Collaborative Modeling with Version Control

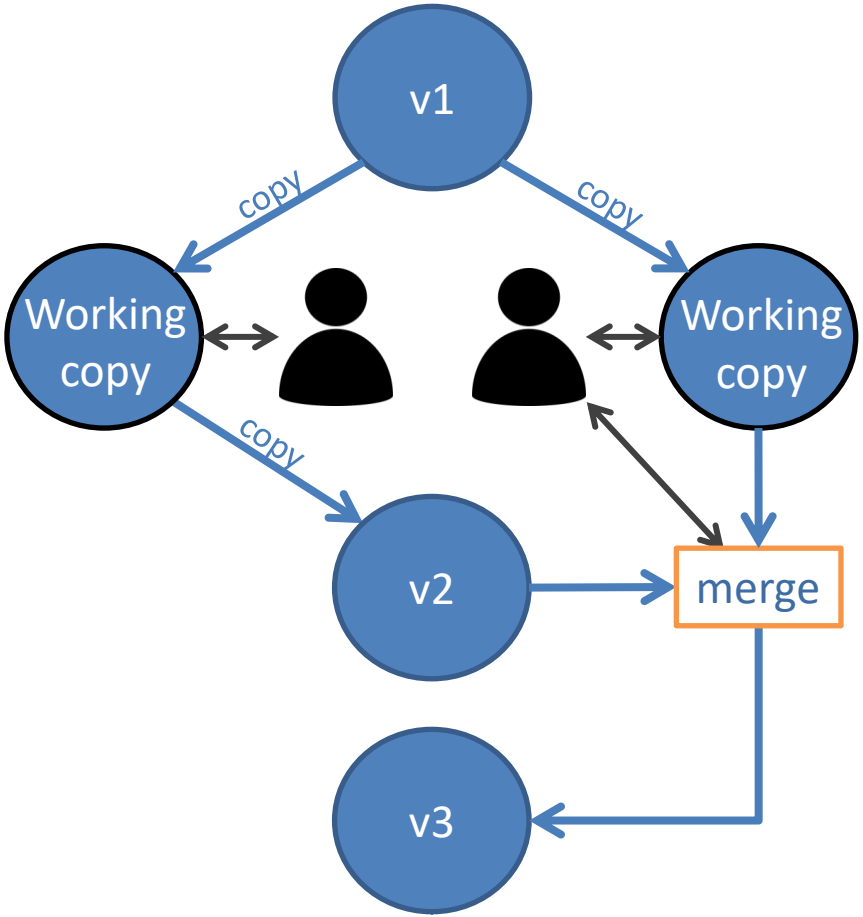
BigMDE, 21 July 2017

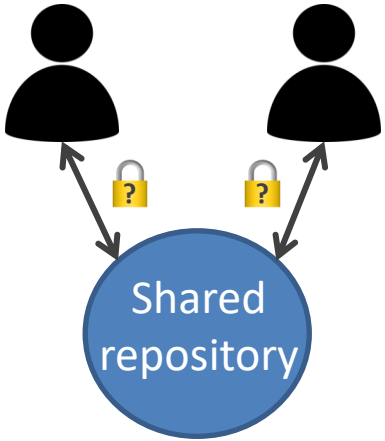
Steven Kelly, MetaCase

Clone *vs.* Share

merge after the fact

continuous integration





versions vs. multi-user

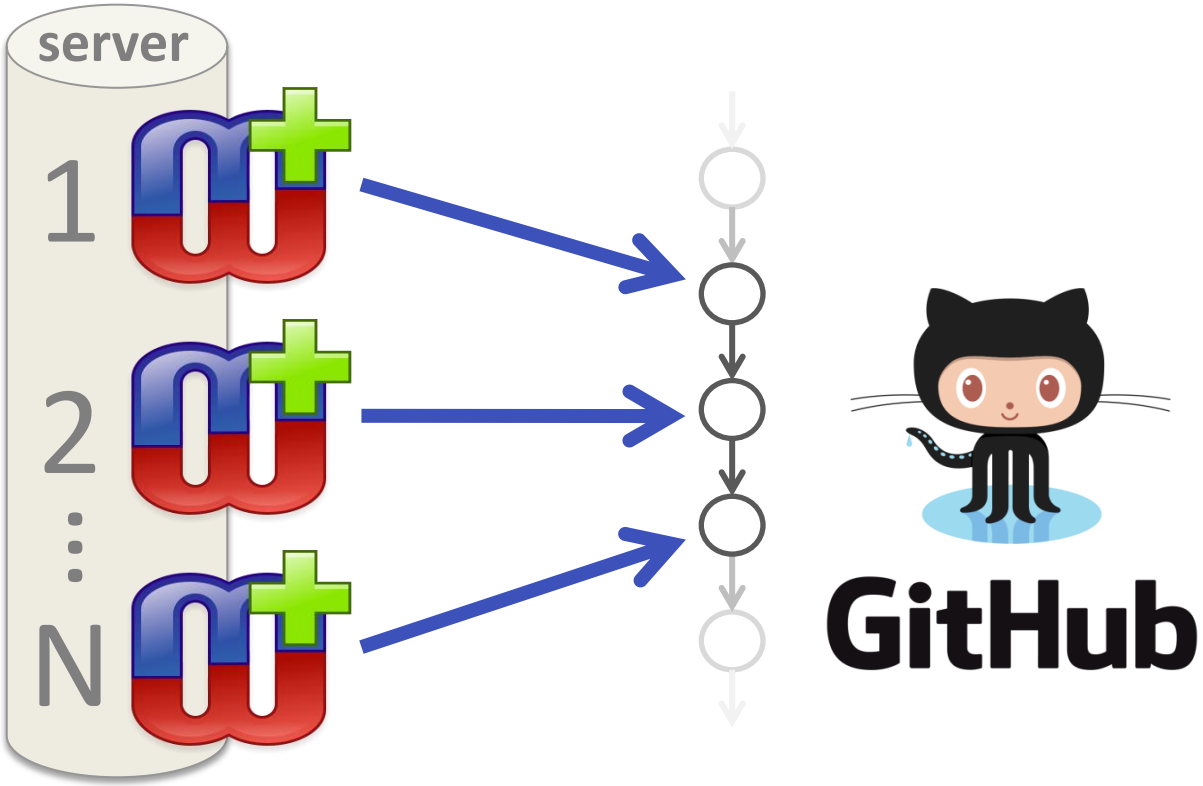
flat files vs. database

ASCII vs. objects

text vs. graphics

The challenge:
mix oil & water

versions ↔ multi-user
flat files ↔ database
ASCII ↔ objects
text ↔ graphics



MetaEdit+

Four purposes for versioning

- Understanding what's been done
  - Diff, version comments
- Archive and backup
- Branching
  - Parallel variants
  - Releases
- Collaboration

New tools

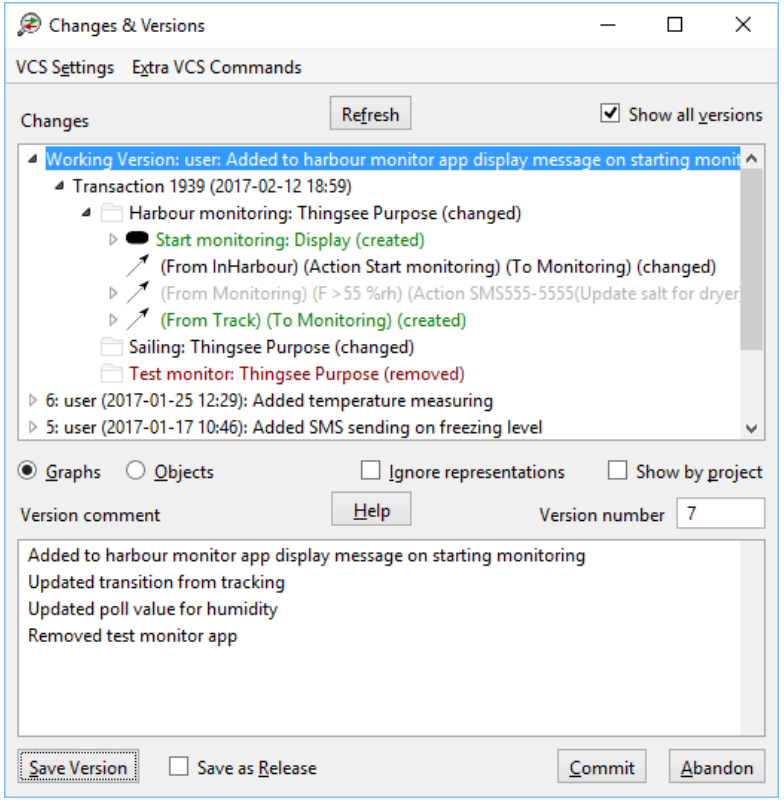
VCS integration

DSML

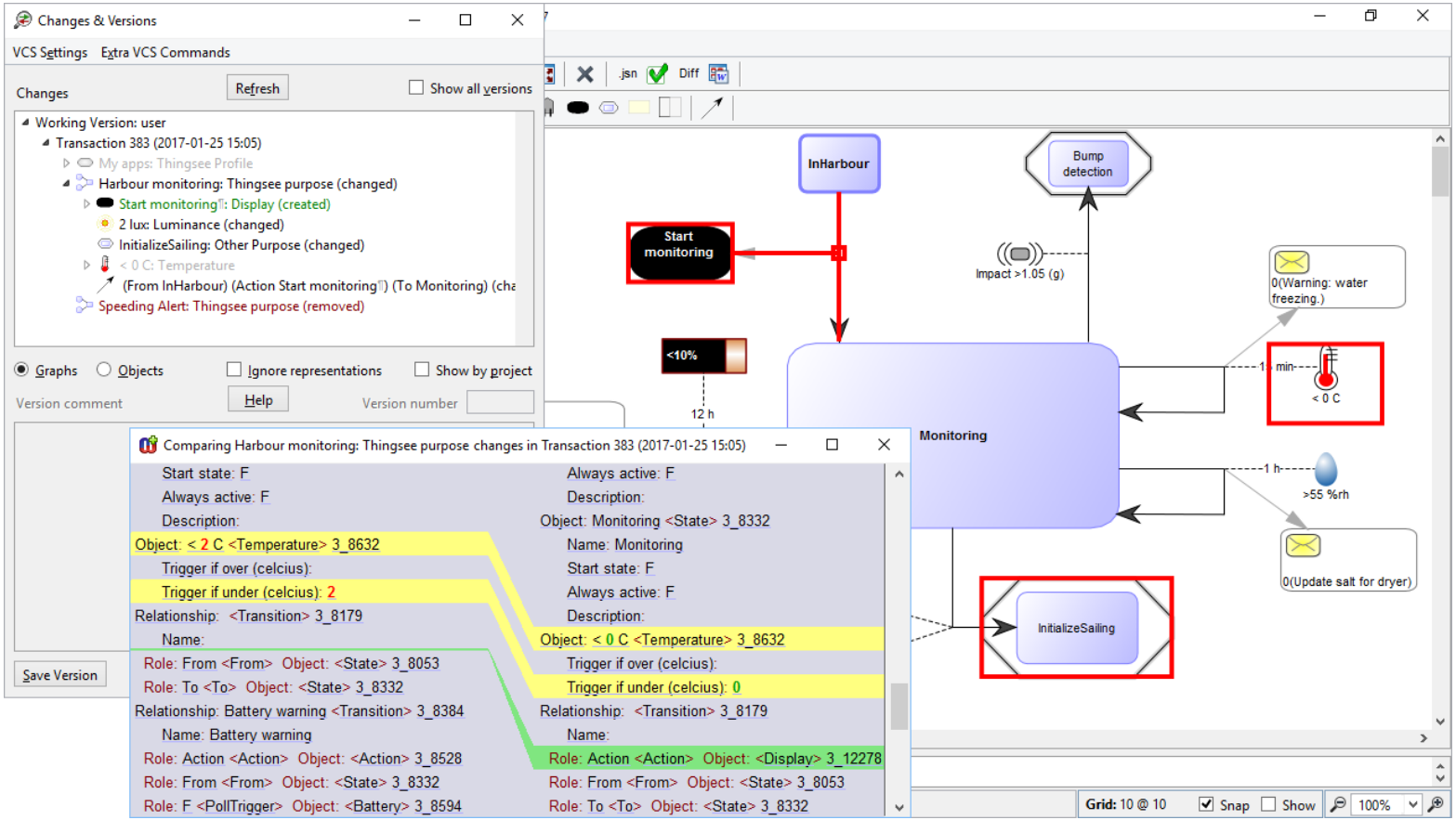
Multi-user

Changes and Versions tools

- Automatic trace of model changes
- See changes graphically directly in your models
- View changes as a tree using your language's structure and symbols, not XML
- Compare changes as a textual diff with live links to models
- Filters (your changes/all changes, data/representation)

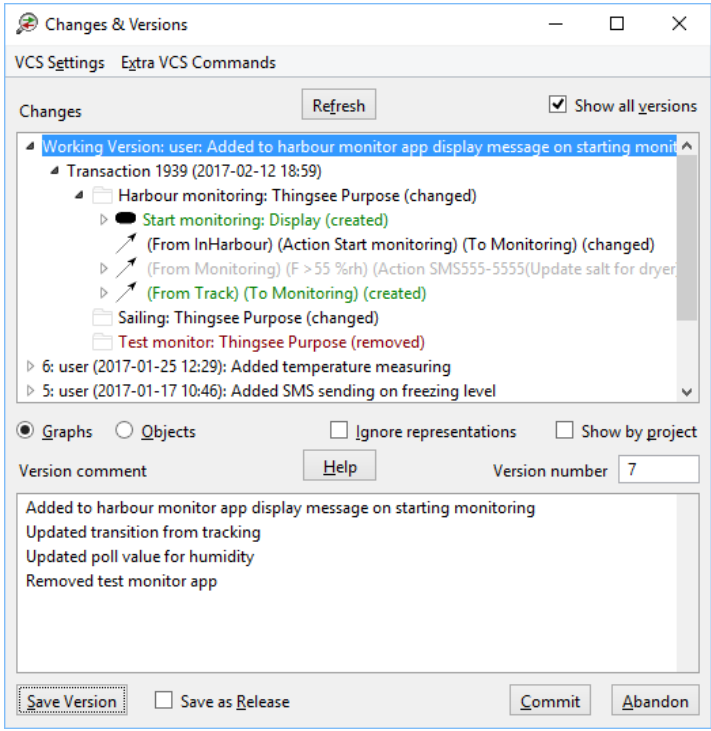


Three ways to view changes

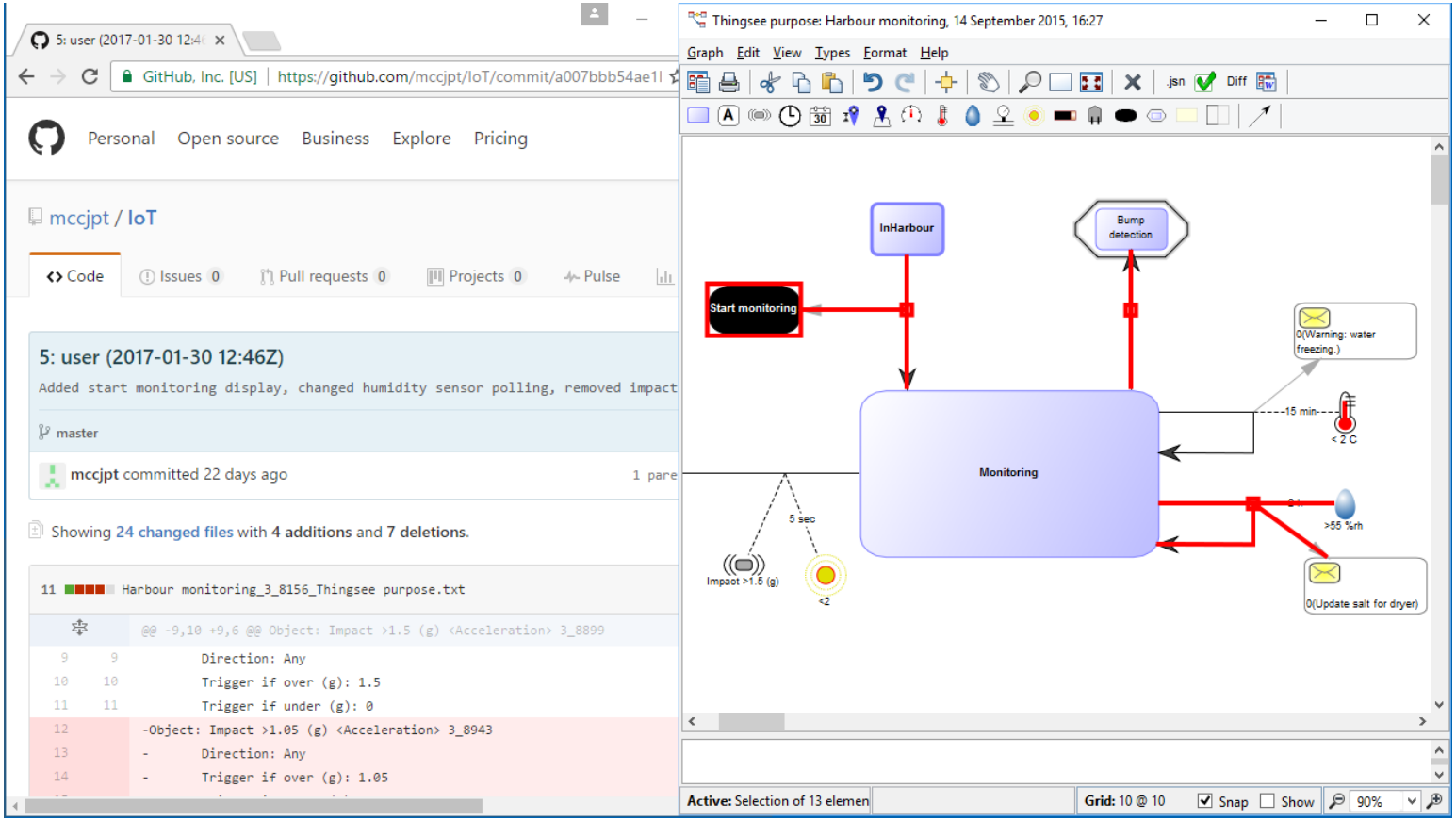


Versioning with external VCSs

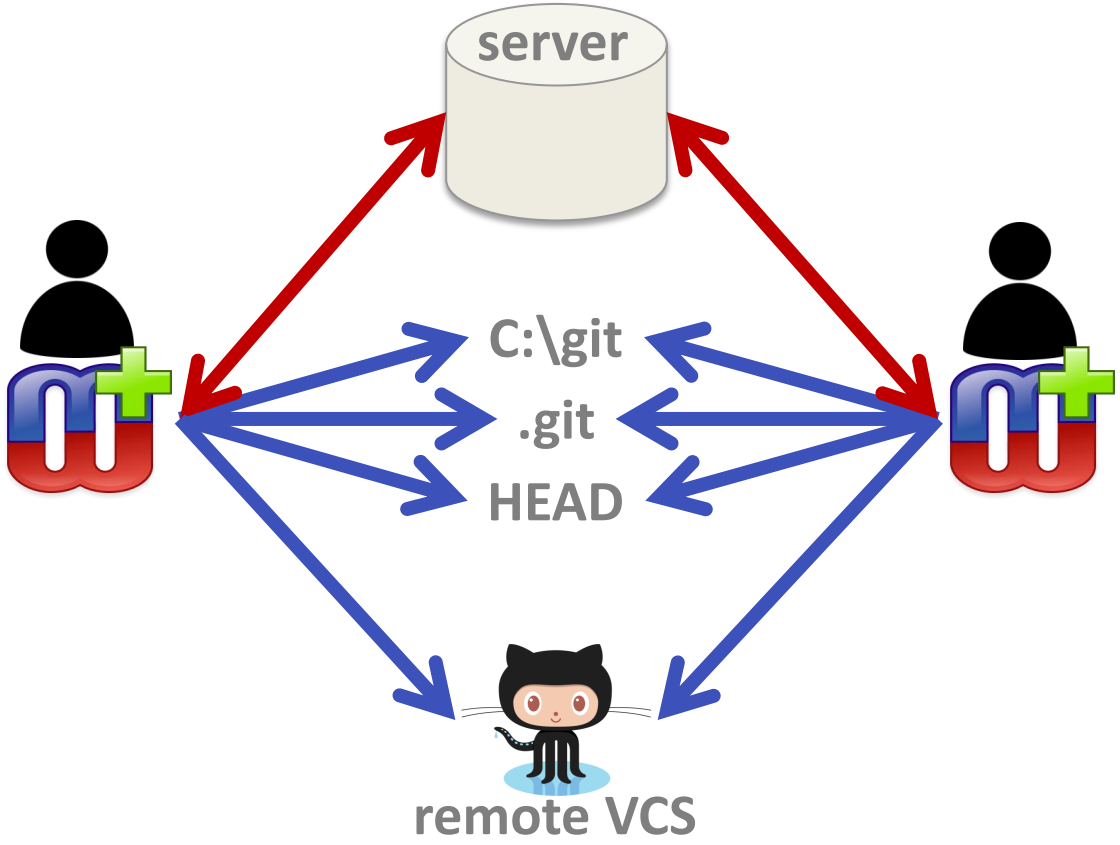
- Simple setup and use
- Automated background execution of versioning commands
- Maintain same model in several places, syncing with GitHub etc.
- Version and diff your language and generators along with models
- Git, SVN predefined
  - extend with your own



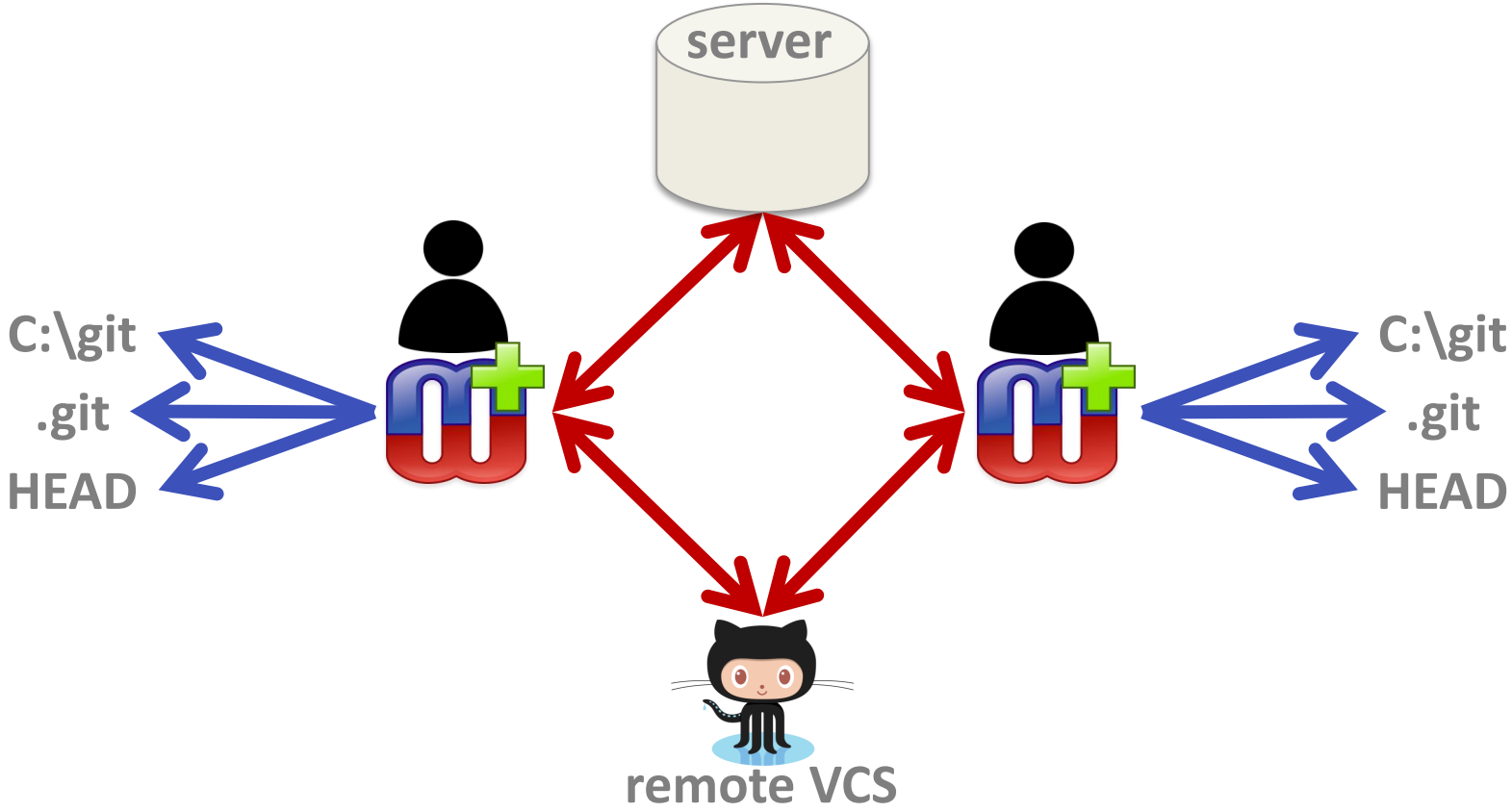
Version Control Integration (e.g. Git)



Shared repo, shared working dir?



Shared repo, working dir per user



Persuading VCSs to leave well alone

- MetaEdit+ has already integrated all users' work
- However, local working VCS info may be behind remote
  - Another user may have versioned after we last versioned
  - VCS will try to merge, reverting files to previous user's ☹
- Need to get VCS up to date, without changing files
  - Sync with remote
  - Update local to remote HEAD
  - Accept exactly these files as next version
- Order of these and actually writing files differs per VCS:
  - Git: write files, `remote update`, `reset --soft`, `add -a`, `commit`, `push`
  - SVN: `svn update`, write files, `TortoiseSVN commit`

Conclusion

- Extended a multi-user modeling tool with versioning
- New native tools for diff & working with versions
  - Three different integrated UIs: tree, graphical, (hyper)text
- Built integration with external VCSs
  - Easy, consistent UI across all VCSs, hides complexity of Git
  - Order of magnitude faster & simpler than best in Eclipse
  - User-extendable to new VCSs
- Works with multi-user repository: best of both worlds
  - No merges, no lock-outs
- Further work: for rare offline cases, add merging import

Thank you!