Pro Visual Studio 2005 Team System

Jeff Levinson and David Nelson

Apress®

Pro Visual Studio 2005 Team System

ISBN-13 (pbk): 978-1-59059-460-5

ISBN-10 (pbk): 1-59059-460-6

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editor: Ewan Buckingham
Technical Reviewers: Gautam Goenka, Bata Chadraa, Anutthara Bharadwaj, Munjal Doshi, Winnie Ng, Joe Rhode, Siddharth Bhatia, Amy Hagstrom, Yogita Manghnani, Tom Patton, Alan Hebert, Bill Essary, Sam Jarawan, John Lawrence, Jimmy Li, Bryan MacFarlane, Erik Gunvaldson, Adam Singer, Chuck Russell, Kathryn Edens, Patrick Tseng, Ramesh Rajagopal, John Stallo, Jochen Seemann, Michael Fanning, Ed Glas, Eric Lee, Bindia Hallauer, Michael Leworthy, Jason Anderson, Michael Koltachev, Boris Vidolov, James Su, Thomas Lewis, Steven Houglum, Bill Gibson, Ali Pasha, Dmitriy Nikonov, Prashant Sridharan
Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Jason Gilmore, Jonathan Gennick, Jonathan Hassell, James Huddleston, Chris Mills, Matthew Moodie, Dominic Shakeshaft, Jim Sumser, Keir Thomas, Matt Wade
Project Manager: Sofia Marchant
Copy Edit Manager: Nicole LeClerc
Copy Editors: Marilyn Smith, Jennifer Whipple
Assistant Production Director: Kari Brooks-Copony
Production Editor: Katie Stence
Compositor: Dina Quan
Proofreader: Nancy Riddiough
Indexer: Brenda Miller
Artist: Kinetic Publishing Services, LLC
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `http://www.springeronline.com`.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail `info@apress.com`, or visit `http://www.apress.com`.

The source code for this book is available to readers at `http://www.apress.com` in the Source Code section. You will need to answer questions pertaining to this book in order to successfully download the code.

For Tami, my wonderful new wife, who kept me going through everything. I love you.
and
My cousin, Dr. Alicia Lindheim, who inspired me to go beyond myself with her courage, conviction, and strength. When the phrase, "where there's a will, there's a way" was written, they were talking about her.

— Jeff Levinson

For Sammi, my love, my inspiration, and my beautiful bride of twenty years, and for our tribe Jake, Josiah, Peter, Grace, and Lydia, my hope. I love you all with all my heart.
and
To my Mom and Dad, who gave me faith and belief in the Lord Jesus Christ, a marriage extending half a century, and a commitment to hard work, integrity, and family.

— David Nelson

Contents at a Glance

Foreword xvii
About the Authors xix
Acknowledgments xxi
Introduction xxiii

CHAPTER 1 Introduction to Visual Studio Team System 1

PART 1 Team Foundation

CHAPTER 2 Team Projects 17
CHAPTER 3 Team Foundation Version Control 59
CHAPTER 4 Project Management 103
CHAPTER 5 Team Work Item Tracking 121
CHAPTER 6 Team Reporting 159
CHAPTER 7 Team Foundation Build 197

PART 2 Team Edition for Software Architects

CHAPTER 8 Application Designer 219
CHAPTER 9 System and Logical Datacenter Designers 255
CHAPTER 10 Deployment Designer 285

PART 3 Team Edition for Software Developers

CHAPTER 11 Class Designer 311
CHAPTER 12 Unit Testing and Code Coverage 339
CHAPTER 13 Static Code Analysis 377
CHAPTER 14 Performance Analysis 403

PART 4 Team Edition for Software Testers

CHAPTER 15 Web Testing 433
CHAPTER 16 Load Testing 471
APPENDIX Command-Line Tools Reference 497

INDEX 501

Contents

Foreword xvii
About the Authors xix
Acknowledgments xxi
Introduction xxiii

CHAPTER 1 Introduction to Visual Studio Team System 1

What Is Visual Studio Team System? 2
What Are the Benefits of Using Visual Studio Team System? 2
Visual Studio Team System Editions 3
Team Foundation 4
Team Edition for Software Architects 5
Team Edition for Software Developers 7
Team Edition for Software Testers 8
Visual Studio Integration Partners Program 10
The Effort Tracking Application 10
Summary 13

PART 1 **Team Foundation**

CHAPTER 2 Team Projects 17

Starting Your Team Project 17
Planning a New Team Project 17
Connecting to the Team Foundation Server 19
Creating a New Team Project 20
Viewing Process Guidance 23
Working with the Team Explorer 24
Introducing VSTS Components 25
Process Templates 25
Project Portal 26
Work Item Tracking 29
Documents 29
Reports 30

Team Builds 30
Version Control 31
Areas and Iterations 32
Project Alerts 32
Customizing the Project Portal 32
Adding an Image 34
Adding Reports 35
Working with Lists 36
Customizing a Process Template 36
Understanding the Process Template Architecture 37
Modifying a Process Template 41
Customizing the Process Guidance 43
Managing Team Foundation Security 45
Managing Windows AD Security 46
Managing Global Team Foundation Security 47
Managing Project Security 52
Managing WSS Security 55
Managing SSRS Security 56
Summary 57

CHAPTER 3 **Team Foundation Version Control** 59

Starting with Version Control 59
Source Control Explorer 61
File/Folder Properties 62
Workspaces 66
Creating Solutions 68
Pending Changes Window 70
Source Files 70
Work Items 71
Check-in Notes 71
Policy Warnings 71
Changesets 72
History 73
Comparing Versions 74
Labeling Versions 76
Retrieving Versions 78
Branching 80

Shelvesets 82
Merging Files 83
Configuring Version Control 89
Configuring Project Settings 90
Check-in Notes 93
IDE Version Control Configuration 93
Plugin Selection 94
Environment 94
Visual Studio Team Foundation 94
Creating Custom Check-in Policies 95
Creating the Policy 95
Registering the Policy 99
Converting from Visual SourceSafe to Team Foundation Version Control 100
Command-Line Access 101
Summary 102

CHAPTER 4 **Project Management** 103

A Week in the Life of a Project Manager (without VSTS) 104
Day 1, Monday 104
Day 2, Tuesday 104
Day 3, Wednesday 104
Day 4, Thursday 104
Day 5, Friday 104
Two Days in the Life of a Project Manager (with VSTS) 105
Day 1, Monday 105
Day 2, Tuesday 105
Using Microsoft Project 105
Retrieving, Adding, and Updating Work Items 107
Adding Attachments and Links 109
Areas and Iterations 112
Column Mapping 112
Using Microsoft Excel 115
Creating Lists in Excel 116
Configuring Lists in Excel 117
Using Visual Studio 119
Summary 120

CHAPTER 5 Team Work Item Tracking 121

Working with Work Items 121
Using the Form View 123
Using the Query View 125
Using the Results (Triage) View 129
Understanding Work Item Types 130
Task Work Items 131
Bug Work Items 133
Risk Work Items 134
Change Request Work Items 136
Review Work Items 137
Requirement Work Items 138
Issue Work Items 140
Configuring Project Alerts for Work Item Tracking 141
Looking Under the Hood 142
Customizing Work Item Types 144
Modifying the Assign To List 145
Creating a New Field That References a Global List 149
Creating a New Work Item Type 152
Assigning Field Reference Names 156
Summary 157

CHAPTER 6 Team Reporting 159

Introducing the Business Intelligence Platform 159
Understanding the Reporting Life Cycle 161
Using Predefined Team Reports 162
Customizing Team Reports 166
Extracting the Report Definition 168
Adding the Report Definition to a Project 168
Modifying the Report 169
Saving the Modified Report Definition 170
Deploying and Viewing Your Report 171
Introducing the Team Foundation Data Warehouse 172
Understanding the Data Warehouse Architecture 173
Exploring the Data Warehouse Schema 174
Managing the Data Warehouse 181
Adding Elements to the Data Warehouse 182

Data Mining with Microsoft Excel 182
Bringing Team Foundation Data into Excel 183
Creating a Report 187
Creating a New Report 189
Summary 196

CHAPTER 7 **Team Foundation Build** 197

Benefits of Automated Builds 197
Using Team Foundation Build 198
Creating a Build Type 198
Running the Build and Viewing Build Results 203
Viewing Build History 208
Customizing the Build Process 211
Reviewing the Build Type Configuration File 211
Retrieving the Build Type 212
Editing the Build File 212
Using the Build Command-Line Tool 213
Setting Up Continuous Integration Testing 214
Summary 215

PART 2 **Team Edition for Software Architects**

CHAPTER 8 **Application Designer** 219

Overview of the Distributed System Designers 220
System Definition Model 220
Benefits of the Distributed Designers 221
Using the Application Designer 223
Getting Started with the Application Designer 224
Defining the Database Component 226
Adding a Web Service Component 227
Connecting the Application Types 228
Defining Operations for ASP.NET Web Service Prototypes 232
Implementing the Application 233
Adding a Web Application Component 240
Adding Comments to Application Diagrams 244
Understanding Connections and Endpoints 245
Adding a Web Service Endpoint from a WSDL File 247

Understanding Constraints and Settings 249
Setting Constraints 249
Searching Settings and Constraints 250
Reverse-Engineering Existing Solutions 251
Troubleshooting Application Diagrams 253
Summary 253

CHAPTER 9 System and Logical Datacenter Designers 255

Using the System Designer 255
Creating a New System Diagram 256
Building a System Diagram from an Application Diagram 261
Nesting Systems 262
Viewing Web Service Details, Settings, and Constraints 264
Overriding Settings and Constraints 265
Using the Logical Datacenter Designer 266
Creating a Logical Datacenter Diagram 267
Importing Settings from IIS 277
Building a Logical Datacenter Diagram for the Sample Application 281
Summary 283

CHAPTER 10 Deployment Designer 285

Using the Explicit Deployment Method 285
Creating a Deployment Diagram 287
Validating a Deployment Implementation 291
Generating the Deployment Report 295
Using the Implicit Deployment Method 296
Building the Deployment Diagram 297
Validating the Diagram 301
Setting Deployment Properties 303
Generating the Deployment Report 305
Summary 307

PART 3 Team Edition for Software Developers

CHAPTER 11 Class Designer 311

Design Goals 311
- Understanding Existing Code 311
- Initial Class Design 312
- Reviewing and Refactoring 312
- Relevant Diagrams and Documentation 312

Microsoft, UML, and Visio 313
Using the Class Designer 314
- Exploring a Class 316
- Viewing Properties 317
- Working with Class Designer Tools and Options 319
- Adding Items to the Class Diagram 320
- Working with Interfaces 327
- Showing Object Relationships 330
- Adding Fields and Properties 333
- Adding Comments 335

Looking Under the Hood 336
Summary 337

CHAPTER 12 Unit Testing and Code Coverage 339

Planning Unit Tests 339
Creating Unit Tests 342
Understanding Unit Tests 346
- Exploring a Test Method 346
- Exploring a Test Class 348

Managing Unit Tests 349
- Using the Test View Window 349
- Using the Test Manager Window 351
- Creating Ordered Tests 353

Setting Up Tests 355
Configuring Test Runs 355
Completing the Test Methods 358
Setting Other Configuration Options 359
Running Tests 359
Viewing the Test Run Information 360
Viewing Code Coverage Results 362
Testing for Exceptions 364
Data-Driven Testing 367
Building a Test Database 367
Preparing the Production Database 368
Setting Up the Test 370
Manual Testing 373
Creating a Manual Test 374
Running a Manual Test 375
Testing Using MSTest 375
Summary 376

CHAPTER 13 Static Code Analysis 377

Static Code Analysis vs. Code Reviews 377
Using PREfast 378
Enabling PREfast Checking 378
Reviewing PREfast Results 379
Enabling, Disabling, and Suppressing PREfast Warnings 383
Annotating Code for PREfast Checks 385
Using FxCop 387
Enabling FxCop 388
Examining FxCop Results 388
Suppressing Messages 390
Configuring FxCop Rules 391
Running FxCop from the Command Line 392
Creating Custom FxCop Rules 392
Summary 401

CHAPTER 14 Performance Analysis 403

Performance Profiling Terms 404
Instrumentation 405
Sampling 407
Running a Performance Test 407

Understanding the Performance Report 409
Summary Tab 409
Functions Tab 411
Caller/Callee Tab 414
Calltree Tab 415
Allocation 416
Objects Lifetime 417
Performance Session Options 418
Target Options 420
Profiling Unit Tests 421
Profiling Web/Load Tests 421
Profiling Production Applications 422
Command-Line Performance Tools 422
Profiling Windows Applications and Windows Services 423
Profiling ASP.NET Applications 425
Summary 430

PART 4 Team Edition for Software Testers

CHAPTER 15 Web Testing 433

Recording Web Tests 434
Test Steps Explained 437
Test Detail Properties 437
Test Options 439
Running Recorded Web Tests 440
Passing or Failing Tests 443
Data-Driven Web Testing 443
Coded Web Tests 447
Coded Data-Driven Tests 449
Extraction Rules 453
Creating Custom Extraction Rules 454
Extract Method 456
Implementing Custom Extraction Rules 456
Validation Rules 457
Creating Custom Validation Rules 459
Web Test Request Plugins 459
Web Test Plugins 459
Testing Web Services 462

Test Results 465
Test Results Schema 466
Publishing Test Results 468
Summary 469

CHAPTER 16 Load Testing 471

Controllers and Agents 472
Administering a Controller 473
Configuring the Test Run 476
Load Test Wizard 477
Scenario 477
Load Pattern 479
Test Mix 480
Browser Mix 481
Network Mix 482
Counter Sets 483
Run Settings 484
Extending Load Test Settings 484
Load Test Window 488
Analyzing Load Test Results 491
Counters Pane 492
Graphs/Tables Pane 493
Points Pane 495
Summary Pane 495
Publishing Test Results 495
Summary 495

APPENDIX Command-Line Tools Reference 497

Server Command-Line Tools 497
Client Command-Line Tools 498

INDEX 501

Foreword

Microsoft has always provided world-class development tools for developers. From the release of Visual Basic 1 to Visual Studio 2005, Microsoft has provided groundbreaking tools to make developers' lives easier. With the release of Visual Studio 2005, we've done a considerable amount of work to help the individual developer be even more productive—refactoring, the My Namespace, edit-and-continue, and improvements in the .NET Framework are just a few examples.

But with Visual Studio 2005, we've expanded our focus beyond the developer to the entire development process itself. Visual Studio Team System takes a larger view of the developer's world. It acknowledges the project management aspects of development, the architecture, and the testing phase of the development life cycle; in other words, Visual Studio Team System takes into account the entire software development life cycle. This shift away from purely an individual's perspective is designed to ease the burden on development organizations by helping every member of the team gain more insight, and oversight, of the software development life cycle.

This larger view of the development process promotes communication and collaboration among groups that in the past almost never spoke with each other. It helps project managers to communicate with architects, architects with developers, and developers with testers. And it helps everyone to communicate with stakeholders and to collaborate with other interested observers. By providing timely reporting of events, project status, development statistics, and other information, organizations can leverage Visual Studio Team System to streamline the development process.

Software development has shifted from groups of developers working in the same building to groups of developers working around the world. This shift crosses geographical boundaries and allows teams to collaborate with each other in real time. Team System enables the organization that owns the code to actually own the code! Nightly builds can be performed in the target environment instead of being built and tested in an environment that usually doesn't match the eventual deployment environment. It also helps organizations to keep better track of how their outsourced teams are progressing with the project.

There is an overwhelming industry trend toward a more agile approach to software development. At the same time, corporations are pushing for improved quality and reduced cost through repeatable processes. Out of the box, Visual Studio Team System provides two process templates aimed at meeting the vast majority of team project needs. The MSF for Agile template is great for teams that may not have used any "formal" methods in the past, while the MSF for CMMI process template complements the Capability Maturity Model Integration (CMMI) process improvement approach developed at Carnegie Mellon. These processes help developers not only to be productive, but also to create a framework for repeatable development processes. These process improvements ultimately lead to higher quality, lower cost software development.

With all of this, Microsoft has absolutely not forgotten that it is developers who write the code and run the tests. One of the Visual Studio Team System development team's primary goals is to help developers to write high-quality code. We call this helping teams drive better quality, early and often. With the new testing tools for developers and testers, code can be written with a higher quality (it follows standards, passes unit tests, and performs optimally) and that quality can be tested for at every step of the way. Team System makes continuous integration testing, build verification tests, and development standards easy to implement and follow. These processes, instead of being complicated and difficult to follow, are made exceptionally easy with Team System and will help developers write and release more stable code.

What's in store for the future then? At our core, the things we do at Microsoft are all about empowering people to drive business success. With Team System, we've taken the first steps toward helping individuals and teams be more productive in the development process. Our vision is to further expand our Team System offering to help all members of an IT team communicate and collaborate more effectively. Some of the areas that we're focusing on in the immediate future are better tools for working with databases, more advanced and complete testing tools, better integration with Microsoft Office, and better support for managing your organization-wide portfolio of projects. Though some of these new tools may not even be released under the "Visual Studio" brand, you can be assured that we will work diligently across all our product groups to deliver solutions that will help you and your organization be more successful.

We believe that Visual Studio 2005 Team System is a hallmark product for the software industry. To be sure, we're all extraordinarily proud of what we've released. But, even more than that, we're excited to see the overwhelming positive reaction to these first few steps in making our customers' lives easier. On behalf of the entire team in Redmond, North Carolina, India, Copenhagen, and elsewhere, thank you for your support, feedback, and encouragement.

Prashant Sridharan
prashant@microsoft.com
Director, Visual Studio
March 2006

About the Authors

JEFF LEVINSON is a Solution Design and Integration Architect for The Boeing Company. He is the author of *Building Client/Server Applications with VB .NET: An Example-Driven Approach* (Apress 2003) and has written several articles for *Visual Studio Magazine*. He speaks at various Microsoft user groups and was a speaker at Microsoft's DevDays 2004. Jeff holds the following certifications: MCSD, MCAD, MCSD.NET, MCDBA, SCJP, and Security+. He is currently finishing his Masters in Software Engineering at Carnegie Mellon University. He and his wife Tami live in Redmond, Washington. He enjoys golfing, reading, running, and spending time with his wife.

DAVID NELSON is an Enterprise Solutions Architect and Associate Technical Fellow for The Boeing Company, where he has been employed for 20 years. His tenure at Boeing has allowed him to become expert at various technologies, including database solutions, grid computing, service orientation, and most recently, Visual Studio Team System. David is currently responsible for architecture and design of computing solutions across Boeing, with primary focus on the application of emergent technologies. He has taught Windows Server System (SQL Server, SharePoint Server, and Windows Server) classes, and is regularly invited to present at national industry conferences.

David resides in the state of Washington with his wife and five children, where he enjoys riding horses and motorcycles. He is building a tree fort with his sons, planting a garden with his daughters, and restoring a horse trailer for his wife.

Acknowledgments

Writing a book—any book—is difficult at best. For a new product on which there really is no material to use to research on your own, it is even more difficult. The members of the Visual Studio Team System development team have been incredibly gracious and giving of their time to answer questions, go over features, and provide support in general while they were going through their development and release cycles. All of the information in this book comes from the authors fooling around with the product, trying to implement it in an enterprise environment, and from the developers and program managers at Microsoft. Having said that, the authors would like to thank the following people from Microsoft (in no particular order), keeping in mind that many, many more helped us bring this book to you: Gautam Goenka, Bata Chadraa, Anutthara Bharadwaj, Munjal Doshi, Winnie Ng, Joe Rhode, Siddharth Bhatia, Amy Hagstrom, Yogita Manghnani, Tom Patton, Alan Hebert, Bill Essary, Sam Jarawan, John Lawrence, Jimmy Li, Bryan MacFarlane, Erik Gunvaldson, Adam Singer, Chuck Russell, Kathryn Edens, Patrick Tseng, Ramesh Rajagopal, John Stallo, Jochen Seemann, Michael Fanning, Ed Glas, Eric Lee, Bindia Hallauer, Michael Leworthy, Jason Anderson, Michael Koltachev, Boris Vidolov, James Su, Thomas Lewis, Steven Houglum, Bill Gibson, Ali Pasha, Dmitriy Nikonov, and Prashant Sridharan.

We owe a special thanks to Gordon Hogenson. Neither of the authors is a C/C++ expert. Because of this, we turned to someone who is an expert for help with a section in Chapter 13 of this book. He wrote an excellent discussion of PREfast, clearly explaining what you can do with it to write better code. Thanks Gordon!

The authors would like to also thank our editor, Ewan Buckingham, who stuck with us through this whole process, which took considerably longer than usual. Sofia Marchant, as the Project Manager for this book, kept us on track. She took care of getting the materials to the right people at the right time for reviews and pushing people to get information back to us. Thanks Sofia! Without our Copy Editors, Marilyn Smith and Jennifer Whipple, this book would not flow nearly as well or be so readable. Thank you for all of your advice and rewording! Katie Stence kept everything on track for our production edits.

Without the hard work of everyone at Apress, this book would not be in your hands now.

Jeff Levinson and David Nelson

In addition to all of the great people at Microsoft and Apress, this book has had an effect on everyone around both David and myself. It has taken a lot of time and effort, more so because of the constantly shifting nature of working with a new product. Along that line, I would like to thank the following people from my team at Carnegie Mellon University: Angela He, Kiran Hedge, Ed Shepley, Drew Gattis, and Michael Rosett. They put up with me while I was trying to get my school work done, write a book, work, and do a couple of other things. It was a great year working with a supportive team. As usual, I would like to thank my family for their support and shoulders to lean on. I would like to thank all of the great people I work with at Boeing, from developers to managers and my coworkers, for their support of this endeavor over the last two years.

Finally, I would like to thank my coauthor, David. When I first envisioned the idea for this book, I knew there were two issues: 1) There was just too much to write about on my own, and 2) I knew this was going to be a very long road. So I convinced David that we should write the book together. He hung in there with me, even when it seemed like we were getting new builds every month (which we were most of the time), Microsoft kept changing the name of portions of the product, and they kept changing the feature set. His wife was ready to cause me serious harm for monopolizing David's time, and he got to see his kids for only an hour a day or so. Sammi, I'm sorry! But we're done, and in the end, it was a great experience. Thanks David!

Jeff Levinson

I would like to thank everyone who has been excited and encouraging regarding this project. It has been a long road, and we have learned much. Thanks to those who have listened, guided, and supported this effort: The Guys (Brad Carpenter, Tim Pearson, John Rivard, Sam Jarawan, Jeff Whitmore, Gerd Strom, and Johnny Couture) are my rock. Thanks also to Dr. Karl Payne, my mentor, teacher, and friend. The Cassandra Team (Roger Parker, Richard Lonsdale, Gary Westcott, Fred Ervin, and John Zhang) are early adopters of ideas and technology. The ValSim Team (Mike Maple, Kaaren Cramer, Jacques Rousseau, Phil Trautman, and others) push the edge of technology and thought. Team Canada (Steven Guo, Rob Hickling, Stig Westerlund, and others) who take beta tools and make products that work. The Architects (Todd Mickelson, Mick Pegg, Dick Navarro, Brad Belmondo, David Rice, Marty Kitna, and others) have vision, trust, and work to "get'r" done.

And lastly, thanks to Jeff Levinson, my partner in this endeavor. I have learned a great deal over the past 22 months (yeah, it really has been that long; I found the first e-mail). I would never have taken on a project like this book without Jeff's encouragement, expertise, and drive. Since this was his second book, he patiently guided me through some of the finer points of authorship. He would often say, "This isn't a blog; you need to write it like a book." Jeff did the greater portion of work, and I appreciate him letting me join him on this journey. I also want to thank his new bride Tami for letting us work at the house and take time away from the more important wedding plans. He's all yours now! Thanks Jeff, it was a great adventure.

David Nelson

Introduction

Software development is undergoing a huge revolution right now. That is not to say that this is different from any other time in the history of software development—it is always undergoing a revolution of one type or another. This "new" revolution is a shift in development methodologies from teams working in tightly knit groups in geographical proximity to a global development strategy. Software development teams are now faced with communicating with each other from half a world away.

This book talks about how to perform this work *effectively* using the new Microsoft Visual Studio Team System (VSTS) product. This book covers all areas of VSTS, from the basics to tips and tricks to make it easier to use. Because of our work with the development team at Microsoft, we have been able to include several undocumented features and describe some of the thought processes involved in developing various portions of VSTS. In addition, as architects in a Fortune 500 company, we have a unique experience in starting to implement VSTS in an enterprise environment.

This book begins with a chapter that introduces VSTS. Chapter 1 provides a general overview of VSTS, its various components and editions, and who should use it and why. This chapter also introduces the sample application that we use throughout the book. Following the first chapter, the book is organized into four parts.

Part 1, Team Foundation: The Team Foundation Server is the mechanism (set of integrated services and stores) that enables the communication and collaboration aspect of VSTS. The web services provide a loosely coupled interface between the various artifacts (work item tracking, version control, build, and test). The operational stores provide a real-time repository of team activity that feeds the aggregated data warehouse for team reporting and analysis. Part 1 of the book covers this crucial component of VSTS.

- *Chapter 2, Team Projects*: This is your first hands-on introduction to VSTS. Chapter 2 walks you through creating a new team project, introduces you to the Project Portal, and explains how VSTS leverages various software development life cycles and provides integrated process guidance. Chapter 2 also discusses Team Foundation Server security, from both the user's and administrator's perspective.

- *Chapter 3, Team Foundation Version Control*: One of the much anticipated new features of VSTS is Team Foundation Version Control, a new, enterprise-class source code control system. This chapter covers all of the aspects of the VSTS source code control system. It also gives an in-depth look at the new check-in policies and touches on how these policies integrate with the work item tracking system.

- *Chapter 4, Project Management*: Microsoft has expended considerable effort to bring project managers into the software development life cycle. VSTS provides integration between Microsoft Project, Excel, and the work item tracking store. Project managers can extend the default mappings to any field available in Microsoft Project. Team Explorer provides rapid triage of work items. The Project Portal and reporting site provide a wealth of information about the status of the team project. This chapter describes all of these features.

- *Chapter 5, Team Work Item Tracking*: Work item tracking is one of the hottest new features in VSTS. This feature allows a project manager to create a work item (a task, a bug, an issue, and so on), assign it to a team member, and track the status of it from beginning to end. Stakeholders can see how a certain item is progressing as well. Work item tracking is a fully extensible system, so project teams can create their own work item types. Work item attachments can include documents, links to other work items, code, or URLs. This chapter covers work item tracking in detail.

- *Chapter 6, Team Reporting*: SQL Server Reporting Services (SSRS) was introduced as an add-on to SQL Server 2000 several years ago. With the new SQL Server 2005 and the new SSRS, Microsoft has made this tool the core of the VSTS reporting infrastructure. This chapter covers the details—from the out-of-the-box reports (associated with each process template) to the specific features on which the VSTS data warehouse allows you to report.

- *Chapter 7, Team Foundation Build*: In the past, performing automated builds required a great deal of extra work using Microsoft tools. This chapter covers the new Team Foundation Build functionality and shows how you can use it to increase the quality of the final software product.

Part 2, Team Edition for Software Architects: This part of the book is dedicated to the new distributed designers in VSTS. These designers allow you to architect an entire application and then implement portions of the application: projects, configurations, and settings.

- *Chapter 8, Application Designer*: In this chapter, an overview of model-driven development, Software Factories, and Domain-Specific Languages leads into a discussion of the Application Designer. The Application Designer allows you to take the first step in a "contract-first" development process, in which you design the interface before writing the application. Having defined the operations for your services, you can implement real code that stays in sync with the models.

- *Chapter 9, System and Logical Datacenter Designers*: Systems are defined as deployable units of the overall application. The level of abstraction provided by the System Designer allows multiple designs to facilitate deployment onto varying datacenters, customer sites, or geographic locations. The Logical Datacenter Designer allows the creation of models depicting interconnected hosts and provides invaluable implementation details to both the application architect and the developer at design time. Chapter 9 describes how to use both of these designers.

- *Chapter 10, Deployment Designer*: As you will learn in this chapter, the Deployment Designer allows architects and developers to deploy systems into the target logical datacenters. The result is instant validation on configuration, setting, or hosting constraint conflicts.

Part 3, Team Edition for Software Developers: Software developers now get the benefits of a concrete modeling language and strong unit testing tools to help them visualize and implement code with higher quality. To augment this capability, developers can analyze their code for common errors and ensure their code meets organizational coding standards. They can also analyze their code for performance impacts and ways to improve the application's performance. This part of the book describes the VSTS tools for modeling code, unit testing, and code analyses.

- *Chapter 11, Class Designer*: UML can be confusing and complicated. It can take a long time to write and even longer to implement. The implementation is often poorly done because UML is an abstract modeling language. As you'll learn in this chapter, the Class Designer is a concrete modeling language for .NET. The Class Designer can both forward- and reverse-engineer code in a way that makes sense with .NET.

- *Chapter 12, Unit Testing and Code Coverage*: Developers now have the ability to test their own code directly from within Visual Studio. You can perform detailed tests and gather code coverage statistics to ensure your code is of high quality and is thoroughly tested. This chapter explains how these VSTS features work.

- *Chapter 13, Static Code Analysis*: Static code analysis deals with examining code in order to ensure that standards were followed and that any known defects are caught ahead of time. This includes managed *and* unmanaged code (C/C++). In this chapter, you will learn about how the FxCop and PREfast utilities can reduce coding errors and increase maintainability.

- *Chapter 14, Performance Analysis*: Is your application not performing as you expected? Does it need more speed? Analyze your application and improve your users' experience with the new VSTS performance analysis tools. You can either instrument your application for detailed analysis or sample it for long-term performance monitoring. Use these techniques for code under development or production code. Chapter 14 describes how.

Part 4, Team Edition for Software Testers: Testing is becoming an increasing critical area of software development. Customers expect fewer bugs out of the box, and that means development teams need to provide more testing resources. This part of the book discusses the new VSTS testing tools and how to use them to create more reliable applications.

- *Chapter 15, Web Testing*: Many companies are switching to web applications as a way to decrease maintenance costs and allow users to access applications from anywhere. Testing can often be difficult and time-consuming. With the new web testing tools, you can now easily create scripts to test your web application *or* web services. This chapter covers web testing in detail.

- *Chapter 16, Load Testing*: Do you want to know how your application will stand up under high load? Are you wondering when it will fail and what you need to do to prevent it from failing? As you will learn in this final chapter, using the new load testing tools, you can identify points of failure and determine strategies for dealing with high-load situations.

At the end of the book, you will find an appendix that contains a list of all of the command-line tools available for use with VSTS (client and server).

So, now that you know what this book contains, let's get started.

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.

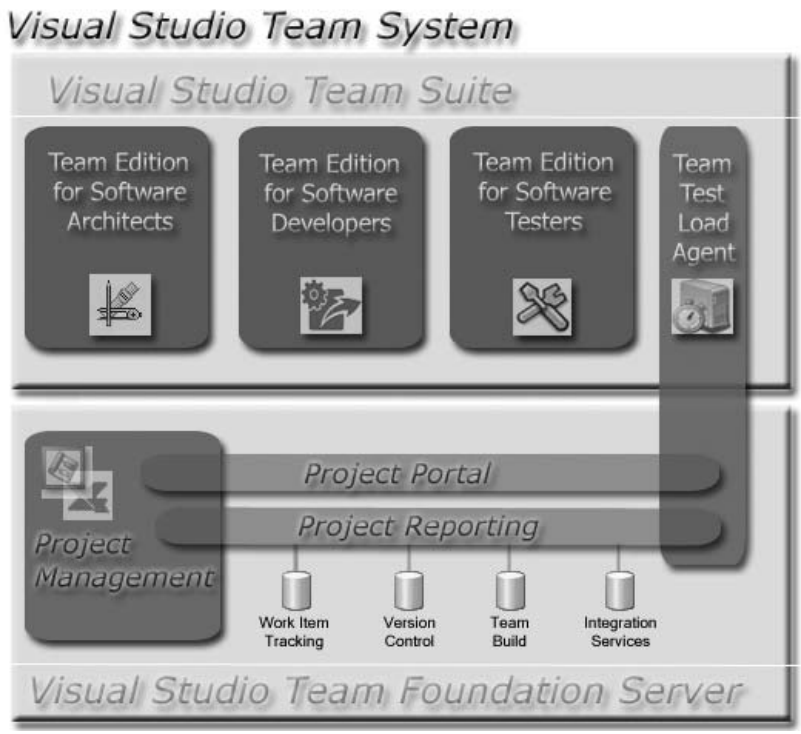

Figure 1-1. *Visual Studio Team System editions and main components*

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,