SE 217 - Principles of Software Engineering

Lecture 5 – Agile Development Methodologies

From: Dennis, Wixom & Tegarden, *Systems Analysis and Design with UML*, 3rd Ed, 2009, Wiley *and* Roger S. Pressman, *Software Engineering : A Practitioner's Approach*, 6th Ed, 2005, McGraw Hill

Agile Development

- Programming-centric
- They focus on streamlining the SDLC by eliminating much of the modeling and documentation overhead and the time spent on those tasks

From: Dennis, Wixom & Tegarden, *Systems Analysis and Design with UML, 3rd Ed, 2009, Wiley*

Agile Development

- Projects emphasize simple, iterative application development
- Examples are eXtreme Programming, Scrum, and the Dynamic Systems Development Method (DSDM)

From: Dennis, Wixom & Tegarden, *Systems Analysis and Design with UML, 3rd Ed, 2009, Wiley*

The Manifesto for Agile Software Development

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

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- 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."

Kent Beck et al

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Motivation

- In modern economy, it is often difficult or impossible to predict how a computer-based system will evolve

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach,* 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Motivation

- Market conditions change rapidly, end-user needs evolve, and new competition threats emerge

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach,* 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Motivation

- In many situations, we are no longer able to define requirements fully before the project begins
- Software engineers must be agile enough to respond to a fluid business environment

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach,* 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

What is "Agility"?

- Effective (rapid and adaptive) response to change
- Effective communication among all stakeholders
- Drawing the customer onto the team
- Organizing a team so that it is in control of the work performed

Yielding …

- Rapid, incremental delivery of software

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Agile Software Development

- Emphasizes rapid delivery of operational software and de-emphasizes the importance of intermediate work products (not always a good thing)

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach,* 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Agile Software Development

- It adopts the customer as a part of the development team and works to eliminate the "us and them" attitude

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach,* 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Agile Software Development

- It recognizes that planning is an uncertain world has its limits and that a project plan must be flexible

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach,* 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

An Agile Process

- Any agile process is characterized in a manner that addresses three key assumptions about the majority of software projects
  - It is difficult to predict in advance which software requirements will persist and which will change
  - For many types of software, design and construction are interleaved
  - Analysis, design, construction and testing are not as predictable as we might like

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

An Agile Process

- Is driven by customer descriptions of what is required (scenarios)
- Recognizes that plans are short-lived

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach,* 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

An Agile Process

- Develops software iteratively with a heavy emphasis on construction activities
- Delivers multiple 'software increments'
- Adapts as changes occur

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach,* 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Politics of Agile Development

- Two sides
  - "Traditional methodologists are a bunch of stick-in-muds who'd rather produce flawless documentation than a working system that meets business needs"
  - "Lightweight, er, 'agile' methodologists are a bunch of glorified hackers who are going to be in for a heck of surprise when they try to scale up their toys into enterprise-wide software"

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Human Factors

- "People factors" are very important in successful agile development
- Process molds to the needs of the people and team

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach,* 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Human Factors

- The key traits that must exist among the people on an agile team and the team itself
  - Competence – innate talent, specific software related skills, and overall knowledge of the process
  - Common focus – goal is to deliver a working software increment to the customer within the time promised
  - Collaboration – team members collaborate with one another, with the customer, and with business managers
  - Decision-making ability – team should be given autonomy for both technical and project issues
  - Fuzzy problem-solving ability
  - Mutual trust and respect
  - Self-organization – The team servers as its own management

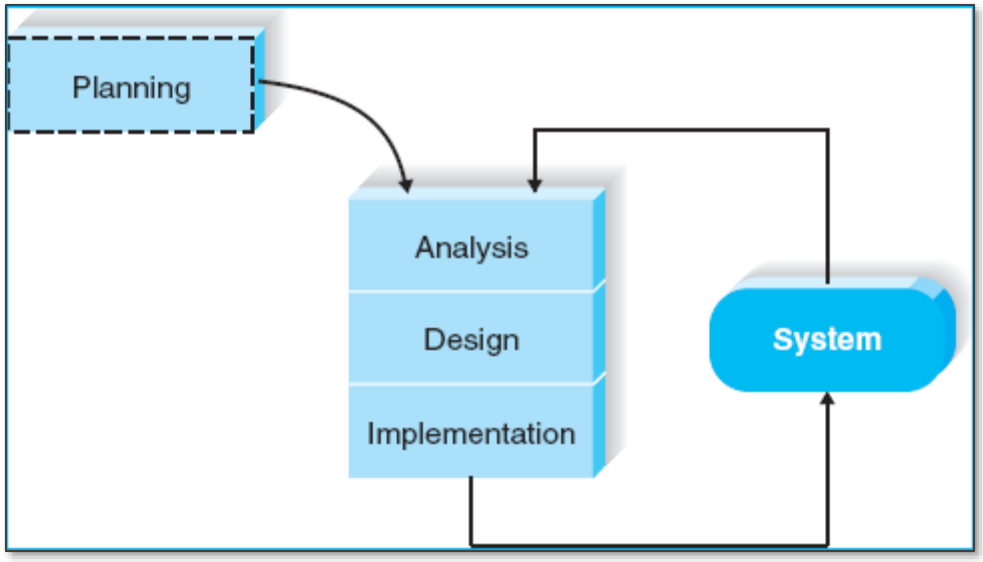
These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Extreme Programming (XP)

- The most widely used agile process, originally proposed by Kent Beck
- Uses an object-oriented approach as its preferred development paradigm

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Extreme Programming (XP)



Extreme Programming

From: Dennis, Wixom & Tegarden, *Systems Analysis and Design with UML, 3rd Ed, 2009, Wiley*

Extreme Programming

- Founded on four core values
  - Communication
    - Developers must provide rapid feedback to end users on a continuous basis
  - Simplicity
    - XP requires developers to follow the KISS principle
  - Feedback
    - Developers must make incremental changes to grow the system
  - Courage
    - Developers must have a quality-first mentality

From: Dennis, Wixom & Tegarden, *Systems Analysis and Design with UML, 3rd Ed, 2009, Wiley*

Extreme Programming

- Key principles it uses
  - Continuous testing
  - Simple coding performed by pairs of developers
  - Close interaction with end users

From: Dennis, Wixom & Tegarden, *Systems Analysis and Design with UML, 3rd Ed, 2009, Wiley*

Extreme Programming

- Testing and efficient coding practices are core to XP
- Relies heavily on refactoring

From: Dennis, Wixom & Tegarden, *Systems Analysis and Design with UML, 3rd Ed, 2009, Wiley*

Extreme Programming

- XP projects begin with user stories that describe what the system needs to do
- Then, programmers code in small, simple modules and test to meet those needs

From: Dennis, Wixom & Tegarden, *Systems Analysis and Design with UML, 3rd Ed, 2009, Wiley*

Extreme Programming

- Users are required to be available to clear up questions and issues as they arise
- Standards are very important, so XP teams use a common set of names, descriptions, and coding practices

From: Dennis, Wixom & Tegarden, *Systems Analysis and Design with UML, 3rd Ed, 2009, Wiley*

Extreme Programming (XP)

- XP Planning
  - Begins with the creation of "user stories"
  - Customer assign a value (priority) to the story
  - Agile team assesses each story and assigns a cost (measured in development weeks)

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

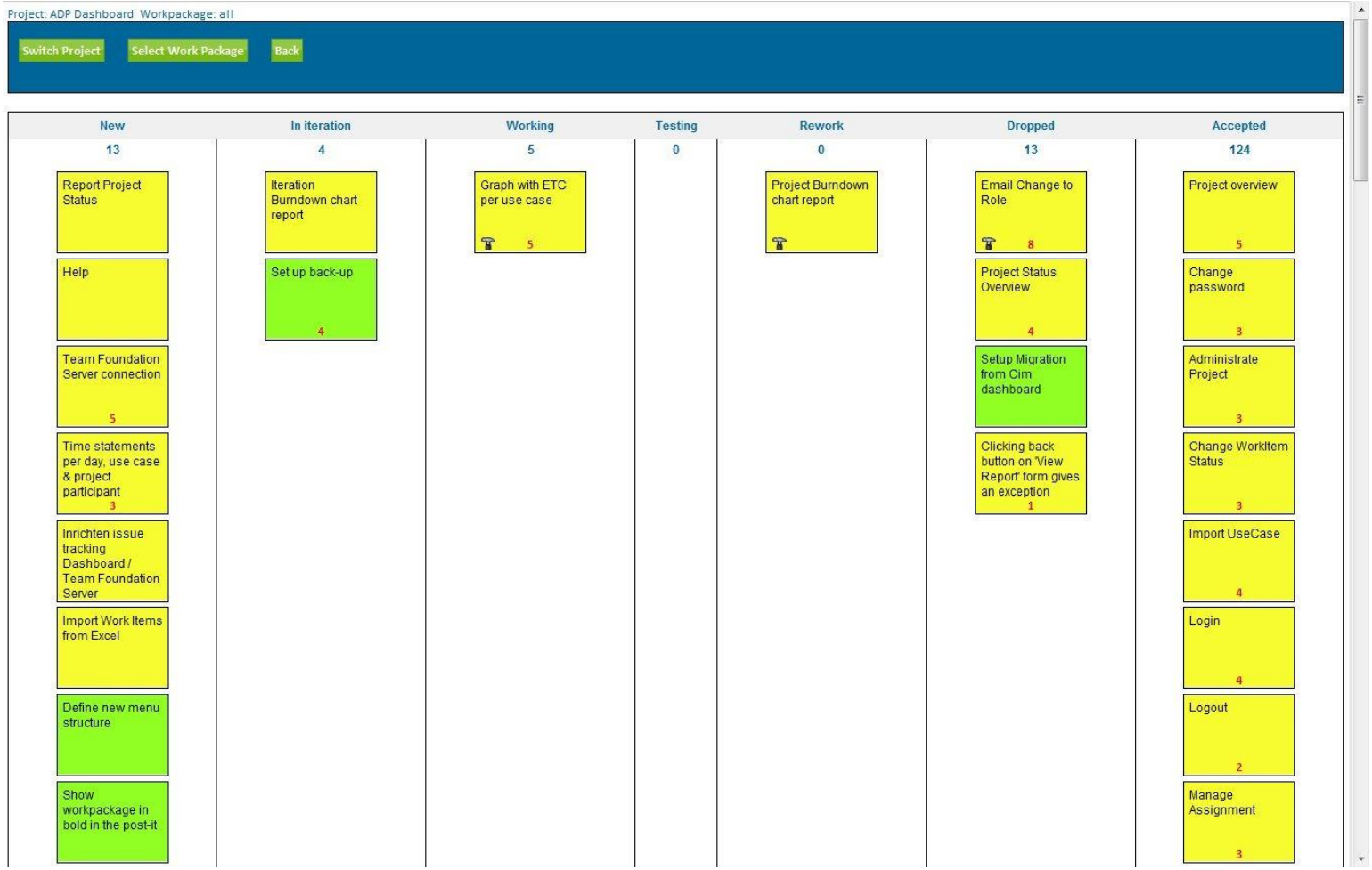
Extreme Programming (XP)

- XP Planning
  - If cost is more than 3 development weeks, the customer is asked to split the story into smaller stories
  - New stories can be written any time

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

An Kanban Agile Dashboard Example

A "Digital" Agile Dashboard Example



http://sanderhoogendoorn.com/blog/index.php/introducing-our-agile-dashboard/

Extreme Programming (XP)

- XP Planning
  - Stories are grouped for a deliverable increment
  - A commitment is made on delivery date
  - XP team orders the stories in one of three ways
    - All stories will be implemented immediately
    - The stories with highest value will be moved up in the schedule
    - The riskiest stories will be moved up in the schedule

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Extreme Programming (XP)

- XP Planning
  - After the first increment "project velocity" (the number of customer stories implemented) is used to help define subsequent delivery dates for other increments

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Extreme Programming (XP)

- XP Design
  - Follows the KIS (keep it simple) principle – the design of extra functionality is discouraged
  - Encourage the use of CRC (class-responsibility-collaborator) cards

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Extreme Programming (XP)

- XP Design
  - For difficult design problems, suggests the creation of "spike solutions"—a design prototype
  - Encourages "refactoring"—an iterative refinement of the internal program design without altering external behavior

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Extreme Programming (XP)

- XP Coding
  - Recommends the construction of a unit test for a story *before* coding commences
  - Encourages "pair programming" – real-time problem solving and quality-assurance

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Extreme Programming (XP)

- XP Testing
  - All unit tests are executed daily
    - "Fixing small problems every few hours takes less time than fixing huge problems just before the deadline"
  - "Acceptance tests" are defined by the customer and executed to assess customer visible functionality

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Extreme Programming

- Advantages
  - Good for small projects
- Disadvantages
  - Not suitable for large projects or teams that are not "jelled"
  - Requires a great deal of discipline; otherwise projects will become unfocused and chaotic
  - Needs a lot of on-site user input, something to which many business units cannot commit

From: Dennis, Wixom & Tegarden, *Systems Analysis and Design with UML, 3rd Ed, 2009, Wiley*

Adaptive Software Development

- Originally proposed by Jim Highsmith
- ASD life cycle incorporates three phases
  - Speculation
  - Collaboration
  - Learning

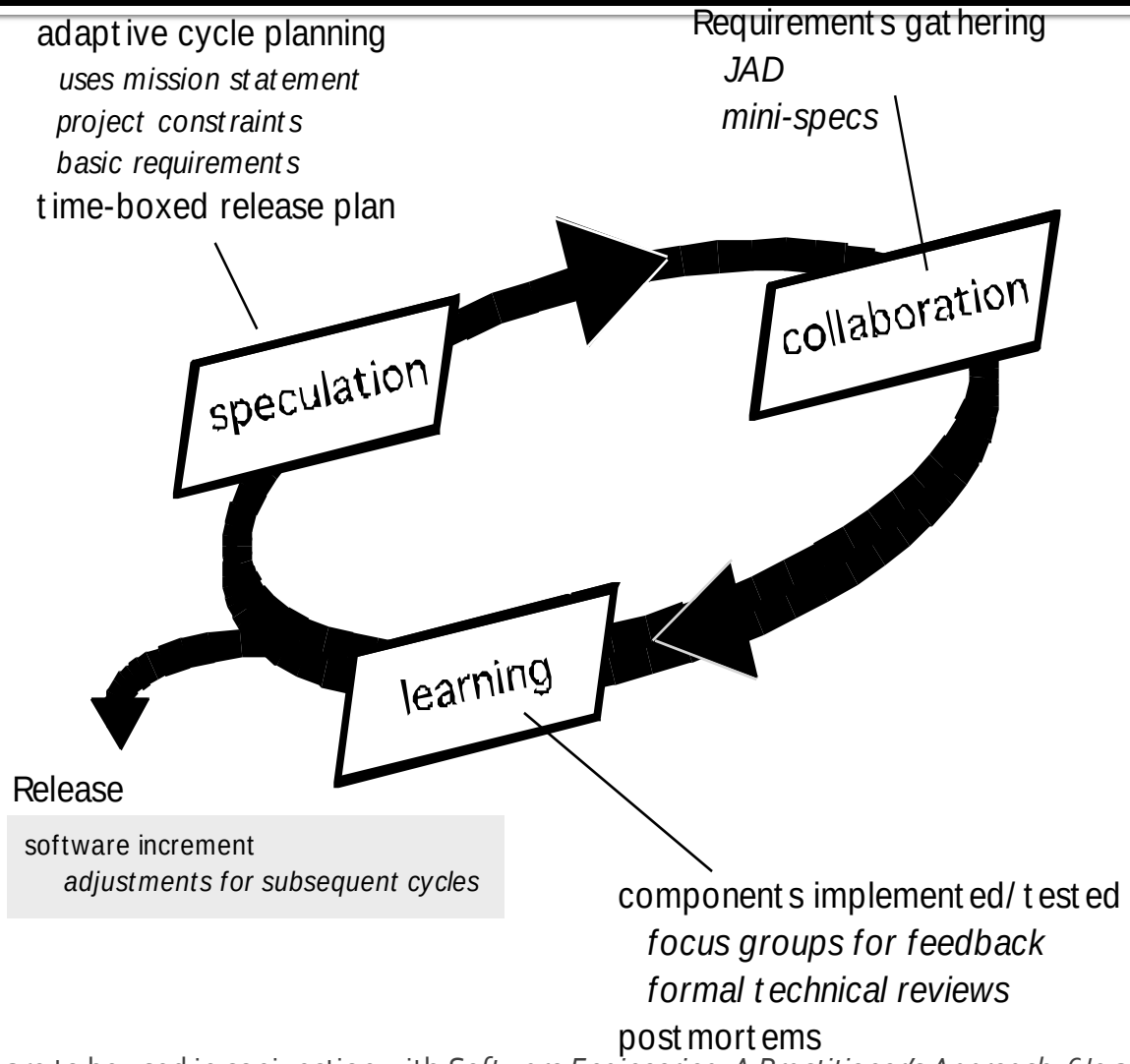
These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Adaptive Software Development

- ASD — distinguishing features
  - Mission-driven planning
  - Component-based focus
  - Uses "time-boxing"
  - Explicit consideration of risks
  - Emphasizes collaboration for requirements gathering
  - Emphasizes "learning" throughout the process

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Adaptive Software Development



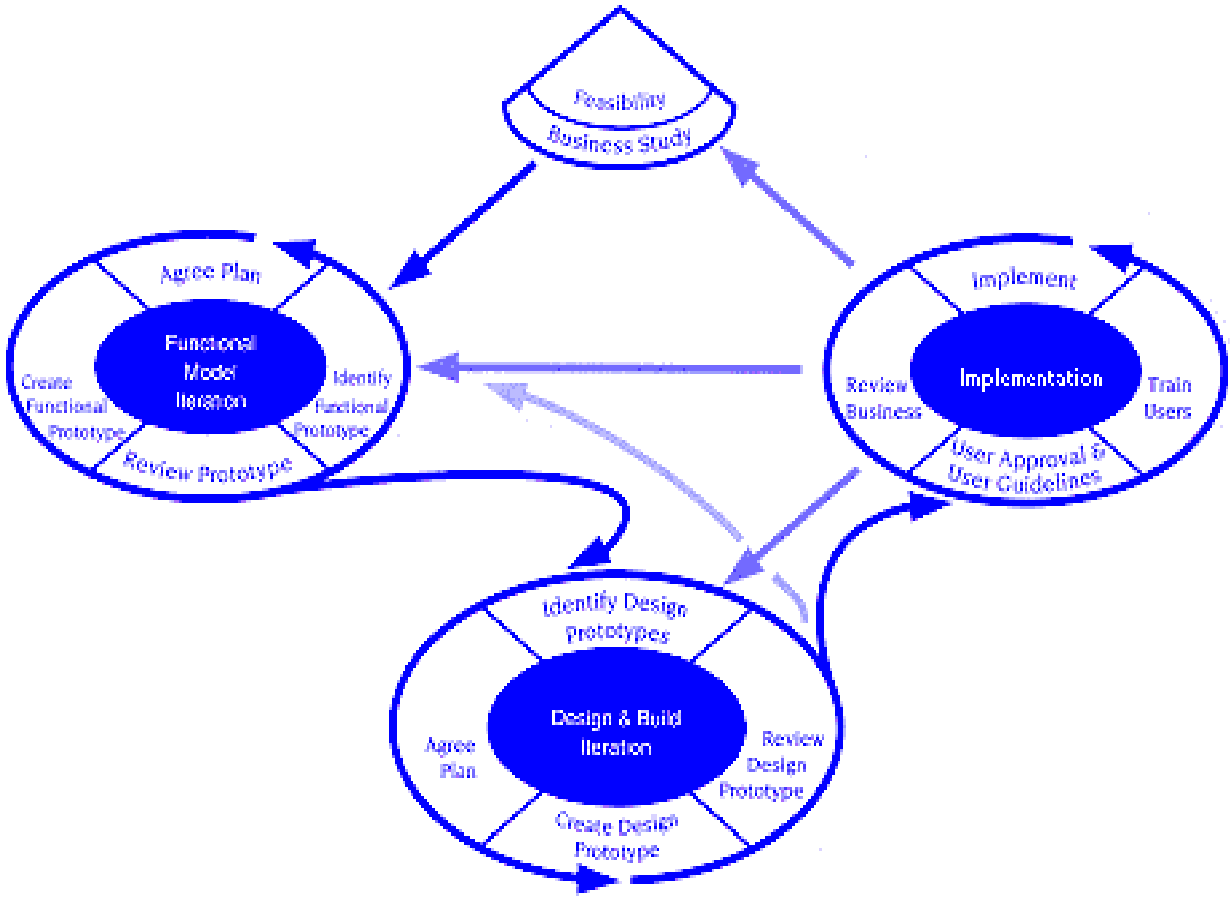
These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Dynamic Systems Development Method

- Promoted by the DSDM Consortium (www.dsdm.org)
- DSDM—distinguishing features
  - Similar in most respects to XP and/or ASD
  - Nine guiding principles
    - Active user involvement is imperative.
    - DSDM teams must be empowered to make decisions.
    - The focus is on frequent delivery of products.
    - Fitness for business purpose is the essential criterion for acceptance of deliverables.
    - Iterative and incremental development is necessary to converge on an accurate business solution.
    - All changes during development are reversible.
    - Requirements are base lined at a high level
    - Testing is integrated throughout the life-cycle.

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Dynamic Systems Development Method



DSDM Life Cycle (with permission of the DSDM consortium)

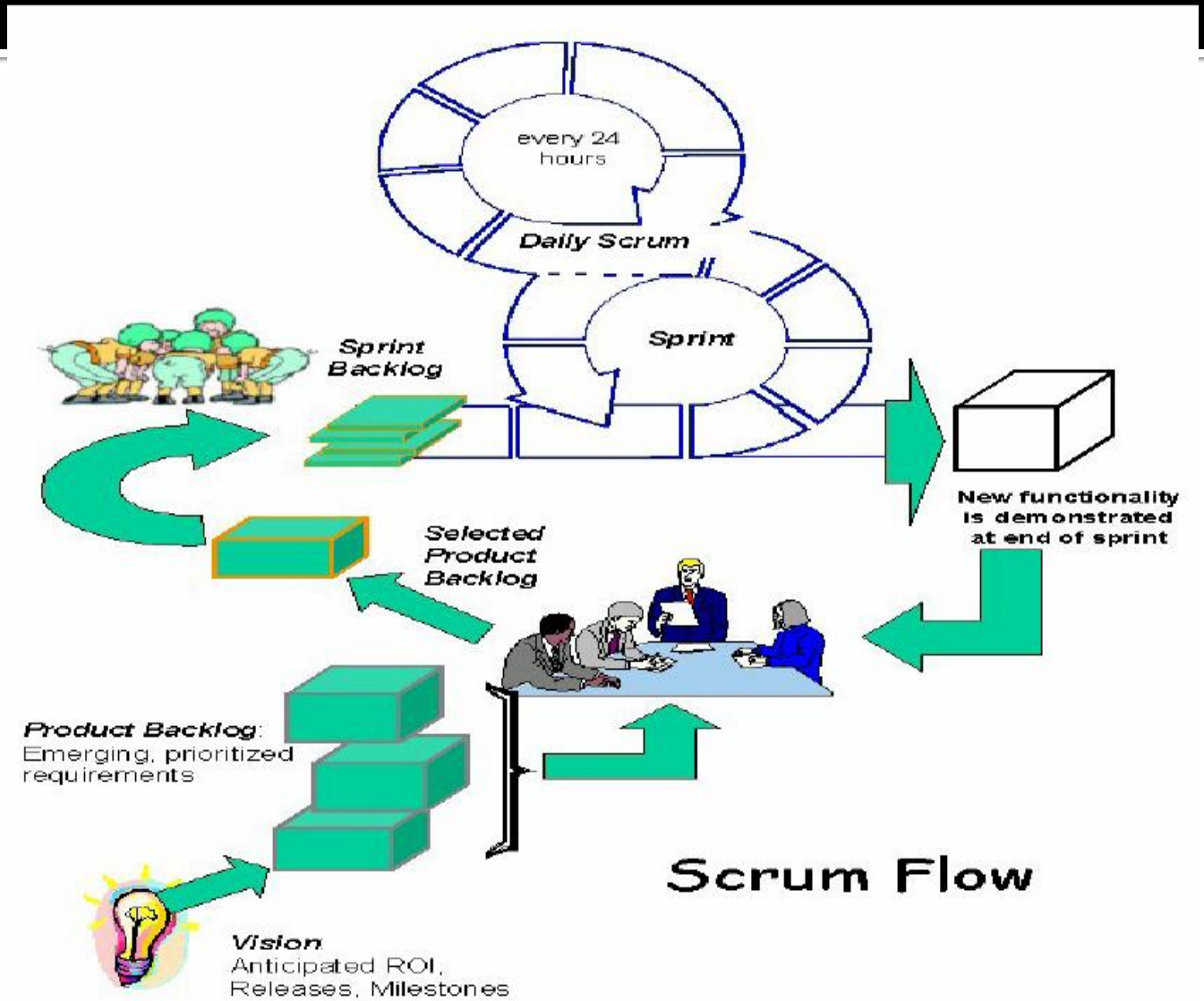
These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Scrum

- Originally proposed by Schwaber and Beedle
- Scrum—distinguishing features
  - Development work is partitioned into "packets"
  - Testing and documentation are on-going as the product is constructed
  - Work occurs in "sprints" and is derived from a "backlog" of existing requirements
  - Meetings are very short and sometimes conducted without chairs
  - "demos" are delivered to the customer with the time-box allocated

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Scrum



These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Scrum

- In rugby, a scrum is used to restart a game
- Creators of the Scrum method believe that no matter how much you plan, as soon as the software begins to be developed, chaos breaks out and plans go out the window

From: Dennis, Wixom & Tegarden, *Systems Analysis and Design with UML, 4th Ed, 2013, Wiley*

Scrum

- The best you can do is to react where the rugby all squirts out
- You then sprint with the ball until the next scrum

From: Dennis, Wixom & Tegarden, *Systems Analysis and Design with UML, 4th Ed, 2013, Wiley*

Scrum

- In the case of the Scrum methodology, a sprint lasts thirty working days
- At the end of the sprint, a system is delivered to the customer

From: Dennis, Wixom & Tegarden, *Systems Analysis and Design with UML, 4th Ed, 2013, Wiley*

Scrum

- Size of a typical Scrum team is no more than seven members
- Teams are self-organized and self-directed
  - No designated team leader

From: Dennis, Wixom & Tegarden, *Systems Analysis and Design with UML, 4th Ed, 2013, Wiley*

Scrum

- Once a sprint has begun, Scrum teams do not consider any additional requirements
- Any new requirement are placed on a backlog of requirements that still need to be addressed

From: Dennis, Wixom & Tegarden, *Systems Analysis and Design with UML, 4th Ed, 2013, Wiley*

Scrum

- At the beginning of every working day, a Scrum meeting takes place
  - All team members stand in a circle and report on what they have accomplished during the previous day, state what they plan to do today and describe anything that blocked progress the previous day

From: Dennis, Wixom & Tegarden, *Systems Analysis and Design with UML, 4th Ed, 2013, Wiley*

Scrum

- At the end of each sprint, the team demonstrates software to the client
- Based on the results of the sprint, a new plan is begun for the next sprint

From: Dennis, Wixom & Tegarden, *Systems Analysis and Design with UML, 4th Ed, 2013, Wiley*

Scrum

- Critisms
  - It is questionable whether Scrum can scale up to develop very large, mission-critical systems
  - How to manage large projects with many teams?

From: Dennis, Wixom & Tegarden, *Systems Analysis and Design with UML, 4th Ed, 2013, Wiley*

Crystal

- Proposed by Cockburn and Highsmith
- Crystal—distinguishing features
  - Actually a family of process models that allow "maneuverability" based on problem characteristics
  - Face-to-face communication is emphasized
  - Suggests the use of "reflection workshops" to review the work habits of the team

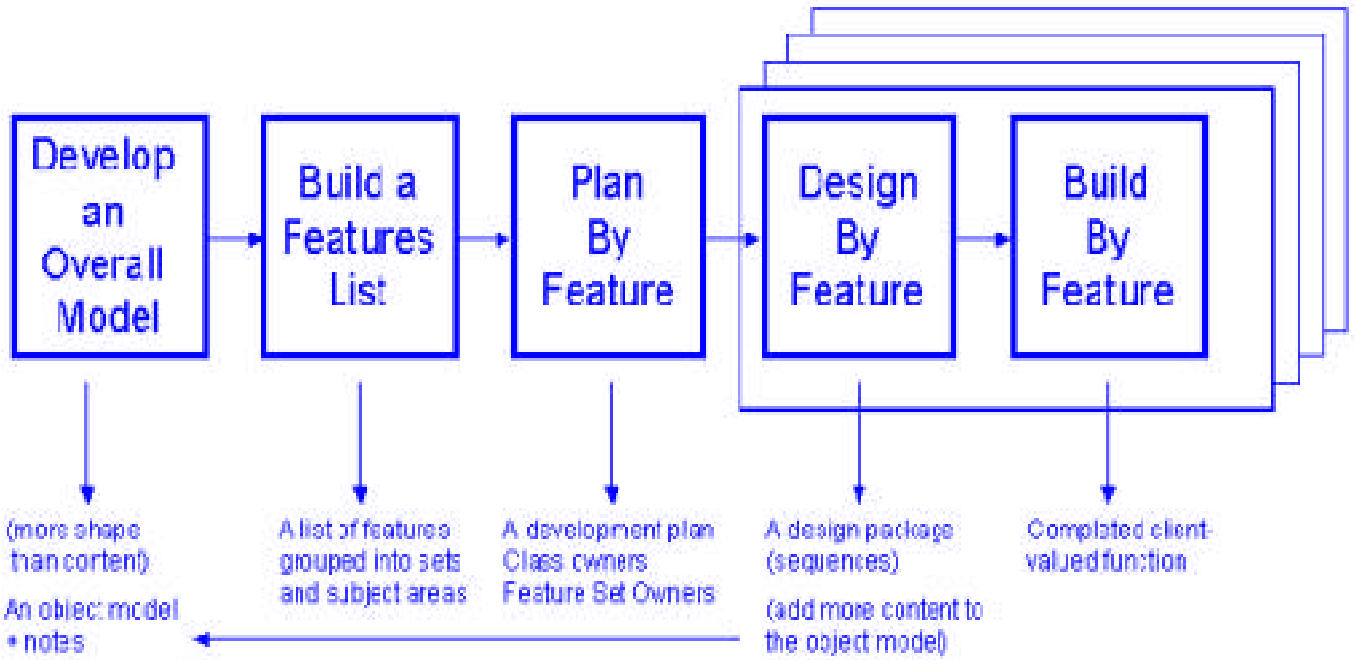
These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Feature Driven Development

- Originally proposed by Peter Coad et al
- FDD—distinguishing features
  - Emphasis is on defining "features"
    - a *feature* "is a client-valued function that can be implemented in two weeks or less."
  - Uses a feature template
    - <action> the <result> <by | for | of | to> a(n) <object>
      - Add the product to a shopping cart
      - Store the shipping-information for a customer
  - A features list is created and "plan by feature" is conducted
  - Design and construction merge in FDD

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Feature Driven Development



Reprinted with permission of Peter Coad

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

Agile Modeling

- Originally proposed by Scott Ambler
- Suggests a set of agile modeling principles
  - Model with a purpose
  - Use multiple models
  - Travel light
  - Content is more important than representation
  - Know the models and the tools you use to create them
  - Adapt locally

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005