ITT 309 MANAGEMENT FOR SOFTWARE ENGINEERS

- **Software & Software Engineering**

Software Engineering: A Practitioner's Approach, 7/e
by Roger S. Pressman

Software Engineering 9/e
By Ian Sommerville

Chapter 1

These slides are designed and adapted from slides provided by *Software Engineering: A Practitioner's Approach, 7/e* (McGraw-Hill 2009) by Roger Pressman and *Software Engineering 9/e* Addison Wesley 2011 by Ian Sommerville

What is Software?

The product that software professionals build and then support over the long term.

Software encompasses: (1) instructions (computer programs) that when executed provide desired features, function, and performance; (2) data structures that enable the programs to adequately store and manipulate information and (3) documentation that describes the operation and use of the programs.

Software products

- Generic products
  - Stand-alone systems that are marketed and sold to **any customer** who wishes to buy them.
  - Examples – PC software such as editing, graphics programs, project management tools; CAD software;
- Customized products
  - Software that is commissioned by **a specific customer** to meet their own needs.
  - Examples – embedded control systems, air traffic control software, traffic monitoring systems.

Software Engineering Definition

The seminal definition:

[Software engineering is] the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.

The IEEE definition:

Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1).

Essential attributes of good software

Product characteristic	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

Software Process

- A process is a collection of activities, actions and tasks that are performed when some work product is to be created

- Purpose of process is to deliver software in a timely manner and with sufficient quality to satisfy those who have sponsored its creation and those who will use it.

Five Activities of a Generic Process framework

- Communication: communicate with customer to understand objectives and gather requirements
- Planning: creates a "map" defines the work by describing the tasks, risks and resources, work products and work schedule.
- Modeling: Create a "sketch", what it looks like architecturally, how the constituent parts fit together and other characteristics.
- Construction: code generation and the testing.
- Deployment: Delivered to the customer who evaluates the products and provides feedback based on the evaluation.

- These five framework activities can be used to all software development regardless of the application domain, size of the project, complexity of the efforts etc, though the details will be different in each case.

- For many software projects, these framework activities are applied **iteratively** as a project progresses. Each iteration produces a software increment that provides a subset of overall software features and functionality.

Umbrella Activities

Complement the five process framework activities and help team manage and control progress, quality, change, and risk.

- Software project tracking and control: assess progress against the plan and take actions to maintain the schedule.
- Risk management: assesses risks that may affect the outcome and quality.
- Software quality assurance: defines and conduct activities to ensure quality.
- Technical reviews: assesses work products to uncover and remove errors before going to the next activity.
- Measurement: define and collects process, project, and product measures to ensure stakeholder's needs are met.
- Software configuration management: manage the effects of change throughout the software process.
- Reusability management: defines criteria for work product reuse and establishes mechanism to achieve reusable components.
- Work product preparation and production: create work products such as models, documents, logs, forms and lists.

Software engineering practice

The Essence of Practice

- How does the practice of software engineering fit in the process activities mentioned above? Namely, communication, planning, modeling, construction and deployment.
- George Polya outlines the essence of problem solving, suggests:

1. *Understand the problem* (communication and analysis).
2. *Plan a solution* (modeling and software design).
3. *Carry out the plan* (code generation).
4. *Examine the result for accuracy* (testing and quality assurance).

Understand the Problem

- *Who has a stake in the solution to the problem?* That is, who are the stakeholders?
- *What are the unknowns?* What data, functions, and features are required to properly solve the problem?
- *Can the problem be compartmentalized?* Is it possible to represent smaller problems that may be easier to understand?
- *Can the problem be represented graphically?* Can an analysis model be created?

Plan the Solution

- *Have you seen similar problems before?* Are there patterns that are recognizable in a potential solution? Is there existing software that implements the data, functions, and features that are required?
- *Has a similar problem been solved?* If so, are elements of the solution reusable?
- *Can subproblems be defined?* If so, are solutions readily apparent for the subproblems?
- *Can you represent a solution in a manner that leads to effective implementation?* Can a design model be created?

Carry Out the Plan

- *Does the solutions conform to the plan?* Is source code traceable to the design model?
- *Is each component part of the solution provably correct?* Has the design and code been reviewed, or better, have correctness proofs been applied to algorithm?

Examine the Result

- *Is it possible to test each component part of the solution?* Has a reasonable testing strategy been implemented?
- *Does the solution produce results that conform to the data, functions, and features that are required?* Has the software been validated against all stakeholder requirements?

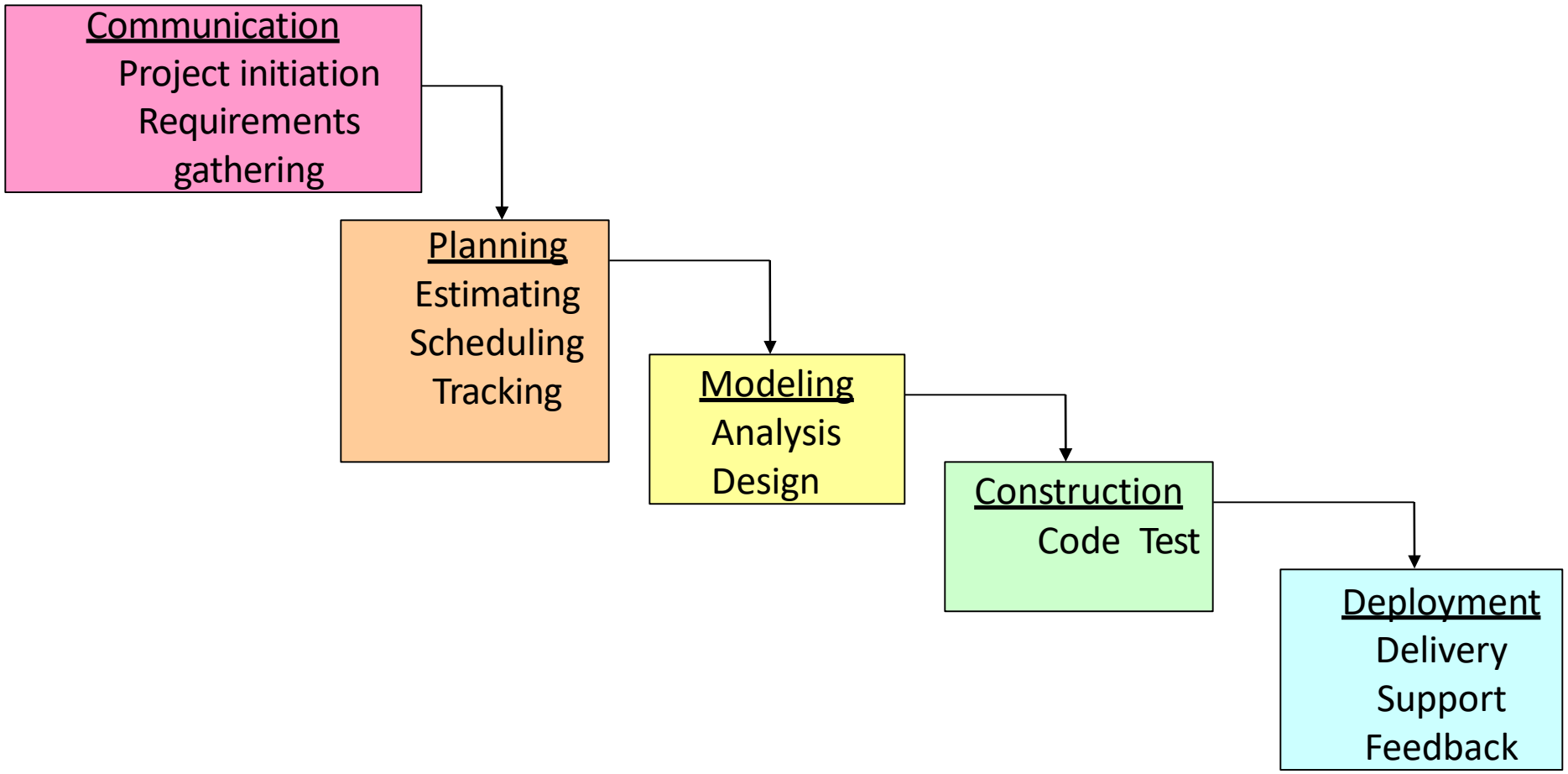
Prescriptive Process Models

- Prescriptive Process Models:
  - Waterfall/The Linear Sequential Model
  - The Incremental Model
  - The RAD Model
- Evolutionary Software Process Models:
  - The Prototyping Model
  - The Spiral Model

Prescriptive Process Model

- Defines a distinct set of activities, actions, tasks, milestones, and work products that are required to engineer high-quality software
- The activities may be linear, incremental, or evolutionary

Waterfall Model (Diagram)



Waterfall Model (Description)

- Oldest software lifecycle model .
- Used when requirements are well understood and risk is low
- Work flow is in a linear (i.e., sequential) fashion
- Used often with well-defined adaptations or enhancements to current software

Waterfall Model (Problems)

- Doesn't support iteration, so changes can cause confusion
- Difficult for customers to state all requirements explicitly and up front
- Requires customer patience because a working version of the program doesn't occur until the final phase.

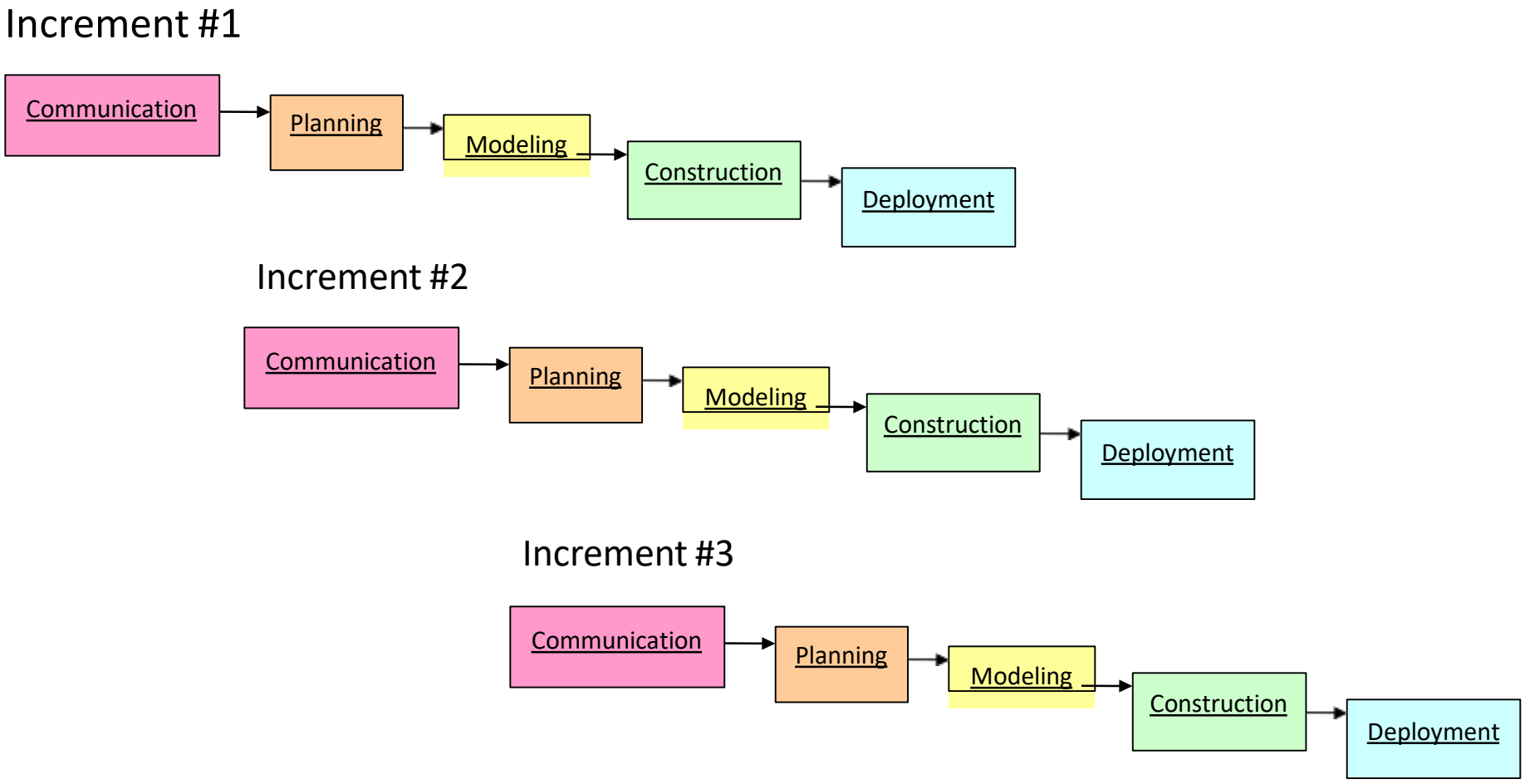
Incremental Model

Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle.

In this model, each module goes through the requirements, design, implementation and testing phases.

Every subsequent release of the module adds function to the previous release. The process continues until the complete system achieved.

Incremental Model (Diagram)



Incremental Model (Description)

- Used when requirements are well understood
- Multiple independent deliveries are identified
- Work flow is in a linear (i.e., sequential) fashion within an increment a
- Iterative in nature; focuses on an operational product with each increment
- Useful also when staffing is too short for a full-scale development

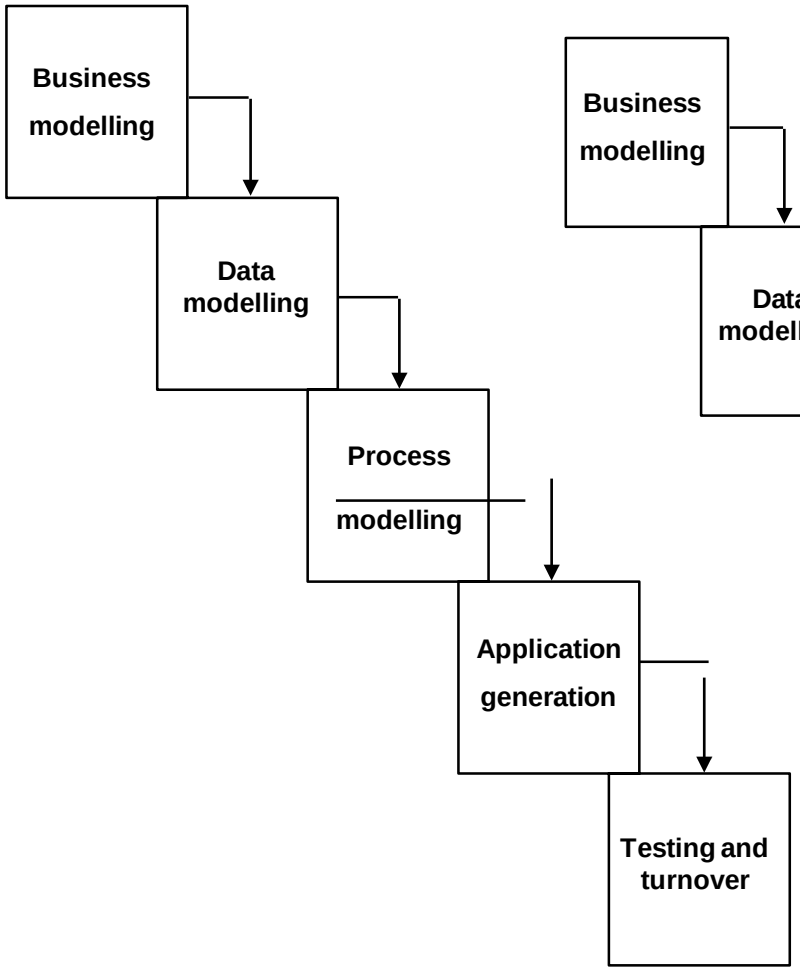
Rapid Application Development Model

- Similar to waterfall but uses a very short development cycle (60 to 90 days to completion)
- Uses component-based construction and emphasises reuse and code generation
- Use multiple teams on scaleable projects
- Requires heavy resources
- Requires developers and customers who are heavily committed

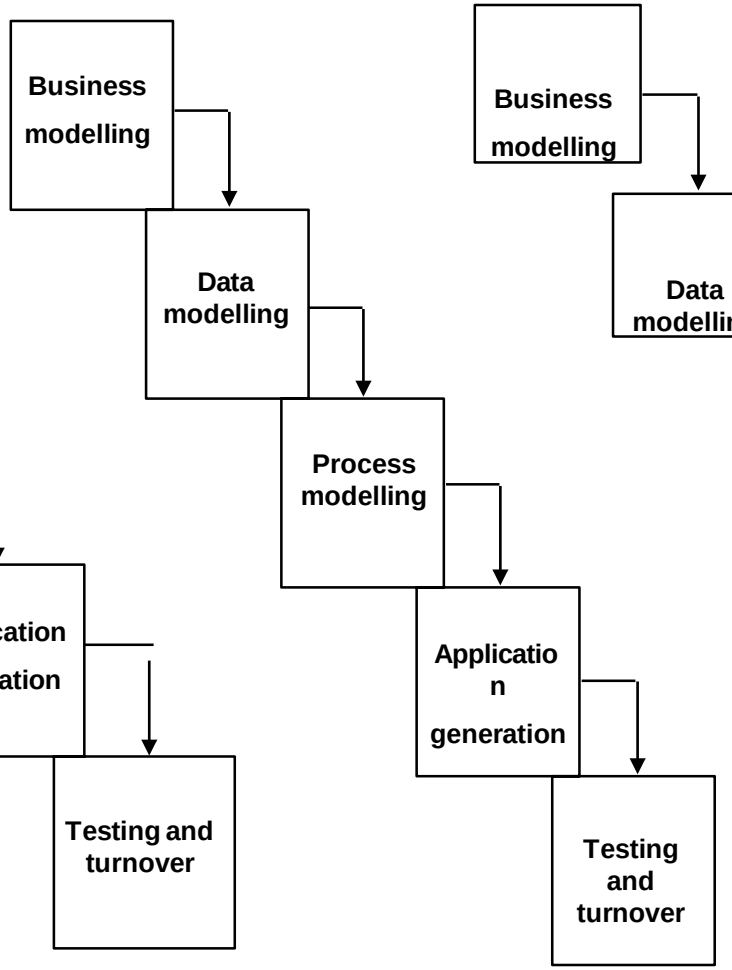
- Difficult to use with new technology

Rapid Application Development

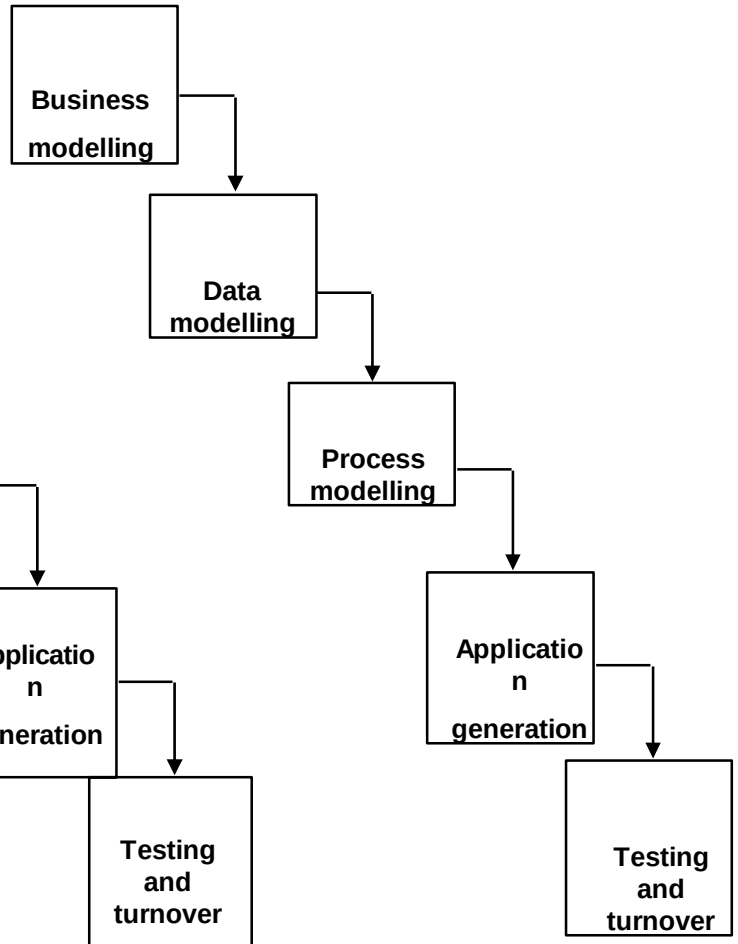
- **Team 1**



Team 2



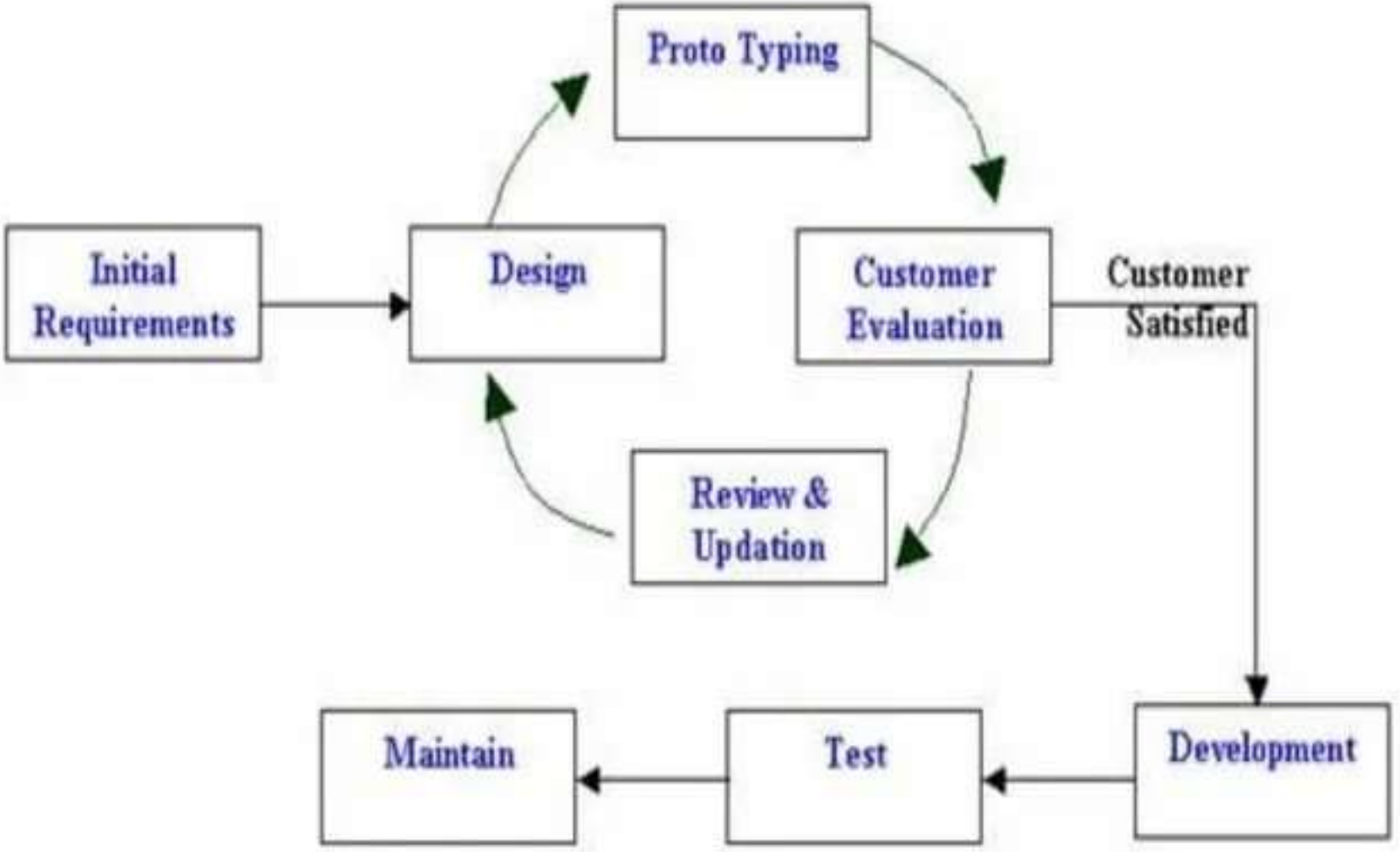
Team 3



Evolutionary Software Process Models

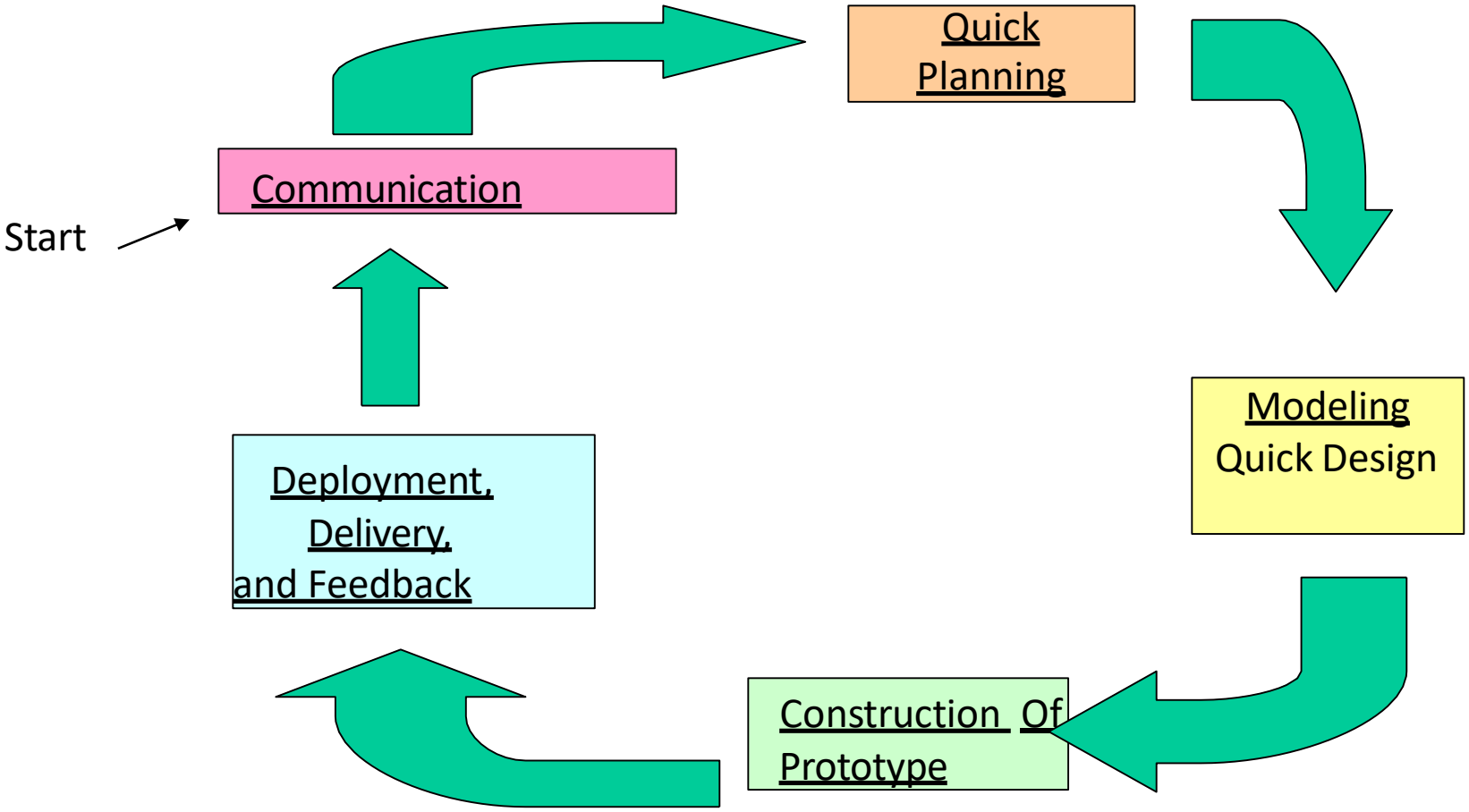
- The Prototyping Model
- The Spiral Model

Prototyping Model



Proto Type Model

Prototyping Model (Diagram)



Prototyping Model

- Follows an evolutionary and iterative approach
- Used when requirements are not well understood
- Serves as a mechanism for identifying software requirements
- Focuses on those aspects of the software that are visible to the customer/user
- Feedback is used to refine the prototype

Prototyping Model

Advantages of the Prototyping Model

Users are actively involved in development. Therefore, errors can be detected in the initial stage of the software development process.

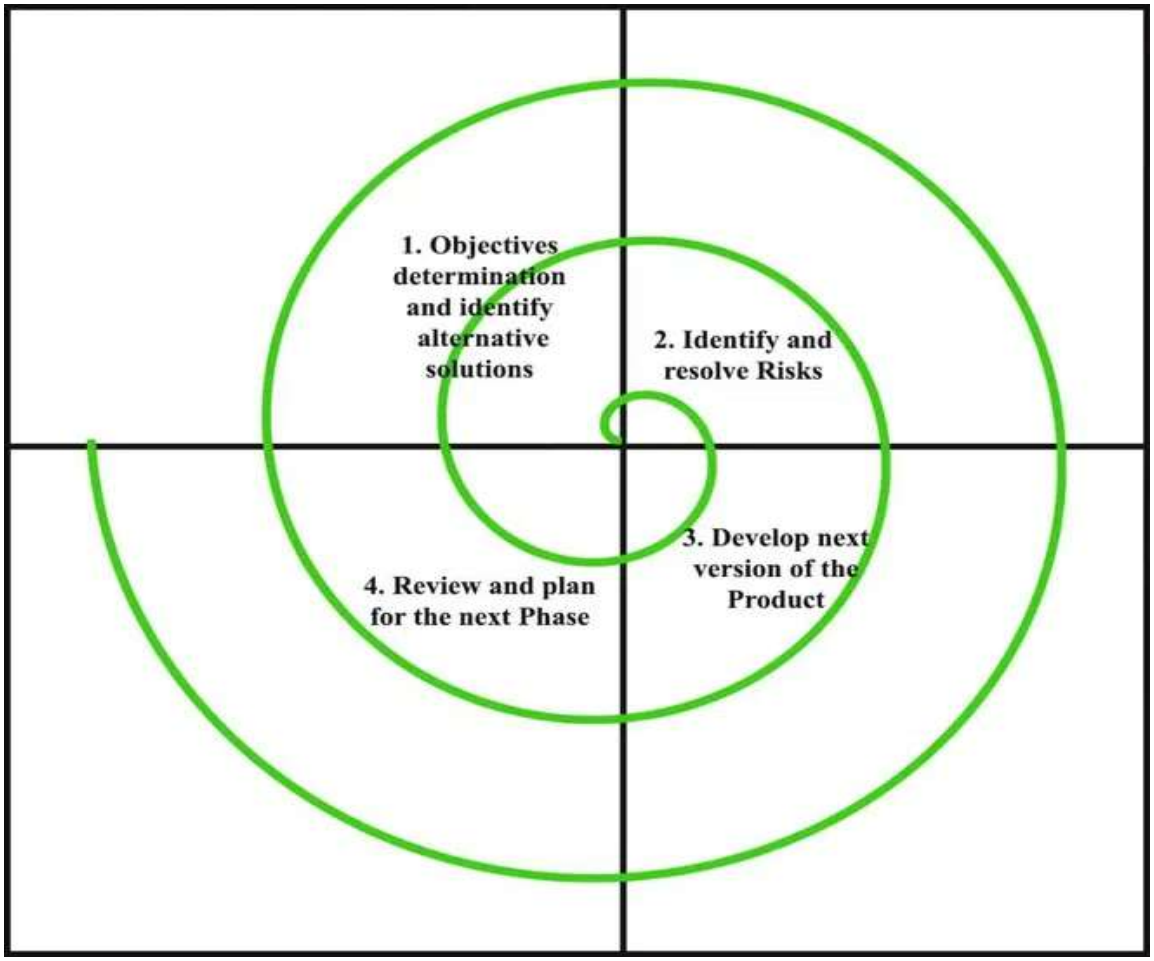
Missing functionality can be identified, which helps to reduce the risk of failure.

Prototyping is also considered as a risk reduction activity.

Helps team member to communicate effectively

Customer satisfaction exists because the customer can feel the product at a very early stage.

Spiral Model



Spiral Model

The Spiral Model is a risk-driven model, meaning that the focus is on managing risk through multiple iterations of the software development process.

The spiral model is used by software engineers and is favored for large, expensive and complicated projects.

When viewed as a diagram, the spiral model looks like a coil with many loops. The number of loops varies based on each project and is often designated by the project manager. Each loop of the spiral is a phase in the software development process

ADVANTAGES & DISADVANTAGES

- Advantages
  - Explicit consideration of risks (alternative solutions are evaluated in each cycle).
  - More detailed processes for each development phase.
- Disadvantages
  - Cost is high.
  - Sometime difficult to implement or too time consuming.

Specialized Process model

Special process models take many features from one or more conventional models.

Types in Specialized process models:

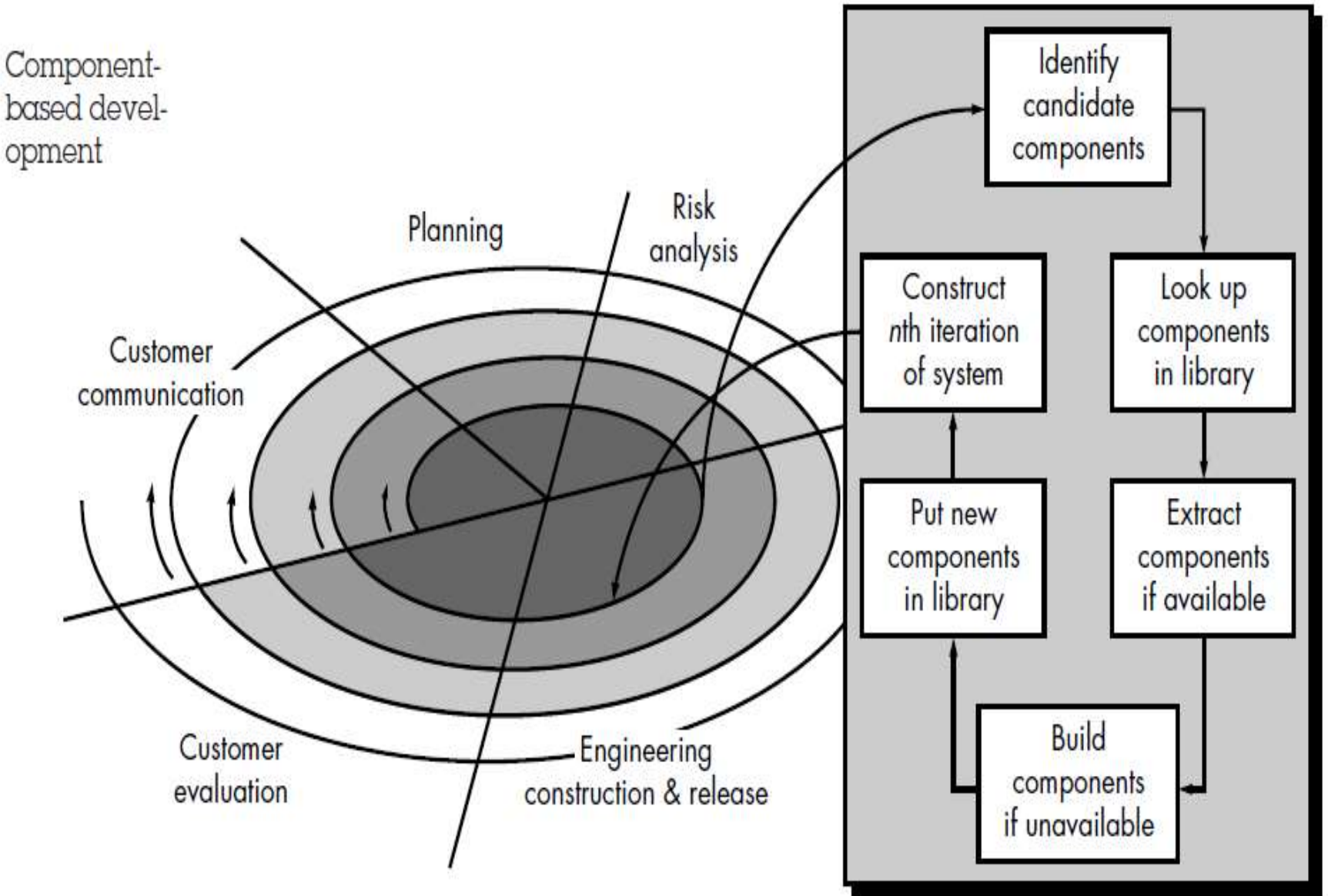
1. Component based development (Promotes reusable components)
2. The formal methods model (Mathematical formal methods are backbone here)
3. Aspect oriented software development (Uses crosscutting technology)

Component based development

The component-based assembly model uses object-oriented technologies. In object-oriented technologies, the emphasis is on the creation of classes

The component based development model incorporates many of the characteristics of the spiral model. It is evolutionary in nature, demanding an iterative approach to the creation of software.

However, the model focuses on prepackaged software components. It promotes software reusability.


Component-based devel-opment
Planning
Risk analysis
Customer communication
Customer evaluation
Engineering construction & release
Identify candidate components
Construct nth iteration of system
Look up components in library
Put new components in library
Extract components if available
Build components if unavailable

Component based development

Modeling and construction activities begin with the identification of candidate components. Candidate components can be designed as either conventional software modules or object oriented packages.

Component based development has the following steps:
1. Available component based products are researched and evaluated for the application domain.
2. Component integration issues are considered.
3. A software architecture is designed to accommodate

Problems in Component based development

The biggest difficulty a developer has to face when it comes to CBSD is identifying the proper components that can cover all the requirements of the product

The formal methods model

The formal methods model encompasses a set of activities that leads to formal
mathematical specification of software.

Formal methods (Formal methods are system design techniques that use rigorously specified mathematical models to build software and hardware systems)enable a software engineer to specify, develop and verify a computer system by applying rigorous mathematical notation.

When mathematical methods are used during software development, they
provide a mechanism for eliminating many of the problems

Problems in formal methods model

The development of formal models is currently time-consuming and expensive.

Because few developers have the necessary background knowledge to apply
formal methods, extensive training is required to others.

It is difficult to use this model as a communication mechanism for technically
unsophisticated people.

Aspect oriented Software Development

A spect Oriented Software Development (AOSD) often referred to as aspect-
oriented programming (AOP), a relatively new paradigm that provides process
and methodology for defining, specifying designing and constructing aspects.

Creating any software or application,even it has many module, which may have some common code (ie security,profiling,transaction management etc..)

Hence these common codes are written in a class.These classes are called aspects

Aspect oriented Software Development

Advantages

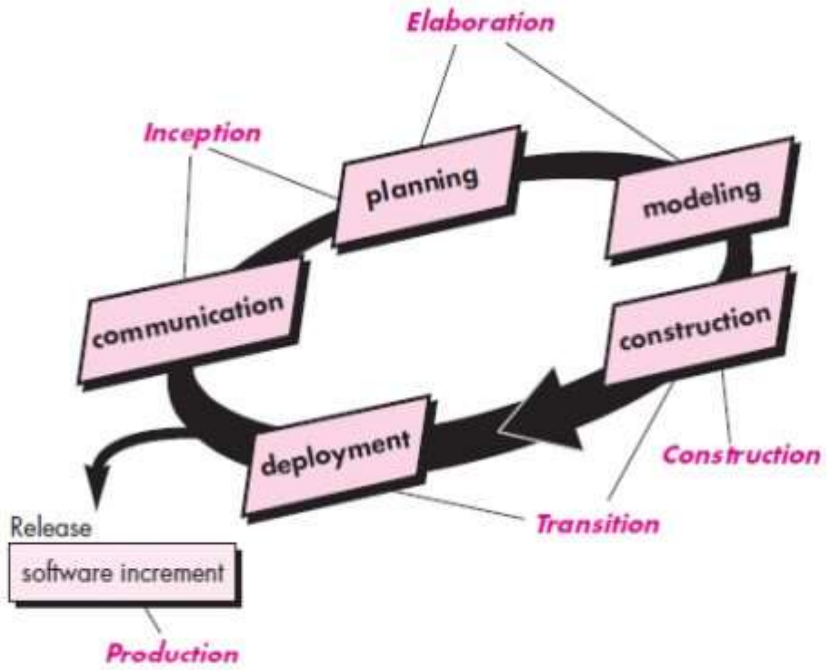
Increase modularity and reusability
Increse maintainability

Disadvantages

Security risk

Unified Process Model

Unified Process is based on the enlargement and refinement of a system through multiple iterations, with cyclic feedback and adaptation. The system is developed incrementally over time, iteration by iteration, and thus this approach is also known as iterative and incremental software development.



Unified Process Model

The key characteristics of the Unified Process are:

It is an iterative and incremental development framework

It is architecture-centric -The description of architecture is the heart and central concern of the process.

It is risk-focused and emphasizes that highest-risk factors be addressed in the earliest stage

It is use-case and UML model driven with nearly all requirements being documented in one of those forms

Inception

Communication and planning are main.
Identifies Scope of the project using use-case model allowing managers to estimate costs and time required.
Customers requirements are identified and then it becomes easy to make a plan of the project.
Project plan, Project goal, risks, use-case model, Project description, are made.
Project is checked against the milestone criteria and if it couldn’t pass these criteria then project can be either cancelled or redesigned.

Elaboration –

Planning and modeling are main.
Detailed evaluation, development plan is carried out and diminish the risks.
Revise or redefine use-case model
Again, checked against milestone criteria and if it couldn’t pass these criteria then again project can be cancelled or redesigned.

Unified Process Mo

Construction –

Project is developed and completed.
System or source code is created and then testing is done.
Coding takes place.

Transition –

Final project is released to public.
Transit the project from development into production.
Update project documentation.
Defects are removed from project based on feedback from public.

Production –

Final phase of the model.
Project is maintained and updated accordingly.

Unified Process Model