



**Software Development
Best Practices 2008
Conference**
Overview and Thoughts

Conference Impressions

- Software development practices and tools are still maturing rapidly
- Agile is accepted, but still no easier than traditional!
- Test Driven, Test Driven, Test Driven
- Automated testing gaining ground in areas besides development, e.g. acceptance testing

Some Agile Concepts

- Never Be Blocked!
- Visibility metrics for customer:
 - Burn down
 - Velocity
- “Agile contracting” new trend

Introducing Agile to an Organization

1. Has a “need for change” awareness
2. Determine what is valuable to the customer and select Agile practices
3. Introduce
 1. To a few small teams first
 2. Only one ‘dimension’ at a time
4. Gain experience and eliminate waste

Match Agile Practices to Organizational Needs

- Not all agile practices created equal
- Different organization suggest require different practices
- **Examples:**
 - Continuous integration reduces time-to-market, quality and visibility
 - Simple design reduces time to market, cost and lifetime

Test Driven Practices: Trends

- Basically embraced by the industry
- But why isn't it widely adopted yet?
 - Not trivial to do
 - Much more difficult with legacy code (most codes)
 - Industry is still learning best practices, e.g.
 - Mock ups
 - Patterns of refactoring
 - The "need to test" code affects the design (e.g. having a testable interface)

Test Driven Practices: Analogy

- Applying a scientific method to code development
 - Form a hypothesis - write a test
 - Perform an experiment - Run the test/write the implementation
 - Repeat

Test Driven Practices: Tool Support

- Besides Xunit Fx's:
 - Growing sets of COTS/OSS support tools: Jmock, EasyMock, Clover, Selenium, Cactus, Agitar, etc.
- Testing parallel code perfectly is currently difficult or impossible
 - A concurrency testing framework exists
- Refactoring tools support is phenomenal for Java, but Fortran?

Test Driven Practices: Some Concepts

- Test code is AT LEAST as important as the code
- Got a large feature?
 - Implement a small “slice” of the feature that crosses all layers of the system
 - e.g. UI--> model --> persistence
- Test “smells”
 - e.g. shared test fixture like a DB = “Test Run War”

The QA Shift

- From verification to specification
- QA now writes acceptance tests
- Again! QA writes tests first, before developers write code

Aspect-Oriented Programming

- Cross-cutting concerns that occur through unrelated sections of code (e.g. logging)
- Add aspects to code without changing existing code
 - Example: “Code that implements interface Play, should have these aspects...”
- Centralized policy decisions for those particular aspects
- Performance is approximately same as non-AOP, although startup is slower
- Still immature, but potentially beneficial

XML

- *The standard data exchange format*
- Lots of tool support
- When should I use it?
 - Data needs to be exchanged
 - Standardize a robust message format
 - Or data is expected to be extended
 - Defining new fields is easy with the tools
 - Data will be stored long-term
 - Easy to understand with the meta-data
- Fortran has an XML parser too (FoX)