

## TOOLS

[Twitter](#) [Like 0](#) [Share](#)

### Three-Way Merging: A Look Under the Hood

By Pablo Santos, December 24, 2013

[Post a Comment](#)

Automating three-way code merges requires considerable sophistication from the version control system.

#### Three-Way Merge Tool Layout

When you run a three-way merge tool, the typical layout of the tool is as illustrated in Figure 5:

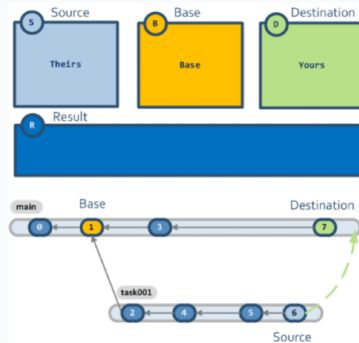


Figure 5.

Good three-way merge tools show four panels:

- "Theirs" (the source of the merge, see the branch diagram in Figure 6), base, and "Yours" (the destination of the merge) in the upper panel.
- The result of the merge in the lower panel.

To me this four-panel representation of the three-way merge is the most intuitive, but some tools present this alternative layout with only three panels (Figure 6):

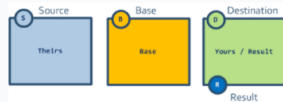


Figure 6.

In this layout, the "destination/yours" (or working copy) and the result of the merge are displayed together.

#### The Importance of Merge Tracking

To run effective three-way merges, you not only need a good three-way merge tool, you need an effective merge engine in your version control tool.

In fact, part of the mission of version control should be to correctly calculate the common ancestor/base on any three-way merge. When people say "git is very good at merging," what they mean is "git is very good tracking the merge history, hence calculating the common ancestor for each file." In my VCS work, we put a lot of effort into the merge engine and calculating the nearest common ancestor.

Let's go back to the three-way merge with a manual conflict that we just solved, and let's check out the branching structure (well, at least one very simple branching structure); see Figure 7:

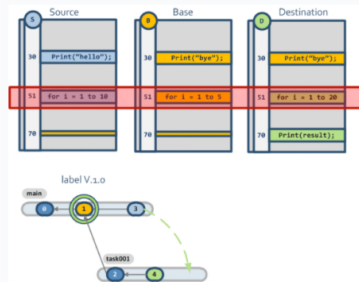


Figure 7.

- Changeset 3: someone working on the "main" branch performed the change of the `Print("Hello")` line
- Changeset 4: meanwhile, on branch "task001," you were doing the addition of the `Print(result);` at line 70.
- And you both modified line 51.

Now, you want to merge the latest changes coming from "main" into your branch "task001." The version control system will find the nearest common ancestor of changesets "3" and "4" and it will use the graph above. The result in this simple case is changeset "1." The "base" version will be retrieved from changeset "1" to do the merge.

Once you solve the manual conflict on line 51, you will be checking in on the branch "task001" and creating a new changeset "5" as in Figure 8:



Figure 8.

Now development continues; somebody will be creating more changes on "main" while you perform a new checkin on "task001." And then you decide you have to merge "task001" back to "main" (Figure 9):



Figure 9.

#### Tools Recent Articles

- Dr. Dobb's Archive
- Jolt Awards 2015: Coding Tools
- Thriving Among the APIs
- Building Node.js Projects in Visual Studio
- Building Portable Games in C++

#### Most Popular

[Stories](#) [Blogs](#)

- RESTful Web Services: A Tutorial
- Developer Reading List: The Must-Have Books for JavaScript
- iOS Data Storage: Core Data vs. SQLite
- Building GUI Applications in PowerShell
- MongoDB with C#: Deep Dive

#### Upcoming Events

[Live Events](#) [WebCasts](#)

- [Video] Shaping the Future of IT: CIO Lightning Series - Interop 19 - Interop 2019
- [Video] An (Irrational) Approach to Interoperability - Interop19 - InteropTX 2018
- Get Insights: Cisco vs Microsoft & More at EC20 Orlando - Enterprise Connect 2020
- Cisco & Amazon Web Services to Keynote at Enterprise Connect 2020 - Enterprise Connect 2020

#### Featured Reports

What's this?

- Rethinking Enterprise Data Defense
- The Definitive Guide to Managed Detection and Response (MDR)
- 2019 SANS Incident Response Survey Report
- 2019 Threat Hunting Report
- Getting Started With Emerging Technologies

[More >>](#)

#### Featured Whitepapers

What's this?

- Getting Started With Emerging Technologies
- [Infographic] Are You Maximizing Value of the Cloud?
- 2019 State of DevOps
- How to Migrate a Data Center Without Compromising Performance
- Annual Phishing Report 2019

[More >>](#)

#### Most Recent Premium Content

##### Digital Issues

- 2014
  - Dr. Dobb's Journal
  - November - Mobile Development
  - August - Web Development
  - May - Testing
  - February - Languages

- Dr. Dobb's Tech Digest
  - DevOps
  - Open Source
  - Windows and .NET programming
  - The Design of Messaging Middleware and 10 Tips from Tech Writers
  - Parallel Array Operations in Java 8 and Android on x86: Java Native Interface and the Android Native Development Kit

- 2013
  - January - Mobile Development
  - February - Parallel Programming
  - March - Windows Programming
  - April - Programming Languages
  - May - Web Development
  - June - Database Development
  - July - Testing
  - August - Debugging and Defect Management
  - September - Version Control
  - October - DevOps
  - November - Really Big Data
  - December - Design

- 2012
  - January - C & C++
  - February - Parallel Programming
  - March - Microsoft Technologies
  - April - Mobile Development
  - May - Database Programming
  - June - Web Development
  - July - Security
  - August - ALM & Development Tools
  - September - Cloud & Web Development
  - October - JVM Languages
  - November - Testing
  - December - DevOps

The version control system will have to calculate the base/common ancestor between "6" and "7."

The common ancestor will be "3" as Figure 10 shows:



Figure 10.

Note that "3" is the common ancestor because the version control is considering the merge that happened between "3" and "5," which you completed before.

What is the benefit of this merge tracking?

Well, if the merge link between "3" and "5" wasn't tracked (as used to happen with old version control systems), then the base would be "1" again, and you would have to again solve the manual conflict you already solved before. However, if the version control system does its job correctly, the ancestor will be identified as "3" and you won't have to waste time on conflicts you already solved.

Now, what would happen here with two-way merge? You would have to solve every difference manually because in two-way merges, every single modification is a conflict since the merge facility doesn't have a way to solve conflicts automatically.

### Conclusion

Often, the questions regarding three-way merge are asked by developers using version control systems lacking good merge tracking such as CVS, Microsoft Visual SourceSafe, and even old versions of Subversion.

Understanding how three-way merge works and why it is so important to have a good merge engine like those in new distributed version control systems is key when looking for a replacement to an aging SCM.

[Pablo Santos](#) is a blogger for Dr. Dobb's and an expert on the operations of version control systems.

Previous 1 2

### Related Reading

#### News

Tools To Build Payment-Enabled Mobile Apps  
Application Intelligence For Advanced Dummies  
Google's Data Processing Model Hardens Up  
Boost.org Committee Battles Library Log-Jam

[More News»](#)

#### Video

The Most Underused Compiler Switches in Visual C++  
Jolt Awards: The Best Books  
Jolt Awards: The Best Programming Utilities  
Developer Reading List

[More Slideshows»](#)

#### Most Popular

Parsing XML Files in .NET Using C#  
Lambdas and Streams in Java 8 Libraries  
Building Scalable Web Architecture and Distributed Systems  
MongoDB with C#: Deep Dive

[More Popular»](#)

### More Insights

#### White Papers

Getting Started With Emerging Technologies  
[Infographic] Are You Maximizing Value of the Cloud?

[More >>](#)

#### Reports

The Future of Network Security is in the Cloud  
Rethinking Enterprise Data Defense

[More >>](#)

#### Webcasts

Enterprise IoT: Rise of the Unmanaged Devices  
Threat & Performance Management: 2 Key Data Sources

[More >>](#)

[INFO-LINK](#)

[Login or Register to Comment](#)

#### DISCOVER MORE FROM INFORMA TECH

InformationWeek  
Dark Reading  
Network Computing

Interop  
Data Center Knowledge  
IT Pro Today

#### WORKING WITH US

Contact Us  
About Us  
Advertise

#### FOLLOW DR. DOBB'S ON SOCIAL



[Home](#) [Cookie Policy](#) [Privacy](#) [Terms](#)

Copyright © 2019 Informa PLC. Informa PLC is registered in England and Wales with company number 8860726 whose registered and head office is 5 Howick Place, London, SW1P 1WG.