Invenio Development Practices

Tibor Šimko
<tibor.simko@cern.ch>

Department of Information Technology
CERN

Invenio User Group Workshop 2012
CERN, May 7–9 2012

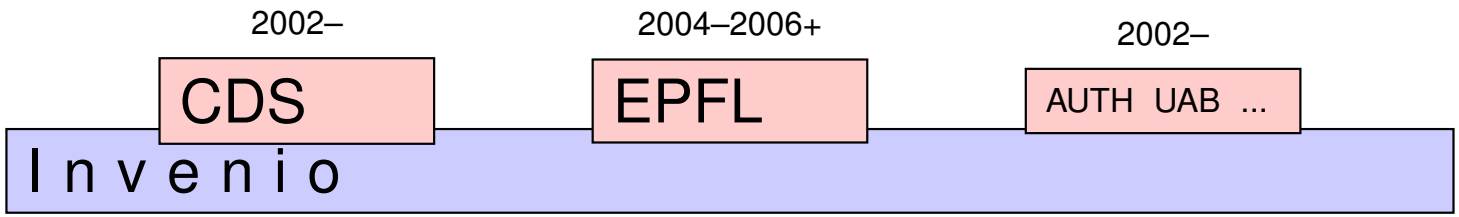
Outline

Outline

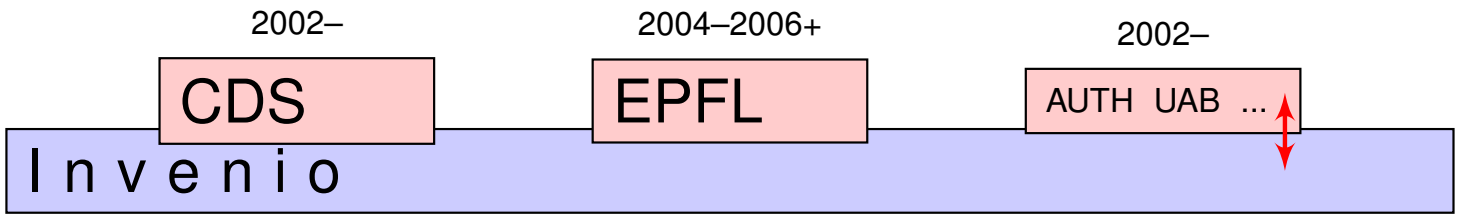
Developer Community

- Invenio source code base:
  - 35+ modules
  - 330,000+ lines of code
- Invenio developer community:
  - 91 authors and contributors since 2002 (including I18N translators)
  - many short-term students, importance of QA practices
- in 2011:
  - 710 commits from 38 developers and contributors

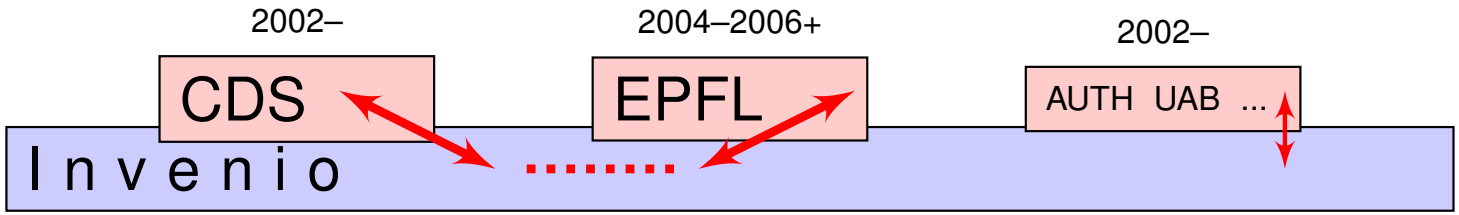
Developer Community



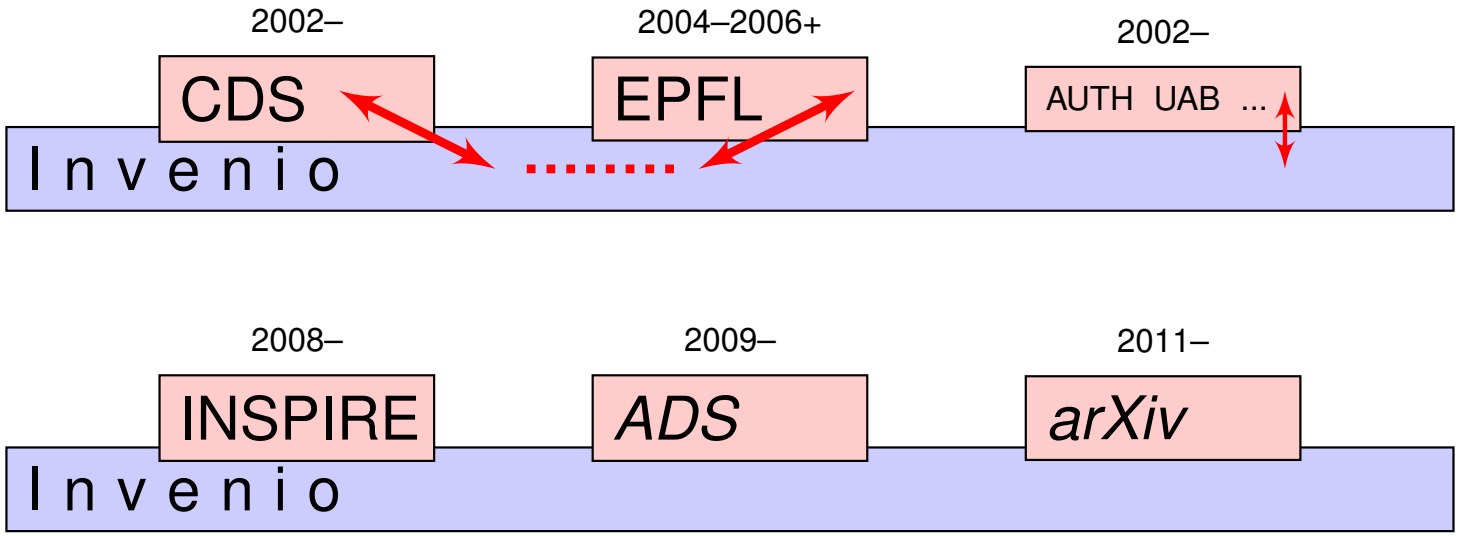
Developer Community



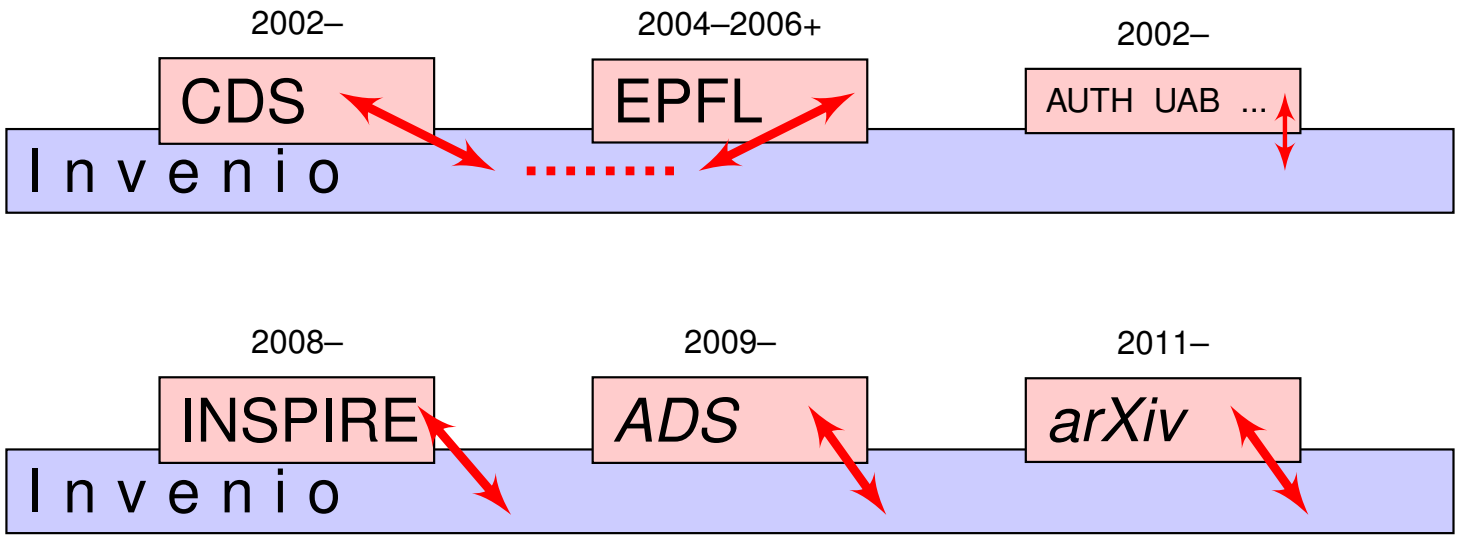
Developer Community



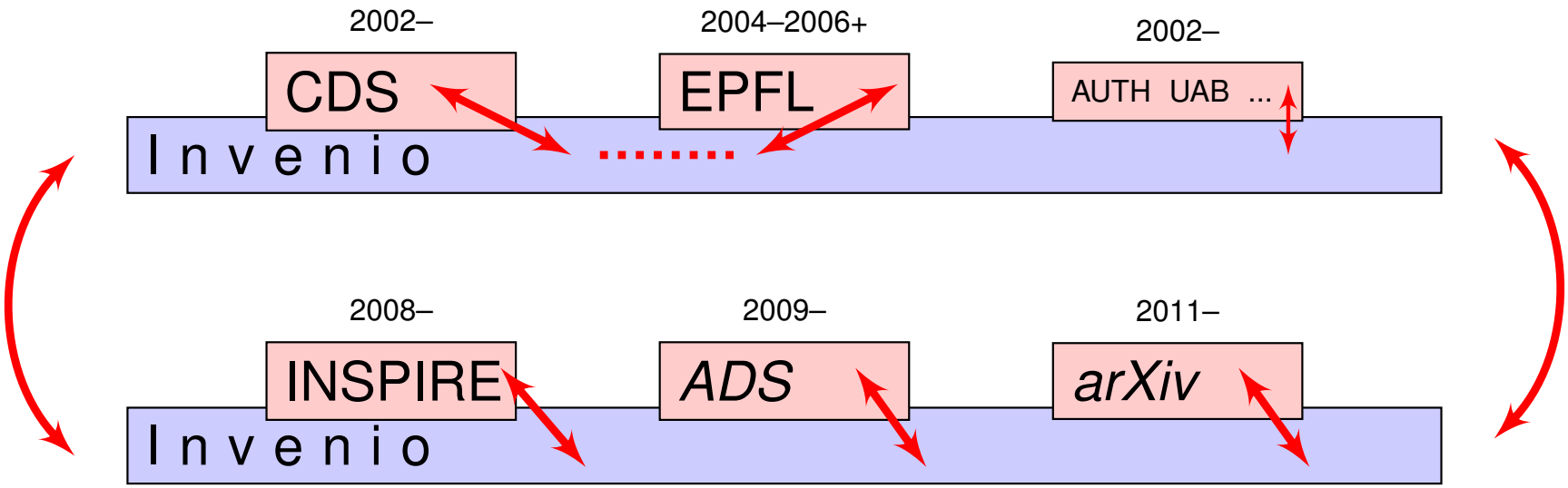
Developer Community



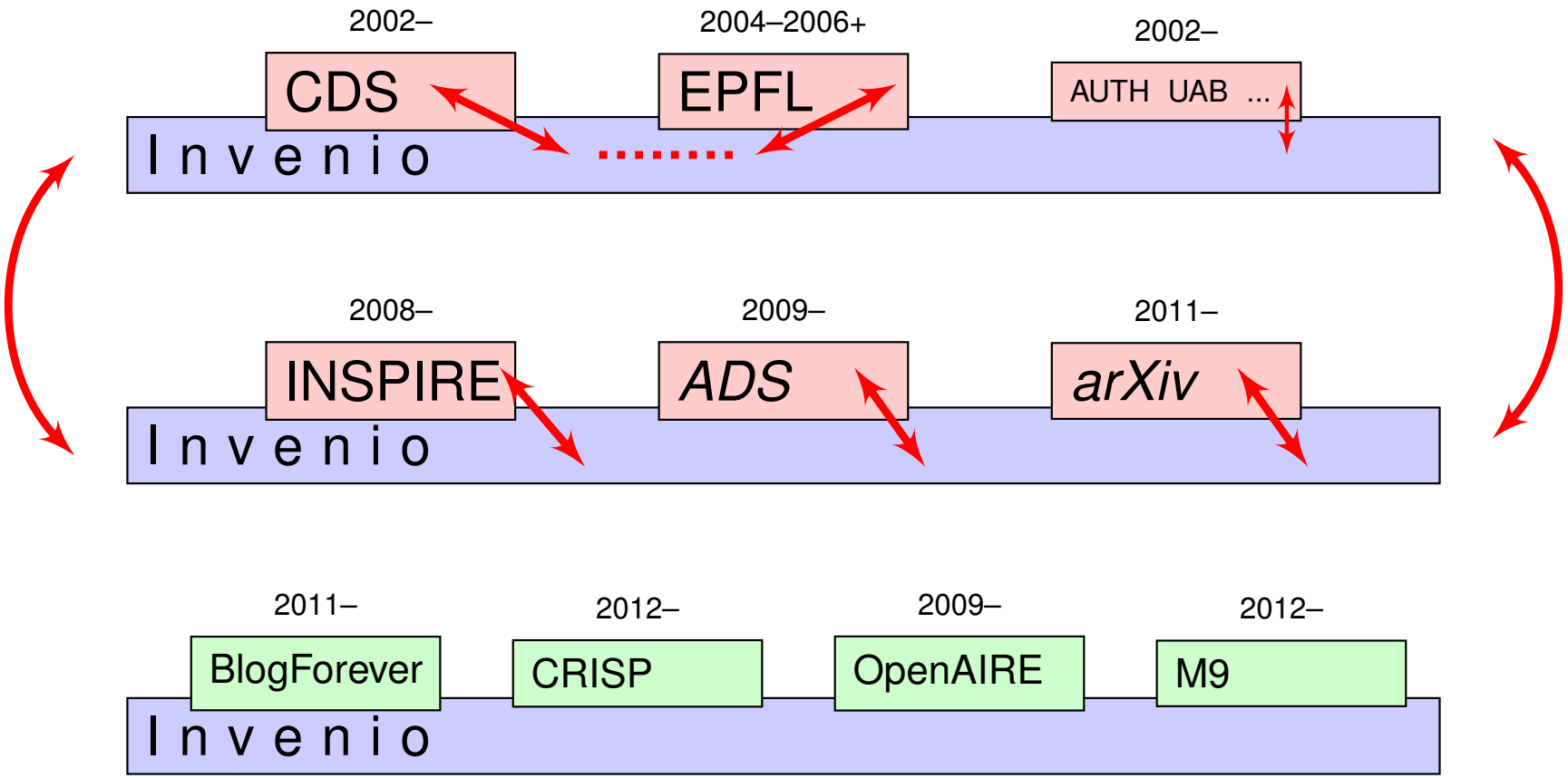
Developer Community



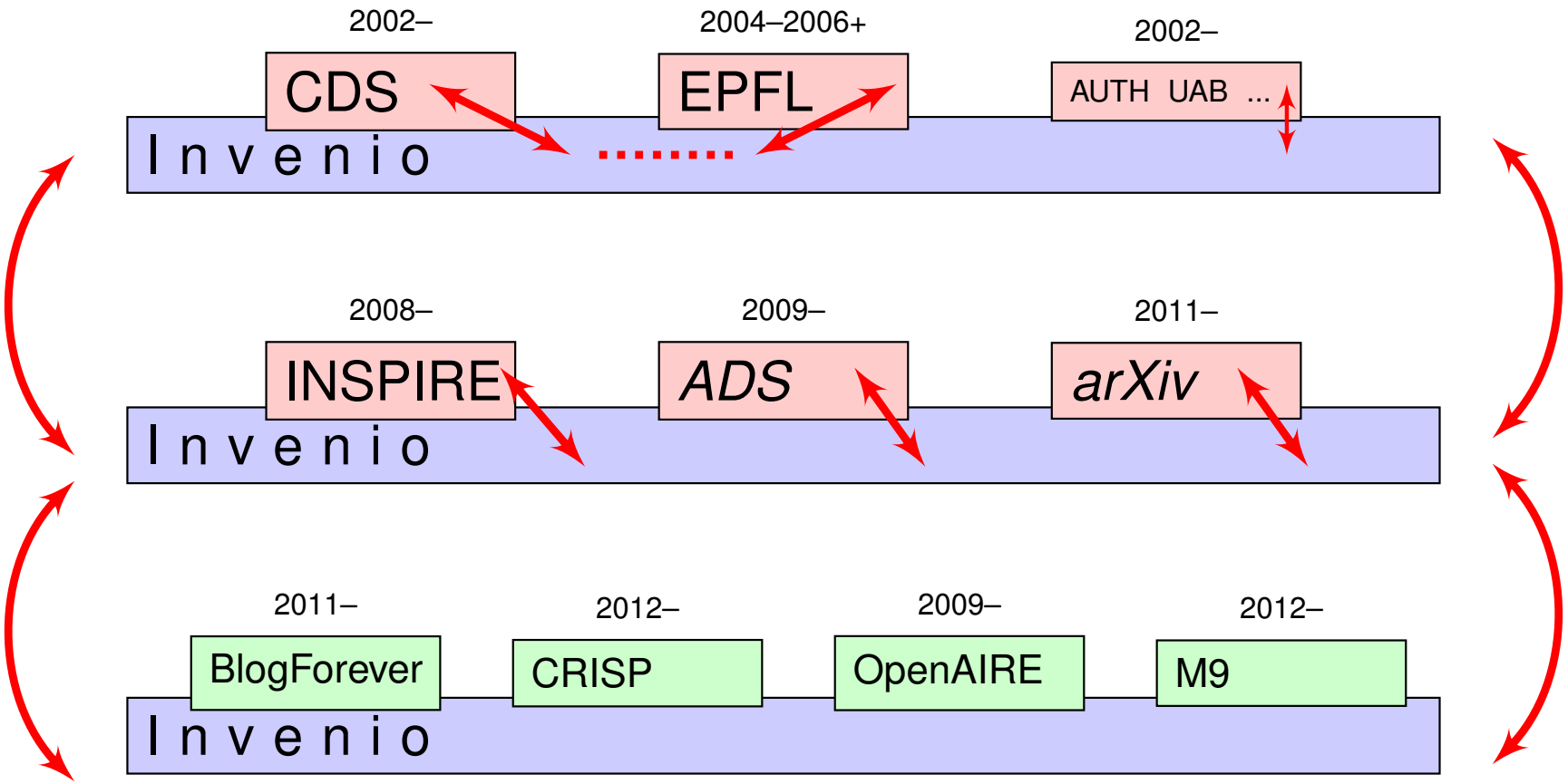
Developer Community



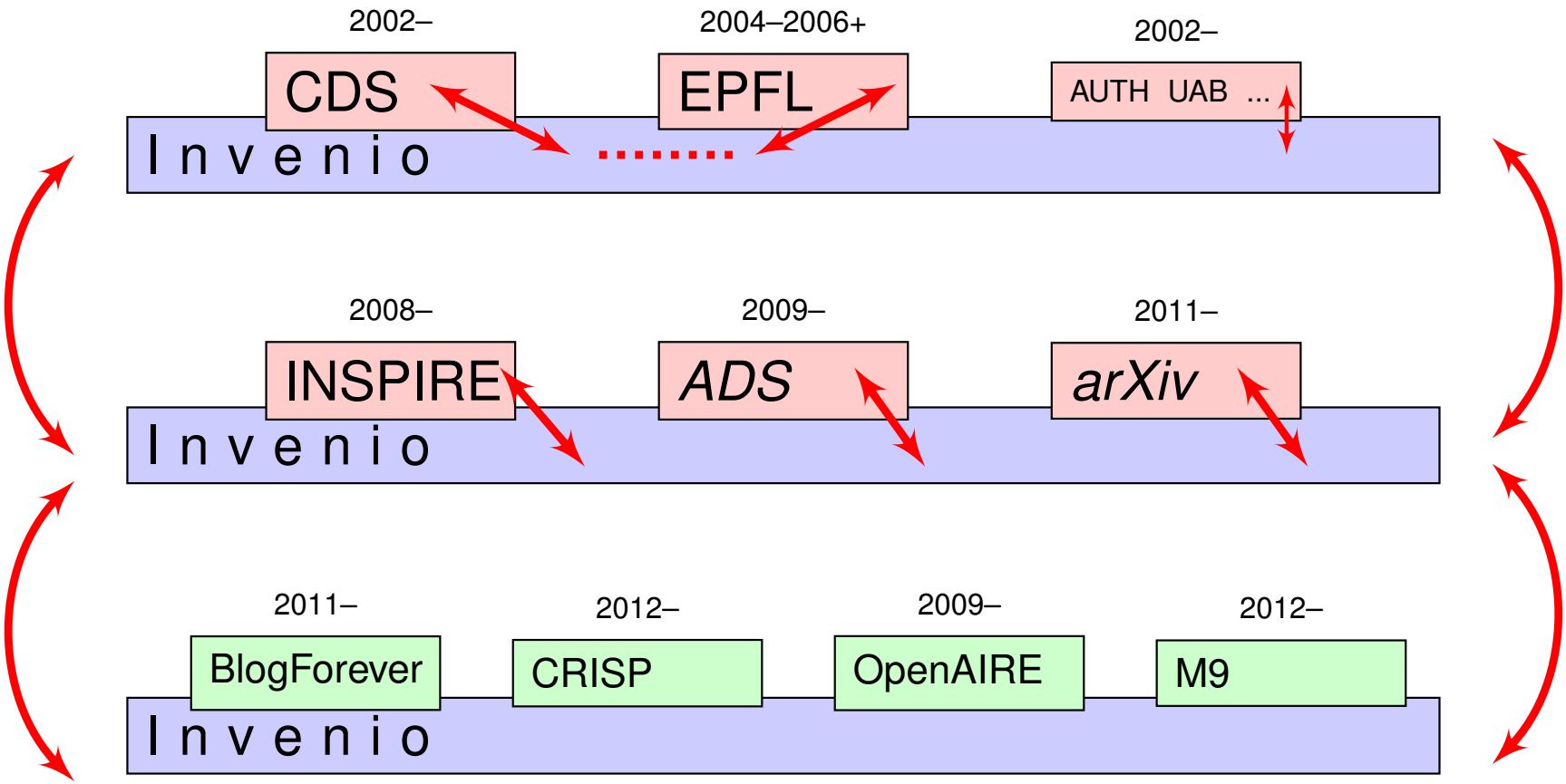
Developer Community



Developer Community



Developer Community



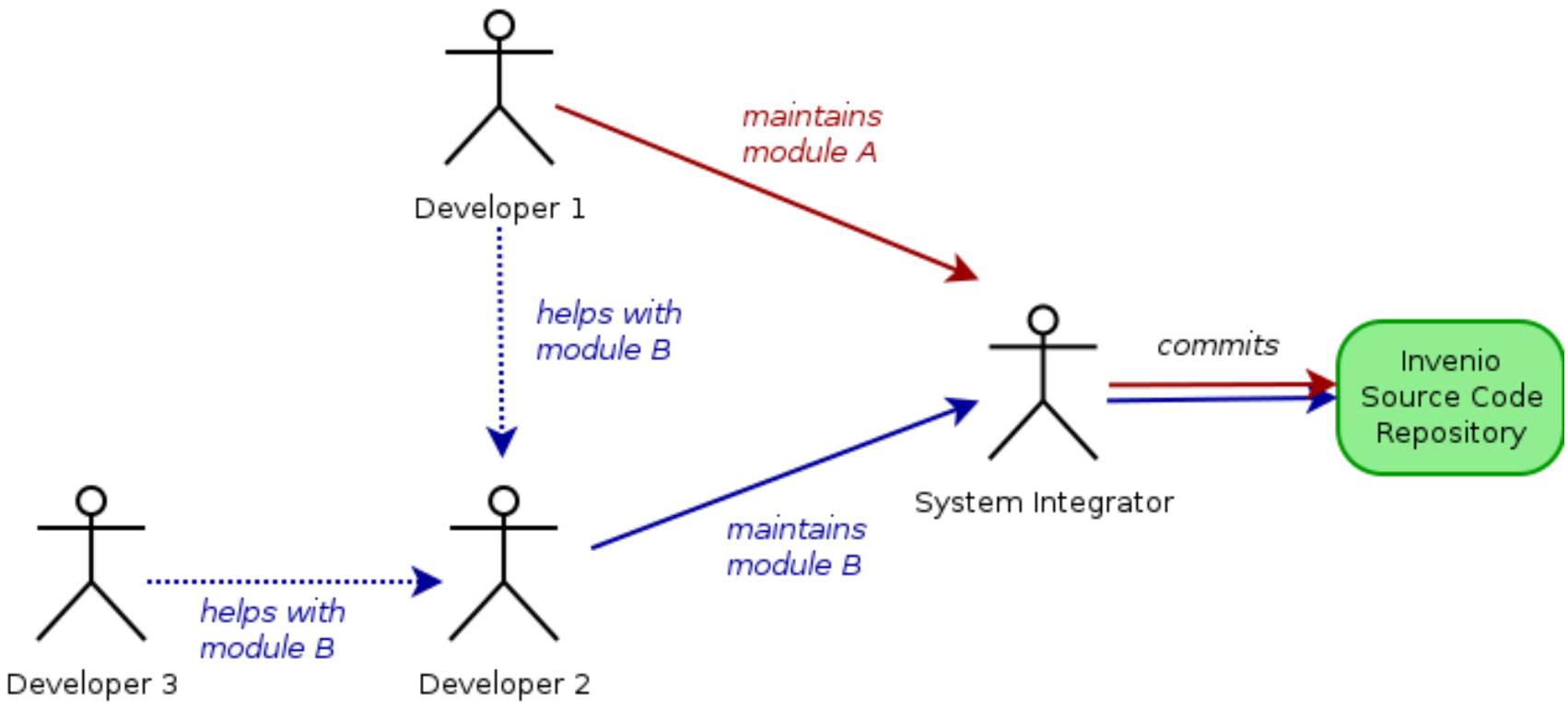
300k LOC - Invenio core sources

10k LOC - INSPIRE overlay sources

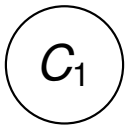
Outline

Git Collaboration

- **pull-on-demand** collaboration model
- inherent code review and QA processes before integration
- modules maintainers aka “integration lieutenants”

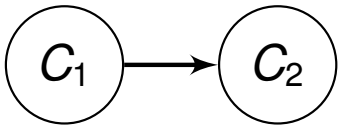


Git Branches



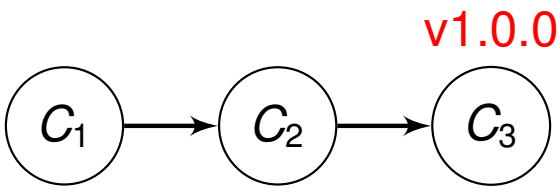
master

Git Branches



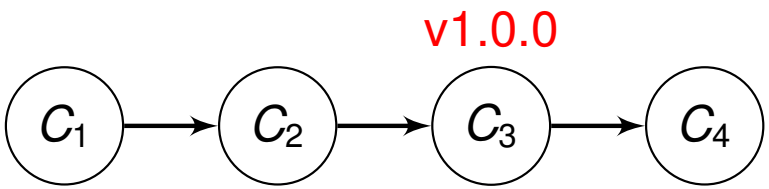
master

Git Branches

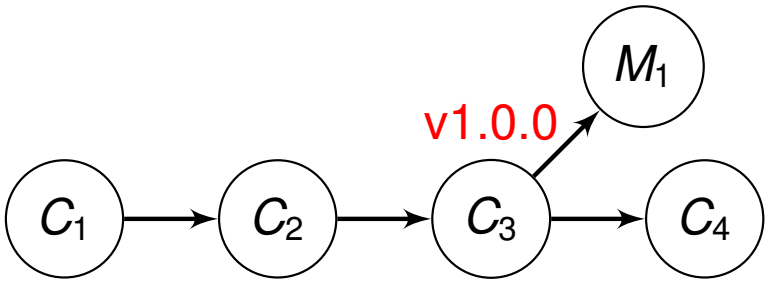


master

Git Branches



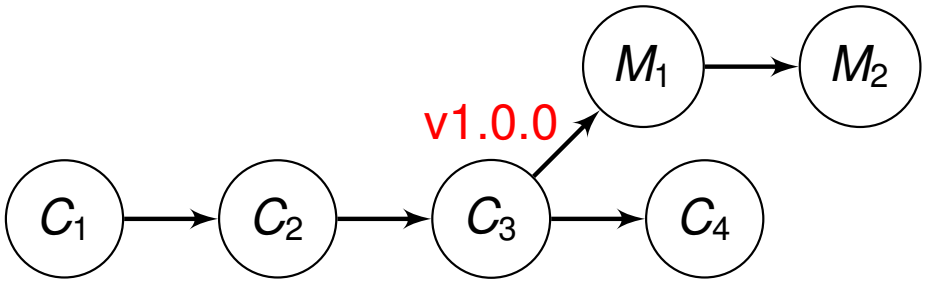
Git Branches



maint-1.0

master

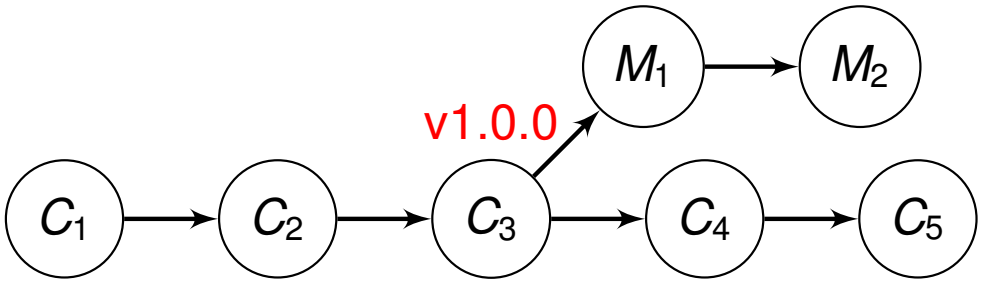
Git Branches



maint-1.0

master

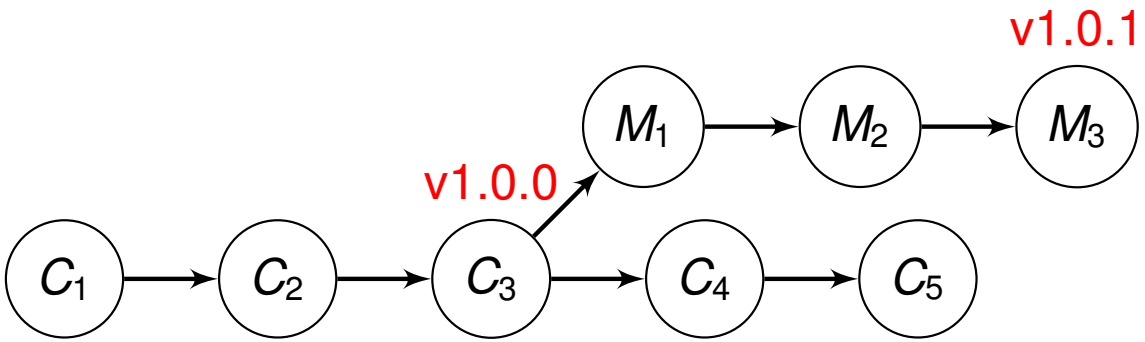
Git Branches



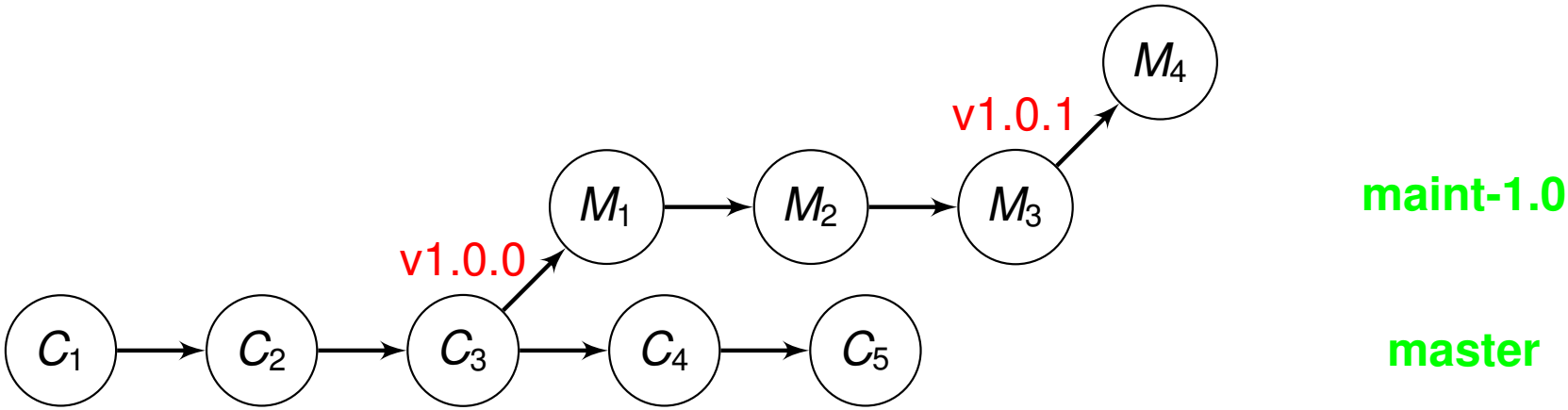
maint-1.0

master

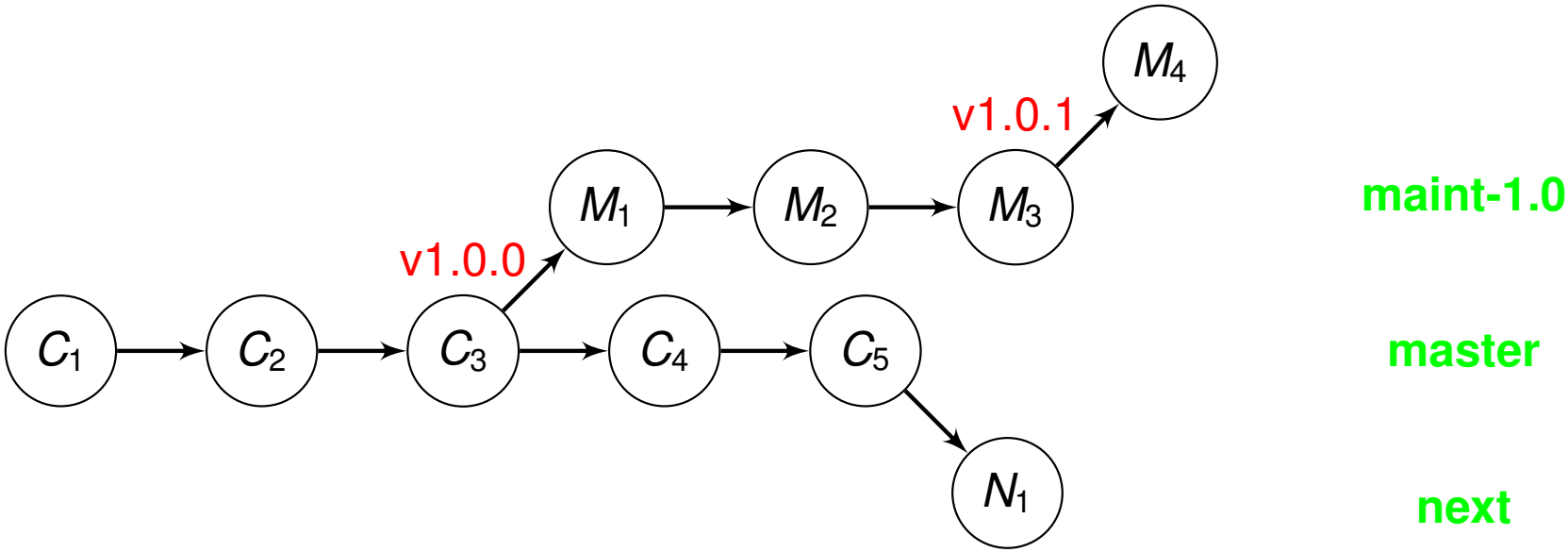
Git Branches



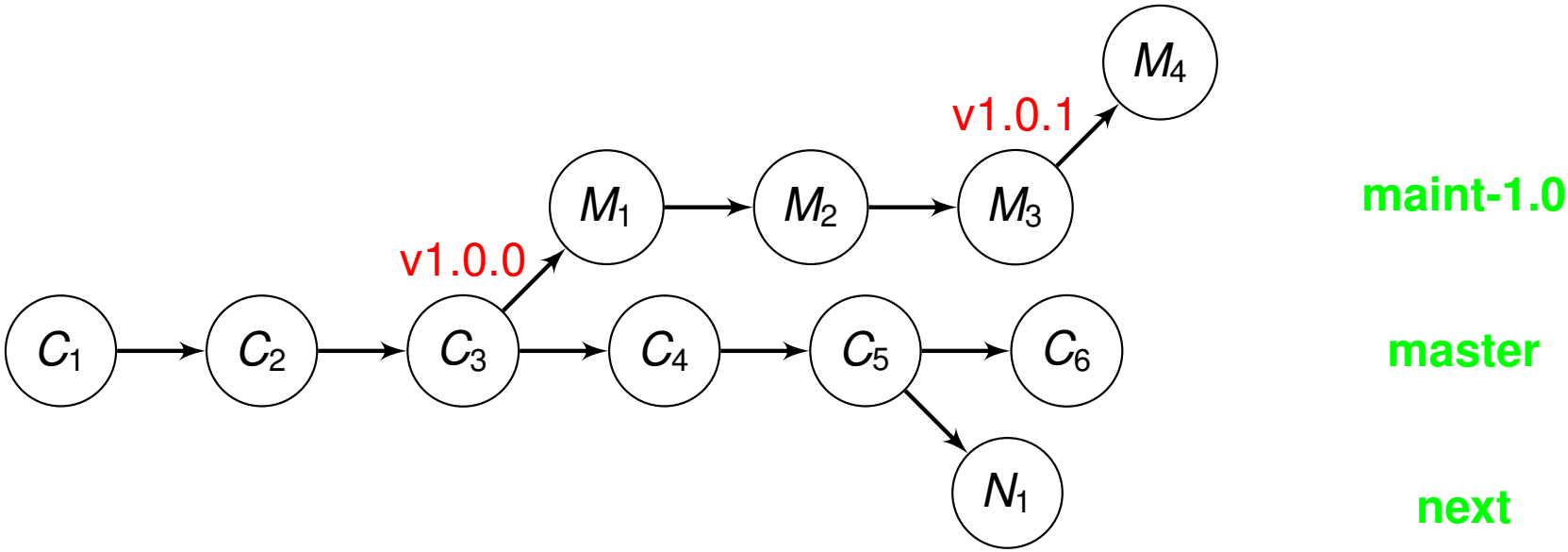
Git Branches



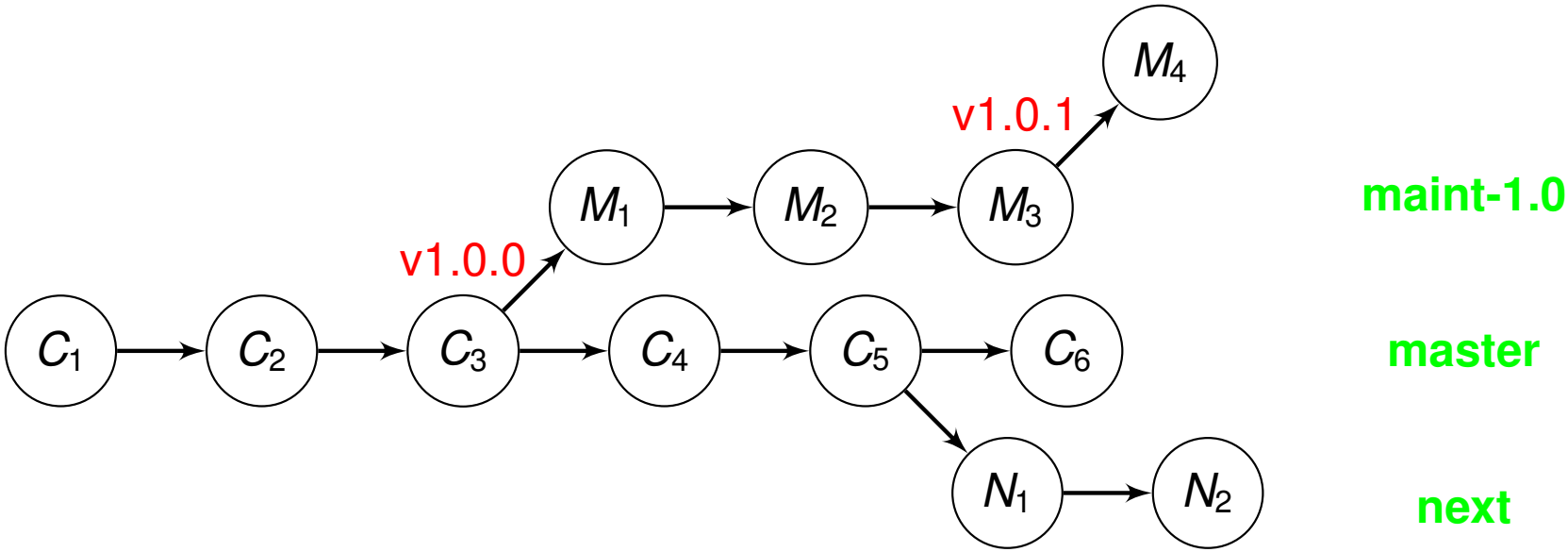
Git Branches



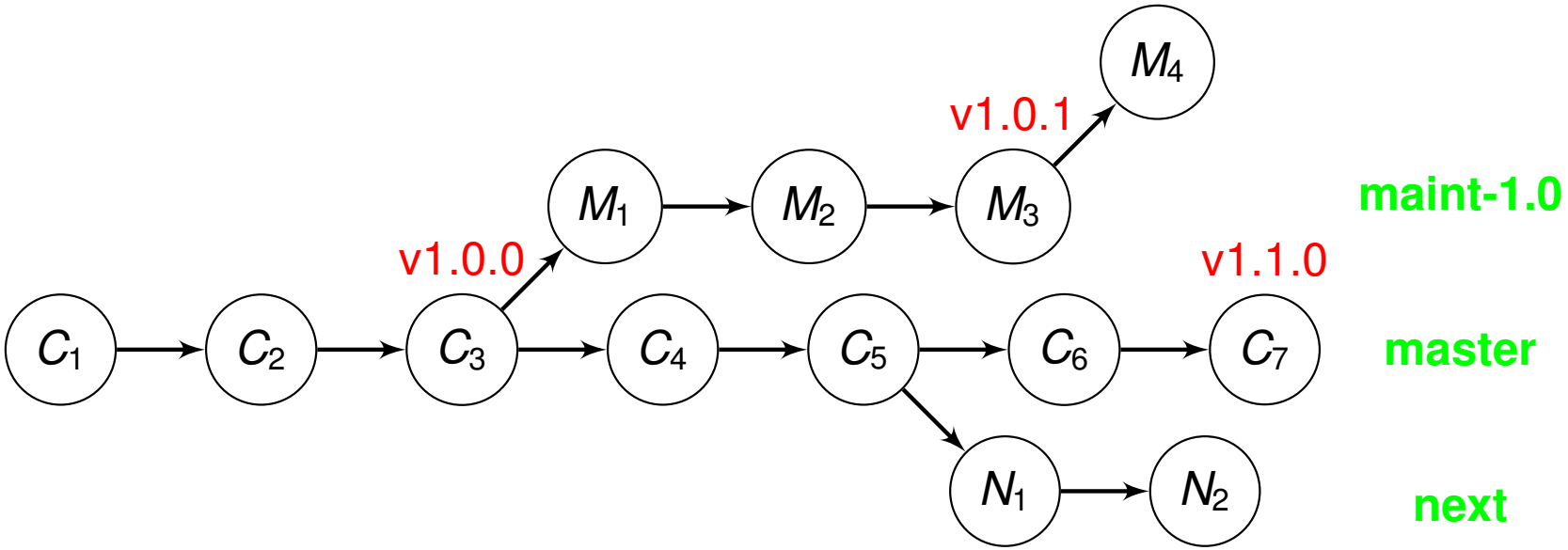
Git Branches



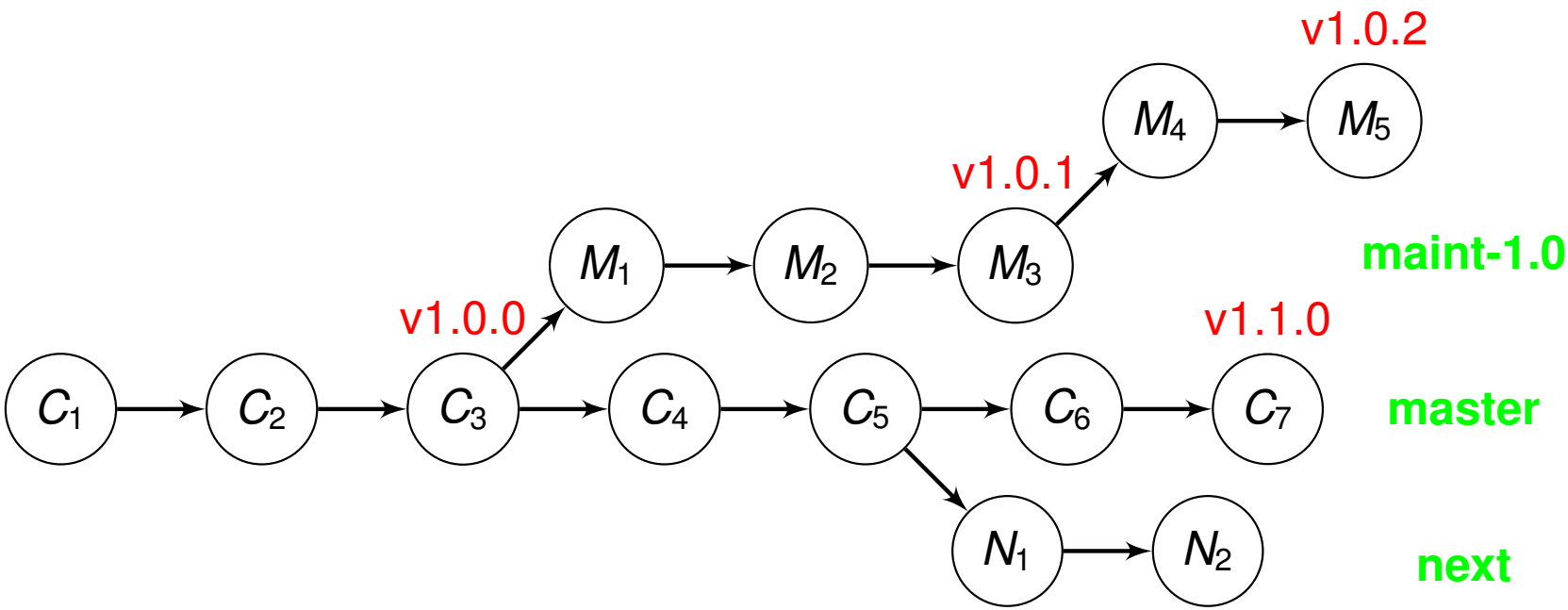
Git Branches



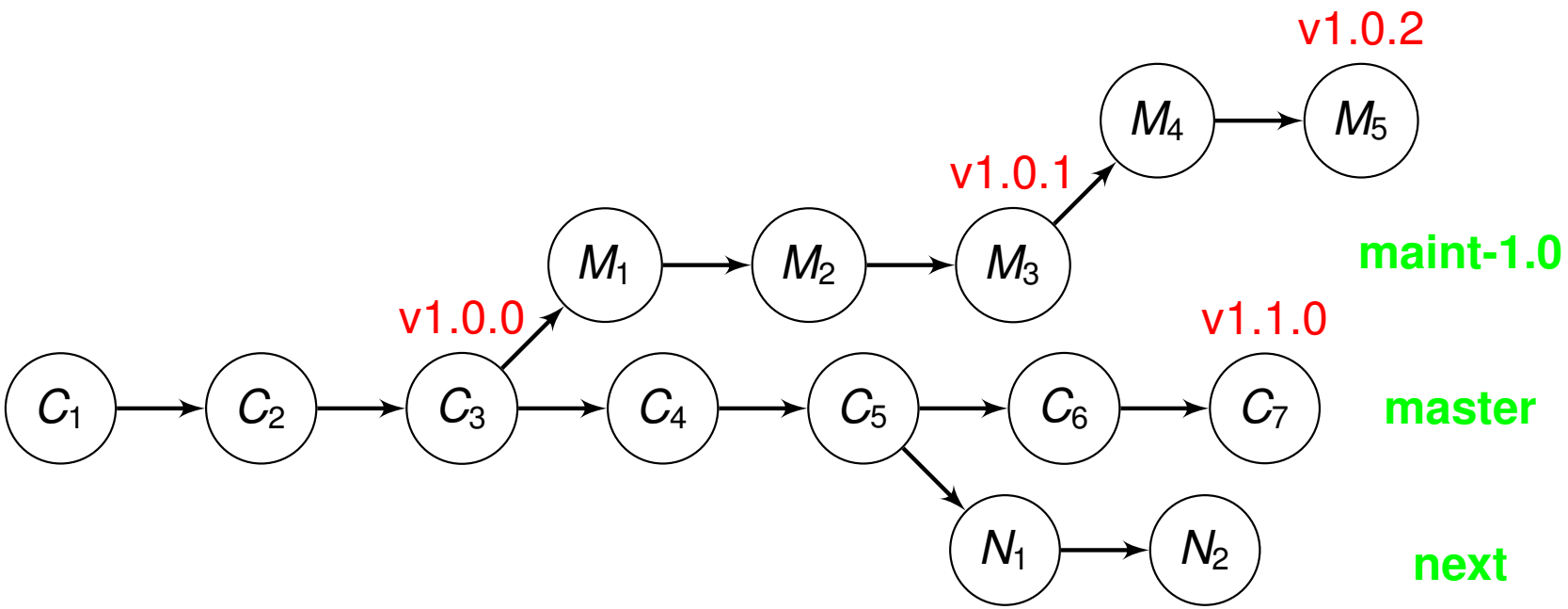
Git Branches



Git Branches



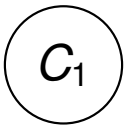
Git Branches



- **maint-X.Y** — release maintenance branches
- **master** — new feature branch
- **next** — things not yet release-ready

Git Development

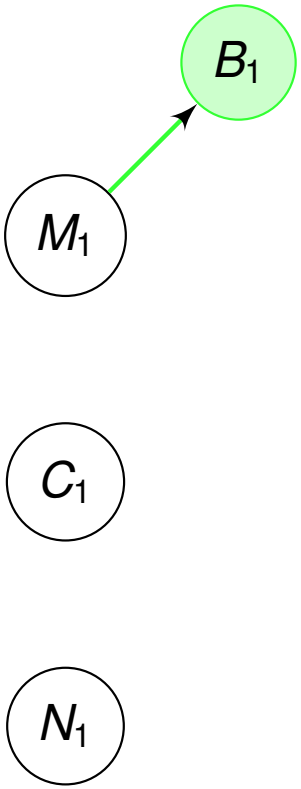
maint-1.0



master

next

Git Development



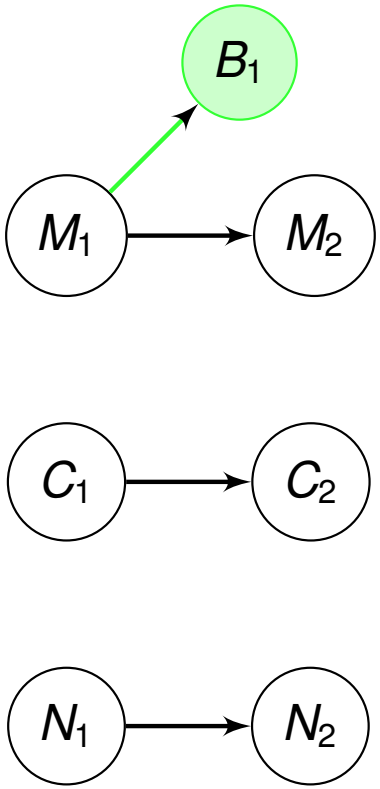
some-bugfix

maint-1.0

master

next

Git Development



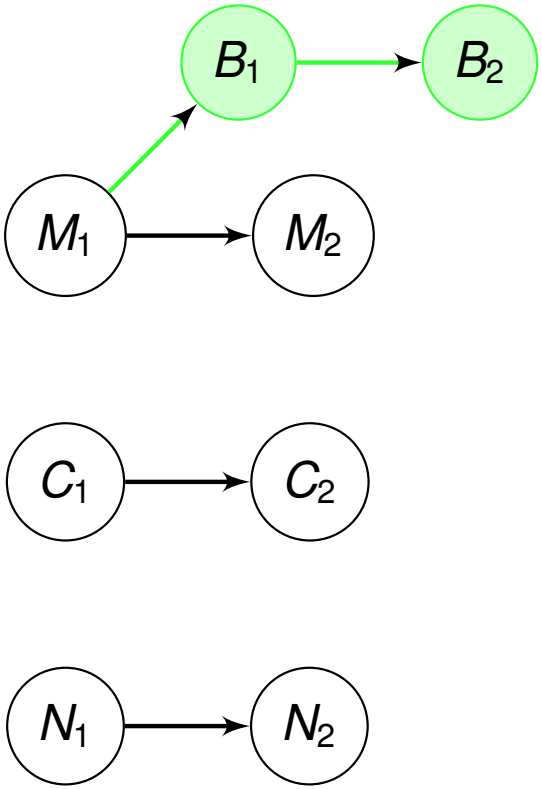
some-bugfix

maint-1.0

master

next

Git Development



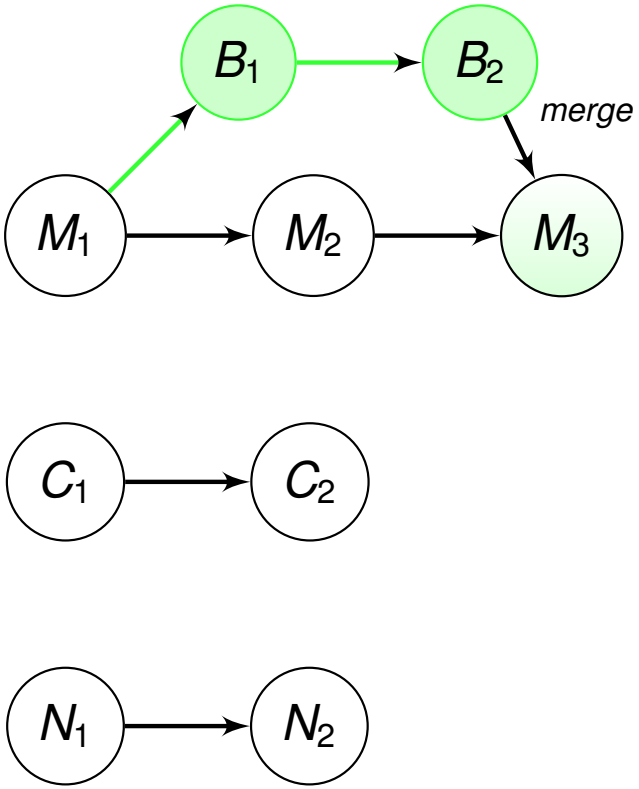
some-bugfix

maint-1.0

master

next

Git Development



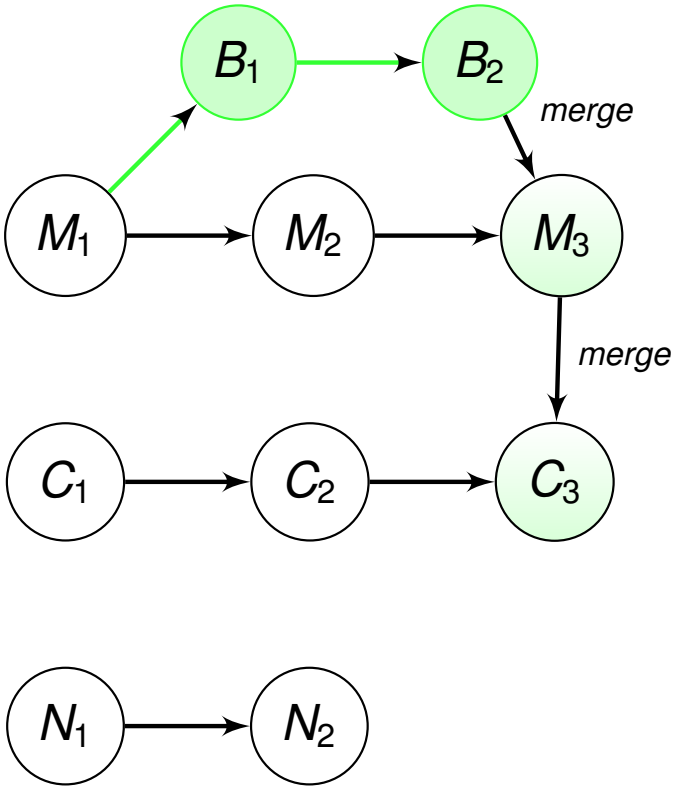
some-bugfix

maint-1.0

master

next

Git Development



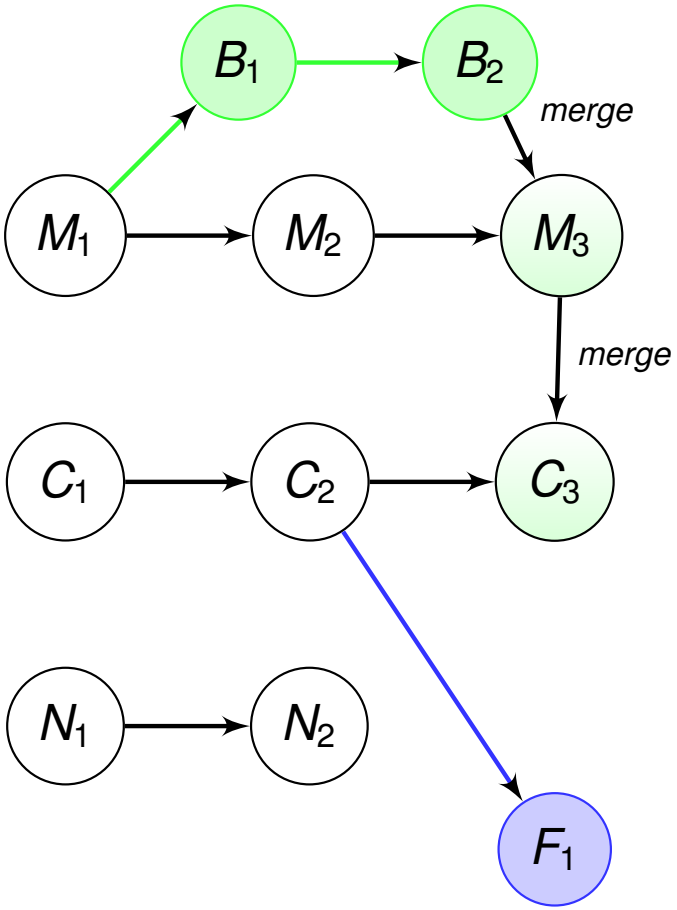
some-bugfix

maint-1.0

master

next

Git Development



some-bugfix

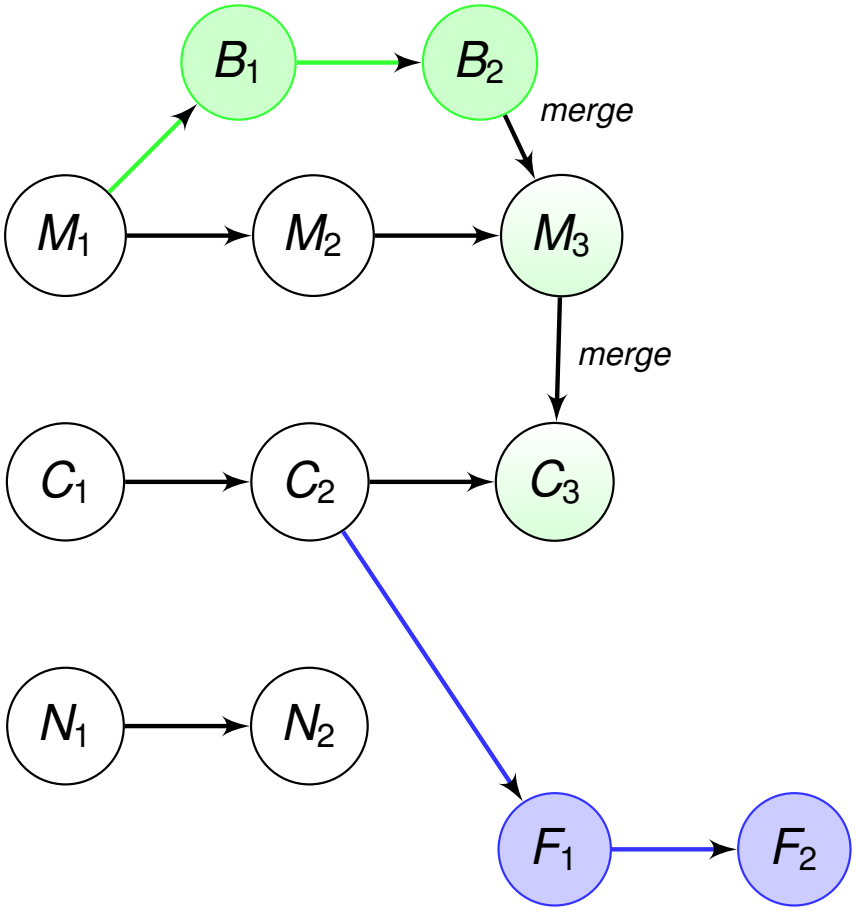
maint-1.0

master

next

some-new-feature

Git Development



some-bugfix

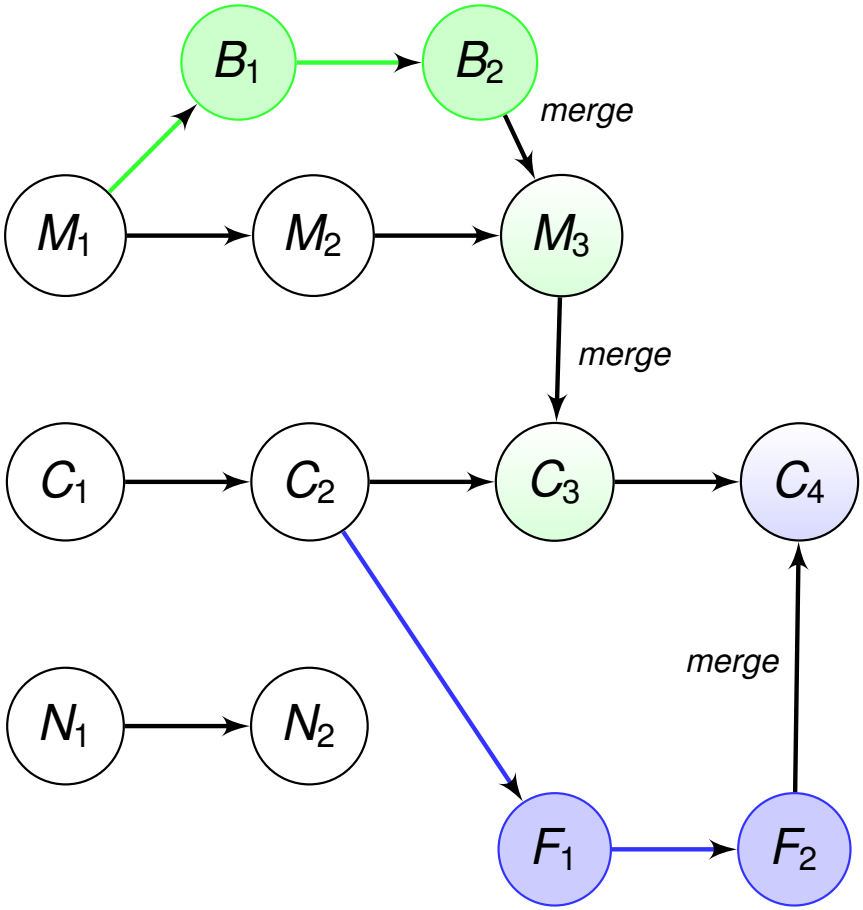
maint-1.0

master

next

some-new-feature

Git Development



some-bugfix

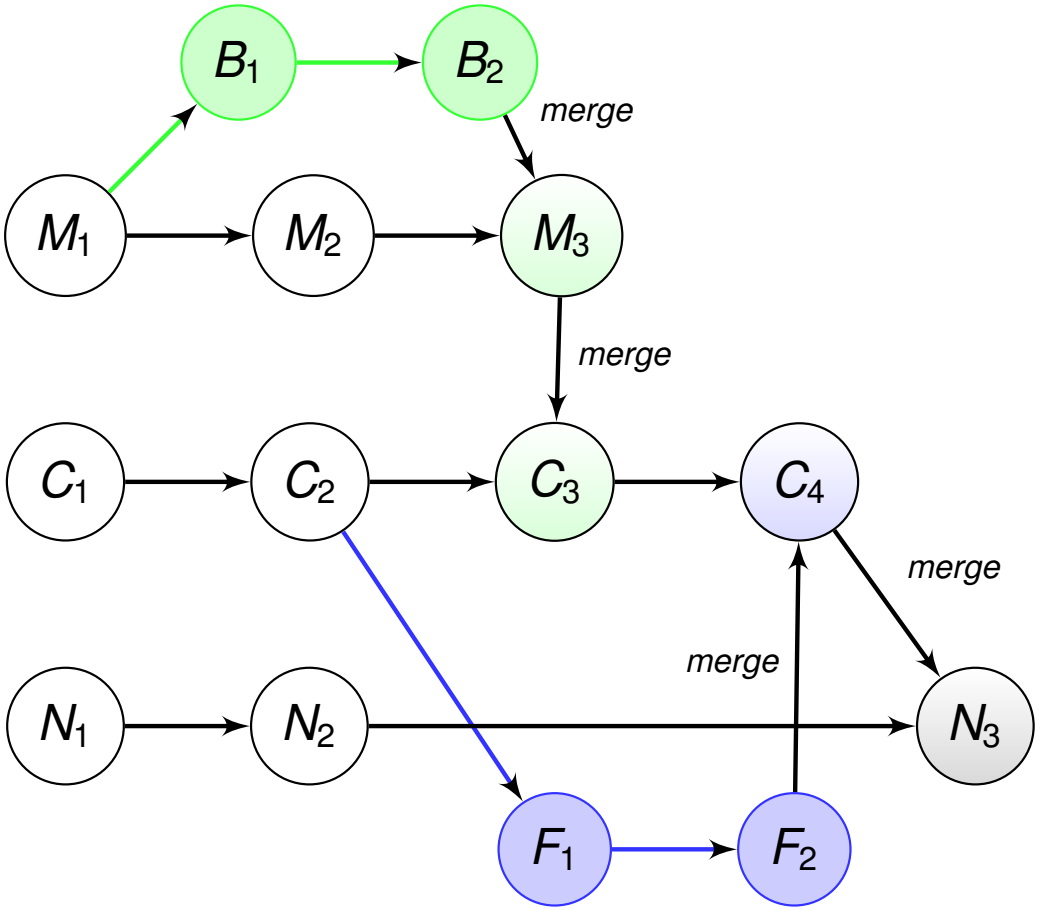
maint-1.0

master

next

some-new-feature

Git Development



some-bugfix

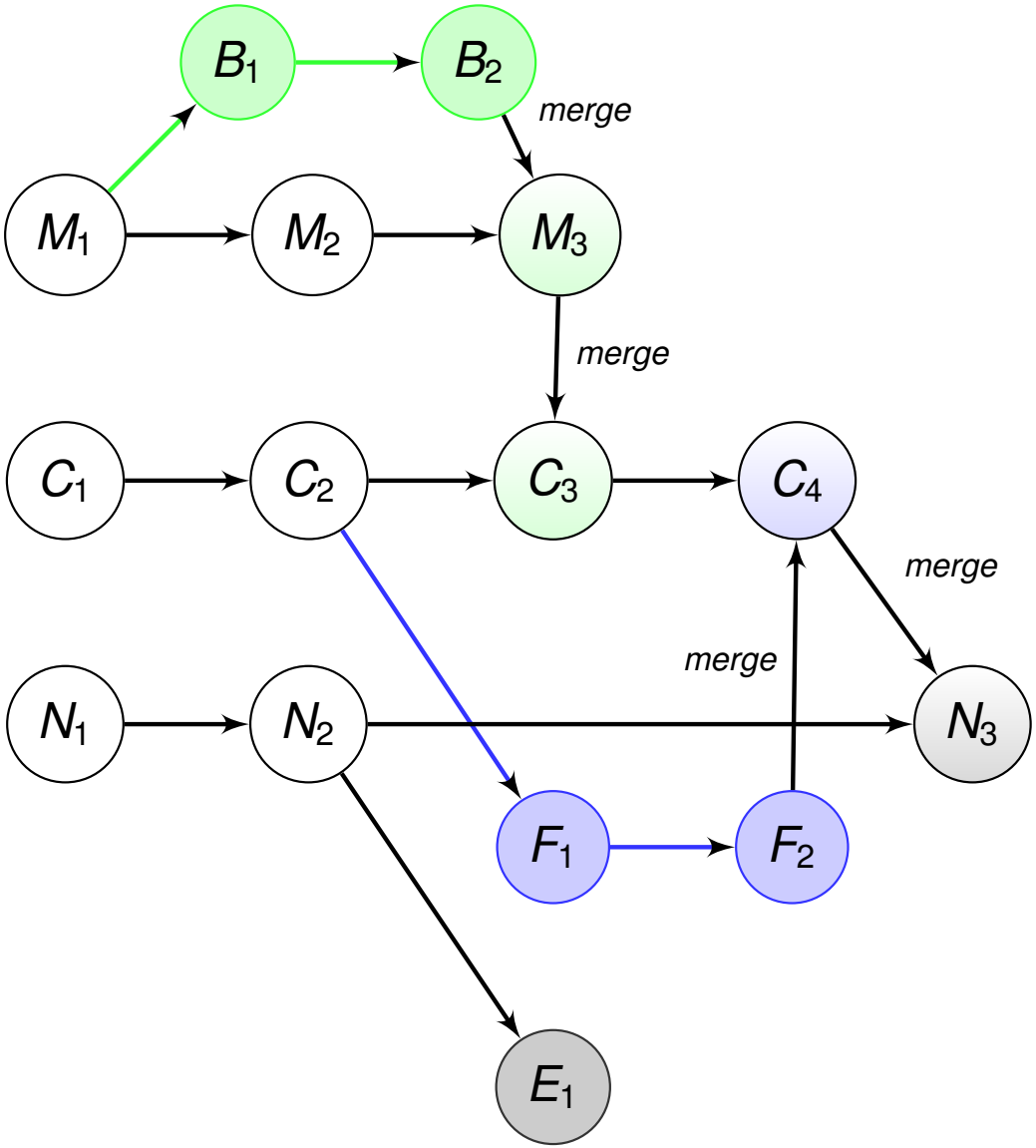
maint-1.0

master

next

some-new-feature

Git Development



some-bugfix

maint-1.0

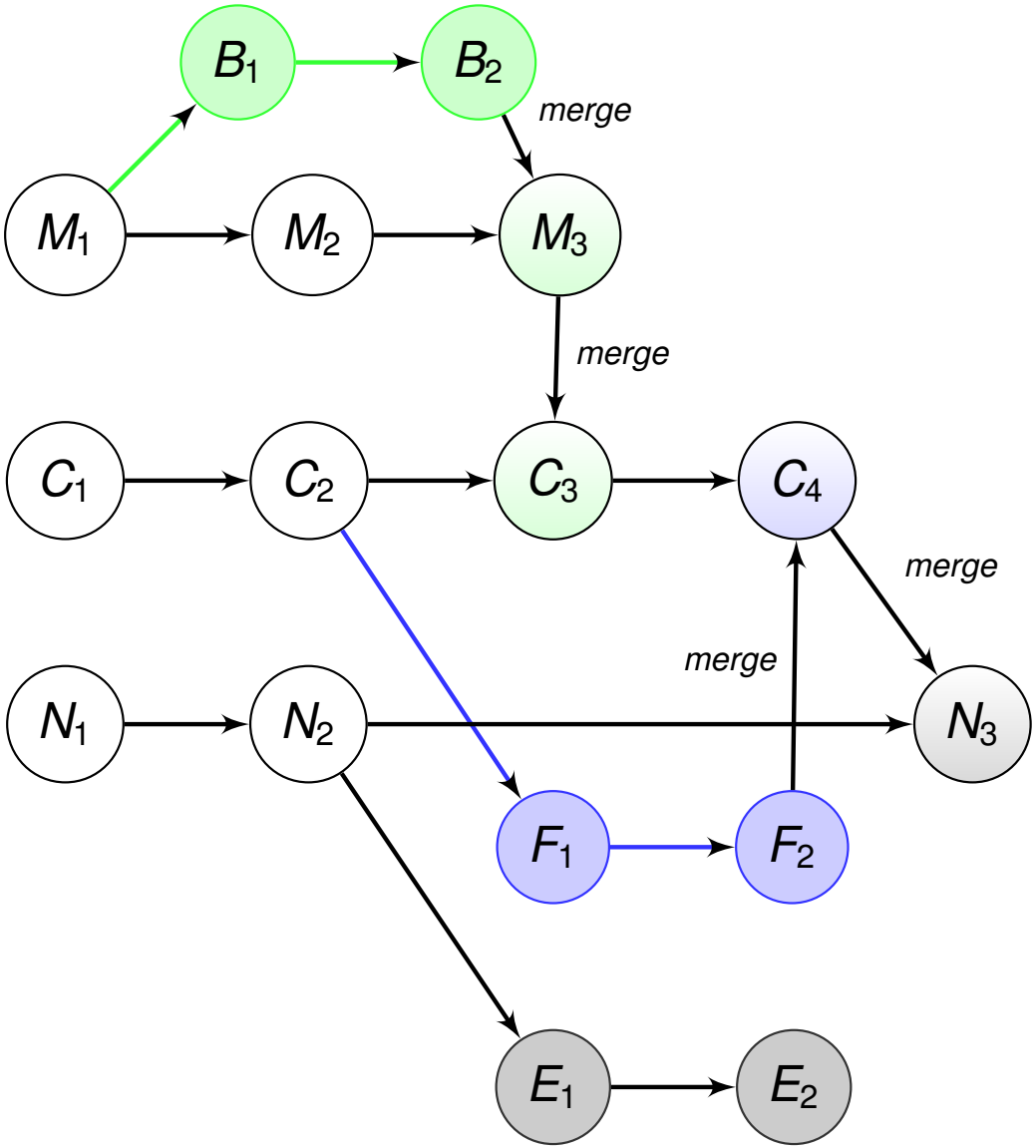
master

next

some-new-feature

some-experimental-feature

Git Development



some-bugfix

maint-1.0

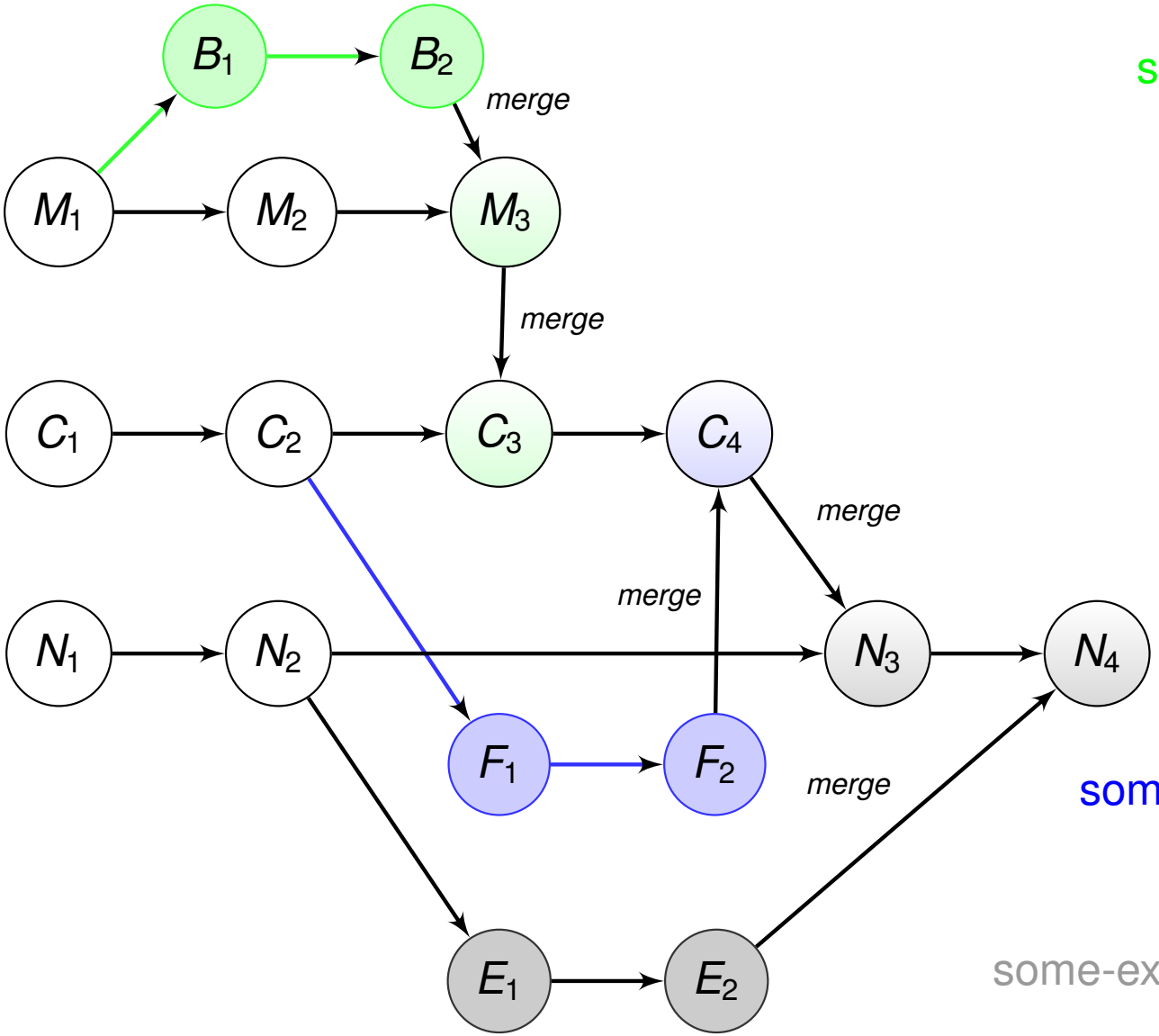
master

next

some-new-feature

some-experimental-feature

Git Development



Outline

Code Quality

- code kwalitee:
  - `invenio-check-kwalitee`
  - `invenio-check-branch`
  - integrate QA tools in your development workflow
- testing:
  - demo site, demo collections, small record sample
  - 610 unit tests
  - 518 regression/functional/acceptance/web tests
  - continuous testing with Bitten
- . . . demo

Unit testing

- **test-driven development** when appropriate
- e.g. before/while developing `strip_accents()`, write:

Example: search_engine_tests.py

```
class TestStripAccents(unittest.TestCase):
    """Test for handling of UTF-8 accents."""

    def test_strip_accents(self):
        """search engine - stripping of accented letters"""
        self.assertEqual("memememe",
             search_engine.strip_accents('mémêmëmè'))
        self.assertEqual("MEMEMEME",
             search_engine.strip_accents('MÉMÊMËMÈ'))
```

Functional testing

- functional/acceptance/regression testing
- testbed site (Atlantis of Institute Fictive Science)
- e.g. Python **mechanize** module to emulate browser

Example: websearch_regression_tests.py

```
class WebSearchSearchEnginePythonAPITest(unittest.TestCase):
    "Check typical search engine Python API calls on the demo data."

    def test_search_engine_python_api_for_failed_query(self):
        "websearch - search engine Python API for failed query"
        self.assertEqual([],
                         perform_request_search(p='aoeuidhtns'))

    def test_search_engine_python_api_for_successful_query(self):
        "websearch - search engine Python API for successful query"
        self.assertEqual([8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
                          18, 47],
                         perform_request_search(p='ellis'))
```

Web testing

- sometimes we need to run tests in real browser
  - e.g. pages with heavy JavaScript
- using **Selenium** extension for Firefox
  - record and replay browser actions
  - test for text existence or non-existence of pages
  - test for link labels and targets

Example: websearch_web_tests.py

```
class InvenioWebSearchWebTests(InvenioWebTestCase):

    def test_search_ellis(self):
        """websearch - web test search for ellis"""
        self.browser.get(CFG_SITE_URL)
        p = self.browser.find_element_by_name("p")
        p.send_keys("ellis")
        p.submit()
        self.page_source_test(expected_text=[
            'Thermal conductivity of dense quark matter ' + \
            'and cooling of stars'])
```

Outline

How to Organise Local Developments

- vanilla upstream Invenio sources (300k LOC)
  (a) using point release `invenio-1.0.0.tar.gz`
  (b) tracking some *maint-X.Y* branch
- site-specific overlay repository (10k LOC) typically contains:
  - custom site style CSS, images, template skin
  - output formatting rules, templates, elements
  - submission masks, conversion templates
  - knowledge bases, curation templates
  - ranking configurations, sorting configurations
- maintaining local differences
  (a) using `quilt` to maintain patches; use case of UAB
  (b) using full `git` power; example for INSPIRE:
    + Invenio DEV
    + Invenio OPS
    + INSPIRE
- . . . whiteboard draft and demo

Outline

Deployment Practices

- production logbook
- deploy from point releases
  - e.g. from `invenio-1.0.0.tar.gz` to `invenio-1.0.2.tar.gz`
  - easy to follow `RELEASE-NOTES`
- deploy from git branches
  - e.g. track *maint-1.0* branch
  - `invenio-create-deploy-recipe`
  - Fabric
- fully-automated vs human-assisted deployment

Outline

How to Contribute Code

- check code kwalitee, include test cases with the code
- respect minimal requirements, e.g. write for `Python-2.4`
  - use `vagrant` virtual development environment
- make conditional use of optional dependencies, e.g. `feedparser`
- write against Atlantis defaults, not against local custom stuff
- make things easily configurable and reusable by others
  - `get_tag_from_name('journal title')` vs hard-coded `773__p`
  - enable/disable custom features e.g. via RBAC access control
  - configurability via `CFG_BIBFOO_XYZZY` variables
  - configurability of code via `pluginutils`
- submit patch against *maint-X.Y* or *master*
  - create ticket
  - send patch by email
  - publish a branch
- share needs and requirements with others
- share solutions and how-to recipes with others
  - `http://invenio-software.org/wiki/HowTo/HowToChangeSiteUrl`