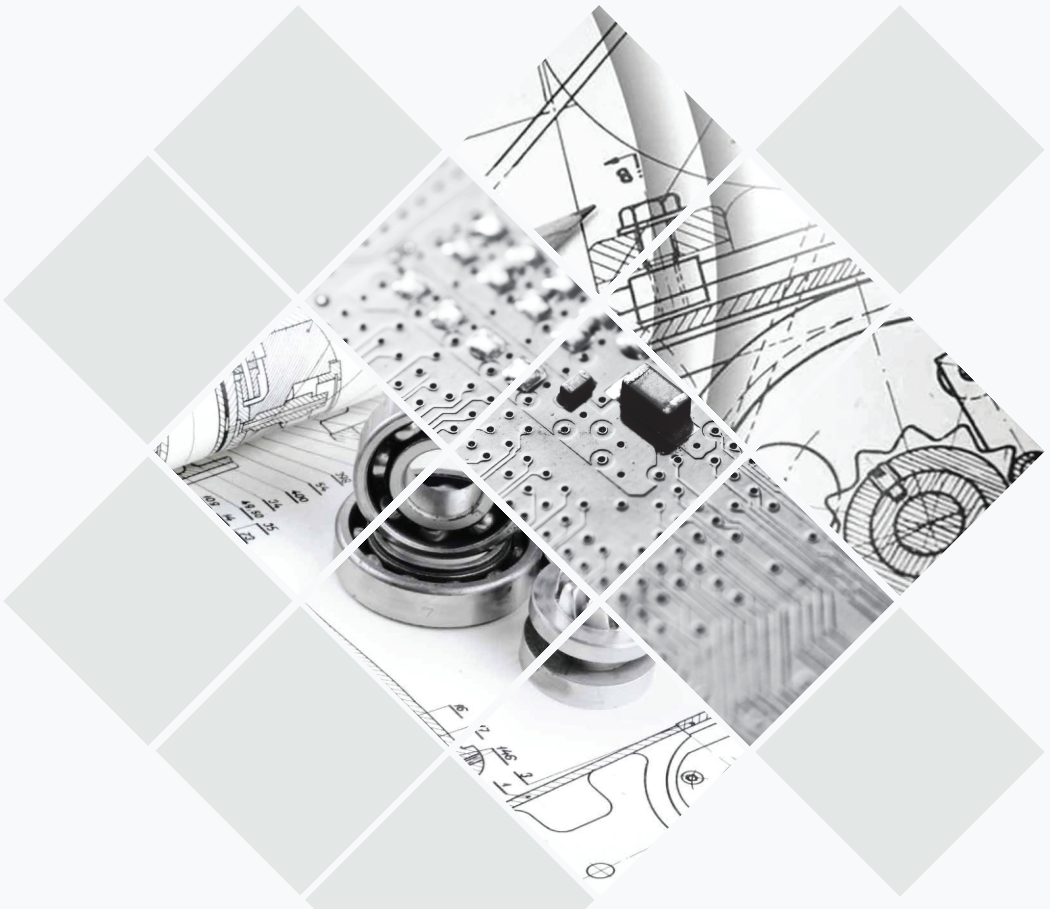
BORN TO ENGINEER

a new generation software test automation framework - CIVIM

Software Testing is the last phase in software development lifecycle which has high impact on the quality of the final product delivered to the customer.



Pradeep P P

QuEST Global

contents

A New General Software Test Automation Framework - CIVIM

Abstract

Software Testing is the last phase in software development lifecycle which has high impact on the quality of the final product delivered to the customer. Even after being a critical phase, it was not given the importance as it actually deserves. The schedule constraints and slippage carry forwarded from the previous phase also make the testing phase more torrent. History reveals that the situation has changed with time, wherein testing is now visualized as one of the most critical, phase of software development. This makes software testing a discipline which demands for continuous and systematic growth.

Software testing is a trade-off between Cost, Time and Quality. The concept of automated testing of the software has arised with increase in importance of the software testing. Test automation gained importance in Industry as it helps in improving the quality of the software at the expenditure of optimum cost and time. A Test Automation Framework is an infrastructure where multiple concepts and tools work together to provide a platform for automating the testing process.

This paper describes on a Test Automation framework that supports the concepts of Continuous Integration, Virtualization, Internationalization and Multiplatform. The framework does not wait for the final stage of life cycle, but catches defects at the early stages and ensures product quality at regular intervals. It takes a major leap in test engineering taking forward the discipline to a much more systematic and institutionalized.

1. Introduction

Quality is one of the most critical factors which govern the business growth of any company. Assurance of quality definitely comes with a cost. Early detection of defects makes a big impact on the project execution cost and schedule. In project environment where requirements creeps and gets added on in an incremental fashion, chance is high for defects to get injected in to the product. Frequent regression testing may be required to filter the defects at early stages of testing. It will be too costly to do regression testing manually in projects that follow Agile, Scrum, and other increment models.

Continuous Integration (CI) is a very powerful process which can bring high quality software products. Automating CI process can save huge effort in building the product and testing. It is an added advantage that the automated testing can be done without manual hindrance.

The number of regression cycles increases as the product needs to support multiple OS and configurations like Windows XP, Windows7 in 32 bit and 64 bit architecture. Most of the products support many platforms. It has a direct impact on the effort required for regression testing. So, it is important that the testing framework supports creation of different platforms virtually and running tests on them.

Internationalization makes the product testing more cumbersome since the language and other regional settings like currency and date time formats is different for different locations. It would be appreciable if the test automation framework ensures test coverage of features on all intended locations.

Maintenance is a stage where lot of effort can involve if the same is not considered at the time of test suite design. Changes in the product can easily be adapted to the automated test suite if the concepts like Keyword driven approach, data independent scripts, functional libraries, and independent Object repositories are taken care.

Decision Analysis and Resolution (DAR) is a good approach for choosing apt tools among the available set in the industry.

A New General Software Test Automation Framework - CIVIM

2. Current industry scenario

It is a common fact that software products support multiple platforms, languages and regional settings. There can be multiple test bed configurations on which product validation can be executed. Since testing is just followed by the customer release, it may not be possible to cover the testing of entire features in all configurations. As per the current industry scenario, complete regression testing is done in one configuration and only basic product features are covered in all others. It is not a safe practice since some defects are visible only in some specific test bed configurations.

CIVIM is an automated testing framework which takes care of testing in all applicable configurations with optimized schedule and cost.

3. Case study

Consider a product development, which involved developers working from three geographical locations. The code was checked in to the source code repository on a daily basis. The project followed agile work flow. Since frequent code changes happened, defects also injected into the code very frequently. Taking a clean build itself is a very hectic task since build errors was a common issue. The product supported two operating systems – Windows XP and Windows 7 in both 32 bit and 64 bit machines. It also supports different languages – English, Dutch, German and French. So, the total number of test bed configurations was counted to be 16.

Since the releases went to customers frequently, Quality Assurance (QA) team had to burn mid night oil most of the time. Complete test coverage in multiple test bed configurations was not possible. On the evolution of new product versions, the QA team had to validate multiple versions of the product versions at the same time, which proved to be a grave situation.

On considering the situation, it was decided by the QA team to design an automated validation system, which can ensure the quality of product in all the possible configurations and catch the defects at the earliest. The goal was to provide a quality report of the product on daily basis. The report shall provide the status of tests performed against different test bed configurations, build details and check in information. This gave birth to a new generation software test automation framework – CIVIM.

3.1 Concepts adopted in CIVIM

CIVIM framework integrates different concepts like Continuous Integration (CI), Virtualization, Internationalization and Multiplatform Support, which ensures health check of product at regular intervals.

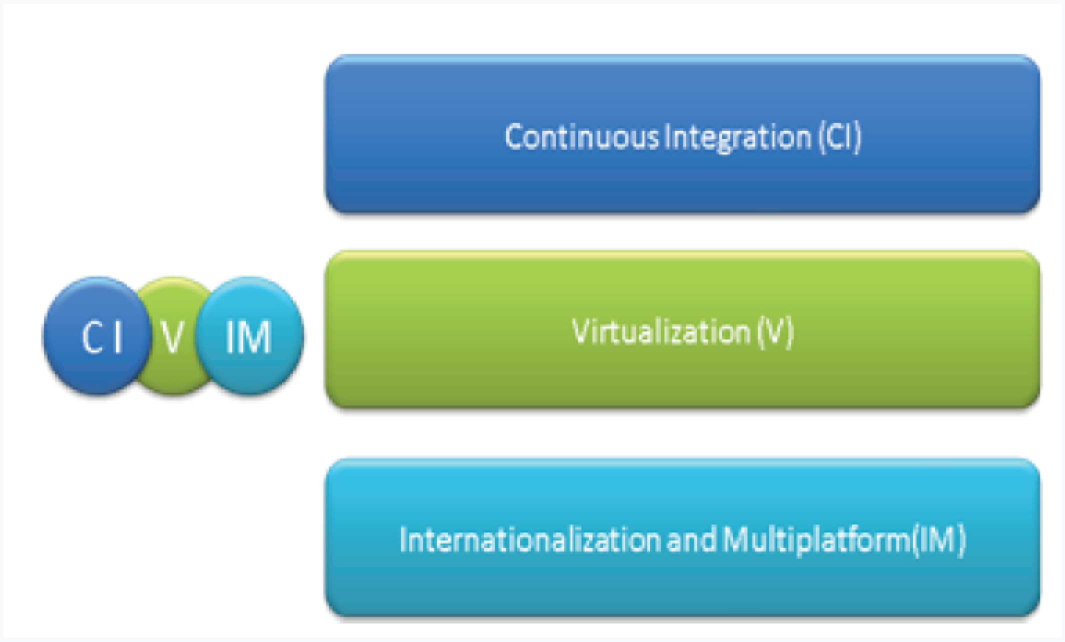

CI
V
IM
Continuous Integration (CI)
Virtualization (V)
Internationalization and Multiplatform(IM)

Figure 1: Concepts of CIVIM

A New General Software Test Automation Framework - CIVIM

Continuous Integration is an approach that implements continuous process of applying quality control. i.e. it ensures product quality at frequent intervals. It replaces the traditional practice of applying quality control after completing all development and reduces the time taken to deliver it.

It is often required to perform product validation on different machines like 32 bit and 64 bit with different operating systems. Virtualization is a technique of creating virtual image of a hardware platform, an OS, a storage device or a network resource. The goal of virtualization is to centralize administrative tasks while improving scalability and workloads. By adopting this idea, it is easy to create different test environments for testing multiplatform support of software products.

Internationalization is the process of designing a software product so that it can be adapted to various languages and regional settings without engineering changes. Automated test scripts may fail to perform if the language and regional settings are changed for the product. In order to perform regression testing corresponding to different languages and regional settings, CIVIM provides a solution. Strings corresponding to different languages and regional

settings are taken outside test scripts and stored in different files. According to the language and regional settings selected, the corresponding string file is loaded and they are referred by test scripts on demand.

3.2 CIVIM- Architecture

The architecture consists of five major blocks: Continuous Integration Server (CI Server), Configuration Management Server (CM Server), Build Server, Backup Server, and Test Server. The tasks to be performed are distributed into different servers so that the load is distributed evenly and total time of execution can be optimized. Each server has got its on responsibility of performing defined tasks and maintenance can be done independently.

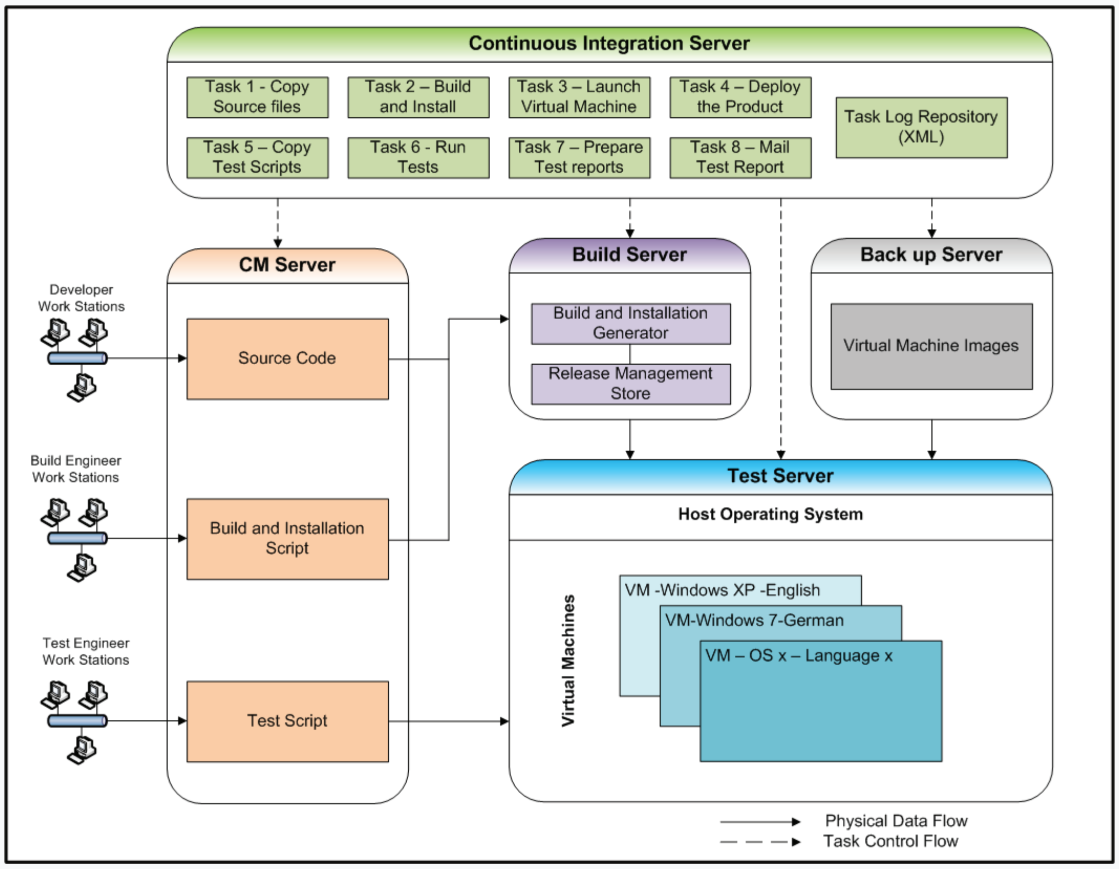

Continuous Integration Server
Task 1 - Copy Source files
Task 2 – Build and Install
Task 3 – Launch Virtual Machine
Task 4 – Deploy the Product
Task 5 – Copy Test Scripts
Task 6 - Run Tests
Task 7 – Prepare Test reports
Task 8 – Mail Test Report
Task Log Repository (XML)
CM Server
Source Code
Build and Installation Script
Test Script
Developer Work Stations
Build Engineer Work Stations
Test Engineer Work Stations
Build Server
Build and Installation Generator
Release Management Store
Back up Server
Virtual Machine Images
Test Server
Host Operating System
Virtual Machines
VM -Windows XP -English
VM-Windows 7-German
VM – OS x – Language x
Physical Data Flow
Task Control Flow

Figure 2: CIVIM Architecture

A New General Software Test Automation Framework - CIVIM

Continuous Integration Server
CI Server is the supervisory module in CIVIM architecture. It schedules the automated continuous integration process nightly. CI Server monitors and controls the operations of all other servers. It also maintains task status in a web based dash board which is visible to all stake holders of the project.

Configuration Management Server
CM Server takes care of version management of application source code, test scripts, build scripts and all supporting documents. Development team and QA team check-in work products into CM Server daily from their local work stations. CI Server may access CM Server for taking the source code and scripts.

Build Server
Build server is responsible for creating daily builds of the product. Upon request from CI Server, Build Server takes source code from CM Server and builds it. Then, it gets installation scripts from CM Server, makes an installation setup of the product and keeps it in a store. It also makes a build report and passes the same to CI server.

Backup Server
Backup Server contains a collection of virtual machine images. For each test bed configuration there is a virtual machine image. Virtual machines are created manually using tools like Microsoft Virtual PC and used on demand. Virtual machines are loaded on Test Server on demand.

Test Server
This is a high performance machine where regression tests run on multiple virtual machines concurrently. CI server initiates loading of virtual machines in Test Server and executes tests on required test bed configurations. Test Server prepares a detailed test report and sends to CI server.

3.3 CIVIM Work flow

The whole automated continuous integration process is scheduled to happen at night. CI Server initiates the process and manages the entire show. CI Server takes decisions as per the success/ failure of each task. The figure 3 depicts the flow of tasks which is scheduled nightly.

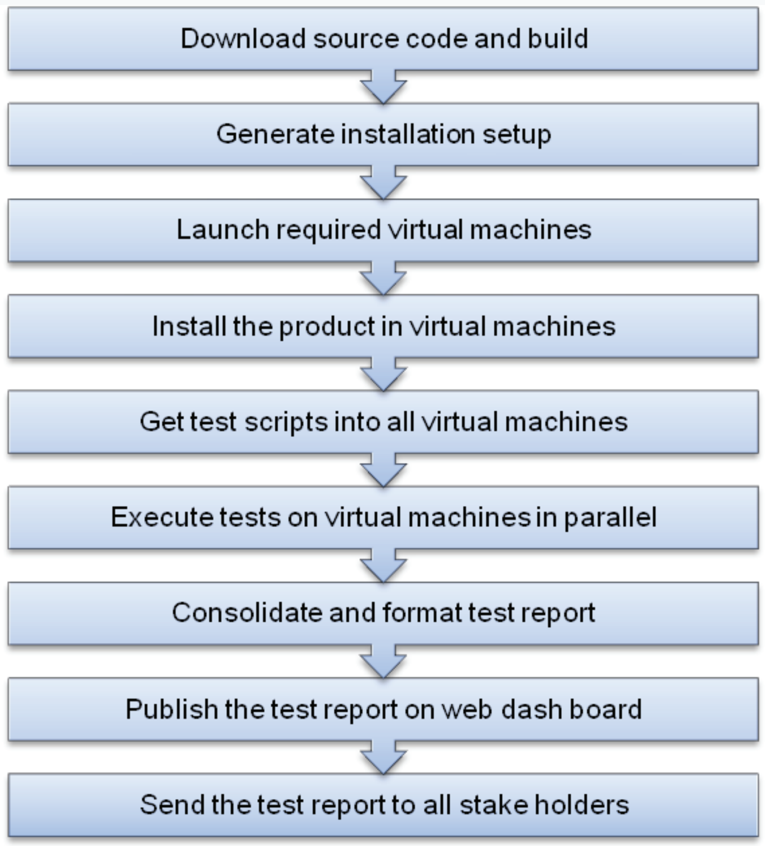

Download source code and build
Generate installation setup
Launch required virtual machines
Install the product in virtual machines
Get test scripts into all virtual machines
Execute tests on virtual machines in parallel
Consolidate and format test report
Publish the test report on web dash board
Send the test report to all stake holders

Figure 3: CIVIM work flow

A New General Software Test Automation Framework - CIVIM

At the scheduled time, CI Server initiates the activity for downloading the source code and build scripts from CM Server to Build Server. The source code is then built and installation setup is created for successful builds. 'Daily installation setups' are kept in a store for future reference.

Once the installation setup is successfully created, CI Server sets up virtual machines corresponding to each test bed configuration in Test Server after taking virtual machine images from Backup Server. Now all the test machines, in the form of virtual machines, with required configurations are ready in Test Server.

The CI server initiates the product setup build from build server and installs it in virtual machines in parallel. To each virtual machine, product setup is installed in parallel from Build Server. Test scripts are then downloaded from CM Server into each virtual machine and tests are executed in parallel. Test reports corresponding to each configuration are created in the corresponding virtual machine.

The test reports are consolidated and are published in a web based dash board. The virtual machines are then automatically switched off.

3.4 Test Suite Design – Some Salient points

Keyword driven approach for the test suite implementation was adopted resulting in eliminating the complexity of code during test case preparation. Scripting expert support is required in keyword implementation where as domain expert is required for test case preparation.

A rich set of function libraries needs to be created and shared across test automation engineers. Proper documentation is essential for the available function libraries.

An Object Repository needs to be created for keeping information regarding user interface (UI) controls of the product. This Object Repository was made independent of test scripts. Any changes in the UI could be adapted to Object Repository without making changes in test scripts. This helped to reduce the time to incorporate the product changes into test suite.

Parameterization of data via excel sheets needs to be used for UI validations like boundary value analysis, illegal entries etc we parameterized data via excel sheets. All input test data and corresponding expected results including error messages, if any were entered in excel sheets. The values in excel sheet are read by script and are placed in corresponding UI fields. The test scripts can be easily structured and modified using this method.

4. Tool Selection Criteria

The selection of a tool among the many available ones should be done wisely. The selection process should consider several factors. Decision Analysis and Resolution (DAR) methodology was used in identifying the tool which best suits the requirements.

DAR is a methodology in which all the identified tools are rated as per a set of criteria. Rating shall be given according to the features supported by the tool. Also the criteria shall be prioritized according to the needs. The most rated tool from DAR shall be selected.

5. Advantages of CIVIM

The various advantages in using the proposed CIVIM framework are:

1. **Greater Return on investment (ROI)**
   Test Automation using CIVIM helps in enhancing the test execution speed at a lower cost and higher ROI. CIVIM framework runs continuously at a faster test execution speed which increases the productivity of the organization.

2. **Unattended testing**
   CIVIM enables unattended testing of the entire software. And requires only very minimal manual involvement.

3. **Well formatted result logs**
   A well formatted test report is available. This helps in easy analysis of the test results which helps in reducing the effort spend on test result analysis and ensures readability of test logs.

4. **Early stage detection**
   Since the framework performs QA check every night, induced defects are caught in the very next day itself.

A New General Software Test Automation Framework - CIVIM

5. Better Infrastructure Utilization

Since the test automation is scheduled to be carried out at night, it helps in the most effective utilization of the infrastructure.

6. Increase over all product quality

CIVIM ensures an increase in the overall product quality. It ensures complete and continuous test coverage at a higher test execution rate. This helps in testing more or complete functionalities of the product as compared to the manual testing. The identification and isolation of the defects can be increased markedly, resulting to increased product quality.

7. Evaluation of the quality of product development

The framework provides the history of test reports on a web dashboard. This helps to understand whether the product quality is enhanced or not based on accumulated bug trend.

8. Efficient regression testing on multiple platforms

CIVIM supports efficient and effective execution of the regression test suite on multiple platforms without consuming time in project schedule.

9. Efficient regression testing of different language and regional settings

The framework supports testing of different language and regional settings of the software product.

6. Drawbacks

1. Requires initial set-up time

Setting up the test automation framework costs a remarkable effort initially even if the framework can be reused across multiple projects in the organization.

2. Needs skilled professional

It requires skilled professionals with experience in test automation.

3. CIVIM is not suitable for projects of short duration

The returns of initial investment for setting up the framework are achieved over a long run. So projects with long duration are only the right candidates.

7. Scope For Improvement

Defect injection can be prevented in the beginning itself, if the developers get a chance to ensure the quality of their code before check-ins. A feature can be brought in CIVIM for the developers to ensure the code quality at their local work stations before the check-in. i.e. there should be a provision in CIVIM to build and test the code locally and generate a test report. This helps them to make quality check-ins to CM Server.

Multiple check-ins are happening to the CM Server on daily basis. When a defect is caught in test report, next task is to assign the bug to the corresponding developer. In some cases, detailed analysis may be needed to understand the root cause of the defect and decide the assignee. If there is a mechanism, to identify the check-in which caused the failure, the defect can automatically be assigned to the corresponding developer. This can be achieved by traversing through an optimized set of versions in CM Server and performing the tests.

8. Conclusion

The paper portraits about a new generation test automation framework integrated with the concepts of virtualization, internationalization and multiplatform support. Even though there is an investment required at the initial stage, the benefits can be harvested in the long run. By the implementation of this framework, the product quality can be realized on daily basis and quality gaps can be filled at the earliest. Management gets a better control over the product releases with adequate test coverage. Product developments with long term scope are ideal candidates for automation using the

A New General Software Test Automation Framework - CIVIM

proposed frame work- CIVIM.

9. References

[1] http://en.wikipedia.org/wiki/Virtualization

[2] http://en.wikipedia.org/wiki/Continuous_integration

[3] http://en.wikipedia.org/wiki/Internationalization

[4] Michael Kelly, Getting started with automated testing: Road map to success

[5] Microsoft MSDN

Author Profile

Pradeep is working as Program Manager at QuEST Trivandrum. He has 10 years of experience in software testing industry and is an Engineering graduate from College of Engineering Trivandrum under Kerala University. He is a Certified Test Manager by STQC and Certified Tester by ISTQB - Advance Level Test Manager. He gives consultation for various testing tools like Selenium, J-Meter, QTP, Rational Functional Tester and Test Complete. He can be contacted at pradeep.pp@quest-global.com

BORN TO ENGINEER

www.quest-global.com