Agile development methods

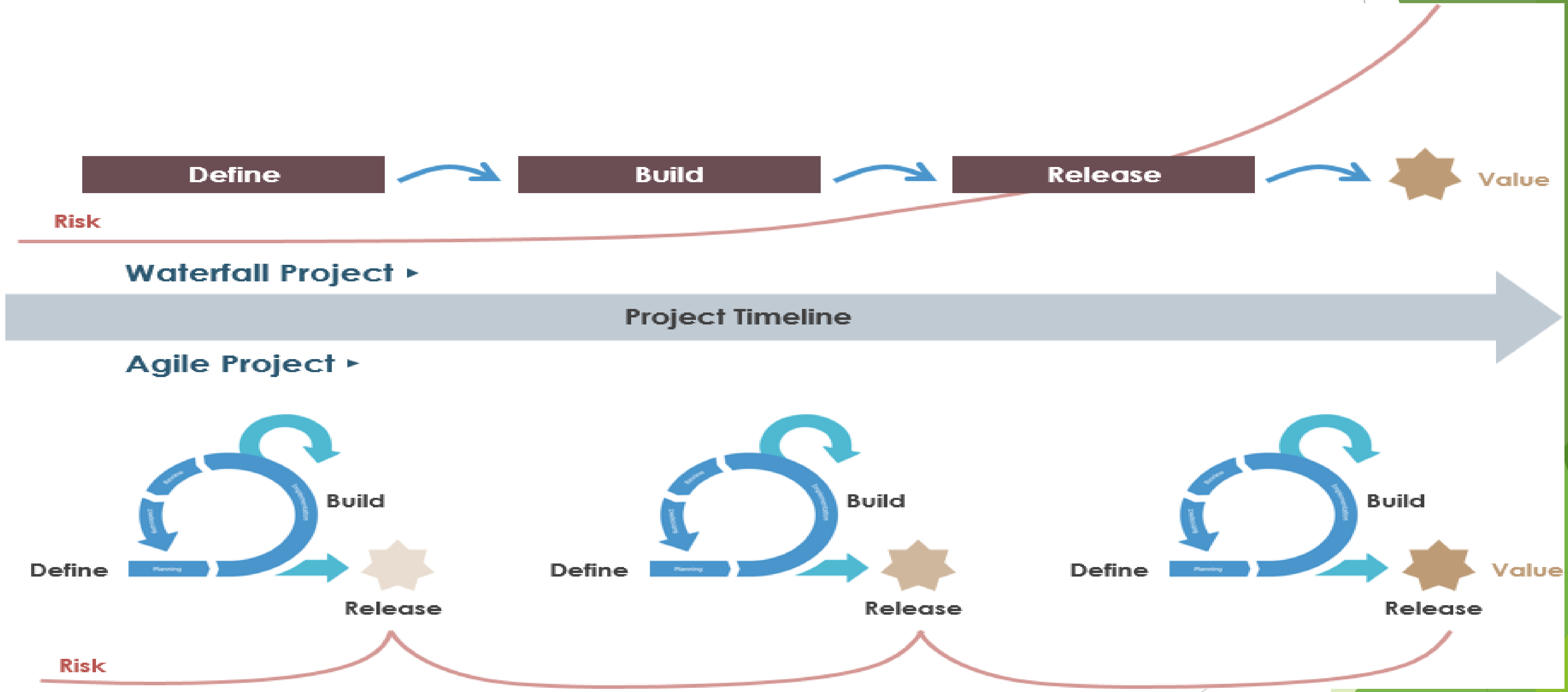
- Agile development

- The agile software development model was proposed in the mid-1990s to overcome the serious shortcomings of the heavy-weight models like waterfall model of development.
- Agile software development is an under which requirements and solutions evolve through the collaborative approach to software development effort of self-organizing and cross-functional teams and their customer(s)/end user(s).
- It advocates adaptive planning, evolutionary development, early delivery, and continual improvement, and it encourages rapid and flexible response to change.
- Agile model is being used as an umbrella term to refer to a group of development processes.
- Agile software development methods support a broad range of the software development life cycle.

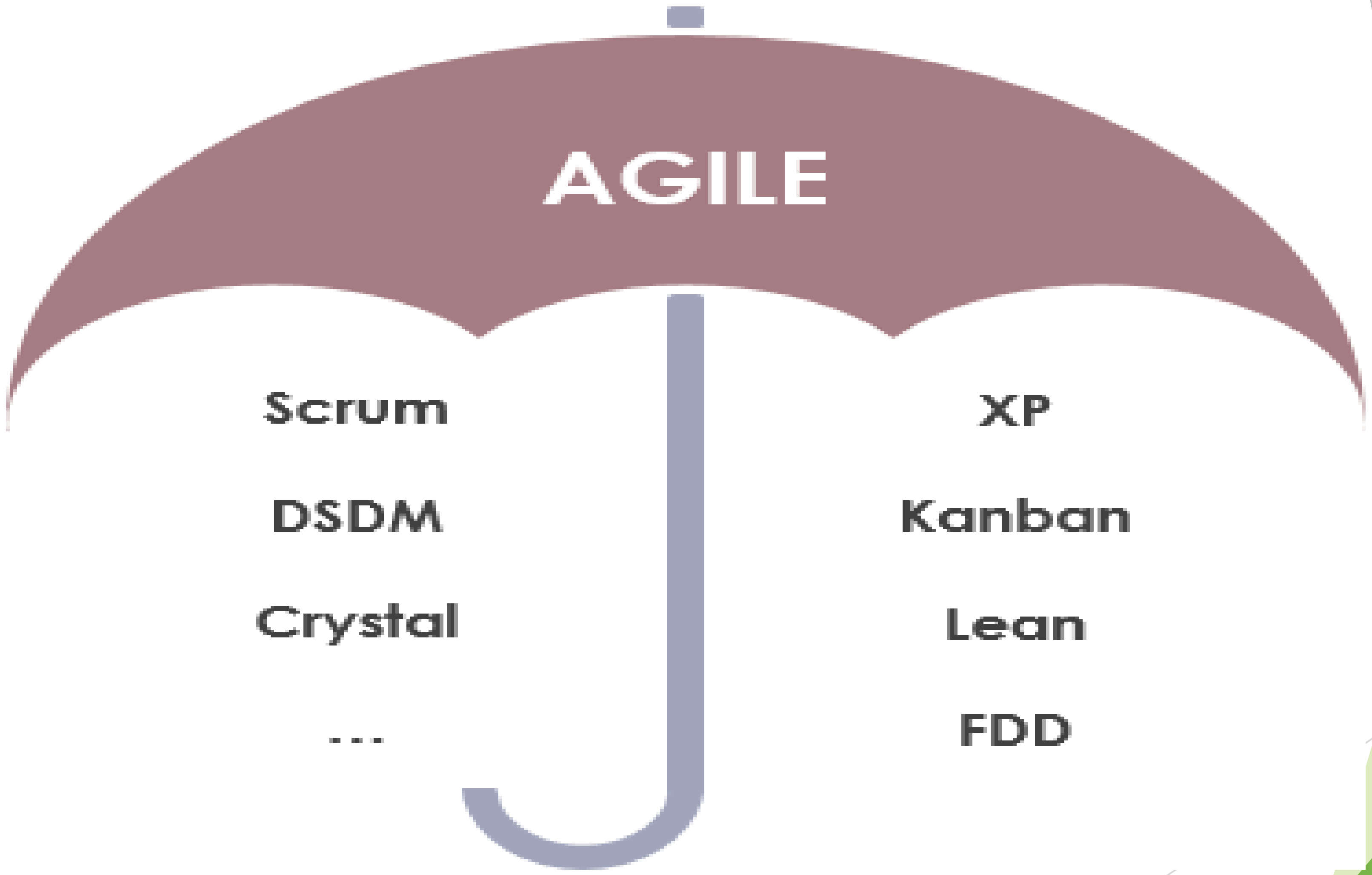
- Some focus on the practices (e.g., XP, pragmatic programming, agile modelling),
- some focus on managing the flow of work (e.g.. Scrum, Kanban).
- Some support activities for requirements specification and development (e.g., FDD),
- some seek to cover the full development life cycle (e.g., DSDM, RUP).

- Agile is a term used to describe approaches to software development emphasizing
- incremental delivery,
- team collaboration,
- continual planning, and
- continual learning, instead of trying to deliver it all at once near the end.

Agile vs Waterfall model



Agile methods



- In the agile model, the requirements are decomposed into many small parts that can be incrementally developed.
- The agile model adopts an iterative approach. Each incremental part is developed over an iteration.
- Each iteration is intended to be small and easily manageable and lasting for a couple of weeks only.
- At a time, only one increment is planned, developed, and then deployed at the customer site.
- No long-term plans are made.
- The time to complete an iteration is called a time box.
- The implication of the term time box is that the end date for an iteration does not change.

- For establishing close contact with the customer during development and to gain a clear understanding of the domain-specific issues, each agile project usually includes a customer representative in the team.
- At the end of each iteration, stakeholders and the customer representative review the progress made and re-evaluate the requirements.
- A distinguishing characteristics of the agile models is frequent delivery of software increments to the customer.
- Agile model emphasize face-to-face communication over written documents.
- It is recommended that the development team size be deliberately kept small (5-9 people)
- It is implicit then that the agile model is suited to the development of small projects.
- However, if a large project is required to be developed using the agile model, it is likely that the collaborating teams might work at different locations.
- In this case, the different teams are needed to maintain as much daily contact as possible through video conferencing, telephone, e-mail, etc.

The agile manifesto has four important values:

- 1. Focus should be more on individuals and interactions instead of processes and tools
- 2. Working software is more important that comprehensive documentation.
- 3. Customer collaboration is more vital than contract negotiation
- 4. The process should respond to change rather than follow a plan

The following important principles behind the agile model were publicized in the agile manifesto in 2001:

- 1. Deliver customer satisfaction by delivering valuable software continuously
- 2. Always accept change of requirements matter how early or late in the project
- 3. Deliver software that works within a shorter timescale
- 4. Both developers and business professionals must work closely together daily throughout the duration of the project
- 5. Information is best transferred between parties in face-to-face conversations
- 6. Motivate people to build a project by creating an environment of appreciation, trust, and empowerment

- Advantages:
- Customer satisfaction by rapid, continuous delivery of useful software.
- Dividing into small groups gives the team the opportunity to focus on the individual stages and work faster.
- Developers can devote more time to interesting tasks and their potential development, instead of preparing formal reports.
- The team can focus on development, testing, and collaboration.
- People and interactions are emphasized rather than process and tools.
- Customers, developers and testers constantly interact with each other.

Disadvantages:

- At the commencement of the project, it is difficult to accurately determine the amount of time and money that will be needed to complete the project due to constantly changing requirements.
- The team needs to have a solid foundation and comparable skilllevel.
- A high level of interaction between the client and the developers is required, which can take time and make the process difficult.
- Lack of attention to documentation can make it difficult for new team members to access needed information.
- There is a danger that the lack of project boundaries will lead to uncontrolled expansion, which can cause the project to never reach completion.

Agile methods

- Extreme Programming
- ASD
- Scrum
- DSDM
- FDD
- LSD
- Agile Modeling
- Agile Unified Process..

Extreme programming

Extreme programming (XP)

- it is one of the most important and widely used agile software development framework and was proposed by Kent Beck in 1999.
- The name of this model reflects the fact that it recommends taking the best practices that have worked well in the past in program development projects to extreme levels.
- Extreme Values
- Extreme Programming is based on five core values communication,

simplicity,

feedback,

respect,

courage.

- These five fundamental values provide the foundation on which the entirety of the Extreme Programming paradigm is built, allowing the people involved in the project to feel confident in the direction the project is taking and to understand their personal feedback and insight.

- The XP process

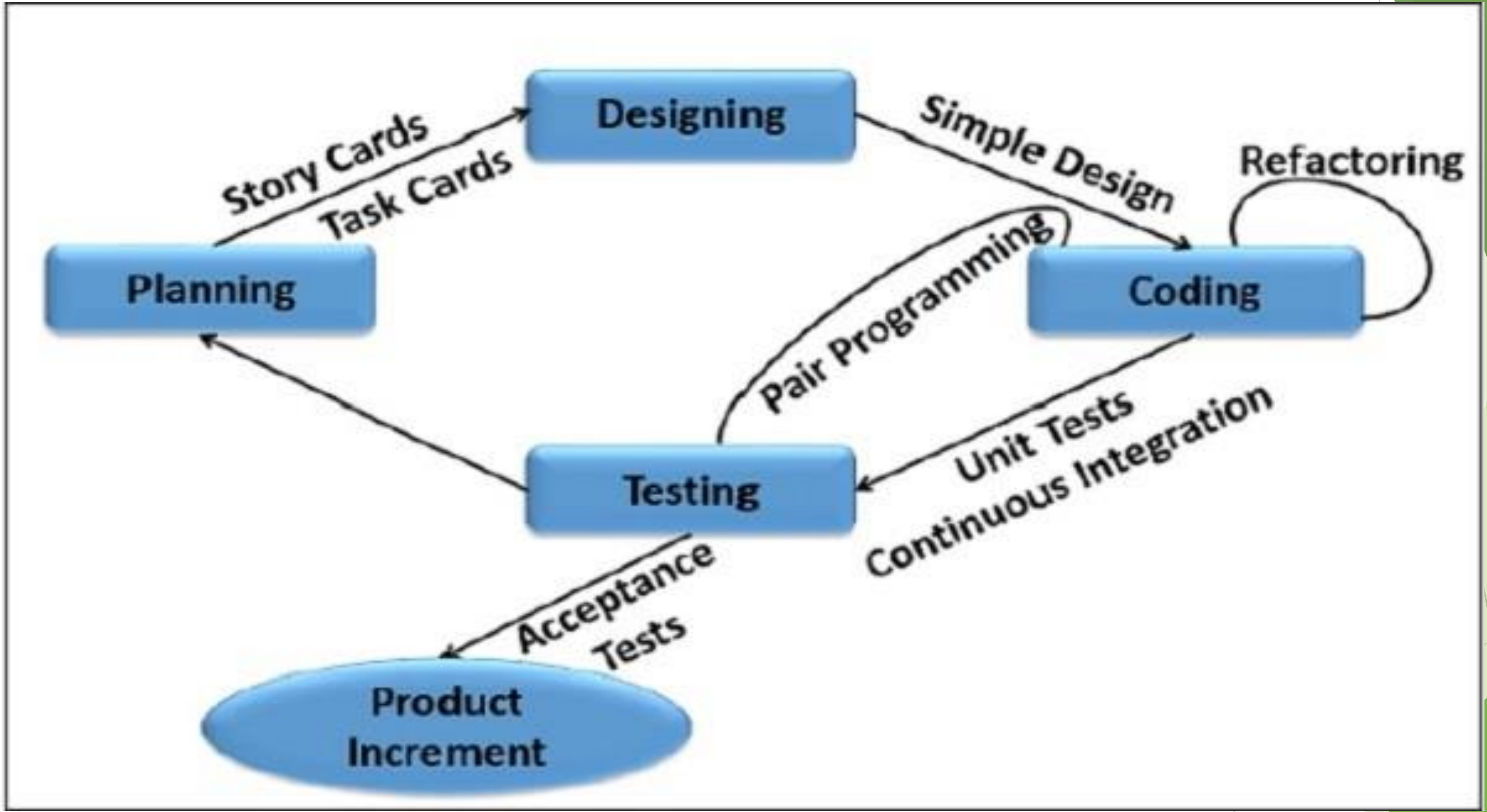
- 4 activities

Planning

Design

Coding

Testing


Designing
Story Cards
Task Cards
Simple Design
Refactoring
Planning
Pair Programming
Coding
Unit Tests
Continuous Integration
Testing
Acceptance Tests
Product Increment

- The XP Process
- Extreme Programming uses an object-oriented approach as its preferred development paradigm and encompasses a set of rules and practices that occur within the context of four framework activities: planning, design, coding, and testing.
- Extreme Planning: The first phase of Extreme Programming life cycle is planning.
- The planning activity 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.
-

- Listening leads to the creation of a set of user stories that describe required output, features, and functionality for software to be built.
- Each story (similar to use cases) 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. 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.

- Extreme Design:
- XP design rigorously follows the KIS (keep it simple) principle.
- A simple design is always preferred over a more complex representation.
- In addition, the design provides implementation guidance for a story as it is written-nothing less, nothing more.
- The design of extra functionality (because the developer assumes it will be required later) is discouraged.
- XP encourages the use of CRC (class-responsibility-collaborator) cards to identify and organize the object-oriented classes that are relevant to the current software increment.

- If a difficult design problem is encountered as part of the design of a story. XP recommends the immediate creation of an operational prototype, called a spike solution, of that portion of the design.
- The spike solution is implemented and evaluated. The objective is to lower risk when true implementation starts and to validate the original estimates for the story containing the design problem.
- XP encourages refactoring continuously as soon as potential code improvements are found.
- Refactoring is a disciplined way to clean up code [and modify/simplify the internal design] that minimizes the chances of introducing bugs.
- This keeps the code simple and maintainable.

- Extreme 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 that will exercise each of the stories that is to be included in the current release (software increment).
- 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. Once the code is complete, it can be unit-tested immediately, thereby providing instantaneous feedback to the developers.
- 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.
- In practice, each person takes on a slightly different role. One types in code (think about the coding details) while the other reviews the code as it is typed in (ensures that coding standards).
- The two programmers switch their roles every hour or so.

- Extreme 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 wrong.
- 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.

Advantageous of XP

- Extreme Programming allows software development companies to save costs and time required for project realization. XP eliminates unproductive activities to reduce costs and frustration of everyone involved. It allows developers to focus on coding.
- Extreme Programming reduces the risks related to programming and project failure. XP ensures that the client gets exactly what he wants.
- Simplicity is a core value of Extreme Programming projects. It creates extremely simple code that can be improved at any moment.
- In Extreme Programming, the whole process is visible and accountable. The developers make concrete commitments about what they will accomplish and show concrete progress.

- Constant feedback demonstrates the software early and often and it enables the developers to listen carefully to the customers and to make any changes needed. Sprints help the team to move in the right direction.
- Extreme Programming creates working software faster. Regulartesting at the development stage ensures detection of all bugs, andthe use of customer approved validation tests to determine thesuccessful completion of a coding block ensures implementation ofonly what the customer wants and nothing more.
- Extreme Programming helps increase employee satisfaction and retention. Extreme Programming is a value-driven approach that sets fixed work time, with little scope for overtime. The breakdown of project scope into subcomponents and the constant customer feedback prevents accumulation of much work to be completed before a tight deadline.
- Extreme Programming promotes teamwork. Everyone is part of the team. Team members work together on everything from requirements to code. Developers work in pairs and never feel alone or forgotten.

Disadvantages of XP

- Code overcomes design. The focus of XP is definitely the code rather than the design. The design is what sells the application, so the customer could be unhappy with the end product if the design is not good enough. Sometimes this can result in failing to implement the software requirements fully.
- Location. XP projects are difficult to implement when the customer is away from the developmental team. Typically the XP interactions are successful when the team members meet face to face. Therefore applying extreme programming limits the range of projects.
- Lack of documentation. The constant changes cannot be documented properly. Thus, there are high risks of unexpected failures that cannot be tracked. Even when bugs are fixed, without accurate documentation it is possible that the same errors can recur.
- Stress. There is a lot of pressure working with tight deadlines. If the developers have high stress levels completing tasks on time, they are more likely to make mistakes while coding. Subsequently, software quality could be reduced due to the scheduling.