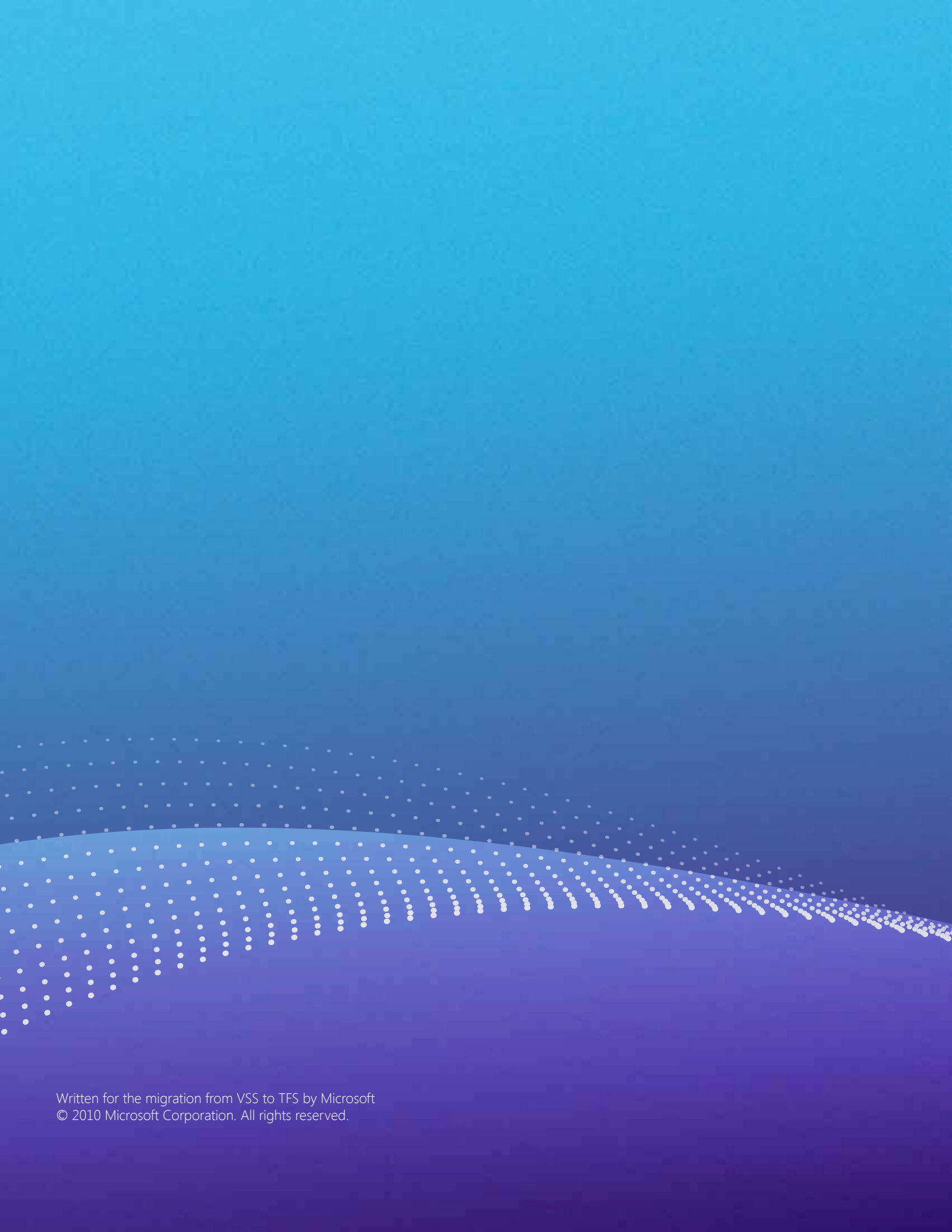


# Upgrading from Visual SourceSafe to Team Foundation Server

The benefits of moving to a modern platform

Written by the creator of Visual SourceSafe and Team Foundation Server  
**Brian Harry**



# Why should you consider moving?

When Visual SourceSafe (VSS) was first on the market, it was a state-of-the-art version control tool which met the needs of developers. It was a time when the Internet barely existed, networks were not commonplace and applications did not share data. Development teams were located all in one place and they were almost always small teams. In other words, the world was a simpler place. The last fifteen years have seen an explosive growth of software development, distributed solutions and the Internet. With that growth, there has been an increase in complexity. Many software development teams need to be able to access code remotely and to more easily track and audit changes to code and have a high performance and very reliable source code repository.

The reality of VSS is that it fails to meet many of these modern requirements and has clearly reached the end of its evolution. No new VSS versions, or updates to current versions, are planned by Microsoft. Team Foundation Server (TFS) is the replacement for VSS and with so many benefits to be gained by moving to Team Foundation Server, there is no better time to move to Team Foundation Server, than today!

# What are the benefits of Team Foundation Server over Visual SourceSafe?

## Performance, Reliability and Scalability

Backed by SQL Server 2008, Team Foundation Server benefits from the performance and scalability of an industry leading, enterprise class database server. For small teams that don't need the power of a full-fledged SQL Server installation, installing on a client OS like Windows7 and using SQL Server Express provides the same powerful engine in a lightweight package. This provides one of the key benefits that VSS never had – you can start small and grow over time. Upgrading from SQL Server Express to the full version of SQL Server is as simple as a backup and restore. Other benefits of SQL Server include the ability to dynamically expand storage as needed, faster read/writes, and clustering as part of a fault tolerant solution.

The tight integration of Team Foundation Server and Visual Studio provide users with a familiar and effective source controlled development environment. Typically Visual SourceSafe developers had to open the VSS user interface outside of Visual Studio to accomplish source control tasks. Because of the Team Foundation Server integration with Visual Studio, virtually every option regarding source code and work items can be accomplished directly through Visual Studio (some administrative features need to be accomplished through the full-featured command line).

One issue that tends to effect VSS installations is the 4GB recommended limit. Teams which are close to this limit or have exceeded it tend to find corruption issues in their source code requiring repair. Restoring this data from a backup is frequently difficult because teams have to rely on system administrators to restore the file share. Here again, Team Foundation Server takes advantage of the power of SQL Server. SQL Server has built-in protection against data corruption but, if that occurs, it can be restored through a SQL Server backup. SQL Server allows for many different data protection schemes from incremental and full back ups to log shipping and mirroring which all provide reliable mechanisms for protecting your data.

## Security

In VSS, security is limited to three settings – Admin, Read/Write and Read. Applying security settings at the file level was possible, but the experience was far from easy. The major drawback to this security was that it was an all or nothing type of security – if you could write to the file you could do anything you wanted with it. And because everything was stored on a file share, anyone with access to the file share could simply delete it all.

While granular security of your source code may not be critical, Team Foundation Server does provide it. Along with being able to control security at the file level, teams can also support specific scenarios such as letting a developer check out a file but not check it in. This is handy in many situations such as peer reviews and outsourced development. And, since it is stored in SQL Server, losing all of your source code because of a delete will never happen again.

## Advanced Features

As a developer, there are times when you want to know which developer modified a specific line of code in a file. You not only want to know when they made the change, you want to know why. With the Annotate feature of Team Foundation Server you get just this capability.



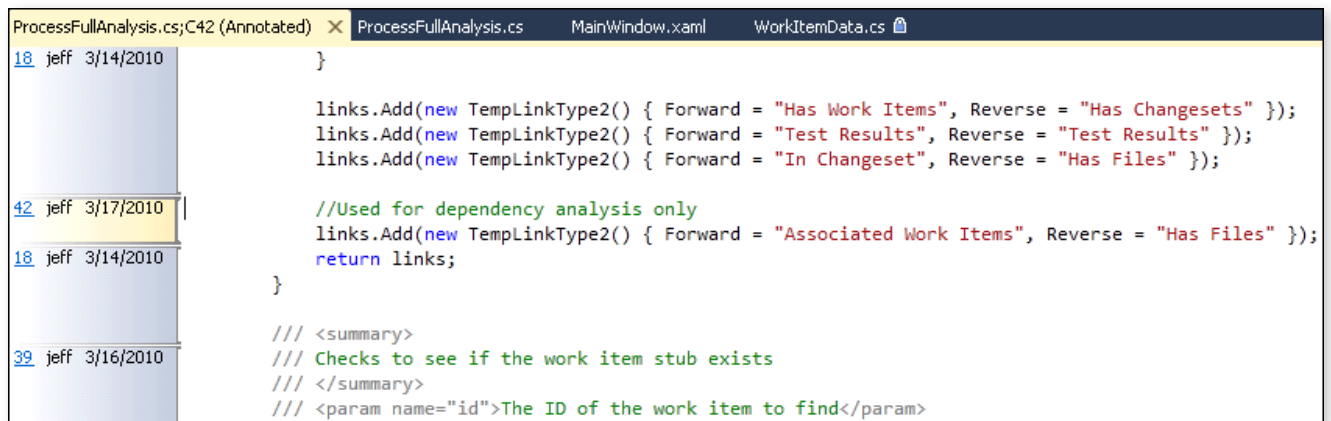


Figure 1 - Team Foundation Server Annotate

Figure 1 shows a file annotation where every single line in the file is identified in terms of who added it and when they added it. This view also allows you to drill into the details of why the change was made as shown in Figure 2. This view also highlights another benefit of Team Foundation Server – atomic check-ins.

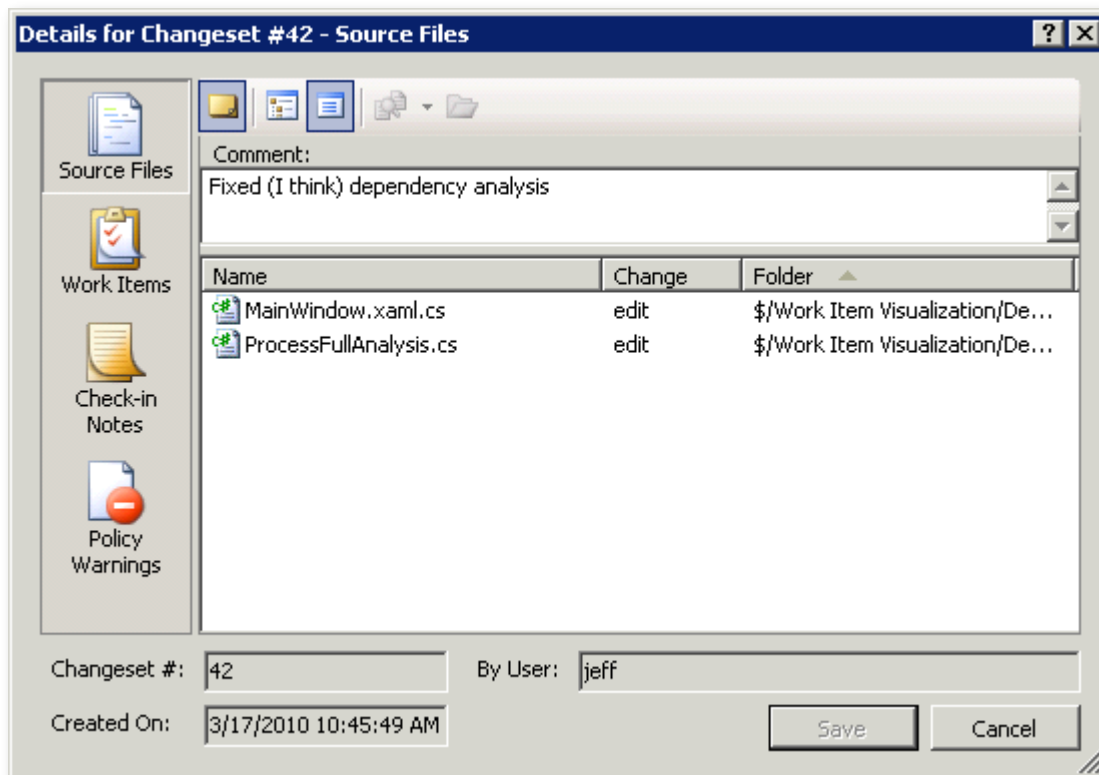


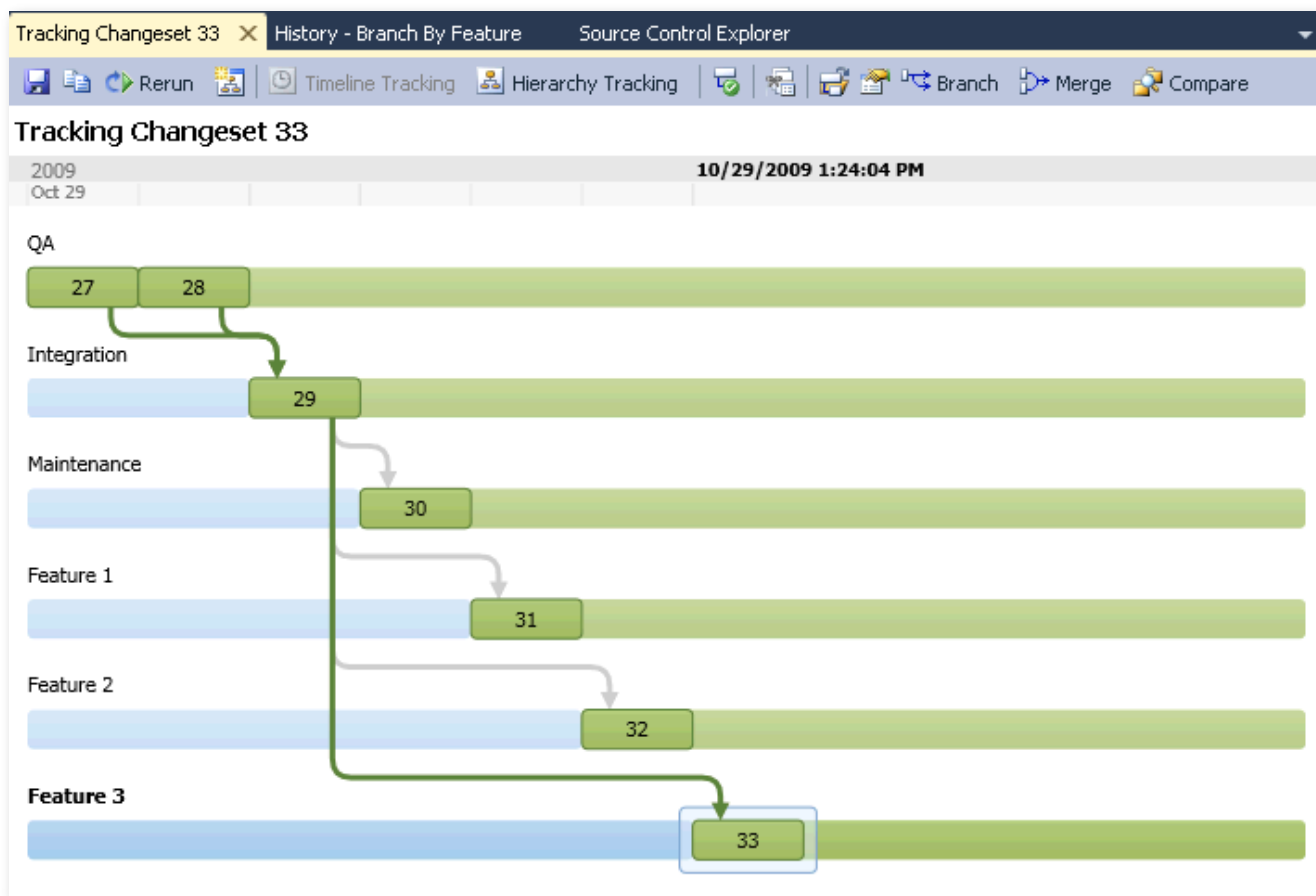
Figure 2 - Changeset Details Dialog

One issue with VSS is that it does not support atomic check-ins. Suppose you had to make a change that spanned three different files (a.cs, b.cs, c.cs) and you checked them into VSS but the check-in for one of the three files failed. The next time any developers performed a Get operation they would probably end up with a broken build due to the missing changes from the partially successful check-in. With atomic check-ins, all files are checked in or none of them are – there is no halfway state. This ensures that the checked-in code is always in a consistent state. In addition to this, in VSS it is virtually impossible to figure out which files were checked in together without examining every timestamp on every file. In Team Foundation Server, the concept of atomic changesets is a fundamental concept. Figure 2 shows the Changeset Details dialog of Team Foundation Server which clearly

shows all of the files that were checked in at a given time. This capability allows for rolling back entire check-ins rather than rolling back files one at a time. Developers can also clearly see what changes were made to each file during the check in, comments and associated work items.

Branching is problematic in VSS for several reasons. One of the biggest is because there is no way to visualize the branching structure or the movement of code across branches. In addition, because VSS used Sharing instead of true Branching there could be conflicts based on where the code was shared to and when changes were made to it. The chief reason for this is that sharing did just that – it provided a reference to the actual file, not a copy. So if one project changed the file it could break other projects accidentally. In later versions of VSS the concept of branching was added on top of sharing such that immediately after a Share operation was performed you could Branch the code. This essentially deleted the reference and created a copy of the file. Merging could then be done as expected.

Team Foundation Server uses a pure branching model – there is no concept of sharing. This promotes better software development practices and eliminates the possibility that when a change is made in one place it will automatically break code in another project. Team Foundation Server 2010 introduces branch visualization so that teams can actually see where the code moved to and the basic structure of their branches. Furthermore, the visualizations are actionable, allowing users to merge directly from the visualization ([Figure 3](#)).



**Figure 3** - Branch Visualization

This relates to another extensibility point – the built-in DiffMerge tool can be replaced with any tool you want and can be made specific to a file extension. For example, you could use Scooter Software's Beyond Compare to diff code files but Altova DiffDog to diff XML files. As more advanced diff and merge tools become available you can take advantage of them.

## Bug and Task Tracking with Team Foundation Server Work Items

Work Items are another key benefit of Team Foundation Server. While not strictly necessary, Work Items provide the ability to trace work back to specific requirements or to assign work to developers and report status to project managers. This not only improves developer efficiency but provides a built-in bug tracking tool. This is a feature that teams can grow into if they desire to use it.

## E-mail Alerts

Team Foundation Server allows developers to subscribe to e-mail alerts when files or folders that are important to them change. Developers can right-click a file or folder and select Alert on Change and they will receive an e-mail alert with the change notification. With Team Foundation Server you don't have to check history to see if someone has changed a file that is important to what you are working on – you'll be notified of it.

## Shelvesets

Shelvesets are almost exactly what they sound like. They allow a developer to put a set of changes on a shelf – temporary storage in Team Foundation Server where they are backed up along with the rest of the files in source control. The difference here is that the files are not checked in to the repository. This allows developers to store their unfinished work on the server at the end of the day, or to share problem code with other developers without breaking anyone else's code.

## Automated Build

Many teams use third party products to pull their source code from VSS and build it periodically. For teams doing agile development, as many small teams are, add-ons may be used to build their code nightly. The problem with this solution is that it requires yet another set of tools and integration (i.e. compatibility) issues to deal with. Team Foundation Server includes a full automated build engine that supports multiple configurations and is scalable. It also includes advanced features such as Windows WorkFlow 4.0 and is extensible in every aspect.

# How VSS and Team Foundation Server Usage Compare

You have seen some of the benefits of upgrading to Team Foundation Server but it's helpful to know how the tools compare in actual usage. [Figure 4](#) shows the Visual SourceSafe interface and [Figure 5](#) shows the Team Foundation Server interface.

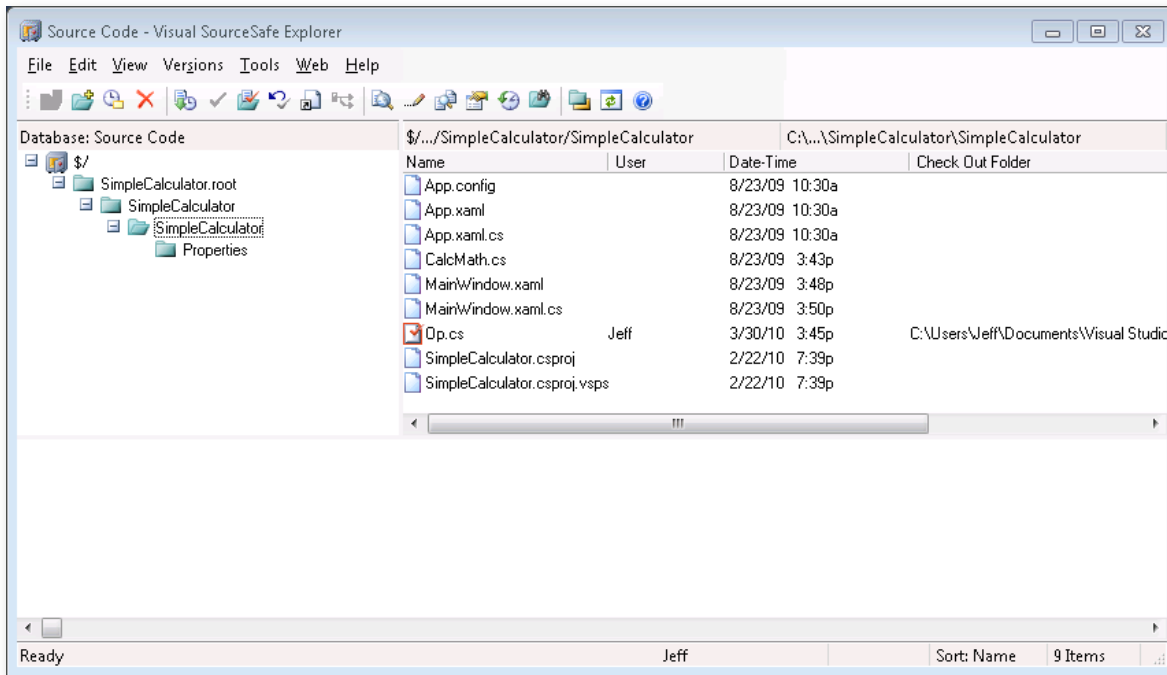


Figure 4 - Visual SourceSafe

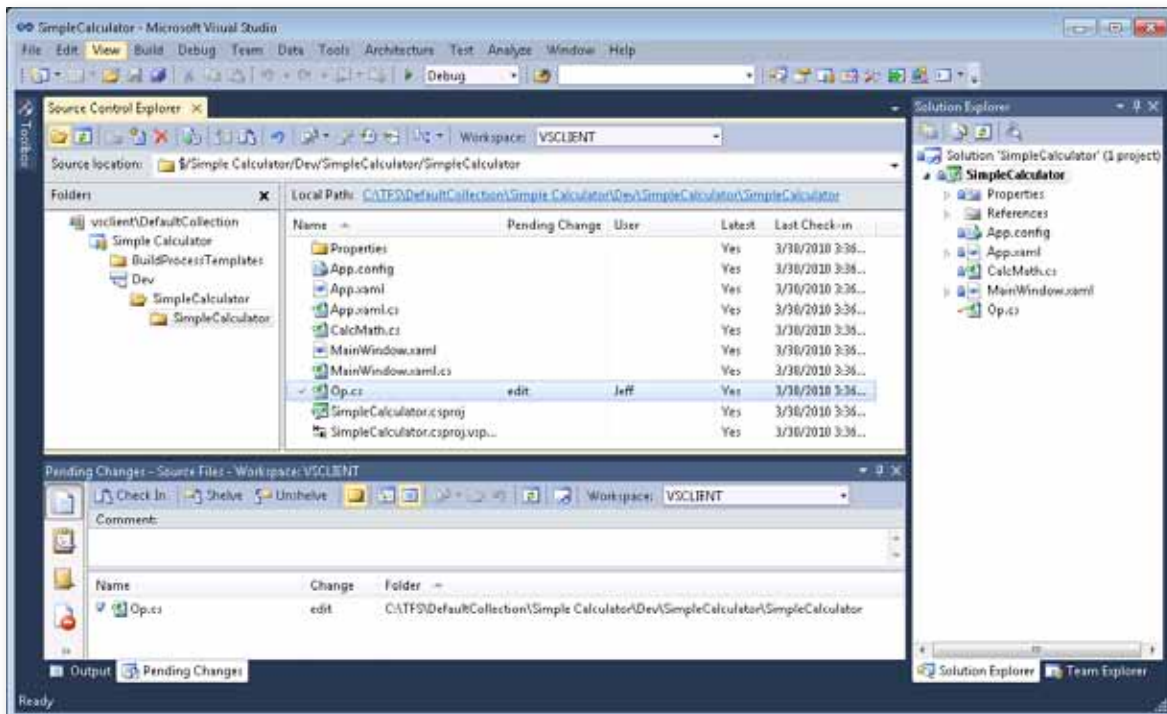


Figure 5 - Team Foundation Server



As you can see, they have a very similar look and feel. The organization and structure of VSS was well-thought-out and that has translated into the look and feel of Team Foundation Server. This makes working with Team Foundation Server very natural for users upgrading from VSS. Another key point to take away from [Figures 4 and 5](#) is that Team Foundation Server is fully integrated into Visual Studio. There is no separate Team Foundation Server user interface for operations. This increases developer efficiency and allows for most source control operations to be performed in a single environment. [Table 1](#) shows many of the common operations in VSS and how they are performed in Team Foundation Server.

**Table 1 – Comparison of some common VSS and Team Foundation Server Operations**

Operation	VSS	Team Foundation Server
<b>Check Out</b>	Right-click a file (in VSS or the IDE) and select Check Out. Alternatively, you can start typing in a file and a Get Latest operation is performed and the file is checked out for editing.)	Identical. In Team Foundation Server whether or not a Get Latest operation on check out is performed is optional.
<b>Check In</b>	Right-click a file (in VSS or the IDE) and select Check In or use the Pending Checkins window. (Figure 6)	Identical. In Team Foundation Server the Pending Checkins window is called Pending Changes. (Figure 7)
<b>View History</b>	Right-click a file (in VSS or the IDE) and select View History.	Identical except that Team Foundation Server does not display a modal dialog box.
<b>Get</b>	Right-click a file (in VSS or the IDE) and select Get. To get a specific version you must view the file history, select the specific version and click Get.	Right-click a file and select Get Latest. To get a specific version, right-click a file and select Get Specific Version. Choose the filter (by history, date, label, latest or workspace version), query for the file and click Get.
<b>Branch</b>	Right-click the target folder in VSS and select Share to. Select the folder to share and click Share. Click Recursive and OK.	In Source Control Explorer, right-click a folder and select Branching, then select Merging > Branch. Click OK.
<b>Wildcard Search</b>	Open VSS and select View > Search (Wildcard or Status) and select the appropriate options.	In Source Control Explorer, right-click a branch/folder/node and select Find in Source Control (Wildcard or Status) and select the appropriate options.
<b>Compare</b>	Right-click a file or folder and select Show Differences.	Identical. In Team Foundation Server the command/menu item is Compare.

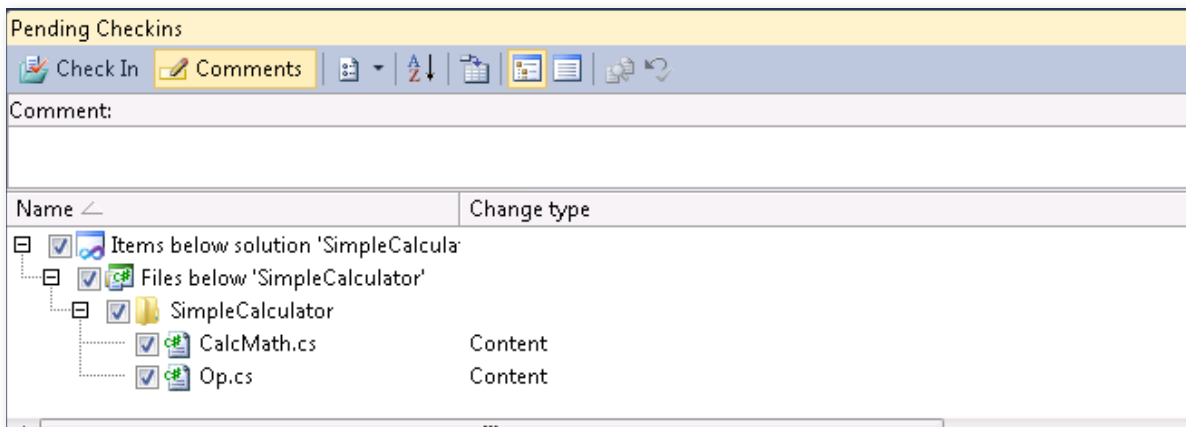


Figure 6 - Visual SourceSafe Pending Checkins pane

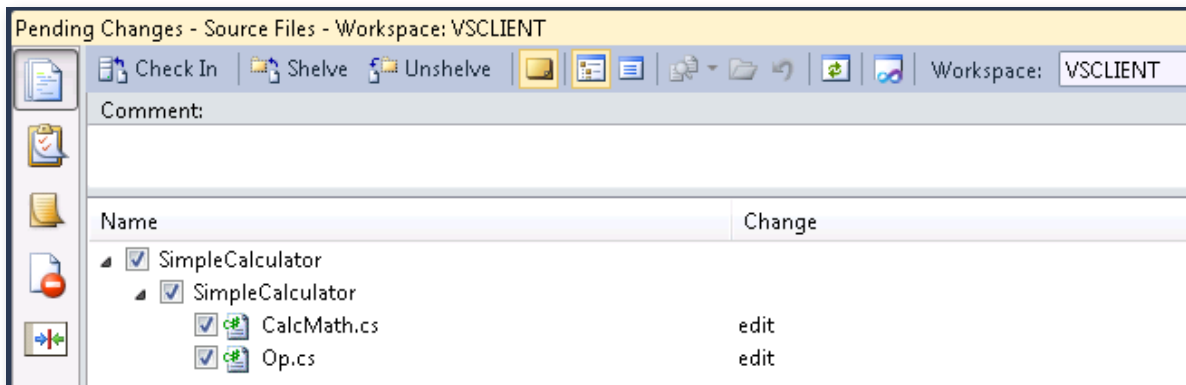


Figure 7 - Team Foundation Server Pending Changes pane

When looking at [Figures 6](#) and [7](#) the most obvious difference is that there are more options available (additional panes are available down the left-hand side of the Team Foundation Server Pending Changes window). Aside from the additional options, the usage is virtually identical.

[Figures 8](#) and [9](#) compare the History windows for both VSS and Team Foundation Server.

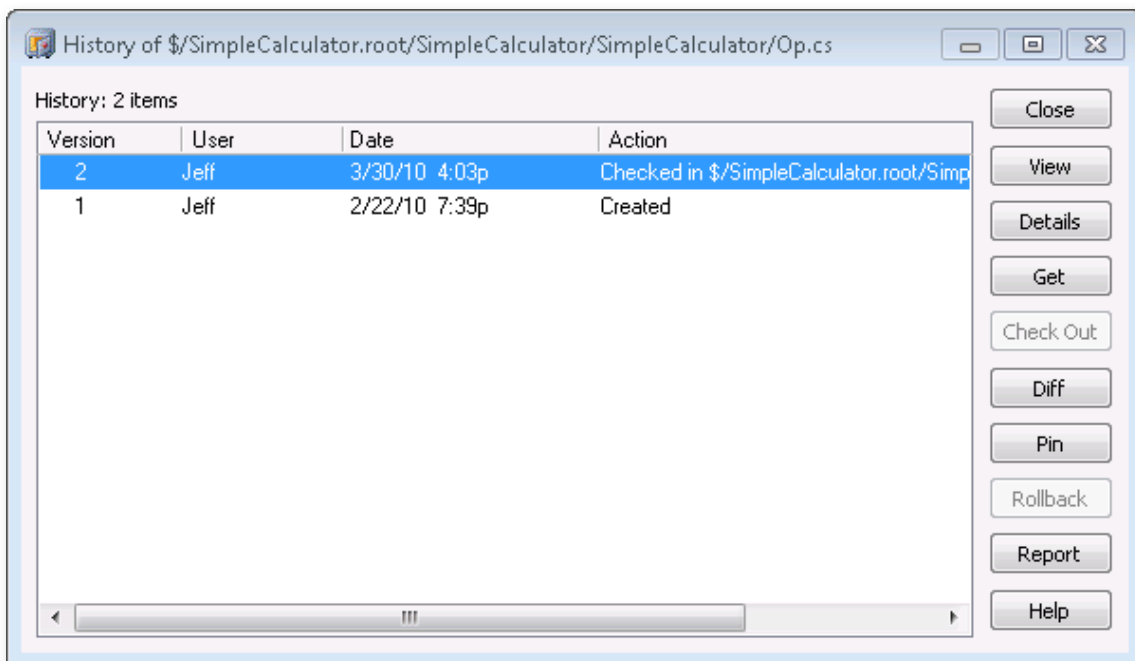


Figure 8 - Visual SourceSafe History dialog

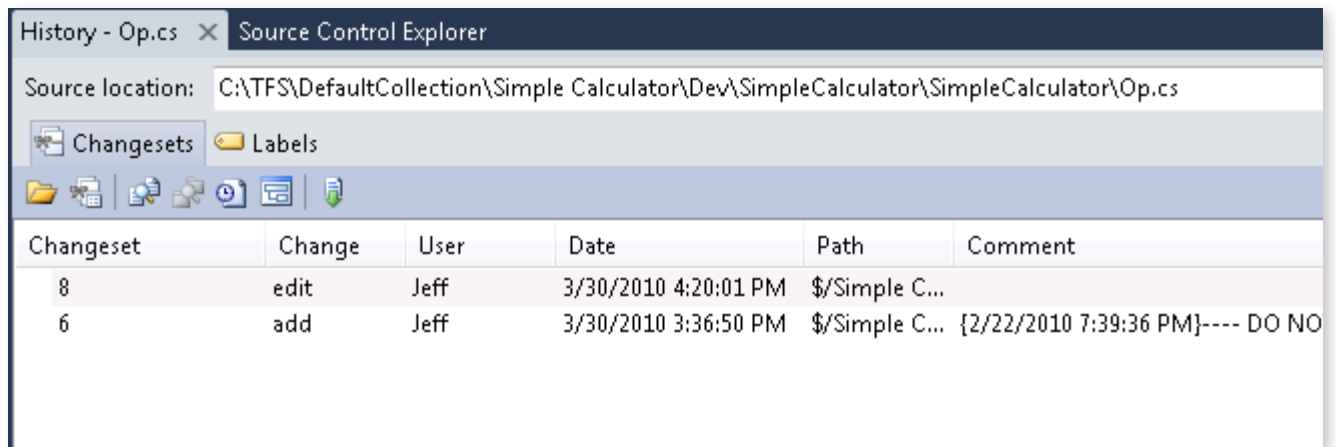


Figure 9 - Team Foundation Server History window

In this comparison, several options are immediately noticeable – there is a Version column in the VSS dialog and a Changeset column in the Team Foundation Server dialog. This is part of the concept discussed above in the Advanced Features section. Changeset numbers are unique across the server whereas VSS Version numbers are only unique when related to a specific file. This means that in Team Foundation Server you can also – without checking a timestamp – determine which set of changes were checked in before or after any other set of changes. In addition, you can select the Labels tab directly from this window to see which labels a file participated in.

# What happened to my favorite features?

Some of the features available in VSS are either not supported in Team Foundation Server or have different implementations. [Table 2](#) contains a list of those features. Other features requiring more explanation, such as Sharing and Pinning, are discussed below.

**Table 2 – Team Foundation Server versions of VSS features**

VSS Feature	Description	What happened to it in Team Foundation Server
<b>Keyword Expansion</b>	Insert placeholders in the header that are replaced on check-in (such as \$Revision: \$ was replaced with \$Revision: 23 \$ so that you know this version of the file is version 23.)	This functionality is not currently supported in Team Foundation Server.
<b>Use Read-Only</b>	Allows files to remain in a read/write or read-only state when retrieved to the local hard drive.	This functionality is not currently supported in Team Foundation Server.
<b>Rollback</b>	The ability to set the latest version of a file to a previous version.	Team Foundation Server does support this functionality but it is command line only in Team Foundation Server 2010 (TF Rollback).
<b>Shadow Folders</b>	The ability to sync the contents of VSS with a file share so that individuals who cannot access VSS can access source code files.	Team Foundation Server has no built-in functionality for this but it can be easily duplicated by using a scheduled task and running TF Get to copy the files to a file share.
<b>Restore Mod</b>	This feature allows developers to set the date/time stamp of a file on their local machine to the time of last check-in, last check-out or when a developer retrieved the file from VSS.	This feature is not supported in Team Foundation Server. The date/time stamp of the file is always the point in time that the developer performed a Get on the file.
<b>Find in Files</b>  <b>View Histor</b>	Search the contents of files for specific words.	Searching files in Team Foundation Server is not supported at this time because of how the information in Team Foundation Server is stored. Currently to search files you must perform a Get to the local hard drive and search the files locally using Windows Search or another utility of your choice.



## Sharing

Sharing is not supported in Team Foundation Server. Team Foundation Server uses branches for a number of reasons. Sharing is a popular solution in VSS because it does not require merges to be performed and keeps files in sync automatically. Sharing, as opposed to branching, creates a reference to the file that is shared. This makes it appear that the file has been copied to a second location when in reality it is the same file. This has serious consequences for development teams working on different features.

The major issue with this solution is that when one team changes the file it also changes for the other team. This means that code can change out from underneath developers and break other changes that they have made.

Any Shared files or folders that you have when you convert to Team Foundation Server will be converted as separate files: the file is actually copied instead of using a reference).

## Pinning

As a concept, Pinning allows you to say, "This file is the last known good file and if you are going to get the latest version of the file you should get this one." This lets developers experiment with later versions by Pinning a good version that everyone would get. If they made any mistakes, other developers were shielded from them. Another use of pinning involved specifying which versions of files were to be incorporated into the final release of the software.

In Team Foundation Server, the best practice is to make use of private branches which let developers perform the same functionality. The benefit of using private branches is that many developers can be working on their own version of the file and those versions can be merged together when ready.

# How do I migrate from VSS to Team Foundation Server?

There are two options for moving from VSS to Team Foundation Server: Point-in-time and full migration. Regardless of which option you choose, it is recommended that you keep a backup copy of your Visual SourceSafe database. This is more important when performing a point-in-time migration so that you maintain the full history of all of your versioned files.

## Point-in-Time

The point-in-time option is basically a Get Latest, unbind from VSS and add to Team Foundation Server. This is the simplest and quickest option but it has one drawback – no history is migrated. In many cases, this is not a big deal because you can just take a copy of VSS and archive it in case you need to get to that history. To perform a point-in-time migration, take the following steps:

1. Open Visual Studio and ensure that Visual SourceSafe is your version control provider (Tools > Options > Source Control).
2. Open the project you want to convert and perform a Get Latest operation.
3. Select File > Source Control > Change Source Control.
4. Unbind the Solution and all Projects and click OK.
5. Close the solution and click Yes when prompted to save the changes.
6. Set Team Foundation Server as the version control provider (Tools > Options > Source Control).
7. Open Source Control Explorer (View > Other Windows > Source Control Explorer).
8. Make sure you have a Team Project created, if not create one first.
9. Select the project where you want the files to go and map that location to your local hard drive.
10. Copy the solution from the existing location to the location you just mapped in Step 9.
11. Open the copied solution in Visual Studio.
12. Right-click the solution and select Add Solution to Source Control.
13. Check in the pending changes.

At this point the entire solution and all projects are now in Team Foundation Server.

## Full Migration

A full migration requires several steps which must be taken in order. A complete guide is included here and additional links are included in the reference section at the end of this paper.

1. Make sure VSS is installed on a machine with Team Explorer.
2. Make sure VSS is version 2005.

Make a copy of your VSS repository and copy it to this machine – don't work off of the production version in case something goes wrong (the likelihood is low but it is always best to be safe your work product). Make sure all of the developers have their code checked in before you make the copy.

3. Install KB947647.

<http://code.msdn.microsoft.com/KB947647/Release/ProjectReleases.aspx?ReleaseId=1028>.

This patches a known issue when converting from VSS to Team Foundation Server with the VSSConverter.

4. Know the administrator password for the VSS installation.
5. Create the Analysis settings files (below is a sample file) (call it analysissettings.xml for the purposes of these instructions) in a location that is part of the path or in the %programfiles(x86)%\Microsoft Visual Studio 10.0\Common7\IDE.

```
<?xml version="1.0" encoding="utf-8"?>
<SourceControlConverter>
  <ConverterSpecificSetting>
    <Source name="VSS">
      <VSSDatabase name="c:\VSSDatabase"1></VSSDatabase>
      <UserMap name="c:\Migrate\Usermap.xml"2></UserMap>
      <SQL Server="SQL Server Name"3></SQL>
    </Source>
    <ProjectMap>
      <Project Source="$ /FolderA"4></Project>
      <Project Source="$ /FolderB"></Project>
    </ProjectMap>
  </ConverterSpecificSetting>
  <Settings>
    <Output file="Analysis.xml"5></Output>
  </Settings>
</SourceControlConverter>
```

6. Open the Visual Studio 2010 Command prompt and enter the following command:  
You can do this at a regular command prompt to as long as you run as an administrator. The VSSConverter.exe application is located at %programfiles(x86)%\Microsoft Visual Studio 10.0\Common7\IDE. If you created the analysissettings.xml file in another location, you will need to provide the full path to the file.  
Vssconverter analyze analysissettings.xml
7. After the analysis has run, verify that there are no errors and examine the warnings to determine the severity. Most of the warnings are just informational messages. If there are errors, fix them and re-run the command in Step 6 until there are no errors.
8. Set up the project structure in Team Foundation Server (i.e. create the team projects that you want to migrate your code to from VSS.
9. Create the migration settings file (below is a sample file).  
Call it migrationsettings.xml for this walkthrough.<sup>6</sup>

---

<sup>1</sup> The path to where the VSS database resides – this is the folder that contains the srcusers.ini file.

<sup>2</sup> The Usermap.xml file will be created for you – this is the path where you want it created (including the file name) so if you want to put everything in c:\migration, make sure c:\migration exists first – it won't create the directories.

<sup>3</sup> Use SQL Server if your VSS database is larger than 4GB in size (by default VSS Converter uses SQL Express which can't store more data than 4GB).

<sup>4</sup> These are the source folders in VSS. \$/ represents the root of VSS and you can use this to analyze an entire VSS repository at once.

<sup>5</sup> This is the analysis file that is created after you run vssconverter. By default it goes in the folder where vssconverter is executed from – you can specify a full path to it though if you want.

<sup>6</sup> If you used the SQL Server tag in the analysis settings file, use it here also.

```

<?xml version="1.0" encoding="utf-8"?>
<SourceControlConverter>
  <ConverterSpecificSetting>
    <Source name="VSS">
      <VSSDatabase name="c:\VSSDatabase"></VSSDatabase>
      <UserMap name="c:\Migrate\Usermap.xml"7></UserMap>
    </Source>
    <ProjectMap>
      <Project Source="$FolderA" Destination="$TeamProjectA"8>
      <Project Source="$FolderB" Destination="$TeamProjectB/ProjectB" />
    </ProjectMap>
  </ConverterSpecificSetting>
  <Settings>
    <TeamFoundationServer name="My_TFS_Server" port="8080" protocol="http"
      collection="tfs/DefaultCollection"></TeamFoundationServer>
    <Output file="Migration.xml"9></Output>
  </Settings>
</SourceControlConverter>

```

#### 10. Update the Usermap.xml.

Every user listed in this file has to be mapped to a valid AD name that has at least read permission in Team Foundation Server. It is critical that you add the users in to Team Foundation Server first. Note also that it doesn't have to be a one-to-one mapping. If an employee left the company and they don't have an AD entry you can simply map them to another employee. This information is used to say who checked in the file but the original username is captured in the comments for that check in.

#### 11. From the Visual Studio 2010 Command Prompt run the following:

**Vssconverter migrate migrationsettings.xml**

At this point the conversion will execute and should run cleanly. Remember to set Visual Studio to use Team Foundation Server as the source control provider. In Visual Studio, from the main menu, select:

**Tools > Options > Source Control and select Visual Studio Team Foundation Server.**

<sup>7</sup> In this context this is the path to the usermap.xml file created during the analysis phase.

<sup>8</sup> This is the actual map that tells TFS where to put the code. In step 6 above you must create the team projects that these will map to. Note also that you don't have to map it to the root. Say for example you have VSS folder "\$/MyProject" and you want to move it in TFS to "\$/MyProject/Root/Dev" you can.

<sup>9</sup> This file is created for you and you can specify the full path to it if you want.



# How do I get Team Foundation Server?

With the introduction of Visual Studio 2010, Team Foundation Server 2010 is included as an MSDN benefit with the following client tools – Visual Studio 2010 Professional, Visual Studio 2010 Premium, Visual Studio 2010 Ultimate and Visual Studio Test Professional 2010. Each MSDN subscription is entitled to one production deployment of Team Foundation Server. In addition, each MSDN subscription includes a Team Foundation Server Client Access License (CAL) so there is nothing to purchase and maintain for developers. Whether you purchase the Professional, Premium, Ultimate or Test Professional edition, you're covered.

If you don't have an MSDN Subscription, Team Foundation Server can be purchased through resellers or online from the Microsoft Store. This license includes an allowance for up to five (5) users without CALs. Additional users may be added with CALs.

# Team Foundation Server is the future – Now

As you have discovered, using Team Foundation Server is not very different from using VSS when used just for version control. With very few exceptions, all of the features of VSS are available in Team Foundation Server and Team Foundation Server provides many more features than are available in VSS. Out of the box, Team Foundation Server may not work exactly like VSS and you may have to tweak a few things but the bottom line is that Team Foundation Server will work the way you want. Benefits, such as using SQL Server for storage, ensure data integrity, scalability, reliability, low maintenance and high performance. Each of these makes life faster and easier for developers so they can spend more time developing and less time managing source code. Combined with highly extensible features of Team Foundation Server and Work Item Tracking, Team Foundation Server provides room to grow at a pace that you are comfortable with – you don't have to use anything you don't want to. Team Foundation Server is the replacement for VSS and it's time to move forward. If you are an MSDN Subscriber, you already have access to Team Foundation Server 2010. If you aren't, you can get started by downloading a trial version today at <http://www.microsoft.com/downloads>.

# References

**These are links to all the pages talking about migrations:**

**VSSConverter Command-Line Utility for Source Control Migration**

<http://msdn.microsoft.com/en-us/library/ms253090.aspx>

**Walkthrough: Migrating from Visual SourceSafe to Team Foundation**

<http://msdn.microsoft.com/en-us/library/ms181247.aspx>

**Analyze Command (VSSConverter)**

<http://msdn.microsoft.com/en-us/library/ms400803.aspx>

**Migrate Command (VSSConverter)**

<http://msdn.microsoft.com/en-us/library/ms400685.aspx>

**How to: Create a Settings File for Source Control Analysis and Migration:**

<http://msdn.microsoft.com/en-us/library/ms253161.aspx>

**Sample VSS Converter Settings File**

<http://msdn.microsoft.com/en-us/library/ms253123.aspx>

**How to: Edit the Source Control Migration User Mapping File**

<http://msdn.microsoft.com/en-us/library/ms253175.aspx>

