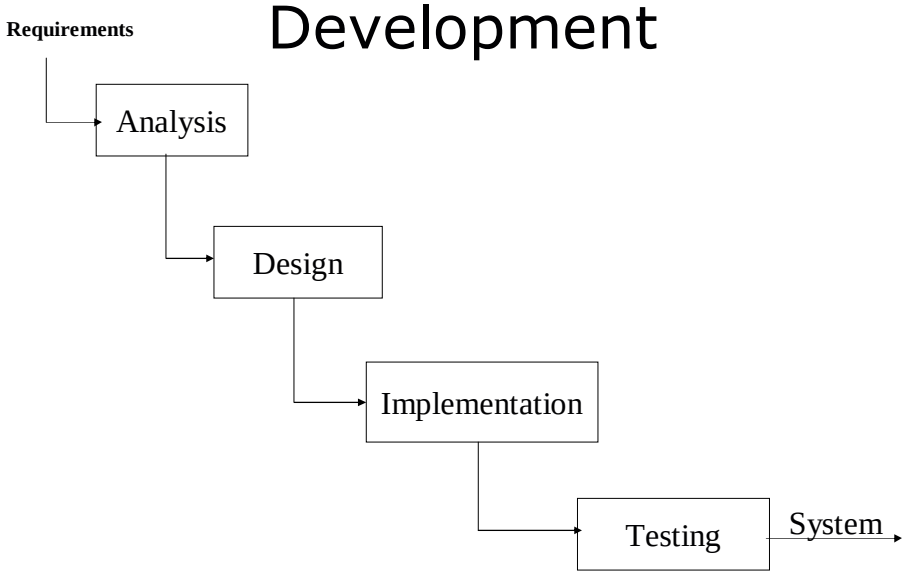
1. Software Development Process

A Process

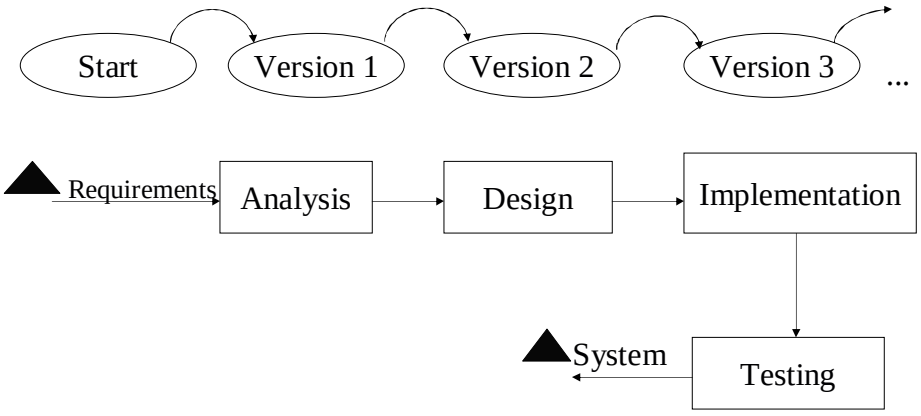
- Unified Process: Component Based Development
  - Components for implementation
  - Interfaces for interconnection

Activities in System Development



Versions of a Project

- Full requirements are not given at start
- Full requirements are not known
- Full development may take several years

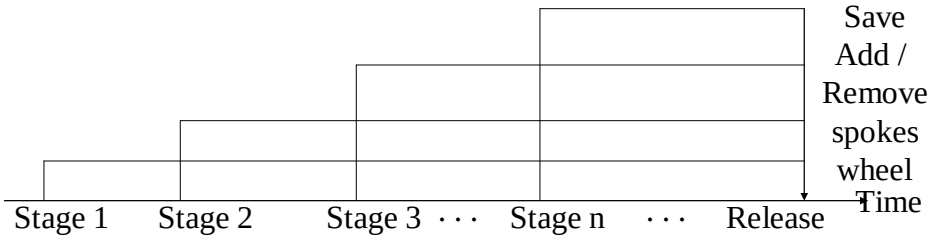


Incremental Development

Reasons:

- Requirements are not very clear
- Requirement changes in the mean time
- Feedback will help "steer" the project
- Feedback helps new requirements

Each stage adds new functionality



Prototyping

- Illustrates the working of system
- Focuses on properties requiring insight
- Allows experimentation
- "Seeing it" promotes opinion & clarification
- Can play with it to understand system dynamics
- Not a product - just a demo

Rapid Prototyping : quick functioning model

Build it - See it - Dump it

Reuse

- Put together set of available components to develop system
- Components are finest level of granularity for reuse

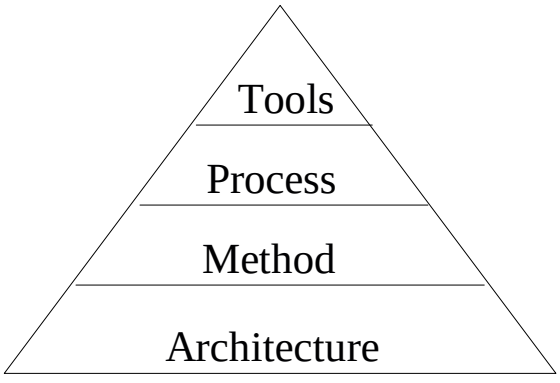
Components:

- Units that provide some functionality
- Powerful and useful
- Well defined interfaces
- Simple to find, understand & use
- Wide application
- Can be used to build other components

Flexible to change

- Alteration must be to single component / module
- Cost effective

Methodology



Architecture, Method, Process & Tool

Architecture:

- Types of model that can be built and characteristic of each model
- What makes the components?

Method:

- Sequence of Steps to realize the goal
- Based on the underlying architecture
- Simplifies development for a given architecture

Architecture, Method, Process & Tool

Process:

- Scaling up of the Method
- Industrialization of a Method
- Activities for the entire lifetime of a product
- Must be specialized into other processes

Tools:

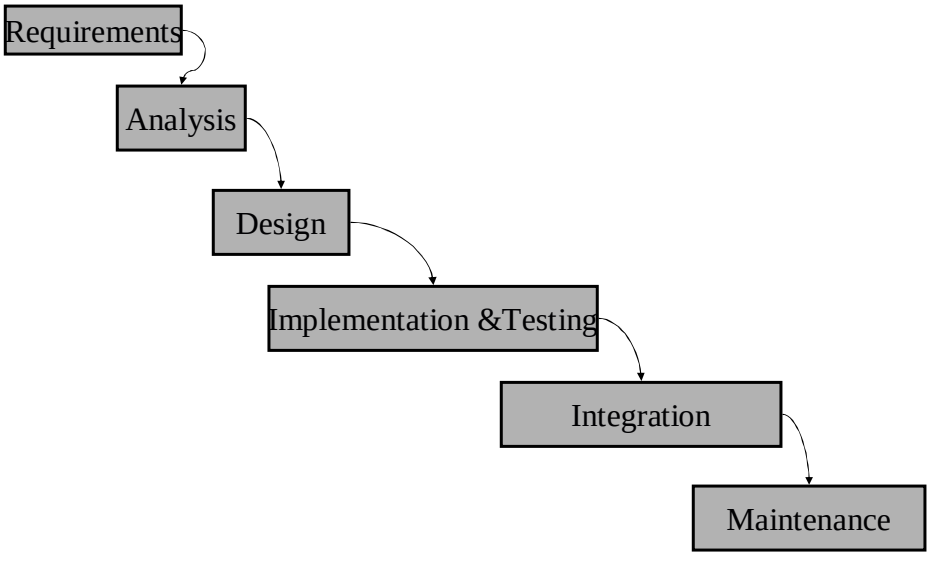
- Computer - aided Systems Engineering (CASE)
- Provides automation for documentation
- Effective to communicate
- Easy to flow from one activity to another
- Improves productivity

Development Approach

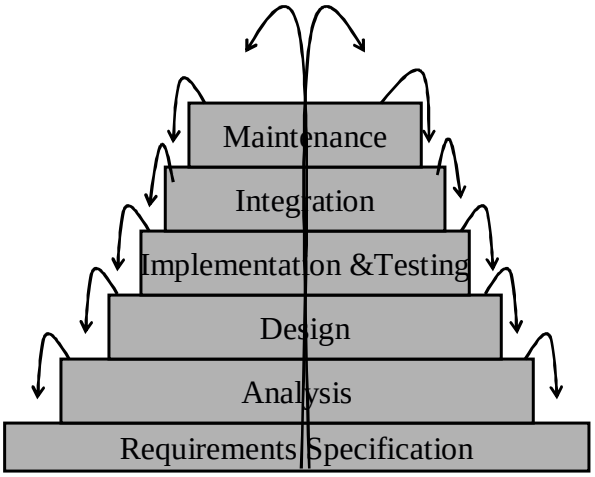
- Waterfall Method
- Spiral Method
- Incremental and Iterative Development

Need a Mature Repeatable Development Process

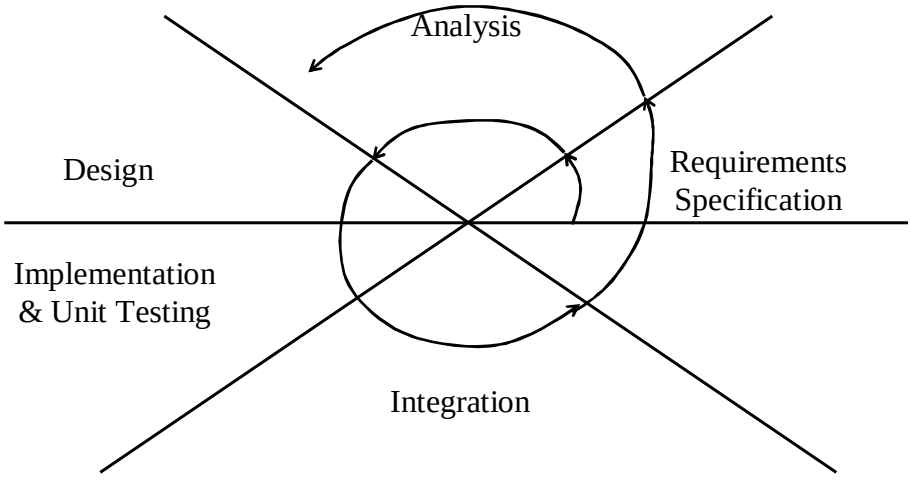
Waterfall Method



Fountain Method



Spiral Method



Iterative and Incremental Life Cycle

- Neither top-down nor bottom-up
- Successive Refinement of the OO Architecture
  - Apply Experience & Results to next iteration

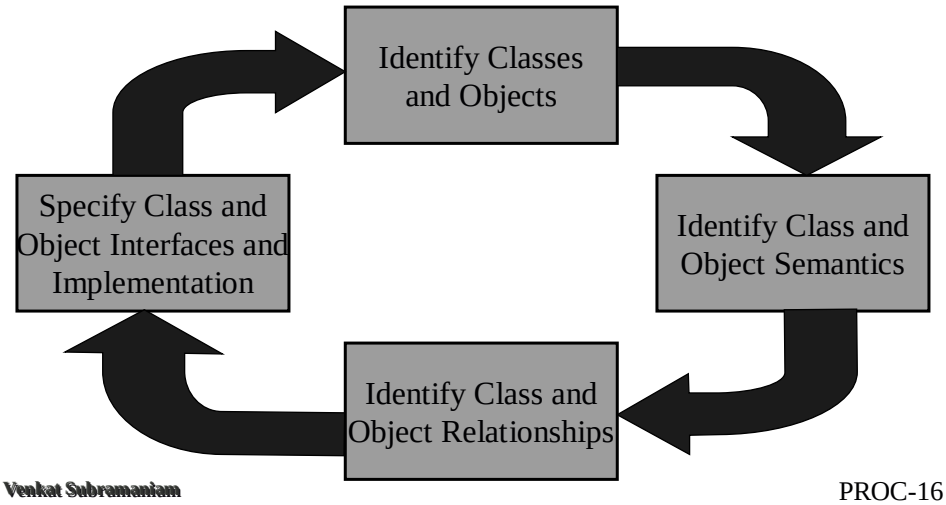
Booch's Approach:

Micro Development Process (Spiral)

Macro Development Process (Waterfall)

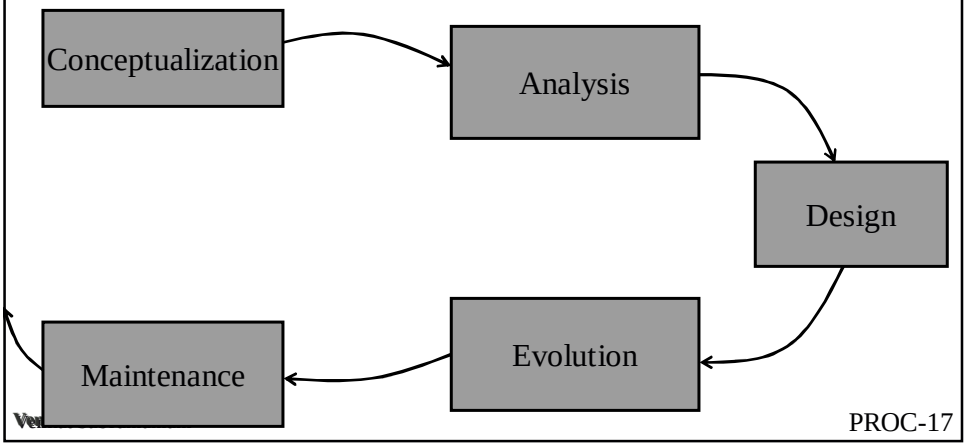
Micro Development Process

Represents Daily Activity of Developers



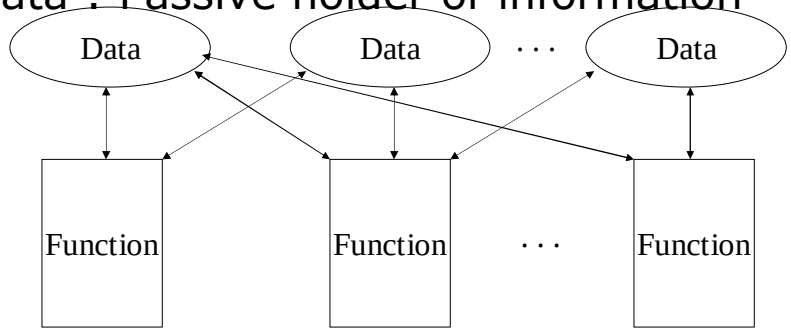
Macro Development Process

Controlling Framework for Micro Process
Helps Assess Risk and Make Early Corrections
Activities on the Scale of Weeks to Months



Function / data Methods

- Treat Functions & data as being separate
- Structured Analysis and Design Technique
- Functions : Active, describes behavior
- Data : Passive holder of information



Function / data Methods : Problems

- Difficult to maintain
- Hard to handle different data format
  - spilled with Switch-CASE statements
- Poor readability
- Change to data structure results in change to several functions
- Unstable - small modifications result in major changes

Object - Oriented Development

- Functions and data highly integrated
- More understandable
- More flexible to modifications
- Stable - modifications are localized

Object - Oriented Analysis

- Activities:
  - Finding the objects
  - Organizing the objects
  - Describing object interaction
  - Defining operations on the objects
  - Defining the object's internals
- Obtain understanding of system based on functional requirements
- Considers functional requirements & data as integrated objects

OOA Activities

- Finding the Objects
  - Nouns in problem statement
  - Easy to find objects
    - difficult to know what is relevant
  - Find ***essential*** objects
- Organizing the Objects
  - Exploring similarity between classes of objects
  - Organizing Classes into hierarchy
    - Inheritance
    - Whole / part
    - Association

OOA Activities continued

- Object Interaction
  - Scenarios or use cases provide insight - which objects work together
  - Helps to define Object interface
- Operations on Objects
  - Simple primitive operations based on the interface
  - Avoid creating complex objects
- Object Implementation / Internals
  - Information to be held, alternate ways of storing
  - Number of instances of the object
  - Attributes may be inherited

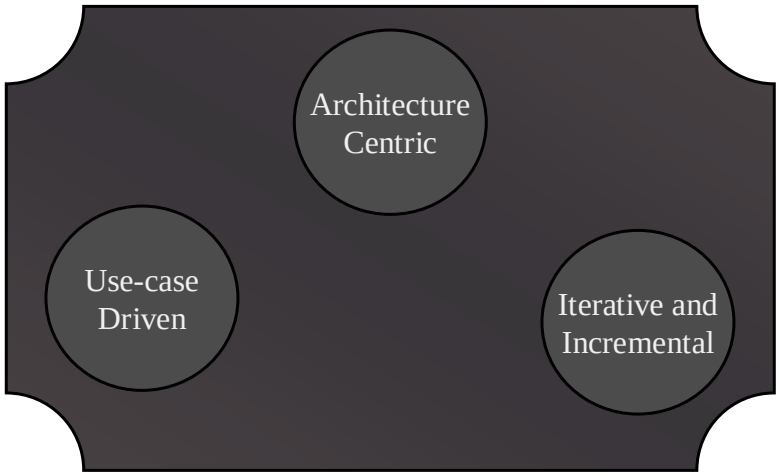
OO Design & Implementation

- Analysis model Designed & Implemented
- Take care of restrictive demands
  - memory, response time, etc.
- Traceability of Objects
- Create & use Components

Object - Oriented Testing

- Integration is not very painful
- Help broadening unit testing to larger units
- Integration testing comes in earlier stages
- Inheritance poses a bit of challenge in testing

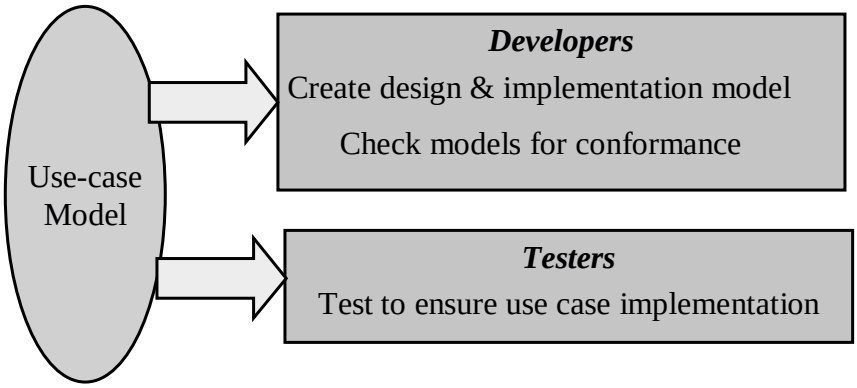
The Unified Process



Use-Case Driven

- Remember the User?!
  - What does the user want and need from the system
- Use-case
  - interaction by the user and the system's response
  - captures functional requirements
- Use-case Model
  - All use-cases make up complete functional model
    - What does the system do for each user?
- What do you do with it?

Use-case Model



Drives the development process