

USING SVN IN GCS

Version: 1.1



Global CyberSoft

A World of Difference

By: **Binh Le**

Course duration: **3 hours**

Last update: **Apr-2011**

www.globalcybersoft.com



Confidential

Page 1

QUALITY MANAGENT SYSTEM

DOCUMENT CHANGE LOG

| Revision | Description | Author | Date | Approved | Date |
|--|--|----------------------------|--------|----------|--------|
| 1.0 | First version | Trung Doan, Dung Nguyen | 2006 | | |
| 1.1 | Revised and updated pages as new SVN structure | Binh Le | Apr-11 | Toan Ngo | Apr-11 |
| This document has been generated from template TL-QMS-005 Revision 3.2 | | | | | |

Table of Contents

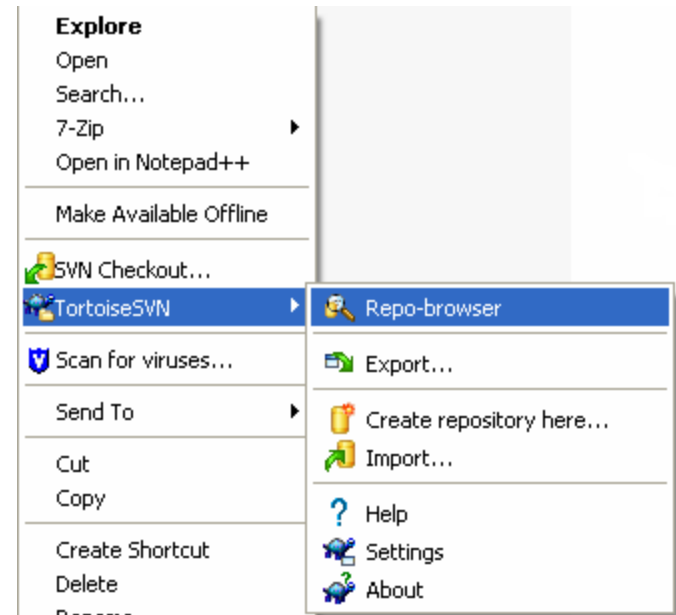
- ❑ Setup TortoiseSVN
- ❑ Grant access right
- ❑ Basic usage of Subversion
- ❑ Manage versions
- ❑ Appendix

Setup TortoiseSVN

Find the latest TortoiseSVN at: [\\cybersoft.vn\GCS\Apps\Free](http://cybersoft.vn/GCS/Apps/Free)

❑ After installing, TortoiseSVN is integrated into Windows Explorer: right click>TortoiseSVN

❑ For stand-alone PC, TortoiseSVN can be used as source control system, it's not necessary to setup a SVN server.



Grant access right to the repository

- ❑ Admin, PM or CM (configuration management) staff to do this.
- ❑ Member must be granted full access right to the destination location which contains the repository.

- ❑ At this location:

\\cybersoft.vn\gcs\GHOSVN\[Division]\[Group]\[Project]\Permission, edit file **authz-svn.conf**, add team member as follows (use Windows login name):

[groups]

svn_owner = CYBERSOFT\xnguyenhong

svn_read = CYBERSOFT\ytranvan

[/]

@svn_owner = rw

@svn_read = r

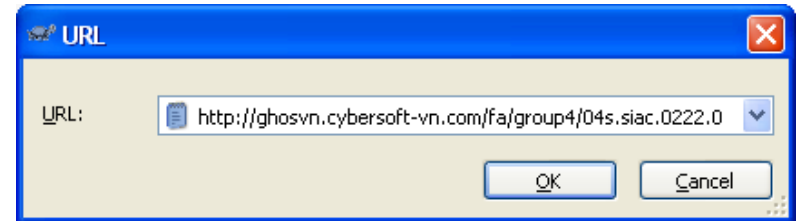
Basic Usage of Subversion

Working cycle

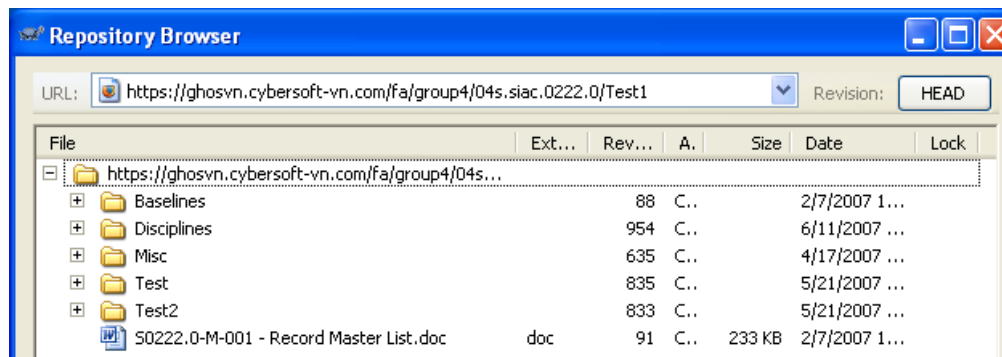
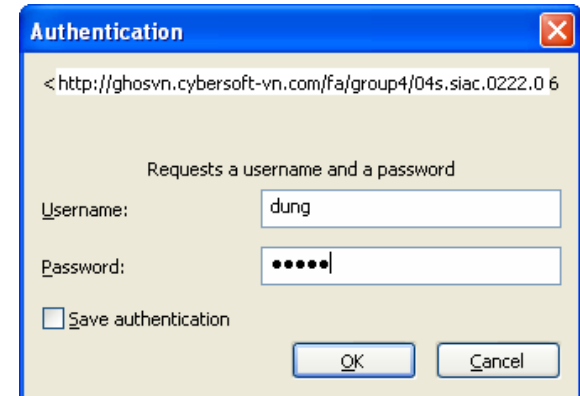
- Browse the repository
- Add folders or files to repository
- Checkout a working copy
- Get lock
- Update working copy
- Commit to the repository
- Resolve conflict if happened
- View log
- View differences
- Undo changes
- Cleanup
- Status of version controlled files/folders

Browse the repository

- ❑ In Windows Explorer, right click>TortoiseSVN>Repo-browser
- ❑ Put the repository link in URL:
You can use https://...



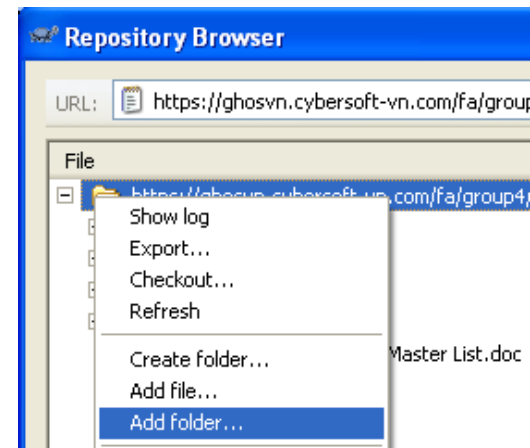
- ❑ Login with username and password:
- ❑ The Repo-browser opened:



Add file/folder to the repository

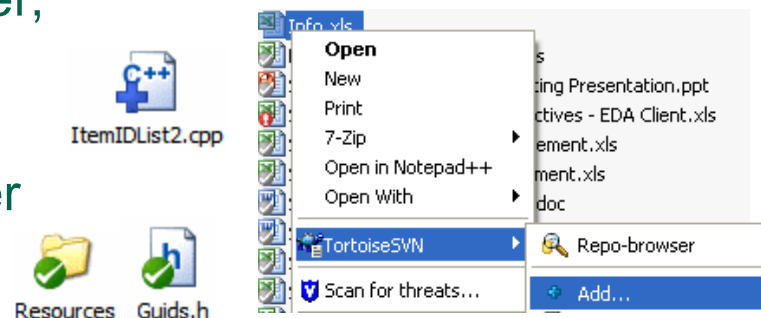
1. From repository browser

- ❑ In the Repo-browser, right click>Add folder...
- ❑ Select source file/folder to be source-controlled.
- ❑ The new file/folder will be added to the repository, under the current selected folder.



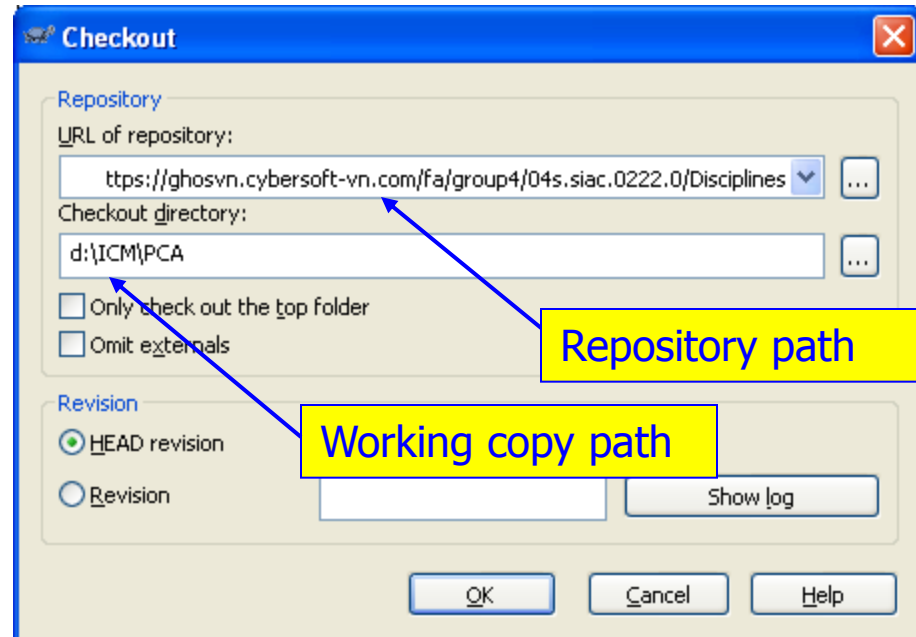
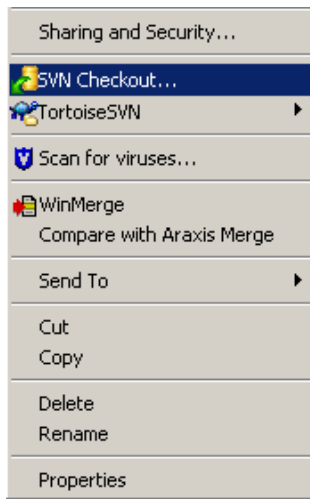
2. From the working copy

- ❑ Copy new file/folder to the working copy
- ❑ In Windows Explorer, select new file/folder, right click>TortoiseSVN>Add...
=> new file/folder has a blue plus
- ❑ TortoiseSVN>Commit... => new file/folder has a check mark



SVN Checkout

- ❑ To obtain a working copy, you need to do a checkout from the repository.
- ❑ In the Windows Explorer, select the working folder, right click> SVN Checkout... OR
- ❑ In Repo-browser, select the folder to checkout, right click> Checkout...

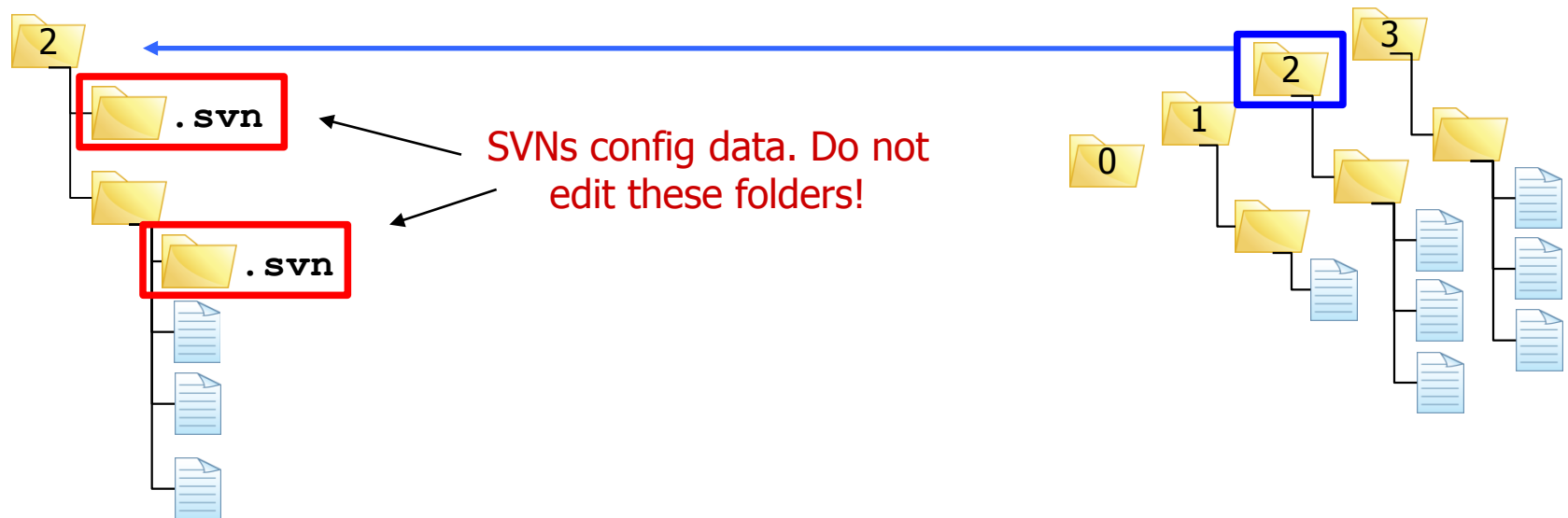


SVN Checkout (cont.)

“Checking out” creates a working copy of a specific revision of the repository

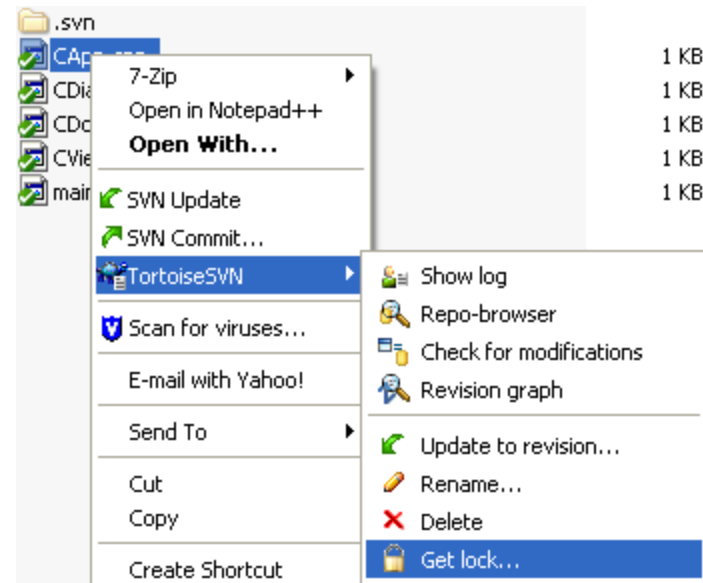
working copy

repository



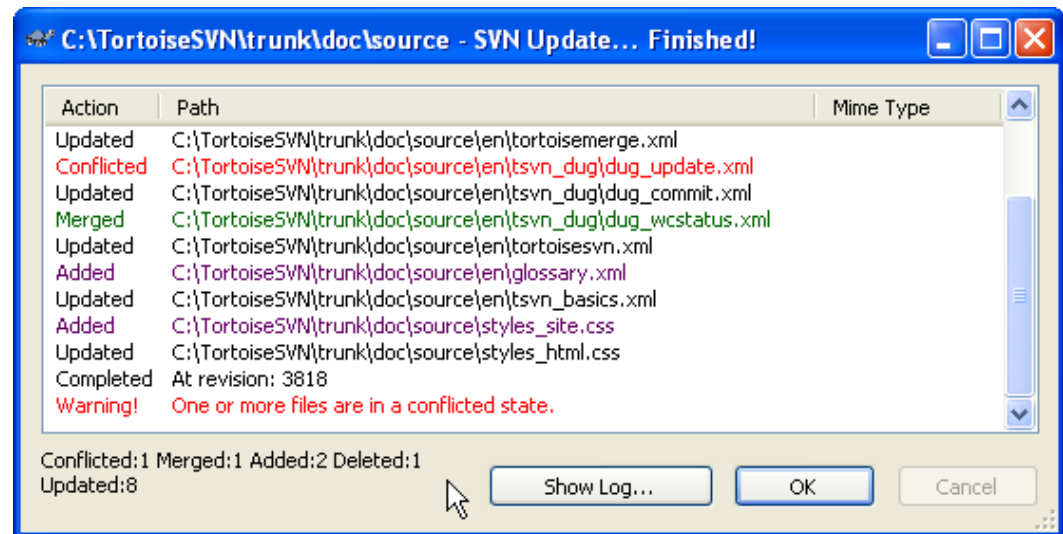
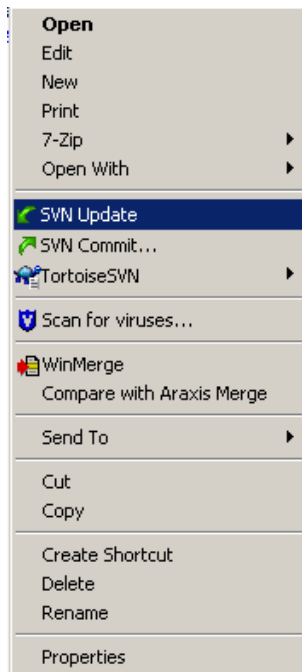
Get Lock

- ❑ “Get Lock” is to avoid the conflict when another user update the repository while you are working on your (local) working copy.
- ❑ In the working copy, right-click>TortoiseSVN>Get lock...
- ❑ The other user will not be able to commit until you release the locked files.



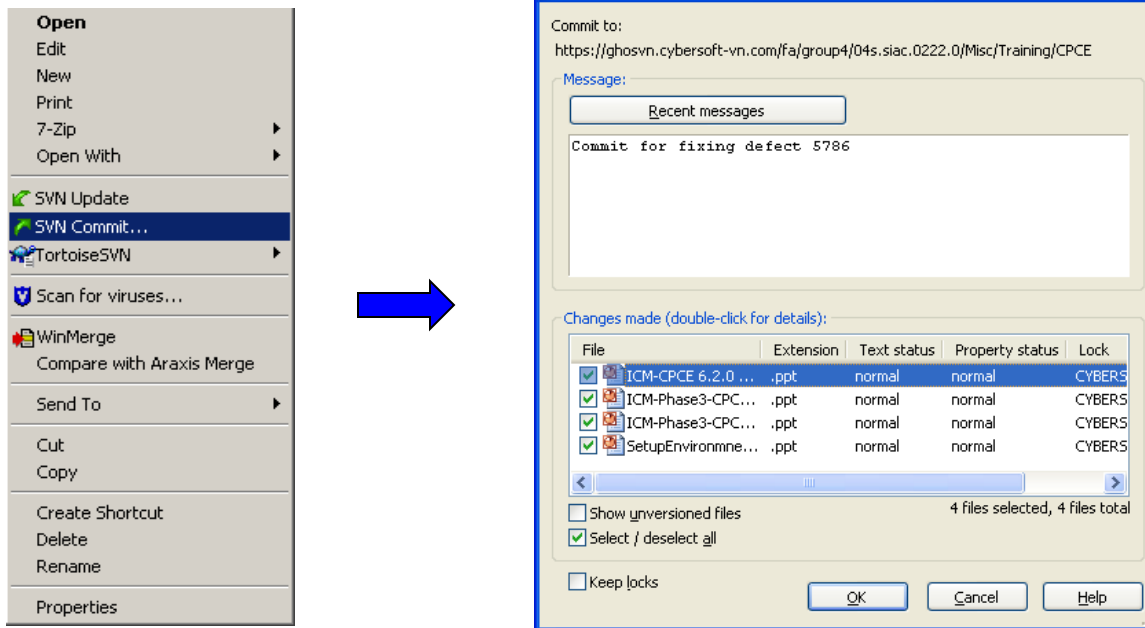
SVN Update

- ❑ To update new changes on the repository to the working copy
- ❑ Periodically, you should ensure that changes done by others get incorporated in your local working copy.
- ❑ At the working copy, select folder to be updated, right click>SVN Update...



SVN Commit

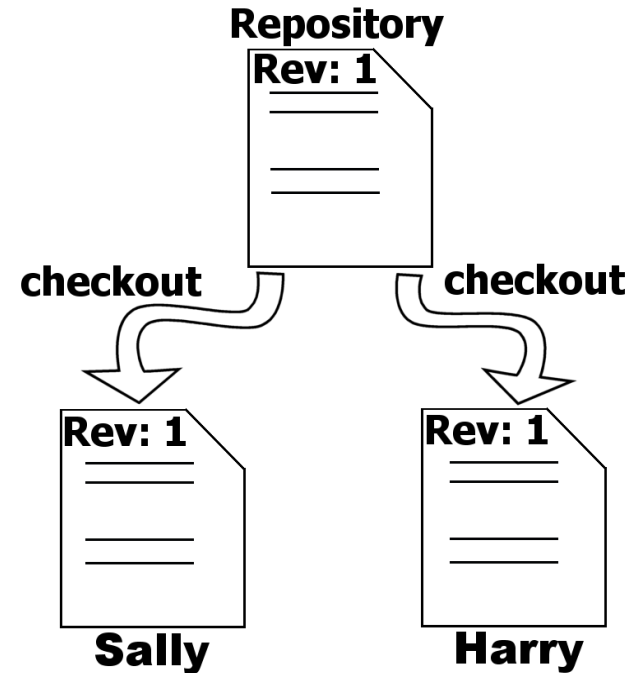
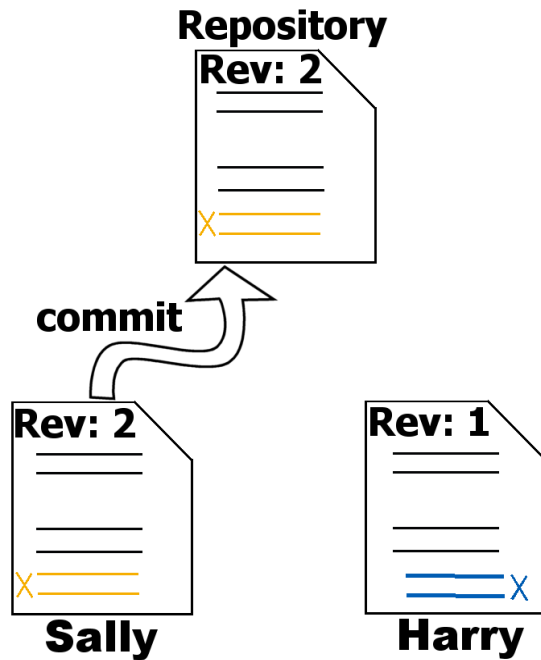
- ❑ Sending the changes you made on your working copy to the repository is known as committing the changes.
- ❑ On the working copy folder, right click>SVN Commit...



Before committing, you should make sure that your working copy is **up-to-date**.

Teamwork – Conflict happened

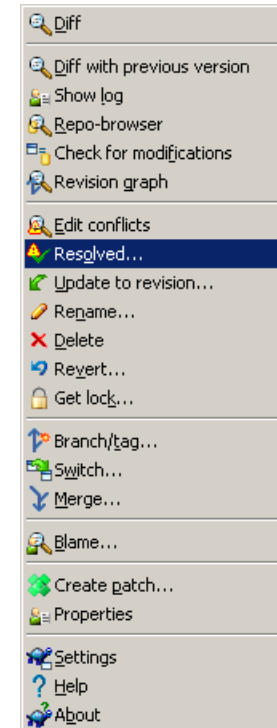
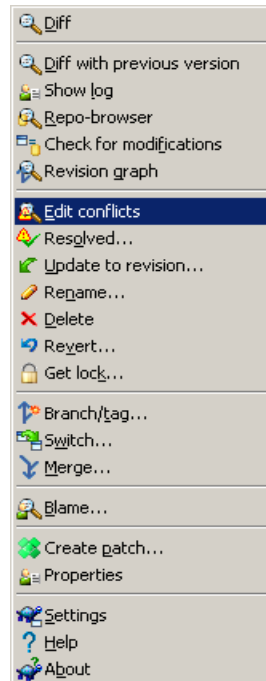
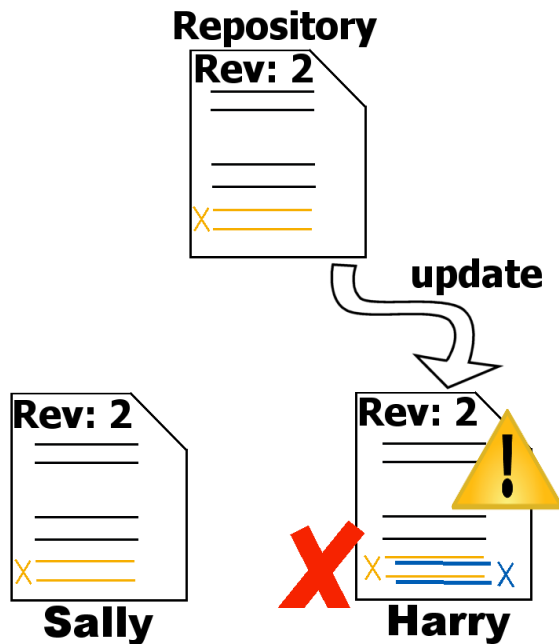
Harry and Sally check out the same file and revision to their working copy
They will change some lines in this particular file.



Sally commits her changes and creates a new revision.
Her changes had been submitted to the repository

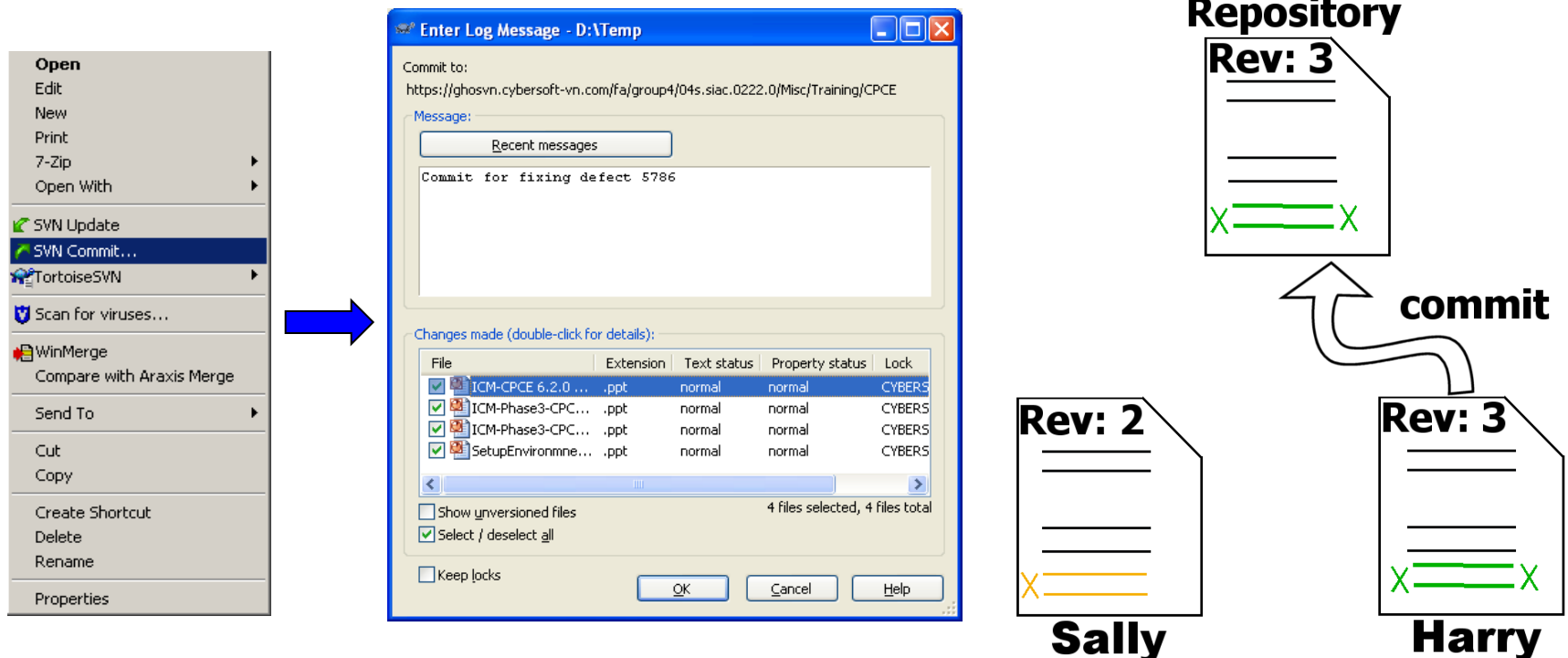
Teamwork - Edit and resolve conflict

- ❑ Harry tries to commit his changes, he receives the well known “out-of-date”-error and an advice to update his working copy.
- ❑ Harry cannot commit a file which is in a conflict state.
- ❑ Harry has to resolve the conflict/s and notifies SVN via **svn resolved**.



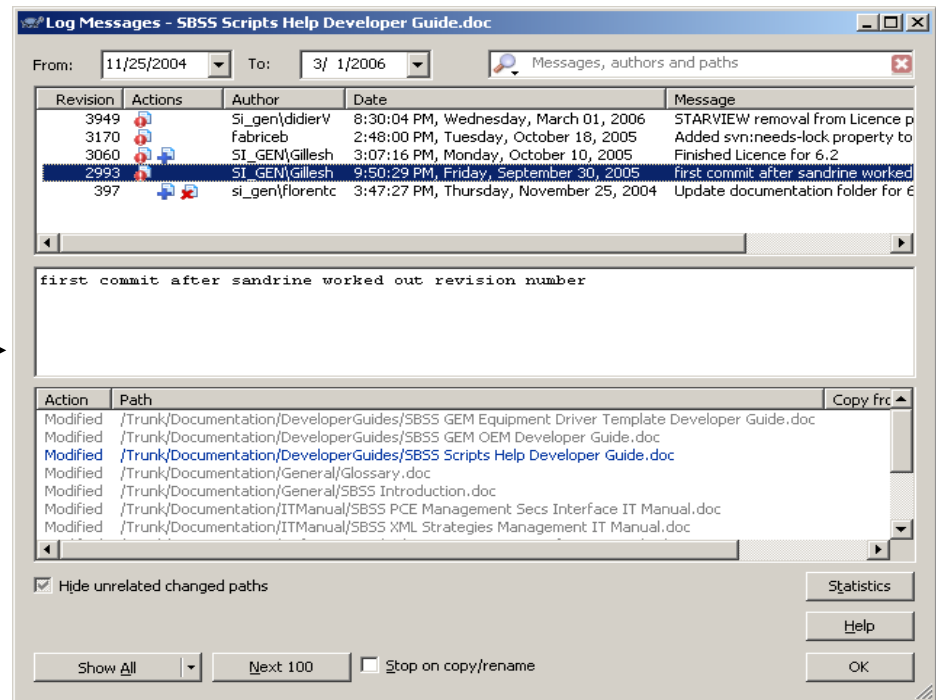
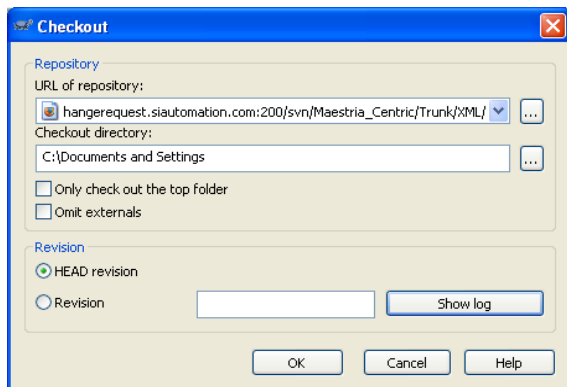
Teamwork - Commit the changes

After resolving the conflict, Harry can commit his changes and creates a new revision

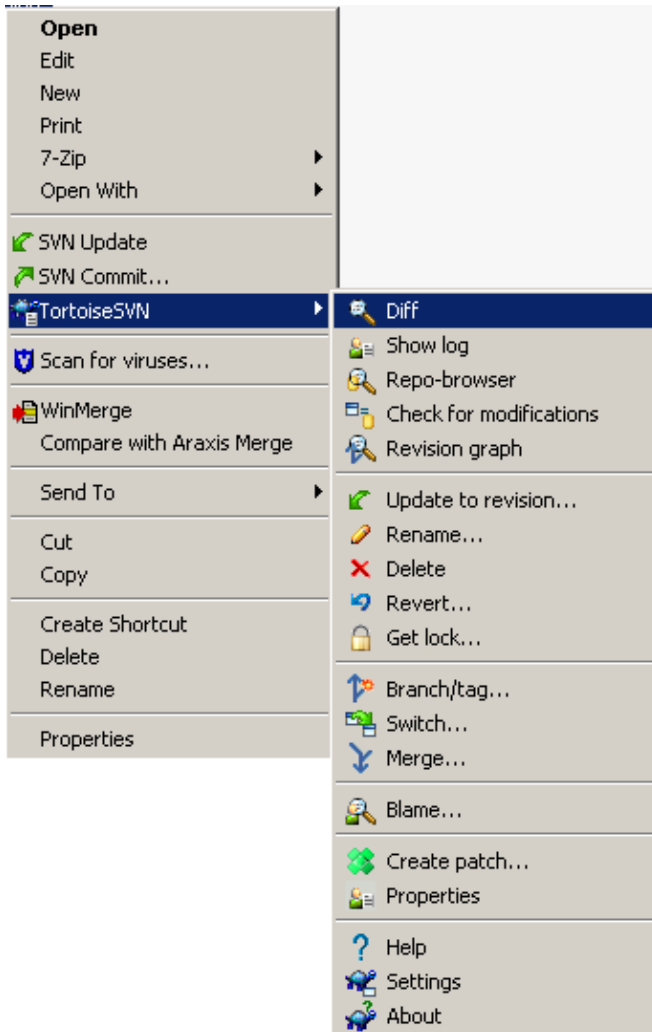


Show Log

- ❑ For every change you make and commit, you should provide a log message for that change.
- ❑ The Revision Log Dialog retrieves all those log messages and allows you to select the desired revision.



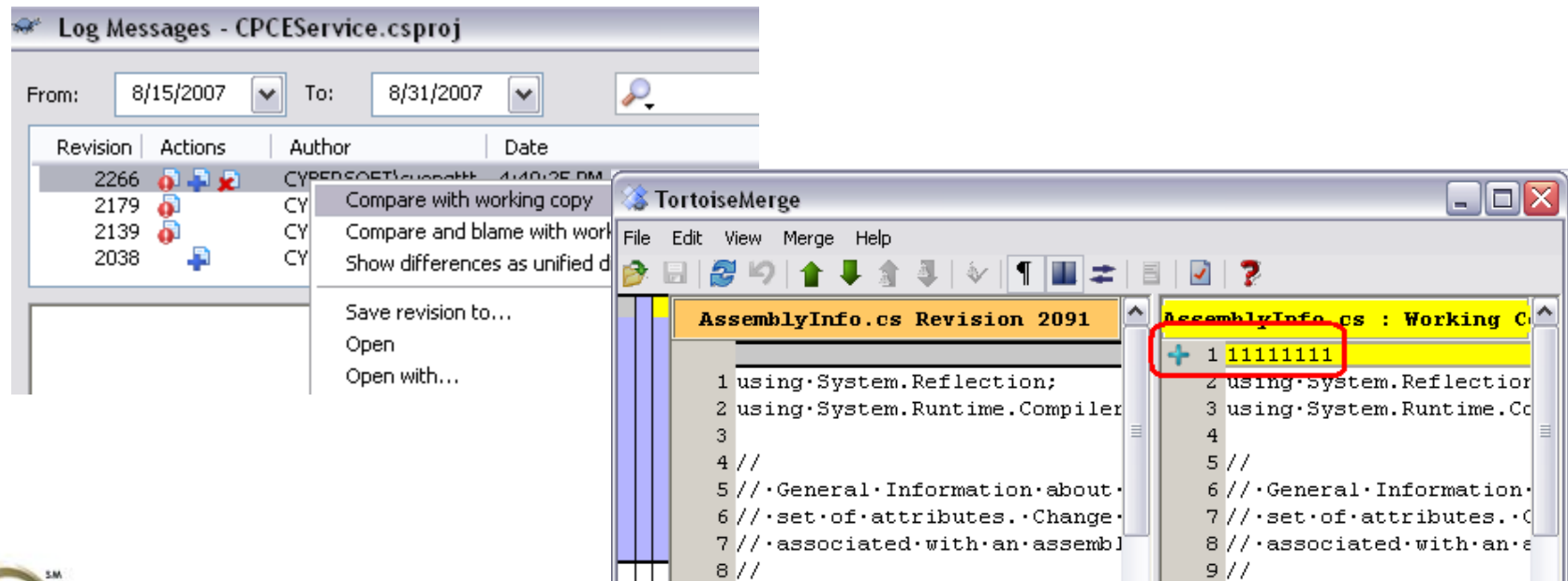
Compare differences



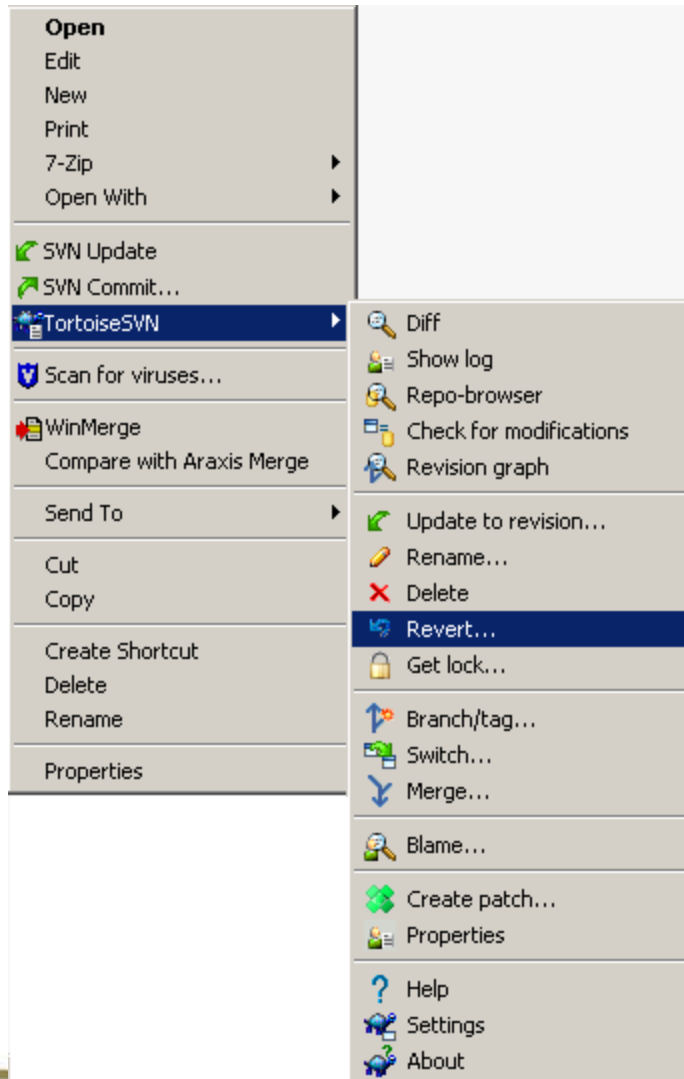
- ❑ You might want to look at the differences between two revisions of the same file, or the differences between two separate files
- ❑ You can use **TortoiseMerge** for viewing differences of text files.
- ❑ For viewing differences of image files, TortoiseSVN also has a tool named TortoiseIDiff
- ❑ Beside that you can use your own favorite diff program if you like

Compare differences (cont.)

- ❑ Compare working copy with a given SVN revision
- ❑ On working copy, select file to compare
- ❑ Right click\Show log => Log messages dialog opened.
- ❑ Select the revision to compare, right click\Compare with local copy
- ❑ The TortoiseMerge dialog opened showing 2 files vertically.

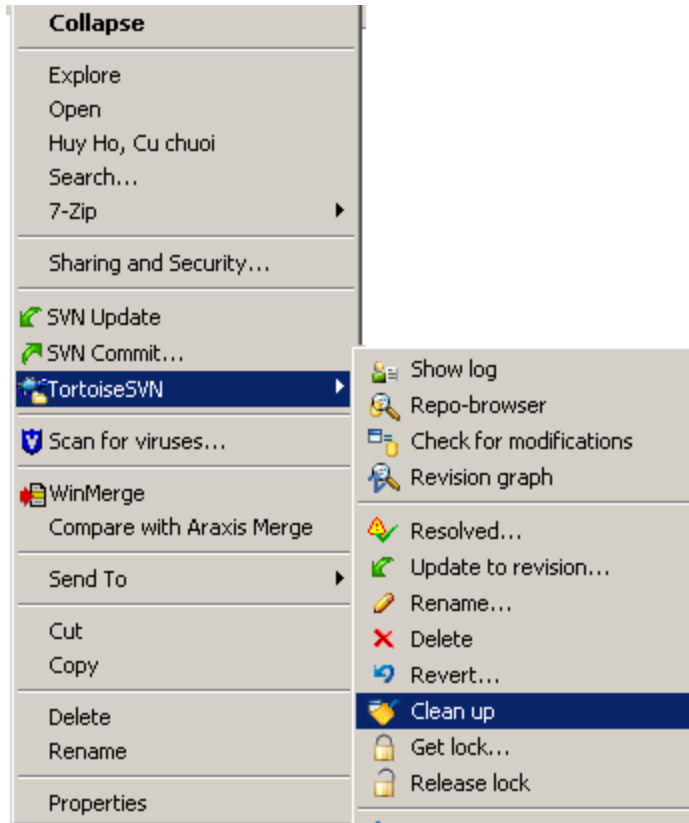


Undo changes



- ❑ If you want to undo all changes you made in a file since the last update you select the command TortoiseSVN – Revert
- ❑ Revert will only undo your local changes

Clean up



- ❑ If a Subversion command cannot complete successfully, perhaps due to server problems, your working copy can be left in an inconsistent state.
- ❑ It is a good idea to do “clean up” at the top level of the working copy

Status of version controlled files/folders



Checkmark: Subversion status is normal



Red exclamation: file has been modified since last update and need to be committed.



Yellow exclamation: a conflict occurs during an update.



This file need to be locked first before editing.



File is locked. Need to unlock for other to commit.



Missing file or file to be deleted under version control.



File to be added to version control.

Question and Answer



Manage Versions

Tagging - Why Tagging?

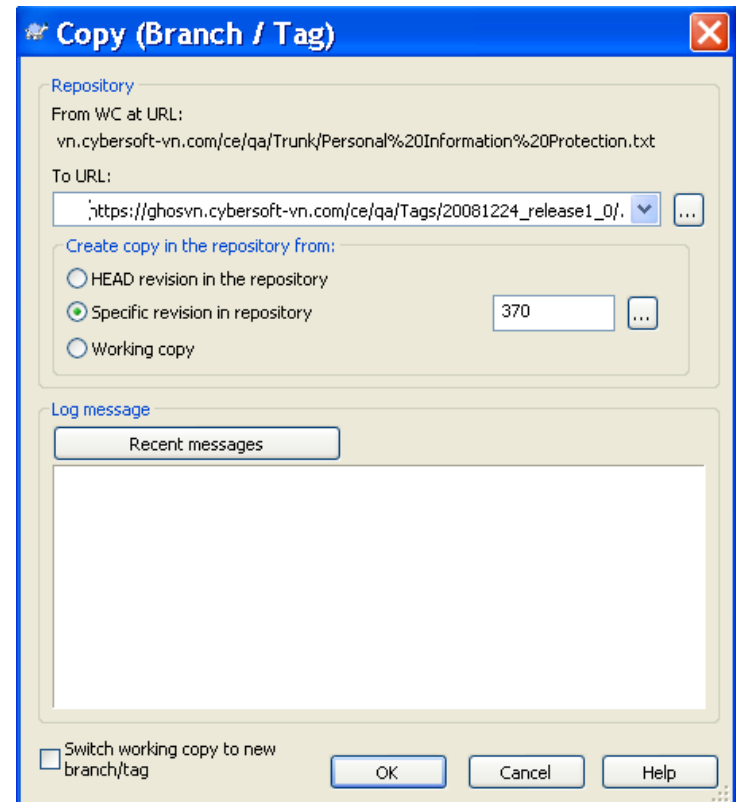
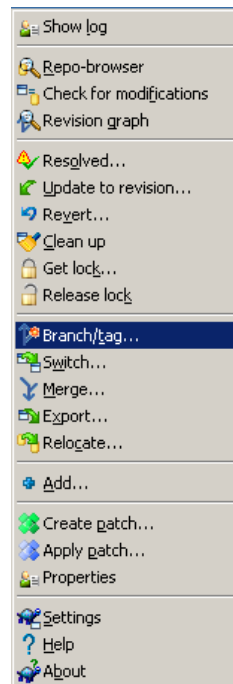
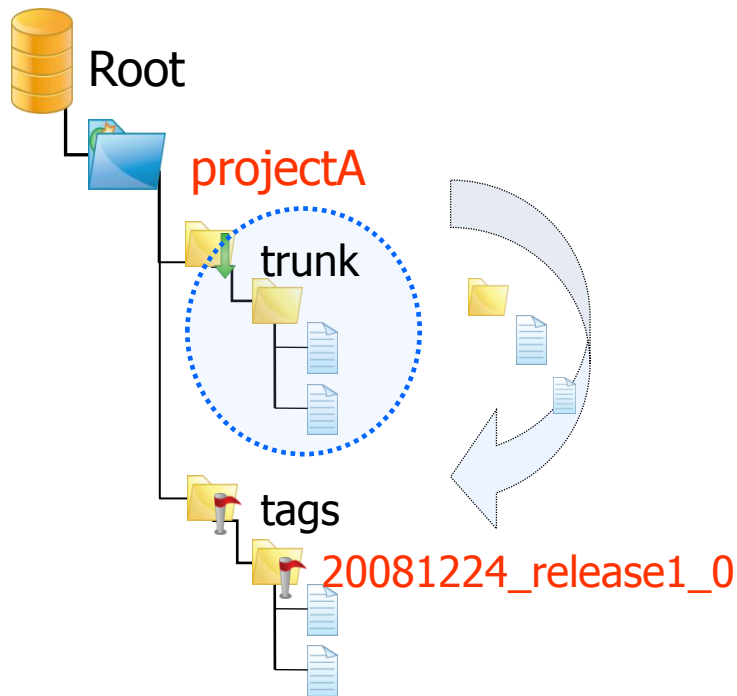
- ❑ Why do we need tags?
 - Mark a release state of a product.
 - Mark a snapshot of the current development.

- ❑ Typical Release names:
 - **yyymmdd_ReleaseName**
 - 20081223_Release1; 20090102_BetaRelease

- ❑ A Tag name should be unique to mark all components of the given product (source code and documentation) and is used to reproduce the state of the tag in the future.

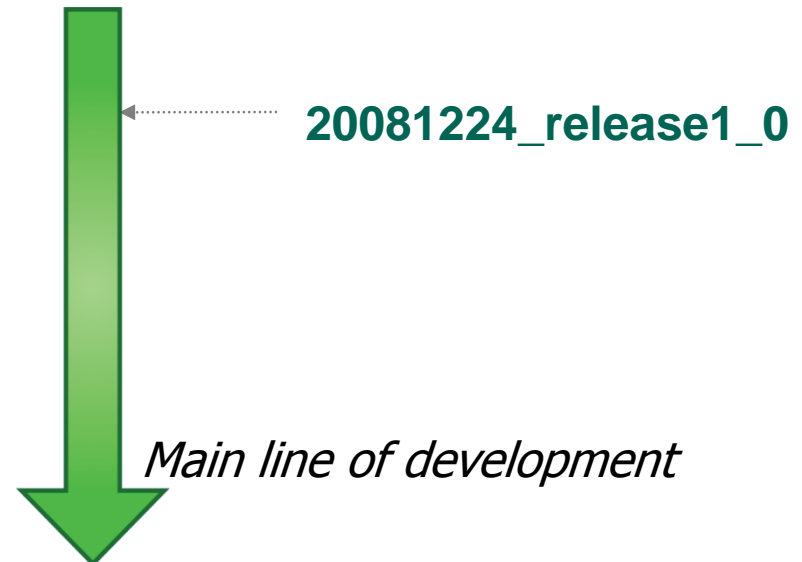
Create a Tag

- ❑ In the Windows Explorer, select the working folder, right click> TortoiseSVN>Branch/tag...
- ❑ Create a tag named **../tags/20081224_release1_0**.
- ❑ Commit message should be provided.



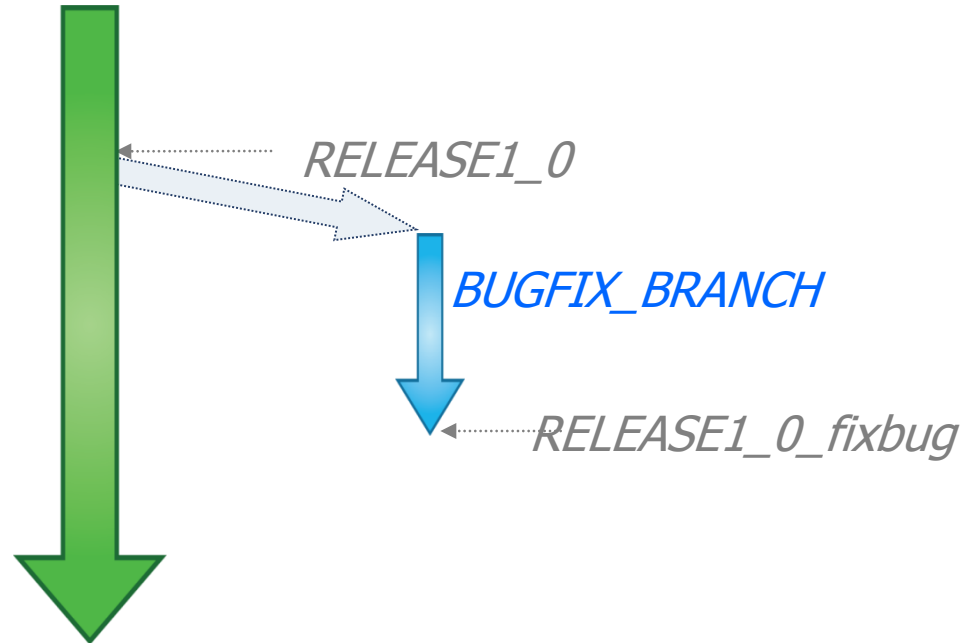
Branches – Why branching?

- ❑ You have delivered a product to your customer. Before you delivered the product you have created a tag, let it named “**20081224_release1_0**”.
- ❑ Your current development team is working on new features. At this time, client found a bug in your release 1.0 and you want to fix the bug to satisfy your customer.
- ❑ In your current development you have enhanced many of the product’s functions and you don’t want to deliver a product with more features which you haven’t finished testing yet.
- ❑ How to solve this situation?

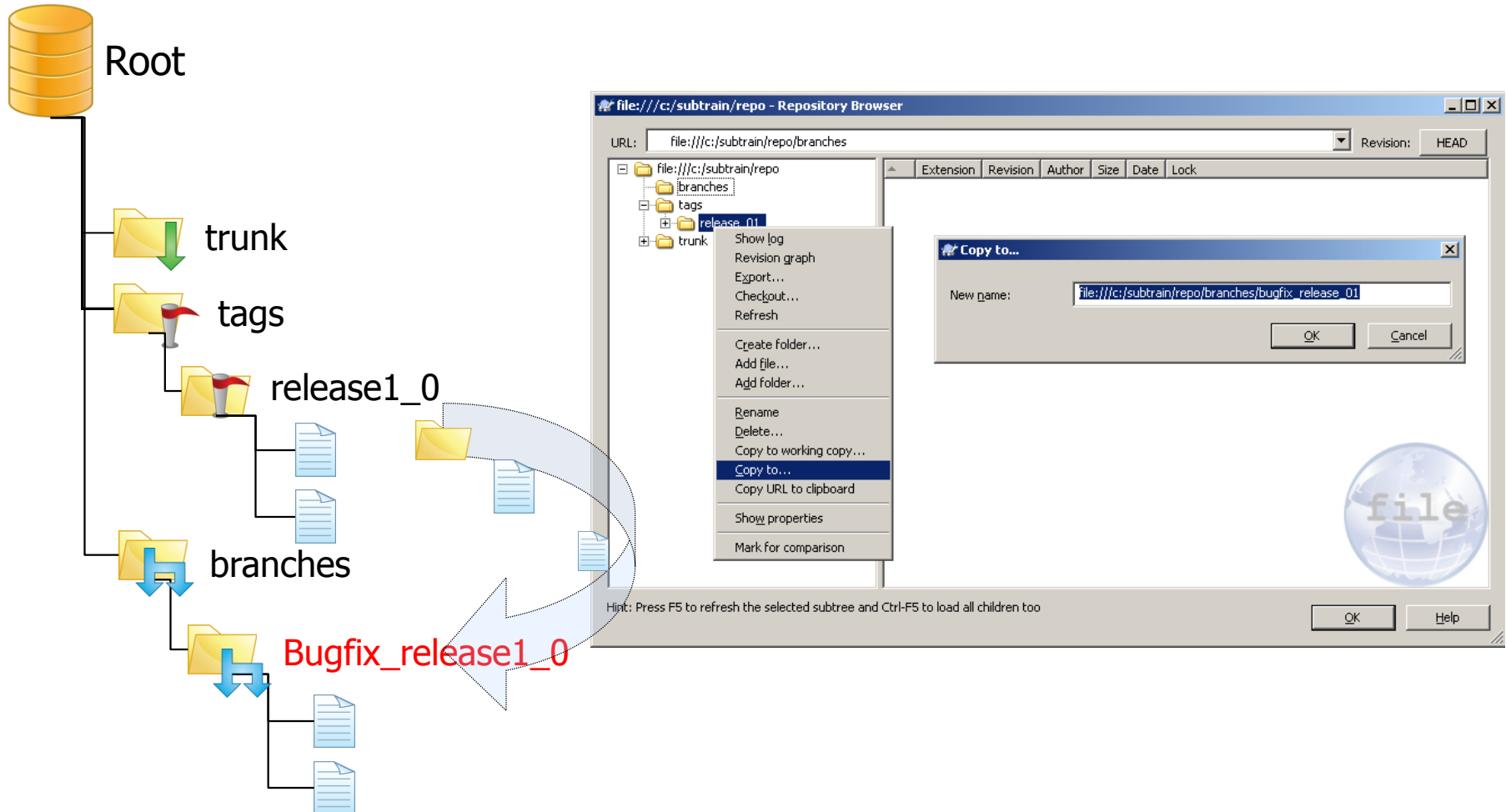


Branches – Why branching?

- ❑ Based on the tag you've created during the delivery you can check out the exact state of the delivery.
- ❑ You create a **Branch** to fix the bug in the software.
- ❑ After you have fixed the bug
- ❑ You can tag the Branch and deliver another version to the customer.
- ❑ You haven't disturbed the current development.



Create Branches



Switching branches

- If you want to work on a different branch, you can “switch” your working copy instead of a new checkout.

1

svn switch

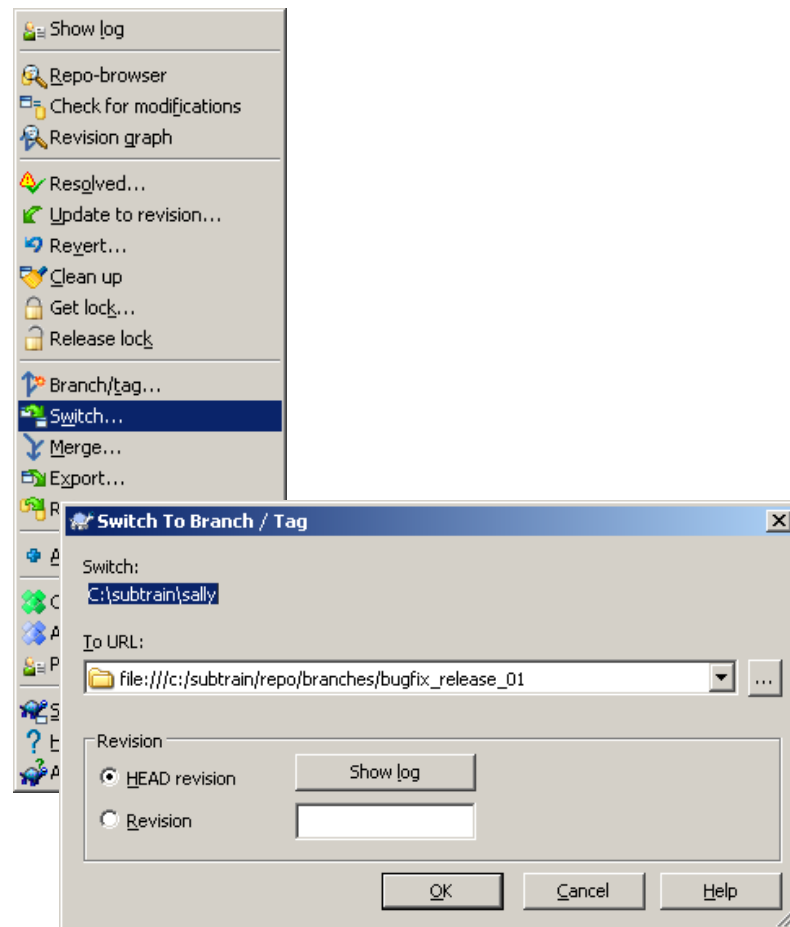
2

url

3

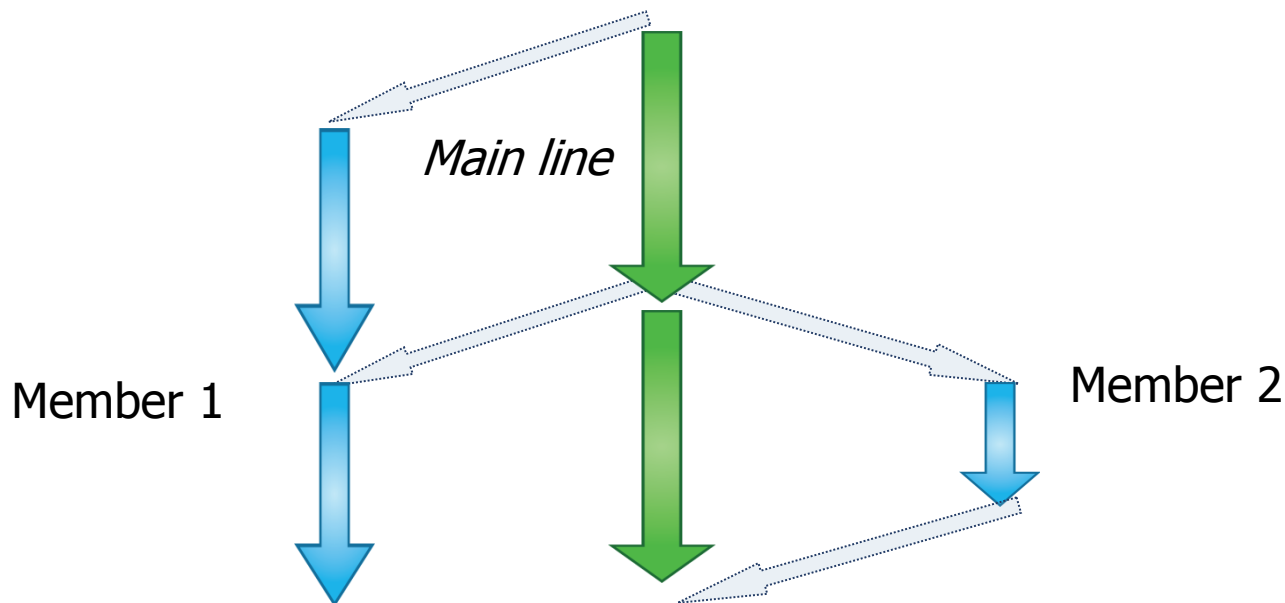
path

- 1) The Subversion “switch”-command.
- 2) The url where your working copy should point to.
- 3) The path of your working copy.



Branching Strategies - Developer Branches

- ❑ Separation of team members can be realized with branches.
- ❑ One branch per team member or several members on a branch. The decision is based on the size of the teams.



Branching Strategies -Developer Branches

❑ Advantages using branches for team work:

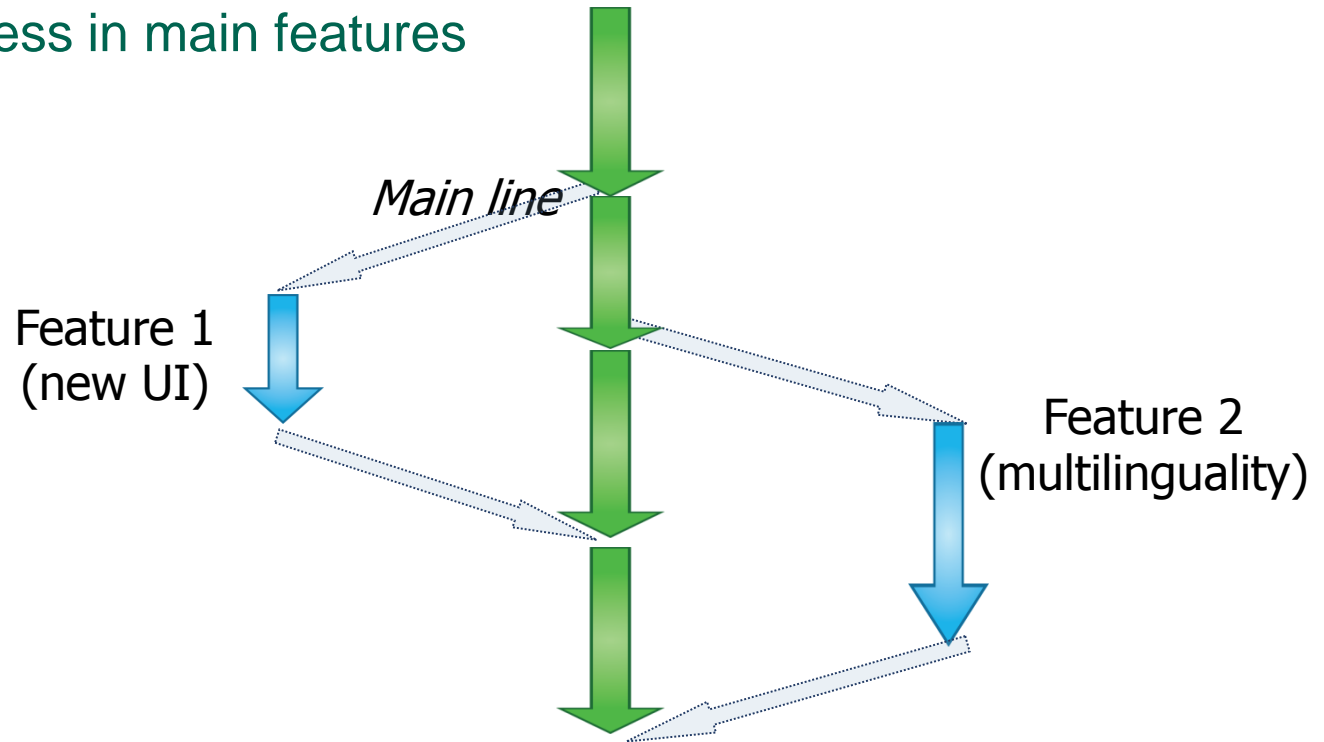
- No changes during development on the main line needed => Code stability.
- Every team member can work in its own environment.

❑ Disadvantages:

- Sometimes the mainline and the branch will diverge if the branch lives too long.
- Large merge effort for maintainer of trunk

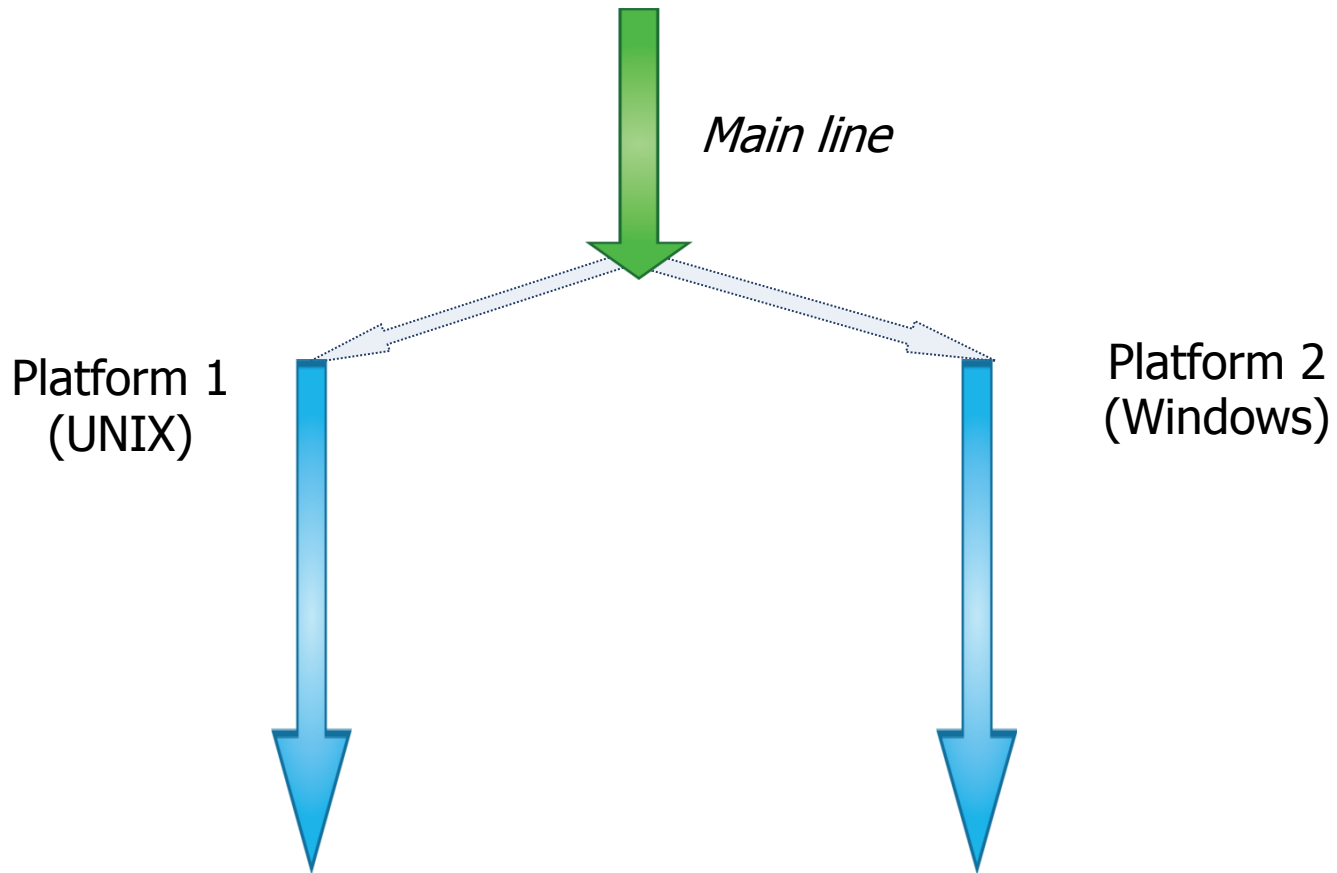
Branching Strategies - Feature Branches

- ❑ Separation by features (one branch each).
- ❑ Branch will be obsolete after merge
- ❑ Trunk will progress in main features



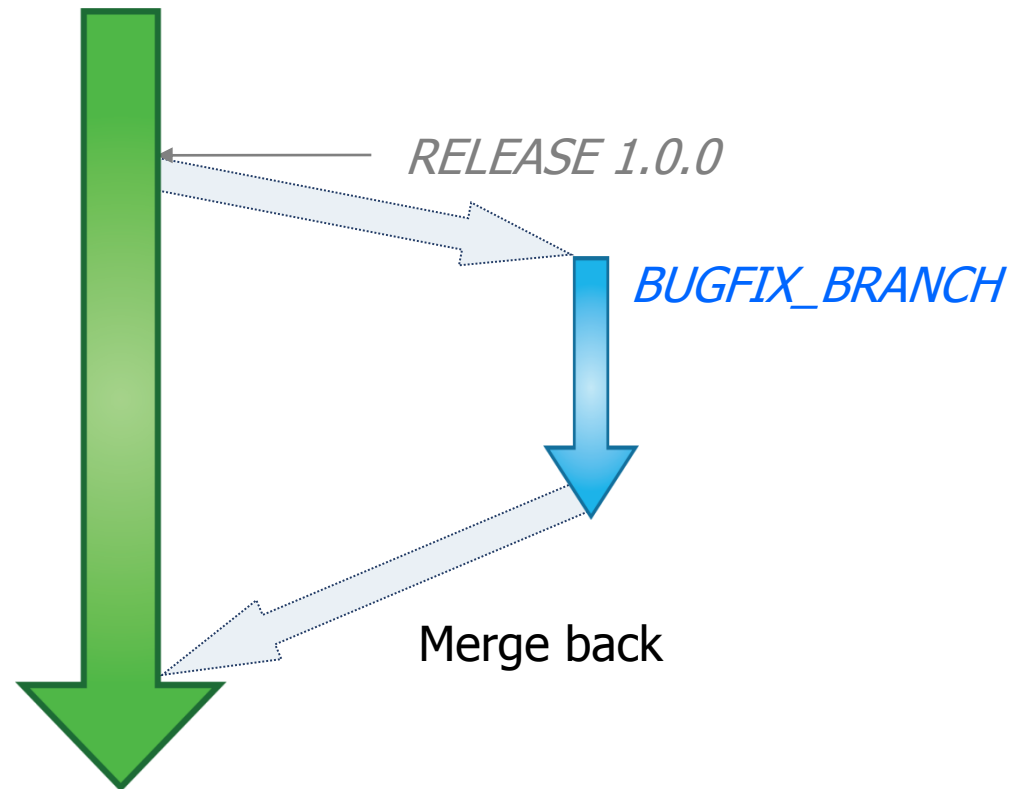
Branching Strategies - Platform Branches

Branches will live as long as the platform is supported

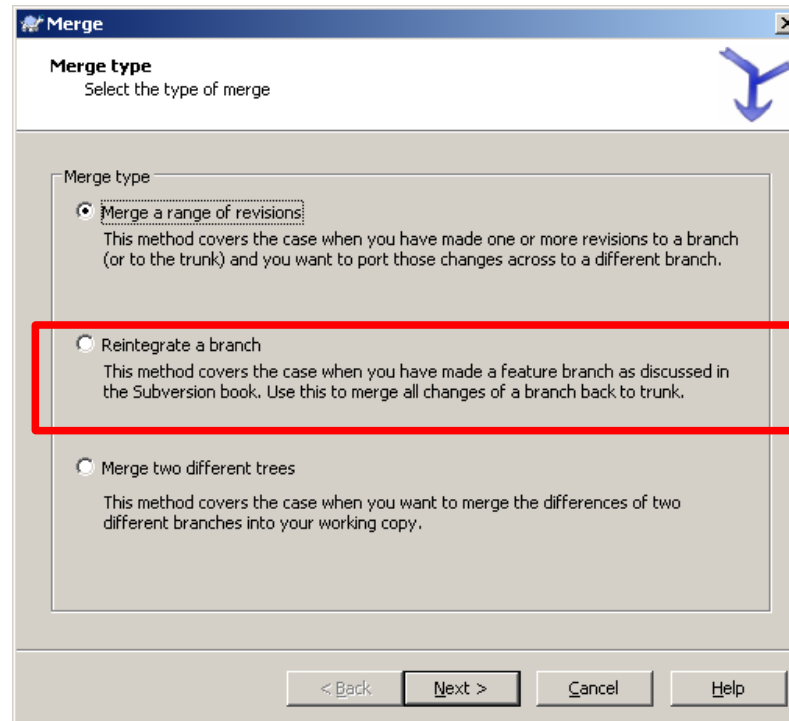
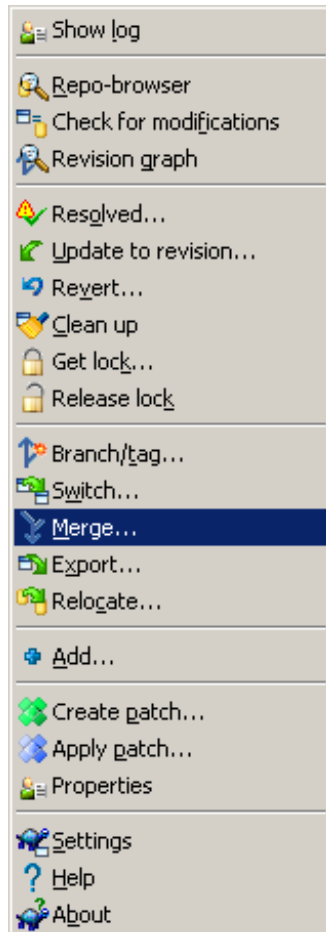


Merging - Merging from a branch

- ❑ What's with the bug you've fixed on the bug-fix-branch?
- ❑ What's about your current development?
- ❑ You have to merge the changes made in the branch back to the main line.



Merging features

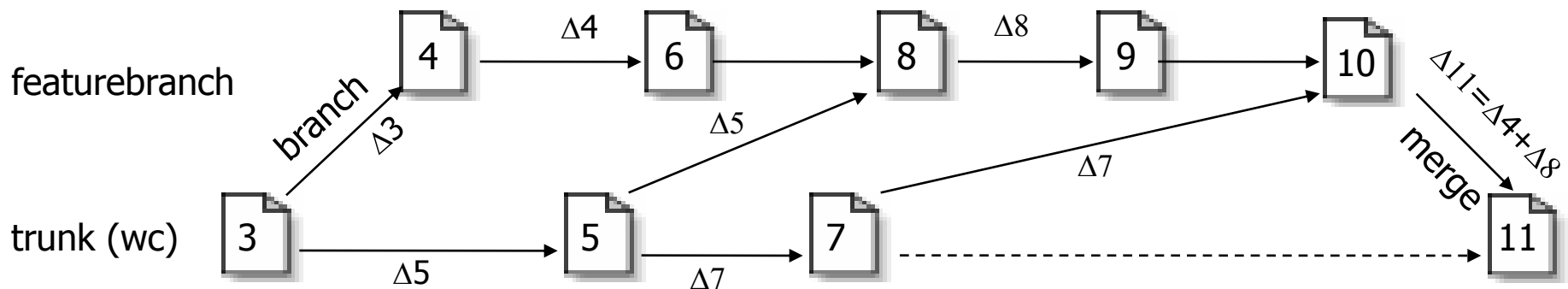


← reintegrate branch

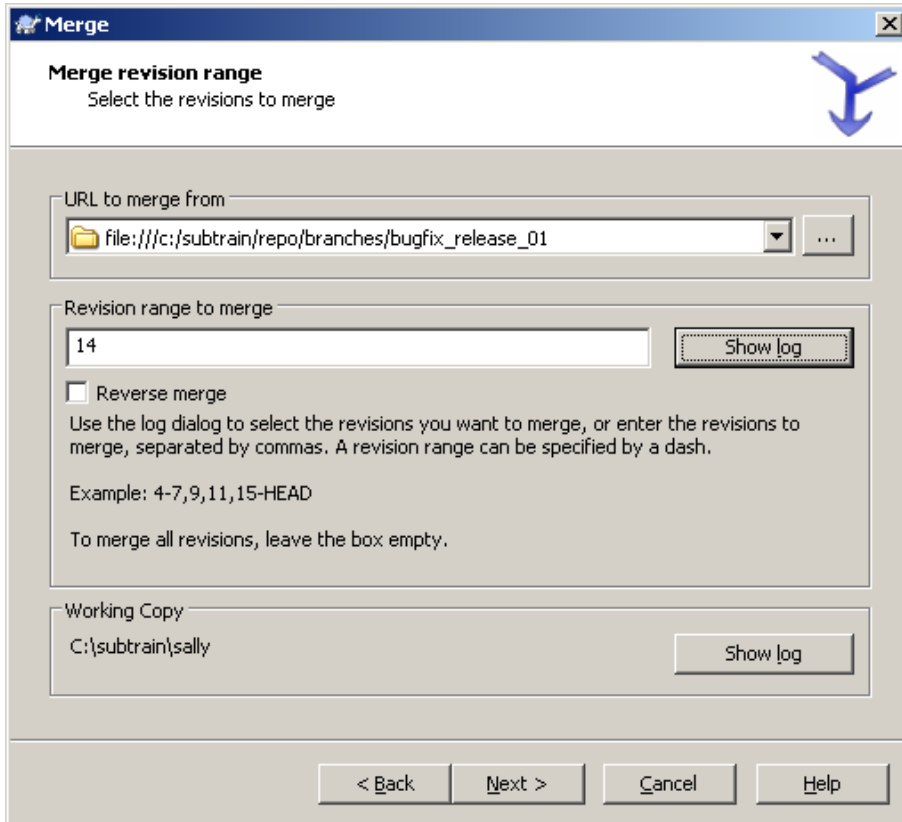
Reintegrate branch



- Reintegrate branch will keep track of all previous merged revisions and will not try to merge them back
- Conditions apply:
 - No switched working copy
 - All revisions from trunk has to be merged into branch before (so example below will not work if only 5 or 7 is merged to branch)



Merging a range of revisions



Merge

Merge revision range
Select the revisions to merge

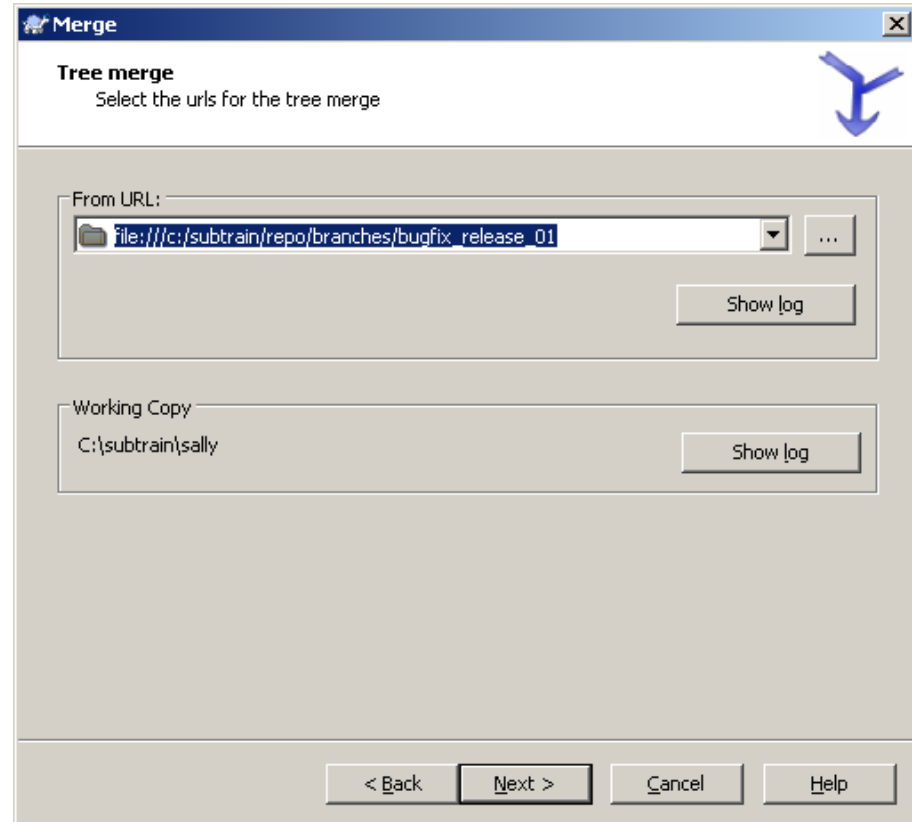
URL to merge from
file:///c:/subtrain/repo/branches/bugfix_release_01

Revision range to merge
14 [Show log](#)

☐ Reverse merge
Use the log dialog to select the revisions you want to merge, or enter the revisions to merge, separated by commas. A revision range can be specified by a dash.
Example: 4-7,9,11,15-HEAD
To merge all revisions, leave the box empty.

Working Copy
C:\subtrain\sally [Show log](#)

< Back Next > Cancel Help



Merge

Tree merge
Select the urls for the tree merge

From URL:
file:///c:/subtrain/repo/branches/bugfix_release_01 [Show log](#)

Working Copy
C:\subtrain\sally [Show log](#)

< Back Next > Cancel Help

Merge two different trees

Merge

Tree merge
Select the urls for the tree merge

From: (start URL and revision of the range to merge)

file:///c:/subtrain/repo/branches/bugfix_release_01

☐ HEAD Revision

☒ Revision 14 Show log

To: (end URL and revision of the range to merge)

file:///c:/subtrain/repo/branches/bugfix_release_01

☐ HEAD Revision

☒ Revision 12 Show log

Working Copy

C:\subtrain\sally Show log

< Back Next > Cancel Help

Merge

Merge options
Select the merge options

Merge options

Merge depth: Working copy

☐ Ignore ancestry

☐ Ignore line endings

☒ Compare whitespaces

☐ Ignore whitespace changes

☐ Ignore all whitespaces

☐ Only record the merge

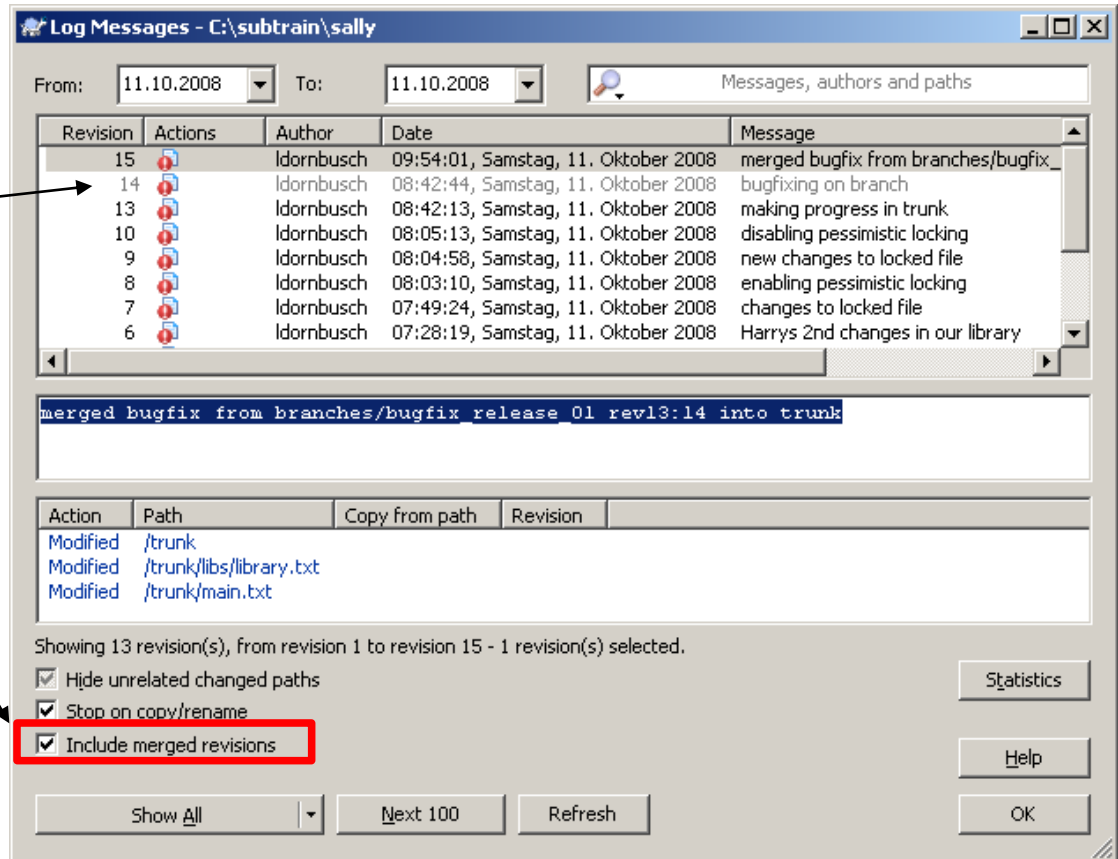
Test merge

< Back Merge Cancel Help

Merge tracking in svn log



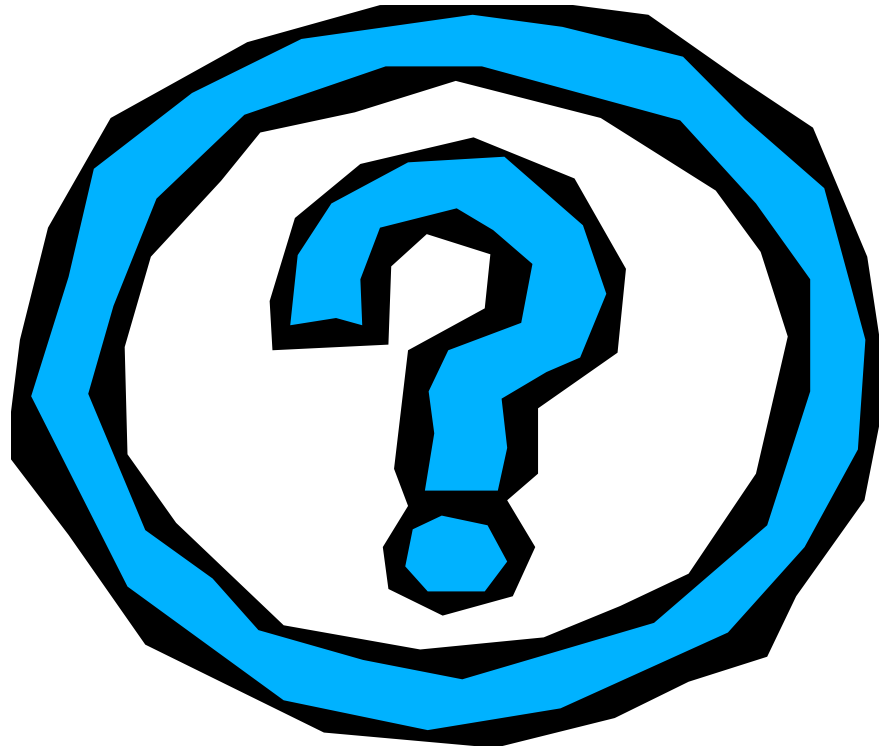
merge tracking



Best Practices - Merging

- ❑ Self-manage the merging, do not let TortoiseSVN to merge files
 - In case there is conflict between your working copy and the repository, you should merge file by yourself.
 - Steps:
 - Perform Update => TortoiseSVN creates 3 files: your original file, the latest file from repository and the SVN merged file.
 - Don't use the merged file.
 - Manually compare and merge your original file with latest file.
 - Overwrite your result onto the SVN merged file.
 - Commit your merged file to the repository.
 - After you've merged commit the changes and provide a log message with information on which revision/branch you have merged (merge tracking).

Question and Answer



Q&A (1)

- ❑ How to obtain a “clean” working copy (original sources only) and exclude .svn folders.
 - Select working copy>right-click>TortoiseSVN>Export
 - Provide the destination folder for the “clean” working copy

- ❑ How to leave (ignore) the files/folders not in source control
 - Select files/folders in working copy not in source control
 - Right-click>TortoiseSVN>Add to ignore list

- ❑ How to add “free” files/folders (in working copy) into source control
 - Select files/folders>right-click>TortoiseSVN>Add
 - Commit the working copy

Q&A (2)

❑ How to break/steal lock

- This is to break the locking on file by override the locking author.
- Check for modifications>select locked files>break lock, OR
- Right click>TortoiseSVN>Get lock, check the Steal option, OK.
- **Recommend: should not use this function unless you have agreement of the PM.**

❑ Cannot login because of wrong SSL cache: clear cache

- First login to a wrong/forbidden link failed => wrong SSL is cached
- Later login (to correct link) also failed because the TortoiseSVN use the wrong cached SSL.
- Solution: right click>TortoiseSVN>Settings>Saved data>Authentication Data>Clear

Q&A (3)

- ❑ How to check out an SVN directory (recursively) to local existing folder? (Similar to get latest version in VSS & overwrite all local)
 - Should checkout to an empty working copy.
 - Existing working copy may cause file conflict or leave files not under source control.
- ❑ How to get specific version of code from SVN?
 - Select file/folder in the working copy
 - TortoiseSVN>Update to revision...
 - Select the option Revision, input desired revision
 - You can Show log to know the log message.
- ❑ How to compare code from working copy and SVN before committing to SVN?
 - In the working copy, select the file that has changed from the last update.
 - Right click>TortoiseSVN>Diff
 - The TortoiseSVN display a dialog which shows two (clear text) files in vertical split.

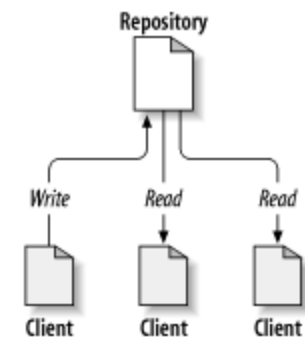
Appendix

- ❑ The repository
- ❑ What is Subversion?
- ❑ Versioning models
- ❑ Subversion in actions

Appendix - The repository

- ❑ The repository is a central store of data.
- ❑ The repository stores information in the *file system tree*, a kind of file server.

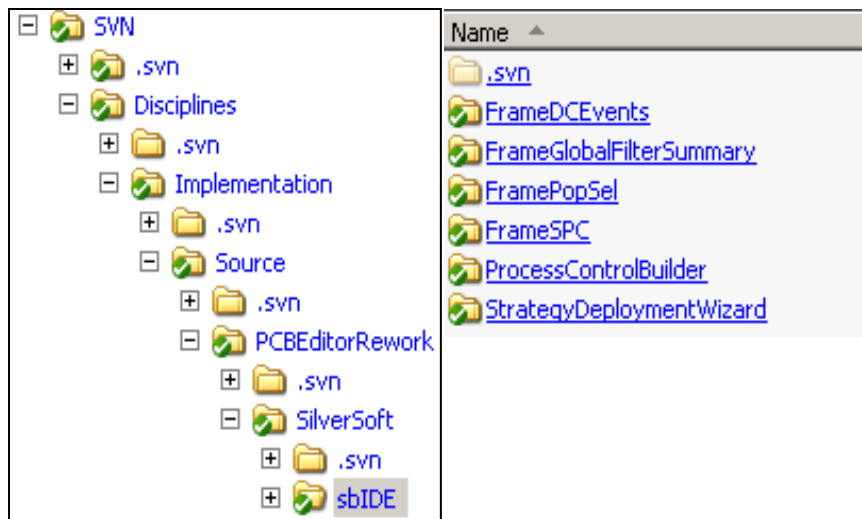
| File | Extension | Revision | Author | Size | Date | Lock |
|---|-----------|----------|------------------|------|---------------------|------|
| ⊟ http://subversion/svnsla/Maestria_Centric | | | | | | |
| ⊟ Branches | | 286 | fabriceb | | 20/11/2006 16:53:53 | |
| ⊕ 6.1.4 | | 137 | si_gen\nicolasl | | 08/06/2006 17:33:24 | |
| ⊕ 6.2.0 | | 136 | florentc | | 08/06/2006 17:28:49 | |
| ⊕ 6.2.1 | | 221 | fabriceb | | 25/10/2006 10:39:15 | |
| ⊕ 6.3.0 | | 280 | si_gen\nflorentc | | 14/11/2006 17:30:38 | |
| ⊟ 6.3.1 | | 275 | si_gen\nflorentc | | 13/11/2006 15:24:51 | |
| ⊕ Database | | 247 | SI_GEN\nNico... | | 31/10/2006 15:49:39 | |
| ⊕ WebApplications | | 275 | si_gen\nflorentc | | 13/11/2006 15:24:51 | |
| ⊕ WindowsApplications | | 220 | fabriceb | | 24/10/2006 15:41:01 | |
| ⊕ XML | | 76 | Si_Gen\nNicol... | | 31/01/2006 17:25:26 | |
| ⊕ Temporary | | 286 | fabriceb | | 20/11/2006 16:53:53 | |
| ⊕ User_Branches | | 126 | fabriceb | | 16/05/2006 18:50:06 | |
| ⊕ Tags | | 1 | fabriceb | | 19/12/2005 10:54:16 | |
| ⊟ Trunk | | 289 | gcs | | 21/11/2006 13:52:14 | |
| ⊕ Database | | 284 | fabriceb | | 17/11/2006 15:38:08 | |
| ⊕ WebApplications | | 289 | gcs | | 21/11/2006 13:52:14 | |
| ⊕ WindowsApplications | | 287 | fabriceb | | 21/11/2006 10:31:54 | |
| ⊕ XML | | 76 | Si_Gen\nNicol... | | 31/01/2006 17:25:26 | |



- ❑ When a client reads data from the repository, it normally sees only the latest version of the file system tree.
- ❑ The client also has the ability to view previous states of the file system.

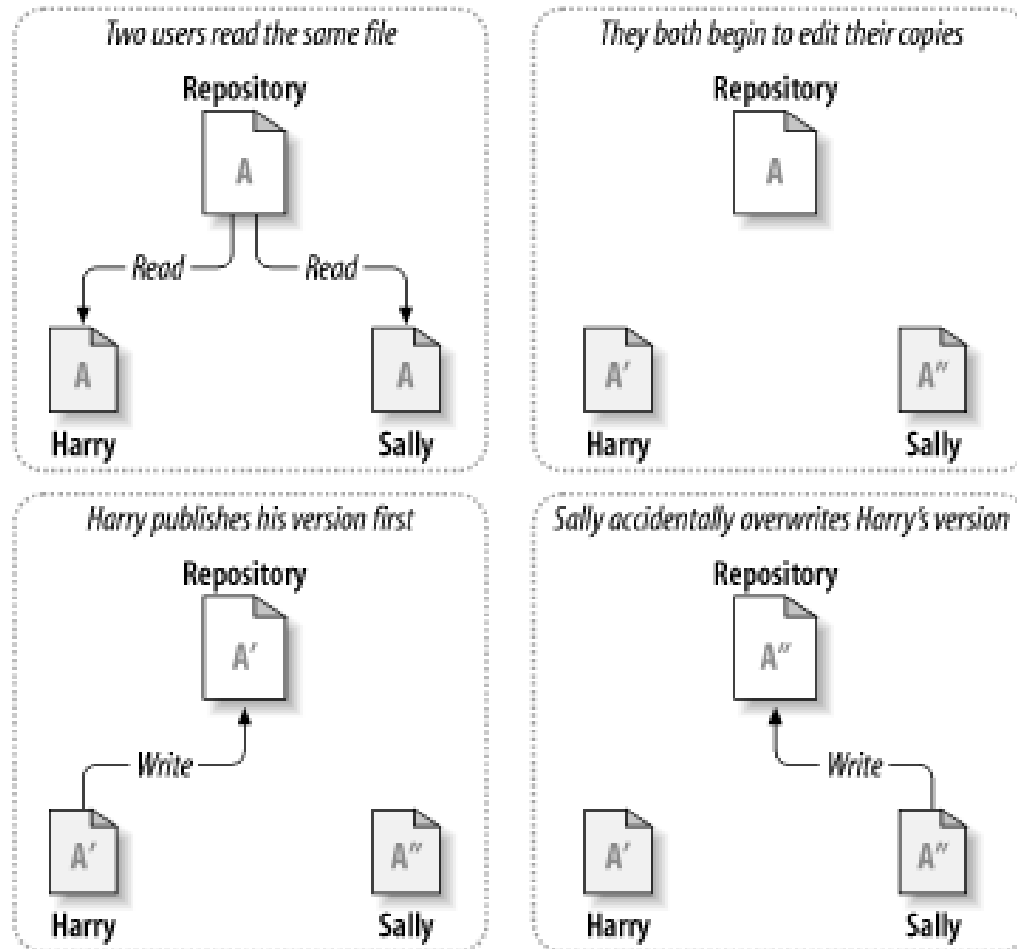
Appendix - What is Subversion?

- ❑ Subversion (abbreviated SVN) is a free open-source version control system.
- ❑ Subversion manages files and folders across time and location.
- ❑ Managed folders and files are placed into a *repository*. A *repository* is much like an ordinary file server.
- ❑ Subversion allows you to recover old versions of your data, or examine the history of how your data changed.



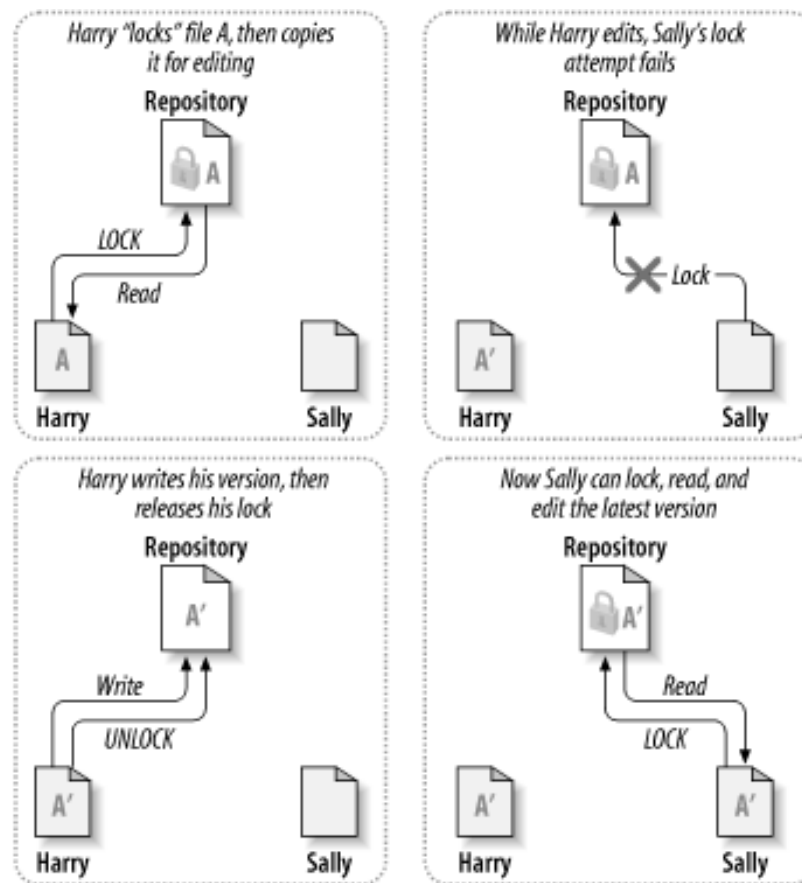
Appendix - Versioning models

- Problem of file-sharing without control: one's data may be accidentally overwritten by others.



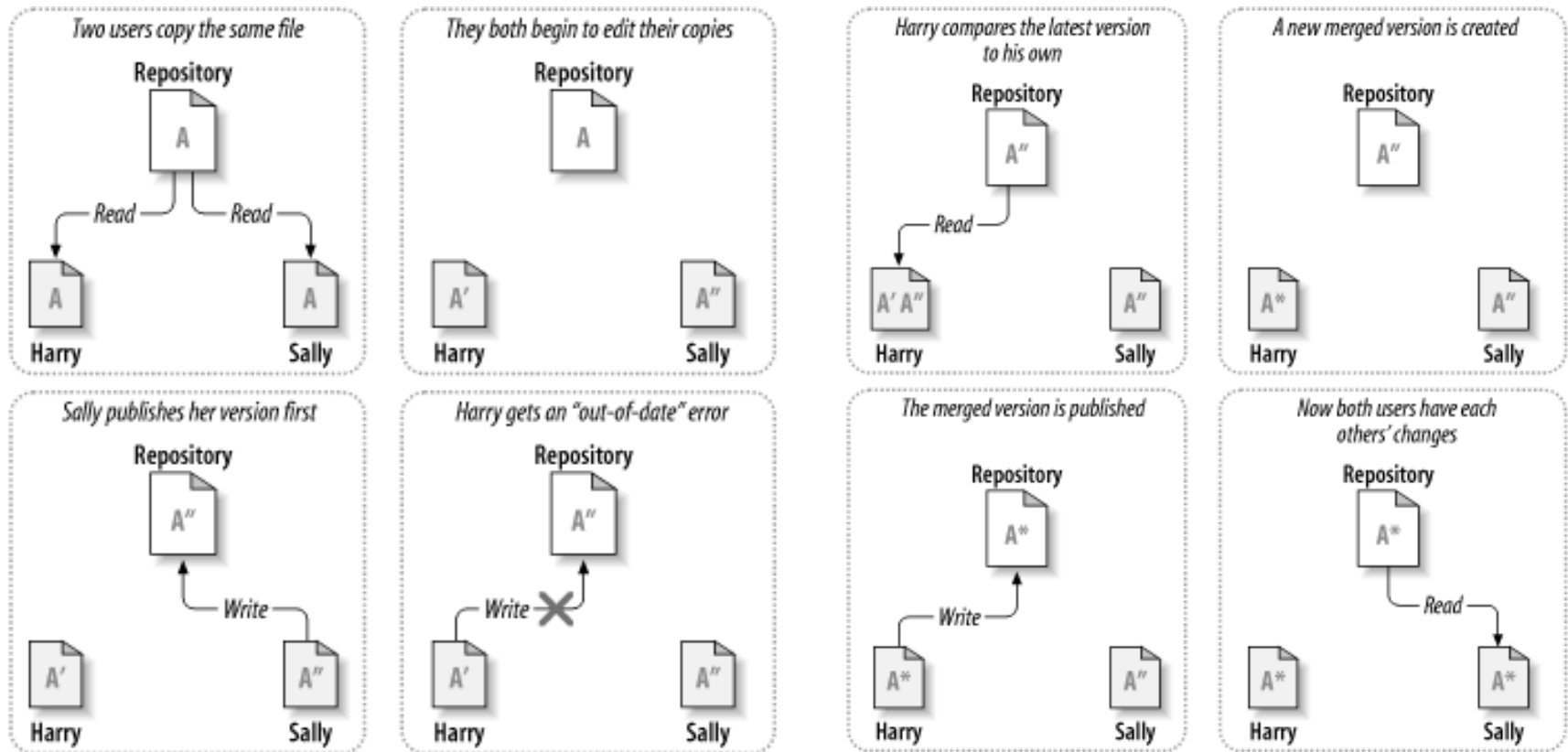
Appendix - Versioning models

- ❑ The Lock-Modify-Unlock solution:
- ❑ It is helpful if this solution is applied for files with binary formats
- ❑ Weak point is **not allow** many persons to work on the same file at the same time.



Appendix - Versioning models

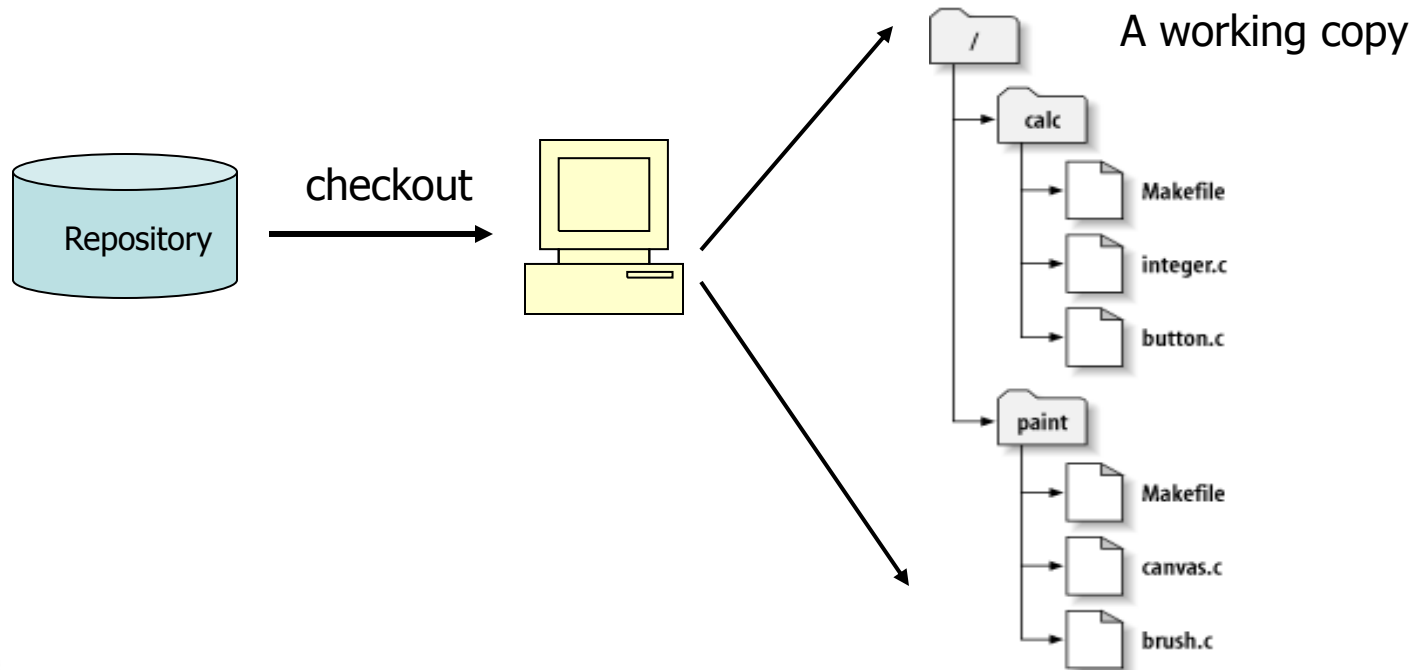
- ❑ The Copy-Modify-Merge solution: allow many persons to work on the same file at the same time, but must apply file merging before submitting.



Appendix - Subversion in actions

□ Working copy

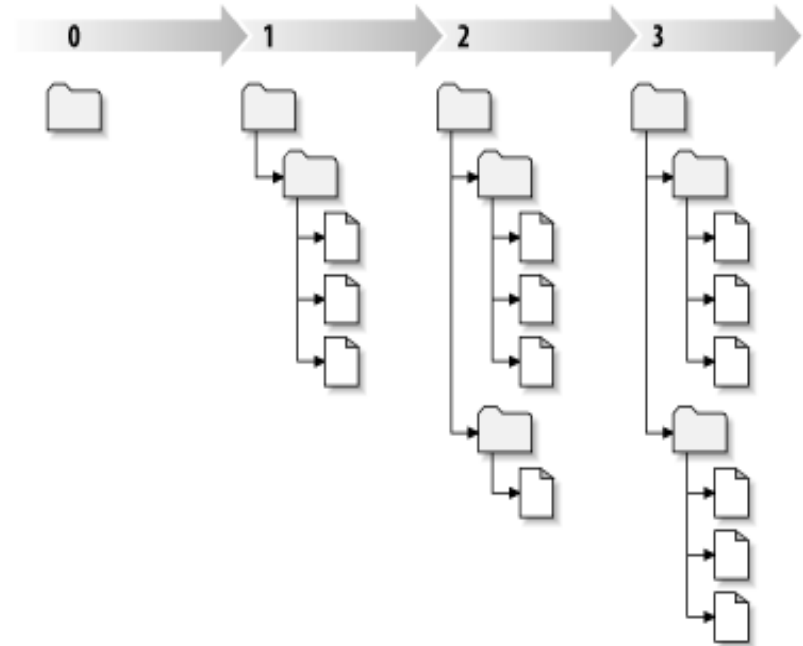
- A Subversion working copy is an ordinary directory tree on your local system.
- The working copy is created by *checkout* (getting) one or many folders/files on the repository and put them on the local computer.
- Subversion will never incorporate other people's changes, nor make your own changes available to others, until you explicitly tell it to do so.



Appendix - Subversion in actions

❑ Revision

- An *svn commit* can publish changes to many files and directories as a single atomic transaction.
- Each time the repository accepts a commit, this creates a new state of the file system tree, called a *revision*.
- Each *revision* is assigned a unique naturally increasing number (n , $n+1$, $n+2$...)



Revision number is assigned for the whole repository, not for each folder.

Appendix - Subversion in actions

- ❑ How working copy tracks the repository
 - What revision your working file is based on (*working revision*).
 - A timestamp recording when the local copy was last updated by the repository.
- ❑ Base on this information, Subversion can tell which of the following four states a working file is in:
 - Unchanged, and current: no change in working copy, no change to that file in the repository
 - Locally changed, and current: changed in working copy, no change to that file in the repository
 - Unchanged, and out-of-date: no change in working copy, changed to that file in the repository
 - Locally changed, and out-of-date: changed in working copy and repository.

References

- ❑ Book: Version Control with Subversion:
 - <http://svnbook.red-bean.com/>
- ❑ Subversion Developer Portal and downloads:
 - <http://subversion.tigris.org/>
- ❑ Subversion Wiki:
 - <http://www.subversionary.org>
- ❑ Subversion Free Tools:
 - <http://subversion.tigris.org/links.html>
 - <http://www.polarion.org>
- ❑ Subversion Forums:
 - <http://www.svnforum.org/>
- ❑ Subversion News
 - <http://svn.haxx.se/>

Thank you for your attending

Appendix: Course detail form

| | | | |
|-----------------|----------------------|-----------------|------------------------------|
| Author | Binh Le | Duration | 3 hours |
| Category | Software Engineering | Type | Theory / Theory and Practice |

| | |
|---|---|
| Examination | Practice on frequently questions related to using SVN in projects |
| Intended Audience | Any software technician, engineer, project leader, project manager, manager who are working in software related projects. |
| Pre-requisites | N/a |
| Completion criteria for the course | Attendee must join at least 80% of course length. |
| Criteria for granting training waivers | Those who has experience on using this tool in previous projects |