Version Control

(Správa verzí)

http://d3s.mff.cuni.cz

Department of Distributed and Dependable Systems D3S

Pavel Parízek

parizek@d3s.mff.cuni.cz

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

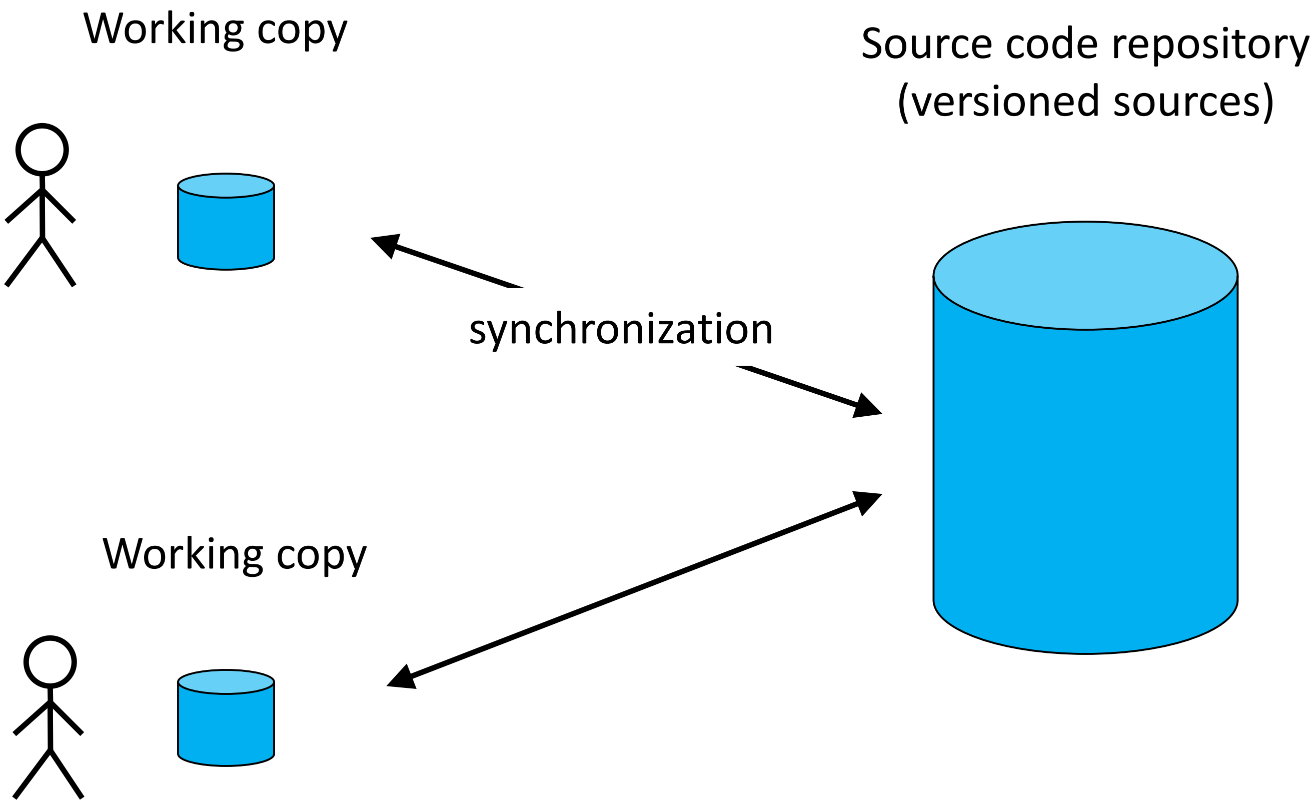
What is it good for ?

- Keeping history of system evolution
  - Tracking progress

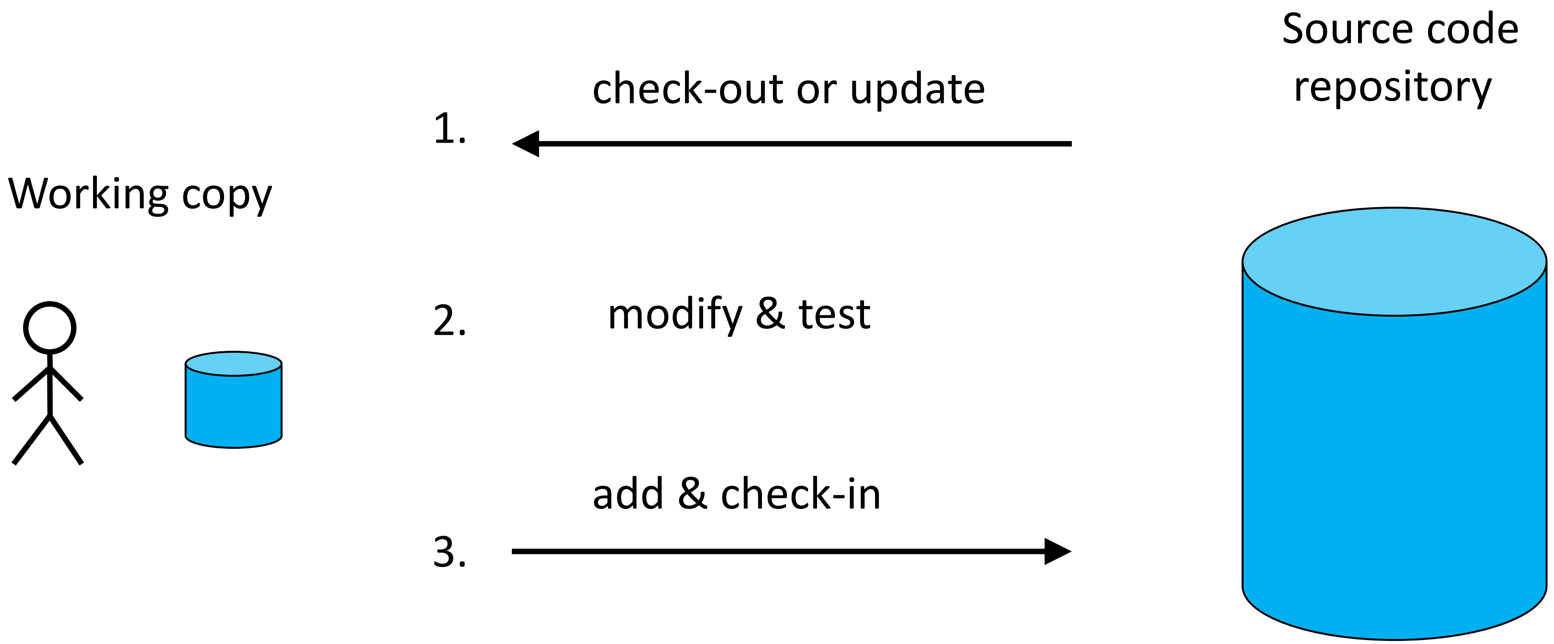
- Allowing concurrent work on the system
  - Teams of developers
  - Possible conflicts

- Easy reverting to a previous version
  - Safer experimentation

Typical architecture



Basic usage scenario



Source code repository

Working copy

1. check-out or update

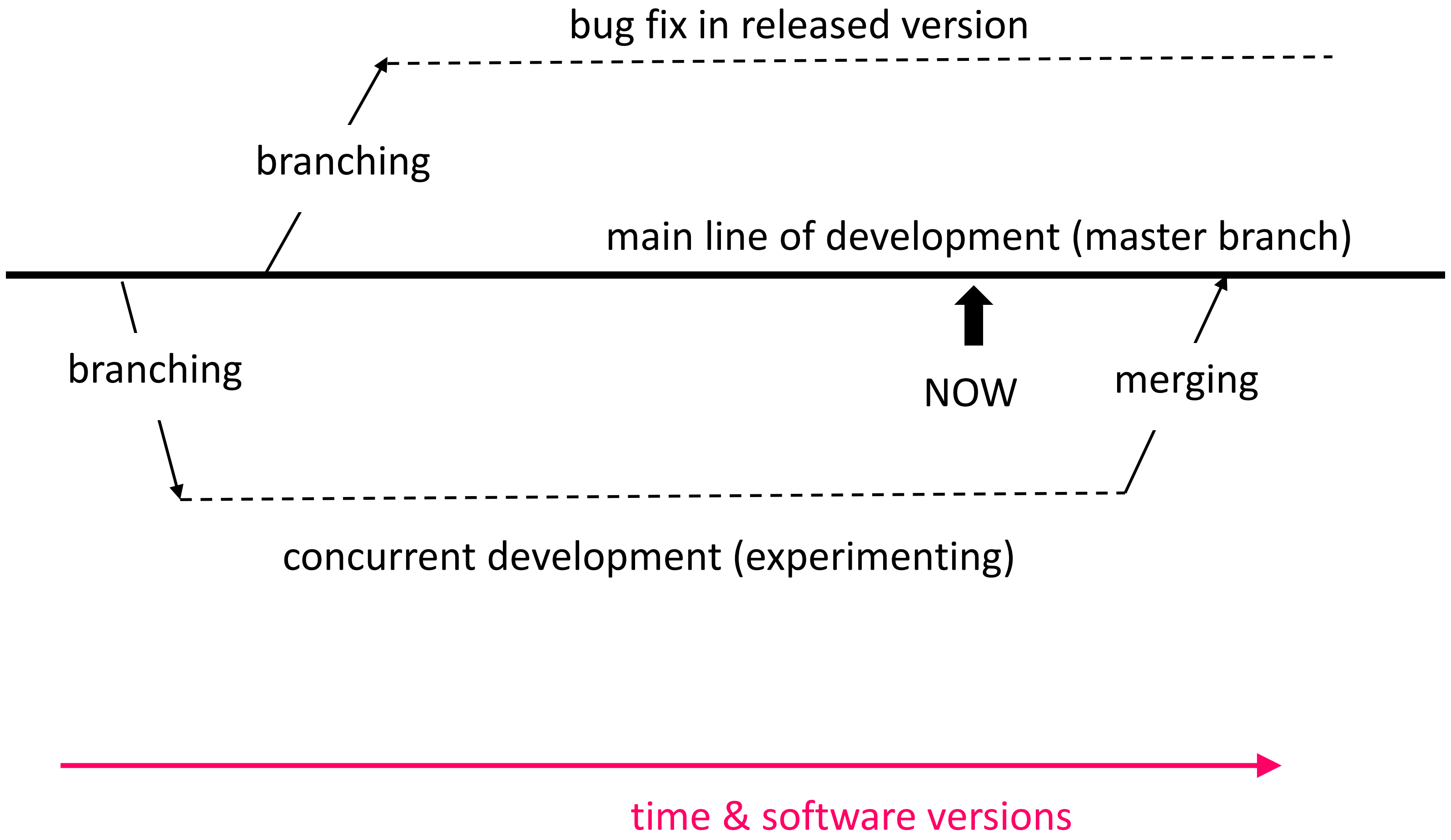
2. modify & test

3. add & check-in

Categories of versioning systems

- Centralized
  - CVS: Concurrent Versioning System
    - The “classic” system
  - SVN: Subversion
    - Currently still used by some open-source projects
- Distributed
  - **Git**, Mercurial, Bazaar

Branches and merging



bug fix in released version

branching

main line of development (master branch)

branching

NOW

merging

concurrent development (experimenting)

time & software versions

Conflicts

- Options
  - Postpone resolving
  - Choose version
  - External merge tool
  - and many others

- Conflict markers
  - <<<<<<< and >>>>>>> in source file
- Three variants of the source file created

Tree conflicts

- Typical cause
  - Renamed files and directories
  - Deleted files

- Solution
  - Make proper changes in the working copy
  - Use patches created with the `diff` command
  - Commit when everything is in a clean state

Tags

- Snapshot with a human-friendly name
- Logical copy of the whole source tree

Best practices: synchronizing developers

- Software developed in large teams
  - People may not be always able to coordinate efficiently
- Solution: Copy-Modify-Merge
  - Concurrent modification of source files
  - Resolving conflicts when they happen
- Alternative: Lock-Modify-Unlock
  - The old classic approach (“before internet”)
  - Does not scale well (exclusive access)
  - Not very robust (people forget to unlock)

Best practices: branches and merging

- Use branches for experimental features
- Create special branch for each feature
- Separate release and development branches
  - Propagating bugfixes from development to stable

- Merge often and synchronize with trunk
  - Lower chance of ugly conflicts occurring
  - Smaller conflicts are easier to resolve
  - Commit often ➔ others will have to merge

Department of Distributed and Dependable Systems

Best practices: further reading

- Patterns for Managing Source Code Branches
  - https://martinfowler.com/articles/branching-patterns.html