Agile Development Methods

The Myths And The Reality: A User Perspective

John Manzo
Principal

USC/CSE Research Review
March 19, 2003
“For the uninitiated, Extreme Programming (or XP) is a popular software development process that encourages a return to the days of little or no documentation, **Design After First Testing**, and, **Constant Refactoring After Programming**. Despite its popularity, not everyone thinks XP is a good idea.”

Source: SoftwareReality.Com
Key Observation

Agile software development is not undisciplined. It simply follows a different discipline.
Topics

- Who Are We, And Why Might Our Perspective Be Of Value?
- A Brief Overview Of Agile+ - An XP-Inspired ADM
- The Myths Surrounding XP
- Wrap Up
- Q & A
Background And Basis...

- AgileTek has a legacy of broad and deep experience in ADMs (mostly XP):
  - One of the earliest adopters of XP
  - Applied to a wide variety of application types
  - For a very varied, distinguished and demanding client base

- This experience has resulted in successive refinement of the practices into an agile software development discipline known as Agile+. 

- AgileTek’s Agile+ adds important practices to XP
  - Enabling it to scale
  - Enhancing XP’s inherent ability to manage risk
  - Increasing its synergy with the S/W CMM
  - Extending its capabilities at the front end to provide a better understanding of the problem(s) being solved
  - Enabling it to work in a regulated environment
A Sampling Of Our Legacy Clients
Agile+ = XP + ∆

+ Business Process Analyses (BPAs)
+ Risk-Based Situation Audits (RBSAs)
+ Componentized Architecture
+ Wall Gantts
+ Automated Contract and Regression Testing
+ Story “Actors”
+ Automatic document generation
- Strict Pair Programming
- 40-Hour Work Week Restriction

+ Flexibility to meet special needs
Recent Example – The “Odyssey” Project
- Complex Industrial Automation Application For Premiere Supplier
- 400,000+ ESLOC
- Written in C++
- 15-Month duration
- Delivered a month and a half early
- Productivity Rate = 43 ESLOC/Coding Hour
- Over 2400 defects were found and fixed before delivery
- Delivered defects were 0.5/KLOC

Experience with ADMs Over five-year period:
- 20+ Projects across highly varied application domains
- Productivity averaged 35 ESLOC/Coding Hour
- Productivity Index (PI) = 22

Manzo, John, Odyssey And Other Code Science Success Stories.
CrossTalk, October 2002
The Myths Surrounding XP

- XP Practices Are Unproven
- XP Adds Risk To A Project
- XP Doesn’t Scale
- XP Is Incompatible With The CMM
- XP Won’t Work In A Regulated Environment
Proven or Unproven?

Belief: ADM practices are unproven

Reality: Most of the essential practices of ADMs have been successfully used for decades to achieve remarkable results
Would Agile Development Methods have worked on one of the most complex and successful systems ever delivered to the DOD? (Rhet.)
At The Core Of AEGIS Combat System Development, Was One Of The Key Tenets Of XP/ Agile+

“Design A Little Learn A Lot”

Build A Little Test A Little

RADM Wayne E. Meyer

Radm Wayne E. Meyer
Intercepting Changing Requirements

Traditional

Agile+

AgileTek.
The “Waterfall” Problem

- Fallacies of waterfall
  - Premised on the ability to define the system’s requirements up front, with a level of precision and exactness that is generally impossible
  - Assumes that once the requirements are defined they will not change.
  - Puts system tests and (by implication) user tests at end
  - After investing in full construction, one can find:
    - Awkwardness for users
    - Unacceptable performance
    - Dangerous susceptibility to user input
    - System does not meet its intent
  - Software remains an abstract entity too long - talking about what it could be not what it will be next

Reason waterfall is still used today is that in 1975 it became enshrined in DOD-STD-2167 (DOD specification for all military software) - ensuring its existence long after most thoughtful practitioners recognized its inadequacy and abandoned it.
<table>
<thead>
<tr>
<th><strong>XP Tenet</strong></th>
<th><strong>AEGIS FCS Practice</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning Process</td>
<td>Build Planning</td>
</tr>
<tr>
<td>▪ Customer defines business value of features</td>
<td>▪ Navy prioritized features and functions</td>
</tr>
<tr>
<td>Iterative Development</td>
<td>Iterative Development</td>
</tr>
<tr>
<td>Simple Design</td>
<td>Evolving Prototypes</td>
</tr>
<tr>
<td>▪ Deliver the simplest program that meets the current requirements</td>
<td>▪ Create a simple prototype and evolve it into the final Build</td>
</tr>
<tr>
<td>Relentless Testing</td>
<td>Relentless Testing</td>
</tr>
<tr>
<td>▪ Validate software at all times</td>
<td>▪ Continuous validation</td>
</tr>
<tr>
<td>▪ Write tests before code</td>
<td>▪ Tests written before code</td>
</tr>
<tr>
<td>▪ Acceptance tests in advance</td>
<td>▪ Acceptance tests known in advance</td>
</tr>
</tbody>
</table>
### XP Tenet vs. AEGIS FCS Practice

<table>
<thead>
<tr>
<th>XP Tenet</th>
<th>AEGIS FCS Practice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refactoring</td>
<td>Code Cleanup</td>
</tr>
<tr>
<td>- Keep the software clean – ready for any change that comes along</td>
<td>- Frequent cleaning and streamlining of the code to enhance performance</td>
</tr>
<tr>
<td>Pair Programming</td>
<td>Two-on-a-pencil (as required)</td>
</tr>
<tr>
<td>Collective Ownership</td>
<td>Collective Ownership</td>
</tr>
<tr>
<td>- All the code belongs to all the programmers</td>
<td>- All the code belongs to all the programmers – Unit Development Notebooks</td>
</tr>
<tr>
<td>Continuous Integration</td>
<td>Frequent Integration</td>
</tr>
<tr>
<td>- Integrate and build the system multiple times a day</td>
<td>- Integrate once a week</td>
</tr>
</tbody>
</table>
### XP Tenet vs. AEGIS FCS Practice

<table>
<thead>
<tr>
<th>XP Tenet</th>
<th>AEGIS FCS Practice</th>
</tr>
</thead>
<tbody>
<tr>
<td>40-Hour Week</td>
<td>52-Hour Week</td>
</tr>
<tr>
<td>On-Site Customer</td>
<td>On-Site Customer</td>
</tr>
<tr>
<td>- Determine requirements</td>
<td>- Interpret requirements</td>
</tr>
<tr>
<td>- Set priorities</td>
<td>- Set priorities</td>
</tr>
<tr>
<td>- Answer questions</td>
<td>- Answer questions</td>
</tr>
<tr>
<td>Coding Standards</td>
<td>Coding Standards</td>
</tr>
<tr>
<td>- To share ownership, all the programmers need to write the code in the same way</td>
<td>- Coding standards understood and enforced</td>
</tr>
<tr>
<td>Metaphor</td>
<td>Actual</td>
</tr>
<tr>
<td>- Guide development with a simple shared story of how the overall system works</td>
<td>- Graphical simulation showing how the system worked</td>
</tr>
<tr>
<td><strong>Agile+</strong> Tenet</td>
<td>AEGIS FCS Practice</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>Business Process Analysis</td>
<td>Systems Analysis</td>
</tr>
<tr>
<td>Delphi Estimation</td>
<td>Delphi Estimation</td>
</tr>
<tr>
<td>“Componentized” Architecture ▪ One-to-one mapping of problem and program space</td>
<td>“Componentized” Architecture ▪ One-to-one mapping of problem and program space</td>
</tr>
<tr>
<td>Wall Ganttts</td>
<td>Daily Stand-Up Meetings</td>
</tr>
<tr>
<td>Automated Contract and Regression Tests</td>
<td>Manual Regression Tests and Runtime “Jails” and “Fortresses”</td>
</tr>
<tr>
<td>Story Actors</td>
<td>--------------------</td>
</tr>
<tr>
<td>Automatic Document Generation</td>
<td>Automatic Flowchart Generation</td>
</tr>
</tbody>
</table>
None of the activities and practices of XP/ADMs are totally unique
Other methods use iterative development (e.g., Spiral)
Emphasis on the key tenets, makes XP/ADMs different from most traditional development methodologies – but does not make them undisciplined nor unproven
“XP frightens or angers some people who encounter it for the first time. None of these ideas are new. Most are as old as Programming. There is a sense in which XP is conservative - all its techniques have been proven over decades (for the implementation strategy) or centuries (for the management strategy)”

Kent Beck
Risk?

Belief: ADMs add risk to a project

Reality: Risk identification and mitigation are inherent in the practices of most ADMs
The greatest risk in software development is not knowing what you don’t know until it’s too late…

Every Agile project begins with an RBSA; then,

Though short iterations, XP/ Agile provides a means to:
- Challenge assumptions
- Make informed choices and decisions
- Evolve understanding
- …early and often

Moreover,
- Project progress is continuously visible to the developers
  - Wall Ganttts
  - Daily Stand-Up Meetings
- … and to our clients
  - Placed at the project’s center
  - Frequent reviews

Unlike some methodologies where Risk Management is “bolted on”, it’s an integral part of XP/Agile
Why It’s Thought That XP Won’t Scale...

Belief: Lack of documentation prevents scaling

Reality: The real problem is communication
The Communication Problem…

\[ p = \text{number of people} \]
\[ i = \text{number of possible interrelationships} \]

\[ p = 3; i = 3 \]
\[ i = \frac{p(p-1)}{2} \]

\[ p = 5; i = 10 \]

\[ p = 8; i = 28 \]
Communication Overhead Is A Problem Regardless Of Methodology

- The “Traditional Solution” goes back centuries: *Divide et Impera*
- In Software Development, we’ve added the concept of:
  - Functional modularity
  - Isomorphic mapping
  - Interface Specifications
- *Agile* makes liberal use of these concepts, and...
- Emphasizes team visibility
An Existence Proof (Almost)

Ericsson Communications has been using an XP-like methodology on a:
- Globally distributed project
- Requiring 1 Million Programmer Hours (~ 480 person-years)

They’ve had to make some refinements (feature teams and module architect) to enable scaling - much as we did in evolving Agile

Thus far they are reporting very encouraging results

Karlsson, Even-André and Andersson, Lars-Göran, Ericsson Telecommunications Systems AB, Stockholm, Sweden
Is Agile Compatible With the S/W-CMM?

Belief: ADMs are incompatible with the S/W-CMM

Reality: ADMs provide excellent ways to achieve the goals of KPAs through Level 3
“The argument that CMM’s ideal of a rigorous, statistically stable process is antithetical to XP is unconvincing”

“XP has disciplined processes and the XP process itself is clearly well defined”

“The SW-CMM tells organizations what to do in general terms but does not say how to do it”

“XP is a set of best practices that contains fairly specific how-to information”

“Modern software projects should capture XP values, regardless of how radically their implementation differs from XP’s”

“XP and CMM are complementary, and can even create synergy”

Meeting Level 2 KPAs (SW-CMM)

- Requirements Management: Excellent
- Project Planning: Good
- Project Tracking: Excellent
- Acquisition Mgmt: N/A(?)
- Quality Assurance: Excellent
- Configuration Mgmt: Required
Meeting Level 3 KPAs
(SW-CMM)

- Org. Process Focus
- Org. Process Definition
- Integrated SW Mgmt
- Product Engineering
- Peer Reviews
- Org. Training Program

[1] These can come for “free” from Organization-wide programs
- Quantitative Process Management (a way to measure the results of following a software process)
- Software Quality Management (achieve specific software quality goals)
ADMs In A Regulated Environment?

Belief: ADMs won’t work in a regulated environment

Reality: ADMs can be excellent at providing the essence of what regulators require
Relevant Regulatory Documents

- CFR 23 Part 11 (Electronic Records/Signatures)
- CFR 23 Part 58 (Labs)
- CFR 23 Part 820 (Med Devices)
- *FDA Guide To The Inspection Of Computerized Systems In Drug Processing*
- *Guide To The Inspection Of Software Development Activities*
- *Guidance For The Content Of Premarket Submission For Software Contained In Medical Devices*
- *General Principles Of Software Validation: Final Guidance for Industry and FDA Staff*
“There are a variety of life cycle models such as: waterfall, spiral, evolutionary, incremental, top-down functional decomposition (or stepwise refinement), formal transformation, etc.

Medical device software may be produced using any of these or other models, as long as adequate risk management and feedback processes are incorporated in the model selected.”

Source: Guidance for FDA Reviewers and Industry
Essence Of FDA Software Guidelines

- Controlling Risk
- Requirements Traceability
- Validation and Verification
<table>
<thead>
<tr>
<th>FDA Requirements</th>
<th>How Satisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>Risk Control</td>
<td>RBSAs</td>
</tr>
<tr>
<td></td>
<td>Short Iterations</td>
</tr>
<tr>
<td></td>
<td>Daily Stand-Up Meetings</td>
</tr>
<tr>
<td></td>
<td>- Visibility</td>
</tr>
<tr>
<td></td>
<td>- Real-time Assessment</td>
</tr>
<tr>
<td></td>
<td>- Total Team Awareness</td>
</tr>
<tr>
<td>Requirements Traceability</td>
<td>Tests written before code</td>
</tr>
<tr>
<td></td>
<td>- Tests based on requirements</td>
</tr>
<tr>
<td></td>
<td>- User signoff</td>
</tr>
<tr>
<td></td>
<td>- Code written to pass tests</td>
</tr>
<tr>
<td></td>
<td>User stories and test cases tracked in ClearCase™</td>
</tr>
<tr>
<td>Verification And Validation</td>
<td>QA Team Independent of Development</td>
</tr>
<tr>
<td></td>
<td>Contract test harnesses</td>
</tr>
<tr>
<td></td>
<td>Test first, last and always philosophy</td>
</tr>
</tbody>
</table>
Controlling Risk With Agile+

- **Iterative Development**
  - Feedback early and often

- **Daily Stand-Up Meetings**
  - Makes risks immediately visible
  - Real-time assessment
  - Total team awareness

- **Risk Characterization**
  - Enables sharp focus to be maintained on worse risks
  - Visibility a pre-requisite to mitigation
Agile Requirements Traceability

- User Stories
- Master User Story List
- Tests written before code
  - Tests based on requirements
  - User signoff
  - Code written to pass tests
- Requirements and Test Cases tracked in ClearCase (or some other CM tool)
- Requirements Traceability Matrix (RTM) is easily generated if required
“Software Verification:” provides objective evidence that the design outputs of a particular phase of the software development life cycle meet all of the specified requirements for that phase. Software verification looks for consistency, completeness, and correctness of the software and its supporting documentation, as it is being developed, and provides support for a subsequent conclusion that software is validated.

“Software Validation:” confirmation by examination and provision of objective evidence that software specifications conform to user needs and intended uses, and that the particular requirements implemented through software can be consistently fulfilled.

Source: General Principles of Software Validation
“A general application of several broad concepts can be used successfully as guidance for software validation. These broad concepts provide an acceptable framework for building a comprehensive approach to software validation.”

Source: *General Principles of Software Validation*
Q/A Team independent of Development
Test first, test last, test always philosophy
Contract Test Harness
– Captures and implements the very essence of FDA V&V

In God We Trust… Everybody else bring a Contract Test Harness!!
<table>
<thead>
<tr>
<th><strong>FDA Audit Item</strong></th>
<th><strong>FDA Requirement</strong></th>
<th><strong>How Agile+ Complies</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Operating Procedures (SOPs)</td>
<td>Recommendation: Follow a set of standards</td>
<td>Agile+ embodies standard practices from CMM, IEEE, and others</td>
</tr>
<tr>
<td>Evidence that procedures are followed</td>
<td>Existence of key documents</td>
<td>- Requirements Spec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Architecture Diagram</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Design Spec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Development Plan</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Validation, Verification &amp; Test Plan</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Traceability Matrices (available on request)</td>
</tr>
</tbody>
</table>
## FDA Audit Items (2 of 2)

<table>
<thead>
<tr>
<th>FDA Audit Item</th>
<th>FDA Requirement</th>
<th>How <strong>Agile+</strong> Complies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device Hazard Analysis (DHA) (For Medical Devices Only)</td>
<td>Conduct DHA to determine <em>Level of Concern</em> of software</td>
<td>Not included in Agile+. However, straightforward for an Agile+ organization to add a special DHA procedure if one does not already exist</td>
</tr>
</tbody>
</table>
Summary

- Decades of experience in software development
- Five years experience in applying ADMs
- More than twenty ADM projects varying widely…
  - Application domains
  - Complexity
  - Size
  - Technologies
  - Client base
- Resulted in lessons learned and successive refinements of a disciplined and flexible XP-based methodology we call Agile+
- This methodology extends “pure” XP beyond its real and imagined limitations and continues to succeed in producing high quality, rapidly developed, cost effective software
Thank You

John Manzo
Principal
jmanzo@agiletek.com
847/640-1805

AgileTek LLC
934 S. Golf Cul de Sac
Des Plaines, IL  60016
www.agiletek.com
Meeting at the Wall-Gantt
The Wall Gantt – How it Works

- Team members select stories and tasks to work on
- Team members “hang tasks” and drive out dependencies
- Team members “load balance” the iteration’s effort
The Wall Gantt – Why it Works

- Provides visibility into the current progress
- Provides clear ownership of development efforts
- Encourages accountability for assigned work
- A tool everyone can use
- The backdrop for Stand-Up Meetings, the center of the project universe

Sample Wall Gantt
Story Actors Bring The Requirements to Life

Roy the Supervisor
- Coordinates machine acceptance
- End user
- Configures user access
- 16th grade education

Diaz the Operator
- Speaks Spanish only
- Reads international symbols
- Works at Acme Catapult in Phoenix
- Second shift
- End user
- 6th grade education

Harry the Hacker
- Works at Acme Catapult
- Underpaid, dissatisfied desktop support tech
- End user
- 14th-16th grade education

Mike the Mechanic
- Plant maintenance
- End User
- 12th grade education

Dave the Designer
- Develops HMI panels
- Designs and programs user screens
- Doesn’t know programming
- OEM personnel
- 12th-16th grade education

Paul the Panel Builder
- OEM guy
- Builds panel in shop
- Puts project on so they can ship the machine
- Dave designs it, Paul builds and tests it
- 12th grade education

Hans der Designer
- Same as Dave, but in German
- End user
- 18th grade education

Toni the Technician
- OEM field serviceperson
- Sets up machine
- Tweaks panels for Roy
- 12th grade education

Josh the Co-op
- Knows VB
- Going to college
- OEM Personnel
- 14th grade education
How It Works (1 of 3)

Discovery Thru Business Process Analysis

Understanding the customer’s Business Environment
- Business basics
- Business problem
- Business solution

Understanding the customer’s Technology Environment
- Support capabilities
- Existing infrastructure
- Strategic technology vision

Solution

Logical Architecture
- Functional Breakdown
- Platform
- Component Technology
- Application Infrastructure
- Development Tools

Physical Architecture
How It Works (2 of 3)

Discovery

Solution

User Story Gathering Session

- User Story
- Gathering Session

Master User Story List

Iteration Planning

- Iteration Planning Design
- Day(s)

Design Day(s)

- Business Analysts
- Developers

Component Design

Specifications

Code

User Story

- Screen designs
- Notes
- User workflow
- Assumptions
- Data I/O

Component Design Specifications
“The Best Strategy is the one that preserves the most options, while actually solving your most pressing problems.” Kent Beck

Story “Sediment”

- Challenge Assumptions
- Inform Choices/Decisions
- Evolve Understanding
Productivity Index

- Productivity Index (PI) - a scale developed by Quantitative Software Management (QSM) depicting the overall development productivity associated with an organization's software development efforts. QSM was founded by Larry Putnam, a pioneering leader in software development metrics.

- The proven best practices used by AgileTek in our Agile+ software development methodology have produced some of the highest Productivity Indexes in the industry¹.

[¹]Our most recent audit revealed an overall average Productivity Index (PI) of 22 (as defined by Larry Putnam in his Measures of Excellence: Reliable Software On Time, Within Budget). This index is a management scale corresponding to the overall process productivity achieved by an organization during the main software build. An index of 25 has been considered among the highest ever recorded.