Scalable Life Cycle Management via Perforce

Adam Breashears and Valerie Hendrickson
NYSE Euronext
June 1, 2011
Agenda

- The Business Needs of an Exchange
- Business Challenges in SDLC Methodologies
- Perforce Environment Setup
- Release Installation Process
- Automated Production Deployment
- Automated Rollback Scenarios
- Life Cycle Management
- Conclusion
- Appendix
Business Makeup
NYSE Euronext Profile

- NYSE Euronext (NYX) is the world’s most diverse exchange group, offering a broad and growing array of financial products and services in Europe and the United States that include cash equities, futures, options, exchange-traded products, bonds, market data, and commercial technology solutions.
- With over 8,000 listed issues globally, NYSE Euronext's equities markets -- the New York Stock Exchange, Euronext, NYSE Arca, and NYSE Amex -- represent nearly 40% of the world's cash equities trading volume, the most liquidity of any global exchange group.
- NYSE Euronext also operates NYSE Liffe, the leading European derivatives business and the world’s second largest derivatives business by value of trading. NYSE Euronext is part of the S&P 500 index and the only exchange operator in the S&P 100 index.
Scale of the Business

The total market cap of all our listed companies is over **$24 Trillion**....

...Compared to country GDPs, that would make us the largest economy in the world.

**Largest Economies by GDP (in Trillions of $)**

- **US** - $15 tr
- **China** - $10 tr
- **Japan** - $4 tr
- **India** - $4 tr
- **Germany** - $3 tr
- **$ 24 tr – NYX Issuer Mkt Cap**
Scale of the Transactions

Globally, we execute $2.5 \text{ trillion} \text{ in financial transactions every day} \ldots \text{.}

\ldots \text{That’s } 42 \text{ times more than Ebay handles in a year.}

Monetary Value of Transactions (in Billions of $)

NYX daily transactions

Annual Ebay transactions
Volume of the Transactions

In the US alone, we process more than 1.4 billion messages every day...

...That’s 3.5 times more than the number of internet searches Google handles daily.

Number of Messages/Transactions (in Millions)

- NYX daily messages
- Daily Google searches
- Visa daily credit card transactions
400+ years of business and technology firsts

We are the inventors of financial exchanges…

- Amsterdam, founded 1602, is the world’s oldest exchange
- Our markets in Brussels, Lisbon, NY and Paris and trace their roots to the 18th and 19th centuries
- We have over 16 centuries of combined experience running financial exchanges

And we have a rich history of innovation in markets…

- NYSE was the first exchange to use a stock ticker to disseminate data (1867)
- NYSE was the first to have telephones on the trading floor (1878)
- We had automated quotation services 18 years before Nasdaq was founded (1953)
- First electronic ticker display board (1966)
- First options exchange in Europe launched in Amsterdam (1978)
- Amex pioneered the Exchange Traded Fund (1993)
- Wireless handheld devices on NYSE’s floor 15 years before ipad invented (1995)
- Euronext becomes first pan-European equities market (2000)
- SFTI was the industry’s first network offering global connectivity (2008)
Business Needs
Release Count Per Year

- Projected 2011 release count based on releases thus far
Server Count Per Year

- Peak in server count due to data center migrations
Business Challenges in Software Deployment Methodologies
Business Challenges

- Rapid growth of deployment
- Increase in code line complexity
- Increasing pressure to reduce development time to prod
- Increasing size of server farm due to demand
- Zero tolerance for system downtime
- Software rollbacks must be rapid, scalable and reliable
- Audit best practice requires traceability of all activity
Business Challenges (continued)

- Scalability must be achieved with zero headcount growth
- Frequent mergers require process scalability
  - Increasing number of business entities to cover
  - High variance in supported operating systems and physical platforms
  - Increasing demand for new products
Two Core Concepts Behind the Business Challenges

- **Life cycle management** - controlling how its software is used throughout its usable lifecycle
  - Detecting unauthorized use/deployment
  - Detecting unauthorized change (config/script/binary)
  - Making sure software still exists on appropriate systems
  - Creating common patterns of usage for unique systems

- **Software Development Life Cycles** is the creating or altering of systems, and the models and methodologies that people use to develop/maintain these systems
  - Tracking creation <-> QA <-> Release <-> Staging <-> Distribution <-> Utilization <-> Deprecation
Environment Mapping Database

- Creating common patterns of usage for unique systems

<table>
<thead>
<tr>
<th>Zone</th>
<th>PRODMAHWAHAMEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Install Script</td>
<td>//utp_depot/RM/Install/US_Options/prod/utp_options_install</td>
</tr>
<tr>
<td>AppFileMgmt Root</td>
<td>//utp_depot/RM/AppFileMgmt/US_Options/prod/amex/</td>
</tr>
<tr>
<td>Block</td>
<td>C</td>
</tr>
<tr>
<td>App User</td>
<td>prod</td>
</tr>
<tr>
<td>Install Server</td>
<td>tu99a</td>
</tr>
<tr>
<td>Install Root</td>
<td>/app/install/prod</td>
</tr>
<tr>
<td>App Root</td>
<td>/app/utp</td>
</tr>
<tr>
<td>Config Path</td>
<td>//utp_depot/RM/ConfigFileMgmt/US_Options/prod/</td>
</tr>
<tr>
<td>Config Staging Path</td>
<td>//utp_depot/RM/ConfigFileMgmt/US_Options/prod_staging</td>
</tr>
<tr>
<td>Tidal Path</td>
<td>//depot/RM/TidalScripts/utp_options/cert/</td>
</tr>
<tr>
<td>Tidal Diff</td>
<td>☑</td>
</tr>
<tr>
<td>Tidal Diff Ignores</td>
<td></td>
</tr>
<tr>
<td>Conf Override</td>
<td>☑</td>
</tr>
<tr>
<td>Client Spec</td>
<td>${server}</td>
</tr>
<tr>
<td>P4 PORT Override</td>
<td></td>
</tr>
<tr>
<td>P4 PORT</td>
<td></td>
</tr>
<tr>
<td>Source Bash Profile</td>
<td></td>
</tr>
<tr>
<td>EOD</td>
<td></td>
</tr>
<tr>
<td>Readiness Check</td>
<td>☑</td>
</tr>
<tr>
<td>Need FQDN</td>
<td>☑</td>
</tr>
<tr>
<td>FQDN</td>
<td>.trad.com</td>
</tr>
</tbody>
</table>
Perforce Environment Setup
Perforce Depot Structure

- **Perforce**: Origin point for the deployment process.

  - **Dev Depot**: Development environment.
    - **Dev PC**: Development computer.
      - **Submit built binaries**: Workflow step for submitting built binaries.
    - **Verification Depot**: Verification environment.
      - **Verification Servers**: Verification servers.
      - **Push verified binaries to QA**: Workflow step for pushing verified binaries to QA.
    - **QA Depot**: Quality Assurance environment.
      - **QA Servers**: QA servers.
      - **Push tested binaries to PROD**: Workflow step for pushing tested binaries to PROD.

  - **PROD Depot**: Production environment.
    - **PROD Servers**: Production servers.
Binaries in Perforce from Dev to Production

- Build branches are fully preserved based only on files that are released
- Dev has no write access to verification application files
  - Forces all code to be submitted via Perforce
  - Audit compliance
- Dev verification zone files kept in Perforce
  - Ensures binaries that pass verification are the same that go to QA
  - Full integration log kept in Perforce for each file
QA Perforce Pros/Cons

• Pros:
  • QA can setup different environments on any release level with minimal overhead.
  • QA can signoff on out of order patch levels
  • Perforce integrations via atomic change lists
  • Component based testing for QA
  • Traceability back to source code based on application files

• Cons: (driven by organizational security practices)
  • No third party package installs
  • No changes that have to be done as the ‘root’ user
  • No OS changes done (example: hugepage size changes)
Automated Production Deployment
Production Deployment

• Advantages
  • Rapid application deployment to production
  • Distributed global proxies to reduce latency
  • Scalability to thousands of servers with minimal overhead
  • Ability to track application files back to source code
  • Installation time is identical whether installing one release or many
  • Efficient and accurate installations

• Disadvantages
  • Third party software is packaged separately
  • RPMs can be sold to a customer in a single package
Production Deployment Process

1. Binaries/application files are integrated from the QA depot to the Production depot

2. PMO puts in a request for a particular patch level
   - Based on the request, the particular server list and change list number will be pulled in from Perforce

3. Install and rollback scripts are auto-generated for that evening’s release(s)

4. Operations runs the script for installations once systems are down and clean
   - No need to worry about order of installations/rollbacks

Integrate from QA to Production

PMO submits request

Generate Install and Rollback scripts

Ops runs the scripts
Preview and Install Logs

• Ability to preview all files going to each server based on client meta-map
• Ability to script a ‘diff’ error to show if the actual install matches what should be getting installed
• All preview and install logs stored in Perforce repository
• Full trace of exactly what version of what file was deployed on what day to what server
Request for Release Deployments

Zone specified in client spec description

Release based on change list description of previous change list

Server names generated from host name in the client spec
Pending Installs

Filter by Application: View All
and/or Date Range: Today

Server list will populate servers to install to and P4HOST

Sync to RELID will match with the description of the change list in Perforce
Automated Rollback Scenarios
Perforce Rollback Pros/Cons

• Pros:
  • Quickly rollback to exact point of software on hundreds of servers in minutes
  • Easy intraday solution during an outage
  • Allows for weekend testing
  • Auto-generated in conjunction with the install script based on what is existing on the client before the install
  • No order dependencies when going to any release version
  • Full accuracy based on Perforce atomic change lists
  • Saves man hours of getting a server back to a particular patch level

• Cons (driven by organizational security practices):
  • Any OS/third party package dependencies must be done manually
  • No changes which require ‘root’ access
Application File Structure for Rollbacks

- Patch level submitted with release description
- Allows sync back to any atomic change list in the event of an outage

<table>
<thead>
<tr>
<th>Revision (Changelist)</th>
<th>Date Submitted</th>
<th>Submitted By</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1192500</td>
<td>4/14/2011 10:52:58 AM</td>
<td>mbulger</td>
<td>UTP_Equities_2.4.2.e@24082</td>
</tr>
<tr>
<td>1192493</td>
<td>4/14/2011 10:44:49 AM</td>
<td>mbulger</td>
<td>UTP_Equities_2.4.2.d@23968</td>
</tr>
<tr>
<td>1175923</td>
<td>4/8/2011 3:27:00 PM</td>
<td>mbulger</td>
<td>UTP_Equities_2.4.2.c@23962</td>
</tr>
<tr>
<td>1175917</td>
<td>4/8/2011 3:19:41 PM</td>
<td>mbulger</td>
<td>UTP_Equities_2.4.2.b@22627</td>
</tr>
<tr>
<td>1175905</td>
<td>4/8/2011 3:12:27 PM</td>
<td>mbulger</td>
<td>UTP_Equities_2.4.2.a@22170</td>
</tr>
<tr>
<td>1175885</td>
<td>4/8/2011 2:47:46 PM</td>
<td>mbulger</td>
<td>UTP_Equities_2.4.2@21616</td>
</tr>
<tr>
<td>1164164</td>
<td>4/4/2011 1:08:13 PM</td>
<td>mbulger</td>
<td>UTP_Equities_2.4.1.s@23771</td>
</tr>
<tr>
<td>1151186</td>
<td>3/30/2011 12:04:45 PM</td>
<td>mbulger</td>
<td>UTP_Equities_2.4.1.r@23707</td>
</tr>
<tr>
<td>1134925</td>
<td>3/24/2011 4:19:43 PM</td>
<td>mcruger</td>
<td>UTP_Equities_2.4.1.q@23538</td>
</tr>
<tr>
<td>1109744</td>
<td>3/15/2011 1:48:56 PM</td>
<td>mbulger</td>
<td>UTP_Equities_2.4.1.o@23405</td>
</tr>
<tr>
<td>1093631</td>
<td>3/9/2011 4:07:45 PM</td>
<td>mcruger</td>
<td>UTP_Equities_2.4.1.n@23309</td>
</tr>
<tr>
<td>1077817</td>
<td>3/3/2011 6:32:28 PM</td>
<td>mcruger</td>
<td>UTP_Equities_2.4.1.m@23038</td>
</tr>
<tr>
<td>1061888</td>
<td>2/25/2011 6:56:05 PM</td>
<td>mbulger</td>
<td>UTP_Equities_2.4.1.l@22402</td>
</tr>
</tbody>
</table>
Life Cycle Management – Meta-data Lockdown
Perforce Life Cycle Control

- RM maintains all production client specs
- Locked Clients
  - Clients for all production systems are locked and owned by production system accounts for that application
  - Locked clients can only be updated by Perforce admins, after being authenticated by Perforce
- Perforce Triggers
  - Triggers are configured to notify RM of attempts to modify production clients
- Perforce Depot Access Control
  - Perforce depot access can be allocated on a file by file level of granularity
  - Depot access is granted on an as needed basis and continually evaluated
Life Cycle Management - Post Production Installation Integrity Checks
Production Integrity

• Situation/Need
  • Production Black Box
  • Variances in production cause outages

• Solution
  • Automated method to compare state of production (application files/configuration files) to Perforce meta-map
  • Reporting framework to alert necessary teams to quickly view the results of comparison
  • Per system/per file basis run to evaluate integrity of systems
  • Files on systems are compared with the meta-map of production in Perforce and any noncompliance is reported via Diff Report.
Post Production Process Workflow

1. Environment Mapping Database
   - GenerateScripts.pl
   - Diff Scripts
     - Release Readiness Scripts
   - Perforce
     (Submit scripts)
   - Prod Depot
   - Perforce
     (Submit logs)
   - Prod Servers
   - Admin Server
   - Perforce
     (Sync up scripts on admin servers)
   - Prod Depot
### Post Installation Reports

<table>
<thead>
<tr>
<th>System Name</th>
<th>Diff Result</th>
<th>Diff Duration</th>
<th>Diff Beg</th>
<th>Diff End</th>
<th>RR Result</th>
<th>RR Duration</th>
<th>Begin RRC</th>
<th>End RRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>utp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Version Viewer

- Driven via Perforce change list description last synced on any given client spec
- History based on the past change lists synced down
Post Production Audit

• All scripts run in production are kept in Perforce
• Full historical tracking of server nightly audits against the Perforce meta-map
• All scripts are auto-generated based on Perforce client specs, hosts, users to allow for no manual changes to scripts to take place
Conclusion
Appendix (examples)
Common Perforce Commands in Automation

• Preview code example:
  • p4 -P $ticket -u $app_user -c $Spec -H $serverindex -d $ROOT sync -n ...\@$CL

• Preview log/install log – diff command
  • p4 diff2 -dw $INSTALL_LOG $PREVIEW_LOG
Client Spec Meta-maps Example

**Development:**

//utp_depot/RM/AppFileMgmt/US_Cash/dev/release_1.2/... //</servername>/


//utp_depot/RM/ConfigFileMgmt/US_Cash/dev/<servername>/db/... //</servername>/db/

**RM/Development Verification:**

//utp_depot/RM/AppFileMgmt/US_Cash/rm/release_1.2/... //</servername>/


//utp_depot/RM/ConfigFileMgmt/US_Cash/rm/<servername>/db/... //</servername>/db/

**QA:**

//utp_depot/RM/AppFileMgmt/US_Cash/qa/release_1.2/... //</servername>/


//utp_depot/RM/ConfigFileMgmt/US_Cash/qa/<servername>/db/... //</servername>/db/

**PROD:**

//utp_depot/RM/AppFileMgmt/US_Cash/prod/... //</servername>/


//utp_depot/RM/ConfigFileMgmt/US_Cash/prod/<servername>/db/... //</servername>/db/
Partial Auto-generated Install Script Example

LOG=${INSTALL_ROOT}/index_service/logs/index_install_prod_tt24108.log
BACKUP_LOG=${INSTALL_ROOT}/index_service/logs/index_service_prod_backup_${DATE}.log
MAIL_RECIPIENT="ReleaseManagement@nyse.com"
echo "" > $LOG
. ${INSTALL_ROOT}/InstallFunctions
VerifySync
DiffItself "${INSTALL_ROOT}/index_service/index_install"
SERVERS="mep01  mep02  mep03  mep04 
APP="index_service"
LABEL=...@1195053
APP_USER="prod"
CheckUser prod
CLIENT='${server}a'
for server in $SERVERS
do
echo "<SERVER_P4SYNC_OUTPUT>" >> $LOG
   ssh $APP_USER@$server ". ~/prod/.bash_profile; umask 0022; export P4CONFIG=; export P4CLIENT=${server} a; export P4USER=prod; export P4PORT=perforce:1666; export P4HOST=${server}; cd /app/index; /usr/local/bin/p4 -u prod sync $LABEL " >> $LOG 2>&1
   echo "" >> $LOG
echo "</SERVER_P4SYNC_OUTPUT>" >> $LOG
done
SubmitInstallLog $LOG
DiffPreviewAndInstallLogs