Configuration management

Software repositories

Build and testing procedures

Giuseppe Lo Presti, German Cancio, Sebastien Ponce
CERN / IT

Castor Readiness Review – June 2006

Outline

- Source repository
  - Structure
- Release workflow status
  - Building tools
  - Packaging process
  - Functional and validation tests
- Future release workflow
  - Use of ETICS

Sources' repository

- CVS
  - **Single** web-browsable repository
    http://isscvs.cern.ch/cgi-bin/cvsweb.cgi/CASTOR2/?cvsroot=castor
  - Includes all sources for the "central services" (shared with Castor 1) and all Castor 2 specific sources
- Current structure
  - **Roughly** a directory per system component
  - Specific directories for dedicated purposes
    - **`config`** for build macros and definitions
    - **`hsmtools`** for admin-side scripts and utilities
    - **`tools`** for developers' utilities
    - **`codeGeneration`** for the UML code generator
    - **`debian`** for all release-related scripts

Sources' repository

- Source code structure
  - Not very homogeneus code structure among different system components, mainly for historical reasons
  - Pure C: RMMaster, NameServer, Tape related daemons
    Hybrid C/C++: stager
    - Sources and headers in dedicated directories
    - Common headers in the `h` directory
  - Pure C++: RH, gcDaemon, core framework, db API, new components (e.g. VDQM 2, Repack 2)
    - Sources and headers in subpackages of the `castor` package
    - Following the Java convention
- Plan
  - Make clear separation among different components and adopt a single schema
    - Ease the build process

Release workflow

- Build process
  - Based on **`imake`**
  - Allows for platform independent makefile generation
    - Not so smoothly on Windows …
  - A set of flags to decide whether to compile each single component in Castor
  - Perfect choice in the early 1990s, now a bit obsolete
    - Lack of time prevented so far a complete re-evaluation of the build process, going for instance to other tools like **`autoconf/automake`** and/or **`gmake`**

Release workflow

- Packaging process for a new release of Castor

1. Update **`debian/changelog`** with logs from CVS
2. Edit the **`ReleaseNotes`** and provide:
   - Quick summary of new features and/or bug fixes
   - Procedure to update from previous release
3. Tag the whole repository with a **`vX_Y_Z_R`** tag
4. **`make tar`**.
   This launches a custom script, which creates a source tarball including ALL Castor code and a single **`.spec`** file, autogenerated on the fly
5. In the target certified machine, for each of the supported platform:
   `rpmbuild -ta castor-x.y.z-r.tar.gz`

Release workflow

- Pros
  - Single version number for the whole system. Useful in the early stages of Castor 2, as several changes were affecting all the components
    - No need to handle compatibility matrix

- Cons
  - Due to limitations of `rpmbuild`, if the packaging process goes wrong, one has to restart from the very beginning, and the **whole** Castor code takes **~ 30'** to compile, slowing down the entire process
  - A single bugfix in one component makes the whole system go for a version transition where it is not strictly necessary anymore
    - Typical case: a tape server change made Castor move to version 2.1.0-2, while the Request Handler remained the same as in 2.0.x

Functional and validation tests

- Current status
  - Not uniform test suites:
    - Regression tests for RFIO and the NameServer
    - Custom test suites for some of the other components
  - Lack of tests in some specific cases
    - For instance the PL/SQL code: but covered by stress tests
- Plan
  - Move to XUnit tests
    - In particular CppUnit for all new components

Functional and validation tests

- Dedicated castor instances for developers
  - all-in-one setup, no RPMs, simply **`make install`**
  - allows for easy test of new functionalities
- Production-like test instance
  - Only Quattor managed RPMs
  - allows for testing the deployment process
    - *see Jan's presentation*
  - cross-version compatibility tests are made here
- Occasionally: Internal Data Challenges
  - Depending on its availability
  - Production size instance to **stress test** and **tune up** the system
    - *see Bernd's presentation*
- After these steps…
  - The release is ready for CERN/Tier1s production instances
  - If needed, the Installation & Setup Wiki page is updated accordingly, in conjunction with the operation team

Future release workflow

- Plan: complete refactoring of the current release workflow
  - **Goal: being able to compile and package each component of Castor independently from the others**
  - This process will also ease the porting of the clients to other platforms (Microsoft Windows already asked long ago)

- Current status
  - A section-wide “Task Force” has been put in place in order to standardize the software process among the FIO software projects
    - Castor already involved

Future release workflow with ETICS

- **ETICS** (http://www.eu-etics.org)
  - E-Infrastructure for Testing, Integration and Configuration of Software
  - Secure access to a build and test infrastructure
  - Repository of components with defined external dependencies
  - Support of **RPM**, **Debian** packages, **MSI** for Windows

- ETICS will help standardizing the process, but Castor has to move to more standard make/RPM tools to meet ETICS requirements
  - Among our Task Force's mandates: being early users of ETICS and leverage their platform
  - Feedback from ETICS will help in our internal refactoring process, which will start by this summer

Comments, questions?