ICIS Software Development and Release Management

ICIS 2006 Workshop

T. Metz, IRRI

Software Release Management

- OSS philosophy: It will be released when it is ready! (When developers are satisfied.)
  - Can lead to high quality products
  - No predictable release date and feature set
- Proprietary philosophy: It will be released on schedule! (When sales/marketing managers want it.)
  - Early users can become beta testers
  - More predicable availability and feature set

When should a new version of the ICIS software be released?

- When enough changes have accumulated
  - Each release has enough changes to make it worthwhile.
  - Flexible for developers, but less predictable for users. What is more important?
- According to a release plan
  - E.g. every 6 months with roadmaps for feature development, time plan for release process (feature freeze, testing, debugging, documentation)

ICIS should be released according to a formal release plan

- Formal release management can scale up
  - More developers, more locations.
  - Database/module interactions.
  - Documentation & training materials.
- Users can plan for upgrading / training
- Aim: Distributed development & global user community
  - Is there an alternative to formal release management?

ICIS collaborative development platform

The ICIS System on CropForge

ICIS Communication Project

- Release (download) of packaged ICIS software for end-users. Only tested and documented versions will be released here.

- Release of ICIS full schema scripts as well as upgrade scripts.

- Tracking systems for bugs, feature requests, etc. for communication with beta-testers and end-users.

- Forums, mailing list, news for communication with the global ICIS community.

The ICIS System on CropForge

ICIS Development Projects

- Projects per programming language, with separate modules per projects. Each module represents a different application (e.g. SetGen, GMSSearch).

- Source code repository (CVS/SVN) with update permission for developers and read-only (anomymous) access for everyone.

- Support provided for global ICIS developer community.

- Lead developers are project administrators.

- Beta versions of applications will be released here.

ICIS Development Project (DLL)

The Role of ICISWiki

Existing

- Technical documentation development
- Training materials development
- Development documentation (GRIMS)
- Community documents

New?

- ICIS release plan
- Module roadmaps

Source code management with CVS

CVS client-server architecture

- separate server
- no shared file systems
- one server process per connection

central
repository

CVS server - CropForge

CVS client - TortoiseCVS
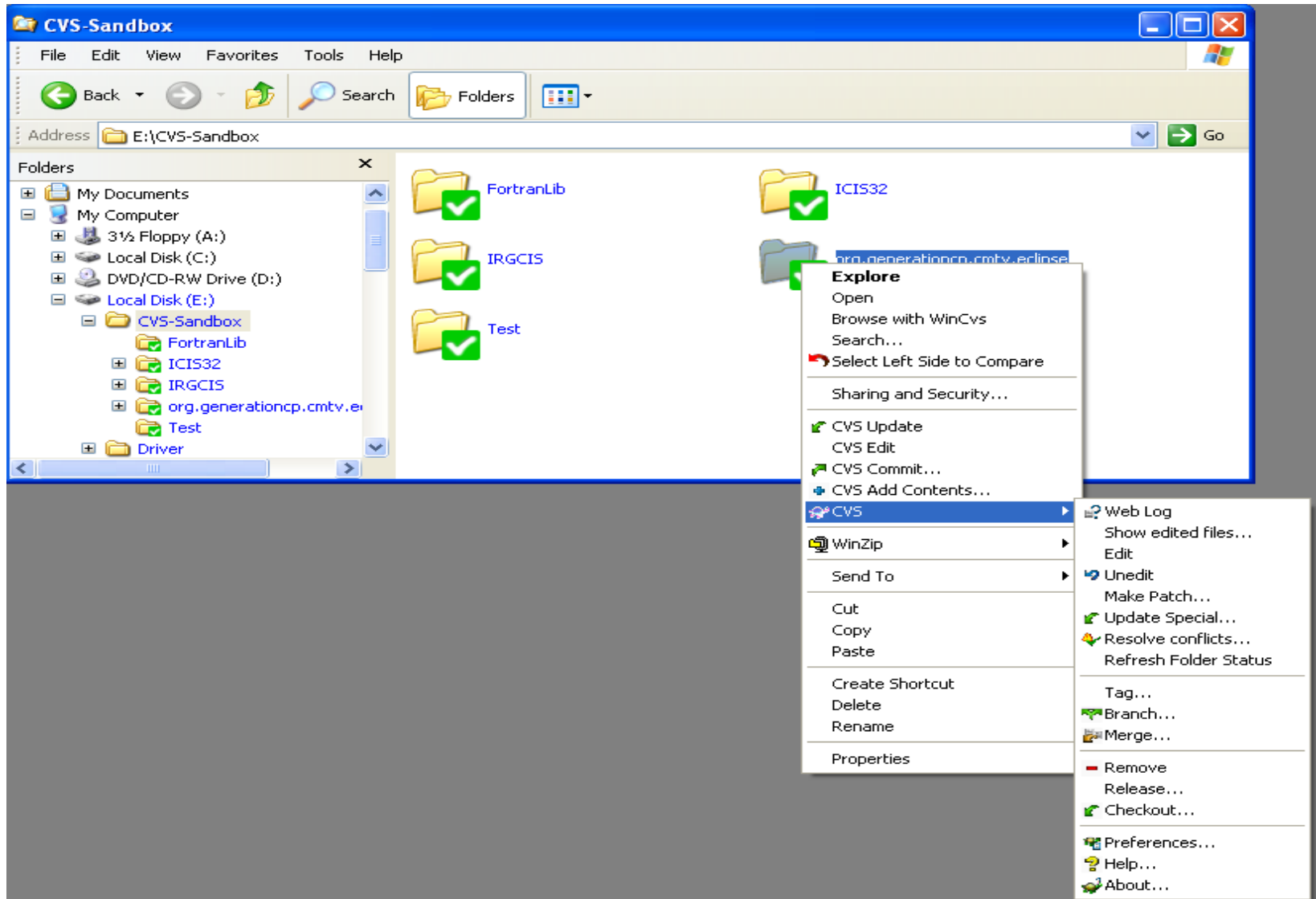
CVS usage model

- check out
  - makes private copy in the working directory
  - multiple copies and multiple versions in different locations are possible
- modify
- update
  - brings working copy up-to-date with repository
- commit
  - moves changes in private copy to repository

CVS principles

check out

CVS repository

developer sandbox

Edit
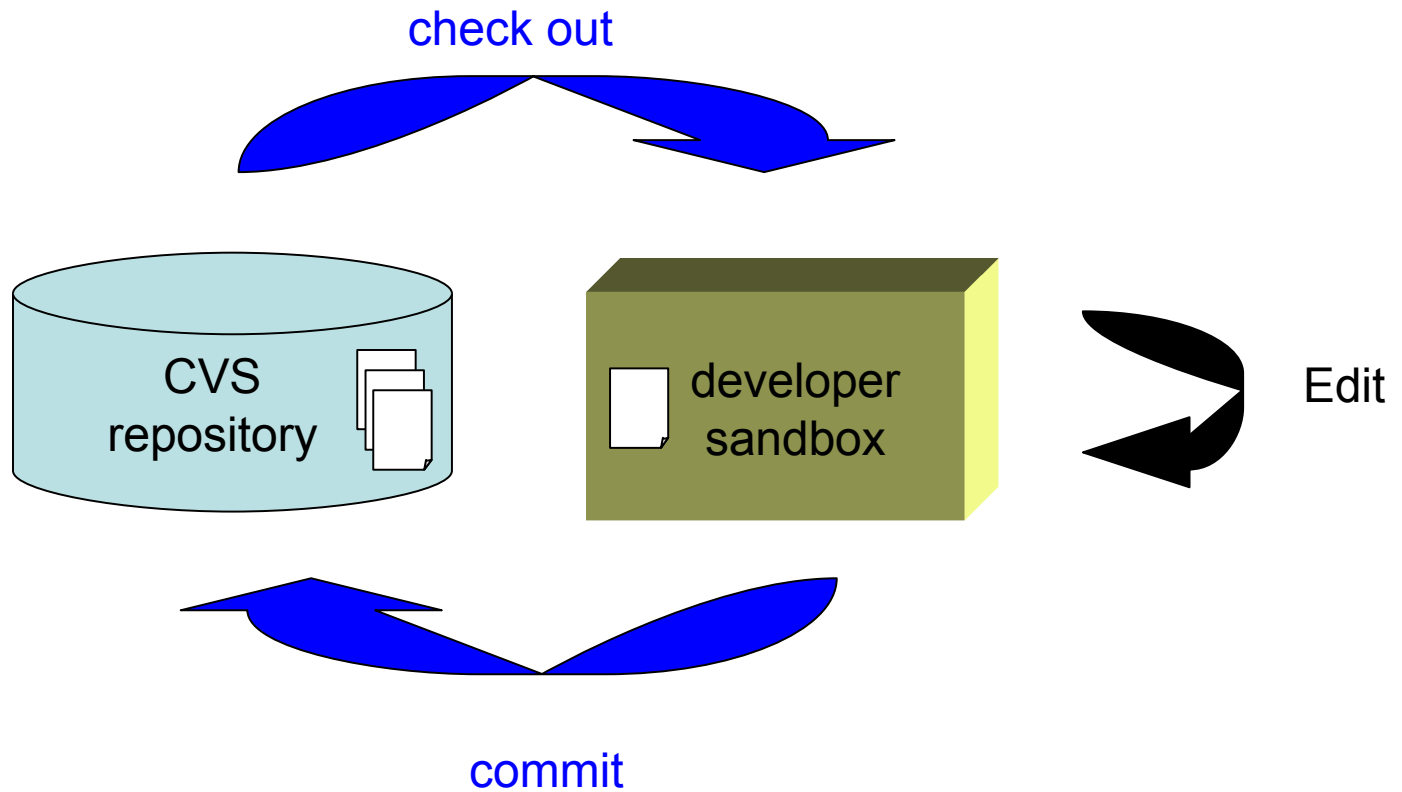
commit

CVS versioning

- version numbering is automatic
- number is incremented every time the file is changed
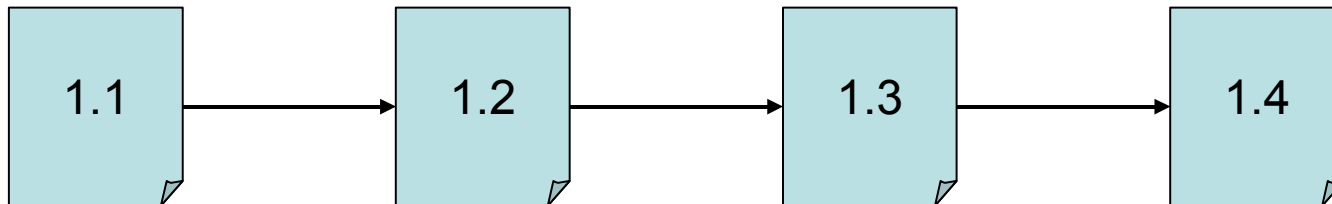
CVS principles
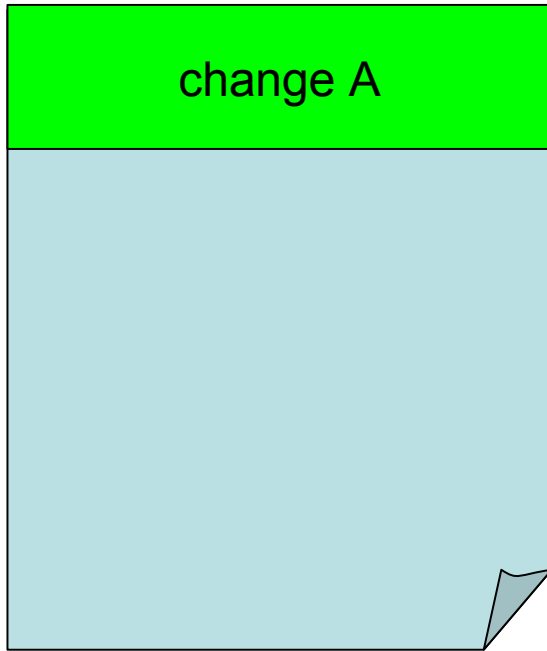
change A

- one developer
- single change
- commit
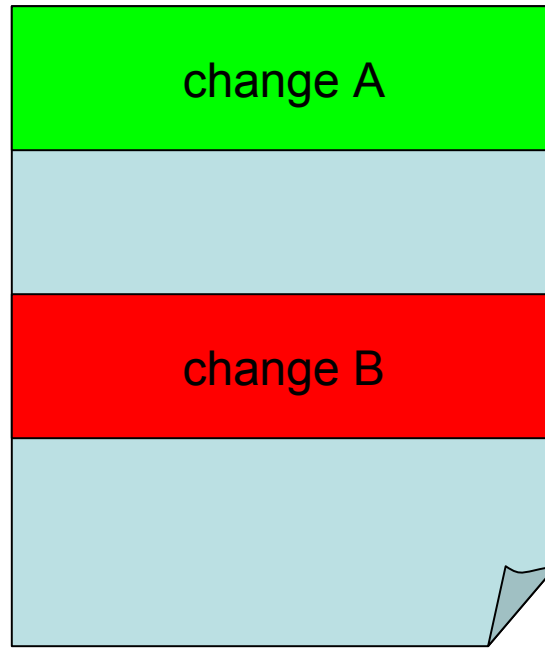
change A
change B

- two developers
- separate changes
- merge
- commit

change A
change A + B
change B
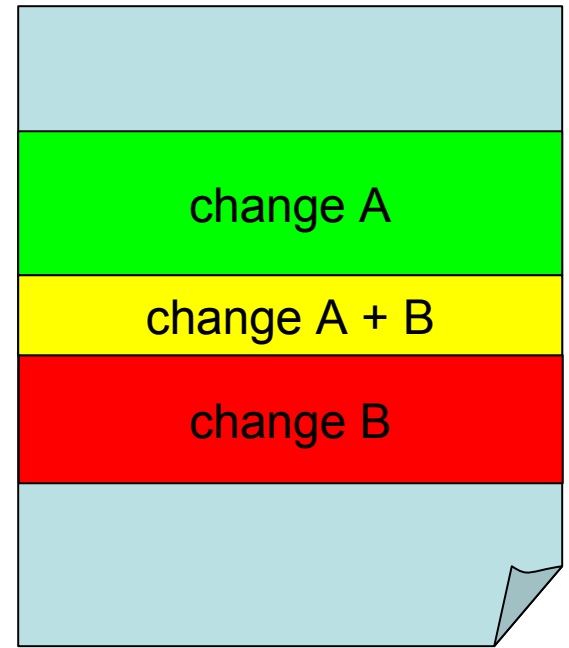
- two developers
- overlapping changes
- merge
- conflict resolution
- commit

CVS principles by example
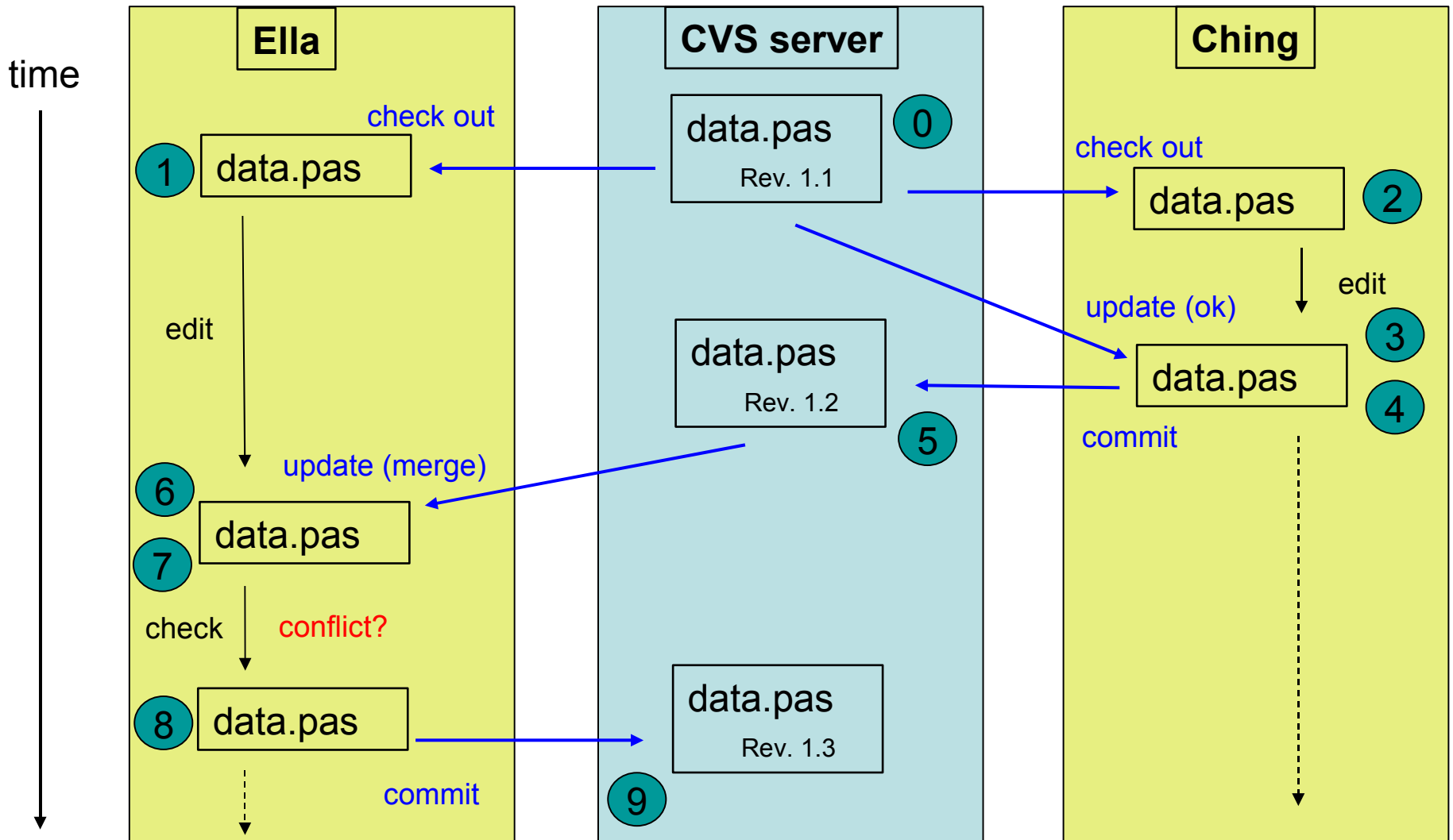
ICIS release management proposal

advanced CVS

ICIS Module Development Process

5.5 development branch

5.5rc1

5.5rc2

5.5 feature freeze

5.5 release branch

5.6rc1

5.5rc3

5.5 stabilization

5.5rc4

5.6 development branch

5.6rc2

Merging

5.5 — 5.5 release

5.5.1

Merging

5.6rc3

5.5 maintenance

5.5.2

5.6 feature freeze

Tagging
Branching
Merging

Code review
Testing
Bug fixing
Documentation

Bug fixing
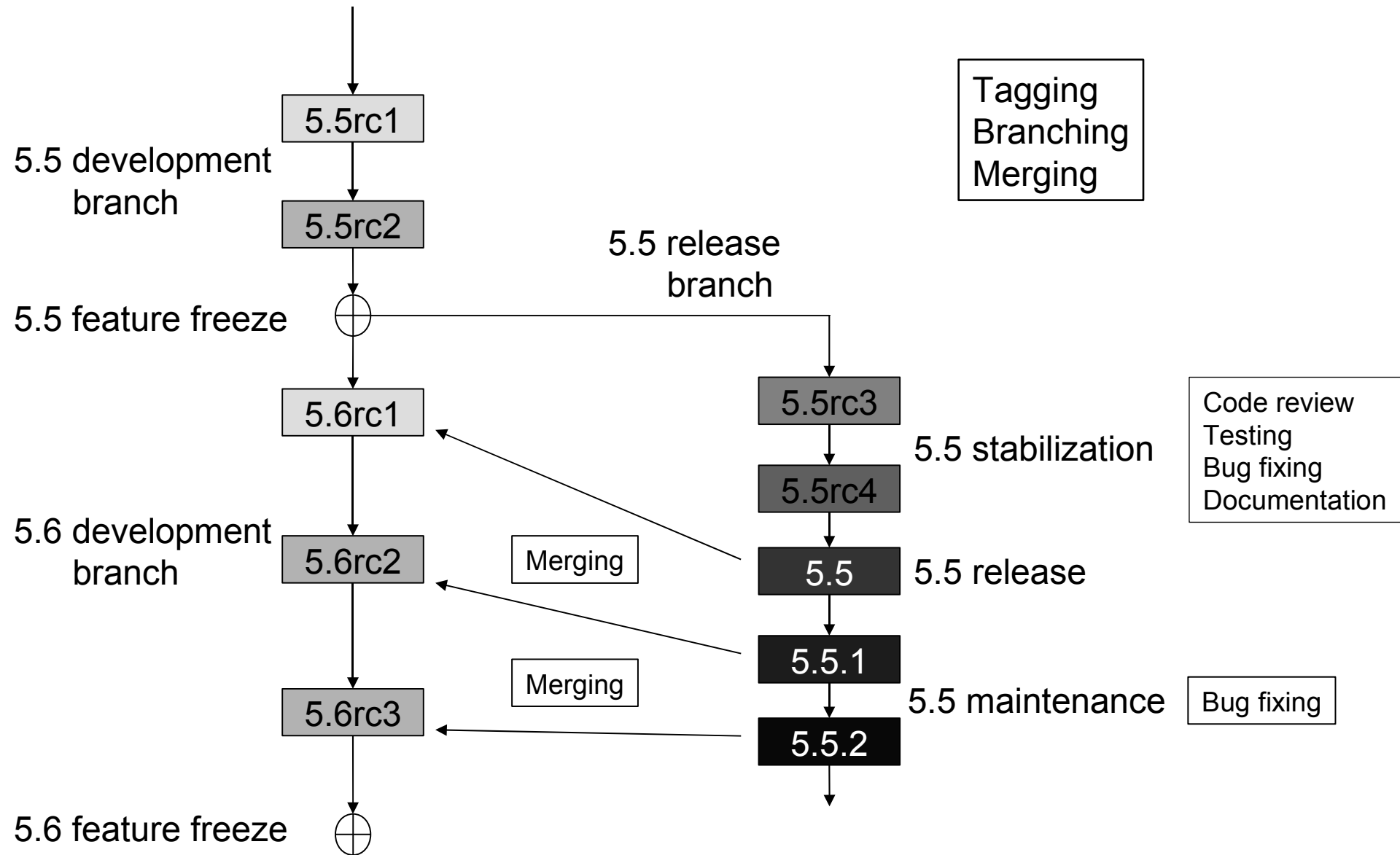
Tagging
- marking a moment in time -
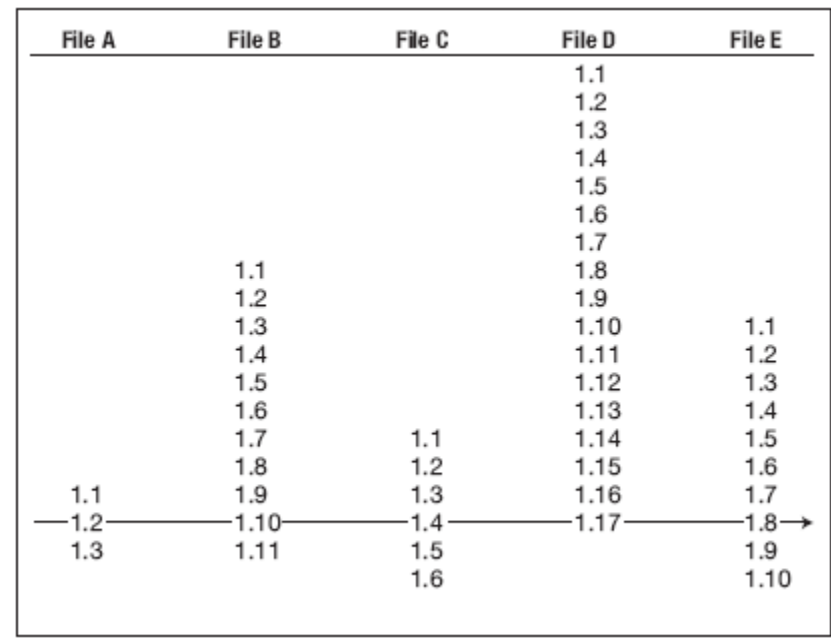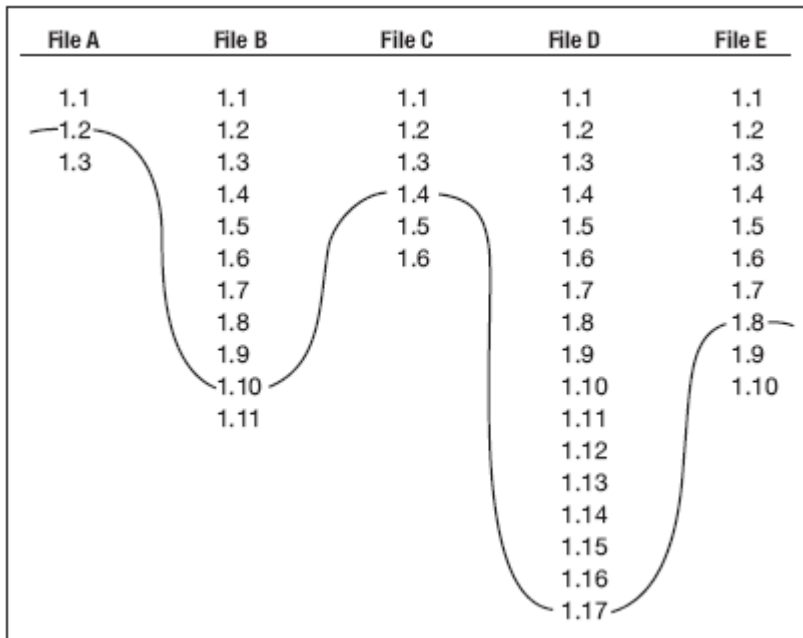
Source: Open Source Development with CVS, Karl Fogel, ISBN: 1-932111-81-6

A tag gives a label to the collection of revisions of one developer's working copy (the working copy should be completely up to date so the tag name is attached to the latest revisions in the repository).
E.g. SetGen-5.5rc2, SetGen-5.5, SetGen-5.5.1
The source code can be checked out by tag.

Tagging with TortoiseCVS
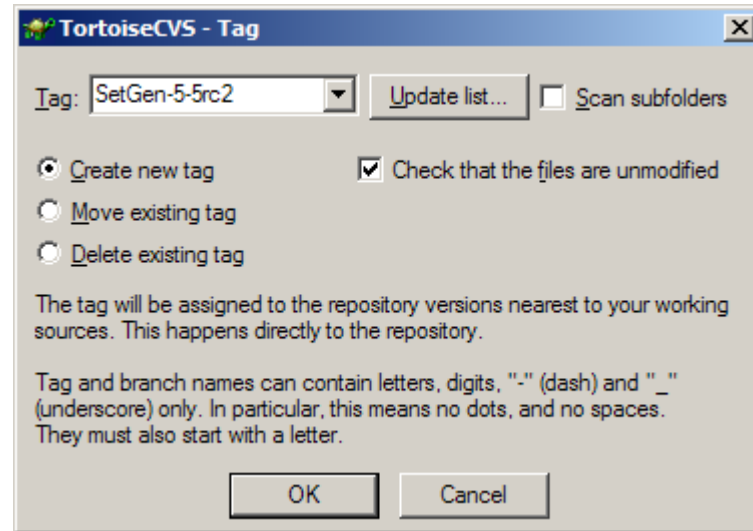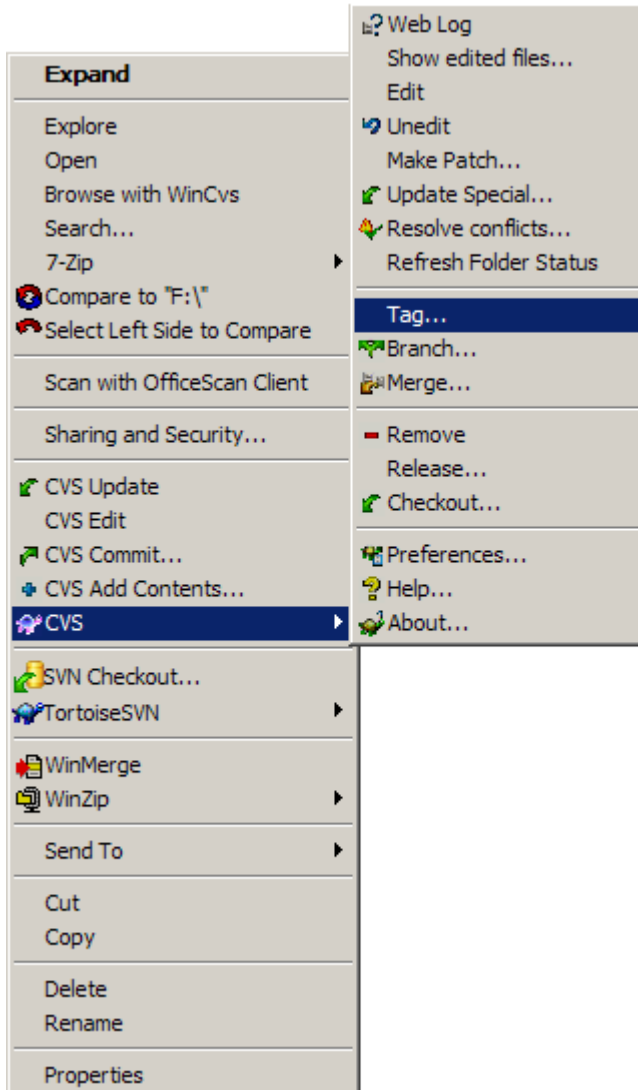
In the Tag Dialog you can enter a label
in the Tag field. A tag must start with a letter,
and may contain letters, digits, "-" (dash) and
"_" (underscore) only (no dots, no spaces).
When pressing the OK button the tag is
immediately applied to the repository, and
no commit is required.

Branching
- separate lines of development -
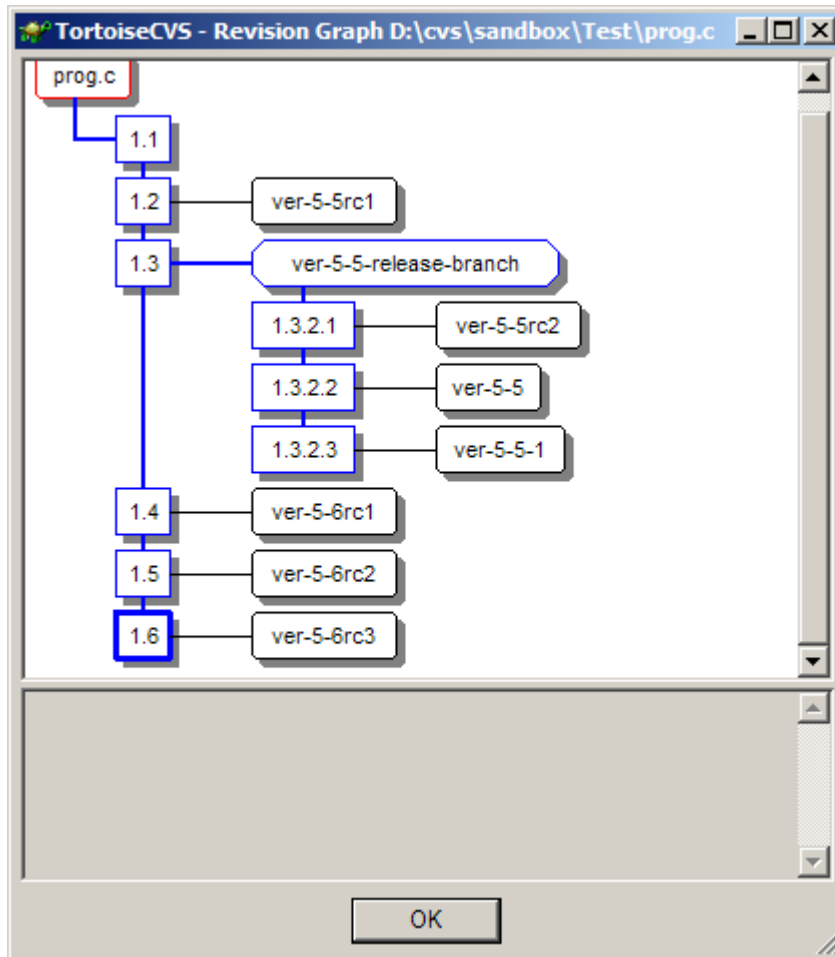
CVS can isolate changes onto a separate line of development (=branch).

The local sandbox can be switched between branches (update special) without maintaining multiple versions of the source code.

Changes (commit) made to one branch do not affect other branches.

Branching in TortoiseCVS
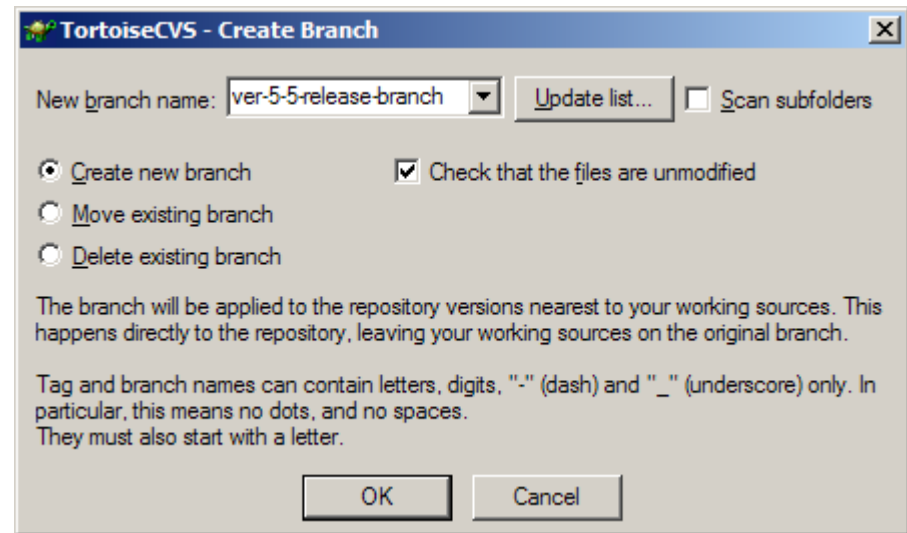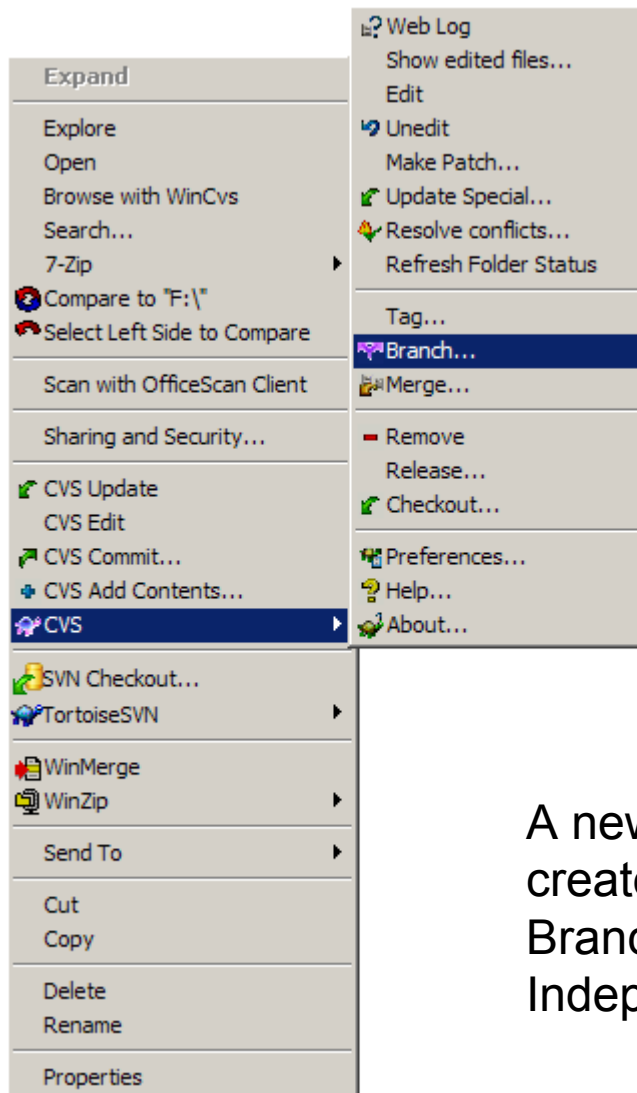
A new branch of the source code of a module can be created using the Branch command.
Branches of source code can be maintained Independently.

# Switching between tags/branches
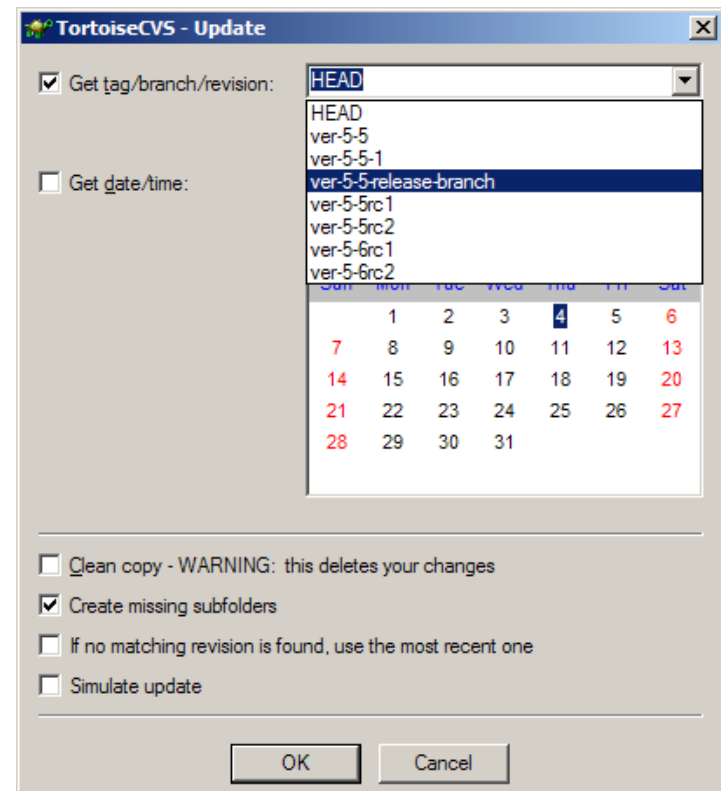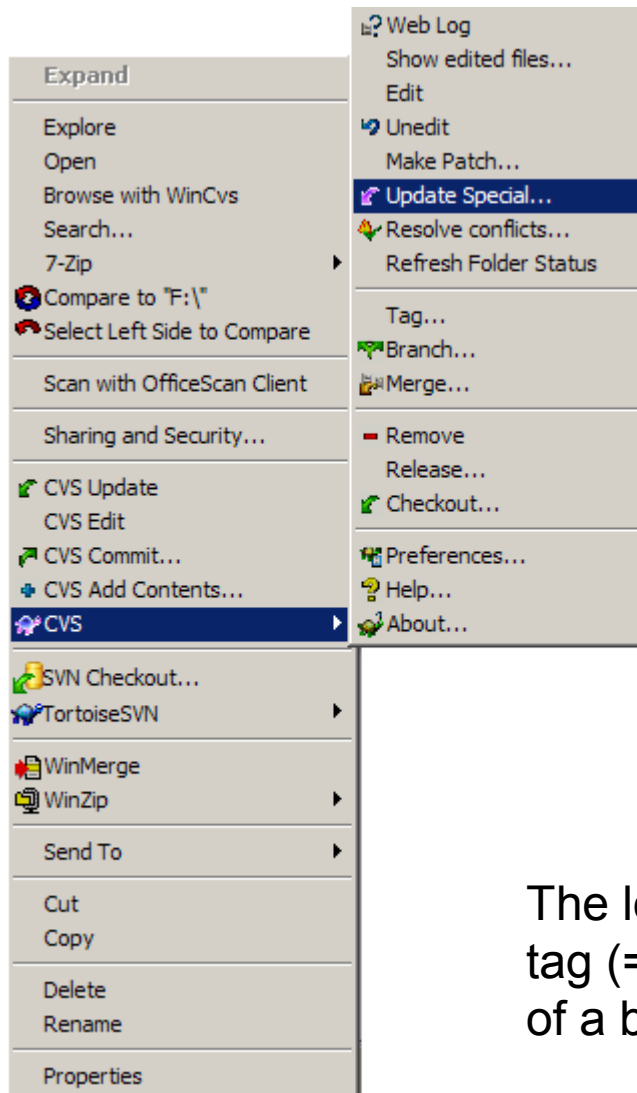
The local sandbox will be "converted" to a selected tag (=snapshot) of the code, or to the last version of a branch by using the Update Special command.

Merging
- migrating changes -
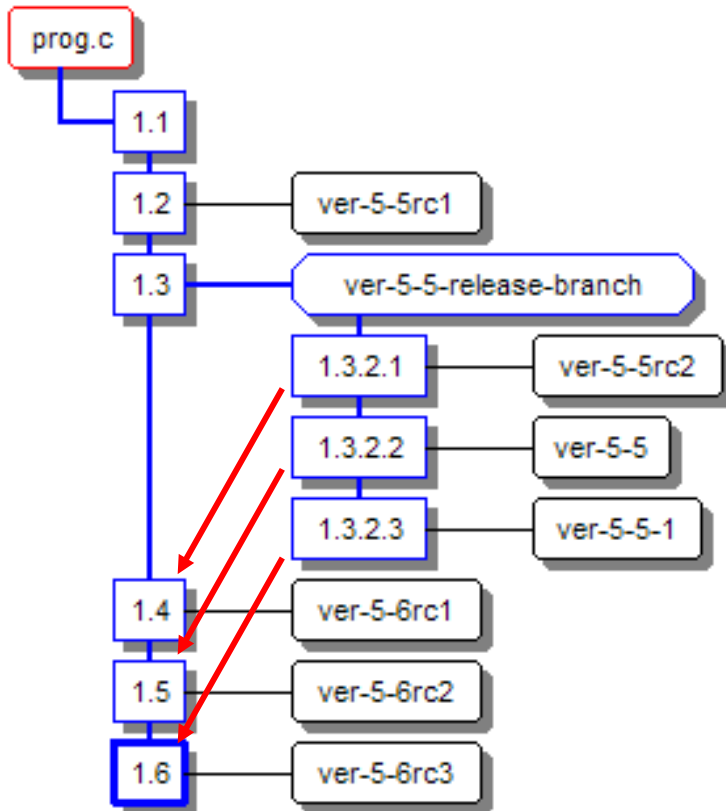
Changes made in one branch (e.g. bug fixes) may need to be applied to another branch.

Merging between branches will transfer such changes, e.g. from a tag (snapshot) in the release branch to the current version of the development branch.

Merging can be done for files, directories, or entire branches.

Merging in TortoiseCVS
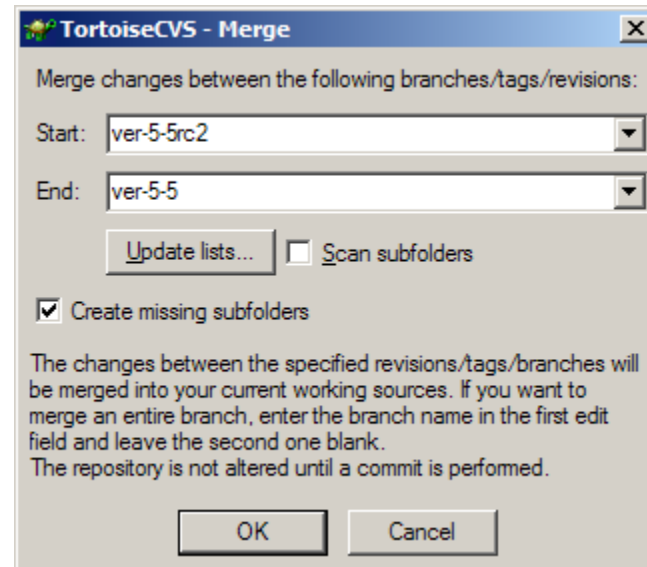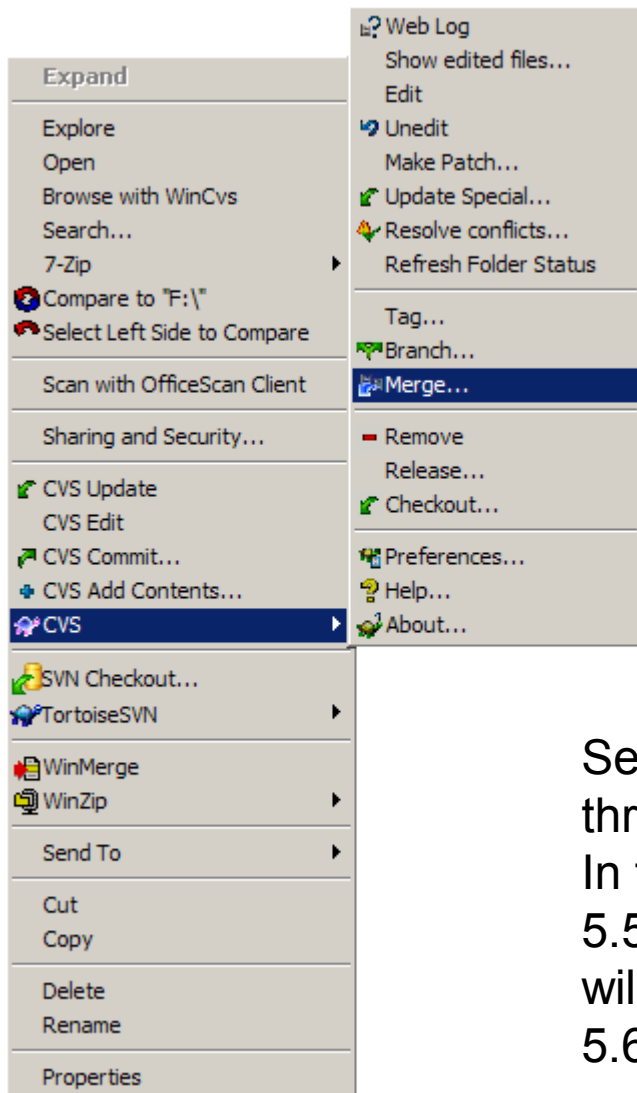
Set the current branch (e.g. HEAD) and then go through the Merge dialog.
In the above selection, the changes made to the 5.5 release branch between ver-5-5rc2 and ver-5-5 will be merged into the current version of the 5.6 development (HEAD) branch.

Release Manager

- Announce events (feature freeze, release).
- Check consistency and compatibility of planned feature set for next release.
- Check consistency and compatibility of implemented feature set at feature freeze.
- Package the release.

Conclusions

- The platforms for collaborative ICIS development, release, and support are in place, but not fully utilized.

- We need more formal plans, procedures, and standards.

- We need additional skills and experience.

- We need to develop more good habits.

References

- Producing Open Source Software, Karl Fogel, 2006. Creative Commons License

  http://www.producingoss.com/producingoss.pdf

- Open Source Development with CVS, Karl Fogel, 2003. General Public License

  http://cvsbook.red-bean.com/OSDevWithCVS_3E.pdf

- Version Management with CVS, Per Cederqvist, 2005. Free Software Foundation Open Content License

  http://ftp.gnu.org/non-gnu/cvs/source/stable/1.11.21/cederqvist-1.11.21.pdf

- TortoiseCVS – software, online help, admin guide General Public License

  http://www.tortoisecvs.org

  http://www.cvsnt.org/CVS_Administration_Guide.pdf