International Journal of Pure and Applied Mathematics
Volume 119 No. 14 2018, 1693-1701
ISSN: 1314-3395 (on-line version)
url: http://www.ijpam.eu
Special Issue

ijpam.eu

A SEMI-MARKOV PROCESS BASED WEB SERVICE RECOMMENDATION FOR ANTI-PATTERN DETECTION USING P-EA ALGORITHM

ARUNACHALAM N[1], COUSALYA D[2], SUBATHRA P[2]
[1] Assistant Professor, [2]UG Students, Department of IT, SMVC, Puducherry

ABSTRACT - Service Oriented Architecture (SOA) is broadly preferred in industries and it is considered as the best emerging technologies. Web service (WS) is an open standard software applications that is used to make interactions with other web applications for sharing of data. At present anti-patterns detection is a manual and consumes more time for software developers. The proposed system provides an automated approach for detecting anti-patterns using P-EA algorithm (P-EA).Then the alternate web service is recommended by the suggestion maker and it is checked further based on its performance, reliability and bottle neck effect by using Semi-Markov Process (SMP) and formal sensitivity analysis technique. SMP uses BPEL for diverting existing web service to alternate web service, so that the user can quickly access new service. Restart in failed process is allowed in BPEL via fault handlers.

Keywords- Service Oriented Architecture, Web service, Parallel evolutionary algorithm, Semi-Markov Process, Business process execution language.

INTRODUCTION

Service oriented architecture is a software design that offers services through application components to a new component, through a network. SOA is autonomous of vendors, products and technologies. A service is a different group of functionality that is accessed remotely and updated independently.

Web service is a client server application that allows the communication between two electronic devices through Web. A web service is a software system that makes machine to machine communication. Web service is a group of protocols for transferring data between two applications which is a language independent way of communications that is java application can communicate with various applications.
Antipattern in web service is defined as a bad design. Normally antipattern is a bug or error or fault that is present in a web service [1]. This will leads to harm for web service. A service cannot be created with any bug in it, if any bug or error recognized in this service then it is an antipattern in that service.
A Web service description language is used to express network services or web services as a set of boundary that works on messages containing either document oriented or procedure oriented information. It is permitted to define about the restrictions and the messages sent ignoring what the message layouts or network procedures are used to communicate.

A composite web service [2] is a mixture of similar web services to execute an action. Composition of web service is completed for the end users who are not fulfils by a single web service, whereas a composite web service formed by combining similar web services is chosen by the end user. This is assumed to be an efficient and a reliable service and thus preferred to use in recent times.

BPEL stands for Business Process Execution Language [5]. WS-BPEL is an XML based language. Permitting user to define business process activities as web services and describe how they can be combined to complete specific tasks. A BPEL process description is characterized at runtime by the process service. Services that participate in BPEL defined processes are considered partner services and are recognized as part of the process definition.

1. LITERATURE SURVEY

Every web service contains a set of error/ bug which may lead to increased bug rate, fragile design and inflexible code. To address these issues antipatterns are identified from web services. Recent work [12] depend on declarative rule specification, in that rules are manually explained for the identification of key symptoms that portray a web service antipattern. According to the authors of [7], these antipatterns are evaluated by using various antipattern types. And all these types are combined together and executed parallel by using [4] P-EA based Genetic algorithm. The end user can quickly develop new service by using high level language [5]. This approach also uses Web service

code-level metrics [9], [13]. And author [6] explains about the efficiency of the web service.

There are various references that are helpful for detecting anti-patterns and providing an alternate web service. Some of them are listed below;

A. Ouni et al [3] SOA is an emerging model that has changed the way software applications are architected, designed and implemented. SOA permits developers to structure their systems as a set of ready-made, reusable and compostable services.

J. R. Koza [4] Web services are the software objects that can be organised, discovered and raised in the distributed environment of the Internet through a set of standards such as SOAP, WSDL and UDDI.

T. Andrews et al [5] QoS management is severe while considering service-oriented enterprise architectures because services have dissimilar QoS characteristics, requesters have different requirements, and service interactions are decoupled.

J. Cardoso et al [7] In an indefinite and varyingsurroundings, a composite service needs to extend its business process and service selection. To obtain the stakeholder needs, quality trade-offs are necessary to alter the composite service in reaction to the changingplatforms.

Z. Zhang et al [8] Service composition provides a flexible way to allow new application functionalities in upcoming generation networks. Composite services are twisted to be more and more complex, and its performance analysis is significant cause for service providers who need to get better QoS.

M. D. Hansen [10] concentrates on analysis of key middleware technologies for recognition of SOA on Java platform. It donates to the understanding of functional and performance features of distributed middleware technologies for recognition of SOA.

C. Mateos et al [11] consists of information from real-world instances of Web service anti-patterns to produce detection rules based on mixtures of metrics and threshold values. It estimates theapproach on a standard of many Web services and five types of Web service anti-patterns.

F. Palma et al [12] compare the execution of cooperative P-EA Approach with random search, two single population-based approaches and one state-of-the-art detection technique not based on heuristic search. Statistical analysis of the attained results reveals that this approach is efficient for detecting anti-pattern, with 89 present of precision and 93 present of recall.

B. Dudley et al [14] initially manual detection of anti-patterns consumes more time. To overcome this problem, we propose a general anti-pattern detection approach for Java EE application. So, we propose a Java EE meta-model, based on that, we use QVT language to specify the detection process of anti-patterns.

M. Harman [16] defines work on the application of optimization methods in software engineering. These optimization methods come from the operations research and met heuristic computation research communities.

A. He et al [17] Since the total number of Web services grows and the diversity increases, it is overbearing to bring semantics to Web services to offer richer explanations. However, various solutions established in the field of semantic annotation suffer from lack of relevant ontologies that are inclusive enough to have as many ideas as possible.

J. Ma et al [19] Composition of web-based services is a procedure that usually needs progressive programming skills and massive knowledge about specific technologies. How to move out web service composition permitting to functional sufficiency and performance is widely studied. Consuming complicated boundary and loading conditions needs FE analysis in order to define accurately enough objective functions. The Results of such class of problems become hard as we use probabilistic algorithms.

J. Kraal et al [26] Growing quantity of articles for the SOA problems specifies that SOA faces extensive problems. It can be a value of the fact that SOA passes the discussion phase of the hypo curve. And it must have some concrete reasons.

R. Marines cu et al[27]while supporting the maintenance of an object-oriented software system, the quality of its design must be estimated using adequate quantification means. In spite of the current wide use of metrics, if used in isolation metrics are oftentimes too

fine grained to quantify systematically an investigated design aspect.

2. EXISTING APPROACH

In existing model, antipattern will be detected in each and every web service which is illustrated in fig 3.1. It uses parallel Evolutionary Algorithm (P-EA) for detecting the antipatterns in a web service. The presence of antipatterns will leads to increased bug rate, fragile design, inflexible code etc. So antipatterns should be detected momentarily as possible. The P-EA algorithm in fig 3.2 combines all the antipattern types together and executes them parallely for the detection of antipatterns [4]. P-EA approach is used because it combines different perspectives and levels to detect antipattern. The diagrammatic scheme for Parallel Evolutionary Algorithm (P-EA) has been figured below.

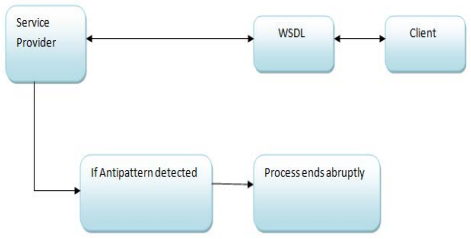


Fig3.1Existing Framework

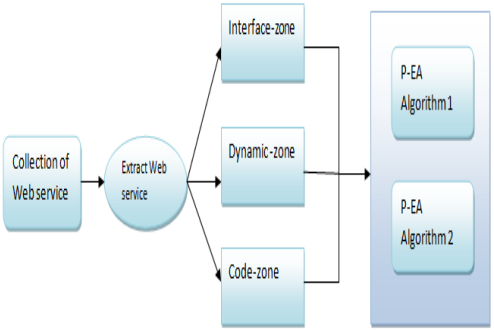


Fig3.2 Cooperative parallel evolutionary algorithms

3. PROPOSED FRAMEWORK

This system is to detect antipattern in web services by using PEA algorithm. The Cooperative parallel evolutionary algorithm which combines all the types of detection methods and executes them parallely to detect the antipattern. When the antipattern has been detected in a particular web service the client cannot able to proceed further. So this approach suggest an alternate web service for the client which is similar to the existing web service. This can be done with the help of the suggestion maker. And finally the new web service will be checked further for its efficiency such as performance, reliability and bottleneck problems. The performance and reliability is checked by using Semi Markov Process (SMP) and bottleneck problem is checked by using formal sensitivity analysis technique. The efficient web service will be choosed and it is directly suggested for the client. The proposed framework is illustrated in fig 4.

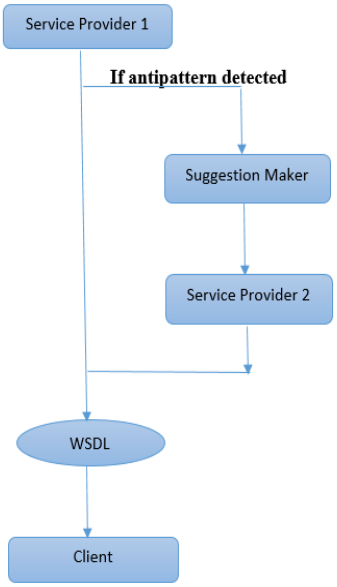


Fig 4.1 Proposed approach

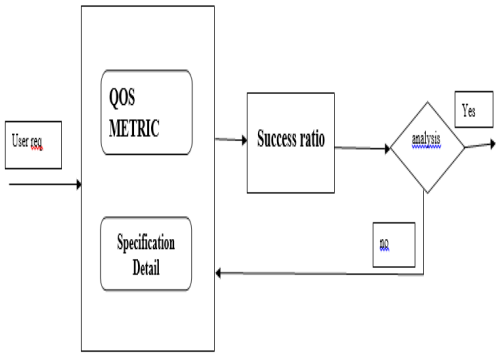


Fig 4.2 QOS Metric

A. Anti-pattern Detection

Initially the detection of antipatterns in web service is manual and it takes much time for the tester. Even though many testing process are done, each and every web service still contains some bug in it. Thepresence of antipattern in web service will leads to increased bug rate, fragile design, inflexible code etc…So an automated approach for detecting antipattern has been included, which uses Parallel Evolutionary Algorithm(P-EA)[4]. This algorithm will combine all the detection methods and executes them parallely and finally detect the antipattern. It includes eight different antipattern types in it and combines all the types together and executes them parallel. These eight antipatterns types [1] are itemized below;

1. God object web service (GOWS)
2. Fine grained web service (FGWS)
3. Chattyweb service (CWS)
4. Data web service (DWS)
5. Ambiguous web service (AWS)
6. Redundant Port Types (RPT)
7. CRUDy Interface (CI)
8. Maybe it is Not RPC (MNR)

B. Illustrative Examples

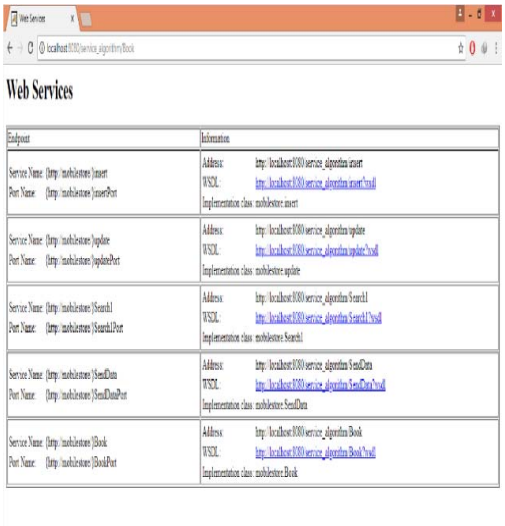
The core identifying aspect of a FGFS, we consider the calculator application provided by Apache Geronimo. Normally simple calculator service is not much complex; which supports numerous simple operations such as addition, subtraction, multiplication, division. Fig shows the WSDL code for the calculator service, which performs the addition operation. It is a simple service. However this includes large number of code for simple operation. Since all services are done through internet, they may be bound by the limitations and costs incurred by communications over those networks.

```
<wsdl:portType name="CalculatorPortType">
   <wsdl:operation name="add">
     <wsdl:input name="add" message="tns:add"/>
     <wsdl:output name="addResponse" message="tns:addResponse"/>
   </wsdl:operation>
</wsdl:portType>

<wsdl:binding name="CalculatorSoapBinding" type="tns:CalculatorPortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/
```

Fig. 4.1. An example of FGWS type of detection

Fig 4.2 illustrates the sample demo of Redundant Port Types (RPT). In this type of antipattern, multiple port Types are replicated with same set of processes. Very frequently, such Types contract with similar messages. Redundant Port Type antipattern may contain some negative impression on considering web services [17].



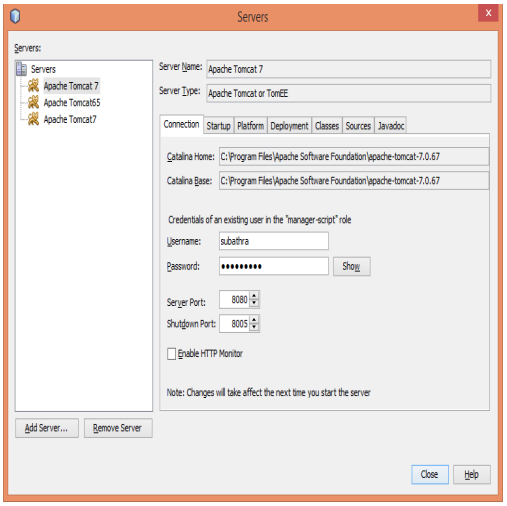


Fig. 4.2. A sample demo of redundant port types of antipattern

C. Alternate Web Service

After detecting the antipattern in a particular web service, the end user may find difficult to access that web service further. And it's a major drawback for the client. So here we suggest an alternate web service for the client to fulfil the client's needs. Suggestion maker contains the entire metaheuristic data of the service provider. Then it will provide the alternate web service by matching the similarity between those two web services. The suggestion maker uses Genetic algorithm [4] for suggesting the alternate web service. It includes two types of algorithm. The first method employs an Evolutionary algorithm based on genetic programming for the purpose of generating the rules for detection of antipattern. And the next method will executes parallel and produce an evolutionary algorithm which generates detectors from web service.

D. Checking Efficiency Of New Web Service

The suggested web service will be checked further for its efficiency. The efficiency includes performance, reliability, and bottlenecks. The performance [8] and reliability is checked by using Semi Markov Process (SMP). The BPEL process related with composite web services can be treated as a Semi-Markov process. The bottleneck problem is checked by using formal sensitivity analysis technique that includes the computation of derivatives of system measures with respect to various model input parameters. Parameters with more sensitivities regularly earn close consideration in the quest to increase system characteristics.

E. Semi-Markov Process

On considering stochastic process, we assume that a state represents a service with one entry point and a one exit point and the Markov property is satisfied at the time of entry into each state.

In the following, we will show the Markov renewal model and its derivation in detail.

Assume that A = {0,…, x} and (S_y, A_y) and y= 0, 1,… be a Markov renewal process. Also a homogeneous Markov chain with the phase space a× [0, ∞, an initial distribution

M = M_i= M {S_0 = i, A_0 = 0} = M {S_0 = i}, i∈A) and transition probabilities, for y= 0, 1

$Q_{ij}(t)=P\{J_{n+1}=j,X_{n+1}\leq t/J_n=i,X_n=s\},(i,s),(j,t)\in X\times[0,\infty)$ ……(1)

In this case (the transition probabilities do not depend on s), the random sequence η_n, n= 0, 1... is also a homogeneous (embedded) Markov chain with the phase space X and the transition probabilities, for n = 0, 1,

$Pij= P\{J_{n+1} = j/J_n= i\} = Q_{ij}(\infty)$, i, j∈X…….(2)

As far as random variable $X_n$is concerned, it can be interpreted as so journey time in state J_n, for n=1, 2…

Assume that the following communication conditions hold:

X is a communicative class of states for the embedded Markov chain J_n. Also assume that the following condition excluding instant transitions holds:

$Q_{ij}(0) = 0$, i,j∈ X.

Let us now introduce a semi-Markov process, $J(t) = J_{N(t)}$, t ≥0, (3)

where$N(t)= \max(n \geq 0: T_n \leq t)$ is the number of jumps in the time interval 0,t, for t ≥0, and $T_n = X_1+\ldots\ldots X_n$,

n= 0,1…..are the sequential moments of jumps, for the semi-Markov process J(t).

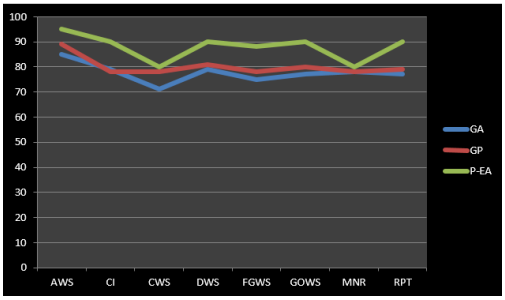
This process has the phase space X, the initial distribution $\bar{p} = (p_i = P\{J(0) = i\}, i \in X)$ and transition probabilities $Q_{ij}(t)$, $t \geq 0$, $i, j \in X$.

F. Bottleneck Detection

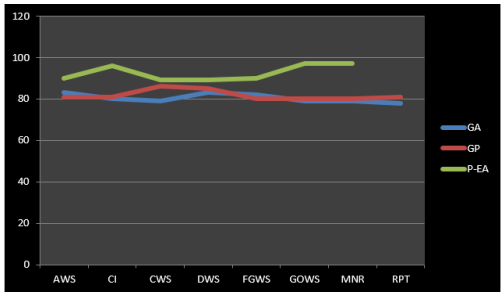
These derivatives can again use to identify the bottleneck [18]. We divided the bottleneck detection into two groups: The first is bottleneck on services and the next is bottleneck on model branching probabilities. Each and every alternate web service will be checked for its efficiency and it will also evaluate both the mean and variance for the entire response time and reliability of composite web service. And finally the web service which is more efficient will be choosing and it is directly suggested for the client. Then the client can directly access the new web service easily.

4. RESULTS

In Fig. 5.1, it denotes that P-EA not has an attachment to the antipattern detection. As labelled the figure, we had nearlyidentical distribution of each antipattern type. On considering weather web services, the delivery is not much stable. It is due to the number of antipattern types detected. Usually, all the eight antipattern types are detected with good precision and recall scores (more than 85 present). Most existing guidelines [12], [14] rely heavily on the notion of size to detect antipatterns. This is sensible for antipatterns like GOWS and FGWS which is related with a notion of size, but for antipatterns like AWS, the notion of size is less important and this makes this type of anomaly hard to detect using structural information.



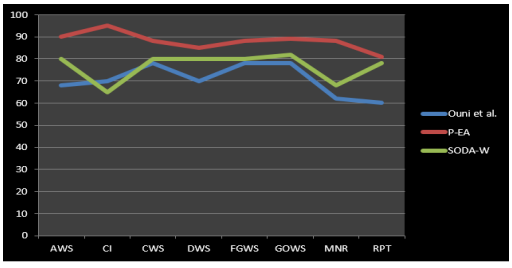
a) Precision



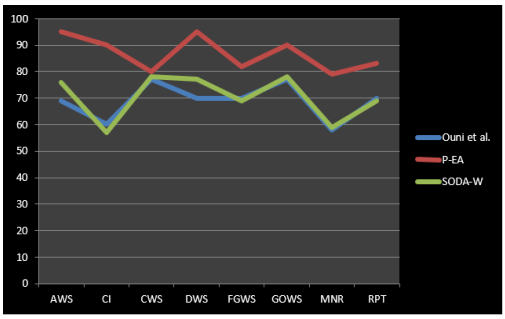
b) Recall

Fig 5.1. Detection results for each antipattern type

In Fig. 5.2 denotes the evaluation result of P-EA, Ouni et al. [3], and SODA-W. While SODA-W specifieshopefulspecial effects with an average precision of 71 present and recall of 83 present, still that is less than P-EA in all the eight considered antipattern types. The main difficult with SODA-W is that it converts the various symptoms which are useful for the detection of specific antipatterns. Certainly, SODA-W is limited to a set of WSDL interface metrics, but neglects the source code of the web service artifacts. In fact, service design may look skilled at the interface level, but that is suggestion to be an antipattern when the source code is not proper design. Additional drawback is number of antipatterns for manual detection can be very large, and then the rules which are expressed in the way of metric mixtures need significant calibration efforts to find the suitable beginning value for each metric. By contrast, this needs some examples of antipatterns to generate detection rules. Fig. 8 also shows that our previous work [3] offers lower detection results to the eight antipattern types with an average of 72 present for both precision and recall.



a) Precision



b) Recall

Fig. 5.2. Comparison results of P-EA, Ouni et al. and SODA-W.

5. CONCLUSION

Thus the SBSE approach is used for detecting web service anti pattern based on Cooperative parallel evolutionary algorithm which executes all the antipattern detection methods parallel and this will redirect the alternate web service to the client by the use of suggestion maker. Also check the efficiency of the new web service such as performance, reliability and Bottle neck problems by using Semi Markov Process. Bottle neck problem is also detected by using formal sensitivity analysis technique. The Semi Markov Process will compute the mean and variance of entire response time and reliability of composite web service. The enhancement of this idea is to evaluate this antipattern detection types on both individual Web services and business process. And, another promising research oriented is to perform the correction, for all the detected antipatterns.

REFERENCES

[1] Ali Ouni, MarouaneKessentini, Katsuro Inoue, "Search-Based Web Service Antipatterns Detection," in IEEE Transactions on services computing, vol. 10, no. 4, 2017.

[2] ZhengZheng, Kishor S. Trivedi, Fellow, Kun Qiu, and Ruofan Xia, "Semi-Markov Models of Composite Web Services for their Performance, Reliability and Bottlenecks," in IEEE Transactions on services computing, vol. 10, no. 3,2017

[3] A.Ouni, R. Gaikovina Kula, M. Kessentini, and K. Inoue, "Web service antipatterns detection using genetic programming," in proc. Genetic Evol. Comput. Conf., 2015, pp. 1351-1358.

[4] J.R. Koza, Genetic programming: On the Programming of computers by means of Natural Selection, Vol.1.Cambridge, MA, USA: MIT Press, 1992.

[5] T. Andrews and F. Curberaet all.., "Business process execution language for web services," 2003.

[6] H.Zheng, J.Yang, W.Zhao, and A.Bouguettaya, "QoS analysis for web service compositions based on probabilistic QoS," in Proc. 9th Int. Conf. Service-oriented comput.,2011, pp. 47-61.

[7] J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K.Kochut, "Quality of service for workflows and web service processes," Web Semantics: Sci., Services Agents World Wide Web, vol. 1, no. 3, pp. 281-308, 2004.

[8] Z. Zhang, Z. Yang, and Q. Liu, "Performance analysis of composite web service," in Proc. IEEE Int. Conf. Granular Comput., 2008, pp. 817-821.

[9] J. L. O. Coscia, M. Crasso, C. Mateos, and A. Zunino, "Estimating web service interface quality through conventional object-oriented metrics," CLEI Electron. J., vol. 16, no.1, 2013.

[10] M. D. Hansen, SOA using Java Web Services. New York City, NY, USA: Pearson Education, 2007.

[11] C. Mateos, A. Zunino, and J. L. O. Coscia, "Avoiding WSDL Bad Practices in Code-First Web Services," SADIO Electron. J. Informat. Oper.Res., vol. 11, no.1, pp. 31-48, 2012.

[12] F. Palma, N. Moha, G. Tremblay, and Y. G. Guesheneuc, "specification and detection of SOA antipatterns in web services," in Proc. 8th Eur. Conf. Software. Archit., 2014, pp. 58-73.

[13] J. Coscia, M. Crasso, C. Mateos, A. Zunino, and S. Misra, "Predicting web service maintainability via object-oriented metrics: A statistics-based approach," in Proc. 12th Int. Conf. Comput. Sci.Its Appl., 2012, vol. 7336, pp. 29-39.

[14] B. Dudley, J. Krozak, K. Wittkopf, S. Asbury, and D. Osborne, J2EE Antipatterns. Hoboken, NJ, USA: Wiley, 2003.

[15] C. Mateos, A.Zunino, and J. L. O. Coscia, "Avoiding WSDL Bad Practices in Code-First Web Services," SADIO Electron. J. Informat. Oper.Res., vol. 11, no.1, pp. 31-48, 2012.

[16] M. Harman, "The current state and future of search based software engineering," in Proc. Future Softw. Eng., 2007, pp. 342-357.

A. Heb, E. Johnston, and N. Kushmeric, “ASSAM: A tool for semi-automatically annotating semantic web services,” in Proc. 3rd Int. Semantic Web Conf., 2004, vol. 3298, pp. 320-334.
[17] J. T. Blake, A. L. Reibbman, and K. S. Trivedi, “Sensitivity analysis of reliability and performability measures for multiprocessor systems,” ACM SIGMETRICS Perform. Eval.Rev., vol. 16, no. 1, pp. 177-186, 1988.
[18] J. Ma and H. P. Chen, “A reliability evaluation framework on composite web service,” in Proc. IEEE Int. Symp. Service-Oriented Syst. Eng., 2008, pp. 123-128.
[19] K. S. Trivedi, Probability and Statistics with reliability, Queuing and Computer Science Applications. New York, NY, USA: Wiley, 2002.
[20] U. N. Bhat and G. K. Miller, Elements of applied Stochastic Processes. New York, NY, USA: Wiley, 2002.
[21] E. P. Kao, “A note on the first two moments of times in transient states in a Semi-Markov process,” J. Appl. Probability, vol. 11, pp. 193-198, 1974.
[22] C. Davis, “The norm of the Schur product operation,” Numerische Math., vol. 4, no. 1, pp. 343-344, 1962.
[23] J. Medhi, Stochastic Processes. New Delhi, India: New Age International, 1994.
[24] J. Fox, An R and S-plus Companion to Applied Regression. Newbury Park, CA, USA: Sage, 2002.
[25] J. Kraal and M. Zemlicka, “Popular SOA antipatterns,” in Proc. Computation World: Future comput., Service Compute., Cognitive, Adaptive, Content, Patterns, 2009, pp.271-276.
[26] R. Marines cu, “Detection strategies: Metrics-based rules for detecting design flaws,” in Proc. IEEE Int. Conf. Software. Maintenance, 2004, pp.350-359.
[27] DmitriiSilvestrov, RaimondoManca. “Reward Algorithms for Semi-Markov Processes", Methodology and Computing in Applied Probability, 2017