

Distributed Version Control Systems

Context

Version Control, also known as Source Control or Revision Control, is the process of managing multiple versions of a piece of information. It is most commonly used in software development, for which source code is a prime element.

Managing the source code throughout time, with multiple cycles and contributors, raises several practical problems that can be major sources of friction and drudgery – thus a serious drain on productivity. Ultimately this means that every moment spent on these problems is a moment not spent on getting the design and function of your project right.

Version Control Systems seek to provide an automated way to track project history over time and to collaborate easily with others. To a large extent these systems determine how easily people can contribute to a project, how new feature development is co-ordinated, how often separate development lines are merged, how code is reviewed and how the support of already released code is organized.

Challenges

The most commonly-used Version Control Systems are CVS, SourceSafe and Subversion, but anyone who ever worked with these systems knows they fail to impress:

- **Mandatory network connection.** Most operations require interaction with the central repository, usually located on a remote server. This means they are not available when you're offline and sometimes even small operations are inexplicably laggy.
- **Commit apprehension.** A developer can only checkpoint his work by committing his changes into the central repository, where it becomes immediately visible for everybody else. For fear of disrupting everybody's work, team members often go days or weeks without committing anything.
- **Merge nightmare.** It's difficult to reconciling changes when merging across branches. This often leads developers to avoid developing new functionalities in separate branches and rather work on the trunk instead, making it much harder to keep a stable version.

All these drawbacks mean that most developers end up writing code without truly benefitting of version control. The good news is that the Version Control space is undergoing a renaissance right now thanks to the increasing popularity of Distributed Version Control Systems such as Bazaar, Git or Mercurial.

Tips & Tricks

What makes this new generation of systems fundamentally different is a new way of thinking about Version Control:

- **Distributed.** Rather than a single, central repository on which clients synchronize, each peer's working copy is itself a full-fledged repository.

- **Better history tracking.** By supporting merge tracking from the ground up, and the more fundamental fact that each revision knows its parents, these systems can reconstruct a Direct Acyclic Graph (DAG) of the revision history. There is no need for fancy metadata, rename tracking and so forth. The only thing you need to store is the state of the tree before and after each change.

The positive outcome of this:

- **Private work.** It's possible to keep track of your ongoing work by committing it locally first, in small steps, until the task is completed. Changes can later be propagated using push or pull operations.
- **Easy Merging.** Because the system can trace the origins of individual changes, merge algorithms actually work. And if something goes wrong, you can just roll back and try the merge again.
- **Cheap branching.** Creating and switching between branches is nearly instant. In addition, branching works well because merging works well.
- **Any workflow.** Because of its distributed nature and super branching system, you can easily implement several types of workflows. Anything from a purely ad hoc model, to a classic centralized model.
- **Fault-tolerant.** Each working copy is effectively a remote backup of the entire repository and change history, providing natural security against data loss.

Combined, the above features mean that developers spend less time on mechanical chores and more time on tasks that add value.

Not unexpectedly, there are some things you want to watch out for:

- **Chaos susceptibility.** By its very nature, Distributed Version Control Systems allow for a higher level of chaos. Yet, it is up to your team to determine the appropriate level of chaos permitted, and to enforce the right workflow early on.
- **Oddity.** If you have been using CVS, SourceSafe or Subversion for a long time you most likely need a little reeducation.
- **Raw tools.** The set of accessory tools are still a bit raw. Particularly if you're are a GUI lover.
- **Large repositories.** Initial cloning of large repositories may be slow, because all branches and revision history are copied.

Nonetheless, if you are still using a centralized system, I strongly encourage you to start exploring the possibilities of Distributed Version Control Systems.

Learn More

- Git Community Book <http://book.git-scm.com/>
- Hg Init: a Mercurial Tutorial <http://hginit.com/>
- Mercurial: The Definitive Guide <http://hgbook.red-bean.com/>
- Bazaar Documentation <http://doc.bazaar.canonical.com/latest/en/>

References

- Brian de Alwis, Jonathan Sillito. “Why are software projects moving from centralized to decentralized version control systems?”, 2009
- Ian Clatworthy, “Distributed Version Control Systems – Why and How”, 2007
- Joel Spolsky, “Distributed Version Control is here to stay, baby”, retrieved 06/05/2010 from <http://www.joelonsoftware.com/items/2010/03/17.html>
- Scott Chacon, “Why Git is better than X”, retrieved 06/05/2010 from <http://whygitisbetterthanx.com/>
- Lenz Grimmer, “Aspects and benefits of distributed version control systems (DVCS)”, retrieved 06/05/2010 from <http://lenzg.net/archives/285-Aspects-and-benefits-of-distributed-version-control-systems-DVCS.html>
- Eric Raymond, “Understanding Version-Control Systems”, retrieved 06/05/2010 from <http://www.catb.org/~esr/writings/version-control/version-control.html>
- Linus Torvalds, “Source code control the way it was meant to be!”, retrieved 06/05/2010 from <http://www.youtube.com/watch?v=4XpnKHJAok8>

Miguel Fonseca

Software Engineer