

# Software Development Life Cycle - Process Models

Week 2

# PROCESS MODELS

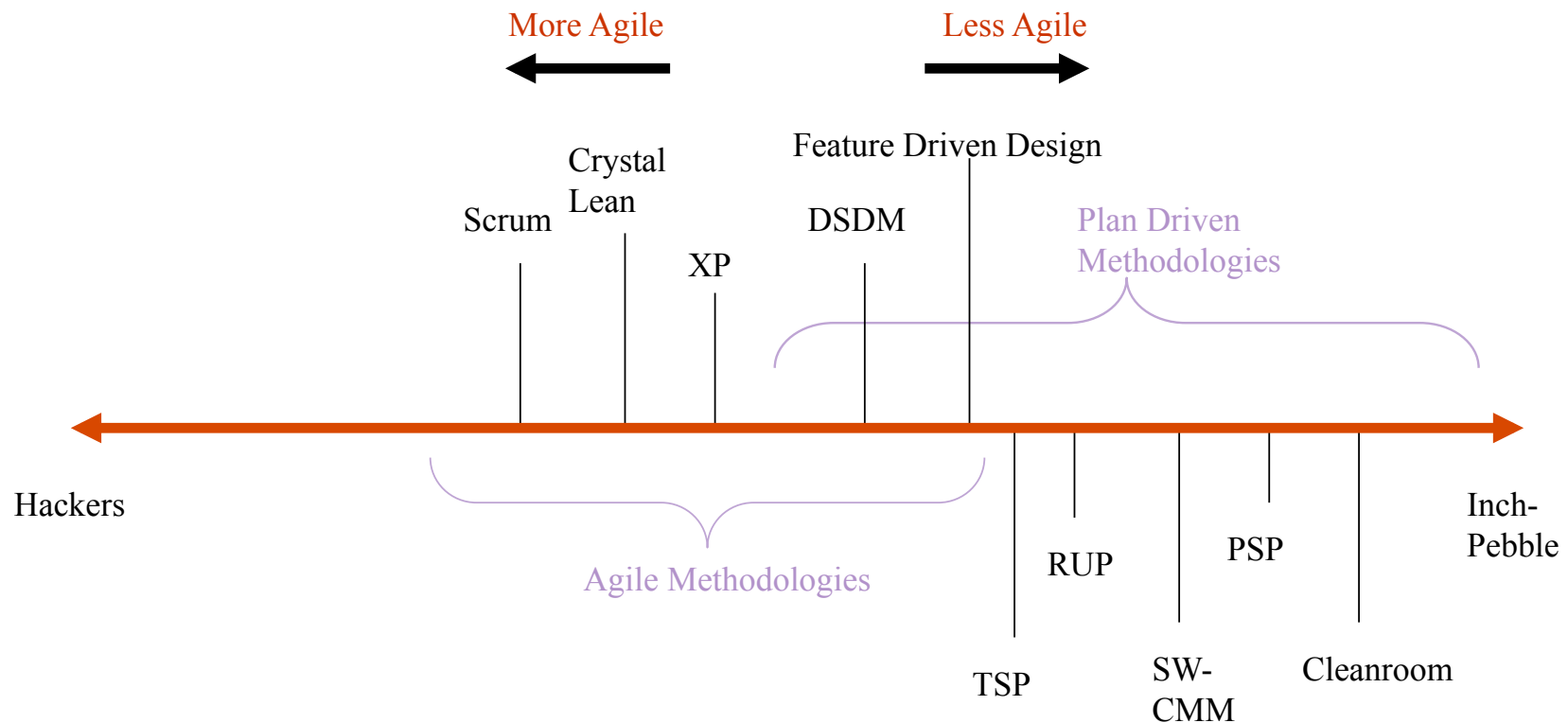
---

- ▶ Many life cycle models have been proposed
- ▶ Traditional Models (plan-driven)
  - ▶ Classical waterfall model
  - ▶ Iterative waterfall
  - ▶ Evolutionary
  - ▶ Prototyping
  - ▶ Spiral model
  - ▶ Rational Unified Process (RUP)
- ▶ Agile Models
  - ▶ eXtreme Programming (XP)
  - ▶ Scrum
  - ▶ Crystal
  - ▶ Feature-Driven Development (FDD)



# The Process Methodology Spectrum

---

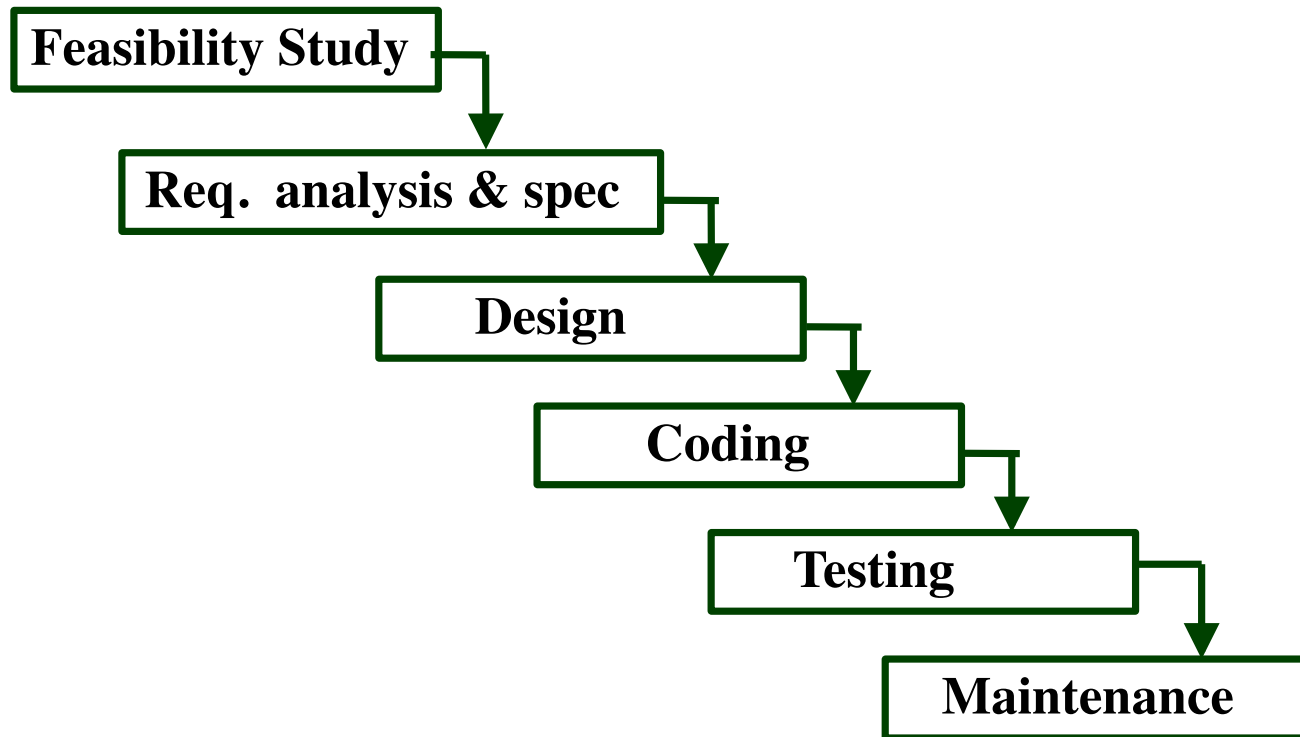


from “Balancing Agility & Discipline” (Boehm & Turner)



# CLASSICAL WATERFALL MODEL

---



# CLASSICAL WATERFALL MODEL (CONT.)

---

- ▶ The guidelines and methodologies of an organization:
  - ▶ called the organization's software development methodology.
- ▶ Software development organizations:
  - ▶ expect fresh engineers to master the organization's software development methodology.



# Problems with Classical Waterfall Model

---

- ▶ Classical waterfall model is idealistic:
  - ▶ assumes that no defect is introduced during any development activity.
  - ▶ in practice:
    - ▶ defects do get introduced in almost every phase of the life cycle.
- ▶ Defects usually get detected much later in the life cycle:
  - ▶ For example, a design defect might go unnoticed till the coding or testing phase



# PROTOTYPING MODEL

---

- ▶ Before starting actual development,
  - ▶ a working prototype of the system should first be built.
- ▶ A prototype is a toy implementation of a system:
  - ▶ limited functional capabilities,
  - ▶ low reliability,
  - ▶ inefficient performance.

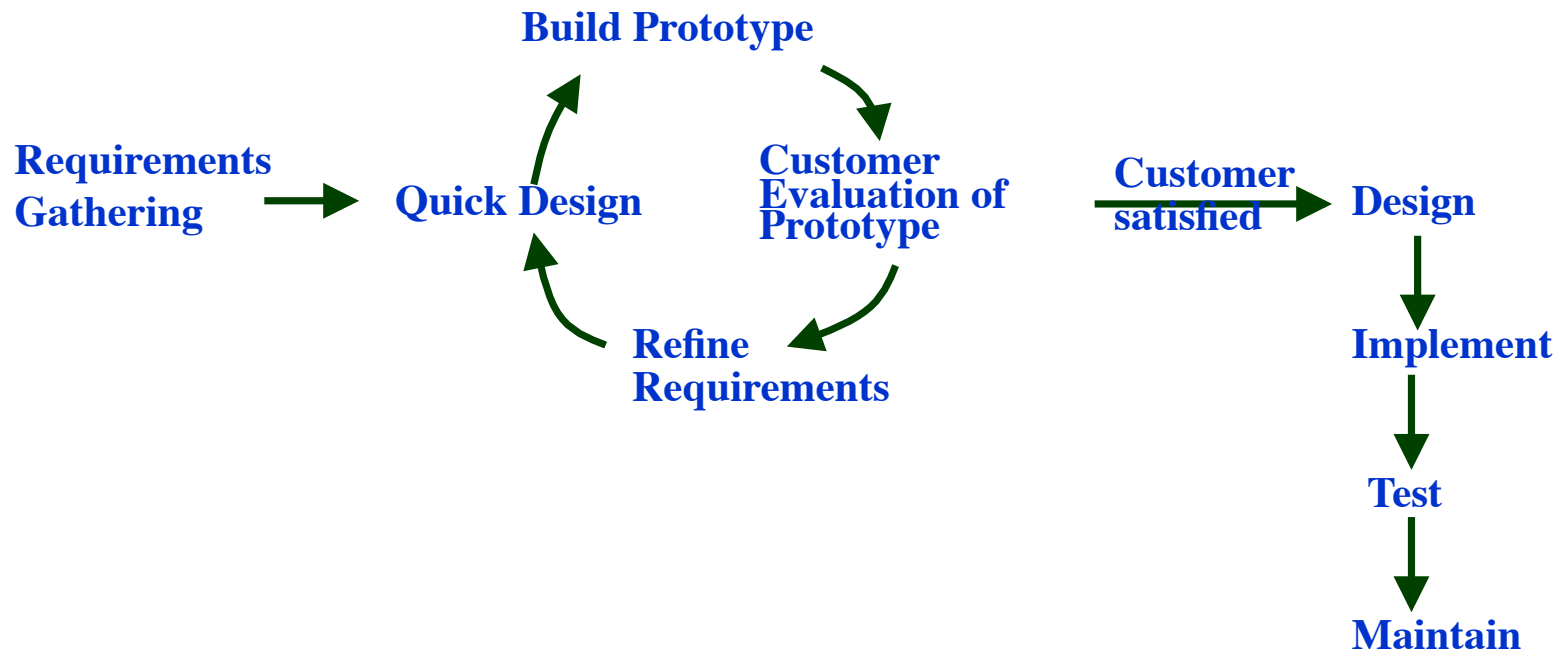
## WHY PROTOTYPE?

---



# PROTOTYPING MODEL (CONT.)

---





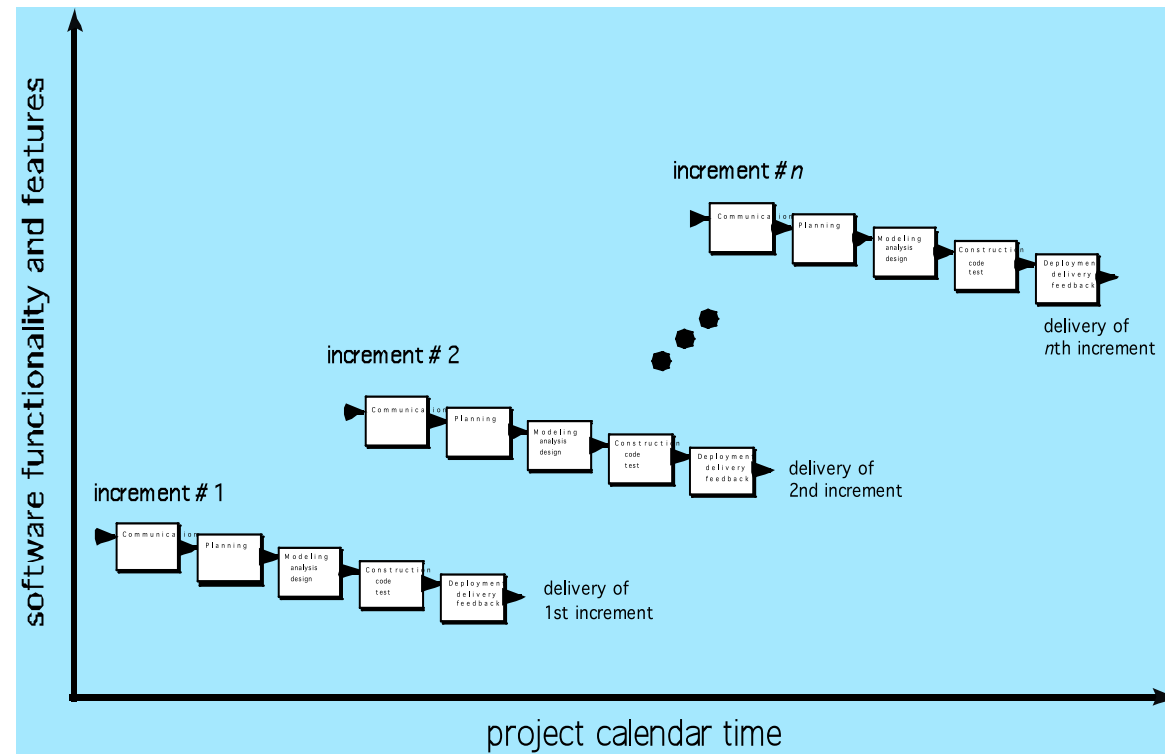
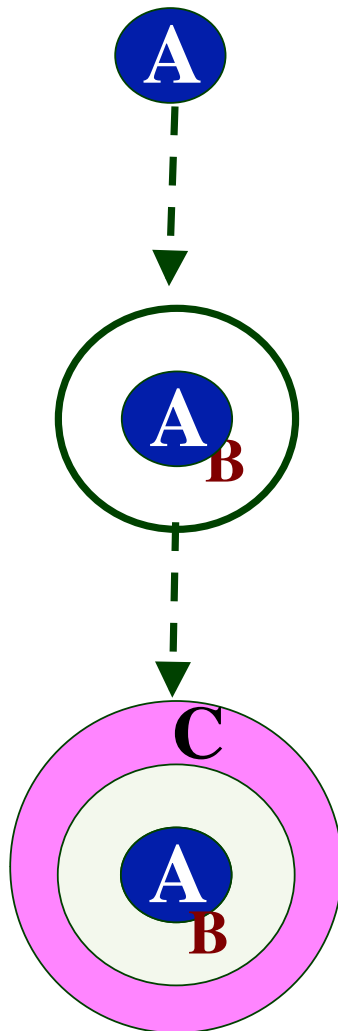
# EVOLUTIONARY MODEL

---

- ▶ Evolutionary model (aka successive versions or incremental model):
  - ▶ The system is broken down into several modules which can be incrementally implemented and delivered.
- ▶ First develop the core modules of the system.
- ▶ The initial product skeleton is refined into increasing levels of capability:
  - ▶ by adding new functionalities in successive versions.



# INCREMENTAL MODEL



# ADVANTAGES OF EVOLUTIONARY MODEL

---

- ▶ Users get a chance to experiment with a partially developed system:
  - ▶ much before the full working version is released,
- ▶ Helps finding exact user requirements:
  - ▶ much before fully working system is developed.
- ▶ Core modules get tested thoroughly:
  - ▶ reduces chances of errors in final product.



# DISADVANTAGES OF EVOLUTIONARY MODEL

---

- ▶ Often, difficult to subdivide problems into functional units:
  - ▶ which can be incrementally implemented and delivered.
  - ▶ evolutionary model is useful for very large problems,
    - ▶ where it is easier to find modules for incremental implementation.



## EVOLUTIONARY MODEL WITH ITERATION (Iterative Incremental Model)

---

- ▶ Many organizations use a combination of iterative and incremental development:
  - ▶ a new release may include new functionality
  - ▶ existing functionality from the current release may also have been modified.



# EVOLUTIONARY MODEL WITH ITERATION

---

- ▶ **Several advantages:**
  - ▶ Training can start on an earlier release
    - ▶ customer feedback taken into account
  - ▶ Markets can be created:
    - ▶ for functionality that has never been offered.
  - ▶ Frequent releases allow developers to fix unanticipated problems quickly.



# SPIRAL MODEL

---

- ▶ Proposed by Boehm in 1988.
- ▶ Each loop of the spiral represents a phase of the software process:
  - ▶ the innermost loop might be concerned with system feasibility,
  - ▶ the next loop with system requirements definition,
  - ▶ the next one with system design, and so on.
- ▶ There are no fixed phases in this model, the phases shown in the figure are just examples.



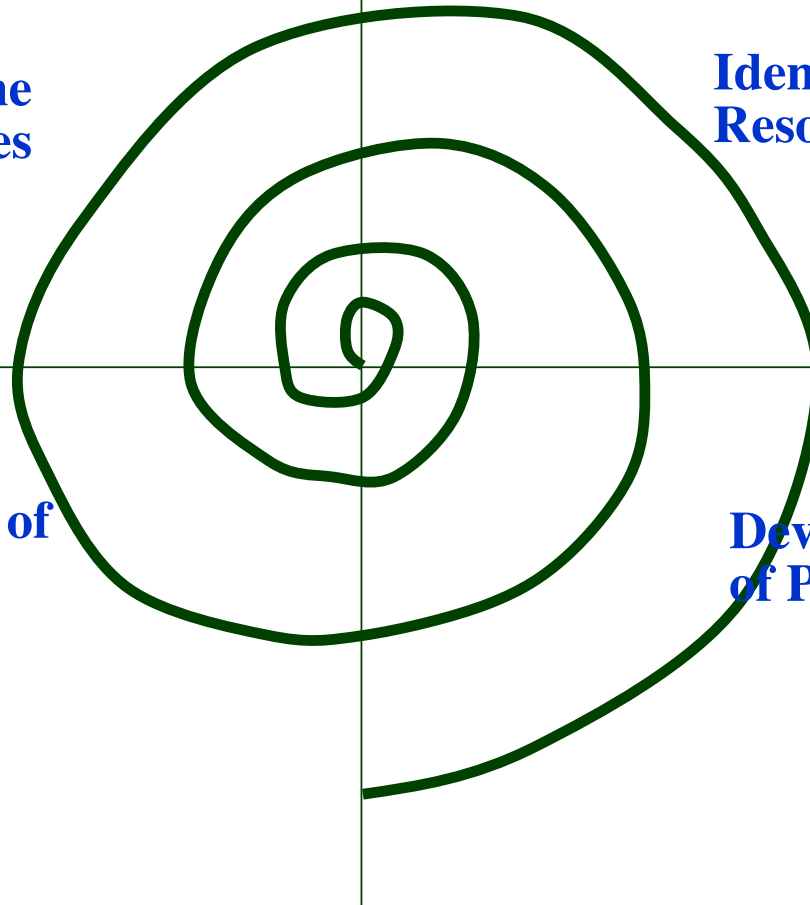
# SPIRAL MODEL (CONT.)

**Determine  
Objectives**

**Identify &  
Resolve Risks**

**Customer  
Evaluation of  
Prototype**

**Develop Next Level  
of Product**





## OBJECTIVE SETTING (FIRST QUADRANT)

---

- ▶ Identify objectives of the phase,
- ▶ Examine the risks associated with these objectives.
  - ▶ Risk:
    - ▶ any adverse circumstance that might hamper successful completion of a software project.
- ▶ Find alternate solutions possible.



# RISK ASSESSMENT AND REDUCTION (SECOND QUADRANT)

---

- ▶ For each identified project risk,
  - ▶ a detailed analysis is carried out.
- ▶ Steps are taken to reduce the risk.
- ▶ For example, if there is a risk that the requirements are inappropriate:
  - ▶ a prototype system may be developed.



## SPIRAL MODEL (CONT.)

---

- ▶ Development and Validation (**Third quadrant**):
  - ▶ develop and validate the next level of the product.
- ▶ Review and Planning (**Fourth quadrant**):
  - ▶ review the results achieved so far with the customer and plan the next iteration around the spiral.
- ▶ With each iteration around the spiral:
  - ▶ progressively more complete version of the software gets built.



# SPIRAL MODEL AS A META MODEL

---

- ▶ Subsumes all discussed models:
  - ▶ a single loop spiral represents waterfall model.
  - ▶ uses an evolutionary approach --
    - ▶ iterations through the spiral are evolutionary levels.
  - ▶ enables understanding and reacting to risks during each iteration along the spiral.
  - ▶ uses:
    - ▶ prototyping as a risk reduction mechanism
    - ▶ retains the step-wise approach of the waterfall model.



# COMPARISON OF DIFFERENT LIFE CYCLE MODELS

---

- ▶ Iterative waterfall model
  - ▶ most widely used model.
  - ▶ But, suitable only for well-understood problems.
- ▶ Prototype model is suitable for projects not well understood:
  - ▶ user requirements
  - ▶ technical aspects



# COMPARISON OF DIFFERENT LIFE CYCLE MODELS (CONT.)

---

- ▶ **Evolutionary model is suitable for large problems:**
  - ▶ can be decomposed into a set of modules that can be incrementally implemented,
  - ▶ incremental delivery of the system is acceptable to the customer.
- ▶ **The spiral model:**
  - ▶ suitable for development of technically challenging software products that are subject to several kinds of risks.



## Agile processes

# Agile 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.



# Some Agile Methods

---

- ▶ ASD - Adaptive Software Development
- ▶ Crystal
- ▶ FDD - Feature Driven Development
- ▶ DSDM - Dynamic Systems Development Method
- ▶ Lean Software Development
- ▶ Scrum
- ▶ XP - eXtreme Programming

## Four Values

---

- ▶ Simplicity
  - ▶ create the simplest thing that could work
- ▶ Communication
  - ▶ face-to-face, not document-to-face
- ▶ Feedback
  - ▶ lots of tests
- ▶ Aggressiveness

## Four Basic Activities

---

- ▶ Coding
  - ▶ cannot do without it
- ▶ Testing
  - ▶ if it cannot be tested it doesn't exist
- ▶ Listening
  - ▶ to those with domain knowledge
- ▶ Designing
  - ▶ to keep the system from decaying

## Twelve Practices

---

1. The Planning Game
2. Small releases
3. Metaphor
4. Simple design
5. Testing
6. Refactoring
7. Pair programming
8. Collective ownership
9. Continuous integration
10. 40-hour week
11. On-site customer
12. Coding standards

# Process Assessment and Improvement

---

- **Standard CMMI Assessment Method for Process Improvement (SCAMPI)** — provides a five step process assessment model that incorporates five phases: initiating, diagnosing, establishing, acting and learning.
- **CMM-Based Appraisal for Internal Process Improvement (CBA IPI)**—provides a diagnostic technique for assessing the relative maturity of a software organization
- **SPICE—The SPICE (ISO/IEC15504)** standard defines a set of requirements for software process assessment.
- **ISO 9001:2000 for Software**—a generic standard that applies to any organization that wants to improve the overall quality of the products, systems, or services that it provides



# SUMMARY

---

- ▶ There are various process models:
  - ▶ Traditional (Plan-driven) Models
  - ▶ Agile Models

Helps to do various development activities in a systematic and disciplined manner.

Also makes it easier to manage a software development effort.

