CHAPTER 1

Introduction to Visual Studio Team System

In the modern world of development, developers no longer work alone or in groups of three or four people in a set of side-by-side cubicles. Today's developers typically work in larger teams scattered across the globe. Developers have become a global commodity. Many companies in the United States perform some type of outsourcing in which they hire developers who work in India, China, Canada, Russia, or other parts of the United States. This presents a unique challenge to software project teams.

Development teams may include project managers, developers, architects, testers, support staff, and others. How do the team members communicate? What information should be shared, and whom should it be shared with? Should some people have access to some code but not other code? These questions apply not only to developers located in different parts of the world, but also to teams that work in the same building or the same city.

The number of issues that face development teams today is huge. The preceding questions cover only the communication of information. This list can be expanded to include (but not limited to) the following:

- What is the application architecture?
- What is our methodology and what are the deliverables?
- How is the application going to be deployed?
- How will the various components communicate with each other?
- What am I responsible for and when do I have to have this work done by?
- Has anyone tested this code yet? Did it pass the tests?
- What are the object dependencies?
- How are we doing change management?

The list of relevant questions grows very quickly. Up until now, there was no easy way to answer these questions except with regular status meetings, a large amount of e-mail, or a lot of expensive phone calls. The information is not always up-to-the-minute accurate, and it takes a lot of time to sift through all of it. These are some of the issues that Microsoft set out to solve with the introduction of Visual Studio Team System (VSTS).

What Is Visual Studio Team System?

VSTS is a suite of tools designed to allow members of a development team to communicate not only with one another, but also with stakeholders, in real time. It also contains a set of tools for developing architectural views of a system, generating code from certain views, testing code at all stages (during and after development), and integrating the development experience with project management.

At a high-level view, VSTS is divided into four areas: integration, architecture, development, and testing. Each of these areas contains tools that cater to a different facet of the development team. Some of the tools are available to all groups of users, and some are targeted at a specific group because they pertain to a responsibility associated with only one role.

But this is a superficial view of VSTS. It is also designed, from the ground up, to help an organization implement an effective development methodology, whether it is the Microsoft Solutions Framework (MSF), the Rational Unified Process (RUP), Extreme Programming (XP), or any of a dozen other types of methodologies. The purpose in implementing a structured methodology is the same as the goals of the rest of the VSTS suite of tools: to build better applications for a lower cost, both in the short term and the long term. This concept of integrating methodology into the development process is ingrained in all aspects of VSTS.

What Are the Benefits of Using Visual Studio Team System?

Who would benefit from using VSTS for their projects? In short, the answer is everyone. Specifically, it benefits project managers, architects, developers, testers, infrastructure architects, users, and stakeholders. Here's how:

- *Project managers* can get up-to-date information on which items on the project schedule are being worked and when they are completed through familiar tools like Microsoft Project and Excel.
- *System architects* can design an application as it applies to the network infrastructure and communicate that to the deployment and development team.
- *Infrastructure support* gets a solid understanding of the deployment needs of the application.
- *Technical architects* can design classes, relationships, and hierarchies that automatically generate skeleton code.
- *Developers* can look at the class diagrams to understand what is occurring. Any changes they make to the code will be reflected in the diagrams—no reverse-engineering of the model is necessary. Code can be effectively unit tested.
- *Testers* can use integrated testing tools, which allow for more thorough testing. Tests can also be run automatically via automated build tools.
- *Application stakeholders* can view reports on the progress of the application through Microsoft SharePoint Services.

As you can see, many individuals can benefit from the use of VSTS. These benefits translate directly in a higher return on investment because everything becomes easier and faster for everyone.

Aside from individuals who benefit from using VSTS, organizations and departments will also find tremendous advantages in using this tool. The first and most obvious benefit is that it promotes communication between team members, which is crucial to the success of a project. It allows for problems to be caught early and solved quickly before they become serious issues that affect the schedule. These problems can range from developers not completing work on time to bugs in the code.

VSTS also allows for the analysis of work across multiple projects. It becomes simple for organizations to track their project history and use that information to predict future project schedules. Projects can be reported on by category, developer, deliverable, milestone, and so on. You literally have the power of an online analytical processing (OLAP) database at your fingertips, filled with all of your project information down to the code level and bug-tracking level. To achieve this type of reporting, you've needed to use several different, costly systems. With VSTS, it is all rolled into one integrated system.

All of these benefits come down to one thing: a higher return on investment with one tool than you would get with combinations of tools. When you use one tool for each area of development—such as Borland JBuilder for development, CVS for source control, Rational ClearQuest for issue tracking, Cognos ReportNet for reporting, Ant for building, and JUnit for testing—it becomes exceedingly difficult to keep things simple. On the other hand, you have the following benefits with VSTS:

- VSTS allows all developers to use one tool with which they are familiar. It does not require a developer to learn how to use six different tools to perform the task.
- VSTS integrates all of the needed functionality, including a project management tool and reporting tool, directly into one interface—something that no other integrated development environment (IDE) can do in its out-of-the-box version.

But let's say that you have an in-house testing tool that you would rather use than the tool that comes with VSTS. Because VSTS is an extensible environment, integrating other tools into it requires a minimal amount of work (depending on what you want to integrate). Many tool vendors have been working with Microsoft to create integration points with their tools so that you can swap them with ones that come with VSTS. You are not locked into a wholly Microsoft solution.

All of these points lead to only one conclusion: there is no downside to using VSTS.

Visual Studio Team System Editions

VSTS comes in three different editions and a core component called Team Foundation. This section describes each of these (which correspond to the sections in this book), their tools, and their goals. While this is the out-of-the box functionality available with VSTS, as noted in the previous section, is also highly extensible. Figure 1-1 shows an overview of the VSTS suite.

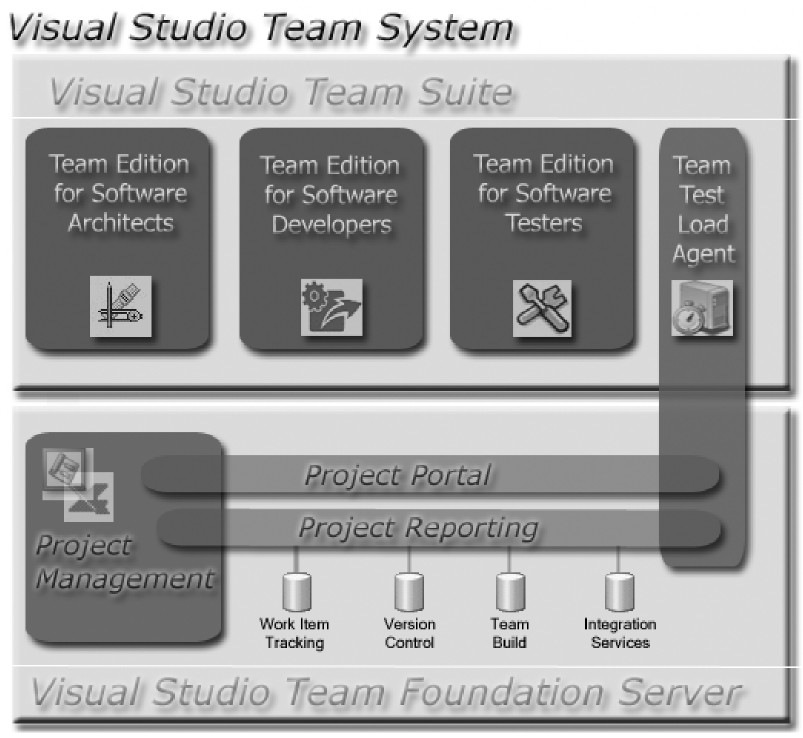

Visual Studio Team System
Visual Studio Team Suite
Team Edition for Software Architects
Team Edition for Software Developers
Team Edition for Software Testers
Team Test Load Agent
Project Portal
Project Reporting
Project Management
Work Item Tracking
Version Control
Team Build
Integration Services
Visual Studio Team Foundation Server

Figure 1-1. *Visual Studio Team System editions and main components*

Team Foundation

Team Foundation is the server-based component of VSTS. It is the *Team* in Team System. Without Team Foundation, all of the other components of VSTS are essentially stand-alone components that run on the client. Once Team Foundation becomes part of the picture, the various client pieces work together as a cohesive unit. Chapters 2 through 7 cover Team Foundation.

As we mentioned previously, VSTS is designed to provide a framework in which applications can be built. Many companies are working to improve their processes by using the Capability Maturity Model Integrated (CMMI) from Carnegie Mellon's Software Engineering Institute (SEI). With VSTS, Microsoft is releasing the only methodology recognized by SEI as being CMMI level 3 compliant. This methodology is the MSF for CMMI Process Improvement, Version 4.0. The template and instructions on how to use the methodology are all included with VTST. So, what is so significant about this? The U.S. Government uses CMMI levels in determining source selections for contract awards.

Team Foundation Version Control

Team Foundation contains a brand-new version control tool, which is designed for large-scale teams and is backed by SQL Server 2005. For developers, it will be a welcome addition to their toolbox and offer an easy alternative to Visual SourceSafe. Also, unlike Visual SourceSafe, Team

Foundation version control supports document (or code module) security. In addition to supporting developers, it also supports project managers and overall application reporting to the stakeholders.

The final touch for the new version control tool is that it allows you to implement policies to make sure that code meets certain requirements before it is checked in. This helps to ensure that the code goes through a consistent, repeatable process check before check-in.

Project Portal

Another key piece of Team Foundation is the Project Portal. This is a Windows SharePoint Services (WSS) site that serves as a central communication tool for the entire team. Stakeholders can go to this website to review the current status of various tasks on the project, view nightly build and test reports, and communicate with team members.

SharePoint also serves as a project documentation repository (with versioning). This is in contrast to how teams typically set up repositories today—in the file system.

Team Foundation Build

Team Foundation Build is a component that allows a server or workstation to become a build machine. Team Foundation Build automatically gets the latest version from the version control tool, compiles it, deploys it, and runs any automated tests (unit or web tests) against the build. The results of the compilation and testing are stored in the VSTS data warehouse.

Work Item Tracking

Work item tracking is another feature of Team Foundation. Work items can be created in Microsoft Project (directly from the work breakdown structure) or Excel and loaded into Team Foundation as a work item. These work items can be assigned to developers. When team members check their items into the version control, they can associate changes with specific work items. The status of these work items is then reflected on the Project Portal. Work item association can be enforced via policies as well.

Reporting

The final feature of Team Foundation is the reporting component, backed by the new version of SQL Server Reporting Services (SSRS). Out of the box, the reports cover areas such as the number of open bugs, closed bugs, and in-work bugs; work item status; build results; and other information.

As an added bonus, the SSRS features an end-user ad-hoc report builder, so users can create their own reports or customize existing reports. This, combined with the VSTS data warehouse, allows an organization to mine the data for trends in the overall software development life cycle.

Team Edition for Software Architects

Various types of architects may be assigned to a project, and each has different responsibilities. The one thing that all architects have in common is that they must communicate aspects of the architecture to stakeholders in various ways. To facilitate building and then communicating an architecture, Team Edition for Software Architects provides a set of designers, as well as other tools to ease the job of the architect. Chapters 8 through 10 cover the Team Edition for Software Architects.

Designers

The four VSTS designers are Application Designer, System Designer, Logical Datacenter Designer, and Deployment Designer. These designers are a core tenant of Microsoft's focus on Model Driven Architecture (MDA). However, VSTS moves models out of the cumbersome, documentation-only realm and into the practical realm.

The problem with modeling with other tools is that the models are abstract representations of the architecture. They do not mean anything from a tangible perspective. The designers in VSTS have a concrete implementation. When you create a model with VSTS, you also generate the configuration for that model, which is based on physical properties of the object to which you are deploying your application. This allows VSTS to check for inconsistencies in your architecture against the actual physical machines with which you will be working.

Domain-Specific Language

On top of this approach, VSTS leverages the concept of Domain-Specific Languages (DSL). DSL is the language in which the concrete implementation of hardware or software is written. This allows the end users of VSTS to build model objects against which specific implementations can be validated.

■ **Tip** Microsoft has released a set of tools specifically for creating domain-specific frameworks. These tools can be found at `http://lab.msdn.microsoft.com/teamsystem/workshop/dsltools/default.aspx`.

The language is a set of metadata that describes the physical implementation of a given configuration. Microsoft has introduced the System Definition Model (SDM) to provide a schema definition for distributed systems. Because these designers are built in concrete terms, they are easily understandable by their intended audience—data architects, infrastructure architects, or other application architects.

Visio

Team Edition for Software Architects also includes everyone's favorite modeling tool: Visio. This tool is available in all editions of VSTS, but will probably be most used by architects.

Visio for Visual Studio allows for the creation of Unified Modeling Language (UML) diagrams and provides the ability to model different views for different audiences of the application. Visio allows you to create those abstract, notional views, which are helpful in trying to figure out and pinpoint what the architecture will be and then communicate it to everyone else.

Team Edition for Software Developers

Team Edition for Software Developers provides tools that allow developers to quickly understand code structure, generate code from models, write unit tests, and analyze code for errors. The goal of these tools is to reduce the amount of time developers need to actually write code and to ensure that the code that is produced is of a higher quality. Chapters 11 through 14 cover the Team Edition for Software Developers.

Class Designer

To understand and generate code, VSTS provides the Class Designer. This is one of the tools available in all editions of VSTS because it is useful to a wide range of people. Architects can use the tool to create high-level class designs. Developers can generate skeleton classes, for which they can then fill in the details. Testers can use the Class Designer to understand the relationship between classes in order to help them analyze errors. We have included the Class Designer in the Team Edition for Software Developers section of the book because, for the most part, the primary audience is the developer.

The Class Designer also dynamically generates code based on the model, reads changes in code, and incorporates those changes into the model. The key point here is that the model and the code *are never out of sync*. This solves the problem of documentation becoming stale.

Unit Testing

Once the general outline of code is produced, tests can be written for the code. They can also be written after the code is finished. It is entirely up to you, but one thing is certain—with the VSTS unit testing tools, testing will be a lot easier, faster, and more streamlined.

Creating a unit test is as easy as right-clicking a class or a method, selecting Create Unit Tests, and filling in a couple of variables. It could also be more complicated, since unit testing supports data-driven testing, which allows for more complex scenarios without having to continually rewrite the unit tests or write many tests for one method. The unit testing functionality is also part of the Team Edition for Software Testers.

As part of the unit testing functionality, VSTS provides a very cool code coverage tool. This tool not only tells you what percentage of your code was covered versus not covered, but it can also highlight code to show you fully covered lines of code, partially covered lines of code, and code that was not covered at all. We'll elaborate on this in Chapter 12, but to give you an idea of how important this tool is, let's consider an example. Suppose you write a method 100 lines long and you run the unit tests against the code. The results all come back as passing, which is good, but the code covered comes back as 60%, which is bad, because 40 lines of code were never touched. This indicates that while all your tests passed, either you did not test something you should have or there is no way to test that code, and so it is dead code that should be removed.

Code Analysis

Since the inception of .NET 1.0, Microsoft has offered a relatively unsupported tool called FxCop (available for free from `http://www.gotdotnet.com`). VSTS incorporates this tool into the IDE so that static code analysis on managed code can be performed as part of a compilation, and policies can be written against the results of the analysis. This tool was originally created to ensure that Microsoft developers were following the correct standards when writing the .NET Framework. So, if you follow the coding recommendations of this tool, you will be writing to the same standards as Microsoft (in terms of format, completeness, and the standards of the .NET Framework).

VSTS also incorporates a tool to help developers of unmanaged code. This tool, called PREfast, has been in use within Microsoft for several years as a means for developers to check their C/C++ code for common errors such as buffer overruns. This analysis tool is run simply by checking a box in the project properties. It is customizable to an extent that allows you to implement checks not included in the out-of-the-box product.

Performance Analysis

VSTS also incorporates performance analysis tools, which allow developers to test their code for bottlenecks. In the past, performance testing around a .NET application typically involved monitoring a lot of Windows Performance Monitor logs, which provided less-than-helpful information.

The new performance analysis tools allow you to either instrument or sample your code, depending on the situation, so you can gather details at the individual method level or at the application level. Best of all, you can institute performance monitoring on a production application to pinpoint specific problems that may not have been apparent during development.

Team Edition for Software Testers

Team Edition for Software Testers is devoted to testing all aspects of your code. It includes the unit testing functionality (described in the preceding section about the Team Edition for Software Developers), load testing, manual testing, and web testing, as well as the Test Manager tool. Chapters 15 and 16 cover the Team Edition for Software Testers.

■ **Note** While VSTS does not include a Windows Forms testing capability, forms can be tested via the manual tests. In addition, the test facilities are highly extensible, and many third-party tools will probably be available to fill this niche!

Test Manager

Test management is a key component of the testing infrastructure because it allows you to organize your tests. You can run these tests in a noninteractive fashion from the command line or from the Team Foundation Build process. You can organize your tests into lists in order to run tests on like processes. Dependencies can be created between tests via an ordered test list and individual tests, or lists of tests can be slated to run at any given time. The Team Edition for Software Developers includes a subset of the Test Manager tool.

Web Testing

More and more applications are moving to the Web in today's Internet- and intranet-based world. Because of this, Microsoft rewrote the Application Center Test (ACT) web testing tool and included it with VSTS. And when we say they rewrote it, we mean it. It is a completely different tool and far, far more powerful than ACT was. You can interactively record tests and play back tests (which are displayed visually for you as the test is running). The tests can be changed in the IDE, and they can be converted to coded tests, which allow you the freedom to do virtually anything in a test you want via managed code instead of scripting, which had to be done with ACT.

All of the information about a test is recorded. If there is something additional you want to record, a custom extraction rule can be coded to do it. If you want to validate a result in the page and take an action based on the result, you can. The tests can also be run with values supplied from a database. That means that the testing can include dynamic navigation of a website. Think times can be added to each step of the test to simulate real-world browsing of a website.

Manual Testing

Another Team Edition for Software Testers feature is manual testing. This allows you to run tests that are based on a list of steps. The pass/fail status of these tests is recorded, just as any additional test is. Code coverage (if enabled) is captured for these tests as well. The steps for these tests can be written in Microsoft Word or in a plain text file in any format your organization may use.

Load Testing

Finally, the Team Edition for Software Testers provides for load testing. Load testing is designed to see how your application (hardware and software) performs under a real-world load. The options available allow for the testing of almost any situation imaginable. You can set the type of browser that is accessing the site, the type of network connection the browser is using to access the site, the way in which the think times are simulated (set times or a normal distribution, if at all), which tests are actually run as load tests, and their distribution. Ramp up times can also be set, or the tests can be run at a constant user load.

You can run the load tests from the local machine, which will simulate all of the connections. Alternatively, you can test via agents and a controller from many different machines. The controller directs the agents to run the test(s) and records the result in one location. A typical setup for this is a lab where you may have 20 test machines hitting one box with a website. This saves you the time and effort of starting the tests on all of the machines individually. The data that is collected is detailed and useful, and there is a lot of it. Every aspect of the test is recorded from both the software and the hardware perspective, and errors are stored for later analysis. The entire test result is saved to the VSTS data warehouse (if you are working with the Team Foundation piece).

Visual Studio Integration Partners Program

Visual Studio Integration Partners (VSIP) is a Microsoft program that gives developers and companies access to the application program interface (API) documentation, extensibility toolkit, and samples for Visual Studio and VSTS. With the toolkit, VSTS supports extensibility in all areas. This extensibility ranges from customizing the designers to incorporating new types of tests (such as Windows Forms tests). Many of these aspects of VSTS are touched upon briefly in upcoming chapters, and some examples are shown. However, Microsoft prefers that developers and companies who wish to provide extensibility for VSTS join the VSIP program. It is free, so there is no reason not to join it.

■**Note** There are various "levels" of the VSIP program. Free access to the extensibility toolkit is the basic level. Additional levels provide partnerships with Microsoft and access to various groups within Microsoft. It is well worth joining for commercial software development companies.

You can access the VSIP website (and join the program) at `http://msdn.microsoft.com/vstudio/partners/default.aspx`. There is a wealth of extensibility information located here.

The Effort Tracking Application

Throughout this book, we'll use a simple application as an example. This is a web-based application that records work engagements and stores them in a SQL Server database. The web application connects to a web service, which connects to the database. The deployment of this application is shown in Figure 1-2.

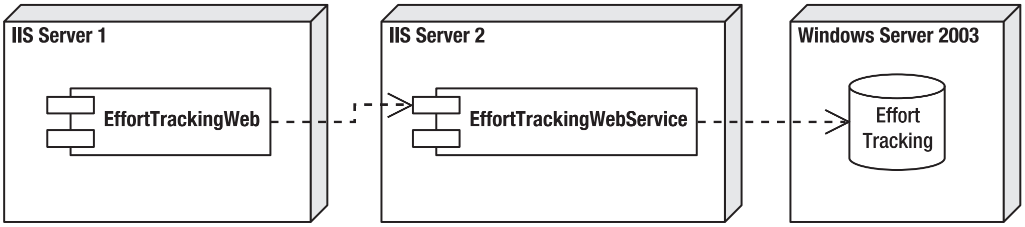


Figure 1-2. *Effort Tracking application logical deployment*

■**Note** The actual physical deployment is such that the website and web service are on the same machine but completely separated so they can be deployed in either configuration. The database itself can be located on the same machine or another machine.

The security is controlled via standard Forms security, where the username and password are stored in the database (obviously not a best practice, but for demonstration purposes only). The database contains four tables, as shown in Figure 1-3.

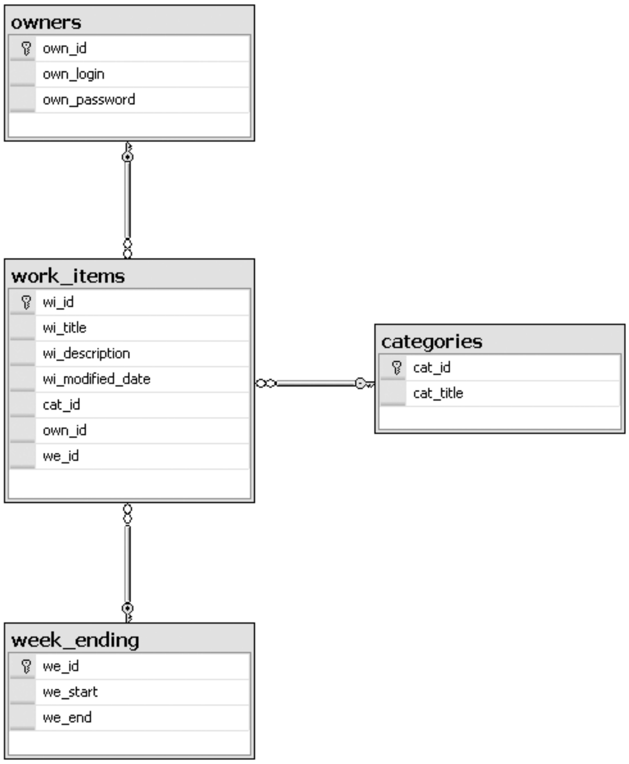
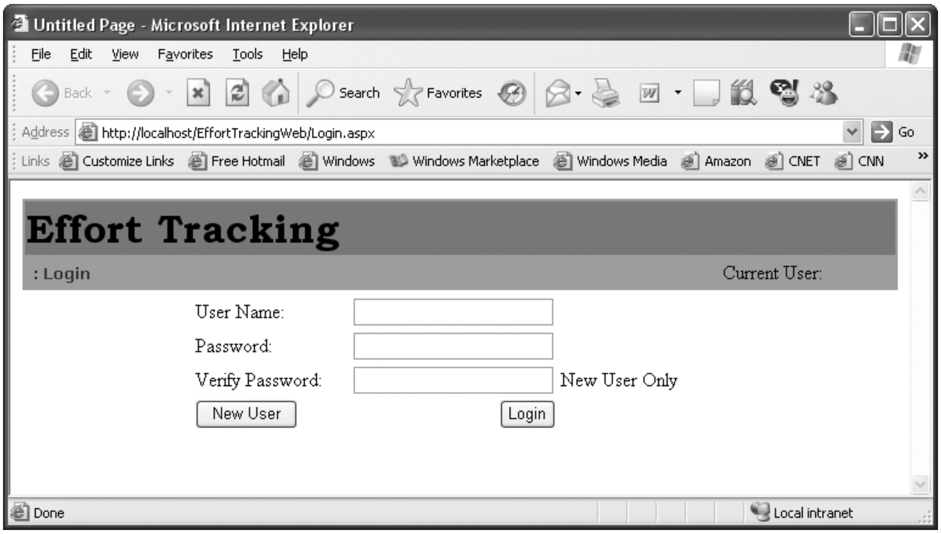


Figure 1-3. *Effort Tracking data model*

The application works as follows:

- User logs on to the system (or registers).
- User is redirected to the Effort Tracking page.
- User selects the week he wants to view.
- User adds a new engagement by clicking Add and entering the title, description, division, and the week ending date, and then clicks OK.
- User can edit or delete a record by clicking the appropriate link. The detail window is displayed, and the user can either confirm the deletion or change and save the record.

The various screens of the application are shown in Figures 1-4, 1-5, and 1-6.


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/EffortTrackingWeb/Login.aspx
Go
Links Customize Links Free Hotmail Windows Windows Marketplace Windows Media Amazon CNET CNN
Done
Local intranet

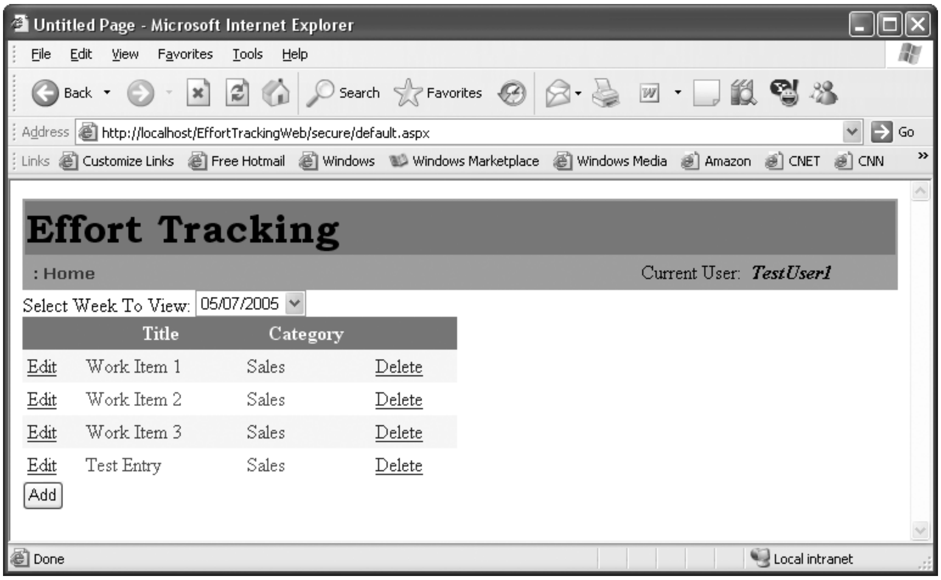
Effort Tracking

: Login Current User:

User Name:		
Password:		
Verify Password:		New User Only

New User Login

Figure 1-4. *Effort Tracking login/add new user screen*


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/EffortTrackingWeb/secure/default.aspx
Go
Links Customize Links Free Hotmail Windows Windows Marketplace Windows Media Amazon CNET CNN
Done
Local intranet

Effort Tracking

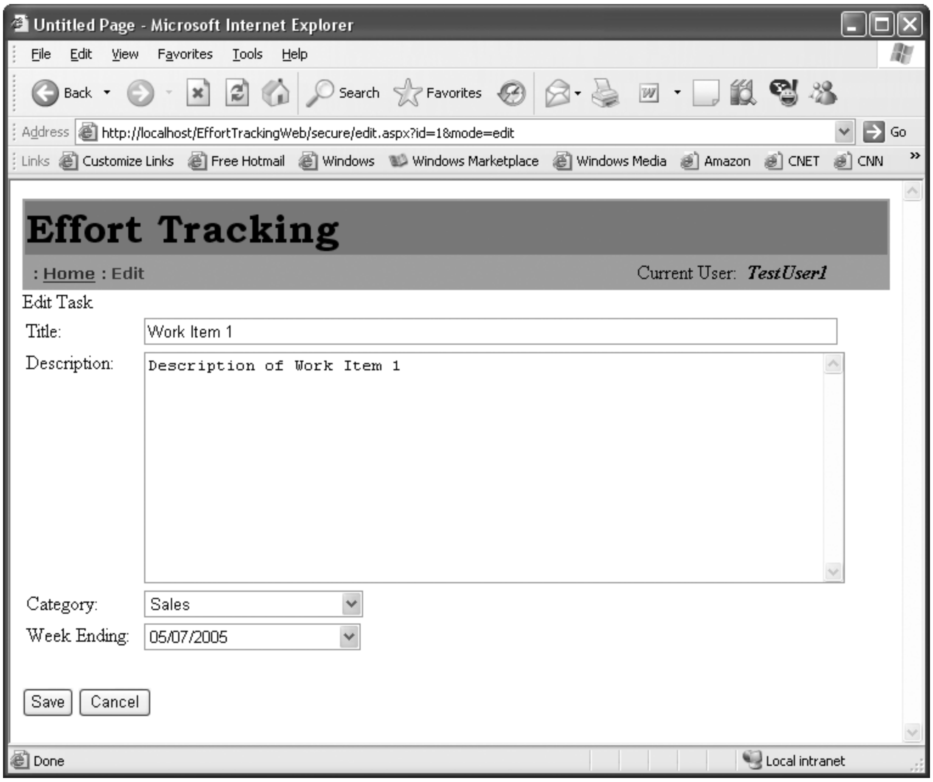
: Home Current User: *TestUser1*

Select Week To View: 05/07/2005

	Title	Category	
Edit	Work Item 1	Sales	Delete
Edit	Work Item 2	Sales	Delete
Edit	Work Item 3	Sales	Delete
Edit	Test Entry	Sales	Delete

Add

Figure 1-5. *Effort Tracking homepage*



http://localhost/EffortTrackingWeb/secure/edit.aspx?id=1&mode=edit

Effort Tracking

: Home : Edit Current User: TestUser1

Edit Task

Title:	Work Item 1
Description:	Description of Work Item 1
Category:	Sales
Week Ending:	05/07/2005

Figure 1-6. *Effort Tracking add/edit/delete page*

The web service comprises the bulk of the code for this application. There are eight methods in the web service, which handle all of the functions of the user interface.

Caution The sample application is designed to be used with the examples included in the various chapters. In many cases, it does not conform to proper development guidelines, especially with regard to security. While we do point out some best practices in this regard, this application should not be used as a model for building any type of secure applications.

Summary

Various problems face development teams today, and these can cause a project to fail. You have seen how VSTS can help organizations and individual projects solve these problems and provide a positive return on investment for everyone.

This chapter has provided a high-level view of VSTS. It described each of the editions of VSTS and an overview of the benefits they offer. Here's a quick summary:

- Team Foundation provides a new version control tool, work item tracking, Team Foundation Build, and core integration and communication features available to all stakeholders in a project.
- Team Edition for Software Architects provides architects the ability to design a system and communicate that design effectively to the stakeholders. It also provides the ability to deploy the design into logical data centers and autogenerate real code.
- Team Edition for Software Developers provides developers with the ability to understand code, generate code, and unit test code quickly and easily. It also provides the ability to analyze code for defects and to ensure conformance to coding standards.
- Team Edition for Software Testers provides testers with the ability to test all aspects of the code. Testing covers web, manual, and load testing. Test management is provided to help organize and describe the testing process.

Chapter 2 introduces the process of creating a team project. You'll explore the methodology templates and process guidance, along with how to customize them. You will also learn how to configure the security for Team Foundation, the Project Portal, and SQL Server.