Agile Development

- **Agile Processes: Chapter 5**

Slide Set to accompany

Software Engineering: A Practitioner's Approach, 8/e

by Roger S. Pressman and Bruce R. Maxim

Slides copyright © 1996, 2001, 2005, 2009, 2014 by Roger S. Pressman

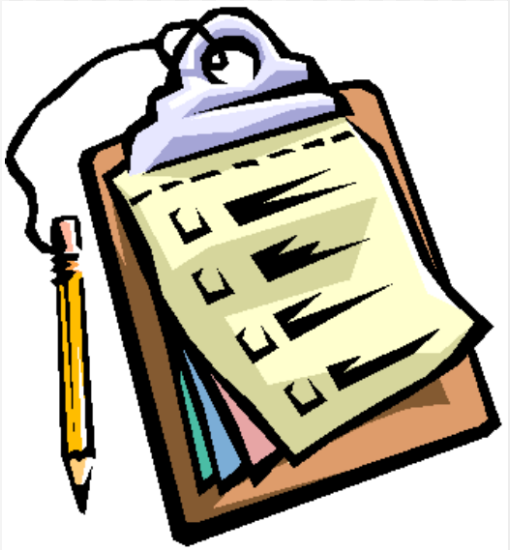
For non-profit educational use only

May be reproduced ONLY for student use at the university level when used in conjunction with *Software Engineering: A Practitioner's Approach, 8/e.* Any other reproduction or use is prohibited without the express written permission of the author.

All copyright information MUST appear if these slides are posted on a website for student use.

Agenda

- Overview of Agile Development
  - Agile Principles
- Agile Processes
  - XP
  - Scrum
  - Other Agile Methodologies



Overview

In 2001, Kent Beck and 16 other noted software developers, writers, and consultants singed an agile manifest

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 slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill, 2014) Slides copyright 2014 by Roger Pressman.

What is “Agility”?

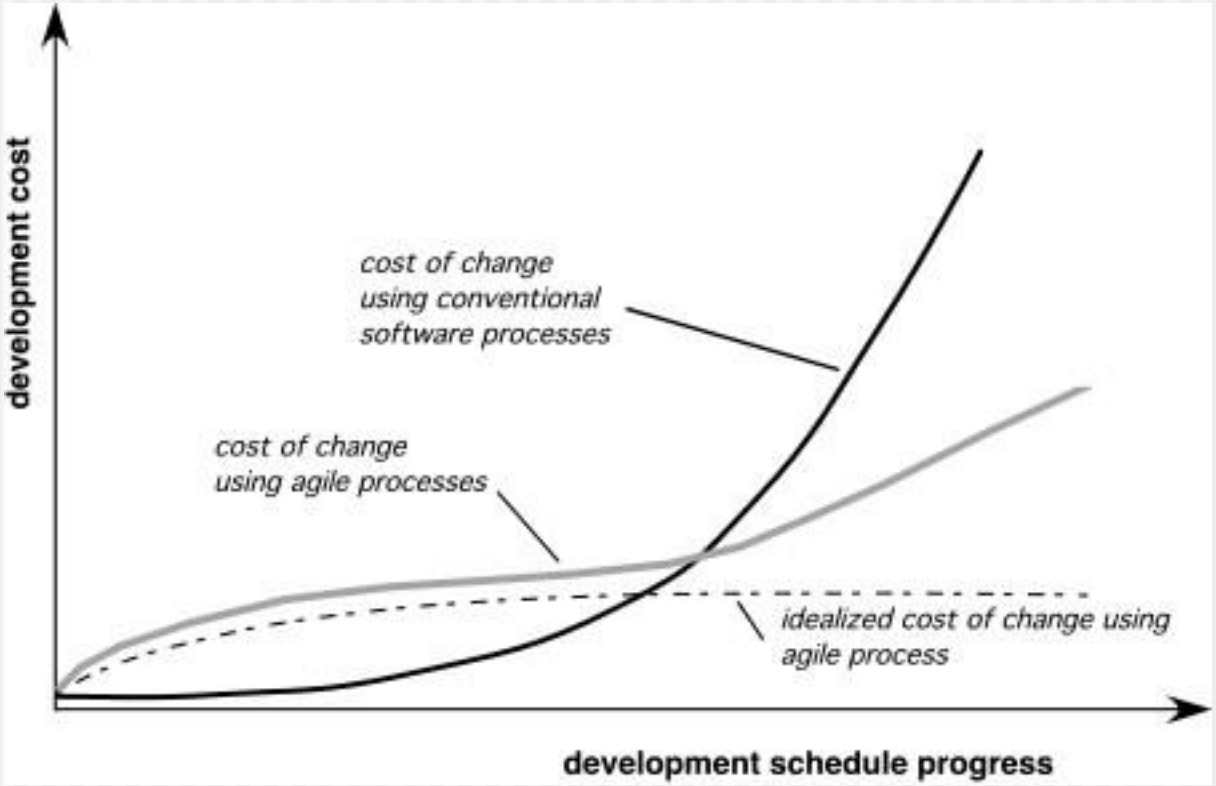
- Effective (rapid and adaptive) response to change
- Effective communication among all stakeholders
- Rapid delivery of operational software
- Drawing the customer onto the team
- Organizing a team so that it is in control of the work performed
  - Flexible planning in an uncertain world

Yielding …

- Rapid, incremental delivery of software

These slides are designed to accompany *Software Engineering: A Practitioner’s Approach, 8/e* (McGraw-Hill, 2014) Slides copyright 2014 by Roger Pressman.

Agility and the Cost of Change



The Applicability of Agile

- Agile development can provide important benefits, but it is not applicable to all projects, all products, all people, and all situations
- In the modern economy, we need agility when:
  - Market conditions change rapidly, end-user needs evolve, and new competitive threats emerge without warning
  - Requirements cannot be fully identified before the project begins (we must be agile enough to respond to a fluid business environment)
- Agile Processes ≈ Light or Lean Processes

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill, 2014) Slides copyright 2014 by Roger Pressman.

Considerations to Accomplish Agility

- Design the process in a way that allows the project team to adapt tasks.
- Conduct flexible planning that understands the fluidity of an agile development approach.
- Eliminate all but the most essential work products and keep them lean.
- Emphasize an incremental delivery strategy that gets working software to the customer as rapidly as feasible.
  - Short-time increments
  - Adaptation keeps pace with change

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill, 2014) Slides copyright 2014 by Roger Pressman.

An Agile Process

- Is driven by customer descriptions of what is required (scenarios)
- Recognizes that plans are short-lived
- Develops software iteratively with a heavy emphasis on construction activities
- Delivers multiple ‘software increments’
- Adapts as changes occur

These slides are designed to accompany *Software Engineering: A Practitioner’s Approach, 8/e* (McGraw-Hill, 2014) Slides copyright 2014 by Roger Pressman.

Agility Principles - I

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

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill, 2014) Slides copyright 2014 by Roger Pressman.

Agility Principles - II

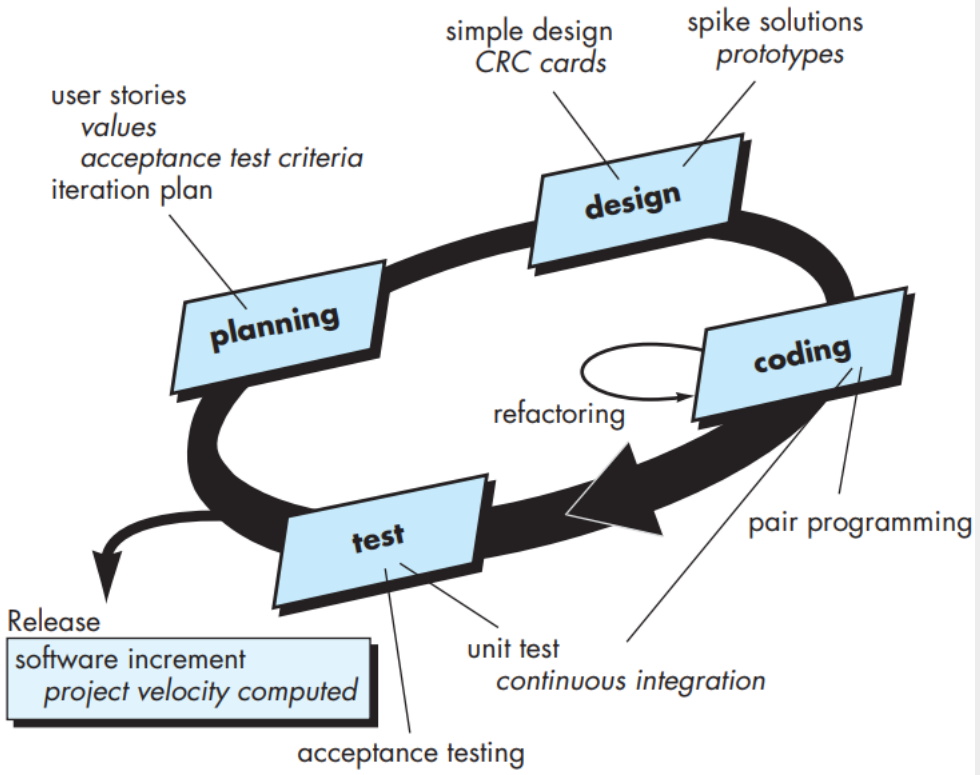
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity – the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

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill, 2014) Slides copyright 2014 by Roger Pressman.

Extreme Programming (XP)

Extreme Programming (XP)

- The most widely used agile process
- Originally proposed by Kent Beck
- XP uses an object-oriented approach

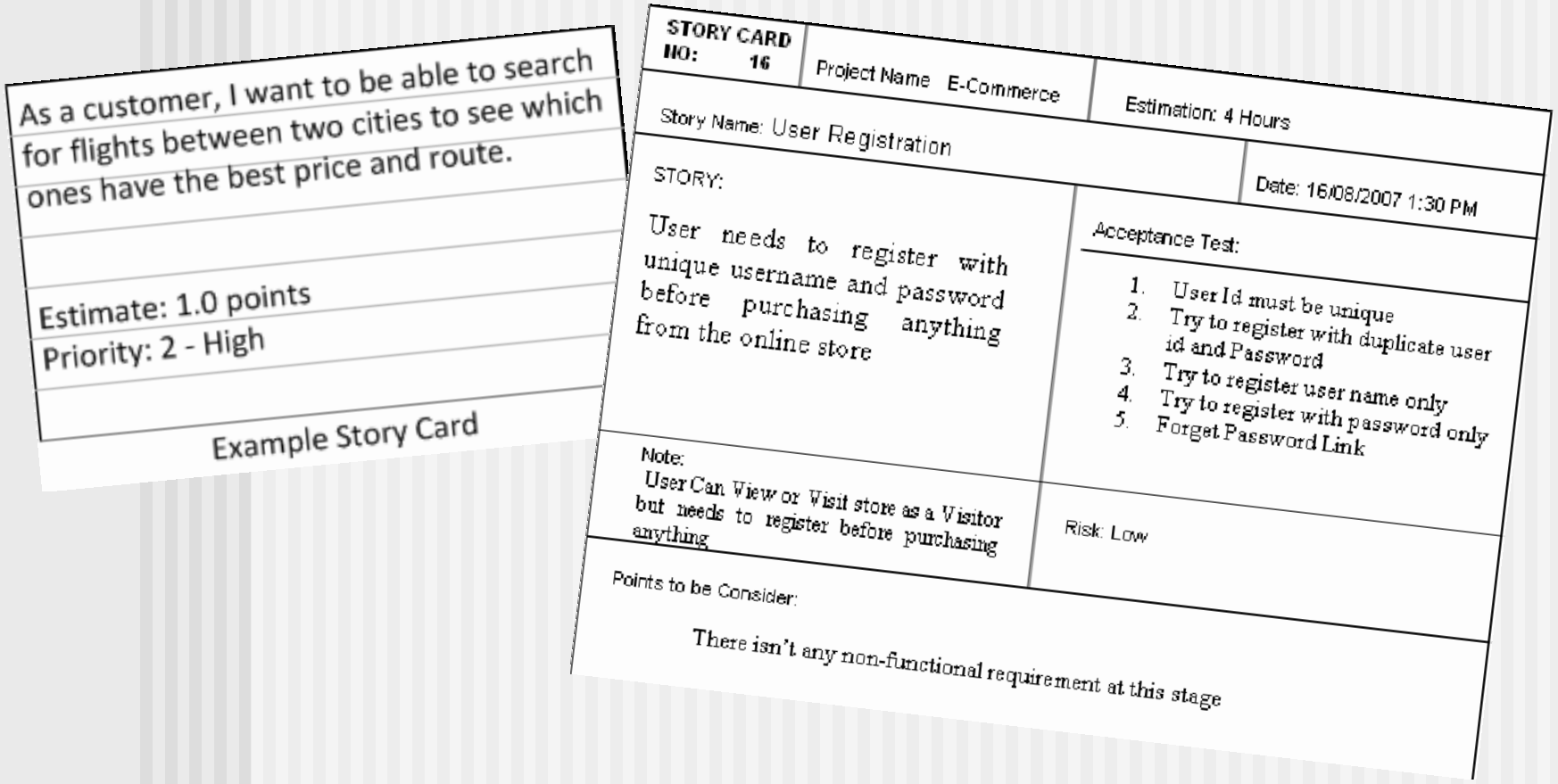


These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill, 2014) Slides copyright 2014 by Roger Pressman.

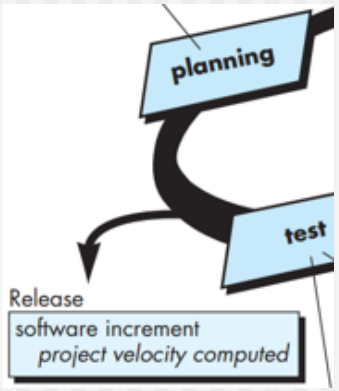
XP Planning

- Begins with the creation of “user stories”
- Each story is written by the customer and is placed on an index card (called story card)
- Customer assigns a value (i.e., a priority) to the story
- Agile team assesses each story and assigns a cost
  - Measured in development weeks
  - Typical story points: 1, 2, 3, or "too big"
  - If the story requires more than three weeks, the customer is asked to split the story into smaller stories
- Stories are grouped for a deliverable increment
- A commitment is made on delivery date

Sample Story Cards

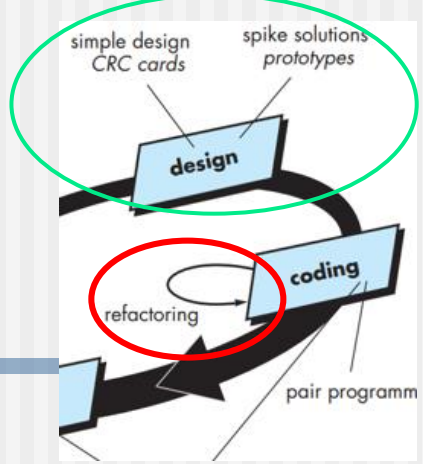


XP Planning (cont.)



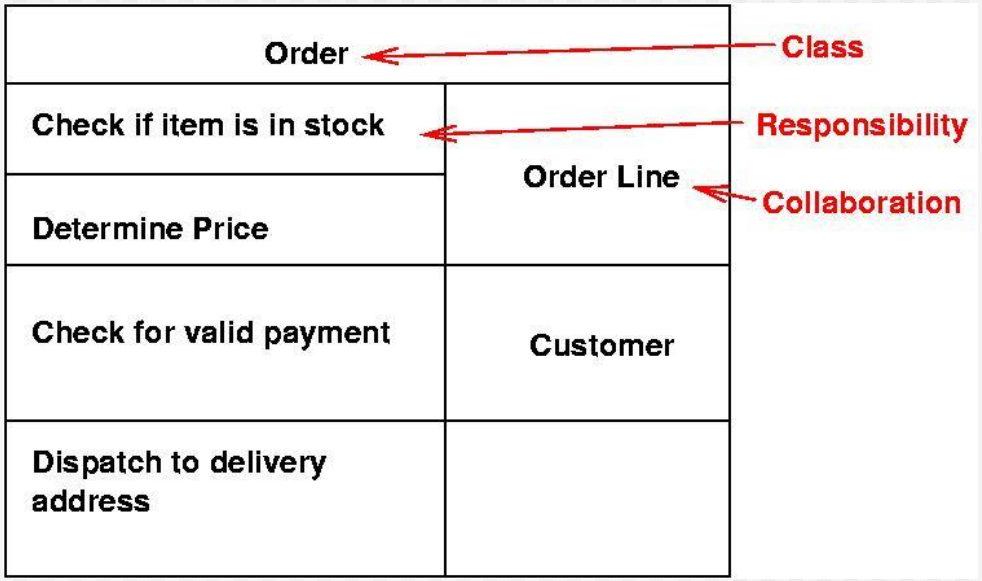
- After each increment, "project velocity" is computed to help define the delivery date of the next increment
  - Project velocity is the number of customer stories implemented during the iteration
- Note: new stories can be written at any time
  - As development work proceeds, the customer can add stories, change the value of an existing story, split stories, or eliminate them
  - The XP team then reconsiders all remaining releases and modifies its plans accordingly

XP Design



- Keep It Simple (KISS principle)
  - The design of extra functionality is discouraged
- Encourages the use of CRC cards (see Chapter 8)
  - Class-Responsibility-Collaborator
- For difficult design problems, suggests the creation of "spike solutions"—a design prototype
- Encourages "refactoring"—an iterative refinement of the internal program design
  - It is a construction technique that is also a design technique
  - Small changes that "can radically improve the design"
- In XP, design occurs both before and after coding commences

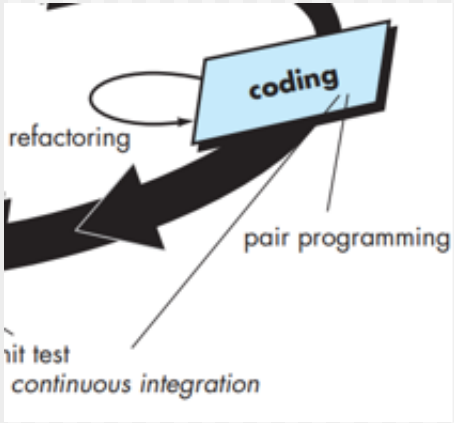
A Sample CRC Card



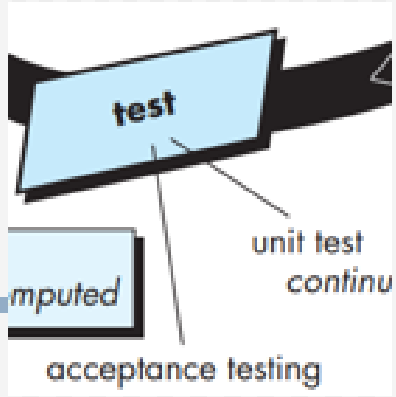
Class Name	
Responsibilities	Collaborators

XP Coding

- Recommends the construction of a unit test for a story ***before*** coding commences
- Encourages "pair programming"
  - Real-time **code review**
- Emphasizes "continuous integration"

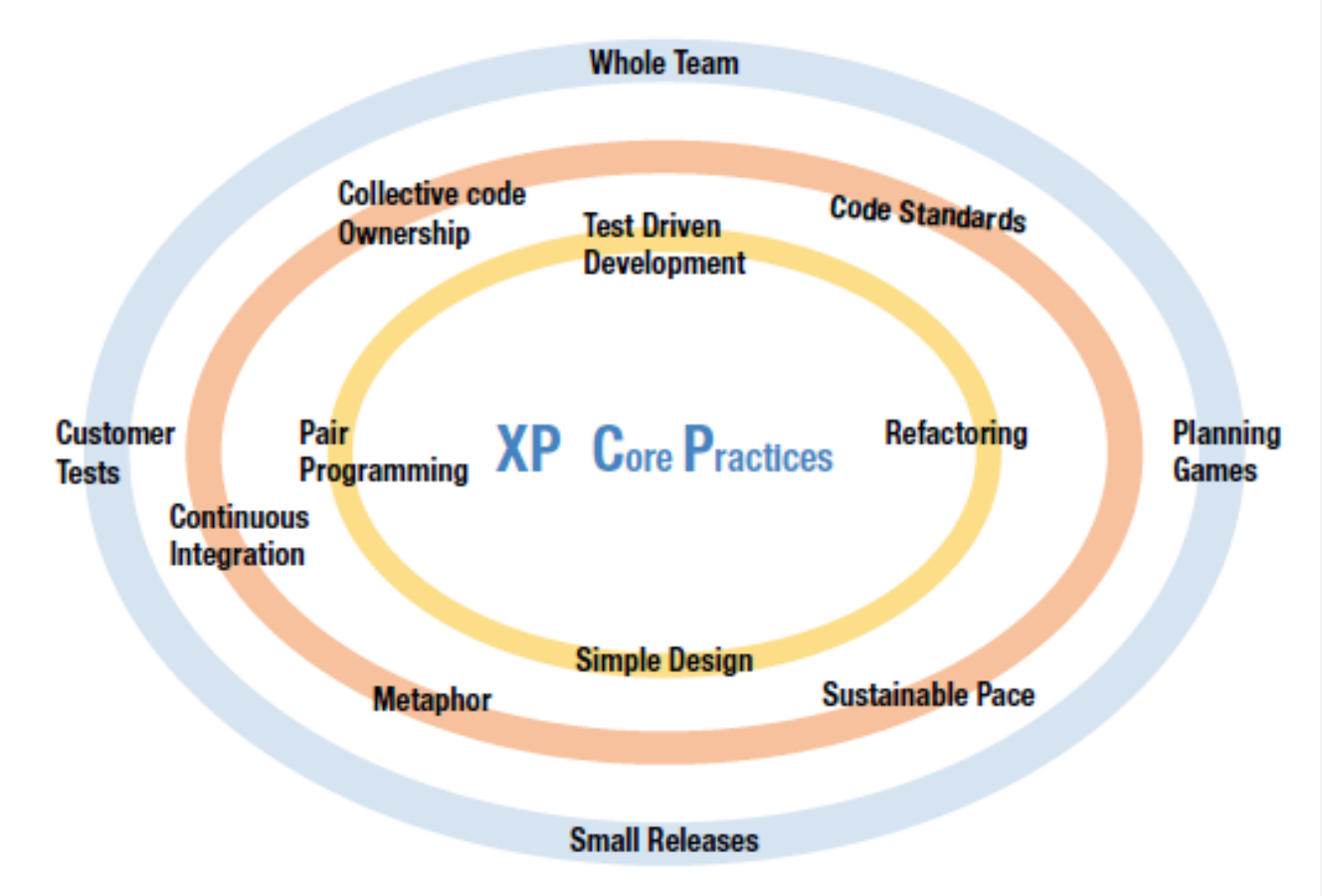


XP Testing



- All unit tests are executed daily
  - Must be **automated**
- Daily testing provides can raise warning flags early
  - "Fixing small problems every few hours takes less time than fixing huge problems just before the deadline"
- **Regression testing**
- "Acceptance tests" are executed to assess customer visible functionality
  - Also called customer tests
  - Derived from user stories

XP Practices



XP Practices: Whole Team

- All contributors to an XP project are **one team**
- Must include a business representative (the ‘Customer’)
  - Provides requirements
  - Sets priorities
  - Guides project
- Team members: programmers, testers, analysts, manager, …

XP Practices: Planning Game

- The planning process within XP:

  The Planning Game
- The game is a meeting that occurs once per iteration
  - Typically once a week
- The planning process is divided into two parts:
  - **Release Planning**
    - What requirements are included in near-term releases?
    - When they should be delivered?
  - **Iteration Planning**
    - Plans the activities and tasks of the developers

XP Practices: Customer Tests

- The Customer defines **automated acceptance tests**
  - for each feature
- Team builds these tests and uses them
  - to prove (to themselves and to the customer) that a feature is implemented correctly
- The best XP teams treat customer tests the same way as programmer tests:
  - Once the test runs, the team ensures that it **keeps running** correctly thereafter
  - System always improves, always notching forward, never backsliding

XP Practices: Small Releases

- Team releases **running**, **tested** software
  - In every iteration
- Releases are **small** and **functional**
- The customer can evaluate and provide **feedback**
- The software is **visible**
  - It is given to the customer at the end of every iteration

XP Practices: Simple Design

- Build software to a **simple** but always **adequate** design
- Start simple and **Keep it that way**
  - Through programmer testing and design improvement
- Good design is essential in XP
- Not a one-time thing
- Teams design and revise design through **refactoring**
  - Throughout the course of the project

XP Practices: Pair Programming

- Software is built by two programmers
  - Sitting side by side, at the same machine
- A better mechanism for **problem solving**
  - Two heads are often better than one
- All production code is reviewed by at least one other programmer
  - **Real-time quality assurance**

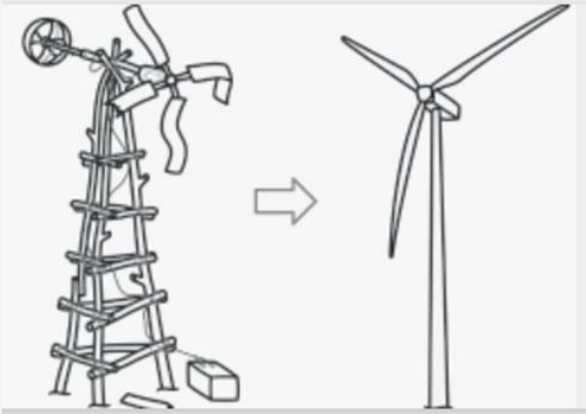
Researches show that pairing produces better code in the same time as programmers working singly

XP Practices: Test-Driven Development

- Writing tests **first**
- Short cycles of adding a test, **and then** making it work
- Nearly 100 percent code coverage
  - Code coverage: The degree to which the source code is tested by the test suite
- The unit tests are all collected together
- Each time a pair releases any code to the **repository** (pairs typically release twice a day or more):
  - Every test must run correctly

XP Practices: Refactoring

- Continuous design improvement process
  - Removal of duplication
  - Increase cohesion
  - Reduce coupling
- Bad Smells and Refactoring techniques
- Supported by comprehensive testing
  - to be sure that as the design improves, nothing is broken
  - customer tests and programmer tests are a critical enabling factor
  - XP practices support each other



Refactoring: clean your code

XP Practices: Continuous Integration

- Keep the system **fully integrated at all times**
- **Daily**, or multiple times a day builds
- Integration is not a phase nor a postponable activity
- Avoid "**integration hell**"
  - where everything broke and no one knew why
- It helps to uncover errors **early**

XP Practices: Collective Code Ownership

- **Any pair** of programmers can improve **any code** at **any time**
- No **‘secure workspaces’**
- All code gets the benefit of many people’s attention
  - increases code quality and reduces defects
- **Avoid duplication**
- Pair with the expert when working on unfamiliar code

XP Practices: Coding Standard

- Follow a common coding standard
- All code in the system must look as if it was written by a single individual
- Code must look familiar, to support collective code ownership
- Coding Conventions
  - Comment conventions
  - Indent style conventions
  - Naming conventions
  - …

XP Practices: Metaphor

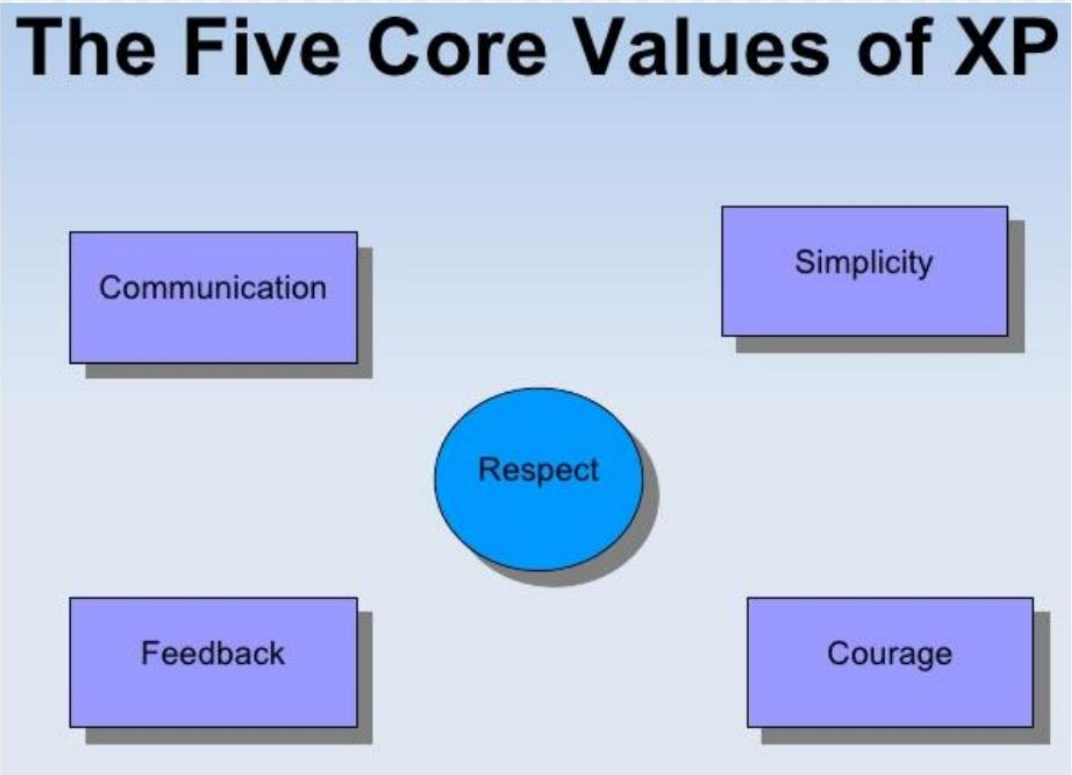
- XP Teams develop a common vision of the system, called "metaphor"
- A simple description of how the program works
- XP teams use a common system of names (i.e., shared vocabulary)
- Ensure everyone understands how the system works
  - where to look for a functionality
  - or where to add a functionality

XP Practices: Sustainable Pace

- Team will produce high quality product when not overly exerted
- Avoid overtime, maintain **40 hour weeks**
- **‘Death march’ projects are unproductive**
  - and do not produce quality software
- Work at a pace that can be **sustained indefinitely**

XP Values

- Communication
- Simplicity
- Feedback
- Courage
- Respect



XP Values: Communication

- ‘We will work together on everything from requirements to code’
- **Poor communication** in software teams:
  - one of the root causes of failure of a project
- Stress on good communication between **all** stakeholders
  - customers, team members, project managers
- Customer representative always on site

Example: Daily Stand Up Meeting

- Purpose: communication among the entire team
- Every morning
- To communicate problems and solutions
- Everyone stands up in a circle to avoid long discussions
- One short meeting
  - Every one is required to attend
  - More efficient than many meetings with a few developers each

XP Values: Simplicity

- ‘Do the **Simplest** Thing That Could Possibly Work’
  - Implement a new capability in the simplest possible way
  - Refactor the system to be the simplest possible code
- ‘You Aren’t Going to Need It’
  - Never implement a feature you don’t need now

XP Values: Feedback

- Always a running system that delivers information about itself in a reliable way
- The system and the code provides feedback on the state of development
- Catalyst for change and an indicator of progress

XP Values: Courage

- We will tell the truth about progress and estimates
- We don't fear anything because **no one ever works alone**
- We will adapt to changes when ever they happen

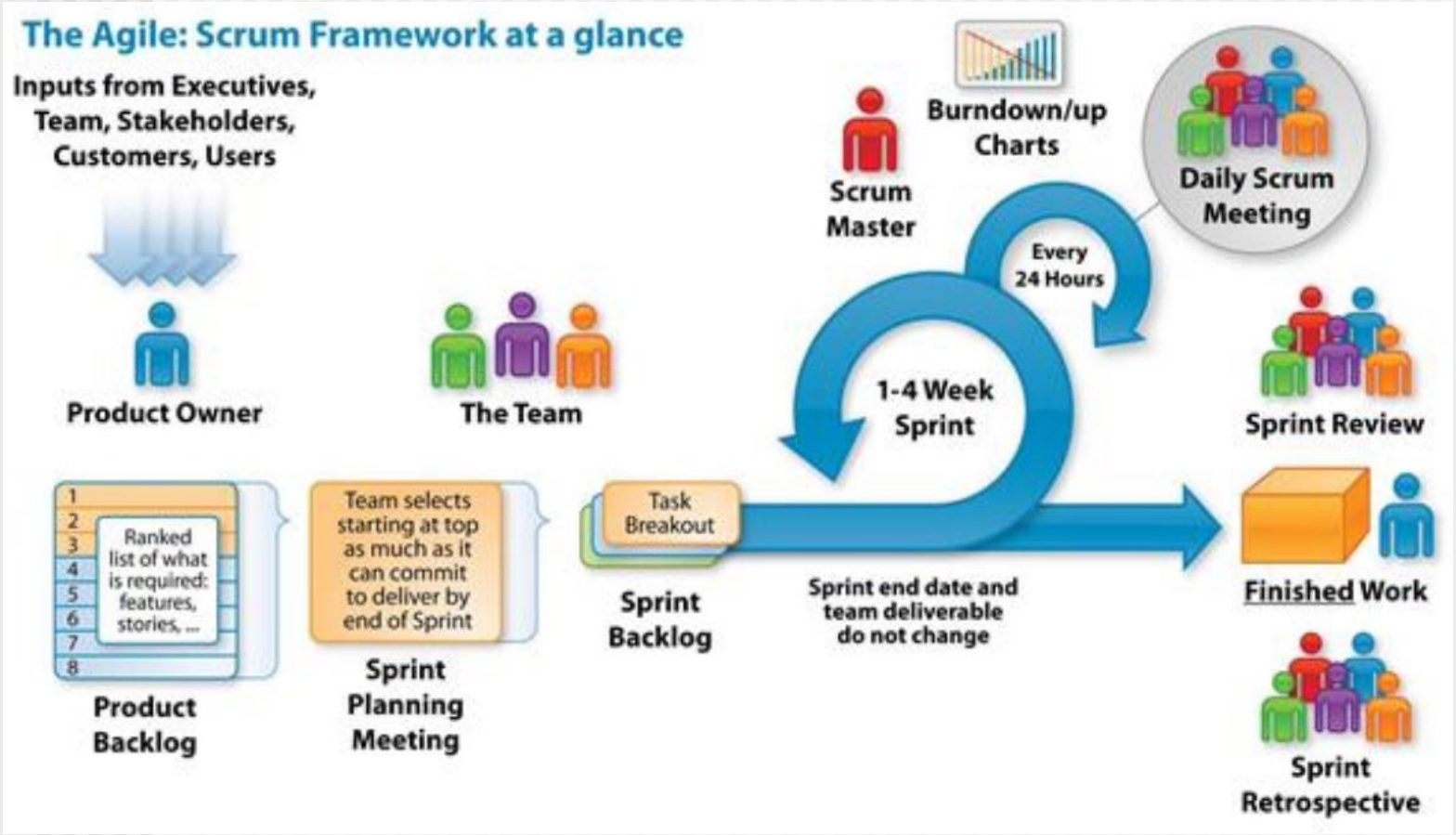
SCRUM

Scrum

- Originally proposed by Schwaber and Beedle
- An iterative and incremental agile methodology
  - For managing product development
- It is agile, so it appreciates:
  - Accept change, Good Communication, Customer Collaboration, …
- Scrum defines specific roles, events, and artifacts

The name is derived from rugby (an activity that occurs during a rugby game) to stress the importance of working as a team in complex product development

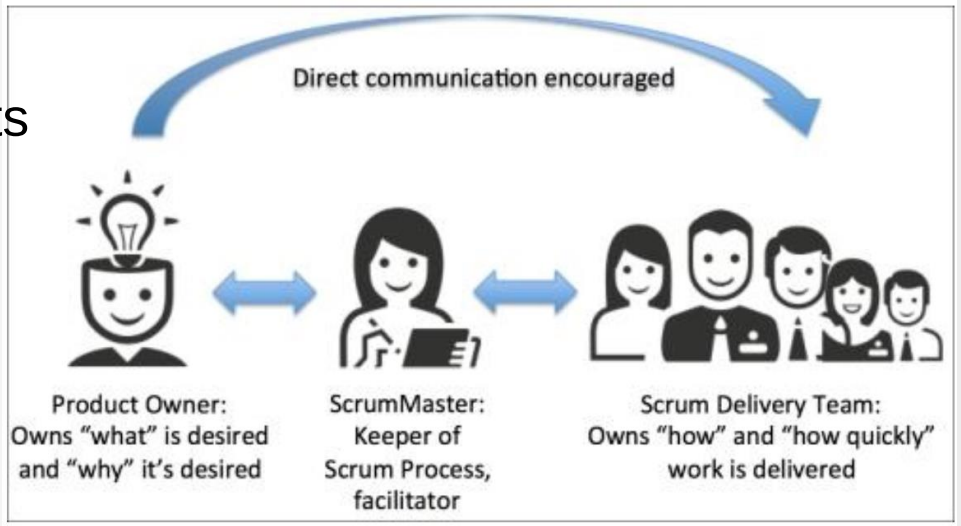
Scrum Framework at a glance



These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill, 2014) Slides copyright 2014 by Roger Pressman.

Scrum Roles

- **Product owner**
  - Represents the stakeholders
  - and is the voice of customer
- **Scrum master**
  - Ensures that the scrum process is used as intended
  - Coaching the team
- **Development team**
  - Responsible for delivering increments



Sprint: an act of running at full speed

Scrum Sprints

- A **sprint** (or iteration)
  - The basic unit of development in scrum
- The sprint is a time-boxed effort (typically two weeks)
- Scrum emphasizes working product at the end of the sprint
  - That is really *done*.
  - E.g. the software has been **integrated**, fully **tested**, **end-user documented**, and is **potentially shippable**
- The number of sprints depends on product complexity and size

Scrum Events

1. **Sprint planning**
   - At the beginning of a sprint
2. **Daily scrum meeting**
   - All members of the development team come prepared
   - Starts precisely **on time** (even if some members are missing)
   - Should happen at the same time and place every day
   - Is limited to **15 minutes**
- At the end of a sprint:
   3. **Sprint review**
   4. **Sprint retrospective**

Daily Scrum Meeting

- Every day
- Very short (typically 15-minute)
- Three key questions are asked and answered by all team members:

1. What did you do since the last team meeting?
2. What obstacles are you encountering?
3. What do you plan to accomplish by the next team meeting?

- Scrum master leads the meeting and assesses the responses from each person

Daily Scrum: Benefits

- Helps the team to uncover potential problems as early as possible
- Lead to “knowledge socialization”
  - promote a self-organizing team structure

At the End of a Sprint:

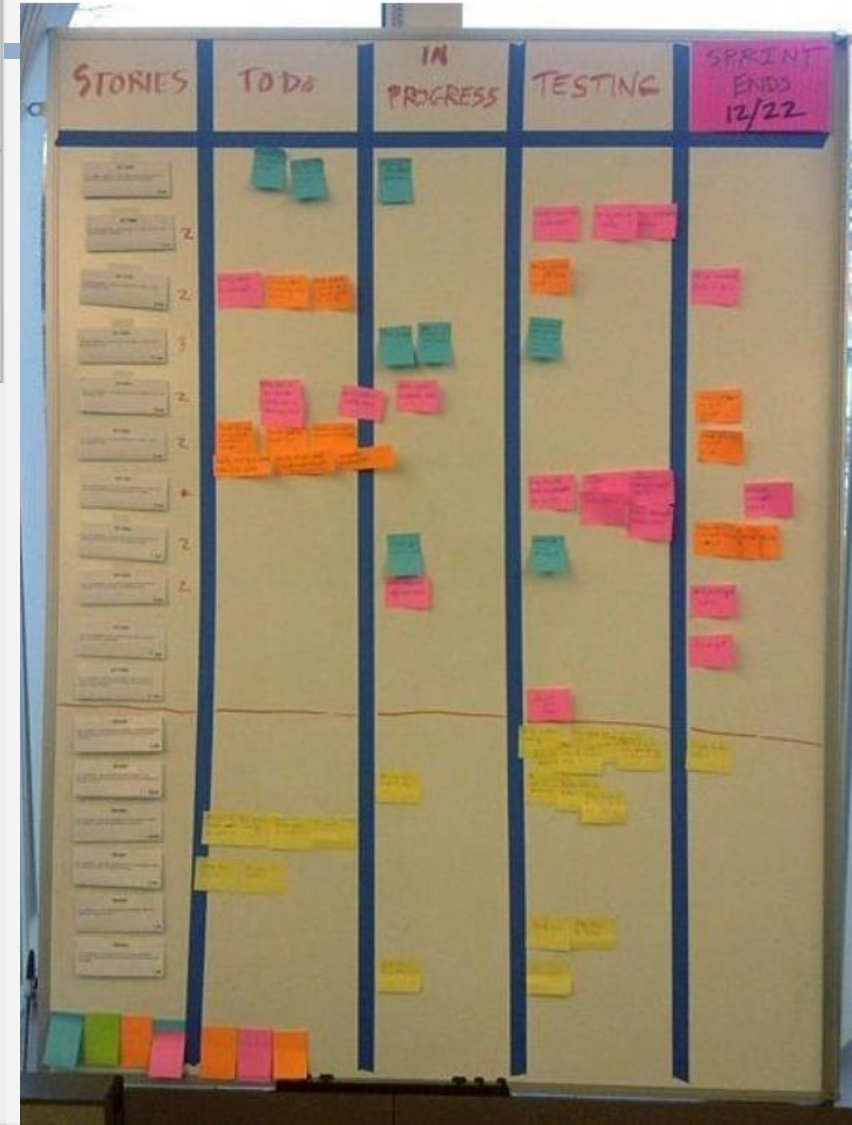
- **Sprint review**
  - Reviews the work that was completed
  - Presents the completed work to the stakeholders (**demo**)
    - Incomplete work cannot be demonstrated
  - The recommended duration: 2 hours for a two-week sprint
- **Sprint retrospective**
  - What went well during the sprint?
  - What could be improved in the next sprint?
  - The recommended duration: 1.5 hours for a two-week sprint
  - This event is facilitated by the **scrum master**

Scrum Artifacts

- **Product backlog**
  - An **ordered** list of product requirements
  - Includes: Features, bug fixes, non-functional requirements, etc.
  - The **product owner** orders the product backlog items
- **Sprint backlog** (*the output of sprint planning*)
  - The works the development team must address during a sprint
  - By selecting product backlog items **from the top** of the product backlog
  - Broken down into tasks by the development team
  - Often a **task board** is used: the state of the tasks of the current sprint
    - Like: "*to do*", "*in progress*", and "*done*"

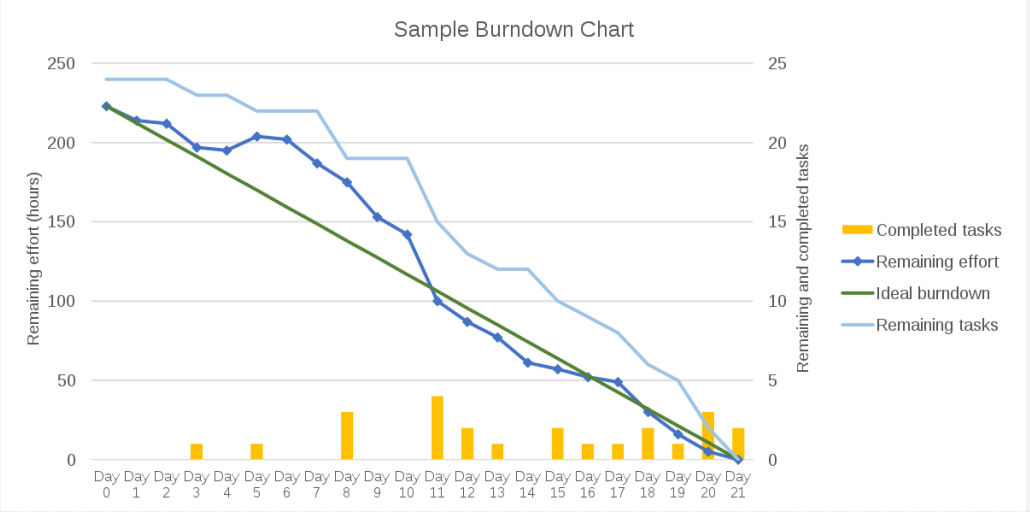
Stories	To Do	In Progress	Testing	Done
This is a sample text. Replace it with your own text.	This is a sample text. Replace it with your own text. This is a sample text. Replace it with your own text. This is a sample text. Replace it with your own text. This is a sample text. Replace it with your own text.	This is a sample text. This is a sample text. This is a sample text.	This is a sample text. This is a sample text. This is a sample text.	This is a sample text. Replace it with your own text. This is a sample text. Replace it with your own text.
This is a sample text. Replace it with your own text.	This is a sample text. This is a sample text. This is a sample text. This is a sample text.	This is a sample text. This is a sample text. Replace it with your own.	This is a sample text. This is a sample text.	This is a sample text. Replace it with your own text.

Scrum Board



Scrum Artifacts (cont.)

- **Product increment**
  - potentially shippable increment (PSI)
- **Sprint burn-down chart**
  - A **public** displayed chart
  - Shows remaining work in the sprint backlog
  - Updated every day:
  - gives a simple view of the sprint progress



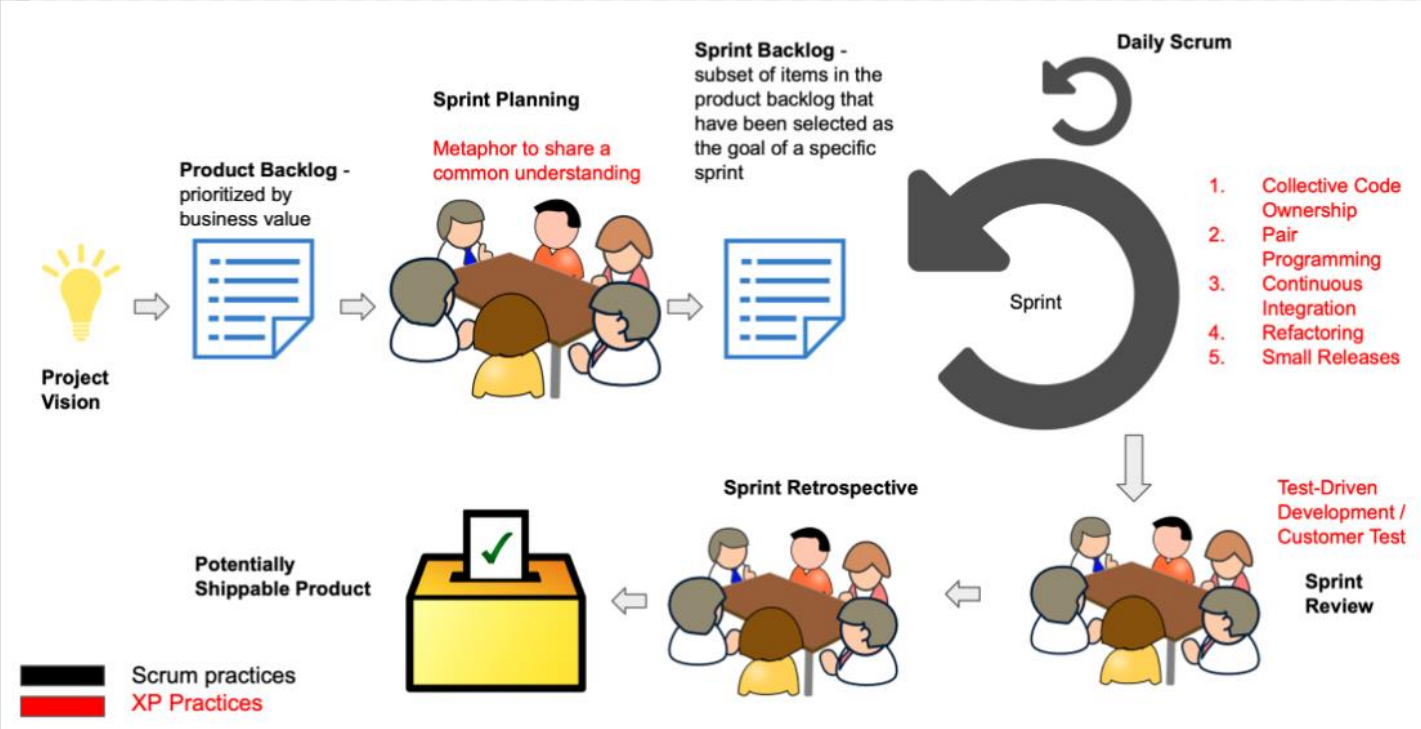
Scrum vs XP?

- Scrum doesn’t prescribe any engineering practices; XP does.
  - Scrum is an agile **management methodology**
  - XP is an agile **engineering methodology**
  - As such they are entirely **complementary**
- Minor differences:
  - XP iterations: 1-2 weeks, Scrum sprints: 1-4 weeks (usually longer)
  - Scrum doesn’t allow changes to the sprint backlog during the sprint

Scrum/XP Hybrid

- Many teams use both Scrum and XP
  - Scrum template for project management
  - XP practices for technical tasks

(XP and Scrum complement each other)



Other Agile Methodologies

Other Agile Methodologies

- Agile Unified Process (AUP)
- Dynamic Systems Development Method (DSDM)
- Agile Modeling (AM)
- Feature Driven Development (FDD)
- Adaptive Software Development (ASD)
- Crystal
- ….

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill, 2014) Slides copyright 2014 by Roger Pressman.

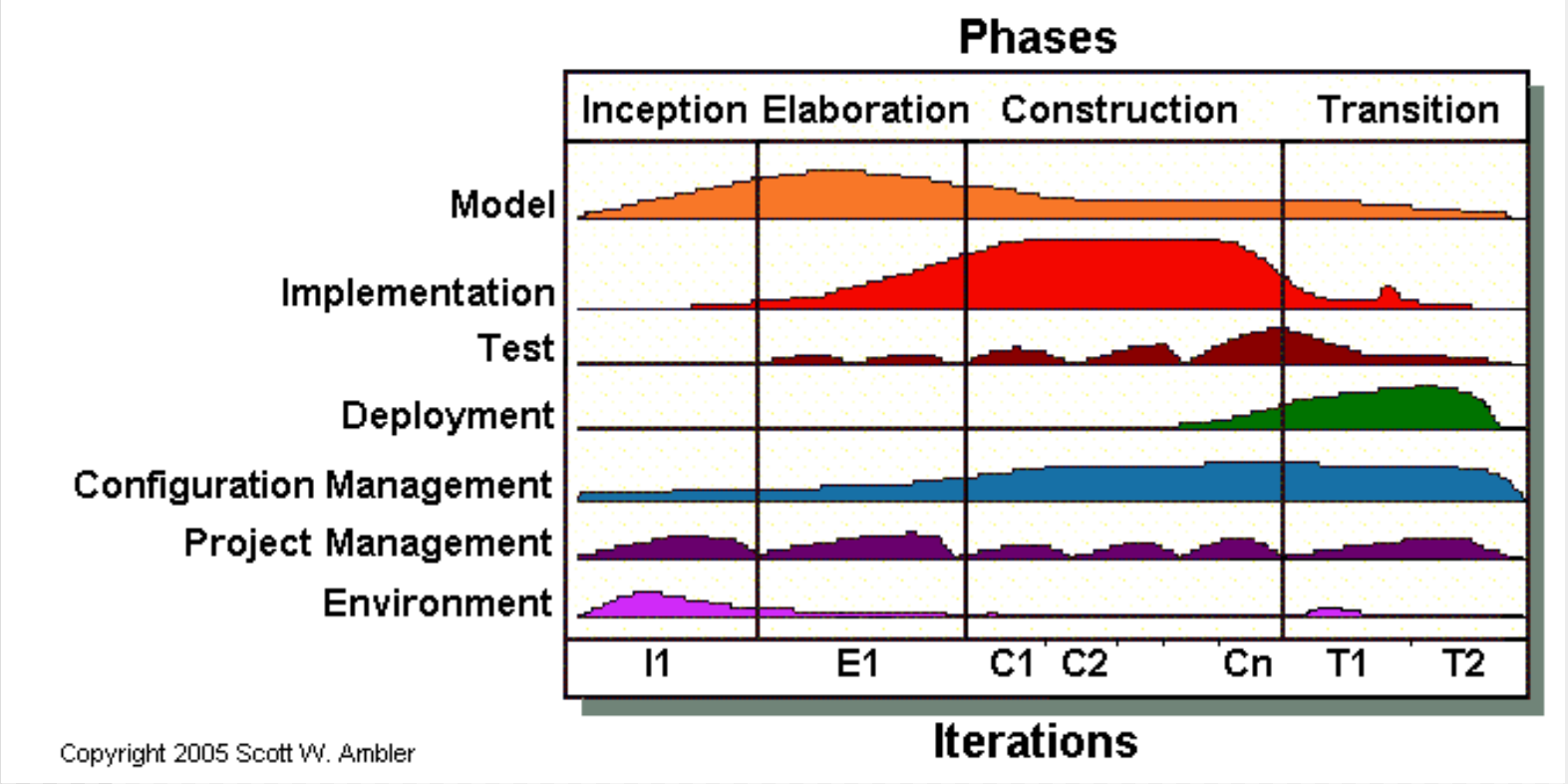
Agile Unified Process

- AUP adopts a "serial in the large" and "iterative in the small" philosophy
- Serial in the Large
  - Four phases:

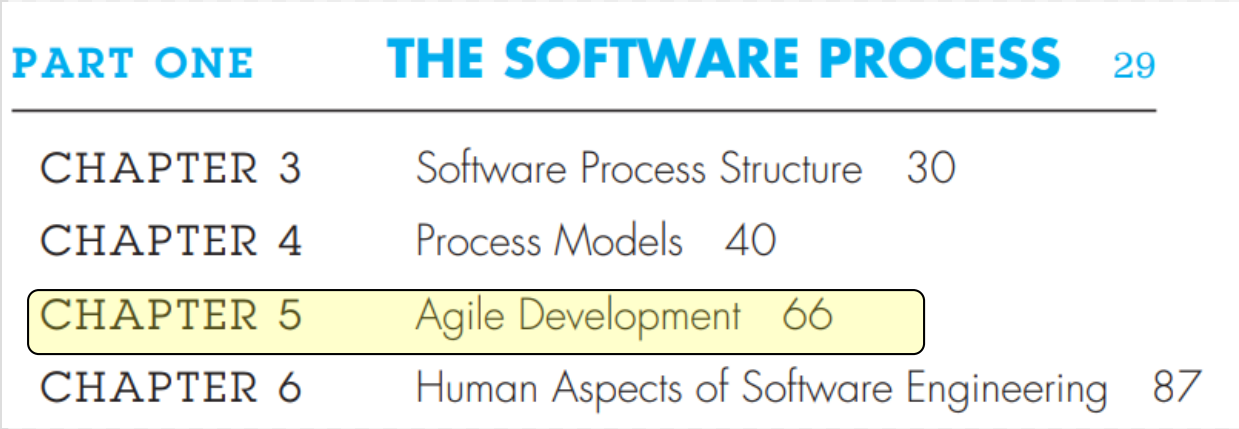
    Inception, Elaboration, Construction, Transition
- Iterative in the small
  - Disciplines are performed in an iterative manner
  - Seven disciplines:
    - Modeling, Implementation, Testing, Deployment, Configuration Management, Project Management, Environment

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill, 2014) Slides copyright 2014 by Roger Pressman.

Agile Unified Process



Further Reading



PART ONE THE SOFTWARE PROCESS 29

- Search about agile processes
  - and read some tutorials/wikis/ppts/…
  - About:
    - XP
    - SCRUM
    - …

The End