Visual Studio - Continuous Integration

Rama Murari[1], **Veerabhadraiah Sandru**[2]

[1,2]**HiTech ISU Assurance CoE**
TATA CONSULTANCY SERVICES LIMITED
Deccan Park, Hi-Tech City, Madhapur, Hyderabad, India.

Abstract

Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least once a day - leading to multiple integrations.

The shift in the development models from waterfall to agile, the distributed environments, parallel development, frequent and rapid feedback, inspections, deployment and repeated automated testing encourages the organizations to adapt continuous integration.

Continuous integration is a practice to be followed by the teams in addition to the usage of tools for automating the process. There are different tools for different elements of the continuous integration like build tools, test tools, code coverage tools, deployment tools etc. These tools are provided by different vendors. Organizations' selection of CI tools becomes tedious as a careful selection of mix and match of the tools has to be done after thorough evaluation.

Visual Studio provides a one stop shop for continuous integration implementation by having features to support all these needs of continuous integration. Organizations which have heavily invested in Microsoft technologies or moving towards it can look at Visual Studio for their CI needs. The tight integration of Microsoft technologies, tools and Visual Studio makes a valuable proposition for customers

This paper provides the quick overview on the philosophy behind continuous integration and gives details of VS support for CI practice and deep dives into the various features of VS that will support CI implementation.

Keywords: *Continuous Integration, Team Foundation Server and Continuous Integration, Tools for Continuous Integration, Microsoft Continuous integration, Visual Studio support for Continuous Integration*

1. Introduction

In earlier days with waterfall development methodology, integration was a task that was taken up only after the individual components are developed. Integration testing would come into picture after completion of development phase. When application to be developed had many complex components and required lots of integrations, it typically resulted in "Integration hell" where the identification of the root cause of the problem was very difficult. This big bang approach of integration testing resulted in cost overruns, cycle time slips, late discovery of defects, poor code quality and delays in time to market.

With the onset of Agile methodology adaption and parallel feature development, there is a need for early and frequent builds with static quality checks for every code change. This paved the way for the genesis of Continuous Integration which laid emphasis on both frequent integrations and tests. FIGURE 1 shows the gradual evolution of Continuous Integration (hereafter referred a CI) from periodic builds to builds with testing for every change.

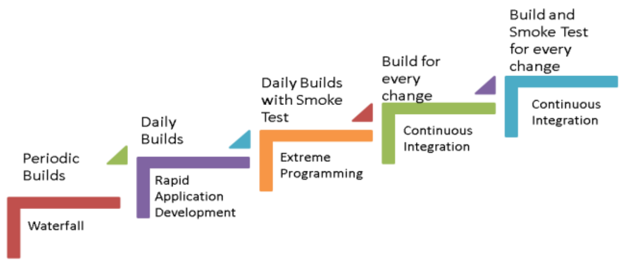


FIGURE 1: Evolution of CI

Next Section covers the key concepts of Continuous Integration.

2. Continuous Integration - Overview

Martin Fowler[1] who is considered the guru of Continuous integration defines CI as follows *Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible.*

CI leads to early detection of code, build, test and deployment failures thus reducing the cost to fixing the failures. Teams have to be proactive rather than reactive for practicing CI. CI addresses the risks faster and in

smaller increments. It creates the confidence in the application being developed as the overall health of the progress can be measured several times a day.
The basic principles of continuous integration are as follows

- Maintain a code repository
- Automate the build
- Make the build self-testing
- Everyone commits to the baseline every day
- Every commit (to baseline) should be built
- Keep the build fast
- Test in a clone of the production environment
- Make it easy to get the latest deliverables
- Everyone can see the results of the latest build
- Automate deployment

Continuous Integration can be either manual or automated. Continuous integration is at its best with automation of different processes in addition to the practices. FIGURE 2 shows the continuous integration work flow and TABLE I provide the type of tools required for CI implementation.

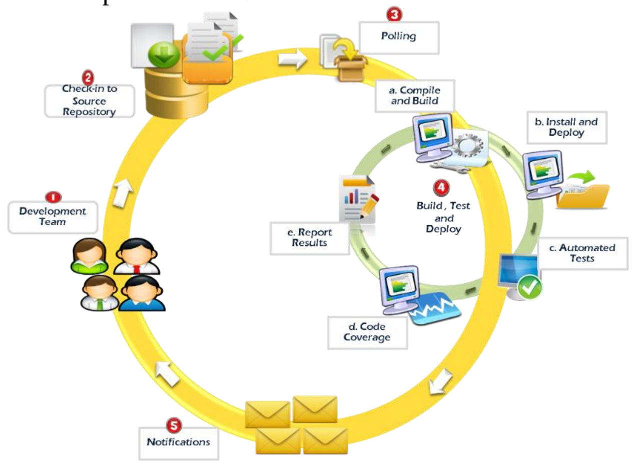


FIGURE 2: Continuous Integration flow

Code changes are committee to the source control by the development team. The CI server continuously polls the changes and triggers the builds periodically. The successful builds are deployed and tests are run to assess the quality and impact of the code changes. The results of the builds, deployment and tests are reported immediately. This process is practiced continuously. In a nutshell, CI gives instant feedback on the code changes, progress of the build, deployment and tests. A typical continuous integration server facilitates all the above activities. This decreases the human inventions by automation and enables continuous integrations.
In the market there are numerous enterprise-ready continuous integration servers which provide rich features such as source control, build management, release management, role-based security ,notifications and seamless integrations with 3rd party tools. Jenkins and Cruise Control are widely used CI servers. The following table gives a list of commercial and open source CI servers.

TABLE I
CONTINUOUS INTEGRATION SERVERS

Solution	Open Source	Latest Version	Source Link	Source
TFS	N	2012	http://tfs.visualstudio.com/	Microsoft
Jenkins	N	1.499	http://jenkins-ci.org/	Oracle
Cruise Contr	Y	2.8.4	http://cruisecontrol.sourceforge.net	Sourceforge
Hudson	Y	3.0.0	http://hudson-ci.org/	Eclipse
TeamCity	Y	7.1.3	http://www.jetbrains.com/teamcity	JetBrains
AnthillPro	N	3.8	http://www.urbancode.com	Urbancode
Bamboo	N	4.3.3	http://www.atlassian.com	Atlassian
Continuum	Y	1.3.8	http://continuum.apache.org	Apache
Continua CI	N	7.0.0.2056	http://www.FinalBuilder.com	VSoft Technologies
Pulse	Y	2.5.15	http://www.zutubi.com/	Zutubi

While Jenkins remains the well known CI server, Microsoft's Visual Studio now has all the key features to support the Continuous Integration without the need for having varied tools for different CI elements. This would be extremely useful for the customers who have embraced Microsoft technologies in the application development environment.

This paper gives the high level overview of the features of CI and how VS/TFS support these features. Microsoft msdn site has further configuration details for CI implementation.

3. Visual Studio (VS) Support for Continuous Integration

FIGURE 3 shows how Visual Studio along with Team Foundation Server supports the CI process. VS supports implementation of CI by offering rich features such as version control, build management (Team foundation Build/TFB), Unit Testing (Visual Studio) and static testing like code coverage, code analysis. In addition to code maintenance and build, TFS/Visual Studio also has extensive features for deployment and installation. TFS supports notifications and alerts. TFS and Visual Studio are interdependent and features that can be triggered from Visual Studio are given here though they may in turn interact with TFS.

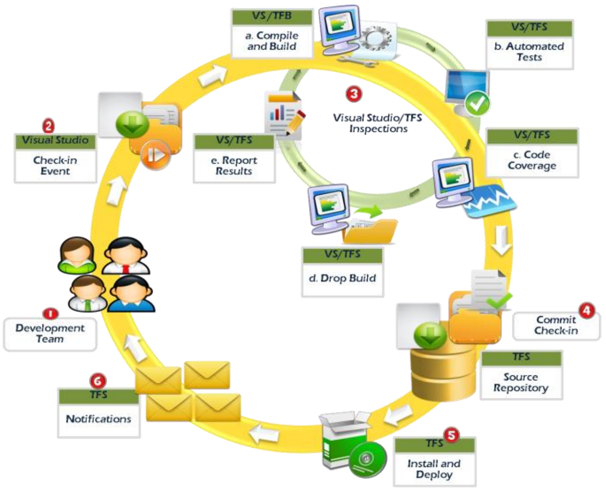


FIGURE 3: CI supporting TFS/VS features

Details of key features of TFS in lieu of the above are given in the next section

4. Details Of VS Key Continuous Integration Features

Visual Studio provides a single point solution for all CI supporting features like source control, build management, test management, work item tracking and lab management. The TFS source control feature helps in code maintenance so that the team has the same version at any given point of time. The code committed to the version control automatically triggers the inspections, builds and unit tests. The TFS build management system (Build controllers/agents) supports compiling and building the code as per the build definition. The lab management in TFS allows creation of virtual environments with required configuration. The Test Controller along with the Test agents run the test cases in the test environments. FIGURE 4 provides the high level architecture of TFS that supports above mentioned features.

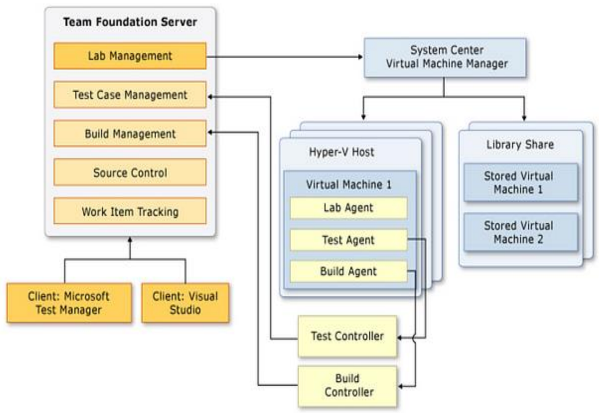


FIGURE I TFS Architecture

TABLE II shows a snapshot of the features available in VS and TFS to support Continuous Integration process.

TABLE II
VISUAL STUDIO: KEY CONTINUOUS INTEGRATION FEATURES

S.No	Activity	TFS/VS
1	Source Repository	TFS
2	CI Server	TFS
3	Build/ Deployment	TFS (TFB)
4	Unit Testing	VS
5	Code Coverage	VS
6	Code Analysis	VS
7	Code Metrics	VS
8	Installation	TFS
9	Reports	TFS/VS
10	Notification	TFS/VS
11	Bug Tracking	TFS
12	Test Automation	VS Coded UI

4.1 Source Control Support in TFS

TFS serves as central repository for source code and the other project artifacts. It stores all the project artifacts and all the changes made to them over time. It provides controlled access to the all the project artifacts. It provides the flexibility and ease to maintain different versions of the application. The code can be merged or restored back to any stage at any point of time. It also supports versioning of database schema to keep data and code in synch.

The most commonly used SCM (Software Configuration Management) tools used are ClearCase, CVS, Git, Mercurial, Perforce, StarTeam, Subversion, MKS, SourceGear Vault, and Visual SourceSafe.

TFS provides the main baseline structure. As per the project needs, branches are created for maintenance of the code.TFS allows the following branching strategies

- *Feature Branching: Branching based on the features to be developed*
- *Release Branching: A branch is created to stabilize the release and later merged to main branch after the release.*
- *Quality Branching: Branching done for different teams with focus on quality*

TFS allows merging of code from different branches.

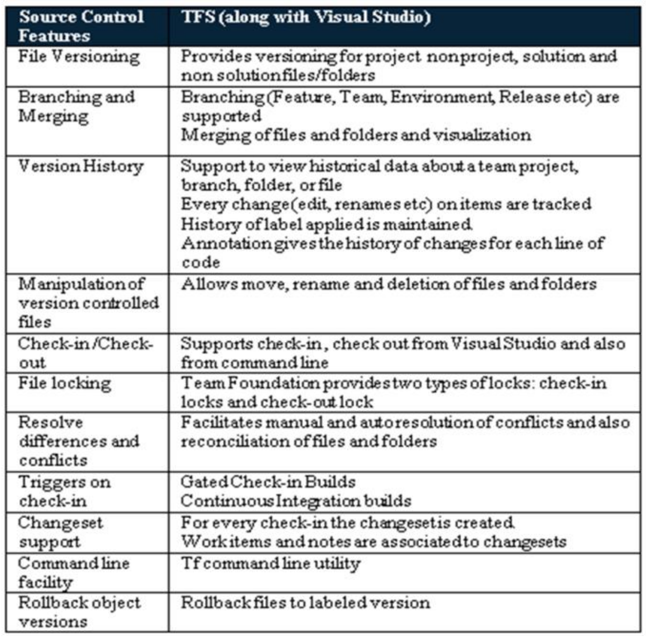
The code is checked out, appropriate changes are made and changes are checked in after conflict resolution. In parallel development, multiple developers can work simultaneous on the same code base. Developers create a local workspace on their local machines. The workspace is mapped to the TFS server code base. TFS proxy server provides local cache for the version controlled information that will synchronize with the master copy in the central TFS server regularly.

Developers make changes to the code in the local workspace. Parallel development might result in conflicting changes. Visual Studio along with TFS aids in comparing the differences and resolving the conflicts either manually or automatically. The changes in the workspace can be shelved. The pending changes in the shelve-sets can be discarded.

When the check-in is triggered, the check-in policies are validated. TFS supports the policy validations for work items, builds, tests, code coverage analysis etc. Additionally if the available policies do not match to the needs of the project, it allows creation of custom policies. After automated tests and code coverage is run, changes are checked into TFS with appropriate comments.

TABLE III gives the common source control features and TFS support for these features

TABLE III
TFS/VS SOURCE CONTROL FEATURES



Source Control Features	TFS (along with Visual Studio)
File Versioning	Provides versioning for project nonproject, solution and non solution files/folders
Branching and Merging	Branching (Feature, Team, Environment, Release etc) are supported Merging of files and folders and visualization
Version History	Support to view historical data about a team project, branch, folder, or file Every change(edit, renames etc) on items are tracked History of label applied is maintained Annotation gives the history of changes for each line of code
Manipulation of version controlled files	Allows move, rename and deletion of files and folders
Check-in /Check-out	Supports check-in , check out from Visual Studio and also from command line
File locking	Team Foundation provides two types of locks: check-in locks and check-out lock
Resolve differences and conflicts	Facilitates manual and auto resolution of conflicts and also reconciliation of files and folders
Triggers on check-in	Gated Check-in Builds Continuous Integration builds
Changeset support	For every check-in the changeset is created Work items and notes are associated to changesets
Command line facility	Tf command line utility
Rollback object versions	Rollback files to labeled version

The check-ins triggers the builds as per the build configuration. The next section gives details of the build architecture and the build process.

4.2 Build Support

TFS supports build using Team Foundation Build. Team Foundation Build uses MSBuild script to control the build process. The code, database schemas and the other project artifacts from the source control are used to build the application executables.

As part of build definition TFS allows specifying the program and testing elements for execution, the triggers points of build, deployment and the status retention of different builds that helps the developers, testers, build/release managers and project managers.

Visual Studio supports the following build triggers.

- Manual : Builds that are triggered manually
- Rolling Check-in Builds: Build triggered after regular intervals of time. Check-ins are accumulated till previous build succeeds
- Continuous Integration Builds: Build triggered after every check-in, code is checked-in even if the tests fail.
- Gated Check-in builds: Check-in is done only after the changes merge and build is successful.
- Scheduled Builds: Builds is trigger on any day and any time.

Visual Studio supports Continuous Integration by triggering Continuous Integration builds or Gated Checkin builds. For Continuous Integration build configuration if build fails the executable is not available till the build is fixed. This interrupts the work of the team, hence Gated checkin configuration helps to overcome this hurdle by not committing the code to TFS if it breaks the build. This helps the teams to work, uninterrupted by failures.

TFS allows the build steps to be specified as workflow that uses Windows Workflow. TFS provides three predefined build definition process templates (default, upgrade and labDefault). The build steps are complex hence Visual Studio has a visual representation to easily understand and modify the workflow as per the project needs.

After the build is completed, executables are deployed in the different environments. TFS also provides the reporting capabilities giving all the build details. Emails and alerts can be triggered for the status reports.

Build reports gives the results of build completion, code analysis and the tests run. For build failures link are provided to the bug work item, based on which the teams are notified and the failure is fixed without delay. Build Report also provides the direct navigation to the changesets, code analysis warning and the test results. The data from the build report is used for metrics reporting.

A daily build shows all code changes and work items incorporated since the last daily build, whereas a release

build shows a list of all changes checked in since the last release.
The following table gives the few key build features of any Build management application and the how these features are supported in Visual Studio/TFS.

TABLE IV
TFS BUILD FEATURES

Build management	Team Foundation Build/ Visual Studio
Parallel builds	BuildInParallel build task parameter should be set to True.
Distributed builds	Multi processor and Multi environment builds
Manually force builds	Manual Build option
SCM trigged builds	Gated Checkin or Continuous Integration Checkin options
SCM poll based builds	Rolling builds option
Temporal build scheduling	Scheduled Build option allows scheduling the build to run a particular day and at a particular time
Incremental Builds	The timestamps of input and output files are compared to determine whether to skip, build, or partially rebuild a target
Trigger Tests during Builds	Supported by Automated Tests and Test Assembly Enable Tests options
Support for Code Analysis	Supported by Perform Code Analysis parameter
Queue and Stop Builds in Progress	VS supports Queuing build in build definition. An In Progress or Postponed build can be stopped from the Queued tab in Build Explorer
Notify Build Failures	TFS supports email notifications
Build Logs	VS supports the build event information logging in console, file, customized logger for multiprocessor and multi environment builds

Lab Management feature supports in setting up the test labs on physical or virtual machines. Templates can be created for environment and can be used for recreation.

Visual Studio helps to run tests remotely, distribute tests across multiple machines, by configuring test controller, test agents, and test settings file. The build definition supports automatic deployment and test runs.

4.3 Code Coverage

Visual Studio facilitates code coverage which measures the percentage of code which is covered by automated tests. Code coverage measurement determines which statements in a body of code have been executed through a test run, and which statements have not. Visual Studio supports code coverage analysis. This is a structural testing technique which compares the test program behavior against the source code
Code coverage analysis helps in finding the testing deficiencies in covering the areas of the program. Test cases can be added for better coverage while redundant test cases can be removed to improve the quality.

TABLE V
VISUAL STUDIO CODE COVERAGE

Coverage	Details	Measure	Color
Code blocks	A set of instructions that run in a sequence with a single entry and exit point and ends when a decision point is reached	**Lines** : Covered/Not Covered **%Lines**: Covered/Not Covered	Light Blue: Line of code covered
Lines of code	An executable line of code	**Blocks** : Covered/Not Covered **%Blocks**: Covered/Not Covered	Reddish brown: Line not covered
Partial lines	The portion of the line, containing multiple blocks that is exercised by the test run	**Partial Lines** : Covered and %Covered	Beige: Portion of code block covered

4.4 Code Analysis

Visual Studio supports code analysis for both managed and unmanaged code. A set of predefined rulesets are defined and categorized. New customized rule-sets can also be created.
The rulesets check the rules of some of the features like Security, Design, Globalization etc. The rules cover wide range of warnings from key aspects such as performance, interoperability, reliability etc. The Code Analysis tool provides warnings/errors that indicate rule violations in libraries or executables.
The following gives the list of rulesets and rules categories.

TABLE VI
VISUAL STUDIO CODE ANALYSIS

RULESETS	
Microsoft All Rules	Microsoft Basic Correctness Rules
Microsoft Minimum Recommended Rules	Microsoft Extended Correctness Rules
Microsoft Globalization Rules	Microsoft Basic Design Rules
Microsoft Security Rules	Microsoft Extended Design Guidelines

RULE-CATEGORIES		
Design Warnings	Portability Warnings	Mobility Warnings
Security Warnings	Interoperability Warnings	Usage Warnings
Globalization Warnings	Reliability Warnings	Code Analysis Policy Errors
Naming Warnings	Performance Warnings	Maintainability Warnings

4.5 Unit Testing

Visual Studio provides Unit Testing framework for testing the code and databases. The generation of unit tests for the methods is made simple. Visual Studio creates the sample test method for the code with the default inputs, outputs and validation statements. Default assertions statements are created which need to be refined with valid inputs, outputs and validation. Additional logic and assertions can also be incorporated. Selected tests can be executed. The test results can be saved to disk, published in TFS or exported as required. Predefined assertions are available and can be extended. The exceptions validation can also be handled in unit tests

Database Unit Testing helps with default schema, schema objects (procedures, functions, triggers etc), security (authentication & authorization) and static data validations. Visual Studio additionally provides the capability to generate data for Unit Testing. The database scripts deployment is also automated for early integration.

The importance of unit testing is increasing with the increase in the evolution of TDD (Test Driven Development) and Agile development. Visual Studio extensively supports TDD by refactoring.

Additionally Code Metrics in Visual Studio gives the quality measure of the source code. This helps in giving feedback on the design of the code.

4.6 Code Metrics

Visual Studio can measure the quality of code by collecting code metrics. Code metrics is a set of software measures that provide developers better insight into the code they are developing at regular intervals which stand as a check for the quality. TABLE VI gives the Visual Studio supported for code metrics

TABLE VII
VISUAL STUDIO CODE METRICS

Metric	Description	Recommended value
Maintainability Index	Represent the ease of code maintenance 20- 100 – Green – Good Maintainability 10-19 – Yellow – Moderate Maintainability 0-9 – Red – Low Maintainability	High
Cyclomatic Complexity	Measures the structural complexity of the code	Low
Depth of Inheritance	The number of class extensions from the root class	Low
Class Coupling	Measures interdependencies between classes	Low
Lines of Code	Approximate measure of number of lines in the code	Low

In addition to these, Visual Studio provides the deployment, notification and reporting features to provide end to end support for CI.

The next section gives the benefits different teams get by practicing Continuous Integration.

5. Continuous Integration Benefits

CI solution provides better sense of the development progress and code quality of applications continuously throughout the development lifecycle. In impementing the CI process in software development all the teams have a collective ownership of the appliation. All the team have their own benefits for CI adaption FIGURE 5 gives the details.

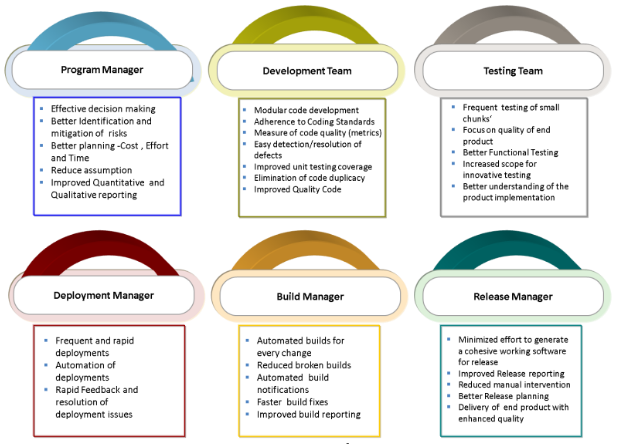

Program Manager
Effective decision making
Better Identification and mitigation of risks
Better planning -Cost , Effort and Time
Reduce assumption
Improved Quantitative and Qualitative reporting
Development Team
Modular code development
Adherence to Coding Standards
Measure of code quality (metrics)
Easy detection/resolution of defects
Improved unit testing coverage
Elimination of code duplicacy
Improved Quality Code
Testing Team
Frequent testing of small chunks'
Focus on quality of end product
Better Functional Testing
Increased scope for innovative testing
Better understanding of the product implementation
Deployment Manager
Frequent and rapid deployments
Automation of deployments
Rapid Feedback and resolution of deployment issues
Build Manager
Automated builds for every change
Reduced broken builds
Automated build notifications
Faster build fixes
Improved build reporting
Release Manager
Minimized effort to generate a cohesive working software for release
Improved Release reporting
Reduced manual intervention
Better Release planning
Delivery of end product with enhanced quality

FIGURE 5: Benefits to CI Team

6. Visual Studio Continuous Integration Best Practices

Some of the best practices to practice CI with TFS/VS are given below [6]

- Use branching to reduce builds failures.
- Commit Code changes in TFS at least once a day
- Synchronize workspace with the code in TFS server and resolve conflicts to ensure no data loss, before check-ins.
- Run Private builds to check build failures, before committing changes to TFS
- Execute automated unit testing with 100% test pass for the each code change
- Inspect the code with Code Coverage, Code Analysis and Code Metrics
- On larger teams, install the build services on a separate server.
- Use a CI build option to get rapid feedback on check-ins.
- Use Gated check-in build option, to fail check-in when build or automation tests fail.
- Use check-in policies to improve code quality.
- Use check-in policies to associate work items with the build.
- Use build notification alerts to for build status.

7. Key Challenges In Implementing Continuous Integration

While CI helps the organization to reduce cost and time to market it increases productivity and quality, it certainly requires process, organization and technology changes. Typically the challenges would be as follow

- Time, effort and cost in evaluation of CI support tools/server
- Time for initial CI setup
- The cost to procure additional hardware and software for CI implementation
- Acceptance of CI best practices by all the stakeholders
- Distractions due to frequent commits, builds and notifications
- Suitability of feature for automation (not all features can be automated)
- Learning curve to gain knowledge, for implementation of CI features
- Multi vendor tools to scatter to all the disciplines of application development
- Execution time for builds, tests and deployment

7.1 Upcoming Trends in Continuous Integration

There are various upcoming trends in CI that need the organizations attention to reduce the time to market, risks for the customer and increase the quality.
For large projects with huge teams which span across various geographies the strategies to best implement CI raises a need for Distributed and multi-stage CI implementation.

7.2 Distributed CI

In a Distributed environment, the load will be transferred to build clients when build cluster is overloaded. There is a primary server that creates replicated servers in the environment. In distributed CI, multiple projects and multiple source control tools are used. The secondary server might be remote or local. The build status of all servers can be monitored and controlled through the primary server. Microsoft TFS Build Architecture supports distributed CI with Build controller and Build Agents and the readiness for this trend adoption.

7.3 Multi-stage CI [7]

This involves integrating at multiple stages. Each developer integrates and tests the individual work and then integration will be done at team level to prevent the failure in the entire project. The work done by different teams is integrated and then released to QA for integration testing. The logical division of the work of the entire project across different teams and individuals, the level of testing and integration done at each level plays a key role. Microsoft TFS alignment to multi stage CI includes its supports for the individual integration and unit testing in the local workspaces. The branching strategies (branch, team, quality, feature) available in TFS enable the multi-stage CI implementation. TFS allows merging of changes from different branches for integration.

7.4 CI on Cloud

"The evolution of continuous delivery and the rise of new cloud platforms will drive new combinations or stratifications in the packaging of development execution and application life cycle management (ALM) functionality." *Jim Duggan; Thomas E. Murphy, Analyst, Gartner*

The hardware and software required for CI implementation is one of the challenges to adapt CI. Organizations that have requirements for multiple operating systems, multiple browsers, multiple programming languages etc are in the process of adapting CI having the avenues open for Cloud CI. The IAAS (Infrastructure as a Service) and SAAS (Software As A Service) cloud services can be used for provisioning the required infrastructure and the software for CI implementation on demand and with better pricing models which will allow organization to save on cost and time and improve quality and productivity. Microsoft provides Windows Azure based Visual Studio Team Foundation Service and SAAS offerings to support CI implementation on cloud.

TFS preparedness for cloud: Cloud-hosted service version of Microsoft's popular Team Foundation Server (TFS) software that provides highly customizable source code and builds management, agile development and team process workflow, issue and work item tracking, and more. Team Foundation Service team projects can be automatically configured to automatically build and deploy to Windows Azure websites or cloud services

7.5 Automated Deployment and Continuous Delivery

"The biggest evolution that we're seeing is people moving from CI to what they're calling continuous deployment," says Jeffrey Hammond, Principal Analyst, Forrester.

Continuous Deployment is the process where the application is continuously deployed in to production for every code change. The aim of continuous deployment is to reduce cycle time and effort involved in the deployment process. Solution for continuous deployment is continuous delivery. It does not deploy on every change, but the deployment process is automated and will be triggered as required.

Conclusions

- CI advantage of rapid feedback, rapid deployment and repeated automated testing leading to rapid delivery ,reduced human interventions and costs have seen a wide enterprise wide acceptance
- The raise in the adaption of agile methodologies and TDD increases the scope of CI implementation because it gives the health of the project several times a day.

- There various open source tools available in the market for organizations to implement CI. CI tools like Jenkins, Cruise control, Hudson work well with 3rd party tool to support CI feature like version control, build management, static testing, automated testing, functional testing, installation and deployment.
- Organizations which have already invested in Microsoft technologies or moving towards it can look at TFS/VS, which offer's rich capabilities for CI practice. The heavy coupling of Microsoft technologies, tools and TFS makes it a commendable valuable proposition for Microsoft savvy customers.

References

[1] Duvall, P., S. Matyas, and A. Glover.CI: Improving Software Quality and Reducing Risk. Addison-Wesley, 2007

[2] Martin Fowler about CI http://www.martinfowler.com/articles/continuousIntegration.html

[3] Microsoft msdn for TFS and Visual Studio. http://msdn.microsoft.com/en-us/library/dd831853(v=vs.100).aspx

[4] Continuous Integration wiki http://c2.com/cgi/wiki?

[5] Manifesto for Agile Software Development. http://agilemanifesto.org/

[6] J.D Meier, Jason Taylor, Alex Mackman, Prashant Bansode, Kevin Jones : Team Development with Visual Studio Team Foundation Server Microsoft Corporation, 2007

[7] Multi Stage Continuous Integration Trends http://www.informationweek.com/multi-stage-continuous-integration

[8] Continuous Integration Flow chart http://www.falafel.com/images/CI_chart.jpg

[9] Veerabhadraiah Sandru, Rama Murari, "Changing Role of Test manager in Changing Situations", IJCEM, vol. 16 issue 1, Jan. 2013.

Author's Biographies

Rama Murari has 14 years of IT experience and is into Software Testing from past 9 years. She has done her masters' from National Institute of Technology, Warangal. She has ISTQB Foundation certification. She has co-authored white papers on testing in QAI, IJCEM, Step-In forums. Her white paper on 'Pandora's White box testing' has been published in STEP-IN's forum. She has been in various roles of developer, business analyst, functional analyst, tester, project leader and program manager for large and renowned accounts. She is working as a Solution Developer in Assurance CoE of HiTech Industry Solution unit of Tata Consultancy Services. Her areas of expertise include Test Process Consulting, Test Management and White box testing.

Sandru Veerabhadraiah has 18 years of experience in IT and 10 years of experience in software testing. He has masters' degree in engineering from Indian Institute of Technology, Madras. He has certifications in ISTQB-Foundation and TOGAF-Foundation. He had earlier published 2 white papers in TCS's Global Technical Architects conference. He also published white papers on testing in QAI, IJCEM and recently conducted a 2 hour tutorial on Transaction Based Pricing at STEP-IN's Hyderabad Software Testing Conference. Currently he is leading the Assurance CoE of HiTech Industry Solution unit of Tata Consultancy Services. His area of expertise include test process consulting, test management, test automation and test methodologies in usability testing, globalization, accessibility testing.