Agile Development

Chapter: 3

Agile Development: The Beginning

- The plan-driven approach.
- Developed by large teams, working for different companies.
- Teams were often geographically dispersed and worked on the software for long periods of time.

An example of this type of software is: the control systems for a modern aircraft, which might take up to 10 years from initial specification to deployment.

- Dissatisfaction with these heavyweight approaches to software engineering led to the development of agile methods in the late 1990s

Agile Development: The 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:

- 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.

Agile Development: 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

Agile Software Process – Three Key Assumptions

1. It is difficult to predict in advance which requirements or customer priorities will change and which will not
2. For many types of software, design and construction activities are interleaved (construction is used to prove the design)
3. Analysis, design, construction and testing are not as predictable (from a planning perspective) as one might like them to be

Principles to Achieve Agility- by Agile Alliance

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 they 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.

Principles to Achieve Agility- by Agile Alliance

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.

The Politics of Agile Development

No one is against agility. The real question is:

- ❑ what is the best way of achieving it?
- ❑ how do you build software that meets customers’ needs today and exhibits the quality characteristics that will enable it to be extended and scaled to meet customers’ need over long term?

There are no absolute answer to either of these questions

Yet, within each agile models (e.g. XP, Scrum), there is a set of ideas.

Human Factors

- **Competence**

  - Encompasses specific software-related skills

  - Skill and knowledge of process

- **Common Focus**

  - All should be focused on one goal (to deliver a working software increment to the customer within the time promised)

  - to achieve this goal, the team will also focus on continual adaptations (small and large)

- **Collaboration**

  - for assessing, analyzing, creating information, the team members must collaborate with one another and all other stakeholders

Human Factors (Contd.)

- **Decision-making ability**

  - Agile team must be allowed the freedom to control its own destiny

- **Fuzzy problem-solving ability**

  - Agile team must accept the fact that the problem they are solving today may not be the problem that needs to be solved tomorrow

- **Mutual trust and respect**

- **Self-organization**

  - The team (1) Organizes itself (2) Organizes Process (3) Organizes Work Schedule

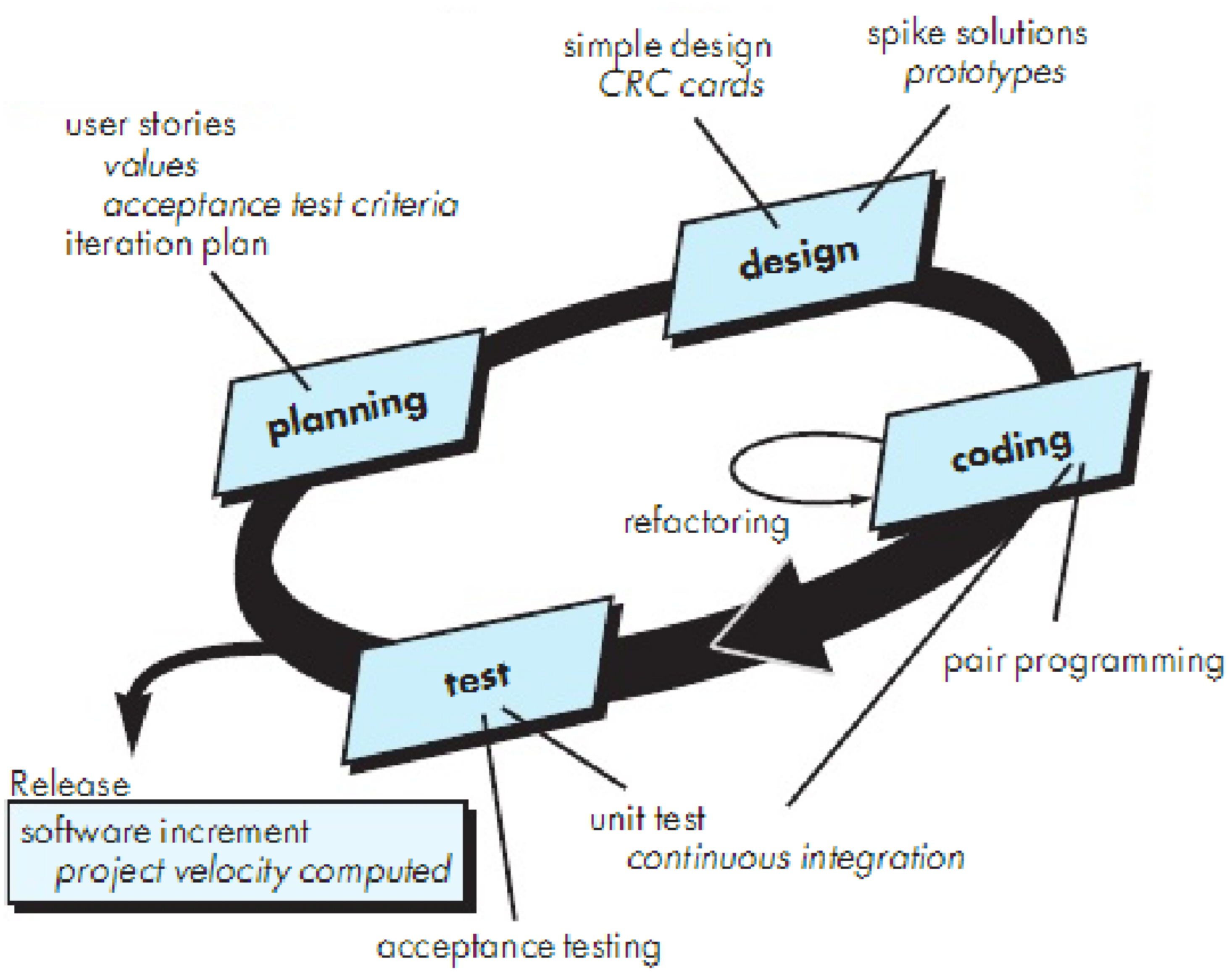
Agile Process Models

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

Extreme Programming (XP)

- 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)



Extreme Programming (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: Planning (User Story)

User Story				
As a ___________	Priority	Team/Member	Status	Estimation
I want to _______				
So that_________				
Acceptance Criteria:________				

Fig: General format of user story.

Title: Online Course Registration				
As a student	Priority	Team	Status	Estimation
• I want to enroll my courses online So that, I don't need to use an hard-copy • I want to pay semester fee online So that, I don't have to carry cash	Low Medium High	☐ DB Designer ☐ Designer ☐ Developer	☐ Pending ☐ Doing ☐ Under-review ☐ Completed	2 Weeks
I know I am done when: • I can enroll my courses in each semester • I can make online payment				

Fig: Example of user story.

Extreme Programming (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): Design (CRC) Models

Class Name	
Responsibilities	Collaborators

Fig: General CRC model diagram.

Customer	
Places an order	Order
Cancel an order	
Enter Name	
Enter Address	
Enter Contact No	
View order history	
Update profile	

Order	
Tracks order date	Order Item
Tracks delivery date	
Records total amount	
Records ordered items	
Maintains order status	

Fig: Example of CRC model diagram for *Customer* and *Order* class.

Extreme Programming (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

Extreme Programming (XP): Testing

- Unit tests should be implemented using a framework to make testing automated
- 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