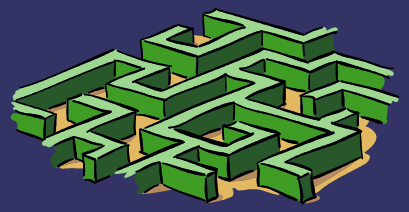
Two Improvements to Subversion

“True” Renames

and

Merging Identical Changes

by
Julian Foad



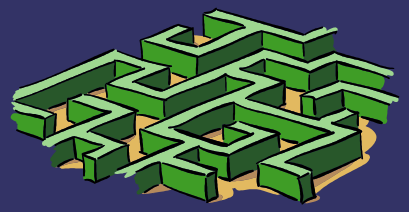
Who Am I?

Worked for CollabNet

on Subversion's tree conflict handling

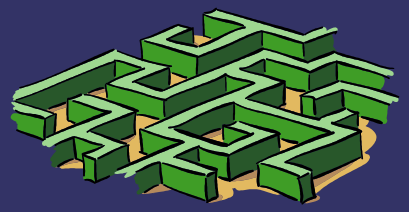
Volunteer contributor to Subversion

This is my opinion, not necessarily the opinion of the Subversion community.



1. “True” Renames

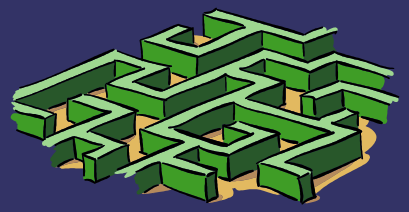
What is the problem?



Contents

Renaming in Subversion now

What is a “true” rename?



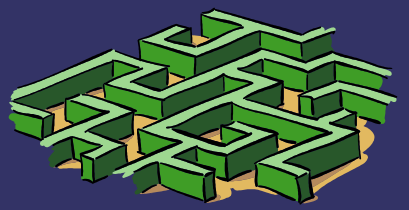
Rename == copy + delete

```
svn rename OLD NEW
  A  NEW
  D  OLD

svn log -v
  D  OLD
  A  /NEW (from /OLD:1)
```

This works OK...

```
svn update
```

find the old name



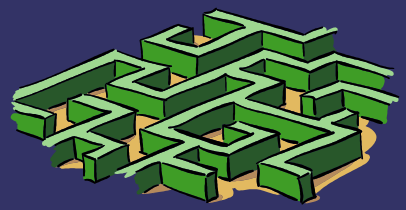
What Works

Propagate to other users

Follow history backwards

```
svn update
  D  OLD
  A  NEW
```

```
svn log -v -r1 NEW
  [...]
  A /OLD
```

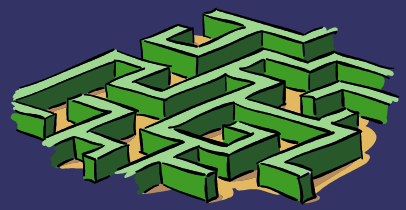


What Doesn't Work

Merging

In branch B

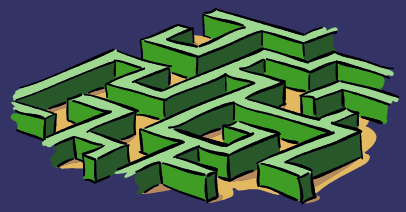
```
svn merge ^/A
  A    NEW
  D    OLD

svn log -vq -rHEAD
  r6 | julianfoad | ...
  Changed paths:
     M /B
     A /B/NEW (from /A/NEW:3)
     D /B/OLD
```



What Doesn't Work

Follow history forwards

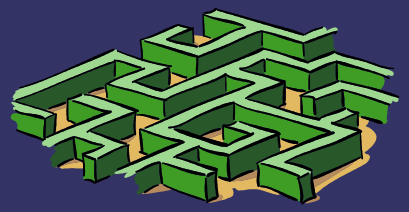
```
svn diff -r1:2 ^/OLD@1
  not found: revision 2,
     path '/OLD'
```



Contents

Renaming in Subversion now

What is a “true” rename?

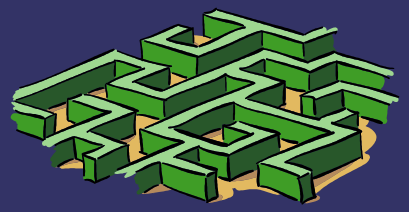


What is a “True” Rename?

copy and delete the same item

- copy from a **relative** path (./OLD)
  - at **this** revision
  - to the new path (./NEW)
- delete the same item (./OLD)

the two halves are **indivisible**

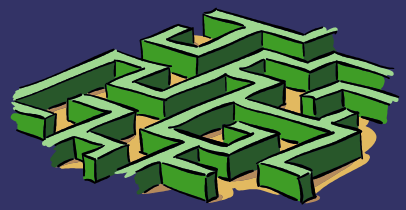


How it Should Work

Merging

In branch B

```
svn merge ^/A
  A    NEW
  D    OLD

svn log -vq -rHEAD
  r5 | julianfoad | ...
  Changed paths:
     M /B
     A /B/NEW (from /B/NEW@4)
     D /B/OLD
```



Development

info from the Working Copy

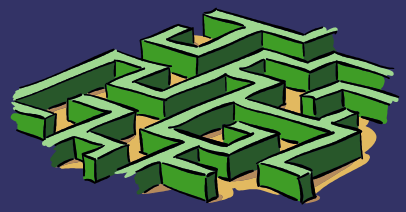
maybe in 1.7 ?

info in the Repository

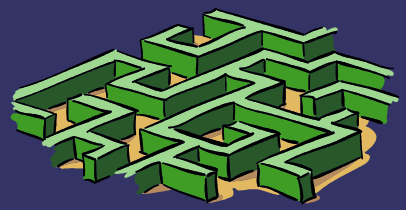
possible in 1.7/1.8 ?

info used in Merging

maybe in 1.9/2.0 ?



2. Two Identical Changes



Two Identical Changes

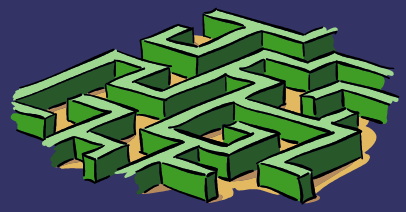
An improvement to tree conflict handling

When we merge two identical changes

- say, “delete file1” and “delete file1”

As Stephen/Neels said, 1.6 detects

- conflicts you need to know about
- some conflicts you wish it would resolve automatically



Example 1

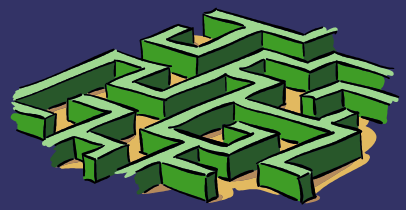
On a feature branch 'B':

commit ...
commit a small patch 'P1'
commit ...

```
Mod    alpha.c
Del    beta.c
```

Catch up from trunk:

```
svn merge ^/trunk
M      alpha.c
   C  beta.c
   ^--- delete/delete
```



Example 1

Why the conflict?

- Somebody already applied 'P1' on trunk
- Maybe this happens frequently
- Subversion combines two identical text changes, so why not this tree change?

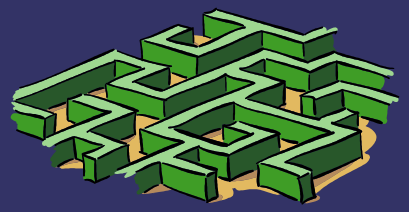
Can you resolve it?

```
svn resolve -accept=mine-full beta.c
```

Can we avoid this inconvenience?

```
svn merge -accept=mine=full
```

...?



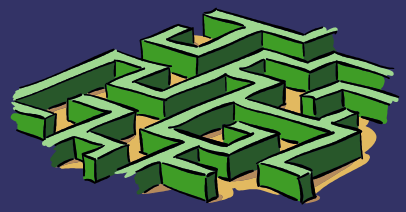
Example 2

On a feature branch 'B':

Modify 'gamma.c' to include all of 'beta.c'
Delete 'beta.c'

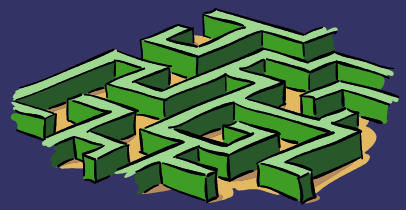
Catch up from trunk:

```
svn merge ^/trunk
M     alpha.c
   C  beta.c
   ^--- delete/delete
```



Example 2

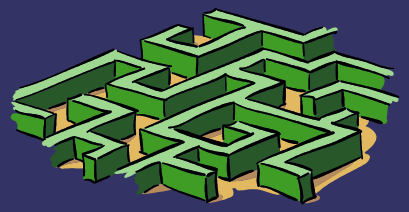
- What's the difference?
  - Two copies of the content from 'beta.c'
  - Subversion doesn't know that
- Why do we want a conflict? Let's say...
  - we rarely apply the same patch on two branches
  - we are versioning documents – no automatic checks
  - we are using an external tool to help resolve conflicts



I Want

I want to select between

- “Relaxed” mode:
  - The same change twice -> do it once
- “Strict” mode:
  - The same change twice -> raise a conflict

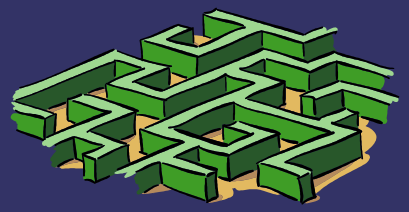


Observations

Important for big trees

Subversion is “relaxed” in text conflicts

The same principle applies to double adds



Status

In v1.6 we have “strict” detection:

- so we can be sure to find all tree conflicts
- so other tools can help
- because rename = copy + delete

To implement a choice:

- UI
- support in WC layer

Questions

Thank you for listening.

- Of course, it is possible to do better
- See tools such as Subclipse

Any questions?