1

Planning a Team System Deployment

It can't be stated enough that planning is an essential part of any process. Without planning, you may unnecessarily complicate the deployment process or, worse yet, miscalculate what your needs are and have to reinstall the product. (As a consultant who has deployed Team System hundreds of times, I speak from experience.)

Deployment does not necessarily mean just the installation of the product. Once Team System has been installed, you must configure it correctly. (For example, you must set up the proper permissions, change the process template to match the target environment, and handle other details including source code migration and extension or customization of tools to make them interact with Team Foundation Server). If you are a consultant, you'll often get client requests to install Team System in, say, two days. Team Foundation Server takes at the very least a day to correctly install and configure in an enterprise environment. Working on a two-day engagement to deploy Team System is the equivalent of asking a home builder to build a house from scratch in two weeks! It's best to set the proper expectations with your clients, work out a deployment plan, and work out realistic timeframes.

In this chapter, you learn the major components of Team System and how to plan your deployment based on your capacity and needs. The final part of the chapter discusses strategies to help you migrate your existing tools and data to Team System.

Team Foundation Server Overview

One of the big challenges of working on a software development team is getting all the members of the team to collaborate seamlessly. There are always so-called silos that are created for each role on a team. For example, developers like to live in Visual Studio and can easily relate to development talk and issues, whereas most project managers have no Visual Studio experience whatsoever; they are used to working with Excel spreadsheets working up use cases, scenarios, and project milestones.

Architects map out application designs in Visio, testers use testing tools. The groups and roles are quite different. Trying to integrate it all is difficult and requires a lot of coordination — and frequently intercession — on the part of the project manager, who must set up status meetings and track deliveries from a number of sources.

To successfully deploy Team System, you'll need cooperation from the entire development team, and especially the operations team. Installing Team System involves setting up accounts in Active Directory, changing firewall port settings, prepping hardware, making sure the licensed software is available on DVD or on a network share, and so forth. Get the operations team involved early in the process and get them to read the Team Foundation Server installation guide (and quiz them on it). Both of these steps will make the deployment process a lot easier.

Team System (and specifically Team Foundation Server) provides tooling and a framework to simplify collaboration between team members. The server allows you to communicate with anyone on your team using a number of tools including the Visual Studio editions, Microsoft Excel, Microsoft Project, and Team Explorer. All team members get access to a common set of services including a Team Portal, version control, build engine, and reporting. Everything is integrated in one location. All of this means that you need fewer status meetings, and you get more transparency and clarity with regards to the work being done within a project.

Team System can be viewed as three logical tiers: the client tier, which includes the Team Editions of Visual Studio 2005; the application tier, in other words, Team Foundation Server; and the data tier (SQL Server 2005), which provides the data management and storage support behind the scenes. This very basic architecture can be seen in Figure 1-1.

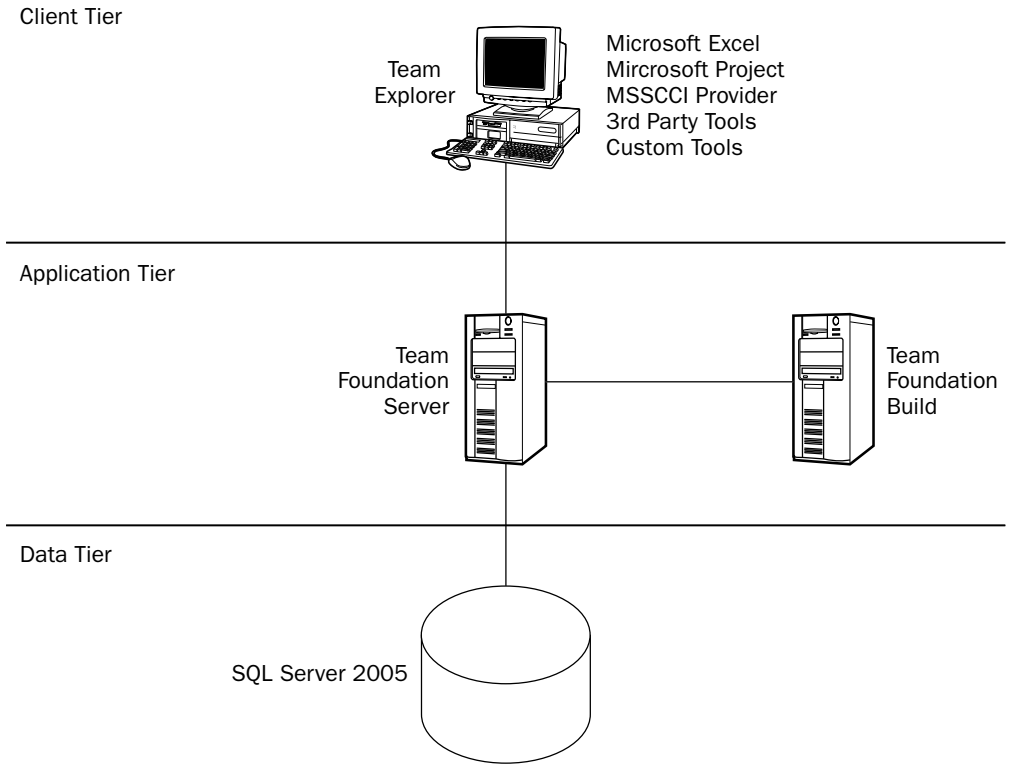

Figure 1-1

When you are planning a deployment, it is important to consider what deployment scenario will work best for your needs and what client/server configuration will give you the best bang for your buck in terms of management and configurability. Let's start by looking at the essential parts that make up the client and server components of Team System.

Team System Overview

Team Foundation Server plays a very important part in Team System. It is the collaborative suite that supports the entire software development life cycle (SDLC). Prior versions of Visual Studio supported developers only and everyone else had to rely on third-party products to achieve any kind of integration. The different tiers of Team System can be further broken down into its client and server components.

> **Brian Harry has posted on his blog a Visio diagram outlining how Team System was deployed within Microsoft. Specifically, it maps out the overall topology of both server and components. You can download the Visio file at the following link:** `http://blogs.msdn.com/bharry/archive/2006/08/22/712746.aspx`**.**

Client Components

Here is a listing of Team System's core client components. You can't really talk about a server without discussing how the clients interact with the server (of course). Along with these clients, the Team Foundation Server API (also known as the Team Foundation Core Services) contains methods that allow you to programmatically connect to Team Foundation Server (and create your own custom clients).

- **Visual Studio 2005 Team Editions** — Even though this is a book on Team Foundation Server, we would be remiss not to cover client features and how they integrate with Team Foundation Server. Refer to the individual editions below for details on the coverage level.
  - **For software architects** — Unfortunately, this book has little to no coverage of the architecture tools. If you are interested in Team Edition for Software Architects, we would like to refer you to *Professional Visual Studio 2005 Team System* (Wrox Press, ISBN: 0764584367).
  - **For software developers** — For developers, the book covers version control management (Chapter 12) and extensibility (Chapters 9 through 11).
  - **For software testers** — For testers, refer to Chapter 15 for information about the Team Test Load Agent. We also cover test case management to a limited degree in Chapter 13.
  - **For database professionals** — Chapter 8 is devoted to Team Edition for Database Professionals and how the data development lifecycle ties into the software development lifecycle.
  - **Visual Studio 2005 Team Suite** — Visual Studio 2005 Team Suite integrates the features of Team Edition for Software Architects, Team Edition for Software Developers, and Team Edition for Software Testers. Starting late 2006, Team Suite will also include Team Edition for Database Professionals.

- **Team Explorer** — Team Explorer ships on the Team Foundation Server media and provides connectivity between Visual Studio 2005 and Team Foundation Server. You can learn more details about Team Explorer in Chapter 2.
- **Third Party Tools** — There are a number of third-party companies developing tools for Team Foundation Server. When appropriate, we have provided you with links to these complementary tools.
- **MSSCCI Provider (for Visual Studio 6.0 and 2003)** — Many companies have made investments in .NET 1.0 and .NET 1.1, and need to support Visual Basic 6.0. The Microsoft Source Code Control Interface provider allows these IDEs to connect to Team Foundation Server.
- **Microsoft Office Excel 2003** — Microsoft Excel is used as a project management tool within Team System. (A developer or any other team member can also use it to manage their work items.) You can learn how to make the most out of Excel in Chapter 13.
- **Microsoft Project 2003** — Microsoft Project has special capabilities and limitations that are documented in Chapter 13.

Server Components

These server components provide the infrastructure backbone for Team System. In this book, we examine each one of these in detail:

- **Team Foundation Server** — Team Foundation Server is comprised of a number of components and services. The installation of the product is covered in Chapter 2, you learn about backup and recovery strategies in Chapter 5, right up to retirement in Chapter 17.
- **Team Foundation Build** — Team Foundation Build provides an automated, integrated build experience. You can learn a great deal more about Team Foundation Build in Chapter 3.
- **Team Test Load Agent** — The Team Test Load Agent is composed of an Agent and Controller, which allows you to test Web applications against a profile of a thousand users or more. There is coverage of these tools in Chapter 15.
- **Team Foundation Server Proxy** — Team Foundation Server Proxy is a tool that helps improve the performance of Team Foundation Version Control over HTTP. We cover the proxy in Chapter 15.
- **Team Foundation Core Services** (TFCS) — Team Foundation Core Services is a set of services and APIs that allow you to extend Team Foundation Server. You can learn more about extensibility in Chapter 9.
- **Active Directory domain controller** — If you are working within a big enterprise, Active Directory (AD) is essential for the management of your user's roles and credentials. You'll find deep coverage of AD in Chapters 2 and 4.
- **Mail server** — Team Foundation Server has the ability to leverage a mail server to send out alerts to your team members. The alert and eventing infrastructure is covered in several chapters, most notably in Chapter 14.

Compiling Your Project Data

There is key data you should compile to plan a deployment. Otherwise, you may be unprepared for the installation and it may be unnecessarily complicated. Here are some of the data points you should collect:

- **Hardware infrastructure** — This includes a full audit of all the target systems that will be used alongside Team System. This audit should be undertaken in consultation with your operations team. Equipment usually takes time to order, therefore you may want to consider dates — a nontrivial requirement because they will affect the dates in which you can start the deployment. Part of the hardware infrastructure review includes writing a maintenance plan, which includes backup/recovery, a maintenance task list for administrators to perform regularly, and so forth. Your operations team should be deeply involved in this process.
- **Software infrastructure** — List all the software required for the deployment, and the software currently in place in your environment. The software check may reveal systems with incompatible software, in which case the operations team must upgrade the target systems before starting the deployment process. A close consultation with your operations team is key to obtaining a solid understanding of the target environment. Another element that can't be understated is licensing. Do you have all the necessary licenses for Team System? A good place to start is look at the 1:1 correspondence between the software you need and licenses you need. After that, you will need to look at client access licenses (CALs) for those who will be accessing Team Foundation Server without a Team Edition.
- **Network infrastructure** — Your network infrastructure may appear deceptively simple, but there are many things to consider including security settings, user account settings, topology challenges, and so forth. For example, if you have governance rules in place, your user accounts may have special restrictions such as password policies that will complicate a deployment. Another scenario is that ports may need to be opened on your network to allow Team Foundation Server communication to flow through. In order for the ports to be opened, a security audit might be required. It's important to note not only the components that are required, but also the process behind those components.
- **Project documentation** — Find all documents relating to your software development process and the documents supporting that process, including templates, lists, and so on. If some of these seem ad hoc, not to worry. Team System provides a solid mechanism for aggregating and publishing software project assets.
- **Build requirements** — The complexity of your build may depend on many factors, including what type of software you are developing, if you are building a multiplatform or distributed solution, and so forth. The key information you will want to record includes what needs to be built, what you are expecting as a release at the end of the build process, and if any special tasks are required along the way (for example, automatically copying files via FTP to another server).
- **Source code** — How much source code do you have? Where is it stored? What policies and procedures have you put in place to track versions and integrate source code? You must document all of this to decide how you will migrate the code and what approach to take to configure Team Foundation Version Control.

Another thing you can do that can't be understated is creating a checklist before deployment. Figure 1-2 shows the installation checklist available in the Visual Studio Team Foundation Installation Guide. It breaks down the installation steps for both single-server and dual-server deployments.

You can create a personal checklist in Excel that reflects the software, hardware, and tasks you need to accomplish to successfully deploy Team System. Figure 1-3 shows an inventory of software, which you can check off as you obtain it. Once all the software has been checked off the list, for example, you know that you are ready for the deployment.

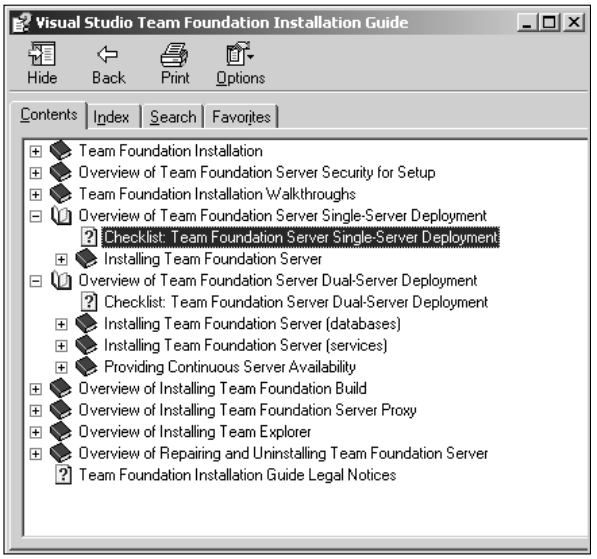

Figure 1-2

	A
1	Team Foundation Server Deployment Checklist
2	Description
3	**Software Requirements** (*on DVD or accessible network share*)
4	- Windows Server 2003
5	- Windows Server 2003 R2
6	- Team Foundation Server RTM
7	- Visual Studio 2005 Team Editions (for each team member)
8	- Team Edition for Software Developers
9	- Team Edition for Software Testers
10	- Team Edition for Software Architects
11	- Team Edition for Database Professionals
12	- Team Suite (for Team Foundation Build server)
13	- Windows SharePoint Services Service Pack 2
14	- Microsoft SQL Server 2005
15	- Developer Edition
16	- Standard Edition
17	- Professional Edition
18	- Enterprise Edition
19	- Microsoft Office 2003 Professional
20	- Microsoft Office 2007 Professional

Figure 1-3

Planning a Deployment

Installing and deploying a product such as Visual Studio 2005 Team System involves a lot of preplanning and forethought. You must consider variables such as capacity, scale, disaster recovery planning, backups, and the underlying infrastructure. There are several important sources you can refer to that will help in your deployment efforts:

- **Existing infrastructure documentation** — If your company had to write up a disaster recovery plan (which may have stemmed from the Year 2000 fiasco), then you are in a great position as all your software and assets have been catalogued.
- **Visual Studio Team Foundation Planning Guide** — Along with the Installation Guide (`TFSInstall.chm`) and Administration Guide (`TFSAdmin.chm`), the Planning Guide will provide you with hints on what to look at in the planning process. Unlike the other guides (which are also great tools to help you plan your deployment), the Planning Guide (`TFSPlanning.chm`) is not available on the Team Foundation Server CD or DVD.
- **Microsoft Operations Framework (MOF)** — The Microsoft Operations Framework provides guidance on the proper implementation of technology based on Microsoft's own internal experience and practices derived from the IT Infrastructure Library (ITIL). You can learn more about MOF at `microsoft.com/technet/itsolutions/cits/mo/mof/`
- **Governance documents** — Your company may need to follow strict or specific rules for legal reasons. You need to consider these governance rules and practices when planning your deployment.

Some of the key questions you have to ask include whether you'll need one or two Team Foundation Servers (depending on the scale of your team). Will you need a proxy server? (In other words, will you support geographically distributed teams?) Do you need more than one build server? If so, where will they be installed? Do you need to create load tests with more than a thousand simulated users? If so, a test rig may be required. Here is a more in-depth look at these variables and how they may affect your implementation of Team System.

Capacity Planning

The best way to look at capacity planning is by thinking about and looking at scenarios. First, we look at the performance and scope of a deployment. Next, we look at deployment on a small and then a larger scale.

Performance and Scope

Team System was designed to work with a single Team Foundation Server instance at the core. Microsoft is currently investigating scenarios where multiple Team Foundation Servers instances are used to scale up to larger sized teams. However, note that Team Foundation Server does not currently support clustering and mirroring. (However, this is supported on the SQL Server 2005 data tier, which is really the key area because all project assets are stored in the database.) Team Foundation Server supports a warm standby scenario, if an issue should occur. For more information about availability within Team System, refer to "Ensuring Team Foundation Server Availability" on MSDN at `http://msdn2.microsoft.com/en-us/library/ms253159.aspx`. The Microsoft Operations Framework also has information about service management functions, including availability management, at `http://www.microsoft.com/technet/itsolutions/cits/mo/smf/smfavamg.mspx`.

The responsiveness of Team System depends greatly on the amount of memory you have in your Team Foundation Server and, most important, your SQL Server 2005 instance. The more memory and processor power you can add in, the better your user experience will be.

In planning your deployment, you must consider whether you will use Team Foundation Server for one or two configurations. Some of the questions you might be asking yourself include the following:

- How many users will you support?
- Are Proxy Servers required (for distributed version control support)?
- Will you support clusters of remote users?
- Where on your network is your build server?
- Do you need test rig and how many test users will you simulate?
- Will you integrate with Active Directory?

Small-to-Medium Deployments

There are two fundamental scenarios to consider for small-to-medium deployments: Let's first define a small-to-medium deployment as one to 2,000 users.

A one-user scenario may include a customer evaluating a demo version of the product, a consultant giving a presentation on Team System (perhaps through a single machine VirtualPC install), or even a small learning environment to help a group of users experiment with the product. The one-user version of Team System is usually installed on a single machine and may contain demo or evaluation versions of the product (rather than the full retail version of the product). The single machine install contains all the components of Team System including the client tier (CT), build engine, data tier (DT), and application tier (AT).

The second scenario in small-to-medium deployments is for two to 2,000 users. In this scenario, Team System is installed on a single server and deployed to a small team. Because there are multiple users, the client tier (in other words, Visual Studio) is installed on machines separate from those with the application tier and data tiers. This allows multiple users to access a single instance of the server. You can also optionally install Team Foundation Build on a separate machine or even on the client's desktop. This deployment model is configured to support workgroups or Active Directory.

Enterprise Deployment

The final scenario to consider is the very large team of more than 2,000+ developers, testers, and architects (assuming a dual-server install). To go beyond 2,000 users, you need to bump up the processor scale and performance along with memory on both the application and database tiers. A large infrastructure requires larger capacity, security, manageability, and support for geographically distributed software teams. Team System supports a large team by dividing the tiers on different machines. (Note that a single machine will not support anywhere near 2,000 users.) To support such a large number of users, you must use Active Directory 2000 or 2003. (Otherwise, from an operational perspective the management of the users and shifting needs will require more overhead than can be afforded.) Team Foundation Proxy allows distributed teams to access the source control portion of the product. You can optionally set up multiple build servers, multiple proxies, and, in some cases, multiple Team Foundation Server!

At the time of writing, Microsoft is internally evaluating Team System as a tool to manage ambitious development projects such as Windows or Office. In fact, the Team Foundation Proxy was designed to effectively tackle latency challenges with remote teams. The Prescriptive Architectural Guidance Group, as well as a number of other small organizations, is currently deploying Team Foundation Server.

For updates about how Team System can scale for larger teams, we highly recommend you look at Brian Harry's blog at `http://blogs.msdn.com/bharry/`.

Network Topologies

Depending on your target environment, the network topology may present a special set of challenges. Here are common deployment models that you may encounter:

Single-Server Deployment (Workgroup Configuration)

A single-server deployment is the simplest configuration option you can pick. It is advised if you want to deploy Team System for testing purposes or for very small teams. Typically, the single-server deployment is set up using a workgroup configuration (although it is possible to set it up on a domain).

We define a single server as a 2.2 GHz Pentium IV or Athlon, with 1 gigabyte (GB) of memory that will support a team of 50 or less. If you double the memory, support goes to 250. If you use dual-processor support on both machines with 4GB memory, support for 500 users is obtained. You may also wish to consider your build requirements, which may further determine the machine sizing.

The MSDN Subscriber Download site provides virtual machine images of a single-server deployment, which makes it very handy for you to test and evaluate Team System in a lab-like environment. You can deploy these virtual machines using either Microsoft VirtualPC (VPC) or Microsoft Virtual Server. VPC requires at least 1.5 gigabytes of memory for even modest usability. Two gigabytes is highly suggested.

The workgroup version of the Team System installation process has a couple of notable limitations. One of the limitations is that the domain users can't login to the server. (See multiserver deployment for details.) Second, your passwords and user accounts must be synchronized with Team Foundation Server. Otherwise, users will not be able to log in to the server.

Dual-Server Deployment

If you are planning to divide the tiers on separate machines, note that you should set up your servers on a domain. In order for your components to access the domain, it is *extremely* important for you to set up your `TFSSETUP`, `TFSREPORTS`, and `TFSERVICE` accounts using domain accounts. Otherwise, `TFSSERVICE` will be unable to effectively interoperate and authenticate with the domain controller.

Architecting Your Active Directory (AD) Structure

To set up and configure Team Foundation Server in dual-server deployment, you must use computers that are joined to an Active Directory domain. With the Workgroup edition, you have the choice whether you want to join it (or not). If you are working within a large infrastructure, Active Directory will be a given for managing your users on Team Foundation Server.

There are several reasons you would want to use Active Directory over the workgroup configuration. First, from a convenience perspective, you can implement single sign-on. Active Directory has security features that help prevent scenarios such as unwanted clients or servers running on your network. Second, and most important of all, is manageability. In Workgroup mode, you have to manually add all users and groups to the server; if you have hundreds of users, this can be a pain from an administrative standpoint because all changes made to the external network will have to be replicated locally on Team Foundation Server. For example, if a developer decides to leave a company running Team System, the administrator will have to remove privileges from not only the network, but also the server.

Team Foundation Server supports Active Directory 2000 and Active Directory 2003. (Windows NT is not supported.) Team Foundation Server will interact with Active Directory 2000 in native mode; mixed mode is not supported. Specifically, Team Foundation doesn't support NT4 and so does not support AD2000 mixed mode. It does support AD2003 mixed mode. Team Foundation Server also supports one-way trusts, full trusts, explicit trusts, and cross-forest trusts.

Team Foundation Server does not support a configuration of the application and data tiers on separate domains (or subdomains). You can't mix domains and workgroups either; they have to be on the same domain. Finally, make sure that your network isn't using Windows NT 4.0 Domain Controllers.

Test Deployment Using Virtualization

You can test your deployment using a variety of tools including Microsoft Virtual PC (VPC) and Microsoft Virtual Server. Virtual PCs are not just for testing. There is a trend to deploy VSTS on the workstation as a VPC. This makes it far easier for developers to get around IT-mandated machine configurations and is simpler to install. They can also be used for evaluation and training. Also, some organizations have adopted VPC on the Team Foundation Server side. Often this is used for evaluation, but also more and more for pilot efforts. There are many advantages to doing so, especially in a test environment:

- Virtual images created from one product are compatible with the other
- There is no need to reinstall Team Foundation Server. The virtual image can be redeployed to a production server quite easily.
- You can create a base installation of Team Foundation Server that allows you to restore the server to a pristine state rather than try to clean up the projects.

Almost all the features of Team System work within a virtual environment with the exception of profiling. The profiler will not work with 100-percent precision in a virtualized environment, because of virtual driver limitations. Think of it; the profiler uses the core system as a baseline to execute the tests. If the core environment is virtualized, it is difficult if not impossible to get accurate performance readings.

After Visual Studio 2005 Beta 2, the profiler does work with VPC; however, the profiler under VPC does not go down to the bare metal as originally planned. The dev/test team removed the exception that detects whether you are on a virtual machine. It is now left up to the user to know what the profiler meaning is. For most users, this is quite fine, as they want only a relative view of their hot spots, not an absolute view.

Andy Leonard documented a way to configure a virtual domain controller with Team Foundation Server. You can read the details on his Web site: `vsteamsystemcentral.com/dnn/`.

Client Planning

Before attempting to install any software within a large-scale environment, it is quite important to plan your deployment. Microsoft itself is still learning how to effectively deploy Team Foundation Server. Because there are so many variables, it is impossible for us (or Microsoft) to provide absolute guidance in all deployment scenarios. However, we can provide best practices based on our personal experience and expertise.

Team Editions

Microsoft has designed three versions of Visual Studio 2005 to support three roles in your typical software development team: Team Edition for Software Developers, Team Edition for Software Testers, and Team Edition for Software Architects. Before you deploy these products, you have to determine which edition best fits your team members. If there is an overlap between responsibilities or tools, then you can install Team Suite.

Team Suite

Team Suite encompasses the functionality of all the other Team Editions. It is useful to install on Team Foundation Build to get the full range of testing capabilities. Project managers should definitely get a copy of the suite product to be able to view architecture, development, and test solutions across the entire project.

Team Explorer

In most development roles, you are required to connect to Team Foundation Server, be it to generate a work item, upload source code, or run a build. Each instance of Visual Studio that will be performing these tasks needs Team Explorer and an accompanying client access license (CAL).

A project manager who has never used Visual Studio can use Microsoft Excel or Project integration to connect to Team Foundation Server. Keep in mind that Team Explorer will have to be installed regardless on their systems — the Office plug-in gets installed as part of the Team Explorer install process — and the project manager's system will also require a CAL.

The only roles that don't require Team Explorer are the client and upper management. They can examine the project portals and reporting features; the only requirement is a browser. No client CALs are required to view content on the Reporting site or the Project Portal.

There are other Team Explorer–like client tools developed by third-party vendors for the Team Foundation Server; these include Teamplain (a Web interface to interact with the work item database) and Teamprise (a Team Explorer–like plug-in for Eclipse developers).

Security Planning

To correctly configure your security within Team Foundation Server, you have to put some thought into what users and roles you will define within your development environment. You must consider several layers of security:

- Network security (enforced by Active Directory)
- Operating System security on the machine hosting Team Foundation Server
- Security within Team Foundation Server
- Security on a project level

Roles (as defined by the Microsoft Solutions Framework) play a main part in determining security settings. Would you want any developer on your team to create or delete projects on the fly? Probably not. Applying proper least-privileged user account (LUA) principles to your access controls is the best approach. In a nutshell, LUA advocates providing users with just enough privileges to do their job — no more, no less.

Where should you start? The first thing you can do is look at your current Active Directory user and group configuration. If possible, map and define groups within Active Directory that correspond to the roles defined by Team System. For example, architect, developer, tester, and project manager.

To save administrative headaches, you can map these groups to the groups within Team Foundation Server by adding the domain groups as part of the server groups. This will simplify the task of adding users in Active Directory and will provide a single point of contact for all user administration.

Here is a practical example: let's say a tester is promoted to be a test lead (which entails project management tasks). You can simply change the user in Active Directory from the tester group to the project management group. As a result, the permissions will trickle down to Team Foundation Server and the new test lead will gain more control over server functionality.

In a workgroup install, you must manually make changes on both the target operating system and within Team Foundation Server. As this is more administrative overhead, consider using the workgroup installation only if you manage a small number of users.

Refer to Chapter 4 for detailed information about configuring and administering security within Team Foundation Server.

Creating a Test Plan

Before you install Team System, you must consider who will be implementing tests, and how tests will be implemented. If you are running a large project, you may require additional build and test rigs to support the extra load. Testing can be done manually or as part of a build. You should think about what best practices you want to put into place to create an environment that fosters test driven development (TDD).

All tests will run seamlessly as part of a build with the exception of manual tests. As a best practice, you should create dedicated test runs of manual tests. The two reasons are that it will make the tests easier to administer and will not impede the run of automated tests.

You should also look at all the tests available in Team System to see which ones you can leverage. Frequent testing improves the quality of your code and, as a result, improves productivity. Chapter 13 has some coverage on how to implement test case management.

Test Rig Considerations

If you try to run a load test with more than a few users within Visual Studio 2005, Visual Studio may become unresponsive or you may experience performance problems at the outset. If you want to run capacity and performance tests, it is best to use a remote test rig (consisting of a test load agent and controller) to run the tests without affecting the overall operating environment.

Using the test runs that you can configure using Team Edition for Software Testers, you can use a test controller to manage several load agents situated on several machines. By distributing the load, you can most effectively test your applications without loss of productivity. It goes without saying that the larger the project, the more test rigs will be required. Let's now look at the individual components. The Team Test Load Agent is discussed in detail in Chapter 15.

Hardware Requirements

The best source of information for system requirements is the installation documentation. One reason we are discussing the hardware requirements in this chapter is because a lot of practical information is not covered in the documentation.

One of the hardware components you can never have enough of is memory. The more memory you have installed on your target machines, the more satisfactory your experience with Team System will be in terms of responsiveness and capacity. If you are to choose which machine to add the memory, choose the database server, as it will deliver the optimal results.

Team Foundation Server

According to the documentation, Team Foundation Server requires a minimum of 1 gigabyte of RAM. (This includes the application tier and the data tier.) Based on private testing, you are better off assigning at least 2 gigabytes of RAM for the server (one gigabyte for the server component, one gigabyte for SQL Server 2005).

Team Foundation Server has preset capacity limits. Read the capacity document at `http://blogs.msdn.com/bharry/archive/2005/11/28/497666.aspx` to learn more about the number of work items the software can handle in heavier load situations.

Team Foundation Build

The hardware requirements for Team Foundation Build requirements are the same as the Visual Studio 2005 system requirements for the most part, except that support is limited to Windows XP Professional with SP2 and Windows Server 2003 with SP1 Standard or Enterprise Edition. If you have a small project with 5 to 20 users, Team Foundation Build requires at minimum a single processor running 766 MHz, an 8-gig hard drive and 256MB of memory. At the most, in a large project (spanning 250 users or more), Team Foundation Build requires a CPU with dual processors running at 2.8 GHz, 80GB of hard drive space, and at least 2GB of RAM.

> **Build requirements also are based on the duration of the project build. If less than 30 minutes, 1.5 GHz with 512MB RAM is fine. For medium size team with projects taking less than two hours to build, 2.6 GHz with 1GB RAM. For a large team, you will want dual 2.8 GHz with 2GB RAM.**

SQL Server 2005

In a dual-server scenario, the data tier of Team Foundation Server requires a CPU with a single (or dual) processor(s) running at 2.2 GHz, 80GB of hard drive space, and 2GB of RAM. This should be ample to support from 100 to 250 users.

If your team is much larger, you should spec out your target server-class machine to have at least a CPU with quadruple processors running at 2.2 GHz, 150GB of free hard drive space, and at the minimum 4GB of RAM.

Visual Studio 2005

At the bare minimum, Visual Studio 2005 requires a CPU with a 2.0 GHz processor, 512MB of RAM, and 8GB of free hard drive space. Even though Microsoft states that 512MB is enough, my practical experience has shown that 1GB is the minimum required for any acceptable level of performance. Microsoft's recommended hardware requirements include a 2.6 GHz processor, 1GB of RAM, and 20GB of free hard drive space.

Other Tools

The minimum specification for a test agent (also known as a test rig) is a single processor running at 600 GHz, 1GB of hard drive space, and at the very least 256MB of RAM. The maximum requirement (assuming you have over 250 users) is a CPU with dual processors running at 2.8 GHz, 8GB of free hard drive space, and at least 2GB of memory.

The test controller has slightly different specifications. The minimum specification (roughly 5 to 20 users) is a CPU with a single processor running at 600 MHz, 1GB of hard drive space, and 256MB of RAM. For larger projects (and a greater number of users) you should spec out a machine with a single processor running at 2.6 GHz, 48GB of hard drive space, and, at the very minimum, 1GB of RAM.

64-Bit Support

Team Foundation Server (the application tier) will not run on 64-bit systems; you must install it on a 32-bit machine. As a result, you obviously can't set up a single-server install (application tier and data tier) on a single 64-bit system. The data tier (SQL Server 2005) will run on a 64-bit machine, you just need to install SQL Server 2005 64-Bit Edition. Team Foundation Build and Visual Studio 2005 will run on 64-bit machines but only in WOW64 compatibility mode.

Software Requirements

To successfully deploy Team Foundation Server, you must take into account the software requirements of each of the components. Here is a drill down of all the required software and additional practical considerations.

Required Service Packs and Software Components

At the time of writing, Team Foundation Server itself does not require any service packs. However, we have outlined the service packs for the components that Team Foundation Server depends on (for example, the operating system).

Team Foundation Server

Team Foundation Server requires Microsoft Windows Server 2003 with Service Pack 1 (SP1). It will not correctly function on any other current operating systems. To function correctly, the following software components must be installed: Windows SharePoint Services Service Pack 2 (SP2), Internet Information Server 6.0 (IIS, which is packaged with Windows Server 2003), and SQL Server 2005 Standard Edition and above. The key for a successful deployment is installing the components in the right order and correctly configured. The most accurate and complete source of information is the Visual Studio Team Foundation Installation Guide (`TFSInstall.chm`). The guide can be found on the CD or DVD installation media for Team Foundation Server.

SQL Server 2005

SQL Server 2005 has more flexibility than Team Foundation Server with regards to the operating system (again, assuming that you install the data tier [DT] on a separate machine than the application tier [AT]). SQL Server 2005 requires at least Windows 2000 Advanced or Datacenter Edition with Service Pack 4 or higher. It is fully supported on Windows 2003 Enterprise or Datacenter Edition, or Windows Small Business Server 2003 Service Pack 1 Standard or Premium Edition.

SQL Server 2005 has 64-bit support. If you try to run SQL Server 2005 32-bit edition on a server that has an x64 bit processor, it will run in Windows On Windows (WOW64) compatibility mode. Otherwise, SQL Server 2005 will effectively run Windows 2003 Service Pack 64-bit X64 Standard, Enterprise, or Datacenter Edition.

Team Foundation Server requires at the very least SQL Server 2005 Standard Edition to run.

Team Foundation Build

Team Foundation Build can only be installed on Windows XP Professional with Service Pack 2 (SP2) and Windows Server 2003 with Service Pack 1 (SP1) Standard or Enterprise Edition.

Visual Studio 2005 and Team Explorer

Visual Studio 2005 will install on Windows 2000, Windows XP Home and Professional, and Windows Server 2003. Note that Visual Studio 2005 is not supported on the Windows 2000 Datacenter Server.

> **If you need to install Visual Studio 2005 on a pre-Service Pack 2 system, Aaron Stebner outlines the steps on his Web log at** `http://blogs.msdn.com/astebner/`**. This may be useful if your operations department hasn't yet deployed Service Pack 2 to all desktops or in test installations.**

All the editions of Visual Studio 2005 also require Internet Explorer 6.0 with Service Pack 1, Microsoft Office 2003 (or greater) with Service Pack 1, Microsoft Data Access Components (MDAC) Version 9.0, and the .NET Framework 2.0 (which is installed as part off the Visual Studio 2005 installation process).

Team Explorer has the same requirements as Visual Studio 2005. It can be used as a standalone tool (a stripped down version of Visual Studio 2005 without any code building functionality). If you have any of the Team Editions installed on your target machine, Team Explorer will install as a plug-in, providing access to Team Foundation Server. Team Explorer is available on the Team Foundation Server CD or DVD media. Figure 1-4 shows a screenshot of the standalone version of Team Explorer.

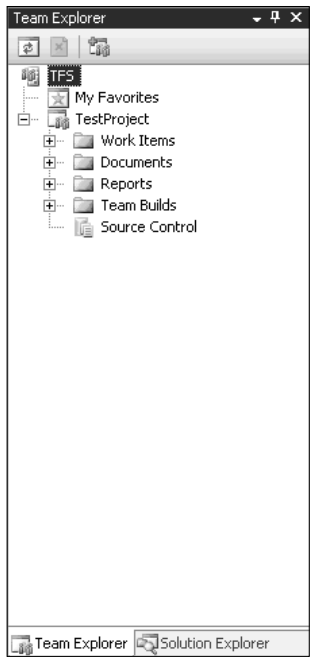

Figure 1-4

Other Tools

The test controller and test rigs are used to run load tests with a large number of simulated users on remote computers. The test controller requires Microsoft Windows Server 2003 with Service Pack 1 (any version of Windows Server 2003 will do), Microsoft SQL Express Edition, and the .NET Framework 2.0. The test rig must be installed on Microsoft Windows Server 2003 with Service Pack 1, Windows XP Professional with Service Pack 2, or Windows 2000 with Service Pack 4. The test rig also requires Microsoft SQL Express Edition and the .NET Framework 2.0.

Unsupported Software

If you want to integrate Microsoft Excel or Microsoft Project with Team Foundation Server, you must install Office 2003 before installing Team Explorer. Earlier versions of Office aren't supported.

If you are running any systems with Windows NT or Windows 98, you will have to upgrade them to Windows XP to connect them to Team Foundation Server.

Migrating and Integrating Your Existing Tools and Assets

When you look at how you can work within Team System, you have to consider whether it makes more sense to migrate or integrate your existing tools into Team System. Migration makes a lot of sense if you want to rid yourself of expensive third-party licensing agreements. For example, Team System provides the ability to do large-scale load tests and obviates the necessity of paying thousands of dollars to support and license a third-party tool.

Integration is a smart option if you have invested a lot of money on existing tools and you wish to leverage that investment. Integration is a little trickier because it often entails extra configuration and programming steps to make both systems "talk" to and integrate with each other. Let's take NUnit as an example of a tool that is very popular and does not have a great integration story with Team Foundation Server. Sure, you can migrate your NUnit code over to the Unit Test Framework within Team System using the migration tool. But if you want true integration with NUnit, you would have to create tools using Team Foundation Server's extensibility hook, which would perform the translation back and forth and allow Team Foundation Server to "understand" the code in build verification tests, during checkins, and so forth.

In this section, we will look at the core features of Team Foundation Server and discuss the challenges and resources available to help you migrate and integrate your existing solutions to Team System.

> **When making a decision about integration or migration, it is important to discuss the implications with a knowledgeable systems integrator or consultant. While a decision may seem straightforward, it may have cost, technical, and architectural implications. It is important that whoever you decide to work with has a solid background on Team System and your existing tools.**

Version Control

Most of the core components of Team Foundation Server are designed to work with the .NET Framework 2.0. This makes Team System a very appealing candidate if you are planning to develop an application from scratch using Visual Studio 2005.

But what should you do if your application was built in VB6 or the .NET Framework 1.1? Here is a matrix covering the common scenarios you may encounter:

Scenario	Action
You have VB 6.0 or VC++ 6.0 and would like to import into Team Foundation Version Control.	You can choose to run Visual Studio 6.0 side-by-side with Visual Studio 2005. You can then import Visual Studio 6.0 projects into Team Foundation Version Control using the command-line tool or the MSSCCI client. Another option is to convert your code from VB or VC++ 6.0 to the .NET Framework 2.0. For example, if you attempt to open a VB 6.0 project in Visual Studio 2005, an upgrade wizard appears. There is also a lot of migration documentation on the MSDN Web site at `http://msdn2.microsoft.com/en-us/library/`.
You would like to import and build .NET 1.0 or 1.1 code in Team System.	In most cases, attempting to open and save a .NET 1.0 or 1.1 solution in Visual Studio 2005 will result in solution that will no longer open in Visual Studio 2002 or 2003. The reason for this is that Visual Studio 2005 supports the .NET Framework 2.0 only. So, what are your options? You can install Visual Studio 2002/2003 and Visual Studio 2005 side-by-side. By opening your older solutions in the older versions of Visual Studio, you will avoid any migration issues. However, until a Team Foundation Source Control plug-in is developed for Visual Studio 2003, the only way you will be able to import your code in Team Foundation Version Control is by using the version-control command-line tool. Another option is upgrading your application to the .NET Framework 2.0. This is facilitated by the built-in upgrade wizard. The downside is that you will have to deal with migration issues; however, the upside is that you will be able to import and export your source code using Team Explorer.

Scenario	Action
	There are two tools available to build .NET 1.1 within Team Foundation Build. The first tool is called MSBuild Toolkit and is available at `http://downloads.interscapeusa.com/MSBuildToolkit_v2_RC.msi`. The second tool is called MSBee — MSBuild Everett Environment. It is being developed by Microsoft. More information is available at `http://blogs.msdn.com/clichten/`.

For ASP.NET 1.0 or 1.1 code, there is a migration wizard to help you migrate your application to ASP.NET 2.0. The easiest way to migrate 1.0 Web services is to recreate them in 2.0 and copy and paste your source code.

> **Keep in mind that launching ASP.NET 2.0 Web sites will automatically launch the local Cassini Web Developer server by default. You should configure the virtual directory in your IIS server to host your application and set Visual Studio to launch without Cassini to get a similar experience as earlier ASP.NET applications. ASP 3.0 applications are compatible but will not compile.**

What if you have code designed for a different platform, for example, Java code? Team Foundation Version Control is versatile and can store different file types including Java projects. Obviously, code from other platforms will not integrate as nicely with the testing tools and the build engine. However, you could launch builds on other platforms using custom build tasks, and integrate external testing tools using either build tasks or generic tests.

Teamprise (`teamprise.com`), one of Microsoft's VSIP customers, has developed a plug-in to allow Eclipse to integrate with Team Foundation Version Control and Team Foundation's work item tracking. Before you decide to integrate code from other platform, however, you must first configure the file types by clicking on Team➪Team Foundation Server Settings➪Source Control File Types. The window shown in Figure 1-5 displays.

Simply add the required file type and you are ready to go. Be sure to select Disabled file merging if your files are binaries or nonsource code files.

> **You will have different levels of support for languages in Team System depending on the language. For example, there is great support for Java in the IDE, whereas other languages are "less" supported. Of course, you can integrate anything with Team System. All that is required is ingenuity and effort.**

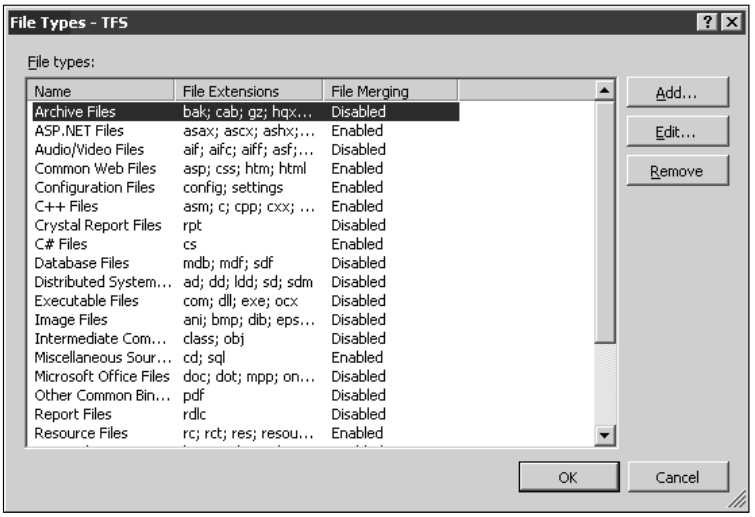

Figure 1-5

When looking at source control, the main question you have to ask yourself is if it makes more sense to maintain the current version or source-control system, or migrate your source projects to Team Foundation Version Control. If your company has invested tens and hundreds of thousands on a source-control system, it may not be in your best interest from a business perspective to migrate all your code. However, if you are planning new development using Visual Studio 2005 and the .NET Framework 2.0 (or .NET Framework 3.0), starting a project within Team System will be a good decision, as you will be able to leverage the deep integration between Visual Studio and Team Foundation Version Control.

> **One of the most important considerations in deciding whether to move your source code to Team System is your history. If the history isn't crucial, then you can check out all of your code and check it into Team Foundation Version Control. If it is important, then you may choose to migrate your code (using the Visual SourceSafe migration tool, a third-party tool like ComponentWare's Converter, or your own custom migration tool using the Team Foundation Version Control API), maintain the code in the existing repository, or even just keep an archive of your "old" code and begin new development on the new platform.**

Figure 1-6 contains a flowchart that will help you decide what approach to take in the migration (or archiving) of your existing code.

One of the most popular migration paths is moving code from Visual SourceSafe to Team System. Visual SourceSafe is a good tool for a small development projects with a small number of developers. However, it does not scale well if your team grows beyond a dozen developers and testers. Team Foundation Version Control is designed to manage large teams of developers and provides first time concurrent check-ins and rollback capabilities.

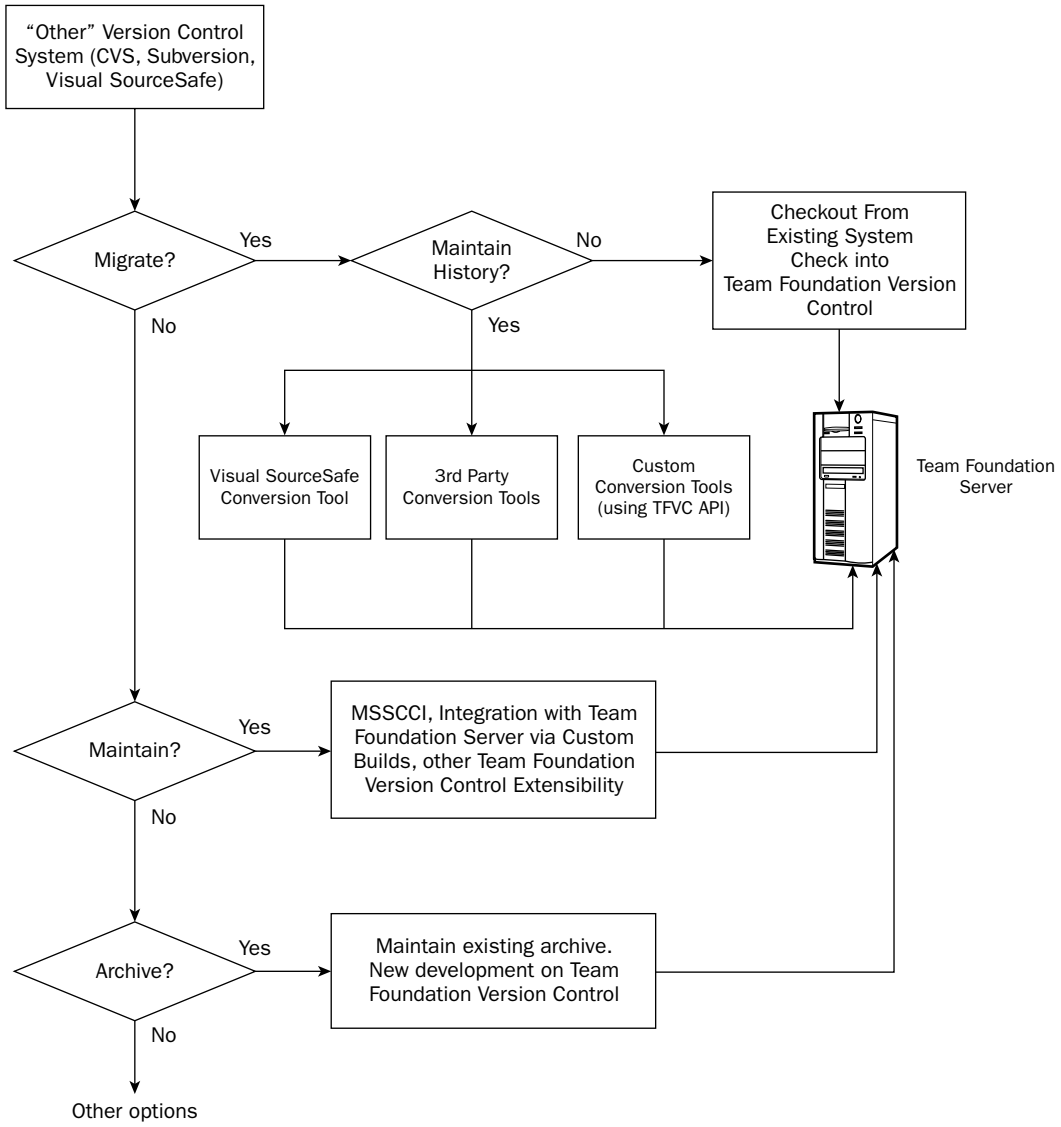

Figure 1-6

One of the biggest misconceptions about Team System's version control is that it is the new version of Visual SourceSafe. This is incorrect; Team System Version Control was written from scratch for Team System and uses SQL Server 2005 as a means of scaling to larger environments and infrastructures.

Microsoft has developed a migration tool called VSSConverter to allow you to migrate your Visual SourceSafe code to Team System. You can find out more details in this MSDN article at `http://msdn2.microsoft.com/en-us/library/ms253060.aspx` and in Chapter 12 in this book.

What if you are using other version control systems such as CVS or SourceGear? Team System does not support these systems out of the box, but nothing prevents you from leveraging these systems directly. If you choose another source control system, keep in mind you will not be able to use the build engine. (Team Foundation Build pulls the sources out of source control before building them.)

What if you want to integrate an existing source-control system with Team System? Luckily, there are a lot of projects in the works to help you. Team Foundation Server provides a wide variety of APIs to allow you to connect and transmit information from Team Foundation Server to your source-control system and vice-versa.

> **Anything that has to do with extensibility is covered in Chapter 9, and can also be found within the Visual Studio 2005 Software Development Kit, which you can download from** `http://affiliate.vsipmembers.com/`**.**

Work Item Tracking

Work item tracking is an important feature of Team Foundation Server for two reasons: First, it provides a way for your team members to track workflow and collaborate right from within Visual Studio. Second, it provides the important link to implementing your process. For example, if the process you are using has a predefined iterative flow, you can create a series of tasks or requirements that will support and enact the iterations and establish your process.

Work item tracking is a baked-in and highly integrated way of managing your work within Team System. But what if you have invested work or budget on other tools? As with any other feature of Team System, you are not forced to use the feature. You can selectively pick what you prefer to use.

If you wish to move over your ClearQuest workflow to the Team System work item database, Microsoft provides a ClearQuest migration tool called WIConverter to help you out. You can learn more about the migration tool at `http://msdn2.microsoft.com/en-us/library/ms253046(en-US,VS.80).aspx`.

You may also be tracking workflow using Microsoft Office Project Server 2003. Fortunately, there are efforts under way to integrate both Team System and Project Server using a special connector. You can learn more about the connector at `gotdotnet.com/workspaces/workspace.aspx?id=b9f69ea5-ace1-4a21-846f-6222a507cc9c`.

The chart in Figure 1-7 shows the different decisions you can make in regards to migrating or integrating your existing work items within Team Foundation Server.

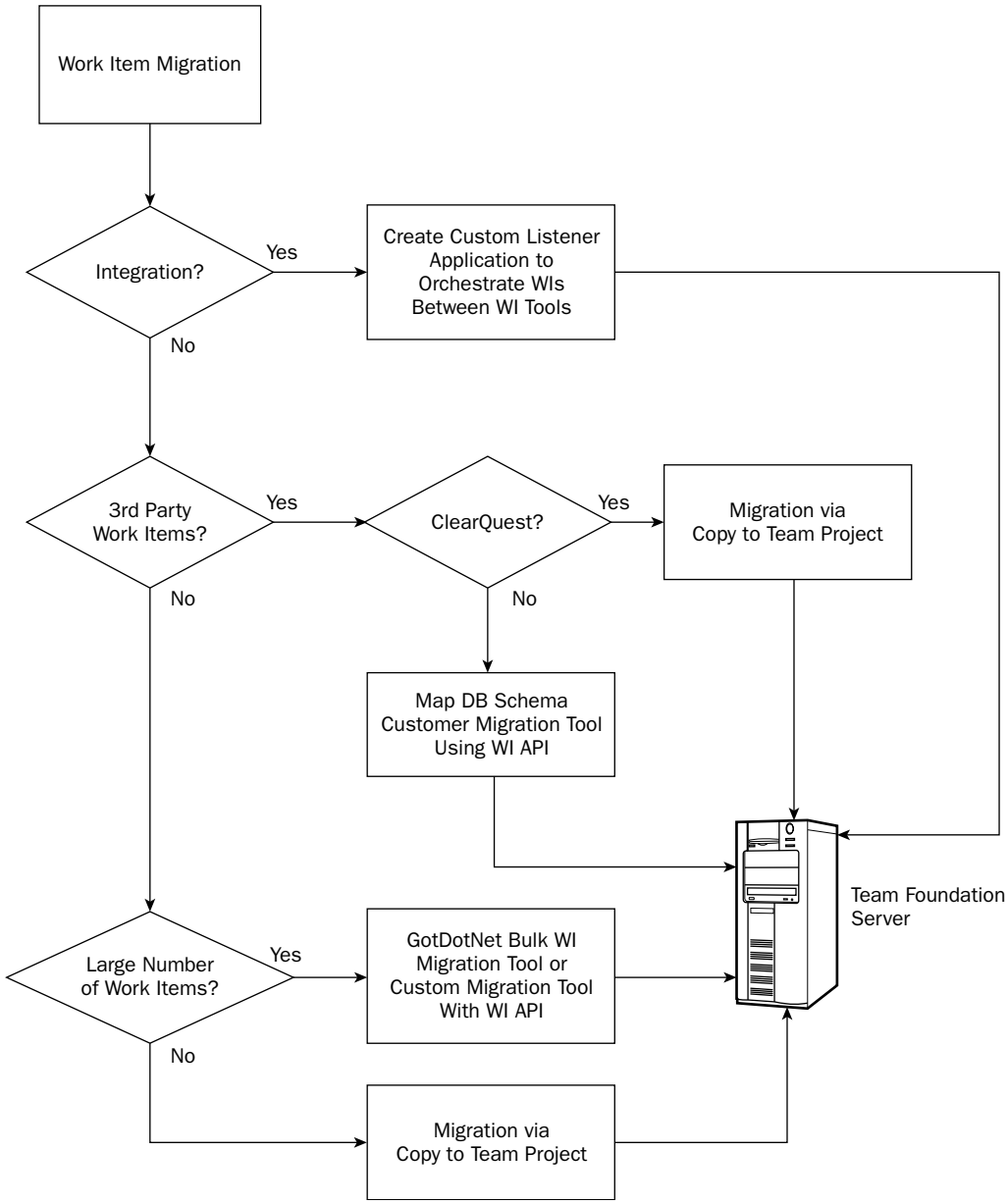
Work Item Migration
Integration?
Yes
No
Create Custom Listener Application to Orchestrate WIs Between WI Tools
3rd Party Work Items?
Yes
No
ClearQuest?
Yes
No
Migration via Copy to Team Project
Map DB Schema Customer Migration Tool Using WI API
Team Foundation Server
Large Number of Work Items?
Yes
No
GotDotNet Bulk WI Migration Tool or Custom Migration Tool With WI API
Migration via Copy to Team Project

Figure 1-7

Reporting

Team System does an amazing thing with reporting: It automatically aggregates project data such as build quality stats, bug counts, work item completion statistics and much more. In the past, you had to manually aggregate all your data by hand and import it into Excel to generate a report. Other tools like Crystal Reports provide the ability to generate graphs and other reports from raw data. One of the disadvantages with third-party tools is that they can be expensive.

Team System leverages SQL Server Reporting Services. Visual Studio 2005 provides a Report Designer to help you create new kinds of reports that can be viewed on the report site. The main idea here is that if you can bring in external data into SQL Server 2005, you can then mine and view the data in an integrated way within Team System.

You can create custom reports using the Report Designer or Excel. You can also create custom work items that contain reportable fields that can be integrated within a custom report. To learn how to create your own reports, refer to Chapter 16.

Build Server

One of the scenarios you may encounter is that you may want to transfer your NAnt scripts (or other build scripts) to Team System. Unfortunately, there is no established tool to help you do this. However, MSDN Channel 9 has a task equivalency chart that shows you the differences and similarities between MSBuild and NAnt. You can view the chart at `http://channe19.msdn.com/wiki/default.aspx/MSBuild.EquivalentTasks`.

In migrating or converting a script from one platform to another, the trick is sometimes just to find the right custom task to do the job. GotDotNet has a great repository of prebuilt custom build tasks at `http://www.gotdotnet.com/Community/UserSamples/Details.aspx?SampleGuid=2cb20e79-d706-4706-9ea0-26188257ee7d`. The GotDotNet project is called ".NET SDC Solution Build & Deployment Process & Tools." You can also download a number of useful open-source build tasks from Tigris. The download link is `http://msbuildtasks.tigris.org/`.

Figure 1-8 shows the decision path to determine if you want to integrate your existing tools (such as CruiseControl) with Team Foundation Server. If you want to migrate your tasks, then there are equivalency charts and custom tasks that will facilitate the process.

Testing Tools

Team System incorporates many testing tools including dynamic analysis, code analysis, manual testing, load testing, unit testing, ordered testing, Web testing, and performance testing.

NUnit Converter is a migration tool created by James Newkirk to port NUnit tests to Team System. Note that unit testing is fully integrated into the toolset as the Unit Test Framework. Note that this framework is available only in the Team Edition for Software Developers, Team Edition for Software Testers, and Team Suite versions of Visual Studio 2005. The migration tool is available on the NUnit Add-ons workspace on GotDotNet at `http://www.gotdotnet.com`.

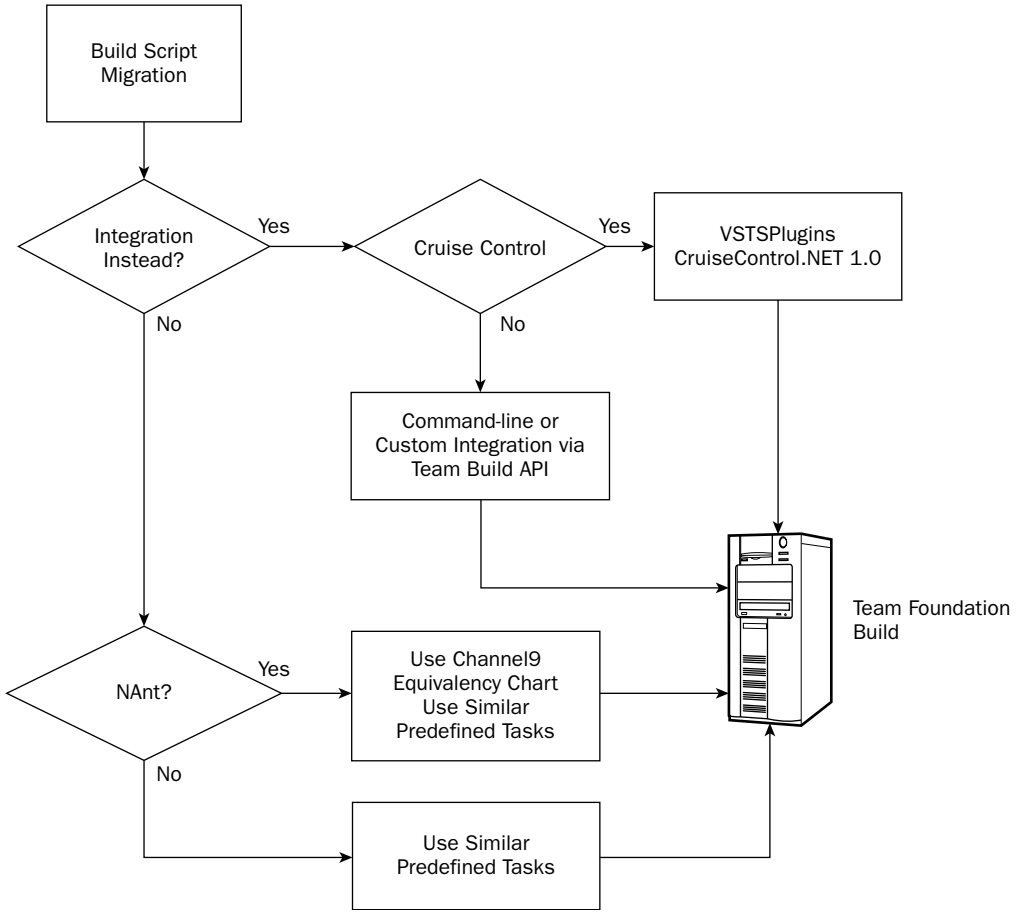

Figure 1-8

The converter tool is integrated in Visual Studio 2005 and requires the installation of the Guidance Automation Extensions to function.

In terms of load testing, Team System comes with an integrated tool. From a practical standpoint, one of the core advantages of the load-testing tool is that it is available to the entire software development team and is a great deal less expensive (from a licensing perspective) than other third-party tools. Another advantage is that the Team System load tester can support a great number of concurrent users.

Unfortunately, at the time of writing there aren't any migration tools to help you migrate Mercury Loadrunner tests to Team System. However, it is possible to integrate the Loadrunner tool using a generic test. The generic test allows you to run external executables (with parameters) and import results into Team System's testing framework.

Licensing Models

Cost is always one of the key questions that keep coming up when considering Team System. In this section, we provide guidance and a simplified view to help you understand the licensing model.

> **The best source of information for licensing is a whitepaper published on MSDN. To access the whitepaper, refer to** `http://go.microsoft.com/fwlink/?LinkId=55933`**.**

Here are some practical guidelines to consider:

- If your team members perform specific roles without overlap, consider obtaining a specific Team Edition version of Visual Studio for each team member. For example, developers would obtain the Team Edition for Software Developers. If there is overlap between roles, you should perhaps consider obtaining the Team Suite.

> **If you plan to use all the testing feature integration with Team Foundation Build, you should obtain a license for Team Edition for Software Developers and Team Edition for Software Testers (or Team Suite). The combination of these two products provides the necessary framework to run any of the tests commonly found in Team System. Note that both these products must be installed on the build server.**

- Team Foundation Server requires a license, and every computer accessing the server using Team Explorer (or any other client) requires a client access license (CAL). In some instances involving a remote "non-employee," a connector can be purchased to allow them access to the server.
- Each instance of Team Foundation Server Proxy requires a Team Foundation Server license. A license is also required for "warm" failover instances of Team Foundation Server.

Where to Get Team System

There are four primary ways of obtaining Team System:

- **Retail** — The components of Team System are available from retail outlets (such as Amazon.com). If you are buying the product retail, keep in mind that (a) you are paying full price, and (b) you are not going to benefit from Software Assurance (SA). Software Assurance guarantees that if Microsoft releases any Team System products off band (such as Team Edition for Database Professionals), you will get a license for it at no extra cost. It also applies if something like a "Team Foundation Server R2" gets released; you will be entitled to a copy.
- **Reseller** — Resellers such as SoftChoice provide good value because they have licensing experts that can find and tailor a licensing package to your situation and environment.
- **MSDN Subscriber Downloads** — If you get a combination of Partner, Academic or GSI Programs and own a MSDN subscription, you may also be able to benefit from substantial savings by upgrading to Team System. You can view more information about each option at the following link: `http://msdn.microsoft.com/vstudio/howtobuy/`.

Summary

This chapter introduced Team Foundation Server including its underlying architecture and the components of the product. Next, you learned how to compile your project data, including assessing security, client, and server requirements.

Later in the chapter, you learned how to plan a deployment and looked at the practical hardware considerations for the smooth operation of the product. You then looked at migration and integration scenarios for each one of the Team Foundation Server components. Finally, you learned about the Team System licensing model and a little about the costs involved.

In the next chapter, you will learn in detail about advanced installation topics and drill down on scenarios to put all of your planning into action.