An Introduction to Agile Software Development

David Moskowitz
Productivity Solutions, Inc.
Agenda

- How we develop software
- BRUF, BDUF, and DWIM
- What's wrong with relay races, or Why Agile?
- What is Agile development?  Prior art?
- When to use Agile
- Requirements: the requested system *versus* the needed system
- How to introduce Agile
- Answer your questions
How We Develop Software

- Determine the need
- **Define/capture** business requirements
  - Note the vocabulary used...
- Design the system, assign the development team and...
  - Harlan Mills: buggy programs
  - Wright Brothers model
BRUF & BDUF

- Big Requirements Up Front
  - No design until all requirements known, defined and captured

- Big Design Up Front
  - No code until the design is done, complete and final

- Works IFF you're religious about sticking to it... but...
  - Request *versus* need!
Waterfall/Relay Race Model

- Waterfall is strictly sequential
  - Strict sequence of stages
  - Mainframe-based; doesn't use modern tech
  - Doesn't accommodate or tolerate change without added cost and time
- A better conceptual model: relay race
  - Isn't a solution for the problem, it's merely a different way of expressing it
Waterfall/Relay Race Model

- Relay races: overlap before baton passed
- Waterfall and relay present a predetermined path and sequence.
  - Neither accommodate change
  - Can’t go up the falls or backward in the race
- Multiple attempts to move away including Boehm's spiral model
  - Iterative: planning, risk analysis, development, and customer evaluation
Why Agile

- Thomas Friedman in *The World Is Flat*: "...the flattening of the world has happened faster and changed rules, roles, and relationships more quickly than we could have imagined."

- How do we respond?
- How do we adapt?
- Is your current software development competitive in a flat world?
Why Agile?

- The height of insanity: repeat the same steps & expect different results.
- New approaches and ideas don’t just occur out of the void.
- New approaches evolve from conditions in which old ideas no longer appear to work.
Agile Development Basics

Agile Manifesto (http://www.agilealliance.org/)

- **Individuals & interactions**
- **Working software**
- **Customer collaboration**
- **Responding to change**

Processes & tools
Comprehensive documentation
Contract negotiation
Following a plan

"Uncovering better ways to develop software. Through these efforts, while there is value to the things on the right, we value the left more."
Agile Principles from DSDM 1

1. Active user involvement is imperative
2. Agile teams make decisions
3. Focus on frequent, rapid delivery
4. Acceptance of deliverables: biz purpose
5. Iterative and incremental development

Agile Principles from DSDM 2

6. All changes are reversible
7. Requirements baselined at a high level
8. Testing integrated into process
9. Collaborative and cooperative approach between all stakeholders is essential

Essence of Agile

- The essence part 1: Eliminate waste including
  - Partial Work
  - Additional features
  - Context switching
  - Bugs/defects...
  - Implementing out of date requirements.
  - Unnecessary processes
  - Wasted motion (paper shuffle, etc.)
  - Waiting for something/someone
  - Communication latency

- The essence part 2: Delivering value
  - What generates value?
  - Helps identify waste, too.
Agile as an Alternative

- The ultimate goal is working software, sooner that addresses need vs. request
  - This is an important driver
  - Old model: delivery long(?) after conception
- Along the way eliminate waste, reduce costs, reduce time to market, and increase quality.
  - Agile requires more testing sooner
- Different methodologies... different sized methodologies different size projects
What Is Agile Development?

- Requirements driven
  - Customers, end-users, stakeholders, during – not before
  - Dynamic Feedback during development, – not after

- Deliver value
  - List the 10 to 20 most important organizational activities
  - Put yourself in the customer's shoes, rate the activity 1 (customers don't care – no value) to 5 (absolutely must have – high value)

- Basis in manufacturing (think JIT)
  - Prior art: automobile industry (Toyota)
What is Agile Development?

- Adaptive
- Goal Driven
- Iterative
- Lean
- Emergent approach

- Self-organizing
- Empowering
- Collaborative
- Active customer involvement
- Frequent delivery
- Incremental
- Time-boxed
- Disciplined
- Continuous integration
- Feature driven
Agile Development Includes

- Continuous innovation and integration
  - Deliver current customer requirements
- Decreased communication latency
  - Stakeholders actively involved in development
- Testing integrated into the development process
  - Doesn't need or include a separate Q&A cycle
  - Continuous integration, multiple builds
- Product adaptability
  - Doesn't preclude future requirements
  - Adapts to changing requirements
Agile Development Includes

- People and process adaptability
  - Respond rapidly to change
- Short delivery cycles
  - Find smaller "sub-features" to deliver something
  - Meet market window
  - Improve ROI
  - Reduce TNWIM
- Reliable results
  - Support business growth and profitability
  - Does it contribute to business value?
What's Different?

Traditional

Fixed: Functionality / requirements
Variable (estimates):
  People & time (Cost & schedule)

• The goal is to deliver something!
• Priority determines increment scope.
• A late requirement change in an agile project can be a competitive advantage.
• 60+% required features rarely or never used!

Agile

Variable: Functionality / requirements
Fixed: People & time

Colorado Software Summit: October 23 – 28, 2005
© Copyright 2005, Productivity Solutions, Inc.
What Different (Visual)?

**Predictive**
- Plan Driven
- Value or Vision Driven
- Iterative
- Phased-based

**Constraints**
- Requirements
- Cost
- Schedule

**Adaptive**
- Process
- Cost
- Value or Vision Driven
- Features

**Estimates**
- Predictive plan
- Adaptive process
- Phased-based

---

David Moskowitz — An Introduction to Agile Software Development
What If…?

- What is the difference between a need and a thneed?
  - Thneed => something you think you need
  - What is the difference between the *requested system* and the *needed system*?

- When are requirements defined?
  - Before coding starts or as part of the process?
  - If you're thinking about "capturing" requirements, it's the wrong mindset!!!


What If...?

- ... requirements aren't baselined at a high level that allows detailed requirements to evolve during development, in concert with users (stakeholders)
- The antithesis is an...
  - ...attempt to define a (nearly) complete set of requirements at the beginning.
  - It is the path over the waterfall
Prioritize Requirements

- Synthetic voting
- Must, Should, Might, NITI
  - Must: fundamental to system. Missing, application will be worthless
  - Should: important and may be promoted to must at a later time. Not absolutely critical to the system (at this time), can be used without.
  - Might: more easily left out of increment
  - NITI: not in this increment/iteration
What Happens When?

- Feasibility determines whether or not Agile is appropriate in the organization for this project.
  - Determines if all principles fit the project
  - Produces initial outline/plan & optional prototype
- Business study sets baselined high level requirements (functional and non-functional) and their initial priority (set by business need)
  - System architecture definition (initial platforms, & architecture, etc). This is allowed to change.
When to Use Agile

- Agile isn't a solution for everything
  - Will the culture support or torpedo it?
- Is functionality visible at the UI?
  - Reports and screens
  - Users verify that software meets needs
  - Agile dictates user involvement
- Can all classes of users be identified?
  - Need complete coverage by users
When to Use Agile

- Large application?
  - If so, can it be split into smaller deliverables?
- Is the project REALLY time constrained?
  - If it isn't, there is no need to compromise deliverables/features, everything must be delivered.
- Requirements flexible & defined only at high level?
  - "Complete" understanding of all deliverables?
    - If yes, you may have too much information.
- Only if customers are & can be available
How to Do Agile??!?!!

- Are you sure you want to?
- It’s really difficult to do – in part, because
  - It requires testing and...
  - Most programmers don’t like to do testing – as an integral part of development.
  - It requires customers...
- Agile development is much more disciplined...
  - Planning, test, code, with continuous integration (either once or multiple times per day)
  - Hard for many groups because of the discipline required.
  - Can be taught/learned
How to Introduce Agile

- Understand the organization
  - Bureaucratic: how quickly, how many steps
  - Hierarchical: tolerate empowerment
  - Innovative: today's cure changes
  - Technical focus: infrastructure suitable
  - Customer focus: with a service mindset
  - Willing to change: how does organization accommodate change
Introduce Agile

- Must be planned and managed just like any other project
  - Don't just "do" Agile

- Gradually (though fiat might be required)
  - Agile is a people-oriented process
  - Get the right people
    - People who want to be involved!
  - Empower them to make decisions

- Are you able to ask the right questions?
Introduce Agile

- Investigate options
  - Crystal, DSDM, RUP, Scrum, XP... ???

- Identify a suitable pilot project
  - Make sure resources are available
    - People and infrastructure

- Set the stage
  - Let people know what to expect

- Define/refine the business value
Introduce Agile

- Start the project
  - Constant prototyping with active user involvement (joined at the hip)
- Monitor progress and process
- What are we learning?
- What did we learn?
- How do you know?
- Are you sure?
Have an Agile Coach

- Coach is not part of any programming team
- Coach should be technically knowledgeable
- Coach responsibility: process first, problem domain second
  - Coach’s role is one of conversation, not coding
  - Coach’s role is to keep the team on track, not be the CTO of the project
  - Coach’s role is assure team can continue development without the coach
- Game mediator
Process

- In most cases agile process doesn’t matter
- Throw out $\frac{3}{4}$ of the (heavy weight) process and get a net improvement in the project and get better quality delivery
- Does infrastructure support strategy?
  - Continuous integration
  - More testing, sooner
Process All Different

- Which is right?

- Common differences between success & failure
  - Success didn’t have a lot of process,
  - Did have co-location (and a customer in the room),
  - Communication (including with the customer),
  - Early delivery

  - The failures (at agile) got so caught up in process that deliveries didn't happen (or didn't happen as often as they should)

- Use of a particular agile methodology doesn’t distinctly correlate with success or failure

  - Pick one that is right for the organization
Success Factors

- Pressure / need for change
- Clear understandable shared vision
- Capacity for change
  - Ability to tolerate a degree of chaos that is part of the process
- Clear first steps / actions
- Customer in the room with developers
- Guidance or help for early projects
- Will asking the difficult questions cause...
Success Factors

- Have the right people (& customers?)?
  - Assign highly skilled, flexible non-rigid people
  - Do they have the right thought process?
  - Are they dedicated & motivated to try something new and see it through?
    - Is there management support?
    - Is there some passion about the possibilities?

- Are the business drivers (value proposition) clearly understood?
It's the Process…

- There isn't a Q&A cycle after development
  - Agile tests more often and sooner
  - Not waterfall!
- End users must be involved during, not after
- If you are PMing and an iteration is late
  - Agile isn't the problem!
  - You aren't the problem, either!
  - Good thing, not bad: address problem earlier
    - Problem might be (and remain) hidden until much later when it would be more expensive to address
Rock the Boat

- In most organizations there really isn't a way to introduce agile without some objection.
- First projects should be important, but not bet the mortgage, mission critical.
  - Important so that there is impetus to complete
  - Not mission critical so that business continues
- Agile development is iterative development
  - Small functional pieces that do something
Really Rock the Boat

- How much documentation or analysis is enough and how much is too much?
- What is the purpose for documentation?
  - Allow new team members to catch up with us
  - Video white board sessions as documentation
  - Audio only, secondary
- Document process
- Defend time devoted to documentation
7 Deadly Sins

- More feedback, not less
- Run tests before/during development, not after
- More code, not more documentation
- Requirements evolve, not BRUF
- User involvement in code, not planning
  - From on-high (BR/DUF) is not agile
- Web front end & try new ideas, not big-bang
- Anticipate change as a competitive advantage, not a cost burden nor a reason for delay
Concept: Agile == Game

- Cooperative game, lets people discuss what happens, "If we do X..."
- Resource limited cooperative game
- Moves consist of communicating
  - All that there is is invention and communication
  - Finite, goal directed
Fallout from the Game

- People make sense of what they’re doing
- Become playful about what they are doing.
  
  ➢ Take that same vocabulary (the game vocabulary) into a live project and people make sense of what they are doing
  
  ➢ Sense of past with projections into the future.
  
  • Almost to the point of getting playful and inventing: "That would be (is) a good move."
Summary

- Relay race development doesn't work
- Agile isn't *the* solution to every problem
- Agile development focuses on delivering the needed system that meets dynamic business requirements
- Agile eliminates waste
- Agile means people and process changes
  - Biggest source of resistance: folks who insist BRUF, BDUF & Gantt are the ONLY way
Questions ? ? ?

If you don't ask, who will?

If not now, when?

There aren't any dumb questions.

The only dumb question is the one not asked!
References (Partial list)

Resources (Partial list)

- http://alistair.cockburn.us/crystal/wiki
- http://www.agilealliance.org/
- http://www.agilemodeling.com/
- http://www.controlchaos.com/
- http://www.dsdm.org/
- http://www.xprogramming.com/
Thank You

For more information:

David Moskowitz
Productivity Solutions, Inc.
147 Ashland Avenue
Bala Cynwyd, PA 19004
+1-610-664-7726
davidm2@usa.net