Perforce Branching
Moving Fast from Theory to Practical Application

Overview

1. Introduction
2. Branch Strategy Basics
   - The Mainline Model
   - Planned and Organic Release Processes
   - More than one MAIN?
3. Directory Structure Considerations
   - How Many Depots?
   - Products, Product Families, and Projects
   - Branch Container Directories
Overview

4. Interview: Release Process Classification
   - Planned vs. Organic?
   - Simple Patches or Complex Maintenance?
   - Hosted, Licensed, or Burn & Ship?
   - Generic Product or Custom?
   - Scale? Large-scale, globally distributed?
   - Sophistication?

5. Sample Case Studies

6. Q&A

Mainline Model Basics

- Recognize the need for temporary isolation of codelines
- Provide a clear path for integration of isolated codelines.
- Recognize the need for temporary divergence.
- Reduce excessive permutations of the code base.
Integration Types - Refresh

(parenthetical comments)

A Refresh:
- is intended to integrate changes in one codeline with changes made in other codelines.
- is an integration from a more formal to a less formal codeline.
- requires potentially complex merge work.
- can introduce instability in the target codeline.
- is best performed by an SME familiar with the software, coding language, requirements, etc.
- is often done as a retail operation, e.g. by subsystem or areas of subject matter expertise.
Integration Types - Promotion

A Promotion:
• is intended to promote exact copies of tested, trusted software closer to Production.
• is an integration from a less formal to a more formal codeline.
• does not require complex resolve because the files are promoted as they are (“copy merged”).
• can be performed as a wholesale operation by a centralized Configuration Management or Release Engineering team who may be unfamiliar with the software.
• promotes the entire codeline from a known state as it meets ever-increasing quality bars for each level of promotion.

Integration Types - Selective

A Selective Integration:
• is intended to “cherry pick” selected changes from a codeline, such as extracting a generic bug fix from a codeline normally used for custom development.
Directory Structure Notes

- More than one MAIN? Sure!
- Product Families
- *Product vs. Project*
- Branch Container Directories
- //Eng/MAIN/ProductX or //Eng/ProductX/MAIN

Version Control Areas (1 of 3)

- **Source Area**
  Contains originally contributed artifacts, such as source code, *.jpg images, etc. Branching Strategies apply mostly within the Source Area.

- **Build Area**
  A "no humans allowed" area, populated strictly by fully automated build processes. Contains compiled executables, libraries, APIs, packages, installers, etc.
Version Control Areas (2 of 3)

✦ Release Area
Tracks exact "As Delivered" files. Contains files branched from Build Area, plus files specific to particular release configuration, e.g. files that vary on a per-runtime environment basis or per-customer basis (even if they use the same build).

✦ Admin Area
Contains spec depot, remote depots, Perforce management scripts, etc.

Version Control Areas (3 of 3)

✦ Import Area
Contains files and histories exactly as they were imported from other version control systems (CVS, VSS, Subversion, ClearCase, etc.). Files are read-only.

✦ Doc Area
Often published by web server, contains various documents (other than those that live in the source area).
Typical Depots (1 of 2)

- //Giz – Source Code for Giz Product Family

- //Giz-Build – Build area, populated only by fully automated build processes (no humans allowed). Contains various build configurations, e.g. 32/64 bit, debug/optimized, or Windows/Mac/Linux/Solaris.

- //Giz-Release – Contains as-released software, suitable for distribution to runtime environments, burning to CDs or firmware, or otherwise delivered.

Typical Depots (2 of 2)

- //3rdParty – Contains Commercial 3rd Party Software, with an optional branching structure to support local modifications.

- //OpenSource – Segregate all open source code used in your software, to promote re-use and simplify “black duck” analysis (analysis of potential legal liabilities introduced by inappropriate use of open source).
Interview Questions (1 of 6)

1. What best describes the primary development/release cycle?
   - Planned
   - Hyper
   - Short
   - Nominal
   - Long
   - Organic

Interview Questions (2 of 6)

2. Classify your Maintenance Requirements
   - None (e.g. Hosted)
   - Simple: Minimal maintenance of released products; the product structure isn’t expected to change appreciably in maintenance
   - Complex: Extensive, large scale development effort is focused on support of released products, which could take years.
Interview Questions (3 of 6)

3. What best describes the deployment model of your product:
   • Hosted: No need to support old releases – your clients run whatever software versions are running in the data center.
   • Licensed Software Product: You need to support customers on multiple releases of your software.
   • Burn & Ship: Major releases are shipped (e.g. burned into firmware or CDs). Patches may be required to shipped software.

Interview Questions (4 of 6)

4. Are all changes generic, or is there any need to support customizations?

5. If customization is required, can it be assumed that any given customer will be on exactly one version?
   • Simple: Yes, any given customer will have exactly one version.
   • No: We need to account for the possibility that a specific customer might use different versions of our product simultaneously (e.g. one version in their Production environment, another in their Training environment, yet another in an Evaluation environment, etc.).
Interview Questions (5 of 6)

6. How many developers/contributors are involved? How many geographic sites are involved? Is there (or are you trying to form) a formal QA organization?

Interview Questions (6 of 6)

7. Do users want Personal Development Branches (aka Sandboxes)?
8. Do you want Per-Bug branches?
Case Study #1: Overview

- Licensed software, large globally distributed development team, with formal QA
- Release Process Characteristics
  - Planned Releases
  - Large, Multi-Site Teams
  - Simple Maintenance
  - No Customization Support – Generic Product Only
  - Personal Development branches used sparingly

Case Study #1: Dir Structure

```
//Eng
Rel/<PROJECT>-R/[<ProductFamily>]/<Product>/...
MAIN/[<ProductFamily>]/<Product>/...
Int/<PROJECT>-Int/[<ProductFamily>]/<Product>/...
Dev/<PROJECT>/[<ProductFamily>]/<Product>/...
PD/<User>/<PROJECT>/[<ProductFamily>]/<Product>/...
```
Case Study #1: Branch Diagram

![Branch Diagram](image_url)

Case Study #1: Dir Diagram

![Dir Diagram](image_url)
Case Study #1: Branch Specs

<table>
<thead>
<tr>
<th>Branch Type</th>
<th>Branch Spec</th>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personal</td>
<td>PD_juser_GIZ-2.0.0.B</td>
<td>//Eng/Dev/GIZ-2.0/Giz/…</td>
<td>//Eng/PD_juser/GIZ-2.0/Giz/…</td>
</tr>
<tr>
<td>Rel</td>
<td>GIZ-1.0-R.B</td>
<td>//Eng/Rel/GIZ-1.0-R/Giz/…</td>
<td>//Eng/MAIN/Giz/…</td>
</tr>
</tbody>
</table>

Note that the default direction is always for a Refresh.

Case Study #2: Overview

- Embedded software, small development team
- Release Process Characteristics
  - Planned Releases
  - Simple Maintenance
  - Customization Support
**Case Study #2: Dir Structure**

```plaintext
//Eng
Custom/<CUSTOMER>-C/[<ProductFamily>]/<Product>/...
Rel/<PROJECT>-R/[<ProductFamily>]/<Product>/...
MAIN/[<ProductFamily>]/<Product>/...
Dev/<PROJECT>//[<ProductFamily>]/<Product>/...
```

**Case Study #2: Branch Diagram**

![Branch Diagram](image)

Legend
- Simple Promotion does not apply for Customization
- Selective Integration "Cherry Picking"

Development
- 5.5 New Features & Fixes

Release
- 5.5 Performance
- 5.0-R (Release)
- 5.5-B (Beta)
- 5.5-R (Rel)

Navy-C
- (Custom)
Case Study #2: Dir Diagram

Case Study #3: Overview

- Hosted Model
- Release Process Characteristics
  - Organic and Planned Release Processes
  - Small Development Team
  - No Customization
  - No Maintenance of old releases
Case Study #3: Dir Structure

//Eng/
PROD/<HostedApp>/...
MAIN/<HostedApp>/...
ODEV/<HostedApp>/...
Dev/<PROJEC>/<HostedApp>/...

Case Study #3: Branch Diagram
Case Study #3: Dir Diagram

Case Study #4: Overview

- Consulting Model
- Release Process Characteristics
  - Organic Release Process
  - No “Production” environment for generic product
  - Extensive Custom Development
  - No formal QA
  - Small Development Team
Case Study #4: Dir Structure

```plaintext
//Eng
Custom/<CUSTOMER>-C/<Product>/...
MAIN/<Product>/...
DEV/<Product>/...
```

Case Study #4: Branch Diagram
Case Study #4: Dir Diagram

Questions?
Thank You!

Perforce Branching
Moving Fast from Theory to Practical Application