Code delivery management across geographically distributed locations

Salvin Michael
Infosys Technologies Ltd
Software Configuration Management Analyst
salvin_michael@infosys.com
Highlights

- Changing Business Scenario
- Challenges posed by the changing scenario
- Geographically distributed parallel development
- Challenges of parallel development in a geographically distributed environment
- Selection of Rational Tools
- Using ClearQuest for defect tracking & UCM integration
- VOB and code stream design to support parallel development
- Managing daily code deliveries and rebases
- Infosys best practices
- Q & A
Changing Business Scenario

- Parallel development by several developer teams located in multiple geographic locations across continents handling concurrent multiple releases.
Challenges posed by the changing scenario

- Orchestrating concurrent development at multiple locations
- Orchestrating concurrent development of multiple releases
- Implementing processes that enable developers to focus on actual development
- Reduce errors caused by building products with the wrong version or some old code which did not include a current fix
- Problems due to incorrect versioning and tracing of the versioning issue
- Complicated build and release process
- Reduce resource dependency
- Faithful reproduction of a past version of software in order to investigate reported problems
Geographically Distributed Parallel Development

This involves

- Combination of Major Releases and Service Packs that use the same version control tool
- Developers located at geographically separated sites, sharing the same repository and same set of applications
- Development/Maintenance happens on same set of files from multiple locations
- Each site develops one or more subcomponents of a large software system
Challenges of parallel development in a geographically-distributed environment

- UK
- India

SCM Infrastructure Management across Sites (i.e. Network, Server, Clients & Software)

- Code synchronization issues between various sites
- Code Management between different sites – Stream strategy, Code Merging
- Communication and collaboration between different sites
- Time zone differences
Rational Tool – ClearCase

IBM Rational® ClearCase®

- Main features
  - Integrated version control
  - Parallel development support
  - Baseline management
- Accessible through local, remote and web interfaces, and leading IDEs including IBM Rational Application Developer
- Easy code reconciliation

ClearCase Multisite

- Extends software configuration management across geographically distributed projects through repository replication
- Enables access to latest information regardless of location, through automatic replication and synchronization
- Developers at different locations can use the same (VOB)
- Each location (site) has its own copy (replica) of the VOB
- Mastership
  - Certain objects are assigned a master replica (or master).
  - In general, an object can be modified or deleted only at its master replica location
Rational Tool - ClearQuest

- IBM Rational® ClearQuest® enables
  - Defect and change tracking across the application life cycle
  - Activity-based change management
  - Integration with IBM Rational ClearCase®
  - Local, Web and IDE access
  - Scalability from small workgroups to geographically distributed enterprises
  - Customizable querying and reporting
  - Developing a compliance-driven framework
    - Flexible workflow management
    - Audit trails and traceability
    - Access control
    - Control of the software development process.
    - Helps to automate and enforce development processes.
Flexible workflow management

- **Release Record** (Release No., Release Type etc)
- **Submit**
- **Incident Record** (Child)
- **Duplicate**
- **Declined**
- **Duplicate**
- **Decline**
- **Re-Analyze**
- **Approved**
- **Slotted**
- **Open**
- **Resolved**
- **Test**
- **Close**
- **Re-Open**
- **Submit**
- **Re-Analyze**
- **Duplicate**
- **Decline**
- **Work Request** (Child)
- **Work Request** (Update Code)
- **Close**
Typical Life Cycle - ClearQuest Integration with UCM

1. **Project Manager Creates Activities in ClearQuest (Define a work Effort)**
   - Assign Activities

2. **Developers**
   - Work on Activities
   - Check-in/Check-out file in ClearCase using RAD
   - The 'Change Set' captures the elements and versions which changed.

3. **Tech Lead**: Deliver the change to Int stream.

4. **Integration Stream**
   - Baseline BL1
   - The development streams are refreshed with the changes contained in the new baseline, 'rebasing' their stream. Re-sets the 'foundation' baseline.
   - After the testing of the changes is complete. A baseline is created that captures the changes delivered by the developers.
## Selection of Rational Tools

<table>
<thead>
<tr>
<th>Function</th>
<th>Available Choices</th>
<th>The Winner &amp; Why</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote Java Development in ClearQuest (CQ) enabled UCM VOBs</td>
<td>CCRC vs. Local Client with Multisited VOBs</td>
<td>Local Client w/ MultiSited VOBs – Latency with CQ enabled VOBs would be too slow.</td>
</tr>
<tr>
<td>Remote access to Base ClearCase documentation</td>
<td>CCWeb vs. Local Client with Multisited VOBs</td>
<td>CCWeb – Saves on bandwidth usage, lower maintenance of MultiSite, volume of use is low, documentation VOBs are huge</td>
</tr>
<tr>
<td>Remote ClearQuest Database Access</td>
<td>Local SQL DB in each embassy or use single centralized SQL DB</td>
<td>Local SQL DB – Heavy use of ClearQuest by developers using UCM and by testers logging defects with large attachments</td>
</tr>
<tr>
<td>IDE for Java (Integrated Development Environment)</td>
<td>RAD6 vs. Eclipse</td>
<td>RAD6 wins because it is tightly integrated with ClearCase for UCM Code Development</td>
</tr>
</tbody>
</table>
Case Study:
Multiple Locations Multiple Release Stream Strategy
Project VOB and Components

- Project_PVOB
  - Project1.0
    - Integration (INT)
    - Development (DEV)
    - Remote_UK
    - Remote_INDIA
    - Service Pack 1
    - Service Pack 2
  - Components
    - Project_Component1
    - Project_Component1
    - Project_Component1
Parallel Stream Strategy – Basic Overview

Major Release Project

- Integration (INT)
- Development (Dev)
- Remote Stream (U.K.)
- Remote Stream (India)
- Service Pack 1
- Service Pack 2
Parallel Stream strategy – Major Release Structure

Release 1.0
Integration (Int)
Development (Dev)
Remote Dev – U.K
Remote Dev - Mysore
Service Pack 1 (SP1)
Remote SP 1 - Mysore
Remote SP 1 - Bangalore
Release 2.0
Seed new project
Baseline Creation
Baseline Creation
Used only for merges
- Delivery
- Rebase
Integration (Int)
Code Merge Process – Master Location

- Tech lead in master replica will create a baseline on the stream and submit a request to get changes from the parent Dev to the child Service Pack stream.
- Code merge team will process the request (rebase) with the specified baseline and communicate the code merge artifacts to the appropriate teams.
- Development team will validate the code merges performed by code merge team and make the final baseline.
- Remote tech leads will submit a rebase request for its subsequent child-streams.
Remote code merge team will process the requested Rebase with the specified baseline and communicate the code merge artifacts to the offshore development team.

Remote development team will validate the code merges performed by code merge team and proceed to work. Remote tech leads make the final baseline end of the day.

Remote tech leads will submit a delivery request to deliver offshore changes to the respective parent onsite stream and the remote code merge team executes the request.

This process will keep both master and remote streams in sync all the time.
SCM – Infosys Best Practice

CPD
(Cross Project Delivery)

Trunk Project

Baseline

Release 2

Dev

INT

SP1(Rel2)
SP2(Rel2)
SP3(Rel2)
SP4(Rel2)
SP5(Rel2)
SP6(Rel2)
Temp stream for Rel 3
Temp stream for Rel 4

Release 1

Dev

INT

SP1(Rel1)
SP2(Rel1)
SP3(Rel1)
SP4(Rel1)
SP5(Rel1)
SP6(Rel1)
Temp stream for Rel 2

Delivery

Rebase
Appendix

- Stream Strategy: UCM Lite
Stream Strategy: UCM Lite

- Development streams are created by administrators
- Developers create local snapshot views on these streams
- Each Major Release (monthly cycle) gets its own UCM Project
- INT stream is the default stream that gets created by ClearCase
- Dev stream is created as a child stream of INT to do the major release development
- Remote Dev streams ( mastered at remote sites) are child streams of Dev streams
- Remote Streams do a daily Rebase and Delivery to take changes and deliver changes to the Dev (parent) stream
- Service Pack streams (SP streams) have a lifespan of 1 week and are spawned as a child of the Dev streams. SP streams may have remote streams
- Next SP stream is then spawned from the previous SP stream and so on…
- When the development effort is complete for a major release, the resulting baseline is delivered to the application’s Trunk project, for use in rebasing subsequent projects and is also used to seed subsequent projects
Glossary

- **VOB**
  - A Versioned Object Base (VOB) is a data repository used by ClearCase for storing versions of file elements, versions of directory elements, and metadata associated with these objects.

- **Project VOB (PVOB)**
  - A VOB that stores UCM items, such as projects, streams, activities, and change sets. Every UCM project must have a PVOB. Multiple projects can share the same PVOB, but ClearCase components cannot cross PVOB boundaries.

- **Component VOB (CVOB)**
  - Each PVOB has a ClearCase object that you use to group a set of related directory and file elements within a UCM project. A Versioned Object Base of this Clearcase object is called a Component VOB (CVOB). Typically, the elements that make up a component are developed, integrated, and released together. A project must contain at least one component, and it can contain multiple components. Projects can share components.
Glossary (Contd.)

- **Activity**
  - An *activity* records the set of files that a developer creates or modifies to complete a development task, such as fixing a bug.

- **Stream**
  - A *stream* is a ClearCase object that maintains a list of activities and baselines and determines which versions of elements appear in your view. (Think branch)

- **Baseline**
  - A *baseline* is a version of one or more components.

- **Rebase**
  - Merges files and directories from the integration stream or feature-specific development stream to their development streams.

- **Deliver**
  - Merges work from the developer’s private work area to the project’s shared work area.
Questions & Answers
THANK YOU