Journal of mathematics and computer Science 7 (2013) 160 - 170

Contents list available at JMCS

Journal of Mathematics and Computer Science

Journal Homepage: www.tjmcs.com

# Measuring the Failure Rate in Service-Oriented Architecture Using Fuzzy Logic

Ali Yavari
Mazandaran University of Sciences and Technology, Mazandaran, Iran
*yavari@ustmb.ac.ir*

Maryam Musavi
Mazandaran University of Sciences and Technology, Mazandaran, Iran
*maryam.musavi@ustmb.ac.ir*

Hossein Momeni
Gorgan University of Agricultural Sciences and Natural Resources, Gorgan, Iran
*momeni@gau.ac.ir*

Mahnaz Hamzehnia
Mazandaran University of Sciences and Technology, Mazandaran, Iran
*mahnaz.hamzehnia@ustmb.ac.ir*

**Article history:**
**Received March 2013**
**Accepted Apri 2013**
**Available online May 2013**

## *Abstract*

Service-Oriented architecture is a developing method to creating distributed programs for flexible and dynamic architecture and is highly suitable for expanding the distributing systems. The use of this architecture has increased the design of software systems. Reliability is the main challenge of this architecture. The fault tolerance mechanism is one of the existing mechanisms for creating reliable services and failure rate measurement of the system is a challenge for fault tolerance. In this paper we propose a method named Service-Oriented SysFailRate Measurement for measuring the failure rate of the system in service-oriented architecture using fuzzy logic. In this system we used incomplete description of a service, lack of integration, incorrect format, and being out of the pre-determined time out factors to measuring the failure rate. Testing rests demonstrated that this approach is promising to significantly improve measuring the failure rate in Service-Oriented architecture.

**Keywords:** Service-Oriented Architecture, Quality of Service, Fault Tolerance, Fuzzy logic.

## 1. Introduction

Service-oriented architecture is regarded as the most attractive approach to expanding the software system which is used for creating very complex practical programs. Using this approach, practical programs based on service-oriented architecture are made according to the service but are not coded. Within the service-oriented architecture, a service is actually a function or a set of functions or actions which a practical program conducts. In older systems which were not based on services, these functions were automatically transformed into systems[1].

Today, service-oriented architecture is widely used as a flexible architecture for expanding dynamic systems [2, 3].This system has feeble connections and dynamic configuration for expanding different distributing systems[4]. The services are self-descriptive in this architecture [2, 5, 6]. Many of the descriptions are kept in component called "service discovery" or "service registry". The service applicant finds a service in "service register" based on its needs and then connects to the service for execution. The structure of this architecture is divided into three components: (1) service provider which is responsible for creating and designing the services, (2) service applicant who uses the provided services, and (3) service register in which the service descriptions are classified. Taking into account the increasing need for high-quality services in service-oriented architecture, it's better to consider the dimensions of service quality in this architecture such as security, availability, and reliability. These dimensions are among the most significant features for developing the fundamental reliable systems [2, 5, 7]. Quality of service (QoS) means all components which mix to define quality of service suggested by one service. Quality of service is essentially used in network's system and is a concept which determines the quality of internet services [8]. In fact, reliability is the ability of a system for preventing failure [9, 10]. The core of service-oriented architecture uses three components: publishing, discovery, and binding. Accordingly, the occurred faults in this architecture divide into fault occurred in publishing stage, discovery, and binding stage. These faults increase the rate of failure for the system[1]. Fault tolerance mechanism is one of the existing mechanisms for creating reliable services[4]. . In this paper we propose a method named Service-Oriented SysFailRate Measurement for measuring the failure rate of the system in service-oriented architecture using fuzzy logic. In this system we used incomplete description of a service, lack of integration, incorrect format, and being out of the pre-determined time out factors to measuring the failure rate. The fuzzy model consists of four modules. The fuzzification module is the first stage in working of any fuzzy model, which transforms crisp input(s) into fuzzy values between range of 0 and 1. In this paper NoComplS, NoCompoS, NoCFormat, and Tout are input variables for system and SysFailRate is output variable. In the second stage, these values are processed in the fuzzy domain by interface engine based on production rules supplied by the domain experts. During second stage, the fuzzy operators are applied. In third stage implication process is applied and then all outputs are aggregated. In fourth and final stage, the processed output is transformed from fuzzy domain to crisp domain by defuzzification module. Testing rests demonstrated that this approach is promising to significantly improve measuring the failure rate in Service-Oriented architecture.

The article then continues as follows: the second section reviews the related literature; the third section explores the issues of motivation; the fourth section recommends the new method; the fifth section evaluates the method; and finally the discussion will follow.

## 2. Review of literature

In this section, first, the structure of service-oriented architecture and faults that occur in different parts are briefly described. In addition, a definition of reliability and its features will be stated.

### 2.1. The structure of service-oriented architecture

Service-oriented architecture is a complex architecture for designing software systems and consists of different components which connect via message. As a result, different faults may occur in different parts which increase the failure rate of the system. Therefore, different faults which may occur in this system should be identified to prevent failure and create the required services for executing the fault tolerance mechanism[1]. Figure 1, clearly shows the faults in structure of this architecture.

According to this figure faults of system are classified in three categories: faults related to Service registry, service requester and Service provider [11-13].

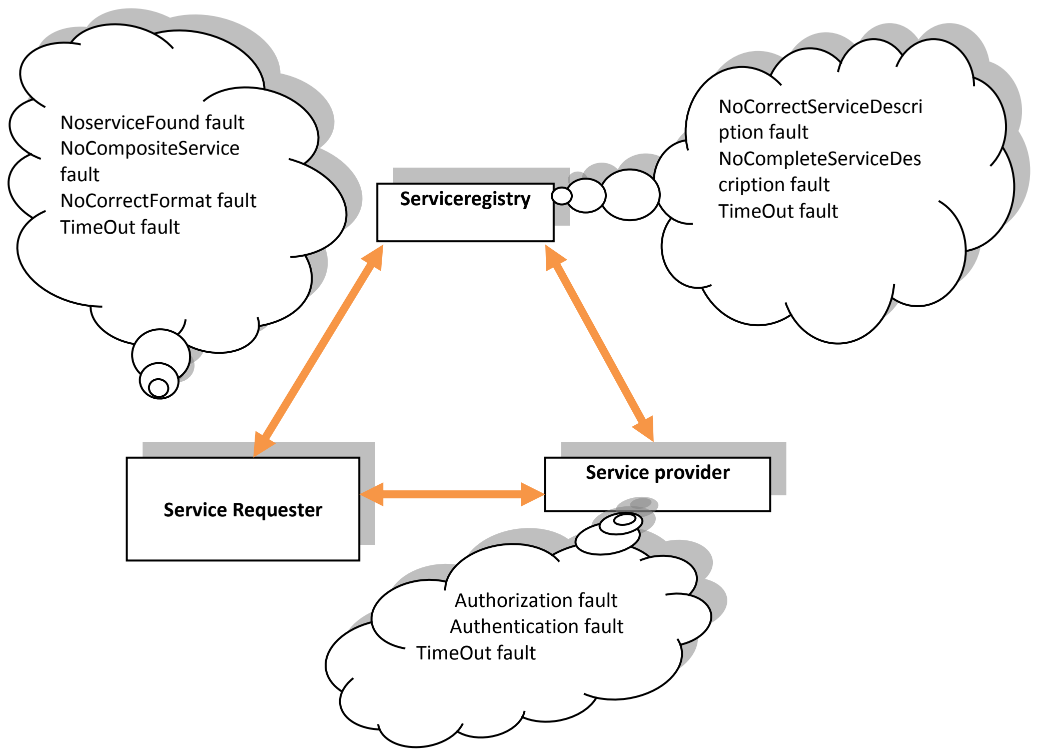


**Figure 1**. Faults in service-oriented architecture[1, 14]

## 2.2. Dependability and its Structure

Quality of service can be defined and determined by means of different parameters. Figure 2 shows this structure. These parameters are as follows:

- **Availability:** whether a web service is available and ready to use or not. It means a web service is available at a special time.
- **Accessibility:** this parameter of quality shows the extent a web service can meet a demand or a special work at a time. This quality is different from that of availability since a service may be available but not accessible. For example, the primary request may be accepted but not processed due to some other requirements.
- Reliability: this parameter shows the capability and quality of the service. One of its features is the number of faults occurred at a given time. The other characteristic is probability which shows whether a request has been made and sent correctly or not.
- **Security:** this parameter of quality shows the capability of different parts' security by using one service. This factor can be influenced by other regular factors.

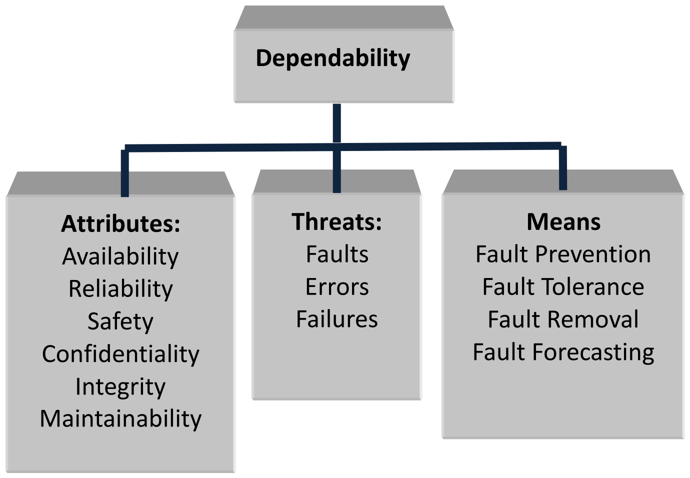


**Figure 2**. The structure of Dependability

### 2.3. Fault, Error and Failure

Faults existing in software system cause error when they activate and this faults propagate in system they will cause failure in software system. Figure 3 shows this relationship

- **Fault:** fault means a defect in a system. The existence of a fault in a system may or may not lead to fault. For example, although the entrance of a system may include fault but it may state conditions which have never caused such fault.
- **Error:** a mistake is the difference between the behaviour and the physical appearance of a system. It occurs when parts of the system face unpredictable conditions. Observing the mistakes is quite difficult without using mechanisms since they come out of inappropriate conditions.
- **Failure:** it happens when a system shows a feature which is not compatible with its special characteristics. A mistake may not necessarily lead to failure. For example, a system may encounter an exception which can be controlled by means of tolerance techniques causing a proper compatibility of the whole system's operations with the features of that exception [8, 9].

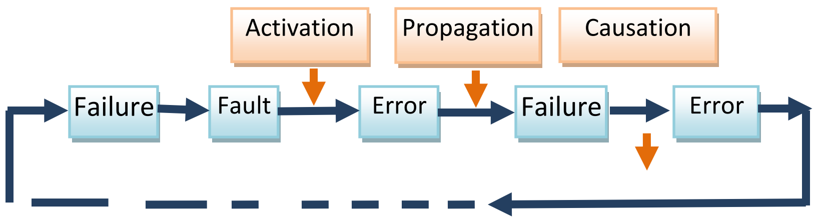


**Figure 3.**A chain of threats for reliability

## 3. Motivation

In fact, faults are parameters which affect the rate of system's failure. If such faults occur in this architecture, the existing mechanisms for fault tolerance will be used to prevent the error and thus decreasing the rate of system's failure. The Fuzzy method which states the relative validity or value of something was used in this article to measure the rate of system's failure. The reason for using Fuzzy Logic was finding the extent which this can be done exactly. The exact measurement of failure's rate can help us in using the mechanisms of fault's tolerance. Previous researches did not address the point that the rate of system's failure should be measured when faults occur. Nor did they pay attention to the point that preventing which fault is preferred over others when several faults occur at the same time or the failure's rate of which fault is more in the system. This can be done accurately by using Fuzzy Logic and thus the rate of system's failure in service-oriented architecture can be obtained. To this, faults are considered as the input parameters of Fuzzy method and failure's rate as the output of measurement system. By giving different values to faults and comparing the outputs with each other, we can find out that the failure's rate of which fault is more in the system.

## 4. Service-Oriented SysFailRate Measurement

Our proposed method makes use of linguistic variables of Fuzzy Logic for determining the rate of system's failure. Fuzzy Logic is a method based on mathematics to work with lack of finality[15]. The abbreviation for our recommended Fuzzy model to unify the method for determining the rate of system's failure is called SysFailRate. SysFailRate is influenced by factors such as incomplete description of a service, lack of integration, incorrect format, and being out of the pre-determined time out. These factors are briefly named as NoComplS, NoCompoS, NoCFormat, and Tout respectively. Figure 4 shows the total image of the recommended system.

The fuzzy model consists of four modules. Each module is one stage of presented system. The fuzzification module is the first stage in working of any fuzzy model, which transforms crisp input(s) into fuzzy values between range of 0 and 1. In this paper NoComplS, NoCompoS, NoCFormat, and Tout are input variables for system and SysFailRate is output variable. In the second stage, these values are processed in the fuzzy domain by interface engine based on production rules supplied by the domain experts. During second stage, the fuzzy operators are applied. In third stage implication process is applied and then all outputs are aggregated. In fourth and final stage, the processed output is transformed from fuzzy domain to crisp domain by defuzzification module.

### 4.1. The Membership Functions for Parameters

The rate of system's failure is between 0 – 1 in this article and the membership functions for this output variable are very low, low, medium, high, and very high. We used the Gaussian Membership for the output variable (rate of system's failure). The function of Gaussian Membership is determined by two parameters (c, σ ) :

$$\mu\,(x) = \; gaussian\,(x, c, \sigma) = e^{-\,1/2((x-c)/\sigma)2} \qquad (1)$$

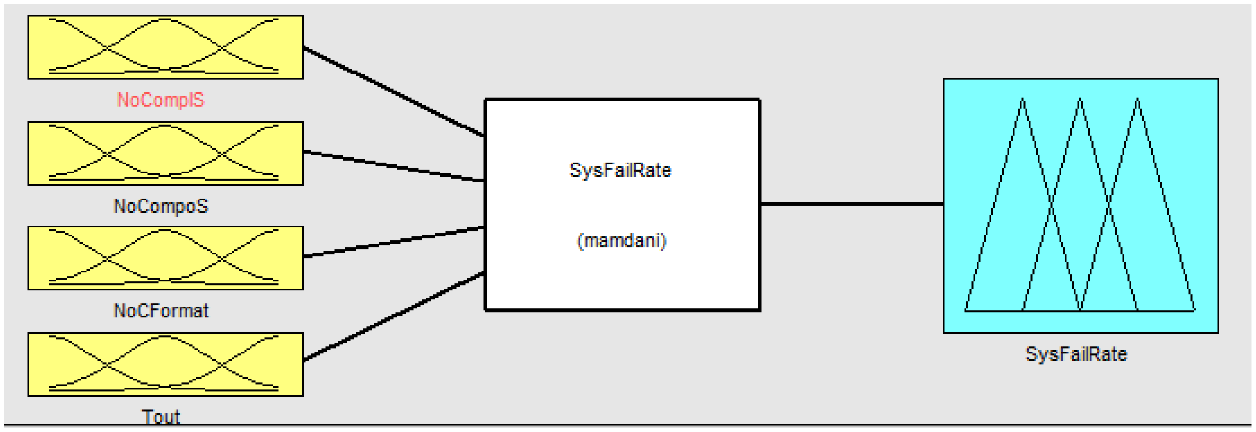


**Figure 4.**The total image of the system

In the above formula, c is the center and σ determines the width of membership function. The range of linguistic phrases for output variable (rate of system's failure) is shown in Table 1.

**Table 1.**The Range of Membership Function forSysFailRate

| Membership Function | Range |
|---|---|
| VeryLow | 0.00-0.23 |
| Low | 0.20-0.43 |
| Medium | 0.40-0.63 |
| High | 0.60-0.83 |
| VeryHigh | 0.80-1.00 |

It's worth mentioning that the overlapping of linguistic phrases increases the accuracy of measurements in fuzzy system. Figure 5 shows membership functions for system failure rate.

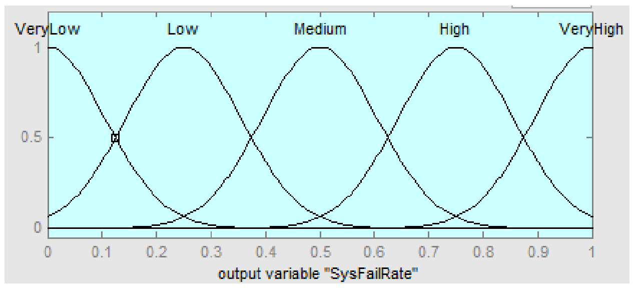


**Figure 5.**Membership Functions for System Failure Rate(SysFailRate)

Three linguistic phrases were assigned for each input parameters of NoComplS, NoCompo,NoCFormat, and Tout. Since we have three linguistic phrases for each parameter, we prevent repetition. For example, for one of the parameters like NoComplS, these phrases are: low, medium, and high. The phrases and their range will be the same for other three parameters. The phrases and their range are shown in Table 2.

**Table 2.**Membership Functions and Their Range for Input Parameters

| Membership Function | Range |
|---|---|
| Low | 0.00 – 0.36 |
| Medium | 0.33 - 0.69 |
| High | 0.66 - 1.00 |

According to the above table, the covering range for Low equals 0.00 – 0.36, for Medium equals 0.33 – 0.69, and finally for High equals 0.66 – 1.00. Figure 6 indicates such parameters and their membership functions. We used triangular membership function for these variables. A triangular membership function is specified using three parameters [a,b,c] as follows:

$$\mu(x) = tringle\ (x, a, b, c) = \begin{cases} 0 & for \quad x \le a \\ (x-a)/(b-a) & for \quad a \le x \le b \\ (d-x)/(d-c) & for \quad b < x \le c \\ 0 & for \quad c \le x \end{cases} \qquad (2)$$

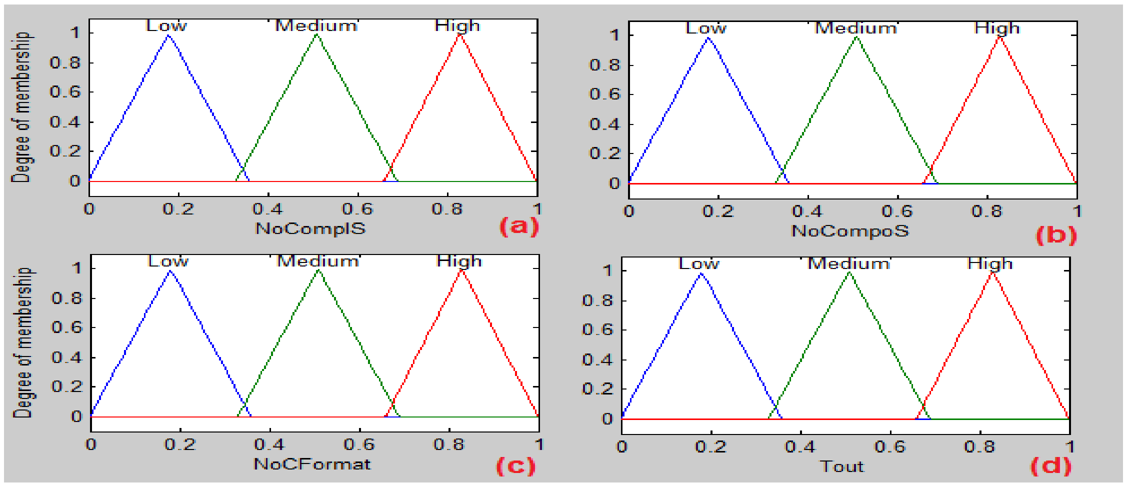


**Figure 6**. Membership Functions for Input variables**.**

Figure 6 shows the membership functions for input variables of fuzzy system. In this figure, (a) is the membership function for NoComplS and (b), (c), and (d) are the membership functions for NoCompoS, NoCFormat, and Tout respectively.

In fact, for accurately measuring the system failure rate which is the main aim of this research, we consider the effect of five parameters (NoComplS, NoCompo,NoCFormat, and Tout) simultaneously and parallel. In fuzzy systems, all rules are considered and operated simultaneously and parallel. In this article, the first parameter which is incomplete service has three linguistic phrases and thus three membership functions. Likewise, we have the same conditions for the following three parameters. Therefore, we have 34 = 81 fuzzy rule some of which are showing inTable 3.

**Table 3.**Fuzzy Rules for Proposed System

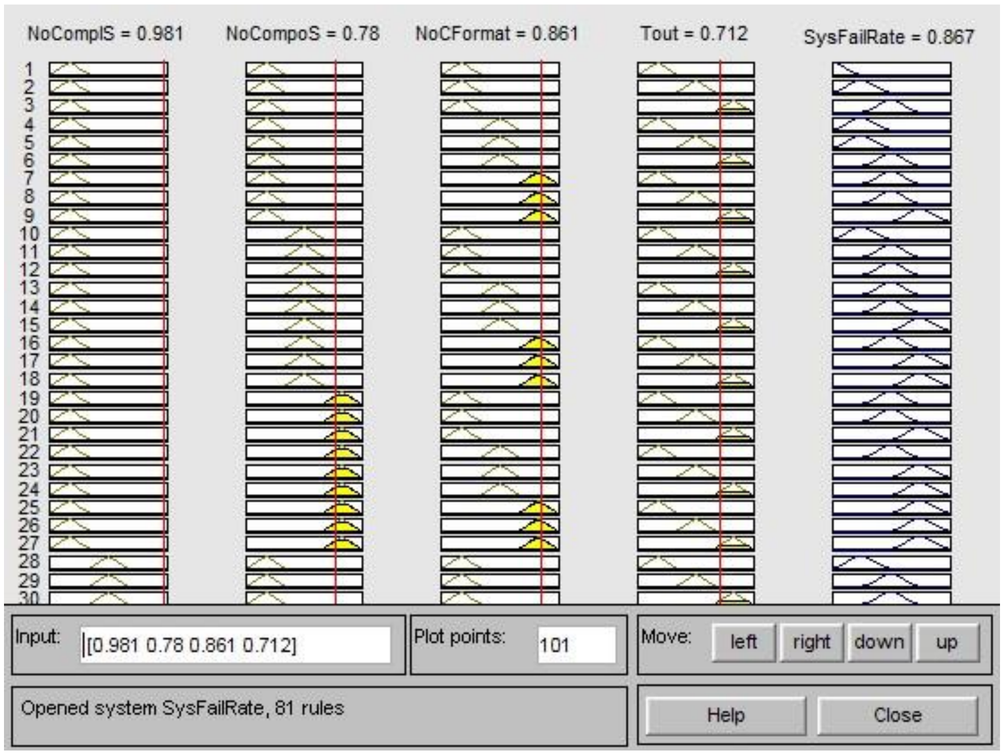
| Rule # | *NoComplS Is* | *NoCompoS is* | *NoCFormat Is* | *Tout is* | *SysFailRate Is* |
|---|---|---|---|---|---|
| 1 | *Low* | *Low* | *Low* | *Low* | *VeryLow* |
| 2 | *Low* | *Low* | *Low* | *Medium* | *Low* |
| 3 | *Low* | *Low* | *Low* | *High* | *Medium* |
| 4 | *Low* | *Low* | *Medium* | *Low* | *Low* |
| 5 | *Low* | *Low* | *Medium* | *Medium* | *Low* |
| 6 | *Low* | *Low* | *Medium* | *High* | *Medium* |

| | | | | | |
|---|---|---|---|---|---|
| 7 | *Low* | *Low* | *High* | *Low* | *Medium* |
| 8 | *Low* | *Low* | *High* | *Medium* | *Medium* |
| ... | ... | ... | ... | ... | ... |
| 79 | *High* | *High* | *High* | *Low* | *VeryHigh* |
| 80 | *Low* | *Low* | *High* | *Medium* | *VeryHigh* |
| 81 | *Low* | *Low* | *High* | *High* | *VeryHigh* |

## 5. Evaluation

In this study, we introduced a method for estimating the system failure rate using fuzzy logic. We used MATLAB software using Microsoft Windows 7 on a computer with AMD Atlon 3.2 GHz processor, 4 Gigabyte main memory. The results showed an accurate estimation. As an example, we assigned input values equal to 0.981 for NoComplS, 0.78 for NoCompoS, 0.861 for NoCFormat, and 0.712 for Tout. The output value of the system which is the system failure rate was found to be 0.867, while this case is considered as a fundamental fault regarding the input values in traditional and non-fuzzy systems(Figure 8). In our recommended system however, this case is stated as a system failure rate of 0.867 which indicates a high accuracy on the part of the system. Having this exact amount, the expert can decide on the way to treat this case.

As another example show in Figure 9, the systems was evaluated in a case in which NoComplS was assigned with a value equals 0.123, and values equal to 0.178, 0.286, and 0.312 were assigned for NoCompoS, NoCFormat, and Tout respectively. the final output of the system which is obtained by processing the incisions made on membership functions, equals 0.114.



Input: [0.981 0.78 0.861 0.712]

Plot points: 101

Opened system SysFailRate, 81 rules

**Figure 8.**An example of system that NoComplS= 0.981, NoCompoS=0.78, NoCFormat= 0.861 and Tout=0.712

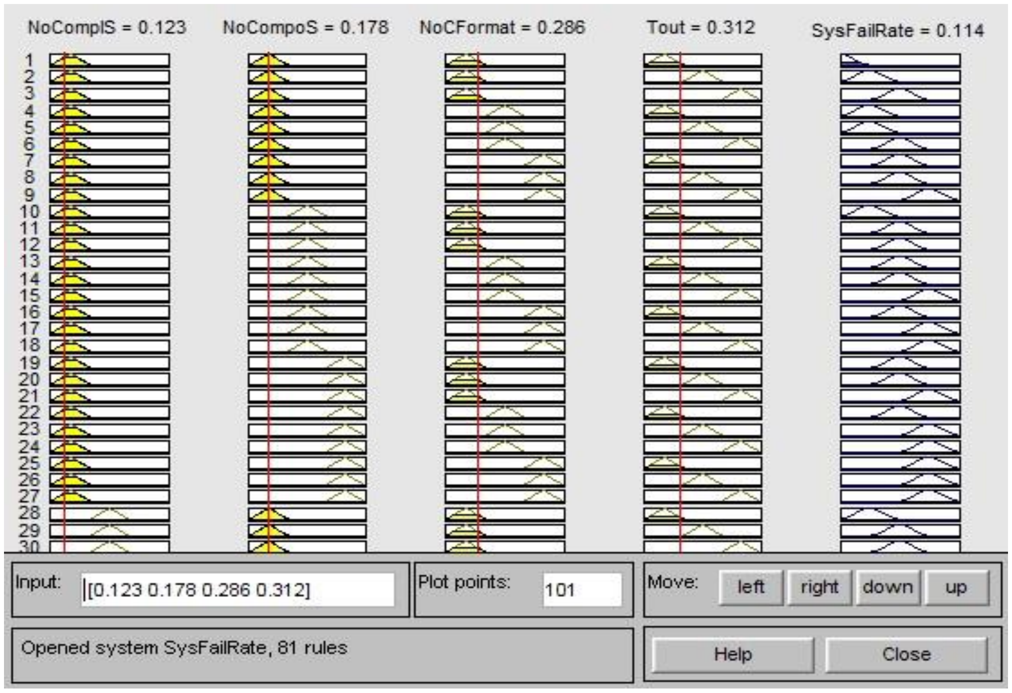


**Figure 9.**An example of system that NoComplS= 0.123, NoCompoS=0.178, NoCFormat= 0.286 and Tout=0.312

## 6. Evaluating Previous Studies

In [4], fault tolerance algorithm was introduced for creating reliability in service-oriented architecture. The idea of redundancy and integration were used in this study to prevent and keep faults constant. Fault tolerance can be repeatedly created in time and space. For operating the redundancy technique, components such as "checker", "converter", and "comparator" are used. In this article, whenever, a factor which causes fault occurs, one of the components of "checker", "converter", and "comparator" or a combination of all will be used according to the type of fault to prevent the fault. In [2], the error tolerance mechanism was analysed to model the services in service-oriented architecture. To do this, the Meta Model of service-oriented architecture was developed. This is a method which supports the error tolerance. A few components were added to the Meta Model for this purpose. A Meta Model and a number of Graph rules were recommended to observe the services and their relationship to identify the fault. In [16], faults were classified and fault injection was used to create reliability and test the system. [17] explains the framework of error tolerance and makes use of a common network based on JXTA18 Protocol to solve the problem. Taking into account the increasing protocols and to support Meta structure codes, the layer approach has been used[18]. Confirmation and mechanism of fault recognition are based on row theory to recognize or identify the services unable to perform the administrative requests. It has also provided a source for service model and a source of fault tolerance control centre from ESP19 based on our fault recognition mechanism. [19]the mechanism of fault recognition was discussed and it was mentioned that this mechanism is based on row theory to recognize the services unable to perform the requests. The mechanism of fault tolerance has also improved the mechanism for measuring the performance and can prove the accuracy of this mechanism.

## 7. Conclusion

Service-oriented architecture has a new developing method for creating distributed systems. It has a flexible and dynamic structure and for this reason it is suitable for expanding the distributing systems. In this paper we presented an effective and accurate method for measuring the system failure rate using fuzzy logic. This method include four stage and developed by Mamdaniinference[20, 21]. The system failure rate can be measured by using factors including incomplete service description, lack of service integration, incorrect format and time out, in fact this factors are inputs of proposed system. Testing rests demonstrated that this approach is promising to significantly improve measuring the failure rate in Service-Oriented architecture.

## References

[1] A. Maurizio, et al., Service oriented architecture: challenges for business and academia, In Hawaii International Conference on System Sciences, Proceedings of the 41st Annual. IEEE (2008).

[2] F. Mahdian, et al. Considering Faults in Service-Oriented Architecture: A Graph Transformation-Based Approach. In Computer Technology and Development, ICCTD'09. International Conference on, IEEE, (2009).

[3] T. Erl, Service-oriented architecture, Prentice Hall, (2004).

[4] F. Mahdian and V. Rafe, Different models of dependable services in Service-Oriented Architecture, In Advanced Computer Theory and Engineering (ICACTE), 3rd International Conference on, IEEE, (2010).

[5] F. Mahdian, et al. Modeling Fault Tolerant Services in Service-Oriented Architecture, in Theoretical Aspects of Software Engineering, TASE. Third IEEE International Symposium on, IEEE, (2009).

[6] D. Krafzig, K. Banke and D. Slama, Enterprise SOA: service-oriented architecture best practices. Prentice Hall PTR, (2005).

[7] M. Rosen, et al., Applied SOA: service-oriented architecture and design strategies, Wiley, (2012).

[8] N. Looker, et al. Pedagogic data as a basis for web service fault models, in Service-Oriented System Engineering, SOSE. IEEE International Workshop, IEEE, (2005).

[9] A. Avizienis, et al., Basic concepts and taxonomy of dependable and secure computing.Dependable and Secure Computing, IEEE Transactions, (2004) 1(1): p. 11-33.

[10] S. Bistarelli and F. Santini, Soft Constraints for Dependable Service Oriented Architectures. Architecting Dependable Systems VI, (2009) p. 76-97.

[11] M.A.C. Bhakti and A.B. Abdullah, towards an autonomic Service Oriented Architecture in computational engineering framework, in Information Sciences Signal Processing and their Applications (ISSPA), 10th International Conference on, (2010).

[12] M.A.C. Bhakti and A.B. Abdullah, towards self-organizing service oriented architecture, in Innovative Technologies in Intelligent Systems and Industrial Applications, CITISIA. IEEE, (2009).

[13] Z.M.S. Almasslawi and D.M.S. Almasslawi, A new ontology for fault tolerance in QoS-enabled service oriented systems, in Computer Science and Automation Engineering (CSAE), IEEE International Conference on, IEEE (2011).

[14] A.M. Karande, V.N. Chunekar and B. Meshram, Working of web services using BPEL workflow in SOA. Advances in Computing, Communication and Control, (2011) p. 143-149.

[15] L.A. Zadeh, Fuzzy logic = computing with words, Fuzzy Systems, IEEE Transactions on, (1996) 4(2): p. 103-111.

[16] S. Bruning, S. Weissleder and M. Malek, A fault taxonomy for service-oriented architecture, in High Assurance Systems Engineering Symposium, HASE'07. 10th IEEE, IEEE, (2007).

[17] S. Hall and G. Kotonya, An adaptable fault-tolerance for SOA using a peer-to-peer framework, in e-Business Engineering, ICEBE. IEEE International Conference on, IEEE, (2007).

[18] H. Chen and C. Zhang, A Queueing-Theory-Based Fault Detection Mechanism for SOA-Based Applications, in E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services. CEC/EEE 2007. The 9th IEEE International Conference on, IEEE, (2007).

[19] Y. Shuo and H. Chen, An Improving Fault Detection Mechanism in Service-Oriented Applications based on Queuing Theory. International Symposium on Service-Oriented System Engineering, SOSE'08,

IEEE, (2008).

[20] J. Jang, Fuzzy inference systems, Upper Saddle River, NJ: Prentice-Hall, (1997).

[21] P.J. King and E.H. Mamdani, The application of fuzzy control systems to industrial processes. Automatica, (1977) 13(3): p. 235-242.