Configuration and Build Management

- Purpose of Software Configuration Management (SCM)
- Software Configuration Management Activities
- Outline of a Software Configuration Management Plan
- SVN and GIT in more detail

CE202 Software Engineering, Autumn term
Dr Michael Gardner, School of Computer Science and Electronic Engineering, University of Essex

Why Software Configuration Management ?

- The problem:
  - Multiple people have to work on software that is changing
  - More than one version of the software has to be supported:
    - Released systems
    - Custom configured systems (different functionality)
    - System(s) under development
    - Software on different machines & operating systems
- *Need for coordination*
- Software Configuration Management
  - manages evolving software systems
  - controls the costs involved in making changes to a system.

What is Software Configuration Management?

- Definition Software Configuration Management:
  - A set of management disciplines within a software engineering process to develop a baseline
  - Software Configuration Management encompasses the disciplines and techniques of initiating, evaluating and controlling change to software products during and after a software project
- Standards (approved by ANSI)
  - IEEE 828: Software Configuration Management Plans
  - IEEE 1042: Guide to Software Configuration Management.

Baseline: "*A specification or product that has been formally reviewed and agreed to by responsible management, that thereafter serves as the basis for further development, and can be changed only through formal change control procedures.*"

Configuration Management Activities (1)

- Software Configuration Management Activities:
  - Configuration item identification
  - Promotion management
  - Release management
  - Branch management
  - Variant management
  - Change management
- No fixed order:
  - These activities are usually performed in different ways (formally, informally) depending on the project type and life-cycle phase (research, development, maintenance).

Possible Selection of Configuration Items

- Problem Statement
- Software Project Management Plan (SPMP)
- Requirements Analysis Document (RAD)
- System Design Document (SDD)
- Project Agreement
- Object Design Document (ODD)
- Dynamic Model
- Object model
- Functional Model
- Unit tests
- Integration test strategy
- Source code
- API Specification
- Input data and data bases
- Test plan
- Test data
- Support software (part of the product)
- Support software (not part of the product)
- User manual
- Administrator manual

Configuration Management Roles

- Configuration Manager
  - Responsible for identifying configuration items
  - Also often responsible for defining the procedures for creating promotions and releases
- Change Control Board Member
  - Responsible for approving or rejecting change requests
- Developer
  - Creates promotions triggered by change requests or the normal activities of development. The developer checks in changes and resolves conflicts
- Auditor
  - Responsible for the selection and evaluation of promotions for release and for ensuring the consistency and completeness of this release.

Software Configuration Management Planning

- Software configuration management planning starts during the early phases of a project
- The outcome of the SCM planning phase is the *Software Configuration Management Plan (SCMP)* which might be extended or revised during the rest of the project
- The SCMP can either follow a public standard like the IEEE 828, or an internal (e.g. company specific) standard.

Outline of a Software Configuration Management Plan (SCMP, IEEE 828-2005)

1. Introduction
   - Describes the Plan's purpose, scope of application, key terms, and references
2. SCM management (WHO?)
   - Identifies the responsibilities and authorities for managing and accomplishing the planned SCM activities
3. SCM activities (WHAT?)
   - Identifies all activities to be performed in applying to the project
4. SCM schedule (WHEN?)
   - Establishes required coordination of SCM activities with other activities in the project
5. SCM resources (HOW?)
   - Identifies tools and physical and human resources required for the execution of the Plan
6. SCM plan maintenance
   - Identifies how the Plan will be kept current while in effect

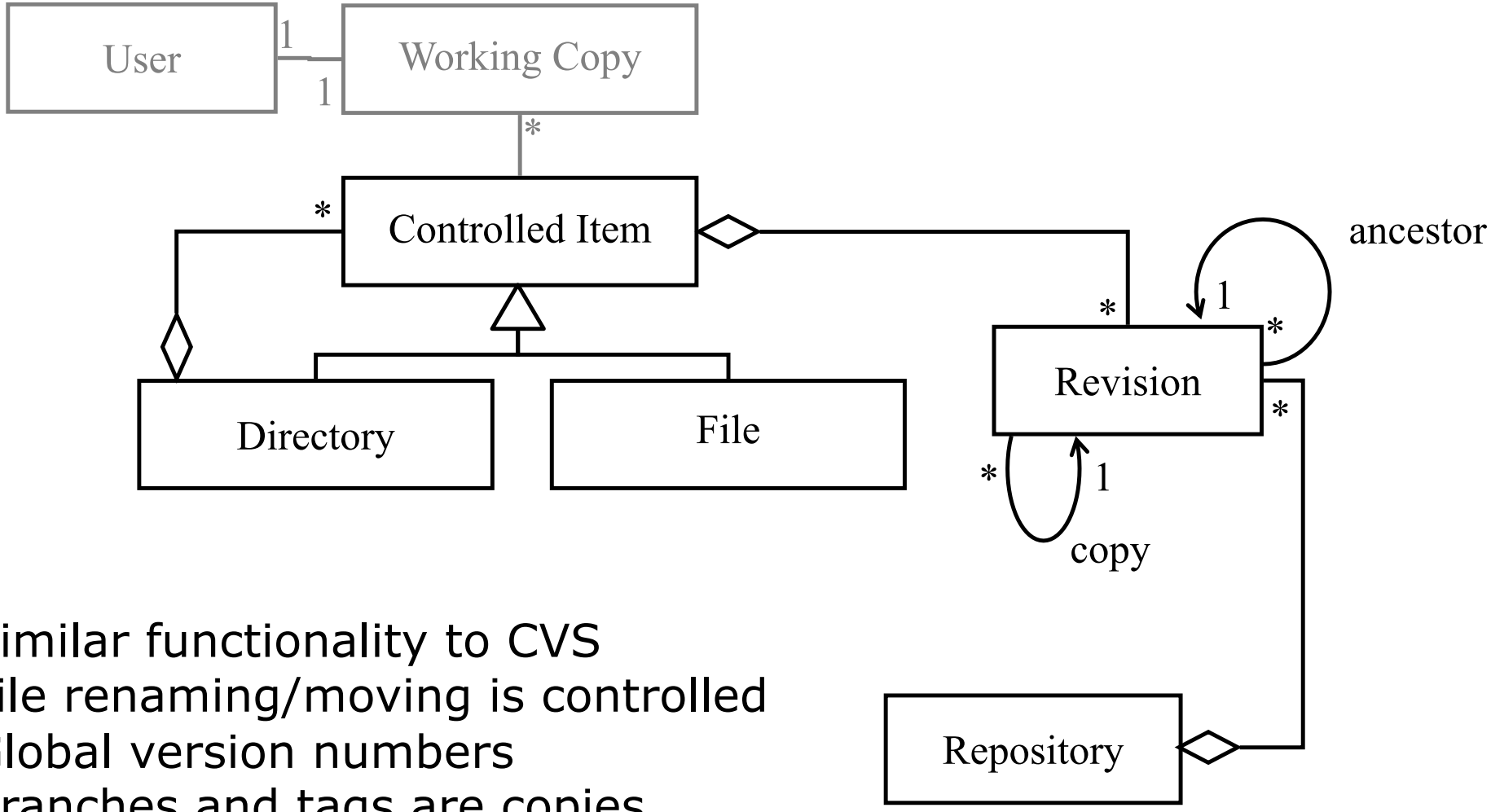
Software Configuration Management Tools

- RCS: The first on the block [Tichy 1975]
- CVS (Concurrent Version Control)
  - based on RCS, allows concurrency without locking
  - http://www.nongnu.org/cvs/Subversion
  - Based on CVS
  - Open Source Project (http://subversion.apache.org)
- Perforce
  - Repository server, keeps track of developer's activities
  - http://www.perforce.com
- ClearCase
  - Multiple servers, process modeling, policy check mechanisms
  - https://www.ibm.com/uk-en/marketplace/rational-clearcase
- GIT

Subversion

- Open Source Project http://subversion.tigris.org/
- Based on CVS
  - Subversion interface and features similar to CVS
  - Commands: checkout, add, delete, commit, diff
- Differences to CVS
  - Version controlled moving, renaming, copying of files and directories
  - Version controlled metadata of files and directories
- Server Options
  - Standalone installation
  - Integrated into the Apache webserver

Subversion Model (UML Class Diagram)

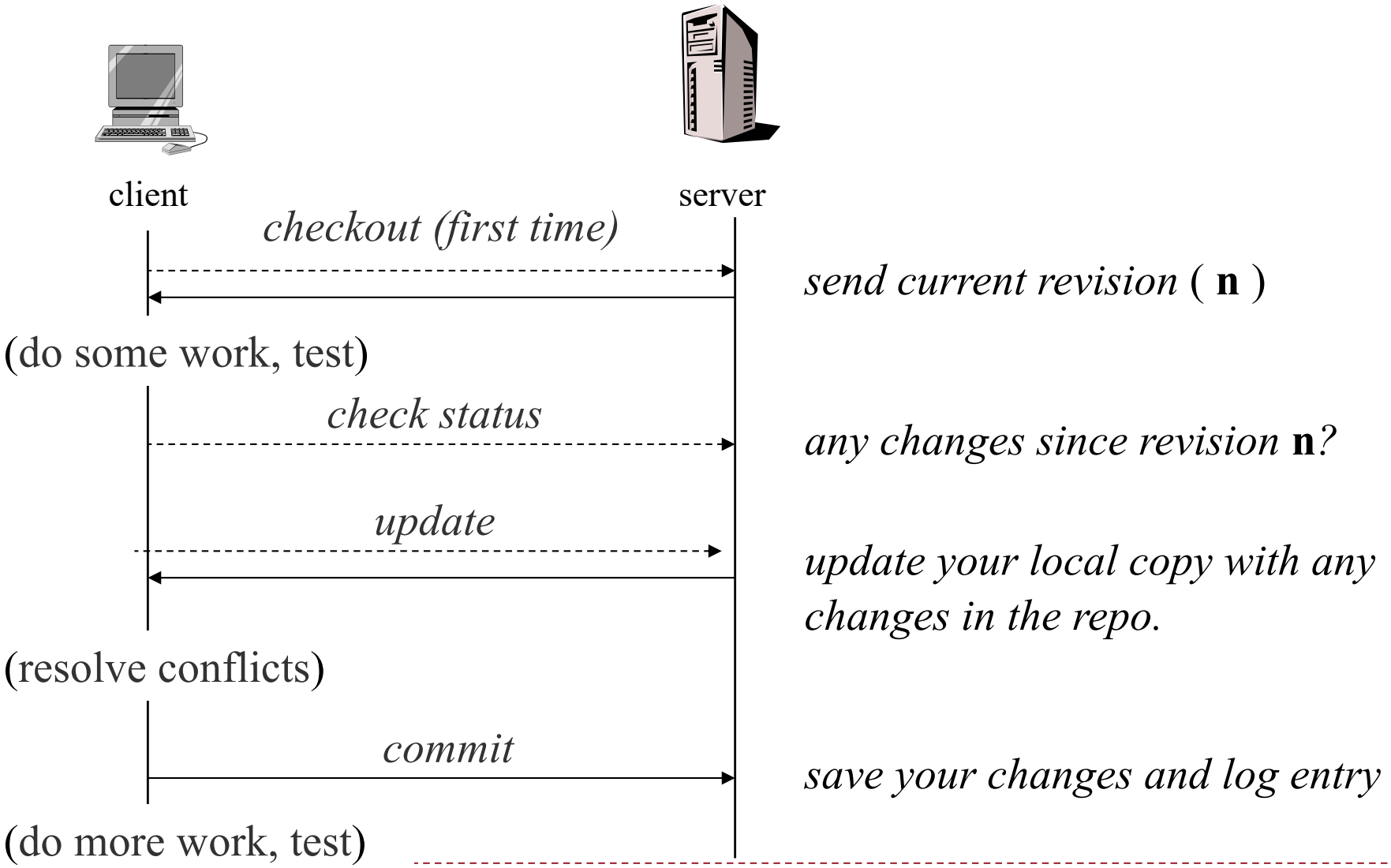


Similar functionality to CVS
File renaming/moving is controlled
Global version numbers
Branches and tags are copies
(can you spot the composite design pattern in this diagram?)

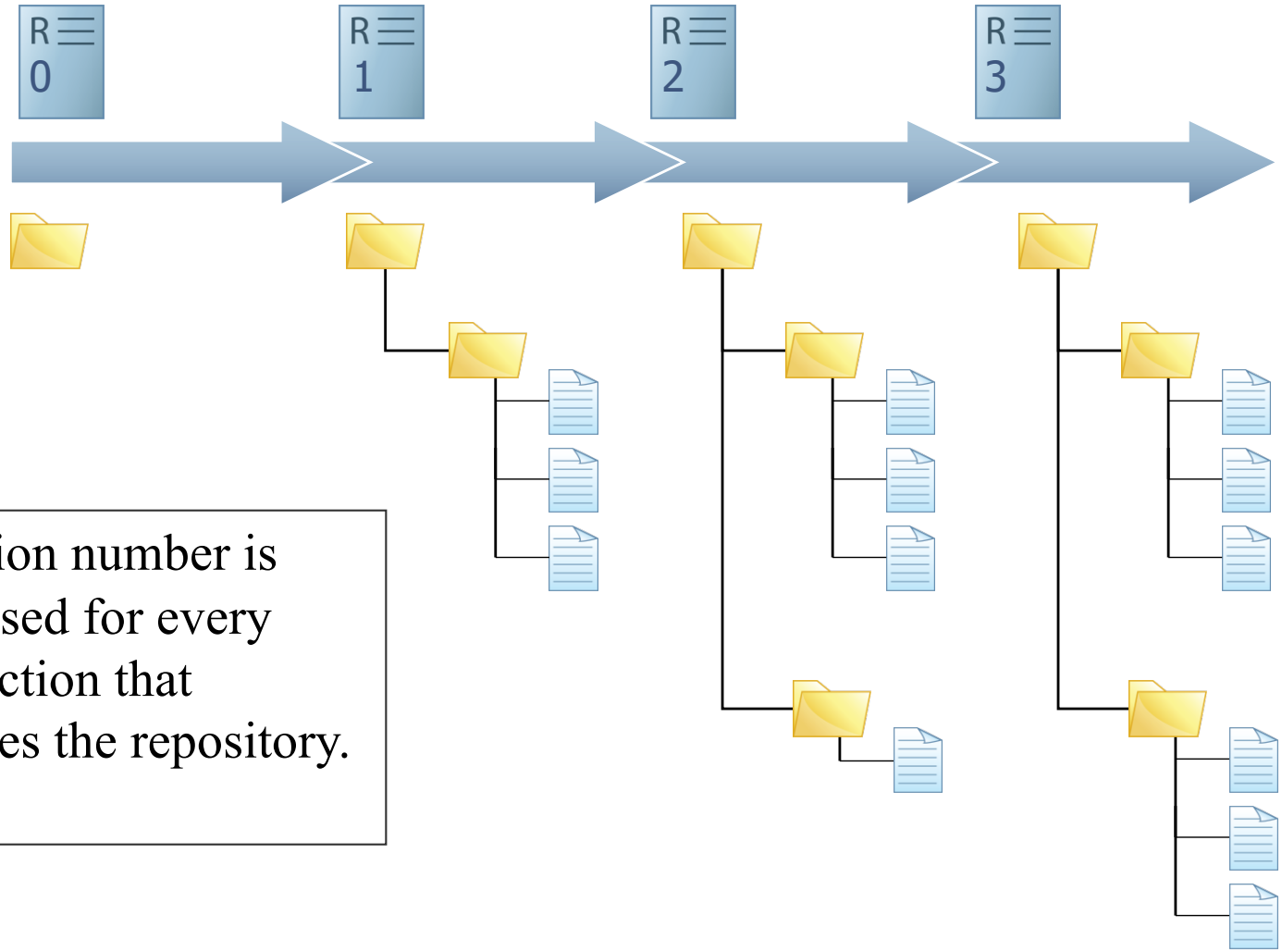
What is version control in SVN?

- manage documents ***over time***
- keep a history of all changes - multiple versions of every file
- coordinate work of multiple authors
- avoid conflicts …and help *resolve* them
- permissions: authenticate and control access to files
- show differences between versions of a file
- document changes -- reason for change

How to Use Version Control



Revision numbers

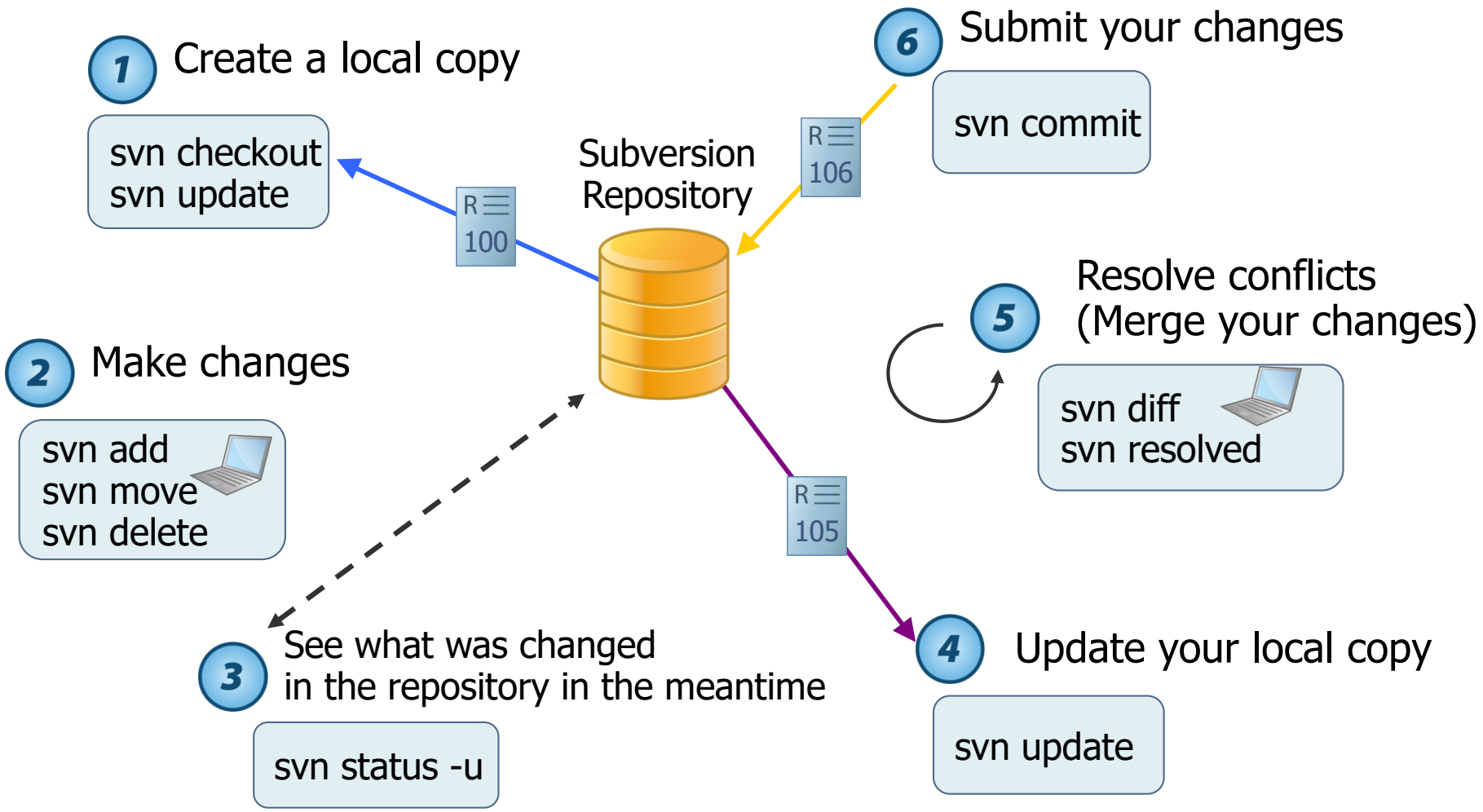


Revision number is increased for every transaction that changes the repository.

Properties of a Repository

- History of *all changes to files and directories.*
  - you can recover any previous version of a file
  - remembers "moved" and "deleted" files
- Access Control
  - Read / write permission for users and groups
  - Permissions can apply to repo, directory, or file
- Logging
  - author of the change
  - date of the change
  - reason for the change

The Work Cycle



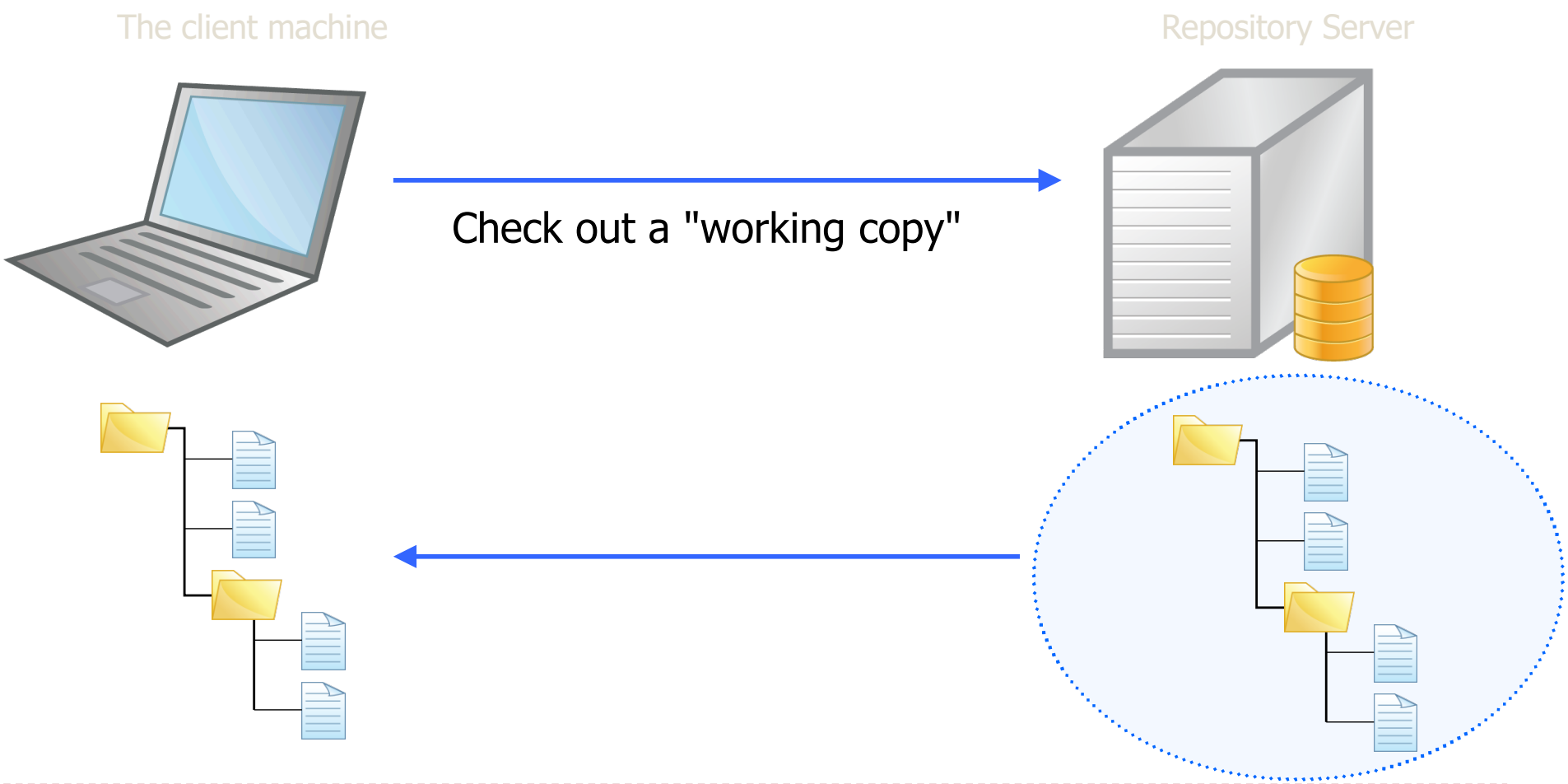
Logging a Revision

Content	what has changed?
Date	when did it change?
Author	who changed it?
Reason	why has it changed?

SVN does this

you enter this

(1) Check Out and the "Working Copy"



(2) Work Cycle: Edit Files

1. Edit files using anything you like.

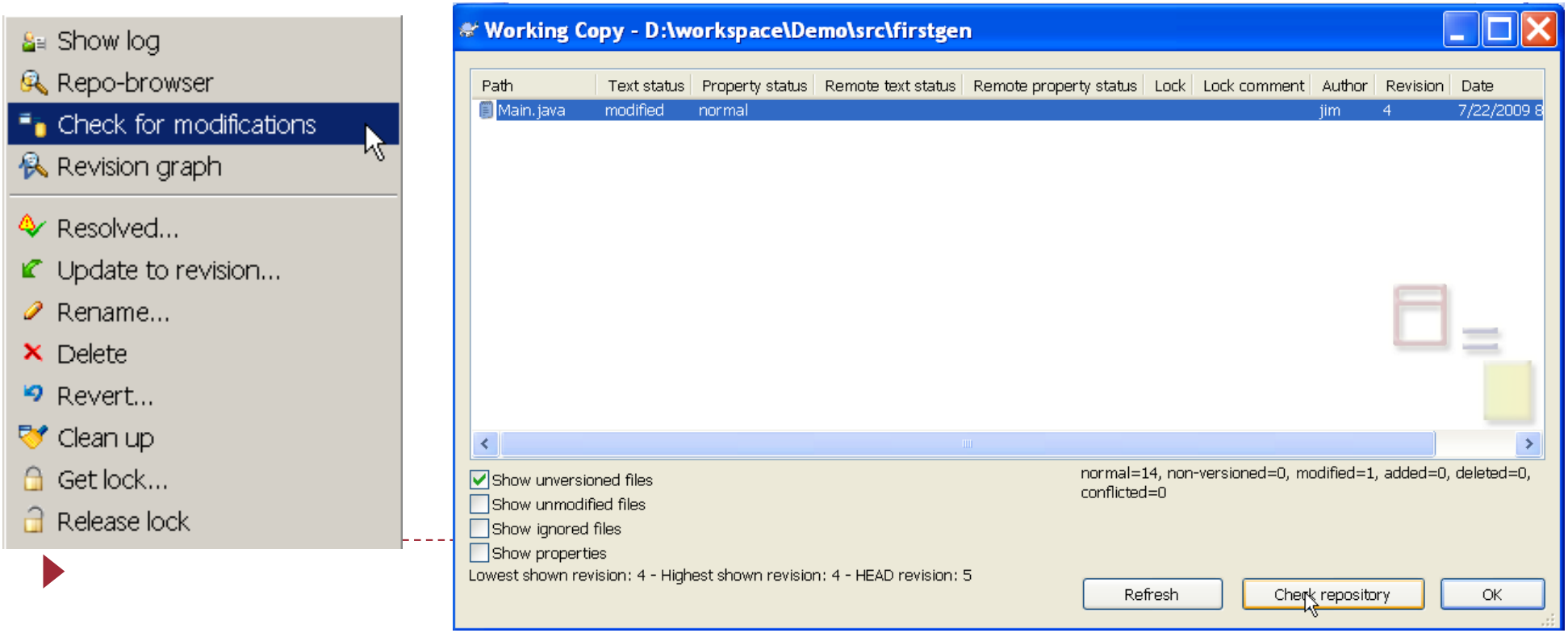
2. **Test Your Work.**

   - Don't commit buggy code to the repository.

Then go to next step…

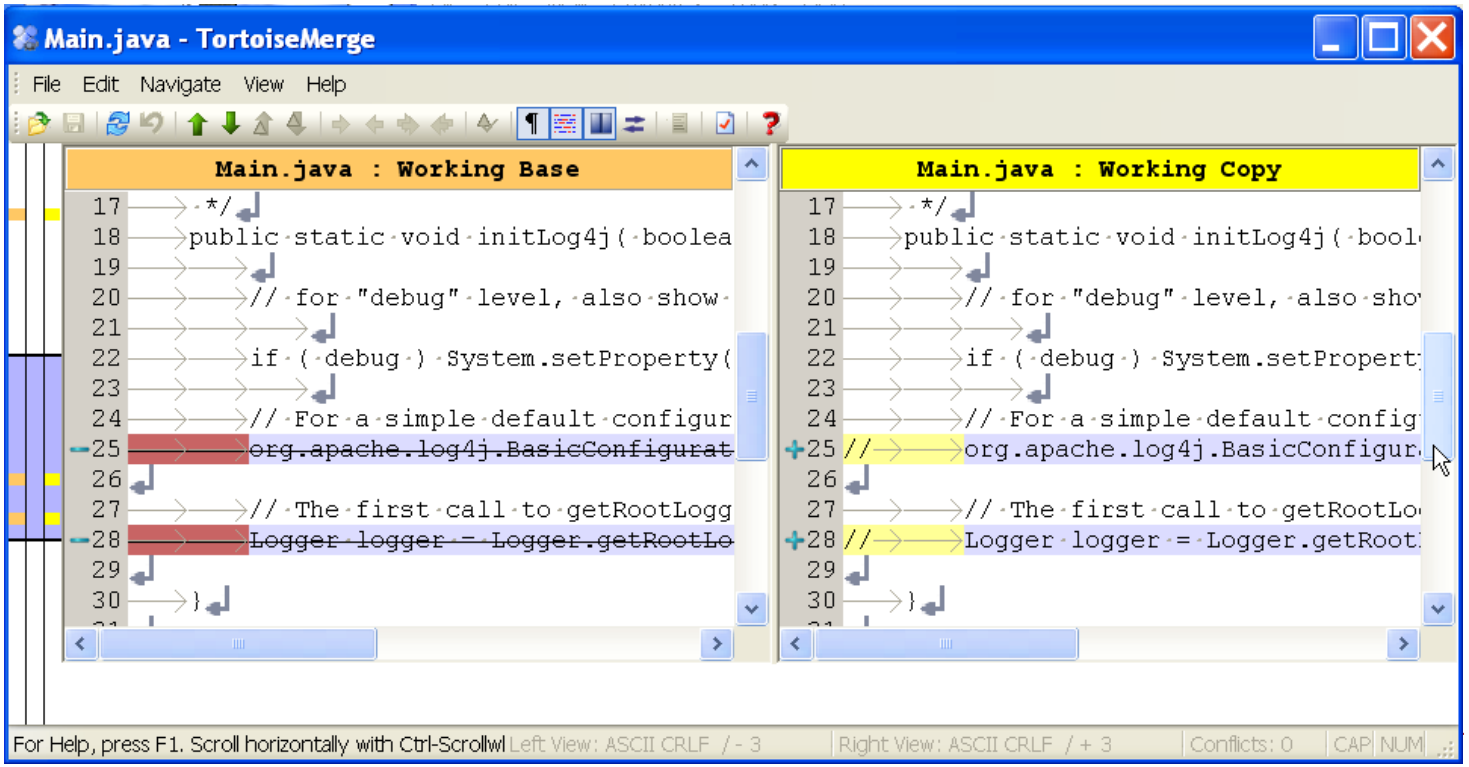
(3) Check for Updates

- Before "committing" your work, check for updates in the repository.
- Something might have changed while you were working.
- Subversion *requires* you to synchronize before commit.



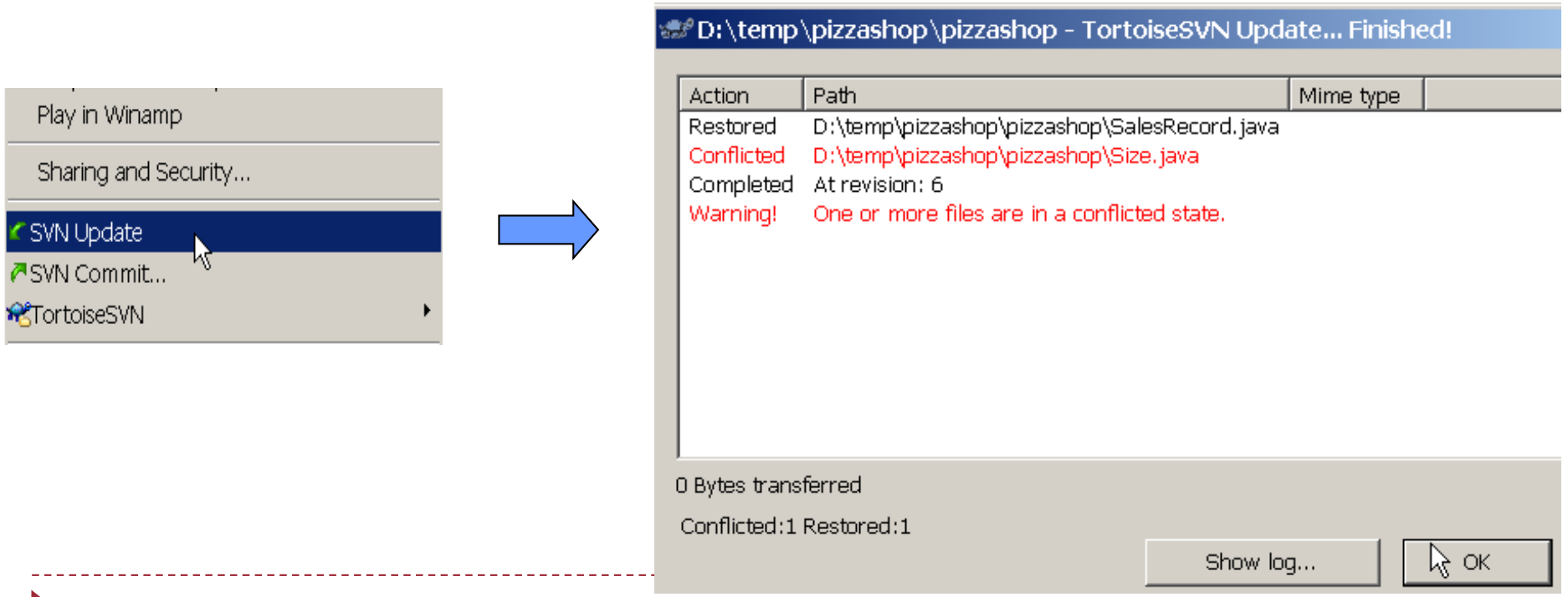
View Differences

- You can compare your version and the base or repo version.
- Select file, right-click => Compare with base



(4) Work Cycle: Update working copy

- If there are any changes on the server,
  then you should "update" your working copy before "commit"-ing your changes.



(5) Resolve Conflicts

- "Conflict" means you have made changes to a file, and the version in the repository has been changed, too.
- So there is a "conflict" between your work and work already in the repository.

Resolving Conflicts

The choices are:

(1) merge local & remote changes into one file.

(2) accept remote, discard your local changes.

(3) override remote, use your local changes.

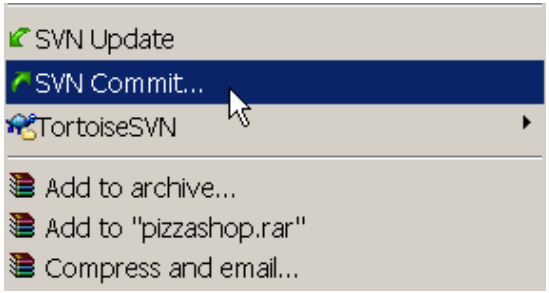
- After resolving all conflicts, mark the file as "resolved".
- Subversion will delete the extra copies.

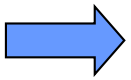
(6) Work Cycle: Commit

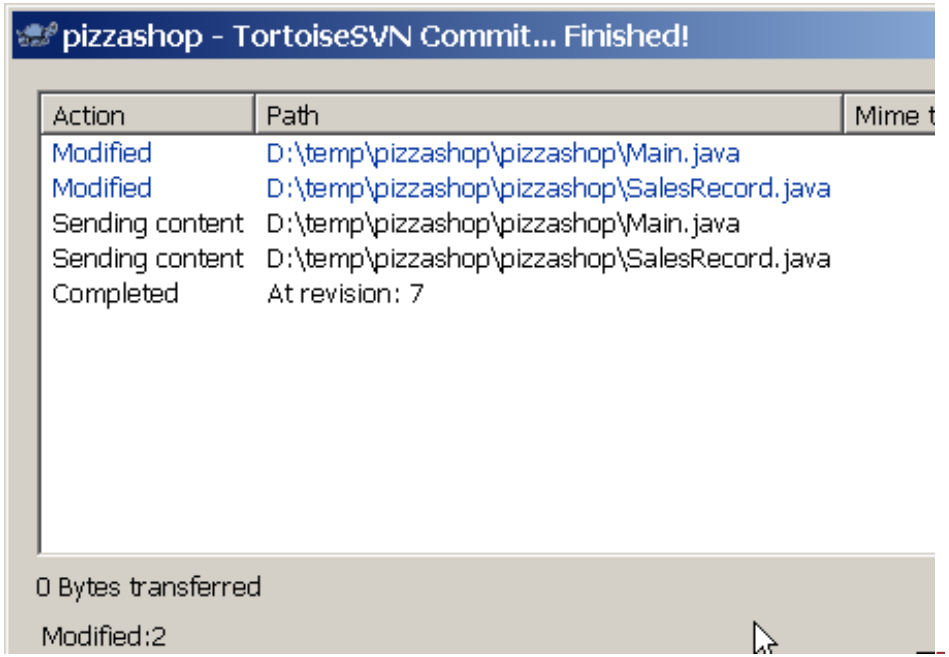
- After "update" and resolving conflicts, commit your work.
- Command line:

```
svn commit -m "description of your changes"
```

- TortoiseSVN:







Git vs SVN

- Git is *much* faster than SVN
  - Coded in C, which allows for a greater amount of optimization
  - Accomplishes much of the logic client side, thereby reducing time needed for communication
  - Developed to work on the Linux kernel, so that large project manipulation is at the forefront of the benchmarks

Git vs SVN

- Git is significantly smaller than SVN
  - All files are contained in a small decentralized .git file
  - In the case of Mozilla's projects, a Git repository was 30 times smaller than an identical SVN repository
  - Entire Linux kernel with 5 years of versioning contained in a single 1 GB .git file
  - SVN carries two complete copies of each file, while Git maintains a simple and separate 100 bytes of data per file, noting changes and supporting operations

Git vs SVN

- Git is more secure than SVN
  - All commits are uniquely hashed for both security and indexing purposes
  - Commits can be authenticated through numerous means
    - In the case of SSH commits, a key may be provided by both the client and server to guarantee authenticity and prevent against unauthorized access

Git vs SVN

- Git is decentralized
  - Each user contains an individual repository and can check commits against itself, allowing for detailed local revisioning
  - Being decentralized allows for easy replication and deployment
  - In this case, SVN relies on a single centralized repository and is unusable without

Git vs SVN

- Git is flexible
  - Due to it's decentralized nature, Git commits can be stored locally, or committed through HTTP, SSH, FTP, or even by Email
  - No need for a centralized repository
  - Developed as a command line utility, which allows a large amount of features to be built and customized on top of it

Git vs SVN

- Data Assurance
  - A checksum is performed on both upload and download to ensure sure that the file hasn't been corrupted.
  - Commit IDs are generated upon each commit
    - Linked list style of commits
    - Each commit is linked to the next, so that if something in the history was changed, each following commit will be rebranded to indicate the modification

Git vs SVN

- Branching
  - Git allows the usage of advanced branching mechanisms and procedures
  - Individual divisions of the code can be separated and developed separately within separate branches of the code
  - Branches can allow for the separation of work between developers, or even for disposable experimentation
  - Branching is a precursor and a component of the merging process

Git vs SVN

- Merging
  - Content of the files is tracked rather than the file itself
    - This allows for a greater element of tracking and a smarter and more automated process of merging
    - SVN is unable to accomplish this, and will throw a conflict if a file name is changed and differs from the name in the central repository
    - Git is able to solve this problem with its use of managing a local repository and tracking individual changes to the code

What is GitLab

- A web interface for Git
- Provides additional features on top of a Git repository
- Developed as a Github clone for self-hosting
- Allows for access to the repository from a web browser
- Issue and milestone tracking implemented
- Support for attachments and code snippets
- Integration of a wiki and wall for project documentation

Use of GitLab

- Issue Tracking
  - Can assign small parts of the project to team members through GitLab
  - This allows the group to know exactly what needs worked on
  - Bugs can also be submitted as an issue, and assigned to a particular developer to address
  - Issues can be closed in a ticketing like system to show which parts of the project have been completed
  - Milestones can be created with issues assigned to them, and a chosen due date applied

Use of GitLab

- Backup
  - "A file does not exist unless it is present in multiple geographical locations"
  - With a local repository on each developer machine, development server, and GitLab server, can have a great amount of data redundancy in the event of a failure, either software or hardware
  - The repository can be stored on the cloud in addition to using a local server
  - Able to pull old versions of the project as well

Summary

- Purpose of Software Configuration Management (SCM)
  - An essential activity in any project with more than 1 developer
- Software Configuration Management Activities
  - Configuration item identification
  - Promotion management
  - Release management
  - Branch management
  - Variant management
  - Change management
- Outline of a Software Configuration Management Plan
- Outline of using SVN and GIT

CE202 Module overview reminder

Autumn term

0. Introduction
1. Modelling notations
   - UML diagrams: use-case, class, interaction diagrams; type diagrams; statecharts
2. Requirements engineering
   - Use-case analysis
   - Object-oriented analysis
3. Software design
   - Principles of software design
   - Object-oriented design
   - Design patterns

Assignment no. 1: requirements and design

   - Software architecture

4. Software validation
   - Inspection
   - Typing
   - Testing
   - Design by Contract
5. Software reliability
   - Probability
   - Metrics
6. Software evolution
   - Laws of software evolution
   - Configuration management

Progress test

That completes the material for CE202!

- No lecture or class next week (week 11)
- Aiming to get the marks for assignment 1 completed by by Friday 13th December
- The progress test is on Wednesday 11th December, 12pm 5.300 B, EBS.2.2, LTB02, LTB04
- In the revision lecture next April I will be giving out hints for what to focus on for the exam – please ensure you attend!
- Have a good break!

Progress Test

- **Wednesday 11th December** 2019
- 12pm in 5.300 B, EBS.2.2, LTB02, LTB04 (check your schedule)
- 40 minutes, 20 multi-choice questions, 1 correct answer per question
- Worth 20%
- Please arrive at 11.50pm so we can start promptly
- Late arrivers will not be given extra time!
- Bring pencils and a rubber
- If you finish early remain seated and quiet until the end
- Covers everything we have covered this term

Progress Test: example question

Which of the following statements best describes the goal of 'Validation testing':

[A] To detect defects (reveal problems)

[B] To ensure the system meets the client's expectations

[C] To ensure that the system handles extreme values correctly

[D] To prove that there are no bugs in the system

The answer in this case is **[B]**

End

- See Bruegge chapter 13 for more detail on the topic of this lecture