Brownfield Application Development in .NET

KYLE BALEY
DONALD BELCHAM

MANNING
Greenwich
(74° w. long.)
1 Understanding brownfield applications 1

1.1 Welcome to Twilight Stars! 2
    Reality check 2 • Day two 3 • Settling in 3
    Taking stock 4

1.2 The components of a brownfield application 4
    Existing codebase 5 • Contaminated by poor programming practices 6 • Potential for reuse or improvement 7

1.3 Inheriting an application 8
    Pain points 9 • Identifying friction 10 • Challenging your assumptions 11 • Balancing the client's expectations 11
    The big rewrite 13

1.4 The challenges of a brownfield application 14
    Technical factors 14 • Political factors 15 • Team morale 17 • Client morale 18 • Management morale 20
    Being an agent for change 20
1.5 Selling it to your boss 24
1.6 Summary 27

PART 1 The ecosystem

2 Version control in brownfield applications 31

2.1 Pain points 32

   Missing references and assemblies 32 • Files locked for checkout 33 • Maintaining production code 33

2.2 What's in a version control system? 34

   Required features 34 • Greasing the wheels 36 • File management 37 • Branching 41

2.3 Organizing your project for version control 43

   Dealing with third-party dependencies 43 • The folder structure 45
   The VCS folder structure 48 • What not to include in the repository 49 • The first check-in and verification 50

2.4 The check-in dance 51
2.5 Tips for working with version control 53

   Check in often 53 • Decide on a labeling/tagging strategy 54

2.6 Summary 55

3 Continuous integration 56

3.1 Pain points 57
3.2 What is continuous integration? 58

   The feedback loop 59 • Closing the feedback loop 59

3.3 Automated builds and the build server 60
3.4 An updated check-in dance 61

   Breaking the build 62 • Removing the pain: a check-in walkthrough 64 • The metronome effect 65

3.5 Setting up a CI process 67

   Compile the application 69 • Run automated tests 72
   Create deployable releases 73

3.6 Build components 76
3.7 Tweaking the CI process 79

   Handling long-running tests 79 • Architectural changes and additions 80 • Labeling the build 80

3.8 Summary 81
4 Automated testing  83
4.1 Pain points  84
4.2 Living in fear of change  85
  Regaining confidence and control  87 • The need for tests  89
4.3 Types of automated tests  89
  State-based tests  90 • Interaction tests  92
4.4 (Re-)integrating existing test projects  94
  Examining existing test projects  94 • Address existing tests  96
  Separate unit tests from integration tests  100 • Reexamining your
tests  102 • Executing the tests in your build script  102
4.5 Adding a new test project  104
4.6 Writing new tests  105
  Tests on existing code  105 • Tests on new code  107
4.7 Integration tests  108
4.8 User interface tests  110
4.9 Managing your database  111
  Autogenerating the database  112 • Dealing with an existing
database  113
4.10 Summary  114

5 Software metrics and code analysis  116
5.1 What are software metrics and code analysis?  117
  Static code analysis  117 • Dynamic code analysis  117
  Why do you need code metrics?  117
5.2 Some metrics of interest  118
  Code coverage  118 • Cyclomatic complexity  119 • Class
coupling  121 • Cohesion  122 • Distance from main
sequence  123 • Depth of inheritance  125 • Maintainability
Index  125
5.3 Becoming a slave to statistics  126
5.4 Implementing statistical analysis  129
  Automating code coverage  129 • Automating other
metrics  131 • Metrics with CI  132
5.5 Other tasks for automation  133
5.6 Summary  133
6  Defect management 135
  6.1 Pain points 136
  6.2 Types of defects 137
    Languishing entries 137 • Incomplete entries 138 • Spurious entries 138 • Feature requests 138
  6.3 The initial defect triage 139
  6.4 Organizing the effort 142
    Tackling the defect list 143 • Maintaining momentum 144
    Dealing with spurious defects 145 • Defects versus features 146
  6.5 Anatomy of a defect entry 149
    Concise 150 • Clear 150 • Complete 150
    Constructive 152 • Closed 153
  6.6 Instilling a culture of quality 154
    Zero Defect Count 155 • Easing fears and concerns from stakeholders 158
  6.7 Working with a defect-tracking system 159
  6.8 Defect reporting 161
    Trending data 161 • Release notes 163
  6.9 Summary 164

PART 2  THE CODE ................................................................. 167

7  Bringing better OO practices to the project 169
  7.1 OO fundamentals 170
    Encapsulation 171 • Inheritance 172 • Abstraction 172
    Polymorphism 173
  7.2 Code “-abilities” 174
    Maintainability 174 • Readability 175 • Testability 176
    Extensibility 176 • Reversibility 176 • Adaptability 178
  7.3 Useful OO principles 179
    Favor composition over inheritance 179 • Separation of concerns/
    single responsibility principle 181 • Open/closed principle 185
    Principle of least knowledge 190 • Dependency inversion
    principle 192 • Interface-based design 192 • Introducing
design principles to a team 194