International Journal of Innovative
Computing, Information and Control
Volume **10**, Number **3**, June **2014**

ICIC International ©2014 ISSN 1349-4198
pp. 1111-1120

ESTIMATING RELIABILITY OF SERVICE-ORIENTED SYSTEMS: A RULE-BASED APPROACH

Ashish Seth[1], Himanshu Agarwal[2] and Ashim Raj Singla[3]

[1]Department of Computer Science
[2]University College of Engineering
Punjabi University
Patiala, India
{ ashish_may13; himagrawal }@rediffmial.com

[3]Department of Information Technology
Indian Institute of Foreign Trade
New Delhi, India
arsingla@iift.ac.in

Received April 2013; revised August 2013

Abstract. *In service-oriented architecture (SOA), the entire software system consists of an interacting group of autonomous services. In order to make such a system reliable, it should inhibit guarantee for basic service, data flow, composition of services, and the complete workflow. This paper discusses the important factor of SOA and their role in the entire SOA system reliability. We focus on the factors that have the strongest effect of SOA system reliability. Based on these factors, we used a fuzzy-based approach to estimate the SOA reliability. The proposed approach is implemented on a database obtained for SOA application, and the results obtained validate and confirm the effectiveness of the proposed fuzzy approach. Furthermore, one can make trade-off analyses between different parameters for reliability.*
Keywords: Reliability estimation, SOA, On-demand, Fuzzy, Rule-based

1. **Introduction.** Service-oriented architecture (SOA) gives you the ability to more easily integrate information technology (IT) systems, provide multi-channel access to our systems, and automate business processes. Like any new investment in technology and infrastructure, it is important to understand the reliability parameters of the system when it claims to provide ad hoc solutions for dynamic requirements. One of the most important factors for any product is its reliability in producing the expected result with maximum accuracy. In software engineering, we have many testing methods available for estimating the reliability of systems. In the context of service-oriented systems (SOSs), estimating system reliability has always been a challenge.

Reliability is one of the most important non-functional requirements for software. Accurately estimating reliability for SOSs is not an easy task, and many researchers have proposed different approaches to SOS reliability estimation [1]. IEEE 610.12-1990 [2] defines reliability as "The ability of a system or component to perform its required functions under stated conditions for a specified period of time". The primary objective of reliability is to guarantee that the resources managed and used by the system are under control. It also guarantees that a user can complete its task with a certain probability when it is invoked.

Software reliability management is defined in IEEE 982.1-1988 [3] as "The process of optimizing the reliability of software through a program that emphasizes software error prevention, fault detection and removal, and use of measurements to maximize

reliability in light of project constraints such as resources, schedule, and performance". Thus any reliable system is one that must guarantee and take care of fault prevention, fault tolerance, fault removal, and fault forecasting. The most suitable models for reliability of SOAs are the ones based on architecture. Goseva-Popstojanova et al. classified three distinct approaches in architecture-based reliability modeling, these are as follows:

state-based models: analytically estimate reliability by using the control flow graph of the software architecture. These models examine the flow of control between the service components to estimate the application reliability.

path-based models: compute all possible execution paths, and reliability is estimated for sets of execution scenarios.

additive models: describe each service-component reliability and focus on estimating the time-dependent failure intensity of the system using failure data for the service components [4,5].

This paper is based on our former study [6], which we have extended to provide more detailed reliability estimation by using a fuzzy-based approach. The rest of the paper is organized as follows. First, we discuss the basic definition of SOA and services. Next, we discuss the work already done in this area in different research studies and summarize them into the form of a table in Section 2. Then the research approach for our work is defined in Section 3. Next, the fuzzy-based algorithm used for estimation is illustrated in Section 4. The experimentation and evaluation results are discussed in Section 5. Finally, the conclusion is drawn in Section 6.

1.1. **Service-oriented architecture.** There are many definitions of SOA, but none is universally accepted. What is central to all, however, is the notion of service. According to Bianco et al. [7], in an SOA system, service:

- is self-contained, highly modular, and can be independently deployed;
- is a distributed component that is available over the network and accessible through a name or locator other than the absolute network address;
- has a published interface, so the users of the service only need to see the interface and can be oblivious to implementation details;
- stresses interoperability such that users and providers can use different implementation languages and platforms;
- is discoverable, meaning users can look it up in a special directory service where all the services are registered; and
- is dynamically bound, which signifies that the service is located and bound at run time. Therefore, service users do not need to have the service implementation available at build time.

These characteristics describe an ideal service. In reality, services implemented in SOA systems lack, or relax, some of these characteristics, such as being discoverable and dynamically bound. Along with this, some of the constraints that apply to the SOA architectural style are as follows [7]:

- Service users send requests to service providers.
- A service provider can also be a service user.
- A service user can dynamically discover service providers in a directory of services.
- An enterprise service bus (ESB) can mediate the interaction between service users and service providers.

1.2. **Service.** Service is an implementation of a well-defined business functionality that operates independent of the state of any other service defined within the system. It has a well-defined set of interfaces and operates through a pre-defined contract between the

client of the service and the service itself, which must be dynamic and flexible to be able to add, remove, or modify services, according to business requirements [6]. Services are loosely coupled, autonomous, and reusable. They have well-defined, platform-independent interfaces, and provide access to data, business processes, and infrastructure, ideally in an asynchronous manner, so that they can receive requests from any source, making no assumptions as to the functional correctness of an incoming request. Services can be written today without knowing how it will be used in the future and may stand on its own or be part of a larger set of functions that constitute a larger service. Thus, services within SOA:

- provide for a network-discoverable and network-accessible interface;
- keep units of work together that change together (i.e., high coupling); and
- build separation between independent units (i.e., low coupling).

From a dynamic perspective, there are three fundamental concepts that are important to understand: the service must be visible to service providers and consumers; the clear interface for interaction between them is defined; and the real world is affected from interaction between services. These services should be loosely coupled and have minimum interdependency, otherwise they can cause disruptions when any service fails or changes.

2. **Related Work.** So far, most of the research on software reliability engineering focuses on system testing and system-level reliability growth models. Approach for the reliability analysis of evolving software systems is well-illustrated in Musa work [8]. However, SOA is not taken into account in any of these approaches. However, Goseva-Popstojanova et al. (2001) and Gokhale (2007) did remarkable work for architecture-based empirical software reliability analysis in relation to architecture-based empirical software reliability analyses [9,10].

Significant work done in the direction of estimation reliability of SOA is summarized below (see Table 1), though work stated below provides the solution for reliability, each paper has focused on a particular area and only meets a part of SOA requirements. Although the reliability of SOA systems cannot be completely estimated, we can estimate the reliability to a larger extent by analyzing the SOA characteristics and identifying the corresponding requirements. We did a thorough study in this work to identify the characteristics and define a corresponding requirement.

3. **Discussion and Research Approach.** This work is an extension of our previous work done to identify the SOA adoption trends [5]. This work is a thorough review of articles and research from the past decade (i.e., from 2001 to 2012). We have identified the factors that are relevant to SOA implementation and the extent to which each factor is crucial to SOA implementation. After filtering through a total of 200 papers, we have identified 95 articles and papers that are relevant to this study. Table 2 below shows the number of articles dealing with each factor and the percentage of the total that they represent.

In this work, we started with a set of questionnaires to identify the factors responsible for system reliability in SOA context among those identified in the first phase of our work. Using GQM (goal, question, metric) technique, metrics are proposed, and the responses are taken from 228 people in the industry. The data, which is based on the feedback and responses, is defined into the following three parameters:

1. *AR:* adhoc requirements/dynamic binding/agility
2. *MG:* migration/legacy system integration
3. *BI:* business and IT collaboration

TABLE 1. Summary of existing work in SOA reliability

S. No	Model/Year	Proposer	Descriptions	Findings	Assumptions made
1	ReServE (2011)	A. Danilecki et al. [2]	proposed a model that ensures that business processes are consistently perceived by client and services	transparently recovers the state of a business process. When a service fails, its SPU can initiate the rollback-recovery process.	For each service there is a Service Proxy Unit (SPU) that monitors the service and detects the failure.
2	SAMM (2010)	F. Brosch et al. [20]	evaluate the impact of different component topologies on the system reliability	Not only the hardware, but also different allocation configuration have influence on the predicted reliability	It lacks to provide higher-level constructs to model common fault-tolerance mechanism
3	Collaborative Reliability Prediction of Service-Oriented Systems (2010)	Z. Zibin et al. [21]	collaborative framework is proposed for predicting reliability of service-oriented systems	employs past failure data of similar service users for making reliability prediction for the current service user.	assumes service component failures are independent.
4	Unified reliability modeling framework (2009)	L. Wang et al. [17]	reliability of simple services is addressed by considering data reliability	service pools as backup alternative	Discrete Time Markov Chains (DTMCs) are used for analysing service composition reliability
5	Analyzed-stock market system (SMS) (2006)	Wang et al. [12]	Mapped to component failure probabilities and predicts the system reliability.	Derived transition probabilities from recorded transitions between components.	Recorded transitions between components via manual instrumentation of the code
6	Architecture-based reliability prediction (2005)	V. Grassi [18]	proposed an automatic and compositional reliability prediction method.	reliability is viewed as a measure of a services ability to successfully carry out its task when invoked	Used information published with each service Discrete-Time Markov Chains (DTMCs) are used for analysing service composition reliability.
7	Anlyzed SHARPE (2005)	Gokhale et al. [13]	For estimating component failure rates	found that the system reliability could be increased, if the fault density per component was reduced.	Used the enhanced non-homogeneous Poisson process model that incorporates the failure intensity of a component
8	SORM, Service-Oriented Software Reliability Model (2004)	Tsai et al. [11]	It consists of two stages: group testing to evaluate the reliability of atomic services; and evaluation of composite services through the analysis of components and their relationships.	It tries to determine the reliability of each component and their relationship.	assignment operations never generate new failures. Condition fails when any data associated with it fails. There is no cyclic dependency among entities.
9	Proposed component model (2001)	Singh et al. [15]	Proposed to transform annotated UML component models into a Bayesian model	Can be used for reliability prediction during the design phase.	Bayesian model is reliability efficient.

TABLE 2. Figures in respective columns represent number of articles; their percentages out of total.

Factors	2001-2004	2005-2008	2009-2012	Total
Governance Issues	6; 3.5	10; 5.9	8; 4.7	24; 14.3
Migration factors	5; 2.9	8; 4.7	9; 5.3	22; 13.1
Legacy Systems Integration	6; 3.5	11; 6.5	5; 2.9	22; 13.1
Change Management	3; 1.7	4; 2.3	4; 2.3	11; 6.5
Adhoc requirements	8; 4.7	9; 5.3	10; 5.9	27; 16.1
Resource Competences	1; 0.5	2; 1.1	3; 1.7	6; 3.5
Security Risk	2; 1.1	3; 1.7	3; 1.7	8; 4.7
Risk Management	1; 0.5	3; 1.7	2; 1.1	6; 3.5
Challenges in scope understanding	0; 0.0	3; 1.7	1; 0.5	4; 2.3
Integration Business and IT	3; 1.7	3; 1.7	5; 2.9	11; 6.5
Return on Investment	3; 1.7	2; 1.1	3; 1.7	8; 4.7
BPM and business agility	3; 1.7	1; 0.5	2; 1.1	6; 3.5
User involvement and Organizational Commitment	3; 1.7	0; 0.0	2; 1.1	5; 2.9
Training and Teaching Methodology	2; 1.1	2; 1.1	3; 1.7	7; 4.1
Total	46; 27.5	61; 36.5	60; 35.9	167; 100

The rules were defined for the inference engine. Three clusters were formed for the input factors (Low, Medium, and High), and five clusters were formed for the output reliability (Very Low, Low, Medium, High, and Very High). Therefore, with 3 clusters and 3 input factors, a total of 27 rules were formed that yield $3^3 = 27$ sets. These 27 sets or classifications can be used to form 27 rules using fuzzy model.

Reliability parameters for SOA-based systems with its constituent factors

1. AR: A system capable of fulfilling the ad hoc on-demand changing requirement of the market is assumed to be efficient and reliable. It is based upon the way the rule engine within the model has been trained to perform dynamic binding when the demand changes or arises. This also covers agility, which is the important issue when someone moves from present legacy systems to SOA-based systems. It is further concluded that the more reliable an SOA system is the more capability it has to handle dynamic binding/ad hoc requirement/agility, i.e., SOA reliability α AR.

2. MG: It is observed that, although an SOA system is strong enough in terms of its capacity to handle the ad hoc market, if there is no provision of integrating the legacy system or migrating successfully from old system to new one within the system; it is not effective and will not guarantee system reliability. Moreover, it is observed that mostly small and medium enterprises (SMEs) using the SOA system be developed from services developed from scratch. i.e., SOA reliability α 1/MG.

3. BI: Within a system, if the collaboration between business process and strategies is aligned with IT capabilities, the system is assumed to be more reliable. Through surveys, it has been observed that, although the powerful IT system is there, it will not be of much valuable to the organization without proper integration within the business strategies, i.e., SOA reliability α BI.

The factors described in the three parameters above assess different properties and characteristics associated with SOA model reliability. These factors can easily be measured through various available methods. The values of these parameters cannot be used independently to measure reliability. Rather, an integrated approach that considers all

three parameters and their relative impact is required for estimating a system's overall reliability.

This paper proposes a fuzzy model of SOA reliability based on the effects of ad hoc requirements, dynamic binding, agility, migration, legacy system integration, and business and IT integration. In our work, we followed Mamdani-type inference, defined it for the Fuzzy Logic Toolbox, which expects the output membership functions to be fuzzy sets. After the aggregation process, there is a fuzzy set for each output variable that needs to be defuzzified. The algorithm is discussed in detailed in the next section.

4. **Fuzzy Rule-Based Reliability Algorithm.**

1. Conduct a thorough survey of literature to identify the factors that are relevant to SOA implementation and the extent to which each factor is crucial to SOA implementation.
2. Identify reliability parameters in SOA context among these factors.
3. Cluster factors into three domain clusters of reliability parameters.
4. Assemble a database for the value of these factors.
5. Design an inference engine based on the rule for identifying reliability clusters.
6. Using Fuzzy logic, perform the following operations:
   (a) fuzzification of the input variables,
   (b) determination of membership functions for the parameters,
   (c) application of the fuzzy operator (AND) in the antecedent,
   (d) implication from the antecedent to the consequent,
   (e) aggregation of the consequents across the rules, and
   (f) defuzzification.

For defuzzification, we used the centroid technique to produce a crisp value in the range $[0, 1]$. For the SOA reliability index, we used a singleton output membership function. It enhances the efficiency of the defuzzification process, because it greatly simplifies the computation required by the more general Mamdani method, which finds the centroid of a two-dimensional function.

4.1. **Designing a rule base for the fuzzy inference engine.**

Rules were designed by considering all the possible combinations of different inputs and reliability parameters (see appendix for the complete set of rule base). All 27 rules were entered to create a rule base. Reliability for all 27 combinations was classified based on expert opinion as Very High, High, Medium, Low, or Very Low. These classifications were used to form 27 rules for the fuzzy model. Rules were fired depending on the particular set of inputs, using Mamdani-style inferences.

4.2. **Membership functions for input parameters and output parameters.**

Membership functions (MFs) were defined for fuzzifying the application. An MF is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1. Out of 11 built in a membership function type, we have used the simplest (i.e., triangular memberships function), and it has the function name *trimf*. It is nothing more than a collection of three points forming a triangle. The degree to which an object belongs to a fuzzy set is denoted by a membership value between 0 and 1. A membership function associated with a given fuzzy set maps an input value to its appropriate membership value. As an example, the input parameter AG was divided into three levels, Low, Medium and High. (the degree of membership function for all input parameters and complete inference engine is found in appendix).

5. Experimentation and Evaluation.

5.1. Ouput computation of the model.

Let us suppose that we have following input models:

Input1 = 7.5
Input2 = 6.2
Input3 = 1.8

When these inputs are fuzzified, we find that Input1 = 7.5 belongs to the Low fuzzy set with membership grade .75; Input2 = 6.2 belongs the Medium fuzzy set with membership grade .25 and the High fuzzy set with membership grade .5, and Input3 = 1.8 belongs to the Low fuzzy set with membership grade .5. With these inputs, rule 15 and 25 are fired. During composition of these rules we get the following:

Min (.75, .25, .75) = .25
Min (.75, .5, .5) = .5

When these two rules are implicated, we find that the first rule gives the Low output to an extent of .25 and second rule gives the Medium output with an extent of .5.

5.2. Defuzzification.

After the aggregation process, there is a fuzzy set for each output variable that needs defuzzification. Perhaps the most popular defuzzification method is the centroid calculation, which returns the center of area under the curve. After obtaining the fuzzified outputs shown in Table 3, we defuzzified them to obtain a crisp value for the output variable '*SOAReliability*'. For this we calculated the center of gravity (COG) of the fuzzy output.

$$\text{Output} = \frac{\int_0^{2.5} .25x dx + \int_{2.5}^{3} (mx+c)x dx + \int_3^5 .5x dx + \int_5^7 (mx+c)x dx}{\int_0^{2.5} .25 dx + \int_{2.5}^{3} (mx+c) dx + \int_3^5 .5 dx + \int_5^7 (mx+c) dx} = 4.3.$$

We simulated the effect of these rules with the MATLAB Fuzzy Logic Toolbox; the reliability for the values above turns out to be 4.3, which is very close to the calculated value.

Table 3. Rule-based output

Input1	Input2	Input3	Output	Degree of membership for output
High	Medium	Low	Medium	Min (.75, .25, .5)=.25
High	Low	Low	Low	Min (.75, .5, .5)=.5

5.3. Result and discussion.

Our experiments simulated the effect of rules with the MATLAB Fuzzy Logic Toolbox; the reliability for the values obtained was found to be very close to the calculated value, thus result obtained justifies our approach by giving accurate estimates. Our experience documented in this paper will be helpful for practitioners in collecting the data necessary for reliability prediction. Researchers are provided a demonstration on how the Fuzzy Logic Toolbox can be used to find the reliability of such system on the basis of certain SOA features. Fuzzy logic is a powerful tool which has the ability to quantify opinions with the ambiguous, contradicting and doubtful inputs.

6. Conclusion.

This paper proposes the fuzzy-based rule based approach to estimate the reliability of SOA-based systems. The estimation is based on real time factors that are grouped into parameters. Our experimental result further justifies this approach by giving accurate estimates. We simulated the effect of rules with the MATLAB Fuzzy Logic Toolbox; the reliability for the values obtained was found to be very close to the calculated value. However, the proposed approach has some limitations. It does not take

the probability of service failure into account, and there may be some other factors on which SOA reliability depends. Further, on the basis of our literature survey, we believe that the parameters we identified cover the most important SOA factors.

REFERENCES

[1] T. Kirti and S. Arun, A rule-based approach for estimating the reliability of component based systems, *Advances in Engineering Software*, pp.24-29, 2012.

[2] *http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=159342.*

[3] *http://www.baskent.edu.tr/∼zaktas/courses/Bil573/IEEE_standards/982_1_2005.pdf.*

[4] K. Goseva-Popstojanova and K. Trivedi, Architecture-based approaches to software reliability prediction, *IEEE Trans. Softw. Eng.*, vol.46, no.7, pp.1023-1036, 2003.

[5] K. Goseva-Popstojanova, A. Mathur and K. Triverdi, Comporation of architecture-based software reliability models, *The 12th International Symposium on Software Reliability Engineering*, pp.22-31, 2001.

[6] A. Seth, A. R. Singla and H. Agarwal, Service oriented architecture adoption trends: A critical survey, *Proc. of Contemporary Computing Communications in Computer and Information Science*, vol.306, 2012.

[7] P. Bianco, R. Kotermanski and P. Merson, *Evaluating a Service-Oriented Architecture*, Software Engineering Institute, 2007.

[8] J. D. Musa, A. Iannino and K. Okumoto, *Software Reliability: Measurement, Prediction, Application*, McGraw-Hill, New York, 2004.

[9] S. S. Gokhale, Architecture-based software reliability analysis: Overview and limitations, *IEEE Trans. on Dependable Secure Comput.*, vol.4, pp.132-140, 2007.

[10] K. Goseva-Popstojanova and K. S. Trivedi, Architecture-based approach to reliability assessment of software systems, *Perform Evaluation*, vol.45, nos.2-3, pp.179-204, 2001.

[11] W. Tsai, D. Zhang, Y. Chen, H. Huang, R. Paul and N. Liao, A software reliability model for web services, *The 8th IASTED International Conference on Software Engineering and Applications*, Cambridge, MA, pp.144-149, 2004.

[12] W. L. Wang, D. Pan and M. H. Chen, Architecture-based software reliability modeling, *J. Syst. Softw.*, vol.79, no.1, pp.132-146, 2006.

[13] V. Grassi, Architecture-based reliability prediction for service oriented computing, *Architecting Dependable Systems III*, pp.279-299, 2005.

[14] S. Arikan, Automatic reliability management in SOA-based critical systems, *European Conference on Service-Oriented and Cloud Computing*, pp.1-6, 2012.

[15] H. Singh, V. Cortellessa, B. Cukic, E. Gunel and V. Bharadwaj, A Bayesian approach to reliability prediction and assessment of component based systems, *Proc. of the 12th International Symposium on Software Reliability Engineering*, pp.12-21, 2001.

[16] A. Danilecki, M. Holenko, A. Kobusinska, M. Szychowiak and P. Zierho_er, Re-SerVe service: An approach to increase reliability in service oriented systems, *Parallel Computing Technologies*, pp.244-256, 2011.

[17] L. Wang, X. Bai and L. Zhou, A hierarchical reliability model of service-based software system, *The 33rd Annual IEEE International Computer Software and Applications Conference*, 2009.

[18] V. Grassi, Architecture-based reliability prediction for service oriented computing, *Architecting Dependable Systems III*, pp.279-299, 2005.

[19] S. Becker, *Coupled Model Transformations for QoS Enabled Component-Based Software Design*, Ph.D. Thesis, University of Oldenburg, Germany, 2008.

[20] F. Brosch, H. Koziolek, B. Buhnova and R. Reussner, Parameterized reliability prediction for component-based software architectures, *Proc. of the 6th Int. Conf. on the Quality of Software Architectures, LNCS*, New York, vol.6093, pp.36-51, 2010.

[21] Z. Zheng and M. R. Lyu, Collaborative reliability prediction of service-oriented systems, *ICSE'10*, ACM 978-1-60558-719-6/10/05, 2010.

Appendix I.

Inference System

```
System    Name = SOAReliabilitykirti
          Type = mamdani
          NumInputs = 3
          InLabels =
                    AdhocRqmt
                    ChangeMgmt
                    BIColloboration
          NumOutputs = 1
          OutLabels = RelaibilityImpact
          NumRules = 27
          AndMethod = min
          OrMethod = max
          ImpMethod = min
          AggMethod = max
          DefuzzMethod = centroid
Input1    Name = AdhocRqmt
          NumMFs = 3
          MFLabels =
                    ARLow
                    ARMedium
                    ARHigh
          Range = [0 10]
Input2    Name = ChangeMgmt
          NumMFs = 3
          MFLabels =
                    CMLow
                    CMMedium
                    CMHigh
          Range = [0 10]
Input3    Name = BIColloboration
          NumMFs = 3
          MFLabels =
                    BILow
                    BIMedium
                    BIHigh
          Range = [0 10]
Output    Name = RelaibilityImpact
          NumMFs = 5
          MFLabels =
                    RI_VLow
                    RI_Low
                    RI_VHigh
                    RI_Medium
                    RI_High
          Range = [0 10]
```

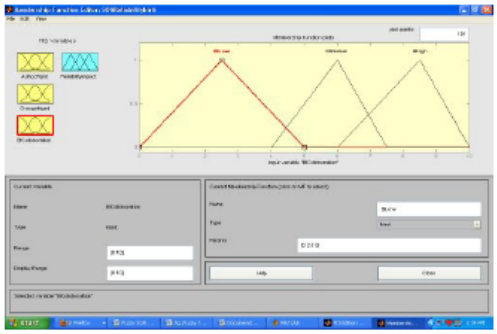
Appendix II.

Membership Function for Input and Output Parameters

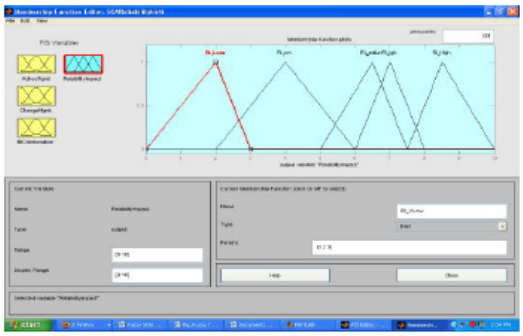
Membership Function for Input Parameter 'AR'

Membership Function for Input Parameter 'MG'

Membership Function for Input Parameter 'BI'



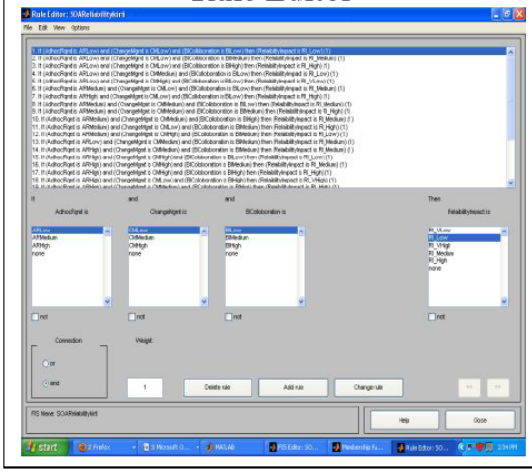
Membership Function for Output Parameter 'BI'



Appendix III.

Rule Editor and Rule Viewer

Rule Editor



Rule Viewer