



Dave Holscher

International Truck and Engine

Joe Batt

Solid Design





## Project Overview

- Diamond Logic Builder (DLB)
  - Thick application for engineers and mechanics to configure, program, test, and simulate variety of truck controllers.
- Hyperion
  - Web based tool for engineering to record electronics designs
  - Database and configurator used by DLB



# Development Process

- Track bugs/enhancement/requirements
- Organize next round of changes into a deliverable
- Code the changes
  - Submit after developer has tested
  - Changes automatically picked up and full test is run
- Release to business for acceptance test



# Development Process

- Release production quality release if acceptable
  - Frequent releases to internal engineering, less frequent releases to external customers.
- Start over with next round regardless if release is accepted



## Development Process

- Short cycles, lots of versions
  - Fast feature growth, less predictable
  - Easy to fix bugs, easy to add bugs, but not all versions reach external customers
- How? Continuous Integration and tools like CruiseControl, JUnit, Ant, etc.



# Continuous Integration

- Definitions

- See

- <http://www.martinfowler.com/articles/continuousIntegration.html>

- <http://c2.com/cgi/wiki?ContinuousIntegration>

- <http://www.xprogramming.com/xpmag/whatisxp.htm#continuous>



# Continuous Integration

- What it means to us
  - Software version control
  - Automated build kick-off
  - Automated builds, testing and deployment
  - Repeatability
  - Fault tolerance



# Development Environment

- Common, well established tools
  - Provide broad base of support
  - Continually improved by the community



# Development Environment

- Using CVSNT
  - Supports NT authentication (client and server)
  - Supports pserver and ext (ssh) protocols
  - See <http://www.cvsnt.com/>
  - Tortoise CVS
    - See <http://www.tortoisecvs.org/>
    - Integrates into Windows shell, works great with CVSNT



# Development Environment

- Automation using Ant
  - How to set it up
  - See <http://ant.apache.org/>
- Using JUnit
  - How to set it up and use it
  - See <http://www.junit.org/>



# Development Environment

- Using JFCUnit
  - How to set it up
  - See <http://jfcunit.sourceforge.net/>
- Database for testing
  - Scripting, using CVS
- Using Ant to generate a build number



## CruiseControl Builds

- <http://cruisecontrol.sourceforge.net/>
- Install is labor intensive, but only done once
- Runs as a service
- Uses Ant to build and run tests
- XML result logs interpreted by J2EE web application for reporting
- Email notifications



## Acceptance Testing & Deployment

- Automating deployment is next step after automating builds
- Development, Test, Production deployment targets in Ant
- Automated deployment documents the process that is often not documented.



## Acceptance Testing & Deployment

- Tag releases when they go functional users for testing
  - Using the build number to identify tag
- Run automated deployment against production against tested tag
  - Try to deploy from the same directory where test version was built



## Automating Web Start

- Use Ant tasks to automate detecting library versions as much as possible
  - Where to look
    - Package version
    - Properties file
    - Method that returns a version
    - Class that prints the version when run
  - What to do if library doesn't have a version



# Automating Web Start

- Using Ant to construct the JNLP app
- Triggering JNLP servlet to refresh version info



## Team Communication

- Tools will not replace communication but can keep everyone aware
- CVS mail notifications via CVSMailer for watching changes
  - How to set it up
  - See <http://web.telia.com/~u86216121/cvsmailer/CVSMailer.html>



# Team Communication

- CruiseControl mail notifications for watching builds
- ViewCVS for making it easy for everyone to see
  - How to integrate with mail notifications



# Gotchas

- JFCUnit
  - Doesn't work headless, need dedicated system.
  - Timing issues
    - Individual tests that re-run themselves.
    - More threads, more issues.
  - Some code changes can simplify testing.



# Gotchas

- CVSNT
  - Uses a lot of CPU cycles, put it on a dedicated server if possible
  - Unix/Linux CVS and SSH more mature
  - No change lists



## Gotchas

- Automated determination of library versions in Ant
  - In some cases a new version of jar must be get to the client even though the version hasn't changed
    - e.g. Getting a new signing certificate
  - What to do



# Tips

- Know your version control
  - Don't be afraid to branch
  - Use tags
- A good CVS client can save a lot of time
  - Tortoise CVS is great for novices
  - Eclipse is great for Java or C++
  - Others?



## Tips

- Don't check in build artifacts that can be regenerated
- Do check in any third party libraries
  - We even check in Eclipse so everyone's configuration matches



# Tips

- Automate everything
  - Formalizing the process is worth the effort
  - Documents the process
- Something will go wrong with your automated deployment
  - Nothing is perfect, have an expert on call



# Tips

- Get a bug tracking system
  - Track everything
    - Issue, bug, clarification, etc.
  - Encourage everyone to read and participate in updating.
  - Know what you are releasing.