Version Control Systems

SENG 403

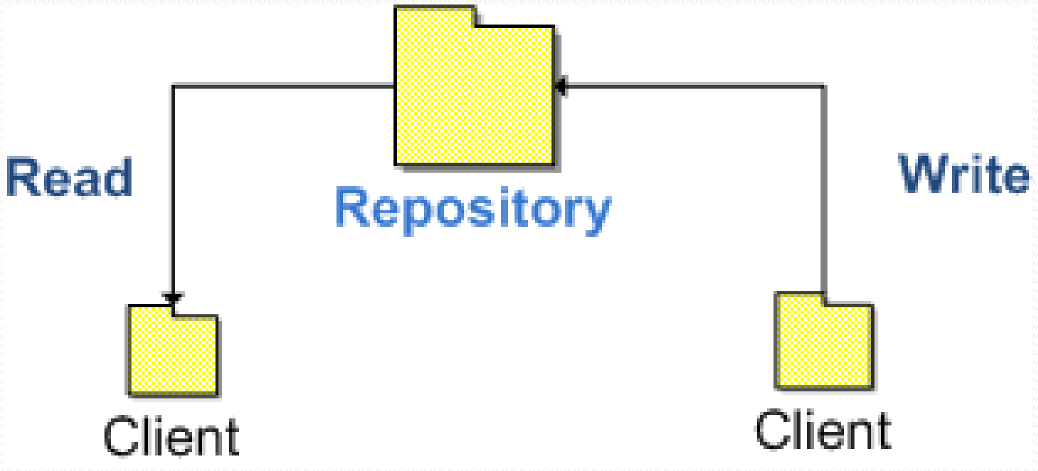
Tutorial 1

Agenda

- Version Control Basics
- Subversion
- Basic actions in Subversion
- Some examples

Version Control Basics

- A version (or revision) control system is able to track incremental versions of files and directories over time
- The core of a version control system is a **repository**



Version Control Server

- The version control server does not work just like a typical file server.
- The repository remembers **each version of files**, as they are changed in the repository.

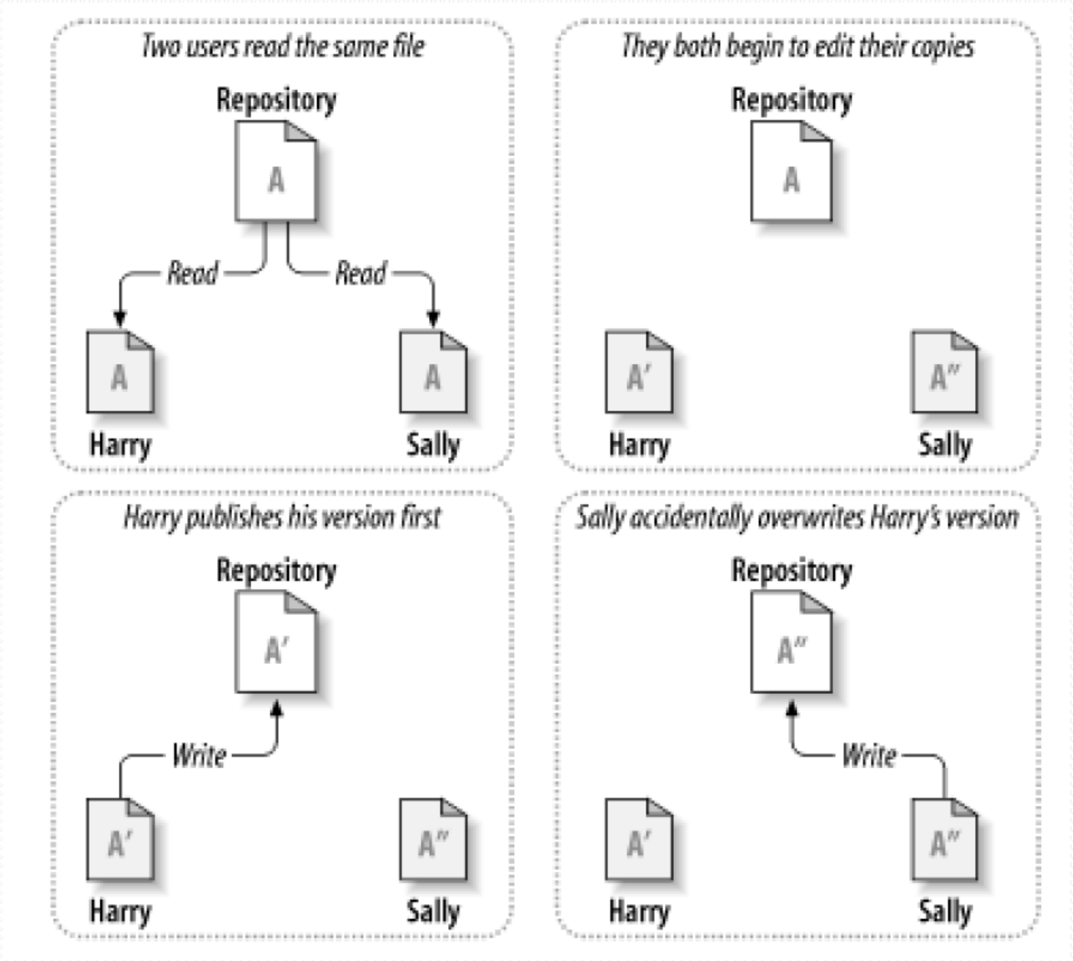
- When a client reads from the repository, it normally gets the latest version. But it can request the previous states of the file system.
- **Working copy**: A local copy of a particular version of a VCS-managed data.

The Main Reasons to Have a VCS

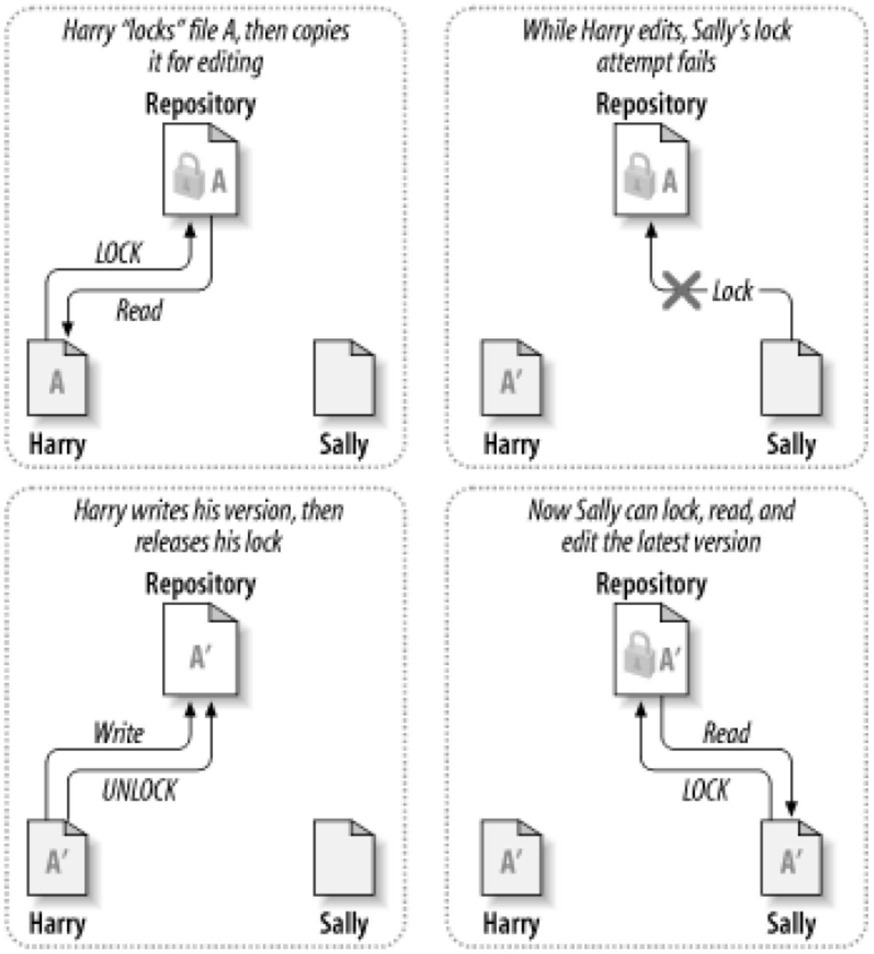
- To track the various versions of digital information over time
- To enable collaborating editing and sharing of data

- We want sharing but we want to avoid accidentally stepping on each other's toes.

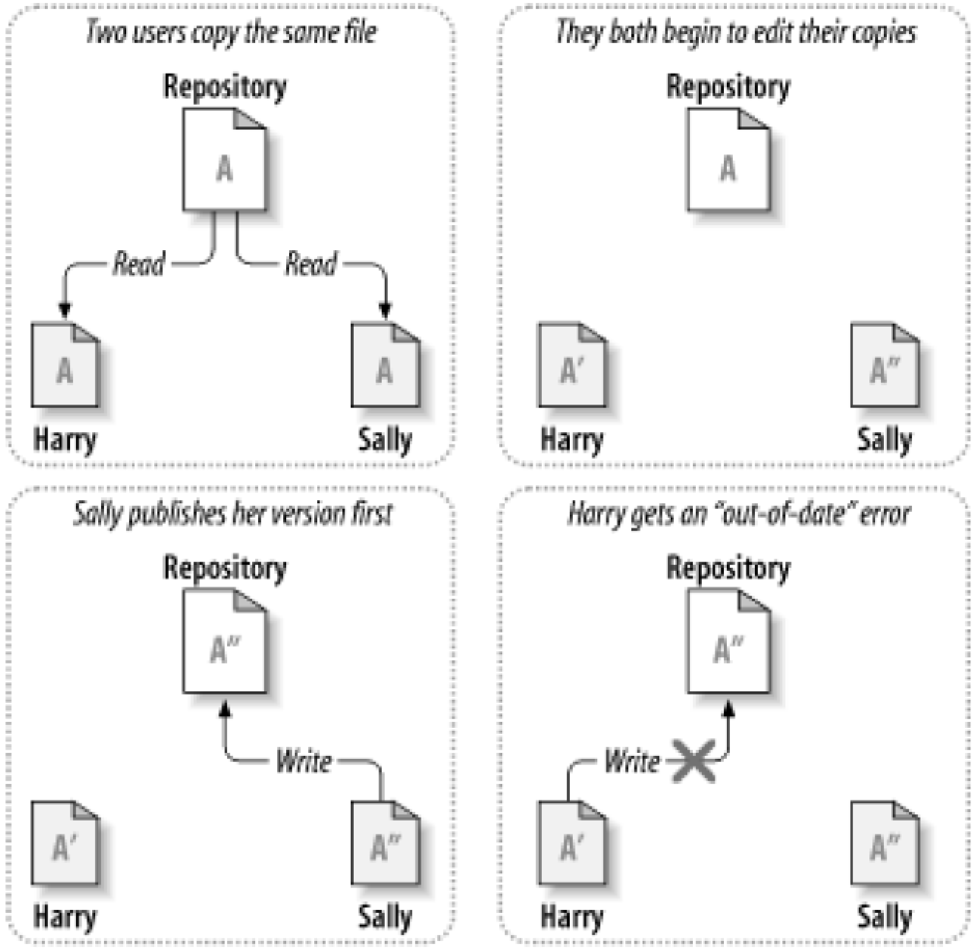
A Typical Problem


Two users read the same file
Repository
A
Read
Read
A
A
Harry
Sally
They both begin to edit their copies
Repository
A
A′
A″
Harry
Sally
Harry publishes his version first
Repository
A′
Write
A′
A″
Harry
Sally
Sally accidentally overwrites Harry's version
Repository
A″
Write
A′
A″
Harry
Sally

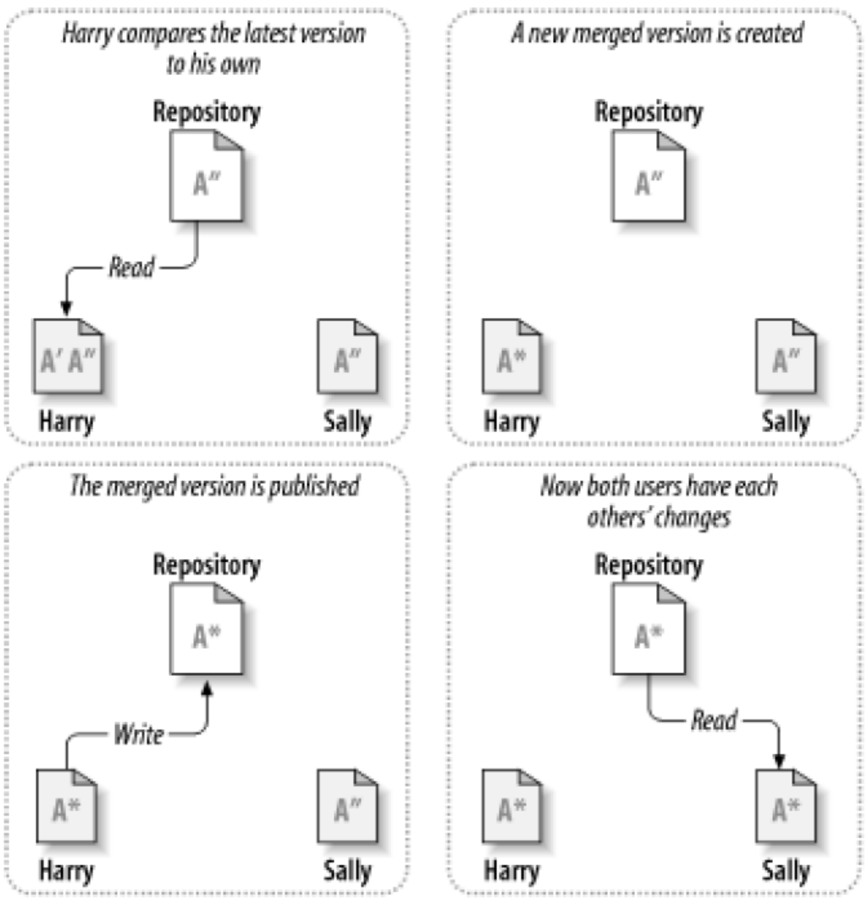
Solution 1:Lock-Modify-Unlock


Harry "locks" file A, then copies it for editing
Repository
A
LOCK
Read
A
Harry
Sally
While Harry edits, Sally's lock attempt fails
Repository
A
Lock
A'
Harry
Sally
Harry writes his version, then releases his lock
Repository
A'
Write
UNLOCK
A'
Harry
Sally
Now Sally can lock, read, and edit the latest version
Repository
A'
Read
LOCK
A'
A'
Harry
Sally

Solution 2: Copy-Modify-Merge

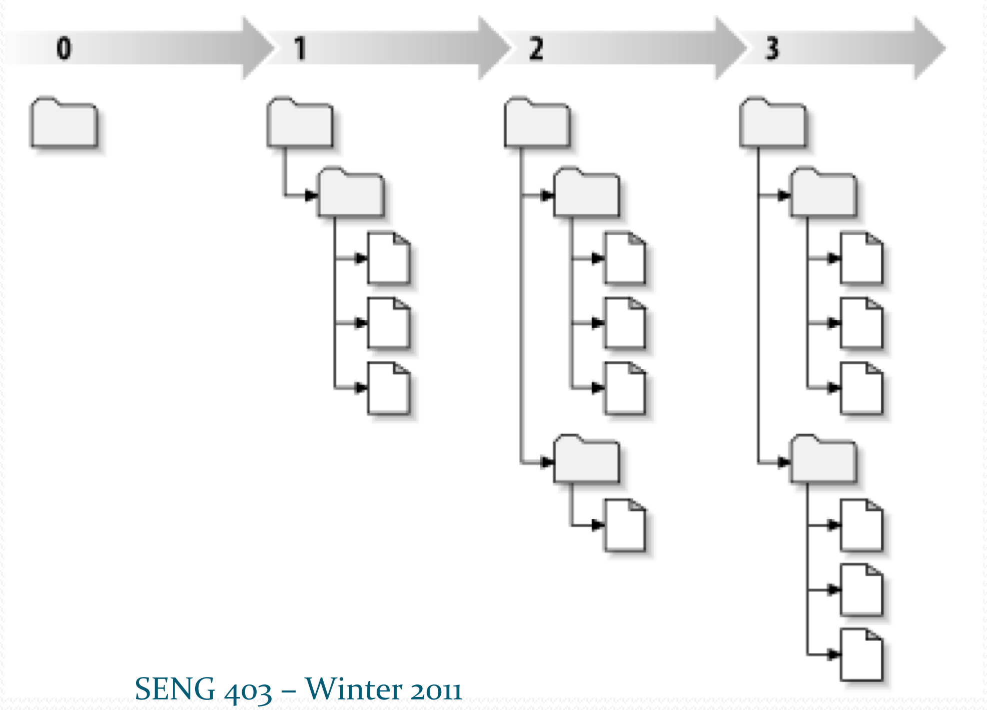

Two users copy the same file
Repository
A
Read
Read
A
A
Harry
Sally
They both begin to edit their copies
Repository
A
A′
A″
Harry
Sally
Sally publishes her version first
Repository
A″
Write
A′
A″
Harry
Sally
Harry gets an "out-of-date" error
Repository
A″
Write
A′
A″
Harry
Sally

Solution 2 (continued)


Harry compares the latest version to his own
Repository
A″
Read
A′ A″
Harry
A″
Sally
A new merged version is created
Repository
A″
A*
Harry
A″
Sally
The merged version is published
Repository
A*
Write
A*
Harry
A″
Sally
Now both users have each others' changes
Repository
A*
Read
A*
Harry
A*
Sally

Subversion Repository

- Subversion clients **commit** in an atomic fashion.
- After each successful commit, a new state of filesystem tree will be created. It is called a **revision**.



Subversion Working Copies

- A working copy is a directory on your local machine, containing a collection of files.
- When you finish making changes, you tell the Subversion client to publish it, so that other people can see the changes.
- To manage merges and conflicts, Subversion keeps track of the revision that your working copy is based on and a timestamp recording when the local copy was last updated by the repository.

Some VCS Terms

- Repository (repo)
- Trunk/Main
- Add
- Revision
- Head
- Check out
- Check in
- Changelog/History
- Update/Sync
- Revert
- Branch
- Merge
- Conflict
- Resolve
- Diff/Change/Delta

SVN Server

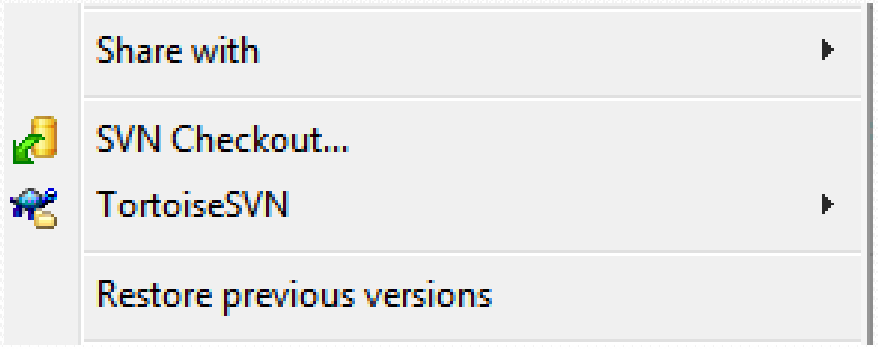
- We need a SVN server to put all the revisions of the project(s) on it.
- One option is to use free source code repository web sites, like SourceForge and CodePlex.
- In this tutorial we will use CodePlex.
- CodePlex supports SVN as well as MS TFS.
- You can define a project and use the VCS Server of the CodePlex.
- Warning: Unless you make the project public within a month, it will be removed after 30 days.

Addressing the Repository

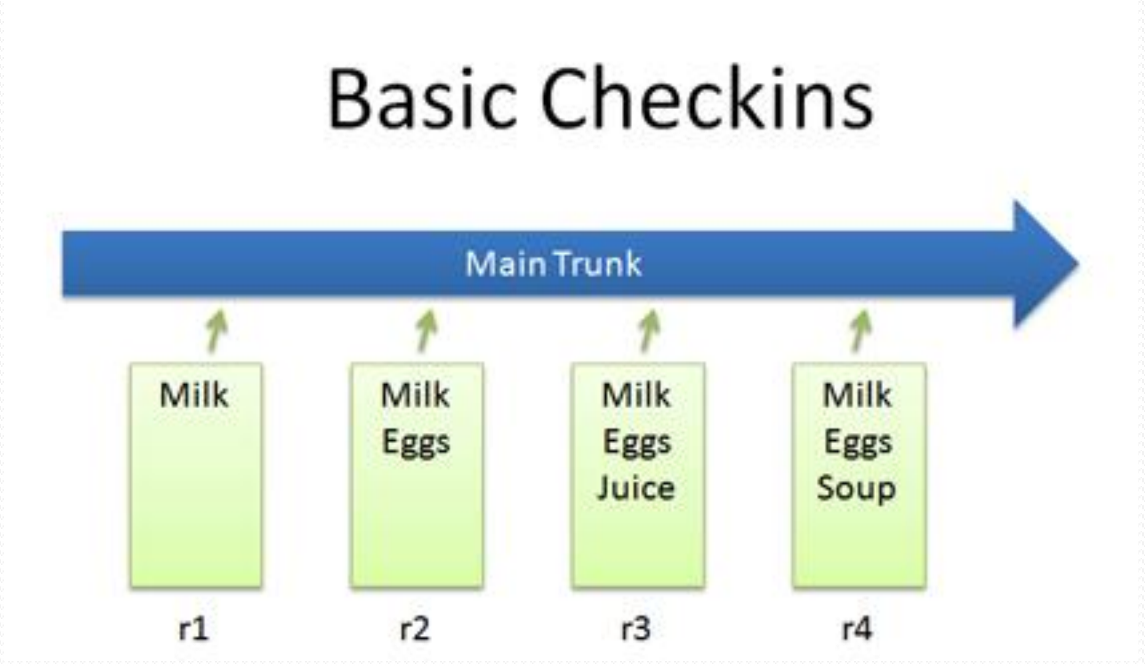
- Subversion client programs use URLs to identify versioned files and directories in Subversion repositories.
- For the most part, these URLs use the standard syntax, allowing for server names and port numbers to be specified as part of the URL:
- `https://smntestproject.svn.codeplex.com/svn/trunk/list.txt`

SVN Client

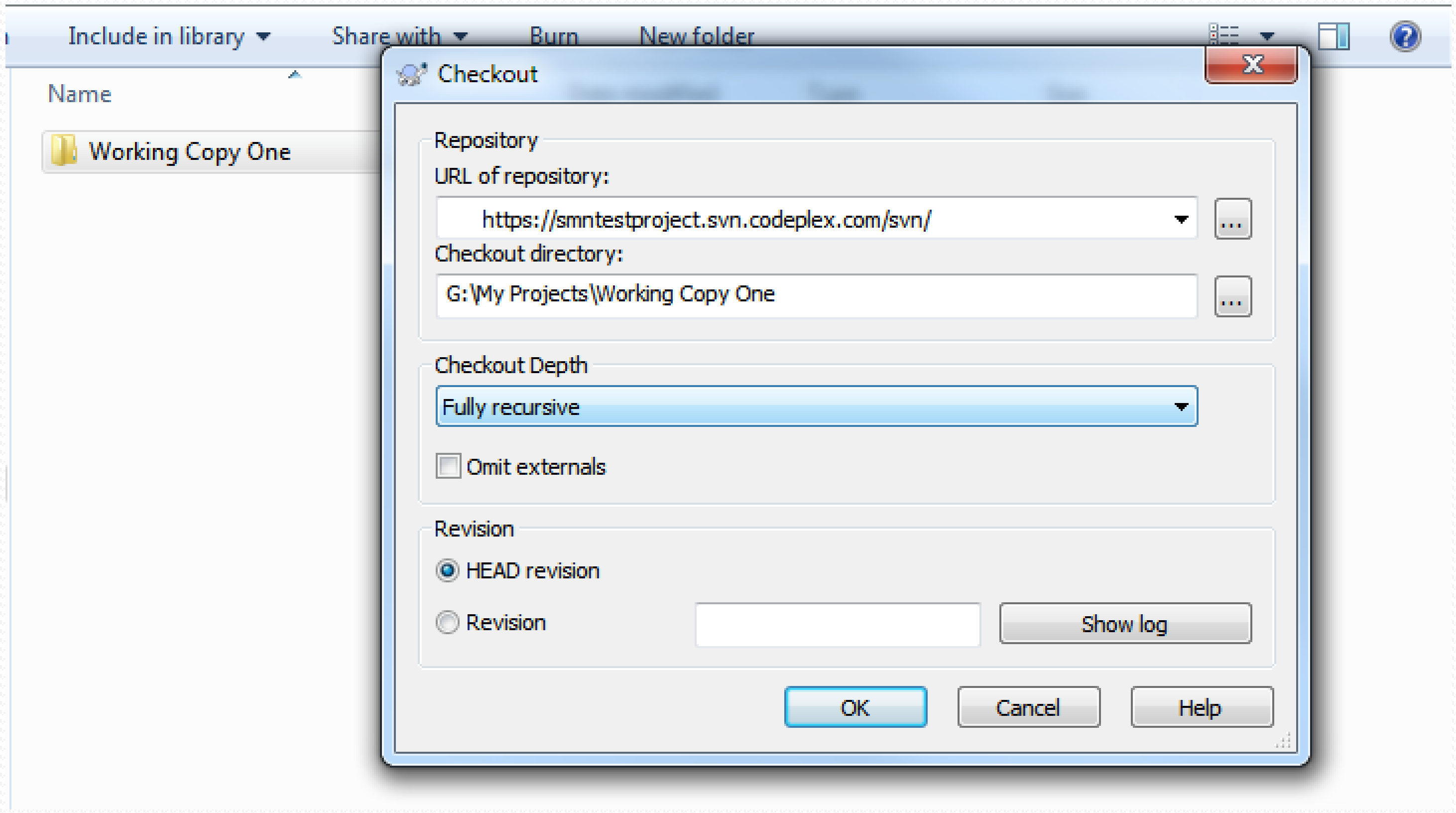
- On Linux/Unix based operating systems, there is a command line SVN Client, called `svn`.
- On Windows, you can install GUI clients.
- TortoiseSVN is a free SVN client.
- When you install it, it will be integrated into Windows Explorer pop-up menu.



Checkins


Basic Checkins
Main Trunk
Milk
Milk
Eggs
Milk
Eggs
Juice
Milk
Eggs
Soup
r1
r2
r3
r4

Example – Creating a Working Copy



Repository

URL of repository: https://smntestproject.svn.codeplex.com/svn/

Checkout directory: G:\My Projects\Working Copy One

Checkout Depth: Fully recursive

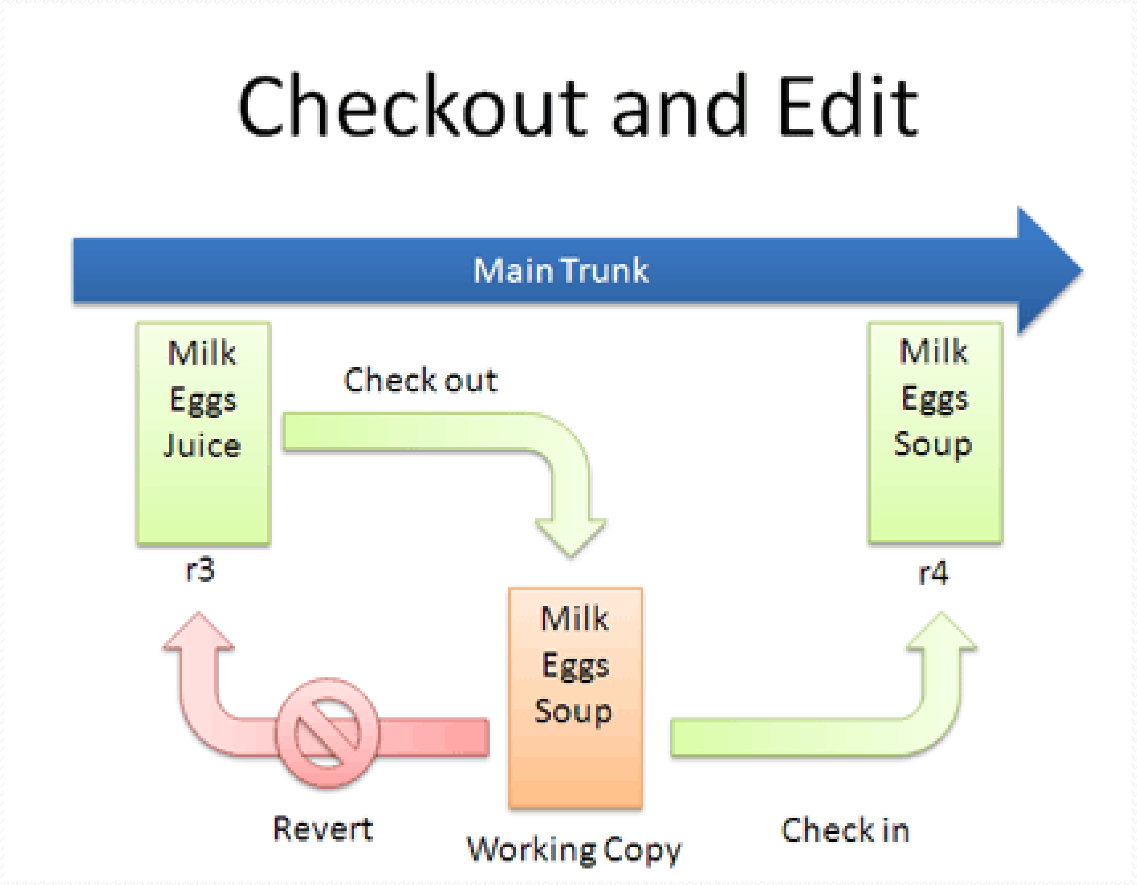
☐ Omit externals

Revision

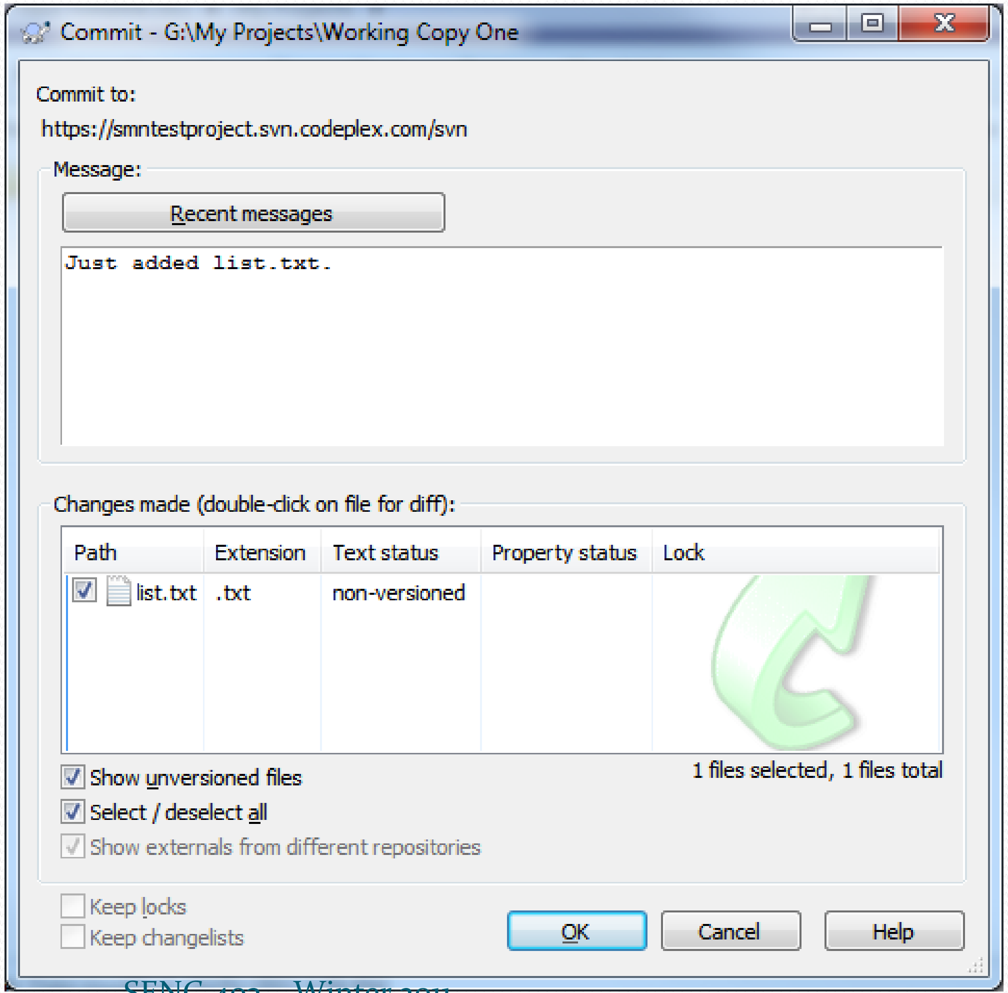
● HEAD revision

○ Revision

Checkouts and Editing



Example - Commit



Commit - G:\My Projects\Working Copy One

Commit to:
https://smntestproject.svn.codeplex.com/svn

Message:

```
Just added list.txt.
```

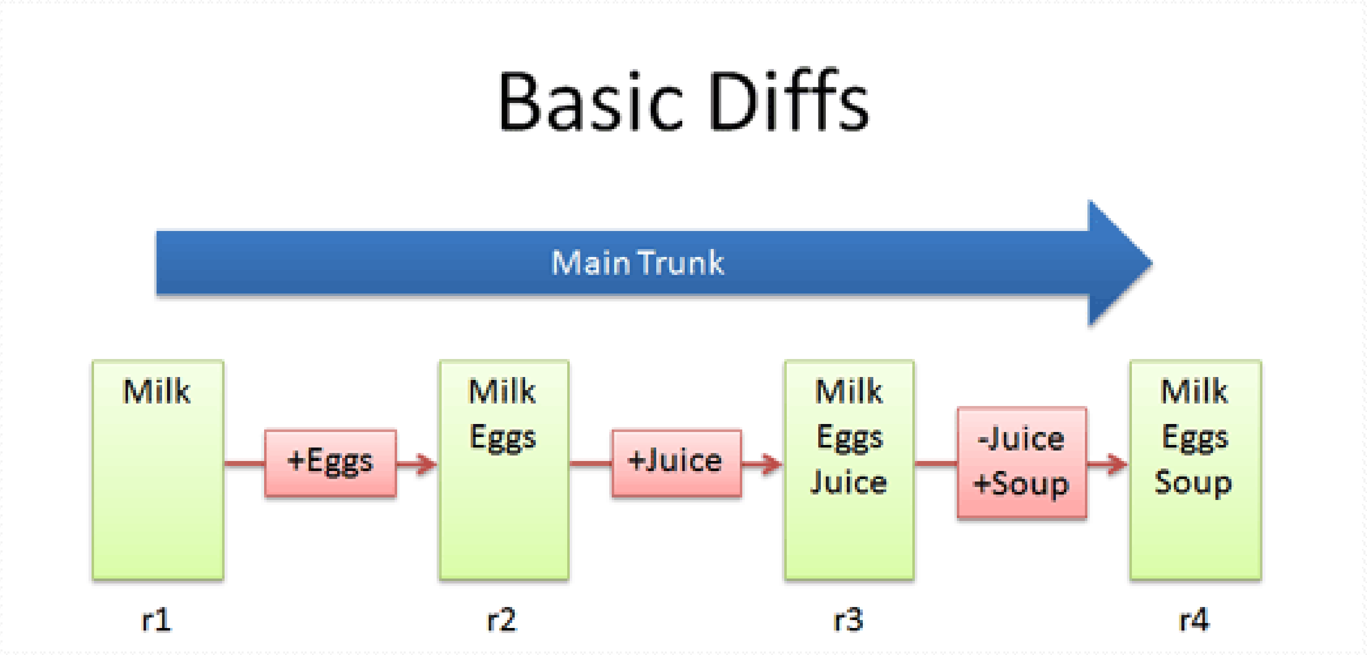
Changes made (double-click on file for diff):

Path	Extension	Text status	Property status	Lock
✓ list.txt	.txt	non-versioned		

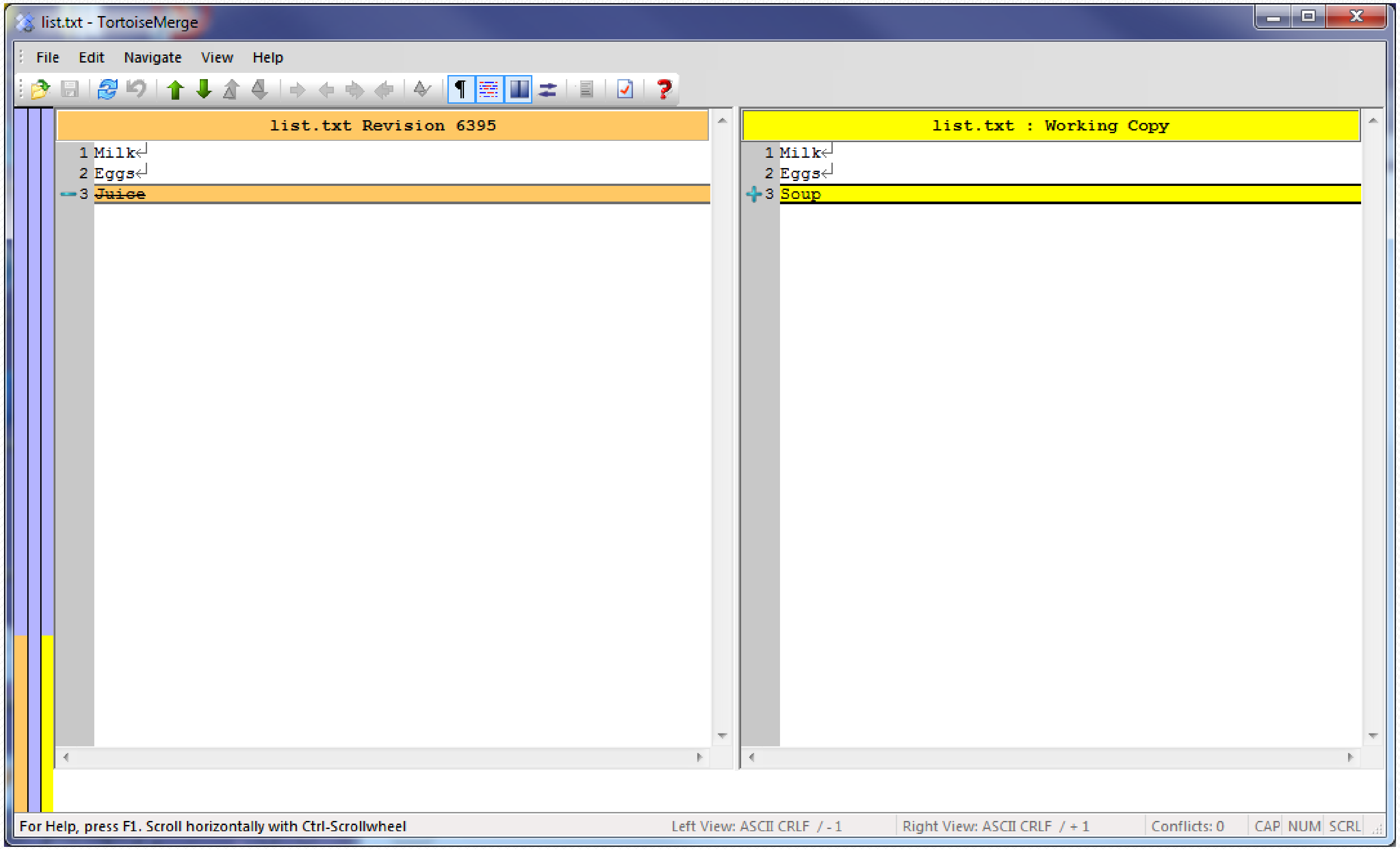
1 files selected, 1 files total

- [✓] Show unversioned files
- [✓] Select / deselect all
- [✓] Show externals from different repositories
- [ ] Keep locks
- [ ] Keep changelists

Basic Diffs


Basic Diffs
Main Trunk
Milk
+Eggs
Milk
Eggs
+Juice
Milk
Eggs
Juice
-Juice
+Soup
Milk
Eggs
Soup
r1
r2
r3
r4

Example – Diff Viewer



list.txt Revision 6395

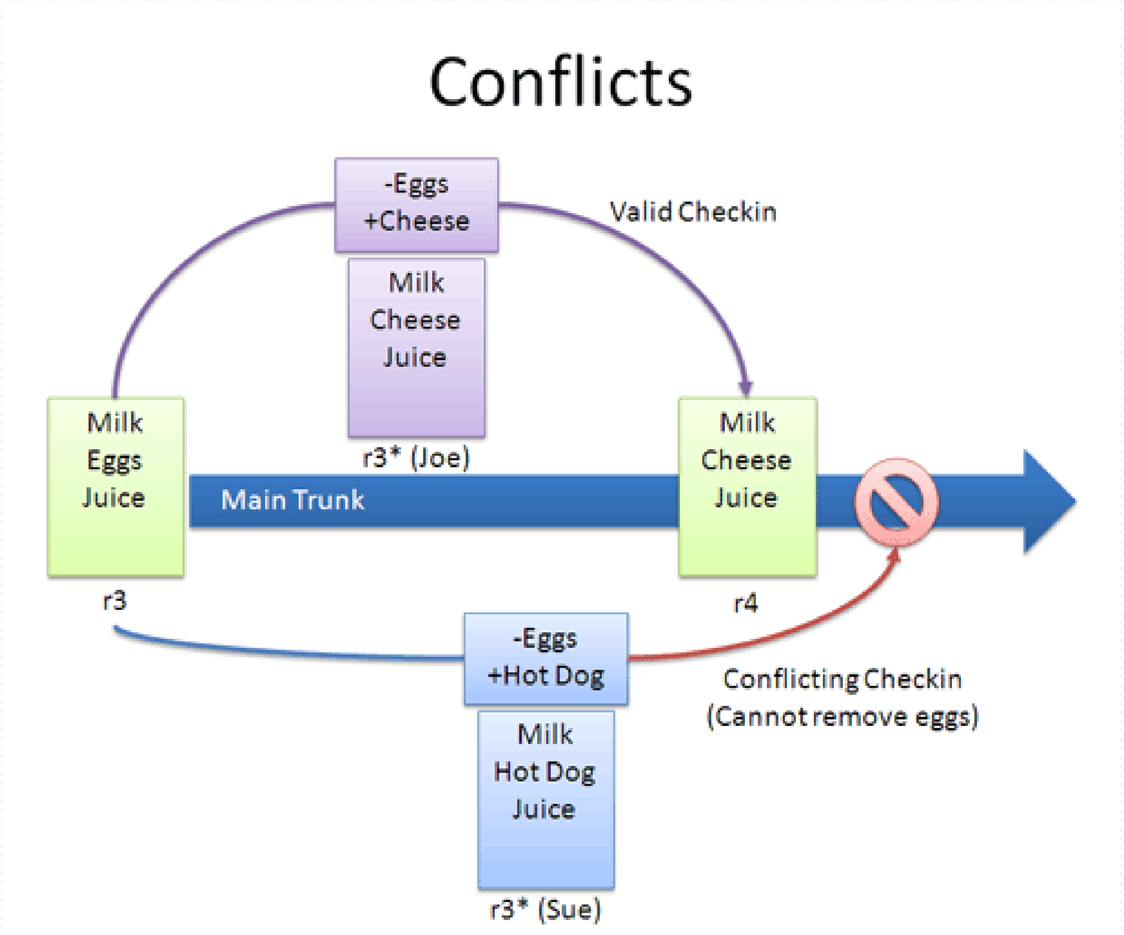
```
1 Milk
2 Eggs
3 ~~Juice~~
```

list.txt : Working Copy

```
1 Milk
2 Eggs
3 Soup
```

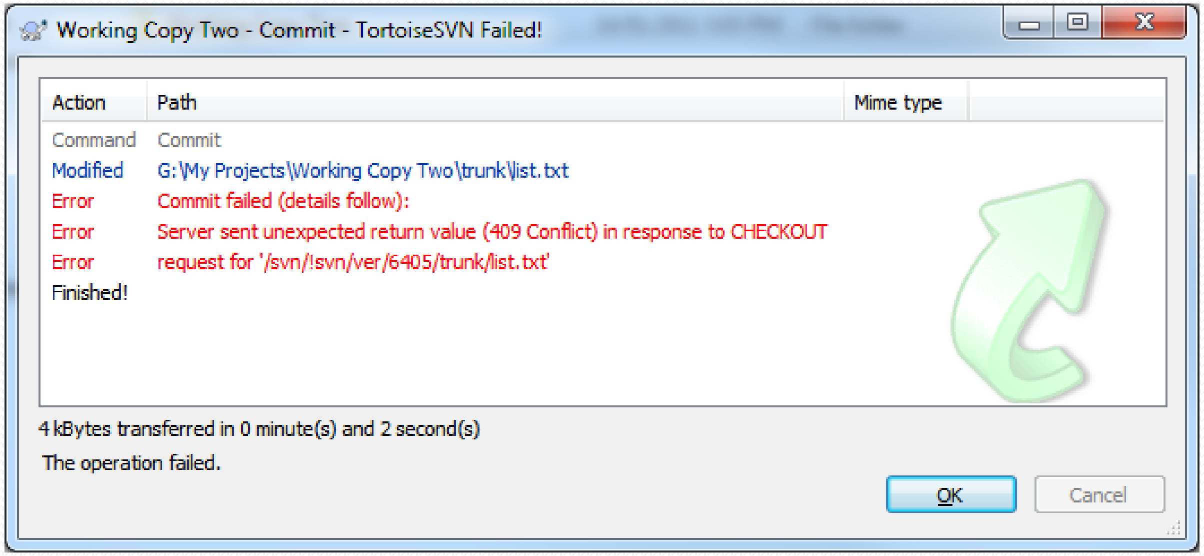
For Help, press F1. Scroll horizontally with Ctrl-Scrollwheel | Left View: ASCII CRLF / - 1 | Right View: ASCII CRLF / + 1 | Conflicts: 0 | CAP NUM SCRL

Conflicts


Conflicts
-Eggs
+Cheese
Valid Checkin
Milk
Cheese
Juice
r3* (Joe)
Milk
Eggs
Juice
Main Trunk
Milk
Cheese
Juice
r3
r4
-Eggs
+Hot Dog
Conflicting Checkin
(Cannot remove eggs)
Milk
Hot Dog
Juice
r3* (Sue)

Conflicts - Example

- Create two working copies of the project
- Add different items at the end of `list.txt`.
- Commit the file from the first working copy.
- Try to commit the second copy. You will get an error.



Working Copy Two - Commit - TortoiseSVN Failed!

```
Action     Path                                                     Mime type
Command    Commit
Modified   G:\My Projects\Working Copy Two\trunk\list.txt
Error      Commit failed (details follow):
Error      Server sent unexpected return value (409 Conflict) in response to CHECKOUT
Error      request for '/svn/!svn/ver/6405/trunk/list.txt'
Finished!
```

4 kBytes transferred in 0 minute(s) and 2 second(s)

The operation failed.

Conflicts – Example (continued)

- To see the conflicts, choose “SVN update” from the pop-up menu, on the second working copy.
- It will put conflicts inside the file, and also create three more files.
- The conflicting area inside the file in question is marked like this

```
<<<<<<< filename
    your changes
=======
    code merged from repository
>>>>>>> revision
```

Conflicts – Example (continued)

- Three additional files are:
- `filename.ext.mine`
  - This is your file as it existed in your working copy before you updated your working copy - that is, without conflict markers.
- `filename.ext.rOLDREV`
  - This is the file that was the BASE revision before you updated your working copy. That is, it the file that you checked out before you made your latest edits.
- `filename.ext.rNEWREV`
  - This is the file that your Subversion client just received from the server when you updated your working copy. This file corresponds to the HEAD revision of the repository.

Conflicts – Example (continued)

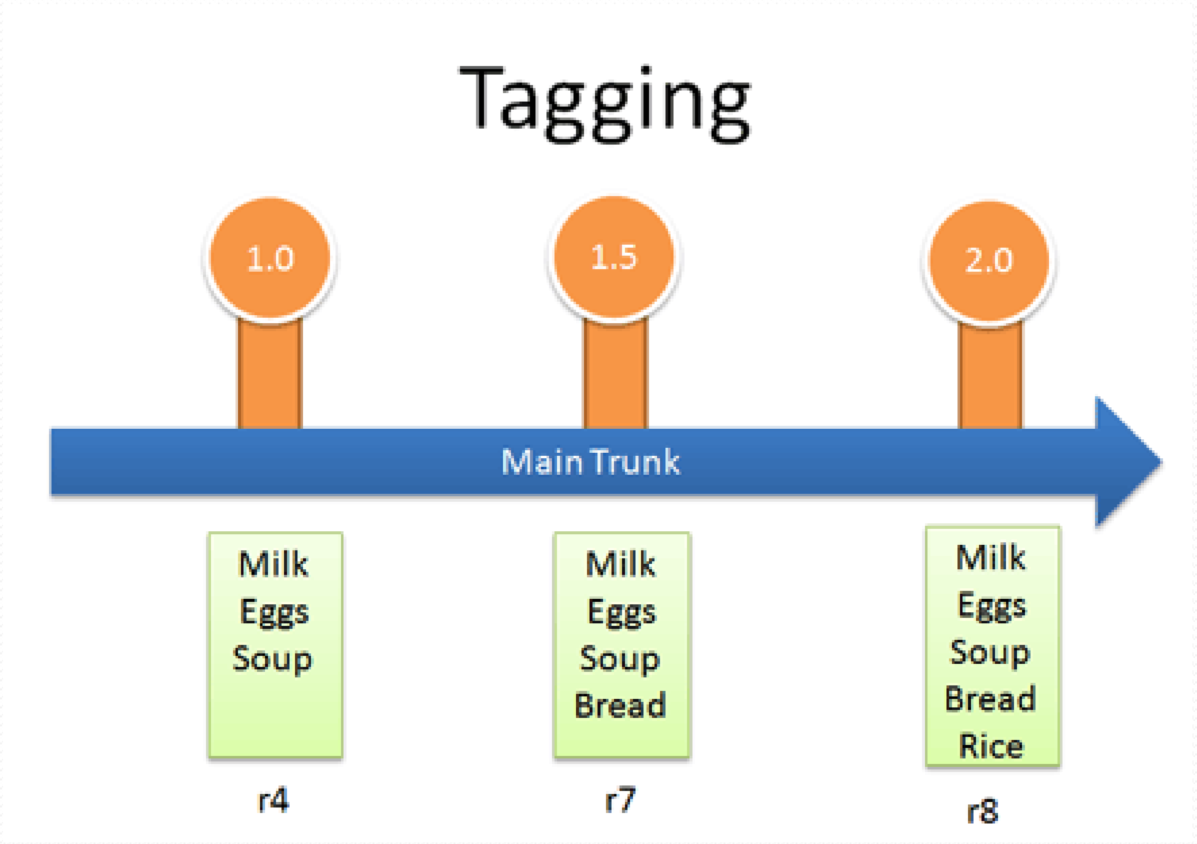
- Open the conflict editor by choosing “Edit conflicts” from the menu.
- You should decide what the code should look like, do the necessary changes and save the file.
- Afterwards execute the command TortoiseSVN → Resolved and commit your modifications to the repository.

Repository Structure

- There is no predefined way to organize stuff in the repository.
- A best practice is to use a structure like this:

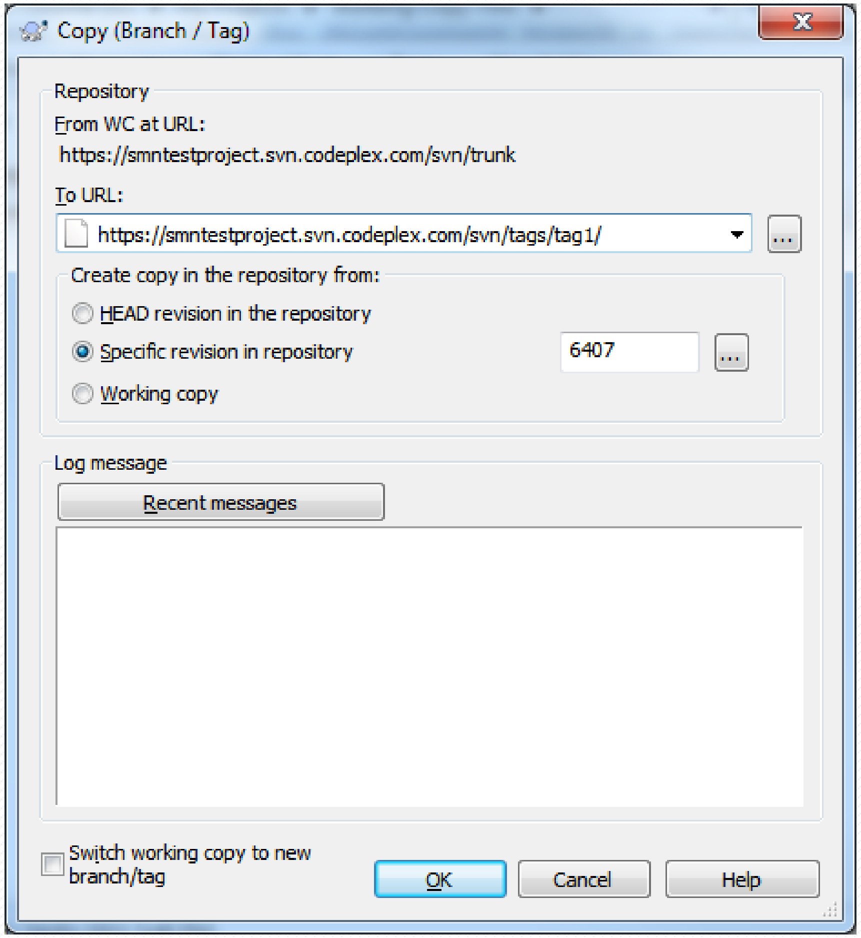
```
trunk
branches/branch1
branches/branch2
…
tags/tag1
tags/tag2
…
```

Tagging


Tagging
1.0
1.5
2.0
Main Trunk
Milk Eggs Soup
Milk Eggs Soup Bread
Milk Eggs Soup Bread Rice
r4
r7
r8

Tagging - Example

- Select the folder in your working copy which you want to copy to a branch or tag, then select the command TortoiseSVN → Branch/Tag….
- You should change the "to URL" value to a new path for the tag.


Copy (Branch / Tag)

Repository

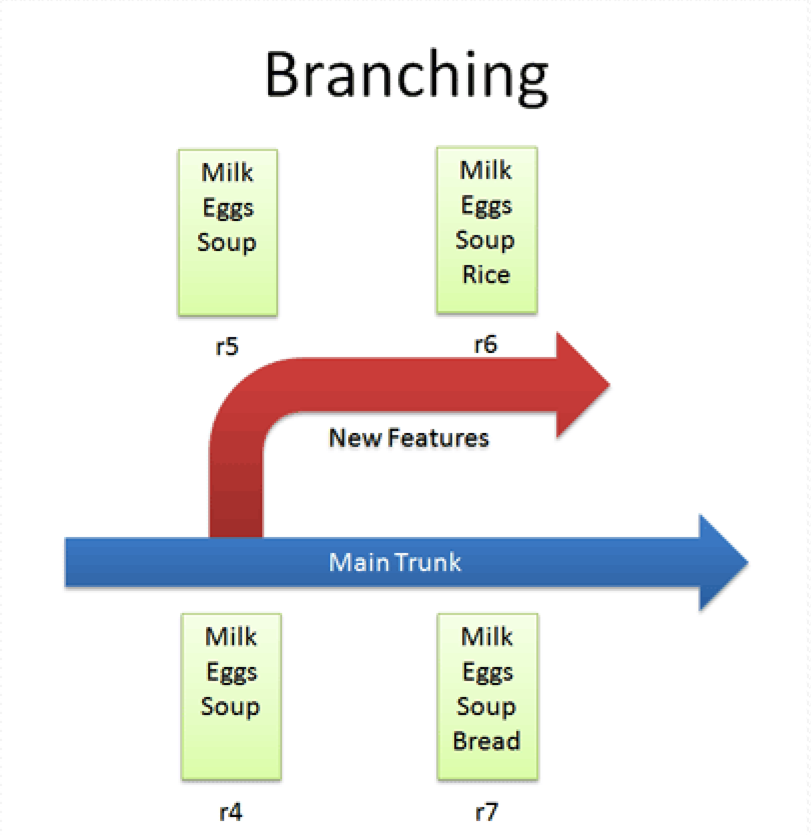
From WC at URL:
https://smntestproject.svn.codeplex.com/svn/trunk

To URL:
https://smntestproject.svn.codeplex.com/svn/tags/tag1/

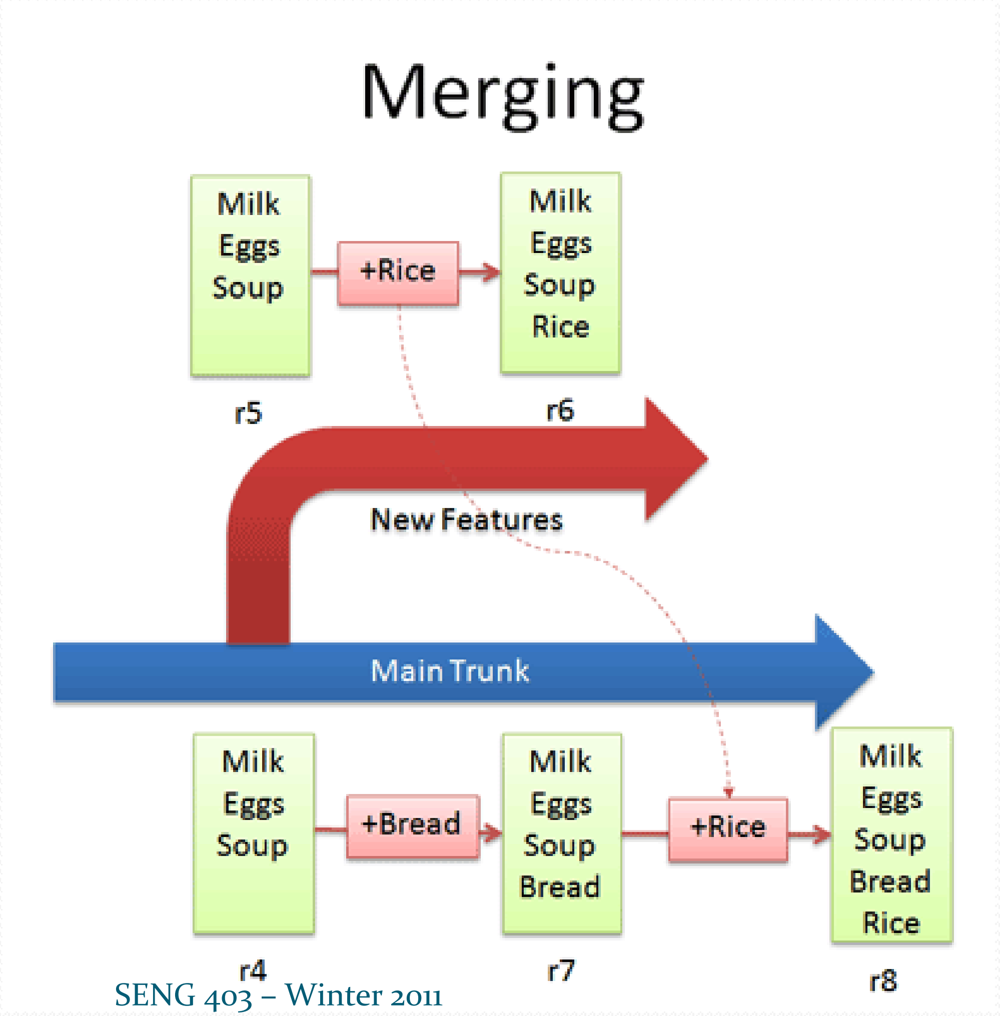
Create copy in the repository from:
- HEAD revision in the repository
- [✓ Specific revision in repository] 6407
- Working copy

Log message

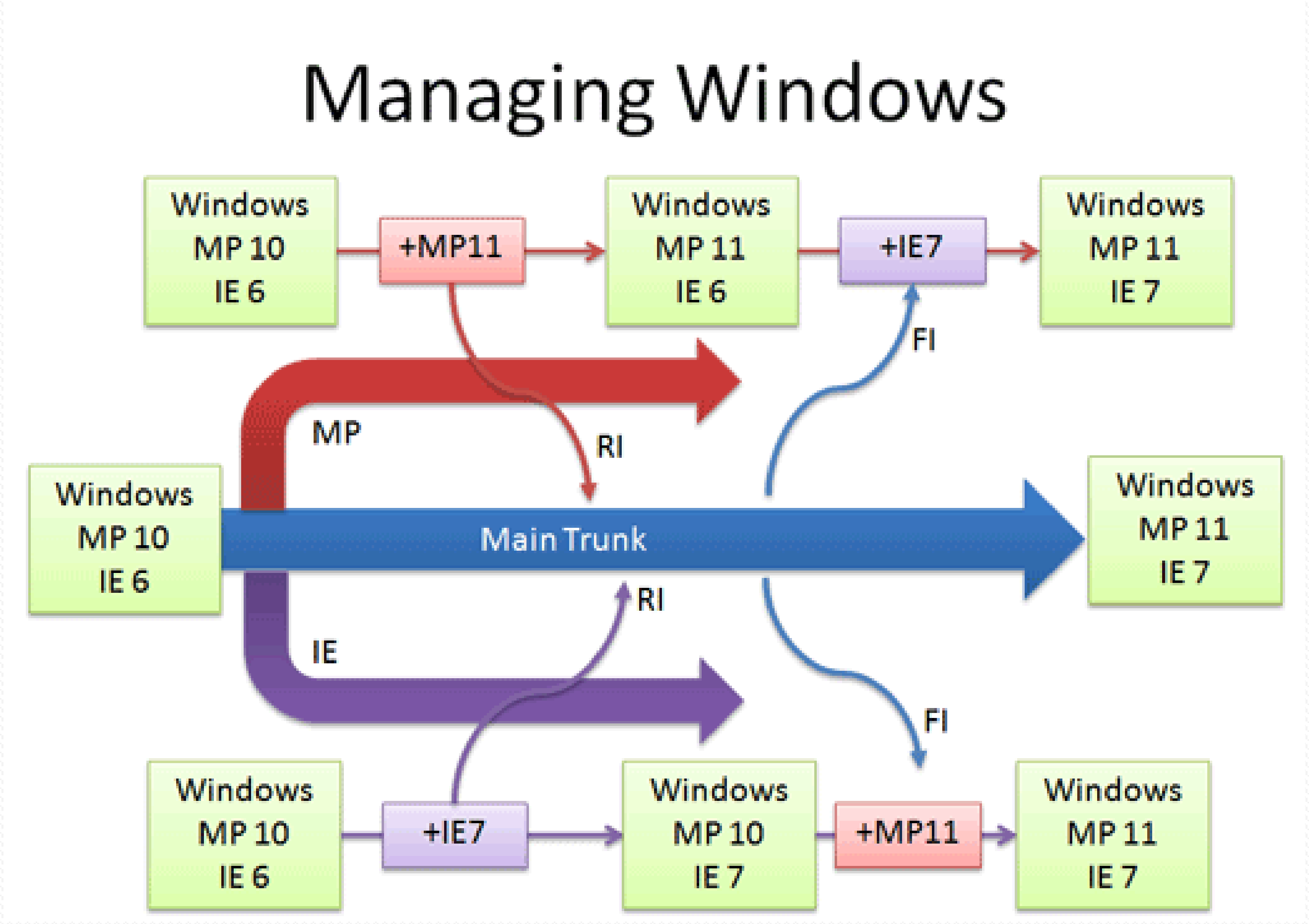
Branching


Branching
Milk
Eggs
Soup
r5
Milk
Eggs
Soup
Rice
r6
New Features
Main Trunk
Milk
Eggs
Soup
r4
Milk
Eggs
Soup
Bread
r7

Merging


Merging
Milk Eggs Soup
+Rice
Milk Eggs Soup Rice
r5
r6
New Features
Main Trunk
Milk Eggs Soup
+Bread
Milk Eggs Soup Bread
+Rice
Milk Eggs Soup Bread Rice
r4
r7
r8

Real-life example: Managing Windows Source Code



References & Further Readings

- Subversion book
  http://svnbook.red-bean.com/
- TortoiseSVN Help: http://tortoisesvn.net/docs/release/TortoiseSVN_en/index.html
- A Visual Guide to Version Control:
  http://betterexplained.com/articles/a-visual-guide-to-version-control/