Geant4 Repository Openness Discussion

Geant4 Collaboration Meeting, Lund

30 August 2018

Why Now?

- The migration of the Geant4 repository to GIT and the adoption of GIT tools and standard processes offers an unique opportunity to re-think the way contributions to the software project are encouraged and then integrated

Encouraging Contributions

- Recognition
  - Author names are part of the GIT history associated to the changed or added lines
- Lower technical barriers
  - Use the ‘current best practices’ to participate to the open source project
- Make the life of maintainers easier
  - Use the same tools/process for external and internal contributions
- Many different kind of contributions
  - Easy bug fixes provided by users
  - Prompt validation of bug-fixes
  - Medium to large functionality changes and additions

World Read Access to ‘master’

- I am assuming that ‘master’ branch in always working and reflects the latest working state of the G4 repository
  - Merge requests (MR) at tested against the ‘master’ branch to validate them in the current [going to be implemented] process
  - Once a set of MRs validated on the nightlies, all of changes are going to be merge into the ‘master’
- Developers work against the ‘master’
  - Bug fixes are provided to version branches and then ‘cherry-picked’ to the master
  - New developments are done against the ‘master’ and the developer need to ‘re-base’ time to time to avoid major divergences
- If we want that contributors to work the same way we must be able to make ‘master’ world read

Contributors Use the Same Tools

- Contributors should provide GIT merge requests (MR) that are then reviewed by the G4 collaboration
  - Same procedure as for the G4 developers
- We should require that the contributed MR works well and is mergeable with the 'master' branch
  - The work of re-basing and solving possible conflicts carried by contributor
- The 'maintainers' may apply perhaps a stronger criteria before accepting an external contribution
  - e.g. evolution proposal approval for sizable changes, coding conventions, all tests passed for all platforms, etc.

Addressing the Possible Concerns

- Risk for contradictory / diverging merge requests
  - Large contributions need to be accompanied by a ‘evolution proposal’ in line with the main project objectives
- Higher costs for evaluating requests and resolve potential conflicts
  - Code conflicts are resolved by contributor
  - Benefits should overcompensate the cost of evaluating requests
  - We can always refuse a contribution
- Ongoing developments should not to be used for productions
  - Untagged ‘master’ should have a disclaimer and warning that has not been validated by the Collaboration
- Developments are subject to publications by developers themselves
  - Development branches can be made private (by forking or access control)