DIRAC software

DIRAC Project

Outline

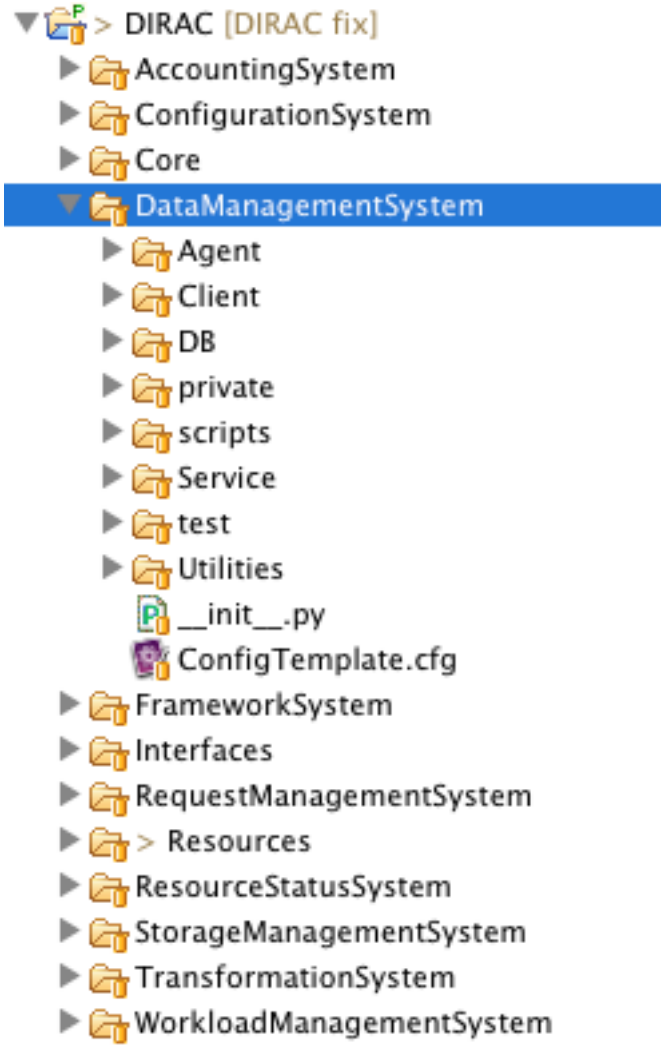
- Software structure
- Software management
- Releases procedure
- Installation
- Updates

Main software components

- Services
  - Passive stateful components responding to requests of (remote) clients
- Agents
  - Active components running in a endless loop and executing periodically their operation
    - Generally stateless (only caching some information)
    - Can not be contacted
- Databases
  - MySQL databases where Services are keeping their state
- Clients
  - The DIRAC functionality is available through clients – API interfaces to Services, Agents, Databases

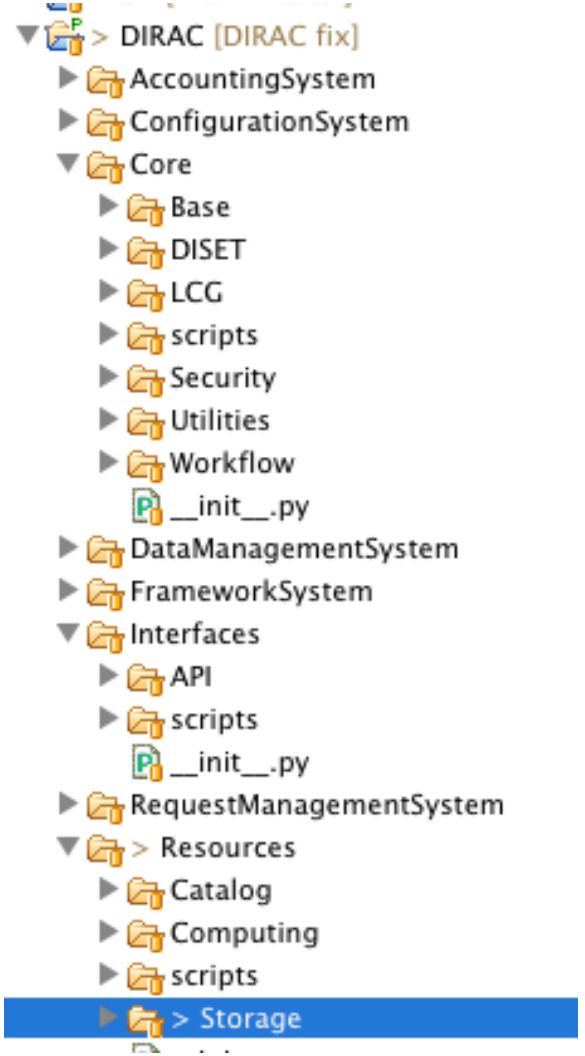
Software Systems

- The basic components are grouped in Systems
- Systems are top level directories
- In each system there are *Service*, *Agent*, *DB* and *Client* subdirectories
- *scripts* directory contains mini applications seen by the users as commands
  - E.g. **dirac-wms-job-submit**



Software Systems

- Other top level *directories*
  - *Core*
    - DISET secure framework
    - Common utilities and tools
  - *Interfaces*
    - DIRAC API – programming interface for external projects, e.g. GANGA
  - *Resources*
    - Clients for various external services: Computing Elements, Storage Elements, Catalogs



Software technologies

- Most of the DIRAC software is written in Python
  - Easy to prototype, read, debug
  - Do not hesitate to look into the code to understand better the functionality
- Much of the code is self-documented
  - Automatic code documentation is generated (epylog, sphinx)
- Some binary platform dependent software
  - PyGSI module to implement GSI standards, based on OpenSSL libraries
  - Written in C++ for efficiency reasons

Software repository

- Using Git software management
  - Very flexible, powerful
  - Excellent branch management
  - A bit difficult to start using it
  - A huge step forward compared to CVS/SVN repositories
- Using Github code repository service
  - https://github.com/DIRACGrid/DIRAC
  - Support for collaborative work
    - Personal code forks, assembling tools
    - Issue tracker, wiki, etc
  - Register in Github if you want to
    - contribute to the DIRAC development
    - report bugs
    - request new functionality features

External software

- The idea of the DIRAC software distribution is that it contains everything necessary to run DIRAC components
  - No assumption is done about any preinstalled software
    - except the native python to run initial *dirac-install* script
  - This makes the distribution heavier but much more reliable
    - No dependency on local environments where DIRAC clients happen to run
- The client external software bundle includes:
  - Python interpreter, openssl libs, …
  - This is installed in pilots as well
- The server includes in addition
  - MySQL, Web server (lighttpd), plotting (matplotlib), sqlite, …

LCG/gLite software

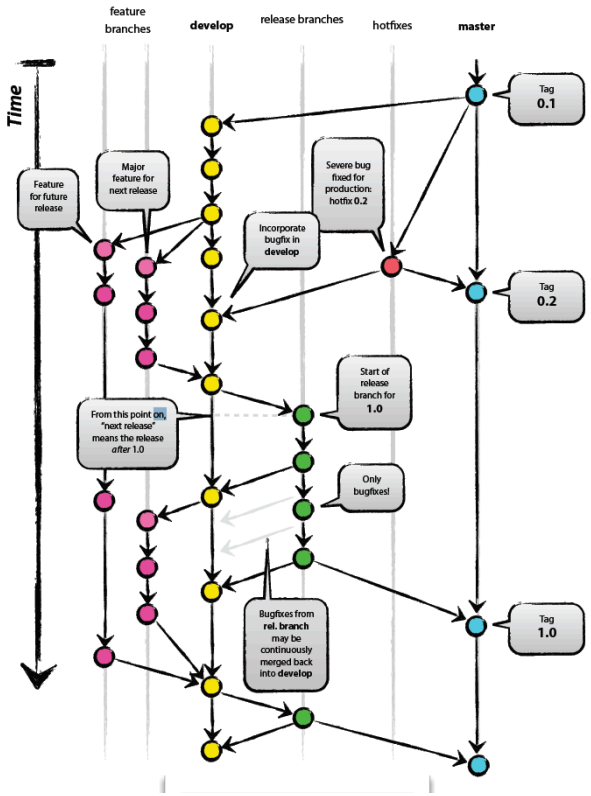
- We are providing a subset of LCG/gLite software necessary to do basic operations with the gLite resources and services
  - This is taken from the CERN Application Area middleware installation
  - Available only on the reference platforms of the gLite middleware: flavors of SL.
- Binaries, libraries, command line tools
  - VOMS tools (voms-proxy-XXX)
  - gLlite commands (glite-wms-job-XXX)
- Python bindings for the *gfal* library
  - Access to the *SRM* storage services, *gridftp* client

Software platform dependency

- The DIRAC software proper can be compiled for any flavor of Linux/Unix, including Mac OS
  - Precompiled binaries are available for the platforms that we encounter
    - Even Windows client was demonstrated (although not maintained)
- If clients must access third party services (LFC, SRM) then there 2 solutions to avoid installing gLite UI
  - Use LCG/gLite bundle provided by DIRAC
    - Limited platforms
  - Use DIRAC proxy services:
    - LFC proxy, StorageElement proxy
    - Clients interact with the services via DISET protocol

Software releases

- Branching and tagging using git tools
  - nvie.com/posts/a-successful-git-branching-model
- DIRAC software versioning:
  - v6r1p5 – major/minor/patch versions
  - Major releases – once in several years
  - Minor releases – once in 1-2 months
    - Need certification testing
  - Patch releases – at any pace
    - Bug fixes, minor functionality updates
    - No need for certification tests



Software release tools

- Tools to build and upload DIRAC releases
  - > dirac-distribution –l DIRAC – r v6r1p5
  - Building both python and binary tar files
- DIRAC Web Portal software is a separate project
  - DIRAC release description contain dependencies:
    - v6r1p4
      {
        Modules = DIRAC,Web:web2011121301
        Externals = v6r0
      }
- The release tools allow also to create and manage specific software projects based on DIRAC
  - Examples: LHCbDIRAC, ILCDIRAC, BelleDIRAC, EELADIRAC
  - In Lyon we will use only general purpose DIRAC software
    - Extensions can be considered

Release procedure

- Collecting developer contributions by applying Github "pull requests"
- Tagging a prerelease, e.g. v6r2-pre1, building distribution tar balls
- Installing the prerelease on the certification machine (at CERN), making a serious of tests
- Repeating the prerelease cycle until it stabilizes
  - Typically 2-4 weeks
- Tagging the final release, deploying it in the release area
  - CERN web server