Journal of Systems Engineering and Electronics
Vol. 34, No. 6, December 2023, pp.1589 – 1601

Formal management-specifying approach for model-based safety assessment

XU Changyi[1], DUAN Yiman[2], and ZHANG Chao[2,*]

1. School of Control Science and Engineering, Key Laboratory of Intelligent Control and Optimization for Industrial Equipment of Ministry of Education, Dalian University of Technology, Dalian 116024, China; 2. State Key Laboratory of Fluid Power and Mechatronic Systems, School of Mechanical Engineering, Zhejiang University, Hangzhou 310027, China

Abstract: In the field of model-based system assessment, mathematical models are used to interpret the system behaviors. However, the industrial systems in this intelligent era will be more manageable. Various management operations will be dynamically set, and the system will be no longer static as it is initially designed. Thus, the static model generated by the traditional model-based safety assessment (MBSA) approach cannot be used to accurately assess the dependability. There mainly exists three problems. Complex: huge and complex behaviors make the modeling to be trivial manual; Dynamic: though there are thousands of states and transitions, the previous model must be resubmitted to assess whenever new management arrives; Unreusable: as for different systems, the model must be resubmitted by reconsidering both the management and the system itself at the same time though the management is the same. Motivated by solving the above problems, this research studies a formal management specifying approach with the advantages of agility modeling, dynamic modeling, and specification design that can be re-suable. Finally, three typical managements are specified in a series-parallel system as a demonstration to show the potential.

Keywords: model-based safety assessment (MBSA), management, availability, reliability, maintainability, continuous time Markov chain.

DOI: 10.23919/JSEE.2023.000154

1. Introduction

Model-based safety assessment (MBSA) [1,2] is an indispensable branch of system engineering [3]. As for a system design document, MBSA merges safety assessment models, such as the continuous time Markov chain (CTMC). They consider the complex behaviors of the system [4,5], and handle the safety requirements verification [6,7], time indicator calculation, etc. [8,9]. Normally, the assessment result helps to resubmit an improvement of the dependability of the system's design. However, the system will not always be static as it is initially designed during its whole life. Various management requirements, such as downtime maintenance, are dynamically set for the system to optimize the system's service [10,11]. These management requirements will always affect the system's dependability. Although faulty behaviors can be successfully considered in traditional manual approaches, there is still a lack of a formal method to stochastically specify the various managements into the models.

Model generation depends on the skill of the system analyst, and the challenges are as follows.

(i) Complexity specifying: if the system's scale and behavior complexity increase, the management modeling task becomes trivial and the assessment accuracy decreases [12–14].

(ii) Dynamically specifying: as the management is dynamically set for the system, the models should be resubmitted whenever a new management requirement arrives. It is not possible to employ a group of analysts responsible for this work during the whole lifetime of the system [15,16].

(iii) Multiply specifying: the management specification is not reusable. Although the same requirement is applied, the safety assessment model must be resubmitted for different systems, regardless of the scalability and complexity of the system [17,18].

However, the existing MBSA framework cannot handle formal, automatic, dynamic, and general modeling solutions to overcome the abovementioned challenges.

Manuscript received November 04, 2022.
*Corresponding author.
This work was supported by the National Natural Science Foundation of China (52105070; U21B2074) and Department of Science and Technology of Liaoning Province China (2033JH1/10400007).

Graphical interface for reliability forecasting (GRIF) [19] has some basic packages for the dependability model generation, such as the Markov chain package and fault tree package. In GRIF, all the management-specified behaviors should be established manually. Management specifications are far from formalized and dynamic. The Boolean-driven Markov process (BDMP) [20] uses the tree models to automatically generate the Markov model. The operator establishes logic gates according to the structure and assigns the leaves according to the components' behaviors. Particularly, the triggers in its model are applied to denote the happening sequence of the components, which is similar to the destination of expressing management requirements. However, these triggers can only describe the limited failure-related operations, such as cold redundancy management (after the main component breaks down, the expletive component starts to work). It is unable to model repairing-related management. AltaRica [21] is another normally used modeling tool, in which the design document and verification task of the safety critical projects can be programmed. It has rich interfaces with system models such as system modeling language (SysML) [22], Event-B [23], and UPPAAL [24], and these interfaces strongly enhance its modeling ability. However, aiming at the management expression, AltaRica only handles it by manually detailed coding programs. This loses formalization and increases the complexity of model generation.

This study aims to achieve a formal management-specifying approach to assess the systems in the form of the CTMC model. The specifying approach is achieved by converting the CTMC into an event-driven automaton (EDA) and designing the specification automaton based on multiple management requirements. With the help of a safety assessment model synchronized with the specification automaton, the system assessment model will be generated.

The main advantage of this study is: there is no need to modify the previous CTMC assessment model due to that any new management requirement can be specified by further synchronizing its coherent specification automaton. In addition, the same management requirement specifying the task is solved by following the same establishing principle for different systems.

In this study, a formal management specifying approach with the advantages of agility modeling, dynamic modeling, and specification design is proposed to be used in a series-parallel system. Firstly, the background, big issues, and basic models are reported. Secondly, the formal specifying approach to solve the problem is used. Finally, a benchmark to further interpret the solution is applied. Through this paper, three widely applied system management requirements, "system downtime maintenance (DM)", "component first failed first repaired (FFFR)" and "special component has the priority to be repaired (PtR)", are specified in the availability, reliability and maintainability CTMC models in a series-parallel system.

1.1 CTMC model

As a stochastic process, the Markov process is named by the Russian mathematician Andrey Markov. System performances are described as variable states, and the transitions are used to illustrate the relationships among the states. The Markov process has the memoryless property: if the value in the current state is already known, the variable value in the next state only depends on this current state without recalling the system's history [25]. The discrete-time Markov chain and CTMC are two model branches of the Markov process. The discrete time Markov chain model is regarded as a system driven by a common time clock, and the transitions among the states are distributed by probability. In contrast, the CTMC considers the occurrence rate of various events, while these events contribute to state transit [26]. A CTMC describes a faulty/repairable system from the viewpoint of safety analysis. There are two states, an initial system working state and a system broken down state, and the transitions are driven by real time, denoted by failure rate and repair rate. If the CTMC is achievable, the equivalent transition matrix and state equation are achievable, which makes mathematical calculations possible [27,28]. As the CTMC is naturally suitable for describing the systems, and its calculating function can be used for the assessment task, the CTMC is a widely applied modeling approach in model-based safety analysis. With the help of the Laplace solution, the CTMC assesses the system by calculating time indicators: for availability CTMC $A(t)$, it calculates the mean time before failure (MTBF); for reliability CTMC $R(t)$, it calculates the mean time to failure (MTTF); for maintainability CTMC $M(t)$, it calculates the mean time to repair (MTTR) [29–31]. The assumption of the CTMC to transition matrix is: (i) the rate is the happening rate of the transition; (ii) the sum of the happening rate xi should be "1", $\sum_{k=0}^{n} \mathrm{xi} = 1$, which means all the possibility is considered.

1.2 Availability CTMC model

Both the failure and repair system characteristics are con-

sidered in availability model. The definition of the availability function is denoted by $A(t)$; in a certain situation, the rate of the system contributes its function at a certain time t. The availability CTMC model presents all the possible functional and dysfunctional situations by state, while the model transition presents the possible failure rates and repair rates [32].

State 1 is the initial state, which means the system is initially working healthily. And, by failure rate "λ", the system breaks down to state 2. From state 2 to state 1, the system can recover by a successfully repaired rate "μ".

1.3 Reliability CTMC model

The performance during healthy working to breaking down is depicted in the reliability model. Reliability is defined to be in a certain situation, and in a certain period $(0, T)$, the system achieves its desired function. At time $t=0$, the system is working completely healthily, and at time $t=T$, total breakdown occurs in the system. In the meantime, in $(0, T)$, the rate of the system ability to achieve its function is denoted by $R(t)$ and $0 \leqslant R(t) \leqslant 1$ [32].

The model starts from state 1 (healthy state); by a failure rate of λ, the system breaks down to state 2, and then, this model ends.

1.4 Maintainability CTMC model

The characteristics from breaking down to finally healthy operation are depicted in the maintainability model. The definition of the maintainability function, denoted by $M(t)$, is the system successfully recovering its rate in a certain situation in a demanded time period $(0, T)$. After the failure occurring in this repairable system, as a quantification description of system reparability by rate, $M(t)$ is: at the beginning time $t=0$, the system is in the failure situation, and before $t=T$, the system recovers to a healthy function.

The initial state of the maintainability model is state 2 (describing the system starting from the brokendown state); by a successfully repaired rate "μ", the system reaches healthy state 1, and then, the model ends [32,33].

2. Management-specifying approach

This part studies the management-specifying approach. Subsection 2.1 generally introduces the workflow of this approach, and it mainly has two important steps: the first step is to convert CTMC to EDA, which will be presented in Subsection 2.2; the second step is to design the specification automaton to specify the MBSE models, which will be presented in Subsection 2.3.

2.1 Management-specifying proposal

The work of Ionescu enables the CTMC model to be transformed with EDA by regarding the occurrence rate of the transition events directly presented by the events [34]. Therefore, when only focusing on the events and the local component transition events are unique (although the component transition event rate values are probably the same, it is supposed that the events are also unique), the occurrence rate of the events in the CTMC can be equivalent to the transition events in EDA [35].

Based on the specifying solution in discrete event system theory [36,37], management can be modeled as specification automata. After the synchronization of EDA and specification automata, EDA is able to contain specifying information.

As it is depicted in Fig. 1, this specifying solution is

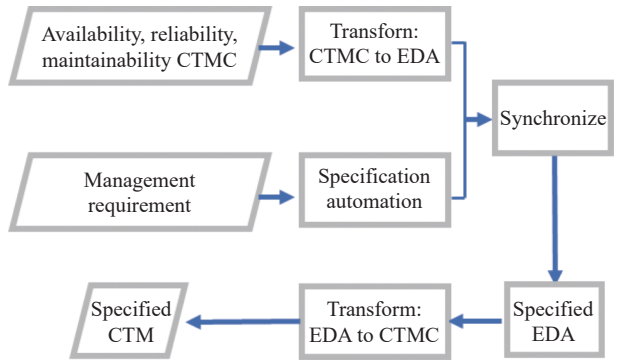


Fig. 1 Management requirements specifying the algorithm

(i) Based on the system structure and the component behaviors, the availability, reliability, and maintainability CTMC models are established as normal.

(ii) These CTMC models are translated into EDA.

(iii) According to the management, the specification automata are designed. The synchronize operation of EDA with specification automata produces the specified safety EDA models.

(iv) The specified EDAs are translated back into CTMCs for the purpose of assessment.

2.2 Equivalence model conversion: CTMC and EDA

Explained in (1) and (2), the system behavior is described as a discrete language $\boldsymbol{L}_{\text{sys}}$. $\boldsymbol{L}_{\text{sys}}$ can be regarded as the union of sublanguages associated with every state, and sublanguage $\boldsymbol{L}_i$ associated with state xi is defined as the universe of event sequences, while the events drive the system from the initial state to the considered state xi:

$$\boldsymbol{L}_{\text{sys}} = \underset{\text{xi}}{\cup} \boldsymbol{L}_i. \tag{1}$$

Based on Arden's rule [38,39], the CTMC can be presented by a sequence of events instead of the event occurrence rates. In (2), $\boldsymbol{A}$ and $\boldsymbol{B}$ are the universe set of event sequences, and the sublanguage $\boldsymbol{L}_i$ is as follows:

$$\boldsymbol{L}_i = \boldsymbol{L}_i \boldsymbol{A} + \boldsymbol{B}. \tag{2}$$

If $\boldsymbol{\varepsilon} \notin \boldsymbol{A}$, (2) has a unique solution: $\boldsymbol{L}_i=\boldsymbol{B}\boldsymbol{A}^*$, $(\cdot)^*$ is adjugate matrix. If $\boldsymbol{\varepsilon} \in \boldsymbol{A}$, based on (2), the solution is $\boldsymbol{L}_i=(\boldsymbol{B}+\boldsymbol{C})\boldsymbol{A}^*$, where $\boldsymbol{\varepsilon}$ is the empty set, $\boldsymbol{\Psi}$ is the event set and $\boldsymbol{C} \subseteq \boldsymbol{\Psi}^*$.

The CTMC is defined by the following:

$$\text{CTMC} = (Q_{\text{mc}};\ E_{\text{mc}};\ \delta_{\text{mc}};\ q_{0\text{mc}}) \tag{3}$$

where Q_{mc} is the state set, E_{mc} is the transition set, δ_{mc} is the transition function explaining the state evolution relationship excited by the transitions, and $q_{0\text{mc}}$ is the initial state. Based on CTMC definition and the system can be presented by the union of sequences of event-associated states, the states xi union in (1) is equivalent to the state set in the CTMC:

$$\{\text{xi}\} = Q_{\text{mc}}. \tag{4}$$

The transition function in the CTMC describes the transition relationship between states and focuses on the events causing the state transition. The transition function δ_{mc} in the CTMC is equivalent to the system describing language $\boldsymbol{L}_{\text{sys}}$:

$$\delta_{\text{mc}} \leftrightarrow \underset{\text{xi}}{U} \boldsymbol{L}_i. \tag{5}$$

The EDA is defined as follows:

$$\text{EDA} = (Q_{\text{EDA}}; E_{\text{EDA}}; \delta_{\text{EDA}}; q_{0\text{EDA}}; q_{m\text{EDA}}) \tag{6}$$

where Q_{EDA} is the state set, E_{EDA} is the transition set, δ_{EDA} is the transition function, $q_{0\text{EDA}}$ is the initial state, and $q_{m\text{EDA}}$ is the marked state. By the same principle, according to (1) and (2), we can also obtain the state set:

$$\{\text{xi}\} = Q_{\text{EDA}}. \tag{7}$$

Additionally, the transition function is described by the system describing language:

$$\delta_{\text{EDA}} \leftrightarrow \underset{\text{xi}}{U} \boldsymbol{L}_i. \tag{8}$$

Assume e_{xy} is one of the system events, and it causes the state to evolve from one state x to another state y by the rate of R-e_{xy}. Because the CMTC and EDA describe the same system, the conclusion is as follows:

$$Q_{\text{EDA}} = \{\text{xi}\} = Q_{\text{mc}}, \tag{9}$$

$$\delta_{\text{mc}} \leftrightarrow \underset{\text{xi}}{U} \boldsymbol{L}_i \leftrightarrow \delta_{\text{EDA}} \tag{10}$$

which can be further explained as follows:

In EDA, $\delta_{\text{EDA}}(x,\ e_{xy}) = y$.

It is equal to the following:

In CTMC, $\delta_{\text{mc}} = (x,\ e_{xy}) = y$, and by the rate R-e_{xy}.

For example, in Fig. 2, the models describe repair with an initial working state S0 and a failure meaning state S1. The failure event (denoted by f_1) occurs at the rate λ, and the system successfully repairing operation (denoted by r_1) occurs at the rate μ. According to (1) and (2):

$$\begin{cases} \boldsymbol{L}_1 = \boldsymbol{L}_2\ \text{f}_1 \\ \boldsymbol{L}_2 = \boldsymbol{L}_1\ \text{r}_1 \end{cases} \tag{11}$$

where $\boldsymbol{L}_1 = (\boldsymbol{L}_1\ \text{r}_1)\text{f}_1$, and the solution is as follows:

$$\begin{cases} \boldsymbol{L}_1 = (\text{f}_1\text{r}_1)^* \\ \boldsymbol{L}_2 = (\text{f}_1\text{r}_1)^*\text{f}_1 \end{cases}. \tag{12}$$

Therefore, for the conversion of the CTMC and EDA, the events drive the state transition, and each event has an occurrence rate shown in the middle of Fig. 2; thus, the state set is as follows:

$$Q_{\text{mc}} = \{\text{S0},\ \text{S1}\} = Q_{\text{EDA}}. \tag{13}$$

In CTMC,

$$\begin{cases} f_{\text{mc}}(\text{S0}, \lambda_{\text{f}_1}) = \text{S1} \\ f_{\text{mc}}(\text{S1}, \mu_{\text{r}_1}) = \text{S0} \end{cases}.$$

In EDA, which is equal to the following:

$$\begin{cases} f_{\text{EDA}}(\text{S0}, \text{f}_1) = \text{S1},\ \text{by the happening rate } \lambda \\ f_{\text{EDA}}(\text{S1}, \text{r}_1) = \text{S0},\ \text{by the happening rate } \mu \end{cases}.$$

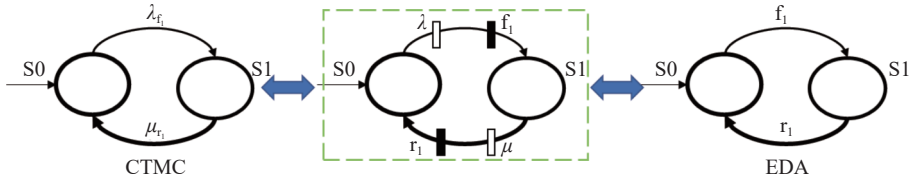


Fig. 2 Model conversion between CTMC and EDA

2.3 Management-specifying method by using an EDA synchronize operation

In the field of discrete event system theory application, the specification automaton is designed to contain the specifying information. With the synchronize solution of the EDA and specification automaton, management-specifying demand is achieved. The specification automaton is defined as follows:

$$\text{SP} = (Q_{\text{SP}}; E_{\text{SP}}; \delta_{\text{SP}}; q_{0\text{SP}}; q_{m\text{SP}}) \tag{14}$$

where Q_{SP} is the state set, E_{SP} is the transition set, δ_{SP} is

the transition function, q_{0SP} is the initial state, and q_{mSP} is the marked state.

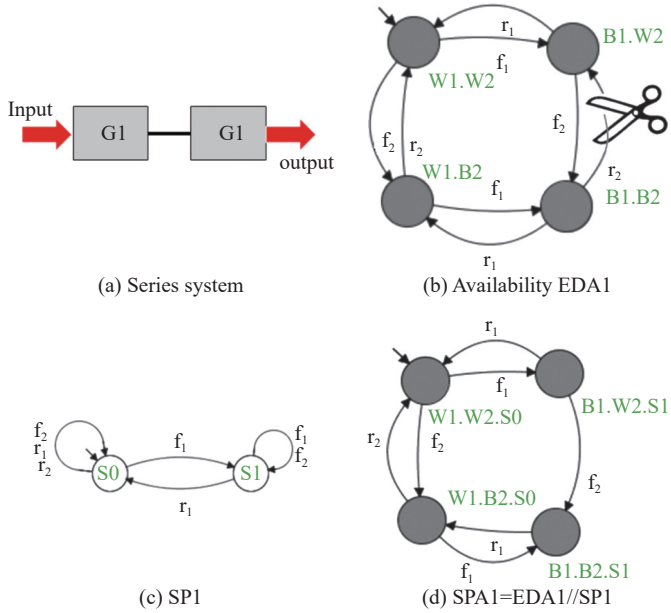
Then, the result of the synchronize operation (denoted "//") [37] is also an automaton, and this specified automaton (denoted "SPA") is defined as follows:

$$\begin{aligned} \mathrm{SPA} = (Q_{\mathrm{SPA}}; E_{\mathrm{SPA}}; \delta_{\mathrm{SPA}}; q_{0\mathrm{SPA}}; q_{m\mathrm{SPA}}) = \mathrm{EDA}//\mathrm{SP} = \\ \mathrm{Ac}(Q_{\mathrm{EDA}} \times Q_{\mathrm{SP}}; E_{\mathrm{EDA}} \cup E_{\mathrm{SP}}; \delta_{\mathrm{SPA}}; \\ q_{0\mathrm{EDA}} \times q_{0\mathrm{SP}};\ q_{m\mathrm{EDA}} \times q_{m\mathrm{SP}}) \end{aligned} \tag{15}$$

where Ac is the accessible part, and the symbol "×" denotes the Cartesian product [40,41] between the state set or transition event set.

The transition function of SPA is inspired by the transition function of SP and EDA, and it is explained by the following equation. Here, the symbol "\" denotes the complement set operation in (16).

Assume the following management requirement: among all the components in the system, the most important one owns the priority to be repaired firstly. Fig. 3 shows a series-structural system constructed by G1 and G2. When both of these components are broken down, G1 has the priority to be repaired first.



(a) Series system (b) Availability EDA1

(c) SP1 (d) SPA1=EDA1//SP1

Fig. 3 "G1 has the priority to repair" model specification

In Fig. 3, series system is the availability model of this system:

$$\begin{cases} \mathrm{EDA1} = (Q_{\mathrm{EDA1}}; E_{\mathrm{EDA1}}; \delta_{\mathrm{EDA1}}; q_{0\mathrm{EDA1}}; q_{m\mathrm{EDA1}}) \\ Q_{\mathrm{EDA1}} = \{\mathrm{W1.W2}, \mathrm{W1.B2}, \mathrm{B1.W2}, \mathrm{B1.B2}\} \end{cases}, \tag{16}$$

these states respectively mean: both G1 and G2 are working; G1 is working, but G2 is broken down; G1 is broken down, but G2 is working; and both G1 and G2 are broken down.

$$\delta_{\mathrm{SPA}}((x,y),e) = \begin{cases} (\delta_{\mathrm{EDA}}(x,e).\delta_{\mathrm{SP}}(y,e)), & e \in E_{\mathrm{EDA}} \cap E_{\mathrm{SP}} \\ (\delta_{\mathrm{EDA}}(x,e).y), & e \in E_{\mathrm{EDA}} \backslash E_{\mathrm{EDA}} \cap E_{\mathrm{SP}} \\ (x.\delta_{\mathrm{SP}}(y,e)), & e \in E_{\mathrm{SP}} \backslash E_{\mathrm{EDA}} \cap E_{\mathrm{SP}} \\ \text{undefined}, & \text{otherwise} \end{cases} \tag{17}$$

For $E_{\mathrm{EDA1}} = \{\mathrm{f_1}, \mathrm{f_2}, \mathrm{r_1}, \mathrm{r_2}\}$, these events represent: the failure event of G1, the failure event of G2, the repair operation of G1, and the repair operation of G2, respectively.

The transition function δ_{EDA1} that evolves the states is shown in Fig. 3. According to the management requirement, when both components are broken down (state B1.B2), the first repair operation should be distributed to G1, which means that transition $\mathrm{B1.B2} \xrightarrow{\mathrm{r_1}} \mathrm{W1.}B2$ is legal and transition $\mathrm{B1.B2} \xrightarrow{\mathrm{r_2}} W1.B2$ is illegal. Therefore, to express a management requirement, the illegal transition should be dismissed by a formal solution.

In Fig. 3, the specification automaton SP1 is

$$\mathrm{SP1} = (Q_{\mathrm{SP1}}; E_{\mathrm{SP1}}; \delta_{\mathrm{SP1}}; q_{0\mathrm{SP1}}; q_{m\mathrm{SP1}}) \tag{18}$$

where $Q_{\mathrm{SP1}} = \{\mathrm{S0}, \mathrm{S1}\}$; $E_{\mathrm{SP1}} = \{\mathrm{f_1}, \mathrm{f_2}, \mathrm{r_1}, \mathrm{r_2}\}$; $q_{0\mathrm{SP1}} = \{\mathrm{S0}\}$; and $q_{m\mathrm{SP1}} = \varnothing$. The transition function is as follows: the self-loop events of S0 are $\{\mathrm{f_2}, \mathrm{r_1}, \mathrm{r_2}\}$, driving no state evolution from S0 to S1, since these events do not break the management requirement. Then, the transition from state S0 to S1 is event $\{\mathrm{f_1}\}$; the statement "if $\mathrm{f_1}$ happens, the

requirement should begin to be respected" is considered; the self-loop of S1 is $\{f_1, f_2\}$, where there is no repair operation r_1 or r_2, since after f_1 happens there should be no repair operation distributed onto G2 and r_1 will be further treated; from S1 to S0 is repair operation r_1, since after f_1 happens G1 should be repaired and turns to the initial state.

The synchronize result of SP1 and EDA1 is

$$\begin{aligned} &\text{SPA1} = (Q_{\text{SPA1}}; E_{\text{SPA1}}; \delta_{\text{SPA1}}; q_{0\text{SPA1}}; q_{m\text{SPA1}}) = \\ &\text{EDA1}/\!/\ \text{SP1} = \\ &\text{Ac}(Q_{\text{EDA1}} \times Q_{\text{SP1}}; E_{\text{EDA1}} \cup E_{\text{SP1}}; \delta_{\text{SPA1}}; \\ &q_{0\text{EDA1}} \times q_{0\text{SP1}};\ q_{m\text{EDA1}} \times q_{m\text{SP1}}) \end{aligned} \tag{19}$$

where $Q_{\text{EDA1}} \times Q_{\text{SP1}} = \{\text{W1.W2.S0}, \text{W1.B2.S0}, \text{B1.W2.S1}, \text{B1.B2.S1}\}$, $E_{\text{EDA1}} \cup E_{\text{SP1}} = \{f_1, f_2, r_1, r_2\}$, and $q_{0\text{EDA1}} \times q_{0\text{SP1}} = \text{W1.W2.S0}$; it is not necessary to denote the marked state in application $q_{m\text{SP1}} \times q_{m\text{SP1}} = \varnothing$.

Based on (17) and because E_{EDA1}=E_{SP1}, this always belongs to the case $e \in E_{\text{EDA}} \cap E_{\text{SP}}$, and the transition function is achievable:

$$\delta_{\text{SPA1}}((x.y), e) = (\delta_{\text{EDA1}}(x, e).\delta_{\text{SP1}}(y, e)) \tag{20}$$

where $e \in E_{\text{EDA1}} \cap E_{\text{SP1}}$.

As EDA1 and SP1 are given and then generated from δ_{EDA1} and δ_{SP1}, the synchronized result transition function δ_{SPA1} is achieved, as shown in Table 1.

Table 1 Management requirement satisfying the transition function of SPA1

SPA1 transition function	Transition event	EDA1 transition function	SP1 transition function
$\delta_{\text{SPA1}}((\text{W1.W2.S2}), f_1) = \text{B1.W2.S1}$	f_1	$\delta_{\text{EDA1}}((\text{W1.W2}), f_1) = \text{B1.W2}$	$\delta_{\text{EDA1}}(\text{S2}, f_1) = \text{S1}$
$\delta_{\text{SPA1}}((\text{W1.W2.S1}), r_1) = \text{W1.W2.S2}$	r_1	$\delta_{\text{EDA1}}((\text{W1.W}_2), r_1) = \text{W1.W2}$	$\delta_{\text{EDA1}}(\text{S1}, r_1) = \text{S2}$
$\delta_{\text{SPA1}}((\text{W1.W2.S2}), f_2) = \text{W1.B2.S0}$	f_2	$\delta_{\text{EDA1}}((\text{W1.W2}), f_2) = \text{W1.B2}$	$\delta_{\text{EDA1}}(\text{S2}, f_2) = \text{S0}$
$\delta_{\text{SPA1}}((\text{W1.B2.S2}), r_2) = \text{W1.W2.S2}$	r_2	$\delta_{\text{EDA1}}((\text{W1.B2}), r_2) = \text{W1.W2}$	$\delta_{\text{EDA1}}(\text{S0}, r_2) = \text{S2}$
$\delta_{\text{SPA1}}((\text{W1.W2.S2}), f_1) = \text{B1.B2.S1}$	f_1	$\delta_{\text{EDA1}}((\text{W1.B2}), f_1) = \text{B1.B2}$	$\delta_{\text{EDA1}}(\text{S2}, f_1) = \text{S1}$
$\delta_{\text{SPA1}}((\text{B1.B2.S1}), r_1) = \text{W1.B2.S2}$	r_1	$\delta_{\text{EDA1}}((\text{B1.B2}), r_1) = \text{W1.B2}$	$\delta_{\text{EDA1}}(\text{S1}, r_1) = \text{S1}$
$\delta_{\text{SPA1}}((\text{B1.W2.S1}), f_2) = \text{W1.B2.S1}$	f_2	$\delta_{\text{EDA1}}((\text{B1.W2}), f_2) = \text{W1.B2}$	$\delta_{\text{EDA1}}(\text{S2}, f_2) = \text{S1}$

This transition function satisfies the management requirement because there is no transition event from state (B1.B2.S1) to state (B1.W2.S1). The transiting detail is shown in Fig.(c). The self-loop events of S1 is only $\{f_1, f_2\}$ without "r_2", so in SP1, it can be explained by $\delta_{\text{SP1}}(\text{S1}, r_2) = \varnothing \neq \text{S1}$. In SPA, the calculation of the value in $\delta_{\text{SPA1}}((\text{B1.B2.S1}), r_2)$is

Since $\delta_{\text{ED1}}((\text{B1.B2}), r_2) = \text{B1.W2}$ and $\delta_{\text{SP1}}(\text{S1}, r_2) = \varnothing$, then

$$\delta_{\text{SPA1}}((x.y), e) = (\delta_{\text{EDA1}}(x, e).\delta_{\text{SP1}}(y, e)) \tag{21}$$

where

$$\begin{cases} e \in E_{\text{EDA1}} \cap E_{\text{SP1}} \\ r_2 \in E_{\text{EDA1}} \cap E_{\text{SP1}} = \{f_1, f_2, r_1, r_2\} \end{cases}, \tag{22}$$

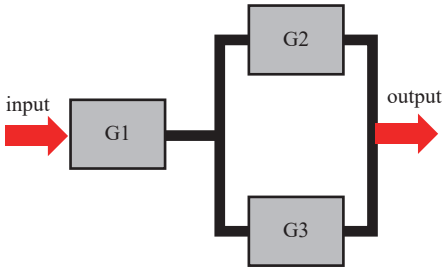
$$\begin{aligned} &\delta_{\text{SPA1}}(((\text{B1.B2}).\text{S1}), r_2) = \\ &(\delta_{\text{EDA1}}((\text{B1.B2}), r_2).\delta_{\text{SP1}}(\text{S1}, r_2)) = \\ &((\text{B1.W2}).\varnothing) \notin E_{\text{SPA1}} = \\ &\varnothing \neq (\text{B1.B2.S1}), \end{aligned} \tag{23}$$

where this state does not exist. As the synchronization result has been dismissed, the G2 repair operation event r_2 from both components is a broken down state (B1.B2.S1), and the management requirement is satisfied.

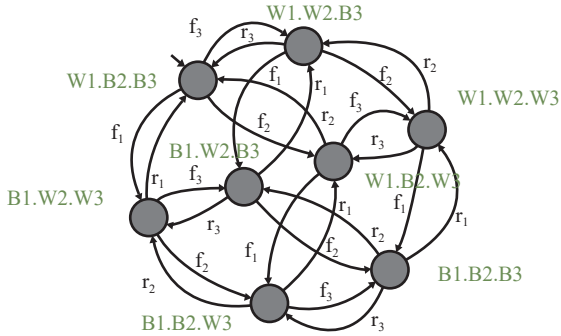
In summary, whenever a management requirement arrives, the CTMC can be translated into EDA. The SP (specification automaton) is designed. By the synchronization operation of EDA and SP, the SPA can be achieved. Translating SPA to the CTMC model, a manageable CTMC that satisfies the management requirement is generated.

3. Management requirements specifying the benchmark

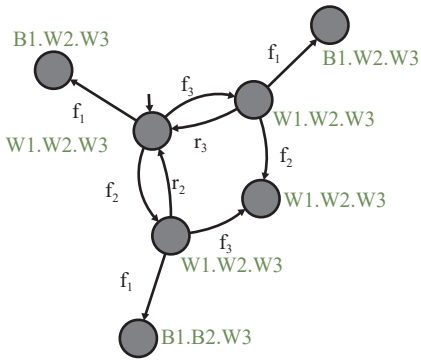
Various management requirements are set in a practical application process, and the variety of system architectures and system behaviors further adds to the complexity of generating the model. This solution has the advantage of agility. Facing various CTMC models, one kind of management requirement has one consistent specifying solution. For example, in Fig. 3, the system architecture is a series of working systems constructed by two components, and in Fig. 4, the system architecture is a series-parallel working system constructed by three components. Obviously, these two different systems have different availability models. However, if the management requirement demand is the same as "One of the components has the priority to be first repaired", the specification automaton must be uniform, as depicted in Fig. 3(c) and Fig. 5(c). Moreover, this uniform specification automaton design is suitable for all CTMC models.



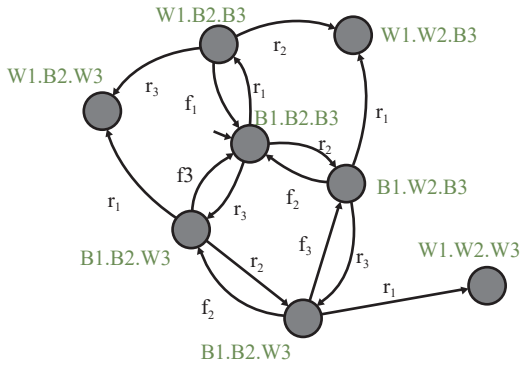
(a) Series-parallel system



(b) Availability EDA

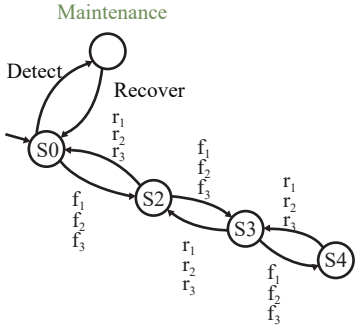


(c) Reliability EDA

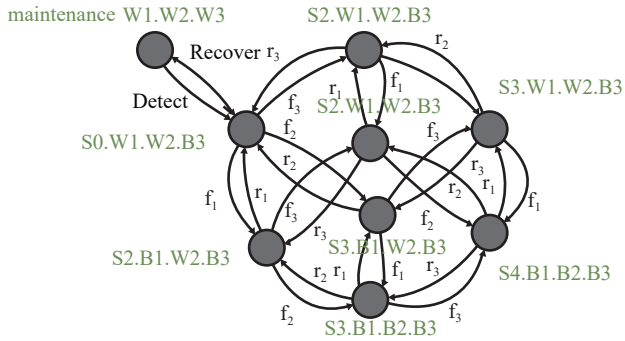


(d) Maintainability EDA

Fig. 4 EDA models

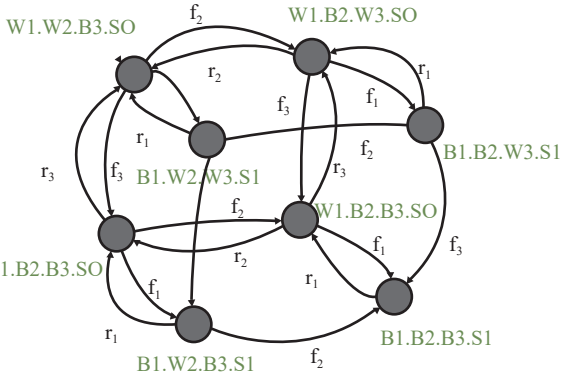


(a) DM1

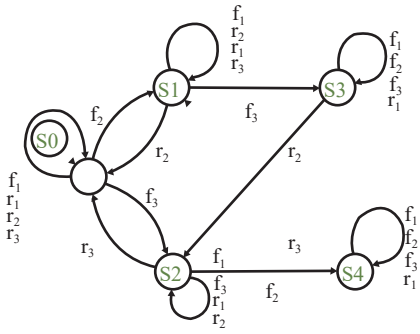


(b) Availability EDA//DM1

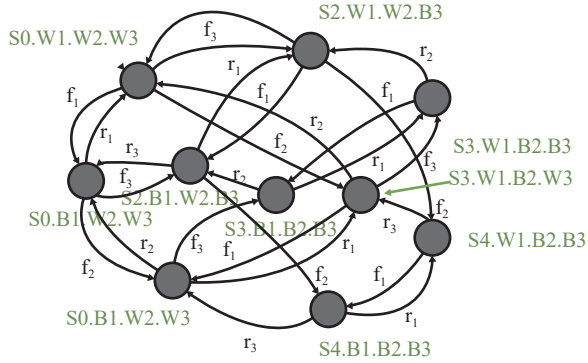
(c) PtR1



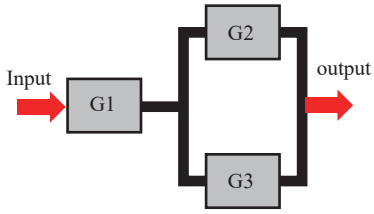
(d) Availability EDA//PtR1



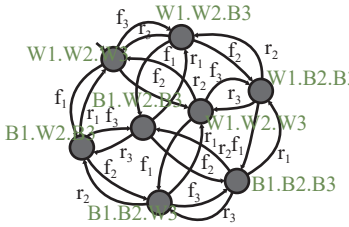
(e) FFFR1



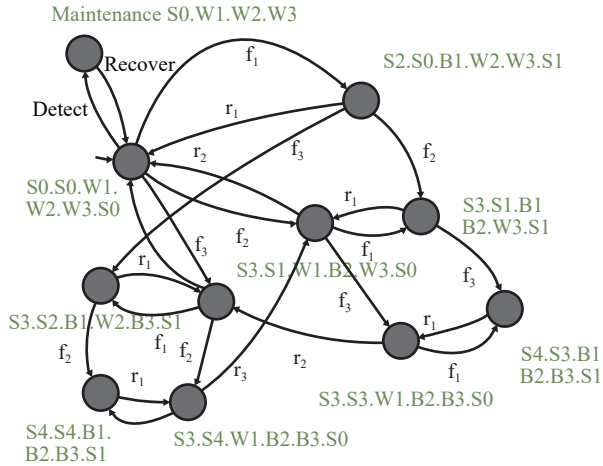
(f) Availability EDA//FFFR1



(g) Series-parallel system



(h) Availability EDA



(i) Availability EDA//DM1//PtR1//FFFR1

Fig. 5 Availability model specified by various management requirements

As the management requirements are uncountable in reality, this part offers a benchmark for three widely used management requirements: DM, PtR, and FFFR. A series-parallel system is used to study this specifying solution. Component G1 has PtR for its relatively important position. Components G2 and G3 are repaired according to FFFR, assuming that G2 and G3 are lying in similar positions. The whole system needs a DM at the initial state, and all the components are healthy ("test" behavior for beginning of maintenance and "recover" behavior for system recovery to work). These management requirements are specified for availability, reliability and maintainability models. Based on the algorithm shown in Fig. 1, as the converted CTMC and EDA are already introduced, this example begins from the system EDA models. After the specified EDA models are achieved, the specified CTMC will be reasonably achievable.

3.1 Repairable series-parallel system

To illustrate the specifying method of different management requirements, a typical series-parallel system is studied. Depicted in Fig. 4(a), this system is constructed by three components, G1, G2, and G3, and they are all repairable if broken down. G1 treats the system input, and its output feeds the input of G2 and G3. G2 and G3 both contribute their output to the system's production.

Moreover, as shown in Table 2, each of these components has unique behaviors, which means that the transitions in the CTMC model and EDA model are unique (λ_i is the failure rate, f_i is the failure event, μ_i is the successfully repaired rate, and r_i is the successfully repaired event).

Table 2 Transition equivalence conversion

CTMC	EDA	
Transition	Transition	By rate
λ_1	f_1	λ_1
λ_2	f_2	λ_2
λ_3	f_3	λ_3
μ_1	r_1	μ_1
μ_2	r_2	μ_2
μ_3	r_3	μ_3

According to the definitions of availability, reliability and maintainability, the corresponding models are established in Fig. 4. Availability denotes the system initially working healthily. With the possible failures of each component, the system is totally broken down, and each component can be repaired. Reliability only focuses on how the system breaks down, and maintainability only focuses on how the system recovers, while during breakdown and recovery, the components can be further broken and repaired in a timely manner. For the state names, "W*i*" is the "working" situation of G*i*, and "B*i*" is the "broken down" situation of G*i*. For example, in the reliability model, the models start from W1.W2.W3, which means that these three components are all working healthily. Whenever G1 breaks down by f_1, the system comes to B1.W2.W3, which means G1 is broken down and G2/G3 are working healthily, and then the model ends because the system is totally broken down.

3.2 Availability model-specifying approach

The downtime maintenance management requirement-specifying solution is illustrated by a new state in which the maintenance operation is added to the original availability model. From the initial state in which all the components are working well, driven by the system test operation, the system reaches the maintenance state. After maintenance, driven by the "recover" operation, the system recovers to work. In Fig. 5(a), the automaton is designed by the semantic "keep the availability model as original and adding a maintenance state into the model": first, the maintenance state changes from the initial state, and then, the transition between states S0, S2, S3, and S4 are $\{f_1, f_2, f_3\}$ and $\{r_1, r_2, r_3\}$, describing the availability model. "From the initial state, a stochastic state is reached, and the failure of G1, G2, and G3 occurs, as well as the repair operations". Based on (3), the synchronized result of the specification automaton and original automaton is in Fig. 5(b). This result satisfies the management requirement by adding a state (maintenance.W1.W2.W2) that represents the system maintenance operation driven by "test" and "recover" events.

Already introduced in the example of the series system in Fig. 3, because G1 is the priority and the first to be repaired, the specification automaton PtR1 in Fig. 5 is designed: in case the failure of G1 happens (in PtR1, the transition is $S0 \xrightarrow{f_1} S1$), no repair operations of G2 or G3 should happen while the only allowed repair r1 if returning to the initial state. The specified model is Availability EDA//PtR1, which satisfies the management requirements; for example, state B1.B2.B3.S1 indicates that all components are broken down, and the output transitions of this state have only one repair r_1: "until component G1 is repaired, G2 and G3 are allowed to be repaired".

Components G2 and G3 should follow the "first failed first repaired" requirement, and the specification automaton is depicted in Fig. 5(e). From the initial state, there are two failure events, G2 and G3. for $S0 \xrightarrow{f_2} S1 \xrightarrow{f_3} S3$, in state S3, G2 fails before G3, so the output repair operation is designed as $S3 \xrightarrow{r_2} S2 \xrightarrow{r_3} S0$, where G2 is repaired before G3; for $S0 \xrightarrow{f_3} S2 \xrightarrow{f_2} S4$, in state S4, G3 fails before G2, so the output repair operation is designed as $S4 \xrightarrow{r_3} S1 \xrightarrow{r_2} S0$, where G3 is repaired before G2. Moreover, the self-loop state always contains the behaviors $\{f_1, r_1\}$ of G1. This self-loop design illustrates the specification automaton only aimed at G2 and G3, without changing event sequences belonging to G1 (no transition function activated for f_1 or r_1 according to the transition function of FFFR1 and availability EDA). Based on (3), the synchronized result shown in Fig. 5(f) satisfies this management requirement.

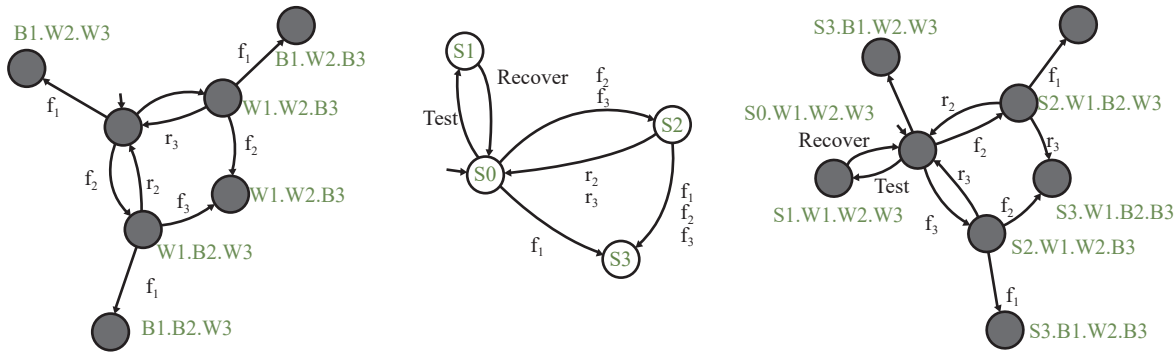
For some cases, after the system has been specified by one requirement and another requirement comes as a supplement, there is no need to design a specification automaton to contain all the information of these two management requirements. With the precondition that "The multiple management requirements aiming at one system are not conflicting", this research offers a stochastic multi-specifying function by continuously varying specification automata, where each specification automaton contains one kind of management requirement, without the necessity of modifying the previous models regarding compatibility. For example, as shown in Fig. 5, as series-parallel system management requirements are as follows: (i) G1 has the priority to be first repaired, as presented by PtR1; (ii) G2 and G3 share the "first broken down, first repaired" condition, as presented by FFFR1; (iii) the system needs downtime maintenance, as presented by DM1. These multiple management requirements are comprehensively specified into the system model by synchronizing the refereed specification models EDA//DM1//PtR1//FFFR1.

In Fig. 5(b), from the initial state S0.W1.W2.W3, there exists an output transition "detect", and it comes to state "Maintenance". With a transition "recover", the system goes back to the initial state. It can be read that this sequence "$S0.W1.W2.W3 \xrightarrow{\text{detect}}$ Maintenance $\xrightarrow{\text{recover}}$ S0. W1.W2.W3" makes the system satisfies the requirement "DM ". In Fig. 5(d), the states "B1.W2.B3.S1 ",

"B1.B2.W3.S1 ", "B1.B2.B3.S1 " respectively means "Component G1 breaks, G2 works, G3 breaks ", "G1 breaks, G2 breaks, G3 works ", and "G1 breaks, G2 breaks, G3 breaks". These states all have the case that G1 and the other components breaks down at the same time. The repairing sequences of these states are "B1.W2.B3.S1 $\xrightarrow{r_1}$, B1.B2.W3.S1 $\xrightarrow{r_1}$, B1.B2.B3.S1 $\xrightarrow{r_1}$", which makes the model satisfies the requirement that G1 has the priority to be first repaired (PtR). In Fig. 5(f), the failure sequence "S0.W1.W2.W3 $\xrightarrow{f_3}$ S2.W1.W2.B3 $\xrightarrow{f_2}$ S4.W1.B2.B3" means that G3 first breaks down and then G2 breaks down, so their repair sequence is "S4.W1.B2.B3 $\xrightarrow{r_3}$ S1.W1.B2.W3 $\xrightarrow{r_2}$ S0.W1.W2.W3", which means the first breaks down first repaired management is satisfied. The all the other of cases "failure and repair " for G2 and G3, for example S1.W1.B2.W3 and then S3.W1.B2.B3. The model satisfies the management FFFR. Furthermore, this verification of the requirement satisfaction is also suitable for the availability EDA//DM1//PtR1//FFFR1 and also suitable for Subsection 3.3 and Subsection 3.4.

3.3 Reliability model-specifying approach

The reliability EDA of the series-parallel system shown in Fig. 6 presents the system from the initially healthy working to the system broken down, and this reliability EDA naturally satisfies the management requirement "G1 should be repaired first" because if G1 is broken down, the model stops without applying a further repair operation to G1 (state B1.W1.W2, state B1.W2.B3 and state B1.B2.W3 are absorbing states); additionally, this reliability EDA also satisfies "G2 and G3 are first broken down and first repaired " because when G2 and G3 are both broken down, the reliability model stops without a further repair operation (state W1.B2.B3 is an absorbing state). Therefore, the only management requirement that should be specified is downtime maintenance.



(a) Relilability EDA (b) DM2 (c) Reliability EDA//DM2

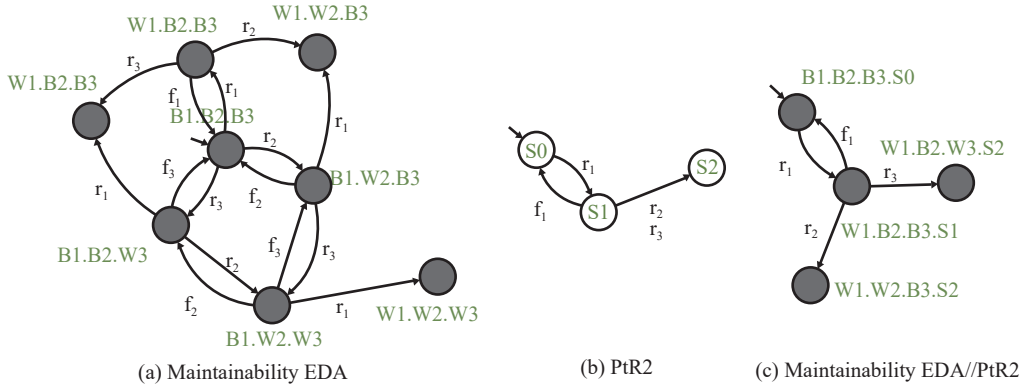
Fig. 6 Reliability model specified by downtime maintenance management requirement

The downtime maintenance management requirement specifying automaton DM2 is shown in Fig. 6, since the same establishment principle is already introduced in DM1, which is: adding a maintenance state to present downtime operation from the initial state. Moreover, the transitions among S0, S2, and S3 are applied to describe the behaviors of the reliability EDA, ensuring that the synchronized result will remain the same transition situation as the reliability EDA. Based on (3), the synchronized result is reliability EDA//DM2, which satisfies this management requirement.

3.4 Maintainability model specifying approach

Unlike the reliability and availability models, the initial state of the maintainability model is B1.B2.B3, because the maintainability model assesses the system characteristics recovering from the total broken down state to working state. Aiming at this series-parallel system, the management requirements "G2 and G3 first fail and are first repaired " or "system downtime maintenance " are not suitable; the only specifying task is for "G1 has the priority to be first repaired".

In Fig. 7, transitions between states S0 and S1 allow component G1, "repair " and "failure ", and transitions from states S1 to S2 further repair G2 and G3. This specification automaton design can ensure that the model can freely repair G1, and after G2 or G3 is further repaired, the series-parallel system recovers to a healthy state. Then, the maintainability model stops, without the possibility of failure occurring in G2 or G3 (depicted in PtR2, where state S2 is an absorbing state); thus, the management requirement is satisfied.



(a) Maintainability EDA (b) PtR2 (c) Maintainability EDA//PtR2

Fig. 7 Maintainability model specified by priority to be first repaired management requirement

According to (3), the specified result is shown in maintainability EDA//PtR2, where the management requirement is satisfied.

This approach can be developed to be a safety-assessment tool package as a software plug-in of the engineering tools. This application could serve the full life cycle of the industrial systems.

In the system designing stage, the hardware structure is integrated to be the EDA, and the planed managements are integrated to be the specification automata (SP). By synchronizing EDA and SP (SPA=EDA//SP), the specified models (SPA) which fuse the managements and the system structure are made. After several times of iterations and upgrades, a safety critically considered system is able to be produced.

In the operation and maintenance stage, various situations can timely happen to the industrial system, for example, A: objective change, B: new management plans, C: maintenance requirements, etc. Whenever these new managements arrive, the system operators work on the tool and assign the corresponding SPs (SP-A, SP-B and SP-C) and the dynamic specified model SPA=EDA//SP-A//SP-B//SP-C is obtained. Dynamic reliability, availability and maintainability models are handled, as well as model-based assessment can be realized in actual system.

4. Conclusions

This research provides a stochastic management requirement specifying approach for system assessment CTMC models by converting the CTMC model into EDA models and designing specification automata based on various management requirements. After the synchronize, operation is applied to the specification automata and the system EDA models, a management requirement specified system EDA model is obtained. With the further conversion from EDA to the CTMC, the management requirement specified CTMC is achievable, and the system can be assessed considering the management requirements. In this paper, a benchmark of a series-parallel system constructed by three faulty and repairable components is applied, while three popular applied DM, PtR, and FFFR management are specified in the availability, reliability, and maintainability model of this series-parallel system.

Due to the agility of this approach, with the precondition of "the multiple management requirements aiming at one system are not contradictory", one principle specification automaton is established according to one kind of management requirement. Moreover, in case multiple management requirements are set, the specifying solution is illustrated by multiplying and synchronizing the corresponding specification automata.

By the normal approach of merging the safety assessment model based on the system architecture and system behaviors, if one new management requirement is set in the system, the safety model should be resubmitted to satisfy this management requirement, which requires redundant work by a system analyst. Additionally, even given the same management requirement, facing a different system, a new safety model should be also resubmitted, regardless of the scalability and complexity of the system. These two problems are solved in this research by management requirement coherent specification automata synchronized with the previous safety assessment model and by following the same specification automaton establishment principle.

Countless management requirements exist according to different stochastic situations and different systems, so this research provides an open issue for specifying approaches.

References

[1] HEGDE J, ROKSETH B. Applications of machine learning methods for engineering risk assessment-a review. Safety science, 2020, 122: 104492.

[2] HENDRIX B, LEWIS T, EMERY M, et al. Model based functional safety-how functional is it? Journal of system safety, 2022, 57: 32−38.

[3] CROWDER J A, HOFF C W. Requirements engineering:

laying a firm foundation. Cham: Springer, 2022: 197−216.
[4] DIATTE K, O'HALLORAN B, VAN BOSSUYT D L. The integration of reliability, availability, and maintainability into model-based systems engineering. Systems, 2022, 10(4): 101.
[5] KOOHSARI A, KALATEHJARI R, MOOSAZADEH S, et al. A critical investigation on the reliability, availability, and maintainability of EPB machines: a case study. Applied Sciences, 2022, 12(21): 11245.
[6] TORENS C, JUENGER F, SCHIRMER S, et al. Machine learning verification and safety for unmanned aircraft-a literature study. Proc. of the AIAA SCITECH 2022 Forum, 2022: 1133.
[7] VARANASI S C, ARIAS J, SALAZAR, E, et al. Modeling and verification of real-time systems with the event calculus and s (CASP). Proc. of the 24th International Symposium, 2022, 13165: 181–190.
[8] LIU Y X, GUSAK A, JING S Y, et al. Fast prediction of electromigration lifetime with modified mean-time-to-failure equation. Materials Letters, 2022, 325: 132880.
[9] LI Y X, MU L X, GAO P Y. Particle swarm optimization fractional slope entropy: a new time series complexity indicator for bearing fault diagnosis. Fractal and Fractional, 2022, 6(7): 345.
[10] HEO T, LIU D P, MANUEL L. Weather window analysis in operations and maintenance policies for offshore floating multi-purpose platforms. Journal of Offshore Mechanics and Arctic Engineering, 2023, 145(4): 041701.
[11] KHATIB M E, ALHOSANI A, ALHOSANI I, et al. Simulation in project and program management: utilization, challenges and opportunities. American Journal of Industrial and Business Management, 2022, 12(4): 731–749.
[12] VIDAL G H, HERNANDEZ J R C, MINNAARD C, et al. Statistical analysis of manufacturing system complexity. The International Journal of Advanced Manufacturing Technology, 2022, 120(5/6): 3427–3436.
[13] JIANG W, ZHOU K Q, SARKHEYLI-HAGELE A, et al. Modeling, reasoning, and application of fuzzy Petri net model: a survey. Artificial Intelligence Review, 2022, 55(8): 6567–6605.
[14] KUMAR K, SUMIT, KUMAR S, et al. Predicting reliability of software in industrial systems using a Petri net-based approach: a case study on a safety system used in nuclear power plant. Information and Software Technology, 2022, 146: 106895.
[15] BASILE D, BEEK M H. T. Contract automata library. Science of Computer Programming, 2022, 221: 102841.
[16] HENZINGER T A, LEHTINEN K, TOTZKE P. History-deterministic timed automata. Proc. of the 33rd International Conference on Concurrency Theory, 2022. DOI: 10.4230/LIPICS.CONCUR.2022.14.
[17] TAI R C, LIN L Y, ZHU Y T, et al. A new modeling framework for networked discrete-event systems. Automatica, 2022, 138: 110139.
[18] CHEN X L, PENG H, WANG J, et al. Supervisory control of discrete event systems under asynchronous spiking neuron P systems. Information Sciences, 2022, 597: 253–273.
[19] CLAVE N, CACHEUX P, DUTERTRE A, et al. GRIF-Bool: risk assessment using a multi-approach Boolean analysis tool. Proc. of the Lambda Mu 20 congress on risk management and operational safety, 2016. DOI:10.4267/2042/61803.
[20] ZHOU T M, LI D L, QIN F, et al. Dynamic Reliability Analysis of Level Control System of Steam Generator based on BDMP. Preprints 2022, 2022080254. https://doi.org/10.20944/preprints202208.0254.v1.
[21] SERRU T, NGUYEN N, BATTEUX M, et al. Generation of cyberattacks leading to safety top event using AltaRica: an automotive case study. Proc. of the 23rd Lambda Mu Congress, 2022. https://hal.science/hal-03814648.
[22] XIE J, TAN W, YANG Z B, et al. SysML-based compositional verification and safety analysis for safety-critical cyber-physical systems. Connection Science, 2022, 34(1): 911–941.
[23] RIVIERE P, SINGH N K, AIT-AMEUR Y. EB4EB: a framework for reflexive Event-B. Proc. of the 26th International Conference on Engineering of Complex Computer Systems, 2022: 71−80.
[24] GUSTAVSSON A, ERMEDAHL A, LISPER B, et al. Towards WCET analysis of multicore architectures using UPPAAL. Proc. of 10th International Workshop on Worst-Case Execution Time Analysis, 2010. https://doi.org/10.4230/OASIcs.WCET.2010.101.
[25] HAGEMANN P, HERTRICH J, STEIDL G. Stochastic normalizing flows for inverse problems: a Markov chains viewpoint. SIAM/ASA Journal on Uncertainty Quantification, 2022, 10(3): 1162–1190.
[26] XU C, HANSEN M C, WIUF C. Structural classification of continuous time Markov chains with applications. Stochastics, 2022, 94(7): 1003–1030.
[27] LUNDTEIGEN M A, RAUSAND M, INGRID B U, et al. Integrating RAMS engineering and management with the safety life cycle of IEC 61508. Reliability Engineering and System Safety, 2009, 94(12): 1894−1903.
[28] XU C Y. Operational dependability model generation. Lyon: University of Lyon, 2020.
[29] DUER S, WOZNIAK M, PAS J, et al. Reliability testing of wind farm devices based on the mean time between failures (MTBF). Energies, 2023, 16(4): 1659.
[30] PENA G, MORENO V, BARRAZA N. Stochastic modeling of the mean time between software failures: a review. JOHRI P, ANAND A, VAIN J, et al, ed. System Assurances, 2022: 355−370.
[31] GAO S. Availability and reliability analysis of a retrial system with warm standbys and second optional repair service. Communications in Statistics-Theory and Methods, 2023, 52(4): 1039–1057.
[32] BOGGERO L, FIORITI M, DONELLI G, et al. Model-based mission assurance/ model-based reliability, availability, maintainability, and safety (RAMS). MADNI A M, AUGUSTINE N, SIEVERS M, ed. Handbook of model-based systems engineering. Cham: Springer, 2022.
[33] MISRA K B. Handbook of advanced performability engineering. Cham: Sprnger, 2021.
[34] IONESCU D. Quantitative evaluation of safety event sequences using probabilistic language theory. Lorraine: University of Lorraine, 2016.
[35] XU C Y, NIEL E, BRINZEI N. Discrete event system formal approaches contribution onto global reliability Markov chain generation. Proc. of the 21st Congress on Risk Control and Operational Safety, 2018. https://hal.science/hal-02064-997/.
[36] GARG V, KUMAR K, MARCUS S. A probabilistic language formalism for stochastic discrete-event systems. IEEE Trans. on Automatic Control, 1999, 44(2): 280–293.

[37] LIN F, YING H. Modeling and control of fuzzy discrete event systems. IEEE Trans. on Systems, Man, and Cybernetics, Part B (Cybernetics), 2002, 32(4): 408–415.
[38] RODGER S H, BILSKA A O, KENNETH H, et al. A collection of tools for making automata theory and formal languages come alive. Proc. of the 28th SIGCSE technical symposium on Computer science education, 1997. https://doi.org/10.1145/268084.268089.
[39] ANAND V, CARROLL A E, BIONDICH P G, et al. Pediatric decision support using adapted Arden Syntax. Artificial intelligence in medicine, 2018, 92: 15–23.
[40] IMRICH W, KLAVZAR S, RALL D F. Topics in graph theory: graphs and their gartesian product. Massachusetts:AK Peters Ltd, 2008.
[41] OH C Y, TOMCZAK J M, GAVVES E, et al. Combinatorial bayesian optimization using the graph cartesian product. Proc. of the 33rd Conference on Neural Information Processing Systems, 2019. https://doi.org/10.48550/arXiv.1902.00448.

Biographies

XU Changyi was born in 1989. He received his bachelor degree in electronic information science and technology from Jilin University, Changchun, China, in 2012. He received his master degree from Chinese Academy of Sciences, Changchun, China in 2015, and Ph.D degree in automatic in University of Lyon, Lyon, France in 2021. He is currently an associate professor in Dalian University of Technology. His research interests are systems engineering, electronic technology, electronic technology, control theory and practice, reliability.
E-mail: changyixu@dlut.edu.cn

DUAN Yiman was born in 1996. She received her B.S. and M.S. degrees from Yanshan University in 2019 and 2022, respectively. She is currently working toward Ph.D. degree in the College of Mechanical Engineering, Zhejiang University, Hangzhou, China. Her research interests include systems engineering, control theory and practice, high-performance mechatronic equipment.
E-mail: ymduan@zju.edu.cn

ZHANG Chao was born in 1990. He received his B.S. degree in 2012 and M.S. degree in 2015, from Northwestern Polytechnical University, Xi'an, China. He received his Ph.D. degree in 2019 from University of Lyon, France. He is currently a professor at the Institute of Mechatronics and Control Engineering, Zhejiang University. His research interests include systems engineering, control theory and practice, high-performance mechatronic equipment.
E-mail: chao.zhang@zju.edu.cn