Houston Alt.NET Continuous Integration Workshop

September 19, 2009

Agenda

- Fowler's CI Principles (Michael Koby)
- Toolset Overview
- Source Control - SVN or Git (Ben Scheirman)
- Scripting Your Build - NAnt (Michael Koby)
- Build Server - TeamCity (Ben Scheirman)
- Putting it All Together (Ben Scheirman)

Martin Fowler's CI Principles

- What is Continuous Integration
- Practices of CI
  - Maintain Single Source Repository
    - Everyone Commits Daily
  - Automate the Build
    - Every Commit Should Build "trunk" on a Build Server
  - Make Build Self Testing

What is Continuous Integration?

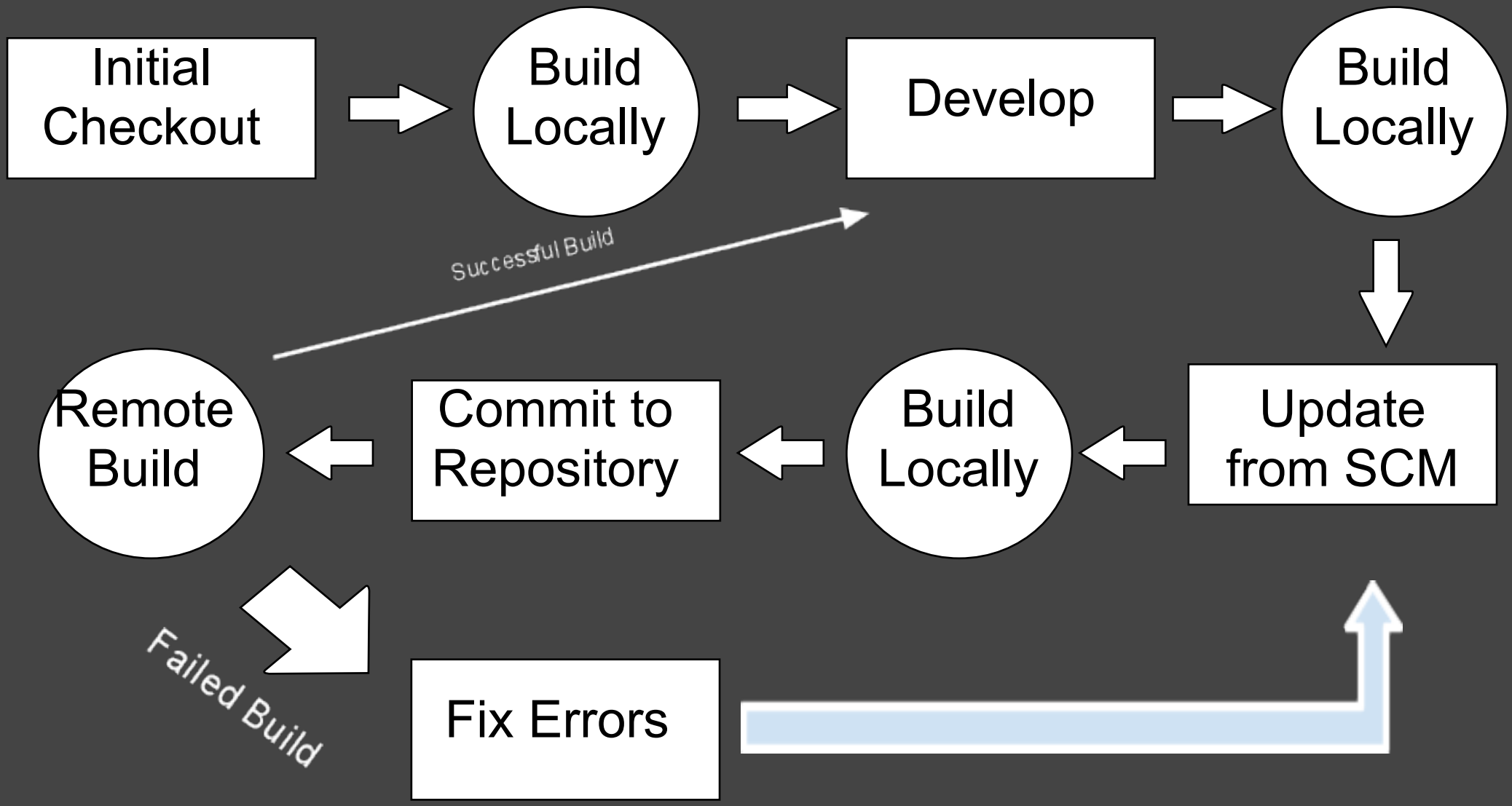
"Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible"

--Martin Fowler

Benefits of Continuous Integration

- Reduces Development Risks
- Gives a Better Understanding of Where You Are, What Works, & How Many Bugs are Left
- Find Developer Conflicts Sooner
- Increased Productivity
- Release Software that Contains Fewer Bugs and is More Stable

Continuous Integration Process



Maintain Single Source Repository

- Use a Source Control System
  - Team Foundation Server
  - Subversion
  - Git
- Everyone works off "trunk" ("master")
- Branches kept to a minimum (bug fixes, temp experiments, massive features/refactoring)
- Everything needed to build the project goes into source control
- IDEAL Setup: New developer should be able to check out from source control and build the project. (No asking "where is [insert filename here]"

Everyone Commits Daily

- Ideally many times per day
- Smaller commits are less risky
  - less chance of conflicts
  - conflicts are much easier to resolve

- Each commit is atomic
  - work on a feature, commit a small, logical chunk of file changes

- Do not commit code that breaks the build
  - Thou shalt not break the build!

Automate the Build or "F5 is not a Build Process"

- Ensure the build can be launched with a single command
- Automated build should Include everything
- Finished build result should be a complete running system on the local machine (or dev/test machine)
- Keep the build fast!
- Failed builds should be looked into immediately

What Should the Build Do?

A build can...

- setup local configuration
- create local database(s)
- run scripts to migrate the db to the latest schema
- generate code
- assign version # the project
- compile solution
- run unit tests
- run integration tests
- run acceptance tests
- generate documentation
- run static analysis (code coverage / naming standards)

Every Commit Should Build on Build Server

- Build server (CI Server) acts as a monitor to the repository
- New commits fire off a new build to ensure stability
- A developer isn't done with a commit/integration until their build is successful

Make Build Self-Testing

- Testing = Unit Tests
- Ensure your build runs unit tests across a large percentage of the code base (code coverage)
- To keep the build fast, one might have to segment tests (run different tests at different times during the build)
- A successful build is one in which all tests run without error

Today's Toolset

- Source Control - Git & Subversion
- Scripting - NAnt (will briefly see Rake, PSake, & Phantom)
- Unit Test - NUnit
- Build Server - TeamCity

Links to all tools mentioned today can be found on the "References" slide

Source Control (Subversion & Git)

Demo Time

Build Scripting (NAnt)

NAnt is a .NET version of the popular java build scripting utility known as Ant. These tools are very similar to MAKE, a popular build scripting tool found in the UNIX/Linux world.

NAnt in 120 Seconds

- XML based scripting
- Internals
  - Tasks
  - Functions
- Extensible
  - NAntContrib (http://nantcontrib.sourceforge.net)

NAnt Script Basics

The most basic NAnt script will include the following:

- XML Definition
- Project
- At least 1 Task (within the project)

That's the most basic of NAnt scripts!

NAnt Script Example

Leaving the presentation and will return in a moment...

Rake (hi mom!)

```
DOT_NET_PATH = "C:/Windows/Microsoft.NET/Framework/v3.5/"
NUNIT_PATH = "tools//nunit-2.4.6//nunit-console.exe"
PACKAGE_PATH = "build/package"

task :default => ["build:all"]

namespace :build do

  task :all => [:compile, :harvest, :test]

  desc "Use MSBuild to build the solution: '#{SOLUTION}'"
  task :compile do
    sh "#{DOT_NET_PATH}msbuild.exe /p:Configuration=#{CONFIG} #{SOLUTION}"
  end
```

Psake

PowerShell based

Written by James Kovacs

http://github.com/JamesKovacs/Psake

```
import System.IO

solution_file = "Phantom.sln"
configuration = "release"
test_assemblies = "src/Phantom.Tests/bin/${configuration}/Phantom.Tests.dll"

target default, (compile, test, deploy, package):
  pass

desc "Compiles the solution"
target compile:
  msbuild(solution_file, { @configuration: configuration })

desc "Executes tests"
target test:
  nunit(test_assemblies)

desc "Creates zip package"
target package:
  zip("build\\${configuration}", 'build\\Phantom.zip')
```

Phantom

Boo based

Boo is a .NET language that feels a lot like Python *(significant whitespace)*

http://github.com/JeremySkinner/Phantom

```
import System.IO

solution_file = "Phantom.sln"
configuration = "release"
test_assemblies = "src/Phantom.Tests/bin/${configuration}/Phantom.Tests.dll"

target default, (compile, test, deploy, package):
  pass

desc "Compiles the solution"
target compile:
  msbuild(solution_file, { @configuration: configuration })

desc "Executes tests"
target test:
  nunit(test_assemblies)

desc "Creates zip package"
target package:
  zip("build\\${configuration}", 'build\\Phantom.zip')
```

Build Server (TeamCity)

Putting it All Together

Resources

Continuous Integration Paper by Martin Fowler
http://martinfowler.com/articles/continuousIntegration.html

Today's Toolset

Git - http://www.git-scm.org/
Subversion - http://subversion.tigris.org/
NAnt - http://nant.sourceforge.net/
TeamCity - http://www.jetbrains.com/teamcity/index.html

Other CI Tools

CruiseControl.NET - http://ccnet.thoughtworks.com/
Rake - http://rake.rubyforge.org/
Phantom - http://github.com/JeremySkinner/Phantom
PSake - http://github.com/JamesKovacs/Psake