

Transitioning to Agile Software Development

2004 DPI Professional Development Week

May 26, 2004

What is Agile Development?

- Typified by heightened attention to individuals and interactions, working software, customer collaboration and responding to change.
- Relies on frequent feedback to verify that the software being produced is actually what's required

Why does it exist?

- People disillusioned with the application of processes for the processes' sake, and the resulting high rate of failure
- Interesting to note that most agile practices have been around for decades, and are simply best practices

How long has Agile been around?

- Mercury space program in 1957
- 1970, Winston Royce proposed two-pass development
- 1972, Trident submarine command & control system
- 1972, US Army ballistic missile defense system
- 1977, Space Shuttle primary avionics software
- 1980's, US Navy AEGIS Combat System
- 1990's, Canadian Automated Air Traffic Control System

What's so great about it?

- Risk identification and mitigation (iterative development)
- “Design a little, build a little, test a little, learn a lot!”
- Responsive to change, allowing adaptation to changing business requirements
- Smaller up-front investment
- See tangible results earlier; the system grows over time

What's so great about it (cont'd)?

- Smaller teams allow more effective verbal communication, reducing (but not eliminating) need for written documentation
- Standish Group CHAOS 2000 Report: Reduce requirements to the bare minimum, provide constant communication systems coupled with a standard infrastructure, add good stakeholders, an iterative development process, management tools, adherence to key roles (PM, etc.) and success is almost guaranteed

Resistance to Change

- Why wouldn't we use Agile if it's so good?
 - No pain, no perceived reason to change
 - "Not invented here"
 - Natural fear of change
 - Management fears loss of control – sees chaos
 - Developers fear accountability – focus is on them
 - No documentation myth
 - Agile development isn't disciplined
- What projects shouldn't use Agile?
 - Any project in which no change will occur between start and finish, which translates to 0% of all projects

How to 'Sell' Agile Development

- Need to identify the Gold Owner; this is the 'executive support' of the Standish Group's reports
- They fear risks:
 - Fear risk through the introduction of a new process
 - Agile processes are less formal, therefore there is a fear of loss of control
 - Fear that a new process will take up too much time to learn and implement, making it too expensive

How to 'Sell' Agile Development (cont'd)

- Need to focus on:
 - Risk reduction through feedback (customer involvement, iterative development)
 - Risk reduction through the focus on delivery highest business value requirements first
 - Risk reduction through short release cycles, providing the ability to adapt to changes in business
 - Risk reduction through the visibility of the process – everyone knows what's really going on, so problems don't 'hide' for months
 - Reduced development costs through the focus on testing – faster delivery, higher quality product, reduced QA effort

Strategies for Transition

- Ensure that the project has the full support of the Goal Donor
- Ensure that the project has a Customer, or group of Customers who speak with one voice; proxies for the Customer (e.g. Business Analysts) are OK, provided that they are empowered to make decisions
- Use an experienced agile coach for the development team, even for a short period (1-3 iterations)

Strategies for Transition (cont'd)

- How large a team can use Agile?
 - As team size increases, more communication is required
 - Works best with face to face interaction, which is difficult with larger teams
 - However, smaller teams can often work more efficiently (9 women can't make a baby in a month)
- Standish Group research has indicated that the optimum is 4 developers, 4 months, and \$500K (US); keep it as small as possible!

Strategies for Transition (cont'd)

- Difficult to switch all development cold turkey
- Start with a small pilot project
- Need close Customer contact from the start (the feedback is critical)
- Perform retrospectives after each iteration and after the release – determine what went right, and what to change the next time the approach is used
- Apply to larger projects (or make the larger projects smaller!)

Strategies for Transition (cont'd)

- Move away from separate Development Team and Maintenance Team mentality – applications should be in maintenance mode as soon as possible
- Have the applications in “shippable” form as early and often as possible
- Allow rigorous testing to be your safety net – automated unit tests, preferable automated acceptance tests

Strategies for Transition (cont'd)

- From a team that is successful with agile, seed other teams with people who make good coaches
- Ensure that the team has all necessary resources
- Try to maintain an open workspace that encourages collaboration
- Development tools:
 - First, and foremost, the xUnit testing frameworks
 - IDE's that support Refactoring (Visual Studio, JBuilder, Eclipse, etc.)
 - Continuous integration tools such as CruiseControl

Learning How to Enjoy Testing

- Begin with a single test for a single method for a single class
- Write a test for a single scenario for the method, then write the code to make the test pass.
- Write another test for another scenario, watch that test fail, then write the code to make it pass
- Use this approach to organically build a comprehensive unit test suite that can be run with the push of a button

Working with Legacy Applications

- Writing tests for existing code is difficult and tedious; start by isolating modules or interfaces and writing tests for them
- Work piecemeal by default
- Each time you go to change legacy code, write unit tests around it first
- When you go to use legacy code, especially unfamiliar code, consider writing unit tests around that code to prove to yourself you know how to use it
- Add a unit test every time you find a problem, no matter where the problem occurs or how simple it is to fix; the tests will quickly grow to the 'correct' size

Bibliography & Resources

- Iterative and Incremental Development: A Brief History; Craig Larman & Victor Basili
(<http://csdl.computer.org/dl/mags/co/2003/06/r6047.htm>)
- Can Agile Methods Remedy Software Risks?; John Manzo, AgileTek (<http://www.agiletek.com>)
- The CHAOS Report (1994), CHAOS: A Recipe for Success (1999), Extreme CHAOS (2001); The Standish Group
(http://www.standishgroup.com/chaos_resources/index.php)
- Extreme Programming Wiki (C2);
(<http://c2.com/cgi/wiki?ExtremeProgramming>)

- This presentation is available at
<http://www.mayford.ca/downloads/dpi2004.pdf>