# Software Engineering

Dr. Rasha Montaser

# Lec 3

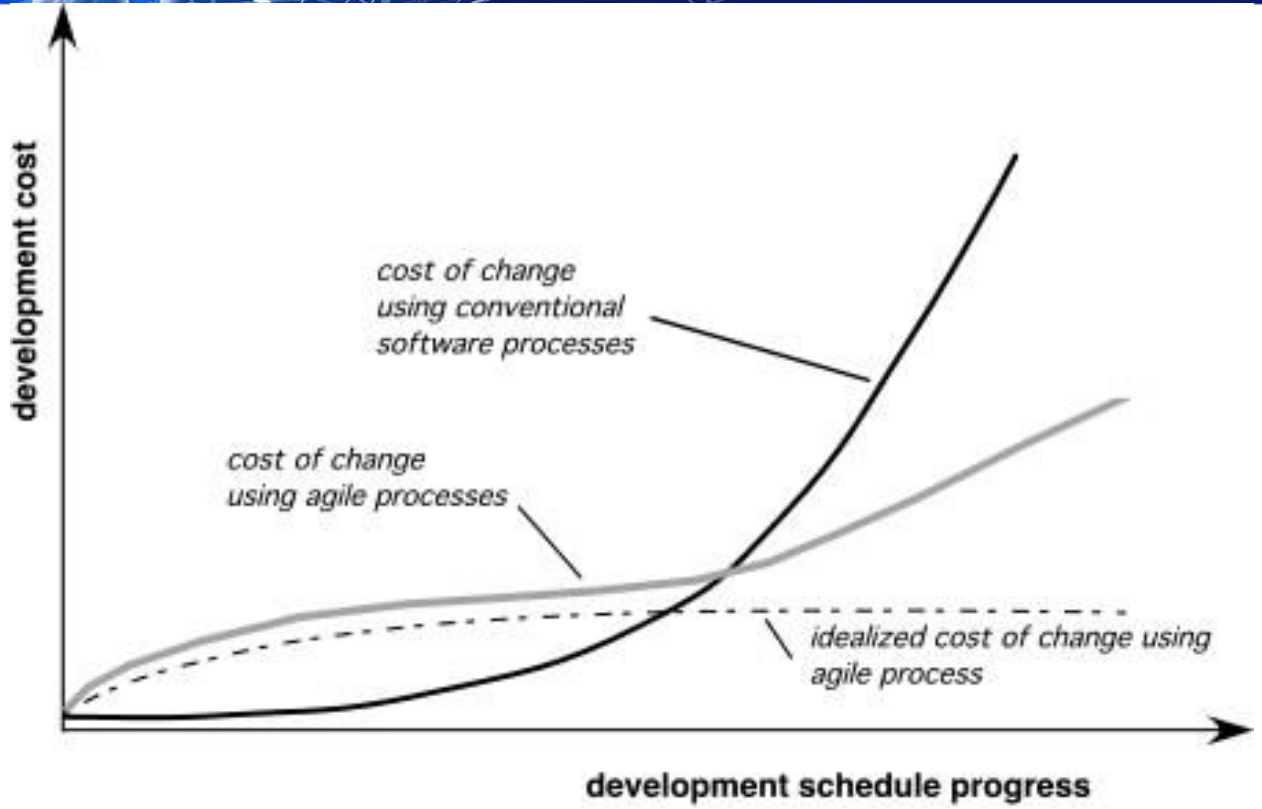
# Agile view of process

Ref: Chapter 4

# What is “Agility”?

- Effective (rapid and adaptive) response to change
- Effective communication among all stakeholders
- Drawing the customer onto the team
- Organizing a team so that it is in control of the work performed

# Agility and the Cost of Change



# An 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 - I

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.

# Agility Principles - II

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.

# Human Factors

- *the process molds to the needs of the people and team,* not the other way around
- key traits must exist among the people on an agile team and the team itself:

  - **Competence. –** all team members have skill and knowledge of the process
  - **Common focus. –** deliver working increment within the time promised
  - **Collaboration. –** collaborate with each other, customers and business manager
  - **Decision-making ability. –** team members have freedom to control their own purpose
  - **Fuzzy problem-solving ability.** ability to solve problems and accept the concept that the problem they are solving today may not be a problem for tomorrow
  - **Mutual trust and respect.-** trust and respect each other
  - **Self-organization.**
    - Organize itself for the work to be done
    - Organize the process to best accommodate its local environment
    - Organize the work schedule to best achieve delivery of the software

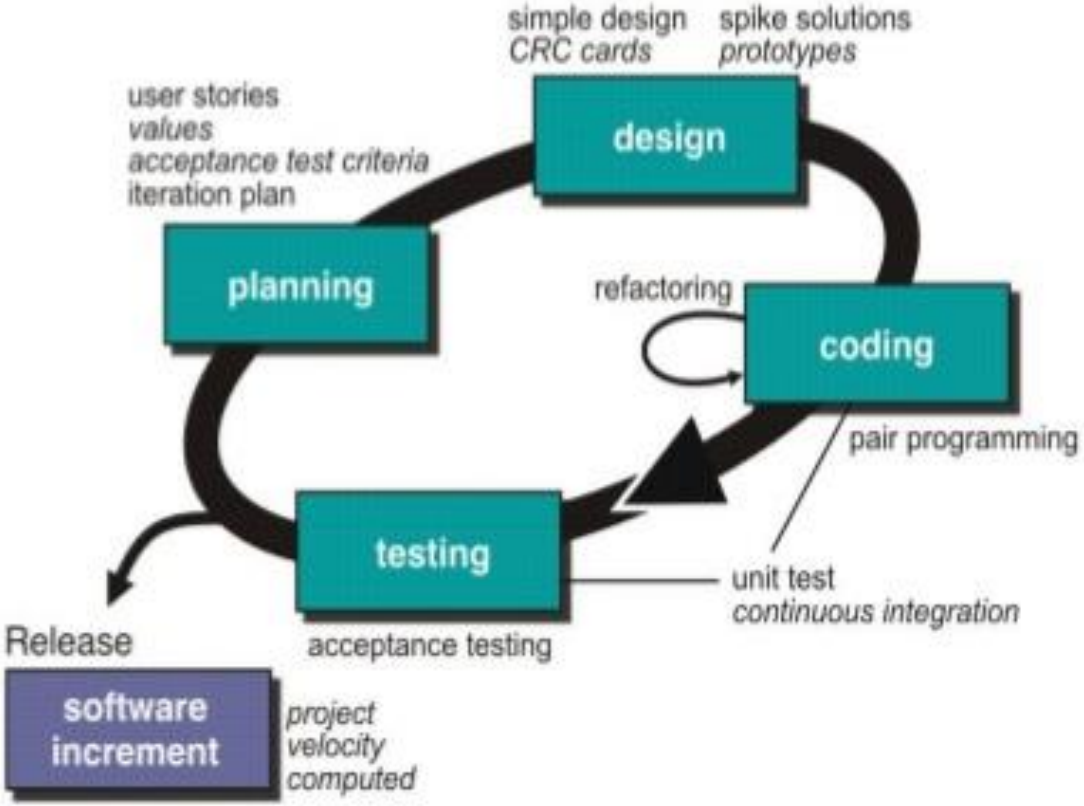
# Agile process Model

1. Extreme programming
2. Adaptive software development
3. Dynamic systems development method
4. scrum

# Extreme Programming (XP)

- The most widely used agile process, originally proposed by Kent Beck
- XP Planning
  - Begins with the creation of "user stories"
  - 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

# Extreme Programming (XP)



# Extreme Programming (XP)

- **XP Design**
  - Create set of stories
  - Each story is written by customer and placed in index card
  - A customer assign value (priority) to the story
  - XP team assess each story and assign cost (development weeks) to it
  - If the story require more than 3 weeks the customer splits it into smaller stories
  - After first release decide project velocity (number of customers stories implemented during the first release)

# Extreme Programming (XP)

- **XP Planning**
  - Follows the KIS (keep it simple) principle
  - Encourage the use of CRC (class responsibility collaborator) cards
  - For difficult design problems, suggests the creation of "spike solutions"—a design prototype
  - Encourages "refactoring"—an iterative refinement of the internal program design

# Extreme Programming (XP)

- **XP Coding**
  - Recommends the construction of a unit test for a store *before* coding commences
  - Encourages "pair programming" – two people work together at one work station to create code for a story

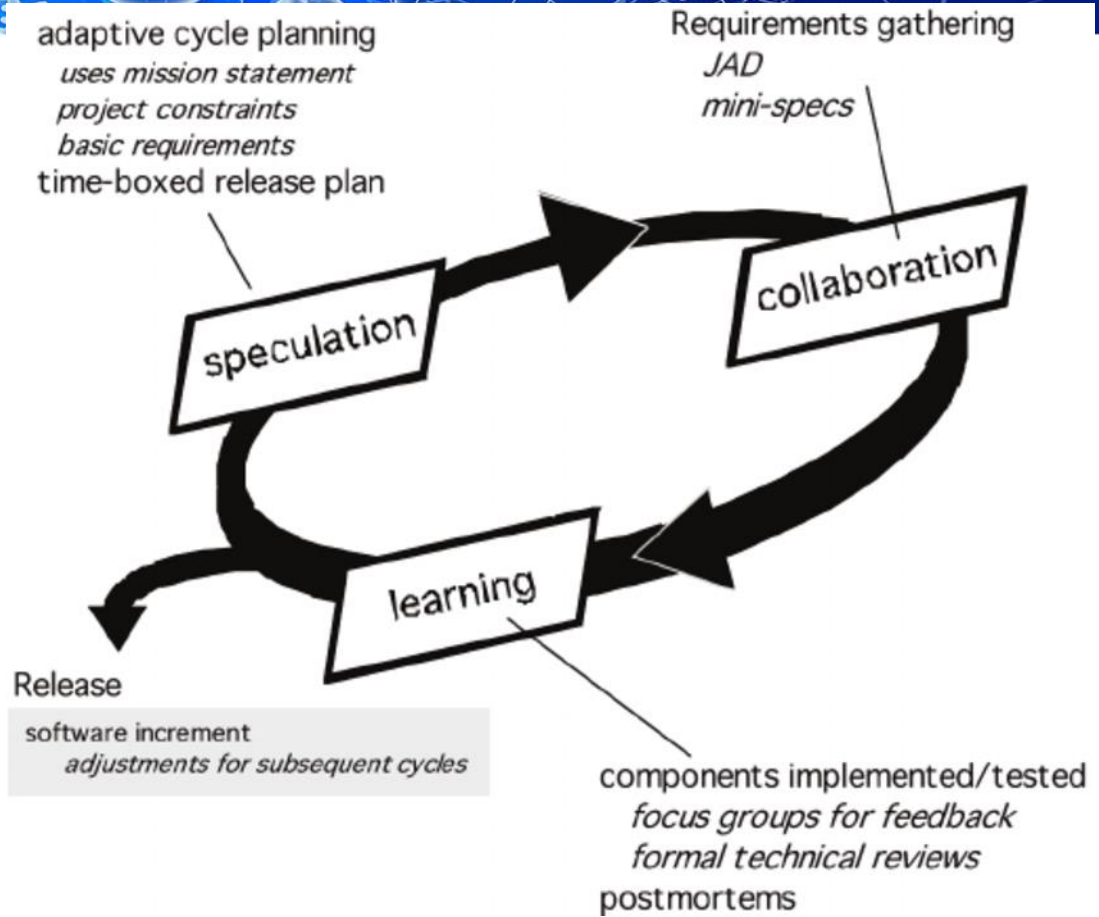
# Extreme Programming (XP)

- **XP Testing**
  - All unit tests are executed daily
  - "Acceptance tests" are defined by the customer and executed to assess customer visible functionality

# Adaptive Software Development

- Originally proposed by Jim Highsmith
- ASD — distinguishing features
  - Used to build complex software and systems
  - Mission-driven planning
  - Component-based focus
  - Explicit consideration of risks
  - Emphasizes collaboration for requirements gathering
  - Emphasizes “learning” throughout the process

# Adaptive Software Development



# Adaptive Software Development

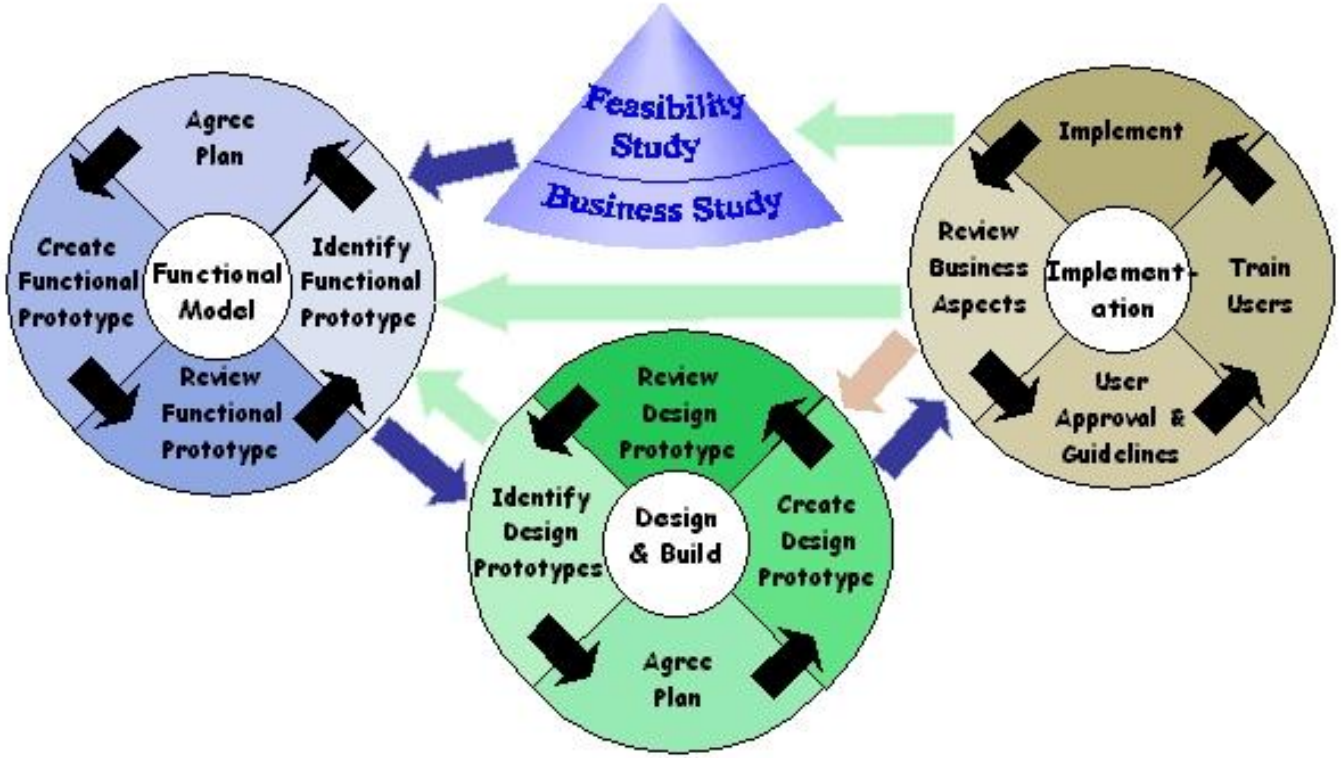
- **Speculation** - Customers state project constrains and basic requirements
- **Collaboration** – motivate people to work together in a way that multiplies their talent and creative output
- **Learning**
  - Customers provide feedback on software increments
  - Team members review software components to improve quality
  - Team members address their own performance and process

# Dynamic Systems Development Method

- Provide framework for building and maintaining systems which meet tight time constraint through the use of incremental prototyping in a controlled project environment
- Similar in most respects to XP and/or ASD
- Nine guiding principles
  - Active user involvement is imperative.
  - DSDM teams must be empowered to make decisions.
  - The focus is on frequent delivery of products.
  - Fitness for business purpose is the essential criterion for acceptance of deliverables.
  - Iterative and incremental development is necessary to converge on an accurate business solution.
  - All changes during development are reversible.
  - Requirements are baselined at a high level
  - Testing is integrated throughout the life-cycle.

# Dynamic Systems Development Method

## The DSDM Development Process



# Feasibility Study

- establishes the basic business requirements and constraints associated with the application to be built and then assesses whether the application is a viable candidate for the DSDM process.

# Business Study

- establishes the functional and information requirements that will allow the application to provide business value; also, defines the basic application architecture and identifies the maintainability requirements for the application.

# Functional model iteration

- produces a set of incremental prototypes that demonstrate functionality for the customer. (Note: All DSDM prototypes are intended to evolve into the deliverable application.) The intent during this iterative cycle is to gather additional requirements by eliciting feedback from users as they exercise the prototype.

# *Design and build iteration*

- revisits prototypes built during *functional model iteration* to ensure that each has been engineered in a manner that will enable it to provide operational business value for end users. In some cases, *functional model iteration* and *design and build iteration* occur concurrently.

# *Implementation*

- places the latest software increment (an "operationalized" prototype) into the operational environment. It should be noted that

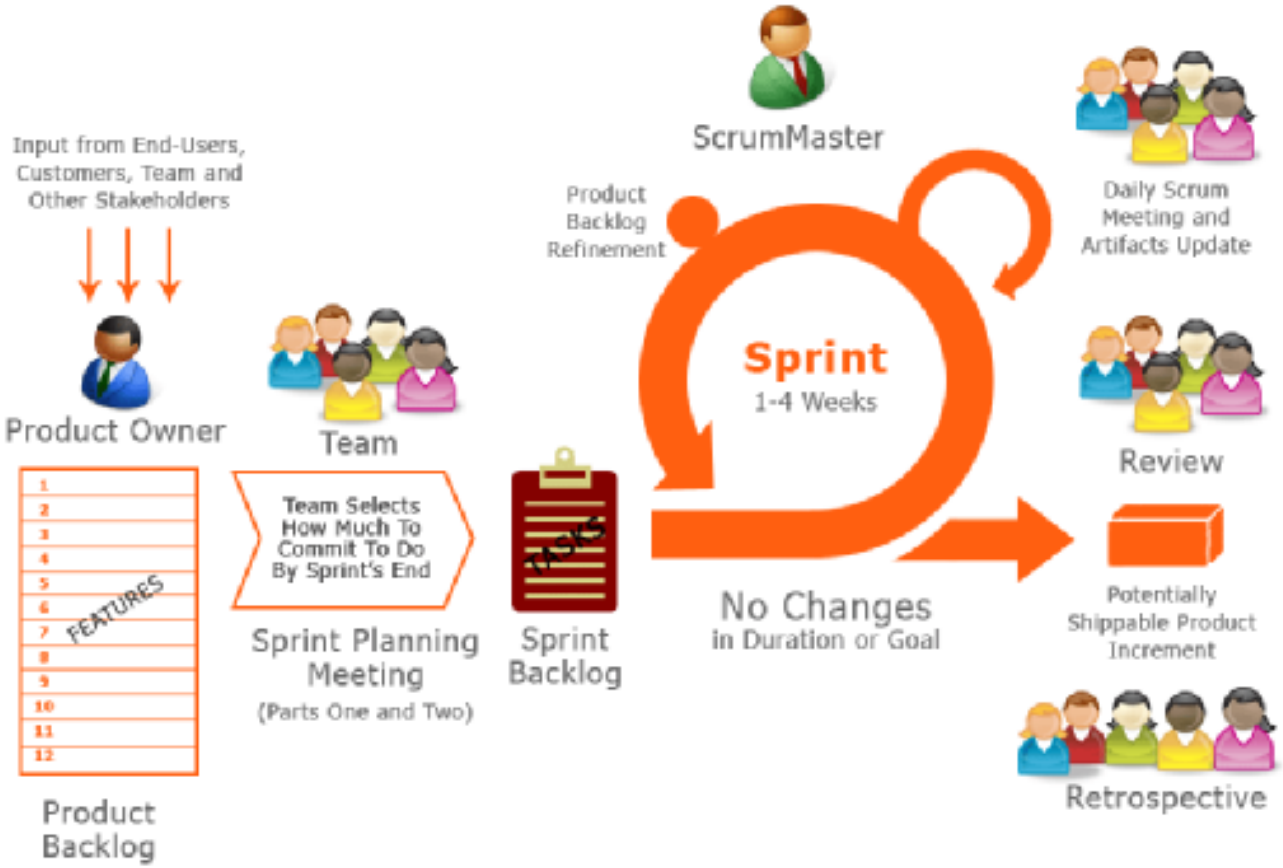
1. the increment may not be 100 percent complete or
2. changes may be requested as the increment is put into place.

- In either case, DSDM development work continues by returning to the functional model iteration activity.

# Scrum

- Scrum—distinguishing features
  - Development work is partitioned into "packets"
  - Testing and documentation are on-going as the product is constructed
  - Work occurs in "sprints" (cycles) and is derived from a "backlog" of existing requirements
  - Meetings are very short and sometimes conducted without chairs
  - "demos" are delivered to the customer with the time-box allocated

# Scrum


ScrumMaster
Input from End-Users, Customers, Team and Other Stakeholders
Product Backlog Refinement
Daily Scrum Meeting and Artifacts Update
Sprint
1-4 Weeks
Product Owner
Team
Review
Team Selects How Much To Commit To Do By Sprint's End
TASKS
FEATURES
1
2
3
4
5
6
7
8
9
10
11
12
No Changes
in Duration or Goal
Potentially Shippable Product Increment
Sprint Planning Meeting
(Parts One and Two)
Sprint Backlog
Retrospective
Product Backlog

# Scrum development activities

- Backlog – personalize list of project requirements or features
- Sprints – work units that are required to achieve requirements defined in the backlog that must fit in predefined time box

# Scrum development activities

- Scrum meetings – short meetings held daily by the scrum team.
  - What did you do since the last team meeting?
  - What obstacles are you encountering?
  - What do you plan to accomplish by the next team meeting?
- Demos – deliver the software increment to the customer so that the functionality that has been implemented can be demonstrated and evaluated by the customers.

# When to use scrum?

- Tight timelines
- Changing requirements
- Business criticality

| AGILE MODEL | EXPLORATORY PROGRAMMING |
| --- | --- |
| Agile model is an incremental delivery process where each incremental delivered part is developed through an iteration after each timebox. | Exploratory programming is an approach of writing programs in an unstructured way. |
| Agile teams, however, do follow defined and disciplined processes and carry out systematic requirements gathering, rigorous design. | Exploratory programming does not follow the rules of software engineering and unstructured coding is done and tested. |
| The central idea of the Agile model is to deliver an incremental version to the customer frequently after each iteration. | Whereas, after coding the software is tested and the founded bugs are fixed. This cycle of testing and bug fixing continues till the software works satisfactorily for the customer. |

| AGILE MODEL | INCREMENTAL DEVELOPMENT MODEL |
| --- | --- |
| Agile model is an incremental delivery process where each incremental delivered part is developed through an iteration after each time box. The main principle of the Agile model is to achieve agility by removing unnecessary activities that waste time and effort. | The requirements of the software are divided into several modules that can be incrementally developed and delivered. The core features are developed first and the whole software is developed by adding new features in successive versions. |
| In the Agile model, end date for an iteration is fixed, it cannot be changed. The development team may have to decide to reduce the delivered functionality to complete that iteration on time. | In the Incremental development model, there is no fixed time to complete the next iteration. |

| AGILE MODEL | SPIRAL MODEL |
| --- | --- |
| The main principle of the Agile model is to achieve agility by removing unnecessary activities that waste time and effort. | The main principle of the Spiral model is risk handling. |
| The Agile model focuses on the delivery of an increment to the customer after each Time-box, so customer interaction is more frequent. | Spiral model mainly deals with various kinds of unanticipated risks but customer interaction is less. |
| Agile model is suitable for large projects that are easy to divide into small parts that can be easily developed incrementally over each iteration. | The Spiral model is suitable for those projects that are prone to various kinds of risks that are difficult to anticipate at the beginning of the project. |
| Agile model does not rely on documentation. | Proper documentation is required for Spiral model. |