Volume 6, Issue 7, July 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Improve the Accuracy of the Current State-of-the-art Prediction of Atomic Web Services Reliability for QOS Aware Recommendation

Dr. V. Goutham, B. Shyla Reddy, B. Priyanka
Teegala Krishna Reddy Engineering College, Meerpet,
Telangana, India

***Abstract:** It is essential to assess non-functional properties of potential service selection candidates during the construction of QoS-aware compound work-flows based on service oriented systems. CLUS, a model for reliability prediction of atomic web services that assess the consistency for an ongoing service invocation constructed on the data accumulated from preceding invocations. To progress the correctness of the present state-of-the-art prediction models, it is here assimilated user-service and environment-specific constraints of the invocation context. To reduce the scalability issues present in the state-of-the-art approaches, the previous invocation data using K-means clustering algorithm is amassed.*

***Keywords:** K-means clustering, prediction, QoS, recommendation, reliability, web services*

I. INTRODUCTION

SOA is an architectural style mostly used for design of information systems for various enterprise solutions, but is also usually applied in a wide-ranging context. The elementary functionalities are usually created as reusable atomic services accessible through publicly available interfaces. To provide more advanced functionalities, SOA allows designers to comprise the atomic services into more complex ones. While erecting compound services, it is vital for the developer to select high quality atomic service candidates [1], [2], as the application quality trusts on both functional and non-functional qualities of the selected candidates. To create an efficient compound application, the developer should be provided with reliable information on both atomic amenities' functionalities and their non-functional dependability attributes, such as response time, reliability, availability [3]. This prediction concept is focused on atomic service reliability, as one of the most important non-functional properties. Here, service reliability is defined as the probability that a service invocation gets completed successfully. Reliability on demand description is more expedient for web services since service invocations are distinct and relatively rare events. According to the implemented definition, the reliability value can be totalled from the past invocation sample as the ratio of the number of successful against the number of total performed invocations. However, obtaining an inclusive past invocation sample proves to be a very challenging task for several reasons. There is a familiar changeability characteristic for service oriented systems in reliability perception from the user's and service provider's standpoint [6]. In general, the reliability value computed considering exclusively the data attained by the service provider can be unrepresentative for a specific user because of the oscillations presented by a variety of invocation context parameters. From user's perspective, further obstacles while collecting data are related to the service usage cost and performance issues. A strategy to overcome the aforementioned reliability assessment challenges is to obtain partial but pertinent history invocation sample, and to utilize prediction algorithms to assess reliability for the missing records. Thus, the following scenario can be employed to recommend the most suitable service candidates. While accessing various services on the Internet, users perceive different reliability properties depending on the given invocation context. The partial invocation sample can be gained by gathering live feedback regarding service usage through collaborative feedback [9], and assembling as many data records as possible from the service providers by performing service monitoring [11]. Prediction algorithms can then be used to estimate reliability for future service invocations based on the collected past invocation sample .Finally, the most reliable service candidates can be recommended according to the predicted reliability values. The academics have projected several prediction models based on collaborative filtering practise often used in modern recommendation systems [12]. Even though the existing collaborative filtering based methods attain end owed performance, they prove complications primarily related to the prediction accuracy in dynamic environments and scalability issues caused by the invocation sample size. Regarding the prediction accuracy, collaborative filtering provides accurate recommendations in static environments where the collected data records are relatively stable. This means that the records remain up-to-date for a reasonably long period of time (e.g. song ratings, product recommendations). However, service-oriented systems are deployed on the Internet, which is a very dynamic environment where service providers register significant load variations during the day [16]. In such a dynamic environment, user perceived service reliability may considerably differ depending of the actual time of invocation. Furthermore, collaborative filtering approaches store reliability values for each user and service pair. Having millions of users and a substantially large number of services, these approaches do not scale.

II. RELATED WORK

A myriad of different approaches for modelling the reliability of traditional software systems have been proposed in the literature [7]. Still, web services are dynamic software artifacts that provide their functionalities via publicly accessible interfaces over the Internet. The Internet is a very dynamic environment in which the service invocation outcome depends on a variety of different impacts that determine the invocation context. As a consequence, the traditional approaches for modelling software reliability are not suitable for assessing the reliability of web services. While designing new models for service oriented systems, the majority of researchers usually focus on studying the reliability of service compositions. Various approaches for predicting the reliability of composite services have been proposed [6]. All these approaches usually assume the atomic service reliability values are already known or rarely suggest how they can be acquired. However, the arguments stated in Section 1 indicate that collecting a comprehensive sample of reliability values is a very difficult task in practice. The most successful approaches for prediction of atomic service reliability are based on the collaborative filtering technique [12]. According to the related literature [12], the basic types of collaborative filtering are: memory-based, model-based and hybrid. The following sections briefly describe each type of collaborative filtering.

2.1 The Memory-Based Collaborative Filtering

The memory-based collaborative filtering is a commonly used technique in nowadays state-of-the-art recommendation systems [7], [8], [9]. This filtering technique extracts the information or patterns by statistically correlating the data obtained from multiple entities like agents, viewpoints or data sources. The benefit of memory based collaborative filtering is that lacking information for a particular entity can be predicted using the available data from the most statistically similar entities. This collaborative filtering type uses user-item matrix to store the data for reliability prediction. Each pui value in the matrix represents the reliability of the service i perceived by the user u. In real service-oriented systems, matrices can contain millions of user and services, while new user-service pairs arise in real time. Furthermore, each user accesses only a small subset of services. Consequently, the user-item matrix is extremely sparse and contains a significant amount of empty cells reflecting missing reliability values that need to be predicted. Memory-based collaborative filtering can be applied in two different ways. The UPCC combines the information collected from different users and predicts missing reliability values using the available data from the most statistically similar users [32]. The IPCC collects the data from different services and predicts missing reliability values based on available values from the most statistically similar services [3]. The Hybrid approach [9], [10] achieves better prediction performance by employing both the data retrieved from similar users and services and predicting missing reliability values as a linear combination of UPCC and IPCC.

2.2 The Model-Based Collaborative Filtering

The model-based collaborative filtering approaches are known to be more computationally complex and difficult to implement. These approaches often combine more complex techniques such as machine learning or data mining algorithms to learn the prediction model by recognizing complex patterns using the training data, and then use the model to make predictions on the real data [12]. For instance, Yu et al. propose a trace norm regularized matrix factorization based approach for prediction of web services reliability [4]. Similarly, Zheng and Lyu also utilize matrix factorization in their approach that assumes a small number of factors that impact the user perceived reliability [35]. However, the mentioned approaches do not include any environment-specific parameters into the prediction process. Typical representatives of model-based collaborative filtering are the approaches based on the linear regression technique [6], such as [7], [8]. Linear regression is also found most suitable for numerical prediction domains. Since there is no existing linear regression models that incorporate environmental parameters in the field of web services reliability prediction.

2.3 The Hybrid Collaborative Filtering

The hybrid collaborative filtering approaches can be very effective in addressing disadvantages of basic memory based collaborative filtering [9]. However, the main disadvantage of these approaches is that their prediction capability often relies on additional domain specific data describing the internals of a system. It proves to be a challenging task to obtain such data in practice. In our recent work [10], we addressed the disadvantages of the collaborative filtering based approaches by improving prediction accuracy and scalability. However, the model we proposed (LUCS) is applicable in the environments where the model's input parameters are highly available. For example, we group services into service classes considering service's computational complexity and we assume each service's class is explicitly known as the input parameter. As the amount of services with missing input parameters increases the prediction accuracy deteriorates. These deficiencies are addressed by CLUS and linear regression approaches described in the following sections.

III. SYSTEM DESIGN

3.1 Clus Prediction Overview

CLUS, a model for prediction of atomic web services reliability. With the aim of improving the prediction accuracy, we define user-, service and environment-specific parameters that determine service invocation context more exactly than the related prediction models. Furthermore, to achieve scalability, we group the collected invocation sample across three different dimensions associated with the defined parameters using the K-means clustering algorithm. The comparative benefit of K-means among other clustering algorithms [11] is that it converges accurately and very quickly when performed on the available service reliability data.

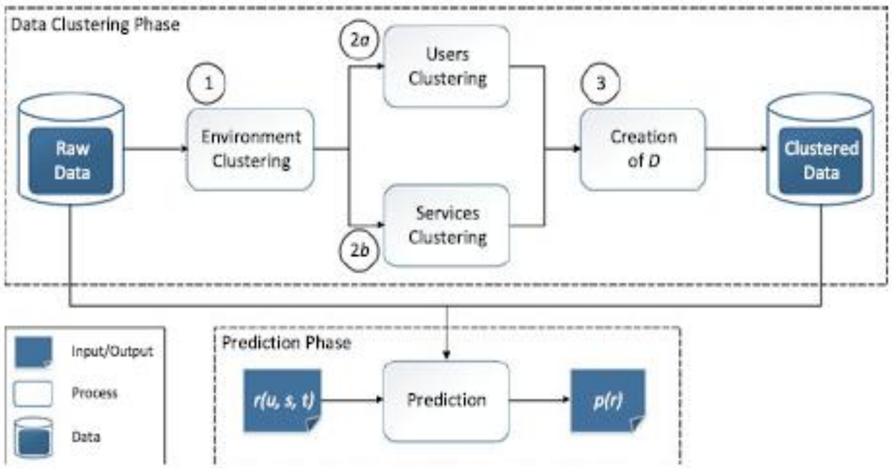


Fig. 1. Reliability prediction overview

Earlier to reliability prediction, we accomplish bunching of the history invocation sample. First, we cluster the time windows associated with the environment conditions according to the reliability performance fetched from the past invocation sample (1). Then, we cluster users and services considering their reliability performance within each time window cluster. Finally we create a three-dimensional space D containing clustered data (3). Once the clustering phase is done, prediction of the atomic services reliability can be performed in the prediction phase.

IV. IMPLEMENTATION: CLUS PREDICTION MODEL

It describes each step of the data clustering process used in CLUS and defines how predictions are calculated from the clustered data. The past invocation sample contains data addressed in order to make scalable and accurate reliability predictions for future service invocations, the data needs to be transmuted into a more structured and compacted form. Hence, we store the data into a three-dimensional space each dimension u, s and e is associated with one group of parameters. In the following sections we describe how particular records are clustered and associated with the corresponding parameters. Finally, we describe the creation of space D, i.e. how each entry in D is calculated and how the reliability is predicted for an ongoing service invocation.

4.1 Environment-Specific Data Clustering:

The intention to correlate each available history invocation record with the service provider load at the time was performed. As already stated, analyses of the collected data from different service providers can discover regularities in the load distribution for certain time periods [3],[6]. Thus, we divide the day in an arbitrary number of time windows, where each time window wi is determined with its start time ti and end time tiþ1. We assume that the environment-specific parameters are stable during a particular time window. Once the time windows are determined, we calculate the average reliability value pwi for each time window wi:

$$\overline{p_{w_i}} = \frac{1}{|W_i|} \sum_{r \in W_i} p_r,$$

where Wi is the set of records within the time window wi, r is the record from the past invocation sample and pr is user perceived reliability for that invocation. Each particular time window is clustered using K-means clustering algorithm [6] into an appropriate environment condition ei according to its average reliability value pwi as described in the following paragraph. In K-means algorithm, the goal is to partition the data points into K clusters. In case of time window clustering, K is equal to the cardinality number of the environment conditions. For each environment condition cluster, we introduce a centroid value which represents the given cluster e_k. Moreover, for each time window, we introduce a corresponding binary indicator variable. The indicator variable has a value 1 if the associated time window Wi is assigned to the cluster e_k, otherwise it has a value 0. Now, we can define an objective function as follows:

$$J = \sum_{W_i \in W} \sum_{e_k \in E} b_{w_i, e_k} ||\overline{p_{w_i}} - \mu_{e_k}||^2$$

K-means algorithm performs an iterative procedure in which each iteration involves two successive steps corresponding to optimizations. Two-stage optimization is repeated until convergence. In order to improve prediction accuracy, past invocation sample should be updated with newly experienced reliability records and obsolete records should be removed. Past invocation sample update may be performed periodically depending on how dynamically the environment changes. Also, each time past invocation sample gets updated, the space D should be recreated.

V. PERFORMANCE EVALUATION

To evaluate prediction accuracy, we use standard error measure root mean square error (RMSE). It computes a quadratic scoring rule which represents average magnitude of errors: where N is the cardinal number of the prediction set, p_j an existing reliability value in the prediction set, while p_j is the predicted reliability value. Note that RMSE can range from 0 to 1. It is a negatively-oriented score, which means that lower values are better we introduced different user-, service- and environment-specific parameters.

$$RMSE = \sqrt{\frac{\sum_j^N (p_j - \hat{p}_j)^2}{N}},$$

We ensured different service-specific parameters by implementing REST ful services with different computational complexity and by placing services in different geographic locations worldwide. Although a myriad of other parameters influence the computational complexity of services, for practical reasons we chose the amount of memory as a parameter that separates services by their computational complexity. Note that the generality of our experiments is preserved and that any other parameter can be chosen as well. Hence, we created seven different service classes that perform matrix multiplication operations on randomly generated matrices, having each class operate on matrices of different rank as shown in Table 1.To introduce another service-specific parameter we placed 49 web services in seven available Amazon EC regions: Ireland, Virginia, Oregon, California, Singapore, Japan and Brazil, having one service class in each region. Each service was deployed on an Amazon machine image running Microsoft Windows Server 2008, 64-bit, IIS 7 and ASP.NET 3.5. To incorporate user-specific parameters in experiments, we simulated users by placing the amount of 50 instances of loadUI tool [13] in different locations within the cloud. The instances were running as agents in the cloud waiting for the test cases to be delivered and committed. The used tool is an open source tool designed for web services "stress-testing" and it supports creation of various test cases. Finally, we introduced different environment-specific parameters by creating test cases with different load generators defined by the time interval between subsequent invocations. We encumbered services with seven different load levels by altering the time interval . For each particular load, a special test case was created and delivered to all agents in the cloud. During every single test case, each agent sent 150 requests to each deployed service. Based on the number of successful requests against the number of total 150 sent requests, the measured reliability value for that particular agent (user), service and load is computed. Upon the test case completion, we collected measured reliability data from agents and restarted the machines hosting services to recover for the next test case. As part of our experiments, the overall amount of around 2:5 million distinct web service invocations was performed.

5.1 The Impact on Computational Performance:

The evaluation results for computational performance of the prediction in dynamic and static environments are depicted. We use the execution time that is required to compute the predictions as the measure of prediction's computational performance. The figures depict aggregated prediction time for the whole testing set in milliseconds in relation to the data density in the logarithmic scale. All memory-based collaborative filtering approaches induce similar computational complexity. Hence, to depict the evaluation results more clearly, we choose the Hybrid approach [9], [10] as their representative. It is presented both data clustering phase (labeled as CLUS cluster) and prediction phase (labeled as CLUS predict). Similarly, the prediction process in LinReg is performed in two phases: learning phase and prediction phase.

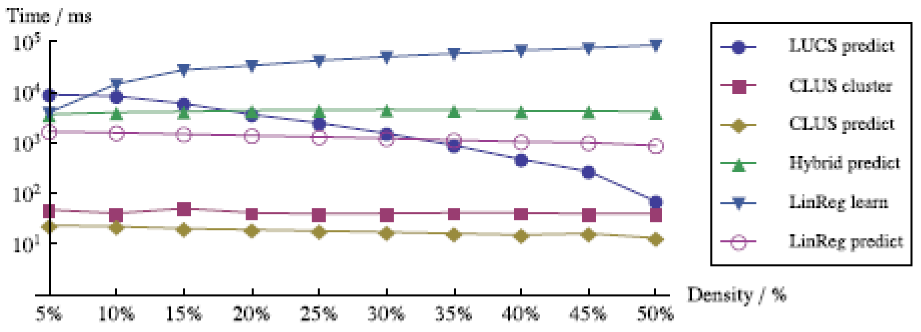


Fig: Prediction time in dynamic environment

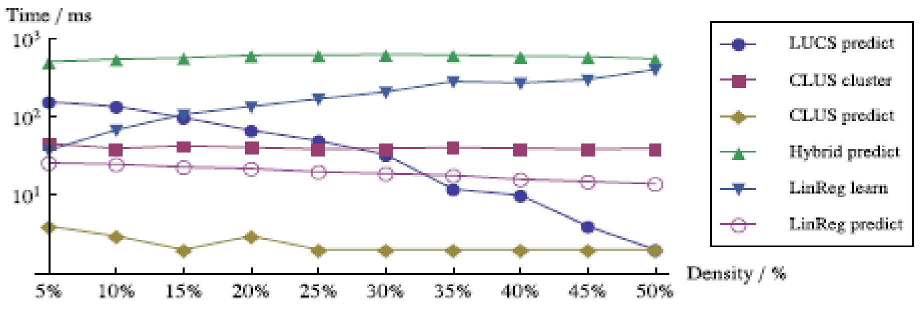
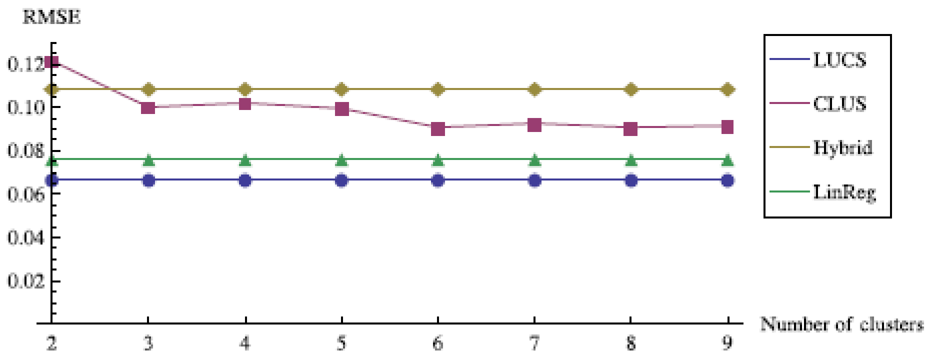
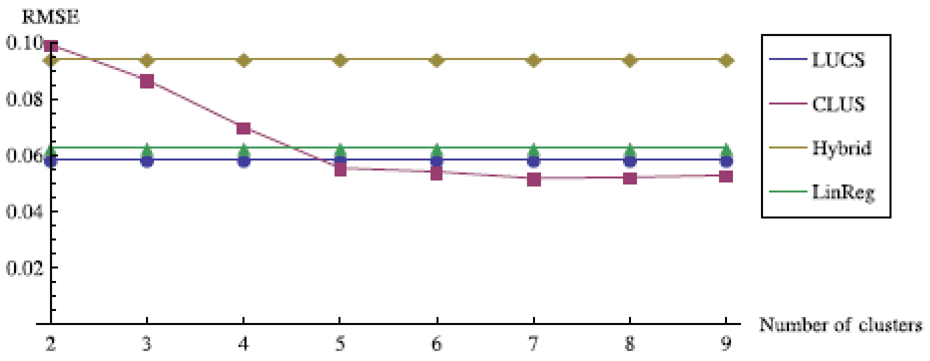


Fig: Prediction time in static environment

CLUS also provides almost constant clustering and prediction time as the data density changes, while LinReg, LUCS and Hybrid manifest similar behaviour like in the dynamic environment.





The evaluation results for prediction accuracy at density of 25 percent are depicted. Expectedly, prediction accuracy of CLUS approach increases as the number of clusters grows. Approaches LUCS, LinReg and Hybrid obtain constant RMSE values of 0:058, 0:062 and 0:094 respectively. In comparison to Hybrid approach, CLUS almost constantly produces more accurate predictions. As the number of clusters grows, CLUS outperforms LUCS and LinReg which compiles to the results presented .

VI. CONCLUSION

The present methods indirectly reflect only user and service-specific parameters of the expectation. On the other hand, we incorporate the environment-specific parameters into the prediction which in turn suggestively reduces the RMSE value. This can specially be observed for CLUS approach at higher data densities. Regarding scalability, it is achieved by grouping similar users and services considering their reliability performance using the K-means clustering algorithm. In this way CLUS reduces the execution time for at least two orders of magnitude when compared to the competing methods. Advantages of these approaches can be found in their tractability which is manifested in a compromise between accuracy and scalability. Increasing the number of clusters in CLUS and complexity of the hypothesis function in LinReg yields more precise predictions at the cost of computational presentation.

REFERENCES

[1] T. Yu, Y. Zhang, and K.-J. Lin, “Efficient algorithms for web services selection with end-to-end qos constraints,” ACM Trans. Web, vol. 1, 2007.
[2] T. H. Tan, E. Andr_e, J. Sun, Y. Liu, J. S. Dong, and M. Chen,“Dynamic synthesis of local time requirement for service composition,” in Proc. Int. Conf. Softw. Eng., 2013, pp. 542–551.
[3] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, “Basic concepts and taxonomy of dependable and secure computing,” IEEE Trans. Dependable Secure Comput., vol. 1, no. 1, pp. 11–33, Jan.–Mar. 2004.
[4] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, “Qos-aware middleware for web services composition,” IEEE Trans. Softw. Eng., vol. 30, no. 5, pp. 311–327, May 2004.
[5] D. Wang and S. T. KISHOR, “Modeling user-perceived reliability based on user behavior graphs,” Int. J. Rel., Qual. Safety Eng., vol. 16, p. 303, 2009.
[6] V. Cortellessa and V. Grassi, “Reliability modeling and analysis of service-oriented architectures,” in Proc. Test Anal. Web Services, 2007, pp. 339–362.
[7] M. R. Lyu, ed., Handbook of Software Reliability Engineering. New York, NY, USA: McGraw-Hill, 1996.
[8] L. Cheung, L. Golubchik, and F. Sha, “A study of web services performance prediction: A client’s perspective,” in Proc. IEEE 19th Int. Symp. Model., Anal. Simul. Comput. Telecommun. Syst., 2011, pp. 75–84.
[9] Z. Zheng and M. R. Lyu, “Collaborative reliability prediction of service-oriented systems,” in Proc. ACM/IEEE Int. Conf. Softw. Eng., 2010, pp. 35–44.
[10] Z. Zheng, H. Ma, M. R. Lyu, and I. King, “Qos-aware web service recommendation by collaborative filtering,” IEEE Trans. Serv. Comput., vol. 4, no. 2, pp. 140–152, Apr.–Jun. 2011.
[11] L. Baresi and S. Guinea, “Event-based multi-level service monitoring,” in Proc. IEEE Int. Conf. Web Services, 2013, pp. 83–90,.
[12] X. Su and T. M. Khoshgoftaar, “A survey of collaborative filtering techniques,” Adv. Artif. Intell., vol. 2009, pp.

4:2–4:2, Jan. 2009.

[13] Y. Wang, W. M. Lively, and D. B. Simmons, "Web software traffic characteristics and failure prediction model selection," J. Comp. Methods Sci. Eng., vol. 9, pp. 23–33, 2009.

[14] Y. Baryshnikov, E. Coffman, G. Pierre, D. Rubenstein, M. Squillante, and T. Yimwadsana, "Predictability of web-server traffic congestion," in Proc. Int. Workshop Web Content Caching Distrib.,2005, pp. 97–103.

[15] M. Andreolini and S. Casolari, "Load prediction models in web based systems," in Proc. Int. Conf. Perform. Eval. Methodolgies Tools, 2006.

[16] Y.-T. Lee and K.-T. Chen, "Is server consolidation beneficial to mmorpg? A case study of world of warcraft," in Proc. IEEE 3rd Int. Conf. Cloud Comput., 2010, pp. 435–442.

[17] M. R. Lyu, "Software reliability engineering: A roadmap," in Proc. 2007 Future Softw. Eng. 2007, pp. 153–170.

[18] L. Cheung, R. Roshandel, N. Medvidovic, and L. Golubchik, "Early prediction of software component reliability," in Proc. Int. Conf. Software Eng., 2008, pp. 111–120.

[19] L. Cheung, I. Krka, L. Golubchik, and N. Medvidovic, "Architecture-level reliability prediction of concurrent systems," in pROC. WOSP/SIPEW Int. Conf. Perform. Eng., 2012, pp. 121–132.

[20] G. Delac, M. Silic, and S. Srbljic, "A reliability improvement method for SOA-based applications," IEEE Trans. Dependable Secure Comput., vol. PP, no. 99, pp. 1–1, 2014.

AUTHORS

[1] Dr V. Goutham is a Professor and Head of the Department of Computer Science and Engineering at TEEGALA KRISHNA REDDY ENGINEERING COLLEGE affiliated to J.N.T.U Hyderabad. He received Ph.d from Acharya Nagarjuna University and M.Tech from Andhra University. He worked for various MNC Companies in Software Testing and Quality as Senior Test Engineer. His research interests are Software Reliability Engineering, software testing, software Metrics, and cloud computing.

[2] Mr.B.Shyla Reddy is working as a Assistant Professor in the Department of Computer Science and Engineering at TEEGALA KRISHNA REDDY ENGINEERING COLLEGE affiliated to J.N.T.U Hyderabad.

.[3] B.Priyanka Department of Computer Science and Engineering at TEEGALA KRISHNA REDDY ENGINEERING COLLEGE affiliated to J.N.T.U Hyderabad.