FRAMEWORKS 2007 CONFERENCE

Building Sustainable Software With Frameworks

Matt Woodward
Principal IT Specialist, U.S. Senate

About Me

- Principal IT Specialist, Office of the Sergeant at Arms, United States Senate
- ColdFusion developer since 1996
- Also develop in Java, Flex, C#
- Release Coordinator and contributing developer for Mach-II
- Co-host of the ColdFusion Weekly Podcast (http://www.coldfusionweekly.com)

Agenda

- High-level discussion of approaches to building sustainable software using frameworks
  - Investigation of the problem (yes, we have a problem)
  - Approaches to attitude adjustment
- Not a lot of code, but we'll look at a sample scenario
- I hope you leave here with a new outlook on your development process!

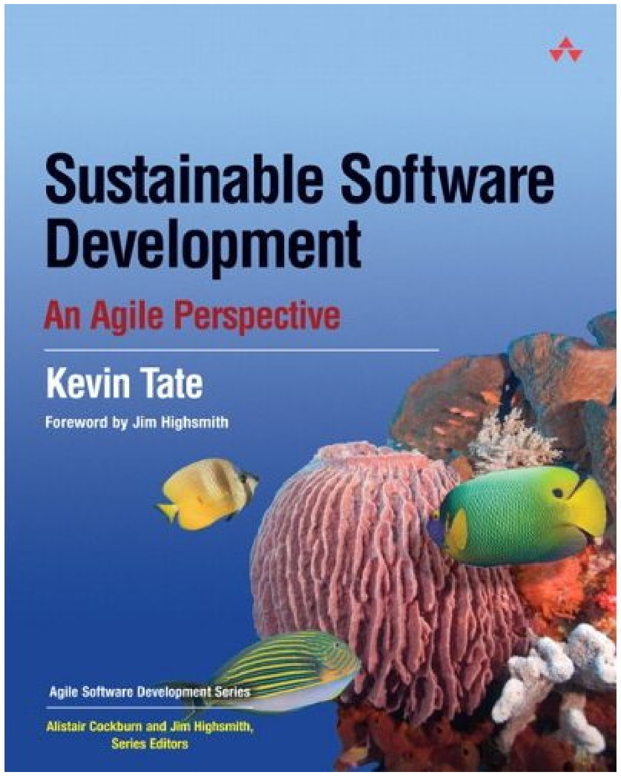
We Have a Problem

- The first step towards recovery is admitting we have a problem
  - Nearly 80% of software projects fail
  - Discuss some of the whys behind this statistic
- If we're lucky and do get our applications out the door, then we have to maintain them
  - Discuss how to design applications for sustainability
  - Investigate how using a framework can help with building sustainable software

What Is Sustainable Software?

- "Sustainable" is borrowed from ecology
  - "meet the needs of the present generation without compromising the ability of future generations to meet their own needs" (Bruntland Report, 1987)
- Build software in such a way that the creation of the application does not prevent its long-term use and sustainability

Sustainable Software Development


Sustainable Software Development
An Agile Perspective
Kevin Tate
Foreword by Jim Highsmith
Agile Software Development Series
Alistair Cockburn and Jim Highsmith, Series Editors

- Excellent book by Kevin Tate
- Extremely practical, agile approach to development
- Many of the goals he outlines are directly supported through the use of a framework

Software Development As Architectural Engineering

- Valid metaphor? Is this what we're trying to build?

Software as Ecosystem

- More appropriate metaphor, because software …
  - Evolves over time
  - Reacts to change
  - Stasis is not possible nor desirable
  - External forces drive change
- Goal: Create a sustainable software ecosystem

Characteristics of Unsustainable Software/Projects

- Technical debt
  - Builds over time, gets worse as more changes are made
  - Hard to pay back
  - How do you boil a frog, or, how do you kill a project?
- Death spiral projects
  - Work harder and faster towards impending doom
- Jumping in place
  - Work long and hard just to keep something running
  - Never have time to make forward progress
  - Reactive, not proactive

Characteristics of Sustainable Software/Projects

- Principles, not practices
  - Don’t blindly follow practices, especially ones that have failed in the past
- Focus is on end goal, not the process
  - User/customer focused
  - Continual feedback from end users
- Application must adapt to change
  - Continual refinement and improvement
  - Flexibility in application architecture

Keys to Sustainable Development

- A Working Product at All Times
  - Daily builds
- Don’t “code then fix” — “fix then code”
  - Use test-driven development to catch problems early
- Have uncompromising standards for code quality
  - Team members hold each other to these standards
- Don’t overdesign the application
  - Build only what the customer needs
- Zero tolerance for defects
  - Defects contribute to technical debt
- Replan often
  - Never be afraid to abandon something that isn’t working, no matter how much you have invested in it
- Continual improvement

A Working Product at All Times

- The software should always be in a working state
  - Not feature complete, but what's there should be of good quality and should work
- Always be ready to ship what's there
- Feature prioritization and end user involvement is critical

Continual Refinement

- Applies to both the software itself and your development process
- Don’t keep doing something that doesn’t work!
- Refactor aggressively during development
  - Radically change code if it will make it better
  - Never be afraid to throw something out and start over, even if you’ve spent a lot of time on it
- Be vigilant about writing the best code you can
  - No broken windows! These increase technical debt
- This is all an attitude more than a skill, but recognizing when to do these things comes with experience

Defect Prevention Over Defect Detection

- Anyone can fix bugs
- Goal should be not to introduce bugs in the first place
  - Don’t make the end users be testers!
- Test-Driven Development (TDD) helps tremendously

Sustainable Principles in Practice

- UI prototypes + continuous user involvement
- No broken windows
- Be uncompromising about defects
- Continuous integration/daily builds
- Don’t neglect performance
  - … but don’t prematurely optimize either
- Coding guidelines and standards
- Application development standards
- Code for platform independence

Where does a framework fit in?

- Coding guidelines and standards
- **Application development standards**
  - Developers using not only the same programming language, but the same application development practices via the framework
- An OO/MVC framework lends itself very well to sustainable development
  - Separation of concerns (MVC)
  - Encourages good object development (tight cohesion, loose coupling)
  - Low-impact maintenance = good sustainability

Using Mach-II as an Example

- MVC
  - Model = business logic
    - Accessed via a service layer that acts as the API for the business logic
    - Totally framework unaware
  - View = UI
    - Should contain ONLY presentation logic
  - Controller = flow control and intermediary between UI and model
    - Communicates with service layer
- Maximum flexibility, minimal impact of changes

OOP, Cohesion, and Coupling

- Well designed objects are far easier to maintain than big hairy procedural code
- Mach-II by its very nature encourages good OO development practices
- Tight Cohesion: do one thing and do it well
  - Low-impact maintenance
- Loose Coupling: minimize dependencies
  - Object A doesn't need to know the inner workings of Object B
  - Think in terms of messages between objects through publicly exposed interfaces
  - Telephone or car analogy

XML Configuration File

- Provides roadmap and behavior of the entire application
- Application behavior and flow can be radically altered with simple changes to the XML configuration file
- Easy for future developers to come in and see exactly what's going on in the application
    - Sustainable by design

Example: Authentication

Login Form Submitted

- Event announced
- Mach-II listener notified
- At Mach-II level, easy to change behavior and flow without impacting anything else

Service Layer

- Called by listener
- Unaware of Mach-II
- Provides API to model

Model and Database Layer

- Gateways and DAOs interact with database
- Extremely easy to swap out databases or even authentication mechanisms (LDAP, etc.)
- Very isolated from rest of the application—low impact maintenance

Code Sample: Authentication

- Show a layered approach to authentication
  - Illustrate how easily authentication methods can be swapped out
  - Illustrate separation of concerns in the various CFCs
  - Illustrate controlling access to events in Mach-II without touching the view layer

Sustainable Team Culture

- Disciplined, responsible development
- Leadership at all levels
- Visionary and tactical approach to development
- Shared sense of urgency
- Highly collaborative, lots of mentoring
- Complementary talents and skills
- Continually improving and learning
- Change tolerant
- Risk aware
- Fun

The #1 Enemy of Sustainability …

- COMPLACENCY! Avoid this through …
  - Demanding active participation by everyone, every day
  - Caring about every aspect of everything you do
  - Never compromising quality
- Every kludge increases technical debt
  - Someday this debt will come due!

Summary

- Write sustainable software!
  - Don’t write throwaway applications
- Frameworks help with sustainability
  - Provide application development standards
  - Application architecture is more clear (standards, XML configuration file)
  - In Mach-II’s case, the framework by nature encourages development of sustainable components
- Care about what you do today and how it impacts your application (and your fellow developers) tomorrow
  - Minimize technical debt
  - Don’t make future developers pay for your shortcuts

Resources

- Tate, Kevin. *Sustainable Software Development: An Agile Perspective*
- Larman, Craig. *Agile & Iterative Development: A Manager's Guide*
- Evans, Eric. *Domain-Driven Design: Tackling Complexity in the Heart of Software*

Questions?

- THANKS!
- Matt Woodward
  mpwoodward@gmail.com
  http://www.mattwoodward.com/blog
  http://www.coldfusionweekly.com