

# Advanced Software Engineering

---

## Lecture 1: Introduction to Software Engineering

Prof. Harold Liu

# Content

---

- ❑ **Basic concept of software engineering**
- ❑ Computer-based systems engineering
- ❑ Software Process
- ❑ Basic elements of software project management

# 1. What is Software?

---

- Software=program+data+document
  - Customized software
  - Generic software, Shrink-wrapped software
  - Embedded software
  - Safety-critical software
  - COTS (Commercial Off-the-Shelf)
- “I will create a software to update the database”.  
(some software, a piece of software, a software system)
- Categorized by its functionalities, scale/size, operational method, required reliability, etc.

## 2. What is Software Engineering?

---

- ❑ Fall, 1968, NATO Technical Committee convened nearly 50 first-class programmers, computer scientists and industry giants, discuss how to cope with "software crisis". Fritz Bauer at the meeting for the first time put forward the "Software Engineering" concept.
- ❑ The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines." --- Fritz Bauer, 1968

- 
- ❑ IEEE Definition: Software engineering is
    - (1) the systematic, standardized, measurable method is applied to software development, operation and maintenance of the process, the upcoming engineering applied to software development process thinking
    - (2) research into (1)
  - ❑ Software engineering goals: low-cost, high-quality, on-time delivery

# Essential Characteristics of Software Engineering

---

- ❑ Concerned about the construction of large programs
- ❑ How to control the complexity
- ❑ Constantly changing software requirements
- ❑ Aimed at improving the efficiency of software development
- ❑ Teamwork is the key to the successful implementation of software engineering
- ❑ Software must effectively support its users
- ❑ Cultural background difference for software product creation.

# 5. What is Software Process?

---

- ❑ Software process is the development of software products for a group of activities and their results.
- ❑ All software process contains four basic activities: Software description, software development, software validation and software evolution.
- ❑ Different software processes organized in different ways in these four activities, which may affect the results of the progress of events.
- ❑ Different bodies may be used to produce the same type of process products.

# Benefits

---


- ❑ A software process defines that in order to achieve the goals, what people need in what way at what time what kind of work done (Goal, Who, When, How, What)
- ❑ For Customer, User, Developer, Manager, a widely applicable software process allows all stakeholders to better understand their role and others' role, as what to do at what time.
- ❑ Promote the integration process to obtain the "best software process."
- ❑ Make the company's internal training standardized.
- ❑ Due to the repeatability of the software process, useful for the development schedule arrangements and cost estimates.



## 6. What is Software Process Model?

---

- ❑ Models and Modeling
- ❑ Software process model is built from a particular perspective on the nature of the software process description.
- ❑ Software process model includes a variety of activities that are constituting the software process, software products, as well as all stakeholders

- 
- 
- ❑ From a different perspective of the software process described, you get a different type of process models. They are:
  - ❑ Workflow model: Describes the sequence of a series of activities, inputs, outputs and inter-dependencie. Here we refer “Activities” as human activity.
  - ❑ Data stream or activity model: the software process described as a set of activities, each of which completes certain data conversion. The referred level of activity in this model is below the workflow model.
  - ❑ Role/action model: Describes the different roles involved in the software process and its responsible activities.

# Paradigm, Methodology

---

- ❑ Waterfall Model
- ❑ Waterfall Model with Maintenance Circle
- ❑ Waterfall Model with Prototyping
- ❑ Spiral Model
- ❑ V Model
- ❑ Phased Development Model
- ❑ Incremental and Iterative Model
- ❑ Rational Unified Process

# 7. Costs and distributions

---

- ❑ Software costs depend on the adopted software process and the type of software being developed.
- ❑ General distribution of costs:  
Description: Design: Development: Integration and Test (15:25:20:40)%
- ❑ High-demand software systems: integration/testing 50:50%
- ❑ From software evolution
- ❑ From maintenance
- ❑ Web-based e-commerce system

# 8. Methodologies

---

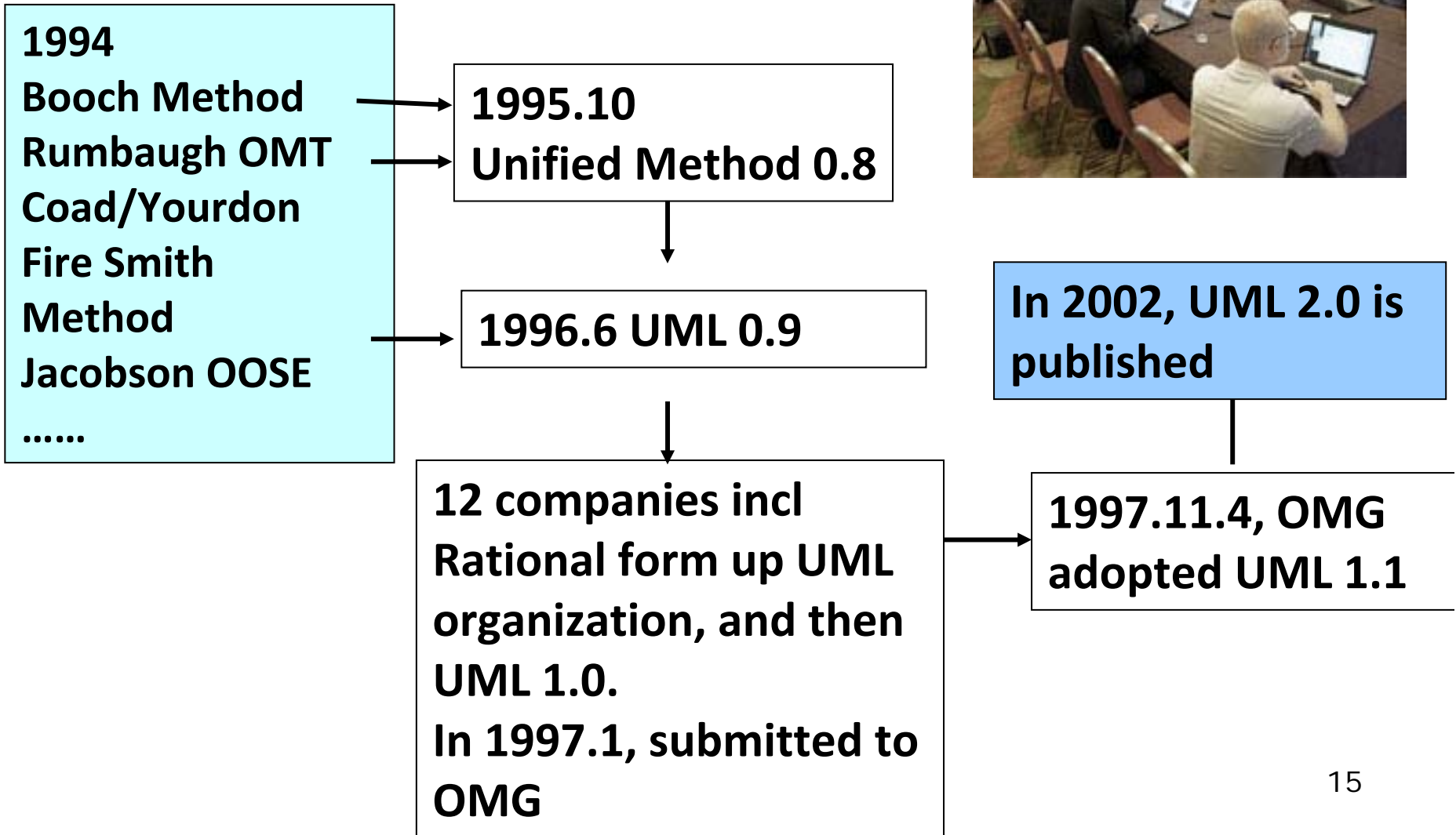
- ❑ Structured Method, by DeMarco, 1978, Yourdon E. and Constantine L. et. al.
- ❑ Jackson System Development, 1983
- ❑ Object-Oriented Method, by Booch, 1994; Rumbaugh et al, 1991 → Design Patterns

# UML (Unified Modeling Language)

---

- In 1994, OO principle has been throughout the entire software life cycle, it already has an impact on 50+ OOA&D methods.
- UML is a kind of modeling language for visualization, decription, structuring and documentation, mainly for the analysis and design phase of system modeling.

# UML 2.0



# RUP

---

- By IBM Rational



- Highlights

- Use-case driven (functionality)
- architecture-centric (design)
- Iterative and incremental developments (implementation)
- Using UML and other tools



# Tendency for Change when Using OO Paradigm (Jacobson et al. 1995)

---

Characteristic of SW product/project	Probability for change
Objects derived from the application	Low
Long-lived information structures	Low
Passive object's attribute	Medium
Sequences of behavior	Medium
<b>Interface with the outside world</b>	<b>High</b>
<b>Functionality</b>	<b>High</b>

# 9. What is CASE?

---

- ❑ Computer-Aided Software Engineering (CASE)
- ❑ A bunch of tools to support the software engineering process
- ❑ Lower-end tools: to generate codes
- ❑ High-end tools  
to draw enterprise model and application requirements specifications

# 10. Problems of Software Engineering

---

- ❑ Legacy systems
- ❑ Changing requirements
- ❑ Delivery
- ❑ ...
  
- ❑ System complexity and details (Mars probe failure)
- ❑ Technological uncertainty (and in turn the understanding of developers)
- ❑ Communication barriers
- ❑ Requirement uncertainty
- ❑ Constant changes lead to the software degradation
- ❑ Artificial and the market risk
- ❑ Costs, reliability, productivity, reusability problems

# 11. Moral responsibility

---

- Professional ethics is very important
  - Always keep employer and clients' information confidential
  - Realistically express his/her ability to work, do not accept work beyond the capability.
  - Should be aware of the laws related to patents, copyrights, etc.
  - Avoid computer misuse.
- ACM/IEEE/BCS and other organizations issued a Code of Professional Conduct and Ethics (involving only basic moral behavior)

# Content

---

- ❑ Basic Concept of Software Engineering
- ❑ **Computer-based System Engineering**
- ❑ Software Process
- ❑ Basic elements of software project management

# 1. System Overall Characteristics

---

- Functional
- Non-functional: reliability, security, privacy, safety issues
- What relates the system reliability:
  - HW
  - SW
  - Operational

## 2. System and Its Surviving Environment

---

- ❑ Environment is a must-have
- ❑ Environment will affect the system's functionality and performance.
- ❑ In the environment there exist a series of interactions with other systems. Thus, the environment can be treated as a stand-alone system.
- ❑ Systems engineer must understand the system environment
  - In many cases, the system is to change the environment.
  - A system can be affected by environmental changes, and such effects may be difficult to estimate.

# 3. System Modeling

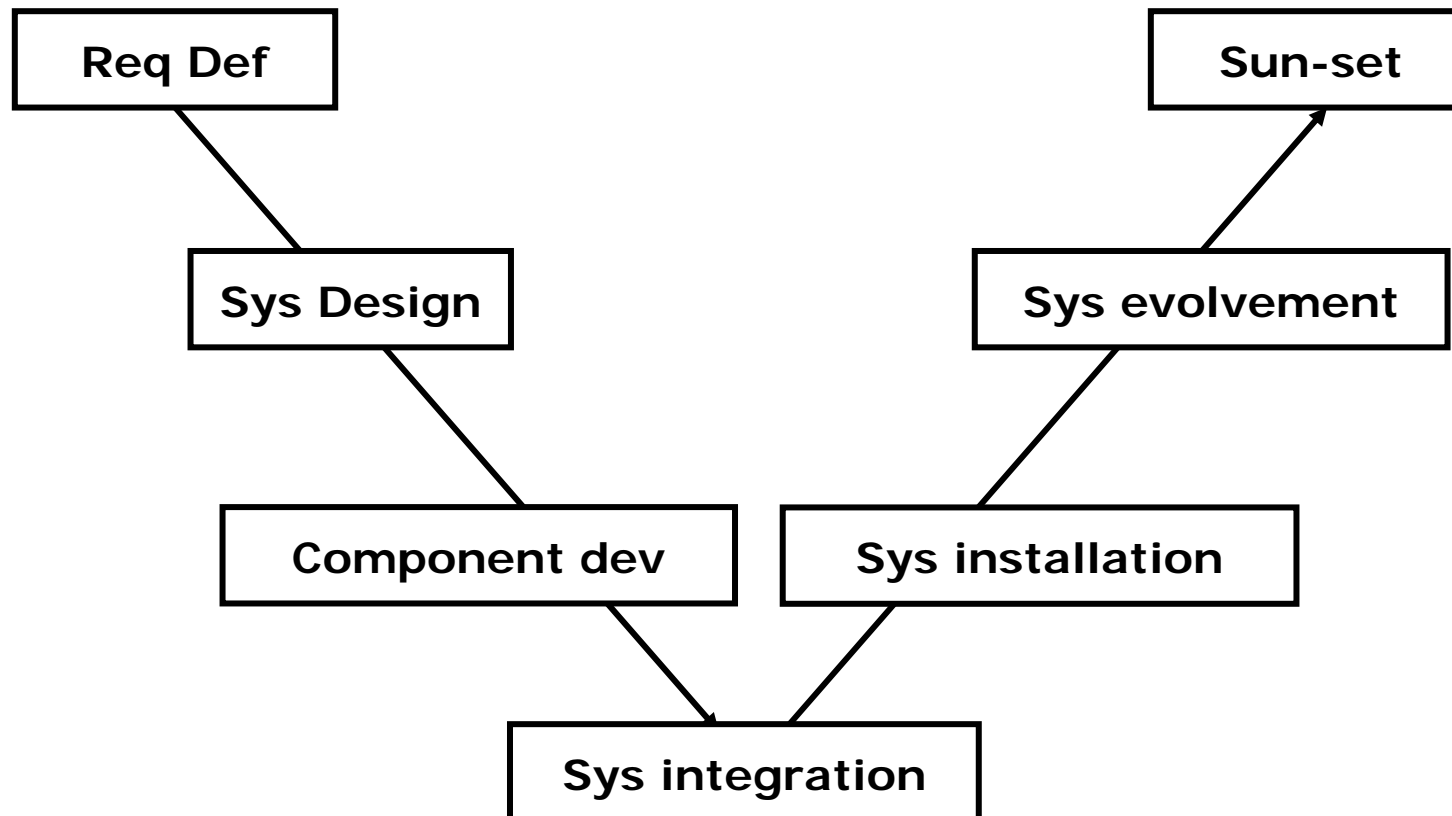
---

- ❑ System modeling describes the components of the system and their relationships (system architecture)
- ❑ System architecture is generally described in block diagram.
- ❑ Subsystems
- ❑ Functional components can be divided into six categories:
  - Sensors: Gather information from the environment.
  - Actuators: incur the environmental changes
  - Computing components: given an input, perform calculations and produce output.
  - Communications
  - Scheduling: coordination of operations between components.
  - Interface



# 4. Process for Sys Engineering

---

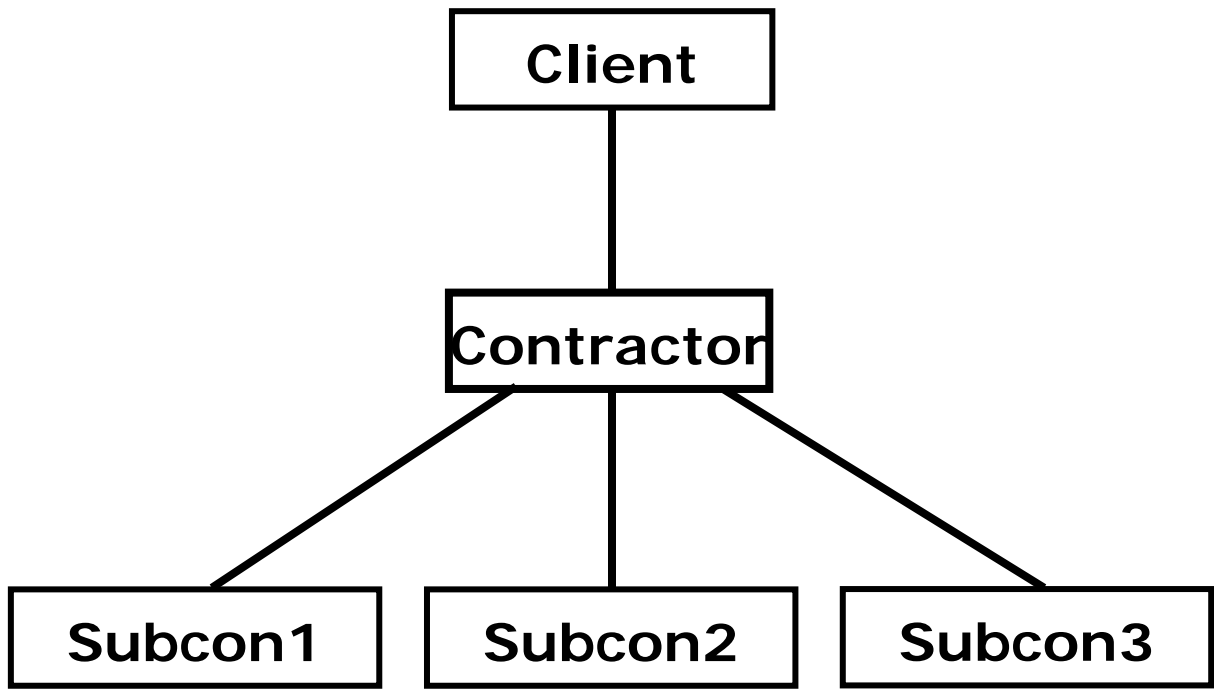


- 
- Subsystem development:
    - Parallel
    - Frequent modifications beyond the scope of a subsystem may happen (HW? SW?)
    - System integration by incremental process
      - Hard to estimate the exact timing for a subsystem
      - Accurate error-positioning (which subsystem causes the failure?) to reduce the disagreement of different subcontractors

# 5. Buying Softwares

---

- ❑ What is the best way of purchase? Who is the best provider?
- ❑ Before purchasing, system-level descriptions and architectural design must be done, to make sure which “subsystem” needs to be purchased.
- ❑ Who is the contractor, and who is the sub-contractor.



# Content

---

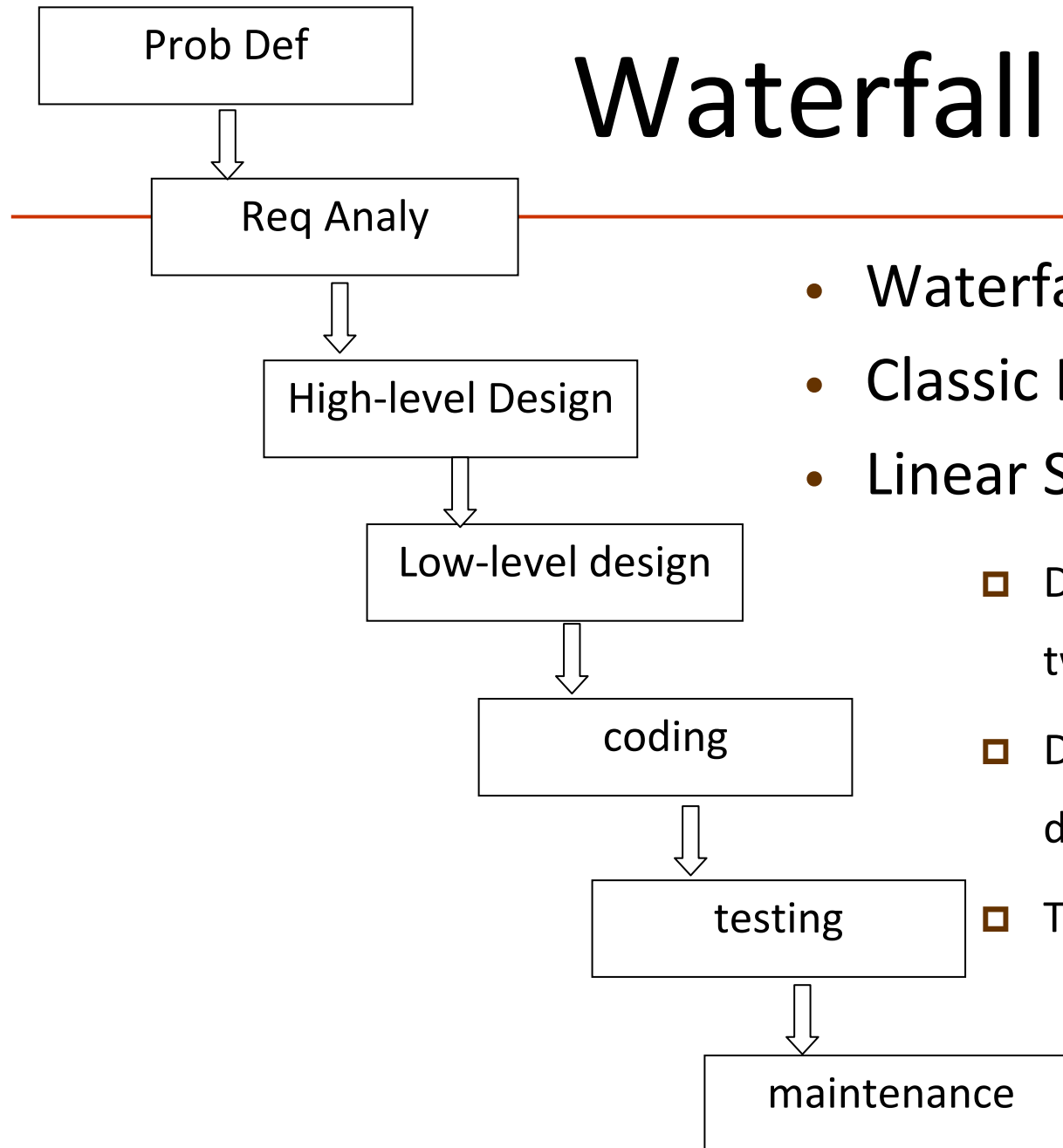
- ❑ Basic Concept of Software Engineering
- ❑ Computer-based System Engineering
- ❑ **Software Process**
- ❑ Basic elements of software project management

# 1. Software Process Model

---

- ❑ An abstract representation of the software process
- ❑ In practice, multiple models are used.
- ❑ Four kinds of models
  - Life-cycle model (waterfall)
  - Evolutionary development model
  - Formal methods
  - Reuse-oriented development model (OO)

# Waterfall



- Waterfall
- Classic Life Cycle Model
- Linear Sequential Model
  - ▣ Dependencies between any two subsequent stages
  - ▣ Defer development after designs
  - ▣ To ensure quality

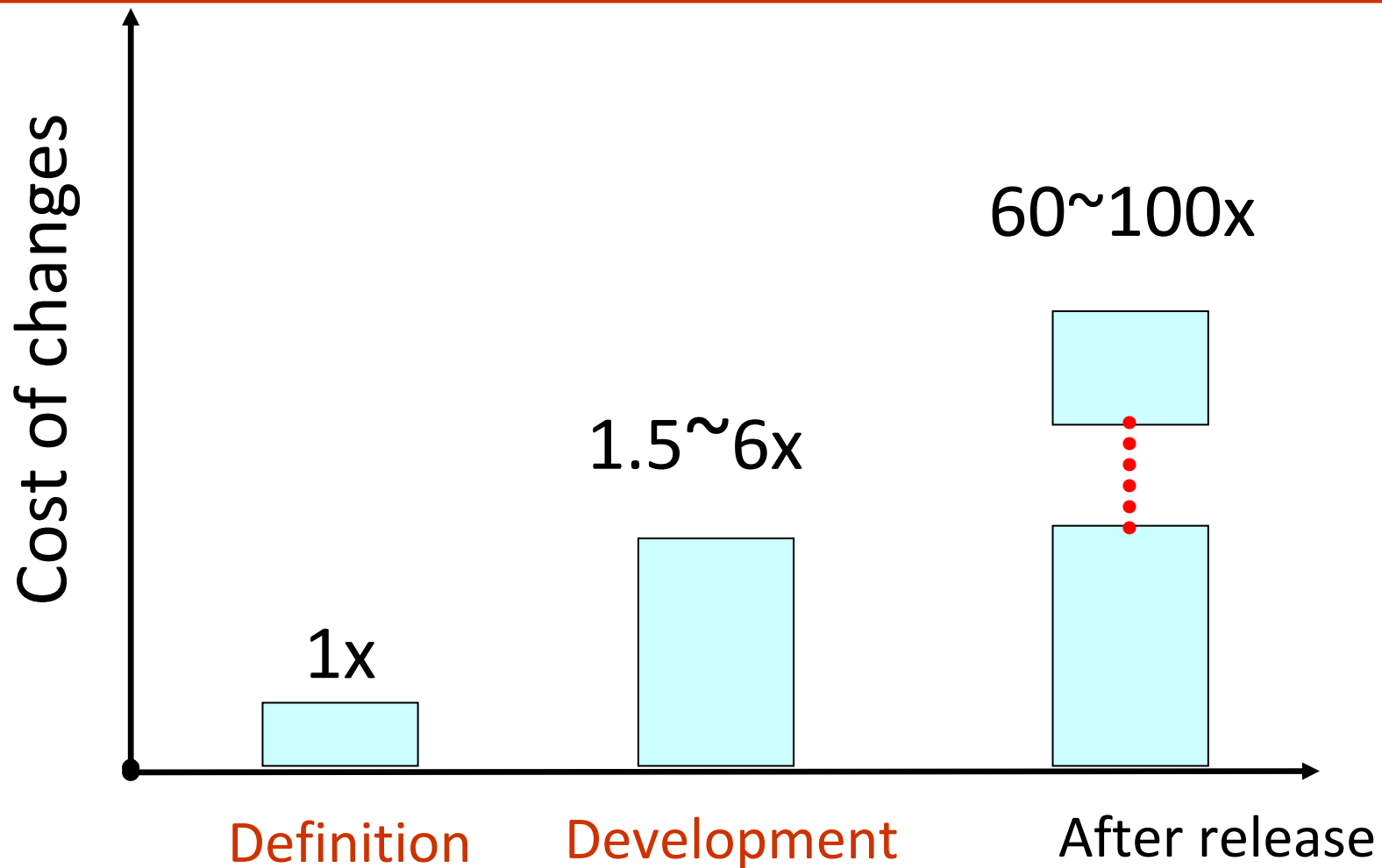
# Pros

---

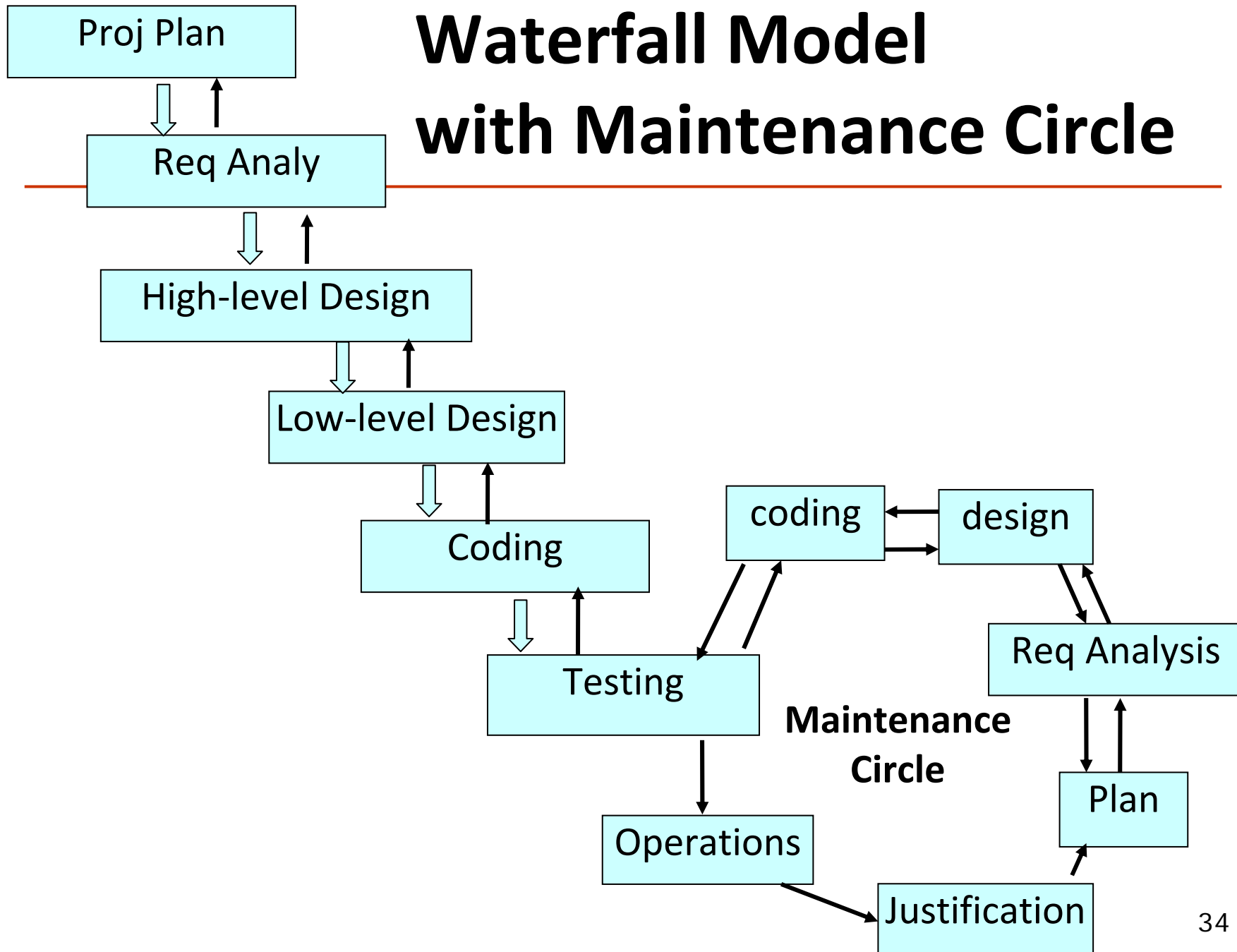
- ❑ Development process is essentially in a linear sequence, manageable
- ❑ Based on "clear and comprehensive requirements", thus able to achieve satisfactory results



# The Impact of Changes




# Waterfall Model with Maintenance Circle

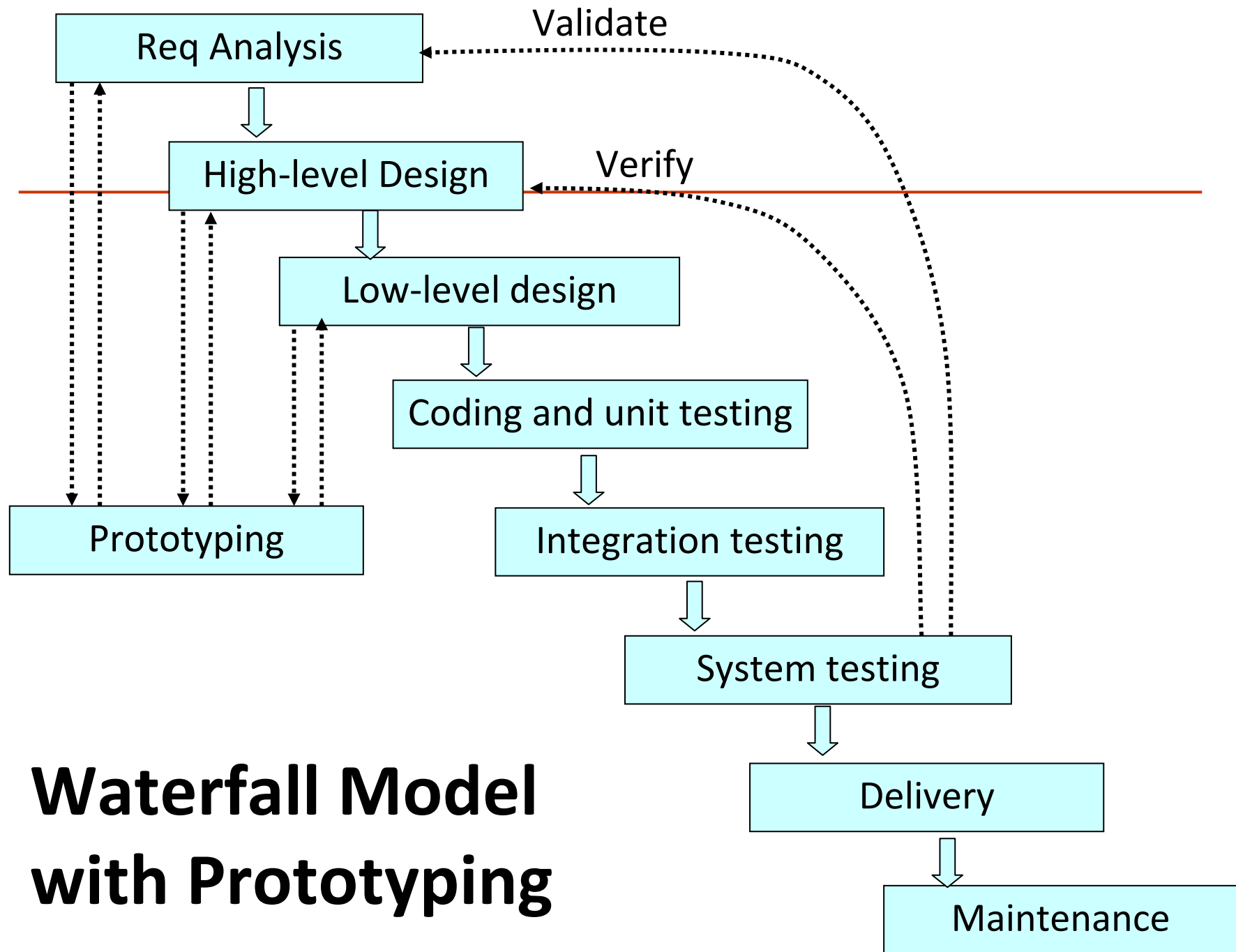


# Evolution Development Model

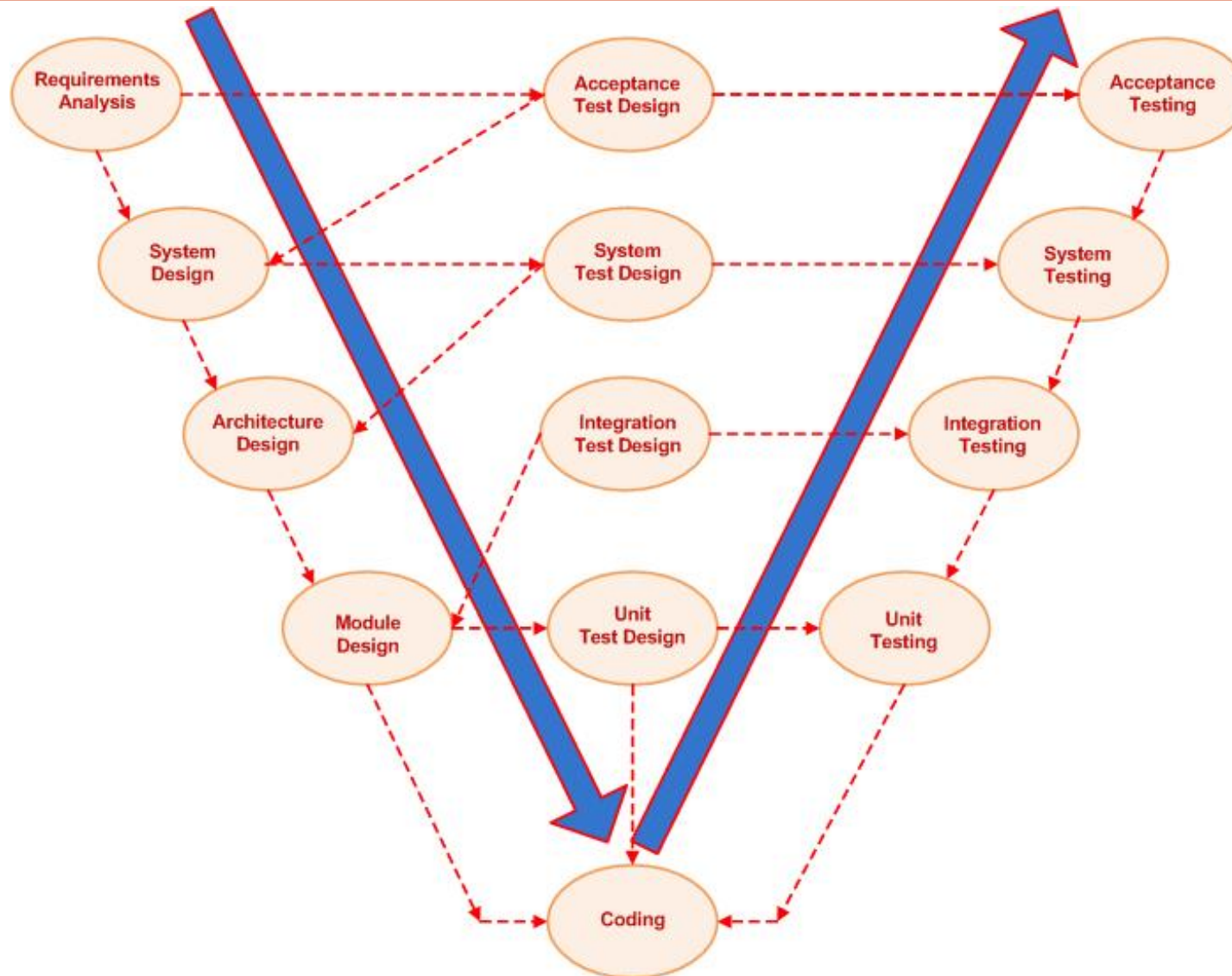
---

- ❑ Based on the initial requirement, first develop a prototype for clients. Then, based on the feedback, continuously improve the system until satisfactory.
- ❑ Requirements can be constantly changed.
- ❑ Two ways of doing it:
  - Working with clients together until delivery
  - Prototype is to understand the clients' needs.

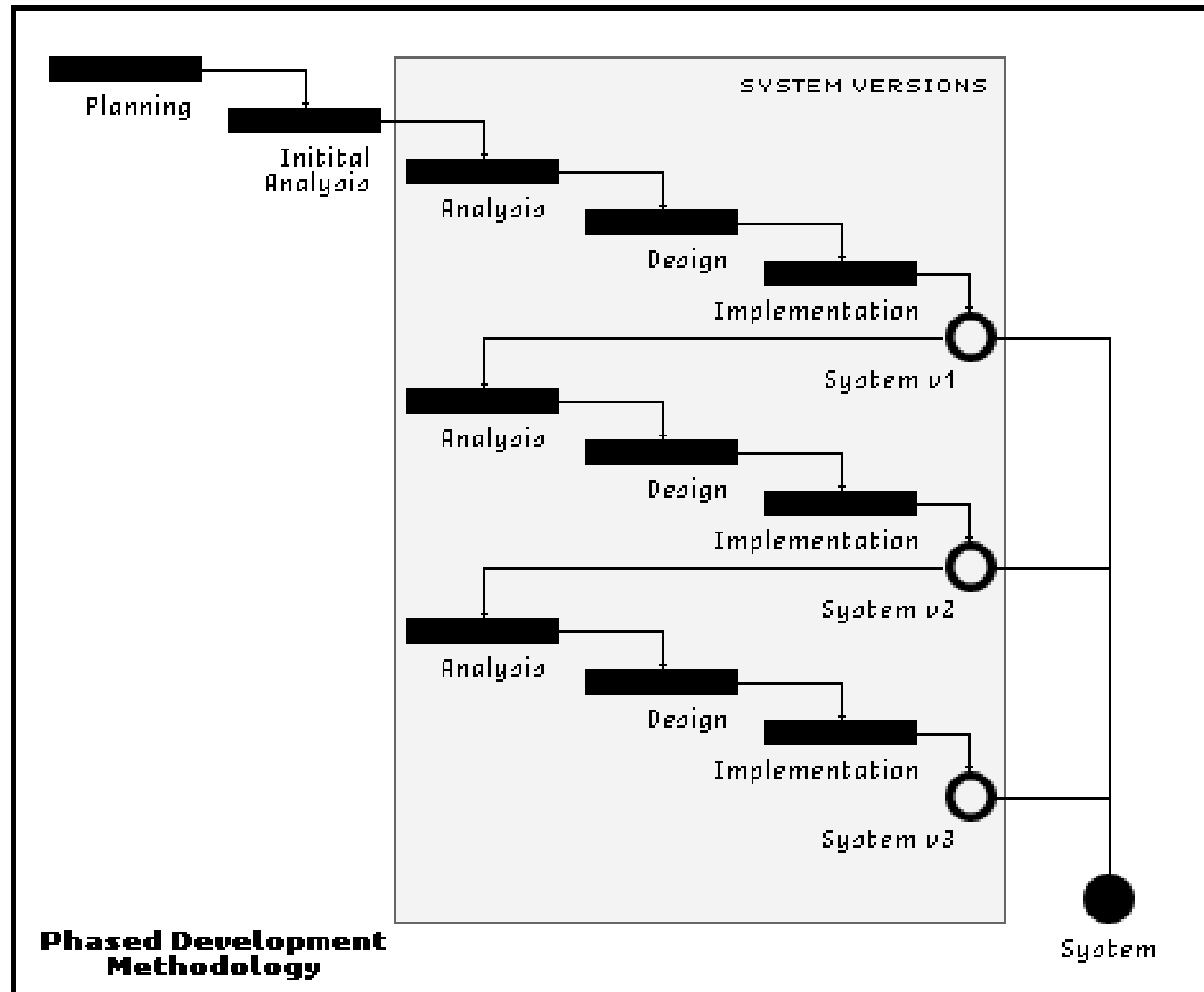
- 
- 
- Problems of evolution dev process:
    - Where is “process”?
    - No clear system architecture
  - In practice, we use the combined model:
    - Waterfall Model with Prototyping
    - V Model
    - Phased Development Model
    - Incremental and Iterative Model
    - Spiral Model



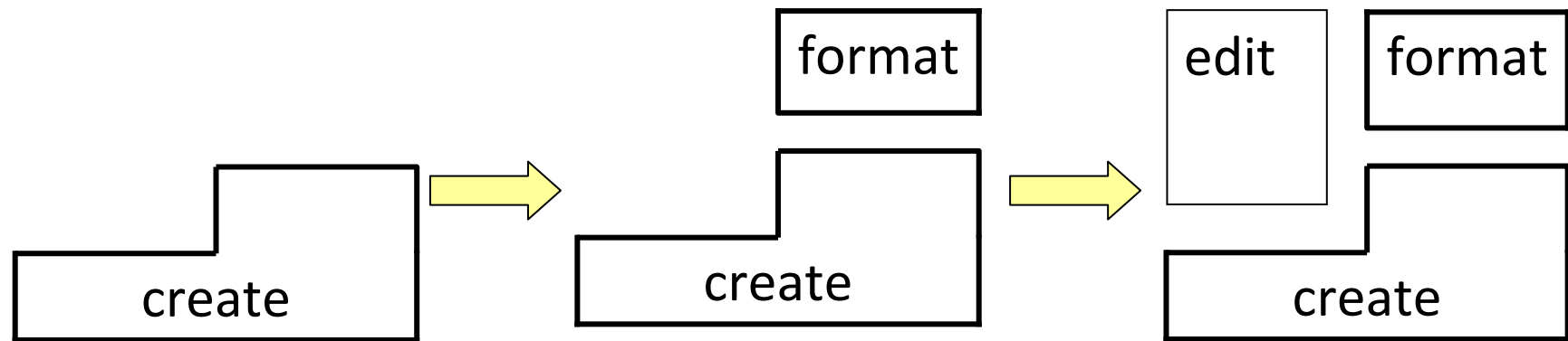
# V-Model



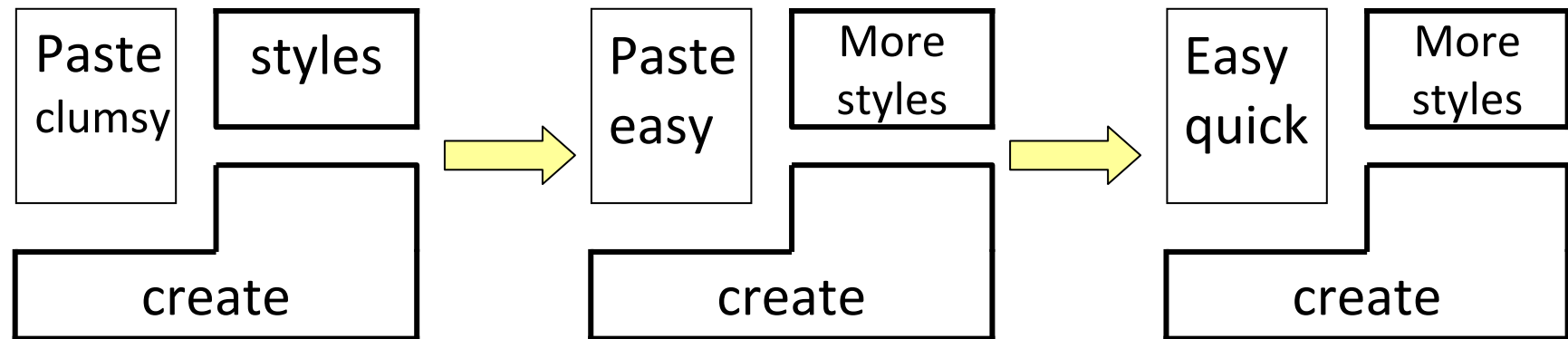
# Phased Development Model



# The Incremental and Iterative Model



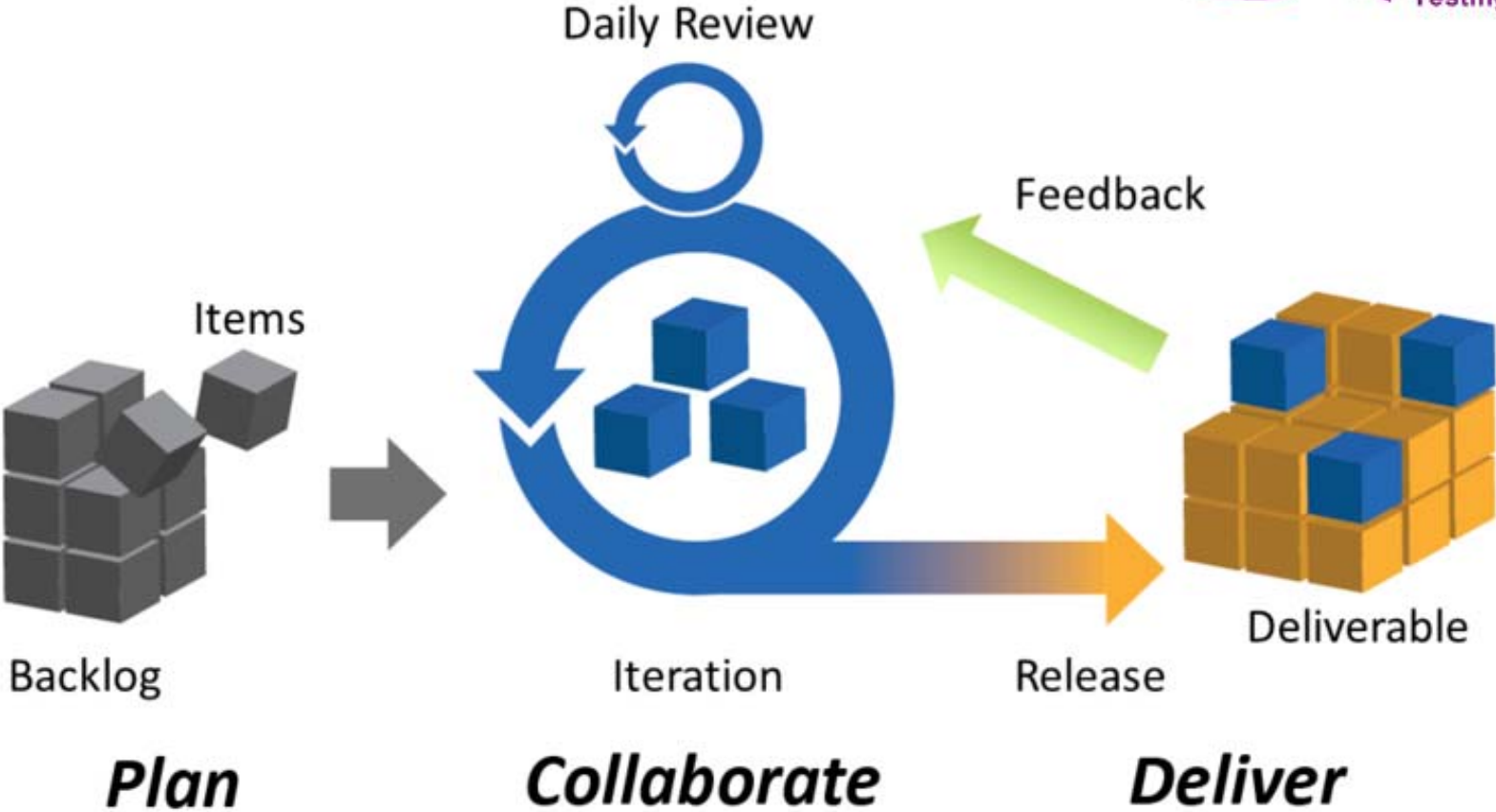
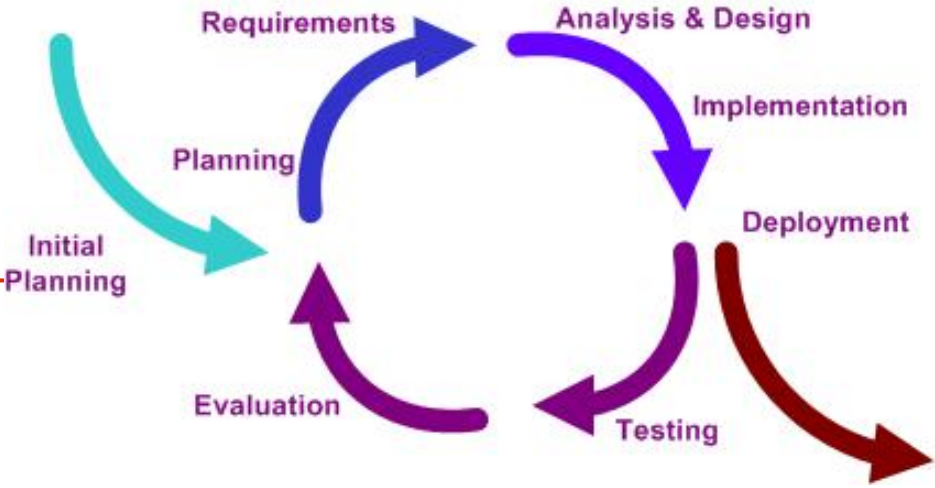
Incremental Development



Iterative Development



# Examples in Agile

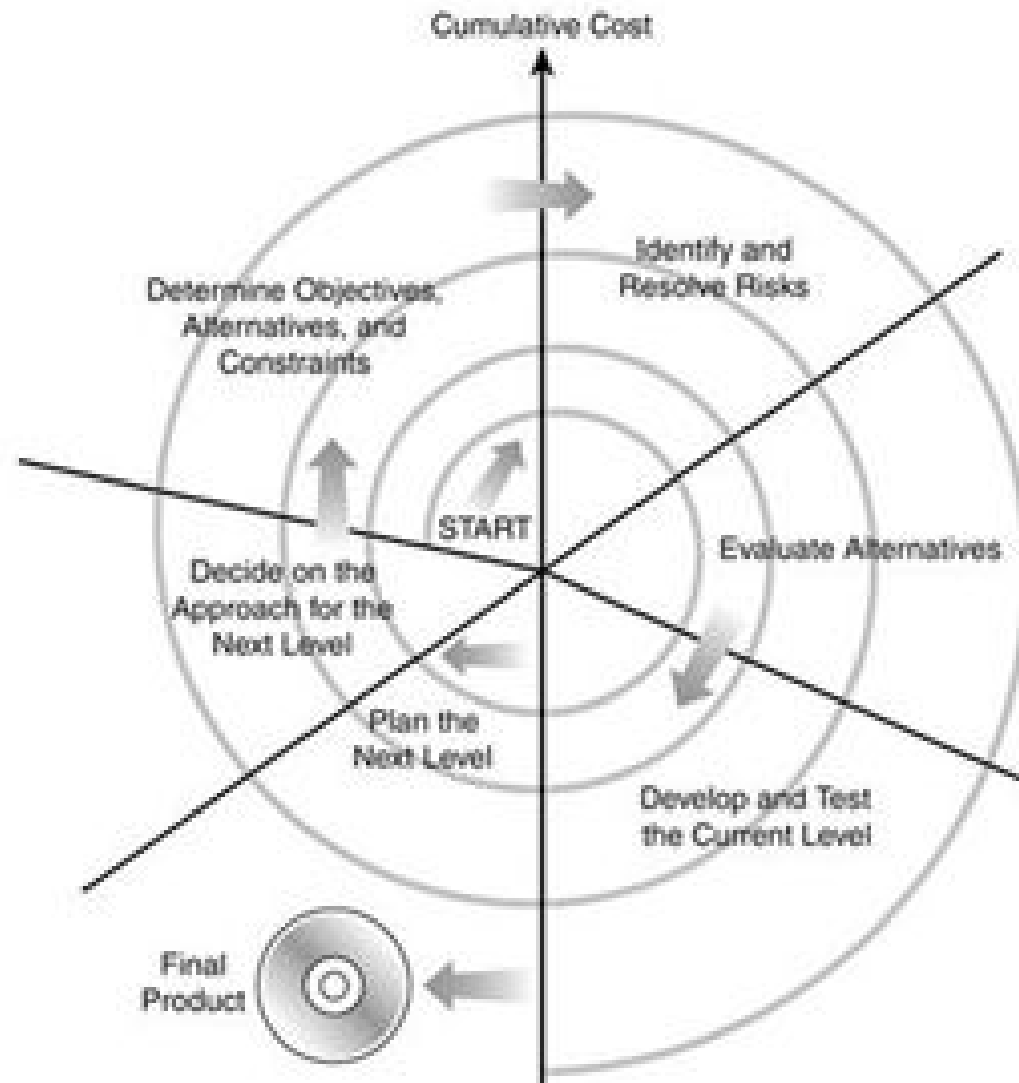


# Benefits of Incremental Development

---

- Can deliver the product with main functionalities within short period of time
- Gradually increase the provided functionalities, so that users have ample time to learn and adapt to the new project

# Spiral Model



# Benefits of Spiral Model

---

- ❑ Software re-use
- ❑ Software quality is important for dev
- ❑ Reduce unnecessary/surplus testing
- ❑ No difference between software maintenance and development

# Rational Unified Process

- Iterative and incremental development
- Use-case driven

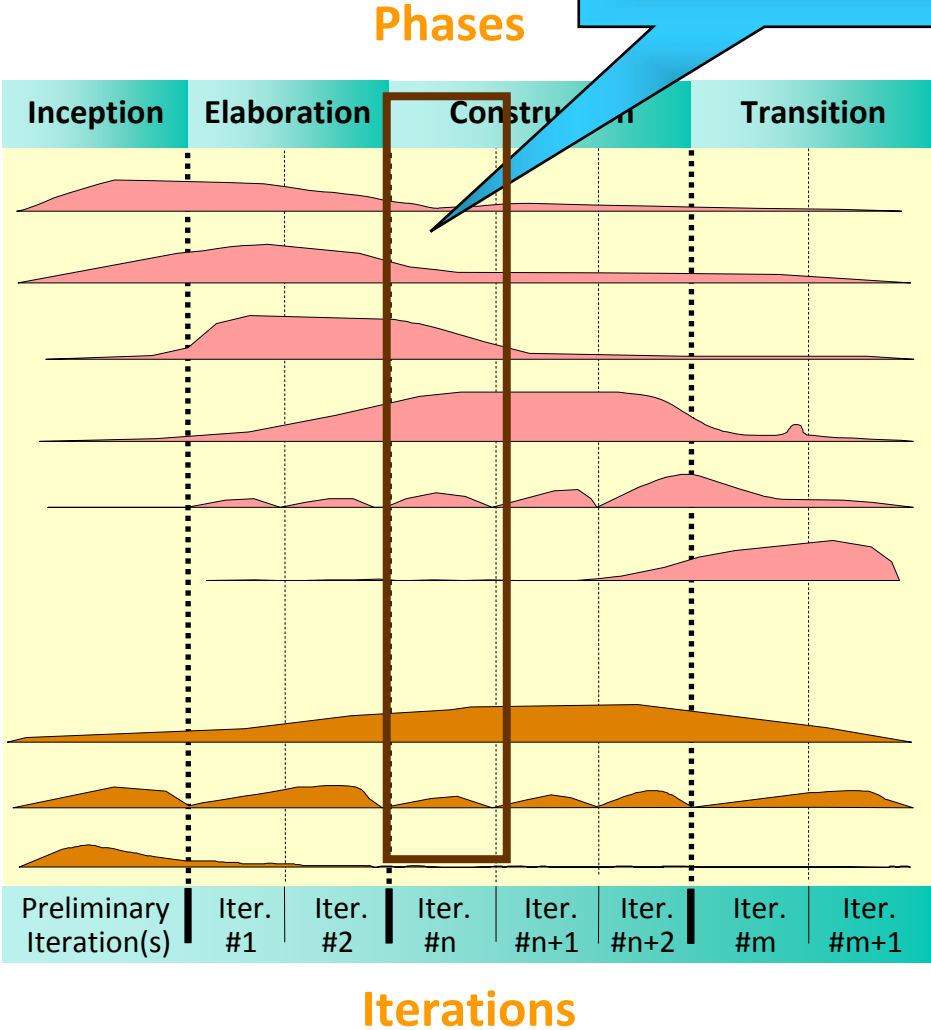
In an iteration, you walk through all workflows

## Process Workflows

- Business Modeling
- Requirements
- Analysis & Design
- Implementation
- Test
- Deployment

## Supporting Workflows

- Configuration & Change Mgmt
- Project Management
- Environment



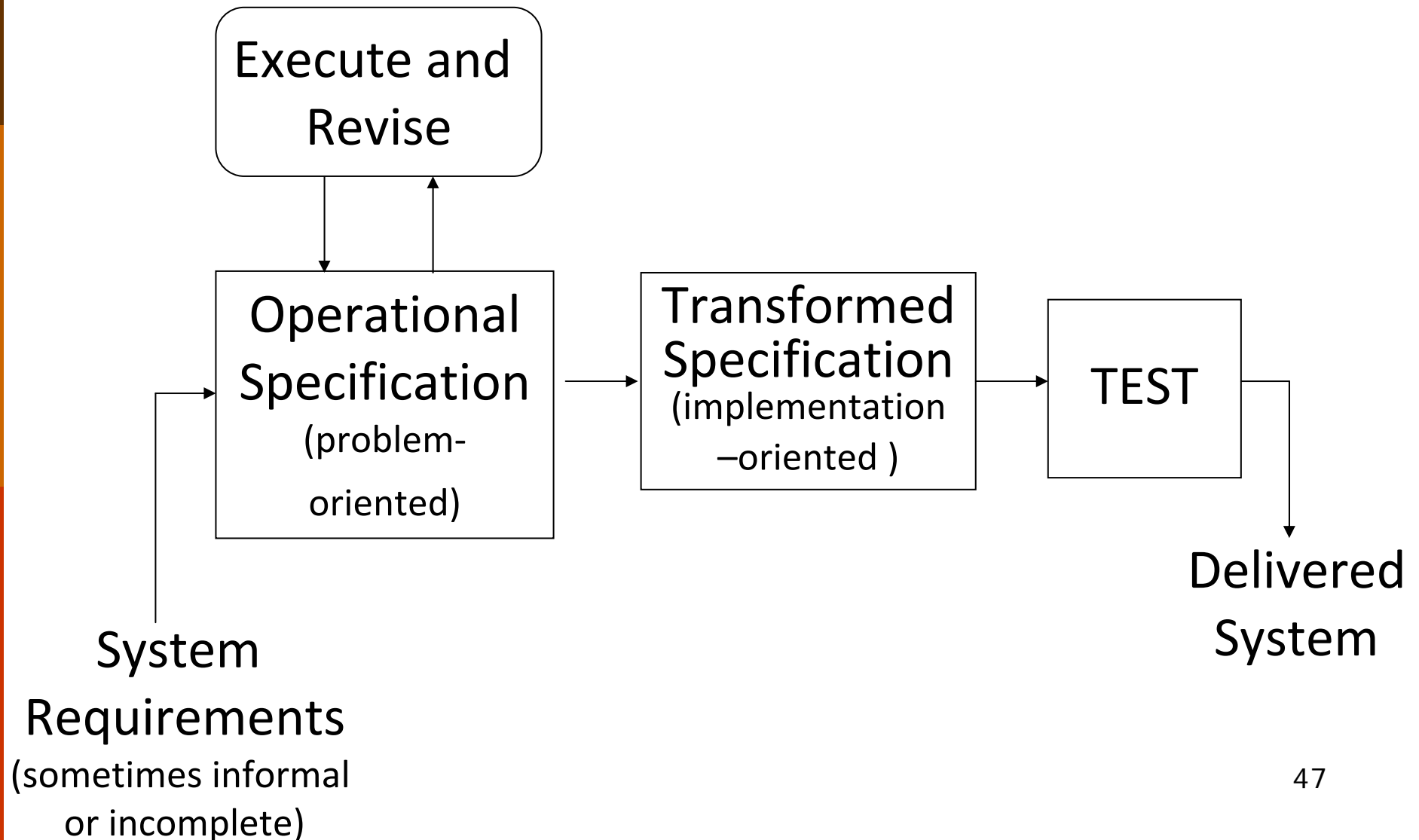
# Formal Methods

---

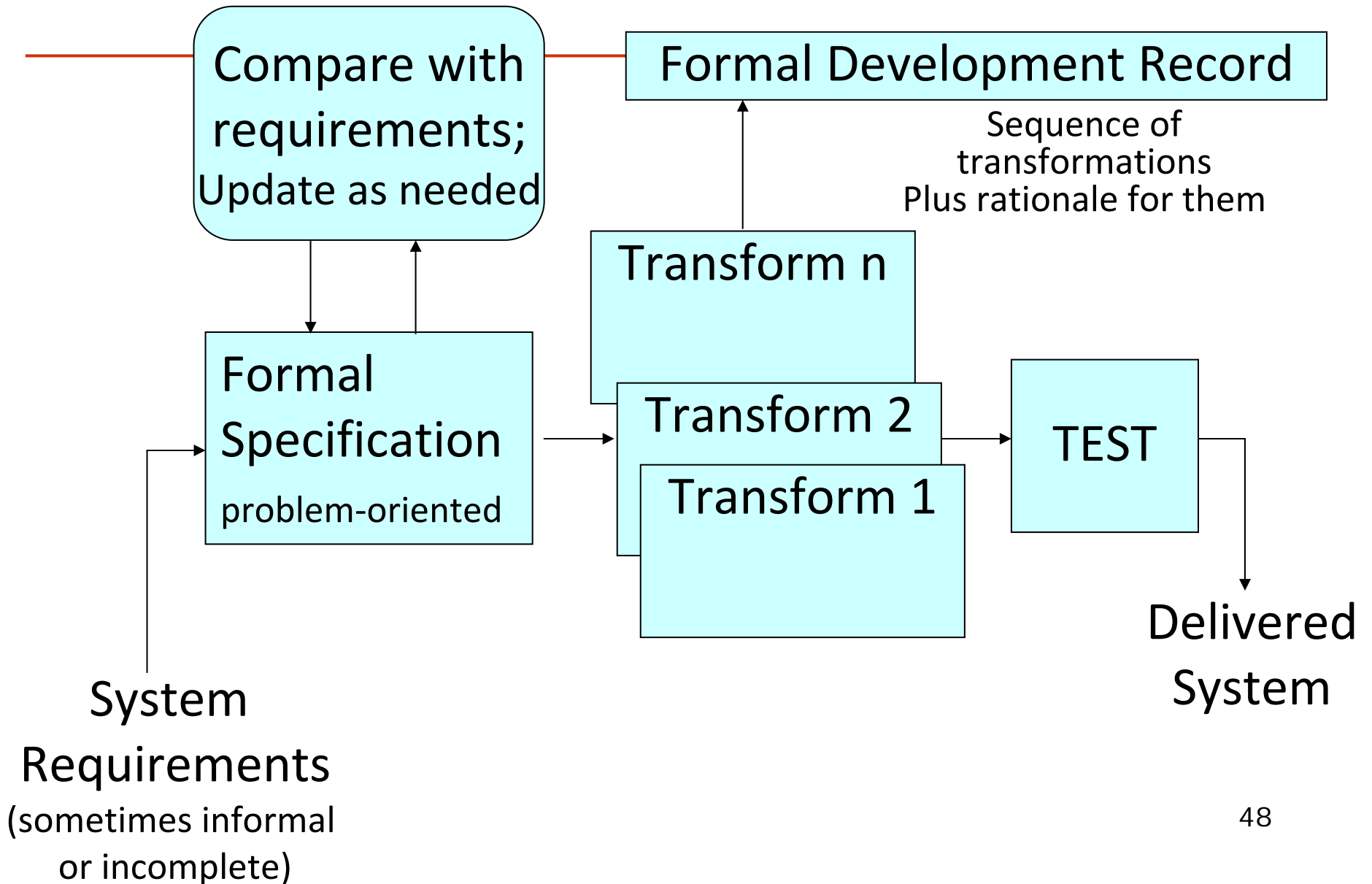
- Similar to the waterfall model, but
  - Requirement descriptions are highly rely on mathematics (e.g., notations, equations...)
  - A series of transformations lead to executable programs
- Cleanroom Process by IBM

# Operational Specification Model

---



# Transformational Model





## 2. Software Descriptions

---

- Feasibility study report
- Requirement analysis
- Requirement descriptions
- Requirement validations

# 3. Software Designs

---

- ❑ Architectural design
- ❑ Subsystem abstract description
- ❑ Interface design
- ❑ Component design
- ❑ Data structure design
- ❑ Algorithm design

# 4. Software Validation

---

- Unit testing
- Functional module testing
- Subsystem testing
- System testing
- Delivery testing (  $\alpha$  、  $\beta$  、  $\lambda$  )

# 5. Software Evolution

---

- ❑ Definition: Software evolution is the term used in software engineering (specifically software maintenance) to refer to the process of developing software initially, then repeatedly updating it for various reasons.
- ❑ The nature of software flexibility determines that software evolution is a must
  - Avoid expensive HW modifications
- ❑ Can be regarded as an integrated process between dev and maintenance

# 6. Software Automation

---

- CASE
- Limitations
  - By essence, software engineering is a creative activity
  - Software process emphasizes the team work

# Content

---

- ❑ Basic Concept of Software Engineering
- ❑ Computer-based System Engineering
- ❑ Software Process
- ❑ **Basic elements of software project management**

# Software Project Management

---

- ❑ clear distinction with other project management.
- ❑ Software PM needs to take a variety of different tasks, one of the most important activities are project planning, estimating costs and schedule control.
- ❑ Project milestone is sth that is expected and planned. When reaching a milestone, PM needs to report the progress to the management board.
- ❑ In order to effectively track/control the project progress, PM can use a variety of charts/diagrams.
- ❑ Should identify and assess the project risk, determine the likelihood of these risks and consequences.

# 1. What to Manage?

---

- ❑ People management
- ❑ Cost estimations
- ❑ Quality control
- ❑ Process improvement
- ❑ Software configuration management



## 2. Project Planning

---

- Essence is to estimate the potential risks and adopt effective method to solve the problems
- Contents:
  - Project plan
  - Quality plan
  - Configuration management plan
  - Maintenance plan
  - Training plan
  - ...

# 3. Resource Planning

---

- PM needs to estimate the amount of “resources” to use in order to complete the project, and organize these resources in an optimal way. They are:
  - HW
  - SW
  - Costs
  - Space
  - People
  - Time
  - ...

# 4. Risk Management

---

- Identify the potential risks and make the plan, to maximally lower their impact to the project
- Three types of risks
  - Product: that have impact on the software quality and performance
  - Project: that have impact on the project progress and resources
  - Business: that have impact on the software company and/or clients
- How to manage the risks?
  - Identification, analysis, plan, and monitoring

# Potential Software Risks

---

Risks	Type	Descriptions
Staff quit	project	Experienced staffs
Re-org	project	
Lack of HW	project	e.g., ordered HW is not delivered
Change of req	project	inevitable
Description delay	Project& product	
Underestimate the system scale	Project& product	Over budget
Poor CASE perf	product	Some CASE tools are very expensive
Tech change	Biz	
Peer competition	Biz	

# Risk Analysis

<b>Risks</b>	<b>Possibility</b>	<b>consequence</b>
<b>Financial problems, to cut the budget</b>	<b>Low</b>	<b>Significant</b>
<b>Lack of qualified staff</b>	<b>High</b>	<b>Significant</b>
<b>Staff report sickness at important stage</b>	<b>Medium</b>	<b>Significant</b>
<b>Defect on re-useable components</b>	<b>Medium</b>	<b>Significant</b>
<b>Req change leads to re-design</b>	<b>Medium</b>	<b>Significant</b>
<b>Re-org</b>	<b>High</b>	<b>Significant</b>
<b>Database problem</b>	<b>Medium</b>	<b>Significant</b>
<b>Underestimate the dev efforts (time)</b>	<b>High</b>	<b>Significant</b>
<b>CASE tools cannot be used</b>	<b>High</b>	<b>Tolerable</b>
<b>Clients not sure of the req</b>	<b>Medium</b>	<b>Tolerable</b>
<b>Lack of training</b>	<b>Medium</b>	<b>Tolerable</b>
<b>Underestimate the software size</b>	<b>High</b>	<b>Tolerable</b>
<b>Codes generated by CASE tools are inefficient</b>	<b>Medium</b>	<b>Negligible</b>

# Causes of Software Risks

---

Type	Potential risks	Examples
<b>Tech</b>	DB, re-useable components, or COTS	HW/SW delivery delay, exposes too many tech problems
<b>Staff</b>	Recruitment, staff changing jobs, training	Tasks are not assigned properly, team internal chaos
<b>Org</b>	Re-org, financial problems	
<b>tools</b>	CASE tools	Using tools are not easy
<b>req</b>	Change, misunderstanding, communications	Too much changes, client complaints
<b>plan</b>	Underestimate the software size, timing, defects	Progress is delayed

# Risk Management Strategies

---

<b>Risks</b>	<b>Strategies</b>
<b>Financial problems</b>	<b>Draft a report to the mgmt team, explain the impact to the project</b>
<b>recruitment</b>	<b>Tell clients the potential risks/delay this may cause</b>
<b>Staff sickness</b>	<b>Re-form the dev team, flexible task assignment, efficient communications.</b>
<b>Component defect</b>	<b>Purpose reliable components</b>
<b>Req change</b>	<b>Evaluate the impact of change, increase the traceability of the dev process</b>
<b>Re-org</b>	<b>Draft a report to the mgmt team, explain the impact to the project</b>
<b>DB performance</b>	<b>Purpose high performance DB</b>