Join the discussion @ p2p.wrox.com

Wrox Programmer to Programmer™

Professional Team Foundation Server 2013

Steven St. Jean, Damian Brady, Ed Blankenship, Martin Woodward, Grant Holliday

Part I

Getting Started

- **CHAPTER 1:** Introducing Visual Studio Online and Team Foundation Server 2013
- **CHAPTER 2:** Planning a Deployment
- **CHAPTER 3:** Installation and Confi guration
- **CHAPTER 4:** Connecting to Team Foundation Server

Chapter 1
Introducing Visual Studio Online and Team Foundation Server 2013

What's in this chapter?

- Getting to know Team Foundation Server 2013
- Understanding what's new in Team Foundation Server 2013
- Acquiring Team Foundation Server 2013

This chapter introduces you to Microsoft Visual Studio Team Foundation Server 2013. Here you learn what it is for, the key concepts needed when using it, and how to acquire it.

For those users already familiar with Team Foundation Server, the discussion in this chapter highlights areas that are new or have changed substantially. However, because understanding the legacy of a technology is always helpful, this chapter also includes some of the history of the Team Foundation Server product, which will help explain how it became what it is today.

This chapter also discusses the improved release model, including the ability to have Microsoft manage hosting, frequent upgrades, and backups by leveraging Visual Studio Online (formerly Team Foundation Service). Later chapters go into more depth with an examination of the architecture of the Team Foundation Server product.

What is Team Foundation Server?

Developing software is difficult—a fact repeatedly proven by how many projects run overtime or over budget, or fail completely. An essential factor in the success of any software development team is how well the members of the team communicate with one another, as well as with the people who wanted the software developed in the first place.

Team Foundation Server provides the core collaboration functionality for your software development teams in a very tightly integrated product. The functionality provided by Team Foundation Server includes the following:

- Project management
- Work item tracking (WIT)
- Version control
- Test case management
- Build automation
- Reporting
- Release management
- Lab and environment management
- Feedback management
- Chat and team communication tools

Each of these topics is explored extensively in this book

Team Foundation Server is a separate server product designed specifically for software engineering teams with developers, testers, architects, project managers, business analysts, and anyone else contributing to software development releases and projects. Logically, Team Foundation Server is made up of the following two tiers,

which can be physically deployed across one or many machines:

- **Application tier**—The *application tier* primarily consists of a set of web services with which the client machines communicate by using a highly optimized, web service-based protocol. It also includes a rich web access site to interact with a server without having to install a client such as Visual Studio.
- **Data tier**—The *data tier* utilizes SQL Server to house the databases that contain the database logic for the Team Foundation Server application, the data for your Team Foundation Server instance, as well as the data for your Team Project Collection. The data stored in the data warehouse database and Analysis Services cube are used by Team Foundation Server's reporting functionality. All the data stored in Team Foundation Server is stored in the SQL Server databases, thus making the system easy to back up.

Team Foundation Server was designed with extensibility in mind. It can integrate with a comprehensive .NET Application Programming Interface (API). It also has a set of events that allow it to integrate with outside tools as first-class citizens. The same .NET programming model and event system are used by Microsoft to construct Team Foundation Server, as well as the client integrations into Visual Studio.

Team Foundation Server has plenty of competitors, including other enterprise Application Lifecycle Management (ALM) systems and purpose-specific products (such as source control systems). The main benefit of having all the different systems available in one product is that Team Foundation Server fully integrates the different systems. This allows for true innovation in the development

tools space, as you will notice with several of the new tools available in this latest release. Instead of worrying about integrating the separate systems yourself, you can take advantage of the work that Microsoft has done for you. Jason Zander, currently Corporate Vice President of development for Windows Azure, makes this particular point well in a blog post originally about Team Foundation Server 2010. You can find the blog post at http://aka.ms/IntegratedALMSolution.

When you compare enterprise ALM products currently on the market, you will discover that Team Foundation Server was designed to be easily customized and extended. Team Foundation Server ensures that developers using any development platform can participate and easily use Team Foundation Server, including Visual Studio, Eclipse-based development, Xcode, and many more.

What is Visual Studio Online?

Installing and configuring Team Foundation Server has traditionally meant a significant investment in time and infrastructure. In addition to the initial setup, maintenance of an on-premises Team Foundation Server instance required ongoing effort.

In October 2012, a hosted Team Foundation Server was released to the general public under the name *Team Foundation Service*. This hosted service meant that a team could make use of many of the features of Team Foundation Service without the significant investment in infrastructure and maintenance. Since its initial release, the product has been continuously extended and improved.

In November 2013, Team Foundation Service was rolled into a new product called *Visual Studio Online*, which incorporates a number of developer services, including

most of the features of an on-premises Team Foundation Server installation as well as Visual Studio, collaboration tools, load testing and build services, a diagnostic service called *Application Insights*, and even an online code editor.

Visual Studio Online is free for teams up to five users, and is also available on a per-user per-month subscription basis. A number of plans are available that include various features. These features include access to a hosted Team Foundation Server with unlimited Team Projects and basic project planning tools, and either the Visual Studio Express or Visual Studio Professional IDE. The plans also include a certain amount of cloud build and load testing time.

What's New in Team Foundation Server 2013?

If you have used legacy versions of Team Foundation Server, you may be curious about what is new in the latest release. As this book demonstrates, it is a big release with considerable new functionality and improvements across the board. While many of these features are explained throughout this book, if you have used a previous version of Team Foundation Server, the features described in the following sections will be new to you. Some of the client-side topics are covered in more detail in the companion book to this volume, *Professional Application Lifecycle Management with Visual Studio 2013 by Mickey Gousset, Martin Hinshelwood, Brian A. Randell, Brian Keller, and Martin Woodward* (Wiley, 2014).

Version Control

One major change with this release was the addition of an alternative, distributed source control option within Team Foundation Server. While this was seen as a surprising

move by many, it was really just addressing a common complaint many organizations had with source control in Team Foundation Server.

There are many powerful project management features provided by Team Foundation Server, but the primary reason for adopting a system like Team Foundation Server will always be source control. Developers have increasingly been moving to distributed version control products, such as Git or Mercurial, and the limitations of the centralized source control system provided by Team Foundation Server was a common reason cited by organizations choosing to use an alternative product. While support for disconnected workspaces was provided with the addition of Local Workspaces in the 2012 release of Team Foundation Server, team members were still limited to a single server-side repository for a workspace.

By adopting Git as a first-class version control alternative, Microsoft has added true distributed version control to the product. Developers can keep a full local repository, which allows them to work with multiple branches and commit locally before pushing their changes to the server.

We want to note two important points with respect to the distributed version control offering in Team Foundation Server. First, the Git implementation is a standard implementation of Git rather than one that has been specifically written for Team Foundation Server. This means you can work with a Git repository in Team Foundation Server in exactly the same way as you would with any other implementation. Second, despite the history of Team Foundation Server, both version control options are considered equals and are supported fully. The project management functions in Team Foundation Server are still available and code changes can still be linked to work

items. Distributed version control is covered in detail in Chapter 7.

Team Collaboration

Team Foundation Server 2012 introduced a built-in set of experiences for requesting, responding, and managing code reviews. It uses the powerful work item tracking experiences behind the scenes as well as some specialized user experiences to help you discuss changes. This was a powerful way of formalizing collaboration between team members.

A new feature has been included in Team Foundation Server 2013 that allows team members to make comments directly against code, right from Team Web Access. You can make comments on an entire file, specific lines of code, or even a complete changeset, and just like code reviews, you can have threaded discussions. This provides a great way for team members to collaborate, directly against the code itself, without having to manage the complete code review process.

A new Team Room has also been added to Team Web Access in the 2013 release. This feature not only allows developers to chat in real time, but it notifies them of relevant events such as build completions and code changes.

Web Access

Team Web Access was completely redesigned in Team Foundation Server 2012 to provide an even better experience for those without any of the traditional clients available. It is friendly to modern browsers, including mobile browsers, and works well with both desktop and tablet devices.

Microsoft has further updated and improved Team Web Access in Team Foundation Server 2013. The agile management tools have also been improved and now have support for tags, backlog and board improvements, updated process templates, and support for Agile Portfolio Management. Some new features have been implemented for Team Web Access as well, including a new Test Hub, Team Rooms for live chat, and work item charts.

Agile Product Management

Additional new experiences added to Team Web Access are agile project management and product planning. The new Agile Planning tools are specifically designed for users practicing Agile development, but can actually be beneficial for those using any process.

The primary changes introduced in this release include Agile Portfolio Management and a set of improved process templates for managing projects.

The great thing about these changes is that they allow you to roll-up requirements so management can see just the level of detail they are interested in. Each team can have its own set of backlog items that contribute to a shared “feature” backlog.

Release Management

Release Management for Visual Studio 2013 is a powerful tool for automating the deployment pipeline for developed applications. Formerly known as InRelease, Microsoft acquired the product from a Canadian company called InCycle in June 2013.

The Release Management tool allows teams to deploy applications to multiple server environments and has strong integration with Team Foundation Service. Complex

release workflows can be defined with a visual interface. Importantly, teams can define the promotion path of a single build through multiple environments and enforce a process of approval and promotion.

> **Note**
>
> *Release Management for Visual Studio 2013 is discussed in more detail in Chapter 20.*

Acquisition Options

Microsoft has also greatly improved how you may acquire Team Foundation Server. Several options are available to you, as discussed in the following sections.

Licensing can be somewhat confusing, but Team Foundation Server licensing follows the licensing pattern of other Microsoft server products. There is a server license. Additionally, with some notable exceptions, each user that connects to the server should have a Client Access License (CAL) for Team Foundation Server.

> **Note**
>
> *For more information about those potential exceptions, or questions about what you will need to purchase, you can seek help from a Microsoft Partner with the ALM Competency or your local Microsoft Developer Tools Specialist, or you can refer to the End-User License Agreement (EULA). A licensing white paper dedicated to Visual Studio, MSDN, and Team Foundation Server is also available at* http://aka.ms/VisualStudioLicensing.

Visual Studio Online

By far, the easiest way to get started with adopting Team Foundation Server is through a new hosted option available directly from Microsoft called Visual Studio Online (formerly Team Foundation Service). It shares a majority of the same code base as the same Team Foundation Server product used on-premises but modified to be hosted from Windows Azure for multiple tenants. It is available at http://tfs.visualstudio.com.

The best part of using Visual Studio Online is that your team need not worry about backups, high availability, upgrades, or other potentially time-consuming administration and maintenance tasks. Another nice thing is that Visual Studio Online customers will receive frequent updates that even include new features before on-premises customers.

Note

Brian Harry, Product Unit Manager for Team Foundation Server, announced that the internal product teams improved their engineering process so well over the past two to three years that they are able to quickly provide more frequent updates. Starting with Team Foundation Server 2012, the product team has provided frequent updates that include the typical performance and bug fixes but also brand new features. The frequency of updates since this announcement has been significantly faster than in the past, with four post-RTM releases to Team Foundation Server 2012 and one Team Foundation Server 2013 release in the past 12 months.

Team Foundation Service customers have seen updates made more frequently than the on-premises edition. Brian mentioned that his teams are able to deploy hotfixes daily, but full-featured updates have thus far been released every three weeks (with a small number of exceptions), which lines up with the internal sprint schedule. You can learn more about this topic from Brian Harry's blog post at `http://aka.ms/TFSReleaseCadence`.

One thing to take away from this discussion is to make sure that your team always uses the latest update of Team Foundation Server if you choose to install it on-premises.

Teams using Visual Studio Online are able to leverage an elastic set of standard build servers. This elastic build service provides standard build machines available and clean for each of your builds. Teams can even integrate their elastic builds with their Windows Azure accounts to provide continuous deployment to instances of their

applications or sites hosted in Windows Azure. Teams can also take advantage of on-premises build servers connected to Visual Studio Online.

Microsoft released the Team Foundation Service as a free trial in late 2012 and rolled it into the commercial Visual Studio Online service in November 2013. They announced that the full feature set will be provided to teams of up to five at no cost. Additionally, MSDN subscribers will be able to leverage Visual Studio Online as an additional benefit to their MSDN subscription.

Visual Studio Online is offered under a subscription model. The Basic plan is free for five users, and subsequent users can be added on a per-user, per-month basis. In addition to the Basic plan, two other user plans are available with differing capabilities and inclusions.

To date, Visual Studio does not have full parity with the on-premises product. For example, the lab management, reporting, and process template customization capabilities are features not currently available. As Visual Studio Online evolves over time, there will be greater, if not full, parity with the on-premises edition.

In the meantime, for teams that would like the full set of features but still have someone else manage their Team Foundation Server instance, options are available through several third-party hosting companies.

Express

Small software engineering teams can leverage an Express version of Team Foundation Server 2013 that is available and free for up to five developers. Team Foundation Server Express is available at http://aka.ms/TFS2013Express. The Express edition includes, but is not limited to, the following core developer features:

- Version control
- Work item tracking
- Build automation

This is a perfect start for small teams that want an on-premises Team Foundation Server instance without any additional costs. If your team grows beyond five, you can always buy CALs for users six and beyond. The Express instance can even be upgraded at any time to take advantage of the full set of features without losing any data.

Trial

One of the easiest ways to acquire Team Foundation Server is on a 90-day trial basis. You can download the full version of Team Foundation Server and try out all of the features without having to purchase a full copy. The DVD ISO image for the trial edition is available at http://aka.ms/TFS2013Downloads.

If you install the trial edition of Team Foundation Server, you can easily apply a product key to activate the trial edition. You could even move the team project collection from the trial server to a different server instance once your team has decided to fully adopt Team Foundation Server.

Alternatively, if you need a 30-day extension, you can perform one extension using the Team Foundation Server Administration Console once you're near the end of the trial period. You can find out more information about extending the trial by visiting http://aka.ms/ExtendTFSTrial.

If you would rather have a virtual machine that is ready to use (including all of the software necessary to demo and evaluate Visual Studio 2013 and Team Foundation Server

2013), you can download the all-up Visual Studio 2013 virtual machine image. The virtual machine has a time limit that starts from the day that you first start the machine. You can always download a fresh copy of the machine to begin your demo experience over.

> **Note**
>
> *You can find the latest version of the virtual machine available at* http://aka.ms/vs13almvm.

Volume Licensing

Microsoft has plenty of options for volume licensing, including Enterprise, Select, Open Value, and Open License Agreements, that will help your company significantly reduce the overall cost of acquiring an on-premises edition of Team Foundation Server. Different options are available based on your company size and engineering team size. This option is by far the most popular choice for companies looking to acquire Team Foundation Server, MSDN subscriptions, and Visual Studio licenses.

If your company acquired an earlier version of Team Foundation Server through a volume licensing program, and also purchased Software Assurance (SA), you may be entitled to a license for Team Foundation Server 2013 without additional cost, if the SA was still active on the date that Team Foundation Server 2013 was released.

> **Note**
>
> *For more information about volume licensing, discuss your options with your Microsoft internal volume licensing administrator, your local Microsoft Developer Tools Specialist, or a Microsoft Partner with ALM Competency. You can find out more information from the Visual Studio Licensing white paper available at http://aka.ms/VisualStudioLicensing.*

MSDN Subscriptions

Beginning with the Visual Studio 2010 release, a full production-use license of Team Foundation Server 2013 is included with each license of Visual Studio that includes an MSDN subscription. Those MSDN subscribers also receive a Team Foundation Server 2013 CAL available for production use.

This now enables developers, testers, architects, and others with an active MSDN subscription to take advantage of Team Foundation Server without additional licensing costs.

> **Note**
>
> *For more information about MSDN subscriptions and for links to download Team Foundation Server 2013, visit the MSDN Subscriber Downloads website at http://msdn.microsoft.com/subscriptions.*

Microsoft Partner Network

Companies that are members of the Microsoft Partner Network and have achieved certain competencies can be

entitled to development and test-use licenses of several of the products included with an MSDN subscription, including Team Foundation Server 2013.

> **Note**
>
> *For more information about the requirements and benefits available for Microsoft Partners, visit* http://partner.microsoft.com.

Retail

If you are not able to use any of the other acquisition methods, you can always acquire Team Foundation Server 2013 through retail channels, including the online Microsoft Store. You can purchase the product directly from Microsoft online at http://aka.ms/TFS2013Retail. It is also available from many other popular retail sites.

One of the nice benefits of purchasing a server license using the retail channel is that you also receive a CAL exclusion for up to five named users. This benefit is available only from licenses purchased through the retail channel, and it is not included with other acquisition avenues discussed in this chapter.

Summary

As you learned in this chapter, Team Foundation Server is a product with lots of features and functionality. This chapter introduced the types of features available, including those new to the latest release. Additionally, you learned about the different acquisition methods for getting the software for Team Foundation Server.

The next few chapters will familiarize you with planning a Team Foundation Server deployment and installing a brand-new server. You will also learn about the different methods available for connecting to your new server. Chapter 2 begins that discussion with an examination of deploying Team Foundation Server.

Chapter 2
Planning a Deployment

What's in this chapter?

- Organizing a plan for adoption
- Setting up timelines for adoption
- Structuring team projects and team project collections
- Hardware and software requirements

Before installing or configuring Team Foundation Server, it is helpful to lay out a plan and to identify the areas that need some preparation work to ensure a successful adoption. This chapter discusses methods for gaining consensus in your organization for the adoption. You will learn about some potential adoption timelines and strategies to ensure a smooth transition for your teams from legacy systems to Team Foundation Server 2013. Finally, the discussion walks you through some of the immediate preparation steps for gathering what you will need before you start the installation and configuration of your new Team Foundation Server environment.

In Chapter 1, you read about the high-level features available in Team Foundation Server 2013, including new features for this release. Now it's time to convince your boss and team that it would be worthwhile to adopt Team Foundation Server 2013. The following sections examine some of the ways you can prepare for your proposal to your team.

Identifying and Addressing Software Engineering Pain

One key to selling the idea of an Application Lifecycle Management (ALM) solution is to identify the pain points that your organization and teams are experiencing, and to address those pain points with possible solutions. You may find that some of the common problems

people are seeking solutions for are the same problems that plague your organization.

This section identifies some common problems that plague many development organizations, and it provides some helpful discussion about how Team Foundation Server and the ALM family of Visual Studio 2013 products attempt to address those pain points. You may have additional pain points that you would like to solve, and it is always good to flesh those out to ensure that you are identifying and solving them as part of your adoption of Team Foundation Server.

This book covers many of the ALM topics because Team Foundation Server is a part of the Visual Studio ALM family of products. You can find out more information about all of the different ALM tools available across the Visual Studio family in the companion book *Professional Application Lifecycle Management with Visual Studio 2013* available at http://www.wiley.com/WileyCDA/WileyTitle/productCd-1118836588.html.

Transparency of the Release or Project

Does your team have difficulty understanding any of the following questions during the release cycle?

- Are we on track to release at the end of the month?
- How much implementation has been accomplished on the requirements in this release?
- How are our testers doing in authoring and executing their test cases?
- What changed in last night's build?
- Why did this set of files change in this recent check-in? Was it used to implement a requirement or fix a bug?
- Which requirements are getting good test coverage versus requirements that are largely untested?
- Is the company investing in the right areas of our products based on feedback from stakeholders?
- How do you balance the capacity of team members against the priority of work you want to accomplish?

- How much work is your team capable of delivering for a given iteration?
- How can you be sure the release in production matches the release that was in a test environment?

Teams that have a problem getting visibility into their release process often want to start by finding a tool that will gather all of the data necessary to easily answer some of these questions. Team Foundation Server is one of the best products available for transparency because it allows you to store all of the different artifacts from the beginning to the end of the software development life cycle. Not only does it provide a way to capture that information, but it also allows you to make informed decisions using rich links between those artifacts and systems. Team Foundation Server provides rich information by exposing the end-to-end relationships that exist across artifacts.

Collaboration across Different Teams and Roles

Some teams have difficulty providing information and communicating across different functional groups. Testers may not feel that they have enough information about bugs returned to them as rejected, and developers may feel that they don't have enough information about the requirements they are supposed to implement or the bugs they are supposed to fix. Stakeholders and business users may not feel that they have an easy way to provide feedback about the applications they interact with so that the software development teams will be able to act appropriately.

If your team is experiencing some of these problems, you may benefit from being able to easily see information and notes about the different artifacts in the software process stored in Team Foundation Server.

Automated Compilation, Testing, Packaging, and Deployment

Teams may end up spending a lot of time at the end of release cycles completing manual steps to compile, package, and deploy their applications. They may be performing these actions on a developer machine, and manually copying to staging and production environments.

These manual steps are often error-prone and can result in unforeseen issues and failed deployments. By taking advantage of the automated build system in Team Foundation Server, your team can reduce the complexity of this process and turn it into a repeatable end-to-end solution that occurs at regular intervals or is triggered by changes introduced by your developers.

Additionally, you can leverage the automated build system to introduce a “gauntlet” of checks that each check-in may go through to verify the quality of those changes by using the gated check-in feature in the Team Foundation Build system. This can help your team reduce entropy by preventing defects from ever being introduced to the version control repository.

> ## Note
>
> *The new Release Management for Visual Studio 2013 product allows teams to define, configure, and automate deployments to multiple target environments. This tool can reduce risk and ensure repeatable deployments that support your organization's release pipeline and approval process.*
>
> *See Chapter 20 for more information about Release Management for Visual Studio 2013.*

Managing Test Plans

The testing or quality assurance departments may be organizing their test cases using Word or Excel documents, which can make it hard to organize your catalog of test cases. Additionally, tracking the execution progress of each test case may be extremely difficult; thus, it becomes difficult to gauge the progress of testing in your release.

Team Foundation Server allows you to manage your collection of test cases; it also allows you to manage the progress of test execution during the release cycle. This includes the ability to track tests across multiple configurations that your product or application needs to support.

Parallel Development

Development teams have notoriously experienced difficulty in managing changes across multiple lines of development that can occur concurrently. By supporting the previous release, while stabilizing the next release, and then also performing early work on a feature release, you can end up having trouble keeping each of those parallel lines of development organized. Integrating changes made between those releases is especially time-consuming and error-prone, especially if developers are manually applying fixes to each of those code lines.

Testers are often curious about how bug fixes and changes have been included in a particular release. And they might often need to figure out if fixes have been applied to other releases of the application.

The Team Foundation Version Control system in Team Foundation Server provides excellent branching and merging tools for easily managing parallel development, including the ability to track changes across branches. This helps you to easily see which branches have changes integrated into them as well as how and when those changes got there.

> **Note**
>
> *The new Distributed Version Control system gives developers more freedom to work on multiple parallel development tasks at once. Using Git, developers can work on multiple branches within local repositories before confidently committing changes to the server.*
>
> *See Chapter 7 for more information about Distributed Version Control with Git.*

Adopting Team Foundation Server

The maximal value from Team Foundation Server is realized when it is used in a team. Therefore, ensuring a successful Team Foundation Server adoption requires alignment with many people in your organization. The following sections should help you avoid some common pitfalls and provide you with some suggestions on

where to start with what may seem like a large and daunting product.

Adoption Timeline

In general, splitting up the adoption by team/application has proven to be a successful approach. Some of the effort may end up being the same for most teams, and lessons you learn from earlier transitions will help you become more successful in the following transitions. Table 2.1 presents a sample adoption timeline.

Table 2.1 Sample Adoption Timeline

Activity	Estimated Time
Planning for deploying Team Foundation Server	One week
Identifying the process to adopt and process template customizations	Two to four days
Designing the branching and merging strategy	One day
Customizing the process template (dependent on the level of customization of the process identified)	One to four weeks

Table 2.2 discusses the additional adoption steps for each team or application.

Table 2.2 Sample Adoption Timeline for Each Team

Activity	Estimated Time
Developing a custom migration tool (needed only if not using one commercially or freely available)	Two to four weeks
Testing the migration	One to two weeks
Initial training sessions for teams (occurs one week before transition)	Half a day for each team member
Migrating source code	One to two days
Migrating work items	One to two days
Follow-up training sessions for teams (occurs one week after transition)	Half a day for each team member

Phased Approach

Another approach that works well is adopting each piece of Team Foundation Server separately in different phases of a larger deployment project. This allows you to plan each phase, execute that adoption, and then train the teams on the new features and processes available in that particular phase.

Some find that teams are better able to absorb training for the new features when they are exposed to them incrementally. You may have a higher success rate by splitting up the adoption project, and then focusing your time and attention on making each part succeed. However, you'll eventually find some teams very eager to adopt future phases, so be sure you don't keep them waiting too long!

When introducing any new tooling into a large organization, it is important that you address the key pain points first. Many companies will identify traceability of work through the development life cycle, and this is often an area that is poorly addressed by existing tooling. For others, the version control system being used may be out-of-date (unsupported) and performing poorly. It is, therefore, usually the version control or

work item tracking components that people begin using when adopting Team Foundation Server.

Luckily, Team Foundation Server is flexible enough that you can still get value from the product when using only one or two components of the system. Once you have adopted both version control and work item tracking, the next area to tackle to gain the most benefit is likely to be Team Foundation Build. By automating your build system and increasing the frequency of integration, you reduce the amount of unknown pain that always occurs when integrating components together to form a product. The key is to gradually remove the unknown and unpredictable elements from the software delivery process, and to always look for wasted effort that can be cut out. Using the new Release Management helps your team deploy the same build across multiple environments, no matter how complex the configuration.

Automating the builds not only means that the build and packaging process becomes less error-prone, but it also means that the feedback loop of requirement traceability is completed. You are now able to track work from the time that it is captured, all the way through to a change to the source code of the product, and into the build that contains those changes.

At this point, you may identify the need to document your test cases and track their execution throughout the release. With the traceability between test cases and requirements, you'll be able to better identify the requirements in your product that are covered appropriately by your testers.

After a period of time, you will have built up a repository of historical data in your Team Foundation Server data warehouse, and you can start to use the reporting features to predict if you will be finished when you expect (for example, is the amount of remaining work being completed at the required rate?). You will also be able to drill into the areas that you might want to improve—for example, which parts of the code are causing the most bugs.

To put all of this together, you will more than likely end up with the following adoption phases, but you will want to adopt them in the order that works for your organization:

- Phase I: Version Control

- Phase II: Work Item Tracking
- Phase III: Automated Builds
- Phase IV: Test Case Management
- Phase V: Reporting
- Phase VI: Virtual Environments and Lab Management

You'll notice that this book has generally been laid out in this order to help you address each area in order of typical adoption.

> **Note**
>
> *After getting used to the tooling, you should look at your overall process and adopted process templates to ensure that all of the necessary data is being captured—and that all the work item types and transitions are required. If there are unnecessary steps, consider removing them. If you notice problems because of a particular issue, consider modifying the process to add a safety net. It is important to adjust the process not only to fit the team and organization, but also to ensure that you adjust your processes when you need to, and not only because you can. See Chapter 12 for more information about process templates, work items, and other topics related to tracking work.*

Hosting Team Foundation Server

For the team to have trust in Team Foundation Server, you must ensure that it is there when they need it, and that it performs as well as possible. For organizations that depend on creating software, your version control and work item tracking repositories are critical to getting your work done. Therefore, those features should be treated on the same level as other mission-critical applications in the organization.

The Team Foundation Server infrastructure is a production environment for your company. Ideally, it should be hosted on a server or multiple servers with adequate resources (both physical memory and disk space). If hosted in a virtual environment, then

you should ensure that the host machine has sufficient resources to handle the load of all guest machines, including superior disk I/O performance.

When planning upgrades, configuration changes, or when performing training, you should use a test Team Foundation Server environment. For some organizations, the test requirements justify the purchase of a hardware platform equivalent to the production environment.

However, for many scenarios, using a virtual Team Foundation environment will provide a suitable environment for testing. These virtual environments are especially useful when developing a new process template or testing work item process modifications. Microsoft provides an evaluation version of Team Foundation Server preconfigured as a virtual hard disk (VHD) file. This is frequently used as a test bed for work item modifications and new process templates.

> ## Note
>
> *Brian Keller, Microsoft's Principal Technical Evangelist for Visual Studio ALM, publishes frequently-updated virtual machines with Team Foundation already set up and populated with working projects. These can be extremely useful for demonstration purposes and for testing new work item templates and plug-ins. You can find the full list of Visual Studio ALM virtual machines at* http://aka.ms/almvms.

Identifying Affected Teams

One essential activity is to identify all of the different teams in your company that would be affected by deploying Team Foundation Server. Following are some examples of those types of affected teams:

- Developers
- Testers/Quality Assurance
- Product/Program Managers

- Project Managers
- Business Analysts
- Designers
- User Experience
- Change Management
- Release Management
- Technical Documentation/User Education
- Technical Support
- Information Technology
- Executives or other stakeholders
- Business Users
- Remote Teams

Generating Consensus

If anything, you should over-communicate any plans for rolling out Team Foundation Server and/or a new process. Change is difficult for some people, and this has been one technique that seems to ease those concerns.

Once you have identified all of the affected teams, it's helpful to generate consensus by suggesting that each team nominate a team member to represent the team as decisions are made about the deployment. This is generally a good way to ensure that everyone is involved, and that information ends up getting disseminated throughout the organization. You can have this “advisory” group help determine how to configure the server and the process that ends up being adopted. Going through this process allows those who are involved to have some buy-in to the deployment and, ultimately, champion the success of the change within their teams.

One word of caution, however, is that you should be sure to have an executive stakeholder in this group who is available to make final decisions when there are disagreements. It's important to ensure that decisions made by the group end up benefiting the business, so having this “final-authority” representative is helpful. The others in the group will feel better about giving their input and