G52GRP 2012–2013: Lecture 3

Project Site and Version Control with Subversion

Henrik Nilsson

University of Nottingham, UK

This Lecture

- Indefero project site
- Why use version control systems?
- Subversion
- Using Subversion

Sharing Code and Documents (1)

Sharing Code and Documents (1)

- Passing copies from person to person using e.g. e-mail or USB memory sticks?

Sharing Code and Documents (1)

- Passing copies from person to person using e.g. e-mail or USB memory sticks?

  Might work for a single document where people “take turns”, but otherwise recipe for disaster!

Sharing Code and Documents (1)

- Passing copies from person to person using e.g. e-mail or USB memory sticks?

  Might work for a single document where people “take turns”, but otherwise recipe for disaster!

  - Who’s got the latest version?

Sharing Code and Documents (1)

- Passing copies from person to person using e.g. e-mail or USB memory sticks?

  Might work for a single document where people “take turns”, but otherwise recipe for disaster!

  - Who’s got the latest version?
  - Who’s got the right to edit?

Sharing Code and Documents (1)

- Passing copies from person to person using e.g. e-mail or USB memory sticks?

  Might work for a single document where people “take turns”, but otherwise recipe for disaster!

  - Who’s got the latest version?
  - Who’s got the right to edit?
  - How to ensure that everyone sees up-to-date versions of everything?
  - . . .

Sharing Code and Documents (2)

- A shared repository is a better idea!

Sharing Code and Documents (2)

- A shared repository is a better idea!
  - A School-hosted Indefero project has been set up for each group. Sign in with your CS credentials at:
    `https://code.cs.nott.ac.uk`

Sharing Code and Documents (2)

- A shared repository is a better idea!
  - A School-hosted Indefero project has been set up for each group. Sign in with your CS credentials at:
    `https://code.cs.nott.ac.uk`
  - Provides additional project management features beyond a shared repository.

Sharing Code and Documents (2)

- A shared repository is a better idea!
  - A School-hosted Indefero project has been set up for each group. Sign in with your CS credentials at:
    `https://code.cs.nott.ac.uk`
  - Provides additional project management features beyond a shared repository.
  - The designated `gp12-XXX` Indefero project should (in most cases) be your central repository for code and documentation.

Sharing Code and Documents (2)

- A shared repository is a better idea!
  - A School-hosted Indefero project has been set up for each group. Sign in with your CS credentials at:
    `https://code.cs.nott.ac.uk`
  - Provides additional project management features beyond a shared repository.
  - The designated `gp12-XXX` Indefero project should (in most cases) be your central repository for code and documentation.
- Additionally, a Unix group has been created for each group on the School's Linux servers.

Indefero Features (1)

- Project Management:
  - Project description (using Markdown syntax).
  - Access rights (who can access and change what: defaults should work for most groups).
  - Detailed configuration of many of the other features (e.g. “tags”, notification e-mails).

Indefero Features (1)

- Project Management:
  - Project description (using Markdown syntax).
  - Access rights (who can access and change what: defaults should work for most groups).
  - Detailed configuration of many of the other features (e.g. “tags”, notification e-mails).
- Source code repository:
  - Version control using Subversion.
  - Not just for code, but can also be used for reports, design documents, and more.

Indefero Features (2)

- Documentation
  - Detailed project description (including aims, vision), agendas, meeting minutes, design documents, QA plans, . . .
  - Hyperlinks between documents
  - Markdown syntax

Indefero Features (2)

- Documentation
  - Detailed project description (including aims, vision), agendas, meeting minutes, design documents, QA plans, . . .
  - Hyperlinks between documents
  - Markdown syntax
- Issue tracking

Indefero Features (2)

- Documentation
  - Detailed project description (including aims, vision), agendas, meeting minutes, design documents, QA plans, . . .
  - Hyperlinks between documents
  - Markdown syntax
- Issue tracking
- Downloads
  - Various releases
  - Source, binaries for various platforms, . . .

Group Project Site Deliverable

- Group Project Site: first “deliverable” of G52GRP

Group Project Site Deliverable

- Group Project Site: first “deliverable” of G52GRP
- Due 2 November

Group Project Site Deliverable

- Group Project Site: first “deliverable” of G52GRP
- Due 2 November
- Designed to get your project site off the ground and ensure everyone understands the basics of Indefero and Subversion

Group Project Site Deliverable

- Group Project Site: first “deliverable” of G52GRP
- Due 2 November
- Designed to get your project site off the ground and ensure everyone understands the basics of Indefero and Subversion
- Nominate a Project Site Master ASAP!

Group Project Site Deliverable

- Group Project Site: first “deliverable” of G52GRP
- Due 2 November
- Designed to get your project site off the ground and ensure everyone understands the basics of Indefero and Subversion
- Nominate a Project Site Master ASAP!
- See the Group Project Handbook for further details.

Other Ways To Share & Coordinate (1)

What about other (possibly external) solutions?
For example:

Other Ways To Share & Coordinate (1)

What about other (possibly external) solutions?
For example:

- GitHub, Gitorious, Bitbucket, . . .
- Google Docs

Other Ways To Share & Coordinate (1)

What about other (possibly external) solutions?
For example:

- GitHub, Gitorious, Bitbucket, . . .
- Google Docs
- Dropbox

Other Ways To Share & Coordinate (1)

What about other (possibly external) solutions?
For example:

- GitHub, Gitorious, Bitbucket, . . .
- Google Docs
- Dropbox
- Facebook

Other Ways To Share & Coordinate (1)

What about other (possibly external) solutions?
For example:

- GitHub, Gitorious, Bitbucket, . . .
- Google Docs
- Dropbox
- Facebook
- Google+
- . . .

Other Ways To Share & Coordinate (2)

OK, and may even be needed, but:

Other Ways To Share & Coordinate (2)

OK, and may even be needed, but:

- Group Project Site deliverable must still be done as specified.

Other Ways To Share & Coordinate (2)

OK, and may even be needed, but:

- Group Project Site deliverable must still be done as specified.
- All documents and code ***must*** be backed up on School servers!

Other Ways To Share & Coordinate (2)

OK, and may even be needed, but:

- Group Project Site deliverable must still be done as specified.
- All documents and code ***must*** be backed up on School servers!
  Temporary unavailability of external hosting, or external host going out of business (or your own, private machines dying, getting stolen, . . .), are ***not*** valid extenuating circumstances.

Other Ways To Share & Coordinate (2)

OK, and may even be needed, but:

- Group Project Site deliverable must still be done as specified.
- All documents and code ***must*** be backed up on School servers!
  Temporary unavailability of external hosting, or external host going out of business (or your own, private machines dying, getting stolen, . . .), are ***not*** valid extenuating circumstances.
- You may have to copy certain data across to the project site for submission purposes.

Other Ways To Share & Coordinate (3)

Additionaly:

Other Ways To Share & Coordinate (3)

Additionaly:

- Social networking sites like Facebook were designed for . . .

Other Ways To Share & Coordinate (3)

Additionaly:

- Social networking sites like Facebook were designed for . . . social networking, not software development.

Other Ways To Share & Coordinate (3)

Additionaly:

- Social networking sites like Facebook were designed for . . . social networking, not software development.
- Consequently, lack key features like version control and issue tracking.

Other Ways To Share & Coordinate (3)

Additionaly:

- Social networking sites like Facebook were designed for . . . social networking, not software development.
- Consequently, lack key features like version control and issue tracking.
- Grops that ***did*** use Facebook reported that the social aspects were a constant source of distraction.

Other Ways To Share & Coordinate (4)

Why Use Version Control Systems? (1)

OK, doc and code shared. Problem solved? No . . .

Why Use Version Control Systems? (1)

OK, doc and code shared. Problem solved? No . . .

- If a team of people involved, how to coordinate the work on the shared source code and documentation?

Why Use Version Control Systems? (1)

OK, doc and code shared. Problem solved? No . . .

- If a team of people involved, how to coordinate the work on the shared source code and documentation?
- As the source and documentation evolves, how to
  - keep track of changes
  - keep track of consistent configurations
  - insulate against “work in progress”
  - . . .

Why Use Version Control Systems? (2)

Version control systems

Why Use Version Control Systems? (2)

Version control systems

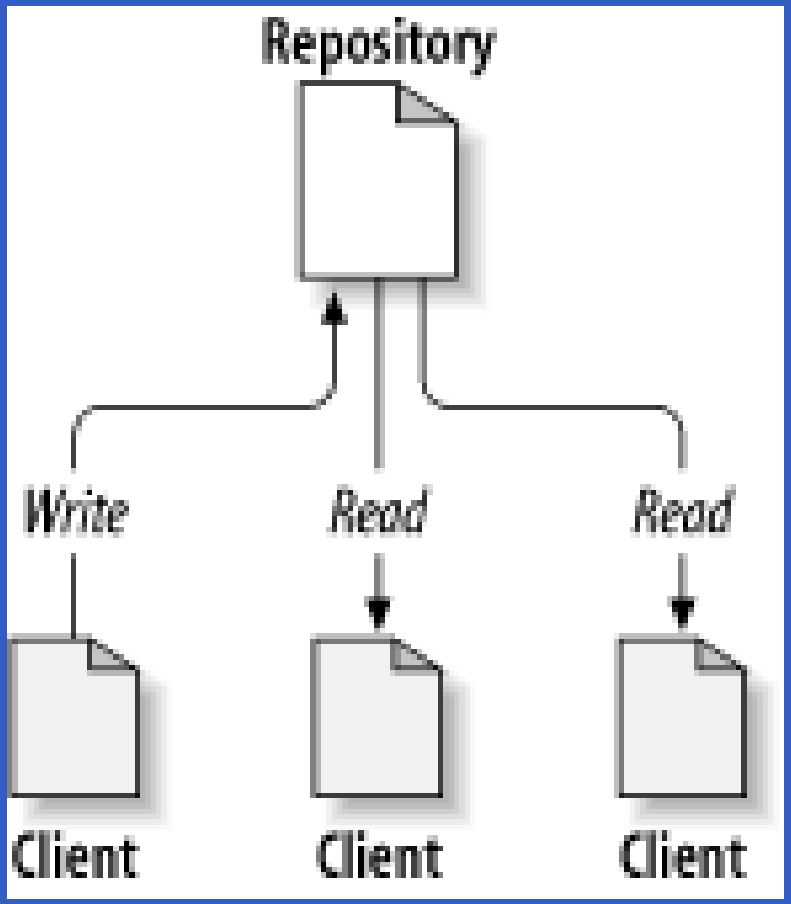
- originally addressed the second problem (hence the name)

Why Use Version Control Systems? (2)

Version control systems

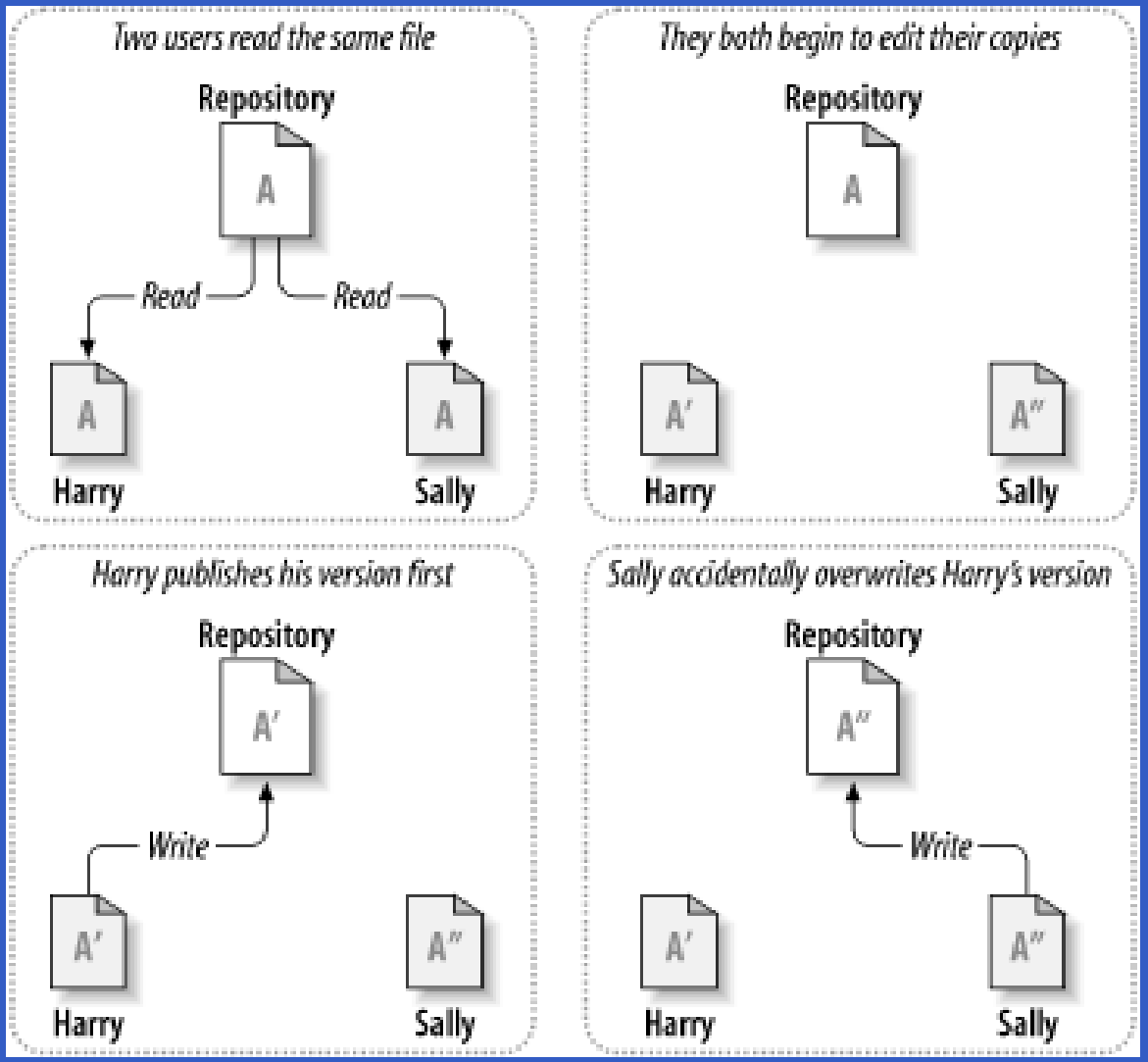
- originally addressed the second problem (hence the name)
- but modern ones also provide very sophisticated support for
  - teams of programmers working on shared source and documentation
  - distributed teams of programmers (over the Internet)

Basic Model

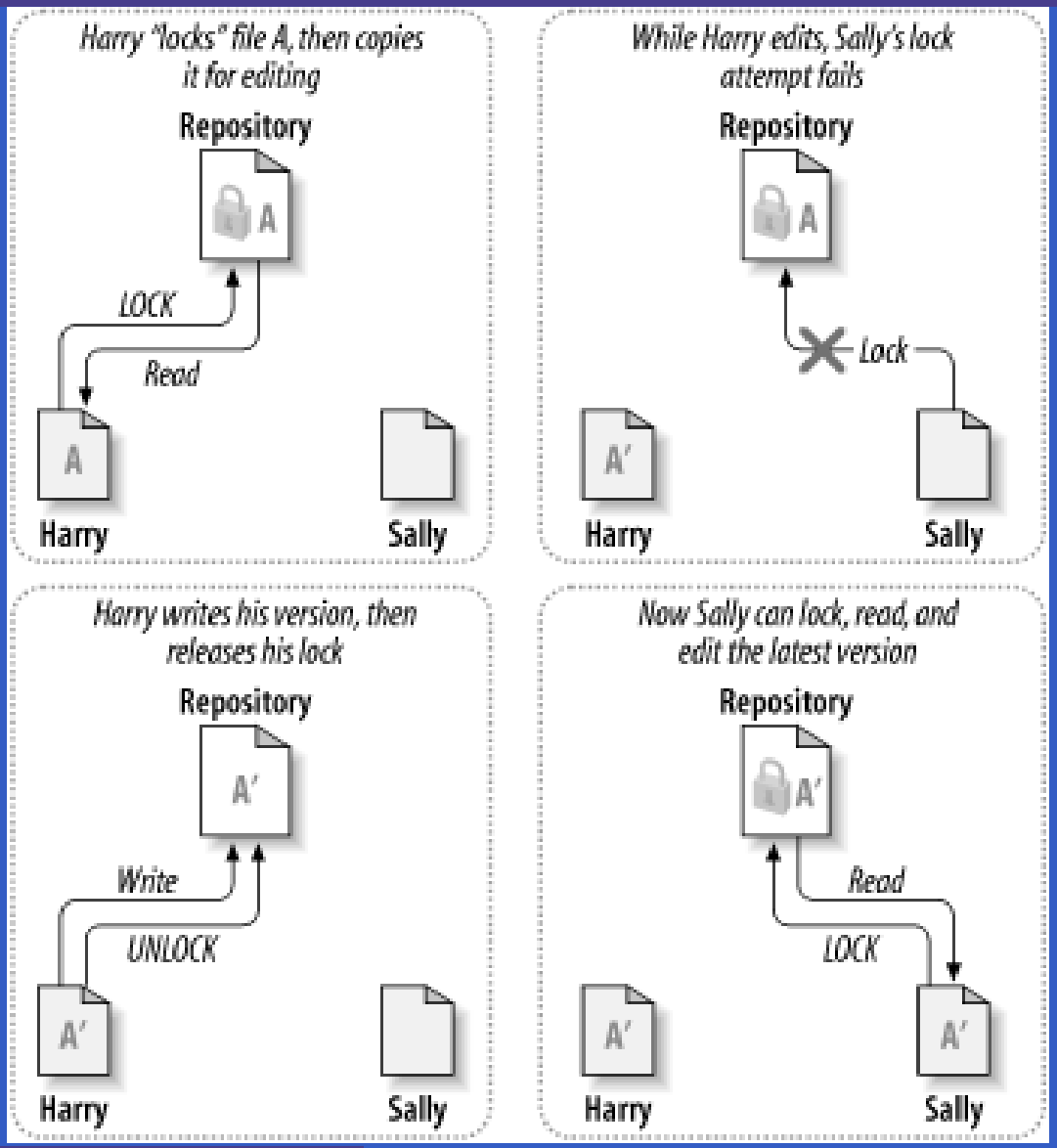


(Pictures from Collins-Sussman, Fitzpatrick, Pilato: Version Control with Subversion.)

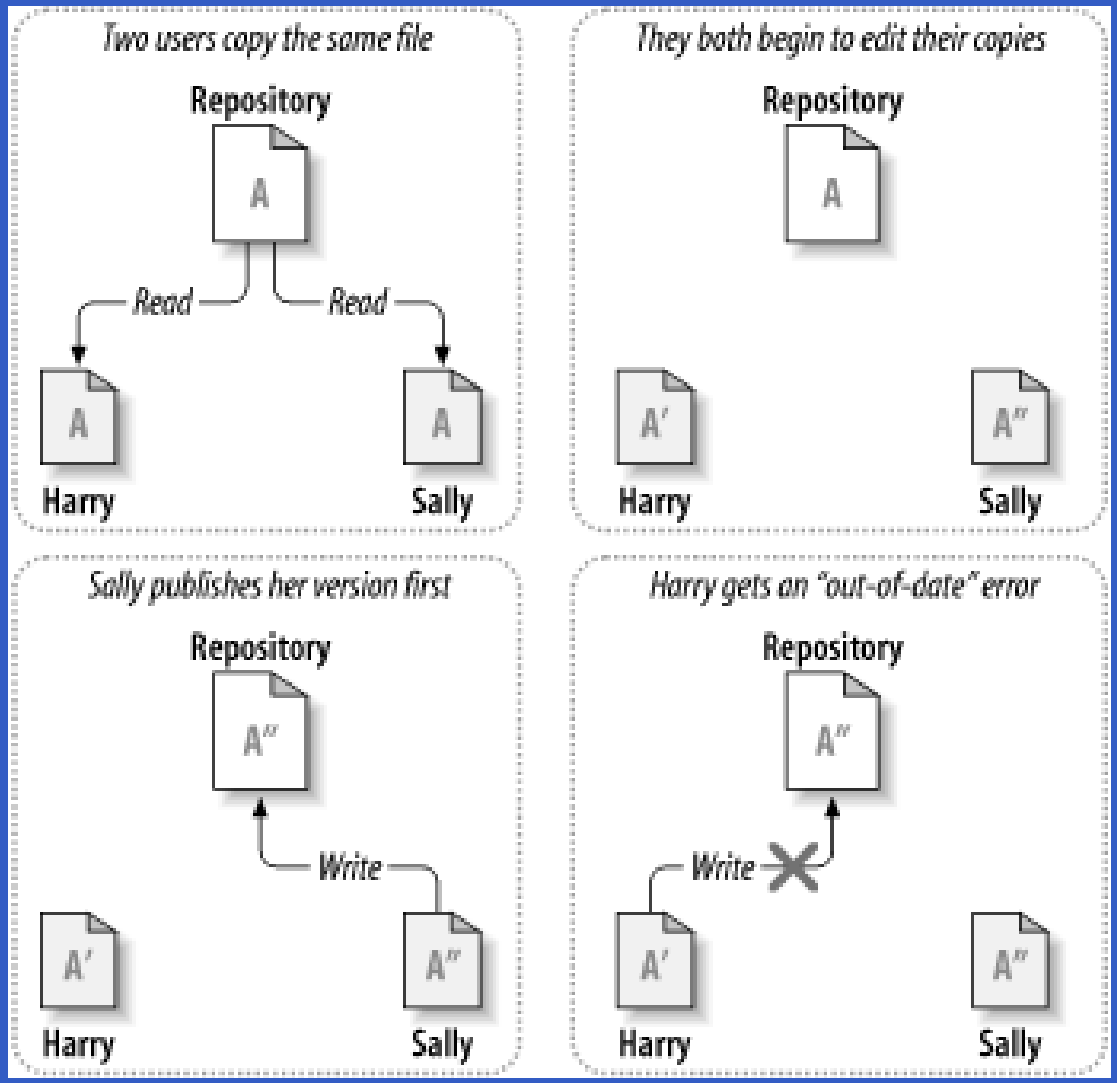
The Problem to Avoid



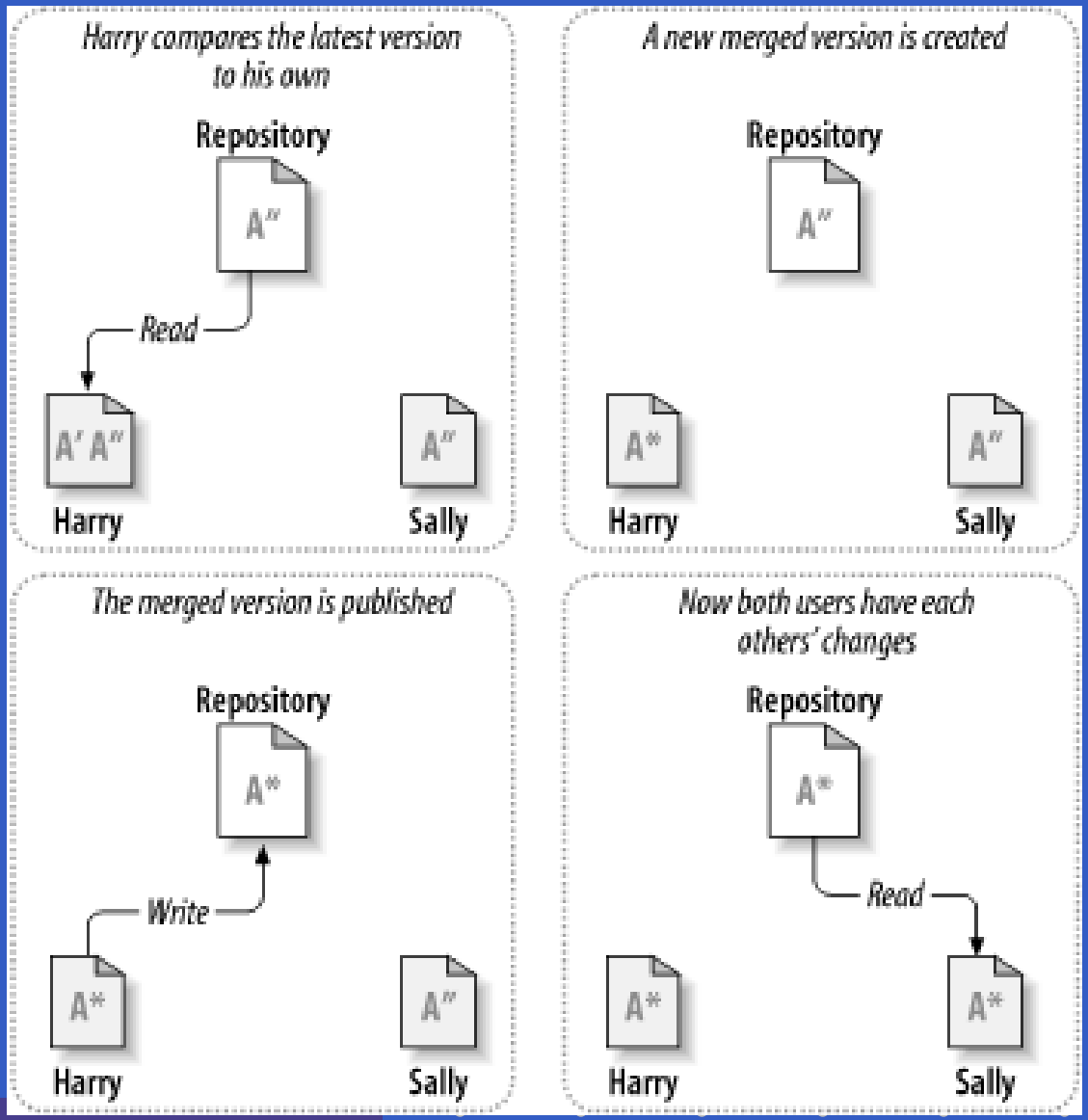
The Lock, Modify, Unlock Model



The Copy, Modify, Merge Model (1)



The Copy, Modify, Merge Model (2)



Version Control

A version control system provides

Version Control

A version control system provides

- a “time travel” facility: arbitrary earlier versions of the repository can be retrieved

Version Control

A version control system provides

- a “time travel” facility: arbitrary earlier versions of the repository can be retrieved
- facilities for supporting parallel, non-interfering development, e.g. through what ***looks*** like separate copies, . . .

Version Control

A version control system provides

- a “time travel” facility: arbitrary earlier versions of the repository can be retrieved
- facilities for supporting parallel, non-interfering development, e.g. through what ***looks*** like separate copies, . . .
- . . . while ***maximizing sharing*** and ***facilitating reintegration*** of lines of development.

What Is Subversion? (1)

- Free, open-source version control system.

What Is Subversion? (1)

- Free, open-source version control system.
- Manages files ***and directories***, allowing older versions of (a part of) a file hierarchy to be retrieved at any point in time, pinpointing changes, keeping track of metadata such as logs for information ***about*** changes, etc.

What Is Subversion? (1)

- Free, open-source version control system.
- Manages files ***and directories***, allowing older versions of (a part of) a file hierarchy to be retrieved at any point in time, pinpointing changes, keeping track of metadata such as logs for information ***about*** changes, etc.
- Handles ***both*** text and binary data (like Word documents, images)

What Is Subversion? (1)

- Free, open-source version control system.
- Manages files ***and directories***, allowing older versions of (a part of) a file hierarchy to be retrieved at any point in time, pinpointing changes, keeping track of metadata such as logs for information ***about*** changes, etc.
- Handles ***both*** text and binary data (like Word documents, images)
- Supports concurrent development (the Copy, Modify, Merge model), both locally and remotely (over a network).

What Is Subversion? (2)

- Also does support locking (mainly intended for binary data that cannot easily be merged: images, Word documents, other application-specific binary data, . . .)

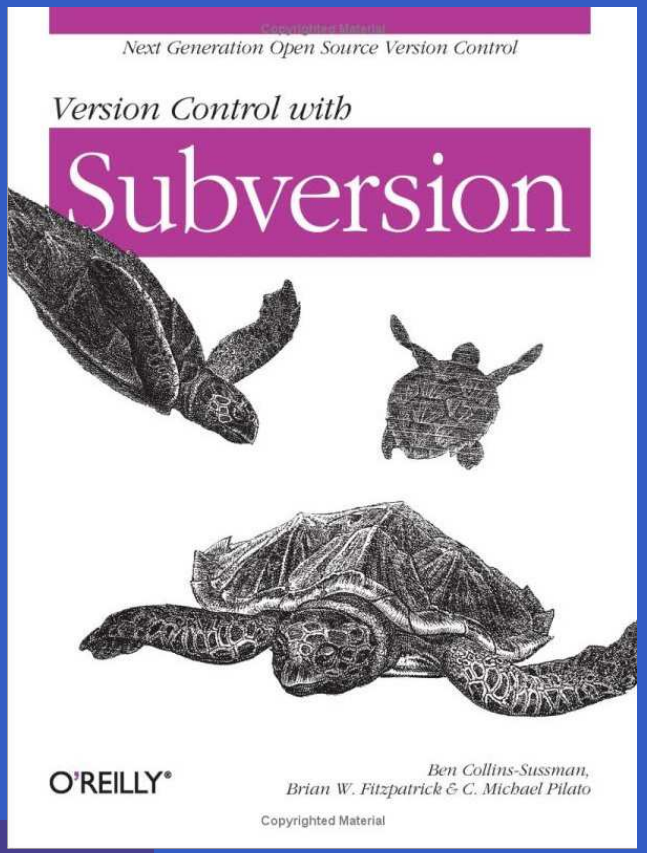
What Is Subversion? (2)

- Also does support locking (mainly intended for binary data that cannot easily be merged: images, Word documents, other application-specific binary data, . . .)
- Other helpful features like
  - file portability (e.g. transparent conversion between CR/LF and LF line ending conventions)
  - automation through hooks (e.g. sending e-mail after changes committed)

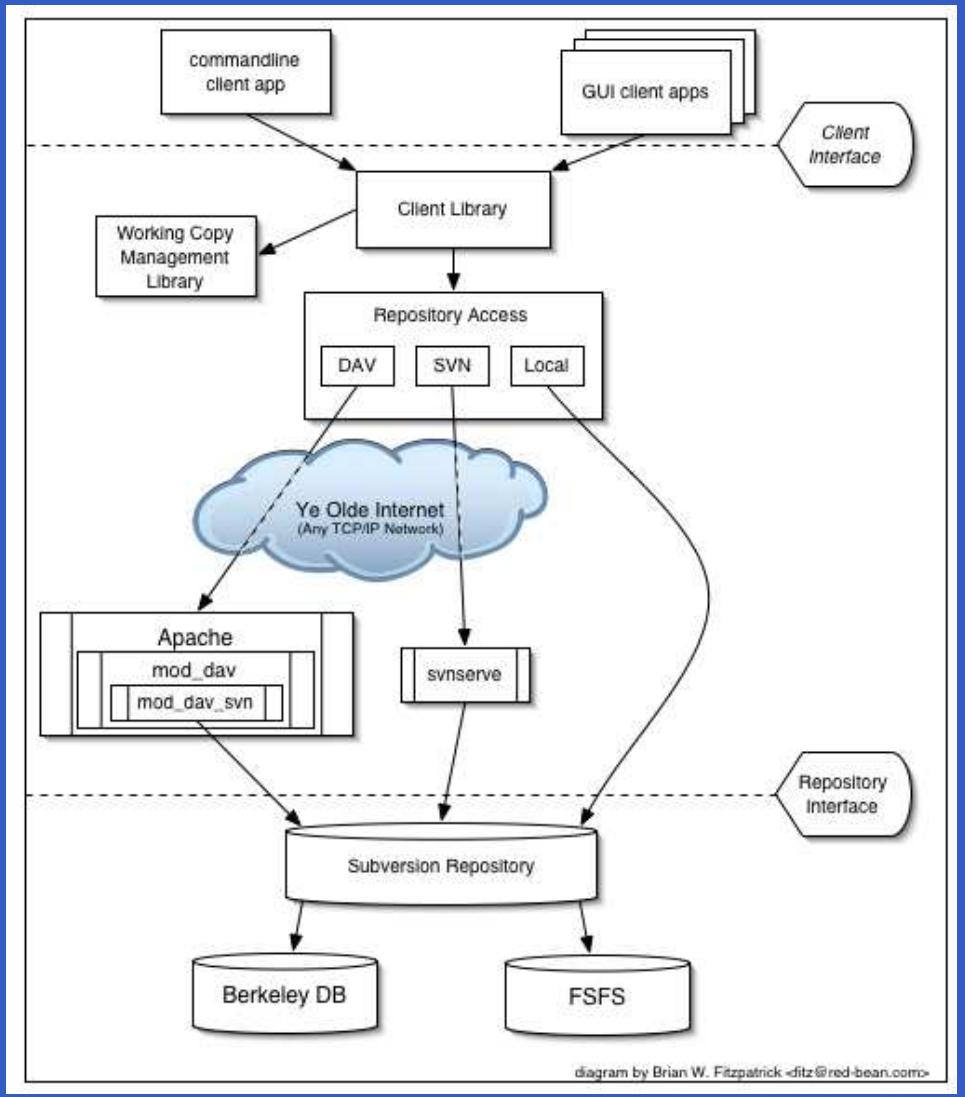
What Is Subversion? (3)

Main reference (freely available on-line):

Collins-Sussman, Fitzpatrick, Pilato:
Version Control with Subversion



Architecture of Subversion



Getting Started

- Use either the command-line interface, or a Subversion client like Tortoise (Windows, shell extension), or from within IDEs like NetBeans, Eclipse (with Subclipse plugin).

Getting Started

- Use either the command-line interface, or a Subversion client like Tortoise (Windows, shell extension), or from within IDEs like NetBeans, Eclipse (with Subclipse plugin).
- Command-line interface used in the following: principles remain the same regardless of access mode.

Getting Started

- Use either the command-line interface, or a Subversion client like Tortoise (Windows, shell extension), or from within IDEs like NetBeans, Eclipse (with Subclipse plugin).
- Command-line interface used in the following: principles remain the same regardless of access mode.
- Check out a copy of the repository, e.g.:

```
svn checkout \
https://code.cs.nott.ac.uk/svn/gp12-nhn
```

Getting Started

- Use either the command-line interface, or a Subversion client like Tortoise (Windows, shell extension), or from within IDEs like NetBeans, Eclipse (with Subclipse plugin).
- Command-line interface used in the following: principles remain the same regardless of access mode.
- Check out a copy of the repository, e.g.:

```
svn checkout \
https://code.cs.nott.ac.uk/svn/gp12-nhn
```

- Change directory: `cd gp12-nhn`.

Accessing the Repository

The command-line Subversion client is called `svn`. It has many subcommands, e.g.:

- `svn list`
- `svn add`
- `svn copy`
- `svn commit`

Accessing the Repository

The command-line Subversion client is called `svn`. It has many subcommands, e.g.:

- `svn list`
- `svn add`
- `svn copy`
- `svn commit`

It is always possible to get help, including on specific subcommands:

- `svn help`
- `svn help copy`

Initial Repository Structure (1)

Let’s pupulate the repository with some initial structure. The Subversion book recommends three main directories for each project:

Initial Repository Structure (1)

Let's pupulate the repository with some initial structure. The Subversion book recommends three main directories for each project:

- ***trunk***: for the main development

Initial Repository Structure (1)

Let's pupulate the repository with some initial structure. The Subversion book recommends three main directories for each project:

- ***trunk***: for the main development
- ***branches***: for branched-off developments (may later be merged back into main branch)

Initial Repository Structure (1)

Let's pupulate the repository with some initial structure. The Subversion book recommends three main directories for each project:

- ***trunk***: for the main development
- ***branches***: for branched-off developments (may later be merged back into main branch)
- ***tags***: named "snap shots" of the development; often a "release".

Initial Repository Structure (1)

Let's pupulate the repository with some initial structure. The Subversion book recommends three main directories for each project:

- ***trunk***: for the main development
- ***branches***: for branched-off developments (may later be merged back into main branch)
- ***tags***: named "snap shots" of the development; often a "release".

Branches and Tags are created by copying: `svn copy`

Initial Repository Structure (2)

Under trunk we might want to have subdirectories for subprojects, e.g.:

- ***src***: for source code
- ***doc***: for documentation

Initial Repository Structure (2)

Under trunk we might want to have subdirectories for subprojects, e.g.:

- ***src***: for source code
- ***doc***: for documentation

Subversion does not make any particular assumptions: the directory structure can be what you like.

Initial Repository Structure (2)

Under trunk we might want to have subdirectories for subprojects, e.g.:

- ***src***: for source code
- ***doc***: for documentation

Subversion does not make any particular assumptions: the directory structure can be what you like.

AND! It is easy to change the structure later by simply moving around files and directories.

Initial Repository Structure (3)

Let's create this directory structure:

```
marian$ mkdir trunk
marian$ mkdir branches
marian$ mkdir tags
marian$ mkdir trunk/src
marian$ mkdir trunk/doc
```

Initial Repository Structure (4)

Let's tell subversion all these directories have been added:

```
marian$ svn add trunk
marian$ svn add branches
marian$ svn add tags
marian$ svn add trunk/src
marian$ svn add trunk/doc
```

Then, let's commit to the central repository:

```
marian$ svn commit
```

Checking Out a Working Copy

Now ***other*** people can check out a working copy of the relevant part of the repository:

```
svn checkout \
https://code.cs.nott.ac.uk/svn/gp12-nhn/trunk
A    trunk/doc
A    trunk/src
Checked out revision 1.
```

Adding a File (1)

Let's add a document:

```
isis-19% cd trunk/doc
isis-20% ooffice
isis-21% ls -l
-rw-r--r-- 1 henrik henrik 8192 Oct 23 01:45
    design.doc
```

The location of the repository is stored with the working copy, so Subversion commands can now be given without giving the repository URL:

```
isis-49% svn status
?      design.doc
```

Adding a File (2)

The status “?” indicates something which is unknown to Subversion. We need to tell Subversion about it:

```
isis-50% svn add design.doc
A  (bin)  design.doc
```

Important! The new document is now added to the ***local working copy***. But it (and other changes) will not be propagated to the central repository until we explicitly perform a ***commit***.

```
isis-51% svn status
A       design.doc
```

Adding a File (3)

Subversion correctly determined that our document is a ***binary*** file. (Could have been configured otherwise.)

```
isis-52% svn proplist design.doc
Properties on 'design.doc':
  svn:mime-type
isis-53% svn propget svn:mime-type \
  design.doc
application/octet-stream
```

Adding a Directory

It is equally easy to add directories:

```
isis-54% mkdir Meeting-2012-10-23
isis-55% svn add Meeting-2012-10-23
A         Meeting-2012-10-23
isis-56% emacs Meeting-2012-10-23/ideas.txt
isis-57% svn add Meeting-2012-10-23/ideas.txt
A         Meeting-2012-10-23/ideas.txt
isis-58% svn status
A       design.doc
A       Meeting-2012-10-23
A       Meeting-2012-10-23/ideas.txt
```

Or use `svn mkdir`.

Committing

Time to propagate the changes to the central repository:

```
isis-59% svn commit \
   -m "System design and ideas from meeting 23 Oct"
Adding          doc/design.doc
Adding          doc/Meeting-2012-10-23
Adding          doc/Meeting-2012-10-23/ideas.txt
Transmitting file data .
Committed revision 2.
```

Making Changes

Let's assume a few typos in the design are fixed:

```
isis-65% svn status
M       design.doc
isis-66% svn commit -m "Fixed typos"
Sending        doc/design.doc
Transmitting file data .
Committed revision 3.
```

Propagating Changes (1)

Assume someone else makes changes and commits. We can check the status against the repository and get log entries

```
isis-70% svn status -u
       *         2   design.doc
Status against revision:      3
isis-70% svn log -r 3 design.doc
------------------------------------------------------------------------
r3 | nhn | 2012-10-24 09:51:00 +0100 (Wed, 24 Oct 201

Fixed typos
------------------------------------------------------------------------
```

Propagating Changes (2)

Let’s bring our working copy up-to-date:

```
isis-71% svn update
U    design.doc
Updated to revision 3.
```

In general, it is good practice to to bring everything up-to-date before starting to make any changes. Minimizes the risk of conflicts.

Conflicts (1)

What if someone else has commited changes before I commit?

Conflict! Text files can, however, be merged.

```
isis-92% svn commit
Sending        Meeting-2012-10-23/ideas.txt
Transmitting file data .svn: Commit failed (details f
isis-83% svn update
C    ideas.txt
Updated to revision 7.
```

Conflicts (2)

The differences are marked in the file. Edit as necessary. Then:

```
isis-93% svn resolved idead.txt
Resolved conflicted state of 'ideas.txt'
isis-94% svn commit
```

(Newer versions has a more sophisticated `svn resolve` command.)

Other useful Subversion commands

Some other commands:

- `svn delete`
- `svn copy`
- `svn diff`
- `svn revert`
- `svn lock`
- `svn unlock`

Be sure to read at least the introductory chapters of the Subversion book (very accessible) and do use `svn help`!