A Framework for Estimating Dependency based on Eccentricity

V.Kavitha , P.Shanmugapriya

Abstract - Version control systems are employed for change management on the existing code base during development phase and maintenance phase. The changes may or may not affect the existing ones. If the changes affect the existing code they may lead to errors (compile time and run time) or instability of the system. Version control systems employ various strategies to cater the needs of various scenarios but it leads to delays. The agility of the system depends on the updating mechanism and the strategy it deploys while handling multiple users and dependent modules at the similar time. The capability to handle changes by version control systems depends upon the strategy to organize the current modules, update them and serve them to no of users parallely. The objective is to propose a system that is found to be an enhanced version of the current system.

Keywords- Version control systems, change management, instability, multiple users, dependent modules, enhanced version.

I. INTRODUCTION

In order to be competitive in the global market, nowadays organizations needs cooperation and association .. They often put their processes or operations in a dynamic and distributed environment as Web Service (WS) and implemented using Service-Oriented Computing (SOC). With the help of SOC technology number of organizations can interact with each other by providing and invoking its operations as web services .

In as service-based business processes , organizations can be a service provider, service consumer or both. Due to the distributed nature, changes in any of the internal business process and involved services affects other processes which are invoking this service. There are inter dependencies between the internal business process and involved services, therefore, the changes happening in one side may affect the other one in certain degrees and can disseminate in the entire service based business process system like the ripple effect which makes handling changes under change management is a setback.

A small change might have a ripple effect on the other activities which may put down the stability of the entire structure. Making changes to the existing code base is difficult during the following situations: when there are live users to be mitigated, when there are multiple developers working on the same product feature, when there is a highly-coupled product. Nowadays, version control systems are good enough to cater these ambiguities. The rate at which the version control system serves the developers can be improved. The dependencies are estimated by the VCS and other tools by parsing each and every part of the program, thereby building a network based on flow of data to find out most dependent module to begin with. The data flow is manipulated based on stacking of the modules and the one on the top is the most dependent module. A super repository, sub repository maintenance also facilitates the change management. When someone works (Write mode) on the most dependent module: a lock is established to restrict others using the same module, a lock is established to restrict others using the dependent modules, wait queue is maintained to give priorities for critical tasks and future access to the same module and dependent modules. The system takes some time to update the changes and re-determine the network flow. The users must additionally wait for this process to complete.

II. IMPLEMENTATION

Let's look at the total time estimate of the current system under working. Let's label the user write time as 'X' and update time as 'Y'. 'X' and 'Y' are dynamic variables i.e., they need not be same in all cases. 'X' and 'Y' may depend upon the person, type of module and various other factors. Now the next developer in queue must wait 'X+Y'. One must also consider 'interrupts' which will lead to reordering of the queue. Let's label the interrupt factor to contribute to 'Z'. 'Z' is again a dynamic variable. Now the total time the next developer must wait is 'X+Y+Z'.

The goal is to enhance the updating process i.e. 'Y'. The new consideration also proposes a significant reduction in 'X' and 'Z' when looked as a whole. The entire network of the program can be updated easily by considering it as a graph. An algorithm which is used for finding dependencies and constructing semantic graph is consist of the following.

Step 1 : Build dependency tree based on Uniform weights
Step 2 : Form an adjacency matrix based on Dependencies

Revised Manuscript Received on October 18, 2019.

V.Kavitha, Dept. of Computer Science and Engg., Sri Sai Ram Engineering College, Chennai, Research Scholar SCSVMV University , Kanchipuram, Tamil Nadu, INDIA .

Dr.P.Shanmugapriya, Dept. of Computer Science and Engg., SCSVMV University , Kanchipuram, Tamil Nadu, INDIA .

Step 3: Find all possible paths using Floyds Warshall algorithm
Step 4 : Maximum distance in each row of the matrix is the eccentricity of that node.
Step 5: Arrange the modules in ascending order based on eccentricity factor.
Step 6: Display one by one, also respond to changes by repeating the above from first step.

The eccentricity plays a vital role in the same. "Eccentricity is the measure of the farthest distance from that particular node to the terminal nodes". The centre has the least eccentricity. Every graph has 1 or 2 centres which can be easily determined by the eccentricity factor. Once it is determined, the centres are the most dependent modules in the whole program. The modules can be subject to changes one by one from centre to the terminals based on the eccentricity factor. This makes it easy for the programmer and also the updating system. We can expect a significant change in the efficiency towards the positive scale. The updating process of new eccentricities is estimated to be quicker compared to the traditional system. The proposal can be termed as 'X+Y+Z-α' where 'α' is the significant improvement. The improvement varies with various factors such as the system capacity, user efficiency, network efficiency, etc.

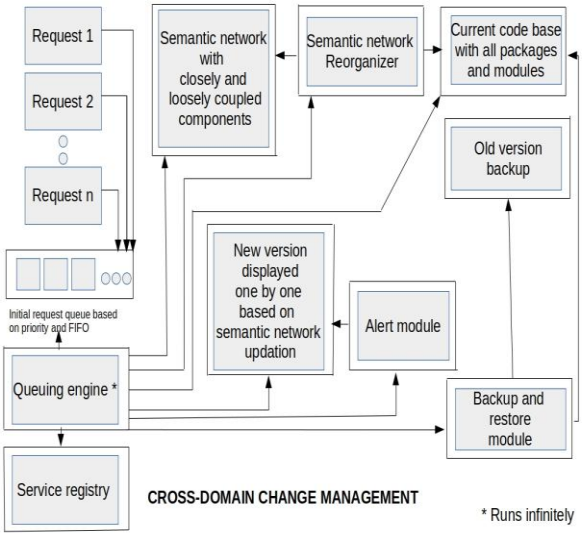

Fig.1. Semantic network is completely built and maintained on eccentricity factor

The requests are put into a priority based queue that works on FIFO strategy with interrupt handling embedded into it. The service registry is referred into if there is a cross domain – change requirement. A semantic network based on eccentricity is built and maintained which displays the modules (one or two) at the centre initially (the ones with the least eccentricity) followed by the others (based on increasing eccentricity). We consider trees here – Reason being that developers don't intentionally create deadlocking modules. If there are any they can be found by the threshold being maintained for each module. A maintenance module reorganizes each and every part of the network based on the dynamic programming approach. A backup and restore module also maintains the previous versions that can be rolled back into. Locks are performed when necessary as done in current Version control systems. Locks are established when there are multiple developers making changes on a module and its dependent modules are also locked in certain situations. The proposed work is verified with an example consists of an environment with set of modules and files. The Fig. 2. shows the dependence modules for each module.

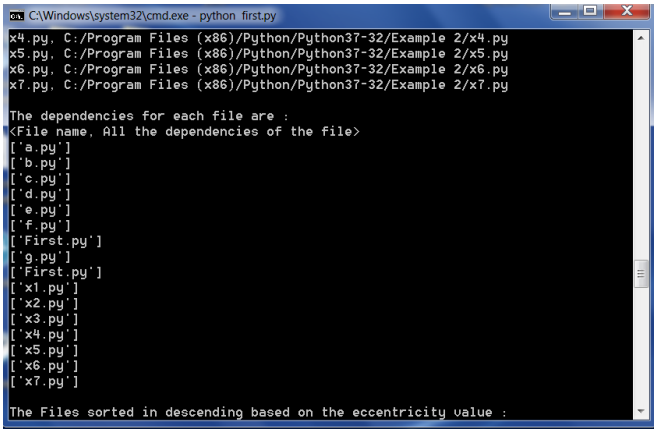

Fig 2. List of dependency Module

The Fig. 3. shows the modules which are dependent along with its eccentric values.

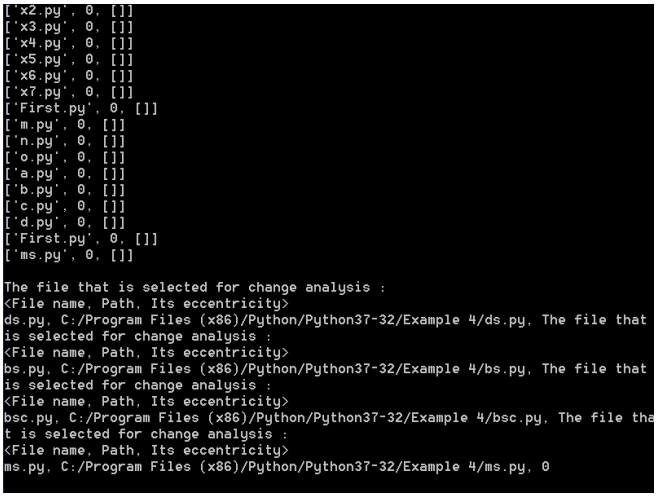

Fig 3 Modules with its eccentric value

The Fig. 4. shows the adjacency matrix of module which is selected for making change along with its eccentric values

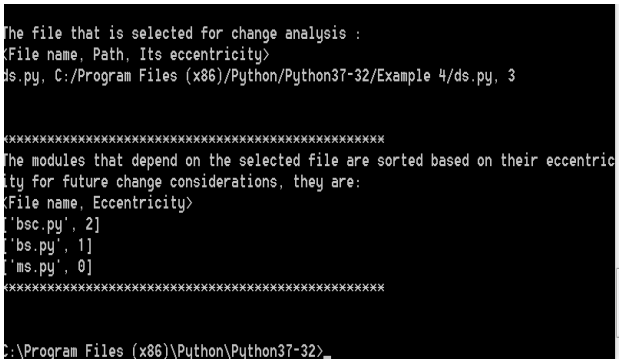

Fig 4. Adjacency matrix of module using eccentric factor

III. FINDINGS & RESULTS

The dynamic programming approach based on eccentricity gives a cutting edge in the entire process thereby improving the process. It is also proposed that if AI is coupled along with this approach the results would be much better as AI reduces human error and reduces 'X+Z' factor as mentioned above thereby improving 'α'. This there by reduces the waiting time for the developers to access the module under lock and dependent modules under lock.

IV. CONCLUSION

The overall objective of the proposed work is to enhance the delay and the smoothness of the cross – domain change management using eccentric values. The proposal doesn't reduce the necessity for lock instead reduces the waiting period.

REFERENCES

1. Rey, S. and Sarrazin, H. How telecoms can get more from Internet Protocol. McKinsey on IT, 2006.
2. N. Crespi. How telecoms can get more from Internet Protocol. Sciences Technology Information projects, 2009.
3. Park, K.H. and Favrel, J. Virtual enterprise-Information system and networking solution. Computers & industrial Engineering **37** (1-2) (1999) 441-444.
4. Madhavji, N.H. Environment evolution: The prism model of changes. IEEE Trans. Software Engineering **18** (5) (1992) 380-392.
5. Cobena, G., Abiteboul, S. and Marian, A. Detecting changes in xml documents. Proceedings of the 18th International Conference on Data Engineering, 2002, 41-52.
6. Aalst, W.M.P.V.D The application of Petri Nets to workflow management. Journal of Circuits, Systems, and Computers **8** (1) (1998) 21-66.
7. Hinz, S., Schmidt, K. and Stahl, C. Transforming BPEL to Petri Nets. Proceedings of the Third International Conference on Business Process Management, 2005.
8. Badouel, E., Llorens, M. and Olivier, J. Modeling Concurrent Systems: Reconfigurable Nets. Proc. Int. Conf. on Parallel and Distributed Processing Techniques and Applications, 2003, 1568-1574.
9. Brambilla, M., Ceri, S., Comai, S. and Tziviskou, C. Exception handling in workflow driven web applications. Proceedings of the 14th international conference on World Wide Web, 2005, 170-179.
10. K. Vijayakumar, Arun C, "Integrated cloud-based risk assessment model for continuous integration", International Journal Reasoning-based Intelligent Systems, Vol. 10, Nos. 3/4, 2018.
11. K. Vijayakumar, S. Suchitra and P. Swathi Shri, "A secured cloud storage auditing with empiricaloutsourcing of key updates", International Journal Reasoning-based Intelligent Systems, *Vol. 11, No. 2, 2019*.
12. G. Indrajith and K.Vijayakumar, "Automatic Mathematical and Chronological Prediction in Smartphone Keyboard" International Journal of Engineering and Computer Science ISSN: 2319-7242Volume 5 Issue 5 May 2016, Page No. 16714-16718.

AUTHORS PROFILE

V.Kavitha, Assoc.Prof., Dept. of computer science and Engineering, Completed her M.E. in CSE and doing her Research in the domain of Service Oriented Architecture. Life member of ISTE and MCSIT. Published papers in various Scopus indexed journals , National and International conferences and Journals.

Dr.P.Shanmugapriya, Assoc.Prof., Dept. of computer science and Engineering, SCSVMV University. Done her research in the filed of Software Engg. Her area of interest is Software Engg. Published papers in various Scopus indexed journals , National and International Conferences and Journals