Robust and Scalable Continuous Integration Framework to Deploy and Maintain nPulse

Monisha P[1], Nirmala Gandhi J[2]

[1] *Student, Department of Computer Science and Engineering, K.S.R. College of Engineering, Anna University, Tiruchengode-637215, Tamilnadu, India.* Email: monishaperiasamy6619@gmail.com

[2] *Assistant Professor, Department of Computer Science and Engineering, K.S.R. College of Engineering, Anna University, Tiruchengode-637215, Tamilnadu, India.* Email: nirmalamuthu@gmail.com

Abstract - Continuous Integration is the most common practice among software developers. It has been around for a while now, but the habits it suggests are far from common practice. Automatic builds, a systematic test suite and binding to the mainline branch every day sound simple at first, but they require a responsible team to implement and persistent care. What starts with improved tooling can be a catalyst for long-lasting change in an organizations shipping culture. Continuous incorporation, distribution and deployment are the software development business practices that enable organizations to regularly and consistently release new features and products. It is important to steadily review and create the approaches, tools, challenges and practices reported for implementing and applying continuous practices. This paper emphases on the continuous integration of enterprise Java application i.e., nPulse, a collaborative framework for continuous integrated delivery based on Jenkins. It covers all the stages of the Software Development Lifecycle starting from managing web containers, auto deploying Web Application Resource, managing database backups, data recovery and developing Application Performance Tools. This platform has the complete suite of tools that need to manage the enterprise application infrastructure.

Keywords - Continuous Integration; Jenkins; Web Application Resource; Java; Tomcat;

.

I. INTRODUCTION

Continuous Delivery and Continuous Integration are related Software Engineering Models used in the Software Industry. They depend on a set of processes and workflows that enable many programmers or developers have right to use the Codebase and submit variations and revisions. Continuous Delivery (CD) is the next step to Continuous Integration (CI). Every modification on the Codebase is instantly deployed to production in CD and often very strict testing and approval mechanism done. Most ultimate processes in CD Involve testing and authorization. As a effect of CI practices, Testing is mostly automated and requires slight to no human intervention. Testing is typically followed by Quality Assurance (QA) which involves approval of the currently performed branch of the Codebase. Perceptual Difference is envisioned to reduce this QA time by supporting in recognizing changes in the UI. This enables quicker and harmless CD.

By adopting to robotic continuous integration, deployment and Delivery practice, the ideas of agile and DevOps can be twisted into practical solutions. Continuous Integration (CI) is an essential section of Agile and DevOps practices as in [1] [8] and [9]. In practice, CI includes a central server which regularly check-ins all the fresh source code modifications as soon as the developers commit them, broadcasting any failures during a build as in [3].

Continuous Delivery is a prolonged version of Agile Development. Continuous Delivery as in [4] takes the notion of the continuous Integration to the next subsequent step. However, automating continuous deployment as in [5] is intricate, time intense and alarming, also it is not classically precise how to go about it. One key resolution to these problems is to just doing industrialize the process of software build, testing, deployment and delivering product reaches the production surroundings.

The main aim of this approach is to increase consistency of automated tests. This will greatly increase rapidity and correctness of user acceptance tests, thus, ensuing in faster deployment rates. Faster release cycles are indispensable that in a market with increasing number of modest agents that are forever on the lookout for missteps that a vendor makes in an attempt to improve market share from their opponent’s loss. We shall focus on CD integrations and other metrics define the efficacy of the proposed tool.

II. OBJECTIVE

nPulse is the mission critical application to its clients. There is an inevitable need to ensure the highest levels of the deliverability and maintainability. Organizations have some basic tools that manages check-ins and build artifacts automatically. It certainly is not enough to cope the demand and its pretty haphazard right now. The current solution does not work for both cloud and onprem deployments – works only for cloud solutions. Whenever Jenkins builds an WAR there is no automated deployment of WARs in Web Container. Operations like Backups, Skimming of logs, Reporting bugs to the support team, cleaning up the disks in cloud servers are done manually.The overall objective of this research is to break down the current delivery model of organizations to solve one problem after another. We need to make sure to automate the processes and the solution works for both cloud and onprem deployments.

Continuous Integration

CI in the above situation become completely necessary. CI obliges that the developers on the developer branch pool their code to the main repository at regular intervals. After integrating this code a variation of build and test automation tasks are done to confirm build reliability. A CI pipeline is typically centrally coordinated with a CI Server. Every modification of codebase stimulates a central build test for that alteration. If permitted the revision is combined with the assembly/production branch. If the tests or the build process failed the revision is marked as a "fail" and the developer is notified.

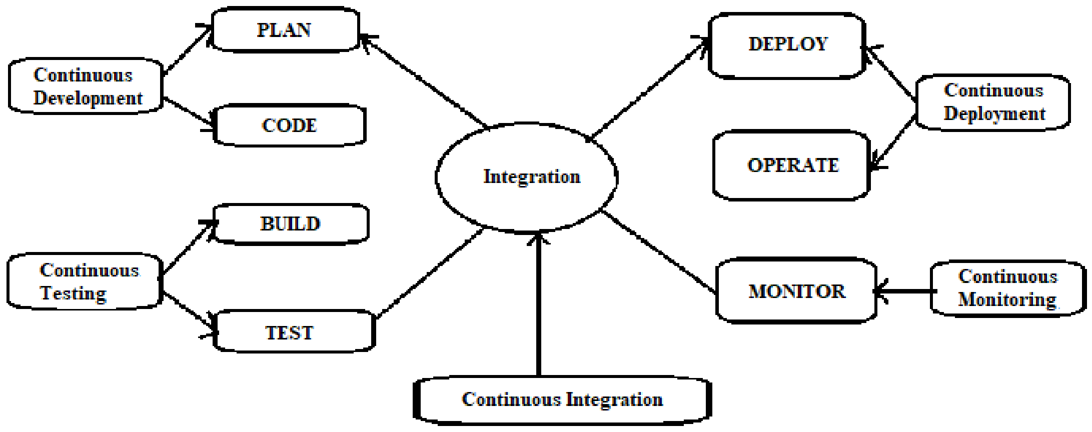


Fig. 1 Process of Continuous Integration

Continuous Delivery

Continuous Delivery as defined by the Agile Alliance is an addition of the Continuous Integration methodology (CI) that tries to shrink the cycle-time (lead time), which is the time occupied for a line of code written in development to be used in the live version of a product which is enclosed to users. CD methods frequently involve the creation of Delivery and Deployment Pipelines which are a sequence of integrations from the point of authoring code to the last deployment to a production surroundings. These Integrations form the base for Continuous Integration (CI). In multi author projects that have a huge number of developers that have right to use to the development branch of the project, it is essential to have integration mechanisms in place such that no single combine can cooperate the state of the production branch.

Build and Test Automation

Build Automations is an essential feature of CD methodology. It denotes to self-contained environments that implement tasks of building from source in a deterministic and steady manner. Developer Local Environments do afford a simple standard for building projects but are distressingly inconsistent across platforms. They thus don't give us a universal picture of build failure with respect to the live deployment. Automation guarantees a build consistency that is not detected on Developer Machines. A build automation agent or server is a significant server that can be On Demand, Scheduled and Triggered.

Testing a build is crucial before deployment. However, a lot of lapsing and unit tests are extremely time consuming and can cause postponements in the lead time estimates of production. This can however be evaded by automating testing by utilizing the boring nature of testing. Unit tests can be written earlier to testing and run in majority by a Test Automation Server. The server is accountable for providing shareholders with data associated with test accomplishment or failure. Most standard TA frameworks provide a modest scripting language to controller the tests themselves and architecting them. Continuous Testing is a subset of TA that is alike to CI and CD and includes running testing tasks as part of the CD pipeline and thus getting instant feedback on the hazards associated with the present build.

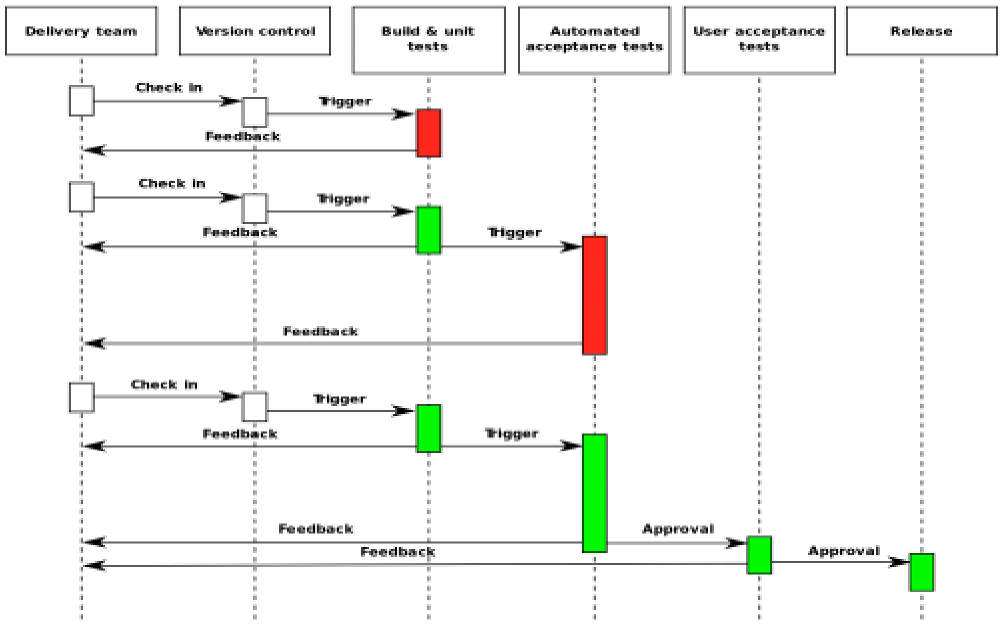


Fig. 2 Flow Diagram of CD Pipeline

Current Tools

The several tools and programming languages used while carrying out the testing is as follows: Subversion, Jenkins, Continuous Integration Server, Java, Hibernate, Google web Toolkit, Shell Scripting, Web container, PostgreSQL. Here are some of the crucial tools and practices.

A source code repository is a place where developers commit and correct code. The source code for this scope of the project is SVN. The source code repository manages whole code that is checked in, so developers don't transcribe over each other's work. Source control has possibly been around for forty years, but it's a major constituent of continuous integration. One of the standard source code repository tools is the Subversion. The build server is an robotics tool that compiles the code in the source code repository into executable code. One Such most popular tools is the Jenkins. Configuration management defines the formation of a server or an environment. For virtual infrastructure the Amazon Web Services and Microsoft Azure are instances of virtual infrastructures. Virtual infrastructures are delivered by cloud vendors that retail infrastructure or platform as a service (PaaS). These infrastructures have APIs to allow you to programmatically produce new machines with conformation management tools.

There are also called private clouds. For example, VMware has the vCloud. Private virtual infrastructures let you to run a cloud on top of the hardware in the data center. Virtual infrastructures joined with automation tools to authorize organizations practicing DevOps with the ability to organize a server without any fingers on the keyboard. If you want to test your brand-new code, you can repeatedly send it to your cloud infrastructure, build the environment and then run all of the tests without human involvement. [7].

Jenkins: Jenkins is a very famous and powerful CI Server that is used by nearly all industry giants to coordinate their software delivery pipeline. Sun Microsystems released the product distinctly as the Jenkins CI Server. Jenkins is written completely in Java and is entirely open source. The Jenkins Automation Server Claims of powerful integrations with almost databases and SCMs such as Git or SVN. As a result of the open source environment of the project Jenkins has a rick plugin system that is continuously modernized and maintained by its contributors. It can be stretched and modified as and when needed by the business. It has a approachable web-based GUI interface to enable on the fly configurations with slight to no code experience. Error reportage in Jenkins is very easier and provides users with inline help.

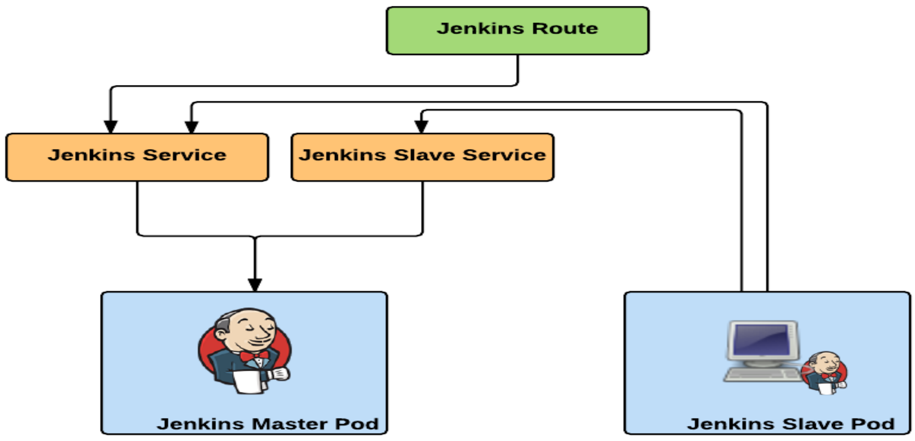


Fig. 3 Jenkins Master Slave Services

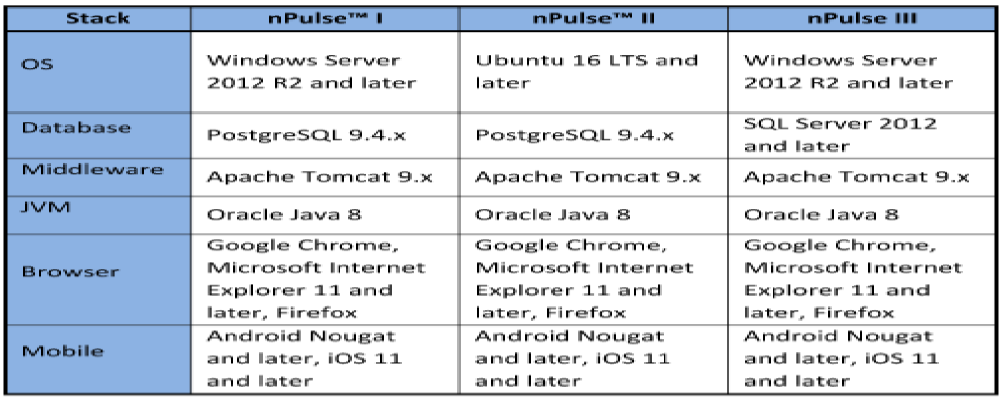
Google Web Toolkit: Google Web Toolkit (GWT) is the toolkit for development which builds and refines complex browser-based applications. GWT is used by several products at Google, including Google AdWords and Orkut. GWT is an open source, totally free, and used by thousands of developers everywhere in the world. It is certified under the Apache License version 2.0.

Being Java based, you can use JAVA IDEs like Eclipse to build a GWT application. Developers can use code to auto-complete management and all types of IDEs. GWT which offers full debugging capability. Developers can be able to debug the client side application just as a Java Application. The goal of GWT is to create tight code, and to let developers to use top notch debugging and development tools. Thus, GWT does not implement Java APIs that are not suitable for browser use, or whose faithful imitation would cause too much bloat.

Ubuntu: The Operating System used for this is project is Ubuntu. Ubuntu is an Open-Source Linux-based Operating System. Both the local machine and the remote servers are Ubuntu. Scripts are written locally and moved to the remote servers using SSH and/or SFTP.

Shell Scripts:A shell script is a list of commands in a computer program that the Unix shell runs which is a command line interpreter. A shell script typically has comments that describe the steps. The different operations achieved by shell scripts are program execution, file manipulation and text printing. Several commands that would be entered manually in a command line interface can be performed automatically using a shell script. This can be done without the user demanding to trigger each command distinctly.

Many complex applications can be written in shell scripts using these features. But there is a trick i.e. shell script languages don't support classes, threading etc. that are only found in cultured programming languages such as arrays, variables, comments etc.



Stack	nPulse™ I	nPulse™ II	nPulse III
OS	Windows Server 2012 R2 and later	Ubuntu 16 LTS and later	Windows Server 2012 R2 and later
Database	PostgreSQL 9.4.x	PostgreSQL 9.4.x	SQL Server 2012 and later
Middleware	Apache Tomcat 9.x	Apache Tomcat 9.x	Apache Tomcat 9.x
JVM	Oracle Java 8	Oracle Java 8	Oracle Java 8
Browser	Google Chrome, Microsoft Internet Explorer 11 and later, Firefox	Google Chrome, Microsoft Internet Explorer 11 and later, Firefox	Google Chrome, Microsoft Internet Explorer 11 and later, Firefox
Mobile	Android Nougat and later, iOS 11 and later	Android Nougat and later, iOS 11 and later	Android Nougat and later, iOS 11 and later

Fig. 4 Stack Diagram of nPulse

Web Applications

Web applications are presently on the rise and certainly the most popular form of applications. They consume the browser as a deployment vector and users view and interrelate with these applications via the browser itself without the need of downloading an external desktop application which is usually huger. Web apps are known for their increased availability and cross-platform nature. Web applications cut over this clutter by restructuring the deployment firmly to web browsers. Their user acceptance is usually led by interfaces and front end interfaces. Most famous websites spend a lot of time and investment on designing and changing their interfaces so as to fascinate more users and/or customers. The User Interface Design and Development is a key constituent in the software development stage. It agrees the layout, the design and the interactivity of the web application. Some of the Common tools used to accomplish this are the HTML which is used for layout specification, CSS for designing and styling the web contents and JavaScript for enabling the interactivity of the web pages.

Development

The code is developed in the integrated developmemt environment such as Eclipse and it is commited to a common source code repository. The Eclipse platform which delivers the base for the Eclipse IDE is composed of plug-ins and is intended to be extensible using additional plug-ins. Being developed using Java, this platform can be used to develop standard customer applications, incorporated development environments and other tools. Eclipse can be used as an IDE for any programming language for which a plug-in is obtainable. A Subversion repository is a collection of files and directories, hustled together in a distinct database that also records a complete history of all the alterations that have ever been made to these files. Theoretically, it is similar to a folder or directory on your computer that may comprise a collection of various, but related, files and directories. A SVN repository is used to store all the files and directories that make up a single project, possibly a collection of interconnected projects. This centralised source code repository may be subversion or Git based on the project requirement. This repository has the complete code which comprises of each changes and features developed by the developers.

Testing

Any web application or applications in general needs to be tested before it gets deployed. Some of the common testing done are Functional Testing which entails that all the hyperlinks, forms, buttons, database connections are working properly in the web application. Interface Testing that ensures the interface between the web server or the backend is working in tandem with the front end or the view. Compatibility Testing makes sure the app performs consistently across browsers ad platforms. Performance Testing which is divided into load ad stress testing. Load testing involves assessing if the application can handle large quantities of user traffic and analysing its limits. But in Jenkins we can perform the automated tests. Any testing plugin can be installed as per the requirement of the code through Manage Plugins and the Jenkins needs to be restarted after appropriate changes are made to configurations. It checks the shared repository at periodic intervals and every check-in is pulled and then the builds application performs unit and integrated tests and packages the application. If this process fails it alerts the developers. So that testing is done at fast pace and in the effective manner. On the other hand when the build is done correctly it gives the build success message to all the developers which indicates that it ready for the deployment.

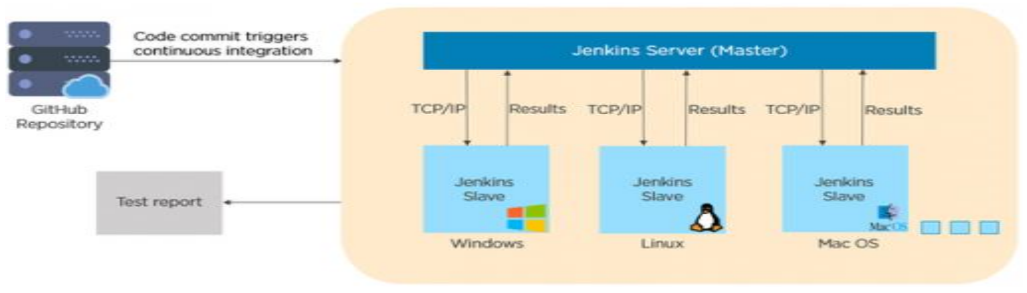


Fig. 5 Jenkins on different Platforms

III. PROPOSED WORK

When all the development, Testing is done perfectly without any faults, next phase is the deployment. This phase can be done in many ways. One such way is the automatic deployment which is part of the Jenkins build job. Using a combination of Java configuration, Ant build targets, shell scripts, and Jenkins scripts and plug-ins, we built a framework that automates the deployments of build artifacts so that the QA team can continue without any human intervention.

After the Jenkins builds a Web Application Resource, it executes the Execute Shell section in which the scripts are embedded. This script configures the Web Application Resource according to the clients requirements. There are virtual infrastructures in which these modifications can be done. A plugin called Deploy to container Plugin is installed which contains configurations to which the application needs to be deployed.

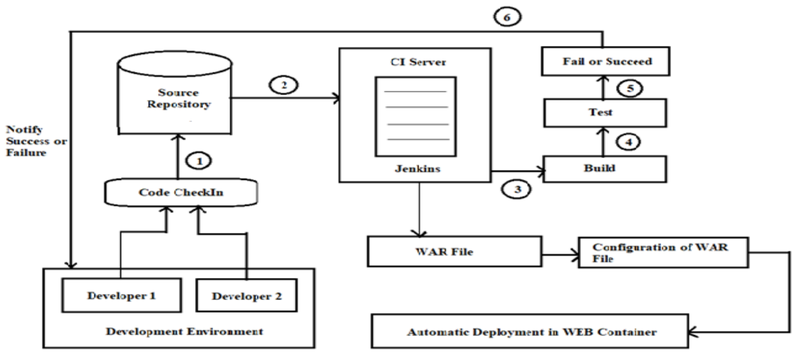


Fig. 6 Architecture of deployment pipeline

IV. CONCLUSION

The main Objective of this work is towards automated deployment of Web Application Resource in a web container through the shell scripts and the hardening of the web servers to protect it from unauthorized users. This type of deployment is efficient and consumes less time compared to manual deployments leaving the developers and the operations team to focus on other aspects of development of organization and these are readily available for the Sales and Operations Planning team. This makes sure the web application is Secure Sockets Layer complaint and all other vulnerabilities are plugged. Security aspect of the delivery is also taken into concern. The Virtual machine deployments is also explored here. The Continuous Integration Server Jenkins takes care of Automated build, Testing and the Deployment.

REFERENCES

[1] T. Amanatidis, N. Mittas, A. Chatzigeorgiou, A. Ampatzoglou and L. Angelis, "The Developer's Dilemma: Factors Affecting the Decision to Repay Code Debt," 2018 IEEE/ACM International Conference on Technical Debt (TechDebt), Gothenburg, 2018, pp. 6266.

[2] N. A. Ernst, S. Bellomo, I. Ozkaya and R. L. Nord, "What to Fix? Distinguishing between Design and Non-design Rules in Automated Tools," 2017 IEEE International Conference on Software Architecture (ICSA), Gothenburg, 2017, pp. 165-168.

[3]J. Nirmala Gandhi, P. Sasikumar, "Image Watermark Detector using Gauss Hermite Expansion in Wavelet Domain," International Journal of Core Research in Communication Engineering, 2015, DOI & ISSN No. 2349-7742.

[4] B. Barta, G. Manz, I. Siket and R. Ferenc, "Challenges of SonarQube Plug-In Maintenance," 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER), Hangzhou, China, 2019, pp. 574-578.

[5]M. Shahin, M. Ali Babar and L. Zhu, "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices," in IEEE Access, vol. 5, pp. 3909-3943, 2017.

[6]C. Vassallo, F. Palomba and H. C. Gall, "Continuous Refactoring in CI: A Preliminary Study on the Perceived Advantages and Barriers," 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), Madrid, 2018, pp. 564-568.

[7] Q. Cao, Y. Qiao and Z. Lyu, "Machine learning to detect anomalies in web log analysis", 2017 3rd IEEE International Conference on Computer and Communications (ICCC), Chengdu, 2017, pp. 519-523.

[8]M. V. Kosti, A. Ampatzoglou, A. Chatzigeorgiou, G. Pallas, I. Stamelos and L. Angelis, "Technical Debt Principal Assessment Through Structural Metrics," 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Vienna, 2017, pp. 329-333.

[9] A. Martini, "AnaConDebt: A Tool to Assess and Track Technical Debt," 2018 IEEE/ACM International Conference on Technical Debt (TechDebt), Gothenburg, 2018, pp. 55-56.

[10]L. Wu and M. Li, "Applying the CG-logistic Regression Method to Predict the Customer Churn Problem," 2018 5th International Conference on Industrial Economics System and Industrial Security Engineering (IEIS), Toronto, ON, 2018, pp. 1-5.

[11] A. Shapochka and B. Omelayenko, "Practical Technical Debt Discovery by Matching Patterns in Assessment Graph," 2016 IEEE 8th International Workshop on Managing Technical Debt (MTD), Raleigh, NC, 2016, pp. 32-35.

[12] H. Yang, "Research on Cost Decision of SpecializedAutomobile Manufacturing Enterprise Based on the Theory of Decision Tree," 2010 International Conference on Digital Manufacturing & Automation, Changsha, 2010, pp. 198-203.

[13]H. Xie and F. Shang, "The study of methods for postpruning decision trees based on comprehensive evaluation standard," 2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), Xiamen, 2014, pp. 903-908.

[14] Manish Virmani (2015) "Understanding DevOps& Bridging The Gap From Continuous Integration To Continuous Delivery", Innovative Computing Technology (INTECH), IEEE, pg. 78 – 82.

[15]Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., &Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation computer systems, 25(6), 599-616.