OPNFV Release Process 2.0

THE LINUX FOUNDATION

LF NETWORKING

Considerations

- Reconcile CNTT requirements with OPNFV
- Decrease emphasis on installers
- Simplify and reduce the number of milestones
- Support OPNFV level requirements planning
- Improve release planning at the project level
- Improve accountability across all project types
- Enable independent releases
- Increase community engagement in the release process

OPNFV Level Requirements

- Why is this important?
  - We need a way to agree upon and to prioritize broad requirements that help to advance our mission, or to take respond to a concern that affects most projects.
    - Example: Python 3 migration
  - We need a way to address CNTT requirements affecting multiple projects.

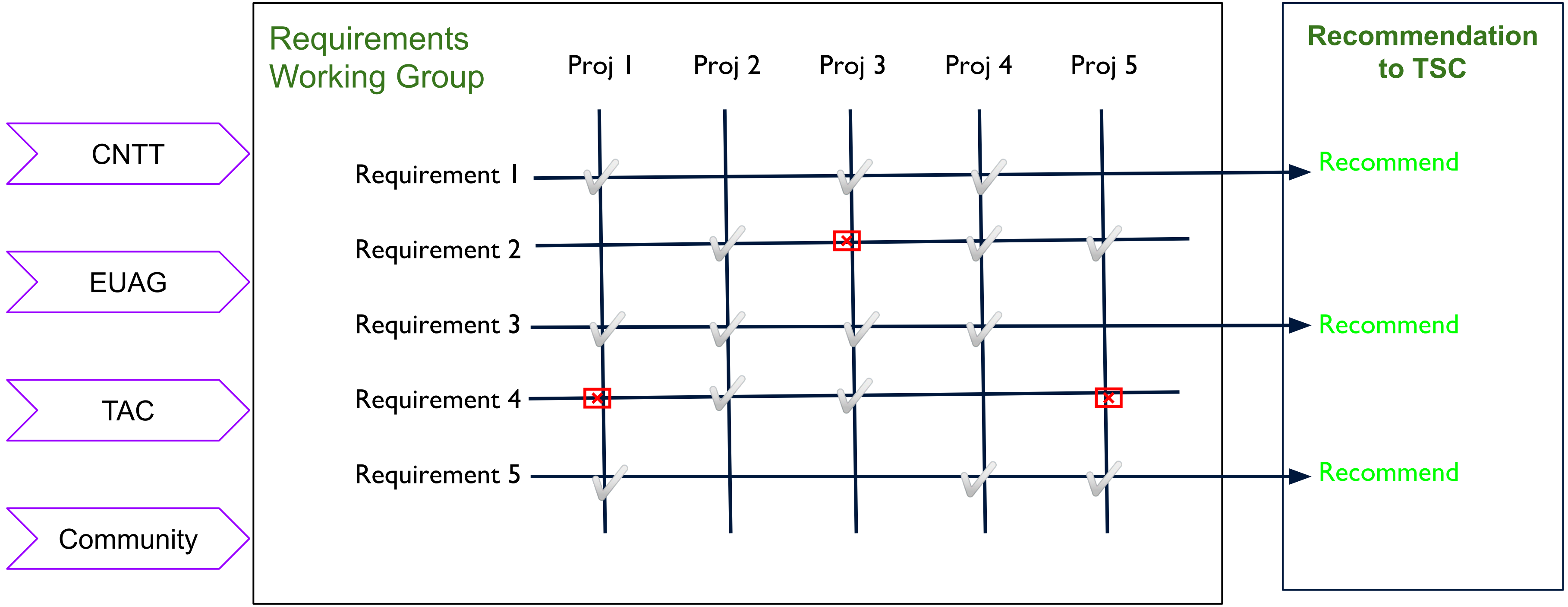
OPNFV Level Requirements

- Overview
  - OPNFV Release Requirements approved by TSC at M1/2/3
    - Requirement is de-scoped if not approved by the TSC
  - Projects agree to prioritize OPNFV level requirements
  - Each requirement has an owner and is documented in JIRA
  - TSC approval at M1 requires support commitment for each requirement from relevant projects
    - Support documented in project release plan

OPNFV Level Requirements

- Requirements Working Group or Subcommittee
  - Gathers requirements recommendations from community
  - Allocates requirements to releases
  - Ensures that requirements have support from affected projects
  - Ensures that requirements have an owner and are well documented in JIRA
  - Recommends a set of prioritized requirements to the TSC for approval for the current release

OPNFV Level Requirements



Project Release Plans

- Template based
- Reviewed and approved as part of release process
- Commitment to OPNFV-level requirements
  - Document how requirement will be met
- Other project objectives for the release
- Specify deliverables
- All work documented in JIRA and assigned to release

Independent Projects

- Definition
  - Not dependent on, or a dependency of, any other project in OPNFV
- Self-declared (if applicable)
- Requirements
  - OPNFV repo
  - CI using OPNFV resources
  - Documentation (TBD) and release notes
  - Self verification (project asserts readiness to release)

Independent Release

- All projects will maintain internal versioning
  - OPNFV release versioning will follow current practice of using the prefix "opnfv-" on version numbers to distinguish them.
- Projects will be required to contribute to OPNFV releases, approximately every 6 months, that meet OPNFV requirements established by the TSC and the release process.
- In addition, projects may release independently, using a Self-Release Process (TBD)

Documentation

- The current documentation is organized around the traditional OPNFV concept of “scenarios,” which is no longer a prominent aspect of OPNFV.
- Need to reconcile CNTT documentation with OPNFV documentation organization and process.
- Ask the DOCS project lead an effort, along with other stakeholders, to develop and propose new documentation structure to the TSC.
- Continue current practice of having milestone requirements for preliminary and final documentation as part of release process.

Integration and Gating

- An integration project will track project test and integration status, and will report this information to the release manager and to the TSC.

Milestones

- Projects must complete tasks at each milestone to be approved to proceed in the release
- Requirements are evaluated at each milestone to determine whether they remain feasible
- Milestones:
  - M0 - Start of Release
  - M1 - Planning
  - M2 - API / Functional Freeze
  - M3 - Code Freeze
  - RC0 - First Release Candidate
  - RCn - Final Release Candidate

Milestones: Planning (M0 ⇒ M1)

- Requirements gathered, reviewed, and approved by TSC
- Project release plans completed, reviewed, and approved
- All work planned for the release is documented in JIRA and assigned to the release (fix version field)
- Risks documented

Milestones: API & Functional Freeze (M1 ⇒ M2)

- Resolve integration blocking issues
- Resolve license scan issues
- Update risks documentation

Milestones: Code Freeze (M2 ⇒ M3)

- Resolve high priority JIRA issues
- Complete preliminary documentation
- Update risks documentation

Milestones: Release Candidate 0, 1, …, x

- Verify release plan, including all planned testing, has been completed
- Resolve high priority JIRA issues
- Prepare and verify release artifacts
- Complete final documentation
- Complete tagging

ToDo

- Establish requirements working group
- Establish integration management project
- Determine new documentation layout/organization
- Develop self-release process
- Develop project release plan template