WhitePaper - Scalable lifecycle management via Perforce

Scalable lifecycle management via Perforce
Managing complex, dynamic production environments where speed & accuracy of deployment intersect with a requirement of deep information about the state of each environment (current and historical) and a guarantee of application integrity.

• Introduction
  o NYSE/EuroNext Business Challenges
    ▪ Recently Merged stock exchanges (July 2007), with Market Capitalization worth $19.93 Trillion(USD), and $4.69 Billion (USD) in gross revenues.
    ▪ Geographically distinct - NYSE is headquartered New York, servicing North America and EuroNext is in Amsterdam, headquartered in Paris, serving Europe.
    ▪ Growing Electronic Commerce/Exchange - 50-75 servers at merge point, rapid growth.
    ▪ Scalability of IT Infrastructure was stunted by manual configuration of servers, desktops, enterprise software
    ▪ No easy way to deploy updates, patches, new binaries
    ▪ No clear DEV -> QA -> RELEASE path, fractured between internal entities.
    ▪ No easy method to roll business back or recover after problems/failures - required by Security and Regulatory Bodies.
    ▪ Server config and program/app updates handled by VI config or tarball deployment (cumbersome, prone to error
    ▪ No easy way to monitor, audit, or track IT changes.
    ▪ Lot of guessing going on, lost productivity - not catastrophic, but unacceptable levels of downtime.
    ▪ Compliance / Audit requirements /Line of Sight limitations
    ▪ Massive variance in technology infrastructure & architecture of merged organizations
  o Overview of role of perforce in organization
    ▪ Global vcs system for both source and application release artifacts
    ▪ Over 3 years, Perforce has allowed IT to expand to thousands of servers (with a spike of double server population during data center migrations), with 99% up-time in North America with proposed expansion to the EU in 2011.

• What is Lifecycle management?
  o Software Lifecycle Management is the concept that an Independent software vendor can control how its software is licensed and used throughout its usable lifecycle. Because software is not always a static product, vendors often add additional features and/or updates for security purposes. To help software vendors
control the lifecycle, other software is often added on top of the original product and distributed with the software which unknown to the customers.

- The Systems Development Life Cycle (SDLC), or Software Development Life Cycle in systems engineering, information systems and software engineering, is the process of creating or altering systems, and the models and methodologies that people use to develop these systems. The concept generally refers to computer or information systems.

- Current industry example
  - Individual silo's producing RPM packages
    - No link to source/build/resulting RPM
  - Packages encoded with binaries/scripts/security permissions/OS settings/dependencies
    - Difference between internally developed software and 3rd party software
  - Packages installed as full RPM's VERIFY
    - Linux dependency
    - Limited ability to monitor change
    - No ability to observe variance based on additions to software directory
    - Can RPM's do txt based comparisons within package? VERIFY

- Perforce driven lifecycle management within a secure environment
  - Perforce specific features
    - Maintaining integrity of a release along pipeline by atomic change points
    - Easy & secure integrations between secure dev/qa/production sections of the repository
    - Detailed trace of application files route along pipeline for audit requirements
    - Massively scalable structure with fast operations
  - Silo's producing source code in common repository (if separate areas)
  - Release build produced by automated extract from repository
  - Build artifacts automatically submitted to QA area in repository (read-only)
  - QA repository is install point for all testing
  - Upon QA sign-off, build artifacts automatically integrated to production area in repository (read-only)
  - The production repository is a meta-map of all production machines
    - Script meta-map
    - Configuration meta-map
    - Binary meta-map
    - Automated updates/installations
  - Production machines are compared to baseline each night for possible change and additional of extra components
Production rollbacks occur within seconds across entire machine farms and are limited only by dependencies on other applications.

Perforce driven human scalability
- 3 yrs ago approx 100 releases/human in RM/year
- Currently 2000 releases/human in RM/year
- 2011 goal 3000 release/human in RM/year

Our implementation of Perforce as a complete Lifecycle management tool

Release Installation Process
- Client specs setup to sync to a particular application area in perforce
- Use of 'sync to changlist' to define a release package
- Changlist role in implementing a release pipeline
- Full metamap to application files on any server
- Allows for multiple full releases to reside in QA
- QA to setup an environment mirroring different release levels with minimal overhead
- Allows for component based testing within releases to pinpoint a software defect within each release branch
- Global solution for all QA to have access to applications being tested
- Minimizes QA environment setup time to focus on moving releases through the lifecycle more timely

Automated Production Deployment
- Distributed p4 proxies to reduce regional latency
- Ability to scale to thousands of servers with minimal overhead
- Rapid application deployment
- The role of Perforce in datacenter migrations
- Release version tracking via Perforce
- Automated rollback for production issues via Perforce
  - Rollback to any version released to production at any time in a matter of minutes for hundreds of servers
  - Rollback an entire application to a functional release at any moment
  - Perforce Advantages over RPM's / tarballs / other deployment
    - Single rollback solution
    - No order dependencies on rollback if changing the version multiple levels back
    - Increased speed to install and rollback any release
    - Accuracy based on metamaps and change lists in Perforce
    - Stability production environment at any chosen release level
  - Saving operations of man hours for deployment per server

Post Production
- Nightly Perforce Diffs for production application integrity
- Validation of application files stored in Perforce to mitigate risk in the production environment
- Creation of a nightly email report for results
- Helps troubleshoot any production issues
- Validation of operational changes throughout the production environment
  - Information visualization
    - Version Viewer
    - History
    - Available servers

- Closing
  - OMG we're so cool
  - and we love perforce to :)

• Closing