Next steps in software quality

Stefan Kluth

Software TIM Berkeley, 11.11.2015

Overview

- Several areas
  - Tools (Scott and Stewart)
  - Education and training
  - Testing and validation
  - Refactoring

Software education review

- Report is on CDS
  - ATL-COM-SOFT-2015-150
  - https://cds.cern.ch/record/2060115
- Some recommendations:
  - Subsystem tutorial and workbooks
  - On-the-job training with experts
  - External courses
  - Software development teams

Subsystem tutorials/workbooks

- Essential to get new developers in
- Should happen now
- Explain algorithms and implementation
- Also contain subsystem specific advice for porting to new FW

On the job training

- How to put into practice?
- Project of new developer with strong and consistent expert support
  - Qualification tasks?
- Accounting of expert effort?
- Also needs a work plan with schedule and project decomposition

External courses

- Some ideas
  - Paralellism/threading/MPI/OpenMP
  - (unit) testing, continuous integration
  - Data oriented programming
  - … your ideas here ...
- Overlaps with self-training
  - A compact course could be more efficient
  - Exposure to external ideas

Software teams

- (Where do) we have them
  - CERN, LBL, BNL, others?
  - Make an inventory, correlate with tasks
  - Voluntary or institutional effort?
  - “organigram” helps new developers to navigate
- Send new developers to teams
  - Qualification projects
  - Push projects

Testing

- Levels of testing
  - Unit testing
  - Integration testing
  - Validation
- We have
  - Little unit testing, and fragmented
  - Some integration tests
  - Validation (by package, RTT)

Unit tests

- Consensus that unit tests are essential
  - Locate bugs quickly
  - Live documentation of code
  - Design aid
  - Enable refactoring for large scale migrations
- Push to have unit tests in all new code
  - And all renovated code
- ATLAS unit tests fragmented
  - Hand-crafted: higher level features missing

Unit tests

- Advocate google tests (gtest/gmock)
  - Last future framework hackathon
  - Stable, simple
  - Supports mock object concept
  - Coverage analysis together with gcc/gcov
  - Large user community ouside HEP
  - Integration into IDEs (clion)
  -

gmock

- Component architecture vs unit tests
  - Component classes difficult to test, need many other components
  - gmock mock objects: derive from real class, implement methods using gmock macros, may also mock parameter and return objects
  - Methods via macros instrumented: observe calls
  - Removes need to setup complete framework
- Needs work to establish in ATLAS

Unit test coverage

- Gtest together with gcc -ftest-coverage and gcov
  - Code is instrumented, gcov produces statistics
  - Not ideal if optimisation is on
  - Analyse gcov output: fraction of code executed by test suites
- Could be automated into (dbg) nightly
  - Compile with coverage, run gcov, accumulate and report results

Unit test next steps

- Integrate gtest/gmock in dev releases
- Migrate AtlasTest package
- Migrate existing tests?
- Inventory of existing unit tests
- Training: tutorial
  - Once we know what to train

Refactoring

- Many packages are migrated to new FW
  - Change code w/o changing behaviour → Refactoring!
  - Opportunity for improvements
  - Apply proper refactoring techniques
    - Get the job done faster and with less bugs
- Education and training
  - External courses for targeted developers(!)
  - Tutorials and workbook sections
  - Refactoring experts to team with subsystem developers?

Refactoring tools

- Refactoring by hand is hopeless at scale ☠
- IDE with refactoring support
  - Decent unit test coverage assumed :)
  - State-of-the-art in java world: compiler integrated into ide, enables complex refactoring tasks
  - C++ much harder due to language complexity
  - Eclipse CDT, clion support this
- Offline refactoring procedures
  - based on libclang, google does this, Rolf investigated for ATLAS

Hackathon topics?

- Dev release integration of gtest/gmock
- AtlasTest gtest/gmock migration
- Unit tests inventory
- Framework component test with gmock
  - An algorithm, service or tool
- Try a cmake ATLAS project in Eclipse CDT or clion