Developer Workspaces Enable Agile Teams

Steve Berczuk
Cyrus Innovation

New England Agile Bazaar
March 2008

About Me

- Software Developer
- Certified Scrum Master
- Author (SCM Patterns Book, CM Crossroads)
- Technical Lead, Cyrus Innovation (Boston)
- More Info
  - www.berczuk.com
  - www.scmpatterns.com
  - www.cyrusinnovation.com
  - steve@berczuk.com

About Cyrus Innovation

- Offices in Boston and New York City
- Software Development
  - Agile Teams
- Agile Coaching and Training

Common Problems

- Not Enough Process:
  - "Builds for me…"
  - "Works for me!"
  - "The build is broken again!"
  - "What branch do I use?"
- Process Gets in the Way:
  - Long Commit Times
  - Serialized Commits
  - Code Freezes
- Long integration times at end of project
  - "Fixing it" in integration
- Silos of Knowledge
  - "How does this code work?"

Agile Software Development

- What it is
- Why you care

Agile Manifesto www.agilemanifesto.org

- *Individuals and Interactions* **over** Processes and Tools
- *Working Software* **over** Comprehensive Documentation
- *Customer Collaboration* **over** Contract Negotiation
- *Responding to Change* **over** Following a Plan

- People build software!
- Use the right tools and processes.
- Focus on things that add direct value.
- Adapt to change; acknowledge that change happens.
- (Common Sense Applied)

Benefits of Agile Methods

- Easier to manage scope
- Build the right thing
- Deliver value more predictably

- Agile methods
  - Emphasize feedback and communication.
  - Avoid process steps that don't add value.
  - Address issues, don't just add processes for comfort.

SCM

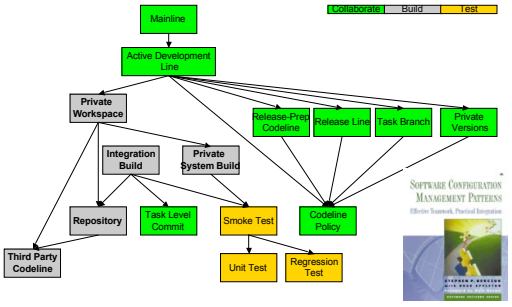
- SCM enables agility
  - Reproducible Workspaces
  - Feedback through builds*
- Concepts in context:
  - Branches, labels, tags
  - Builds
  - Workspaces
- Different levels of scale

Agile Context

- SCM is Part of the Puzzle:
  - Architecture
  - Software Configuration Management
  - QA/Testing
  - Culture/Organization

The Goal: Working software that delivers value.

The SCM Pattern Language



What is a Workspace?

- Everything you need to code and test.
- Includes:
  - Source
  - Databases
  - Typical Data

Role of Workspaces in Agile Teams

- Work Quickly and Independently
  - But don't interrupt anyone else
- Collective Ownership
  - Get started quickly
- Feedback
  - Create similar environments for developers, testers, integration

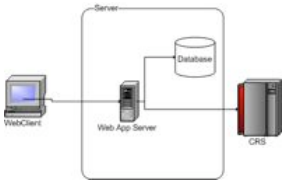
Private Workspace

- You want to support an *Active Development Line*.
- **How do you keep current with a dynamic codeline and also make progress without being distracted by your environment changing from beneath you?**

Private Workspace

- Create a *Private Workspace* that contains everything you need to build a working system.
  - You control when you get updates.
- Before integrating your changes:
  - Update your workspace.
  - Build your workspace and Test your code and the system. (Private System Build)
- (Defer additional validations to the Integration Build)
- Have an automated way to create workspaces from a repository. (Repository)

Private Workspace Example



- Workspace
  - App Server
  - Database Schema
  - Code for Web App
  - Test CRS Login
  - (Build/Deploy and Configuration Tools & Scripts)

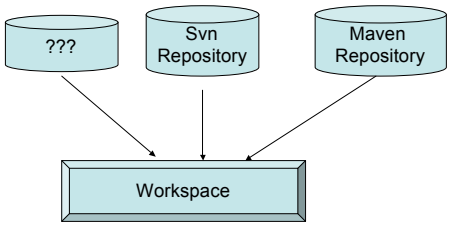
Repository

- *Private Workspace* and *Integration Build* need components.
- **How do you get the right versions of the right components into a new workspace?**

Repository (Solution)

- Have a single point of access for everything.
  - Use this mechanism at all levels (dev, integration build, etc)
  - No hard coded information.
- Have a mechanism to support easily getting things from the *Repository*.
  - Install Version Manager Client
  - Get Project from Version Management
  - Build, Deploy, Configure (Ant target, Maven goal)
  - Simple, repeatable process.
- Still to do:
  - Manage external components: *Third Party Codeline*

Repository



Creating Workspaces

- Simple and Automated
  - Install SCM Client and Build Tool
  - Checkout project file
  - Run "workspace" target
    - Gets files and builds
- Tools
  - Ant, Maven
  - Scripts
- Factor specifics into configuration
  - No "hard coding"

Private Workspace + Repository

- Add a new developer quickly.
- Create test environments.
- Create build environments.
- Reproduce problems quickly.
- Have an implicit check for inflexible configurations.

Private System Build

- You need to build to test what is in your *Private Workspace*.
- **How do you verify that your changes do not break the system before you commit them to the *Repository*?**

Private System Build (Forces)

- Developer Workspaces have different requirements than the system integration workspace.
- The system build can be complicated.
- Checking things in that break the *Integration Build* is bad.

Private System Build (Solution)

- Build the system using the same mechanisms as the central integration build, a *Private System Build*.
  - This mechanism should match the integration build.
  - Do this before checking in changes!
  - Update to the codeline head before a build.
- Unresolved:
  - Testing what you built: *Smoke Test*

Integration Build

- What is done in a *Private Workspace* must be shared with the world.
- **How do you make sure that the code base always builds reliably?**

Integration Build (Forces)

- People do work independently.
- *Private System Build*s are a way to check the build.
- Building everything may take a long time.
- You want to ensure that what is checked-in works.

Integration Build (Solution)

- Do a centralized build for the entire code base.
  - Use automated tools: Cruise Control, SCM tool Triggers, etc
- Still Unresolved:
  - Testing that the product of the build still works: *Smoke Test*
  - Build products may need to be available for clients to check out
  - Figure out what broke a build: *Task Level Commit*

… + Build and Test Patterns

- Enable rapid change
- Reduce risk for broken builds
- Debug deployment process

Creating an Agile SCM Environment

- Decide on a goal.
- Choose an appropriate Codeline Structure
  - set up the related policy.
- Create a process to set up workspaces
  - Private
  - Integration
- Build & Deploy is an Iteration 0 Story.
- Integrate frequently at all levels
  - Developer Workspace
  - Integration Build
- Deploy frequently.
- Test.

Agile Results

- More frequent Deliveries
- Fewer Surprises
- Happier Clients