Introducing Subversion 1.8

Hosted by Adrian Bridgwater

wandisco

Local Move Updating

Philip Martin

What is local move updating?

- Local moves are uncommitted moves
- Moved items can be out-of-date
- Move updating to resolve tree conflicts
- Client-side solution

A simple Java class

```
$ cat Foo.java
package com.wandisco.example;
public class Foo {
    private int count;
    public Foo(int initCount) {
        count = initCount;
    }
}
$
```

A local move

```
$ svn move Foo.java Bar.java
$ vi Bar.java
$ cat Bar.java
package com.wandisco.example;
public class Bar {
    private int count;
    public Bar(int initCount) {
        count = initCount;
    }
}
$
```

Out-of-date failed commit

```
$ svn commit
svn: E170004: File 'Foo.java' is out of date
$ svn cat Foo.java@HEAD
package com.wandisco.example;
public class Foo {
    private int count;
    public Foo(int initCount) {
        count = initCount;
    }
    public boolean newFeature() {
        return count > 0;
    }
}
```

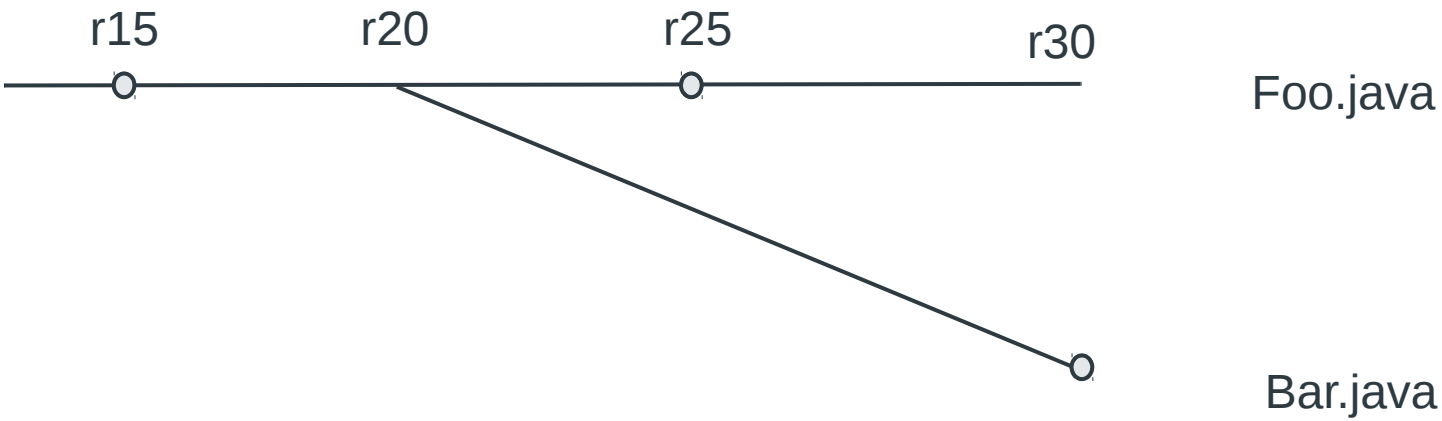
Update gives a conflict

```
$ svn update --accept postpone
   C Foo.java
At revision 28.
Summary of conflicts:
  Tree conflicts: 1
```

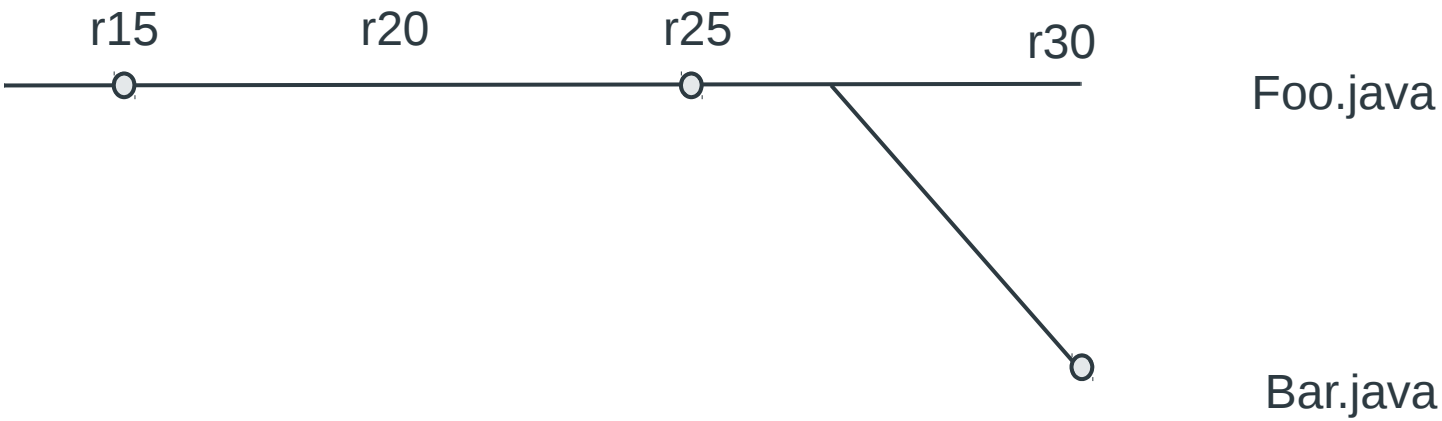
The wrong solution

```
$ svn merge ^/com/wandisco/example/Foo.java Bar.java
$ cat Bar.java
package com.wandisco.example;
public class Bar {
    private int count;
    public Bar(int initCount) {
        count = initCount;
    }
    public boolean newFeature() {
        return count > 0;
    }
}
```

The wrong history



The right history



The right move

```
$ cat Bar.java
package com.wandisco.example;
public class Bar {
    private int count;
    public Bar(int initCount) {
        count = initCount;
    }
    public boolean newFeature() {
        return count > 0;
    }
}
```

The correct solution

```
$ # 1.7 solution
$ svn diff Bar.java > my-changes.patch
$ svn revert Foo.java Bar.java
$ svn move Foo.java Bar.java
$ svn patch my-changes.patch Bar.java

$ # 1.8 solution
$ svn resolve --accept mine-conflict Foo.java
```

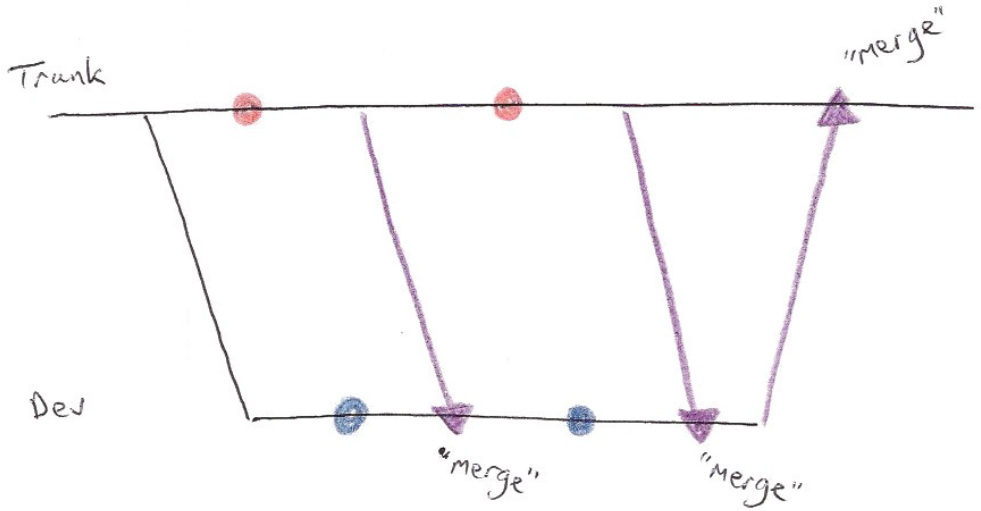
Supported features

- Moved directories and files
- Nested moves within moves
- Text/property changes, adds/deletes
- Prevents "half moves”

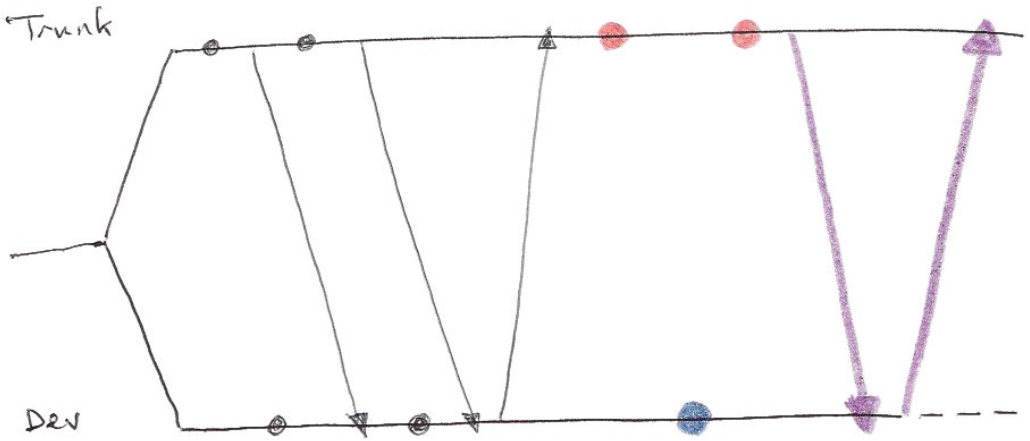
Automatic Merging in Subversion 1.8

Julian Foad

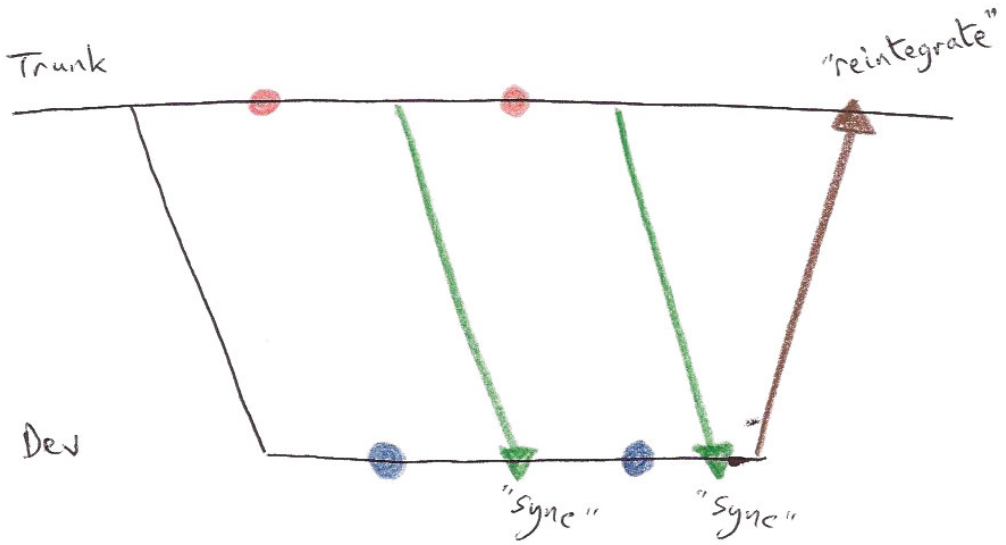
Automatic Merging – The Goal (1)



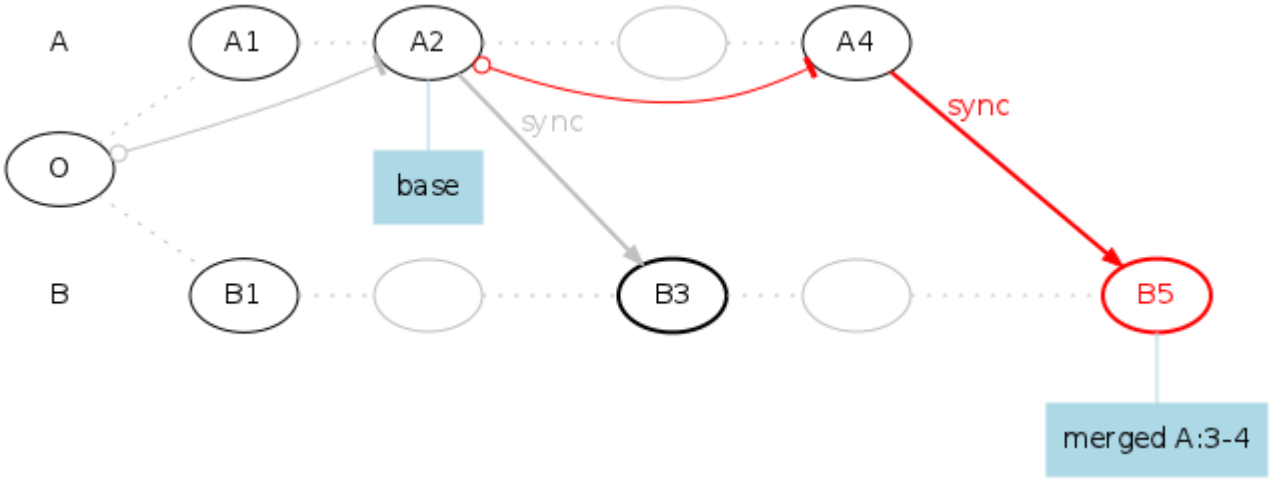
Automatic Merging – The Goal (2)



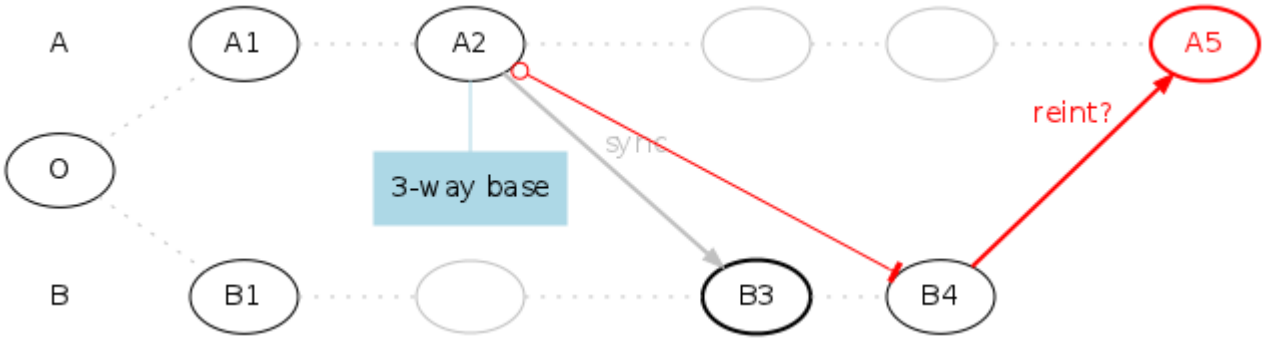
Merging in Svn 1.7



Sync vs. Reintegrate (1)



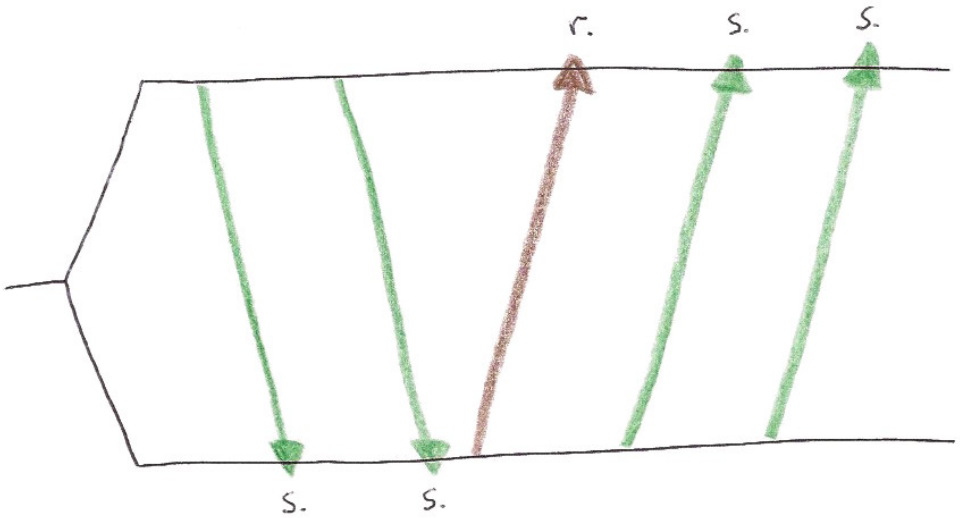
Sync vs. Reintegrate (2)



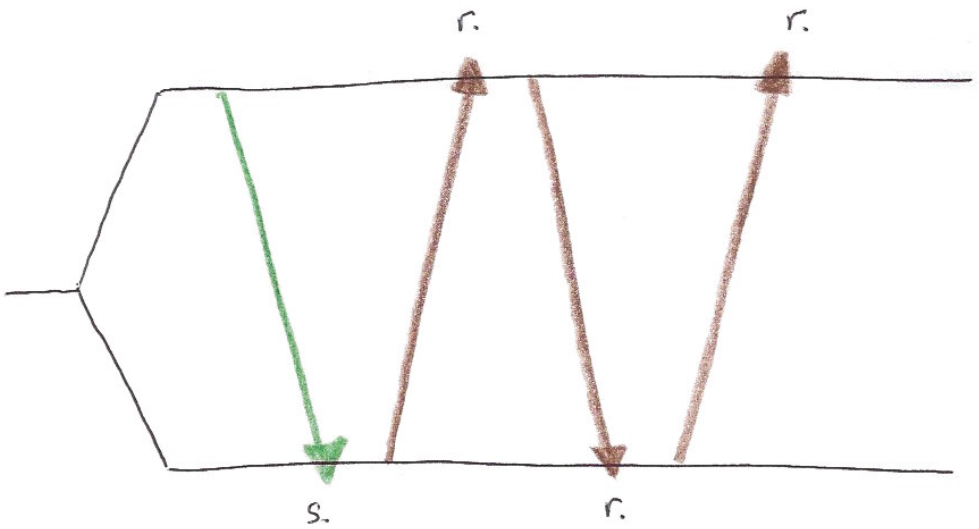
Sync vs. Reintegrate (3)

	sync	reintegrate
base node	on source branch	on target branch
skip cherry-picked revs?	yes	no
fill in partly-merged subtrees?	yes	no
handle local mods in the WC?	yes	no

Same Direction = Sync



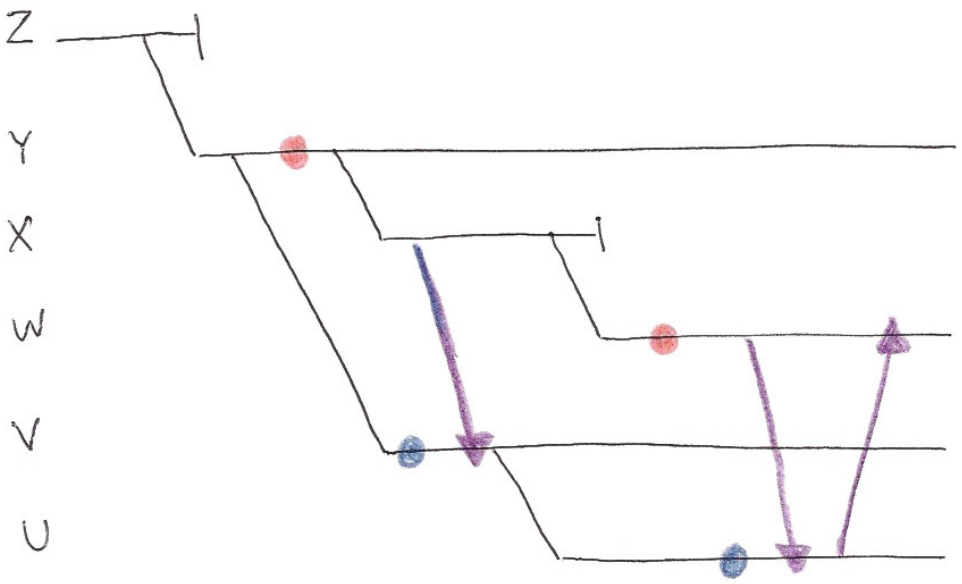
Change Direction = Reintegrate



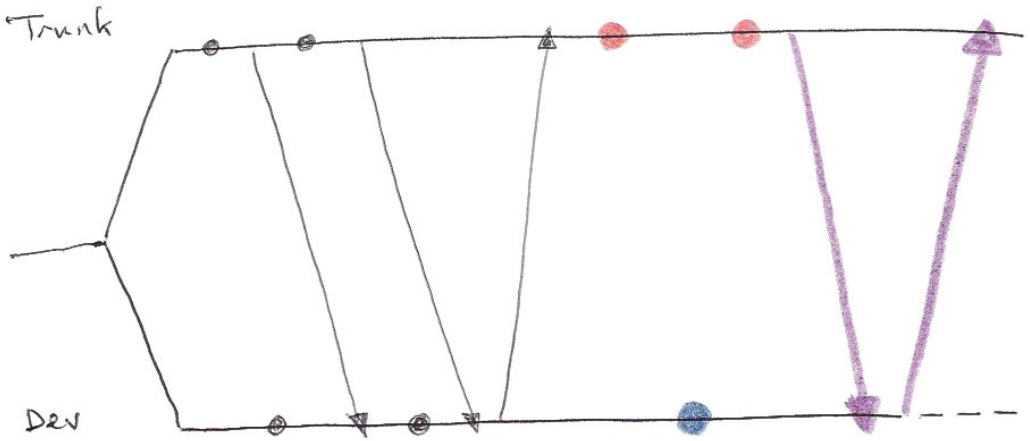
How does it work?

- Find the best base
  - Find the latest rev of A synced to B and of B synced to A.
  - Choose the more recent base.
- Select which code to run
  - Run sync if base on source
  - Run reintegrate if base on target

Confusing (1)



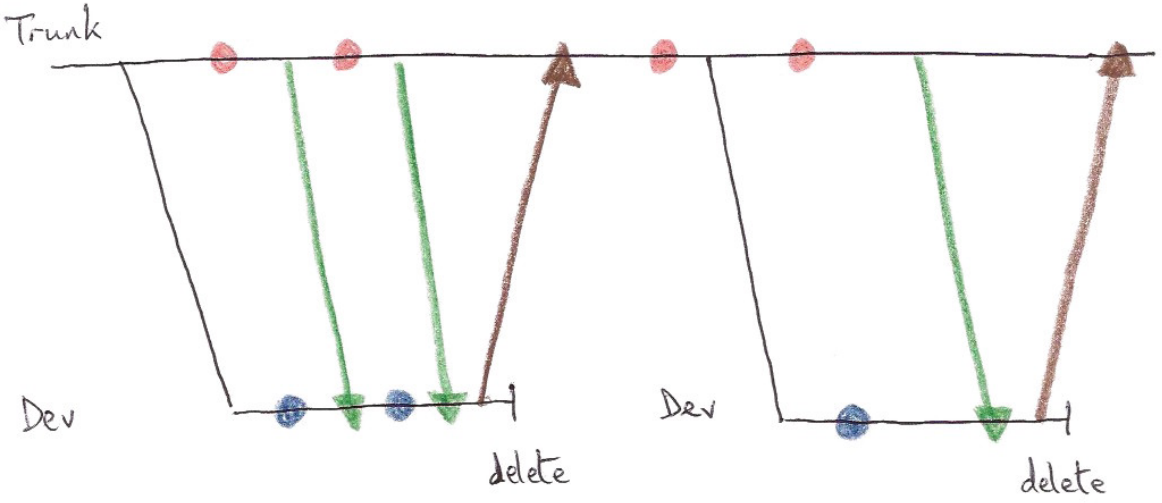
Continue Work on Branch



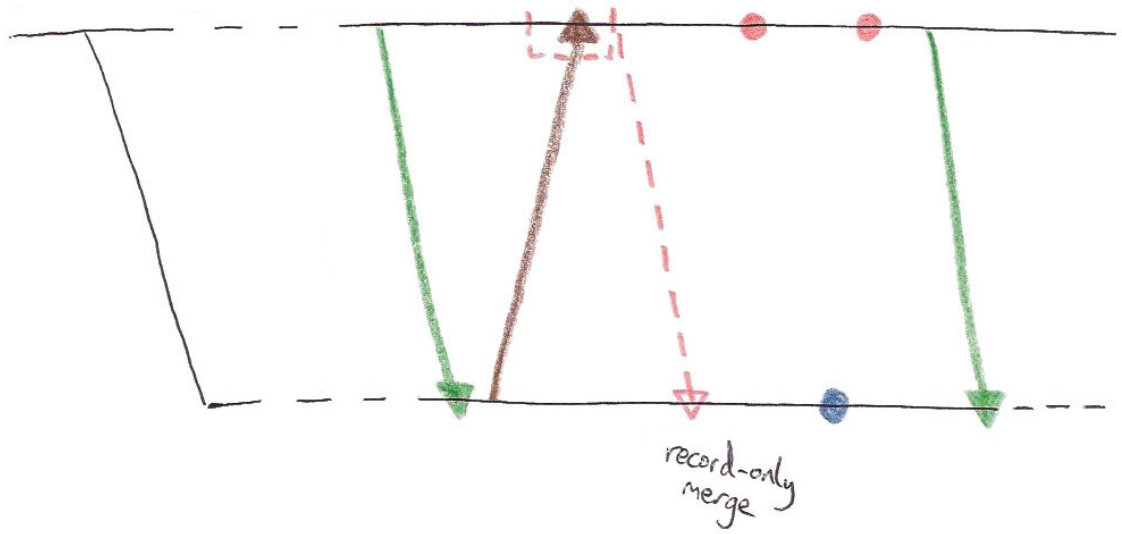
Continue – Options in 1.7

- Delete
- Keep Alive

Continue – Delete and Re-branch



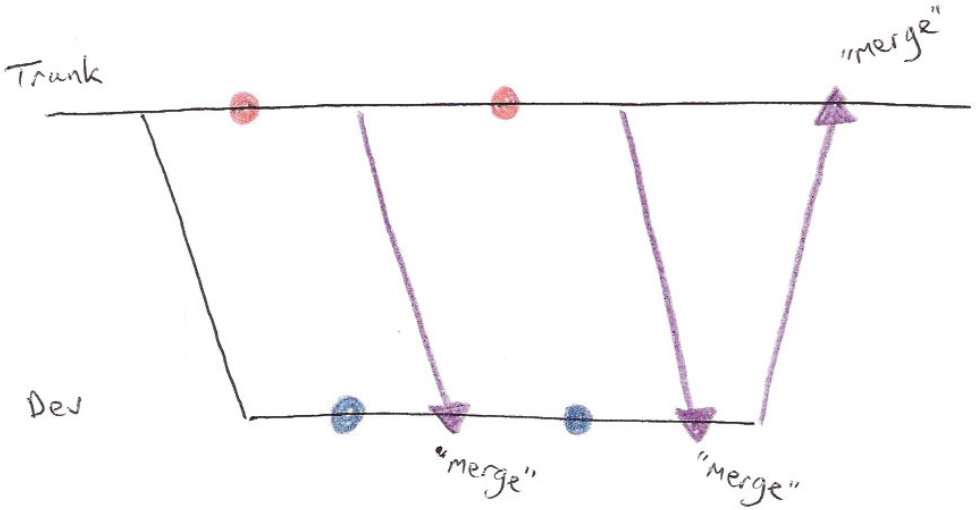
Continue – Keep-Alive



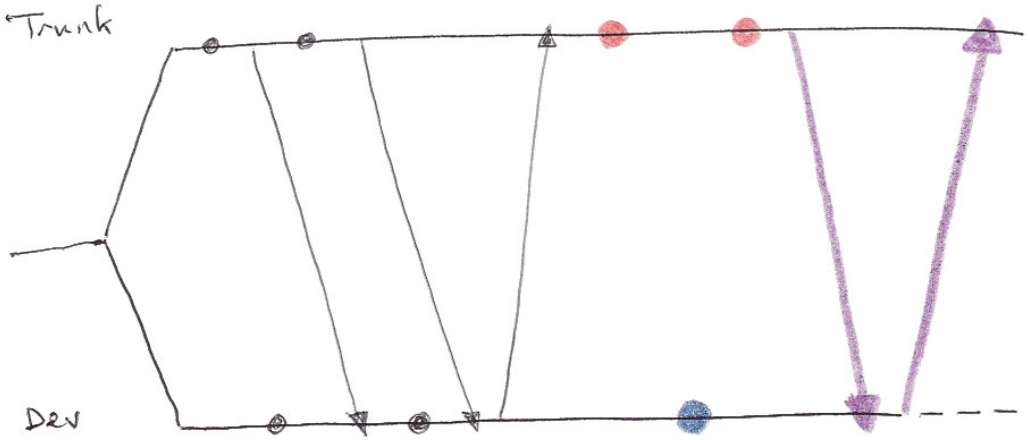
Limitations

- not yet symmetric inside
  - limitations NOT symmetric
  - results are symmetric
- reintegrate (change-direction) merges
  - no cherry-picked revisions
  - no subtree-specic mergeinfo
  - no local mods in WC
  - no sparse WC
- in line with usage & best practice

Automatic Merging



Automatic Merging



History of Merging

- 1.0 Diff & Apply
- 1.5 Merge Tracking
- 1.5 Reintegrate
- 1.8 Symmetric

Summary

- Automatically selects 'reintegrate' mode
- To-and-fro merging
- Continue development after reintegrating

Thank you

www.wandisco.com