Professional
Visual Studio® 2005
Team System

Jean-Luc David,
Tony Loton,
Erik Gunvaldson,
Christopher Bowen,
Noah Coad,
Darren Jefford
Professional
Visual Studio® 2005
Team System

Jean-Luc David,
Tony Loton,
Erik Gunvaldson,
Christopher Bowen,
Noah Coad,
Darren Jefford
Credits

Executive Editor
Bob Elliott

Development Editor
Brian MacDonald

Contributors
Joe Sango
Etienne Tremblay
Steven Borg
Mike Diehl
Mickey Gousset

Technical Editors
Steven Borg
Jason Camp
Rob Caron
Steve Cook
Mickey Gousset
Bill Eassary
Doug Neumann
Bruce Taimana
David Anderson
Sam Guckenheimer
Khushboo Sharan
Tom Patton
Yogita Manghnani
Randy Miller
John Turley
Susan Llewellyn
Dave Schmitt
Shawn Elliott
Rick Rainey
Wayne Ewington
Barry Fridley
Yulin Jin
Georg Drobný
Esther Fan
Ajoy Krishnamoorthy
Dominic Hopton
Carsten Lemm
Cissy Ho
Prasad Kakulamarri
Jenniger Norberg
Bill Gibson

Production Editor
William A. Barton

Copy Editor
Luann Rouff

Editorial Manager
Mary Beth Wakefield

Production Manager
Tim Tate

Vice President and Executive Group Publisher
Richard Swadley

Vice President and Publisher
Joseph B. Wikert

Media Development Specialists
Angela Denny
Kit Malone
Travis Silvers

Project Coordinator
Kristie Rees

Graphics and Production Specialists
Karl Brandt
Andrea Dahl
Mary J. Gillot
Denny Hager
Joyce Haughey
Alicia South

Quality Control Technicians
Laura Albert
Joe Niesen

Proofreading and Indexing
Christine Sabooni, Techbooks
About the Authors

Jean-Luc David is a current Team System MVP and a Toronto-based trainer, consultant, and author. He is also the CEO/Lead Developer for Stormpixel Solutions, a company focused on developing .NET solutions for the desktop, mobile devices, and the web. He has published numerous technical articles on ASPToday.com, C|NET, Builder.com, and XML.COM. He also co-authored Professional Javascript, Second Edition and Professional WinFX Beta for WROX Press and Wiley Publishing. Jean-Luc has the distinction of being the first Canadian to receive the Microsoft Visual Developer-.NET Most Valuable Professional (MVP) award. His recent activities include presiding over the Toronto Windows Server User Group, a collective of over 853 IT Professionals in the greater Toronto Area. He is also part of the administrative team of the East of Toronto .NET User Group and is a speaker in the MSDN/INETA Canada Speakers Bureau.

Erik Gunvaldson is the Technology Development Manager in the Microsoft Enterprise Partners Group where he is currently focused on process methodologies and driving the Software Factories vision across Microsoft partners. Prior to the fall of 2005, Erik was the first Microsoft Technical Evangelist for Visual Studio 2005 Team System. In this role, Erik was responsible for the Team System Technology Adoption Program (TAP) for partners and enterprise customers. Before coming to Microsoft, Erik worked for several large enterprises in roles ranging from C++ developer and software manager to distributed application architect. Erik’s professional goal is to automate the building of software solutions to the point where it is 90 percent inspiration and 10 percent perspiration. When not spending time thinking about software, Erik enjoys spending time with his wonderful wife, Anna, their beautiful daughter, Katrina, and their big black lab, Joe.

Noah Coad is currently a Program Manager in the Developer and Test Tools product unit of Visual Studio Team System. His focus is on developer-driven testing, including unit testing and code coverage. As a community lead for Team System, Noah is responsible for engaging with MVPs, regional directors, and other key influencers. He is a former C# MVP and a developer at heart. While obtaining a B.S. in Computer Engineering from Texas A&M University, Noah worked as a contract programmer, taught .NET, and helped lead the initial C# online community. He enjoys mountain biking, hard-core coding, creating gizmos with microcontrollers, and spending time with his beloved wife, Dawn.

Darren Jefford is an Application Development Consultant working for Microsoft in the U.K. In his spare time (of which there isn’t much), he likes to be with his young family, follow Formula 1, play the guitar, and tinker with digital photography.

Tony Loton is a Microsoft Certified Professional for .NET Solution Architectures and MSF 3.0. He works through his company, LOTONtech Limited (www.lotontech.com) as an independent consultant, instructor, and freelance author in addition to holding an appointment as Associate Lecturer for the United Kingdom’s Open University. Tony graduated in 1991 with an honors degree in Computer Science with Management, and has authored many published works, including the book Professional UML with Visual Studio .NET and a Visual Studio 2005 article series for the MSDN Developer Center.
Christopher Bowen is the Lead Applications Architect at Monster.com in Maynard, Massachusetts, where he works on the design, implementation, and optimization of Monster’s applications. Chris is highly involved in the .NET development community, contributes to running the Boston .NET User Group, and speaks on a variety of subjects at area developer events. He is a member of Microsoft’s Patterns & Practices Customer Advisory Board and the Microsoft East Region Architect Council. Christopher holds a masters of science in Computer Science and a bachelors of science in Management Information Systems from Worcester Polytechnic Institute in Worcester, Massachusetts.
Acknowledgments

This book is dedicated to my beautiful, loving wife, Miho, for supporting me in a million ways. My family also deserves a loud, big round of applause for their ongoing patience and support. First and foremost, I would like to thank Rob Caron (and the rest of the Team System product group) for providing invaluable technical advice, comments, and resources (even at all hours of the night). Next comes Noah Coad, who is undeniably the nicest guy I’ve ever met and a great friend. He co-authored a chapter in this book despite his busy schedule, and helped out in countless ways. Erik Gunvaldson has contributed a great deal to this book—his breadth and depth of knowledge on Team System made an indelible impact on its contents. Mickey Gousset, a fellow Team System Most Valuable Professional (MVP) and all-around amazing guy, dedicated hours of his time to provide fantastic technical reviews in record time and really made a positive contribution—thanks, Mickey! Other product team members who have made significant contributions to the book include Sam Guckenheimer, Sean Sandys, Jennifer Norberg, Dominic Hopton, Ajay Sudan, Ayesha Mascarenhas, Buck Hodges, Khushboo Sharan, Bindia Hallauer, Beny Rubenstein, Chris Lucas, Rick LaPlante, Prasad Kakulamarri, Ed Hickey, and countless more. Next, I would like to thank all my co-authors for your amazing chapters—without you this book wouldn’t exist: Erik, Noah, Darren Jefford, Christopher Bowen, and Tony Loton. Other MVPs and experts who have authored parts of the book (in its final stages) include Joe Sango, Etienne Tremblay, Steven Borg, Mickey Gousset, and Mike Diehl. Joel Semeniuk and Baryr Gervin provided some great feedback and technical edits on some of the chapters. A very special thanks goes out to the editors and publisher at Wiley Publishing: Robert Elliott for his ongoing support and advice. Other Wiley/Wrox contributors include Helen Russo and Bill Barton. Last, I’d like to thank Chris Dufour for being a great friend and soundboard. —Jean-Luc David

Many thanks to Thomas Delrue for providing the SDM sample at the end of Chapter 7. After teaching Microsoft .Net and related technologies in Belgium, Thomas moved to the United States to join the Visual Studio team at Microsoft Corporation. He currently works in the Distributed Systems Designers team as a Software Design Engineer in Test. He can be reached via his personal e-mail at thomas.delrue@tiscali.be. —Erik Gunvaldson

Special thanks to Dawn Coad, my lovely, sweet wife, who has been my inspiration and support. She is truly God’s gift and makes my life a joy. Many thanks to Jean-Luc David, who made this book possible and is one of our top Team System MVPs, and my team, including Chris Lucas, Rob Caron, Tom Arnold, Tom Marsh, and Dominic Hopton, who provided feedback and support for the material. —Noah Coad

I’d like to thank my fantastic wife, Julieann, and our wonderful children, Lucy and Toby, for their love, support, and patience. I’d also like to thank Duncan Pocklington, Dave Thomas, and the DSL Tools team in Cambridge, all of whom helped to get the boxes joined together. Last, but not least, thanks to Microsoft, Keith Everitt, Totem, and Simon Dutton, all of whom helped me get to where I am today! —Darren Jefford

I’d like to thank Bob Elliot for getting me and Jean-Luc together at the outset, and Brian MacDonald for his professional and friendly editorial advice throughout. I’d also like to thank Simon Williams, a colleague who took no reward for looking through my very first chapter drafts before anyone else saw them. As always, I’d like to acknowledge my close family—Debs, Becky, and Matt. You don’t have to live with me; they do! —Tony Loton
Thanks and love go to my wife, Jessica, and our daughters, Deborah and Rachel, for their understanding and support throughout the creation of this book. Thanks to my family and friends for their understanding when I had to bow out of activities to research and write. Thanks to the Team System staff at Microsoft for their support and to the reviewers who helped to improve my chapters. Special thanks go to Jean-Luc David for inviting me to join in the creation of this book. Finally, I’d like to express thanks, love, and gratitude to my parents—Peter, my father, and Joan, my mother, in loving memory.

—Christopher Bowen
### Contents

About the Authors vii  
Acknowledgments ix  
Introduction xxv  

**Part One: Team Architect**  1  

**Chapter 1: Introducing the Visual Designers**  3  

| Why Design Visually? | 4  
|----------------------|---|  
| Microsoft’s Modeling Strategy | 5  
| Model-driven development | 5  
| Domain-specific languages | 6  
| Software Factories | 7  
| From Objects to Services | 8  
| Objects and compile-time reuse | 8  
| Components and deploy-time reuse | 9  
| Distributed components and run-time reuse | 10  
| Distributed services and the service-oriented architecture | 11  
| Visual Designers | 12  
| Application Designer | 13  
| Logical Datacenter Designer | 14  
| System Designer and Deployment Designer | 15  
| Class Designer | 16  
| Code synchronization | 17  
| Introducing the StockBroker Case Study | 18  
| Designer Relationships and Team System Integration | 19  
| Process flow (roles and activities) | 19  
| Object flow (artifacts) | 20  
| Settings and constraints | 21  
| Source control and item tracking | 21  
| Introducing the System Definition Model | 22  
| Meta-models | 22  
| SDM and the Team System meta-model | 22  
| What about UML? | 23  
| Capitalizing on your investment in UML skills | 23  
| Capitalizing on your investment in UML artifacts | 24  |
Defining a Logical Datacenter for the StockBroker Applications 65
  The MarketMaker Zone 65
  The StockMarket Zone 67
  The StockBroker Zone 68
  The Client Zone 70
Settings and Constraints 71
  Constraints versus settings 71
  GenericServer settings and constraints 72
  IISWebServer settings and constraints 72
  WebSiteEndpoint settings and constraints 74
  ZoneEndpoint settings and constraints 74
  Custom settings and user-defined constraints 75
  Import IIS Settings Wizard 76
Defining Your Own Reusable Prototypes 77
  StockBrokerZone prototype 77
  Team-working with prototypes 78
Summary 78

Chapter 4: Defining Systems and Evaluating Deployments
Using System Designer and Deployment Designer 81

Introducing System Designer and Deployment Designer 82
  Introducing System Designer 82
  Introducing Deployment Designer 82
  Toolbox and System View 82
Defining a Default Deployment from Application Designer 83
  The System View 85
  Binding applications to servers 85
  Differences between default deployments and system deployments 88
Defining and Deploying Systems 89
  System definition for the full application design 89
  System definition for a subset of the application design 89
  Defining deployment 90
  Nested systems 91
Creating System Diagrams from Scratch 92
System Settings and Constraints 92
Deployment Reports 93
  Human-readable deployment reports 94
  Machine-readable deployment reports and automated deployment 95
The UML Deployment Diagram: Similar, but Not the Same 97
  Dynamic Modeling 98
Summary 98
Contents

Chapter 5: Class Modeling Using Class Designer 101

Class Designer Toolbox and Types 102
From Code to Class Diagrams 103
StockBroker revisited — from code to class diagrams 104
Visualizing members 108
From Class Diagrams to Code 109
Designing the StockDeal classes using Class Designer 110
Working with methods 115
Working with other types 116
Working with interfaces 118
Reproducing class diagrams in hardcopy 119
Advanced Topics 120
Code synchronization 120
.NET Framework classes on Class Diagrams 121
Patterns 121
Building your own pattern library 122
Dynamic modeling 125
Comparison with UML Class Diagrams 128
Summary 129

Chapter 6: DSL Tools 131

Domain-Specific Languages 131
Creating a Designer 132
Creating the Project 133
Creating the Navigation Language 134
Creating the Graphical Language 141
Defining the shapes 142
Defining the connectors 143
Defining the toolbox 144
Providing a Shape Map 144
Providing a Connector Map 145
Using the Designer 147
Using the Domain Model 149
Generating Code 151
Designers and VSTS 154
Summary 156

Chapter 7: Dynamic Systems Initiative and the System Definition Model 157

Dynamic Systems Initiative 159
System Definition Model 160
SDM Architecture 160
Design-time validation 163
### Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDM in Visual Studio 2005 Team Edition for Software Architects</td>
<td>164</td>
</tr>
<tr>
<td>Deployment</td>
<td>165</td>
</tr>
<tr>
<td>Modeling layers of a system</td>
<td>166</td>
</tr>
<tr>
<td>SDM types</td>
<td>167</td>
</tr>
<tr>
<td>SDM schema structure</td>
<td>169</td>
</tr>
<tr>
<td>Definition complex type</td>
<td>170</td>
</tr>
<tr>
<td>ObjectDefinition complex type</td>
<td>172</td>
</tr>
<tr>
<td>RelationshipDefinition complex type</td>
<td>175</td>
</tr>
<tr>
<td>Member complex type</td>
<td>176</td>
</tr>
<tr>
<td>Settings</td>
<td>179</td>
</tr>
<tr>
<td>Constraints</td>
<td>182</td>
</tr>
<tr>
<td>SDM documents</td>
<td>186</td>
</tr>
<tr>
<td>Prototypes and the Distributed System Designers</td>
<td>189</td>
</tr>
<tr>
<td>The SDM SDK</td>
<td>190</td>
</tr>
<tr>
<td>the SDM command-line compiler</td>
<td>191</td>
</tr>
<tr>
<td><strong>Walkthrough: SDM Sample</strong></td>
<td>192</td>
</tr>
<tr>
<td>SystemDefinitions</td>
<td>193</td>
</tr>
<tr>
<td>Settings</td>
<td>195</td>
</tr>
<tr>
<td>Endpoints</td>
<td>196</td>
</tr>
<tr>
<td>Communication and delegation</td>
<td>200</td>
</tr>
<tr>
<td>Containment</td>
<td>201</td>
</tr>
<tr>
<td>Hosting</td>
<td>206</td>
</tr>
<tr>
<td>Flows</td>
<td>207</td>
</tr>
<tr>
<td>Constraints</td>
<td>208</td>
</tr>
<tr>
<td>Creating your own constraints</td>
<td>211</td>
</tr>
<tr>
<td>SdmaC</td>
<td>216</td>
</tr>
<tr>
<td>ProtoGen</td>
<td>216</td>
</tr>
<tr>
<td>The registry</td>
<td>217</td>
</tr>
<tr>
<td>Visual Studio</td>
<td>218</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td>219</td>
</tr>
</tbody>
</table>

**Part Two: Team Developer**

**Chapter 8: Managed Code Analysis**

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>The Need for Analysis Tools</strong></td>
<td>223</td>
</tr>
<tr>
<td><strong>Using Managed Code Analysis</strong></td>
<td>224</td>
</tr>
<tr>
<td>Built-in Managed Code Analysis rules</td>
<td>225</td>
</tr>
<tr>
<td>Enabling Managed Code Analysis</td>
<td>226</td>
</tr>
<tr>
<td>Executing static code analysis</td>
<td>227</td>
</tr>
<tr>
<td>Working with rule violations</td>
<td>230</td>
</tr>
</tbody>
</table>
## Contents

**Using the Command-Line Analysis Tool**  
FxCopCmd options  
FxCopCmd project files  
Build process code analysis integration

**Creating Code Analysis Rules**  
Reflection and Introspection  
Creating a new rule

**Summary**

---

**Chapter 9: Code Analysis for C/C++**

**Understanding Static Code Analysis**

**How the C/C++ Code Analyzer Works**

**Visual Studio 2005 Integration**  
Enabling and disabling C/C++ Code Analysis  
Setting warning levels in Visual Studio 2005  
Viewing code analysis warnings and errors

**Command-line Support**

**Annotation Support**  
Working with annotations in C++  
Working with annotations in C

**#pragma Support**

**Integrating with Team Foundation Server**  
Creating check-in policies using C/C++ code analysis  
Setting check-in notes for your C/C++ code  
Creating work items from code analysis warnings

**Identifying and Minimizing Noise**

**Tool Limitations**

**Summary**

---

**Chapter 10: Application Verifier**

**Setup and Configuration**

**Troubleshooting Heaps**

**Troubleshooting Handles**

**Troubleshooting Locks**

**Application Verifier End-To-End**

**Programming Application Verifier**

**Summary**
Chapter 11: Refactoring and Code Snippets

Refactoring from Class Diagrams
- Extract Interface 284
- Implement Abstract Class 287

Refactoring in Code
- Rename 288
- Encapsulate field 291
- Extract Method 294
- Promote local variable to parameter 296
- Reorder parameters 297
- Remove parameters 298
- Generate Method Stub 299

Improving Code Quality with Code Snippets
- Using code snippets in Visual Basic 300
- Using code snippets in Visual C# 302
- Code Snippets Manager 303

Summary 305

Chapter 12: Profiling and Performance

Introduction to Performance Analysis
- Types of profilers 307
- Team System profiling 308

Using the Team Developer Profiler
- Creating a sample application 309
- Creating a performance session 311
- Performance Explorer 313
- Configuring a sampling session 318
- Configuring an instrumentation session 319
- Executing a performance session 320
- Managing session reports 321
- Reading and interpreting session reports 321

Command-line Execution
- Configuring instrumentation 330
- Setting environment variables 331
- Executing a profiling session 332
- Generating reports 332

Modifying Instrumentation Programmatically 333

Common Profiling Issues
- Debugging symbols 335
- Instrumentation and code coverage 335
## Contents

**Virtual machines** 336  
**ASP.NET profiling** 336  
**Summary** 336  

### Part Three: Team Tester 337

#### Chapter 13: Test Case Management 339

- **Test Projects** 339  
  - Creating a test project 340  
  - Setting test project options 341  
- **Managing Test Cases** 342  
  - Test Manager 342  
  - Test View 348  
- **Working with Test Results** 349  
  - Test Results window 349  
  - Sorting, grouping, and filtering tests results 350  
  - Exporting test results 350  
  - Using Test Results XML (TRX) files 350  
  - Publishing results 351  
- **Using Ordered Tests** 352  
  - Creating an ordered test 352  
  - Ordered test properties 353  
  - Executing and analyzing ordered tests 354  
- **Summary** 355  

#### Chapter 14: Unit Testing with the Unit Test Framework 357

- **Unit Testing Concepts** 358  
  - Benefits of unit testing 358  
  - Writing effective unit tests 359  
  - Test-driven development 359  
  - Third-party tools 361  
- **Team System Unit Testing** 361  
  - Creating your first unit test 362  
  - Managing and running unit tests 364  
  - Test run configuration 366  
  - Test results 367  
  - Debugging unit tests 368  
- **Programming with the Unit Test Framework** 368  
  - Initialization and cleanup of unit tests 368  
  - Using the Assert methods 371  
  - Using the CollectionAssert class 374
## Contents

Using the StringAssert class 375
Expecting exceptions 376
Defining custom unit test properties 377
TestContext class 377
Creating data-driven unit tests 378

**Accessing Nonpublic Members from Tests** 379
- Using PrivateObject to access nonpublic instance members 379
- Using PrivateType to access nonpublic static members 381

**Code Generation** 382
- Generating tests from code 382
- Generating code from tests 386

**Unit Testing ASP.NET Applications** 387
- ASP.NET unit test attributes 387
- Creating ASP.NET unit tests 388

**Code Coverage** 389
- Enabling code coverage 390
- Viewing code coverage results 390

**Summary** 392

## Chapter 15: Web and Load Testing 393

### Web Tests 394
- Creating a sample web application 394
- Creating and configuring web tests 396
- Data-driven web tests 407
- Coded web tests 409

### Load Tests 411
- Creating and configuring load tests 411
- Editing load tests 419
- Executing load tests 422
- Viewing and interpreting load test results 422

### Command-line Test Execution 426
- Executing tests 426
- Executing test lists 426
- Other test options 427

### Distributed Load Tests 427
- Installing controllers and agents 427
- Configuring controllers 428
- Configuring agents 428
- Test run configuration 429
- Running a distributed load test 430
- Viewing a distributed load test 430

**Summary** 430
## Chapter 16: Manual Testing

### Test Automation versus Manual Testing
- Test automation
- Manual testing

### Test Cases

### Planning Your Manual Tests

### Manual Testing in Team System
- Creating manual tests
- Configuring manual test options
- Manual test templates
- Manual test structure

### Managing Manual Tests
- Using the Test Manager
- Logging your manual test as a bug work item
- Publishing the test results on Team Foundation Server

### Summary

## Chapter 17: Generic Testing

### Creating and Running a Generic Test
- Creating a generic test
- Running the generic test

### Creating an External Testing Tool
- Scripting Host example
- Managed Code example

### Extended Return Results
- Summary report XML schema
- Example summary report XML file

### Wiring to Existing Test Systems
- Creating a wire to NUnit
- JScript conversion code

### Summary

## Part Four: Team Foundation

## Chapter 18: Team Foundation Architecture

### What Is Team Foundation Server?
- Team processes
- Microsoft Solutions Framework (MSF)
- Project management guidance
- Team Foundation features
Contents

Version control 484
Work item tracking 487
Team Project Portal 488
Team Foundation Build 488
Team Foundation Reporting 488
Team Foundation core services 489
Team Edition Test Load Agent 489

Team Foundation Architecture 489
Clients of Team Foundation 490
Team Explorer 490
Team Foundation Build 494
Team Foundation Reporting Warehouse architecture 494
Configuration 496

Summary 497

Chapter 19: Project Management Tools and Team Reporting 499

Creating a Software Project 500
Why software development projects fail 501
Using the New Team Project Wizard 502

Administering Project Details 505
Connecting to the Team Foundation Server 505
Configuring Project Settings 507
Project Alerts 511
Team Portal customization and extensibility 513
Deleting a Team Project 514

Work Item Tracking 516
Anatomy of a Work Item 516
Creating Work Items Using Visual Studio 517
Creating and running work item queries 518
Creating and assigning work items using Microsoft Office 519
Creating and customizing work item types 522

Team Foundation Server Reporting 524
Working with Team Reports 524
Viewing reports using Microsoft Excel pivot tables 529
Building custom reports 530
Report site security 533

Summary 534

Chapter 20: Team Foundation Version Control 535

Comparing Team Foundation Source Control and Visual SourceSafe (VSS) 2005 536
Source code migration 540
**Setting Up Version Control** 541
- Setting up security roles 542
- Setting up your workspace 543

**Using the Source Control Explorer** 544
- Workspaces 546
- Adding Projects to the Source Repository 548

**Check-in and Check-out** 549
- Checking in an item 549
- Checking out an Item 550
- Creating and administering check-in policies 551

**Branching and Merging** 553
- Branching 554
- Merging 555

**Shelving** 557

**Command-Line Tools** 560

**Summary** 561

---

**Chapter 21: Microsoft Solutions Framework** 563

**About the Microsoft Solutions Framework** 564
- A conceptual overview of MSF 565
- Obtaining the Microsoft Solutions Framework 566

**MSF for Agile Software Development** 566
- About Agile development 568

**MSF for CMMI Process Improvement** 569
- Capability Level 3 571
- CMMI Level 3 vs. MSF for CMMI process improvement 571

**Choosing the Right Process** 573

**Fundamental Concepts** 574
- Process template components 574
- Iterative process 575
- Cycles 575
- Personas 576
- Governance, tracks, and activities 577
- Work item types 578
- Reports 582

**Team Model** 582
- Advocacy 584
- Mindsets 585

**Roles** 585
- Business analyst 586
- Project manager 587
- Architect 587
### Contents

- Developer 588
- Tester 588
- Release manager 589
- User experience 589

**Workstreams** 589

**Summary** 590

---

**Chapter 22: Process Templates** 591

- **New Team Project Wizard** 592
- **Customization Tools** 592
  - XML/XSL editors 593
  - Microsoft InfoPath 2003 593
  - MSF HTML Build utility 594
  - Third-party tools 594
- **Customizing Process Guidance** 595
  - Anatomy of process guidance (ProcessGuidance.xml) 595
  - Creating workstreams and activities 596
  - Creating work item documentation 599
  - Creating custom roles 600
  - Cross-references 600
  - Viewing your custom guidance 601
- **Customizing Process Templates** 601
  - Downloading a process template to your desktop 601
  - Navigating a process template 602
  - Modifying template definitions 603
  - Uploading process templates in Team Foundation Server 622
  - Deleting process templates 623
- **Testing Your Custom Process Template** 623
- **Summary** 623

---

**Chapter 23: Team Foundation Build** 625

- **Setting Up Team Foundation Build** 627
  - Planning a build strategy 628
  - Setting up your own build lab 629
  - Team System integration 629
  - Setting up build security 630
- **Using Team Foundation Build** 630
  - Creating build types 630
  - E-mail notifications 635
  - Executing a local build 636
  - Getting the Team Foundation Build status 637
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command-line interface</td>
<td>638</td>
</tr>
<tr>
<td>Starting a build</td>
<td>638</td>
</tr>
<tr>
<td>Deleting a build</td>
<td>638</td>
</tr>
<tr>
<td>Stopping a build</td>
<td>639</td>
</tr>
<tr>
<td>Setting up a nightly build</td>
<td>639</td>
</tr>
<tr>
<td>Continuous integration (CI)</td>
<td>640</td>
</tr>
<tr>
<td>Team Foundation Build Reporting</td>
<td>641</td>
</tr>
<tr>
<td>Build Customization and Extensibility</td>
<td>644</td>
</tr>
<tr>
<td>Creating MSBuild tasks</td>
<td>645</td>
</tr>
<tr>
<td>Summary</td>
<td>646</td>
</tr>
<tr>
<td>Chapter 24: Deployment and Administration</td>
<td>647</td>
</tr>
<tr>
<td>Setting Up Your Operations Team</td>
<td>648</td>
</tr>
<tr>
<td>Planning a Deployment</td>
<td>649</td>
</tr>
<tr>
<td>Unsupported deployment scenarios</td>
<td>650</td>
</tr>
<tr>
<td>Capacity planning</td>
<td>650</td>
</tr>
<tr>
<td>User accounts</td>
<td>651</td>
</tr>
<tr>
<td>Firewall configuration</td>
<td>652</td>
</tr>
<tr>
<td>Network topologies</td>
<td>652</td>
</tr>
<tr>
<td>Creating a test plan</td>
<td>654</td>
</tr>
<tr>
<td>Test rig considerations</td>
<td>655</td>
</tr>
<tr>
<td>Considerations for Migrating and Integrating Your Existing Tools</td>
<td>655</td>
</tr>
<tr>
<td>Version control</td>
<td>656</td>
</tr>
<tr>
<td>Work-item tracking</td>
<td>657</td>
</tr>
<tr>
<td>Reporting</td>
<td>657</td>
</tr>
<tr>
<td>Testing tools</td>
<td>657</td>
</tr>
<tr>
<td>Backup and Recovery</td>
<td>658</td>
</tr>
<tr>
<td>Licensing Models</td>
<td>659</td>
</tr>
<tr>
<td>Where to get Team System</td>
<td>659</td>
</tr>
<tr>
<td>Summary</td>
<td>660</td>
</tr>
<tr>
<td>Index</td>
<td>661</td>
</tr>
</tbody>
</table>
In June of 1999, Microsoft started to reevaluate how Visual Studio was being used as part of a software development process. Microsoft’s Developer Division sustained the developer community at large for years, but did not adequately address all of the challenges of software development within a team environment. For example, what Microsoft tools were out there for architects before Team System? There really wasn’t that much—the most notable tool is Microsoft Visio, which is used to design UML diagrams. There are also some other lesser known design tools in Visio. However, there isn’t any solid two-way integration between Visio diagrams and code within Visual Studio. Another problem relates to the tools themselves. To effectively set up a software development shop, you have to purchase several third-party tools (or use open-source tools) to handle tasks such as version control and team communications. A mishmash of tools is even more difficult to integrate.

Here is a common scenario: You are compiling a list of tasks or requirements to send out to your team using products such as Microsoft Excel, Word, or Project. You then have to be able to communicate these tasks to the rest of your team and respond when they have completed the task or the status changes. To be able to achieve this, you have to do a lot of cutting and pasting. On top of that, the process is very deliberate—there is no automation or shortcuts. You also have to handle a lot of your communication by e-mail. The problem with e-mail is that your team members can lose messages; there may be a server outage that may prevent your messages from coming through, and so forth.

Before Team System, the only version control product Microsoft had to offer was Microsoft Visual SourceSafe. SourceSafe is an excellent tool for a single developer or a very small team, but it doesn’t scale very well if you have a large number of developers working concurrently. In addition, it isn’t really designed to be used over the Web.

Visual Studio’s Enterprise Tools team launched a project code-named “Burton” to create a team product that would integrate well with Visual Studio and tackle the challenges of integration and scalability.

“Burton” is named after a very popular snowboard and clothing company based in Vermont. Many Microsoft employees enjoy skiing in their free time and as a result, many of the product code names are based on skiing-related locations and equipment. For example, Windows XP was code-named after Whistler Mountain in British Columbia.

In May of 2004, after five years of research, planning, and development, Microsoft announced and presented Visual Studio 2005 Team System to the public. Team System consists of experience and tools used internally at Microsoft for years, such as PREfast, Product Studio, and Source Depot. Rick LaPlante (the general manager for Team System) delivered a keynote presentation and an end-to-end demo that changed the way people looked at Visual Studio. The first major change in the Visual Studio product is the availability of tools targeted at all roles within an IT organization: architects, developers, testers, and project managers. Another big focus is on the software development life cycle (also known as the SDLC). Within the product, you can now use the Microsoft Solutions Framework, which not only provides a framework for developing software, but also includes process guidance and integration.
Fast forward to today: Team System is now in stores and you’ve picked up a copy of this book to learn more about it. Professional Visual Studio 2005 Team System will provide you with an end-to-end overview of the important features of the product. In the Wrox spirit, we have included a lot of practical information and hands-on walk-throughs to put Team System through its paces, regardless of your role within your organization.

One of the most important aspects of Team System is that it is designed to fit your needs and not the other way around. Unlike other Microsoft products such as Office, Team System isn’t designed for out-of-the-box use. You have to spend some time configuring it and applying your policies and best practices—in essence, tailoring it to the way you develop software. Why make changes if you have established a good development process and everything is working well? Alternatively, if your approach is more ad hoc, Team System will provide you with a good framework to work out a suitable and workable process.

We will make no bones about it—Team System is a huge product. Think of this book as your starting point into Team System. Like other Professional books by Wrox, we not only dive into the important features, but also the grittier details.

Team System is composed of several logical tiers. Figure 1 features a very high-level logical overview of Team System’s architecture.

![Visual Studio 2005 Team System Diagram](image)

Following is a detailed explanation of each of the tiers included in Team System.
**Client tier**

The client tier is composed of several role-based versions of Visual Studio 2005 and other tools that are designed to connect to the application and data tiers of Team System. Here is a description of the four Team System client products:

- **Visual Studio 2005 Team Suite** incorporates the functionality of Visual Studio 2005 Team Edition for Software Architects, Visual Studio 2005 Team Edition for Software Developers, and Visual Studio 2005 Team Edition for Software Testers all bundled into one. This product is geared toward team members who want (or need) to take on several roles. The Team Suite product is also useful for installing on the Team Foundation Build server to incorporate the test capabilities of both the Developer and Tester versions of the product.

- **Visual Studio 2005 Team Edition for Software Architects** includes tools to support the application and infrastructure architects during the initial design phase and ongoing development of a software project. It includes several designers, such as the Application Modeling, Infrastructure Modeling, and Deployment Modeling tools. This tool should not be confused with the Visual Studio .NET 2003 Enterprise Architect product. The older Enterprise Architect tool was considered the most complete version of Visual Studio .NET 2003. The “equivalent” tool for Team System is the Visual Studio 2005 Team Suite.

- **Visual Studio 2005 Team Edition for Software Developers** has Static Code Analyzers for both managed and unmanaged code, Dynamic Code Analyzers (also commonly known as AppVerifier), Unit Testing, Coverage, and Code Profiling tools.

  *If you want to enforce static code analysis or dynamic code analysis on your build, you have to add these tests to a test task and make sure that the Team Edition for Software Developers version is installed on your build machine (or you can alternatively use Team Suite).*

- **Visual Studio 2005 Team Edition for Software Testers** has features such as Test Case Management, Performance and Load Testing, Unit Testing, Code Coverage, and Manual Testing. If you want to enable any of these tests in your builds, you have to install this version of Visual Studio 2005 on your build server (or Team Suite).

Once you are finished installing Visual Studio, you need to install the Team Explorer. Team Explorer is a client plug-in that will enable you to connect to Team Foundation Server, create projects, and administer all of the project details. Team Explorer can also be installed as a standalone tool. It has the same “look and feel” as Visual Studio, and it has a light memory footprint. Note that you can’t write code within the standalone version of this tool (you still need Visual Studio to be able to do that). You can also use it for source control operations side by side with Visual Studio .NET and other development tools. It’s also appropriate for project managers.

Although not officially part of Team System, Microsoft Office Professional 2003 (specifically Word and Excel), and Microsoft Project are used to interface with both the application tier and the data tier. Excel and Project can be used to manage work items and requirements; Word is used to create manual tests. Excel can be used to get a nice view of your project data by connecting directly to the data tier using pivot tables.

Chapters 1 through 17 cover most of the client-based tools. If you are interested in project management and reporting, you should definitely take a look at Chapter 19.
**Application tier**

Team Foundation Server incorporates the application and data tiers. This server has many integrated and centralized functions, including project management capabilities (a SharePoint-based Team Project Portal site, work item tracking, and reporting), build automation, and a version control system. Team Foundation Server also provides support for fine-grained security, policies, and other administrative capabilities. Chapters 18–24 cover each feature in detail.

One of the topics we won’t be covering in detail is the Team Foundation Core Services. The reason we are excluding this information is because the programmability features and API are very much in flux at the time of writing. For current documentation and samples, we highly recommend that you visit the Visual Studio Extensibility Center website at www.vsipdev.com.

**Data tier**

Team System’s data tier is based on Microsoft SQL Server 2005. This product was specifically selected to reduce the complexity of the application and speed up time to market. At this point, Team System is not compatible with third-party data sources and does not integrate with Enterprise Data Services (EDS). Team System’s data tier is used to store all of Team System’s data, such as source code, work items, reporting data, and project settings. You can find coverage on the data warehouse in Chapter 19. You can also find out about how to back up and recover your data store (and other data-related topics) in Chapter 24.

**Modern Software Development Challenges**

Software developers share common challenges, regardless of the size of your team. Businesses require a high degree of accountability; software has to be developed in the least amount of time with no room for failure. Some of these challenges include the following:

- **Integration problems.** Prior to Team System, most of the tools you commonly use to operate software development teams come from third-party vendors. Integrating these tools is a big challenge—in many cases, to make it all work you have to repeatedly copy project data into different software packages. 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 can be mapped incorrectly, causing delays and introducing errors. Global teams require solid design, process, and software configuration management all integrated in one. There aren’t many software packages that can deliver all of these features, and those that do exist tend to be inordinately expensive.

- **Segmentation of roles.** Specialization can be a huge problem on a team. Experts can assume that other departments are aware of information, which doesn’t end up in the status reports but may greatly affect the project as a whole. Interdepartmental communications is a huge and prevalent problem.