Chapter 3

An Agile View of Process

Software Engineering: A Practitioner’s Approach

6th Edition

Roger S. Pressman

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**

What is Agility?

- Effective response to change
- Effective communication among all stakeholders
- Drawing the customer onto the team; eliminate the “us and them” attitude
- Organizing a team so that it is in control of the work performed
- Rapid, incremental delivery of software

Principles to achieve agility – by the Agile Alliance (1)

1. Highest priority -> satisfy the customer
2. Welcome changing requirements
3. Deliver working software frequently
4. Business people and developers must work together
5. Build projects around motivated individuals
6. Emphasize face-to-face conversation

Principles to achieve agility – by the Agile Alliance (2)

7. Working software is the primary measure of progress
8. Agile processes promote sustainable development
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 designs emerge from self-organizing teams
12. The team tunes and adjusts its behavior to become more effective

Agile Software Process – Three Key Assumptions

- Difficulty in predicting changes of requirements and customer priorities
- For many types of s/w, design and construction are interleaved
- Analysis, design, construction, and testing are not as predictable as we might like

Agile Software Process

- An agile process must be adaptable
- It must adapt incrementally
- Requires customer feedback
- An effective catalyst for customer feedback is an operational prototype

The Politics of Agile Development

- There is considerable debate about the benefits and applicability of agile software development
- No one is against agility. The real question is:
  - What is the best way to achieve it?

Human Factor

- Key point:
  - The Process molds to the needs of the people and team, not the other way around
- A number of key traits must exist among the people on an agile team and the team itself
  - Competence
  - Common focus
  - Collaboration
  - Decision-making ability
  - Fuzzy problem-solving ability
  - Mutual trust and respect
  - Self-organization

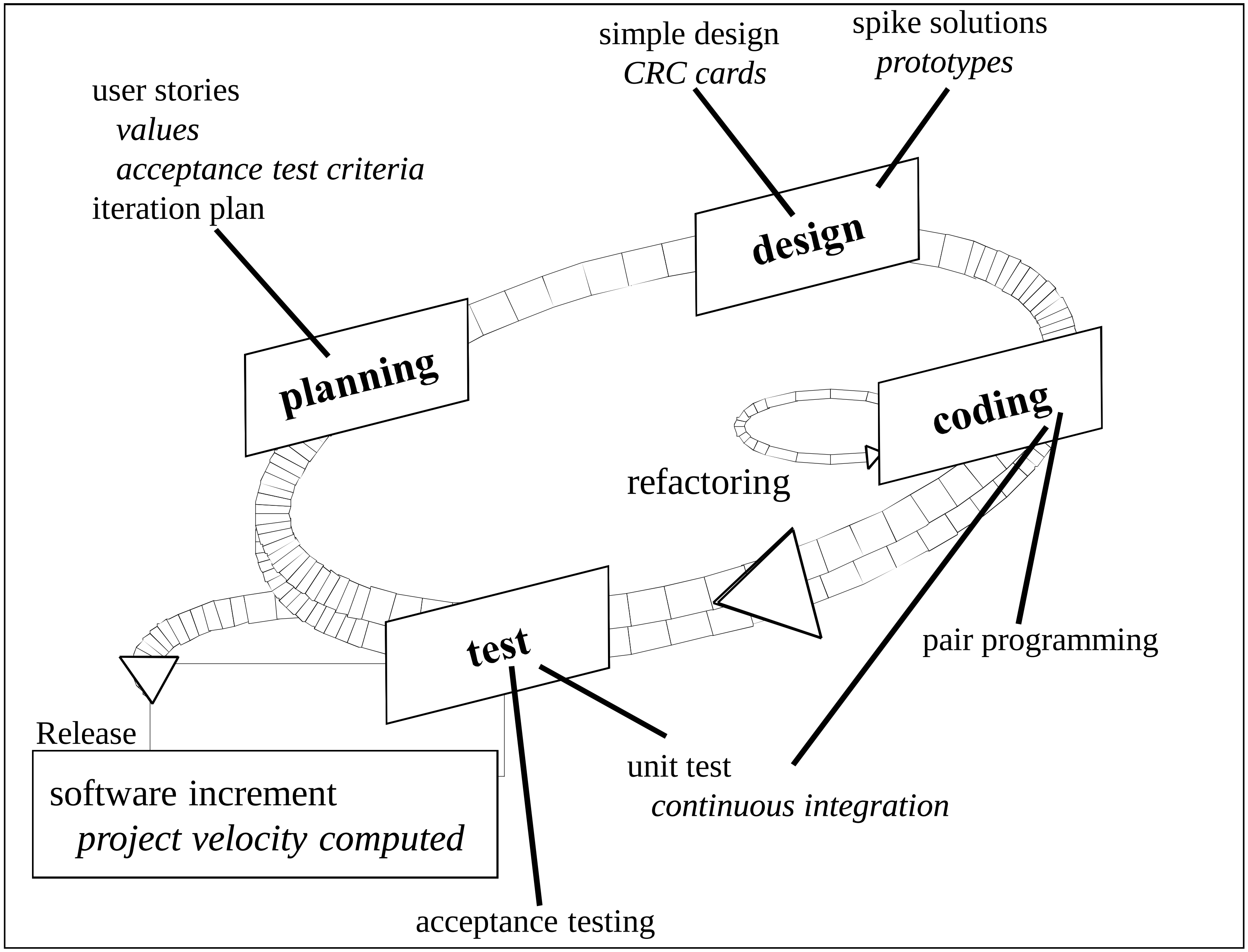
Agile Process Models

- Extreme Programming (XP)
- Adaptive Software Development (ASD)
- Dynamic Systems Development Method (DSDM)
- Scrum
- Crystal
- Feature Driven Development (FDD)
- Agile Modeling (AM)

Extreme Programming (XP) - 1

- The most widely used agile process, originally proposed by Kent Beck [BEC99]
- XP uses an object-oriented approach as its preferred development paradigm
- Defines four (4) framework activities
  - Planning
  - Design
  - Coding
  - Testing

Extreme Programming (XP) - 2



XP - Planning

- Begins with the creation of a set of stories (also called *user stories*)
- Each story is written by the customer and is placed on an index card
- The customer assigns a value (i.e. a priority) to the story
- Agile team assesses each story and assigns a cost
- Stories are grouped to for a deliverable increment
- A commitment is made on delivery date
- After the first increment “project velocity” is used to help define subsequent delivery dates for other increments

XP - Design

- Follows the KIS (keep it simple) principle
- Encourage the use of CRC (class-responsibility-collaborator) cards (Chapter 8)
- For difficult design problems, suggests the creation of "*spike solutions*"—a design prototype
- Encourages "*refactoring*"—an iterative refinement of the internal program design
- Design occurs both before and after coding commences

XP - Coding

- Recommends the construction of a series of unit tests for each of the stories before coding commences
- Encourages "*pair programming*"
  - Mechanism for real-time problem solving and real-time quality assurance
  - Keeps the developers focused on the problem at hand
- Needs continuous integration with other portions (stories) of the s/w, which provides a "smoke testing" environment (Chapter 13)

XP - Testing

- Unit tests should be implemented using a framework to make testing automated. This encourages a regression testing strategy.
- Integration and validation testing can occur on a daily basis
- *Acceptance tests*, also called *customer tests*, are specified by the customer and executed to assess customer visible functionality
- Acceptance tests are derived from user stories