# Code review ideas

Rob Kutschke
Erica Snider
*Fermilab*

LArSoft Steering Group Meeting
April 8, 2016

## Code reviews

- We are developing a process for implementing a regime of regular / periodic code reviews.
- Why?
  - Can think of this partly in terms of managing risk
    - “Bad” code can impose high costs
      - Excessive resource utilization from poor programming practices.
        - *Often discovered during operations, immediately promoting it to a “crisis” situation*
      - Lack of compliance with architecture / design guidelines can:
        - *Make it costly to maintain code*
        - *Make it difficult / impractical to adapt code to meet physics goals*
        - *Reduce its utility within the software suite by needlessly limiting its potential scope*
    - These costs can appear suddenly when a new requirement / opportunity arises
  - Code reviews can help mitigate this risk (but cannot eliminate it!)
  - The Collaboration needs to decide how much risk it is willing to take
    - Balance the cost of reviews vs. risk of missing some (maybe future) physics goal

## What would a review look for?

- Can imagine that several types of things can be examined, each with a different level of scrutiny / effort / expertise needed
  - Easiest
    - Compliance with documentation, code formatting, stylistic conventions
  - Harder, closer look at the content of the code
    - Compliance with large-scale architecture and design guidelines and principles
      - Are services and modules structured properly?
      - Are tools and utilities properly encapsulated in classes / functions?
      - Is code structured to allow detector interoperatiliby?
      - Etc.
  - Hardest, most time consuming, detailed examination of the code
    - Compliance with C++ best practices
    - Algorithmic / numerical issues
    - Efficient use of data structures

    This type of review can identify many small inefficiencies that are otherwise difficult to isolate using conventional profiling tools

## What about the review procedure?

- Can also imagine several processes, ways to organize reviews
  - Examination at intake
    - Would necessarily need to be a cursory review.
    - Might be able to assess compliance with major design guidelines
    - Could be done by one individual
    - Difficult to do in the current environment in which there are many commits coming from many people
  - Examination of existing code
    - Anywhere from one person to a small group, depending on the code element in question and the depth of the review
    - Need a process to determine target code, define goals
  - Examination of an entire sub-system or systems
    - Multiple people needed over a longer time period
    - Need a process to determine target systems, define goals

# Toward a proposal

- Lots of possibilities, so:
  - The process should provide a lot of discretion as to code targets and depth
  - Also should specify a strong, on-going process by which to gather input and feedback from experiments and partner projects
- Some ideas
  - Identifying targets and review depth / goals
    - Core team can propose targets, goals, format in consultation with experiments
      - Coordination Meeting
      - Coordinator's Meeting
      - Steering Group

      depending upon scope of review
    - Experiment representatives can request reviews (via offline representatives)
      - Must also offer effort to assist with review process in general
    - Computing infrastructure or partner projects
      - Initiate review targeting usage of relevant resources, services, products, etc
    - Code owners can request a review

## Toward a proposal

- Some ideas (cont'd)
  - Assembling review “committees”
    - Provide the core team discretion in assembling teams appropriate to the review
      - Core team members
      - SCD domain experts
      - Experiment domain experts
        - *Critical to get involvement from those close to code, or those impacted by code*
      - External reviewers in the case of broad scope, if impact warrants
    - This must be done in close consultation with experiments
  - Results of review
    - Written review report
      - Length, detail commensurate with type and depth of review
    - Vital that effort be committed prior to the actual review – from the experiments, projects, SCD – to implement the recommendations
      - Reviewers should be available for consultation during this phase
      - The process should include tracking the progress of the implementation

## Toward a proposal

- Some ideas (cont'd)
  - Other considerations
    - Should extract “lessons learned” and make those available in concise form
      - Use as material in any future courses, tutorials, guidance documentation
      - More generally, provide developers with tools and knowledge to write better code
        - *The workshop this summer, for instance*
    - Continuity
      - Need to ensure that some people brought into the effort are “groomed” to become review leaders
- Next steps
  - Perform a “dry run” to develop a process on PMA algorithm this spring
    - Present and discuss the process at the workshop this summer
  - Aim at having a real process in place later in the summer
    - Include in the implementation plan