



Automation

Tips and Techniques

Jeff Martin, mkodo
Tim Bacon, Prime Eight

Overview

- Why automate?
- What can be automated?
- How can I apply automation to my project?
- When can automation be applied?

Why you need Automation

- Prevent mistakes
- Reduce tedium
 - “Pragmatic programmers are lazy”
- Increase productivity
- Capture knowledge
 - Lower the truck number
- Maintain security
- Apply or enforce standards

What is Automation - approach

- Ad hoc
 - “Hail the power of the command line”
 - IDE macros, shell / batch scripts, cron jobs
 - Think: refactor mercilessly, avoid spaghetti
- Integrated
 - Suite or framework of events, actions, and information flow
 - Think: YAGNI

What can be automated?

- All traditional project phases
 - Design
 - Code
 - Compile
 - Test
 - Release
- Minimal 'continuous integration' in XP is automation of compile and test

How to automate

Activity

- Generate
- Build
- Exercise
- Do
- Inspect
- Alert

Tool

- XSL, Anakia, PHP, <modelling tool>, ...
- Make, Ant
- Xunit (and many extensions)
- Batch scripts - cmd, sh, expect
Embedded scripting - BSF, VBA
- grep, tail, <metrics tool>
- Email, web page. SNMP, syslog, ...

Automation Frameworks

- Alexandria
- Gump
- Cruise Control
- Roll-your-own
 - Interactive or background operation
 - Transparency is key: separate intent from implementation
 - Open source software very useful

Hints and tips

- Gotchas
- Dependencies
 - Test data
 - Test classes
 - Environment
- Metrics and automation

Gotchas

- Automation uncovers issues that may not otherwise be apparent
- Code needs to be automatable
 - Requirement is consistent, traceable execution
 - Explicit call stack
 - Entry points well-defined
 - Processes and threads exit in one place
- Environment may need refactoring too
 - Common file system on every machine
 - Single machine dedicated to automation tasks

Managing Dependencies

- Dependencies can prevent repeatable results after small changes
 - Repeatability is key to automation
- Understand the dependencies you find
 - Either eliminate them
 - Or accept them

Test data dependencies

- **Problem**
 - Adding one new test breaks existing tests because the new test alters data used by existing tests
- **Solution**
 - Isolate tests from data
 - Isolate data from tests

Test data dependency solutions

- Mock Objects
- Object Mother
- Random (but repeatable) test data
 - Hard to do, especially for boundary conditions
- Command pattern
 - Must undo consistently
- Inter-test database refresh
 - Very expensive

Test case dependencies

- Problem
 - Single code change breaks multiple tests and it's unclear which to start fixing first
- Solution
 - Ensure test cases exercise disjoint code
 - Separate tests by degree of dependency (e.g. functional tests depend on unit tests)

Test case dependency solutions

- Mock Objects for unit tests
 - What about functional tests?
- Run test cases in random sequence
 - Best way to uncover the dependencies is to uncover the dependencies
- Run tests in dependency order
 - Make dependencies explicit: 1 or many methods, 1 or many classes, ...

Environment dependencies

- **Problem**

- Live system exhibits different behaviour to development system

- **Solution**

- System knows what code will be executed (e.g. version)
- System knows its code's execution environment

Environment dependency solutions

- Be specific: what environment can you guarantee that the application will run in?
 - OS, VM, libraries, CORBA, LDAP, DBMS, hardware, file system, memory, ACLs...
 - Probe environment at startup and installation
- Understand performance dependencies
 - Run tests in-process, out-of-process, and cross-network
- Patch releases depend on known state

Automation and metrics

- Automation simplifies metric collection
- But data is not information
 - Is yesterday's weather relevant today?
 - Is LOC relevant at all?
- You will act on what you measure
 - Or worse, you will be acted on
- XUnit extensions helpful
 - Test cases for quality, throughput, load,

When to Automate

- Start of a project
- As team or scope grows
- When duplication becomes repetition
 - “Three strikes and you automate”
- All the time
 - Automation is refactoring for the project

When not to Automate

- Start of a project!
 - Predicting the future is futile
- When there's a simpler thing that could possibly work
 - That you haven't done manually n times before

Summary

- Automation is your friend
 - It benefits you and your project
- There are a lots of tools that help you
 - But you won't need them all
- Automation fits the agile paradigm
 - Apply it first where you need it most
 - Adopt an iterative and incremental approach

Resources

- timBacon@primeEight.co.uk or jeff@mkodo.com
- Ant and Alexandria <http://jakarta.apache.org/>
- Cruise Control
<http://cruisecontrol.sourceforge.net/>
- JUnit and extensions <http://www.junit.org/>
- 'The pragmatic programmer' ISBN0-201-61622X
- 'Continuous integration'
<http://www.martinfowler.com/>