Quality Control

Procedures and tools

Industry tests throughout software development cycle

http://www.aonix.com/Products/Testing/10xpart3.html

- **Requirements testing**
  - Software should be tested against an understanding of what it is supposed to do
  - Tools:
    - ➢ Requirements verification: check Syntax, Semantics, Testability
    - ➢ Requirements modelling: generate use-cases to cover the requirements
    - ➢ Requirements validation: generate test-cases from the use-cases
- **Design testing**
  - Same tools as requirements testing, but at the component level rather than system level
- **Code testing**
  - Easiest phase if above done properly
  - Tools:
    - ➢ Metrics reporter to measure complexity in data flow, data structure, control flow. Helps to identify which parts of code need most testing.
    - ➢ Code checker to look for misplaced pointers, uninitialised variables, deviations from standard etc.. To be used BEFORE code inspections (if any)
    - ➢ Code instrumentor plus structure coverage analyser to measure structural coverage of test-cases

Babar

http://hepunx.rl.ac.uk/BFROOT/www/Computing/Programming/QC/QCHome.html
http://hepunx.rl.ac.uk/BFROOT/www/Computing/Programming/QA/QAHome.html

- **Quality Control**
  - Code+design rules and guidelines (CodeCheck)
  - Release procedures
  - Memory leaks (manually, Insure++)
  - Profiling
  - Not clear how much is enforced (info is rather old)
- **Quality Assurance**
  - Software libraries to create and fill histograms
    - Release QA: Broad check on physics plots
    - Production QA: specialised checks by sub-detector, for simulation, digitisation+pileup, reconstruction
  - Documentation and tools to produce and compare histograms against reference set
    - c.f. Aleph online, Aleph RQ
    - c.f. SICB quality checking....
  - In production, results on the web
- **Problem reporting and tracking**
  - Remedy, ARweb

ALICE

F.Carminati, http://alisoft/offline/development.html

- **Enforced:**
  - Alice Coding Conventions
    - Checked with RuleChecker (see CHEP2000 presentation)
  - Packaging rules
    - Makefile structure, subdirectory structure, rootification, dependencies
  - Each package must have a test macro
    - To exercise large part of capabilities
- **Planned:**
  - Code reviews

ATLAS

M.Stavrianakou, D.Burkhart
http://atddoc.cern.ch/Atlas/DaqSoft/sde/Welcome.html
http://atlasinfo.cern.ch/Atlas/GROUPS/SOFTWARE/HELP/librarian/index.html

- **Online (Back-end DAQ)**
  - Documents to be delivered at each step of software process
    - Big emphasis on inspections of documents
    - Successful, but very manpower intensive (can it scale?)
- **ATLAS Software Process (ASP)**
  - Similar approach, failed in offline world (too heavy/strict)
- **New approach under discussion**

New Atlas approach

http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/qc/QC_Process_v1.ps

- **Onion model for strictness of rules**
  - Responsibility for QC with software developers
- **Quality Criteria:**
  - Quality of design
    - clear, modular, compliant with architecture. Quality of interfaces
  - Documentation
    - problem statement and algorithm description, design document, users' guide, example (including testing procedure and reference results)
  - Coding Conventions (CodeWizard)
  - Robustness (Insure++, metrics)
  - Maintainability (readability, portability, internal diagnostics)
  - Performance (physics quantities, speed vs. precision)
- **Implementation:**
  - Support developer with checking tools, code fragments, document templates
  - Validation via inspections, walkthroughs, reviews, tests
    - Including testing plan
  - Only packages that have passed QC can be released
    - Strictness of validation criteria to evolve

Software Process Improvement

- Bottom-up approach, avoids imposing procedures
  - Make it easy to check rules, agree within each project on what to check
  - Establish Process
    - Document existing processes
  - Process Improvement
    - Identify possible improvements, analyse costs, prioritise
    - Procedures constantly optimised
  - Process Assessment
    - Measure effectiveness of process in achieving goals
- Implementation:
  - 23 processes documented (many are trivial!)
  - Tools identified, "partly deployed"
    - Insure++, CodeWizard, McCabe (metrics), Remedy
- QA responsibility of developers
  - Verification by librarian and SPI manager

??LHCb??

- **Document and evolve existing processes**
  - Coding and documentation guidelines
  - Release procedures
  - Testing
  - ...
- **Evaluate and commission popular tools**
  - CodeWizard, Insure++, Remedy, ...
  - Put in production for core software
- **Develop QA test environment**
  - Inspiration from Babar, Aleph online+RQ, ...
- **Study Atlas and CMS processes**
  - Biggest hurdle is acceptance by developers. Can we learn from what Atlas and CMS (and Babar) actually implement?