Merge and Performance Improvements

To-and-Fro Merging



Introduction	Sync & Reintegrate 0000000000000	Why Symmetric?	Implementation 0000	Results 000000	Next 0000	Summary
					-	
	who much o'' on	±				

The "reintegrate" option



Introduction	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0000	Summary

The symmetric merge



Introduction	Sync & Reintegrate 0000000000000	Why Symmetric?	Implementation 0000	Results 000000	Next 0000	Summary
Outline						



- 2 Why Symmetric?
- Implementation











Introduction	Sync & Reintegrate ●0000000000000	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0000	Summary
The Feature B	ranch Pattern					
Outline						

Sync & Reintegrate

• The Feature Branch Pattern

• Why Sync & Reintegrate?

2 Why Symmetric?

Implementation

④ Results



	Sync & Reintegrate 0●000000000000	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0000	Summary
The Feature B	ranch Pattern					
Feature	Branch					



	Sync & Reintegrate 00●0000000000	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0000	Summary
The Feature B	ranch Pattern					
		~				

Feature Branch with Sync



	Sync & Reintegrate 000000000000	Why Symmetric? 00000000000000	l mplementation 0000	Results 000000	Next 0000	Summary
The Feature B	ranch Pattern					
Release	Branch					



Introduction	Sync & Reintegrate 0000●00000000	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0000	Summary
The Feature B	ranch Pattern					
Other F	Patterns					



	Sync & Reintegrate 000000000000	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0000	Summary	
The Feature B	Branch Pattern						ĺ
Other F	Patterns						





	Sync & Reintegrate 0000●00000000	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0000	Summary
The Feature B	ranch Pattern					
Other F	Patterns					







	Sync & Reintegrate 0000●00000000	Why Symmetric? 000000000000000	Implementation 0000	Results 000000	Next 0000	Summary
The Feature B	ranch Pattern					
Other F	Patterns					









	Sync & Reintegrate 00000●0000000	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0000	Summary
The Feature B	ranch Pattern					
Diff &	Apply					



	Sync & Reintegrate 000000●000000	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0000	Summary
The Feature B	Franch Pattern					
Merge V	Which Chang	es?				



	Sync & Reintegrate 0000000●00000	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0000	Summary
The Feature B	ranch Pattern					
Subsyst	ems					

what changes are needed?

diff & apply

Introduction	Sync & Reintegrate ○○○○○○○●○○○○	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0000	Summary
Why Sync & R	eintegrate?					
Outline						

Sync & Reintegrate The Feature Branch Patte Why Sync & Reintegrate?

2 Why Symmetric?

Implementation

④ Results



	Sync & Reintegrate ○○○○○○○○●○○○	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0000	Summary
Why Sync & R	eintegrate?					
Sync &	Reintegrate					



	Sync & Reintegrate ○○○○○○○○○○○○○	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0000	Summary
Why Sync & R	eintegrate?					
How Sv	nc Works					



Youngest Common Ancestor all changes on source target's mergeinfo eligible changes 3-way base 3-way source-right record mergeinfo

0	0
A1, A2	A1, A2, A3, A4
nil	A:1-2
A1, A2	A3, A4
pred(A1)	pred(A3)
A2	A4
"A:1-2"	"A:3-4"

	Sync & Reintegrate ○○○○○○○○○○	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0000	Summary
Why Sync & R	eintegrate?					
How Sv	nc Works					



Youngest Common Ancestor all changes on source target's mergeinfo eligible changes 3-way base 3-way source-right record mergeinfo O O A1, A2 A1, A2, A3, A4 nil A:1-2 A1, A2 A3, A4 pred(A1) pred(A3) A2 A4 "A:1-2" "A:3-4"

	Sync & Reintegrate ○○○○○○○○○○	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0000	Summary
Why Sync & R	eintegrate?					
How Sv	nc Works					



Youngest Common AncestorOOall changes on sourceA1, A2A1, A2, A3, A4target's mergeinfonilA:1-2eligible changesA1, A2A3, A43-way basepred(A1)pred(A3)3-way source-rightA2A4record mergeinfo"A:1-2""A:3-4"

	Sync & Reintegrate ○○○○○○○○○○○○○	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0000	Summary
Why Sync & R	eintegrate?					
11 C.						

How Sync Works



Youngest Common Ancestor all changes on source target's mergeinfo eligible changes 3-way base 3-way source-right record mergeinfo O O A1, A2 A1, A2, A3, A4 nil A:1-2 A1, A2 A3, A4 pred(A1) pred(A3) A2 A4 "A:1-2" "A:3-4"

	Sync & Reintegrate ○○○○○○○○○●○○	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0000	Summary
Why Sync & R	eintegrate?					
	2.4.4					

How Sync Works



Youngest Common Ancestor all changes on source target's mergeinfo eligible changes 3-way base 3-way source-right record mergeinfo

0	0
A1, A2	A1, A2, A3, A4
nil	A:1-2
A1, A2	A3, A4
pred(A1)	pred(A3)
A2	A4
"A:1-2"	"A:3-4"

	Sync & Reintegrate ○○○○○○○○○○○	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0000	Summary
Why Sync & R	eintegrate?					
		1				

How Reintegrate Works



Youngest Common Ancestor all changes on *target source*'s mergeinfo eligible changes 3-way base 3-way source-right record mergeinfo O A1, A2, A3, A4 A:1-2 diff(A2, B4) A2 B4 "B:1-4"

	Sync & Reintegrate ○○○○○○○○○○○	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0000	Summary
Why Sync & R	eintegrate?					
		1				

.

How Keintegrate Works



Youngest Common Ancestor all changes on target source's mergeinfo

. A1, A2, A3, A4 A:1-2

	Sync & Reintegrate ○○○○○○○○○○	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0000	Summary
Why Sync & R	eintegrate?					
Haw Da	intograto M/a	rlice				

How Reintegrate Works



	Sync & Reintegrate ○○○○○○○○○○○	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0000	Summary
Why Sync & R	eintegrate?					
Differen	ces					

Table: Differences

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

	Sync & Reintegrate 0000000000000	Why Symmetric? ●000000000000000	Implementation 0000	Results 000000	Next 0000	Summary
Reintegrate can	n be confusing					
Outline						

🕕 Sync & Reintegrate

2 Why Symmetric?

• Reintegrate can be confusing

- Continue after reintegrate
- To-and-Fro Merging

Implementation





27

		Sync & Reintegrate 0000000000000	Why Symmetric? 000000000000000000000000000000000000	Implementation 0000	Results 000000	Next 0000	Summary
l	Reintegrate can	be confusing					
	Confusin	g					



Introduction	Sync & Reintegrate 0000000000000	Why Symmetric?	Implementation 0000	Results 000000	Next 0000	Summary
Continue after	reintegrate					
Outline						

🕕 Sync & Reintegrate

2 Why Symmetric?

Reintegrate can be confusing

Continue after reintegrate

To-and-Fro Merging

Implementation

④ Results



	Sync & Reintegrate 0000000000000	Why Symmetric?	Implementation 0000	Results 000000	Next 0000	Summary
Continue after	reintegrate					
Continu	е					



Introduction	Sync & Reintegrate 0000000000000	Why Symmetric?	Implementation 0000	Results 000000	Next 0000	Summary
Continue after	reintegrate					
Continu	е					

Delete

Keep Alive

Introduction	Sync & Reintegrate 0000000000000	Why Symmetric? 000000000000000000000000000000000000	Implementation 0000	Results 000000	Next 0000	Summary
Continue after	reintegrate					
Continu	е					

- Delete
- Keep Alive

	Sync & Reintegrate 0000000000000	Why Symmetric?	Implementation 0000	Results 000000	Next 0000	Summary
Continue after	reintegrate					
Continu	e					



	Sync & Reintegrate 0000000000000	Why Symmetric? ०००००●००००००००	Implementation 0000	Results 000000	Next 0000	Summary
Continue after	reintegrate					
Continu	0					



	Sync & Reintegrate 0000000000000	Why Symmetric? ○○○○○○●○○○○○○	Implementation 0000	Results 000000	Next 0000	Summary
Continue after	reintegrate					
Delete &	& re-branch					



	Sync & Reintegrate 0000000000000	Why Symmetric? ○○○○○○●○○○○○○	Implementation 0000	Results 000000	Next 0000	Summary
Continue after	reintegrate					
Delete &	& re-branch					


	Sync & Reintegrate 0000000000000	Why Symmetric?	Implementation 0000	Results 000000	Next 0000	Summary
Continue after	reintegrate					
Keep A	live					



Awkward extra step

Doesn't work properly, in general

	Sync & Reintegrate 0000000000000	Why Symmetric?	Implementation 0000	Results 000000	Next 0000	Summary
Continue after	reintegrate					
Keep A	live					



• Awkward extra step

Doesn't work properly, in general

	Sync & Reintegrate 0000000000000	Why Symmetric? ○○○○○○●○○○○○○	Implementation 0000	Results 000000	Next 0000	Summary
Continue after	reintegrate					
Keep A	live					



- Awkward extra step
- Doesn't work properly, in general

	Sync & Reintegrate 0000000000000	Why Symmetric? ००००००००●०००००	Implementation 0000	Results 000000	Next 0000	Summary
Continue after	reintegrate					
Keep A	ive					



	Sync & Reintegrate 0000000000000	Why Symmetric?	Implementation 0000	Results 000000	Next 0000	Summary
Continue after	reintegrate					
Keen-al	ive problem					



Introduction	Sync & Reintegrate 0000000000000	Why Symmetric?	Implementation 0000	Results 000000	Next 0000	Summary
To-and-Fro Me	rging					
Outline						

🕕 Sync & Reintegrate

Why Symmetric?

- Reintegrate can be confusing
- Continue after reintegrate
- To-and-Fro Merging

Implementation

④ Results



42

	Sync & Reintegrate 0000000000000	Why Symmetric?	Implementation 0000	Results 000000	Next 0000	Summary
To-and-Fro Me	rging			_	_	_

Merge the same way with *sync*



	Sync & Reintegrate 0000000000000	Why Symmetric?	Implementation 0000	Results 000000	Next 0000	Summary
To-and-Fro Me	rging					

Merge the same way with sync



	Sync & Reintegrate 0000000000000	Why Symmetric? ०००००००००००	Implementation 0000	Results 000000	Next 0000	Summary
To-and-Fro Me	rging			_	_	_

Merge the opposite way with *reintegrate*



	Sync & Reintegrate 0000000000000	Why Symmetric? ○○○○○○○○○○○○	Implementation 0000	Results 000000	Next 0000	Summary
To-and-Fro Mei	rging					_

Merge the opposite way with reintegrate



	Sync & Reintegrate 0000000000000	Why Symmetric?	Implementation 0000	Results 000000	Next 0000	Summary
To-and-Fro Mei	ging					
Surprise To-and-Fro	! A∣ready Works					

• Same direction again

- o sync
- Change direction
 - reintegrate

	Sync & Reintegrate 0000000000000	Why Symmetric?	Implementation 0000	Results 000000	Next 0000	Summary
To-and-Fro Me	rging					
Surprise To-and-Fro	! Already Works					

- Same direction again
 - sync
- Change direction
 - reintegrate

Introduction	Sync & Reintegrate 0000000000000	Why Symmetric?	Implementation 0000	Results 000000	Next 0000	Summary
To-and-Fro Mei	ging					
Surprise To-and-Fro	! A∣ready Works					

- Same direction again
 - sync
- Change direction
 - reintegrate

	Sync & Reintegrate 0000000000000	Why Symmetric?	Implementation 0000	Results 000000	Next 0000	Summary
To-and-Fro Mei	ging					
Surprise To-and-Fro	! A∣ready Works					

- Same direction again
 - sync
- Change direction
 - reintegrate

Introduction	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation ●000	Results 000000	Next 0000	Summary			
Symmetric Algorithm									
Outline									



Why Symmetric?

Implementation

Symmetric Algorithm

4 Results



51

	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0●00	Results 000000	Next 0000	Summary		
Symmetric Algorithm								
Algorith	m							

• Find the best base

- Find the latest rev of A synced to B and of B synced to A.
- Choose the more recent base.
- Then, ideally...
 - Identify cherry-picks.
 - Break into 3-way merges, skipping the cherry-picks
 - Perform the 3-way merges and mergeinfo addition.
- o but currently...
 - Run "sync" if base on source
 - Run "reintegrate" if base on target

52

	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0●00	Results 000000	Next 0000	Summary			
Symmetric Algorithm									
Algorith	m								

Find the best base

- Find the latest rev of A synced to B and of B synced to A.
- Choose the more recent base.

• Then, ideally...

- Identify cherry-picks.
- Break into 3-way merges, skipping the cherry-picks
- Perform the 3-way merges and mergeinfo addition.

o but currently...

- Run "sync" if base on source
- Run "reintegrate" if base on target

	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0●00	Results 000000	Next 0000	Summary		
Symmetric Algorithm								
Algorith	m							

- Find the best base
 - Find the latest rev of A synced to B and of B synced to A.
 - Choose the more recent base.
- Then, ideally...
 - Identify cherry-picks.
 - Break into 3-way merges, skipping the cherry-picks
 - Perform the 3-way merges and mergeinfo addition.
- o but currently...
 - Run "sync" if base on source
 - Run "reintegrate" if base on target

	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0●00	Results 000000	Next 0000	Summary		
Symmetric Algorithm								
Algorith	m							

- Find the best base
 - Find the latest rev of A synced to B and of B synced to A.
 - Choose the more recent base.
- Then, ideally...
 - Identify cherry-picks.
 - Break into 3-way merges, skipping the cherry-picks.
 - Perform the 3-way merges and mergeinfo addition.
- o but currently...
 - Run "sync" if base on source
 - Run "reintegrate" if base on target

	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0●00	Results 000000	Next 0000	Summary		
Symmetric Algorithm								
Algorith	m							

- Find the best base
 - Find the latest rev of A synced to B and of B synced to A.
 - Choose the more recent base.
- Then, ideally...
 - Identify cherry-picks.
 - Break into 3-way merges, skipping the cherry-picks.
 - Perform the 3-way merges and mergeinfo addition.
- but currently...
 - Run "sync" if base on source
 - Run "reintegrate" if base on target

	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0●00	Results 000000	Next 0000	Summary		
Symmetric Algorithm								
Algorith	m							

- Find the best base
 - Find the latest rev of A synced to B and of B synced to A.
 - Choose the more recent base.
- Then, ideally...
 - Identify cherry-picks.
 - Break into 3-way merges, skipping the cherry-picks.
 - Perform the 3-way merges and mergeinfo addition.
- but currently...
 - Run "sync" if base on source
 - Run "reintegrate" if base on target

	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0●00	Results 000000	Next 0000	Summary			
Symmetric Algorithm									
Algorith	m								

- Find the best base
 - Find the latest rev of A synced to B and of B synced to A.
 - Choose the more recent base.
- Then, ideally...
 - Identify cherry-picks.
 - Break into 3-way merges, skipping the cherry-picks.
 - Perform the 3-way merges and mergeinfo addition.
- o but currently...
 - Run "sync" if base on source
 Run "reintegrate" if base on target

	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0●00	Results 000000	Next 0000	Summary			
Symmetric Algorithm									
Algorith	m								

- Find the best base
 - Find the latest rev of A synced to B and of B synced to A.
 - Choose the more recent base.
- Then, ideally...
 - Identify cherry-picks.
 - Break into 3-way merges, skipping the cherry-picks.
 - Perform the 3-way merges and mergeinfo addition.
- o but currently...
 - Run *"sync"* if base on source
 - Run "reintegrate" if base on target

	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0●00	Results 000000	Next 0000	Summary		
Symmetric Algorithm								
Algorith	m							

- Find the best base
 - Find the latest rev of A synced to B and of B synced to A.
 - Choose the more recent base.
- Then, ideally...
 - Identify cherry-picks.
 - Break into 3-way merges, skipping the cherry-picks.
 - Perform the 3-way merges and mergeinfo addition.
- o but currently...
 - Run "sync" if base on source
 - Run "reintegrate" if base on target

	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0●00	Results 000000	Next 0000	Summary	
Symmetric Algorithm							
Algorith	m						

- Find the best base
 - Find the latest rev of A synced to B and of B synced to A.
 - Choose the more recent base.
- Then, ideally...
 - Identify cherry-picks.
 - Break into 3-way merges, skipping the cherry-picks.
 - Perform the 3-way merges and mergeinfo addition.
- o but currently...
 - Run "sync" if base on source
 - Run "reintegrate" if base on target

	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 00●0	Results 000000	Next 0000	Summary	
Symmetric Algorithm							
Limitati	ons						

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

	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 000●	Results 000000	Next 0000	Summary		
Symmetric Algorithm								
C	C	ata a						

Sync before reintegrating



Introduction	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0000	Results ●00000	Next 0000	Summary
Results						
Outline						



- Why Symmetric?
- Implementation







	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0000	Results 0●0000	Next 0000	Summary
Results						

Leave out *-reintegrate*

```
+ svn merge --reintegrate ^/B A  # v1.7
--- Merging differences between repository URLs into 'A':
A  A/pickle
--- Recording mergeinfo for merge between repository URLs into
'A':
U  A
```



	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0000	Results 00●000	Next 0000	Summary
Results						_

Use the same *merge* command



		Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0000	Results 000●00	Next 0000	Summary
l	Results						
	<u> </u>						

On-line Help for svn merge

```
$ svn help merge # v1.7
merge: Merge changes into a working copy.
usage:
  1. merge SOURCE[@REV] [TARGET_WCPATH]
     (the "sync" merge)
 2. merge [-c M[,N...] | -r N:M ...] SOURCE...
     (the "cherry-pick" merge)
 3. merge --reintegrate SOURCE[@REV] [TARGET_WCPATH]
     (the "reintegrate" merge)
  4. merge SOURCE1[@N] SOURCE2[@M] ...
     (the "2-URL" merge)
. . .
```

	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0000	Results 000●00	Next 0000	Summary
Results						
O 11						

On-line Help for svn merge

```
$ svn help merge # v1.8
merge: Merge changes into a working copy.
usage:
    1. merge SOURCE[@REV] [TARGET_WCPATH]
      (the "automatic" merge)
    2. merge [-c M[,N...] | -r N:M ...] SOURCE...
      (the "cherry-pick" merge)
    3. merge SOURCE1[@N] SOURCE2[@M] ...
      (the "2-URL" merge)
```

	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0000	Results 0000●0	Next 0000	Summary
Results						

Continue after reintegrating



	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0000	Results 00000●	Next 0000	Summary
Results						
Usability	y tweaks					

Catch source/target mismatch

- source unrelated to target
- source same as target
- source is a subtree of target (or vice-versa)

	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next ●000	Summary
The Next Step						
Outline						

- 🕕 Sync & Reintegrate
- Why Symmetric?
- Implementation





• The Next Step

	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0●00	Summary
The Next Step						
History						

• 1.0 Diff & Apply

- 1.5 Merge Tracking
- 1.5 Reintegrate
- 1.8 Symmetric
- Next step
| | Sync & Reintegrate
0000000000000 | Why Symmetric?
00000000000000 | Implementation
0000 | Results
000000 | Next
0●00 | Summary |
|---------------|-------------------------------------|----------------------------------|------------------------|-------------------|--------------|---------|
| The Next Step | | | | | | |
| History | | | | | | |

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

	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0●00	Summary
The Next Step						
History						

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

	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0●00	Summary
The Next Step						
History						

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

	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 0●00	Summary
The Next Step						
History						

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

	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 00●0	Summary
The Next Step						
Rename	Tracking De	sion				

Redesign

 assume we'll be able to tell merge algo which src node matches which tgt node

Modularize

- a merge algorithm
- a provider of rename info
- a module to apply changes to WC
- a mergeinfo read/write module

Refactor

- use merge logic for merge
- use merge logic for update & switch
- move merge logic to the server?

	Sync & Reintegrate 0000000000000	Why Symmetric? 000000000000000	lmplementation 0000	Results 000000	Next 00●0	Summary
The Next Step						
Rename	Tracking De	sign				

Redesign

 assume we'll be able to tell merge algo which src node matches which tgt node

Modularize

- a merge algorithm
- a provider of rename info
- a module to apply changes to WC
- a mergeinfo read/write module

Refactor

- use merge logic for merge
- use merge logic for update & switch
- move merge logic to the server?

	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	Implementation 0000	Results 000000	Next 00●0	Summary
The Next Step						
Rename	Tracking De	sign				

Redesign

• assume we'll be able to tell merge algo which src node matches which tgt node

Modularize

- a merge algorithm
- a provider of rename info
- a module to apply changes to WC
- a mergeinfo read/write module

Refactor

- use merge logic for merge
- use merge logic for update & switch
- move merge logic to the server?

	Sync & Reintegrate 0000000000000	Why Symmetric? 000000000000000	Implementation 0000	Results 000000	Next 000●	Summary
The Next Step						
3-way T	ree Merge					

- in: rename tracking info
- responsible for
 - moves / renames
 - tree conflicts
- the rest (file merging) stays the same

	Sync & Reintegrate 0000000000000	Why Symmetric? 00000000000000	lmplementation 0000	Results 000000	Next 0000	Summary
				_	_	_
Summa	ry					

- No more "-reintegrate": it's automatic
- To-and-Fro Merging
- Mergeinfo summary
- Sanity checks