Join the discussion @ p2p.wrox.com

Wrox Programmer to Programmer™

Professional Application Lifecycle Management with Visual Studio® 2013

Mickey Gousset, Martin Hinshelwood, Brian A. Randell, Brian Keller, Martin Woodward

Chapter 1
Introduction to Application Lifecycle Management with Visual Studio 2013

What's in this chapter?

- Defining application lifecycle management
- Learning about the Visual Studio 2013 product family
- Seeing ALM in action using Visual Studio Ultimate 2013

In June of 1999, Microsoft started to re-evaluate how Visual Studio was being used as part of the software development process. Microsoft was continuing to serve the needs of an individual programmer through the highly productive “code-focused rapid-application-development” features of Visual Studio, but wasn't doing much to help programmers work together as a team. And what about software architects—how should they be working with the programming team? And what about testers and project managers?

Many teams had begun to set up their own solutions using a mixture of third-party, in-house, and vendor-provided tools to address such challenges as version control, bug tracking, and team communications. But this mishmash of tools can be tricky to set up and maintain, and even more difficult to integrate and report across. Microsoft sought to address this challenge by providing an integrated set of tools designed to address the needs of the entire software development team. Thus, Visual Studio Team System was born, and was first released with the Visual Studio 2005 product line.

At the heart of Team System, *Team Foundation Server* was created to provide a hub for all members of the development team to collaborate. Team Foundation Server is uniquely positioned from its predecessors across the industry by being the first tool of its kind built from the ground up to provide an integrated solution for many capabilities that had historically been offered as standalone tools. Team Foundation Server provides a unified solution for storing source code (along with a history of changes), work item tracking (which can include bugs, requirements, and so on), and automated builds. By providing a single solution with all of these capabilities, Microsoft delivered the ability to link all these artifacts for end-to-end traceability, reporting, process enforcement, and project management.

Team System also included “client” functionality, which surfaced in the various editions of Visual Studio development tools. Visual Studio seamlessly integrated with Team Foundation Server, but much of this tooling could also be used independently or with third-party source control solutions. Visual Studio Team System also introduced role-specific tooling that lived outside of the core Visual Studio development environment by recognizing that team members such as project managers are oftentimes more comfortable using tools such as Excel or Project, both of which could be used to manage and track work that lived in Team Foundation Server.

Team System was built from a foundation of tools and technologies that Microsoft had been using internally for many years to build some of the most complex software projects ever undertaken. Team System appealed not only to programmers, but to all members of the development team—architects, application developers, database developers, and project managers.

Three years later, Visual Studio Team System 2008 evolved from the previous version to include even more tools and functionality for all members of the project team to use. Two years after that, Visual Studio 2010 added even more functionality, including an entirely new set of tools for generalist testers (also referred to as manual testers), bringing a new audience of prospective users into the same set of tooling used by the rest of the team.

Application Lifecycle Management

Along with the release of Visual Studio 2010, Microsoft also stopped using the sub-brand “Team System” to describe these capabilities. Instead, Microsoft started referring to these tools as the *application lifecycle management* (also referred to as ALM) capabilities of Visual Studio. Application lifecycle management is a term that has gained momentum in the development industry to describe the way an application is managed from its conception, through its creation and deployment, to its eventual retirement.

It is important to note that application lifecycle management is a more comprehensive concept than its popular predecessor, *software development lifecycle* (SDLC). SDLC is primarily focused on the core coding activities that comprise the creation of an application's life, beginning with a requirement for an application and ending when that application is built and delivered. Application lifecycle management recognizes that requirements aren't simply born out of thin air. They evolve based on business needs, or ideas for new opportunities, and stakeholders who are considered external to the development team may still play a role during the development of an application in helping to refine requirements and provide feedback on implementations. Application lifecycle management also recognizes that a development team's job isn't done the moment they hand off a “finished” application. The development team will likely be called upon to help troubleshoot the application when things go wrong in the deployed environment, or to create subsequent versions of the application based on feedback from users or analytics from the operations team. Visual Studio itself has matured over time to grow from being a tool targeted squarely at programmers during the software development lifecycle to becoming a true solution for end-to-end application lifecycle management.

Visual Studio 2013 Product Lineup

Table 1.1 outlines the product lineup for Visual Studio 2013.

Table 1.1 Visual Studio 2013 Product Lineup

Product Name	**Description**
Microsoft Visual Studio Ultimate 2013 with MSDN	The comprehensive suite of application lifecycle management tools for software teams to help ensure quality results from design to deployment.
Microsoft Visual Studio Premium 2013 with MSDN	A complete toolset to help developers deliver scalable, high-quality applications.
Microsoft Visual Studio Professional 2013 with MSDN	The essential tool for basic development tasks to assist developers in implementing their ideas easily.
Microsoft Visual Studio Test Professional 2013 with MSDN	The primary tool for manual and generalist testers who need to define and manage test cases, execute test runs, and file bugs.
Microsoft Visual Studio Express 2013 for Web	A free version of Visual Studio 2013 that provides the core tools for creating web applications and services.

Microsoft Visual Studio Express 2013 for Windows	A free version of Visual Studio 2013 that provides the core tools for creating Windows Store apps.
Microsoft Visual Studio Express 2013 for Windows Desktop	A free version of Visual Studio 2013 that enables the creation of desktop applications in C#, Visual Basic, and C++.
Microsoft Visual Studio Team Foundation Server 2013	The server component for team development, version control, work item tracking, build automation, project management, lab management, and reporting.
Microsoft Visual Studio Team Foundation Server Express 2013	A free edition of Team Foundation Server that provides most of the same capabilities (including version control, work item tracking, and build automation), with some limitations, for a team of up to five users.

Visual Studio Premium contains all the functionality of Visual Studio Professional, and Visual Studio Ultimate contains all the functionality of Visual Studio Premium. Visual Studio Premium and Ultimate also include all of the functionality available in Visual Studio Test Professional.

There are a few additional standalone tools and technologies that comprise the Visual Studio 2013 family that are not listed. For example, in Chapter 10 you learn about the new Microsoft Feedback Client, which stakeholders use to provide rich feedback about an application that is stored in Team Foundation Server. In Chapter 3, you learn about Team Explorer Everywhere, which Eclipse developers use to work with Team Foundation Server. You learn about these additional tools throughout this book, but Table 1.1 showcases the primary products that Microsoft markets as part of the Visual Studio 2013 product family.

For a detailed breakdown of the functionality available in each product, a comparison chart is available at www.visualstudio.com.

NOTE

Software licensing is potentially a complex topic. It is important to ensure that the members of your team are adequately licensed to use Visual Studio and the related technologies that make up your development and testing environments. The Visual Studio Licensing whitepaper attempts to synthesize all of the licensing requirements for Visual Studio, Team Foundation Server, and related technologies into an easy-to-read format. You can find the latest version of the Visual Studio Licensing whitepaper at http://www.microsoft.com/visualstudio/licensing.

Application Lifecycle Management Challenges

Software developers share common challenges, regardless of the size of their teams. Businesses require a high degree of accountability—software must be developed in the least amount of time, and there is no room for failure.

Some of these challenges include the following:

- *Tool integration problems*—Most tools commonly used by software development teams come from third-party vendors. Integrating with those tools can pose a major challenge—in many cases, it requires duplicating or copying data into multiple systems. Each application has a learning curve, and transmitting information from one application to another (incompatible) application can be frustrating and time consuming.
- *Geographically distributed teams*—Many development and management tools don't scale for geographically distributed teams. Getting accurate reporting can be difficult, and there is often poor support for communication and collaborative tools. As a result, requirements and specifications might be captured incorrectly, causing delays and introducing errors. Global teams require solid design, process, and software configuration management to be integrated into one package. There aren't many software packages that can deliver all these features, and those that do exist tend to be incredibly expensive.
- *Segmentation of roles*—Specialization can be a huge problem on a team. Experts can assume that other departments are aware of information that doesn't end up in the status reports but that may greatly affect the project as a whole. Interdepartmental communication is a huge and prevalent challenge. These barriers exist between developers and testers, developers and stakeholders, developers and operations, and even developers and other developers.
- *Bad reporting*—This is an offshoot of the segmentation problem. In most cases, reports must be generated manually by each team, which results in a lack of

productivity. There aren't any effective tools that can aggregate all the data from multiple sources. As a result, the project lead lacks the essential data to make effective decisions.

- *Lack of process guidance*—Ad hoc programming styles simply don't scale. If you introduce an off-cycle change to the code, it can cascade into a serious problem requiring hours and days of work. Today's software has a high level of dependencies. Unfortunately, most tools don't incorporate or enforce process guidance. This can result in an impedance mismatch between tools and process.
- *Testing as a second-class citizen*—Shorter cycles and lack of testing can introduce code defects late in the process. Additionally, poor collaboration between developers and testers often results in wasted back-and-forth effort and software defects.
- *Communication problems*—Most companies use a variety of communication methods (such as email, instant messaging, memos, and sticky notes) to send information to team members. You can easily lose a piece of paper, or delete an important email message, if you are not careful. There aren't many centralized systems for managing team communications. Frequent and time-consuming status meetings are required to keep the team on track, and many manual processes are introduced (such as sending email, as well as cutting and pasting reports).

Companies introduce methodologies and practices to simplify and organize the software design process, but these methodologies must be balanced. The goal is to make the process predictable because, in a predictable environment, methodologies keep projects on track. It is often said that predictability reduces complexity. Conversely, methodologies add tasks to the process (such as generating reports). If your developers spend too much time doing these tasks, they'll be less productive, and your company won't be able to react competitively.

Enter Visual Studio 2013

There are three founding principles behind the application lifecycle management capabilities of Visual Studio 2013: *productivity*, *integration,* and *extensibility*.

Productivity is increased in the following ways:

- *Collaboration*—Team Foundation Server centralizes all team collaboration. Bugs, requirements, tasks, test cases, feedback, code reviews, source code, and builds are all managed via Team Foundation Server 2013. All reporting is also centralized, which makes it easy for project leads to track the overall progress of the project, regardless of where the metrics are coming from.
- *Manage complexity*—Software development projects are more complex than ever, and are getting more complex year by year. Team Foundation Server helps to manage this complexity by centrally tracking your entire software development process, ensuring that the entire team can see the state and workflow of the project at any given time.

Integration is improved in the following ways:

- *Integrated tools*—These facilitate communication between departments. More importantly, they remove information gaps. With the Visual Studio 2013 family of products, integration isn't an afterthought—it's a core design consideration for the toolset.

- *Role-specific tools*—Instead of asking every member of an extended development team to conform to using the same tool, such as Visual Studio, Microsoft recognizes that many members of a team already have a preferred tool that they use every day. Correspondingly, Microsoft has integrated into those tools directly to provide comfortable interfaces back to Team Foundation Server —whether it's Visual Studio, Eclipse, Excel, Project, Project Server, or simply a web browser.
- *Visibility*—Visual Studio and Team Foundation Server increase the visibility of a project. Project leads can easily view metrics related to the project and can proactively address problems by identifying patterns and trends.

Extensibility is provided in the following ways:

- *Team Foundation Core Services API*—Most of the platform is exposed to the developer, providing many opportunities for extensibility and the creation of custom tools that integrate with Team Foundation Server.
- *IDE*—The Visual Studio integrated development environment (IDE) itself is extensible, allowing third parties and end users to add everything from additional tool capabilities to new language compilers to the development environment.

Application Lifecycle Management in Action

To best demonstrate how Visual Studio 2013 can help in the process of application lifecycle management, let's run through a typical scenario with a fictional software development company called eMockSoft. eMockSoft has recently signed a partnership with a distributor to release its catalog of products. The distributor has requested a secure website to manage inventory and pricing information for internal and external partner organizations.

Let's look at the scenario as it applies to application lifecycle management and the Visual Studio 2013 tools.

Requirements

The business analyst meets with the project sponsor and other stakeholders to obtain requirements for the project. During this discussion, the business analyst and an application designer use the PowerPoint Storyboarding capabilities of Visual Studio 2013 to build a storyboard that visually models the application they believe their stakeholders are asking for. They share this storyboard with the stakeholders to review the proposed user interface, workflows, and transitions. The stakeholders provide valuable feedback that helps to refine the design, even before a single line of code is written.

The storyboard then becomes the basis of new requirements that inform the development team about what the project sponsor expects the software to deliver. The project manager uses the new web-based Agile planning tools to store these requirements in Team Foundation Server. She then works with the development team to decompose these requirements into tasks that the team will implement on an iterative basis. She also uses Microsoft Project to create a more detailed project schedule based on this work by importing work items.

The infrastructure architect can now begin the system design.

System Design and Modeling

Based on the client specifications, the infrastructure architect can use the UML tools in Visual Studio 2013 to define the architecture for the website. These designs help to inform the programming team about what to implement. As the architecture evolves, the infrastructure architect will use the dependency graph generation tools to analyze the application's architecture and propose architectural changes that can improve code maintainability and quality.

Code Generation

The developer receives work assignments and reviews the UML diagrams that were designed by the architect. The developer writes the necessary code, and does some preliminary testing, using the static code analysis and unit testing tools built into Visual Studio. Throughout the day, the developer checks the code and tests into Team Foundation Server 2013. As work is completed, the developer uses the new web-based task board provided with Team Foundation Server to track the progress of his work and keep the rest of the team updated about his status.

When necessary, the developer uses the built-in code review tooling to invite peer developers to view and comment on the code he is writing. This entire conversation is preserved within Team Foundation Server, making it possible to later conduct audits to discover why certain decisions were made about implementation choices.

Testing

The tester checks the progress of the development team by monitoring the nightly builds and automated tests. Using the lab management capabilities of Team Foundation Server 2013, each nightly build triggers the automatic creation of a virtual environment that is ready each morning for the tester to use. The tester uses Visual Studio Test Professional to author, manage, and execute a suite of manual test cases each day to surface potential bugs for the development team. The tester files bugs in Team Foundation Server that are assigned to the development team to fix.

All bug reports are stored in Team Foundation Server, and provide team members and project stakeholders with full visibility into the progress of the project. The bugs automatically contain a rich set of information for the developer, including a video of the test case being run by the tester, screenshots, an event log from the time the test was being run, and a pointer to a snapshot of the virtual environment where it was uncovered. The developer uses all this information to quickly diagnose and fix the bug.

Feedback

When the development team has finished an initial version of the website, they decide to ask the original stakeholders to review their progress to ensure that they are on the right track. The business analyst uses Team Foundation Server 2013 to request feedback from the appropriate stakeholders on the areas of the application that are ready for review. Each stakeholder receives an email along with an invitation to provide feedback. The stakeholders use the new Microsoft Feedback Client to capture their feedback as they are using the new application. The Feedback Client enables each stakeholder to capture a video recording of the application as they are using it, along with notes, screenshots, and audio annotations describing what they like and what they would like to see changed. This feedback is rich and timely, helping the development team refine their implementation before the iteration is finished.

Operations

After the application has been built and signed off by the testing team, it's ready to be deployed in the on-premises datacenter. eMockSoft uses System Center 2012 R2 to monitor the production servers, so the testing team is quickly alerted in the event that the application breaks or begins performing slowly. Using System Center Operations Manager, an operations engineer can choose to assign the issue to engineering, which automatically creates a bug in Team Foundation Server, including rich diagnostics from the Operations Manager's application performance monitoring capabilities. If a developer needs even more information to diagnose an issue, she can ask the operations team to capture an IntelliTrace file from the running application, which she can use to review everything that happened during the application's execution and look for clues about how to resolve such an issue. By using these types of tools, the company can ensure better collaboration between the development and operations team than had been achieved in the past.

Putting It into Context

This is a simple example that examines just a few of the ways in which Visual Studio 2013 can assist with application lifecycle management. Throughout this book, you discover other examples that can help your team become a more cohesive unit and ship better software.

Summary

In this chapter you learned about the overall Visual Studio 2013 product family and how it has been designed to help you address the entire application lifecycle management of your development projects. The rest of this book dives more deeply into how you can apply these tools to your own team.

Part I
Team Foundation Server

Chapter 2
Introduction to Team Foundation Server

What's in this chapter?

- Understanding Team Foundation Server
- Learning the core concepts central to Team Foundation Server
- Getting access to Team Foundation Server and connecting to it for the first time
- Learning about what's new in Team Foundation Server 2013
- Planning your Team Foundation Server adoption

Because Team Foundation Server is so fundamental to the Application Lifecycle Management offering from Microsoft, later chapters go into more depth about utilizing different aspects of the product, such as how to use it to plan your work, how to use version control when developing software, and how to use the build automation capabilities. In each case, the use of Team Foundation Server is explained within the context of the task you are doing — but before we can do that you need to know what Team Foundation Server is, what it provides, and how to get it.

Although a full treatment of Team Foundation Server is necessary in a book about Microsoft's Application Lifecycle Management solution, this book deliberately focuses on how to *use* Team Foundation Server to develop software and effectively organize your teams. Team Foundation

Server is highly customizable and extensible by an administrator. The book *Professional Team Foundation Server 2013* (Wrox, 2014) is targeted at administrators of Team Foundation Server and individuals who want to customize their instance heavily, although Chapter 7 of this book gives you a small taste of the customizations that are possible and provides a starting point to learn more.

What Is Team Foundation Server?

Developing software is difficult, a fact that is repeatedly proven by how many projects fail. Developing software is a creative endeavor, not a manufacturing process. Consequently, an essential factor in the success of any software development team is how well the members of the team communicate with each other and with the people who wanted the software developed in the first place.

Microsoft Visual Studio Team Foundation Server 2013 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 and planning
- Work item tracking (WIT)
- Version control
- Test case management
- Build automation
- Reporting
- Virtual lab management

Team Foundation Server is separate from Visual Studio. Logically, Team Foundation Server is made up of the following two tiers, which can be physically deployed across one or many machines, physical or virtual:

- *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.
- *Data tier* — The *data tier* is made up of two or more SQL Server databases containing the database logic of the Team Foundation Server application, along with the data for your Team Foundation Server instance. The data stored in the databases is used by Team Foundation Server's reporting functionality. All the data stored in Team Foundation Server is stored in these SQL Server databases, thus making it easier to back up.

Team Foundation Server was designed with extensibility in mind. There are comprehensive APIs in .NET and Java for integrating with Team Foundation Server, and a set of events that enables outside tools to integrate with Team Foundation Server as first-class citizens. The same APIs and event system are used by Microsoft itself in the construction of Team Foundation Server, as well as the client integrations into Visual Studio, Microsoft Office, and Eclipse.

Team Foundation Server has competitors, including other enterprise Application Lifecycle Management suites and purpose-specific solutions (such as source control, a build server, or a work tracking system). As discussed in Chapter 1, the main benefit of having all these capabilities in one product is the tight integration that Microsoft has been able to achieve between the tools that you use to develop software and the tools that you use to communicate with your team and your stakeholders.

Acquiring Team Foundation Server

Team Foundation Server is a server-side product that must be acquired, installed, and configured. There are several options available for purchasing access to a server for your team. To begin with, you should decide if you want to run the Team Foundation Server inside your own firewall or if you want to explore a hosted Team Foundation Server offering.

Hosted Team Foundation Server

The easiest way to acquire Team Foundation Server is to rent it from a provider and access it over the Internet. Trial options are available, which means you can get started with no cost, and there is no need to wait for hardware to be purchased. When it comes to hosted options, there are two main routes: hosting from Microsoft or hosting from a third-party provider.

However, hosting is not suitable for everyone. Some organizations have a legal obligation to keep the data that they would store inside Team Foundation Server inside the firewall; others may require the tight user identity integration provided by Team Foundation Server's Active Directory integration. Others are just not comfortable making their source code, work items, and build accessible from any machine over the Internet. For these types of organizations, a hosted solution probably isn't the answer.

Visual Studio Online

Microsoft makes available a massive cloud-hosted instance of Team Foundation Server, part of Visual Studio Online at http://www.visualstudio.com. This is new commercial branding for the service that is in a preview at http://tfspreview.com.

As of the end of 2013, this is now a full commercial service available for customers who want to purchase Team Foundation services for their team at a low, predictable cost. Depending upon how you license Visual Studio (if at all), you'll find a variety of plans, including free, that provide access to the rich features of Team Foundation Server, but in a purpose-built cloud implementation.

Visual Studio Online is hosted on Windows Azure and makes use of all the services provided by Microsoft's cloud operating system to ensure high availability, resiliency, and a full backup of your data. However, because the system is scaled to support the thousands of users who access it over the Internet—and because it is just the basic core Team Foundation services that are available—Visual Studio Online comes with some limitations compared with a full on-premises installation. For example, currently there is no integration with SharePoint for a project portal and document library. There are also limited reporting features currently available and restrictions to the amount of customization that you can do to the server instance.

However, Visual Studio Online provides all the version control, work item tracking, build automation, and project management capabilities of Team Foundation Server. Being available over the Internet makes it very easy to use when your team is distributed globally, and it is easy to get started on using the service. All you need to do is visit www.visualstudio.com, create an account, and your team can be up and running before you have finished reading this chapter. Access to Visual Studio Online is controlled by federated Internet-based credentials; at the time of writing you need to have a free Microsoft Account from to authenticate with the service.

Because Visual Studio Online is maintained by the Team Foundation Server team at Microsoft, it is always running the very latest version of the server software during their development process. Therefore, new features will show up on Visual Studio Online before they are made available in the standard retail installation of Team Foundation Server via an update or a new major release. For this reason, you may notice some differences between some of the screens displayed in the figures of this book and the appearance of Visual Studio Online at the time of reading.

NOTE

This cloud-hosted version of Team Foundation Server from Microsoft is the same in many ways as the Team Foundation Server available elsewhere and installed on your own servers, but there are some ways in which it operates differently (such as with regard to authentication). Throughout the rest of the book, we distinguish between the "hosted service" behavior and the regular (that is, "on-premises") behavior when it is important to do so — however, the majority of this book describes the behavior of Team Foundation Server in general, regardless of where it is installed.

Third Party–Hosted Team Foundation Server Providers

Many commercial companies can host your Team Foundation Server for you over the Internet for a small charge. They have the advantage that they have all the Team Foundation Server administrative knowledge in-house and have a great deal of experience running their servers for many customers. As these companies are dealing on a different scale than that of Microsoft's hosted service, they can often be much more flexible in the capabilities they provide (at a cost). Depending on the hosting provider, you may also be able to purchase SharePoint portal capabilities, along with a full reporting instance, and get the same capabilities as if you were running Team Foundation Server in-house without having to go through the up-front costs of acquiring the hardware to run Team Foundation Server or purchasing the software licenses in full, before use.

The version of Team Foundation Server used by the third-party hosted providers is exactly the same as the version you would get if you installed it on premises. The only difference is that Team Foundation Server is running in their data centers or private clouds and your team accesses it over the Internet. In this book, behavior categorized as *on-premises* refers to the behavior you would expect to see from your third party–hosted Team Foundation Server provider as opposed to the *hosted service* behavior provided by Microsoft's hosted offering (www.visualstudio.com).

NOTE

Microsoft provides a list of companies offering commercial hosting services for Team Foundation Server at http://aka.ms/tfshosting.

As mentioned previously, in some organizations, using a third party to host such important data as your company's source code is not acceptable, and some other companies may actually be required by law to keep such data within the bounds of the corporate firewall. In those instances an on-premises option is the only one available.

On-Premises Installation

The way that the vast majority of customers enjoy the features of Team Foundation Server is by locally installing a version of the software inside the firewall. Trial versions of Team Foundation Server are available for you to download and install locally so you can get up and running quickly. You can also download a prebuilt virtual machine from Microsoft with all the software necessary to help you evaluate the product.

NOTE

You can find the latest version of the virtual machine at http://aka.ms/VS11ALMVM or you can download the Express or Trial version of Team Foundation Server to install locally at http://aka.ms/tfs2013.

To purchase Team Foundation Server to run locally, you can acquire the software in retail or via a MSDN Subscription, a Volume Licensing purchase, or through a Microsoft Partnership agreement.

Also available, first introduced in the 2012 release, is a version called Team Foundation Server Express. This includes the core developer features — such as version control, work item tracking, and build automation — all of which is available free of charge for individuals and teams of up to five users. The Express edition comes with a few limitations, namely: no support for SharePoint integration, limited to five named users, supports only SQL Express (so no reporting and a maximum database size of 10GB), and no sprint/backlog planning or feedback management.

You can upgrade from a Trial or Express edition of Team Foundation Server to a full edition at any time without losing any data. In addition you can purchase additional

Client Access Licenses (CALs) if you require more than the five named users that come with the Express edition.

NOTE

For more information about installing or administrating a Team Foundation Server instance, see Professional Team Foundation Server 2013 by Steven St. Jean, Damian Brady, Ed Blankenship, Martin Woodward, and Grant Holliday (Wrox, 2014).

Team Foundation Server Core Concepts

Let's take a look at some of the core concepts that are critical to understanding Team Foundation Server. If you have been using previous versions of Team Foundation Server for a while (especially the previous Team Foundation Server 2012 release), then you might want to skip to the “What's New in Team Foundation Server 2013” section later in this chapter.

Figure 2.1 provides an overview of the Team Foundation Server components, which are explained in the following sections.

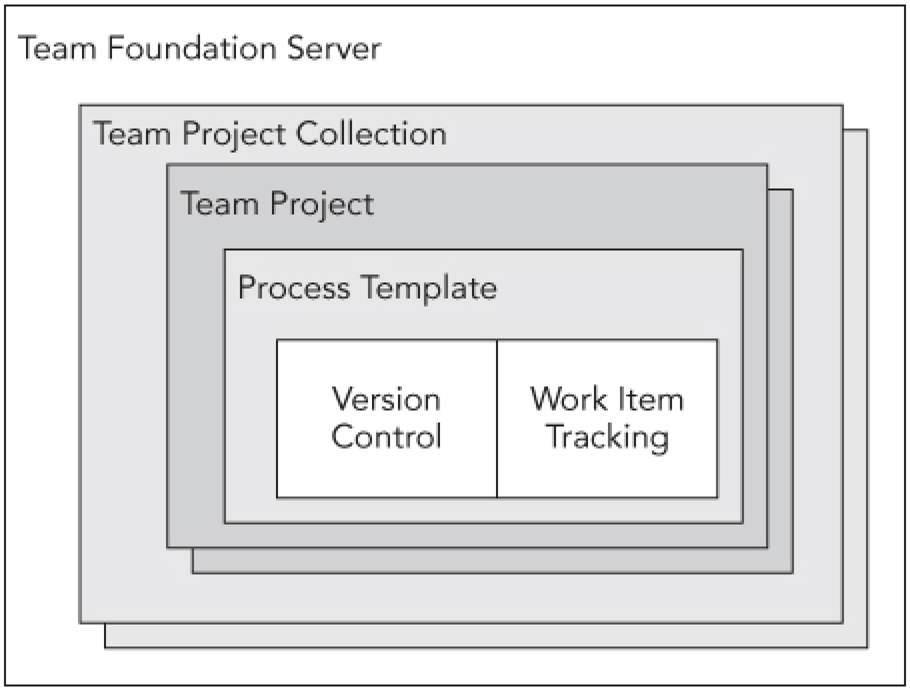

Figure 2.1

In addition to the components shown in Figure 2.1, understanding the concepts of teams and team builds is necessary for a complete understanding of Team Foundation Server. Those concepts are also covered in the following sections.

Team Foundation Server

A Team Foundation Server instance can be physically split into many different machines. The *application tier* refers to the running web application that is handling all requests for data from client machines running Visual Studio. The data in a Team Foundation Server instance is stored in a *data tier*, which is essentially a SQL Server installation being accessed by the application tier. Although the application tier and the data tier are logically separate, you can have both installed on a single physical machine. As the application tier is the level at which you access a Team Foundation Server instance, the application tier machine name is often referred to as simply the Team Foundation

Server. You refer to your Team Foundation Server by name or URL (that is, `tfsserver` or`http://tfsserver:8080/tfs)` when Team Foundation Server is installed in the default virtual directory in IIS on the default port. When talking to a Team Foundation Server hosted over the Internet, you most often use the full URL, such as https://proalm.visualstudio.com.

Team Foundation Server can scale to support a very large number of active users, depending on the hardware supporting it. Therefore, for most organizations, Team Foundation Server instances tend to be scoped according to who pays for the installation and operation of the instance, not by scaling limitations of the server.

Team Project Collection

The *team project collection* concept was first introduced in Team Foundation Server 2010. This is a container for team projects. Each server has one or many team project collections, and a project collection can have zero or more team projects.

The team project collection is the main level of isolation between instances on a server. In a hosted Team Foundation Server, the collection is what is provided as your *account*. Global security groups take effect at the project collection level. The identifiers for work items and for changesets in version control are all numbered with sequential IDs that are unique at the project collection level.

A team project collection has a one-to-one relationship with a database instance in SQL Server. Therefore, you can back up and restore at the project collection level. You can move project collections between Team Foundation Servers, and you can split the project collection to break up the distribution of team projects between the resulting collections. Using this process, you can move a team project into a new collection by cloning the existing project collection and then deleting the appropriate team projects from each of the cloned project collections.

Each Team Foundation Server instance has a default project collection, usually called `DefaultCollection`. As project collections were not introduced until the 2010 release, older clients that were created for Team Foundation Server 2008 will only be able to see this default collection.

Team Project

A *team project* is a collection of work items, code, tests, or builds that encompass all the separate tools that are used in the lifecycle of a software development project. A team project can contain any number of Visual Studio solutions or projects, or, indeed, projects from other development environments. A team project is usually a fairly long-running thing with multiple areas and iterations of work.

You need at least one team project to start working with Team Foundation Server. When the team project is created, the following are also created by default:

- Path in version control (if using Team Foundation Version Control)
- Default work item queries
- Default areas and iterations
- Default team

If you're using a Team Foundation Server instance that is also attached to a SharePoint and SQL Server Reporting Services instance, then the following are also created:

- Team project website

- Document library
- Stock reports

WARNING

It is not possible to rename a team project after it's been created. Also, the number of team projects in the team project collection has a performance effect on the system, so you do not want to have more than around 250 teams per project collection. Therefore, you want to think carefully before creating a new team project.

It is often useful to experiment with Team Foundation Server features in a sandboxed test instance of Team Foundation Server. Many people download the Team Foundation Server Trial virtual machine image from Microsoft for this purpose or get an account for a Microsoft-hosted Team Foundation Service instance at http://www.visualstudio.com, but some organizations have enterprise-wide test instances of Team Foundation Server for people to experiment in.

The granularity that you choose for your team project has important implications for how you structure your work and when you move from one team project to another.

Team projects are intended to represent the largest unit of work in your organization. For example, in Microsoft Developer Division, the whole of a Visual Studio release lives in a single team project with Team Foundation Server as an area of that project.

A team project has a single process template, and changes made to the process template of a running team project affect that team project only. The default reports and work