



Index

A

- Account setup, 724–725
- Action fixtures, 470–476
- Action tables, 498
- Active Bugs queries, 670
- Activity diagrams, 414
- Administrator rights
 - ClickOnce, 620
 - InstallShield, 606
 - Software installation, 619
 - Windows Installer, 603
- Agile Alliance, 36–37
- Agile Model-Driven Development (AMDD), 361, 372–374
- Agile Modeling, 357, 361
 - practices, 364–365
 - principles, 363–364
 - values, 363
- Agile teams, 65
 - building and integrating by, 214
 - inappropriate work for, 69–70
 - nature of, 65–66
 - roles
 - associated, 75–76
 - customer, 70–72
 - developer, 72–73
 - self-organizing, 68
 - size, 68–69
- Agile values
 - Agile Alliance, 36–37
 - tools and values, 33–35
- Alexander, Christopher, 416
- Ambler, Scott
 - Agile Modeling, 363, 366
 - AMDD, 372
 - Coding Guidelines, 132, 195
 - Refactoring Databases, 592
 - UML, 392
- AMDD (Agile Model-Driven Development), 361, 372–374
- Analysis Class diagrams, 399–401
- Analysis Patterns, 419–420
- Appleton, Brad, 144
- Application Designer (AD) tool, 408, 628, 638
 - application of, 648
 - description, 26
 - diagrams
 - creating, 638–647
 - System Designer diagrams from, 649–651
 - settings and constraints, 648
- Applications architect role, 75
- Apply Modeling Standards practice, 366
- Apply Patterns Gently practice, 366, 422
- Architects
 - on Agile teams, 406
 - developers as, 72–73
 - PC licensing issues, 745–746
- Architectural models, 368, 405–406
 - evolving, 408–410
 - skeletal, 406–407
 - system metaphor for, 411
- Associated roles, 75–76

Atherton, James, 69
Atomic check-in, 134

B

Backups
 shared folders for, 128
 Team Foundation Server version control, 170–171

Bad smells in code, 422

Baselines, 160
 checking in and labeling, 164–166
 configuration management, 142–143

BDUF. *See* Big Design Up Front

Beck, Kent, 37, 64
 on code duplication, 267
 on code problems, 422
 on costs, 541
 on Domain Modeling, 396
 on pattern languages, 416
 on planning, 530
 on software design, 348

Berczuk, Stephen P., 144

Big-bang deployment, 590

Big Design Up Front (BDUF) thinking, 63, 348, 369, 372

Big Visible Charts (BVCs), 673–674

Binder, Robert
 on test cases, 270, 279
 on Testing Object-Oriented Systems, 221

BisSubscribe tool, 246

Black-box testing, 219–220

Booch, Grady, 285, 377

Bottom up approach
 tasks planning, 567
 test-driven development, 268

Box, Don, 160

Branching in version control, 138–139, 146, 185–187

Brooks, Frederick, 34, 521

Brown-bag sessions, 568

Budgets
 in plan control, 539
 in project life cycle, 58
 in story estimates, 520–521

Bugs
 queries, 670–671
 reports, 554–555, 594
 tasks, 532
 work items, 96, 548

Build Labs, 213
 customer tests in, 482–486
 licensing issues, 747–748

Build Validation Tests (BVTs), 227, 233–235, 465, 481

Builds
 managing, 245
 identification, names, 247–248
 notifications, 246
 reports, 248–249, 671
 notes for, 158
 daily builds, 227–228
 integration builds, 225–227
 local builds, 222–223
 test cycles, 222–228

Build Drop folder, 215

Business analysts role, 71

Buy or build decisions for tools, 34

BVCs. *See* Big Visible Charts

BVTs. *See* Build Validation Tests

C

Capability Maturity Model Integration (CMMI), 100–101, 103–104

Changesets, 134–135

Check in, 130
 constraints, 193–194
 policies
 overriding, 201–203
 static code analysis as, 198–200
 in version control systems, 134
 for work items and builds, 146
 Items settings, 169

Check-out, 129
 in version control systems, 129
 only files, 192–193

Class Designer tool, 23
 language workbenches, 428
 for modeling, 370, 385–389

Class diagrams, 156
 Class Designer for, 388
 for structural models, 412
 UML, 377–382
 Visio for Enterprise Architects for, 389

Classes
 in CRC, 397
 dependencies, 251–253
 names, 198, 400–401
 testing, 222
 for user interface design, 347

- ClickOnce technology, 620
 - basic concepts, 621
 - publishing and deploying, 622–624
 - suitable applications for, 620–621
- Client parts of VSTS, 17
 - Team Explorer window, 20–22
 - Visual Studio Professional, 18–19
- Coaching
 - in Agile teams, 75
 - in storytest-driven development, 457
- Cockburn, Alistair, 404
- Code
 - branching, 138–139, 146, 185–187
 - coverage, 325–330
 - duplication, 140
 - patterns in, 420
 - standards, 132, 195, 200–201
- Code Analyzer tool, 331
- Code and Tests practice, 155–156
- Code Coverage tool, 24–25, 326–330
- Coding Guidelines, 132, 195
- Cohn, Mike, 515, 519
- Collaborations
 - CRC, 398–399
 - pair programming, 38
 - requirements gathering, 619
- Column fixtures, 460–464
- Communication
 - as Agile value
 - Agile development, 121
 - Extreme Programming, 39–40
- Component-based development, 425
- Component diagrams
 - in architectural models, 409
 - in structural models, 412
- Components
 - distributed, 626
 - factories for, 429–430
 - Installation Designer view settings, 609–610
 - Reusable, 425, 429–430
 - in Windows Installer, 598–599, 601
- Conflicts in version control, 182–183
- Connecting to team projects, 93–94
- Constantine, Larry, 39, 402
- Constraints
 - Application Designer, 648
 - check-in, 193–194
 - Logical Datacenter Designer, 634–635
 - System Designer, 652
- Consultant role, 75
- Consumer endpoints, 641–642
- Continuous integration, 134, 356
- Continuous Integration practice, 132, 224
- Contributors, 82
- Cost estimation for customer stories, 519–521
- Courage as Agile value
 - Agile development, 122
 - Extreme Programming, 41
- CRC (Class, Responsibilities, and Collaborations), 396
- Create Several Models in Parallel practice, 412
- Cross-functional teams, 67
- CruiseControl.NET tool, 224
- Crystal Clear, 404
- Cunningham, Ward
 - and CRC, 396
 - and Extreme Programming, 37
 - and FIT, 445, 478
 - on pattern languages, 416
- Customer stories, 404–405, 511–512
 - completing, 563
 - estimating, 516–521
 - in Extreme Programming, 59–61, 63
 - generating, 514–515
 - life cycle, 541–542
 - overview, 512
 - prioritizing, 521–525
 - for specifying requirements, 55–57
 - Stories practice, 513–514
 - tasks in, 288–289
 - for Test-First Programming, 287–289
- Customer Acceptance tests, 439–440
 - automated, 487–491
 - in Build Lab, 482–486
 - FIT. *See* Framework for Integrated Test (FIT)
 - fixtures in
 - action, 472–476
 - column, 461–464
 - row, 466–470
 - introducing to teams, 491–498
 - overview, 443
 - planning, 562
 - storytest-driven development, 454–458

- Customers
 - in Agile teams, 70–72
 - in Agile software development, 36
 - in Six Sigma, 691
 - identifying, 76
 - pulling value, 686
- CVS version control systems, 133
- D**
- Daily Builds, 224, 227–228
 - in customer testing, 457
 - in project management, 566
 - Team Foundation Builds, 239–240
- Daily Deployment practice, 583–585
- Data Definition Language (DDL) scripts, 390
- Data deployment, 591–592
- Data migration, 583
- Databases
 - connection strings to, 643
 - Model diagrams, 390
 - with Windows Installer, 600–602
- Death-march projects, 508
- Deleting
 - build products, 243–244
 - team projects, 87
- Deltas, 135
- DeMarco, Tom, 41, 73
- Dependencies
 - in building, 251–253
 - in class diagrams, 379
 - in prioritizing stories, 524–525
- Depict Models Simply practice, 367
- Deployment, 577–578
 - with ClickOnce technology, 622–624
 - data, 591–592
 - first iteration, 589
 - Incremental Deployment practice, 590–591
 - Installation programs. *See* Installation programs
 - bottlenecks in, 585–586
 - Daily Deployment practice, 583–585
 - deployment teams in, 586–587
 - release process, 582–584
 - monitoring, 592–594
 - preparing for, 587–592
 - stubs and scaffolding in, 590–591
- Deployment Designer (DD) tool, 628, 653
 - application of, 658
 - description, 26
 - diagrams, 653–657
 - properties, 655
- Derby, Esther, 697
- Design
 - Agile teams for, 66–68
 - for building and integrating, 214
 - for patterns, 420
 - for Six Sigma, 691–692
 - Patterns, 416–417, 420, 423
- Developers
 - in Agile teams, 72–73
 - identifying, 76
 - licensing issues, 743–745
- Development phases, 54–55
- Development environment, 215
- Diagrams
 - Analysis Class, 399–401
 - Application Designer, 638–647
 - class, 156
 - Class Designer for, 388
 - for structural models, 412
 - UML, 377–382
 - Visio for Enterprise Architects for, 389
 - class package, 409
 - Deployment Designer, 653–657
 - for models
 - dynamic, 413–414
 - free-form, 375–376, 407–410
 - structural, 412–413
 - tips, 391–392
 - UML, 377–385
 - System Designer, 649–652
- Dim files
 - preparing, 614–617
 - working with, 618
- Directories
 - adding to version control, 160–164
 - mapping, 154, 177–178, 180
 - in Windows Installer, 601
- Discard Temporary Models practice, 366, 368
- Distributed System Designers (DSDs), 628–629
- Distributed systems, 625
 - architecture, 625–629
 - Deployment Designer, 653–658
- DLL Hell, 160
- Documents
 - in project management, 551–554
 - in Team Explorer window, 20
 - version control for, 156–158

- Documents folder, 156–158, 552
- Domain Modeling, 396–401
- Domain-Specific Languages (DSLs), 66, 192
 - language workbenches for, 427–429
 - software factories, 429–430
- Drop Site folder, 216, 243
- DSDM (Dynamic Systems Development Method), 522
- DSDs (Distributed System Designers), 628–629
- DSI (Dynamic Systems Initiative), 627
- Dynamic Code Analyzer tool, 24, 331, 334–337
- Dynamic models, 413–414
- Dynamic Scan tool, 610

E

- Eclipse support, 147
- Economics of software development, 688–690
- Elective process, 102
- End user role, 71
- Energized Work practice, 323–324
- Enterprise architect role, 75
- Estimating, 507–508
 - in project management, 567
 - stories, 515–516
 - absolute vs. relative values, 517–518
 - budgeting in, 521
 - relative estimate scales, 519
 - size, 516–517
 - task points for, 519–520
- Evolving
 - architectural models, 408–410
 - legacy code, 422–424
- Executive role in Agile teams, 75
- Exporting process templates, 112–113
- Extending VSTS, 30–31
- Extracting data from Team Foundation Server, 674–682
- Extreme Programming (XP), 37, 57–58
 - Agile values in
 - communication, 39–40
 - courage, 41
 - feedback, 40–41
 - respect, 43
 - simplicity, 42–43
 - project life cycle in, 58
 - transition to, 709–711

F

- Facade Pattern
 - evolving legacy code with, 422–424
 - information for, 417–419
- Facilitators for retrospectives, 698
- Failing tests
 - fixing, 296–299
 - writing, 272–273
- Features
 - vs. tasks, 514, 539, 567
 - Windows Installer, 598–601
- Feedback
 - in Agile, 55
 - as Agile value
 - Agile development, 121
 - Extreme Programming, 40–41
- Files
 - access to, 192–193
 - adding to version control, 160–164
- First iteration, 589
- FIT. *See* Framework for Integrated Test (FIT)
- FitNesse IDE, 454
- FLEXnet AdminStudio, 587
- Folders
 - access to, 192–193
 - for repositories, 150–151
 - shared, 128–129
- Forward-engineering, 647
- Fowler, Martin
 - and Extreme Programming, 37
 - on language workbenches, 427–428
 - on patterns, 415, 419–420, 422
 - on refactoring, 280–281, 312, 424
 - Remove the Middle Man pattern, 318
 - UML Distilled*, 377, 392
- Framework for Integrated Test (FIT)
 - fixtures, 459
 - action, 470–476
 - column, 460–464
 - custom, 476–479
 - row, 465–470
 - standard, 459–460
 - installing and running, 447–453
 - overview, 445–447
 - with storytest-driven development, 454–458
 - with Team Foundation Build, 481
 - automated customer tests, 487–491
 - customer test introduction, 491–498
 - customer tests in Build Lab, 482–486
 - test organization, 453–454

- Free-form diagrams
 - for models, 375–376
 - for skeletal architectural models, 407–410
- Friendly Assemblies, 285
- Functional components, 16
- Functional tests, 219–220, 447
- FxCop program, 196, 200

G

- Gamma, Erich, 37, 416
- Generalizing specialists, 68
- Generating business value, 683
 - lean thinking in, 683–688
 - linking to other process improvement initiatives, 690–692
 - software development economics, 688–690
- Generic tests
 - Adapter tool, 25
 - with MSTest, 486
 - wrapping FIT in, 482–485
- Glass box testing, 220
- Gold-plating code, 684–685
- Groups
 - for modeling, 366–369
 - for projects, 82
- Guckenheimer, Sam
 - on bugs, 532, 555
 - and Extreme Programming, 58
 - on release process, 458
 - on reports, 682
 - on value, 522

H

- Hacking vs. Extreme Programming, 62–63
- Helm, Richard, 416
- Hierarchy of iterations, 547
- Horizontal markets, 426–427
- Hot-desk area, 756–758
- Humility, 363

I

- Ideal days, 525
- Identification of builds, 247–248
- IDEs (Integrated Development Environments), 16
 - Visual Studio, 18
 - InstallShield, 605–613

- Idioms, 420
- Implementation models, 411–412
 - dynamic, 413–414
 - sequence diagrams in, 382
 - structural, 412–413
- Importing
 - datacenter settings, 636–637
 - process templates, 114
 - source files, 166
- Incremental builds, 245, 250–251
- Incremental Deployment practice, 590–591
- Incremental Design practice, 371–372
- Individuals
 - in Agile software development, 36
 - with Six Sigma, 692
 - Individuality, 760
- Informative Workspace practice, 673
- Infrastructure architect role, 75
- Installation Designer view, 604, 608–611
- Installation programs, 597
 - ClickOnce technology, 620
 - basic concepts, 621
 - publishing and deploying, 622–624
 - suitable applications for, 620–621
 - developing, 613–614
 - Dim files for, 617–618
 - InstallShield, 604–613
 - Windows Installer, 597–598
 - basic concepts, 598–600
 - operation, 600–602
 - security for, 602–604
- Installer role, 71
- Installing third-party libraries, 158–159
- InstallShield, 605–613
- InstallShield Collaboration tool, 614–617
- Instrumentation for performance, 331–332
- Integration. *See also* Building and integrating
 - integrating
- Interaction designer role, 71
- INVEST acronym, 515
- Iterate to Another Artifact practice, 412
- Iteration Burn rate, 520–521
- Iteration Planning Boards, 561
- Iteration zero
 - AMDD in, 373
 - on timelines, 705
 - version control in, 171

- Iterations
 - in Agile Modeling, 365
 - in Extreme Programming
 - and increments, 58–59
 - productional-quality code from, 60–61
 - and release cycles, 59–60
 - fixed periods for, 514
 - planning life cycle, 556–558, 565–566
 - plans, 534–535
 - in project management, 568
 - in project structure, 546–547
- J**
- Jacobson, Ivar, 57, 404
- Jeffries, Ron
 - on customer stories, 56
 - and Extreme Programming, 37
- Johnson, Ralph, 416
- Jones, Allen, 420
- Jones, Daniel, 683
- K**
- Kerievsky, Joshua, 281, 420, 423
- Kerth, Norman, 697
- L**
- Labels
 - baselines, 164–166
 - favorites, 315
 - version control, 138–139, 146, 184–185
- Language workbenches, 427–429
- Languages, pattern, 416–417
- Larsen, Diana, 697
- Lean thinking, 36, 683–684
 - customs in, 686
 - perfection seeking in, 686–687
 - specifying value in, 684
 - value flow in, 685–686
 - value stream in, 684–685
- Legacy code, evolving, 422–424
- Libraries
 - installing, 158–159
 - performance analysis, 335–337
- Licensing issues
 - architect PC, 745–746
 - BuildLab PC, 747–748
 - developer PCs, 743–745
 - multiprocessor PCs and multicore processors, 749
 - pair programming, 745
 - Primary Domain Controller, 739–740
 - standby TFS, 748
 - Team Foundation Server, 740–742
 - team size, 749–750
 - tester PC, 746–747
- Life cycles
 - customer stories, 541–542
 - planning. *See* Planning life cycle
- Lists
 - for modeling in pairs, 371
 - of tests, 454
 - in test-driven development, 269
 - in Test-First Programming, 278
- Load Testing tool, 25
- Local builds, 185, 222–223, 232
- Lock and Merge feature, 136–138, 146
- Locking graphics files, 183
- Lockwood, Lucy, 402
- Logical Data Model (LDM) diagrams, 413
- Logical Datacenter Designer (LDD) tool, 628–629
 - applications of, 637–638
 - description, 26
 - diagrams, 633
 - properties, settings, and constraints for, 634–637
 - working with, 629–632
- M**
- Manifesto of the Agile Alliance, 529
- Manual Testing tool, 25
- Mapping directories, 154, 177–178, 180
- MDA (Model-Driven Architecture), 372
- Mentor role, 73
- Merging in version control, 136–138, 167, 180–183
- Metaphors for architectural models, 411
- Metrics for process frameworks, 107–110
- Microsoft Solutions Framework (MSF), 95
 - activities in, 97
 - MSF for Agile, 85–86, 95, 102–104
 - MSF for CMMI, 95, 100–101, 103–104
 - MSF for XP, 102–104
 - roles in, 97
 - tracks and governance checkpoints in, 98–100
 - work items in, 96–97
- Mining for gold, 701, 703–704, 708–709

Model-Driven Architecture (MDA), 372
Models, 357

Agile. *See* Agile Modeling (AG)

architectural, 368, 405–406

evolving, 408–410

skeletal, 406–407

system metaphor for, 411

Class Designer for, 385–389

Domain Modeling, 396–401

free-form diagrams for, 375–376

implementation, 411–412

dynamic, 413–414

sequence diagrams, 382

structural, 412–413

introduction, 361–362

UML diagrams, 377

class, 377–382

sequence, 382–385

use case, 403–404

user interface, 401–403

Visio for Enterprise Architects for,
389–390

Moore, Geoffrey, 14

MoSCoW rules, 522

Mothballing projects, 61

MSBuild engine, 229–232

MSF. *See* Microsoft Solutions Framework
(MSF)

MSF4XP process template, 551

MSTest, 486

Mugridge, Rick, 478

Multiple-checkout approach, 136–138, 167

Multiplicity indicators, 380

Myers, Glen, 279

N

Namespaces for stubs and scaffolding, 591

Naming Rules list, 199

Network diagrams, 409–410

Notes

for builds, 158

in class diagrams, 380

in Document folder, 157

Notifications, build, 246

Non-functional requirement, 220

Nouns

for class names, 401

in CRC, 397

Noyes, Brain, 622

NUnit tool, 24, 267

O

Object diagrams, 412

Object Management Group, 377

Object Modeling Technique (OMT), 377

Object Test Bench tool, 388

Objects in sequence diagrams, 383–384

Office space and layout, 753

OLAP database, 675, 677–678

OMT (Object Modeling Technique), 377

One-click installs, 619–624

One-time plans, 528

Open and Honest Communication principle,
366

Optimistic locking, 136–138

Ordered Test Adapter tool, 25

OSPACS team, 1

background, 1–2

organizational structure and personas,
2–5

road map for, 6–8

Overriding check-in policies, 201–203

P

Package dependencies, 251–253

Package diagrams, 413

Pair programming practice, 38

Pair programming, 37–39, 43

in Agile Modeling, 370–371

desks for, 755

for implementation models, 411–412

licensing issues, 745

in Shared Code practice, 132

in Test-Driven Development, 356

Parallel models, 365

Pascal, Blaise, 348

Patch Packages (.msp), 602

PATH environment variable, 449, 723

Pattern-happy condition, 421

Patterns, 415

Domain-Specific Languages for, 426–430

example, 417–419

implementation, 424–426

languages, 416–417

sources, 419–421

working with, 421–424

Patterns & Practices initiative, 593

Performance analysis, 331

build configuration for, 335

example profiling session, 332–334

instrumentation for, 332

- sampling for, 332
 - system performance, 337
- Permissions
- build, 92
 - files and folders, 192–193
 - projects, 82
 - Team Foundation Build, 235–236
 - workspaces, 92–93
- Phased development processes, 54–55
- Plans and planning, 507–508, 527
- controlling, 538–542
 - customer tests, 562
 - planning life cycle, 556
 - nature of, 527–530
 - rate of progress measures, 529–530
 - repeated execution vs. one-time, 528
 - time scales in, 530–538
- Plug-ins
- process templates, 111–112
 - Visual Studio source control selections, 168
- Policies for source code
- coding standards in, 195
 - overriding, 201–203
 - rules updating for, 201
 - static code analysis for, 196–200
- Poppendieck, Mary, 528–529
- Practices in Agile Modeling, 364–365
- Preproduction Environment, 585
- Principles in Agile Modeling, 363–364
- Prioritizing, 507–508
- stories, 521–522, 567
 - business risk in, 522–523
 - dependencies in, 524–525
 - technical risk in, 523
 - value in, 522
- Problem analysis, 687–688
- Process frameworks, 107
- description, 23
 - metrics for, 107–110
 - process improvement, 110–116
- Process Guidance, 23, 84–85
- Process technician role
- build management, 245
 - for developers, 72
- Process templates, 86
- changing, 115
 - importing and exporting, 112–114
- Product manager role, 71
- Production environment, 592–594
- Programmer role for developers, 72
- Programmer tests, 352
- Programming episodes, 60, 222, 560
- Project Administrators, 82, 88
- Project managers
- responsibilities, 77
 - roles, 75
- Project Portal, 27–28, 546, 551–552
- for projects, 83–84, 90–91
 - for version control, 157
 - templates for, 112
- Projects and project management, 545–546
- documents in, 551–554
 - planning life cycle. *See* Planning life cycle
 - reports in, 554–555
 - structure, 82, 546–547
- Prove It with Code practice, 373
- Provider endpoints, 641–643
- Proxy endpoints, 650–651
- Publishing
- with ClickOnce technology, 622–624
 - in remote deployment, 603
- Purchaser role, 71
- Putman, David, 690

Q

- Quality
- from Extreme Programming, 60–61
 - for models, 364
 - in plan control, 539, 541
- Quality Indicators queries, 671
- Quality of Service (QoS) tests, 220–221
- Quarterly Cycle practice, 538
- Queries
- adding, 109–110
 - in project management, 547–551
 - for projects, 83
 - standard, 670–672
- Queries folder, 546
- Query Builder, 679

R

- Rational Unified Process (RUP), 54, 377
- RDL (Report Definition Language), 675, 681
- Readers, 82, 88
- Real Customer Involvement practice, 43, 456
- Reeves, Jack, 66, 347, 686

- Refactoring, 127, 281, 303, 312, 318, 420, 422, 424
 - collection types in, 319–322
 - Remove the Middle Man pattern, 318–319
 - work breaks in, 322–323
 - opportunities, 316–317
 - for patterns, 420
 - tasks for, 561
 - in Test-First Programming, 280–281, 299–300
 - Refactoring Databases, 592
 - Refactoring to Patterns, 420
 - Regression testing, 219
 - Reinertsen, Donald, 66, 528
 - Relationships in class diagrams, 379, 382
 - Release manager role, 72
 - Releases and release process, 582–584
 - customer testing in, 457–458
 - to deployment teams, 586–587
 - in Extreme Programming, 59–60
 - notes for, 157
 - plans, 536–537, 566–567
 - in value stream, 685
 - in version control systems, 141
 - Remote access
 - Team Foundation Server, 489–490
 - Team Foundation Version Control, 146
 - Remote deployment, 603–604
 - Repeated execution plans, 528
 - Report Builder, 678
 - Report Definition Language (RDL), 675, 681
 - Report Designer, 678–681
 - Reports, 83
 - builds, 248–249
 - deployment, 657
 - in project management, 554–555
 - in Team Explorer window, 20
 - templates for, 112
 - Report Site, 16, 678
 - Repositories
 - folders for, 150–151
 - for sharing information, 129–131
 - in version control system. *See* Version control
 - Requirement models, 368, 395–396
 - customer stories, 404–405
 - Domain Modeling and CRC cards, 396–401
 - use case models, 403–404
 - user interface models, 401–403
 - Requirements gathering, 614–617
 - Respect as Agile value
 - Agile development, 122
 - Extreme Programming, 43
 - Responding to change, 36
 - Responsibilities in CRC, 397–398
 - Retrospectives, 697
 - overview, 697–698
 - planning, 699
 - preparation for, 698–699
 - Reusable application blocks, patterns in, 421
 - Reuse-Release Equivalence Principle, 425
 - Reverse-engineering AD diagrams, 646–647
 - Roles
 - associated, 75–76
 - customer, 70–72
 - developer, 72–73
 - Rolling back versions
 - deltas in, 135
 - shared folders for, 128
 - in version control systems, 134–135, 183–184
 - Rolling wave planning, 528
 - Root Cause Analysis practice, 687–688
 - Round-trip facilities, 647
 - Row fixtures, 465–470
 - RUP (Rational Unified Process), 54, 377
- ## S
- Sampling for performance, 331–332
 - Sanity tests, 582
 - Scaffolding in deployment, 590–591
 - Scenarios
 - in MSF 4.0, 96
 - in sequence diagrams, 382
 - in use case models, 403
 - Scheduling daily builds, 239–240
 - SCM (Software Configuration Management), 142–144
 - Scope in plan control, 541
 - SDLC (software development life cycle), 58–59
 - SDM (System Definition Model), 627–628
 - Securities page, 169
 - Security
 - groups
 - membership, 88–89
 - setting up, 724–725
 - repository, 143
 - team project settings, 91–94

- version control, 133, 169–170
- Windows Installer, 602–604
- Self-organizing teams, 68
- Sequence diagrams
 - for dynamic models, 413
 - UML, 382–385
- Server parts of VSTS, 27
 - Project Portal, 27–29
 - Team Foundation Build, 29
 - Team Foundation Server, 27
- Service-Oriented Architecture (SOA), 626–627
- Setup, 715–716
 - machine and user identification, 726–727
 - network, 720–721
 - single evaluation servers, 717–719
 - software installation, 722–723
 - system settings, 723
 - user accounts and security groups, 724–725
- Shared Code practice, 131–132, 356
- SharePoint Services, 742–743
- Sharing information, 127–128
 - repositories for, 129–131
 - Shared Code practice, 131–132
 - shared folders for, 128–129
- Sharp Tools, 34
- Shelvet (TFVC), 146, 187–190
- Simple Design, 346
- Simplicity
 - as Agile value
 - Agile development, 122
 - Extreme Programming, 42–43
- Single-checkout approach, 136
- Single Code Base practice, 140–141, 356
- Sit Together practice, 73–74
- Six Sigma, 691–692
- Size
 - Agile teams, 68–69
 - stories, 516–517
- Skeletal architectural models, 406–407
- Slack practice, 568, 559
- Smoke tests, 219, 221
- SMS (Systems Management Server), 246, 587, 619
- SOA (Service-Oriented Architecture), 626–627
- Software
 - automated testing, 217–221
 - development economics of, 688–690
 - small team requirements, 733–739
 - software factories for, 429–430
 - values and traditions, 35
- Software Configuration Management (SCM), 142–144
- Software development area, 754–755
- Software development life cycle (SDLC), 58–59
- Software factories, 429–430
- Software Project Environment (SPE), 14–15
- Source code.
 - Class Designer for, 385–389
 - importing, 166
 - policies
 - coding standards in, 195, 200–201
 - overriding, 201–203
 - rules updating for, 201
 - static code analysis for, 196–200
 - protecting, 191–194
 - repositories for, 129–131
 - in Team Explorer window, 20
- SourceSafe tool, 144
- Spike folder, 149–151
- Spike tasks, 531
- SQL Server Analysis Services (SSAS), 675
- SQL Server Business Intelligence
 - Development Studio, 678
- SQL Server Integration Services (SSIS), 675
- SQL Server Reporting Services (SSRS), 554, 675
- Standard queries and reports, 670–672
- State diagrams, 413–414
- Static Code Analysis tool, 23
 - rules updating for, 201
 - for source code policies, 196–200
- STDD (storytest-driven development), 454–455
 - costs and benefits, 455
 - testers, 457
- Stereotypes in class diagrams, 380
- Storage space, 755
- Stories. *See* Customer stories
- Stories practice, 513–514
- Story points, 525
- Storytest-driven development (STDD), 454–455
 - costs and benefits, 455
 - testers, 457
- Strict locking, 136, 183
- Structural models, 412–413

- Structural tests, 220
- Stubs in deployment, 590–591
- Subversion version control systems, 133
- Support role
 - in Agile teams, 75
 - for customers, 71
- Support sites, 594
- Support tasks, 531
- Synchronizing
 - Daily Builds, 227, 239
 - deployment, 591–592
 - libraries, 159
 - repositories, 129, 140, 151
 - shared data, 27
- System Definition Model (SDM), 627–628
- System Designer (SD) tool, 628, 649
 - applications of, 652–653
 - description, 26
 - diagrams
 - creating, 649–652
 - Deployment Designer diagrams from, 654–655
 - settings and constraints, 652
- System metaphor, 411
- System performance, 337
- System settings, 723
- System tests, 582
- Systems Management Server (SMS), 246, 587, 619

T

- TableFixture class, 477, 479
- Tables
 - for customer tests
 - information, 493–495
 - sequences of, 496–497
 - for modeling in pairs, 371
- Task boards, 557–558, 560
- Task cards, 532–535
- Task points, 519–520
- Task work item, 96, 548
- Tasks
 - vs. features, 514, 567
 - plans, 531–533
 - in stories, 288–289
- TDD. *See* Test-driven development (TDD)
- Team Builds, 29–30. *See also* Team Foundation Build (TFB)
- Team Continuity practice, 77–78
- Team Foundation Build (TFB), 16, 29, 229–230
 - build management, 245–249
 - Build Validation Test for, 233–235
 - Daily Builds, 239–240
 - deleting build products, 243–244
 - FIT with, 481
 - automated customer tests, 487–491
 - customer test introduction, 491–498
 - customer tests in Build Lab, 482–486
 - integration builds, 240–243
 - MSBuild engine role in, 231–232
 - operation of, 230–231
 - permissions for, 235–236
 - scaling up team integration builds, 249–253
 - setting up, 230
 - types, 237–238
- Team Foundation Server (TFS), 16, 27, 81, 149
 - backup and restore, 170–171
 - Client Tier, 16
 - extracting data from, 674–682
 - folders for, 149–151
 - importing source files, 166
 - licensing issues, 740–742
 - remote access to, 489–490
 - reports from. *See* Technical reports
 - repositories in, 174
 - security settings, 92–93, 169–170
 - source control options, 168–169
 - structure for, 149–160
 - Team Project options, 167–168
 - Visual Studio options, 168–169
 - for workgroups, 720–721, 740–741, 751
- Team Foundation Version Control (TFVC), 127, 173, 581
 - branching in, 185–187
 - in coding, 173–176
 - features, 145–147
 - merging changes in, 180–183
 - rolling back to previous versions, 183–184
 - shelves in, 187–190
 - source code policies in, 195–203
 - source code protection in, 191–194
 - workspaces, 177–180

- Team Projects, 21, 81–82
 - artifacts from, 82–85
 - deleting, 87
 - documents, 152, 156–158
 - membership, 88–89
 - MSF for, 85–86
 - policy settings migration, 201
 - portal and report site access, 90–91
 - security settings, 91–94
 - service access, 89–90
 - version control options, 167–168
- Teams. *See* Agile teams
- Technical risk in prioritizing stories, 523
- Technical writer role, 71
- Templates
 - changing, 115
 - importing and exporting, 112–114
 - for processes, 86, 111–116
 - for Setup Project, 604
 - projects, 421
- Ten Minute Build practice, 218, 224–225, 227, 356
- Test adapters, 284
- Test Case Management tool, 25
- Test cases, 270
- Test cycles, 222
 - daily builds, 227–228
 - integration builds, 225–227
 - local builds, 222–223
- Test-driven development (TDD), 53, 60, 63, 261–262, 265
 - class diagrams for, 378
 - code coverage, 325–330
 - cycles in, 271–277
 - list of tests in, 269
 - nature of, 265
 - operating system development team story, 262–263
 - performance. *See* Performance analysis
 - refactoring. *See* Refactoring
 - rhythm of, 265–266
 - in Shared Code practice, 132
 - Test-First Programming. *See* Test-First Programming (TFP)
 - test harness for, 269–271
 - top down vs. bottom up approach, 268
 - in version control, 174
- Test-First Programming (TFP)
 - applying, 277
 - code coverage, 330
 - conclusion, 301
 - cycles in, 271–277
 - finding tests for, 278–279
 - implementing, 294–300
 - list of tests in, 278
 - for modeling in pairs, 371
 - for performance, 336
 - refactoring in, 280–281, 299–300
 - stories for, 287–289
 - test lists for, 289–293
 - with user interface. *See* User interface
 - Visual Studio Projects for, 283–287
- Test-First Programming (TFP) practice, 266–267
- Test fixtures, 445
- Test harnesses
 - in test-driven development, 269–271
 - in Test-First Programming, 284–285
 - vs. test runners, 483
- Test lists, 278, 289–293, 454
- Test Load Agent, 749
- Test Manager window
 - for Build Validation Tests, 234
 - Visual Studio Professional, 19
- Test managers
 - OSPACS team, 3
- Test runners, 483
- Testable attribute, 515
- Testers
 - for developers, 72
 - PC licensing issues, 746–747
 - in storytest-driven development, 457
- Testing Object-Oriented Systems, 221
- Testing policy, 194
- Tests
 - customer. *See* Customer tests
 - FIT. *See* Framework for Integrated Test (FIT)
 - for integration, 216–217
 - for interface design, 347
 - TDD. *See* Test-driven development (TDD)
 - TFP. *See* Test-First Programming (TFP)
 - in value stream, 685
 - in Waterfall process, 54
 - Web services, 644–646

TFB. *See* Team Foundation Build (TFB)
 TFP. *See* Test-First Programming (TFP)
 TFS. *See* Team Foundation Server (TFS)
 TFS Data Warehouse, 675–678
 TFS Installation Guide, 16
 TFS Trial Edition setup, 719
 TFVC. *See* Team Foundation Version Control (TFVC)
 Thin user interface layers, 344–346
 Thin vertical slices, 556
 Third-party libraries, 158–159
 Time scales
 in planning, 530–538
 in project management, 568
 Timelines, 700–701
 creating, 701–703
 discoveries from, 706–707
 mining for gold process, 703–704
 for project structure, 705–706
 results from, 708–709
 Tools
 buy or build decisions, 34
 OSPACS team impressions, 45–46
 VSTS, 22
 all editions, 22–23
 Visual Studio Architect Edition, 26–27
 Visual Studio Developer Edition, 22–24
 Visual Studio Tester Edition, 24–25
 Top down approach, 268
 Toyota Production System, 683, 687
 Tracker role, 73
 Traditional projects vs. agile development, 53–57
 Trainer role
 in Agile teams, 75
 for customers, 71
 Translation of TFS data, 675
 Twain, Mark, 289

U

UML *Distilled*, 377, 392
 UML Modeling tool, 23
 Unified Modeling Language (UML)
 diagrams, 358–359, 377
 class, 377–382
 Class Designer for, 388
 sequence, 382–385
 Unit Test tool, 24–25

Unit tests, 220
 performance session for, 333–334
 setting up, 285–287
 support for, 283–285
 Update Only When It Hurts practice, 366
 Usage statistics, support for, 594
 Use cases, 57, 382, 403–404
 User interface, 339
 action fixtures, 470–476
 Big Design Up Front, 348
 defining, 340
 modeling, 401–403
 sample design, 346–348
 task lists for, 341
 thin layers, 344–346
 Windows Forms for, 342–344
 User stories. *See* Customer stories

V

Validation
 Build Validation Tests, 233–235
 Deployment Designer diagrams, 656–657
 Value, 665–666
 in Agile Modeling, 363
 generating. *See* Generating business value
 in prioritizing stories, 522
 technical reports. *See* Technical reports
 Values. *See* Agile values
 Velocity
 as rate of progress measure, 529–530
 in story cost estimation, 520
 Version control, 123–124, 133
 atomic check-in, 134
 conclusion, 147–148
 integration in, 134
 labeling and branching, 138–139, 146
 locking and merging, 136–138
 rolling back versions, 134–135
 security, 133
 Single Code Base practice, 140–141
 Software Configuration Management, 142–144
 support for, 144–147
 for Team Documents, 156–158
 templates for, 112
 TFS. *See* Team Foundation Server (TFS)
 TFVC. *See* Team Foundation Version Control (TFVC)
 Virtual PC environment, 217, 599, 717–718

- Visibility symbols, 380
- Visio for Enterprise Architects tool, 23, 389–390
- Vision statement, 157–158
- Visual Studio
 - solutions, projects, and directories in, 151–154
 - source control options, 168–169
 - version control system integration in, 144–145
- Visual Studio Industry Partner (VSIP) program, 30
- Visual Studio Professional, 18–19
- Visual Studio SDK, 30–31
- Visual Studio Team Edition for Architects, 17, 26–27, 410
- Visual Studio Team Edition for Database Professionals, 17
- Visual Studio Team Edition for Developers, 17, 22–24
- Visual Studio Team Edition for Testers, 17
- Visual Studio Tester Edition, 24–25
- Vlissides, John, 416
- VSIP (Visual Studio Industry Partner) program, 30

- W**
- Wake, William, 281, 515
- Waterfall projects
 - phases in, 54–55
 - value generated by, 689–690
- Web services, 644–646
- Web Services Description Language (WSDL), 627
- Web Test tool, 25
- Weekly Cycle practice, 536
- White box testing, 220
- Whiteboards
 - for customer tests, 492–493
 - in office space, 755–756
- Whole Team practice, 67–68
- Williams, Laurie, 39
- Window Scheduled Task Wizard, 239
- Windows
 - Code Coverage Results, 326–328
 - Error List, 196–198, 657
 - Pending Changes, 181–182
 - Performance Explorer, 333–334
 - Query Editor, 676–677
 - Query Results, 549
 - Solution Explorer, 18
 - Source Control Explorer, 184
 - Team Build Report, 242
 - Team Explorer window, 20–22, 86
 - Test Results, 241, 295–296
 - Test View, 19, 298–300
- Windows Forms applications
 - creating, 607
 - for user interface, 342–344, 475
- Windows Installer, 597–598
 - basic concepts, 598–600
 - operation, 600–602
 - security for, 602–604
- Windows SharePoint Services (WSS), 112, 156–157
- WinFITRunner, 454
- Wirfs-Brock, Rebecca, 396–397
- Wizards
 - Import IIS Settings, 636–637
 - Performance, 333
 - Release, 611–613
 - Report Server Project, 678
 - Team Build Type, 237
 - Team Project, 85–86, 129
 - Virtual Machine, 718
- Womack, James, 683
- Work items
 - adding, 108–109
 - Check-in Policy constraint, 194
 - for projects, 83, 547–551
 - templates for, 112
- Work products, 83
- Workgroups, TFS, 720–721, 740–741, 751
- Workspace
 - permissions, 92–93
 - for TFVC, 177–180
- WSDL (Web Services Description Language), 627
- WSS (Windows SharePoint Services), 112, 156–157

- X–Z**
- XP. *See* Extreme Programming (XP)
- Yesterday's Weather rule, 530
- Zones in Logical Datacenter Designer, 631–633