How do you deal with the light side and dark side of distributed version control systems?

Asked 16 years, 4 months ago Modified 9 years, 2 months ago

Viewed 431 times

2

I've had some discussions recently at work about switching from Subversion to a DVCS like bazaar, and I would like to get other people's opinion.

I've managed to crystallize my reluctance to do so into a simple parallel.

> Version Control can be used well or badly.

The 'light side' of version control is when you use it to keep track of your changes, are able to go back to older versions when you break stuff, and when you publish your changes so your peers can see your work-in-progress.

The 'dark side' of version control is when you don't use it properly so you don't 'checkpoint' your work by committing regularly, you keep a bunch of changes in your local checkout, and you don't share your changes with others as you make them.

Subversion makes both the light side and the dark side relatively hard. All the basics work, but few people really use branching in Subversion (beyond tagging and releasing) because merging is not straightforward at all. The support for it in Subversion itself is terrible, and there are scripts like svnmerge that make it better, but it's still not very good. So, these days, with good branching and merging support considered more and more like the necessity it is for collaborative development, Subversion doesn't match up.

On the other hand, the 'dark side' is pretty tough to follow too. You only need to be bitten once by not having your local changes commited once in a while to the online repository, and breaking your code with a simple edit you don't even remember making. So you end up making regular commits and people can see the work you're doing.

So, in the end Subversion ends up being a good middle-of-the-road VCS that, while a bit cumbersome for implementing the best practices, still makes it hard to get things very wrong.

In contrast, with a DVCS the cost of either going completely light side or dark side is a lot lower. Branching and merging is a lot simpler with these modern VCS systems. But the distributed aspect makes it easy to work in a set of local branches on your own machine, giving you the granular commits you need to checkpoint your work, possibly without ever publishing your changes so

others can see, review, and collaborate. The friction of keeping your changes in your local branches and not publishing them is typically lower than publishing them in some branch on a publically available server.

So in a nutshell, here's the question: if I give our developers at work a DVCS, how can I make sure they use it to go to the 'light side', still publish their changes in a central location regularly, and make them understand that their one week local hack they didn't want to share yet might be just the thing some other developer could use to finish a feature while the first one is on holiday?

If both the light side and the dark side of DVCS are so easy to get to, how do I keep them away from the dark side?

svn version-control

Share

Improve this question

Follow

edited Oct 8, 2015 at 7:21

willeM_ Van Onsem

475k ● 33 ● 470 ● 608

asked Aug 25, 2008 at 22:38

Thomas Vander Stichele

36.5k ● 14 ● 58 ● 60

4 Answers

Sorted by: Highest score (default)

3

If there are developers on your team that don't want to share their "one week local hack" then thats the problem, not the source control tool you are using. A better term for the "dark side" you are describing is "the wrong way" of coding for a team. Source control is a tool used to facilitate collaborative work. If your team is not clear about the fact that the goal is to share the work, then the best reason to use source control is not even applicable.

Also, I think you might be a little confused about distributed source control. There is no publishing to a central locations. Some branches are more important than others and there exists many many branches. Keeping that in mind, I think that distributed source control really works best for popular open source projects. I'm under the perception that centralized source control is still better for development withing a company or some other clearly defined entity.

Share Improve this answer

Follow

answered Aug 25, 2008 at 23:48

Nick Zalutskiy

15.4k ● 7 ● 54 ● 50

2 Yes. Social problems can't usually be solved with technical solutions, unless they're extremely narrow. When someone isn't being a team player, you need to talk to them. – wnoise Sep 18, 2008 at 22:22

Nick,

2

while I agree that the problem is 'not sharing your work', the main argument is that tools promote a certain work flow and not all tools apply to all problems with equal friction. My concern is that a DVCS makes it easier to not share your work since you don't have the drawbacks of not sharing your work you get with SVN. If the friction of not sharing is lower than the friction of not sharing (which it is in DVCS), a developer, all else being equal, might easily choose the path of least friction.

I don't think I'm confused about distributed source control. I know that there is no 'central location' by default. But most projects using DVCS still have the concept of a 'master' branch at a 'central location' that they release from. For the purpose of my question though, I only make the distinction between 'private' (only accessible by the developer) and 'public' (accessible by all other developers) branches.

Share Improve this answer

Follow

answered Aug 25, 2008 at 23:58

Thomas Vander Stichele

36.5k ● 14 ● 58 ● 60

1

You make a distinction between 'private' and 'public' branches, but the only real difference between these cases is whether the branch's repository is only available locally or company-wide. A central repository is *only one way* to have company-wide availability.

Instead, why not say that *all* repositories must be publically available throughout the company? For

example, you could make all developer run their own local VCS server, and share their branches via zeroconf.

Share Improve this answer

Follow

answered Aug 26, 2008 at 9:48

Kai

5,320 ● 5 ● 31 ● 36

0

I believe svn's merging has been somewhat overhauled in the latest release.

Share Improve this answer

Follow

answered Aug 25, 2008 at 23:49

DrPizza

18.3k ● 7 ● 42 ● 53