SOFTWARE ENGINEERING (15B11CI513)

Credits :- 4 **Contact Hours :-** 3-1-0

Lecture 2: Software Process Model

Software Process

(IEEE) *A sequence of steps* (activities, methods, practices, and transformations) *performed for a given purpose* (develop and maintain software)

Steps:

- Specification
- Design and Implementation
- Validation
- Evolution

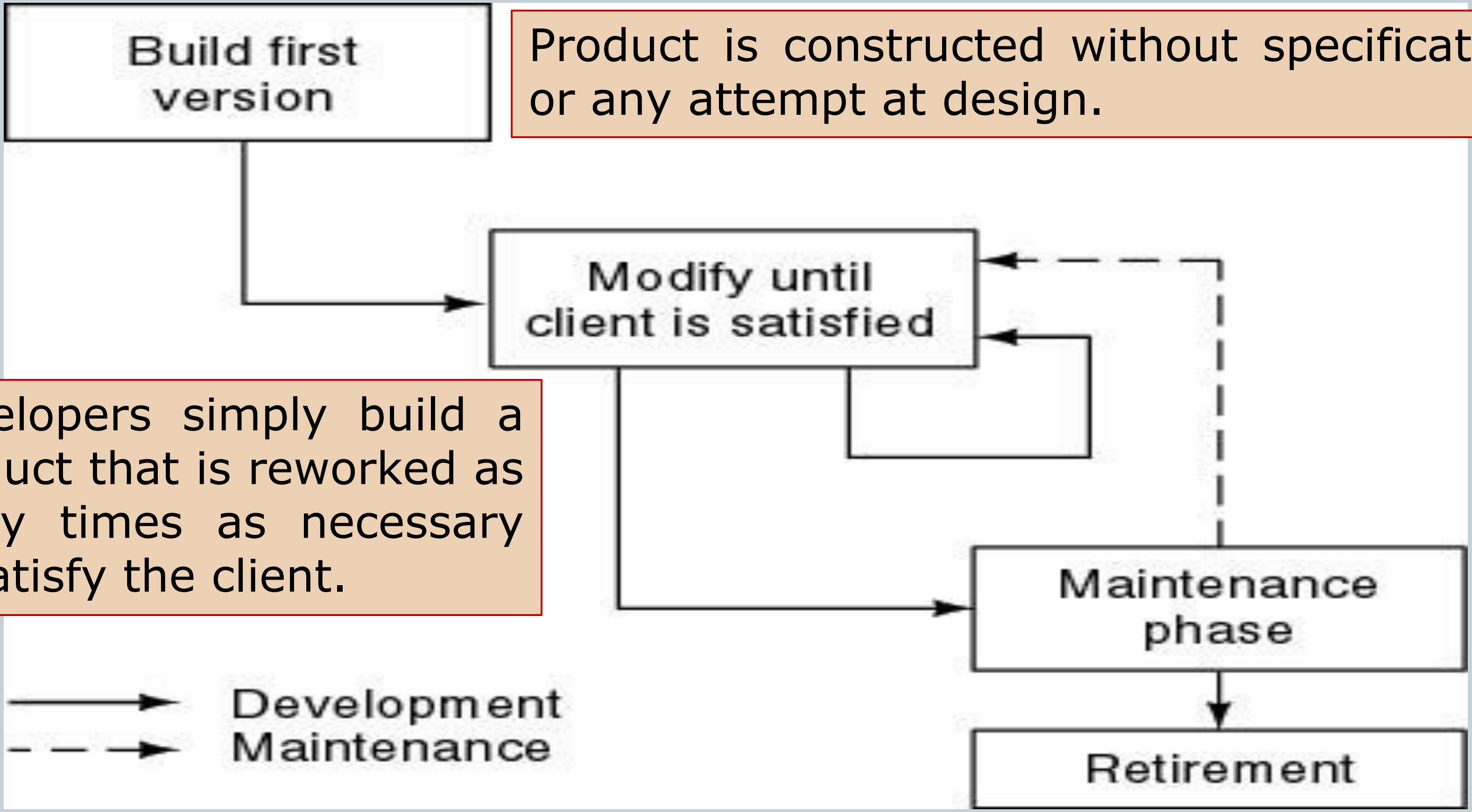
❖ A “software process model” is an abstract representation of a process.

❖ It presents a description of a process from some particular perspective

Software Process Model

- A Model has different Processes in it.
- It is followed to produce high quality software.
- It provides stability, control, and organization to a process.
- They are adapted to meet the needs of software engineers and managers for a specific project.

Build and Fix Model



Product is constructed without specification or any attempt at design.

Developers simply build a product that is reworked as many times as necessary to satisfy the client.

Build and Fix Model

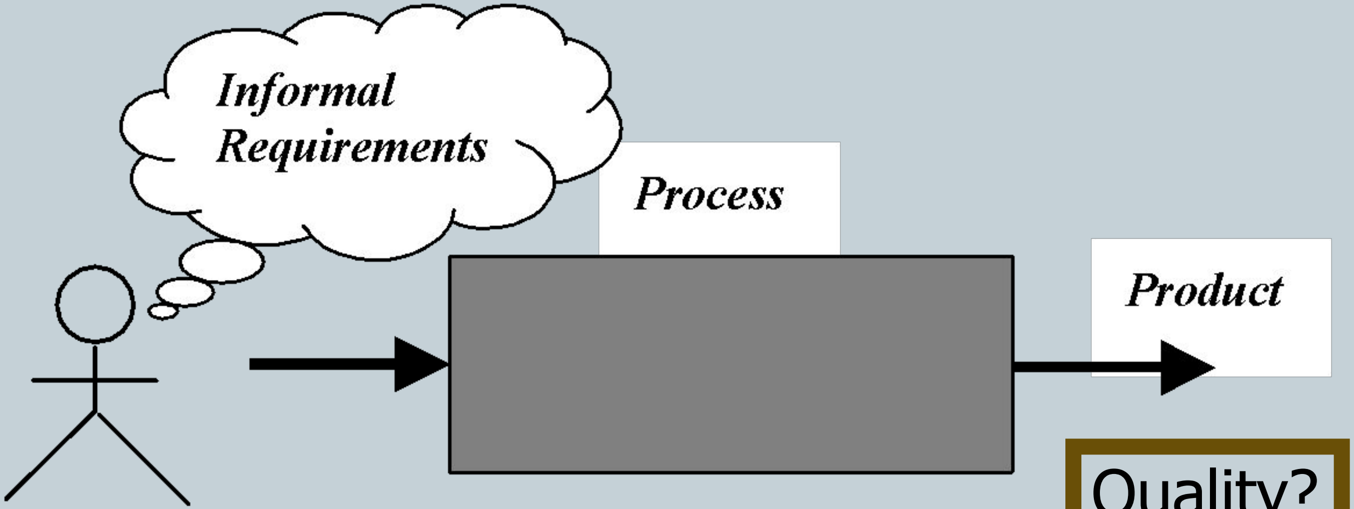
The earlier approach

- Product is constructed without specification or any attempt at design.
- developers simply build a product that is reworked as many times as necessary to satisfy the client.
- model may work for small projects but is totally unsatisfactory for products of any reasonable size.
- Maintenance is high.
- Source of difficulties and deficiencies
  - impossible to predict
  - impossible to manage

Why Models are needed?

- Symptoms of inadequacy: the software crisis
  - scheduled time and cost exceeded
  - user expectations not met
  - poor quality

Process as a "black box"



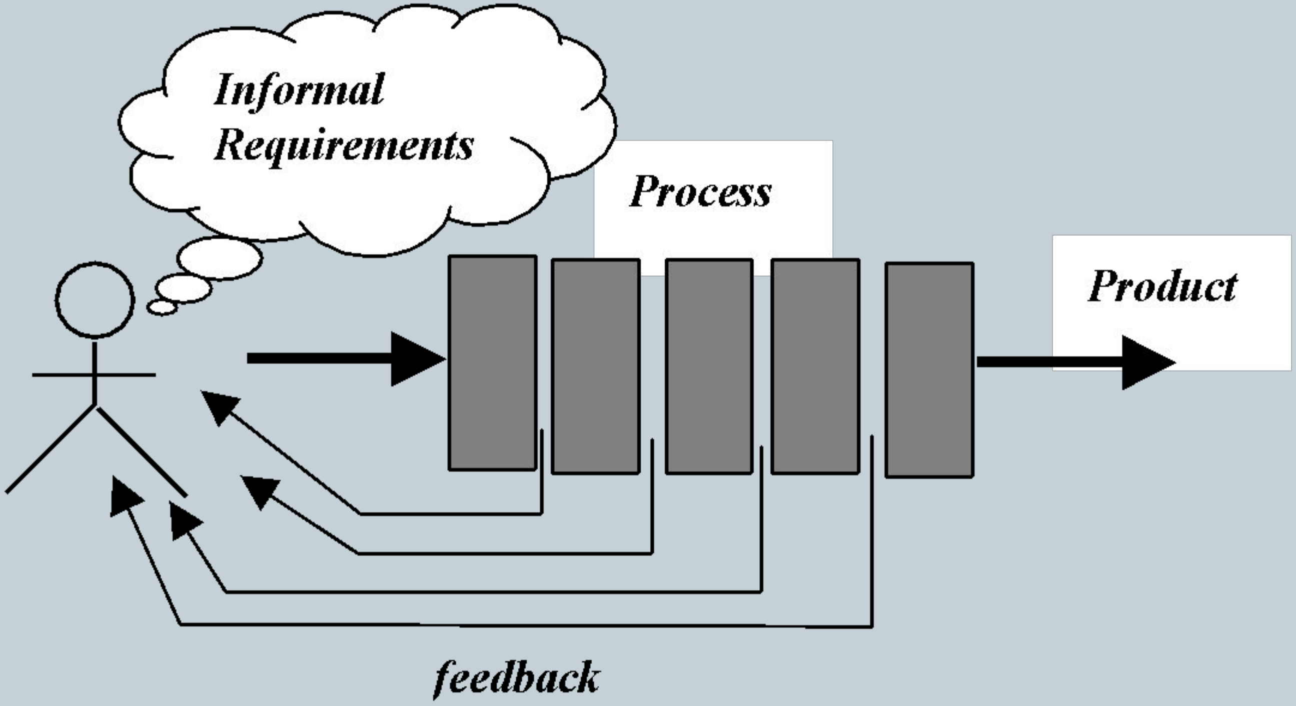
Uncertain /
Incomplete requirement
In the beginning

Quality?

Problems

- The assumption is that requirements can be fully understood prior to development
- Interaction with the customer occurs only at the beginning (requirements) and end (after delivery)
- Unfortunately the assumption almost never holds

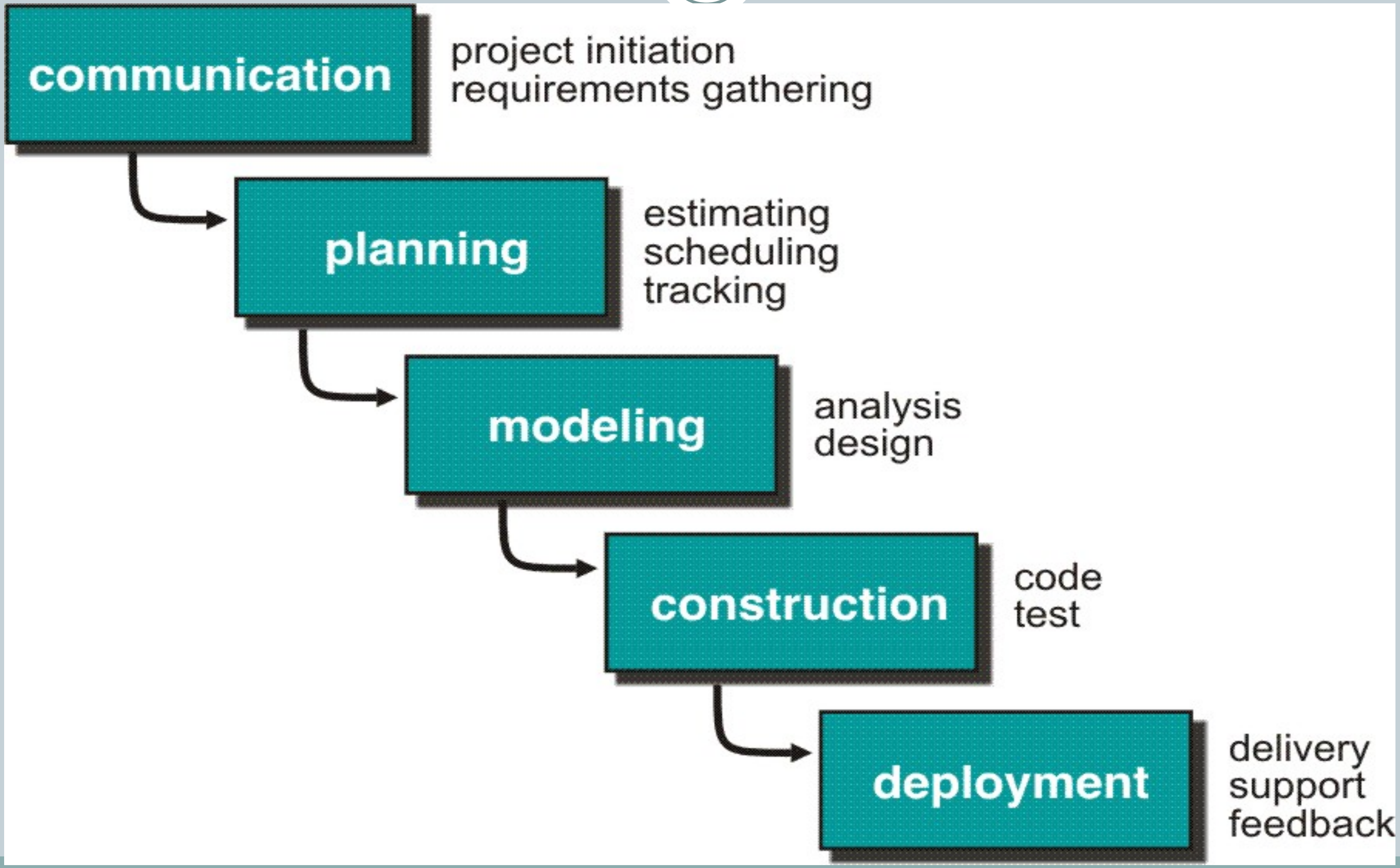
Process as a "white box"



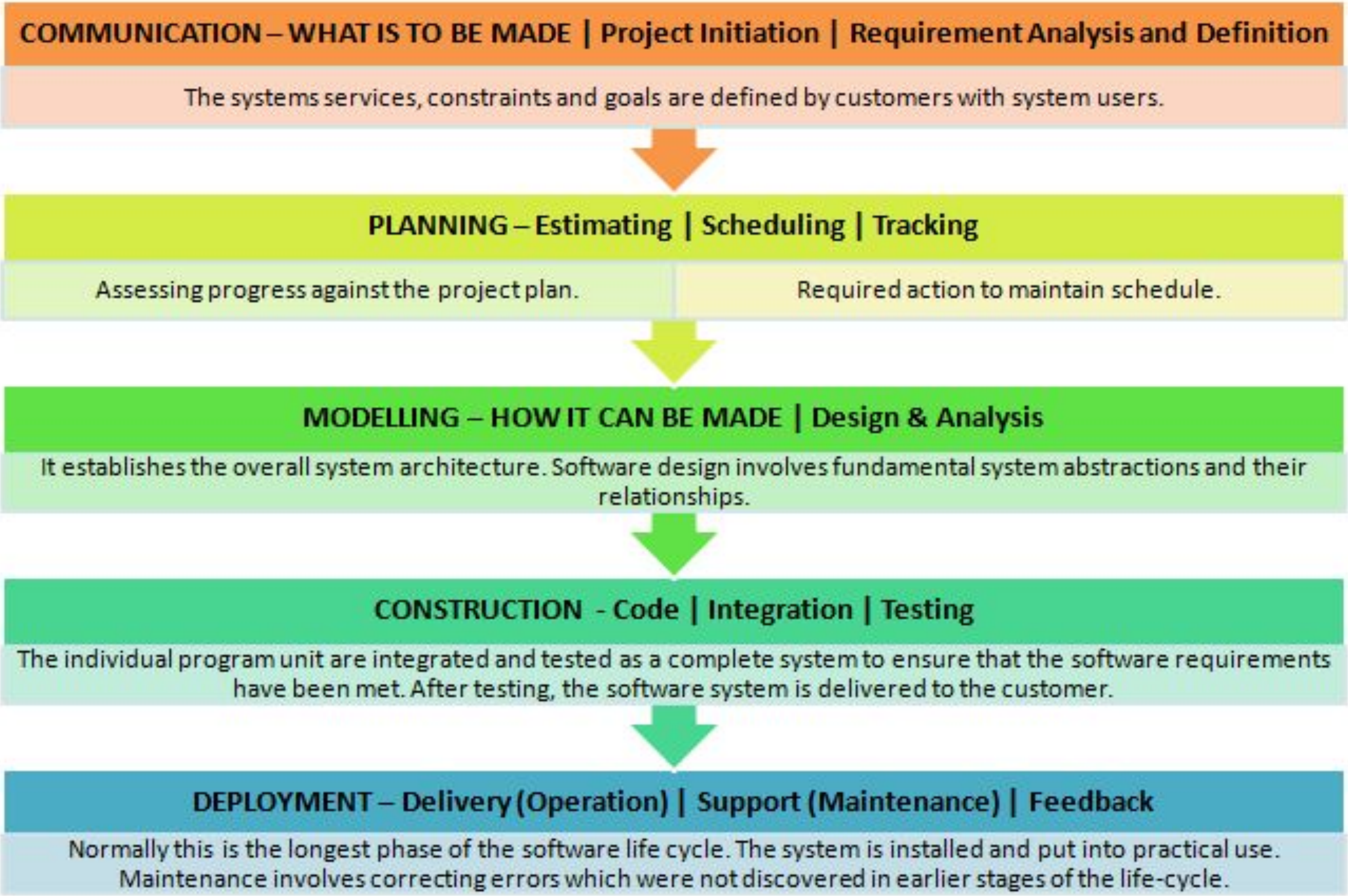
Advantages

- Reduce risks by improving visibility
- Allow project changes as the project progresses
  - based on feedback from the customer

Waterfall Model or Classic Life Cycle



Waterfall Model



COMMUNICATION – WHAT IS TO BE MADE | Project Initiation | Requirement Analysis and Definition

The systems services, constraints and goals are defined by customers with system users.

PLANNING – Estimating | Scheduling | Tracking

Assessing progress against the project plan.

Required action to maintain schedule.

MODELLING – HOW IT CAN BE MADE | Design & Analysis

It establishes the overall system architecture. Software design involves fundamental system abstractions and their relationships.

CONSTRUCTION - Code | Integration | Testing

The individual program unit are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered to the customer.

DEPLOYMENT – Delivery (Operation) | Support (Maintenance) | Feedback

Normally this is the longest phase of the software life cycle. The system is installed and put into practical use. Maintenance involves correcting errors which were not discovered in earlier stages of the life-cycle.

Waterfall Model- **Limitations**

- The nature of the requirements could not change during development.
- The model implies that you should attempt to complete a given stage before moving on to the next stage
- Does not account for the fact that requirements constantly change.
- Customers can not use anything until the entire system is complete.
- The model implies that once the product is finished, everything else is maintenance.
- Surprises/changes at the end are very expensive.
- Some teams sit ideal for other teams to finish

- *Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.*

Waterfall Model- **Limitations**

Problems:

- Real projects are rarely follow the sequential model.
- Difficult for the customer to state all the requirement explicitly.
- Assumes patience from customer - working version of program will not available until programs not getting change fully.