Availability Analysis of Systems Deploying Sequences of Environmental-Diversity-Based Recovery Methods

Kun Qiu, Zheng Zheng, *Senior Member, IEEE*, Kishor S. Trivedi, *Fellow, IEEE*, and Ivan Mura, *Member, IEEE*

***Abstract*—Mandelbug-caused software failures are significant threats to system availability, especially in the context of mission-critical and safety-critical systems. However, there is still no systematic method for keeping the software free from Mandelbugs before release. To guarantee the availability of systems suffering from Mandelbugs, environmental-diversity-based fault tolerance techniques have been proposed to recover from the failures caused by them. In this article, we develop and study an analytic model to assess the availability of systems that utilize a sequence of environmental-diversity-based recovery methods. Improving over previous relevant studies, the availability formula we obtain in this article works for any number of recovery methods the system is equipped with; it is also independent on both the nature of those recovery methods and the order of their utilization. In addition, we consider the problem of how to arrange the set of available recovery methods to achieve the largest system availability. Based on the results of our analysis, we develop an open-source tool, called OPENS, which assists in the calculation of the optimal system availability. We validate the effectiveness of the proposed modeling approach in two ways, namely by comparing our results with those obtained for specific systems considered in relevant studies and by conducting numerical analyses for more general scenarios of its application.**

***Index Terms*—Fault tolerance, imperfect coverage, Mandelbug, recovery methods, semi-Markov process (SMP).**

NOMENCLATURE

s	Sequence of recovery methods chosen in the system for failure mitigation.
rr_i	ith recovery method in the sequence s.
$\text{MTTF}(s)$	System's overall mean time to failure, as a function of the sequence of recovery methods s.
$\text{MTTR}(s)$	System's overall mean time to recovery, as a function of the sequence of recovery methods s.
$\text{SSA}(s)$	System's steady-state availability, as a function of the sequence of recovery methods s.
c_i	Coverage of rr_i.
$\overline{c_i}$	Imperfect coverage of rr_i, and $\overline{c_i} = 1 - c_i$.
h_{U_H}	System's MTTF after a human-fix.
h_{DD}	Mean time to execute failure diagnosis and detection.
h_{D_H}	Mean time to execute the human-fix.
h_{D_i}	Mean time to execute rr_i.
h_{U_i}	System's MTTF after a successful recovery by executing rr_i.

Manuscript received January 23, 2020; revised April 18, 2020 and July 3, 2020; accepted August 30, 2020. Date of publication October 6, 2020; date of current version August 31, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61772055 and Grant 61872169, in part by the Technical Foundation Project of Ministry of Industry and Information Technology of China under Grant JSZL2016601B003, and in part by the Equipment Preliminary R&D Project of China under Grant 41402020102. Associate Editor: C. Y. Huang. *(Corresponding author: Zheng Zheng.)*

Kun Qiu and Zheng Zheng are with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China (e-mail: qiukun@buaa.edu.cn; zhengz@buaa.edu.cn).

Kishor S. Trivedi is with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (e-mail: kst@ee.duke.edu).

Ivan Mura is with the Department of Electrical and Computer Engineering, Duke Kunshan University, Kunshan 215316, China (e-mail: ivan.mura@dukekunshan.edu.cn).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TR.2020.3023032

I. INTRODUCTION

AVAILABILITY is a critical attribute of dependability for systems that require continuous operation for months or years, such as servers and safety-critical systems. Even a very short unavailability of such systems could lead to tremendous economic losses. Because of the critical role played by software in current systems, it is pivotal to ensure high software availability in order to guarantee availability of the entire system. Even though considerable effort is spent on techniques such as model checking and testing to prevent failures in the field, failures due to software are too common in practice.

To better address software failures, it is essential to understand their nature. Based on empirical studies on released software projects, including MySQL, Linux, several from NASA, etc., Grottke and Trivedi [1]–[3] observe that the causes of failures, namely software faults (or bugs), can be categorized into two types, namely Bohrbugs and Mandelbugs, based on their reproducibility. The reproducibility indicates the extent of the difficulty in identifying a failure reproduction pattern. Failures caused by Bohrbugs are easily reproduced by applying specific workloads. In contrast, a Mandelbug denotes a fault that is elusive, and failures caused by such bugs are hard to reproduce. According to [4]–[6], the root cause of the elusive behaviors is that Mandelbug failure manifestation is affected not only by the direct program inputs (or workload) but also by the system's

current execution environment. The execution environment may include other software, such as concurrent applications running shared resources, the underlying operating system, as well as hardware entities, such as disks, memory, and firmware. Because of these environmental dependencies, it is hard to systematically detect and remove Mandelbugs. If Mandelbugs escape the testing phase, they can pose severe challenges to the availability of the released system.

On the other hand, the elusive nature of Mandelbugs introduces an opportunity for improving the system availability at the operational time by means of fault tolerance techniques. According to [3], [4], and [7], the software failures caused by Mandelbugs can be recovered by a new type of fault tolerance technique based on environmental diversity. The existing methods falling into this type mainly include restart, reboot, and reconfigure, and we refer to these methods as environmental-diversity-based recovery methods (recovery methods for short) in this context. The rationale behind the claim that the recovery methods can alleviate the impact of Mandelbugs is that the current execution environments will be changed after recovery and the fault activation preconditions may not be satisfied again.

Environmental diversity is easier to implement than traditional fault tolerance techniques, such as design diversity [8] and data diversity [9], to address Mandelbugs-triggered failures. Techniques such as reboot or restart are less complex to deploy and require limited effort for their implementation. Thereby, it is common for a complex software system to be equipped with distinct recovery methods that can be used on several subentities at the same time to cover different types of failures [10]. For instance, the IBM SIP Application Server cluster contains a large number of subentities, including proxy servers and application servers [11], and each subentity is equipped with several recovery methods [11]. In general, it is possible for a complex software system to be equipped with any number of recovery methods, and those recovery methods may be of a very distinct nature and be arranged in different orders to cope with failures. When multiple and diverse recovery methods are available, and many options exist about the order of their application, the following questions become very relevant.

RQ1: How does one evaluate the availability of a system utilizing a sequence of environmental-diversity-based recovery methods?

RQ2: How does one optimize the sequence of the recovery methods to maximize the system availability?

To the best of our knowledge, only the work in [12] and its extension [10] have dealt with the first of the above research questions, and there is still no systematic study heading toward the second research question. The main reason of this limitation is that none considered a scenario where the number of recovery methods is arbitrary and their order of application is not fixed. Regarding RQ1, the focus of [10] and [12] is on the evaluation of the availability for systems equipped with exactly four specific recovery methods, which are arranged in a fixed order. Taking into account the possibility that recovery methods may have very distinct natures and be arranged in different orders makes answering RQ1 require a more general analytic model, one that cannot be obtained straightforwardly from [10]. Regarding the answer to RQ2, it is absolutely not obvious because multiple traits associated with every recovery method require balancing for obtaining the optimal sequence.

In this article, we conduct an analytic study of system availability for systems that are equipped with a sequence of environmental-diversity-based recovery methods. We first propose a general recovery process model to describe the system behavior in addressing failures caused by various types of faults, including both Bohrbugs and Mandelbugs. The model allows the target system to utilize any number of recovery methods that may have a distinct nature and be arranged in different orders. We then utilize the formalism of the semi-Markov process (SMP) [13] to capture the recovery sequences and derive the formula for system availability. Based on the formula, we further study the problem of optimizing the sequence of available recovery methods for the target system. In addition, we develop an open-source tool, namely OPENS, to assist in the system availability calculation and optimization. The validation of our analytic approach is carried out by two steps. Since the system considered in the published work [10] can be viewed as a special instance of the one considered in the present work, we thereby compared the two availability formulas for that particular system, and we found that the two formulas are actually identical. On the other hand, we conducted numerical analyses for more general scenarios of the application of our approach.

Rest of this article is organized as follows. Section II provides the background of this research. Section III describes the proposed recovery process model. Section IV presents the stochastic model of the system and its steady-state availability (SSA) solution. Section V proposes a recovery sequence optimization approach, and Section VI presents the numerical results. Finally, Section VII concludes this article.

II. Background

In this section, we provide the background on Mandelbug-caused software failures and the environmental-diversity-based fault tolerance techniques. After that, we further clarify the motivation for this research.

A. Mandelbug-Caused Software Failures

A software failure is defined as an event that occurs when the delivered service deviates from the correct service [14]. Before a failure event takes place, according to the definition in [14], there are following three necessary phases.

i) Fault (or bug) activation, in which the dormant fault is activated by certain program inputs and an error is produced.

ii) Error propagation, in which the error is successively transformed among internal software subentities, such as procedures and functions.

iii) Failure manifestation, in which the error propagates across the boundary of the software and creates an incorrect service.

After a failure occurs, a natural question is how to reproduce the failure, in order to detect and then fix the bug (fault) that caused the failure. The studies in [1]–[3] observe that various failures could have very different levels of reproducibility. Software failure reproducibility is used to indicate the ability or

the ease in identifying fault activation patterns that cause the failure to manifest. Based on their reproducibility, Grottke and Trivedi [1] classify the bugs that are the causes of failures into two types: Bohrbugs and Mandelbugs. The term Bohrbug is used to denote an easy-to-reproduce bug that can be reproduced deterministically by executing the failure-specific workloads, which stand for the sequence of program inputs that cause the failure to be observed. By contrast, the term Mandelbug is used to denote the hard-to-reproduce bug whose reproduction process is elusive, soft, and not systematic. In other words, under the failure-specific workloads, a Mandelbug may not manifest itself as a failure even after repeated attempts.

After studying failure mechanisms of Mandelbugs, researchers in [4] observed that their elusive behavior is caused by the execution environment of the software. Here, the notion of "execution environment" denotes all other entities around the concerned software entity. It may include software entities, such as concurrent applications running shared resources, the underlying operating system, as well as hardware entities, such as disks, memory, and firmware, and even humans.

We consider three real examples to show how the environments affect the manifestation of Mandelbugs.

1) Bug #38691 in MySQL [15], whose activation is affected by the scheduler of its execution environment. This fault is hard to reproduce and it took almost two months to fix it. The bug report states that the MySQL service component, namely mysqld, can "sometimes" lead to database corruptions. After inspections, it was detected that the corruptions are introduced when optimizing locks for multiple tables in a function called mysql_multi_update_prepare().
2) Bug #11805 in Linux kernel 2.6.26 [16], whose activation is affected by a hardware component of the environment. This fault can be reproduced only when the physical memory is temporarily full. After reproduction, a segmentation fault occurs, and the machine enters a frozen state. This fault was introduced in a function called xfs_buf_get_noaddr().
3) Bug #776706 in Chrome browser 62.0 [17], the error propagation process of which is affected by other firmware inside the operating system. This issue can cause the CPU utilization of the process of the Chrome browser to rise to almost 100%, and it is caused by an incompatibility between the Chrome browser and the GPU firmware.[1]

In general, Mandelbugs can account for a proportion of bugs that cannot be ignored. For example, the proportion of Mandelbugs in MySQL has been estimated to be approximately 38.0% [18]. When a Mandelbug manifests itself, recovering the systems back to normal through repairing is usually a tedious and time-consuming task. For instance, it took almost two months to fix Bug #38691 in MySQL [15]. Therefore, effectively addressing the Mandelbug-caused failures is the key of ensuring the availability of the system. Environmental-diversity-based fault tolerance techniques can be applied as a cost-effective solution, and they are introduced in the following section.

[1][Online]. Available: https://bugs.chromium.org/p/chromium/issues/detail?id=776706

B. Environmental Diversity

Environmental diversity is a set of strategies used to tolerate faults (Mandelbugs) in a software system just as design-diversity-based techniques were meant to tolerate Bohrbugs in a software system. Fault tolerance is a technique for reducing the consequences of residual software faults and thus enhance the dependability of a system [19]. There are following three critical phases in the use of fault tolerance techniques.

- Failure detection, in which an erroneous state is identified by a chosen set of detection mechanisms. Usually, the detection is carried out by verifying the calculated results against expected software outputs or using metamorphic properties [20].
- Failure diagnosis, in which the cause of the erroneous state is assessed on the basis of relevant information, such as system log files, failure phenomenon descriptions, etc. The diagnosis could be carried out either by manual verification or by automatic methods [21].
- Failure recovery, in which predetermined techniques are applied in an attempt to recover the system from the erroneous state to a normal state.

Similar to other methods of fault tolerance, the environmental-diversity-based techniques also contain the above three phases, which are further described in Section III. To better illustrate the specific characteristics of environmental-diversity-based techniques, we briefly introduce traditional fault tolerance techniques, namely design diversity and data diversity, in the following.

In design diversity [8], multiple diverse versions of the target software are separately developed and deployed in redundant configurations. The multiple versions have the same requirements and thereby are expected to provide the same functionality, but they are usually developed by different teams, hoping that the faults introduced by one team will not be the same as those introduced by other teams. When a failure is detected, the software could be recovered by failing over to the other versions (or voting on the results from multiple versions). Since it is clearly time consuming and costly to develop a complex software system, let alone multiple versions, design diversity is thereby not a cost-effective solution for dealing with failures caused by Mandelbugs.

In data diversity [9], once a failure is detected, we need to re-express the failure-causing input into another one and re-execute the software. The newly generated output, if valid, is transformed back into the originally expected output. Development of the input and output re-expression algorithms is the primary challenge of this technique. Since data diversity only considers failure triggers related to program inputs and ignores the effects of program environments, it is of limited value in addressing failures caused by Mandelbugs.

The environmental diversity is proposed to mainly address Mandelbugs-caused failures [3], [7]. An encountered failure is recovered by re-executing the input data on the same or other identical replicas of the software but executing in different execution environments. Several different environmental-diversity-based recovery methods have been proposed in the

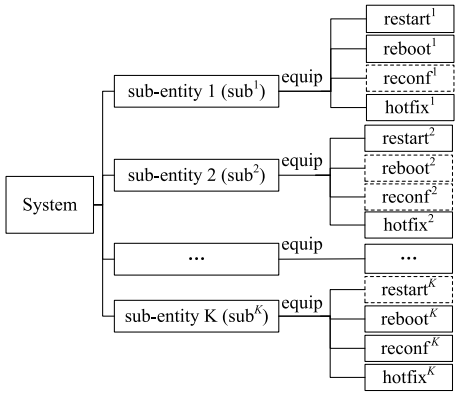

Fig. 1. Example of a system with K subentities and four types of recovery methods.

literature, such as restart, reboot, reconfigure, and hot-fix [7], [10]. The techniques can effectively mitigate the Mandelbug-caused failures because the execution environment will change after the recovery and the fault activation preconditions may not be satisfied again. For example, for Bug #11805 described in the previous section, a software failure is caused by temporary full usage of the physical memory. As a result, rebooting the machine could release and regain the memory and eventually recover the system back to a normal state in a short time. As another example, the failure caused by Bug #776706 of Chrome [17] could be alleviated by reconfiguring the firmware or drivers since it is caused by incompatibility issues.

All fault tolerance approaches make use of redundancy to recover failures. Taking full advantage of the fact that the execution environments can influence the failure manifestation processes of Mandelbugs, environmental diversity can generate redundancy in a shorter time and in more economical ways than do design diversity and data diversity. However, it is also essential to evaluate the effectiveness of environmental-diversity-based techniques in terms of their effect on system availability, which we approach in the following sections.

III. Recovery Process Description

A. Problem Description

In this article, inspired by Trivedi *et al.* [10], [12], which proposed systems equipped by four recovery methods, we consider extending the analysis to systems equipped with a set of M types available recovery methods that can be deployed on different K system subentities. The possible combinations of subentities and recovery method types define the available recovery actions that can be executed in the system to mitigate Mandelbug-caused failures.

Fig. 1 shows an example of a system with K subentities denoted as $\text{Sub} = \{\text{sub}^1, \text{sub}^2, \ldots, \text{sub}^K\}$ and a set with $M = 4$ types of recovery methods: restart, reboot, reconfigure, and hot-fix. Let (a) $\text{Rs} = \{\text{restart}^1, \text{restart}^2, \ldots, \text{restart}^K\}$ denote the set of restarts, and restart^i denote restarting sub^i; (b) $\text{Rb} = \{\text{reboot}^1, \text{reboot}^2, \ldots, \text{reboot}^K\}$ denote the set of reboots, and reboot^i denote rebooting sub^i; (c) $\text{Rc} = \{\text{reconf}^1, \text{reconf}^2, \ldots, \text{reconf}^K\}$ denote the set of reconfigures, and reconf^i denote reconfiguring sub^i; and (d) $\text{Rh} = \{\text{hotfix}^1, \text{hotfix}^2, \ldots, \text{hotfix}^K\}$ denote the set of hot-fixes, and hotfix^i denote fixing sub^i online. It should be noted that not all types of recovery methods may be applicable to certain subentities. For example, if sub^i is a procedure, rebooting it is clearly not possible. Therefore, the set of available recovery methods for the considered system would be a proper subset of $\text{Rs} \cup \text{Rb} \cup \text{Rc} \cup \text{Rh}$. Fig. 1 denotes available recovery types by framing them with a solid-line rectangle.

In general, let $\text{SRM} = \{\text{rr}^1, \text{rr}^2, \ldots, \text{rr}^n\}$ be the set of recovery methods available for the system, a subset of all the deployments of the M types of recovery methods on the K subentities. Consequently, each rr^i is the result of the application of a specific type of recovery method to one of the system subentities. Once a failure occurs, a straightforward strategy is to sequentially apply all the methods in SRM for the failure mitigation. Let $s = \langle \text{rr}_1, \text{rr}_2, \ldots, \text{rr}_n \rangle$ denote the chosen mitigation sequence, where rr_i is an element of SRM and denotes the ith attempted recovery method.

In this article, the quantitative analysis on the considered system aims to answer the following question: *What is the system availability when the system is utilizing a sequence of recovery methods s?* Then, the answer to this question enables us to address the next research question: *What is the optimal sequence s, i.e., the one that maximizes the system availability?* The search space can be extremely large since it corresponds to the set of permutations of SRM. In the following, we first review the related work and then describe our approach.

B. Related Work

The main objective of this work is to conduct a quantitative analysis of the system availability and optimize it. We will use a modeling approach to describe system recovery behaviors when a sequence of recovery methods is applied to mitigate Mandelbug-caused failures. Many studies have been published in the literature presenting availability analysis methods for systems equipped with software rejuvenation techniques [1], which take actions before the manifestations of failures to prevent them from occurring. For example, in [22]–[25], stochastic reward nets models [26] have been applied to evaluate the availability of systems whose time to failure and time to repair can be adequately modeled as exponentially distributed random variables. Continuous-time Markov chains (CTMCs) [27] have been applied in [28] and [29] to model the behaviors of virtual machines. However, we notice there are only a few studies focused on analyzing the availability of systems equipped with recovery methods that take actions after a Mandelbug-caused failure occurs. To the best of our knowledge, the only relevant ones are [12] and its extension [10]. Since software rejuvenation and failure recovery are distinct techniques to increase system availability, the system behaviors resulting from applying them are very different. Therefore, it is infeasible to directly apply the methods developed for the analysis of systems equipped with software rejuvenation techniques to the problem of interest in

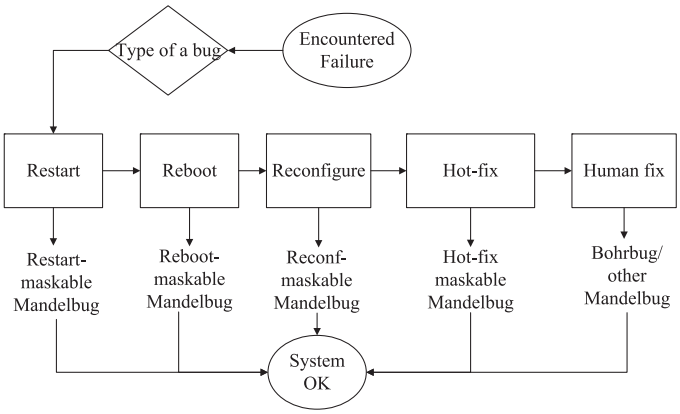

Fig. 2. Flowchart of recovery after a failure for a system equipped with four recovery methods arranged in a given order.

this article. Considering the relevance of [10] and [12] to our work, we provide more details of their work in the following.

The authors in [10] considered two types of recovery strategies: one whereby recovery methods are selected for application according to the result of the bug-type diagnosis step, and another one in which a fixed sequence of available recovery methods is applied. Since we will be focusing on strategies that do not depend on the diagnosis results, the first one is not relevant to the analysis in this article and we shall consider just the latter. Fig. 2 shows the flowchart of the recovery methods employed sequentially to restore the system after the occurrence of a failure. The recovery methods are arranged as follows: restart is first applied, followed by reboot, reconfigure, hot-fix, and human-fix. After the encountered failure is mitigated or repaired, the system enters a normal state denoted as "System OK." Clearly, this is a special case of our more general setting.

In [10], two criteria are considered for arranging the recovery methods in the sequence shown in Fig. 2. First, recovery methods should be sorted in ascending order in expected execution time [7]. Normally, restart is the fastest method, followed by reboot, reconfigure, and hot-fix. Second, for the specific type of system studied in [10], the dormant bugs can be classified into the following four hierarchical categories.

i) Restart-Maskable-Mandelbug, which can be mitigated by restart, reboot, reconfigure, or hot-fix.
ii) Reboot-Maskable-Mandelbug, which can be mitigated by reboot, reconfigure, or hot-fix.
iii) Reconf-Maskable-Mandelbug, which can be mitigated by reconfigure or hot-fix.
iv) Bohrbug and other types of Mandelbug, which can be addressed by hot-fix or human-fix.

The sequence in Fig. 2 is thereby generated by taking full advantage of the above hierarchical relationships among categories.

While the approach proposed in [10] can be applied to systems with a fixed sequence of specific recovery methods, it cannot handle the following situations.

- *Systems can apply recovery techniques on their subentities in a flexible way; therefore the number of recovery techniques may not be fixed.* We consider more general scenarios that could have multiple restarts, reboots, reconfigures, and hot-fixes deployed on different subentities as illustrated in Fig. 1. Here, we are thereby not restricting ourselves to systems with a designated number of recovery methods, which is the assumption in [10].
- *The system mean time to failures (MTTFs) after the failure successfully being mitigated by distinct recovery methods can be different.* According to the results proposed in [6], it is observed that recovery methods may have an impact on the MTTF. For example, Qiu *et al.* [6] find that the average time to failure of a system after being reconfigured is significantly larger than that of a system after being restarted. This phenomenon is understandable because reconfiguring a system involves not only restarting that system but also carrying out additional operations.
- *The recovery methods can be arranged in multiple reasonable sequences.* For example, depending on the specific environment being reconfigured after the failure and the faults that could trigger it, a practitioner can give reconfigure a higher priority over reboot. However, a sequence in which reboot is attempted prior to reconfigure could also be reasonable. It must be noticed that the two sequences could result in very different system availabilities, as we explain in the following.

The above scenarios require a more general approach to model system recovery behaviors, which we will introduce in the following.

C. Our Approach

We first describe the recovery process model proposed for the system considered in this article; this model is described in the form of a flowchart in Fig. 3. As illustrated in this figure, once a failure is detected, the system will attempt all its available recovery methods one after another. If the failure cannot be mitigated by any of the recovery methods, then a human-fix or other back-up methods will be applied to fix the bug. We say that a Mandelbug is masked by a recovery technique if the failure caused by that bug does not reoccur with a high probability after the application of that recovery method. The failure due to that bug is then said to be mitigated.

The first essential aspect of our model is that it considers the impact of applying the available recovery methods in different possible sequences. From the flowchart shown in Fig. 3, it is clear that whether a bug can be masked by a recovery method depends not only on the method itself but also on its predecessor recovery methods in the sequence of recovery methods applied. According to Fig. 3, the failure due to a bug will be mitigated by rr_i when the bug is maskable by rr_i and it is not maskable by $(i-1)$ predecessors of rr_i. Let C_i denote the set of bugs that are actually being masked by rr_i but that cannot be masked by any of the recovery methods applied prior to it, and A_i denote the set of bugs that can be masked by rr_i. Then, the following relationship between C_i and the A_i holds:

$$C_i = A_i \setminus \{A_1 \cup A_2 \cup \ldots \cup A_{i-1}\}.$$

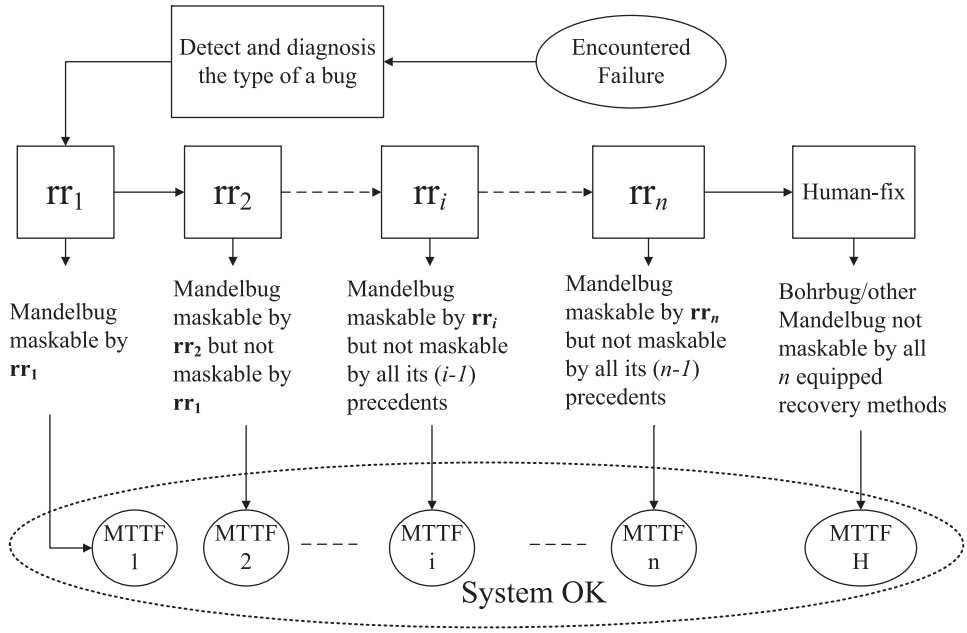

Fig. 3. Flowchart of a system with n recovery methods that may result in distinct system MTTFs.

For instance, if a system only uses two recovery methods x and y, and the sequence $s = \langle x, y \rangle$ is utilized, we have $C_1 = A_x$ and $C_2 = A_x - A_y$. By contrast, if the sequence $s = \langle y, x \rangle$ is utilized, we have $C_1 = A_y$ and $C_2 = A_y - A_x$. In the general case, to take into consideration the influences of the particular sequence of recovery methods on the system availability, we will leverage the concept of conditional probability for the analysis in Section IV.

Another important aspect illustrated by the flowchart is that we associate each recovery method rr_i with a particular value of the system MTTF. We consider that distinct recovery methods may result in different system MTTFs because each of them will change the environment of the system to a certain extent, as explained in Section III-B. To emphasize this point, we detail the working state of the system into $n + 1$ substates, by which a substate "MTTF i" denotes the up state of the system after executing the recovery method rr_i, and "MTTF H" denotes the up state when the system is repaired by the human-fix.

Comparing the flowcharts in Figs. 2 and 3, it is clear that the one shown in Fig. 2 can be treated as a special case of Fig. 3 if we assign restart to rr_1, reboot to rr_2, reconfigure to rr_3, hot-fix to rr_4, and set all "MTTF i" equal to "MTTF H." Based on the above discussions about the system recovery behavior, in the following section, we build an analytic model to carry out a quantitative analysis for the system availability.

IV. System Availability Modeling and Analysis

A. SMP Formulation

In this section, we propose an SMP model to represent and analyze the time-dependent evolution of the state of the system. Our choice of an SMP stems from the following two assumptions that can be justifiably made based on the behaviors of the system.

1) The distribution of the time spent in each one of the steps of the recovery process depicted in Fig. 3 can follow a general distribution. Assuming all sojourn time to be exponentially distributed would make the model much easier to treat analytically. However, the recovery actions are different by their very nature, and using the same constant shape distribution to model all of them would be very restrictive and likely to result in a poorly representative model.
2) The time spent in each step of the flowchart shown in Fig. 3 is independent of the time the system spent in the previously occupied states. We can reasonably assume that executing a specific recovery step requires a time that only depends on the actions executed in the recovery process itself. Furthermore, since no event-time distributions are competing, we can conclude that the system state evolution follows an SMP behavior rather than a more complex one, such as a Markov regenerative process [13].

Based on these assumptions, we resort to an SMP model [13] to represent how the system alternates between up states and recovery stages. Our model of the system is a stochastic process $\{Z(t) \mid t \geq 0\}$, where $Z(t)$ is the state of the system at time t, $t \geq 0$. The state space S_Z of this stochastic process is the set $S_Z = \{DD, U_H, D_H\} \cup \{U_i, D_i | i = 1, 2, \ldots, n\}$. A description of what each state in the SMP model represents is given in Table I. The last column of the table introduces the notation for the cumulative distribution function (CDF) of the sojourn time that the stochastic process spends per visit to each of its states.

The state-transition diagram of the SMP is shown in Fig. 4. Notice that the diagram is very similar to the flowchart in Fig. 3. We can imagine the evolution of the system from state DD as a sequence of transitions that transfer the control from one recovery method to another until that failure is successfully mitigated. Each state in the diagram is labeled with the CDF of

TABLE I
STATES OF THE SMP MODEL

State	System condition being modeled	System available	Sojourn time CDF
DD	A failure has occurred: fault diagnosis and isolation are in progress	NO	$D(t)$
D_i	A failure has occurred and recovery action rr_i is being executed	NO	$G_i(t)$
U_i	Recovery action rr_i completed successfully and the system is working	YES	$F_i(t)$
D_H	None of recover actions rr_i worked and a manual repair is in progress	NO	$G_H(t)$
U_H	The system is working after the successful completion of a manual repair	YES	$F_H(t)$

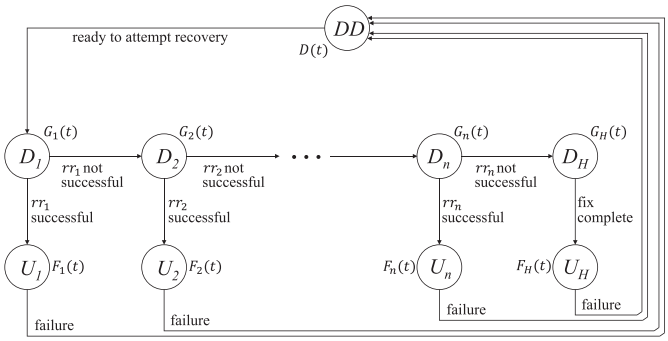

Fig. 4. State-transition diagram for the model of a system with a sequence of n of recovery methods.

the state sojourn time, whose notation is introduced in the last column of Table I. Each transition in the diagram is annotated with the event that in the modeled system causes the state change. We assume that the state sojourn time as well as the events that determine the choice of the next state to transition are independent of time and, hence, the SMP we define is time homogeneous.

Our metric for the quality of a particular sequencing of recovery actions is the SSA of the system. We notice that our SMP model is guaranteed to have a steady-state probability vector, as it has a finite state space and each state is positive recurrent [13], [27], [30], i.e., it has a finite expected return time for any meaningful choice of the holding time distributions.

Obtaining the SSA metric requires evaluating the steady-state probability vector and summing up the probabilities of all the states in which the system is up. In our case, if $\vec{\pi}$ denotes the $(2n+3)$-dimensional vector of the steady-state probabilities, we can compute SSA as follows:

$$\text{SSA} = \pi_{U_H} + \sum_{i=1}^{n} \pi_{U_i}. \tag{1}$$

We will analytically compute the steady-state probability vector for the SMP shortly by using a standard approach that passes through the embedded discrete-time Markov chain of the process. However, before that, we will utilize the notion of imperfect coverage for the recovery methods we consider in the studies of [13], [31], and [32].

B. Imperfect Coverage of Recovery Techniques

We admit the possibility that a given recovery method may not be able to make the environment diverse enough so that the fault is not masked by that step. We define the coverage of a recovery method to be the probability that an encountered failure can be mitigated by its actions. For each rr_i in the sequence s, c_i will denote the conditional probability that the application of rr_i in the sequence s system will bring the system to a state where the Mandelbug will not be reactivated. We assume that with probability 1 the human intervention will bring the system to a state in which the Mandelbug that caused the failure will be not activated. For the sake of a simpler notation, we denote by $\overline{c_i}$ the complementary probability $1 - c_i$, i.e., the conditional probability that rr_i fails to bring the state of the system to one where operations can resume (i.e., the Mandelbug is still active). The latter is known as imperfect coverage [13], [31], [32].

Following the symbol definition used in the last section, we let A_i denote the set of bugs that can be masked by the execution of rr_i, $1 \leq i \leq n$. Then, the coverage c_i is the probability that the encountered failure was caused by a bug in A_i but not in $A_1 \cup A_2 \cdots \cup A_{i-1}$, which we can be written as follows:

$$c_i = P\left(A_i | \overline{A_1}\,\overline{A_2} \cdots \overline{A_{i-1}}\right) = \frac{P\left(A_i \cap \overline{\bigcup_{j=1}^{i-1} A_j}\right)}{P\left(\overline{\bigcup_{j=1}^{i-1} A_j}\right)}. \tag{2}$$

where $P(X)$ denotes the probability that the encountered failure belongs to set X.

We notice that previous studies published in the literature commonly assumed the coverage of recovery method to have the nature of an unconditional probability [13], [31], [32]. Here, we introduce a more general approach to calculate the coverage as given by (2), recognizing the dependence of the coverage of recovery methods on the specific order of their application.

If the intersection of any two sets A_i and A_j is empty, i.e., each recovery method in the sequence can mask a set of Mandelbugs that cannot be masked by any other methods, the conditional probability (2) simplifies to become

$$c_i = P\left(A_i | \overline{A_1}\,\overline{A_2} \cdots \overline{A_{i-1}}\right) = P\left(A_i\right). \tag{3}$$

Therefore, in this independence scenario, the values of coverage no longer depend on the sequence. Their evaluation can be carried out by simpler experiments that aim at determining the individual capability of each recovery technique in masking Mandelbugs.

In the case when the recovery capabilities of recovery methods are in a hierarchical relation (such as the four recovery methods described in Section III-B), it means that $A_i \subset A_{i+1}$, $i = 1, 2, \ldots, n-1$, and the expression in (2) simplifies as follows:

$$c_i = P\left(A_i | \overline{A_1}\,\overline{A_2} \cdots \overline{A_{i-1}}\right) = P\left(A_i | \overline{A_{i-1}}\right). \tag{4}$$

Since $A_i\overline{A_{i-1}} = A_i \setminus A_{i-1}$ and we are guaranteed that the unconditional coverage factors will be such that $P(A_{i-1}) < P(A_i)$ for $i = 2, 3, \ldots, n$, we can rewrite (4) as follows:

$$c_i = \frac{P(A_i) - P(A_{i-1})}{1 - P(A_{i-1})}. \tag{5}$$

Notice that a hierarchical arrangement is compliant with the rationale of an ordered deployment of recovery methods. Whenever the sequence s of recovery methods has such a property, we can apply the formula in (5) to estimate the coverage factors c_i.

Otherwise, the calculations of the c_i should follow a two-step procedure. First, draw the Venn diagram of the set of maskable bugs for all recovery methods and the human-fix method. Then, calculate c_i based on (2) and the estimation of each set size. For example, suppose a system is equipped with a sequence of recovery methods $s = \langle \mathrm{rr}_1, \mathrm{rr}_2, \mathrm{rr}_3, \mathrm{rr}_4 \rangle$. Let us define

$$p_i = P\left(A_i \cap \overline{\bigcup_{j=1}^{i-1} A_j}\right),\ i = 1, 2, \ldots, 5. \tag{6}$$

Since we assume that all bugs can be recovered by a human-fix, A_5 is thereby the universal set, $P(A_5) = 1$, and $p_1 + p_2 + p_3 + p_4 + p_5 = 1$. Following (2), we have

$$\begin{aligned}
c_1 &= P(A_1) = p_1 \\
c_2 &= P\left(A_2 | \overline{A_1}\right) = \frac{P\left(\overline{A_1} A_2\right)}{P\left(\overline{A_1}\right)} = \frac{p_2}{1 - p_1} \\
c_3 &= P\left(A_3 | \overline{A_1}\,\overline{A_2}\right) = \frac{P\left(\overline{A_1}\,\overline{A_2} A_3\right)}{P\left(\overline{A_1}\,\overline{A_2}\right)} \\
&= \frac{P\left(\overline{A_1}\,\overline{A_2} A_3\right)}{P\left(\overline{A_1}\right) - P\left(\overline{A_1} A_2\right)} = \frac{p_3}{1 - p_1 - p_2} \\
c_4 &= P\left(A_4 | \overline{A_1}\,\overline{A_2}\,\overline{A_3}\right) = \frac{P\left(\overline{A_1}\,\overline{A_2}\,\overline{A_3} A_4\right)}{P\left(\overline{A_1}\,\overline{A_2}\,\overline{A_3}\right)} \\
&= \frac{P\left(\overline{A_1}\,\overline{A_2}\,\overline{A_3} A_4\right)}{P\left(\overline{A_1}\,\overline{A_2}\right) - P\left(\overline{A_1}\,\overline{A_2} A_3\right)} = \frac{p_4}{1 - p_1 - p_2 - p_3}.
\end{aligned} \tag{7}$$

In real applications, the values of p_i can be estimated by the proportion of bugs that fall into the corresponding set; see, for instance, the related empirical studies [2], [3], and [33] on this topic. However, here we will not touch upon the general problem of estimating the values of coverage but leave the topic as a future work.

C. System SSA Solution

In this section, we obtain an explicit analytical expression of SSA, by determining the steady-state probabilities of the SMP states and replacing them in (1). Informally, we consider the sequence of states of the SMP process at the epochs of state transitions. Such a sequence satisfies the Markov property and indeed forms a discrete-time Markov chain, from whose steady-state analysis, we can obtain the steady-state probability vector of the SMP [13].

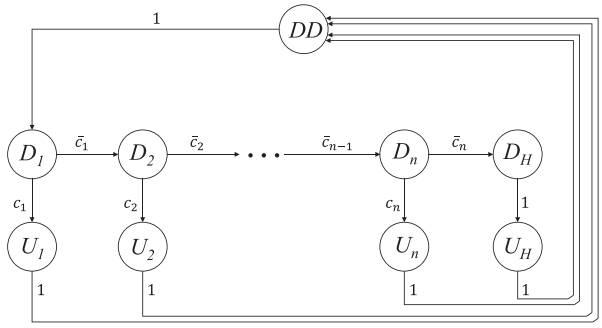

Fig. 5. State-transition diagram for the EDTMC of the SMP model.

Formally, let $t_0 = 0$, t_1, $t_2, \ldots, t_n$ denote the time epochs at which the SMP $\{Z(t) \mid t \geq 0\}$ undergoes a state transition. The sequence of states $\{X_n = Z(t_n) \mid n \geq 0\}$ forms an embedded discrete-time Markov chain (EDTMC, hereafter), whose state space S_X is the same as S_Z, the state space of the SMP.

Based on our assumption of time homogeneity for the SMP, we are guaranteed that the EDTMC satisfies the homogeneity property as well and, hence, it is fully characterized by its one-step transition probability matrix $P = [p_{j,k}]$. Note that the assumption of the Markov chain requires that $\sum_k p_{j,k} = 1$. The nonzero entries of the transition probability matrix P of the EDTMC are defined as follows:

$$p_{j,k} = \begin{cases} 1 & j = DD \text{ and } k = D_1 \\ c_i & j = D_i \text{ and } k = U_i,\ i \in I \\ \overline{c_i} & j = D_i \text{ and } k = D_{i+1},\ i \in I \\ 1 & j = U_i \text{ and } k = DD,\ i \in I \end{cases} \tag{8}$$

where I is defined as the set $I = \{1, 2, \ldots, n\} \cup \{H\}$. The state-transition diagram of the EDTMC is isomorphic to that of the SMP, and it is shown in Fig. 5. Since it is a finite-state aperiodic discrete-time Markov chain, it has a unique steady-state probability vector [32].

Let us denote by $\vec{\nu}$ the vector of steady-state probabilities of the EDTMC $\{X_n \mid n \geq 0\}$, $\vec{\nu} = [\nu_{DD}, \nu_{D_1}, \nu_{U_1}, \nu_{D_2}, \nu_{U_2}, \ldots, \nu_{D_n}, \nu_{U_n}, \nu_{D_H}, \nu_{U_H}]$. According to [13], $\vec{\nu}$ is the solution of the following set of linear equations:

$$\begin{aligned} \vec{\nu} &= \vec{\nu} \cdot P \\ \text{subject to } & \vec{\nu} \cdot \vec{e} = 1 \end{aligned} \tag{9}$$

where P is the transition probability matrix defined in (8), and $\vec{e}$ is a properly sized column vector with all entries equal to 1. By first assuming that ν_{DD} is known, using successive substitutions [13] to solve the system of linear equations, we obtain the following solution for the entries of vector $\vec{\nu}$:

$$\begin{aligned}
\nu_{D_i} &= \nu_{DD} \cdot \gamma_i,\ i = 1, 2, \ldots, n \\
\nu_{D_H} &= \nu_{DD} \cdot \gamma_{n+1} \\
\nu_{U_i} &= \nu_{D_i} \cdot c_i,\ i = 1, 2, \ldots, n \\
\nu_{U_H} &= \nu_{D_H}
\end{aligned} \tag{10}$$

where, for the sake of a simpler notation, we defined $\gamma_i = \prod_{j=1}^{i-1} \overline{c_j}$ for $i \geq 2$ and $\gamma_1 = 1$. From the above equations, by

using the normalizing condition in (9), we obtain the following expression for ν_{DD}:

$$\nu_{DD} = \frac{1}{2 + \sum_{i=1}^{n+1} \gamma_i}. \tag{11}$$

As described in [13], from the steady-state solution $\vec{\nu}$ of the EDTMC, we can obtain the vector $\vec{\pi}$ of the SMP steady-state probabilities. Let us denote by h_i the average holding time (or sojourn time)[2] of state i of the SMP, where index i takes values in the state space S_Z. We can obtain the entries of vector $\vec{\pi}$ as follows:

$$\pi_i = \frac{\nu_i \cdot h_i}{\sum_{j \in S_Z} \nu_j \cdot h_j},\ i \in S_Z. \tag{12}$$

Then, using (12) and substituting into (1), we obtain the following expression for the $\mathrm{SSA}(s)$ of the system:

$$\mathrm{SSA}(s) = \frac{\nu_{DD} \cdot \gamma_{n+1} \cdot h_{U_F}}{\Delta} + \sum_{i=1}^{n} \frac{\nu_{DD} \cdot \gamma_i \cdot c_i \cdot h_{U_i}}{\Delta}$$

$$= \frac{\nu_{DD}}{\Delta} \left(\gamma_{n+1} \cdot h_{U_F} + \sum_{i=1}^{n} \gamma_i \cdot c_i \cdot h_{U_i} \right) \tag{13}$$

where Δ is the SMP steady-state normalizing constant [13] such that

$$\Delta = \nu_{DD} \cdot h_{DD} + \sum_{i=1}^{n} \nu_{DD} \cdot \gamma_i \cdot h_{D_i} + \nu_{DD} \cdot \gamma_{n+1} \cdot h_{D_H}$$

$$+ \sum_{i+1}^{n} \nu_{DD} \cdot \gamma_i \cdot c_i \cdot h_{U_i} + \nu_{DD} \cdot \gamma_{n+1} \cdot h_{U_H}.$$

With some algebraic manipulations, we can simplify (13) and obtain the following expression for $\mathrm{SSA}(s)$ of the system:

$$\mathrm{SSA}(s) = \left(1 + \frac{h_{\mathrm{DD}} + \sum_{i=1}^{n} \gamma_i \cdot h_{D_i} + \gamma_{n+1} \cdot h_{D_H}}{\sum_{i=1}^{n} \gamma_i \cdot c_i \cdot h_{U_i} + \gamma_{n+1} \cdot h_{U_H}} \right)^{-1}. \tag{14}$$

Notice that the expression in (14) for $\mathrm{SSA}(s)$ is in the standard form given as

$$\mathrm{SSA}(s) = \left(1 + \frac{\mathrm{MTTR}(s)}{\mathrm{MTTF}(s)} \right)^{-1} \tag{15}$$

where $\mathrm{MTTR}(s)$ denotes the system's overall mean time to repair (or recovery) and $\mathrm{MTTF}(s)$ denotes the system's overall MTTF.

Equation (14) provides a formula for the system availability as a function of the characteristics of the recovery methods, i.e., the expected duration of their execution, the expected uptime the system experiences after their successful completion, and the coverage each of them provides against failures. Notice that the specific sequencing of the recovery methods is taken into account in (14) by the coefficients γ_i.

To verify our solution, we have compared the analytic availability formula obtained in [10] and the one obtained in this article. We found that the two formulas provide the same result under the settings given in [10]. The detailed comparison process is presented in the appendix. We further validate the solution by the numerical analyses in Section VI.

[2]If $H_i(t)$ denotes the CDF of the sojourn time in state i, then the expected sojourn time is given by $h_i = \int_0^\infty (1 - H_i(u))du$.

V. Optimal Recovery Sequence Analysis

In addition to computing the availability for a system with a given sequence of recovery methods, it is important to determine the optimal sequence that can result in the largest system availability for a chosen set of recovery methods. Recall that, we let $\mathrm{SRM} = \{\mathrm{rr}^1, \mathrm{rr}^2, \ldots, \mathrm{rr}^n\}$ denote the set of recovery methods available in the system. Each permutation of the set I of indexes $\{1, 2, \ldots, n\}$ defines a possible sequence s of recovery methods to be applied for failure mitigation. Let us denote by $P(\mathrm{SRM})$ the set of all $n!$ possible permutations of SRM. With this notation, the availability optimization problem can be formally described as the one that finds a sequence s^* such that

$$\mathrm{SSA}(s^*) = \max_{s \in P(\mathrm{SRM})} \mathrm{SSA}(s).$$

Owing to (15), we can restate the maximization problem as the following minimization problem:

$$\min_{s \in P(\mathrm{SRM})} \left\{ \frac{\mathrm{MTTR}(s)}{\mathrm{MTTF}(s)} \right\}. \tag{16}$$

This sequence optimization problem is not trivial at all. The major source of its complexity is the dependence of coverage values on the sequence of recovery methods. Indeed, by its very definition, the value of c_i depends on which recovery methods are executed prior to rr^i in the sequence. To approach the problem of identifying optimal sequences, let us first assume that the recovery methods chosen are such that their order of execution does not affect the coverage probabilities. An example of such a system is the one consisting of a set of modules such that each module is equipped with one recovery method for dealing with a specific set of failures. For such a system, the set of failures that can be masked by one recovery method is disjoint from the set of any other one.

With this assumption, each recovery method rr^i is quantitatively characterized by the tuple (c^i, h_D^i, h_U^i), where c^i is the coverage of rr^i, h_D^i the rr^i mean execution time, and h_U^i the system MTTF after successful failure mitigation by rr^i, respectively. When using the notation with subscripts in $s = \langle \mathrm{rr}_1, \mathrm{rr}_2, \ldots, \mathrm{rr}_n \rangle$, by rr_i we mean the ith recovery method in the sequence, which does not necessarily correspond to rr^i, the ith recovery method in set SRM. To maintain a correspondence between the recovery methods in a sequence s and those in the set SRM, we introduce an index mapping function $\delta : I \to I$ such that, for each index i of rr_i in a sequence s, δ returns the corresponding index j of rr^j in SRM. Thus, for instance, $h_{D_{\delta(i)}}$ and $c_{\delta(i)}$ would be the expected execution time and the coverage of the ith recovery method in the sequence, respectively. If $\delta(i) = j$, i.e., the ith method in the sequence is rr^j, then $h_{D_{\delta(i)}} = h_D^j$ and $c_{\delta(i)} = c^j$.

In the following, we present three scenarios for which the optimal recovery sequence s^* can be found by a simple ordering of the tuples characterizing the recovery methods.

A. Ordering Based on Execution Time

We consider in this scenario a set of recovery methods that all have the same c^is and h_U^is. The following proposition provides a simple algorithmic approach to solve the SSA(s) optimization problem.

Proposition 1: If the recovery methods in SRM are such that the following statements hold:

1) $h_U^i = h_U$;
2) $c^i = c$;

for $i = 1, 2, \ldots, n$, then the optimal sequence $s^* = \langle \mathrm{rr}_1, \mathrm{rr}_2, \ldots, \mathrm{rr}_n \rangle$ is such that $h_{D_{\delta(i)}} \leq h_{D_{\delta(i+1)}}$, $i = 1, 2, \ldots, n-1$.

Proof: Owing to the assumptions in 1) and 2) above, we have $h_{U_i} = h_U$ and $c_i = c$, respectively. Then, the expression to be minimized in (16) can be rewritten as follows:

$$\frac{h_{DD} + \sum_{i=1}^{n} (1-c)^{i-1} \cdot h_{D_{\delta(i)}} + (1-c)^n \cdot h_{D_H}}{\sum_{i=1}^{n} (1-c)^{i-1} \cdot c \cdot h_U + (1-c)^n \cdot h_{U_H}}. \quad (17)$$

Since the denominator of the previous expression does not depend on the chosen order of the sequence, then optimizing SSA(s) of the system is equivalent to minimizing the numerator. Furthermore, h_{DD} and h_{D_H} are both constants that are added to the numerator for each sequence, and therefore our problem becomes that of finding

$$\min_{s \in P(\mathrm{SRM})} \sum_{i=1}^{n} (1-c)^{i-1} \cdot h_{D_{\delta(i)}}.$$

Because the coefficients $(1-c)^{i-1}$ in the linear combination above are decreasing with i, the minimum value will be achieved by a sequence in which $h_{D_{\delta(i)}}$s are arranged in a nondecreasing order, thus concluding the proof. □

Proposition 1 shows that when both the coverage and the system MTTF after successful failure mitigation are the same among all recovery methods in SRM, a recovery sequence s that gives higher priority to recovery methods with the shorter expected execution time will result in the best SSA(s).

B. Ordering Based on Expected Time to Failure

We consider here the case where the expected execution time and the coverage values of recovery methods are the same.

Proposition 2: If the recovery methods in SRM are such that the following statements hold:

1) $h_D^i = h_D$;
2) $c^i = c$;

for $i = 1, 2, \ldots, n$, then the optimal sequence $s^* = \langle \mathrm{rr}_1, \mathrm{rr}_2, \ldots, \mathrm{rr}_n \rangle$ is the one such that $h_{U_{\delta(i)}} \geq h_{U_{\delta(i+1)}}$, for $i = 1, 2, \ldots, n-1$.

Proof: Owing to assumptions 1) and 2), we have $h_{D_i} = h_D$ and $c_i = c$, respectively. Then, the expression in (16) can be rewritten as follows:

$$\frac{h_{DD} + \sum_{i=1}^{n} (1-c)^{i-1} \cdot h_D + (1-c)^n \cdot h_{D_H}}{\sum_{i=1}^{n} (1-c)^{i-1} \cdot c \cdot h_{U_{\delta(i)}} + (1-c)^n \cdot h_{U_H}}. \quad (18)$$

Since the numerator of the previous expression does not depend on the chosen order of the sequence, then optimizing $SSA(s)$ of the system is equivalent to maximizing the denominator. Moreover, since h_{U_H} is a constant time included for each sequence, we can restate the maximization problem as follows:

$$\max_{s \in P(\mathrm{SRM})} \sum_{i=1}^{n} (1-c)^{i-1} \cdot c \cdot h_{U_{\delta(i)}}.$$

Following the same argument we used when proving the previous proposition, the optimal result is achieved by the sequences in which $h_{U_{\delta(i)}}$s are arranged in a nonincreasing order, thus completing the proof. □

Proposition 2 shows that when both the value of coverage and the expected execution time are the same among all recovery methods in SRM, a recovery sequence that gives higher priority to the recovery method with larger system MTTF after successful failure mitigation will result in the best SSA(s).

C. Ordering Based on Coverage

We consider in this scenario a set of recovery methods that all have the same h_D^is and h_U^is. The following proposition provides a simple algorithmic approach to solve the SSA(s) optimization problem.

Proposition 3: If the recovery methods in SRM are such that the following statements hold:

1) $h_D^i = h_D$;
2) $h_U^i = h_U$;

for $i = 1, 2, \ldots, n$, then the optimal sequence $s^* = \langle \mathrm{rr}_1, \mathrm{rr}_2, \ldots, \mathrm{rr}_n \rangle$ is such that $c_{\delta(i)} \geq c_{\delta(i+1)}$, $i = 1, 2, \ldots, n-1$.

Proof: Owing to the assumptions in 1) and 2), we can rewrite the expression in (16) as follows:

$$\frac{h_{DD} + h_D + h_D \sum_{i=1}^{n} \prod_{j=2}^{i-1} \overline{c_{\delta(j)}} + \prod_{j=1}^{n} \overline{c_{\delta(j)}} \cdot h_{D_H}}{\sum_{i=1}^{n} \prod_{j=1}^{i-1} \overline{c_{\delta(j)}} \cdot c_{\delta(i)} \cdot h_U + \prod_{i=1}^{n} \overline{c_{\delta(i)}} \cdot h_{U_H}}. \quad (19)$$

Since the second term in the denominator in the above expression can be written as follows:

$$\prod_{i=1}^{n} \overline{c_{\delta(i)}} \cdot h_{U_H} = \prod_{i=1}^{n} \overline{c_{\delta(i)}} \cdot h_U + \prod_{i=1}^{n} \overline{c_{\delta(i)}} \cdot (h_{U_H} - h_U) \quad (20)$$

then, the denominator of (19) takes the following form:

$$h_U \left[\sum_{i=1}^{n} \prod_{j=1}^{i-1} \overline{c_{\delta(j)}} \cdot c_{\delta(i)} + \prod_{i=1}^{n} \overline{c_{\delta(i)}} \right] + \prod_{i=1}^{n} \overline{c_{\delta(i)}} \cdot (h_{U_H} - h_U). \quad (21)$$

We prove in the appendix (see Lemma 1) that the expression within square brackets is equal to 1, thus (19) can be rewritten as follows:

$$\frac{h_{DD} + h_D + h_D \sum_{i=1}^{n} \prod_{j=2}^{i-1} \overline{c_{\delta(j)}} + \prod_{j=1}^{n} \overline{c_{\delta(j)}} \cdot h_{D_H}}{h_U + \prod_{j=1}^{n} \overline{c_{\delta(j)}} \, (h_{U_H} - h_U)}. \quad (22)$$

Since the denominator of the previous expression does not depend on the chosen order of the sequence, then optimizing

SSA(s) of the system is equivalent to minimizing the numerator. Moreover, h_{DD}, h_D, and h_{D_H} are constants, and therefore our problem becomes that of finding

$$\min_{s\in P(\text{SRM})} \sum_{i=1}^{n} \prod_{j=2}^{i-1} \overline{c_{\delta(j)}}.$$

Let us denote by $T(s) = \overline{c_{\delta(1)}} + \overline{c_{\delta(1)}} \cdot \overline{c_{\delta(2)}} + \cdots + \overline{c_{\delta(1)}} \cdot \overline{c_{\delta(2)}} \cdot \ldots \cdot \overline{c_{\delta(n-1)}}$ the function of the coverage factors to be minimized. We prove by contradiction that the optimal sequences must be such that $c_{\delta(i)} \geq c_{\delta(i+1)}$, $i = 1, 2, \ldots, n-1$. Let us assume that there exists an index k, $1 \leq k < n$, for which $c_{\delta(k)} < c_{\delta(k+1)}$, i.e., the condition stated in the proposition is not satisfied by the optimal sequence s^*. Then, we can construct another sequence s' that has the same order of recovery methods as s^* except for the kth and $(k+1)$th items, which are swapped. Then, it is easy to check that

$$T(s^*) - T(s') = \overline{c_{\delta(1)}} \cdot \overline{c_{\delta(2)}} \cdot \ldots \cdot \overline{c_{\delta(k-1)}} \left(\overline{c_{\delta(k)}} - \overline{c_{\delta(k+1)}}\right)$$

because all the other terms in the difference cancel out. Since we assumed $c_{\delta(k)} < c_{\delta(k+1)}$, then we have $\overline{c_{\delta(k)}} > \overline{c_{\delta(k+1)}}$; therefore, $T(s^*) - T(s') > 0$, which means s^* is not optimal, thus generating a contradiction and concluding the proof. □

Proposition 3 shows that when both the expected execution time and the system MTTF after successful failure mitigation are the same among all recovery methods in SRM, a recovery sequence s that gives higher priority to recovery methods with larger values of coverage will result in the best SSA(s).

Propositions 1–3 do not cover all the situations in optimizing the sequence of a chosen set of recovery methods. However, their preconditions can be easily checked and, when applicable, they can effectively reduce the search time to find the optimal sequence.

VI. Numerical Analyses

In this section, we explore the application of the analytic results obtained in Sections IV and V to a set of scenarios that can characterize several interesting cases of software systems. Moreover, we realize a validation step for the model by checking that its results respect a set of partial ordering properties that can be stated for the analyzed cases of systems. Notice that even though the expression for SSA(s) in (14) appears to be a closed formula, it in fact depends on the coverage parameters c_is, whose values may vary with the chosen order of recovery methods in the sequence. In the analysis presented in this section, we have the same assumption made for Section V by assuming that the recovery methods chosen are such that their order of execution does not affect the coverage probabilities and, hence, the coverage parameters of recovery methods remain constant in the considered cases. To find the optimal sequence, we shall be using a combination of exhaustive search and the sorting approach. Our analysis process encompasses the following three steps.

Setup. To identify key system characteristics to be considered as independent variables in the analysis of the response, i.e., system availability. These characterizing traits are called factors.[3] We shall also be defining for each factor a set of levels that correspond to specific conditions that can be realistically found in a software system. The possible assignments of levels to the factors define a sample of interesting system configurations for analysis. Each assignment (called treatment) is representative of the execution environment and recovery characteristics of a class of availability-critical software systems.

Evaluation. To determine the optimal sequencing of recovery methods for each treatment and the corresponding system availability, by using exhaustive search and the results proved by Propositions 1–3, when possible.

Validation. To test to what extent the results obtained with our modeling approach respect a set of qualitative properties that can be stated on the availability of the system.

In the following, we provide details about the process outlined above.

A. Setup

Since the response is SSA(s), we identify factors by screening those aspects of the system that can have an effect on the parameters c_is, h_{D_i}s, and h_{U_i}s appearing in (14) that can change the magnitude of the parameters or affect their relative differences.

Let us first consider the coverage parameters c_is. Since the recovery techniques in SRM are meant to address Mandelbug-caused failures, we expect the proportion of this bug type to be an important factor. Failures caused by Bohrbugs cannot be mitigated by methods in SRM, so the magnitude of the coverage parameters is affected by the Mandelbug proportion. According to [2], [10], and [33], the Mandelbug proportion is highly dependent on the system complexity. A system can be coarsely classified as having a low, medium, or high complexity, and the higher the complexity of the system, the larger the proportion of Mandelbugs. Accordingly, we classify the factor of the Mandelbug proportion (MP, hereafter) into three levels: low, medium, and high levels. The variability of the coverage of recovery methods, i.e., the difference between the magnitudes of the c_i parameters, will be influenced by the nature of the recovery method itself and the architecture of the system. For instance, if SRM consisted of n recovery methods, each one of type restart, where each restart is to be applied to one of the n independent subsystems of a large modular system, then we expect that the coverage parameters would be quite similar in magnitude, i.e., there would be reduced relative differences between them. In contrast, if restart, reconfigure, and hot-fix recovery methods were considered, or if they were applied to subsystems of different scales in the system, then the values of coverage would be very different. Since there is very limited experimental data to rely upon and we would like to preserve generality, we introduce in our analysis an abstract factor called coverage relationship (CR, hereafter), for which we design two levels: similar level, when the coverage of recovery methods have (almost) the same magnitudes, and escalated level, in which the coverage of recovery methods take values that differ significantly.

[3]We borrow the terminology used here from the design of experiments branch of statistics; see, for instance [34].

TABLE II
EXPERIMENTAL PLAN

Treatment	MP	CR	MRR	MFR
T_1	Low	Similar	Similar	Similar
T_2	Low	Similar	Similar	Escalated
T_3	Low	Similar	Escalated	Similar
T_4	Low	Similar	Escalated	Escalated
T_5	Low	Escalated	Similar	Similar
T_6	Low	Escalated	Similar	Escalated
T_7	Low	Escalated	Escalated	Similar
T_8	Low	Escalated	Escalated	Escalated
T_9	Medium	Similar	Similar	Similar
T_{10}	Medium	Similar	Similar	Escalated
T_{11}	Medium	Similar	Escalated	Similar
T_{12}	Medium	Similar	Escalated	Escalated
T_{13}	Medium	Escalated	Similar	Similar
T_{14}	Medium	Escalated	Similar	Escalated
T_{15}	Medium	Escalated	Escalated	Similar
T_{16}	Medium	Escalated	Escalated	Escalated
T_{17}	High	Similar	Similar	Similar
T_{18}	High	Similar	Similar	Escalated
T_{19}	High	Similar	Escalated	Similar
T_{20}	High	Similar	Escalated	Escalated
T_{21}	High	Escalated	Similar	Similar
T_{22}	High	Escalated	Similar	Escalated
T_{23}	High	Escalated	Escalated	Similar
T_{24}	High	Escalated	Escalated	Escalated

Regarding the parameters h_{D_i}s and h_{U_i}s, we take the same pragmatic approach as above by introducing two factors, called MTTR relationship (MRR, hereafter) and MTTF relationship (MFR, hereafter), which are applied to characterize the influence of the type of recovery methods and the architecture of the system on the variability of the values of parameters h_{D_i} and h_{U_i}, $i = 1, 2, \ldots, n$, respectively. For each of these two last factors, we shall consider similar and escalated levels, whose meanings are the same as that for the CR factor.

By considering the four factors and their levels, we design the factorial experimental plan shown in Table II, which consists of 24 treatments corresponding to the same number of software system types. For each treatment, we analyze three samples, the numbers of recovery methods of which are 3, 5, and 7. Thus, 72 (24×3) samples are evaluated in total. For each sample, to calculate SSA(s), the following settings of parameters are considered.

1) System-Level Parameters: The values of the following parameters are assigned based on studies in the literature when possible and from reasonable guesses otherwise:

- $h_{DD} = 5$ min, referring to the work in [10];
- $h_{D_H} = 720$ min, i.e., the average human-fix duration is 24 h;
- $h_{U_H} = 43\,200$ min, which corresponds to a rate of one failure per month.

In our numerical analyses, we only consider fixed values for the above three parameters, without any variability, because their magnitudes will not affect the results of the optimal sequence of available recovery methods.

2) Recovery Method Parameters: The three levels of the CR factor correspond to the following ranges for the probability $P(A)$, as introduced in Section IV-B.

- A low level is characteristic of systems for which approximately 5%–8% of the bugs are Mandelbugs. For example, for the Apache AXIS, approximately 7.5% of the bugs are Mandelbugs.
- A medium level is for systems whose proportion of Mandelbugs is in the interval 0.2 and 0.32.
- A high level is for quite complex software systems, such as the Linux OS, for which approximately half of issues are classified as Mandelbugs. The range we assume here is 0.5–0.8.

The values used above are found in the results of the experimental work [10], [33]. The proportion of Mandelbugs directly affects the coverage of recovery methods, and the impact depends on the level of the CR factor as follows.

- *Similar.* We set $c^i = c/n, 1 \leq i \leq n$. Thus, low-, medium-, and high-level treatments are $c = 0.05$, $c = 0.2$, and $c = 0.5$, respectively, and n is the number of recovery methods.
- *Escalated.* We set $c^1 = c/n$, where c and n are the same as the ones in the case above, and we let $c^{i+1} = c^i + \alpha \cdot c$, $1 \leq i < n$, and $\alpha = 0.02$.

Regarding the settings of the *mean time to recovery*, depending on the level of the MRR factor we set the following.

- *Similar.* $h_D^i = h_D, 1 \leq i \leq n$, with h_D equal to 5 min [10].
- *Escalated.* $h_D^1 = h_D$ and $h_D^{i+1} = h_D^i + \delta$, $1 \leq i < n$, where $\delta = 20$.

Regarding the settings of the MTTF, depending on the level of the MFR factor we set the following.

- *Similar.* $h_U^i = h_{U_H}, 1 \leq i \leq n$.
- *Escalated.* $h_U^1 = h_{U_H}$, and $h_U^{i+1} = h_U^i + \beta \cdot h_{U_H}$, $1 \leq i < n$, where $\beta = 1.5$.

B. Evaluation

To support the identification of the optimal sequence of recovery methods, we developed an open-source tool, called optimal environmental diversity-based fault tolerance analysis software (OPENS). OPENS is developed in Python, and it is endowed with a graphical user interface that allows setting the values of parameters related to the system and recovery methods. Then, OPENS finds the sequence s^* of recovery methods that maximizes the steady availability SSA(s) by exhaustively searching and evaluating through (14) all possible permutation sequences. A screenshot of OPENS graphical user interface is shown in Fig. 6. The source code and the documentation of OPENS can be freely downloaded from Github.[4]

Table III shows the optimal sequences determined by OPENS for each treatment listed in Table II. We can observe that for many of the treatments, the optimal sequences can also be obtained in a straightforward way by applying the results provided by Propositions 1–3 introduced in the previous section. To be exact,

[4][Online]. Available: https://github.com/Quentinbuaa/opens.

TABLE III
OPTIMAL SEQUENCE AND THE APPLICABLE PROPOSITION FOR ALL TREATMENTS

Treatment	Optimal Sequence			Applicable Proposition
	n=3	n=5	n=7	
T_1	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3 \rangle$	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3, \mathrm{rr}^4, \mathrm{rr}^5 \rangle$	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3, \mathrm{rr}^4, \mathrm{rr}^5, \mathrm{rr}^6, \mathrm{rr}^7 \rangle$	Proposition 1
T_2	$\langle \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^7, \mathrm{rr}^6, \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	Proposition 2
T_3	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3 \rangle$	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3, \mathrm{rr}^4, \mathrm{rr}^5 \rangle$	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3, \mathrm{rr}^4, \mathrm{rr}^5, \mathrm{rr}^6, \mathrm{rr}^7 \rangle$	Proposition 1
T_4	$\langle \mathrm{rr}^3, \mathrm{rr}^1, \mathrm{rr}^2 \rangle$	$\langle \mathrm{rr}^1, \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^2 \rangle$	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3, \mathrm{rr}^4, \mathrm{rr}^5, \mathrm{rr}^6, \mathrm{rr}^7 \rangle$	–
T_5	$\langle \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^7, \mathrm{rr}^6, \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	Proposition 3
T_6	$\langle \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^7, \mathrm{rr}^6, \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	–
T_7	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3 \rangle$	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3, \mathrm{rr}^4, \mathrm{rr}^5 \rangle$	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3, \mathrm{rr}^4, \mathrm{rr}^5, \mathrm{rr}^6, \mathrm{rr}^7 \rangle$	–
T_8	$\langle \mathrm{rr}^3, \mathrm{rr}^1, \mathrm{rr}^2 \rangle$	$\langle \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^1, \mathrm{rr}^3, \mathrm{rr}^2 \rangle$	$\langle \mathrm{rr}^7, \mathrm{rr}^6, \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^1, \mathrm{rr}^3, \mathrm{rr}^2 \rangle$	–
T_9	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3 \rangle$	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3, \mathrm{rr}^4, \mathrm{rr}^5 \rangle$	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3, \mathrm{rr}^4, \mathrm{rr}^5, \mathrm{rr}^6, \mathrm{rr}^7 \rangle$	Proposition 1
T_{10}	$\langle \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^7, \mathrm{rr}^6, \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	Proposition 2
T_{11}	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3 \rangle$	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3, \mathrm{rr}^4, \mathrm{rr}^5 \rangle$	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3, \mathrm{rr}^4, \mathrm{rr}^5, \mathrm{rr}^6, \mathrm{rr}^7 \rangle$	Proposition 1
T_{12}	$\langle \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^7, \mathrm{rr}^6, \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^1, \mathrm{rr}^2 \rangle$	–
T_{13}	$\langle \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^7, \mathrm{rr}^6, \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	Proposition 3
T_{14}	$\langle \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^7, \mathrm{rr}^6, \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	–
T_{15}	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3 \rangle$	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3, \mathrm{rr}^4, \mathrm{rr}^5 \rangle$	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3, \mathrm{rr}^4, \mathrm{rr}^5, \mathrm{rr}^6, \mathrm{rr}^7 \rangle$	–
T_{16}	$\langle \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^7, \mathrm{rr}^6, \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^1, \mathrm{rr}^2 \rangle$	–
T_{17}	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3 \rangle$	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3, \mathrm{rr}^4, \mathrm{rr}^5 \rangle$	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3, \mathrm{rr}^4, \mathrm{rr}^5, \mathrm{rr}^6, \mathrm{rr}^7 \rangle$	Proposition 1
T_{18}	$\langle \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^7, \mathrm{rr}^6, \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	Proposition 2
T_{19}	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3 \rangle$	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3, \mathrm{rr}^4, \mathrm{rr}^5 \rangle$	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3, \mathrm{rr}^4, \mathrm{rr}^5, \mathrm{rr}^6, \mathrm{rr}^7 \rangle$	Proposition 1
T_{20}	$\langle \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^7, \mathrm{rr}^6, \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^1, \mathrm{rr}^2 \rangle$	–
T_{21}	$\langle \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^7, \mathrm{rr}^6, \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	Proposition 3
T_{22}	$\langle \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^7, \mathrm{rr}^6, \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	–
T_{23}	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3 \rangle$	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3, \mathrm{rr}^4, \mathrm{rr}^5 \rangle$	$\langle \mathrm{rr}^1, \mathrm{rr}^2, \mathrm{rr}^3, \mathrm{rr}^4, \mathrm{rr}^5, \mathrm{rr}^6, \mathrm{rr}^7 \rangle$	–
T_{24}	$\langle \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^2, \mathrm{rr}^1 \rangle$	$\langle \mathrm{rr}^7, \mathrm{rr}^6, \mathrm{rr}^5, \mathrm{rr}^4, \mathrm{rr}^3, \mathrm{rr}^1, \mathrm{rr}^2 \rangle$	–

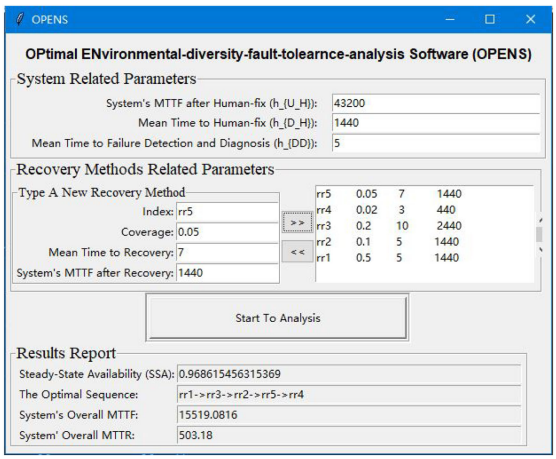

Fig. 6. Screenshot of the interface of OPENS.

Table III pinpoints the applicable propositions for 12 out of 24 treatments, as shown in its last column. According to Table III, it can be checked that the optimal sequences of recovery methods determined by OPENS and by the propositions are identical.

Fig. 7 shows the system maximum SSA for the 24 treatments, for each of the three cases when $n = 3$, $n = 5$, and $n = 7$. The optimal sequences and the corresponding optimal values of $\mathrm{SSA}(s)$ are obtained by running OPENS. In Fig. 7, the results are shown with reference to the baseline system availability (horizontal line in the chart) when no recovery methods apart from the human-fix are applied. In this case, $\mathrm{SSA}(s)$ of the system is given by

$$\mathrm{SSA}(s) = \frac{h_{U_H}}{h_{U_H} + h_{DD} + h_{D_H}}.$$

With the numerical values we assigned to the parameters, $\mathrm{SSA}(s)$ corresponds to 0.967634. In Fig. 7, a bar beneath the horizontal line of the baseline availability indicates that utilizing additional recovery methods can only worsen system availability. As can be observed, this occurs for most treatments for which the MP = Low or MRR = Escalated. This is because the application of recovery methods is ineffective in recovering the majority of failures and, hence, their execution time will add up to system downtime without a significant contribution to the uptime of the system.

C. Validation

By considering the influence of the four factors on system availability, we expect that when comparing the availability of two treatments, the ordering relationships specified by the following properties will hold. For the sake of a more compact notation, we shall denote by MP_i, CR_i, MRR_i, and MFR_i the levels of the four factors of a treatment T_i, and by $\mathrm{SSA}_i(s^*)$ its optimal steady-state system availability.

Property 1: For any pair of treatments T_i and T_j that only differ in the MP level, $\mathrm{MP}_i \geq \mathrm{MP}_j \Rightarrow \mathrm{SSA}_i(s^*) \geq \mathrm{SSA}_j(s^*)$.

Here, $\mathrm{MP}_i \geq \mathrm{MP}_j$ denotes $\sum_{k=1}^{n} c_i^k \geq \sum_{k=1}^{n} c_j^k$, where c_x^k stands for the coverage of the kth available recovery methods of

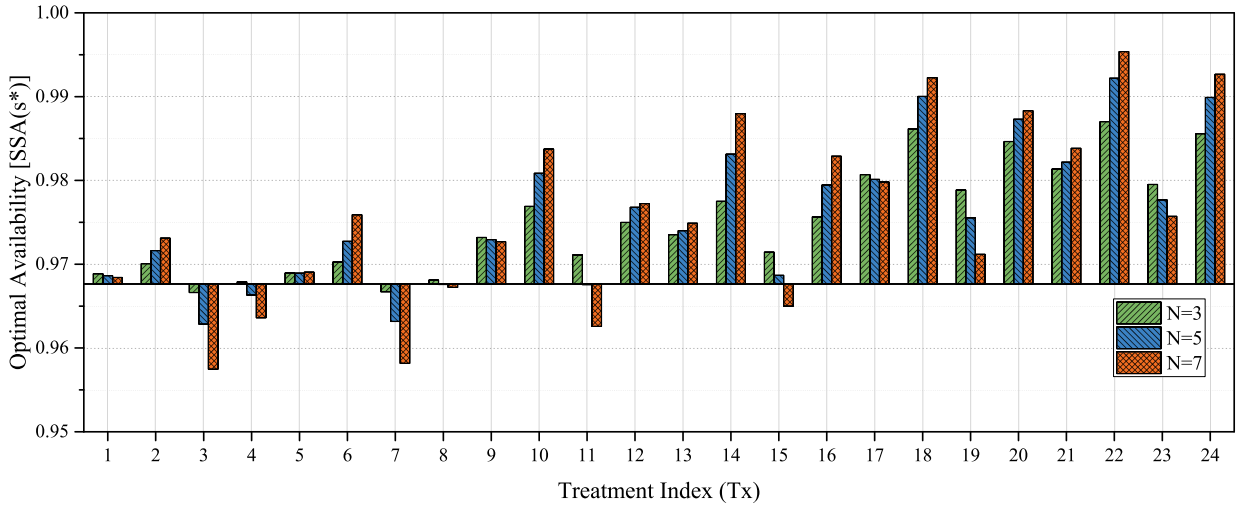

Fig. 7. System's optimal availability [SSA(s^*)]] by 24 treatments when $n = 3$, $n = 5$, and $n = 7$.

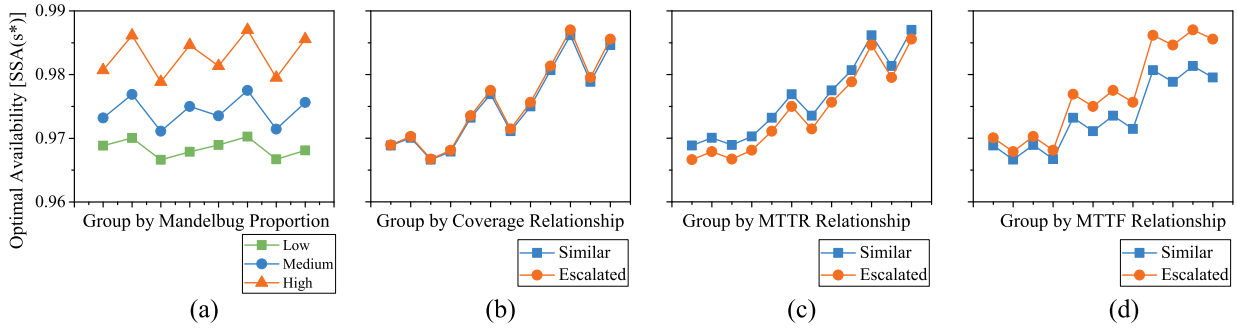

Fig. 8. System's optimal availability[SSA(s^*)] by groups of treatments which have only one different factor setting when $n = 3$.

T_x. This is expected because treatments with a higher MP stands for a larger number of encountered failures can be recovered without applying the human-fix method, therefore shortening the expected overall downtime.

Property 2: For any pair of treatments T_i and T_j that only differ in the CR level, $\text{CR}_i \geq \text{CR}_j \Rightarrow \text{SSA}_i(s^*) \geq \text{SSA}_j(s^*)$.

$\text{CR}_i \geq \text{CR}_j$ denotes $c_i^k \geq c_j^k$, for all $1 \leq k \leq n$. Then, the ordering between SSA of T_i and T_j holds because for any kth recovery method in T_i, it can deal with more types of failures than the corresponding one in T_j.

Property 3: For any pair of treatments T_i and T_j that only differ in the MRR factor, $\text{MRR}_i \geq \text{MRR}_j \Rightarrow \text{SSA}_i(s^*) \leq \text{SSA}_j(s^*)$.

$\text{MRR}_i \geq \text{MRR}_j$ denotes $h_{D_i}^k \geq h_{D_j}^k$, for all $1 \leq k \leq n$, and $h_{D_x}^k$ stands for the mean time to recovery of the kth available recovery method of T_x. Then, the ordering between the SSA of T_i and T_j holds because for any $1 \leq k \leq n$, the expected execution time of the kth recovery method in T_i is smaller than the corresponding one in T_j and, hence, T_i has reduced downtime.

Property 4: For any pair of treatments T_i and T_j that only differ in the MFR factor, $\text{MFR}_i \geq \text{MFR}_j \Rightarrow \text{SSA}_i(s^*) \geq \text{SSA}_j(s^*)$.

$\text{MFR}_i \geq \text{MFR}_j$ denotes $h_{U_i}^k \geq h_{U_j}^k$, for all $1 \leq k \leq n$, and $h_{U_x}^k$ stands for the system MTTF after a failure is successfully mitigated by executing the kth available recovery method of T_x. Then, the ordering between the SSA of T_i and T_j holds because for any $1 \leq k \leq n$, the expected time to failure of the system after the successful application of the kth recovery method in T_i is larger than the one of T_j and, hence, T_i has longer uptime.

These properties are used to validate the system availability analytic approach we proposed. For this purpose, we verify whether the availability optimization results obtained by the model satisfy the four properties. We consider the case when $n = 3$, and we report in Fig. 8(a)–(d), comparing $\text{SSA}(s^*)$ by grouping treatments that have only one different factor setting. From the charts shown in Fig. 8, we can draw the following conclusions.

- Property 1 is satisfied by the results, since according to Fig. 8(a), the treatments with higher MPs have larger $\text{SSA}(s^*)$ than the corresponding treatments with lower MPs.
- Property 2 is satisfied since according to Fig. 8(b), treatments with CR = Escalated have larger values than their corresponding treatments in the CR = Similar level.

- Property 3 is satisfied since according to Fig. 8(c), the treatments with the MRR = Escalated have smaller values than their corresponding treatments in the MRR = Similar level.
- Property 4 is satisfied according to Fig. 8(d), in which the treatments with the MFR = Escalated have larger values than the corresponding treatments in the MFR = Similar level.

The treatments for $n = 5$ and $n = 7$ also satisfy those four properties. Their analysis is not presented for the sake of brevity.

D. Discussion

In this section, we have evaluated the applicability of the results we obtained in Sections IV and V. The four properties we present for the optimal solutions can help understand the influences that various parameters of the system and of the recovery methods have on the system availability. For example, Property 1 indicates that the deployment of the same set of recovery methods will have different impacts on systems with distinct proportions of Mandelbugs. Specifically, the larger the proportion of Mandelbugs the system contains, the higher the expected availability achieved by applying the same set of recovery methods. Properties 2–4 can also be exploited to define algorithmic approaches for guiding and enhancing the applications of environment-diversity-based fault tolerance techniques in practice.

VII. Conclusion

In this article, we developed and studied an analytic model to assess the availability of the systems that utilize a sequence of environmental-diversity-based recovery methods. We made three main contributions to this area: First, we proposed and analyzed a recovery process model to describe the system recovery behaviors for any number of recovery methods, which may have distinct characteristics and be arranged in arbitrary orders. In our modeling approach, we formalized the recovery process as an SMP and, hence, derived the system availability formula. This result was validated by applying it to a special case studied in a published work and by numerical analyses for more general application scenarios. Second, this is the first attempt to explore the problem of determining the optimal sequence for a set of available recovery methods, and we proposed a set of propositions to address this optimization problem. Finally, we developed an open-source tool, namely OPENS, to assist the system availability calculation and the determination of the optimal sequence.

The effectiveness of our formal analysis results and the scope of their application have been evaluated and illustrated through the numerical analyses. The numerical analysis results show that our model can help system architects and developers to decide whether or not to deploy the environmental-diversity-based fault tolerance techniques on their systems. If adopted, the proposed optimal sequencing methods can help them to figure out the best arrangement. In addition, the proposed properties can assist the practitioners in improving the adopted recovery methods.

Future work in this area will be conducted to extend our proposals to include the following two aspects. First, relaxing the time-homogeneity assumption of the transition time among the system states by considering the system's MTTF could vary with the system's operational time. Second, developing a more comprehensive strategy to optimize the sequence of recovery methods to reduce the calculation overhead resulting from the exhaustive search.

Appendix

In this appendix, we compare the system availability formulas derived in this article (14) and the one in [10], which is given by (23) as shown at the bottom of this page, to prove that they provide the same result under equivalent system conditions.

The system studied in [10] can be treated as a special case of the problem considered in this article, and the availability formula for the system can be obtained by setting the variables for (14) as follows:

- $n = 4$, since the system studied in [10] has four recovery methods in a fixed sequence, that is $s = \langle$ restart, reboot, reconfigure, hot-fix $\rangle$.
- $h_{U_i} = h_{U_H}$ $(i = 1, 2, 3, 4)$ because the MTTF of the system is assumed to be independent of the recovery method.
- $p_1 + p_2 + p_3 + p_4 + p_5 = 1$, where p_1, p_2, p_3, p_4, and p_5, respectively, denote the probability that an encountered failure is caused by a Restart-Maskable-Mandelbug, a Reboot-Maskable-Mandelbug, a Reconf-Maskable-Mandelbug, a Hot-fix-Maskable-Mandelbug, and a Bohrbug or a Mandelbug of other types, respectively.

$$
\begin{aligned}
&\text{MTTR} \\
&= p_1 \left[E\left[D_{1ad}\right] + \left(1 - p_{1ad}\right) E\left[D_{1md}\right] + E\left[D_{1dg}\right] + d_{11} E\left[D_{1rs}\right] + d_{12} E\left[D_{1rb}\right] + d_{13} E\left[D_{1rc}\right] + d_{14} E\left[D_{1hf}\right] \right] \\
&\quad + p_2 \left[E\left[D_{2ad}\right] + \left(1 - p_{2ad}\right) E\left[D_{2md}\right] + E\left[D_{2dg}\right] + d_{21} \left(E\left[D_{2rs}\right] + E\left[D_{2rb}\right] \right) + d_{22} E\left[D_{2rb}\right] \right. \\
&\quad \left. + d_{23} E\left[D_{2rc}\right] + d_{24} E\left[D_{2hf}\right] \right] \\
&\quad + p_3 \left[\begin{array}{l} E\left[D_{3ad}\right] + \left(1 - p_{3ad}\right) E\left[D_{3md}\right] + E\left[D_{3dg}\right] + d_{31} \left(E\left[D_{3rs}\right] + E\left[D_{3rb}\right] + E\left[D_{3rc}\right] \right) + d_{32} \left(E\left[D_{3rb}\right] \right. \\ \left. + E\left[D_{3rc}\right] \right) + d_{33} E\left[D_{3rc}\right] + d_{34} E\left[D_{3hf}\right] \end{array} \right] \\
&\quad + p_4 \left[\begin{array}{l} E\left[D_{4ad}\right] + \left(1 - p_{4ad}\right) E\left[D_{4md}\right] + E\left[D_{4dg}\right] + d_{41} \left(E\left[D_{4rs}\right] + E\left[D_{4rb}\right] + E\left[D_{4rc}\right] + E\left[D_{4hf}\right] \right) \\ + d_{42} \left(E\left[D_{4rb}\right] + E\left[D_{4rc}\right] + E\left[D_{4hf}\right] \right) + d_{43} \left(E\left[D_{4rc}\right] + E\left[D_{4hf}\right] \right) + d_{44} E\left[D_{4hf}\right] + \left(1 - p_{4hf}\right) E\left[D_{4bf}\right]. \end{array} \right]
\end{aligned}
\tag{23}
$$

TABLE IV
SYMBOL MAPPING RELATIONSHIP BETWEEN THE TWO STUDIES

Index	Symbols in this paper	Symbols in [10]
1	h_{DD}	$E[D_{ad}] + E[D_{dg}]$
2	h_{D_1}	$E[D_{rs}]$
3	h_{D_2}	$E[D_{rb}]$
4	h_{D_3}	$E[D_{rc}]$
5	h_{D_4}	$E[D_{hf}]$
6	h_{D_H}	$E[D_{bf}]$
7,8,9	p_i $(i = 1, 2, 3)$	p_i $(i = 1, 2, 3)$
10	p_4	$p_4 \cdot p_{4hf}$
11	p_5	$p_4(1 - p_{4hf})$

- c_i $(i = 1, 2, 3, 4)$ can be obtained based on (5) as follows:

$$\begin{aligned} c_1 &= p_1 \\ c_2 &= \frac{p_2}{1 - p_1} \\ c_3 &= \frac{p_3}{1 - p_1 - p_2} \\ c_4 &= \frac{p_4}{1 - p_1 - p_2 - p_3}. \end{aligned} \tag{24}$$

For fairness of comparison and considering the assumptions in this article, the variables in (23) should be configured as follows:

- $E[D_{ad}] = E[D_{1ad}] = E[D_{2ad}] = E[D_{3ad}] = E[D_{4ad}]$;
- $E[D_{md}] = E[D_{1md}] = E[D_{2md}] = E[D_{3md}] = E[D_{4md}]$;
- $E[D_{dg}] = E[D_{1dg}] = E[D_{2dg}] = E[D_{3dg}] = E[D_{4dg}]$;
- $E[D_{rs}] = E[D_{1rs}] = E[D_{2rs}] = E[D_{3rs}] = E[D_{4rs}]$;
- $E[D_{rb}] = E[D_{1rb}] = E[D_{2rb}] = E[D_{3rb}] = E[D_{4rb}]$;
- $E[D_{rc}] = E[D_{1rc}] = E[D_{2rc}] = E[D_{3rc}] = E[D_{4rc}]$;
- $E[D_{hf}] = E[D_{1hf}] = E[D_{2hf}] = E[D_{3hf}] = E[D_{4hf}]$;
- $p_{1ad} = p_{2ad} = p_{3ad} = p_{4ad} = 1$;
- $d_{11} = 1, d_{12} = d_{13} = d_{14} = 0$;
- $d_{21} = 1, d_{22} = d_{23} = d_{24} = 0$;
- $d_{31} = 1, d_{32} = d_{33} = d_{34} = 0$;
- $d_{41} = 1, d_{42} = d_{43} = d_{44} = 0$.

Due to the inconsistency of the symbols used in the two studies, we substitute the symbols used in [10] with the ones defined in this article, and the mapping relationship is shown by Table IV.

For the clarity of description, let SSA_a, SSA_b, MTTF_a, MTTF_b, MTTR_a, and MTTR_b denote the availability, MTTF, and MTTR obtained in this article and [10], respectively.

Before we provide that the MTTF of our system and the one in [10] are the same, we need to prove the following Lemma.

Lemma 1: The following equality holds:

$$\sum_{i=1}^{n} \left(c_i \cdot \prod_{j=1}^{i-1} \overline{c_k} \right) + \prod_{i=1}^{n} \overline{c_i} = 1.$$

Proof: To see that this is true, it suffices to extract from the summation the last term, and observe that

$$c_n \prod_{j=1}^{n-1} \overline{c_k} + \prod_{i=1}^{n} \overline{c_i} = c_n \prod_{j=1}^{n-1} \overline{c_k} + \overline{c_n} \prod_{i=1}^{n-1} \overline{c_i} = \prod_{i=1}^{n-1} \overline{c_i}.$$

This process can be repeated for the $(n-1)$th term of the summation and so on. After $n-1$ steps, the expression reduces to $c_1 + \overline{c_1}$, which is obviously 1. □

We can now prove the following proposition.

Proposition 4: $MTTF_a = MTTF_b$.

Proof: Since $h_{U_H} = h_{U_i}$ and $n = 4$, the denominator of (14) can be rewritten as follows:

$$h_{U_H} \left(\sum_{i=1}^{4} \left(c_i \cdot \prod_{k=1}^{i-1} \overline{c_k} \right) + \prod_{i=1}^{4} \overline{c_i} \right).$$

Owing to Lemma 1, we conclude that $\text{MTTF}_a = h_{U_H}$. Because the system's MTTF is assumed to be constant in [10], we immediately determine that $\text{MTTF}_b = h_{U_H}$. Therefore, we have $\text{MTTF}_a = \text{MTTF}_b$ and, hence, this concludes the proof. □

Next, we prove the following proposition.

Proposition 5: $MTTR_a = MTTR_b$.

Proof: According to (14), since $p_1 + p_2 + p_3 + p_4 + p_5 = 1$, we can obtain

$$\begin{aligned} &\text{MTTR}_a \\ &= h_{DD} + h_{D_1} + h_{D_2}(1 - c_1) + h_{D_3}(1 - c_1)(1 - c_2) \\ &\quad + h_{D_4}(1 - c_1)(1 - c_2)(1 - c_3) + h_{D_H} \textstyle\prod_{i=1}^{4} (1 - c_i) \\ &= h_{DD} + h_{D_1} + h_{D_2}(1 - p_1) \\ &\quad + h_{D_3}(1 - p_1 - p_2) + h_{D_4}(1 - p_1 - p_2 - p_3) \\ &\quad + h_{D_H}(1 - p_1 - p_2 - p_3 - p_4) \\ &= h_{DD} + h_{D_1} + h_{D_2}(p_2 + p_3 + p_4 + p_5) \\ &\quad + h_{D_3}(p_3 + p_4 + p_5) + h_{D_4}(p_4 + p_5) + p_5 h_{D_H}. \end{aligned} \tag{25}$$

After substituting the symbols of (23) with the ones used in this article, we can obtain

$$\begin{aligned} &\text{MTTR}_b \\ &= p_1(h_{DD} + h_{D_1}) + p_2(h_{DD} + h_{D_1} + h_{D_2}) \\ &\quad + p_3(h_{DD} + h_{D_1} + h_{D_2} + h_{D_3}) \\ &\quad + (p_4 + p_5)\left(h_{DD} + h_{D_1} + h_{D_2} + h_{D_3} + h_{D_4} \right. \\ &\qquad \left. + \frac{p_5}{p_4 + p_5} h_{D_H} \right) \\ &= (p_1 + p_2 + p_3 + p_4 + p_5)(h_{DD} + h_{D_1}) \\ &\quad + (p_2 + p_3 + p_4 + p_5) h_{D_2} + (p_3 + p_4 + p_5) h_{D_3} \\ &\quad + (p_4 + p_5) h_{D_4} + p_5 h_{D_H} \\ &= h_{DD} + h_{D_1} + h_{D_2}(p_2 + p_3 + p_4 + p_5) \\ &\quad + h_{D_3}(p_3 + p_4 + p_5) + h_{D_4}(p_4 + p_5) + p_5 h_{D_H}. \end{aligned} \tag{26}$$

According to (25) and (26), we have $\text{MTTR}_a = \text{MTTR}_b$, thus concluding the proof.

Since $\text{SSA} = \{1 + \frac{\text{MTTR}}{\text{MTTF}}\}^{-1}$, with Propositions 4 and 5, we have $\text{SSA}_a = \text{SSA}_b$.

REFERENCES

[1] M. Grottke and K. S. Trivedi, "Fighting bugs: Remove, retry, replicate, and rejuvenate," *IEEE Comput.*, vol. 40, no. 2, pp. 107–109, Feb. 2007.

[2] M. Grottke, A. P. Nikora, and K. S. Trivedi, "An empirical investigation of fault types in space mission system software," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2010, pp. 447–456.

[3] J. Alonso, M. Grottke, A. P. Nikora, and K. S. Trivedi, "An empirical investigation of fault repairs and mitigations in space mission system software," in *Proc. IEEE/IFIP 43rd Annu. Int. Conf. Dependable Syst. Netw.*, 2013, pp. 1–8.

[4] D. Cotroneo, R. Pietrantuono, S. Russo, and K. Trivedi, "How do bugs surface? A comprehensive study on the characteristics of software bugs manifestation," *J. Syst. Softw.*, vol. 113, pp. 27–43, 2016.

[5] D. G. Cavezza, R. Pietrantuono, J. Alonso, S. Russo, and K. S. Trivedi, "Reproducibility of environment-dependent software failures: An experience report," in *Proc. IEEE 25th Int. Symp. Softw. Rel. Eng.*, 2014, pp. 267–276.

[6] K. Qiu, Z. Zheng, K. S. Trivedi, and B. Yin, "Stress testing with influencing factors to accelerate data race software failures," *IEEE Trans. Rel.*, vol. 69, no. 1, pp. 3–21, Mar. 2020.

[7] K. S. Trivedi, M. Grottke, and E. Andrade, "Software fault mitigation and availability assurance techniques," *Int. J. Syst. Assurance Eng. Manage.*, vol. 1, no. 4, pp. 340–350, 2010.

[8] Avizienis and Kelly, "Fault tolerance by design diversity: Concepts and experiments," *IEEE Comput.*, vol. 17, no. 8, pp. 67–80, Aug. 1984.

[9] P. E. Ammann and J. C. Knight, "Data Diversity: An approach to software fault tolerance," *IEEE Trans. Comput.*, vol. 37, no. 4, pp. 418–425, Apr. 1988.

[10] M. Grottke, D. S. Kim, R. Mansharamani, M. Nambiar, R. Natella, and K. S. Trivedi, "Recovery from software failures caused by mandelbugs," *IEEE Trans. Rel.*, vol. 65, no. 1, pp. 70–87, Mar. 2016.

[11] K. Trivedi, D. Wang, D. J. Hunt, A. Rindos, W. E. Smith, and B. Vashaw, "Availability modeling of SIP protocol on IBM© WebSphere," in *Proc. 14th IEEE Pac. Rim Int. Symp. Dependable Comput.*, 2008, pp. 323–330.

[12] K. S. Trivedi, R. Mansharamani, S. K. Dong, M. Grottke, and M. Nambiar, "Recovery from failures due to mandelbugs in IT systems," in *Proc. IEEE Pac. Rim Int. Symp. Dependable Comput.*, 2011, pp. 224–233.

[13] K. S. Trivedi and A. Bobbio, Reliability and Availability Engineering: Modeling, Analysis, and Applications. Cambridge, U.K.: Cambridge Univ. Press, 2017.

[14] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secure Comput.*, vol. 1, no. 1, pp. 11–33, Jan. 2004.

[15] E. Schoenfelder, "Bug #38691 segfault/abort in "UPDATE ...JOIN" while "FLUSH TABLES WITH READ LOCK"," Aug. 2008. [Online]. Available: https://bugs.mysql.com/bug.php?id=38691

[16] T. Maluta, "Bug #11805 mmounting XFS produces a segfault," Aug. 2008. [Online]. Available: https://bugzilla.kernel.org/show_bug.cgi?id=11805

[17] Ericrk, "Issue #776706: Chrome uses a lot of my PC resources since the new update," Feb. 2018. [Online]. Available: https://bugs.chromium.org/p/chromium/issues/detail?id=776706

[18] D. Cotroneo, M. Grottke, R. Natella, R. Pietrantuono, and K. S. Trivedi, "Fault triggers in open-source software: An experience report," in *Proc. IEEE 24th Int. Symp. Softw. Rel. Eng.*, 2013, pp. 178–187.

[19] L. L. Pullum, *Software Fault Tolerance Techniques and Implementation*. Norwood, MA, USA: Artech House, 2001.

[20] T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic Testing: A new approach for generating next test cases," Hong Kong Univ. Sci. Technol., Hong Kong, Tech. Rep. HKUST-CS98-01, 1998.

[21] X. Du, Z. Zheng, G. Xiao, and B. Yin, "The automatic classification of fault trigger based bug report," in *Proc. IEEE Int. Symp. Softw. Rel. Eng. Workshops*, 2017, pp. 259–265.

[22] T. A. Nguyen, D. Min, E. Choi, and T. D. Tran, "Reliability and availability evaluation for cloud data center networks using hierarchical models," *IEEE Access*, vol. 7, pp. 9273–9313, 2019.

[23] M. Torquato and M. Vieira, "Interacting SRN models for availability evaluation of VM migration as rejuvenation on a system under varying workload," in *Proc. IEEE Int. Symp. Softw. Rel. Eng. Workshops*, 2018, pp. 300–307.

[24] A. M. M. Paing, "Analysis of availability model based on software aging in SDN controllers with rejuvenation," in *Proc. IEEE Conf. Comput. Appl.*, 2020, pp. 1–7.

[25] R. Muka, F. B. Haugli, H. Vefsnmo, and P. E. Heegaard, "Information inconsistencies in smart distribution grids under different failure causes modelled by stochastic activity networks," in *Proc. AEIT Int. Annu. Conf.*, 2019, pp. 1–6.

[26] M. R. Lyu, *Softw. Fault Tolerance*. Hoboken, NJ, USA: Wiley, 1995.

[27] J. K. Blitzstein and J. Hwang, *Introduction to Probability*. Boca Raton, FL, USA: CRC Press, 2019.

[28] Z. Chen, X. Chang, Z. Han, and L. Li, "Survivability modeling and analysis of cloud service in distributed data centers," *Comput. J.*, vol. 61, no. 9, pp. 1296–1305, 2018.

[29] X. Chang, T. Wang, R. J. Rodríguez, and Z. Zhang, "Modeling and analysis of high availability techniques in a virtualized system," *Comput. J.*, vol. 61, no. 2, pp. 180–198, 2018.

[30] W. J. Stewart, Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling. Princeton, NJ, USA: Princeton Univ. Press, 2009.

[31] J. B. Dugan and K. S. Trivedi, "Coverage modeling for dependability analysis of fault-tolerant systems," *IEEE Trans. Comput.*, vol. 38, no. 6, pp. 775–787, Jun. 1989.

[32] K. S. Trivedi, Probability and Statistics with Reliability, Queuing and Computer Science Applications, 2nd ed. Hoboken, NJ, USA: Wiley, 2002.

[33] G. Xiao, Z. Zheng, B. Yin, K. S. Trivedi, X. Du, and K.-Y. Cai, "An empirical study of fault triggers in the Linux operating system: An evolutionary perspective," *IEEE Trans. Rel.*, vol. 68, no. 4, pp. 1356–1383, Dec. 2019.

[34] D. C. Montgomery, Design and Analysis of Experiments. Hoboken, NJ, USA: Wiley, 2017.

Kun Qiu received the B.S. degree in automation from the Hefei University of Technology, Hefei, China, in 2013. He is currently working toward the Ph.D. degree in automatic control with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China.

His current research interests include software testing and software reliability analysis.

Zheng Zheng (Senior Member, IEEE) received the Ph.D degree in computer software and theory from the Chinese Academy of Science, Beijing, China, in 2006.

In 2014, he was a Research Scholar with the Department of Electrical and Computer Engineering, Duke University, Durham, NC, USA. He is currently a Professor with of automatic control with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. His current research interests include software dependability modeling, software testing, and software fault localization.

Kishor S. Trivedi (Life Fellow, IEEE) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology Bombay, Mumbai, India, in 1968, and the M.S. and Ph.D. degrees in computer science from the University of Illinois at Urbana-Champaign, Champaign, IL, USA, in 1972 and 1974, respectively.

He is currently the Fitzgerald Hudson Chair with the Department of Electrical and Computer Engineering, Duke University, Durham, NC, USA. His research interests include reliability, availability, performance, and survivability of computer and communication systems and software dependability.

Dr. Trivedi was the recipient of the IEEE Computer Society Technical Achievement Award. He was a Golden Core Member of the IEEE Computer Society.

Ivan Mura (Member, IEEE) received the B.Sc. degree in computer science and the Ph.D. degree in computer science engineering from the University of Pisa, Pisa, Italy, in 1994 and 1999, respectively.

He worked with academia, research organizations, and industry, covering teaching and management positions. He is currently an Associate Professor of Electrical and Computer Engineering with Duke Kunshan University, Suzhou, China. His current research interests include techniques and applications of predictive modeling for artificial and living systems and the integration of measurement data into axiomatic modeling.