TFS as source-control: what do you love? what do you hate?

Asked 14 years, 7 months ago Modified 9 years, 11 months ago

Viewed 4k times

17

I've used TFS for about 18 months now and I'm really not excited about it. It seems like the worst of the current versions of SCMs on the market.

I think this thread will help people decide if TFS is for them vs. other source control systems. While TFS does a lot more than that, I think that source control is so critical to software development that any system (or combination thereof) that you pick needs to consider source control first.

What are the good things about TFS vs. other source controls -- **what does it do well** that no one else does?

What are the **things that TFS is bad at** that everyone else seems to do just fine?

tfs

Share

Improve this question

Follow

edited May 8, 2010 at 20:26

community wiki
4 revs, 2 users 100%
jcollum

11 Looks to me that you are looking for a `guilty` verdict more than a discussion. – Oded May 8, 2010 at 19:13

3 I don't know about the worst. Have you tried ClearCase? (not intending to start a flamewar here, I just have to use CC at work and am not a happy customer) I agree, source control is very critical to software development. If it takes someone a week to get hooked into something, then that is not the correct solution. Good Question – Rob Goodwin May 8, 2010 at 19:15

What do you dislike about it? – Raj Kaimal May 8, 2010 at 19:16

5 Have you ever heard of Serena Dimensions? TFS is a dream in comparison. – Oded May 8, 2010 at 19:19

1 How about VSS or CVS? I think TFS is a step up from them at least. – Tom Cabanski May 8, 2010 at 19:21

12 Answers

Sorted by: Highest score (default)

34

Pros

- Fundamentally it's a sound system. Robust and reliable.
- Integrated with work items, reporting, etc.
- The power tools are really good.

- [edit] It is improving, and has taken good jumps forwards with 2010, 2012, 2013
- TFS is highly accessible for custom tools. There's a rich API that makes it so easy to write dashboards and other tools to get at the data in TFS. And as all the data is stored in SQL, you can browse it and query it directly if need be. I've worked with many different SCMs over the years and have never found one that is so open and accessible - everything (user stories, tasks, bugs, issues, test plans, iterations, source code control & branches, builds, unit testing, continuous integration) is just there at your fingertips. This is an awesome feature of TFS. A lot of the UI failings of TFS have been addressed in a few afternoons writing tools and a dashboard for my team to use. And let's face it, if you write your own, it does *exactly* what you need.

Cons

- There is one area where the robustness fails miserably: If you apply several changes to a file (add, rename, edit) in "one go" it gets horribly confused. If you don't check in these actions separately, both TFS2005 and TFS2008 **crash** when you go to merge those changes across branches. In 2010 onwards it no longer crashes, but it often doesn't correctly check in the changes, so you have to go in and clean up a mess of missing and incorrectly named files.

- There is no standalone source control browser. It's integrated into VS, which is really annoying when you want to just work on source control items without needing to run up another copy of VS. Of course, you can give your artist a Team Explorer, but let's ask ourselves if an artist who only ever wants to view the files, check out, check in, and GLV really needs a fully blown complicated VSTS instance running to achieve it? In addition, the integration is so poor that you can't realistically use TFS from the Solution explorer (it simply lies about what you have checked out, and is so unreliable when you apply actions from that window that you soon learn to open the source control window and work in there, which defeats the point of it being integrated in the first place) [edit: The file explorer extension is excellent - close to a standalone browser - and is simple and easy to use. The main drawback of it is lack of proper integration with file commands - to rename or delete files you must remember to use the TFS submenu, or you will rename/delete *locally* and this screws up source control completely as TFS knows nothing of the changes you have made. This unfortunately means that only 'advanced' TFS users can be trusted to use it. So, essentially, it's still a case of "no stand alone browser" for most users]
- The user interface sucks (but is improving, at least on the web-access side). Sure, it works, but there is so much that could be done to make it efficient, pleasant, and more foolproof to use. e.g. [prior to

2012] When you click "check in" it ticks all remaining un-checked-in items so that if you accidentally click Check in again in future, it checks in a load of stuff you didn't want to. And after this, it would be so easy to supply an "undo last checkin" option to quickly roll it back - but there isn't one. [Edit: The UI is improved, but these specific problems are still present in VS2010, although it does now have a check-in confirmation dialog that reduces the risk of accidental checkins][edit: in 2012 it's much better, but they've gone mad and rolled all the separate TFS dialogs into a single window, which was a serious step backwards. The pending changes window doesn't work nearly as well as in 2010 - it is harder to find things, it takes more clicks to achieve the same things, and if you check in a file from anywhere all the currently 'included' files get chucked into 'excluded' so if you have several things on the go they all get mixed together]

- Workspaces. In most cases, every team member has to have essentially the same workspace mapping, slaved off a local root folder. We need 7 mappings defined, which takes about 5 minutes to set up. There is no way to push the workspace definition from the server. There is no *[edit]easy[/edit]* way to duplicate a workspace so you can use an existing one (or another users one) as a starting point. No, you have to manually re-enter all the bindings over and over and over and over. If you change your active workspace in the source control explorer, it doesn't get synced to your pending changes window, so you spend 15 minutes wondering why the file you merged from your other branch just isn't listed. [edit: This is getting better with 2010/2012, as you can see workspaces on other PCs and copy and paste them more easily, but it's still a pretty clumsy UI]

- It has changesets, but you can't bundle items into separate changesets in your pending checkins list as you can in Perforce, you can only associate them with a changeset by actually checking them in. You can really only work on one changeset at a time, or you have to separate the files out manually in your pending list as you go to check in. [still very poor in 2012]
- The merge tools are terrible. As in: they simply don't work, and unnecessarily introduce bugs into your code if you rely on the automatic merge. These tools are just as bad as they were when I first used SourceSafe in 1994. So the first thing you have to do after buying a *very costly* VSTS licence is replace the merge tools with something that actually works. And that means that every time you get a merge conflict, you must select each file. Choose to resolve the conflict and ok. Choose to use your 3rd party merge tool and ok. Then merge. Then save. Then choose to accept your merged changes. (You should be able to choose "automatic merge" and have it simply use the third party merge tool that actually works without hitting you with a barrage of pointless and annoying dialogs that always default to the wrong option) [Edit:

InVS2010 the merge tools are still awful. But the front-end UI is much improved (merging a conflict now takes a single click rather than 4 or 5 clicks - a massive improvement when you have to merge many files][In 2012 there have been further improvements, but they are still 'ok' rather than good]

- It doesn't sync between running instances of VS. So if you check in a file in one VS, another one will still list that file in your pending checkins. (it's clearly easy to sync it because any changes made by the power tools windows-explorer extension are reflected in VS instantly). [Edit: In 2012 they have fixed this problem. Now every time you switch to the pending changes view it spends 15 seconds refreshing (in 2010 it cached it and showed it instantly but it was occasionally out of date)]
- Branching is the standard way of working these days. So you'd expect the branch/merge tools to make this quick and easy. But no. [edit: Big improvements were made in 2010 and 2012, but merging is terribly supported - it is really labour intensive. Just little things like only being able to merge a contiguous set of changes, so if you want to merge 5 changes that are not contiguous you have to do them one by one, but each time you open the dialog it starts from scratch instead of remembering where you were, what you last merged, the list of availablke changesets, etc. You should be able to select any changesets you want and it should automate the rest]

- If you GLV (get latest version of) a solution, and some of the projects in it have been changed, VS repeatedly asks if you wish to reload each changed project. It is about 10x faster to close your solution, then GLV, then open the solution again than to GLV with it open. If I'm GLV'ing then of course I want to reload the projects! When I buy my food at the supermarket they don't ask me for every item "do you wish to take this item home with you?". [Edit: Still broken in VS2010][Fixed in 2012. Hurrah!]
- [edit] If two team members add a new project to a solution, then when the second person goes to check in, they must (obviously) resolve a merge conflict. However, TFS treats the .sln as a text file, and *corrupts* it (it adds the two project entries but the project count is effectively only incremented once). It would be so easy to fix the sln format to make the files mergeable.
- [edit] I don't do any source control operations from within the Solution Explorer window, as it has been rather unreliable ever since "integration" first came along. Even in 2008 it usually has random "checked out" icons on files that are not checked out, and recursive operations sometimes do weird things. Almost every source control 'glitch' we have is a result of someone starting an operation from the Solution Explorer. Luckily, I prefer to work in a Source Control window anyway.[2012: Sorry, can't tell you if this is fixed, as I haven't used this feature since 2008]

- [edit] Where to start with the Source Control Bindings window? VS could say *"Your Source Control settings have been corrupted again for no obvious reason. I never could get the hang of Thursdays. Shall I fix this for you? [YES]"*, but instead, it shows a complicated, confusing dialog full of information that makes no sense to anybody, resulting in a UI so scary that it makes junior programmers soil themselves. The trick is to ignore the whole window, hide behind your desk and click the "fix it" button, and it fixes it.
- [edit - added 12/2010] When you Get source code, especially when resolving merge conflicts, other windows are often brought to the front (either the Solution Explorer jumps in front of my Pending Changes view, which I have docked in the same tabbed area, or the Source Control window vanishes behind another document window. This is really annoying when you have another file to merge or another folder to Get, as you have to keep "finding" the Source Control/Pending Changes windows. Getting code should not constantly reorder my document/tool windows.[2012: Still broken]
- [edit - added 1/2014] With TFS 2012/2013, there is a choice of Server or Local workspaces. Server is the name for the old system where you must be online with the server to check files out. Local is the new default and makes a copy of the entire source repository on your computer, allowing you to make edits to any files without needing to check them out first. TFS then diffs your files against its local copy to

work out what you changed. This sounds good, and for many people it probably is good, but it has some serious drawbacks that you should be aware of:

- As you no longer check out files, they do not get locked when you edit them, and thus several people can edit any given file simultaneously, requiring a merge operation when they check in. This is fine for text-based source code files, but results in difficult situations or lost work when the files are unmergeable. Unmergeable or non-automatically mergeable files include Solution, Project, Resource (resx), XAML and any other XML files - so this causes a lot of problems in a development environment. If (like us) you also want to store Word and Excel documents and binary files under source control, local workspaces are positively dangerous. We have lost several days of work because someone unwittingly used a local workspace and then it was not practicable to merge their changes. You can reconfigure the TFS server to make Server workspaces the default to defend against this.
- With Local workspaces you have to keep *two* copies of everything on your computer. When we upgraded TFS we suddenly found everyone lost 25GB of disk space, and it took several weeks to work out where the disk space had gone! This was a major problem for us because we all use SSDs and it is only now (2014) that SSDs are getting large/cheap enough that we

can afford to be so inefficient with our disk space.

- In the few weeks that we used local workspaces we had several incidents where TFS corrupted files or lost changes, presumably due to bugs in the implementation. Quite simply, we cannot accept anything less than 100% reliability for our source control system.
- TFS is getting much easier to manage; these days if you don't want to customise anything too much you can set up a server in a very short time (hours) and setting up continuous integration builds and backups etc is extremely easy. On the flip side, while I found it very easy to set up backups of a TFS database, restoring that database and getting up and running after our server bricked itself was another matter - it took 4 days to work through all the unnecessary blocking problems (e.g. you have to restore the backup form a network drive, the data can't be local. When I tried to restore the image to the rebuilt server, TFS kept telling me there were no databases that could be restored. When I got past that, TFS wouldn't use the databases because they didn't match the host server (because that server was gone, the OS had been reinstalled). It took a lot of searching and fettling to get the backup to restore. Restoring should "just work"!

As you can see, most of the above are just trivial UI gripes. There is such a lot that could be improved about the UI. But the actual underlying product is good. I prefer TFS to pretty much every other SCM I've used over the last 28 years.

I wouldn't even mind the poor UI so much, except that it is one of the core UIs developers have to use on an hour-by hour basis, and they have to pay such a lot to get it. If the subscription money from a *single* developer was invested on improving the UI it would make a *massive* difference to the usability of TFS! It's painful to think that TFS is merely *good* or *ok* when it could so easily be *excellent* with a bit of nice UI.

Share Improve this answer Follow

edited Jan 19, 2015 at 22:20

answered May 8, 2010 at 20:24

Jason Williams
57.9k ● 11 ● 109 ● 140

2 Anyone care to explain why they downvoted? My answer isn't a rant, it's a factual list of the biggest problems in my daily use of TFS, (I'm just the messenger, not the team that hasn't fixed these very real problems). Having said that, my team still uses TFS despite these irritations, because it is a good SCM, if frustrating to use at times. If you think there are important pro/con points I've missed, then why not simply add an answer of your own? – Jason Williams May 9, 2010 at 21:05

3 Note that, while it's not discoverable, you can replicate workspace mappings pretty easily. Edit your workspace, click on the first mapping, shift+Click on the last mapping, press Ctrl+C to copy to clipboard. Now you can paste those mappings into a new workspace. – Jim Lamb May 10, 2010 at 13:47

@Jim: Cool, thanks. I'm sure I've tried that and failed, but maybe that was in 2005... – Jason Williams May 10, 2010 at 18:26

@Jim: Yes that works, but only for cross computer workspaces. If you want to have multiple workspaces on the same computer, you still have to edit every mapping to the new local directories. TFS 2010's public workspaces also help with the sharing across developer idea. – Jim Deville May 19, 2010 at 6:23

14

Hates

1. Doesn't track changes to files unless you've checked them out, so if you edit a file in Notepad++ TFS is unaware that anything changed.
2. It's very easy for someone to check out a fille and lock it so that nobody else can make changes. TFS shouldn't drop this ability, but it certainly should make it much harder to do than it is currently.
3. The methods to undo a commit or two is very unclear, so much so that I'm never quite sure if it worked or not.

4. The way that TFS makes files read only unless you check them out is obnoxious, though it does help me remember to check files out before I save the edits I've made.

Loves

1. I suppose built-in integration with visual studio is nice, if you like that kind of thing (I don't)

Share Improve this answer Follow

answered May 8, 2010 at 19:31

kubi
49.3k ● 19 ● 98 ● 119

---

2 Starting from number four (and related to number one) - it only makes sense to lock them and having to explicitly check it is a good part of the reason why. You have no implied "ownership" of a file until you go to work on it. If "fixing" locking is that large a panacea on your team, talk to you administrator about denying lock permission. As for "undo" I am glad that it isn't really built in apart from rolling back from the command line. "Undo" alters history. I would rather see a developer checkout the previous version and do a forced check-in explaining why they are doing something. – Joseph Ferris May 9, 2010 at 11:15

I agree with your point sbout checking out a previous version and doing a forced check-in. This is what I do, the problem is that I'm quite sure if it worked or not. I'd prefer if TFS had a git-like revert command. – kubi May 9, 2010 at 14:43

---

1 Adressing your first point, I don't see why locking and checkout is necessary or what it gets me. Why can't TFS tell that I've edited a file and automatically check it out for me

when I do? It does this when i edit files through visual studio, it should do it when I edit them outside the IDE. – kubi May 9, 2010 at 14:45

It is about integration, really. Visual Studio knows how to check out something, where something like Notepad++ does now. In order for that checkout to happen automatically, there would need to be some sort of background process running on the machine that would watch your workspaces and detect that changes have been made, notifying TFS. Not exactly sure of all the mechanics of how Visual Studio does it, but whatever it is would be available through the TFS web services (how VS talks to TFS). My guess would be that MS did not want to have a persistent process running in the background. – Joseph Ferris May 9, 2010 at 19:28

1 Hate #1 is a nightmare. I'm pissed off daily cause of it. – Arnis Lapsa May 27, 2010 at 11:10

9

I am a member of the Team Foundation Server team at Microsoft. There are a lot of very valid issues raised here. Some of them are addressed in the 2010 release. Others remain as issues, but we do recognize them and are working to improve the developer experience with the next release. Discussions like this are great for helping us make sure we're solving the right problems.

Here is some info on issues that are at least partially addressed today in the 2010 version:

Stand alone client

For non-developer customers that want to use the product outside of VS, they can use the Windows Shell extension powertool.

If you have users (developers or not) that need to access TFS from non-Window machines, they can use Team Explorer Everywhere. This is supported on platforms including Mac & Linux.

Copy workspace

There are two ways to copy a workspace today. The 1st is by using the workspace template command at the cmd line. Ex.

Tf /workspace /new /template[workspace name/owner to copy from]

Alternatively, you can open a workspace in the UI, select all of the mappings, copy them, & then paste them into a file/email. Someone else can than paste those same mappings into their workspace.

It would definetly be great if you could simply specify a default workspace that clients automatically pick up, but we don't have this today.

Merging robustness

The scenario described where you do an add, rename, add & then have problems when you merge has been addressed in TFS 2010.

Branch/Merge as a 1st class experience

In TFS 2010, branches are now 1st class objects in TFS. You can visualize your branches & even track changes as they move through the branch. Branching is also now a fast server based operation.

Get Latest Version of multiple projects

You can do this today by choosing the TFS instance node in source control explorer & then selecting get latest. This is the equivalent of the root folder ($).

File locking

By default TFS never locks files when users checks them out. This is the way we use TFS at Microsoft & how we see the majority of our customers using TFS. It is possible to enable users to explicitly lock files. Some customers find this desirable, but it is not the default path experience.

Share Improve this answer Follow

answered May 11, 2010 at 18:00

community wiki
Jamie Cool

"By default TFS never locks files when users checks them out": this is totally dependent on the type of the file. RPT files (CR) are always locked when checked out. – jcollum May 24, 2010 at 18:08

Thanks @Jamie, it's always nice to hear from the developer who is actively trying to support us, instead of just hearing us

whinging about the things we dislike :-) TFS/VSTS is ever improving, I just wish MS would put a bit more resource into it and help you guys to make it better faster! The new features in 2010 sound great (especially the branching) - I'm just waiting for my new server to arrive so I can get our TFS upgraded so we can take advantage of it all.
– Jason Williams May 24, 2010 at 20:54

6

Con: Checkout model. Many applications do not deal well with files that are marked as read-only then change to writable (Word 2007, Notepad). So you open a file, edit the file, try to save then you're told that you can't save because it's read-only. Great, now you have to Save As..., delete the original and renamed the new one to the old name. If there's an upside to having local files be read-only I don't see it. I really prefer Subversion's approach to this.

The one upside to making files read-only is that it reminds you to check them out. However that's really just a symptom of the check-out model.

Share Improve this answer Follow

edited May 11, 2010 at 20:42

community wiki
2 revs
jcollum

In my opinion that's the fault of those applications that fail to deal with this correctly. – antred Sep 23, 2014 at 19:46

3

I think that TFS is the single best ALM product on the market today. Looking at it from only a source control platform is slanted. I have used many products in my career to date: VSS, SVN, Git, StarTeam, CC/Harvest, and ClearCase - apart from TFS. Personally, I cringe at the thought of going back to anything other than TFS.

TFS is an extremely powerful platform. My biggest problem with it is often related to people not knowing how to use it or using it incorrectly. It is not meant to be an application that "just works". Sure, you can use it for basic source control without learning much about it - but if that is all you use it for, then you really are better off using one of the less robust tools out there. In reality, what TFS does not give you is the way to interpret features how you want to. It is specifically built from the ground up to support process and not just be a repository.

Share Improve this answer Follow

answered May 9, 2010 at 11:32
Joseph Ferris
12.7k ● 3 ● 48 ● 73

@joseph: I agree with you about the process. Unfortunately, if you have to use the process, you need a user interface. This is where TFS falls over. It's frustrating because the technical foundation is so strong, but the user interface is so archaic, clunky, and unnecessarily difficult. It would take so little to fix that, it's really frustrating to see how little it has advanced from 2005 to 2008 to 2010. (To put this into perspective, poor UI is the failing point of most SCMs. TFS is still quite usable, but it could be *awesome* instead of pedestrian) – Jason Williams May 10, 2010 at 19:44

I am with you on the the 2005 to 2008 part, but I keep finding great new things in 2010 every day. Just today, I couldn't check out a shelveset in 2008, so I fire up 2010 and it detects a conflict in the workspace that 2008 could not deal with. Branching and merging is a dream now, too. And Team Build has just blown me away. I think they know there is a lot to do, and it comes down to where to focus the effort. They really did hit some of the bigger sore points in the last version, just not all of them. It will all be better in 2012 - assuming the Mayans are wrong. ;-) – Joseph Ferris May 10, 2010 at 22:54

1 Considering its price, it should be a best-in-class source control as well as an ALM. It simply is not. For me it's not about the UI. It's about the decisions that the TFS-source-control team made that *every other SCM* team made differently. TFS is an oddball in the source control world. – jcollum May 10, 2010 at 23:46

Also, my question specifically says "TFS as source control". You chose to say "yeah but it's also an ALM!!". -1 – jcollum May 10, 2010 at 23:47

@joseph: We're using VSTS2010, but we're unlikely to be able to upgrade our TFS server for a couple of months, so we're stuck with TFS 2008. 2010 does look much better for branching. – Jason Williams May 11, 2010 at 5:39

3

Con: Timestamps. There's no way to set TFS to use the remote last-modified timestamp as the local last-modified timestamp. The local file's timestamp only tells me when I got the file. If I get a file that's 2 years old, there's no way to know that based on the local timestamp.

Other source controls that I have used have this ability.

Share Improve this answer Follow

answered May 14, 2010 at 19:39

community wiki

jcollum

I don't believe this is a desireable feature in the context of Microsoft's toolset. The MS C++ build tools use file dates to scan for changes in dependencies, resulting in re-compilation. If you modified a file, compiled, reverted the modified file to source control (with your file date feature), and re-compiled, you would still have a binary with the modified version of the file because the source file would be older than the binary as measured by file timestamps.
– David Gladfelter May 26, 2010 at 17:48

OK, not desirable for C++ then. Desirable for everyone else? Maybe. I definitely want it. At least make it a setting.
– jcollum May 27, 2010 at 20:36

3

Cons:

- workspace version: You can't identify the version of a workspace without doing a recursive search.
- terrible offline experience. attrib -r + tfpt online shouldn't be the way to work offline. Give me something like git that allows me to track status, undo and make changes. I'm even fine if it only stores the difference between the workspace version and current.
- Merging robustness: a changed file on the server + a local edit on different lines is not a conflict. a writeable file should not be an automatic conflict. The automerge button should NOT exist, because it should never be a scenario.

- Workspaces: the idea of being able to rearrange the source structure is just odd, and causes issues. the requirement of having both branches mapped in order to merge is odd. The requirement of having to do an operation multiple times, because my workspace mapping doesn't have a true root folder is wrong.
- Full reliance on remote server: There are some nice things about having all these things stored on the server, but really, you could store information locally and then upload it when needed. Keep pending changes, workspace mappings, basic undo history locally, etc.

Pros

- Shelvesets: I love these, and wish support for them was brought to the local disk as well (think git stash)
- Source control view in VS: It's pretty cool to be able to view the entire repository without downloading it. There are some usability issues, but the overall idea is cool.
- Workspaces: yep, both places. While re-arranging a repo is odd, the ability to only download what you need is pretty awesome. I often wish I could choose a root folder and then check box the paths I need, but oh well.

Share Improve this answer Follow

answered May 19, 2010 at 6:43

community wiki
Jim Deville

2

The lack of rollback has been my biggest pain point.

Share Improve this answer Follow

answered Mar 3, 2012 at 3:47

community wiki
user370770

2

Dislikes:

- Using the history to figure out what has been done is cumbersome to say the least. You have to click on every single history entry to see what files were changed, and then you need to go through a context menu to get a diff.
- Working while disconnected from the network is a big no-no. Ever heard of working on an airplane?
- No Windows Explorer integration for when you work with files outside of VS (think TortoiseSVN).
- Process methodologists (configuration managers) love to not allow shared check-outs. This is absolutely horrible for example for config files that you need to modify for testing.

- SC gets confused with complex move/delete operations.
- SC does not recognize when a checked out file has not changed. For example, service reference updates check out all related files and often regenerate the exact same content. These files should implicitly be removed from check-ins because they just add noise when you look at your changeset later.

Likes:

- Shelving.

Anybody guessed which is my favorite SCM system? SVN + TortoiseSVN + VisualSVN :-)

Share Improve this answer Follow

edited Jun 1, 2012 at 15:36

community wiki
2 revs, 2 users 73%
Christoph

Yes, no ability to get a diff report is lame. Most other SCMs have that. – jcollum May 27, 2010 at 20:37

You can bulk compare by right clicking on a folder and choosing compare. I believe tf folderdiff does the same. – j.i.h. Apr 28, 2015 at 19:32

1

Search functionality is not implemented in TFS 2010 ? VSS we have search in file; TFS 2008 we have search file ...

Share Improve this answer Follow

answered May 27, 2010 at 10:55

community wiki
muralidharan TR

1

Con: If you want to move multiple files to a subfolder of the existing location, you have to do that **one at a time**. Wow, that's horrible.

Share Improve this answer Follow

answered May 27, 2010 at 19:44

community wiki
jcollum

Oh, unless you want to go to the command line. That happens a lot in TFS. Want to do something slightly unusual? Command line. – jcollum May 27, 2010 at 20:35

0

The lack of true rollback support and the inability to rename a TFS Project are my two main pet peeves with TFS. Other than that, I've been very happy with it for 2-3 years.

The fact that certain applications do not support in-edit changes from read-only to writable (forcing you to reopen the file in question) is annoying but is really a problem with those specific applications. The fact that a file is read-only while not checked out has certain uses, one of which being that it reminds you to check out the file. It does occasionally, however, lead to confusion when trying to get specific revisions of files. Writable files are not re-downloaded unless you enable a flag, because they're considered local edits.

answered May 11, 2010 at 18:04

community wiki
user315772