eawag
aquatic research

Best practice in scientific computing

Version control – `git`

Philipp Meier

November 25, 2014

Eawag: Swiss Federal Institute of Aquatic Science and Technology

Today's workshop

1. Introduction to version control
2. Discussion about future workshops
   - Topics
   - Infrastructure
   - ...
3. Hands-on session: Installing and using `git`

eawag aquatic research

Best practice in scientific computing

1. Write programs for people, not computers.
2. Let the computer do the work.
3. Make incremental changes.
4. Don't repeat yourself.
5. Plan for mistakes.
6. Optimize software only after it works correctly.
7. Document design and purpose, not mechanics.
8. Collaborate.

eawag aquatic research

Best practice in scientific computing

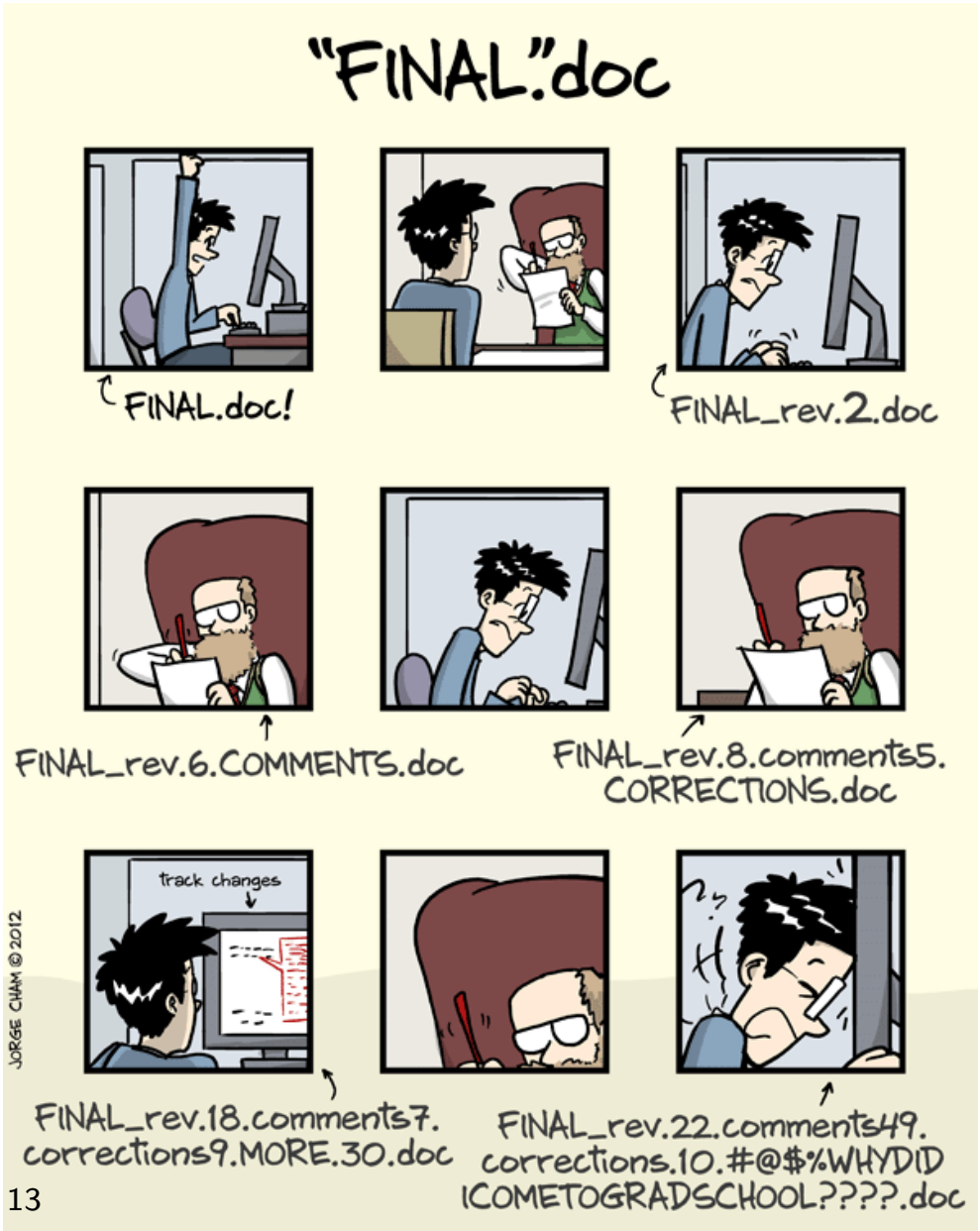
1. Write programs for people, not computers.
2. Let the computer do the work.
3. Make incremental changes.
4. Don't repeat yourself.
5. Plan for mistakes.
6. Optimize software only after it works correctly.
7. Document design and purpose, not mechanics.
8. Collaborate.

Best practice in scientific computing

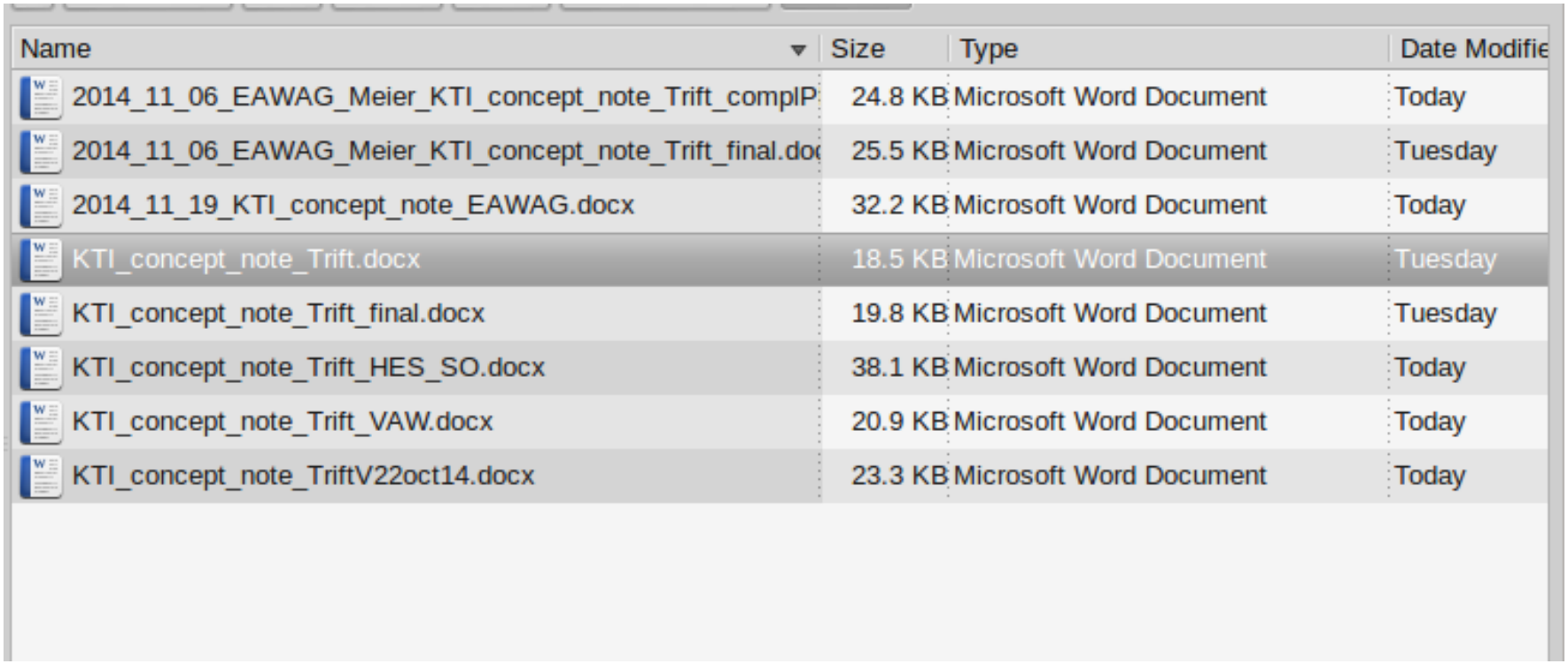
Where can version control help?

1. Write programs for people, not computers.
2. Let the computer do the work.
3. Make incremental changes.
4. Don't repeat yourself.
5. Plan for mistakes.
6. Optimize software only after it works correctly.
7. Document design and purpose, not mechanics.
8. Collaborate.

Version control: what is it?



Version control: what is it?



Name	Size	Type	Date Modifie
2014_11_06_EAWAG_Meier_KTI_concept_note_Trift_complP	24.8 KB	Microsoft Word Document	Today
2014_11_06_EAWAG_Meier_KTI_concept_note_Trift_final.doc	25.5 KB	Microsoft Word Document	Tuesday
2014_11_19_KTI_concept_note_EAWAG.docx	32.2 KB	Microsoft Word Document	Today
KTI_concept_note_Trift.docx	18.5 KB	Microsoft Word Document	Tuesday
KTI_concept_note_Trift_final.docx	19.8 KB	Microsoft Word Document	Tuesday
KTI_concept_note_Trift_HES_SO.docx	38.1 KB	Microsoft Word Document	Today
KTI_concept_note_Trift_VAW.docx	20.9 KB	Microsoft Word Document	Today
KTI_concept_note_TriftV22oct14.docx	23.3 KB	Microsoft Word Document	Today

Version control: what is it?

- A sophisticated backup

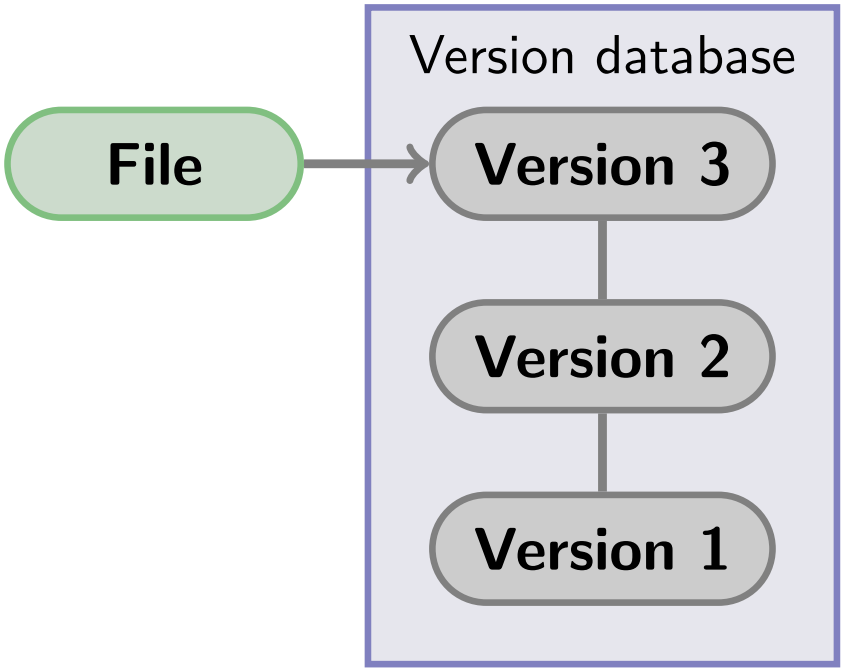
- A platform for collaboration

Version control: what is it?

- A sophisticated backup
  - Save useful versions of your work
  - Preserve specific versions of your documents
  - Keep a record of who made changes and when
- A platform for collaboration
  - Collaborators can work simultaneously on the same set of documents
  - Changes of all collaborators are preserved

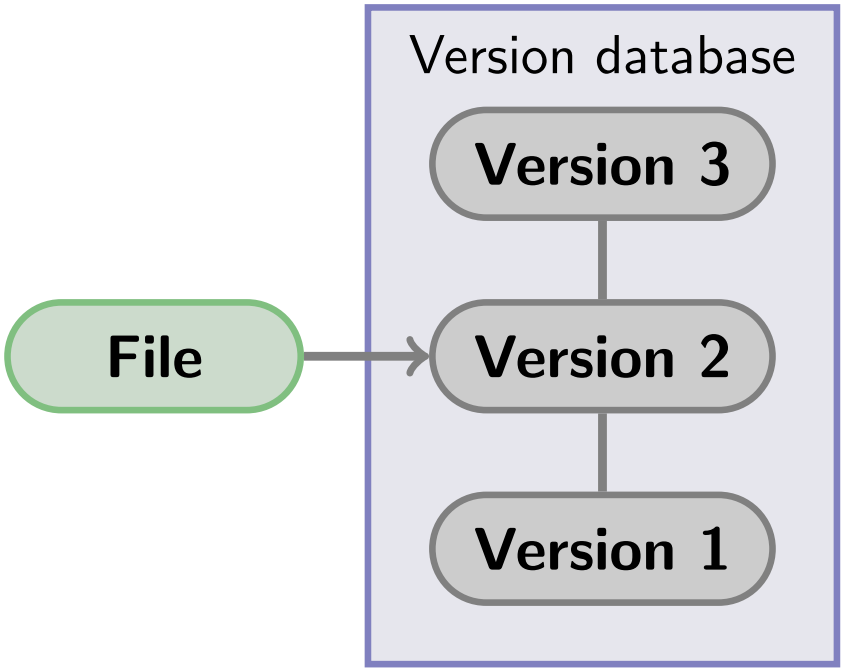
Version control: what is it?

- A sophisticated backup
  - Save useful versions of your work
  - Preserve specific versions of your documents
  - Keep a record of who made changes and when
- A platform for collaboration
  - Collaborators can work simultaneously on the same set of documents
  - Changes of all collaborators are preserved



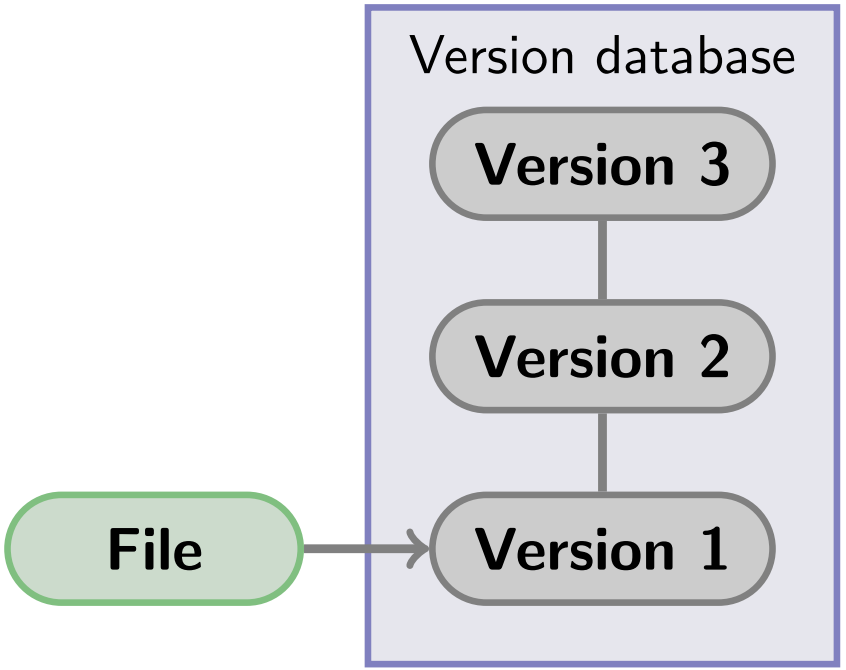
Version control: what is it?

- A sophisticated backup
  - Save useful versions of your work
  - Preserve specific versions of your documents
  - Keep a record of who made changes and when
- A platform for collaboration
  - Collaborators can work simultaneously on the same set of documents
  - Changes of all collaborators are preserved



Version control: what is it?

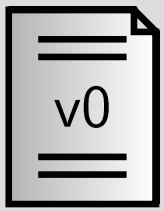
- A sophisticated backup
  - Save useful versions of your work
  - Preserve specific versions of your documents
  - Keep a record of who made changes and when
- A platform for collaboration
  - Collaborators can work simultaneously on the same set of documents
  - Changes of all collaborators are preserved



Version control: use case

Collaboration through e-mail:

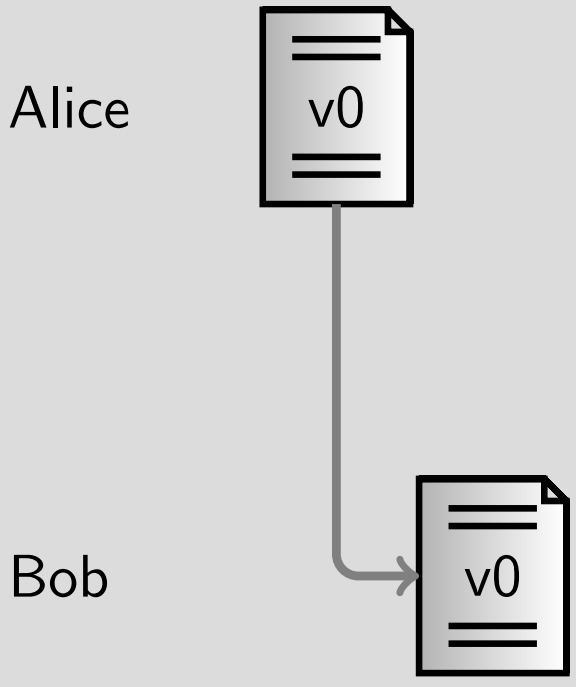
Alice



Bob

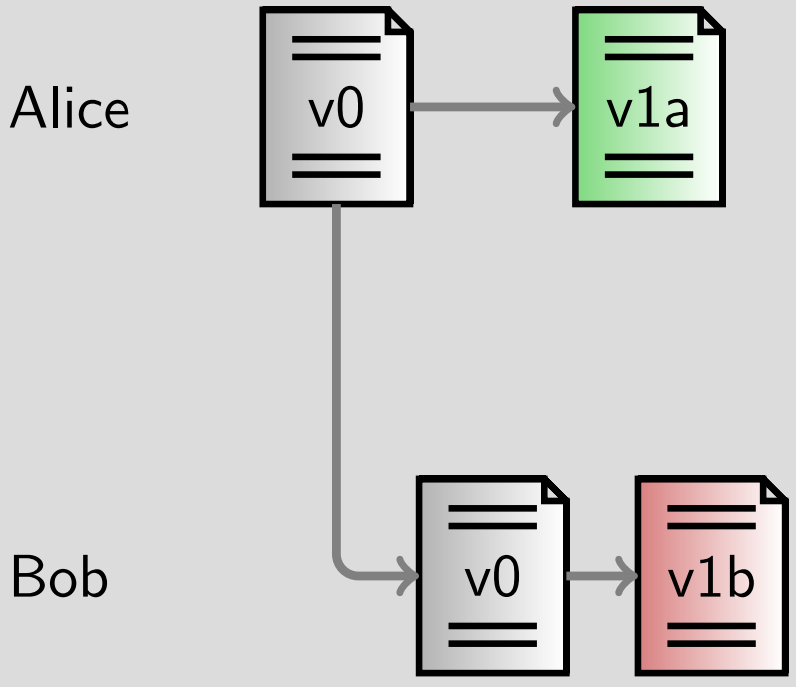
Version control: use case

Collaboration through e-mail:



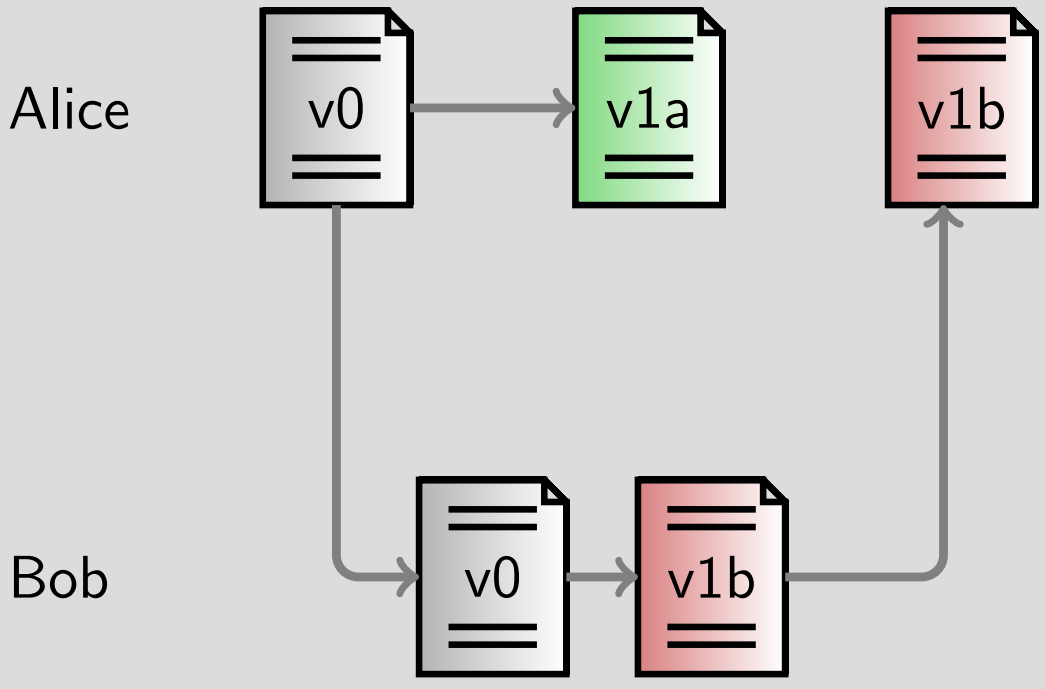
Version control: use case

Collaboration through e-mail:



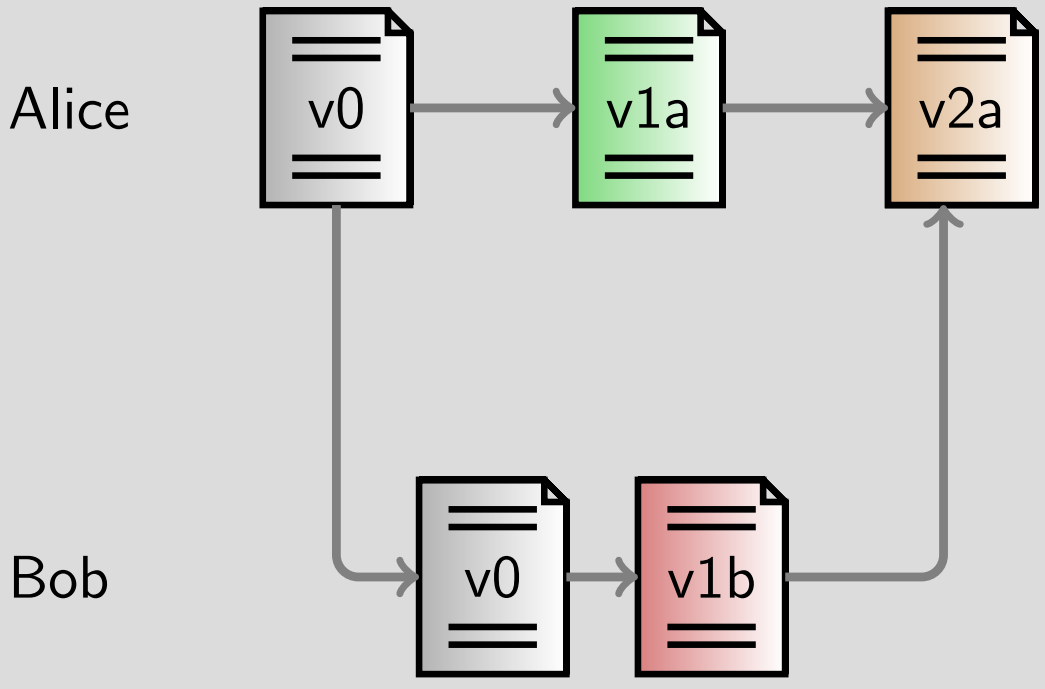
Version control: use case

Collaboration through e-mail:



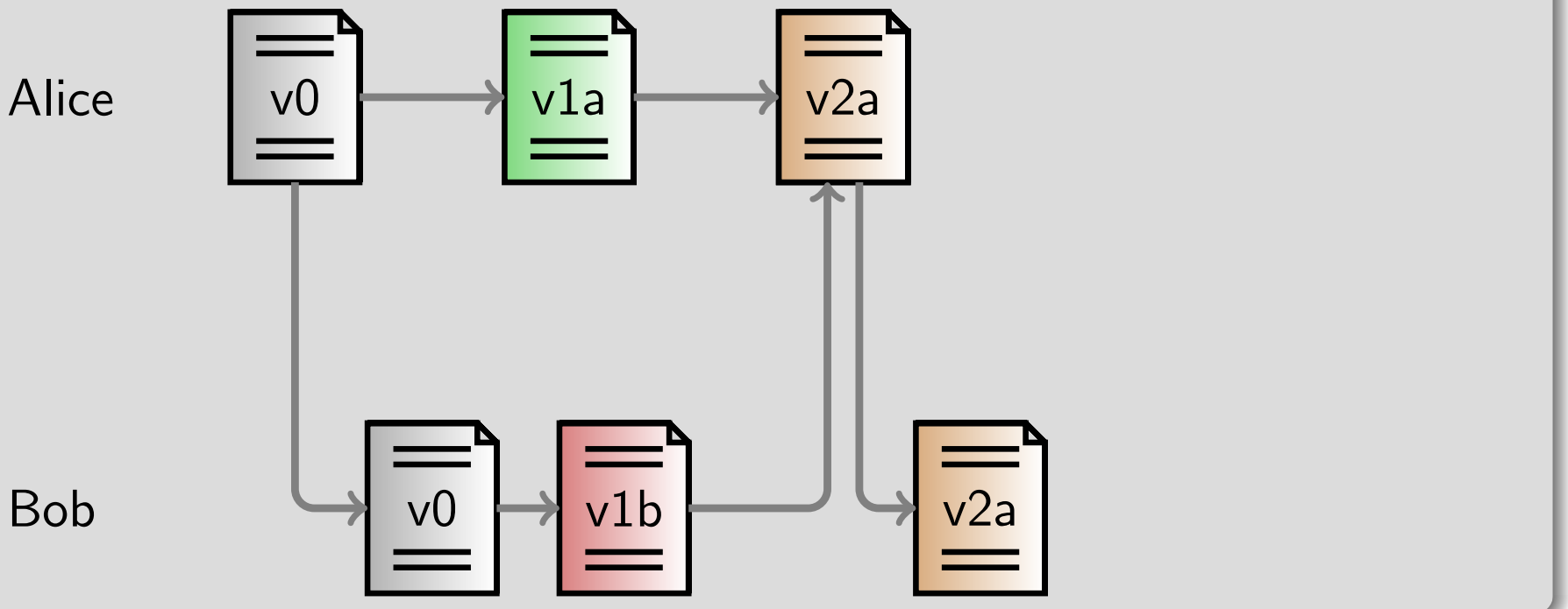
Version control: use case

Collaboration through e-mail:



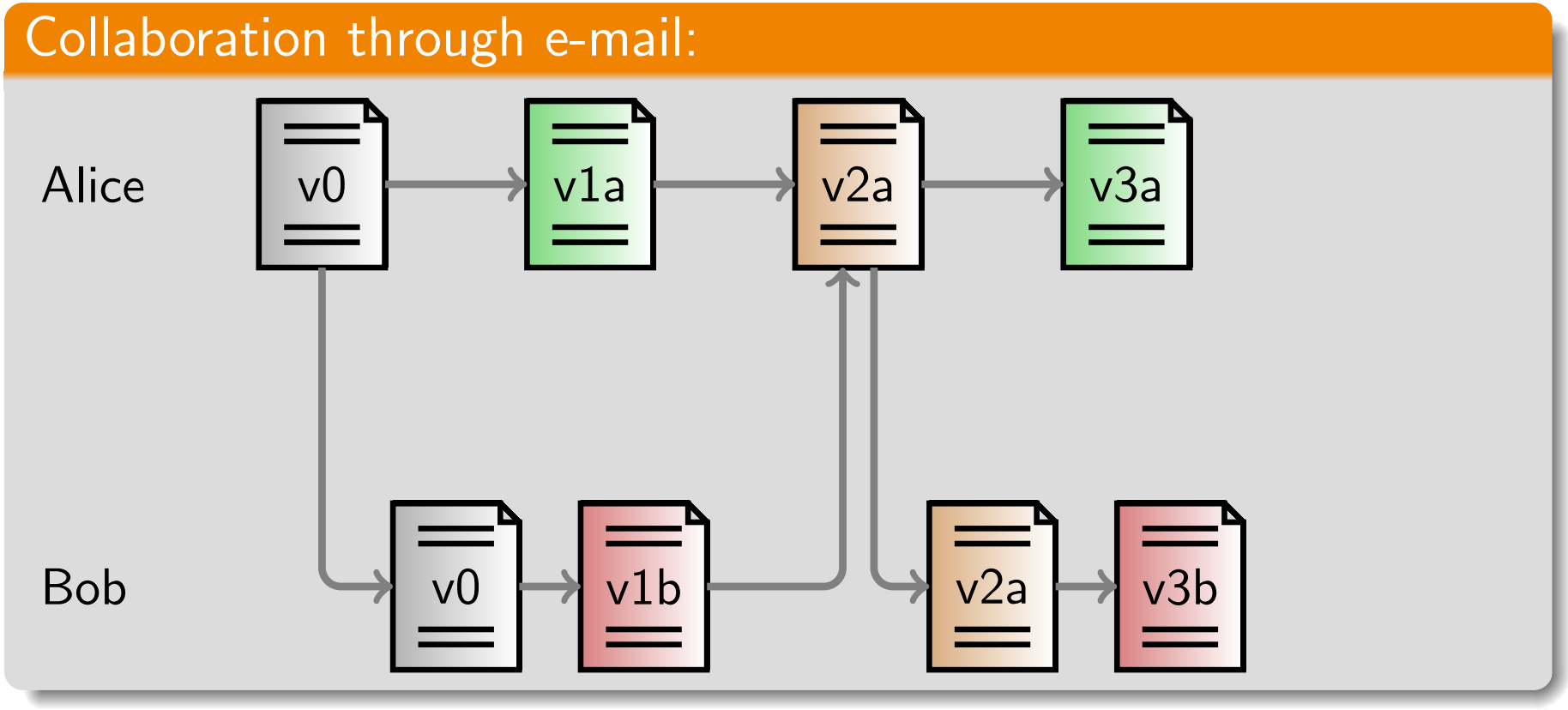
Version control: use case

Collaboration through e-mail:



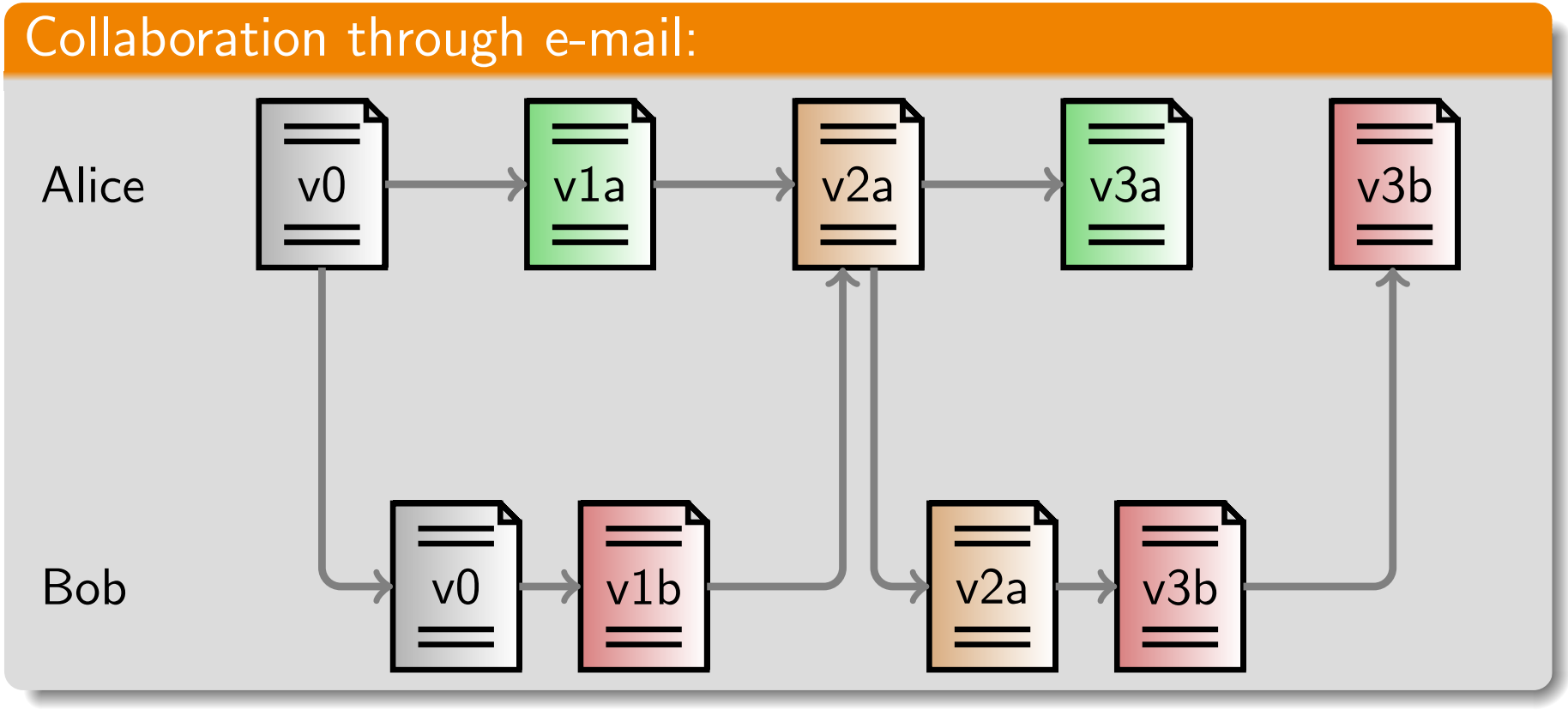
Version control: use case

Collaboration through e-mail:



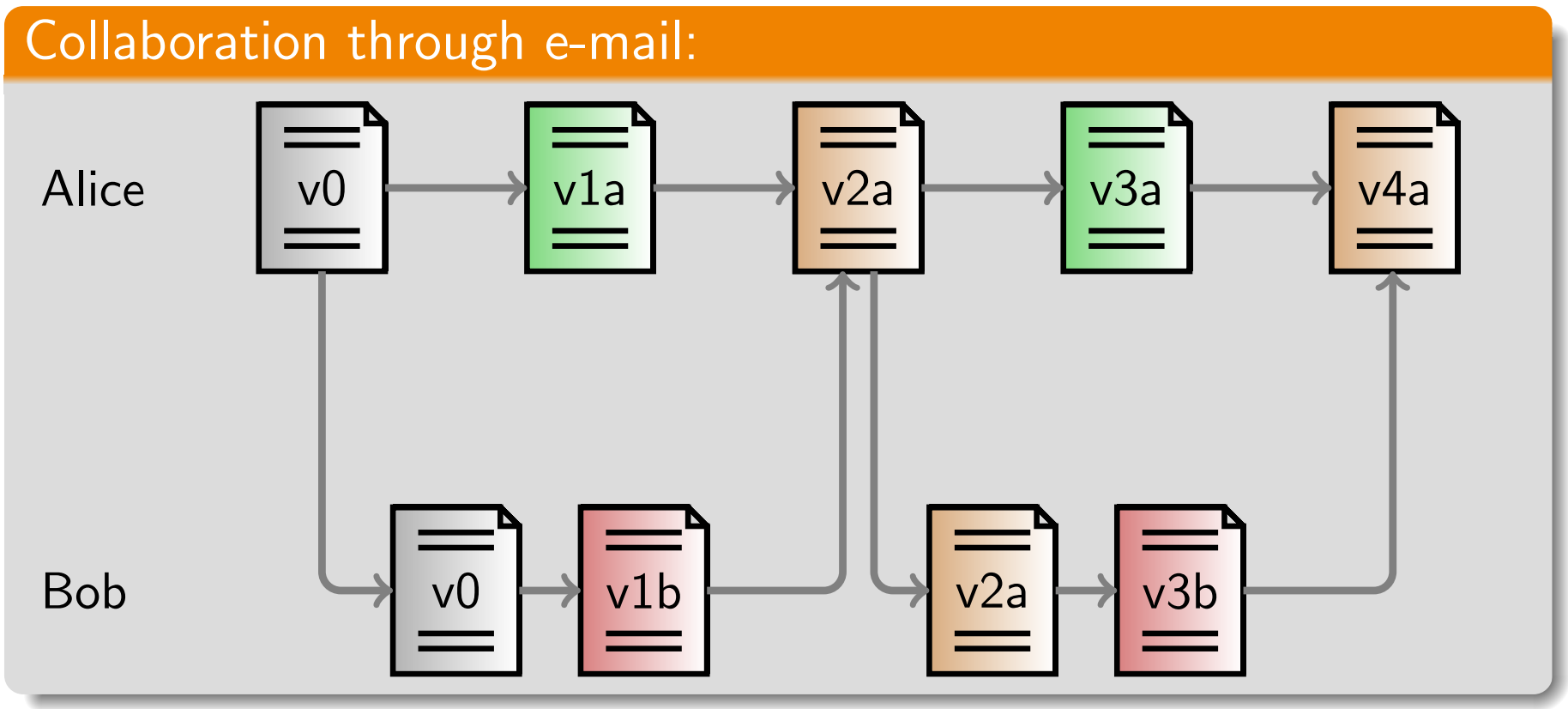
Version control: use case

Collaboration through e-mail:



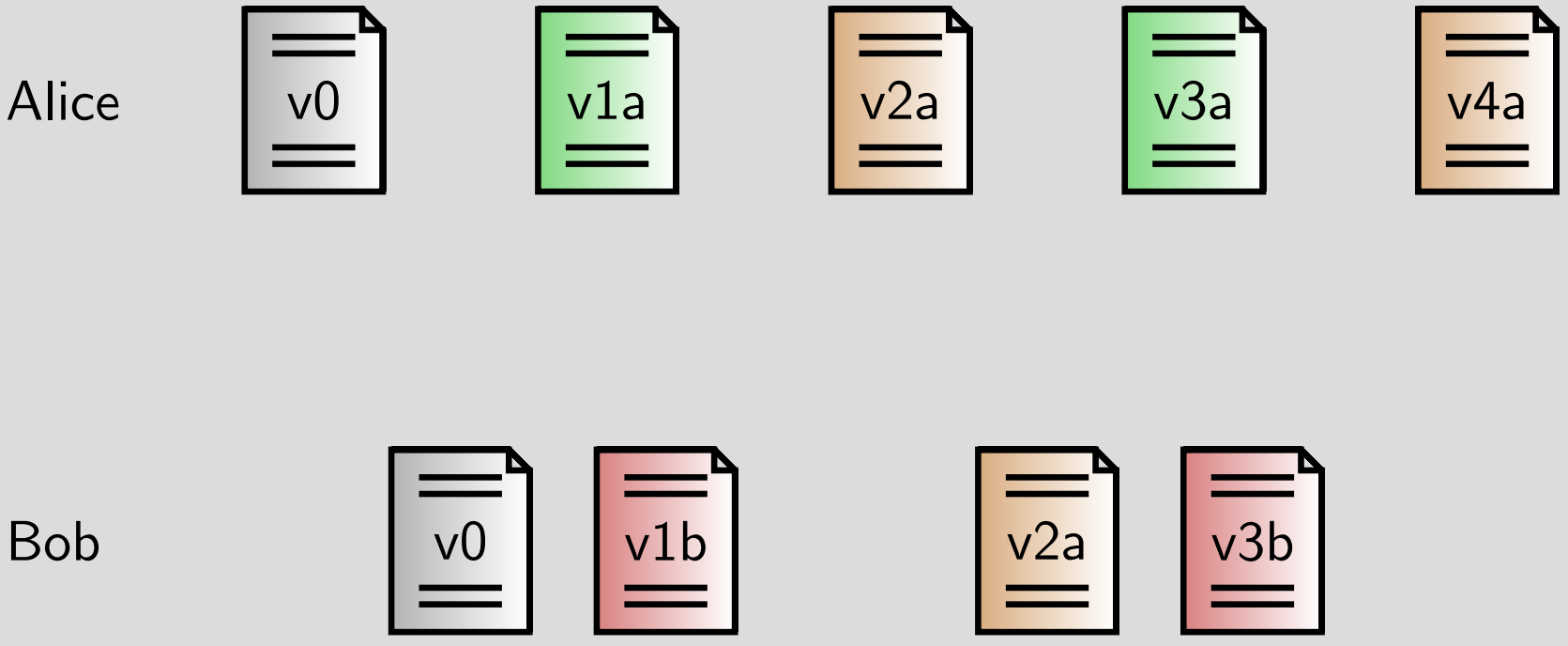
Version control: use case

Collaboration through e-mail:



Version control: use case

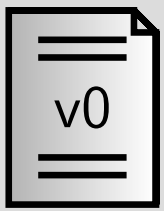
Collaboration through e-mail:



Version control: use case

Collaboration through central data storage:

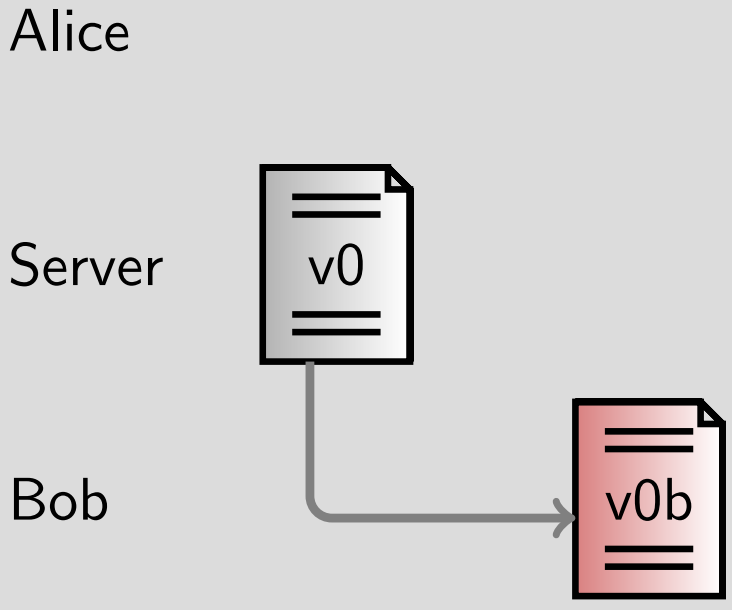
Alice

Server 

Bob

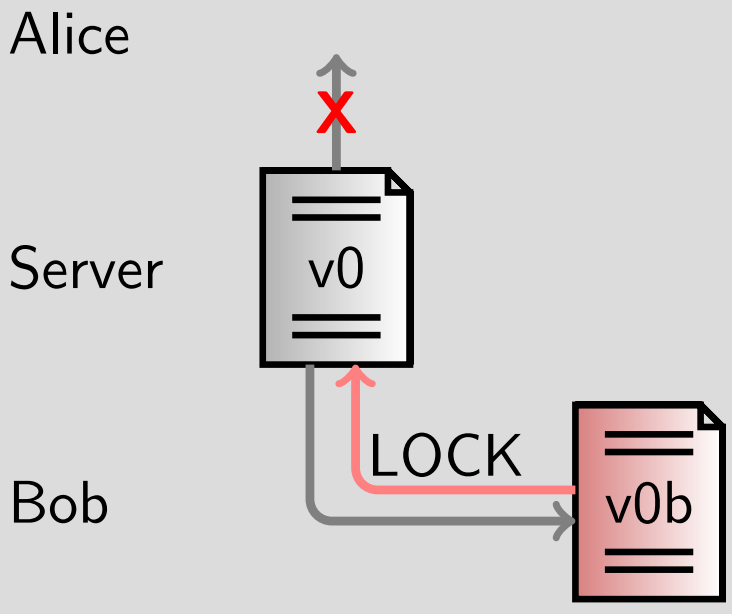
Version control: use case

Collaboration through central data storage:



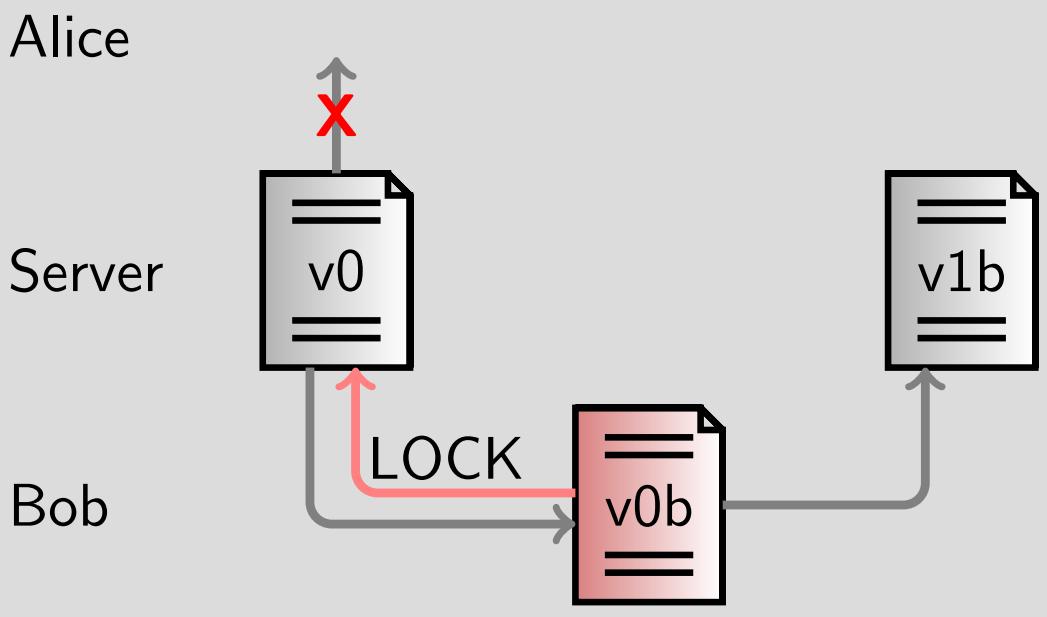
Version control: use case

Collaboration through central data storage:



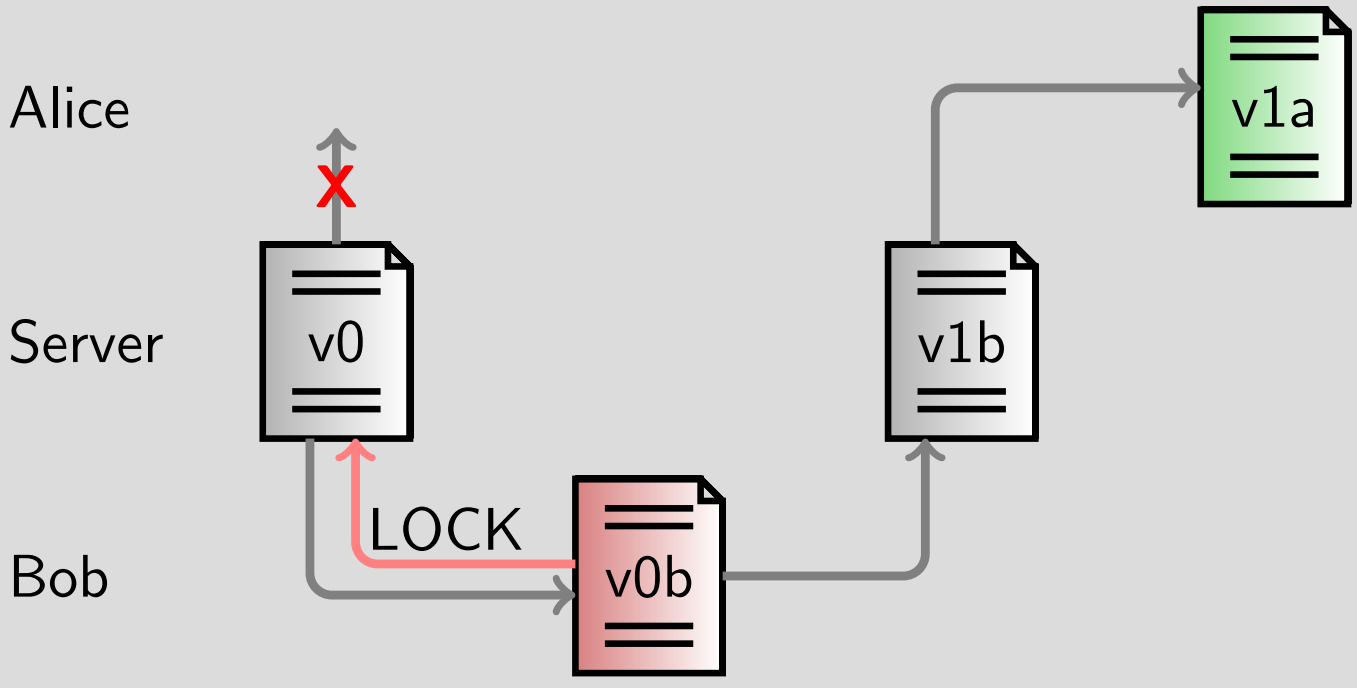
Version control: use case

Collaboration through central data storage:



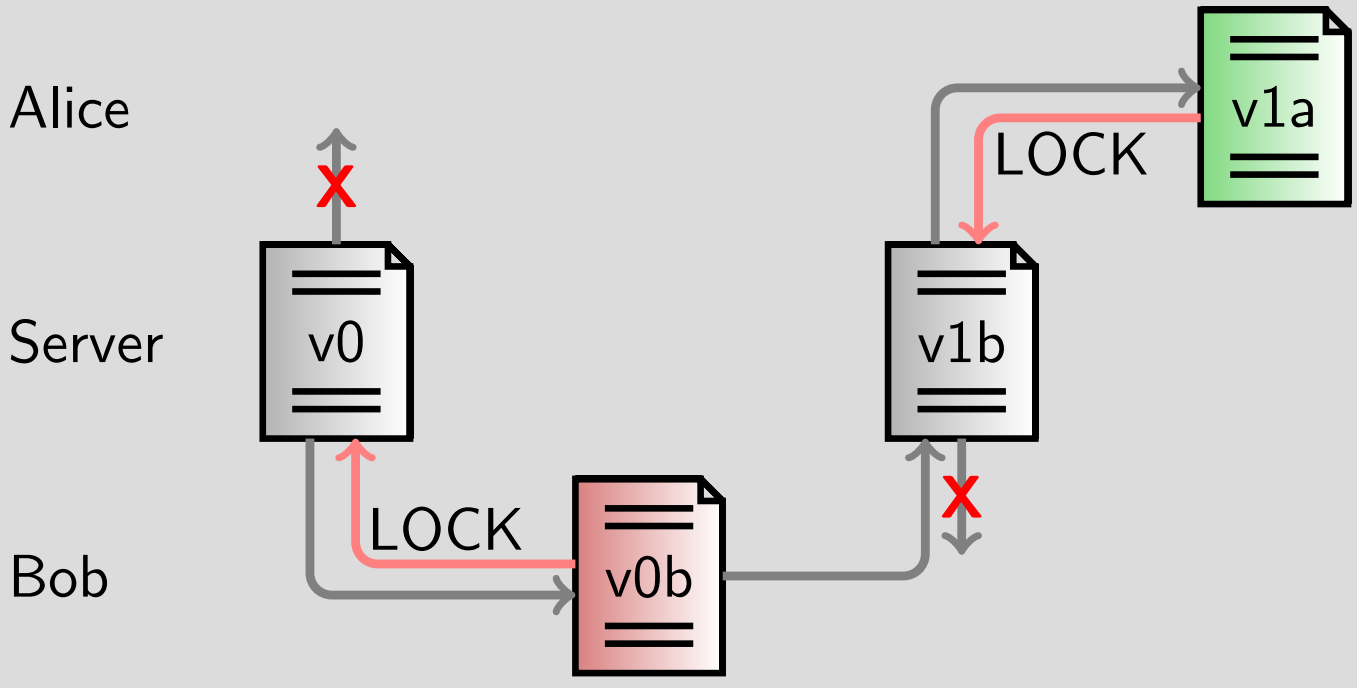
Version control: use case

Collaboration through central data storage:



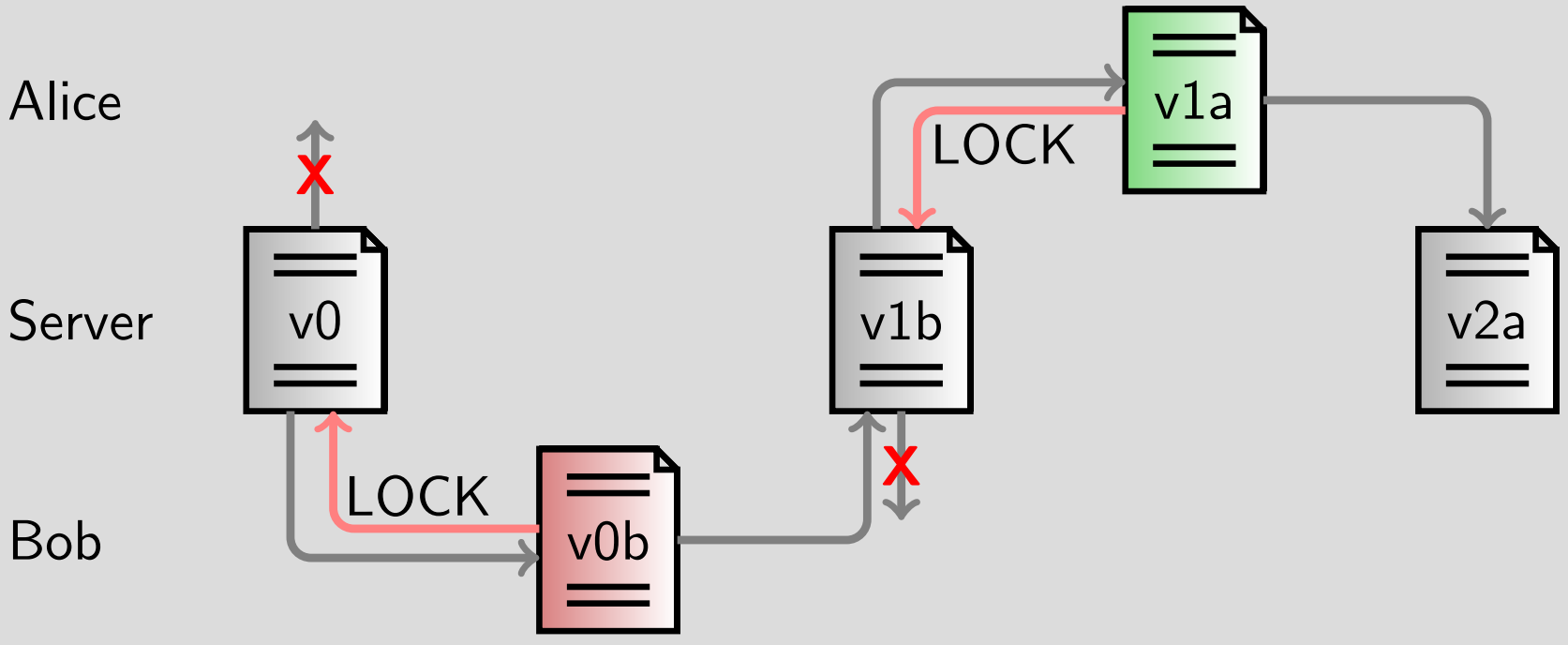
Version control: use case

Collaboration through central data storage:



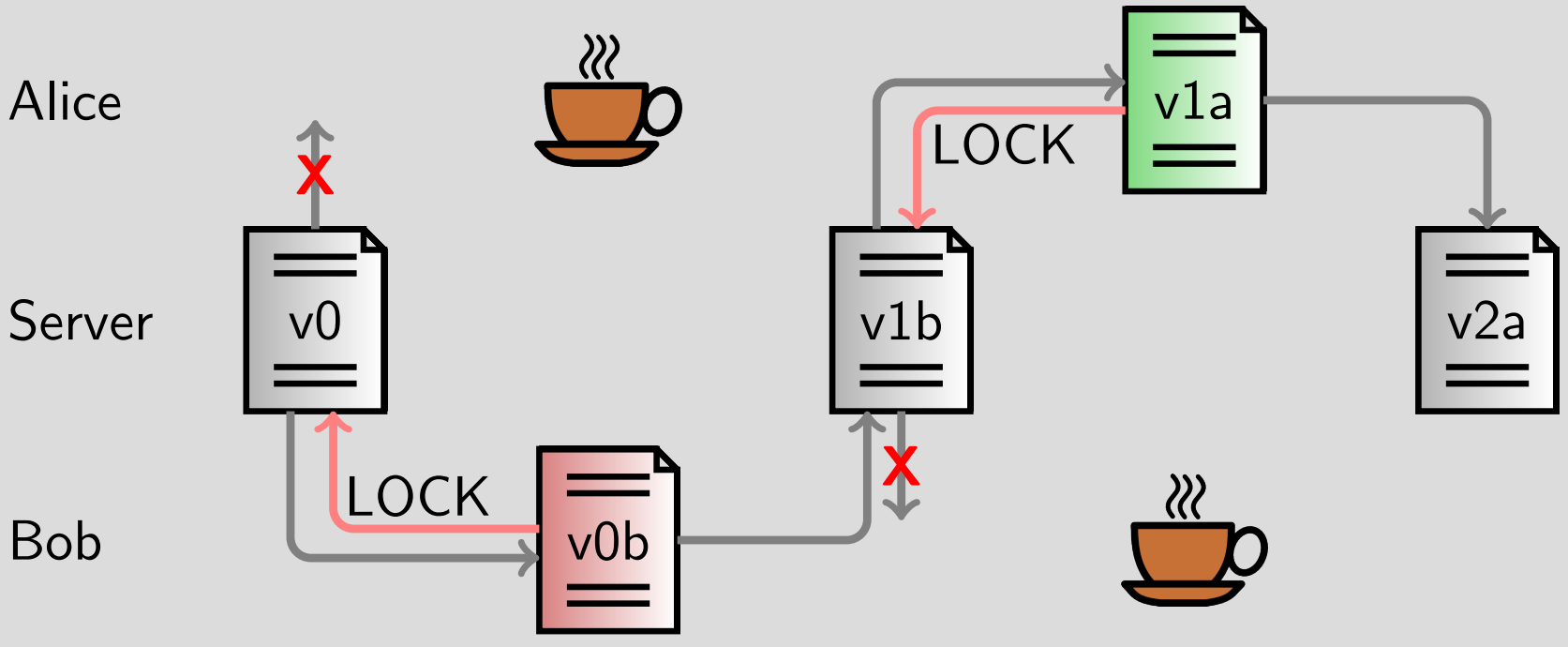
Version control: use case

Collaboration through central data storage:



Version control: use case

Collaboration through central data storage:

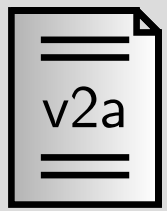


Version control: use case

Collaboration through central data storage:

Alice

Server



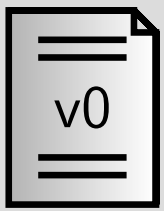
Bob

Version control: use case

Collaboration with version control system:

Alice

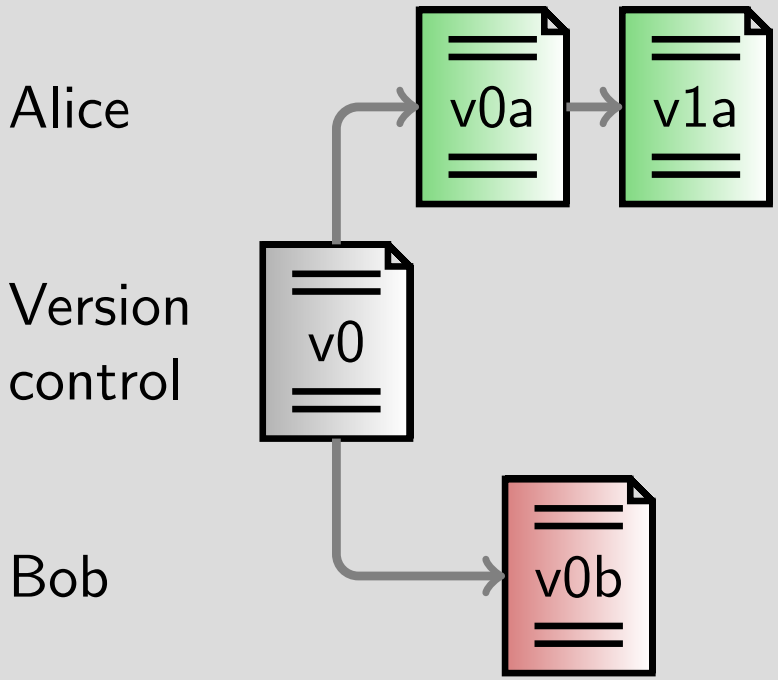
Version control



Bob

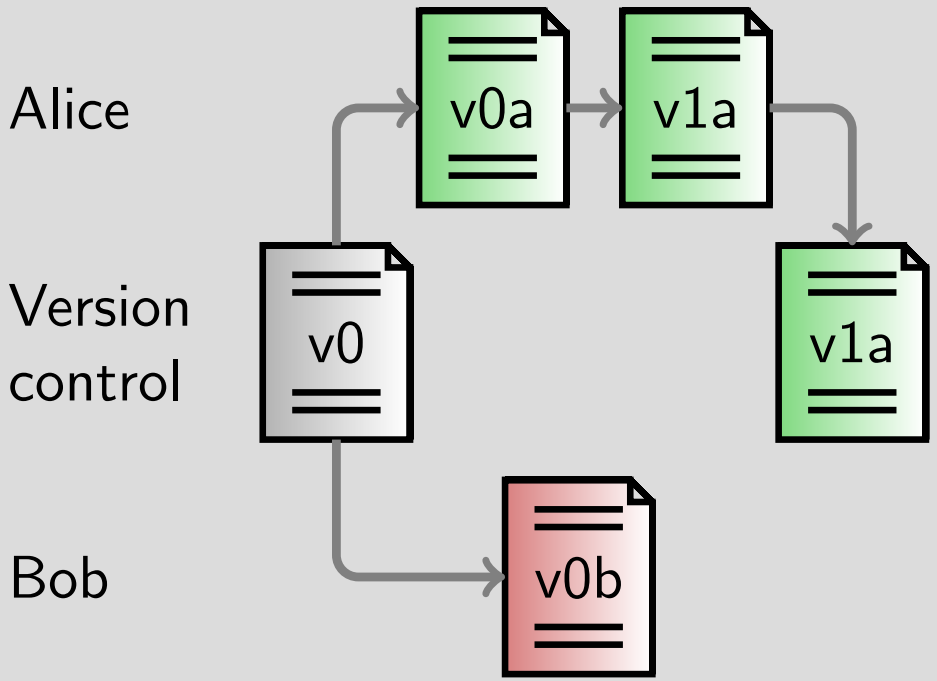
Version control: use case

Collaboration with version control system:



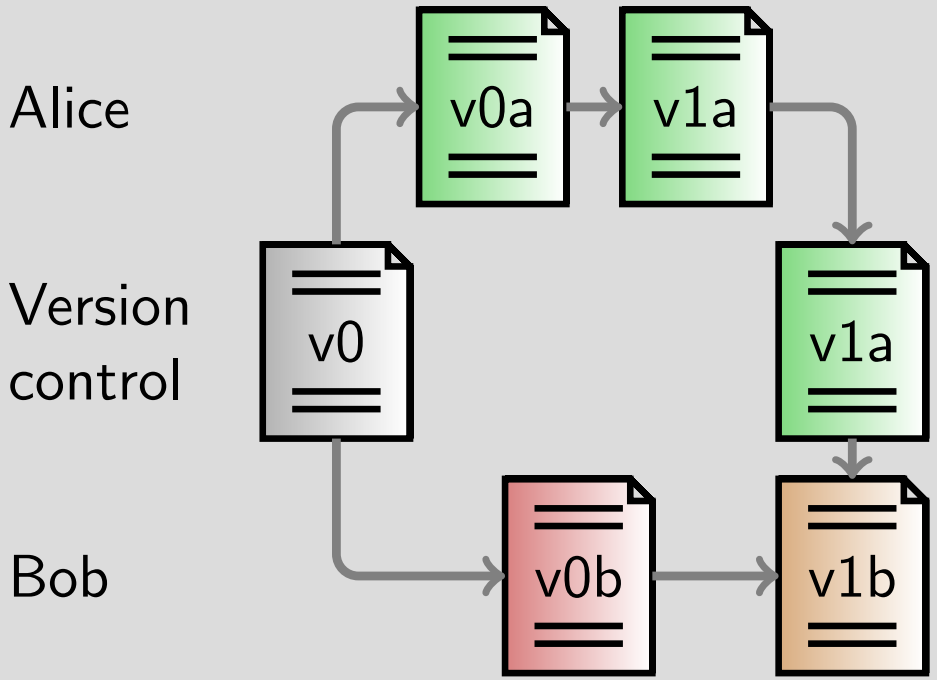
Version control: use case

Collaboration with version control system:



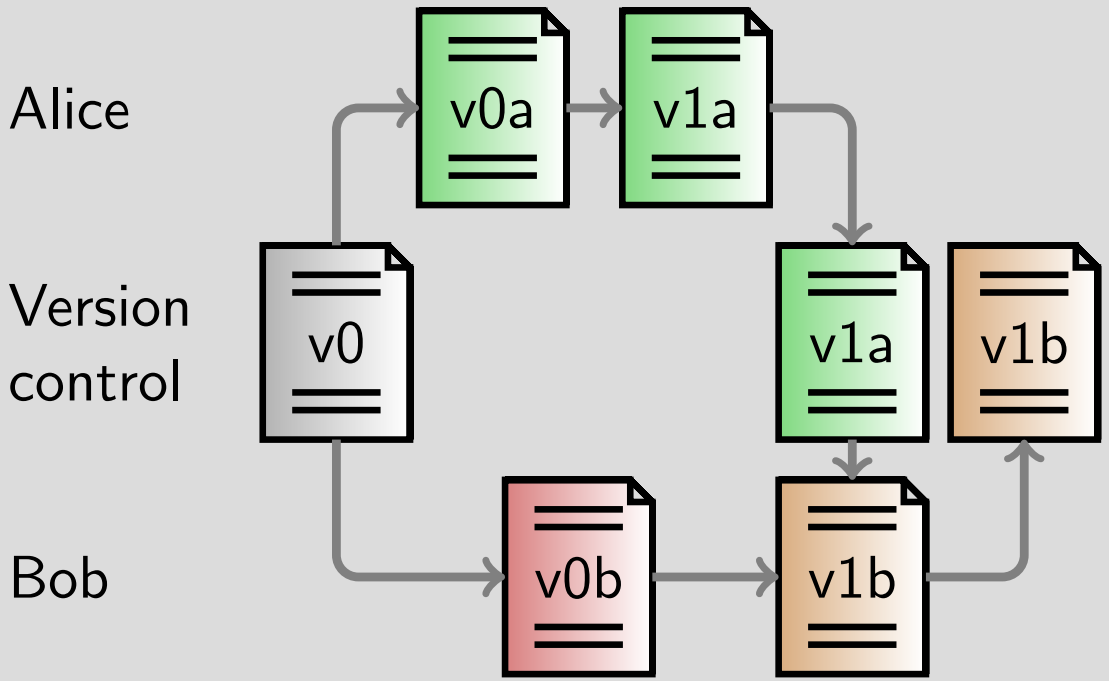
Version control: use case

Collaboration with version control system:



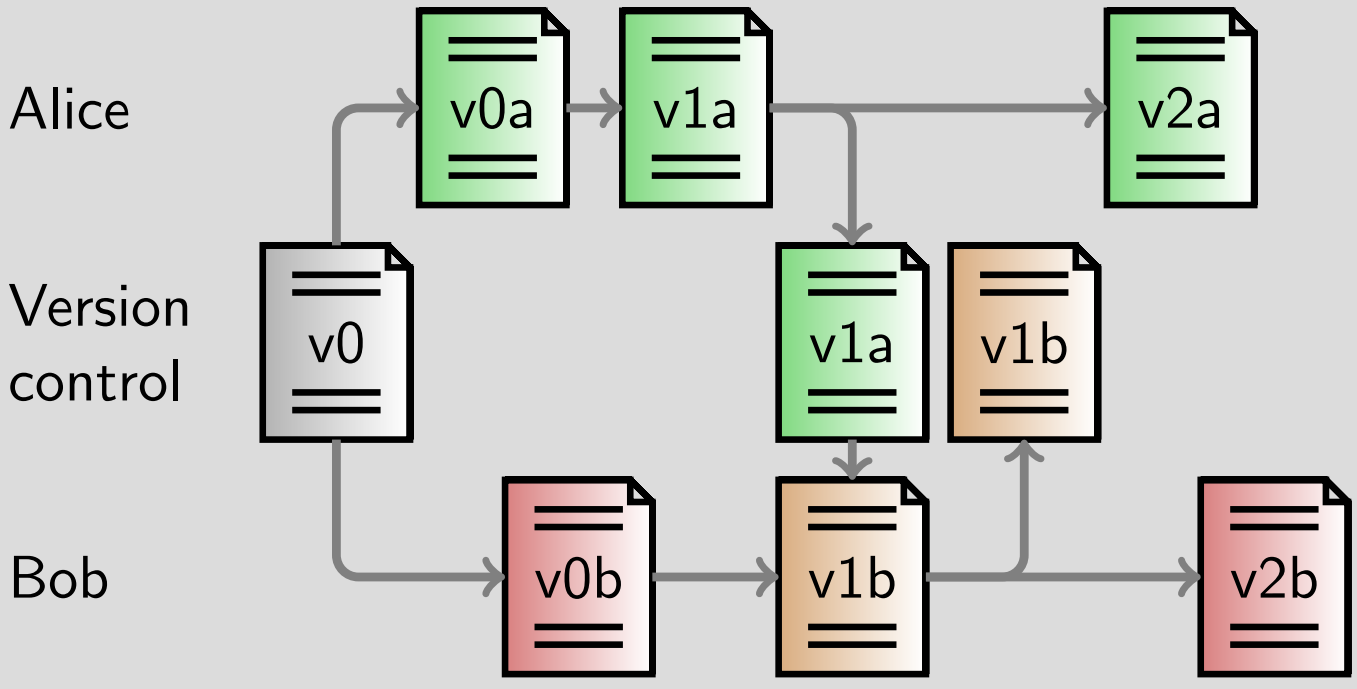
Version control: use case

Collaboration with version control system:



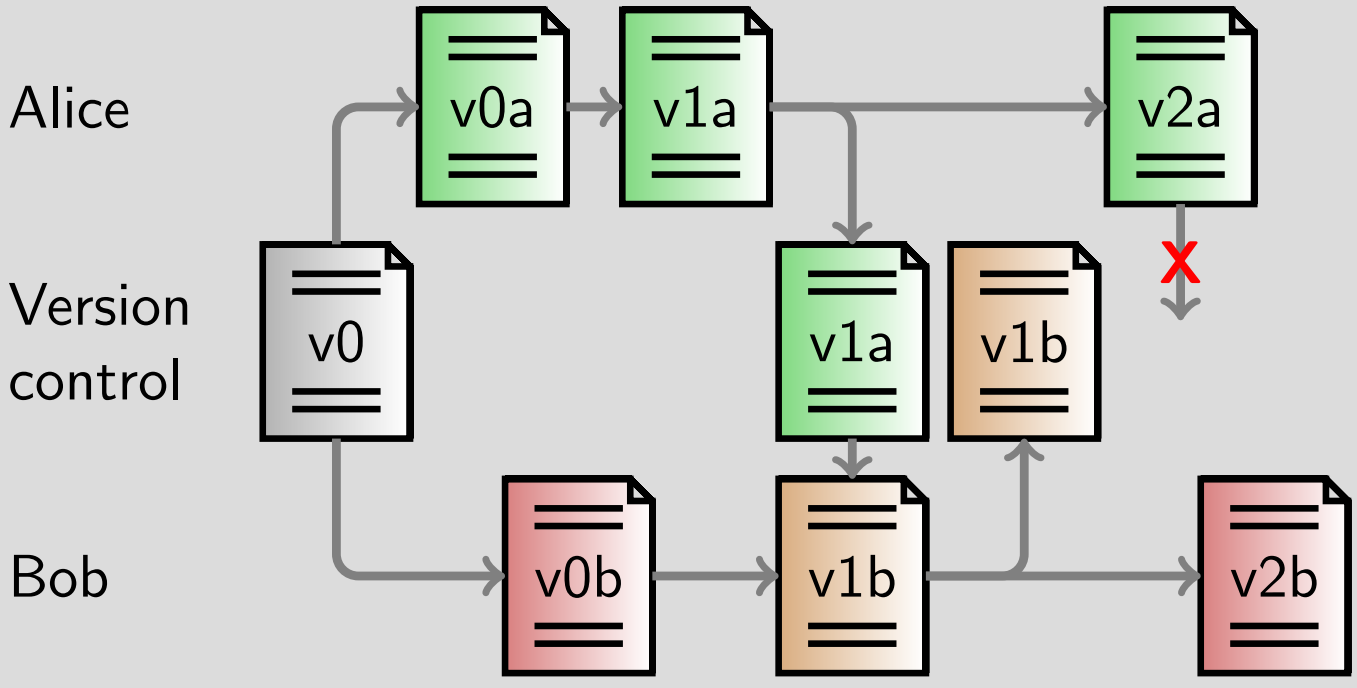
Version control: use case

Collaboration with version control system:



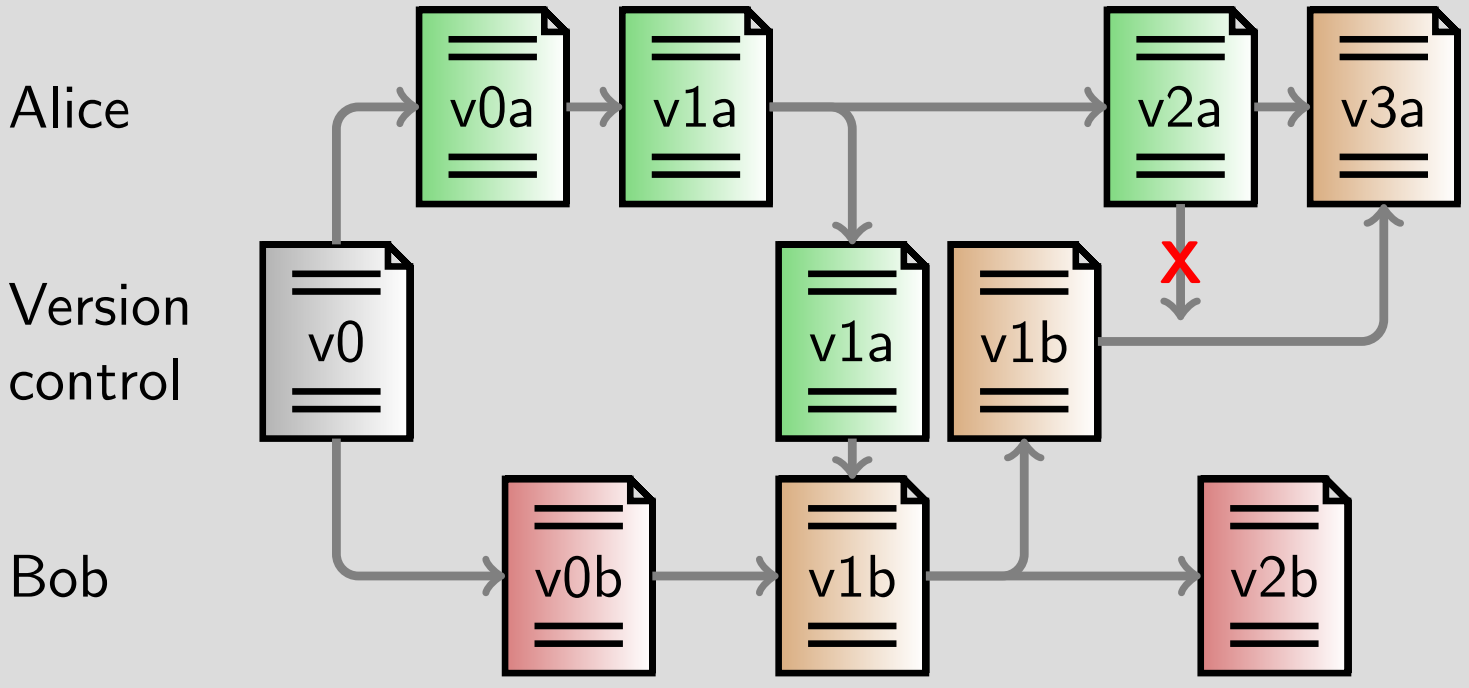
Version control: use case

Collaboration with version control system:



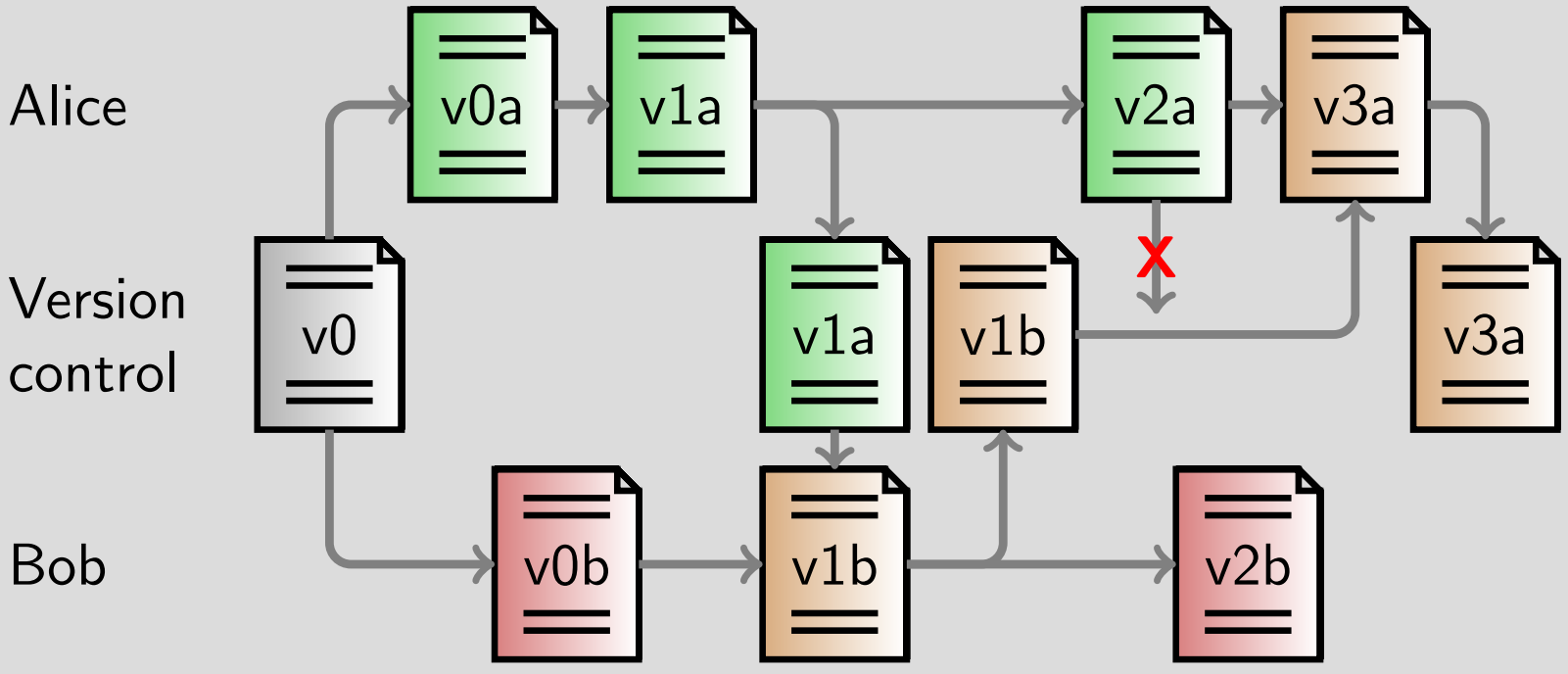
Version control: use case

Collaboration with version control system:



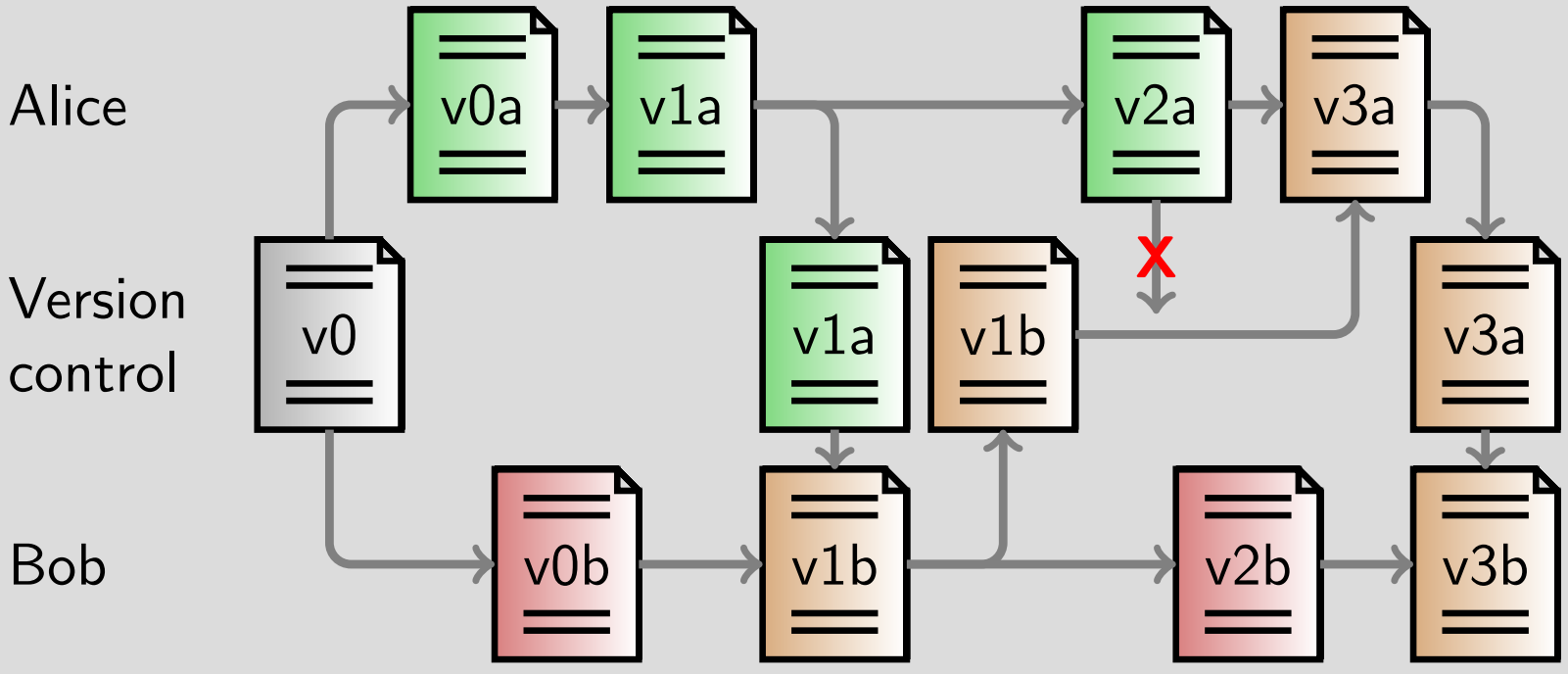
Version control: use case

Collaboration with version control system:



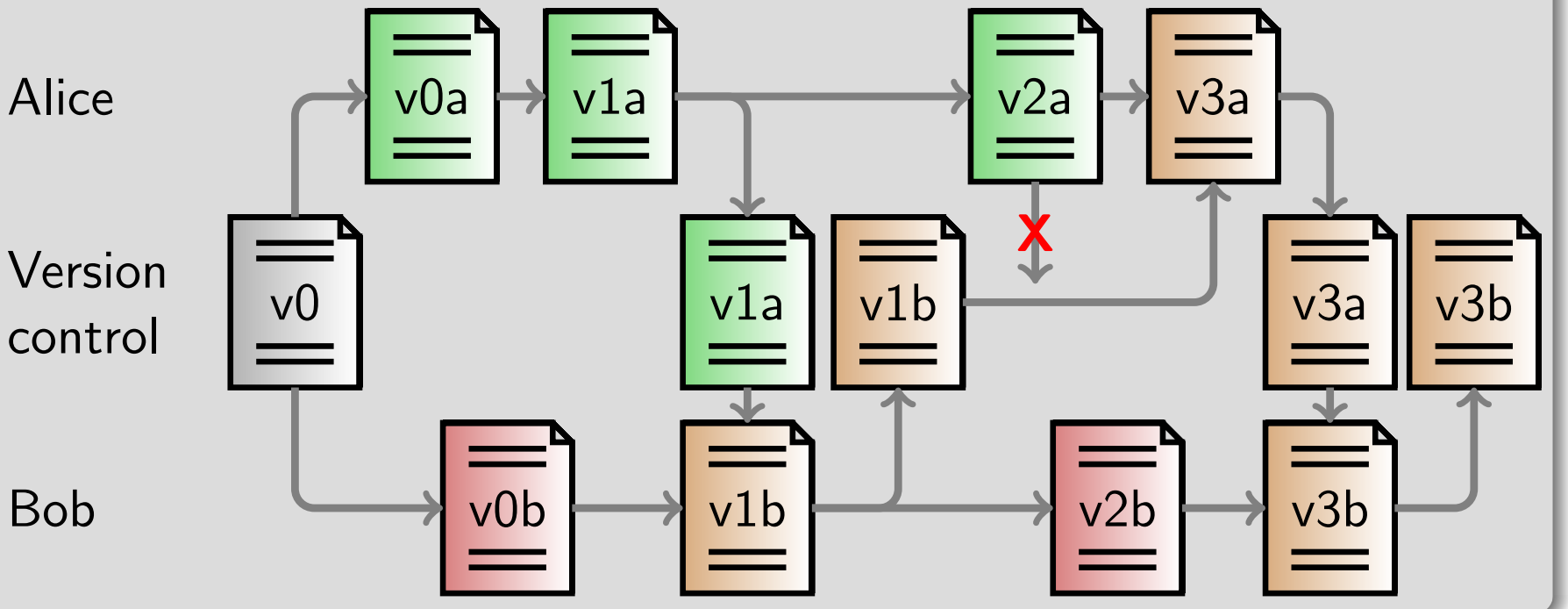
Version control: use case

Collaboration with version control system:



Version control: use case

Collaboration with version control system:

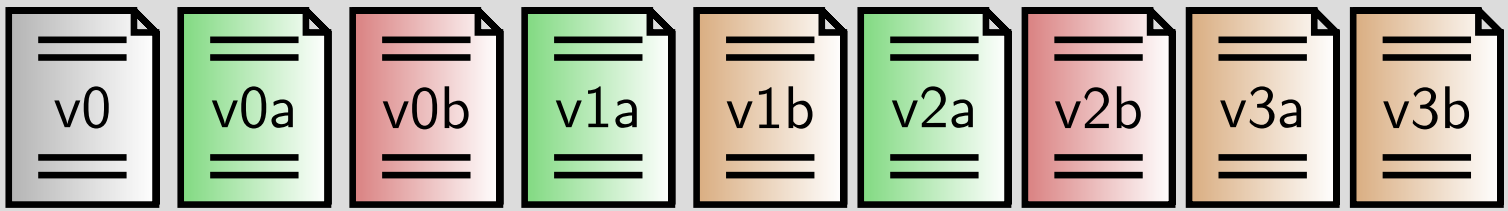


Version control: use case

Collaboration with version control system:

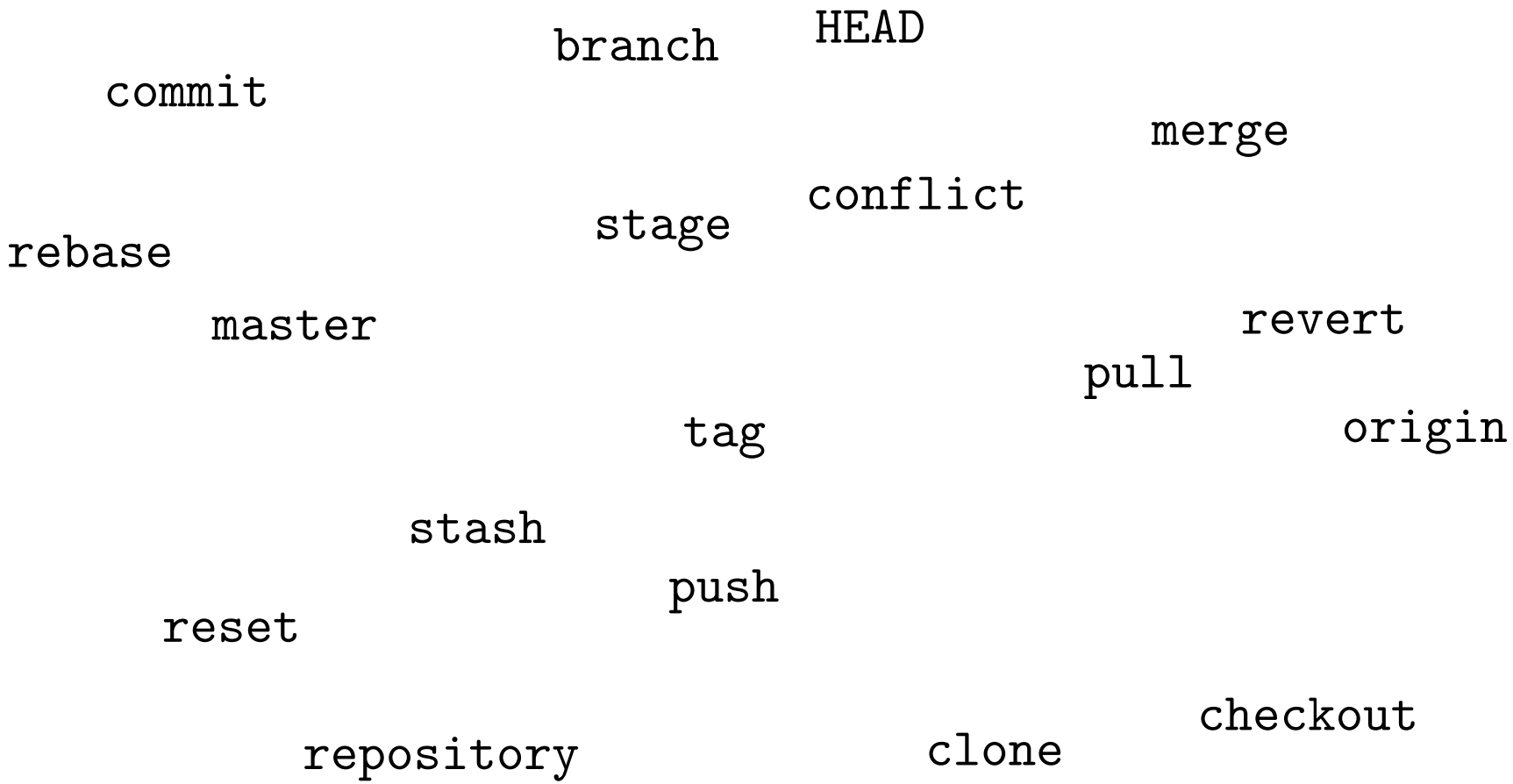
Alice

Version control



Bob

Version control: a secret language?



What is git?

- Built as version control tool for the Linux kernel
  - Built for speed
  - Designed for large projects
  - Designed for many collaborators
- Distributed version control system
  - Fast: all operations are performed locally
  - No backup needed: every copy holds the full database
  - No network connection needed
- Records a snapshot for each version of files
  - Files can be reverted to any state
  - Allows working on different versions simultaneously
- Works best with line based text files
  - Source code etc.

Basic commands

Basic commands

Create repository

```
git init
```

Add files

```
git add <file>
```

Basic commands

Create repository

```
git init
```

Add files

```
git add <file>
```

Commit

```
git commit -m "Commit message."
```

Commit all changes:

```
git commit -a -m "Commit message."
```

Basic commands

Create repository

```
git init
```

Add files

```
git add <file>
```

Commit

```
git commit -m "Commit message."
```

Commit all changes:

```
git commit -a -m "Commit message."
```

Create branch

```
git branch <branch name>
```

Basic commands

Create repository

```
git init
```

Add files

```
git add <file>
```

Commit

```
git commit -m "Commit message."
```

Commit all changes:

```
git commit -a -m "Commit message."
```

Create branch

```
git branch <branch name>
```

Switch version

```
git checkout <branch>
git checkout <tag>
git checkout <version>
```

Basic commands

Create repository

```
git init
```

Add files

```
git add <file>
```

Commit

```
git commit -m "Commit message."
```

Commit all changes:

```
git commit -a -m "Commit message."
```

Create branch

```
git branch <branch name>
```

Stash changes

```
git stash
git stash list
git stash apply
```

Switch version

```
git checkout <branch>
git checkout <tag>
git checkout <version>
```

Basic commands

Create repository

```
git init
```

Add files

```
git add <file>
```

Commit

```
git commit -m "Commit message."
```

Commit all changes:

```
git commit -a -m "Commit message."
```

Create branch

```
git branch <branch name>
```

Show status

```
git status
```

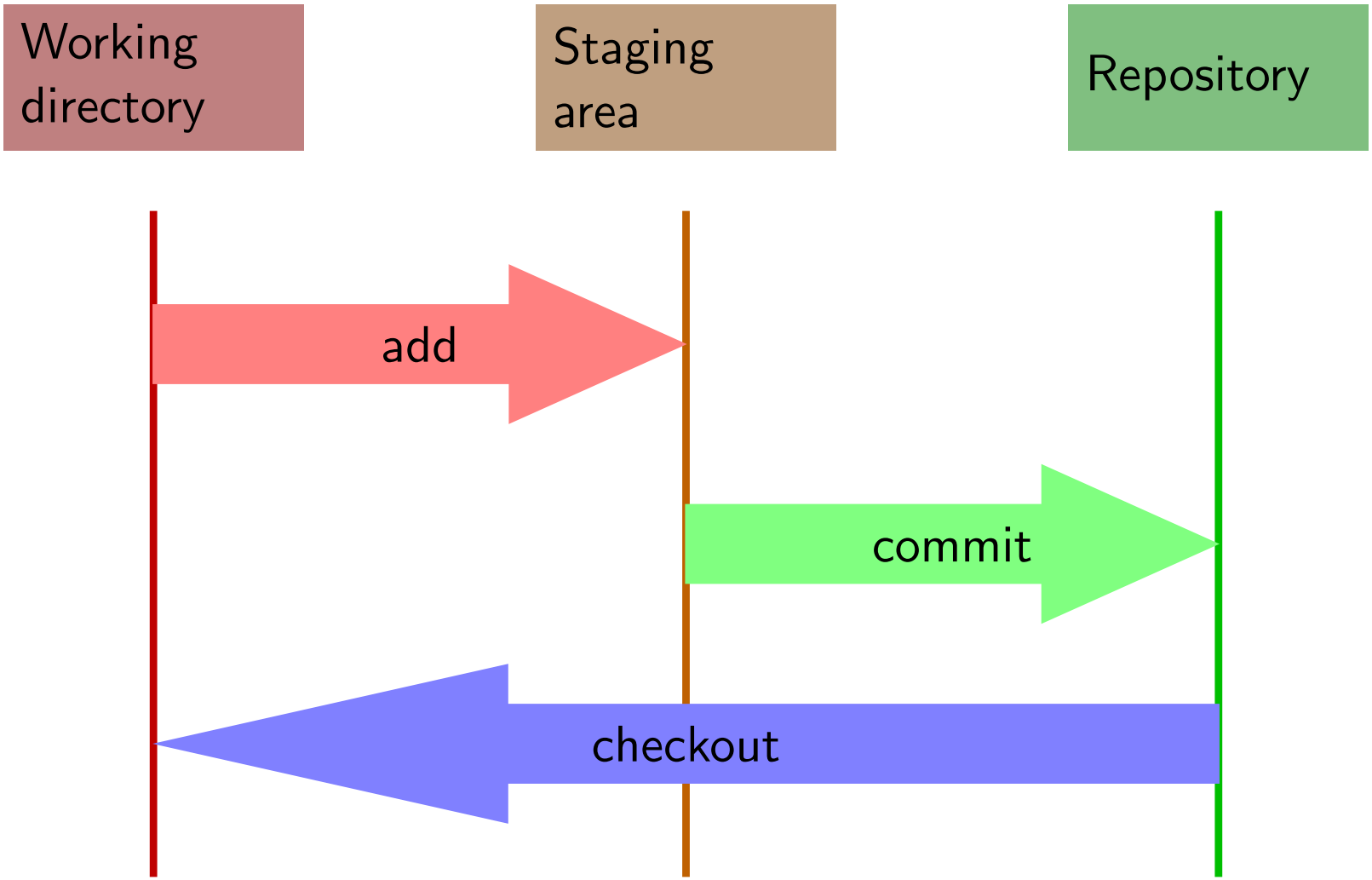
Stash changes

```
git stash
git stash list
git stash apply
```

Switch version

```
git checkout <branch>
git checkout <tag>
git checkout <version>
```

Basic commands



Basic commands

Create repository

```
git init
```

Add files

```
git add <file>
```

Commit

```
git commit -m "Commit message."
```

Commit all changes:

```
git commit -a -m "Commit message."
```

Create branch

```
git branch <branch name>
```

Show status

```
git status
```

Stash changes

```
git stash
git stash list
git stash apply
```

Switch version

```
git checkout <branch>
git checkout <tag>
git checkout <version>
```

Basic commands

Create repository

```
git init
```

Add files

```
git add <file>
```

Commit

```
git commit -m "Commit message."
```

Commit all changes:

```
git commit -a -m "Commit message."
```

Create branch

```
git branch <branch name>
```

Show status

```
git status
```

Create tag

```
git tag -a <tag> -m"Description"
git tag
```

Stash changes

```
git stash
git stash list
git stash apply
```

Switch version

```
git checkout <branch>
git checkout <tag>
git checkout <version>
```

Basic commands

Create repository

```
git init
```

Add files

```
git add <file>
```

Revert changes

```
git reset --hard
```

Show status

```
git status
```

Stash changes

```
git stash
git stash list
git stash apply
```

Commit

```
git commit -m "Commit message."
```

Commit all changes:

```
git commit -a -m "Commit message."
```

Create branch

```
git branch <branch name>
```

Create tag

```
git tag -a <tag> -m"Description"
git tag
```

Switch version

```
git checkout <branch>
git checkout <tag>
git checkout <version>
```

Basic commands

Create repository

```
git init
```

Add files

```
git add <file>
```

Revert changes

```
git reset --hard
```

Show status

```
git status
```

Stash changes

```
git stash
git stash list
git stash apply
```

Commit

```
git commit -m "Commit message."
```

Commit all changes:

```
git commit -a -m "Commit message."
```

Create branch

```
git branch <branch name>
```

Merge

```
git merge <other branch>
```

Create tag

```
git tag -a <tag> -m"Description"
git tag
```

Switch version

```
git checkout <branch>
git checkout <tag>
git checkout <version>
```

git

Links

git: `http://git-scm.com/`
GitHub: `http://github.com`

GUIs:

- **TortoiseGit**: `https://code.google.com/p/tortoisegit/`
- **SmartGit**: `http://www.syntevo.com/smartgit/`

Editors:

- **Vim**: `http://www.vim.org/`
- GitGutter for vim:
  `https://github.com/airblade/vim-gitgutter`
- **Sublime**: `http://www.sublimetext.com/`
- GitGutter for sublime:
  `https://github.com/jisaacks/GitGutter`