Experience with GIT in CMS

V. Ivanchenko

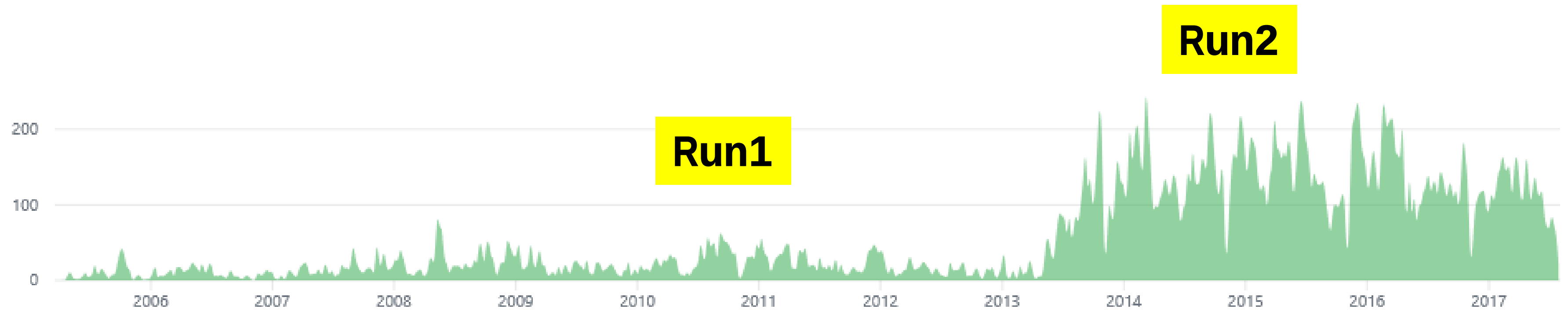
CERN & Tomsk State University, Tomsk, Russia

27 September 2017

Disclamer

- This is not an official CMS talk but only my private impression about git, CMSSW organisation and CMS software process
  - No warranty and no responsibility and no liability for this material

Number of commits to CMSSW as a function of time



- At the end of 2013 git was adopted for CMSSW development
  - There is a clear visible step from cvs to git in 2014
- Why number of commits such increased? My guesses
  - Much more easy control pull requests than tags
  - Instruments of git gives developer more confidence what he/she is doing
  - More automatisation
  - CMS reduced amount of private code
- Migration of CMSSW simulation to 10.0 with MT mode was smooth thanks to git
  - Only 2 developers were involved but substantial part of CMS SIM code was revised

CMSSW approach for git

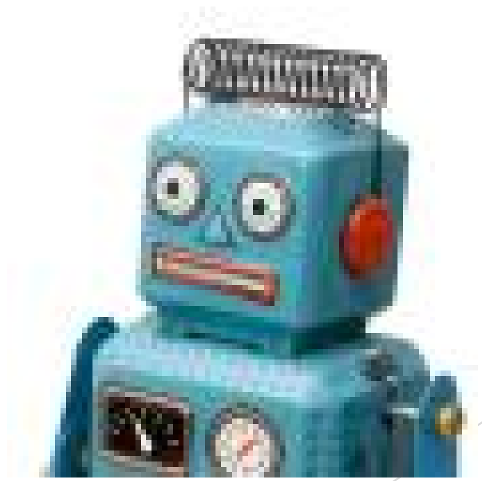
- There was a serious discussion on the migration
  - However, there was no real alternative well presented
  - GitHub was chosen at that moment
    - no equivalent CERN cervices was available at the end of 2013
- After migration there was no real oppositions to new software process
  - Couple of tutorials and compact documentation were provided
  - Usually 1 person is responsible for git support
    - with help of other members of CMS core software team
  - There was no protesters (at least, known to me)
- A subset of basic git instruments is used everyday
  - For me as a user it is not more complicate than g4svn
    - Bot commands
    - Scram commands

CMSSW and git

- CMS support today 29 git projects
  - 70 core developers
- CMSSW is the main project, 1st created, it has today
  - 95 branches
  - 663 contributors/users (not all make PRs)
  - 20615 pull requests (PR) since 2013
  - 167 PRs still open
- Geant4 project was created ~1 year ago, it has for today
  - 10 branches
    - corresponding to Geant4 production releases/patches
  - 4 contributors
  - 20 PRs, all closed

CMS software developers

- Any CMS developer may submit PR, close his/her PR, and comment on any PR
- Experience CMS developer has the additional privilege to start testing of a PR
- L2 is a category responsible who can approve or reject a PR for their category
  - Core, gen, sim, reco… - equivalent to Geant4 category coordinators
- L1 is the main managers who may merge or close any PR
  - Equivalent Geant4 release manager
- Bot is doing main manipulations with PRs:
  - Notification L1 and L2 via e-mail
    - I get ~100 e-mails from Bot daily
  - Build branch + PR
  - Submit tests to Jenkins
  - Making statistics and graphics



CMS software process

- Developer submit PR to a branch
  - Add description (including links to talks, plots, other PRs)
  - Submission to the master branch without restrictions
  - Submission for previous branches require a discussions at release meeting
  - L2 and L1 may reject any PR at any moment
- L1, L2 or experience developer may trigger testing
  - Bot starts tests from code quality check (clang-tidy set of checks)
  - If passed, Bot build branch + PR
  - If passed, Bot submit Jenkins tests
  - If passed, Bot start comparisons versus baseline
  - Each L2 should approved/reject PR or explain why it is not signed
    - If there are concerns the developer may update PR and testing will be re-triggered
- L1 merge successful PR to the branch
- Reference versions of a master branch are created approximately once in 2 weeks
- There are few production releases per year
  - Focused on the experiment goals
- There are many patch or minor releases by request of experiment
- https://github.com/cms-sw/cmssw/pulls