

Dancing with Elephants

Dealing with Open Source Projects



Presented by D. Keith Casey, Jr.
keith@caseysoftware.com

The Scenario

- Numerous building blocks
- Numerous developers and teams
- Fully customizable
- Frequent development on each
- Changing interfaces and behaviors

The Problem

- Numerous building blocks
- Numerous developers and teams
- Fully customizable
- Frequent development on each
- Changing interfaces and behaviors

Tipping Point

- Most organizations have numerous applications deployed throughout the stack
- Improved security, licensing, costs, etc, etc
- Customization – the biggest strength – also the biggest risk
- *Not Invented Here -> Not Invented Anywhere*

Case Study #1

- Java 1.4.2
 - Tomcat 4.1.29
 - Mysql 4.0
 - Database driver
 - [nothing]
 - org.w3c.dom
 - Custom pooling
 - Java 1.5
 - Tomcat 5.5
 - Mysql 4.1
 - Database driver
 - Ant 1.6
 - JDOM 1.0
 - Included pooling
-
- 7 days times 2 developers = 112 hours
 - Loss of Credibility, Stress, etc

Principles

- Leadership
- Communication & Coordination
- Functional Validation
- Development Principles

Leadership

- Most effective
- Inside information, deeper understanding of team dynamics, constraints, etc
- Opportunity to direct roadmap and goals
- Most expensive in terms of time, effort, and stress

Communication & Coordination

- Highly effective
- Inside information, deeper understanding of team dynamics, constraints, etc
- Ideal for teams who can offer frequent and sometimes significant involvement
- Our primary route

Case Study #2

- 3 Developers' Contributions:
 - 2 dotProject – core, modules, advocacy, beta
 - 1 PEAR – debugging, refactoring
 - 1 Smarty – beta, debugging
 - 1 Gallery
 - 1 SugarCRM
 - 1 phpBB
 - 1 financial contributor to Mantis
 - 1 financial contributor to Mambo

Case Study #2 (cont)

- dotProject HelpDesk Module
 - Allows for form or email-based submission of issues, trouble tickets, etc
 - Highly used but abandoned the team
- Coordination
 - Integration with existing Issue Tracking Systems including Mantis, Eventum, Scarab, FogBugz, and Bugzilla
 - Common set of methods
 - Adapter/Gateway pattern = pluggable interface
 - *Scarab, FogBugz, and Bugzilla – not yet*

Coordination

- Roadmaps. Roadmaps. Roadmaps.
- Core project team will lose sight of complementary projects
- Ever-growing scope
- Duplication of effort
- Ie. dotProject HelpDesk

Functional Validation

- Integration Unit Testing

- Unit Testing asserts the smallest unit of code “works”
- Systems Tests assert that the whole system “works”
- What about in the stuff in between?

- *“Note that the IUTs test the “public”, or “outside” usage of the library. The IUTs can be used as a litmus test to ensure that the library retains binary backwards compatibility with earlier versions. My standard is that once a library is shipped as outside of beta, then the existing IUTs must not change. If an IUT fails, then binary compatibility has broken.”*
- Source: http://groboutils.sourceforge.net/testing-junit/art_iut.html

Enemy Unit Tests - Why?

- Our own code is subject to Unit Testing
- Supporting libraries are developed by unknown developers of unknown abilities, priorities, loyalties, and goals
- Trust But Verify
- 100% coverage is not necessary

Codebase Education

- More Unit, Integration, and Enemy Testing = more knowledge
- Fast way to evaluate libraries and estimate usefulness – Code spikes, Build vs Buy/Find
- Fastest way to climb the curve
- Evaluate code of new team members

Development Principles

- Forethought and planning
- Evaluate alternatives, scope, and goals
- Design Patterns
- Agile Methodologies (TDD, XP) due to shifting requirements, interfaces, etc

Planning & Evaluation

- Establish version baselines, dependencies, and lifecycle as soon as possible
- Determine which portions are your problem and which are irrelevant
- Choose libraries with active development, public roadmaps, and active communities

A Pragmatic Approach...

- Don't Repeat Yourself (DRY)
- Eliminate Effects Between Unrelated Things
- Design by Contract
- Find Bugs Once
- Design Using Services
- Separate Views from Models
- *Source: The Pragmatic Programmer, 2000*

Design Patterns

- Remote Facade / Service Layer
 - Defines an application's boundary and its set of available operations from the perspective of interfacing client layers
 - Enforces encapsulation and limits the damage done by codebase drifting
 - The model for the dotProject HelpDesk

Design Patterns (cont)

- Two-step View – User Interfaces
 - Splits the transformation into two stages. The first converts from model data into an established logical presentation. The second converts from the logical presentation and applies the specific formatting or “paint”.
 - Useful when backend data structures vary – for a variety of reasons - but the presentation should be the same.

Case Study #4

- XML Import System
- Initially 2 different formats from 12 sources
- Now 7 different formats from 237 sources
- Some items upwards of 4k/day

- Implemented Two-step View

A message I got...

- New team member on a Java project with 35+ third-party libraries (OSS, Share, Com)
- Concerns on maintenance, re-integration, changes, code quality, growing complexity, learning curve for new people, etc
- Whole libraries included with small portions being used
- Key Quote: *“Although I'm convinced that the use of such libraries is a good thing, I'm in doubt in this particular case (at least for the number of libraries used).”*

Strategy

- Confirm required libraries
- Track down information on the libraries including the originating group, the dev status, and – if possible – the source
- Develop UT's (IUT's and EUT's)
- Determine which libraries have viable replacements
- Minimize growth while ensuring compatibility!
- REMEMBER: *If it's not broke, don't fix it...
but have a plan in place.*

Principles

- Leadership
 - Most effective, most expensive
- Communication & Coordination
 - Very effective and satisfying, can be expensive
- Functional Validation
 - Testing - Integration (IUT), Enemy (EUT)
 - Educational, Offers Evaluation Methods
- Development Principles
 - Development should be purposeful, not fitful
 - Well designed interfaces and clean separation

Sources

- Martin Fowler - <http://martinfowler.com/>
 - Patterns of Enterprise Architecture, Refactoring
- Kent Beck
 - Test Driven Development
- David Astels
 - A Practical Guide to eXtreme Programming
 - Test-Driven Development
- The Pragmatic Programmers – Andy Hunt & Dave Thomas -
 - <http://PragmaticProgrammer.com/>
 - Pragmatic Programmer, Pragmatic Project Automation
- Agile Alliance - <http://www.agilealliance.org/>
- GroboUtils - <http://groboutils.sourceforge.net/>
- xUnit Testing Frameworks - <http://www.junit.org/>
- Web-based Project Management Tool - <http://dotProject.net/>

Contact

- Keith Casey - keith@CaseySoftware.com
- CaseySoftware, LLC
- <http://CaseySoftware.com>
- Blog: <http://blogs.CaseySoftware.com>