

# The Process



Modified from Roger S. Pressman, Software Engineering:  
A Practitioner's Approach 8<sup>th</sup> Edition, McGraw Hill, 2014

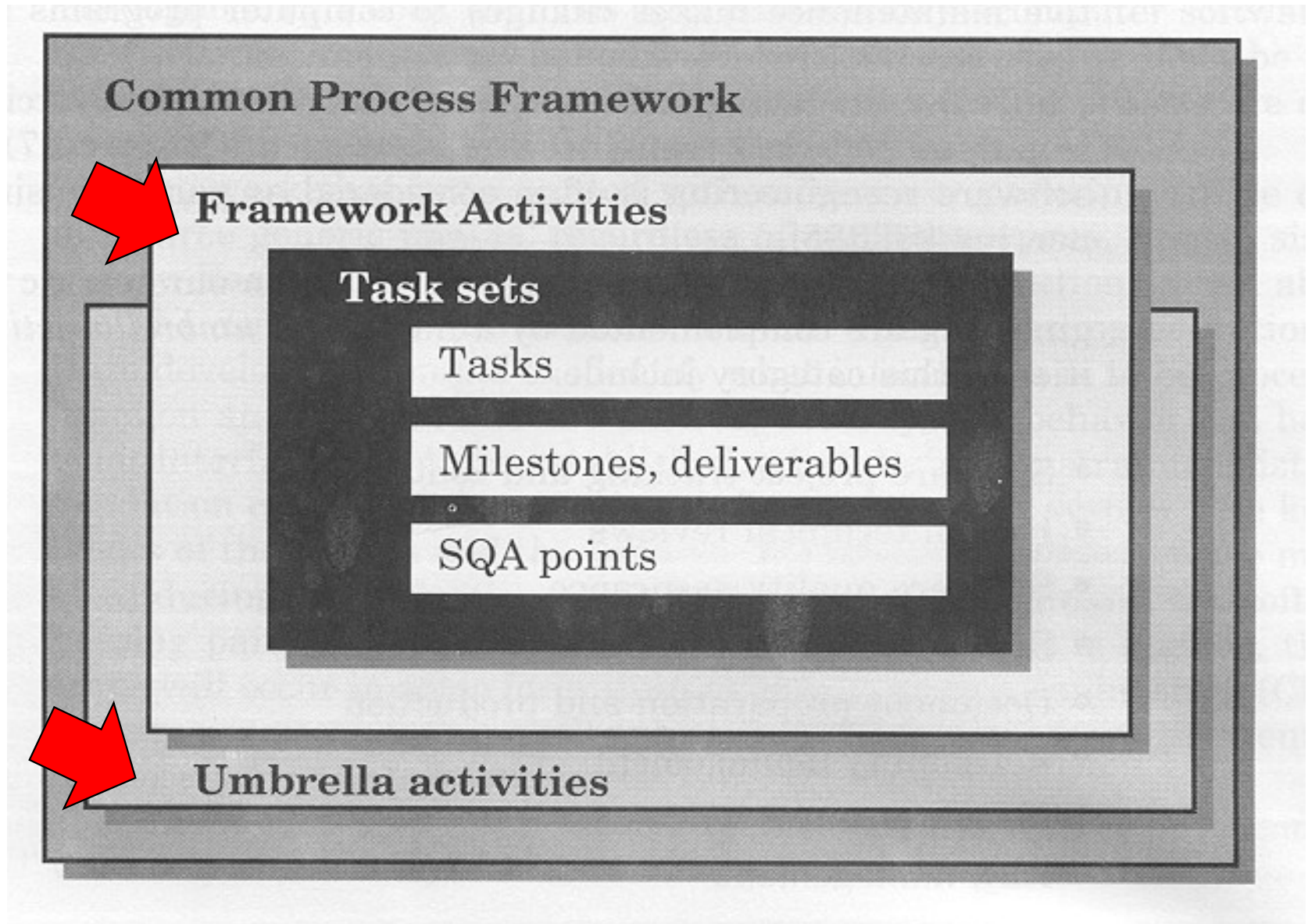
# Roger Pressman's 4P

- **Product** - Software product
- **Process** - SDLC (Waterfall, SCRUM, etc.)
- **Project** - Project Mgt. (PMBOK version 7)  
(5 process groups – project initiating, project planning, project executing, project monitoring and control, project close)
- **People** - Team (organizational structure – projectized, functional, matrix)

# What is Software Engineering?

- 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, 1969]
- the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of it [IEEE, 1990]
- includes the process of analyzing and synthesizing by using a variety of methods, tools, procedures, and paradigms [Pfleeger, 1998]

# The Software Process

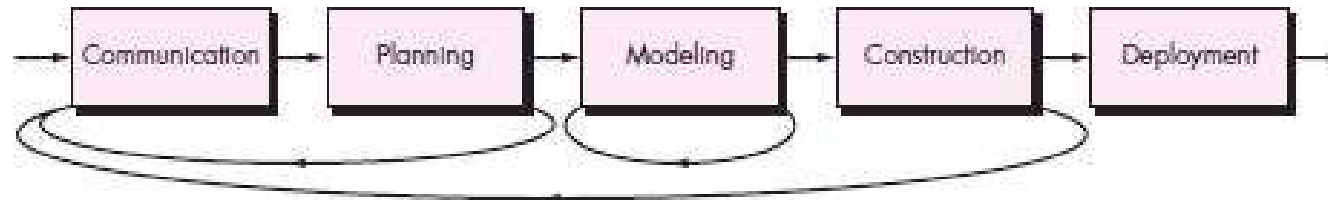


# Generic Process Framework

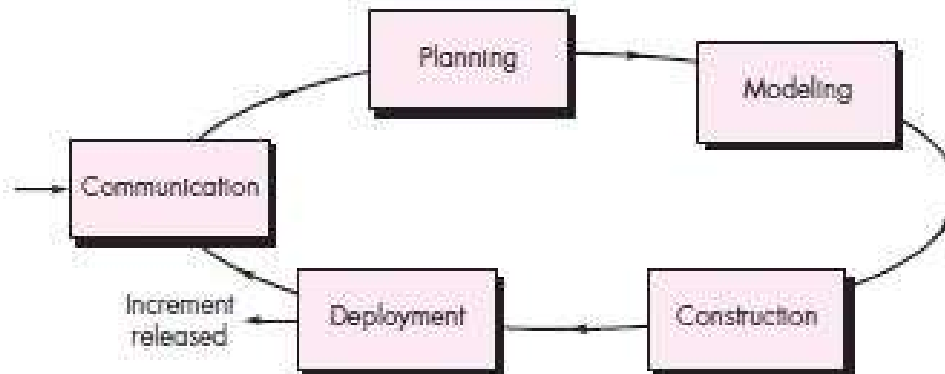
- A generic process framework for software engineering defines **five** framework activities **communication, planning, modeling, construction, and deployment.**
- In addition, a set of umbrella activities- project tracking and control, risk management, quality assurance, configuration management, technical reviews, and others-are applied throughout the process.



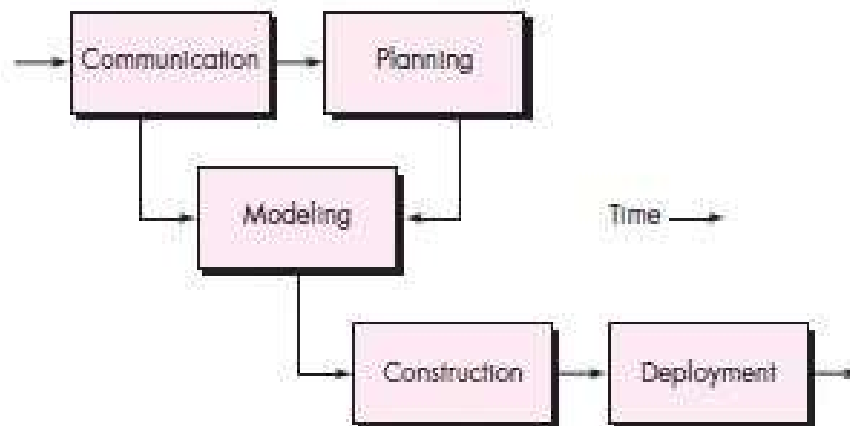
(a) Linear process flow



(b) Iterative process flow



(c) Evolutionary process flow



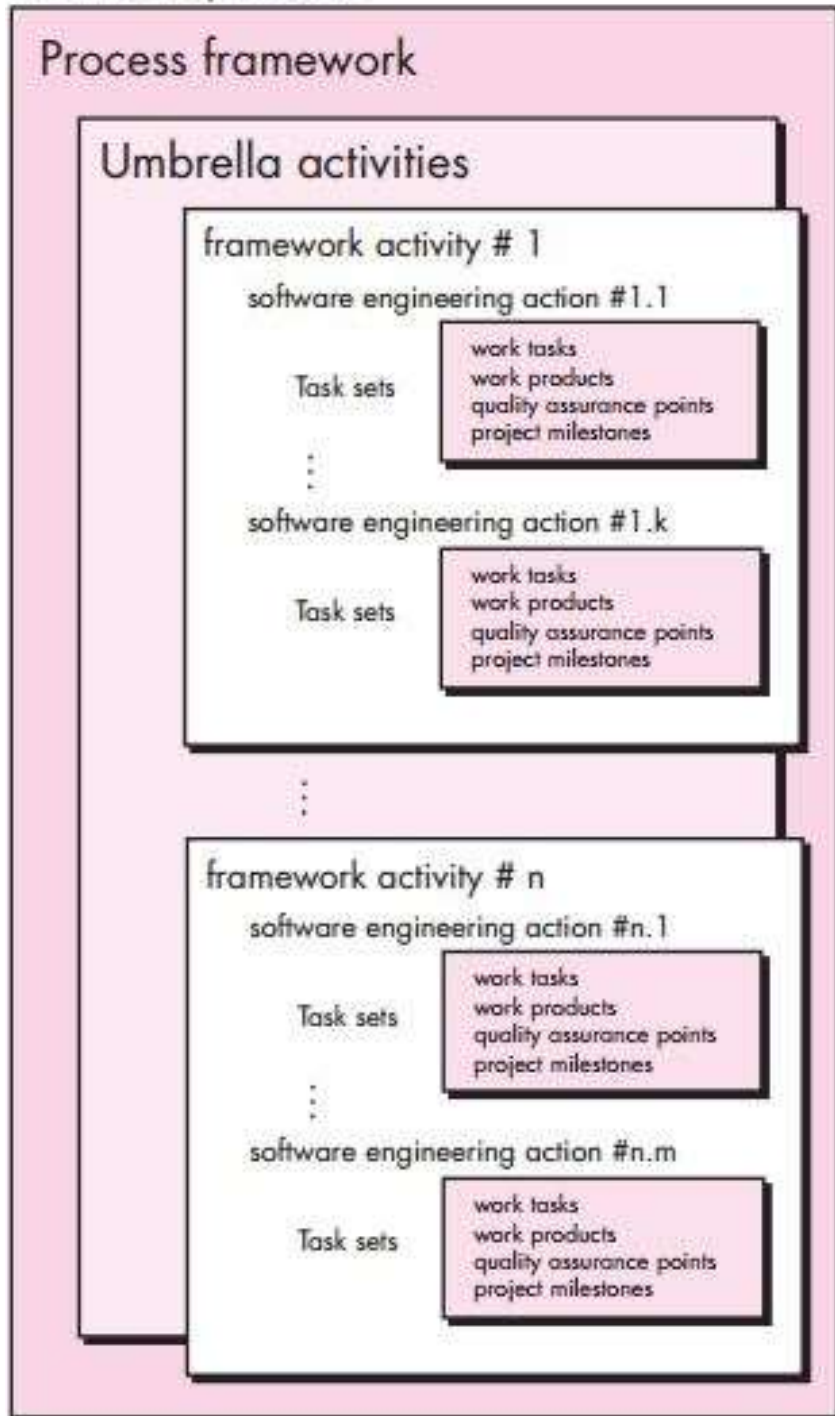
(d) Parallel process flow

Linear Sequential  
Process Model

Several  
types of  
Framework  
activities

Unified Process Model

## Software process



## Waterfall model

Process framework :

Framework act#1: Req. Elicitations

SE Actions 1.1: Kickoff meeting

Tasksets:

Task 1.1.1: Create agenda

work product?

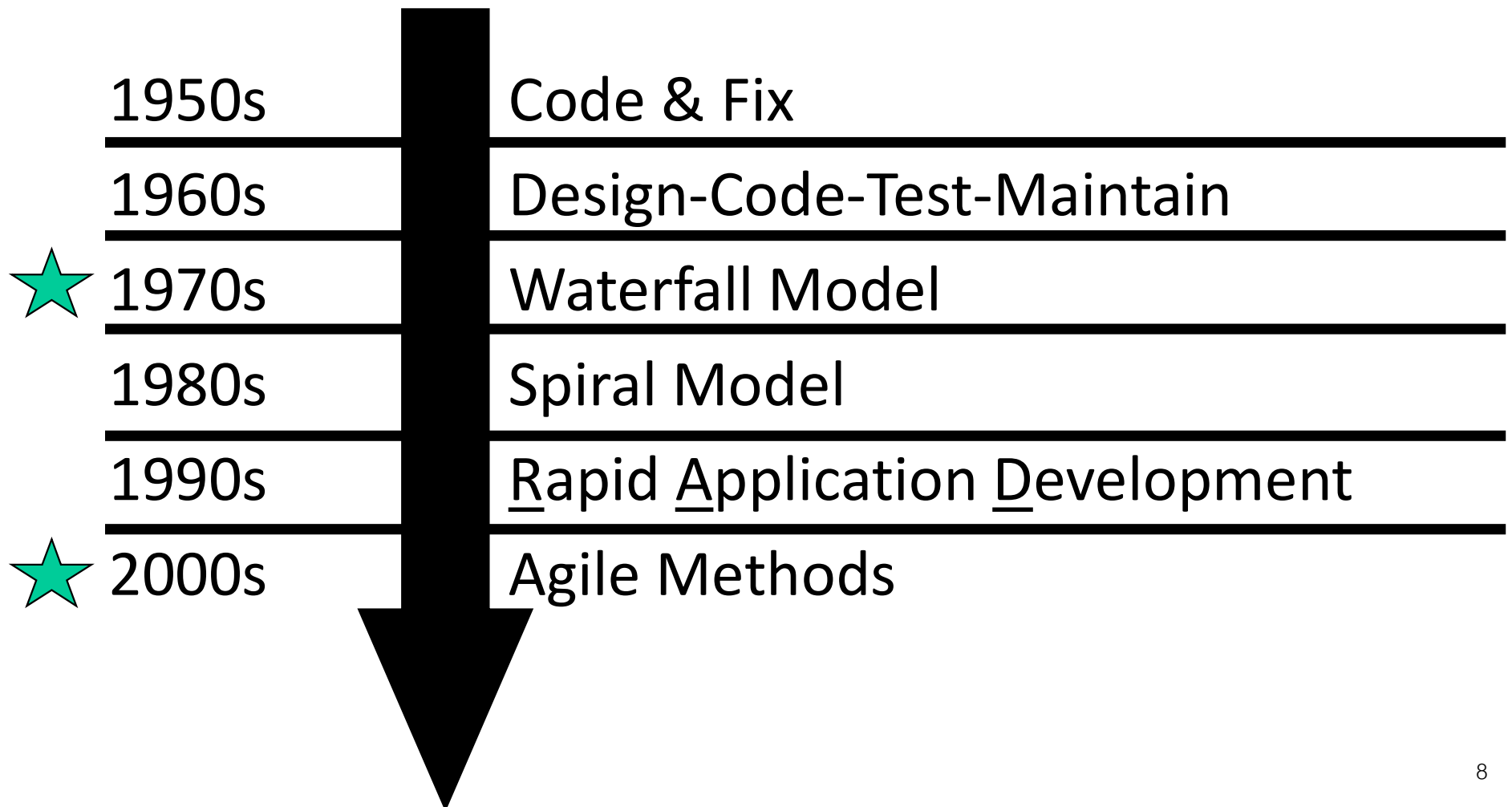
QA Points?

Milestone?

Task 1.1.2: Assign roles

Framework act#2: .....?

# Timeline of Methodologies

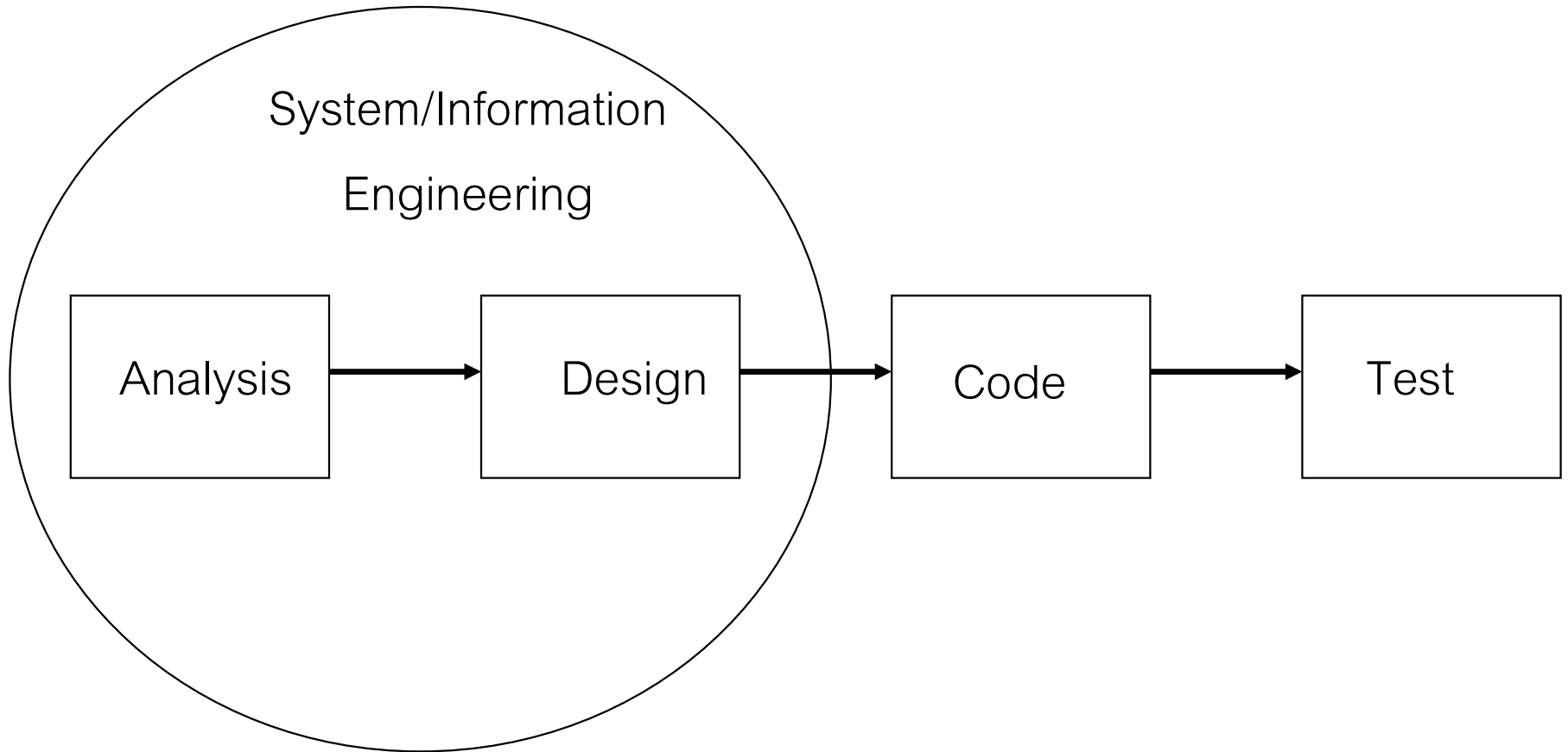




# The Linear Sequential Model

- Classical Life Cycle or Waterfall model consists of activities:
  - System/information and engineering modeling
  - Software Requirement Analysis
  - Design
  - Code Generation
  - Testing
  - Support

# The Linear Sequential Model



# Reference

- Royce, W.W., 1970, "*Managing the Development of Large Software Systems*", Proceedings of IEEE WESCON 26 (August), pp.1–9.

# Winston W. Royce

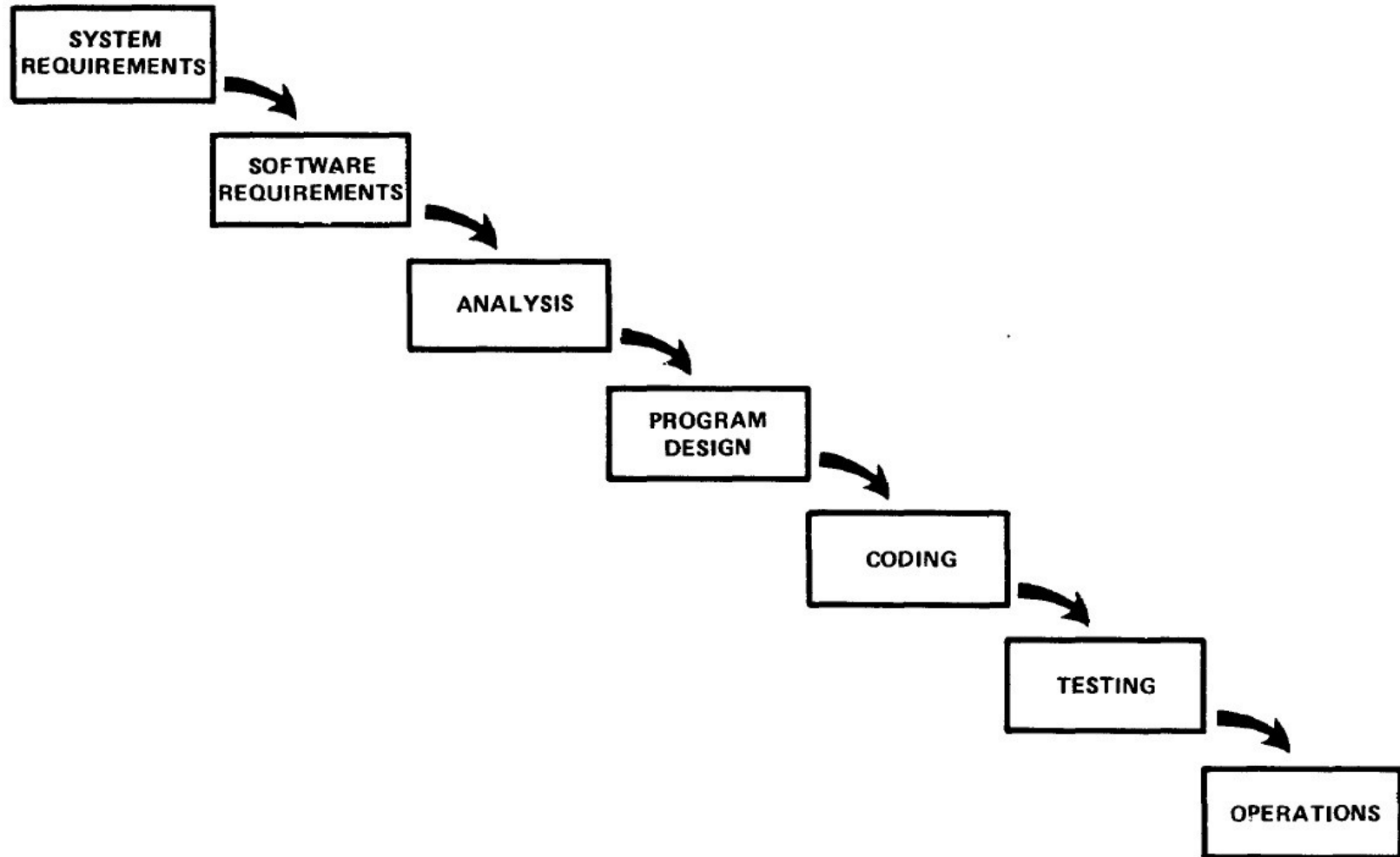
- Born in 1929.
- Died in 1995.
- An American computer scientist, director at Lockheed Software Technology Center in Austin, Texas, and one of the leaders in software development in the second half of the 20th century.
- He was the first person to describe the “Waterfall model” for software development, although Royce did not use the term "waterfall" in that article, nor advocated the waterfall model as a working methodology.



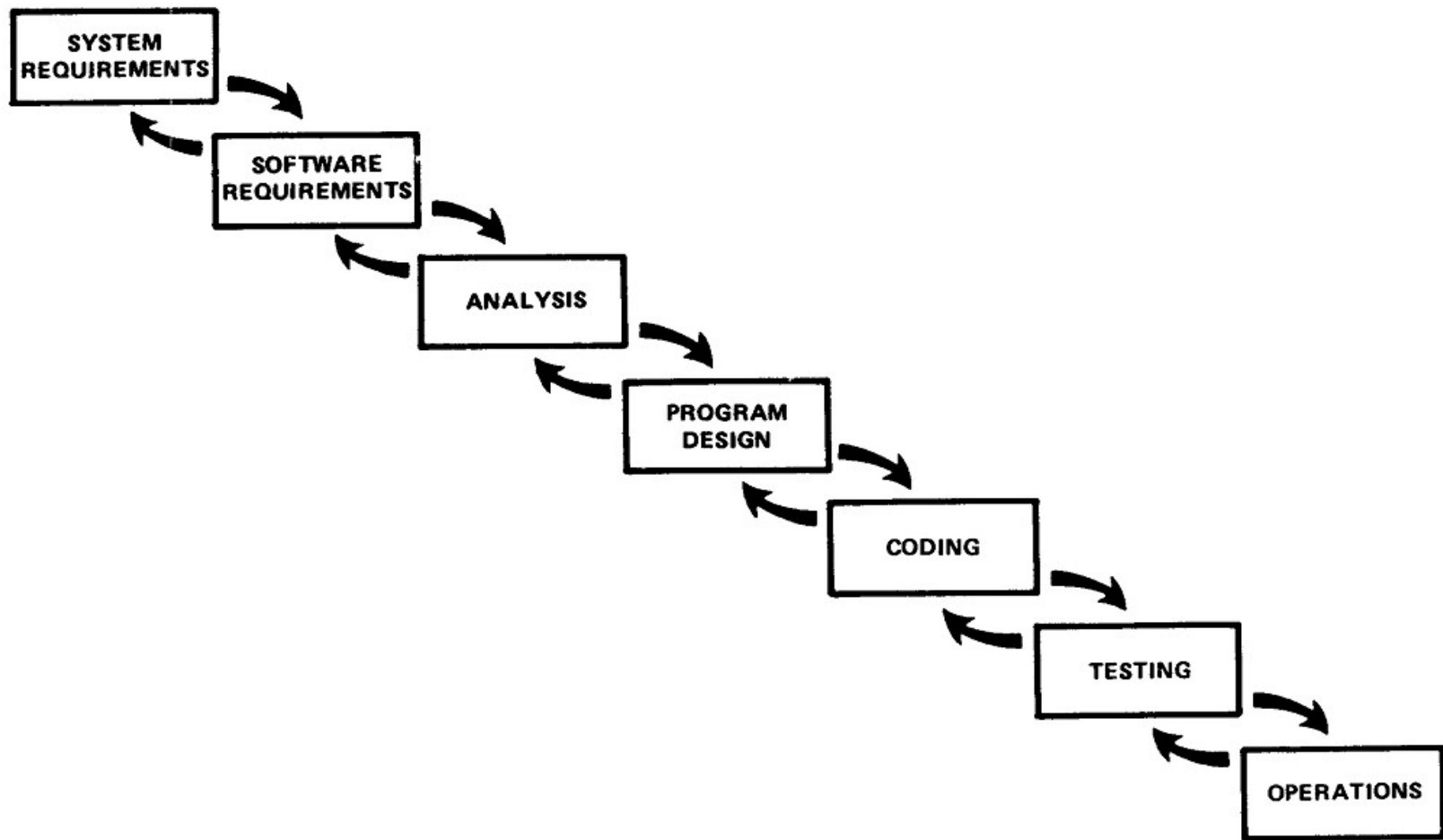
# Waterfall Model

- Royce **1970**
- the stages are depicted as cascading from one to another
- one stage should be completed before the next begins
- has been used in variety context, for example it was a basis for deliverables in U.S. Department of Defense Contracts and defined in DOD**2167**-A
- deliverables are produced in each process
- the model will be very useful in helping developers lay out what they need to do

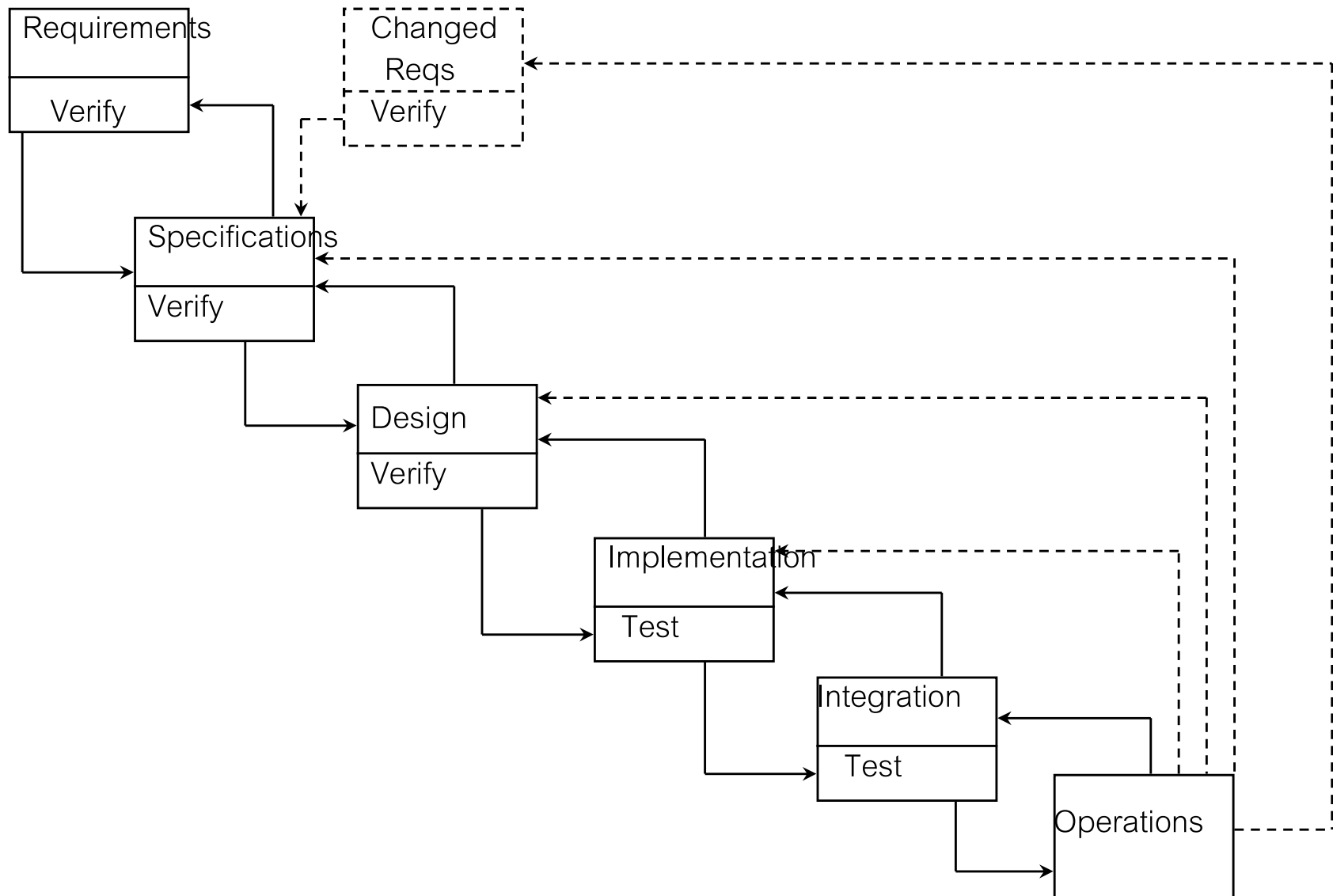
# Large Developments



# Iterative Relationship between Successive Development Phases



# Waterfall Model





# Drawbacks of the Waterfall Model

- it shows how each phase terminates in the production of some artifact, but there **is no insight into how** each activity transforms one artifact to another
- it provides **no guidance** on how to handle changes to products and activities
- it is **too late** to show the customers what the product looks like

## Waterfall Model (cont.)

- **prototyping** enables customers and developers to examine some aspect of the proposed system
- often, the **user interface is built and tested as a prototype**, so that the users understand what the new system will be like

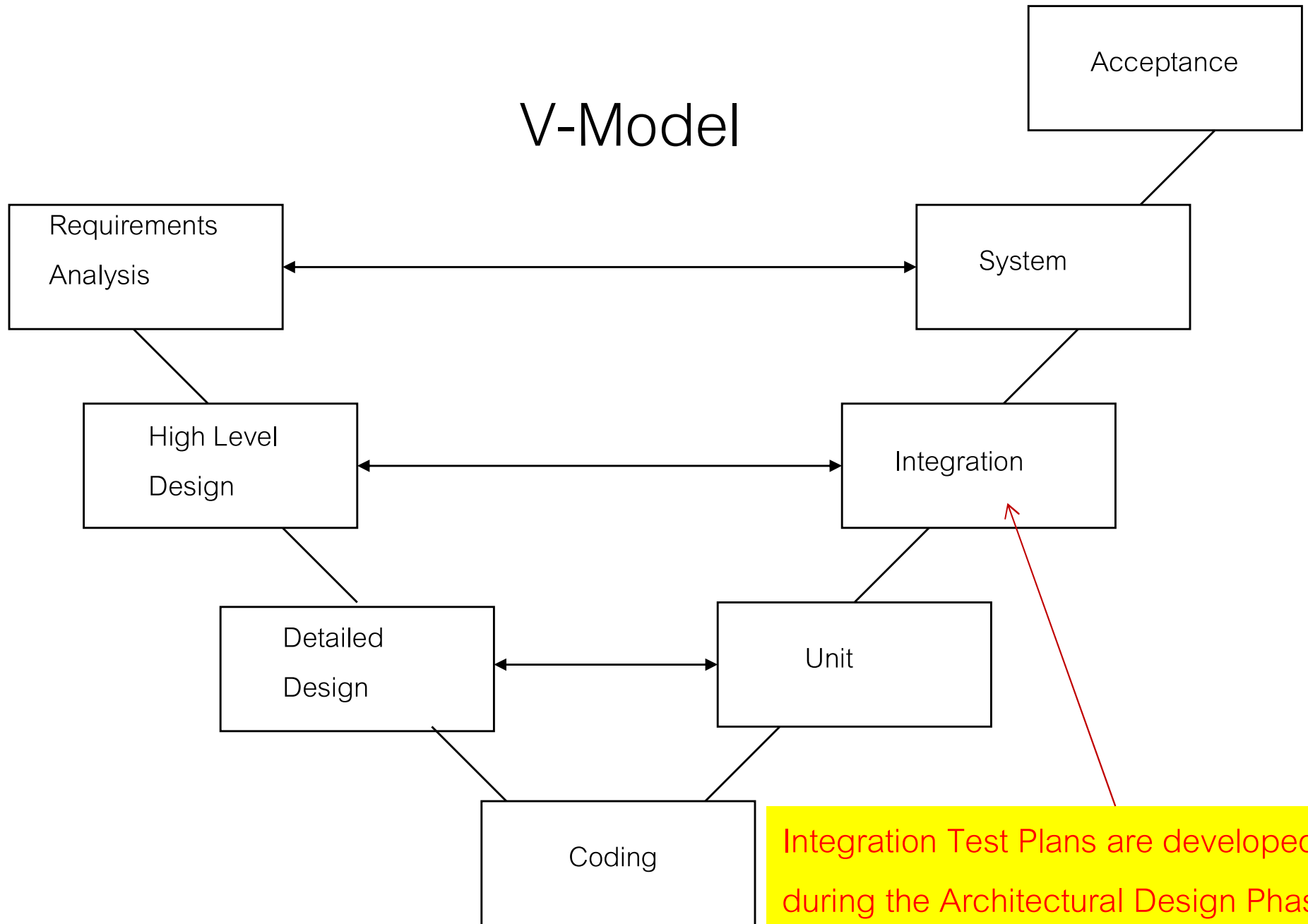
# V-Model

- is a variation of the waterfall model that demonstrates how the testing activities are related to analysis and design
- analysis and design activity on the left
- testing and maintenance on the right
- unit and integration testing are used to verify the program design
- system testing should verify the system design
- acceptance testing is conducted to validate the requirements

## V-Model (cont.)

- there is linkage of the left side with the right side
- it implies that if problems are found during verification and validation, then the left side of V can be reexecuted to fix and improve the requirements, design, and code before the testing steps on the right side are reenacted

# V-Model

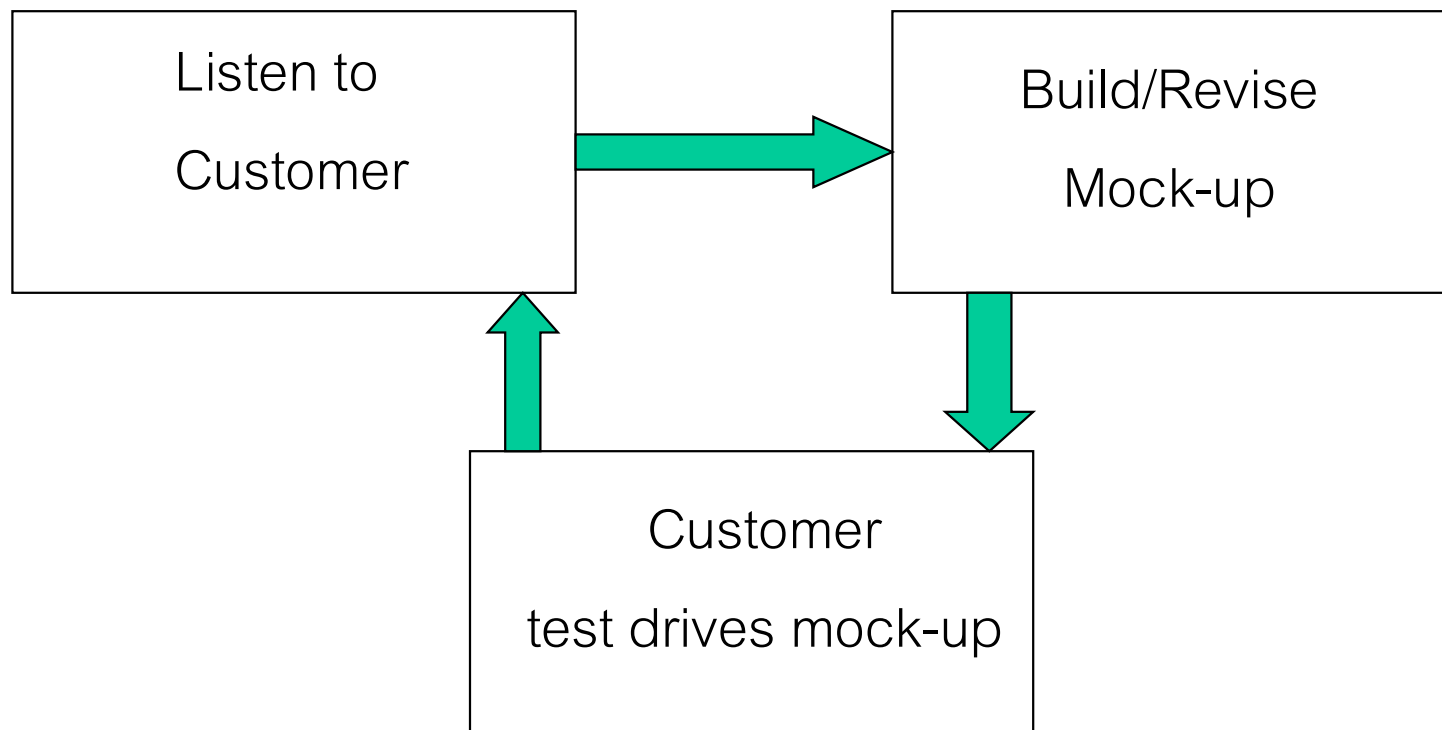


Integration Test Plans are developed during the Architectural Design Phase. เช่นเดียวกันกับ Unit, System Test ด้วย

# The Prototyping Model

- allows all or **part of a system to be constructed quickly** to understand and clarify issues
- begins with requirements gathering
- **possible screens, tables, reports, and other system output** are produced
- as the users and customers **decide on what they want**, the requirements are revised
- once everyone agrees on the requirements, the developers move on to the design

# The Prototyping Paradigm



# The RAD Model

- RAD stands for **Rapid Application** Development
- uses **component-based** construction
- enables a development team to create a “fully functional system” within a very short time (**60**-90 days)



# Phases of the RAD Model

- Business modeling:
  - What information drives the business process?
  - What information is generated?
  - Who generates it?
  - Where does the information go?
- Data Modeling: identify attributes of each object and relationships between objects

## Phases of the RAD Model (cont.)

- Process Modeling: create process description for adding, modifying, deleting, or retrieving data objects
- Application generation:
  - RAD uses fourth generation techniques
  - reuse existing program components or create reusable components
- Testing and Turnover:
  - new components must be tested
  - all interfaces must be exercised

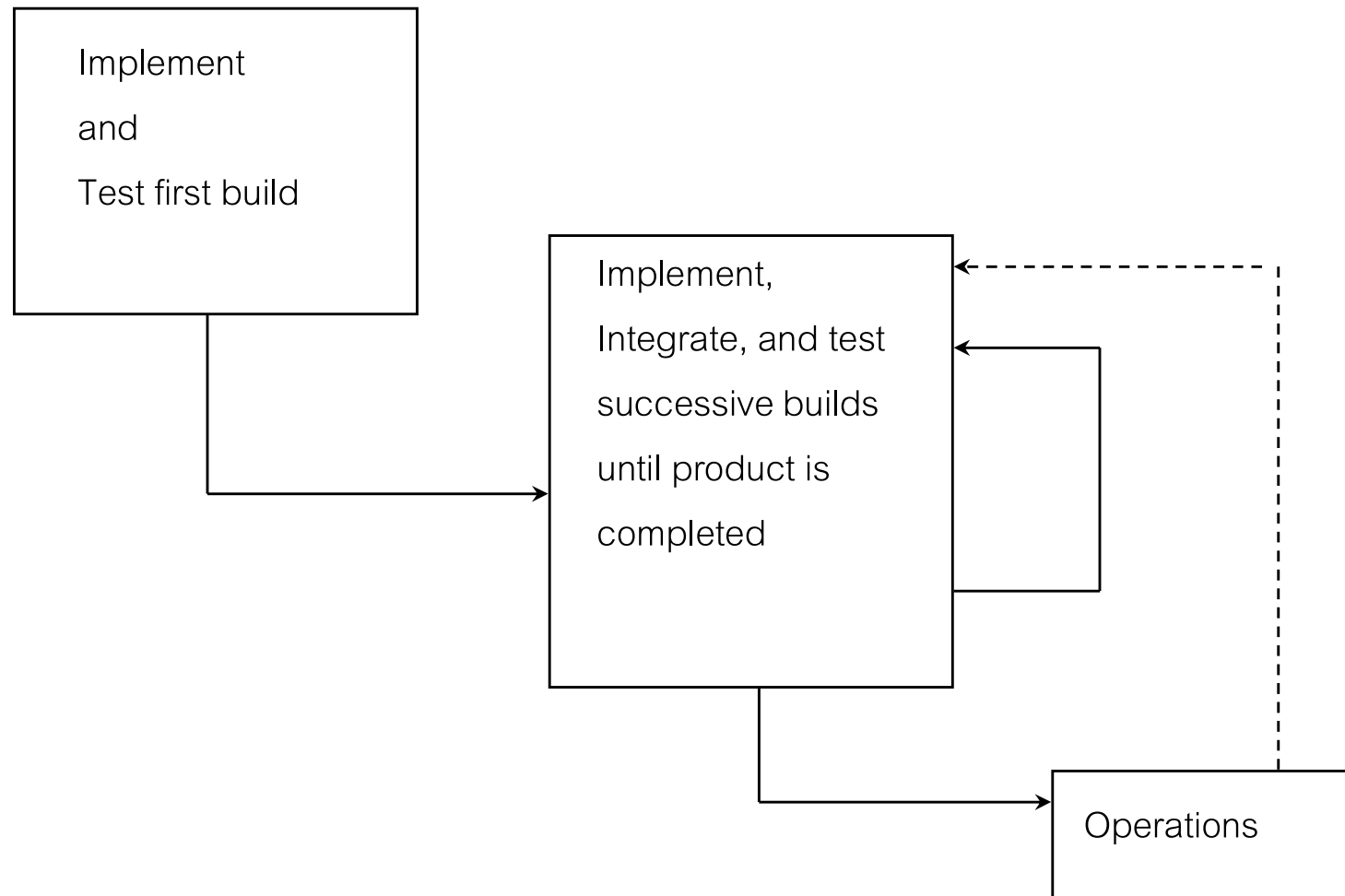
# The Incremental Model

- wait for a long time for software systems to be ready
- one way **to reduce cycle time** is to use phased development
- the system is designed so that it can be delivered in pieces
- enable the users to have some functionality while the rest is being developed
- the system **is partitioned into subsystems by functionality**
- the releases are defined by beginning with one small function
- then **adding functionality** with each new release

# The Incremental Model

- phased development is desirable because
  - **training** can begin on an early release
  - **markets can be created early** for functionality that has never before been offered
  - **frequent release** allow developers to fix unanticipated problem quickly

# The Incremental Model



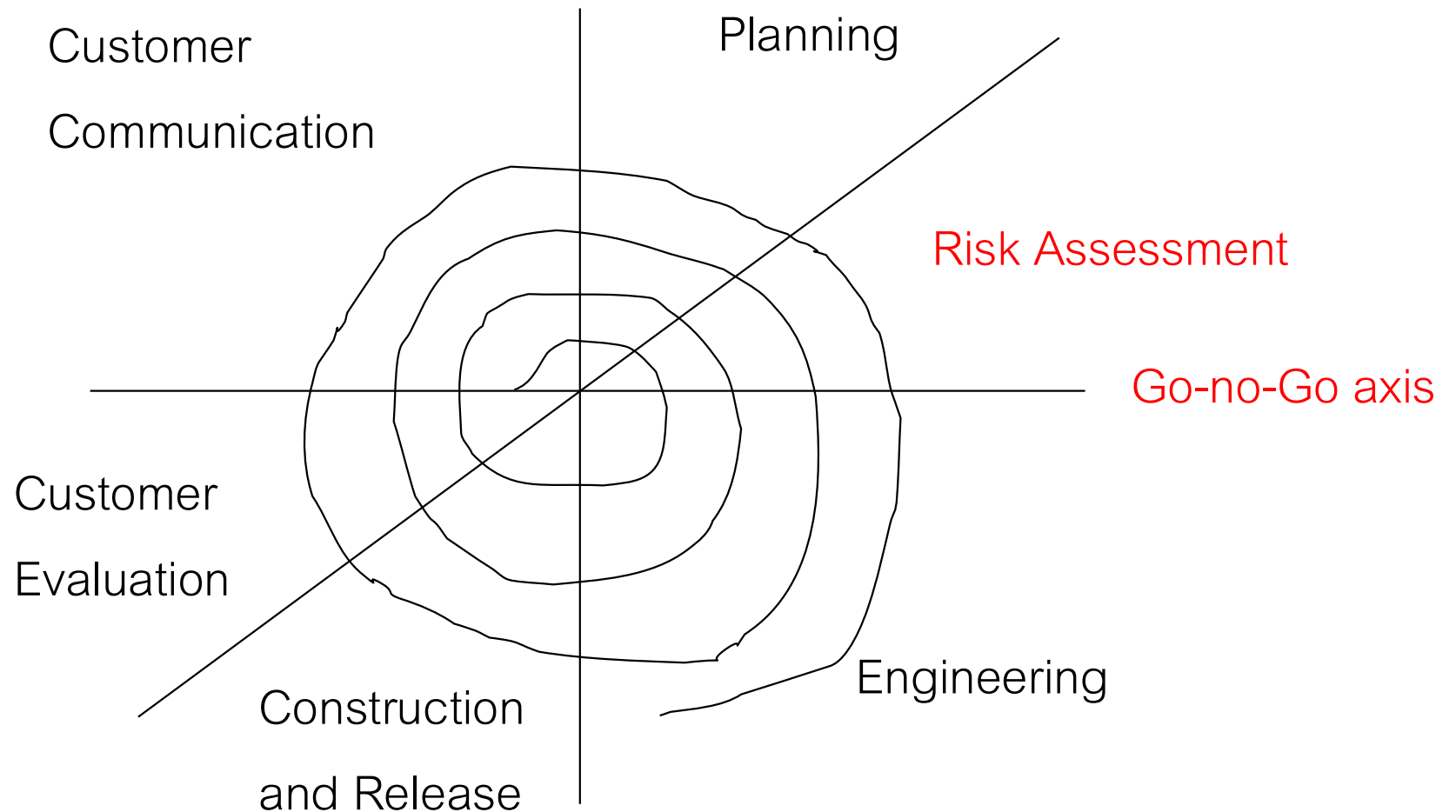
# The Spiral Model

- was developed by Boehm in 1988
- combine development activities with risk management to minimize and control risk
- is like iterative development
- begins with requirements and initial plan (including budget, constraints, and alliterative staffing)
- next, evaluate risks and prototype alternatives
- next, produce a “concept of operations” document that describes at a high level how the system works
- specify set of requirements

## The Spiral Model (cont.)

- iteration products are concept of operations document, requirements, design, and testing
- with each iteration
  - do risks analysis
  - develop prototype to verify feasibility or desirability
- when risks are identified, the project managers must decide how to eliminate or minimize the risk

# The Spiral Model

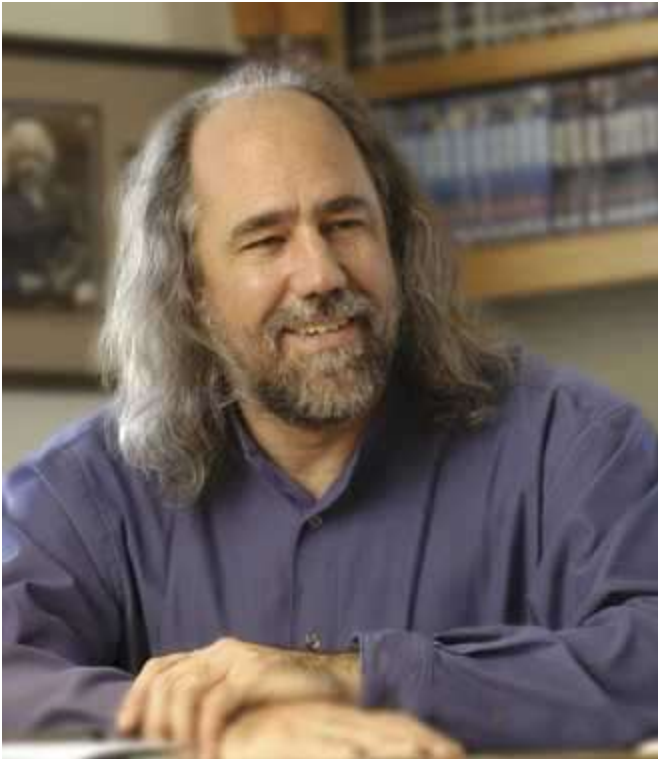




# UML - A Brief History

- During the **1980s** and the early **1990s**, OO methods and programming languages gained a widespread audience
- Many OOA methods were proposed but no individual method dominated the software engineering view
- during the **early 1990s** James Rumbaugh, Grady Booch, and Ivar Jacobson began working on a “**unified method**” that would combined best features of other methods

# Grady Booch



- **Grady Booch** (born February 27, 1955) is an American software engineer, and Chief Scientist, Software Engineering in IBM Research. Booch is best known for developing the Unified Modeling Language with Ivar Jacobson and James Rumbaugh.
- Booch is best known for developing the Unified Modeling Language with Ivar Jacobson and James Rumbaugh
- The Booch method is a technique used in software engineering. It is object modeling language and methodology that was widely used in object-oriented analysis and design. It was developed by Booch while at Rational Software.

# Ivar Jacobson



- **Ivar Jacobson** (born 1939) is a Swedish computer scientist, known as major contributor to UML, Objectory, RUP and aspect-oriented software development.

# James Rumbaugh



- **James Rumbaugh** (born 1947) is an American computer scientist and object methodologist who is best known for his work in creating the Object Modeling Technique (OMT) and the Unified Modeling Language (UML)


## UML - A Brief History (cont.)

- The result of their work was UML (Unified Modeling Language)
- by **1997**, UML became an industry standard for object-oriented software development
- UML does not provide the process framework to guide project teams in their application of the technology
- Thus, the Unified Process was developed
- Focus on reliance to future changes and reuse
- Its process flow is iterative and incremental

UML 2.5x (in 2023)

# Official version: UML 2.5.1

[HOME](#) [SITE MAP](#) [LEGAL](#) [f](#) [t](#) [in](#) [✎](#) [v](#)

 Standards Development Organization

[ABOUT ▾](#) [CERTIFICATIONS ▾](#) [RESOURCES ▾](#) [SPECIFICATIONS ▾](#) [MEMBERSHIP ▾](#)

ABOUT THE UNIFIED MODELING LANGUAGE SPECIFICATION VERSION 2.5.1

2.5.1 • UML • SPECIFICATIONS

UML®

## Unified Modeling Language

A specification defining a graphical language for visualizing, specifying, constructing, and documenting the artifacts of distributed object systems.

**Title:** Unified Modeling Language  
**Acronym:** UML®  
**Version:** 2.5.1  
**Document Status:** formal ⓘ  
**Publication Date:** December 2017  
**Categories:** [Modeling](#) [Software Engineering](#) [Platform](#)  
[IPR Mode ⓘ](#) [RF-Limited ⓘ](#)



# Agile Development

- Watch VDO
- มี Quiz ครึ่งหน้า

