Software Development using Build and Smoke Tests

Mohan Srinivasan,
Senior Project Manager,
ANZ Information Technology,
Bangalore, India

Introduction

Most projects have either schedule or effort overruns, or both, which leads to customer dissatisfaction. This is a widespread problem faced by most project managers. What usually happens is that issues that crop up during the integration testing, lead to delays in delivery of software.

We received a major project of 100 person years to be delivered in 9 months. Our goal was not only to meet the client's expectations, but to exceed them. We started looking at various ways and means to meet this expectation. This included reviewing existing tools and techniques and also looking at best practices prevalent across the globe. The team decided that it would not be possible to achieve our goal, unless they did something radical.

We decided that one of the techniques the team would adopt was a weekly "Build and Smoke Test".

Definitions:

Build: Build is defined as any of various versions of a software product that are being developed for release to users.

Smoke Test: Online Computing directory defines Smoke test as follows:

1. A rudimentary form of testing applied to electronic equipment following repair or reconfiguration, in which power is applied, and the tester checks for sparks, smoke, or other dramatic signs of fundamental failure.
2. By extension, the first run of a piece of software after construction or a critical change.

There is an interesting semi-parallel to this term among typographers and printers: When new typefaces are being punch-cut by hand, a "smoke test" (hold the letter in candle smoke, then press it onto paper) is used to check out new dies.

Implementing for the first time is by no means an easy task. It requires a lot of discipline and changes to the way you do your work.

Frequency of Build

The principle behind weekly "Build and Smoke Testing" is to build the components and perform smoke testing every week, so as to reduce integration issues later. It could also be performed daily if several components are developed every day.

We tried implementing it on a daily basis but found that few components were getting developed on a daily frequency. We ended up building the same system two days in a row, and hence decided that the frequency of build could be weekly.

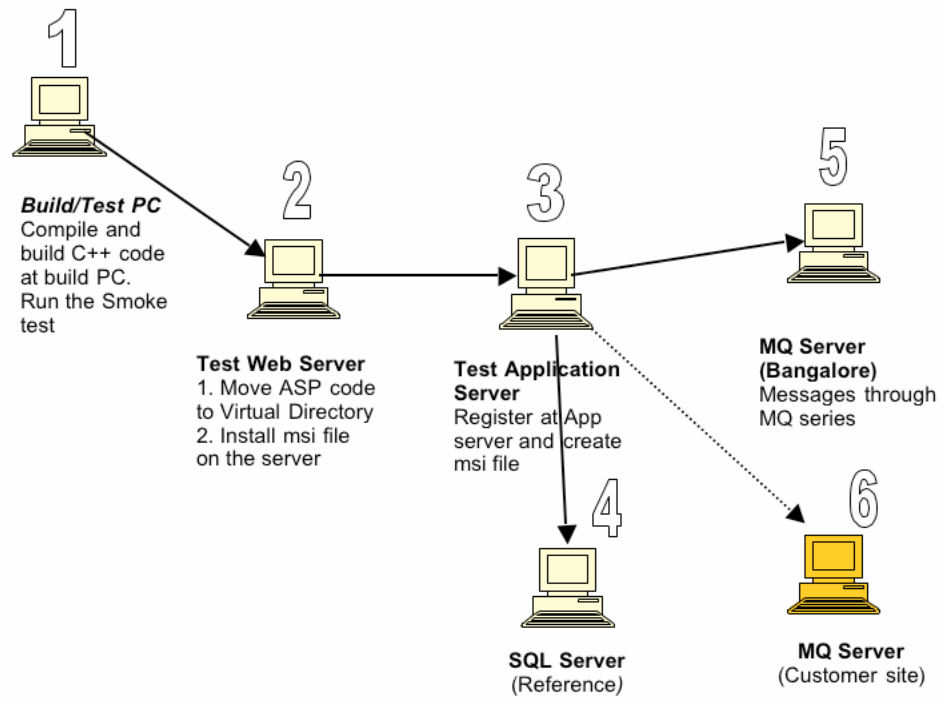
We first organised training sessions on weekly "Build and Smoke Testing". The team felt that it was a good concept and it would definitely help them. They also suggested that it would be time consuming, if weekly testing was done manually and a "Build and Smoke" test tool should be developed.

Implementing Weekly Build

It was quite important that the "Build and Smoke Testing" activity be incorporated in the Project plan. This ensured that adequate resources are allocated for the build activity and we knew what components are undergoing weekly testing. The key is to know the progress of the project accurately.

The weekly build gave an enormous amount of visibility to the Project Manager as he could see the components being integrated, and issues being resolved each week. Earlier the issues would be known only during integration testing. Some of the issues would be major and it would take up a lot of effort and consequently impact the schedule.

There was a Schedule drawn for weekly "Build and Smoke Testing" for each project, and a manager responsible for overseeing the test activity. It ensured that the complete activity was done in a clean environment (a build PC). The Process used is outlined in Appendix A



Broken Builds

It had to be ensured that the build was not broken. If the build breaks, then the following needs to be done:

- First check the build report for information, if that info is not sufficient, contact the manager to determine who is responsible. Contact that person and ask them to fix it.
- If the build does not compile or fails the smoke test, the manager should contact the developer responsible for that particular function and ask them to fix the code.
- Whenever the build breaks, it should be documented, and corrected in the version control system immediately. However, the weekly build that was broken should not be delayed or rebuilt. If a working weekly build is required for testing purposes, a second build may be created and posted with the weekly builds.
- All developers in the team should be instructed to check-in any code only after the entire module is tested to determine if their changes had an adverse effect.

Getting Customer to view the Product early

Adopting a weekly build process in the programme provided immense benefits to us. We could reduce time for integration testing. The integration issues were also reduced and the project team could be confident in delivering the product on time.

Usually in software development, “needs” and “wants” of a customer are different. Even if the system is designed with great care, using experienced analysts and the latest design techniques, it could fail to solve the customer's problem.

The product we were delivering may be as per the “wants”, but may not fulfil customer’s “needs”. The weekly “Build and Smoke Test” aims to address this issue effectively. The customer could see the product evolving mid-way and get a feeling of comfort. Any changes desired by customer could also be incorporated well in advance. The customer may view the product and let us know if any functionality had been missed out. Once the feedback from the customer is received, the software can be revised to meet their needs.

Conclusion:

We implemented weekly “Build and Smoke Testing” in addition to other tools and were able to deliver the projects with productivity improvement of 50%. The projects were also delivered with nil effort and schedule variance.

Appendix-A - Process for Weekly build and Smoke Test

The following details the process for Weekly build:

1. Clean out any possible old source files.
2. Move the current C++ code from VSS to Build PC. Get the log of the current version and move it to a file.
3. Compile, build C++ code on the Build PC.
4. On compilation, build output file is created with name, build_output_ddmmyyyy, where ddmmyy is the current date
5. Record any compile errors, write reports, and inform (email) module owners with compiler errors. Reports are built as follows:

i. After all the information from a Weekly build has been gathered, a Weekly Build Report is posted on the web. The Weekly build template contains

- Builder's name
- date as today's date,
- Build Number with a DDMMYYY for instance 31-01-2001 would be 31012001
- Included components, which is as per version control System output

ii. For every error, an entry to the table is made with name of the component, which reported the error and the error reported as per format given below:

Build Report for Project

BUILDER:		**DATE:**	
Build Number:		Build Status:	Success/Fail
Modules Included	Module Name		
Total Modules:			
Modules Successfully Built	Module Name	Developer Name	Log File
Modules Successfully built:			
Modules Failed	Module Name	Developer Name	Log File
Broken Modules:			

iii. For each error, which is there in into the build report, (Component name, Name of developer), use the email to email the owner of that module with the module name and error that broke the build.

iv. After the build report is complete, host it on the Project Web

6. The following lists the process for Smoke test:

i. Register the C++ components at Application Server and create msi file.

ii. Move the ASPs code to the virtual directory on Web Server. Install msi file at Web server.

iii. Record Rational test scripts on the Test machine

iv. Automated Rational Test script to be run from Test PC and the following smoke test report is produced.

Smoke Test Report

BUILD NUMBER:		**DATE:**	
Tester:			
Sl. No.	Test Case	Result	Reason
Total Cases:			
Total Failures:			
Total Warnings:			

The Author: Mohan Srinivasan is working as Senior Project Manager with ANZ Information Technology, Bangalore, India. He is responsible for the Information Technology requirements of Personal Financial Services of ANZ, Melbourne.

Mohan holds a Masters Degree in Industrial Engineering from National Productivity Council, New Delhi. He has around 15 years of experience in the software industry and has been managing projects in the areas of Mainframe, Client server and Web Technologies for clients in US, UK and Asia.

He is an ISO 9000 certified Lead Assessor and has been involved in CMM Level 4/5 assessments.

References:

1. Daily build – Rapid development and control, Swedish Engineering Industries 1999

ANZ Information Technology Private Limited (ANZIT), Bangalore, is a 100% fully owned subsidiary of the Australia and New Zealand Banking Group Ltd (ANZ), headquartered in Melbourne Australia. ANZ Information Technology is a leading provider of end-to-end solutions to ANZ Bank worldwide.

Our solutions have been homegrown within major banks and have delivered positive business results. Our insight into banking significantly minimizes the adoption and implementation risk associated with large-scale technology initiatives.

For more information, log on to http://www.anz-it.com/

Project Perfect is a project management software consulting and training organisation based in Sydney Australia. Their focus is to provide creative yet pragmatic solutions to Project Management issues.

Project Perfect sell "Project Administrator" software, which is a tool to assist organisations better manage project risks, issues, budgets, scope, documentation planning and scheduling. They also created a technique for gathering requirements called "Method H"™, and sell software to support the technique. For more information on Project tools or Project Management visit www.projectperfect.com.au