Experience in Software Development and Management

Xiuqin Wu

IPAC, Caltech

Overview

- Issue tracking systems (features and bug fixes)
- Version control systems
- Development processes
- Code review
- Integration and test

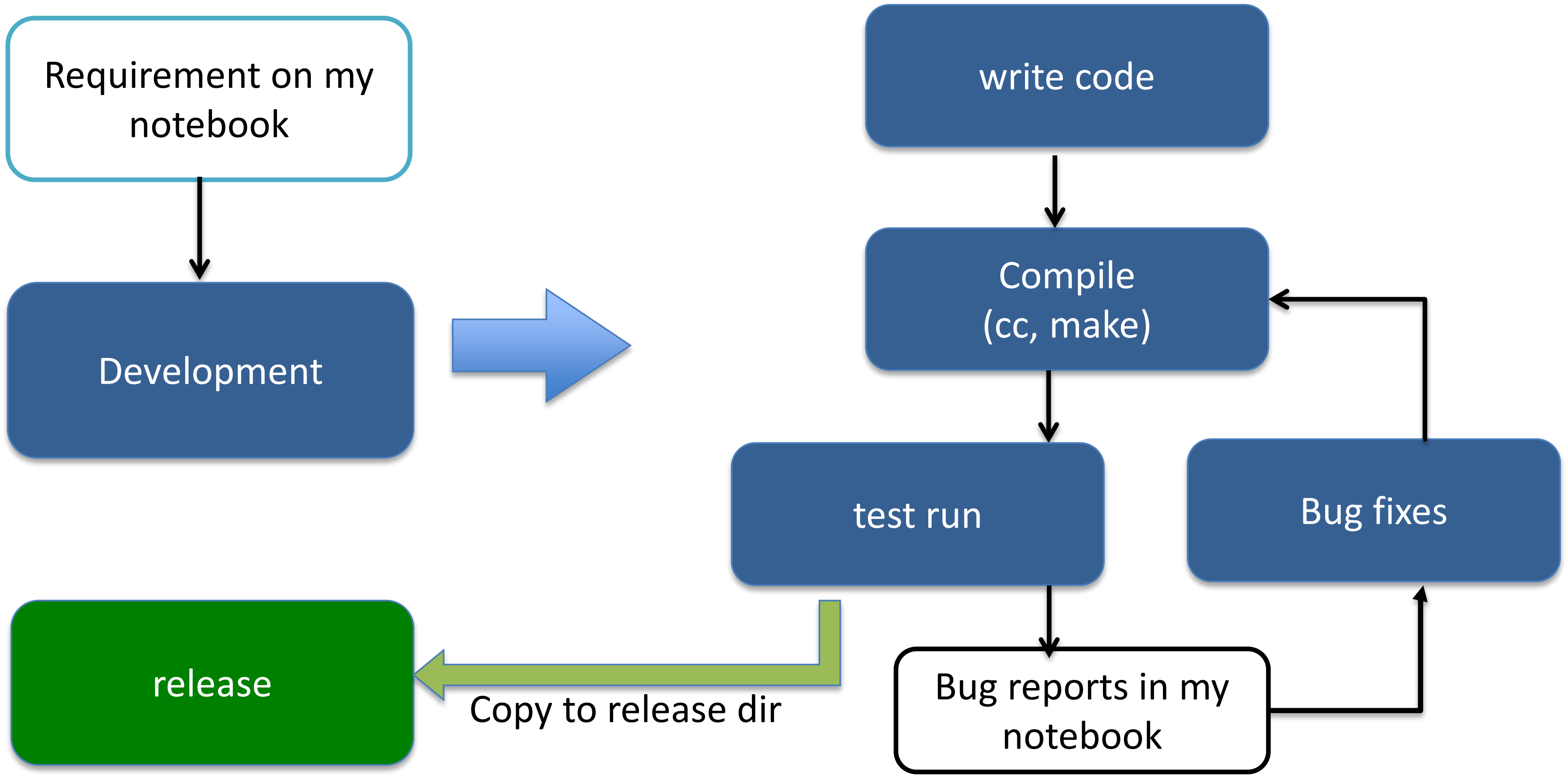
Issue Tracking Tools

Name	Initial Release	Stable/Latest Release	My Use Period
My notebook			1990 ~
GNATS	1992	2/28/2015 v4.2.0	1998 ~ 2017
TestTrack (Helix ALM)	1996	7/15/2019 v2019.3.0	2011 ~ 2014
Jira ✓	2002	9/9/2019 v8.4	2014 ~

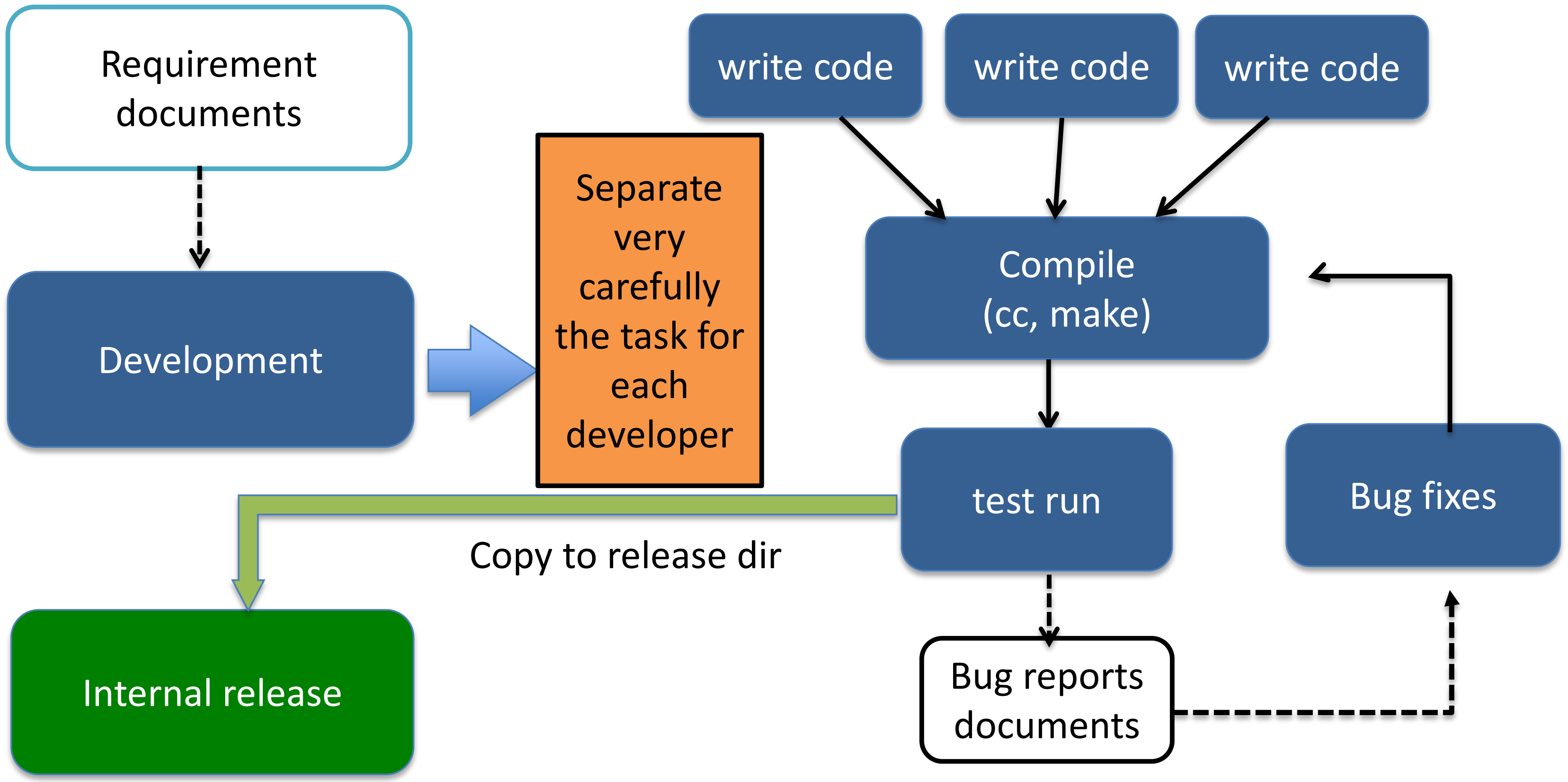
Version Control Tools

Name	Initial Release	Stable/Latest Release	My Use Period
Unix cp			1990 ~1998
CVS (concurrent version system)	11/19/1990	5/8/2008 v1.11.23	1998 ~ 2012
AccuRev	5/18/1999	01/2015 v6.2	2011 ~ 2014
Git ✓	4/7/2005	8/16/2019 v2.23.0	2013 ~
SVN (Apache Subversion)	10/20/2000	7/25/2019 v1.9.12, 1.10.6, 1.12.2	2019 ~

Development Process
(single developer, early 90s)



Development Process
(small team 3 people)

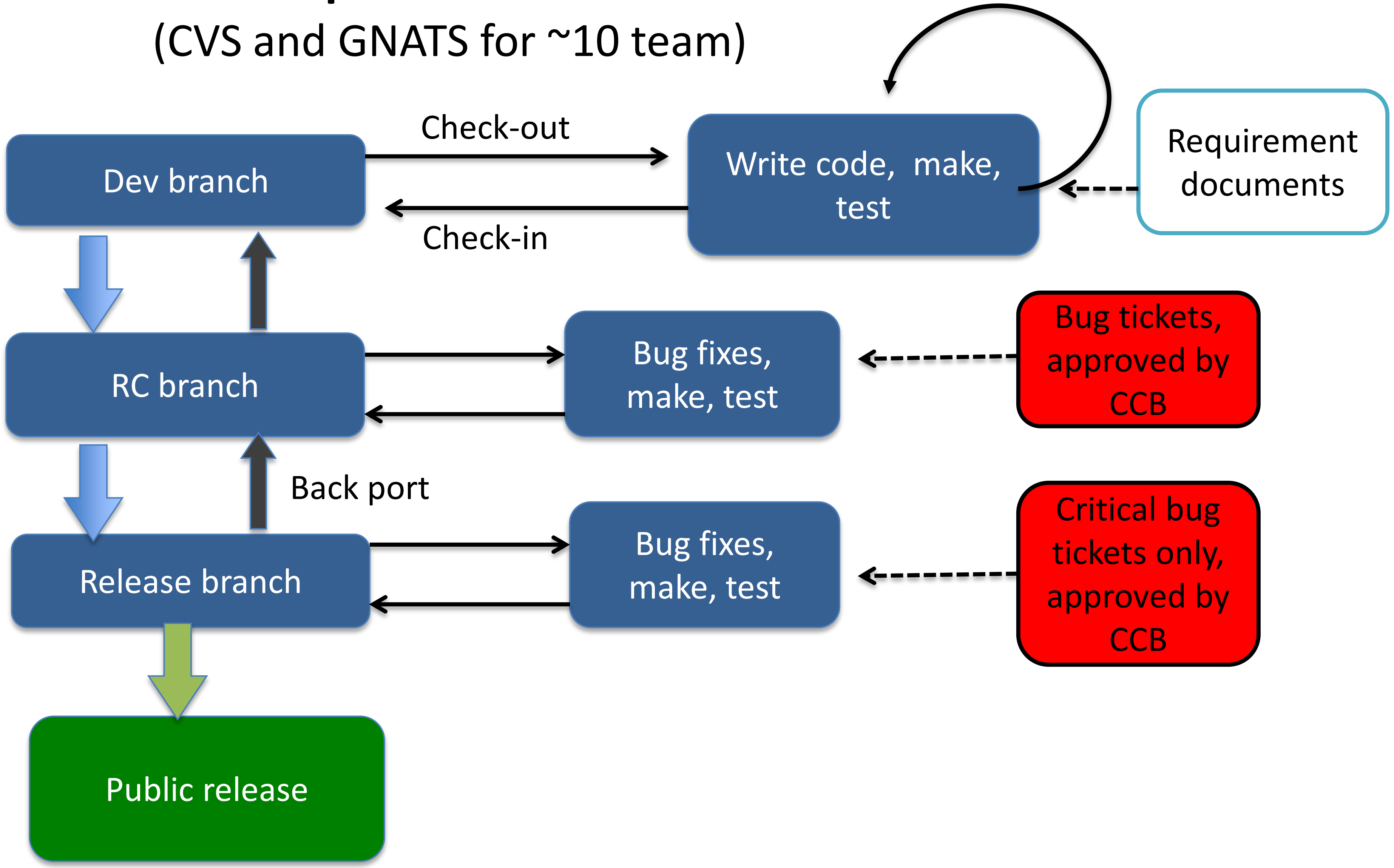


Development Process
(~10 people team)

- Communications among team members needed increases exponentially:
  - $n*(n-1)/2$ $n=3 \rightarrow 3$, $n=10 \rightarrow 45$
- Issue tracking and version control management were desperately needed
- Those two systems served us well for Spitzer uplink development
  - CVS
  - GANTS
- Decisions and criteria in configuring the systems
  - Simple work flow in CVS
    - No user or branches, to reduce the complexity and get everyone on board quickly
    - Use limited branches: dev, RC, release
    - Bug fixes on RC branch within a major release and back port to dev branch
  - Simple life cycle of a ticket
    - New, assign, development, built, tested, close
    - The feature/bug fix can be released only when the associated ticket is closed by testing team

Development Process

(CVS and GNATS for ~10 team)



Limitations with CVS

- No user or bug branches made it hard to isolate the issues
- Cherry-picking could be troublesome
- No adequate tools for code review

Development Process
(Git/GitHub for large/distributed teams)

- Started using Git in-house in 2013, and soon adopted GitHub
  - LSSST adopted GitHub
  - Easy to use UI, especially the code review feature
- Adopted new process for development cycle
  - Dev branch is the base branch for development
  - Feature/bug branch for every ticket
  - Pull request(PR) t when the branch is ready for code review
  - Each PR has to be approved by code reviewer(s) before merging
  - Each branch bee to rebase with Dev to resolve conflicts before merging

Lessons Learned

- Involve developers early in setting up the process
  - Pilot studies of couple of options
  - Team discussion of options and pros and cons
  - It may take longer to adopt the process, but well worth it once everyone is on board.
- Branch naming convention
  - We use project prefix, ticket number and few words of description, i.e FIREFLY-372_MocDisplay, IRSA-1805-details-abstract-tab
  - Make it easy to connect branch to project and ticket, help reviewers to identify the branch
- Tools are important
  - Jira scrum board makes sprint management easy
  - GitHub makes code review so much easier to implement

Code Review

- Benefits
  - Uncover issues and bugs early
  - Good learning process for junior developers
  - Promote good coding style
  - Improve code readability
  - Cross training among team members
- Challenges
  - Developers personal feelings
  - No easy tools before GitHub
    - Organizing a review takes quite some effort
    - Picking the code to review, assigning reviewers, dealing with the review results
  - Burden on senior developers

Integration and Test (I&T)

- Critical for project
  - Could wreck the whole schedule if not planned well in advance
- Continuous Integration (CI)
  - Automated daily build and deployment of Dev(RC) branch
  - Testers get involved during development
- Test as early as possible -- in ticket branch
  - Automated build and deployment for each pull request (Jenkins, Docker, Kubernetes)
  - Code review, unit test, manual test before merging ticket branch into dev branch

Golden Age of Development (Glorious Tools)

- Feature rich IEDs
  - IntelliJ, Eclipse, PyCharm, Atom, Sublime
- Debugger
  - All the wonderful debuggers in IDE and DevTools
- Version control and Issue tracking tools
  - GitHub, Jira
- Build tools
  - ant, maven, gradle

Thanks to all the people I have worked with on all the projects over the years!