CONTINUOUS DEVELOPMENT & INTEGRATION OF MODULES WITH AUTOMATED TESTING IN AGILE S/W DEVELOPMENT METHOD

[1]Sohail Hussain, [2]Dr. Mohan Aradhya,
[1]Student, [2]Assistant Professor,
[1]Department of Master of Computer Applications,
[1]RV College of Engineering®, Bengaluru, India.

Abstract: Continuous Development & Integration is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early. By integrating regularly, you can detect errors quickly, and locate them easily. It is absolutely necessary to automate tests on agile projects as the number of test cases will continue to grow with each successive sprint. The relationship between customers and suppliers remains a challenge in agile software development. The transition from a focus on agile methods on team level with emphasis on team performance is illustrated by the focus on pair programming and test first development, to a broader organizational understanding where more focus is put on value of the developed product. The main reason for adopting agile methods are to increase productivity, increase product and service quality and to reduce development cycle times and delivery time to market.

IndexTerms **-Selenium, Continuous Integration, Regression Suite, Agile Methodology, Automated Testing.**

I. INTRODUCTION

Regression suite were traditionally performed in production to ensure that newly released features and updates didn't compromise any existing functionality. It can help to ensure that units of code don't break the build when merged into the larger trunks. Wherever you need validation when one component is merged with another, regression tests can help to ensure that everything works as expected. Regression testing plays a vital role in ensuring that the product remains operational when a new feature is added with each software update. As each change carries a risk of introducing new regressions in the next release, frequent regression testing should be done to ensure that the bugs are caught quickly. Regression suite ensure that code that was previously functioning correctly does not regress when changes are made. Using a comprehensive suite of unit level regression tests provides a safety net, making sure that code changes do not break existing functionality. The test suite can be generated according to a specified structural code coverage target. These autogenerated tests can then be used as a baseline safety net to identify regression bugs when code is changed. This prevents time consuming and expensive system tests as well as identifying the location of errors accurately.

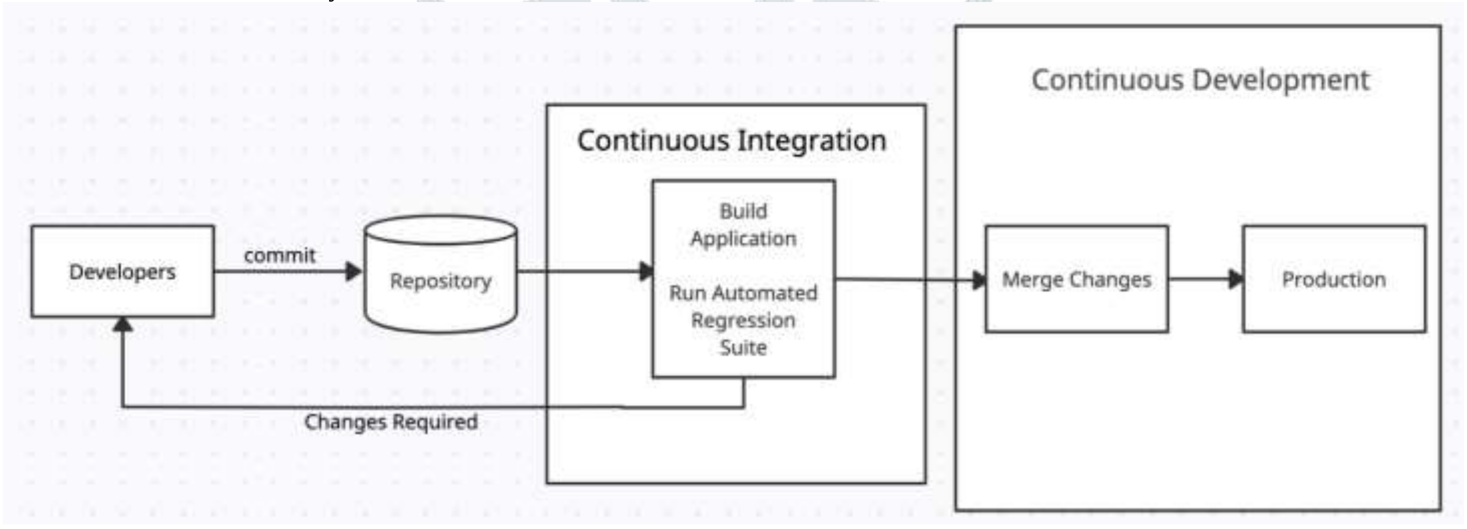

Fig 1. Architectural Diagram

It can help to ensure that units of code don't break the build when merged into the larger trunks. Wherever you need validation when one component is merged with another, regression tests can help to ensure that everything works as expected.
Agile practices with Continuous Integration and Continuous Delivery (CI/CD) pipeline approach has increased the efficiency of projects. In agile, new features are introduced to the system in each sprint delivery, and although it may be well developed, the delivery failures are possible due to performance issues. Automatically running regression tests means that tests can be run easily for every build. Automated regression can automatically generate a suite of passing unit tests by parsing the source code and determining all possible paths through the code. While there is always an effort to optimize the regression suite, there is also an

attempt to provide required coverage to ensure application does not break down in production. Distributed agile teams typically are characterized with ever-increasing size of regression test suite. The focus should be on automated regression testing for high-risk areas using risk-based testing. Automated testing is critical to that goal. There's no way to automate delivery to users if there is a manual, time-consuming step within the delivery process. Automated testing ensures quality at every stage of development by ensuring new commits do not introduce any bugs, so the software remains deployment ready at all times. Traditional software development methodologies are not enough to fulfill nowadays business requirements. Adaptation of Agile practices enables flexibility.

II. LITERATURE SURVEY

With increasing competition in software market, organizations pay significant attention and allocate resources to develop and deliver high-quality software at much accelerated pace. Continuous Integration (CI), Continuous Delivery (CDE), and Continuous Deployment (CD), called continuous practices for this study, are some of the practices aimed at helping organisations to accelerate their development and delivery of software features without compromising quality. Whilst CI advocates integrating work-in-progress multiple times per day, CDE and CD are about ability to quickly and reliably release values to customers by bringing automation support as much as possible. Continuous practices are expected to provide several benefits such as: (1) getting more and quick feedback from the software development process and customers; (2) having frequent and reliable releases, which lead to improved customer satisfaction and product quality; (3) through CD, the connection between development and operations teams is strengthened and manual tasks can be eliminated. A growing number of industrial cases indicate that the continuous practices are making inroad in software development industrial practices across various domains and sizes of organizations. At the same time, adopting continuous practices is not a trivial task since organizational processes, practices, and tool may not be ready to support the highly complex and challenging nature of these practices.

The first part of a CD process is Continuous Integration (CI). CI is a software development practice where developers integrate code frequently verified by an automated build (including test), to detect defects as quickly as possible. The second part of a CD process is Continuous Deployment (CDP). CDP is the ability to deliver software more frequently to customers and benefit from frequent customer feedback. However, according to non-academic articles, there's a missing part: Continuous Testing (CT). Two sources , define CT as the process of executing automated tests as part of the software delivery pipeline to obtain immediate feedback on the business risks associated with a software release candidate. Testing is considered by Humble and Farley as the key factor for getting CD, and they present a Deployment Pipeline (DP) as a CD model composed with different testing stages. However, while instructions on how to adopt these stages are given by these authors, some organizations have not adopted this practice at large yet and some of them have found it challenging.

The aim of Development team is pushing for the feature changes, whereas the goal of Operational team is striving for stability. Agile and DevOps are two emerging practices that provide big transition to the cultural change of every software organization for productive development. By adopting to automated continuous integration, deployment and Delivery practice, the ideas of agile and DevOps can be turned into practical solutions. Continuous Integration (CI) is a crucial section of Agile and DevOps practices as in and. CI is a key practice that frequently integrates a build with mainline shared repository by each developers in a developing team. This type of integration avoids a developer's local copy of code from drifting too extreme as soon as new code is affixed by other developers, avoiding disastrous integration conflicts. In practice, CI comprises of a centralized server which constantly check-ins all the new source code changes as soon as the developers commit them, reporting any failures during a build as in. CI server compiles, build, and tests every single fresh version of code committed to the main repository, it makes sure that the whole developing team is notified any time the mainline repository holds broken code. In advance, the CI server also deploy the verified application to quality assurance otherwise staging environment, safeguarding the Agile dream of a regular working version of the software product. Development teams take more than a few days for the Deployment process, even in cases of using automated CI to make sure that the code has been completely tested.

By adopting to automated continuous integration, deployment and Delivery practice, the ideas of agile and DevOps can be turned into practical solutions. Continuous Integration (CI) is a crucial section of Agile and DevOps practices as in and. CI is a key practice that frequently integrates a build with mainline shared repository by each developers in a developing team. This type of integration avoids a developer's local copy of code from drifting too extreme as soon as new code is affixed by other developers, avoiding disastrous integration conflicts. In practice, CI comprises of a centralized server which constantly check-ins all the new source code changes as soon as the developers commit them, reporting any failures during a build as in. CI server compiles, build, and tests every single fresh version of code committed to the main repository, it makes sure that the whole developing team is notified any time the mainline repository holds broken code. In advance, the CI server also deploy the verified application to quality assurance otherwise staging environment, safeguarding the Agile dream of a regular working version of the software product. Development teams take more than a few days for the Deployment process, even in cases of using automated CI to make sure that the code has been completely tested.

III. TECHNOLOGICAL DEVELOPMENT

Automated Regression Test is the testing area where we can automate most of the testing efforts. We run all the previously executed test cases on a new build. This means that we have a test case set available and running these test cases manually is time-consuming. We know the expected results, so automating these test cases is time-saving and is an efficient regression test method. The extent of automation depends upon the number of test cases that are going to remain applicable overtime. If test cases are varying from time to time, the application scope goes on increasing and then automation of regression procedure will be the waste of time. The Need of Regression Testing mainly arises whenever there is requirement to change the code and we need to test whether the modified code affects the other part of software application or not. Moreover, regression testing is needed, when a new feature is added to the software application and for defect fixing as well as performance issue fixing. Software maintenance is an activity which includes enhancements, error corrections, optimization and deletion of existing features.

Challenges and issues are -:

- With successive regression runs, test suites become fairly large. Due to time and budget constraints, the entire regression test suite cannot be executed.
- Minimizing the test suite while achieving maximum Test coverage remains a challenge.
- Determination of frequency of Regression Tests, i.e., after every modification or every build update or after a bunch of bug fixes, is a challenge.

Tools used are -:

1. **Selenium:** This is an open source tool used for automating web applications. Selenium can be used for browser-based regression testing.
2. **Quick Test Professional (QTP):** HP Quick Test Professional is automated software designed to automate functional and regression test cases. It uses VBScript language for automation. It is a Data-driven, Keyword based tool.
3. **Rational Functional Tester (RFT):** IBM's rational functional tester is a Java tool used to automate the test cases of software applications. This is primarily used for automating regression test cases and it also integrates with Rational Test Manager.

IV. APPLICATIONS OF THE CONCEPT

1. **Adaptive Software Development:** It is a technique that helps you build complex systems and software applications, by focusing on the evolution of software systems by hastening up their creation. ASD plays a vital role in the dynamics of self-organizing software development teams.

2. **Scrum:** SCRUM offers a collection of practices that should be adhered to, so that you can arrive at a consistent process within a framework. Employing the use of development cycles called Sprints, a SCRUM process stands apart from other Agile methods since it offers certain specific practices and concepts.

3. **Sprint Level Regression:** Sprint Level Regression is done mainly for the new functionality or the enhancement that is done in the latest sprint. Test cases from the test suite are selected as per the newly added functionality or the enhancement that is done.

4. **End-to-End Regression:** End-to-End Regression includes all the test cases that are to be re-executed to test the complete product end to end by covering all the core functionalities of the Product.

V. CONCLUSION

The main objective using Agile was to help the engineers with team-building, joint responsibility, self-managing engineers and to integrate their different individual design packages into an overall process design. Agile was well suited as there was a fair amount of unknowns to be developed and integrated. Here design iterations could be used to make incremental progress. Agile was well suited as no or limited planning was done earlier to implement the various projects. Most of the projects were small, not justifying the large upfront planning, that the traditional PM-Systems requires. There was a large amount of uncertainty; as the scope for every project was not clearly defined and developed sufficiently for a project at implementation phase. An effective regression strategy, save organizations both time and money. As per one of the case study in banking domain, regression saves up to 60% time in bug fixes (which would have been caught by regression tests) and 40% in money. Regression Testing is one of the important aspects as it helps to deliver a quality product by making sure that any change in the code whether it’s small or large does not affect the existing or old functionality. A lot of automation tools are available for automating the regression test cases, however, a tool should be selected as per the Project requirement. A tool should have the ability to update the test suite as the Regression test suite needs to be updated frequently.

REFERENCES

[1] Mojtaba Shahin; Muhammad Ali Babar; Liming Zhu, Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices, IEEE Access (Volume: 5), 2017

[2] S.A.I.B.S. Arachchi; Indika Perera, Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management, IEEE Moratuwa Engineering Research Conference (MERCon), 2018

[3] Thorsten Rangnau; Remco v. Buijtenen; Frank Fransen; Fatih Turkmen, Continuous Security Testing: A Case Study on Integrating Dynamic Security Testing Tools in CI/CD Pipelines, IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC), 2020

[4] Carmine Vassallo; Fiorella Zampetti; Daniele Romano, Continuous Delivery Practices in a Large Financial Organization, IEEE International Conference on Software Maintenance and Evolution (ICSME), 2016

[5] K. Sree Poornalinga; Dr. P. Rajkumar, Continuous Integration, Deployment and Delivery Automation in AWS Cloud Infrastructure, International Research Journal of Engineering and Technology (IRJET), Volume: 03 Issue: 05 | May-2016

[6] Sergejs Bobrovskis; Aleksejs Jurenoks, A Survey of Continuous Integration, Continuous Delivery and Continuous Deployment, Institute of Applied Computer Systems, Riga Technical University, Latvia

[7] Yasmine SKA; Janani P has mentioned in this paper A Study and Analysis of Continuous Delivery, Continuous Integration in Software Development Environment, JETIR, Volume 6, Issue 9, September 2019

[8] Jarosław Berłowskia, Patryk Chruściela, Marcin Kasprzyka, Iwona Konanieca, Marian Jureczkob, Highly Automated Agile Testing Process, e-Informatica Software Engineering Journal **(**EISEJ**),** Volume 10, Issue 1, 2016

[9] Pratibha Singh, Puja Patel, Impact of agile testing over traditional testing, Jaipur International Journal of Converging Technology and Management (IJCTM) Volume 1, issue 2, 2015