Presenting a new fuzzy system for web service selection aimed at dynamic software rejuvenation

Kimia Rezaei Kalantari¹ · Ali Ebrahimnejad² · Homayun Motameni³

Received: 23 February 2020 / Accepted: 7 June 2020 / Published online: 20 June 2020
© The Author(s) 2020

Abstract

As an effective technique to counteract software aging, software rejuvenation is applied in continuously running applications such as web service-based systems. In such systems, web services are allocated depending on the requirements of receivers and the facilities of servers. One of the challenging issues during assignment of web services is how to select the appropriate server to minimize faults. In this paper, we proposed dynamic software rejuvenation in the form of a proactive fault-tolerance technique based on fuzzy system. While including a threshold for the rejuvenation of each web service, we carried out the training phase based on the features of the service providers as well as the receivers' requirements. The results of simulations revealed that our strategy can mitigate the failure rate of web services by 45, 40, 23, and 12% in comparison with the non-fuzzy, regression-based, Markov-based, and ACOGELS-based web service rejuvenation strategies, respectively.

Keywords Software rejuvenation · Fuzzy system · Web services · Fault tolerance · Software aging

Introduction

Given the staggeringly fast growth of internet technology, the main concern revolves around the reliability of web applications. The critical issue in providing reliability involves the performance and availability of web applications/servers. A definition to software aging can be “a growing weakening of the internal state of software”. When software applications run persistently for long periods of time, the corresponding software processes give rise to aging. The processing aging can be associated with shortage of memory and non-release file locks, data corruption in the operating environment, and the like. Aging will undermine the performance of a device and eventually cause the application to fail. As a dynamic fault tolerance technique, software rejuvenation is adopted to solve the software aging problem.

Nonetheless, software rejuvenation barely solves the root cause of software aging. Therefore, software rejuvenation will run cyclically in a predetermined or scheduled time to preserve the firmness of the software system.

The two primary approaches to determining the timing for software rejuvenation are time-based and inspection-based. Time-based approaches specify the optimal rejuvenation time through analyzing the relationship between the software system state and the presumed distribution of system fault. The inspection-based policy is normally pursued through continuous monitoring at runtime and fault behavior using statistical data analysis in an attempt to estimate the intervals of software rejuvenation. Nevertheless, the fairly intricate computing process generally results in greater resource fatigue in the software system. Therefore, frequent inspections at runtime in a software system may lead to certain levels of fault and cost overheads. This should be accounted for along with the fault and cost of failure in an effort to achieve maximum efficiency [1].

As fundamental software artifacts, web services intend to build service-oriented applications [2]. The web service architecture standard, as displayed in Fig. 1, is composed up of three parts, namely Web Service Providers, Web Services Brokers, and Web Service Requester. According to the standard web service architecture, the web service provider is meant to host web services and applications.

✉ Ali Ebrahimnejad
aemarzoun@gmail.com; a.ebrahimnejad@qaemiau.ac.ir

¹ Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran

² Department of Mathematics, Qaemshahr Branch, Islamic Azad University, Qaemshahr, Iran

³ Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran

Fig. 1 Web service architecture [4]

The web service broker normally renders the web service publicly available, while the web service requester contributes by linking the web service operation with the application environment under this architecture [3, 4].

Given the fact that software rejuvenation relies on a wide variety of criteria that are often qualitative and vary under different circumstances, we decided to employ the fuzzy method in this paper. Fuzzy logic-based solutions are cost-effective for a wide range of applications when compared to the traditional methods [5]. The fuzzy logic is used, because the impact level of such criteria need to be determined by several stages of training. Additionally, the system idle time during the rejuvenation routine was minimized by two or more versions instead of one version per service. However, this was achieved by specifying the number of optimal versions per service in a way that (1) the overhead of using multiple versions is minimized; and (2) the system idle time is minimized for each service.

The rest of this paper is organized as follows. “**Literature review**” describes previous relevant studies on rejuvenation. Then, our newly proposed method is explored in “**Proposed method**”. “**Simulation**” describes the experiments and tests. And finally, “**Conclusions**” outlines the results.

The main contributions of this study are summarized as follows: (1) To reduce the failed requested web services, in this study, we use fuzzy allocation that facilitates flexible implementation between the requested and allocated web services in allocation and rejuvenation processes. (2) In our proposed method, services not allocable upon user requests due to low reliability are included in the rejuvenation process while allocating other appropriate services. (3) The suggested method has dynamic management which depends on availability of the whole system. (4) Ignoring any predetermined rejuvenation time in the proposed method can increase the performance of the systems by reducing downtime.

Literature review

One definition to software aging can be “a growing weakening of the internal state of software”. Software rejuvenation is a recent, dynamic technique of fault tolerance adopted to tackle the software aging problem. Older techniques of software fault tolerance tend to be reactive, i.e., they engage in repairing a fault event after its occurrence, which in turn causes a great deal of repair overhead. While software rejuvenation is known as an active technique preventing fault prior to its occurrence or slowing down its occurrence, the cost of such overhead is far lower than that of fault repairing. This technique has been proposed to dealing with the aging process. Vaidyanathan et al. [1] investigated an inspection-based method for preventive maintenance in a software operating system, which was inspected periodically at specific intervals based on the Markov’s model. The optimum inspection interval has been obtained to minimize faults and costs. The performance of this method is justifiable in simple software; however, it requires improvement in more complex software. Ning et al. [6] studied a multi-granularity software rejuvenation policy, assuming that inspection intervals follow an exponential distribution. These studies were limited because of considering constant time and distribution that degrades the system’s performance.

As an essential requirement for web service-based systems, reliability is expected to provide services with high availability, high fault-tolerance, and dynamic deployment capabilities [7, 8]. In this regard, Dantas et al. [9] presented several availability models for private cloud architectures based on Eucalyptus platform. Cloud computing systems basically provide access to large datasets and computing resources. One of the major reasons behind the sluggish growth of cloud computing, however, is security [10–12]. Araujo et al. [13] proposed a software framework mainly used as a service for private cloud computing. This research concentrated on the software effectiveness within the framework of Eucalyptus in terms of system workloads and high demands for remote storage and virtual models. Moreover, they set forth an approach that offers a time-series analysis for rejuvenation scheduling in an attempt to mitigate the fault time by predicting the appropriate time for rejuvenation. Meenakshi et al. [14] suggested another approach that allocated resources with minimal waste in cloud computing by sending requests to the request tuner.

With the emergence of peer-to-peer systems, there has been greater emphasis on the need to identify and utilize diverse groups, multiple features, and distributed and dynamic resources. Bandara and Jayasamana [15] investigated a number of problems concerning peer-to-peer systems. Solutions were investigated in these systems for

collecting and discovering new resources with regard to their architecture, search overheads, and load balancing. They additionally offered software rejuvenation in peer-to-peer systems as an ideal solution to improvement of such features. Guo et al. [16] adopted discrete web services in an effort to measure software aging failures and software rejuvenation strategies based on the assumption that discrete web services provide a loose connection feature to the server. The method in their research used several linear regression methods to calculate the aging of individual web services. Almutawakel et al. [17] presented a new fuzzy technique based on CSP exclusively for cloud resource allocation. This method set forth a new hybrid resource allocation approach based on multi-agent system (MAS), distributed constraints satisfaction problems (DCSP), and the fuzzy logic (FL) model. In [18], the extension of the Fuzzy Possibilistic Clustering Algorithm (FPCM) adopted Type-2 Fuzzy Logic Techniques to enable this method to handle a higher degree of uncertainty in a dataset and eventually solve real-world problems where data clustering is involved.

Mooij et al. [19] demonstrated that semi-automatic software rejuvenation in industrial operations is feasible and cost-effective. They employed specific domain models involving abstract implementation details while applying a practical combination of manual and automated techniques. In [8], software rejuvenation coupled with check-pointing occasionally saved the system state on a reliable storage in a way that the mission task could be resumed from the last-save checkpoint. This work was limited due to the assumption of full-state independent rejuvenation that leads to system degradation. In [20], the optimal service selection was intensely complicated, posing a serious challenge to Smart Cloud Storage management. Fuzzy logic can facilitate the selection of the perfect response to customer requirements in terms of QoS service quality and costs. In addition, it will help to predict the number of software faults over the course of development. In [21], researchers presented a new hybrid aging prediction model, which ideally incorporates the Autoregressive Integrated Moving Average (ARIMA) and Long–Short-Term Memory (LSTM) models to achieve more appropriate fitting of the linear pattern as well as mining the non-linear relationship in the time-series of computing resource usage data for cloud services. Their model came with certain shortages, since it assumed that the linear and non-linear components of the sequence are simply additive. Santoshs et al. [22] proposed a new technique to compare the capabilities of six fault prediction methods for the potential number of faults.

Generally speaking, the majority of previous methods proposing to optimize software rejuvenation in various software systems have disadvantages. In this study, we attempted

to cover most of such disadvantages, a few of which have been summarized below:

1. The methods proposed for optimization of software rejuvenation in software systems are mostly one-dimensional. In fact, only one aspect of software rejuvenation has been considered. For instance, Meng et al. [23] considered their non-sequential inspection intervals to optimize software rejuvenation. They did not, however, take into account other dimensions effective in software rejuvenation. In that study, Machida and Miyoshi [24] considered an optimal stopping problem for software rejuvenation. However, they did not explore sequences at different time intervals.
2. The methods proposed for software rejuvenation mostly involve static management in software systems. In fact, they consider a static structure for software rejuvenation. However, they do not adopt different methods for software rejuvenation at the time of running the software under different circumstances [25–29].
3. There are numerous software rejuvenation methods stopping the software system during software rejuvenation. In fact, user requests are left unanswered at the time of rejuvenation. This policy of implementation mechanism simplifies the procedure, but decreases the system's level of availability [30, 31].
4. There are certain software rejuvenation methods including the software version. In addition, the fault tolerance is low in the systems. When an error occurs at the time of rejuvenation of this single version of the software, the system availability is lowered [16, 32].

The fault tolerance for web services is regarded as a critical research topic in selection of a web service. The conventional fault tolerance web service models pose certain limitations [33, 34]. For example, there are essential sources of fault tolerance not integrating, or not focusing on various attributes of service quality such as performance and accessibility so as to satisfy user requirements.

In their endeavor to resolve these constraints, Nguyen et al. [35] proposed a Bayesian fault tolerance model for web services. It includes several criteria such as reliability, availability, and safety during the evaluation of fault tolerance of web services. Furthermore, their method has substantially eliminated the problems of user preferences and fault tolerance through several service quality parameters. Yahyaoui [36] proposed a game theory model based on fault tolerance, collaborating with web services. In this research, each partner was modeled as a game, where a web service provider in each round bids for collaborating in achieving a specific task. A web service is the winner of the game when it achieves minimum cost based on fault tolerance.

Liu et al. [3] introduced a fault tolerance evaluation model with reasonable correction logic. This method made improvements in the SOA framework by incorporating a trust management module. At the next stage, the trust evaluation model was proposed based on an amendatory subjective logic by viewing trust relationships of service entities in web service networks as a micro-scale network. To conduct internal software rejuvenation, Torquato et al. [37] presented a number of availability models based on Stochastic Petri Nets to evaluate two VM live migration approaches. The method hinged on the redundancy schemes known as Warm-Standby and Cold-Standby. In their review article, Wahab et al. [38] explored the categorization of different types of web services including single, mixed, and associative web services. They further clarified the effective criteria for evaluation of fault tolerance in each web service mentioned earlier. In [39, 40], fault-tolerance techniques were proposed based on meta-heuristics in an effort to fight software aging in continuously running applications such as web service-based systems. However, the methods are limited due to impossible application in systems requiring short response times.

Proposed method

This section presents a new method for solving the shortcomings of previous works mentioned in the last sections.

In web service-based applications, evaluation of Quality of Service values is an important issue and also difficult to calculate due to uncertain, poorly available information about quality parameters of web services, as well as their non-linear relations [41]. In the proposed method, the servers provide web services based on receiver requests under two scenarios. First, they are expected to have minimal reliability requirements for those services. Second, there should be maximum matching between the requested web service and the allocated web service as far as possible. If allocation takes place exactly based on the request in web services (as in previous rejuvenation methods), a major problem in fault-tolerance systems is the curtailed availability for specific infeasible requested services.

In an effort to tackle this deficiency, the new method employed a type of fuzzy allocation that facilitates flexible implementation between the requested and allocated web services. The new application of fuzzy dubbed Fuzzy Dynamic Web Service Rejuvenation (FDWSR) can adjust the membership functions initially defined by experts.

In addition, the fuzzy logic applied in this problem not only enhances the fault tolerance, but also renders the newly proposed method so unique.

Although considering fuzzy logic can improve fault tolerance in these systems, a few other properties made the

method unique. For example, most of the previous rejuvenation methods stop the system at the time of rejuvenation; that is, user requests could not be answered at the time of rejuvenation. This policy of the implementation mechanism simplifies the procedure, but reduces the system's availability. In the proposed method, the web services cannot be allocated to users' requests because of their unsuitable availability.

In our method, services not allocable upon user requests due to low reliability are included in the rejuvenation process while allocating other appropriate services.

On the other hand, foregoing methods have static management; that is, they consider a static structure for software rejuvenation at specific times. The suggested method has dynamic management which depends on availability of the whole system.

Ignoring any predetermined rejuvenation time in the proposed method can increase the performance of the systems by reducing downtime.

In the proposed method, the systems requiring rejuvenation do not receive any request from users, but there are certain alternative systems available to respond to user's request. This, in turn, prevents the system from stopping. In the suggested method for the last shortcoming in previous related works, several web service providers were considered for responding to receivers. In this case, when one server stops, the whole system does not afford halt, thus increasing accessibility at rejuvenation time. In this study, we adopted single web services, for which we set criteria such as run time, threshold of reliability, and maximum deadline on the receiver side which requests the web service. On the server side, we considered criteria such as reliability, failure rate, and rejuvenation rate. In this study, we calculated the mean time to failure (MTTF) for all web services to calculate the inspection periods for rejuvenation. We also set the inspection period to be the lowest MTTF for all web services.

In this research, the servers provide web services based on receivers' request for web services under two conditions. First, they need to have minimal reliability requirements for those services. Second, there should as far as possible be maximum matching between the web service requested and the web service allocated. The reliability of web services in this research was considered exponentially according to Eq. (1), where λ is the rate of failure:

$$R(t) = e^{-\lambda t}. \quad (1)$$

It is probably the most important distribution in reliability work and is used almost exclusively for reliability prediction. This distribution has the advantage of easily estimated parameter (λ).

Fuzzy system structures are adopted to identify a suitable web service for allocating. At the same time, they are suitable web services for rejuvenating, because they consider

(1) 100% of the adoption of web service based on receivers' request that might give rise to substantial restriction and fail many requests; (2) the great response of fuzzy logic to uncertainty and its logical reasoning to uncertain problems like software aging.

Fuzzy logic was originally introduced by Zadeh [42] as a mathematical strategy to represent vagueness in everyday life. In all sorts of intelligent algorithms, such as computational intelligence, artificial intelligence, machine learning, and so forth, one way to compute systems with uncertainty is to use Type-1 or Type-2 Fuzzy Logic Systems [43–46].

The general structure of the proposed algorithm is displayed in Table 1. In this structure, the first line defines the number of stages in a scenario that includes a set of requested web services. At each stage of the run, the appropriate server must be found for each of the requested web services. In the second line, *RWS-Num* denotes the total number of web services requested. For each web service requested at any stage, if the web service fails or expires, it will no longer need to be reviewed (lines 3–5). In line 6, we find the appropriate server for the current web service using a fuzzy system with a structure described later.

If a suitable server is not found for the current web service, the necessary changes should be made to the current web service. In fact, one should be subtracted from the time period, and providing that it does not have any time left, it should fail. If the suitable servers were found in the subsequent levels and the allocation happened, the slight service could not run in the acceptable time (lines 8 and 9). Upon finding the appropriate servers, to run the current web

service, this service will be run with the most appropriate server. In addition, changes will be made on the features of the current web service (lines 10 and 11).

At each stage, after making decisions for all web services, changes should be made to the receivers for the next stages (lines 14–22). If the current condition is the server rejuvenation, the rejuvenation procedure should continue for it. Upon completion of the rejuvenation process, it should be switched to the ready mode (lines 15–17). The threshold value depends on the structure of the web service provider and is based on the value of the server's reliability that provides the desired level of service in this scenario. The threshold in cloud computing for data centers is usually defined by the manufacturers. Different thresholds were considered in the suggested method based on the server's structures. If the reliability of a server is lower than the threshold, the rejuvenation procedure should be performed on it for the next stages. As a result, its mode should be changed to rejuvenation (lines 18–20). At the end of each stage, the last changes should be applied to each service for later stages (line 21).

In the structure of the newly proposed algorithm in Table 1, the most important step is choosing the appropriate servers. In this article, the fuzzy system was used since considering 100% web service compliance with the receivers' request may bring about numerous limitations and may result in many failures in addressing the requests.

In the newly proposed fuzzy system, the reliability of the match rate of user's requested web service and the reliability of web service providers are calculated. In fact, it intends to perform the system conformity to determine whether a

Table 1 General structure of the proposed software rejuvenation algorithm

```

1 For step=1:Max_Ext(RWSA)
2 for i=1: RWS_Num
3 if (RWSA(i).Fail) || (RWSA(i).Ext==0)
4 continue with next RWS
5 end
6 find fuzzy suitable web service(RWSA(i),WSA);
7 if (not find )
8 consider change attributes for RWSA(i)
9 else
10 Run by Best find suitable web service
11 consider change attributes for RWSA(i) and Best find suitable web service
12 end
13 end
14 for i=1:WS_Num
15 if Status of WSA(i) is rejuvenation
16 run rejuvenation for WSA(i)
17 end
18 if Reliability of WSA(i) < threshold
19 perform WSA(i) in rejuvenation Status
20 end
21 consider change attributes for WSA(i) for next step
22 end
23 end

```


service provider can execute a service in a suitable extent. Because the input/output parameters are relative in this system, the proposed method used the Mamdani fuzzy system to find the appropriate web service provider. Based on the proposed fuzzy system, there are supposedly two inputs and one output. The first input of fuzzifier identifies the reliability of the requested web service and consists of three membership functions with labels of “low”, “medium”, and “high”. The thresholds are defined based on the distribution of the parameters in dataset. The second input of fuzzifier identifies the reliability of the web service provider, i.e., the membership functions and numerical intervals are exactly the same as one to truly check the accordance of two input parameters. The output fuzzy variable shows the similarity rate in the fuzzy system. If the value of similarity rate between the web service requester and the web service provider surpasses 70%, service provider could, according to the fuzzy rules, be considered a candidate for allocation. If this value becomes less than 30%, allocation will be impossible. Between these two values, allocation could be done with a numerical ratio. Fuzzy rules were then defined to specify the appropriate logic mapping. We considered five fuzzy rules that determined all possible allocations of the web services and were considered via service provider. For example, the third rule expresses the fact that if the value of the requested web service reliability is medium and the value of web service provider reliability is not low, the allocation output fuzzy variable will be in the nearly certain acceptance status (i.e., allocation will be done).

According to the rules extracted, in case the reliability of the requested web service is greater than the current web service, then allocation will not take place and the less reliable web service will be included in the rejuvenation process.

To obtain the best fuzzy system, several systems were considered with different membership functions and defuzzification methods (according to Table 4). Meanwhile, the best fuzzy system with minimum number of failure and runtime was selected in progress of the work.

Fuzzy rules were then defined to specify the appropriate logic mapping in Table 2. We considered five fuzzy rules that determined all possible allocations of the web services and were considered via service provider. For example, the third rule expresses the fact that if the value of the requested web service reliability is medium and the value of web service provider reliability is not low, the allocation output

fuzzy variable will be in the nearly certain acceptance status (i.e., allocation will be done).

Simulation

In this article, we conducted simulations on Windows 7 and a Core i5 system. Meanwhile, we simulated fault tolerance features for web service providers using the MATLAB programming language. In this simulation, we then made evaluation possible based on scenarios of web service requests that allow the scenarios to be created randomly as well as implementing the scenarios that exist in the previous dataset.

Since one of the most important issues of software rejuvenation is ever-increasing accessibility in web service providers and the reduction of failed web service request numbers, this objective can be attained. Several methods were adopted in this research based on scenarios given in Table 3 and according to the number of failed web services.

In this simulation, the request scenarios of the receivers were tested based on suitable web services selection with/without a fuzzy system. We can see a few of the simulations scenarios in Table 3.

In this simulation, to find the appropriate fuzzy system, 48 fuzzy systems were considered based on different fuzzy membership functions, inferences, and defuzzification according to Table 4.

In the new method, the fuzzy system has seven attributes. The first attribute shows the kind of fuzzy system that considers Mamdani for all systems. As can be seen in the following, several fuzzy systems with different membership functions and

Table 3 Specifications of different scenarios to run in simulation

Data set name	Number of Receivers	Number of different web services	Different types of web services
Data1	100	50	10
Data2	100	50	5
Data3	80	50	10
Data4	80	50	5
Data5	60	40	10
Data6	60	40	5
Data7	50	40	10
Data8	50	40	5

Table 2 Fuzzy rules applied to find the appropriate server

1. If (Reliability RWS is low), then (allocation is accepted)
2. If (reliability RWS is medium) and (reliability WS is low), then (Allocation is rejected)
3. If (reliability RWS is medium) and (reliability WS is not low), then (allocation is accepted)
4. If (reliability RWS is high) and (reliability WS is not high), then (allocation is rejected)
5. If (reliability RWS is high) and (reliability WS is high), then (allocation is accepted)

Table 4 Different considered fuzzy systems in the proposed method

FIS NAME	Fis type	Mf type	And method	Or method	Implication method	Aggregation method	Defuzzification method
FDWSR1	Mamdani	Zmf, pimf, smf	Min	Max	Min	Max	Centroid
FDWSR2	Mamdani	Zmf, pimf, smf	Prod	Max	Prod	Max	Centroid
FDWSR3	Mamdani	Zmf, pimf, smf	Min	Probor	Min	Probor	Centroid
FDWSR4	Mamdani	Zmf, pimf, smf	Prod	Probor	Prod	Probor	Centroid
FDWSR5	Mamdani	Zmf, pimf, smf	Min	Max	Min	Max	Bisector
FDWSR6	Mamdani	Zmf, pimf, smf	Prod	Max	Prod	Max	Bisector
FDWSR7	Mamdani	Zmf, pimf, smf	Min	Probor	Min	Probor	Bisector
FDWSR8	Mamdani	Zmf, pimf, smf	Prod	Probor	Prod	Probor	Bisector
FDWSR9	Mamdani	Trapmf	Min	Max	Min	Max	Centroid
FDWSR10	Mamdani	Trapmf	Prod	Max	Prod	Max	Centroid
FDWSR11	Mamdani	Trapmf	Min	Probor	Min	Probor	Centroid
FDWSR12	Mamdani	Trapmf	Prod	Probor	Prod	Probor	Centroid
FDWSR13	Mamdani	Trapmf	Min	Max	Min	Max	Bisector
FDWSR14	Mamdani	Trapmf	Prod	Max	Prod	Max	Bisector
FDWSR15	Mamdani	Trapmf	Min	Probor	Min	Probor	Bisector
FDWSR16	Mamdani	Trapmf	Prod	Probor	Prod	Probor	Bisector
FDWSR17	Mamdani	Gaussmf	Min	Max	Min	Max	Centroid
FDWSR18	Mamdani	Gaussmf	Prod	Max	Prod	Max	Centroid
FDWSR19	Mamdani	Gaussmf	Min	Probor	Min	Probor	Centroid
FDWSR20	Mamdani	Gaussmf	Prod	Probor	Prod	Probor	Centroid
FDWSR21	Mamdani	Gaussmf	Min	Max	Min	Max	Bisector
FDWSR22	Mamdani	Gaussmf	Prod	Max	Prod	Max	Bisector
FDWSR23	Mamdani	Gaussmf	Min	Probor	Min	Probor	Bisector
FDWSR24	Mamdani	Gaussmf	Prod	Probor	Prod	Probor	Bisector
FDWSR25	Mamdani	Zmf, pimf, smf	Min	Max	Min	Max	Mom
FDWSR26	Mamdani	Zmf, pimf, smf	Prod	Max	Prod	Max	Mom
FDWSR27	Mamdani	Zmf, pimf, smf	Min	Probor	Min	Probor	Mom
FDWSR28	Mamdani	Zmf, pimf, smf	Prod	Probor	Prod	Probor	Mom
FDWSR29	Mamdani	Zmf, pimf, smf	Min	Max	Min	Max	Lom
FDWSR30	Mamdani	Zmf, pimf, smf	Prod	Max	Prod	Max	Lom
FDWSR31	Mamdani	Zmf, pimf, smf	Min	Probor	Min	Probor	Lom
FDWSR32	Mamdani	Zmf, pimf, smf	Prod	Probor	Prod	Probor	Lom
FDWSR33	Mamdani	Trapmf	Min	Max	Min	Max	Mom
FDWSR34	Mamdani	Trapmf	Prod	Max	Prod	Max	Mom
FDWSR35	Mamdani	Trapmf	Min	Probor	Min	Probor	Mom
FDWSR36	Mamdani	Trapmf	Prod	Probor	Prod	Probor	Mom
FDWSR37	Mamdani	Trapmf	Min	Max	Min	Max	Lom
FDWSR38	Mamdani	Trapmf	Prod	Max	Prod	Max	Lom
FDWSR39	Mamdani	Trapmf	Min	Probor	Min	Probor	Lom
FDWSR40	Mamdani	Trapmf	Prod	Probor	Prod	Probor	Lom
FDWSR41	Mamdani	Gaussmf	Min	Max	Min	Max	Mom
FDWSR42	Mamdani	Gaussmf	Prod	Max	Prod	Max	Mom
FDWSR43	Mamdani	Gaussmf	Min	Probor	Min	Probor	Mom
FDWSR44	Mamdani	Gaussmf	Prod	Probor	Prod	Probor	Mom
FDWSR45	Mamdani	Gaussmf	Min	Max	Min	Max	Lom
FDWSR46	Mamdani	Gaussmf	Prod	Max	Prod	Max	Lom
FDWSR47	Mamdani	Gaussmf	Min	Probor	Min	Probor	Lom
FDWSR48	Mamdani	Gaussmf	Prod	Probor	Prod	Probor	Lom

defuzzification method were considered so as to obtain the best fuzzy system:

$$\text{Trapmf}(x, a, b, c, d) = \begin{cases} 0 & x \leq a \\ \frac{x-a}{b-a} & a \leq x \leq b \\ 1 & b \leq x \leq c \\ \frac{d-x}{d-c} & c \leq x \leq d \\ 0 & d \leq x \end{cases} \quad (2)$$

$$\text{smf}(x, a, b) = \begin{cases} 0 & x \leq a \\ 2\left(\frac{x-a}{b-a}\right)^2 & a \leq x \leq \frac{a+b}{2} \\ 1 - 2\left(\frac{x-a}{b-a}\right)^2 & \frac{a+b}{2} \leq x \leq b \\ 1 & x > b \end{cases} \quad (3)$$

$$\text{pimf}(x, a, b, c, d) = \begin{cases} 0 & x \leq a \\ 1 - 2\left(\frac{x-a}{b-a}\right)^2 & \frac{a+b}{2} \leq x \leq b \\ 1 & b \leq x \leq c \\ 1 - 2\left(\frac{x-c}{d-c}\right)^2 & c \leq x \leq \frac{c+d}{2} \\ 2\left(\frac{x-c}{d-c}\right)^2 & \frac{c+d}{2} \leq x \leq d \\ 0 & x > d \end{cases} \quad (4)$$

$$\text{zmf}(x, a, b) = \begin{cases} 1 & x \leq a \\ 1 - 2\left(\frac{x-a}{b-a}\right)^2 & a \leq x \leq \frac{a+b}{2} \\ 2\left(\frac{x-a}{b-a}\right)^2 & \frac{a+b}{2} \leq x \leq b \\ 0 & x > b \end{cases} \quad (5)$$

$$\text{gaussmf}(x, \sigma, c) = e^{-\frac{(x-c_i)^2}{2\sigma_i^2}} \quad (6)$$

$$\min(a, b) = \begin{cases} b, & b < a \\ a, & b \geq a \end{cases} \quad (7)$$

$$\max(a, b) = \begin{cases} a, & b < a \\ b, & b \geq a \end{cases} \quad (8)$$

$$\text{prod}(a, b) = a \times b \quad (9)$$

$$\text{probor}(a, b) = a + b - (a \times b) \quad (10)$$

$$x_{\text{centroid}} = \frac{\int x \mu_i(x) dx}{\int \mu_i(x) dx} \quad (11)$$

$$\int_{\alpha}^{x_{\text{bisector}}} \mu_i(x) dx = \int_{x_{\text{bisector}}}^{\beta} \mu_i(x) dx \quad (12)$$

$$\text{hgt}(Z_k) = \mu_{B^k}(z), \quad z \in Z \quad (13)$$

$$z^* = \sup_{z \in Z} \{z \in Z \mid \mu_{B^k}(z) = \text{hgt}(Z_k)\} \quad (14)$$

$$\text{MOM}[C(z)] = \frac{\int_a^b z dz}{\int_a^b dz} = \frac{1}{2}(a + b). \quad (15)$$

In the next stage, to obtain the best fuzzy system, several systems with different membership functions and defuzzification methods (according to Eqs. 2–15) were considered. Meanwhile, the best fuzzy system with minimum number of failures and runtimes was selected in progress of the work. For a given domain x and a, b, c, d parameters, Eq. (2) returns the corresponding y values for the trapezoidal membership function.

The argument x represented a real number of web service reliability and parameters a, b, c , and d were set by expert depending on system condition and must satisfy the inequalities: $a < b \leq c < d$. The minimum and maximum values of the trapezoid are assumed to be 0 and 1. The parameters $[a \ b \ c \ d]$ correspond to the x values of the trapezoid corners.

Three polynomial-based curve membership functions, given in Eqs. (3–5), are the Z , S , and P_i curves, which are all named because of their shape as zmf , smf , and pimf . The function zmf is the asymmetrical polynomial curve open to the left, smf is the mirror-image function that opens to the right, and pimf is zero on both extremes with rise in the middle. Another fuzzy membership function often used to represent vague, linguistic terms is the Gaussian, which is given by Eq. (6), where c_i and σ_i are the center and width of the i th fuzzy set A^i .

In fuzzy logical operations, any number of well-defined methods can fill in for the AND operation or the OR operation. In the Fuzzy Logic Toolbox, two built-in AND methods (min (minimum) and prod (product)) and two built-in OR methods (max (maximum) and the probabilistic OR method (probor)) are supported. All of these fuzzy inferences are shown in Eq. (7–10). In Eq. (11), x^* is the x -coordinate of center of gravity and $\int_{i=1}^n \mu_i(x) dx$ denotes the area of the region bounded by the curve μ_i . In this equation, x_i is a sample element and n represents the number of elements in the sample. In the bisector defuzzification method, α represents $\min\{x \mid x \in X\}$ and β shows $\max\{x \mid x \in X\}$.

Equation (13) shows the maximum defuzzifier that chooses the point in Z corresponding to the height of

$\mu_{B^k}(z)$. $\text{hgt}(Z_k)$ can be a set of points with max membership function. One of the "smallest/largest/mean" of maximum defuzzifier can be used in maximum defuzzifier as the largest of maximum values, as shown in Eq. (14). In mean of maximum method (Eq. 15), the defuzzified value is taken as the element with the highest membership values.

Since the fuzzy systems considered in this study are dedicated to dynamic rejuvenation of web services, we named them FDWSR. In the tests conducted for evaluation, we first considered different fuzzy systems in the proposed method, and then simulated this method for a non-fuzzy system. We also compared our method with the rejuvenation of web services based on multiple linear regression approach which was proposed in [47], combination of Ant Colony Optimization and Gravitational Emulation Local Search based approach in [40], and the two-granularity inspection-based software rejuvenation policy based on the Markov regenerative process in [6]. In all the methods, the comparison criteria were runtime and the number of failed web services.

Table 5 provides the number of failed web services, for the mentioned scenarios (in Table 3) using different fuzzy methods. As can be seen, the fuzzy systems 2, 4–8, 13–16, 22, 24, and 43–44 have lower failed web services.

The runtimes of scenarios using different fuzzy methods are presented in Table 6. As can be seen, the fuzzy system 13 has the shorter runtime in comparison to the other fuzzy systems with the same number of failures (on average 5.38 s).

According to the obtained results in Tables 5 and 6, the fuzzy system 13 is the most appropriate fuzzy system owing to its minimum number of failures and runtimes. Now, we compare the results of the proposed method based on FDWSR13 against other methods. In Fig. 2, the performance of the proposed method has been illustrated. As Fig. 2 displays, the suggested method has the most flexibility and the least failed web service numbers in different scenarios. For instance, in dataset 8, the fuzzy-based method has ten failed web services, i.e., three, five, five, and two numbers approximately less than non-fuzzy, multiple linear regression approach, inspection-based policy based on the Markov regenerative method and ACOGELS approach, respectively. As Fig. 2 demonstrates, the fuzzy method, in comparison with non-fuzzy, regression-based, inspection-based methods, and ACOGELS approach, has on average less than 45%, 40%, 23%, and 12% failed web service, respectively. As results demonstrate, the proposed fuzzy rejuvenation method in comparison with state-of-the-art strategies has lower failed web service rate. Through an evaluation, it was shown that our new fault tolerance scheme brings promising improvement on the availability of system. The reason for this improvement versus non-fuzzy approach is the restriction of considering 100% of combination between web service receivers' request and web service providers' reliability in non-fuzzy method. The basic disadvantage of

regression-based against our method is the reliance on stopping the software system at the time of software rejuvenation in comparison with the proposed method. A large number of unstructured parameters and a constant time inspection trigger considered in Markov regenerative method served to confine it more than our newly proposed method. Despite its efficiency, the ACOGELS-based approach is not as good as the proposed method for dealing with large-scale combinatorial problems because of its time complexity.

Despite its relatively desirable performance, this algorithm is barely more appropriate than the newly proposed method due to its time complexity.

Several users request web services with different quality of service parameters. For example, a user asks about a type of service with minimum 70% of availability. According to the proposed fuzzy allocation algorithm aiming to reduce the failed requested web services, the best services that meet the request is found. Some services with poor quality that could not allocate to any requests are found and simultaneously enter the rejuvenation process. In rejuvenation process, the quality parameters of unassigned web services are improved and will be ready to allocate in subsequent requests. In other words, the allocated services that running continuously for a certain time get older and becomes less immune. For this purpose, the proposed algorithm dynamically checked the quality of web services and then it decides which services are suitable for allocating and which of other services should enter into the rejuvenation process.

In the following, the proposed method has been compared against the existing algorithms in terms of runtime as mentioned in Fig. 3. Runtime refers to the amount of time taken for a computer program to perform a task. According to Fig. 3, the runtime of our proposed method has, on average, 21% more than Markov-based, Regression-based, and Non-Fuzzy methods, and 42% less than ACOGELS-based approach.

Conclusions

Software rejuvenation is helpful in preventing the failures in the running process of the software. The rejuvenation policies are powerful tools for improvement of the performance in web service-based systems. In this paper, the effective use of fuzzy system for dynamic software rejuvenation was investigated based on the characteristics of service providers and the requirements of service requestors. In addition, the allocation of web services and the threshold for rejuvenation were checked dynamically. Hence, the systems requiring rejuvenation were detected.

We detected systems that required rejuvenation prior to failure and then rejuvenated them. In this study, fuzzy system structures were adopted to identify a suitable web

Table 5 Failed web services using different fuzzy system

System	Dataset							
	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Data8
FDWSR1	4	4	1	0	7	0	6	10
FDWSR2	4	4	1	0	7	0	5	10
FDWSR3	4	4	1	0	7	0	6	10
FDWSR4	4	4	1	0	7	0	5	10
FDWSR5	4	4	1	0	7	0	5	10
FDWSR6	4	4	1	0	7	0	5	10
FDWSR7	4	4	1	0	7	0	5	10
FDWSR8	4	4	1	0	7	0	5	10
FDWSR9	5	4	1	2	7	0	6	10
FDWSR10	4	4	1	2	7	0	6	10
FDWSR11	5	4	1	2	7	0	6	10
FDWSR12	4	4	1	2	7	0	6	10
FDWSR13	4	4	1	0	7	0	5	10
FDWSR14	4	4	1	0	7	0	5	10
FDWSR15	4	4	1	0	7	0	5	10
FDWSR16	4	4	1	0	7	0	5	10
FDWSR17	9	8	4	2	7	2	7	15
FDWSR18	5	4	1	2	7	0	6	12
FDWSR19	8	8	4	2	7	1	6	15
FDWSR20	5	4	1	2	7	0	6	11
FDWSR21	5	5	3	0	7	0	5	13
FDWSR22	4	4	1	0	7	0	5	10
FDWSR23	5	4	1	0	7	0	5	11
FDWSR24	4	4	1	0	7	0	5	10
FDWSR25	5	4	1	0	7	0	5	10
FDWSR26	5	4	1	0	7	1	5	10
FDWSR27	4	5	1	0	7	0	5	11
FDWSR28	4	4	2	1	7	0	5	10
FDWSR29	5	5	1	0	7	0	6	10
FDWSR30	4	5	1	0	7	0	5	10
FDWSR31	5	4	2	0	7	0	5	10
FDWSR32	5	4	1	1	7	0	5	10
FDWSR33	5	4	2	1	7	0	5	10
FDWSR34	5	4	1	0	7	1	5	11
FDWSR35	4	4	2	0	7	1	6	10
FDWSR36	4	5	1	0	7	1	5	10
FDWSR37	4	3	2	0	7	1	5	10
FDWSR38	6	4	1	0	7	1	6	10
FDWSR39	5	4	1	0	7	1	6	10
FDWSR40	4	5	1	0	7	0	5	10
FDWSR41	4	4	2	0	7	0	5	11
FDWSR42	4	5	1	0	7	0	5	10
FDWSR43	4	4	1	0	7	0	5	10
FDWSR44	4	4	1	0	7	0	5	10
FDWSR45	5	3	2	0	7	0	5	10
FDWSR46	5	3	1	0	7	0	5	10
FDWSR47	5	3	2	0	7	0	5	10
FDWSR48	5	4	1	0	7	0	6	10

Table 6 Runtime of scenarios with different fuzzy methods

System	Dataset								
	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Data8	Means result
FDWSR1	6.3917	4.7183	6.5103	6.8391	6.4331	5.991	3.3503	4.7018	5.61695
FDWSR2	7.3278	6.5776	5.88	9.0002	8.1373	5.8714	4.2936	2.9598	6.255963
FDWSR3	5.1415	5.509	4.7908	7.4903	7.2747	7.3157	4.1231	4.5994	5.780563
FDWSR4	6.628	4.2042	5.8363	6.8407	6.2158	8.8107	3.5807	3.941	5.757175
FDWSR5	4.5405	5.46	5.3876	8.1886	7.1908	8.0524	3.5269	4.619	5.870725
FDWSR6	6.4689	6.2601	5.8437	6.3275	7.9363	7.9874	4.176	3.9233	6.1154
FDWSR7	6.5523	6.4945	6.6909	8.7033	6.0049	7.6949	3.9016	4.2032	6.2807
FDWSR8	4.9819	5.6002	6.8286	9.2791	7.6665	8.0979	3.4142	3.9331	6.225188
FDWSR9	6.398	7.9301	6.5208	8.1735	6.6153	7.8288	2.2572	3.8052	6.191113
FDWSR10	4.4675	5.6247	6.8858	8.323	9.383	10.4399	4.279	2.8972	6.537513
FDWSR11	6.2554	5.6089	5.9471	8.312	7.6353	7.1208	2.922	4.6366	6.054763
FDWSR12	6.1141	6.0133	6.5762	7.8616	5.4703	8.4309	4.0198	4.3057	6.098988
FDWSR13	4.3981	6.7382	4.5417	7.3163	5.8238	7.2614	3.3839	3.5913	5.381838
FDWSR14	6.1334	5.8362	8.7523	8.1763	8.544	7.1163	2.3812	3.733	6.334088
FDWSR15	6.0914	5.4788	6.5653	8.2999	6.5993	7.224	4.3259	2.5879	5.896563
FDWSR16	4.7956	5.5031	5.7178	7.2066	6.2609	7.666	3.6993	4.0985	5.618475
FDWSR17	6.8671	5.9956	5.0536	9.0148	7.7665	7.7266	2.9821	2.8189	6.02815
FDWSR18	4.5699	4.1237	5.6473	8.8657	5.5869	8.4377	3.996	4.3724	5.69995
FDWSR19	6.8592	6.3905	7.3104	7.453	7.3375	7.9312	3.5358	3.6904	6.3135
FDWSR20	4.6126	5.7728	4.8542	8.6453	7.4182	7.4887	4.5791	4.3881	5.969875
FDWSR21	6.4381	5.9045	6.9361	6.1576	7.9407	8.4849	3.8135	4.7897	6.308138
FDWSR22	4.6897	6.3487	6.5971	8.3451	7.4013	6.0823	3.7635	4.0434	5.908888
FDWSR23	5.1786	4.2385	4.6713	8.6717	7.4768	6.3785	4.1235	4.6719	5.67635
FDWSR24	7.0017	4.4631	6.9019	8.3528	7.6624	6.3132	3.7031	4.1929	6.073888
FDWSR25	9.3649	7.9717	9.4381	12.7884	10.5488	12.7421	4.2593	4.5932	8.963313
FDWSR26	8.7824	7.7106	7.5480	10.5714	9.2853	9.7500	3.9789	3.9947	7.702663
FDWSR27	7.5861	7.0922	7.6193	10.0611	9.0825	9.8274	4.0202	4.5183	7.475888
FDWSR28	7.5552	6.8289	7.4735	7.5552	9.2609	9.6525	3.9700	4.5155	7.101463
FDWSR29	7.5529	6.9417	7.9034	8.2167	9.0076	10.1018	4.0989	4.8518	7.33435
FDWSR30	7.8413	6.7854	6.9020	8.2129	9.0594	9.6201	4.1003	4.6812	7.150325
FDWSR31	7.6190	5.9917	7.1801	10.0112	9.5577	10.4718	3.9910	4.8122	7.454338
FDWSR32	7.7623	6.5445	7.7123	10.0015	9.2743	9.9567	3.9878	4.3212	7.445075
FDWSR33	7.4745	6.6518	7.3496	10.1315	9.2768	10.0012	3.9913	4.7169	7.4492
FDWSR34	7.6602	6.6371	7.4969	10.2276	9.1502	9.5708	3.9155	4.4148	7.384138
FDWSR35	7.8738	6.7311	7.4512	10.0257	9.4799	8.9797	9.5309	4.4460	8.064788
FDWSR36	7.5513	6.6313	7.4647	10.0640	9.0018	9.7564	3.8958	4.4640	7.353663
FDWSR37	7.5502	6.7761	7.5314	10.0830	8.9949	9.6475	4.0112	4.4808	7.384388
FDWSR38	7.5505	6.6842	7.8120	10.0965	9.0689	9.5797	3.9603	4.4281	7.397525
FDWSR39	7.8648	6.7015	7.4809	10.0950	9.0686	9.5710	3.9268	4.4166	7.39065
FDWSR40	7.5520	6.7385	7.5343	10.2016	8.9607	9.5916	3.9179	4.4300	7.365825
FDWSR41	7.6899	6.6854	7.6591	10.1056	9.0946	9.5793	3.9260	4.4684	7.401038
FDWSR42	7.6868	6.8165	7.4300	10.0415	8.9732	9.4828	3.8745	4.4224	7.340963
FDWSR43	7.4580	6.4986	7.3967	9.9868	8.9471	9.7467	3.9595	4.4264	7.302475
FDWSR44	7.4779	6.5661	7.5743	10.0619	9.1110	9.5174	4.3124	4.4838	7.3881
FDWSR45	7.7865	6.7291	7.5468	10.1935	9.1174	9.6208	3.9405	4.4550	7.4237
FDWSR46	7.4772	6.6945	7.5057	10.0795	9.0956	9.5759	3.9345	4.4441	7.350875
FDWSR47	7.5346	6.6366	7.5352	10.0324	9.4218	9.6149	3.9470	4.4639	7.3983
FDWSR48	7.5749	6.6592	7.5640	10.1182	9.1178	9.6249	3.9335	4.5022	7.386838

Fig. 2 Number of failed web services in fuzzy, non-fuzzy, regression-based and inspection-based methods

Fig. 3 Performance of the proposed method in terms of time against existing methods

services for allocating and at the same time finding suitable web services for rejuvenation.

The results obtained from the proposed method have on average less than 30% of failed web services in comparison with the well-known previous methods. Furthermore, the runtime is, on average, 21% longer than Markov-based,

Regression-based, and Non-Fuzzy methods, and 42% shorter than ACOGELS-based approach.

Contrary to such increase in runtime, the 36% reduction in the number of failed web services on average compared to Non-Fuzzy, Regression-based, and Markov-based approach shows the high fault tolerance of the newly proposed method.

It should be noted that the runtime of the proposed method is slightly high for online environments and requires more hardware overhead. For future work, to overcome this limitation, parallel algorithms can be used. Moreover, other criteria such as reliability and validity can be considered to improve the accuracy of the proposed method and choose the right service providers of web services. Moreover, this method should be considered along with clustering methods for real-world environments where the number of web services and requesters is far higher. We look forward to enhancing the resources allocation process using the other methods of fuzzy multi-criteria decision-making (MCDM) such as fuzzy TOPSIS and fuzzy AHP to optimize resource allocation systems using QoS constraints with multiple resource criteria. Moreover, the proposed mechanism can be enhanced through different thresholding schemes and task priority-based migration scheduling. Additionally, a combination of neural network and fuzzy method is suggested.

Acknowledgements The authors would like to thank the anonymous reviewers and the associate editor for their insightful comments and suggestions.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Mongiello M, Colucci S, Vogli E, Grieco L, Sciancalepore M (2016) Run-time architectural modeling for future internet applications. *Complex Intell Syst* 2:111–124
2. Hani U, Ibrahim ZH (2016) Review on web service architectures and service oriented architecture modeling techniques for monitoring scholarly publication performance. *Inform Syst Res Innov* 10:8–14
3. Liu FM, Wang L, Gao L, Li H, Zhao H, Men SM (2014) A Web Service trust evaluation model based on small-world networks. *Knowl-Based Syst* 57:161–167
4. Grottke M, Nikora A, Trivedi K (2010) An empirical investigation of fault types in space mission system software. *IEEE Conference on Dependable systems and networks*, pp 447–456
5. Zarandi MH, Sadat AA, Sotudian Sh, Castillo O (2020) A state of the art review of intelligent scheduling. *Artif Intell Rev* 53:501–593
6. Ning G, Trivedi K.S, Hu H, Cai K.Y (2011) Multi-granularity software rejuvenation policy based on continuous time Markov chain. In: *IEEE International Workshop on Software Aging and Rejuvenation*, pp 32–37
7. Adeli M, Mazinan A.H (2019) An empirical investigation of fault types in space mission system software. In: *IEEE Conf. on Dependable systems and network. Complex and Intelligent Systems*.
8. Levitin G, Xing L, Huang H (2019) Optimization of partial software rejuvenation policy. *Reliab Eng Syst Saf* 182:63–72
9. Dantas J, Matos R, Araujo J, Maciel P (2015) Eucalyptus-based private clouds: availability modeling and comparison to the cost of a public cloud. *Computing* 97(11):1121–1140
10. Alharbi F, Atkins A, Stanier C (2016) Understanding the determinants of cloud computing adoption in Saudi healthcare organisations. *Complex Intell Syst* 2:155–171
11. Xiuguo W (2018) A security-aware data replica placement strategy based on fuzzy evaluation in cloud. *J Intell Fuzzy Syst* 35(1):243–255
12. Zhang P, Zhou M (2018) Dynamic cloud task scheduling based on a two-stage strategy. *Autom Sci Eng* 15:772–783
13. Araujo J, Matos R, Alves V, Maciel P, Souza F, Trivedi KS (2014) Software aging in the eucalyptus cloud computing infrastructure: characterization and rejuvenation. *ACM J Emerg Technol Comput Syst* 10(1):1–22
14. Meenakshi A, Sirmathi H, Anitha J (2019) Cloud computing based resource provisioning using K-mean clustering and GWO prioritization. *Soft Comput* 23:10781–10791
15. Bandara H, Jayasumana AP (2013) Collaborative applications over peer-to-peer systems-challenges and solutions. *Peer-to-Peer Netw Appl* 6:257–276
16. Guo J, Song X, Wang Y, Zhang B, Li X (2012) The measurement of software aging damage and rejuvenation strategy for discrete web services. *Adv Mater Res* 2:432–437
17. Almutawakel A, Kazar O, Bali M, Belouaar H, Barkat A (2019) Smart and fuzzy approach based on CSP for cloud resources allocation. *Int J Comput Appl*. <https://doi.org/10.1080/1206212X.2019.1701241>
18. Rubio E, Castillo O, Valdez F, Melin P, Gonzalez CI, Martinez G (2017) An extension of the fuzzy possibilistic clustering algorithm using type-2 fuzzy logic techniques. *Adv Fuzzy Syst* 154:1–8
19. Mooij A, Eggen G, Hooman J, Wezep H (2015) Cost-effective industrial software rejuvenation using domain-specific models. *Conf Theory Pract Model Transform* 9152:66–81
20. Santoshs R, Sandeep K (2017) An empirical study of some software fault prediction techniques for the number of fault prediction. *Soft Comput* 21(24):7417–7434
21. Levitin G, Xing L, Xiang Y (2020) Optimizing software rejuvenation policy for tasks with periodic inspections and time limitation. *Reliab Eng Syst Saf* 197:1–31
22. Esposito C, Ficco M, Palmieri F, Castiglione A (2016) Smart cloud storage service selection based on fuzzy logic. *IEEE Trans Theory Evid Game Theory* 65(8):2348–2362
23. Meng H, Liu J, Hei X (2015) Modeling and optimizing periodically inspected software rejuvenation policy based on geometric sequences. *Reliab Eng Syst Saf* 133:184–191
24. Machida F, Miyoshi N (2015) An optimal stopping problem for software rejuvenation in a job processing system. *Int Workshop Softw Reliab Eng* 3:139–143
25. Okamura H, Dohi T (2013) Dynamic software rejuvenation policies in a transaction-based system under Markovian arrival processes. *Perform Eval* 70:197–211
26. Ning G, Zhao J, Lou Y, Alonso J, Matias R, Trivedi KS, Yin BB, Cai KY (2016) Optimization of two-granularity software rejuvenation policy based on the markov regenerative process. *IEEE Trans Reliab* 65(4):1630–1646

27. Kulkarni P (2015) Software rejuvenation and workload distribution in virtualized system. *Int J Innov Res Comput Commun Eng* 3(6):5966–5973
28. Umesh IM, Srinivasan GN (2016) Optimum software aging prediction and rejuvenation model for virtualized environment. *Indones J Electr Eng Comput Sci* 3(3):572–578
29. Silva L, Madeira H, Gabriel J (2006) Software Aging and Rejuvenation in a SOAP-based Server. In: *Proceedings of 5th IEEE Int. Symposium on Network Computing and Applications*, pp 81–90
30. Dohi D, Okamura H, Trivedi K S (2012) Optimizing software rejuvenation policies under interval reliability criteria. In: *Proceeding of 9th Conference on Ubiquitous Intelligence and Computing*; Fukuoka; Japan, pp 478–485
31. Cotroneo D, Iannillo AK, Natella R, Pietrantuono R, Russo S (2015) The software aging and rejuvenation repository. *IEEE Trans Softw Reliab Eng* 2:108–113
32. Agepati R, Gundala N, Amari S (2013) Optimal software rejuvenation policies. In: *2013 Proceedings Annual Reliability and Maintainability Symposium*, pp 341–347
33. Xia Y, Dafang Zh, Jinmin Y (2016) An adaptive fault tolerance strategy for cloud storage. *KSII Trans Internet Inform Syst* 10(11):5290–5304
34. Baek S, Lee J, Lee B (2018) Improving fault traceability of web application by utilizing software revision information and behavior Model. *KSII Trans Internet Inform Syst* 12(2):817–828
35. Nguyen H.T, Zhao W, Yang J (2010) A trust and reputation model based on bayesian network for web services. In: *Proceeding of 8th Conference on Web Services*; Miami; USA, pp 251–258
36. Yahyaoui H (2012) A trust-based game theoretical model for web services collaboration. *Knowl-Based Syst* 27:162–169
37. Torquato M, Umesh IM, Maciel P (2018) Model for availability and power consumption evaluation of a private cloud with VMM rejuvenation enabled by VM live migration. *Supercomputing* 74:4817–4841
38. Wahab OA, Bentahar J, Otrok H, Mourad A (2015) A survey on trust and reputation models for web services. *Decis Support Syst* 74:121–134
39. Rezaei Kalantari K, Ebrahimnejad A, Motameni M (2020) Dynamic software rejuvenation in web services: a whale optimization algorithm-based approach. *Turk J Electr Eng Comput Sci* 28:890–903
40. Rezaei Kalantari K, Ebrahimnejad A, Motameni M (2020) An efficient improved ant colony optimization algorithm for dynamic software rejuvenation in web services. *IET Softw*. <https://doi.org/10.1049/iet-sen.2019.0018>
41. Ghayathri J, Pannirselvam S (2016) Categorization of web services based on QOS constraint using decision tree classifier. *Innov Technol Creat Eng* 6(3):338–344
42. Zadeh LA (1965) Fuzzy sets. *Inform Control* 8:338–352
43. Moreno JE, Sanchez MA, Mendoza O, Rodríguez-Díaz A, Castillo O, Melin P, Castro JR (2020) Design of an interval type-2 Fuzzy model with justifiable uncertainty. *Inform Sci* 513:206–221
44. Castillo O, Amador-Angulo L, Castro JR, Garcia-Valdez M (2016) A comparative study of type-1 fuzzy logic systems, interval type-2 fuzzy logic systems and generalized type-2 fuzzy logic systems in control problems. *Inform Sci* 354:257–274
45. Castillo O, Cervantes L, Soria J, Sanchez M, Castro JR (2016) A generalized type-2 fuzzy granular approach with applications to aerospace. *Inform Sci* 354:165–177
46. Zarandi MH, Soltanzadeh S, Mohammadi A, Castillo O (2019) Designing a general type-2 fuzzy expert system for diagnosis of depression. *Appl Soft Comput* 80:329–341
47. Jia SH, Hou CH, Wang J (2017) Software aging analysis and prediction in a web server based on multiple linear regression algorithm. In: *9th IEEE Conference on Communication Software and Networks*, pp 1452–1456

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.