EAST WEST UNIVERSITY

# CSE412
## Software Engineering

**Nishat Tasnim Niloy**

Lecturer

Department of Computer Science and Engineering

Faculty of Science and Engineering

# Topic 2

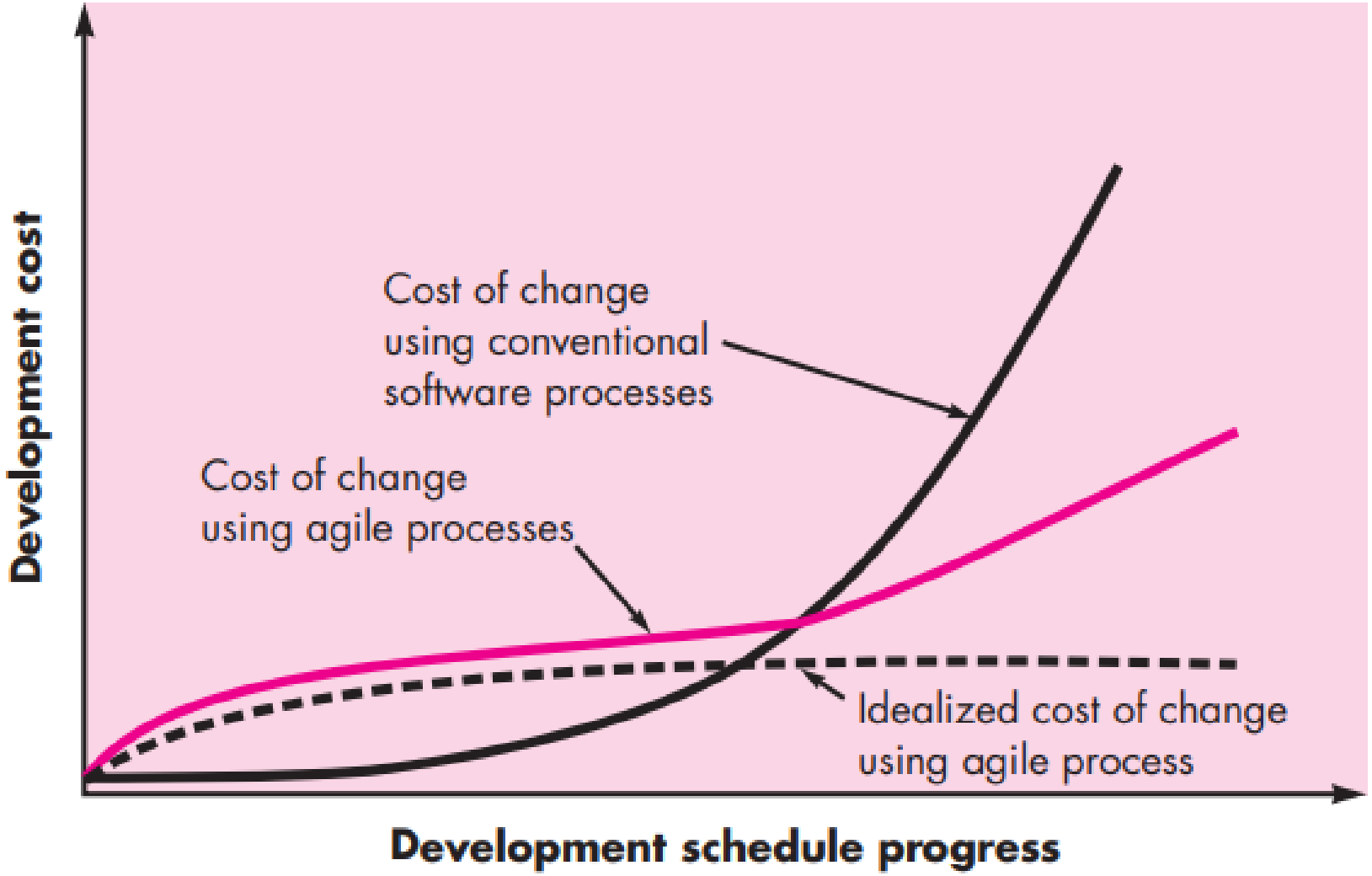
## Agile Development

Adapted from *Software Engineering: A Practitioner's Approach* by Roger Pressman, 7th Edition, Chapter three

# WHAT IS AGILITY?

- Response to the changing requirements appropriately
- Effective communication among all stakeholders
- Drawing the customer onto the team
- Organizing a team so that it is in control of the work performed
- Rapid, incremental delivery of software

# AGILITY AND THE COST OF CHANGE



# WHY IS AGILITY NECESSARY?

- It is difficult to predict in advance which software requirements will persist and which will change. It is equally difficult to predict how customer priorities will change as the project proceeds.
- For many types of software, design and construction are interleaved. It is difficult to predict how much design is necessary before construction is used to prove the design.
- Analysis, design, construction, and testing are not as predictable (from a planning point of view) as we might like.

# 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

# AGILITY PRINCIPLES

- Satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Businesspeople and developers must work together daily throughout the project.

# AGILITY PRINCIPLES

- Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

# AGILITY PRINCIPLES

- Continuous attention to technical excellence and good design enhances agility.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly

# HUMAN FACTOR

- Competence
- Common Focus
- Collaboration
- Decision-making ability
- Fuzzy-problem solving ability
- Mutual Trust and Respect
- Self-organization

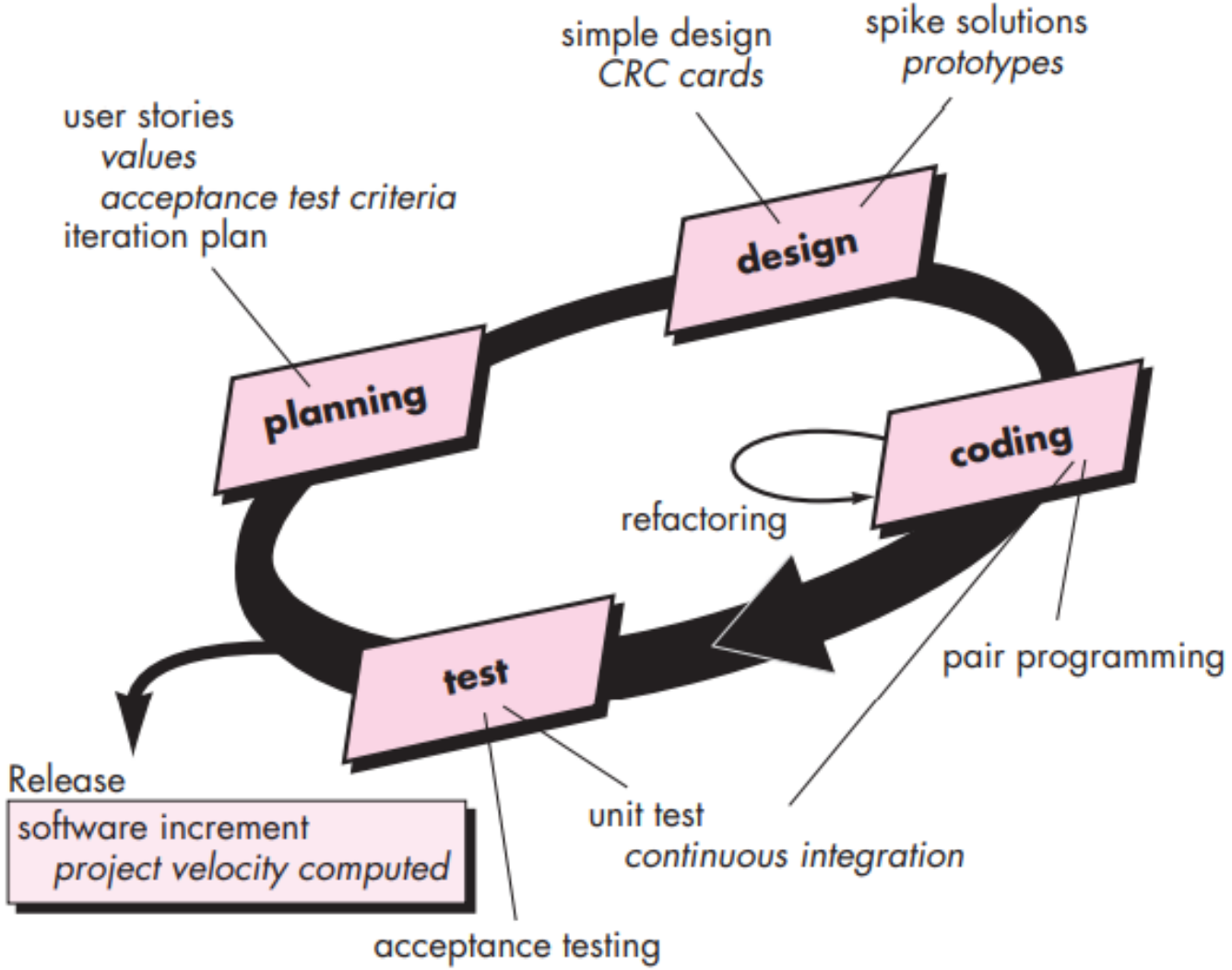
# COMMON AGILE PROCESSES

- Extreme Programming (XP)
- Adaptive Software Development (ASD)
- Scrum
- Dynamic Systems Development Method (DSDM)
- Crystal
- Feature Drive Development (FDD)
- Lean Software Development (LSD)
- Agile Modeling (AM)
- Agile Unified Process (AUP)

# EXTREME PROGRAMMING (XP)

- Most widely used approach to agile software development
- XP is built upon values, principles, and practices
- Its goal is to allow small to mid-sized teams to produce high-quality software and adapt to evolving and changing requirements.
- XP emphasizes the technical aspects of software development. Extreme programming is precise about how engineers work since following engineering practices allows teams to deliver high-quality code at a sustainable pace.
- Extreme programming is about good practices taken to an extreme.
  - pair-programming
  - Early testing (before the production code is written)

# 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

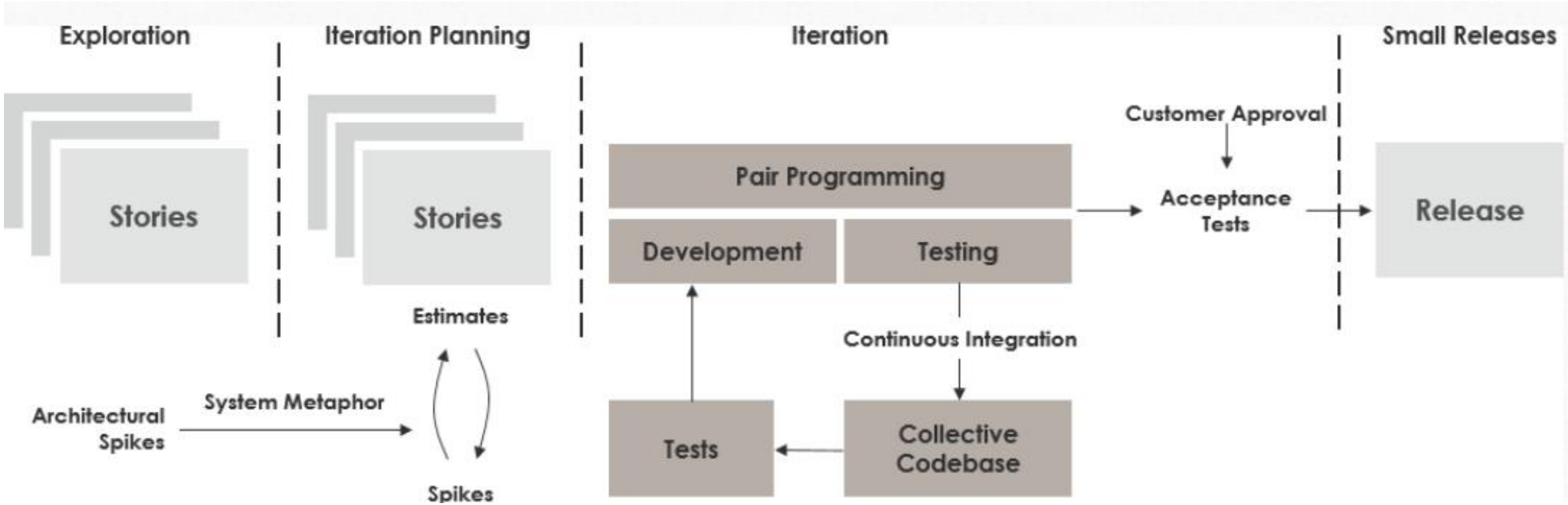
# KEY XP ACTIVITIES

- Coding
- Testing
- Listening
- Designing
- Feedback
- Simplicity
- Pair programming
- Continuous integration
- Refactoring
- Collective code ownership
- Planning Game
- On-site Customer

# ROLES IN XP

- **Tracker**: The tracker goes around and asks each programmer how he or she is doing, listens to the answer, and acts if things seem to be going off track. Actions include suggesting a CRC session, setting up a meeting with the customer, or asking a coach or another programmer to help.
- **Customer**: The customer writes user stories, specifies functional tests, sets priorities, explains stories, and views CRC sessions.
- **Programmer**: The programmer estimates stories, defines engineering tasks from stories, estimates how long stories and tasks will take, and implements stories and unit tests.
- **Tester**: The tester implements and runs functional tests, graphs results, and makes sure people know when test results decline.
- **Coach**: The coach schedules meetings, makes sure the meeting process is followed, and records results of each meeting and passes them to the tracker.

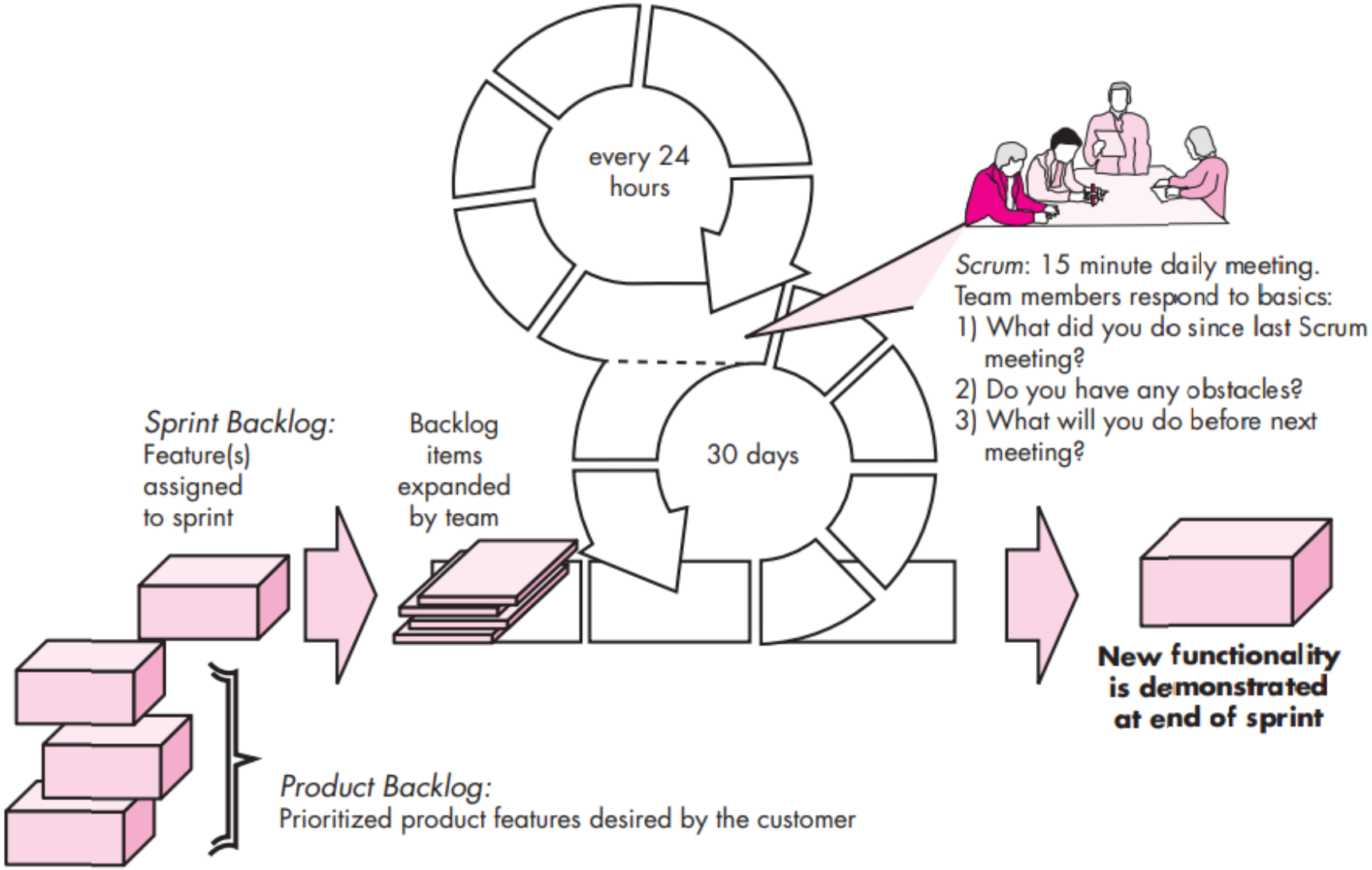
# XP SUMMARY


Exploration
Iteration Planning
Iteration
Small Releases
Stories
Stories
Estimates
Customer Approval
Pair Programming
Acceptance Tests
Release
Development
Testing
Continuous Integration
Architectural Spikes
System Metaphor
Tests
Collective Codebase
Spikes

# SCRUM

- Scrum is an agile development methodology used in the development of Software based on an iterative and incremental processes.
- Scrum is adaptable, fast, flexible and effective agile framework that is designed to deliver value to the customer throughout the development of the project.
- The primary objective of Scrum is to satisfy the customer's need through an environment of transparency in communication, collective responsibility and continuous progress.
- The development starts from a general idea of what needs to be built, elaborating a list of characteristics ordered by priority (product backlog) that the owner of the product wants to obtain.

# SCRUM


every 24 hours
30 days
Scrum: 15 minute daily meeting.
Team members respond to basics:
1) What did you do since last Scrum meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog:
Feature(s) assigned to sprint
Backlog items expanded by team
New functionality is demonstrated at end of sprint
Product Backlog:
Prioritized product features desired by the customer

# ROLES IN SCRUM

- **Scrum master:** The person who leads the team guiding them to comply with the rules and processes of the methodology
- **Product owner (PO):** S/He is the representative of the stakeholders and customers who use the software. They focus on the business part and is responsible for the ROI of the project.
- **Team:** A group of professionals with the necessary technical knowledge who develop the project jointly carrying out the stories they

# BENEFITS OF SCRUM

- Easily Scalable
- Compliance of expectations
- Flexible to changes
- Time to Market reduction
- Higher software quality
- Timely Prediction
- Reduction of risks

# SCRUM VS XP

**Scrum**

- Typically, from two weeks to one month long
- Do not allow changes into their sprints. Once the sprint planning meeting is completed and a commitment made to deliver a set of product backlog items, that set of items remains unchanged through the end of the sprint.
- Scrum product owner prioritizes the product backlog, but the team determines the sequence in which they will develop the backlog items.
- Doesn't prescribe any engineering practices

**XP**

- Typically, one or two weeks long
- Much more amenable to change within their iterations. If the team hasn't started work on a particular feature, a new feature of equivalent size can be swapped into the XP team's iteration in exchange for the un-started feature
- Work in a strict priority order. Features to be developed are prioritized by the customer
- Provides engineering practices, like-TDD, pair programming, simple design, refactoring

# REFACTORING

- Programming team look for possible software improvements and make these improvements even where there is no immediate need for them.
- This improves the understandability of the software and so reduces the need for documentation.
- Changes are easier to make because the code is well-structured and clear.
- However, some changes requires architecture refactoring, and this is much more expensive.

# PAIR PROGRAMMING

- In XP, programmers work in pairs, sitting together to develop code.
- This helps develop common ownership of code and spreads knowledge across the team.
- It serves as an informal review process as each line of code is looked at by more than 1 person.
- It encourages refactoring as the whole team can benefit from this.
- Measurements suggest that development productivity with pair programming is similar to that of two people working independently.
- Pairs are created dynamically so that all team members work with each other during the development process.
- The sharing of knowledge that happens during pair programming is very important as it reduces

# Summary

- Brief introduction regarding agile process
- Different agile processes
  - Model descriptions
  - Model illustrations
  - Benefits and limitations