Test Orchestration

A framework for Continuous Integration and Continuous Deployment

Nikhil Rathod
Department of Computer Science and Engineering
Walchand College of Engineering
Sangli, India
rathod0707@gmail.com,
nikhil.rathod@walchandsangli.ac.in

Anil Surve
Department of Computer Science and Engineering
Walchand College of Engineering
Sangli, India
anil.surve@walchandsangli.ac.in

***Abstract*—Theenterprises follow Agile Software Development methodology to because business requirements changes frequently. InAgile Software Developmentmethodology, it is essential to continuously integrate the component into a main trunk of a project to test the new component of the system. Then test all the component of the project; this happens frequently so it needs to streamlineprocesses to orchestrate the tests. So it is difficult to manage the software development life cycle for those changes and maintain the software code quality. To maintain the product quality it is essential to integrate the product component and need to deploy a product on pre-production environment and test the product. Hence the need for Continuous Integration and Continuous Delivery process for software product.The popularization ofDevOps, and cloud computing has revolutionized the software delivery process- making it faster and affordable for business to release their software continuously. Hence Enterprises need for reliable and predictable delivery process of software. The objective of the paper is to design an effective framework for automated testing and deployment to help to automate the code analysis, test selection, test scheduling, environment provisioning, test execution,results analysis and deployment pipeline. Test orchestration framework typically very complicated to develop such pipeline to make software reliable, and bug free. For environment provisioning can be provided through virtualization and cloud computing.**

Keywords—ContinuousIntegration (CI); Continious Deployment (CD);DevOps;Agile Software Development; Test Oracle; Test Orchestration; Cloud Computing; Delivery Pipeline;

I. INTRODUCTION

The biggest business challenge facing all enterprises today is the way to convey more features to their clients. To develop such product and maintaining the software quality involves pain, costs, constraints, and external reviews. So Enterprises can no longer afford unpredictable, lengthy and ineffective release processes that barely support one update every couple of months [1]. So, enterprises are accelerating and streamlining their delivery pipeline to stay "ahead of the game" and succeed in today's competitive marketplace.

The first step towards achieving streamlined release process is to provide integrated insight into the testing tracks – a single, shared panel for the entire development, QA and release team [2]. This panel enables all manual and automated activities to be tracked and executed [1]. Once the all tasks are automated and orchestrated then they are able to identify and prevent bottlenecks in advance. In addition, they can full audit trail that tracks the evolution of releases, envision the degree of deviation from the original plan, and quickly highlights unpredictable projects [1]. Any time the software can consider to for pipeline orchestration should promote process and data analysis, and provide reporting, allowingthe team to track the effectiveness of their efforts [3].

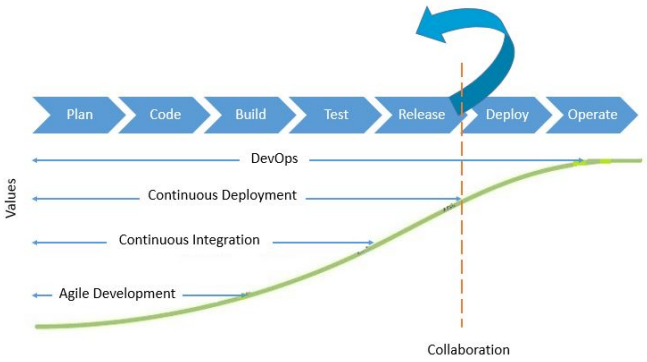

Figure Collabration Diagram

Today's enterprise, most of the software uses manual acceptance testing to verify that a piece of software productconforms to its functional and non-functional requirements [4]. A few automated tests do exist, but they are often poorly maintained and out-of-date, and require supplementing with extensive manual testing [1]. This results in defect detection much later in the product life cycle or they end up getting discovered by end users. This invariably leads to delays in product delivery exacerbated by compromised quality.

II. BACKGROUND

An agile software development process for clarity, early customer feedback and tiny delivery cycles. The development is test driven, and all code commits are version controlled and activate automated builds [1,5]. Continuous integration is a software development practice in which secluded changes are instantlytested and reported on when the code is committed to a large code repository. Each change is verified by an automated build, test to detect an integration error [4]. The goal of CI is to provide a speedy reply so that if a defect is introduced into the code repository, it can be identified and rectified as soon as

978-1-4799-6272-3/15/$31.00(c)2015 IEEE

possible. Continuous integration software frameworks can be used to automate the build and testing [5]. There are many continuous integration framework are freely available for e.g. Jenkins, CruiseControl.CI approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly [6].

In the software industry, achieving software quality involves pain, costs, constraints, and external reviews. Fortunately, with the agile movement, software quality and higher productivity can both be achieved [3, 7]. We can categorize software quality into two major parts such as external quality and internal quality. External quality investigates at how well the software productfulfills its functional requirements. Internal quality investigates at how well the software product is designed to make it maintainable and capable of incorporating new changes [3]. Industry statistics show that on average, eighty percent of the cost of a software productis spent on the maintenance; therefore internal quality is a major key component of the cost of the software. That is the reasons for managing the code quality of the software product has become a major pertain for any enterprise that is elaborate in building software product [8]. Normal approaches to managing code quality propose to test code timely, mainly at the end of a development phase of the software development life cycle. Therefore, there is an urgent need for a new approach that clearly gives ownership of code quality to the developer one that special importance quality throughout the development process and has a quick feedback to ensure fast revolution of quality problems.

III. DESIGN

One approach to solve this problem is to automate all phases of test execution, namely, Continuous Run phase, Continuous Code Analysis Phase and Report Generation [9]. Such Continuous Integration enables developers, testers, build and operations personnel to work together effectively by using a deployment pipeline. A deployment or engineering pipeline automates most of the phases of the software development and deployment life cycle.

Continuous delivery automation generates the bridge between development and operations [1]. It firstly from the entrance conditions and it process by which deployable builds are deployed into pre-production (virtualized) environments [6]. Further, release automation process establishes a control framework, ensuring audit and control, security and notification management.It follows the testing that happens during CI and pushes changes to a production system. This makes sure a version of code is accessible at all times.

Continuous integration and continuous deployment have evolved, but there is no such framework which has both. Automation framework is a set of assumptions, concepts and tools that provide support for automated software build [10], testing and release the software product through regression testing, and test result analysis and continuous code quality check [11]. So the idea is that Design an efficient framework which will automate the CI to CD to achieve the early release of the software.

There are a few steps in the software development life cycle. Code is developed, builds,tested, and integrate to a dev-environment. Once the code is committed by the developer, it is integrated into a productcode repository, where it must interact with other components that may be developed by other developers [12]. Before the code is committed to code repo, few safety steps should be taken to ensure that new developed component doesn't impact the code of other components within the software product.

This framework uses a build-deploy-test workflow on the target platform to deploy and test software application when itruns a build. This lets schedule and trigger the build, test, deployment, and report of software product with one build process [11]. Build-deploy-test-report workflows work with test orchestration to deploy software product to a targetsystem environment and run tests on them as a piece of the build process [5]. It can also use well defined workflows to generate and restore snapshots that create a clean target environment before build, run tests, report and save the state of target environment when a test fails. This makes sure that each test is not having an effect on the test environment from few last test runs. In addition, it makes certain that testers can accurately recreate that state of a test environment when they reproduce bugs.

There are various phases of software development life cycle i.e. Design, development, testing, deployment and support, but majorly automation needawhileto build, test, report and deploy the software product [5]. So while designing continuous automation of framework i.e. Test Orchestration framework, major components are in building automation, test automation, report and deployment automation.

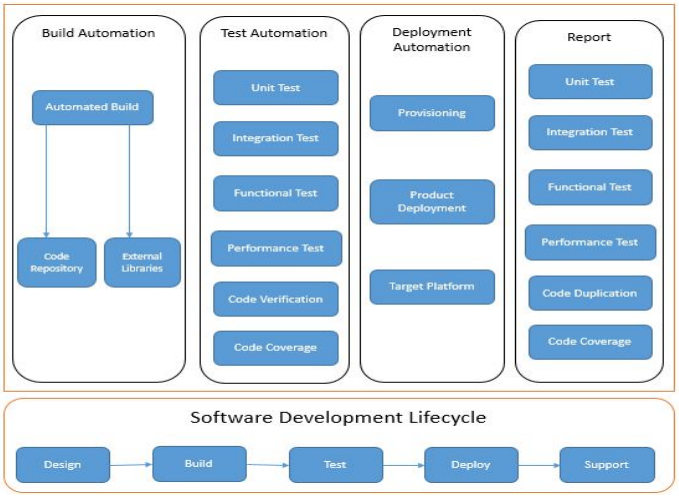

Figure 1 Test Orchestration Architecture Diagram

A. Build Automation

In software development, continuous integration (CI) actualizes nonstop methodologies of applying quality control - little bits of exertion, connected oftentimes. Consistent combination intends to enhance the nature of programming, and to lessen the time taken to convey it, by supplanting the customary practice of applying quality control in the wake of finishing all advancement.

A continuous integration (CI) tool is no longer simply something that is “nice to have,” during project development. As somebody who has invested more of a chance than I want to examine wading through records and verifying references, traceability, record form and configuration yields are appropriately recorded in a Design History File (DHF), I would like to make the benefit of utilizing CI to mechanize such repetitive and lapse inclined difficult work clear [13].

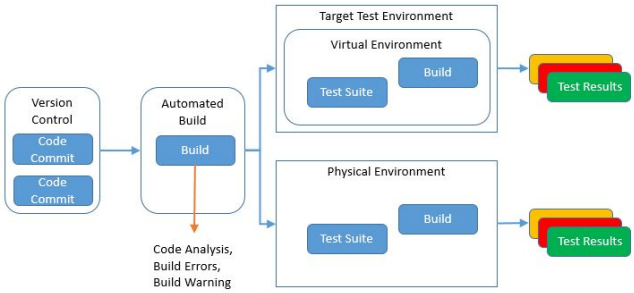

Figure 2Build Automation

The continuous integration is an absolute necessity;itmeans that both the CI tool and the process are needed. A CI tool takes an attempt sometimes weak attemptof the developer to make large amounts of documentation consistently referable and forces the automated system to do what it does best. The use of a CI tool is not simply an abstruse practice for those who are part of its enterprise [11]. As continuous integration is something that good software development enterprises have always attempted, but have too often failed to utilize software product to ease the process.

Continuous Integration mention to both the compilation and build of a software product and the continuous testing, releasing and quality control [11]. This means that throughout the product development, at every stage, the development team will have a build available with at least incomplete documentation, report and testing included [3]. The CI automated build is used to perform certain specific tasks at certain times. In general, this simply means that builds are carried out onthe target system environment that closely matches the actual system production environment. In addition, a CI environment should be used to provide a statistical reply on building performance, tests, and enterprise of a version control system and token systems [5]. In a development environment, the team may use a version control tool, i.e. Subversion to link to tokens. In this way, any CI build will be associatedwith a specific file change set, thereby providing association to Issues, Requirements.

A software development team should attempt to perform CI builds frequently adequate that additional version control update happens between commit and build, and such that no errors can become apparent without developers noticing them and correcting them promptly [6]. This means that for a product that is under development, it should arrange that a checking triggers a build, teston developer’s commit. Likewise, it is generally a good practice for the developer committing a change set of code to verify that own change set does not break the other component of the product.

Build automation is the approach used during software development where the component source code is compiled into a low level language with diversebuild script. The build automation process has become common practice with the evolution of software development practices [7]. As software component has evolved from procedural structured programming to object oriented programming, distributed modules, it's become ananalytical dependency in the management of these products. This can be done through various varieties of tools like Ant, Maven for automatically build of their software product.

Build automation is a best practice for generating code as it bringsthe potential to securebetter quality softwareproduct. The automation of unit testing is the first reason for this enhancement [1]. By forcing the execution of automated testingearlier to integratingthecomponents within the compiled code, the software product is likely to have less errors during the software deployment cycle.Building automation we need to maintain a code repository and external libraries.

Another key benefit of building automation is the ability to track impacts onother components orthe latest previousbuilds [8]. A task within the automated build includes generating a revision number, version number of the codeof version subversion control tools [6]. This version number allows the test orchestrator and testers of the software product as a reference point and when a new error or failure was introduced into the target environment.Build automation has formed, the process easy for compiling software productand made easy to manage within software development teams [9]. Due to a buildautomation process, there is increased productivity and higher quality software product by enabling reusable component for all software builds.

B. Testing Automation

Test automation is the use of testing frameworks to test software product to command the test execution and the comparison of actual outcomes with expected outcomes.Some software testing tasks, such as vast low-level interface, regression testing, can be strenuous and time consuming to do manually. In addition, a manual approach might not always be effective in locating certain defects [12]. Once automated tests have been matured, they can be run immediately and frequently. This can be a cost effective method for fullregression testing of software component or product that have a lasting maintenance life. Even minor patches over the lifetime of the product can cause existing features to break, which were working at an earlier point in time [12].

An emerging trend in software testing life cycle is the use of testing frameworks such as the xUnit frameworks i.e. JUnit and NUnit that allow the bringing of unit tests to determine whether various segments of the code are acting as expected under few circumstances [12]. Test suits describe test cases that need to be run on the component to verify that the component runs as expected [1].

Code driven test automation is a key feature of agile software development, where it is known as test-driven development (TDD) [12]. Unit tests are written to define the component before the code is written. However, these unit tests

evolve and are extended as coding growth, issues are identified and the code is subject to refactoring [5, 12]. Only when all the tests for all they expect features pass is the code considered complete. Protector argue that it produces software component that is both more reliable code that means quality and less costly than code that is tested by manual exploration [12].

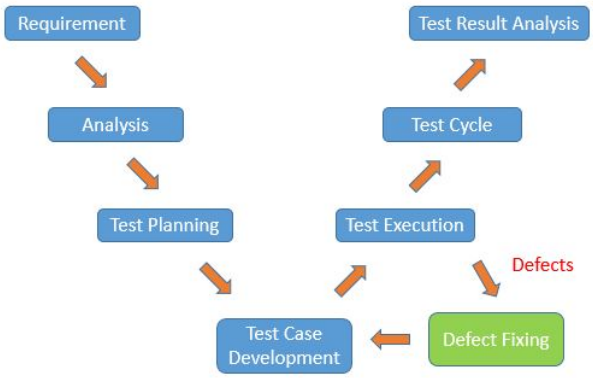

Figure 3Testing Life Cycle

It is considered more reliable because the code coverage is better, and because it is run continuously during software development rather than once at the end of software development life cycle [12]. The developer discovers defects rapidly upon making a change, when it is least expensive to fix. Finally, code refactoring is safer; transforming the code into a simpler form with less code duplication, but equivalent behavior, is much less likely to introduce new defects [12].

C. Reporting

Test results analysis plays a very important role in the software development life cycle. The test process itself is crucial to the success of new software products. It is only through efficient system testing that the quality and safety of a software product can be guaranteed [2, 8]. For software businesses, there is no excuse for a poor quality. This is the reason why enterprises use several testing methods – to perfect their product before releasing the product.

However, it must be noted that the test process does not end with just system testing [4]. A common mistake is omittingthe analysis of test results. While the tests are run for finding errors or bug in the product or component, it is the analysis that transcribes that same error [8]. Without the interpretation, the testing governess would be useless. To further explain the reason why failure to get the analysis is a mistake, let us take the software product interface test.

Though there are many testing approaches available, developers have considered a software interface test, unit test, code quality test integration test, regression test and performance testing are the most important system test to ensure the quality of a program [2]. The interface is composed of sets of commands, dashboards, and other features that grant communication between the user and underlying hardware. According to the developers, the very advantage of interfacetesting is in the fact that it is attached on the feedback of end users themselves. There are various test results and log is generated during the testing process [8]. Each of them is analyzed and various test results are generated. Some of these aspects include functionality, being user-friendly, and the performance of the new software product or component. It is not necessary to communicate with the user. At the end of the testing period, the test orchestration system expected to endorse the feedback made by the system to the developer [4, 13]. Here analysis starts to play a role. The report must be interpreted in such a way that the error in the program should be given an appropriate solution.

In the example of software testing, developers would definitely need analysis of the test result; itcould make the corresponding revisions based on the feedback of the test orchestration system [13]. Remember that the aim of any developer is to perfect the new software product. This can only be done with comprehensive, regression testing processes. Performance test results analysis report may have thousands of charts and reports showing root cause to performance bottlenecks and remission [8, 13]. It may records every transaction, facts, step, and monitored resource timing value to a Results and Asset Repository. It can build its own asset repository to store the logs and results. The asset repository contains a catalogue of scenarios (scenario, log, test metadata, use cases, steps, and test model and test results) to store the results of executions (executions, facts, transaction results, and steps times) [2, 7]. Likewise, enterprises are very keen on improving the product quality. This is possible because their earnings from the software product would rely so much on its potential. Thus, an enterprisewould need software test results analysis to be able to ensure quality and guarantee good returns.

D. Deployment Automation

Continuous Deployment (CD) is closely related to Continuous Integration (CI) and refers to the release the software product into production that passes the automated tests. Essentially, “it is the practice of releasing every build component to users”. By adopting both CIand CD, test orchestration not only reduce risks and catch defects quickly, but also move rapidly software development [9]. With low-risk releases, the enterprise or development team can rapidly adapt to functional requirements and user needs. This allows for greater association between continuous integration, DevOps and continues delivery, feeding real change in the enterprise, and turningthe release process into a faster delivery pipeline [9].

The technical track of releases rarely proceeds absolutely in most enterprises. Many tasks, especially in product deployment, test environment provisioning, and configuration are more made automatically [1, 9]. Those initiatives usually involve introducing Continuous Integration and Continuous Deployment Automation, a measure of test automation, and on-demand provisioning and configuration of virtualized test environments [10]. The goal then is to integrate these to create an end to end pipeline that can orchestrate the appropriate builds, environment provisioning, testing, and test result analysis and deployment steps [10].These steps can be

associated using something as simple as chained processes or builds in Continuous Integration tool, or you can make use of one of a number of devoted automation orchestrators [10].

Since a streamlined, fully automated deployment process has remained end goal, it is critical that the test orchestration framework supports fully automated CD pipelines and DevOpsstyle automated platforms. To start simulating the deployment process, and ensure that all steps are automated [10].

IV. SUMMERY

Test Orchestration is a unique framework for Continuous Integration (CI) and Continuous Deployment (CD) in enterprise use cases. It can improve the visibility and reporting, allowing quick recommendations based on previous test build and test result analysisIt also could have a method for automatically and reliably checking the outputs of the software Under Tests (SUT) to see if the results reveal any failure. This method evaluates executable log file analyzer that can be applied directly to the test result checking.

As Test Orchestration is streamlined pipeline will be effectively operatedin all phases of the software development life cycle to evaluate the future of the software product. To achieve regression mode risk free and effortless continuous delivery process to reduce the period of delayed delivery cost overrun& poor quality delivered software.

We are now in the process of applying this work to a commercial system with industrial partners and would like to extend our work further in that direction as well.

ACKNOWLEDGMENT

We express our sincere thanks to all the authors, whose papers in the area of continuous integration, continuous deployment, DevOps and cloud computing are published in various conference proceedings and journals, and to all authors and organizations of referring websites.

REFERENCES

[1] Kruse, Peter M., et al. "Logging to Facilitate Combinatorial System Testing. *"Future Internet Testing*. Springer International Publishing, 2014. 48-58.

[2] Andrews, James H., And Yingjun Zhang. "General test result checking with log file analysis." Software Engineering, IEEE Transactions on 29.7 (2003): 634-648.

[3] XpoLog, "Augmented Search for Software Testing for Testers, Developers, and QA Managers New frontier in big log data analysis and application intelligence" White Paper May 2013.

[4] Aharon, Michal, et al. "One graph is worth a thousand logs: Uncovering hidden structures in massive system event logs." Machine Learning and Knowledge Discovery in Databases. Springer, Berlin Heidelberg, 2009. 227-243

[5] Cinque, Marcello, Domenico Cotroneo, and Antonio Pecchia. "Event logs for the analysis of software failures: A rule-based approach." Software Engineering, IEEE Transactions on 39.6 (2013): 806-821.

[6] Fronza, Ilenia, et al. "Failure prediction based on log files using Random Indexing and Support Vector Machines." Journal of Systems and Software 86.1 (2013): 2-11.

[7] Liu, Chen, et al. "R2Fix: Automatically generating bug fixes from bug reports. "Software Testing, Verification and Validation (ICST), IEEE Sixth International Conference on. IEEE, 2013.

[8] Weigert, Stefan, Matti Hiltunen, and Christof Fetzer. "Mining large distributed log data in near real time." Managing Large-scale Systems via the Analysis of System Logs and the Application of Machine Learning Techniques. ACM, 2011.

[9] Akerele, Olumide, Muthu Ramachandran, and Mark Dixon. "System Dynamics Modelling of Agile Continuous Delivery Process." Agile Conference (AGILE), 2013. IEEE, 2013.

[10] DevOps (2014, November 4) [Online] Available:http://devops.com/blogs/improve-orchestration-deliver-features-customers-fasterl

[11] Continuous Integration (2011, Jun 23) [Online] Available:http://matthewrupert.net/2011/06/23/continuous-integration-on-software-medical-device-projects-part-1/

[12] Test Automation (2014, November 24) [Online] Available:http://en.wikipedia.org/wiki/Test_automation

[13] Importance of Test Result Analysis (2009, May 7) [Online] Available:http://www.testingexcellence.com/importance-of-software-test-results-analysis/