Source Control

Brian Caufield
CFUGCNY
10-14-08

Agenda

- What is it?
- Why Should I use it?
- How can I use it in my development process?
- What are my options?

Source Control - What is it?

Evolved from engineering development, primary function is to be able to return to any point in the development process.

- Source Repository
  - The server based file storage system
- Local repository
  - Working Copy on the developers system

Source Control - Why should I use it?

- Multiple versions running simultaneously
  Lets you revert to an explicit version of a project, for web development this is primarily used for dev, staging and production servers.

- Team Development
  Source control allows the exact and current version of the software to be available to all members of the team.

- Accountability/Transparency
  "Dif" tools allow for a quick way to evaluate changes to code

What kinds of Source Control are there?

- File Locking
  Only one developer can work on a file at a given time
  Issues; files left locked for too long, forgetting to checkin blocks other developers, no oversight till check-in.
  Encourages by passing the lock and leads to a synchronization mess.

- Version Merging
  Concurrent Access - Multiple developers can work on the same file simultaneously, any conflicts have to be merged.

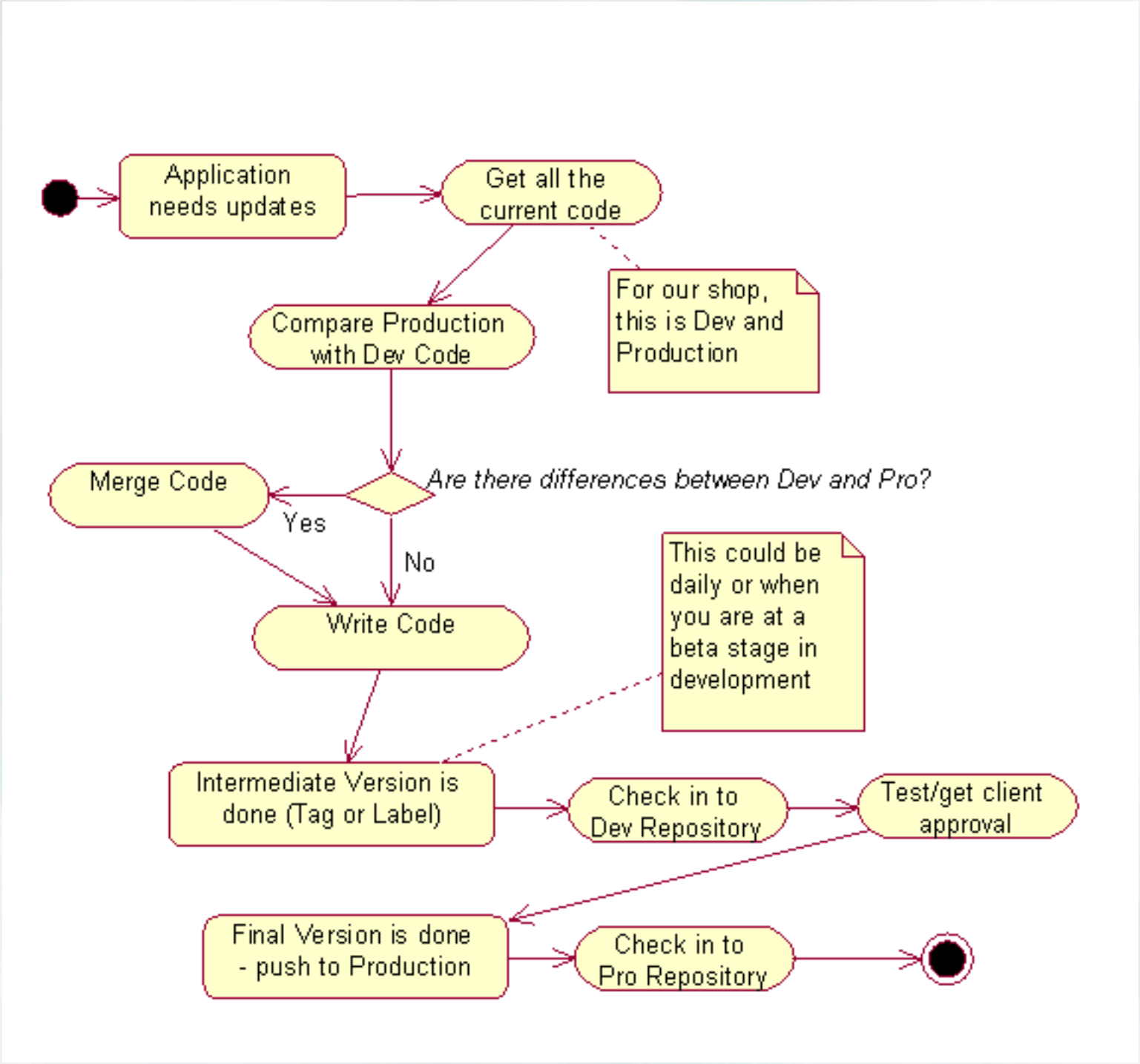
How can I use it in my development practice?

- Projects have three branches: Dev, Staging and Production.
  - All Projects are local (on your harddrive) and checked-out from Source control at the beginning of a project, or every day if you are collaborating in a team.
- Compare the different versions
  - Merge as needed
- Check in code when you reach milestones
  - Dev code should be checked in regularly and should work.

Source Control Practices (continued)

- When Dev code reaches a deployment target, it branches to the Staging line of Source Control.
  - Always Compare when pushing to another server
- After Staging passes all QC, it moves to the Production line and all changes from Staging are merged back into Dev.

(See Activity Diagram)


Application needs updates
Get all the current code
For our shop, this is Dev and Production
Compare Production with Dev Code
Are there differences between Dev and Pro?
Merge Code
Yes
No
Write Code
This could be daily or when you are at a beta stage in development
Intermediate Version is done (Tag or Label)
Check in to Dev Repository
Test/get client approval
Final Version is done - push to Production
Check in to Pro Repository

Glossary

- Baseline
  An approved revision of a document or source file from which subsequent changes can be made. See the discussion of baselines, labels, and tags.
- Branch
  A set of files under version control may be branched or forked at a point in time so that, from that time forward, two copies of those files may be developed at different speeds or in different ways independently of the other.
- Change
  A change (or diff, or delta) represents a specific modification to a document under version control. The granularity of the modification considered a change varies between version control systems.

Glossary (Continued)

- Change list
  On many version control systems with atomic multi-change commits, a changelist, change set, or patch identifies the set of changes made in a single commit. This can also represent a sequential view of the source code, allowing source to be examined as of any particular changelist ID.
- Checkout
  A check-out (or checkout or co) creates a local working copy from the repository. Either a specific revision is specified, or the latest is obtained.
- Commit
  A commit (checkin, ci or, more rarely, install, submit or record) occurs when a copy of the changes made to the working copy is written or merged into the repository.

Glossary (Continued)

- Conflict
  A conflict occurs when two changes are made by different parties to the same document, and the system is unable to reconcile the changes. A user must resolve the conflict by combining the changes, or by selecting one change in favour of the other.
- Dynamic stream
  A stream (a data structure that implements a configuration of the elements in a particular repository) whose configuration changes over time, with new versions promoted from child workspaces and/or from other dynamic streams. It also inherits versions from its parent stream.
- Export
  An export is similar to a check-out except that it creates a clean directory tree without the version control metadata used in a working copy. Often used prior to publishing the contents.

The most recent commit.

Glossary (Continued)

- Head
  The most recent commit.
- Import
  An import is the action of copying a local directory tree (that is not currently a working copy) into the repository for the first time.
- Label
  See tag.
- Mainline
  Similar to Trunk, but there can be a Mainline for each branch.
- Merge
  A merge or integration brings together two sets of changes to a file or set of files into a unified revision of that file or files.

Glossary (Continued)

- Repository
  The repository is where the current and historical file data is stored, often on a server. Sometimes also called a depot (e.g. with SVK, AccuRev and Perforce).
- Resolve
  The act of user intervention to address a conflict between different changes to the same document.
- Revision
  Also version: A version is any change in form. In SVK, a Revision is the state at a point in time of the entire tree in the repository.
- Tag
  A tag or label refers to an important snapshot in time, consistent across many files.

Glossary (Continued)

- Trunk
  The unique line of development that is not a branch (sometimes also called Baseline or Mainline)
- Update
  An update (or sync) merges changes that have been made in the repository (e.g. by other people) into the local working copy.
- Working copy
  The working copy is the local copy of files from a repository, at a specific time or revision. All work done to the files in a repository is initially done on a working copy, hence the name. Conceptually, it is a sandbox.