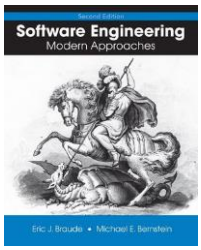
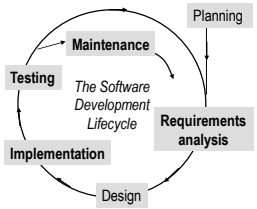
Software Engineering

Modern Approaches



Eric Braude and Michael Bernstein

Chapter 4: Agile Software Processes



Phases most relevant to this chapter are shown in bold

Learning goals of this chapter

- How did agile methods come about?
- What are the principles of agility?
- How are agile processes carried out?
- Can agile processes be combined with non-agile ones?

Agile Processes

- Prior to 2001, group of methodologies shared following characteristics:
  - Close collaboration between programmers and business experts
  - Face-to-face communication (as opposed to documentation)
  - Frequent delivery of working software
  - Self-organizing teams
  - Methods to craft the code and the team so that the inevitable requirements churn was not a crisis

Agile Alliance – Agile Manifesto

The Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

1. **Individuals and interactions** over processes and tools
2. **Working software** over comprehensive documentation
3. **Customer collaboration** over contract negotiation
4. **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Agile Principles

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

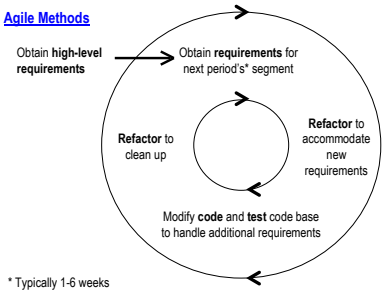
Agile Principles

- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

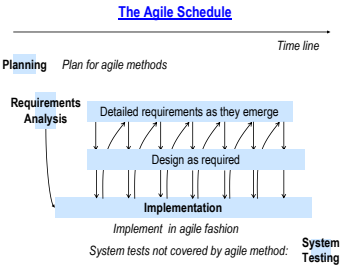
Agile Methods

Agile Processes MANIFESTO → RESPONSES:	1. Individuals and interactions over processes and tools	2. Working software over comprehensive documentation	3. Customer collaboration over contract negotiation	4. Responding to change over following a plan
a. Small, close-knit **team** of **peers**	y			y
b. Periodic **customer** requirements **meetings**	y		y	y
c. **Code-centric**		y		y
d. **High-level** requirements statements only			y	y
e. **Document as needed**			y	y
f. **Customer reps** work within team	y			y
g. **Refactor**				y
h. **Pair** programming and no-owner code	y			
i. Unit-test-intensive; Acceptance-**test-driven**		y	y	
j. **Automate** testing		y	y	

Agile Cycle



Agile Schedule



Extreme Programming (XP)

- Kent Beck, 1996
- Project at Daimler Chrysler
- Simple and efficient process

XP Values

The "Values" of Extreme Programming *1 of 2*

1. **Communication**
   - Customer on site
   - Pair programming
   - Coding standards
2. **Simplicity**
   - Metaphor: entity names drawn from common metaphor
   - Simplest design for current requirements
   - Refactoring

Beck: Extreme Programming Explained

XP Values (cont.)

The "Values" of Extreme Programming *2 of 2*

3. **Feedback** always sought
   - Continual testing
   - Continuous integration (daily at least)
   - Small releases (smallest useful feature set)
4. **Courage**
   - Planning and estimation with customer user stories
   - Collective code ownership
   - Sustainable pace

Beck: Extreme Programming Explained

XP Principles

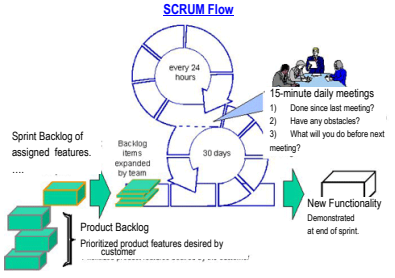
- Planning Process
- Small Releases
- Test-driven Development
- Refactoring
- Design Simplicity
- Pair Programming
- Collective Code Ownership
- Coding Standard
- Continuous Integration
- On-Site Customer
- Sustainable Pace
- Metaphor

SCRUM

- Developed in early 1990s
- Based on assumption:
  - development process is unpredictable and complicated
  - can only be defined by a loose set of activities.
- Development team empowered to define and execute the necessary tasks to successfully develop software

SCRUM

SCRUM Flow



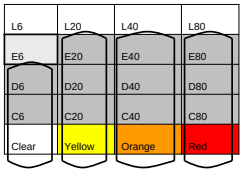
Quoted and edited from http://www.controlchaos.com/

Crystal

- Alistair Cockburn
- Family of agile methods
- Frequent delivery, close communication and reflective improvement

Crystal

Coverage of Different Crystal Methodologies



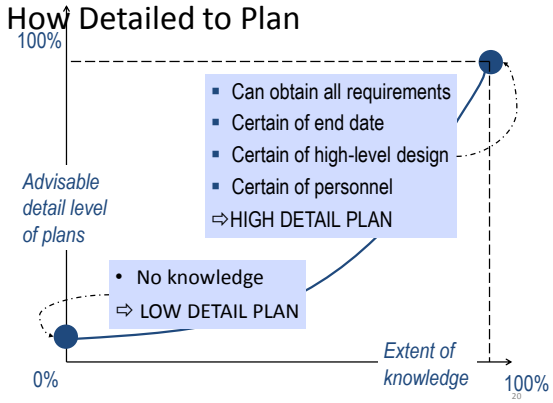
L = loss of life
E = loss of essential moneys
D = loss of discretionary moneys
C = loss of comfort

Adapted from Crystal Clear, Alistair Cockburn

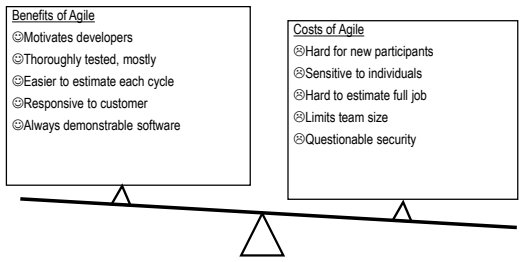
Crystal Properties

- Frequent delivery
- Reflective improvement
- Close communication
- Personal safety
- Focus
- Easy access to expert users
- Technical environment with automated testing, configuration management and frequent integration

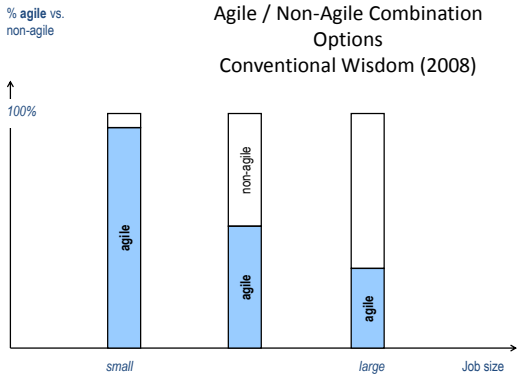
How Detailed to Plan



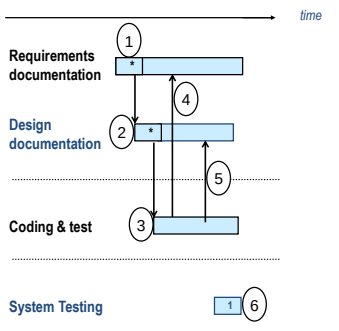
Agile / Non-Agile Tradeoff



Agile / Non-Agile Combination Options Conventional Wisdom (2008)

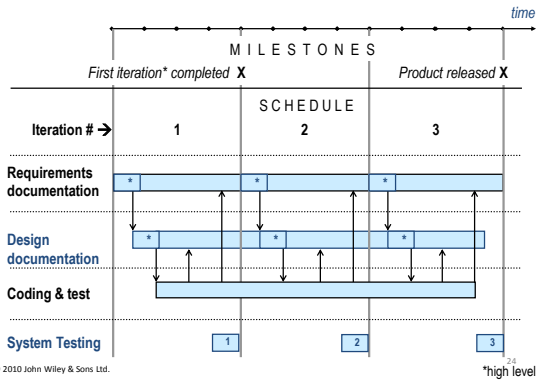


Integrating Agile with non-Agile Methods *1: Time Line*



* High level

Integrating Agile with non-Agile Methods *2*: Iterations



*high level

Agile-Driven Approach to Large Jobs: Each Iteration