem. Traditional analysis methods [11] such as Fault Tree Analysis (FTA) [12], Functional Hazard Analysis (FHA), Failure Mode and Effect Analysis (FMEA), Event Tree Analysis (ETA) and Common Cause Failure Analysis (CCFA) [13] can only deal with static analysis for systems. These methods have limited ability on analyzing the state correlations because details of systems are ignored, such as strategies of partitions operation, and they are hard to be employed to describe system runtime environment as well. Beside these traditional analysis methods, there are some studies trying to solve other problems considering systems states. For example, Wang Yun-Sheng [14] divided states for partitions when analyzing the reliability for software partitions of IMA with the stochastic petri nets. Dajiang Suo [15] also has done the same work when checking the real-time and logical properties of reconfiguration in IMA. However, Wang Yun-Sheng did not take the scheduling mechanism into consider and Dajiang Suo only verified real-time and other logical attributes.

To deal with the state correlation problem, this paper proposes a novel method with an Improved Colored Generalized Stochastic Petri Net (ICGSPN), where Multi-constraint is added to solve the constraint satisfying problem for configuration verification.

The rest of the paper is organized as follows. An overview of the petri net and IMA is presented in section 2. Then, a new multi-constant analysis method for state correlation problem is proposed in section 3, including multi-constraint, improved CGSPN with its modelling and analysis method. In section 4, a case study about an aircraft radar system is conducted to demonstrate our method. Finally, section 5 make a conclusion of our work and some further work.

## II. BACKGROUND

The aim of this paper is to find out how the components' states of an IMA system influence one another under a condition of a configuration. Furthermore, the configuration can be verified with constraints proposed in this paper. To achieve our goal, petri nets are introduced as our simulation tools and IMA systems are introduced as our study objects.

### A. PN [17]

The concept of Petri Nets comes originally from Carl Adam Petri's dissertation in 1962. Generally speaking, Petri Nets are a family of formal notations to describe information processing systems which are featured as being distributed, concurrent, asynchronous, parallel nondeterministic, and stochastic. Both of the architecture and dynamic behaviors of systems can be modelled and analyzed with them. Due to these benefits of Petri Nets with the mature simulation technology, Petri Nets are chosen as a tool for this study. Marked Petri Nets and two extensions are presented as follows.

#### 1) MARKED PN [17]

There are three types of structure elements in marked Petri nets which are places, transitions and arcs. Places are used to describe system states, graphically shown as circles. And tokens in places represent the allocation of system resources. Transitions are represented as system events, graphically shown as rectangles. The arcs describe relationships of states and events of systems. Based on the 3 elements mentioned above, 2 functions are added into the marked petri net, which are the capacity functions and weighted functions. These functions bring the convenience of modelling for the real world systems. This marked Petri net is introduced with a 6-tuple [17]: PN $=$ (S, T, F, K, W, $M_O$). This marked Petri nets can describe system states and system events which can trigger state transition and limited information (tokens) of system states. Based on marked Petri nets, some other petri nets are developed with more elements for enhance description ability, such as time petri nets, stochastic petri nets and generalized stochastic petri nets.

#### 2) GSPN [18]

Time petri nets are introduced for analyzing systems with the property of time. In the time petri net models timed transitions are specified, which provide time delays of firing for transitions. However, when the time delays of transitions are stochastic, a stochastic petri net is needed, which provides flexibility and complexity for transitions. Generalized Stochastic Petri Nets (GSPNs) have immediate transitions based on stochastic petri nets. A GSPN is defined as an 8-tuple [18]: GSPN $=$ (P, T, $\Pi$, I, O, H,$M_O$, W). GSPN can make a more detailed description of a system with its elements mentioned in the tuple. However, tokens without color sets have the limit that they cannot describe complex properties of system states, such as the size and type of some resources. The Colored GSPN is a good option for this modelling purpose.

#### 3) CGSPN [18]

As to colored GSPN, color sets with tokens are combined with GSPNs, thus making it possible to fold the representation of systems without losing any key information. Although these advanced petri nets have no more stronger simulating ability for systems, they can simplify and clarify modeling for complex systems. Colored Generalized Stochastic Petri Nets (CGSPNs) were introduced by Chiola in 1988 with a 9-tuple [19]: CGSPN $=$ (P, T, C, P(.), I(.), O(.), H(.), W(.), $M_O$). Based on GSPN, CGSPN [20] can fulfill our needs for modelling:

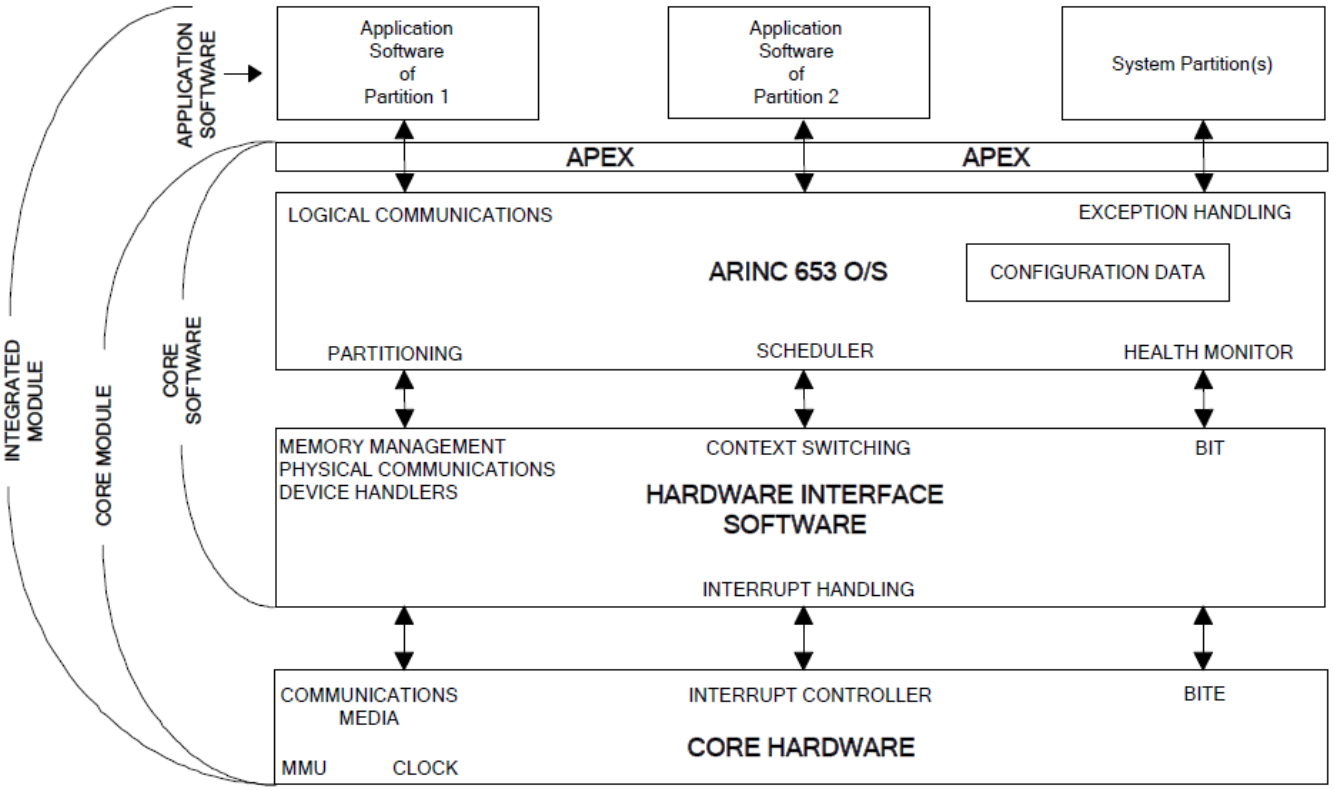



**FIGURE 1.** The architecture of ARINC 653 systems [22].

(1) Tokens with color sets help to describe system states with their multiple properties.

(2) The concept of system time is considered, such as scheduling and system events with or without time delays. Stochastic events can be described.

### B. A BRIEF INTRODUCTION ON IMA

The object of our work is the IMA system. In the following sections, firstly an overview of IMA architectures is taken. Then the resource sharing strategy as well as IMA fault recovering strategy are introduced as important contents, which will be modelled in the following works. In the end, IMA inter-partition configuration is shown as a basis for modelling.

#### 1) IMA SOFTWARE ARCHITECTURES

It is emphasized that the key of integration for IMA systems is the high-level resource sharing. Architectures of IMA implementations are mostly layered with interfaces between layers for communicating. Allied Standards Avionics Architecture Council (ASAAC) [21] standards, the Generic Open Architecture (GOA) framework and ARINC 653 [22] are common layered architectures for IMA systems. Typically they are based on the separated layer: the application layer, the operating system layer, the module support layer. Fig. 1 shows the architecture of ARINC 653 systems, which is a layered structure with interfaces between each layer.

**TABLE 1.** Shared resources.

| Shared hardware resources | Shared software resources |
|---|---|
| *Back plane* | Real time executive |
| *Power supply* | Built-in test |
| *CPU & Memory* | On-board maintenance system protocol |
| *Data bus* | I/O processing |
| *I/O* | Application |

As is shown in the Fig. 1, the applications are merged into an integrated system by resource sharing. ARINC 653 employs temporal and spatial partitioning, the scheduler, and the health monitor for the implementation of resource sharing mechanism. So the key features of IMA systems are:

(1) Resources of the IMA platform can be shared by multiple applications.

(2) The IMA platform provides robust partitioning strategies to maintain the system predictability and the controllability.

(3) Resources of IMA systems can be scheduled well and IMA components can recovery after failures with the help of the IMA configuration.

#### 2) IMA RESOURCE SHARING STRATEGIES

Under a typical IMA architecture, sharing resources include shared hardware resources and shared software resources,

**TABLE 2. Configuration tables.**

| Configuration tables | Partition configuration tables | Connection configuration tables | Shared resources configuration tables | Module schedule tables |
|---|---|---|---|---|
| Configuration items | Partition Name<br>Partition Identifier<br>Sampling Port<br>Queuing Port<br>Name Reference(Shared Data Region)<br>… | Channel Identifier<br>Channel Name<br>Source<br>Destination<br>… | Shared IO Size<br>Shared Data Size<br>Shared Data Region Name<br>… | Major Frame Seconds<br>Window Duration Seconds<br>… |

which are shown in Table 1. Besides, there are shared data resources as well in system, such as input data of sensors or data shared by different tasks. These resources can be accessed with the temporal and spatial partitioning mechanism, which separates applications of different partitions to prevent fault propagation and guarantee system reliability and safety.

The partitioning mechanism is implemented with a two-level hierarchy scheduling [23] method. The first level is within a partition time slot when tasks are running. And the tasks are scheduled based on their predefined priorities. The second level is a cyclic partition schedule that allocates time slots to partitions. All the partitions can only be executed in their allocated slots. In other words, no partition can preempt any other partition because it has no priority. This two-level scheduling method ensures that the temporal and spatial partitions are working properly. Temporal partitioning guarantees a partition's monopoly use of resources assigned in this partition time slot. While spatial partitioning physically separates resources owned by different partitions. In other words, one partition cannot access other partition's resources.

Although the partitioning mechanism guarantees the separation between applications, which makes behaviors of applications are much more robust, system functions always need communication between partitions. The communication makes states of system components influence one another. What's more, partitions always share resources for their specific needs, which makes states of shared resources influence states of partitions too.

### 3) IMA FAULT RECOVERING STRATEGY

Health monitor (HM) functions are responsible for the implementation of the IMA fault recovering strategy. They can report and handle hardware, application and O/S software faults and failures by providing HM configuration tables and an application level error handler process.

As shown in Fig. 2, the IMA fault recovering strategy handles errors by three levels: process level, partition level and module level. There are partition HM, multi-partition HM and module HM tables in accordance with these three levels. Thus, the recovery actions include actions at these levels. For example, due to the partition behavior capability, the partition can take the recovery action of restarting this partition, which is defined as the partition level error recovery actions. The decision logic defined in the HM configuration tables helps to isolate faults and prevent failures from propagating.

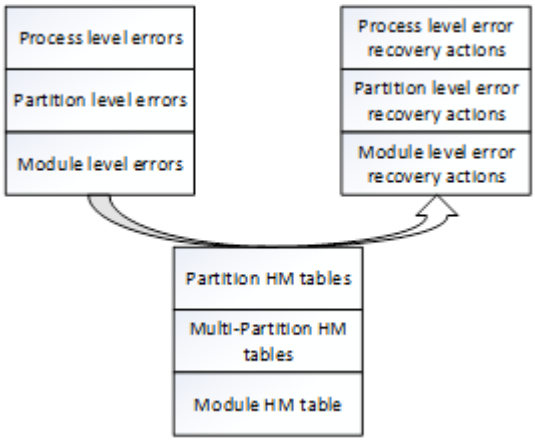



**FIGURE 2. Fault recovery strategy.**

### 4) IMA RECONFIGURATION

System configurations must satisfy the timing requirements, memory usage requirements, external interface requirements of partitions, and health monitor requirements. This paper focuses on the partition configuration tables, connection configuration tables, shared resources configuration tables as well as module schedule tables.

Table 2 shows the possible configuration items in configuration tables. Partition configuration includes partition name, sampling ports and queuing ports of this partition, shared data region names to which this partition accesses, etc. And other configuration items can be seen in this table.

In this paper, the IMA system models are built based on the IMA configuration.

## III. MULTI-CONSTRAINT

Configuration and reconfiguration of IMA systems are about "assembling" an artefact [24] well. While IMA system integrators "assemble" this artefact, available resources and function requirements must be taken into consideration. Both of the pre-defined functional and non-functional

requirements as well as the preferences of the operator should be met. In other words, the verification of configuration files is a kind of constraint programming problem.

As is mentioned above, the causes of component fault state of an IMA system are component faults and the constraint violation. It is assumed that the reliability of components has been obtained. Thus, the factor on which we focus is the constraint violation, which is also known as a constraint satisfaction problem.

There are three types of constraints in the process of analyzing the state correlation in IMA systems: resource constraints, functional constraints and dynamic constraint. Definitions are dedicated in the following.

(1) Resources studied in this paper are the sharing software and data resources. The concept of time is considered in the resource constraints. At a certain time, the allocation of a shared resource must satisfy system functional or non-functional requirements. Typical resource constraints are the constraints of memory consumption, bandwidth consumption and resource real-time requirements.

Definition 1 A resource constraint is defined by the expression:

$$R_{ti} \geq U_{ti} \tag{1}$$

Where:

$$1) \quad R_{ti} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ . \\ . \\ . \\ r_m \end{pmatrix} @t_i \tag{2}$$

$R_{ti}$ is the set of system resources' size at a given time. $r_1$ is the i-th resource. $@t_i$ is the element of time.

$$2) \quad U_{ti} = \begin{pmatrix} req_1 \\ req_2 \\ req_3 \\ . \\ . \\ . \\ req_n \end{pmatrix} @t_i = \begin{pmatrix} req_1 @t_i \\ req_2 @t_i \\ req_3 @t_i \\ . \\ . \\ . \\ req_n @t_i \end{pmatrix} \tag{3}$$

$U_{ti}$ is the set of requirements of system resource users at a given time. $req_i$ is the requirement for the i-th resource user. $@t_i$ is the element of time.

$$3) \quad req_k @t_i = \begin{pmatrix} U_{1k} \\ U_{2k} \\ U_{3k} \\ . \\ . \\ . \\ U_{nk} \end{pmatrix} @t_i = \begin{pmatrix} U_{1k} @t_i \\ U_{2k} @t_i \\ U_{3k} @t_i \\ . \\ . \\ . \\ U_{nk} @t_i \end{pmatrix} \tag{4}$$

$req_k @t_i$ is the requirement of the i-th resource user at a given time. $U_{jk}$ is the j-th resource user's requirement for the size of the k-th resource.

(2) A function is a process or task performed by an IMA system, which can be accomplished by more than one software application or partition. It can be implemented by allocating and managing system resources. Therefore, a function constraint is associated with resource constraints and system schedule. Typical function constraints are the constraints of task execution sequences and the application execution frequency.

Definition 2 A function constraint is defined by a Boolean expression:

$$function() = true \tag{5}$$

Where:

$$function() \rightarrow (R_{ti} \geq U_{ti}) \oplus schedule \tag{6}$$

$function()$ is a set of system functions. $R_{ti} \geq U_{ti}$ is the resource constraint and *schedule* is system scheduling. Eq. (6) means that the function constraints are related to resource constraints and system scheduling.

(3) A System event cannot only trigger the reconfiguration, but they can affect resource constraints or function constraints. This event can be an operator request for an operating mode change or a component fault. The change of resource constraints or function constraints caused by a system event is called as a dynamic constraint. The dynamic constraint contributes to the state correlation analysis method without ignoring the dynamic property of an IMA system.

Definition 3 A dynamic constraint is defined as a Boolean expression:

$$dynamic(R_{ti} \geq U_{ti}) \ \& \ dynamic(function()) = ture \tag{7}$$

Where:

$dynamic(R_{ti} \geq U_{ti})$ is the change of resource constraints and $dynamic(function())$ is the change of function constraints.

These three kinds of constraints can help to check component states of IMA systems, which will be used in the following sections.

## IV. A NEW MULTI-CONSTANT ANALYSIS METHOD FOR STATE CORRELATION

Depending upon resources accessibility or required functionality, it is allowed of the possibility of different configurations to be used due to the flexible nature of IMA. With the development of reconfiguration technology, it is possible to support the large space of configurations. This paper is dedicated to analyzing the state correlation of an IMA system on the basis of its configuration, which means that the reconfigurable IMA systems can also be suitable for the method. The input of this model is IMA configurations.

The purpose of this paper is to introduce a dramatic simulation method to verify the final system component states under the conditions of a kind of configuration. In this paper, the causes of fault states are divided into 2 categories: component faults and constraint violations. As is shown in Fig.3, component faults or constraint violations can make a component under fault state. Furthermore, it can cause system state

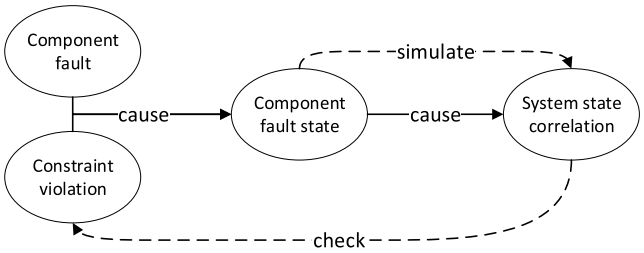



**FIGURE 3.** State correlation analysis.

correlation at system runtime. In this section, a model based analysis method is proposed to analyze state correlation. After the model is built, the analysis is processed though the model simulation. Besides, the constraint violations can be checked out as well. This section is organized as follows. Modelling method based on ICGSPN is firstly presented with the details of constraints. After that, the analysis process is described step by step to finish state correlation analysis, during which the fault condition can be addressed by the relationship of the fault states.

### A. AN IMPROVED CGSPN (ICGSPN)

As an advanced petri net, CGSPN has many advantages to simulate systems with dynamic behaviors. However, there are some limitation for state correlation analysis. In this section, some improvements based on CGSPN are proposed for specifying IMA resource sharing and fault recovering strategies. These improved CGSPN can also provide an approach to deal with the constraint satisfying problems with adding the constraints into ICGSPN models, besides, a system event could be described as an element in ICGSPN models.

System dynamic behaviors are related to system time. In ICGSPN models, the time attribute is bond with three elements: tokens with time stamps, arcs with time function and transitions with time function. Each token has a color and time stamp. Colored tokens can contain multiple attributes of system component. Arc expression functions control the change of their related places' tokens. And transitions cannot only describe events with or without timing intervals, but also control firing though the guarding function being true. In addition, there may be a firing probability for a transition.

The definition of ICGSPN is given as follows.

Definition 4 ICGSPN is a 10-tuple:

$$ICGSPN = (P, T, C, \Pi(.), I(.), O(.), H(.), W(.), M_0, G(.)) \tag{8}$$

Where:

P: the finite set of places;

T: the finite set of transitions, which includes timed and immediate transitions. $P \cap T = \emptyset$, $P \cup T \neq \emptyset$.

C: color classes, $C = \{C_1, \ldots, C_n\}$ and $C_i \cap C_j = \emptyset$.

$\Pi : T \rightarrow N$ associates priorities with transitions, mapping transitions into natural numbers as a priority function.

I, O, H: they are the presentation of input, output and inhibitor functions respectively labeled on the input, output and inhibitor arcs. Variables of these functions are bound to values from color sets. Besides, output arc functions are allowed to specify time delays by adding the at-sign (@).

W: W is a function defined on the set of transitions. The function is either a weight function for firing probability (weight) of an immediate transition or a distribution function for the firing delay (rate) of a timed transition.

M0: M0 is the initial marking describing the initial state of the system. (i.e. , the initial number of the tokens in places).

G: G is a guarding function, which can control the firing of a transition. Each transition t is mapped to an expression with multiple variables, which are bound to values from color sets. A transition may be enabled if its guarding function returns a true value.

The improvements of ICGSPN are:

(1) O in ICGSPN allows the output arc appending a specified time delay by adding the at-sign (@).

(2) W in ICGSPN is different from CGSPN. W in CGSPN is called as rate or weight. Rates are for timed transitions and weights are for immediate transitions. Rates in CGSPN define the exponential distributions of the delays associated with transitions, while rates in ICGSPN define all distributions of the delays associated with transitions. Weights in CGSPN are used for the probabilistic resolution of conflicts of immediate, while weights in ICGSPN are used for firing probabilities of immediate transitions.

(3) As a Boolean function, G can control the firing of a transition. In other words, firing of a transition can be enabled if and only if this transition's guarding function is satisfied: G = TURE.

With the proposed ICGSPN, the firing rules are changed. If and only if these conditions are satisfied, transitions can be enabled:

(1) Each input place p of a transition t is marked with at least w(p, t) colored tokens, where w(p, t) is the weight of the arc from p to t.

(2) The time of a time stamp should be equal to current model time. A time stamp is added to a color set by appending the keyword "timed" to its declaration, which is used to measure time in integer or reals.

(3) Guarding functions should be satisfied.

(4) A transition should be enabled in a given probability.

The execution of ICGSPN is time driven with the improvement proposed above. Hence we can simulate systems without ignoring their dynamic behaviors. Stochastic of system is also considered by specify W of ICGSPN. Color sets help to describe attributes of systems. While arc functions, guarding functions and the topology of places, transitions and arcs help to describe the logic of systems. In the following, ICGSPN is applied to modeling and simulating.

### B. THE MODELING PROCESS BASED ON ICGSPN WITH MULTI-CONSTRAINT

Based on the IMA configurations, the ICGSPN models are built, where resource sharing and fault recovering strategies

are taken into consideration. In addition, the multi-constraint is added into ICGSPN models.

As is mentioned above, the resource sharing strategies include the partitioning mechanism and communication behaviors. Furthermore, the partitioning mechanism is about scheduling which decides resource allocation, and the communication behaviors are about resource sharing and interoperation. Thus states of resources and resource users can be influenced by the resource sharing mechanism. In general, resource users are partitions. Except the resource sharing strategy, the fault recovering strategies directly affect the states of resources and partitions. For example, a partition can change from a fault state into another normal state if it has a successful recovering action.

The following rules can facilitate IMA systems models with ICGSPN elements:

(1) Places present component states, processors and notations for showing the fault partitions. Component states are divided into two parts: partition states and shared resource states. Partition states consist of initial, block, running, fault and complete states. Besides, the state schedule presents the partitions waiting for CPU to be scheduled. Shared resource states consist of normal and fault states. Processors are exclusive because only one partition is running for a single processor. Except for component states and processors, notations are defined as places to show the simulation results, which are the fault partitions at specific time.

(2) Transitions present system events, including expected and unexpected events with or without time delays. System events are divided into partition events, shared resource events, events that trigger dynamic constraints changing and aids for finishing the models' logic. Partition events consist of scheduling, ready, execute, and fail, recovery and over events. As a partition event, scheduling is managed with task scheduling algorithms. The event ready means a partition accesses to a shared resource. And the events execute and over are added into ICGSPN models to present the execution and finish for a task of a partition. The events fail and recovery mean partition faults and fault recovering. Shared resource events consist of fail and recovery events, which can make the transitions between normal and fault states. Additionally, the event that triggers dynamic constraints changing is presented as a transition in the ICGSPN models. Besides, to finish the logic of an IMA system, some extra transitions should be added as assistance.

(3) Tokens in places are bonded with color sets, which represent system attributes: resource types, size and so on. Besides, tokens can show states of partitions and resources.

(4) Arcs connecting places and transitions represent flow directions of tokens. With an arc expression, the quantity of the place's tokens is controlled.

(5) A guarding function has a bond with a transition, which can be used as a constraint for verifying IMA configurations. Besides, there is another usage for guarding transition. With the help of current system time function time (), the firing distribution of a transition is defined by a guarding function.

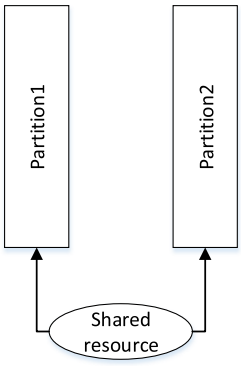



**FIGURE 4.** A simple system architecture.

For example, time ()>exponential (0.5) means the firing rate is 0.5.

To demonstrate how to build an ICGSPN for the state correlation analysis, a simple configuration is employed as an example. As Fig. 4 illustrates, this IMA architecture consists of one module with two partitions. Each partition only have one application, and the two partition work together as a function. The two partitions run on a processor sharing a resource, and they are scheduled on statically predefined time slots within a major time frame. Each partition has its offset and duration within this major time frame. The requirements for this shared resource of each partition and the available size of this shared resource can be acquired by the configuration file. A system event E is predefined to trigger the dynamic constraint changing. The notations size1 and size2 in Table 3 represent partition requirements and available size of the shared resource mentioned above. If the system event E occurs, size2 is the parameters of partition requirements and available size of the shared resource, otherwise size1 is the parameters. Details are shown in Table 3 and Table 4. Table 5 shows system events with their distributions.

At the beginning of this modelling process, multi-constraint should be instantiated as follows:

**TABLE 3.** Resource allocation.

| Component | Size 1 | Size 2 |
|---|---|---|
| P1 / MB | 12 | 12 |
| P2 / MB | 4 | 4 |
| Shared resource / MB | 16 | $\frac{size1}{2}$ |

**TABLE 4.** Partition scheduling information.

| Partition | $P_1$ | $P_2$ |
|---|---|---|
| Offset / ms | 0 | 10 |
| *Duration / ms* | 10 | 20 |

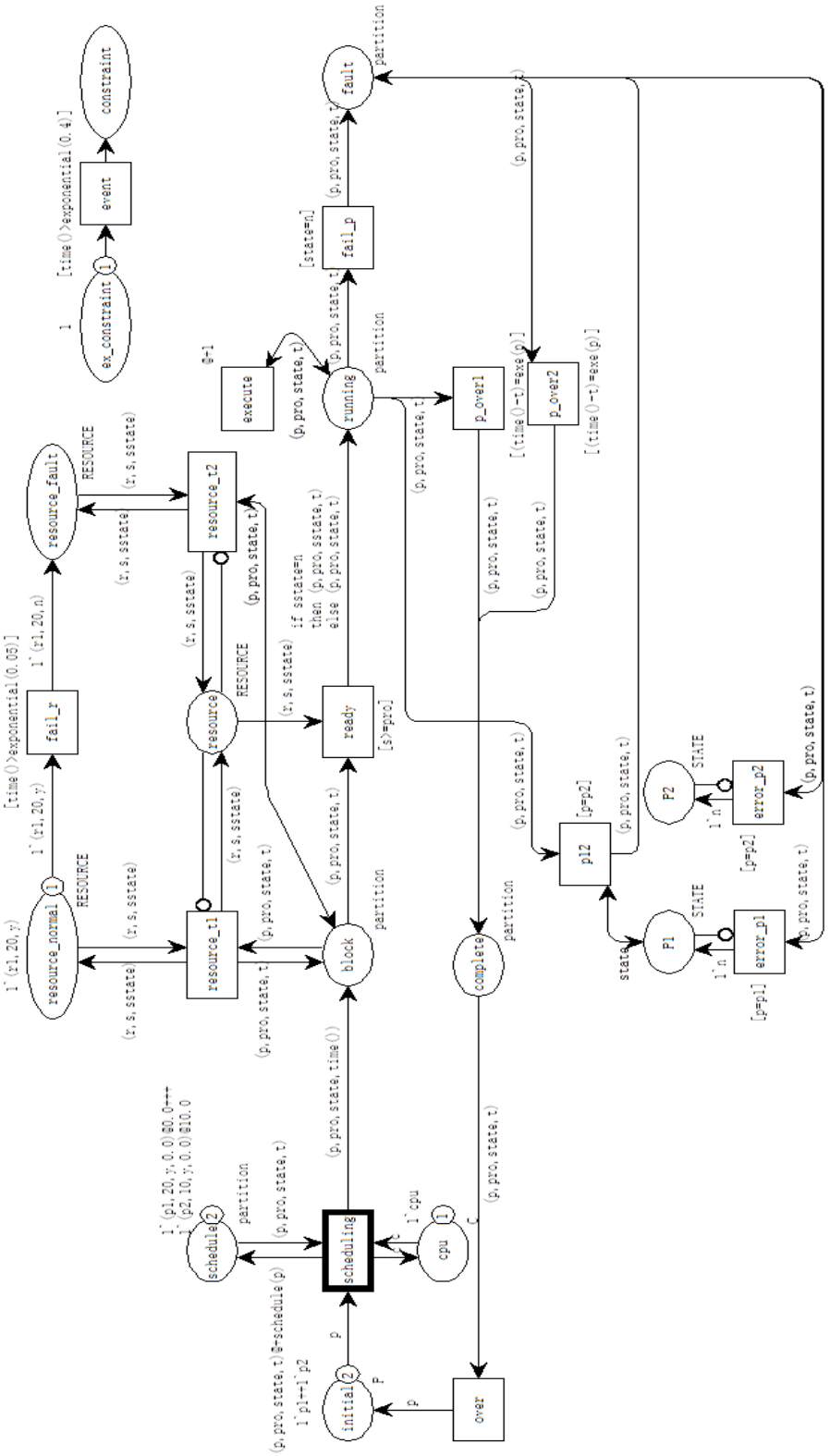



**FIGURE 5.** An example of ICGSPN model.

*Constraint 1:* Resource constraint:

$$size\_sr \geq \begin{pmatrix} size\_P1 \\ size\_P2 \end{pmatrix} @ti \tag{9}$$

*size_sr* is the available size of this shared resource, and *size_P*1, *size_P*2 represent the required size of partitions P1 and P2. *@ti* means the shared resource is accessed by partitions at system runtime.

**TABLE 5. System events with their distributions.**

| Event | Distribution of time when the event occurs/parameter |
|---|---|
| P1 fault | 0 |
| P2 fault | 0 |
| Shared resource fault | Exponential/0.05 |
| Shared resource fault recovery | 0 |
| Event triggering dynamic constraint changing | Exponential/0.4 |

*Constraint 2:* Function constraint:

$$function() = sequence(P1, P2) = true \tag{10}$$

A function is realized by the sequential execution of partitions, where applications reside. This function constraint is a Boolean function.

*Constraint 3:* Dynamic constraint:

$$dynamic(size\_sr \geq \begin{pmatrix} size\_P1 \\ size\_P2 \end{pmatrix} @ti) = \frac{size\_sr}{2} \geq \begin{pmatrix} size\_P1 \\ size\_P2 \end{pmatrix} @ti \tag{11}$$

*dynamic*() means the change of constraint 2 and constraint 3. In this dynamic constraint, only constraint 1 changes when a system event occurs.

With the modelling rules and instantiated constraints, an ICGSPN model is built as following Fig. 5. The partitions have the states: initial, block, running, fault and complete, presented with ellipses. While the shared resource has the states: normal (resource_normal) and fault (resource_fault), presented with rectangles. Partitions may change into fault state by though the transition fail_p. And the shared resource may change into fault state though the transition fail_r (Its firing rate is 0.05 with a function: time()>exponential(0.05).). Scheduling is modelled with the help of places (CPU and schedule), a transition (scheduling) and arc expression. The communication between partitions is modelled as a transition p12, which means P1 send error data to P2. In addition, an event that triggers dynamic constraint changing is modelled as a transition event. The resource constraint is shown as a guarding function (s >= pro) bonding with the transition ready. Finally, partitions fault states can be figured out by observing tokens in places P1 and P2.

Though model simulation, two results are straightforward:

(1) System states can be assured. Which component is in fault state can be find out though model simulation.

(2) IMA system configurations can be verified. If the configuration is not designed reasonably by integrators, constraints cannot satisfied, which can be figure out though simulation.

## V. A CASE STUDY

### A. A CASE DESCRIPTION

In this section, a case study is conducted on an IMA system based on this proposed method. The utility of results from the analysis will be discussed in the end. This case is about an aircraft radar system in an IMA module with a device, which can help to assure accurate navigation and flight safety. As is shown in Fig. 6, there are one device and four partitions in the aircraft radar system. The device radar is shared by the partitions: P1, P2, P3 though the bus. And communication is depicted in Fig. 6. P1 sends image processing data to partition P2, and P3 sends target detection processing data to partition P4.

After the introduction of this architecture, the configuration and other information of this aircraft radar system are detailed in the following tables: Table 6, Table 7 and Table 8. Now that the meanings of elements in tables have been discussed above, the introduction is omitted here.

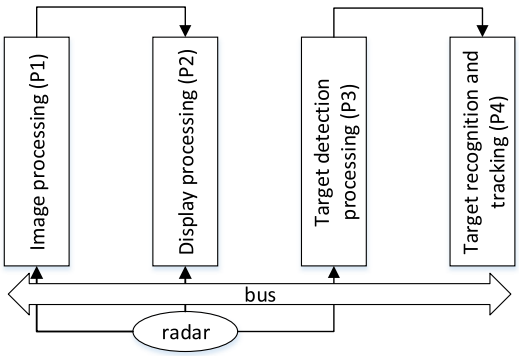



**FIGURE 6. The architecture of the aircraft radar system.**

**TABLE 6. Partition scheduling of the aircraft radar system.**

| Partition | P1 | P2 | P3 | P4 |
|---|---|---|---|---|
| Offset / ms | 0 | 10 | 30 | 40 |
| Duration / ms | 10 | 20 | 10 | 10 |

**TABLE 7. Resource allocation of the aircraft radar system.**

| Component | Size1 | Size2 |
|---|---|---|
| P1/MB | 4 | 4 |
| P2/MB | 2 | 2 |
| P3/MB | 8 | 8 |
| Shared resource/MB | 16 | (Size1)/4 |

**TABLE 8. System events with their distributions of the aircraft radar system.**

| Event | Distribution of time when the event occurs/parameter |
|---|---|
| P1/P2/P3/P4 fault | 0 |
| P2 fault recovering | Exponential/0.85 |
| Shared resource fault | Exponential/0.01 |
| Shared resource fault recovery | 0 |
| Event triggering dynamic constraint changing | Exponential/0.7 |

### B. MODELING AND SIMULATION ANALYSIS

With the system description above, multi-constraint is instantiated as follows:

*Constraint 1:* Resource constraint:

$$size\_sr \geq \begin{pmatrix} size\_P1 \\ size\_P2 \\ size\_P3 \end{pmatrix} @ti \tag{12}$$

$size\_sr$ is the available size provided by the radar, and $size\_P1$, $size\_P2$, $size\_P3$ represent the required size of partitions P1, P2 and P3. $@ti$ means the shared resource is accessed by partitions at system runtime.

*Constraint 2:* Function constraint:

$$function1() = sequence(P1, P2) = true \tag{13}$$

$$function2() = sequence(P2) = true \tag{14}$$

$$function3() = sequence(P3, P4) = true \tag{15}$$

There are three functions in the aircraft radar system. For example, function1 requires the sequential execution of P1 and P2.

*Constraint 3:* Dynamic constraint:

$$dynamic(size\_sr \geq \begin{pmatrix} size\_P1 \\ size\_P2 \\ size\_P3 \end{pmatrix} @ti) = \frac{size\_sr}{4} \geq \begin{pmatrix} size\_P1 \\ size\_P2 \\ size\_P3 \end{pmatrix} @ti \tag{16}$$

In this dynamic constraint, constraint 1 changes when a system event occurs.

Fig. 7 shows the ICGSPN model of this aircraft radar system, which is similar to Fig. 6. The declaration of this ICGSPN model is depicted in Table 9, where color sets (colset.), variables (var.), constants (val.) and functions (fun) are defined. Similarly there are five states for each partition and two states for the radar. This system's resource sharing strategy and fault recovering events have been considered in the model. Besides, the constraints are added into this model. Details are omitted as they have been discussed before.

Though the simulation, system states are ensured and the configuration is verified. In terms of system states, fault states of partitions are focused on, which are important for system safety analysis. As to the verification of the configuration, the constraint satisfying problems are focused on. Table 10 offers the notations for system events. For instance, we mark the radar fault event as the notation AY. Table 11 shows the simulation results. The results can be divided into two parts: partitions in fault states and the conditions whether the constraints are satisfied or not. It is worth notified that N (P3) in Table 11 means P3 doesn't satisfy the constraint 3, while Y means the opposite. Corresponding to the number of Table 11, dynamic behaviors are described in the following:

(1) When the system events are AN and CN, all the partitions work well. The simulation result of ICGSPN model is shown in Fig. 8.

**TABLE 9.** Declaration of the ICGSPN model.

```
colset C=with cpu; colset P=with p1|p2|p3|p4; colset STATE=with y|n;
colset partition=product P*int*STATE*real timed; colset R=with r1;
colset RESOURCE= product R*int*STATE timed;
var c:C;var p:P; var pr:P; var r:R;var t:real; var s:int; var state:STATE;
var sstate:STATE; var pro:int;
val t1=50; val t2=50; val t3=50; val t4=50; val T1=10.0; val T2=20.0;
val T3=10.0; val T4=10.0;
fun schedule(par) = case par of
p1=>t1|p2=>t2|p3=>t3|p4=>t4;
fun exe(p) = case p of
p1=>T1|p2=>T2|p3=>T3|p4=>T4;
```

**TABLE 10.** Resource allocation of the aircraft radar system.

| System event | Occur(notation) | Not occur(notation) |
|---|---|---|
| Radar fault | AY | AN |
| P2 fault recovering | BY | BN |
| Event triggering dynamic constraint changing | CY | CN |

**TABLE 11.** The simulation result.

| Number | System events | Partitions in the fault state | Whether satisfying the constraints or not (Y/N) | | | | |
|---|---|---|---|---|---|---|---|
| | | | Constraint1 | Constraint2 | | | Constraint3 |
| | | | | Function1 | Function2 | Function3 | |
| 1 | AN,CN | —— | Y | Y | Y | Y | Y |
| 2 | AY,BN,CN | P1,P2,P3,P4 | Y | N | N | N | Y |
| 3 | AY,BY,CN | P1,P3,P4 | Y | N | Y | N | Y |
| 4 | AN,CY | P3,P4 | N | Y | Y | N | N(P3) |
| 5 | AY,BN,CY | P1,P2,P3,P4 | N | N | N | N | N(P3) |
| 6 | AY,BY,CY | P1,P3,P4 | N | N | Y | N | N(P3) |

(2) When AY, BN and CN occur, partitions P1, P2 and P3 go into fault state in sequence. Finally, partition P4 gets into fault state as it communicates with partition P3. The simulation result is shown in Fig. 9.

(3) When AY, BY and CN occur, partition P1 and P3 go into fault state as they access to the radar. Partition P2 goes into fault state at the beginning of its time slot, then recovers with its successful recovery action. Partition P4 goes into fault state at the end of a major time frame. The simulation result is shown in Fig. 10.

(4) When AN and CY occur, P1 and P2 work well, while P3 goes into fault state because it doesn't satisfy constraint 3. P4 falls into fault state after P3 fails to work. The simulation result is shown in Fig. 11.

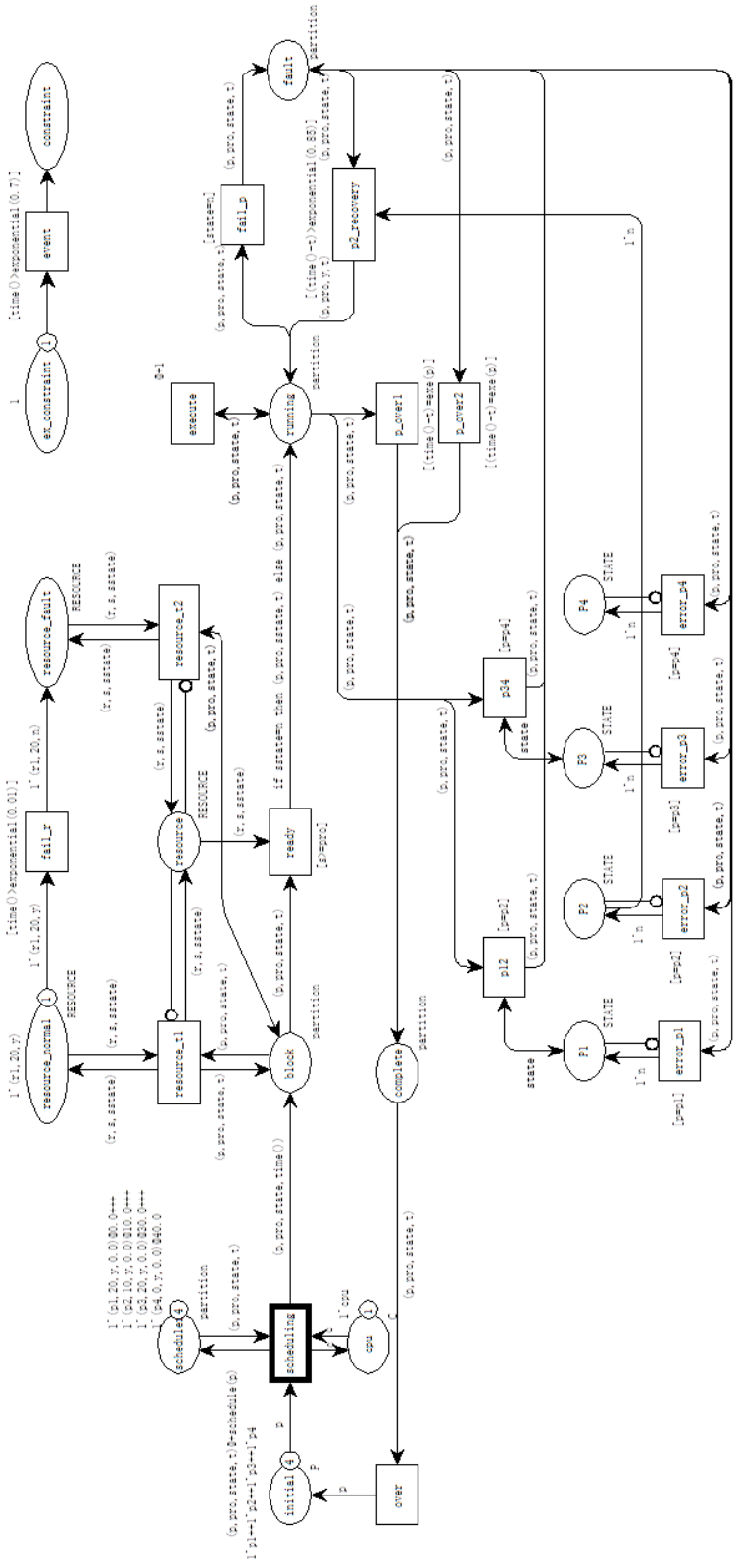



**FIGURE 7.** The ICGSPN model of the aircraft radar system.

(5) When AY, BN and CY occur, the result is the same as number 2. P1, P2, P3, P4 fall into fault state in sequence. The simulation result is shown in Fig. 12.

In conclusion, the state correlation analysis of aircraft radar system is finished with the result detailed in Table 11.

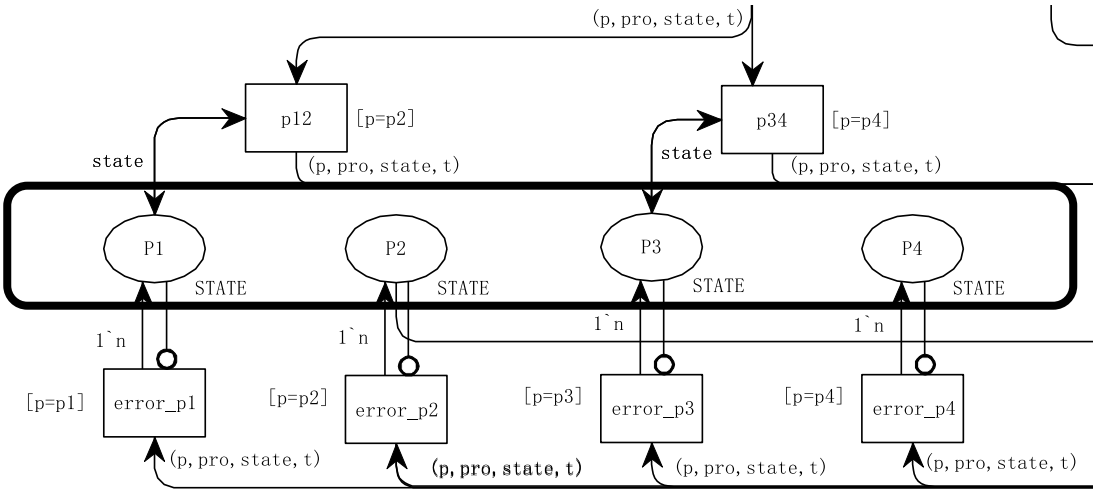



**FIGURE 8.** The simulation result (number 1).

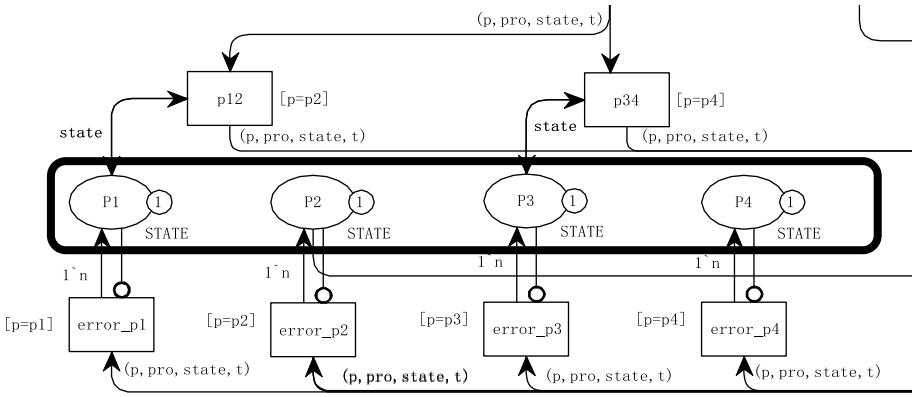



**FIGURE 9.** The simulation result (number 2).

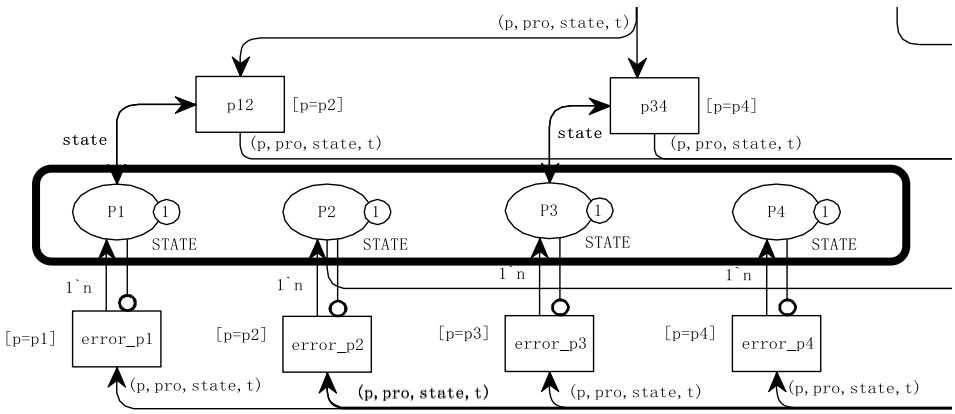



**FIGURE 10.** The simulation result (number 3).

### *C. DISCUSSION*

The following conclusions can be drawn with the simulation results:

(1) Compare to traditional methods: Other methods only tell us the module fails because it have the limit to analyze the partitions or scheduling strategies, while the dynamic analysis method in this paper tells us exactly which components go wrong.

(2) Shared resource states can influence states of partitions associated with the shared resource.

(3) States of partitions can influence one another through the inter-partition communication behaviors.

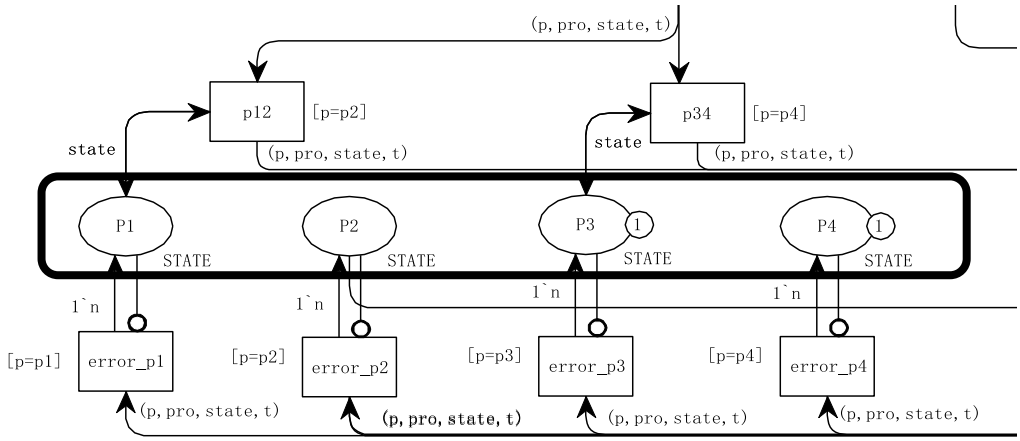



**FIGURE 11.** The simulation result (number 4).

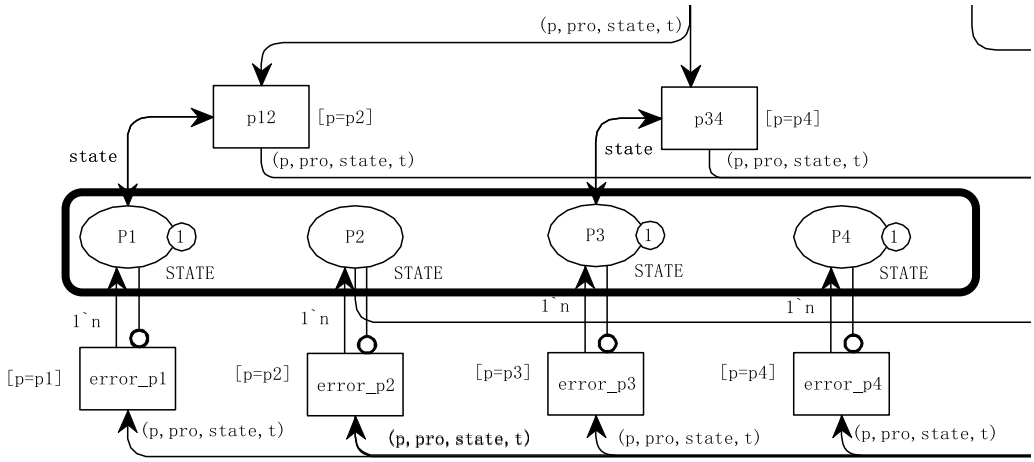



**FIGURE 12.** The simulation result (number 6).

(4) The configuration of this aircraft radar system satisfies the resource constraint if the event don't trigger dynamic change of constraints.

(5) The configuration doesn't satisfy the resource constraint as well as the dynamic constraint with the event mentioned in 3).

(6) Fault states of partitions affect the function constraints.

(7) Successful fault recovery actions are important for handling the abnormal events and unreasonable designs of system.

The following suggestions are offered for the design of this aircraft radar system:

(1) Reliability of shared resource must be increased as the shared resource is directly associated with three partition.

(2) An improvement of partition fault handling is needed, which can also improve the system robustness.

(3) The allocation of shared resource size should be more than 32 MB, figured out with the formula:

$$\frac{size\_sr}{4} \geq size\_P3$$

Where:

$$size\_P3 = 8MB$$

## VI. CONCLUSION AND FURTHER WORK

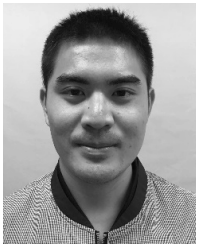
Current analysis methods have limits to ensure system states and verify configurations with considering system runtime behaviors. In this paper, we present a dynamic analysis method with state correlation taking the consideration of IMA fault recovery. In particular, architectures of IMA systems, the resource sharing and fault recovering strategies are studied. After that, three kinds of constraints are specified in the next step. Based on the basic and advanced petri nets, an ICGSPN is presented as the modelling tool for simulation. With the ICGSPN, the modelling method is conducted, including the constraint instantiation and modelling rules. In the end, state correlation and verification of configurations can be figured out through model simulating.

The method proposed by this paper solves the problem of analyzing IMA system state correlation dynamically. As the development of IMA systems, the complexity of system

architecture and behaviors, however, increases rapidly, which will bring the problem of state space explosion. And it cannot help to analyze for the multi-core system. In the future, we will enhance the ICGSPN models with a multi-layer structure of modelling with multi-core strategies for solving the problems. At the same time, further extension of mathematical definition in ICGSPN will be done so that more modelling elements concerning state correlation can be described. And more comprehensive state correlation analysis can be conducted afterwards.

## REFERENCES

[1] A. Mairaj, "Preferred choice for resource efficiency: Integrated Modular Avionics versus federated avionics," in *Proc. IEEE Aerosp. Conf.*, Mar. 2015, pp. 1–6.
[2] C. B. Watkins and R. Walter, "Transitioning from federated avionics architectures to Integrated Modular Avionics," in *Proc. IEEE/AIAA 26th Digit. Avionics Syst. Conf. (DASC)*, Dallas, TX, USA, Oct. 2007, pp. 2.A.1-1–2.A.1-10.
[3] G. R. Mettam and L. B. Adams, "How to prepare an electronic version of your article," in *Introduction to the Electronic Age*, B. S. Jones and R. Z. Smith, Eds. New York, NY, USA: E-Publishing, 1999, pp. 281–304.
[4] S. Jameson *et al.*, "Data fusion for the apache longbow: Implementation and experiences," Dept. Amer. Helicopter Soc. Tech. Rep. SKU: F61-2005-000331, 2005.
[5] *ASAAC Standards, Fault Management*, NASA, Jan. 2005, vol. 2.
[6] L. Sagaspe, G. Bel, P. Bieber, F. Boniol, and C. Castel, "Safe allocation of avionics shared resources," in *Proc. IEEE Int. Symp. High-Assurance Syst. Eng.*, Oct. 2005, pp. 25–33.
[7] P. Parkinson and L. Kinnan, "Safety-critical software development for integrated modular avionics," *Embedded Syst. Eng.*, vol. 11, no. 7, pp. 40–41, 2003.
[8] R. Hilbrich and K. R. van Kampenhout, "Dynamic reconfiguration in NoC-based MPSoCs in the avionics domain," *Proc. 3rd Int. Workshop Multicore Softw. Eng.*, 2010, pp. 56–57.
[9] E. Al-Shaer and S. Al-Haj, "FlowChecker: Configuration analysis and verification of federated OpenFlow infrastructures," in *Proc. 3rd ACM Workshop Assurable Usable Secur. Configurat.*, 2010, pp. 37–44.
[10] A. Ahmed, B. Kayis, and S. Amornsawadwatana, "A review of techniques for risk management in projects," *Benchmarking, Int. J.*, vol. 14, no. 1, pp. 22–36, 2007.
[11] S. A. Lapp and G. J. Powers, "Computer-aided synthesis of fault-trees," *IEEE Trans. Rel.*, vol. 26, no. 1, pp. 2–13, Apr. 1977.
[12] M. Ben-Daya, *Failure Mode and Effect Analysis*. London, U.K.: Springer, 2009, pp. 75–90.
[13] A. Shrestha and L. Xing, "Common-cause failure analysis for dynamic hierarchical systems," in *Proc. 21st Int. Conf. Adv. Inf. Netw. Appl. Workshops*, 2007, pp. 166–171.
[14] W. Yun-Sheng, L. Hang, and H. Xuan, "The stochastic Petri net based reliability analysis for software partition integrated modular avionics," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 30, no. 4, pp. 30–37, Apr. 2015.
[15] D. Suo, J. An, and J. Zhu, "AADL-based modeling and TPN-based verification of reconfiguration in integrated modular avionics," in *Proc. 18th Asia–Pacific Softw. Eng. Conf.*, 2011, pp. 266–273.
[16] G. Montano, "Dynamic reconfiguration of safety-critical systems: Automation and human involvement," Ph.D. dissertation, Univ. York, York, U.K., 2011.
[17] T. Murata, "Petri nets: Properties, analysis and application," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.
[18] J. Zhu, Q. Yang, W. Huang, and R. Lu, "A formal model of satellite communication system network control protocol based on generalized stochastic Petri nets," in *Proc. IEEE Int. Conf. Comput. Commun. (ICCC)*, Oct. 2015, pp. 340–346.
[19] P. Buchholz, "Hierarchies in colored GSPNs," in *Proc. Int. Conf. Appl. Theory Petri Nets.*, 1993, pp. 106–125.
[20] F. LIU, J. ZHAO, and B. GUO, "An extended CGSPN-based simulation method for the evaluation and resources optimization of maintenance support systems," in *Proc. Comput. Simulation*, vol. 5. 2005, p. 023.
[21] *ASAAC Standards, Fault Management*, NASA, Tech. Rep. DEFSTAN 00-78/1-2005, Jan. 2005, vol. 2.
[22] *Arinc Specification 653 Part 1, Required Services*, Airline Electron. Eng. Commission, Nov. 2010.
[23] J. Chen, C. Du, and P. Han, "Scheduling independent partitions in integrated modular avionics systems," *PLoS ONE*, vol. 11, no. 12, p. e0168064, 2016.
[24] S. Mittal and F. Frayman, "Towards a generic model of configuraton tasks," *IJCAI*, vol. 89, pp. 1395–1401, Aug. 1989.



**RONGBIN HAN** is currently pursuing the Master's degree with the Reliability and Systems Engineering Department, Beihang University.

**SHIHAI WANG** is currently a Lecturer with the Reliability and Systems Engineering Department, Beihang University. He also leads several Master's degree students at the school including the program on safety of software.

**BIN LIU** is currently a Professor with the Reliability and Systems Engineering Department, Beihang University.

**TINGDI ZHAO** is currently a Professor with the Reliability and Systems Engineering Department, Beihang University.

**ZHIAO YE** is currently pursuing the Master's degree with the Reliability and Systems Engineering Department, Beihang University.