Extreme Programming (XP)

- One of the **widely** used approach of agile software development.
- XP uses a set of **five values** that establish a foundation for all work per formed
  - **Communication**
  - **Simplicity**
  - **Feedback**
  - **Courage**
  - **Respect**

1. Communication

- For effective communication between software engineers and other stakeholders XP emphasizes close, **yet informal (verbal) collaboration between customers and developers.**
- The establishment of effective **metaphors** for communicating **important concepts**, **continuous feedback**, and **the avoidance of voluminous documentation** as a communication medium

1. Communication

- a **metaphor** is “a story that everyone— customers, programmers, and managers— can tell about how the system works”

2. Simplicity

- To achieve simplicity, XP restricts developers to design only for immediate needs rather than consider **future needs.**
- The intent is to create a **simple design** that can be easily implemented in code
- If the design must be improved, it can be **refactored** at a later time

3. Feedback

- Feedback is derived from three sources: the implemented software itself, the customer, and other software team members.
- Team members deliver software frequently, get feedback about it, and improve a product according to the new requirements.
- By designing and implementing **an effective testing strategy** the software (via test results) provides the agile team with feedback.
- XP makes use of the **unit test** as its primary **testing tactic**

4. Courage

- XP team strictly **adhere discipline**.
- For example, there is often significant pressure to design for future requirements.
- Most software teams succumb, arguing that “designing for tomorrow” will save time and effort in the long run.

4.Courage

- An agile XP team must have the discipline (courage) to design for today, recognizing that future requirements may change dramatically, thereby demanding substantial rework of the design and implemented code.

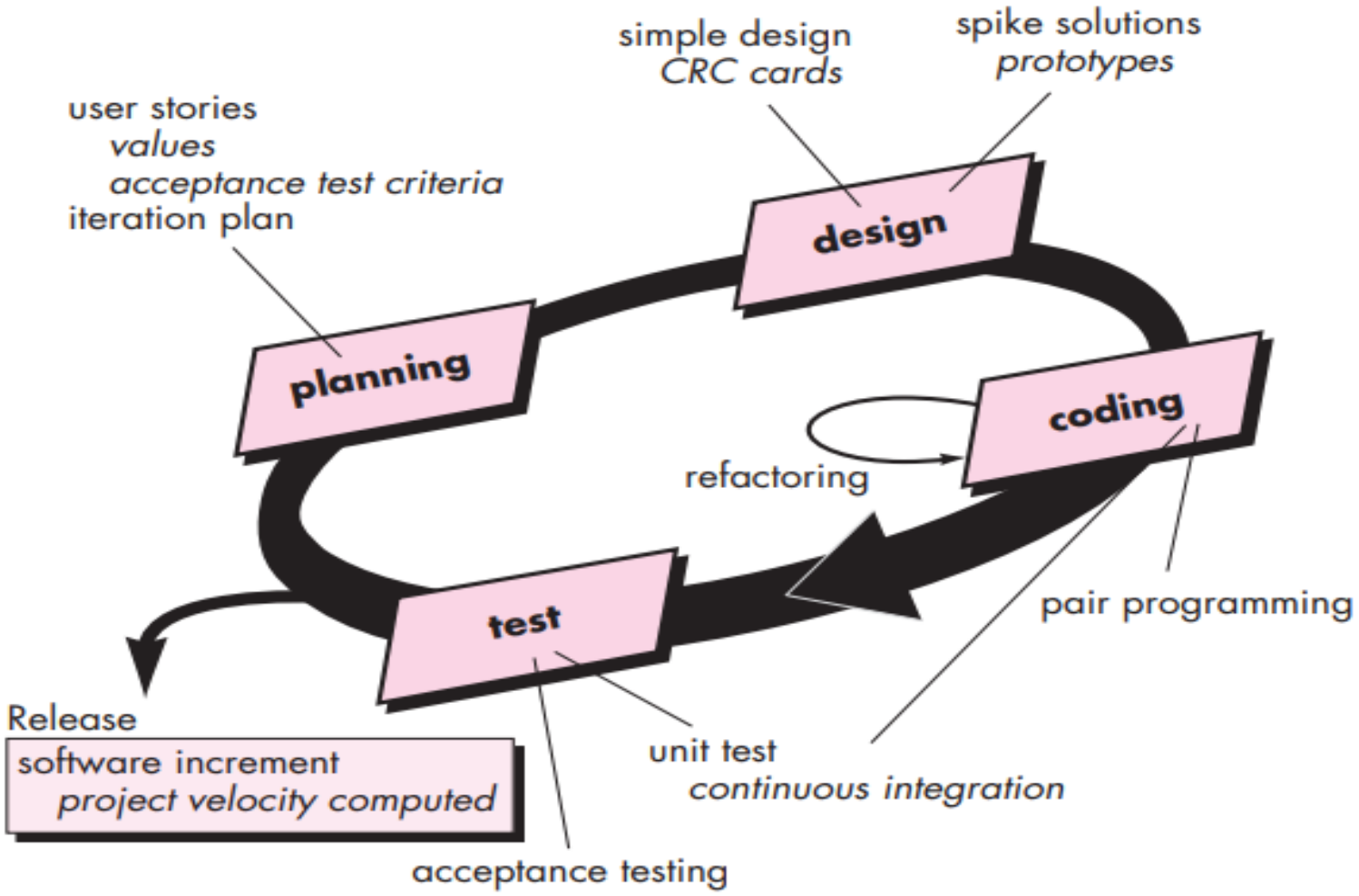
5. Respect

- The agile team respect among it members, between other stakeholders and team members, and indirectly, for the software itself.
- As they achieve successful delivery of software increments, the team develops growing respect for the XP process

The XP Process

- Extreme Programming uses an **object-oriented** approach as its preferred development paradigm
- Encompasses a set of rules and practices that occur within the context of four framework activities:
  - **planning**
  - **design**
  - **coding**
  - **testing.**

XP Process


simple design
CRC cards
spike solutions
prototypes
user stories
values
acceptance test criteria
iteration plan
design
planning
coding
refactoring
test
pair programming
Release
software increment
project velocity computed
unit test
continuous integration
acceptance testing

1. Planning

- The planning activity also called **the planning game begins** with listening—a requirements gathering activity that enables the technical members of the XP team to understand the **business context** for the software and to get a broad feel for required output and major features and functionality.

1. Planning

- Listening leads to the creation of a set of **“stories”** also called **user stories** that describe required output, features, and functionality for software to be built
- Each story similar to use is written by the customer and is placed on an **index card.**
- The customer assigns a **value** i.e a priority to the story based on the overall business value of the feature or function.

1. Planning

- Members of the XP team then assess each story and assign **a cost**—measured in development weeks—to it.
- If the story is estimated to require more than three development weeks, the customer is asked to split the story into smaller stories and the assignment of value and cost occurs again.
- It is important to note that new stories can be written at any time.

1. Planning

- Customers and developers work together to decide how to group stories into the next release (the next software increment) to be developed by the XP team.
- Once a basic commitment (agreement on stories to be included, delivery date, and other project matters) is made for a release, the XP team orders the stories that will be developed in one of three ways:

1. Planning

1. all stories will be implemented immediately (within a few weeks),

2. the stories with highest value will be moved up in the schedule and implemented first

3. the riskiest stories will be moved up in the schedule and implemented first.

2. Design.

- XP design rigorously follows the **KIS (keep it simple) principle.**
- A simple design is always preferred over a more complex representation.
- The design of extra functionality (because the developer assumes it will be required later) is discouraged.

2. Design

- XP encourages the use of **CRC cards** as an effective mechanism for thinking about the software in an object-oriented context.
- CRC (class-responsibility collaborator) cards identify and organize the object-oriented classes that are relevant to the current software increment.

2. Design

- If a difficult design problem is encountered as part of the design of a story, XP recommends the immediate creation of an operational prototype of that portion of the **design Called a spike solution**
- XP encourages **refactoring**—a construction technique that is also a method for design optimization
- **Refactoring** is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves the internal structure

3. Coding

- After stories are developed and preliminary design work is done, the team does not move to code, but rather develops a series of **unit tests**
- Once the unit test has been created, the developer is better able to focus on what must be implemented to pass the test.
- Nothing extraneous is added **(KIS).**
- Once the code is complete, it can be unit-tested immediately, thereby providing instantaneous feedback to the developers

4. Coding

- A key concept during the coding activity **is pair programming.**
- XP recommends that two people work together at one computer workstation to create code for a story.
- This provides a mechanism for real time problem solving (two heads are often better than one) and real-time quality assurance (the code is reviewed as it is created

4. Coding

It also keeps the developers focused on the problem at hand.

In practice, each person takes on a slightly different role.

- One person – coding details of particular design
- Another person – Coding standards

3. Testing

- There creation of **unit tests** before coding commences is a key element of the XP approach.
- The unit tests that are created should be implemented using a **framework** that enables them to be **automated** (hence, they can be executed easily and repeatedly).
- This encourages a **regression testing strategy** whenever code is modified (which is often, given the XP refactor□ing philosophy).

3. Testing

- As the individual unit tests are organized into a "**universal testing suite",** integration and validation testing of the system can occur on a daily basis.
- This provides the XP team with a continual indication of progress and also can raise warning flags early if things go awry.
- "Fixing small problems every few hours takes less time than fixing huge problems just before the deadline."

3. Testing

- **XP acceptance tests**, also **called customer tests**, are specified by the customer and focus on overall system features and functionality that are visible and reviewable by the customer.
- **Acceptance tests** are derived from user stories that have been implemented as part of a software release.