PAPER Special Section on Dependable Computing

Identification of Smallest Unacceptable Combinations of Simultaneous Component Failures in Information Systems*

Kumiko TADANO[†a)], ***Nonmember***, **Jianwen XIANG**[†], ***Member***, **Fumio MACHIDA**[†], ***and*** **Yoshiharu MAENO**[†], ***Nonmembers***

SUMMARY Large-scale disasters may cause simultaneous failures of many components in information systems. In the design for disaster recovery, operational procedures to recover from simultaneous component failures need to be determined so as to satisfy the time-to-recovery objective within the limited budget. For this purpose, it is beneficial to identify the smallest unacceptable combination of component failures (SUCCF) which exceeds the acceptable cost for recovering the system. This allows us to know the limitation of the recovery capability of the designed recovery operation procedure. In this paper, we propose a technique to identify the SUCCF by predicting the required cost for recovery from each combination of component failures with and without two-person cross-check of execution of recovery operations. We synthesize analytic models from the description of recovery operation procedure in the form of SysML Activity Diagram, and solve the models to predict the time-to-recovery and the cost. An example recovery operation procedure for a commercial database management system is used to demonstrate the proposed technique.

key words: *automatic model synthesis, recovery operation procedure, stochastic reward net (SRN), time to recover (TTR)*

1. Introduction

Large-scale disasters such as major earthquakes may cause simultaneous failures of many components in information systems. When the failures occur and customer requirements of time-to-recover (TTR) completely from the failures are not satisfied, penalty costs are charged. Meanwhile, measures to reduce the TTR such as addition of redundant components or assignment of a skilled system operator increase labor or development cost.

Since the budget for disaster recovery (DR) is generally limited in the organization of the owner of the system, it is necessary for system designers to take into account not only TTR but also recovery cost for DR. The system designers need to design recovery operation procedures which satisfy the requirements in consideration with many combinations of simultaneous failures of system components.

For such designs of operation procedures for disaster recovery, it is beneficial to identify the smallest unacceptable combinations of component failures (SUCCF) which exceeds the acceptable recovery cost. The identified SUCCF can be useful for system designers to understand the limitation of the current design and to plan further improvements of recovery operation procedure or to request compromise on the current requirements.

The identification of SUCCF is challenging due to the following reasons:

- Information systems are often composed of many components and have a large numbers of possible combinations of component failures.
- Depending on the combination of the component failures, the required recovery operations and required cost are changed.
- Recovery operations may have restrictions in the order of execution (e.g., to recover a component A, we may need to recover another component B in advance).
- Operations' execution may affect current system states (e.g., successful execution of an operation may recover some failed components), which causes dynamic changes of the required recovery operations.

From the above reasons, it is not realistic to test all the combinations of failures in real system environments.

To deal with the problem, it is reasonable to utilize model-based evaluation approach to assess the design of the recovery operation procedures using only several fundamental parameters' values measured in a target system. However, since the model-based assessments typically require special expertise in mathematical modeling, it would be difficult for practical system designers to manually build a correct analytic model to assess their design of recovery operation procedures.

An automated analytic model synthesis method is necessary to support the model-based assessment. Several researchers proposed techniques to automatically derive stochastic models for availability/reliability/performance assessment such as Dynamic Fault Tree (DFT) [1], Repairable Fault Tree [2], Timed Petri Net (TPN) [3], Stochastic Petri net (SPN) [4], Generalized Stochastic Petri Nets (GSPN) [5]–[7], Deterministic and Stochastic Petri Nets (DSPN) [8], Stochastic Well-formed Net (SWN) [9] and Stochastic Reward Nets (SRN) [10]–[12], from widely-used design description languages such as Unified Modeling Language (UML) [13], System Modeling Language (SysML) [14] and Architecture Analysis & Design Language (AADL) [15]. However, the models derived by these methods do not provide the means to identify the SUCCF.

Manuscript received January 18, 2013.
Manuscript revised April 28, 2013.

[†]The authors are with NEC, Kawasaki-shi, 216–8666 Japan.

*This is the extended version of the preliminary paper [16] which has appeared in the IEEE published Proceedings of the 18th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2012).

a) E-mail: k-tadano@bq.jp.nec.com

DOI: 10.1587/transinf.E96.D.1941

In this paper, we propose a technique to identify the SUCCF by predicting the required cost for recovery from each possible combination of component failures. We synthesize analytic models from the description of recovery operation procedure in the form of SysML Activity Diagram, and solve the models to predict the time to recovery and the associated cost. The feasibility of the proposed technique is evaluated in an example of recovery operation procedures for a commercial database management system. This paper is an extension of our previous work [16], which further incorporates cross-check of operations' execution for reduction of human errors into the models, and evaluates the effect of the cross-check on the recovery cost in the case study. In addition, we incorporate a new translation rule into the model synthesis to represent a flow in the case of operation failure which has been often neglected in the system reliability evaluation.

This paper is organized as follows. Section 2 describes the concept of SUCCF. Section 3 proposes a technique to identify the SUCCF. Section 4 shows a case study which applies the proposed technique to a recovery operation procedure for a commercial database management system. Section 5 gives our conclusion.

2. SUCCF

This section describes the concept of SUCCF. First, we define UCCF as the unacceptable combinations (sets) of components failures which exceed the acceptable cost for recovering the system. SUCCF is the subset of UCCF that has the minimum number of elements. In other words, any UCCF can be reduced to a SUCCF by removing its elements.

For example, let us assume that there are four components A, B, C, and D in a system, whose failures are denoted as F_A, F_B, F_C, and F_D. The requirement of recovery cost of the system is C_{req}. The values of cost exceed C_{req} when the combinations of component failures $\{F_A, F_B\}$, $\{F_A, F_B, F_C\}$, $\{F_A, F_B, F_D\}$, or $\{F_A, F_B, F_C, F_D\}$ occurred simultaneously. In this case, $\{F_A, F_B\}$ is the only one SUCCF, because its number of component failures is the smallest among the four combinations.

SUCCF can be useful for system designers to understand the limitation of the current design and to plan further improvements of recovery operation procedure or to request compromise on the current requirements.

The SUCCF concept is similar to the concept of minimal cut set (MCS) which has been traditionally used in Fault Tree Analysis (FTA) [17]. A MCS is a smallest combination of basic events (e.g. component failures) which may lead to a specific undesired top event at the system level (e.g. whole system failure). The MCSs of a fault tree can be used to understand the structural vulnerability of the system. However, FTA is not applicable for recovery operations with complex dependencies as mentioned in Sect. 1, and this is the main reason that we introduce the SUCCF concept.

3. SUCCF Identification Method

In this section, we present a method to predict the TTR and the required cost for recovery from each combination of failures by automatically synthesizing an analytic model from a recovery operation procedure. A mechanism to identify SUCCF is subsequently introduced.

3.1 Recovery Operation Procedure

Recovery operation procedure is a procedure to recover failed components and consists of sub-procedures for recovering parts of system components. Each sub-procedure contains system administrative operations such as replace, reboot, data restoration and configuration changes etc. and is specified in documents or manuals in advance. When a disaster occurs and causes simultaneous component failures, system operators are responsible for component recovery following to the recovery operation procedure. Since the required sub-procedures vary with the combinations of the failed components, the system operators first need to figure out the damage of the systems accurately (i.e. identify the failed components), and then decide the sub-procedures to be executed for system recovery. We assume the system is composed of components, each of which has two states; available or unavailable. Although a component is not failed, it is considered unavailable if its functionality is not provided due to some reasons. According to the different combinations of failed states of the components, the required sub-procedures in the recovery operation procedure can be varied.

3.2 Overview of the Identification Method of SUCCF

The proposed method consists of two functions: *analytic model synthesis* for predicting the TTR and cost, and *identification of SUCCF*, as shown in Fig. 1. First, system designer describes a recovery operation procedure using SysML Activity Diagram. Next, analytic model synthesis function automatically translates the Activity Diagram to an analytic model in the form of Stochastic Reward Nets (SRN) [18] by organizing predefined SRN model modules stored in *translation rules repository*. Finally, the identification function localizes SUCCF by analyzing the TTR and cost based on the synthesized SRN model. The identified SUCCF and the associated values of the TTR and cost are returned to the system designer. The system designer then can refine the design of the recovery operation procedure according to the SUCCF.

3.3 Activity Diagram

Activity Diagram is used to specify a recovery operation procedure by a system designer. Figure 2 shows an example of an Activity Diagram representing a recovery operation procedure. The following five types of nodes are used to

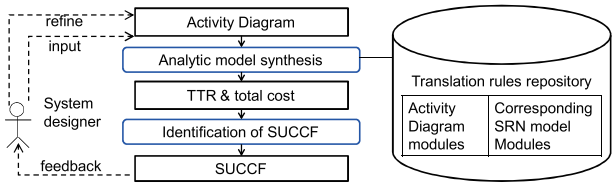


Fig. 1 Overview of the proposed method.

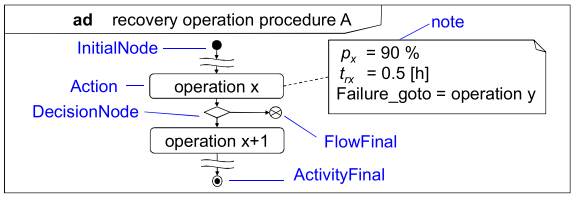


Fig. 2 An example of Activity Diagram.

represent features of the recovery operation procedure.

- An Action, denoted by a rectangle, represents a unit operation (e.g., replace, reboot, data restoration etc.) of the recovery operation procedure.
- A DecisionNode, denoted by a rhombus, represents a conditional branch whose output (yes/no) is determined according to a state of a certain system component(s) (e.g. the component is up or down, backup files exist or not). The system designer determines which output corresponds to the failure state of the component, because it is used for the analytic model synthesis. The state is changed by success of execution of a certain operation(s) (such operations do not exist for unrecoverable failures).
- An InitialNode, denoted by a filled circle, represents the starting point of control flow of the recovery operation procedure.
- An ActivityFinal, denoted by a circle with an internal filled circle, represents an ending point where the failed system is entirely recovered (e.g. the latest data of a failed database server are recovered).
- A FlowFinal, denoted by a circle with a cross, represents another ending point where the system remains a broken part (e.g. a failed database server is recovered with data corruption by loss of redo logs).

We defined the following parameters: time to complete checking the state t_{cy} for a DecisionNode d_y, success probability of a recovery operation p_x and time to complete the recovery operation t_{rx} for an Action a_x. The parameters' values are specified in a note associated with the nodes by the system designer. These parameters' values are utilized as transition rates/probabilities in the synthesized SRN. In real recovery operation procedures, the system designer often does not explicitly specify what the operator should do next in the case of operation failure. However, operation failure may occur. For example, unintended UNIX/Linux command-line mistakes are not uncommon for many users, and sometimes have undesirable effect on their systems. In a human reliability analysis (HRA) method called HEART [19], if no obvious means of reversing an unintended action are provided, the expected human error rate is multiplied by eight. In order to have the system designer specify next activity to be done in the case of operation failure, we also introduced *failure_goto*, which specifies an Activity Diagram module (mentioned in Sect. 3.5) of the next activity.

Target system components for each operation are specified using SysML allocations, which represent various relationships between the SysML elements (such as blocks, action nodes, etc.). We define two stereotypes of allocations among Actions and DecisionNodes to represent relationship in the recovery operation procedure as follows.

- *control with condition*: An allocation with this stereotype from an Action representing an operation op_1 to a DecisionNode d_1 means that success of execution of op_1 changes the output of d_1. The change(s) by the success of execution is described as a condition in a note associated with the allocation. On the other hand, allocation with this stereotype from a DecisionNode d_2 to a DecisionNode d_3 means that the output of d_3 is changed according to the output of d_2. The change(s) by d_2 is described as a condition in a note associated with the allocation as well.
- *prior*: Some operations have dependencies in order of the operations' execution. A typical example is that to recover from a failure, success of execution of an operation op_1 is required before success of execution of an operation op_2. In many real recovery operation procedures, order of execution of operations is described under the assumption that each operation will be successfully done, but cases of operation failures are not adequately described. To have the system designer describe clearly the dependencies constraints, we introduce this stereotype. When there are two allocations with stereotype *control with condition* from two Actions a_1 and a_2 for op_1 and op_2 to a DecisionNode d_1, allocation with stereotype *prior* from a_1 to a_2 means that success of execution of op_1 is required to change the output of d_1 before that of op_2.

By assigning the stereotypes in SysML diagrams before the synthesis of the SRN model, the system designer does not have to be aware of the underlying SRN models in the analysis of SUCCF.

3.4 Cross-Checking

In execution of recovery operation procedures for disaster recovery, human errors are prone to occur because operators could be highly stressed under time pressure in extreme situations and need to perform rarely-used complex recovery operation procedures. Human error rates have been predicted mainly based on five approaches [20]: investigation of already-occurred incidents, experiment in real environment, observation of practical operations, simulation

and probabilistic risk assessment (PRA). In nuclear power plants and chemical plants, PRA is mostly used to analyze human errors. In the framework of PRA, human reliability analysis (HRA) was developed. There are many techniques for HRA such as THERP [21], HEART [19], CREAM [22] and SPAR-H [23]. A detailed comparative evaluation of various HRA techniques can be found in [24].

Cross-check of operations by two people is a widely-used technique for human error reduction which is applicable to various kinds of mission critical systems such as nuclear power plants, military and medical services. A control method of execution of system administrative operations in an information system by two system administrators is proposed in [25]. The method in [25] prevents the system administrators from executing undesirable operations from security perspective. However, to our best knowledge, the effect of cross-check of recovery operations in recovery operation procedures for information systems on the recovery cost has not been quantitatively evaluated in the existing literature.

To address this issue, the proposed method identifies the SUCCF by incorporating the effect of cross-check for recovery operation procedures in information systems. We assume that system operator(s) tries to perform recovery operations according to the pre-defined recovery operation procedure. Intentional malicious operations are not included in the scope of the paper. We also assume that the checks by two system operators are independently done. Under the assumption, the success probability of operation with cross-check p_{x_cc} is calculated by the following formula:

$$p_{x\_cc} = 1 - (1 - p_x) \cdot (1 - p_{xc}) \quad (1)$$

Where p_x and p_{xc} are the success probabilities of operation x by the two system operators, respectively. The values of p_x and p_{xc} could be the same if the system operators have the same level of expertise. The proposed method uses p_x as the success probability of operation x in the case without cross-check, and p_{x_cc} in the case with cross-check instead of p_x for model analysis. Both of the analysis results are shown to the system designer for comparison.

By incorporating the effect of cross-check, the proposed method can evaluate the recovery cost including labor cost for adding operators for cross-check and downtime cost by decreasing TTR due to reduction of human errors.

3.5 Model Synthesis from Recovery Operation Procedure

The proposed method synthesizes an analytic model in the form of SRN from the input recovery operation procedure in the Activity Diagram. The synthesized SRN model consists of three types of sub-models: system state model, control flow model, and recovery operation model. A control flow model represents the entire control flow of the input recovery operation procedure. A recovery operation model represents a recovery operation for a specific component. A system state model represents state transitions (properly functioning or failed) of a specific part of the system.

The SRN model is synthesized according to the following three steps based on the translation rules shown in Table 1.

Step 1. Split the Activity Diagram into Activity Diagram modules (AD modules)
Step 2. Translate each AD module into an SRN model module
Step 3. Compose the translated SRN model modules into an integrated SRN model

In Step 1, the proposed method splits the input Activity Diagram into the AD modules composed of nodes and outgoing edges, as shown in the third column in Table 1. Edges drawn by gray dashed lines of the AD modules represent edges of their previous AD modules. Edges drawn by black dashed lines of AD modules represent allocations. The AD modules (a), (b), (c), (d), and (e) contain nodes and outgoing edges to the next AD modules. The AD modules (f), (g), and (h) contain the allocations only. The numbers of incoming edges to the Activity Diagram module are fixed to 0 for AD model module (a). The numbers of incoming edges to (b), (c), (d) and (e) depend on the control flow of the input Activity Diagram. In Step 2, each of nodes is translated to the corresponding SRN model module(s) shown in the 4–6th columns in Table 1. In Step 3, according to the relation connections of the original AD modules, the proposed method composes the translated SRN model modules into one SRN model for analysis. The output arc(s) of each SRN model module is connected to the place(s) of SRN model module(s) translated from the next AD module(s) in the original Activity Diagram. If the next AD module is (b)/(c)/(d)/(e), the output arc is connected to the place $\mathrm{P}_{\mathrm{pre}\theta}/\mathrm{P}_{\mathrm{exec}x}/\mathrm{P}_{\mathrm{unrecv}\phi}/\mathrm{P}_{\mathrm{recv}}$.

In Table 1, we utilize the following naming convention for guard functions of the synthesized model. If a transition has a guard function, the name of the transition has the name of the guard function as its subscript. The name of each guard function starts with "g", and its subscript is the name of the place where a token enables the transition.

The detailed translation from the AD modules to the SRN model modules is as follows.

(a) The InitialNode module contains an InitialNode and an outgoing edge. The InitialNode module is translated into the starting place of the control flow model and an outgoing edge from the place.
(b) The DecisionNode module contains a DecisionNode and two outgoing edges. The DecisionNode module is translated into (1) places, transitions and arcs of the control flow model, and (2) two places of the system state model. Regarding (1), when a token moves to $\mathrm{P}_{\mathrm{pre}\theta}$ from the previous SRN model modules translated from AD modules (a), (b), or (c), the token moves to $\mathrm{P}_{\mathrm{dec}\theta}$ with the transition rate $1/t_1$ [1/h]. If output of a DecisionNode d_θ is yes/no, then $\mathrm{t}_{\mathrm{gyes}\theta}/\mathrm{t}_{\mathrm{gno}\theta}$ fires and the token moves to the place of the SRN model module translated from the next AD module. As for (2), $\mathrm{P}_{\mathrm{yes}\theta}$

Table 1 Translation rules for synthesizing a SRN model from a recovery operation procedure in an activity diagram.

Input Activity Diagram modules			Output SRN model modules						
	Module name	Concrete syntax example	Control flow model	Recovery operation model	System state model	Transition			
						Name	Rate[1/h]/ Probability	Priority	Guard function
Activities	(a) InitialNode module		P_{init}, t_i	n/a	n/a	t_i	1	2	n/a
	(b) DecisionNode module	d_θ, [yes], [no], Is θ properly functioning? $t_{c\theta}$ = t_1 [hrs]	$P_{pre\theta}$, $P_{dec\theta}$, $T_{dec\theta}$, $t_{gyes\theta}$, $t_{gno\theta}$	n/a	$P_{no\theta}$, $P_{yes\theta}$	$T_{dec\theta}$	$1/t_1$	0	n/a
						$t_{gyes\theta}$	1	1	$g_{yes\theta}$
						$t_{gno\theta}$	1	1	$g_{no\theta}$
	(c) Action module	op_x, r_x = p_1 %, t_{rx} = t_2 [hrs], Failure_goto$_x$: None / d_θ	t_{gfailx}, P_{execx}, $t_{gfinishx}$, $P_{returnx}$, P_{gotox}, $t_{gbeforex}$	$P_{finishx}$, $t_{greturnx}$ / $t_{greturnxgotox}$, t_{xf}, t_x, P_{failx}, P_{opx}, t_{failx}, $t_{successx}$, $P_{branchx}$, T_{gexecx}, $P_{beforex}$	n/a	$t_{gfinishx}$	1	1	$g_{finishx}$
						$t_{gbeforex}$	1	1	$g_{beforex}$
						T_{gexecx}	$1/t_2$	0	g_{execx}
						$t_{successx}$	p_1	1	n/a
						t_{failx}	1- p_1	1	n/a
						$t_{greturnx}$ / $t_{greturnxgotox}$	1	1	$g_{returnx}$ / $g_{returnxgotox}$
	(d) FlowFinal module		$P_{unrecv\phi}$	n/a	n/a	n/a	n/a	n/a	n/a
	(e) ActivityFinal module		P_{recv}	n/a	n/a	n/a	n/a	n/a	n/a
Allocations	(f) Control with condition module I	d_θ, <<control>>, op_x, d_θ : [no/yes] -> [yes/no]	n/a	n/a	$P_{yes/no\theta}$, t_{gopx}, $P_{no/yes\theta}$	t_{gopx}	1	2	g_{opx}
	(g) Control with condition module II	d_θ, <<control>>, d_ψ, If d_ψ = [yes/no] then d_θ : [no/yes] -> [yes/no]	n/a	n/a	$P_{yes/no\theta}$, $t_{gyes/no\psi}$, $P_{no/yes\theta}$	$t_{gyes/no\psi}$	1	2	$g_{yes/no\psi}$
	(h) Prior module	d_θ, <<control>>, op_x, <<prior>>, <<control>>, op_y	n/a	n/a	$P_{yes/no\theta}$, t_{gopy}, $P_{int\theta}$, t_{gopx}, $P_{no/yes\theta}$	t_{gopx}	1	2	g_{opx}
						t_{gopy}	1	2	g_{opy}

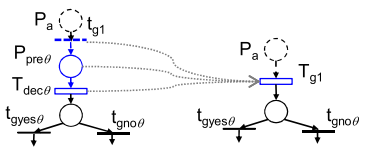


Fig. 3 Merging a place and transitions into one transition.

and $P_{no\theta}$ represent the two outputs of d_θ (yes, no), one of which corresponds to the failure state determined by the system designer as mentioned in Sect. 3.3. A token is in either $P_{yes\theta}$ or $P_{no\theta}$. The initial marking varies automatically when the proposed method performs model analysis over different combinations of component failures. The transition between these places is determined by (f) or (g). If the number of input arcs of d_θ is only one, the immediate transition t_{g1} of the previous SRN model module, the place $P_{pre\theta}$, and the timed transition $T_{dec\theta}$ is merged into the timed transition T_{g1} with the transition rate $1/t_1$ as shown in Fig. 3, since the reachability graphs are the same.

(c) The Action module contains an Action and an outgoing edge. The Action module is translated into (1) places, transitions and arcs of the control flow model, and (2) a recovery operation model. The transition rate and probability of T_{gexecx} and $t_{successx}$ are determined by the parameters' values of p_x and t_{rx}. The translated SRN model modules (1) and (2) interact with each other as follows. When a token moves to P_{execx} of (1), T_{gexecx} of (2) is enabled. In (2), the token in $P_{beforex}$ moves to $P_{branchx}$ with the transition rate $1/t_2$ [1/h]. Then, the $t_{successx}$ fires at the transition probability p_1 and the token moves to $P_{finishx}$ through P_{opx}. If the Action is associated with DecisionNode d_ψ by the allocation with stereotype ≪control≫, the token in P_{opx} enables the transition t_{gopx} (translated from (f)). On the other hand, when the token is in $P_{branshx}$, t_{failx} fires at the transition probability $1 - p_1$. Then the token moves directly to $P_{finishx}$. When the token is in $P_{finishx}$, $T_{finishx}$ in (1) is enabled and the token moves to $P_{returnx}$. When the token is in $P_{returnx}$, the token in (2) return to $P_{beforex}$ again. When the token is in $P_{beforex}$, $t_{gbeforex}$ in (1) fires, and the token moves to the place of the SRN model module translated from the next AD module. The part drawn by gray solid lines are generated only when the value of *failure_goto* is not "none". The outgoing edge from P_{gotox} is connected to the place of the SRN model module translated from the AD module specified as the value of *failure_goto*.

(d) The FlowFinal module contains a FlowFinal. The FlowFinal module is translated into a place. The place is one of the ending places of the control flow model. The token in this place represents that the recovery operation procedure finished without perfect recovery.

(e) The ActivityFinal module contains an ActivityFinal.

The ActivityFinal module is translated into a place. The place is one of the ending places of the control flow model. The token in the place means that the recovery operation procedure finished with perfect recovery.

(f) The Control with condition module I contains an allocation with stereotype ≪control≫ from an Action a_x to a DecisionNode d_θ. This module is translated into a transition, an input arc and an output arc, which connect the two places of the system state model translated from the DicisionNode module containing d_θ. Direction of the transition is from the failure state to the properly-functioning state determined by the system designer. If yes/no of the output of d_θ corresponds to the failure state, the direction is from $P_{yes\theta}/P_{no\theta}$ to $P_{no\theta}/P_{yes\theta}$. By the guard function g_{opx}, only when the token is in P_{opx}, t_{gopx} fires and the token moves from $P_{yes\theta}/P_{no\theta}$ to $P_{no\theta}/P_{yes\theta}$.

(g) The Control with condition module II contains an allocation with stereotype ≪control≫ from a DecisionNode d_ψ to a DecisionNode d_θ. This module is translated into a transition, an input arc and an output arc, which connect the two places of the system state model translated from the DicisionNode module containing d_θ. Direction of the transition is the same as (f). By the guard function $g_{yes\psi}/g_{no\psi}$, only when the token is in $P_{yes\psi}/P_{no\psi}$, $t_{gyes\psi}/t_{gno\psi}$ fires and the token moves from $P_{yes\theta}/P_{no\theta}$ to $P_{no\theta}/P_{yes\theta}$.

(h) The Prior module contains an allocation with stereotype ≪control≫ from an Action a_x to an Action a_y. The Prior module is translated into an intermediate place between the two places of the system state model translated from the DicisionNode modules containing d_θ, two transitions, two input arcs and two output arcs, which connect the three places. Direction of the transition is the same as (f). By the guard functions g_{opx} and g_{opy}, only when the token moves to P_{opx} and then the token moves to P_{opy}, the token moves from $P_{yes\theta}/P_{no\theta}$ to $P_{no\theta}/P_{yes\theta}$.

We omit translation rules for other nodes/allocations since they are not often used in real recovery operation procedures documents. Extension dedicated to a specific system would be easily done by adding new translation rules if necessary.

3.6 Model Analysis

Based on the synthesized SRN model, the proposed method analyzes the TTR and cost for each combination of component failures. After the proposed method analyzes the TTR, the recovery cost is calculated based on the value of the TTR. This process is repeated automatically for a part of the possible combinations of component failures by changing the initial marking, using pruning rules described in the next subsection.

1) *TTR*

We obtain TTR for each combination of component failures using SPNP [26] by computing the time from when a token is deposited in P_{init} to when the probability at which a token is in P_{recv} reaches 95%. That is, the recovery operation procedure completes with perfect recovery of the system within the time at the probability 95%. The system designer can change the probability if necessary. We assume the transition rates of the timed transitions in the synthesized model are exponentially distributed.

Table 2 Parameters for calculation of the recovery cost.

Parameter	Description
C_{labor}	Labor cost for failure-recovery operations per hour
TTR	TTR obtained by model analysis [hours]
t_{req}	TTR requirement [hours]. This value represents acceptable downtime for the system. If period of down time exceeds this value, penalty cost is charged to the excess.
C_{req}	Cost requirement. This value represents acceptable cost for recovery of the system.
$t_{violation}$	Period of down time exceeding t_{req} [hours]
C_d	Penalty cost for $t_{violation}$ per hour

2) *Recovery Cost*

Using the values of *TTR* obtained by the analysis based on the synthesized model and parameters shown in Table 2, we calculate the recovery cost for each combination of component failures that lead to system failure by the following equation.

$$C_{total} = C_d \cdot t_{violation} + C_{labor} \cdot TTR \tag{2}$$

Generally, if we assign engineers with higher skill to recovery from system failure, $t_{violation}$ and TTR may become smaller. However, at the same time, C_{labor} may become bigger, therefore the total recovery cost may not be always reduced. To cope with this tradeoff, C_{total} is defined as the summation of penalty cost for down time and the labor cost for recovery.

3.7 Identification of SUCCF

For identification of SUCCF, the system designer specifies the cost requirement for recovery for his/her system. For instance, the requirement of recovery cost C_{req} is set to \$300000. The value is different from system to system.

Figure 4 shows the identification algorithm. The input is possible combinations of component failures in a system, and the output is SUCCF. The input combinations are selected in ascending order of the number of the included component failures. Each combination is checked whether its recovery cost violates C_{req}. This process is repeated until all the SUCCF have been identified. The initial combination is randomly selected from the combinations with only one component failure. After a SUCCF is determined, the algorithm does not process the combinations that have larger numbers of component failures than the SUCCF.

In the algorithm, the following steps are performed. In Step 1, we set an initial value of a loop counter for the algorithm ($j = 1$). In Step 2, we select a combination with j component failures. If we identify SUCCF by analyzing the recovery cost of all the possible combinations of component failures, the amount of calculation increases rapidly as

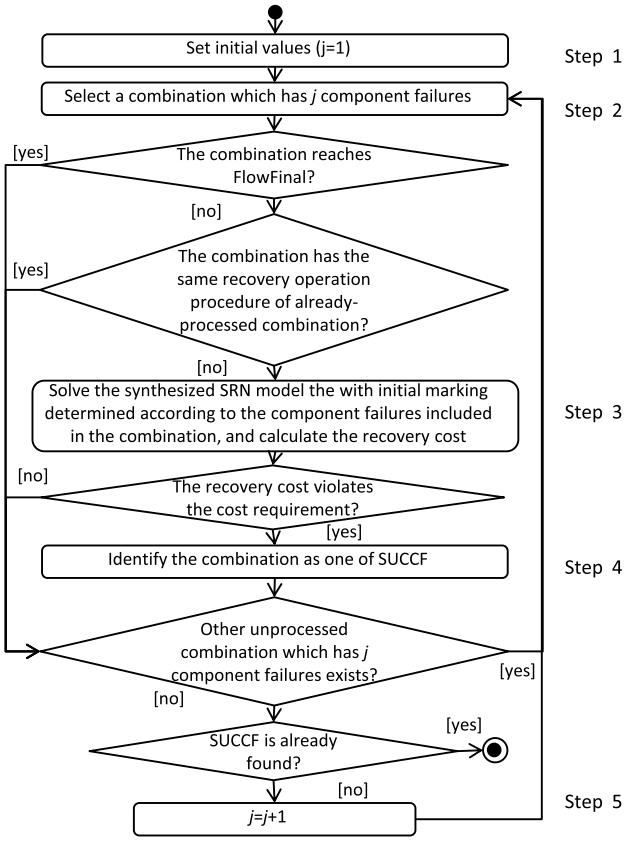


Fig. 4 Algorithm of identification of SUCCF.

the number of components increases. To reduce the amount of calculation, the proposed method utilizes the following pruning rules.

- The combinations of component failures which reach FlowFinal in the Activity Diagram are omitted from analysis, since they do not achieve perfect recovery of the system. For evaluation of these combinations, more appropriate metrics such as data availability [27] can be used.
- If different combinations have the same recovery operation procedure, a repeat of the same analysis is omitted. For instance, the combination of a physical server failure and the combination of the physical server failure and the failure of the OS on the physical server might share the same recovery operation procedure. This is because the OS needs to be recovered after the recovery of the server regardless of the failure of the OS itself.

In Step 3, we solve the synthesized SRN model with the initial marking determined according to the component failures in the combination (e.g., if failure of a component A is included in the combination, a token is placed in the place of the system state model which is determined as failure state of the component A), and calculate the recovery cost. If the recovery cost violates C_{req}, we identify the combination as one of SUCCF (Step 4). For the other unprocessed combinations with the same j component failures, we redo the process from Step 2. After checking all the combinations with j component failures, the algorithm goes for the combinations with $j+1$ component failures (Step 5) if no SUCCF has been found in the combinations with j component failures. Otherwise the identified SUCCF (with j component failures) are output.

The identified SUCCF are finally delivered to the system designer so that he/she can improve the design of the recovery operation procedures.

The identification algorithm is mainly carried out by the enumeration of the combinations of component failures. The enumeration is done in ascending order of the number of included component failures. Two specific pruning rules are proposed to reduce the complexity. For systems where the length of SUCCF is relatively small and the pruning rules can work effectively, the algorithm may quickly identify the SUCCF. The potential limitations of the algorithm are listed below.

- Since the algorithm starts from the combinations with one component failure, if the value of the cost requirement is not small, the algorithm may not end until a relatively big j. As a result, it might take long calculation time to identify the SUCCF.
- If the input combinations seldom have the same recovery operation procedures or rarely reach FlowFinal, the pruning rules may fail to reduce the calculation cost.
- The calculation cost for the parameters (such as TTR and recovery cost) of the algorithm highly depends on the number of states and complexity of the synthesized SRN model as well. To our knowledge, SRN models synthesized from recovery operation procedures of the real systems have relatively large number of states because many detailed recovery operations are required. Reduction of calculation amount for solving SRN model is also essential to apply the proposed method to large systems.

4. A Case Study

We applied the proposed method to a recovery operation procedure for a widely-used commercial database management system Oracle 10 g/11 g as a case study. From an Activity Diagram describing the recovery operation procedure for Oracle 10 g/11 g the proposed method synthesizes the SRN and identifies SUCCF. We evaluate the effect of cross-check for execution of the recovery operation procedure.

4.1 Target Recovery Operation Procedure

Figure 5 shows the recovery operation procedure in the Activity Diagram. The target recovery operation procedure is for failure-recovery of Oracle 10 g/11 g with single instance at archive log mode in Ref. [28]. For mission critical systems, the archive log mode is often used.

In the target recovery operation procedure, there

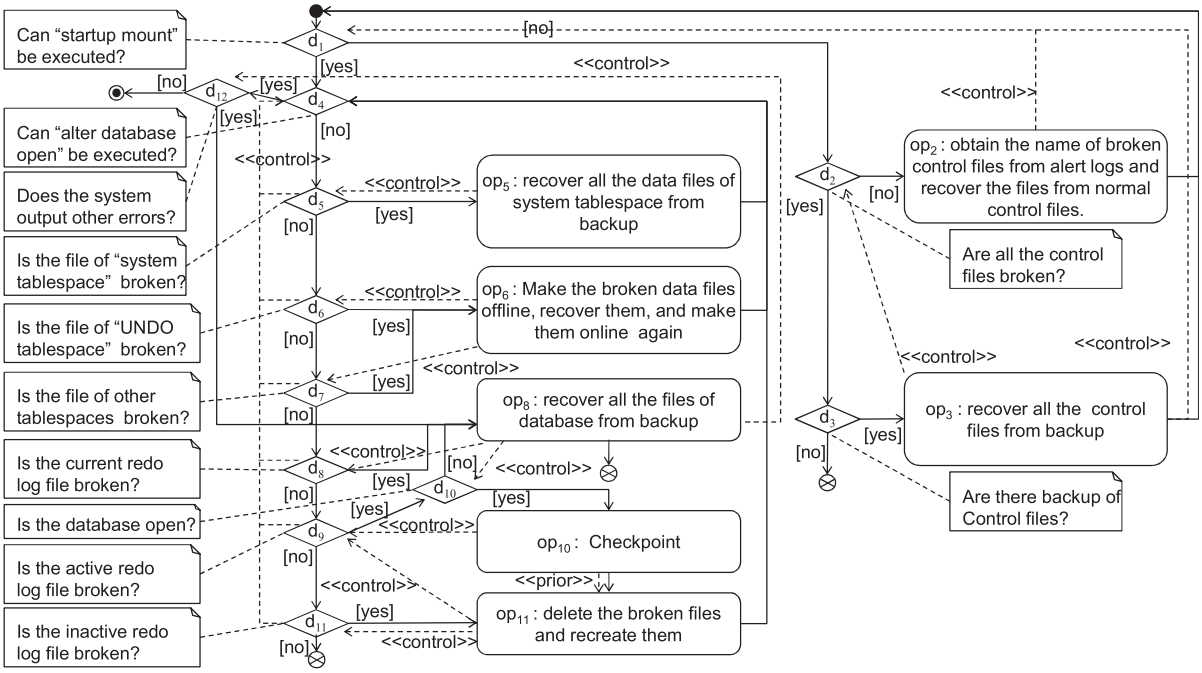


Fig. 5 Recovery operation procedure for Oracle 10 g/11 g.

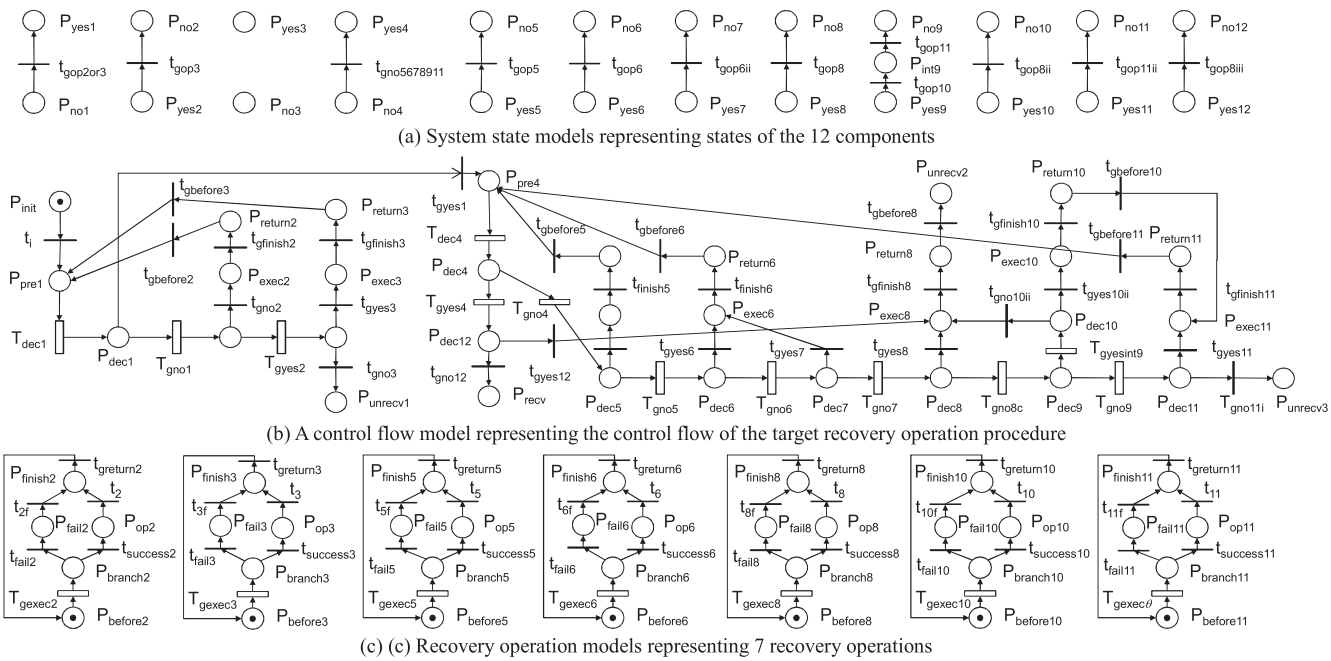


Fig. 6 Synthesized SRN model from the recovery operation procedure, which composed of 3 kinds of sub models (a), (b), and (c).

are twelve components, each of which is in properly-functioning or failed state (e.g., control files exist or not). The current state of y-th component is checked at the DecisionNode d_y ($y = 1, 2, \ldots, 12$) by an operator. What is checked at d_y is described in the Note associated with d_y. Depending on the current states of the twelve components, the required sub operation procedures are determined. (e.g., in Fig. 5, if 1st component is in failed (unavailable) state, the output of DecisionNode d_1 is [no], the output of DecisionNode d_2 is [yes], and "op_2" is executed. After "op_2" recovers the failed state, the output of d_1 is changed to [yes]. Then, the output of DecisionNode d_4 is [yes], the output of DecisionNode d_{12} is [no], and finally we reach ActivityFinal). Each of the sub-procedure includes one recovery operation. The order of their execution dynamically changes depending on the current states of the components. The recovery

Table 3 Parameter values of the synthesized SRN model.

Parameter	Description
p_x	94% for low skill and 99% for a high skill for Action op$_x$ (x = 2,3,5,6,8,10,11)
t_{r2} , t_{r3}	1/3 [hours]
t_{r5}, t_{r10}	1/30 [hours]
t_{r6}, t_{r8}	1/2 [hours]
t_{r11}	1/15 [hours]
t_{cy}	1/30 [hours] for DecisionNode d$_y$(y = 1,2,3,4,5,6,7,8,9,10,11,12)
Failure_goto$_x$	None (x = 2,3,5,6,8,10,11)

operation which recovers failed state of the component is defined as the Action associated with the DecisionNode for the component with stereotype ≪control≫, as described in Sect. 3.5.(f).

For simplicity, recovery operations for other system components such as operating system, a physical server, and a network switch are not included. Because of the space limitation, notes for AD modules (f) and (g) are omitted in Fig. 4. For DecisionNodes d_1 and d_4, "[no] → [yes]" is written in the note associated with stereotype ≪control≫, while "[yes] → [no]" is written for other DecisionNodes.

4.2 Synthesized SRN Model

Figure 6 shows the synthesized SRN model from the recovery operation procedure in the Activity Diagram. The synthesized SRN model is composed of three kinds of sub models: (a) System state models representing the twelve states of the target system (b) a control flow model representing the control flow of the target recovery operation procedure, and (c) recovery operation models representing the seven recovery operations. The guard functions are represented by the same naming convention of the transitions mentioned in Sect. 3.

4.3 Model Analysis to Identify SUCCF

1) *Parameter values*

Based on the synthesized SRN model, *TTR* to each combination of failures are computed using parameter values shown in Table 3 with and without cross-check based on the method in Sect. 3.4. Note that the parameters' values may vary depending on the systems. Since the success probability of the recovery operation of op$_x$ p_x differs depending on the system operator's skill, we set two levels of the system operator's skill: *low* (p_x = 94%) and *high* (p_x = 99%). The values of p_x are determined by the SPAR-H method developed based on the most well-known HRA technique THERP. SPAR-H calculates the human error probability (HEP) of an action (operation) by multiplying the nominal HEP for actions (=0.001) by the following eight performance shaping factor (PSF) multipliers: *Available Time, Stress/Stressors, Complexity, Experience/Training, Procedures, Ergonomics/HMI, Fitness for Duty, Work Processes*. The values of the multipliers are determined according to pre-defined PSF Levels. Here we assume the following assignment of PSF levels.

- Stress/Stressors: System operators would be highly stressed in emergency situations of large-scale disaster, so we select *High* (x 2).
- Complexity: In contrast to daily routine operations, recovery for simultaneous component failures is complex task. So we select *Moderately complex* (x 2).
- Experience/Training: We select *High* (x 0.5) for high system operator's skill, and *Low* (x 3) for low system operator's skill.
- Procedure: In the target recovery operation procedure, general operations for data recovery are concretely described, but what needed to be done in the case of operation failure is not mentioned. So we select *Available, but poor* (x 5).

The remaining 5 PSFs are rated nominal (x 1). Based on the above assignment, HEP for a low-skilled operator is calculated to 6% (p_x = 94%), and HEP for a high-skilled operator is 1% (p_x = 99%). For the rest of parameters of the synthesized model t_{rx} and t_{cy}, we use arbitrary but reasonably selected parameters' values, since some parameters values are not available or confidential.

Then, using the analyzed values of TTR and parameters' values shown in Table 4, the recovery cost for each of the combinations of the component failures is calculated. We assume that the labor cost increases in proportion to the system operator's skill level. We also calculated recovery cost with various values of parameters, but we show only a part of the results due to the space limitation.

As for the cost requirements, we set C_{req} to 8000 and 15000 [$]. The proposed method identifies SUCCF which exceed each value of C_{req}.

Table 4 Parameter values for calculation of recovery cost.

Parameter	Description
C_{labor}	$500 per hour for low skill and $1000 per hour for high skill
TTR	Obtained from model analysis
t_{req}	1 [hours]
C_{req}	$8000 and $15000
C_d	$7000 per hour

Table 5 SUCCF for low-skilled operator(s).

Requirement of recovery cost C_{req} (CDF=95%)	Total number of failures in SUCCF	SUCCF	TTR		Recovery cost C_{total}	
			Without cross-check	With cross-check	Without cross-check	With cross-check
8000	2	F_4,F_6,	2.2	2.06	9,500	9,480
		F_4,F_7	2.31	2.15	10,325	10,200
15000	3	F_4,F_6,F_9	3.13	2.81	16,475	15,480
		F_4,F_6,F_{11}	2.99	2.77	15,425	15,160
		F_4,F_7,F_9	3.24	2.91	17,300	16,280
		F_4,F_7,F_{11}	3.1	2.87	16,250	15,960
		F_1,F_4,F_7	3.03	2.83	15,725	15,640

2) *Analysis results*

Tables 5 and 6 show combinations of SUCCF whose C_{total} exceed C_{req} when system operator's skill level is low and high, respectively. The total numbers of SUCCF are 7 for low skill level and 8 for high skill level.

When system operator's skill level is low, in Table 5, the values of C_{total} of all the SUCCF can be reduced by incorporating cross-check. The results show that for the afore-

Table 6 SUCCF for high-skilled operator(s).

Requirement of recovery cost C_{req} (CDF=95%)	Total number of failures in SUCCF	SUCCF	TTR		Recovery cost C_{total}	
			Without cross-check	With cross-check	Without cross-check	With cross-check
8000	2	F_4,F_6,	2.07	2.05	9,560	11,450
		F_4,F_7	2.17	2.14	10,360	12,260
15000	3	F_4,F_6,F_9	2.85	2.8	15,800	18,200
		F_4,F_6,F_{11}	2.8	2.76	15,400	17,840
		F_4,F_7,F_9	2.94	2.89	16,120	18,650
		F_4,F_7,F_{11}	2.89	2.85	16,520	19,010
		F_1,F_4,F_6	2.76	2.73	15,080	17,570
		F_1,F_4,F_7	2.85	2.82	15,800	18,380

mentioned seven SUCCF, incorporating cross-check is effective to reduce their values of C_{total}. This is because that the amount of decrease in downtime cost by reduction of HEP with cross-check exceeds increase in labor cost for incorporating cross-check.

On the other hand, when system operator's skill level is high, as shown in Table 6, C_{total} with cross-check is always higher than C_{total} without cross-check for both values of C_{req}, although TTR with cross-check is shorter than TTR without cross-check. The result indicates that system operator's skill level is high enough, hence further improvement of the success probability of an operation is not effective to reduce the downtime cost. In this case, cross-check is not a good strategy to reduce the recovery cost. When the values of C_{total} of SUCCF cannot be reduced by incorporating cross check, other measures to reduce TTR such as automation of the manual recovery operations for the components by scripts or introduction of their redundant components should be used to reduce cost. Since introduction of cross-check is not always effective depending on system operator's skill level, we need to investigate appropriate system operator's skill level to reduce the recovery cost.

Although not shown in this section, we also identified the SUCCF with cross-check in the case of a combination of a high-skilled operator and a low-skilled operator. The cost for every SUCCF in such a case is always smaller than that with two high-skilled operators, and larger than that with two low-skilled operators.

In summary, these results indicate that it is important to know appropriate skill levels of system operators in order to reduce the recovery cost by cross-check, and to select effective improvement measures in terms of cost-efficiency.

5. Conclusion

We have presented a technique to identify the SUCCF in disaster recovery scenario by predicting the required cost for recovery from each combination of component failures. To predict TTR and required cost, SRN is automatically synthesized from Activity Diagram representing a recovery operation procedure. Possible combinations of component failures are enumerated using pruning rules on the synthesized SRN and, for given cost requirement, SUCCF are identified by analyzing SRN for each combination. The feasibility of the proposed technique is evaluated in a recovery operation procedure for Oracle 10 g/11 g database management system with and without cross-check of execution of recovery operations. We describe the recovery operation procedure in Activity Diagram and translate it to SRN. For a given parameter values, we identify SUCCF. SUCCF are classified into two cases and effective measures for reducing the recovery cost is discussed for each case.

Future work will focus on improvement of the proposed method by quantitative evaluation of scalability of the proposed method, development of a technique to obtain more accurate parameters' values and their distribution based on both automatic measurement and human reliability analysis. We plan to incorporate the proposed method into our model-based system design tool with automatic analysis of system performance and reliability called CASSI [29] in order to provide more useful information to system designers.

References

[1] G.J. Pai and J.B. Dugan, "Automatic synthesis of dynamic fault trees from UML system models," Proc. 13th Int. Symp. on Software Reliability Engineering (ISSRE 2002), pp 243–254, Nov. 2002.

[2] S. Bernardi, F. Flammini, S. Marrone, J. Merseguer, C. Papa, and V. Vittorini, "Model-driven availability evaluation of railway control systems," Proc. 30th Int. Conf. on Computer Safety, Reliability and Security (SAFECOMP 2011), Sept. 2011.

[3] A. Bondavalli, I. Maizik, and I. Mura, "Automated dependability analysis of UML designs," Proc. 2nd Int. Symp. on Objectoriented Real-time distributed Computing (ISORC'99), May 1999.

[4] R.H. Khan and P.E. Heegaard, "Translation from UML to SPN model: A performance modeling framework for managing behavior of multiple collaborative sessions and instances," Proc. Int. Conf. on Computer Design and Applications (ICCDA 2010), March 2010.

[5] A.E. Rugina, K. Kanoun, and M. Kaâniche, "A system dependability modeling framework using AADL and GSPNs," DSN 2006 Workshops on Software Architectures for Dependable Systems (WADS 2006), pp.14–38, June 2006.

[6] A.E. Rugina, K. Kanoun, and M. Kaâniche, "The ADAPT tool: From AADL architectural models to stochastic Petri nets through model transformation," EDCC 2008, pp.85–90, May 2008.

[7] F. Flammini, S. Marrone, N. Mazzocca, R. Nardone, and V. Vittorini, "Model-driven V&V processes for computer based control systems: A unifying perspective," Proc. 5th Int. Symp. on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA2012), Oct. 2012.

[8] S. Bernardi, J. Merseguer, and D.C. Petriu. "A dependability profile within MARTE," Software and Systems Modeling, vol.12, no.3, pp.313–336, 2011.

[9] S. Bernardi and J. Merseguer, "Performance evaluation of UML design with stochastic well-formed Nets," J. Systems and Software, vol.80, no.11, pp.1843–1865, Nov. 2007.

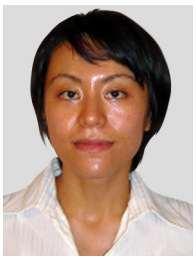
[10] F. Machida, D.S. Kim, E. Andrade, and K.S. Trivedi, "Candy: Component-based availability modeling for cloud service management using SysML," Proc. 30th IEEE International Symposium on Reliable Distributed Systems (SRDS 2011), 2011.

[11] K. Tadano, J. Xiang, M. Kawato, and Y. Maeno, "Automatic synthesis of SRN models from system operation templates for availability analysis," Proc. 30th Int. Conf. on Computer Safety, Reliability and Security (SAFECOMP 2011), pp.296–309, Sept. 2011.

[12] G. Huszerl, I. Majzik, A. Pataricza, K. Kosmidis, and M. Dal Cin, "Quantitative analysis of UML statechart models of dependable systems," The Computer J., vol.45, no.3, pp.260–277. May 2002.

[13] OMG, "OMG Unified Modeling Language (OMG UML), Super-

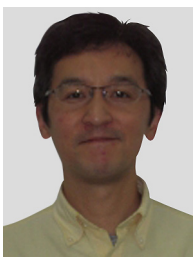
stracture Version 2.3," http://www.omg.org/spec/UML/2.3, accessed June 1, 2012.
[14] OMG, "OMG systems modeling language (OMG SysML) version 1.2," http://www.omg.org/spec/SysML/1.2/, accessed June 10, 2012.
[15] SAE International, "The SAE Architecture Analysis & Design Language (AADL)," http://standards.sae.org/as5506a/, accessed May 12, 2012.
[16] K. Tadano, F. Machida, J. Xiang, and Y. Maeno, "Identification of minimal unacceptable combinations of simultaneous component failures in information systems," Proc. 18th IEEE Pacific Rim Int. Symp. on Dependable Computing (PRDC 2012), pp.21–30, Nov. 2012.
[17] W.E. Vesely, F.F. Goldberg, N.H. Roberts, and D.F. Haasl, "Fault tree handbook," U.S. Nuclear Regulatory Commission, Washington, D.C, Tech. Rep. NUREG-0492, Jan. 1981.
[18] K.S. Trivedi, Probability and Statistics with Reliability, Queuing, and Computer Science Applications, John Wiley, New York, 2001.
[19] J. Williams, "Toward an improved evaluation analysis tool for users of HEART," Int. Conf. on Hazard Identification and Risk Analysis, Human Factors and Human Reliability in Process Safety, pp.261–280, Jan. 1992.
[20] K. Itoh, A. Kuwano, and A. Komatsubara, Human engineering handbook, Asakura Publishing, Tokyo, 2003.
[21] A.D. Swain and H.E. Guttman, Handbook of Human Reliability Analysis with Emphasis on Nuclear Power Plant Applications, NUREG/CR-1278, Sandia National Laboratories, 1983.
[22] E. Hollnagel, Cognitive Reliability and Error Analysis Method (CREAM), Elsevier, 1998.
[23] D.I. Gertman, H.S. Blackman, J.L. Marble, J.C. Byers, and C.L. Smith, "The SPAR-H Human Reliability Analysis Method," NUREG/CR-6883, INL/EXT-05-00509, Idaho National Laboratory, 2005.
[24] V. Gerdes, "HRA Techniques; A selection matrix," Microelectron Reliability, vol.35, no.9-10, pp.1215–1231, 1995.
[25] S. Potter, S.M. Bellovin, and J. Nieh, "Two-person control administration: Preventing administration faults through duplication," Proc. 23rd Conf. on Large installation system administration (LISA'09), pp.15–27, Nov. 2009.
[26] C. Hirel, B. Tuffin, and K.S. Trivedi, "SPNP: Stochastic Petri nets. version 6.0," Proc. 11th Int. Conf. on Modelling Tools and Techniques for Computer and Communication System Performance Evaluation (TOOLS 2000), pp.354–357, March 2000.
[27] X. Yin, J. Alonso, F. Machida, E. Andrade, and K. Trivedi, "Availability modeling and analysis for data backup and restore operations," Proc. 31st IEEE International Symposium on Reliable Distributed Systems (IEEE SRDS 2012), Oct. 2012.
[28] drk7jp, "Recovery operation procedure document and backup scripts for Oracle database," http://www.drk7.jp/MT/archives/001544.html, accessed Jan. 18, 2013.
[29] S. Izukura, K. Yanoo, T. Osaki, H. Sakaki, D. Kimura, and J. Xiang, "Applying a model-based approach to it systems development using sysml extension," Proc. ACM/IEEE 14th Int. Conf. on Model Driven Engineering Languages and Systems, pp.563–577, Oct. 2011.



Kumiko Tadano received the B.S. and M.S. degrees in physics and systems and information engineering from University of Tsukuba in 2005 and 2007, respectively. She is currently a researcher at NEC Corporation. Her research interests include system availability modeling and analysis by taking into account human factors in operation of information systems.

Jianwen Xiang received the B.S. and M.S. degrees in Computer Science from Wuhan University in 1997 and 2000, respectively. He received his PhD in Computer Science from Japan Advanced Institute of Science and Technology in 2005. He is currently a research scientist of NEC Corporation. His research interests include reliability engineering, formal methods, and software engineering.

Fumio Machida received the B.S. and M.S. degrees from Tokyo Institute of Technology in 2001 and 2003, respectively. He is currently a researcher in NEC Laboratory for Analysis of System Dependability (LASD). His research interest includes modeling and analysis of the system/data availability for designing, improving and assuring dependable ICT services. He is a member of IEEE.



Yoshiharu Maeno received the B.S. and M.S. degrees in physics from Tokyo University, Japan, and Ph.D. degree in business science from Tsukuba University. He is a principal researcher at NEC Corporation. He is interested in stochastic process, network theory, statistical analysis, complexity sciences and their application to socio-economic design problems. He is a member of the IEEE, APS, and INSNA. He received the Young Researchers' Award from the IEICE.