配置管理系统TortoiseSVN

摘要

- 为什么我们需要配置管理?什么是svn和TortoiseSVN?
- 软件安装
- 名词解释
- 图标重载
- SVN的工作模式
- TortoiseSVN的功能演示
- 项目目录设置
- 人员分配与组织结构

为什么我们需要配置管理?

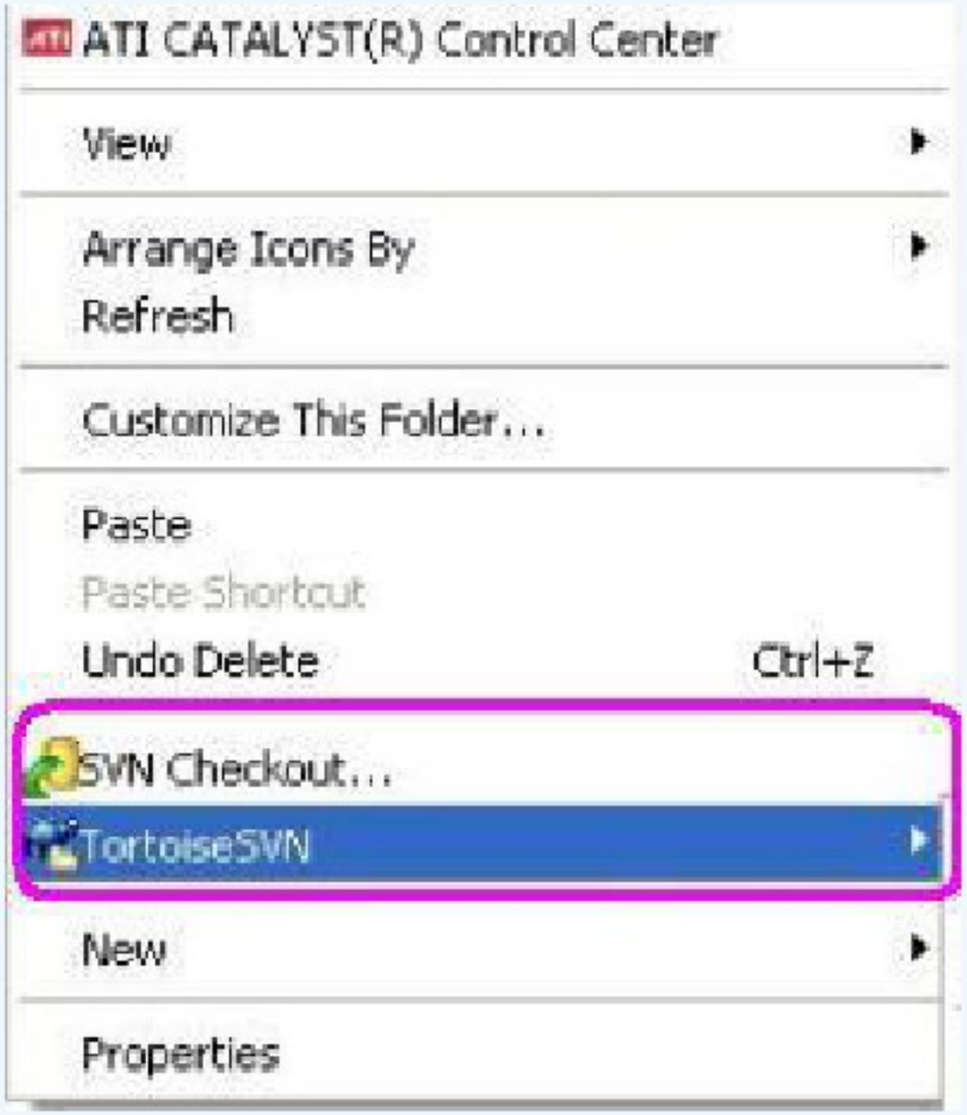
- 代码回归
- 多人协作开发
- 版本发布
  - 发布新版本时,不中止开发进程
  - 有特殊版本时, 可以创建分支
- 不同版本具有相同的bug 时， 进行修复
- 追踪变化(trac)
- 自动构建(cruisecontrol)

什么是svn 和 TortoiseSVN?

- svn(Subversion)是个自由、开源的版本控制系统
- svn的版本库可以通过网络访问
- TortoiseSVN是Subversion在Windows操作系统上的图形客户端程序，开源、免费

软件安装

- 需要有管理员权限
- 安装完毕后，在“资源管理器”里点击右键，会有如下菜单出现。



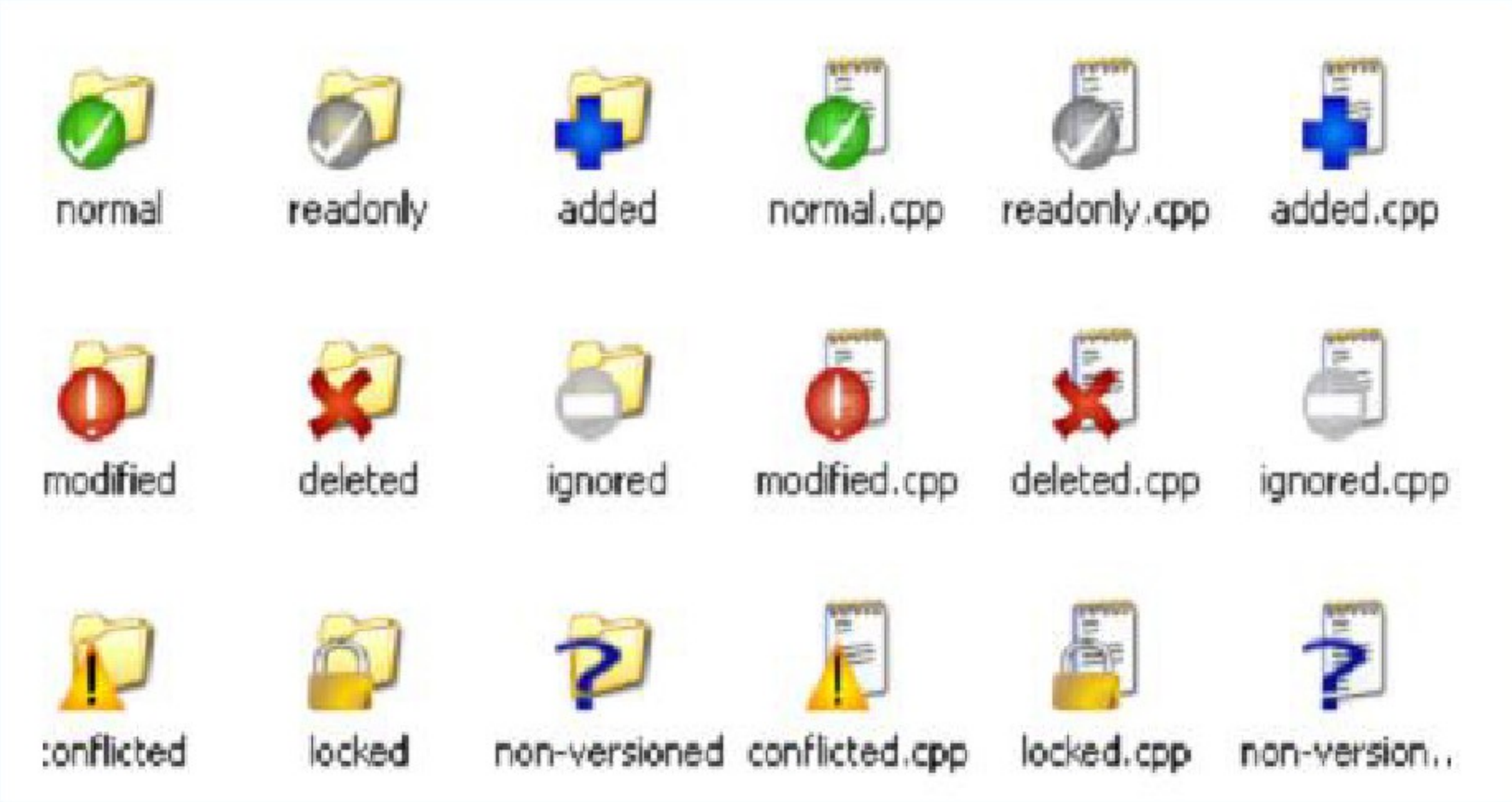
名词解释

- 版本库(Repository)的概念
  - 版本库是svn文件共享的核心，它储存所有的数据。按照文件树形式存储数据——包括文件和目录。
- 工作副本（Working copies）的概念
- 检出工作副本的概念

  你的版本库就像是一台时间机器，它记录了所有提交的修改，允许你检查文件或目录以及相 关元数据的历史。
- 提交修改的概念

  可以理解为上传修改文件的记录，就是把工作副本里面修改过的文件的记录上传到版本库里面。

图标重载

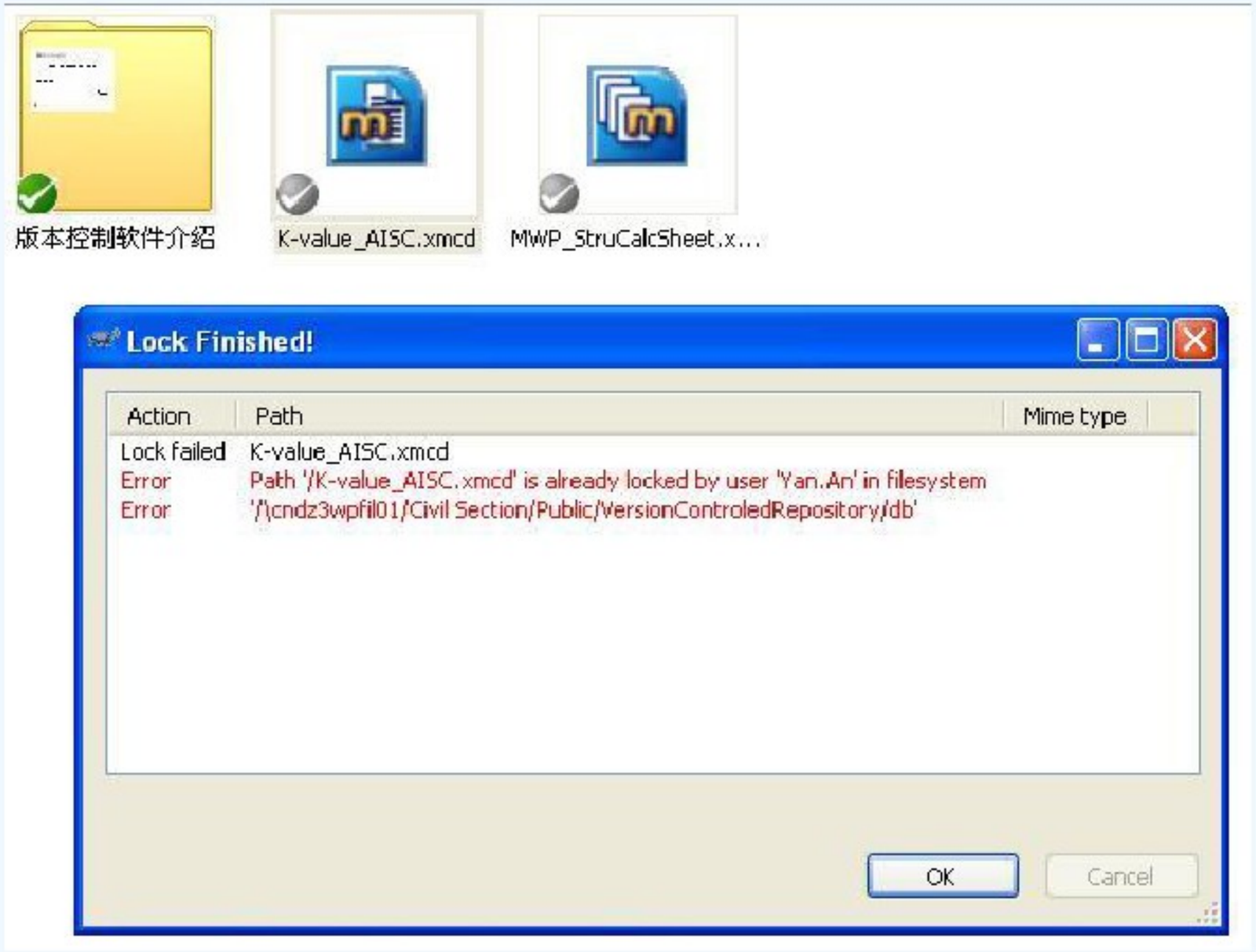

normal
readonly
added
normal.cpp
readonly.cpp
added.cpp
modified
deleted
ignored
modified.cpp
deleted.cpp
ignored.cpp
:onflicted
locked
non-versioned
conflicted.cpp
locked.cpp
non-version..

SVN的工作模式

- 协同工作的两种模式
  - “copy-edit-merge” (Subversion默认的模式)
    - 适用于纯文本文件的协同工作，TortoiseSVN自动合并不同成员对同一文件的修改，如果修改有冲突，需要人工取舍。
    - 此模式适合软件开发这种工作。
  - “Lock-Modify-Unlock” （锁模式）我们采用这种模式
    - 适用于纯文本文件及二进制文件的协同工作。比如dwg,mcd,doc,ppt,xls等文件格式。
    - 这种模式适合工程设计人员的合作。

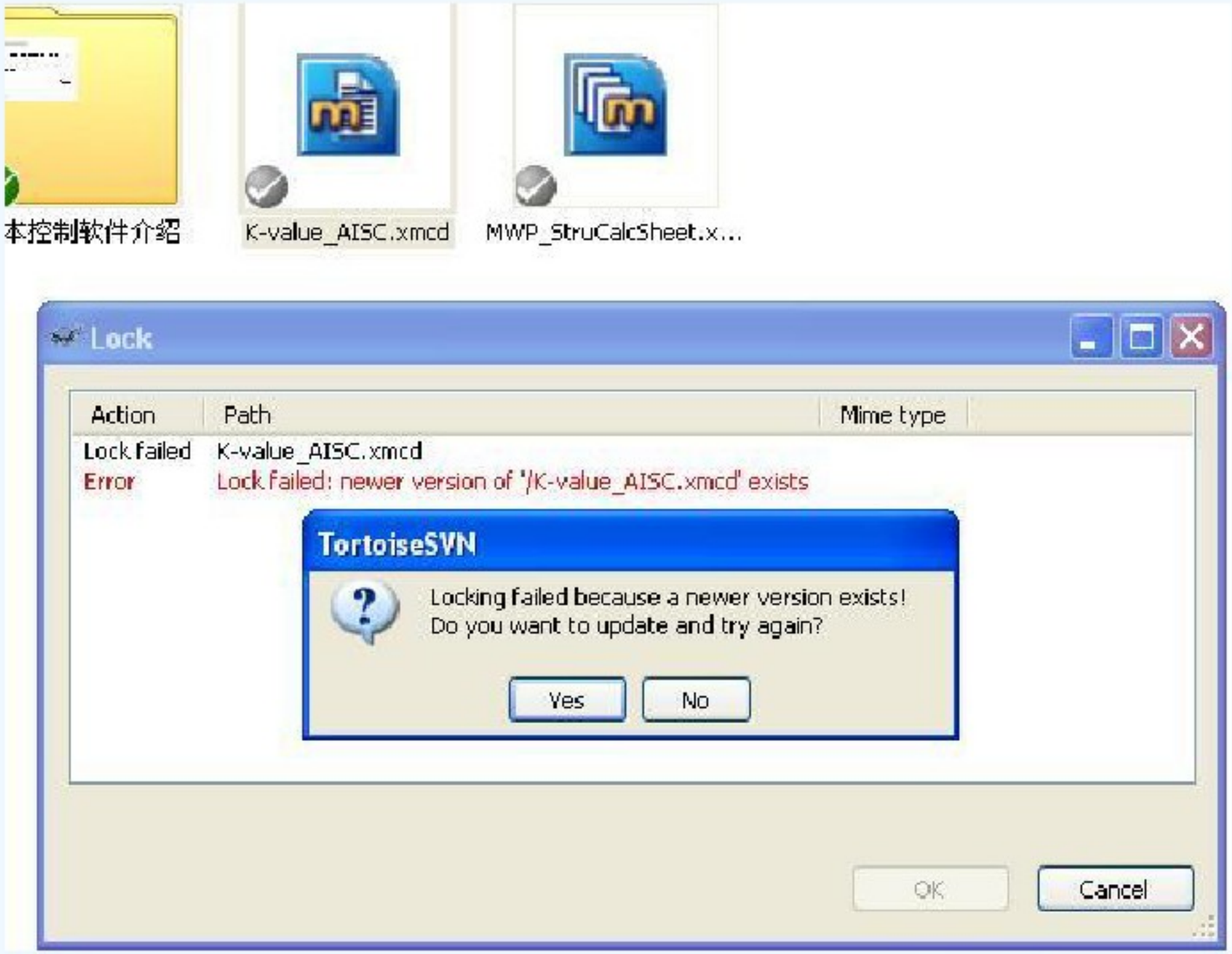
锁模式

- 某文件已经被同学甲锁定，此时同学乙也在自己的计算机上试图锁定该文件，会发生什么情况？（会有提示：该文件已被同学甲锁定）
- 某试衣间已经被同学甲锁定，此时同学乙也试图进入此试衣间，会发生什么情况？（试衣间内有人，请稍候）



当某成员试图锁定某文件时

TortoiseSVN会检测版本库中的该文件的版本是不是比Working Copy的新，如果比Working copy新，那么只有用户更新了Working copy才能对其成功实施锁定。这就保证了任何一个成员对文件修改时，总是在最新版本基础上进行修改。



提交的艺术

- 采用“Lock-Modify-Unlock”这种模式进行协同工作，需要多进行的两个步骤：
  - Get Lock（锁定文件）
  - Commit（提交,提交成功后程序自动把锁释放掉）
- 合理选择提交频度和时机
  - 两次提交之间尽量进行同一类操作，比如，统改图层后提交一次，统改标注线型后再提交一次。目的：一旦业主要求有反复，我们也很容易把以前工作成果从库中提出来，减少返工。
  - 建议一小时左右提交一次。
  - 养成按时提交的习惯，试衣间是公共财产，文件也是。

有人总在试衣间咋办

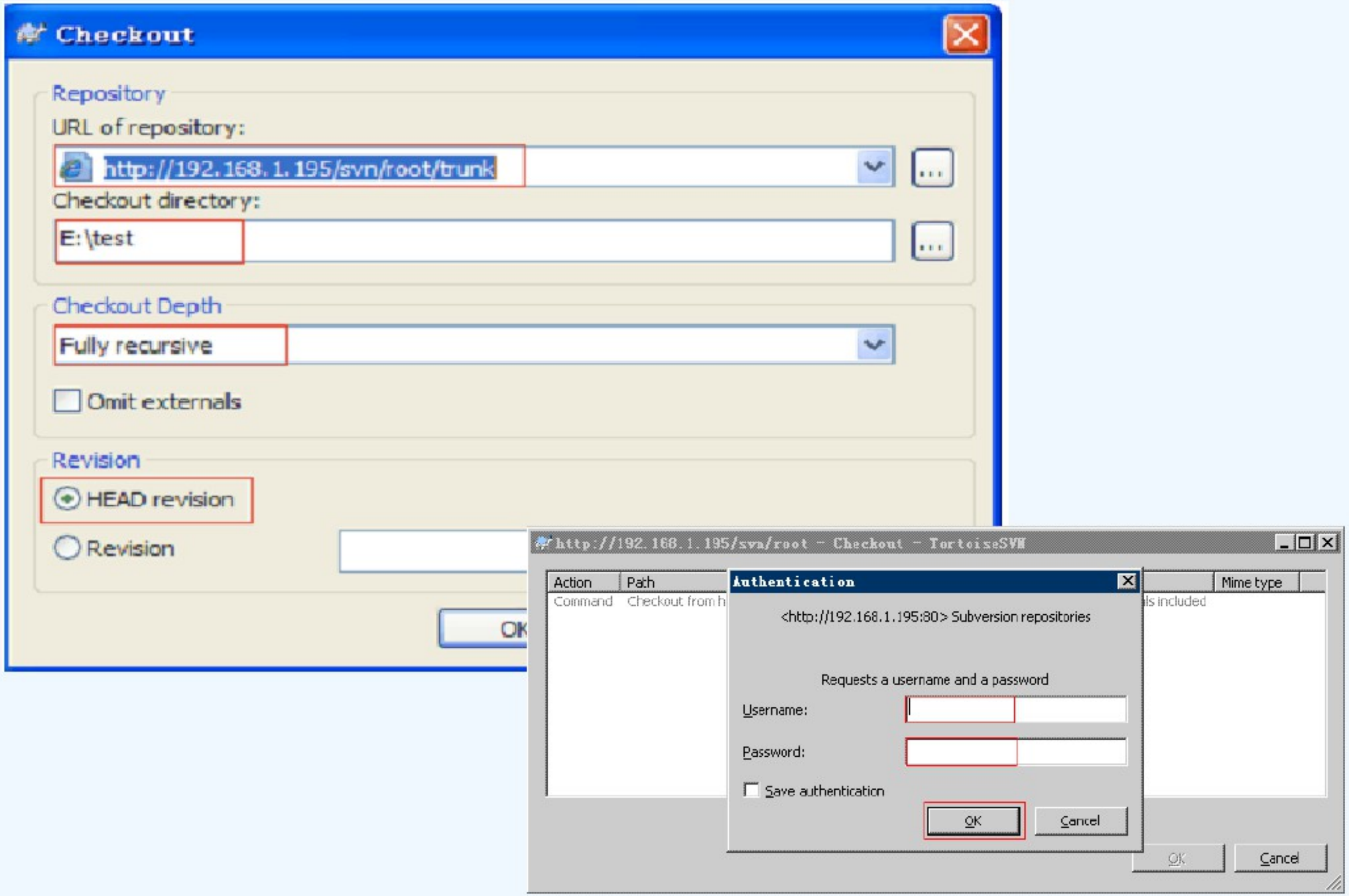
- 如果有人在试衣间内土遁了或者晕倒了，我们该怎么办？当然是救人要紧。
  - 砸锁（Steal the locks）

TortoiseSVN的功能演示

1. Checkout 检出工 作 副 本
2. Commit 修改文件后提交
3. Update 从服务器端更新本地文件的版本
4. Get Lock/Release Lock 对文件的加锁解锁操作
5. Add/Delete/Rename 增加/删除/重名文件或文件夹
6. Update to revision 更新至版本
7. Show log 显示日志
8. Revision Graph 版本示意图
9. Revert 取消上一次的操作（只针对客户端，服务端不做改动）
10. Branch/Tag 分支和标记
11. 其它功能

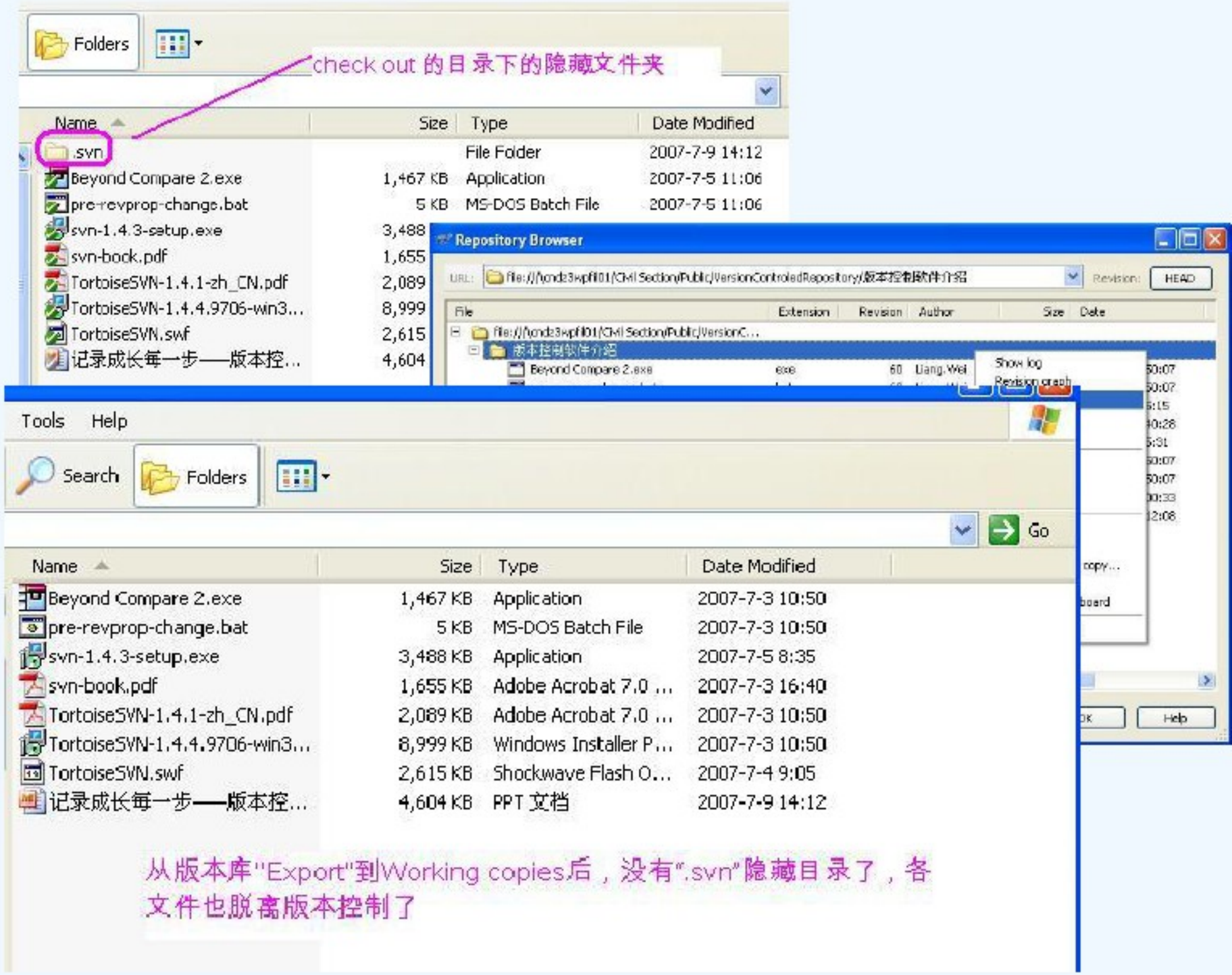
1. Checkout 检出工作副本

- 初次检出到本地文件夹时，在本地新建一个空文件夹，作为checkout项目的目录,这相当于拷贝，此文件夹目录相当于你本地机器一个普通的目录，为了得到工作拷贝，
- 具体操作如下图：


Checkout
Repository
URL of repository:
http://192.168.1.195/svn/root/trunk
Checkout directory:
E:\test
Checkout Depth
Fully recursive
Omit externals
Revision
HEAD revision
Revision
OK
http://192.168.1.195/svn/root - Checkout - TortoiseSVN
Action
Path
Command
Checkout from h
Mime type
Authentication
<http://192.168.1.195:80> Subversion repositories
Requests a username and a password
Username:
Password:
Save authentication
OK
Cancel
OK
Cancel

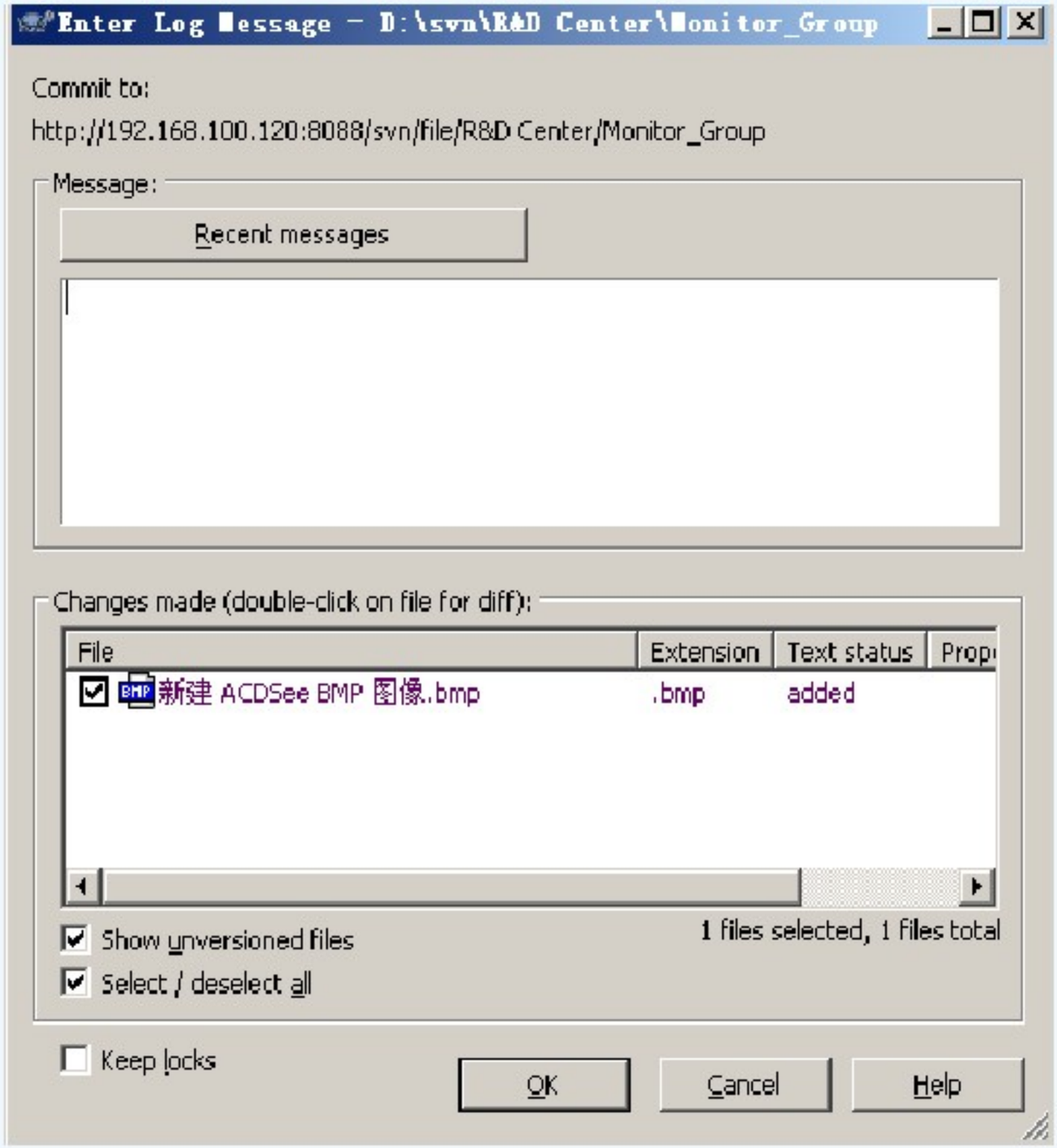
check out 和export的区别

- check out 方式获得文件后，文件仍处于SVN版本控制中
- export 导出当前版本的数据，文件脱离SVN版本控制



2. Commit 修改文件后提交

- (修改文件夹中的内容后文件夹会自动变成带有红色标记)
- 大家可以发现,文件/文件夹已经发生了变化，由之前绿色的勾变成了红色的感叹号，这表明该文件已经发生了更改，变的与服务器的文件不一样了


Enter Log Message - D:\svn\R&D Center\Monitor_Group
Commit to:
http://192.168.100.120:8088/svn/file/R&D Center/Monitor_Group
Message:
Recent messages
Changes made (double-click on file for diff):
File
Extension
Text status
Prop
新建 ACDSee BMP 图像.bmp
.bmp
added
Show unversioned files
1 files selected, 1 files total
Select / deselect all
Keep locks
OK
Cancel
Help

log message 信息填写规则

- 好的log message和糟糕的log message
  - log message主要记录的是每次的修改内容。建议把一些重要数据、关键操作写到log message中，比如“根据管道工程师山姆同志在2007年7月4日的邮件内容修改结构布置”，可以顺便把山姆同志的邮件内容拷贝到log message中。
  - 注：修改人和提交时间由软件自动记录，无需人工写入log message

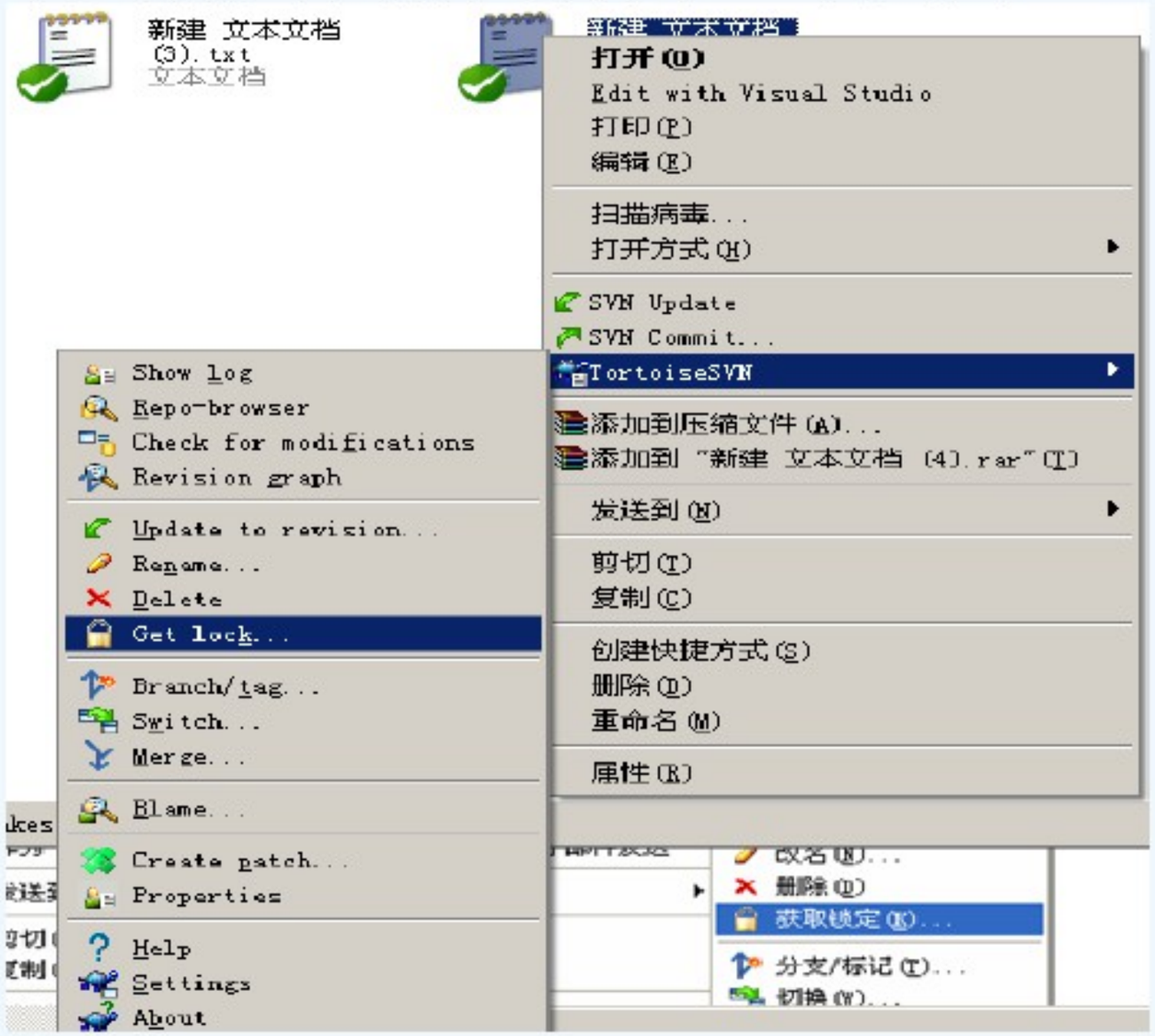
good log message	not good
移动右上角架子标注由原来的1'6"改为1'	改标注
增加注释，说明设备V3000的参数出处	增加 说明
原BOS EL 69'-11 1/2"改为72'-1 1/2"	改标高
修改最南侧和最北侧的W12的梁与TOWER中心线的距离为1'1" 修改5'挑台的挑梁为W16x57	改距离

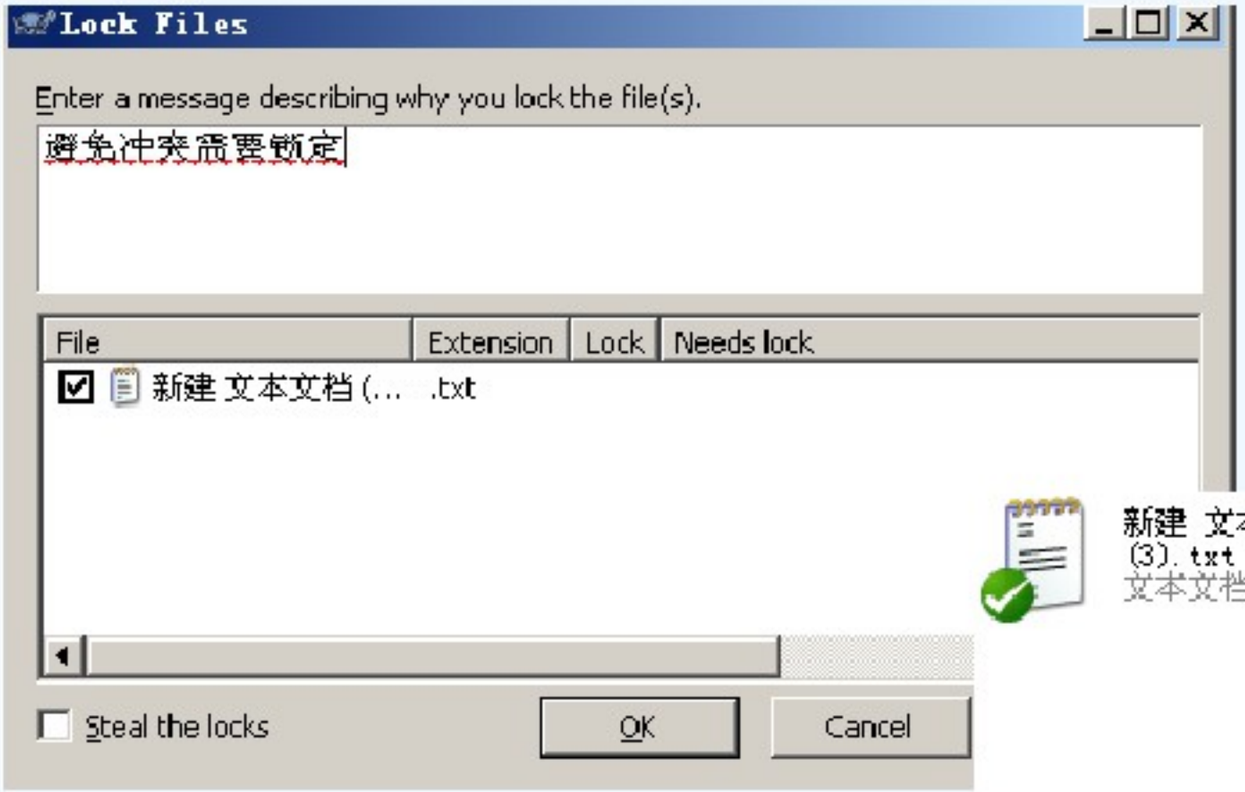
3. Update 从服务器端更新本地文件的版本

- 如果服务器上的版本库已经是第7版本,自己本机的版本还是第1版本。
- 这时你可以更新自己本地的版本为第7版本。

4. Get Lock/Release Lock 对文件的加锁解锁操作

- 当我们同时修改一个项目下的文件时，需要锁定文件（可以是一个文件夹或者具体的一个页面）这样可以避免冲突。
- **加锁：**

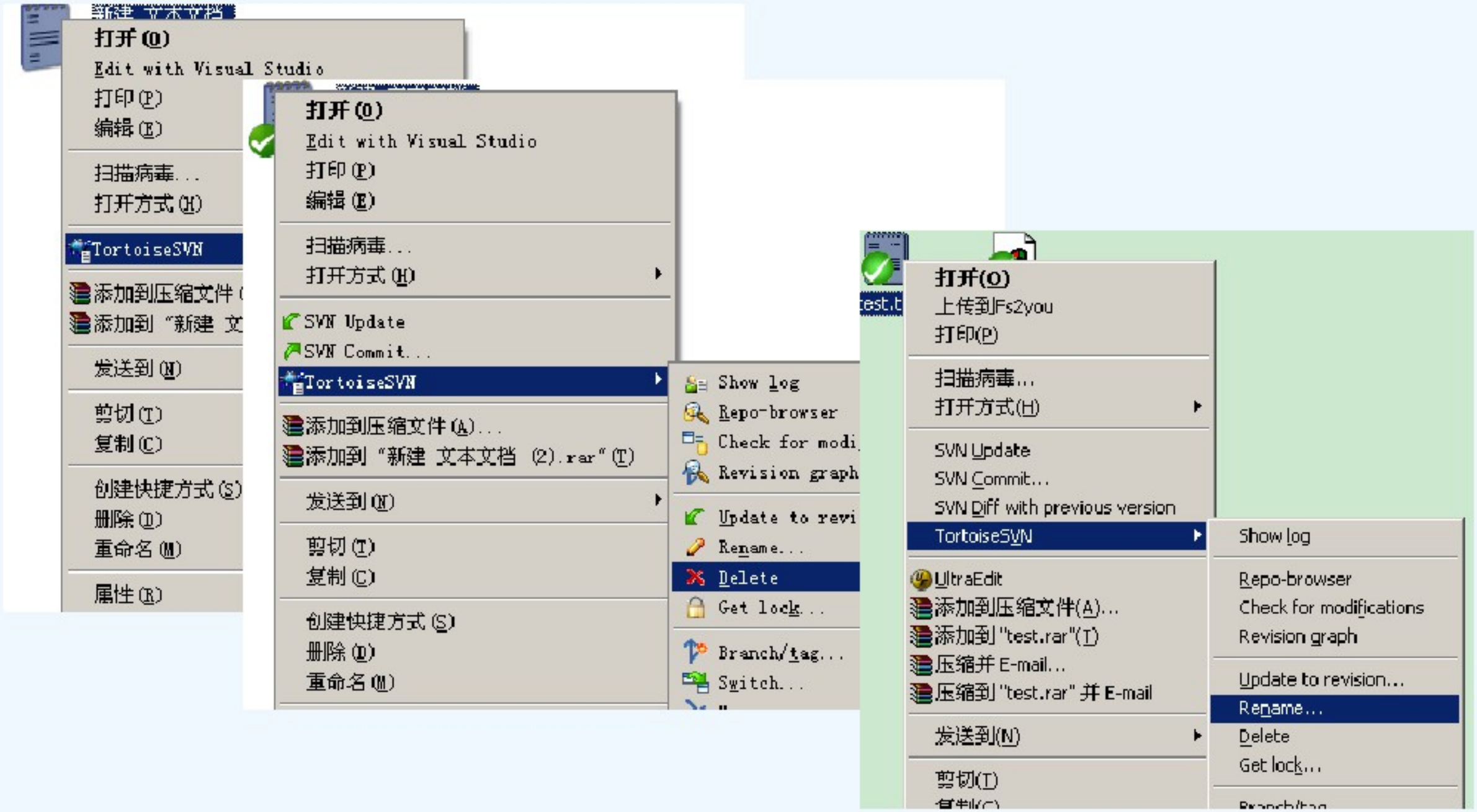



Lock Files
Enter a message describing why you lock the file(s).
避免冲突需要锁定
File
Extension
Lock
Needs lock
新建 文本文档 (...
.txt
Steal the locks
OK
Cancel

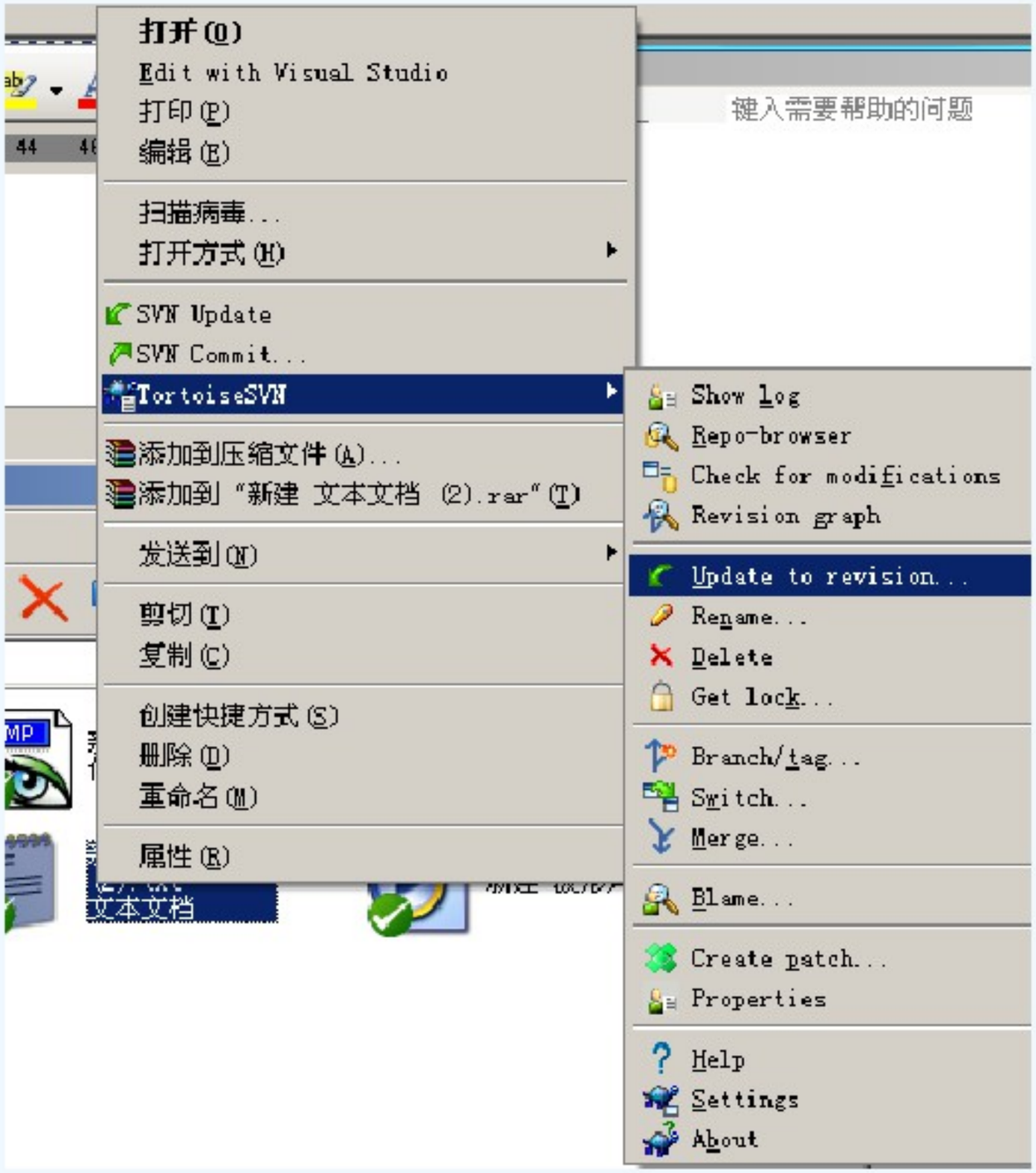
•解锁：

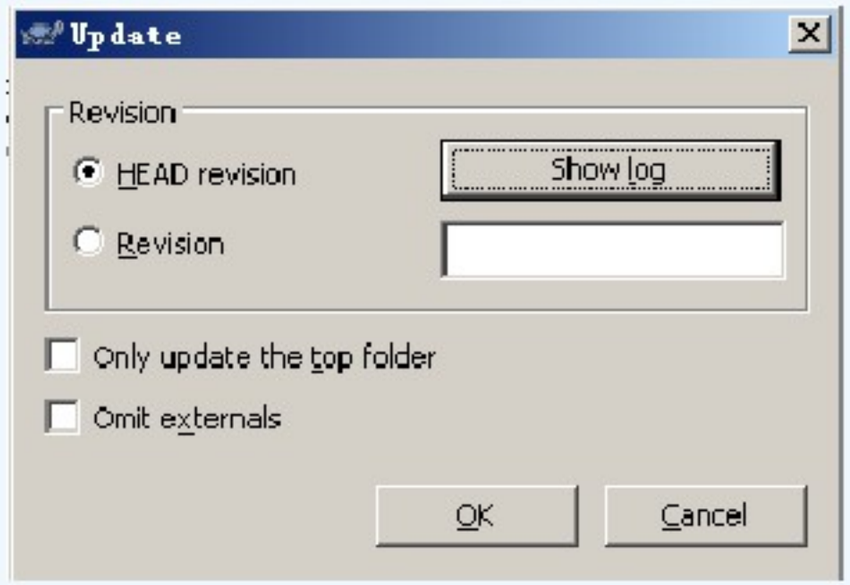
新建 文本文档 (3).txt
文本文档
新建 文本文档
打开(O)
Edit with Visual Studio
打印(P)
编辑(E)
扫描病毒...
打开方式(H)
SVN Update
SVN Commit...
TortoiseSVN
添加到压缩文件(A)...
添加到"新建 文本文档 (4).rar"(T)
发送到(N)
剪切(T)
复制(C)
创建快捷方式(S)
删除(D)
重命名(M)
属性(R)
Show log
Repo-browser
Check for modifications
Revision graph
Update to revision...
Rename...
Delete
Release lock
Branch/tag...
Switch...
Merge...

5. Add/Delete/Rename 增加/删除/重名文件或文件夹

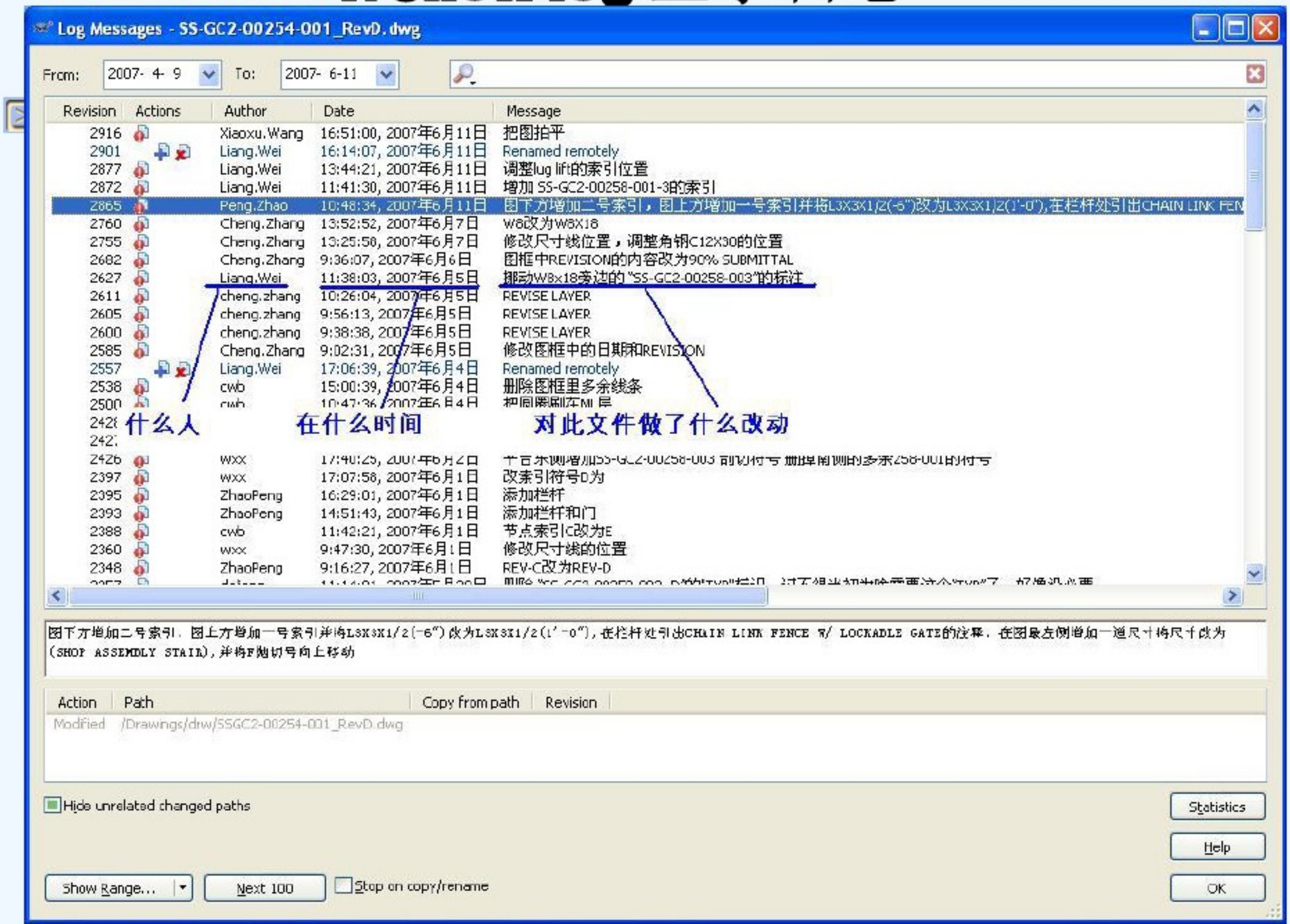


6. Update to revision 更新至版本

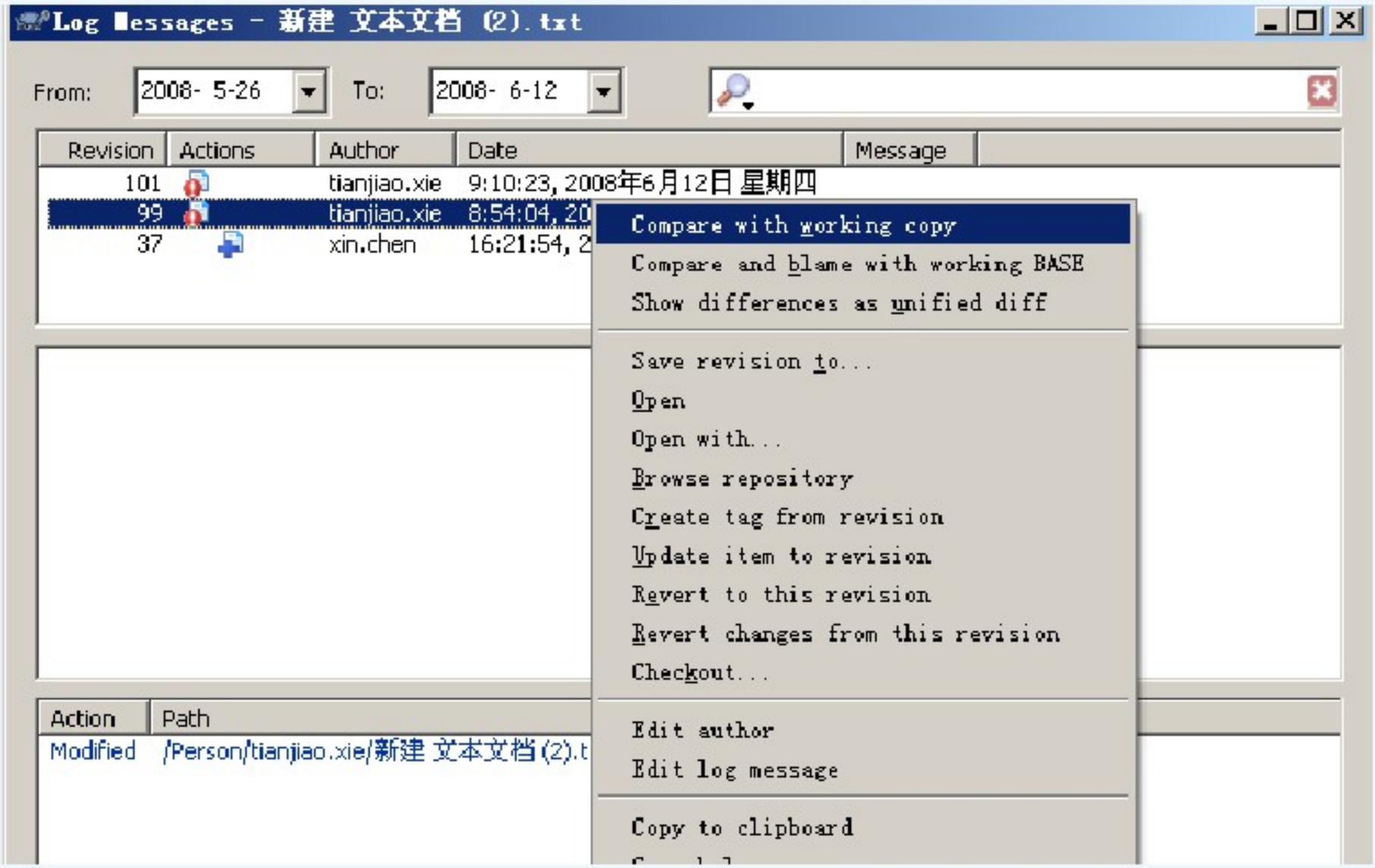




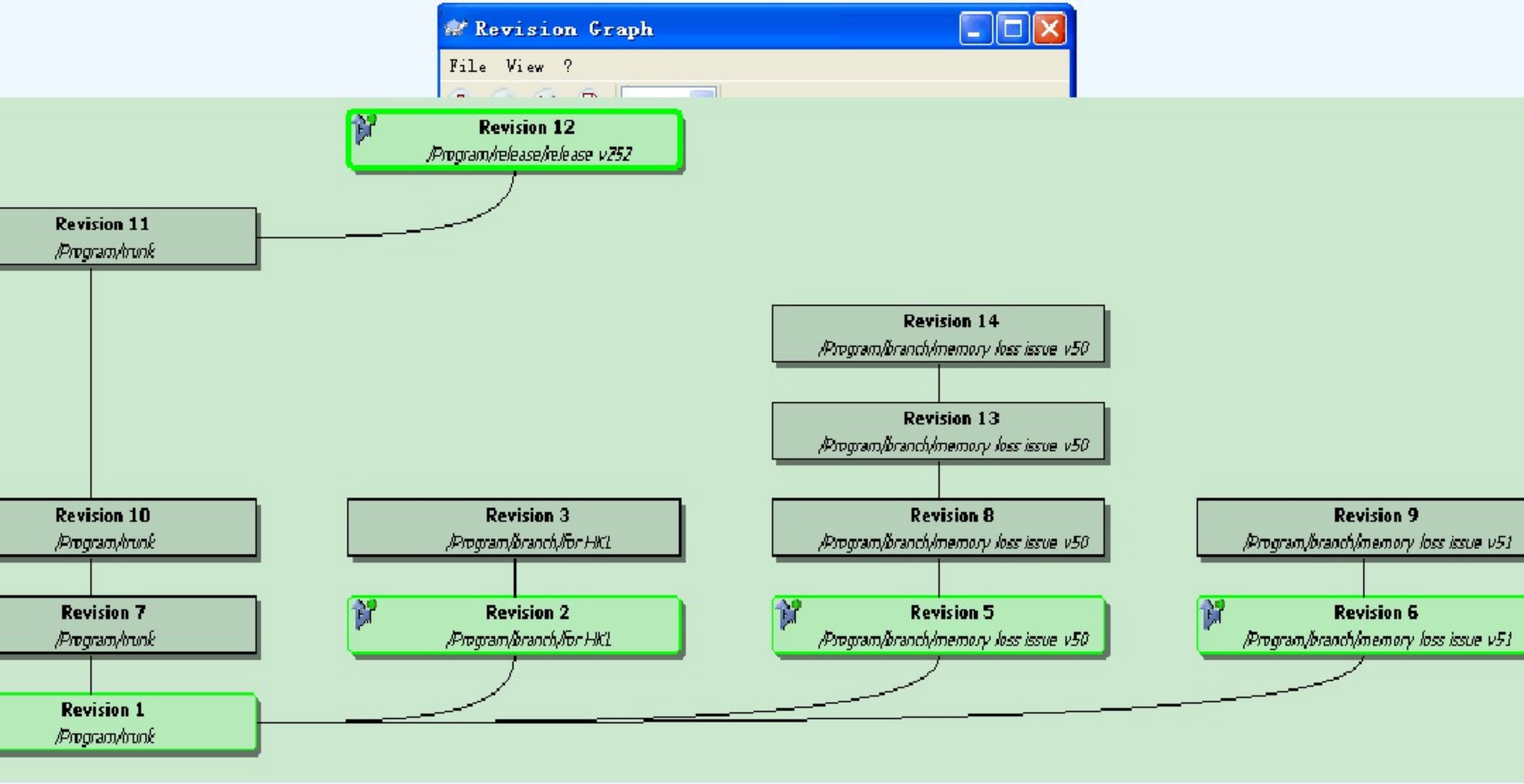
7. Show log 显示日志



- 我们可以看到文件版本的一些信息，还有文件做了什么操作。
- 若想知道前后2个版本有什么差别，更改了哪些东西，选中文件，如上图。
- 右键，选择Compare with working copy

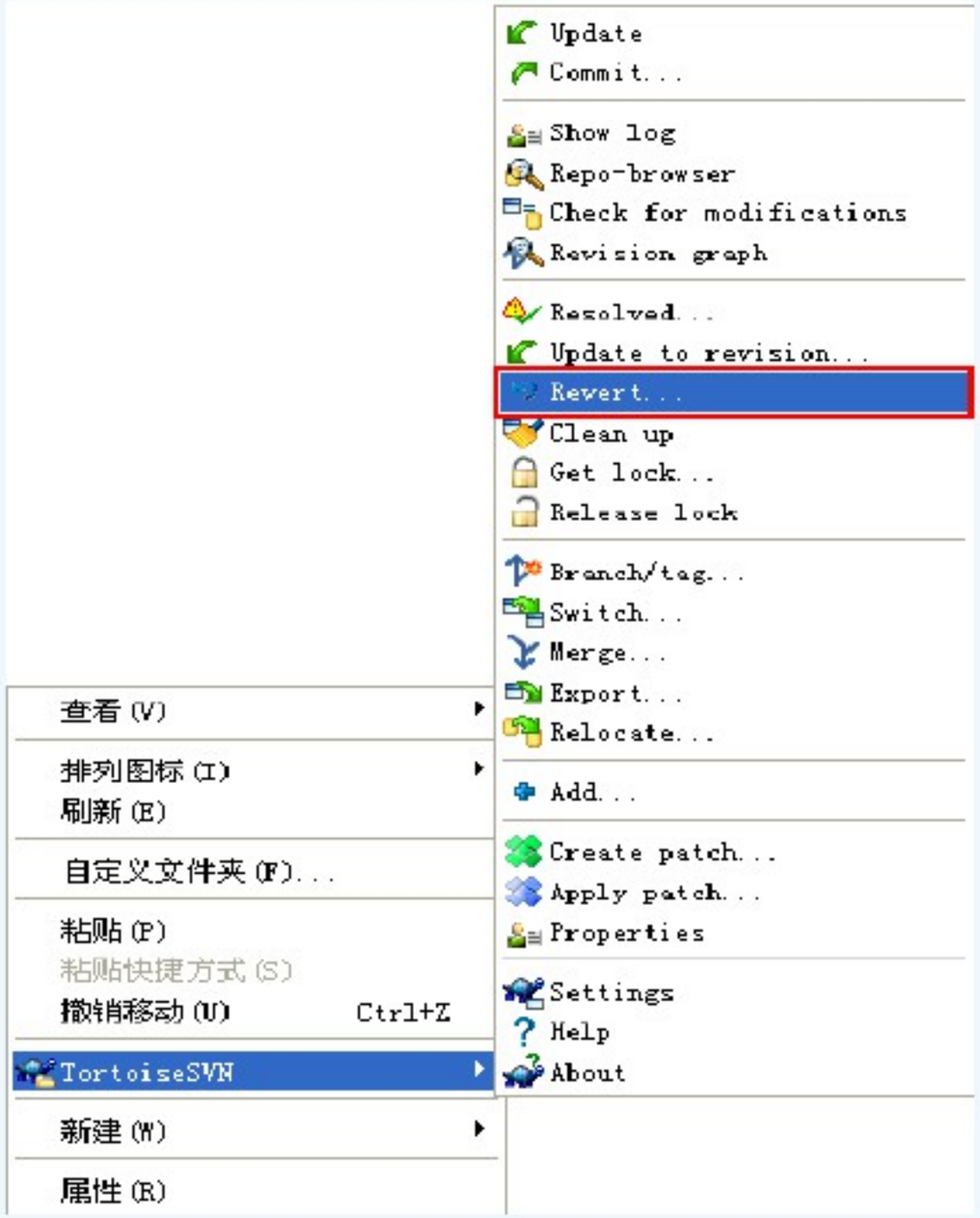


8. Revision Graph 版本示意图



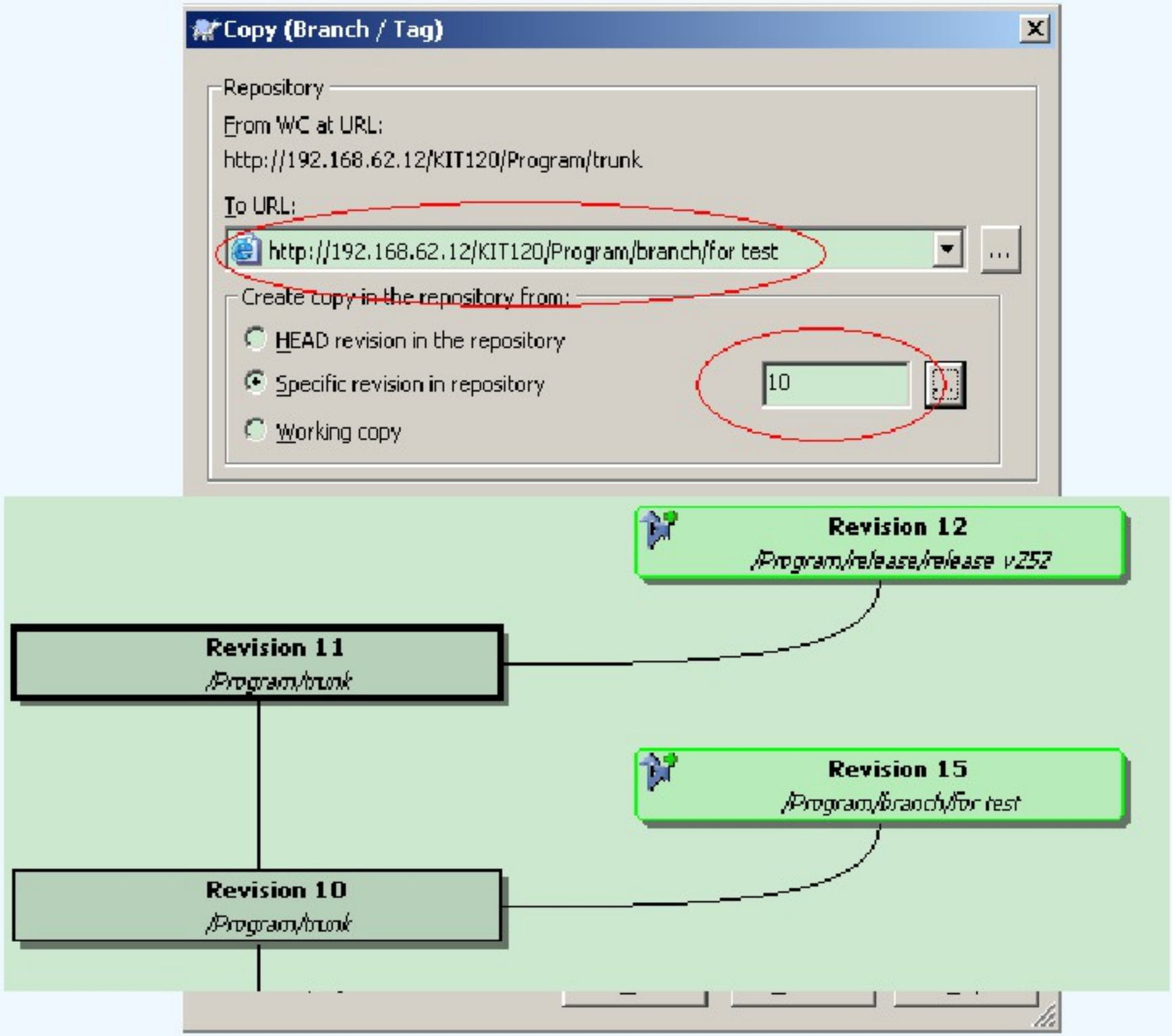
9. Revert 取消上一次的操作

- 注意：只针对客户端，服务端不做改动



10. Branch/Tag 分支和标记

- 注：在想要分支的目录下或对单个文件做如下操作：

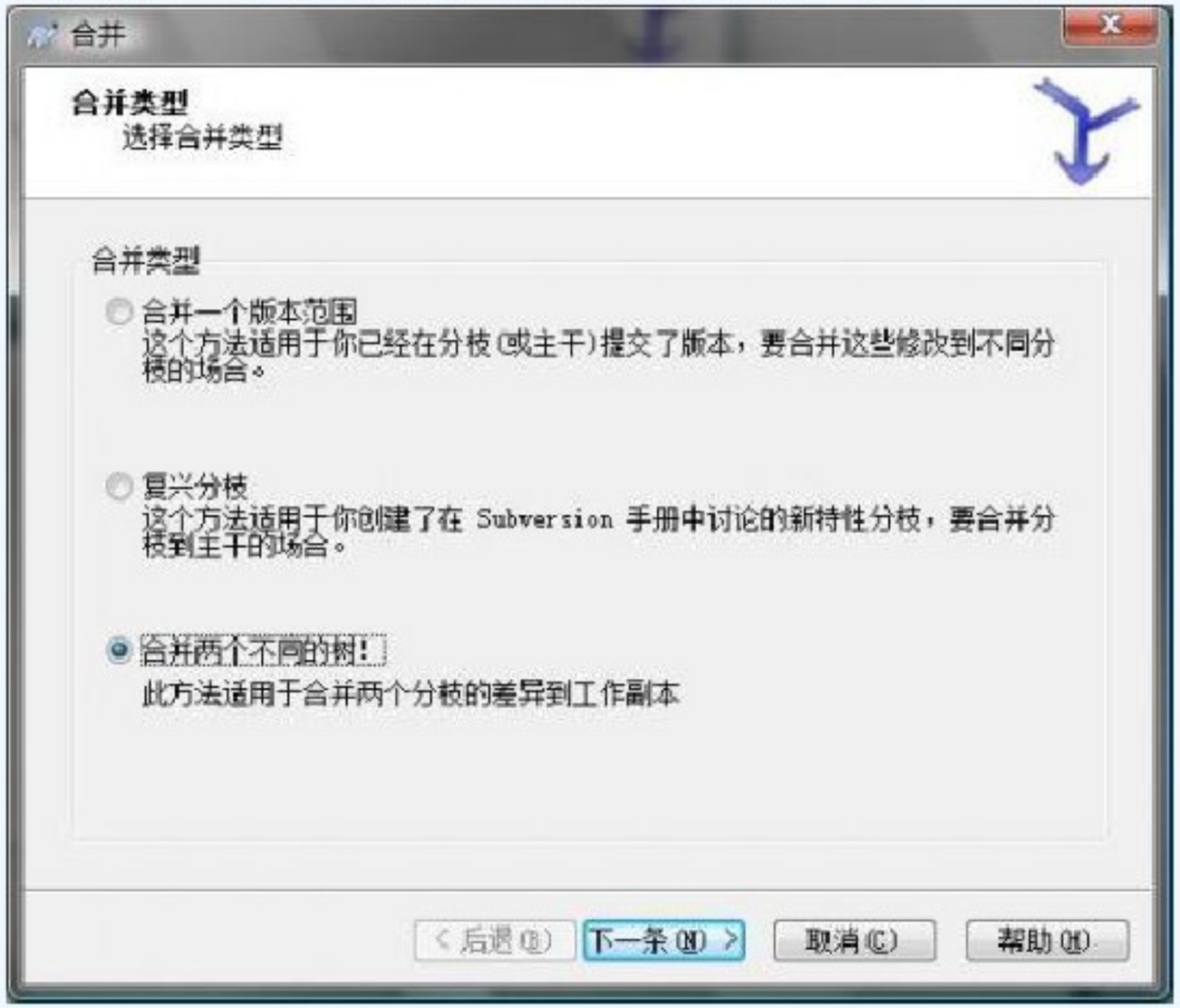

Copy (Branch / Tag)
Repository
From WC at URL:
http://192.168.62.12/KIT120/Program/trunk
To URL:
http://192.168.62.12/KIT120/Program/branch/for test
Create copy in the repository from:
HEAD revision in the repository
Specific revision in repository
10
Working copy
Revision 12
/Program/release/release v252
Revision 11
/Program/trunk
Revision 15
/Program/branch/for test
Revision 10
/Program/trunk

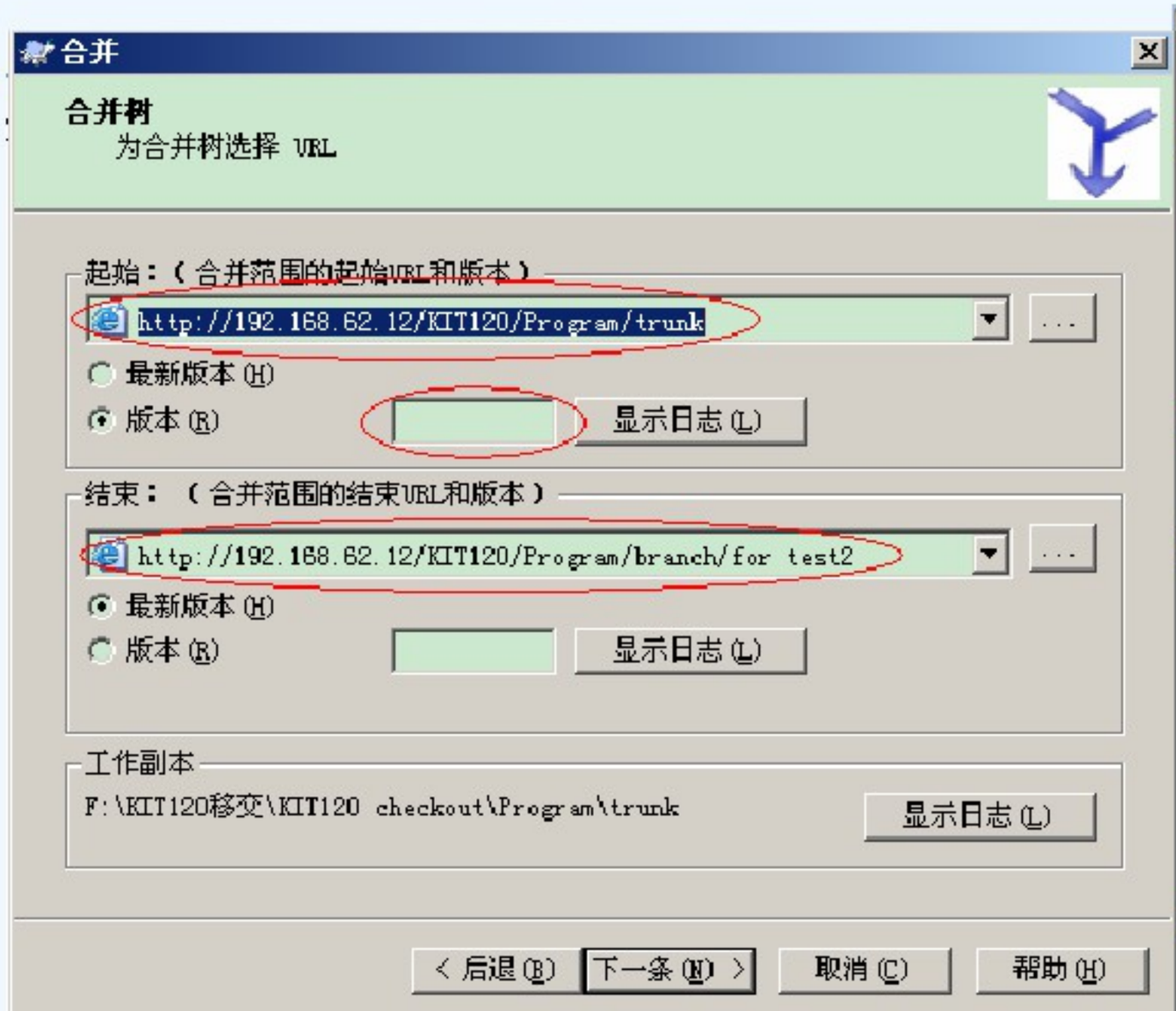
11.其它功能

- Import 导入
- Repo-browser 版本库浏览器
- Relocate 重定位
- Merge 合并---解决冲突
- Blame 追溯
- Diff 差异

Merge 合并

- 第一步：将工作副本切换到主干上：这一步很重要，当前的工作副本的是要合并的目的地
- 第二步：在主干根目录文件夹上右键，如下图，选择最后一项





这是最重要的一步，需要注意的地方有：

1.起始URL：选择主干目录的URL（应当和当前工作副本的URL一致，看起来不合理，其实细想却是合理的，这个是所谓的合并点，也就是分歧开始的地方）

2.结束URL：选择要合并的分支的URL，也就是所有的分歧所在。

3.起始和结束的版本：起始版本应当找到最后一次同步时的版本，如果从没有同步过（第一次合并），则选择创建分支时的版本，结束版本一般是最新版本，如果你不想将某些内容合并进主干的话，也可以选择一个合并点。

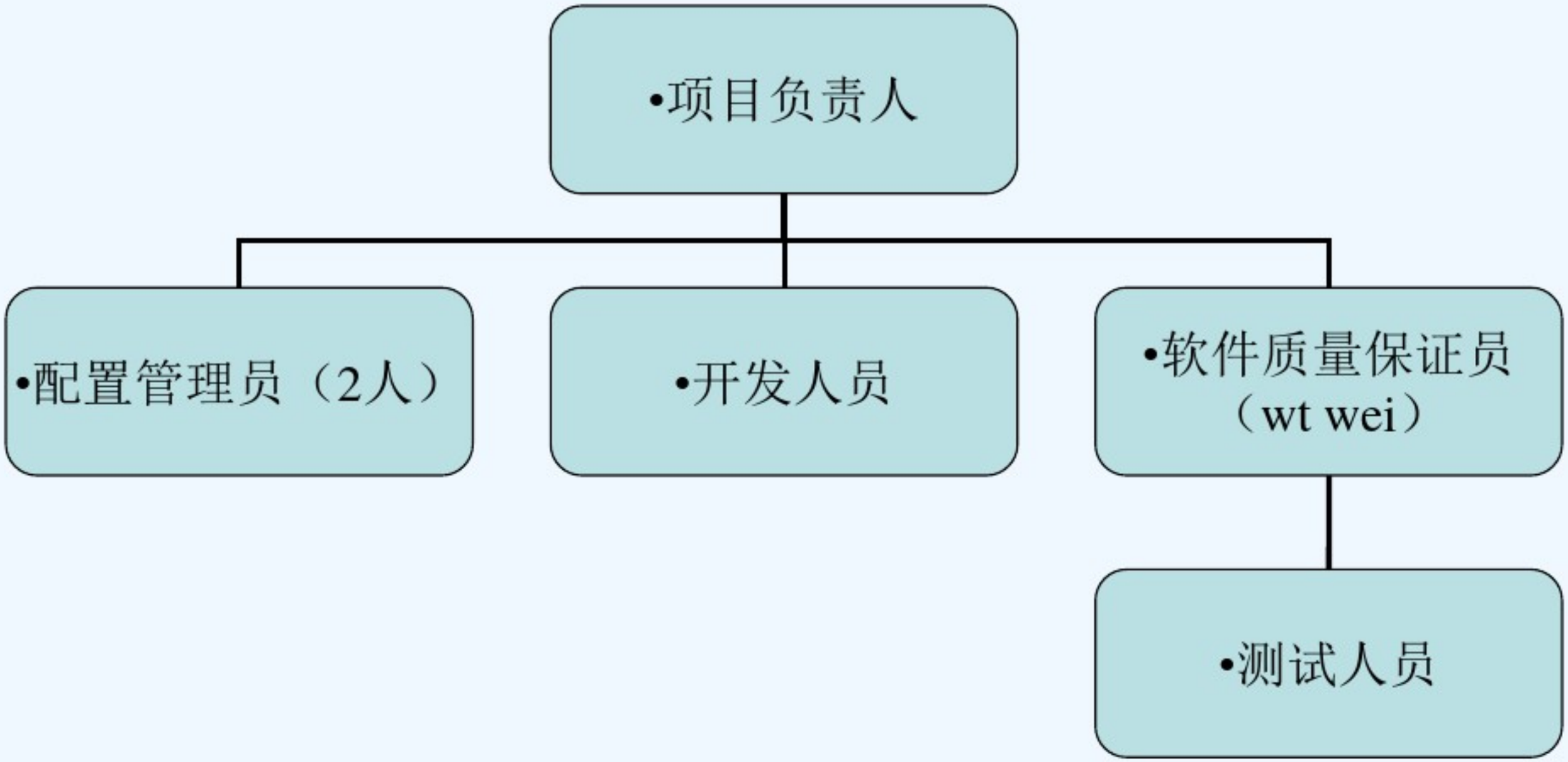
到这里，下面就顺理成章了，如果有冲突的话，手工解决冲突，提交就可以了。

- 我的理解：将分支合并到主干上，首先需要在主干的工作副本下进行，合并的范围是从主干的上次合并的版本开始（FROM）到分支上最新的版本结束（TO），如果是第一次合并，则从主干创建分支的版本开始，所以每次合并好做好说明，否则忘记了，下次再合并就有点麻烦，其实，应当尽量避免一个分支合并多次，分支的作用一般为了解决bug等，一旦bug对应结束了，分支的使命就结束了，以后再出现其他的问题，应当重新建立分支，这样就不会出现多次合并的问题了。
- 合并的工作是把主干或者分支上合并范围内的所有改动列出，并对比当前工作副本的内容，由合并者手工修改冲突，然后提交到服务器的相应目录里。如果当前工作副本是主干，则合并的范围是分支上的改动，如果工作副本是分支的，则合并范围是主干上的改动，并且一定要注意，合并的起始位置URL一定和当前的工作副本的URL是相同的。

项目目录设置

- Documents
- Program
  - Trunk
  - Branch
  - Release
- Tools

人员分配与组织结构



The end! Thank you!