TASC

The Challenge of Agile Estimating

Christina Donadi
Heather Nayhouse

SCEA/ISPA National Conference, Albuquerque, New Mexico

June 2011

Agenda

- Overview of Agile Development
  - Importance of Agile Development
  - Definition
  - Key Differences between Agile and Traditional Development
  - Benefits
  - Manifesto (aka Principles)
- Agile Development Lifecycle
- Multiple Methodologies
- Agile Development from a Cost Perspective
  - General Metrics & Rules of Thumb
  - Major Cost Driver Comparison
- Cautions Against Agile Development

TASC

Overview of Agile Development

FROM NOW ON WE'RE GOING TO CHANGE HOW WE DO THINGS HERE. WE NEED TO BE AGILE.
WHAT DOES IT MEAN?
NOTHING REALLY CHANGES, EXCEPT THAT NOW MANAGERS, DESIGNERS AND ARCHITECTS HAVE AN EXCUSE FOR CHANGING THE REQUIREMENTS ON FRIDAYS AT 4 PM.

#113 - "AGILE DEVELOPMENT, EXAPLAINED" - BY SALVATORE IOVENE, FEB. 21ST 2009 HTTP://WWW.GEEKHEROCOMIC.COM/

Importance of Agile Development

- "The enemy that the United States is fighting is unlike any enemy fought in the past, demonstrating different tactics, techniques, and procedures from those found in conventional warfare. To respond to that enemy, there is a greater need for speed, agility, and responsiveness." – Army General David H. Petraeus, February 2010[1]
- 2010 National Defense Authorization Act mandated implementing the "agile model" for IT acquisitions[2]
- Understanding and acceptance throughout the entire DoD community is still needed[3]
  - Opportunity for the government and contractors to become involved and knowledgeable
- Increasing need for quick turnaround capabilities[4]
  - Response to 9/11
  - U.S. Air Force and U.S. Army

1. Anderson, Frank. "Adaptive, Responsive, and Speedy Acquisitions." Defense AT&L. Defense Acquisition University, Jan. 2010. Web. 26 Feb. 2011.
2. Richard, Cheng K. "On Being Agile." 23 Sept. 2010. Web. 26 Feb. 2011. <http://www.nextgov.com/nextgov/ng_20100923_7965.php>.
3. Portelli, Bill. "Agile Practices Need to Evolve Dramatically in US Defense." Information Week. 17 Dec. 2010. Web. 26 Feb. 2011.
4. Reagan, Rex B., and David F. Rico. "Lean and Agile Acquisition and Systems Engineering. A Paradign Whose Time Has Come." Defense AT&L (2010): 48-52. Print.

Overview of Agile Development - Definition

- Agile software development refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams
- The development cycle is designed to produce short, potentially-shippable results (fully functional code and capabilities) at the end of each release
- The release implemented is a subset of the total capabilities that will be implemented when the application or system is complete
- Evolution of the plan, the scope, and the forecast of progress toward completion continue over the life of the release cycles
- Focuses on short development cycles to receive the benefit of early, concrete, and continuing feedback from customer and users
- Emphasizes working software as the primary measure of progress

Strives to set goals and plan the path of a project to a release while maintaining this flexibility and openness to change based on feedback and new information

*Cohn, Mike. Agile Estimating and Planning. Upper Saddle River, NJ: Prentice Hall Professional Technical Reference, 2006. Print.

TASC

Key Differences – Traditional vs. Agile

- With traditional software development methods:
  - User needs evolve beyond the original requirements by the time the product is delivered
  - Changes must be implemented via engineering change proposals (ECPs) after the delivery of the product
  - Large upfront effort to define requirements, come to a decision on a solution, and document the path forward
  - With agile development methods:
  - Decisions can be delayed and options preserved until the most information is available
  - Customer and users get working capabilities quickly and are able to suggest changes to design as needed during the course of development, rather than at the end
  - Produces less written documentation than other methods typically because attempts to perform “just enough” up-front design in order to minimize rework
  - It includes requirements analysis, development environment setup, and the establishment of an initial architecture for the product

It is a plan, **not** a commitment
Dating the design but not yet married to it!

TASC

Traditional Development

- Common software development methodologies used in industry

Software Life Cycle Methodology	Description
Waterfall	Conventional, "theoretical" methodology
Incremental	Breaks developent into clearly-defined, stand alone system increments
Evolutionary	Built to satisfy requirements as they evolve
Spiral	Risk based analysis of alternatives approach

*Cost Estimating Body of Knowledge, Module 12: Software Cost Estimating

Overview of Agile Development - Benefits

1. Allows Flexibility of Shifting Priorities
   - Ability to quickly change direction to provide a promising way forward to effectively handle the increased speed of change in an industry
   - Allows for optimization on a known goal over the short term as well as for large-scale change in project goals over the long term
2. Establishing Trust & Promoting User Ownership
   - Allows the customer and users to suggest changes to design as needed during the course of development rather than at the very end, where it may not be possible or, at the very least, will be more expensive to implement
3. Reducing Risk & Uncertainty
   - Agile methodologies fix requirements at the release level, rather than the project level
4. Conveying Information
   - Agile methodologies use a cross-functional team working within a series of sprints to build functionality incrementally to produce a software product
5. Supporting Better Decision Making
   - Allows the delaying of decisions not based on what you think will happen, rather decisions based on the reality of the situation (previous work)

Agile Development allows for **flexibility** in design requirements and **user ownership** of the end product

*Cohn, Mike. Agile Estimating and Planning. Upper Saddle River, NJ: Prentice Hall Professional Technical Reference, 2006. Print.

Overview of Agile Development –Manifesto

- Agile Development values:

Individuals and interactions **over** processes and tools

Working software **over** comprehensive documentation

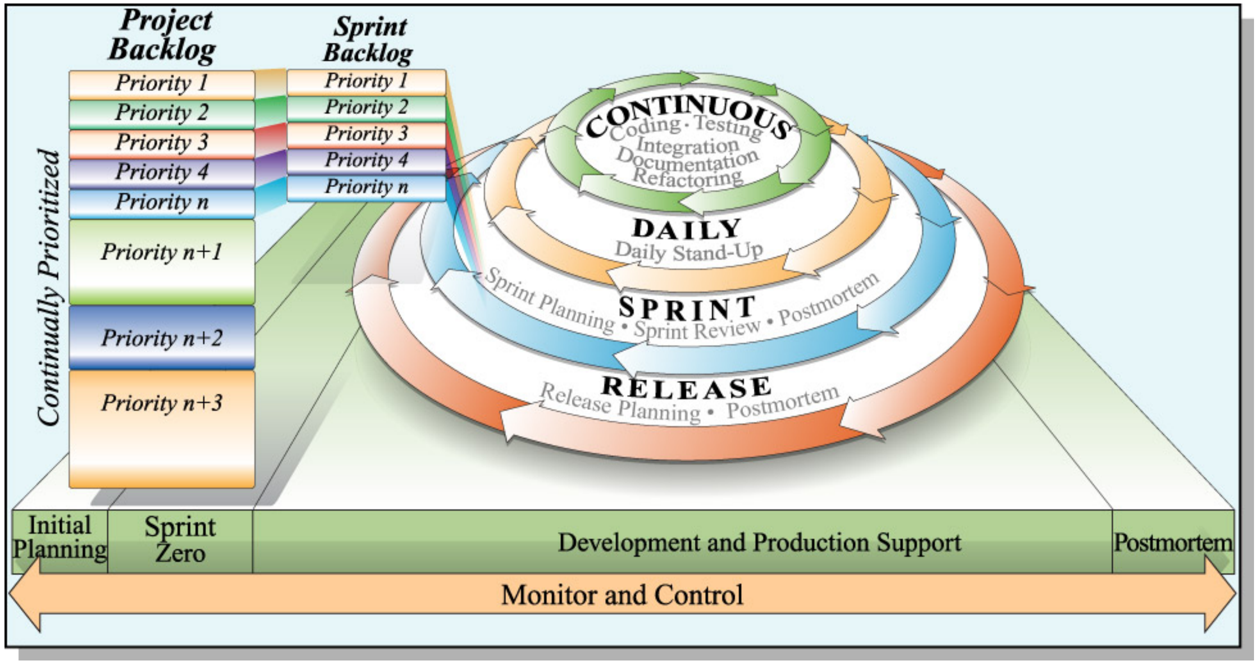
Customer collaboration **over** contract negotiation

Responding to change **over** following a plan

*Manifesto for Agile Software Development. Web. 07 May 2010. <http://agilemanifesto.org/>.

Agile Development Lifecycle

- The Agile project lifecycle can be thought of as an iterative process four levels deep:



1. Each release (a large-scale iteration in itself) features a planning phase, a development phase composed of a series of sprints, a release retrospective, and a transition to production
2. Each sprint features a planning phase, some number of days of product development, a sprint review for demonstration and stakeholder feedback, and a retrospective
3. Each day of product development features a planning phase known as a daily stand-up that reviews the previous day and plans the upcoming day
4. Each day of product development is a continuous loop of coding, testing, integration, and documentation

*Introduction to Agile Methodologies & Project Lifecycle, TASC, Inc., 2009-2010

TASC

Multiple Agile Development Methodologies

TASC

Multiple Methodologies

- Agile Modeling
- Extreme Programming (XP)
- Scrum
- Feature Driven Development (FDD)
- Agile Unified Process (AUP)
- Dynamic Systems Development Method (DSDM)

Multiple Methodologies

- Agile Modeling
  - Practice-based methodology for modeling and documentation of software-based systems
  - Intended to be a collection of values, principles, and practices for modeling software that can be applied on a software development project in a more flexible manner than traditional modeling methods
  - Agile Modeling is a supplement to other Agile methodologies such as:
    - Extreme Programming ("XP")
    - Agile Unified Process
    - Scrum
- Extreme Programming (XP) - Simplicity
  - A software engineering methodology created by Kent Beck and Ron Jeffries at Chrysler Corporation
  - Prescribes a set of daily stakeholder practices that embody and encourage particular values
  - Proponents regard ongoing changes to requirements as a natural, inescapable, and desirable aspect of software development projects
  - Adaptability to changing requirements at any point during the project lifecycle is a more realistic and better approach than attempting to define all requirements at the beginning and then expending effort to control changes to the requirements

*Jones, Capers. Estimating Software Costs: Bringing Realism to Estimating. New York: McGraw-Hill Companies, 2007. Print.

Multiple Methodologies

- SCRUM – Prioritized Business Value
  - Focuses largely on project management process, and largely does not address engineering practices.
  - Is a "process skeleton," which contains sets of practices and predefined roles with the main roles being:
    - "ScrumMaster", who maintains the processes (typically in lieu of a project manager)
    - "Product Owner", who represents the stakeholders
    - "Team", a cross-functional group of about 7 people who do the actual analysis, design, implementation, testing, etc.
- Feature-Driven Development (FDD) – Business Model
  - A model-driven short-iteration process that is driven from a client-valued functionality perspective and consists of the five following activities:
    - Develop Overall Model
    - Build Feature List
    - Plan by Feature
    - Design by Feature
    - Build by Feature

*Jones, Capers. Estimating Software Costs: Bringing Realism to Estimating. New York: McGraw-Hill Companies, 2007. Print.

Multiple Methodologies

- Agile Unified Process (AUP) – Manage Risk
  - Simplified version of the IBM Rational Unified Process (RUP) developed by Scott Ambler.
  - RUP is an iterative software development process framework
  - Describes a simple, easy to understand approach to developing business application software using agile techniques and concepts yet still remaining true to the RUP
  - Applies agile techniques including Test Driven Development (TDD), Agile Modeling, Agile Change Management, and Database Refactoring to improve productivity.
- Dynamic Systems Development Method (DSDM)
  - Methodology originally based upon the Rapid Application Development methodology
  - Iterative and incremental approach that emphasizes continuous user involvement
  - Goal is to deliver software systems on time and on budget while adjusting for changing requirements along the development process

*Jones, Capers. Estimating Software Costs: Bringing Realism to Estimating. New York: McGraw-Hill Companies, 2007. Print.

TASC

Multiple Methodologies - Comparison

	Strengths	Weaknesses
XP	✓Strong technical practices. ✓Customer ownership of feature priority, developer ownership of estimates. ✓Frequent feedback opportunities. ✓Most widely known and adopted approach, at least in the U.S.	✓Requires onsite customer. ✓Documentation primarily through verbal communication and code. For some teams these are the only artifacts created, others create minimal design and user documentation. ✓Difficult for new adopters to determine how to accommodate architectural and design concerns.
Scrum	✓Complements existing practices. ✓Self organizing teams and feedback. ✓Customer participation and steering. ✓Priorities based on business value. ✓Only approach here that has a certification process.	✓Only provides project management support, other disciplines are out of scope. ✓Does not specify technical practices. ✓Can take some time to get the business to provide unique priorities for each requirement.
FDD	✓Supports multiple teams working in parallel. ✓All aspects of a project tracked by feature. ✓Design by feature and build by feature aspects are easy to understand and adopt. ✓Scales to large teams or projects well.	✓Promotes individual code ownership as opposed to shared/team ownership. ✓Iterations are not as well defined by the process as other Agile methodologies. ✓The model-centric aspects can have huge impacts when working on existing systems that have no models.

*Derived from Multiple Sources; available upon request

TASC

Multiple Methodologies - Comparison

	Strengths	Weaknesses
AUP	✓Robust methodology with many artifacts and disciplines to choose from. ✓Scales up very well. ✓Documentation helps communicate in distributed environments. ✓Priorities set based on highest risk. Risk can be a business or technical risk.	✓Higher levels of ceremony may be a hindrance in smaller projects. ✓Minimal attention to team dynamics. ✓Documentation is much more formal than most approaches mentioned here.
DSDM	✓An emphasis on testing is so strong that at least one tester is expected to be on each project team. ✓Designed from the ground up by business people, so business value is identified and expected to be the highest priority deliverable. ✓Has specific approach to determining how important each requirement is to an iteration. ✓Sets stakeholder expectations from the start of the project that not all requirements will make it into the final deliverable.	✓Probably the most heavyweight project compared in this survey. ✓Expects continuous user involvement. ✓Defines several artifacts and work products for each phase of the project; heavier documentation. ✓Access to material is controlled by a Consortium, and fees may be charged just to access the reference material.

*Derived from Multiple Sources; available upon request

TASC

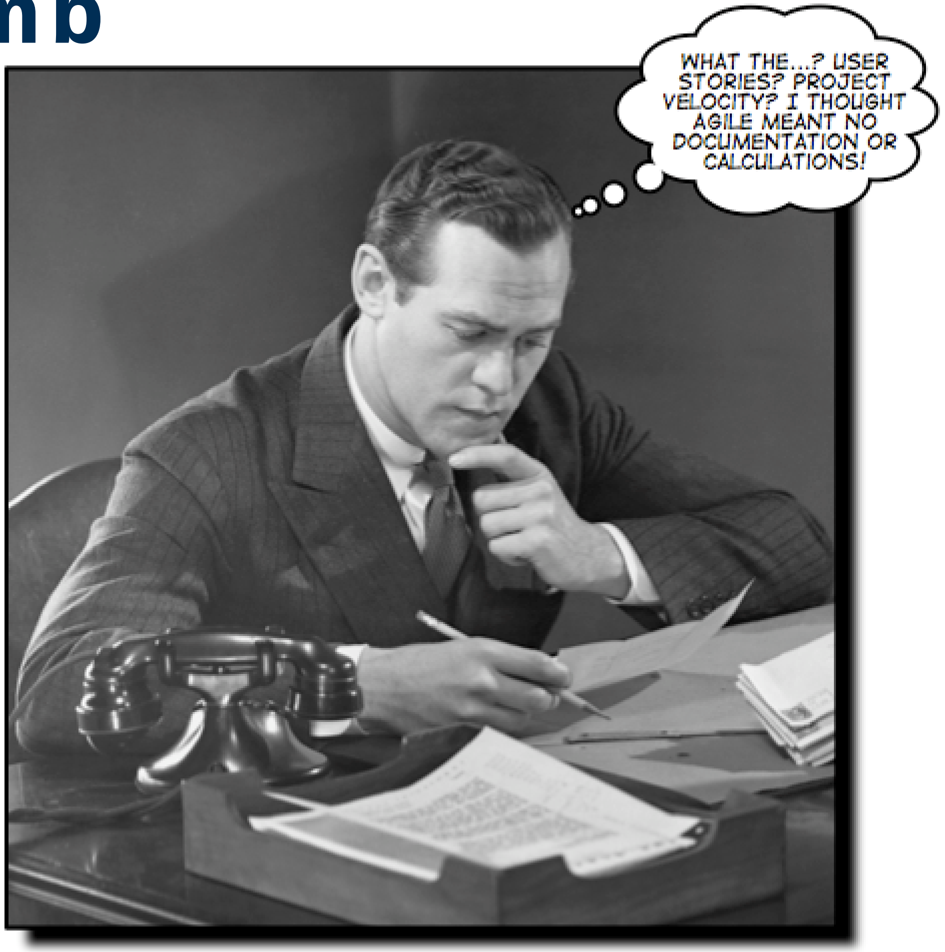
Multiple Methodologies - Comparison

Condition	XP	Scrum	FDD	AUP	DSDM
Small Team	√	√	X	X	√
Highly Volatile Requirements	√	√	√	-	X
Distributed Teams	X	√	√	√	X
High Ceremony Culture	X	X	-	√	√
High Criticality Systems	X	-	-	-	X
Multiple Customers / Stakeholders	X	√	-	-	X

which conditions favor (√), discourage (X), or are neutral (-) with respect to the specific conditions listed

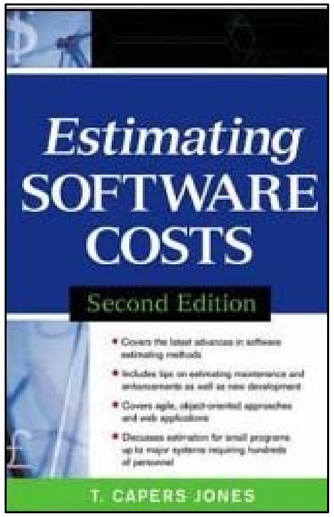
TASC

General Metrics & Rules of Thumb


WHAT THE...? USER STORIES? PROJECT VELOCITY? I THOUGHT AGILE MEANT NO DOCUMENTATION OR CALCULATIONS!

General Metrics & Rules of Thumb

- Development Productivity
  - US average is 10 function points per staff month, which is to 500 Java statements per staff month
  - Includes all activities from requirements through delivery
- Development Assignment Scope
  - Average amount of work assigned to one person
  - US average is about 150 function points (per assignment)
- Development Schedule
  - Grows longer as applications become larger
  - Approximated by raising the size of the application to a specific power
  - The US average is to the power of 0.4
- Defect Potential
  - Current US average is about 5 defects/ function point
  - Specifically broken into requirements, design, code, documents, and bad fixes – 1, 1.25, 1.75, .6, .4
  - Static value and does not change for small or large software applications
- Bad Fix Injection
  - When you fix a bug, you may introduce a new one
  - The US average is 7%
  - For large applications with error-prone modules, the bad fix injection rate has potential to be over 50%



*Jones, Capers. Estimating Software Costs: Bringing Realism to Estimating. New York: McGraw-Hill Companies, 2007. Print.

General Metrics & Rules of Thumb

- Defect – Removal Efficiency
  - The removal of the number of bugs found in development in addition to the number of bugs found after using the product for a pre-determined period of time before it becomes operational
  - Current US average is 85% (so 15% of bugs will be in the system once it's up and running)
  - Failing software projects are typically less than 80% efficient at bug removal
- Maintenance Assignment Scope
  - Average amount of legacy application that one person can maintain in the course of a year
  - Tasks including fixing latent bugs and performing minor updates
  - US average is about 750 function points ~= 37,500 Java statements
- Cost Per Function Point
  - Variable cost information; average US cost at $1200 per function point due to large differences in team composition, salary, etc.
  - Average cost for maintenance which is $150 /function point/calendar year
  - Note: range for development runs from less than $250 for some small agile projects up to more than $5000 per function point for some large defense systems
  - Note: range for maintenance runs from less than $50 - $600+/function point for maintenance

*Jones, Capers. Estimating Software Costs: Bringing Realism to Estimating. New York: McGraw-Hill Companies, 2007. Print.

On Average Metrics

	Traditional	Agile Development
Development Productivity	10 function points per staff month	36 function points per staff month
Development Assignment Scope	150 function points	100 function points
Development Schedule	Approximated by raising the size of the application to a specific power to the power of 0.4	Approximated by raising the size of the application to a specific power to the power of 0.33
Defect Potential	5 defects per function point	3.5 defects per function point
Bad Fix Injection	7%	2%
Defect Removal Efficiency	85%	92%
Maintenance Assignment Scope	750 function points	1500 function points

*Jones, Capers. Estimating Software Costs: Bringing Realism to Estimating. New York: McGraw-Hill Companies, 2007. Print.

Major Cost Driver Comparison

	Traditional	Agile Development
$$$$$	Finding & fixing bugs	Finding & fixing bugs
$$$$	Producing Paper Documents	Meeting & Communications
$$$	Meeting & Communications	Coding
$$	Coding	Producing Paper Documents
$	Project Management	Project Management

Cautions Against Agile Development

Cautions against Agile Development

- Lack of Support
- Government Mandates
- Capability Maturity Model Integration (CMMI) Compliance
- Security requirements
- Different meanings of agile development
- Function points vs. Source Lines Of Code (SLOC)
- Major System Acquisition Programs
- Replacing paper specifications with face to face client meetings is not always suitable

Questions

TASC