Fermilab U.S. DEPARTMENT OF ENERGY Office of Science

# PIP-II Technical Workshop

# Software Development Strategy Discussion

Tanya Levshina
SCD/CPI

1-4 December 2020

A Partnership of:
- US/DOE
- India/DAE
- Italy/INFN
- UK/UKRI-STFC
- France/CEA, CNRS/IN2P3
- Poland/WUST

# Outline

- Key to successful collaboration in global inter-organizational software development projects
- Selection of a shared repository platform
- Project organization

# Key Ingredients for a Successful Project

- Clear understanding of
  - requirements and scope
  - roles and roadmap
- Communication and ongoing reviews
- Documentation
- Well structured code
- Shared repository

# Shared code repository

- Effectively share code in a managed way to speed up development
- Avoid code duplication
- Encourage best practices in code development
- Improve code quality and provide means for collaboration

# Open-Source Version Control Software (VCS)

| Software | History | Current Status | Repository model | Atomic Commit | Signed revision | Merge Tracking | Interactive Commit | Users |
|---|---|---|---|---|---|---|---|---|
| CVS | First publicly released July 3, 1986; based on RCS | Maintained but new features not added. Last release from 2008. | Client-server | No | No | No | No | NetBSD, OpenBSD |
| Subversion (SVN) | Started in 2000 by CVS developers with goal of replacing CVS | Active | Client-server | Yes | No | Yes | Partial | ASF, gcc, SourceForge, FreeBSD, GoogleCode, PuTTY, Apache OpenOffice, TWiki |
| Mercurial | Started by Matt Mackall in 2005 | Active | Distributed | Yes | Yes | Yes | Yes | Was used by Facebook and Mozilla at some point |
| GNU Bazaar | Canonical Ltd | Active | Distributed | Yes | Yes | Yes | Yes | MariaDB, Squid, PyChart, Linux Foundation |
| GIT | Started by Linus Torvalds in April 2005. | Active | Distributed | Yes | Yes | Yes | Yes | Linux kernel, Android, Bugzilla, LG, jQuery, DragonFly BSD, GNOME, GNU Emacs, KDE, MySQL, Ericsson, Microsoft, Huawei, Apple, Amazon |

# VCS Properties

- Preserved and traceable history
- History includes not only files but directories (folder) structure
- No single central repository
- Ability to maintain a repository while offline
- Access control: view/contribute/maintain
- Branching/merge support. Merges should be easy especially when there are no conflicts.
- Tag management
- Built-in diff capabilities
- Rollback capabilities
- Efficient binary file support

GIT is a clear winner for the time being

# Requirements for VC Platform (I)

- GUI to perform all major operations, preferably from browser
- Access Control (Security)
- Code review tools
- Configurable approval interface
- Integration with bug tracking systems
- Integration with software build tools
- Release management
- Integration with Continuous Integration (CI) tools
- Documentation

# Requirements for VC Platform (II)

- Search
- Notifications
- Statistics/metrics and visualization
- Various deployment schemas
- Integration with 3rd party tools
- Integration with IDE

# Comparison of VC Platforms

| Name | Users (M) | Projects (M) | Code review | Bug Tracking | Web hosting | Wiki | Access Control | Build System | Self hosting |
|---|---|---|---|---|---|---|---|---|---|
| GitHub | 40 | 100 | Yes | Yes | Git, SVN | Yes | Private/ Public | 3rd-party Travis CI, Appveyor (addition of "Actions" to GHES) | GitHub Enterprise (Comercial) |
| BitBucket | 5 | N/A | Yes | Yes (Jira integration) | Git | Yes | Private/ Public | Yes | BitBucket Server (Commercial) |
| Launchpad | 4 | 0.04 | Yes | Yes | Git, Bazaar | No | Public | Ubuntu | Yes (Ubuntu only?) |
| SourceForge | 3.7 | 0.5 | Yes | Yes | Git, SVN, Mercurial | Yes | Public | No | Free |
| GitLab | 0.1 | 0.5 | Yes | Yes | Git | Yes | Private/ Public | Yes | Community Edition(Free)/Enterprise Edition (Commercial) |

# VC Platform Pricing

## GitHub:

- Single developer - FREE unlimited private/public repositories (new feature of 2019) and 500MB package size
- Team - $4/user/month (with the user and team permissions management) and 2GB package size
- Business - $21/user/month and 500GB package size

## GitLab:

- The community edition is free for an unlimited number of users
- Enterprise Editions:
  - Starter is $4/user/month
  - Premium for $19/user/month
  - Gold $99/user/month

## Bitbucket:

- Small teams (up to 5 users) can use it for free
- Teams pay $1/user/month or  the unlimited users' plan, which costs $200/month.

# Security and Access Control (GitHub)

- Dependabot alerts & security update
- Public secret scanning
- Code scanning  on all public repositories
- Role-based access control
- Could enable 2FA
- GitHub Enterprise is integrated with an organization's authentication  (LDAP, SAML SSO).
- GitHub Enterprise provides audit logs and monitoring

# Security and Access Control (GitLab)

- Code is tested upon commit for security threats
- The security team can see and manage unresolved vulnerabilities captured as a by-product of software development.
- SAML SSO and enforced 2FA
- Premium and Gold provide:
  - CI/CD for GitHub
  - Audit events and log
- Gold provides:
  - Credentials inventory and management
  - Secret detection: prevent secrets from accidently leaking into your Git history.
  - License compliance: automatically search project dependencies for approved and unapproved licenses

# PIP-II IT Software Status

- LLRF and EPICs IOC firmware/software repository is GIT
- Controls related software is in CVS and GIT
- VC Platform is Redmine (Open-Source) hosted at Fermilab
- Multiple projects are used for LLRF firmware/software repositories in Redmine
- Major issues:
  - code redundancy
  - storage of binaries (e.g qar files) instead of source code
  - lack of tagging/release policy

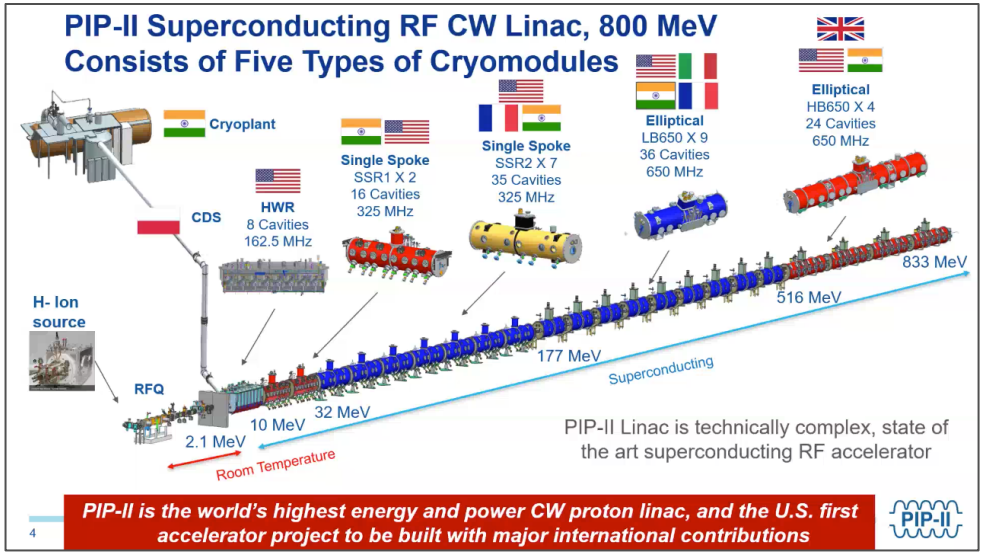
# Fermilab VC Platform Roadmap

- Strong push to use GitHub
  - Fermilab developers could contribute code to GitHub Open-Source products.
  - Fermilab developers who are creating a new Open-Source product at GitHub should get an approval from Partnership Agreements and Intellectual Property Management team. Software that is visible to the public and could be reused should have a reference to an appropriate software license. The guidelines for license selection is provided by Fermilab Legal Office. Computing Sector is currently using BSD 3-clause License or Apache 2.0
- The Core Computing Division supports GitHub Enterprise on-Prem. They are running PIP-II IT instance with 20 licenses. In order to get to an access a user should be registered with Fermilab.
- The Scientific Computing Division has some experience with GitLab on-Prem.

# PIP-II Software Organization

- The development of the software for the PIP-II will take several years.
- The maintenance of the PIP-II software will take many years.
- There are multiple developers and multiple organizations with different approaches for software development and different organizational rules.
- That is why it is crucial to figure out how we want to structure a project, so it is possible to:
  - Understand the layout and dependencies
  - Understand the purpose of a particular subproject/module
  - Avoid duplication of development efforts and promote reuse
  - Decide on branching/tagging and release procedure

# What Do We Want to Build?




The slide from Lia Merminga's presentation

- We would need to identify:
  - the structural elements and their interfaces
  - the behavior of the elements specified by the collaborations among those elements
  - the organization of the elements into progressively larger subsystems
- We would need to decide on architectural style and agree on standards in order to build these elements and their interfaces.

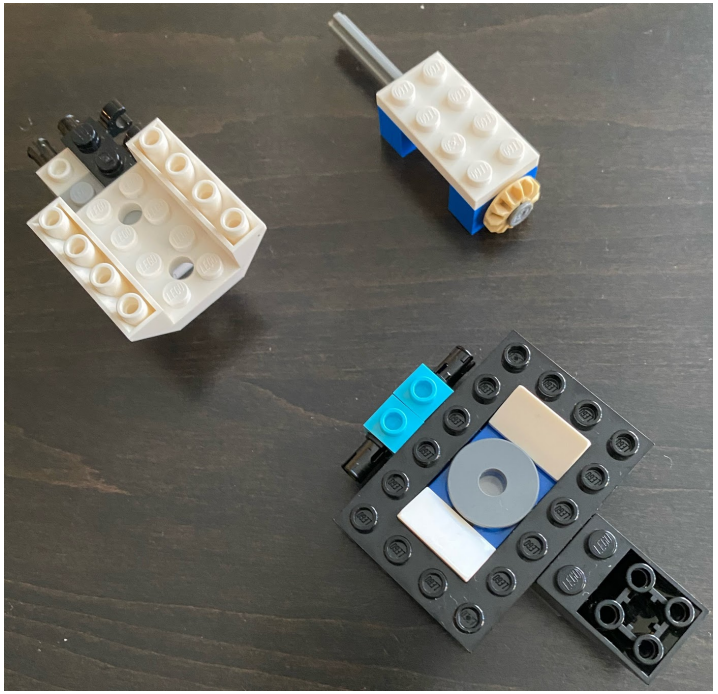
# What Do We Want to Build?

# Project Structure Organization Examples (I)

Chaos

Organization by teams/ institutions

# Project Structure Organization Examples (II)



Rumbaugh Model - objects in the code mirror objects in real world.

**Don't repeat yourself** (**DRY**) principle: "Every piece of knowledge must have a single, unambiguous representation within a system"

# Topics for discussion

- How to move forward on selection of VC Platform?
- What should be contributed to existing Open-Source projects; started as a new Open-Source project; or remain private?
- How to organize code that is being written right now so the structure would be suitable for the future development. What are the code and documentation?
- What is a review policy for contributed code?
- Should CI be part of release process?
- What are the requirements for logging and monitoring?
- What are cyber security related aspects of the project?