Tagging ACTS-0.1.0

ACTS developers meeting

2016-05-10

What is missing?

- formal stuff:
  - a license !!!
  - change log – not that important for first tag…
  - contribution guide
  - better cmake configuration with more options
- code-wise:
  - documentation !!!
  - unit tests
  - standalone geometry example
  - fitter (probably only in 0.2.X)
- target: we have postponed this many times, but we should aim to have ACTS-0.1.0 by the end of this week

License

- HSF recommendations and information on licencing
- clearly: should be usable by others
- should allow for extensions/modifications (under same/modified licence?)
- usable in commercial/proprietary software? *probably not an issue*
- restrictions on the licence of the incorporating project?
- who are the copyright holders? technically this is everyone who contributed some piece of code, we should keep a list...

Other formal stuff

- change log: can be easily generated in the future from JIRA
  => requirements:
  - no direct commits on master (not even a single one)
  - all merge requests must close one (or more) JIRA tickets
  - JIRA tickets should contain a decent descriptions
  - strive for more careful review of merge requests
- cmake configurations and options:
  - add more build options to cmake, make it more robust
  - implement 'configure –help' functionality similar to how it is done in ROOT cmake
- add version.h

Contribution guide

- Who can contribute?
  - Anyone with access to CERN gitlab and CERN e-groups
    => (external) CERN users
- Contribution possible through:
  - forked repositories (CI setup not yet tested for this, will do)
  - directly in ACTS repository (based on e-group membership?)
    => which option do we want to support?
- workflow proposal:
  - semi-ff approach: rebase before merge, then no-ff merge
    => linear project history, easy to remove feature branches from master again
- some conventions on code style, naming etc

Documentation and testing

- doxygen, doxygen, doxygen
- every method, member variable and function must be documented
- all documentation should go into header files
- in many cases the documentation is still improvable:
  - do not repeat the function name
  - state any pre-/post-conditions, pit fails, things to look out for
  - describe all input parameters and the return value
- NO @author statements

- unit tests are completely missing:
  - coordinate transformations
  - geometry consistency (correctly linked, containment...)
  - propagation, navigation, extrapolation

Next steps

- once we are happy with the state of the repository, create a new branch 'release-0.1.X'
- in this branch set version number etc
- make a first tag 0.1.0
- all bugs of this version are fixed in this branch, new tags 0.1.1 etc are created, the bug fixes are cherry-picked into master if needed